**Please note that Cypress is an Infineon Technologies Company.**
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

**Continuity of document content**
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

**Continuity of ordering part numbers**
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CY8CPROTO-064B0S3

# PSoC 64 “Secure Boot” Prototyping Kit Guide

Doc. # 002-29505 Rev. *B

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com

## Copyrights

© Cypress Semiconductor Corporation, 2020–2021. This document is the property of Cypress Semiconductor Corporation and its subsidiaries (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress shall have no liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. CYPRESS DOES NOT REPRESENT, WARRANT, OR GUARANTEE THAT CYPRESS PRODUCTS, OR SYSTEMS CREATED USING CYPRESS PRODUCTS, WILL BE FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (collectively, “Security Breach”). Cypress disclaims any liability relating to any Security Breach, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any Security Breach. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. “High-Risk Device” means any device or system whose failure could cause personal injury, death, or property damage. Examples of High-Risk Devices are weapons, nuclear installations, surgical implants, and other medical devices. “Critical Component” means any component of a High-Risk Device whose failure to perform can be reasonably expected to cause, directly or indirectly, the failure of the High-Risk Device, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any use of a Cypress product as a Critical Component in a High-Risk Device. You shall indemnify and hold Cypress, its directors, officers, employees, agents, affiliates, distributors, and assigns harmless from and against all claims, costs, damages, and expenses, arising out of any claim, including claims for product liability, personal injury or death, or property damage arising from any use of a Cypress product as a Critical Component in a High-Risk Device. Cypress products are not intended or authorized for use as a Critical Component in any High-Risk Device except to the limited extent that (i) Cypress’s published data sheet for the product explicitly states Cypress has qualified the product for use in a specific High-Risk Device, or (ii) Cypress has given you advance written authorization to use the product as a Critical Component in the specific High-Risk Device and you have signed a separate indemnification agreement.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

# Contents


**Safety and Regulatory Compliance Information** 5

**1. Introduction** 6

1.1 Kit Contents ..... 6
1.2 Getting Started ..... 7
1.3 Code Examples ..... 7
1.4 Board Details ..... 7
1.5 Additional Learning Resources ..... 10
1.6 Technical Support ..... 10
1.7 Documentation Conventions ..... 11
1.8 Acronyms ..... 11

**2. Software Installation** 13

2.1 Before You Begin ..... 13
2.2 Install Software ..... 13

**3. Kit Operation** 15

3.1 Theory of Operation ..... 15
3.2 KitProg3 ..... 19
3.2.1 Programming and Debugging ..... 19
3.2.2 USB-UART Bridge ..... 19
3.2.3 USB-I2C Bridge ..... 20

**4. Running Code on PSoC 64 “Secure Boot” MCUs** 21

4.1 Provisioning Overview ..... 21
4.2 Create ModusToolbox example application ..... 23
4.3 Provision the Device ..... 25
4.4 Build and Program the Example Application ..... 29
4.5 Additional Code Examples ..... 31

**5. Hardware** 32

5.1 Schematics ..... 32
5.2 Hardware Functional Description ..... 32
5.2.1 CYB06445LQI-S3D42 (U1) ..... 32
5.2.2 PSoC 5LP based KitProg3 (U2) ..... 33
5.2.3 Serial Interconnection between PSoC 5LP and PSoC 64 Device ..... 34
5.2.4 Power Supply System ..... 35
5.2.5 Expansion Connectors ..... 37
5.2.6 Quad SPI Flash ..... 40
5.2.7 LEDs ..... 40
5.2.8 User Buttons ..... 41

5.2.9 System Crystals ....42
5.2.10 10-pin SWD/JTAG Programming Header ....42
5.3 Bill of Materials ....42
5.4 Frequently Asked Questions....43

**Revision History** **44**

# Safety and Regulatory Compliance Information

The CY8CPROTO-064B0S3 PSoC® 64 “Secure Boot” Prototyping Kit is intended for use as a development platform for hardware or software in a laboratory environment. The board is an open-system design, which does not include a shielded enclosure. Due to this reason, the board may cause interference to other electrical or electronic devices in close proximity. In a domestic environment, this product may cause radio interference. In such cases, take adequate preventive measures. Also, do not use this board near any medical equipment or RF devices. Users are advised to test and evaluate this kit in an RF development environment.

This kit is not a finished product and when assembled may not be resold or otherwise marketed unless all required authorizations are first obtained. Contact support@cypress.com for details.



PSoC 64 “Secure Boot” Prototyping Boards contain electrostatic discharge (ESD) sensitive devices. Electrostatic charges readily accumulate on the human body and any equipment, which can cause a discharge without detection. Permanent damage may occur on devices subjected to high-energy discharges. Proper ESD precautions are recommended to avoid performance degradation or loss of functionality. Store unused PSoC 64 “Secure Boot” Prototyping Boards in the protective shipping package.



End-of-Life/Product Recycling

The end-of-life cycle for this kit is five years from the date of manufacture mentioned on the back of the box. Contact your nearest recycler to discard the kit.

## General Safety Instructions

### ESD Protection

ESD can damage boards and associated components. Cypress recommends that you perform procedures only at an ESD workstation. If an ESD workstation is unavailable, use appropriate ESD protection by wearing an anti-static wrist strap attached to a grounded metal object.

### Handling Boards

The CY8CPROTO-064B0S3 PSoC 64 “Secure Boot” Prototyping Kit is sensitive to ESD. Hold the board only by its edges. After removing the board from its box, place it on a grounded, static-free surface. Use a conductive foam pad, if available. Do not slide the board over any surface.

# 1. Introduction

Thank you for your interest in the CY8CPROTO-064B0S3 PSoC 64 “Secure Boot” Prototyping Kit. The PSoC 64 “Secure Boot” Prototyping Kit enables you to evaluate and develop your applications using the PSoC 64 Line of Secured MCUs (hereafter called “PSoC 64”).

PSoC 64 is a high-performance, ultra-low-power and secure MCU platform, purpose-built for IoT applications. The PSoC 64 based on the PSoC 6 MCU platform, features out-of-the-box security functionality. This line provides an isolated root of trust (RoT) with true attestation and provisioning services. In addition, these MCUs deliver a pre-configured secure execution environment which supports system software of various IoT platforms and provides secure provisioning, key storage, and firmware management.

The CY8CPROTO-064B0S3 “Secure Boot” Prototyping Kit carries a PSoC 64 device (CYB06445LQI-S3D42). In addition, the board features an onboard programmer/debugger (KitProg3), a 512-Mbit Semper™ NOR flash (S25HL512T), a micro-B connector for USB device interface, two user LEDs and one user push button. The board supports operating voltages of 1.8 V, 2.5 V and 3.3 V.

You can use ModusToolbox® software to develop and debug your PSoC 64 projects. ModusToolbox software is a set of tools that enable you to integrate Cypress devices into your existing development methodology.

If you are new to PSoC 6 MCU and ModusToolbox software, refer to the application note AN228571 - Getting Started with PSoC 6 MCU on ModusToolbox to help you familiarize with the PSoC 6 MCU and help you create your own design using the ModusToolbox software.

## 1.1 Kit Contents

The CY8CPROTO-064B0S3 package has the following contents.

- 3 × PSoC 64 “Secure Boot” Prototyping Board
- USB Type-A to Micro-B cable
- Quick Start Guide (printed on the kit package)

This kit contains 3 identical boards which can be provisioned with keys and policies for evaluation. Once the kit is provisioned with a specific set of keys/policies, it will be bound to them unless re-provisioned. The 3 boards give you the flexibility to experiment with different policies or keys. Inspect the contents of the kit; if you find any part missing, contact your nearest Cypress sales office for help: www.cypress.com/support.

## 1.2 Getting Started

This guide will help you to get acquainted with the CY8CPROTO-064B0S3 “Secure Boot” Prototyping Kit:

- The Software Installation chapter on page 13 describes the installation of the kit software. This includes ModusToolbox software which will be used to develop, program and debug applications on to the device.
- The Kit Operation chapter on page 15 describes the major features of the kit and functionalities such as programming, debugging, and the USB-UART and USB-I2C bridges.
- The Running Code on PSoC 64 “Secure Boot” MCUs chapter on page 21 describes multiple PSoC 64 code examples that will help you understand how to create your own PSoC 64 applications.
- The Hardware chapter on page 32 provides a detailed hardware description, methods to use the onboard NOR Flash, kit schematics, and the bill of materials (BOM).

## 1.3 Code Examples

Code examples for ModusToolbox software can be found at https://github.com/cypresssemiconductorco/Code-Examples-for-ModusToolbox-Software.

## 1.4 Board Details

The PSoC 64 “Secure Boot” Prototyping Board has the following features:

- CY8CPROTO-064B0S3 board contains a PSoC 64 device
- 512-Mbit external Quad SPI Semper NOR flash that provides a fast, expandable memory for data and code
- KitProg3 onboard SWD programmer/debugger, USB-UART and USB-I2C bridge functionality via a Micro-B connector
- A second Micro-B connector for USB device interface
- 1.8 V, 2.5 V and 3.3 V operation is supported
- Two user LEDs, a user button and a reset button
- One Mode button, one Status LED and one Power LED for the KitProg3

Figure 1-1 shows the pinout of the PSoC 64 “Secure Boot” Prototyping Board.

Figure 1-1. Prototyping Board Pinout

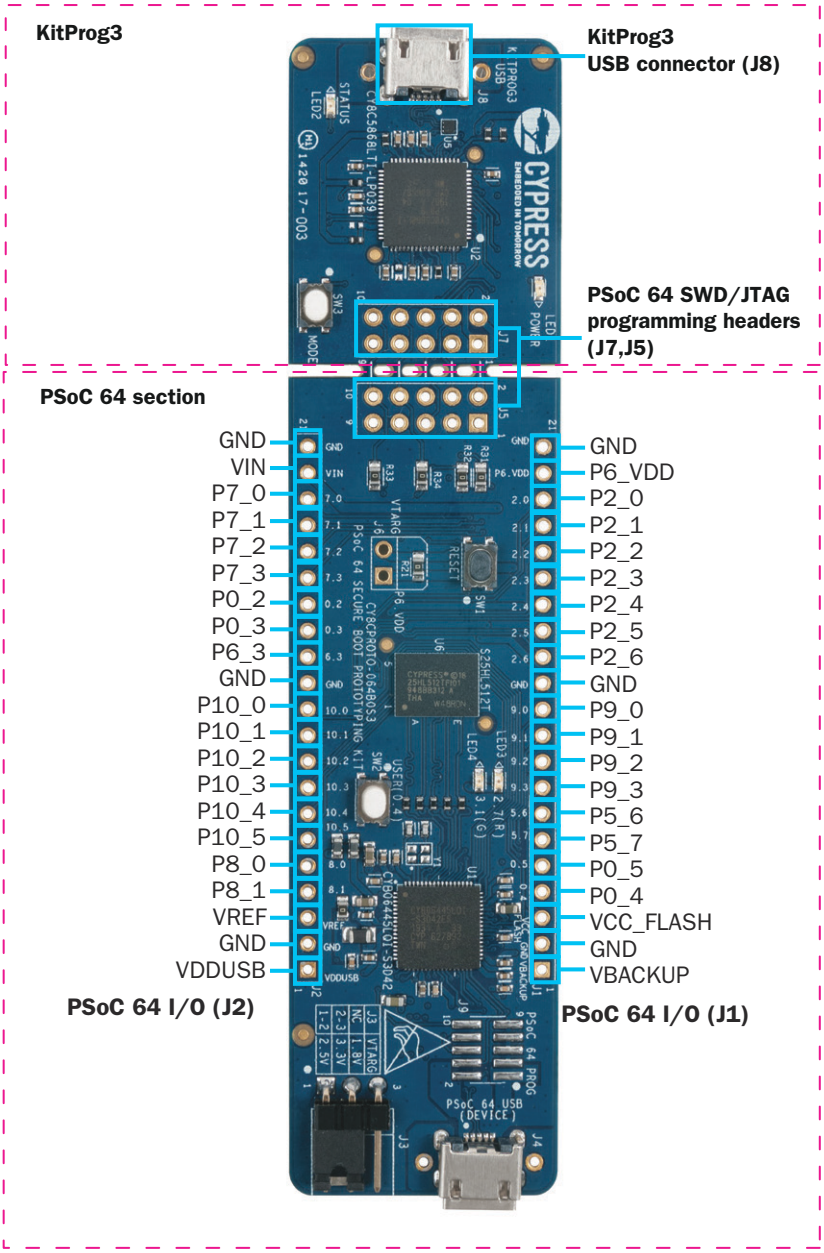


Table 1-1. Board Pinout

| PSoC 64 Pin | Header Mapping | Primary On-board Function | Secondary On-board Function | Connection details |
|---|---|---|---|---|
| P0[0] | – | WCO_IN | – | – |
| P0[1] | – | WCO_OUT | – | – |
| P0[2] | J2.15 | GPIO | – | – |
| P0[3] | J2.14 | GPIO | – | – |
| P0[4] | J1.4 | User Button (SW2) with Hibernate wakeup capability | GPIO | Connected to ground as active LOW logic by default. |
| P0[5] | J1.5 | GPIO | – | – |
| P2[0] | J1.19 | GPIO | – | – |
| P2[1] | J1.18 | GPIO | – | – |
| P2[2] | J1.17 | GPIO | – | – |
| P2[3] | J1.16 | GPIO | – | – |
| P2[4] | J1.15 | GPIO | – | – |
| P2[5] | J1.14 | GPIO | – | – |
| P2[6] | J1.13 | GPIO | – | – |
| P2[7] | – | Red User LED (LED3) | GPIO | Connected in active LOW configuration. |
| P3[1] | – | Green User LED (LED4) | GPIO | Connected in active LOW configuration. |
| P5[0] | – | UART RX | GPIO | Remove R33 to disconnect from KitProg3 UART TX. |
| P5[1] | – | UART TX | GPIO | Remove R34 to disconnect from KitProg3 UART RX. |
| P5[6] | J1.7 | GPIO | – | – |
| P5[7] | J1.6 | GPIO | – | – |
| P6[3] | J2.13 | GPIO | – | – |
| P6[4] | J5.4 | I2C_SCL | TDO_SWO | Remove R31 to disconnect from KitProg3 I2C_SCL, load R62 to connect to TDO_SWO. |
| P6[5] | J5.6 | I2C_SDA | TDI | Remove R32 to disconnect from KitProg3 I2C_SDA, load R63 to connect to TDI. |
| P6[6] | J5.9 | SWDIO | GPIO | – |
| P6[7] | J5.7 | SWDCLK | GPIO | – |
| P7[0] | J2.19 | GPIO | – | – |
| P7[1] | J2.18 | GPIO | – | – |
| P7[2] | J2.17 | GPIO | – | – |
| P7[3] | J2.16 | GPIO | – | – |
| P7[7] | – | CMOD | – | – |

