AN845

PRODUCTION PROGRAMMING OF EM35X CHIPS

(Formerly document 120-5065-000)

This document describes the flash programming interface of EM35x chips, and is intended for production programming of chips on a manufacturing line. The description assumes all chips being programmed are new from the distributor and have never been programmed before, but may still contain test code in the MFB. The process described in this document is intended to program a final, fixed image and is not intended for use during application development. This document also assumes the connection between the chip being programmed and the programmer is reliable and error free.

New in This Revision

Updated programming steps to add the “Erase CIB” step and better described programming write protection.

Contents

SILICON LABS

1 Introduction

Production programming of flash on EM35x chips is accomplished using flashloader firmware. The programmer communicates with the chip using the Serial Wire and JTAG Interface (SWJ). The SWJ allows the programmer to read from any address and write to any RAM or register address. The SWJ is used to install, execute, and communicate with the flashloader firmware in RAM. The flashloader performs all of the flash manipulation operations using a simple set of commands that operate on shared memory buffers. This application note focuses on the use of the flashloader firmware to manipulate flash contents, how the flashloader should be used in a production programming environment, and the issues surrounding flashloader use including program image, gang programming, and serialization.

2 Pin Connections

Table 1 lists all of the EM35x pins, their names, and their descriptions as they apply to production programming. Refer to the EM35x data sheet and reference manual for further information on the chip's pins and all of their functionalities.

Part orientation errors may be detected by monitoring VREG_OUT after the part is released from reset during the Powerup and CPU Capture step. Once nRESET is released, the 1.8 V plane sourced by VREG_OUT on Pin 15 of the chip should stabilize between 1.7 and 1.9 volts. If the voltage measured is outside of this range, a wrong orientation error should be indicated.

Table 1. Programming Pin Usage

EM35x Pin	Name	Description
1	VDD_24MHZ	1.8 V supply
2	VDD_VCO	1.8 V supply
3	RF_P	Not required for programming
4	RF_N	Not required for programming
5	VDD_RF	1.8 V supply
6	RF_TX_ALT_P	Not required for programming
7	RF_TX_ALT_N	Not required for programming
8	VDD_IF	1.8 V supply
9	NC	Do not connect
10	VDD_PADSA	1.8 V supply
11	PC5	Not required for programming
12	nRESET	Active low chip reset (internal pull-up)
13	PC6	Not required for programming
14	PC7	Not required for programming
15	VREG_OUT	Regulator output (1.8 V)
16	VDD_PADS	2.1-3.6 V pads supply
17	VDD_CORE	1.25 V digital core supply decoupling
18	PA7	Not required for programming
19	PB3	Not required for programming
20	PB4	Not required for programming
21	PA0	Not required for programming
22	PA1	Not required for programming
23	VDD_PADS	2.1-3.6 V pads supply

EM35x Pin	Name	Description
24	PA2	Not required for programming
25	PA3	Not required for programming
26	PA4	Not required for programming
27	PA5	Not required for programming
28	VDD_PADS	2.1-3.6 V pads supply
29	PA6	Not required for programming
30	PB1	Not required for programming
31	PB2	Not required for programming
32	SWCLK	Serial Wire clock input/output with debugger Selected when in Serial Wire mode (see JTMS description, Pin 35)
	JTCK	JTAG clock input from debugger Selected when in JTAG mode (default mode, see JTMS description, Pin 35) Internal pull-down is enabled
33	JTDO	JTAG data out to debugger Selected when in JTAG mode (default mode, see JTMS description, Pin 35)
34	JTDI	JTAG data in from debugger Selected when in JTAG mode (default mode, see JTMS description, Pin 35) Internal pull-up is enabled
35	JTMS	JTAG mode select from debugger Selected when in JTAG mode (default mode) JTAG mode is enabled after powerup or by forcing nRESET low Select Serial Wire mode using the ARM-defined protocol through a debugger Internal pull-up is enabled
	SWDIO	Serial Wire bidirectional data to/from debugger Enable Serial Wire mode (see JTMS description) Select Serial Wire mode using the ARM-defined protocol through a debugger Internal pull-up is enabled
36	PB0	Not required for programming
37	VDD_PADS	2.1-3.6 V pads supply
38	PC1	Not required for programming
39	VDD_MEM	1.8 V supply
40	JRST	JTAG reset input from debugger Selected when in JTAG mode (default mode, see JTMS description) Internal pull-up is enabled
41	PB7	Not required for programming
42	PB6	Not required for programming
43	PB5	Not required for programming
44	VDD_CORE	1.25 V digital core supply decoupling
45	VDD_PRE	1.8 V supply
46	VDD_SYNTH	1.8 V supply
47	OSCB	24 MHz crystal oscillator or left open when using external clock input on OSCA
48	OSCA	24 MHz crystal oscillator or external clock input. External clock input is a 1.8 V square wave.
49	GND	Ground supply pad in the bottom center of the package forms Pin 49. See Ember's various reference design documentation for PCB considerations.

