esp-dev-kits

Mar 06, 2023

ESP32-C3 BOARDS

1 Supported Development Boards **3**
1.1 Related 8

[??]

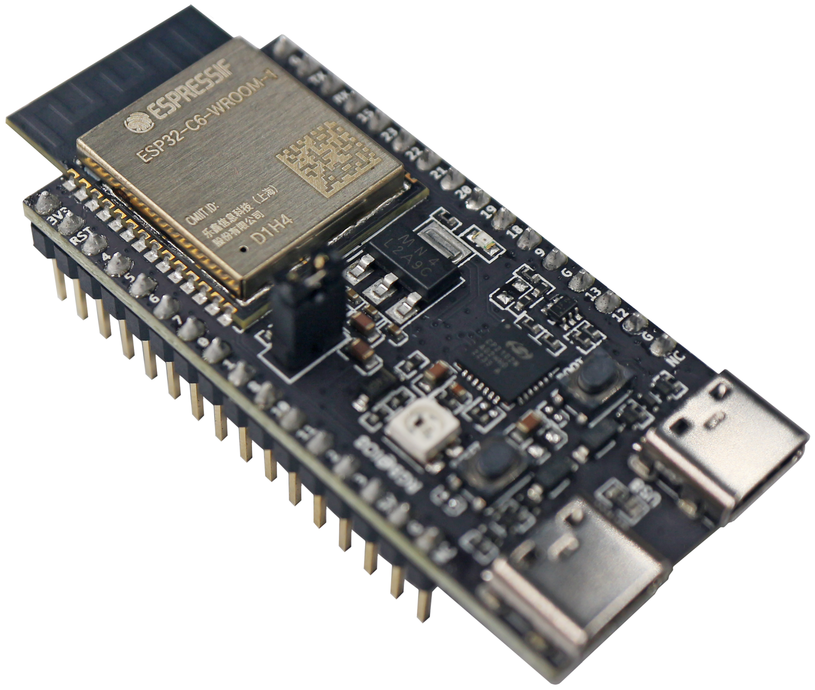
This is the documentation for esp-dev-kits.

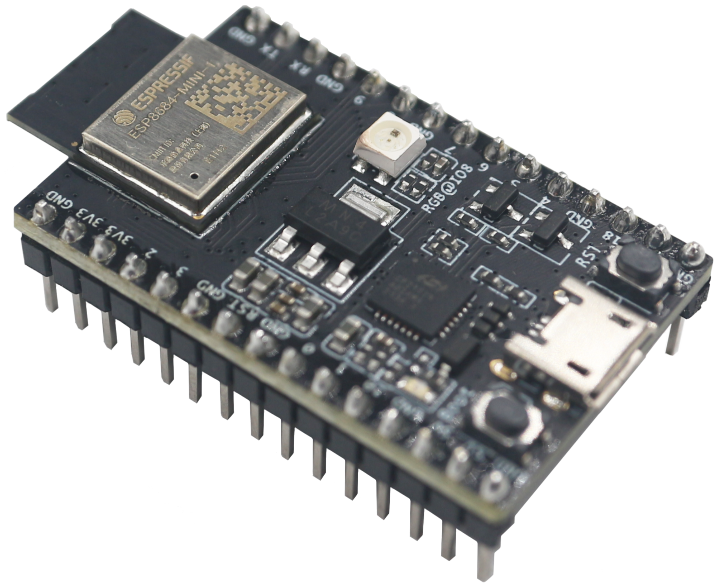
CHAPTER

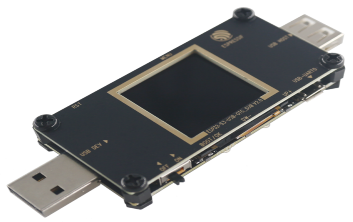
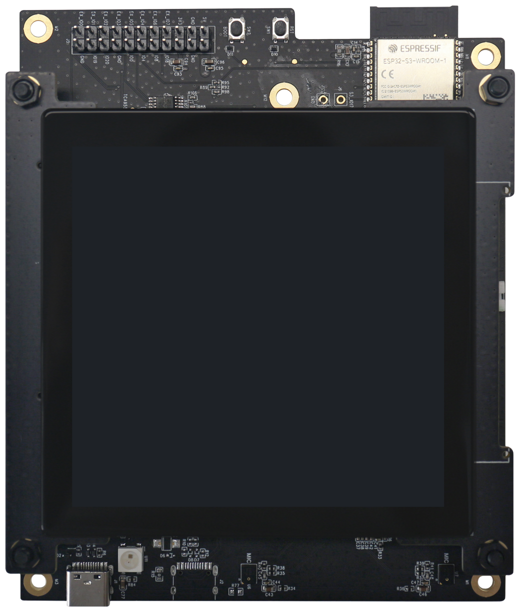
ONE

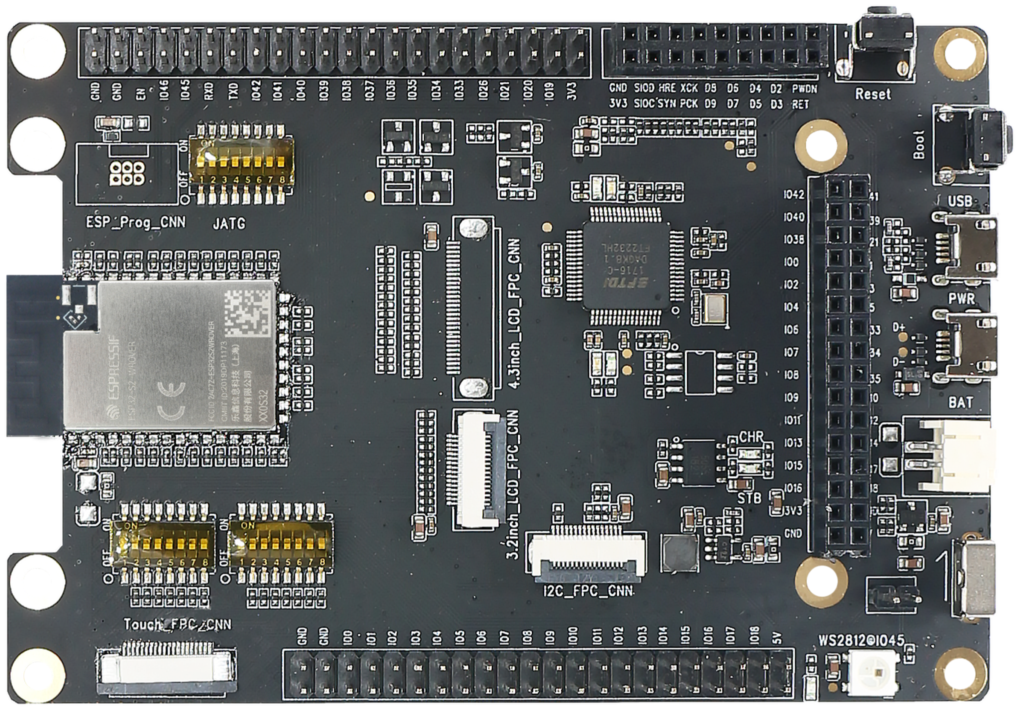
SUPPORTED DEVELOPMENT BOARDS

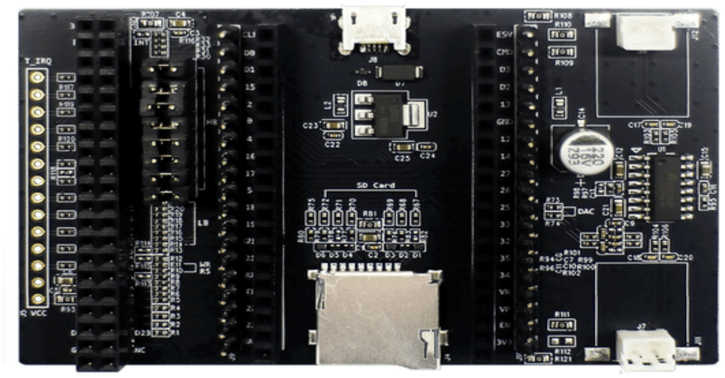
ESP32-C3 Development Boards	
ESP32-C3-LCD-EV-BOARD	

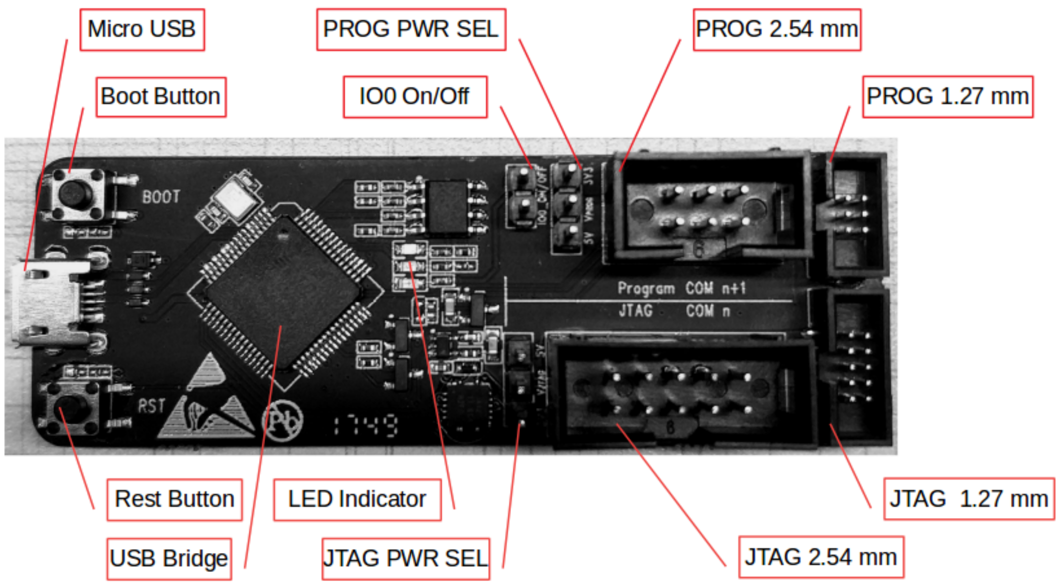
ESP32-C6 Development Boards	
	
ESP32-C6-DevKitC-1	

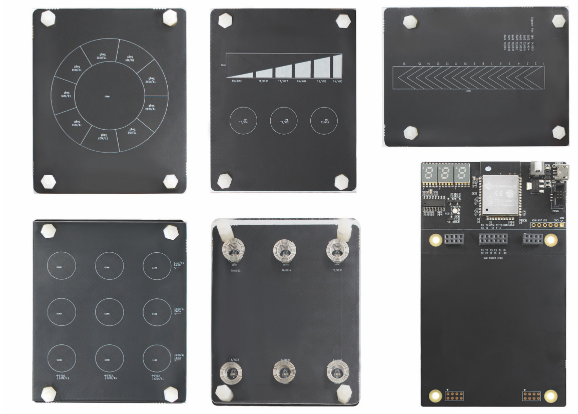
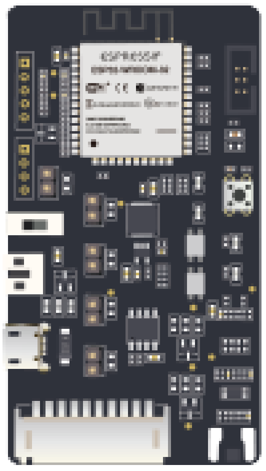
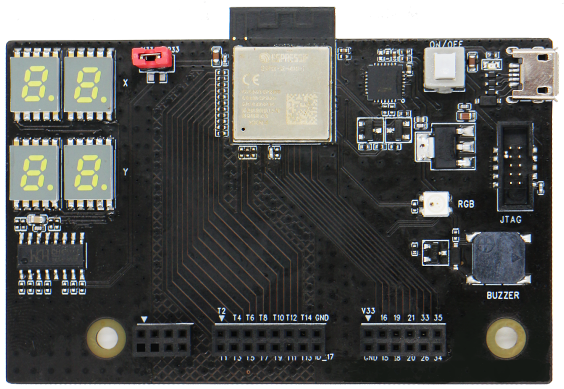
ESP8684 Development Boards	
	
ESP8684-DevKitM-1	

ESP32-S3 Development Boards	
	
ESP32-S3-USB-OTG	ESP32-S3-LCD-EV-BOARD

ESP32-S2 Development Boards	
	
ESP32-S2-Kaluga-1	

ESP32 Development Boards	
	
ESP32-LCDKit	

Other Boards	
 Micro USB PROG PWR SEL PROG 2.54 mm Boot Button IO0 On/Off PROG 1.27 mm Rest Button LED Indicator JTAG 1.27 mm USB Bridge JTAG PWR SEL JTAG 2.54 mm	
ESP-Prog	

EOL (End of Life) Boards	
	
ESP32-S2-HMI-DevKit-1	ESP32-Sense-Kit
	
ESP32-MeshKit-Sense	ESP32-S2-Touch-DevKit-1

1.1 Related

- ESP-IDF Get Started
- ESP-IDF Programming Guide
- ESP Product Selector

1.1.1 ESP32-C3-LCD-EV-BOARD

[□□]

ESP32-C3-LCD-EV-BOARD is a development board used to evaluate and verify the application of ESP32-C3 screen. It is composed of mainboard and subboard.

ESP32-C3-LCD-EV-BOARD

[□□]

Note: This document is not the final version and is still being updated.

1.1.2 ESP32-C6-DevKitC-1

[□□]

ESP32-C6-DevKitC-1 is an entry-level development board based on ESP32-C6-WROOM-1, a general-purpose module with a 8 MB SPI flash. This board integrates complete Wi-Fi, Bluetooth LE, Zigbee, and Thread functions.

ESP32-C6-DevKitC-1

[□□]

This user guide will help you get started with ESP32-C6-DevKitC-1 and will also provide more in-depth information.

ESP32-C6-DevKitC-1 is an entry-level development board based on ESP32-C6-WROOM-1, a general-purpose module with a 8 MB SPI flash. This board integrates complete Wi-Fi, Bluetooth LE, Zigbee, and Thread functions.

Most of the I/O pins are broken out to the pin headers on both sides for easy interfacing. Developers can either connect peripherals with jumper wires or mount ESP32-C6-DevKitC-1 on a breadboard.

The document consists of the following major sections:

- *Getting Started*: Overview of ESP32-C6-DevKitC-1 and hardware/software setup instructions to get started.
- *Hardware Reference*: More detailed information about the ESP32-C6-DevKitC-1's hardware.
- *Hardware Revision Details*: Revision history, known issues, and links to user guides for previous versions (if any) of ESP32-C6-DevKitC-1.
- *Related Documents*: Links to related documentation.

Getting Started

This section provides a brief introduction of ESP32-C6-DevKitC-1, instructions on how to do the initial hardware setup and how to flash firmware onto it.

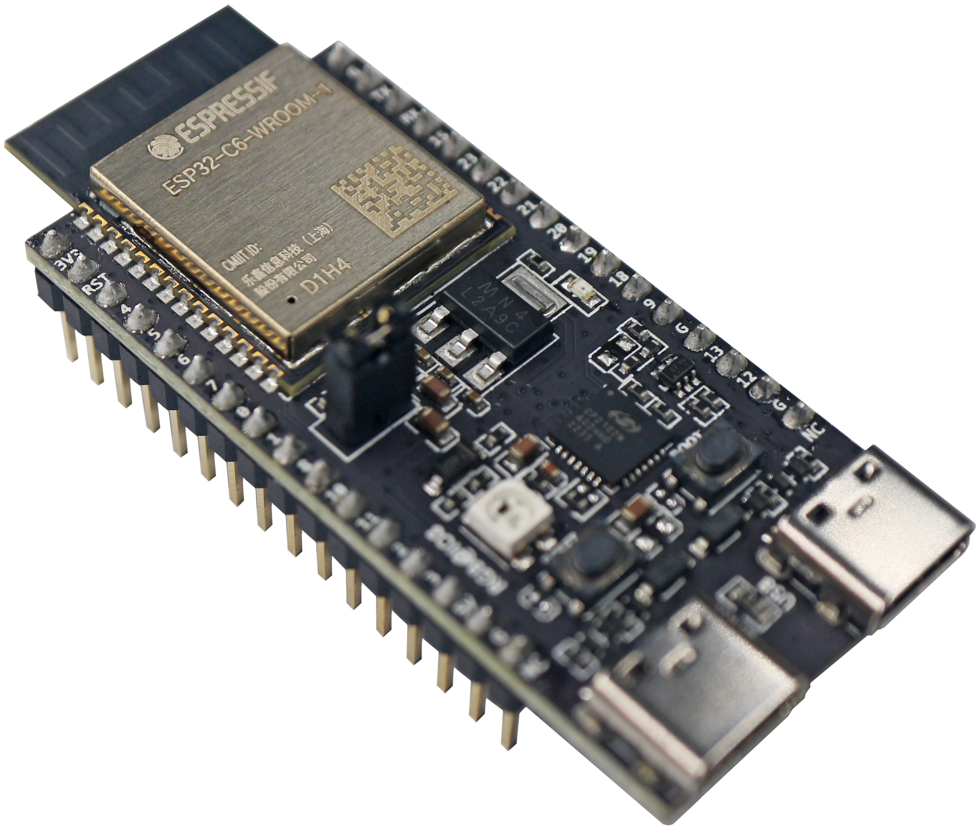


Fig. 1: ESP32-C6-DevKitC-1

Description of Components

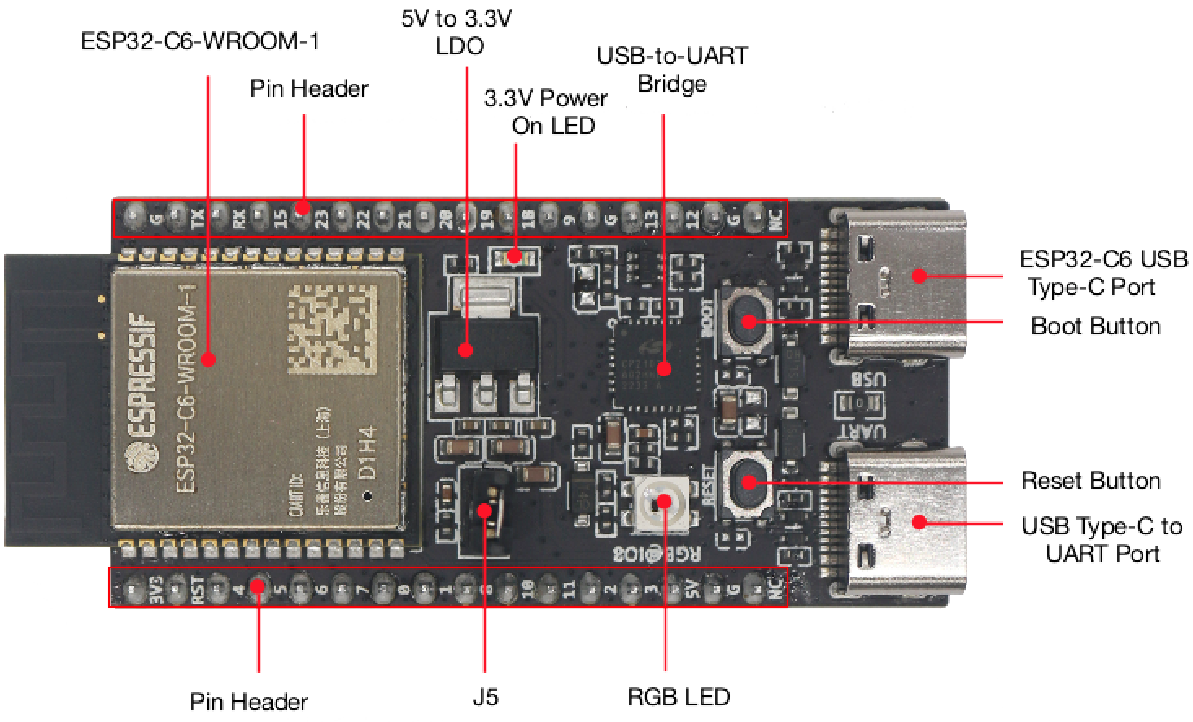


Fig. 2: ESP32-C6-DevKitC-1 - front

The key components of the board are described in a clockwise direction.

Key Component	Description
ESP32-C6-WROOM-1	ESP32-C6-WROOM-1 is a general-purpose module supporting Wi-Fi 6, Bluetooth 5, and IEEE 802.15.4 (Zigbee 3.0 and Thread). This module is built around the ESP32-C6 chip, and comes with a PCB antenna and a 8 MB SPI flash. For more information, see ESP32-C6-WROOM-1 Datasheet.
Pin Header	All available GPIO pins (except for the SPI bus for flash) are broken out to the pin headers on the board.
5 V to 3.3 V LDO	Power regulator that converts a 5 V supply into a 3.3 V output.
3.3 V Power On LED	Turns on when the USB power is connected to the board.
USB-to-UART Bridge	Single USB-to-UART bridge chip provides transfer rates up to 3 Mbps.
ESP32-C6 USB Type-C Port	The USB Type-C port on the ESP32-C6 chip compliant with USB 2.0 full speed. It is capable of up to 12 Mbps transfer speed (Note that this port does not support the faster 480 Mbps high-speed transfer mode).
Boot Button	Download button. Holding down **Boot** and then pressing **Reset** initiates Firmware Download mode for downloading firmware through the serial port.
Reset Button	Press this button to restart the system.
USB Type-C to UART Port	Used for power supply to the board, as well as the communication with the ESP32-C6 chip via the on-board USB-to-UART bridge.
RGB LED	Addressable RGB LED, driven by GPIO8.
J5	Used for current measurement. See details in Section *Current Measurement*.

Start Application Development

Before powering up your ESP32-C6-DevKitC-1, please make sure that it is in good condition with no obvious signs of damage.

Required Hardware

- ESP32-C6-DevKitC-1
- USB-A to USB-C cable
- Computer running Windows, Linux, or macOS

Note: Be sure to use a good quality USB cable. Some cables are for charging only and do not provide the needed data lines nor work for programming the boards.

Software Setup

Please proceed to ESP-IDF Get Started, which will quickly help you set up the development environment then flash an application example onto your board.

ESP-AT Support

The ESP32-C6-DevKitC-1 supports ESP-AT software that provides a set of AT commands with which you can quickly integrate wireless connectivity features into your product without a need for embedded application development of the module on this development board.

The software is available as a pre-built binary that can be downloaded from ESP-AT repository.

For more information about using ESP-AT, including information on how to customize pre-built binaries, please refer to ESP-AT User Guide.

Contents and Packaging

Retail orders

If you order a few samples, each ESP32-C6-DevKitC-1 comes in an individual package in either antistatic bag or any packaging depending on your retailer.

For retail orders, please go to https://www.espressif.com/en/company/contact/buy-a-sample.

Wholesale Orders

If you order in bulk, the boards come in large cardboard boxes.

For wholesale orders, please check Espressif Product Ordering Information (PDF)

Hardware Reference

Block Diagram

The block diagram below shows the components of ESP32-C6-DevKitC-1 and their interconnections.

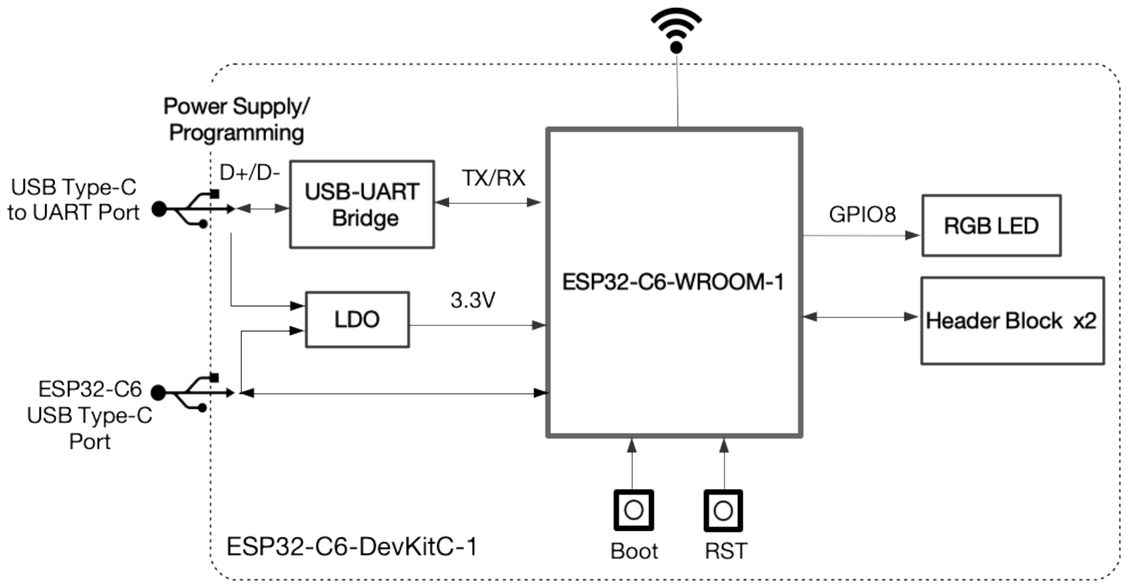


Fig. 3: ESP32-C6-DevKitC-1 (click to enlarge)

Power Supply Options

There are three mutually exclusive ways to provide power to the board:

- USB Type-C to UART Port, default power supply
- 5V and GND pin headers
- 3V3 and GND pin headers

It is recommended to use the first option: USB Type-C to UART Port.

Current Measurement

The J5 headers on ESP32-C6-DevKitC-1 (see J5 in Figure *ESP32-C6-DevKitC-1 - front*) can be used for measuring the current drawn by the ESP32-C6-WROOM-1 module:

- Remove the jumper: Power supply between the module and peripherals on the board is cut off. To measure the module's current, connect the board with an ammeter via J5 headers.
- Apply the jumper (factory default): Restore the board's normal functionality.

Note: When using 3V3 and GND pin headers to power the board, please remove the J5 jumper, and connect an ammeter in series to the external circuit to measure the module's current.

Header Block

The two tables below provide the **Name** and **Function** of the pin headers on both sides of the board (J1 and J3). The pin header names are shown in Figure *ESP32-C6-DevKitC-1 - front*. The numbering is the same as in the ESP32-C6-DevKitC-1 Schematic (PDF).

J1

No.	Name	Type[1]	Function
1	3V3	P	3.3 V Power Supply
2	RST	I	High: enables the chip; Low: disables the chip.
3	4	I/O/T	MTMS[3], GPIO4, LP_GPIO4, LP_UART_RXD, ADC1_CH4, FSPIHD
4	5	I/O/T	MTDI[3], GPIO5, LP_GPIO5, LP_UART_TXD, ADC1_CH5, FSPIWP
5	6	I/O/T	MTCK, GPIO6, LP_GPIO6, LP_I2C_SDA, ADC1_CH6, FSPICLK
6	7	I/O/T	MTDO, GPIO7, LP_GPIO7, LP_I2C_SCL, FSPID
7	0	I/O/T	GPIO0, XTAL_32K_P, LP_GPIO0, LP_UART_DTRN, ADC1_CH0
8	1	I/O/T	GPIO1, XTAL_32K_N, LP_GPIO1, LP_UART_DSRN, ADC1_CH1
9	8	I/O/T	GPIO8[2][3]
10	10	I/O/T	GPIO10
11	11	I/O/T	GPIO11
12	2	I/O/T	GPIO2, LP_GPIO2, LP_UART_RTSN, ADC1_CH2, FSPIQ
13	3	I/O/T	GPIO3, LP_GPIO3, LP_UART_CTSN, ADC1_CH3
14	5V	P	5 V power supply
15	G	G	Ground
16	NC	–	No connection

[1] P: Power supply; I: Input; O: Output; T: High impedance.

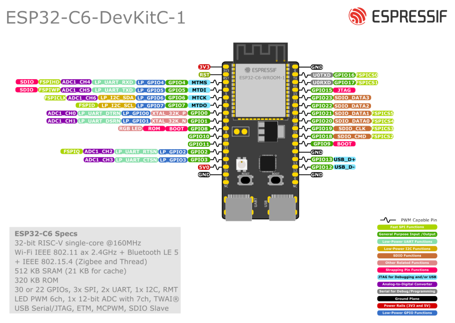
[3] MTMS, MTDI, GPIO8, GPIO9, and GPIO15 are strapping pins of the ESP32-C6 chip. These pins are used to control several chip functions depending on binary voltage values applied to the pins during chip power-up or system reset. For description and application of the strapping pins, please refer to ESP32-C6 Datasheet > Section *Strapping Pins*.

[2] Used to drive the RGB LED.

J3

No.	Name	Type	Function
1	G	G	Ground
2	TX	I/O/T	U0TXD, GPIO16, FSPICS0
3	RX	I/O/T	U0RXD, GPIO17, FSPICS1
4	15	I/O/T	GPIO15[3]
5	23	I/O/T	GPIO23, SDIO_DATA3
6	22	I/O/T	GPIO22, SDIO_DATA2
7	21	I/O/T	GPIO21, SDIO_DATA1, FSPICS5
8	20	I/O/T	GPIO20, SDIO_DATA0, FSPICS4
9	19	I/O/T	GPIO19, SDIO_CLK, FSPICS3
10	18	I/O/T	GPIO18, SDIO_CMD, FSPICS2
11	9	I/O/T	GPIO9[3]
12	G	G	Ground
13	13	I/O/T	GPIO13, USB_D+
14	12	I/O/T	GPIO12, USB_D-
15	G	G	Ground
16	NC	–	No connection

Pin Layout



ESP32-C6 Specs
32-bit RISC-V single-core @160MHz
Wi-Fi IEEE 802.11 ax 2.4GHz + Bluetooth LE 5 + IEEE 802.15.4 (Zigbee and Thread)
512 KB SRAM (21 KB for cache)
320 KB ROM
30 or 22 GPIOs, 3x SPI, 2x UART, 1x I2C, RMT
LED PWM 6ch, 1x 12-bit ADC with 7ch, TWAI®
USB Serial/JTAG, ETM, MCPWM, SDIO Slave

Fig. 4: ESP32-C6-DevKitC-1 Pin Layout (click to enlarge)

Hardware Revision Details

No previous versions available.

Related Documents

- ESP32-C6 Datasheet (PDF)
- ESP32-C6-WROOM-1 Datasheet (PDF)
- ESP32-C6-DevKitC-1 Schematic (PDF)
- ESP32-C6-DevKitC-1 PCB Layout (PDF)
- ESP32-C6-DevKitC-1 Dimensions (PDF)
- ESP32-C6-DevKitC-1 Dimensions source file (DXF)

For further design documentation for the board, please contact us at sales@espressif.com.

1.1.3 ESP8684-DevKitM-1

[??]

The ESP8684-DevKitM-1 is an entry-level development board based on ESP8684-MINI-1, a general-purpose module with 1 MB/2 MB/4 MB SPI flash. This board integrates complete Wi-Fi and Bluetooth LE functions.

ESP8684-DevKitM-1 v1.1

[??]

The older version: *ESP8684-DevKitM-1*.

This user guide will help you get started with ESP8684-DevKitM-1 and will also provide more in-depth information.

The ESP8684-DevKitM-1 is an entry-level development board based on ESP8684-MINI-1, a general-purpose module with 1 MB/2 MB/4 MB SPI flash. This board integrates complete Wi-Fi and Bluetooth LE functions.

Most of the I/O pins on the module are broken out to the pin headers on both sides of this board for easy interfacing. Developers can either connect peripherals with jumper wires or mount ESP8684-DevKitM-1 on a breadboard.

The document consists of the following major sections:

- *Getting Started*: Overview of the board and hardware/software setup instructions to get started.
- *Hardware Overview*: More detailed information about the board's hardware.
- *Hardware Revision Details*: Hardware revision history, known issues, and links to user guides for previous versions (if any) of the board.
- *Related Documents*: Links to related documentation.

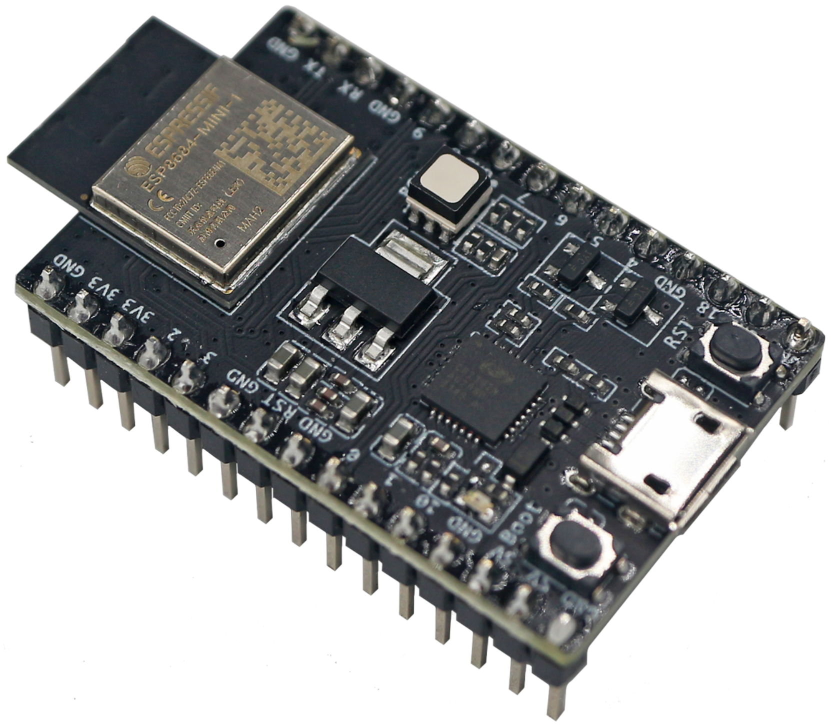


Fig. 5: ESP8684-DevKitM-1 with ESP8684-MINI-1 module

Getting Started

This section provides a brief introduction of ESP8684-DevKitM-1, instructions on how to do the initial hardware setup and how to flash firmware onto it.

Description of Components

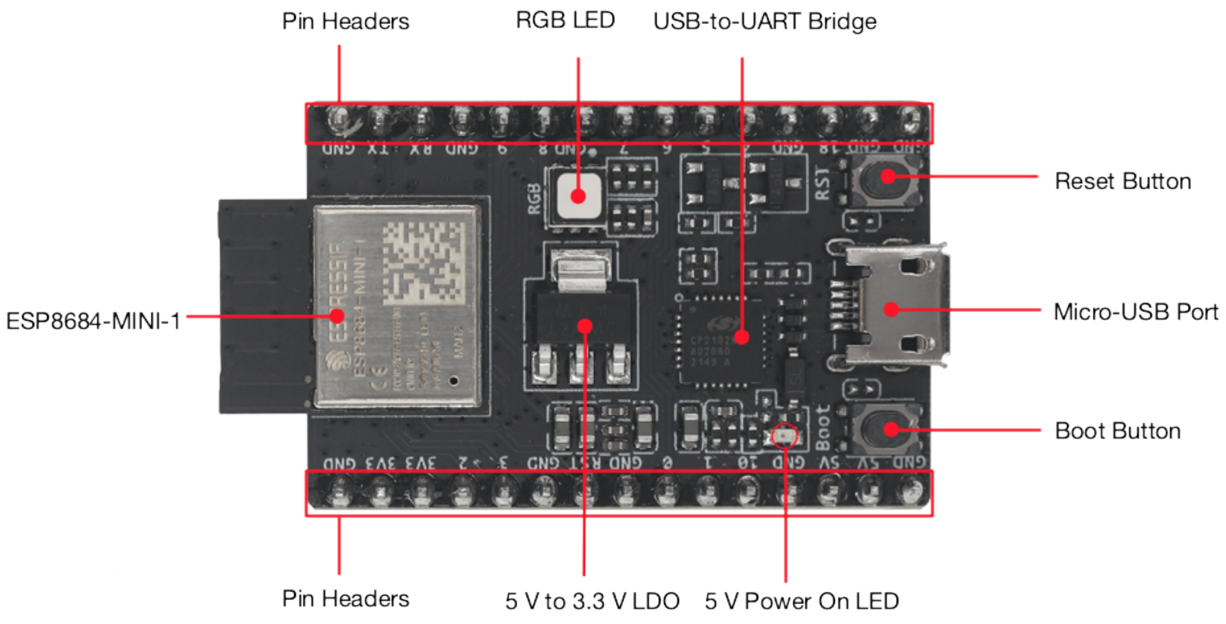


Fig. 6: ESP8684-DevKitM-1 - front

The key components of the board are described in a counter-clockwise direction.

Key Component	Description
ESP8684-MINI-1	ESP8684-MINI-1 from Espressif is a powerful and general-purpose module that offers Wi-Fi and Bluetooth LE coexistence. It has a PCB antenna and a 1 MB/2 MB/4 MB SPI flash.
Pin Headers	All available GPIO pins are broken out to the pin headers on the board. For details, please see *Header Block*.
5 V to 3.3 V LDO	Power regulator that converts a 5 V supply into a 3.3 V output.
5 V Power On LED	Turns on when the USB power is connected to the board.
Boot Button	Download button. Holding down **Boot** and then pressing **Reset** initiates Firmware Download mode for downloading firmware through the serial port.
Micro-USB Port	USB interface. Power supply for the board as well as the communication interface between a computer and the ESP8684 chip.
Reset Button	Press this button to restart the system.
USB-to-UART Bridge	Single USB-to-UART bridge chip provides transfer rates up to 3 Mbps.
RGB LED	RGB LED, driven by GPIO0, GPIO1 and GPIO8.

Start Application Development

Before powering up your board, please make sure that it is in good condition with no obvious signs of damage.

Required Hardware

- ESP8684-DevKitM-1
- USB 2.0 cable (Standard-A to Micro-B)
- Computer running Windows, Linux, or macOS

Note: Be sure to use an appropriate USB cable. Some cables are for charging only and do not provide the needed data lines nor work for programming the boards.

Software Setup

Please proceed to Get Started, where Section Installation Step by Step will quickly help you set up the development environment then flash an application example onto your board.

ESP-AT Support

The ESP8684-DevKitM-1 supports ESP-AT software that provides a set of AT commands with which you can quickly integrate wireless connectivity features into your product without a need for embedded application development of the module on this development board .

The software is available as a pre-built binary that can be downloaded from ESP-AT repository.

For more information about using ESP-AT, including information on how to customize pre-built binaries, please refer to ESP-AT User Guide.

Contents and Packaging

Retail Orders

If you order a few samples, each board comes in an individual package in either antistatic bag or any packaging depending on your retailer.

For retail orders, please go to https://www.espressif.com/en/company/contact/buy-a-sample.

Wholesale Orders

If you order in bulk, the boards come in large cardboard boxes.

For wholesale orders, please go to https://www.espressif.com/en/contact-us/sales-questions.

Hardware Overview

Block Diagram

The block diagram below shows the components of ESP8684-DevKitM-1 and their interconnections.

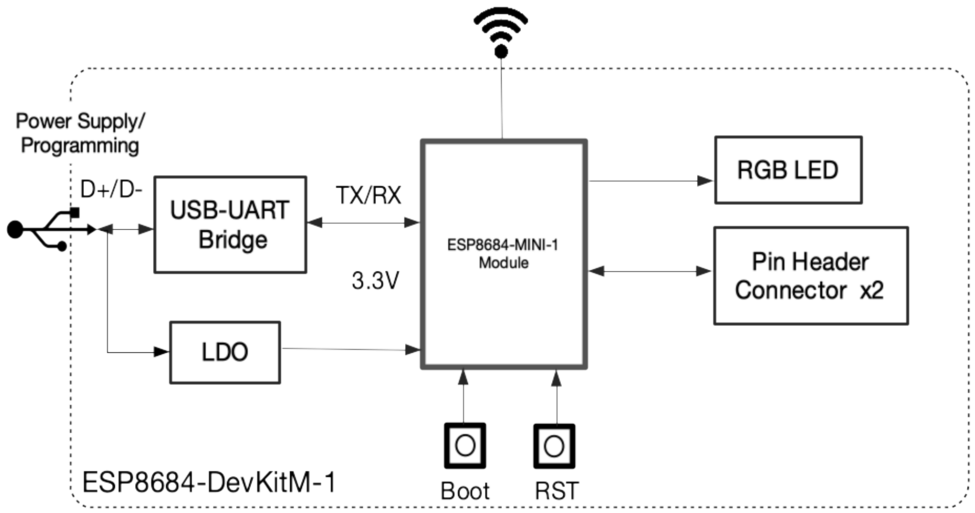


Fig. 7: ESP8684-DevKitM-1 (click to enlarge)

Power Supply Options

There are three mutually exclusive ways to provide power to the board:

- Micro-USB Port, default power supply (recommended)
- 5V and G (GND) pins
- 3V3 and G (GND) pins

Header Block

The two tables below provide the **Name** and **Function** of the pins on both sides of the board (J1 and J3). The pin names are shown in *ESP8684-DevKitM-1 - front*. The numbering is the same as in the Board Schematic (PDF).

J1

No.	Name	Type[1]	Function
1	G	G	Ground
2	3V3	P	3.3 V power supply
3	3V3	P	3.3 V power supply
4	2	I/O/T	GPIO2, ADC1_CH2, FSPIQ
5	3	I/O/T	GPIO3, ADC1_CH3
6	G	G	Ground
7	RST	I	Reset; High: enable; Low: powers off
8	G	G	Ground
9	0	I/O/T	GPIO0, ADC1_CH0, LED Red
10	1	I/O/T	GPIO1, ADC1_CH1, LED Green
11	10	I/O/T	GPIO10, FSPICS0
12	G	G	Ground
13	5V	P	5 V power supply
14	5V	P	5 V power supply
15	G	G	Ground

J3

No.	Name	Type[1]	Function
1	G	G	Ground
2	TX	I/O/T	GPIO20, U0TXD
3	RX	I/O/T	GPIO19, U0RXD
4	G	G	Ground
5	9	I/O/T	GPIO9[2]
6	8	I/O/T	GPIO8[2], LED Blue
7	G	G	Ground
8	7	I/O/T	GPIO7, FSPID, MTDO
9	6	I/O/T	GPIO6, FSPICLK, MTCK
10	5	I/O/T	GPIO5, ADC2_CH0, FSPIWP ,MTDI
11	4	I/O/T	GPIO4, ADC1_CH4, FSPIHD ,MTMS
12	G	G	Ground
13	18	I/O/T	GPIO18
14	G	G	Ground
15	G	G	Ground

[1] P: Power supply; I: Input; O: Output; T: High impedance.

[2] GPIO8 and GPIO9 are strapping pins of the ESP8684 chip. These pins are used to control several chip functions depending on binary voltage values applied to the pins during chip power-up or system reset. For description and application of the strapping pins, please refer to ESP8684 Datasheet > Section *Strapping Pins*.

Pin Layout

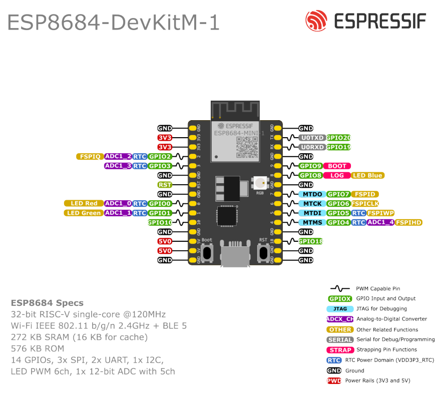

ESP8684-DevKitM-1
ESPRESSIF
ESP8684-MINI-1
GND
3V3
3V3
FSPIQ ADC1_2 RTC GPIO2
ADC1_3 RTC GPIO3
GND
RST
GND
LED Red ADC1_0 RTC GPIO0
LED Green ADC1_1 RTC GPIO1
GPIO10
GND
5V0
5V0
GND
GND
U0TXD GPIO20
U0RXD GPIO19
GND
GPIO9 BOOT
GPIO8 LOG LED Blue
GND
MTDO GPIO7 FSPID
MTCK GPIO6 FSPICLK
MTDI GPIO5 RTC FSPIWP
MTMS GPIO4 RTC ADC1_4 FSPIHD
GND
GPIO18
GND
GND
RGB
Boot
RST
ESP8684 Specs
32-bit RISC-V single-core @120MHz
Wi-Fi IEEE 802.11 b/g/n 2.4GHz + BLE 5
272 KB SRAM (16 KB for cache)
576 KB ROM
14 GPIOs, 3x SPI, 2x UART, 1x I2C,
LED PWM 6ch, 1x 12-bit ADC with 5ch
PWM Capable Pin
GPIOX GPIO Input and Output
JTAG JTAG for Debugging
ADCX_CH Analog-to-Digital Converter
OTHER Other Related Functions
SERIAL Serial for Debug/Programming
STRAP Strapping Pin Functions
RTC RTC Power Domain (VDD3P3_RTC)
GND Ground
PWD Power Rails (3V3 and 5V)

Fig. 8: ESP8684-DevKitM-1 Pin Layout (click to enlarge)

Hardware Revision Details

Initial release

Main differences compared to the previous version:

- Addressable RGB LED in the previous version of the board has been changed to an RGB LED with discrete inputs for each color.
- The addressable LED was connected to GPIO8, and the new discrete LED is connected to GPIO0, GPIO1, and GPIO8.

Note: Both versions of ESP8684-DevKitM-1 are available on the market.

Related Documents

- ESP8684 Datasheet (PDF)
- ESP8684-DevKitM-1 Schematic (PDF)
- ESP8684-DevKitM-1 PCB layout (PDF)
- ESP8684-DevKitM-1 Dimensions (PDF)
- ESP8684-DevKitM-1 Dimensions source file (DXF) - You can view it with Autodesk Viewer online

For further design documentation for the board, please contact us at sales@espressif.com.

