EZ-PD™ CCGx Host SDK user guide

About this document

Scope and purpose

This guide provides information and instruction for using the EZ-PD™ CCGx Host Software Development Kit (SDK). The CCGx SDK is a software solution that allows users to harness the capabilities of the Type-C controllers from Infineon to create PD Port Controller applications for notebook computers, desktops, etc.

Intended audience

This document helps application developers understand how to use all the APIs and the tools included with the EZ-PD™ CCGx Host Software Development Kit (SDK).

Document conventions

Table 1 Conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, table and figure captions, screen names, windows, dialog boxes, menus, and sub-menus
Italics	Denotes variable(s) and reference(s)
Courier New	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
>	Indicates that a cascading sub-menu opens when you select a menu item

About this document

Abbreviations and definitions

Table 2 Abbreviations

Abbreviation	Definition
AFC	Adaptive Fast Charging
AMA	Alternate Mode Adapter
APDO	Augmented Power Data Object
API	Application Programming Interface
AVS	Adjustable Voltage Supply
BC	Battery Charging
BMC	Bi-phase Manchester Code
BOD	Brown-Out Detect
CC	Configuration Channel
CDP	Charging Downstream Port
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DCP	Dedicated Charging Port
DFP	Downstream Facing Port
DRP	Dual Role Port
EC	Embedded Controller
ECC	Elliptic Curve Cryptography
EEPROM	Electrically erasable programmable read-only memory
EMC	Electromagnetic Compatibility
EMCA	Electronically Marked Cable Assemblies
EPR	Extended Power Range
EROS	External Requirements Objective Spec
FS	Full Speed
FWCT	Firmware Configuration Table
GNU	GNU is Not Unix (recursive acronym)
GPIO	General Purpose IO
HPI	Host Processor Interface
IDE	Integrated Development Environment
I^2C	Inter-IC Communications interface
INTR	Interrupt
IROS	Internal Requirements Objective Specification
ISR	Interrupt Service Routine
JTAG	Joint Test Action Group
KB	kilobyte, 1024 bytes
LSB	Least Significant Bit
MSB	Most significant Bit

About this document

Abbreviation	Definition
MTK	Manufacturing Test Kit
NPP	New Product Plan
OCP	Over Current Protection
OVP	Over Voltage Protection
PD	Power Delivery
PDO	Power Delivery Object
PHY	Physical Layer
PPS	Programmable Power Supply
PSVP	Pre-Silicon Validation Platform
QC	Quick Charge
RSA	Rivest–Shamir–Adleman, public key cryptography algorithm
SAS	SOC Architecture Spec
SBU	Sideband Use. See Type-C specification
SCB	Serial Communication Block
SCP	Short Circuit Protection
SDP	Standard Downstream Port
SHA	Secure Hash Algorithm, cryptographic hash function
SIP	System-In-Package. Packages that integrate multiple Silicon dies that need to work together.
SOP	Start of Packet symbol directed at USB PD receptacle.
SOP’	Start of Packet symbol directed at USB PD plug near the Downstream Facing Port (DFP)
SOP’’	Start of Packet symbol directed at USB PD plug near the Upstream Facing Port (UFP)
SOP*	Collectively denoting Start of Packet symbol (SOP, SOP’ and SOP’’)
SRAM	Static Random Access memory
SROM	Supervisory ROM
SWD	Serial Wire Debug
TBT	Thunderbolt. High-speed peripheral connect technology from Intel.
UCSI	USB Type-C Connector System Software Interface
UFP	Upstream Facing Port
USB	Universal Serial Bus
USBFS	Full Speed Universal Serial Bus
USB PD	USB Power Delivery
UVP	Under Voltage Protection
VDM	Vendor Defined Message
WDT	Watch Dog Timer
XRES	External Reset

About this document

Reference documents

This user guide should be read in conjunction with the following documents:

- 001-94079 USB PD solution system architecture specification
- 001-94611 EZ-PD™ CCGx integration firmware IROS
- 001-56977 M0S8 system architecture specification (SAS)
- 002-00157 EZ-PD™ CCGx solution EROS

This user guide should be read in conjunction with the following Industry Standard Specifications:

- USB Power Delivery Specification, Revision 3.1, Version 1.6 (USB-IF)
- USB Type C Cable and Connector Specification, Revision 2.1 (USB-IF)
- VESA DisplayPort Alt Mode on USB Type-C Standard, Version 2.1 (VESA)
- USB Type-C Connector System Software Interface Revision 1.1

Table of contents

About this document **1**
Table of contents **5**
1 Introduction **8**
1.1 USB Type-C and Power Delivery 8
1.1.1 USB Type-C highlights 8
1.2 EZ-PD™ Type-C controllers 8
1.3 EZ-PD™ CCGx Host SDK 9
1.3.1 SDK directory structure 10
2 SDK installation **12**
2.1 SDK installation 12
2.2 Tool dependencies 12
2.2.1 PSoC™ Creator 12
2.2.2 Arm® MDK compiler and tools 12
2.2.3 Python 12
2.2.4 EZ-PD™ Configuration Utility 12
2.3 Hardware dependencies 13
3 Getting started with EZ-PD™ CCGx Host SDK **14**
3.1 Preparing to use PSoC™ Creator 14
3.1.1 Environment updates 14
3.2 Using the reference projects 15
3.2.1 Copying the project with PSoC™ Creator 15
3.2.1.1 From Start Page 15
3.2.1.2 From Code Examples 17
3.2.2 Compiling the project with PSoC™ Creator 18
3.2.2.1 Selecting the compiler toolchain 18
3.2.2.2 Compiling the reference project 19
3.2.3 Programming CCGx using the EZ-PD™ Configuration Utility 23
3.3 Updating EZ-PD™ CCGx configuration 25
3.3.1 Configuration parameters 25
3.3.2 EZ-PD™ Configuration Utility 39
4 Customizing the firmware application **40**
4.1 Solution structure 40
4.2 Compile time options 42
4.3 Part number update 47
4.3.1 Device selector 47
4.3.2 Bootloader selection 48
4.3.2.1 Bootloader + Asymmetric Dual Binary Architecture 49
4.3.2.2 Hybrid Boot Architecture 49
4.3.3 Build and re-compile 49
4.4 USB PD Specification Revision support 50
4.5 CYPD6227-96BZXI_notebook_tbt application 51
4.5.1 PSoC™ Creator schematic 52
4.5.2 Compile time options 54
4.5.3 On-chip resource usage 57
4.5.4 Secondary extension 57
4.5.5 EZ-PD™ CCG6SF support 58
4.6 CYPD6227-96BZXI_notebook_dualapp_tbt application 58

Table of contents

4.7 CYPD6227-96BZXI_notebook_dualapp application 59
4.7.1 PSoC™ Creator schematic 60
4.7.2 Compile time options 62
4.7.3 On-chip resource usage 64
4.7.4 EZ-PD™ CCG6SF support 64
4.8 CYPD6125-40LQXI_notebook application 65
4.8.1 PSoC™ Creator schematic 66
4.8.2 Compile time options 69
4.8.3 On-chip resource usage 71
4.9 CYPD5126-40LQXI_notebook application 72
4.9.1 PSoC™ Creator schematic 73
4.9.2 Compile time options 76
4.9.3 On-chip resource usage 77
4.10 CYPD5125-40LQXI_notebook application 78
4.10.1 PSoC™ Creator schematic 80
4.10.2 Compile time options 82
4.10.3 On-chip resource usage 84
4.11 CYPD5225-96BZXI_notebook application 85
4.11.1 PSoC™ Creator schematic 85
4.11.2 Compile time options 86
4.11.3 On-chip resource usage 86
4.12 CYPD4126-24LQXI_notebook application 87
4.12.1 PSoC™ Creator schematic 88
4.12.2 Compile time options 91
4.12.3 On-chip resource usage 92
4.13 CYPD4126-40LQXI_notebook application 93
4.13.1 On-chip resource usage 93
4.14 CYPD4226-40LQXI_notebook application 94
4.14.1 PSoC™ Creator schematic 94
4.14.2 Compile time options 95
4.14.3 On-chip resource usage 96
4.15 CYAC1126-40LQXI_notebook_rkl application 96
4.15.1 PSoC™ Creator schematic 97
4.15.2 Compile time options 99
4.15.3 On-chip resource usage 101
4.16 CYPD3125-40LQXI_notebook application 102
4.16.1 PSoC™ Creator schematic 102
4.16.2 Compile time options 104
4.16.3 On-chip resource usage 106
4.17 CYPD3126-42FNXI_notebook application 106
5 Firmware architecture 107
5.1 Firmware blocks 107
5.2 SDK usage model 109
5.3 Firmware versioning 110
5.4 Flash memory map 112
5.4.1 ACG1F flash memory map 112
5.4.2 CCG5/CCG5C/CCG6 flash memory map 113
5.4.3 CCG6DF firmware FLASH memory map 114
5.4.3.1 Hybrid Boot Architecture 114
5.4.3.2 Bootloader + Dual Asymmetric Application Architecture 114

Table of contents

5.5 Bootloader 115
5.5.1 Secondary base flow 116
5.6 Firmware operation 117
5.6.1 Fault handling 118
6 Firmware APIs 120
6.1 Data structures 120
6.2 API summary 121
6.2.1 Device Policy Manager (DPM) API 121
6.2.2 Application layer API 127
6.2.3 Alternate mode API 133
6.2.4 Hardware Adaptation Layer (HAL) API 134
6.2.4.1 GPIO API 134
6.2.4.2 I2C API 135
6.2.4.3 Flash API 137
6.2.4.4 Timer API 138
6.2.5 Firmware update API 139
6.3 API usage examples 140
6.3.1 Boot API usage 140
6.3.1.1 Perform device reset 140
6.3.1.2 Jump to bootloader 141
6.3.1.3 Jump to alternate firmware 141
6.3.2 GPIO API usage 142
6.3.2.1 Configuring a CCGx pin as an edge triggered interrupt input 142
6.3.2.2 Connecting a pin to the internal ADC 143
6.3.3 Timer API usage 144
6.3.4 HPD API usage 144
6.3.5 Sleep mode control 145
6.3.6 DPM API usage 146
6.3.6.1 Enabling a PD port 146
6.3.6.2 Disabling a PD port 146
6.3.6.3 Sending a DISCOVER_ID VDM 148
6.3.6.4 Getting current PD port status 149
6.3.6.5 Issue a DR_SWAP where required 150
6.3.6.6 Change the source capabilities 150
6.3.7 Solution level examples 152
6.3.7.1 PD event handling 152
6.3.7.2 Application callback registration 152
6.3.7.3 Change the source PDO selection logic 153
6.3.8 Alternate mode handling 155
6.3.8.1 Implementing the alternate mode handlers 155
6.3.8.2 Alternate modes configuration 160
Revision history 162
Disclaimer 164

1 Introduction

1.1 USB Type-C and Power Delivery

USB Type-C is the new USB-IF standard that solves several challenges faced by today's Type-A and Type-B cables and connectors. USB Type-C uses a slimmer connector (measuring only 2.4 mm in height) to enable increasing miniaturization of consumer and industrial products. The USB Type-C standard is gaining rapid support by enabling small form-factor, easy-to-use connectors, and cables that can transmit multiple protocols. In addition, it offers power delivery up to 100 W – a significant improvement over the 7.5 W for previous standards.

1.1.1 USB Type-C highlights

- New reversible connector measuring only 2.4 mm in height.
- Compliant with USB Power Delivery Specification, providing up to 100 W.
- Increases the data bandwidth to 40 Gbps with USB4.
- Combines multiple protocols in a single cable, including DisplayPort, PCIe, and Thunderbolt.

1.2 EZ-PD™ Type-C controllers

Infineon offers the EZ-PD™ line of Type-C controllers, which currently include six product families:

- EZ-PD™ CCG1: Industry's first programmable Type-C port controller
- EZ-PD™ CCG2: Industry's smallest programmable Type-C port controller
- EZ-PD™ CCG3: Industry's most integrated Type-C port controller
- EZ-PD™ CCG4: Industry's first dual-port Type-C port controller
- EZ-PD™ CCG5: Infineon's second generation dual-Port Type-C port controller
- EZ-PD™ CCG5C: Infineon's second generation single-Port Type-C Port controller
- EZ-PD™ CCG3PA: Infineon's USB Type-C power adapter / power bank port controller
- EZ-PD™ CCG6: Single port Type-C port controller with integrated load switch controller
- EZ-PD™ CCG6F: Single port Type-C port controller with integrated load switch
- EZ-PD™ CCG6DF: Dual-Port Type-C port controller with integrated load switch
- EZ-PD™ CCG6SF: Single-Port Type-C port controller with integrated load switch
- EZ-PD™ ACG1F: Single-Port Type-C port controller with integrated load switch

Visit the Type-C controller web page for more details on these product families and a feature comparison.

1.3 EZ-PD™ CCGx Host SDK

The CCGx Host Software Development Kit (SDK) is a software solution that allows users to harness the capabilities of the Type-C controllers from Infineon to create PD port controller applications for notebook computers, desktops, etc.

The following applications are supported by the SDK:

- CCG6SF-RevB based single port Thunderbolt/USB4 PD/RaptorLake port controller solution.
- CCG6SF-RevB based single port AMD Phoenix PD port controller solution.
- CCG6DF based dual port Thunderbolt/USB4 PD/RaptorLake port controller solution.
- CCG6DF based single port AMD Phoenix PD port controller solution.

Host SDK version 3.5 onwards, support for legacy devices CCG3, CCG4, CCG5, CCG5C, and CCG6 is discontinued and only supported in older versions.

Host SDK version 3.6 onwards, support for legacy devices ACG1F and CCG6xF-RevA is discontinued and only supported in older versions.

The SDK provides a firmware stack compatible with Type-C and USB PD specifications, along with the necessary drivers and software interfaces required to implement Notebook/Host applications using the CCG6DF and CCG6SF PD controllers.

The key features of CCGx notebook port controller solutions are:

- Compliant to USB PD specification Revision 3.1, Version 1.6
- Compliant to USB Type-C Specification Revision 2.1
- Compliant to DP2.1 Specification.
- Supports Type-C VBUS overvoltage protection (OVP) and over-current protection (OCP).
- Supports OCP on VConn supply (CCG5, CCG5C, CCG6, CCG6SF, and CCG6DF only).
- Supports VBUS short-circuit protection (SCP) and reverse-voltage protection (RCP) on CCG6, CCG6SF and CCG6DF devices.
- Supports over-temperature protection (OTP) on CCG6SF and CCG6DF devices.
- Supports DisplayPort alternate mode as a DFP_U/DFP_D.
- Supports Thunderbolt3 alternate mode in DFP and UFP roles.
- Supports USB4 operation in DFP and UFP roles.
- Supports Host Processor Interface (HPI) for runtime control of power profiles, modes, etc.
- Supports field firmware upgrades over I2C Interface.
- Supports USB Type-C Connector System Software Interface (UCSI v1.1) protocol.
- Supports I2C slave interface and interrupt to notify Platform Controller (SoC) based on PD state changes.
- Supports I2C master interface to configure Intel Hayden Bridge Retimers as required based on PD state changes.
- Supports I2C master interface to configure kandou retimers for AMD Phoenix platforms.

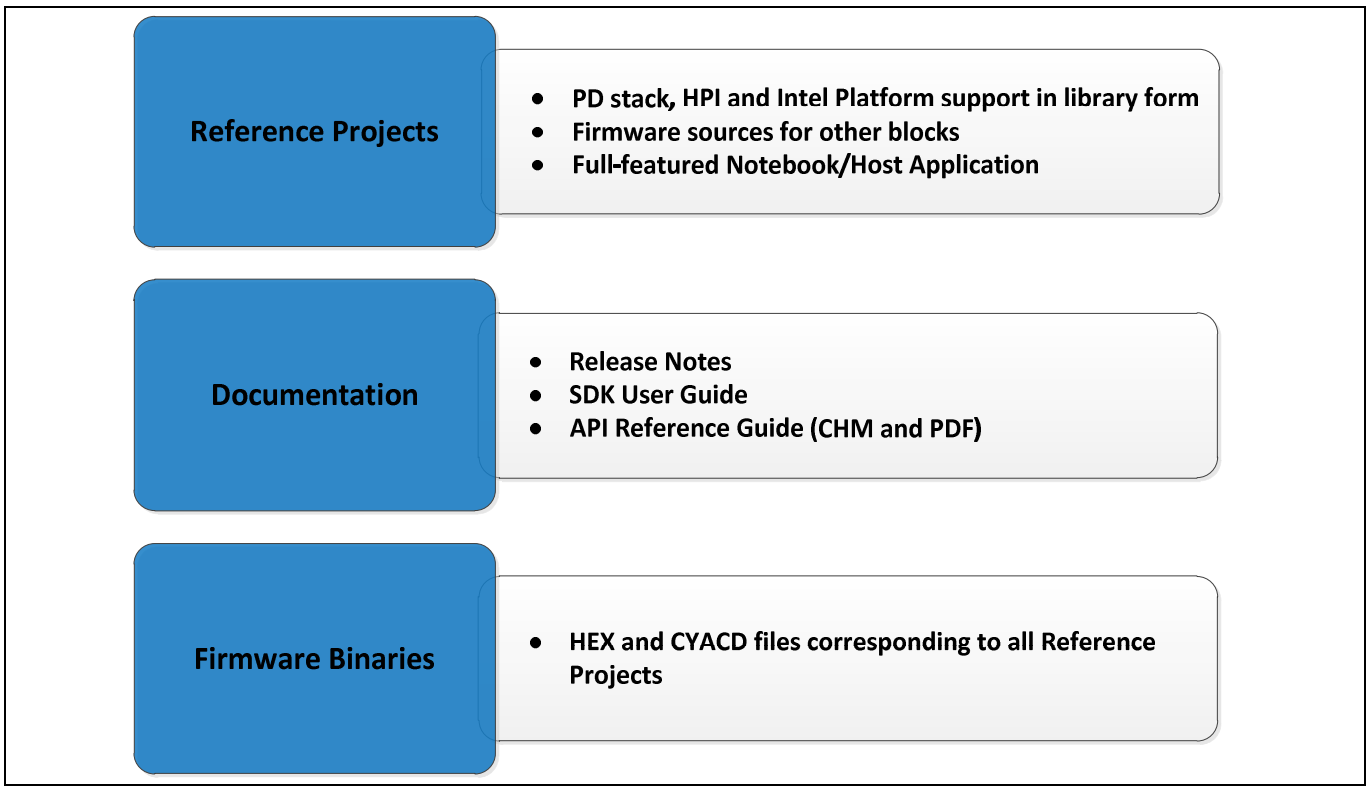


Figure 1 CCGx Host SDK components

The CCGx Host SDK consists of several basic components as shown in Figure 1.

The SDK also includes reference projects implementing standard Type-C applications and documentation that guides the user in customizing existing applications or creating new applications.

1.3.1 SDK directory structure

At the top level, the following folders are present:

- **Documentation**: The docs folder contains the EZ-PD™ CCGx Host SDK documentation, which includes release notes, user guide, and an API reference guide.
- **Firmware**: The firmware folder contains the firmware stack sources, reference projects, and pre-built firmware binaries targeted for the kits and reference designs from Infineon.

  binaries: The binaries folder contains the pre-built firmware binaries

  projects: The projects folder contains the sources and PSoC™ Creator workspaces for the port controller designs.

  - **CYPD6228-96BZXI_notebook_dualapp_tbt**: Intel Raptor Lake USB4 DRD port controller application using CCG6SF R1 (single port) or CCG6DF R1 (dual port).
  - **CYPD6228-96BZXI_notebook_dualapp_amd**: AMD Phoenix port controller application using CCG6SF R1 (single port) or CCG6DF R1 (dual port).
    - **CYPD6228-96BZXI_notebook_dualapp_xxx.cywrk**: PSoC™ Creator workspace settings for the application.
    - **CYPD6228-96BZXI_notebook_dualapp_xxx.cydsn**: PSoC™ Creator projects container directory.

- **Bootloader**: Bootloader binaries used in the application.
- **common**: Application-specific source files.
- **src**: General CCGx firmware sources and headers.
- **lib**: USB PD stack, HPI, and Intel platform support libraries.
- **CYPD6227-96BZXI_notebook_dualapp_xxx.cyprj**: PSoC™ Creator build settings for the application.
- **backup_fw.cydsn**: Container for the reduced feature secondary (backup) application which is combined with the main application.

The src folder inside each reference application has the following sub-folders:

- **app:** The app folder contains the top-level application layer functionality that implements the required USB PD controller functions. This includes functionality such as PDO evaluation and contract negotiation, fault detection, and handling, BC 1.2 charging support, etc.
  - **app/alt_mode:** The alternate mode-specific implementation can be found in this directory.
  - **app/intel_tbt:** This folder contains the headers for the Intel platform support including the I2C slave interface to the platform SoC and the I2C master interface to control retimers.
  - **app/amd:** This folder contains the headers for the AMD platform support including the I2C master interface to the platform SoC and the I2C master interface to control retimers.
- **hpiss:** The hpiss folder contains the API interface definition for the Host Processor Interface implemented by the CCGx firmware.
- **pd_common:** The pd_common folder contains the headers for the core Type-C and USB PD stack for the CCGx device. This includes the Hardware Abstraction Layer (HAL), the Type-C port manager, the USB PD protocol layer, the USB PD policy engine, and the Device Policy Manager (DPM).
- **pd_hal:** The pd_hal folder contains the low-level driver header and source files for the USB PD hardware block.
- **scb:** The scb folder contains the API interface definition for the dedicated I2C slave driver using the Serial Controller Blocks (SCB) on the CCGx device. Because I2C slave mode is the most commonly-used interface for CCGx, a specially optimized driver is provided for the same.
- **system:** The system folder contains header and source files relating to the CCGx device hardware and registers, bootloader and flash access functions, low-level drivers for the GPIO blocks on the CCGx device, and a soft timer implementation that is used by the firmware stack.
- **ucsi:** The ucsi folder contains the implementation of the USB Type-C Connector System Software Interface (UCSI v1.1). The UCSI protocol is implemented on top of the HPI I2C slave interface using separate registers and commands.

Figure 2 shows the installed directory structure of the CCGx Host SDK, along with descriptions for all of the important folders.

```
EZ-PD™ CCGx Host SDK                              CCGx Host SDK Directory
│
└─CCGx                                            CCGx Host Directory
  ├─Documentation                                 Documentation: User guide, API guide, etc.
  └─Firmware
    ├─binaries                                    Pre-built firmware binaries
    └─projects                                    Reference Projects
      ├──CYPD6228-96BZXI_notebook_dualapp_tbt CCG6DF R1 based USB4 DRD controller project
      └──CYPD6228-96BZXI_notebook_dualapp_amd CCG6DF R1 based AMD USB4 DRD controller project
```

Figure 2 CCGx Host SDK directory structure

2 SDK installation

2.1 SDK installation

Once installed, the directory structure will be as shown in Figure 2.

2.2 Tool dependencies

2.2.1 PSoC™ Creator

Infineon's Type-C controllers are based on Infineon's PSoC™ 4 programmable system-on-chip architecture, which includes programmable analog and digital blocks, an Arm® Cortex®-M0 core, and internal flash memory.

The PSoC™ Creator IDE is used for configuring the CCGx devices; to develop and compile the firmware applications and optionally to program the devices using SWD. This version of the SDK requires PSoC™ Creator 4.5 (4.5.0.7) or higher.

This version of PSoC™ Creator can be installed and used on a computer along with previous versions of PSoC™ Creator.

The PSoC™ Creator release includes the GNU Arm® compiler tools.

2.2.2 Arm® MDK compiler and tools

All reference projects in the Host SDK require the Arm® MDK tools (www2.keil.com/mdk5) for compilation. Please obtain a license for the MDK toolchain from Arm® to work with these projects.

Trying to compile the projects using the GNU Arm® tools provided with PSoC™ Creator will result in build errors because of firmware binary size that exceeds the device flash memory constraints.

2.2.3 Python

The USB4/Thunderbolt host reference projects make use of a set of Python scripts to customize the binaries at the end of the build process. These Python scripts are invoked as part of the build process itself and result in build failure if Python is not installed on the computer.

Please install Python 3.7 or later from www.python.org/downloads to ensure that these post-build steps can work properly.

2.2.4 EZ-PD™ Configuration Utility

The CCGx devices are shipped with a pre-programmed bootloader that allows the firmware on the device to be updated through an I2C interface, the CC channel, or the USB interface, which is part of the Type-C interface. All of the parts supported by the Host SDK make use of an I2C bootloader, which means that firmware updates are done through the I2C interface.

The EZ-PD™ Configuration Utility is a Windows-based application, which can be used to program the CCGx devices on Infineon-provided kits (DVKs and EVKs) through the bootloader interface.

The EZ-PD™ Configuration Utility relies on an Infineon USB controller, which can connect to the CCGx device through I2C for programming. Therefore, it will only work with the Infineon-provided kits or other hardware, which includes the Infineon USB – I2C bridge devices.

The EZ-PD™ Configuration Utility is also used for creating custom configurations for the CCGx firmware application, which includes aspects such as the supported power profiles, protection schemes, and so on.

This version of the SDK requires the latest EZ-PD™ Configuration Utility version 1.3.9, which includes support for programming and configuring CCG2, CCG3, CCG4, CCG5, CCG5C, CCG6, CCG6DF, and CCG6xF devices.

2.3 Hardware dependencies

The CY4531 kit (www.infineon.com/cms/en/product/evaluation-boards/cy4531) can be used to evaluate the CCG3 firmware solution.

The CY4541 kit (www.infineon.com/cms/en/product/evaluation-boards/cy4541) can be used to evaluate the CCG4 firmware solutions. The kit ships with the CYPD4225-40LQXI controller which supports only USB PD 2.0 by default. The CCG4 device on the board can be swapped with CYPD4126-40LQXI or CYPD4226-40LQXI devices to work with the projects in the SDK.

Infineon does not provide Evaluation Kits for CCG5, CCG5C, CCG6, CCG6SF, or CCG6DF devices at present. CCG5, CCG5C, CCG6, CCG6SF, and CCG6DF solutions should use the reference schematics available in the respective datasheets.

3 Getting started with EZ-PD™ CCGx Host SDK

3.1 Preparing to use PSoC™ Creator

3.1.1 Environment updates

Many of the reference projects in the CCGx Host SDK make use of a post-build script that combines code from multiple projects into a single binary. These scripts make use of the CyElfTool.exe application which is part of the PSoC™ Creator installation. To enable the projects to locate the executable and run successfully, the path where PSoC™ Creator has been installed needs to be added to the PATH environment variable in the system.

Figure 3 shows the PATH variable updated to add the standard install path for PSoC™ Creator 4.5 (4.5.0.7). If you have installed PSoC™ Creator at a different location, please use the appropriate path.

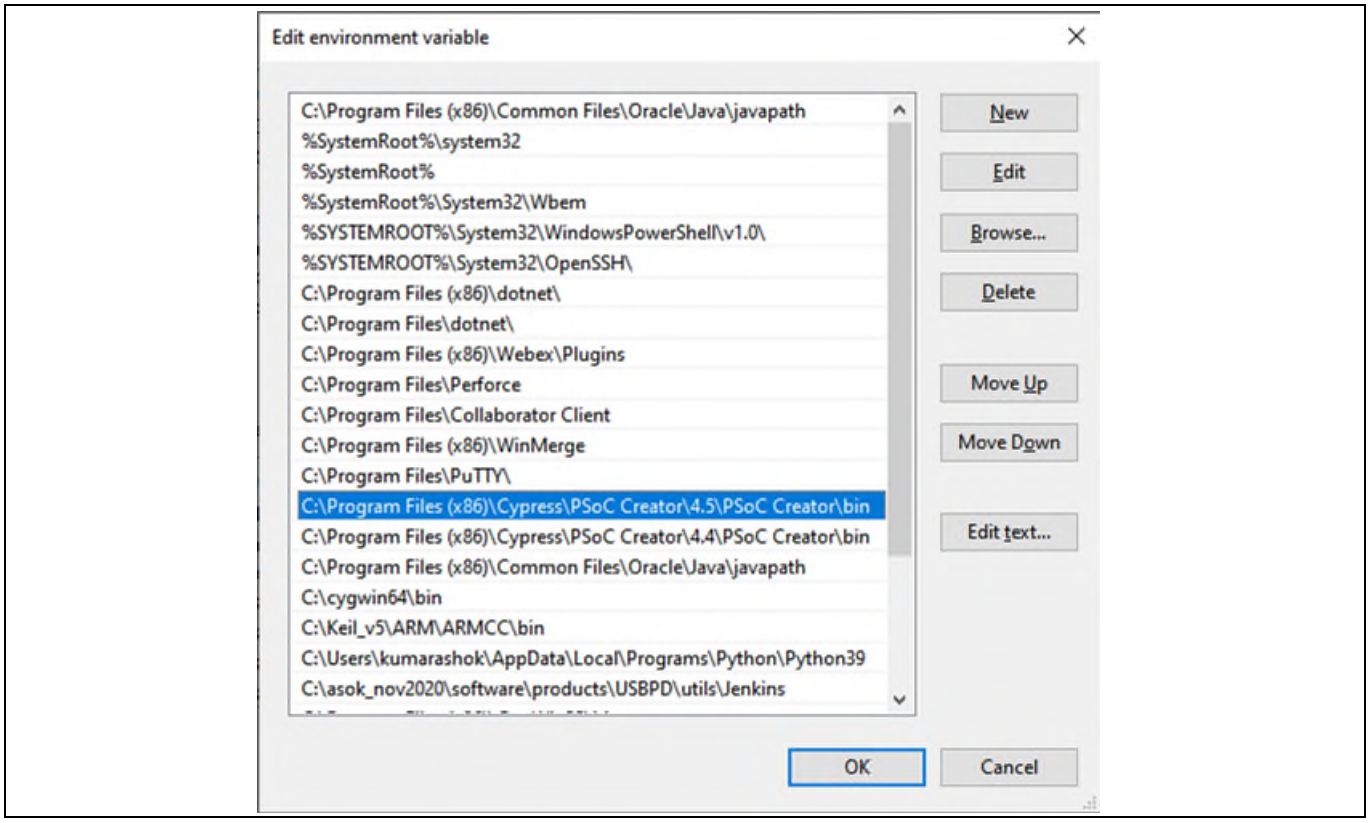


Figure 3 Adding PSoC™ Creator binary folder to PATH

3.2 Using the reference projects

As Figure 2 shows, the SDK includes reference projects for the target applications that can be used to obtain a jump-start in the process of developing CCGx-based notebook port controllers (notebook, for short) applications.

This version of SDK provides the following reference projects for the following applications:

1. **CYPD6228-96BZXI_notebook_dualapp_tbt:** This project implements a dual Type-C port controller for USB4 DRD platforms using the CYPD6228-96BZXI (CCG6DF R1) device. This project also generates a binary that implements a single Type-C port controller for USB4 DRD platforms using the CYPD6128-48LQXI (CCG6SF R1) device.
2. **CYPD6228-96BZXI_notebook_dualapp_amd:** This project implements a dual Type-C port controller for AMD USB4 DRD platforms using the CYPD6228-96BZXI (CCG6DF R0) device. This project also generates a binary that implements a single Type-C port controller for USB4 DRD platforms using the CYPD6128-48LQXI (CCG6SF R0) device.

Each reference project is provided in the form of a PSoC™ Creator workspace. The workspace can be opened using the PSoC™ Creator and the projects can be customized and compiled.

Note: These projects are designed to work with specific devices mentioned above. Changing the target part number using Device Selector will cause the firmware build to fail.

3.2.1 Copying the project with PSoC™ Creator

PSoC™ Creator allows SDK example projects to be copied to a different location without affecting the original installed files. There are mainly two ways of doing this: using the Start Page to copy the workspaces and using the Code Examples.

3.2.1.1 From Start Page

The SDK example projects are listed under Kits→ EZ-PD™ CCGx Host SDK on the Start Page. Click on the workspace name to copy it. When copying the workspace, the complete workspace directory along with all the projects associated with the workspace is copied to the selected destination location. PSoC™ Creator automatically opens the copied workspace after completing the copy. Figure 4 shows the startup page for PSoC™ Creator, expanded to show the reference firmware projects provided with EZ-PD™ CCGx Host SDK.

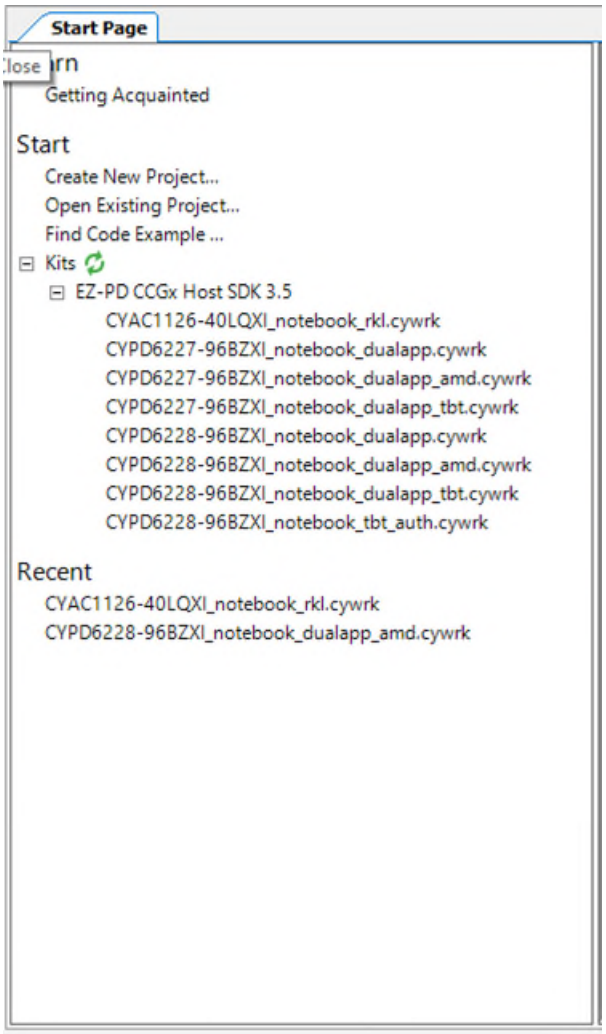


Figure 4 PSoC™ Creator startup page

Note: If Start Page is not open, it can be accessed via **View->Other Windows->Start Page**.

3.2.1.2 From Code Examples

PSoC™ Creator lists the SDK examples under the Code Examples Dialog. This dialog can be accessed via **File->Code Example ...** or during new project creation.

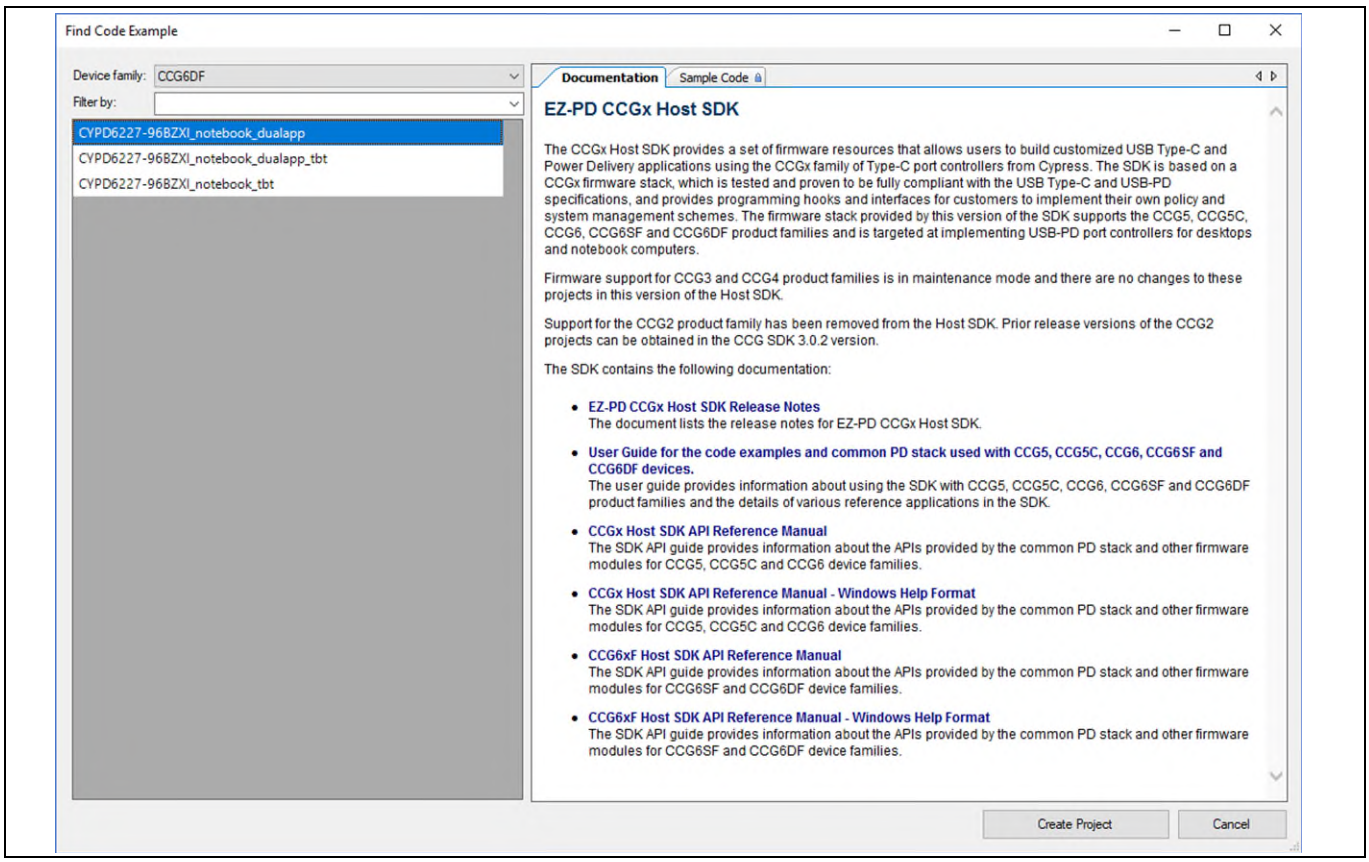


Figure 5 PSoC™ Creator code examples dialog

Figure 5 shows the example projects from Code Examples Dialog. The required Device family can be selected to narrow down the list of examples and the required project can be copied by clicking the Create Project button. When using Code Examples Dialog to copy the project, the complete project directory and the selected project shall be copied. It should be noted that the workspace and the files outside the project directory are not copied. All files required for the SDK examples projects are under the project directory and so this behavior shall not have any impact.

Note: Because a common code example is used to generate binaries for CCG6DF and CCG6SF devices, use the CCG6DF device family to locate and select the reference project for CCG6SF devices as well.

*Note: The bootloader project shall not be present in the workspace created through the code example dialog, and shall require to be explicitly added to the workspace. These projects will still be available under the project root folder and you can use the **File → Add → Existing project** option to add them to the workspace.*

The **File → New → Project** menu option also allows you to create a new project based on existing example projects. Choose the desired device from the list of EZ-PD™ CCGx Host SDK items as the Target Kit in the Create Project dialog as shown in Figure 6.

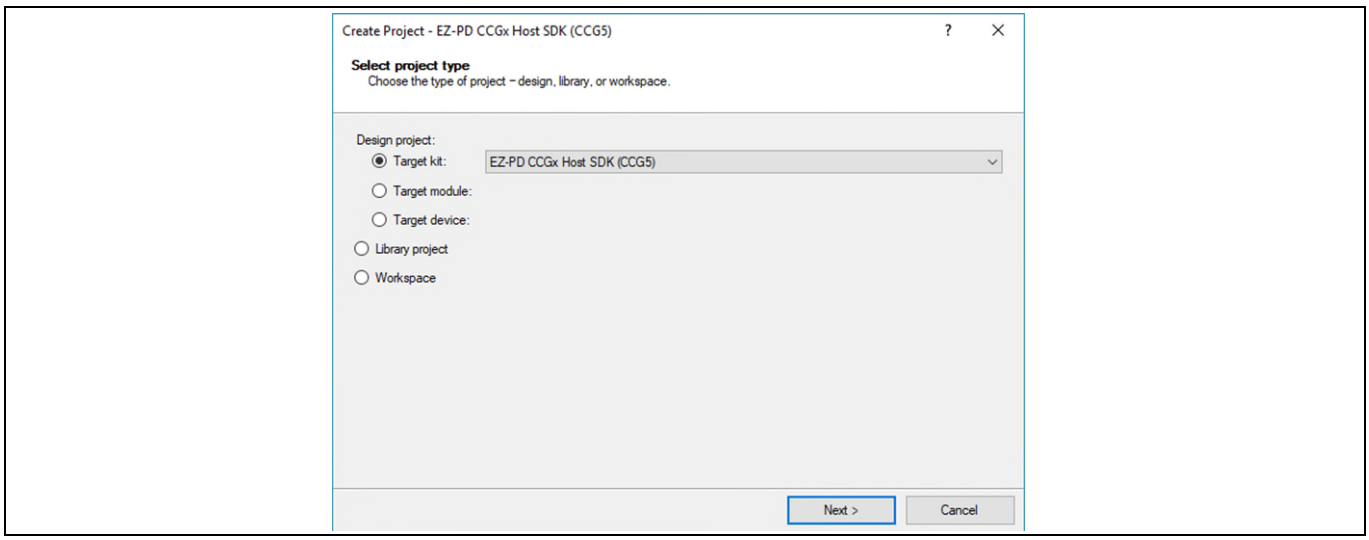


Figure 6 Selection of target kit for creation on new project

Then choose Code Example and select the desired code example as shown in Figure 5.

*Note: When copying via Create Project option, a wrong device part number may get selected. In this case, the user is expected to change the part number to the correct one using **Device Selector Dialog.***

3.2.2 Compiling the project with PSoC™ Creator

This section walks you through the procedure to open the reference projects and build them using PSoC™ Creator. The CYPD5125-40LQXI_notebook project is used as an illustration in the following descriptions.

3.2.2.1 Selecting the compiler toolchain

The code examples provided in the Host SDK can only be compiled using the Arm® MDK toolchain (µVision V5.25 or later). Because the CCGx parts supported by the SDK have up to 128 KB of flash, a full version of the MDK tools is required to make use of this option.

The following steps should be followed to build the code examples using the MDK toolchain.

1. Download and install MDK from www2.keil.com/mdk5. A limited period evaluation license can be used or a full license can be purchased from Arm®.
2. Provide the path to the Arm® compiler toolchain in the PSoC™ Creator settings, by using the Project Management → Arm® Toolchains option under the Tools → Options menu, as shown in Figure 7.

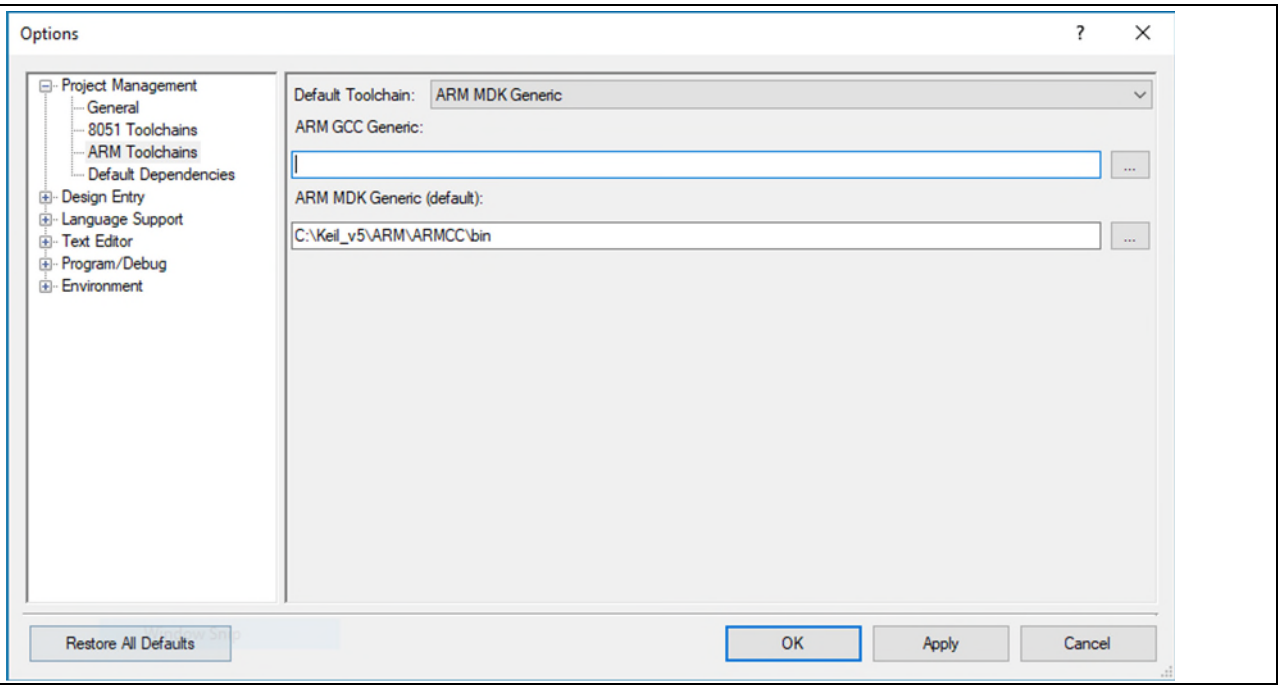


Figure 7 Pointing PSoC™ Creator to the Arm® MDK compiler path

3.2.2.2 Compiling the reference project

The following steps describe the process to compile the reference project and generate the binaries using the PSoC™ Creator IDE and the Arm® MDK tool-chain.

1. Navigate to the project folder using Windows Explorer. The project folder contents will look as shown in Figure 8.
2. The **CYPD6228-96BZXI_notebook_dualapp_tbt.cywrk** file is the PSoC™ Creator workspace file that can be opened using the PSoC™ Creator IDE. If you have installed multiple Creator versions, ensure that the appropriate PSoC™ Creator version (Creator 4.5 or later) is used to open the workspace.

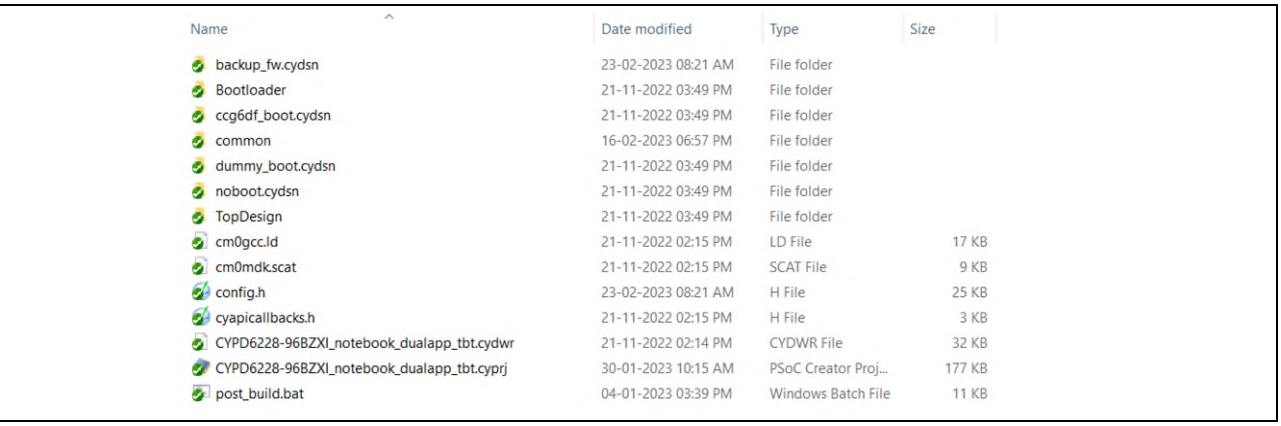


Figure 8 Contents of the reference project folder

3. After you open the workspace, note that there are three projects:

a) ***CYPD6228-96BZXI_notebook_dualapp_tbt.cydsn***: This is the main firmware project for the application. This application is designed to work on top of the bootloader pre-programmed on the CCG6xF device. More details on this project are provided in later sections.

b) ***backup_fw.cydsn***: This is a limited feature version of the notebook PD port controller application using the CCG5 device. This is used as a secondary firmware binary which allows recovery in case the device firmware gets corrupted during an upgrade process.

c) ***noboot.cydsn***: The main firmware project in **CYPD6228-96BZXI_notebook_dualapp_tbt.*cydsn*** does not support runtime debugging through the SWD interface. The *noboot.cydsn* is a version of the same firmware application, which does not depend on the bootloader. This firmware overwrites the complete device flash and expects that the device will be programmed through SWD.

