MOTOROLA

Embedded SDK (Software Development Kit)

Targeting Motorola 56F800 Demonstration Board

SDK157/D
Rev. 1, 03/24/2003

MOTOROLA intelligence everywhere™ | digital dna™

Contents

List of Tables

List of Figures

About This Book

This manual describes the applications for the 56F800 Demonstration Board.

Audience

This document targets software developers using the 56F800 Demonstration Board.

Organization

- **Chapter 1, Introduction**—provides a brief overview of this document
- **Chapter 2, Directory Structure**—provides a description of the required core directories
- **Chapter 3, 56F800 Demonstration Board Applications**—describes the available demonstrations for the 56F800 Demonstration Board

Suggested Reading

We recommend that you have a copy of the following references:

- *Targeting Motorola DSP56F80x Platform Manual,* SDK126/D
- *568F00 Demonstration Board User's Manual,* TBD
- *DSP56800 Family Manual,* DSP56800FM/AD
- *DSP56F80x User's Manual*, DSP56F801-7UM/AD
- *Inside CodeWarrior: Core Tools*, Metrowerks Corp.

Conventions

This document uses the following notational conventions:

Typeface, Symbol or Term	Meaning	Examples
`Courier Monospaced Type`	Code examples	`//Process command for line flash`
Italic	Directory names, project names, calls, functions, statements, procedures, routines, arguments, file names, applications, variables, directives, code snippets in text	...and contains these core directories: *applications* contains applications software... ...CodeWarrior project, *3des.mcp* is... ...the *pConfig* argument.... ...defined in the C header file, *aec.h*....
Bold	Reference sources, paths, emphasis	...refer to the **Targeting DSP56F80x Platform** manual.... ...see: **C:\Program Files\Motorola\Embedded SDK\help\tutorials**
Blue Text	Linkable on-line	...refer to Chapter 7, License....
Number	Any number is considered a positive value, unless preceded by a minus symbol to signify a negative value	3V -10 DES^{-1}
ALL CAPITAL LETTERS	# defines/ defined constants	# define INCLUDE_STACK_CHECK
Brackets [...]	Function keys	...by pressing function key [F7]
Quotation marks, "..."	Returned messages	...the message, "Test Passed" is displayed.... ...if unsuccessful for any reason, it will return "NULL"...

Definitions, Acronyms, and Abbreviations

The following list defines the acronyms and abbreviations used in this document. As this template develops, this list will be generated from the document. As we develop more group resources, these acronyms will be easily defined from a common acronym dictionary. Please note that while the acronyms are in solid caps, terms in the definition should be initial capped ONLY IF they are trademarked names or proper nouns.

ADC	Analog-to-Digital Converter
COP	Computer Operating Properly
DSP	Digital Signal Processor or Digital Signal Processing
FFT	Fast Fourier Transform
GPIO	General Purpose Input/Output
ISR	Interrupt Service Request
POT	Potentiometer
PWM	Pulse Width Modulator
SDK	Software Development Kit

References

The following sources were used to produce this book:

1. *Targeting Motorola DSP56F80x Platform Manual,* SDK126/D
2. *56F800 Demonstration Board User's Manual,* TBD
3. *DSP56800 Family Manual,* DSP56800FM/AD
4. *DSP56F80x User's Manual*, DSP56F801-7UM/AD
5. *Embedded SDK Programmer's Guide,* SDK 101/D

Chapter 1
Introduction

1.1 Overview

The 56F800 Demonstration Board is a low-cost board that allows a user to execute preprogrammed demonstrations, as well as to develop his own applications using free CodeWarrior tools. The 56F800 Demonstration Board consists of a 60 MIP 56F801 hybrid controller, a microphone attached to the ADC, a potentiometer (POT) attached to the ADC, a button attached to an external interrupt (IRQ) and 10 LEDs. Pads have also been included on the board so a user can access all of the 56F801's peripherals. The Demonstration Board does not have an external crystal, so the 56F801 must use its internal oscillator.

For more information about developing software for this demonstration board, please refer to the sections about the 56F801 in the **Targeting Motorola DSP56F80x Platform** manual. This document describes the SDK's on-chip drivers, off-chip drivers, and libraries that are available for the 56F801. It also describes how to get started with CodeWarrior and the SDK. For more specific 56F801 information, refer to the **DSP56F80x User's Manual**.