ESP8684-DevKitM-1

[??]

The latest version: *ESP8684-DevKitM-1 v1.1*.

This user guide will help you get started with ESP8684-DevKitM-1 and will also provide more in-depth information.

The ESP8684-DevKitM-1 is an entry-level development board based on ESP8684-MINI-1, a general-purpose module with 1 MB/2 MB/4 MB SPI flash. This board integrates complete Wi-Fi and Bluetooth LE functions.

Most of the I/O pins on the module are broken out to the pin headers on both sides of this board for easy interfacing. Developers can either connect peripherals with jumper wires or mount ESP8684-DevKitM-1 on a breadboard.

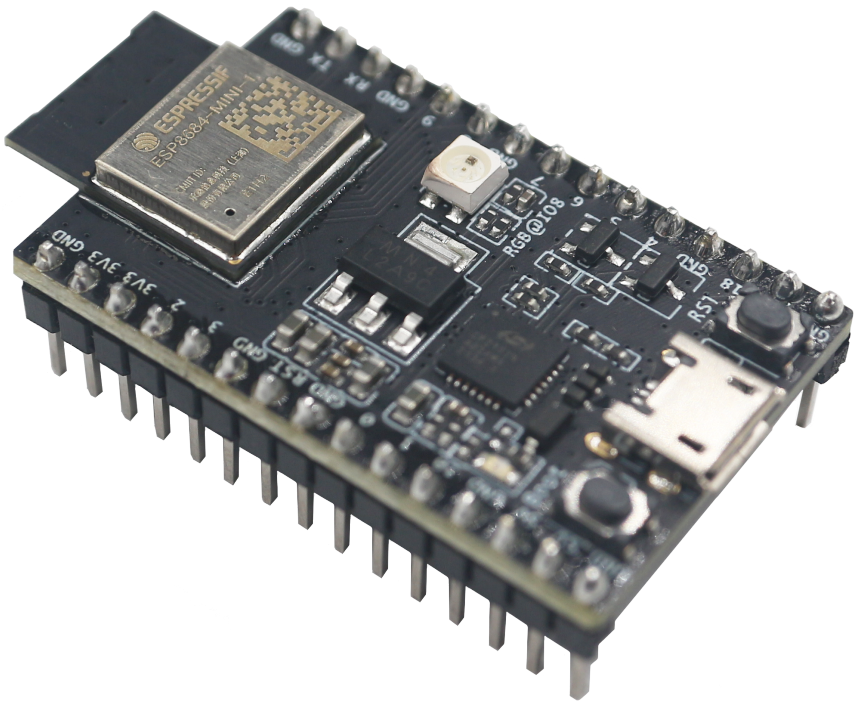


Fig. 9: ESP8684-DevKitM-1 with ESP8684-MINI-1 module

The document consists of the following major sections:

- *Getting Started*: Overview of the board and hardware/software setup instructions to get started.
- *Hardware Overview*: More detailed information about the board's hardware.
- *Hardware Revision Details*: Hardware revision history, known issues, and links to user guides for previous versions (if any) of the board.
- *Related Documents*: Links to related documentation.

Getting Started

This section provides a brief introduction of ESP8684-DevKitM-1, instructions on how to do the initial hardware setup and how to flash firmware onto it.

Description of Components

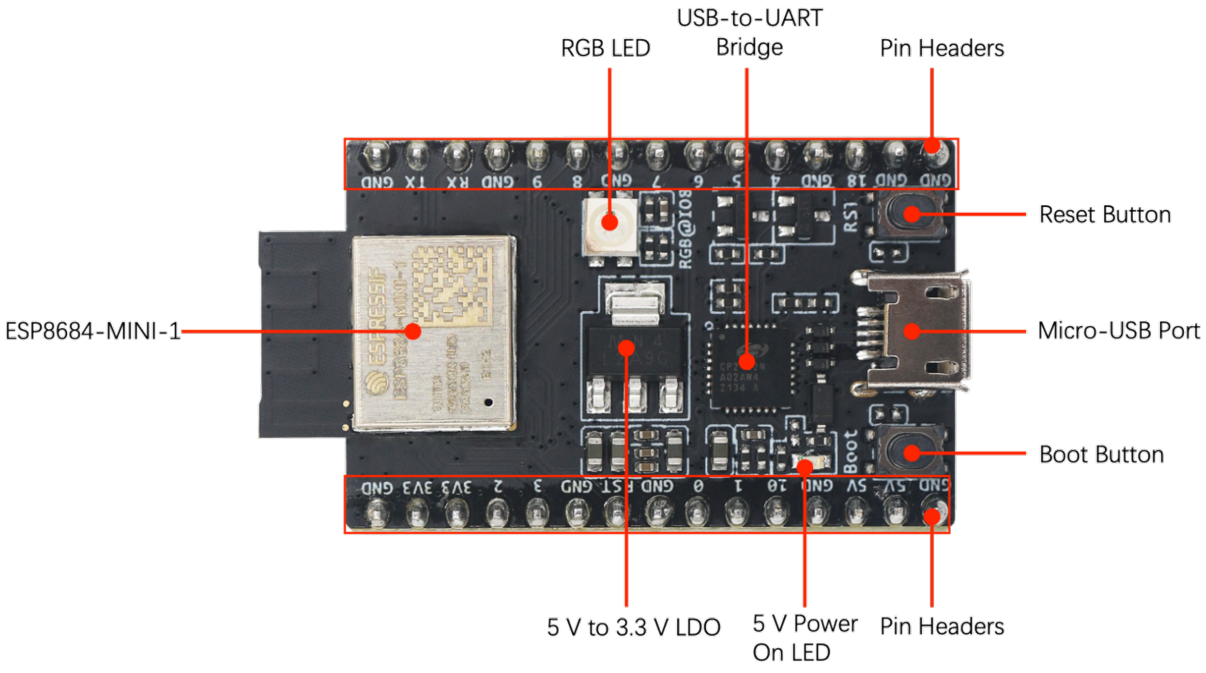


Fig. 10: ESP8684-DevKitM-1 - front

The key components of the board are described in a counter-clockwise direction.

Key Component	Description
ESP8684-MINI-1	ESP8684-MINI-1 from Espressif is a powerful and general-purpose module that offers Wi-Fi and Bluetooth LE coexistence. It has a PCB antenna and a 1 MB/2 MB/4 MB SPI flash.
5 V to 3.3 V LDO	Power regulator that converts a 5 V supply into a 3.3 V output.
5 V Power On LED	Turns on when the USB power is connected to the board.
Pin Headers	All available GPIO pins are broken out to the pin headers on the board. For details, please see *Header Block*.
Boot Button	Download button. Holding down **Boot** and then pressing **Reset** initiates Firmware Download mode for downloading firmware through the serial port.
Micro-USB Port	USB interface. Power supply for the board as well as the communication interface between a computer and the ESP8684 chip.
Reset Button	Press this button to restart the system.
USB-to-UART Bridge	Single USB-to-UART bridge chip provides transfer rates up to 3 Mbps.
RGB LED	Addressable RGB LED, driven by GPIO8.

Start Application Development

Before powering up your board, please make sure that it is in good condition with no obvious signs of damage.

Required Hardware

- ESP8684-DevKitM-1
- USB 2.0 cable (Standard-A to Micro-B)
- Computer running Windows, Linux, or macOS

Note: Be sure to use an appropriate USB cable. Some cables are for charging only and do not provide the needed data lines nor work for programming the boards.

Software Setup

Please proceed to Get Started, where Section Installation Step by Step will quickly help you set up the development environment then flash an application example onto your board.

ESP-AT Support

The ESP8684-DevKitM-1 supports ESP-AT software that provides a set of AT commands with which you can quickly integrate wireless connectivity features into your product without a need for embedded application development of the module on this development board.

The software is available as a pre-built binary that can be downloaded from ESP-AT repository.

For more information about using ESP-AT, including information on how to customize pre-built binaries, please refer to ESP-AT User Guide.

Contents and Packaging

Retail Orders

If you order a few samples, each board comes in an individual package in either antistatic bag or any packaging depending on your retailer.

For retail orders, please go to https://www.espressif.com/en/company/contact/buy-a-sample.

Wholesale Orders

If you order in bulk, the boards come in large cardboard boxes.

For wholesale orders, please go to https://www.espressif.com/en/contact-us/sales-questions.

Hardware Overview

Block Diagram

The block diagram below shows the components of ESP8684-DevKitM-1 and their interconnections.

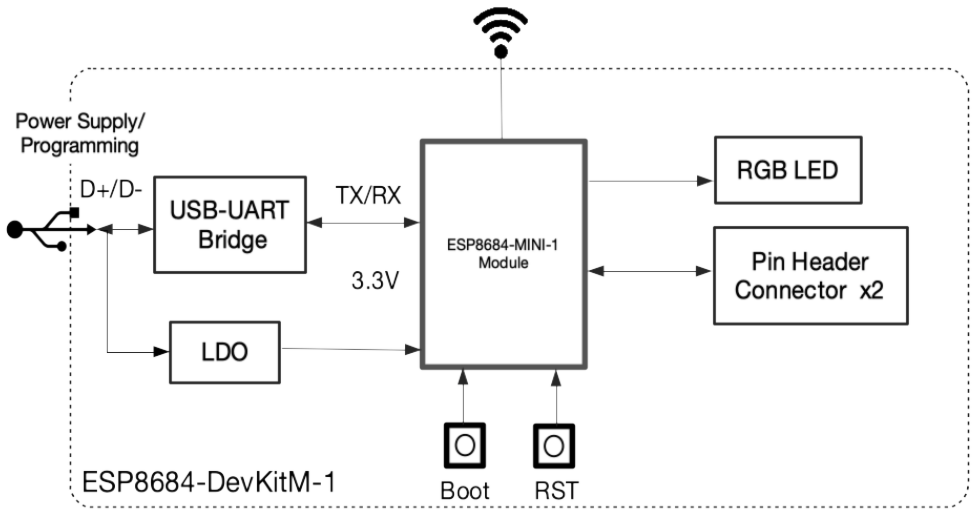


Fig. 11: ESP8684-DevKitM-1 (click to enlarge)

Power Supply Options

There are three mutually exclusive ways to provide power to the board:

- Micro-USB Port, default power supply (recommended)
- 5V and G (GND) pins
- 3V3 and G (GND) pins

Header Block

The two tables below provide the **Name** and **Function** of the pins on both sides of the board (J1 and J3). The pin names are shown in *ESP8684-DevKitM-1 - front*. The numbering is the same as in the Board Schematic (PDF).

J1

No.	Name	Type[1]	Function
1	G	G	Ground
2	3V3	P	3.3 V power supply
3	3V3	P	3.3 V power supply
4	2	I/O/T	GPIO2, ADC1_CH2, FSPIQ
5	3	I/O/T	GPIO3, ADC1_CH3
6	G	G	Ground
7	RST	I	Reset; High: enable; Low: powers off
8	G	G	Ground
9	0	I/O/T	GPIO0, ADC1_CH0
10	1	I/O/T	GPIO1, ADC1_CH1
11	10	I/O/T	GPIO10, FSPICS0
12	G	G	Ground
13	5V	P	5 V power supply
14	5V	P	5 V power supply
15	G	G	Ground

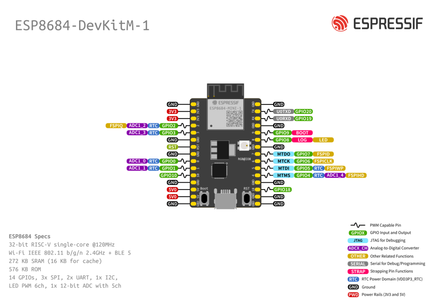
J3

No.	Name	Type[1]	Function
1	G	G	Ground
2	TX	I/O/T	GPIO20, U0TXD
3	RX	I/O/T	GPIO19, U0RXD
4	G	G	Ground
5	9	I/O/T	GPIO9[2]
6	8	I/O/T	GPIO8[2], RGB LED
7	G	G	Ground
8	7	I/O/T	GPIO7, FSPID , MTDO
9	6	I/O/T	GPIO6, FSPICLK , MTCK
10	5	I/O/T	GPIO5, ADC2_CH0, FSPIWP ,MTDI
11	4	I/O/T	GPIO4, ADC1_CH4, FSPIHD ,MTMS
12	G	G	Ground
13	18	I/O/T	GPIO18
14	G	G	Ground
15	G	G	Ground

[1] P: Power supply; I: Input; O: Output; T: High impedance.

[2] GPIO8 and GPIO9 are strapping pins of the ESP8684 chip. These pins are used to control several chip functions depending on binary voltage values applied to the pins during chip power-up or system reset.

Pin Layout



ESP8684 Specs
32-bit RISC-V single-core @120MHz
Wi-Fi IEEE 802.11 b/g/n 2.4GHz + BLE 5
272 KB SRAM (16 KB for cache)
576 KB ROM
14 GPIOs, 3x SPI, 2x UART, 1x I2C,
LED PWM 6ch, 1x 12-bit ADC with 5ch

Fig. 12: ESP8684-DevKitM-1 Pin Layout (click to enlarge)

Hardware Revision Details

This is the first revision of this board released.

Related Documents

- ESP8684-DevKitM-1 Schematic (PDF)
- ESP8684-DevKitM-1 PCB layout (PDF)
- ESP8684-DevKitM-1 Dimensions (PDF)
- ESP8684-DevKitM-1 Dimensions source file (DXF) - You can view it with Autodesk Viewer online

For further design documentation for the board, please contact us at sales@espressif.com.

1.1.4 ESP32-S3-USB-OTG

[□□]

ESP32-S3-USB-OTG is a development board that focuses on USB-OTG function verification and application development.

Application examples for this board can be found at Examples .

ESP32-S3-USB-OTG

[□□]

ESP32-S3-USB-OTG is a development board that focuses on USB-OTG function verification and application development. It is based on ESP32-S3 SoC, supports Wi-Fi and BLE 5.0 wireless functions, and supports USB host and USB device functions. It can be used to develop applications such as wireless storage devices, Wi-Fi network cards, LTE MiFi, multimedia devices, virtual keyboards and mice. The development board has the following features:

- Onboard ESP32-S3-MINI-1-N8 module, with built-in 8 MB flash
- Onboard USB Type-A host and device interface, with built-in USB interface switching circuit
- Onboard USB to serial debugging chip (Micro USB interface)

- Onboard 1.3-inch LCD color screen, supports GUI
- Onboard SD card interface, compatible with SDIO and SPI interfaces
- Onboard charging IC, can be connected to lithium battery

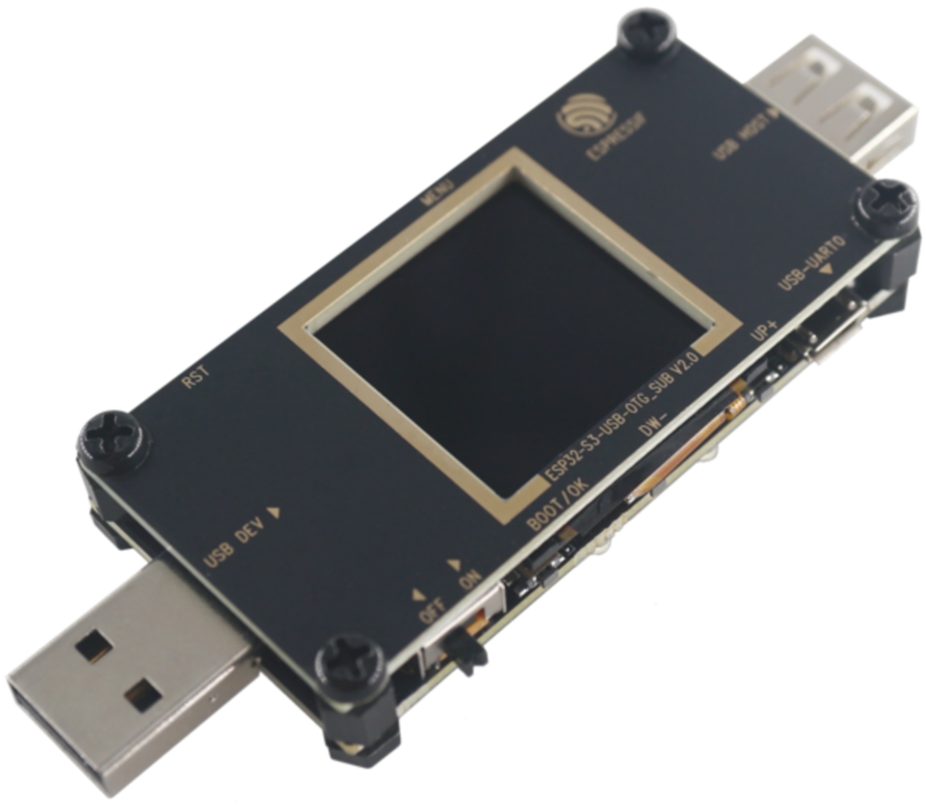


Fig. 13: ESP32-S3-USB-OTG (click to enlarge)

The document consists of the following major sections:

- *Getting Started*: Provides a brief overview of ESP32-S3-USB-OTG and necessary hardware and software information.
- *Hardware Reference*: Provides detailed hardware information of ESP32-S3-USB-OTG.
- *Related Documents*: Provides links to related documents.

Getting Started

This section describes how to start using ESP32-S3-USB-OTG. It includes introduction to basic information about ESP32-S3-USB-OTG first, and then on how to start using the development board for application development, as well as board packaging and retail information.

Description of Components

The ESP32-S3-USB-OTG development board includes the following parts:

- **Motherboard:** ESP32-S3-USB-OTG motherboard is the core of the kit. The motherboard integrates the ESP32-S3-MINI-1 module and provides an interface of the 1.3-inch LCD screen.

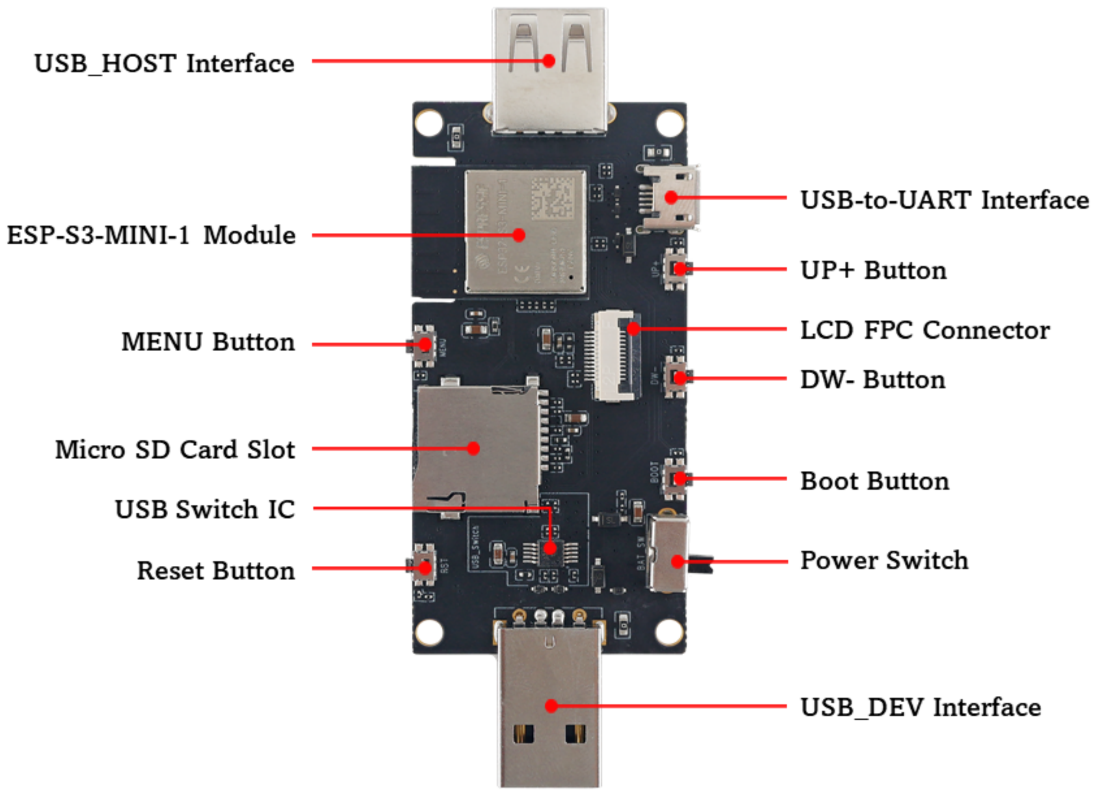


Fig. 14: ESP32-S3-USB-OTG Top View (click to enlarge)

The following table starts with the USB_HOST Interface on the left, and introduces the main components in the above figure in an anticlockwise order.

Main components	Description
USB_HOST Interface	USB Type-A female port, used to connect other USB devices.
ESP32-S3-MINI-1 Module	ESP32-S3-MINI-1 is a powerful, generic Wi-Fi + Bluetooth LE MCU module that has a rich set of peripherals. It has strong ability for neural network computing and signal processing. ESP32-S3-MINI-1 comes with a PCB antenna and is pin-to-pin compatible with ESP32-S2-MINI-1.
MENU Button	Menu button.
Micro SD Card Slot	Micro SD card can be inserted. Both four-line SDIO and SPI mode are supported.
USB Switch IC	By setting the level of USB_SEL, you can switch USB peripherals to make them either connected to the USB_DEV interface or the USB_HOST interface. USB_DEV will be connected by default.
Reset Button	Press this button to restart the system.
USB_DEV Interface	USB Type-A male port, can be connected to the USB host, and also used as a lithium battery charge power source.
Power Switch	Switch to ON to use battery power. Switch to OFF to power off battery.
Boot Button	Download button. Holding down Boot and then pressing Reset initiates Firmware Download mode for downloading firmware through the serial port.
DW- Button	Down button.
LCD FPC Connector	Used to connect the 1.3-inch LCD screen.
UP+ Button	Up button.
USB-to-UART Interface	A Micro-USB port used for power supply to the board, for flashing applications to the chip, as well as for communication with the chip via the on-board USB-to-UART bridge.

The following table starts with the Yellow LED on the left, and introduces the main components in the above figure in an anticlockwise order.

Main components	Description
Yellow LED	Driven by GPIO16, set high level to turn on.
Green LED	Driven by GPIO15, set high level to turn on.
Charging LED	During charging, the red light is on, which will be turned off when charged.
Battery Solder Joints	3.6 V lithium battery can be welded to power the motherboard.
Charging Circuit	Used to charge lithium battery.
Free Pins	Idle pins that can be customized.
USB-to-UART Bridge	Single USB-to-UART bridge chip provides transfer rates up to 3 Mbps.

- **Subboard:** ESP32-S3-USB-OTG-SUB mount the 1.3-inch LCD screen

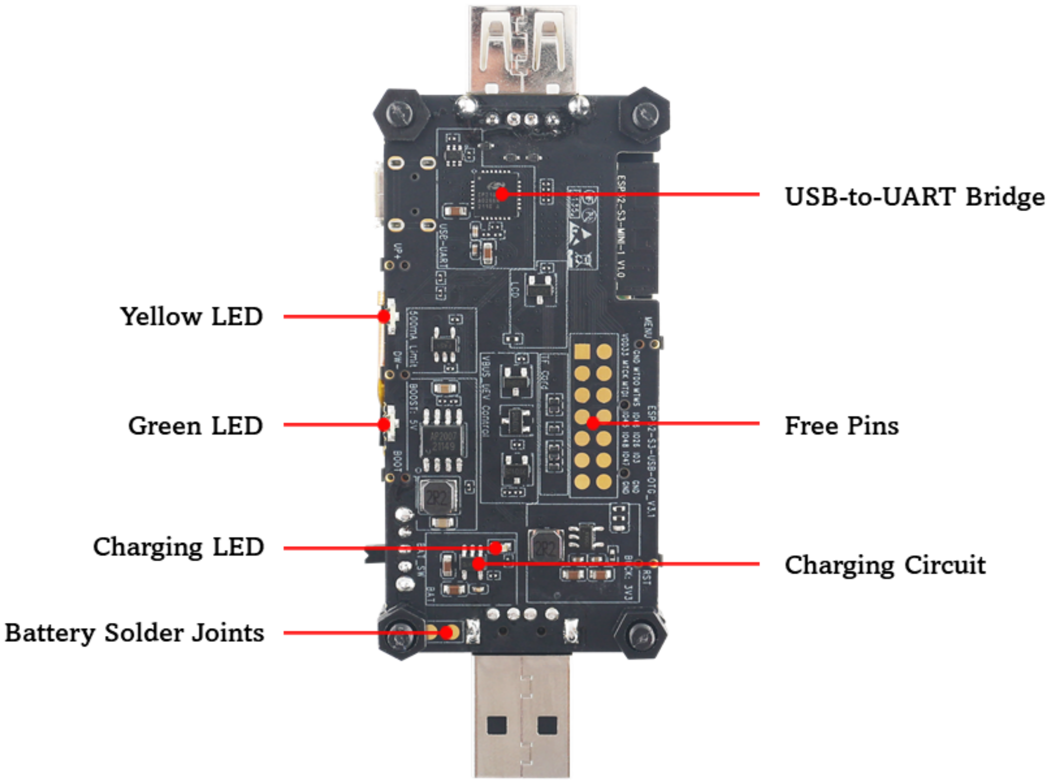


Fig. 15: ESP32-S3-USB-OTG Bottom View (click to enlarge)

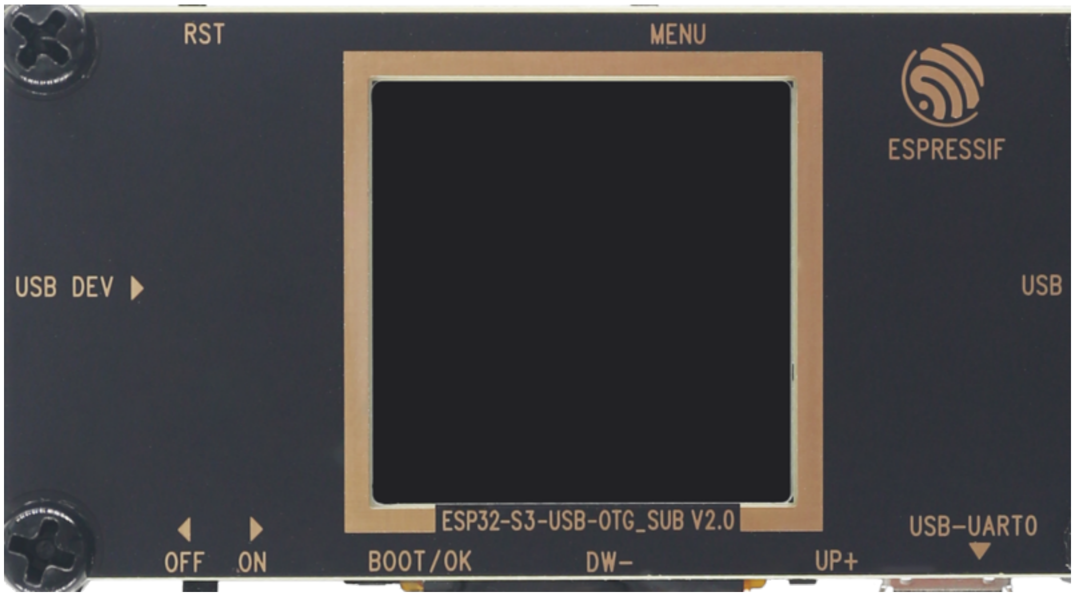


Fig. 16: ESP32-S3-USB-OTG Subboard (click to enlarge)

Start Application Development

Before powering on the ESP32-S3-USB-OTG, please make sure that the development board is intact.

Required Hardware

- ESP32-S3-USB-OTG
- A USB 2.0 data cable (standard A to Micro-B)
- Computer (Windows, Linux or macOS)

Software Setup

Please proceed to Get Started, where Section Installation Step by Step will quickly help you set up the development environment and then flash an application example onto your board.

Project Option

An example is provided for ESP32-S3-USB-OTG under the folder Examples .

You can configure project options by entering `idf.py menuconfig` in the example directory.

Contents and Packaging

Retail Orders

If you order a few samples, each board comes in an individual package in either an antistatic bag or any packaging depending on your retailer.

Which contains the following parts:

- Motherboard:
  - ESP32-S3-USB-OTG
- Subboard:
  - ESP32-S3-USB-OTG_SUB
- Fastener
  - Mounting bolt (x4)
  - Screw (x4)
  - Nut (x4)

For retail orders, please go to https://www.espressif.com/zh-hans/company/contact/buy-a-sample.

Fig. 17: ESP32-S3-USB-OTG Package (click to enlarge)

Wholesale Order

If purchased in bulk, the development board will be packaged in a large cardboard box.

For wholesale orders, please go to https://www.espressif.com/en/contact-us/sales-questions.

Hardware Reference

Block Diagram

The block diagram below shows the components of ESP32-S3-USB-OTG and their interconnections.

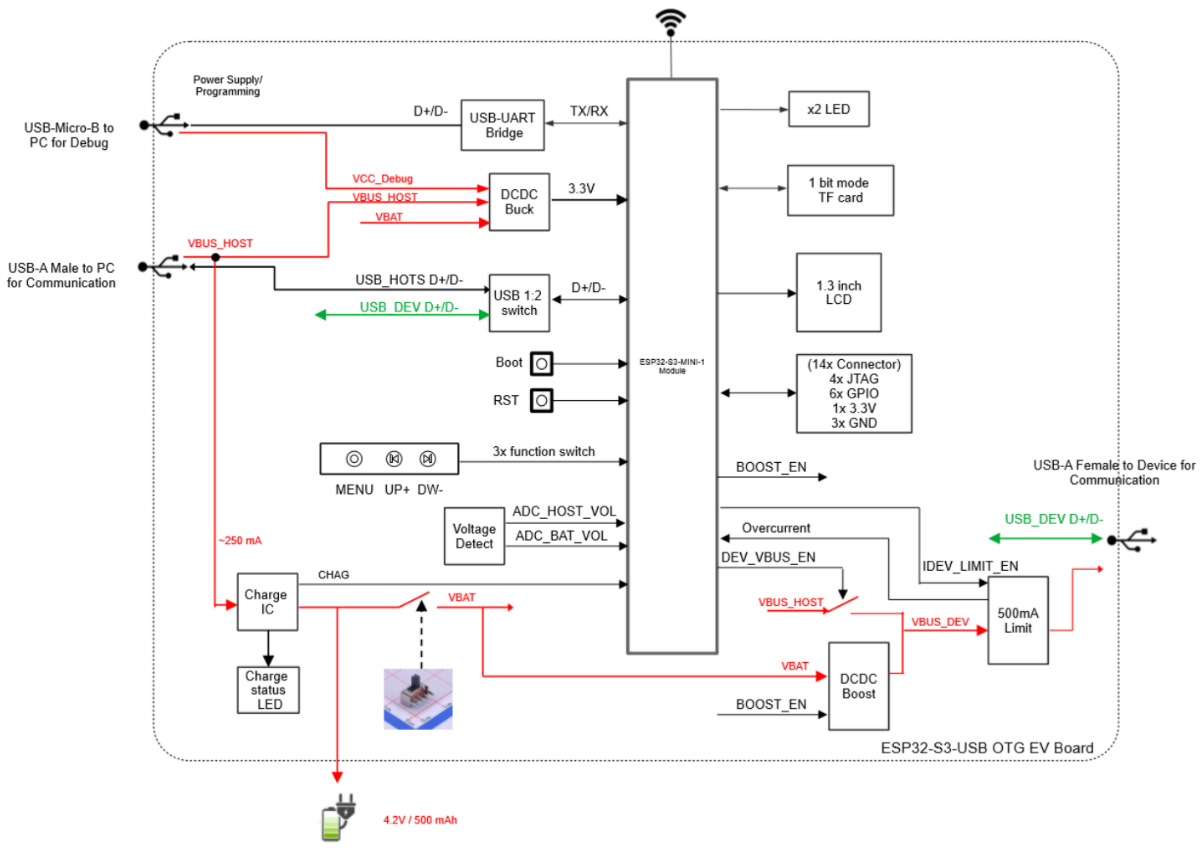


Fig. 18: ESP32-S3-USB-OTG Block Diagram (click to enlarge)

Please note that the external interface corresponding to the `USB_HOST D+ D-` signal in the functional block diagram is `USB DEV`, which means that ESP32-S3 is used as a device to receive signals from other USB hosts. The external interface corresponding to the `USB_DEV D+ D-` signal is `USB HOST`, which means that ESP32-S3 acts as a host to control other devices.

Power Supply Options

There are three power supply methods for the development board:

1. Power supply through the `Micro_USB` interface
   - Use the USB cable (standard A to Micro-B) to connect the motherboard to a power supply device, and set battery switch to OFF. Please note that in this power supply mode, only the motherboard and display are powered.
2. Power supply through the `USB_DEV` interface
   - Set `DEV_VBUS_EN` to high level, and set the battery switch to OFF. This mode can supply power to the `USB HOST` interface. The lithium battery will be charged at the same time (if the lithium battery is installed)
3. Power supply through the battery
   - Set `BOOST_EN` to high level, and set the battery switch to ON. You should solder a 1-Serial lithium battery (3.7 V ~ 4.2 V) to the power solder joint reserved on the back of the motherboard first. This mode can supply power to the `USB HOST` interface at the same time. The battery interface description is as follows:

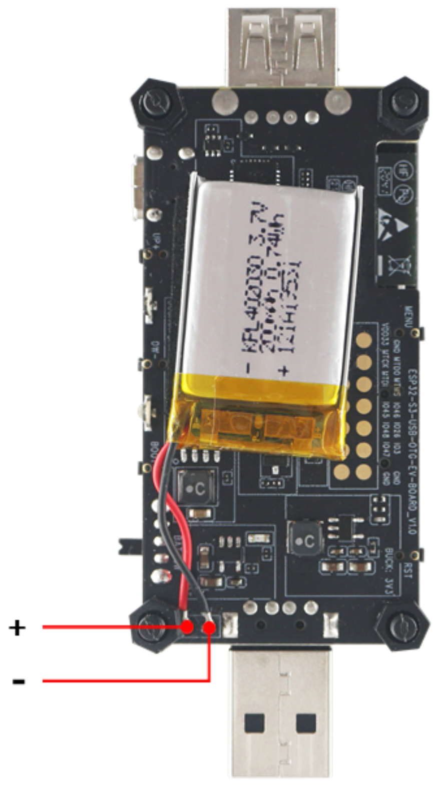


Fig. 19: Battery Connection (click to enlarge)

USB HOST Interface Power Options

The USB HOST interface (Type-A female port) can supply power to the connected USB device. The power supply voltage is 5 V and the maximum current is 500 mA.

- There are two power supply methods for the USB HOST interface:
  1. Power is supplied through the USB_DEV interface, and the 5 V power is directly from the power source connected to the interface.
  2. Power is supplied through the lithium battery, and the 3.6 V ~ 4.2 V voltage of the lithium battery is boosted to 5 V through the Boost circuit. The working status of Boost IC can be controlled by BOOST_EN/GPIO13, set high to enable Boost.

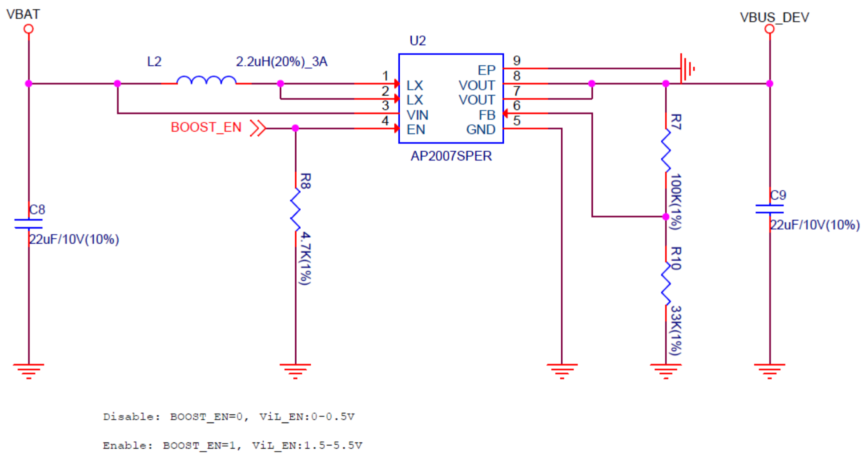


Fig. 20: Boost Circuit (click to enlarge)

- USB HOST interface power supply selection:

BOOST_EN	DEV_VBUS_EN	Power Source
0	1	USB_DEV
1	0	Battery
0	0	No output
1	1	Undefined

- 500 mA current limiting circuit:
  1. The current limiting IC MIC2005A can limit the maximum output current of the USB HOST interface to 500 mA. Please set the IDEV_LIMIT_EN (GPIO17) to high level to enable the current-limiting IC to output voltage.

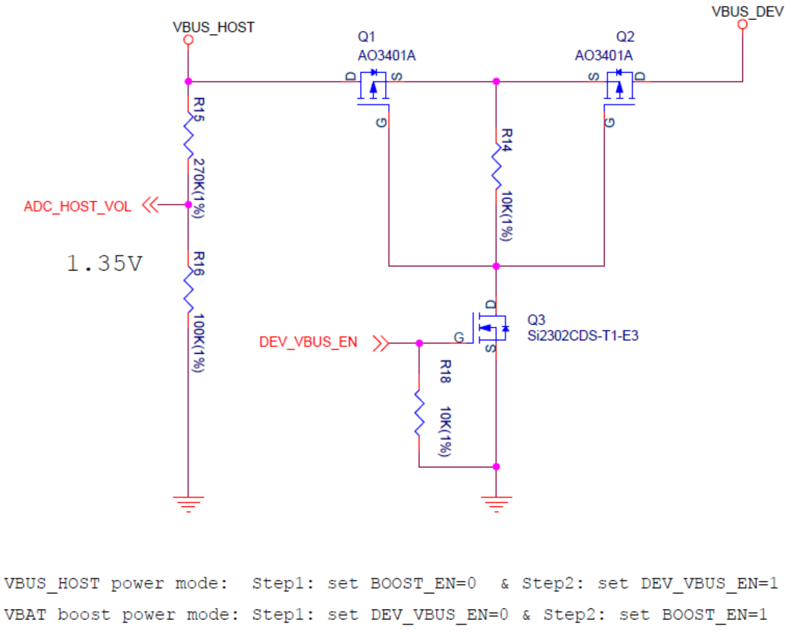

VBUS_HOST
VBUS_DEV
Q1
AO3401A
Q2
AO3401A
R15
270K(1%)
ADC_HOST_VOL
1.35V
R16
100K(1%)
R14
10K(1%)
Q3
Si2302CDS-T1-E3
DEV_VBUS_EN
R18
10K(1%)
Switch to VBUS_HOST power mode: Step1: set BOOST_EN=0 & Step2: set DEV_VBUS_EN=1
Switch to VBAT boost power mode: Step1: set DEV_VBUS_EN=0 & Step2: set BOOST_EN=1

Fig. 21: Power Switch Circuit (click to enlarge)

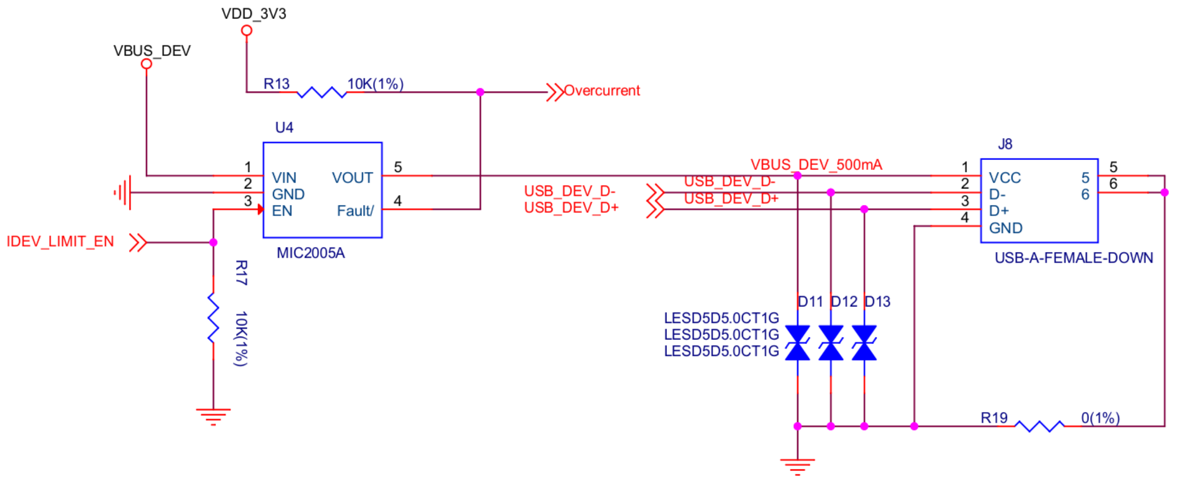


Fig. 22: 500 mA Current Limiting Circuit (click to enlarge)

USB Interface Switch Circuit

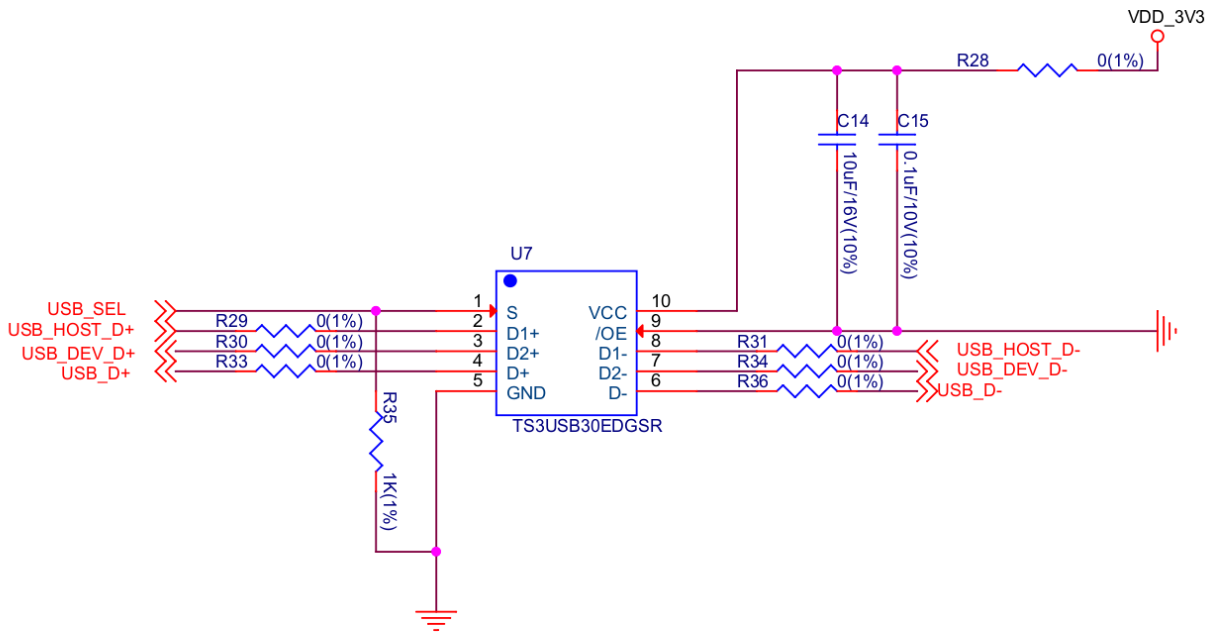


Fig. 23: USB Interface Switch Circuit (click to enlarge)

- When **USB_SEL** (GPIO18) is set to high level, the USB D+/D- Pin (GPIO19, 20) will be connected to `USB_DEV D+ D-`. Then you can use the `USB HOST` interface (Type-A female Port) to connect other USB devices.
- When **USB_SEL** (GPIO18) is set to low level, the USB D+/D- Pin (GPIO19, 20) will be connected to `USB_HOST D+ D-`. Then you can use the `USB DEV` interface (Type-A male port) to connect to a host like a PC.
- **USB_SEL** is pulled low level by default.

LCD Interface

Please note that this interface supports connecting SPI interface screens. The screen controller used by this development board is :dev-kits:` ST7789 <esp32-s3-usb-otg/datasheet/ST7789VW_datasheet.pdf>`, and `LCD_BL` (GPIO9) can be used to control the screen backlight.

SD Card Interface

Please note that the SD card interface is compatible with 1-wire, 4-wire SDIO mode and SPI mode. After being powered on, the card will be in 3.3 V signaling mode. Please send the first CMD0 command to select the bus mode: SD mode or SPI mode.