3 Serial Wire and JTAG Interface

The EM35x includes a standard Serial Wire and JTAG (SWJ) Interface. Serial Wire is an ARM® standard, bi-directional, two-wire protocol designed to replace JTAG, and provides all the normal JTAG debug and test functionality. JTAG is a standard five-wire protocol providing debug and test functionality. In addition, the two Serial Wire signals (SWDIO and SWCLK) are overlaid on two of the JTAG signals (JTMS and JTCK). This arrangement keeps the design compact and allows tools to switch between Serial Wire and JTAG as needed, without changing pin connections. Therefore, a programmer interfacing with the EM35x may choose Serial Wire or JTAG as desired.

The key functionality that the SWJ provides is the ability to read and write 8, 16, and 32 bit quantities to absolute memory addresses in the chip. The programmer uses standard memory reads and writes to access a small selection of memory-mapped registers, as well as to load and interact with the RAM-based flashloader. All memory reads and writes used during production programming are 32 bit.

The details of the Serial Wire and JTAG protocols, as well as the debug port and access port (DAP) in the EM35x that implement the SWJ, are beyond the scope of this document. Since Serial Wire and JTAG are standard protocols, they are described in other ARM® documents. Ember does not document the details of the DAP since the DAP is standard ARM® IP and is too complex to duplicate in Ember documents. For further information on the SWJ, refer to the ARM® CoreSight™ Components Technical Reference Manual (DDI 0314C). This document is available from ARM's Infocenter website at http://infocenter.arm.com, and Ember support.

There are key details of the internal JTAG scan chain needed when interacting with the EM35x over JTAG. These details are specific to the EM35x, but the rest of the JTAG interaction is described in ARM® CoreSight™ Components Technical Reference Manual.

- 2 JTAG devices on the scan chain
- Total IR length is 8
- Device 0 IR length is 4
- Device 1 IR length is 4
- Device 1 is Ember's Cortex™-M3 with ID. Depending on the chip, this ID can be 0x069A962B, 0x269A962B, or 0x069AA62B.
- Ember's Cortex™-M3 is the only device you should interact with
- The DR length is a function of the value in IR (refer to ARM® documents)

Note: The maximum clock speed of the SWJ, as seen on the SWCLK/JTCK pin, is 2.5 MHz.

4 Memory Organization

Refer to the chip's data sheet and reference manual for complete details of the flash and RAM, including memory layout diagrams. The discussion here highlights specific aspects of the flash especially relevant to production programming.

4.1 Flash

The flash memory is divided into three separate blocks:

- Main Flash Block (MFB) – The MFB is the largest block of flash and holds the executable image being programmed into the device. The MFB begins at address 0x08000000 for all parts and its sizing varies between chips.
- Fixed Information Block (FIB) – The FIB stores manufacturing data that is fixed by Ember during chip production and is not writable.
- Customer Information Block (CIB) – The CIB stores customer data that is not executable, including manufacturing tokens. The lower addresses of the CIB are dedicated to special storage called option bytes. Option bytes are tied directly to certain chip behavior including read protection and write protection of flash. The CIB is 2 kB large on all chips and its addressing varies between chips.

The flashloader executing on chip will validate the addresses being programmed and indicate an invalid address error if the programmer attempts to modify invalid addresses. All addresses are referred to as full 32 bit absolute addresses. The smallest writable unit of flash is a half-word, 16 bits, and the smallest erasable unit of flash is a page, 2 kB. The erased state of flash is 0xFFFF. Flash can be read in 8, 16, and 32 bit quantities.

While it is possible to erase the MFB one page at a time and therefore erase only a subset of the MFB, it is recommended that production programming always starts by performing a mass erase of the MFB. This method ensures the MFB is put into a clean state and is faster than many single page erases across the flash.

Note: Mass erase erases only the MFB, and erases the entire MFB.

The option bytes at the bottom of the CIB have special behavior that differs from all other flash. The chip's flash interface requires that each option byte be written with a companion byte that is the inverse of the option byte. Therefore, a single option byte requires writing a full 16 bit, structured quantity. One of the option bytes configures flash read protection and the other option bytes control flash write protection. Refer to the appropriate data sheets and reference manuals for complete details of the option bytes.

Note: Because the option bytes configure flash read and write protection, special care must be taken when manipulating the option bytes. Read and write protection, as defined by the option bytes, is only updated when the nRESET pin is asserted and the chip is reset. Therefore, it is valid to erase and write the read and write protection bytes and then manipulate other flash as long as the nRESET pin is not asserted. Once read protection or write protection is set and nRESET is asserted, no further flash read or write operations should be attempted, except for disabling read or write protection (by reprogramming the option bytes).

Note: Read protection is only disabled by writing the value 0xA5 to the read protection byte. Any other value, including the erased state of 0xFF, will enable read protection.

Note: It is the programming image creator's responsibility to ensure any option bytes being programmed contain the correct inverse option byte value. If the inverse option byte is not correct, the programming operation will fail verification.

