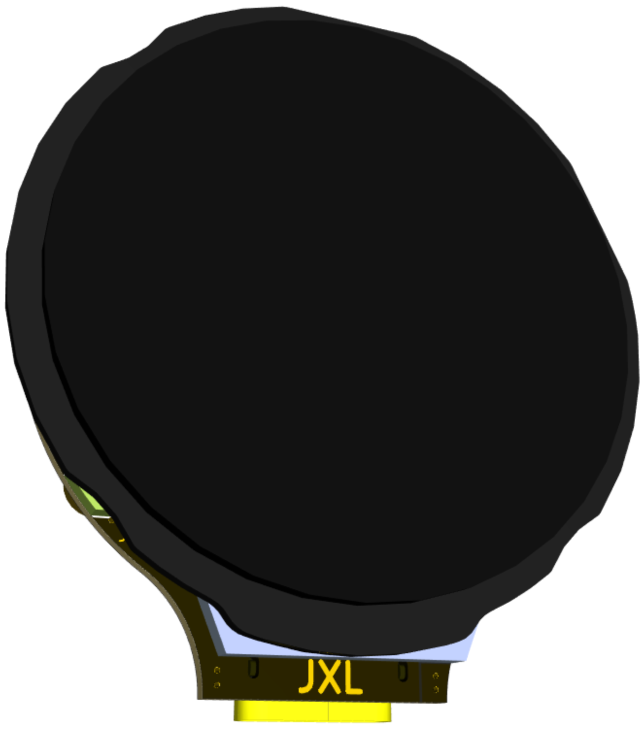
1.28-inch ESP32S3 round TFT-I80 screen development board

Development Reference Documentation



Time: 2025/04/15

Version V1.0

Revision History

date	Version	Release Notes
2025-04-15	V1.0	-First release

1. Document Description

The main body of the development board isESP32-S3As the core, and with a variety of peripherals and power supply

Road design, the development board can be opened by long pressing the middle button3SaccomplishPower on/off, this document expands

The functions of each section of the board are analyzed to facilitate user development.

The document disassembles and analyzes the development board according to the following aspects. Note that the code part needs

To cooperate with data engineeringFullFunctionTestRead and learn together.

ESP32-S3 Core Parameters

The core of ESP32-S3 isESP32-S3Chip, externalFLASH & PSRAM,by

and download modeControl buttonsandantennaIt consists of several major parts.

The operating frequency and basic function parameters of the ESP32-S3 chip will not be described in detail here. Please refer to the official

Chip manual.

This development board has external plug-ins16MBofFLASH,useFour-wire SPIto connect.

The PSRAM part is built into the ESP32-S3 chip and is shipped with8MBand16MBTwo versions

For users to choose,PSRAM internalOPIConnection method, be sure to selectOPI PSRAM,

Otherwise it will not be recognized.

The only development boardUSB portConnected to the chipIO19 D-andIO20 D+Two references

Feet are used forDownload and debug, so when downloading, please selectUSB Download, instead of UART download,

For users who want to use the USB port for serial printing, turn on USB CDC On Boot

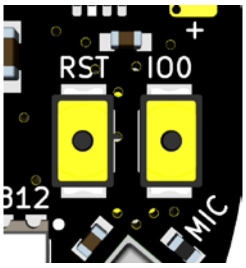
The function can use syntax such as Serial.print to print debugging information.

For some users, the code option is set to turn off USB when burning.

After that, the USB port will no longer be found, and the computer will no longer recognize the USB port, and the

USB download function can detect USB port but cannot download. In this case, you need to use

The two onboard micro physical buttons enter the download mode, as shown below



Press and holdIO0Without releasing the button, press it againRSTThen release the buttonRSTPress the button and then release it

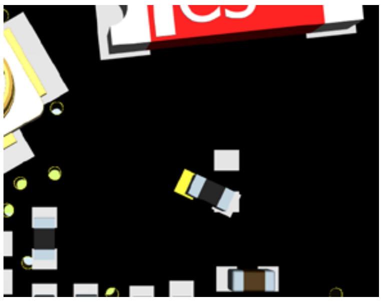
openIO0Press the button, ESP32-S3 will enterDownload ModeIt is worth noting that the IO0 button here

It is not recommended to use the key as a regular physical key to realize the function, because IO0 has been used as a touch reset

use.

