WT0132P4-A1
Getting Started Guide for AWS IoT Core

Table of Contents

1 Document information

1.1 Document revision history

Date	Modified by	Description
November 28, 2024	Vans	First release

1.2 Applicable operating systems for this guide

This guide is applicable to the ESP32-P4 chips with FreeRTOS system.

2 Overview

WT0132P4-A1, an integrated NOR FLASH small core board based on ESPRESSIF ESP32-P4 chip designed by Wireless-Tag Technology Co., Limited. The core processor chip, ESP32-P4, can be stacked with 16MB or 32MB PSRAM in the package, and contains a high-performance (HP) system and a low-power (LP) system; the HP system adopts a RISC-V dual-core processor with a main frequency up to 400MHz, and contains a JPEG encoder/decoder, pixel-processing gas pedal, H.264 video encoder, and a MIPI interface; it has powerful image and voice processing capabilities.

3 Hardware description

3.1 Datasheet

The link to the product datasheet: https://en.wireless-tag.com/product-item-56.html.

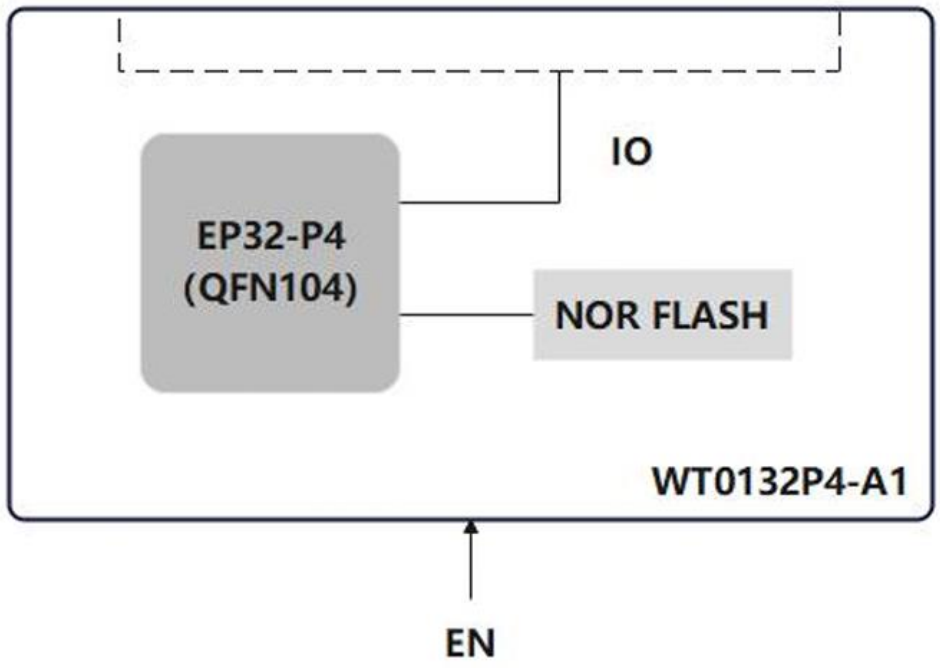
3.2 Standard kit contents

The contents of the standard shipping hardware package as indicated below:

- Chip: ESP32-P4, 16MB Flash
- Memory: 16MB or 32MB PSRAM
- Operating ambient temperature: -40-85°C
- Processing Interfaces: JPEG Codec, PPA, ISP, H264 Encoder
- MIPI CSI-2, MIPI DSI
- 100Mbps Ethernet

Please links to the page on our company website for more detail:
https://en.wireless-tag.com/product-item-56.html.

3.3 Additional hardware references



4 Set up your development environment

4.1 Tools installation (IDEs, Toolchains, SDKs)

.

1. IDE based
   - Eclipse Plugin
   - VSCode Extension
   - Arduino

2. idf.py

The idf.py command-line tool provides a front-end for easily managing your project builds, deployment and debugging, and more. It manages several tools, for example:

- CMake, which configures the project to be built.
- Ninja, which builds the project.
- Esptool.py, which flashes the target.

Can read more about configuring the build system using idf.py here.

3. ESP-IDF

For the manual procedure, please select according to your operating system.

- Windows Installer
- Linux and macOS

4. Optimizing the compiler

In ESP-IDF, you can force compiler optimization by modifying the compiler options. You can set the compiler optimization options in the CMakeLists.txt file.

In your project's CMakeLists.txt file, add the following lines to set the compiler optimization options

```
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -O3")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -O3")
```

5. SDK

Esp-idf , A tutorial on setting up the sdk environment can be found in the Getting Started guide. ESP-IDF is Espressif's official IoT Development Framework for the ESP32, ESP32-S, ESP32-C and ESP32-H series of SoCs.