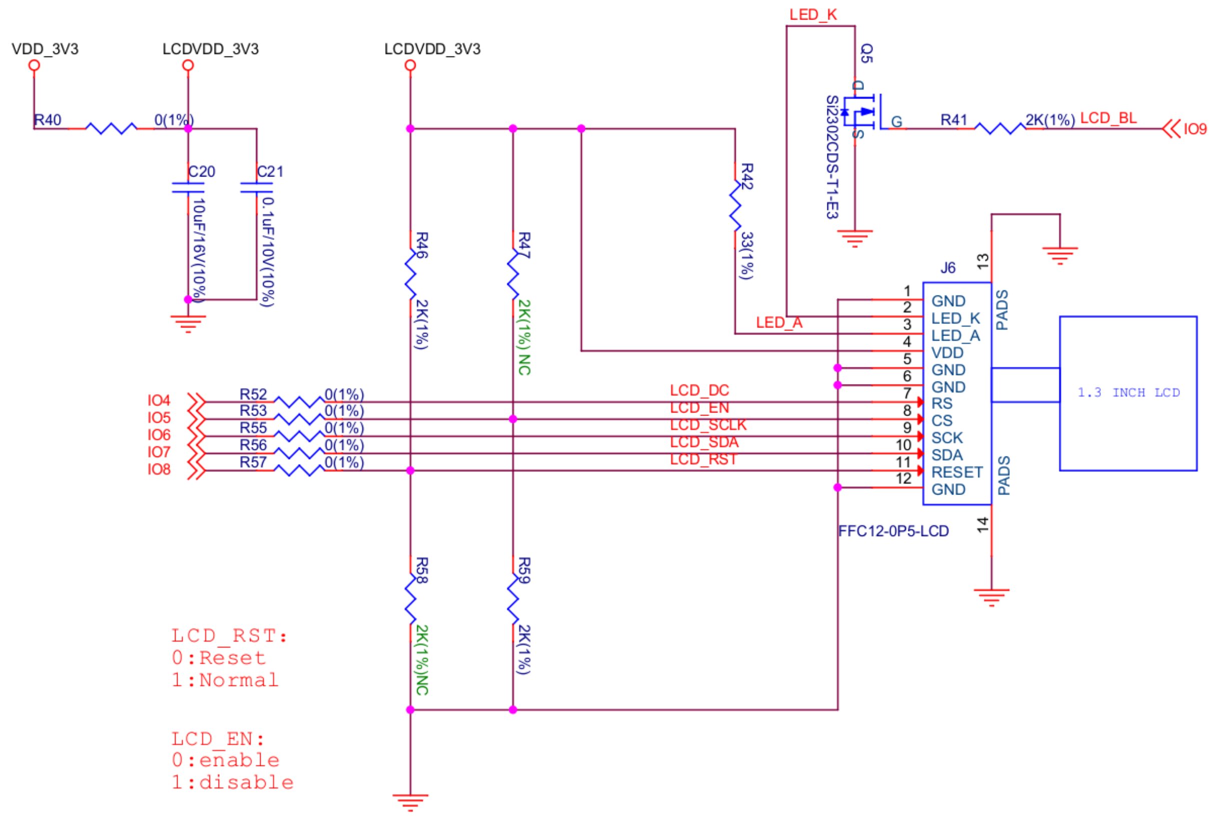


Fig. 24: LCD Interface Circuit (click to enlarge)

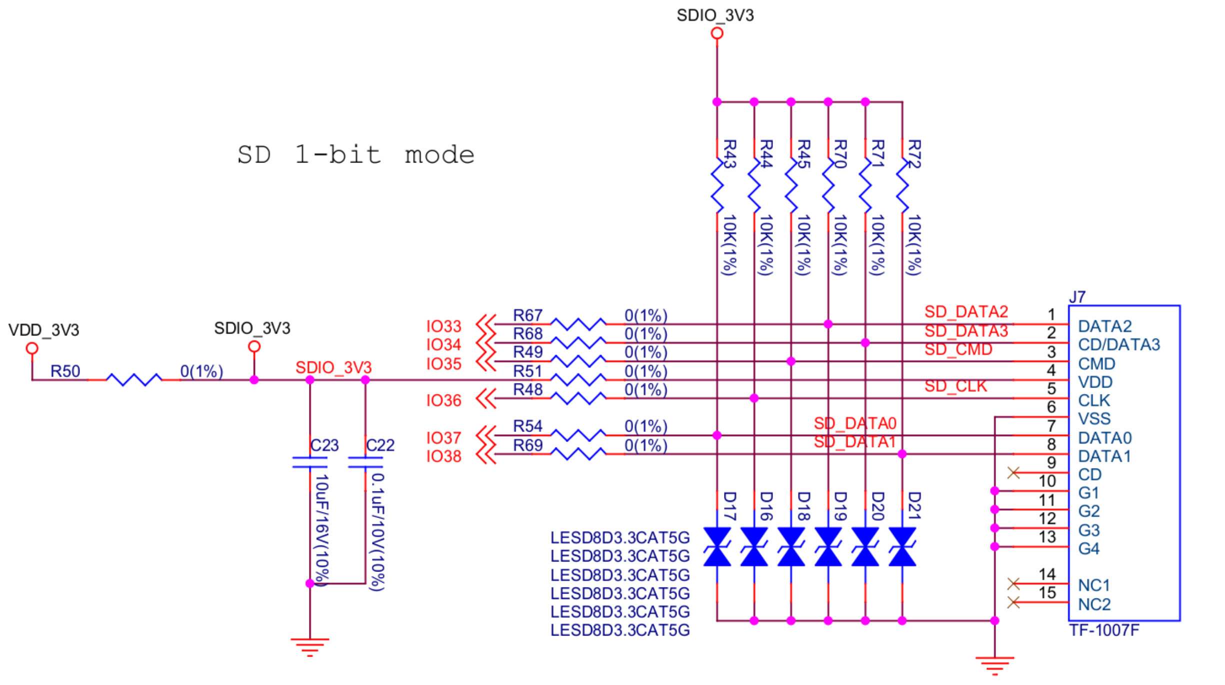


Fig. 25: SD Card Interface Circuit (click to enlarge)

Charging Circuit

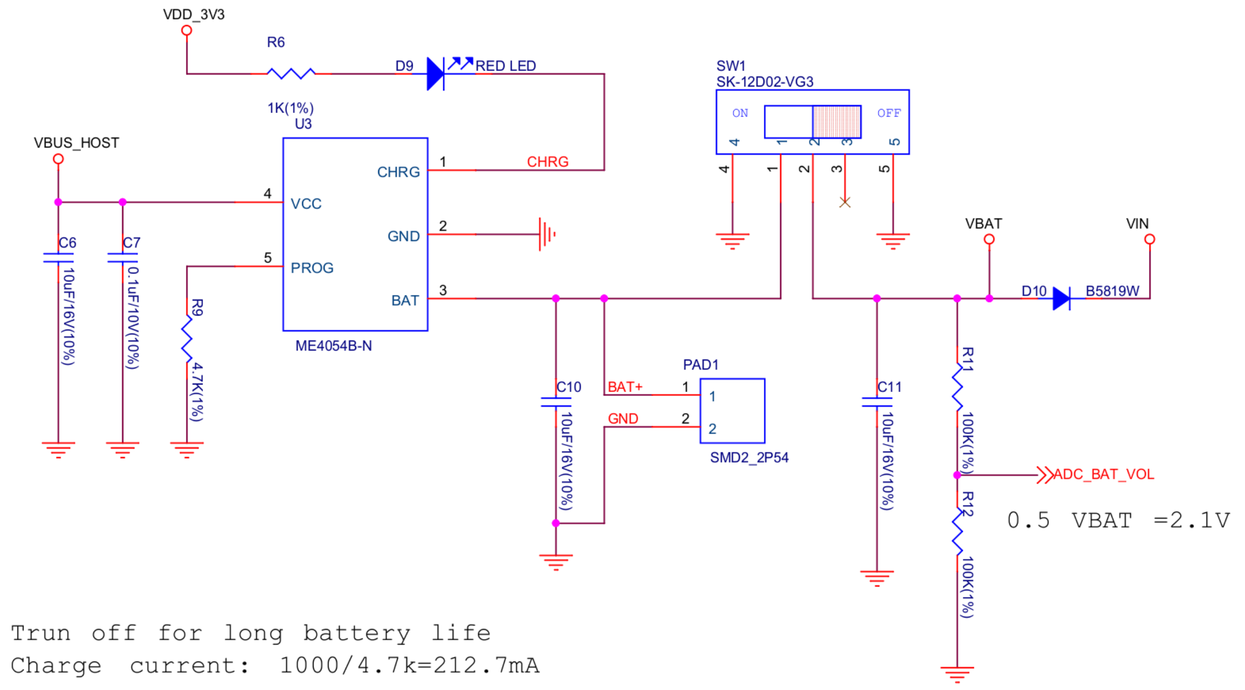


Fig. 26: Charging Circuit (click to enlarge)

Please note that the Type-A male port can be connected to a power adapter that outputs 5 V. When charging the battery, the red indicator LED is on, after fully charged, the red indicator LED is off. When using the charging circuit, please set the battery switch to OFF. The charging current is 212.7 mA.

Pin Layout

Function pin:

No.	ESP32-S3-MINI-1 Pin	Description
1	GPIO18	USB_SEL: Used to switch the USB interface. When high level, the USB_HOST interface is enabled. When low level, the USB_DEV interface is enabled.
2	GPIO19	Connect with USB D-.
3	GPIO20	Connect with USB D+.
4	GPIO15	LED_GREEN: the light is lit when set high level.
5	GPIO16	LED_YELLOW: the light is lit when set high level.
6	GPIO0	BUTTON_OK: OK button, low level when pressed.
7	GPIO11	BUTTON_DW: Down button, low level when pressed.
8	GPIO10	BUTTON_UP: UP button, low level when pressed.
9	GPIO14	BUTTON_MENU: Menu button, low level when pressed.
10	GPIO8	LCD_RET: used to reset LCD, low level to reset.
11	GPIO5	LCD_EN: used to enable LCD, low level to enable.
12	GPIO4	LCD_DC: Used to switch data and command status.
13	GPIO6	LCD_SCLK: LCD SPI Clock.
14	GPIO7	LCD_SDA: LCD SPI MOSI.
15	GPIO9	LCD_BL: LCD backlight control.
16	GPIO36	SD_SCK: SD SPI CLK / SDIO CLK.
17	GPIO37	SD_DO: SD SPI MISO / SDIO Data0.
18	GPIO38	SD_D1: SDIO Data1.
19	GPIO33	SD_D2: SDIO Data2.
20	GPIO34	SD_D3: SD SPI CS / SDIO Data3.
21	GPIO1	HOST_VOL: USB_DEV voltage monitoring, ADC1 channel 0.
22	GPIO2	BAT_VOL: Battery voltage monitoring, ADC1 channel 1.
23	GPIO17	LIMIT_EN: Enable current limiting IC, high level enable.
24	GPIO21	0VER_CURRENT: Current overrun signal, high level means overrun.
25	GPIO12	DEV_VBUS_EN: High level to enable DEV_VBUS power supply.
26	GPIO13	BOOST_EN: High level to enable Boost boost circuit.

Extended pin:

No.	ESP32-S3-MINI-1 Pin	Description
1	GPIO45	FREE_1: Idle, can be customized.
2	GPIO46	FREE_2: Idle, can be customized.
3	GPIO48	FREE_3: Idle, can be customized.
4	GPIO26	FREE_4: Idle, can be customized.
5	GPIO47	FREE_5: Idle, can be customized.
6	GPIO3	FREE_6: Idle, can be customized.

Related Documents

- ESP32-S3 Datasheet (PDF)
- ESP32-S3-MINI-1/1U Datasheet (PDF)
- Espressif Product Selection Tool
- ESP32-S3-USB-OTG Schematic Diagram (PDF)
- ESP32-S3-USB-OTG PCB Layout Drawing (PDF)
- ST7789VW Datasheet (PDF)

1.1.5 ESP32-S3-LCD-EV-BOARD

[⍰⍰]

ESP32-S3-LCD-EV-BOARD is a development board for evaluating and verifying esp32s3 screen interactive applications. It has the functions of touch screen interaction and voice interaction.

ESP32-S3-LCD-Ev-Board

[⍰⍰]

This user guide will help you get started with ESP32-S3-LCD-Ev-Board and will also provide more in-depth information.

The document consists of the following sections:

- *Board Overview*: Overview of the board hardware/software.
- *Start Application Development*: How to set up hardware/software to develop applications.
- *Hardware Reference*: More detailed information about the board's hardware.
- *Hardware Revision Details*: This is the first revision of this board released.
- *Sample Request*: How to get a sample board.
- *Related Documents*: Links to related documentation.

Board Overview

ESP32-S3-LCD-Ev-Board is an ESP32-S3-based development board with a touchscreen. Together with different subboards, ESP32-S3-LCD-Ev-Board can drive LCDs with IIC, SPI, 8080, and RGB interfaces. It houses dual array microphones, supports voice recognition and near/far-field voice wake-up, and features screen and voice interaction. The board caters to development needs for touchscreen products with different resolutions and interfaces.

Feature List

The main features of the board are listed below:

- **Module Embedded:** ESP32-S3-WROOM-1 module with 16 MB flash and 8 MB PSRAM
- **Display:** Compatibility with various subboards and support for displays with `RGB`, `8080`, `SPI`, and `I2C` interfaces. Please refer to *LCD Subboards* for more information
- **Audio:** Audio Codec + ADC amplifier and dual microphones
- **USB:** USB to serial port chip plus USB Type-C download/debug

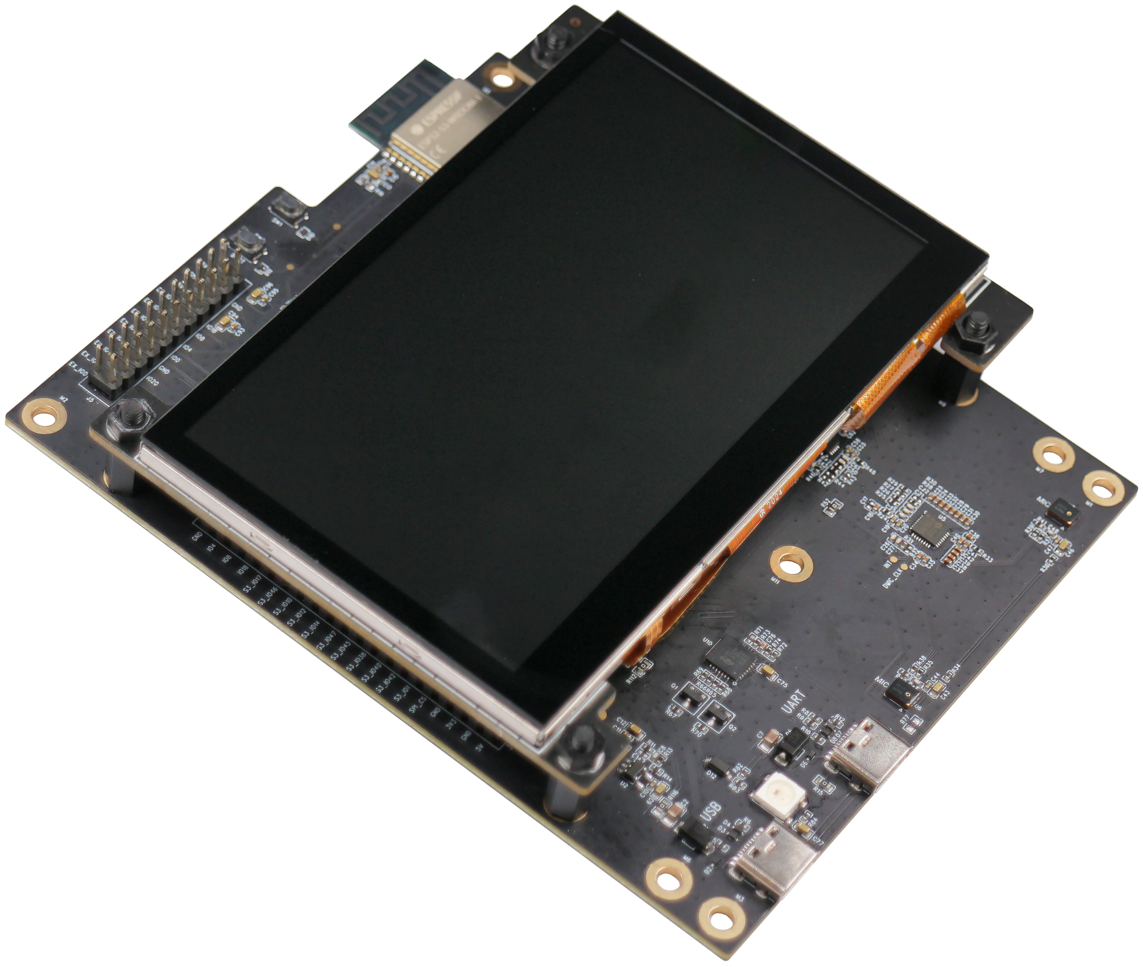


Fig. 27: ESP32-S3-LCD-Ev-Board with ESP32-S3-WROOM-1 Module

Block Diagram

The block diagram below shows the components of ESP32-S3-LCD-Ev-Board and their interconnections.

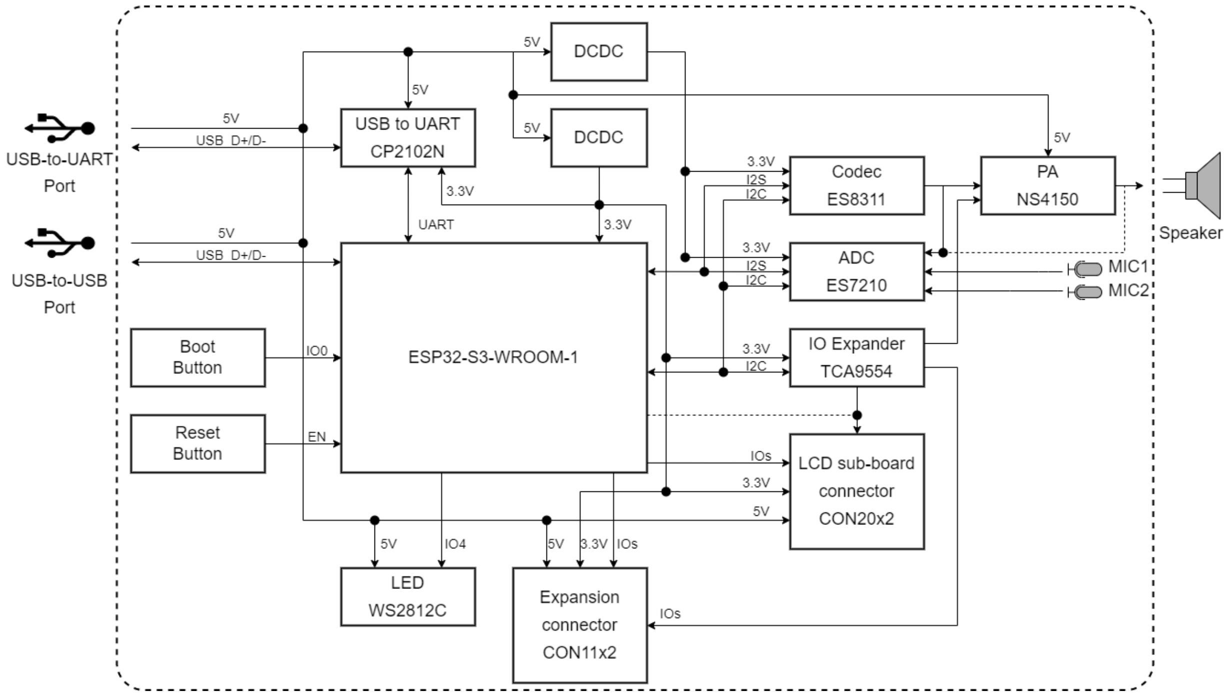


Fig. 28: ESP32-S3-LCD-Ev-Board Block Diagram (Click to Enlarge)

Description of Components

The ESP32-S3-LCD-Ev-Board development board consists of a mainboard and a subboard.

Mainboard

ESP32-S3-LCD-Ev-Board_MB is the core of the kit, which integrates the ESP32-S3-WROOM-1 module and provides ports for connection to the LCD subboard.

The key components of the board are described in a counter-clockwise direction.

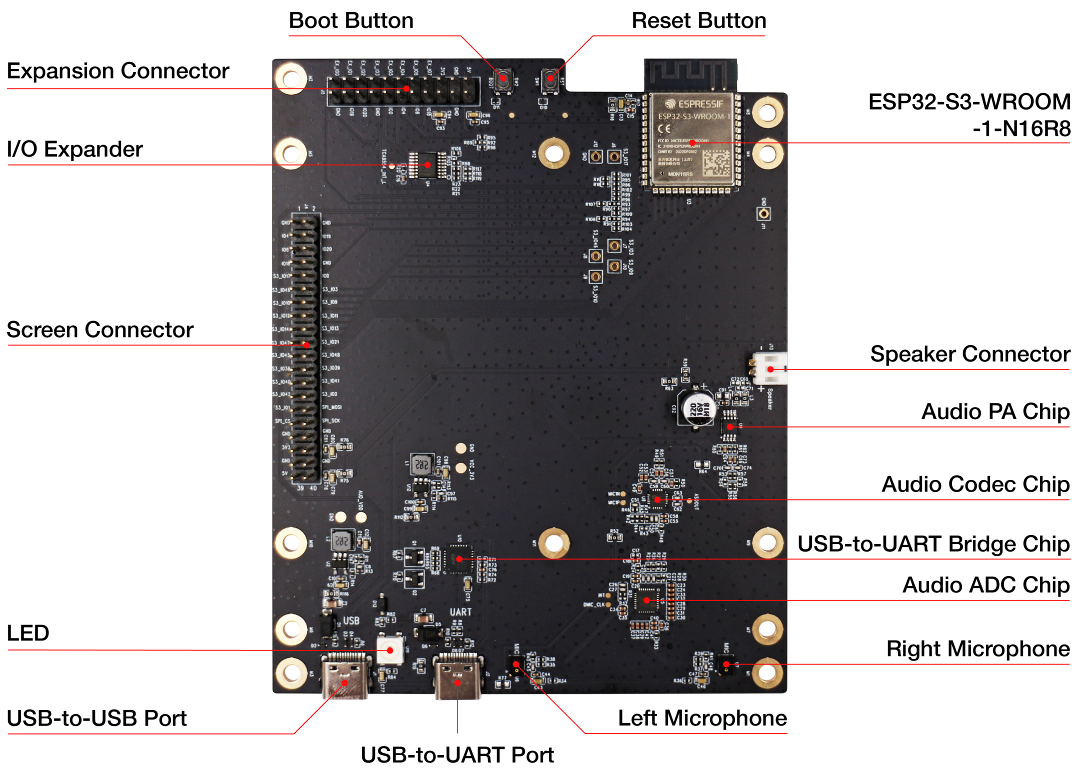


Fig. 29: ESP32-S3-LCD-Ev-Board - Front (Click to Enlarge)

Key Component	Description
ESP32-S3-WROOM-1-N16R8 Module	ESP32-S3-WROOM-1-N16R8 is a generic Wi-Fi + Bluetooth LE MCU module that is built around the ESP32-S3 series of SoCs. It is integrated with 16 MB flash and 8 MB PSRAM. On top of a rich set of peripherals, the acceleration for neural network computing and signal processing workloads provided by the SoC makes the module an ideal choice for a wide variety of application scenarios related to AI and Artificial Intelligence of Things (AIoT).
Reset Button	Press this button to reset the system.
Boot Button	Holding down the Boot key and momentarily pressing the Reset key initiates the firmware upload mode. Then you can upload firmware through the serial port or USB.
Expansion Connector	Provides connections for all I/O expander pins, all power supply pins, and some module pins.
I/O Expander	TCA9554 is a device that provides 8 bits of general purpose parallel I/O expansion. It controls the I/O mode and level via two-line bidirectional I2C bus, offering a simple solution when additional I/Os are needed.
LCD Board Connector	Three different types of LCD subboards can be connected via connectors with 2.54 mm pitch.
LED	Supports configuring the RGB LED display to indicate status or behavior.
USB-to-USB Port	Provides power to the entire system (choose either USB-to-USB or USB-to-UART port). It is recommended to use at least a 5V/2A power adapter to ensure stable power supply. Used for USB communication between the PC and the ESP32-S3-WROOM-1 module.
USB-to-UART Port	Provides power to the entire system (choose either USB-to-USB or USB-to-UART port). It is recommended to use at least a 5V/2A power adapter to ensure stable power supply. Used for serial communication between the PC side and the ESP32-S3-WROOM-1 module.
Left Microphone	On-board microphone, connected to ADC.
Right Microphone	On-board microphone, connected to ADC.
Audio ADC Chip	*ES7210 <http://www.everest-semi.com/pdf/ES7210%20PB.pdf>* is a high performance, low power 4-channel audio ADC for applications of microphone arrays. Featuring Acoustic Echo Cancellation (AEC), it is an ideal choice for music and voice applications.
USB-to-UART Bridge Controller	CP2102N, the single-chip USB-to-UART bridge controller, provides up to 3 Mbps connection for software download and debugging.
Audio Codec Chip	ES8311 is a low-power mono audio codec that includes a single-channel ADC and DAC, low noise pre-amplifier, headphone driver, digital audio, analog mixing, and gain function. It connects to the ESP32-S3-WROOM-1 module via I2S and I2C buses to process audio through hardware instead of the audio application.
Audio Amplifier	NS4150 is a low EMI, 3 W mono class D audio amplifier used to drive speakers by amplifying the audio signal from the audio codec chip.
Speaker Connector	External speaker playback is possible with the help of the audio amplifier.

LCD Subboards

The mainboard can be used together with three different kinds of subboards:

Board Name	Display (Inch)	Resolution (Px)	LCD Driver (Interface)	Touch Driver
ESP32-S3-LCD_Ev_Board_SUB1	0.96	128 x 64	SSD1315 (I2C)	N/A
	2.40	320 x 240	ST7789V (SPI)	XTP2046
ESP32-S3-LCD_Ev_Board_SUB2	3.50	480 x 320	ST7796S (8080)	GT911
	3.95	480 x 480	GC9503CV (RGB)	FT5x06
ESP32-S3-LCD_Ev_Board_SUB3	4.30	800 x 480	Unknown (RGB)	GT1151

- **ESP32-S3-LCD-Ev-Board_SUB1** subboard has two interfaces, which support connection to a 2.4-inch display with the SPI interface or a 0.96-inch display with the I2C interface. This board is not yet configured, so it is not further explained here.
- **ESP32-S3-LCD-Ev-Board_SUB2** subboard has two interfaces, which support connection to a display with the RGB interface or a display with the 8080 parallel interface. The current subboard has a 3.95-inch touchscreen with the RGB565 interface and 480x480 resolution. The LCD driver IC is GC9503CV and the touchscreen driver IC is FT5x06.
- **ESP32-S3-LCD-Ev-Board_SUB3** subboard only supports a 4.3-inch touchscreen with the RGB565 interface and 800x480 resolution. The touchscreen driver IC is GT1151.

Software Support

The ESP32-S3-LCD-Ev-Board development framework is ESP-IDF. ESP-IDF is a FreeRTOS-based SoC development framework with a bunch of components including LCD, ADC, RMT, and SPI. An example is provided for ESP32-S3-LCD-Ev-Board under the folder Examples. You can configure project options by entering `idf.py menuconfig` in the example directory.

Note:

- ESP-IDF v5.0 and above are recommended for development.
- ESP32-S3 RGB driver only supports the 16-bit RGB565 and 8-bit RGB888 interface.
- To configure ESP-IDF with the 120 MHz octal PSRAM using patches, please see here.

Start Application Development

This section provides instructions on how to do hardware and software setup and flash firmware onto the board to develop your own application.

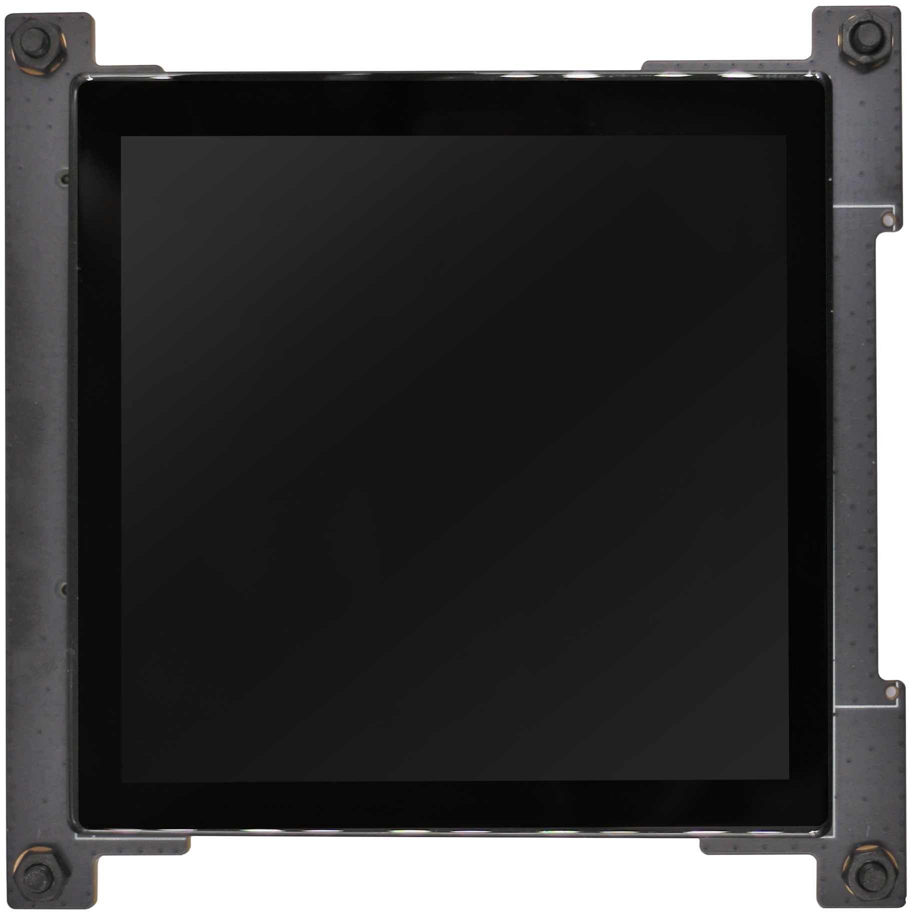


Fig. 30: ESP32-S3-LCD-Ev-Board_SUB2 - Front (Click to Enlarge)

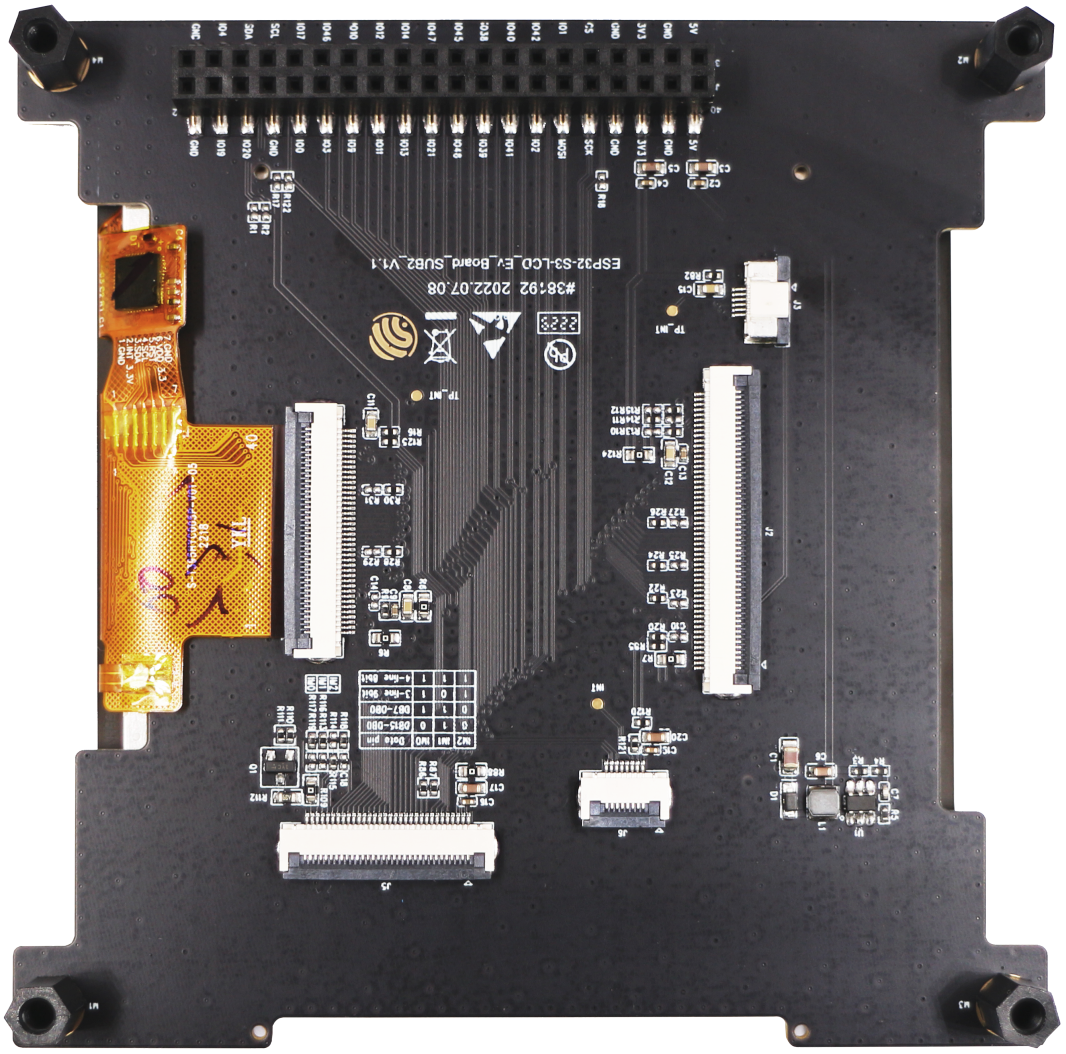


Fig. 31: ESP32-S3-LCD-Ev-Board_SUB2 - Front (Click to Enlarge)

Fig. 32: ESP32-S3-LCD-Ev-Board_SUB3 - Front (Click to Enlarge)

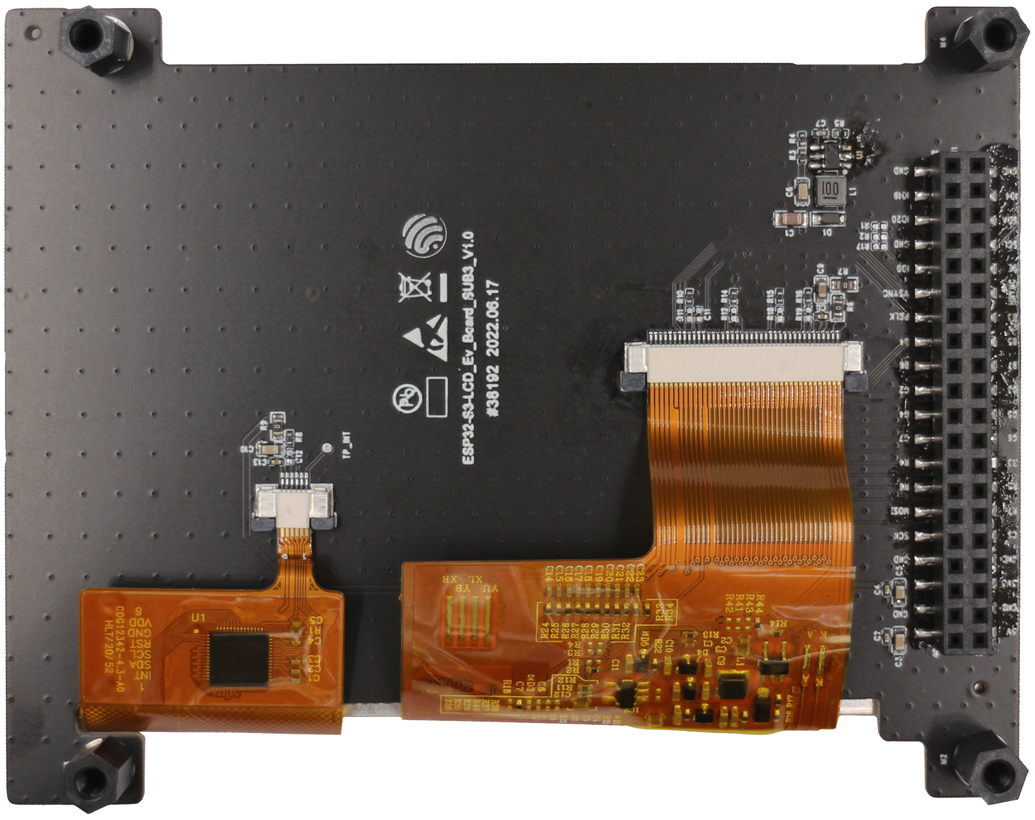


Fig. 33: ESP32-S3-LCD-Ev-Board_SUB3 - Back (Click to Enlarge)

Required Hardware

- 1 x ESP32-S3-LCD-Ev-Board_MB
- 1 x LCD subboard
- 1 x USB 2.0 cable (standard Type-A to Type-C)
- 1 x PC (Windows, Linux, or macOS)

Note: Please make sure to use the appropriate USB cable. Some cables can only be used for charging, not for data transfer or program flashing.

Optional Hardware

- 1 x Speaker

Hardware Setup

Prepare the board for loading of the first sample application:

1. Connect the LCD subboard to the **LCD Board Connector**.
2. Plug in the USB cable to connect the PC with the board.
3. The LCD lights up and you can start to interact with it.

Now the board is ready for software setup.

Software Setup

To learn how to quickly set up your development environment, please go to Get Started > Installation.

For more software information on developing applications, please go to *Software Support*.

Hardware Reference

This section provides more detailed information about the board's hardware.

GPIO Allocation

The table below provides the allocation of GPIOs exposed on terminals of ESP32-S3-WROOM-1 module to control specific components or functions of the board.

Table 1: ESP32-S3-WROOM-1 GPIO Allocation

Pin	Pin Name	Function
1	GND	GND
2	3V3	Power supply
3	EN	RESET

Continued on next page

Table 1 – continued from previous page

Pin	Pin Name	Function
4	IO4	LED
5	IO5	I2S_MCLK
6	IO6	I2S_CODEC_DSDIN
7	IO7	I2S_LRCK
8	IO15	I2S_ADC_SDOUT
9	IO16	I2S_SCLK
10	IO17	LCD_DE
11	IO18	I2C_SCL
12	IO8	I2C_SDA
13	IO19	USB_D-
14	IO20	USB_D+
15	IO3	LCD_VSYNC
16	IO46	LCD_HSYNC
17	IO9	LCD_PCLK
18	IO10	LCD_DATA0
19	IO11	LCD_DATA1
20	IO12	LCD_DATA2
21	IO13	LCD_DATA3
22	IO14	LCD_DATA4
23	IO21	LCD_DATA5
24	IO47	LCD_DATA6
25	IO48	LCD_DATA7
26	IO45	LCD_DATA8
27	IO0	BOOT
28	IO35	No connection
29	IO36	No connection
30	IO37	No connection
31	IO38	LCD_DATA9
32	IO39	LCD_DATA10
33	IO40	LCD_DATA11
34	IO41	LCD_DATA12
35	IO42	LCD_DATA13
36	RXD0	UART_RXD0
37	TXD0	UART_TXD0
38	IO2	LCD_DATA14
39	IO1	LCD_DATA15
40	GND	GND
41	EPAD	GND

The pins on the I/O expander connected to the module can be used for different functions.

Table 2: I/O Expander GPIO Allocation

IO Expander Pin	Pin Name	Function
1	A0	GND
2	A1	GND
3	A2	GND
4	P0	PA_CTRL
5	P1	LCD_SPI_CS
6	P2	LCD_SPI_SCK
7	P3	LCD_SPI_MOSI
8	GND	GND
9	P4	Free
10	P5	Free
11	P6	Free
12	P7	Free
13	INT	No connection
14	SCL	I2C_SCL
15	SDA	I2C_SDA
16	VCC	Supply voltage

Power Distribution

Power Supply over USB

There are two ways to power the development board via USB power port.

- Via `USB-to-USB` port

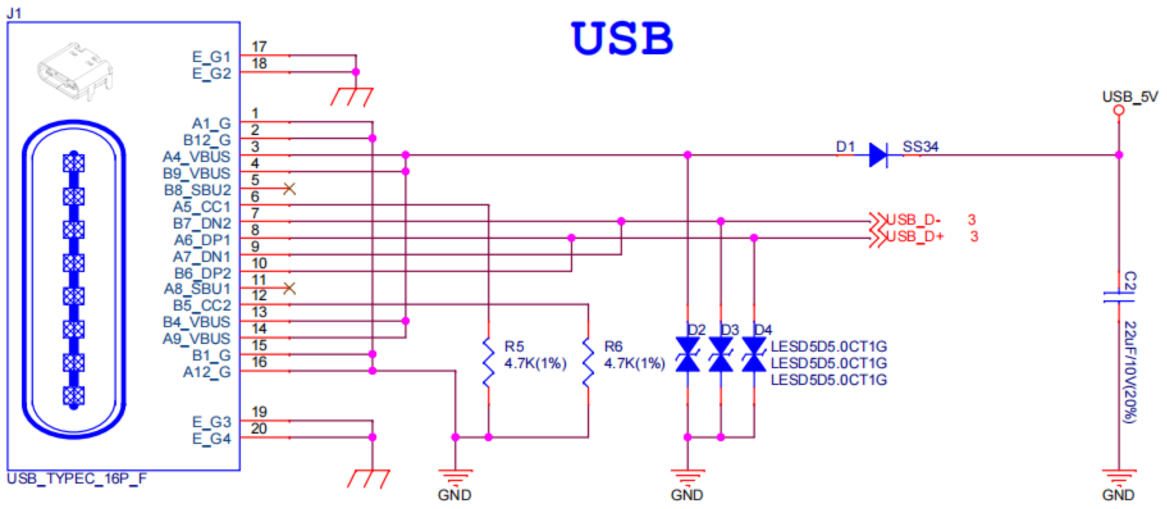


Fig. 34: ESP32-S3-LCD-Ev-Board - USB-to-USB Power Supply

- Via `USB-to-UART` port

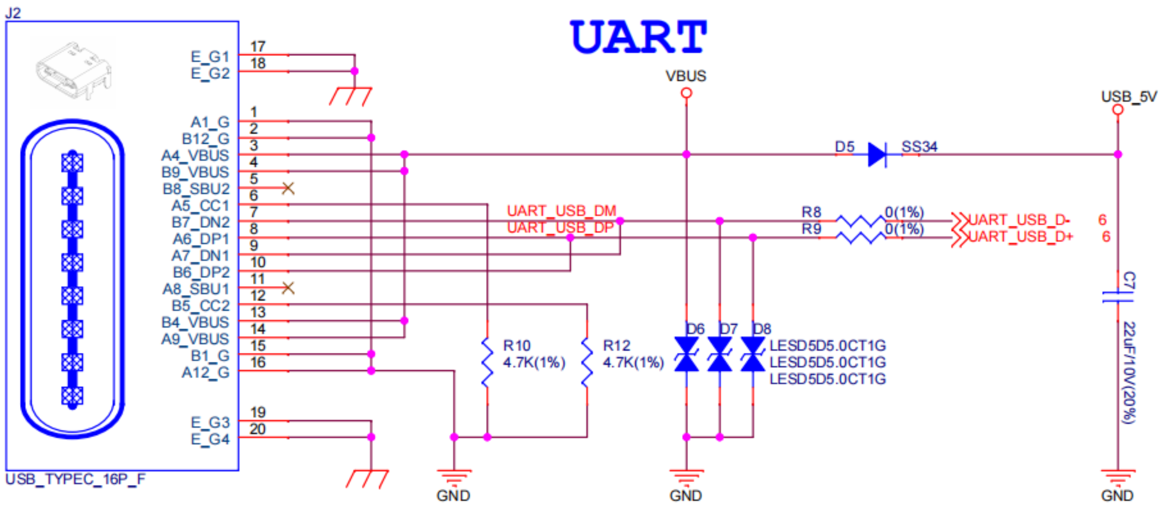


Fig. 35: ESP32-S3-LCD-Ev-Board - USB-to-UART Power Supply

Independent Audio and Digital Power Supply

ESP32-S3-LCD-Ev-Board features independent power supplies for the audio components and ESP module. This should reduce noise in the audio signal from digital components and improve the overall performance of the components.

