CY8CPROTO-064S1-SB

PSoC 64S1 Secure Boot Prototyping Kit User Guide

Doc. # 002-28075 Rev. **

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

Copyrights

Contents

Safety and Regulatory Compliance Information

The CY8CPROTO-064S1-SB PSoC 64S1 Secure Boot Kit is intended for development purposes only. 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 SecureBoot 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 PSoC64 SecureBoot 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-064S1-SB PSoC 64S1 Secure Boot 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.

Regulatory Compliance Information

The CY8CPROTO-064S1-SB PSoC 64S1 SecureBoot Prototyping Kit as shipped from the factory has been verified to meet with requirements a Class A compliant product meeting requirement for CE.

1. Introduction

Thank you for your interest in the CY8CPROTO-064S1-SB PSoC 64S1 SecureBoot Prototyping Kit. The PSoC 64S1 SecureBoot Prototyping Kit enables you to evaluate and develop your applications using the PSoC 64 Line of Secure MCUs (hereafter called “PSoC 64 MCU”).

PSoC 6 MCU is a high-performance, ultra-low-power and secure MCU platform, purpose-built for IoT applications. The PSoC 64 Secure MCU line, 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-064S1-SB Secure Boot kit carries a PSoC 64 MCU. In addition, the board features an onboard programmer/debugger (KitProg3), a 128-MBit NOR flash, a micro-B connector for USB device interface, two user LEDs and one user push button. The board supports operating voltages of 1.8V, 2.5V and 3.3V for PSoC 64 MCU.

1.1 Kit Contents

The CY8CPROTO-064S1-SB package has the following contents.

- 3x PSoC 64 SecureBoot Prototyping Board
- USB Type-A to Micro-B cable
- Quick Start Guide, printed on kit package

This kit contains 3 identical boards which can be provisioned keys and policies for evaluation. Once the kit is provisioined with a specific set of keys/policies, it will be permanently bound to them. 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 Code Examples

Code examples for Mbed OS can be found at the Mbed OS examples page.

Note that some MbedOS examples will work not for this kit, specifically the examples using external communication functions such as WiFi/BT/BLE. See Import MbedOS example on page 24 to learn how to import examples from the repository and build using mbed-cli.

1.3 Getting Started

This guide will help you to get acquainted with the CY8CPROTO-064S1-SB Secure Boot Kit:

- The Software Installation chapter on page 14 describes the installation of the kit software. This includes the MbedOS to develop, program and debug applications on to the device.
- The Kit Operation chapter on page 16 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 PSoC64 Secure MCUs chapter on page 22 describes multiple PSoC 64 MCU code examples that will help you understand how to create your own PSoC 6 MCU projects.
- The Appendix on page 30 provides a detailed hardware description, methods to use the onboard NOR Flash, kit schematics, and the bill of materials (BOM).

1.4 Board Overview

The PSoC 64S1 SecureBoot Prototyping Board has the following features:

- CY8CPROTO-064S1-SB that contains PSoC 64 MCU
- 128-Mbit external Quad SPI NOR Flash that provides a fast, expandable memory for data and code
- KitProg3 onboard SWD programmer/debugger, USB-UART and USB-I2C bridge functionality
- A Micro-B connector for USB device interface
- 1.8 V, 2.5 V and 3.3 V operation of PSoC 64 MCU is supported
- Two user LEDs, a user button, and a reset button for PSoC 64 MCU
- One Mode button, one Status LED and one Power LED for KitProg3

Figure 1-1 shows the pinout of the Prototyping Board.

Figure 1-1. Prototyping Board Pinout

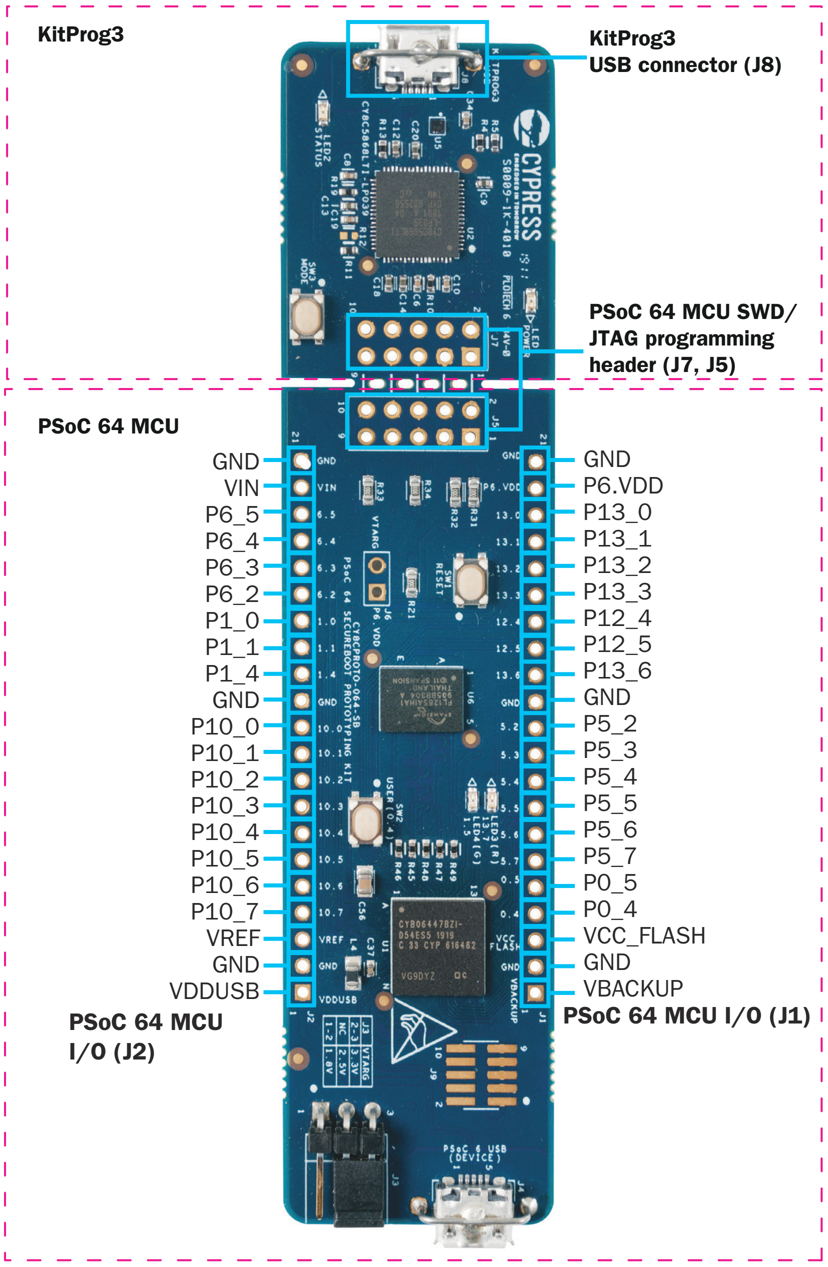


Table 1-1. Board Pinout

PSoC 6 MCU Pin	Header Mapping	Primary On-board Function	Secondary On-board Function	Connection details
P0[4]	J1.4	User Button with Hibernate wakeup capability	GPIO	Connected to ground as active LOW logic by default.
P0[5]	J1.5	GPIO	–	–
P1[0]	J2.15	GPIO	–	–
P1[1]	J2.14	GPIO	–	–
P1[4]	J2.13	GPIO	–	–
P1[5]	–	Green User LED (LED4)	–	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[2]	J1.11	GPIO	–	–
P5[3]	J1.10	GPIO	–	–
P5[4]	J1.9	GPIO	–	–
P5[5]	J1.8	GPIO	–	–
P5[6]	J1.7	GPIO	–	–
P5[7]	J1.6	GPIO	–	–
P6[0]	–	I2C_SCL	GPIO	Remove R31 to disconnect from KitProg3 I2C_SCL
P6[1]	–	I2C_SDA	GPIO	Remove R32 to disconnect from KitProg3 I2C_SDA
P6[2]	J2.16	GPIO	–	–
P6[3]	J2.17	GPIO	–	–
P6[4]	J2.18	GPIO	TDO_SWO	–
P6[5]	J2.19	GPIO	TDI	–
P6[6]	–	SWDIO	GPIO	–
P6[7]	–	SWDCLK	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	–	–
P10[6]	J2.5	GPIO	–	–
P10[7]	J2.4	GPIO	–	–
P11[0]	–	QSPI Flash Reset	–	–