The antenna uses the onboard antenna by default, and the reserved IPEX needs to change the resistor switch.

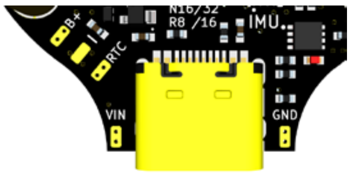
For the IPEX antenna, you need to move the 0Ω resistor at the bottom to the other side for soldering.



3. Power Supply & Power On/Off

The power supply section provides externalVIN inputpads,USB,as well asBAT batteryThree power supplies

Way.



inVIN padandUSB power supplyDirect connection, please use with caution, do not use voltage

Exceed5.5V, and both can charge the battery.

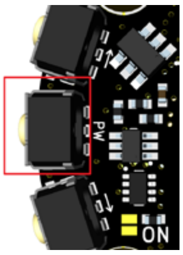
USB power supply and battery power supply use two ideal diodes to realize UPS switching function.

The flow then passes throughPower control switchPower supply toTPS63802 Buck-BoostChip, voltage regulator3.3Vlose

Output for ESP32-S3 and other boards.

The power control switch is controlled by the middle button.Long press for 3 secondsThe above startup or shutdown,

Therefore, it can be used as a normal button at ordinary times.

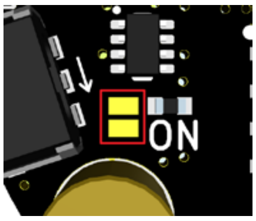


After shutdown, only the current consumption of the switch control part and the battery charging part is measured.

State current7ua.

If you need to power on instead of controlling by buttons, you can short-circuit the following figure

Short-circuit the two solder points shown in the figure and then power on to turn on the device.



4. Lithium Battery Charging & Power Detection

Battery can be selected3.7V rechargeable lithium batteryThe charging part uses the BQ25170 chip

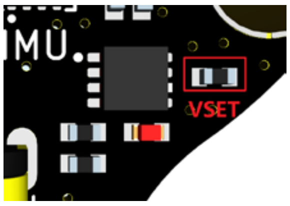
Charge the battery.

BQ25170 supports the following batteries

- Lithium-ion battery: 4.05V, 4.1V,4.2V, 4.35V, 4.4V
- Lithium iron phosphate battery: 3.5V, 3.6V, 3.7V

The default setting charging voltage is4.2V, suitable for use with lithium batteries calibrated at 3.7V

According to the chip manual, adjust by modifying the resistor at the bottom



The corresponding relationship is as follows

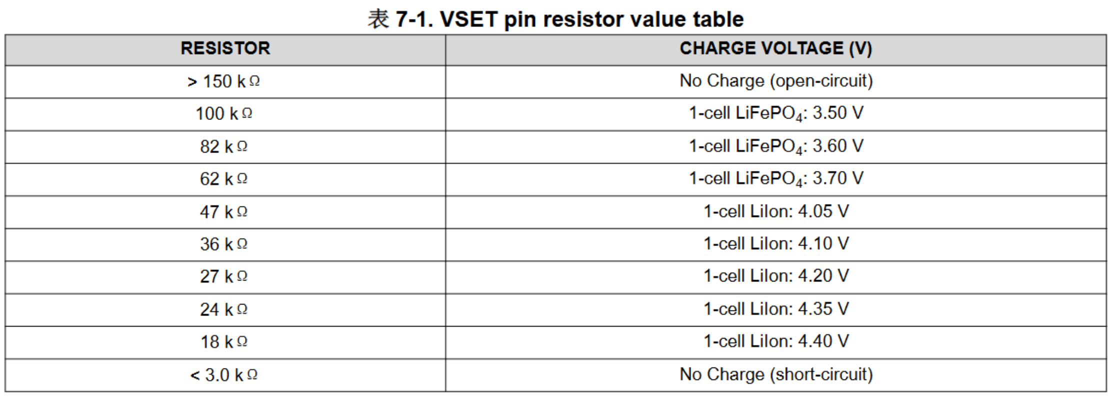
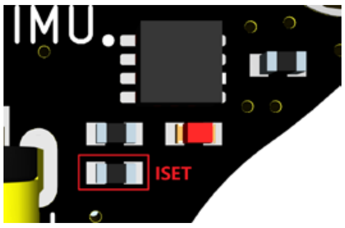