4.2 RAM

The RAM memory exists as a single block of memory. The bottom of the RAM is always address 0x20000000 for all parts and the sizing varies between chips. The flashloader s37 image file does not define a value for every byte in the RAM, but does define an address and value for every byte that must be programmed. Any byte in RAM that is not defined by the flashloader s37 image file may be left in any state.

While RAM is executable, it does not execute out of reset. Therefore, one of the programming steps defines how to capture the CPU and execute the flashloader after the flashloader has been written into RAM.

5 Description and Creation of Programming Image

5.1 File Format

Production programming uses the standard Intel HEX-32 file format. The normal development process for EM35x chips involves creating and programming images using the Motorola S-record file format, specifically the S37 file format. The s37 files are intended to hold applications, bootloaders, manufacturing data, and other information to be programmed during development. The s37 files, though, are not intended to hold a single image for an entire chip. For example, it is often the case that there is an s37 file for the bootloader, an s37 file for the application, and an s37 file for manufacturing data. Since production programming is primarily about installing a single, complete image with all the necessary code and information, the file format used is Intel HEX-32 format. While s37 and hex files are functionally the same (they simply define addresses and the data to be placed at those addresses) Ember has adopted the conceptual distinction that a single hex file contains a single, complete image often derived from multiple s37 files.

Since no special addressing or programming is used with EM35x chips, standard hex file formats are used and can be interpreted without extra knowledge. All bytes in flash have their own absolute 32 bit address. There is no strict requirement that a hex image defines every byte to be programmed on a chip. If a byte must be programmed and is not defined by the hex image, the byte should be programmed to the erased state, 0xFF. Any byte that is not programmed will be left in the erased state, 0xFF, when programming is complete, due to the fact that a mass erase is performed and the CIB is page erased. The only restriction is that the smallest writable quantity is 16 bits. The addresses of any data provided to the flashloader must be 16 bit aligned, and the data must be multiples of 16 bits.

5.2 Creation of a Programming Image

Programming images in the hex format are created from the standard em3xx_load.exe development tool, which is part of the ISA3 Utilities. The ISA3 Utilities are part of the EmberZNet 4.0 or later stack release. The em3xx_load.exe tool is very versatile and capable of generating hex files from a variety of sources. The sources include multiple s37 files, existing chips, and manual file patching operations. Ultimately, em3xx_load.exe allows the developer to merge a variety of sources into a final hex image and modify individual bytes in that image if necessary.

6 Programming Overview

The Programming Details section is organized in the general programming flow. The basic production programming flow is as follows:

1. Powerup and CPU Capture.
2. Install and Execute Flashloader Firmware,.
3. Disable Read/Write Protection.
4. Mass Erase MFB.
5. Program MFB.
6. Erase CIB.
7. Program CIB.
8. Final Verification.

The Erase CIB step is only required if write protection will be enabled. The Disable Read/Write Protection step will leave read protection disabled by writing 0xA5. Because disabling Read Protection is done by writing the option byte, the Erase CIB step is necessary before Read Protection can be programmed to a different value and enable write protection.

If the Program CIB step enables read or write protection, the new protection setting will not take effect until the chip is next reset by either asserting the nRESET pin or performing a standard power-on reset. Therefore, the programming flow does not have to worry about flash protection because protection will not take effect until production programming has completed.

7 Programming Details

This section details all of the individual steps that comprise the complete production programming process.

Note: The numerical values provided in these programming steps are values intrinsic to the EM35x chip and will never change. The names in all capital letters are values tied to a specific flashloader image and are defined in the header files that accompany the flashloader s37 image file.

7.1 Powerup and CPU Capture

1. Apply power while nRESET is held low.
2. Hold nRESET low and wait 10 µs after power has stabilized.
3. Do not release nRESET less than 30 µs after asserting nRESET.
4. While nRESET is low, logically attach the programmer to the SWJ by setting the CDBGPWRUPREQ and CSYSPWRUPREQ signals defined by ARM®.
5. Release nRESET and wait 100 µs.
6. Validate the SWJ communications path is operational by reading the Silicon ID at address 0x40004000 and verifying this value is 0x069A962B, 0x269A962B, or 0x069AA62B depending on the chip.
7. Perform a standard ARM® CortexTM-M3 CPU Halting Core Reset.

7.2 Install and Execute Flashloader Firmware

1. Write the value 0x00000307 to the address 0x40000018.
2. Write the entire flashloader firmware image to the chip's RAM. The flashloader image is a standard s37 file format indicating what bytes to write and the 32-bit absolute addresses to write to in the memory map.
3. Read the flashloader firmware image from RAM to verify the flashloader is installed properly.