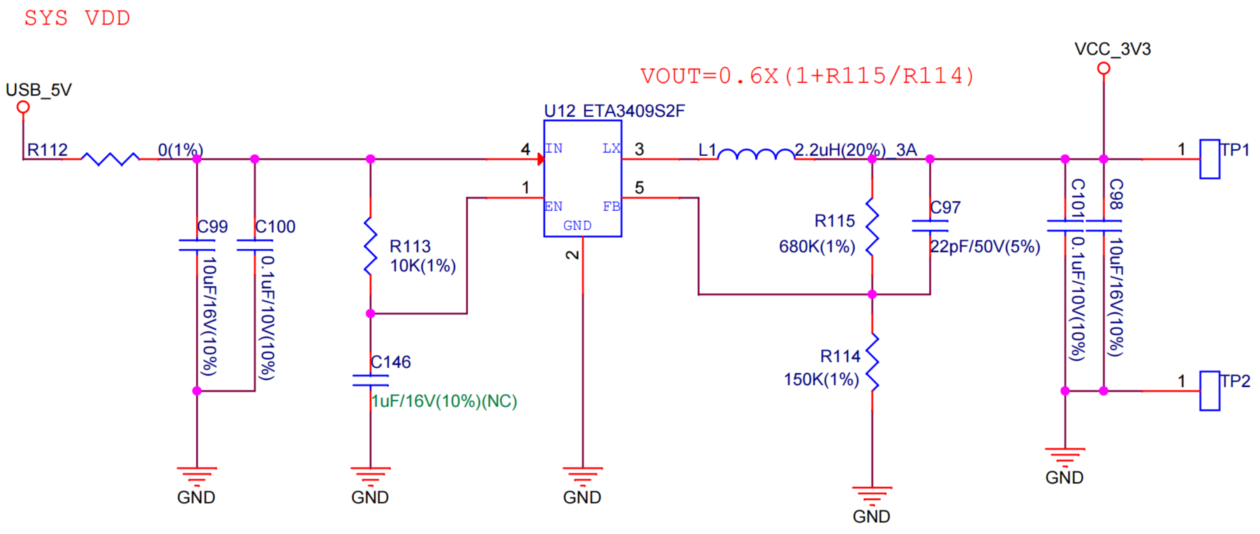


Fig. 36: ESP32-S3-LCD-Ev-Board - Digital Power Supply

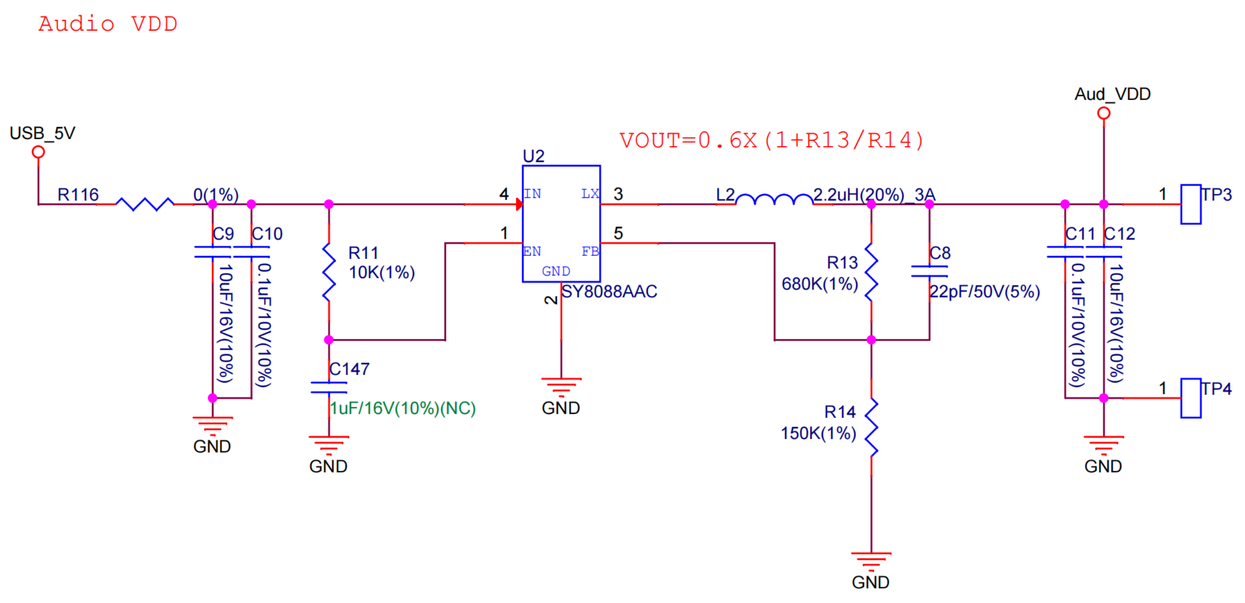


Fig. 37: ESP32-S3-LCD-Ev-Board - Audio Power Supply

AEC Path

The acoustic echo cancellation (AEC) path provides reference signals for AEC algorithm.

ESP32-S3-LCD-Ev-Board provides two compatible echo reference signal source designs. One is Codec (ES8311) DAC output (DAC_AOUTLP/DAC_AOUTLP), the other is PA (NS4150) output (PA_OUT+/PA_OUT+). The former is a default and the recommended selection. Resistors R54 and R56 shown in the figure below should not be installed.

The echo reference signal is collected by ADC_MIC3P/ADC_MIC3N of ADC (ES7210) and then sent back to ESP32-S3 for AEC algorithm.

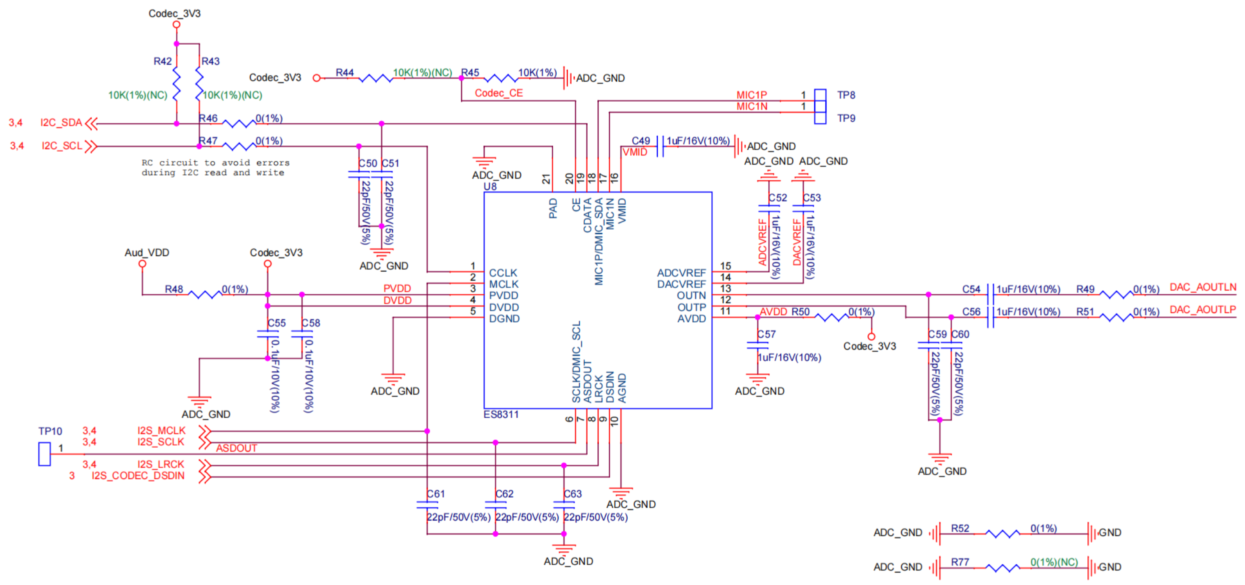


Fig. 38: ESP32-S3-LCD-Ev-Board - AEC Codec DAC Output (Click to Enlarge)

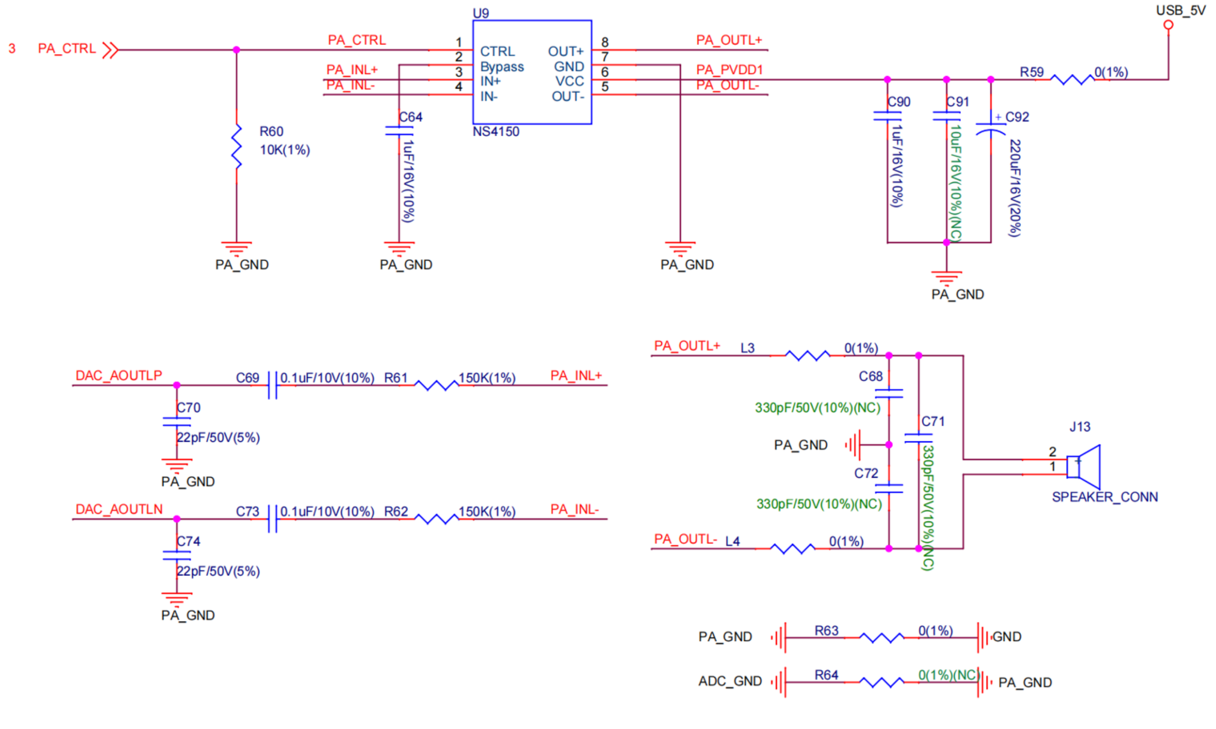
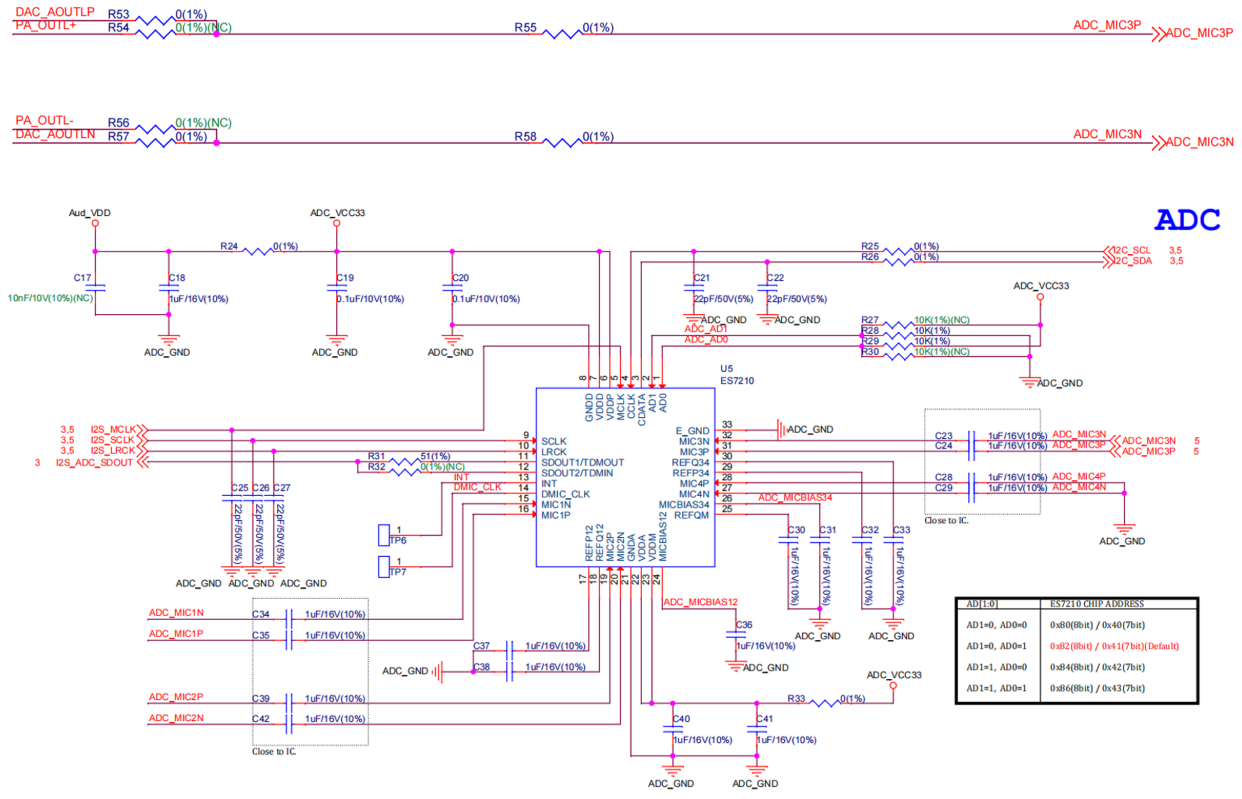


Fig. 39: SP32-S3-LCD-Ev-Board - AEC PA Output (Click to Enlarge)



AD[1:0]	ES7210 CHIP ADDRESS
AD1=0, AD0=0	0x80(8bit) / 0x40(7bit)
AD1=0, AD0=1	0x82(8bit) / 0x41(7bit)(Default)
AD1=1, AD0=0	0x84(8bit) / 0x42(7bit)
AD1=1, AD0=1	0x86(8bit) / 0x43(7bit)

Fig. 40: ESP32-S3-LCD-Ev-Board - AEC Reference Signal Collection (Click to Enlarge)

Hardware Setup Options

Automatic Download

There are two ways to put the development board into the download mode.

- Press the Boot and Reset buttons. Release the Reset button first and then the Boot button.
- The download is performed automatically by the software. The software uses the DTR and RTS signals from the serial port to control the status of the EN and IO0 pins.

Hardware Revision Details

No previous revisions.

Sample Request

This development board can be used to assess high performance HMI solutions and is not yet available for sale. For sample requests, please contact us at sales@espressif.com.

Related Documents

- ESP32-S3 Datasheet
- ESP32-S3-WROOM-1 Datasheet
- ESP Product Selector
- ESP32-S3-LCD-EV-BOARD-MB Schematics
- ESP32-S3-LCD-EV-BOARD-MB PCB Layout
- ESP32-S3-LCD-EV-BOARD-SUB1 Schematics
- ESP32-S3-LCD-EV-BOARD-SUB1 PCB Layout
- ESP32-S3-LCD-EV-BOARD-SUB2 Schematics
- ESP32-S3-LCD-EV-BOARD-SUB2 PCB Layout
- ESP32-S3-LCD-EV-BOARD-SUB3 Schematics
- ESP32-S3-LCD-EV-BOARD-SUB3 PCB Layout
- TCA9554 Datasheet
- ES7210 Datasheet
- ES8311 Datasheet

For further design documentation for the board, please contact us at sales@espressif.com.

1.1.6 ESP32-S2-Kaluga-1

[□□]

The ESP32-S2-Kaluga-1 kit v1.3 is a development kit by Espressif.

Application examples for this kit can be found at Examples .

ESP32-S2-Kaluga-1

[□□]

For the latest version, please go to ESP32-S2-Kaluga-1 Kit.

Reference Documentation

[□□]

Schematic

- ESP32-S2-Kaluga-1 v1.2 Mainboard (PDF)
- ESP-LyraT-8311A v1.2 Audio Board (PDF)
- ESP-LyraP-LCD32 v1.2 Screen (PDF)
- ESP-LyraP-TOUCHA v1.2 Touchpad (PDF)
- ESP-LyraP-CAM v1.2 Camera (PDF)

Datasheet

- ESP32-S2 Datasheet
- LCD ST7789□esp32-s2-kaluga-1/datasheet/LCD_ST7789.pdf
- Camera OV2640□esp32-s2-kaluga-1/datasheet/Camera_OV2640.pdf
- Audio ES8311□esp32-s2-kaluga-1/datasheet/Audio_ES8311.pdf

1.1.7 ESP32-LCDKit

[□□]

ESP32-LCDKit is an HMI (Human Machine Interface) development board with the ESP32-DevKitC at its core.

ESP32-LCDKit

[中文]

Overview

ESP32-LCDKit is an HMI (Human Machine Interface) development board with the ESP32-DevKitC at its core. ESP32-LCDKit is integrated with such peripherals as SD-Card, DAC-Audio, and can be connected to an external display. The board is mainly used for HMI-related development and evaluation.Development board reserved screen interface type: SPI serial interface, 8-bit parallel interface, 16-bit parallel interface.

You may find HMI-related examples running with ESP32-LCDKit in HMI Example.

For more information on ESP32, please refer to ESP32 Series Datasheet.

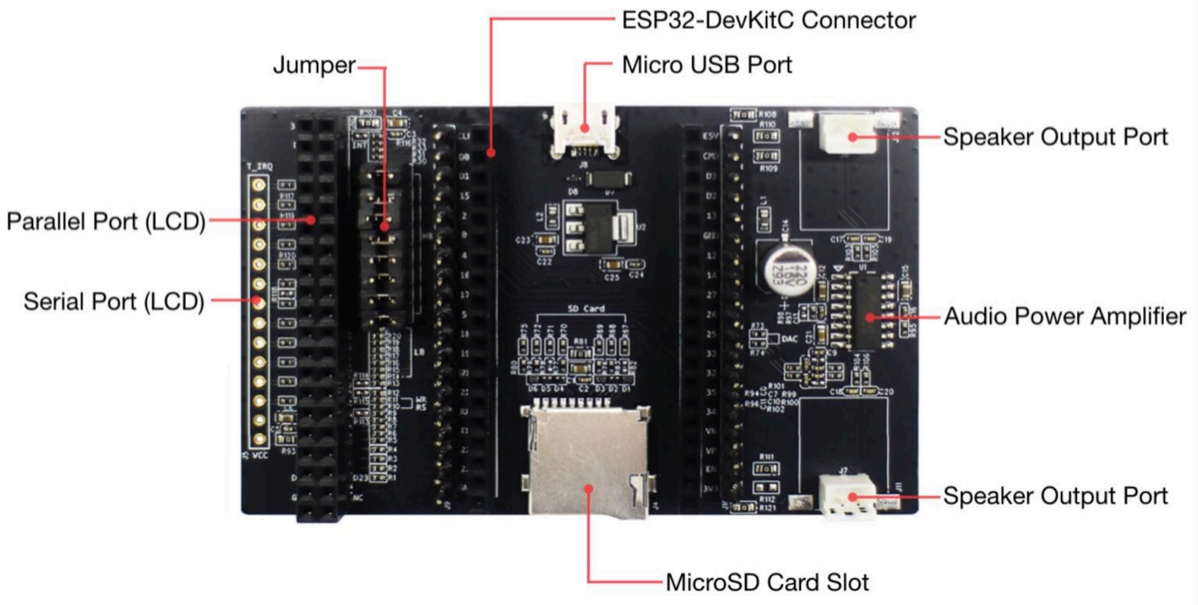


Fig. 41: ESP32-LCDKit

Block Diagram and PCB Layout

Block Diagram

The figure below shows the block diagram for ESP32-LCDKit.

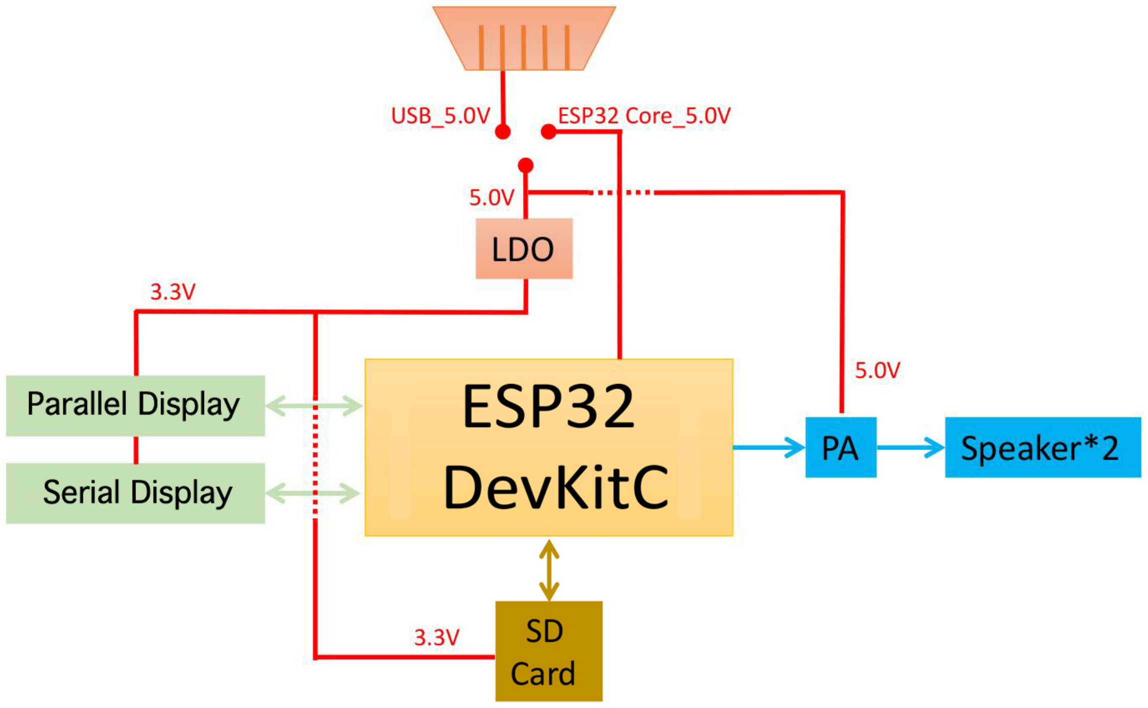


Fig. 42: ESP32-LCDKit Block Diagram

PCB Layout

The figure below shows the layout of ESP32-LCDKit PCB.

Descriptions of PCB components are shown in the following table:

Components	Description
Display connection module	Allows to connect serial or parallel LCD displays (8/16 bit)
ESP32 DevKitC connection module	Offers connection to an ESP32 DevKitC development board
SD-Card module	Provides an SD-Card slot for memory expansion
DAC-Audio module	Features an audio power amplifier and two output ports for external speakers

Functional Modules

This section introduces the functional modules (interfaces) of ESP32-LCDKit and their hardware schematics.

- Schematic
- PCB Layout

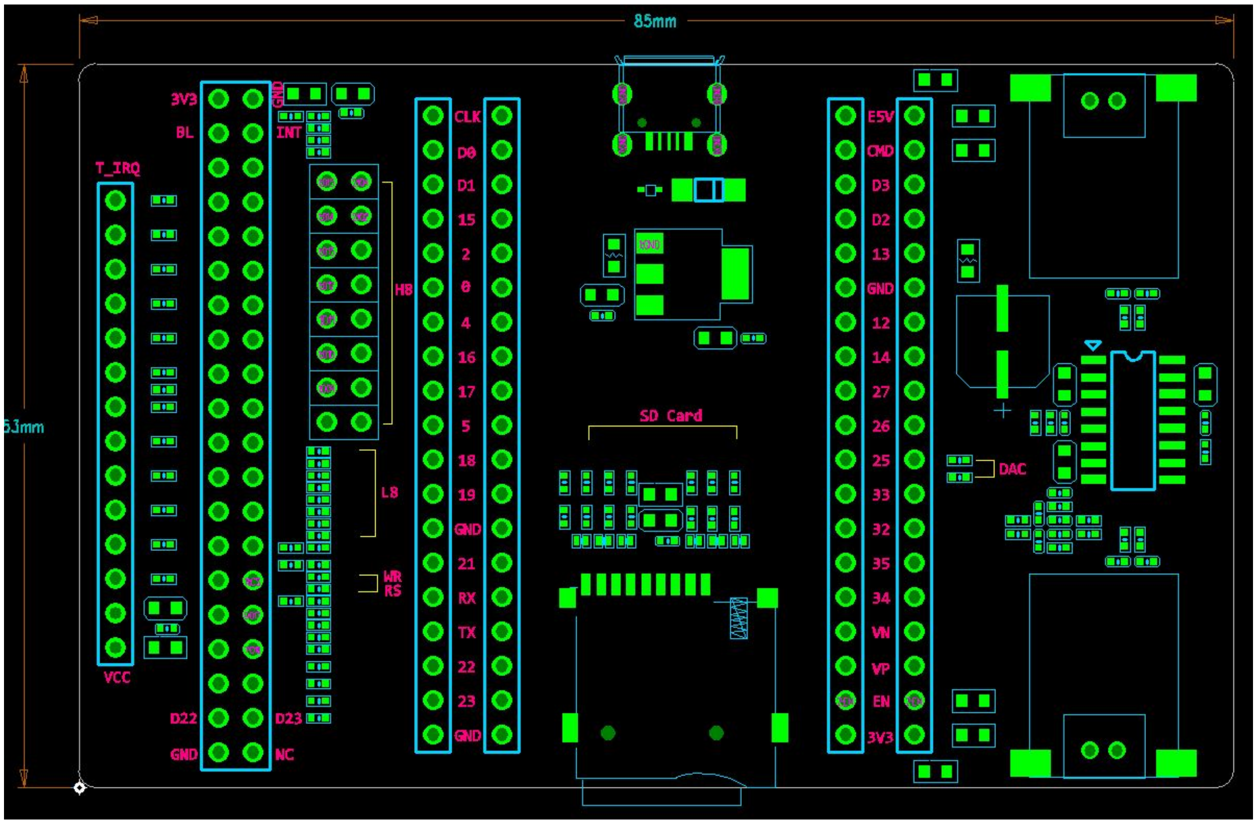


Fig. 43: ESP32-LCDKit PCB Layout

ESP32 DevKitC Connection Module

For the HMI-related development with ESP32-LCDKit, you also need the ESP32 DevKitC development board.

The figure below shows the schematics for the ESP32 DevKitC connection module.

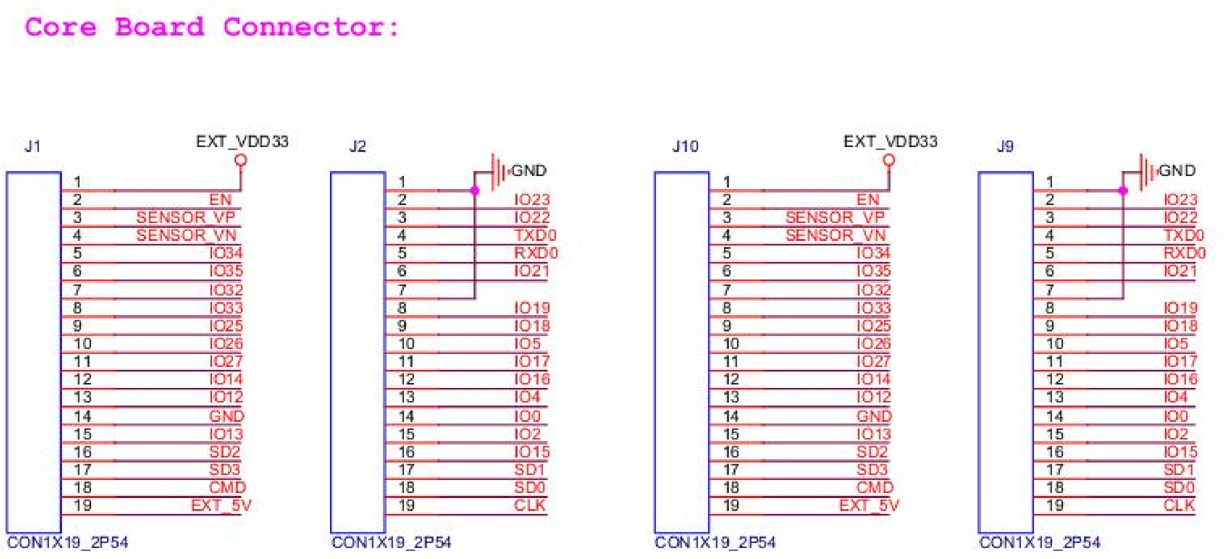


Fig. 44: ESP32 DevKitC Connection Module

Power Supply Management Module

The figure below shows the schematics for the USB power supply management module.

Display Connection Module

The display connection module supports the following interfaces:

- SPI serial interface
- 8-bit parallel interface
- 16-bit parallel interface

With this module, you can connect ESP32-LCDKit to an external display and interact with the pre-programmed GUI if the display has a touchscreen.

The figure below shows the schematics for this module.

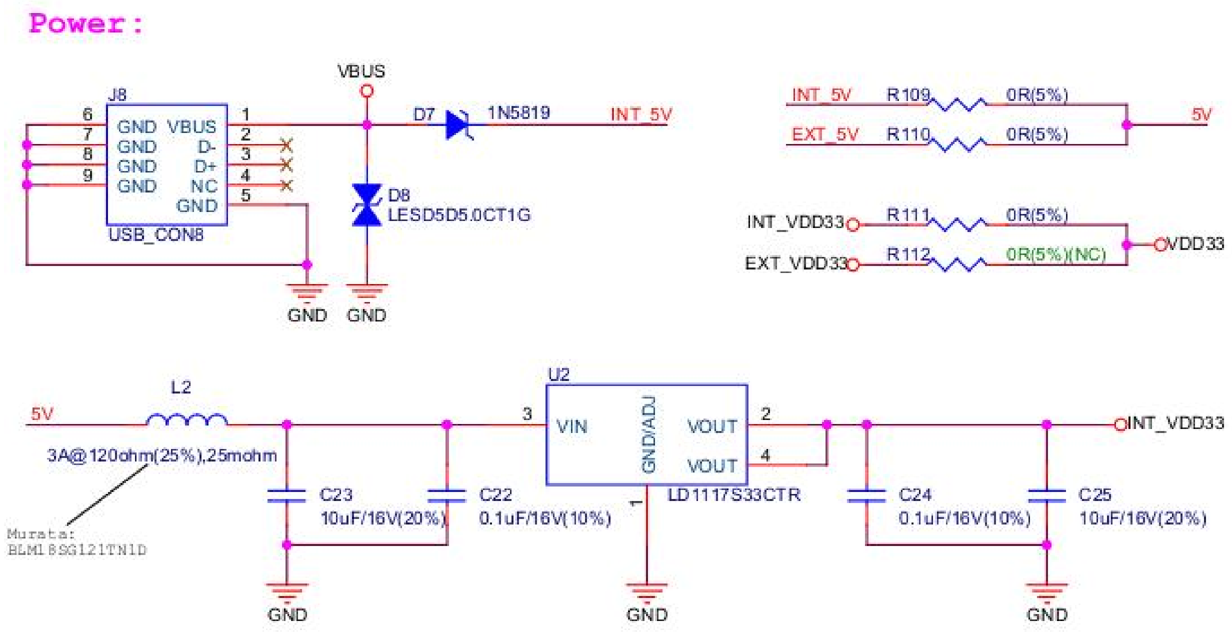


Fig. 45: ESP32-LCDKit Power Supply Module

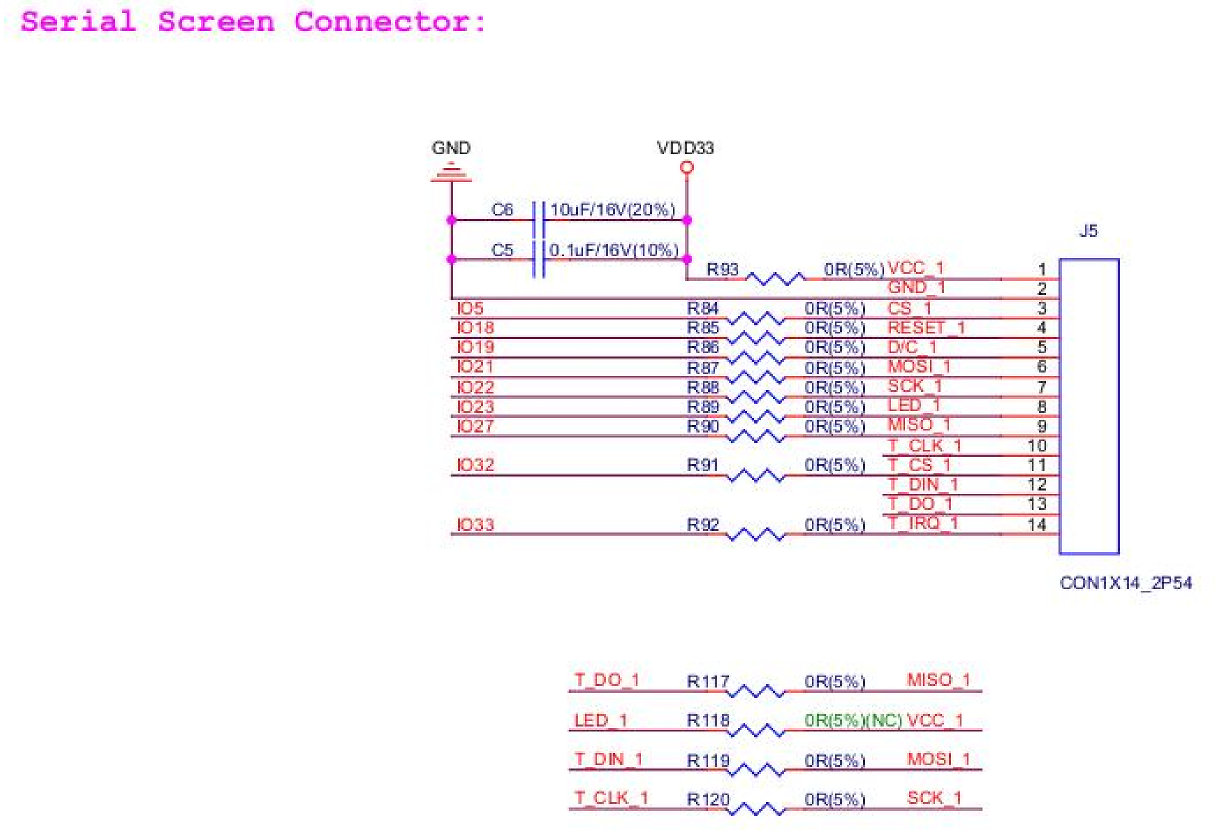


Fig. 46: ESP32-LCDKit Display Connection Module

SD-Card and DAC-Audio Modules

The SD-Card module provides an SD Card slot for memory expansion. The DAC-Audio module features the MIX3006 power amplifier and two output ports for connection of external speakers.

The figure below shows the schematics for the SD-Card and DAC-Audio modules.

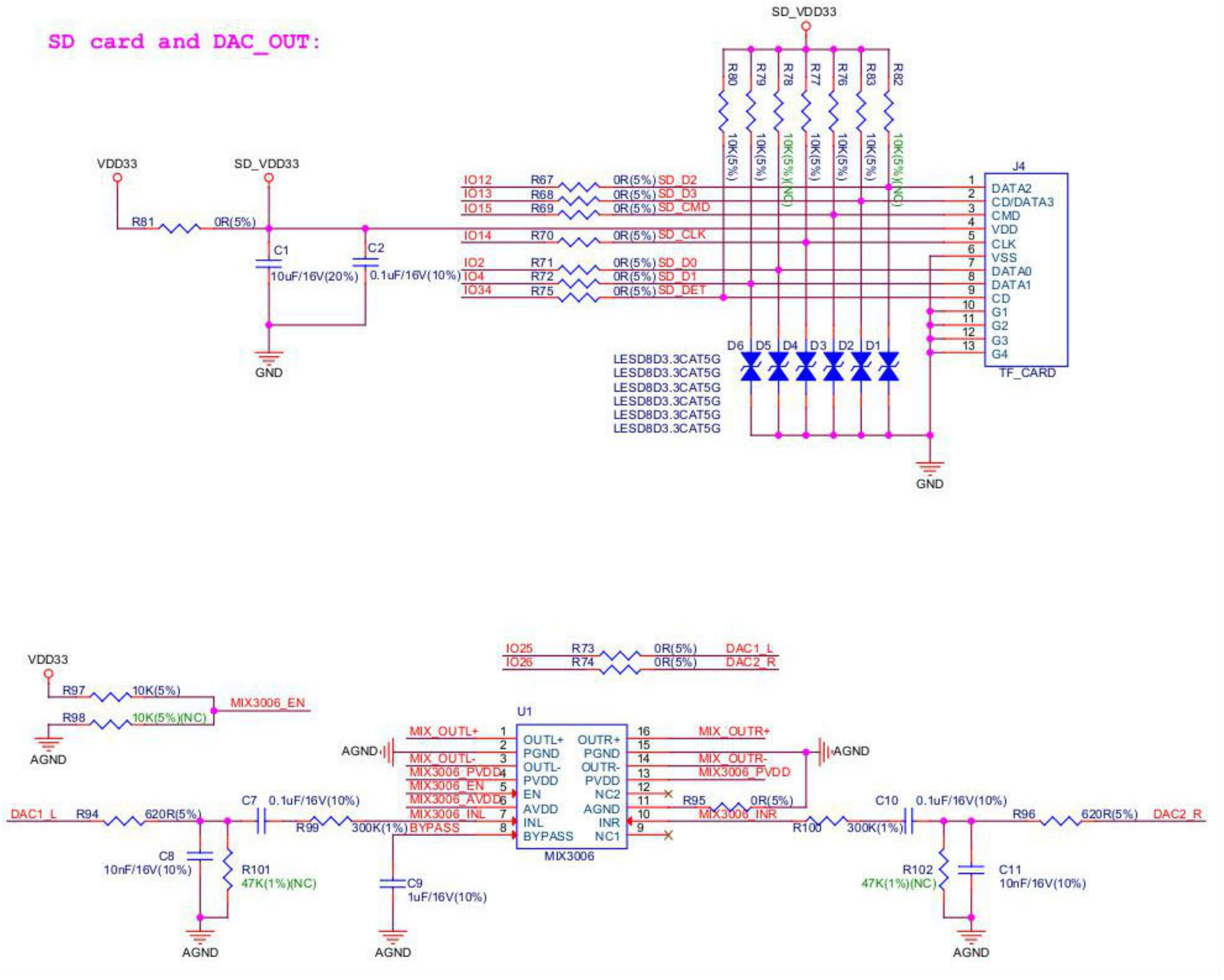


Fig. 47: SD-Card and DAC-Audio Modules

Related Documents

- ESP32-LCDKit Schematic
- ESP32-LCDKit PCB Layout

1.1.8 ESP-Prog

[??]

ESP-Prog is one of Espressif's development and debugging tools, with functions including automatic firmware downloading, serial communication, and JTAG online debugging.

ESP-Prog

[??]

This user guide will help you get started with ESP-Prog and will also provide more in-depth information.

ESP-Prog is one of Espressif's development and debugging tools, with functions including automatic firmware downloading, serial communication, and JTAG debugging. ESP-Prog's automatic firmware downloading and serial communication functions are supported on ESP8266, ESP32, ESP32-S2, ESP32-S3, and ESP32-C3, while the JTAG debugging is supported only on ESP32, ESP32-S2, ESP32-S3, and ESP32-C3.

ESP-Prog can be easily connected to a PC with the use of only one USB cable. Then, the PC can identify the board's Program and JTAG interfaces (functions) by their port numbers.

Given that the power supply voltage may vary on different user boards, either of the ESP-Prog interfaces can provide 5 V or 3.3 V power supply through pin headers, in order to ensure power compatibility. Power on ESP-Prog could be toggled between 3.3 V and 5 V, but the RX/TX & JTAG signals will always be at the 3.3 V level.

The document consists of the following major sections:

- *Getting started*: Overview of the board and hardware/software setup instructions to get started.
- *Hardware Reference*: More detailed information about the board's hardware.
- *Related Documents*: Links to related documentation.

Getting Started

This section provides a brief introduction of ESP-Prog on how to do the initial hardware setup.

Description of Components

The key components of the board are described in a clockwise direction.

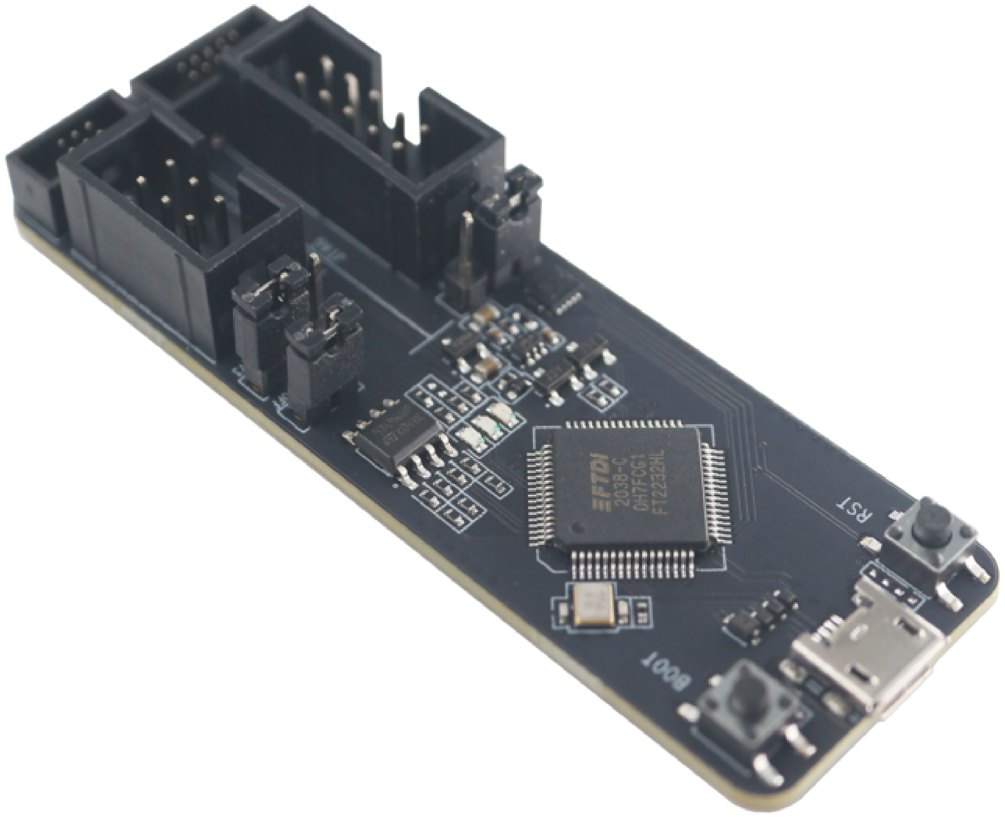


Fig. 48: ESP-Prog (click to enlarge)

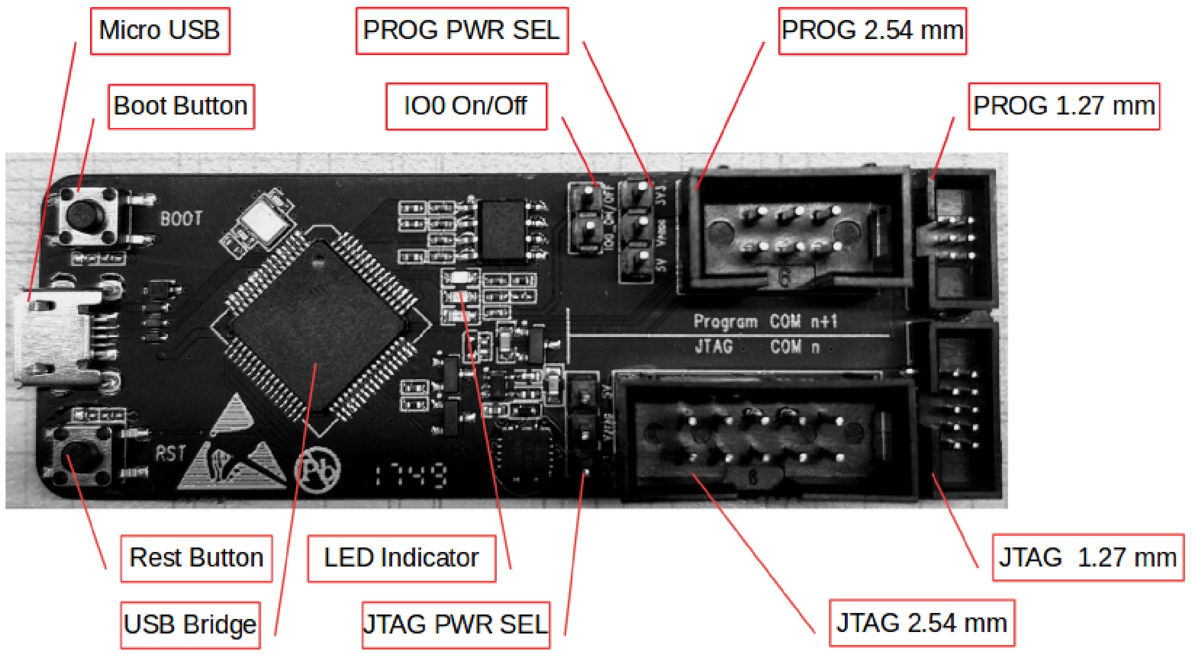


Fig. 49: ESP-Prog - front (click to enlarge)

Key Component	Description
Micro USB	Interface between PC and ESP-Prog.
Boot Button	Download button. Holding down boot and then pressing reset initiates Firmware Download mode for downloading firmware through the serial port.
IO0 On/Off	Pin Header to set the state of GPIO0 strapping pin.
PROG PWR SEL	Pin Header to select power input for the Program interface.
PROG 2.54 mm	Program interface with 2.54 mm (0.1”) pin pitch.
PROG 1.27 mm	Program interface with 1.27 mm (0.05”) pin pitch.
JTAG 1.27 mm	JTAG interface with 1.27 mm (0.05”) pin pitch.
JTAG 2.54 mm	JTAG interface with 2.54 mm (0.1”) pin pitch.
JTAG PWR SEL	Pin Header to select power input for the JTAG interface.
LED Indicator	LED to indicate ESP-Prog state. There are three LED modes: red, green, and blue. The red LED lights up when the system is connected to the 3.3 V power. The green LED lights up when ESP-Prog is downloading data. The blue LED lights up when ESP-Prog is receiving data.
USB Bridge	Single USB-to-UART bridge chip provides up to 3 Mbps of transfer rate.
Reset Button	Press this button to restart the system.

