CE221653 – PSoC 4 Bootloader and Bootloadable

Objective

This code example demonstrates the basic operation of the PSoC® Creator™ Bootloader and Bootloadable Components.

Requirements

Tool: PSoC Creator 4.2

Programming Language: C (Arm® GCC 5.4.1 and Arm MDK 5.22)

Associated Parts: All PSoC 4 parts

Related Hardware: CY8CKIT-040, CY8CKIT-041-40XX, CY8CKIT-041-41XX, CY8CKIT-042, CY8CKIT-042-BLE, CY8CKIT-042-BLE-A, CY8CKIT-044, CY8CKIT-046, CY8CKIT-048, CY8CKIT-149

Overview

The code example consists of two projects:

- Bootloader – provides the ability to update the firmware via an I^{2}C interface without using an external programmer.
- Bootloadable – an example of a user application that the Bootloader downloads and installs.

Hardware Setup

This example project is configured by default to run on the CY8CKIT-042 development kit from Cypress Semiconductor. The project can be simply migrated to any supported kit by changing the target device with **Device Selector** called from the project's context menu. Table 1 lists the supported kits and corresponding devices.

This example uses the kit's default configuration. Refer to the kit guide to ensure the kit is configured correctly.

Table 1. Supported Kits and Devices

Development Kit	Series	Device
CY8CKIT-040	PSoC 4000	CY8C4014LQI-422
CY8CKIT-041-40XX	PSoC 4000S	CY8C4045AZI-S413
CY8CKIT-041-41XX	PSoC 4100S	CY8C4146AZI-S433
CY8CKIT-042	PSoC 4200	CY8C4245AXI-483
CY8CKIT-042-BLE	PSoC 4200 BLE	CY8C4247LQI-BL483
CY8CKIT-042-BLE-A	PSoC 4200 BLE	CY8C4248LQI-BL483
CY8CKIT-044	PSoC 4200M	CY8C4247AZI-M485
CY8CKIT-046	PSoC 4200L	CY8C4248BZI-L489
CY8CKIT-048	PSoC Analog Coprocessor	CY8C4A45LQI-483
CY8CKIT-149	PSoC 4100S Plus	CY8C4147AZI-S475

The pin assignments for the supported kits are provided in Table 2. For these kits, the project includes control files to automatically assign pins with respect to the kit hardware connections during the project build. To change the pin assignments, over-ride the control file selections in the Pin Editor of the Design Wide Resources by selecting the new port or pin number.

Table 2. Pin Assignments

Development Kit	Pin Assignment			
	Bootloader Project			Bootloadable Project
	\I2C_Slave:scl	**\I2C_Slave:sda**	**Bootloader_StatusLED**	**Bootloadable_StatusLED**
CY8CKIT-040	P1[2]	P1[3]	P0[2]	P1[1]
CY8CKIT-041-40XX	P3[0]	P3[1]	P3[6]	P2[6]
CY8CKIT-041-41XX				
CY8CKIT-042	P3[0]	P3[1]	P0[3]	P0[2]
CY8CKIT-042-BLE	P3[5]	P3[4]	P3[7]	P3[6]
CY8CKIT-042-BLE-A				
CY8CKIT-044	P4[0]	P4[1]	P6[5]	P2[6]
CY8CKIT-046	P4[0]	P4[1]	P5[4]	P5[3]
CY8CKIT-048	P4[0]	P4[1]	P1[6]	P2[6]
CY8CKIT-149	P3[0]	P3[1]	P3[4]	P5[5]

Software Setup

For this code example, you will need the Bootloader Host software which is shipped with the PSoC Creator. The configuration of the Bootloader Host described in Operation section.

Operation

Common preparation

1. Plug your kit board into your computer's USB port.
2. If you using kit, different than CY8CKIT-042, select the new target device for Bootloader and Bootloadable projects. To change the project's device, go to Workspace Explorer and launch **Device Selector** from the project's context menu. This action must be done for both projects.

Bootloader

3. Build the Bootloader project and program it into the device. Choose **Debug** > **Program**. For more information on device programming, see PSoC Creator Help.
4. Confirm that the kit's blue LED is ON.

