AN13073

OTA update via TFTP on LPC5460x

Rev. 0 — 12/2020 Application Note

Contents

1 Overview

1.1 LPC54600 Overview

The LPC546xx is a family of ARM Cortex-M4-based microcontrollers for embedded applications featuring a rich peripheral set with very low power consumption and enhanced debug features. The LPC546xx family includes up to 512 KB of flash, 200 KB of on-chip SRAM, up to 16 KB of EEPROM memory, a quad SPI Flash Interface (SPIFI) for expanding program memory, one high-speed and one full-speed USB host and device controller, Ethernet AVB, LCD controller, Smart Card Interfaces, SD/MMC, CAN FD, an External Memory Controller (EMC), a DMIC subsystem with PDM microphone interface and I^2S, five general-purpose timers, SCTimer/PWM, RTC/alarm timer, Multi-Rate Timer (MRT), a Windowed Watchdog Timer (WWDT), ten flexible serial communication peripherals (USART, SPI, I^2S, I^2C interface), Secure Hash Algorithm (SHA), 12-bit 5.0 Msamples/sec ADC, and a temperature sensor.

1.2 Overview of OTA update

In an embedded application, it is usually required to update firmware after the product is deployed. This is done manually with physical access to the product. One such method is to connect the target embedded device to a host via a cable, such as a PC, via a COM cable. This uses a USART interface to transfer the firmware update. There may be many similar options depending on the available interfaces on the MCU (I^2C, SPI, Ethernet, USB, and so on). A more convenient way is to use an SD card or USB disk to update the firmware. However, all these methods require a person to be present and physical device intervention.

With the development of wireless communication options, MCU technology, and IoT applications, the option for OTA (Over-The-Air) connections is being introduced for firmware update in embedded applications. There are many and various solutions for OTA updates, some of which may be quite complex. However, the basic features of OTA update are that they can be automatic and remote. This overcomes some disadvantages of traditional methods to bring a far more convenient firmware update, albeit with some additional product costs (MCU resources and implementation complexity).

This application note introduces a simple OTA update solution with a TFTP (Trivial File Transfer Protocol) through a WiFi module. Figure 1 Shows the simple block of the solution.

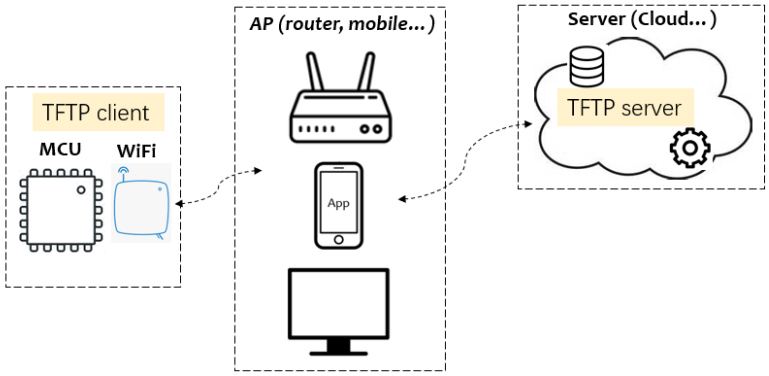

Figure 1. TFTP OTA solution block

2 OTA update via TFTP

The basic idea of the solution is that the TFTP client on the MCU gets the binary file containing the firmware via a WiFi Access Point (AP), such as a router, from a remote host on which a TFTP server runs. After received by the MCU, the file is stored in the external QSPI flash. After the powerup or reset, the updated firmware is written to the internal flash and the MCU boots up with the updated firmware.

The details of the implementation are introduced further on in this document.

2.1 Hardware and software development resources

- **Hardware resources**

  The hardware board is the LPCXpresso5460x evaluation board connected to a WiFi module shield through the Arduino interface. Figure 2 shows the board and the resources.