表 7-1. VSET pin resistor value table

RESISTOR	CHARGE VOLTAGE (V)
> 150 kΩ	No Charge (open-circuit)
100 kΩ	1-cell LiFePO$_4$: 3.50 V
82 kΩ	1-cell LiFePO$_4$: 3.60 V
62 kΩ	1-cell LiFePO$_4$: 3.70 V
47 kΩ	1-cell LiIon: 4.05 V
36 kΩ	1-cell LiIon: 4.10 V
27 kΩ	1-cell LiIon: 4.20 V
24 kΩ	1-cell LiIon: 4.35 V
18 kΩ	1-cell LiIon: 4.40 V
< 3.0 kΩ	No Charge (short-circuit)

The maximum current is 800ma, which is adjustable and the default current setting is the maximum600mA



The corresponding relationship is as follows $I_{CHG} = K_{ISET}/R_{ISET}$

in I_{CHG} The charging current that needs to be set K_{ISET} is the gain factor, with a typical value of 270-330

The value is 300, from which the resistance required for the corresponding charging current can be calculated R_{ISET},default499

Ω.

When there is no battery connected, the charging indicator light will be off when connected to the USB port or VIN power supply.

In the flashing state, it will light up when the battery is connected and charging, and go out when it is fully charged.

The chip does not start charging when the voltage drops below 4.2V, but it has a threshold value below which it starts charging.

This is a normal phenomenon. Please refer to the chip manual in the materials for details.

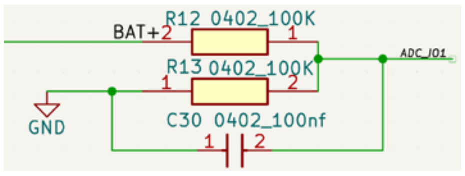
It is worth noting thatTPS63802 Buck-Boostchip, can be as low as2V or lessVoltage

Working under the condition of no onboard battery over discharge circuit, the battery selection is best to use

Battery with protection board model.

The main method of power detection is to detect the battery voltage. The battery voltage is detected by two100Kelectricity

Connect to ESP32-S3 after voltage divisionIO1Port, refer to the figure below



When the battery is not connected, the voltage will befloatStatus, the above figure is a reference circuit and will not increase the development

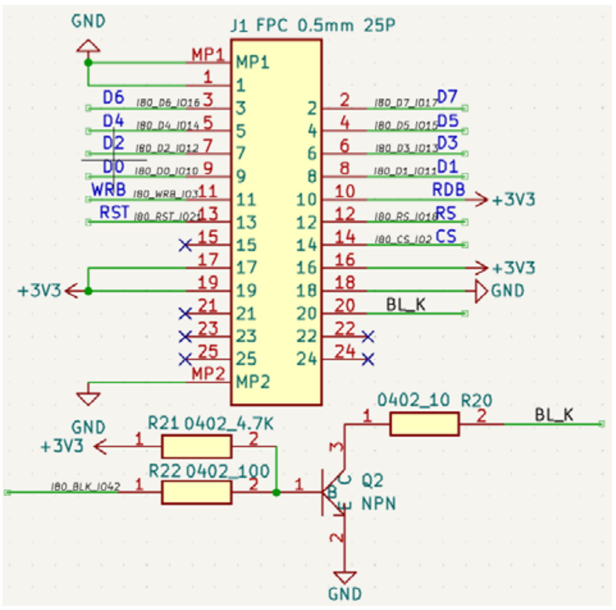
Board static power consumption, static power consumption will still be7uaabout.

5. Screen & Capacitive Touch

The screen is a circular display area.1.28inResolution240x240IPS color display

The screen is driven byGC9A01, the hardware driver interface isI80 interface, some documents are called MCU

For the connection between ESP32-S3 and ESP32-S3, please refer to the figure below.



D0-D7	Data pin	IO10-17
RST	Reset pin	IO21
WRB	Write Enable	IO3
RS	Data/Command	IO18
CS	Chip Select	IO2
BLK	Backlight control	IO42

The provided test code usesArduino_GFX_Library The library is driven by

There are multiple versions of this library, and they are not fully compatible. It is recommended to use the packaged version in the document.

1.4.7After decompression, add it to the path where the Arduino IDE stores the library.