It provides a self-sufficient SDK for any generic application development on these platforms, using programming languages such as C and C++.

4.2 Additional software references

ESP32P4 Chip Manual : Provides specifications for the ESP32P4 chip.

ESP-IDF Programming Guide for ESP32P4: Extensive documentation for the ESP-IDF development framework.

5 Set up device hardware

5.1 Product Images

5.2 Pin Description

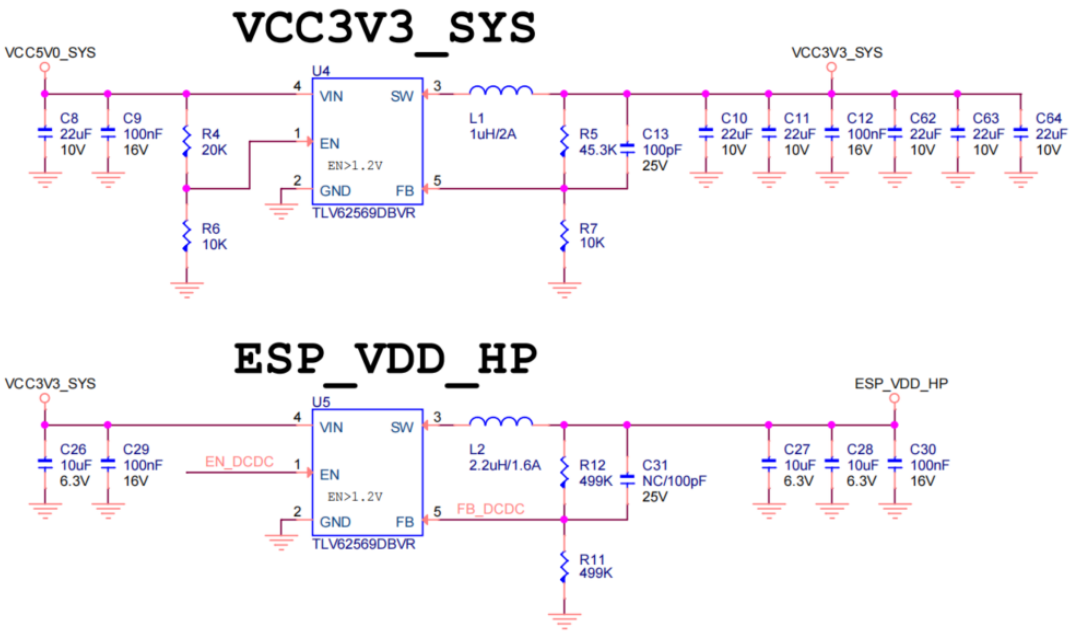
Pin	Name	Description
1	GND	GROUND
2	DSI_DATAP1	MIPI DSI PHY DATAP1
3	DSI_DATAN1	MIPI DSI PHY DATAN1
4	DSI_CLKN	MIPI DSI PHY CLKN
5	DSI_CLKP	MIPI DSI PHY CLKP
6	DSI_DATAP0	MIPI DSI PHY DATAP0
7	DSI_DATAN0	MIPI DSI PHY DATAN0
8	GND	GROUND
9	CSI_DATAN0	MIPI CSI PHY DATAN0
10	CSI_DATAP0	MIPI CSI PHY DATAP0
11	CSI_CLKP	MIPI CSI PHY CLKP
12	CSI_CLKN	MIPI CSI PHY CLKN
13	CSI_DATAN1	MIPI CSI PHY DATAN1
14	CSI_DATAP1	MIPI CSI PHY DATAP1
15	GND	GROUND
16	USB_DM	USB2 OTG PHY DM
17	USB_DP	USB2 OTG PHY DP
18	GND	GROUND
19	GPIO24	GPIO24, USB1P1 N0
20	GPIO25	GPIO25, USB1P1 P0
21	GND	GROUND
22	GPIO26	GPIO26, USB1P1 N1
23	GPIO27	GPIO27, USB1P1 P1