Bootloadable

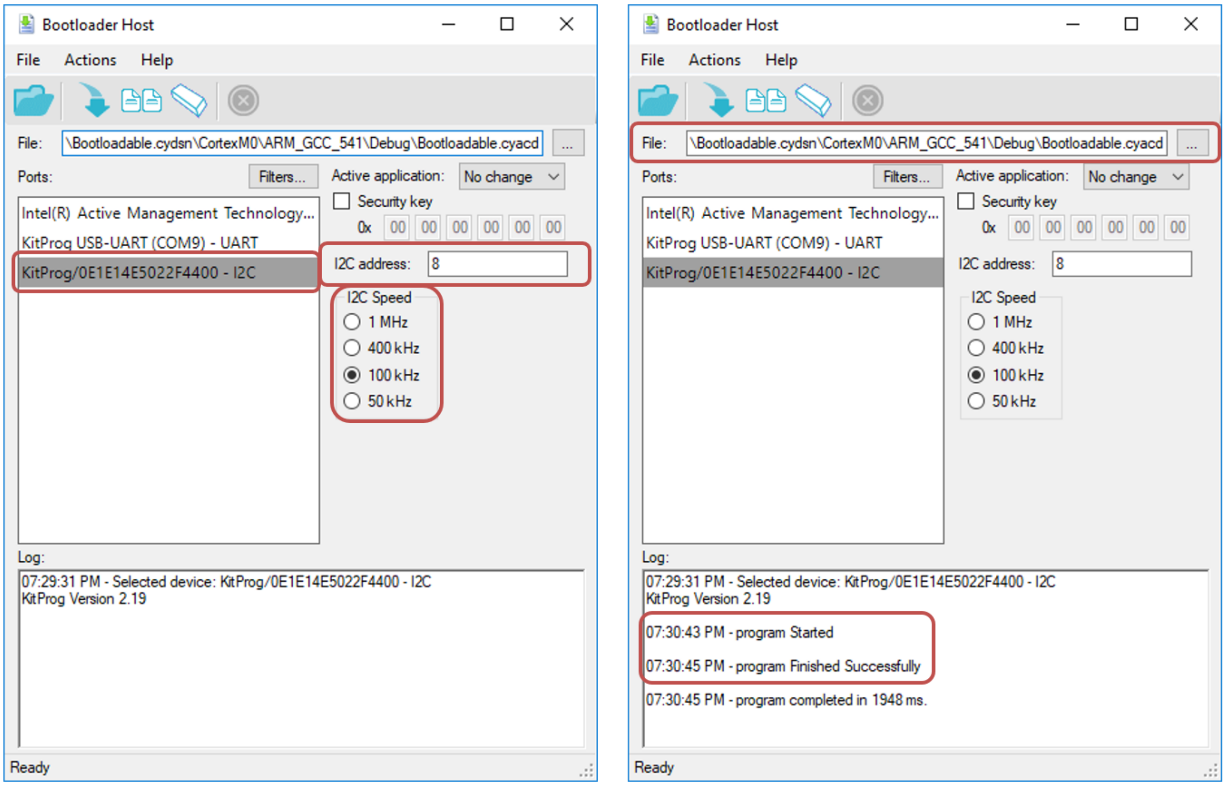
5. In **Workspace Explorer,** set the Bootloadable project as active.
6. In the Top Design, double click on the Bootloadable Component and go to the **Dependencies** tab. Specify the path to the Bootloader project HEX and ELF files by pressing the **Browse…** button. By default, these files are located in the Debug or Release folder of the Bootloader project: *...\Bootloader.cydsn\CortexM0\ARM_GCC_541\Debug\Bootloader.hex*. The path to the Bootloader ELF file is automatically populated with the path to the *.elf file.

Note: For PSoC 4000S, 4100S, and PSoC Analog Coprocessor Bootloader, the HEX file is located in the Debug or Release folder of the Bootloader project: *...\Bootloader.cydsn\CortexM0p\ARM_GCC_541\Debug\Bootloader.hex.*

7. Go to **Build** > **Build Bootloadable**. Confirm that build process completed without errors.
8. Go to **Tools** > **Bootloader Host**. In the **Ports** list, select the kit's USB-I2C bridge: **KitProg/X…X – I2C**. Set the next port configuration: **I2C address** - 0x08, **I2C Speed** – 100 kHz (Figure 1).