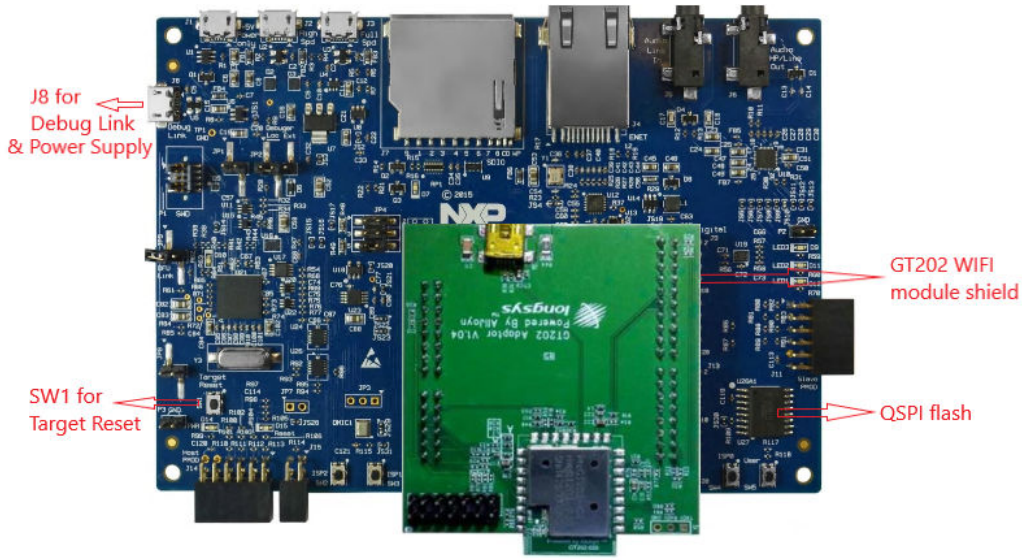

Figure 2. LPCXpresso5460x evaluation board with WiFi module shield

Remarks:

1. The LPCXpresso5460x evaluation board and design files are at the official NXP website:

https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/lpc54000-cortex-m4-/lpcxpresso54628-development-board:OM13098

2. The GT202 WiFi module shield can be found at the following third-party website:

https://www.arrow.com/en/products/gt202-gc3013-frdm4-kit/shenzhen-longsys-electronics-co-ltd

- **Software resources**

  Based on the above hardware, there are some WiFi examples with the WiFi driver and stack based on FreeRTOS in the LPCXpresso5460x SDK.

To develop the reference code of the solution based on the SDK, it is required to implement a TFTP client conforming to the TFTP protocol and an example to show the solution.

2.2 How to get a file via TFTP client

TFTP is a simple file transfer protocol which allows a client to transfer a file from/to a remote host server. It is based on the UDP for transmission with the retry and timeout mechanism. In the OTA update application, the entire procedure for the TFTP client to get a file from a server is as follows:

1. The TFTP client sends a read request with the name of the file to be read to the server.
2. The server sends a block (512 B) of the file data with the block number (starting from 1) to the client after receiving the request.
3. The client returns an acknowledgement with the received block number to the server after receiving the data successfully.
4. Thus, one transmission is completed successfully. Repeat step 2 and step 3 until all data in the file are transmitted/ received.
5. The server will retry to send until a timeout is reached, if it fails to get the acknowledgement from the client. The current transaction then fails with a timeout error.

Figure 3 shows the procedure of the TFTP client to get a file from the server:

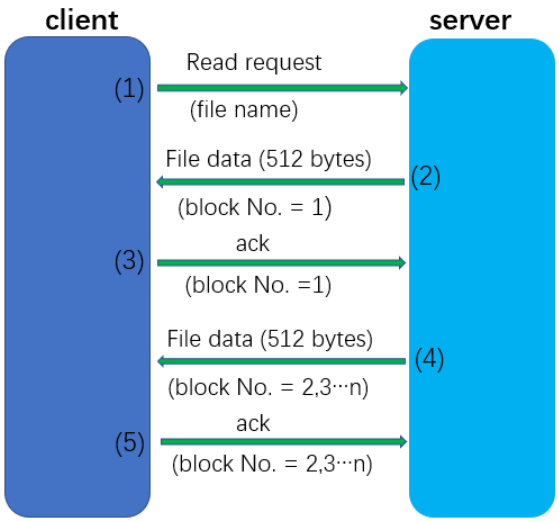

Figure 3. Procedure of the TFTP client to get a file from the server

NOTE

the last data may be <512 B

The basic APIs of the TFTP client functions are implemented in the *wlan_qcom_tftp.c* file in the reference software. They are listed in Table 1.

