Freescale Semiconductor
Application Note

Document Number: AN3391
Rev. 0, 01/2007

LFAE Bootloader Example and Interface for use with AN2546

by: Daniel McKenna
MCD Applications,
East Kilbride Scotland

Contents

1 Introduction

This application note describes an example of a load flash and execute (LFAE) secondary bootloader. The application has been developed to enhance the existing load RAM and execute (LRAE) primary bootloader described in AN2546 by allowing arbitrary code to be written to flash memory.

The intent of this document is to provide an example of how the reader may implement a secondary bootloader. A secondary bootloader may be used to program application code into a device in-system or during a mass production flow via existing CAN or SCI interfaces, removing the need for a BDM interface for programming an application during production. It assumes the primary bootloader has already been programmed into the device via BDM or third party programmer.

An interface utility running on a PC with a 32-bit Windows operating system transmits all data to the MCU. Communication is supported over a CAN or SCI connection. This interface is a simple example of how to set and monitor transmissions.

A brief synopsis of the steps involved in this process is shown below. The scenario described assumes the LRAE primary bootloader has already been loaded to the flash memory of the MCU.

1. The interface and primary bootloader establish the means and speed of communication.
2. The interface transmits the LFAE secondary bootloader to the MCU that stores it to RAM and then executes it.
3. The LFAE secondary bootloader takes control and re-establishes a connection with the interface
4. The interface transmits application code to the MCU that stores the code in flash.
5. Upon completion of transmission, both bootloaders are removed and the code is executed.

The bootloader and interface software described in this document is for example only and comes with no guarantees and no support.

1.1 Overview

The purpose of this application note is to use a modified version of the LRAE bootloader (described in AN2546) to download a larger, secondary bootloader that can then write application code to flash.

The software for this application is available in AN3391SW.

1.2 Acronyms

- CAN – Controller area network
- LFAE – Load flash and execute
- LRAE – Load RAM and execute
- LSB – Least significant byte
- MCU – Microcontroller unit
- MSB – Most significant byte
- RAM – Random access memory
- SCI – Serial communication interface
- BDM - Background Debug Module

1.3 S-Record (SREC)

The bootloader accepts the binary image of the application in form of an S-record (SREC) file. SREC files represent binary object code in a printable ASCII format, making them easily transportable between different systems. Lines in an SREC begin with a prefix such as S1 (header) or S9 (footer). The pertinent prefixes for the bootloaders are S1 and S2.

An S1 line describes data stored in logical memory whereas an S2 line describes data held in global memory. However, S1 and S2 lines have the same template seen here:

Type	Record Length	Address	Data	Checksum

The type field contains the aforementioned prefix. Record length is a 1 byte (2 ASCII characters) long value that states the number of bytes contained in all proceeding fields. The address field gives the start address the line should be written to. In an S1 line, the address is 2 bytes long (e.g. a 16 bit address stored in local memory). In an S2 line, the address is 3 bytes long (the standard 16 bits plus a 7 bit PPAGE value). The data field contains the memory loadable data. Finally, calculate the checksum by summing the record length, address, and data field; then, perform a modulo 256 and take the ones complement.

2 Requirements

This section gives a brief overview of software and hardware required to load an application to the MCU via a primary and secondary bootloader.

2.1 Hardware

The method detailed in this application note requires an S12XE MCU with the primary bootloader installed in flash. For testing purposes, the EVB9S12XEP100 Kit was used.

A PC with a 32-bit Windows compatible OS is required to run the interface. The PC must have an SCI port or a PC-CAN card for CAN-based communications. For testing purposes, the Vector CANalyzer card was used.

2.2 Software

All coding for the MCU was carried out in CodeWarrior V5.70.

To ensure a fast and smooth transition between the primary and secondary bootloaders, the LRAE required some minor alterations.

- An extra flag should be added to the CANFlags structure to denote whether the bootloader is using the SCI or CAN interface.
- The CAN frame received flag (canrflg.rxf) should be cleared before control is passed over to the secondary bootloader.

3 Functional Description

This section elaborates on how the secondary bootloader is loaded into RAM and details the fundamentals of its operation to show how it collaborates with the interface utility to download and launch an application.

3.1 Loading Secondary Bootloader

To ensure compatibility with the primary bootloader, the secondary bootloader has been arranged in memory so the addresses in the SREC are strictly sequential (i.e. it contains a value for all memory locations within the scope of the record). See Figure 1 for a basic flowchart of the interface utility transmitting the secondary bootloader’s SRECs.

