The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN203980

FM4 S6E2Cx Series Over The Air Update 32-Bit Microcontroller With Embedded Dual Flash

Target products: S6E2Cx series

This document explains the use of reference software "Over the Air Update with Embedded Dual Flash Memory" (OTA-D).

Contents

Introduction

Organization of This Document

Please read this document in conjunction with "SK-FM4-216-ETHERNET Wi-Fi Sample Kit Setup Manual."

Before reading this document, we recommend reading the following documents.

Reference Documents

Flash memory programming manual

http://www.spansion.com/downloads/s6e2cc_mn709-00007-e.pdf

S6E2Cx series data sheet

http://www.spansion.com/downloads/s6e2cc_ds709-00009-e.pdf

1 Overview

This document explains the use of reference software "Over the Air Update with Embedded Dual Flash Memory" (OTA-D). This OTA-D updates not only user software but also wireless communication software.

Wireless communication related software and other user software is located in one bank of the embedded flash memory. The latest update target software is downloaded from the server to the other bank of the embedded flash memory.

After the completion of the download, the MCU is reset. The target software update is finished by executing the latest target software from the other bank of embedded flash memory.

Note:

- Do not use the OTA-D reference software for anything other than your reference for developing your applications.

This software is subject to change without notice. This reference software shows one of the general uses with the SK-FM4-216-ETHERNET.

- Cypress may in no way be held responsible for damage occurring as a result of the use of this software.

1.1 Background

Previously, software wasn't updated after software embedded in products was shipped to the market. If any software update was required, the application itself was recalled and updated locally using wired connectivity. Because of the increasing diversity of software used in applications, and the decrease of the development time, updating the software embedded in products in the field is necessary. Over The Air Update software solves the problem of how to update software already being used in the field.

2 OTA-D Reference Software

The OTA-D reference software uses the "dual flash mode," which allows accessing Flash Macro #0 and Flash Macro #1 independently, and the "re-map" function which switches the used Flash Macro as main Flash. Their functions are built into the MCU in the S6E2Cx series of MCUs.

- For further information about the MCU future of re-map and the dual flash mode, refer to the Flash Memory Programming Manual mentioned under References at the beginning of this document.

- This document targets OTA-D Reference Software version 1.0

2.1 Basic Functionality

The OTA-D reference software consists of four sections:

- Bootloader – software that runs initially
- Information data – user software metadata, such as the revision history
- OTA-D – software library for Over the Air Update.
- User software – software prepared by the users. In this document, the programming of LCD display control is used as example user software.

The OTA-D reference software, the SK-FM4-216-ETHERNET, the Wi-Fi board, and the prepared server application for OTA-D demonstrate the Over The Air Update function.

The behavior of the OTA-D reference software is shown in Figure 1. The work flow for writing to the embedded flash memory and address remapping of flash memory are shown in Figure 2