Table 1-1. Board Pinout *(continued)*

| PSoC 64 Pin | Header Mapping | Primary On-board Function | Secondary On-board Function | Connection details |
|---|---|---|---|---|
| P8[0] | J2.5 | GPIO | – | – |
| P8[1] | J2.4 | GPIO | – | – |
| P10[0] | J2.11 | GPIO | – | – |
| P10[1] | J2.10 | GPIO | – | – |
| P10[2] | J2.9 | GPIO | – | – |
| P10[3] | J2.8 | GPIO | – | – |
| P10[4] | J2.7 | GPIO | – | – |
| P10[5] | J2.6 | GPIO | – | – |
| P11[0] | – | QSPI Flash Reset | – | – |
| P11[1] | – | QSPI Flash Int | – | – |
| P11[2] | – | QSPI Flash CS | – | – |
| P11[3] | – | QSPI Flash DATA3 | – | – |
| P11[4] | – | QSPI Flash DATA2 | – | – |
| P11[5] | – | QSPI Flash DATA1 | – | – |
| P11[6] | – | QSPI Flash DATA0 | – | – |
| P11[7] | – | QSPI Flash CLK | – | – |
| P12[6] | – | ECO_IN | – | Load crystal at Y1 to use ECO. |
| P12[7] | – | ECO_OUT | – | Load crystal at Y1 to use ECO. |
| USB.DP | – | – | – | USB D+ |
| USB.DM | – | – | – | USB D– |

## 1.5 Additional Learning Resources

Cypress provides a wealth of data at www.cypress.com/psoc64 to help you to select the right PSoC device for your design and to help you to quickly and effectively integrate the device into your design.

## 1.6 Technical Support

For assistance, go to www.cypress.com/support. Visit community.cypress.com to ask your questions in Cypress developer community.

You can also use the following support resources if you need quick assistance:

- Self-help (Technical Documents)
- Local Sales Office Locations

## 1.7 Documentation Conventions

Table 1-2. Document Conventions for Guides

| Convention | Usage |
|---|---|
| `Courier New` | Displays file locations, user entered text, and source code:<br>`C:\...cd\icc\` |
| *Italics* | Displays file names and reference documentation:<br>Read about the *sourcefile.hex* file in the *PSoC Creator User Guide*. |
| [**Bracketed, Bold**] | Displays keyboard commands in procedures:<br>[**Enter**] or [**Ctrl**] [**C**] |
| File > Open | Represents menu paths:<br>File > Open > New Project |
| **Bold** | Displays commands, menu paths, and icon names in procedures:<br>Click the **File** icon and then click **Open**. |
| Times New Roman | Displays an equation:<br>2 + 2 = 4 |
| Text in gray boxes | Describes cautions or unique functionality of the product. |

## 1.8 Acronyms

Table 1-3. Acronyms Used in this Document

| Acronym | Definition |
|---|---|
| ADC | Analog-to-Digital Converter |
| AES | Advanced Encryption Standard |
| BOM | Bill of Materials |
| CPU | Central Processing Unit |
| CRC | Cyclic Redundancy Check |
| DAP | Debug Access Port |
| DC | Direct Current |
| Del-Sig | Delta-Sigma |
| DMA | Direct Memory Access |
| ECO | External Crystal Oscillator |
| ESD | Electrostatic Discharge |
| ETM | Embedded Trace Macrocell |
| GDB | GNU Debugger |
| GPIO | General-Purpose Input/Output |
| HID | Human Interface Device |
| HSM | Hardware Security Module |
| $I^2C$ | Inter-Integrated Circuit |
| IDE | Integrated Development Environment |
| IMO | Internal Main Oscillator |
| IoT | Internet of Things |
| JTAG | Joint Test Action Group |

Table 1-3. Acronyms Used in this Document *(continued)*

| Acronym | Definition |
|---|---|
| JWT | Java Web Token |
| LDO | Low Dropout Regulator |
| LED | Light-emitting Diode |
| MCU | Micro-Controller Unit |
| OEM | Original Equipment Manufacturer |
| PC | Personal Computer |
| PDL | Peripheral Driver Library |
| PSoC | Programmable System-on-Chip |
| PWM | Pulse Width Modulation |
| QSPI | Quad Serial Peripheral Interface |
| RoT | Root of Trust |
| RTOS | Real Time Operating System |
| SAR | Successive Approximation Register |
| SDK | Software Development Kit |
| SHA | Secure Hash Algorithm |
| SMIF | Serial Memory Interface |
| SPI | Serial Peripheral Interface |
| SRAM | Serial Random Access Memory |
| SWD | Serial Wire Debug |
| UART | Universal Asynchronous Receiver Transmitter |
| USB | Universal Serial Bus |
| WCO | Watch Crystal Oscillator |

# 2. Software Installation

This chapter describes the steps to install the software tools and packages on a PC for using the CY8CPROTO-064B0S3 PSoC 64 “Secure Boot” Prototyping Kit. This includes ModusToolbox software on which the applications will be built and used for programming. The detailed steps for installation on macOS and Linux can be found in the “Secure Boot” SDK user guide.

## 2.1 Before You Begin

To install Cypress software, you will require administrator privileges. However, they are not required to run the software once it has been installed. Before you install the kit software, close any other Cypress software that is currently running.

## 2.2 Install Software

Follow these steps to install the software:

1. Install the latest version of ModusToolbox software which should be version 2.3.1 or later.

> The installation of ModusToolbox 2.3.1 software or later, on a Windows PC provides all the tools required to build, program and provision devices. Windows users may skip the rest of this chapter.

2. Install Python 3.7 or later on your Linux or macOS computer if it isn’t already installed.
3. Set up the appropriate environment variable:

   **Linux:** Most distributions of Linux should already have python2 and python3 installed. To verify that python by default points to python3 run:

   ```
   python --version
   ```

   If python3 is not set as default run following commands. The number at the end of each command denotes a priority

   ```
   update-alternatives --install /usr/bin/python python /usr/bin/python2.7 1
   update-alternatives --install /usr/bin/python python /usr/bin/python3.7 2
   ```

**macOS:** By default, ‘python’ points to /usr/bin/python which is python2. To make 'python' and 'pip' resolve to python3 versions, execute the following:

```
echo 'alias python=python3' >> ~/.bash_profile
echo 'alias pip=pip3' >> ~/.bash_profile
source ~/.bash_profile
```

Make sure that you have the latest version of pip installed, use the following command.

```
python -m pip install --upgrade pip
```

To verify that ‘python’ and ‘pip’ by default point to python3, run:

```
python --version
Python 3.7.4
pip --version
pip 19.0.3 from /Library/Frameworks/Python.framework/Versions/3.7/lib/
python3.7/site-packages/pip (python 3.7)
```

4. Install the “Secure Boot” SDK package by running the following from a terminal/command prompt.

```
pip install -U cysecuretools
```

During installation, there may be error messages when installing colorama, protobuf and jsonschema. These can be safely ignored.

5. Install the libusb dependency for pyOCD. Please check README.md for the latest instructions on installing libusb. From the README.md file:

   How to install libusb depends on your OS:

   - macOS: use Homebrew: brew install libusb
   - Linux: should already be installed.

# 3. Kit Operation

This chapter introduces you to various features of the CY8CPROTO-064B0S3 PSoC 64 “Secure Boot” Prototyping Kit, including the theory of operation and the onboard KitProg3 programming and debugging functionality, USB-UART and USB-I2C bridges.

## 3.1 Theory of Operation

The CY8CPROTO-064B0S3 PSoC 64 “Secure Boot” Prototyping Kit is built around the PSoC 64 chip. Figure 3-1 shows the block diagram of the PSoC 64 device. For details of device features, see the device datasheet.

Figure 3-1. PSoC 64 Block Diagram

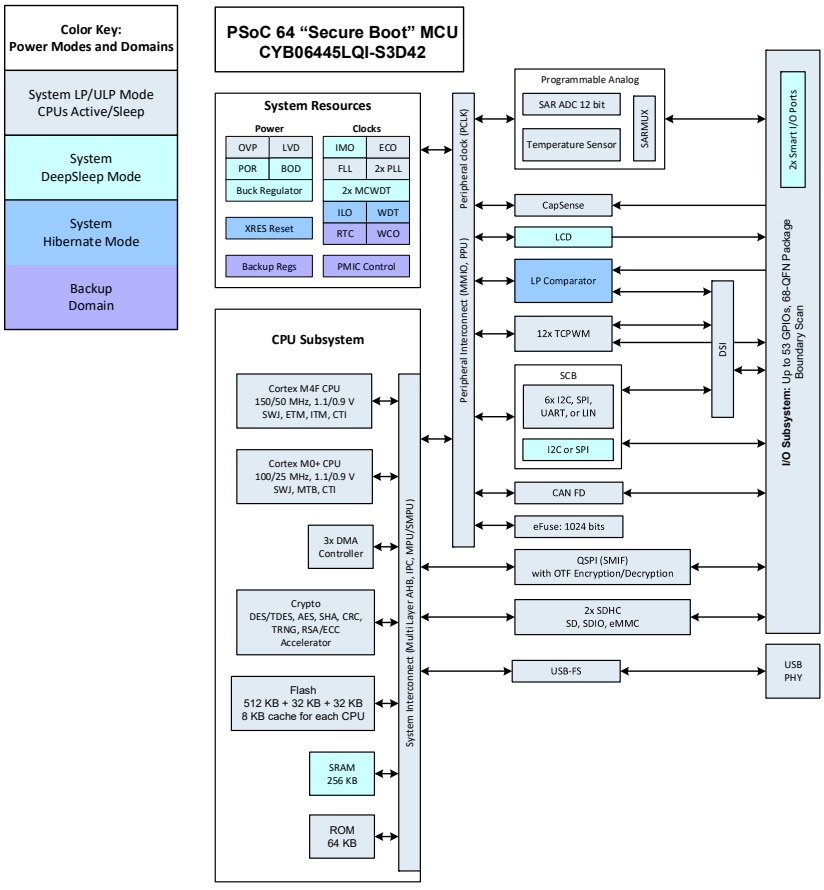
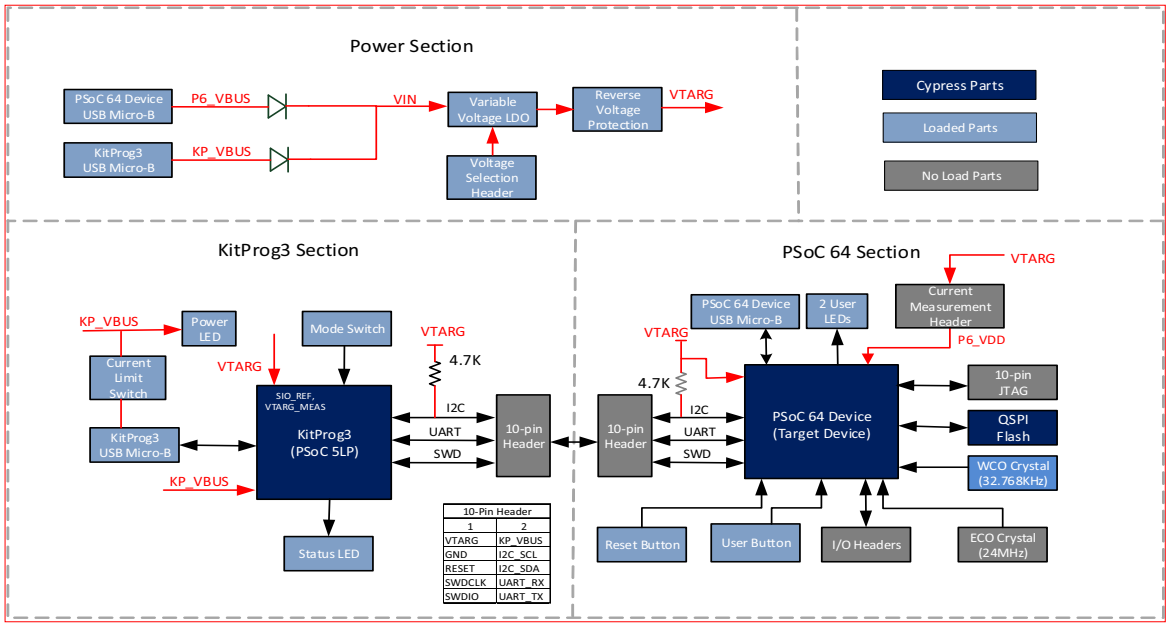
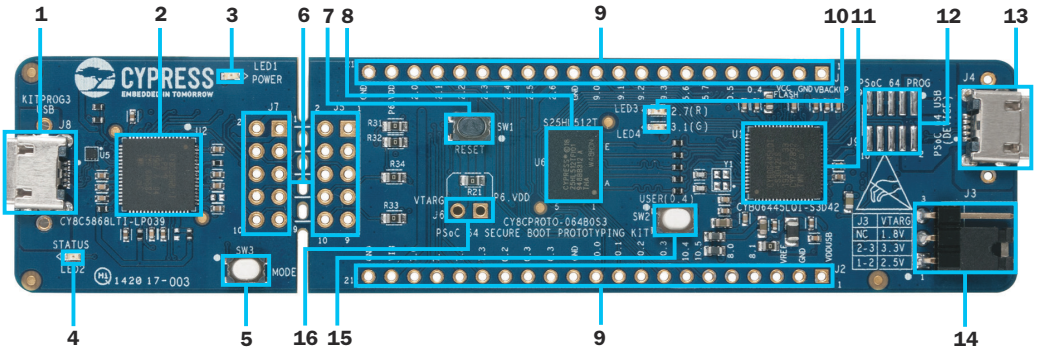