Note: *If the project was copied using Code Examples, then only the main project shall be seen on the workspace. The other two projects can be added to the workspace using **Add Existing Project** option.*

4. The ***CYPD6228-96BZXI_notebook_dualapp_tbt.cydsn*** project is set as the default project for the workspace. Choose the **Build CYPD6228-96BZXI_notebook_dualapp_tbt** menu option from the **Build** menu or the pop-up menu obtained by right-clicking on the project name.
5. Ensure that the compiler Toolchain is set to **Arm® MDK Generic**. This can be verified / modified in the Build settings for the project. The Build Settings Dialog can be opened by right-clicking the corresponding project. Figure 9 shows the Build Settings Dialog.

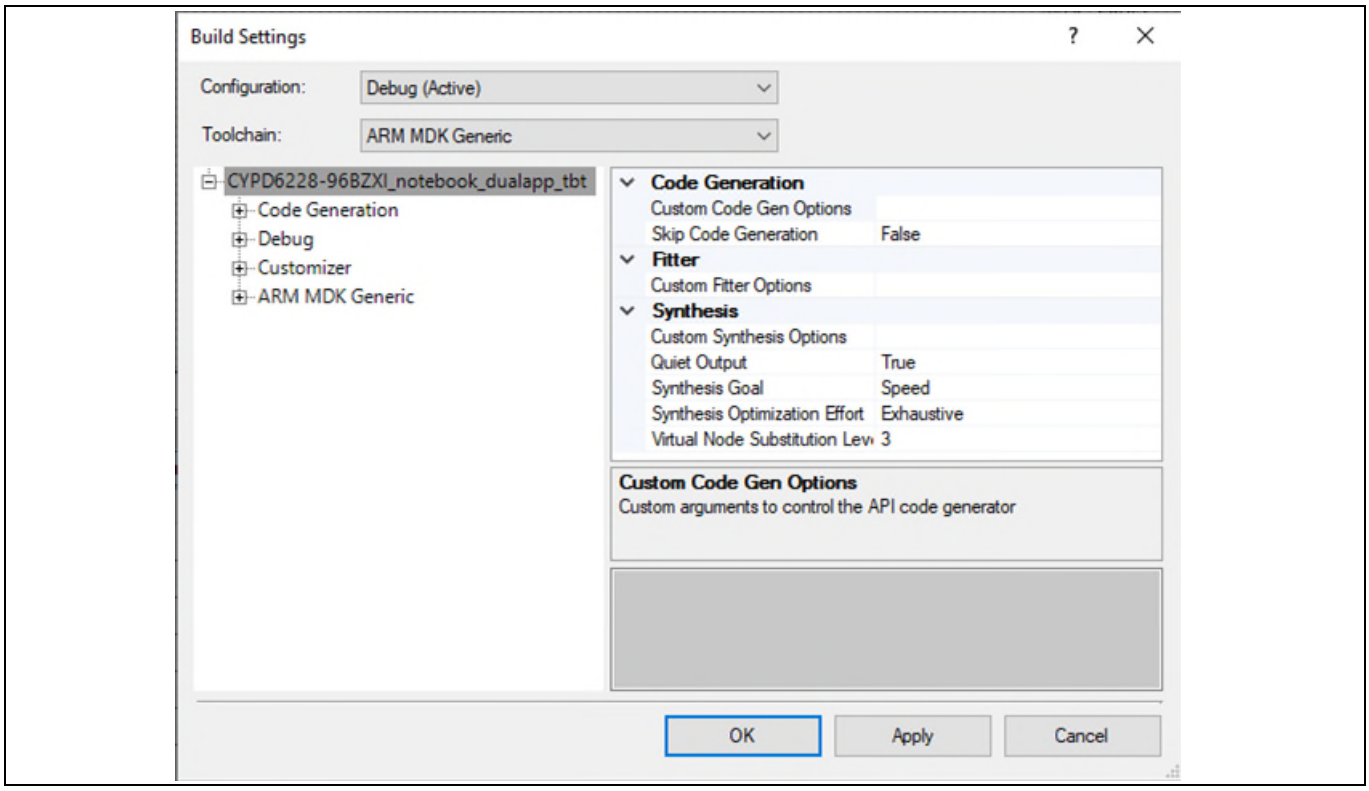


Figure 9 Project Build Settings Dialog

6. You may receive a pop-up window asking for permission to make the project file writeable. Select **Yes** to allow the project build to go through. The complete build process may take about two to three minutes. The output window at the bottom of the IDE will look as shown in Figure 10 at the end of the build process.

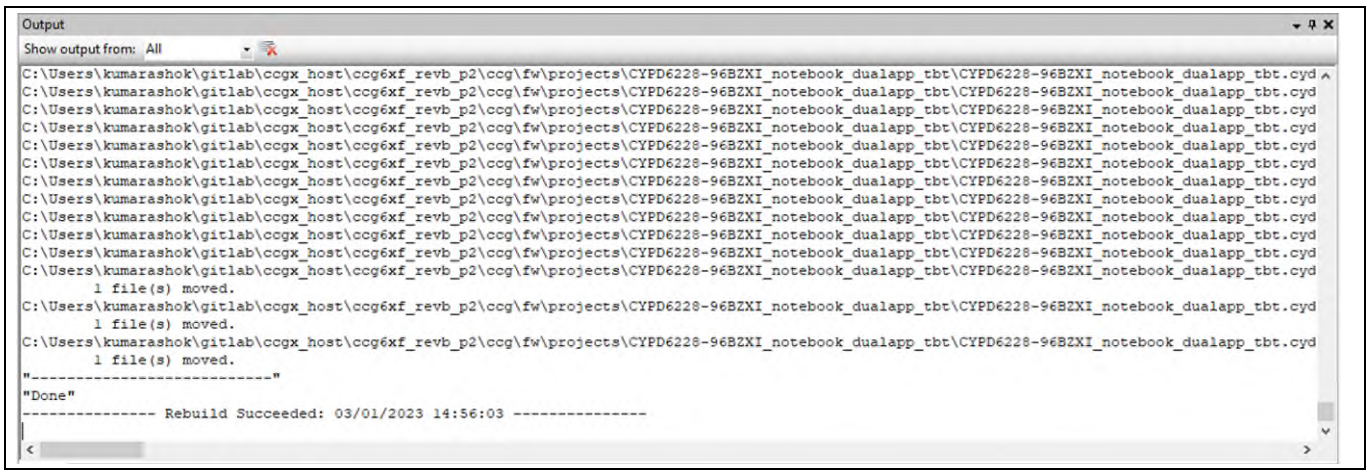


Figure 10 Output Window after the build is complete

7. The build process may throw errors because of missing Creator components as shown in Figure 11. This error can be resolved by right-clicking on the Workspace name and selecting Update Components. A Component Update Tool dialog as shown in Figure 12 will pop up. Select Next and then Finish to allow PSoC™ Creator to download the latest component versions. This update sequence will only need to be completed on the first Host SDK project that you compile.

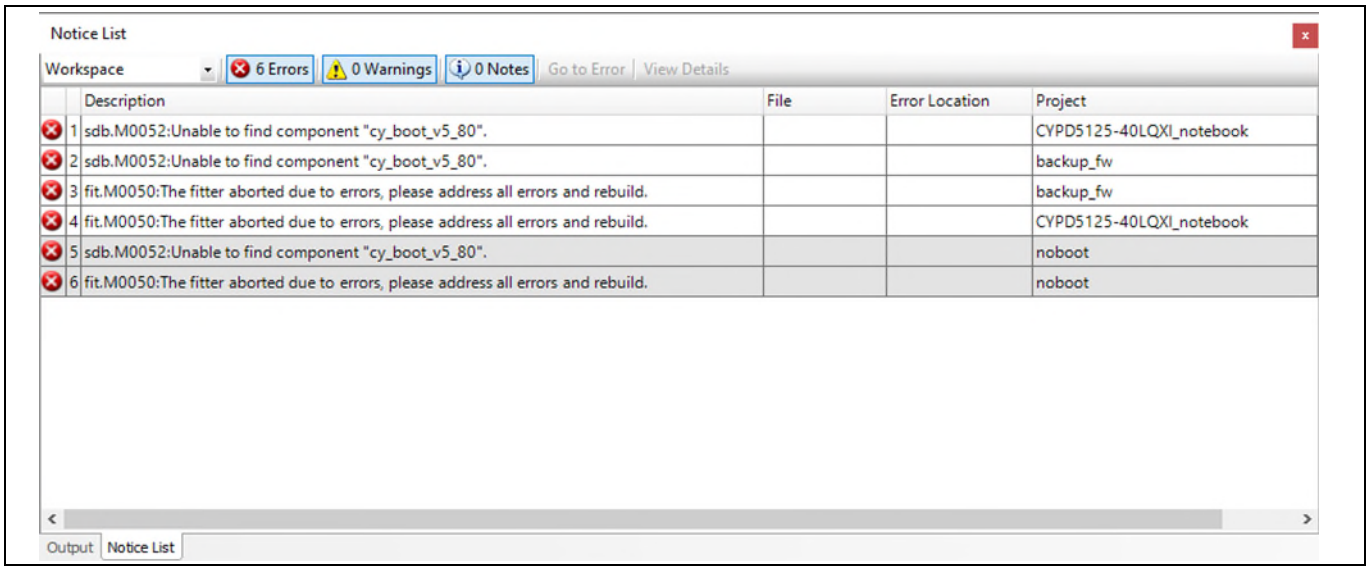


Figure 11 CCG5 build error because of missing Creator components

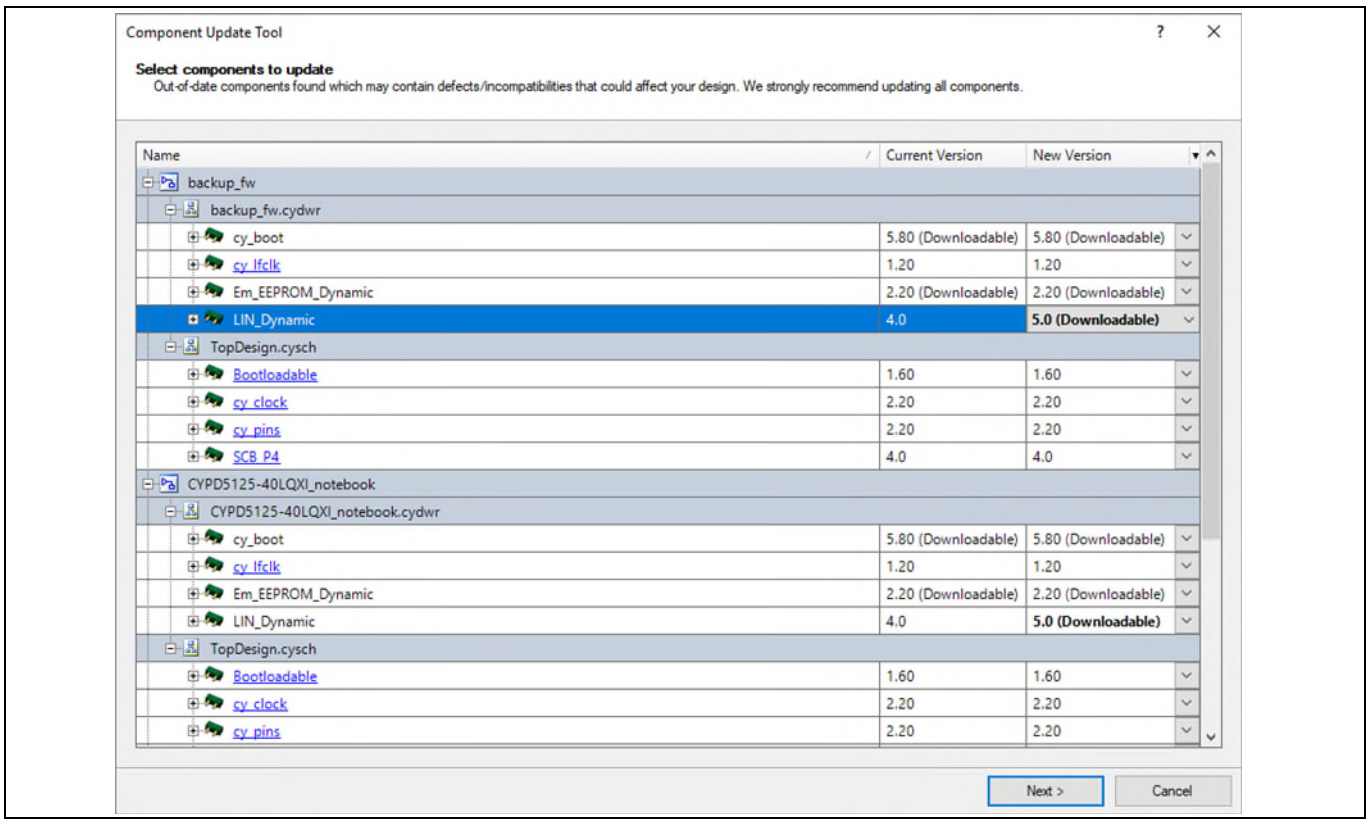


Figure 12 Component update tool dialog showing available updates

8. If the post-build script which combines the backup and primary firmware images throws an error as shown in Figure 13, please add the PSoC™ Creator binary folder to the PATH variable in the system environment and then restart PSoC™ Creator. See the section Environment updates for more details.

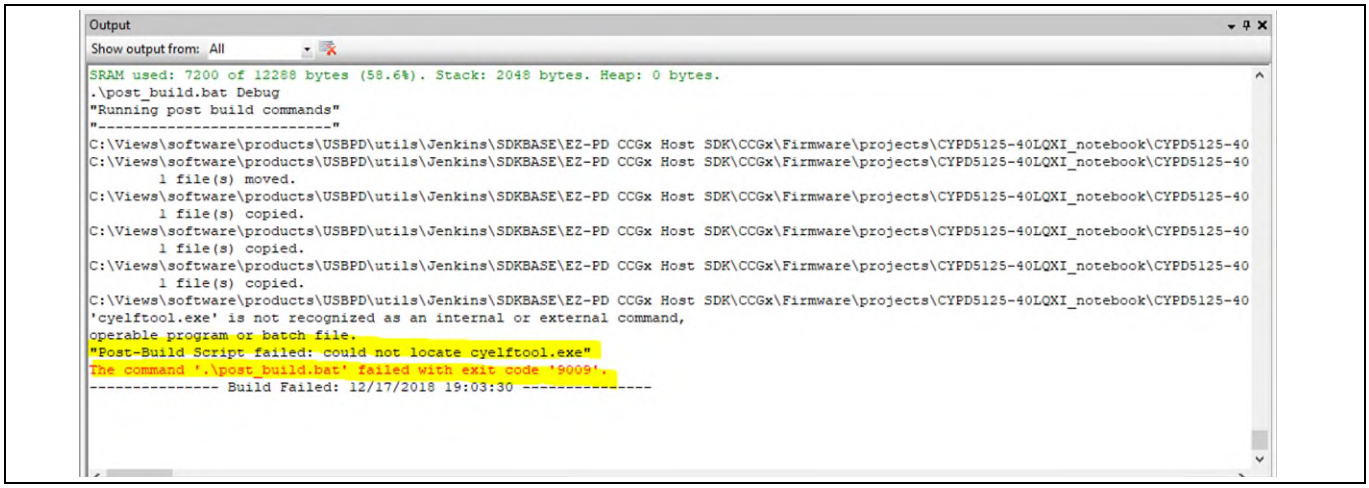


Figure 13 Post-build script error in reference projects

9. Now, navigate to the project folder using Windows Explorer to locate the compiled firmware binaries. Navigate to the CYPD6228-96BZXI_notebook_dualapp_tbt.cydsn\CortexM0\ARM_MDK_Generic\Debug folder for the output files. The following three files are the most important output files generated by the build process:

a) ***CYPD6228-96BZXI_notebook_dualapp_rpl.hex***: This is an SWD programmable binary file in the Intel Hex format that combines the bootloader as well as the notebook port controller firmware application.

b) ***CYPD6228-96BZXI_noteboo_dualapp_rpl_2.cyacd***: This binary file contains the full-featured firmware application that can be loaded to the CCG5 device flash through the I2C interface. The format of the file is documented here. The EZ-PD Configuration Utility accepts firmware binaries in the cyacd format and programs them to the CCG6DF device.

c) ***CYPD6228-96BZXI_notebook_dualapp_rpl_1.cyacd***: This binary file contains the limited-feature backup firmware application that can be loaded to the CCG device flash through the I2C interface. The format of the file is documented here. The EZ-PD™ Configuration Utility accepts firmware binaries in the cyacd format and programs them to the CCG6DF device.

3.2.3 Programming CCGx using the EZ-PD™ Configuration Utility

If the CCG6DF evaluation board is being used, the firmware binaries built using the above procedure can be loaded onto the CCG6DF device using the EZ-PD™ Configuration Utility. This section provides step-by-step instructions for updating the firmware with the EZ-PD™ Configuration Utility. See the EZ-PD™ Configuration Utility user guide for more details.

1. Power up the CCG6DF evaluation board using the 20 V power adapter and connect a USB cable from the Mini-B connector to the host PC.
2. Wait for driver detection and binding for the USB-Serial controller on the board. The driver for this controller can be obtained by searching on Windows Update. Once the driver binding is successful, a “USB-Serial (Single Channel) Vendor 1” device will be listed under ‘Universal Serial Bus Controllers’ in the **Device Manager** window. See Figure 14 for the expected device listing.

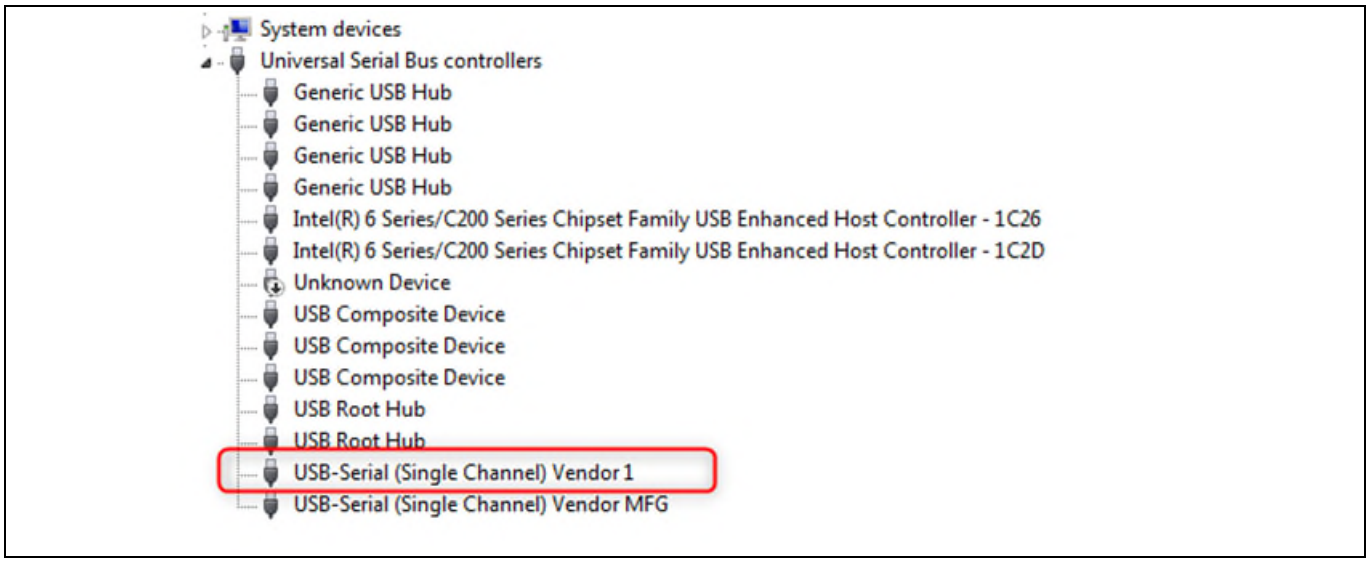


Figure 14 Device Manager View showing USB-Serial Bridge device

3. If the automatic driver installation does not succeed, you can download and use the USB Serial Windows Driver Installer. See the USB-Serial Windows Driver installation guide document as well.

4. Open the EZ-PD™ Configuration Utility GUI. If the device driver binding is successful, the GUI should report a device connected on the lower border of the UI as shown in Figure 15.

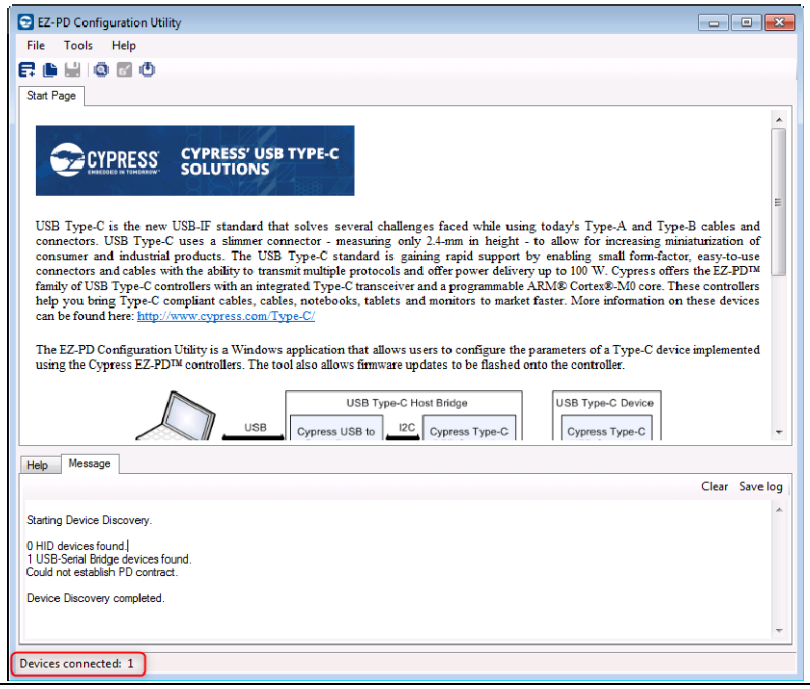


Figure 15 Configuration Utility detecting the connected CCG5 evaluation board

5. Go to **Tools → Firmware Update**. The utility detects and identifies the device at this stage. A firmware update dialog appears at the end of this process (see Figure 16).
6. When you click on the required node (**Notebook**) in the device tree, the UI displays information about the CCGx device and the current firmware running on it.

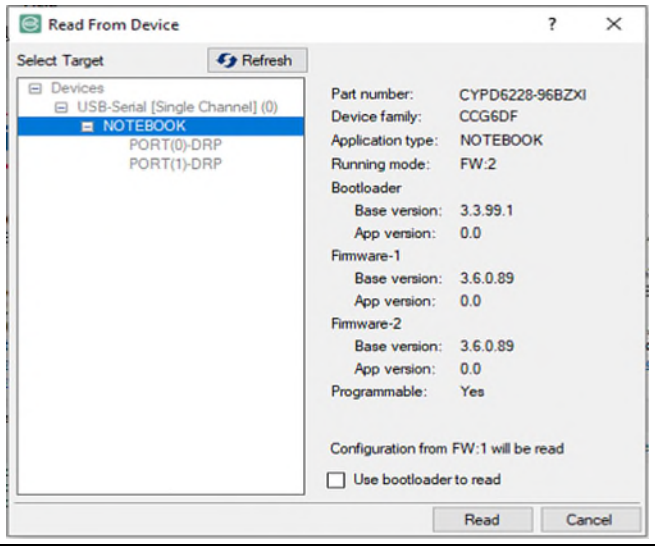


Figure 16 Firmware update dialog

7. Navigate to the folder containing the firmware binaries generated during the firmware build, and select the *CYPD6228-96BZXI_notebook_dualapp_rpl_1.cyacd* and *CYPD6228-96BZXII_notebook_rpl_2.cyacd* files in the two firmware path options in the dialog.

8. Check the "*Use bootloader to flash*" option so that both banks of firmware can be updated in one step; and click on the **Program** button to start the firmware update. The full update will take about 30 seconds.
9. Once the update process is complete, use the **Tools** > **Read from Device** option to bring up a dialog that can show the current firmware version. If the firmware project from the SDK is used without any changes, the new running firmware version should match the version of the firmware downloaded.
10. Because the Firmware-2 binary is a full-featured application, the CCG5 bootloader is designed such that it loads the Firmware-2 whenever it is present (independent of the firmware version).

3.3 Updating EZ-PD™ CCGx configuration

3.3.1 Configuration parameters

The CCGx firmware design uses a configuration table, which specifies several parameters that control device functionality. These parameters include:

- The VDM responses sent by the device for various PD messages.
- The power profiles supported by the device as a provider and as a consumer.
- The port roles supported by the device (Source/Sink/Dual Role).
- Enable/disable flags and parameters that control the various firmware features.

These parameters are stored in a configuration table so that they can be updated/customized without updating the firmware. The utility provides an interactive GUI through which all of the contents of the configuration table can be updated.

Table 3 shows the list of configuration parameters that are relevant to the various Notebook/Host PD port controller solutions. Refer to the EZ-PD™ Configuration Utility User Manual for a description of the various UI screens presented by the utility to configure these parameters. Each UI screen also provides tool-tips that guide you through the process of defining the firmware configuration.

Please note that changing the Source PDO configuration is not recommended while using the EVKs because the default settings in the solution correspond to the actual kit hardware design.

Table 3 List of configuration parameters for Host applications

Configuration parameter	Description	Default value	Change allowed
Device parameters			
Part number	CCG MPN used in the project.	<As per Project>	Select the part number matching the application.
Manufacturer Info	String used in the PD 3.0 Manufacturer Info message.	"Infineon"	Can be changed.
Vendor ID	Vendor ID value used in the ID Header VDO, Manufacturer Info, and Source Capabilities Extended messages.	0x04B4	Can be changed.
Config table major version	Config table format revision. Used by firmware to ensure compatibility.	2	The version of the configuration table format. Leave this with the default value.

Configuration parameter	Description	Default value	Change allowed
PD revision	USB PD spec revision supported by the firmware. **Note:** This parameter only affects the UI behavior and does not change firmware functionality. The parameter value should be based on the actual feature selection in the firmware binary.	3.1	Set based on firmware features.
Cable discovery count	Number of cable discovery attempts to be made	0x14	Can be changed in the range of 1 to 20.
Dead battery event mask (0x)	Default value of HPI event mask that will be applied to all PD ports on the device.	0x0000	Can be changed.
Port information			
Product ID	Product ID used in Product VDO, Source Capabilities Extended, and Manufacturer Info messages.	<As per Project>	The USB Product ID assigned for the product.
Port role	Power role configuration for the port: Sink, Source, or Dual Role	Dual role	Change not recommended for a host platform.
Default port role	Determines the start-up state used by the Type-C state machine when the port role is DRP. **Note:** Changing this parameter does not guarantee the port role used when there is a Type-C connection.	Source	Change not recommended for a host platform.
DRP preferred role	Enable Source or Sink preference on a DRP port.	Source	Change not recommended for a host platform.
Current level	Choose the Rp current source used as a Type-C source.	3 A	Can be changed.
Is source battery connected	Legacy parameter: Don't care in current stack.	No	Don't care
Is sink battery connected	Legacy parameter: Don't care in current stack.	No	Don't care

Configuration parameter	Description	Default value	Change allowed
Sink USB suspend	Determines the No USB Suspend bit in the request message sent by the CCGx device.	No	Can be changed.
Sink USB communication	Determines the USB Communications capable bit in PD messages.	Yes	Change not recommended.
Rp-Rd Toggle	Whether the DRP port should use an automatic Rp-Rd toggle sequence.	Yes	Should be YES for DRP ports.
Rp supported	Bit-map representing supported Rp current source values. The actual Rp value can be chosen from this list at run-time through the HPI interface.	Default, 1.5 A and 3.0 A	Can be changed
Is source externally powered	Whether the power source is externally powered.	No	Can be changed based on system design.
Is sink externally powered	Whether the power sink is externally powered. **Note:** Should be set to the same value as Is source externally powered.	No	Can be changed based on system design.
Cable discovery enable	Whether cable discovery is enabled. Power supply will be limited to 3 A when cable discovery is not enabled.	Yes	Change not recommended
Dead battery enable	Whether Type-C state machine should directly transition to Attached.SNK state if VBus is present on power-up.	Yes	Change not recommended
Error recovery enable	Whether Type-C error recovery sequence is enabled.	Yes	Change not recommended
DR_SWAP response	Default response to be used for DR_SWAP requests.	ACCEPT	Can be changed
PR_SWAP response	Default response to be used for PR_SWAP requests.	ACCEPT	Can be changed
VCONN_SWAP response	Default response to be used for VCONN_SWAP requests to become VConn source. Swap to allow the port partner to become VConn source is always accepted.	ACCEPT	Can be changed

Configuration parameter	Description	Default value	Change allowed
FRS Enable	Whether Fast Role Swap as source (sink → source) is enabled.	FRS Receive	Can be set to FRS receive or None. FRS transmit is not supported.
VConn retain	Whether the port should continue to supply VConn even if the AMA or Cable reports that VConn supply is not required.	No	Can be changed.
Device IDs			
USB host support	Whether the port implements a USB host controller.	Yes	Should be Yes for desktops and notebooks.
USB device support	Whether the port implements a USB device.	No	Should be No for desktops and notebooks.
Modal operation supported	Whether any Type-C alternate modes are supported as an UFP.	Yes	Set this to No if no UFP alternate modes are required.
USB Vendor Id	Vendor ID. Copied from the Device parameters screen.	0x04B4	This gets changed through Port Information node.
Product type (UFP)	Product type to be reported as an UFP.	Peripheral	Can be changed
Product type (DFP)	Product type as a DFP.	Host	Change not recommended
USB-ID compliance XID	XID value to be used in the Cert Stat VDO.	0	Can be changed
USB Product ID	Product ID. Copied from the Device parameters screen.	<As per project>	This gets changed through Port Information node.
BCD device	BCD Device value to be reported in Product VDO.	0	Can be changed
UFP VDO 1			
UFP VDO version	Version Number of the VDO	1.3	Should not be changed
USB 2.0 device capability	Is device USB 2.0 capable	Capable	Can be changed
USB 3.2 device capability	Is device USB 3.2 capable	Capable	Can be changed
USB 4 device capability	Is device USB 4 capable	Capable	Can be changed
TBT3 mode enable	Support of TBT3 data mode	Yes	Can be changed
Other data mode enable	Enable/disable Alternate Modes that reconfigure the signals on the USB Type-C connector (except for TBT3)	No	Can be changed

Configuration parameter	Description	Default value	Change allowed
Custom mode enable	Enable/disable Alternate Modes that do not reconfigure the signals on the USB Type-C connector.	No	Can be changed
USB Highest Speed	The USB Highest Speed field shall indicate the port's highest signaling capability.	Gen3	Can be changed
UFP VDO 2			
USB4 Min Power (W)	Minimum power in watts required to function in USB4 operation	15	Can be changed
USB4 Max Power (W)	Maximum power in watts required to function in USB4 operation	15	Can be changed
USB3 Min Power (W)	Minimum power in watts required to function in USB3 operation	15	Can be changed
USB3 Max Power (W)	Maximum power in watts required to function in USB3 operation	15	Can be changed
DFP VDO			
DFP VDO version	Version Number of the VDO	1.2	Should not be changed
USB 2.0 host capability	Is host USB 2.0 capable	Capable	Can be changed
USB 3.2 host capability	Is host USB 3.2 capable	Capable	Can be changed
USB 4 host capability	Is host USB 4 capable	Capable	Can be changed
Port Number	Unique port number to identify a specific port on a multi-port device	0	Can be changed
Source PDO 0			
Dual role power	Both power source and sink functions are supported.	Yes	Can be changed
USB suspend support	USB link suspend is supported	No	Can be changed
USB communication capable	USB host or device function is supported	Yes	Can be changed
Data role swap	Data role swap is supported	Yes	Can be changed
Unchunked extended messages supported	Device support send and receive extended messages in a single unchunked message	Yes	Should not be changed

Configuration parameter	Description	Default value	Change allowed
Peak current (0x)	Additional peak current source capability. See USB PD Spec for encoding	0	Can be changed
Voltage (mV)	Output voltage level in mV units	5000	Can't be changed
Maximum current (mA)	Maximum current that can be sourced in mA units	3000	Can be changed
Sink PDO			
Sink PDO 0	First sink PDO: Must be a 5 V fixed supply PDO.	5 V @ 0.9 A	Current can be changed.
Sink PDO 1	Second sink PDO.	7 to 21 V @ 0.9 A	Can be changed.
SCEDB Configuration: Source Capabilities Extended Data Block			
VID	Vendor ID	0x04B4	Can be changed
PID	Product ID	<As per Project>	Can be changed
XID	USB-IF assigned ID for the product.	0	Can be changed
FW Version	Firmware version	1	Can be changed
HW Version	Hardware version	1	Can be changed
Voltage regulation load step slew rate (mA/µs)	Load variation allowed per us by the voltage regulator.	150	Not allowed
Voltage regulation load step magnitude (%loC)	Range of load variation allowed (%) allowed.	25	Can be changed
Holdup Time (ms)	Time in ms for which the output voltage remains in holdup on interruption of input supply.	0	Can be changed
LPS compliant	IEC power supply compliance status.	No	Can be changed
PS1 compliant	IEC power supply compliance status.	No	Can be changed
PS2 compliant	IEC power supply compliance status.	No	Can be changed
Low touch current EPS	Touch current status bit.	No	Can be changed
Ground pin	Whether Ground pin is supported.	Supported	Can be changed
Touch temp	Specifies IEC standard used to determine surface temperature.	Default: IEC 60950-1.	Can be changed
Source Inputs - External supplies	External supply type.	Constrained external supply	Can be changed
Source Inputs - Internal batteries	Whether batteries are present.	No	Can be changed

Configuration parameter	Description	Default value	Change allowed
Hot swappable batteries	Number of hot swappable batteries.	0	Can be changed
Fixed batteries	Number of fixed batteries.	0	Can be changed
Source PD Power	PD power supply capability in Watts.	60	Can be changed. Should match Source PDOs.
SKEDB configuration: Sink capabilities extended data block			
VID (0x)	The 16-bit Vendor ID (VID) assigned to the Source's vendor by the USB-IF	4B4	Can't be changed
PID (0x)	The 16-bit Product ID (PID) assigned by the Source's vendor	<As per Project>	Can't be changed
XID (0x)	Value provided by the USB-IF to assign to product	00000000	Can be changed
FW Version (0x)	Firmware version number	1	Can be changed
HW Version (0x)	Hardware version number	1	Can be changed
SKEDB Version (0x)	SKEDB Version	1	Can't be changed
Load step slew rate (mA/µs)	Indicates the Load Step Slew Rate and Magnitude that this Sink prefers	150	Can be changed
Percent overload	Percent overload in 10% increments. Values higher than 250 are clipped to 250%.	0	Can be changed
Overload period (ms)	Overload period in ms. Value is stored in table as 20 ms unit	0	Can be changed
Duty cycle	Duty cycle in 5% increments	0	Can be changed
VBUS Voltage drop tolerance	Can tolerate VBUS Voltage drop	No	Can be changed
LPS Compliant	Whether sink is LPS compliant	No	Can be changed
PS1 Compliant	Whether sink is PS1 compliant	No	Can be changed
PS2 Compliant	Whether sink is PS2 compliant	No	Can be changed
Touch Temp	The IEC standard used to determine the surface temperature of the Sink's enclosure	Default	Can be changed
Hot swappable batteries	Number of hot swappable batteries (0..4)	0	Can be changed

Configuration parameter	Description	Default value	Change allowed
Fixed batteries	Number of fixed batteries (0..4)	0	Can be changed
PPS charging supported	Ability of Sink to use a PPS Source for fast charging	No	Can be changed
VBUS powered	Ability of Sink to be sourced by Vbus	Yes	Can be changed
Mains powered	Ability of Sink to be sourced by an external mains power supply	No	Can be changed
Battery powered	Ability of Sink to be sourced by a battery	Yes	Can be changed
Unlimited battery powered	Ability of Sink to be sourced by a battery with essentially infinite energy (e.g. a car battery)	No	Can be changed
Sink minimum PDP	The Minimum PDP required by the Sink to operate without consuming any power from its Battery(s) should it have one.	5	Can be changed
Sink operational PDP	The PDP the Sink requires to operate normally. For Sinks with a Battery, it is the PDP Rating of the charger supplied with it or recommended for it.	15	Can be changed
Sink maximum PDP	The Maximum PDP the Sink can consume to operate and charge its Battery(s) should it have one.	20	Can be changed
Peak current[1]			
Percentage overload (%)	Percentage of overload allowed.	0	Can be changed
Overload period (ms)	Period in ms of the rolling average time window for overloading.	0	Can be changed
Duty cycle (%)	Percentage of time for which overloading persists within the rolling window.	0	Can be changed
Vbus voltage drop	Whether there can be additional VBus drop due to overload.	No	Can be changed

[1] Three sets of peak current values are to be provided as per USB PD specification.

Configuration parameter	Description	Default value	Change allowed
DP mode parameters			
Modes supported	Supported DisplayPort pin assignments.	CDE	Can be changed
Mux Control	Whether CCGx is responsible for configuring the Type-C data switch to enable/disable display connection.	Controlled by CCGx	Can be changed based on hardware design.
Mode trigger	Whether CCGx firmware should automatically enter DisplayPort mode and enable display output when a Type-C display is detected.	Automatic	Can be changed
Preferred DP Mode	Legacy parameter: Don't care	4-lane DP	Not applicable for Notebook/Desktop designs.
Overvoltage protection			
Enable	Whether OVP (as sink and source) is enabled.	Enabled	Can be changed
OVP Threshold	Max. allowed excess voltage as percentage of expected value.	20%	Can be changed
Debounce period	Debounce period in µs.	10 µs	Can be changed
Retry count	Number of recovery and retry attempts to be made before disabling Type-C connection with a faulty device.	2	Can be changed
Overcurrent protection			
OCP enable	Whether OCP (as source) is enabled.	Yes	Can be changed
OCP Threshold (%)	Max. allowed overload as percentage of maximum expected current.	20%	Can be changed
Debounce period	Debounce period in ms. Used to ignore brief surges and load variations.	10 ms	Can be changed
Retry count	Number of recovery and retry attempts to be made before disabling Type-C connection with a faulty device.	2	Can be changed
OCP Threshold-2 (%)	Reserved for future use	50	Not supported

Configuration parameter	Description	Default value	Change allowed
Debounce period-2 (ms)	Reserved for future use	1 ms	Not supported
Sense resistance (milli ohm)	Sense resistor impedance	5	Change is not recommended
Tuning resistance (ohm)	Current sense tuning resistor impedance	0	Not supported
Short-circuit protection[2]			
Enable	Whether SCP (as source) is enabled.	Enabled	Can be changed
SCP Threshold (%)	Threshold above nominal operating current to be treated as SC fault	50	Can be changed
Debounce period (µs)	Debounce period (in µs) that should be applied before SC is detected	1	Can be changed
Retry count	Number of recovery and retry attempts to be made before disabling Type-C connection with a faulty device.	2	Can be changed
VConn overcurrent protection[3]			
Enable	Whether VConn OCP is enabled	Yes	Can be changed
Threshold (%)	Reserved for future use.	30	Not supported as of now.
Debounce period (ms)	Debounce period in ms. Used to ignore momentary load variations. **Note:** Setting to this to values greater than 1 ms can cause device damage in case of overload.	1	Change not recommended.
Retry count	Number of VCONN OCP events that are allowed before CCGx shuts down the port completely.	0	Not supported
Over-temperature protection[4]			
Enable	Enable OTP protection	Yes	Can be changed

[2] Only supported on the CCG6, CCG6DF, and CCG6SF device families.

[3] Only supported on the CCG5, CCG5C, CCG6, CCG6DF, and CCG6SF device families.

[4] Only supported on the CCG6SF and CCG6DF device families.

Configuration parameter	Description	Default value	Change allowed
Thermistor type 1	Type of thermistor is being used (Negative Temperature Co-efficient, Positive Temperature Co-efficient or Internal BJT sensing).	Internal BJT	Can be changed
Cutoff value 1	Temperature at which OTP cutoff is to be performed	100	Can be changed
Restart value 1	Temperature at which system operation can be resumed	85	Can be changed
Debounce period (ms)	Period (in ms) that should be applied before over-temperature is detected	0	Can be changed
Reverse current protection[5]			
Enable	Whether RCP (as source) is enabled	Yes	Can be changed
Retry count	Number of recovery and retry attempts to be made before disabling Type-C connection with a faulty device.	2	Can be changed
Charging configuration - Source[6]			
Charging mode	Legacy charging mode supported: None, DCP or Apple charging	BC 1.2	Can be changed
QC 4.0 enabled	Reserved for future use	No	Not supported in host projects.
Thunderbolt host configuration			
Enable	Enable thunderbolt host configuration settings	Yes	Can be changed
Preferred power role	Preferred power role to be enforced using PR-SWAP.	No preference	Can be changed
Preferred data role	Preferred data role to be enforced using DR-SWAP.	DFP	Change is not recommended
HPD Handling	Handle the HPD signal via GPIO or I2C	Virtual HPD	Can be changed
VPRO capable	Host supports VPRO docks	Yes	Can be changed

[5] Only supported on the CCG6, CCG6SF, and CCG6DF device families.

[6] Only supported on the CCG5, CCG5C, and CCG6 device families.

Configuration parameter	Description	Default value	Change allowed
SBU MUX configuration	SBU MUX configuration to be used (SBU MUX pass-through, SBU MUX without polarity change, Full SBU MUX configuration)	SBU MUX pass-through[7]	Can be changed
USB4 data role	USB4 data role	Dual-Role	Can be changed
USB3 data role	USB3 data role	Dual-Role	Can be changed
USB4 host support	Enable/Disable USB4 host support	Enable	Can be changed
Support TBT tunneling	TBT tunneling over USB4 support	Yes	Can be changed
Support DP tunneling	DP tunneling over USB4 support	Yes	Can be changed
Support PCIe tunneling	PCIe tunneling over USB4 support	Yes	Can be changed
Non-thunderbolt MUX	Non-thunderbolt MUX is used in the design	No	Feature not supported
Alternate Mode 0			
SVID#0 (0x)	Thunderbolt 3 alternate mode SVID	8087	Can be changed. Remove this if Thunderbolt 3 is not supported.
Supported in DFP (checkbox)	Thunderbolt 3 alternate mode is supported by DFP	Checked	Can be changed
Supported in UFP (checkbox)	Thunderbolt 3 alternate mode is supported by UFP	Checked	Can be changed
Alternate Mode 1			
SVID#0 (0x)	Display Port alternate mode SVID	FF01	Can be changed. Remove this if Display Port is not supported.
Supported in DFP (checkbox)	Display Port alternate mode is supported by DFP	Checked	Can be changed
Supported in UFP (checkbox)	Display Port alternate mode is supported by UFP	Not checked	Not supported in host projects.
Custom Alt Mode configuration			
Enable	Enable custom alternate mode	Yes	Can be changed
Alternate mode SVID#0 (0x)	Custom alternate mode SVID	0	Can be changed
Supported in DFP	Custom Port alternate mode is supported by DFP	No	Can be changed

[7] SBU MUX configuration is only applicable to CCG5, CCG5C, and CCG6 devices.

Configuration parameter	Description	Default value	Change allowed
Supported in DFP	Custom Port alternate mode is supported by UFP	No	Can't be changed
Custom host configuration			
Accessory mode	Accessory mode enable/disable mode	Enable	Can be changed
Rp detach	Enable/disable disconnect detect mechanism using Rp in Sink role	Enable	Can be changed
Power threshold	Minimal power (in mW) required to turn FET ON if source provides at least this value	4500	Can be changed
Request max power	Request max current provided by the port partner instead of the current mentioned in the sink capabilities	No	Can be changed
Accept PR_SWAP if externally powered	Whether to accept PR_SWAP even if there is an external powered bit is set	No	Can be changed
Source PDO selection algorithm	Source PDO selection algorithm: Default, Highest power, Highest current, Highest voltage	Default	Can be changed
Intel platform configuration			
Platform selection	ICL/TGL/RKL platform	TGL	Can be changed
Retimer count	Number of retimers per PD port	1	Can be changed
I2C retimer #1 address (0x)	I2C address of retimer #1	40	Can be changed
SoC I2C address (0x)	I2C address to be used for the SoC slave interface	50	Can be changed
SoC Mux initialization delay (0x)	SoC Mux initialization delay in milli seconds	19	Can be changed
SoC Mux config delay (0x)	Delay between two Mux update events	E	Can be changed
Clear HPD_IRQ ACK method	Clear implicitly: SoC will implicitly clear its own HPD_IRQ bit after it is set to 1 by the PD Controller. Clear explicitly: PD Controller will explicitly clear the HPD_IRQ bit back to 0 before setting it to 1 again.	Implicit	Can be changed
AMD configuration			

Configuration parameter	Description	Default value	Change allowed
AMD platform type	AMD SoC used in the design (Renoir, Rembrandt_A0, Rembrandt_B0)	Rembrandt_B0	Can be changed
Mode	Indicates the preferred AMD handling mode (Polling, Interrupt)	Interrupt	Can be changed
I2C Address (0x)	Configuring AMD APU I2C Slave address for selected port	54	Can be changed
Index	AMD APU index parameter for selected port	0	Can be changed
Preferred data role	Preferred data role to be enforced using DR-SWAP.	DFP	Change is not recommended
Preferred power role	Preferred power role to be enforced using PR-SWAP.	No preference	Can be changed
Retimer type	Retimer type which is used in AMD design (No Retimer used, Retimer PI3DPX1205A, Retimer PS8828, Retimer PS8830, Autoselection of PS8828 or PS8830 Retimers, KB800x (B0 silicon), KB800x (B1 silicon)).	Autoselection	Can be changed
I2C Retimer address (0x)	Retimer address for selected port	8	Can be changed
USB4 supported	Indicates whether AMD APU supports Extended Write Payload format	True	Can be changed
USB4 data role	USB4 data role	Dual-Role	Can be changed
USB3 data role	USB3 data role	Dual-Role	Can be changed
USB4 host support	Enable/Disable USB4 host support	Enable	Can be changed
Support TBT tunneling	TBT tunneling over USB4 support	Yes	Can be changed
Support DP tunneling	DP tunneling over USB4 support	Yes	Can be changed
Support PCIe tunneling	PCIe tunneling over USB4 support	Yes	Can be changed
User parameters			
Parameters 1 to 8	Reserved for customer specific usage.	00	Can be changed

3.3.2 EZ-PD™ Configuration Utility

EZ-PD™ Configuration Utility is a Windows application that allows the user to define the configuration parameter values through a set of UI screens and convert the configuration into a binary format that can be programmed onto the device or embedded into the firmware source.

