

# Freescale DSC Flasher

## Digital Signal Controller Stand-Alone Flash Programmer

by *Freescale Semiconductor, Inc.*

## 1 Introduction

The Freescale DSC Flasher is a stand-alone flash programmer for the MC56F8xxx family of Digital Signal Controller products. It is meant as an alternative to the *CodeWarrior Flash Programmer* that is integrated within the CodeWarrior development tool. Any and all S-records must be generated by CodeWarrior first and tested using the CodeWarrior built-in Flash Programmer. The Freescale *DSC Flasher: Stand-Alone DSC Flash Programmer* application is provided under a command-line interface and is intended to easily integrate with scripts. The application command-line instructions are provided in [Section 2.1.1](#), “DSC Flasher usage.”

## 2 Flash Programmer description

The DSC Flasher uses S-records to program or verify the target device. [Section 3](#), “[Generating an S-record using CodeWarrior](#)” provides detailed instructions. If you use a Freescale Tower (TWR) device, remove the SBDM CONNECT and OSBDM EN jumpers prior to programming. The simplified flow chart in [Figure 1](#) outlines the application

### Contents

1. Introduction . . . . .	1
2. Flash Programmer description . . . . .	1
3. Generating an S-record using CodeWarrior . . . . .	10
4. Adding a device . . . . .	13
5. Revision history . . . . .	15

communication flow. The Freescale DSC Flasher configures the CodeWarrior Connection Server (CCS) for the specified target device and run controller (CW USB TAP). A series of CCS commands are executed based on this configuration to verify, program, or erase a chip’s flash memory.

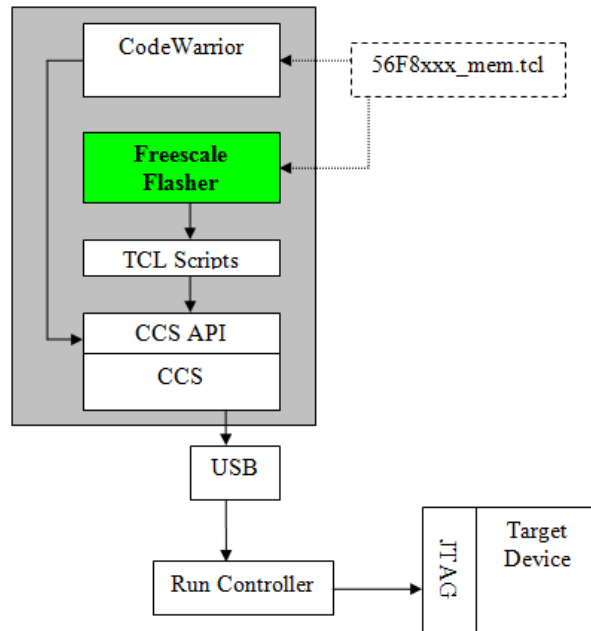


Figure 1. DSC Flasher Application Communication Diagram

## 2.1 Command line

The command-line interface is effective when the Freescale DSC Flasher is integrated within scripts or executed by other programs which control the manufacturing process. The application usage is defined in [Section 2.1.1, “DSC Flasher usage.”](#) Mandatory parameters are indicated by <> angle brackets and optional parameters are indicated by [] brackets. [Section 2.1.1, “DSC Flasher usage”](#) also specifies the command-line terminal usage of the -?|-h|--help options.

### 2.1.1 DSC Flasher usage

Usage: fflasher <TargetDevice> <UTAP|ETAP> <PROGRAM|VERIFY|ERASE> [--srec[:file1 file2...fileN ]] [-m|--mass] [-l|--lock] [-v|--verbose] [(-j|--jtag) <JTAG Clock (kHz)>] [(-t|--timeout) <CCS Timeout (sec)>] [(-r|--remote) <IP Address>] [--log <file>] [-h|--help] [-?] [-d|--devices]

- <TargetDevice> MANDATORY—Declares the target device to access
- <UTAP|ETAP> MANDATORY—Communicates with the target device through the specified run controller
- <PROGRAM|VERIFY|ERASE> MANDATORY—Performs the specified operation on target device  
 PROGRAM—Program and verify the target device  
 VERIFY—Verify the data on the device matches the S-record  
 ERASE—Erase the target device