Figure 3-2. Block Diagram of Prototyping Board




Refer to KitProg3 on page 19 and Hardware Functional Description on page 32 for more details on these sections.

Figure 3-3. PSoC 64 “Secure Boot” Prototyping Board - Top View




1. KitProg3 USB connector (J8)
2. KitProg3 (PSoC 5LP) programmer and debugger (CY8C5868LTI-LP039, U2)
3. Power LED (LED1)
4. KitProg3 status LED (LED2)
5. KitProg3 programming mode selection button (SW3)
6. KitProg3 10-pin interface headers (J7, J5)*
7. Reset button (SW1)
8. 512-Mbit Semper™ NOR flash memory (S25HL512T, U6)
9. PSoC 64 I/O headers (J1, J2)*
10. PSoC 64 user LEDs (LED3, LED4)
11. PSoC 64 chip (CYB06445LQI-S3D42, U1)
12. PSoC 64 program and debug header (J9)*
13. PSoC 64 USB device connector (J4)
14. Power selection jumper (J3)
15. PSoC 64 user button (SW2)
16. PSoC 64 current measurement header (J6)*

***Not loaded by default**

The PSoC 64 “Secure Boot” Prototyping Board has the following peripherals:

1. **KitProg3 USB connector (J8):** The USB cable provided along with the board connects between this USB connector and the PC. It is used to power the entire board as well as use the onboard KitProg3 programmer/debugger.
2. **KitProg3 (PSoC 5LP) programmer and debugger (CY8C5868LTI-LP039, U2):** The PSoC 5LP device (CY8C5868LTI-LP039) serving as KitProg3, is a multi-functional system, which includes a SWD programmer, debugger, USB-I2C bridge, and USB-UART bridge. For more details, see the KitProg3 User Guide.
3. **Power LED (LED1):** LED1 is an amber LED that indicates the status of power supplied to the board. It is only applicable when powered using KitProg3.
4. **KitProg3 status LED (LED2):** Amber LED (LED2) indicates the status of KitProg3. For details on the KitProg3 status, see the KitProg3 User Guide.
5. **KitProg3 programming mode selection button (SW3):** This button can be used to switch between various modes of operation of KitProg3. For more details, see the KitProg3 User Guide.
6. **KitProg3 interface headers (J7, J5):** These headers bring out the SWD, USB-UART and USB-I2C interface of the KitProg3. This is used to program and debug the PSoC 64 device. If the KitProg3 section is broken away, it can be used to program any supported Cypress device over the 5-pin SWD interface. Note that VTARG is an input to KitProg3, and therefore the target must be powered externally. The 10-pin connector includes a pin for VBUS used to provide 5 V to the on-board regulator in the PSoC 64 section of the board. For more details on the KitProg3, see the KitProg3 User Guide.

7. **PSoC 64 reset button (SW1):** This button is used to reset the PSoC 64 device. This button connects the PSoC 64 reset (XRES) pin to ground when pressed.
8. **Cypress 512-Mbit Semper NOR flash memory (S25HL512T, U6):** An S25HL512TFABHI010 NOR flash of 512-Mbit capacity is connected to the serial memory interface (SMIF) of the PSoC 64 device. The NOR device can be used for both data and code memory with execute-in-place (XIP) support and encryption.
9. **PSoC 64 I/O headers (J1, J2):** These headers provide connectivity to PSoC 64 GPIOs. Many of these I/Os are also connected to on-board peripherals.
10. **PSoC 64 user LEDs (LED3, LED4):** The red LED (LED3) can operate at the entire operating voltage range of the PSoC 64 chip, whereas the green LED (LED4) works only at 2.5 V and higher. Both LEDs are active LOW, so the pins must be driven to ground to turn ON the LED.
11. **PSoC 64 chip (CYB06445LQI-S3D42, U1):** CYB06445LQI-S3D42 is a PSoC 64 “Secure Boot MCU” with 448 KB of flash and 152 KB of SRAM available for user applications. The chip features out-of-the-box security functionality, providing an isolated root-of-trust with true attestation and provisioning services.
12. **PSoC 64 program and debug header (J9):** This 10-pin header allows you to program and debug the PSoC 64 chip using an external programmer such as MiniProg4. Note that this is not loaded by default.
13. **PSoC 64 USB device connector (J4):** The USB cable provided with the Prototyping Kit can also be connected between this USB connector and the PC to use the PSoC 64 USB device functionality in your applications.
14. **System Power selection jumper (J3):** This jumper is used to select the PSoC 64 supply voltage (P6.VDD) between three voltages: 1.8 V, 2.5 V, and 3.3 V. It must be set to 2.5 V during provisioning.
15. **PSoC 64 user button (SW2):** This button can be used to provide an input to the PSoC 64 device. Note that by default, the button connects the PSoC 64 device pin to ground when pressed, so you need to configure the PSoC 64 pin as a digital input with resistive pull-up for detecting the button press. This button can also provide a wake-up source from low-power modes of the device.
16. **PSoC 64 current measurement header (J6):** This header can be used to measure the current consumption of the PSoC 64 using an ammeter. It is not loaded by default and is bypassed using a zero-ohm resistor across the two pins.

See Hardware Functional Description on page 32 for details on various hardware blocks.

## 3.2 KitProg3

The PSoC 64 “Secure Boot” Prototyping Board can be programmed and debugged using the onboard KitProg3. KitProg3 also has USB-UART and USB-I2C functionality. KitProg3 supports CMSIS-DAP Bulk mode and DAPLink mode for programming the target MCU using SWD. A Cypress PSoC 5LP device is used to implement KitProg3 functionality. For more details on the KitProg3 functionality, see the KitProg3 User Guide.

### 3.2.1 Programming and Debugging

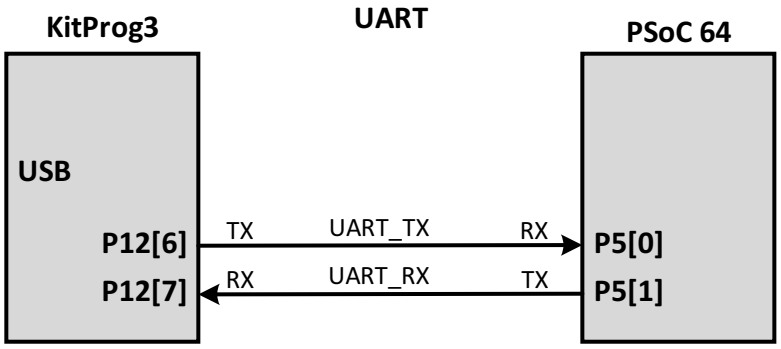
This section presents a quick overview on how to connect the kit and select the KitProg3 mode for programming and debugging. For detailed instructions, see **Help** > **ModusToolbox IDE Documentation** > **User Guide**.

Connect the board to the PC using the USB cable at KitProg3 USB connector J8. The kit enumerates as a composite device if you are connecting it to your PC for the first time. KitProg3 can operate either in CMSIS-DAP Bulk mode or DAPLink mode (default). ModusToolbox requires CMSIS-DAP Bulk mode to program or debug the device while the provisioning process requires DAPLink mode. The status LED (Amber) is always ON in CMSIS-DAP Bulk mode and ramping ON/OFF at a 2-Hz rate in DAPLink mode. Press the Mode Switch and release quickly to switch between these modes. If you do not see the desired LED status, see the KitProg3 User Guide for details on the KitProg3 status and troubleshooting instructions.

### 3.2.2 USB-UART Bridge

The KitProg3 on the board can act as a USB-UART bridge. The UART lines between the PSoC 64 device and KitProg3 are hard-wired on the board, as Figure 3-4 shows. For more details on the KitProg3 USB-UART functionality, see the KitProg3 User Guide.

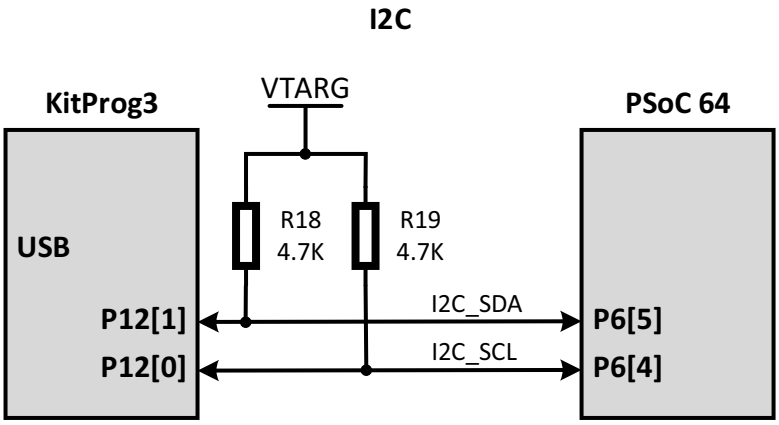
Figure 3-4. UART Connection between KitProg3 and PSoC 64 Device



### 3.2.3 USB-I2C Bridge

The KitProg3 can function as a USB-I2C bridge and communicate with the Bridge Control Panel (BCP) software which acts as an I2C master. The I2C lines on the PSoC 64 chip are hard-wired on the board to the I2C lines of the KitProg3, with onboard pull-up resistors as Figure 3-5 shows. The USB-I2C supports speeds of 50 kHz, 100 kHz, 400 kHz, and 1 MHz. For more details on the KitProg3 USB-I2C functionality, see the KitProg3 User Guide.

Figure 3-5. I2C Connection between KitProg3 and PSoC 64 Chip



# 4. Running Code on PSoC 64 “Secure Boot” MCUs

The CY8CPROTO-064B0S3 PSoC 64 “Secure Boot” Prototyping Kit can run code examples available on ModusToolbox software. However, prior to running any code on the PSoC 64 line of secured MCUs, they must first be provisioned with keys and device security policies so only signed code can be executed.

This section will go through the process to provision, build, program and run the *“Secure Blinky LED FreeRTOS”* example. Before going through the detailed steps, an overview of the provisioning process will be presented.

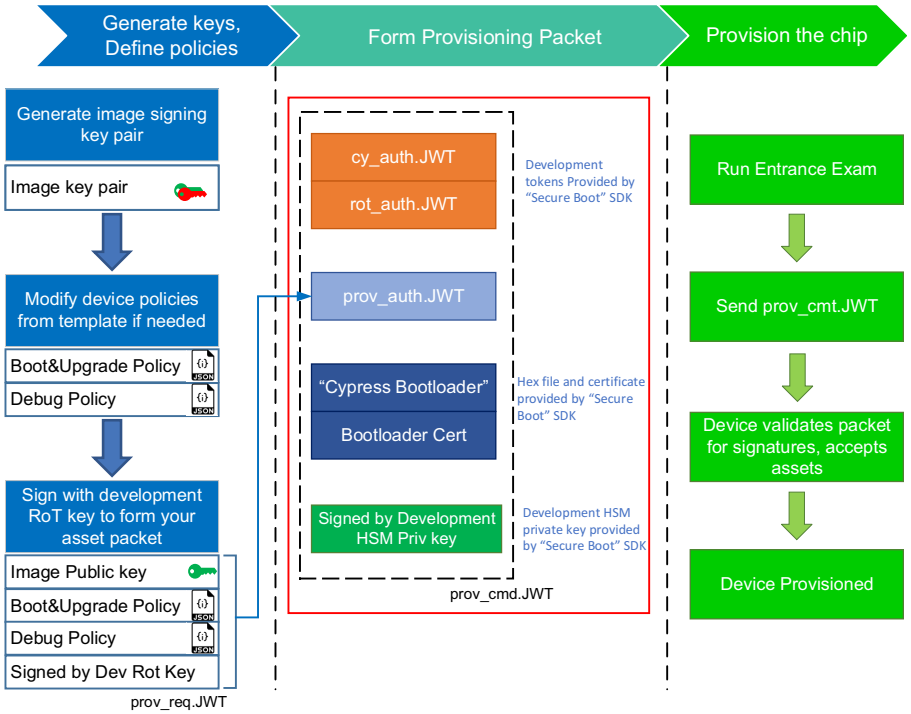
## 4.1 Provisioning Overview

Provisioning is a process by which secure assets like keys and security policies are injected into the device. This step typically occurs in a secure manufacturing environment that has a Hardware Security Module (HSM).