Too much elaboration

In the sample code

```
#include <Arduino_GFX_Library.h> //Add library file reference
```

Initialize it with the following code

```
#define GFX_BL 42
#define BL_Freq 5000
unsignedintBL_Brightness = 255;
Arduino_DataBus *bus =newArduino_ESP32LCD8(18/* DC */, 2/* CS */, 3/* WR */, -1/* RD */,10/* D0 */, 11/* D1 */, 12/* D2 */, 13
/* D3 */, 14/* D4 */, 15/* D5 */, 16/* D6 */, 17/* D7 */);
Arduino_GFX *gfx =newArduino_GC9A01(bus, 21/* RST */, 0/* rotation */,true/* IPS */);
```

```
gfx->begin();//Initialize LCD
gfx->fillScreen(BLACK);//Background color Black
```

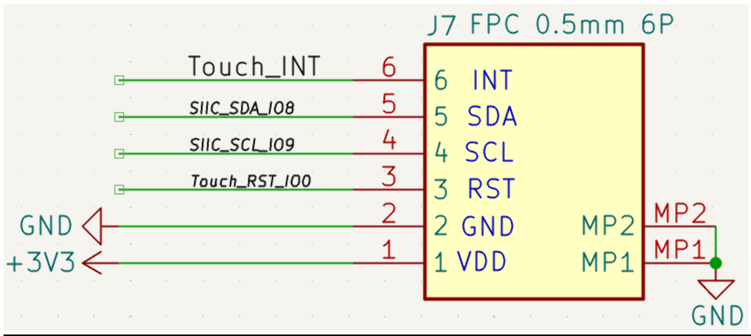
You can then callARDUINO GFX LibraryDisplay driver provided by the function

Please refer to gfx->fillScreen(BLACK);For functions such as setting the background color to black, please refer to the library author's

github:https://github.com/moononournation/Arduino_GFX

Capacitive touch useCST816Touch dedicated chip, usingIICDrive, with reset and center

Break pin, IIC device address0x15The hardware connection with ESP32-S3 is shown in the figure below.



SDA	Data pin	IO8
SCL	Clock pin	io9
RST	Reset pin, low level active	IO0
INT	Interrupt pin	TCA6408-P0

Worth notingRSTandINTPin, RST pin is usedIO0Therefore, the IO0 button in the previous text has no

Do not use it as a normal button, otherwise it will affect the touch function.

The interrupt pin is connected toTCA6408Expansion chipP0Port, should detect when touch is detected

TCA6408Interrupt pinTriggered as an interrupt event and detected by IICP0Port

Status to determine whether there is a touch.

The pins used by IIC are defined at the bottom of the sample code.

```
//IIC
#define SCL 9
#define SDA 8
```

The following code defines the reset pin, IIC device address0x15as well asID RegisterAddress and

The meaning of the value in the register

```
//CST816
#define TouchRST 0
#define TouchI2CAddr 0x15

#define ChipIdRegister 0xA7
#define CST716ChipId 0X20
#define CST816SChipId 0XB4
#define CST816TChipId 0XB5
#define CST816DChipId 0XB6
#define CST826ChipId 0X11
#define CST830ChipId 0X12
#define CST836UChipId 0X13
```

The following code resets the touch chip and detects the chip ID to determine the specific model of the touch chip

```
pinMode(TouchRST, OUTPUT);
digitalWrite(TouchRST, LOW);
delay(10);
digitalWrite(TouchRST, HIGH);
delay(50);

Wire.beginTransmission(TouchI2CAddr);
Wire.write(ChipIdRegister);
Wire.endTransmission(false);
Wire.requestFrom(TouchI2CAddr, 1,true);
ChipID = Wire.read();
```

```
        Serial.printf("\r\nTouchChipID: 0x%02X",ChipID);
        if(ChipID == CST716ChipId) Serial.println(",Touch chip model :CST716");
        else if(ChipID == CST816SChipId) Serial.println(",Touch chip model:CST816S");
        else if(ChipID == CST816TChipId) Serial.println(",Touch chip model:CST816T");
        else if(ChipID == CST816DChipId) Serial.println(",Touch chip model:CST816D");
        else if(ChipID == CST826ChipId) Serial.println(",Touch chip model:CST826");
        else if(ChipID == CST830ChipId) Serial.println(",Touch chip model:CST830");
        else if(ChipID == CST836UChipId) Serial.println(",Touch chip model:CST836U");
        else Serial.println(",error!");
```