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

PSoC 6 MCU Pin	Header Mapping	Primary On-board Function	Secondary On-board Function	Connection details
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[4]	J1.15	GPIO	–	–
P12[5]	J1.14	GPIO	–	–
P13[0]	J1.19	GPIO	–	–
P13[1]	J1.18	GPIO	–	–
P13[2]	J1.17	GPIO	–	–
P13[3]	J1.16	GPIO	–	–
P13[6]	J1.13	GPIO	–	–
P13[7]	–	Red User LED (LED3)	GPIO	Connected in active LOW configuration
USB.DP	–	–	–	–
USB.DM	–	–	–	–

1.5 Additional Learning Resources

Cypress provides a wealth of data at www.cypress.com/psoc6 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, visit Cypress Support or contact customer support at +1(800) 541-4736 Ext. 3 (in the USA) or +1 (408) 943-2600 Ext. 3 (International).

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: `C:\...cd\icc\`
Italics	Displays file names and reference documentation: Read about the *sourcefile.hex* file in the *PSoC Creator User Guide*.
[**Bracketed, Bold**]	Displays keyboard commands in procedures: [**Enter**] or [**Ctrl**] [**C**]
File > Open	Represents menu paths: File > Open > New Project
Bold	Displays commands, menu paths, and icon names in procedures: Click the **File** icon and then click **Open**.
Times New Roman	Displays an equation: $2 + 2 = 4$
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
API	Application Programming Interface
BOM	Bill of Materials
CM	Contract Manufacturer
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSD	CapSense Sigma Delta
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
I^2S	Inter-IC Sound

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

Acronym	Definition
IC	Integrated Circuit
IDAC	Current Digital-to-Analog Converter
IDE	Integrated Development Environment
IMO	Internal Main Oscillator
IoT	Internet of Things
JTAG	Joint Test Action Group
JWT	Java Web Token
LDO	Low Dropout Regulator
LED	Light-emitting Diode
LPO	Low Power Oscillator
MCU	Micro-Controller Unit
PC	Personal Computer
PCM	Pulse Code Modulation
PDL	Peripheral Driver Library
PDM	Pulse Density Modulation
PSoC	Programmable System-on-Chip
PWM	Pulse Width Modulation
QSPI	Quad Serial Peripheral Interface
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-064S1-SB Secure Boot Kit. This includes the Mbed OS on which the projects will be built and used for programming.

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.
2. Plug in your CY8CPROTO-064S1-SB from the Kitprog3 USB header (J8) to your PC with the provided USB cable.
3. Install Mbed OS from the following page and follow the instructions on it.
   https://os.mbed.com/docs/mbed-os/v5.13/tools/installation-and-setup.html
4. Install the pre-requisites of the Secure Boot SDK.
   a. Install Python 3.7.4 (or later) on your computer. You can download it from https://www.python.org/downloads/.
   b. The default Python37 install directory is *C:/User/<username>/AppData/Local/Programs/Python/Python37-32/python.exe*.
   You can change this to C:/Python37 during installation.
   c. Add Python to your path. For example, for a Windows system, add the python.exe file location to the system variable "Path". For example: *C:\Python37\python.exe*
   d. Similarly, add the Python home folder */Scripts* subfolder your path. For example, for a Windows system, add the Python home folder/Scripts to the "Path" system variable.
   For example: *C:\Python37\Scripts*
   e. If you have any older Python installations, ensure that Python37 in the path above any older installations such as Python27 (or remove the older version from the path).
5. Update mbed-cli by running the following command on a command line.

```
pip install -U mbed-cli
```

6. Install Secure Boot SDK package by running the following command.

```
pip install cysecuretools
```

During installation, there can be possible errors when installing colorama, protobuf and jsonschema. These can be safely ignored.

3. Kit Operation

This chapter introduces you to various features of the CY8CPROTO-064S1-SB PSoC 64S1 SecureBoot Prototyping Kit, including the theory of operation and the onboard programming and debugging functionality.

3.1 Theory of Operation

The CY8CPROTO-064S1-SB PSoC 64S1 SecureBoot Prototyping Kit is built around the PSoC 64 MCU. Figure 3-1 shows the block diagram of the PSoC 64 MCU device. For details of device features, see the device datasheet.

Figure 3-1. PSoC 6 MCU Block Diagram

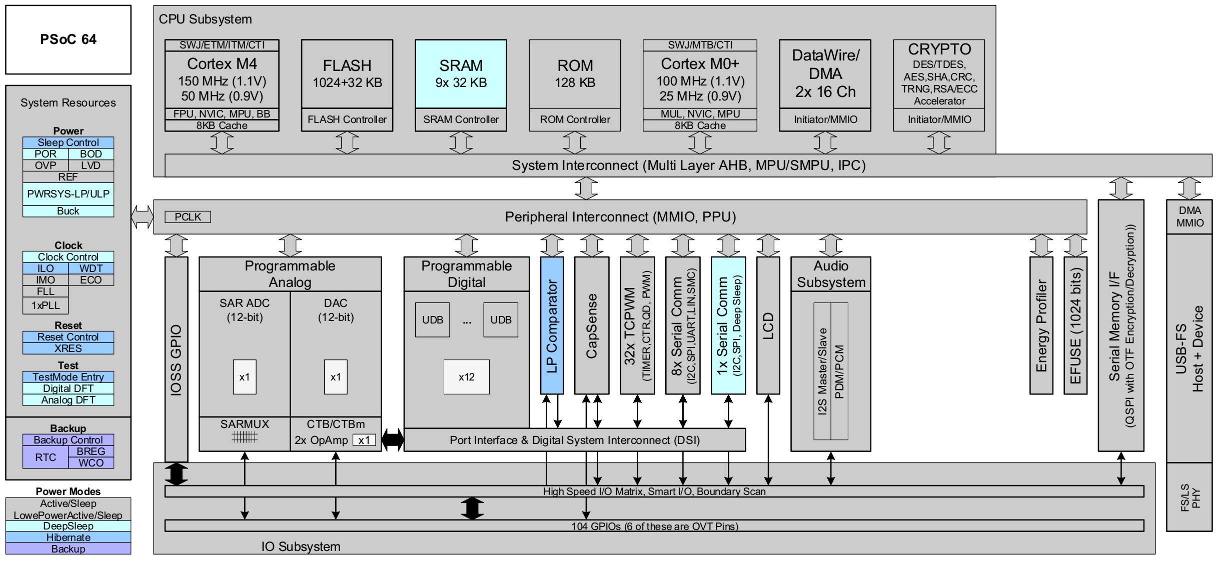
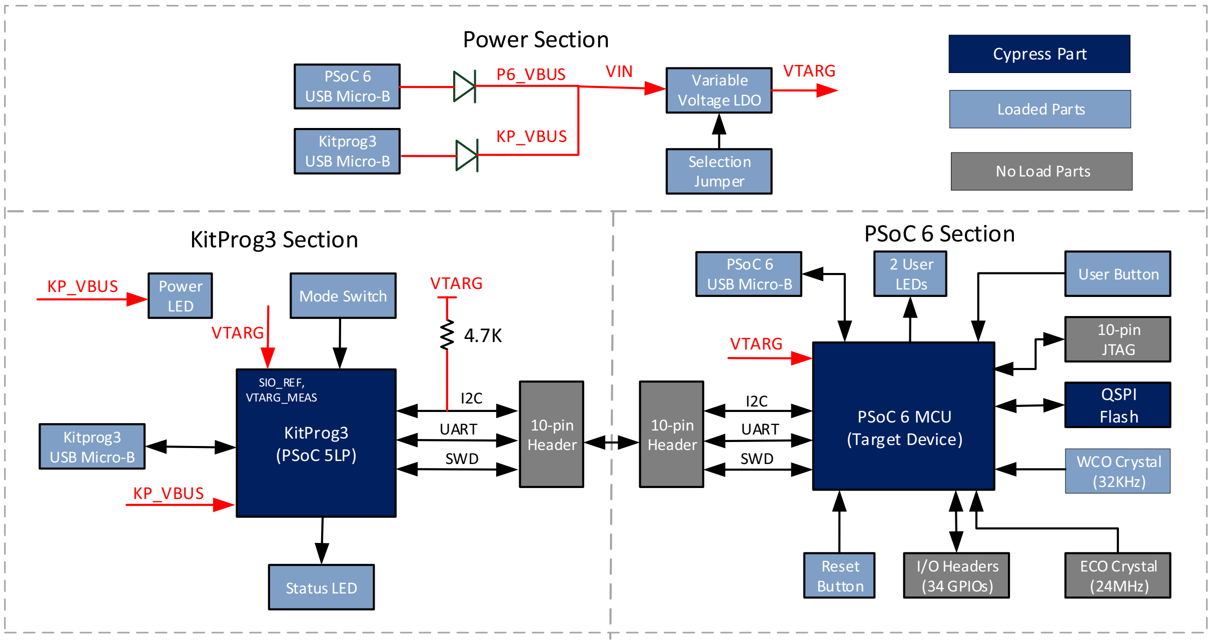
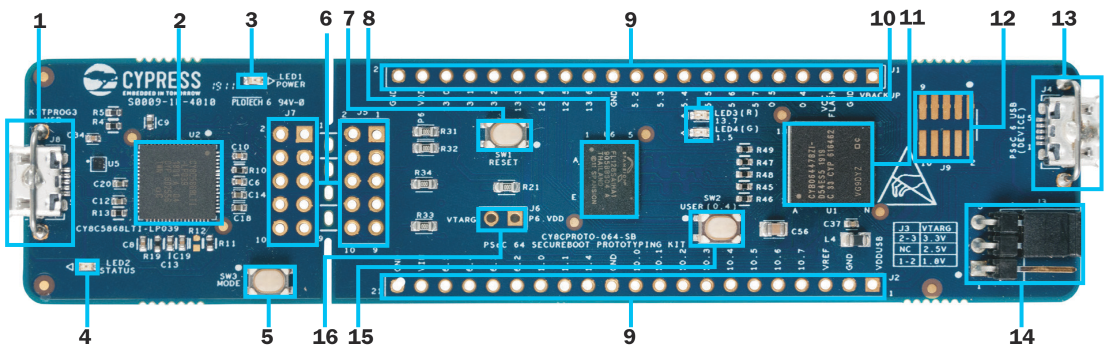