For a more detailed overview of what provisioning entails, see Chapter 2 of the “Secure Boot” SDK user guide.

In the context of evaluating this kit, the provisioning flow can be visualized as follows:

Figure 4-1. Provisioning Flow



For evaluation purposes, the “Secure Boot” SDK provides the following assets to easily provision your device:

1. A development Cypress Authorization JWT token (cy_auth); this authorizes a development HSM keypair which is used by your PC to provision the chip.
2. A development OEM authorization JWT token (rot_auth); this authorizes a development RoT keypair which can be used to sign your assets, such as image keys and policies.

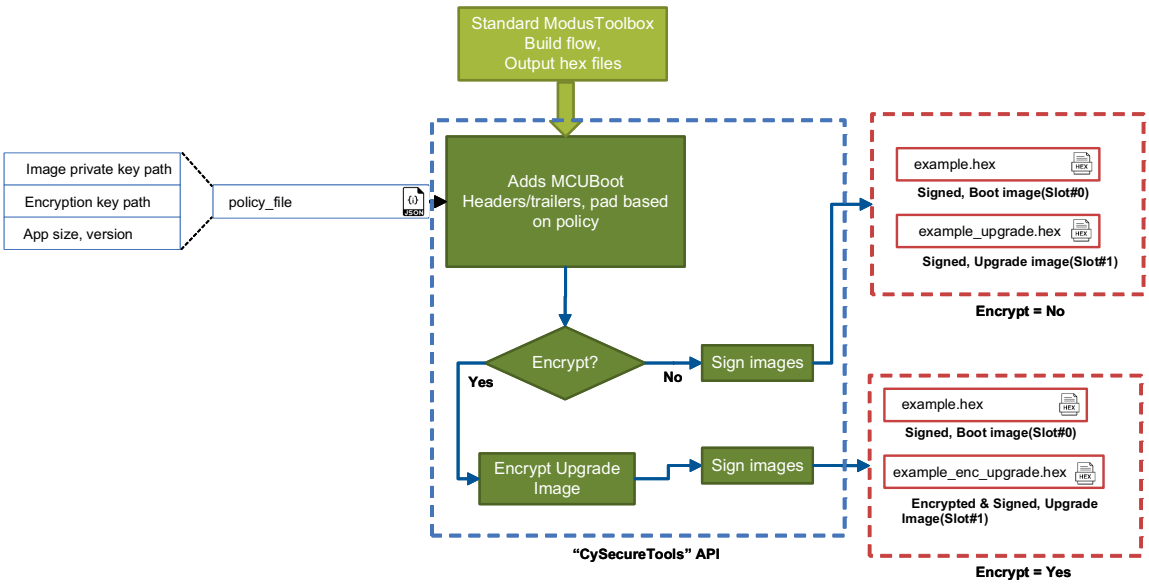
In addition, the SDK provides tools to do the following:

1. Generate image keys
2. Form provisioning packets
3. Scripts to run the entrance exam and provisioning process on you development PC

Once the chip has been provisioned with the Public Image key, it will only boot images signed by the associated Private key. Optionally, the image can be encrypted if the Boot and Upgrade policy specifies it.

The signing and encryption process is a post build script provided by the “Secure Boot” SDK. The build and encrypt/signing flow for a CY8CPROTO-064B0S3 target using ModusToolbox software make process is shown below.

Figure 4-2. Build and Encrypt/Signing Flow

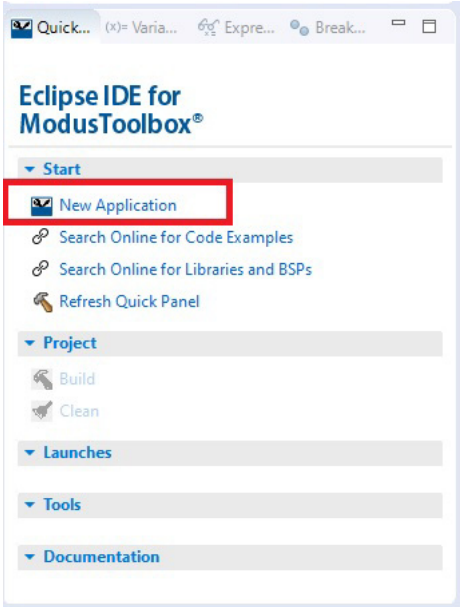


## 4.2 Create ModusToolbox example application

Now that an overview of the provisioning process has been provided, the steps to create an application, provision the device, and then build, program, run the application will be shown in detail.

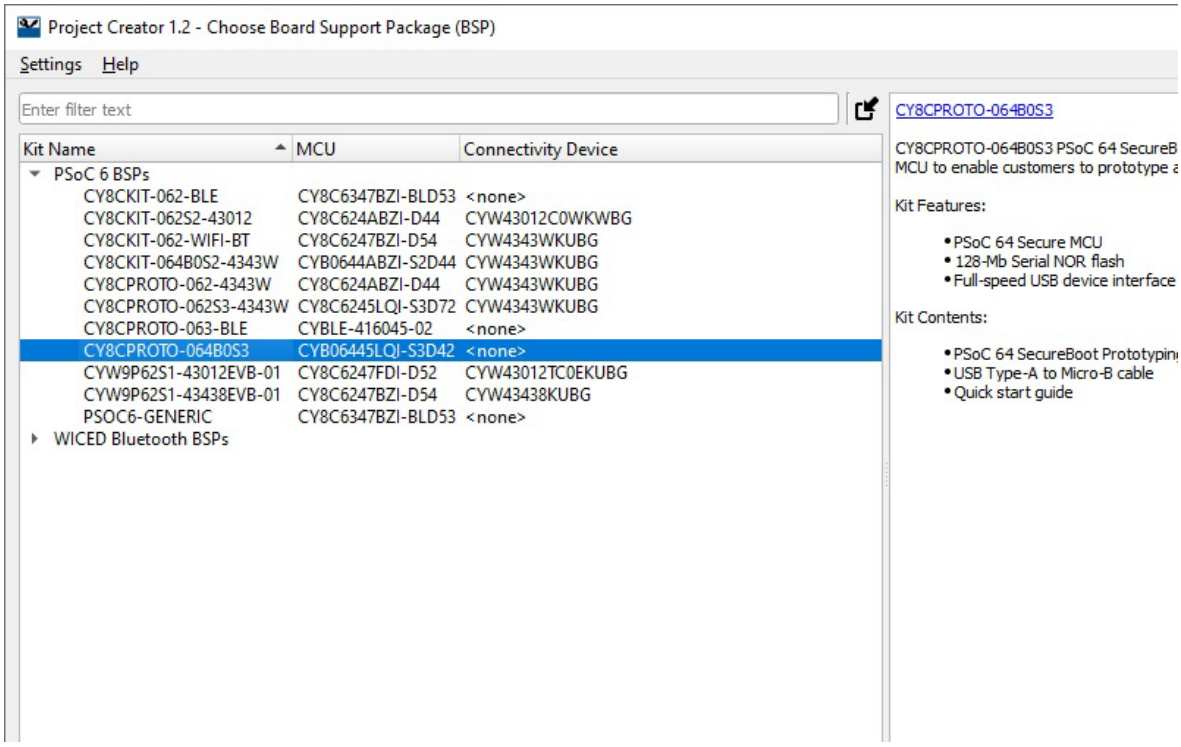
1. Open the Eclipse IDE for ModusToolbox and create a new application.

Figure 4-3. Create New Application



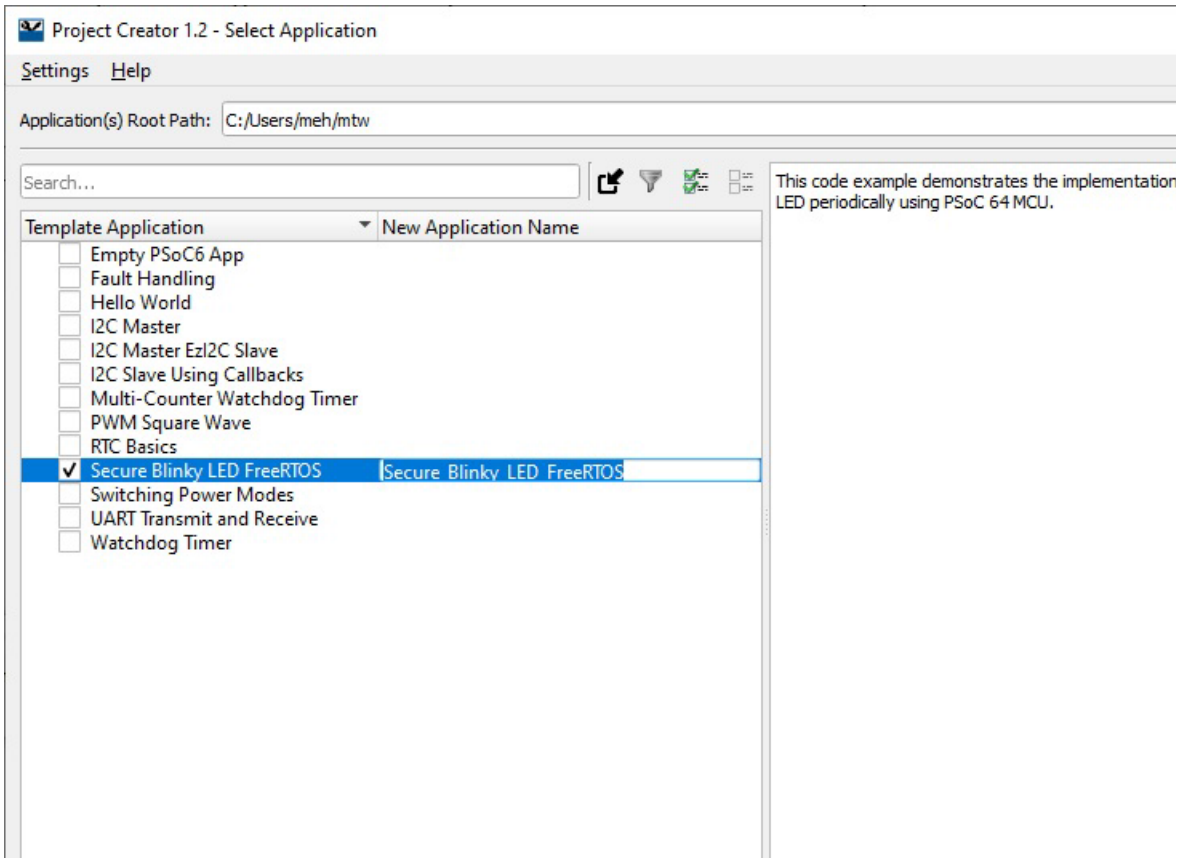

2. Select the CY8CPROTO-064B0S3 target and click **Next**.

Figure 4-4. CY8CPROTO-064B0S3 Target Selection



3. Select the *“Secure Blinky LED FreeRTOS”* example and click **Create** to create the application.

Figure 4-5. Secure Blinky LED FreeRTOS Example Selection



## 4.3 Provision the Device

1. **Navigate to your ModusToolbox application directory folder in a command-line program:**

   For Windows users, use the command line program “modus-shell” instead of a standard Windows command line application. Modus shell provides access to all ModusToolbox tools including “CySecureTools” that is used to provision a device. You can access modus-shell by typing “modus-shell” in the Windows search box in the Windows menu.

   For example, in Windows if you created your ModusToolbox application in “`C:/ModusWorkspaces/mtw/Secure_Blinky_LED_FreeRTOS`” then navigate to the folder.

2. **Setup your project workspace for “CySecureTools”.**

   **What does this step do?**

   “CySecureTools” provides a default policy which can be used to quickly setup the chip with a set of development parameters like leaving the CM4 DAP (Debug Access Port) open to reprogram the chip. Based on the selected target, this step sets up all the necessary files in your workspace that are used for subsequent steps.