4. Write the address of the bottom of RAM, 0x20000000, to the address 0xE000ED08.
5. Write the stack pointer value to the CPU's stack pointer register, R13, using a standard ARM® Cortex™ M3 CPU Stack Pointer Write. The stack pointer value is a literal, 32-bit number named STACK_POINTER_INIT, and is defined with the flashloader image.
6. Write the program counter value to the CPU's program counter register, R15, using a standard ARM® Cortex™ M3 CPU Program Counter Write. The program counter value is a literal, 32-bit number named PROGRAM_COUNTER_INIT, and is defined with the flashloader image.
7. Perform a standard ARM® Cortex™-M3 CPU Single Step.
8. Using a CPU Stack Pointer Write, rewrite the stack pointer with the STACK_POINTER_INIT literal.
9. Using a CPU Program Counter Write, rewrite the program counter with the PROGRAM_COUNTER_INIT literal.
10. Clear the flashloader's command and status shared memory by writing 0x00000000 to SHAREDMEM_COMMAND and SHAREDMEM_STATUS, defined with the flashloader image.
11. Perform a standard ARM® Cortex™ M3 CPU Run.
12. Read from SHAREDMEM_COMMAND, until this address contains the value COMMAND_IDLE.
13. Read from SHAREDMEM_STATUS, until this address contains the value STATUS_BOOTED.

7.3 Disable Read/Write Protection

1. Initiate disabling of read protection by writing the value COMMAND_DISABLE_RDPROT to SHAREDMEM_COMMAND. This inherently disables write protection as well.
2. Read from SHAREDMEM_COMMAND, until this address contains the value COMMAND_IDLE.
3. Read from SHAREDMEM_STATUS. If SHAREDMEM_STATUS does not contain the value STATUS_SUCCESS, the operation failed and the chip should be recorded as failed.
4. Repeat the step Powerup and CPU Capture.
5. Repeat the step Install and Execute Flashloader Firmware.

7.4 Mass Erase MFB

6. Initiate a mass erase by writing the value COMMAND_MASS_ERASE to SHAREDMEM_COMMAND.
7. Read from SHAREDMEM_COMMAND, until this address contains the value COMMAND_IDLE.
8. Read from SHAREDMEM_STATUS. If SHAREDMEM_STATUS does not contain the value STATUS_SUCCESS, the operation failed and the chip should be recorded as failed.

7.5 Program MFB

The target flash address must be 16 bit aligned. A minimum of 16 bits and a maximum of 2 kB of data may be written in a single program command.

Loop over all MFB data, programming a block, at most 2 kB, per program command:

1. Write the starting address for this block to SHAREDMEM_DATAADDRESS.
2. Write the data to be installed into flash to SHAREDMEM_DATABUFFER.
3. Write the size, in bytes, of the block of data to be programmed to SHAREDMEM_DATALENGTH. This size must be a multiple of 16 bits.
4. Initiate a write operation by writing the value COMMAND_PAGE_WRITE to SHAREDMEM_COMMAND.
5. Read from SHAREDMEM_COMMAND, until this address contains the value COMMAND_IDLE.
6. Read from SHAREDMEM_STATUS. If SHAREDMEM_STATUS does not contain the value STATUS_SUCCESS, the operation failed and the chip should be recorded as failed.

7.6 Erase CIB

The Erase CIB step is only required if write protection is to be enabled. The Disable Read/Write Protection step will leave read protection disabled by writing 0xA5. Because disabling Read Protection leaves the option byte programmed, the Erase CIB step is necessary before Read Protection can be programmed to a different value and enable write protection. Enabling read protection is best done by programming the option byte with the value 0x00 and its inverse 0xFF.

If the Erase CIB step is used, but read protection should remain disabled, then the read protection option byte must be written the disable value of 0xA5.

1. Write the base address for the CIB to SHAREDMEM_DATAADDRESS.
2. Initiate a page erase operation by writing the value COMMAND_PAGE_ERASE to SHAREDMEM_COMMAND.
3. Read from SHAREDMEM_STATUS. If SHAREDMEM_STATUS does not contain the value STATUS_SUCCESS, the operation failed and the chip should be recorded as failed.

7.7 Program CIB

Programming the CIB follows the same sequence of steps described in Program MFB.

7.8 Final Verification

After programming activities have completed, the programmer should directly readout all MFB and CIB addresses and verify those addresses match the data in the source hex image. Addresses not defined by the hex image should be verified as still being in the erased state, 0xFF.

At this point, production programming is complete.

8 Gang Programming

Because the production programming process described in this document relies on the SWJ interface and uses a lot of handshake style communication with the chip, it is impractical to share GPIO between EM35x chips and drive multiple chips from a single controller. Because the Serial Wire interface uses a single bidirectional data line for all communications, it is not possible to tie the SWDIO pin from multiple chips together, and therefore each chip must be controlled independently. It is possible to link multiple chips together in a standard JTAG scan chain configuration and access memory on each chip individually. In theory, it might be possible to tie the JTAG interface of multiple chips together, as long as the controller can still access the JTDO pin of each chip individually. Silicon Labs has not attempted this mode of operation, cannot say if it will or will not work, and cannot say if it is practical.