<code>[--srec[:file1 file2 ... fileN ]]</code>	Include this list of S-record(s) when verifying, erasing pages, or programming
<code>[-m --mass]</code>	Mass erase the target device
<code>[-l --lock]</code>	Lock the device after programming
<code>[-v --verbose]</code>	Verbose information is printed to console and log
<code>[(-j --jtag) &lt;JTAG Clock (kHz)&gt;]</code>	Set the JTAG clock speed (in kHz) (default: 1000)
<code>[(-t --timeout)&lt;CCS Timeout (sec)&gt;]</code>	Changes the network timeout (in seconds) that the CCS uses when communicating with remote devices or servers. (default: 10)
<code>[(-r --remote) &lt;IP Address&gt;]</code>	Remotely connects to run controller over IP address
<code>--log &lt;file&gt;</code>	Saves a log in this file location
<code>[-h --help]</code>	Displays this message
<code>[-?]</code>	Displays this message

## 2.2 Command line operation

When the Freescale DSC Flasher is called from a script or another program, the output messages can be redirected to a file for later inspection by using the **--log** option. For easy integration into scripts, the final status of the programming operation is also provided through error code messages. The following sub-sections detail the mandatory and optional command-line parameters.

### 2.2.1 Mandatory parameters

Position of the mandatory parameters on the command line must follow a strict order:

```
fflasher <TargetDevice> <UTAP|ETAP> <PROGRAM|VERIFY|ERASE>
```

#### 2.2.1.1 Part number/<TargetDevice>

The mandatory target device parameter indicates the device part number that will be programmed, verified, or erased. This device is connected to a run controller. To view a list of supported devices, the following command must be launched from the command line:

```
fflasher --devices
```

#### 2.2.1.2 Run controller/<UTAP | ETAP>

The application supports two run controllers, UTAP and ETAP. A run control device sends CCS commands to the target device in order to read and write flash memory. The role of the run control device can be seen in [Figure 1](#).

### 2.2.1.3 Operation/<PROGRAM|VERIFY|ERASE>

The application can program, verify, and erase flash units on a target device. By default, only the required pages will be erased from the target device if the operation is set to PROGRAM or ERASE. These pages exist in the S-record. To erase the entire device, set the `-m|--mass` switch on the command line. When PROGRAM is selected on the command line, the target device will automatically verify that the S-record has been successfully programmed.

## 2.2.2 Optional parameters

The position of optional parameters on the command line are not fixed and can be declared in any order.

### 2.2.2.1 --srec[:file1 file2 ... fileN ]

This option indicates the list of S-records to include when programming, verifying, or erasing pages of the target device. The S-record(s) chosen must be generated by CodeWarrior. The `--srec[:file1 file2 ... fileN ]` parameter is mandatory, unless selecting ERASE as the Operation/<PROGRAM|VERIFY|ERASE> and performing a mass erase with the `-ml--mass` switch.

**Example:** `fflash MC56F84789 utap verify --srec:C:\Documents\dsc_heartbeat.S`

### 2.2.2.2 -ml--mass

This option mass erases the flash on the target device, which is useful when programming or erasing a secured device. This command unsecures a device and performs the operation specified on the command line. When this command is not specified, only the pages in the flash memory that require programming will be erased.

**Example:** `fflash MC56F84789 utap program -m --srec:C:\Documents\dsc_heartbeat.S`

**Example:** `fflash MC56F84789 utap erase -m`

### 2.2.2.3 -ll--lock

This option will lock the device by enabling flash security. The DSC Flasher will initialize the target to lock before the S-record is programmed. Notice this only enables flash security. Any advanced options, such as back door enable and the back door key will require a loadable S-record or ELF file that defines this data.

**Example:** `fflash MC56F84789 utap program --lock --srec:C:\Documents\dsc_heartbeat.S`

### 2.2.2.4 -v|--verbose

This option prints more verbose information to the console and log. Verbose messages consist mainly of CCS commands.

### Example 1. Verbose Console Example

---

Performing PROGRAM operation on MC56F8006 through the UTAP . . .

```
Reading S-Record : C:\Users\B44469\workspace_launcher\DSC_Flasher\srecords\mc56f8006.elf.p.S
Erasing Page(s) from Target Device . . .
ccs::fill_mem 0 0x00000000 2 2 0x00000080 { 0xFFFF }
ccs::fill_mem 0 0x00000100 2 2 0x00000080 { 0xFFFF }
ccs::fill_mem 0 0x00000200 2 2 0x00000080 { 0xFFFF }
Finished S-Record : C:\Users\B44469\workspace_launcher\DSC_Flasher\srecords\mc56f8006.elf.p.S
Erase Complete
```

```
Reading S-Record : C:\Users\B44469\workspace_launcher\DSC_Flasher\srecords\mc56f8006.elf.p.S
S-record ID: PROGRAM
Programming Target Device . . .
ccs::write_mem 0 0x00000000 2 2 { 0x0004 words }
ccs::write_mem 0 0x00000000 2 2 57684 121 57684 121
ccs::write_mem 0 0x00000000 2 2 { 0x022d words }
ccs::write_mem 0 0x00000000 2 2 57684 121 57684 ...
Finished S-Record : C:\Users\B44469\workspace_launcher\DSC_Flasher\srecords\mc56f8006.elf.p.S
Programming Complete: (9 lines, 7 records)
```

---

#### 2.2.2.5 -jl--jtag <JTAG Clock (kHz)>

This option tells the CCS which frequency (kHz) to run the JTAG interface. The default value is 1000 kHz. Faster clock speeds enable faster programming, however if the clock is set too fast, the DSC chip may not be able process incoming data quick enough which eventually leads to a programming failure.