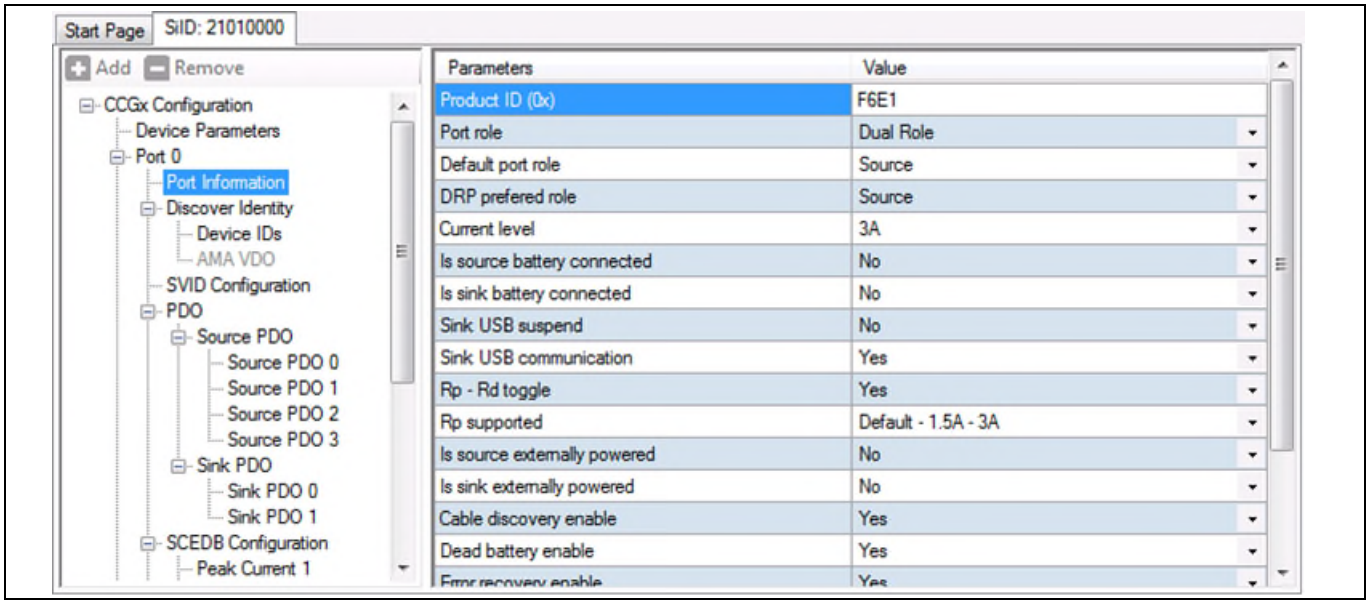


Figure 17 PD Port Configuration using EZ-PD™ Configuration Utility

Figure 17 shows a snapshot of the UI screens used for configuring the CCG5 firmware as an example. The entire device configuration is completed by navigating through all the nodes shown on the left side window of the UI.

The various fields are interrelated and should be updated to be mutually consistent. After all the parameters are defined, click on the 'Save' button (or go to File → Save As) to save a copy of the configuration to the disk. The configuration is stored in the form of an XML file. The utility also generates two additional output files that help the user in deploying the configuration.

1. A *cyacd* file is generated, which can be used to program the new configuration data to the device. The EZ-PD™ utility itself uses the *cyacd* file for device programming.
2. A *.c* file is generated, which can be included in the firmware project to compile a new binary that embeds the desired configuration. More details on the use of this file are provided in later sections of this guide.

After the configuration is saved, use the **Tools → Configure Device** option to program the configuration to the device. The utility issues a warning if the firmware version is older than the current version, and you have the option of aborting the configuration update at this stage.

4 Customizing the firmware application

As shown in the section Updating EZ-PD™ CCGx configuration, a major part of the CCGx host application functionality can be modified without having to change any of the firmware sources.

Any changes to the hardware design around the CCGx device will, however, require changes to the firmware sources implementing the application. This chapter walks through the process of updating the firmware implementation to work with a different hardware design.

Note: As the firmware sources and reference projects are installed in the **Program Files** folder, it is not recommended that you make changes to the originally installed version of these files. You can create a copy of the Firmware folder from the SDK installation, and use the copy for making any changes. The code example option of *PSoC™* Creator will allow you to make copies of the projects. This will ensure that you have a clean version of the files that you can revert to as well. See the section *Copying the project with PSoC™ Creator* for more details.
Because the target application remains the same, it is expected that the changes are limited to aspects such as the mechanism for voltage selection, FET control, data path MUX/Switch control, and so on. This does not involve changes to the core functionality implemented by the CCGx device.

4.1 Solution structure

The CCGx solution structure is shown in Figure 18. The figure uses the CYPD'6228-96BZXI_notebook_dual workspace as a reference. The source and header files used in the solution are grouped into different folders.

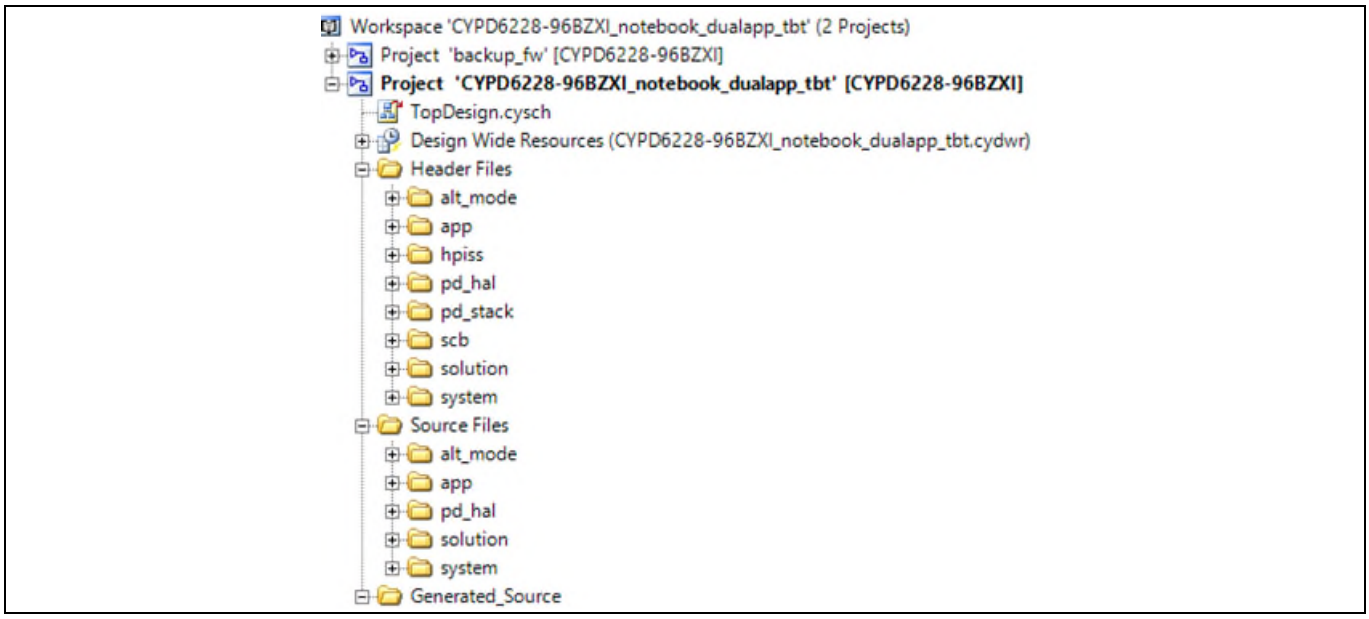


Figure 18 CCG6DF Notebook Solution structure

- **Solution**: The solution folders contain header and source files that provide user configurations, user hardware-specific functions, and custom code modules. It is expected that these files will need to be changed to match the hardware design and requirements for all customer implementations. The solution-level sources include:
  - **config.h**: Header file that enables/disables firmware features and provides macros or function mappings for hardware-specific functions such as FET control and voltage selection.

  - **stack_params.h**: Configuration parameters used by the PD stack and application layers that manage the PD power negotiation. The content of this file is not expected to be changed by users.
  - **alt_modes_config.h**: Header file that selects the alternate modes that are supported by the firmware when CCGx is a Downstream Facing Port (DFP) or Upstream Facing Port (UFP).
  - **solution.h**: Header file providing constant, variable, and function declarations for external hardware controls implemented by CCG5 firmware.
  - **config.c**: This source file contains the default run-time configuration for the CCGx notebook application and has been generated using the EZ-PD™ Configuration Utility.
  - **solution.c**: This source file contains the functions that control the data switch and/or external buck-boost regulators used in the system for data connection and power control.
  - **main.c**: This source file contains the main application entry point.
- **app**: The app folders contain header and source files that implement the device policy decisions such as power contract negotiation roles, port role management, power protection schemes, Vendor Defined Message (VDM) handling, and so on. The default implementation provided in the source form uses the configuration table and runtime customizations provided by the EC to handle these tasks. The files can be updated if there is a need to change the way policy decisions are implemented by the CCG firmware. The app source files include:
  - **app.c**: This is the top-level application source file that connects the PD stack to the alternate modes manager as well as the solution level code.
  - **pdo.c**: This source file implements the Power Data Object (PDO) and Request Data Object (RDO) handlers that define the power contract negotiation rules.
  - **psource.c**: This source file implements the power source-related state machines and tasks.
  - **psink.c**: This source file implements the power sink related state machines and tasks.
  - **swap.c**: This source file implements the various swap request handlers.
  - **vdm.c**: This source file implements the handlers for VDMs received by the CCGx device.
  - **ucsi.c**: This source file implements the UCSI interface which allows the OS policy manager to manage the Type-C port functionality.
  - **battery_charging.c:** This source file implements the BC 1.2 (CDP and DCP) support for the CCGx notebook application.
- **pd_hal**: The pd_hal folder header and source files that implement the low-level drivers for the PD stack. The header file definitions should not be modified as these are used by the PD stack library and conflicting definitions can result in undefined behavior. The pd_hal level sources include:
  - **pdss_mx_hal.c:** The hardware interface file to the PD block.
  - **hal_ccgx.c**: This source file implements various protection tasks, which are specific to the CCG device architecture.
- **alt_mode**: This folder contains header and source files that implement the alternate mode manager functions for when CCG is functioning as DFP and when CCG is functioning as UFP.
- **pd_common**: Since the PD stack is provided in the library form, the pd_common folder only contains header files that provide data structure definitions and function declarations for the PD stack. The header file definitions should not be modified as these are used by the PD stack library and conflicting definitions can result in undefined behavior.
- **hpiss**: Header file which defines the Host Processor Interface related functions.
- **scb**: Header file which defines the generic I2C slave driver used for HPI and other functions.

- **system**: This folder contains the base system-level functionality such as GPIO, soft timer implementation, flash driver, and firmware upgrade handlers. The header file definitions should not be modified as these are used by the PD and HPI stack libraries and conflicting definitions can result in undefined behavior.

4.2 Compile time options

The notebook port controller application supports a set of features that can be enabled/disabled using compile time options. These compile time options are set in the config.h header file that you can find under the solution folder, and are summarized in Table 4. These parameters are a super-set of the parameters supported across all the reference applications. See the application-specific sections below for the set of parameters supported by each application.

Table 4 Selectable firmware features in config.h

Pre-processor switch	Description	Values
APP_VBUS_SRC_FET_ ON_P1	Function/Macro to turn on the port 1 source FET.	Map to a function or macro which disables the source (provider) FET associated with PD port 1.
APP_VBUS_SRC_FET_ OFF_P1	Function/Macro to turn off the port 1 source FET.	Map to a function or macro which enables the source (provider) FET associated with PD port 1.
APP_VBUS_SRC_FET_ ON_P2	Function/Macro to turn on the port 2 source FET. Only applicable for a dual-port solution.	Map to a function or macro which disables the source (provider) FET associated with PD port 2.
APP_VBUS_SRC_FET_ OFF_P2	Function/Macro to turn off the port 2 source FET. Only applicable for a dual-port solution.	Map to a function or macro which enables the source (provider) FET associated with PD port 2.
APP_VBUS_SNK_FET_OFF_P1	Function/Macro to turn on the port 1 sink FET.	Map to a function or macro which disables the sink (consumer) FET associated with PD port 1.
APP_VBUS_SNK_FET_ON_P1	Function/Macro to turn off the port 1 sink FET.	Map to a function or macro which enables the sink (consumer) FET associated with PD port 1.
APP_VBUS_SNK_FET_OFF_P2	Function/Macro to turn on the port 2 sink FET. Only applicable for a dual-port solution.	Map to a function or macro which disables the sink (consumer) FET associated with PD port 2.
APP_VBUS_SNK_FET_ON_P2	Function/Macro to turn off the port 2 sink FET. Only applicable for a dual-port solution.	Map to a function or macro which enables the sink (consumer) FET associated with PD port 2.

Pre-processor switch	Description	Values
APP_DISCHARGE_FET_ON_P1	Function/Macro to turn on the port 1 VBus discharge circuit.	Map to a function or macro which enables the VBus discharge circuit associated with PD Port 1. The internal discharge FET on the CCG device can be used for this purpose.
APP_DISCHARGE_FET_OFF_P1	Function/Macro to turn off the port 1 VBus discharge circuit.	Map to a function or macro which disables the VBus discharge circuit associated with PD Port 1. The internal discharge FET on the CCG device can be used for this purpose.
APP_DISCHARGE_FET_ON_P2	Function/Macro to turn on the port 2 VBus discharge circuit. Only applicable for a dual-port solution.	Map to a function or macro which enables the VBus discharge circuit associated with PD Port 2. The internal discharge FET on the CCG device can be used for this purpose.
APP_DISCHARGE_FET_OFF_P2	Function/Macro to turn off the port 2 VBus discharge circuit. Only applicable for a dual-port solution.	Map to a function or macro which disables the VBus discharge circuit associated with PD Port 2. The internal discharge FET on the CCG device can be used for this purpose.
CCG_PROG_SOURCE_ENABLE	Select whether power source supports variable (not a discrete set) of VBus voltage settings.	Set to 1 if there is a programmable regulator to provide the power output. Set to 0 if the regulator only supports discrete voltage settings of 5 V, 9 V, 12 V, 15 V and 20 V.
APP_VBUS_SET_VOLT_P1	Function/Macro to select the source voltage (VBus output) on PD port 1. Only required when CCG_PROG_SOURCE_ENABLE is set to 1.	Map to a function or macro which updates the on-board regulator to source the required output voltage.
APP_VBUS_SET_VOLT_P2	Function/Macro to select the source voltage (VBus output) on PD port 2. Only required when CCG_PROG_SOURCE_ENABLE is set to 1.	Map to a function or macro which updates the on-board regulator to source the required output voltage.

Pre-processor switch	Description	Values
APP_VBUS_SET_5V_P1 APP_VBUS_SET_9V_P1 APP_VBUS_SET_12V_P1 APP_VBUS_SET_15V_P1 APP_VBUS_SET_20V_P1	Function/Macro to set the output voltage on Port 1 to the desired level. Only required when CCG_PROG_SOURCE_ENABLE is set to 0.	Map to functions or macros which select the output voltage from the on-board regulator. The implementation for unsupported voltages can be left as NOP.
APP_VBUS_SET_5V_P2 APP_VBUS_SET_9V_P2 APP_VBUS_SET_12V_P2 APP_VBUS_SET_15V_P2 APP_VBUS_SET_20V_P2	Function/Macro to set the output voltage on Port 2 to the desired level. Only required when CCG_PROG_SOURCE_ENABLE is set to 0.	Map to functions or macros which select the output voltage from the on-board regulator. The implementation for unsupported voltages can be left as NOP.
BATTERY_CHARGING_ENABLE	Enable/disable BC 1.2 (CDP/DCP) source support on the USB ports.	Set to 1 to enable BC 1.2 power source support. Only supported on CCG5, CCG5C and CCG6.
SYS_DEEPSLEEP_ENABLE	Enable/disable putting the CCGx device into a low power mode when idle.	Can be set to 0 to save flash space where required.
VBUS_OVP_ENABLE	Enable/disable detection and handling of VBus overvoltage faults.	Recommend setting this to 1 in all cases.
VBUS_OVP_MODE	Select mode of OVP detection and handling.	0 ➔ Not supported. 1 ➔ OVP detection by OV comparator and handling by firmware. 2 ➔ OVP detection by OV comparator and automated hardware based handling. Should be set to 2 for proper operation.
VBUS_OCP_ENABLE	Enable/disable detection and handling of VBus overcurrent faults.	1 ➔ Type-C VBUS OCP enable. 0 ➔ Type-C VBUS OCP disable.
VBUS_OCP_MODE	Select mode of OVP detection and handling.	0 ➔ Use external load switch 1 ➔ Not supported 2 ➔ OCP detection by internal CSA with automatic hardware based handling. 3 ➔ OCP detection by internal CSA with firmware based debounce and handling. Should be set to 3 for proper operation.

Pre-processor switch	Description	Values
VBUS_SCP_ENABLE	Enable/disable detection and handling of VBus short-circuit faults.	1 ➔ Type-C VBUS SCP enable. 0 ➔ Type-C VBUS SCP disable. Only supported on CCG6, CCG6DF and CCG6SF.
VBUS_RCP_ENABLE	Enable/disable detection and handling of VBus Reverse-Current faults.	1 ➔ Type-C VBUS RCP enable. 0 ➔ Type-C VBUS RCP disable. Only supported on CCG6, CCG6DF and CCG6SF.
VCONN_OCP_ENABLE	Enable/disable detection and handling of VConn overcurrent faults.	1 ➔ Type-C VConn OCP enable. 0 ➔ Type-C VConn OCP disable.
OTP_ENABLE	Enable/disable OTP feature	1 ➔ OTP feature enable. 0 ➔ OTP feature disable. Only supported on CCG6DF and CCG6SF.
CCG_UCSI_ENABLE	Enable/disable the UCSI interface.	1 ➔ UCSI interface enable. 0 ➔ UCSI interface disable.
DFP_ALT_MODE_SUPP	Enable/disable the alternate mode discovery and handling state machine when the port managed by CCG5 is a DFP.	1 ➔ Enable alternate modes when CCG5 is DFP. 0 ➔ Disable alternate modes when CCG5 is DFP.
DP_DFP_SUPP	Enable/disable DisplayPort source (DFP_U/DFP_D) functionality.	1 ➔ Enable DP source functionality. 0 ➔ Enable DP sink functionality.
TBT_DFP_SUPP	Enable/disable Thunderbolt alternate mode operation when CCG5 is a DFP. This can only be enabled in system designs that include a Thunderbolt controller from Intel.	1 ➔ Enable TBT alternate mode. 0 ➔ Disable TBT alternate mode.
UFP_ALT_MODE_SUPP	Enable/disable the alternate mode discovery and handling state machine when the port managed by CCG5 is a UFP.	1 ➔ Enable alternate modes when CCG5 is UFP. 0 ➔ Disable alternate modes when CCG5 is UFP.
TBT_UFP_SUPP	Enable/disable Thunderbolt alternate mode operation when CCG5 is a UFP. This can only be enabled in system designs that include a Thunderbolt controller from Intel.	1 ➔ Enable TBT alternate mode. 0 ➔ Disable TBT alternate mode.

Pre-processor switch	Description	Values
RESET_ON_ERROR_ ENABLE	Selects whether to enable / disable CCG device reset on error (watchdog expiry or hard fault)	1 ➔ Enable reset option 0 ➔ Disable reset option
STACK_USAGE_CHECK_ENABLE	Enable/disable periodic checking of available margin in the runtime stack.	1 ➔ Enable stack margin checks 0 ➔ Disable stack margin checks
APP_PD_REV3_ENABLE	Enable/disable PD 3.0 Support	1 ➔ Enable PD 3.0 0 ➔ Disable PD 3.0 Only applicable for CCG6DF and CCG6SF.
APP_USB4_DATA_ENABLE	Enable/Disable USB 4.0 Support	1 ➔ Enable PD 3.0 0 ➔ Disable PD 3.0 Only applicable for CCG6DF and CCG6SF.
ICL_ENABLE	Enable/Disable Intel Ice Lake features	1 ➔ Enable Intel Host Platform specific features 0 ➔ Disable Intel Host Platform specific features
ICL_SLAVE_ENABLE	Enable/Disable Ice Lake PMC Specific support	1 ➔ Enable ICL PMC Specific features 0 ➔ Disable ICL PMC Specific features
BB_RETIMER_ENABLE	Enable/Disable Burnside Bridge retimer Interface	1 ➔ Burnside Bridge Retimer (BBR) interface enable 0 ➔ BBR interface disable
ICL_ALT_MODE_HPI_DISABLED	Enable/Disable Alt Mode Related HPI Commands	1 ➔ Disable alt mode related HPI commands. 0 ➔ Enable alt mode related HPI commands.
ICL_ALT_MODE_EVTS_DISABLED	Enable/Disable Alt Mode related HPI Events	1 ➔ Disable alt mode related HPI Events. 0 ➔ Enable alt mode related HPI Events.
CY_PD_RW_PD_RESP_MSG_DATA	Enable/Disable Generic PD response data HPI command	1 ➔ Enable Generic PD response data. 0 ➔ Disable Generic PD response data. Note: Battery Status and Battery Capability messages are implemented under this feature.

In addition to the user-selectable parameters listed above, the config.h file also defines a set of parameters that configure the behavior of the PD stack and other modules in the CCG6xF firmware. Changes to these parameters are not recommended for optimal operation. These parameters are described in Table 5.

Table 5 Firmware configuration definitions in config.h

Pre-processor switch	Description	Values
CCG_PD_REV3_ ENABLE	Whether to enable PD 3.0 support or not. PD 3.0 operation is recommended. More details in section 4.4.	1 ➔ PD 3.0 support enabled 0 ➔ PD 3.0 support disabled
CCG_FRS_RX_ENABLE	Enable/disable handling of Fast Role Swap requests from a PD 3.0 power source.	1 ➔ Enable FRS receive function. 0 ➔ Disable FRS receive function.
CCG_FRS_TX_ENABLE	Enable/disable generation of Fast Role Swap request as a PD 3.0 power source.	1 ➔ Enable FRS transmit function. 0 ➔ Disable FRS transmit function.
CCG_SINK_ONLY	Configure the CCG device in a Sink-only power role. Used in Backup firmware applications.	1 ➔ CCG implements Sink-only Type-C state machine. 0 ➔ CCG implements DRP Type-C state machine.

4.3 Part number update

The SDK workspaces are geared to work with appropriate applications. But there can be instances where the target part needs to be changed. The following are the steps to be done for changing the part.

4.3.1 Device selector

The part number for the project in the workspace needs to be updated to match the new part. This can be achieved using the device selector. Right-click the project on the Workspace Explorer window and select Device Selector. As shown in Figure 19, use the Device Selector dialog to change the part number. The part number selection can also be done when copying the example project according to the section From Code Examples. If this is already done, then this step can be avoided.

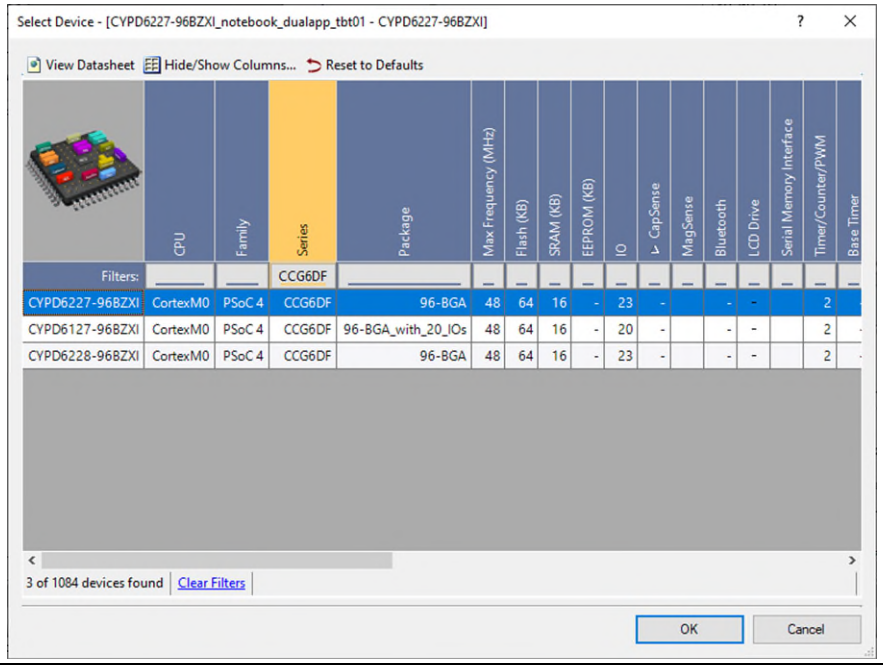


Figure 19 Device selector dialog for changing part number

4.3.2 Bootloader selection

When the application part number is modified, the corresponding bootloader binaries should also be updated. In the schematics tab of the application, double-click the Bootloadable component. Update the bootloader binary files in the Dependencies Tab as shown in Figure 20. Choose the bootloader binary that matches the part number.

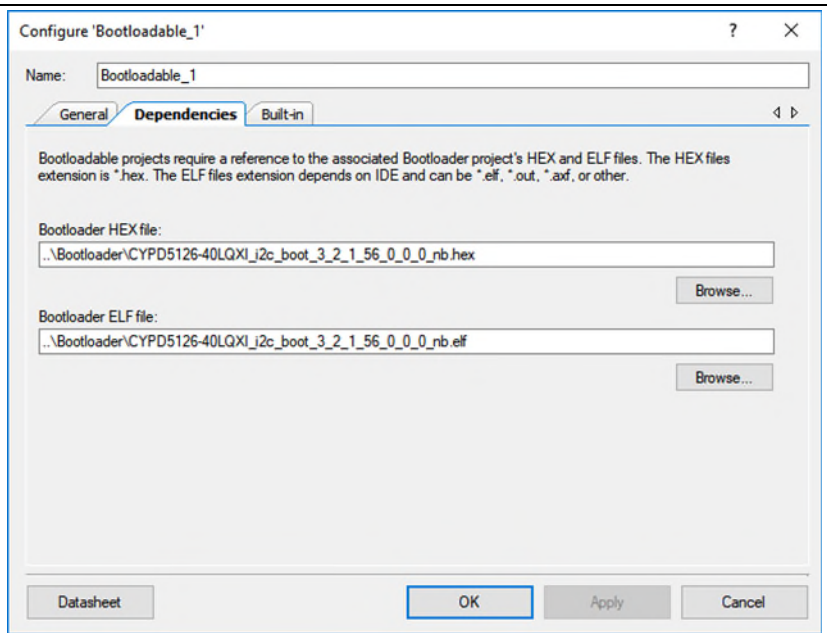


Figure 20 Bootloader binary file update option

CCGx host firmware makes use of two different boot architectures depending on the reference project used. The bootloader usage in the project depends on the boot architecture as well as the application type which is being configured.

4.3.2.1 Bootloader + Asymmetric Dual Binary Architecture

Most of the reference projects in the Host SDK make use of the Bootloader + Asymmetric Dual Binary Architecture. These applications make use of an I2C bootloader which is placed at the base of the device flash and two different application binaries (full-featured primary application and reduced-feature backup application). See section 5.5 for details of the bootloader operation and boot sequence.

4.3.2.1.1 Bootloader in Backup Firmware application

The actual I2C boot-loader for the CCGx device is used in the Backup firmware application. Fully tested pre-compiled boot-loader binaries for each of the CCGx devices are made available in the corresponding Bootloader folder.

The ***i2c_boot.cydsn*** project which is part of the code example folder can be used to compile a new customized boot-loader where required. Figure 20 shows the bootloader binary selection option for the backup firmware project.

4.3.2.1.2 Bootloader in Primary Firmware application

The primary firmware project uses a dummy boot-loader binary which is not added into the final HEX file that is generated. The post build script which runs at the end of the build process replaces the dummy boot-loader with the real boot-loader which it selects from the backup firmware binary.

A pre-compiled binary for the dummy boot-loader is provided in the Bootloader folder under each CCGx project workspace. Because the dummy boot-loader is not really used in the final binaries, there is no need to change this binary.

4.3.2.2 Hybrid Boot Architecture

In this case, the bootloader functionality is integrated into the reduced-feature backup application; and the project only generates the backup and primary binaries.

The CYPD6227-96BZXI_notebook_tbt reference project which makes of this boot architecture in this version of the Host SDK.

4.3.2.2.1 Bootloader in Hybrid Architecture

In this case, a fixed bootloader is not used; and the bootloader binary is refreshed every time the Backup Firmware project is compiled.

The primary application uses the HEX file generated during the backup firmware build as the bootloader.

4.3.3 Build and re-compile

After the part number and bootloader binary files have been updated, the workspace is ready to be compiled and used normally. After the part number has been updated, the firmware cannot be used correctly on the kit and requires the correct part to be used.

4.4 USB PD Specification Revision support

The example applications provided for CCGx devices support USB PD specification revision 3.0 by default. Because the PD 3.0 support requires significant code addition, this leaves little room for the addition of customer-specific code in these applications.

It is possible to gain more space in the CCG device flash by restricting the applications to USB PD Revision 2.0 support. The following steps are required to switch applications between PD 3.0 and PD 2.0 support:

1. Five versions of the PD stack libraries which support different feature sets are provided for each CCGx device family. The libraries can be found in the **lib\<ccg_family>\mdk** folder inside the reference project folder**.**
   a) ccgx_pd3.lib: Stack library with Type-C DRP, USB PD Revision 3.0, and USB4 support enabled.
   b) ccgx_pd.lib: Stack library with Type-C DRP, Try.SRC/Try.SNK, USB PD Revision 2.0, and cable discovery support enabled.
   c) ccgx_pd_lite.lib: Stack library with Type-C DRP and USB PD Revision 2.0 support enabled.
   d) ccgx_pd_snk.lib: Stack library with Type-C Sink operation, USB PD Revision 2.0, and cable discovery support enabled.
   e) ccgx_pd_lite_snk.lib: Stack library with Type-C Sink operation and USB PD Revision 2.0 support enabled.

The appropriate library version can be used to link the firmware application based on the required features. By default, the primary binary uses the ccgx_pd3.lib library and the backup binary use the ccgx_pd_lite_sink.lib library.

2. Because the application code also needs to be customized based on the Type-C/PD stack configuration, a few pre-processor macro values need to be updated so that they match with the library used, as shown in Table 6.

Table 6 Feature enable macro dependency with various stack library versions

Macro	ccgx_pd3	ccgx_pd	ccgx_pd_lite	ccgx_pd_snk	ccgx_pd_lite_snk
CCG_PD_REV3_ENABLE	1	0	0	0	0
CCG_FRS_RX_ENABLE	1	0	0	0	0
CCG_FRS_TX_ENABLE	1	0	0	0	0
CCG_SINK_ONLY	0	0	0	1	1
CCG_TRY_SRC_SNK_DISABLE	0	0	1	Don’t care	Don’t care
CCG_CBL_DISC_DISABLE	0	0	1	0	1
CCG_VCONN_DISABLE	0	0	1	0	1

3. The configuration table contents for the application should be changed based on the specification version to be supported. The **SRC_PDO**, **SNK_PDO**, and **DISCOVER_ID** response parameters in the configuration table have fields that are defined only for PD 3.0. These values should be adjusted as required when switching between PD revisions.

4.5 CYPD6227-96BZXI_notebook_tbt application

The CYPD6227-96BZXI_notebook_tbt application implements a USB4 Dual-Role Device (DRD) port controller for desktop and notebook platforms. Table 7 summarizes the features supported by the primary and backup versions of this firmware solution.

Table 7 CYPD6227-96BZXI_notebook_tbt firmware features

Feature	Support in primary firmware	Support in backup firmware
DRP with Type-C 2.0 spec support	Yes	No
Sink with Type-C 2.0 spec support	No	Yes
Try.SRC and Try.SNK support	Yes	No
USB PD Revision 2.0 support	Yes	Yes
USB PD Revision 3.0 support with Fast-Role Swap Receive	Yes	No
Cable discovery support	Yes	No
USB4 support	Yes	No
DFP Data Role Preference	Yes	Yes
Thunderbolt3 alternate mode as DFP and UFP	Yes	No
DP alternate mode as DFP (DFP_U/DFP_D)	Yes	No
Support for other alternate modes as DFP	Yes	No
Support for alternate modes as UFP	No	No
Tiger lake SoC support	Yes	Partial[8]
Retimer Support	Yes	Partial[9]
HPI commands for firmware update	No	Yes
HPI event notifications (connections, faults, etc.)	Yes	Yes
HPI commands for PD port management	Yes	No
HPI user extension support	Yes	Yes
Deep sleep for power saving	Yes	No
Software Watchdog and Stack Monitoring	Yes	No
VBus OVP Support (source/sink)	Yes	Yes (Sink only)
VBus OCP Support (source only)	Yes	No
VConn OCP Support	Yes	No
VBus SCP Support	Yes	No

[8] SoC Retimer Debug Mode register (0x5D) writes are not supported in this binary.

[9] Retimer Debug Mode register (0x07) writes, Retimer Firmware update features, and Retimer register write retries are not supported in this binary.

Feature	Support in primary firmware	Support in backup firmware
VBus RCP support	Yes	No
OTP support	Yes	No
UCSI 1.1 support	Yes	No

The following sub-sections describe the project structure and implementation in more detail. The primary firmware implementation is described in detail, as the backup firmware is only a sub-set of the primary firmware.

4.5.1 PSoC™ Creator schematic

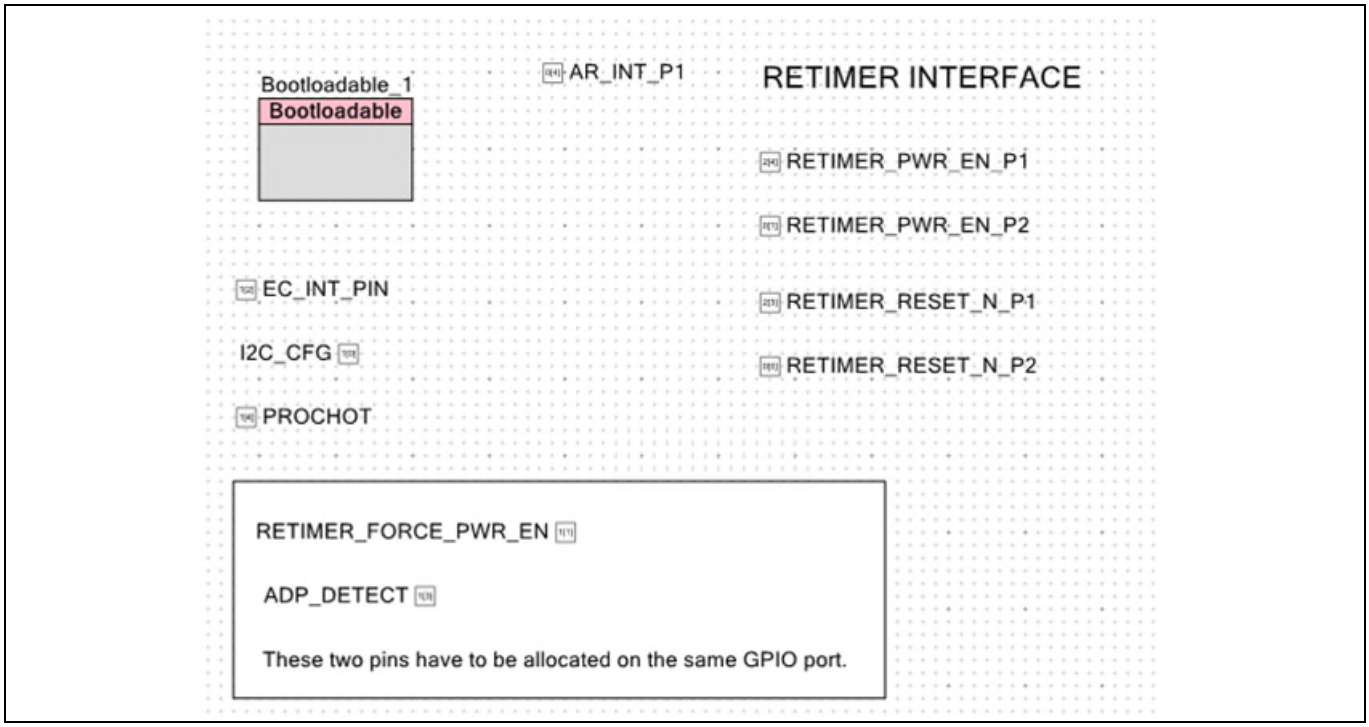


Figure 21 PSoC™ Creator Schematic for CYPD6227-96BZXI_notebook_tbt project

Open TopDesign.cysch file in the PSoC™ Creator project. Schematic contains hardware resources used in the application. The schematic elements are split across multiple sheets, and Figure 21 shows all the active elements together.

In CCG6DF/CCG6SF projects, the system clock and I2C interface configuration is done automatically by the device start-up code. Hence, these components are not shown in the schematic. In addition to the GPIO interfaces shown above, the CCG6DF USB4 DRD project makes use of three sets of SCB (I2C) interfaces:

1. HPI Interface: I2C slave interface which allows Embedded Controller (EC) in the system to monitor/control the CCG device operation.
2. SoC Interface: I2C slave interface which allows the Intel Platform SoC to identify the PD connection state.
3. Retimer Interface: I2C master interface which is used by CCG6DF to configure the Burnside Bridge Retimer based on the PD connection state.

The selection of some of these elements is fixed due to the capabilities of the CCG6xF device. Table 8 points out the changes allowed in the schematic design.

Note: *There is a similar file in the noboot.cydsn project folder as well. If debugging is being used, the schematic dependent changes should be replicated there as well.*

Table 8 Schematic elements in CCG6xF dual-single-port notebook design

Schematic element	Description	Changes allowed
Bootloadable_1	This is a software block which interacts with the boot-loader on the CCG6DF device.	Change not recommended. If the backup binary has been fixed, it can be used here.
EC_INT_PIN	Output interrupt signal from CCG6DF to the Embedded Controller.	Change not recommended.
I2C_CFG	Control signal used to define the I2C slave address used in the HPI interface.	Change not recommended.
AR_INT_P1	Output interrupt signal from CCG6xF Port#1 to the SoC	Can be changed
AR_INT_P2	Output interrupt signal from CCG6xF Port#2 to the SoC	IO is not used and can be removed.
PROCHOT	Output signal used to signal PROCHOT from CCG6xF	Can be changed
OC_FAULT	Output signal used to signal VBUS OCP fault to the Embedded Controller	IO is not used and can be removed.
RETIMER_FORCE_PWR_EN	Input signal used to signal Retimer force power enable from Embedded Controller to CCG6xF	Can be changed subject to the constraint that it should be placed in the same GPIO port as ADP_DETECT.
ADP_DETECT	Input signal used to signal ADP Detect from Embedded Controller to CCG6xF	Can be changed subject to the constraint that it should be placed in the same GPIO port as RETIMER_FORCE_PWR_EN
RETIMER_PWR_EN_P1	Output signal used to signal P#1 Retimer Power Enable	Can be changed
RETIMER_PWR_EN_P2	Output signal used to signal P#2 Retimer Power Enable	Can be changed
RETIMER_RESET_N_P1	Output signal used to signal P#1 Retimer Reset Enable	Can be changed
RETIMER_RESET_N_P2	Output signal used to signal P#2 Retimer Reset Enable	Can be changed

Closely associated with the Schematic is the Design Wide Resources (DWR) view, which maps each schematic element to a pin, clock, or hardware block on the CCG6DF device. Open the CYPD6227-96BZXI_notebook_tbt.cydwr file to see the DWR settings for the project.

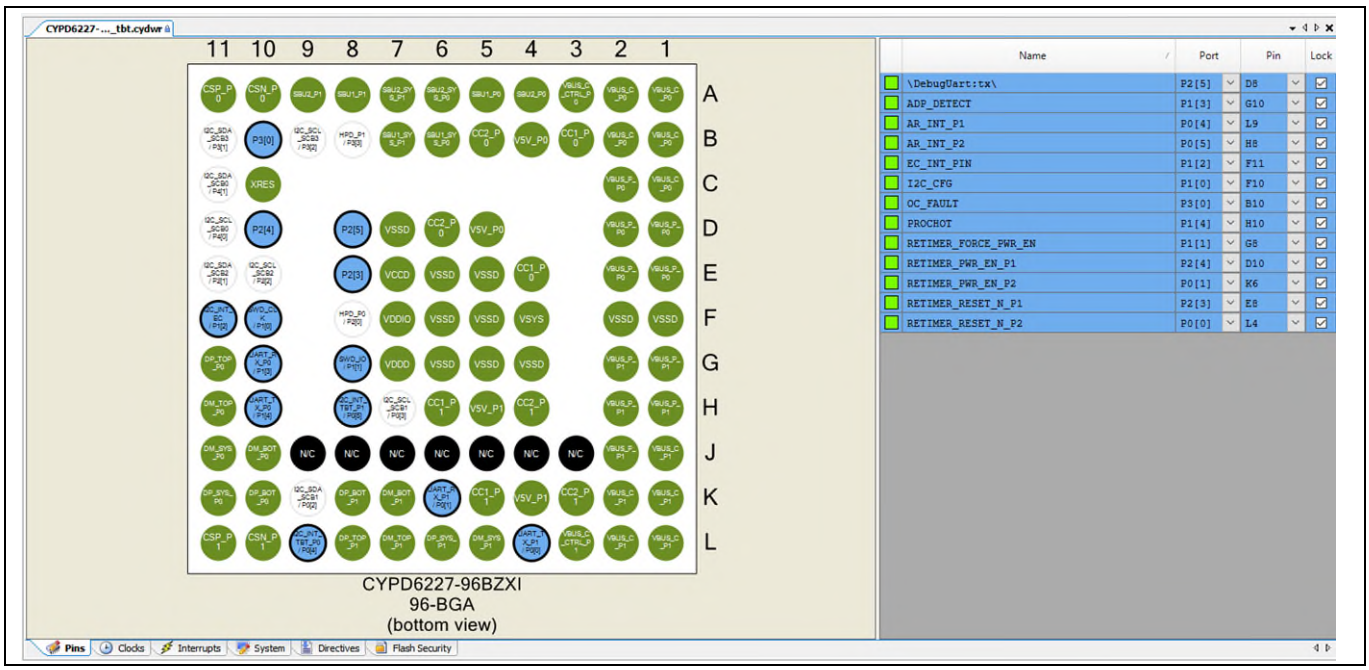


Figure 22 Design Wide Resource (DWR) view of CYPD6227-96BZXI_notebook_tbt project

As shown in Figure 22, the DWR view has several tabs, which configure aspects such as pin mapping, interrupt mapping, clock selection, flash security, and so on. It is recommended that you restrict any changes to the DWR to the pin mapping view. Do not change the clock, interrupt, system, or flash configurations. Even in the pin mapping editor, the changes should be subject to the constraints outlined in Table 8.

4.5.2 Compile time options

The CYPD6227-96BZXI_notebook_tbt application supports a set of features that can be enabled/disabled using compile time options. These compile time options are set in the config.h header file that you can find under the solution folder, and are summarized in Table 9.

Note: While features can be enabled/disabled as desired, changes to the mode selected for various protection schemes are not recommended.

Table 9 Compile time options in CYPD6227-96BZXI_notebook_tbt application

Pre-processor switch	Description	Values
SYS_DEEPSLEEP_ENABLE	Enable/disable putting the CCG6F device into a low power mode when idle.	Can be set to 0 to save flash space where required.
VBUS_OVP_ENABLE	Enable/disable detection and handling of VBus overvoltage faults.	Recommend setting this to 1 in all cases.

Pre-processor switch	Description	Values
VBUS_OVP_MODE	Select mode of OVP detection and handling.	0 ➔ OVP detection using the ADC comparator. 1 ➔ OVP detection by OV comparator and handling by firmware. 2 ➔ OVP detection by OV comparator and automated hardware based handling. Should be set to 2 for proper operation.
VBUS_OCP_ENABLE	Enable/disable detection and handling of VBus overcurrent faults.	1 ➔ Type-C VBUS OCP enable. 0 ➔ Type-C VBUS OCP disable.
VBUS_OCP_MODE	Select mode of OVP detection and handling.	0 ➔ Using external OCP hardware 1 ➔ Internal OCP with neither software debounce nor automatic FET control. 2 ➔ Internal OCP with automatic FET control by hardware when an OCP event is detected. 3 ➔ Internal OCP with software debounce using delay in milliseconds from the configuration table. Should be set to 3 for proper operation.
VCONN_OCP_ENABLE	Enable/disable detection and handling of VConn overcurrent faults.	1 ➔ Type-C VConn OCP enable. 0 ➔ Type-C VConn OCP disable.
VBUS_SCP_ENABLE	Enable/disable detection and handling of short-circuit faults.	1 ➔ VBUS SCP feature enable. 0 ➔ VBUS SCP feature disable.
VBUS_RCP_ENABLE	Enable/disable detection and handling of reverse current faults.	1 ➔ VBUS RCP feature enable. 0 ➔ VBUS RCP feature disable.
OTP_ENABLE	Enable/disable the detection and handling of the over-temperature condition.	1 ➔ Enable OTP feature. 0 ➔ Disable OTP feature.
DFP_ALT_MODE_SUPP	Enable/disable the alternate mode discovery and handling state machine when the port managed by CCG is a DFP.	1 ➔ Enable alternate modes when CCG6DF is DFP. 0 ➔ Disable alternate modes when CCG6DF is DFP.
DP_DFP_SUPP	Enable/disable DisplayPort source (DFP_U/DFP_D) functionality.	1 ➔ Enable DP source functionality. 0 ➔ Disable DP source functionality.