Chapter 2
Directory Structure

2.1 56F800 Demonstration Board Directory Structure

Figure 2-1 illustrates the directory structure for the 56F800 Demonstration Board.

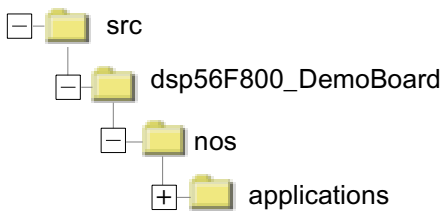

Figure 2-1. 56F800 Demonstration Board Directories

In the 56F800 Demonstration Board directory structure, only demo projects are located in the *applications* directory. All other files are located in */src/dsp56801evm*. Please refer to the **Targeting Motorola DSP56F80x Platform** manual for a description of these files.

Chapter 3
56F800 Demonstration Board Applications

3.1 Applications

The following applications have been provided by Motorola to easily demonstrate some of the features of the 56800 family of hybrid controllers.

3.1.1 Quad Timer Flashing LEDs Demonstration

The Quad Timer Flashing LEDs Demonstration illustrates use of five channels of the Quad Timer. The first timer goes off on an overflow, which starts the second timer. The second timer goes off on Compare 1, which starts the third timer. The third timer goes off on Compare 2, which starts the third timer, the fourth timer goes off on Compare 3, which starts the fifth timer. The fifth timer goes off on Compare 4, which starts the first timer. Each column of LEDs reflects the status of the corresponding timer, and as a result, the LEDs on the running application are also lighted continuously.

3.1.1.1 Set-up for Quad Timer Flashing LEDs Demonstration

Demonstration Board Jumper Settings:

Use default settings as shown in the **56F800 Demonstration Board User's Manual**.

3.1.1.2 Procedure for Quad Timer Flashing LEDs Demonstration

- Using CodeWarrior, open:
  ...\nos\applications\quadtimer\quadtimer.mcp
- Build and download the project; when the debugger reaches *main()*, it will stop
- Select *Run* in the debugger to continue executing the test
- At this point, the five columns of LEDs will turn on and off sequentially

3.1.2 Timer Driver Flashing LEDs Demonstration

This demonstration exercises Timer0, Timer1, Timer2, Timer3, and Timer4 peripherals located in the 56F801 processor. It initializes Timer1 using clock ID CLOCK_AUX1 to 1 second with a reload value of 1 second. Timer2 is initialized using clock ID CLOCK_AUX2 to 0.5 seconds with a reload value of 0.5 seconds. Timer3 is initialized using clock ID CLOCK_AUX3 to 0.25 seconds with a reload value of 0.25 seconds. Timer 4 is initialzed using clock ID CLOCK_AUX4 to 0.125 seconds with a reload value of 0.125 seconds. When these timers expire, the application changes the on/off state of the LEDs. Figure 3-1 shows the net effect of these activities.

For interaction with Timer0, the application uses the SDK timer services. The SDK timer services reserves Timer0 with clock ID CLOCK_REALTIME, and utilizes this timer for *nanosleep* functionality. This application calls the *nanosleep* interface with a time-out value of 0.5 seconds in a tight loop. The net effect of this call is suspension of the *main* function execution for 0.5 seconds.

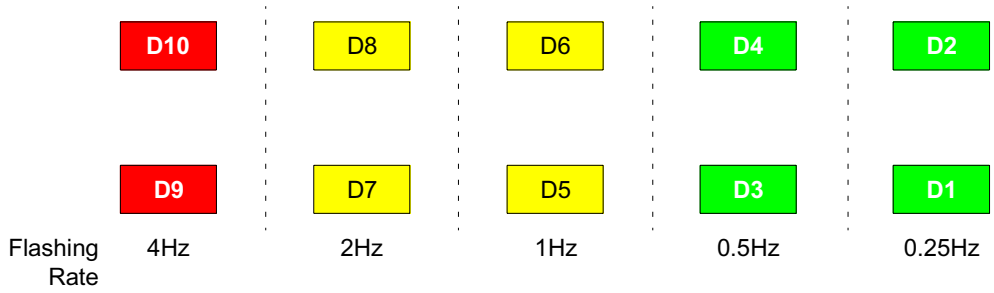

Figure 3-1. LEDs in the Timer Driver Demonstration

3.1.2.1 Set-up for Timer Driver Flashing LEDs Demonstration

Demonstration Board Jumper Settings:

Use default settings as shown in the **56F800 Demonstration Board User's Manual**.