Figure 1. OTA-D Reference Software Processing Flow

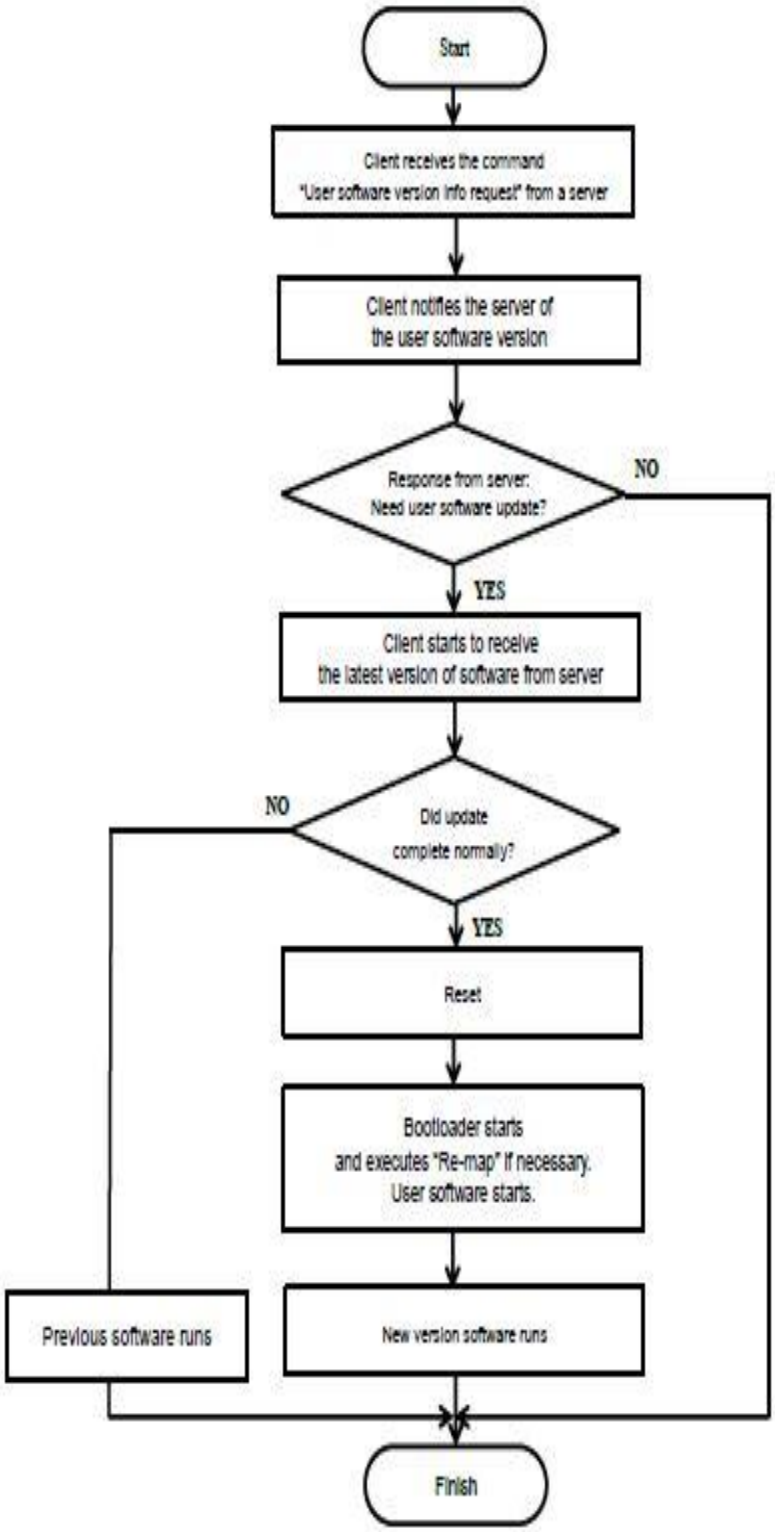


Figure 2. OTA-D Reference Software Physical Memory Address Map

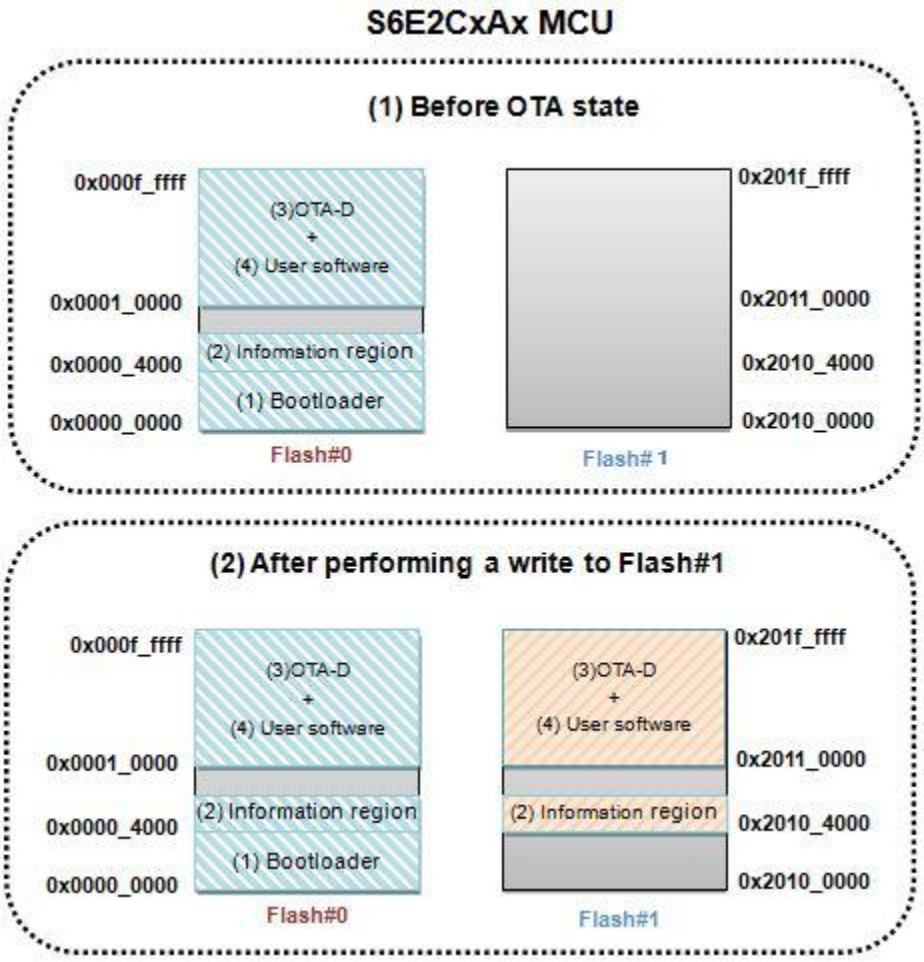
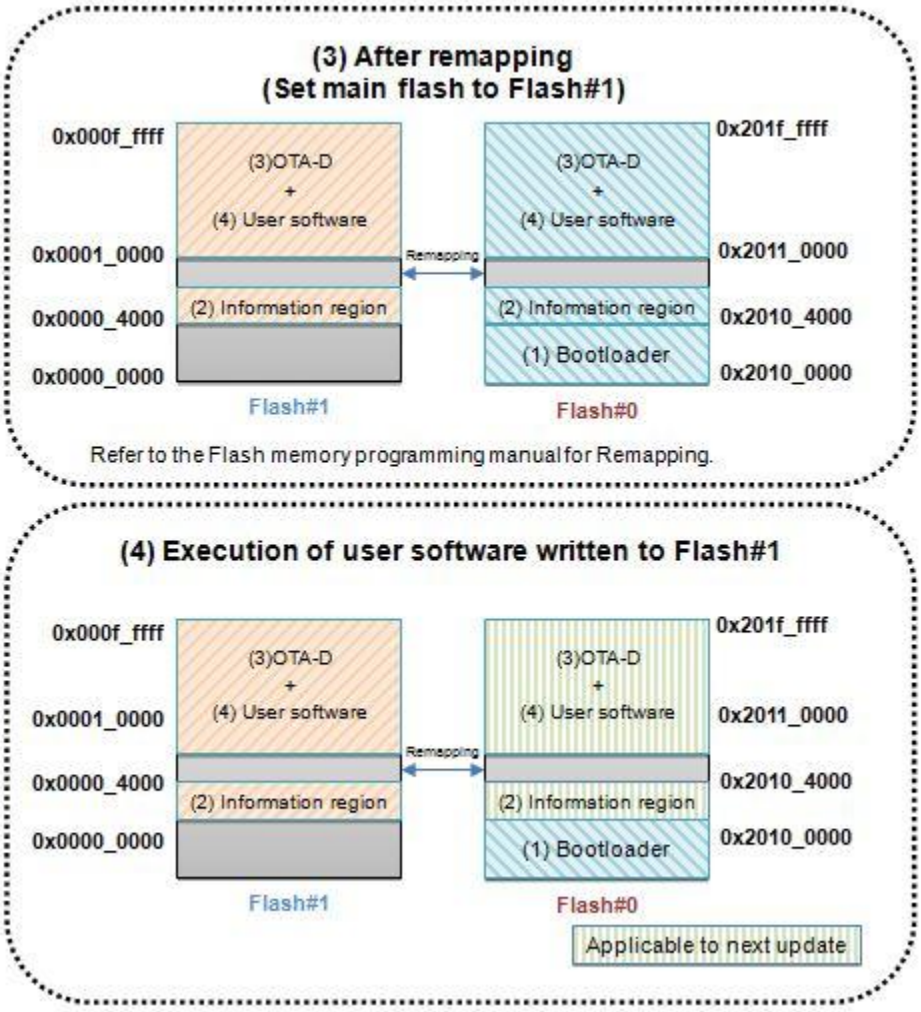


Figure 3 OTA-D Reference Software Physical Memory Address Map



2.1.1 Bootloader

The bootloader is always located in the Flash#0 bank. It is copied to RAM at MCU startup and runs first in RAM.

It assesses the latest version by using the information data of Flash#0 and Flash#1 and changes Flash#0 to Flash#1 with the re-map function in the S6E2CxAx (if the latest user software is stored in Flash#1).

After the bootloader runs, it jumps to the user software (0x0001_0000) from RAM.

The user software is always run at MCU embedded flash memory (0x0001_0000-0x000f_ffff).

See Figure 3 for a re-map image

2.1.2 Information Region

The information region stores user software information such as the version and level.

In the OTA-D reference software, the bootloader uses this information for managing the version of the user software.

2.1.3 OTA-D

The OTA-D is the library that is linked in user software.

See “Operation Sequence between OTA-D Server and Client”.

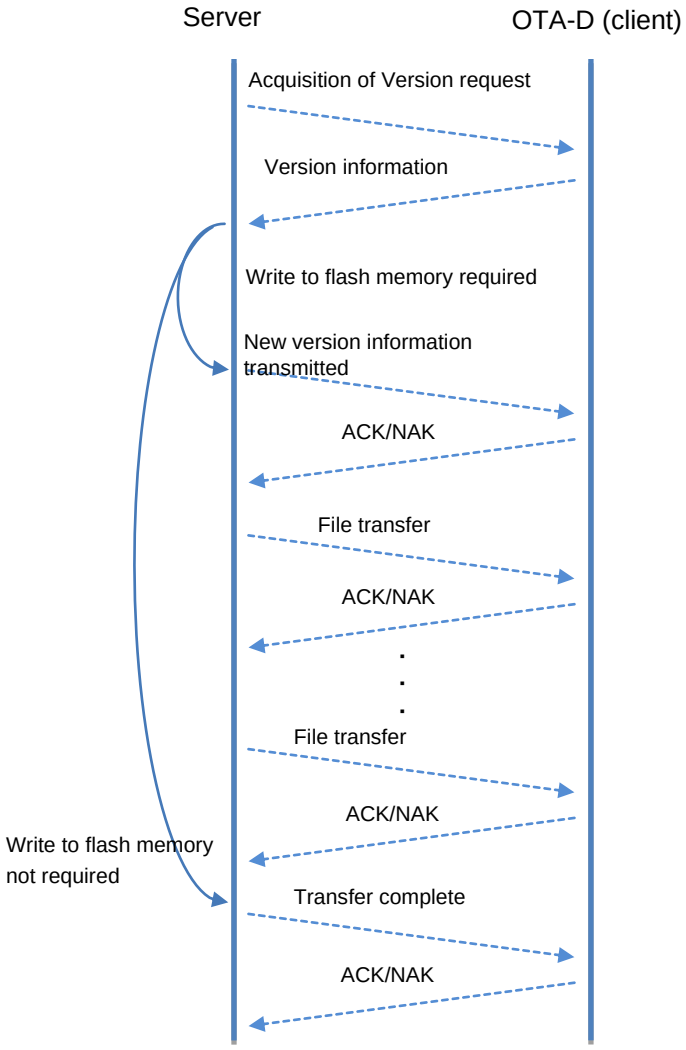
2.1.4 User Software

The user software is compiled with the OTA-D library.

In the OTA-D reference software, user software has an OTA-D library, Wi-Fi function, real-time OS, and LCD demo function.

They are all updated at the same time.

2.2 Operation Sequence between OTA-D Server and Client



2.3 Function Details

This section explains the OTA-D reference software functions and interface.

2.3.1 Bootloader

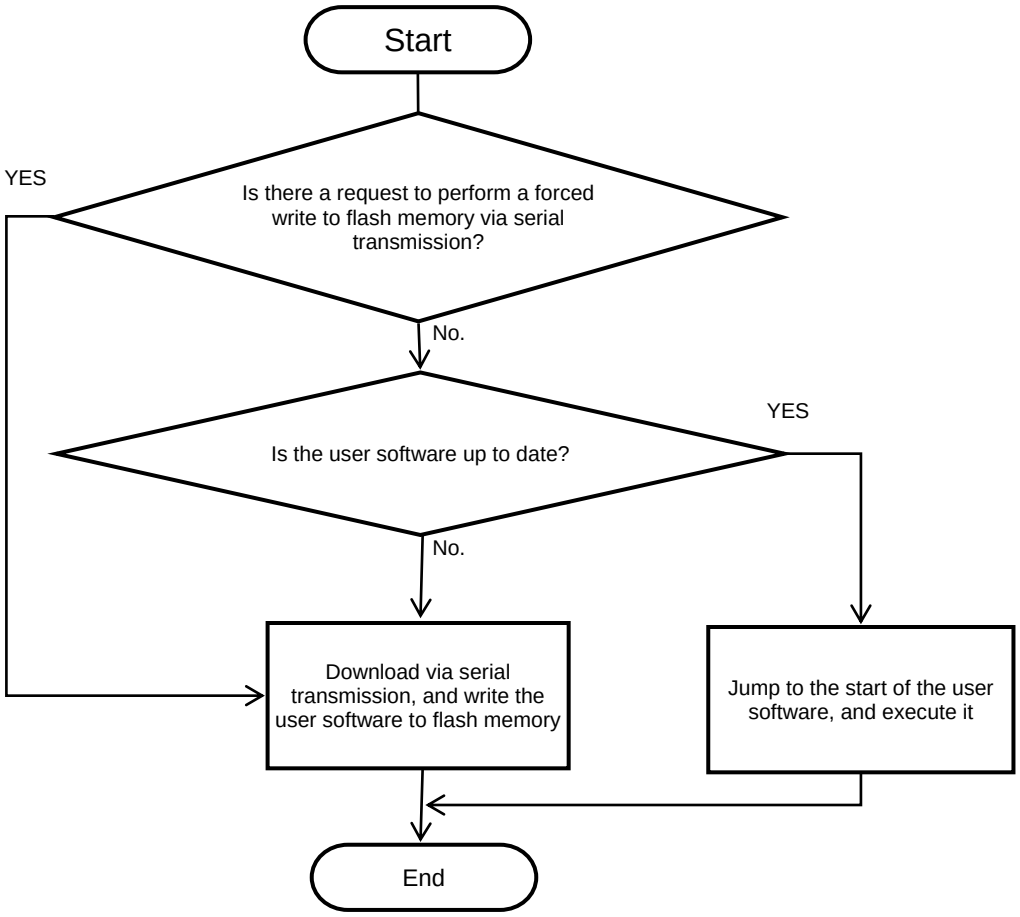
The bootloader runs after the MCU is reset. It contains the following functions.