Figure 3-2. Block Diagram of Prototyping Board



Refer to KitProg3 on page 20 and Hardware Functional Description on page 30 for more details on these sections.

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



1. KitProg3 USB connector (J8)	9. PSoC 64 MCU I/O headers (J1, J2)
2. KitProg3 (PSoC 5LP) programmer and debugger (CY8C5868LTI-LP039, U2)	10. PSoC 64 MCU user LEDs (LED3, LED4)
3. Power LED (LED1)	11. PSoC 64 MCU (CYB06447BZI-D54, U1)
4. KitProg3 status LED (LED2)	12. PSoC 64 MCU program and debug header (J9)
5. KitProg3 programming mode selection button (SW3)	13. PSoC 64 USB device connector (J4)
6. KitProg3 10-pin interface header (J7, J5)	14. Power selection jumper (J3)
7. Reset button (SW1)	15. PSoC 64 MCU user button (SW2)
8. Cypress 128-Mbit serial NOR flash memory (S25FL128S, U6)	16. PSoC 64 MCU current measurement header (J6)

The PSoC 64 Secure Boot Kit 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. KitProg3 also supports custom applications. 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 (CMSIS-DAP/Bulk, CMSIS-DAP/HID mode and DAPLink mode). For more details, see the KitProg3 User Guide.
6. **KitProg3 interface header (J7, J5):** This header brings out the SWD, USB-UART and USB-I2C interface of the KitProg3. This is used to program and debug the PSoC 64 MCU. If KitProg3 section is broken away, it can be used to program any 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 MCU section of the board. For more details on the KitProg3, see the KitProg3 User Guide.
7. **PSoC 64 MCU reset button (SW1):** This button is used to reset the PSoC 64 MCU. This button connects the PSoC 64 MCU reset (XRES) pin to ground.

8. **Cypress 128-Mbit serial NOR flash memory (S25FL128, U6):** The S25FL128SAGBHIA10 NOR flash of 128-Mbit capacity is connected to the serial memory interface (SMIF) of the PSoC 64 MCU. The NOR device can be used for both data and code memory with execute-in-place (XIP) support and encryption.
9. **PSoC 64 MCU I/O header (J1, J2):** These headers provide connectivity to PSoC 64 MCU GPIOs. Most of these I/Os are also connected to on-board peripherals.
10. **PSoC 64 MCU user LEDs (LED3, LED4):** The red LED (LED3) can operate at the entire operating voltage range of the PSoC 64 MCU, 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 MCU (CYB06447BZI-D54, U1):** CYB06447BZI-D54 is a PSoC 64 Secure Boot MCU with 896 KB of flash and 184 KB of SRAM. The chip features out-of-the-box security functionality, providing an isolated root-of-trust with true attestation and provisioning services.
12. **PSoC 64 MCU program and debug header (J9):** This 10-pin header allows you to program and debug the PSoC 64 MCU using an external programmer such as MiniProg4. Note that this is not loaded by default.
13. **PSoC 6 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 MCU USB device applications.
14. **System Power selection jumper (J3):** This switch is used to select the PSoC 64 MCU's supply voltage (P6.VDD) between three voltages: 1.8 V, 2.5 V, and 3.3 V.
15. **PSoC 64 MCU user button (SW2):** This button can be used to provide an input to the PSoC 64 MCU. Note that by default, the button connects the PSoC 64 MCU pin to ground when pressed, so you need to configure the PSoC 64 MCU pin as a digital input with resistive pull-up for detecting the button press. This button also provides a wake-up source from low-power modes of the device.
16. **PSoC 64 MCU Current measurement header (J6):** This header can be used to measure the current consumption of PSoC 6 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 30 for details on various hardware blocks.

3.2 KitProg3

The PSoC 64 Secure Boot Kit can be programmed and debugged using the onboard KitProg3. KitProg3 also has USB-UART and USB-I2C functionality. KitProg3 supports CMSIS-DAP and DAPLink 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.

Before programming the device, ensure that Cypress Programmer software are installed on the computer. See the link https://www.cypress.com/products/psoc-programming-solutions for more information.

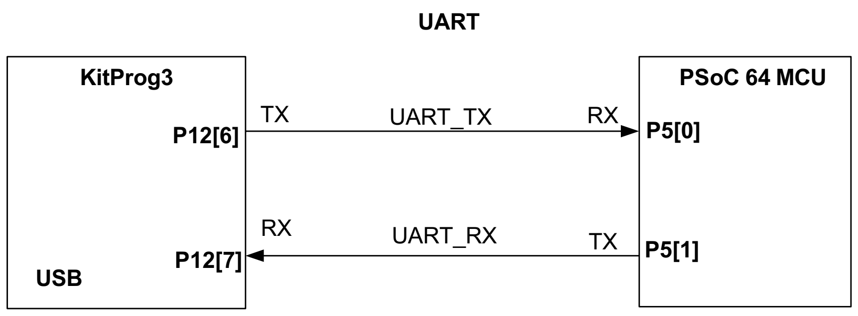
3.2.1 Programming and Debugging

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 HID mode, CMSIS-DAP Bulk mode (default) or DAPLink mode. Programming is faster with the Bulk mode. The status LED (Amber) is always ON in Bulk mode, is ramping ON/OFF at 1-Hz rate in HID mode and ramping ON/OFF at 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

KitProg3 on the board can act as a USB-UART bridge. The UART lines between the PSoC 64 MCU 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.