3.1.2.2 Procedure for Timer Driver Flashing LEDs Demonstration

- Using CodeWarrior, open:
  ...\nos\applications\timers\timers.mcp
- Build and download the project; when the debugger reaches *main()*, it will stop
- Select *Run* in the debugger to continue executing the demonstration
- Once the application is running, the application changes the on/off state of the LEDs

3.1.3 Frequency Spectrum with Amplitude Demonstration

This demonstration exercises the ADC, PWM, and GPIO in the 56F801 processor. It samples the ADC at a rate of 8000 samples/second. The demonstration then performs an FFT on the sampled data to determine the frequency content of the analog signal. Finally, it uses the 10 LEDs, which are connected to the PWM and GPIO, to display the frequency content and amplitude.

The first column of LEDs refers to the DC component of the audio signal. The second, third, fourth, and fifth columns of LEDs refer to, respectively, the following frequency components: 1kHz, 2kHz, 3kHz, and 4kHz.

For an illustration of the LEDs during this demonstration, see **Figure 3-2**.

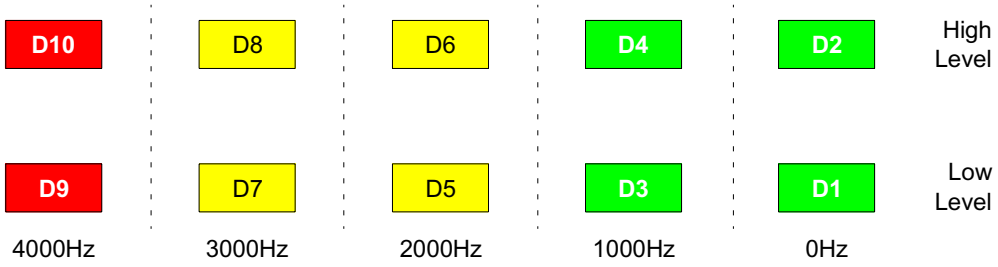

Figure 3-2. LEDs in the Frequency Spectrum with Amplitude Demonstration

3.1.3.1 Set-up for Frequency Spectrum with Amplitude Demonstration

Demonstration Board Jumper Settings:

Use default settings as shown in the **56F800 Demonstration Board User's Manual**.

3.1.3.2 Procedure for Frequency with Amplitude Spectrum Demonstration

- Using CodeWarrior, open:
  ...\nos\applications\FreqSpectrum\FreqSpectrum.mcp
- Open *main.c* and define DEMO_BOARD_WITH_AMPLITUDE to be 1
- Build and download the project; when the debugger reaches *main()*, it will stop
- Select *Run* in the debugger to continue executing the demonstration
- Input audio signals (less than 4kHz) into the microphone
- The LEDs should turn on/off, depending on the frequencies contained in the audio signal
- Pausing the demonstration
  - — While the demonstration is executing, the LEDs will flash
  - — If the IRQ button is pushed, the demonstration will pause and the LEDs will indicate the demonstration's state at the time it was paused
  - — The demonstration will continue when the IRQ button is pressed again

3.1.4 Frequency Spectrum without Amplitude Demonstration

This demonstration exercises the ADC, PWM, and GPIO in the 56801 processor. It samples the ADC at a rate of 8000 samples/second. The demonstration then performs an FFT on the sampled data to determine the frequency content of the analog signal. Finally, it uses the 10 LEDs, which are connected to the PWM and GPIO, to display the frequency content.

The first LED in the top row refers to the DC component of the audio signal. The second through fifth LEDs in the top row refer to, respectively, the following frequency components: 500Hz, 1.0kHz, 1.5kHz, 2.0kHz. The first through fourth LEDs in the bottom row refer to, respectively, the following frequency components: 2.5kHz, 3.0kHz, 3.5kHz, and 4.0kHz.