- Force a write to MCU embedded flash memory via serial communication.
- Indicate which flash bank stores the latest data by checking information in each flash bank.
- Select the latest user software, jump to the first memory address (0x0001_0000) of target user software, and start the software.
- Download new user software via serial transmission if the user software needs to be updated.

The bootloader itself cannot support wireless connectivity. Serial transmission is required to write user software to embedded flash memory. See Write User Software to Flash Memory via Serial Transmission.

Figure 4 shows the bootloader operation flow.

Figure 4. Bootloader Operation Flow



2.3.2 Information Region

User software information is saved in the information region. See Table **1**for more information about data type.

Table 1. Detail of Data Type Information in the Information Region

0x0000_4000 / 0x2010_4000	+3	+2	+1	+0
	app load count		check sum	

check sum: Information region check sum

When writing to the user software flash memory is complete, calculate the checksum and write the result.

Check sum calculation method: check sum = ~SUM + 1 + MAGIC_NUM

(SUM is the result of adding information (6 byte) in the information region other than the checksum, in signed half-word format. MAGIC_NUM is a magic number, defined as 0x1C5A.)

app-load count: Download count of user software

After the completion of writing user software to the flash memory,

in case of writing to 0x0000_4002, write the value of *0x2010_4002+1.

in case of writing to 0x2010_4002, write the value of *0x0000_4002+1.

0x0000_4000 / 0x2010_4000	+3	+2	+1	+0
	version_low		version_high	

version_high: Stores the version (higher 16 bits) of the user software.

version_low: Stores the version (lower 16 bits) of the user software.

Writes the version explained in (2) Sending version information

The concept of “version_high” and “version_low” is used on this reference software as follows. “version_high” selects the version with the numerically larger value, regardless of the “version_low” value.

When “version_high” are the same, “version_low” selects the version with the numerically larger version_low value.

When “version_low” and “version_high” are the same, selects the version with the numerically larger app load count value.

E.g., Control of version number 2.08

version_high=2, version_low=08

2.3.3 OTA-D Library

ota_version_section

"ota_version_section" is the name of the code section. It contains the version data of the OTA-D library.

Specifically, it defines the flash memory address on the embedded MCU for locating the "ota_version_section". It is defined in s6e2ccAH.icf, which is in the directory:

"project\spansion_s6e2cx_8189em_with_ota\EWARM-RELEASE\config\" as follows

```
define symbol __OTA_LIB_version__start__ = 0x00090000;
place at address mem:__OTA_LIB_version__start__ { section ota_version_section };
```

0x00090000 is the start address of the memory area.

ota_section

"ota_section" is the name of the code section. It contains the OTA-D library.

Specifically, it defines the flash memory address on the embedded MCU for locating the "ota_section". It is defined in s6e2ccAH.icf, which is in the directory:

"project\spansion_s6e2cx_8189em_with_ota\EWARM-RELEASE\config\" as follows

```
define symbol __OTA_LIB_start__ = 0x00090010;
place at address mem:__OTA_LIB_start__ { section ota_section };
```

0x00090010 is the start address of the memory area.

ota_version

ota_version is an "ota_version_t" type variable in the OTA-D library. It contains the version data for the OTA-D library.

Table 2. "ota_version_t" Type Structure List

Type Name	Definition	Data Type	Value
ota_version_t	version_high	uint16_t	OTA-D library version (higher 16 bits)
	version_low	uint16_t	OTA-D library level (lower 16 bits)

ota_update_process

- C programmable language interface

int errorcode = ota_update_process(ota_parameters_t * ota_parameters)

- Parameters
  - Input: ota_parameters server information received from user software
  - Output: errorcode processing results returned by OTA update function

Table 3. ota_parameters_t Type Structure List

Type Name	Definition	Data Type	Value
ota_parameters_t	ipaddress	unsigned char *	OTA server IP address
	port	unsigned short	OTA server port number
	ssl_flg	unsigned short	Flag to set whether to use SSL 1: Use SSL 0: Do not use SSL

Table 4. OTA Update Function Execution Results List

Macro Name	Return Value	Value
OTA_UPDATE_SUCESS	0	OTA updated successfully.
OTA_NETWORK_ERROR	-1	Network error occurred.
OTA_VERSION_ERROR	-2	Already latest version, so update unnecessary.
OTA_OTHER_ERROR	-3	Other error

This function updates user software by getting IP addresses and port number of the OTA server through “ota_parameter”. The return value from this function confirms the results of OTA update operations.

Command Packet

This section explains the command packets used to control the OTA-D.

The meaning of each piece of data is explained below. The () at the end of each code unit indicate set points.

Table 5. Data of Command Packet

Name	Byte Count	Details
STX	1	(0x02) Used for Header of each command packet
LEN	2	(all byte counts for IDX, CMD, DATA) Express Command packet length
IDX	2	(0～65535) Express Data packet number
CMD	1	Express Command type -Command line sent from server to client (0x05) Acquisition command for user software version information running on the client (0x01)Transmission command for user software version information managed at server. (0x02)Transmission command for sending the latest user software (0x04)Transmission complete notification to client -Command line sent from client to server (0x05) Return version info : return version information of the user software running on the client. (0x06) Response acknowledgment to server as no error. (0x15) Response negative acknowledgment to server as error happened
DATA	n	Used for express data (specific form is depend on CMD type)
CS	1	Check sum (Two’s complement of IDX, CMD, DATA: CS = ~(IDX+CMD+DATA) + 1)
ETX	1	(0x03) Indicates the last data of the command packet

Command Types

This section explains the types of commands.

See Table 5 for DATA and IDX in the following explanation.

1. Acquisition of User Software Version (CMD=0x05)

This command is used for the server to get the version information for the software running on the client.

That information is stored in the DATA area at the transmission period as shown below.

Name	Byte Count	Value
-	1	Unused (0x00)

This information is also used for a response to an acquisition command to return the user software version running on the client. That information is stored in the DATA area at OTA-D as shown below.

Name	Byte Count	Value
version_high	2	User software version on client side (higher 16 bits)
version_low	2	User software version on client side (lower 16 bits)

2. Transmission of User Software Version (CMD=0x01)

Transmission command for user software version information managed on the server.

The data is stored in the DATA area at transmission as shown below.

Name	Byte Count	Value
version_high	2	User software version on server side (higher 16 bits)
version_low	2	User software version on server side (lower 16 bits)

This data is stored in the DATA area if the OTA-D's response is acknowledged.

Name	Byte Count	Value
ErrorCode	1	Unused (Don't care)

The data is stored in the DATA area if the OTA-D's response is not acknowledged.

Name	Byte Count	Value
ErrorCode	1	Error code -1: Data package error

3. Transmission of the Latest User Software (CMD=0x02)

This transmission command sends the latest user software from the server to the client.

The data is stored in DATA at transmission as shown below. IDX starts at 1, and increments a variable by 1 with every packet. When it exceeds 65535, the IDX starts again from 0.

Name	Byte Count	Value
StartAddress	4	Starting address of user software data in MCU embedded flash memory
ProgramData	256 (Max)	Data writing to MCU embedded flash memory

This data is stored in DATA if the OTA-D response is acknowleged

Name	Byte Count	Value
ErrorCode	1	Unused (Don't care)

The data is stored in DATA if the OTA-D response is not acknowledged

Name	Byte Count	Value
ErrorCode	1	Error code -1: Data package error -2: Error writing to flash

4. Transmission Complete (CMD=0x04)

Transmission complete notification to client

The data stored in DATA at transmission as shown below.

Name	Byte Count	Value
-	1	Unused (0x00)

This data is stored in DATA if the OTA-D response is acknowledged.

Name	Byte Count	Explanation
ErrorCode	1	Error code -1: Data package error -2: Error writing to flash

2.3.4 User Software

Refer to Software Block Diagram in the SK-FM4-216-ETHERNET Wi-Fi Sample Kit Setup Manual for more detail about user software.

3 Preparation of OTA-D Reference Software

This chapter explains how to prepare the OTA-D reference software.

3.1 Preparation Procedure Summary

1. Create a bootloader and write it to the MCU embedded flash memory.

See 3.4 Bootloader for details.

2. Create an OTA-D library.

See 3.5 OTA-D library for details.

3. Link the user software with the OTA-D library.

See 3.6 User software for details.

4. Write the user software to the MCU embedded flash memory.

Connect the SK-FM-216-ETHERNET board and the PC with a serial cable. Start the dedicated serial TCPIP OTA server on the PC. Then write to the flash memory.

See 3.6.2 Write user software to flash memory via serial transmission for details.

*Details on preparing the SK-FM4-216-ETHERNET and PC are listed in 3.2 Set up SK-FM4-216-ETHERNET and 3.3 Set up PC.

3.2 Set up SK-FM4-216-ETHERNET

To prepare the SK-FM4-216-ETHERNET, see 4.1 Hardware Setup in the SK-FM4-216-ETHERNET Wi-Fi Sample Kit Setup Manual.