Pre-processor switch	Description	Values
TBT_DFP_SUPP	Enable/disable Thunderbolt alternate mode operation when CCG is a DFP.	1 ➔ Enable TBT alternate mode support in DFP role. 0 ➔ Disable TBT alternate mode support in DFP role.
UFP_ALT_MODE_SUPP	Enable/disable the alternate mode discovery and handling state machine when the port managed by CCG is a UFP.	1 ➔ Enable alternate modes when CCG6DF is UFP. 0 ➔ Disable alternate modes when CCG6DF is UFP.
TBT_UFP_SUPP	Enable/disable Thunderbolt alternate mode operation when CCG is a UFP.	1 ➔ Enable TBT alternate mode in UFP role. 0 ➔ Disable TBT alternate mode in UFP role.
RESET_ON_ERROR_ ENABLE	Selects whether to enable / disable CCG device reset on error (watchdog expiry or hard fault)	1 ➔ Enable reset option 0 ➔ Disable reset option
CCGX_POWER_ROLE_SINK_ONLY	Enable/disable the DRP	1 ➔ Enable PD Sink only 0 ➔ PD Port is DRP Used in backup application.
APP_PD_REV3_ENABLE	Enable/disable PD 3.0 Support	1 ➔ Enable PD 3.0 0 ➔ Disable PD 3.0 Used in backup application.
APP_USB4_DATA_ENABLE	Enable/Disable USB 4.0 Support	1 ➔ Enable USB4 negotiation 0 ➔ Disable USB4 negotiation
ICL_ENABLE	Enable/Disable Intel Ice Lake features	1 ➔ Enable Intel Ice Lake features 0 ➔ Disable Intel Ice Lake features
ICL_SLAVE_ENABLE	Enable/Disable Ice Lake PMC Specific support	1 ➔ Enable ICL PMC Specific features 0 ➔ Disable ICL PMC Specific features
BB_RETIMER_ENABLE	Enable/Disable Burnside Bridge retimer Interface	1 ➔ BBR interface enable. 0 ➔ BBR interface disable.
ICL_ALT_MODE_HPI_DISABLED	Enable/Disable Alt Mode Related HPI Commands	1 ➔ Disable alt mode related HPI commands. 0 ➔ Enable alt mode related HPI commands.
ICL_ALT_MODE_EVTS_DISABLED	Enable/Disable Alt Mode related HPI Events	1 ➔ Disable alt mode related HPI Events. 0 ➔ Enable alt mode related HPI Events.

4.5.3 On-chip resource usage

The CYPD6227-96BZXI_notebook_tbt reference application uses the following on-chip resources for various functionalities.

1. Watchdog: The Watchdog is used as a generic timer for task scheduling and as a mechanism to trigger device reset in case of firmware lock-up.
2. SCB resources
   a) SCB0 is used for HPI slave (I2C) implementation.
   b) SCB1 is used as I2C master to control the Retimer.
   c) SCB2 is used as SoC slave (I2C) implementation.
3. TCPWM resources are not used and are available.
4. USB PD resources
   a) ADC block

      The single ADC available is used for VBus measurement so as to track voltage changes and detect disconnection of a power source. The AMUXB input of the ADC is connected internally to a divided version of VBus for measurement. AMUXA input can be used for other purposes on a time-shared basis.

      ADC block along with internal BJT based temperature sensing is used to detect the OTP condition.

   b) Comparators

      The OV comparator is used to detect overvoltage conditions by comparing a divided version of VBus against a programmable threshold.

      The UV comparator is unused and can be enabled as required.

      The VSYS_DET comparator is used to detect changes to the VSYS supply.

   c) Signal MUX and switch

      The SBU switch is used to connect the Type-C side SBU1 and SBU2 pins to the Intel Platform SoC.

      The DPDM MUX is used to connect the DP_TOP/DM_TOP or DP_BOT/DM_BOT pins to the DP_SYS/DM_SYS pins to enable USB 2.0 data connection.

4.5.4 Secondary extension

The CYPD6227-96BZXI_notebook_tbt project makes use of a Hybrid Boot architecture where the bootloader (firmware update) functionality is embedded into the backup firmware itself. This means that the backup firmware itself cannot be updated in the field (if the backup binary is corrupted, the device cannot function anymore).

It is recommended that users who make use of this architecture write protect the backup binary before deploying it on their systems for production. One limitation of write protecting the backup binary is that the PD port configuration parameters for this binary (configuration table) cannot be modified.

A special mechanism is provided to override the configuration table embedded in the write-protected backup binary. This is called the secondary extension project. A sample implementation of the secondary extension which overrides the config parameters of the backup binary can be found in the workspace.

If this configuration table override is not required, the secondary extension can be removed; thereby freeing up about 2 KB of flash space.

4.5.5 EZ-PD™ CCG6SF support

A common workspace is used to generate the binaries for CCG6DF as well as CCG6SF devices. This approach is chosen because a sizable portion of the stack and driver source code on these devices has been moved into ROM. The common ROM code requires that shared data structures be defined for a dual-port part, and will automatically adjust if the actual part in use supports only one PD port.

Python-based post-build scripts are used to generate separate binaries for use on CCG6DF and CCG6SF devices. The following binaries are generated as part of the build process:

1. CYPD6127-48LQXI_notebook_icl.hex: CCG6SF based single-port Thunderbolt3 Port Controller for Ice Lake Platforms.
2. CYPD6127-48LQXI_notebook_rkl.hex: CCG6SF based single-port USB4 Port Controller for Rocket Lake + Maple Ridge Platforms.
3. CYPD6127-48LQXI_notebook_tgl.hex: CCG6SF based single-port USB4 Port Controller for Tiger Lake Platforms.
4. CYPD6227-96BZXI_notebook_icl.hex: CCG6DF based dual-port Thunderbolt3 Port Controller for Ice Lake Platforms.
5. CYPD6227-96BZXI_notebook_rkl.hex: CCG6DF based dual-port USB4 Port Controller for Rocket Lake + Maple Ridge Platforms.
6. CYPD6227-96BZXI_notebook_tgl.hex: CCG6DF based dual-port USB4 Port Controller for Tiger Lake Platforms.

The difference between the binaries for Ice Lake, Tiger Lake, and Rocket Lake platforms is only in the customized configuration table.

While the configuration table settings used in CCG6DF and CCG6SF binaries are different, the main difference is applied automatically by the firmware by identifying the PD part that is in use.

4.6 CYPD6227-96BZXI_notebook_dualapp_tbt application

The CYPD6227-96BZXI_notebook_dualapp_tbt code example is functionally equivalent to the CYPD6227-96BZXI_notebook_tbt example described in section CYPD6227-96BZXI_notebook_tbt application.

The only difference between the two examples is that this uses the standard Bootloader + Dual asymmetric binary boot architecture which is used by all other host projects.

4.7 CYPD6227-96BZXI_notebook_dualapp application

The CYPD6227-96BZXI_notebook_dualapp application implements a Notebook PD Port Controller for desktop and notebook platforms. Table 10 summarizes the features supported by the primary and backup versions of this firmware solution.

Table 10 CYPD6227-96BZXI_notebook_dualapp firmware features

Feature	Support in primary firmware	Support in backup firmware
DRP with Type-C 2.0 spec support	Yes	No
Sink with Type-C 2.0 spec support	No	Yes
Try.SRC and Try.SNK support	Yes	No
USB PD Revision 2.0 support	Yes	Yes
USB PD Revision 3.0 support with Fast-Role Swap Receive	Yes	No
Cable discovery support	Yes	No
USB4 support	No	No
DFP Data Role Preference	No	No
Thunderbolt3 alternate mode as DFP and UFP	No	No
DP alternate mode as DFP (DFP_U/DFP_D)	Yes	No
Support for other alternate modes as DFP	Yes	No
Support for alternate modes as UFP	No	No
Tiger lake SoC support	No	No
Retimer Support	No	No
HPI commands for firmware update	Yes	Yes
HPI event notifications (connections, faults, etc.)	Yes	Yes
HPI commands for PD port management	Yes	No
HPI user extension support	Yes	Yes
Deep sleep for power saving	Yes	No
Software Watchdog and Stack Monitoring	Yes	No
VBus OVP support (source/sink)	Yes	Yes (Sink only)
VBus OCP support (source only)	Yes	No
VConn OCP support	Yes	No
VBus SCP support	Yes	No
VBus RCP support	Yes	No
OTP support	Yes	No
UCSI 1.1 support	Yes	No

The following sub-sections describe the project structure and implementation in more detail. The primary firmware implementation is described in detail, as the backup firmware is only a sub-set of the primary firmware.

4.7.1 PSoC™ Creator schematic

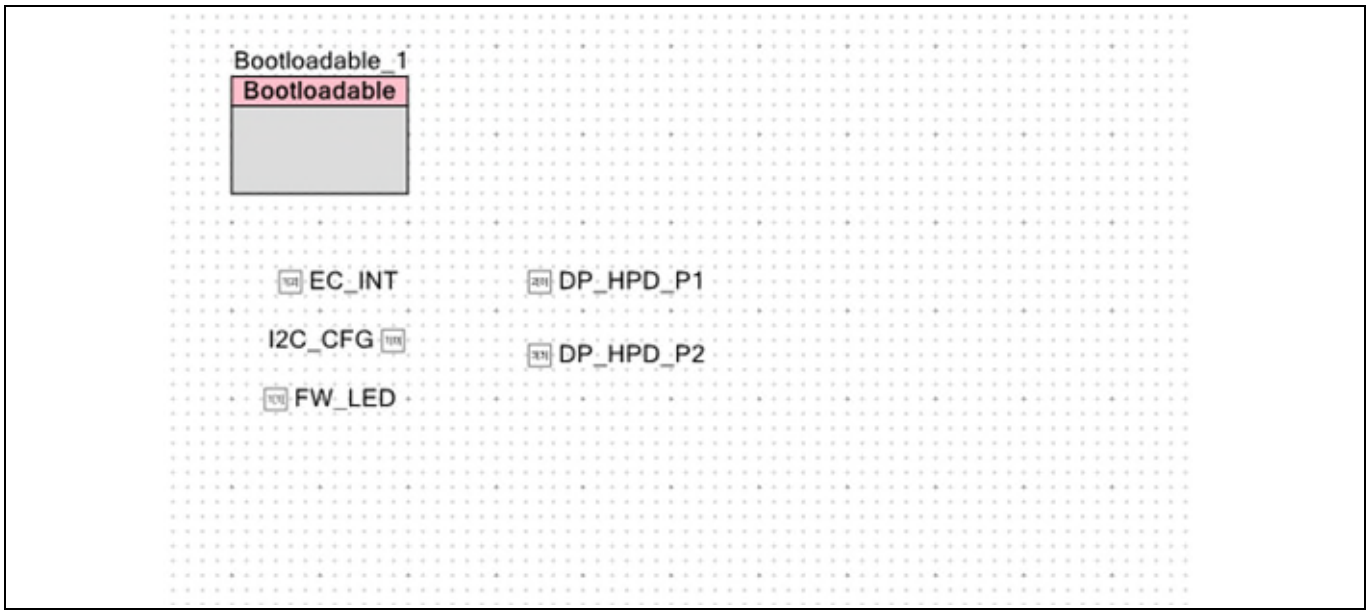


Figure 23 PSoC™ Creator Schematic for CYPD6227-96BZXI_notebook_dualapp project

Open TopDesign.cysch file in the PSoC™ Creator project. Schematic contains hardware resources used in the application. The schematic elements are split across multiple sheets, and Figure 23 shows all the active elements together.

In CCG6DF/CCG6SF projects, the system clock and I2C interface configuration is done automatically by the device start-up code. Hence, these components are not shown in the schematic. In addition to the GPIO interfaces shown above, the CCG6DF Notebook project makes use of two sets of SCB (I2C) interfaces:

1. HPI Interface: I2C slave interface which allows Embedded Controller (EC) in the system to monitor/control the CCG device operation.
2. Regulator / MUX Interface: I2C master interface used by CCG to configure the buck-boost voltage source and Type-C Signal MUX used in the design.

The selection of some of these elements is fixed due to the capabilities of the CCG6xF device. Table 11 points out the changes allowed in the schematic design.

Note: There is a similar file in the noboot.cydsn project folder as well. If debugging is being used, the schematic-dependent changes should be replicated there as well.

Table 11 Schematic Elements in CYPD6227-96BZXI_notebook_dualapp design

Schematic element	Description	Changes allowed
Bootloadable_1	This is a software block which interacts with the boot-loader on the CCG6DF device.	Should not be changed.

Schematic element	Description	Changes allowed
EC_INT	Output interrupt signal from CCG6DF to the Embedded Controller.	Should not be changed as the bootloader uses this pin.
I2C_CFG	Control signal used to define the I2C slave address used in the HPI interface.	Should not be changed as the bootloader uses this pin.
FW_LED	Pin toggled by the firmware to indicate device operation.	Can be changed/disabled.
DP_HPD_P1	HotPlug Detect pin used to connect to DisplayPort controller for Port-1 in the system.	Should not be changed.
DP_HPD_P2	HotPlug Detect pin used to connect to DisplayPort controller for Port-2 in the system.	Should not be changed.

Closely associated with the Schematic is the Design Wide Resources (DWR) view, which maps each schematic element to a pin, clock, or hardware block on the CCG6DF device. Open the CYPD6227-96BZXI_notebook_dualapp.cydwr file to see the DWR settings for the project.

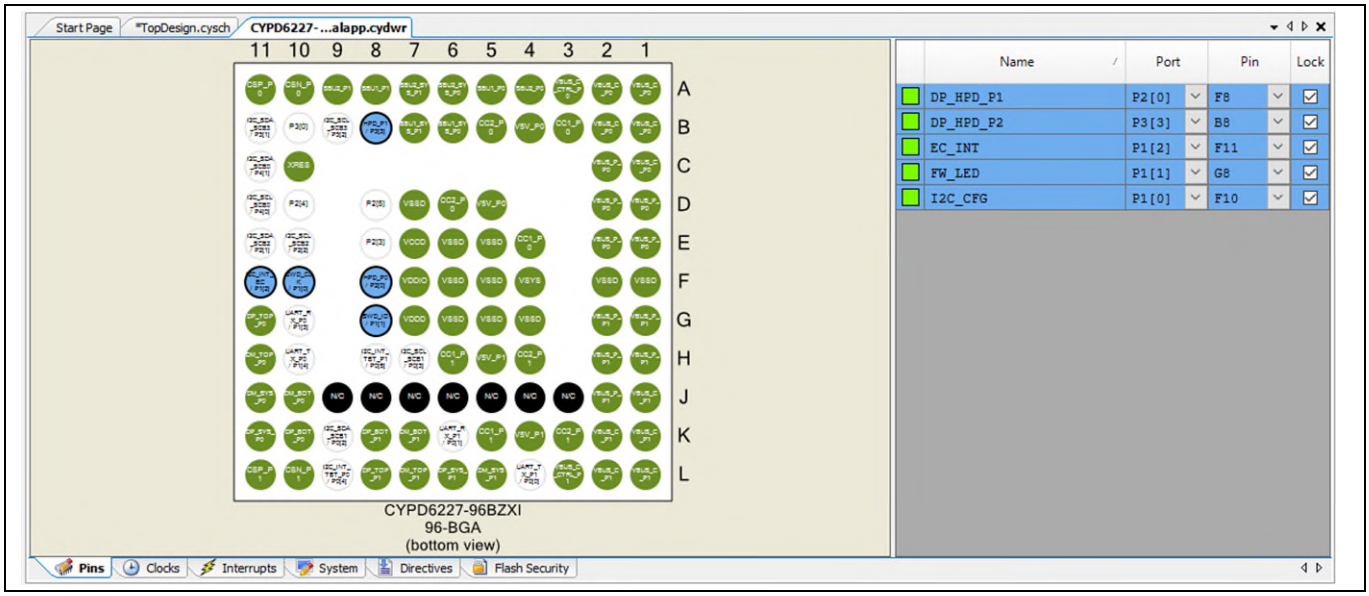


Figure 24 Design Wide Resource (DWR) view of CYPD6227-96BZXI_notebook_dualapp project

As shown in Figure 24, the DWR view has several tabs, which configure aspects such as pin mapping, interrupt mapping, clock selection, flash security, and so on. It is recommended that you restrict any changes to the DWR to the pin mapping view. Do not change the clock, interrupt, system, or flash configurations. Even in the pin mapping editor, the changes should be subject to the constraints outlined in Table 11.

4.7.2 Compile time options

The CYPD6227-96BZXI_notebook_tbt application supports a set of features that can be enabled/disabled using compile time options. These compile time options are set in the config.h header file that you can find under the solution folder, and are summarized in Table 12.

Note: While features can be enabled/disabled as desired, changes to the mode selected for various protection schemes are not recommended.

Table 12 Compile time options in CYPD6227-96BZXI_notebook_dualapp application

Pre-processor switch	Description	Values
SYS_DEEPSLEEP_ENABLE	Enable/disable putting the CCG6F device into a low power mode when idle.	Can be set to 0 to save flash space where required.
VBUS_OVP_ENABLE	Enable/disable detection and handling of VBus overvoltage faults.	Recommend setting this to 1 in all cases.
VBUS_OVP_MODE	Select mode of OVP detection and handling.	0 ➔ OVP detection using the ADC comparator. 1 ➔ OVP detection by OV comparator and handling by firmware. 2 ➔ OVP detection by OV comparator and automated hardware based handling. Should be set to 2 for proper operation.
VBUS_OCP_ENABLE	Enable/disable detection and handling of VBus overcurrent faults.	1 ➔ Type-C VBUS OCP enable. 0 ➔ Type-C VBUS OCP disable.
VBUS_OCP_MODE	Select mode of OVP detection and handling.	0 ➔ Using external OCP hardware 1 ➔ Internal OCP with neither software debounce nor automatic FET control. 2 ➔ Internal OCP with automatic FET control by hardware when an OCP event is detected. 3 ➔ Internal OCP with software debounce using delay in milliseconds from the configuration table. Should be set to 3 for proper operation.
VCONN_OCP_ENABLE	Enable/disable detection and handling of VConn overcurrent faults.	1 ➔ Type-C VConn OCP enable. 0 ➔ Type-C VConn OCP disable.

Pre-processor switch	Description	Values
VBUS_SCP_ENABLE	Enable/disable detection and handling of short-circuit faults.	1 ➔ VBUS SCP feature enable. 0 ➔ VBUS SCP feature disable.
VBUS_RCP_ENABLE	Enable/disable detection and handling of reverse current faults.	1 ➔ VBUS RCP feature enable. 0 ➔ VBUS RCP feature disable.
OTP_ENABLE	Enable/disable the detection and handling of the over-temperature condition.	1 ➔ Enable OTP feature. 0 ➔ Disable OTP feature.
DFP_ALT_MODE_SUPP	Enable/disable the alternate mode discovery and handling state machine when the port managed by CCG is a DFP.	1 ➔ Enable alternate modes when CCG6DF is DFP. 0 ➔ Disable alternate modes when CCG6DF is DFP.
DP_DFP_SUPP	Enable/disable DisplayPort source (DFP_U/DFP_D) functionality.	1 ➔ Enable DP source functionality. 0 ➔ Disable DP source functionality.
TBT_DFP_SUPP	Enable/disable Thunderbolt alternate mode operation when CCG is a DFP.	1 ➔ Enable TBT alternate mode support in DFP role. 0 ➔ Disable TBT alternate mode support in DFP role.
UFP_ALT_MODE_SUPP	Enable/disable the alternate mode discovery and handling state machine when the port managed by CCG5 is a UFP.	1 ➔ Enable alternate modes when CCG5 is UFP. 0 ➔ Disable alternate modes when CCG5 is UFP.
TBT_UFP_SUPP	Enable/disable Thunderbolt alternate mode operation when CCG is a UFP.	1 ➔ Enable TBT alternate mode in UFP role. 0 ➔ Disable TBT alternate mode in UFP role.
RESET_ON_ERROR_ ENABLE	Selects whether to enable / disable CCG device reset on error (watchdog expiry or hard fault)	1 ➔ Enable reset option 0 ➔ Disable reset option
CCGX_POWER_ROLE_SINK_ONLY	Enable/disable the DRP	1 ➔ Enable PD Sink only 0 ➔ PD Port is DRP Used in backup application.
APP_PD_REV3_ENABLE	Enable/disable PD 3.0 support.	1 ➔ Enable PD 3.0 0 ➔ Disable PD 3.0 Used in backup application.

4.7.3 On-chip resource usage

The CYPD6227-96BZXI_notebook_dualapp reference application uses the following on-chip resources for various functionalities.

1. Watchdog: The Watchdog is used as a generic timer for task scheduling and as a mechanism to trigger device reset in case of firmware lock-up.
2. SCB resources
   a) SCB0 is used for HPI slave (I2C) implementation.
   b) SCB2 is used in I2C master mode to configure the buck-boost voltage source and Type-C Signal MUX.
3. TCPWM resources are not used and are available.
4. USB PD resources
   a) ADC block

      The single ADC available is used for VBus measurement so as to track voltage changes and detect disconnection of a power source. The AMUXB input of the ADC is connected internally to a divided version of VBus for measurement. AMUXA input can be used for other purposes on a time-shared basis.

      ADC block along with internal BJT based temperature sensing is used to detect the OTP condition.

   b) Comparators

      The OV comparator is used to detect overvoltage condition by comparing a divided version of VBus against a programmable threshold.

      The UV comparator is unused and can be enabled as required.

      The VSYS_DET comparator is used to detect changes to the VSYS supply.

   c) Signal MUX and switch

      The SBU Switch is used to connect the Type-C side SBU1 and SBU2 pins to the DisplayPort controller.

      The DPDM MUX is used to connect the DP_TOP/DM_TOP or DP_BOT/DM_BOT pins to the DP_SYS/DM_SYS pins to enable USB 2.0 data connection.

4.7.4 EZ-PD™ CCG6SF support

A common workspace is used to generate the binaries for CCG6DF as well as CCG6SF devices. This approach is chosen because a sizable portion of the stack and driver source code on these devices has been moved into ROM. The common ROM code requires that shared data structures be defined for a dual-port part, and will automatically adjust if the actual part in use supports only one PD port.

Python-based post-build scripts are used to generate separate binaries for use on CCG6DF and CCG6SF devices. The following binaries are generated as part of the build process:

1. CYPD6127-48LQXI_notebook_dualapp.hex: CCG6SF based single-port Notebook Port controller
2. CYPD6227-96BZXI_notebook_dualapp.hex: CCG6DF based dual-port Notebook port controller.

While the configuration table settings used in CCG6DF and CCG6SF binaries are different, the main difference is applied automatically by the firmware by identifying the PD part that is in use.

4.8 CYPD6125-40LQXI_notebook application

The CCG6 (CYPD6125-40LQXI_notebook) application implements a Type-C PD port controller for desktop and notebook platforms. Table 13 summarizes the features supported by the primary and the backup versions of this firmware solution.

Table 13 CYPD6125 (CCG6) notebook firmware features

Feature	Support in primary firmware	Support in backup firmware
DRP with Type-C 2.0 spec support	Yes	Yes
Try.SRC and Try.SNK support	Yes	No
Type-C Sink only operation	No	Yes
USB PD Revision 2.0 support	Yes	Yes
USB PD Revision 3.0 support with Fast-Role Swap Receive	Yes	No
Cable discovery support	Yes	No
DFP Data Role Preference	No	No
DP alternate mode as DFP (DFP_U/DFP_D)	Yes	No
Support for other alternate modes as DFP	Yes	No
Support for alternate modes as UFP	No	No
HPI commands for firmware update	Yes	Yes
HPI event notifications (connections, faults, etc.)	Yes	Yes
HPI commands for PD port management	Yes	No
HPI user extension support	Yes	Yes
Deep sleep for power saving	Yes	No
Software Watchdog and Stack Monitoring	Yes	No
VBus OVP support (source/sink)	Yes	Yes
VBus OCP support (source only)	Yes	No
VConn OCP support	Yes	No
VBus SCP support	Yes	No
VBus RCP support	Yes	No
BC 1.2 support (CDP, DCP) as source	Yes	No
Apple charger support as source	Yes	No
UCSI 1.1 support	Yes	No

The following sub-sections describe the project structure and implementation in more detail. The primary firmware implementation is described in detail, as the backup firmware is only a sub-set of the primary firmware.

4.8.1 PSoC™ Creator schematic

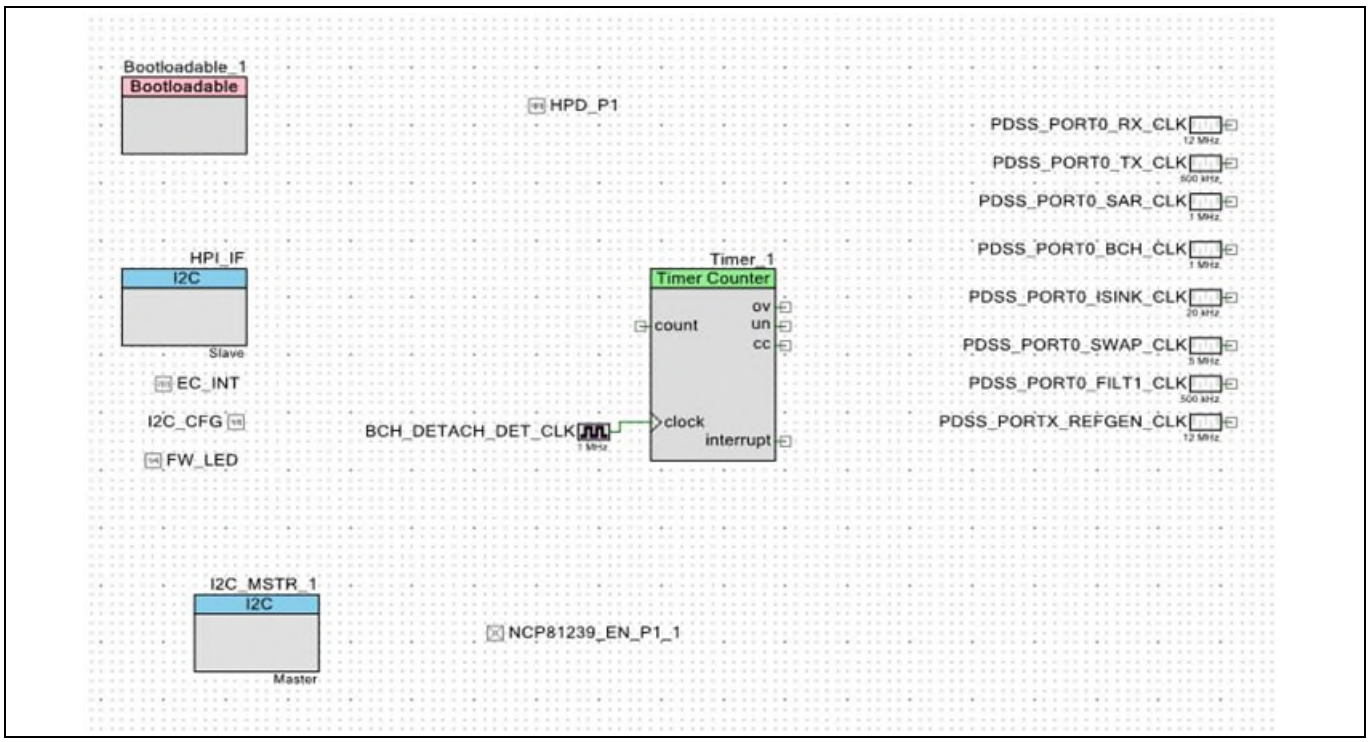


Figure 25 PSoC™ Creator Schematic for CYPD6125 notebook project

Open TopDesign.cysch file in the PSoC™ Creator project. Schematic contains hardware resources used by power adapters such as clocks, voltage selection IOs, etc. The schematic elements are split across multiple sheets, and Figure 25 shows all the active elements together.

The selection of some of these elements is fixed due to the capabilities of the CCG6 device and the bootloader design. Table 14 points out the changes allowed in the schematic design.

Note: There is a similar file in the noboot.cydsn project folder as well. If debugging is being used, the schematic-dependent changes should be replicated there as well.

Table 14 Schematic elements in CCG6 single-port notebook design

Schematic element	Description	Changes allowed
Bootloadable_1	This is a software block which interacts with the boot-loader on the CCG6 device.	Change not recommended. The bootloader binary used can be updated if a custom bootloader is required.
PDSS_PORT0_RX_CLK	This is an internal clock that is used for the RX portion of the USB PD block.	No changes are allowed.
PDSS_PORT0_TX_CLK	This is an internal clock that is used for the TX portion of the USB PD block.	No changes are allowed.

Schematic element	Description	Changes allowed
PDSS_PORT0_SAR_CLK	This is an internal clock that is used for the analog portion of the USB PD block.	No changes are allowed.
PDSS_PORT0_BCH_CLK	This is an internal clock that is used for the Battery Charging.	No changes are allowed.
PDSS_PORT0_ISINK_CLK	This is an internal clock that is used for the soft start of the PFET gate driver of the USB PD block.	No changes are allowed.
PDSS_PORT0_SWAP_CLK	This is an internal clock that is used by FR-SWAP transmit/receive part of the USB PD block.	No changes are allowed.
PDSS_PORT0_FILT1_CLK	This is an internal clock that is used for the filters used for debouncing various comparator outputs in the USB PD block.	No changes are allowed.
PDSS_PORTX_REFGEN_CLK	This is an internal clock that is used for the reference generator block of the USB PD block.	No changes are allowed.
BCH_DETACH_DET_CLK	Clock used for the BC detach detection.	No changes are allowed.
HPI_IF	This is an I2C slave block through which the CCG6 communicates with the Embedded Controller in the Notebook design.	No changes are allowed as the HPI_IF is also used by the boot-loader which is fixed.
EC_INT	Output interrupt signal from CCG6 to the Embedded Controller.	Can be changed only in cases where the EC does not require an interrupt and will poll CCG6 for interrupt notifications.
I2C_CFG	Control signal used to define the I2C slave address used in the HPI interface.	No changes are allowed as the HPI_IF is also used by the boot-loader which is fixed.
FW_LED	GPIO used to toggle an LED periodically to indicate firmware operation.	Can be changed or removed. The APP_FW_LED_ENABLE definition should be set to 0 if this function is being removed.
I2C_MSTR_1	I2C master block used by the CCG6 to control the buck-boost regulator and data switch devices.	Can be changed/removed as required.
NCP81239_EN_P1_1	Output signal used to enable the onboard buck-boost regulator.	Can be changed/removed as required.

Schematic element	Description	Changes allowed
HPD_P1	DisplayPort HotPlug Detect signal output.	Can be removed if DP source function is not required. Changes are not allowed.
Timer_1	Counter block used for the BC detach detection.	No changes are allowed.

Closely associated with the Schematic is the Design Wide Resources (DWR) view, which maps each schematic element to a pin, clock, or hardware block on the CCG6 device. Open the CYPD6125-40LQXI_notebook.cydwr file to see the DWR settings for the project.

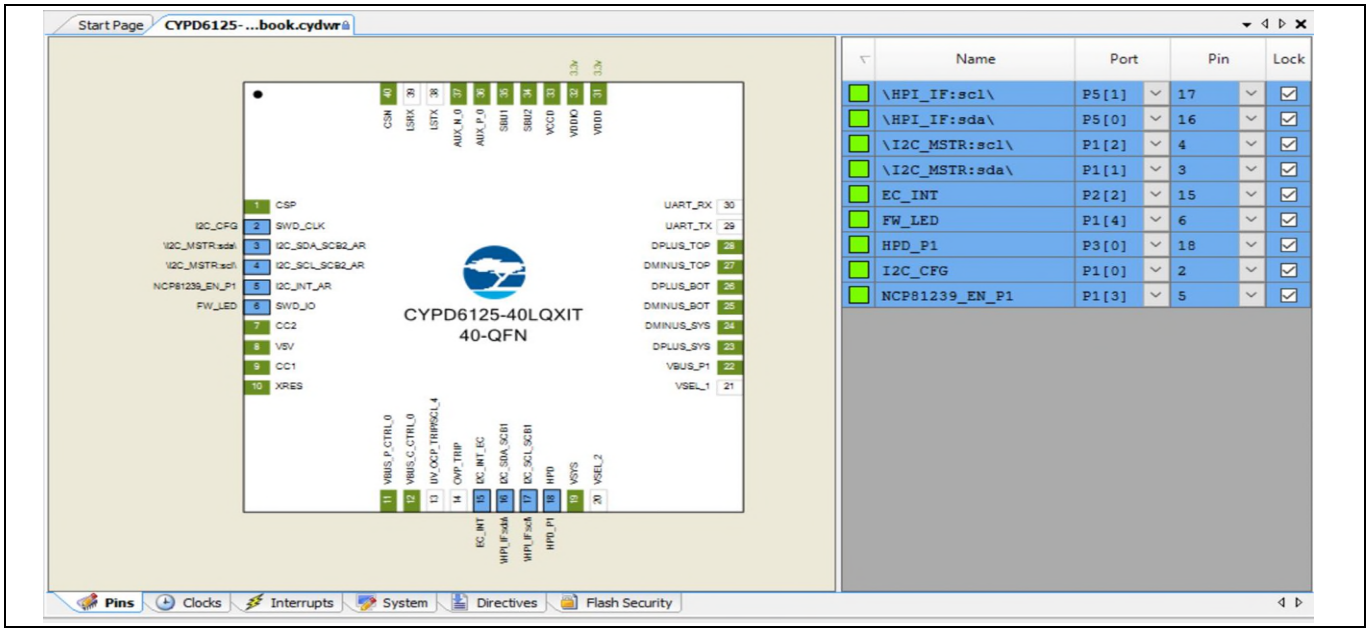


Figure 26 Design Wide Resource (DWR) view of CCG6 single-port notebook design

As shown in Figure 26, the DWR view has several tabs, which configure aspects such as pin mapping, interrupt mapping, clock selection, flash security, and so on. It is recommended that you restrict any changes to the DWR to the pin mapping view. Do not change the clock, interrupt, system, or flash configurations. Even in the pin mapping editor, the changes should be subject to the constraints outlined in Table 14.

4.8.2 Compile time options

The CYPD6125-40LQXI_notebook application supports a set of features that can be enabled/disabled using compile time options. These compile time options are set in the config.h header file that you can find under the solution folder, and are summarized in Table 15.

Note: While features can be enabled/disabled as desired, changes to the mode selected for various protection schemes are not recommended.

Table 15 Compile time options in CCG6 single-port notebook application

Pre-processor switch	Description	Values
BATTERY_CHARGING_ENABLE	Enable/disable BC 1.2 (CDP/DCP) and Apple source support on the USB ports.	Set to 1 to enable BC 1.2 and Apple source support. Set to 0 to disable BC 1.2 and Apple source support.
SYS_DEEPSLEEP_ENABLE	Enable/disable putting the CCG6 device into a low power mode when idle.	Can be set to 0 to save flash space where required.
VBUS_OVP_ENABLE	Enable/disable detection and handling of VBus overvoltage faults.	Recommend setting this to 1 in all cases.
VBUS_OVP_MODE	Select mode of OVP detection and handling.	0 ➔ OVP detection using the ADC comparator. 1 ➔ OVP detection by OV comparator and handling by firmware. 2 ➔ OVP detection by OV comparator and automated hardware based handling.
VBUS_OCP_ENABLE	Enable/disable detection and handling of VBus overcurrent faults.	1 ➔ Type-C VBUS OCP enable. 0 ➔ Type-C VBUS OCP disable.
VBUS_OCP_MODE	Select mode of OVP detection and handling.	0 ➔ Using external OCP hardware 1 ➔ Internal OCP with neither software debounce nor automatic FET control. 2 ➔ Internal OCP with automatic FET control by hardware when an OCP event is detected. 3 ➔ Internal OCP with software debounce using delay in milliseconds from the configuration table.

Pre-processor switch	Description	Values
VCONN_OCP_ENABLE	Enable/disable detection and handling of VConn overcurrent faults.	1 ➔ Type-C VConn OCP enable. 0 ➔ Type-C VConn OCP disable.
VBUS_SCP_ENABLE	Enable/disable detection and handling of short-circuit faults.	1 ➔ VBUS SCP feature enable. 0 ➔ VBUS SCP feature disable.
VBUS_RCP_ENABLE	Enable/disable detection and handling of reverse current faults.	1 ➔ VBUS RCP feature enable. 0 ➔ VBUS RCP feature disable.
DFP_ALT_MODE_SUPP	Enable/disable the alternate mode discovery and handling state machine when the port managed by CCG6 is a DFP.	1 ➔ Enable alternate modes when CCG6 is DFP. 0 ➔ Disable alternate modes when CCG6 is DFP.
DP_DFP_SUPP	Enable/disable DisplayPort source (DFP_U/DFP_D) functionality.	1 ➔ Enable DP source functionality. 0 ➔ Enable DP sink functionality.
TBT_DFP_SUPP	Enable/disable Thunderbolt alternate mode operation when CCG6 is a DFP. This can only be enabled in system designs that include a Thunderbolt controller from Intel.	1 ➔ Enable TBT alternate mode. 0 ➔ Disable TBT alternate mode.
UFP_ALT_MODE_SUPP	Enable/disable the alternate mode discovery and handling state machine when the port managed by CCG6 is a UFP.	1 ➔ Enable alternate modes when CCG6 is UFP. 0 ➔ Disable alternate modes when CCG6 is UFP.
TBT_UFP_SUPP	Enable/disable Thunderbolt alternate mode operation when CCG6 is a UFP. This can only be enabled in system designs that include a Thunderbolt controller from Intel.	1 ➔ Enable TBT alternate mode. 0 ➔ Disable TBT alternate mode.
RESET_ON_ERROR_ ENABLE	Selects whether to enable / disable CCG6 device reset on error (watchdog expiry or hard fault)	1 ➔ Enable reset option 0 ➔ Disable reset option
STACK_USAGE_CHECK_ENABLE	Enable/disable periodic checking of available margin in the runtime stack.	1 ➔ Enable stack margin checks 0 ➔ Disable stack margin checks

4.8.3 On-chip resource usage

The CYPD6125-40LQXI_notebook reference application uses the following on-chip resources for various functionalities.

1. Watchdog: The Watchdog is used as a generic timer for task scheduling and as a mechanism to driver device reset in case of firmware lock-up.
2. SCB resources
   a) SCB0 is used for HPI slave (I2C) implementation.
   b) SCB1 is used as an I2C master to control the NCP81239 buck-boost controller and the PI3DPX1205 Type-C Redriver MUX.
   c) SCB2 and SCB3 are not used and are available.
3. TCPWM resources
   a) TCPWM timer 0 is used to time the output of the DP/DM line comparators to detect USB device disconnections.
   b) TCPWM timer 1 is not used and is available.
4. USB PD resources
   a) ADC block

      The single ADC available is used for VBus measurement so as to track voltage changes and detect disconnection of a power source. The AMUXB input of the ADC is connected internally to a divided version of VBus for measurement. AMUXA input can be used for other purposes on a time-shared basis.

   b) Comparators

      The OV comparator is used to detect overvoltage conditions by comparing a divided version of VBus against a programmable threshold.

      The UV comparator is unused and can be enabled as required.

      The VBUS_MON comparator is unused and can be enabled as required.

      The VSYS_DET comparator is used to detect changes to the VSYS supply.

      The DP_SYS comparator is used to compare the voltage on the DP_SYS pin against a programmable reference.

      The DM_SYS comparator is used to compare the voltage on the DM_SYS pin against a programmable reference.

   c) Signal MUXes

      The SBU MUX is used to connect the SBU1 and SBU2 pins to the AUX_P and AUX_N pins while connected to a Type-C display.

      The DPDM MUX is used to connect the DP_TOP/DM_TOP or DP_BOT/DM_BOT pins to the DP_SYS/DM_SYS pins to enable USB 2.0 data connection.

   d) Charger detect block

      The charger detects block is enabled only when CCG6 is the power source connected to a Type-C sink. The block function can be changed between CDP and Apple brick through runtime HPI commands.

4.9 CYPD5126-40LQXI_notebook application

The CCG5C (CYPD5126-40LQXI_notebook) application implements a Type-C PD port controller for desktop and notebook platforms. Table 16 summarizes the features supported by the primary and backup versions of this firmware solution.

Table 16 CYPD5126 (CCG5C) notebook firmware features

Feature	Support in primary firmware	Support in backup firmware
DRP with Type-C 2.0 spec support	Yes	Yes
Try.SRC and Try.SNK support	Yes	No
Type-C Sink only operation	No	Yes
USB PD Revision 2.0 support	Yes	Yes
USB PD Revision 3.0 support with Fast-Role Swap Receive	Yes	No
Cable discovery support	Yes	No
DFP Data Role Preference	No	No
DP alternate mode as DFP (DFP_U/DFP_D)	Yes	No
Support for other alternate modes as DFP	Yes	No
Support for alternate modes as UFP	No	No
HPI commands for firmware update	Yes	Yes
HPI event notifications (connections, faults, etc.)	Yes	Yes
HPI commands for PD port management	Yes	No
HPI user extension support	Yes	Yes
Deep sleep for power saving	Yes	No
Software Watchdog and Stack Monitoring	Yes	No
VBus OVP support (source/sink)	Yes	Yes
VBus OCP support (source only)	Yes	No
VConn OCP support	Yes	No
VBus SCP support	No	No
VBus RCP support	No	No
BC 1.2 support (CDP, DCP) as source	Yes	No
Apple charger support as source	No	No
UCSI 1.1 support	Yes	No

The following sub-sections describe the project structure and implementation in more detail. The primary firmware implementation is described in detail, as the backup firmware is only a sub-set of the primary firmware.

4.9.1 PSoC™ Creator schematic

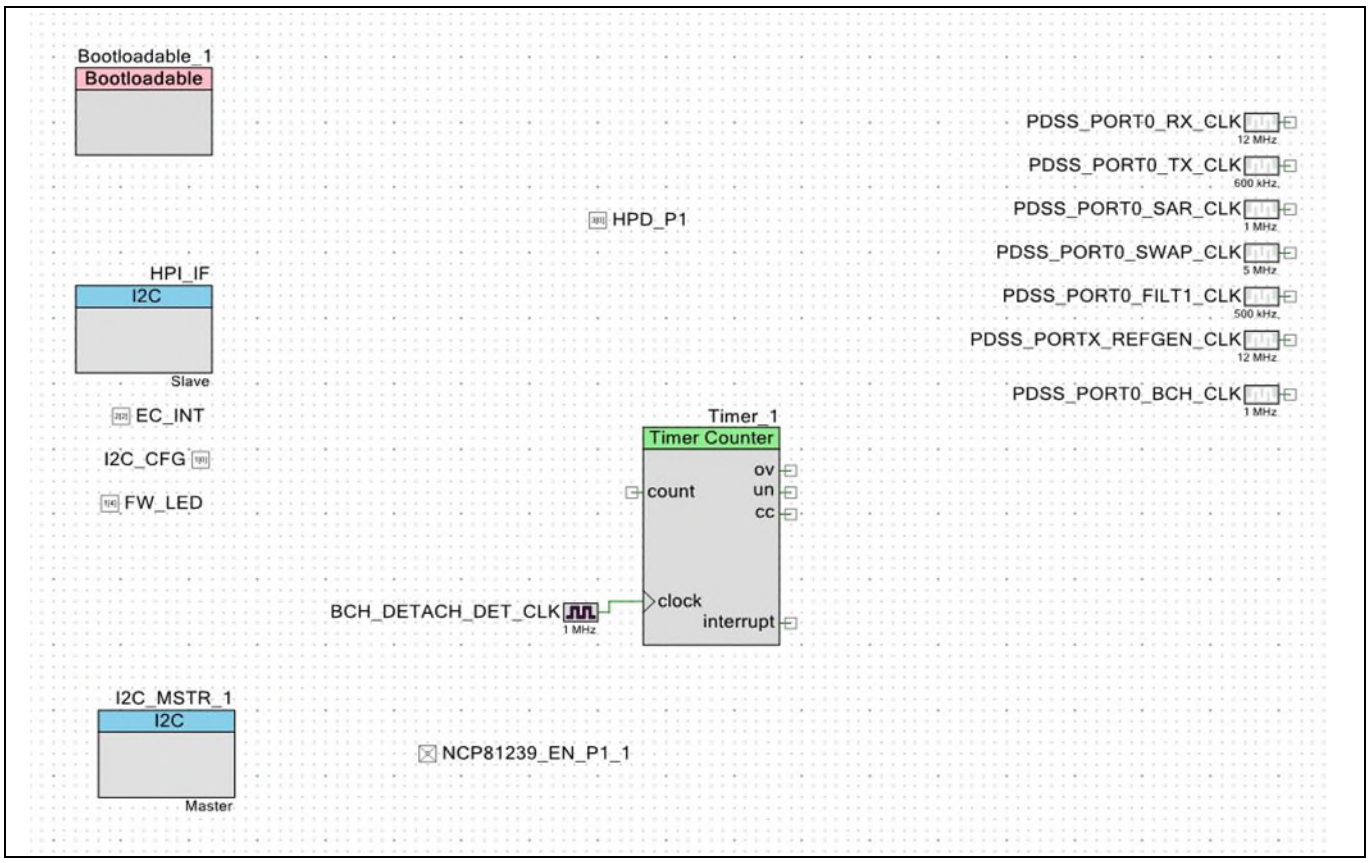


Figure 27 PSoC™ Creator schematic for CYPD5126 notebook project

Open TopDesign.cysch file in the PSoC™ Creator project. Schematic contains hardware resources used by power adapters such as clocks, voltage selection IOs, etc. The schematic elements are split across multiple sheets, and Figure 27 shows all of the active elements together.

The selection of some of these elements is fixed due to the capabilities of the CCG5C device and the bootloader design. Table 17 points out the changes allowed in the schematic design.

Note: There is a similar file in the noboot.cydsn project folder as well. If debugging is being used, the schematic-dependent changes should be replicated there as well.

Table 17 Schematic elements in CCG5C single-port notebook design

Schematic element	Description	Changes allowed
Bootloadable_1	This is a software block which interacts with the boot-loader on the CCG5C device.	Change not recommended. The bootloader binary used can be updated if a custom bootloader is required.
PDSS_PORT0_RX_CLK	This is an internal clock that is used for the RX portion of the USB PD block.	No changes are allowed.

Schematic element	Description	Changes allowed
PDSS_PORT0_TX_CLK	This is an internal clock that is used for the TX portion of the USB PD block.	No changes are allowed.
PDSS_PORT0_SAR_CLK	This is an internal clock that is used for the analog portion of the USB PD block.	No changes are allowed.
PDSS_PORT0_SWAP_CLK	This is an internal clock that is used by FR-SWAP transmit/receive part of the USB PD block.	No changes are allowed.
PDSS_PORT0_FILT1_CLK	This is an internal clock that is used for the filters used for debouncing various comparator outputs in the USB PD block.	No changes are allowed.
PDSS_PORTX_REFGEN_CLK	This is an internal clock that is used for the reference generator block of the USB PD block.	No changes are allowed.
PDSS_PORT0_BCH_CLK	This is an internal clock that is used for the Battery Charging.	No changes are allowed.
BCH_DETACH_DET_CLK	Clock used for the BC detach detection.	No changes are allowed.
HPI_IF	This is an I2C slave block through which the CCG5C communicates with the Embedded Controller in the Notebook design.	No changes are allowed as the HPI_IF is also used by the boot-loader which is fixed.
EC_INT	Output interrupt signal from CCG5C to the Embedded Controller.	Can be changed only in cases where the EC does not require an interrupt and will poll CCG5C for interrupt notifications.
I2C_CFG	Control signal used to define the I2C slave address used in the HPI interface.	No changes are allowed as the HPI_IF is also used by the boot-loader which is fixed.
FW_LED	GPIO used to toggle an LED periodically to indicate firmware operation.	Can be changed or removed. The APP_FW_LED_ENABLE definition should be set to 0 if this function is being removed.
I2C_MSTR_1	I2C master block used by the CCG5C to control the buck-boost regulator and data switch devices.	Can be changed/removed as required.
NCP81239_EN_P1	Output signal used to enable the onboard buck-boost regulator.	Can be changed/removed as required.

Schematic element	Description	Changes allowed
HPD_P1	DisplayPort HotPlug Detect signal output.	Can be removed if DP source function is not required. Changes are not allowed.
Timer_1	Counter block used for the BC detach detection.	No changes are allowed.