Start Application Development

Before powering up your board, please make sure that it is in good condition with no obvious signs of damage.

Required Hardware

- ESP-Prog
- USB 2.0 cable (Standard-A to Micro-B)
- Computer running Windows, Linux, or macOS
- Dupont lines or flat cables provided by Espressif for connecting the development board to ESP-Prog

Note: Be sure to use an appropriate USB cable. Some cables are for charging only and do not provide the needed data lines nor work for programming the boards.

Hardware Setup

1. Connect the ESP-Prog board and the PC USB port via a USB cable.
2. The PC then detects the two ports of ESP-Prog, indicating that the board is connected successfully. If the ports are not detected install the FT2232HL chip driver on your PC.
3. Select the output power voltage for the Program/JTAG interfaces, using PROG PWR SEL/JTAG PWR SEL pin headers.
4. Connect the ESP-Prog and ESP user board with the flat cables provided by Espressif.
5. Start programming (downloading) or JTAG debugging, using the official software tools or scripts provided by Espressif.

Software Setup

Please proceed to Get Started with ESP-IDF, where Section Installation Step by Step will quickly help you set up the development environment.

Contents and Packaging

Retail Orders

Each ESP-Prog board comes in an individual package.

The contents are as follows:

- Development board

  ESP-Prog
- Cables

  Two flexible flat cables:

  - One cable to connect to 2*5-PIN 2.54 mm male shrouded box header.

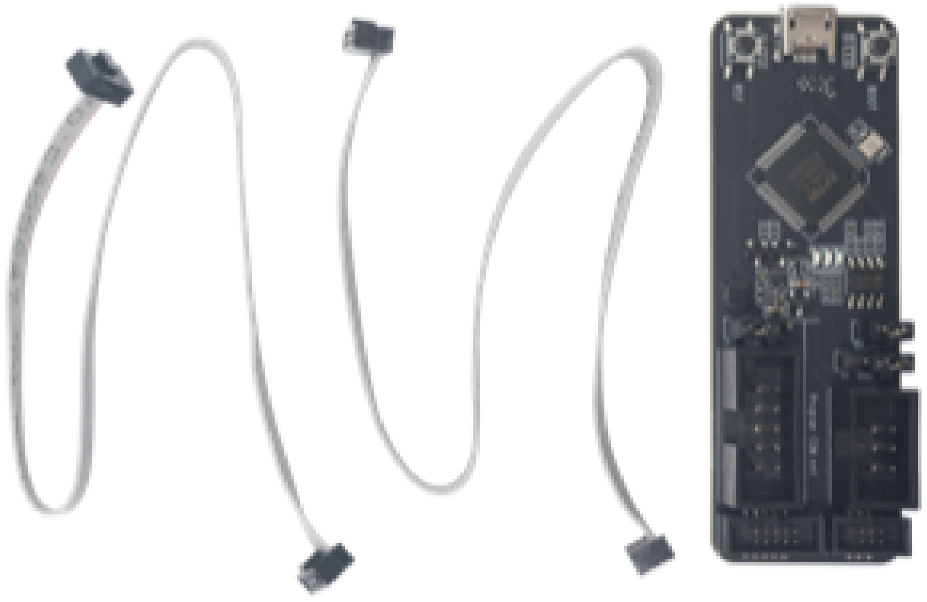


Fig. 50: ESP-Prog Package Contents

- One cable to connect to 2*3-PIN 1.27 mm male shrouded box header.

If you order a few samples, each board comes in an individual package in either antistatic bag or any packaging depending on your retailer.

For retail orders, please go to https://www.espressif.com/en/company/contact/buy-a-sample.

Wholesale Orders

If you order in bulk, the boards come in large cardboard boxes.

For wholesale orders, please go to https://www.espressif.com/en/contact-us/sales-questions.

Hardware Reference

Block Diagram

The block diagram below shows the components of ESP-Prog and their interconnections.

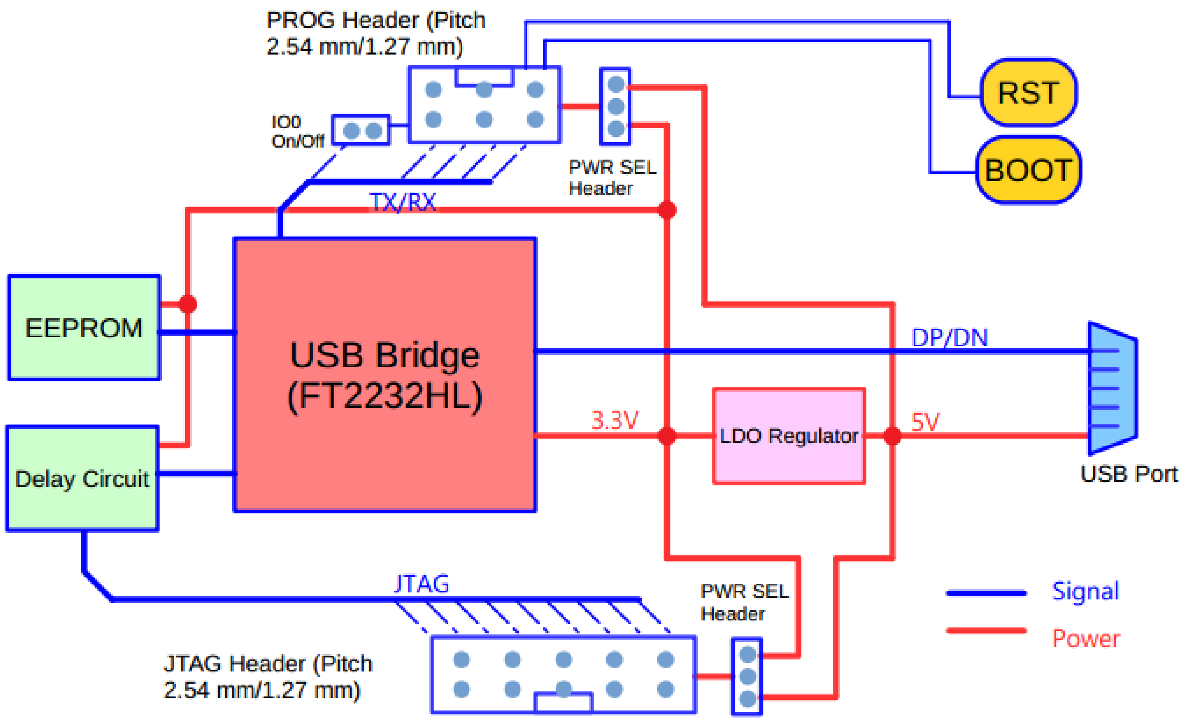


Fig. 51: ESP-Prog Block Diagram (click to enlarge)

Power Supply Options

There are three mutually exclusive ways to provide power to the board:

- ESP-Prog USB Port, default power supply (recommended)
- 5 V and G (GND) pins
- 3.3 V and G (GND) pins

Header Block

The two tables below provide the **Name** and **Function** of the pins on both sides of the board (Program Interface and JTAG Interface). The pin names are shown in the front view of the ESP-Prog board. The numbering is the same as in the ESP-Prog Schematic (PDF).

Program Interface

No.	Name	Function
1	ESP_EN	Enable signal
2	VDD	Power supply
3	ESP_TXD	TX pin
4	GND	Ground
5	ESP_RXD	RX pin
6	ESP_IO0	Strapping pin

JTAG Interface

No.	Name	Function
1	VDD	Power supply
2	ESP_TMS	JTAG TMS pin, mode selection
3	GND	Ground
4	ESP_TCK	JTAG TCK pin, clock input
5	GND	Ground
6	ESP_TDO	JTAG TDO pin
7	GND	Ground
8	ESP_TDI	JTAG TDI pin
9	GND	Ground
10	NC	None

Related Documents

- ESP-Prog Schematic (PDF)
- ESP-Prog PCB Layout (PDF)
- ESP-Prog Dimensions (PDF)

For further design documentation for the board, please contact us at sales@espressif.com.

Reference Documentation

[??]

Introduction to Functions

The Working Mode of USB Bridge

ESP-Prog uses FT2232HL, which is provided by FTDI, as its USB Bridge Controller chip. The board can be configured to convert the USB 2.0 interface to serial and parallel interfaces that support a wide range of industry standards. ESP-Prog uses FT2232HL's default dual-asynchronous serial interface mode available after installing the FT2232HL chip driver on their PCs.

Note: The PC is able to identify the ESP-Prog's two ports by their port numbers. The bigger port number represents the Program interface, while the other one represents the JTAG interface.

Communication Interface

ESP-Prog can connect to ESP32 user boards using both the Program interface and the JTAG interface.

- **Program Interface**

  The Program interface has six pins, including the UART interface (ESP_TXD, ESP_RXD), boot mode selection pin (ESP_IO0) and reset pin (ESP_EN). The design for the Program interface on the user board should follow the reference provided in the figure below.

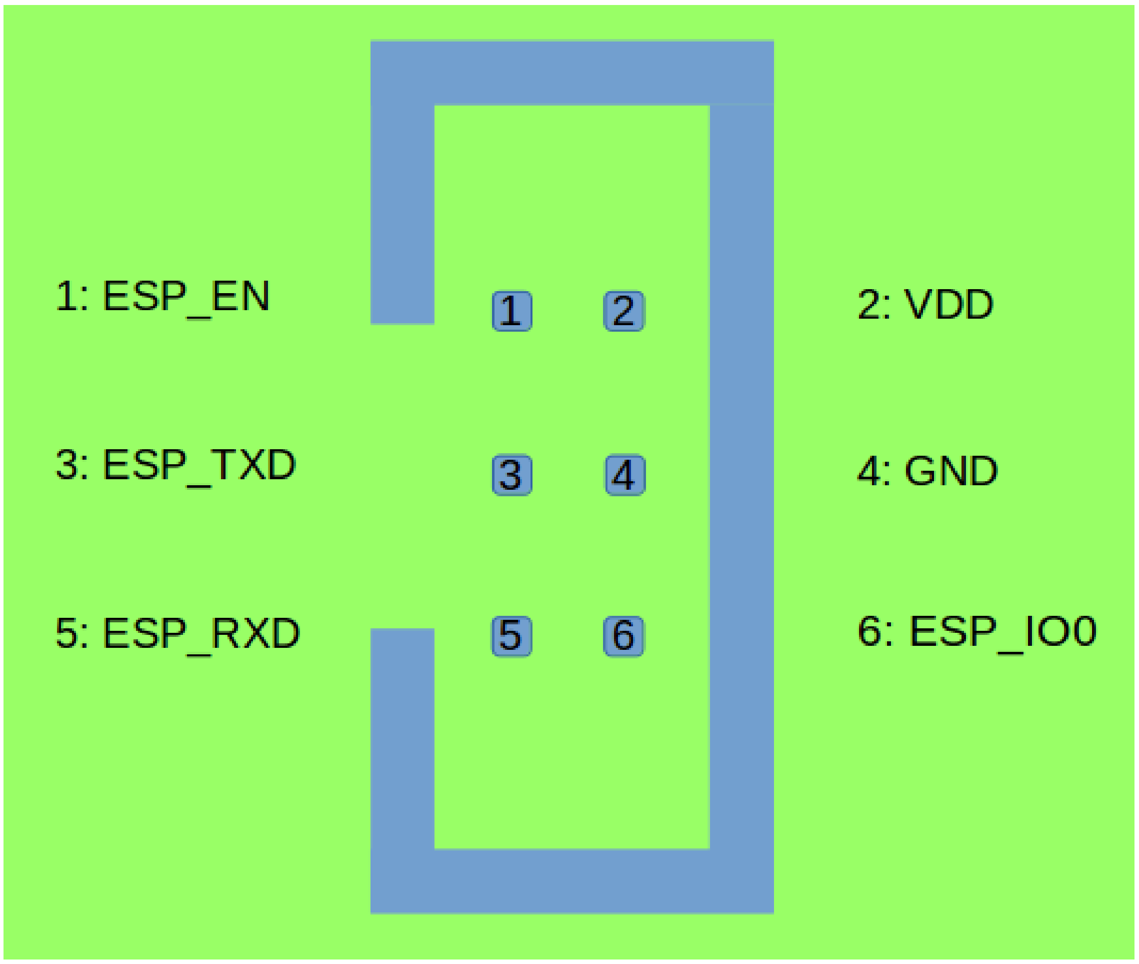


Fig. 52: Program Interface (click to enlarge)

- **JTAG Interface**

  The design for the JTAG interface on the user board should follow the reference provided in the figure below.

- **Fool-proof Design**

  The ESP-Prog board uses header connectors (DC3-6P/DC3-10P) which support reverse-current circuitry protection. In such cases, it is recommended that users also use header connectors on their user boards, such as

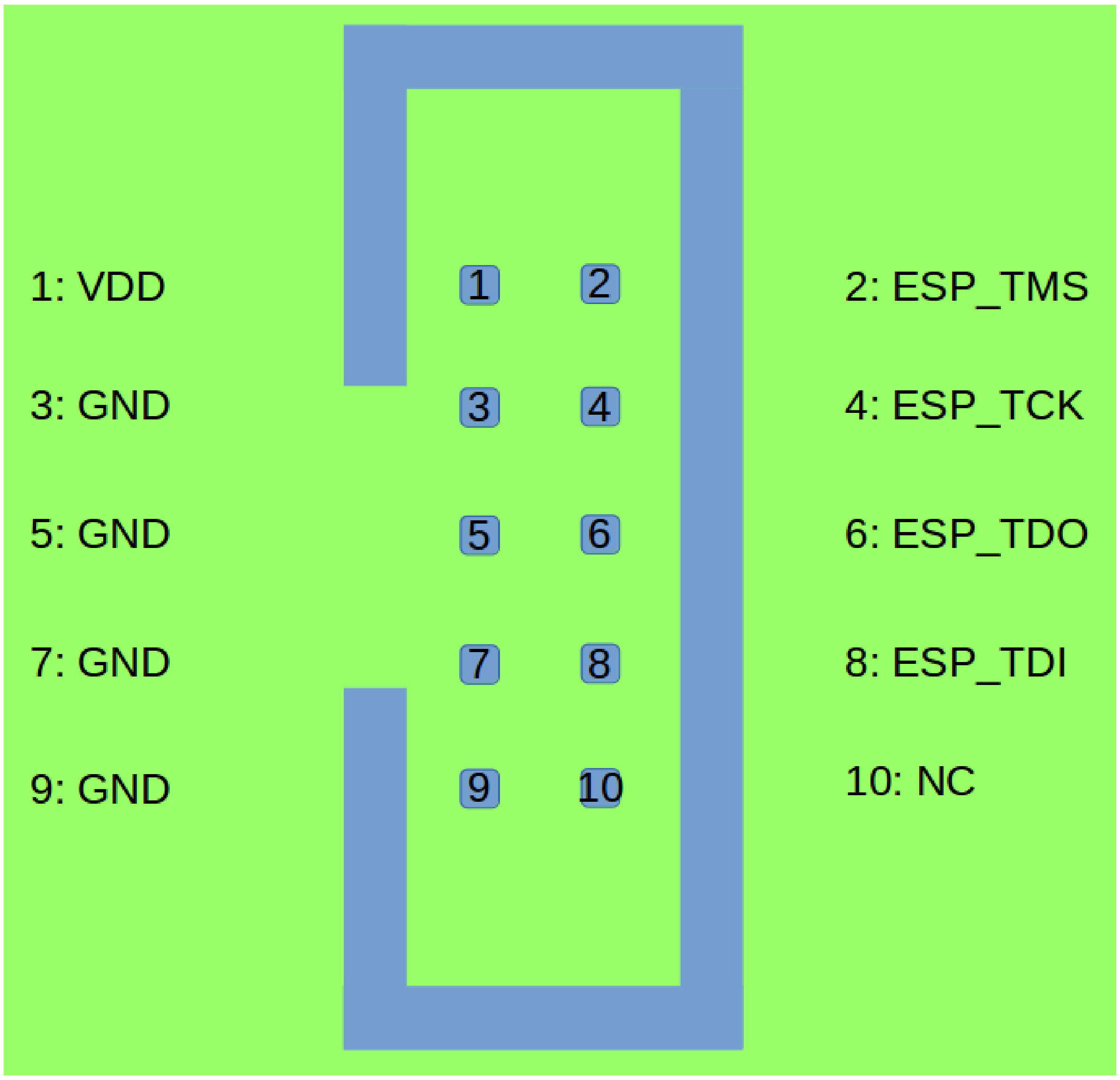


Fig. 53: JTAG Interface (click to enlarge)

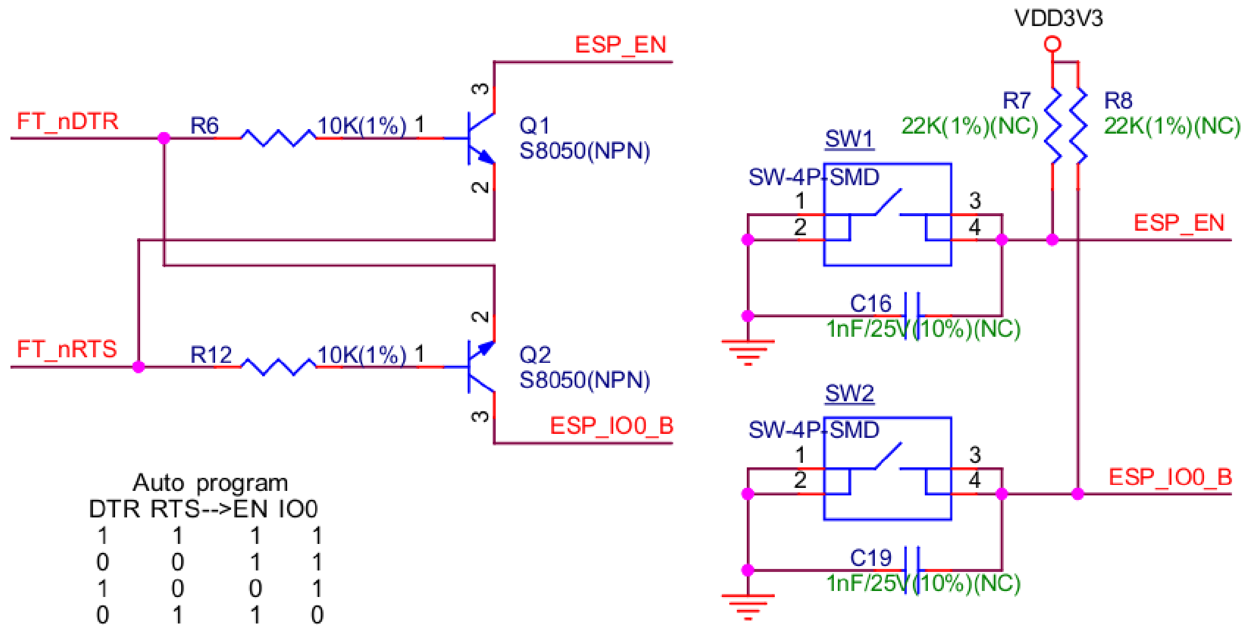
`FTSH-105-01-S-DV-*` or `DC3-*P`.

Note: Keying of the plugs and sockets to insert the plug is in one specific orientation, which means each socket of ESP-Prog corresponds to the plugs on the cable and using a mismatched cable might lead to a wrong order of connection. Please use the cables provided by Espressif.

Automatic Downloading Function

ESP-Prog supports automatic downloading. After connecting the Program interface of ESP-Prog to the user board, the downloading program can download data or run programs automatically by controlling the states of the start-mode selection pin (ESP_IO0) and reset pin (ESP_EN), which spares the users from manually restarting the device and selecting the downloading modes. The two buttons on the ESP-Prog board enable users to reset and control the boot mode of the device manually.

The schematics of the automatic downloading circuit is displayed below.



Auto program

DTR	RTS-->	EN	IO0
1	1	1	1
0	0	1	1
1	0	0	1
0	1	1	0

Fig. 54: Automatic Downloading Circuit (click to enlarge)

Delay Circuit

The delay circuit of ESP-Prog includes the bus buffer, inverter, MOSFET, first-order RC circuit, and other components. This delay circuit ensures that the ESP32 chip can power up or reset itself before connecting with the JTAG signal, thus protecting the chip from the influence of JTAG on power-up or reset.

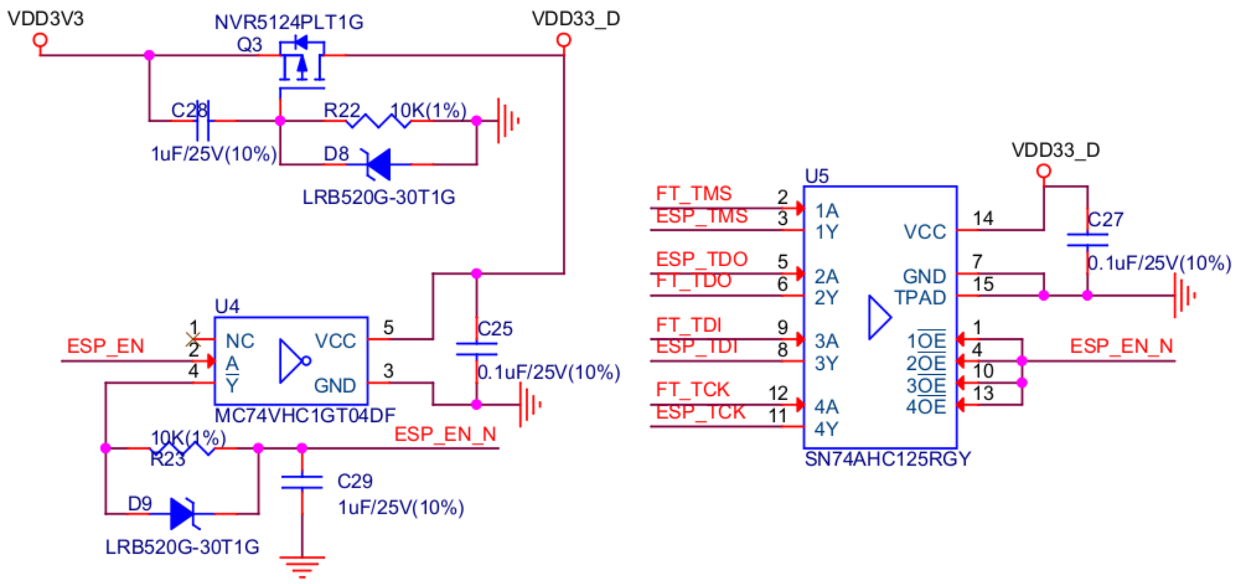


Fig. 55: ESP-Prog Delay Circuit (click to enlarge)

LED Status Indication

- The red LED lights up when the system is connected to the 3.3 V power.
- The green LED lights up when ESP-Prog is downloading data to ESP32.
- The blue LED lights up when ESP-Prog is receiving data from ESP32.

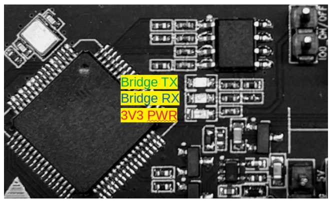


Fig. 56: LED Status (click to enlarge)

Pin Headers

Users can select the power supply for the Program and JTAG interfaces via the Pin Header to Select Power Supply, and select the boot modes of ESP8266 and ESP32 via the IO0 On/Off Pin.

- **Pin Header to Select Power Supply**

  The pin header in the middle is the power input pin for each interface. When this pin is connected to 5 V, the power output of the interface is 5 V. When this pin is connected to 3.3 V, the power output of the interface is 3.3 V.

- **IO0 On/Off Pin**

  Pin IO0 can be set to select ESP8266's and ESP32's boot modes. This pin can be used as a common GPIO, after the chip is powered on. By removing a jumper from the pin header, users can disconnect Pin IO0 manually to protect the operation of the user board from the influence of ESP-Prog's automatic downloading circuit.

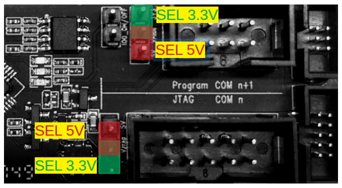


Fig. 57: Pin Headers (click to enlarge)

For further design documentation for the board, please contact us at sales@espressif.com.

1.1.9 ESP32-S2-HMI-DevKit-1

[??]

ESP32-S2-HMI-DevKit-1 has been specifically designed for human-machine interfaces in smart-home automation controllers, smart speakers with display, smart alarm clocks, etc.

Application examples for this board can be found at Examples .

ESP32-S2-HMI-DevKit-1 V1.0

[??]

This user guide will help you get started with ESP32-S2-HMI-DevKit-1 and will also provide more in-depth information.

ESP32-S2-HMI-DevKit-1 has been specifically designed for human-machine interfaces in smart-home automation controllers, smart speakers with display, smart alarm clocks, etc. This development kit supports rapid secondary development, since developers can take advantage of the kit's various onboard resources and expansion interfaces, in order to develop various functions.

The main features of the board are listed below:

- **Module Embedded:** ESP32-S2-WROVER module with 4 MB flash and 2 MB PSRAM
- **Display:** 4.3-inch TFT-LCD which uses 16-bit 8080 parallel port with 480×800 resolution and 256-level hardware DC backlight adjustment circuit, connected to an I2C capacitive touch panel
- **Audio:** Audio amplifier, built-in microphone, speaker connector
- **Storage:** microSD card connector
- **Sensors:** 3-axis accelerometer, 3-axis gyroscope, ambient light sensor, temperature and humidity sensors
- **Expansion:** SPI header, TWAI interface (compatible with CAN 2.0), I2C ADC, UART/Prog header
- **LEDs:** Programmable RGB LED and IR LED
- **Buttons:** Wake Up and Reset buttons
- **USB:** 1 x USB-C OTG (DFU/CDC) port, 1 x USB-C debug port
- **Power Supply:** 5V and 3.3V power headers
- **Optional Rechargeable Battery:** 1,950 mAh single-core lithium battery with a charge IC

The document consists of the following major sections:

- *Getting started*: Overview of the board and hardware/software setup instructions to get started.
- *Hardware Reference*: More detailed information about the board's hardware.

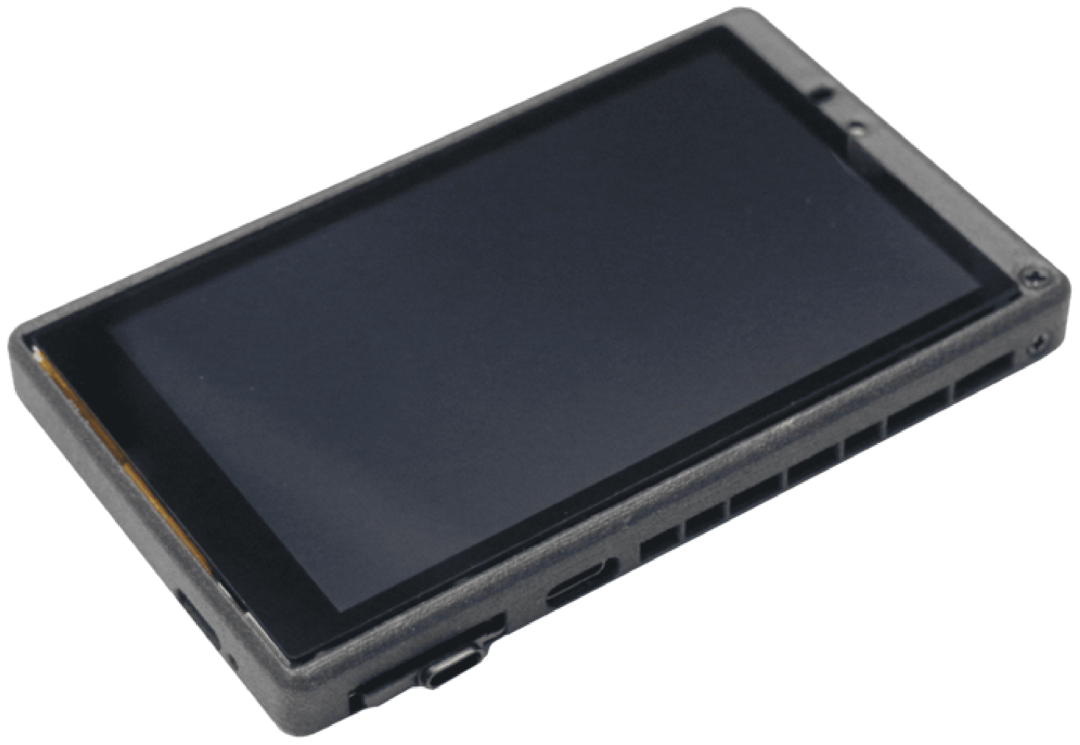


Fig. 58: ESP32-S2-HMI-DevKit-1 with ESP32-S2-WROVER module

- *Related Documents*: Links to related documentation.

Getting Started

This section provides a brief introduction of ESP32-S2-HMI-DevKit-1, instructions on how to do the initial hardware setup and how to flash firmware onto it.

Description of Components

ESP32-S2-HMI-DevKit-1 is an HMI development board designed based on the ESP32-S2. The following figure describes its key on-board resources:

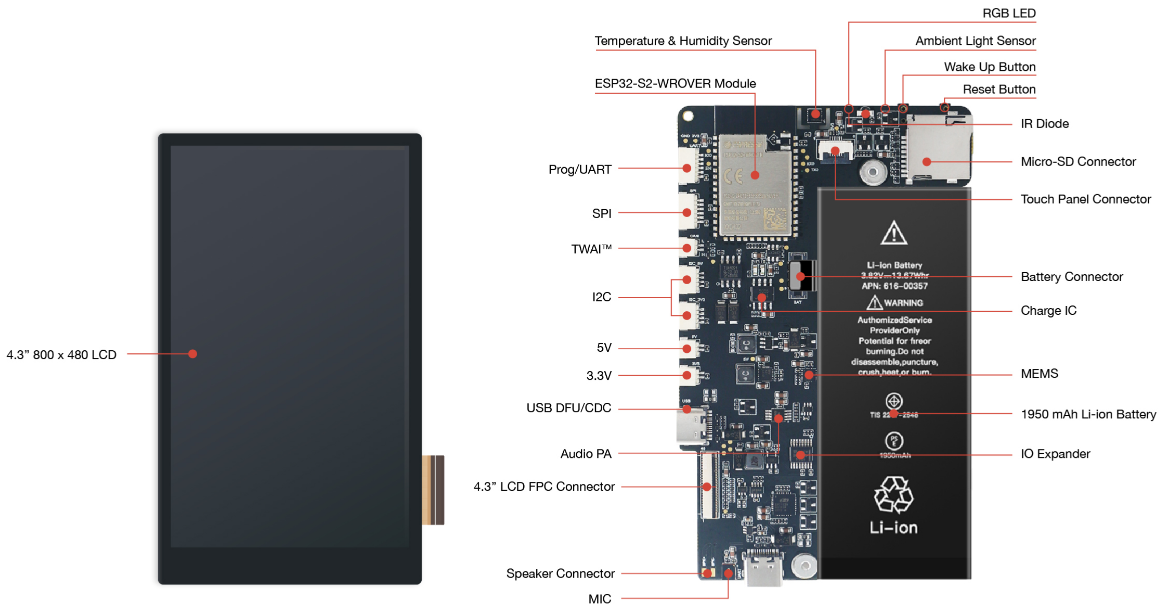


Fig. 59: ESP32-S2-HMI-DevKit-1 (click to enlarge)

The key components of the board are described in a clockwise direction.

Key Component	Description
ESP32-S2-WROVER Module	ESP32-S2-WROVER is a powerful, generic Wi-Fi MCU module that integrates ESP32-S2. It has a PCB antenna, a 4 MB external SPI flash and an additional 2 MB PSRAM.
Temperature & Humidity Sensor	Temperature & humidity sensors for detecting ambient temperature and humidity. Read via the I2C bus.
RGB LED	Addressable RGB LED (WS2812), driven by GPIO21. Can switch between the LED and IR LED via IO expander.
Ambient Light Sensor	Ambient Light Sensor used to detect ambient light intensity. Read via the I2C bus.
Wake Up Button	Download button. Press and hold the **Boot** button while pressing the **Reset** button to initiate the "firmware download" mode to download the firmware via the serial port. This button can also be configured to wake up the device from deep sleep mode.
Reset Button	Press this button to restart the system.
IR LED	Infrared emitting diode, driven by GPIO21. Can switch between RGB LED and IR LED via IO expander.
Charge IC	Charge battery.
MEMS	3-axis accelerometer and 3-axis gyroscope.
1950 mAh Li-ion Battery	An optional 1,950 mAh rechargeable lithium battery.
IO Expander	Expand GPIO through I2C bus.
MIC	On-board simulated microphone.
Audio PA	Audio amplifier.
4.3" 800 × 480 LCD	A 4.3-inch TFT-LCD which uses 16-bit 8080 parallel port with 480×800 resolution, and 256-level hardware DC backlight adjustment circuit. It is connected with an I2C capacitive touch panel overlay.

Start Application Development

Before powering up your board, please make sure that it is in good condition with no obvious signs of damage.

Required Hardware

- 1 x PC loaded with Windows, macOS or Linux (Linux operating system is recommended)
- 1 x ESP32-S2-HMI-DevKit-1
- 1 x USB-C cable (it is recommended to prepare two USB-C cables if you want to evaluate MCU's USB functions)
- 1 x Speaker (8 Ohm, 2 W)
- 1 x microSD card (some examples may have large storage needs)

Note: Be sure to use an appropriate USB cable. Some cables are for charging only and do not provide the needed data lines nor work for programming the boards.

Hardware Setup

To facilitate your quick evaluation of all examples, please follow these steps to set up the board:

1. Insert microSD card into the connector. Please make sure all the important data is backed up, as the microSD card may be formatted if its partition format is not FAT.
2. If you need to evaluate the audio playback function, please connect the speaker pad near the USB port on the bottom of the board to the supplied speaker, or to another speaker with a similar size (8 Ohm, 2 W).

Software Setup

First, please make sure you have configured the ESP-IDF development environment correctly. To ensure this, please enter `idf.py --version` in your terminal and if the output is similar to `ESP-IDF v4.2-dev-2084-g98d5b5dfd`, it means you have installed ESP-IDF correctly. For detailed information about installation and configuration, please refer to ESP-IDF Get Started.

After configuration completed, you can switch back to the `esp-dev-kits/esp32-s2-hmi-devkit-1` directory. All code examples are placed under the examples directory, you can build projects by running `idf.py build`.

Project Options

Various examples are provided for ESP32-S2-HMI-DevKit-1 as shown below:

- Printing "Hello world!" on screen: esp32-s2-hmi-devkit-1/examples/get-started/hello_world
- Blinking WS2812 LED and showing the color on screen: esp32-s2-hmi-devkit-1/examples/get-started/led_blink
- Starting a UI to configure Wi-Fi credential: esp32-s2-hmi-devkit-1/examples/get-started/provision
- Acquiring audio with ADC from the output of analog MIC: esp32-s2-hmi-devkit-1/examples/audio/audio_record
- Playing music: esp32-s2-hmi-devkit-1/examples/audio/music_player
- Shutting down selected board area into a deep sleep: esp32-s2-hmi-devkit-1/examples/examples/power
- Using Freetype to render fonts: esp32-s2-hmi-devkit-1/examples/freetype
- Using on-board sensors: esp32-s2-hmi-devkit-1/examples/sensors
- Using smart panel: esp32-s2-hmi-devkit-1/examples/smart-panel
- Viewing files on SD card: esp32-s2-hmi-devkit-1/examples/storage/sdcard_fatfs
- USB flash disk: esp32-s2-hmi-devkit-1/examples/storage/usb_msc

You can configure project options by entering `idf.py menuconfig` in each example directory.

Please make sure to correctly configure the following options in menuconfig:

- `(Top) > HMI Board Config > HMI board`: Select board version. By default, please select `ESP32-S2-HMI-DevKit-V2`;
- `(Top) > HMI Board Config > Audio HAL`: Select audio output interface, whether to use PWM or DAC;
- `(Top) > HMI Board Config > LCD Drivers`: Select display IC type for LCD. By default, ESP32-S2-HMI-DevKit-1 uses RM68120 as its display IC;
- In `(Top) > Component config > ESP32S2-specific`, please go to the `Support for external, SPI-connected RAM` option:

  - Go to `SPI RAM config > Set RAM clock speed`, and set the PSRAM clock as `80 MHz clock speed`;

- `(Top) -> Component config -> FreeRTOS`: set `Tick rate (Hz)` as 1000.

In each example folder, we have provided a default configuration file named `sdkconfig.defaults`, with above options configured correctly.

ESP-IDF Version Dependencies

The `esp32-s2-hmi-devkit-1/examples/storage/usb_msc` example needs to be built in IDF v4.3, while other examples can be built in IDF v4.2 and later versions.

Notice: Due to strict regulation on battery export, for deliveries outside of China mainland, we are shipping ESP32-S2-HMI-DevKit-1 without the battery. As a substitute, you can use the iPhone 5 replacement battery (the connector type is non-standard). The battery capacity is not critical.

Contents and Packaging

Retail Orders

If you order one or several samples of the kit, each ESP32-S2-HMI-DevKit-1 development kit comes in an individual package.

Fig. 60: ESP32-S2-HMI-DevKit-1 package

The contents are as follows:

- Development board - ESP32-S2-HMI-Devit-1
- Cables - SH1.25 to 2.54mm cables x 7

For retail orders, please go to https://www.espressif.com/en/company/contact/buy-a-sample.

Wholesale Orders

If you order in bulk, the boards come in large cardboard boxes.

For wholesale orders, please go to https://www.espressif.com/en/contact-us/sales-questions.

Hardware Reference

Block Diagram

The block diagram below shows the components of ESP32-S2-HMI-DevKit-1 and their interconnections.

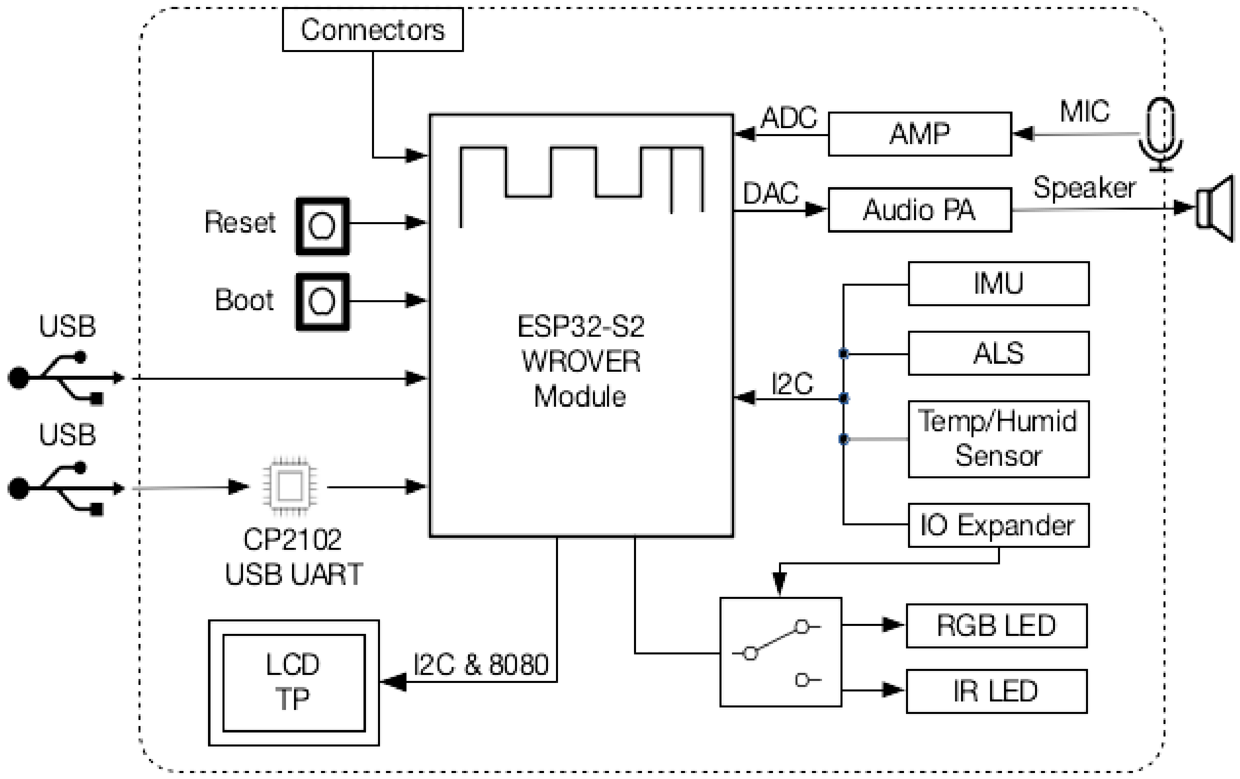


Fig. 61: ESP32-S2-HMI-DevKit-1 block diagram

Power Supply Options

The power of the ESP32-S2-HMI-DevKit-1 development board is divided into a 5 V power domain and a 3.3 V power domain, so as to reduce power consumption, improve power efficiency and support battery charging, part of which can be controlled by software whereas the other part is fixed in the hardware design.

To reduce current consumption, the preloaded firmware will power off all controlled power domains and put all ICs in low-power mode.

For more information, please refer to *Power*.

Connectors

It provides multiple extended interfaces for customized development. The connectors of the board are described in a clockwise direction. Please refer to *ESP32-S2-HMI-DevKit-1 key on-board resources*.

Connectors	Description
Speaker Connector	To connect a speaker.
4.3" LCD FPC Connector	(Reserved) Connect to the supported 4.3" LCD extension board using the FPC cable.
USB DFU/CDC	1 x USB-C OTG (DFU/CDC) port, 1 x USB-C debug port.
3.3 V Connector	3.3 V power header.
5 V Connector	5 V power header.
I2C Connector	I2C connector with 5 V/3.3 V power supply options.
TWAI interface (compatible with CAN 2.0)	Two-wire automotive interface.
SPI	Can connect devices on the SPI bus if the SD card is not in use.
Prog/UART	This interface is used to observe log output and firmware flash.
microSD Connector	Insert microSD card to expand the storage space of the device.
Battery Connector	To connect a battery.

Below is the back view of the board for your reference.

Related Documents

- ESP32-S2 Datasheet (PDF)
- ESP32-S2-WROVER Datasheet (PDF)
- ESP32-S2-HMI-DevKit-1 Schematic (PDF)
- ESP32-S2-HMI-DevKit-1 PCB Layout (PDF)
- ESP32-S2-HMI-DevKit-1 Dimensions (PDF)
- ESP32-S2-HMI-DevKit-1 Dimensions Source File (DXF) - You can view it with Autodesk Viewer online

For further design documentation for the board, please contact us at sales@espressif.com.

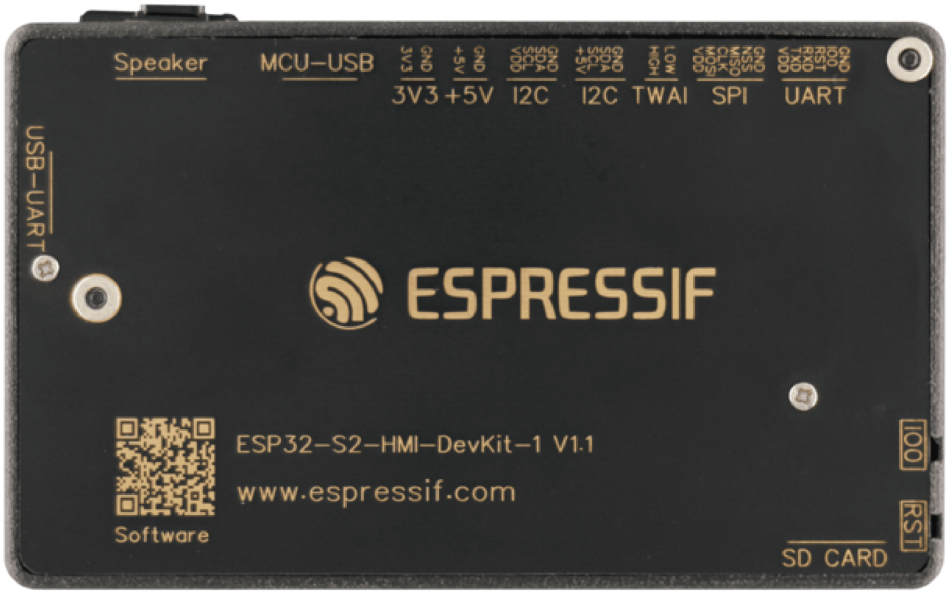


Fig. 62: ESP32-S2-HMI-DevKit-1 - back view

Reference Documentation

[⍰⍰]

Audio

[⍰⍰]

The ESP32-S2-HMI-DevKit-1 development board supports audio playback and recording. You can find such examples under the `esp32-s2-hmi-devkit-1/examples/audio/` directory.

Audio Playback

The ESP32-S2-HMI-DevKit-1 development board can output audio via DAC or PWM. It is recommended to use PWM for audio output since it has lower noise and higher resolution (DAC has 8-bit resolution, while PWM can reach up to 12-bit resolution at 19.2 kHz of sampling rate).