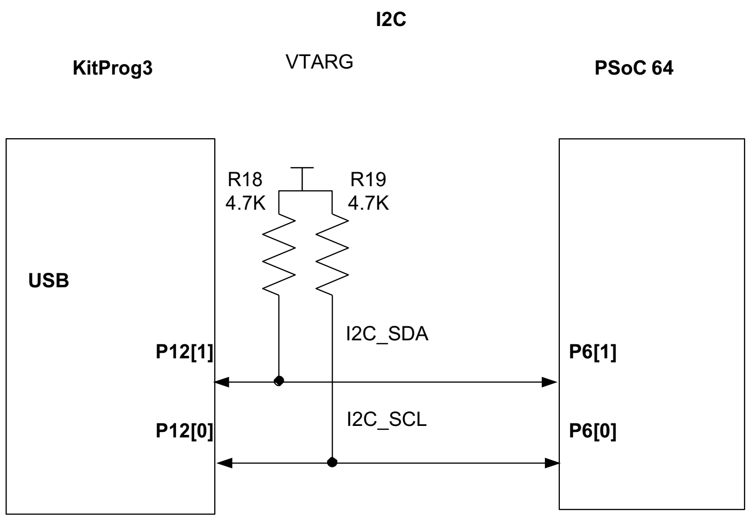
Figure 3-4. UART Connection between KitProg3 and PSoC 64 MCU



3.2.3 USB-I2C Bridge

The KitProg3 can function as a USB-I2C bridge and communicate with the Bridge Control Panel (BCP) software. The I2C lines on the PSoC 64 MCU 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 I2C 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 MCU



4. Running Code on PSoC64 Secure MCUs

The CY8CPROTO-064S1-SB PSoC 64S1 SecureBoot Prototyping Kit can run code examples available on Mbed OS. However, prior to running any code on the PSoC 64 line of secure 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 and run a signed version of *mbed-os-example-blinky* code.

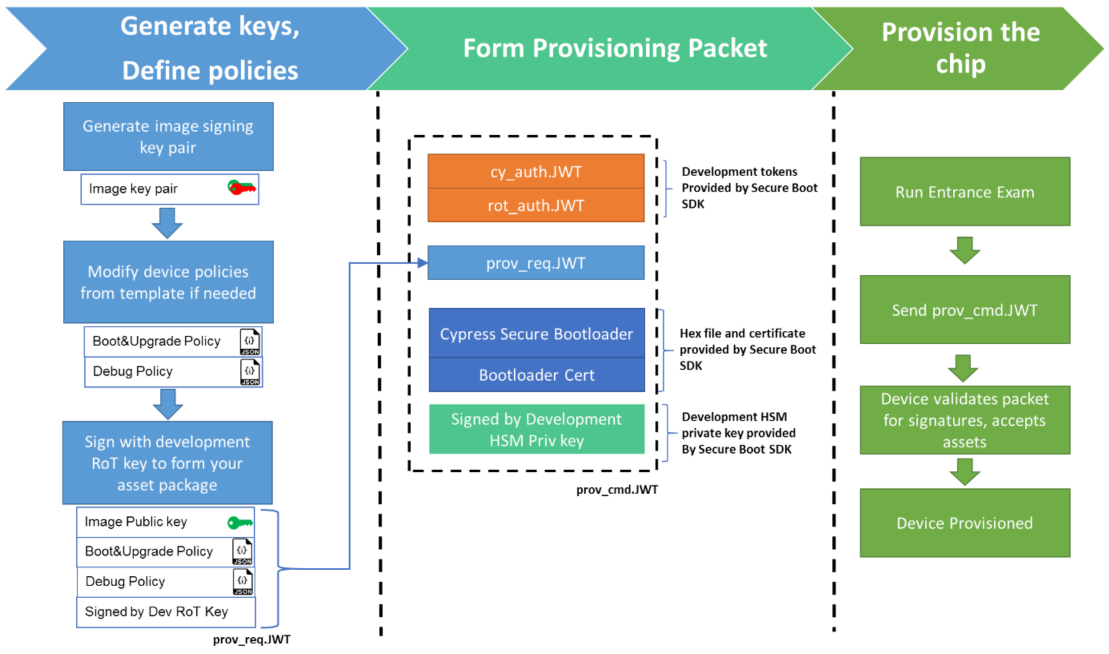
4.1 Provisioning Overview

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


Generate keys, Define policies
Form Provisioning Packet
Provision the chip
Generate image signing key pair
Image key pair
Modify device policies from template if needed
Boot&Upgrade Policy
Debug Policy
Sign with development RoT key to form your asset package
Image Public key
Boot&Upgrade Policy
Debug Policy
Signed by Dev RoT Key
prov_req.JWT
cy_auth.JWT
rot_auth.JWT
Development tokens Provided by Secure Boot SDK
prov_req.JWT
Cypress Secure Bootloader
Bootloader Cert
Hex file and certificate provided by Secure Boot SDK
Signed by Development HSM Priv key
Development HSM private key provided By Secure Boot SDK
prov_cmd.JWT
Run Entrance Exam
Send prov_cmd.JWT
Device validates packet for signatures, accepts assets
Device Provisioned

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

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

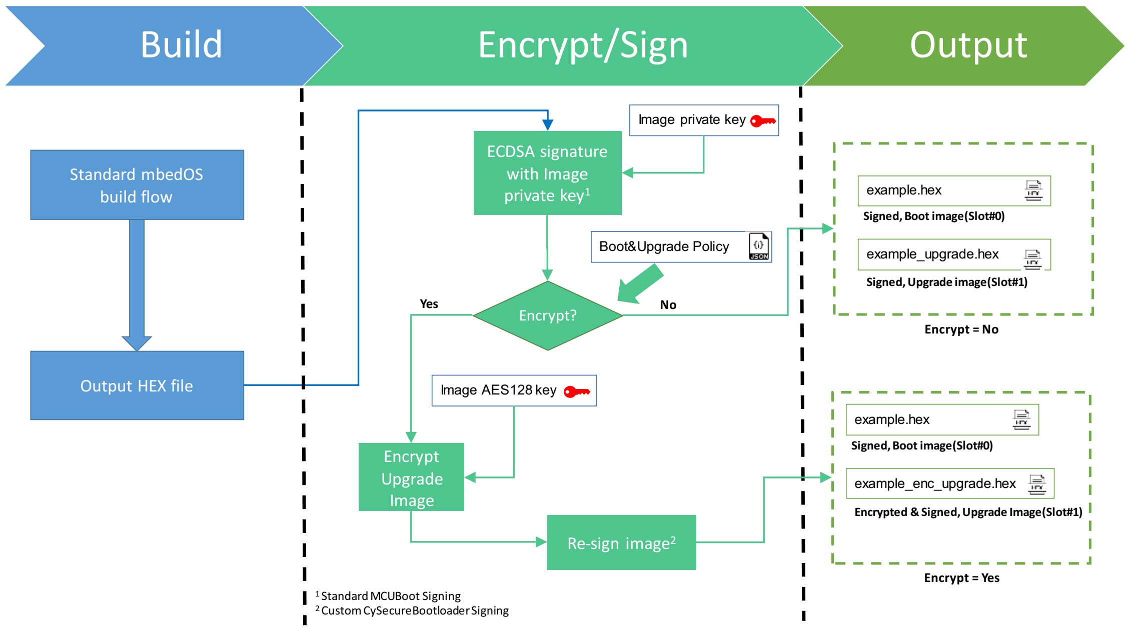
In addition, the SDK provides tools to do the following:

1. Generate image keys
2. Form provisioning packets
3. Scripts to run 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-064S1-SB target using mbed-cli is shown below.