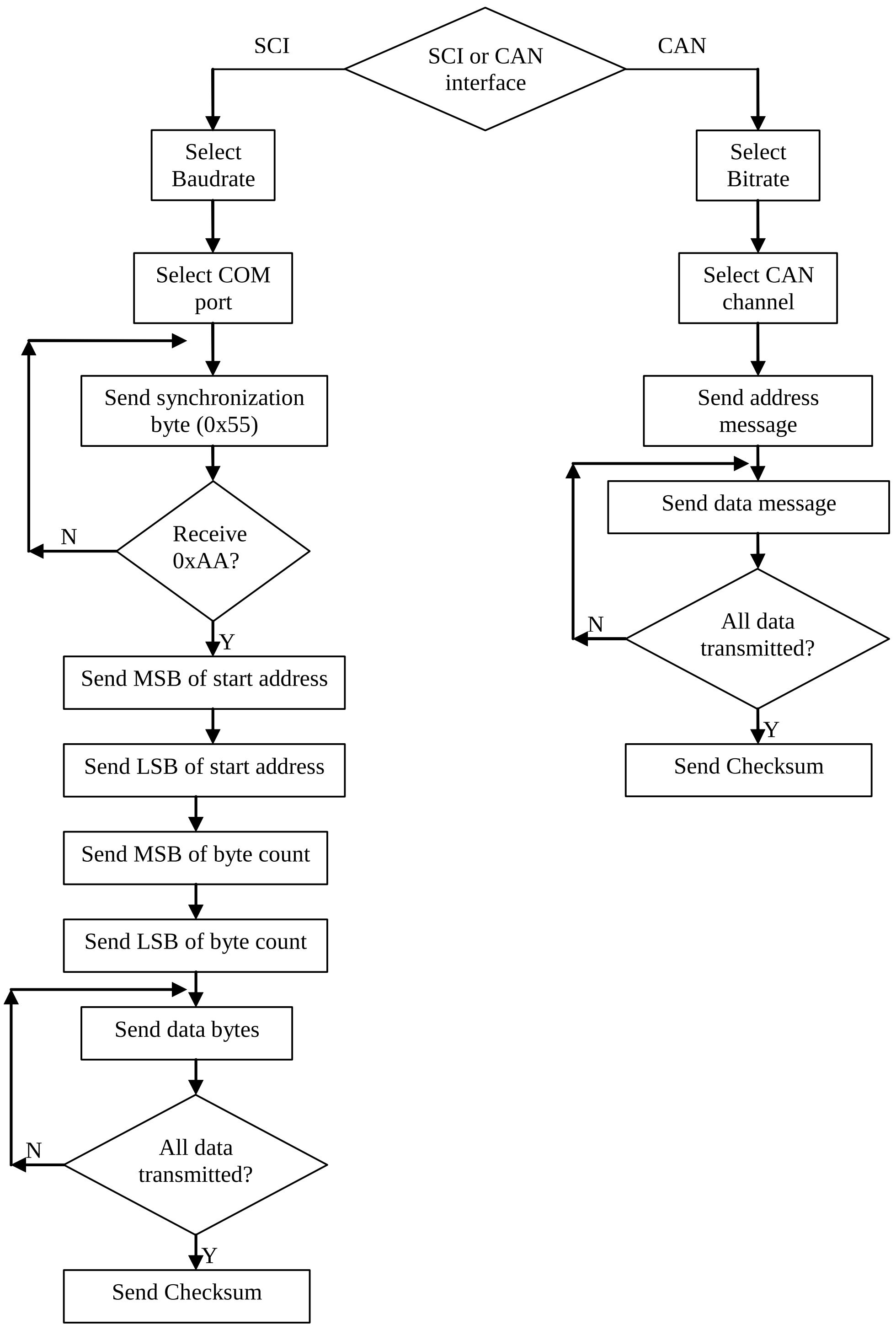
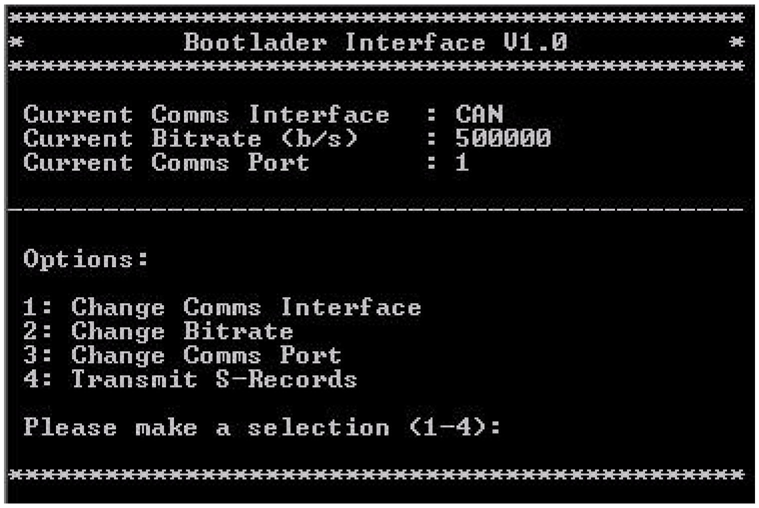