9 Serialization

The EM35x comes pre-programmed with an EUI64 when shipped from the factory. This Ember EUI64 exists in the FIB and is not modifiable. It may, however, be desirable for a customer to override the default EUI64 with their own unique serial numbers for each chip or set other per-chip unique manufacturing tokens. This override can be done by customizing the CIB programming so that the values to be written to flash are taken from some other source, such as a database, in addition to the programming image.

Details of pre-defined manufacturing tokens in the CIB are available in Ember application notes and the HAL source header. Refer to ”AN710: Bringing Up Custom Devices for the EM35x SoC or NCP Platform”, “AN708: Setting Manufacturing Certificates and Installation Codes for the EM35x SoC Coprocessor Platforms”, and hal/micro/cortexm3/token-manufacturing.h. These documents are available from the Silicon Labs website.

10 ARM® Cortex™-M3 CPU Manipulation Details

The following sections give an overview of the basic CPU manipulation necessary to accomplish production programming. The CPU manipulation is standard Cortex™-M3 functionality. Refer to the ARM Cortex™-M3, r1p1, Technical Reference Manual (DDI 0337D) for all the necessary details. Specifically, the Core Debug chapter and the Nested Vectored Interrupt Controller chapter of the Technical Reference Manual describe the registers needed and how to use them. This document is available from ARM's Infocenter website at http://infocenter.arm.com and Ember support.

10.1 Halting Core Reset

This process relies on the serial command line interface (CLI) to the Ember application framework. If the CLI is no longer supported or accessible on your network's Trust Center or incoming HAN device, please refer to the "Procedure for Production/Field Deployments" later in this section.

10.2 Stack Pointer Write

The Core Debug registers are used to write the Stack Pointer to a specific value.

10.3 Program Counter Write

The Core Debug registers are used to write the Program Counter to a specific value.

10.4 Single Step

The Core Debug registers are used to perform a single step of the CPU.

10.5 Run

The Core Debug registers are used to halt and run the CPU.

11 Flashloader Firmware Timing

Table 2 shows the timing of the flashloader commands. The timing was measured on-chip from the moment the flashloader left the idle state, COMMAND_IDLE, until the flashloader returned to the idle state.

Note: The timing of COMMAND_MASS_ERASE is for the worst case scenario. The worst case scenario is defined as the largest flash size: 512 kB.

Note: The timing of COMMAND_PAGE_WRITE is for the worst case scenario. The worst case scenario is defined as writing a full 2 kB of data where all data is 0x0000.

Table 2. Flashloader Firmware Timing

Flashloader Command	Execution Time of Command
COMMAND_MASS_ERASE	307 ms
COMMAND_PAGE_ERASE	21 ms
COMMAND_PAGE_WRITE	56 ms
COMMAND_DISABLE_RDPROT	21 ms

12 Flashloader Firmware and Interface

The following are the three files that define the flashloader firmware and the interface with the firmware. While these three files are added to the document here, the latest versions are available to download from the developer support site.

- flashloader-cmd-stat.h defines the COMMAND and STATUS values used when interacting with the flashloader.
- flashloader-em35x.h defines the memory addresses used when interacting with the flashloader.
- flashloader-em35x.s37 is Motorola S-record file that defines the actual flashloader firmware that is to be loaded into and executed from RAM.

The following is the flashloader-cmd-stat.h file:

```
/ *
 * Header file defining the command and status values needed in order
 * to interact with the flashloader.
 *
 * /
#ifndef __FLASHLOADER_CMD_STAT_H__
#define __FLASHLOADER_CMD_STAT_H__

#define COMMAND_IDLE             0xC0000001
#define COMMAND_PAGE_ERASE       0xC0000002
#define COMMAND_PAGE_WRITE       0xC0000003
#define COMMAND_DISABLE_RDPROT   0xC0000004
#define COMMAND_MASS_ERASE       0xC0000005
#define STATUS_BOOTED            0xA0000001
#define STATUS_INVALID_CMD       0xA0000002
#define STATUS_SUCCESS           0xA0000003
#define STATUS_BUSY              0xA0000004
#define STATUS_VERIFY_ERASE_FAIL 0xA0000005
#define STATUS_PROG_FAIL         0xA0000006
#define STATUS_VERIFY_PROG_FAIL  0xA0000007
#define STATUS_BAD_ADDR_OR_LEN   0xA0000008

#endif //__FLASHLOADER_CMD_STAT_H__
```

The following is the flashloader-em35x.h file:

```
/*
 * Header file for flashloader-em35x.s37, defining the values
 * needed in order to interface with this flashloader.
 *
 * NOTE: This file is autogenerated.  Do NOT edit this file directly.
 *
 */
#ifndef __FLASHLOADER_SHAREDMEM_H__
#define __FLASHLOADER_SHAREDMEM_H__

#define CHIP_NAME             "EM35X"
#define PROGRAM_COUNTER_INIT  0x20000220
#define STACK_POINTER_INIT    0x20000200
#define SHAREDMEM_COMMAND     0x200017f0
#define SHAREDMEM_DATAADDRESS 0x200017f8
#define SHAREDMEM_DATALENGTH  0x200017fc
```

```
#define SHAREDMEM_STATUS      0x200017f4
#define SHAREDMEM_DATABUFFER  0x20001800

#endif //__FLASHLOADER_SHAREDMEM_H__
```