Figure 4-2. Build and Encrypt/Signing Flow


Build
Encrypt/Sign
Output
Standard mbedOS build flow
Output HEX file
Image private key
ECDSA signature with Image private key¹
Boot&Upgrade Policy
Encrypt?
Yes
No
Image AES128 key
Encrypt Upgrade Image
Re-sign image²
example.hex
Signed, Boot image(Slot#0)
example_upgrade.hex
Signed, Upgrade image(Slot#1)
Encrypt = No
example.hex
Signed, Boot image(Slot#0)
example_enc_upgrade.hex
Encrypted & Signed, Upgrade Image(Slot#1)
Encrypt = Yes
¹ Standard MCUBoot Signing
² Custom CySecureBootloader Signing

4.2 Import MbedOS example

1. From your command-line, import the blinky example project using the following command:

```
mbed import mbed-os-example-blinky
```

2. Navigate into the code example's mbed-os folder by typing the following commands:

```
cd mbed-os-example-blinky
```

3. Check the version of your Mbed OS library by typing the following command:

```
mbed ls
```

4. If the Mbed-OS version is lower than 5.14, upgrade it by typing the following command:

```
mbed update mbed-os-5.14
```

4.3 Provision the Device

1. **Navigate to the following folder in your command window:**

   When you imported your example, the root folder is mbed-os-example-blinky. The root folder has a subfolder titled mbed-os, in the following steps <mbed-os> refers to the above subfolder.

   <mbed-os>\targets\TARGET_Cypress\TARGET_PSOC6\sb-tools\

2. **Prepare your local mbed workspace.**

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 port open to reprogram the chip. This step makes a copy of the policy local to your mbed-os-example-blinky workspace so you can modify it if needed.

Run the following command:

```
*Copy default policy from cysecuretools into mbed target*
```

```
python -c "import os; import shutil; import cysecuretools;
os.makedirs('policy') if not os.path.exists('policy') else None;
shutil.copy2(os.path.join(os.path.dirname(cysecuretools.__file__),
'targets/cy8cproto_064s1_sb/policy/policy_single_stage_CM4.json'),
'./policy/policy_single_stage_CM4.json')"
```

(Or)

You can manually replace the policy file from your local python cysecuretools installation into the *sb-tools\policy* folder.

Note:

You can also choose to copy the policy_single_stage_CM4_smif.JSON to enable external memory. If you choose this alternate policy, ensure that the secure_image_parameters.json file is updated with the correct policy and key names.

This file is located at the below folder,

<mbed-os>\targets\TARGET_Cypress\TARGET_PSOC6\TARGET_CY8CPROTO_064_SB\

3. **Create new keys.**

What does this step do?

The CySecureTools looks at the provided policy, which specifies how many keys are needed to provision the chip. By default, only one key pair is generated under the /keys/ folder with the name USERAPP_CM4_KEY.

If ‘encrypt’ is enabled, then an additional 128-bit AES key is generated which can be used to encrypt the application image. The ‘encrypt’ field is present in the policy_single_stage_CM4.json file, specifically in the same fields containing “id”: 4. For a full description of all fields, please refer to the Secure Boot SDK User Guide.

Run the following command:

```
*Create keys, using specified policy:*
```

```
python -c "from cysecuretools import CySecureTools;tools =
CySecureTools('CY8CPROTO-064S1-SB', 'policy/
policy_single_stage_CM4.json'); tools.create_keys();"
```

4. **Create provisioning packet.**

What does this step do?

The CySecureTools looks at the provided policy, forms a prov_cmd.jwt with the inputs provided in the policy. The output JWT packet is what will be sent to the PSoC64 Secure MCU for completing provisioning.

For exact details on what the all the policy fields mean, please refer to the Secure Boot SDK User Guide.

Run the following command:

```
*Create provisioning packets, using specified policy:*
```

```
python -c "from cysecuretools import CySecureTools; tools =
CySecureTools('CY8CPROTO-064S1-SB', 'policy/
policy_single_stage_CM4.json'); tools.create_provisioning_packet();"
```

5. **Connect your CY8CPROTO-064S1-SB kit and run the entrance exam.**

   ATTENTION: Remove Jumper J3 to supply the chip with 2.5V and plug in the kit to the PC. The 2.5V supply is necessary for Step 6, where PSoC64 eFuses are blown. KitProg3 must be in CMSIS-DAP HID mode for provisioning. The Status LED (LED2) will be ramping ON/OFF slowly (~1Hz) in this mode. Press and release the Mode button (SW3) one or more times until the KIt-Prog3 is in CMSIS-DAP mode.

What does this step do?

The Entrance exam is a test routine which does 3 things:

a. Verify if Device is in SECURE UNCLAIMED mode
b. Verify if FlashBoot has not been modified/tampered
c. Verify if User flash is empty and no code is running before any provisioning takes place

Failing the entrance exam will return an error in the command line. If there is any firmware running on the device, it can be erased using tools like the Cypress Programmer. mbed-cli does not natively provide any commands to erase the chip.

Note that this step is performed in Step 6, when you provision the chip as well so you can choose to skip this step if needed.

Run the below command:

```
*Run entrance exam *
```

```
python -c "from cysecuretools import CySecureTools; tools =
CySecureTools('CY8CPROTO-064S1-SB', 'policy/
policy_single_stage_CM4.json'); tools.entrance_exam();"
```

6. **Perform provisioning.**

   ATTENTION: PSoC 6 supply voltage of 2.5V is required to perform provisioning. The 2.5V requirement is because this step involves blowing eFuses to change the device lifecycle to SECURE CLAIMED. If the chip is not at 2.5V, it may cause provisioning to fail and permanently lock the chip in a dead state.

What does this step do?

This step sends the prov_cmd.JWT to the PSoC64 to finish provisioning

Run the below command:

```
*Provision chip, using specified policy:*
```

```
python -c "from cysecuretools import CySecureTools; tools =
CySecureTools('CY8CPROTO-064S1-SB', 'policy/
policy_single_stage_CM4.json'); tools.provision_device();"
```

4.4 Build and Program the Mbed OS Example Project

1. In order to build and program the project, move the kit to DAPlink mode by pressing and releasing the mode button (**SW3**) one or more times until the Status LED (LED2) is ramping ON/OFF at a fast (~2Hz) rate.
2. Navigate back to your *mbed-os-example-blinky* folder.
3. Build and program the application using the following command (Note: the -f option programs the device after the build completes. Omit that option if you want to build without programming.):

```
mbed compile -m CY8CPROTO_064_SB -t GCC_ARM -f
```

Note 1:

When using the target CY8CPROTO_064_SB, Mbed OS has a post-build signing script which uses the following key to sign the output image:

<mbed-os>\targets\TARGET_Cypress\TARGET_PSOC6\sb-tools\keys\USERAPP_CM4_KEY_PRIV.pem

Ensure that you are using the **same** key that you used to provision the device; otherwise PSoC64 Secure Boot will fail. Success or Failure of the Secure Boot process can be observed by opening the Kitprog3 COM port with a terminal application of your choice (e.g., Putty/TeraTerm) and setting the baud rate to 115200.

Note 2:

If you get the error, “Target board you compiled for is not connected to your system” run the following command to add the kit and then try programming again:

```
mbedls --mock 1907:CY8CPROTO_064_SB
```

4. If the PSoC64 is successfully programmed, observe the Red LED blinking at 1 Hz. In addition, mbedOS stats are printed every 10 seconds over the Kitprog3 UART at 9600 baud.

4.5 Additional Code Examples

Additional code examples for PSoC64 can be found on the Cypress git repository

https://github.com/cypresssemiconductorco

A. Appendix

A.1 Schematics

Refer to the schematic files available on kit webpage.

A.2 Hardware Functional Description

This section explains in detail the individual hardware blocks of the PSoC 64S1 Secure Boot Prototyping Board.

A.2.1 CYB06447BZI-D54 (U1)

PSoC 6 MCU is a high-performance, ultra-low-power and secure MCU platform, purpose-built for IoT applications. The PSoC 64 Secure 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. The CYB06447BZI-D54 device has 896 KB of flash and 184 KB of SRAM.