Figure 1. Flow of Interface Utility Transmitting the Secondary Bootloader

3.1.1 SCI Interface

If an SCI port is chosen for transmission of the data, the baud rate must be chosen from a list of compatible speeds. Declare which of the PC’s COM ports the serial connection should be made with. The interface utility then transmits a synchronization byte (0x55) and pauses to receive an acknowledgement byte (0xAA). Failure to receive acknowledgement within the delay period of 50 ms causes the interface to repeat the procedure. After completion of this step, processing of the SREC file can commence.



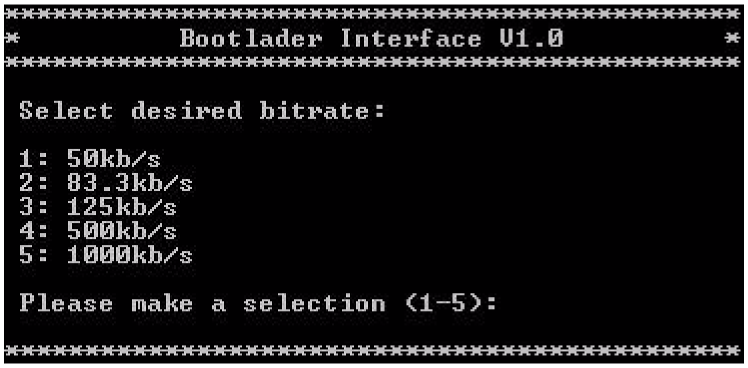
```
***********************************************
*           Bootlader Interface V1.0          *
***********************************************

 Current Comms Interface   : CAN
 Current Bitrate (b/s)     : 500000
 Current Comms Port        : 1

-----------------------------------------------

 Options:

 1: Change Comms Interface
 2: Change Bitrate
 3: Change Comms Port
 4: Transmit S-Records

 Please make a selection (1-4):

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

Figure 2. Main Menu of Bootloader Interface

As the secondary bootloader is stored in local RAM, the interface opens the SREC and searches for the first S1 line. From this, it obtains and transmits the start address of the code, MSB first. The byte count of all data to be transmitted is calculated by summing the value in the length field on each line (less the address and checksum bytes). This is also transmitted MSB first. The bootloader is now ready to accept data; therefore, the interface utility begins extracting the data from each line of the SREC and transmits it one byte at a time. After completion of this step, the checksum (calculated by summing the value of all transmitted bytes modulo 256) is also transmitted. If the checksum matches the value calculated by the bootloader, the secondary bootloader is executed and takes control of the S12XE.

3.1.2 CAN Interface

After selecting to transmit the data via a CAN connection, choose which channel to use and the bitrate to transmit the data. After completion of this step, the interface follows the same flow as with an SCI except in this instance, as CAN packets can store up to 8 bytes, the address and byte count data are sent in their own distinctively identified messages. Furthermore, the data can also be sent 8 bytes at a time.



```
*********************************************
*          Bootlader Interface V1.0         *
*********************************************

 Select desired bitrate:

 1: 50kb/s
 2: 83.3kb/s
 3: 125kb/s
 4: 500kb/s
 5: 1000kb/s

 Please make a selection (1-5):

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

Figure 3. CAN Bitrate Selection Menu

The specific identifiers for each message, as well as the correct arrangement of data within the message are fully detailed in AN2546.