24	GPIO28	GPIO28, GPSPI SPI2 CS, EMAC PHY RXDV, DBG_PSRAM_D
25	GPIO29	GPIO29,GPSPI SPI2 D, EMAC_PHY_RXDO, DBG_PSRAM_Q
26	GPIO30	GPIO30, GPSPI SPI2 CK, EMAC_PHY_RXD1, DBG_PSRAM_WP
27	GPIO31	GPIO31, GPSPI SPI2 Q, EMAC_PHY_RXER, DBG_PSRAM_HOLD
28	GPIO32	GPIO32, I3C MST SCL, GPSPI SPI2 HOLD EMAC RMII CLK, DBG_PSRAM_DQ4
29	GPIO33	GPIO33, I3CMST_SDA, GPSPI SPI2 WP EMAC PHY TXEN, DBG_PSRAM_DQ5
30	GPIO34	GPIO34, GPSPI SPI2 IO4, EMAC PHY TXDO, DBG_PSRAM_DQ6
31	GPIO35	GPIO35,GPSPI SPI2 IO5, EMAC PHY TXD1, DBG_PSRAM_DQ7 (IO35 pulls down into download mode)
32	GND	GROUND
33	GPIO36	GPIO36, GPSPI SPI2 IO6, EMAC PHY TXER, DBG_PSRAM_DQSO (Default IO35,36 pull-up to enter SPI Boot mode)
34	GPIO37	GPIO37,UART0_TXD,GPSPI SPI2 IO7
35	GPIO38	GPIO38,UART0 RXD,GPSPI SPI2 DQS
36	GPIO39	GPIO39, SD1_CDATA0_PAD, REF_50M_CLK_PAD
37	GPIO40	GPIO40, SD1_CDATA1_PAD, GMAC_PHY_TXEN_PAD
38	GPIO41	GPIO41, SD1_CDATA2_PAD, GMAC_PHY_TXDO_PAD
39	GPIO 42	GPIO42, SD1_CDATA3_PAD, GMAC_PHY_TXD1_PAD
40	GPIO43	GPIO43, SD1_CCLK_PAD, GMAC_PHY_TXER_PAD
41	VCC	POWER
42	GND	GROUND
43	GPIO44	GPIO44, SD1_CCMD_PAD, GMAC_RMII_CLK_PAD
44	GPIO45	GPIO45, SD1_CDATA4_PAD, GMAC_PHY_RXDV_PAD
45	GPIO46	GPIO46, SD1_CDATA5_PAD, GMAC_PHY_RXDO_PAD
46	GPIO47	GPIO47, SD1_CDATA6_PAD, GMAC_PHY_RXD1_PAD
47	GPIO48	GPIO48, SD1_CDATA7_PAD, GMAC_PHY_RXER_AD
48	GPIO49	GPIO49, GMAC_ PHY_TXEN_PAD, ADC2_CHANNEL2
49	GPIO50	GPIO50, GMAC_RMII_CLK_PAD, ADC2_CHANNEL3
50	GPIO51	GPIO51, GMAC_PHY_RXDV_PAD, ADC2_CHANNEL4, ANA_COMP0
51	GPIO52	GPIO52, GMAC_PHY _RXD0_PAD, ADC2_CHANNEL5, ANA_COMP0
52	GPIO53	GPIO53, GMAC_PHY_RXD1_PAD, ADC2_CHANNEL6, ANA_COMP1
53	GND	GROUND
54	GPIO54	GPIO54, GMAC_PHY_RXER_PAD, ADC2_CHANNEL7, ANA_COMP1
55	GPIO2	GPIO2, MTCK, LP_GPIO2, TOUCH_CHANNEL0
56	GPIO3	GPIO3, MTDI, LP_GPIO3, TOUCH_CHANNEL1
57	GPIO4	GPIO4, MTMS, LP_GPIO4, TOUCH_CHANNEL2
58	GPIO5	GPIO5, MTDO, LP_GPIO5, TOUCH_CHANNEL3
59	GPIO6	GPIO6, SPI2_HOLD_PAD, LP_GPIO6, TOUCH_CHANNEL4
60	GPIO7	GPIO7, SPI2_CS_PAD, LP_GPIO7, TOUCH_CHANNEL5
61	GPIO8	GPIO8, UART0_RTS_PAD, SPI2_D_PAD，LP_GPIO8, TOUCH_CHANNEL6
62	GPIO9	GPIO9, UART0_CTS_PAD, SPI2_CK_PAD, LP_GPIO9, TOUCH_CHANNEL7
63	GPIO10	GPIO10, UART1_TXD_PAD, SPI2_Q_PAD, LP_GPIO10, TOUCH_CHANNEL8