The output signal generated through the IO port goes to the digital potentiometer TPL0401 first for lossless volume adjustment, and then passes the 100 nF isolation capacitor C33 and the 200 kOhm resistor R52. This RC circuit controls the cut-off frequency at around 8 Hz. On top of that, this signal will be sent to the 3 W class-D audio power amplifier NS4150 to set the gain to 1.5 times, thus amplifying the maximum output signal from 3.3 V to 4.95 V (slightly lower than the PA supply, 5 V) so as to maximize the output volume while minimizing saturation distortion.

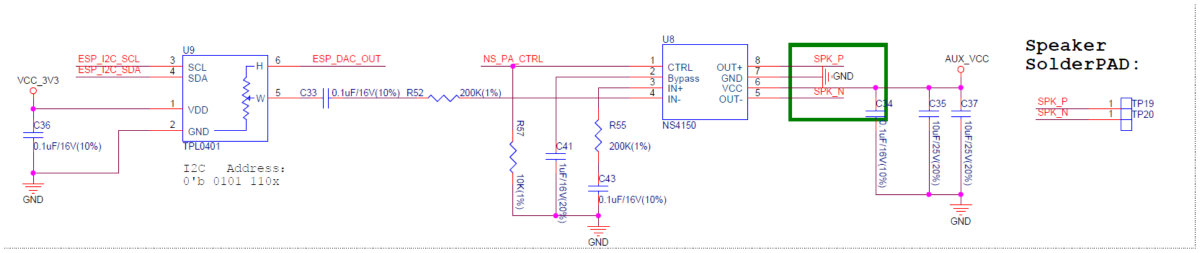


Fig. 63: ESP32-S2-HMI-DevKit-1 audio playback schematic (click to enlarge)

The audio PA is powered by the 5 V power domain. Before using the audio playback function, please make sure this power domain is powered on (refer to *5 V Power Domain* section).

Audio Recording

The ESP32-S2-HMI-DevKit-1 development board can record audio data from the analog microphone via an internal ADC.

The board is equipped with an analog microphone with a sensitivity of -38 dB. And it will send the output signal to the operational amplifier TLV6741 with a fixed gain to amplify the signal.

The microphone and operational amplifier mentioned above are powered by a controlled 3.3 V power domain. Before using the audio recording function, please make sure this power domain is powered on (refer to *3.3 V Power Domain* section).

Please use the Timer Group interrupt to record audio data. Do not use code such as the following format in tasks for audio recording:

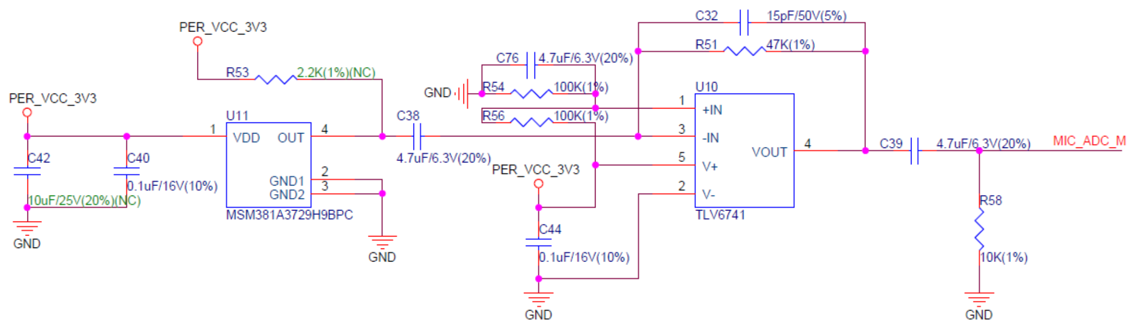


Fig. 64: ESP32-S2-HMI-DevKit-1 microphone schematic (Click to enlarge)

```
size_t index = 0;
uint16_t audio_data[configMAX_ACQUIRE_COUNT];
do {
    audio_data[index] = adc1_get_raw(CONFIG_AUDIO_CHANNEL);
    ets_delay_us(1000000 / CONFIG_AUDIO_FREQ);
} while (++index < CONFIG_MAX_ACQUIRE_COUNT)
```

The above format will cause the CPU to be occupied, thus triggering the task watchdog (if it is not disabled), and make other tasks with lower priorities (e.g., IDLE Task) not able to operate normally.

When recording data via ADC with the interrupt function, you need to re-write the ADC recording function to `IRAM_ATTR` so as to reduce response time, and place the variables to DRAM. Also, please do not use any semaphore in this function. For more information about implementation examples, please refer to `audio/audio_record` under the `examples` directory.

ADC Accuracy

The ADC of ESP32-S2 has a high level of repeatability despite the fact that the lack of reference voltage and using Buck power supply may result a high overall noise.

The ADC is configured with 13-bit resolution and 11 dB attenuation, corresponding to a full-scale voltage of 2.6 V. After polling the 2.5 V voltage of the AD584T reference voltage via ADC1_CH8, we convert the 4096-time uncalibrated raw values into voltage values and get the following data:

As shown in the above figure, most of the uncalibrated data error is within the range of ±0.005 V with a standard deviation (σ) of 3.339 LSB (0.00106 V). These errors are mainly from the absolute accuracy, i.e., the bias. Therefore, the distortion and noise of sounds sampled via ADC can be kept at a relatively low level.

The AD584T has a peak output noise of 50 uV at 10 V within the range of 0.1 ~ 10 Hz, and a peak output noise divided by the internal high-precise laser-adjusted resistor at 2.5 V. And an up to 30 mA push-pull capability is provided by the transistor measured at 10 V. Its output noise at 2.5 V is lower than the resolving power of a 16-bit ADC, which therefore can be used as a testing reference.

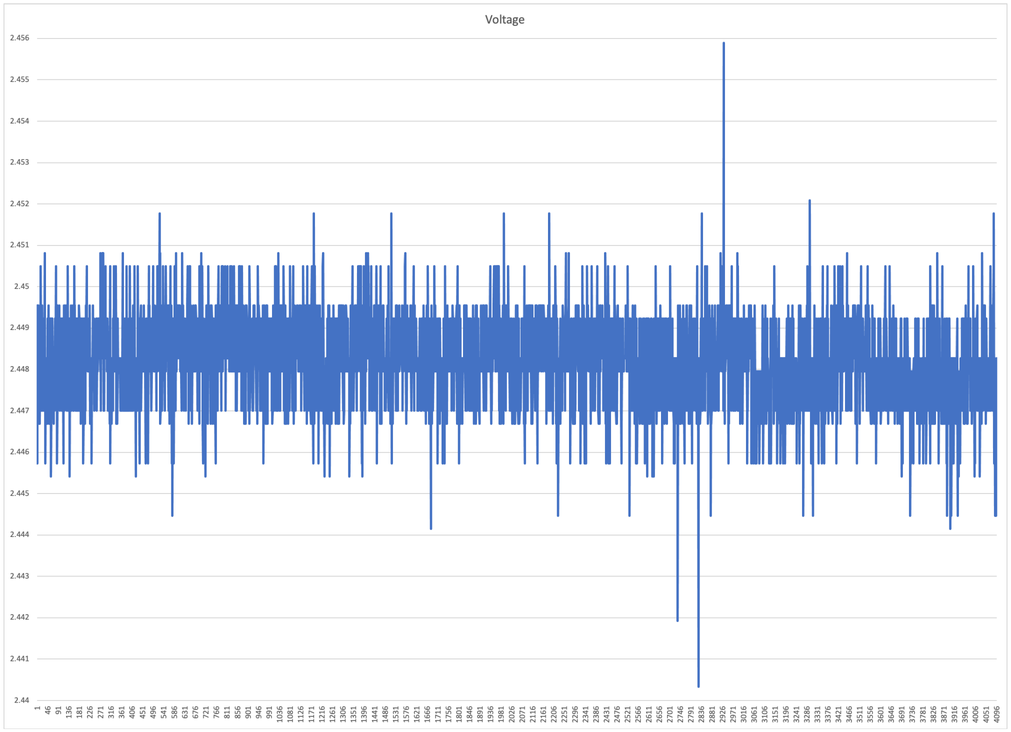


Fig. 65: ESP32-S2-HMI-DevKit-1 ADC (click to enlarge)

Display and Touch Panel

[??]

The ESP32-S2-HMI-DevKit-1 has a 4.3-inch color TFT-LCD with 800×480 resolution, and a touch panel with an I2C interface. The interface type and data bit width of this display is controlled by programmable pins. This development board has been configured as 16-bit 8080 parallel communication via resistors.

This LCD uses RM68120 as its display IC and FT5436 as its touch IC.

Communication

The display IC of the LCD used in ESP32-S2-HMI-DevKit-1 has been configured for 16-bit 8080 parallel communication, with a total of 18 GPIOs used, i.e., 16 data lines (LCD_D0…LCD_D15), a bit clock signal (LCD_WR) and a data/command distinguish signal (LCD_DC/LCD_RS).

The touch IC uses the I2C interface to communicate with MCU and can share this interface with other I2C's ICs, and thus does not need to use additional GPIOs. The touch IC supports interrupt signal output. The interrupt signal is first sent to the P2 pin of the IO expander, and the falling edge from this pin will generate a low level in the interrupt output pin of the IO expander, so that the MCU receives this interrupt signal. In this case, you can read the input level register of the IO expander to check whether this interrupt is from the touch IC. Once a read operation completed, the interrupt flag will be cleared.

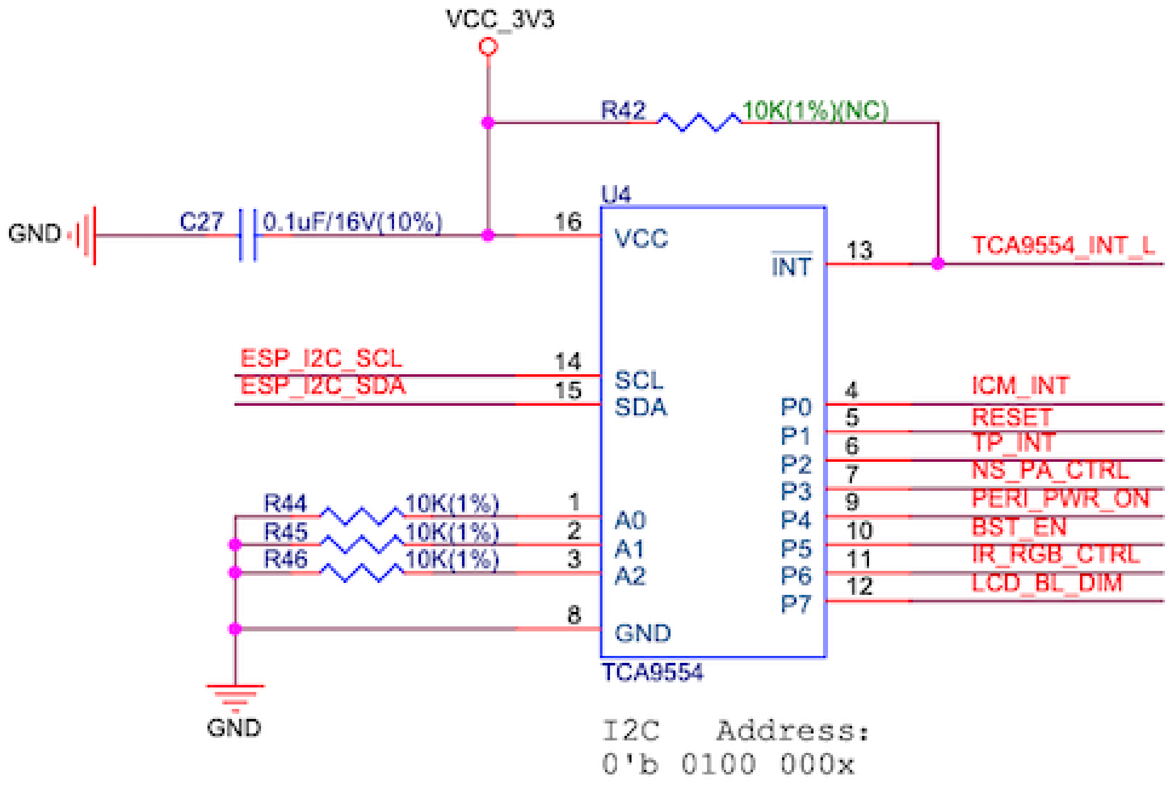


Fig. 66: ESP32-S2-HMI-DevKit-1 IO expander schematic

Backlight

As the LEDs are connected in series, they need to be drived by constant current via the Booster circuit. The rated current is 18 mA and the voltage is approximately 24 V (may not be accurate, only for reference). To prevent the feedback voltage of the Booster circuit always being 0 when the display is not connected, and thus causing high voltage loaded to both ends of the backlight filter capacitor C21, please make sure this capacitor can withstand 38 V.

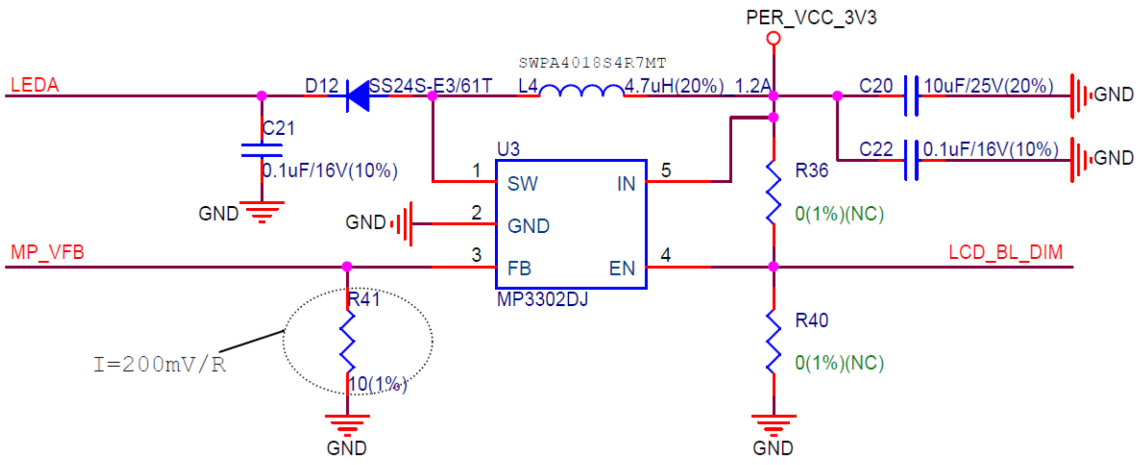


Fig. 67: ESP32-S2-HMI-DevKit-1 backlight PWM dimming schematic (click to enlarge)

Since PWM dimming may cause display flicker and some Booster IC do not support high-frequency PWM signal control, this development board provides an option to use DC dimming circuit to reach high performance, as shown in the figure below:

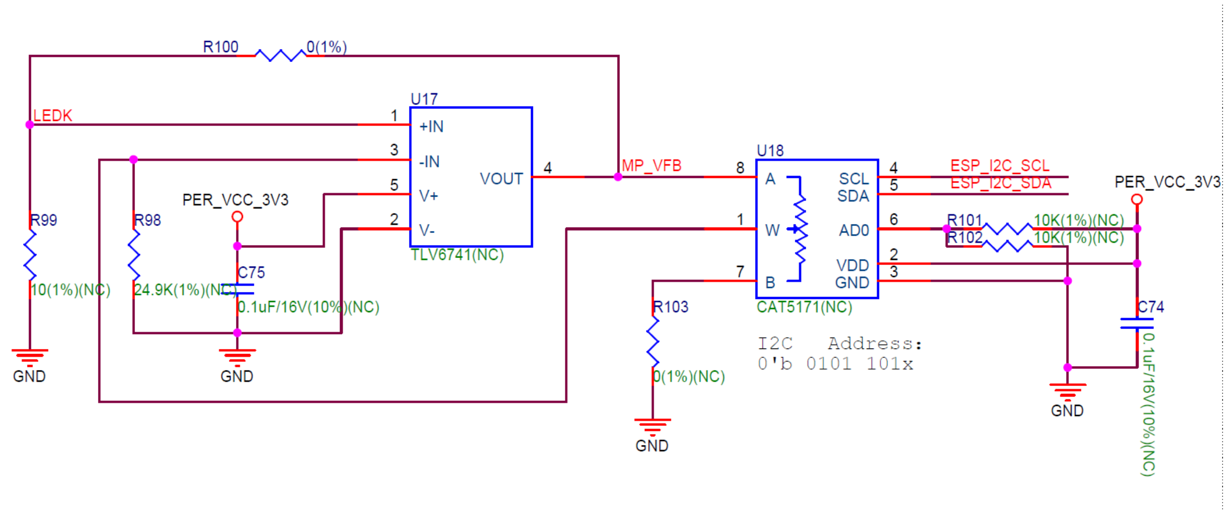


Fig. 68: ESP32-S2-HMI-DevKit-1 DC dimming circuit schematic (click to enlarge)

This DC dimming circuit inputs the VFB voltage to the operational amplifier TLV6741, whose gain resistor is a digital potentiometer that can be modified via the I2C bus. This digital potentiometer is CAT5171, with 256 levels of resolution and a maximum resistance value of 50 kOhm.

The EN pin of the Booster IC is controlled by the P7 pin of the IO expander in high level. If you want to keep the contents while turning off the display, please set this pin to low level so as to disable backlight.

Touch

The capacitive touch panel on the development board uses a touch IC with a resolution of 800×480 and supports up to 5-point touch and hardware gesture recognition.

The hardware of this display IC does not support screen rotation itself. Therefore, for scenarios where the panel is needed to be rotated, you may need to convert the data read by the touch IC through calculating its relative value to the resolution or via certain software. Multi-touch is supported by hardware and we provide some APIs for reading the multiple touch points. However, since the LVGL used in the GUI library does not support multi-touch processing for now, you may need to handle the data of these touch points in the application layer yourself.

Power

[⍰⍰]

The power of the ESP32-S2-HMI-DevKit-1 development board is divided into a 5 V power domain and a 3.3 V power domain, so as to reduce power consumption, improve power efficiency and support battery charging. Part of the power domain can be controlled by software whereas the other part is configured as permanently enabled in hardware design.

To reduce current consumption, the preloaded firmware will power off all controlled power domains and put all ICs to low-power mode.

3.3 V Power Domain

Most of the ICs and modules are powered by the 3.3 V power domain, which can be divided into an uncontrolled 3.3 V power domain and a controlled 3.3 V power domain.

The uncontrolled 3.3 V power domain cannot be powered off via software, and provides power for the Buck circuit. When there is a power supply from USB, this power domain will obtain power from the 5 V input through the USB cable; when USB is disconnected, it will obtain 3.6 ~ 4.2 V power from the lithium battery. This power domain mainly provides power for the ESP32-S2-WROVER module and other devices which can enter low-power mode via software control.

The controlled 3.3 V power domain comes from the uncontrolled 3.3 V power domain and is turned on/off via a PMOS control switch, which is connected to the P4 pin of the IO expander. This power domain mainly provides power for ICs with higher static power consumption and cannot enter low-power mode.

5 V Power Domain

The 5 V power domain of the development board provides power for the audio amplifier and the TWAI® transceiver. It obtains power from the following resources:

- The USB port
- The power input from external 5 V power port
- The power passing through the Booster circuit from the lithium battery

The power obtained from USB and the external 5 V power input supplies power for all devices (except CP2102N) that require 5 V power and cannot be disconnected by software. When obtaining power from the lithium battery, the EN pin level of the Booster IC can be controlled via the P5 pin of the IO expander to enable 5 V power in high level.

The power input through the USB port on the bottom of the board is split into two lines: one provides power for CP2102N while the other becomes USB_5V after passing through a diode. The CP2102N will only be powered up when this USB port is connected, since it only needs to be in operation when the PC is connected. Any 5 V power input will cause the Booster IC to be powered off and charge the on-board lithium battery via the charging IC.

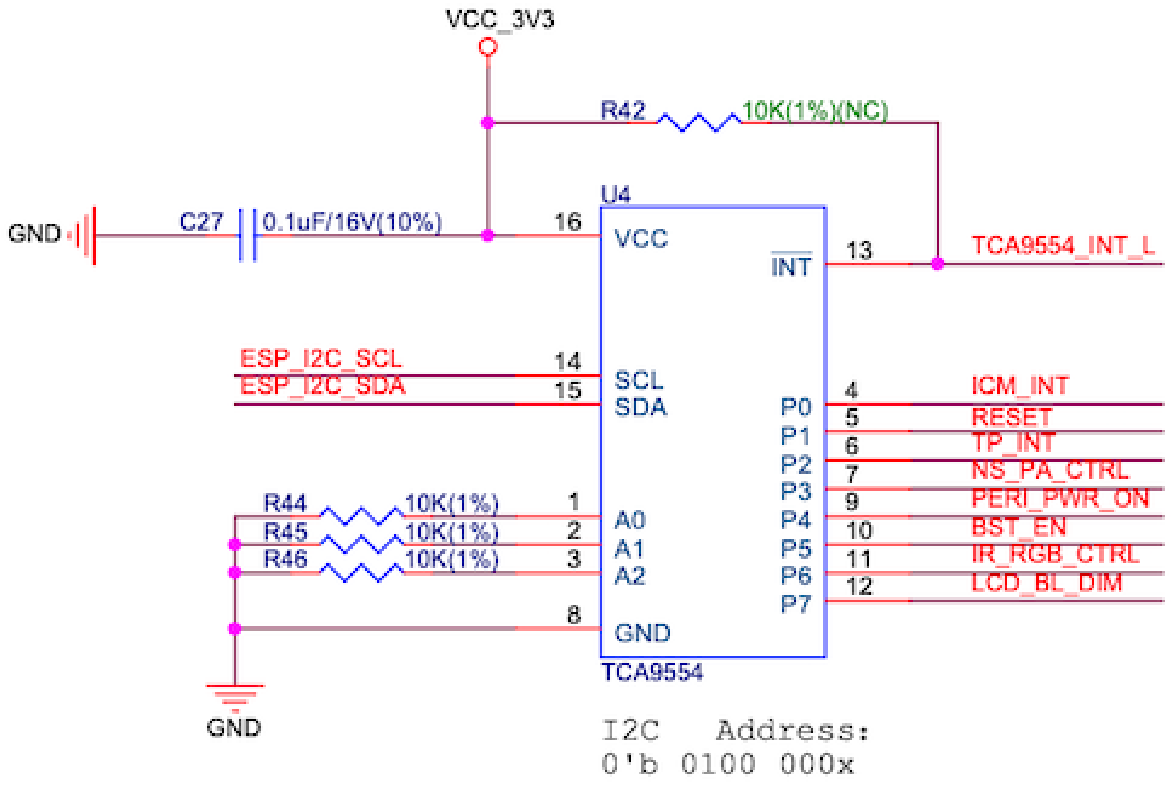


Fig. 69: ESP32-S2-HMI-DevKit-1 IO expander schematic

Power Dependencies

The following functions depend on the 5 V power domain:

- TWAI® (selects available power supply from USB 5 V or Booster 5 V automatically)
- Audio amplifier (gets power supply from USB 5 V, if it fails, will try from the battery)
- 5 V power output connector (the same as TWAI®)

The following functions depend on the controlled 3.3 V power domain:

- Micro-SD card
- Microphone and its bias circuit, and operational amplifier
- Display and touch function
- WS2812 RGB LED and IR LED
- IR LED

Power State

When the development board is connected via the USB cable, the 5 V power domain is powered on automatically and the charging IC outputs voltage to supply power for the battery. In this case, the controlled 3.3 V power domain is controlled by the P4 pin of the IO expander.

When the development board is powered by the battery, the controlled 3.3 V power domain is controlled by the P4 pin of the IO expander while the 5 V power domain is controlled by the P5 pin of the IO expander, and the charging IC will not work.

1.1.10 ESP32-Sense-Kit

[??]

The ESP32 touch sensor development kit, ESP32-Sense-Kit, is used for evaluating and developing ESP32 touch sensor system.

ESP32-Sense-Kit

[??]

Overview

The ESP32 touch sensor development kit, ESP32-Sense-Kit, is used for evaluating and developing ESP32 touch sensor system. ESP32-Sense-Kit consists of one motherboard and multiple daughterboards. The motherboard contains a display unit, a main control unit and a debug unit. The daughterboards have touch electrodes in different combinations or shapes, such as linear slider, wheel slider, matrix buttons and spring buttons, depending on the application scenarios. Users can design and add their own daughterboards for special usage cases.

The following image shows the whole ESP32-Sense-Kit.

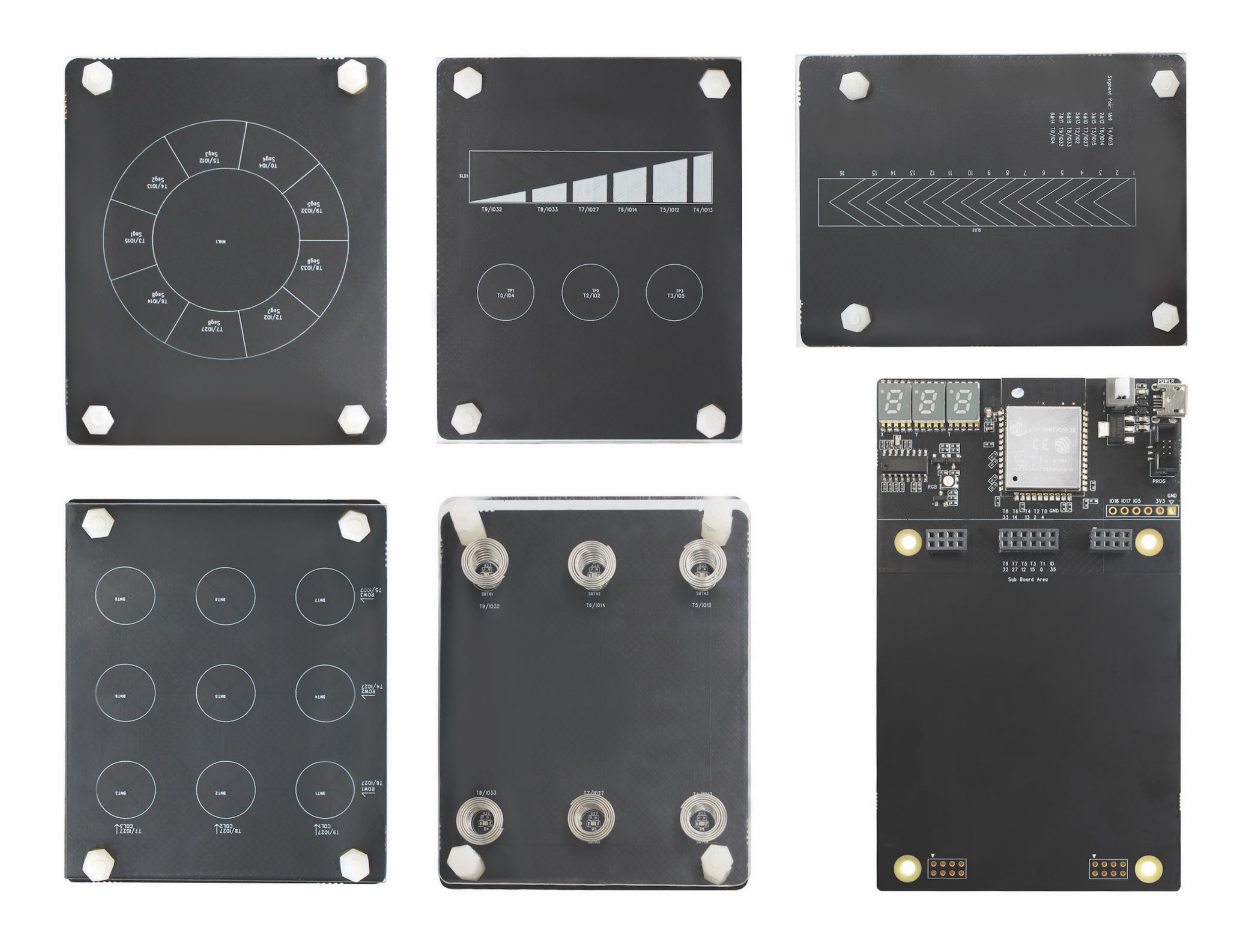


Fig. 70: ESP32-Sense-Kit

Preparation

- **Install overlay**

  If plastic is used for the overlay, the recommended thickness is 3 mm or less. Because air reduces touch sensitivity, any air gaps between the daughterboard and overlay must be eliminated. You can use double-sided adhesive tape to fill in the air gap. For the daughterboard with metal springs, 7 mm stud bolts should be used to install the overlay.

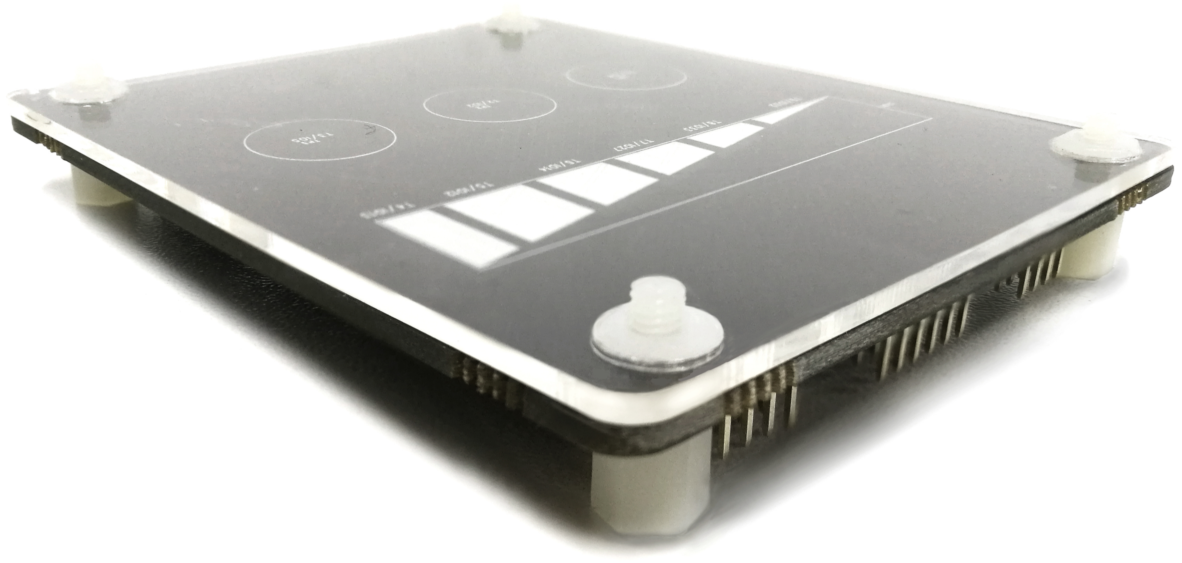


Fig. 71: Install Overlay

- **Install daughterboard**

  Use a connector to connect motherboard with daughterboard. You can use four 7 mm plastic stud bolts to have the daughterboard steadily parallel to the motherboard, as shown in the image below:

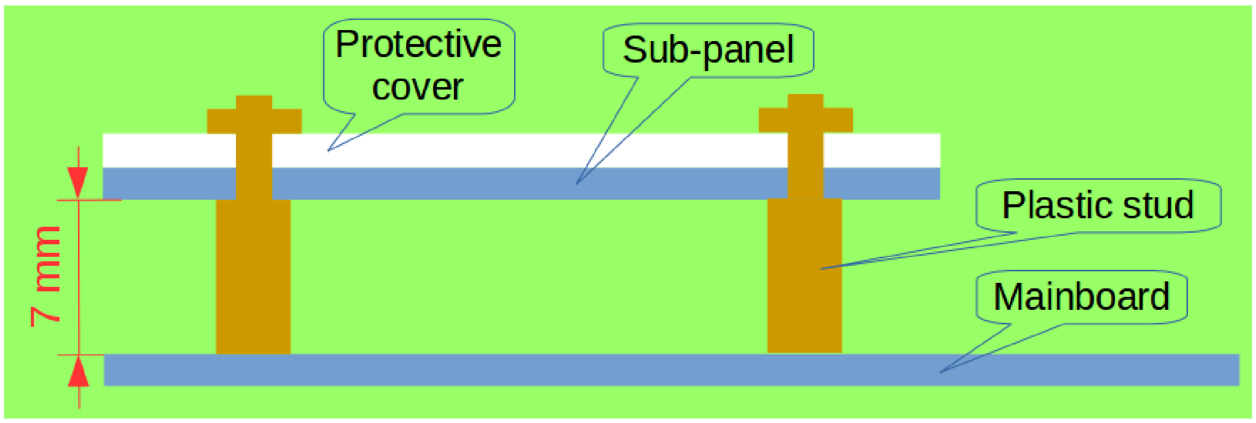


Fig. 72: Install Daughterboard

- **Set ESP-Prog debugger**

  ESP-Prog is used as the program download tool and power supply. ESP-Prog has two sets of jumpers: IO0 jumper and power supply jumper. Choose 5 V power supply for the latter. IO0 can be used both for selecting boot mode (download mode or working mode) and as a touch pin. As a result, it should be disconnected if used as a touch pin in working mode. The image below shows the settings for ESP-Prog.

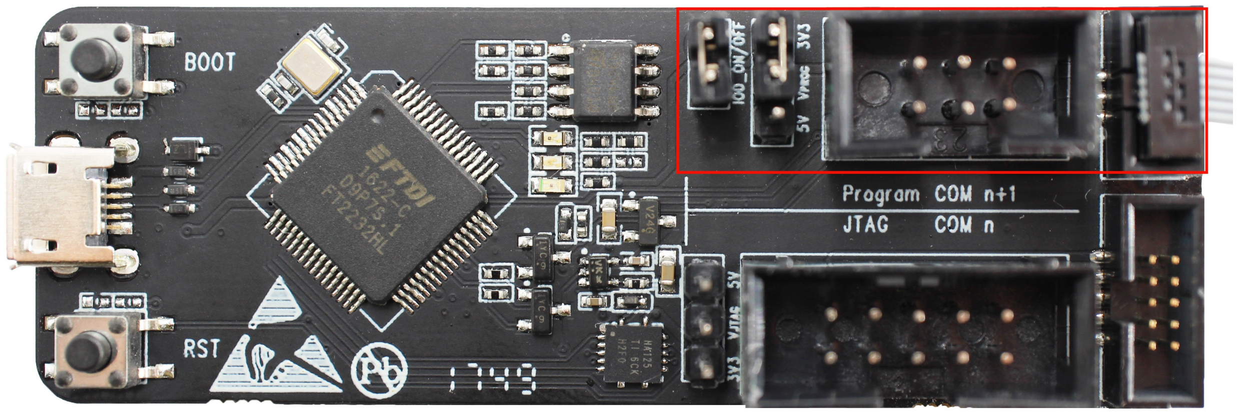


Fig. 73: Set ESP-Prog Debuggers

- **Connect ESP-Prog with motherboard**

  The ESP-Prog has a Jtag interface and a Program interface. Connect ESP-Prog and the motherboard through the Program interface.

- **Download programs**

  Run `make menuconfig` to configure the config settings for ESP32-Sense Project, as the screenshot below shows. Run `make flash` to download programs into the development board.

- **Replace daughterboard**

  ESP32 will detect the divided voltage of the voltage divider on the daughterboard when it is powered on to identify different daughterboards. Re-power on the development board after replacing the daughterboard.

Hardware Resources

Motherboard

- **Function Block Diagram**

  The image below shows the function block diagram of the motherboard.

- **Motherboard Components**

  The display unit includes three segment displays and an RGB circuit. The debug unit includes the ESP-Prog debugger interface. The main control unit includes the ESP32 module. The mini USB is the power supply.

- **Power Management System**

  The mini USB and ESP-Prog can both be the power supply for ESP32-Sense Kit. They do not interfere with each other thanks to the protection diode. The mini USB can only serve as the power supply, while ESP-Prog also supports automatic firmware downloading. The figure below shows the schematics of the power management system.

- **Display Unit**

  The display unit on the motherboard can intuitively feedback touch event. The three 7-segment displays show the location of the pad that is being touched and the duration of a touch event. The segment displays are driven by

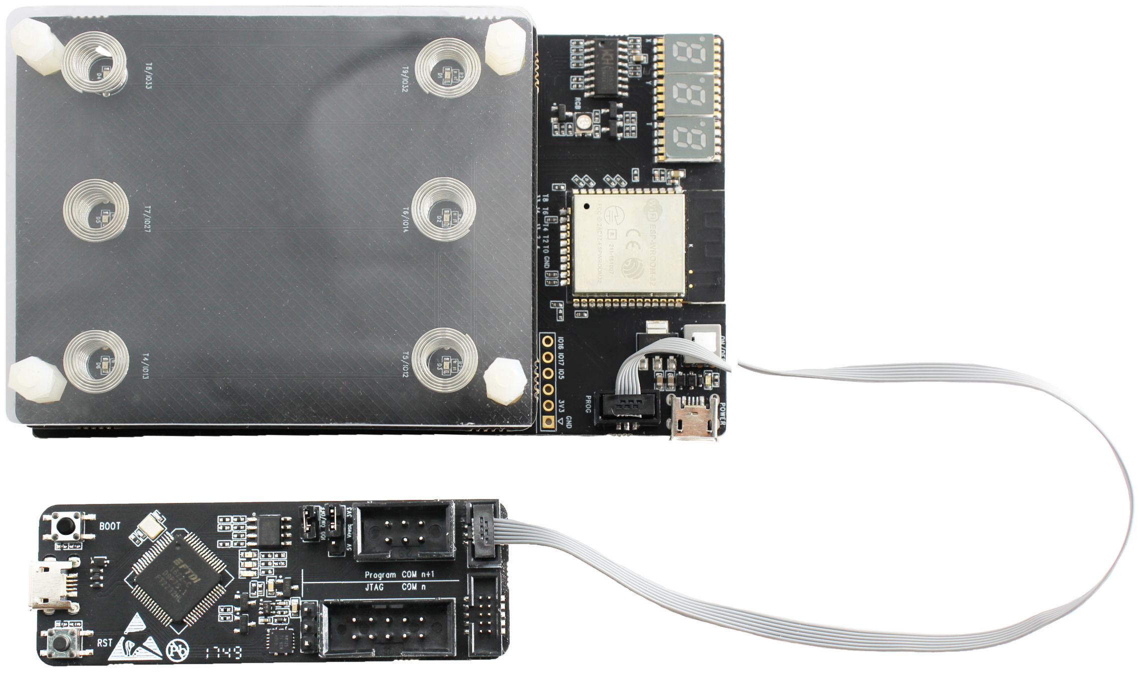
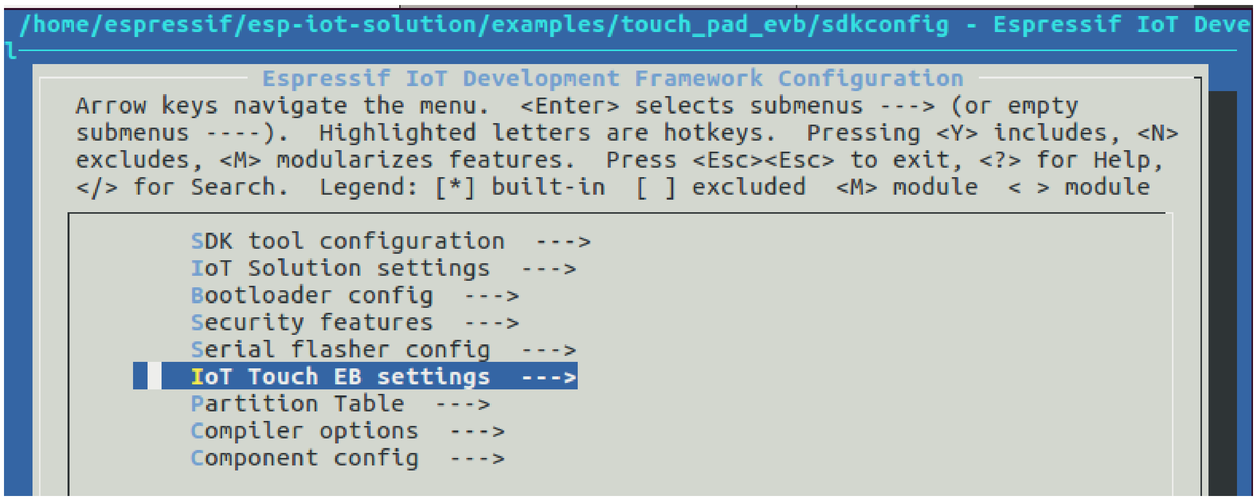
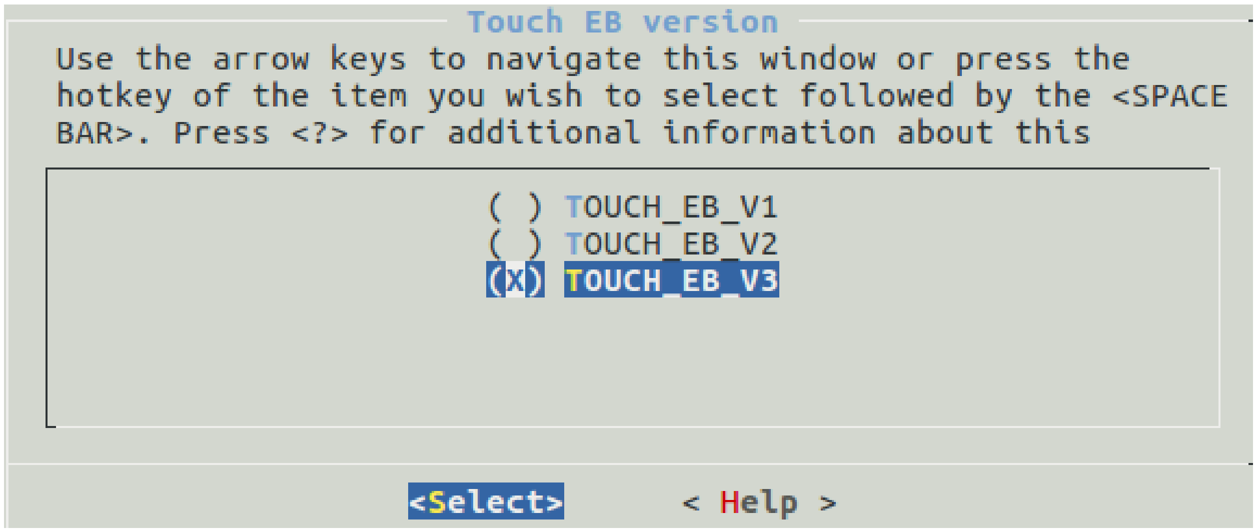


Fig. 74: Connect ESP-Prog with Motherboard



```
/home/espressif/esp-iot-solution/examples/touch_pad_evb/sdkconfig - Espressif IoT Deve
l
                Espressif IoT Development Framework Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N>
  excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help,
  </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module

          SDK tool configuration  --->
          IoT Solution settings  --->
          Bootloader config  --->
          Security features  --->
          Serial flasher config  --->
          IoT Touch EB settings  --->
          Partition Table  --->
          Compiler options  --->
          Component config  --->
```



```
Touch EB version
Use the arrow keys to navigate this window or press the
hotkey of the item you wish to select followed by the <SPACE
BAR>. Press <?> for additional information about this

( ) TOUCH_EB_V1
( ) TOUCH_EB_V2
(X) TOUCH_EB_V3

<Select>      < Help >
```