Closely associated with the Schematic is the Design Wide Resources (DWR) view, which maps each schematic element to a pin, clock, or hardware block on the CCG5C device. Open the *CYPD5126-40LQXI_notebook.cydwr* file to see the DWR settings for the project.

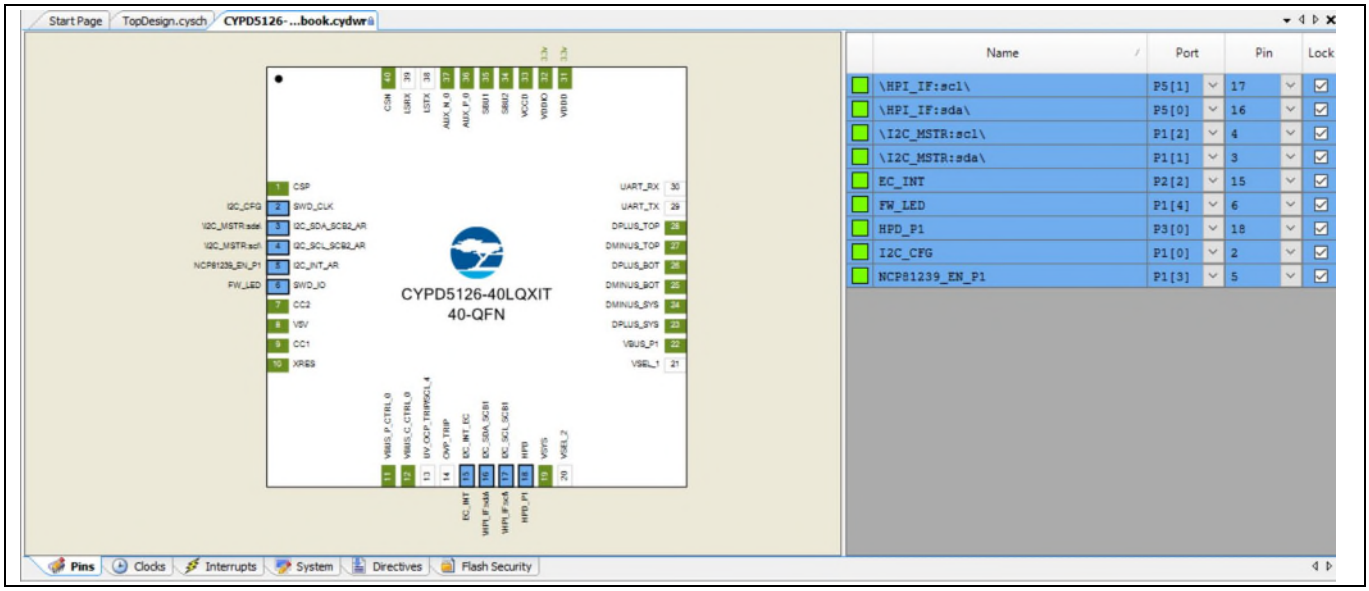


Figure 28 Design Wide Resource (DWR) view of CCG5C single-port notebook application

As shown in Figure 28, the DWR view has several tabs, which configure aspects such as pin mapping, interrupt mapping, clock selection, flash security, and so on. It is recommended that you restrict any changes to the DWR to the pin mapping view. Do not change the clock, interrupt, system, or flash configurations. Even in the pin mapping editor, the changes should be subject to the constraints outlined in Table 17.

4.9.2 Compile time options

The CYPD5126-40LQXI_notebook application supports a set of features that can be enabled/disabled using compile time options. These compile time options are set in the config.h header file that you can find under the solution folder, and are summarized in Table 18.

Note: While features can be enabled/disabled as desired, changes to the mode selected for various protection schemes are not recommended.

Table 18 Compile time options in CCG5C single-port notebook application

Pre-processor switch	Description	Values
BATTERY_CHARGING_ENABLE	Enable/disable BC 1.2 (CDP/DCP) and Apple source support on the USB ports.	Set to 1 to enable BC 1.2 and Apple source support. Set to 0 to disable BC 1.2 and Apple source support.
CCG_BC_12_IN_PD_ENABLE	Enable/disable CDP operation after USB PD contract is in place.	Set to 1 to enable support for CDP negotiation even after PD contract is in place. Set to 0 (default) to disable support for CDP once PD contract is in place.
CCG5_CDP_WAIT_DURATION	Specify a timeout (in seconds) period from Type-C attach at which point BC 1.2 support will be disabled. This can be set to a non-zero value for power savings in cases where a Type-C to Type-A cable is left connected to the port without a device attached.	Timeout period in seconds. 0 (default) means no timeout.
SYS_DEEPSLEEP_ENABLE	Enable/disable putting the CCG5C device into a low power mode when idle.	Can be set to 0 to save flash space where required.
VBUS_OVP_ENABLE	Enable/disable detection and handling of VBus overvoltage faults.	Recommend setting this to 1 in all cases.
VBUS_OCP_ENABLE	Enable/disable detection and handling of VBus overcurrent faults.	1 ➔ Type-C VBUS OCP enable. 0 ➔ Type-C VBUS OCP disable.
VCONN_OCP_ENABLE	Enable/disable detection and handling of VConn overcurrent faults.	1 ➔ Type-C VConn OCP enable. 0 ➔ Type-C VConn OCP disable.
CCG_UCSI_ENABLE	Enable/disable the UCSI interface	1 ➔ UCSI interface enable. 0 ➔ UCSI interface disable.

Pre-processor switch	Description	Values
DFP_ALT_MODE_SUPP	Enable/disable the alternate mode discovery and handling state machine when the port managed by CCG5C is a DFP.	1 ➔ Enable alternate modes when CCG5C is DFP. 0 ➔ Disable alternate modes when CCG5C is DFP.
DP_DFP_SUPP	Enable/disable DisplayPort source (DFP_U/DFP_D) functionality.	1 ➔ Enable DP source functionality. 0 ➔ Enable DP sink functionality.
TBT_DFP_SUPP	Enable/disable Thunderbolt alternate mode operation when CCG5C is a DFP. This can only be enabled in system designs that include a Thunderbolt controller from Intel.	1 ➔ Enable TBT alternate mode. 0 ➔ Disable TBT alternate mode.
UFP_ALT_MODE_SUPP	Enable/disable the alternate mode discovery and handling state machine when the port managed by CCG5C is a UFP.	1 ➔ Enable alternate modes when CCG5C is UFP. 0 ➔ Disable alternate modes when CCG5C is UFP.
TBT_UFP_SUPP	Enable/disable Thunderbolt alternate mode operation when CCG5C is a UFP. This can only be enabled in system designs that include a Thunderbolt controller from Intel.	1 ➔ Enable TBT alternate mode. 0 ➔ Disable TBT alternate mode.

4.9.3 On-chip resource usage

The CYPD5126-40LQXI_notebook reference application uses the following on-chip resources for various functionalities.

1. Watchdog: The Watchdog is used as a generic timer for task scheduling and as a mechanism to driver device reset in case of firmware lock-up.
2. SCB resources
   a) SCB0 is used for HPI slave (I2C) implementation.
   b) SCB1 is used as I2C master to control the NCP81239 buck-boost controller and the PI3DPX1205 Type-C Redriver MUX.
   c) SCB2 and SCB3 are not used and are available.
3. TCPWM resources
   a) TCPWM timer 0 is used to time the output of the DP/DM line comparators to detect USB device disconnections.
   b) TCPWM timer 1 is not used and is available.
4. USB PD resources
   a) ADC block

The single ADC available is used for VBus measurement so as to track voltage changes and detect disconnection of a power source. The AMUXB input of the ADC is connected internally to a divided version of VBus for measurement. AMUXA input can be used for other purposes on a time-shared basis.

b) Comparators

The OV comparator is used to detect overvoltage condition by comparing a divided version of VBus against a programmable threshold.

The UV comparator is unused and can be enabled as required.

The VBus_MON comparator is unused and can be enabled as required.

The VSYS_DET comparator is used to detect changes to the VSYS supply.

The DP_SYS comparator is used to compare the voltage on the DP_SYS pin against a programmable reference.

The DM_SYS comparator is used to compare the voltage on the DM_SYS pin against a programmable reference.

c) Signal MUXes

The SBU MUX is used to connect the SBU1 and SBU2 pins to the AUX_P and AUX_N pins while connected to a Type-C display.

The DPDM MUX is used to connect the DP_TOP/DM_TOP or DP_BOT/DM_BOT pins to the DP_SYS/DM_SYS pins to enable USB 2.0 data connection.

d) Charger detect block

The charger detect block is enabled only when CCG5C is the power source connected to a Type-C sink. The block function can be changed between CDP and DCP through runtime HPI commands.

4.10 CYPD5125-40LQXI_notebook application

The CCG5 (CYPD5125-40LQXI_notebook) application implements a Type-C PD port controller for desktop and notebook platforms. Table 19 summarizes the features supported by the primary and backup versions of this firmware solution.

Table 19 CCG5 single-port notebook firmware features

Feature	Support in primary firmware	Support in backup firmware
DRP with Type-C 1.3 spec support	Yes	Yes
Try.SRC and Try.SNK support	Yes	No
Type-C Sink only operation	No	Yes
USB PD Revision 2.0 support	Yes	Yes
USB PD Revision 3.0 support with Fast-Role Swap Receive	Yes	No
Cable discovery support	Yes	No
DFP Data Role Preference	No	No
DP alternate mode as DFP (DFP_U/DFP_D)	Yes	No

Customizing the firmware application

Feature	Support in primary firmware	Support in backup firmware
Support for other alternate modes as DFP	Yes	No
Support for alternate modes as UFP	No	No
HPI commands for firmware update	Yes	Yes
HPI event notifications (connections, faults, etc.)	Yes	Yes
HPI commands for PD port management	Yes	No
HPI user extension support	Yes	Yes
Deep sleep for power saving	Yes	No
Software Watchdog and Stack Monitoring	Yes	No
VBus OVP support (source/sink)	Yes	Yes
VBus OCP support (source only)	Yes	No
VConn OCP support	Yes	No
VBus SCP support	No	No
VBus RCP support	No	No
BC 1.2 support (CDP, DCP) as source	Yes	No
UCSI 1.1 support	Yes	No

The following sub-sections describe the project structure and implementation in more detail. The primary firmware implementation is described in detail, as the backup firmware is only a sub-set of the primary firmware.

4.10.1 PSoC™ Creator schematic

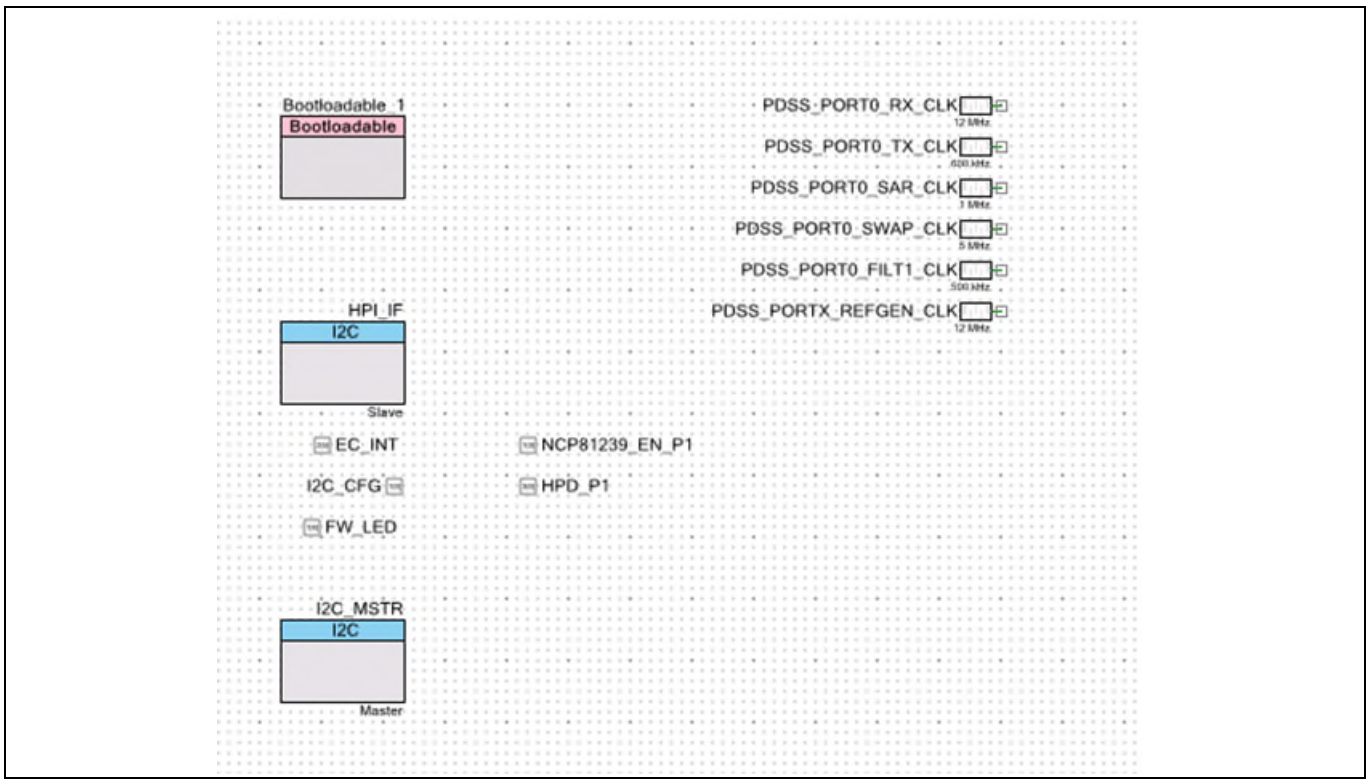


Figure 29 PSoC™ Creator schematic for CYPD5125 notebook project

Open TopDesign.cysch file in the PSoC™ Creator project. Schematic contains hardware resources used by power adapters such as clocks, voltage selection IOs, etc. The schematic elements are split across multiple sheets, and Figure 29 shows all the active elements together.

The selection of some of these elements is fixed due to the capabilities of the CCG5 device and the bootloader design. Table 20 points out the changes allowed in the schematic design.

Note: There is a similar config file in the noboot.cydsn project folder as well. If debugging is being used, the schematic-dependent changes should be replicated there as well.

Table 20 Schematic elements in CCG5 single-port notebook design

Schematic element	Description	Changes allowed
Bootloadable_1	This is a software block which interacts with the boot-loader on the CCG5 device.	Change not recommended. The bootloader binary used can be updated if a custom bootloader is required.
PDSS_PORT0_RX_CLK	This is an internal clock that is used for the RX portion of the USB PD block.	No changes are allowed.
PDSS_PORT0_TX_CLK	This is an internal clock that is used for the TX portion of the USB PD block.	No changes are allowed.

Schematic element	Description	Changes allowed
PDSS_PORT0_SAR_CLK	This is an internal clock that is used for the analog portion of the USB PD block.	No changes are allowed.
PDSS_PORT0_SWAP_CLK	This is an internal clock that is used by FR-SWAP transmit/receive part of the USB PD block.	No changes are allowed.
PDSS_PORT0_FILT1_CLK	This is an internal clock that is used for the filters used for debouncing various comparator outputs in the USB PD block.	No changes are allowed.
PDSS_PORTX_REFGEN_CLK	This is an internal clock that is used for the reference generator block of the USB PD block.	No changes are allowed.
HPI_IF	This is an I^2C slave block through which the CCG5 communicates with the Embedded Controller in the Notebook design.	No changes are allowed as the HPI_IF is also used by the boot-loader which is fixed.
EC_INT	Output interrupt signal from CCG5 to the Embedded Controller.	Can be changed only in cases where the EC does not require an interrupt and will poll CCG5 for interrupt notifications.
I2C_CFG	Control signal used to define the I^2C slave address used in the HPI interface.	No changes are allowed as the HPI_IF is also used by the boot-loader which is fixed.
FW_LED	GPIO used to toggle an LED periodically to indicate firmware operation.	Can be changed or removed. The APP_FW_LED_ENABLE definition should be set to 0 if this function is being removed.
I2C_MSTR	I^2C master block used by the CCG5 to control the buck-boost regulator and data switch devices.	Can be changed/removed as required.
NCP81239_EN_P1	Output signal used to enable the onboard buck-boost regulator.	Can be changed/removed as required.
HPD_P1	DisplayPort HotPlug Detect signal output.	Can be removed if DP source function is not required. Changes are not allowed.

Closely associated with the Schematic is the Design Wide Resources (DWR) view, which maps each schematic element to a pin, clock, or hardware block on the CCG5 device. Open the *CYPD5125-40LQXI_notebook.cydwr* file to see the DWR settings for the project.

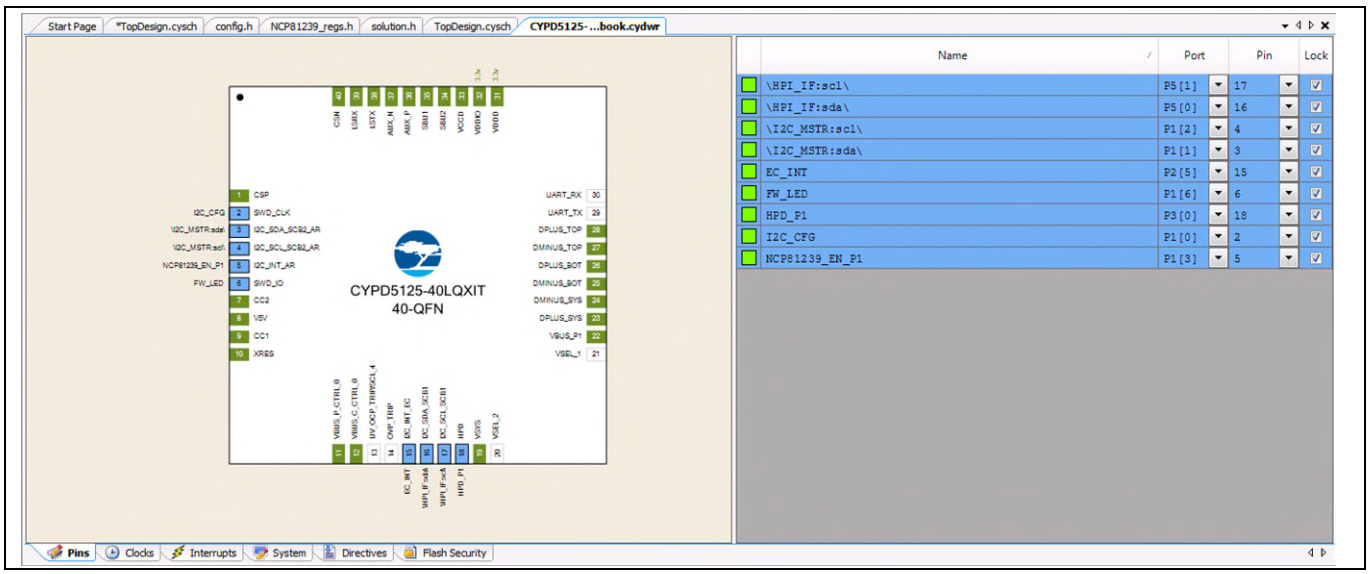


Figure 30 Design Wide Resource (DWR) view of CCG5 single-port notebook application

As shown in Figure 30, the DWR view has several tabs, which configure aspects such as pin mapping, interrupt mapping, clock selection, flash security, and so on. It is recommended that you restrict any changes to the DWR to the pin mapping view. Do not change the clock, interrupt, system, or flash configurations. Even in the pin mapping editor, the changes should be subject to the constraints outlined in Table 20.

4.10.2 Compile time options

The notebook port controller application supports a set of features that can be enabled/disabled using compile time options. These compile time options are set in the config.h header file that you can find under the solution folder, and are summarized in Table 21.

Table 21 Compile time options in CCG5 single-port notebook application

Pre-processor switch	Description	Values
BATTERY_CHARGING_ENABLE	Enable/disable BC 1.2 (CDP/DCP) source support on the USB ports.	Set to 1 to enable BC 1.2 source support. Set to 0 to disable BC 1.2 source support.
CCG_BC_12_IN_PD_ENABLE	Enable/disable CDP operation after USB PD contract is in place.	Set to 1 to enable support for CDP negotiation even after PD contract is in place. Set to 0 (default) to disable support for CDP once PD contract is in place.

Pre-processor switch	Description	Values
CCG5_CDP_WAIT_DURATION	Specify a timeout (in seconds) period from Type-C attach at which point BC 1.2 support will be disabled. This can be set to a non-zero value for power savings in cases where a Type-C to Type-A cable is left connected to the port without a device attached.	Timeout period in seconds. 0 (default) means no timeout.
SYS_DEEPSLEEP_ENABLE	Enable/disable putting the CCG5 device into a low power mode when idle.	Can be set to 0 to save flash space where required.
VBUS_OVP_ENABLE	Enable/disable detection and handling of VBus overvoltage faults.	Recommend setting this to 1 in all cases.
VBUS_OCP_ENABLE	Enable/disable detection and handling of VBus overcurrent faults.	1 ➔ Type-C VBUS OCP enable. 0 ➔ Type-C VBUS OCP disable.
VCONN_OCP_ENABLE	Enable/disable detection and handling of VConn overcurrent faults.	1 ➔ Type-C VConn OCP enable. 0 ➔ Type-C VConn OCP disable.
CCG_UCSI_ENABLE	Enable/disable the UCSI interface	1 ➔ UCSI interface enable. 0 ➔ UCSI interface disable.
DFP_ALT_MODE_SUPP	Enable/disable the alternate mode discovery and handling state machine when the port managed by CCG5 is a DFP.	1 ➔ Enable alternate modes when CCG5 is DFP. 0 ➔ Disable alternate modes when CCG5 is DFP.
DP_DFP_SUPP	Enable/disable DisplayPort source (DFP_U/DFP_D) functionality.	1 ➔ Enable DP source functionality. 0 ➔ Enable DP sink functionality.
TBT_DFP_SUPP	Enable/disable Thunderbolt alternate mode operation when CCG5 is a DFP. This should only be enabled in system designs that include a Thunderbolt controller from Intel.	1 ➔ Enable TBT alternate mode. 0 ➔ Disable TBT alternate mode.
UFP_ALT_MODE_SUPP	Enable/disable the alternate mode discovery and handling state machine when the port managed by CCG5 is a UFP.	1 ➔ Enable alternate modes when CCG5 is UFP. 0 ➔ Disable alternate modes when CCG5 is UFP.

Pre-processor switch	Description	Values
TBT_UFP_SUPP	Enable/disable Thunderbolt alternate mode operation when CCG5 is a UFP. This should only be enabled in system designs that include a Thunderbolt controller from Intel.	1 ➔ Enable TBT alternate mode. 0 ➔ Disable TBT alternate mode.

4.10.3 On-chip resource usage

The CYPD5125-40LQXI_notebook reference application uses the following on-chip resources for various functionalities.

1. Watchdog: The Watchdog is used as a generic timer for task scheduling and as a mechanism to driver device reset in case of firmware lock-up.
2. SCB resources
   a) SCB0 is used for HPI slave (I2C) implementation.
   b) SCB1 is used as an I2C master to control the NCP81239 buck-boost controller and the PI3DPX1205 Type-C Redriver MUX.
   c) SCB2 and SCB3 are not used and are available.
3. TCPWM resources
   a) TCPWM blocks (0 and 1) are unused and are available.
4. USB PD resources
   a) ADC block

      The single ADC available is used for VBus measurement to track voltage changes and detect disconnection of a power source. The AMUXB input of the ADC is connected internally to a divided version of VBus for measurement. AMUXA input can be used for other purposes on a time-shared basis.

   b) Comparators

      The OV comparator is used to detect overvoltage conditions by comparing a divided version of VBus against a programmable threshold.

      The UV comparator is unused and can be enabled as required.

      The VBus_MON comparator is unused and can be enabled as required.

      The VSYS_DET comparator is used to detect changes to the VSYS supply.

   c) Signal MUXes

      The SBU MUX is used to connect the SBU1 and SBU2 pins to the AUX_P and AUX_N pins while connected to a Type-C display.

      The DPDM MUX is used to connect the DP_TOP/DM_TOP or DP_BOT/DM_BOT pins to the DP_SYS/DM_SYS pins to enable USB 2.0 data connection.

d) Charger detect block

The charger detects block is enabled only when CCG5 is the power source connected to a Type-C sink. The block function can be changed between CDP and DCP through runtime HPI commands.

4.11 CYPD5225-96BZXI_notebook application

The dual-port CCG5 notebook application is very similar to the single-port application. The supported features are the same and are summarized in Table 19. The only changes between the projects are additional elements in the schematic design, compile-time configuration changes, and changes to the PD stack and HPI libraries.

4.11.1 PSoC™ Creator schematic

Open TopDesign.cysch file in the PSoC™ Creator project to access the power adapter's schematic. Schematic contains hardware resources used by power adapters such as clocks, voltage selection IOs, etc.

Figure 31 shows the various schematic elements used in the CCG5 dual-port notebook solution. The elements are almost the same as those listed in Table 20 with a few additions. The additions specific to the dual-port project are described in Table 22.

Note: *There is a similar config file in the noboot.cydsn project folder as well. If debugging is being used, the schematic-dependent changes should be replicated there as well.*

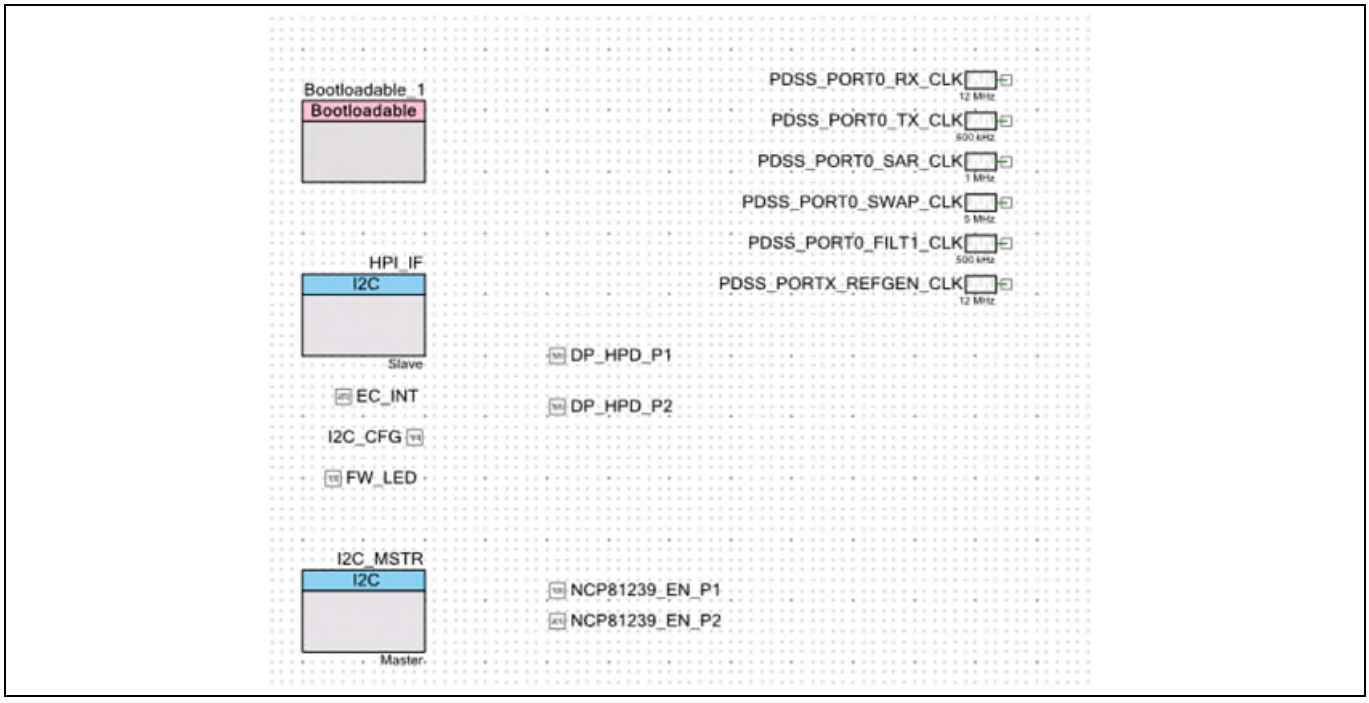


Figure 31 PSoC™ Creator schematic for CCG5 dual-port notebook application

Table 22 Additional schematic elements in CCG5 dual-port notebook solution

Schematic element	Description	Changes allowed
NCP81239_EN_P2	Output signal used to enable the onboard buck-boost regulator for the second PD port.	Can be changed/removed as required.
HPD_P2	DisplayPort HotPlug Detect signal output for the second PD port.	Can be removed if DP source function is not required. Changes are not allowed.

4.11.2 Compile time options

See the section Compile time options for details on how to enable/disable various features in the application firmware.

4.11.3 On-chip resource usage

The CYPD5225-96BZXI_notebook reference application uses the following on-chip resources for various functionalities.

1. Watchdog: The Watchdog is used as a generic timer for task scheduling and as a mechanism to driver device reset in case of firmware lock-up.
2. SCB resources
   a) SCB0 is used for HPI slave (I2C) implementation.
   b) SCB1 is used as an I2C master to control the NCP81239 buck-boost controller and the PI3DPX1205 Type-C Redriver MUX.
   c) SCB2 and SCB3 are not used and are available.
3. TCPWM resources
   a) TCPWM blocks (0 and 1) are unused and are available.
4. USB PD resources
   a) ADC block

      The single ADC available (per PD block) is used for VBus measurement to track voltage changes and detect disconnection of a power source. The AMUXB input of the ADC is connected internally to a divided version of VBus for measurement. AMUXA input can be used for other purposes on a time-shared basis.

   b) Comparators

      The OV comparator is used to detect overvoltage conditions by comparing a divided version of VBus against a programmable threshold.

      The UV comparator is unused and can be enabled as required.

      The VBus_MON comparator is unused and can be enabled as required.

      The VSYS_DET comparator (only available on the first PD port) is used to detect changes to the VSYS supply.

c) Signal MUXes

The SBU MUX is used to connect the SBU1 and SBU2 pins to the AUX_P and AUX_N pins while connected to a Type-C display.

The DPDM MUX is used to connect the DP_TOP/DM_TOP or DP_BOT/DM_BOT pins to the DP_SYS/DM_SYS pins to enable USB 2.0 data connection.

d) Charger detect block

The charger detects block is enabled only when CCG5 is the power source connected to a Type-C sink. The block function can be changed between CDP and DCP through runtime HPI commands.

4.12 CYPD4126-24LQXI_notebook application

This reference project is a single-port notebook PD port controller application using the CYPD4126-24LQXI part from the CCG4 family. The functionality is like that of the CCG5 notebook applications with the following exceptions:

1. GPIOs are used for gate control of the Provider and Consumer FETs. An external FET will be required to generate the actual high-voltage gate control signal.
2. GPIO-based source voltage selection (5 V, 9 V, 15 V, and 20 V) is used instead of I2C-based regulator control.
3. The onboard ADC is used to implement the overvoltage detection and protection feature. In addition, there is no capability to automatically disable power paths by the hardware on fault detection. Firmware intervention is required which means that the latency of fault handling will be higher.
4. Overcurrent detection is not supported by the CCG4 device. So, the application relies on a fault indication from an external load switch to detect overcurrent (overload) faults.
5. There is no support for legacy charging (BC 1.2) protocols.

Table 23 shows the features supported by each of the CCG4 notebook applications.

Table 23 CCG4 notebook application features

Feature	Support in primary firmware	Support in backup firmware
DRP with Type-C 1.3 spec support	Yes	Yes
Try.SRC and Try.SNK support	Yes	No
USB PD Revision 2.0 support	Yes	Yes
USB PD Revision 3.0 support with Fast-Role Swap Receive	Yes	No
Cable discovery support	Yes	No
DP alternate mode as DFP (DFP_U/DFP_D)	Yes	No
Support for other alternate modes as DFP	Yes	No
Support for alternate modes as UFP	No	No
HPI commands for firmware update	Yes	Yes
HPI event notifications (connections, faults, etc.)	Yes	Yes
HPI commands for PD port management	Yes	No

Feature	Support in primary firmware	Support in backup firmware
HPI user extension support	Yes	Yes
Deep sleep for power saving	Yes	No
Software Watchdog and Stack Monitoring	Yes	No
VBus OVP support (source/sink)	Yes	Yes
VBus OCP support (source only)	Yes (based on external load switch)	Yes (based on external load switch)
VConn OCP support	No	No
VBus SCP support	No	No
VBus RCP support	No	No
BC 1.2 support (CDP, DCP) as source	No	No
Apple charger support as source	No	No
UCSI 1.1 support	Yes	No

4.12.1 PSoC™ Creator schematic

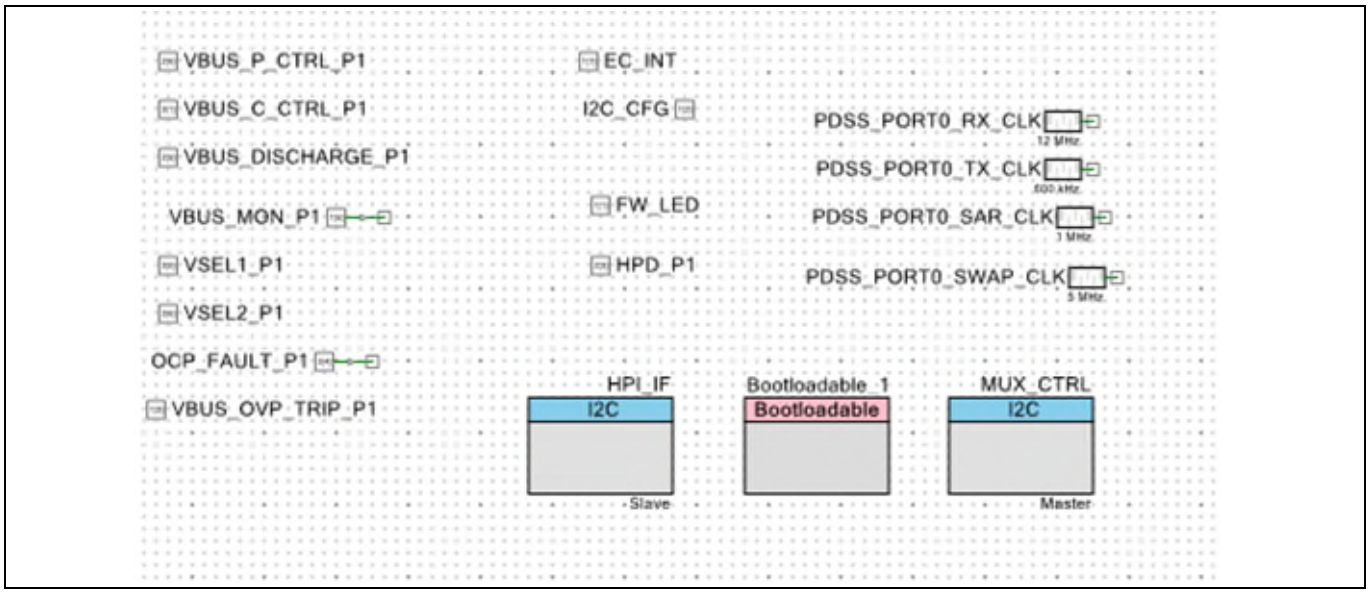


Figure 32 PSoC™ Creator schematic for CYPD4126-24LQXI_notebook project

Most aspects of the hardware design around the CCG4 device are captured in the schematics associated with the PSoC™ Creator firmware project.

The PSoC™ Creator schematic can be found in the *TopDesign.cysch* file, which is part of each PSoC™ Creator project. Double-click on this file to open the schematic editor window (see Figure 32).

The schematic shows how the internal resources of the CCG4 device are used in the design. This includes all the internal clocks used by the design, the various serial interfaces, and all the GPIO pins used to communicate with external elements.

The analog input pins of the CCG4 device are shown with a red wire connected to it on the right side. See the VBUS_MON_P1 signal for example.

Digital input pins are shown with a green wire connected to it on the right side. See the OCP_FAULT_P1 signal for example.

Digital output pins are shown with the corresponding pin mapping annotated on the left side. See the VBUS_P_CTRL_P1 signal for example.

Table 24 shows the various schematic elements used in the CCG4 notebook project. The selection of some of these elements is fixed due to the capabilities of the CCG4 device and the bootloader design. The table also points out the changes allowed in the schematic design.

Table 24 Schematic elements in CCG4 24-pin QFN notebook design

Schematic element	Description	Changes allowed
Bootloadable_1	This is a software block, which interacts with the bootloader on the CCG4 device.	Change not recommended. The bootloader binary used can be updated if a custom bootloader is required.
HPI_IF	This is an I2C slave block through which the CCG4 communicates with the Embedded Controller in the Notebook design.	No changes are allowed as the HPI_IF is also used by the bootloader which is fixed.
MUX_CTRL	This is an I2C master block used by CCG4 to configure the Parade Type-C Interface switch on the CY4541 kit.	This block can be changed / replaced by other mechanisms (such as GPIOs), which can control the interface switch on the target design.
PDSS_PORT0_RX_CLK	This is an internal clock that is used for the RX portion of the USB PD block.	No changes are allowed.
PDSS_PORT0_TX_CLK	This is an internal clock that is used for the TX portion of the USB PD block.	No changes are allowed.
PDSS_PORT0_SAR_CLK	This is an internal clock that is used for the analog portion of the USB PD block.	No changes are allowed.
PDSS_PORT0_SWAP_CLK	This is an internal clock that is used by the Fast Role Swap detect logic to time the incoming Fast Role Swap request.	No changes are allowed.
EC_INT	This is an output pin used to interrupt the Embedded Controller when there is a state change.	No changes are allowed as EC_INT is also used by boot-loader.
I2C_CFG	This is an input pin used to select the I2C slave address used on the HPI interface.	No changes are allowed as EC_INT is also used by boot-loader.

Schematic element	Description	Changes allowed
HPD_P1	This is the Hotplug Detect output pin from CCG4 to the DisplayPort controller on the notebook.	This pin can be removed if DisplayPort is not used. If used, the pin mapping cannot be changed.
FW_LED	This is the firmware activity LED pin.	Actual control is via the GPIO module APIs. See the APP_FW_LED_ENABLE compile-time option for more information.
VSEL1_P1 VSEL2_P1	These are output pins used to select the source voltage to be provided on the Type-C port.	These can be changed based on the voltage selection mechanism in the target hardware.
VBUS_P_CTRL_P1 VBUS_C_CTRL_P1	Output pins used to control the provider and consumer FETs in the design.	These can be changed based on the FET control mechanism in the target hardware.
VBUS_DISCHARGE_P1	Output pin used to control the VBus discharge path in the design.	These can be changed based on the discharge control mechanism in the target hardware.
VBUS_MON_P1	Input pin used to monitor the voltage on VBus.	No changes are allowed as the connectivity to the internal comparators is fixed.
OCP_FAULT_P1	Input pin that notifies CCG4 that an overcurrent condition has been detected.	This can be removed if OCP fault detection circuitry is not available. If used, the name of the pin should not be changed. However, any available GPIO can be used for this purpose.
VBUS_OVP_TRIP_P1	Output pin from CCG4 that is used for a fast turn-off of the VBus supply in case of overvoltage.	This can be removed if OVP trip functionality is not used. If used, the name of the signal and its pin mapping should not be changed.

Closely associated with the schematic is the Design Wide Resources (DWR) view, which maps each schematic element to a pin, clock, or hardware block on the CCG4 device. Open the ***CYPD4126-24LQXI_notebook.cydwr*** file to see the DWR settings for the project.

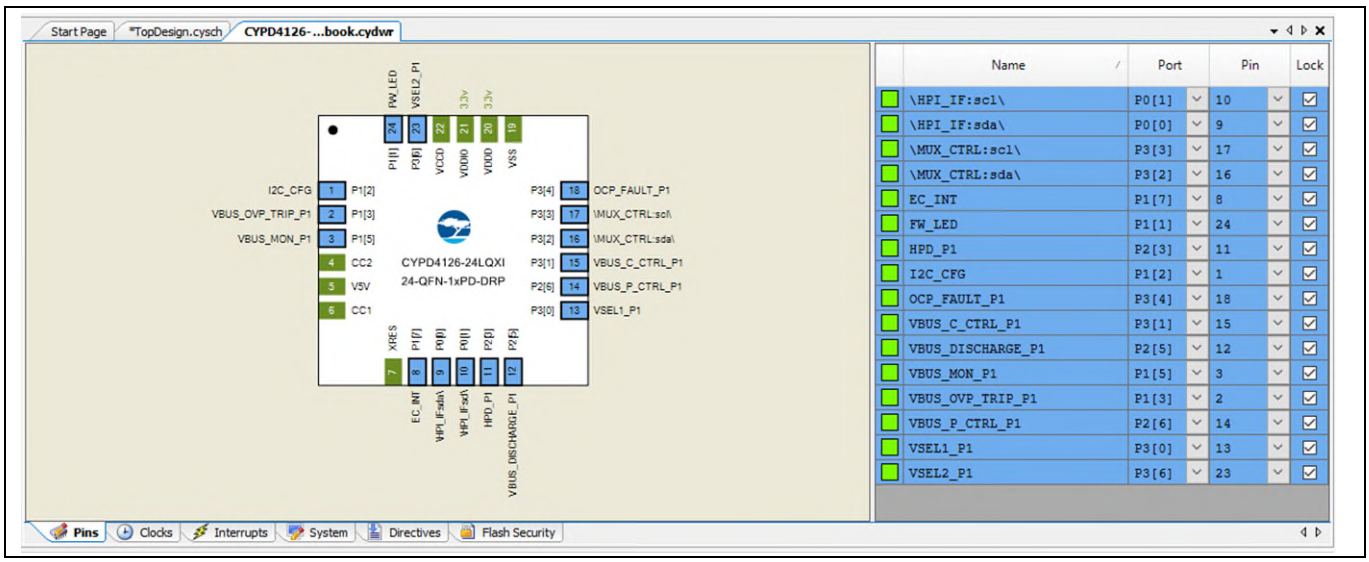


Figure 33 DWR project settings for CYPD4126-24LQXI_notebook

As shown in Figure 33, the DWR view has several tabs, which configure aspects such as pin mapping, interrupt mapping, clock selection, flash security, and so on. It is recommended that you restrict any changes to the DWR to the pin mapping view. Do not change the clock, interrupt, system, or flash configurations. Even in the pin mapping editor, the changes should be subject to the constraints outlined in Table 24.

4.12.2 Compile time options

The CCG4 Notebook port controller application supports a set of features that can be enabled/disabled using compile time options. These compile time options are set in the config.h header file that you can find under the solution folder, and are summarized in Table 25.

Table 25 Compile time options for CCG4 24-QFN notebook application

Option	Description	Values
VBUS_OVP_ENABLE	Enable flag for the internal comparator-based overvoltage protection (OVP) scheme. Even if the OVP feature is enabled using this definition, it can be disabled at run-time using the configuration table.	1 for OVP enable 0 for OVP disable
VBUS_OCP_ENABLE	Enable flag for the external load switch based Over Current Protection (OCP) scheme.	1 for OCP enable 0 for OCP disable
VBUS_OVP_TRIP_ENABLE	Enable flag for a direct supply trip capability from CCG hardware on OVP event. Enabling this requires appropriate circuitry on the target hardware.	1 for OVP-TRIP enable 0 for OVP-TRIP disable

Option	Description	Values
SYS_DEEPSLEEP_ENABLE	Enable flag for the low power module which keeps CCG in Deep Sleep mode at all possible times.	1 for low power enable 0 for low power disable
DFP_ALT_MODE_SUPP	Enable flag for Alternate mode support when CCG is a DFP.	1 for alternate mode enable 0 for alternate mode disable
DP_DFP_SUPP	Enable flag for DisplayPort support when CCG is a DFP.	1 for DisplayPort enable 0 for DisplayPort disable
APP_FW_LED_ENABLE	Enable flag for firmware activity LED indication. When enabled, the user LED blinks at 1-second intervals and the user switch cannot be used. Since the LED uses the SWD_IO GPIO, it is necessary to disable it if debugging via SWD. This LED can be used for development support but is recommended to be left in the OFF state to save power in production designs.	1 for LED enable 0 for LED disable
CCG_UCSI_ENABLE	Enable flag for UCSI features	1 for UCSI enable 0 for UCSI disable

4.12.3 On-chip resource usage

The CYPD4126-24LQXI_notebook reference application uses the following on-chip resources for various functionalities.

1. Watchdog: The Watchdog is used as a generic timer for task scheduling and as a mechanism to driver device reset in case of firmware lock-up.
2. SCB resources
   a) SCB0 is used for HPI slave (I2C) implementation.
   b) SCB3 is used as an I2C master to control the PS8740B Type-C redriving switch.
   c) SCB1 and SCB2 are not used and are available.
3. TCPWM resources
   a) TCPWM blocks (0 and 1) are unused and are available.
4. USB PD resources
   a) ADC block

      ADC0 is used to monitor a divided version of the VBus supply voltage for overvoltage detection. The AMUXA input to the ADC is connected to the VBUS_MON_P1 pin to enable this monitoring. Use of ADC0 for any other measurements is not recommended as this will compromise the latency of overvoltage detection and handling.

ADC1 is used to monitor a divided version of the VBus supply voltage for control of voltage transitions and detection of source disconnection. A separate ADC is used to measure the same VBUS_MON_P1 input so that ADC0 can be left configured for OV detection at all times. The AMUXB input to the ADC can be used for other analog measurements.

b) Gate control

CCG4 does not support any active gate drivers and can only provide GPIO signals which can be used to indirectly enable/disable the Provider and Consumer FETs. On the CYPD4126-24LQXI device, any free GPIOs can be used for gate control.

4.13 CYPD4126-40LQXI_notebook application

This reference project is a single-port notebook PD port controller application using the CYPD4126-40LQXI part from the CCG4 family. The functionality is like that of the CYPD4126-24LQXI notebook application with only one change:

1. The CYPD4126-40LQXI can automatically control the GPIOs used to turn the provider and consumer FETs on/off. For this reason, the VBUS_FET_INTERNAL_CTRL parameter is set to 1 in this project.

For the project schematic, DWR settings and compile time settings; see the CYPD4126-24LQXI_notebook project.

4.13.1 On-chip resource usage

The CYPD4126-40LQXI_notebook reference application uses the following on-chip resources for various functionalities.

1. Watchdog: The Watchdog is used as a generic timer for task scheduling and as a mechanism to driver device reset in case of firmware lock-up.
2. SCB resources
   a) SCB0 is used for HPI slave (I2C) implementation.
   b) SCB3 is used as an I2C master to control the PS8740B Type-C redriving switch.
   c) SCB1 and SCB2 are not used and are available.
3. TCPWM resources
   a) TCPWM blocks (0 and 1) are unused and are available.
4. USB PD resources
   a) ADC block

      ADC0 is used to monitor a divided version of the VBus supply voltage for overvoltage detection. The AMUXA input to the ADC is connected to the VBUS_MON_P1 pin to enable this monitoring. Use of ADC0 for any other measurements is not recommended as this will compromise the latency of overvoltage detection and handling.