3.1.3 Memory Map

The secondary bootloader resides in local RAM between the logical addresses 0x2000 – 0x2525. To operate, the secondary bootloader also needs access to variables from the primary bootloader, situated between 0x3FD0 – 0x3FDA. The primary bootloader is situated in local flash, between 0x4000 – 0x440A. Prior to execution of the secondary bootloader, the local memory map of the S12XE is as depicted on Figure 4.

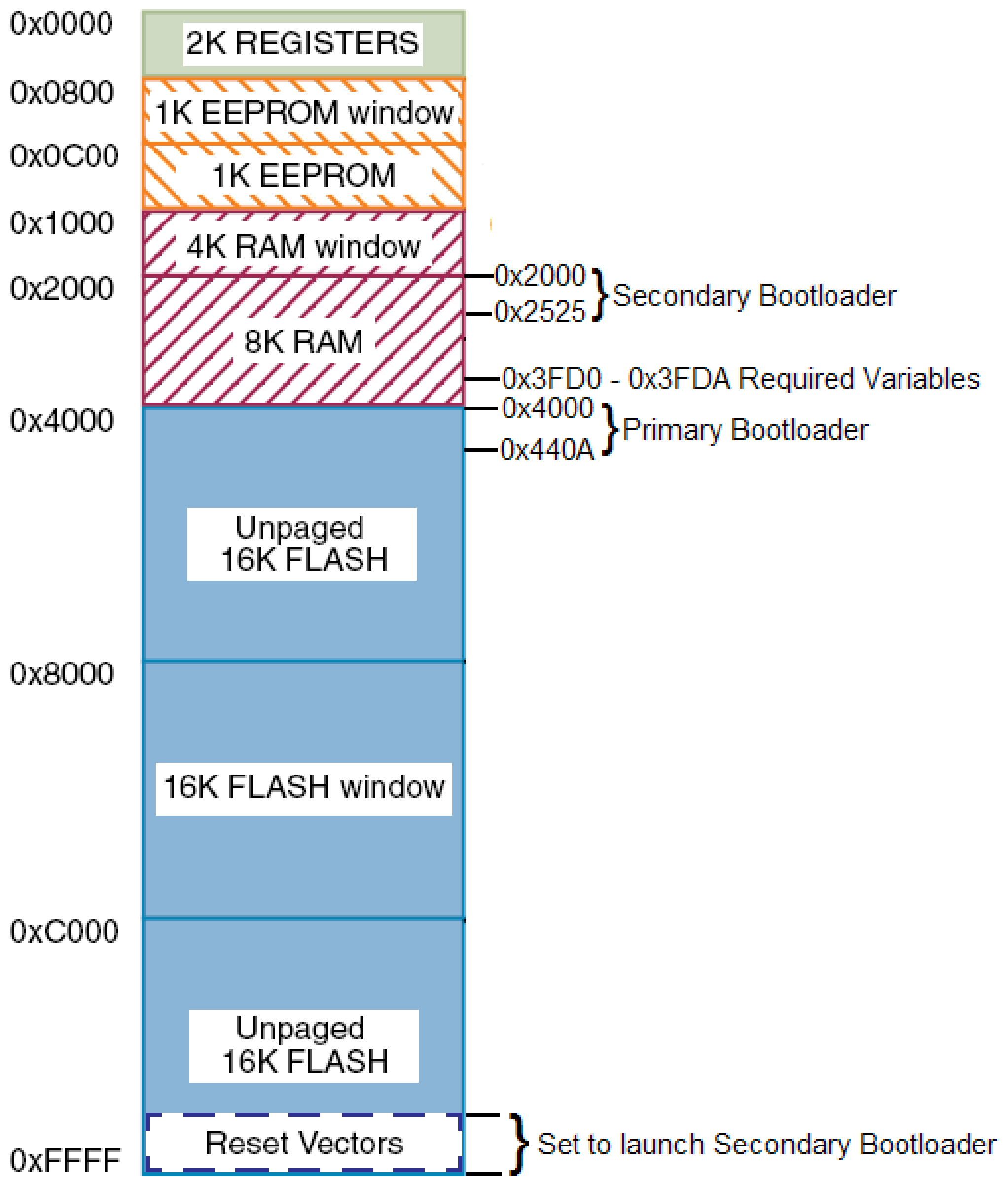


Figure 4. Memory Map After Installation of Secondary Bootloader

3.2 Loading Application

After the secondary bootloader takes control of the S12XE, it begins preparing to receive the SRECs of the application. To identify which communication interface to use, the secondary bootloader looks at several variables stored by the primary bootloader. It is able to do this because the variables are stored in RAM and the S12XE has not been reset during the changeover process.