Fig. 75: Download Programs

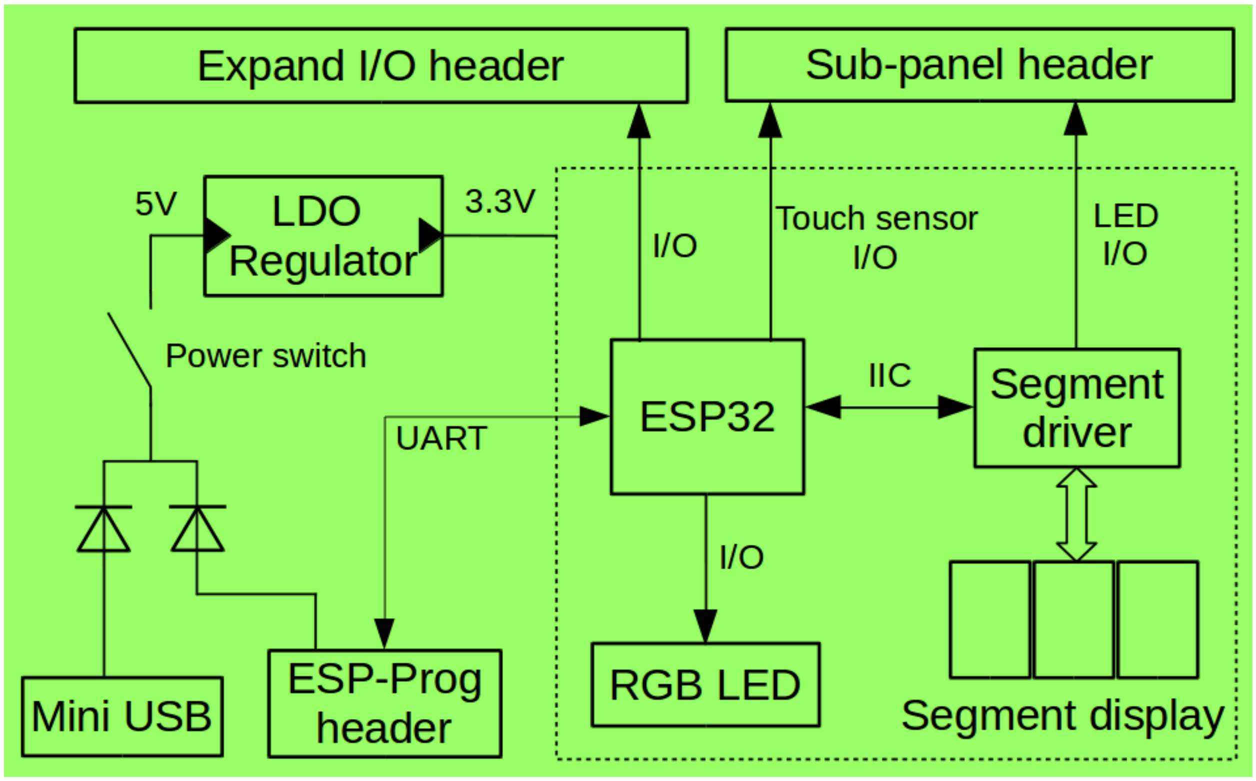


Fig. 76: Function Block Diagram

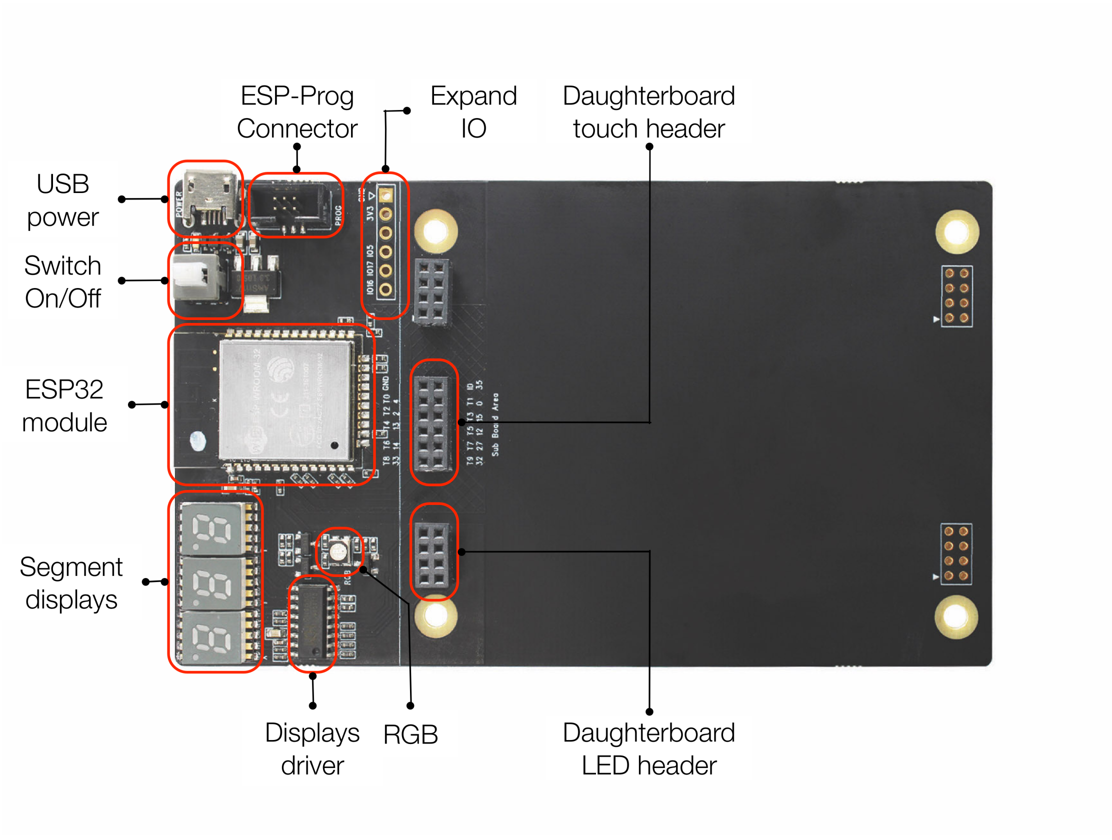


Fig. 77: Motherboard Components

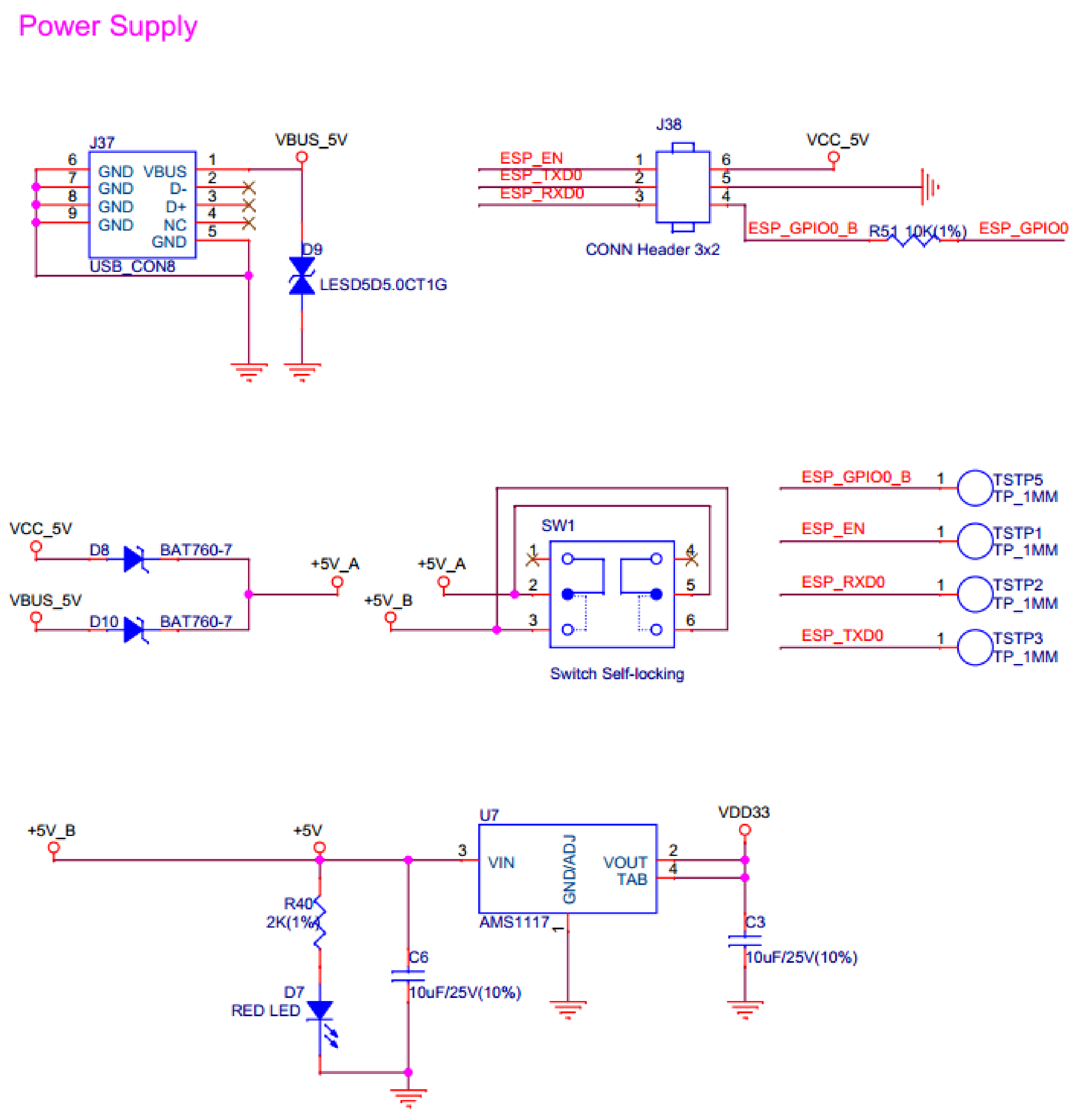
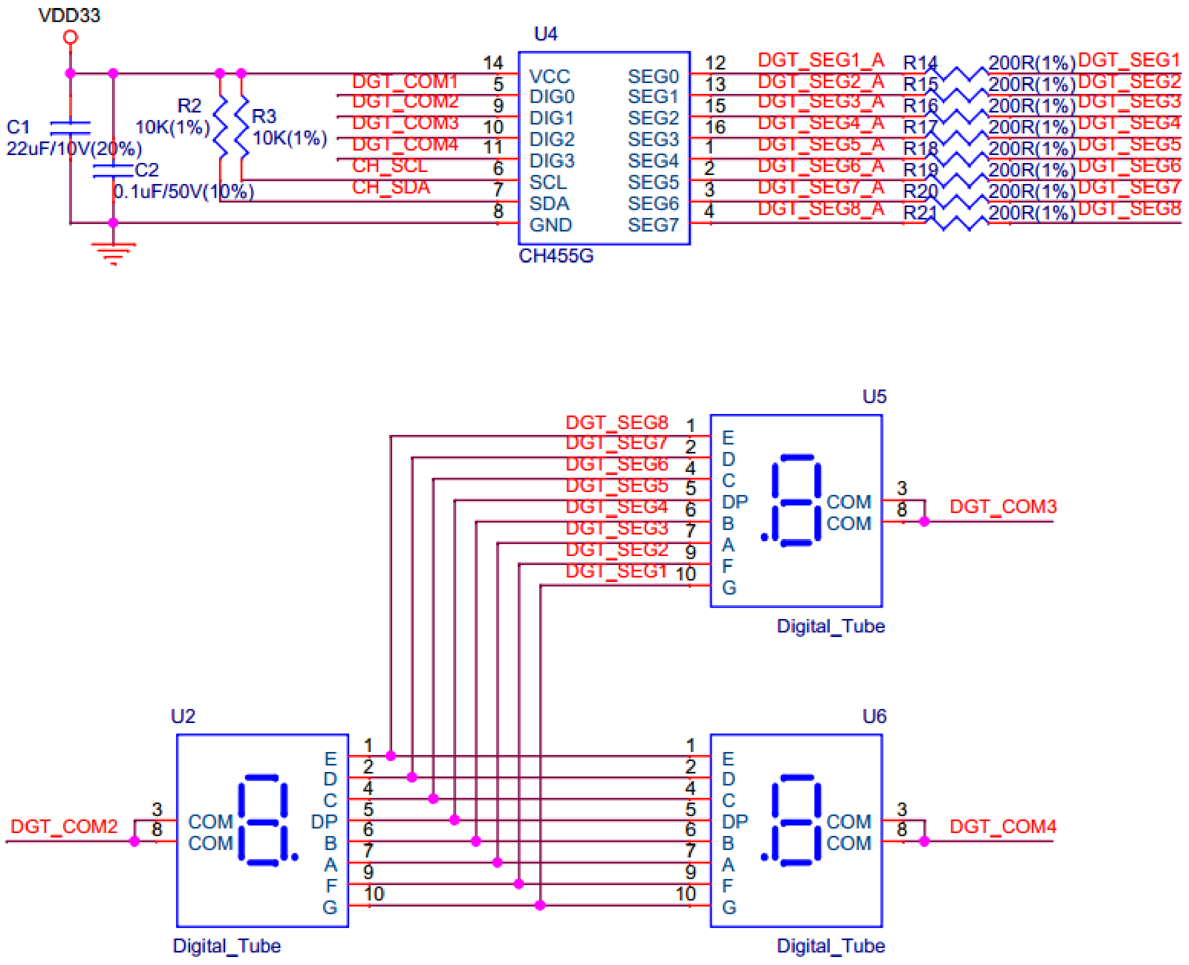


Fig. 78: Power Management System

CH455G chip, and controlled through I2C interface. The RGB LED reflects the colors when a touch event occurs. When a finger moves on the slider, the RGB LED will show the change of colors.

The figure below shows the schematics of the display unit:



Daughterboard

- **Divided resistance**

  The touch electrodes are arranged in different combinations depending on the application scenario. Each daughterboard has a voltage divider that has a unique value. The program running on motherboard reads the divider value through ADC and thus each daughterboard can be identified. The voltage divider is shown below:

Daughterboard	Divided resistance (Kohm)	ADC reading (Min)	ADC reading (Max)
Spring button	0	0	250
Linear slider	4.7	805	1305
Matrix button	10	1400	1900
Duplex slider	19.1	1916	2416
Wheel slider	47	2471	2971

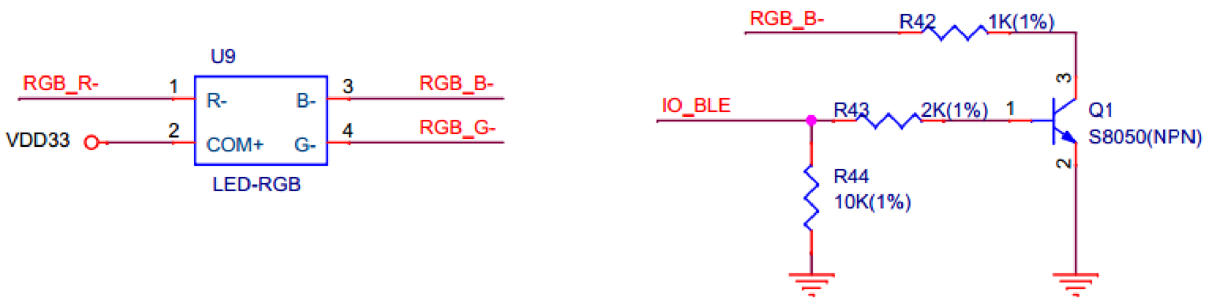


Fig. 79: Diaplay Unit

Application Programs

ESP32-Sense Project within ESP32 IoT Solution repository contains the application programs for ESP32-Sense Kit. The directory structure is shown below:

```
.
├── main
│   ├── evb_adc.c              //Identifies different daughterboards through ADC.␣
↪Sets unique ADC threshold for each daughterboard.
│   ├── evb.h                  //Configures settings for motherboard, including touch␣
↪threshold、ADC I/O、IIC I/O, etc.
│   ├── evb_led.cpp             //Initialization program of RGB LED.
│   ├── evb_seg_led.c          //Driver for digital tube.
│   ├── evb_touch_button.cpp   //Driver for touch button.
│   ├── evb_touch_wheel.cpp    //Driver for wheel slider.
│   ├── evb_touch_matrix.cpp   //Driver for matrix button.
│   ├── evb_touch_seq_slide.cpp //Driver for duplex slider.
│   ├── evb_touch_slide.cpp    //Driver for linear slider.
│   ├── evb_touch_spring.cpp   //Driver for spring button.
│   ├── Kconfig.projbuild
│   └── main.cpp               //Entry point.
├── Makefile
└── sdkconfig.defaults
```

Configure Settings

When using overlays of different thicknesses or materials, users need to reset the change rate of touch readings on each channel, that is, the sensitivity. This parameter is calculated from the pulse count value. The calculation formula is: (Non-touch value - Touch value) / Non-touch value, where "Non-touch value" refers to the pulse count value when there is no touch event, and "Touch value" refers to the pulse count value when a touch event occurs. Users need to take a measurement and obtain these two values. When the system is initialized, the touch threshold is automatically calculated from the change rate of touch readings. The touch threshold is directly proportional to the change rate.

When the change rate is set, users can write it into `evb.h` file.

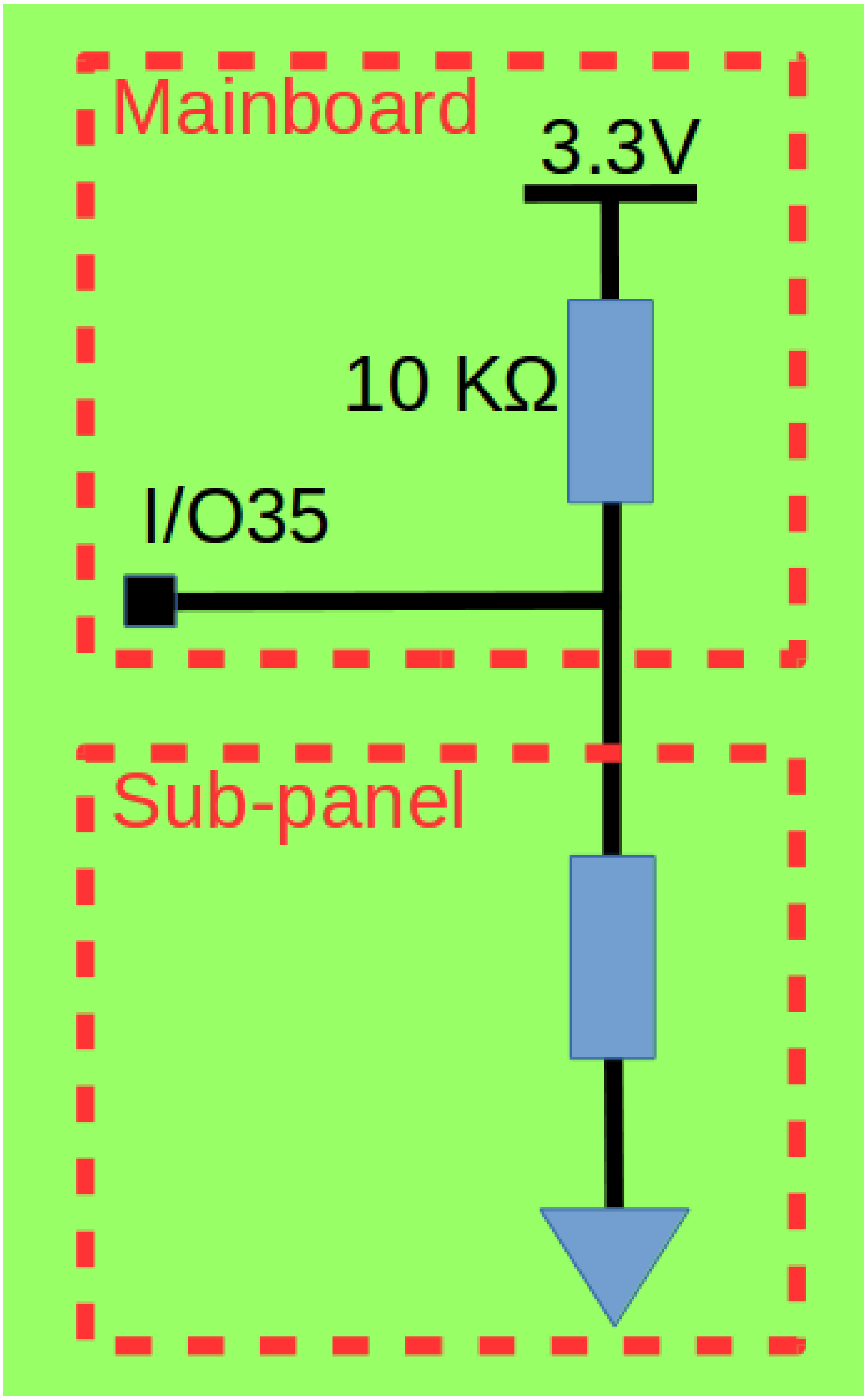
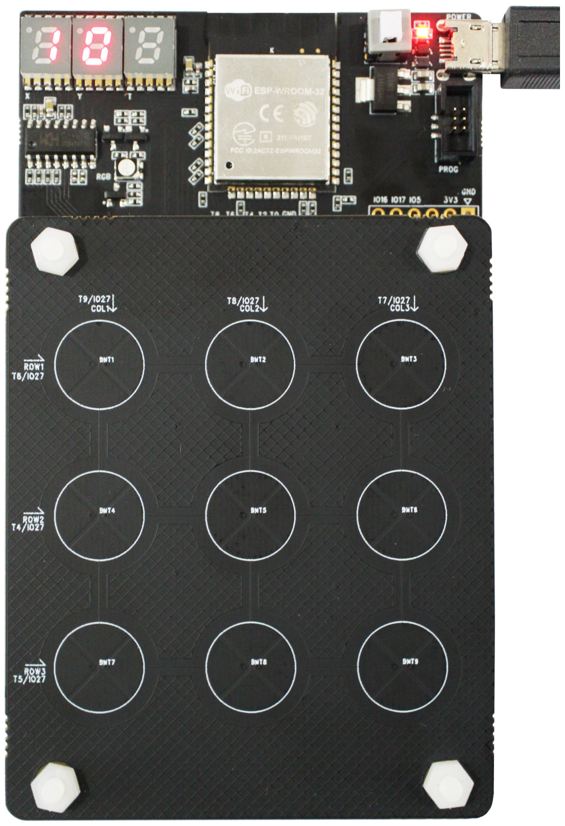
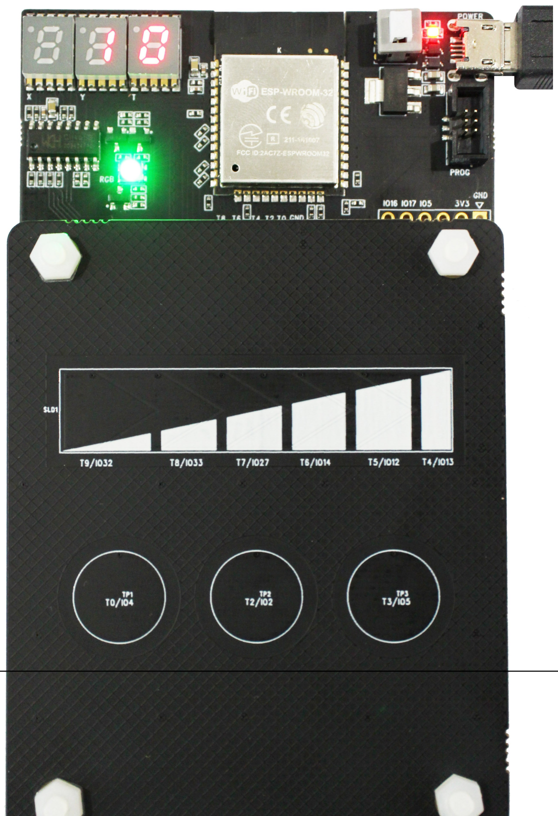
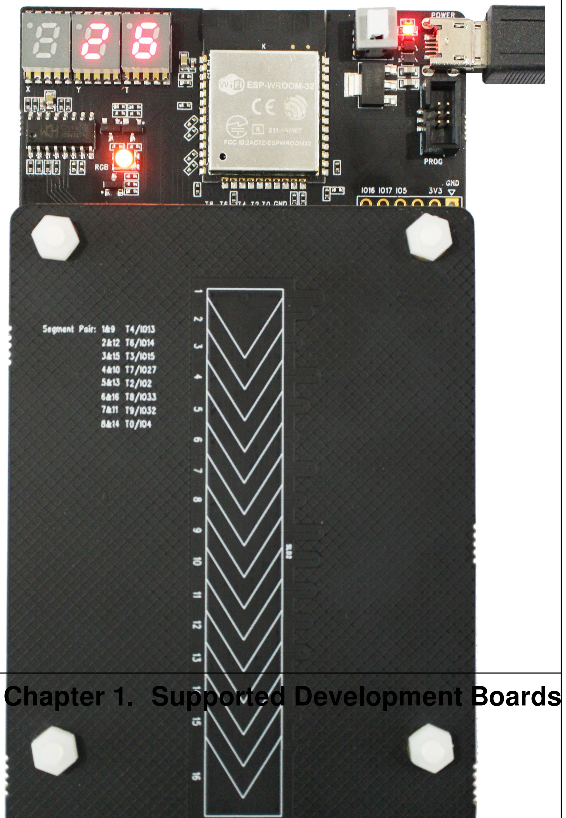


Fig. 80: Voltage Divider

The divided resistance on the motherboard is 10 KΩ. The table below lists the divided resistance on each daughterboard.

Demo

	
Spring Button	Matrix Button
	

Related Resources

- **Schematic**
  - ESP32-Sense-Kit Mainboard Schematic
  - ESP32-Sense-Kit Subboard Schematic
- **PCB Layout**
  - ESP32-Sense-Kit Mainboard PCB Layout
  - ESP32-Sense-Kit Subboard PCB Layout
- **Set up Software Environment**
  - ESP-IDF is the SDK for ESP32. You can refer to Get Started for how to set up the ESP32 software environment.
  - ESP-Prog is the debugger for ESP32 that features download and debugging functions.
- **ESP32 IoT Solution**
  - ESP32 IoT Solution project is based on ESP-IDF and contains multiple projects.
  - ESP32-Sense Project contains the programs for ESP32-Sense Kit that can be downloaded to the development board to enable touch sensor function.
- **Hardware Manuals**
  - Please click ESP32-Sense Kit Reference Design to download the hardware resources including schematics, PCB reference design, BOM and other files.
- **Useful References**
  - Espressif website.
  - ESP32 programming guide: It hosts extensive documentation for ESP-IDF ranging from hardware guides to API reference.
  - ESP32 touch sensor design: It is the reference design manual for the ESP32 touch sensor system.
- **Technical Support**
  - If you need technical support regarding ESP32-Sense-Kit, please submit a new issue referring to the ESP32-Sense Project.
- **How to buy**
  - WeChat Account: espressif_systems.
  - Purchase consulting.

1.1.11 ESP32-S2-Touch-Devkit-1

[▯▯]

ESP32-S2-Touch-Devkit-1 is a development kit that is aimed at helping evaluate and develop capacitive touch sensor applications on ESP32-S2. It is made up of a Motherboard-Subboard structure.

ESP32-S2-Touch-Devkit-1

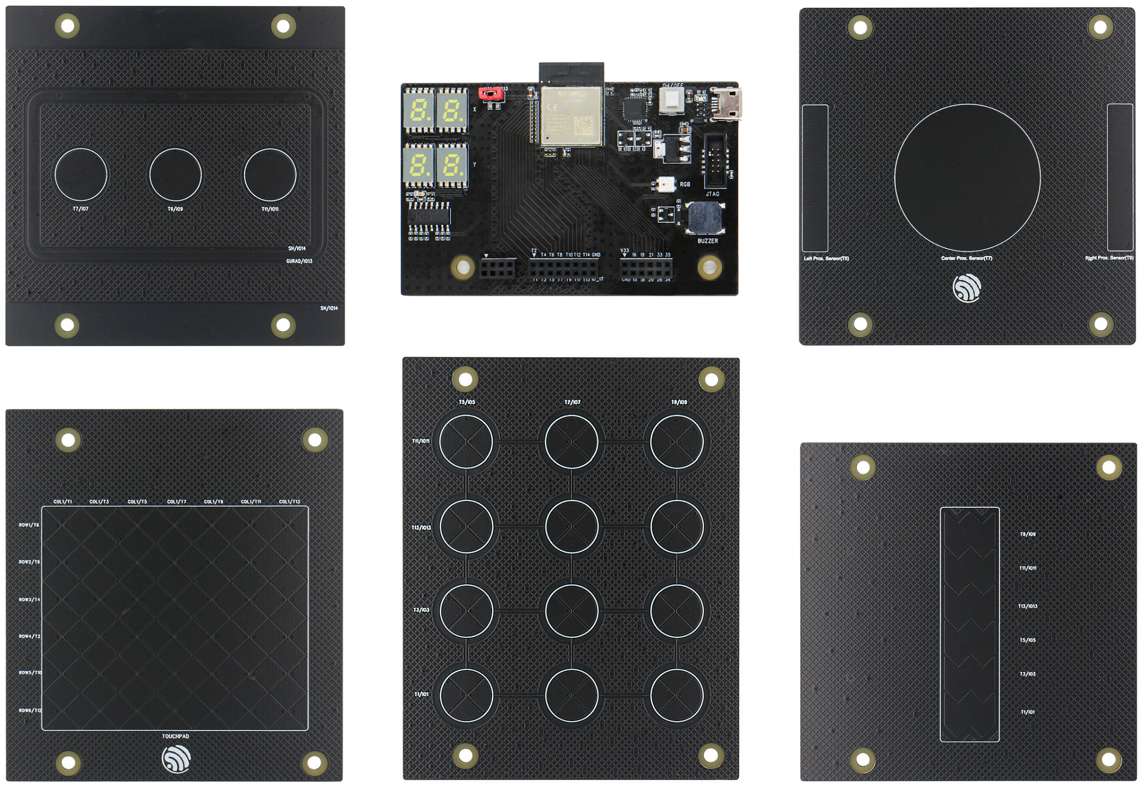


Fig. 81: ESP32-S2-Touch-Devkit-1 Board-set

Overview

ESP32-S2-Touch-Devkit-1 is a development kit that is aimed at helping evaluate and develop capacitive touch sensor applications on ESP32-S2. It is made up of Motherboard-Subboard structure. The motherboard of ESP32-S2-Touch-Devkit-1 integrates ESP32-S2-MINI-1 controller module and several useful little components such as buzzer, digital tube, RGB light, and so on. There are several kinds of subboards in ESP32-S2-Touch-Devkit-1 with different kinds of capacitive touch sensor pads, developers can choose one of them and connect it with motherboard so that they could develop different kinds of capacitive touch sensor applications. The motherboard and subboard use the pin header/female pin header as the socket connector which makes it plug in and plug out smoothly.

Motherboard

Subboards

- Button-board: Three capacitive touch buttons with waterproof sensor.

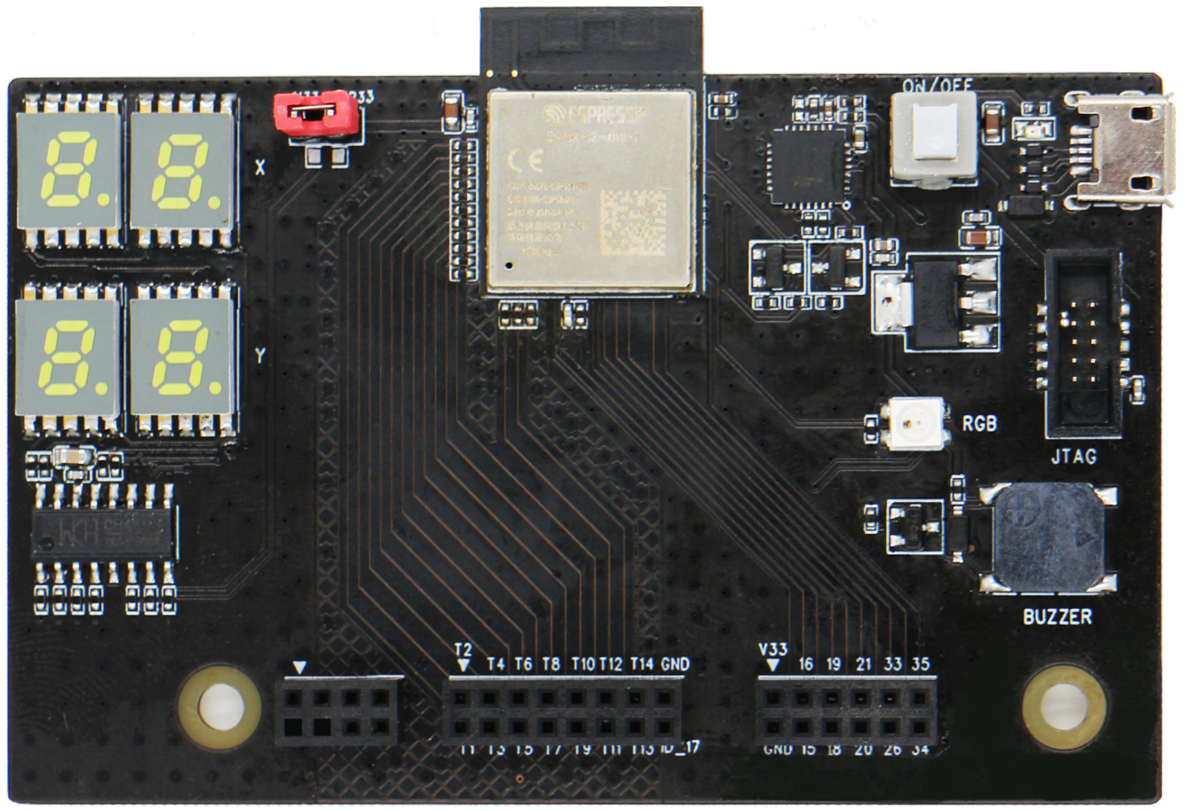
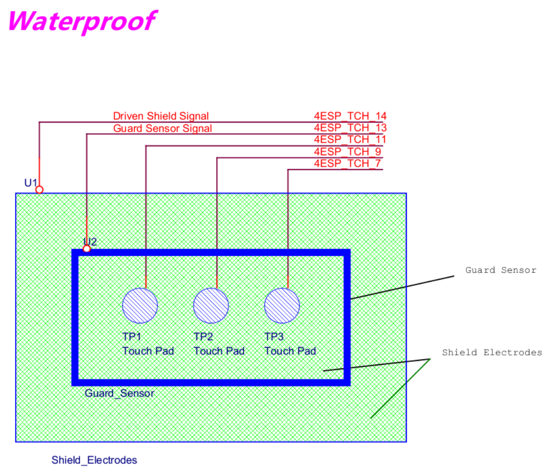
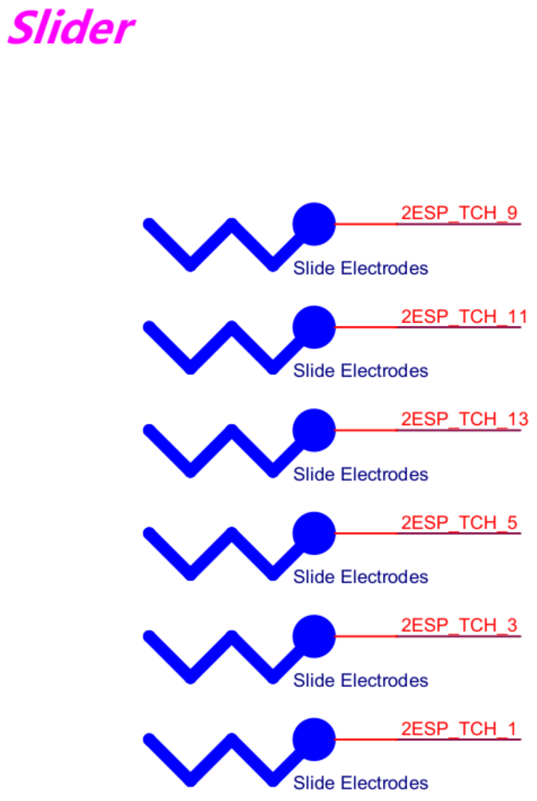


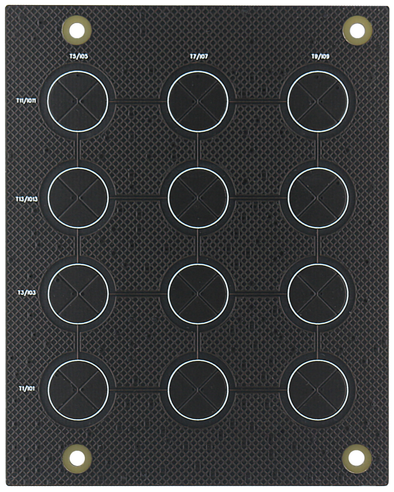
Fig. 82: Motherboard

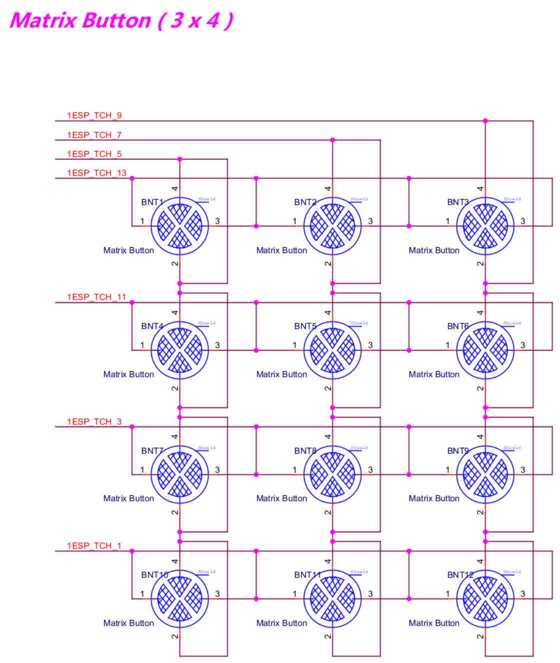
T7/IO7 T9/IO9 T11/IO11 SH/IO14 GURAD/IO13 SH/IO14	 Waterproof Driven Shield Signal Guard Sensor Signal 4ESP_TCH_14 4ESP_TCH_13 4ESP_TCH_11 4ESP_TCH_9 4ESP_TCH_7 U1 U2 TP1 Touch Pad TP2 Touch Pad TP3 Touch Pad Guard Sensor Shield Electrodes Guard_Sensor Shield_Electrodes
Button-board Water-proof sensor	

- Slider-board: Capacitive touch linear slider, the relative distance of slider has up to 8-bit precision.

T9/IO9 T11/IO11 T13/IO13 T5/IO5 T3/IO3 T1/IO1	 Slider 2ESP_TCH_9 Slide Electrodes 2ESP_TCH_11 Slide Electrodes 2ESP_TCH_13 Slide Electrodes 2ESP_TCH_5 Slide Electrodes 2ESP_TCH_3 Slide Electrodes 2ESP_TCH_1 Slide Electrodes
Slider-board Slider	

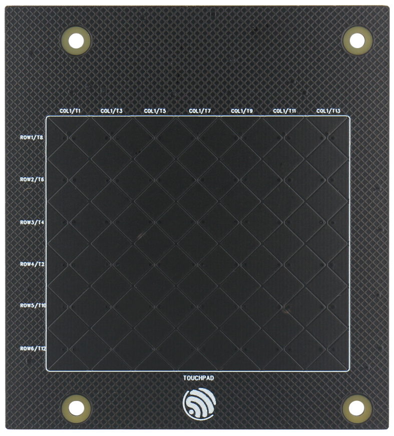
- Matrix-board: 3 x 4 capacitive touch matrix button, 7 channels of touch sensor make up of 12 capacitive touch buttons.

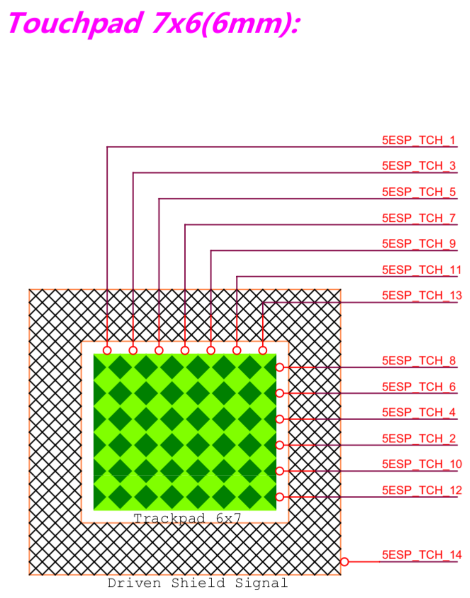




Matrix-board Matrix button

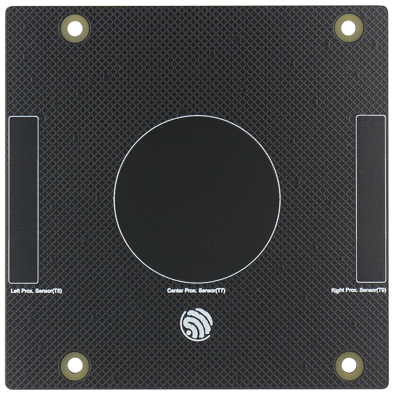
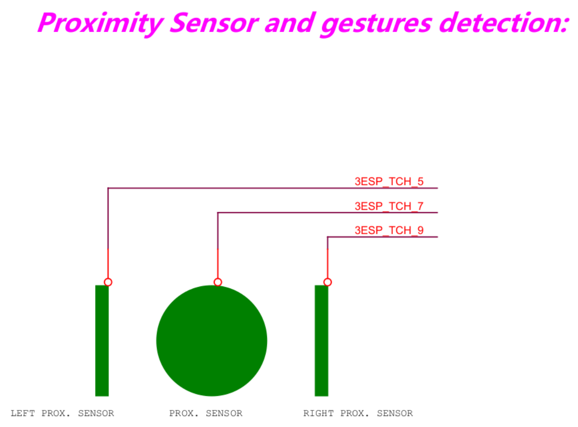
- Touchpad-board: 7 x 6 two-dimension capacitive touchpad.





Touchpad-board Touchpad

- Proximity-board: Three capacitive touch proximity sensors.

 Left Prox. Sensor(T5) Center Prox. Sensor(T7) Right Prox. Sensor(T9)	 Proximity Sensor and gestures detection: 3ESP_TCH_5 3ESP_TCH_7 3ESP_TCH_9 LEFT PROX. SENSOR PROX. SENSOR RIGHT PROX. SENSOR
Proximity-board Sensor	

Get Started

Install SDK and it's dependencies:

1. Install ESP-IDF, please refer to ESP-IDF Programming Guide.
2. Use the release-4.3 version of ESP-IDF.

```
cd esp-idf #Enter esp-idf folder
git checkout release/v4.3 #Checkout release-v4.3 version of ESP-IDF
git submodule update #Maybe is needed
```

Get ESP32-S2-Touch-Devkit-1 project:

1. Clone esp-dev-kits repo.

```
git clone https://github.com/espressif/esp-dev-kits.git
```

2. Enter ESP32-S2-Touch-Devkit-1 project folder.

```
cd esp-dev-kits/esp32-s2-touch-devkit-1
```

The project's structure is shown as followed:

```
├── CMakeLists.txt
├── components              #components' driver
│   ├── board_detect    #subboard detector
│   ├── buzzer              #buzzer driver
│   ├── digital_tube    #digital tube controller driver
│   ├── rgb_light       #RGB light driver(ws2812)
│   ├── subboards       #Subboards' application source file
│   └── touch_element       #Touch element library
├── main          #Main application demo logic
```

(continues on next page)

(continued from previous page)

```
│   ├── CMakeLists.txt
│   └── main.c
```

Build & Flash ESP32-S2-Touch-Devkit-1 demo project:

1. Enable ESP-IDF environment variable.

```
cd esp-idf
. ./export.sh
```

2. Build & Flash.

```
cd esp-dev-kits/esp32-s2-touch-devkit-1
idf.py set-target esp32s2 #Enable esp32s2 platform
idf.py build flash
```

3. Monitor log output.

```
idf.py monitor
```

Example output:

```
I (2880) Touch Demo: Slider sub-board plug in
I (22480) Touch Demo: Slider sub-board plug out
I (22480) Touch Demo: Nothing detected
I (41540) Touch Demo: Touchpad sub-board plug in
I (47700) Touchpad Board: Touchpad pressed, position: [0, 5]
I (47710) Touchpad Board: Position: [0, 5]
I (47720) Touchpad Board: Position: [0, 5]
I (47730) Touchpad Board: Position: [0, 6]
I (47740) Touchpad Board: Position: [0, 6]
I (47750) Touchpad Board: Position: [0, 6]
I (47760) Touchpad Board: Position: [0, 6]
I (47770) Touchpad Board: Position: [0, 7]
I (47780) Touchpad Board: Position: [0, 8]
I (47790) Touchpad Board: Position: [0, 9]
I (47800) Touchpad Board: Position: [0, 9]
I (47810) Touchpad Board: Position: [1, 10]
I (47820) Touchpad Board: Position: [2, 11]
I (47830) Touchpad Board: Position: [2, 12]
I (47840) Touchpad Board: Position: [3, 13]
I (47850) Touchpad Board: Position: [4, 14]
I (47860) Touchpad Board: Position: [5, 15]
I (47870) Touchpad Board: Position: [6, 16]
I (47880) Touchpad Board: Position: [7, 16]
I (47890) Touchpad Board: Position: [9, 17]
I (47900) Touchpad Board: Position: [10, 18]
I (47910) Touchpad Board: Position: [11, 18]
I (47920) Touchpad Board: Position: [11, 19]
I (47930) Touchpad Board: Position: [12, 20]
I (47940) Touchpad Board: Position: [13, 21]
I (47950) Touchpad Board: Position: [14, 21]
I (47960) Touchpad Board: Position: [14, 22]
I (47970) Touchpad Board: Position: [15, 23]
I (47980) Touchpad Board: Position: [15, 23]
I (47990) Touchpad Board: Position: [15, 24]
I (48000) Touchpad Board: Position: [15, 24]
```

(continues on next page)

(continued from previous page)

```
I (48010) Touchpad Board: Position: [15, 24]
I (48020) Touchpad Board: Position: [16, 24]
I (48030) Touchpad Board: Position: [16, 24]
I (48040) Touchpad Board: Position: [16, 24]
I (48050) Touchpad Board: Position: [16, 23]
I (48060) Touchpad Board: Position: [16, 23]
I (48070) Touchpad Board: Position: [16, 22]
I (48080) Touchpad Board: Position: [16, 22]
I (48090) Touchpad Board: Position: [16, 21]
I (48100) Touchpad Board: Touchpad released, position: [16, 21]
```

Notes

- Some new Touch Sensor features (Touchpad, Touch proximity) are not supported in ESP-IDF Touch Element library, so we copy Touch Element from ESP-IDF components into this demo project's components' folder and add the necessary features. They will appear in the future version of ESP-IDF.

Troubleshooting

Q1: Why Proximity-board is connected with Motherboard, they don't work or work abnormally?

A1: Though all of those subboards are hot-swappable theoretically, the Proximity-board needs to startup in an ideal environment(Far away from your hands). If it goes wrong, you can reset it mandatorily by releasing the power-switch.