Table 1. APIs of TFTP client functions

Function name	Description
int tftpPRQ(char *file_name)	Send a read request to the server with an input parameter of the file name
int tftpGet(int timeout, char* *buf, int *buf_len)	Get data from the server with the output of the data and data size
int tftpACK(char *block_no)	Ack to server with the block number of received data bytes

2.3 OTA update solution via TFTP

2.3.1 Solution overview

There are two stages of firmware update: getting the firmware data and programming the firmware to the flash for booting. The NXP MCUs with an on-chip flash generally support the In-Application Programming (IAP). In this solution, the LPC5460x IoT kit board contains a QSPI flash device and the LPC5460x has on-chip flash, so the firmware image received through the WiFi module can be stored into the QSPI flash, and then programmed into an internal flash via IAP calls when the board powers up. The solution reference code has two parts of code: secondary bootloader program (SBL) and the application firmware:

- SBL: code which either loads and boots a new firmware image or directly boots old firmware after checking if there is new firmware in the QSPI flash.
- Firmware: the FreeRTOS-based code that implements the application functions consists of three main tasks:
    1. Main task: keep toggling a LED.
    2. OTA update task: frequently checks if the firmware file is updated on the TFTP server. If yes, get the file and program it into the QSPI flash.
    3. Network setup task: set up the WiFi network for the OTA update when needed. It is required to be done manually through a serial terminal console only when it powers up.

The memory assignment of both the internal and external flash is shown in Figure 4.

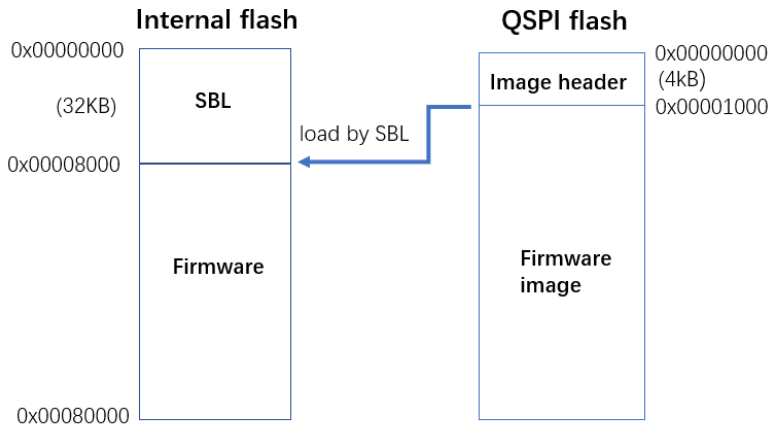

Figure 4. Memory assignment of internal and external flash

NOTE

the image header contains the data that shows the control information of the firmware image (see the section below for the details).

2.3.2 Important data definition

As shown above, there is an important data structure named “image header” which defines the critical information about the firmware image. The SBL determines how to load the firmware image from the QSPI flash into the internal flash according to the information in this image header. The information is updated when the new firmware image is received successfully via the TFTP or loaded to the internal flash successfully.

The data structure of the image header is shown in Figure 5.

```
/* definitions of the updated_flag in image header */
#define EXFLASH_APP_UPDATED_FLAG 0x5a5a5a5a //flag shows image updated in external flash
#define INFLASH_APP_UPDATED_FLAG 0xa5a5a5a5//flag shows image updated in internal flash

typedef struct _image_header
{
    uint32_t image_size;//total size of firmware image
    uint32_t updated_flag;//flag of the image to be updated
    uint32_t version;//version of the firmware image
}image_hd_t;
```

Figure 5. Data structure definition of image header

NOTE

the data definition in the SBL is consistent with the one in the firmware.

For more details about how the information is used in the OTA update, see the section below.

2.3.3 OTA update procedure

The flow chart of the solution for the OTA update via the TFTP is designed as shown in Figure 6.