The variable SCI_or_CAN is investigated first to find which communication interface to use. SCI_Ptr or CAN_Ptr, which are pointers to the actual hardware peripheral, are used during communication. All speed and timing information remains present from the primary bootloader.

Because it is not permitted to program any flash location more than once, the sector containing the reset vectors and the program of the primary bootloader is erased. This is a necessary operation that has the unfortunate side effect of leaving the S12XE vulnerable to entering an unrecoverable state should any error or power failure occur before loading of the application is complete.

Unlike the secondary bootloader, the application's SRECs do not have to contain sequential addresses. This allows the reset vectors to be stored in any SREC.

3.2.1 Transmitting the Application

Figure 5 shows a flow chart of the steps taken by the interface utility to transmit an application through SCI or CAN interface.

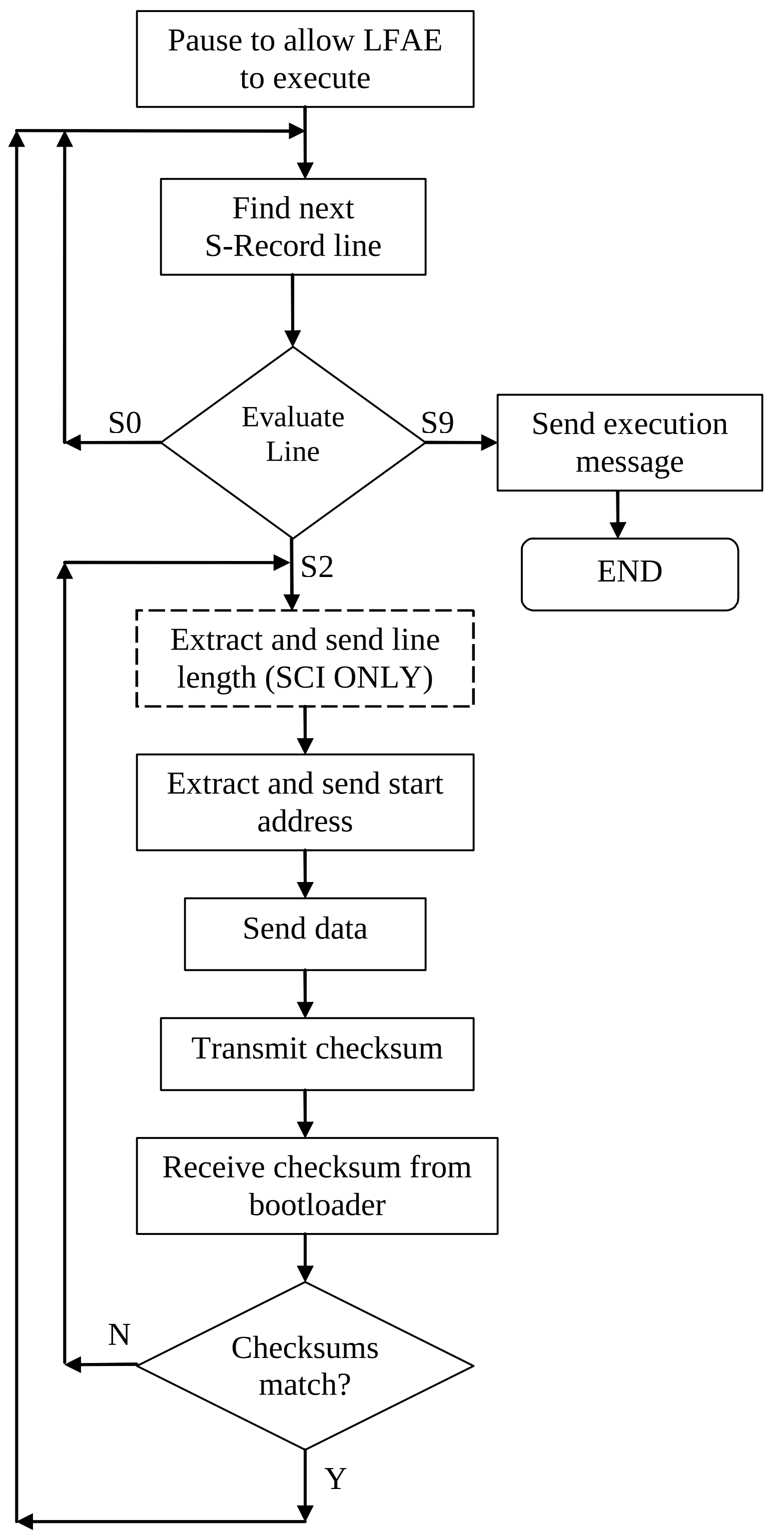

Pause to allow LFAE to execute
Find next S-Record line
Evaluate Line
S0
S9
Send execution message
END
S2
Extract and send line length (SCI ONLY)
Extract and send start address
Send data
Transmit checksum
Receive checksum from bootloader
Checksums match?
N
Y

Figure 5. Transmit Flow of the Interface Utility

The interface utility pauses for 1000 ms to ensure the secondary bootloader has had time to execute and erase the necessary flash blocks. The details of each step from here depend on which communications interface is used.

3.2.1.1 SCI

If an S2 line is found, the interface extracts the line length and transmits it to the bootloader. Afterward, the proceeding 3-byte address is transmitted in this order: PPAGE value, MSB, and then LSB. After completion of this step, the data is transmitted one byte at a time. Finally, the checksum is extracted from the SREC line and sent to the bootloader for verification. If the checksums match, the bootloader responds by transmitting 0x80. Otherwise, the bootloader repeats transmission of the SREC line.

If an S9 line is found, the interface must notify the bootloader that the download is complete. The bootloader expects to receive the line length of the next S2 field. By sending a length of zero, the bootloader knows that the download is complete.

3.2.1.2 CAN

The payload size is automatically embedded in the CAN frames; therefore, there is no need to include this information in the data payload itself. The first packet sent to the bootloader contains the address field. Special frame IDs are used for the address, data, and checksum frames to simplify the reception algorithm. The ID of each frame type is specified in Table 1. The data and checksum packets follow the address frame. The interface utility transmits a special execute frame to start execution of the application.