**Example:** `fflash MC56F84789 utap program --srec:C:\Documents\dsc_heartbeat.S -j 800`

#### 2.2.2.6 -tl--timeout <CCS Timeout (sec)>

This option tells the tool how long to wait in seconds for the CCS to respond before it will abort and return an error. The default value is 10 seconds.

**Example:** `fflash MC56F84789 utap erase -m --timeout 5`

#### 2.2.2.7 -rl--remote <IP Address>

If connecting via CodeWarrior USB TAP remotely, this options connects to the <IP Address> specified. The same operations and options can then be performed on the target device as on a local host.

## NOTE

The IP address of the local computer is 127.0.0.1.

### Examples:

Connects to the local computer, at port 1000

```
fflash MC56F84789 utap erase -m -r 127.0.0.1:1000
```

Connects to the remote CCS at IP address 1.2.3.4

```
fflash MC56F84789 utap erase -m -r 1.2.3.4
```

Connects to the remote CCS at IP address 1.2.3.4 at port 1000

```
fflash MC56F84789 utap erase -m -r 1.2.3.4:1000
```

### 2.2.2.8 --log <file>

This option forces all messages shown on the screen to be written into the log file. If the file already exists, it will be overwritten. The logfile can be an absolute path, otherwise, when only a <filename> is given, it will be stored in the .logs folder of the Freescale DSC Flasher application.

**Example:** `fflash MC56F84789 utap erase -m --log mylog.txt`

## NOTE

The log pathname is always printed in the console upon execution of any instance of the Freescale DSC Flasher, as long as the **--log** option is specified.

### 2.2.2.9 -dl--devices

This options prints all supported devices to the console.

### 2.2.2.10 -?|-h|--help

This option displays the usage definitions provided in [Section 2.1.1, “DSC Flasher usage.”](#)

## 2.3 GUI version

The GUI version enables a user to quickly program a device. The GUI is typically the preferred programming option when only programming one s-record to a single device. [Figure 2](#) shows the main screen of the GUI.

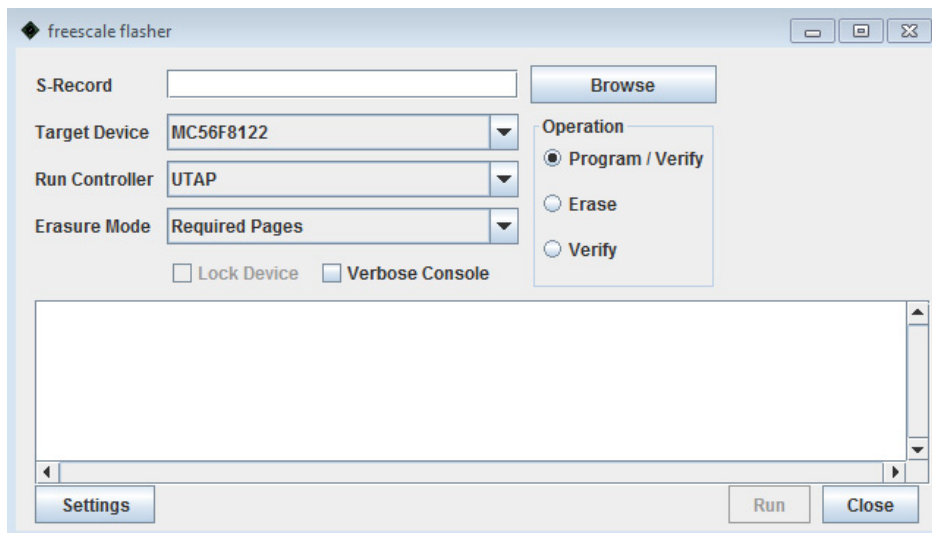


Figure 2. DSC Flasher GUI Main Screen

## 2.3.1 GUI operation

To program, erase, or verify a target device, select the options on the main screen as shown in [Figure 2](#). Set the options on the setting screen shown in [Figure 3](#) and then click Run.

### 2.3.1.1 S-record

Click Browse to search for the S-record, or copy-and-paste the S-record into the S-record text box. The S-record chosen must be generated by CodeWarrior. This option is identical to the `--src[:file1 file2 ... fileN ]` option on the command line.