9. In **Bootloader host,** select **File** > **Open,** and select the Bootloadable file *Bootloadable.cyacd*. By default, this file is located in the Debug or Release folder of the Bootloadable project: *…\Bootloadable.cydsn\CortexM0\ARM_GCC_541\*. Select **Actions** > **Program** to load the Bootloadable application. Ensure that the programming finished successfully (Figure 1). After the Bootloadable project is successfully uploaded, a software reset occurs, and the device starts executing the project.

Figure 1. Bootloader Host Configuration and Programming



File: \Bootloadable.cydsn\CortexM0\ARM_GCC_541\Debug\Bootloadable.cyacd

Ports:
Intel(R) Active Management Technology...
KitProg USB-UART (COM9) - UART
KitProg/0E1E14E5022F4400 - I2C

Active application: No change

☐ Security key

0x 00 00 00 00 00 00

I2C address: 8

I2C Speed
○ 1 MHz
○ 400 kHz
● 100 kHz
○ 50 kHz

Log:

```
07:29:31 PM - Selected device: KitProg/0E1E14E5022F4400 - I2C
KitProg Version 2.19
```

```
07:29:31 PM - Selected device: KitProg/0E1E14E5022F4400 - I2C
KitProg Version 2.19

07:30:43 PM - program Started

07:30:45 PM - program Finished Successfully

07:30:45 PM - program completed in 1948 ms.
```

Note: To perform successful bootloading operation, the Bootloader on the target device must be active. Check the kit's blue LED's state and Press the kit's **Reset** button if the Bootloader is not active.

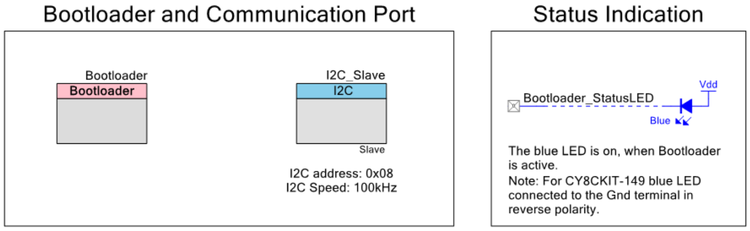
10. Ensure that the kit's green LED is ON. After a few seconds, a software reset occurs and the Bootloadable application returns to the Bootloader – the blue LED turns ON again.

Design and Implementation

Bootloader

The Top Design Schematic of the Bootloader project is shown in Figure 2.

Figure 2. Top Design Schematic of Bootloader Project


Bootloader and Communication Port
Status Indication
Bootloader
Bootloader
I2C_Slave
I2C
Slave
I2C address: 0x08
I2C Speed: 100kHz
Bootloader_StatusLED
Vdd
Blue
The blue LED is on, when Bootloader is active.
Note: For CY8CKIT-149 blue LED connected to the Gnd terminal in reverse polarity.

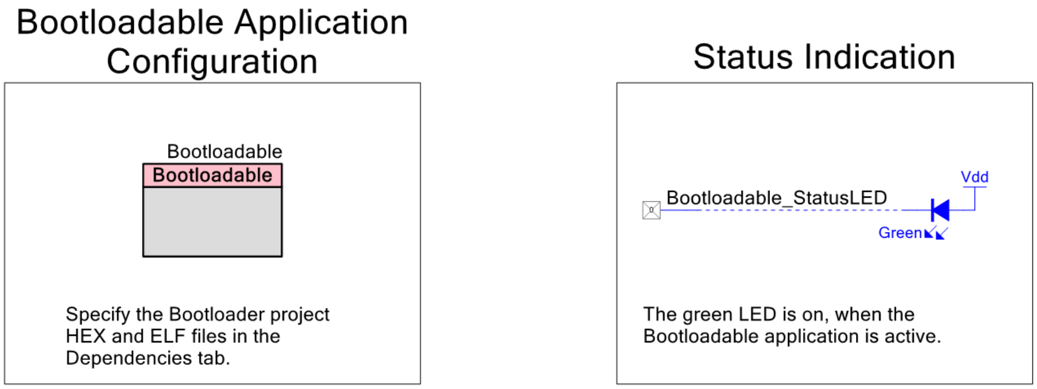
The Bootloader Component uses the I^{2}C (SCB mode) Component configured in the Slave mode to communicate with the host. The Bootloader receives and executes commands from the host, then passes the responses to these commands back to the host; via the I^{2}C interface. The Bootloader collects and arranges the received data and manages the actual writing of flash through a simple command/status register interface.