The following is the flashloader-em35x.s37 file:

```
S0180000656D3335782D666C6173686C6F616465722E73333778
S31520000200000020020210200209502002095020020F5
S31520000210A70E01010000000000000000000000000001
S3152000022080B50D48016821F4607141F4C071016008
S3152000023001684 1F0070101600848094901 6072B66A
```

```
S31520000240084801684908490001 6000F08BFCBDE8B8
```

```
S315200004E0DFF8941509685C22DFF8903502FB0131AC
S315200004F0AB6813EB4703D1F838C063450ED3C96BFD
S31520000500AB6813EB4702914208D339F81710C9B2EA
S31520000510F8B2FFF785FF4045CED033E039F8171003
S3152000052031802168C907FCD420212160216811F07F
S31520000530140FC1D0102121600421216041688907 50
S31520000540FCD40021016080206060DFF8440519E0BA
S3152000055041688907FCD40021016080206060AE6874
S31520000560002701E0781C87B2E868B7EB500F08D265
S3152000057036F8020B39F817108842F3D0DFF8140545
S3152000058000E04046BDE8F28380B5FFF7DFFEDFF8E6
S31520000590E4044FF404714160DFF8DC1409685C223E
S315200005A0DFF8D83402FB0131896B4AF2A5520A8062
S315200005B00168C907FCD420210160DFF8B414026861
S315200005C012F0140F0DD01022026004220260802245
S315200005D0426048688007FCD400200860DFF8B00439
S315200005E002BD4A689207FCD400220A60802141603D
S315200005F0DFF8A40402BD10B5FFF7A8FEDFF87414D7
S3152000060004204860442048600868C007FCD42020A5
S3152000061000860DFF85C0442689207FCD4002202607E
S3152000062080204860DFF85C040A68D20602D51022D2
S3152000063000A6010BDDFF8401409685C22DFF83C34FC
S3152000064002FB01318A68CB685B1C991A002300E003
S315200006505B1CB3EB910F05D252F8044B14F1010F3A
S31520000660F6D010BDDFF8300410BD2DE9F041DFF8DB
S3152000067001044FFF7C0FF6060DFF81C546068A84292
S3152000068030D10020DFF81414A26821F81020401C75
S31520000690B0F5806FF8D35C26DFF8DC73DFF8DC83F7
S315200006A0386806FB00804169E16080680BE0FFF74F
S315200006B0F1FE60606068A84214D1A068396806FB24
S315200006C0018149690818A060386806FB0080C06867
S315200006D0A1688842EBD2FFF78EFF60606068A8426F
S315200006E008BF6560BDE8F0812DE9F84F0D469046BC
S315200006F004464FF6FF7A51460026C1460027DFF80A
S3152000070009CB304E0385D00F01BFA01467F1CAF4223
S3152000071000ED24F45F6D12BF81610761CC144514601
S31520000720B6F5806FEED9DFF85803DFF8741305E0CD
S315200007302BF81610DFF84803DFF85C134160BDE89C
S31520000740F18FDFF860130A68DFF85C331B68DB077C
S3152000075058BF4008032809D34B6D43F001034B656E
S3152000076000B689B1A1B031B0B8342F9D3704770B58A
S3152000077004460D46CD4E0820F066F06E000701D4E3
S31520000780C74870BD3460F06E40F4147000F037F83E
S31520000790F06E40F48070F066A10705D5280440F479
S315200007A07F4040F0FF0003E045F07F4040F47F00AB
S315200007B030670C20FFF7C5FFF06E40F02000F06692
S315200007C01620FFF7BEFFF06E20F02000F066012015
S315200007D0FFF7B7FFF06E20F4007000F010F8F06E0F
S315200007E020F4A870F0660320FFF7ABFF0020F06628
S315200007F02088A84201D0A74870BDA74870BDF066E2
S31520000800007209EE738B5A94C0820E066E06E000771
S3152000081001D4A34832BD9A48816821608568E06E7C
S31520000820 40F04000E066E06E40F40160E066FFF7CD
S31520000830E7FFE06E40F48070E06645F2F050FFF787
S3152000084080FFE06E20F40060E066FFF7D9FFE06EDF
S3152000085020F4A870E0660320FFF773FF0020E0660F
S315200008605C22864B00E0401C80B28349096802FB6B
```

```
S315200008700131496AB0EB910F06D255F8041B11F1EC
S31520000880010FF0D07F4832BD834832BD884B1B68AC
S315200008909B070BD400F296700378002B01BF437898
S315200008A08B42C078904201D10120704700207047CA
S315200008B00024714D04E0761CB6B2032E09D3641CC5
S315200008C07C48007884421BDA00265C206B4900FBBA
S315200008D0041707EB8600006D4FF040210968884217
S315200008E0E9D107F24D00817801B12C6097F84D20AF
S315200008F04178B869FFF7CAFF0028DCD02C60DEE714
S315200009005F4E6D4F286810F1010F03D137606B4899
S3152000091070 60FEE74FF0000BDFF8808100F0D6F81C
S315200009203760674870 60DFF8709102E05748706062
S3152000093037603068B842FCD062487060B068D6F83C
S315200009400CA05B490968890705D531684E4A914252
S315200009501CBF0021316029685C22484B02FB013113
S3152000096032 68594BD21A0AD0521E27D0521E62D054
S315200009705 21E6AD0521E6BD0521E6CD0D6E7C207CA
S315200009801AD48A68904224BFCA68824203D30220BE