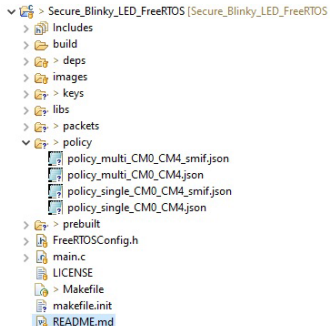
   The --target option for “CySecureTools” functions can either be the family name “cyb06xx5” or the kit name “cy8cproto-064b0s3”.

   Run the following command:

   ```
   cysecuretools --target cyb06xx5 init
   ```

   After running this step, the project workspace will have an added 'policy' directory with multiple template policies in it. This step also creates the keys, packets, and prebuilt directories. For more information on the policies, please refer to the “Secure Boot” SDK user guide.

Figure 4-6. Project Workspace having a new 'policy' directory




The four template policy files shown for this example are:

policy_multi_CM0_CM4_smif.json - Multiple application images, serial memory interface enabled

policy_multi_CM0_CM4.json - Multiple application images, serial memory interface disabled

policy_single_CM0_CM4_smif.json - Single application image, serial memory interface enabled

policy_single_CM0_CM4.json - Single application image, serial memory interface disabled

3. **Create new keys.**

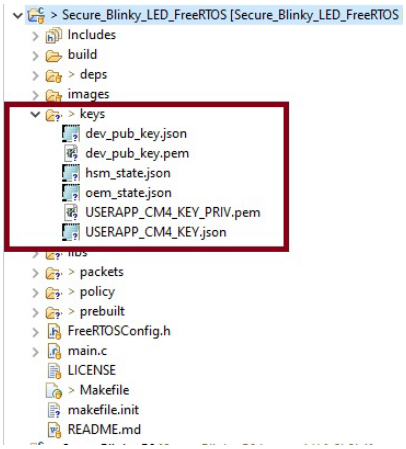
> **What does this step do?**
>
> “CySecureTools” reads the provided policy and generates the keys defined.
>
> Depending on the policy chosen, there can be multiple keys generated under the “keys” folder.
>
> By default only one key, the USERAPP_CM4_KEY, a P-256 Elliptic curve key-pair is generated.
>
> “CySecureTools” generates keys in two formats, PEM and JSON. Both the PEM and JSON files represent the same key. For a full description of all fields, please refer to the “Secure Boot” SDK user guide.

Run the following command:

```
cysecuretools --target cyb06xx5 --policy ./policy/
policy_single_CM0_CM4.json create-keys
```

After running this step, the ‘keys’ directory will have public/private key pairs and symmetric key pairs (if encrypt is enabled) generated according to the policy. These keys are used either to sign (or) encrypt images built by ModusToolbox software.

Figure 4-7. Application Workspace having a new ‘keys’ directory




4. **Connect your CY8CPROTO-064B0S3 PSoC 64 “Secure Boot” Prototyping Kit to PC using the provided USB cable through the KitProg3 USB connector.**

**ATTENTION:** Make sure jumper shunt on J3 is placed in VTARG = 2.5 V position (between pin 1 and 2) before plugging in the kit to the PC. The 2.5 V supply is necessary for Step 5, where PSoC 64 eFuses are blown. KitProg3 must be in DAPLink mode for provisioning. The Status LED (LED2) will be ramping ON/OFF (~2Hz) in this mode. Press and release the Mode button (SW3) one or more times until the KitProg3 is in DAPLink mode.

5. **Perform provisioning.**

**ATTENTION:** PSoC 64 supply voltage of 2.5 V is required to perform provisioning. The 2.5 V requirement is because this step involves blowing eFuses to change the device lifecycle to “SECURE UNCLAIMED”. If the chip is not at 2.5 V, it may cause provisioning to fail and permanently lock the chip in a dead state.

**What does this step do?**

The “CySecureTools” provision-device API does the following steps:

- Reads the provided policy and forms the final provisioning packet, named prov_cmd.jwt.
- Performs the entrance exam.
- Provisions the device by sending the prov_cmd.jwt to the PSoC 64 “Secure Boot MCU”.

Before running this step, you can modify the default policy to match your end use-case. For most development use-cases, you don’t need to modify it.

The Entrance exam is a test routine that does the following things:

- Verify that the Device is in the correct lifecycle stage.
- Verify that Boot Code has not been modified/tampered.
- Verify that User flash is empty and no code is running before any provisioning takes place.

Failing the entrance exam returns an error in the command line. If there is any firmware running on the device, existing firmware can be erased using the tools like Cypress Programmer.

Run the following command:

```
cysecuretools --target cyb06xx5 --policy ./policy/
policy_single_CM0_CM4.json provision-device
```

**KitProg3 driver issues:**

There can be sporadic issue with KitProg3 and drivers can prevent the kit from being recognized or cause other failures during the provisioning flow. Please see the Chapter 7 of the KitProg3 User Guide for information on how to resolve this.

**Note:** The provisioning procedure may take several seconds. Do not disconnect or reset the device during this procedure.

**Note:** The 2.5 V voltage option is provided only for the purpose of provisioning. For normal operation of the kit after provisioning, please load the jumper shunt on J3 at 3.3 V position (between pin 2 and 3).

**Note:** The entrance exam can be run separately without provisioning to verify the lifecycle stage of a device by using the following command:

```
cysecuretools --target cyb06xx5 --policy ./policy/
policy_single_CM0_CM4.json entrance-exam
```

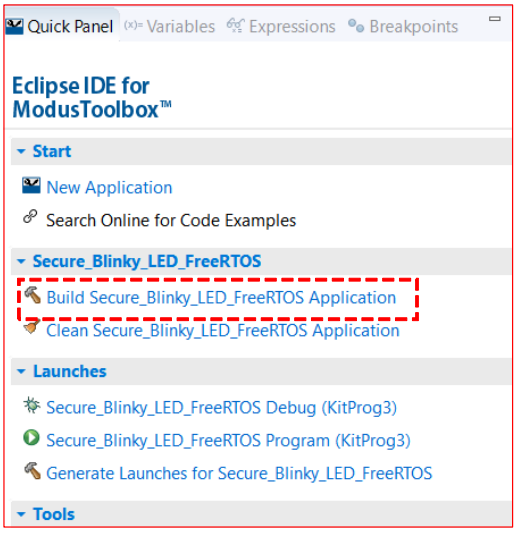
**Note:** In case you have a kit which has already been provisioned in the past with different credentials, you can re-provision the chip with new keys & policies by running the below command. Re-provisioning is possible if the policy provisioned into the chip allows this, by default all policies provided in “CySecureTools” allow re-provisioning. Note that the Entrance Exam is not run during a re-provisioning process.

```
cysecuretools --target cyb06xx5 --policy ./policy/
policy_single_CM0_CM4.json re-provision-device
```

## 4.4 Build and Program the Example Application

1. From the Eclipse IDE, click on the “Build Secure_Blinky_LED_FreeRTOS Application” link in the Quick Panel.

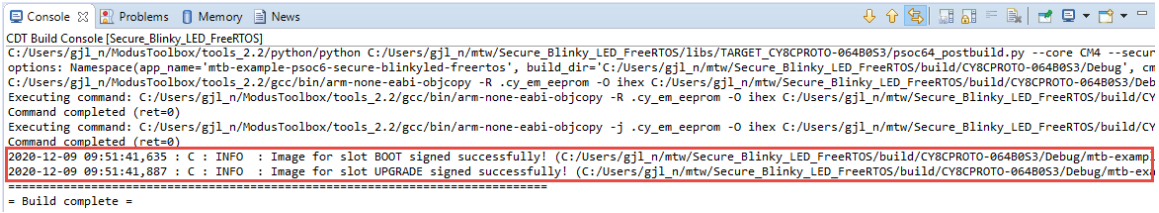
Figure 4-8. Build Secure_Blinky_LED_FreeRTOS Application Selection




**Note:** Ensure that you have clicked the application in the explorer otherwise the option will not be visible.

2. Once build is complete, you will see in the ModusToolbox console that “CySecureTools” has signed the image with the keys you generated in the previous section. Note that this message is before the memory consumption table in the build console window.

Figure 4-9. ModusToolbox Console after Build is Complete




**Note:**

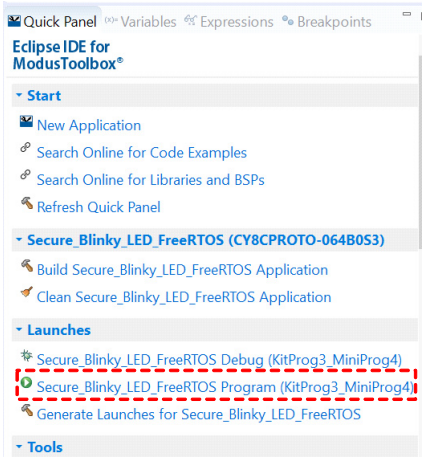
When using the CY8CPROTO_064B0S3 target, the ModusToolbox BSP has a post-build image signing command located in

*“<ModusToolbox project>/libs/TARGET_CY8CPROTO-064B0S3/CY8CPROTO-064B0S3.mk”*

This post-build script uses the policy file to find the private key path used to sign the application. If a different/invalid key is used then the “Secure Boot” process will fail.

3. The KitProg3 will still be in DAPLink mode from the provisioning step. It will need to be changed to CMSIS-DAP Bulk mode to program or debug the project. To switch to CMSIS-DAP Bulk mode, press and release the mode switch (SW3) on the kit until the status LED (LED2) is constantly on (not ramping). The kit will stay in this mode even after a power cycle, until the mode button is pressed again.
   To program the kit with the example application, click on the “Secure_Blinky_LED_FreeRTOS Program (KitProg3_MiniProg4)” in the Quick Panel. The status of the programming sequence will be displayed in the Console window.

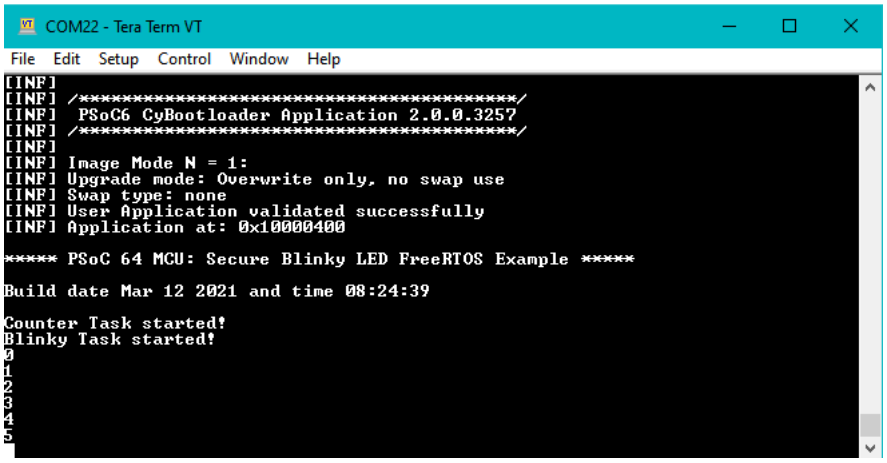
Figure 4-10. Secure_Blinky_LED_FreeRTOS Program (KitProg3_MiniProg4) Launch




4. Observe LED3 (Red) blinking every second. Open a serial terminal with a baud rate of 115200 to see the message as shown below.

   **Note:** if you open the terminal after programming the kit, just press/release the reset button to see the message on the terminal window.

Figure 4-11. UART Output



## 4.5 Additional Code Examples

Additional code examples for PSoC 64 device can be found on the Cypress git repository

https://github.com/cypresssemiconductorco

Note that most PSoC 6 MCU code examples will run on PSoC 64 devices. If you wish to run other code examples on the existing kit, you can follow the same steps outlined in Section 4.2 to 4.4 with the alternate project imported into your workspace. There are two alternative flows for this.

1. If you want to re-use existing keys and policies provisioned into the kit, please copy the ‘keys’ and ‘policy’ folder from your existing project workspace into your new project workspace. In this flow, you will re-use the existing key and policy file to sign firmware and program.
2. If you want to create new keys and policies, please follow the re-provisioning note provided in Section 4.3, step 5.

# 5. Hardware

## 5.1 Schematics

Refer to the schematic files available on the kit webpage.

## 5.2 Hardware Functional Description

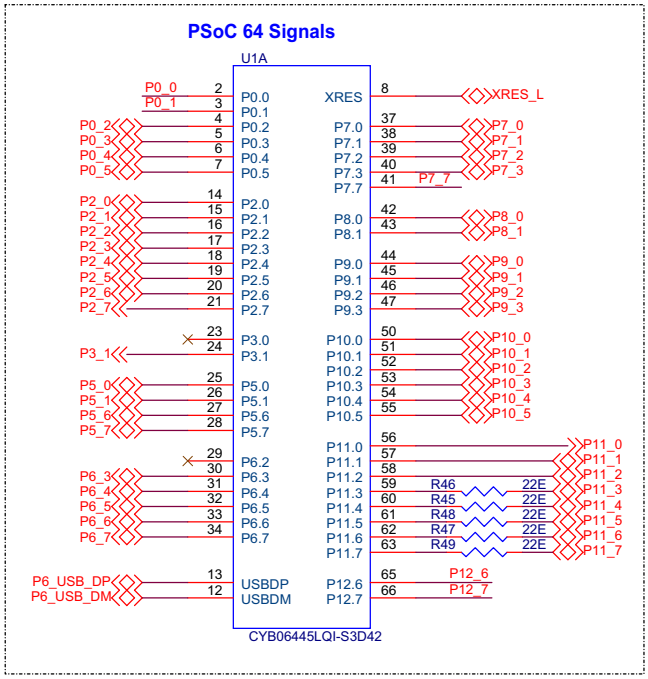
This section explains in detail the individual hardware blocks of the PSoC 64 “Secure Boot” Prototyping Board.