The following code enables TCA6408 interrupt event detection

```
pinMode(TCA6408Int,INPUT_PULLUP);
//Registering interrupt service function
attachInterrupt(digitalPinToInterrupt(TCA6408Int), TCA6408HandleInterrupt, FALLING);
```

Execution function after interrupt detection and touch detection function

```
voidTCA6408HandleInterrupt(void)
{
 TCA6408EventFlag =true;
 }
```

```
voidmy_AllInt()
{
 if(TCA6408EventFlag){
  intTCA6408IntValue = 0;
  Wire.beginTransmission(TCA6408I2CAddr);
  Wire.write(TCA6408InputPortReg);
  Wire.endTransmission(false);
  Wire.requestFrom(TCA6408I2CAddr, 1,true);
  TCA6408IntValue = Wire.read();

  if((TCA6408IntValue & 0x01) == 0x00) {TouchEventFlag =true; Serial.print("\r\nTouch Int");}
  if((TCA6408IntValue & 0x02) == 0x00) {Serial.print("\r\nIMU Int1");}
  if((TCA6408IntValue & 0x04) == 0x00) {Serial.print("\r\nIMU Int2");}
  if((TCA6408IntValue & 0x08) == 0x00) {Serial.print("\r\nSW UP Int");}
  if((TCA6408IntValue & 0x10) == 0x00) {Serial.print("\r\nSW PW Int");}
  if((TCA6408IntValue & 0x20) == 0x00) {Serial.print("\r\nSW Down Int");}
  if((TCA6408IntValue & 0x40) == 0x00) {Serial.print("\r\nCharging...");}
  if((TCA6408IntValue & 0x80) == 0x00) {Serial.print("\r\nRTC Int");}

  TCA6408EventFlag =false;
  }
 }
```

Coordinate detection function

```
voidmy_touch_read(lv_indev_drv_t * indev_drv, lv_indev_data_t * data){
  //Store the pressed coordinates and status
  lv_coord_t last_x = 0;
  lv_coord_t last_y = 0;
  unsignedintX_H4 = 0;
  unsignedintX_L8 = 0;
  unsignedintY_H4 = 0;
  unsignedintY_L8 = 0;
  //True if there is touch, false otherwise
  if(TouchEventFlag) {
    Wire.beginTransmission(TouchI2CAddr);
    Wire.write(0x03);
    Wire.endTransmission(false);
    //Wire.beginTransmission(TouchI2CAddr);
    Wire.requestFrom(TouchI2CAddr, 1,true);
    X_H4 = Wire.read();

    Wire.beginTransmission(TouchI2CAddr);
    Wire.write(0x04);
    Wire.endTransmission(false);
    //Wire.beginTransmission(TouchI2CAddr);
    Wire.requestFrom(TouchI2CAddr, 1,true);
    X_L8 = Wire.read();

    Wire.beginTransmission(TouchI2CAddr);
    Wire.write(0x05);
    Wire.endTransmission(false);
    //Wire.beginTransmission(TouchI2CAddr);
    Wire.requestFrom(TouchI2CAddr, 1,true);
    Y_H4 = Wire.read();

    Wire.beginTransmission(TouchI2CAddr);
    Wire.write(0x06);
    Wire.endTransmission(false);
    //Wire.beginTransmission(TouchI2CAddr);
    Wire.requestFrom(TouchI2CAddr, 1,true);
    Y_L8 = Wire.read();
```

```
      last_x = 0xFF - (X_H4 << 8 | X_L8)&0X0FFF;
      last_y = (Y_H4 << 8 | Y_L8)&0X0FFF;
      data->point.x = last_x;
      data->point.y = last_y;

      TouchEventFlag =false;
      //Serial.printf("Touch Point:%02X , %02X \r\n",last_x,last_y);
      data->state = LV_INDEV_STATE_PR;
    }
    else{
      data->state = LV_INDEV_STATE_REL;
    }
  }
```

NoticeThe above code needs to be used with the complete code, not copied directly, for