The Bootloader Component is configured to wait for a command from the host for two seconds after a device reset. If the bootloader does not receive a command within the specified timeout interval, the active Bootloadable application in the flash is executed after the timeout.

Bootloadable

The Bootloadable Top Design Schematic is shown in Figure 3.

Figure 3. Top Design Schematic of Bootloadable Project


Bootloadable Application Configuration
Bootloadable
Bootloadable
Specify the Bootloader project HEX and ELF files in the Dependencies tab.
Status Indication
Bootloadable_StatusLED
Vdd
Green
The green LED is on, when the Bootloadable application is active.

The Bootloadable project is an example of an application that can be downloaded and installed using the Bootloader. The Bootloadable Component defines and configures the Bootloadable project. You must specify the Bootloader project HEX and ELF files in the **Dependencies** tab of the Bootloadable Component to get information about the Bootloader project. This application enables the status LED, waits for a few seconds, and then returns to the Bootloader.

Components and Settings

Table 3 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

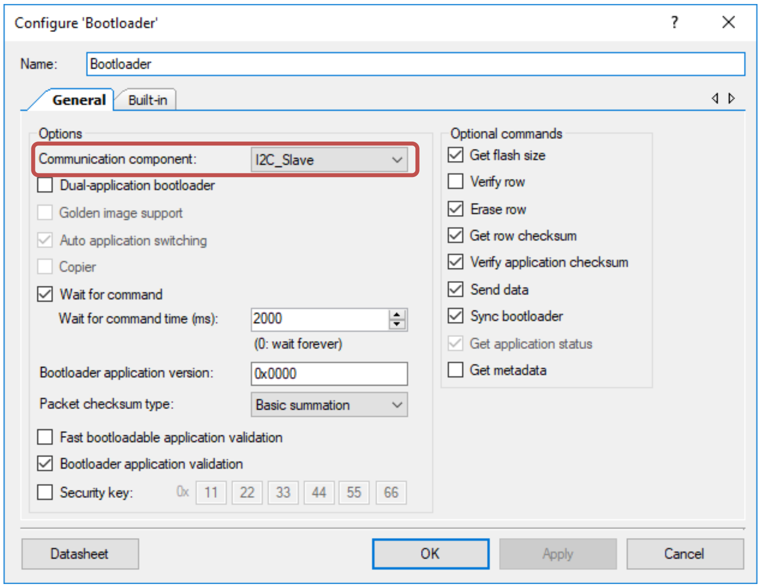
Table 3. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
Bootloader Project			
Bootloader	Bootloader	Manages firmware update process.	See Figure 4
I2C (SCB mode)	I2C_Slave	Communication interface for firmware update.	None
Digital Output Pin	Bootloader_StatusLED	Drives LED to show the Bootloader status.	HW connection: OFF
Bootloadable Project			
Bootloadable	Bootloadable	Configures Bootloadable project.	See Figure 5
Digital Output Pin	Bootloadable_StatusLED	Drives LED to show the Bootloadable status.	HW connection: OFF

For information on the hardware resources used by the Component, see the Component datasheet.

Figure 4 shows the Bootloader Component configuration with highlighted non-default settings for this code example.

Figure 4. Bootloader Component Settings



Name: Bootloader

Communication component: I2C_Slave

☐ Dual-application bootloader

☐ Golden image support

☑ Auto application switching

☐ Copier

☑ Wait for command

Wait for command time (ms): 2000

(0: wait forever)