### 5.2.1 CYB06445LQI-S3D42 (U1)

The PSoC 64 microcontroller is a high-performance, ultra-low-power and secured MCU platform, purpose-built for IoT applications. The PSoC 64 “Secure Boot MCU” line, based on the PSoC 6 MCU platform, features out-of-the-box security functionality. The line provides an isolated RoT with true attestation and provisioning services. In addition, these MCUs deliver a pre-configured secure execution environment which supports system software of various IoT platforms and provides secure provisioning, key storage, and firmware management.

For more information, see the PSoC 64 web page and the datasheet.

Figure 5-1. Schematics of PSoC 64 Signals

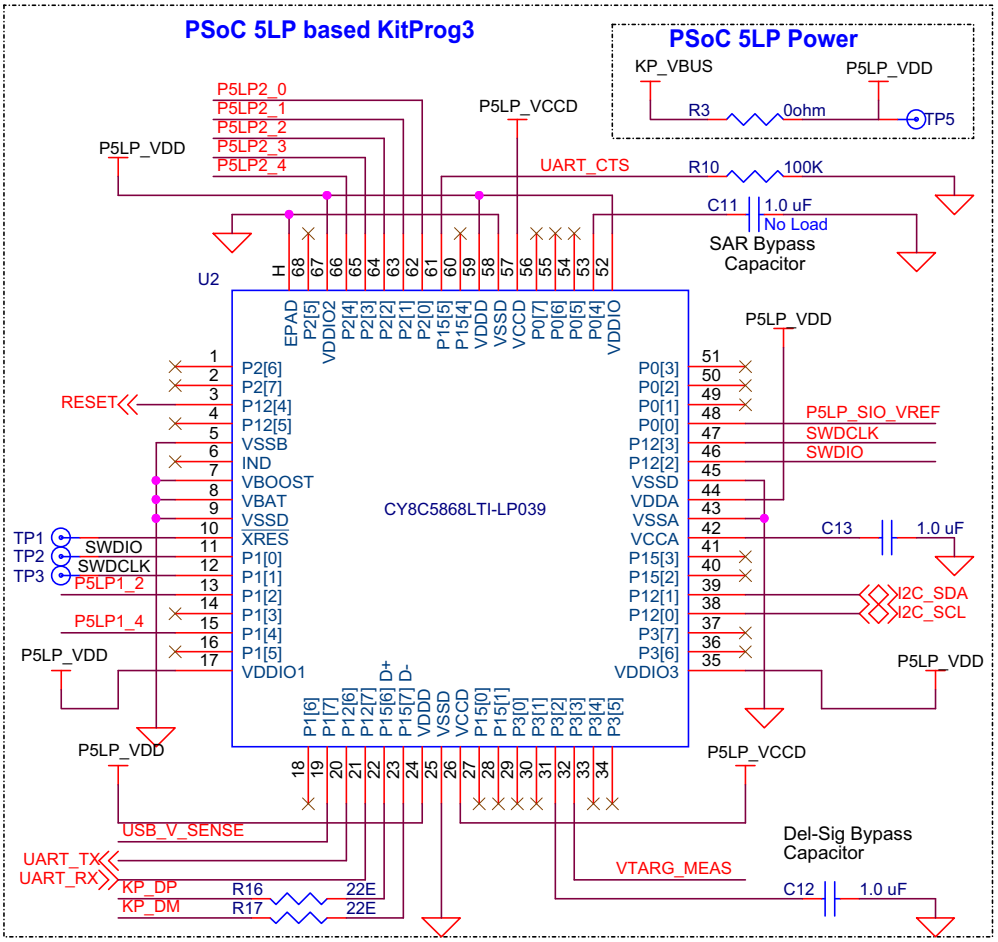


### 5.2.2 PSoC 5LP based KitProg3 (U2)

An onboard PSoC 5LP (CY8C5868LTI-LP039) device is used as KitProg3 to program and debug the PSoC 64 device. The PSoC 5LP device connects to the USB port of the PC through a USB connector and to the SWD and other communication interfaces of the PSoC 64 device.

The PSoC 5LP device is a true system-level solution providing MCU, memory, analog, and digital peripheral functions in a single chip. For more information, visit the PSoC 5LP web page. Also, see the CY8C58LPxx Family datasheet.

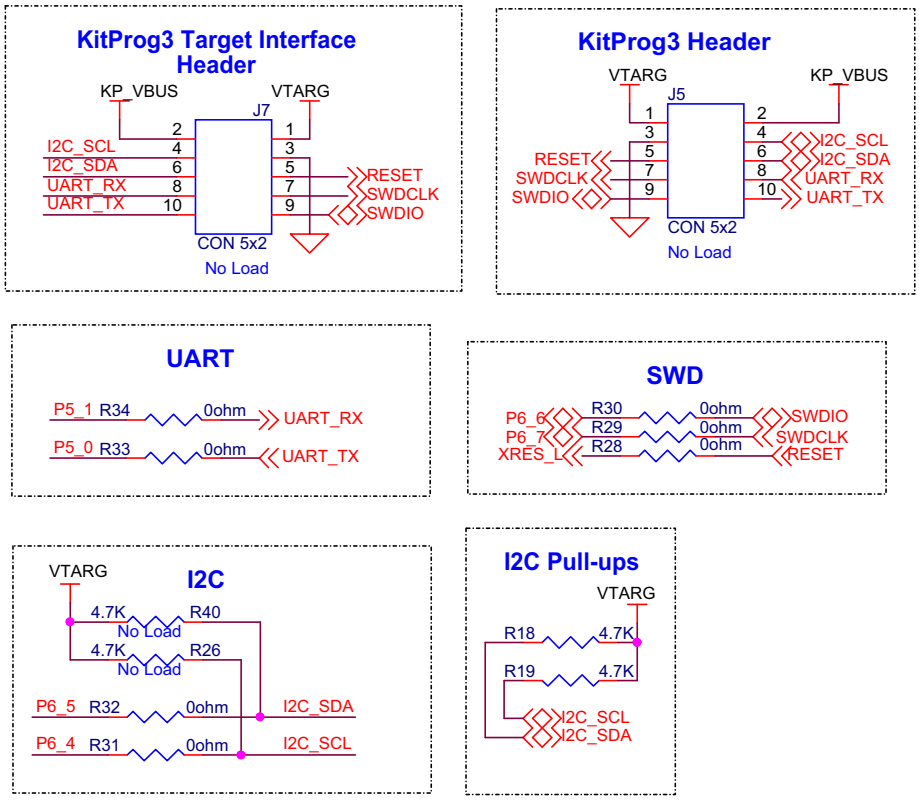
Figure 5-2. Schematics of PSoC 5LP based KitProg3 and PSoC 5LP Power



### 5.2.3 Serial Interconnection between PSoC 5LP and PSoC 64 Device

In addition of its use as an onboard programmer, the PSoC 5LP device functions as an interface for the USB-UART and USB-I2C bridges, as shown in Figure 5-3. The USB-Serial pins of the PSoC 5LP device are hard-wired to the I2C/UART pins of the PSoC 64 chip. These pins are on J7 on the KitProg3 section and on J5 on the PSoC 64 section.

Figure 5-3. Schematics of Programming and Serial Interface Connections




**Note:** R18 and R19 are located on the KitProg3 section of the board and are loaded by default. If the PSoC 64 device section is separated, there is a provision to load R26 and R40 located on that section to use as I2C pullups.

## 5.2.4 Power Supply System

The power supply system on this board is versatile, allowing the input supply to come from the following sources:

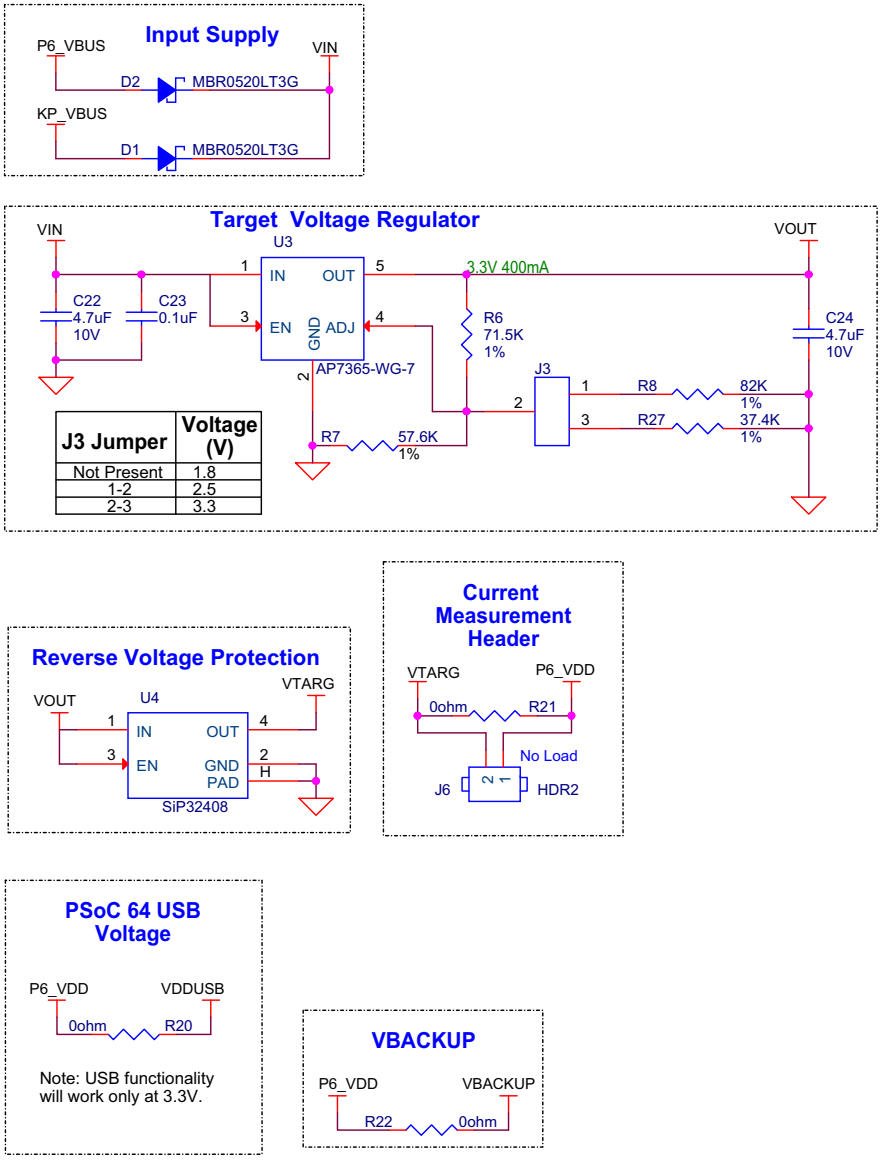
- 5 V from the onboard USB Micro-B connectors (**J4** and **J8**)
- 5 V from external power supply through VIN header **J2.20**

The power supply system is designed to support 1.8 V, 2.5 V or 3.3 V operation of the PSoC 64 device. 5 V provided from the USB port (**J8**) is used for the operation of KitProg3. USB bus voltages are routed through an ORing circuit to allow powering the PSoC 64 chip from either USB port.

One linear regulator is used to achieve either 1.8 V, 2.5 V or 3.3 V to power the PSoC 64 chip and peripherals. Figure 5-4 shows the schematics of the voltage regulator and power selection circuits.

The voltage selection is made through jumper (**J3**).

Figure 5-4. Schematics of Power Supply System



### 5.2.4.1 Measure PSoC 64 Device Current Consumption

To measure the PSoC 64 current, follow these steps:

1. Remove the zero-ohm resistor R21 and install a 2-pin header at J6.
2. Connect an ammeter across the 2-pin header J6.

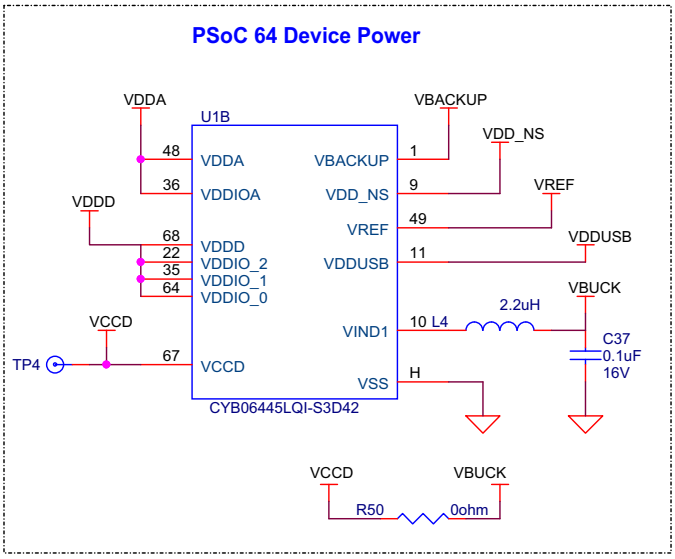
This method can be used with KitProg3 USB (J8), VIN or with the power supplied to one of the VTARG pins (J5.1 or J7.1), but NOT when supplying power to one of the P6.VDD pins (J1.20) or when powered through PSoC 64 USB (J4).