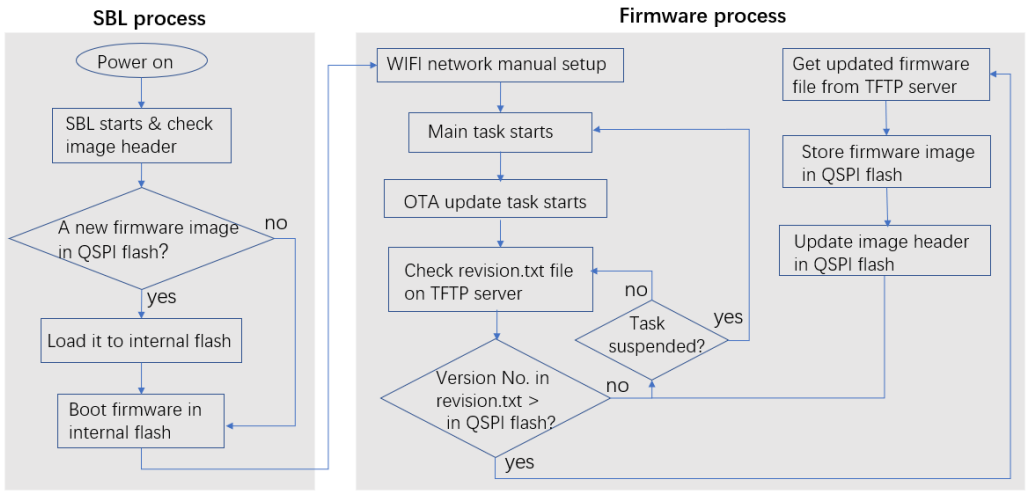

Figure 6. OTA update procedure

When the system powers up, the SBL runs and boots the firmware or determines whether there is new firmware to be loaded into the internal flash from the external QSPI.

The firmware starts the WiFi network setup task with the highest priority first after the system is initialized. Configure and set up the WiFi network manually via the serial terminal console, including selecting an AP from the scanned APs nearby, connecting the AP, requiring the DHCP address for the local target board, and setting and pinging the TFTP server IP address. When a network ping is successful, the application indicates that the setup is completed successfully and that you may exit the setup manually.

After exiting the setup, the firmware goes to the main task to toggle a LED. While the main task is running, the OTA update task frequently checks the version content in the *revision.txt* file on the TFTP server. The content indicates the version of the firmware image file located currently on the TFTP server. If the content value (the default is '0000') is more than the one stored in the image header on the QSPI flash, it indicates that the firmware image file is updated. If this occurs, the OTA update task gets the file and stores it in the QSPI flash. When it is completed, the size and version of the new image is updated in the image header and the updated flag is also marked as updated in the external flash. This updated firmware is loaded into the internal flash and booted up by the SBL after the next power cycle or reset. If the version information in the *revision.txt* file is the same as the one in the image header, no update action is done.

The OTA update is processed automatically without user intervention.

2.4 Solution limitations and improvements

The solution is an example to demonstrate the OTA update via TFTP. There are some limitations in the implementation and some improvements may be done for the actual applications:

- TFTP is not secure for internet access. The TFTP function might be disabled in some APs.
- The WiFi network setup must be done at each powerup in the example, because the setup parameters are stored in the SRAM. In an actual application, these can be stored in a non-volatile memory (EEPROM). Thus, the setup is only done once if there is no need to modify it.
- The WiFi network setup in the example is done via a serial terminal console. In the application, it may be done using a user interface based on LCD display and keypad inputs for the setup.
- In the example, the content integrity of firmware in the flash is not checked before booting. In the application, it is recommend to check this using a CRC or a similar method.

3 Demonstration

This section shows how to demonstrate the OTA update via the TFTP with the example. To demonstrate it, there are two versions of examples provided with different main tasks in the reference code: the main task in one example is to toggle a red LED and the other is to toggle another red LED. Thus, it is easy to observe whether the firmware is updated successfully.