64	GND	GROUND
65	GPIO11	GPIO11, UART1_RXD_PAD, SPI2_WP_PAD, LP_GPIO11, TOUCH_CHANNEL9
66	GPIO12	GPIO12, UART1_RTS_PAD, LP_GPIO12, TOUCH_CHANNEL10
67	GPIO13	GPIO13, UART1_CTS_PAD, LP_GPIO13, TOUCH_CHANNEL11
68	GPIO14	GPIO14, LP_GPIO14, LP_UART_TXD_PAD, TOUCH_CHANNEL12
69	GPIO15	GPIO15, LP_GPIO15, LP_UART_RXD_PAD, TOUCH_CHANNEL13
70	CHIP PU	Enable P4 chip (internal 10K pull-up)
71	GPIO0	GPIO0, LP_GPIO0, XTAL_32K_N
72	GPIO1	GPIO1, LP_GPIO1, XTAL_32K_P
73	GND	GROUND
74	GPIO16	GPIO16, ADC1_CHANNEL0
75	GPIO17	GPIO17, ADC1_CHANNEL1
76	GPIO18	GPIO18, ADC1_CHANNEL2
77	GPIO19	GPIO19, ADC1_CHANNEL3
78	GPIO20	GPIO20, ADC1_CHANNEL4
79	GPIO21	GPIO21, ADC1_CHANNEL5
80	GPIO22	GPIO22, ADC1_CHANNEL6
81	GPIO23	GPIO23, ADC1_CHANNEL7, REF_50M_CLK_PAD

table 1 IO Descriptions

5.3 VCC design instructions



6 Setup your AWS account and permissions

If you do not have an existing AWS account and user, refer to the online AWS documentation at Set up your AWS Account. To get started, follow the steps outlined in the sections below:

- Sign up for an AWS account
- Create an administrative user
- Open the AWS IoT console

Pay special attention to the Notes.

7 Create resources in AWS IoT

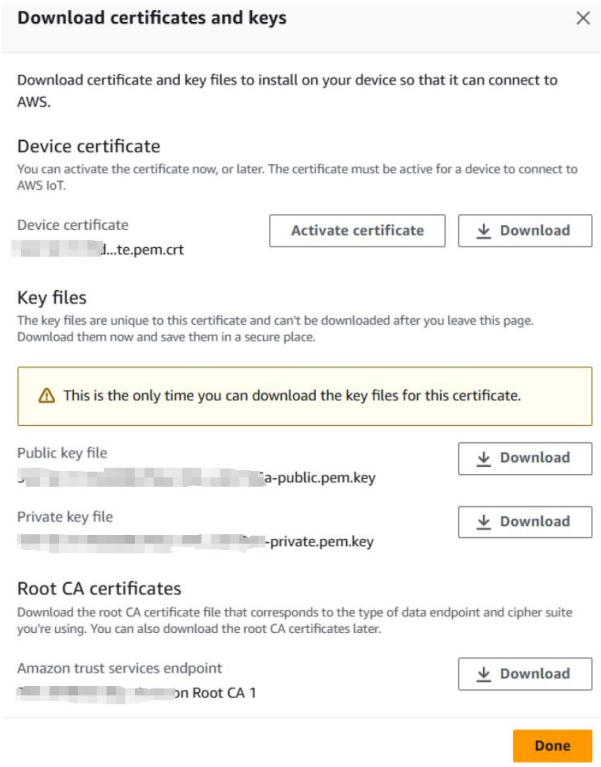
Refer to the online AWS documentation at Create AWS IoT Resources. Follow the steps outlined in these sections to provision resources for your device:

- Create an AWS IoT Policy
- Create a thing object

Pay special attention to the Notes.

8 Provision the device with credentials

During “Create a thing object”, you will encounter the requirement to download the certificate in the last step, as shown below, keep it, which is the corresponding server certificate of the device.



9 Build the demo

This tutorial uses the esp-idf/example/protocol/mqtt/ssl_mutual_auth example to test the device's connection to AWS-IoT-Core.

9.1 Engineering Configuration

1. After entering the project, you need to replace the three certificates in the main directory. The certificates to be replaced are stored in the connection toolkit you downloaded earlier. The replacement corresponds to the following:
   - The .client.crt file is the client certificate; use the .pem.crt file instead.
   - The .client.key file is the client key; use the .private.pem.key file instead.
   - The .mosquitto.org.crt file is the server-side secret key; use the CA1.pem file instead.
2. Replace the link to the mqtt server accessed by the project and add the client_id configuration entry.The link is replaced with the link used when connecting to the device, and the client_id used "basicPubSub".
   Note: The link needs to be prefixed with mqtt://
3. Activate the IDF environment, configure the chip as ESP32P4 and modify the WiFi configuration information of the project via menuconfig.
   Note: Configuration path for WiFi configuration: Connection Configuration Example ->WiFi SSID / WiFi Password

10 Run the demo

Take the project introduced in the previous chapter, burn it into your device, and open the serial port debugging assistant.

11 Verify messages in AWS IoT Core

Open "MQTT test cilent", set the Subscribe topic to "sdk/test/python" and click "Subscribe". button.

12 Troubleshooting

If the connection fails when connecting to mqtt, it is recommended to check whether the above steps have been completed, whether the client_id and uri have been correctly modified, and whether the certificate has been correctly replaced.
If there is something you don't understand, you can also refer here.