      ADC1 is used to monitor a divided version of the VBus supply voltage for control of voltage transitions and detection of source disconnection. A separate ADC is used to measure the same VBUS_MON_P1 input so that ADC0 can be left configured for OV detection at all times. The AMUXB input to the ADC can be used for other analog measurements.

b) Gate control

CCG4 does not support any active gate drivers and can only provide GPIO signals which can be used to indirectly enable/disable the Provider and Consumer FETs. On the CYPD4126-40LQXI device, a pair of optimized GPIOs are provided for gate control. Use of these specific VBUS_C_CTRL_P1 and VBUS_P_CTRL_P1 pins allows hardware control of the FETs for low latency during a Fast Role Swap process.

4.14 CYPD4226-40LQXI_notebook application

This reference project is a dual-port notebook PD port controller application using the CYPD4226-40LQXI part from the CCG4 family. The functionality supported on each port is identical to that supported in the CYPD4126-40LQXI_notebook project.

4.14.1 PSoC™ Creator schematic

Figure 34 shows the schematic associated with the CYPD4226-40LQXI_notebook project. The schematic elements are similar to those in the CYPD4126-24LQXI_notebook project; with the port-specific elements being replicated for the second PD port.

Table 26 shows the additional schematic elements in the CYPD4226-40LQXI_notebook project as compared to the CYPD4126-24LQXI_notebook and CYPD4126-40LQXI_notebook projects.

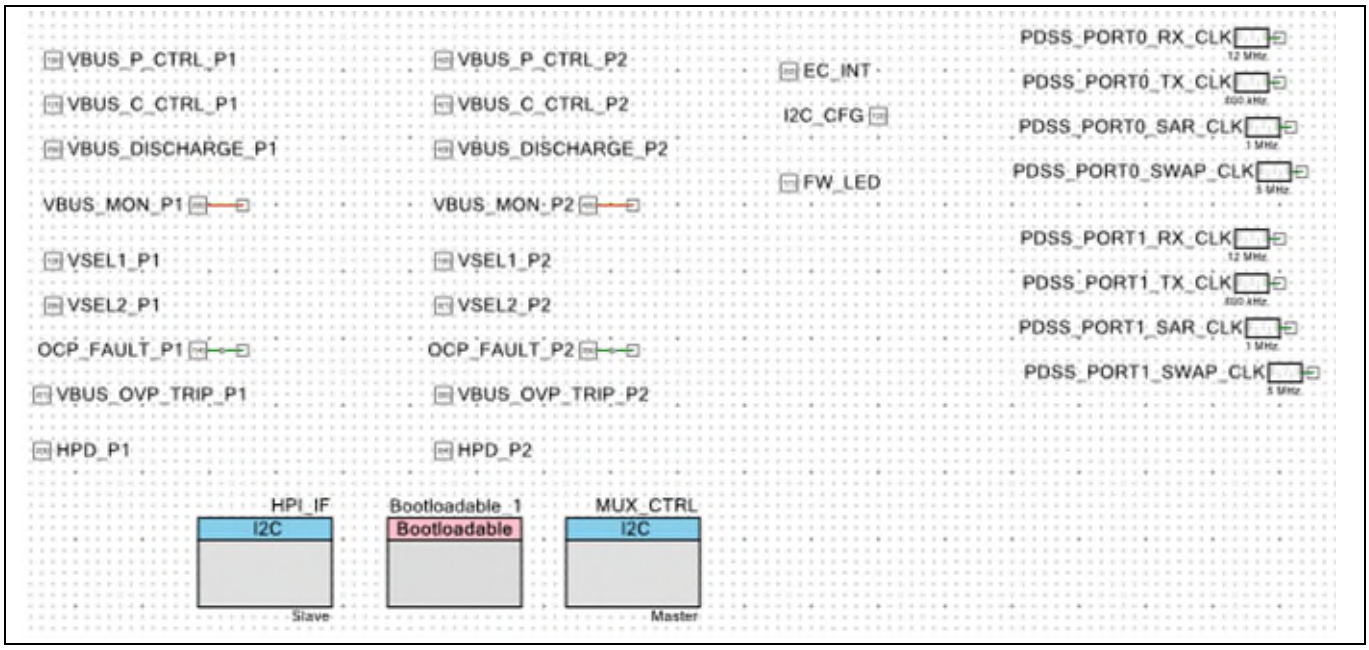


Figure 34 PSoC™ Creator schematic for CYPD4226-24LQXI_notebook project

Table 26 Additional schematic elements in dual-port CCG4 notebook design

Schematic element	Description	Changes allowed
PDSS_PORT1_RX_CLK	This is an internal clock that is used for the RX portion of the USB PD block.	No changes are allowed.

Schematic element	Description	Changes allowed
PDSS_PORT1_TX_CLK	This is an internal clock that is used for the TX portion of the USB PD block.	No changes are allowed.
PDSS_PORT1_SAR_CLK	This is an internal clock that is used for the analog portion of the USB PD block.	No changes are allowed.
PDSS_PORT1_SWAP_CLK	This is an internal clock that is used by the Fast Role Swap detect logic to time the incoming Fast Role Swap request.	No changes are allowed.
HPD_P2	This is the Hotplug Detect output pin from CCG4 to the DisplayPort controller on the notebook.	This pin can be removed if DisplayPort is not used. If used, the pin mapping cannot be changed.
VSEL1_P2 VSEL2_P2	These are output pins used to select the source voltage to be provided on the Type-C port.	These can be changed based on the voltage selection mechanism in the target hardware.
VBUS_P_CTRL_P2 VBUS_C_CTRL_P2	Output pins used to control the provider and consumer FETs in the design.	These can be changed based on the FET control mechanism in the target hardware.
VBUS_DISCHARGE_P2	Output pin used to control the VBus discharge path in the design.	These can be changed based on the discharge control mechanism in the target hardware.
VBUS_MON_P2	Input pins used to monitor the voltage on VBus.	No changes are allowed as the connectivity to the internal comparators is fixed.
OCP_FAULT_P2	Input pin that notifies CCG4 that an overcurrent condition has been detected.	This can be removed if OCP fault detection circuitry is not available. If used, the names of the pins should not be changed. However, any available GPIO can be used for this purpose.
VBUS_OVP_TRIP_P2	Output pin from CCG4 that are used for a fast turn-off of the VBus supply in case of overvoltage.	This can be removed if OVP trip functionality is not used. If used, the names of the signals and their pin mapping should not be changed.

4.14.2 Compile time options

See the section Compile time options for details on the compile time options available for the project.

4.14.3 On-chip resource usage

The CYPD4226-40LQXI_notebook reference application uses the following on-chip resources for various functionalities.

1. Watchdog: The Watchdog is used as a generic timer for task scheduling and as a mechanism to driver device reset in case of firmware lock-up.
2. SCB resources
   a) SCB0 is used for HPI slave (I2C) implementation.
   b) SCB3 is used as an I2C master to control the PS8740B Type-C redriving switch.
   c) SCB1 and SCB2 are not used and are available.
3. TCPWM resources
   a) TCPWM blocks (0 and 1) are unused and are available.
4. USB PD resources
   a) ADC block

      ADC0 is used to monitor a divided version of the VBus supply voltage for overvoltage detection. The AMUXA input to the ADC is connected to the VBUS_MON_Px pin to enable this monitoring. Use of ADC0 for any other measurements is not recommended as this will compromise the latency of overvoltage detection and handling.

      ADC1 is used to monitor a divided version of the VBus supply voltage for control of voltage transitions and detection of source disconnection. A separate ADC is used to measure the same VBUS_MON_Px input so that ADC0 can be left configured for OV detection at all times. The AMUXB input to the ADC can be used for other analog measurements.

   b) Gate control

      CCG4 does not support any active gate drivers and can only provide GPIO signals which can be used to indirectly enable/disable the Provider and Consumer FETs. On the CYPD4226-40LQXI device, optimized GPIOs are provided for gate control. Use of these specific VBUS_C_CTRL_Px and VBUS_P_CTRL_Px pins allows hardware control of the FETs for low latency during a Fast Role Swap process.

4.15 CYAC1126-40LQXI_notebook_rkl application

This reference project implements a single-port notebook PD port controller using the CYAC1126-40LQXI device from the ACG1F family. Table 27 summarizes the features supported by the ACG1F notebook application.

Table 27 ACG1F notebook application features

Feature	Supported
Dual Role Type-C v1.2 compliant port	Yes
Try.SRC configuration support	Yes
Try.SNK configuration support	Yes
USB PD revision 2.0 support	No
USB PD revision 3.0 support	No
Fast Role Swap Receive support	Yes
DisplayPort source state machine	Yes
Thunderbolt (DFP/UFP) state machine	No

Feature	Supported
Firmware upgrade support through HPI	Yes
PD status and event reporting through HPI	Yes
PD command and VDM tunneling support through HPI	Yes
BC 1.2 (CDP) source support	No
VBus overvoltage protection	Yes
VBus overcurrent protection	Yes
VConn overcurrent protection	No

4.15.1 PSoC™ Creator schematic

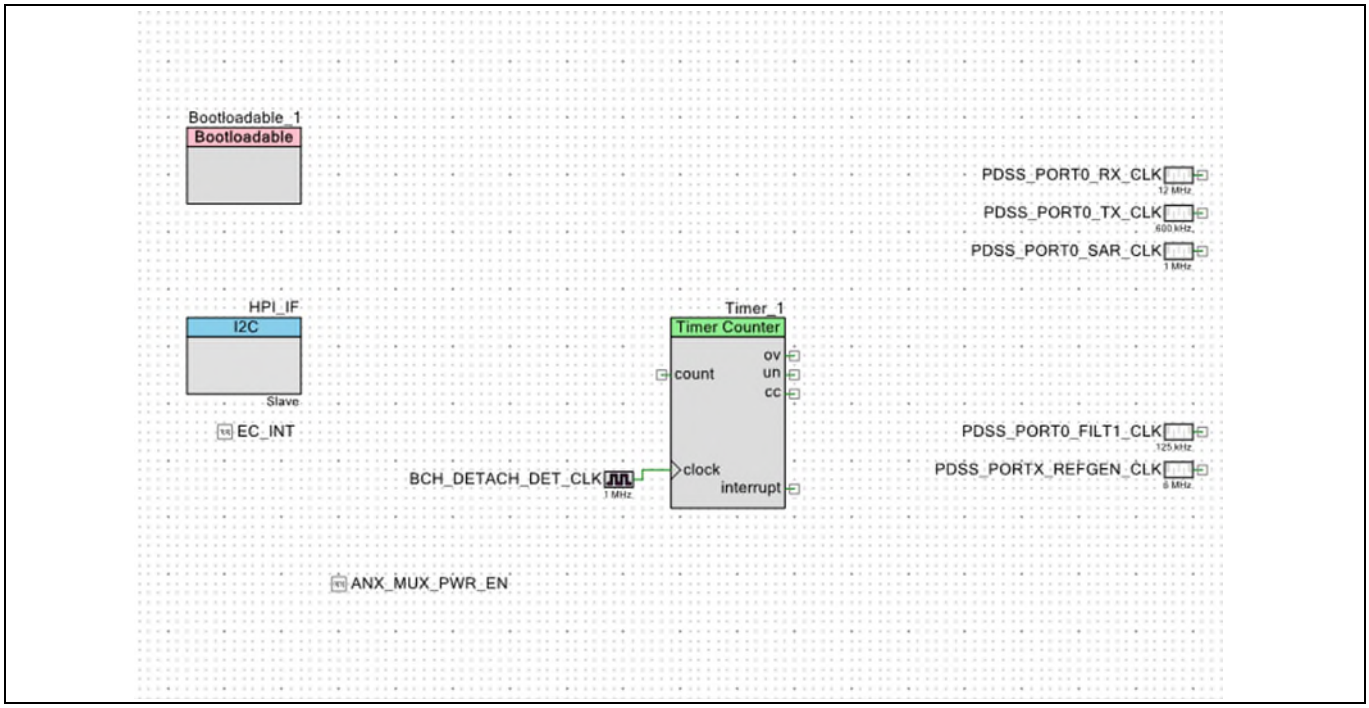


Figure 35 PSoC™ Creator Schematic for ACG1F Notebook

Open TopDesign.cysch file in the PSoC™ Creator project. Schematic contains hardware resources used in the application. The schematic elements are split across multiple sheets, and Figure 35 shows all the active elements together.

In ACG1F projects, the system clock and I2C interface configuration is done automatically by the device start-up code. Hence, these components are not shown in the schematic. In addition to the GPIO interfaces shown above, the CCG6DF USB4 DRD project makes use of three sets of SCB (I2C) interfaces:

1. HPI Interface: I2C slave interface which allows Embedded Controller (EC) in the system to monitor/control the CCG device operation.
2. SoC Interface: I2C slave interface which allows the Intel Platform SoC to identify the PD connection state.

The selection of some of these elements is fixed due to the capabilities of the CCG6xF device. Table 28 points out the changes allowed in the schematic design.

Table 28 Schematic elements in ACG1F - Single-port notebook design

Schematic element	Description	Changes allowed
Bootloadable_1	This is a software block which interacts with the boot-loader on the CCG6DF device.	Change not recommended. If the backup binary has been fixed, it can be used here.
EC_INT_PIN	Output interrupt signal from CCG6DF to the Embedded Controller.	Change not recommended.
I2C_CFG	Control signal used to define the I2C slave address used in the HPI interface.	Change not recommended.

Closely associated with the Schematic is the Design Wide Resources (DWR) view, which maps each schematic element to a pin, clock, or hardware block on the ACG1F device. Open the *CYAC1126-40LQXI_notebook_rkl.cydwr* file to see the DWR settings for the project.

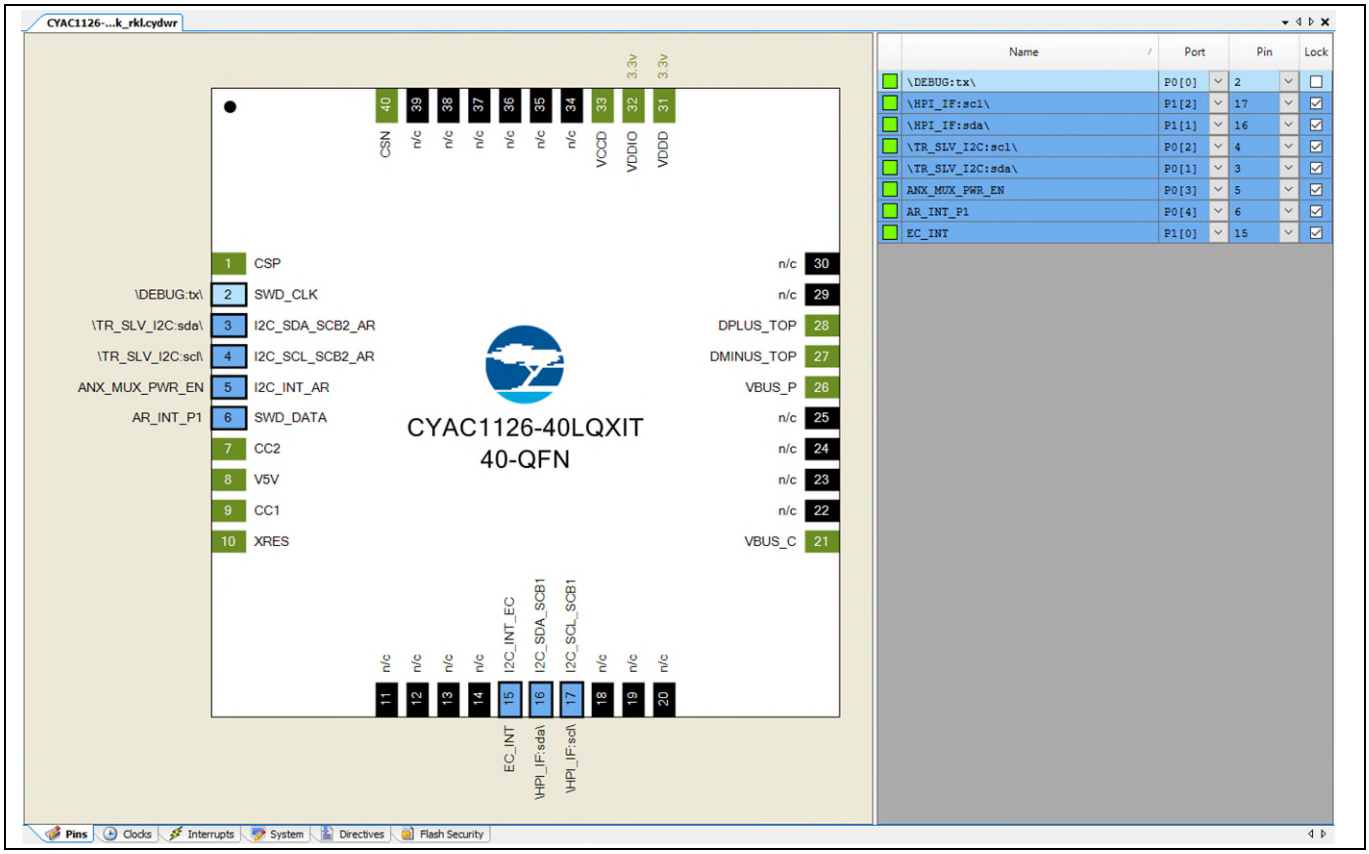


Figure 36 Design Wide Resource (DWR) view of CYPD6227-96BZXI_notebook_tbt project

As shown in Figure 36 the DWR view has several tabs, which configure aspects such as pin mapping, interrupt mapping, clock selection, flash security, and so on. It is recommended that you restrict any changes to the DWR to the pin mapping view. Do not change the clock, interrupt, system, or flash configurations. Even in the pin mapping editor, the changes should be subject to the constraints outlined in Table 28.

4.15.2 Compile time options

The CYAC1126-40LQXI_notebook_rkl application supports a set of features that can be enabled/disabled using compile time options. These compile time options are set in the config.h header file that you can find under the solution folder, and are summarized in Table 29.

Note: While features can be enabled/disabled as desired, changes to the mode selected for various protection schemes are not recommended.

Table 29 Compile time options in CYAC1126-40LQXI_notebook_rkl application

Pre-processor switch	Description	Values
SYS_DEEPSLEEP_ENABLE	Enable/disable putting the CCG6F device into a low power mode when idle.	Can be set to 0 to save flash space where required.
VBUS_OVP_ENABLE	Enable/disable detection and handling of VBus overvoltage faults.	Recommend setting this to 1 in all cases.
VBUS_OVP_MODE	Select mode of OVP detection and handling.	0 ➔ OVP detection using the ADC comparator. 1 ➔ OVP detection by OV comparator and handling by firmware. 2 ➔ OVP detection by OV comparator and automated hardware based handling. Should be set to 2 for proper operation.
VBUS_OCP_ENABLE	Enable/disable detection and handling of VBus overcurrent faults.	1 ➔ Type-C VBUS OCP enable. 0 ➔ Type-C VBUS OCP disable.
VBUS_OCP_MODE	Select mode of OVP detection and handling.	0 ➔ Using external OCP hardware 1 ➔ Internal OCP with neither software debounce nor automatic FET control. 2 ➔ Internal OCP with automatic FET control by hardware when an OCP event is detected. 3 ➔ Internal OCP with software debounce using delay in milliseconds from the configuration table. Should be set to 3 for proper operation.

Customizing the firmware application

Pre-processor switch	Description	Values
VCONN_OCP_ENABLE	Enable/disable detection and handling of VConn overcurrent faults.	1 ➔ Type-C VConn OCP enable. 0 ➔ Type-C VConn OCP disable.
VBUS_SCP_ENABLE	Enable/disable detection and handling of short-circuit faults.	1 ➔ VBUS SCP feature enable. 0 ➔ VBUS SCP feature disable.
VBUS_RCP_ENABLE	Enable/disable detection and handling of reverse current faults.	1 ➔ VBUS RCP feature enable. 0 ➔ VBUS RCP feature disable.
OTP_ENABLE	Enable/disable the detection and handling of the over-temperature condition.	1 ➔ Enable OTP feature. 0 ➔ Disable OTP feature.
RESET_ON_ERROR_ ENABLE	Selects whether to enable / disable CCG device reset on error (watchdog expiry or hard fault)	1 ➔ Enable reset option 0 ➔ Disable reset option
CCGX_POWER_ROLE_SINK_ONLY	Enable/disable the DRP	1 ➔ Enable PD Sink only 0 ➔ PD Port is DRP Used in backup application.
APP_PD_REV3_ENABLE	Enable/disable PD 3.0 support.	1 ➔ Enable PD 3.0 0 ➔ Disable PD 3.0 Used in backup application.
APP_USB4_DATA_ENABLE	Enable/Disable USB 4.0 support	1 ➔ Enable USB4 negotiation 0 ➔ Disable USB4 negotiation
ICL_SLAVE_ENABLE	Enable/Disable Ice Lake PMC Specific support	1 ➔ Enable ICL PMC Specific features 0 ➔ Disable ICL PMC Specific features

4.15.3 On-chip resource usage

The CYAC1126-40LQXI_notebook_rkl reference application uses the following on-chip resources for various functionalities.

1. Watchdog: The Watchdog is used as a generic timer for task scheduling and as a mechanism to trigger device reset in case of firmware lock-up.
2. SCB resources
   a) SCB0 is used for HPI slave (I2C) implementation.
   b) SCB1 is used as I2C master-master-slave to control the SoC & Mux.
3. TCPWM resources are not used and are available.
4. USB PD resources
   a) ADC block

      The single ADC available is used for VBus measurement so as to track voltage changes and detect disconnection of a power source. The AMUXB input of the ADC is connected internally to a divided version of VBus for measurement. AMUXA input can be used for other purposes on a time-shared basis.

      ADC block along with internal BJT based temperature sensing is used to detect the OTP condition.

   b) Comparators

      The OV comparator is used to detect overvoltage condition by comparing a divided version of VBus against a programmable threshold.

      The UV comparator is unused and can be enabled as required.

      The VSYS_DET comparator is used to detect changes to the VSYS supply.

   c) Signal MUX and switch

      The SBU Switch is used to connect the Type-C side SBU1 and SBU2 pins to the Intel platform SoC.

      The DPDM MUX is used to connect the DP_TOP/DM_TOP or DP_BOT/DM_BOT pins to the DP_SYS/DM_SYS pins to enable USB 2.0 data connection.

4.16 CYPD3125-40LQXI_notebook application

This reference project implements a single-port notebook PD port controller using the CYPD3125-40LQXI device from the CCG3 family. Table 30 summarizes the features supported by the CCG3 notebook application.

Table 30 CCG3 notebook application features

Feature	Supported
Dual Role Type-C v1.2 compliant port	Yes
Try.SRC configuration support	Yes
Try.SNK configuration support	Yes
USB PD revision 2.0 support	Yes
USB PD revision 3.0 support	Yes
Fast Role Swap Receive support	Yes
DisplayPort source state machine	Yes
Thunderbolt (DFP/UFP) state machine	No
Firmware upgrade support through HPI	Yes
PD status and event reporting through HPI	Yes
PD command and VDM tunneling support through HPI	Yes
BC 1.2 (CDP) source support	No
VBus overvoltage protection	Yes
VBus overcurrent protection	Yes
VConn overcurrent protection	No

4.16.1 PSoC™ Creator schematic

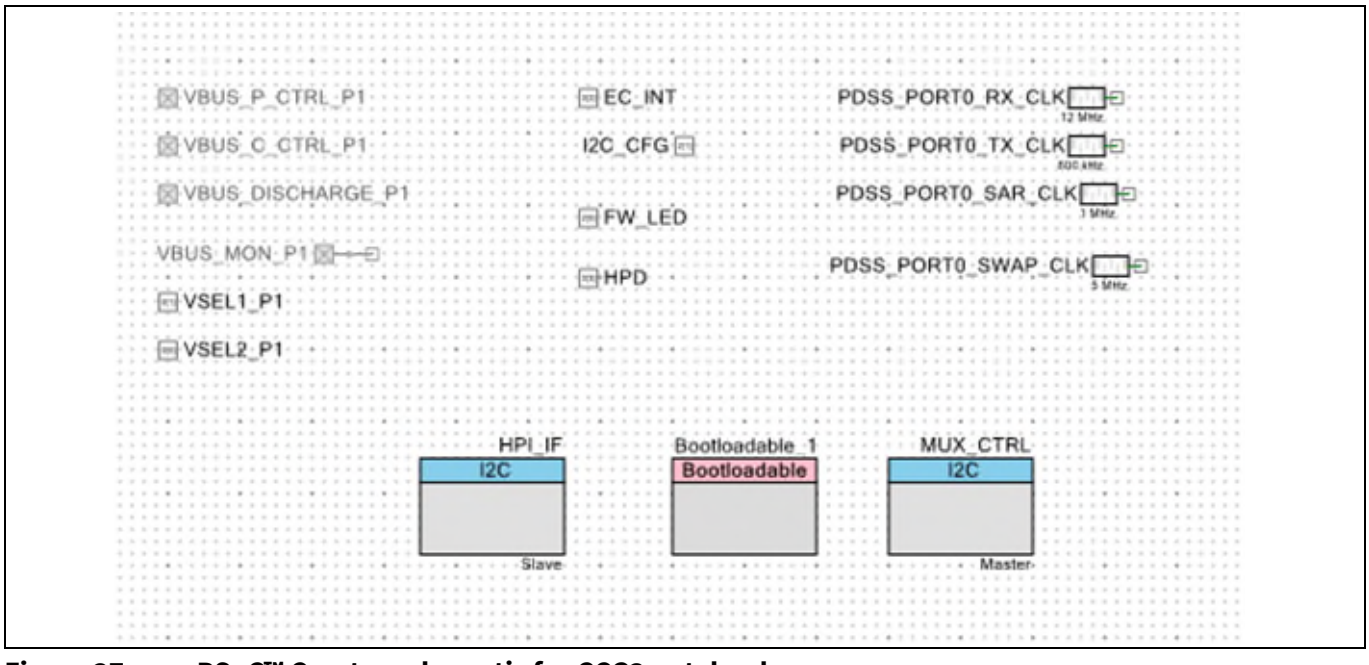


Figure 37 PSoC™ Creator schematic for CCG3 notebook

Customizing the firmware application

Most aspects of the hardware design around the CCG3 device are captured in the schematics associated with the PSoC™ Creator firmware project.

The Creator schematic can be found in the TopDesign.cysch file, which is part of each Creator project. Double-click on this file to open the schematic editor window (see Figure 37).

The schematic shows how the internal resources of the CCG3 device are used in the design. This includes all of the internal clocks used by the design, the various serial interfaces, and all of the GPIO pins used to communicate with external elements.

Table 31 shows the various schematic elements used in the CCG3 notebook project. The selection of some of these elements is fixed due to the capabilities of the CCG3 device and the bootloader design. The table also points out the changes allowed in the schematic design.

Note: *The VBUS_P_CTRL_P1, VBUS_C_CTRL_P1, VBUS_DISCHARGE_PI, and VBUS_MON_P1 elements are shown greyed out because we use the dedicated hardware features of the CCG3 device for gate driver control, VBus discharge control and VBus measurement. Therefore, the corresponding I/Os do not need to be instantiated in the project.*

Table 31 Schematic elements in CCG3 notebook design

Schematic element	Description	Changes allowed
Bootloadable_1	This is a software block which interacts with the boot-loader on the CCG3 device.	No changes should be made to this element.
HPI_IF	This is an I2C slave block through which the CCG3 communicates with the Embedded Controller in the Notebook design.	No changes are allowed as the HPI_IF is also used by the boot-loader which is fixed.
MUX_CTRL	This is an I2C master block used by CCG3 to configure the Parade Type-C Interface switch on the CY4531 kit.	This block can be changed / replaced by other mechanisms (such as GPIOs) which can control the interface switch on the target design.
PDSS_PORT0_RX_CLK	This is an internal clock that is used for the RX portion of the USB PD block.	No changes are allowed.
PDSS_PORT0_TX_CLK	This is an internal clock that is used for the TX portion of the USB PD block.	No changes are allowed.
PDSS_PORT0_SAR_CLK	This is an internal clock that is used for the analog portion of the USB PD block.	No changes are allowed.
PDSS_PORT0_SWAP_CLK	This is an internal clock that is used by the Fast Role Swap detect logic to time the incoming Fast Role Swap request.	No changes are allowed.

Schematic element	Description	Changes allowed
EC_INT	This is an output pin used to interrupt the Embedded Controller when there is a state change.	No changes are allowed as EC_INT is also used by boot-loader.
I2C_CFG	This is an input pin used to select the I2C slave address used on the HPI interface.	No changes are allowed as EC_INT is also used by boot-loader.
HPD	This is the Hotplug Detect output pin from CCG3 to the DisplayPort controller on the notebook.	This pin can be removed if DisplayPort is not used. If used, the pin mapping cannot be changed.
FW_LED	This is the firmware activity LED pin.	Actual control is via the GPIO module APIs. See the APP_FW_LED_ENABLE compile-time option for more information.
VSEL1_P1 VSEL2_P1	These are output pins used to select the source voltage to be provided on the Type-C port.	These can be changed based on the voltage selection mechanism in the target hardware.

4.16.2 Compile time options

Table 32 shows the compile-time selectable features supported by this application.

Table 32 Compile time options in CCG3 notebook application

Pre-processor switch	Description	Values
VBUS_OVP_ENABLE	Enable overvoltage protection handling on VBus. This feature can be turned off using the configuration table, even if it is enabled here.	1 for VBus OVP enable 0 for VBus OVP disable
VBUS_OVP_MODE	Select the OVP handling mechanism.	0 is reserved value 1 for firmware based FET turn-off on OV detection. 2 for hardware based FET turn-off on OV detection.
VBUS_OCP_ENABLE	Enable overcurrent protection on VBus supply when CCG3 is acting as the power source. This feature can be turned off using the configuration table, even if it is enabled here.	1 for VBus OCP enable 0 for VBus OCP disable

Pre-processor switch	Description	Values
VBUS_OCP_MODE	Select handling mechanism for overcurrent faults detected by the firmware.	2 for hardware based FET turn-off on OC detection. 3 for firmware based debounce and FET turn-off on OC detection.
SYS_DEEPSLEEP_ENABLE	Enable flag for the low power module which keeps CCG in deep sleep mode at all possible times.	1 for low power enable 0 for low power disable
DFP_ALT_MODE_SUPP	Enable Alternate Mode handling when CCG is DFP.	1 for alternate mode enable 0 for alternate mode disable
DP_DFP_SUPP	Enable DisplayPort Alternate mode when CCG is DFP. This requires DFP_ALT_MODE_SUPP.	1 for DisplayPort enable 0 for DisplayPort disable
APP_FW_LED_ENABLE	Enable flag for firmware activity LED indication. When enabled, the user LED blinks at 1-second intervals and the user switch cannot be used. Since the LED uses the SWD_IO GPIO, it is necessary to disable it if debugging via SWD. This LED can be used for development support but is recommended to be left in the OFF state to save power in production designs.	1 for LED enable 0 for LED disable

4.16.3 On-chip resource usage

The CYPD3125-40LQXI_notebook reference application uses the following on-chip resources for various functionalities.

1. Watchdog: The Watchdog is used as a generic timer for task scheduling and as a mechanism to driver device reset in case of firmware lock-up.
2. SCB resources
   a) SCB2 is used for HPI slave (I2C) implementation.
   b) SCB3 is used as I2C master to control the PS8740B Type-C Redriver switch.
   c) SCB0 and SCB1 are not used and are available.
3. TCPWM resources
   a) TCPWM timers 0 and 1 are not used and is available.
4. USB PD resources
   a) ADC block

      ADC0 is unused and can be used for analog measurement purposes.

      ADC1 is used to monitor a divided version of the VBus supply voltage for control of voltage transitions and detection of source disconnection. The AMUXA input of the ADC is connected to internal VBus resistor divider to enable this monitoring. The AMUXB input to the ADC can be used for other analog measurements.

   b) UVOV block

      The device has a dedicated UVOV block which is used to detect the overvoltage conditions by monitoring a divided version of the VBus voltage.

   c) Signal MUXes

      The SBU MUX is used to connect the SBU1 and SBU2 pins to the AUX_P and AUX_N pins while connected to a Type-C display.

   d) Charger detect block

      The charger detect block on CCG3 is currently not enabled and can be used for implementation of legacy charging protocols such as BC1.2.

4.17 CYPD3126-42FNXI_notebook application

This reference project is a clone of the CYPD3125-40LQXI_notebook project targeted for the CSP package of the CCG3 device. See the section CYAC1126-40LQXI_notebook_rkl application for details of this project.

5 Firmware architecture

5.1 Firmware blocks

The CCGx firmware architecture allows users to implement a variety of USB PD applications using the CCG devices and a fully tested firmware stack. A block diagram of the CCGx firmware architecture is shown in Figure 38.

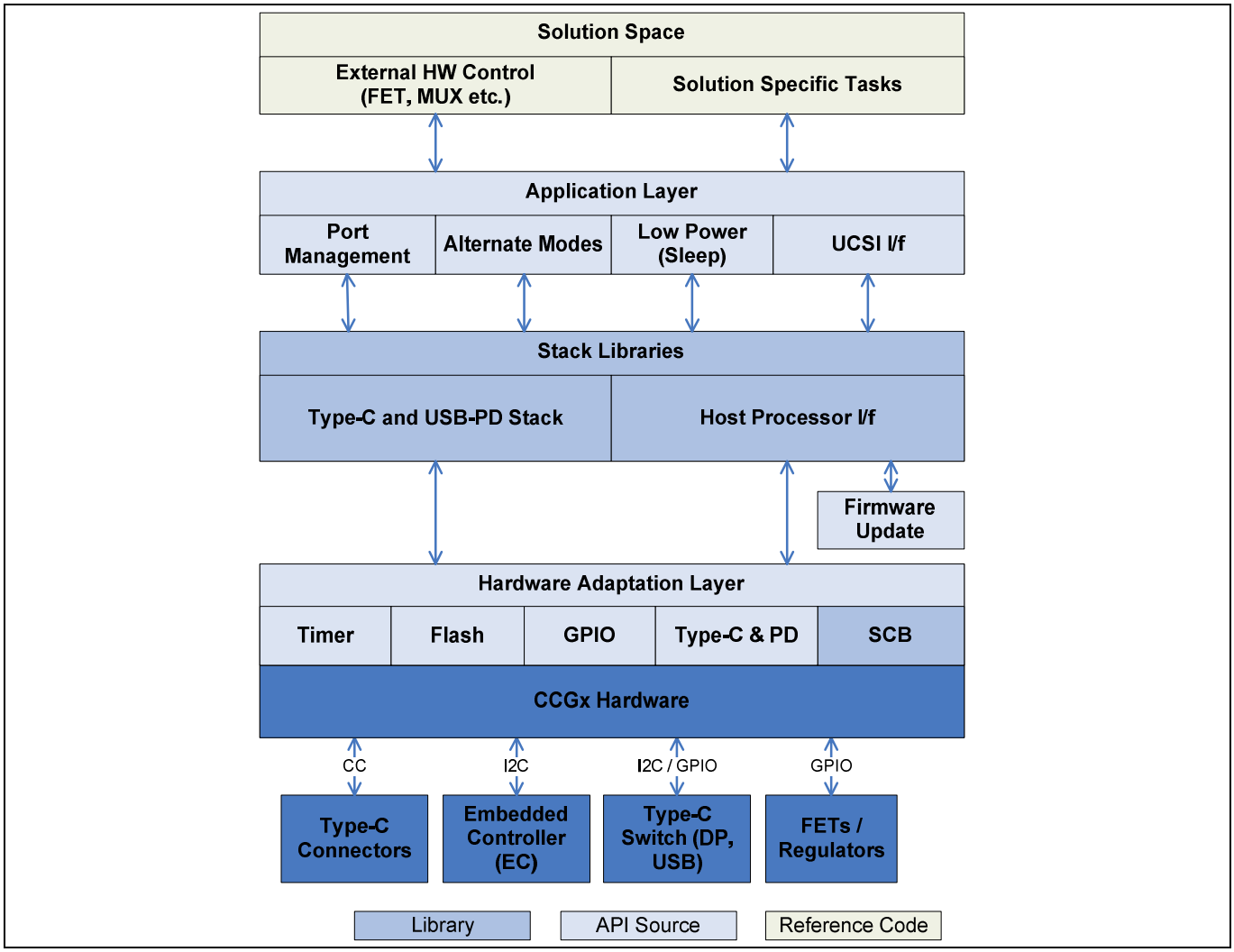


Figure 38 CCGx firmware block diagram

Firmware architecture

The CCGx firmware architecture contains the following components:

- Hardware Abstraction Layer (HAL): This includes the low-level drivers for the various hardware blocks on the CCG device. This includes drivers for the Type-C and USB PD block, Serial Communication Blocks (SCBs), GPIOs, flash module, and timer module.
- USB Type-C and USB PD Protocol Stack: This is the complete USB PD protocol stack that includes the Type-C and USB PD port managers, USB PD protocol layer, the USB PD policy engine, and the device policy manager. The device policy manager is designed to allow all policy decisions to be made at the application level, either on an external Embedded Controller (EC) or in the CCG firmware itself.
- Host Processor Interface (HPI): The Host Processor Interface (HPI) is an I2C-based control interface that allows an Embedded Controller (EC) to monitor and control the USB PD port on the CCG device. The HPI is the means to allow the PC platform to control the PD policy management. This interface is not applicable for Power Adapters and Power Banks.
- Firmware update module: This is a firmware module that allows the device firmware maintained in internal flash to be updated. The functions provided by this module are invoked through the HPI module based on flash read/write commands received from the EC.
- Port Management: This module handles all of the PD port management functions including the algorithm for optimal contract negotiations, source and sink power control, source voltage selection, port role assignment, and swap request handling.
- Alternate Modes: This module implements the alternate mode handling for CCG as a DFP and UFP. A fully tested implementation of DisplayPort alternate mode with CCG as DFP is provided. The module also allows users to implement their own alternate mode support in both DFP and UFP modes.
- Low Power: This module attempts to keep the CCG device in the low-power standby mode as often as possible to minimize power consumption.
- UCSI: The UCSI module implements a set of registers and commands which the EC can use to implement the USB Type-C System Software Interface.
- External Hardware Control: This is a hardware design-dependent module, which controls the external hardware blocks such as FETs, regulators, and Type-C switches.
- Solution specific tasks: This is an application layer module where any custom tasks required by the user solution can be implemented.

5.2 SDK usage model

Users of the CCG solution must follow these steps to use the SDK components:

1. Load the solution workspace using PSoC™ Creator.
2. Edit the project schematics and solution configuration header file if needed.
3. Use the EZ-PD™ Configuration Utility to build the configuration table and copy the generated C source file into the Creator project if necessary. The configuration table can also be updated by editing the config.c file in PSoC™ Creator Source Editor.
4. Build the application projects using the PSoC™ Creator. The firmware binaries will be generated in ELF, HEX, and CYACD formats suitable for SWD programming, MiniProg, and the EZ-PD™ configuration utility.
5. Load the firmware binary onto the target hardware for evaluation and testing.

This usage flow is illustrated in Figure 39. Many of these steps, such as changing the compile time configurations and using the EZ-PD™ Configuration Utility to change the configuration table, are only required if the customer wants to change the way the application works.

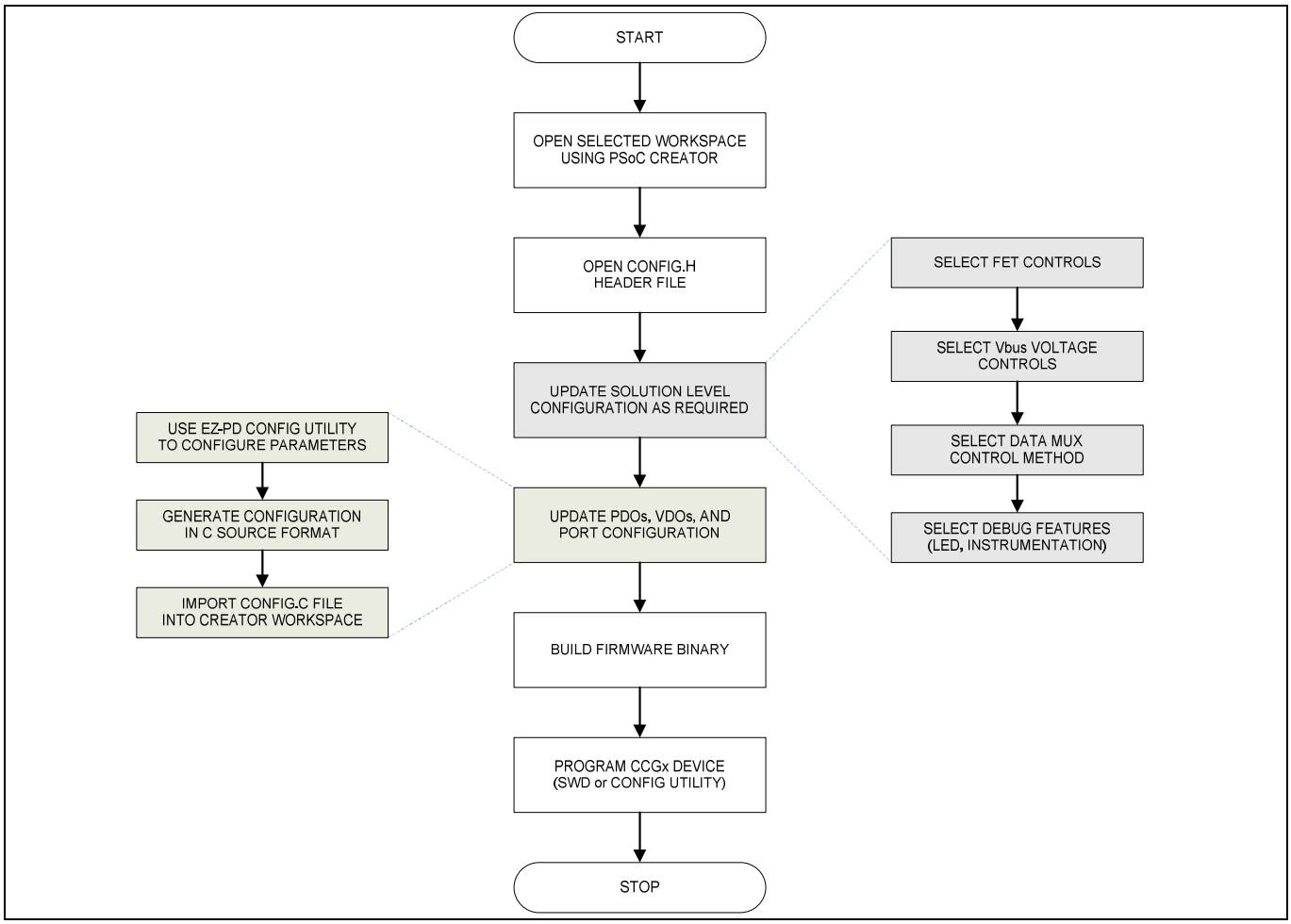


Figure 39 SDK usage flow

5.3 Firmware versioning

Each project has a firmware version (base version) and an application version number.

The base firmware version number shall consist of a major number, minor number, and patch number in addition to an automatically updated build number.

The base firmware version applies to the whole stack and is common for all applications and projects using the stack. The version information can be found in the ***src/system/ccgx_version.h*** header file.

The application version shall be modified for individual customers based on requirements. This shall have a major version, minor version, external circuit specification, and application name. This version information can be updated by users as required, and is located in the ***Firmware/projects/<project_name>/common/app_version.h*** header file.

Note: Ensure that you do not change the application name from the value defined for the CCG application type. The application type information is used by the EZ-PD™ configuration utility to interpret the configuration table content.

The version number information for each firmware shall be stored in an eight-byte data field and shall be retrieved over the firmware upgrade interface. The following table denotes the version structure and format.

Table 33 CCGx firmware version structure

Bit field	Name	Description
[15:0]	Base FW Build number	This field corresponds to base firmware version and shall be automatically incremented during nightly build. This field should not be manually edited. This field is expected to be reset on every SNPP release cycle and not modified throughout the release.
[23:16]	Base FW Patch version number	This field corresponds to base firmware patch version number. This field shall be updated manually by the core PD team for base firmware releases. This field shall be incremented for every intermediate release done to customer or an actual patch release performed for a previous full release.
[27:24]	Base FW Minor version number	This field corresponds to base firmware minor version number. This field shall be updated manually by the core PD team for base firmware releases. This field is generally updated once for every SNPP release cycle at ES100 RC build. The exception is when an intermediate customer release which breaks compatibility.

<table>
<tr><th>Bit field</th><th>Name</th><th>Description</th></tr>
<tr><td>[31:28]</td><td>Base FW Major version number</td><td>This field corresponds to base firmware major version number. This field shall be updated manually by the core PD team for base firmware releases.
The major number is generally updated on a major project level change or when we have cycled through all minor numbers. The number shall be determined at the beginning of every SNPP release cycle.</td></tr>
<tr><td>[47:32]</td><td>Application Name / number</td><td>This field is left for any application / customer-specific changes to be done by applications team.
By default, this field shall be released by the base firmware version team will have the following values:
<table><tr><td>Notebook</td><td>“nb”</td></tr><tr><td>Power Adapter</td><td>“pa”</td></tr><tr><td>Power Bank</td><td>“pb”</td></tr><tr><td>Alternate Mode Adapter (AMA)</td><td>“aa”</td></tr></table>
Note: This information is used by EZ-PD™ Configuration Utility to determine the application type and should not be modified for standard applications.</td></tr>
<tr><td>[55:48]</td><td>External circuit number</td><td>This field is left for any application / customer-specific changes to be done by applications team. By default, this field shall be released by the base firmware team as 0.
The circuit number values from 0x00 to 0x1F are reserved for base firmware team. This is because base firmware team may have to support same application on multiple platforms in the future.</td></tr>
<tr><td>[59:56]</td><td>Application minor version number</td><td>This field is left for any application / customer-specific changes to be done by applications team. By default, this field shall be released by the base firmware team as 0.</td></tr>
<tr><td>[63:60]</td><td>Application major version number</td><td>This field is left for any application / customer-specific changes to be done by applications team. By default, this field shall be released by the base firmware team as 0.</td></tr>
</table>

5.4 Flash memory map

5.4.1 ACG1F flash memory map

ACG1F device has 16 KB of flash and 32 KB of SROM memory. The first 8KB of the SROM memory is the Supervisory ROM used to store the boot and configuration routines. The remaining 24 KB of ROM is the User ROM. The User ROM and the 16 KB Flash is used for storing the application code. Firmware upgrade shall happen from the I2C boot-loader.

It follows the boot architecture show under the Single Application. The flash memory map for the device is shown in Figure 42.