Table 1. Packet Types and Their Corresponding Identifiers

PACKET	Address	Data	Checksum	Execute
IDENTIFIER	0x020	0x040	0x080	0x010

3.2.2 Receiving and Storing the Application

This section details how the secondary bootloader receives the data and writes it to flash.

The CAN and SCI implementations work in a similar manner. Each SREC line is dealt with on an individual basis. As flash can only be written one phrase (8 bytes) at a time, data is stored in an array as it arrives. If a valid checksum is received, data is written to flash one phrase at a time. After each phrase is written, the address pointer is incremented by eight.

Due to the nature of flash programming, the starting address of the first SREC must fall at a phrase boundary.

3.2.3 Executing the Application

After receiving the execute instruction from the interface utility, the bootloader causes the S12XE to reset by writing to an unimplemented memory location. After reset, the S12XE fetches the power on reset vector, which now points to the first instruction of the application.

4 Compatibility with S12XD

The software accompanying this application note is designed to run on an S12XE and some minor changes are required to ensure it operates correctly on an S12XD device. The header files must be replaced by their S12XD counterparts. This takes into account the differing register positions between the devices. Ensure the registers the code uses have the same bit polarity in the S12XD and S12XE as changes are common between derivatives.

The only other incompatibility is the flash programming segment. The S12XE uses error correction coding to increase data integrity protection. ECC is not available on S12XD. This leads to the S12XD flash having a write size of 2 bytes as opposed to 8 bytes. The registers that manage the write operations also differ significantly with the S12XD.

To ease the transition, the code that carries out the flash writing operation is stored in a separate source file: S12XE_Flash.c. This file can be altered without affecting any other part of the program.

5 Summary

The LFAE bootloader and accompanying interface is the perfect companion for the LRAE bootloader developed in A2546. Together, they provide a solution that allows an application to be loaded to flash on a new S12XE without the need for costly additional equipment.

6 References

Martyn Gallop, *HCS12 Load RAM and Execute Bootloader User Guide*, Freescale Application Note AN2546, 2004

www.freescale.com/files/microcontrollers/doc/app_note/AN2546.pdf

S12XD and S12XE Family Compatibility, Freescale, 2006

http://www.freescale.com

Motorola S-Records

http://www.amelek.gda.pl/avr/uisp/srecord.htm

Ramon de Klein, *Serial library for C++*, 2001

http://www.codeproject.com/system/serial.asp

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3391
Rev. 0
01/2007