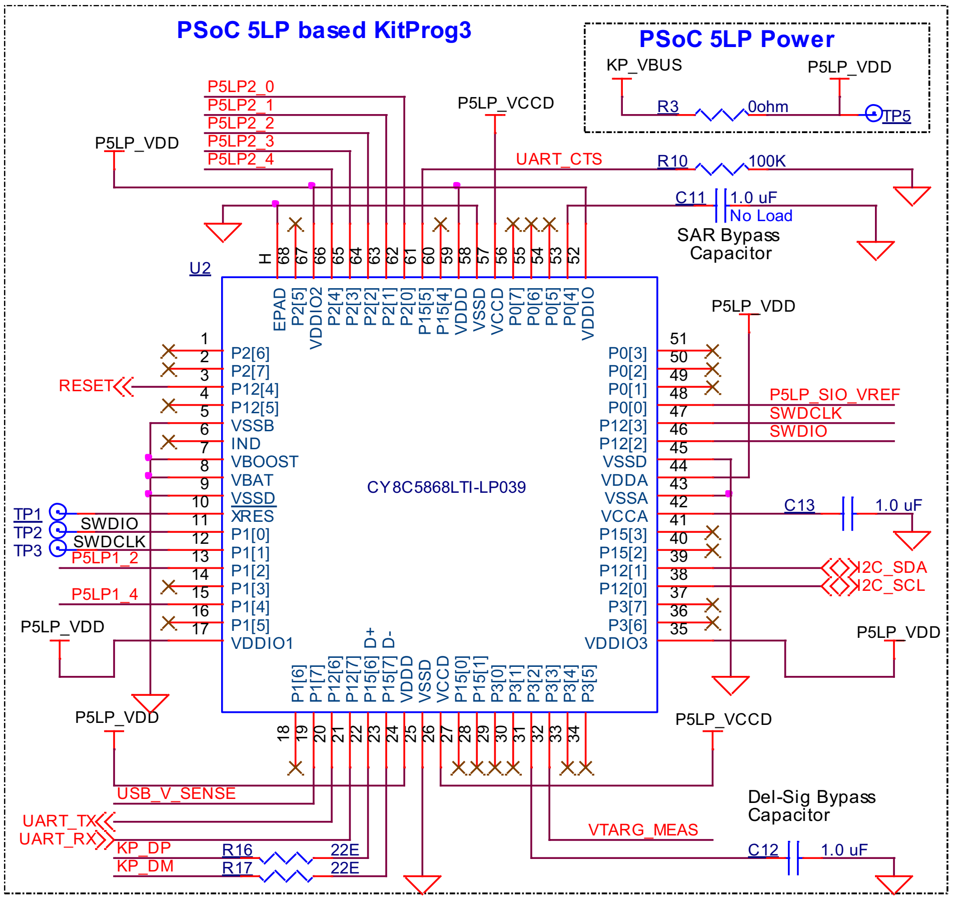
For more information, see the PSoC 6 MCU web page and the datasheet.

A.2.2 PSoC 5LP (U2)

An onboard PSoC 5LP (CY8C5868LTI-LP039) device is used as KitProg3 to program and debug the PSoC 64 MCU. 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 PSoC 64 MCU.

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.

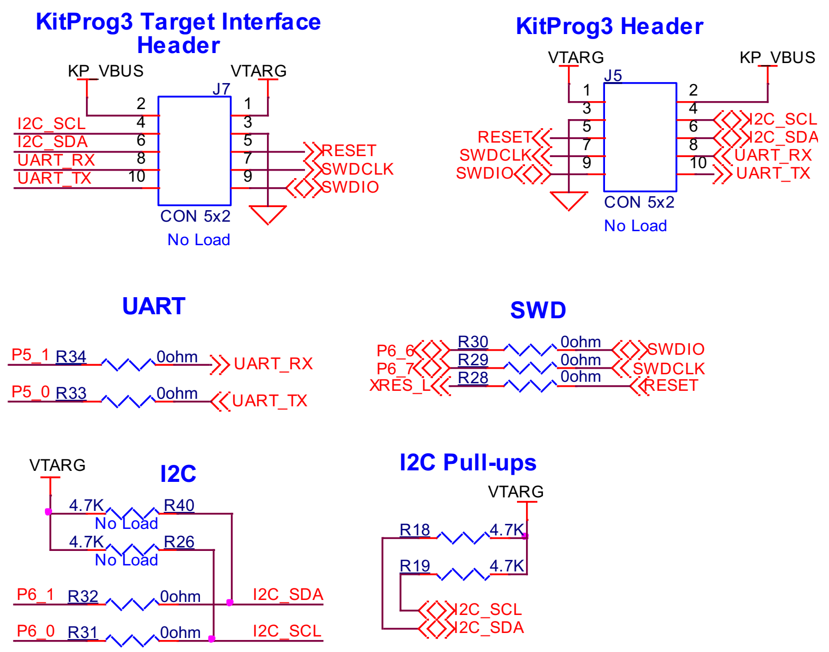
Figure A-1. Schematics of PSoC 5LP based KitProg3 and PSoC 5LP Power



A.2.3 Serial Interconnection between PSoC 5LP and PSoC 64 MCU

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 A-2. The USB-Serial pins of the PSoC 5LP device are hard-wired to the I2C/UART pins of the PSoC 64 MCU. These pins are on J7 on the KitProg3 section and on J5 on the PSoC 64 MCU section.

Figure A-2. Schematics of Programming and Serial Interface Connections



A.2.4 Power Supply System

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

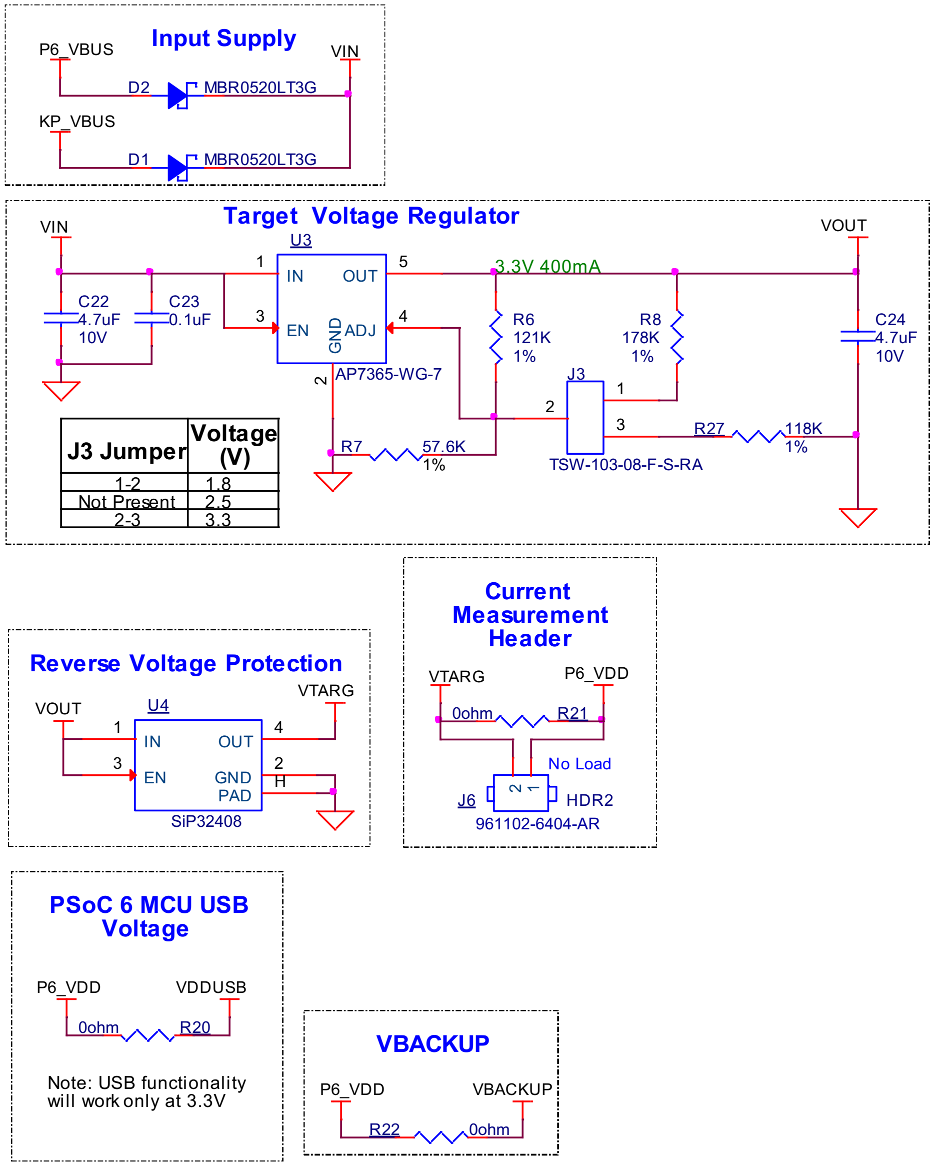
- 5 V from the onboard USB Micro-B connectors (**J4** and **J8**)
- 5 V from external power supply through VIN header **J2.20**

The power supply system is designed to support 1.8 V, 2.5 V and 3.3 V operation of the PSoC 64 MCU. 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 MCU 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 MCU and peripherals. Figure A-3 shows the schematics of the voltage regulator and power selection circuits.