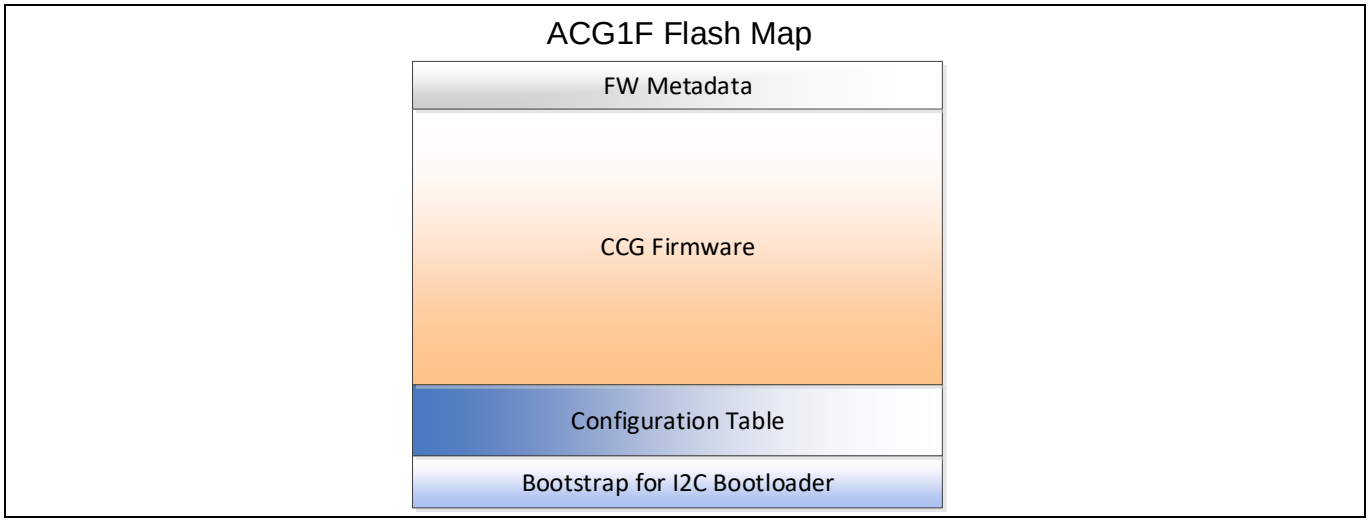


Figure 40 ACG1F firmware flash memory map

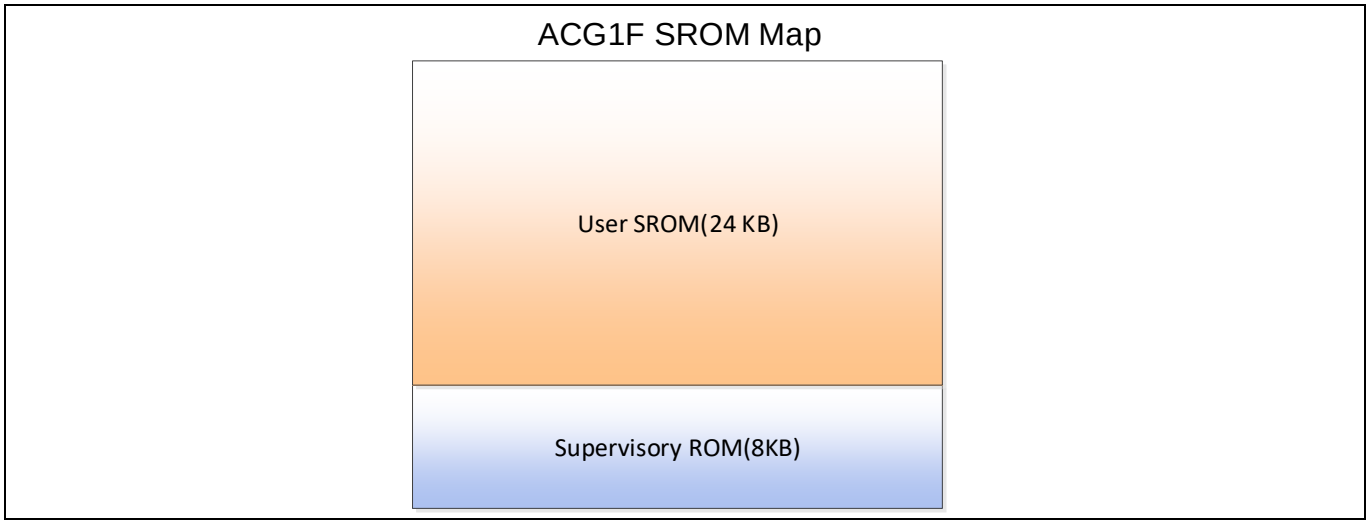


Figure 41 ACG1F firmware SROM memory map

5.4.2 CCG5/CCG5C/CCG6 flash memory map

CCG5/CCG5C/CCG6 has 128-KB flash memory that is designated to store a bootloader along with the primary and backup firmware applications. Each of the primary and backup firmware binaries are associated with the corresponding configuration table and metadata. The flash memory map for the device is shown in Figure 42.

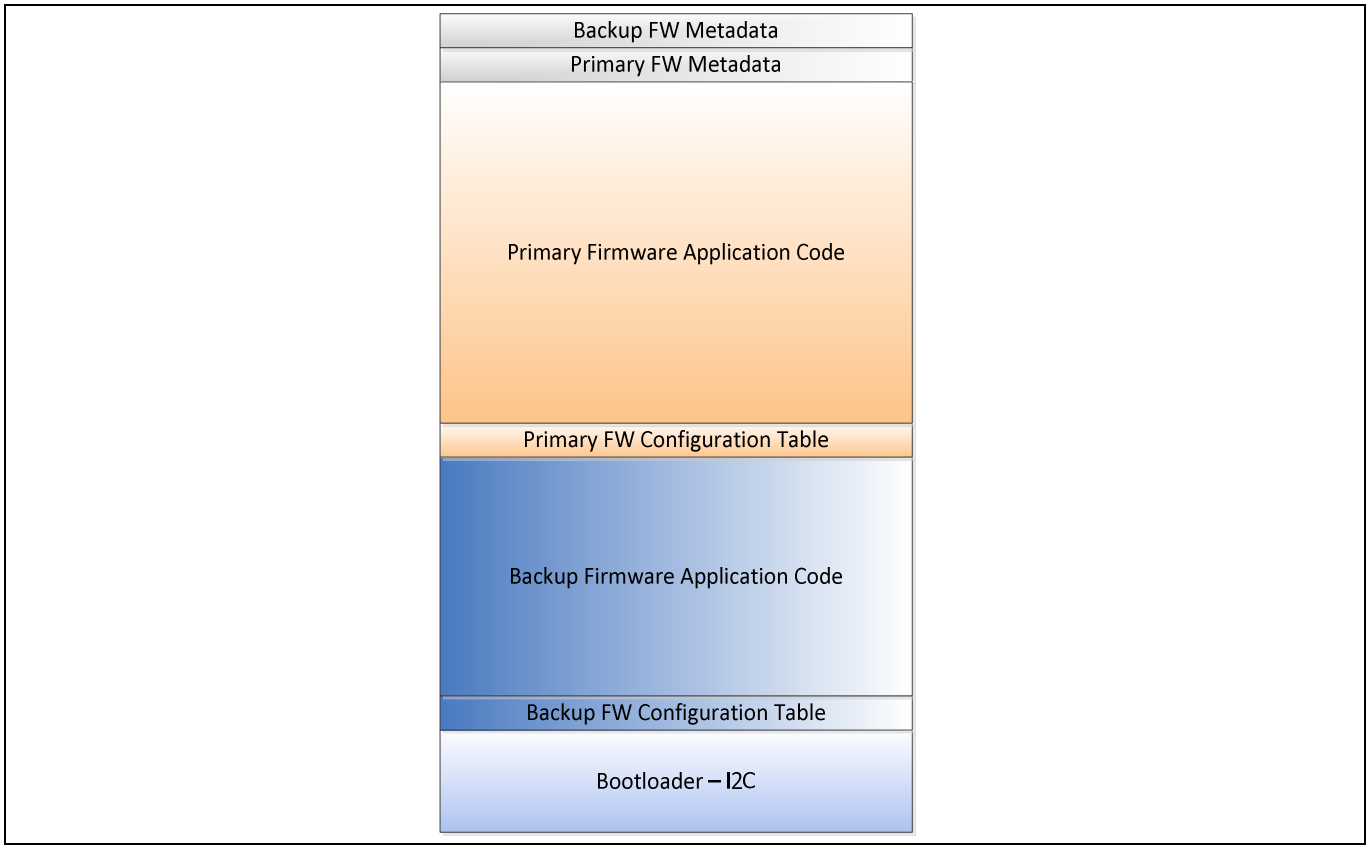


Figure 42 CCG5/CCG5C/CCG6 flash memory map

The bootloader is used to update firmware and configuration over the I2C-interface. It is allocated a fixed area. This memory area can only be written to from the SWD interface. The bootloader uses 5 KB of memory.

The configuration table holds the default PD configuration for the CCG application and is located at a fixed offset from the start of the firmware binary. The size of the configuration table for each application is 1 KB.

The metadata area holds metadata about the firmware binary. The firmware metadata follows the definition provided by the PSoC™ Creator bootloader component; and includes firmware checksum, size, and start address.

In addition to 128 KB of flash memory, the CCG6 device has 8 KB of user SROM. Common functions of both the backup firmware and the primary firmware have been placed in this user SROM.

5.4.3 CCG6DF firmware FLASH memory map

CCG6DF has 64 KB of FLASH and 96 KB of SROM of which 88 KB is User ROM.

CCG6DF firmware applications support two different boot architectures each of which has its own memory map format.

5.4.3.1 Hybrid Boot Architecture

When the Hybrid Boot Architecture is used, the bootloader functionality is combined with one of the application binaries, and this application is called as Secondary Base application. Only the Secondary Base application supports updating the flash.

This Secondary Base firmware supports reduced USB Type-C and PD functionality while the Primary application is being updated. In this architecture, the secondary base (backup) binary is fixed and cannot be updated in the field.

If the user wants to retain the ability to modify the configuration table used by the backup binary in the field, a secondary extension binary can be deployed in addition to the primary application. The secondary extension is a very limited application which has the capability to override the configuration parameters used by the secondary base.

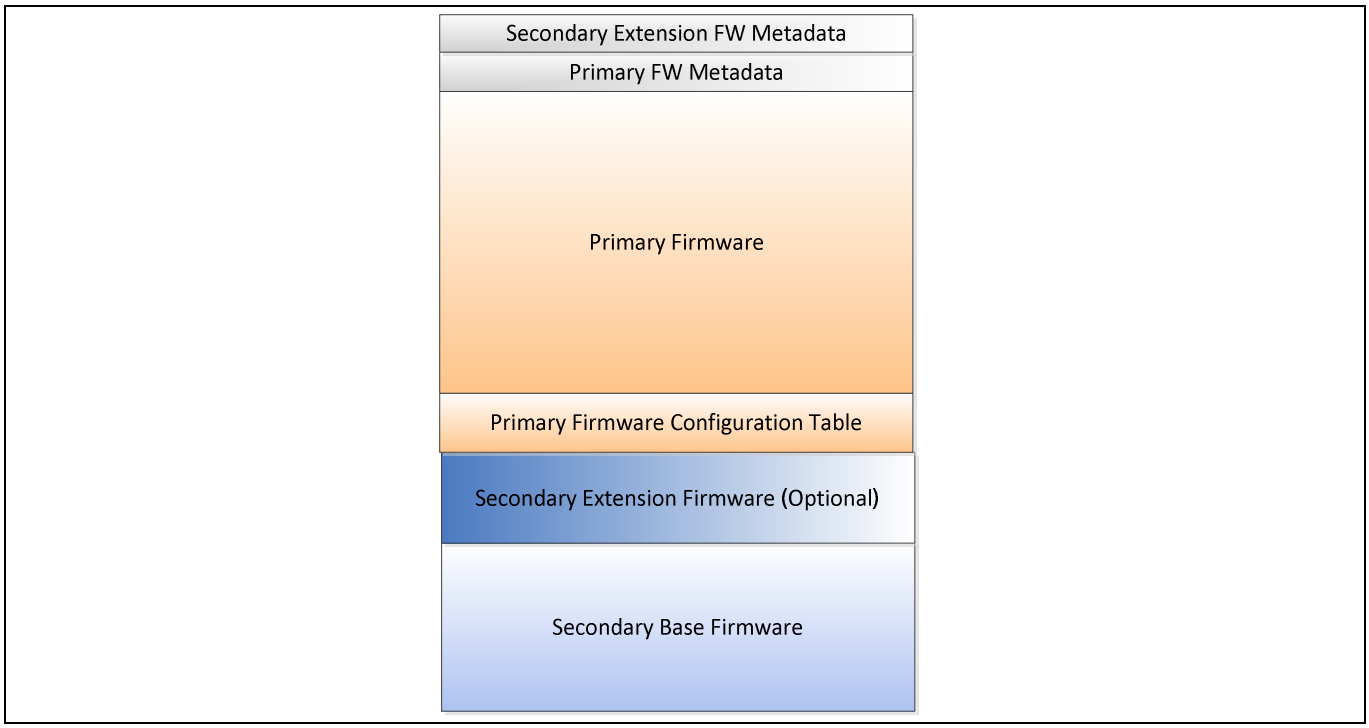


Figure 43 CCG6DF Hybrid Architecture flash memory map

5.4.3.2 Bootloader + Dual Asymmetric Application Architecture

It is possible to support the standard bootloader + dual asymmetric application architecture described in section ACG1F flash memory map on CCG6DF and CCG6SF devices as well. In such cases, the flash memory map will look the same as in Figure 42. The only difference is that the total flash size is 64 KB and the flash row size is 128 bytes.

5.5 Bootloader

The Flash-based bootloader mainly functions as a boot-strap and is the starting point for firmware execution. It validates the firmware based on the checksum stored in Flash. The boot-strap also includes the flashing module in notebook applications. The bootloader flow diagram follows.

In the case of a dual asymmetric binary architecture, the primary firmware always has higher priority and will be loaded if present. The bootloader only loads the backup firmware if the primary has been corrupted or when there is an explicit request to load the backup.

Figure 44 shows the bootloader flow diagram when a Dual Asymmetric binary architecture is used.

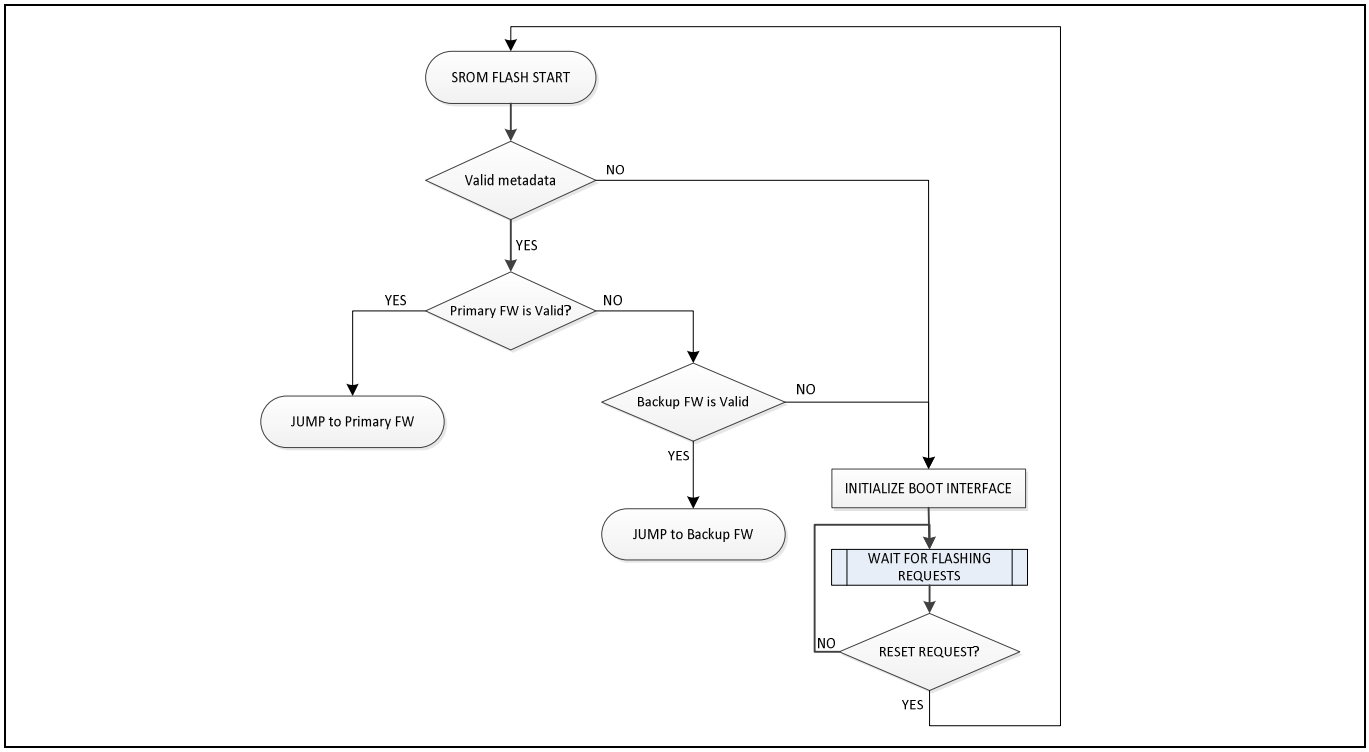


Figure 44 Dual Binary Bootloader flow diagram

5.5.1 Secondary base flow

When the bootloader functionality is integrated into the backup binary as in the case of the Hybrid Boot Architecture, the boot flow is different.

Figure 45 shows the flow diagram of the secondary base application. If the primary application is not present or corrupted, the secondary base looks for a valid secondary extension. If the secondary extension is present, the configuration table embedded in that binary is applied.

The secondary base then enables the PD port(s) in Sink-only mode and looks for connection while any I2C commands are being processed. This implementation allows the notebook platform to be powered from the Type-C port and enumerates any USB devices while the firmware update is being performed.

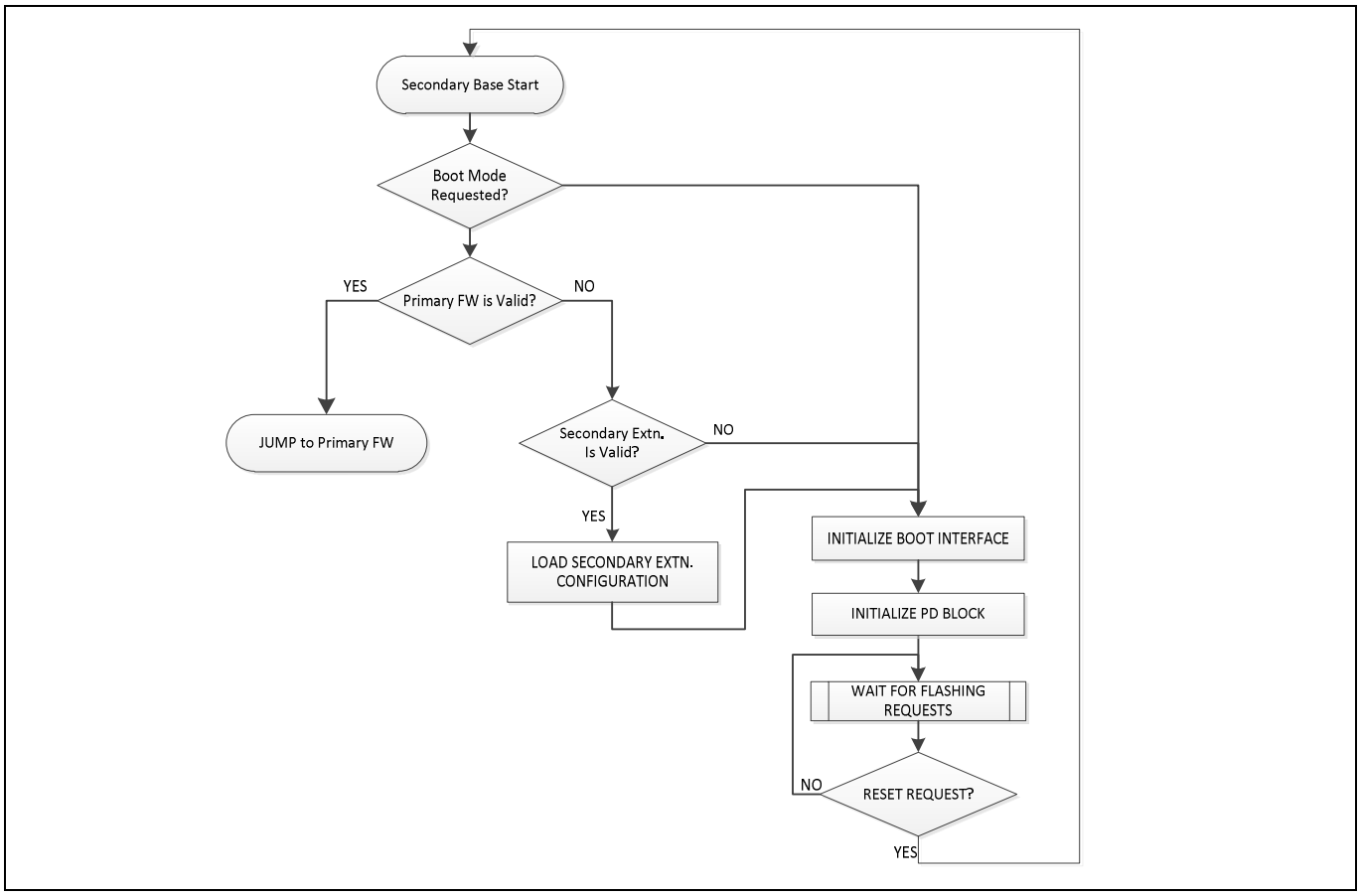


Figure 45 Secondary base boot flow diagram

5.6 Firmware operation

Figure 46 shows the firmware initialization and operation sequence. The notebook firmware is implemented in the form of a set of state machines and tasks that need to be performed periodically.

The code flow for the application is implemented in the ***common\main.c*** file. As can be seen from the main () function, the implementation is a simple round-robin loop, which services each of the tasks that the application has to perform.

All of the PD management, HPI command handling, and VDM handling are encapsulated in the task handlers in the CCGx Firmware Stack. See the CCGx FW API guide document for more details on these functions and handlers.

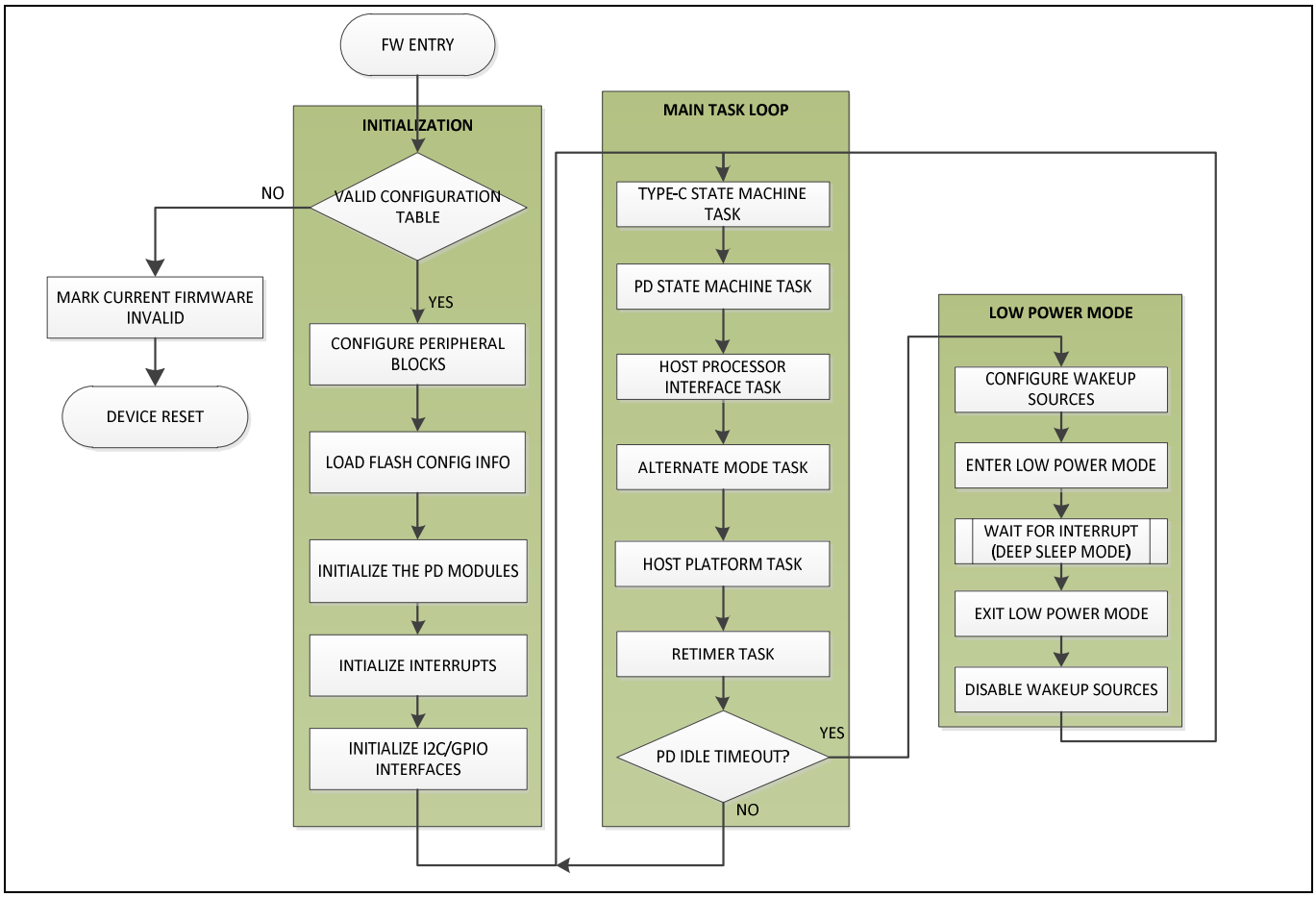


Figure 46 Notebook firmware flow diagram

5.6.1 Fault handling

Each CCGx device family supports different forms of fault detection and handling capabilities. Table 34 summarizes the various kinds of fault detection and handling supported in various applications in the SDK.

Table 34 Summary of fault handling features in notebook applications

Type of fault	CCG4 support	CCG3 support	CCG5 / CCG5C support	CCG6 / CCG6DF / CCG6SF Support	Comments
VBus overvoltage	Use external resistor divider and ADC for detection. Firmware based handling with ~50 µs latency (typical).	Internal resistor divider and dedicated OV comparator. Hardware can turn FET off on fault with programmable debounce.			Hard Reset and recovery will be attempted for a configurable number of retries. Firmware suspends the port and waits for physical disconnection after all retries have elapsed.
VBus overcurrent	No on-device support. Firmware supports interrupt input from external load switch. Firmware based handling with debounce in ms units.	High-side current sensing across a sense resistor. Firmware based FET turn-off after debounce in ms units.			
VBus short-circuit	Not supported	Not supported	Not supported	High-side current sensing across a sense resistor. Hardware based FET turn-off in µs units.	Hard Reset and recovery will be attempted till a configurable number of retries. Firmware suspends the port and waits for physical disconnection after all retries have elapsed.
VBus reverse-current	Not supported	Not supported	Not supported	High-side current sensing across a sense resistor; Comparator based higher VBus detection; and VBus OV based detection. Hardware based FET turn-off in µs units.	

Type of fault	CCG4 support	CCG3 support	CCG5 / CCG5C support	CCG6 / CCG6DF / CCG6SF Support	Comments
VConn overcurrent	Not supported	Not supported	Hardware based detection and automatic VConn switch disable.		Firmware exits any alternate modes which require VConn to be present. VConn will be re-enabled after a delay, if retries are enabled.
CC line overvoltage	Not supported	Not supported	Hardware based detection and CC line disable.		Firmware suspends the port and waits for physical disconnection.
SBU line overvoltage	Not supported	Not supported	Hardware based detection and SBU MUX disconnection.		
Over-temperature	Not supported	Not supported	Not supported	On CCG6DF and CCG6SF uses internal BJT based temperature sensing using the ADC for detecting the OTP condition	Firmware suspends the port and waits for physical disconnection.

6 Firmware APIs

This section provides a summary of the APIs provided by the PD stack and other layers in the CCGx firmware solution. Only the APIs that are expected to be used directly from user code are documented here. See the API reference guide for more details on the data structures used and APIs.

6.1 Data structures

Table 35 lists the important data structures used by the APIs.

Table 35 List of important data structures

Data structure	Description
dpm_pd_cmd_buf_t	Data structure holding the command parameters for DPM commands. See section 6.3.6.3 for usage.
dpm_status_t	The data structure holds the status information for the device policy manager. This data is retrieved by dm_get_info function. The application / solution layer can retrieve the DPM status using this. The data should not be modified outside of DPM. See section 6.3.6.4 for usage.
pd_do_t	Union to hold a PD data object. All USB PD data objects are 4-byte values which are interpreted according to the message type, length, and object position. This union represents all possible interpretations of a USB PD data object. Refer to the USB PD specification for details on each field.
pd_config_t	The data structure holds the device configuration data as stored by the EZ-PD™ Configuration Utility. These parameters are located at fixed offset from start of the firmware so that the device can be configured without having to recompile and modify the firmware binary. The structure consists of header fields holding header information as well as checksum followed by port specific configuration data.
pd_port_config_t	The data structure holds the port specific configuration data stored as part of pd_config_t structure. Refer to the API Reference Guide for more details.
ovp_settings_t	The data structure holds the over voltage protection settings selected via configuration utility. This allows the selection on fault thresholds, debounce period and retry count. See the API reference guide for more details.
ocp_settings_t	The data structure holds the over current protection settings selected via configuration utility. This allows the selection on fault thresholds, debounce period and retry count. See the API reference guide for more details.
scp_settings_t	The data structure holds the short-circuit protection settings selected via configuration utility. This allows the selection on fault thresholds, debounce period and retry count. See the API reference guide for more details.
rcp_settings_t	The data structure holds the reverse current protection settings selected via configuration utility. This allows the selection on retry count. See the API reference guide for more details.
app_cbk_t	This is a function pointer array for all PD stack handlers. The structure allows to override / customize handling for various PD stack controls from the solution space. See section 6.3.7.2 for usage.
app_resp_t	The data structure holds the response information for callbacks from the PD stack.

Data structure	Description
app_status_t	Various state machine parameters stored for rereference from the generic application implementation for the DPM handlers registered via app_cbk_t.

6.2 API summary

6.2.1 Device Policy Manager (DPM) API

These functions are declared in the ***src/pd_common/dpm.h*** header file.

Table 36 List of DPM APIs

Function	Description	Parameters	Return
dpm_init	Initialize the Device Policy Manager interface for a given USB PD port. For a dual-port part, the dpm_init needs to be done separately for each port.	port: Port to be initialized app_cbk: Structure with function pointers that the PD stack can call to handle various events.	Call status.
dpm_start	Start the PD state machine on the given USB PD port.	port: Port to be enabled.	Call status.
dpm_stop	Stop the PD state machine on the given USB PD port.	port: Port to be disabled	Call status.
dpm_disable	This function disables PD port operation and limits it to receiving hard reset signaling.	port: Port on which PD operation is to be disabled.	Call status.
dpm_deepsleep	Check for PD state machine idle state and prepare for deep sleep.	port: Port to be checked.	1 if deepsleep is possible. 0 if PD state machine is busy.
dpm_sleep	Check for PD state machine idle state and prepare for sleep.	port: Port to be checked.	1 if sleep is possible. 0 if PD state machine is busy.
dpm_wakeup	Update the PD block after device resumes from deep sleep.	port: Port to be re-enabled.	Always returns 1.
dpm_task	PD state machine task. This should be called periodically from the main application.	port: Port to be serviced.	Call status.

Firmware APIs

Function	Description	Parameters	Return
dpm_get_info	Get the DPM status for the device. This is mainly intended for use within other layers in the Infineon-provided firmware modules.	port: Port whose status is to be queried.	Pointer to the DPM status structure.
dpm_update_ def_cable_cap	Update the default cable current characteristics. The default spec limit is 3 A. But in captive cable designs, this can be overridden to match actual design.	def_cur: New default current setting in 10 mA units.	None.
dpm_get_ def_cable_cap	Get the current default cable capabilities current setting.	None	Default cable current setting in 10 mA units.
dpm_update_ snk_wait_cap_period	Update the sink wait for source capabilities period (tTypeCSnkWaitCap)	period: Wait period in ms.	None
dpm_get_ snk_wait_cap_period	Get the current tTypeCSnkWaitCap setting for the stack.	None	Wait period in ms.
dpm_update_ mux_enable_wait_period	Function to specify the delay to be used between the APP_EVT_TYPEC_ATTACH which is used to enable the Data Mux and the turning ON of the power source.	period: Delay to be provided (in ms) between MUX enable and VBus turning ON	None
dpm_get_ mux_enable_wait_period	Returns the current delay between the MUX enable and VBus turn ON.	None	Current delay in ms
dpm_pd_command	Initiate a PD command such as VDM, DR_SWAP etc.	port: Port on which command is to be initiated. cmd: Command to be initiated. buf_ptr: Command parameters. cmd_cbk: Callback to be called at the end of command.	Call status.

Firmware APIs

Function	Description	Parameters	Return
dpm_typec_command	Initiate a Type-C command such as Rp change.	port: Port on which command is to be initiated. cmd: Command to be initiated. cmd_cbk: Callback to be called at the end of command.	Call status.
dpm_update_ swap_response	Update the response that CCG will send for various swap commands.	port: Port whose swap handler is to be changed. value: Bitmap in the following format. Bits 1:0 => DR_SWAP response Bits 3:2 => PR_SWAP response Bits 5:4 => VCONN_SWAP response 0 => ACCEPT 1 => REJECT 2 => WAIT 3 => NOT_SUPPORTED	Call status.
dpm_update _src_cap	Update the source capabilities PDO list. The provided values will replace the settings from the configuration table.	port: Port to be updated. count: Number of PDOs in list. Maximum allowed value is 7 pdo: Pointer to array containing PDOs.	Call status.
dpm_update _src_cap_mask	Change the mask that enables specific source PDOs from the list.	port: Port to be updated. mask: New PDO enable mask.	Call status.

Firmware APIs

Function	Description	Parameters	Return
dpm_update _snk_cap	Update the sink capabilities PDO list. The provided values will replace the settings from the configuration table.	port: Port to be updated. count: Number of PDOs in list. Maximum allowed value is 7 pdo: Pointer to array containing PDOs.	Call status.
dpm_update_ snk_cap_mask	Change the mask that enables specific sink PDOs from the list.	port: Port to be updated. mask: New PDO enable mask.	Call status.
dpm_update_ snk_max_min	Change the min/max current fields associated with each Sink PDO.	port: Port to be updated. count: Number of PDOs in list. Maximum allowed value is 7 max_min: Pointer to array containing new Min/Max operating current values.	Call status.
dpm_update_ port_config	Change the USB PD configuration: port role, default role etc. The port should have been disabled using dpm_typec_command (DPM_CMD_PORT_DISABLE) before the change is attempted.	port: Port to be updated. role: New port role setting. dflt_role: New default port role for DRP. toggle_en: DRP toggle enable flag try_src_en: Try.SRC enable flag	Call status.
dpm_is_rdo_valid	Generic stack implementation to verify an RDO with the current source capabilities.	port: Port to be checked. rdo: The RDO to be verified.	Call status.
dpm_get_polarity	Get the current polarity of the Type-C connection	port: Port to be queried.	0 if CC1 is connected 1 if CC2 is connected

Firmware APIs

Function	Description	Parameters	Return
dpm_typec_ deassert_rp_rd	De-assert both Rp and Rd on the specified PD port.	port: Port to be updated. channel: Channel on which terminations are to be disabled.	Call status.
dpm_update_ port_status	Update the USB PD port status that will be returned by CCG as part of a Get_Status response.	port: Port to be updated. status_p: Pointer to buffer containing buffer status. offset: Number of bytes of offset to be applied while updating the status. byte_cnt: Number of bytes of status being updated.	None
dpm_get_ pd_port_status	Get the current USB PD port status.	port: Port to be queried	32-bit port status.
dpm_downgrade_ pd_port_rev	Downgrade the PD port revision from 3.0 to 2.0.	port: Port to be updated.	Call status
dpm_update_ ext_src_cap	Updated the Extended Source Capabilities returned by the CCG device.	port: Port to be updated. buf_p: Pointer to buffer containing the extended source capabilities.	None
dpm_update_ frs_enable	Update the Fast-Role Swap feature support in the CCG PD state machines. The change will only take effect after a fresh contract negotiation.	port: Port to be updated. frs_rx_en: Whether FRS receive is to be enabled. frs_tx_en: Whether FRS transmit is to be enabled.	None
dpm_prot_reset	Resets protocol layer TX and RX counter for a specific sop type.	port: Port to be updated sop: SOP type to do the reset.	Call status

Firmware APIs

Function	Description	Parameters	Return
dpm_prot_reset_rx	Resets protocol layer RX only counter for a specific sop type.	port: Port to be updated sop: SOP type to do the reset.	Call status
dpm_pe_stop	This function stops the policy engine. Used in fault scenario wherein PD protocol need to be stopped but type c manager still runs.	port: Port to be updated	Call status
dpm_set_alert	Sets alert ADO on fault.	port: Port to be updated alert_ado: The ADO object to be updated	Call status
dpm_clear_ hard_reset_count	Clears the hard reset count.	port: Port to be updated	Call status
dpm_set_ fault_active	Indicate that there is a fault condition active.	port: Port to be updated	Call status
dpm_clear_ fault_active	Clear all fault conditions	port: Port to be updated	Call status
dpm_send_ hard_reset	Function to send a HardReset to the port partner	port: Port on which Hard Reset needs to be sent. reason: Reason for the hard reset. Used for internal status tracking only.	Call status
dpm_update_ ndiscover_ identity_count	Function to change the maximum number of Discover Identity messages sent to the EMCA from the default value of nDiscoverIdentityCount.	count: Number of Discover Identity messages that need to be sent. Should be non-zero.	None
dpm_get_stack_ config	Function to retrieve the configurable switch values at runtime	None	Structure indicating current stack configuration.
dpm_refresh_ src_cap	Regenerate the source capabilities from the available PDOs and PDO mask. Also update current setting based on the cable characteristics.	port: Port to be updated	0 – If failed 1 – If successful

Function	Description	Parameters	Return
dpm_refresh_ snk_cap	Regenerate the sink capabilities from the available PDOs and PDO mask.	port: Port to be updated	0 – If failed 1 – If successful
dpm_update_ mux_enable_ wait_period	Update the delay to be used between MUX enable and VBus enable.	period: Delay in ms.	None

6.2.2 Application layer API

Table 37 lists the application layer APIs provided by the CCGx Host SDK. These function declarations and definitions can be found under the ***src/app*** folder.

Table 37 List of application layer APIs

Function	Description	Parameters	Return
app_init	Initializes the application layer for operation.	None	None
app_task	Perform application layer tasks. This includes VDM handling and alternate mode implementation.	port: Port on which the application task is to be performed.	1 if successful. 0 if failure
app_event_handler	Application level handler for PD stack events. This updates the internal application state, and then calls the solution level event handler.	port: Port on which event occurred. evt: Type of event. dat: Event data provided by stack.	None
app_get_status	Get the current application status information.	port: Port to be queried.	Pointer to application status structure.
app_sleep	Check whether the application layer is ready for device low power mode.	None	true if application layer is idle. false if application layer is busy.
app_wakeup	This is called after device wakes up from sleep, and can be used to restore any state that was saved as part of sleep entry.	None	None
system_sleep	Top level sleep mode entry function. This should be called from the main loop to allow CCG device to consume minimal power.	None	None

Firmware APIs

Function	Description	Parameters	Return
eval_src_cap	Evaluates the source capabilities advertised by the port partner and identifies the optimal power contract setting.	port: PD port on which SRC. CAP has been received. src_cap: Source capabilities that were received. app_resp_handler: Callback function to be called to report decision.	None
eval_rdo	Evaluate a PD request (RDO) received and decide whether to accept/reject.	port: PD port on which request has been received. rdo: Received RDO value. app_resp_handler: Callback function to be called to report decision.	None
psnk_set_voltage	Power sink (consumer) handler for voltage change. Default implementation sets up the OVP voltage level based on the provided voltage.	port: Port to be updated. volt_50mV: Expected VBus voltage in 50 mV units.	None
psnk_set_current	Power sink (consumer) handler for operating current change. The default implementation does not do anything.	port: Port to be updated. cur_10 mA: Expected operating current in 10 mA units.	None
psnk_enable	Enable the power sink path.	port: Port to be updated.	None
psnk_disable	Disable the power sink path.	port: Port to be updated.	None

Firmware APIs

Function	Description	Parameters	Return
psrc_set_voltage	Set the desired voltage for the power source (provider) output. This function is expected to make the regulator updates as well as set the OVP thresholds based on the voltage. This can be updated if the voltage selection mechanism should be changed.	port: Port to be updated. volt_mv: Expected VBus voltage in mV units.	None
psrc_set_current	Set the current level for the power source output. The default implementation does not do anything.	port: Port to be updated. cur_10mA: Expected operating current in 10 mA units.	None
psrc_enable	Enable the power source output. The power_ready_handler is called once Vbus voltage has stabilized at the desired level.	port: Port to be updated. pwr_ready_handler: Application handler callback function.	None
psrc_disable	Disable the power source output. The power_ready_handler is called once Vbus voltage has stabilized at vSafe0V.	port: Port to be updated. pwr_ready_handler: Application handler callback function.	None
vconn_enable	Enable the VConn supply.	port: Port to be updated. channel: Selected CC line	Call status.
vconn_disable	Disable the VConn supply.	port: Port to be updated. channel: Selected CC line	Call status.
vconn_is_present	Check whether VConn supply is present.	port: Port to be queried.	true if VConn is present false if VConn is absent.

Firmware APIs

Function	Description	Parameters	Return
vbus_is_present	Check whether VBus is present within a specific range.	port: Port to be queried. volt: Expected VBus voltage. per: Allowed variance in voltage as percentage of expected voltage.	true if VBus is in range. false if VBus is not in range.
vbus_discharge_on	Enable the VBus discharge path.	port: Port to be updated.	None
vbus_discharge_off	Disable the VBus discharge path.	port: Port to be updated.	None
eval_dr_swap	Evaluate a DR_SWAP request from the port partner.	port: Port to be updated. app_resp_handler: Callback function to be called to report decision.	None
eval_pr_swap	Evaluate a PR_SWAP request from the port partner.	port: Port to be updated. app_resp_handler: Callback function to be called to report decision.	None
eval_vconn_swap	Evaluate a VCONN_SWAP request from the port partner.	port: Port to be updated. app_resp_handler: Callback function to be called to report decision.	None
vdm_data_init	Initialize the VDM handler data structure with data from configuration table.	port: Port to be updated.	None

Firmware APIs

Function	Description	Parameters	Return
vdm_update_data	Update the VDM handler data structure with custom data.	port: Port to be updated. id_vdo_cnt: Number of DOs in Discover ID response. id_vdo_p: Array containing the Discover ID response. svid_vdo_cnt: Number of DOs in Discover SVID response. svid_vdo_p: Array containing the Discover SVID response. mode_resp_len: Total length of all Discover Mode responses. mode_resp_p: Array containing actual Discover Mode responses.	None
eval_vdm	Evaluate a received PD VDM and respond to it.	port: Port on which VDM is received. vdm: Received VDM pointer. vdm_resp_handler: Callback to be notified about the VDM response.	None

Function	Description	Parameters	Return
app_ovp_enable	Enable the overvoltage Protection function.	Port: Port on which OVP is to be enabled. volt_50mV: Allowed maximum voltage in 50 mV units. pfet: Whether the CCG device is power source. ovp_cb: Callback function to be called when OVP is detected.	None
app_ovp_disable	Disable the OVP function on a PD port.	port: Port on which OVP is to be disabled. pfet: Whether CCG is a power source at this time.	None

Table 38 lists the functions that the PD stack and application layer expect to be implemented at the solution level. These functions must be implemented in the source files within the PSoC™ Creator project workspace. If the target application does not require one or more of these functions; a stub implementation that does nothing should still be provided.

Table 38 Solution-level functions

Function	Description	Parameters	Return
mux_ctrl_init	Initialize the Type-C switch and corresponding control interface. The Type-C data pins should be isolated from the USB and DisplayPort controller pins at this stage.	port: Port to be updated.	true if successful. false if failure
mux_ctrl_set_cfg	Update the Type-C switch to enable/disable the desired USB and DisplayPort connections.	port: Port to be updated. cfg: Desired data connection mode. polarity: Polarity of current Type-C connection. 0 for CC1 and 1 for CC2.	true if successful. false if failure

Function	Description	Parameters	Return
sln_pd_event_handler	This is top level handler for system event notifications provided by the PD stack. The default implementation of this function calls the HPI event handler so that the EC can be notified about these events.	port: Port on which event occurred. evt: Type of event. data: Event data provided by stack.	None
app_get_callback_ptr	Function that returns a structure filled with callback function pointers for various system events. Default implementations for all of these functions are provided under the app folder, and the structure can be initialized with the corresponding pointers.	port: Port to be queried.	Pointer to structure containing callback function pointers. This structure should remain valid throughout the device operation.

6.2.3 Alternate mode API

This section documents the alternate mode-related API provided in the CCGx Host SDK. These APIs are defined in the sources under ***src/app/alt_mode*** and are summarized in Table 39.

Table 39 List of alternate mode APIs

Function	Description	Parameters	Return
enable_vdm_task_mngr	Enabled the alternate mode manager.	port: Port to be updated.	None
vdm_task_mngr_deinit	De-initialize the alternate mode manager.	port: Port to be updated.	None
is_vdm_task_idle	Check if the alternate mode manager is idle, so that device can enter low power mode. Each port has to be queried separately.	port: Port to be queried.	true if the manager is idle. false if the manager is busy.
vdm_task_mngr	Alt. mode manager state machine task. This is called from app_task.	port: Port to be serviced.	None
eval_rec_vdm	Evaluate VDM message received.	port: Port to be updated. vdm_rcv: Pointer to received attention VDM.	true if VDM is to be ACKed. false if VDM is to be NACKed.

6.2.4 Hardware Adaptation Layer (HAL) API

This section documents the API provided as part of the Hardware Adaptation Layer (HAL), which provides drivers for various hardware blocks on the CCG device.

6.2.4.1 GPIO API

The PSoC™ Creator GPIO component and associated APIs can be used in all CCG projects. However, the SDK also provides a set of special API for reduced memory footprint. These APIs are defined in the ***src/system/gpio.c*** file and are summarized in Table 40.

Table 40 List of GPIO APIs

Function	Description	Parameters	Return
hsiom_set_config	Update the IO matrix configuration for a given pin.	port_pin: CCG pin identifier. hsiom_mode: Desired IO configuration	None
gpio_set_drv_mode	Select GPIO drive mode for a given pin.	port_pin: CCG pin identifier. drv_mode: Desired drive mode	None
gpio_hsiom_set_config	Update IO matrix and drive mode for a given pin.	port_pin: CCG pin identifier. hsiom_mode: Desired IO configuration drv_mode: Desired drive mode value: Desired output state	None
gpio_int_set_config	Configure interrupt associated with a given pin.	port_pin: CCG pin identifier. int_mode: Desired interrupt configuration.	None
gpio_set_value	Update the output value of a given pin. The IO configuration and drive mode for the pin should have been set separately.	port_pin: CCG pin identifier. value: Desired output state	None
gpio_read_value	Get the current state of a given pin.	port_pin: CCG pin identifier.	true if the pin is high. false if the pin is low.
gpio_get_intr	Check if there is an active interrupt associated with the given pin.	port_pin: CCG pin identifier.	true if interrupt is active. false if interrupt is not active.

Function	Description	Parameters	Return
gpio_clear_intr	Clear any interrupts associated with the given pin.	port_pin: CCG pin identifier.	None

6.2.4.2 I2C API

The Serial Communication Block component in PSoC™ Creator can be used with the CCG device. However, the SDK provides a dedicated I^2C slave mode driver, which is optimized for HPI implementation. These API definitions are provided in ***src/scb/i2c.c*** and are summarized in Table 41.