After measuring the current consumption, populate resistor R21 or place a shorting jumper across the two jumper pins for normal operation of the kit.

**Note:**

If KitProg3 is connected to a PC and VTARG is not connected to P6.VDD (either by removal of R21 or header J6 is left open), PSoC 64 chip will be back-powered through the KitProg3 to PSoC 64 interfaces (SWD, I2C, UART). For accurate current measurement, detach theKitProg3 section from the PSoC 64 section.

Figure 5-5. Schematics of PSoC 64 Device Power



## 5.2.5 Expansion Connectors

### *5.2.5.1 Functionality of the J1 and J2 Headers (Target Board)*

The target PSoC 64 section contains two single inline headers (J1 and J2). These 1×21-pin headers have 0.1-inch spacing and include a subset of the GPIOs available on the PSoC 64 device.

Figure 5-6. J1 and J2 Headers

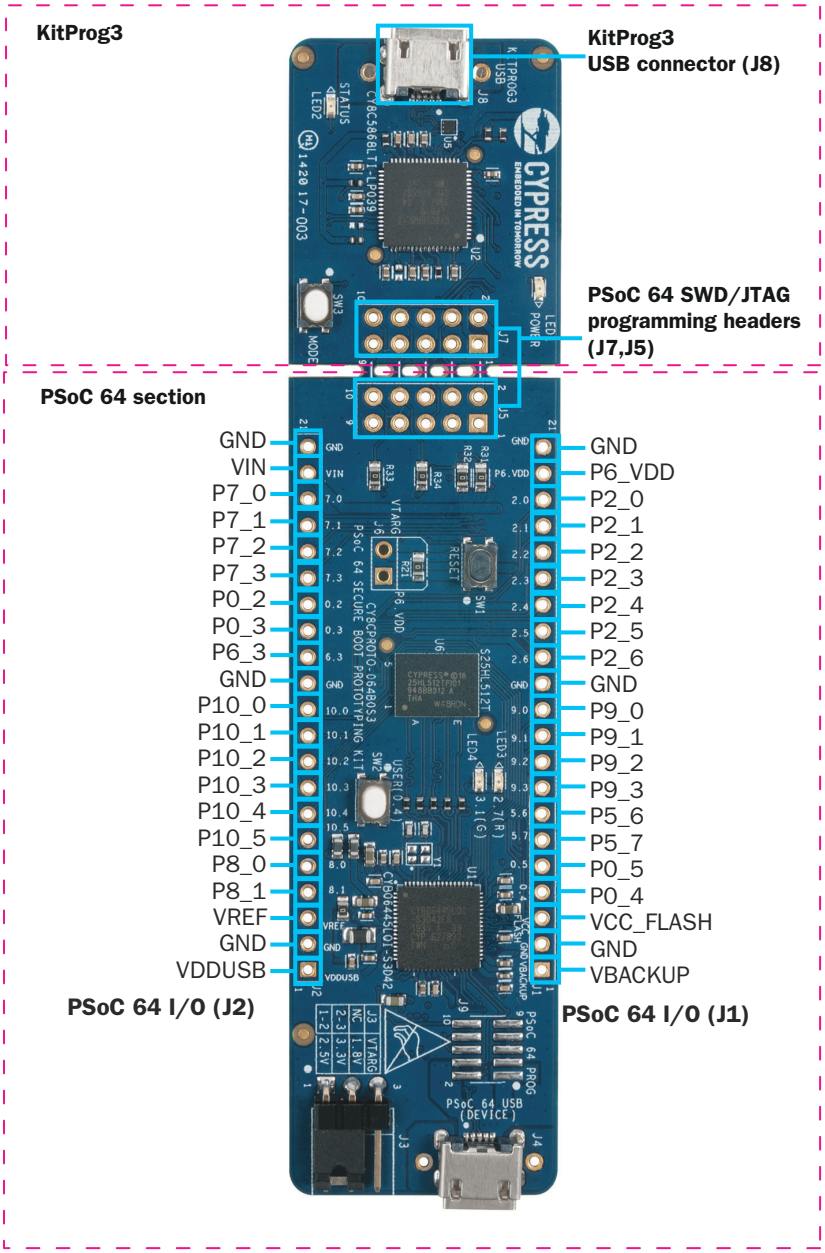


Table 5-1. Pin Details of J1 and J2 Headers

| PSoC 64 “Secure Boot” Prototyping Board GPIO Header (J2) | | | PSoC 64 “Secure Boot” Prototyping Board GPIO Header (J1) | | |
|---|---|---|---|---|---|
| **Pin** | **Signal** | **Description** | **Pin** | **Signal** | **Description** |
| J2_01 | VDDUSB | USB Power | J1_01 | VBACKUP* | Backup Power |
| J2_02 | GND | Ground | J1_02 | GND | Ground |
| J2_03 | VREF | SAR ADC Vref | J1_03 | VCC_FLASH | QSPI Flash Power |
| J2_04 | P8[1] | GPIO | J1_04 | P0[4] | GPIO/User SW2 |
| J2_05 | P8[0] | GPIO | J1_05 | P0[5] | GPIO |
| J2_06 | P10[5] | GPIO | J1_06 | P5[7] | GPIO |
| J2_07 | P10[4] | GPIO | J1_07 | P5[6] | GPIO |
| J2_08 | P10[3] | GPIO | J1_08 | P9[3] | GPIO |
| J2_09 | P10[2] | GPIO | J1_09 | P9[2] | GPIO |
| J2_10 | P10[1] | GPIO | J1_10 | P9[1] | GPIO |
| J2_11 | P10[0] | GPIO | J1_11 | P9[0] | GPIO |
| J2_12 | GND | Ground | J1_12 | GND | Ground |
| J2_13 | P6[3] | GPIO | J1_13 | P2[6] | GPIO |
| J2_14 | P0[3] | GPIO | J1_14 | P2[5] | GPIO |
| J2_15 | P0[2] | GPIO | J1_15 | P2[4] | GPIO |
| J2_16 | P7[3] | GPIO | J1_16 | P2[3] | GPIO |
| J2_17 | P7[2] | GPIO | J1_17 | P2[2] | GPIO |
| J2_18 | P7[1] | GPIO | J1_18 | P2[1] | GPIO |
| J2_19 | P7[0] | GPIO | J1_19 | P2[0] | GPIO |
| J2_20 | VIN | Input Voltage | J1_20 | P6_VDD* | Target Voltage Input |
| J2_21 | GND | Ground | J1_21 | GND | Ground |

* **Note:** P6_VDD and VBACKUP should never exceed 3.6 V.

### 5.2.5.2 *Functionality of J7 and J5 Headers (KitProg3 to PSoC 64 Device)*

The KitProg3 and target sections of the board each contain a 2 × 5-pin header. These headers provide a physical connection between the two devices. This connection contains the following signals:

1. SWD interface required to program/debug the target PSoC 64 device, power, ground, and reset
2. UART interface to the PSoC 64 device
3. I2C interface to the PSoC 64 device

Figure 5-7. J7 and J5 Headers

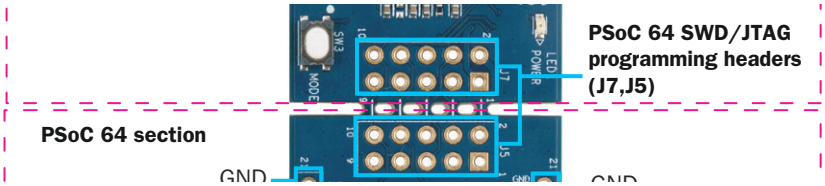



Table 5-2. Pin Details of J7 Header

| KitProg3 Header (J7) | | | | | |
|---|---|---|---|---|---|
| **Pin** | **Signal** | **Description** | **Pin** | **Signal** | **Description** |
| J7_1 | VTARG | Power | J7_2 | KP_VBUS | Power |
| J7_3 | GND | Ground | J7_4 | P12[0] | I2C_SCL |
| J7_5 | P12[4] | Reset | J7_6 | P12[1] | I2C_SDA |
| J7_7 | P12[3] | SWD_CLK | J7_8 | P12[7] | UART_RX |
| J7_9 | P12[2] | SWD_IO | J7_10 | P12[6] | UART_TX |

Table 5-3. Pin Details of J5 Header

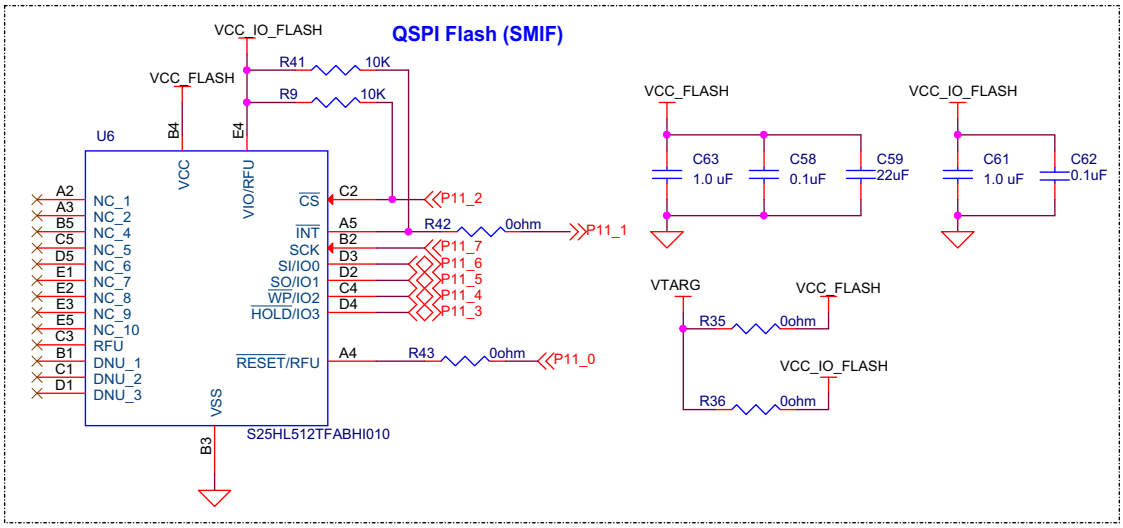
| PSoC 64 Header (J5) | | | | | |
|---|---|---|---|---|---|
| **Pin** | **Signal** | **Description** | **Pin** | **Signal** | **Description** |
| J5_1 | VTARG | Power | J5_2 | VTARG | Power |
| J5_3 | GND | Ground | J5_4 | P6[4] | I2C_SCL |
| J5_5 | XRES_L | Reset | J5_6 | P6[5] | I2C_SDA |
| J5_7 | P6[7] | SWD_CLK | J5_8 | P5[1] | UART_TX |
| J5_9 | P6[6] | SWD_IO | J5_10 | P5[0] | UART_RX |

When the boards are separated, the KitProg3 board can be used to program other target devices supported by KitProg3 via J7. Headers J5 and J7 can be used to reconnect the KitProg3 and PSoC 64 sections of the kit.

## 5.2.6 Quad SPI Flash

The board has a Cypress Semper NOR flash memory (S25HL512TFABHI010) of 512-Mbit capacity. The NOR Flash is connected to the serial memory interface (SMIF) of the PSoC 64 device. The NOR Flash device can be used for both data and code memory with execute-in-place (XIP) support and encryption. Note that the policy used during provisioning must support the SMIF interface to be able to store code in the external flash. The policies that enable the SMIF interface are provided in the templates.

Figure 5-8. Schematics of Quad SPI Flash

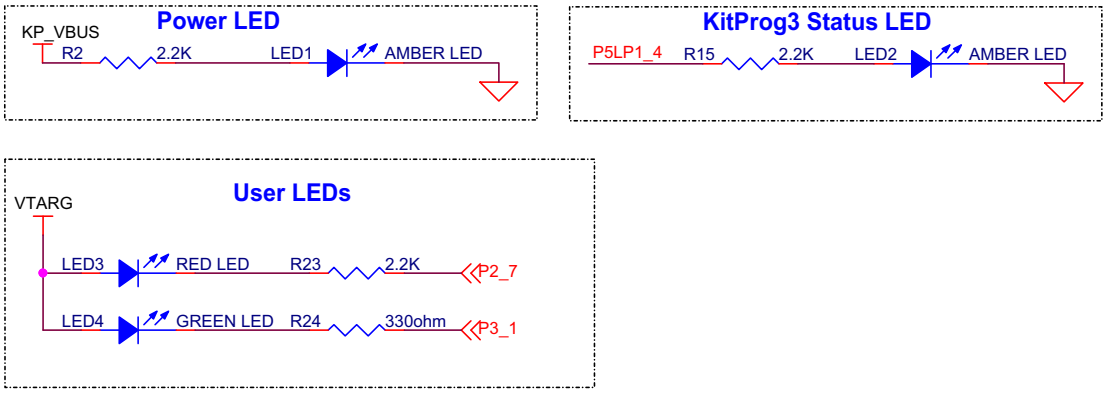



## 5.2.7 LEDs

**LED2** (Amber) indicates the status of KitProg3 (See the KitProg3 User Guide for details). **LED1** indicates the status of power supplied to KitProg3.