S31520000990FFF7F1FCCBE78A6A904224BFCA6A8242FB
S315200009A001D30320F4E78A69904224BFC9698142B2
S315200009B002D3FFF727FFBAE73948B8E700F001026C
S315200009C00AF001031A432FD10AEB00028B689842E2
S315200009D022BFCB685B1C934206D28B6A984222BF09
S315200009E0CB6A5B1C934202D3FFF754FD1DE08B6953
S315200009F0984222BFC969491C914215D304464FF03B
S31520000A00000BBBEB5A0F11D238F81B102046FFF70C
S31520000A10AEFE70607068484502BFA41C0BF1010051
S31520000A201FFA80FBEDD001E01D4870604FF0000BEF
S31520000A3000F04CF87CE7FFF7A7FD0320FFF79BFCAF
S31520000A40FFF7A2FDC6F8049072E7FFF7D4FD6EE724
S31520000A50FFF70BFE6CE7D8F800205146FFF744FE5F
S31520000A6066E70000048000402301674 5AB89EFCD8F
S31520000A702C4000400C800040EC170020080C002081
S31520000A80F0170020050000A0040000C0020000A00E
S31520000A90060000A0070000A0030000A00018002008
S31520000AA0080000A0382000401C4000401480004070
S31520000AB01C8000408012002001000 0C0090000A018
S31520000AC0010000A0040000A0020000C04FF6FF7045
S31520000AD028F81B000BF1010BBBF5806FF8DB704784
S31520000AE0401810F0030308D0C91A1FD3DB0748BFEC
S31520000AF000F8012D28BF20F8022D130030B414462B
S31520000B001546103928BF20E93C00FAD8490728BFE6
S31520000B1020E90C0048BF40F8042D890028BF20F8A2
S31520000B20022D48BF00F8012D30BC7047C91818BFE8
S31520000B3000F8012DCB0728BF00F8012D70470000D3
S31520000B400A0242EA11214840C0F30311484080EAD4
S31520000B5000304101C9B20206D20E41EA02214840C4
S31520000B6080B27047DFF8080051210160704700000D
S31520000B70008000040454D333531000000454D33356A
S31520000B8058000000454D33353600000454D3335BD
S31520000B9037000000454D33353831000454D33359B
S31520000BA03832000454D333538350000454D333554
S31520000BB03836000454D333538370000454D33353E
S31520000BC03838000454D333539000000454D333562
S31520000BD03931000454D333539320000454D333526
S31520000BE03935000454D333539360000454D33350E
S31520000BF03937000454D333539380000534B5936C7
```

```
S31520000C0036313037000000000740B00207C0B0020AA
S31520000C1000000008FFFF0108000002000008000095
S31520000C2000000408FF07040800080000000008000070
S31520000C3000080408FF0F040800080000000008000050
S31520000C40000804080F080408000000020FF2F0020D9
S31520000C500030000005010101 2B969A062B969A2654
S31520000C6000000000840B00207C0B00200000000800
S31520000C70FFFF020800000300000800000000004082F
S31520000C80FF07040800080000000080000000008040808
S31520000C90FF0F0408000800000000800000000080408F0
S31520000CA00F0804080000002 0FF2F0020003000005D
S31520000CB0110F01002B969A062B969A26000000000B
S31520000CC08C0B00207C0B002000000008FFFF020890
S31520000CD000000300000800000000000408FF070408C5
S31520000CE00000800000000800000000080408FF0F0408A0
S31520000CF000008000000008000000000804080F08040887
S31520000D0000000020FF2F002000300000302010019
S31520000D102B969A062B969A2600000000940B00200C
S31520000D207C0B002000000008FFFF030800000400E1
S31520000D3000008000000000808FF0708080008000057
S31520000D400000800000000080808FF0F08080008000037
S31520000D500008000000080808 0F080808000000 2006
S31520000D60FF7F0020008000000A0602012BA69A06BB
S31520000D7000000000000000000009C0B00207C0B0020DF
S31520000D8000000008FFFF03080000040000008000020
S31520000D9000000808FF070808000800000000080000F7
S31520000DA000080808FF0F080800080000000080000D7
S31520000DB0000808080F08080800000020FF7F002010
S31520000DC00008000000B0702002BA69A0600000000F8
S31520000DD000000000A40B00207C0B0020000000086F
S31520000DE0FFFF07080000008000008000000000808B0
S31520000DF0FF07080800080000000800000000808088F
S31520000E00FF0F08080008000000008000000008080876
S31520000E100F08080800000020FF7F0020008000047
S31520000E200C0802002BA69A06000000000000000015
S31520000E30AC0B00207C0B002000000008FFFF0708F9
S31520000E40000000800000800000000000808FF07080846
S31520000E500008000000008000000080808FF0F080826
S31520000E600008000000008000000080808 0F0808080D
S31520000E7000000020FF7F002000800000 0D090200F6
S31520000E802BA69A06000000000000000000B40B0020EC
S31520000E907C0B002000000008FFFF070800000080068
S31520000EA0000080000000000808FF070808000800000E6