Figure 3-3 shows LEDs during the Frequency Spectrum without Amplitude demonstration.

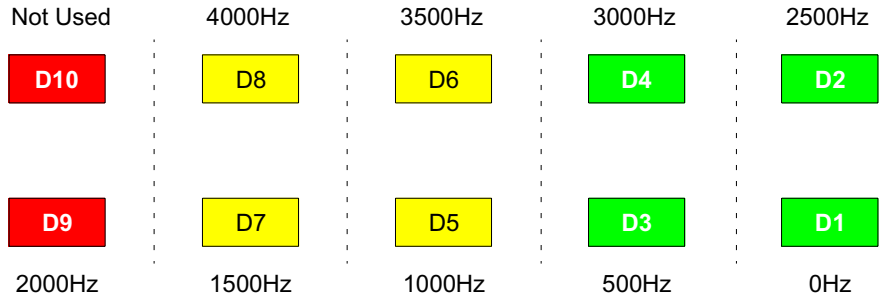

Figure 3-3. LEDs in the Frequency Spectrum without Amplitude Demonstration

3.1.4.1 Set-up for Frequency Spectrum without Amplitude Demonstration

Demonstration Board Jumper Settings:

Use default settings as shown in the **56F800 Demonstration Board User's Manual**.

3.1.4.2 Procedure for Frequency without Amplitude Spectrum Demonstration

- Using CodeWarrior, open:
  ...\nos\applications\FreqSpectrum\FreqSpectrum.mcp
- Open *main.c* and define DEMO_BOARD_WITH _AMPLITUDE to be 0
- Build and download the project; when the debugger reaches *main()*, it will stop
- Select *Run* in the debugger to continue executing the demonstration
- Input audio signals (less than 4kHz) into the microphone
- The LEDs should turn on/off, depending on the frequencies contained in the audio signal
- Pausing the demonstration
  - — While the demonstration is executing, the LEDs will flash
  - — If the IRQ button is pushed, the demonstration will pause and the LEDs will indicate the demonstration's state at the time it was paused
  - — The demonstration will continue when the IRQ button is pressed again

3.1.5 Frequency Detector Demonstration

This demonstration exercises the ADC, PWM, and GPIO in the 56F801 processor. It samples the ADC at a rate of 8000 samples/second. The demonstration then performs an FFT on the sampled data to determine the frequency content of the analog signal. Finally, it uses the 10 LEDs, which

are connected to the PWM and GPIO, to display the binary representation of the strongest frequency detected.

See **Figure 3-4**, which illustrates LEDs during the Frequency Detector demonstration.

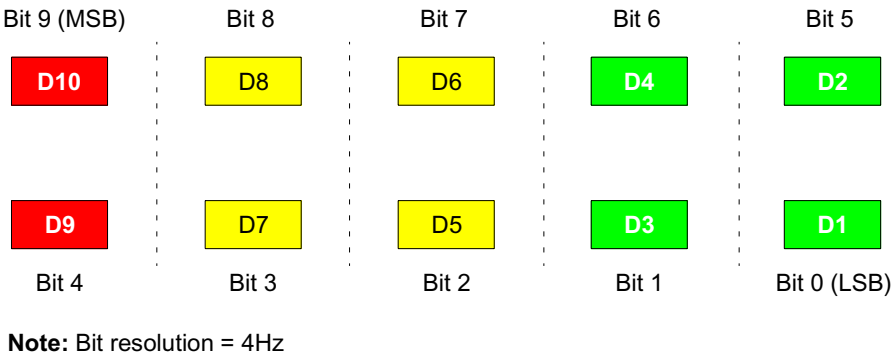

Figure 3-4. LEDs during the Frequency Detector Demonstration

3.1.5.1 Set-up for Frequency Detector Demonstration

Demonstration Board Jumper Settings:

Use default settings as shown in the **56F800 Demonstration Board User's Manual**.