Learning the role of shared progress for code analysis

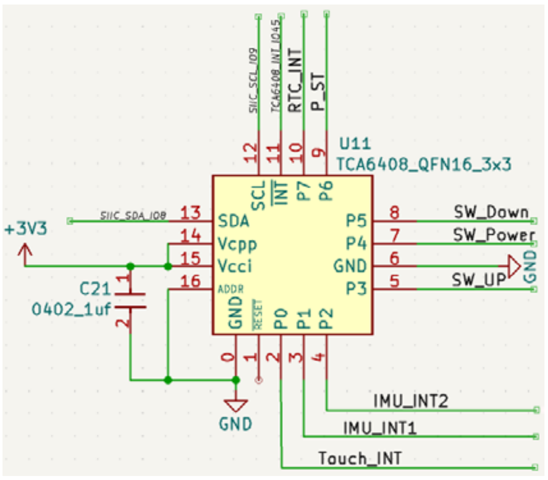
6. TCA6408 Expansion IO

The TCA6408 expansion chip can beIICCommunication extension8Supports input detection and output

IO port with output function and interrupt trigger function. All 8 IOs on the development board are used as

Input DetectionFunction, IIC device address0x20, and ESP32-S3 and the functions of each port

Refer to the table or figure below.



SCL	Clock pin	io9
SDA	Data pin	IO8

INT	Interrupt pin, low level pulse after triggering	IO45
P0	Screen touch interrupted, pulled up	- -
P1	QMI8658-IMU interrupt 1, pulled up	- -
P2	QMI8658-IMU interrupt 1, pulled up	- -
P3	Upper side button, pulled up	- -
P4	Middle button on the side, pulled up	- -
P5	The button at the bottom of the side has been pulled up	- -
P6	USB plug-in/charge detection, pulled up	- -
P7	RTC clock interrupt, pulled up	- -

The TCA6408 chip is controlled by 4 registers as shown below

register	Register address	illustrate
Input Port	0x00	Used to read port level
Output Port	0x01	Control port output level
Polarity reversal	0x02	Setting 1 Output level inversion
Configuration	0x03	1 High impedance input enabled 0 Output Enable

On the development board, we only need to use the input port register to read the status.match

Set registerThe default value is 1, which means the input detection status can be set or not.

P0-P7The input devices/ports connected to the ports have pull-up resistors and are high-voltage by default.

Ping, read as0xFF, when triggered, it is low level, and INT will change from high level to low

Level output until the interrupt is cleared

The basic driving idea is as follows

Registering interruption events

Interrupt event trigger

Query port status

Determine the specific trigger event

The following code snippet defines the interrupt/IIC pin, the IIC device ground of TCA6408.

Address, as well as register and flag related information

```
//IIC
#define SCL 9
#define SDA 8
```

```
//TCA6408
#define TCA6408Int 45
#define TCA6408I2CAddr 0x20

#define TCA6408ConfigurationReg 0x03
#defineTCA6408ConfigurationData 0xFF
#define TCA6408InputPortReg 0x00

Ticker TCA6408InterruptTicker;
volatile boolTCA6408EventFlag =false;
volatile boolTouchEventFlag =false;
```

The following code snippet is to initialize the TCA6408 content

```
voidTCA6408Init(){
 intTCA6408TempData = 0;

 Wire.beginTransmission(TCA6408I2CAddr);
 Wire.write(TCA6408ConfigurationReg);
 Wire.write(0x55);
 Wire.endTransmission(true);
 delay(10);
 Wire.beginTransmission(TCA6408I2CAddr);
 Wire.write(TCA6408ConfigurationReg);
 Wire.endTransmission(false);
 Wire.requestFrom(TCA6408I2CAddr, 1,true);
 TCA6408TempData = Wire.read();

 if(0x55 == TCA6408TempData) Serial.print("\r\nTCA6408 pass!");
 elseSerial.print("\r\nTCA6408 fail!");
 delay(10);
 Wire.beginTransmission(TCA6408I2CAddr);
 Wire.write(TCA6408ConfigurationReg);
 Wire.write(TCA6408ConfigurationData);
 Wire.endTransmission(true);

 delay(10);
 }
```

Interrupt event registration

```
pinMode(TCA6408Int,INPUT_PULLUP);
//Registering interrupt service function
attachInterrupt(digitalPinToInterrupt(TCA6408Int), TCA6408HandleInterrupt, FALLING);
```

```
voidTCA6408HandleInterrupt(void)
{
 TCA6408EventFlag =true;
 }
```