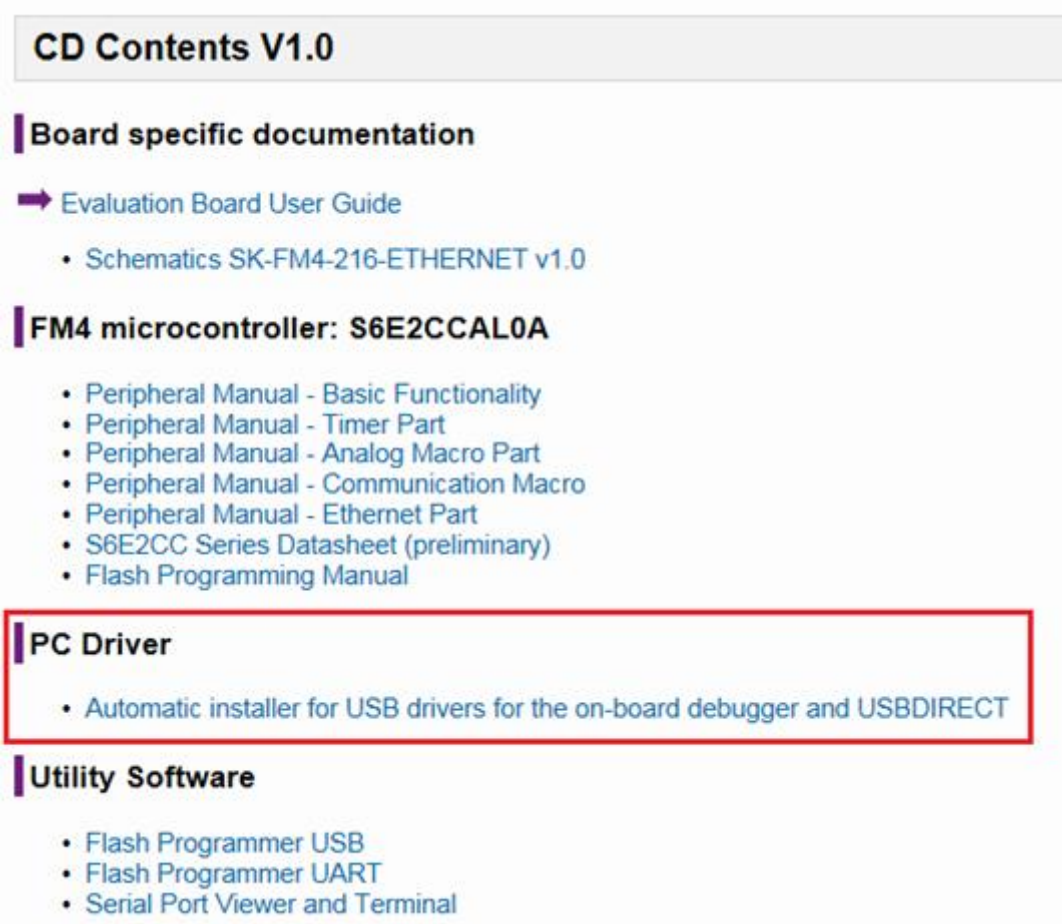
3.3 Set up PC

For information on the recommended PC environment, see 4.1.1 Operating Environment (for reference) in the SK-FM4-216-ETHERNET Wi-Fi Sample Kit Setup Manual. For a prepared PC, install the dedicated drivers for serial transmission using a USB cable between the SK-FM4-216-ETHERNET and the PC. OpenSSL installation to PC will be required for wireless encrypted transmission between the server and the SK-FM4-216-ETHERNET (client).

3.3.1 USB Driver Installation

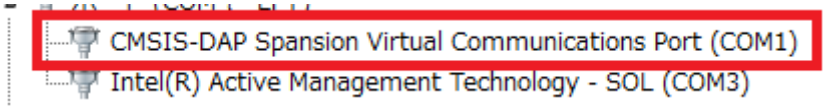
Install the USB driver from the DVD-ROM provided with the SK-FM4-216-ETHERNET to the PC. For more detail, see Figure 5.

Figure 5. USB Driver Installation



After installation, start the device manager of Windows and check that the USB driver has installed correctly. The assigned port will differ depending on the PC.

Figure 6. Checking after Installation



3.3.2 Preparation of Encrypted Transmission

This section explains the procedure for performing encrypted transmissions.

1. Install SSL on the PC as shown in 3.3 Set up PC.

For encrypted transmission, you must install Open SSL from the following link to the PC used as SSL server.

http://slproweb.com/products/Win32OpenSSL.html

The recommended versions are the 32 bit versions, v.1.01J and V 1.0L. The recommended install directory is C:\OpenSSL-Win32.

Note:

- Cypress offers no guarantees regarding the quality of OpenSLL.
- Installing versions of OpenSSL other than those recommended or installing them to a directory other than that recommended here will require a rebuild of the OTA server software.

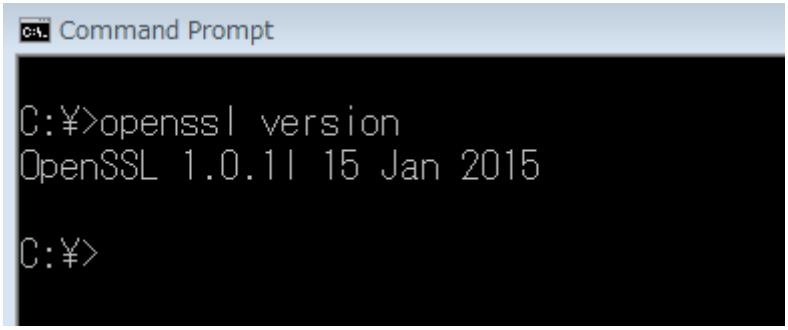
2. Environmental variable "Path" settings

After the installation has completed, add c:\OpenSSL-Win32\bin to the "Path" environmental variable.

3. Install check

Execute an “openssl version” command on Command Prompt. Display of “OpenSSL version” means the installation is completed.

Figure 7. OpenSSL Version Check



4. Server program rebuild

Rebuilding the server program will be required if you installed versions of OpenSSL other than those recommended or installed them to a directory other than that recommended above.

See below for server program preparation

(a)Install "Visual Studio Express 2013 for Desktop"

The recommended environment for rebuilding a server program is "Visual Studio Express 2013 for Desktop." Operation in the "Visual Studio Community" series has not been verified.

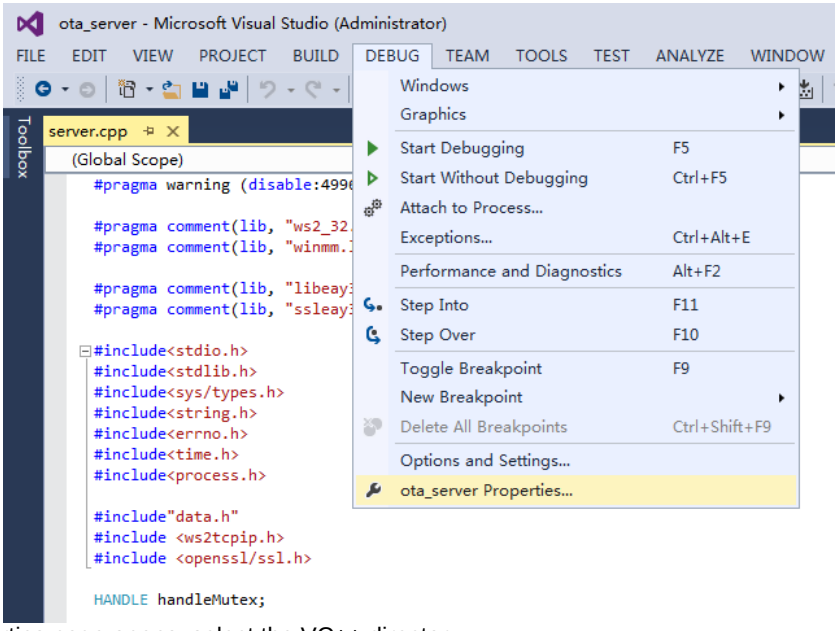
(b) Open the project file

"v02.5a_(SK-FM4-216-ETHERNET)\tools\ota_server_ssl_source\ota_server.sin"

(c) Change the properties

From the menu, select "DEBUG" --> "ota_server properties."

Figure 8. Debug Menu Properties

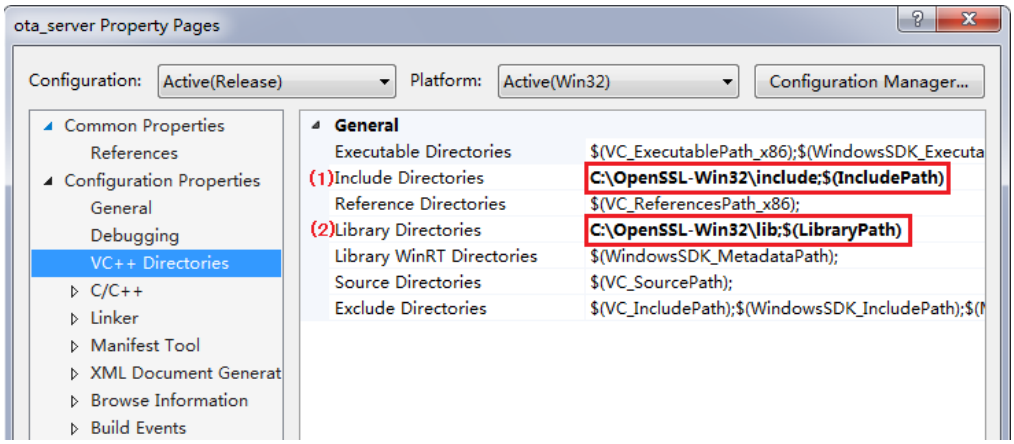


After the properties page opens, select the VC++ directory.