The board has two user-controllable LEDs (**LED3**, **LED4**) connected to the PSoC 64 pins (**P2[7]**, **P3[1]**) for user applications.

Figure 5-9. Schematics of LEDs

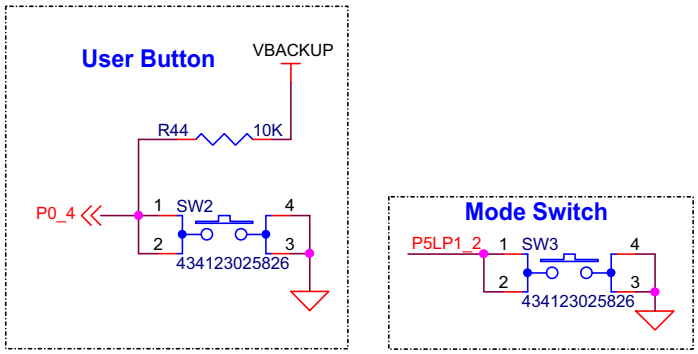


### 5.2.8 User Buttons

The target PSoC 64 board contains a user button (**SW2**) connected to the P0[4] pin on the PSoC 64 chip. This button can be used for general user inputs or for wakeup during Hibernate mode.

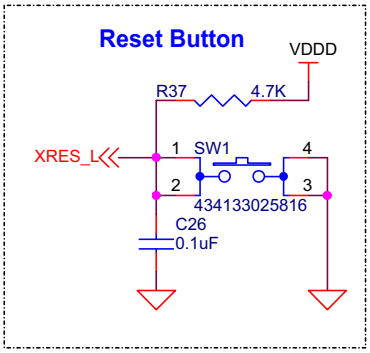
The board has a reset button and mode select button. The reset button (**SW1**) is connected to the XRES pin of the PSoC 64 device and is used to reset the device. The mode select button (**SW3**) is connected to the PSoC 5LP device for programming mode (Refer to the KitProg3 User Guide for details). All the buttons connect to ground on activation (active LOW).

Figure 5-10. Schematics of Push Buttons




When the Reset button (**SW1**) is pressed, the XRES line of the PSoC 64 chip is pulled to ground, which resets the target device.

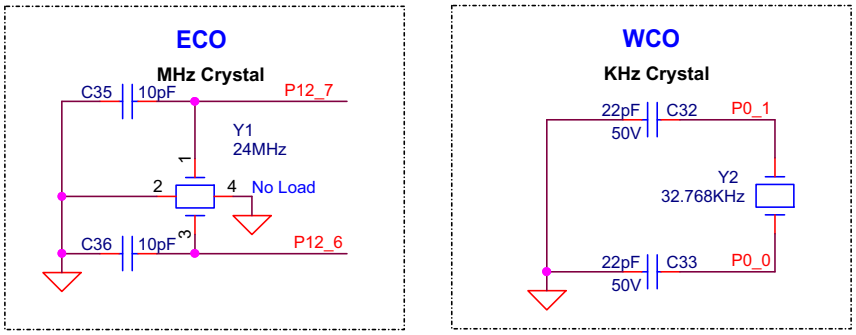
Figure 5-11. Schematics of Reset Button



### 5.2.9 System Crystals

Two different crystal oscillator inputs are available on the board. The WCO kHz crystal (32.768 kHz) is populated and is used for timing. A footprint for the ECO MHz crystal and load capacitors are on the board so that you can easily select the crystal of your choice. The ECO is optional and only required when the internal clock must be more accurate than the internal main oscillator (IMO). The internal FLL and PLL help to provide a wide range of options with either the ECO or IMO.

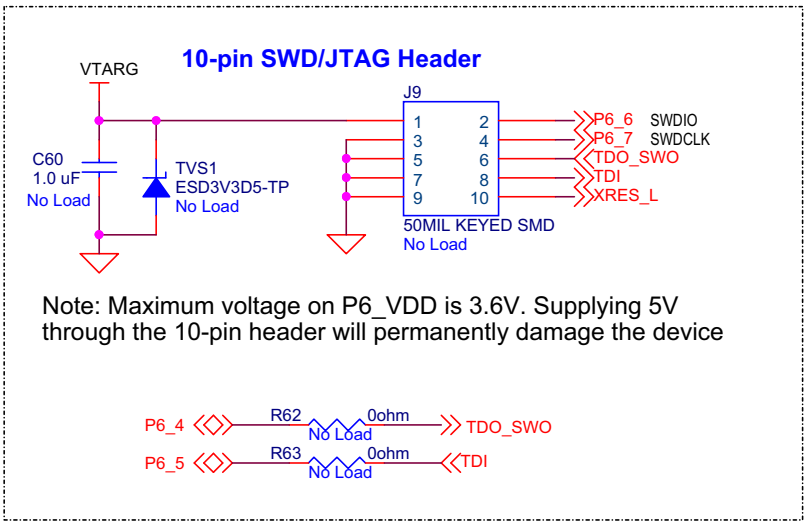
Figure 5-12. Schematics of Crystal Oscillator




### 5.2.10 10-pin SWD/JTAG Programming Header

The board has a provision for the 10-pin header **J9**, which allows you to program and debug the PSoC 64 device using an external programmer such as MiniProg4. This is not loaded by default.

Figure 5-13. Schematics of 10-pin SWD/JTAG Header




## 5.3 Bill of Materials

Refer to the BOM files in the kit webpage.

## 5.4 Frequently Asked Questions

1. How does CY8CPROTO-064B0S3 handle voltage connections when multiple power sources are plugged in?

   There are three different options to power the baseboard; KitProg3 Micro-B USB connector (**J8**), PSoC 64 device's Micro-B USB connector (**J4**), and External DC supply via VIN connector (**J2.20**). The voltage from each of the USB connectors passes through a current limiting switch that also protects against reverse voltage. The output of both current limit switches is given to VCC_5V that is also present on **J2.20**.

2. What are the input voltage tolerance ranges? Is there any overvoltage protection on this kit?

   There is no over-voltage protection on this kit. Input voltage levels are as follows:

Table 1-4. Input voltage levels

| Supply | Typical i/p voltage | Absolute max |
|---|---|---|
| USB Micro-B connector (**J4, J8**) | 4.5 V to 5.5 V | 5.5 V |
| VIN connector (**J2.20**) | 5 V to 5.5 V | 6 V |
| Program and Debug header (**J9**) | 1.8 V to 3.3 V | 3.6 V |

3. Why is the voltage of the kit restricted to 3.3 V? Can't it drive external 5 V interfaces?

   The PSoC 64 chip is not meant to be operated at voltages greater than 3.6 V. Powering the PSoC 64 chip to more than 4 V will damage it. You cannot drive the I/O system with > 3.6 V supply voltages.

4. I am unable to program the target device.
   a. Check if **R21** is present or **J6** to ensure that the jumper shunt is present on the board.
   b. Verify that the KitProg3 is in the CMSIS-DAP Bulk mode, indicated by the status (**LED2**) on steady, not blinking. If it is blinking, press the mode button to select the CMSIS-DAP Bulk mode.
   c. Update your KitProg3 version to latest using the steps mentioned in KitProg3 User Guide.

5. Does the kit get powered when I power the kit from another Cypress kit through the **J2.20** header?

   Yes, the VIN pin on the **J2.20** header is the supply input/output pin and can take up to 5.5 V.

6. Can I use this Kit as a programmer to program external PSoC devices?

   Yes, the onboard KitProg3 can program any supported target device connected to **J7** header but only after the KitProg3 section is separated from the PSoC 64 section of the board.

# Revision History

## Document Revision History

| Document Title: CY8CPROTO-064B0S3 PSoC 64 “Secure Boot” Prototyping Kit Guide<br>Document Number: 002-29505 | | | |
|---|---|---|---|
| **Revision** | **ECN Number** | **Issue Date** | **Description of Change** |
| ** | 6778499 | 10/09/2020 | New kit guide. |
| *A | 7040748 | 12/10/2020 | Updated Software Installation chapter on page 13:<br>Updated “Install Software” on page 13:<br>Updated description.<br>Removed “Updating tools for ModusToolbox 2.1 or older”.<br>Updated Running Code on PSoC 64 “Secure Boot” MCUs chapter on page 21:<br>Updated “Create ModusToolbox example application” on page 23:<br>Updated description.<br>Updated “Provision the Device” on page 25:<br>Updated description.<br>Removed figure “Command Prompt”.<br>Updated “Build and Program the Example Application” on page 29:<br>Updated Figure 4-9. |
| *B | 7062683 | 04/23/2021 | Updated Document Title to read as “CY8CPROTO-064B0S3 PSoC 64 “Secure Boot” Prototyping Kit Guide”.<br>Replaced “Secure Boot” with ““Secure Boot”” in all instances across the document.<br>Replaced “PSoC 64 Secure MCU” with “PSoC 64 “Secure Boot” MCU” in all instances across the document.<br>Replaced “ModusToolbox” with “ModusToolbox software” in all instances across the document.<br>Replaced “CySecureTools” and “cysecuretools” with ““CySecureTools”” in all instances across the document.<br>Updated Introduction chapter on page 6:<br>Updated description.<br>Updated hyperlinks.<br>Updated “Board Details” on page 7:<br>Updated description.<br>Updated Figure 1-1.<br>Updated Table 1-1. |

## Document Revision History *(continued)*

| Document Title: CY8CPROTO-064B0S3 PSoC 64 “Secure Boot” Prototyping Kit Guide<br>Document Number: 002-29505 | | | |
|---|---|---|---|
| **Revision** | **ECN Number** | **Issue Date** | **Description of Change** |
| *B (cont.) | 7062683 | 04/23/2021 | Updated Software Installation chapter on page 13:<br>Updated description.<br>Updated “Install Software” on page 13:<br>Updated description.<br>Updated hyperlinks.<br>Updated Kit Operation chapter on page 15:<br>Updated description.<br>Updated “Theory of Operation” on page 15:<br>Updated description.<br>Updated hyperlinks.<br>Updated Figure 3-1.<br>Updated Figure 3-2.<br>Updated Figure 3-3.<br>Updated “KitProg3” on page 19:<br>Updated description.<br>Updated “USB-UART Bridge” on page 19:<br>Updated description.<br>Updated Figure 3-4.<br>Updated “USB-I2C Bridge” on page 20:<br>Updated description.<br>Updated Figure 3-5.<br>Updated Running Code on PSoC 64 “Secure Boot” MCUs chapter on page 21:<br>Updated description.<br>Added hyperlinks in required places.<br>Updated “Provisioning Overview” on page 21:<br>Updated description.<br>Updated Figure 4-1.<br>Updated Figure 4-2.<br>Updated “Create ModusToolbox example application” on page 23:<br>Updated description.<br>Updated “Provision the Device” on page 25:<br>Updated description.<br>Updated “Build and Program the Example Application” on page 29:<br>Updated description.<br>Updated Figure 4-10 (Updated caption only).<br>Updated Figure 4-11.<br>Updated “Additional Code Examples” on page 31:<br>Updated description. |

## Document Revision History *(continued)*

| Document Title: CY8CPROTO-064B0S3 PSoC 64 “Secure Boot” Prototyping Kit Guide<br>Document Number: 002-29505 | | | |
|---|---|---|---|
| **Revision** | **ECN Number** | **Issue Date** | **Description of Change** |
| *B (cont.) | 7062683 | 04/23/2021 | Updated Hardware chapter on page 32:<br>Updated “Hardware Functional Description” on page 32:<br>Updated “CYB06445LQI-S3D42 (U1)” on page 32:<br>Updated description.<br>Updated hyperlinks.<br>Updated Figure 5-1.<br>Updated “PSoC 5LP based KitProg3 (U2)” on page 33:<br>Updated description.<br>Updated “Serial Interconnection between PSoC 5LP and PSoC 64 Device” on page 34:<br>Replaced “MCU” with “Device” in heading.<br>Updated description.<br>Updated “Power Supply System” on page 35:<br>Updated description.<br>Updated Figure 5-4.<br>Updated “Measure PSoC 64 Device Current Consumption” on page 36:<br>Replaced “MCU” with “Device” in heading.<br>Updated description.<br>Updated Figure 5-5.<br>Updated “Expansion Connectors” on page 37:<br>Updated “Functionality of the J1 and J2 Headers (Target Board)” on page 37:<br>Updated description.<br>Updated Figure 5-6.<br>Updated Table 5-1.<br>Updated “Functionality of J7 and J5 Headers (KitProg3 to PSoC 64 Device)” on page 39:<br>Replaced “MCU” with “Device” in heading.<br>Updated description.<br>Updated Figure 5-7.<br>Updated Table 5-3.<br>Updated “Quad SPI Flash” on page 40:<br>Updated description.<br>Updated “LEDs” on page 40:<br>Updated description.<br>Updated “User Buttons” on page 41:<br>Updated description.<br>Updated “10-pin SWD/JTAG Programming Header” on page 42:<br>Updated description. |