3.1.5.2 Procedure for Frequency Detector Demonstration

- Using CodeWarrior, open:
  ...\nos\applications\FreqDetector\FreqDetector.mcp
- Build and download the project; when the debugger reaches *main()*, it will stop
- Select *Run* in the debugger to continue executing the demonstration
- Input audio signals (less than 4kHz) into the microphone
- The LEDs should indicate the binary representation of the strongest frequency detected
- Pausing the demonstration
  - — While the demonstration is executing, the LEDs will flash
  - — If the IRQ button is pushed, the demonstration will pause and the LEDs will indicate the demonstration's state at the time it was paused
  - — The demonstration will continue when the IRQ button is pressed again

3.1.6 Potentiometer-Controlled LEDs Demonstration

This demonstration exercises the ADC, PWM, and GPIO in the 56F801 processor. A voltage divider that is controlled by a Potentiometer (POT) is attached to AN6 of the ADC. This demonstration continuously samples the ADC and uses the 10 LEDs, which are connected to the PWM and GPIO, to display the amplitude of the signal.

Since the voltage range is 0 to 3V, each LED represents an increment of 0.3V.

Figure 3-5 shows the LEDs during the POT-Controlled LEDs demonstration.

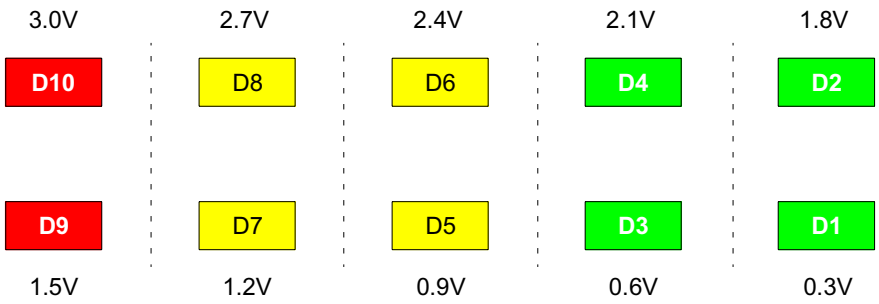

Figure 3-5. LEDs in the POT-Controlled LEDs Demonstration

3.1.6.1 Set-up for POT-Controlled LEDs Demonstration

Demonstration Board Jumper Settings:

Use default settings as shown in the **56F800 Demonstration Board User's Manual**.

3.1.6.2 Procedure for POT-Controlled LEDs Demonstration

- Using CodeWarrior, open:
  ...\nos\applications\POTControlledLEDs\POTControlledLEDs.mcp
- Build and download the project; when the debugger reaches *main()*, it will stop
- Select *Run* in the debugger to continue executing the demonstration
- Adjust the POT to vary the voltage into AN6
- The LEDs will indicate the amplitude of input signal with a resolution of 0.3V

3.1.7 Serial Bootloader

The Serial Bootloader has been developed to load and run a proprietary user application presented as an S-Record file into the Program and Data memory. The Serial Bootloader is located in the dedicated Program memory region, called Boot Flash. The Serial Bootloader supports the simplest serial protocol, so a standard serial terminal program can be used on the host PC.

The Serial Bootloader application reads the S-Record file of a user application (for example, generated by CodeWarrior) via serial interface, parses this S-Record file, and stores needed data in Program and Data Flash memory. When the processing of the S-Record file is finished, the Bootloader launches the loaded application. If any error occurs while loading the S-Record file, the Bootloader outputs an error message with an error number via the serial line and resets the processor.

3.1.8 Files

The Serial Bootloader application includes the following files:

- **..\nos\applications\serial_bootloader\bootloader.mcp**, project file

- **..\nos\applications\serial_bootloader\bootloader.c**, main program
- **..\nos\applications\serial_bootloader\bootloader.h**, header file with common parameters
- **..\nos\applications\serial_bootloader\com.c**, communication module
- **..\nos\applications\serial_bootloader\com.h**, header for communication module
- **..\nos\applications\serial_bootloader\sparser.c**, S-Record format parser module
- **..\nos\applications\serial_bootloader\sparser.h**, header for S-Record parser module
- **..\nos\applications\serial_bootloader\prog.c**, Flash programming module
- **..\nos\applications\serial_bootloader\prog.h**, header for Flash programming module
- **..\nos\applications\serial_bootloader\bootstart.c**, start-up module
- **..\nos\applications\serial_bootloader\constdata.asm**, description of strings data
- **..\nos\applications\serial_bootloader\resetvector.asm**, Reset and COP Reset interrupt vectors description
- **..\nos\applications\serial_bootloader\TargetDirectives**, Board name definition
- **..\nos\applications\serial_bootloader\config\linker.cm**d, linker command file used for Boot Flash
- **..\nos\applications\serial_bootloader\config\flash.cfg**, Metrowerks CodeWarrior configuration file to work with Flash