### 2.3.1.2 Target device

This pull-down menu enables the user to select the target device for the flash operation. These devices can also be seen in `./cfg/devices.cfg` (see [Section 4, “Adding a device”](#)). This option is identical to the `Part Number/<TargetDevice>` selection on the command line.

### 2.3.1.3 Run controller

This pull-down menu enables the user to select the run controller which will perform the flash operation. This section is identical to the `Run Controller/<UTAP|ETAP>` selection on the command line.

### 2.3.1.4 Erasure mode

This pull-down menu enables the user to select the erasure mode to erase all or part of the target device. Selecting Required Pages will erase only pages that are required for programming or erasing as specified by the included S-record. Selecting Mass Erase will erase all of the flash memory on the target device, which is useful when programming or erasing a secured device. The mass erase command unsecures a

device. The Erasure Mode option is ignored when verifying a target device. This option is similar to the **-ml--mass** option on the command line.

### 2.3.1.5 Operation

These buttons enable you to select between programming, erasing, or verifying a target device. This section is identical to the Operation/<PROGRAM|VERIFY|ERASE> option on the command line. Selecting PROGRAM will program the flash memory and then verify that the flash memory matches the data in the S-record.

### 2.3.1.6 Lock device

This option will lock the device by enabling flash security. The DSC Flasher will initialize the target device to lock before the S-record is programmed. Notice this only enables flash security. Any advanced options, such as back door enable and the back door key will require a loadable S-record or ELF file that defines this data. This option is identical to the **-il--lock** option on the command line.

### 2.3.1.7 Verbose console

This option will print verbose messages to the console window within the main screen of the GUI. Verbose messages consist mainly of CCS commands. This option is similar to the **-vl--verbose** option on the command line.

## 2.3.2 CCS settings

The CCS settings screen is specific to CCS settings. This includes modifying the JTAG clock speed, CCS network timeout, and specifying the IP Address of a CCS remote session.

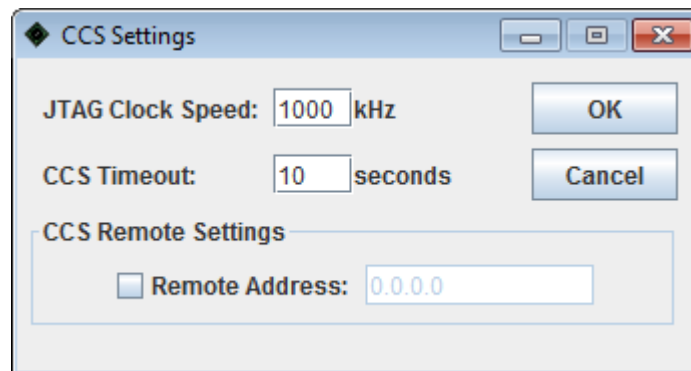


Figure 3. DSC Flasher GUI Settings Screen

### 2.3.2.1 JTAG clock speed

This section initializes the CCS frequency (kHz) at which to run the JTAG interface. The default value is 1000 kHz. Faster clock speeds enable faster programming, however when the clock is set too fast, the DSC chip may not be able to process incoming data quickly enough. This may eventually lead to a programming failure. This is identical to the **-j|--jtag<JTAGCLOCK>** option on the command line.



### 2.3.2.2 CCS timeout

This section indicates the time to wait in seconds for the CCS to respond before it will abort and return an error. The default value is 10 seconds. This is identical to the `-t|--timeout<CSS Timeout (sec)>` option on the command line.

### 2.3.2.3 Remote address

When connecting via a run controller remotely, this options connects to the <IP Address> specified. The same operations and options can then be performed on the target device as on a local host. This is identical to the `-r|--remote <IP Address>` option on the command line. [Figure 4](#) through [Figure 6](#) provide examples showing the connections given the remote address entry.

#### Remote address examples:

Connects to the local computer, at port 1000.