Table 41 List of I2C driver APIs

Function	Description	Parameters	Return
i2c_scb_init	Initialize the I2C slave block and set driver parameters.	scb_index: SCB index to be used. mode: Mode of I2C block operation. clock_freq: Expected bit rate on the interface. slave_addr: Slave address to be used. slave_mask: Mask to be applied on the slave address to detect I2C addressing. cb_fun_ptr: Callback function to be called for read/write/error notifications. scratch_buffer: Pointer to scratch buffer to be used to received incoming data. scratch_buffer_size: Size of scratch buffer.	None
i2c_scb_deinit	De-initialize the specified I2C slave block.	scb_index: SCB index to be used.	None

Firmware APIs

Function	Description	Parameters	Return
i2c_scb_write	Write data into the I2C block transmit FIFO.	scb_index: SCB index to be used. source_ptr: Location of data to be written. size: Size of data to be written. Should be 8 bytes or lesser. count: Return parameter indicating actual size of data written.	None
i2c_reset	Reset the I2C block.	scb_index: SCB index to be used.	None
i2c_slave_ack_ctrl	Used to enable/disable clock stretching in the device address stage.	scb_index: SCB index to be used. enable: Enable device address acknowledgement.	None
i2c_scb_is_idle	Check whether the I2C module is idle.	scb_index: SCB index to be used.	true if the block is idle. false if the block is busy.
i2c_scb_enable_wakeup	Enable I2C device addressing as a wakeup source from low power mode.	scb_index: SCB index to be used.	None

6.2.4.3 Flash API

The flash API provides the core functionality used for CCG configuration and firmware updates. These are wrappers over the PSoC™ Creator provided flash APIs, and implement checks to ensure that a firmware binary is not corrupted by writing while it is being accessed. The flash related API are defined in ***src/system/flash.c*** and are summarized in Table 42.

Table 42 List of flash API

Function	Description	Parameters	Return
flash_enter_mode	Enable flash updates through the specified interface. Flash updates are only allowed through one interface (I2C, CC, etc.) at a time.	is_enable: Whether to enable or disable flash access. mode: Flash access interface data_in_place: Specifies whether the data to be written to flash should be used in place, or a copy should be made on the stack.	None
flash_access_enabled	Check whether flash access through any of the specified interfaces is enabled.	modes: Bitmap representing the flash interfaces to be checked.	true if flash access is enabled. false otherwise.
flash_set_access_limits	Set limits regarding the flash rows that can be accessed. The firmware application should identify the memory range that it is using, and use this API to protect it from updates.	start_row: Lowest flash row that can be accessed. last_row: Highest flash row that can be accessed. md_row: Metadata row that can be accessed. bl_last_row: Last row used by boot-loader. Any flash row above this value can be read.	None
flash_row_clear	Clear the contents of the specified flash row.	row_num: Row to be cleared.	Flash erase status

Function	Description	Parameters	Return
flash_row_write	Write the desired content into a flash row. This is a blocking operation.	row_num: Flash row to be updated. data: Buffer containing flash data. cbk: Must be zero as non-blocking writes are not supported by the device.	Flash write status
flash_row_read	Read the content of a flash row into the provided buffer.	row_num: Flash row to be read. data: Buffer to read the data into.	Flash read status.

6.2.4.4 Timer API

The CCG firmware stack uses a soft timer implementation for various timing measurements. The soft timer granularity is 1 ms, and it uses a single hardware timer. If the timer block used is WDT (WatchDog Timer), the timers can be used across device sleep modes; and it is possible to use a tickless implementation which reduces interrupt frequency. The soft timer-related APIs are defined in ***src/system/timer.c*** and are summarized in Table 43.

Table 43 List of timer API

Function	Description	Parameters	Return
timer_init	Initialize the timer module. This enables the hardware timer and interrupt as well.	None	None
timer_start	Start one soft timer (one shot).	instance: Soft timer group or instance. Separate timer groups are maintained for each PD port. id: ID of timer to be started. Timer IDs are assigned statically. period: Timer period in milliseconds. cb: Timer expiry callback.	true if successful. false if failure.
timer_stop	Stop a running soft timer.	instance: Soft timer group or instance. id: ID of timer to be stopped.	None

Function	Description	Parameters	Return
timer_is_running	Check whether a soft timer is running.	instance: Soft timer group or instance. id: ID of timer to be queried.	true if running. false if not running.
timer_stop_all	Stop all soft timers in a group.	instance: Timer group to be stopped.	None
timer_stop_range	Stop all soft timers whose IDs fall in a range.	instance: Timer group to be stopped. start: Lowest timer ID to be stopped. stop: Highest timer ID to be stopped.	None
timer_num_active	Get number of active timers in a group.	instance: Soft timer group or instance.	Count of active timers.
timer_enter_sleep	Prepare the timer module for sleep entry.	None	None

6.2.5 Firmware update API

CCG application support firmware updates through interfaces like HPI (I2C) and CC (Unstructured VDMs). The firmware update APIs are common functions that are used by each of these protocol modules to implement the firmware update functionality.

The firmware update-related APIs are defined in ***src/system/boot.c*** and are summarized in Table 44.

Table 44 Firmware update API

Function	Description	Parameters	Return
boot_validate_fw	Validate the firmware image in device flash.	fw_metadata: Pointer to metadata regarding the firmware image.	Valid/invalid status.
boot_validate_ configtable	Validate the configuration table in the device flash.	table_p: Pointer to the configuration table.	Valid/invalid status.
get_boot_mode_reason	Compute the boot mode reason register value by validating firmware and configuration tables.	None	Boot mode reason bitmap value.
boot_get_boot_seq	Get the flash sequence number associated with a given firmware binary.	fwid: ID of firmware binary to be queried.	Flash sequence number value.
sys_set_device_mode	Set the current firmware mode for the CCG device.	fw_mode: Firmware mode to be set.	None

Function	Description	Parameters	Return
sys_get_device_mode	Get the current firmware mode for the CCG device.	None	Current firmware mode.

6.3 API usage examples

This section provides a few examples for the usage of the APIs documented under the section API summary. See the API reference guide for more details.

Most of the PD operations are initiated using the dpm_pd_command() and dpm_typec_command() APIs. These APIs are non-blocking and only initiate the operation. A callback function can be passed to the API; and it will be called on completion of the operation. Completion of these operations will require the tasks in the main loop to be executed, and therefore, the caller cannot block waiting for the callback to arrive.

If there is a need to wait for the operation to complete and then initiate other operations, this can be done in two ways:

1. Initiate the follow-on operations from the callback function itself.
2. Modify the main loop to detect the callback arrival, and then initiate the next operation after this.

6.3.1 Boot API usage

The bootloader and firmware application communication in the CCGx Host SDK is built using the PSoC™ Creator Bootloader and Bootloadable components. This section shows how the PSoC™ Creator bootloader and bootloadable components along with the wrapper APIs in the SDK to transfer control from the application firmware to the bootloader or to the application in the alternate memory bank.

6.3.1.1 Perform device reset

Since the order in which the bootloader prioritizes firmware images is fixed, resetting the device causes the device to boot back into the same mode that it previously was in. The CySoftwareReset() API can be used to initiate a CCG device reset.

Code Listing 1

```
/* Include relevant header files. */
#include <project.h>

void reset_ccgx_device (void)
{
      /* Initiate device reset. */
      CySoftwareReset ();
}
```

6.3.1.2 Jump to bootloader

This operation is not required because firmware update and flash read functionality is provided by the application firmware itself. Also, the bootloader is a fixed binary application which cannot be updated to include additional functionality.

However, you can use the Bootloadable component API to transfer control to the bootloader from the application firmware. You can do this by specifying the boot type for the next run using the Bootloadable_SET_RUN_TYPE() macro and then initiating a reset using CySoftwareReset().

Code Listing 2

```
/* Include relevant header files. */
#include <project.h>
#include <boot.h>

void jump_to_bootloader(void)
{
      /* Select the boot mode for the next run. */
      Bootloadable_SET_RUN_TYPE(CCG_BOOT_MODE_RQT_SIG);
      /* Initiate device reset. */
      CySoftwareReset();
}
```

6.3.1.3 Jump to alternate firmware

As described in section 5, the CCGx firmware projects are set up such that they generate two copies of the application firmware. While both of these copies are expected to be equivalent, there may be cases where you need to use a fixed backup firmware along with a dynamically updated primary firmware. In such cases, it would be desirable to transfer control to the backup firmware in order to update the primary firmware.

The APIs shown in the section Firmware update API can also be used to transfer control to the alternate firmware binary. You will need to determine the identity of the current firmware binary (FW1 or FW2) and then initiate the switch accordingly.

Code Listing 3

```
/* Include relevant header files. */
#include <project.h>
#include <boot.h>

void jump_to_alternate_fw(void)
{
      /* Set the next boot mode based on the current FW ID. */
      if (sys_get_device_mode() == SYS_FW_MODE_FWIMAGE_1)
```

Code Listing 3

```
        {
                Bootloadable_SET_RUN_TYPE (CCG_FW2_BOOT_RQT_SIG);
        }
        else
        {
                Bootloadable_SET_RUN_TYPE (CCG_FW1_BOOT_RQT_SIG);
        }

        /* Initiate device reset. */
        CySoftwareReset();
}
```

6.3.2 GPIO API usage

All of the APIs provided by the PSoC™ Creator Pins component can be used in CCGx firmware solutions. In addition to these, specific APIs to perform common GPIO functions are provided in the CCGx Host SDK. The list of GPIO APIs is provided in the section GPIO API.s

6.3.2.1 Configuring a CCGx pin as an edge triggered interrupt input

The gpio_hsiom_set_config() API is used to set the I/O mapping and drive mode settings for a given CCGx pin. The gpio_int_set_config() API is used to enable interrupt functionality on a CCGx pin. The following code snippet shows how the pin P3[1] on CCGx can be configured as an input signal triggering interrupts on a falling edge.

Code Listing 4

```
/* We are using P3.1 as the interrupt pin. */
#define INTR_GPIO_PORT_PIN            (GPIO_PORT_3_PIN_1)

/* The ISR vector number corresponds to PORT3. */
#define CCGX_PORT3_INTR_NO            (3u)

/* ISR for the GPIO interrupt. */
CY_ISR (gpio_isr)
{
        /* Clear the interrupt. */
        gpio_clear_intr (INTR_GPIO_PORT_PIN);

        /* Custom interrupt handling actions here. */
        ...;
```

Code Listing 4

```
}

/* Function to configure and enable the interrupt. */
void configure_intr_input (void)
{
      /* Configure the IO modes for the pin. */
      gpio_hsiom_set_config (INTR_GPIO_PORT_PIN,
            HSIOM_MODE_GPIO, GPIO_DM_HIZ_DIGITAL, false);

      /* Configure the interrupt mode for the pin. */
      gpio_int_set_config (INTR_GPIO_PORT_PIN,
            GPIO_INTR_FALLING);

      /* Set the ISR routine and enable the interrupt. */
      CyIntSetVector (CCGX_PORT3_INTR_NO, gpio_isr);
      CyIntEnable (CCGX_PORT3_INTR_NO);
}
```

6.3.2.2 Connecting a pin to the internal ADC

See the CCGx device datasheet to identify pins that can be connected to the internal ADC blocks through the Analog MUX configuration. The hsiom_set_config() API can be used to connect a specific pin to the ADC.

Code Listing 5

```
#define VBUS_MON_PORT_PIN (GPIO_PORT_3_PIN_1)

void connect_vbus_mon_to_adc (void)
{
      /* Connect the pin to AMUXB. */
      hsiom_set_config (VBUS_MON_PORT_PIN, HSIOM_MODE_AMUXB);
}
```

6.3.3 Timer API usage

The CCGx Host SDK provides a soft timer module, which can be used for task scheduling. The timer APIs allow users to create one-shot timer objects with callback notification on timer expiry.

Soft timers are identified using a single-byte timer ID, and the caller should ensure that the timer ID used does not collide with timers used elsewhere. This is facilitated by reserving the timer ID range from 0xE0 to 0xFF for use by the user application code. These timer IDs are not used internally within the CCGx firmware stack and are safe for use.

A soft timer is started using the timer_start() API and can be aborted using the timer_stop() API.

Code Listing 6

```
#define APP_TIMER_ID            (0xF0)

static void timer_expiry_callback(uint8_t instance, timer_id_t id)
{
      /* Start the desired task here. */
      ...;
}

/* Use a timer to schedule task to be run delay_ms milliseconds later. */
void schedule_task(uint16_t delay_ms)
{
      /* Start an application timer to wait for delay_ms.

         Devices with two USB-PD ports support two sets of timers, and
         the set to be used is selected using the first parameter. */
      timer_start(0, APP_TIMER_ID, delay_ms, timer_expiry_callback);
}
```

6.3.4 HPD API usage

The ***HotPlugDetect*** output pin from CCGx is used in Notebook implementations to signal interrupts from the far-end DisplayPort (DP) peripheral to the DP controller in the Notebook system. The DP peripheral will signal HPD events to the CCGx Notebook controller through PD messages, and CCGx will relay these HPD events to the DP controller through the GPIO output.

The hpd_transmit_init() and hpd_transmit_sendevt() APIs are used to initialize the HPD transmit logic and to send event notifications to the DP controller respectively.

Code Listing 7

```
/* Callback that notifies user of completion of HPD signaling. */
static void hpd_callback(uint8_t port, hpd_event_type_t event)
```

Code Listing 7

```
{
      if (event == HPD_COMMAND_DONE)
      {
            /* Requested HPD command is complete. */
      }
}

/* Initialize the HPD transmit logic for USB-PD port 0, and register
   the command completion callback. */
void initialize_hpd_logic(void)
{
      hpd_transmit_init(0, hpd_callback);
}

/* Send HPD_IRQ to the DP controller. */
void send_hpd_irq(void)
{
      /* Asynchronous mode: Do not wait for completion. */
      hpd_transmit_sendevt(0, HPD_EVENT_IRQ, false);
}
```

6.3.5 Sleep mode control

The decision to enter the device deep sleep mode to save power is made at the application level. The system_sleep() function call in the main loop can be disabled if the deep sleep mode entry is to be disabled.

6.3.6 DPM API usage

6.3.6.1 Enabling a PD port

The dpm_start() API can be used to enable a PD port for operation. The dpm_init() API should have been called prior to doing this.

Code Listing 8

```
bool enable_pd_port(uint8_t port)
{
      if (dpm_start(port) == 0)
      {
            /* DPM start failed. Handle errors. */
            return (false);
      }
return (true);
}
```

6.3.6.2 Disabling a PD port

The dpm_stop() API should not be used to directly disable a PD port, as the port might already be in the contract. The dpm_typec_command() API must be used to initiate the DPM_CMD_PORT_DISABLE command. This will ensure that the port is disabled safely and the VBus voltage is discharged to a safe level before the completion callback is issued.

Code Listing 9

```
static volatile bool pd_disable_completed = true;
static volatile bool pd_disable_issued    = false;

/* Callback for the PD disable command. */
static void pd_port_disable_cb(uint8_t port, dpm_typec_cmd_resp_t resp)
{
      pd_disable_completed = true;
      /* Other APIs can be started here, if required. */
}

bool disable_pd_port(uint8_t port)
{
      /* Store state of operation. */
      pd_disable_issued    = true;
      pd_disable_completed = false;
```

Code Listing 9

```
	/* Initiate port disable. */
	if (dpm_typec_command(port, DPM_CMD_PORT_DISABLE,
			pd_port_disable_cb) != CCG_STAT_SUCCESS)
	{
		/* Handle error here. */
		pd_disable_issued = false;
		return false;
	}

	/* Port disable has been queued. We cannot block for callback.
	   Wait for callback in the main loop.
	 */
	return true;
}

int main ()
{
	/* Init tasks here. */
	...;

	while (1)
	{
		/* Call regular task handlers (DPM, APP, HPI) here. */
		...;

		if ((pd_disable_issued) && (pd_disable_completed))
		{
			/* Port is now disabled. */
			...;
			pd_disable_issued = false;
		}
	}
}
```

6.3.6.3 Sending a DISCOVER_ID VDM

The dpm_pd_command() API should be used to send VDMs and other PD commands to the port partner. The send operation is non-blocking and the completion callback will notify that the operation is complete. Note that the main loop should continue to run for proper completion of the VDM operation.

Code Listing 10

```
static volatile bool    abort_cmd = false;
static dpm_pd_cmd_buf_t cmd_buf;

static void pd_command_cb(uint8_t port, resp_status_t resp,
        const pd_packet_t *vdm_ptr)
{
      uint32_t response;
      if (status == RES_RCVD)
      {
            /* Response received. Check handshake. */
            response = vdm_ptr->dat[0].std_vdm_hdr.cmd_type;
            switch (response)
            {
                  case CMD_TYPE_RESP_ACK:
                        /* ACK received. */
                        ...;
                        break;
                  case CMD_TYPE_RESP_BUSY:
                        /* BUSY received. */
                        ...;
                        break;
                  case CMD_TYPE_RESP_NAK:
                        /* NACK received. */
                        ...;
                        break;
            }
            /* Next operation can be started from here. */
      }
}

bool send_discover_id(uint8_t port)
{
      /* Store state of operation. */
```

Code Listing 10

```
        pd_command_issued    = true;
        pd_command_completed = false;

        /* Format the command parameters.
           Single DO with standard Discover_ID command to SOP controller.
           Timeout is set to 100 ms.
         */
        cmd_buf.cmd_sop      = SOP;
        cmd_buf.cmd_do[0]    = 0xFF008001;
        cmd_buf.no_of_cmd_do = 1;
        cmd_buf.timeout      = 100;

        /* Initiate the command. Keep trying until accepted. */
        while (dpm_pd_command(port, DPM_CMD_SEND_VDM,
 &cmd_buf, pd_command_cb) != CCG_STAT_SUCCESS)
        {
                /* Can implement a timeout/abort here. */
                if (abort_cmd)
                        return false;
        }
        /* Command has been queued. We cannot block for callback here. */
        return true;
}
```

6.3.6.4 Getting current PD port status

The Device Policy Manager interface layer in the CCGx PD stack maintains a status data structure that provides complete status information about the USB PD port.

This structure can be retrieved using the dpm_get_info() API. The API returns a const pointer to a dpm_status_t structure which includes the following status fields:

1. **attach**: Specifies whether the port is currently attached.
2. **cur_port_role**: Specifies whether the port is currently a Source or a Sink
3. **cur_port_type**: Specifies whether the port is currently a DFP or an UFP
4. **polarity**: Specifies the Type-C connection polarity (CC1 or CC2 being used)
5. **contract_exist**: Specifies whether a PD contract exists
6. **contract**: Specifies the current PD contract (voltage and current) information.
7. **emca_present**: Specifies whether CCGx as DFP has detected a cable marker
8. **src_sel_pdo**: Specifies the PDO that CCGx as source used to establish contract
9. **snk_sel_pdo**: Specifies the Source Cap that CCGx as sink accepted to establish contract

10. **src_rdo**: Specifies the RDO that CCGx received for PD contract
11. **snk_rdo**: Specifies the RDO that CCGx as Sink sent for PD contract.

6.3.6.5 Issue a DR_SWAP where required

CCGx in a Notebook Port controller application is expected to function as a DFP. You can check the current port type of the CCGx device and initiate a DR_SWAP if CCGx is a UFP, so that we can ensure that the supported alternate modes can be enabled.

The current port type is checked as described in the section Getting current PD port status and the dpm_pd_command() API can be used to initiate a DR_SWAP.

Code Listing 11

```
/* Function to initiate a DR_SWAP if CCGx is UFP. */
void dr_swap_if_required()
{
const dpm_status_t *dpm_stat = dpm_get_info (0);
dpm_pd_cmd_buf_t param;
ccg_status_t status;

if (dpm_stat->cur_port_type == PRT_TYPE_UFP)
{
/* CCGx is UFP. Initiate DR_SWAP.
   We keep retrying while the PD port is in a busy state. */
      param.cmd_sop = SOP;
      do {
            status = dpm_pd_command (0, DPM_CMD_SEND_DR_SWAP,
                  &param, cmd_callback);
      } while (status == CCG_STAT_BUSY);

}
}
```

6.3.6.6 Change the source capabilities

The dpm_update_src_cap() and dpm_update_src_cap_mask() APIs can be used to update the source capabilities supported by CCGx.

At any time, CCGx can support a set of maximum seven source capabilities. These seven capabilities are maintained in the form of a the cur_src_pdo array in the dpm_status_t structure. A subset of these PDOs can be enabled at runtime using a PDO enable bit mask setting. The current PDO enable mask value can be read from the src_pdo_mask field of the dpm_status_t structure.

The PDO enable mask can be changed using the dpm_update_src_cap_mask() API.

Firmware APIs

The set of PDOs can be changed using the dpm_update_src_cap() API. The PDO enable mask will also need to be updated after updating the set of PDOs.

Code Listing 12

```
/* Function to configure and enable a desired source PDO. */
void select_source_pdo(pd_do_t new_pdo)
{
	const dpm_status_t *dpm_stat = dpm_get_info (0);
	uint8_t index;
	bool pdo_found = false;

	/* See if the new_pdo is already part of the list. */
	for (index = 0; index < dpm_stat->src_pdo_count; index++)
	{
		if (dpm_stat->src_pdo[index].val == new_pdo.val)
		{
			pdo_found = true;
			break;
		}
	}

	if (pdo_found)
	{
		/* PDO found. Just enable it. */
		dpm_update_src_cap_mask(0,
(dpm_stat->src_pdo_mask | (1 << index)));
}
else
	{
/* PDO not found, update the PDO list and enable it.
   Note: For this example, we are replacing the complete list
   with a single PDO. This needs be updated to retain the
   other required PDOs. */
		dpm_update_src_cap(0, 1, &new_pdo);
		dpm_update_src_cap_mask(0, 1);
}
}
```

See the Alternate Mode module source for more examples of using the DPM APIs.

6.3.7 Solution level examples

6.3.7.1 PD event handling

The PD events raised by the stack are handled at the solution level in the sln_pd_event_handler() function. In the normal case where policy decisions are handled through the EC, it is sufficient to pass the events onto the EC through the HPI interface. See below for a sample implementation of the event handler.

Code Listing 13

```
/* Solution PD event handler */
void sln_pd_event_handler(uint8_t port, app_evt_t evt, const void *data)
{
    /* Pass the event onto the EC through HPI. */
    hpi_pd_event_handler(port, evt, data);
}
```

6.3.7.2 Application callback registration

The application callbacks that handle various operations requested by the PD stack are registered through a structure that contains pointers to all the functions. The callbacks are registered using the app_get_callback_ptr() function. A sample implementation of this function is shown below.

Code Listing 14

```
/*
 * Application callback functions for the DPM. Since this application
 * uses the functions provided by the stack, loading with the stack
defaults.
 */
const app_cbk_t app_callback =
{
    app_event_handler,    /* Event handler. */
    psrc_set_voltage,     /* Source voltage update function. */
    psrc_set_current,     /* Source current update function. */
    psrc_enable,          /* Enable source FET. */
    psrc_disable,         /* Disable source FET. */
    vconn_enable,         /* Enable VConn supply. */
    vconn_disable,        /* Disable VConn supply. */
    vconn_is_present,     /* Check if VConn is present. */
    vbus_is_present,      /* Check if VBus is in the expected range. */
    vbus_discharge_on,    /* Enable VBus discharge path. */
    vbus_discharge_off,   /* Disable VBus discharge path. */
```

Code Listing 14

```
    psnk_set_voltage,     /* Set sink voltage. */
    psnk_set_current,     /* Set sink current. */
    psnk_enable,          /* Enable sink FET. */
    psnk_disable,         /* Disable sink FET. */
    eval_src_cap,         /* Evaluate source power capabilities. */
    eval_rdo,             /* Evaluate partner power request. */
    eval_dr_swap,         /* Evaluate DR_SWAP command. */
    eval_pr_swap,         /* Evaluate PR_SWAP command. */
    eval_vconn_swap,      /* Evaluate VCONN_SWAP command. */
    eval_vdm,             /* Evaluate received VDM. */
    eval_fr_swap,                /* Evaluate received FR swap */
    vbus_get_value,             /* Retrieve the VBUS voltage */
    psrc_get_voltage,           /* Get the current voltage level enabled by
the power source. */
    update_rdo                   /* Allow solution layer to make changes
to the RDO. */
};
app_cbk_t* app_get_callback_ptr(uint8_t port)
{
    /* Solution callback pointer is same for all ports */
    (void)port;
    return ((app_cbk_t *)(&app_callback));
}
```

6.3.7.3 Change the source PDO selection logic

The eval_src_cap() callback function is invoked by the PD stack on receiving source capabilities message from the Source. The default implementation of the same is available in ***src/app/pdo.c***. This function can be overridden during application callback registration (section Application callback registration).

The eval_src_cap() and is_src_acceptable_snk() functions in ***src/app/pdo.c*** can be used as a template and a custom function can be implemented in the solution.

For example, If an additional check needs to be done for maximum current support in the source PDO, then this can be done by changing the eval_src_cap() callback function to my_eval_src_cap().

Code Listing 15

```
/* Custom function to check if the source PDO is acceptable or not. */
void my_is_src_acceptable_snk(uint8_t port, pd_do_t* pdo_src, uint8_t
snk_pdo_idx)
{
```

Code Listing 15

```
    ...
    case PDO_FIXED_SUPPLY:
        if(fix_volt == pdo_snk->fixed_snk.voltage)
        {
            compare_temp = GET_MAX (max_min_temp, pdo_snk-
>fixed_snk.op_current);
            if (pdo_src->fixed_src.max_current >= compare_temp)
            {
                /* Added new check for absolute maximum current. */
                if (pdo_src->fixed_src.max_current <= MY_MAX_SNK_CURRENT)
                {
                    op_cur_power[port] = pdo_snk->fixed_snk.op_current;
                    out = true;
                }
            }
        }
        break;
    ...
}
/* Function to evaluate source PDO message. */
void my_eval_src_cap (uint8_t port, const pd_packet_t* src_cap,
app_resp_cbk_t
app_resp_handler)
{
    ...
    for(snk_pdo_index = 0u; snk_pdo_index < dpm->cur_snk_pdo_count;
snk_pdo_index++)
{
for(src_pdo_index = 0u; src_pdo_index < num_src_pdo; src_pdo_index++)
{
    if(my_is_src_acceptable_snk(port, (pd_do_t*)(&src_cap-
>dat[src_pdo_index]),
snk_pdo_index))
    {
                ...
    }
    ...
}
```

Code Listing 15

```
}
}
```

See the notebook project source files (main.c) for more examples of the solution level code.

6.3.8 Alternate mode handling

Support for USB PD alternate modes is a critical part of the CCG firmware functionality. Support for the DisplayPort alternate mode is pre-built into the Notebook and dongle applications. Users can add additional alternate mode support to the firmware. The procedure to add an additional alternate mode handler to the firmware includes two steps:

1. Implementing the handlers for the alternate modes. This includes code that will discover UFP capabilities and handle attention messages in a DFP role, and/or code that will receive and handle alternate mode requests as an UFP.
2. Registering the alternate mode handlers with the manager.

6.3.8.1 Implementing the alternate mode handlers

The CCG firmware stack provides a generic alternate mode manager which holds information about the supported alternate modes. This manager will invoke the handler functions specific to the alternate modes registered in the firmware application.

When CCG is a DFP, the alternate mode manager will discover whether the connected UFP supports any alternate modes for which handlers have been registered; and then call the associated handler functions.

When CCG is a UFP, the alternate mode manager will check whether incoming VDMs correspond to any registered alternate modes; and then call the associated handler functions.

6.3.8.1.1 Alternate mode data structure

The alt_mode_info_t structure serves as the interface between the alternate mode manager and the handlers for each alternate mode. An array of such structures is maintained by the alternate mode manager. This structure incorporates the fields shown in Table 45.

Table 45 List of alternate mode information structure members

Field	Type	Description
mode_state	enum alt_mode_state_t	State of the alternate mode. Can be one of DISABLED, IDLE, INIT, SEND_CMD, WAIT_FOR_RESP, FAIL, or EXIT.
sop_state[]	Array of enum alt_mode_state_t	Alternate mode state corresponding to each PD packet type (SOP, SOP' and SOP''). This is useful in cases where the alternate mode requires any EMCA cables to support the mode as well.

Field	Type	Description
vdo_max_numb	Unsigned char	Maximum number of VDOs that the alternate mode handler can handler.
obj_pos	uint8_t	Alternate mode object position of port partner.
cbl_obj_pos	uint8_t	Alternate mode object position of the cable.
alt_mode_id	Unsigned char	The alternate mode id for this mode.
vdm_header	pd_do_t	Holds the VDM header for the received messages.
vdo	Array of pd_do_t pointers	Pointers to buffers where alt. mode VDOs with various packet types should be stored. The storage is provided by the alternate mode handler and used by the manager.
vdo_numb	Array of uint8_t	Current number of VDOs used for processing in VDO buffers. Array of the maximum number of VDOs for each VDM message type: SOP, SOP', SOP"
cbk	Function pointer	Callback used by the alternate mode manager to invoke the specific alternate mode handler.
is_active	bool	Indicates if alternate mode is active
custom_att_obj_pos	bool	Object position field in Att VDM used by alt mode as custom object position
uvdm_supp	bool	Flag to indicate if alt mode support unstructured VDMs.
set_mux_isolate	bool	Flag to indicate if MUX should be set to safe state while ENTER/EXIT alt mode is being processed.

6.3.8.1.2 Alternate mode handling flow

The alternate mode handler uses the mode state to communicate (receive/send VDM) with the alternate mode manager.

- IDLE state is responsible for received VDM processing;
- WAIT_FOR_RESP state is responsible for received VDM response processing;
- FAIL state is responsible for the failed VDM processing;

- INIT state is responsible for initialization of the alt mode (DFP only);
- EXIT state is responsible for de-initialization/exit of the alt mode (DFP only);
- SEND state should be set by alt mode to inform the alt modes manager that a VDM packet is ready and should be sent to the port partner.

The alternate mode handler can use internal states to process the received VDM or VDM responses. These states are not used by the alternate mode manager.

DFP handling

When the alternate mode is DFP when the main DFP state machine operates with five states:

1. The INIT state is used to initiate alternate mode handling. This state uses in two cases:
   a) When DFP alternate mode registrations are successful and need to initiate alternate mode handling.
   b) When the Alt modes manager initiates asynchronous entry of the alternate mode.
   c) During initialization next steps should be done:

      Set sop_state array variables as ALT_MODE_STATE_SEND in the dependence if SOP/SOP'/SOP" VDMs should be send while entering the mode

      Save pointers to the VDO buffer in the info structure

      Assign enter mode VDOs if needed

      Save the alt mode DFP state machine function in the info structure

      Save the App command handler function in the info structure

      Set alternate mode state as enter

      Additional initialization code as required by the alternate mode.
2. IDLE state is used to analyze received VDM related to the alt mode. When a VDM is received, the following steps should be completed:
   a) Get the received command from the VDM header
   b) Run custom analysis of the received VDM
   c) If a response VDM should be sent after analysis, then change the internal alternate mode state in compliance with the specific command number, fill the VDO buffer and set alternate mode state to the SEND state.
3. WAIT_FOR_RESP state is used to process/analyze received VDM response. When VDM response is received, next steps should be done:
   a) Get internal alternate mode depending on command from VDM header
   b) Set alternate mode state to IDLE
   c) Analyze the received VDM response
   d) If another VDM should be sent based on the received response, then change the internal alternate mode state in compliance with the specific command number, fill the VDO buffer and set alternate mode state to the SEND state.
4. FAIL state is used to make decision when sent VDM was failed (e.g., NAKed response, Good CRC was not received, etc.)
5. EXIT state is used when the alternate mode manager initiates asynchronous exit of the alternate mode. In this case, alternate mode should send exit mode command to the port partner.

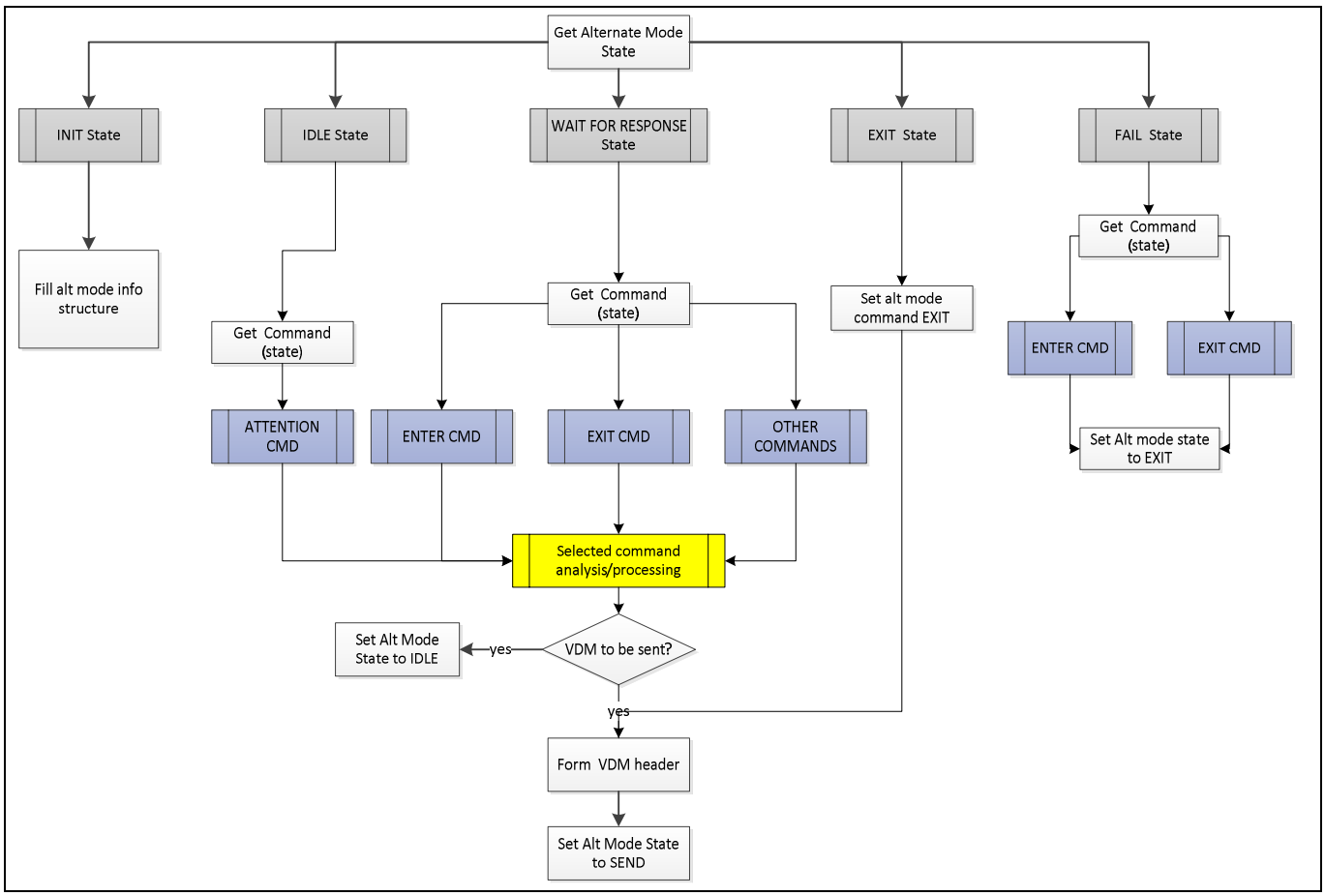


Figure 47 Alternate mode handler state machine for DFP

UFP handling

Alternate mode handling when CCG is UFP operates in one of two states:

1. IDLE state is used to analyze received VDMs related to the alternate mode. When a VDM is received, the following should be done:
   a) Analyze the VDM based on the alternate mode rules.
   b) If the VDM received is valid and an ACK response is to be sent, then set the alternate mode state to IDLE.
   c) If an alternate mode-specific response is to be returned, fill the vdo array with data to be sent and set the alternate mode state to SEND.
2. WAIT_FOR_RESP state is used to process/analyze received VDM responses. When a VDM response is received, the following steps should be performed:
   a) Get internal alternate mode depending on the command from the VDM header
   b) Set alternate mode state to IDLE
   c) Analyze the response received
   d) If another VDM should be sent after analysis (e.g. attention), then change the internal alternate mode state in compliance with the specific command number, and fill the VDO buffer and alternate mode state to the SEND state.

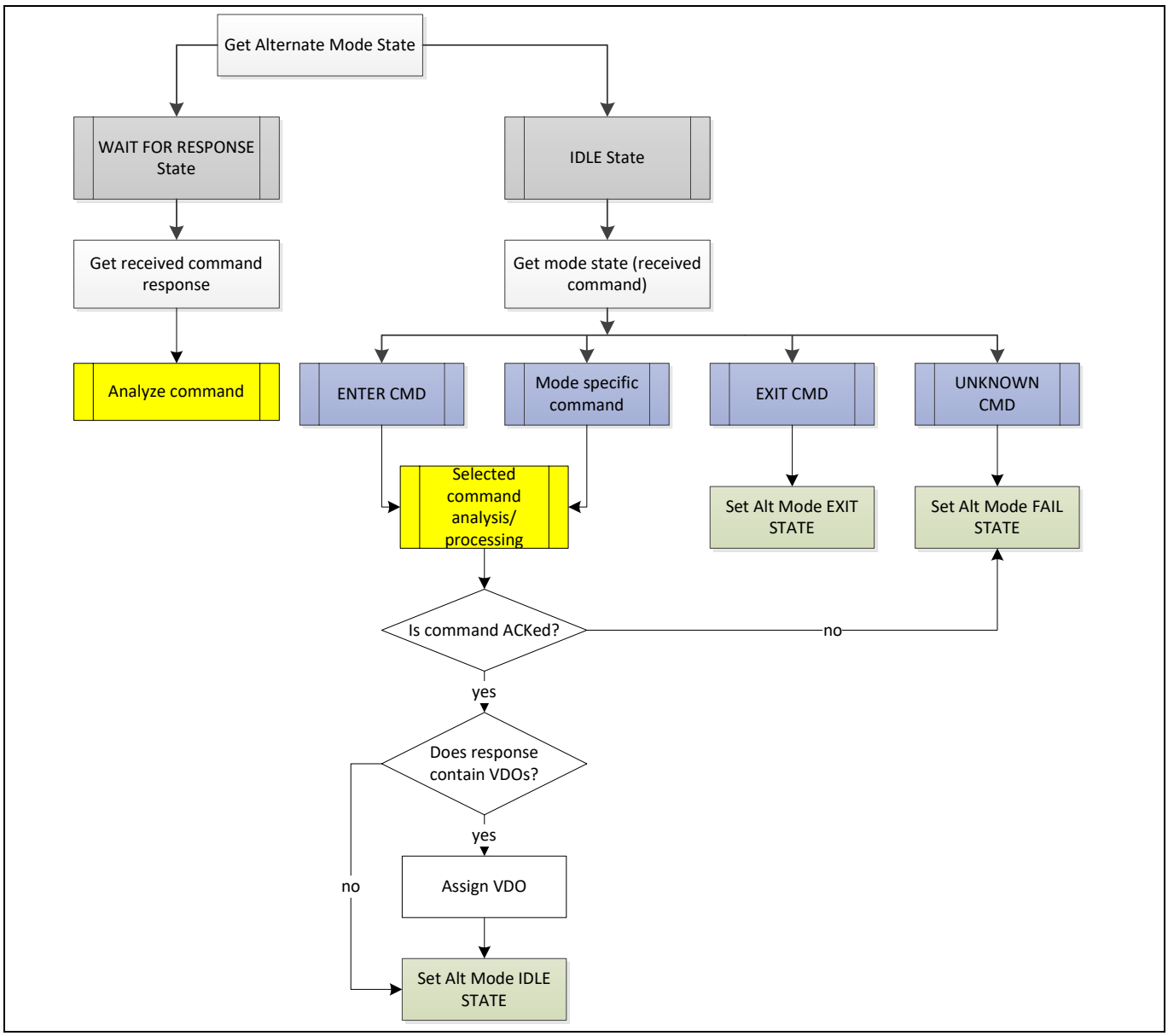


Figure 48 Alternate mode handler state machine for UFP

6.3.8.2 Alternate modes configuration

Any project that uses at least one Alternate mode should have:

- An ***alt_modes_config.h*** file that holds the info to register the Alternate mode in the Alternate mode's manager.
- ***Base alternate modes*** section in the configuration table. This section holds the information about Alternate modes (SVID, priority (entry queue) of Alternate modes, compatibility with other Alternate modes, DFP/UFP Alternate modes enable/disable masks) are supported by CCG.

6.3.8.2.1 alt_modes_config.h file structure

The table below (example) represents the supported Alternate modes:

Code Listing 16

```
#if ((DFP_ALT_MODE_SUPP) || (UFP_ALT_MODE_SUPP))
reg_am_t reg_alt_mode [] =
{
#if ((DP_DFP_SUPP) || (DP_UFP_SUPP))
{DP_SVID, reg_dp_modes},
#endif /* ((DP_DFP_SUPP) || (DP_UFP_SUPP)) */

#if ((TBT_DFP_SUPP) || (TBT_UFP_SUPP))
{INTEL_VID, reg_intel_modes},
#endif /* ((TBT_DFP_SUPP) || (TBT_UFP_SUPP)) */
};
```

Structures about the hold info about Alternate mode SVID and the pointer to the function to register Alternate mode in the Alternate mode's manager.

6.3.8.2.2 Alternate modes configuration

To configure alternate modes, some configurations should be provided in the configuration (see Table 3).

To add an alternate mode in the configuration table – go to Base Alternate Modes section.

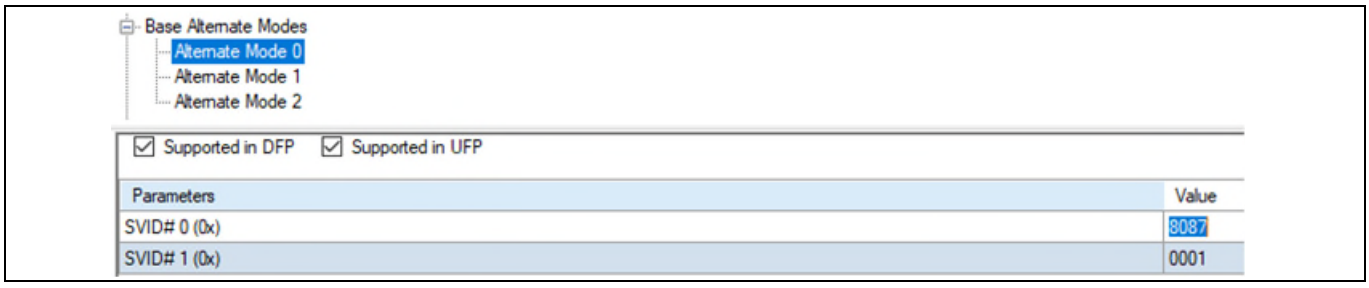


Figure 49 Base Alternate Modes settings example

Forlease G6SFenabled and VDM state machines re-started after a delay, if retries are enabled.nd enumerate any USB devices while the example in Figure 49, there the list of 3 alternate modes Alternate mode 0, Alternate mode 1, and Alternate mode 2. These 3 alternate modes correspond to 3 different alternate modes which already added in the FW.

SVID#0 in configuration corresponds to base alternate mode SVID.

Supported in DFP and Supported in UFP checkboxes are used to enable SVID#0 alternate mode processing while CCG is DFP and UFP data role.

If SVID#0 alternate mode could be run with simultaneously with another alternate modes (SVID) SVID#1, SVID#2…SVID#7 could be added. In case if SVID#0 alternate mode could not run with another alternate mode fill just SVID#0 field.

Note: *Any alternate modes added to the config table should have appropriate handler files in the firmware sources.*

Revision history

Document revision	Date	Description of changes
**	2018-06-29	Initial release
*A	2018-12-20	Updates for CCGx Host SDK Revision 3.3.0 Section 1.2: Updated list of Type-C controller families. Section 1.3: Updated SDK feature list. Section 1.3.1: Updated directory structure and description. Section 3.1: Added instruction for updating PSoC™ Creator device packs. Section 3.2: Updated list of reference projects. Section 3.3: Added complete list of configuration parameters. Section 4.5: Added section on CYPD6125-40LQXI_notebook reference project. Section 4.6: Added section on CYPD5126-40LQXI_notebook reference project.
*B	2019-04-29	Updates for CCGx Host SDK Revision 3.3.1 Section 1.2: Updated list of Type-C controller families. Section 1.3: Updated SDK feature list. Section 1.3.1: Updated directory structure and description. Section 3.2.2.1: Added new section describing the process of compiling the projects with MDK toolchain. Section 4.5: Added section on CYPD6126-96BZXI_notebook reference project.
*C	2019-06-12	Section 2.2.1: Updated version of PSoC™ Creator required by the SDK. Section 3.1: Removed section describing procedure to obtain new device updates. Section 3.2: Removed note stating that CCG3 firmware is in maintenance mode as these projects have been updated as part of this release. Section 3.2.2: Added new steps describing how to resolve build errors due to missing PSoC™ Creator components.
*D	2020-07-10	Updates for CCGx Host SDK Revision 3.4 Section 1.3: Updated SDK directory structure. Section 2.2: Updated list of tool dependencies. Section 4.5: Added section on CCG6DF and CCG6SF projects. Section 5.4: Added flash map information for CCG6DF and CCG6SF. Section 5.5: Added bootloader flow for CCG6DF and CCG6SF Section 5.6: Updated firmware flow to show newly added tasks.
*E	2022-02-21	Updates for CCGx Host SDK Revision 3.5 Section 1.3.1: Updated SDK directory structure Section 2.2: Updated tool dependencies Section 3.1.1: Environment updates Section 3.2: Added new projects CCG6xF R1 & ACG1F

Revision history

Document revision	Date	Description of changes
		Section 3.2.1: Updated Figure 3-2 & Figure 3-3 with Host SDK 3.5 Section 4.5, 4.6: Added CYPD6227 application details Section 4.15: Added ACG1 application details. Section 5.4: Added ACG1F Flash Memory Map
*F	2022-08-26	Updates for CCGx Host SDK Revision 3.5.1 Section 3.2.3: Updated Figure 3-14 with v3.5.1 build details.
*G	2022-09-22	Migrated to Infineon template.
*H	2023-03-15	Updates for CCGx Host SDK Revision 3.6 Removed USB Type-C TBT Alt mode spec from the reference documents section. Because it is now part of the Type-C spec Section 1.3: Updated with Raptor Lake and Phoenix details & USB specifications Section 1.3.1: Updated the projects with RPL and PHX Section 3.1.1: Added PSoC™ Creator build details. Section 3.2: Updated the reference project to CCG6DF Section 3.2.2.2: Updated with CCG6DF project details Section 3.2.3: Updated with CCG6DF project details Section 3.3.1: Table 3 updated UFP and DFP VDO version as per the latest stack Section 4.2: Added "CY_PD_RW_PD_RESP_MSG_DATA" as compile time options Minor edits throughout the document to align with the latest template

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2023-03-15
Published by

Infineon Technologies AG
81726 Munich, Germany

© 2023 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?
Email: erratum@infineon.com

Document reference
002-24327 Rev. *H

Important notice
The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie")

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.