These files are located in the SDK installation directory. The Bootloader application uses only SDK files *port.h*, *arch.h* and *periph.h* from the **..\nos\include** directory. It can be compiled and used without other parts of the SDK.

3.1.9 Target Configuration

The Bootloader uses a minimum of the hybrid controller's resources in order to reduce the possibility of conflict with the application being downloaded. Specifically, it uses the 56F801's SCI 0, Port E, and PLL peripherals, as well as internal RAM for data buffering. The SCI baud rate value is calculated statically with the assumption that the oscillator frequency is 8MHz. The 56F801's operational frequency is set to 60MHz. All peripherals used by the Bootloader are returned to their initial state prior to the execution of the application programmed.

The Bootloader application relies on the *TargetDirectives* file to define the target for which the application will be built. When this file contains #define DSP56801EVM and #define DSP56F800_DEMO_BOARD, this application is built assuming the existence of an internal relaxation oscillator and corresponding calibration value. At run time, the application automatically calibrates the internal relaxation oscillator to run at 8MHz.

3.1.9.1 Set-up

To use the Bootloader:

- Populate the Demonstration Board with the RS-232 circuitry (only pads are provided on the board)
- Program the Bootloader into Boot Flash
- Using a serial cable, connect the Demonstration Board's RS-232 connector with the host PC's COM serial port
- Remove the Host Enable jumper (JP1) to allow the board to run as a stand-alone without a debugger

3.1.9.1.1 Demonstration Board Jumper Settings

To load the Bootloader into the 56F800 Demonstration Board, the following jumper settings are required:

- Set jumper JP1 (Host Enable)

To start a previously loaded Bootloader on the 56F801 board, the following jumper settings are needed:

- Do not set jumper JP1 (Host Enable)

3.1.9.2 Build

To build the Serial Bootloader, open the *bootloader.mcp* project file in the CodeWarrior IDE and execute the *Project/Make* command. This will build and link the Serial Bootloader application.

3.1.9.2.1 Download into Boot Flash

To download, build the Bootloader from CodeWarrior and load it into the board by choosing the *Project/Debug* command in the CodeWarrior IDE. Make sure that jumpers are set for loading as described in Section 3.1.9.1.1.

3.1.9.2.2 Host Terminal Program Set-up

A host terminal program is used to communicate with Bootloader. The following description assumes that Microsoft Windows HyperTerminal program is used. To configure Microsoft HyperTerminal to communicate with Bootloader:

- Start HyperTerminal
- Create a new connection
- Select the COM port previously connected to EVM
- Set:

Baud rate	115200 bps
8N1	8 data bits, no parity, 1 stop bit character format
Flow control protocol	Xon/Xoff

- HyperTerminal can now display Bootloader messages

3.1.9.3 Execute

To execute the Serial Bootloader application after loading it into Flash, set jumpers as described in Section 3.1.9.1.1. and push the RESET button.

If the terminal program is set up properly and the EVM and the Host PC are properly connected, the terminal program will display a Bootloader start-up message:

```
"(c) 2000-2001 Motorola Inc. S-Record loader. Version 1.3"
```

To load the S-Record file, select the *Transfer/Send text file* from the HyperTerminal menu and select a file. When the S-Record file is loaded and the application is started, the terminal window displays a message similar to this:

```
"(c) 2000-2001 Motorola Inc. S-Record loader. Version 1.3

Loaded 0x044d Program and 0x000a Data words.
Application started."
```

The download rate is about 7660 bytes of S-Record file per second loaded from the terminal, or about 1740 words of Program or Data memory stored into Flash per second.