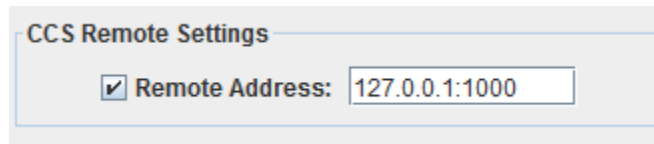


Figure 4. Remote Address to Local Computer Setting

Connects to the remote CCS at IP address 1.2.3.4

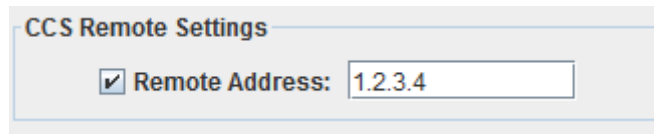


Figure 5. Remote Address to Remote CCS Setting

Connects to the remote CCS at IP address 1.2.3.4 at port 1000

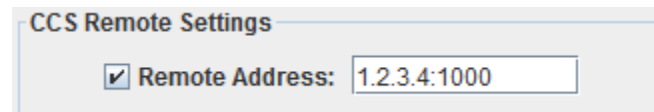


Figure 6. Remote Address to Remote CCS on Specified Computer Setting

## 2.4 Error codes and messages

The Freescale DSC Flasher generates error messages in various situations. The error message are unique but all fall into the error code categories 0 through 9 as described in detail in [Table 1](#). These error codes are displayed in error messages when the execution completes. Error Code 0 does not display any error message. Example error and completion messages are given following the table.

Table 1. Error Codes and Description

Error Code	Description
0	Operation Finished Successfully

**Table 1. Error Codes and Description (continued)**

1	S-record File not Found or Corrupted
2	Flash Configuration File not Found
3	Error in Configuration File
4	Unsupported Target Device
5	Flash Verification Error
6	Incorrect Parameters
7	System Error
8	Chip is Secured
9	CCS Error (See API for description)

**Example:** Error 8: Device is secure. Set the Mass Erase switch to unsecure the device.

**Example:** Flasher ERASE complete.

### 3 Generating an S-record using CodeWarrior

The following procedure to generate an S-record uses CodeWarrior v10.3. These steps are not guaranteed to work using any other versions. The S-record file is generated based on the linker setup. The linker will generate three different S-record files:

- **output\_file.p.S**—contains data and code to be stored in program memory locations.
- **output\_file.x.S**—contains data and code to be stored in data memory locations.
- **output\_file.S**—combination of the two previous files.

In the combined S-record file, data memory locations are distinguished from program memory locations by having addresses greater than 0x02000000 (Word Address). The combined file is the accepted format for the application.

The correct linker setup is described in the following steps:

1. Open CodeWarrior and right-click the project folder in which you want to create an S-record.
2. Select Properties in the right-click menu
3. In the Properties window, expand C/C++ builder and select Settings.
4. In the settings window, expand DSC Linker and select Output.
5. In the Linker Output menu, check the Generate S-record File box. CodeWarrior will now generate S-records on the next build. The Freescale flash memory will accept any combination of options that the Linker set-up dialog will support, however Generate Byte Addresses must remain unchecked.
6. Click Apply, then Okay.
7. Build the project by selecting Project -> Build All. The S-records can now be found in the project folder within the expanded Flash folder of the target device (FLASH\_LDM / FLASH\_SDM).