S31520000EB000008000000000808 08FF0F080800080000C6
S31520000EC00000800000000808080F0808080000002095
S31520000ED0FFFF002000000001000E0A02002BA69A0642
S31520000EE000000000000000000BC0B00207C0B00204E
S31520000EF000000008FFFF07080000080000080000A7
S31520000F000000000808FF070808000800000000 8000085
S31520000F1000080808FF0F0808000800000000 8000065
S31520000F20000808080F08080800000020FFFF00201E
S31520000F300000001000803020 02BA69A060000000000C
S31520000F4000000000C40B00207C0B00200000000 8DD
S31520000F50FFFF0708000000800000800000000008083E
S31520000F60FF0708080008000000008000000000808081D
S31520000F70FF0F08080008000000008000000000808080 5
S31520000F800F08080800000020FFFF002000000100D5
```

```
S31520000F90090402002BA69A060000000000000000AB
S31520000FA0CC0B00207C0B002000000008FFFF03086C
S31520000FB0000004000008000000000808FF070808D9
S31520000FC0000008000008000000080808FF0F0808B5
S31520000FD00000080000080000000808080F0808089C
S31520000FE000000020FF7F0020008000001310020078
S31520000FF02BA69A060000000000000000D40B00205B
S315200010007C0B002000000008FFFF030800000400FE
S315200010100008000000000808FF0708080000080074
S315200010200008000000080808FF0F08080000080054
S315200010300008000000080808 0F080808000000202 3
S31520001040FF7F00200080000014110200 2BA69A06C4
S315200010500000000000000000DC0B00207C0B0020BC
S3152000106000000008FFFF0708000008000008000035
S3152000107000000808FF070808000008000008000014
S3152000108000080808FF0F0808000008000008000 0F4
S31520001090000808080F08080800000020FF7F00202D
S315200010A000800000151202002BA69A060000000000
S315200010B000000000E40B00207C0B0020000000084C
S315200010C0FFFF07080000080000080000000008 08CD
S315200010D0FF0708080000080000080000000808 08AC
S315200010E0FF0F080800000800000800000008080894
S315200010F00F08080800000020FF7F00200080000065
S31520001100161302002BA69A0600000000000000001D
S31520001110EC0B00207C0B002000000008FFFF0708D6
S3152000112000000800000800000000 0808FF07080863
S3152000113000000800000800000008 0808FF0F080843
S31520001140000008000008000000080808 0F0808082A
S3152000115000000020FFFF0020000001001014020004
S315200011602BA69A060000000000000000F40B0020C9
S315200011707C0B002000000008FFFF07080000080085
S315200011800008000000000808FF0708080000080003
S315200011900008000000080808FF0F080800000800E3
S315200011A000080000000808080F08080800000020B2
S315200011B0FFFF002000000100121502002BA69A0650
S315200011C00000000000000000FC0B00207C0B00202B
S315200011D000000008FFFF070800000800000800 00C4
S315200011E000000808FF0708080000080000080000A3
S315200011F000080808FF0F08080000080000080000 83
S31520001200000808080F08080800000020FFFF00203B
S31520001210000001000F0B02002BA69A06000000001A
S3152000122000000000000000000000000000000000 98
S3152000123000000000000000000000000000000000 88
S3152000124000000000000000000000000000000000 78
S3152000125000000000000000000000000000000000 68
S3152000126000000000000000000000000000000000 58
S3152000127000000000000000000000000000000000 48
S30D2000128011000000FFFFFFFF33
S70520000221B7
```

13 After Reading this Document

If you have questions or require assistance with the procedures described in this document, contact Silicon Labs customer support. The Silicon Labs Community, located at http://community.silabs.com, provides a wide array of hardware and software documentation such as FAQs, reference designs, user guides, application notes, and open discussion of technical questions concerning Silicon Labs products. To gain access to the latest software and tools for the EM35xx platforms and contact Silicon Labs technical support directly, please visit the Silicon Labs Technical Support portal at https://siliconlabs.force.com.

.

CONTACT INFORMATION

Silicon Laboratories Inc.

400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

Please visit the Silicon Labs Technical Support web page for ZigBee products:
www.silabs.com/zigbee-support and register to submit a technical support request