To generate the two different versions of a firmware image, it is only required to switch the "**APP_VERSION**" macro definition in the *main.c* file in the "wifi_tftp_spifi_app" project. See Figure 7 for details.

```
#define APP_VERSION (0)//=0, LED1 shining in the version; =1, LED2 shining
#if APP_VERSION == 0
/* LED1 */
#define APP_BOARD_TEST_LED_PORT 3U
#define APP_BOARD_TEST_LED_PIN 14U
#else
/* LED2 */
#define APP_BOARD_TEST_LED_PORT 3U
#define APP_BOARD_TEST_LED_PIN  3U
#endif
```

Figure 7. Macro definition for two versions of firmware

Additionally, the firmware image must be converted to a binary file for the OTA update via the TFTP. The name of the binary file is fixed as *app.bin*, which is consistent with the macro definition (*#define APP_IMG_NAME "app.bin"*) in the *main.c* file in the "wifi_tftp_spifi_app" project. The *revision.txt* file mentioned in OTA update procedure is also defined (*#define APP_REV_NAME "revision.txt")* in the *main.c* file.

3.1 Hardware environment and setup

- LPCXpresso546xx Evaluation Board with GT202 WiFi module shield. See Figure 2.

- Personal computer

- USB cable

- WiFi access point, such as a router or a mobile phone

Hardware setup:

- Connect the PC to the USB port with the debugger and the VCOM labeled as "J8/Debug Link" on the target board via a micro USB cable.
- Make sure that the AP is available for connection.
- Connect the PC to the AP.

3.2 Software environment and setup

- KEIL MDK v5.27.
- Serial terminal program (PuTTY) with the serial line setup: 115200+8+1+N.
- TFTP server (WinAgents TFTP server manager).
- The reference design software package named “LPC5460X_tftp_spifi_ota” for the solution is attached to this application note. It contains two projects:
  - wifi_tftp_spifi_sbl.uvprojx: for SBL.
  - wifi_tftp_spifi_app.uvprojx: for firmware.

3.2.1 TFTP server setup

The local wireless network is built for the OTA update without a cloud server in this demo. The TFTP server APP is installed on the PC instead of uploading to a cloud service. In both situations, the setup of the TFTP server is the same and simple. As an example, the WinAgents TFTP server manager for the OTA update setup is as follows:

1. Connect to the TFTP server: click the “OK” button to connect with the default settings on the pop-up window with opening the TFTP server APP. Click “Yes” if asked to start the server.
2. TFTP server settings: it is good to use the default settings (port=69, timeout = 5 sec, and retries = 5). Click the icon or the “TFTP server settings” menu to change.
3. **Manage virtual folders**: the *TFTP* virtual folder is the file folder visible to the TFTP clients under a certain name. On the server setup, a new root *TFTP* folder is created; it is the default folder. It is required to set the local path of the files to be downloaded during the initial setup. Thus, the files are mapped to the virtual folder which is under the root folder. The TFTP clients see/get the files directly in the root folder.

The steps to set the local path are as follows (see Figure 8):

- Click the “Manage Virtual Folders” icon and the window with the default configurations appears.
- Select the path of the firmware binary file *[SW package installation path]* *\LPC5460X_tftp_spifi_ota\applications\wifi_tftp_spifi_app\mdk* as the local path in the pop-up dialog window by clicking the “Edit” button.
- The setup is completed and the files under the local path are displayed after clicking “OK” and then “Close”.

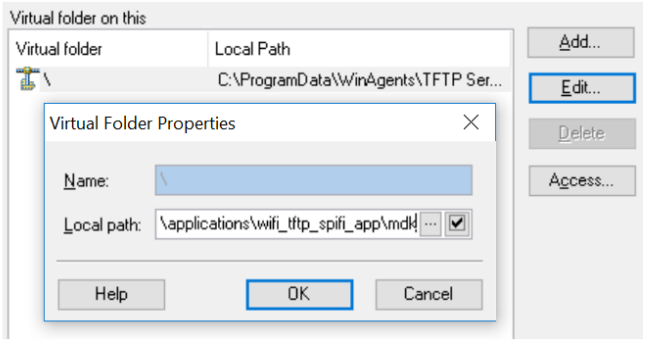

Figure 8. Local path setup on TFTP server

3.3 Running and results

After setting up the hardware and software environments and opening the terminal program and the TFTP server, open the “wifi_tftp_spifi_sbl.uvprojx" project under the *\LPC5460X_tftp_spifi_ota\applications\wifi_tftp_spifi_sbl\mdk* path and the “wifi_tftp_spifi_app.uvprojx” project under the *\LPC5460X_tftp_spifi_ota\applications\wifi_tftp_spifi_app\mdk* path using KEIL

