File Format Overview

File Format Overview

File Format Overview

Simplicity Commander works with different file formats: .bin, .s37, .ebl, .gbl, and .hex. Each file format serves a slightly different purpose. The file formats supported by Simplicity Commander are summarized below.

Motorola S-record (s37) File Format

Silicon Labs uses the Simplicity Studio as its Integrated Development Environment (IDE) and leverages the IAR Embedded Workbench for ARM platforms. This tool combination produces Motorola S-record files, s37 specifically, as its output. For more information on Motorola S-record file format, see https://en.wikipedia.org/wiki/SREC_(file_format). In Silicon Labs development, an s37 file contains programming data about the built firmware and generally only represents a single piece of firmware—application firmware or bootloader firmware—but not both. An application image in s37 format can be loaded into a supported target device using the Simplicity Commander `flash` command. The s37 format can represent any combination of any byte of flash in the device. The Simplicity Commander `convert` command can also be used to read multiple s37 files and hex files; output an s37 file for combining multiple files into a single file; and modify individual bytes of a file.

Update Image File Formats

An update image file provides an efficient and fault-tolerant image format for use with Silicon Labs bootloaders to update an application without the need for special programming devices. Two image formats are supported: Gecko Bootloader (GBL) format for use with the Silicon Labs Gecko Bootloader introduced for use with EFR32 devices and Ember Bootloader (EBL) format for use with legacy Ember bootloaders. See *UG103.6: Application Development Fundamentals: Bootloading* for more details about these image file formats and bootloader use with different platforms.

Update image files are generated by the Simplicity Commander `gbl create` or `ebl create` command. These formats can only represent firmware images; they cannot be used to capture Simulated EEPROM token data (as described by *AN703: Using Simulated EEPROM Version 1 and Version 2 for the EM35x and EFR32 Series 1 SoC Platforms*). GBL upgrade files may contain data that gets flashed outside the main flash.

Bootloaders can receive an update image file either over-the-air (OTA) or via a supported peripheral interface, such as a serial port, and reprogram the flash in place. Update image files are generally used in later stage development and for upgrading manufactured devices in the field.

During development, bootloaders should be loaded onto the device using the .s37 or .hex file format. If the Gecko Bootloader with support for in-field bootloader upgrades is used, it is possible to perform a bootloader upgrade using a GBL update image. For other bootloaders or file formats, do not attempt to load a bootloader image onto the device as an update image.

Intel HEX-32 File Format

Production programming uses the standard Intel HEX-32 file format. The normal development process for EFR32 chips involves creating and programming images using the s37 and ebl file formats. The s37 and ebl files are intended to hold applications, bootloaders, manufacturing data, and other information to be programmed during development. The s37 and ebl 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. Because 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—Silicon Labs has adopted the conceptual distinction that a single hex file contains a single,

complete image often derived from multiple s37 files. You can use the Simplicity Commander `convert` command to read multiple hex files and s37 files; output a hex file for combining multiple files into a single file; and modify individual bytes of a file.

> Note: Simplicity Commander is capable of working identically with s37 and hex files. All functionality that can be performed with s37 files can be performed with hex files. Ultimately, with respect to production programming, Simplicity Commander `flash` command allows the developer to load a variety of sources onto a physical chip. The `convert` command can be used to merge a variety of sources into a final image file and modify individual bytes in that image if necessary.

The following table summarizes the inputs and outputs for the different file formats used by Simplicity Commander.

	Inputs					Outputs					
	ebl	s37	hex	bin	chip	ebl	s37	hex	bin	chip	rps
flash		X	X	X						X	
readmem					X		X	X	X		
convert		X	X	X			X	X	X		
ebl create		X	X	X		X					
ebl parse	X						X	X	X		
rps create		X	X	X							X
mfg917		X	X	X			X	X	X		