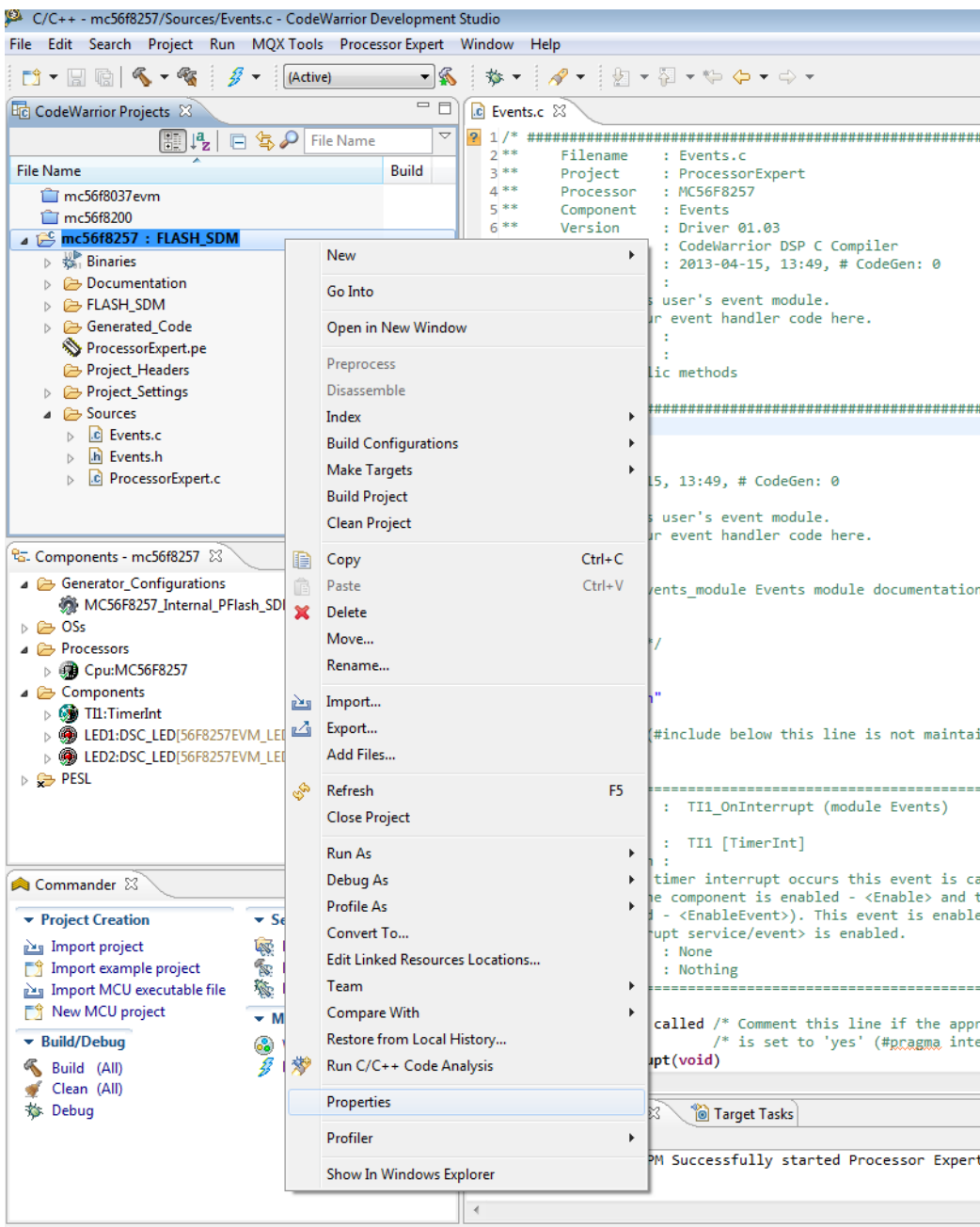


Figure 7. Adding a Device, Steps 1–2

### Generating an S-record using CodeWarrior

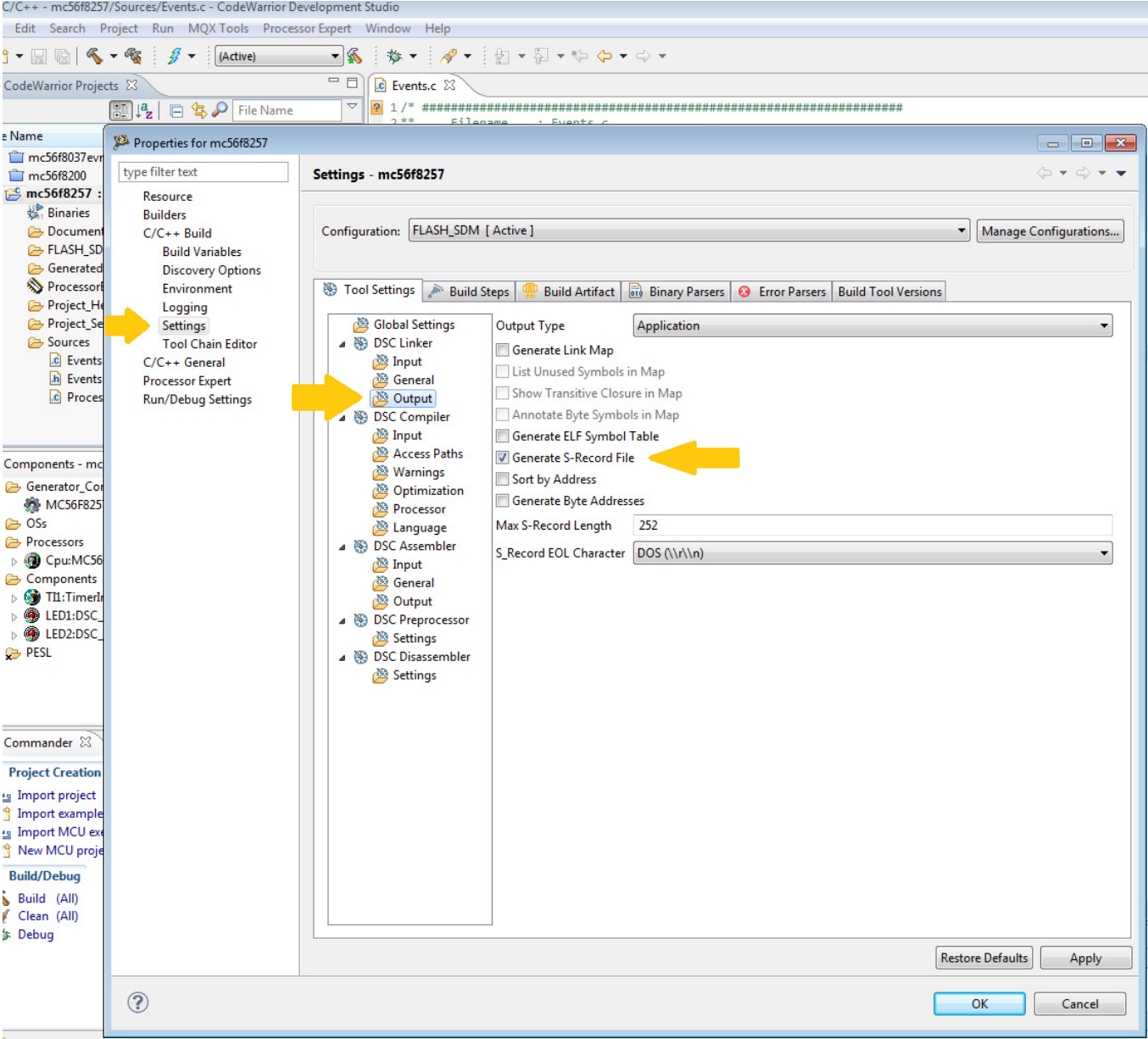


Figure 8. Adding a Device, Steps 3–6

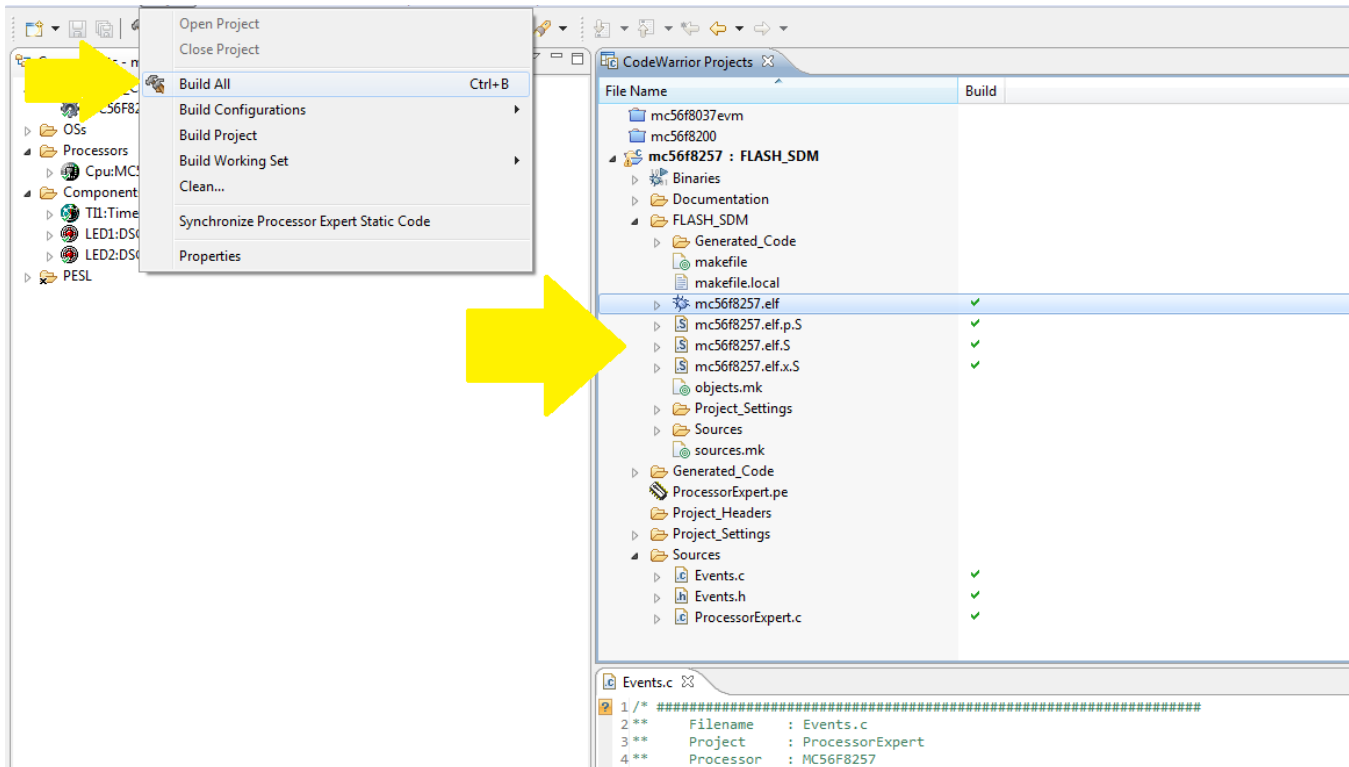


Figure 9. Adding a Device, Step 7

## 4 Adding a device

There exists the option for a user or developer to add their own device. This is useful if the user wants to initialize registers prior to writing or erasing the flash memory of a device.