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

Figure A-3. Schematics of Power Supply System



J3 Jumper	Voltage (V)
1-2	1.8
Not Present	2.5
2-3	3.3

A.2.4.1 Measure PSoC 64 MCU Current Consumption

To measure the PSoC 64 MCU 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 either with USB power 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).

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 MCU will be back-powered through the KitProg3 - PSoC 64 MCU interfaces (SWD, I2C, UART). Therefore, for the most accurate current measurements, detach the Kitpro3 section of the kit from the PSoC64 MCU section.

A.2.5 Expansion Connectors

A.2.5.1 Functionality of the J1 and J2 Headers (Target Board)

The target PSoC 64 MCU 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 MCU.

Figure A-4. J1 and J2 Headers

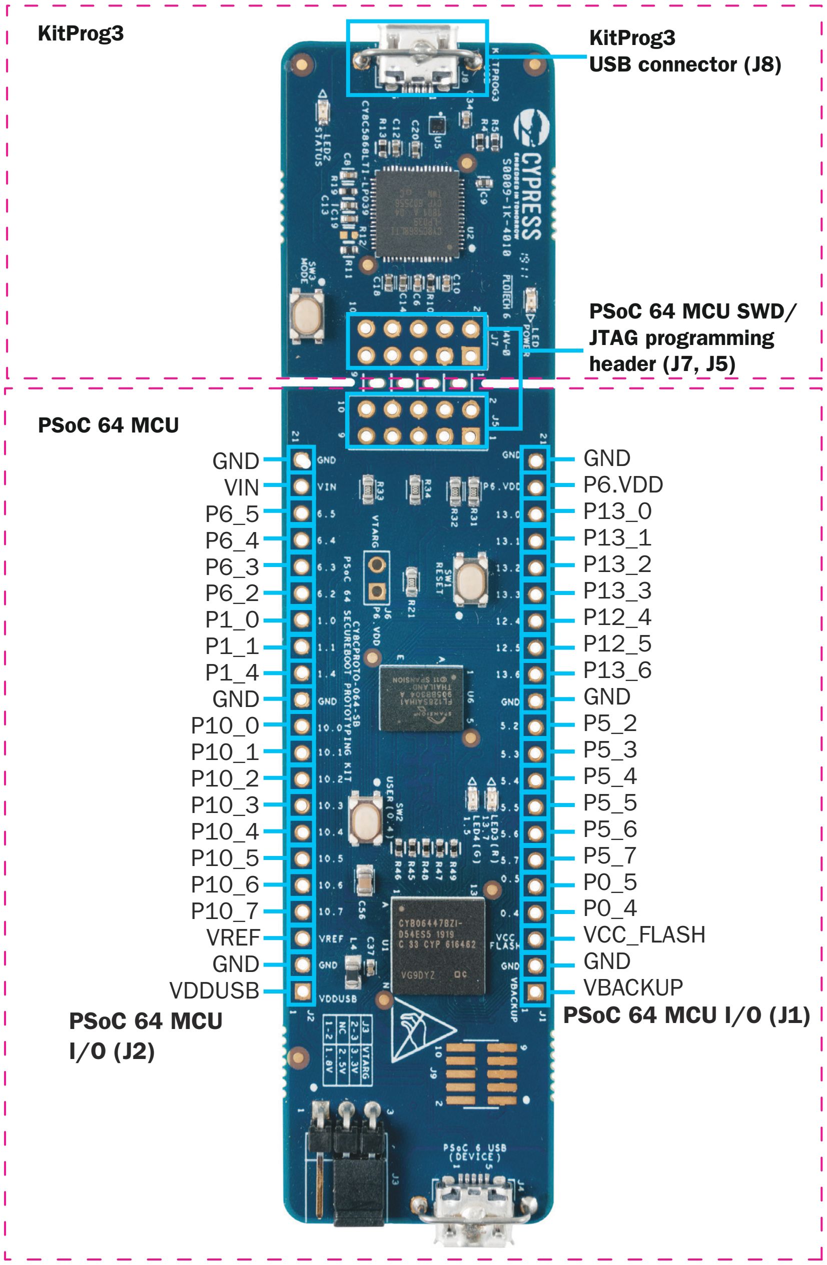


Table A-1. Pin Details of J1 and J2 Headers

PSoC 64 MCU Prototyping Board GPIO Header (J2)			PSoC 64 MCU 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	P10.7	GPIO	J1_04	P0.4	GPIO/User SW2
J2_05	P10.6	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	P5.5	GPIO
J2_09	P10.2	GPIO	J1_09	P5.4	GPIO
J2_10	P10.1	GPIO	J1_10	P5.3	GPIO
J2_11	P10.0	GPIO	J1_11	P5.2	GPIO
J2_12	GND	Ground	J1_12	GND	GPIO
J2_13	P1.4	GPIO	J1_13	P13.6	GPIO
J2_14	P1.1	GPIO	J1_14	P12.5	GPIO
J2_15	P1.0	GPIO	J1_15	P12.4	GPIO
J2_16	P6.2	GPIO	J1_16	P13.3	GPIO
J2_17	P6.3	GPIO	J1_17	P13.2	GPIO
J2_18	P6.4	GPIO	J1_18	P13.1	GPIO
J2_19	P6.5	GPIO	J1_19	P13.0	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.

A.2.5.2 *Functionality of J7 and J5 Headers (KitProg3 to PSoC 6 MCU)*

The KitProg3 and target boards each contain a 2×5-pin header. These headers provide a physical connection between the two devices. This connections contains

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

Figure A-5. J7 and J5 Headers

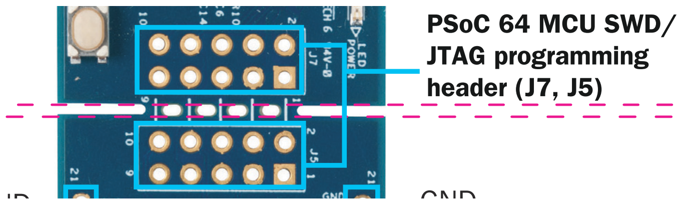


Table A-2. Pin Details of J7 Header

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 A-3. Pin Details of J5 Header