If an error is detected while loading the S-Record file, the Bootloader displays the error message and resets the processor. For example, if an S-record file contains a character that is not permitted for S-Records, the following message is displayed:

```
"(c) 2000-2001 Motorola Inc. S-Record loader. Version 1.3

Error # 0002
Resetting the processor."
```

After this message, the Bootloader resets the processor and waits for the S-Record file again. Other loading errors are described in **Table 3-1**.

Table 3-1. Error Codes for the Serial Bootloader Application

Error Code	Error Title	Possible Reasons	What to Do
1	Data Receive Error	• Any SCI error except Noise Error (Overrun, Frame Error, Parity Error)	• Check connections with Host PC and settings for host terminal program
2	Invalid Character	• Received character is not "S" or any hexadecimal digit	• Verify that S-Record file does not contain any invalid characters • Check connections and send mode in the terminal program
3	Invalid S-Record Format	• Invalid record type; permitted type is 0,3,7 • S-Record length is less than address plus checksum length	• Verify S-Record file
4	Wrong S-Record Checksum	• Calculated Checksum of the received S-Record did not match the received Checksum in the S-Record	• Check S-Record file • Check connections and send mode in terminal program
5	Buffer Overrun	• Internal data buffer was full	• Terminal program did not stop after receiving Xoff character • Confirm that the terminal program supports Xon/Xoff flow control protocol
6	Flash Programming Error	• The resulting word in Flash does not equal the word that was trying to be programmed into Flash	• The Bootloader tries to program Flash only once; there is a problem with internal DSP Flash
7	Internal Error	• Bootloader data corrupted	• Try to reload Bootloader via CodeWarrior

If an application previously loaded via the Bootloader uses the SDK variable BSP_BOOTLOADER_DELAY, (see **Section 3.1.9.4**), the Bootloader waits for the S-Record file

only until the required time-out expires, then launches the application. When this happens, the terminal window contains a message similar to this:

```
"(c) 2000-2001 Motorola Inc. S-Record loader. Version 1.3

Application started."
```

3.1.9.4 Requirements for a Loaded Program

If the application is loaded via the Bootloader, it must meet the following requirements:

- **Particular start address for application** - The entry point for the loaded application must be located at address 0x0080 in Program memory; i.e., immediately after the interrupt table
- **Application COP vector** - To use COP interrupt vector in an application loaded via Bootloader, the entry point for the COP ISR must be located at address 0x0082 in Program memory
- **Application start delay variable must be set at address 0x0085 in Program Flash** - The SDK provides a configuration variable for this purpose. For details on managing the Bootloader without SDK support, see file *config\vector.c* in the SDK.
- **Restricted resources use** - The application cannot occupy Reset and COP interrupt vectors, and cannot place code into Boot Flash memory area. There is no way to place any initialized variable from the application into internal data RAM while loading. This memory area is used by Bootloader. All data from the S-Record file that address to the restricted area will be ignored.

All applications built with the SDK can be loaded via the Serial Bootloader. The SDK provides a fixed address for application entry point; redirection for the application COP vector; and has a configuration variable (BSP_BOOTLOADER_DELAY) for the *appconfig.h* file that determines the application's start delay time-out.

Possible values of BSP_BOOTLOADER_DELAY:

0	Disable the Bootloader, start the application immediately. After loading the application with this setting, there are only two ways to reenter Bootloader: • Reload the Bootloader from Metrowerks CodeWarrior into Boot Flash • Reprogram the value of the start delay variable located at address 0x0083 in the Program Flash to zero value (0x0000) in the loaded application
0 - 254	Set waiting time for the start of the S-Record file for 0-255 seconds before the start of a previously loaded application
255	Set waiting time to infinity. After reset, the Bootloader waits for the S-Record file without counting down any time-out

If BSP_BOOTLOADER_DELAY is not set in the *appconfig.h* file, the default value for time-out is 30 seconds.

Index

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1–303–675–2140 or 1–800–441–2447

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3–20–1, Minami–Azabu. Minato–ku, Tokyo 106–8573 Japan. 81–3–3440–3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852–26668334

Technical Information Center: 1–800–521–6274

HOME PAGE: http://www.motorola.com/semiconductors/