### 4.1 Adding a target device

To add a target device, a user must access and edit two files:

1. The device initialization file (`.\ccs\bin\MC56F8xxxx_init.tcl`)
2. The device configuration file (`.\cfg\devices.cfg`)

If a device has been successfully added, the user will see it from the pull-down menu for the target devices within the GUI, or by enabling the `--devices` switch on the command line.

#### 4.1.1 Initialization file

The device `init.tcl` files are attainable from CodeWarrior or from the device reference manual:

1. Start `ccs.exe` from `$yourCWDDirectory/MCU/ccs/bin`.
2. Connect to the run controller (EX: `config cc utap; show cc`).
3. In CW debug configurations click Edit and select Manual connection and uncheck connect to TAP. CodeWarrior will use the existing default server listening port when connecting.
4. In CCS, type: `log tcl filename.tcl`.

5. In CodeWarrior, start debugging until CodeWarrior stops at the breakpoint.
6. In CCS, type: log quiet.
7. In CCS, type: puts \$ccsDataPath. ccsDataPath is a directory where the log file is saved.

Within this log you will find CCS commands that are part of the initialization file. Follow the structure in the existing initialization files found in the Freescale DSC Flasher application ./ccs/bin directory. The initialization files are divided into two sections:

1. Config\_chain
2. Config\_template #6-#27

The config\_template commands are unique to each family, and sometimes each device. An example for the MC56F84789 target device is listed in [Figure 10](#):

```

30 #####
31 # INITIALIZE TEMPLATE CONFIGURATIONS #
32 #####
33
34 # number of flash block units
35 ccs::config_template $chain_pos 6 0x00000000
36 ccs::config_template $chain_pos 6 0x00000002
37
38 # Primary program flash memory
39 # establish current flash unit
40 ccs::config_template $chain_pos 7 0x00000000
41
42 # current flash unit address
43 ccs::config_template $chain_pos 8 0x00000000
44
45 # current flash unit size in bytes - 256K
46 ccs::config_template $chain_pos 9 0x00040000
47
48 # current flash unit memory space
49 ccs::config_template $chain_pos 10 0x00000002
50
51 # current flash unit memory bank
52 ccs::config_template $chain_pos 11 0x00000000
53
54 # current flash unit interleave flag
55 ccs::config_template $chain_pos 12 0x00000000
56
57 # current flash unit page size in words
58 ccs::config_template $chain_pos 13 0x00000400

```

**Figure 10. Example Template Configurations for MC56F84789**

The initialization file is responsible for initializing the following for each flash unit:

- Flash unit base address
- Flash unit size in bytes
- Flash unit memory bank

- Flash unit interleave flag
- Flash unit page size in words
- Flash unit memory space

#### NOTE

Memory Space #0 = Program RAM

Memory Space #1 = Data RAM

Memory Space #2 = Program Flash

Memory Space #3 = Data Flash

### 4.1.2 Configuration file

After the initialization file has been written, the user must link the init file to a string/name for the device. This is done in the configuration file. In addition to naming the device, the user must specify the endianness (BIG || LITTLE) of the architecture, and the number of bytes per word (addressability) for the architecture. Each device requires only one line, and comments can be made with preceding or trailing //.

The syntax for declaring a new device:

```
ModelName: initScriptForModel : bytesPerWord : Endianness
```

Colons must separate each value. Examples are shown in [Figure 11](#).

```
1 // ModelNumber: initScriptForModel : bytesPerWord : Endianness
2 // ModelNumber : ex - MC56F84789
3 // initScript : Relative Path ( in relation to fflasher.exe ) to the initialization tcl script for the ModelNumber
4 // bytesPerWord : Number of Bytes per Word According to the Architecture of the Chip
5 // Endianness : BIG || LITTLE
6
7 //MC56F800x Family
8 MC56F8002 : .\ccs\bin\MC56F8006_init.tcl : 2 : LITTLE ;
9 MC56F8006 : .\ccs\bin\MC56F8006_init.tcl : 2 : LITTLE ;
10
11 //MC56F801x Family
12 MC56F8013 : .\ccs\bin\MC56F8013_init.tcl : 2 : LITTLE ;
13
```

Figure 11. devices.cfg

## 5 Revision history

Revision 0 is the initial release of this document.

**How to Reach Us:**

**Home Page:**

[freescale.com](http://freescale.com)

**Web Support:**

[freescale.com/support](http://freescale.com/support)

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [freescale.com/SalesTermsandConditions](http://freescale.com/SalesTermsandConditions).

Freescale, the Freescale logo, CodeWarrior, and the Energy Efficient Solutions logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Tower is a trademark of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2013 Freescale Semiconductor, Inc.