MDK and build both projects. The “wifi_tftp_spifi_app” project is built with the macro definition “#define APP_VERSION (0)”. The first time the demo is run, the internal and external flash on the target board should be cleared by using the “erase” command in the project of SBL. The external flash programing algorithm must be added first. Then download the SBL and firmware APP to the internal flash on the target board.

After the power cycle or reset, the SBL runs first and boots the firmware. Enter the WiFi network setup task to set up the network manually. It shows the setup menu in the terminal program window on the PC (see Figure 9).

```
============ WiFi Network Setup Menu =============
  s  scan & select AP
  c  connect AP
  p  Ping tftp server IP
  r  request DHCP address
  v  view IP configuration
  D  disconnect AP
  m  Print this menu
  x  Exit setup
```

Figure 9. WiFi network setup menu

Menu introductions

- Press “s” to scan the available APs and select your AP to be connected.
- Press “c” to connect to the selected AP and request the DHCP for the target board.
- Press “p” to ping the TFTP server IP.

The WiFi network setup is completed with the above steps.

- Press “r” to request the DHCP for the target board.
- Press “v” to view the IP configurations containing the connected AP info (SSID and password), the DHCP address, and so on.
- Press “D” to disconnect the AP.
- Press “m” to print the commands menu.
- Press “x” to exit the network setup after it is completed.

The steps to set up the network manually are as follows:

1. **Scan and select the AP:**

   Type “s” and the available APs nearby are scanned and listed (see Figure 10).

```
---------Scaned AP List-------------
 No Ch RSSI  MAC                SSID
 0  1   40  d0:72:dc:8a:4b:90  NXP
            RSN: 802.1X / CCMP
            WPA: /
 1  1   39  d0:72:dc:8a:4b:91  NXPGUEST
            open AP
 2  6   33  84:b2:61:6d:67:e0  NXP
            RSN: 802.1X / CCMP
            WPA: /
 3  6   33  84:b2:61:6d:67:e1  NXPGUEST
            open AP
 4 11   17  f6:14:4b:70:ab:20  SCC-GUEST
            open AP
 5 11   16  f6:14:4b:70:ab:21  SCC-WEB
            open AP
--------------------
Select 'No' for one AP to connect :
Type '0','1'...(<10) or 'a','b'...for 10, 11...
(Or type 's' to scan again if AP not listed):
```

Figure 10. Scanned APs list

As shown in the console, repeat typing “s” to scan until the connected AP (check the SSID in the list) is found and listed. Select the AP to be connected by typing the number under the “No” item for the SSID (for example, typing “4” to select the “SCC-GUEST” AP). It is printed, as shown in Figure 11.

```
Select 'No' for one AP to connect :
Type '0','1'...(<10) or 'a','b'...for 10, 11...
(Or type 's' to scan again if AP not listed):4
SSID = 'SCC-GUEST'
Password:
```

Figure 11. Select AP to be connected

Input the password for the AP, if needed. In this demo, there is no password with the open AP. Type “enter” to ignore it.

2. **Connect the AP and request the DHCP address:**

   Next, type “c” to connect the AP and request the DHCP for the target board. If it is done successfully, the “connected” status and the DHCP address are printed, as shown in Figure 12).

```
Reading connection params
  opMode=0 (Station)
  phyMode=mixed
  ssid=SCC-GUEST
EVENT: CLIENT connected
Getting DHCP address...
DNS 0: 202.96.134.133
DNS 1: 1.2.4.8
addr: 192.168.100.118 mask: 255.255.252.0 gw: 192.168.100.1
```