Include the directory in the folder that installed OpenSSL in “Install Directories” (see (1) of Figure 9).

Set lib directory in the folder which installed OpenSSL in “Library Directories” (see (2) of Figure 9).

Figure 9. Property Changes



(d) Rebuild

Perform a rebuild.

(e) Replace executable files

After rebuilding, the "ota_server.exe" executable file is created in “02.5a_(SK-FM4-216-ETHERNET)\tools\ota_server_with_ssl_source\Release." Copy "ota_server.exe" to "02.5a_(SK-FM4-216-ETHERNET)\tools\ota_server_ssl\tcpip."

3.4 Bootloader

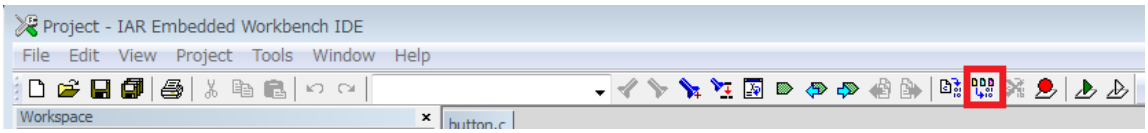
This section explains the procedure for creating a bootloader, and the method to write to MCU embedded flash memory.

3.4.1 Bootloader Creation Procedure

Launch the bootloader EWARM project file.

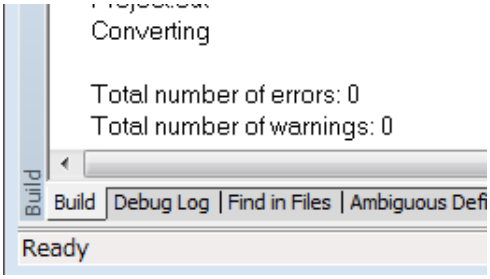
(v02.5a_(SK-FM4-216-ETHERNET)\project\spansion_bootloader\EWARM-RELEASE\Project.eww). Click the "make" button on the tool bar and rebuild the project.

Figure 10. Perform a Bootloader Build



Check the build output information, and check whether the build was successful.

Figure 11. Check the Bootloader Build



3.4.2 Write the Bootloader to MCU Embedded Flash Memory

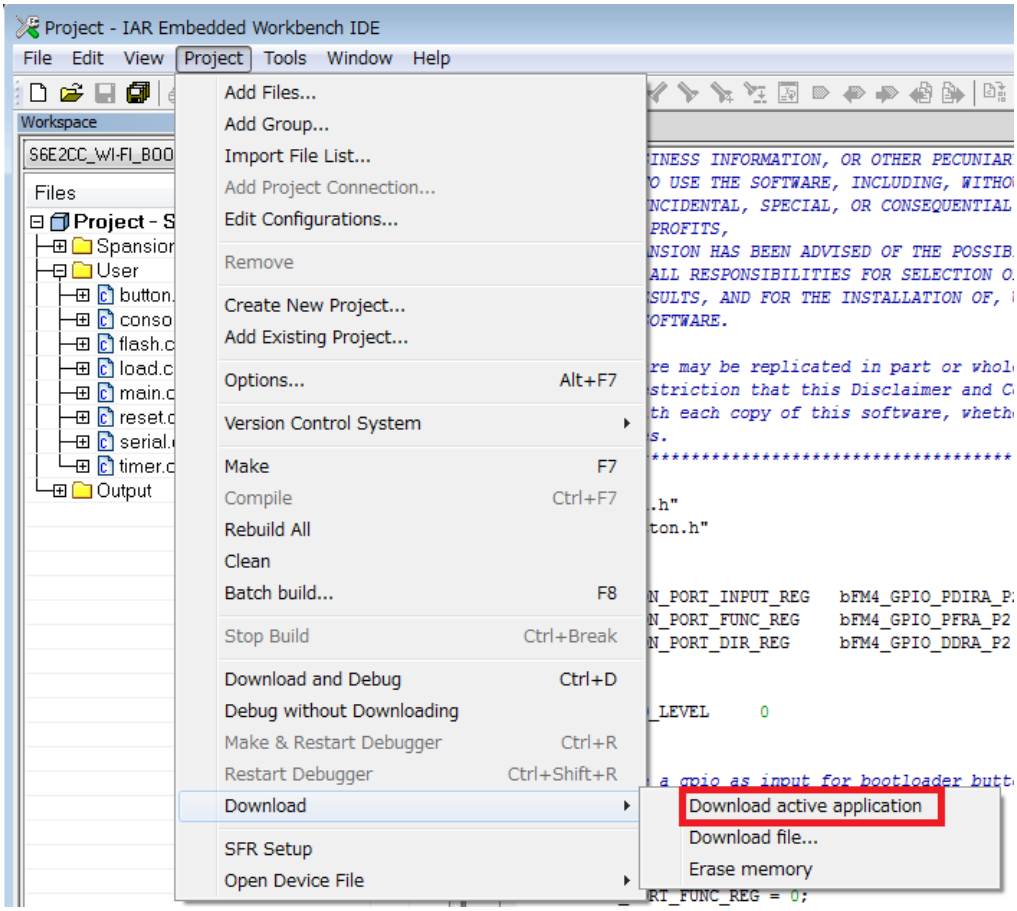
Connect the PC to the SK-FM4-216-ETHERNET JTAG port by ICE*.

Write it from the PC to the SK-FM4-216-ETHERNET.

In the bootloader EWARM project window, select "Project" --> "Download" --> "Download active application" from the menu bar.

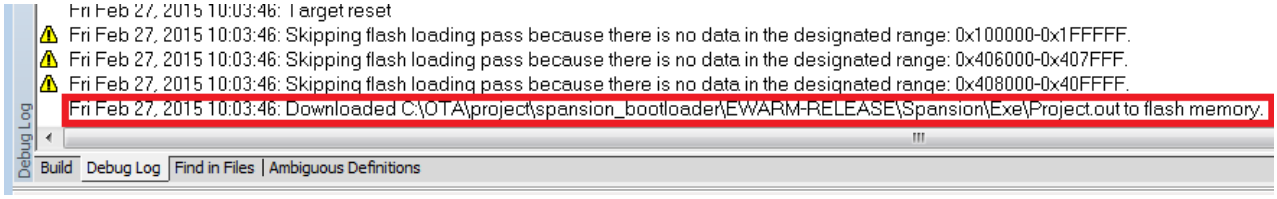
*Configure ICE type-specific settings by opening the option screen under "Project" --> "Options," and then set the "Debugger" --> "Setup" --> "Driver" parameters.

Figure 12. Write the Bootloader to Flash Memory



Confirm that the flash memory write was successful by checking the debug log information.

Figure 13. Check the Results of Writing the Bootloader to Flash Memory



3.5 OTA-D Library

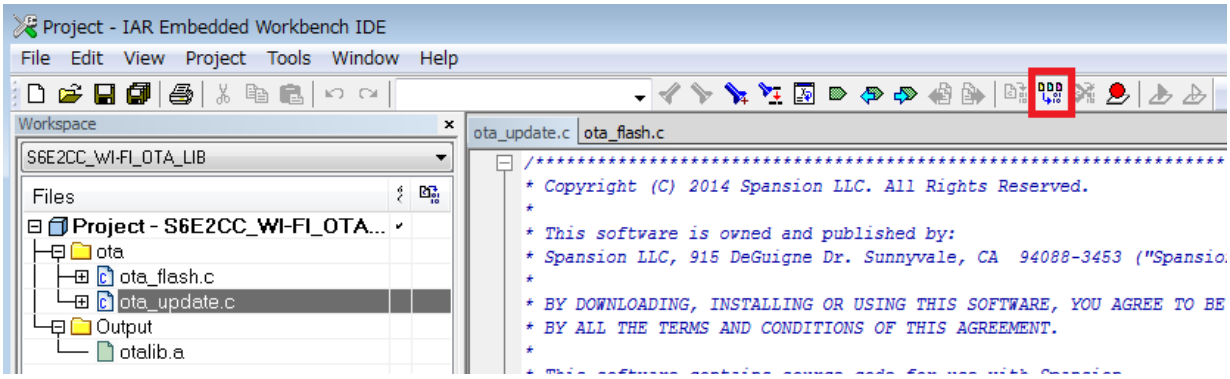
This section explains the procedure for creating the OTA-D library.

3.5.1 OTA-D Library Creation Procedure

Open the OTA-D library EWARM project file (v02.5a_(SK-FM4-216-ETHERNET)\project\spansion_ota\EWARM-RELEASE\Project.eww). Click the "make" button on the tool bar to build the project and generate an OTA-D library file v02.5a_(SK-FM4-216-ETHERNET)\component\common\ota\lib\iar\otalib.a.

This file is linked to user software.

Figure 14. Perform the OTA-D Library BuildThis file is linked to user software.