Bootloader application version: 0x0000

Packet checksum type: Basic summation

☐ Fast bootloadable application validation

☑ Bootloader application validation

☐ Security key: 0x 11 22 33 44 55 66

☑ Get flash size

☐ Verify row

☑ Erase row

☑ Get row checksum

☑ Verify application checksum

☑ Send data

☑ Sync bootloader

☑ Get application status

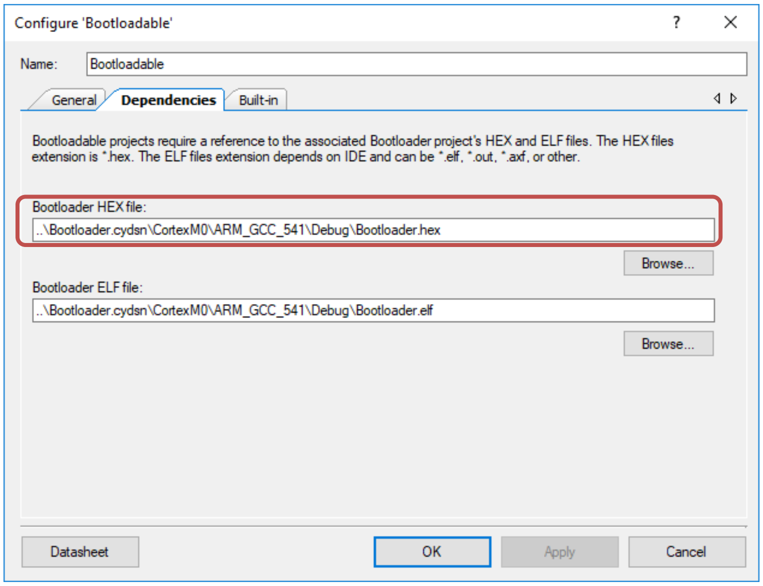
☐ Get metadata

For the correct Bootloadable Component operation, associate your Bootloadable project with the HEX file of the Bootloader project. This way, the Bootloadable project gets the information about the Bootloader project (for example, properly calculates where it belongs in memory).

The Bootloader ELF file is automatically populated with the path to the *.elf file, if it is located in the same folder with the specified HEX file. You can always update this option and specify the path to the ELF file manually.

Figure 5 shows the Bootloadable Component configuration with the highlighted non-default settings for this code example.

Figure 5. Bootloadable Component Settings



Name: Bootloadable

Bootloadable projects require a reference to the associated Bootloader project's HEX and ELF files. The HEX files extension is *.hex. The ELF files extension depends on IDE and can be *.elf, *.out, *.axf, or other.

Bootloader HEX file:
..\Bootloader.cydsn\CortexM0\ARM_GCC_541\Debug\Bootloader.hex

Bootloader ELF file:
..\Bootloader.cydsn\CortexM0\ARM_GCC_541\Debug\Bootloader.elf

Reusing This Example

This example is designed for the kits shown in Table 1. To port the design to a different PSoC 4 device and/or kit, change the target device using **Device Selector** and update the pin assignments in the Design Wide Resources Pins settings as needed.

Related Documents

For a comprehensive list of PSoC 3, PSoC 4, and PSoC 5LP resources, see KBA86521 in the Cypress community.

Application Notes	
AN79953 – Getting Started with PSoC 4	Introduces the PSoC 4 architecture and development tools.
AN73854 – Introduction to Bootloaders	Describes the bootloader theory and technology.
AN86526 – PSoC 4 I2C Bootloader	Describes an I^2C-based bootloader for PSoC 4.
AN68272 – PSoC 3, 4, 5LP, and PSoC Analog Coprocessor UART Bootloader	Describes a UART-based bootloader for PSoC 3, PSoC 4, PSoC 5LP, and PSoC Analog Coprocessor.
PSoC Creator Component Datasheets	
Bootloader and Bootloadable	Supports the process of updating the device flash memory
Serial Communication Block (SCB)	Supports the hardware SCB block
Pins	Supports connection of hardware resources to physical pins
Device Documentation	
PSoC 4 Datasheets	PSoC 4 Technical Reference Manuals
Development Kit (DVK) Documentation	
PSoC 4 Kits	
Tool Documentation	
PSoC Creator	Look in the downloads tab for Quick Start and User Guides

Document History

Document Title: CE221653 – PSoC 4 Bootloader and Bootloadable

Document Number: 002-21653

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5958764	MYKZTMP1	12/28/2017	New code example
*A	6475071	OLPO	02/06/2019	Added links to PSoC 4 resources and PSoC Creator documentation.

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 MCU

Cypress Developer Community

Community | Code Examples | Projects | Videos | Blogs | Training| Components

Technical Support

cypress.com/support

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