Related Documents

Schematic

- ESP32-S2-Touch-Devkit-1 Motherboard Schematic
- Button Subboard Schematic
- Slider Subboard Schematic
- Matrix Button Subboard Schematic
- Touchpad Subboard Schematic
- Proximity Subboard Schematic

Other Documents

- Touch Element Library Programming Guide
- ESP32-S2-MINI-1 Datasheet
- ESP32-S2 Datasheet
- ESP32-S2 Technical Reference Manual

1.1.12 ESP32-MeshKit-Sense

[⍰⍰]

ESP32-MeshKit-Sense is a development board with an ESP32 module at its core. It features peripherals, such as a temperature and humidity sensor, an ambient light sensor, etc. The board can be interfaced with screens. The board is mainly used to detect the current consumption of ESP32 modules in a normal operation state or in sleep mode, when connected to different peripherals.

ESP32-MeshKit-Sense

[⍰⍰]

Overview

ESP32-MeshKit-Sense is a development board with an ESP32 module at its core. It features peripherals, such as a temperature and humidity sensor, an ambient light sensor, etc. The board can be interfaced with screens. The board is mainly used to detect the current consumption of ESP32 modules in a normal operation state or in sleep mode, when connected to different peripherals.

For more information on ESP32, please refer to ESP32 Datasheet.

Block Diagram and PCB Layout

Block Diagram

The figure below shows the block diagram of ESP32.

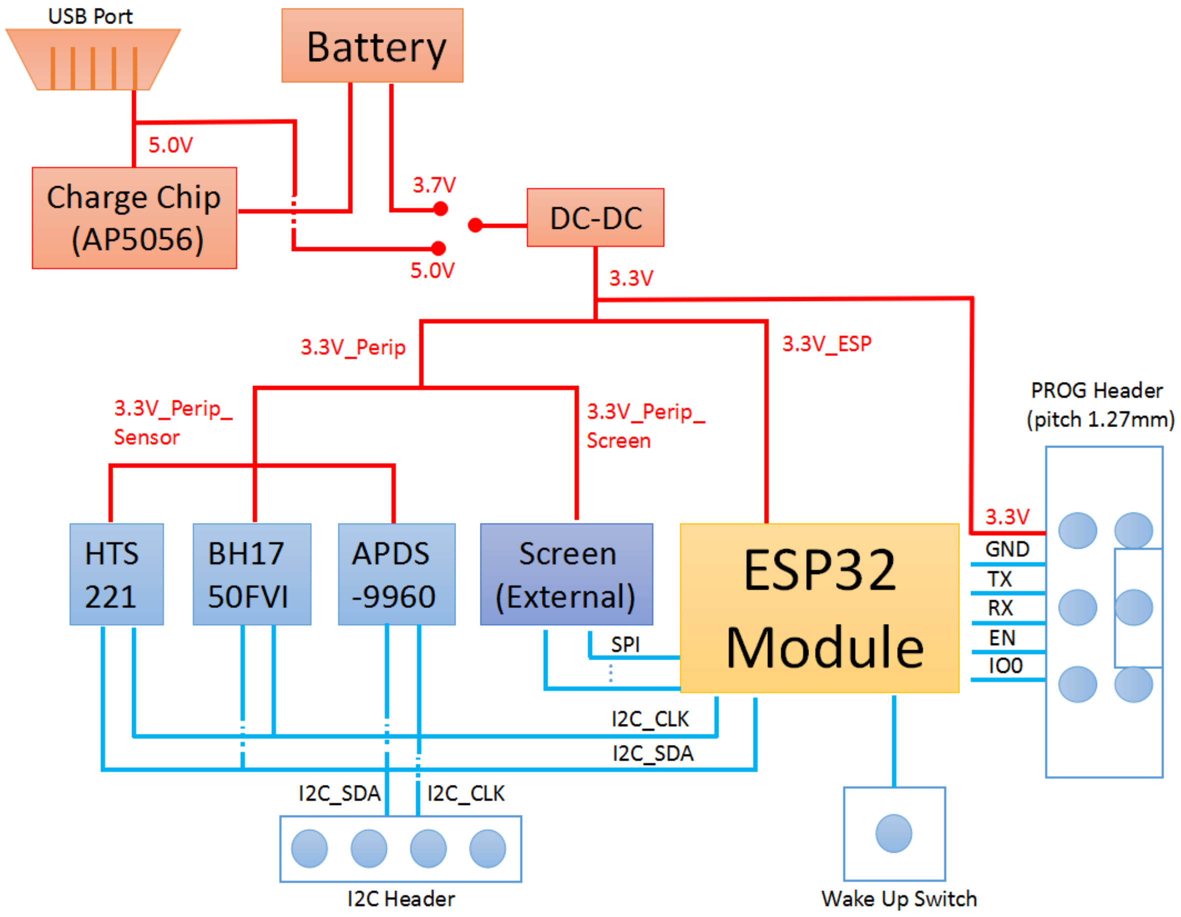


Fig. 83: ESP32 Block Diagram

PCB Layout

The figure below shows the layout of ESP32-MeshKit-Sense PCB.

Functional Descriptions of PCB Layout are shown in the following table:

PCB Elements	Description
EXT5V	5 V input from USB
CH5V	input from the electrical charging chip
CHVBA	output from the electrical charging chip
VBA	Connects to the positive electrode of the battery
SUFVCC	When the switch is toggled to the ”ON” position, it is connected to the power input. When the switch is toggled to the ”OFF” position, the power supply is disconnected.
DCVCC	Input from power management chip DC-DC
3.3V	3.3 V power output from power supply management chip
3.3V_PER	3.3 V power supply for all peripherals
3.3V_ESP	3.3 V power supply for all ESP32 modules
3.3V_SEN	3.3 V power supply for the three on-board sensors
3.3V_SCR	3.3 V power supply for the off-board screen
Charge	Battery charging indicator, D5 is a red light, indicating that charging is undergoing; D6 is a green light, indicating that charging is complete.
Sensor	Power indicator, indicating that 3.3V_Perip_Sensor is enabled
Screen	Power indicator, indicates that 3.3V_Perip_Screen is enabled
WiFi / IO15	Signal indicator, indicating that Wi-Fi connection is working properly
Network / IO4	Signal indicator, indicating the board is properly connected to the server

Functional Modules

This chapter mainly introduces each functional module (interface) and the hardware schematics for them.

Power Supply Management Module

Power Supply Management Module

The development board can be powered by battery and the AP5056 power supply management chip can be used to charge the battery. The AP5056 is a complete constant current constant voltage linear charger for single cell lithium-ion batteries. It has 4.2 V of preset charge voltage and 1 A of programmable charge current.

When both the USB power supply and the battery power supply are available, the system selection of power supply will be: VBUS is high, Q4 is in cut-off state, VBAT (battery power) is automatically cut off from the system power supply, and the USB supplies power for the system.

The figure below shows the schematics for USB/BAT power supply management.

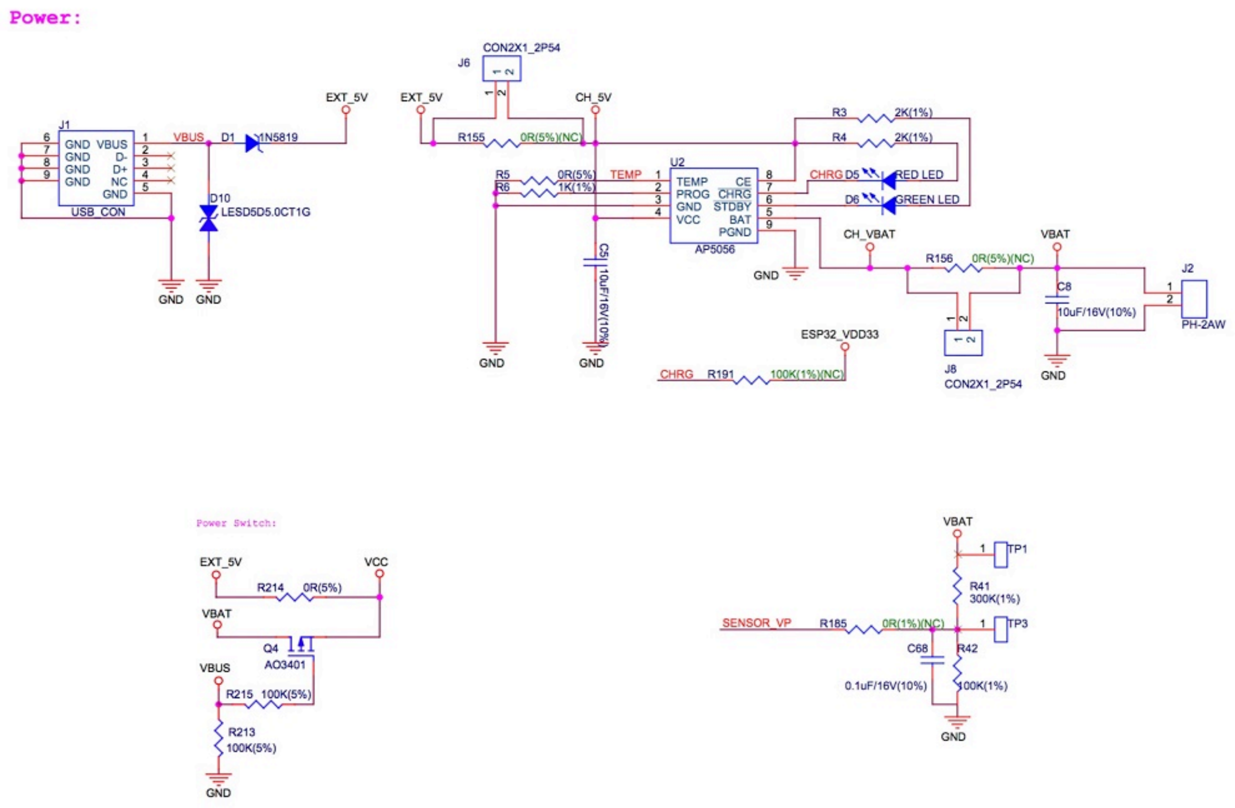
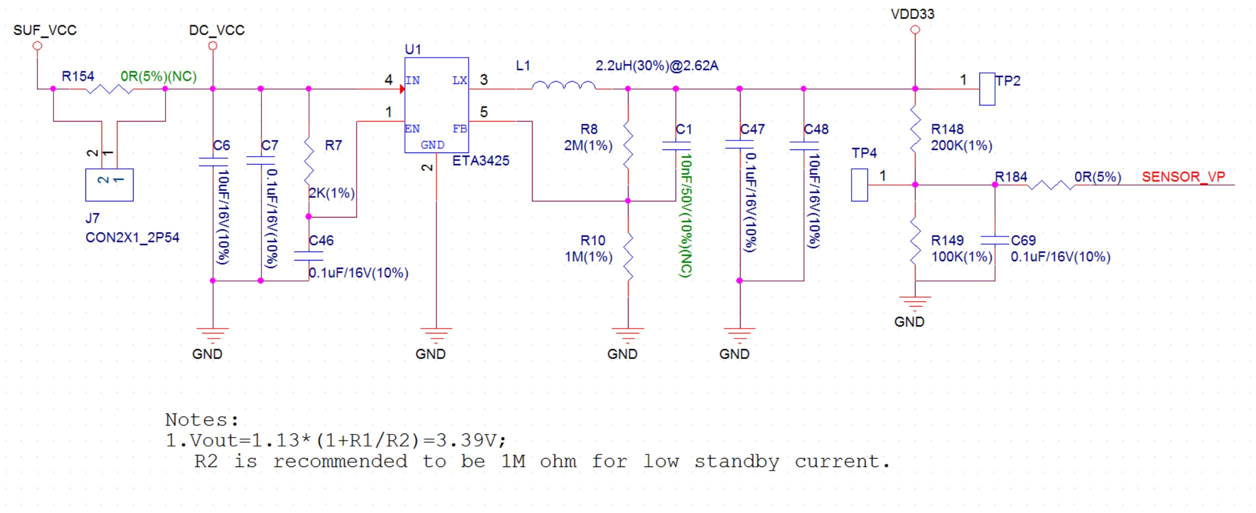


Fig. 84: USB/BAT Power Supply Management Schematics

Power Supply Management for Peripherals

First of all, the input from the USB or BAT is converted by the power management chip into a 3.3 V voltage to power the circuit. The power management chip on the board is ETA3425, which has an output voltage of 3.3 V and a maximum output current of 600 mA.

The figure below shows the schematics for peripheral power supply.



```
Notes:
1.Vout=1.13*(1+R1/R2)=3.39V;
  R2 is recommended to be 1M ohm for low standby current.
```

Fig. 85: Peripheral Power Supply Schematics

The main VDD33 circuit has two branches:

- ESP32_VDD33, used to power the ESP32 module module
- VDD33_PeriP, used to power all peripherals.

The connection between them can be controlled via the pin header and jumper cap. The figure below shows the schematics for ESP32_VDD33.

The VDD33_PeriP branch circuit also has two sub-branches

- VDD33_PeriP_Screen, dedicated power supply for the external screen
- VDD33_PeriP_Sensor, power supply for the three sensors

The connection of the two can be controlled by the module GPIO+MOS. The figure below shows the schematics for VDD33_PeriP.

Boot & UART

The development board is integrated with a PROG Header, which can be connected to a ESP-PROG development board via a cable. Users can then connect the micro USB of the ESP-PROG development board to a PC for ESP32-MeshKit-Sense firmware download and debugging.

The figure below shows the schematics for Boot & UART Circuit.

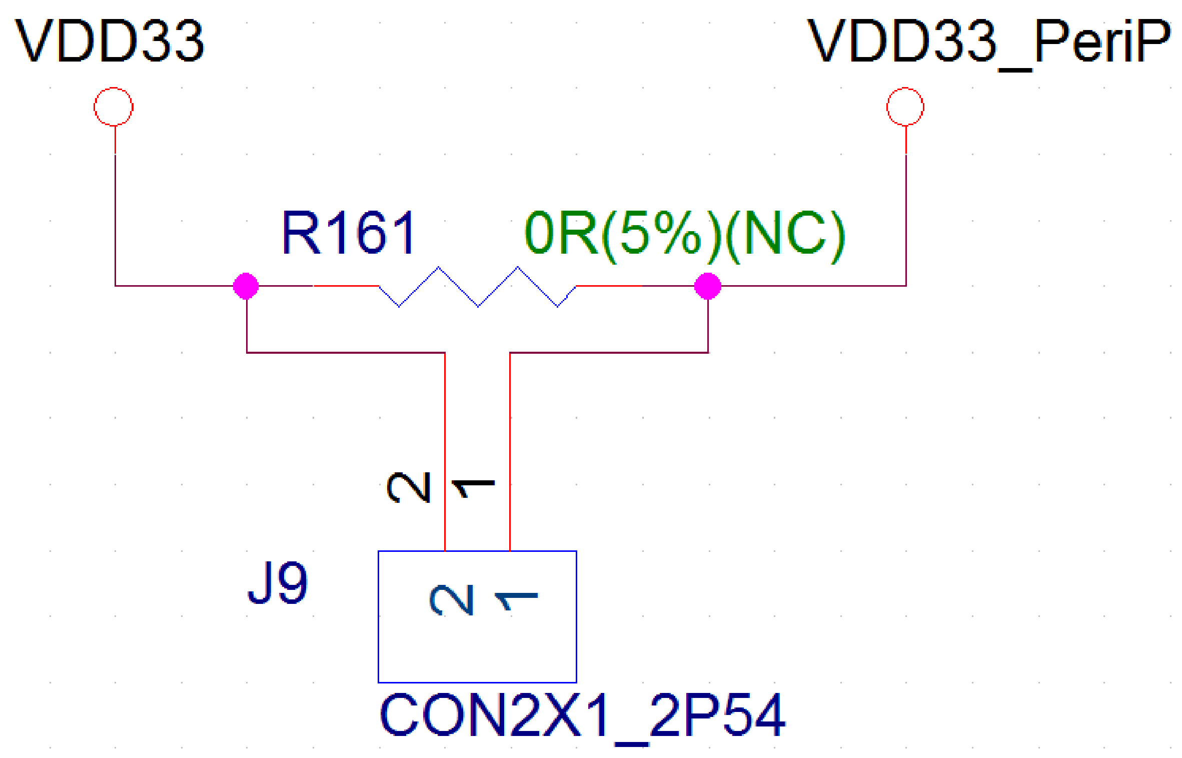


Fig. 86: ESP32_VDD33 Schematics

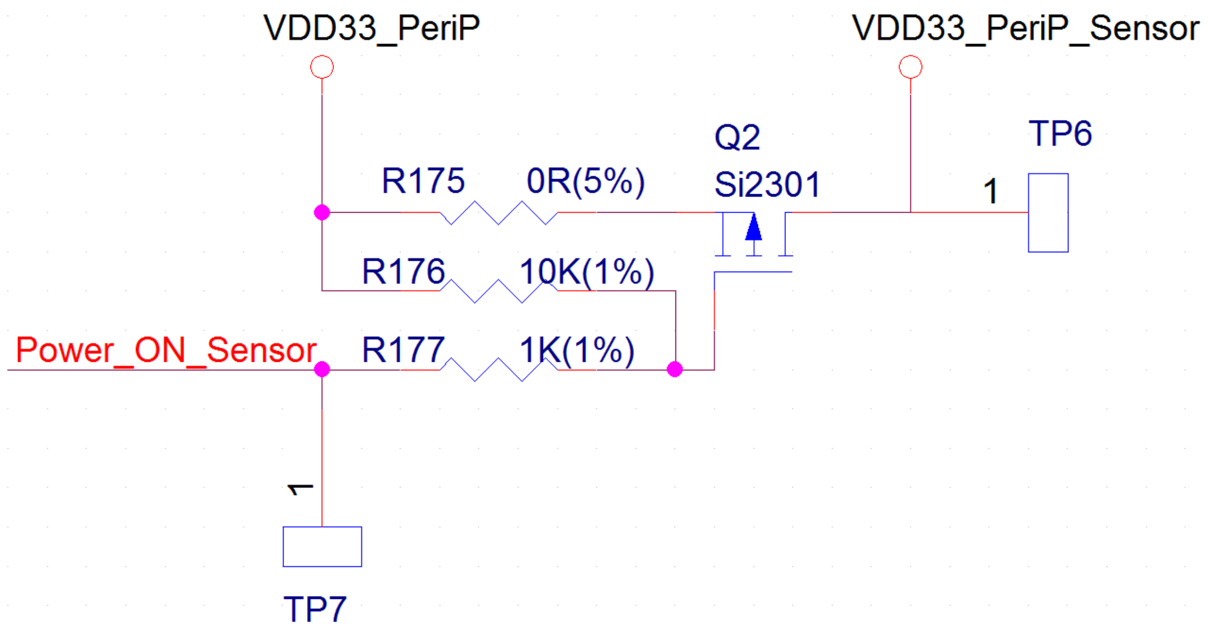


Fig. 87: VDD33_PeriP Schematics

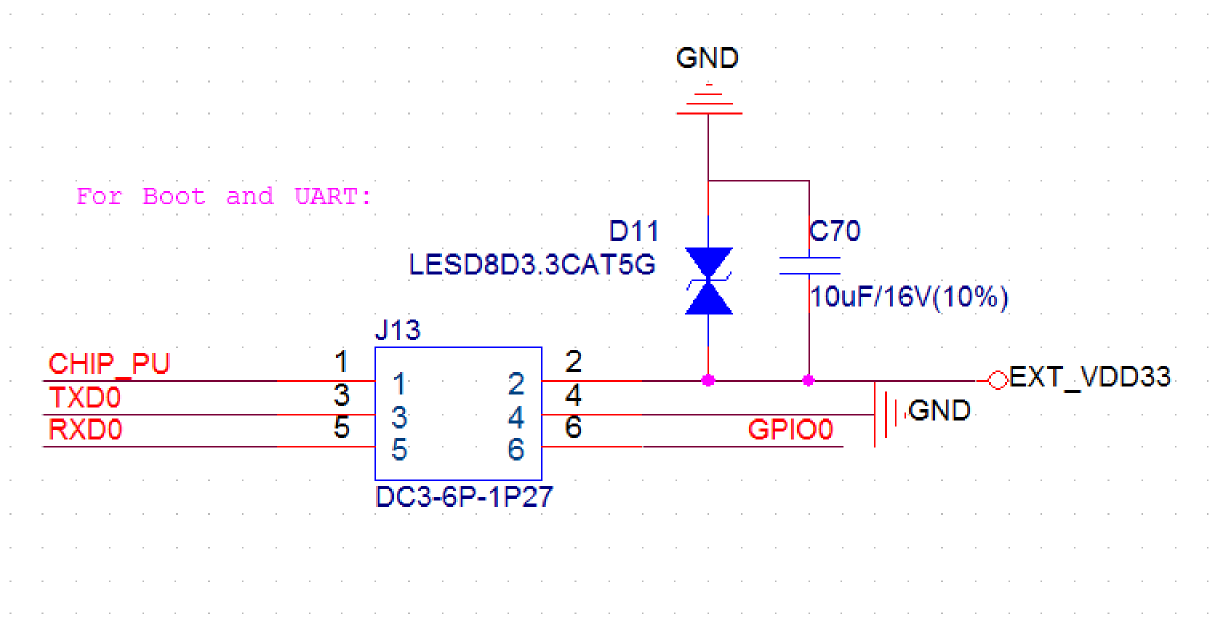


Fig. 88: Boot & UART Circuit

Module for Wakeup from Sleep

The board has a button connected to the pin IO34, which is a pin in the RTC domain. When the chip is in sleep, pressing the button will wake up ESP32.

The figure below shows the schematics for wakeup-from-sleep module.

External Screens

The development board is integrated with a screen connector that can connect different external screens to the board via cables.

The figure below shows the schematics for external screens.

Sensors

Temperature and Humidity Sensor

The HTS221 is an ultra-compact sensor for relative humidity and temperature. A 3.3 V power supply and I2C interface on the board are dedicated to HTS221.

The figure below shows the schematics for the temperature and humidity sensor.

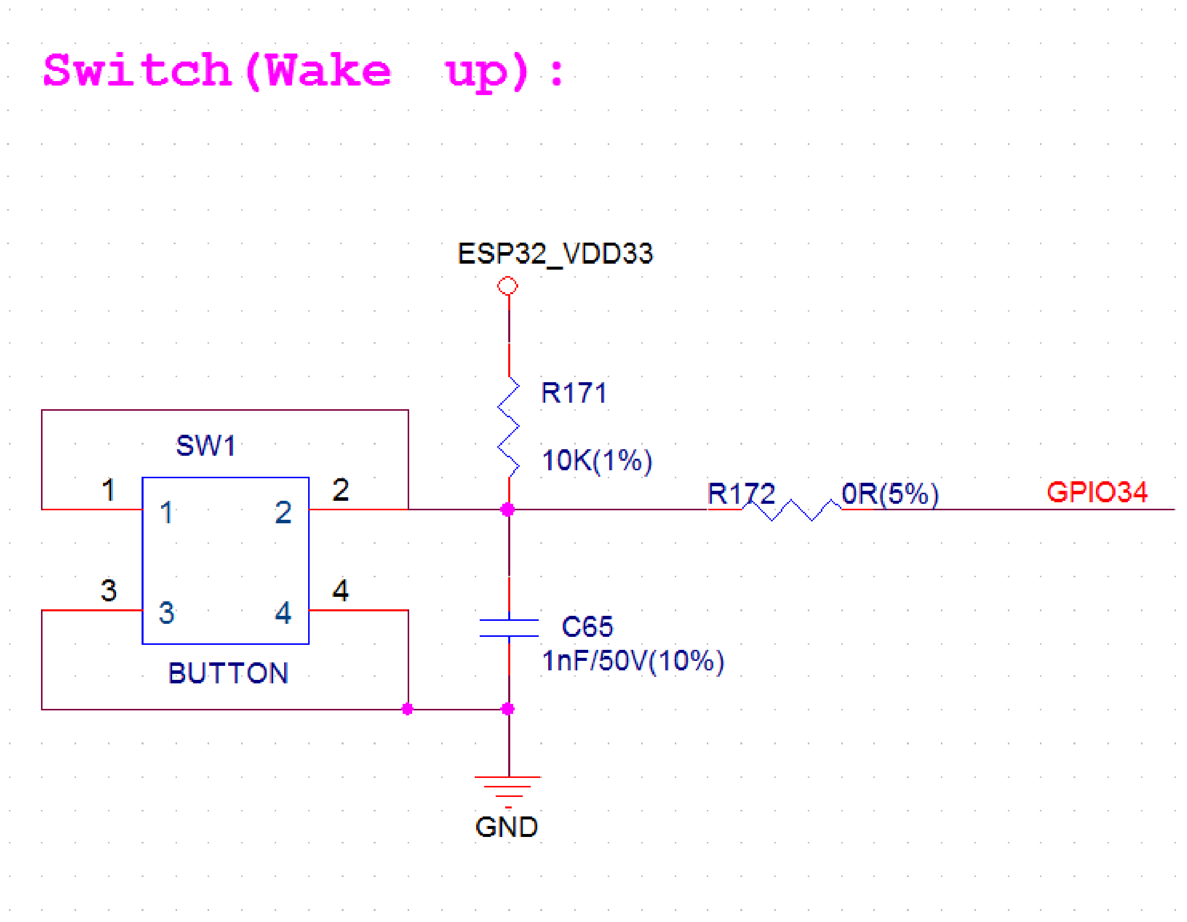


Fig. 89: Wake-from-Sleep Module Schematics

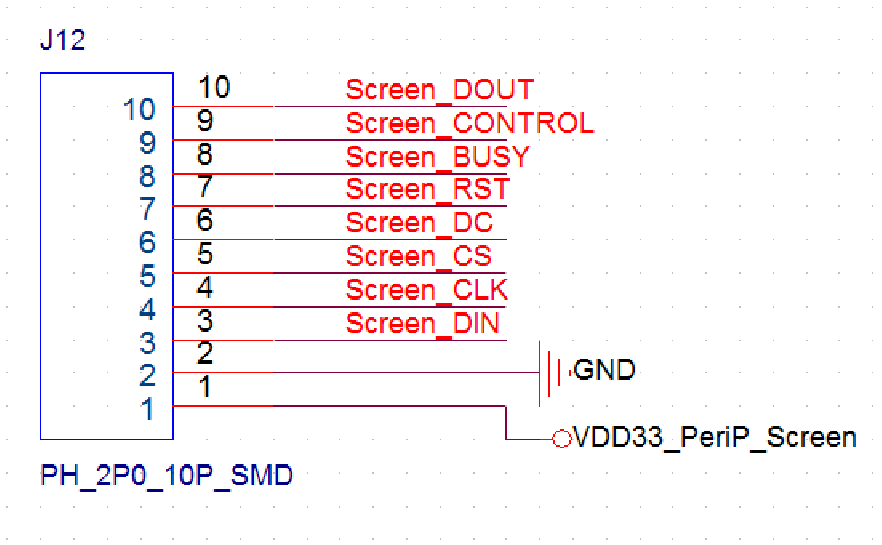


Fig. 90: Schematics for External Screens

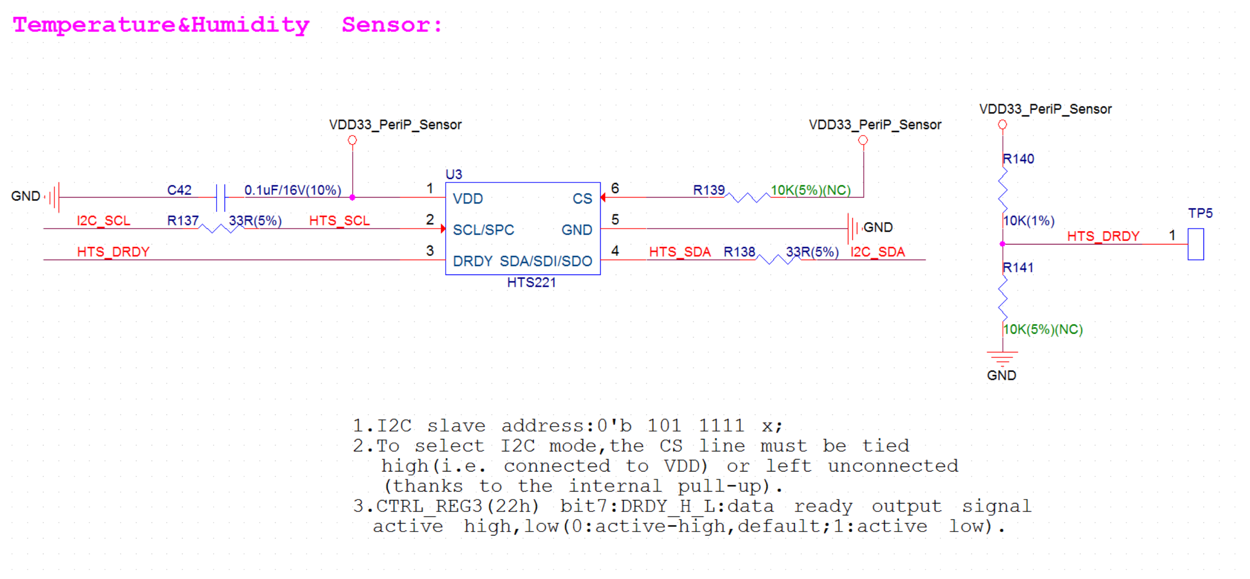

Temperature&Humidity Sensor:
VDD33_PeriP_Sensor
GND
C42
0.1uF/16V(10%)
U3
VDD
CS
SCL/SPC
GND
DRDY
SDA/SDI/SDO
HTS221
I2C_SCL
R137
33R(5%)
HTS_SCL
HTS_DRDY
R139
10K(5%)(NC)
HTS_SDA
R138
33R(5%)
I2C_SDA
R140
10K(1%)
R141
10K(5%)(NC)
TP5
1.I2C slave address:0'b 101 1111 x;
2.To select I2C mode,the CS line must be tied high(i.e. connected to VDD) or left unconnected (thanks to the internal pull-up).
3.CTRL_REG3(22h) bit7:DRDY_H_L:data ready output signal active high,low(0:active-high,default;1:active low).

Fig. 91: Temperature and Humidity Sensor Schematics

Ambient Light Sensor

The BH1750FVI is a digital ambient light sensor. A 3.3 V power supply and I2C interface on the board are dedicated to HTS221.

The figure below shows the schematics for the ambient light sensor.

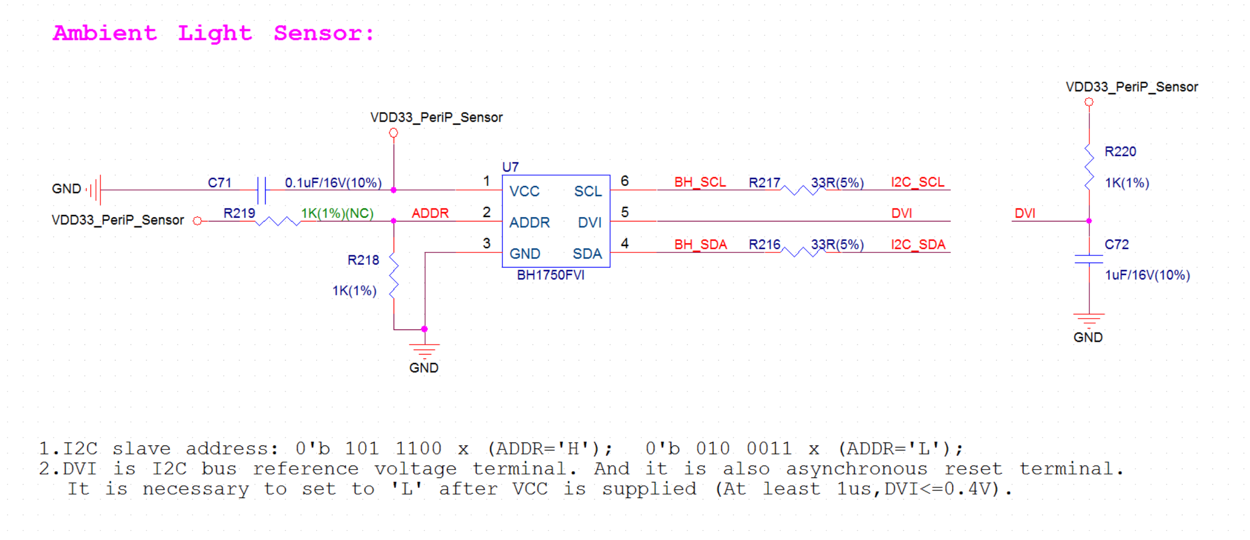
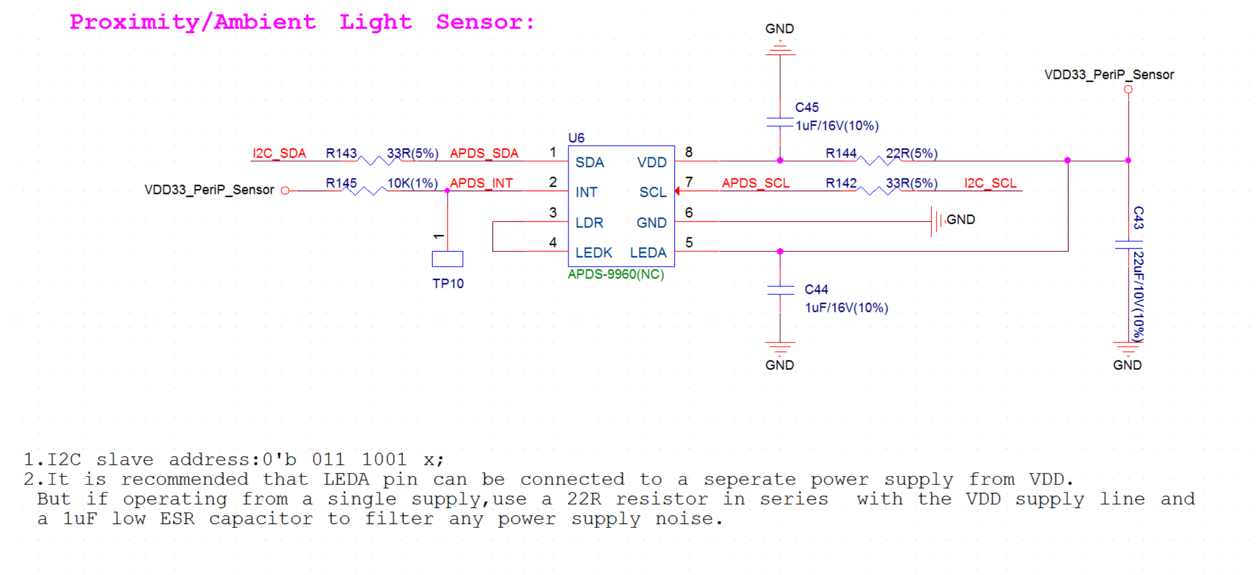

Ambient Light Sensor:
VDD33_PeriP_Sensor
GND
C71
0.1uF/16V(10%)
R219
1K(1%)(NC)
ADDR
R218
1K(1%)
U7
VCC
SCL
ADDR
DVI
GND
SDA
BH1750FVI
BH_SCL
R217
33R(5%)
I2C_SCL
DVI
BH_SDA
R216
33R(5%)
I2C_SDA
R220
1K(1%)
C72
1uF/16V(10%)
1.I2C slave address: 0'b 101 1100 x (ADDR='H'); 0'b 010 0011 x (ADDR='L');
2.DVI is I2C bus reference voltage terminal. And it is also asynchronous reset terminal. It is necessary to set to 'L' after VCC is supplied (At least 1us,DVI<=0.4V).

Fig. 92: Ambient Light Sensor Schematics

Ambient Brightness Sensor

The APDS-9960 is a ambient brightness sensor featuring advanced gesture detection, proximity detection, digital Ambient Light Sense (ALS) and Color Sense (RGBC). It also incorporates an IR LED driver. The development board uses 3.3V power supply and I2C interface. It should be noted that this device is not surface-mounted by default.

The figure below shows the schematics for the ambient brightness sensor.



```
1.I2C slave address:0'b 011 1001 x;
2.It is recommended that LEDA pin can be connected to a seperate power supply from VDD.
  But if operating from a single supply,use a 22R resistor in series  with the VDD supply line and
  a 1uF low ESR capacitor to filter any power supply noise.
```

Fig. 93: Ambient Brightness Sensor Schematics

Example

See esp-mdf/examples/development_kit/sense.

Related Documents

- ESP32-MeshKit-Sense Schematic
- ESP32-MeshKit-Sense PCB Layout

1.1.13 Contributions Guide

We welcome contributions to the esp-dev-kits project!

How to Contribute

Contributions to esp-dev-kits - fixing bugs, adding features, adding documentation - are welcome. We accept contributions via Github Pull Requests.

Before Contributing

Before sending us a Pull Request, please consider this list of points:

- Is the contribution entirely your own work, or already licensed under an Apache License 2.0 compatible Open Source License? If not then we unfortunately cannot accept it.
- Does any new code conform to the esp-dev-kits : *Style Guide* ?
- Does the code documentation follow requirements in Documenting-code ?
- Is the code adequately commented for people to understand how it is structured?
- Are comments and documentation written in clear English, with no spelling or grammar errors?
- If the contribution contains multiple commits, are they grouped together into logical changes (one major change per pull request)? Are any commits with names like "fixed typo" squashed into previous commits?
- If you're unsure about any of these points, please open the Pull Request anyhow and then ask us for feedback.

Pull Request Process

After you open the Pull Request, there will probably be some discussion in the comments field of the request itself.

Once the Pull Request is ready to merge, it will first be merged into our internal git system for in-house automated testing.

If this process passes, it will be merged onto the public github repository.

Legal Part

Before a contribution can be accepted, you will need to sign our contributor-agreement. You will be prompted for this automatically as part of the Pull Request process.

Related Documents

esp-dev-kits [illegible]

[illegible]

- [illegible]
- [illegible]
- [illegible]
- [illegible] ESP-IDF [illegible]
- [illegible]

[illegible]

- components[illegible] README [illegible]
- docs[illegible]rst [illegible]API [illegible]
- examples[illegible]
- tools[illegible]CI [illegible]

[illegible]

- [illegible] .c [illegible] .h [illegible]
- [illegible] .h [illegible] .h [illegible]
- [illegible]
- [illegible]
- [illegible];
- [illegible]

```
#ifndef _IOT_I2C_BUS_H_
#define _IOT_I2C_BUS_H_

#endif
```

- [illegible] extern "C" [illegible] C/C++ [illegible]

```
#ifdef __cplusplus
extern "C"
{
#endif

//c code

#ifdef __cplusplus
}
#endif
```

[illegible]

- [illegible] VSCODE [illegible] Doxygen Documentation Generator [illegible]
- [illegible]
- [illegible]

[illegible]

```
/**
* @brief
*
* @param port
* @param conf
* @return i2c_bus_handle_t
```

(continues on next page)

(continued from previous page)

```
*/
i2c_bus_handle_t iot_i2c_bus_create(i2c_port_t port, i2c_config_t* conf);
```

[illegible]

```
/**
* @brief Create an I2C bus instance then return a handle if created successfully.
* @note Each I2C bus works in a singleton mode, which means for an i2c port only one␣
↪group parameter works. When
* iot_i2c_bus_create is called more than one time for the same i2c port, following␣
↪parameter will override the previous one.
*
* @param[in] port I2C port number
* @param[in] conf Pointer to I2C parameters
* @return i2c_bus_handle_t Return the I2C bus handle if created successfully, return␣
↪NULL if failed.
*/
i2c_bus_handle_t iot_i2c_bus_create(i2c_port_t port, i2c_config_t* conf);
```

- [illegible]

```
// Copyright 2019-2020 Espressif Systems (Shanghai) PTE LTD
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at

//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.
```

[illegible]

- [illegible]
- [illegible] `static`[illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]`snake_case` [illegible] `_`；
- [illegible]

□□□□□□	□□□□	□□
iot_type_xxx	iot_sensor_xxx; iot_board_xxx; iot_storage_…	□□□□□ iot □□
type_xxx	imu_xxx; light_xxx; eeprom_xxx	□□□□□□□□
name_xxx	mpu6050_xxx;	□□ driver□□□□□□□□□□□□□□□□
xxx_creat / xxx_delete		□□□□□
xxx_read / xxx_write		□□□□

□□□□

- □□□□□□□□□□□□□□□□□□□□□ get_ set_ □□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□ static□
- □□□□□□□□ g_ □□□□□□□□□□□ s_ □□□
- □□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□ NULL;
- □□□□□□ snake_case □□□□□□□□□□□□□□□□ _□
- □□□□□□□□□□ data □□□□□ dat□
- □□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□
- □□□□□□□

□□	□□	□□
□□□□	□□□□	x
□□□□□□	static □□ □ g_ □□□□□□	static uint32_t g_connect_num = 0;
□□□□□□	static □□ □ s_ □□□□□□	static uint32_t s_connect_num= 0;
□□□□□□	□□□□□ i j k	
□□□□	abbreviations-in-code	addr,buf ,cfg , cmd, , ctrl,

- □□□□□□

□□	□□	□□	□□	□□	□□	□□	□□
addr	address	id	identifier	len	length	ptr	pointer
buf	buffer	info	information	obj	object	ret	return
cfg	command	hdr	header	param	parameter	temp	temporary□temperature
cmd	command	init	initialize	pos	position	ts	timestamp

[illegible]

- [illegible] `snake_case` [illegible] `_t` [illegible]

```c
typedef int signed_32_bit_t;
```

- [illegible] typedef [illegible]

```c
typedef enum {
    MODULE_FOO_ONE,
    MODULE_FOO_TWO,
    MODULE_FOO_THREE
} module_foo_t;
```

[illegible]

[illegible] ESP-IDF [illegible]

1. [illegible]

[illegible] 4 [illegible] tab [illegible] 4 [illegible]

2. [illegible]

[illegible]

```c
void function1()
{
    do_one_thing();
    do_another_thing();
                                // INCORRECT, don't place empty line here
}
                                // place empty line here
void function2()
{
                                // INCORRECT, don't use an empty line here
    int var = 0;
    while (var < SOME_CONSTANT) {
        do_stuff(&var);
    }
}
```

[illegible] 120 [illegible]

3. [illegible]

[illegible]

```
if (condition) {     // correct
    // ...
}

switch (n) {         // correct
    case 0:
        // ...
}

for(int i = 0; i < CONST; ++i) {     // INCORRECT
    // ...
}
```

[illegible]

```
const int y = y0 + (x - x0) * (y1 - y0) / (x1 - x0);     // correct

const int y = y0 + (x - x0)*(y1 - y0)/(x1 - x0);         // also okay

int y_cur = -y;                                          // correct
++y_cur;

const int y = y0+(x-x0)*(y1-y0)/(x1-x0);                 // INCORRECT
```

. [illegible] -> [illegible]

[illegible]:

```
gpio_matrix_in(PIN_CAM_D6,   I2S0I_DATA_IN14_IDX, false);
gpio_matrix_in(PIN_CAM_D7,   I2S0I_DATA_IN15_IDX, false);
gpio_matrix_in(PIN_CAM_HREF, I2S0I_H_ENABLE_IDX,  false);
gpio_matrix_in(PIN_CAM_PCLK, I2S0I_DATA_IN15_IDX, false);
```

- [illegible]PIN_CAM_VSYNC[illegible]
- [illegible]

4. [illegible]

[illegible]

```
// This is correct:
void function(int arg)
{

}

// NOT like this:
void function(int arg) {

}
```

[illegible]

```c
if (condition) {
    do_one();
} else if (other_condition) {
    do_two();
}
```

5. [illegible]

// [illegible] // [illegible] / * * / [illegible]

[illegible]

- [illegible]

```c
void init_something()
{
    setup_dma();
    // load_resources();                // WHY is this thing commented, asks the␣
↪reader?
    start_timer();
}
```

- [illegible] git [illegible]

```c
void init_something()
{
    setup_dma();
    // TODO: we should load resources here, but loader is not fully integrated yet.
    // load_resources();
    start_timer();
}
```

- `#if 0 ... #endif` [illegible] `#if 0 ... #endif` [illegible]
- [illegible] git [illegible]

```c
void init_something()
{
    setup_dma();
    // XXX add 2016-09-01
    init_dma_list();
    fill_dma_item(0);
    // end XXX add
    start_timer();
}
```

6. [illegible]

commit [illegible] LF[illegible]Unix[illegible]

Windows [illegible] git [illegible] checkout [illegible] CRLF[illegible]Windows [illegible] core.autocrlf [illegible] commit [illegible] LF [illegible] Github [illegible] MSYS2 [illegible] Unix [illegible] ESP-IDF [illegible] LF[illegible]Unix [illegible]

[illegible] commit [illegible] LF [illegible] MSYS2 [illegible] Unix [illegible] Unix[illegible] IDF [illegible] checkout [illegible]

```
git rebase --exec 'git diff-tree --no-commit-id --name-only -r HEAD | xargs dos2unix &
↪& git commit -a --amend --no-edit --allow-empty' master
```

[illegible] master [illegible]

[illegible]

```
dos2unix FILENAME
```

[illegible]

```
git commit --amend
```

7. [illegible]

[illegible] astyle [illegible]

[illegible]

[illegible]

```
tools/format.sh components/my_component/file.c
```

CMake [illegible]

- [illegible] 4 [illegible]
- [illegible] 120 [illegible]/[illegible]
- [illegible] endforeach()[illegible]endif() [illegible]
- [illegible](with_underscores)[illegible]
- [illegible](with_underscores)[illegible]
- [illegible](WITH_UNDERSCORES)[illegible]
- [illegible] cmake-lint [illegible]