3.6 User Software

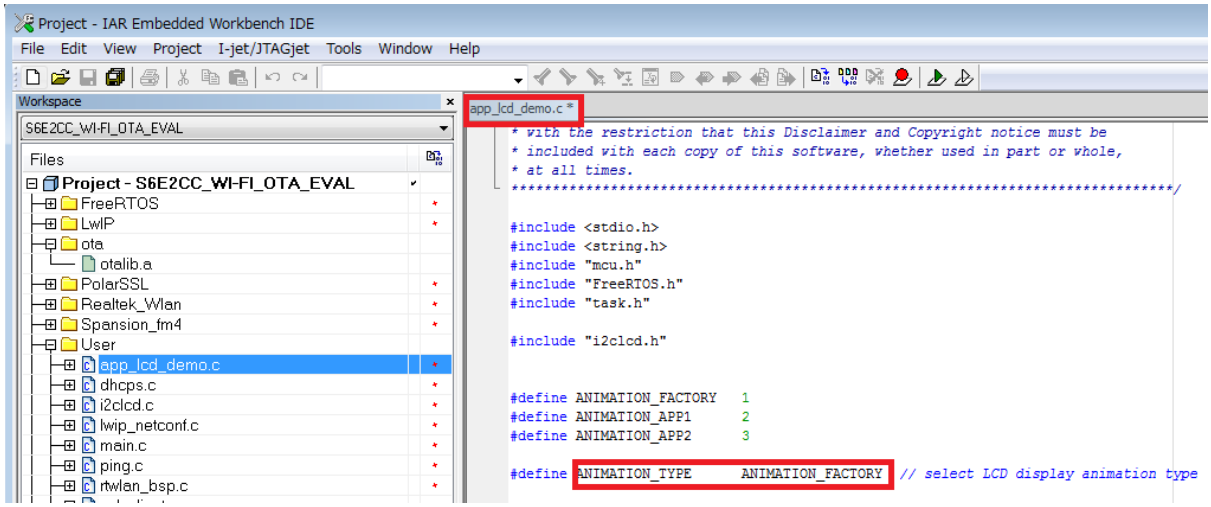
This section explains the procedure for creating user software and the method for writing to flash memory.

3.6.1 User Software Creation Procedure

Launch the EWARM project (=" Project.eww") file for user software on v02.5a_(SK-FM4-216-ETHERNET)\project\spansion_s6e2cx_8189em_with_ota\EWARM-RELEASE.

Set "ANIMATION_TYPE macro" of the app_lcd_demo.c as "ANIMATION_FACTORY" as shown Figure 15

Figure 15. Create the User Software (Create a HEX File as a Factory Default)

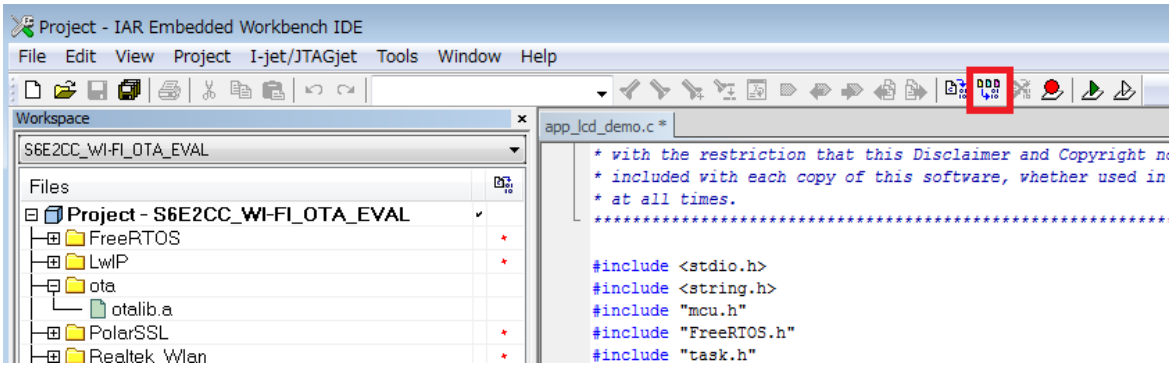


Click the "build" button on the tool bar to build the project.

The result shows that the hex format file (=”Project.hex”) is generated on v02.5a_(SK-FM4-216-ETHERNET)\project\spansion_s6e2cx_8189em_with_ota\EWARM-RELEASE\Spansion\Exe.

Rename it to "Project_factory_reset.hex" as the product factory file. It is used by the serial OTA server

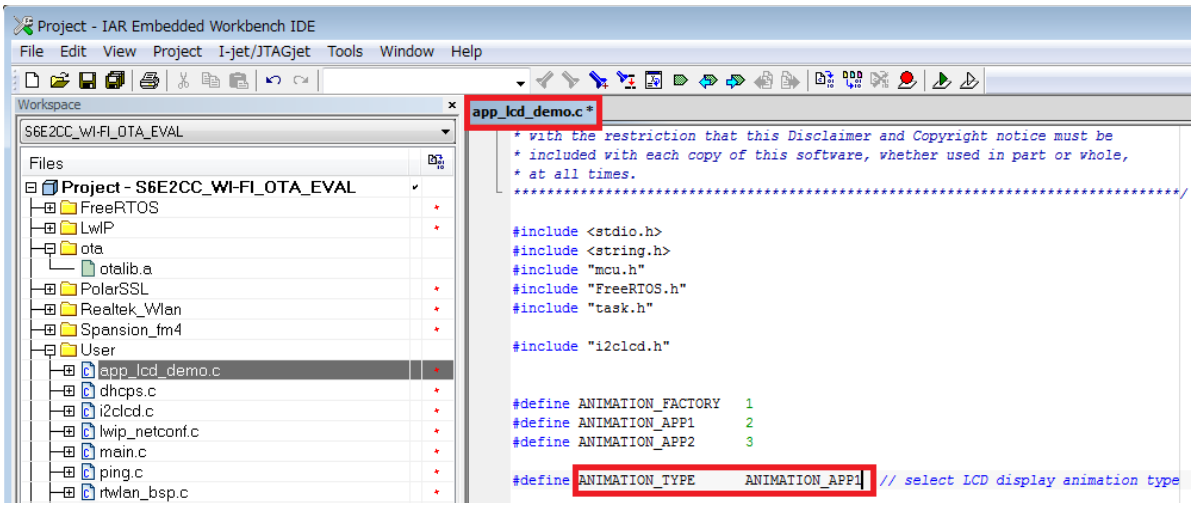
Figure 16. Perform a User Software Build



Next, set the “ANIMATION_TYPE macro” of the “app_lcd_demo.c” as “ANIMATION_APP1”.

See Figure 17.

Figure 17. Create User Software (Create a HEX File as an Application 1)



Build the project and generate the hex format file (=”Project.hex”).

This hex format file is generated on

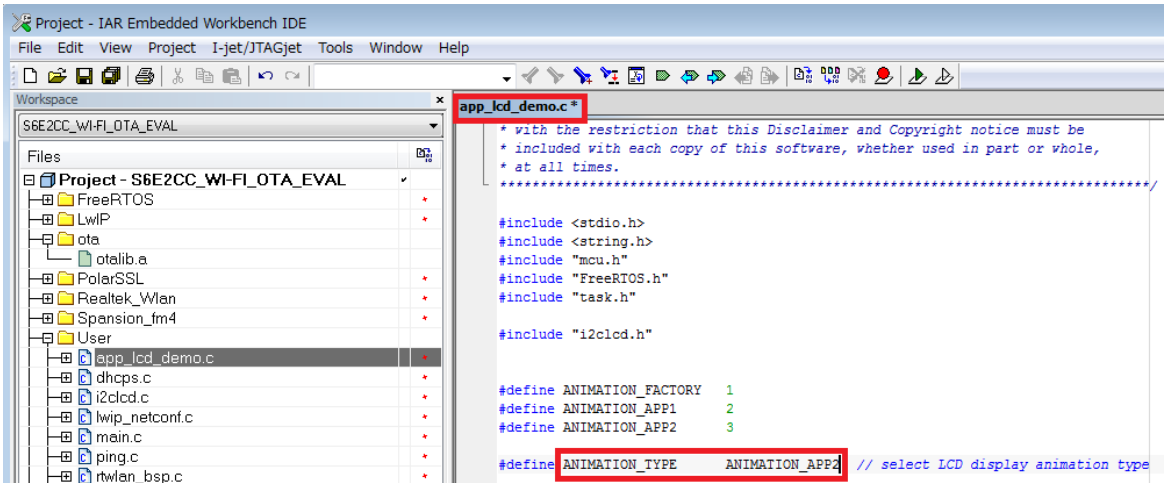
v02.5a_(SK-FM4-216-ETHERNET)\project\spansion_s6e2cx_8189em_with_ota\EWARM-RELEASE\Spansion\Exe.

Rename it to "Project_app1.hex". It is used by the TCPIP OTA server.

Next, set the “ANIMATION_TYPE macro” of the app_lcd_demo.c as “ANIMATION_APP2”.

See Figure 18

Figure 18. Create User Software (Create a HEX File as Application 2)



Build the project and generate the hex format file (=”Project.hex”).

This hex file is generated on

“v02.5a_(SK-FM4-216-ETHERNET)\project\spansion_s6e2cx_8189em_with_ota\EWARM-RELEASE\Spansion\Exe”.

Rename it to "Project_app2.hex". It is used by the TCPIP OTA server.

Note:

- The difference between “Project_app1.hex” and “Project_app2.hex” is the LCD display.

3.6.2 Write User Software to Flash Memory via Serial Transmission

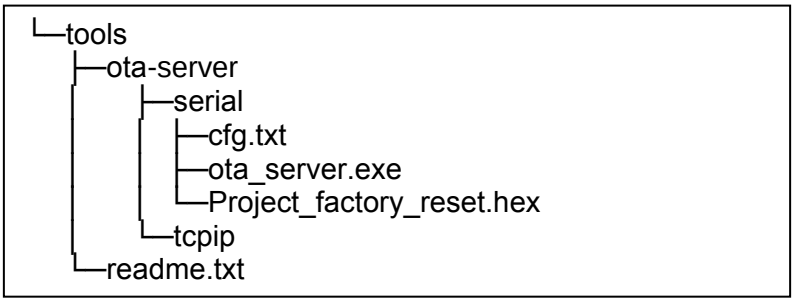
Connect a PC to the SK-FM4-216-ETHERNET USB port using a USB cable.