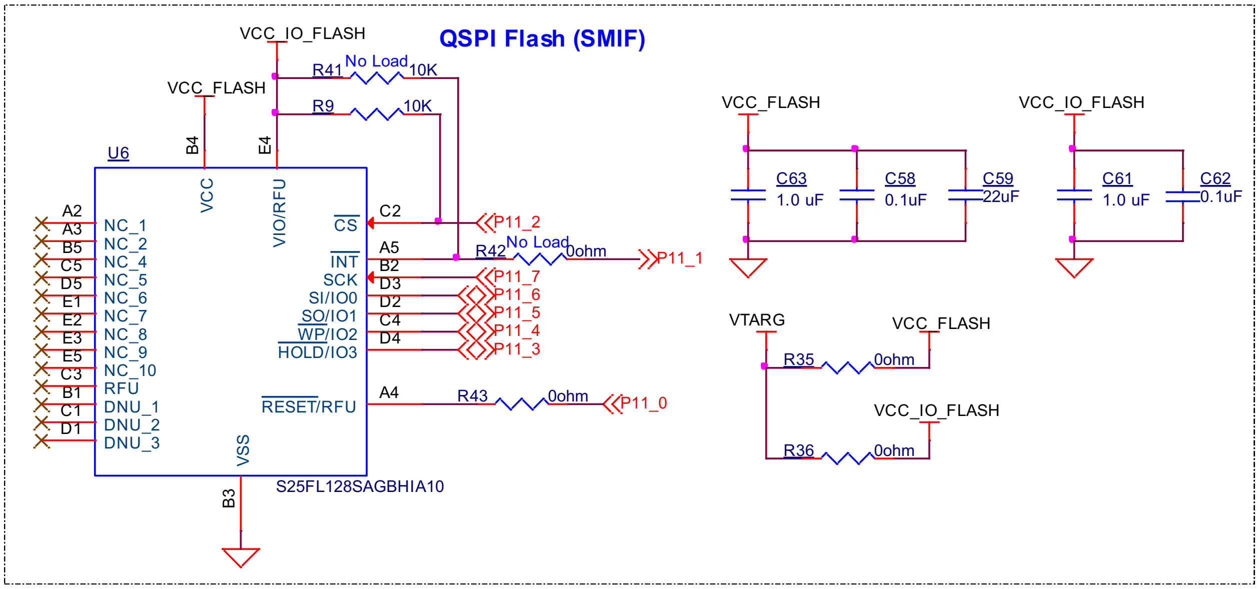
PSoC 64 MCU Header (J5)					
Pin	**Signal**	**Description**	**Pin**	**Signal**	**Description**
J5_1	VTARG	Power	J5_2	VTARG	Power
J5_3	GND	Ground	J5_4	P6[0]	I2C_SCL
J5_5	XRES_L	Reset	J5_6	P6[1]	I2C_SDA
J5_7	P6[7]	SWD_CLK	J5_8	P5[1]	UART_TX
J5_9	P6[6]	SWD_IO	J5_10	P5[0]	UART_RX

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 6 sections of the kit.

A.2.6 Quad SPI Flash

The board has a Cypress NOR Flash memory (S25FL128SAGBHIA10) of 128 Mbit capacity. The NOR Flash is connected to the serial memory interface (SMIF) of the PSoC 64 MCU. The NOR Flash device can be used for both data and code memory with execute-in-place (XIP) support and encryption.

Figure A-6. Schematics of Quad SPI Flash

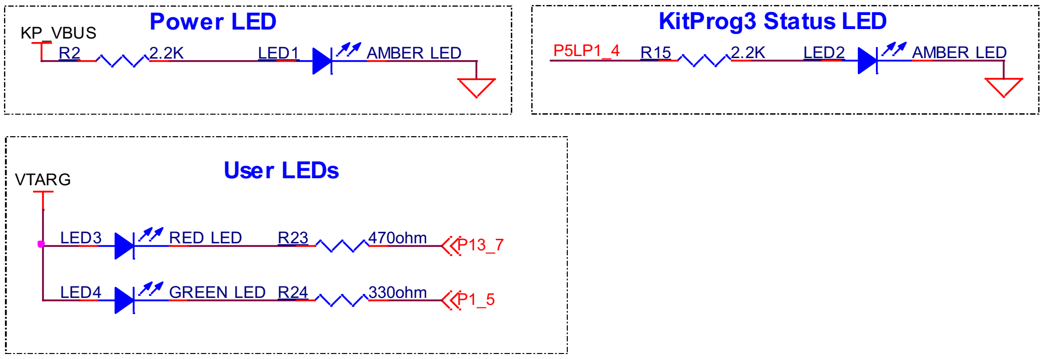


A.2.7 LEDs

LED2 (Amber) indicates the status of KitProg3 (See the KitProg3 User Guide for details). **LED1** indicates the status of power supplied to PSoC 5LP.

The board also has two user-controllable LEDs (**LED3**, **LED4**) connected to the PSoC 64 MCU pins (**P13.7**, **P1.5**) for user applications.

Figure A-7. Schematics of LEDs

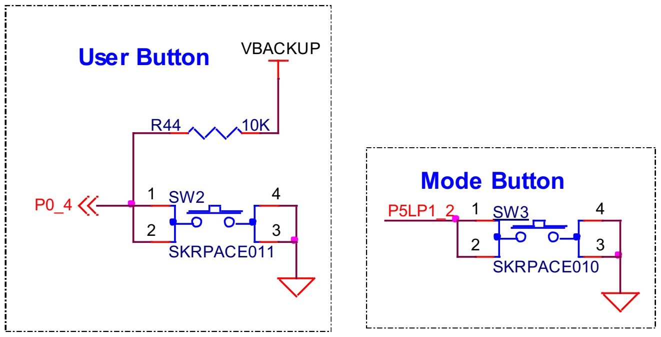


A.2.8 User Buttons

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

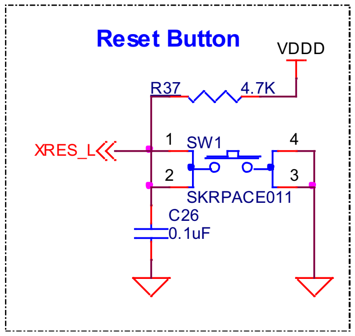
The board has a reset button and mode select button. The reset button (**SW1**) is connected to the XRES pin of the PSoC 64 MCU and is used to reset the device. The mode select button (**SW3**) is connected to the PSoC 5LP device for programming mode and custom app selection (Refer to the KitProg3 User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

Figure A-8. Schematics of Push Buttons



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

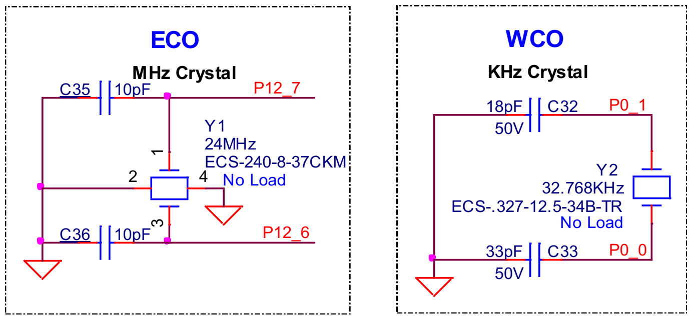
Figure A-9. Reset Button



A.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. 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.

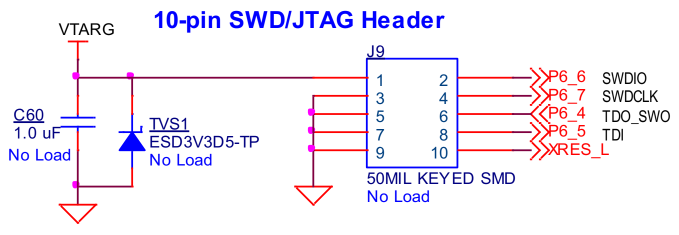
Figure A-10. Schematics of Crystal Oscillator



A.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 MCU using an external programmer such as MiniProg4. This is not loaded by default.

Figure A-11. 10-pin SWD/JTAG Header



Note: Maximum voltage on P6_VDD is 3.6V. Supplying 5V through the 10-pin header will permanently damage the device

A.3 Bill of Materials

Refer to the BOM files in the kit webpage.

A.4 Frequently Asked Questions

1. How does CY8CPROTO-064S1-SB 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 MCU 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 A-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 MCU is not meant to be operated at voltages greater than 3.6 V. Powering the PSoC 64 MCU to more than 4 V will damage the chip. You cannot drive the I/O system with > 3.6 V supply voltages.
4. I am unable to program the target device.
   a. Check **J3** to ensure that the jumper shunt is present on the board.
   b. Update your KitProg3 Version to v1.01 or later 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 **J4** header but only after the KitProg3 section is separated from the PSoC 64 MCU section of the board.

Revision History

Document Revision History

Document Title: CY8CPROTO-064S1-SB PSoC 64S1 Secure Boot Prototyping Kit User Guide Document Number: 002-28075			
Revision	**ECN Number**	**Issue Date**	**Description of Change**
**	6658229	11/05/2019	New kit guide.