Figure 12. Connect AP and request DHCP address

3. **Ping TFTP server IP:**

   Type “p” to ping the TFTP server IP. The default IP is “0”, as shown in Figure 13.

```
Key 'p': Ping tftp server IP
tftp server ip = 0.0.0.0
Change the tftp IP address? [y or n]:
```

Figure 13. Ready to set TFTP server IP

Type “y” and input the server IP address (it is the IP of the wireless network on the PC). After pressing “enter”, it pings the IP address and shows “OK” if successful. At this point, the network setup is completed and it can be closed by typing “x”. See Figure 14 for details.

```
Change the tftp IP address? [y or n]:y

ftp server IP address (format: xx.xx.xx.xx):192.168.100.196
tftp server ip = 192.168.100.196
Pinging 192.168.100.196... OK (83 ms)
Setup completed! Can exit with typing 'x'
```

Figure 14. Set and ping TFTP server IP

After typing “x”, the WiFi network setup task exits and the main task runs - the red LED (“LED1”) on the target board flashes. After several seconds, the OTA update task starts to get the *revision.txt* file from the TFTP server to check if the revision of the firmware is updated (see OTA update procedure for details). Initially, it is only checking periodically (as shown in Figure 15) without any update action.

```
Checking updated app on tftp server...
Received bytes from 192.168.100.196:63678...
Block: 1
```

Figure 15. Checking updated firmware

To demonstrate the firmware update (as mentioned in Demonstration), the “APP_VERSION” macro definition must be changed to 1 and the code must be rebuilt to generate the updated *app.bin* file. The content of the *revision.txt* file must be changed to “0001” to indicate that a firmware upgrade occurred. When the OTA update task runs, it detects the update and then gets and stores the updated firmware image file in the QSPI flash. Figure 16 shows the program output when this occurs.

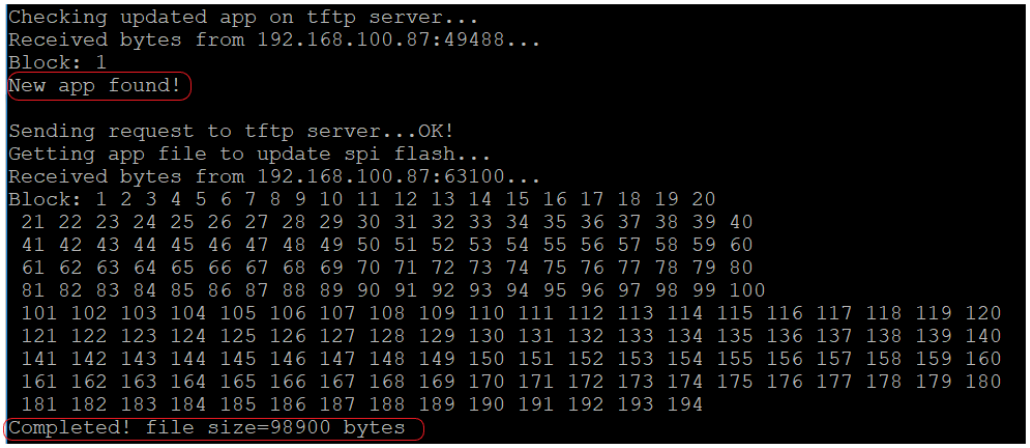

Figure 16. New firmware found and downloaded

After the updated firmware is loaded into the QSPI, the current firmware continues to run if no reset or power cycle occurs. The red LED (“LED1”) continues to flash and the OTA update task continues to check for further firmware updates.

After the reset or power cycle, the new firmware is booted by the SBL. The log in Figure 17 shows the program output when this occurs. Another red LED (“LED2”) on the target board flashes. It indicates that the OTA update via the TFTP is completed successfully.

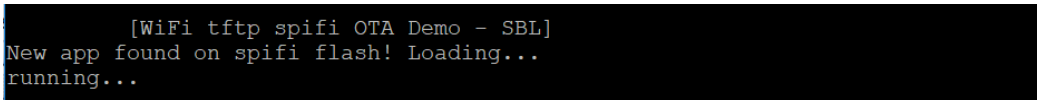

Figure 17. New firmware booted

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

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

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 12/2020

Document identifier: AN13073

arm