Copy “Project_factory_reset.hex”* as the user software into

“v02.5a_(SK-FM4-216-ETHERNET)\tools\ota_server\serial” where the SERIAL OTA server is resident.

* This file was created in3.6.1 User Software Creation Procedure

Figure 19. Serial OTA Server Directory Structure



Note:

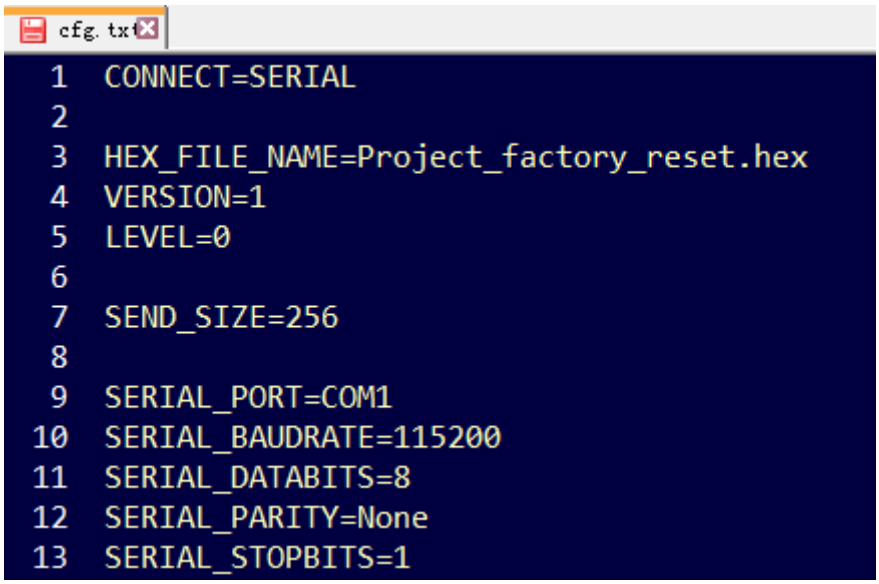
- The serial OTA server enables parsing of only hex format files.

Set “SERIAL_PORT” in the configuration file (=” cfg.txt”) to the connected COM port number in the user environment.

Refer to “readme.txt” on “v02.5a_(SK-FM4-216-ETHERNET)\tools\ota_server”.

It explains the setting fields in the configuration file.

Figure 20. Contents of the Serial OTA Server Configuration File



The serial OTA server is a tool for writing to embedded flash memory with a bootloader via a serial cable.

The tool will be used in one of two situations:

- No user software with bootloader is yet on the embedded flash memory of the SK-FM4-216-ETHERNET board, or the software is not up to date.
- User software with bootloader is already on the embedded flash memory of the SK-FM4-216-ETHERNET board and is up to date.

In the first situation, where there is not up-to-date user software with bootloader already on the embedded flash memory of the SK-FM4-216-ETHERNET.

Step 1: Launch the serial OTA server (See Figure 19)

Step 2: Click the reset button (RST) on the SK-FM4-216-ETHERNET.

In the second situation, where there is user software with bootloader already on the embedded flash memory of the SK-FM4-216-ETHERNET.

Step 1: Launch the serial OTA server (See Figure 19

Step 2: Keep user button (SW4) pressed.

Step 3: Click the reset button (RST) and reset the SK-FM4-216-ETHERNET.

Step 4: Press and hold the user button (SW4) for at least 5 seconds.

After the above steps, the serial OTA server will start transmit a user software (=”Project_factory_reseet.hex”). See Figure 21.

After the transmission, Wi-Fi information is displayed as shown in Figure 22

Figure 21. Serial OTA Server Startup Screen

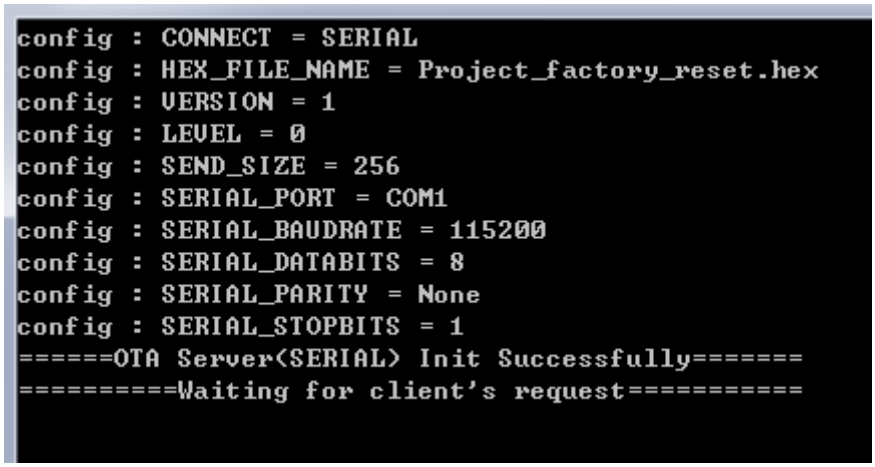


Figure 22. Serial OTA Server Execution Screen

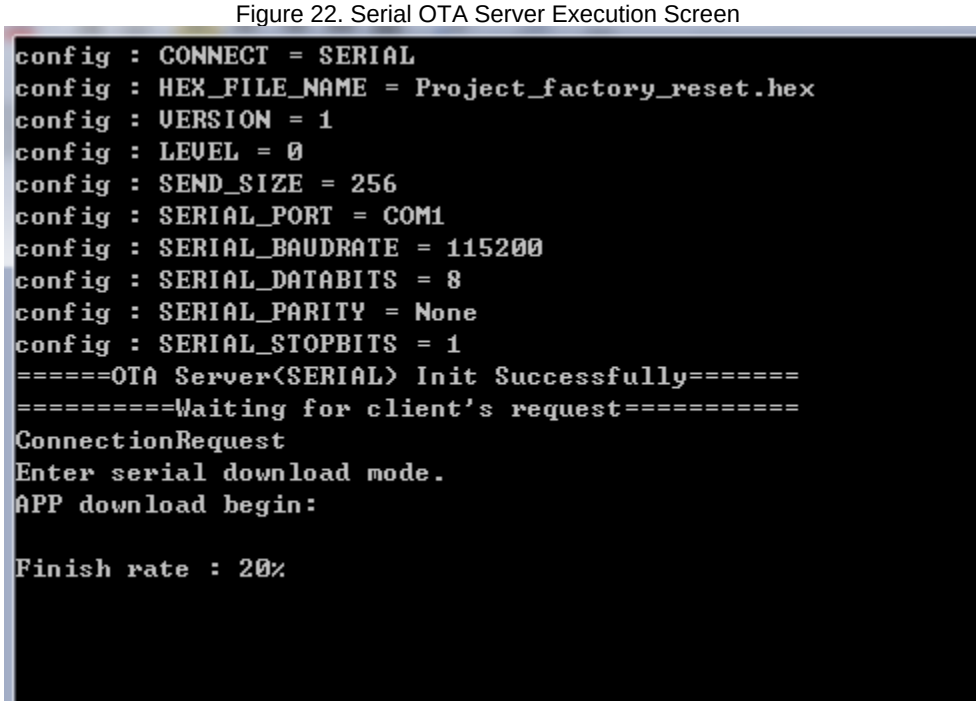
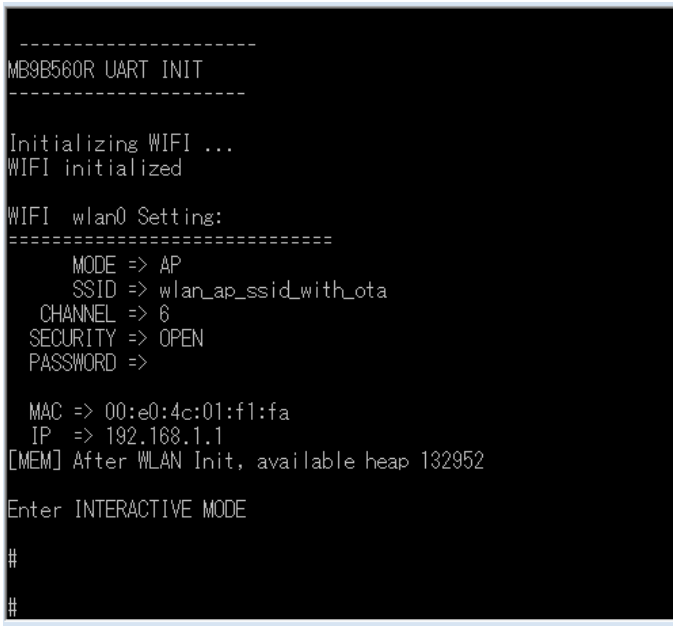


Figure 23. Wi-Fi Information Display after Writing to Flash Memory Completes on Serial OTA Server



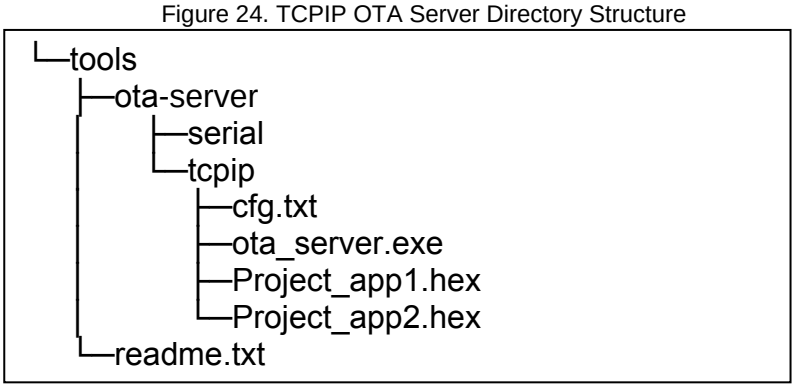
4 Running OTA-D Reference Software

This chapter explains how to run the OTA-D reference software.

1. The PC connects to SSID "wlan_ap_ssid_with_ota". Set the IP address to 192.168.1.4 as a reference.
2. See 3.6.1 User software creation procedure and make Project_app1.hex and Project_app2.hex.

Copy the Project_app1.hex and Project_app2.hex to the v02.5a_(SK-FM4-216-ETHERNET)\tools\ota_server\tcpip folder, where the TCIP-IP OTA server program is stored.

Figure 24. TCPIP OTA Server Directory Structure



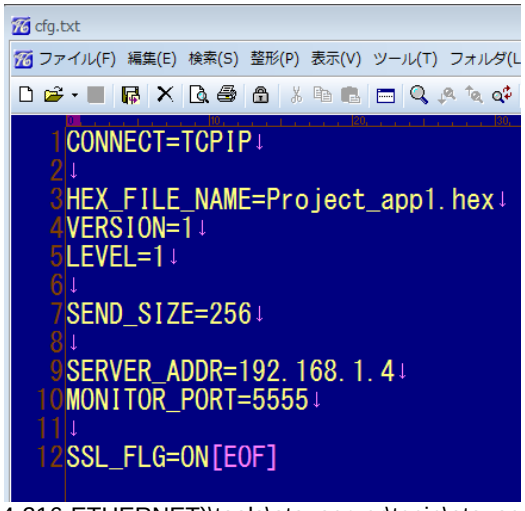
Note:

- The TCPIP OTA server program enables parsing of only hex format files.

3. See Figure 4 2. Edit the v02.5a_(SK-FM4-216-ETHERNET)\tools\ota_server\tcpip\cfg.txt of the TCPIP server configuration file.

(refer to v02.5a_(SK-FM4-216-ETHERNET)\tools\ota_server\readme.txt for an explanation of the configuration fields in the configuration file).

Figure 25. TCPIP OTA Server Configuration File ContentChange SSL_FLG to SSL_FLA=OFF when if not using SSL

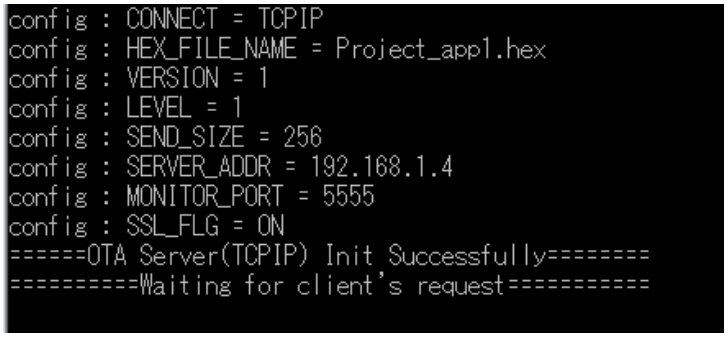


4. Launch v02.5a_(SK-FM4-216-ETHERNET)\tools\ota_server\tcpip\ota_server.exe and check that it starts properly.

Note:

- Configure the Windows firewall to not block this application

Figure 26. TCPIP OTA Server Start Screen



5. Any commands interact with user software via a serial interface. Serial OTA server or Teraterm is recommended for terminal emulators. Enter "help" in the terminal emulator to display a list of commands. Only the "wifi_start_ota_update" command is an OTA-D function command.

Figure 27. Wi-Fi Command List

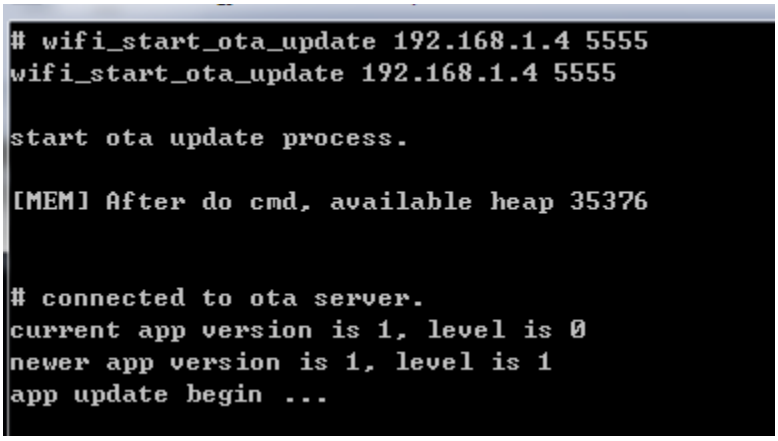
6. Input the "wifi_start_ota_update 192.168.1.4 5555 ssl". This OTA-D command and arguments start user the software update.

"192.168.1.4" is the IP address of the TCPIP OTA server.

"5555" is the port number of the SK-FM4-216-ETHERNET.

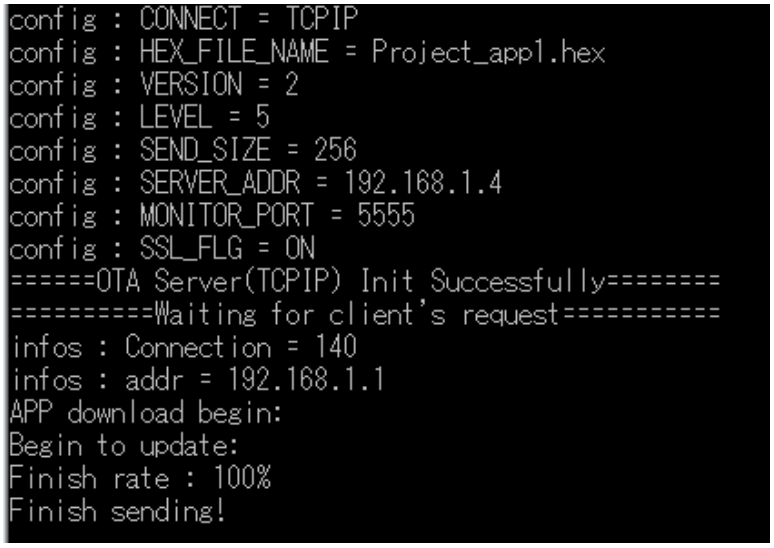
“ssl” enables the Secure Socket Layer (SSL). In situations without SSL, “ssl” is omitted.

Figure 28. TCPIP OTA-D Function Command Screen



After the user software has successfully updated, "Finish sending!" appears on the terminal emulator.

Figure 29. TCPIP OTA Server Execution Screen



7. This section explains how to set new user software written to the flash memory.

Copy the new user software hex file (e.g., Project_app2.hex, see Figure 30) into the v02.5a_(SK-FM4-216-ETHERNET)\tools\ota_server\tcpip folder, where the OTA server program is stored.

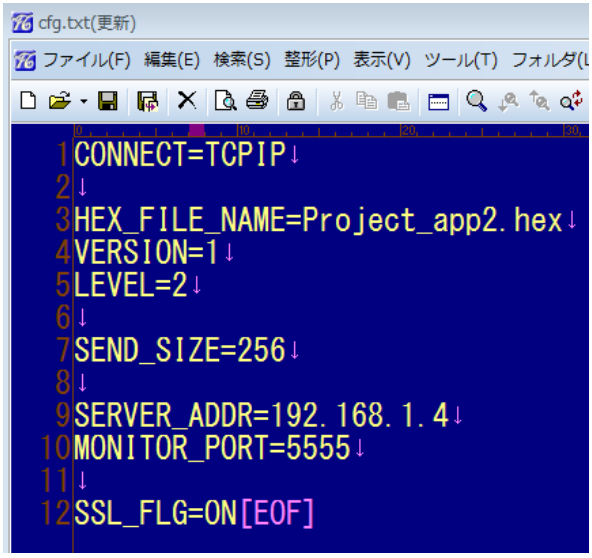
Edit "HEX_FILE_NAME," "VERSION," and "LEVEL," in the OTA server configuration file (="cfg.txt").

Restart the TCPIP OTA server.

Note:

- Restarting the TCPIP OTA server (=”ota_server.exe”) is required for activating the new configuration, after editing the TCPIP OTA server configuration file (="cfg.txt").

Figure 30. Contents of the TCPIP OTA Server Configuration File (When Writing Project_app2.hex to Flash Memory)



5 Examples of OTA-D Use

An example of actual use of this product is the planned release of the "S6E2Cx series 'Over The Air update with Dual Bank eFlash Application" (tentative name).

Document History

Document Title: AN203980 - FM4 S6E2Cx Series Over the Air Update 32-Bit Microcontroller with Embedded Dual Flash

Document Number: 002-03980

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	–	YUIS	06/15/2015	Initial Release
*A	5029280	YUIS	11/30/2015	Migrated Spansion Application Note S6E2Cx_AN709-00019-1v0-E to Cypress format.
*B	5872370	AESATP12	09/06/2017	Updated logo and copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709