Query trigger events

```
voidmy_AllInt()
{
 if(TCA6408EventFlag)
 {
  intTCA6408IntValue = 0;
  Wire.beginTransmission(TCA6408I2CAddr);
  Wire.write(TCA6408InputPortReg);
  Wire.endTransmission(false);
  Wire.requestFrom(TCA6408I2CAddr, 1,true);
  TCA6408IntValue = Wire.read();

  if((TCA6408IntValue & 0x01) == 0x00) {TouchEventFlag =true; Serial.print("\r\nTouch Int");}
  if((TCA6408IntValue & 0x02) == 0x00) {Serial.print("\r\nIMU Int1");}
  if((TCA6408IntValue & 0x04) == 0x00) {Serial.print("\r\nIMU Int2");}
  if((TCA6408IntValue & 0x08) == 0x00) {Serial.print("\r\nSW UP Int");}
  if((TCA6408IntValue & 0x10) == 0x00) {Serial.print("\r\nSW PW Int");}
  if((TCA6408IntValue & 0x20) == 0x00) {Serial.print("\r\nSW Down Int");}
  if((TCA6408IntValue & 0x40) == 0x00) {Serial.print("\r\nCharging...");}
  if((TCA6408IntValue & 0x80) == 0x00) {Serial.print("\r\nRTC Int");}

  TCA6408EventFlag =false;
  }
 }
```

Through the expanded IO port above, the following functions can be achieved through the TCA6408 chip:

Function

screen touchInterrupt trigger event

Inertial Measurement UnitInterrupt trigger event

3 physicalKey interruptTrigger Event

Is the USB plugged in or the battery inChargingStatus detection

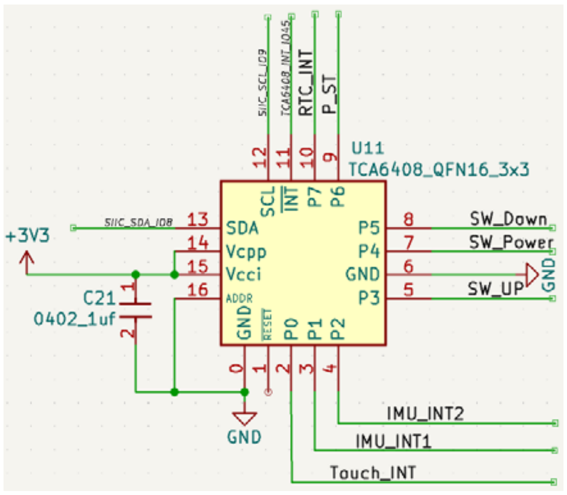
Interrupt triggering of RTC clock chip to achieve low power consumptionScheduled wake-upFunction

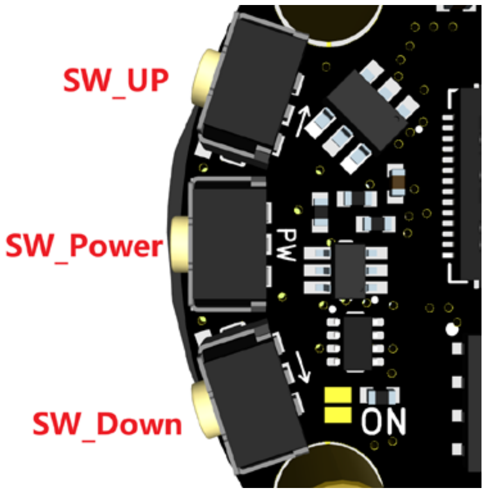
7. Physical buttons

The development board has5physical buttons, two of whichMode Controland reset is not a general

Use the buttons, the three side buttons are connected toTCA6408ofP3 P4 P5, key combination

The connection reference and position are shown in the figure





It is worth mentioning that the middle button is not only used as a normal button but also serves as the power on/off button.

The function of closingLong press for 3 secondsWill make the development boardPower onorShutdownOf course, once again,

Short-circuit the ON position pad below to disable the power on/off function.

For the usage of the three buttons, please refer to the previous TCA6408 expansion IO.

Because all three buttons are connected to TCA6408, the application development of 3 buttons

It is basically equivalent to the application of TCA6408.

8. QMI8658 Gyroscope

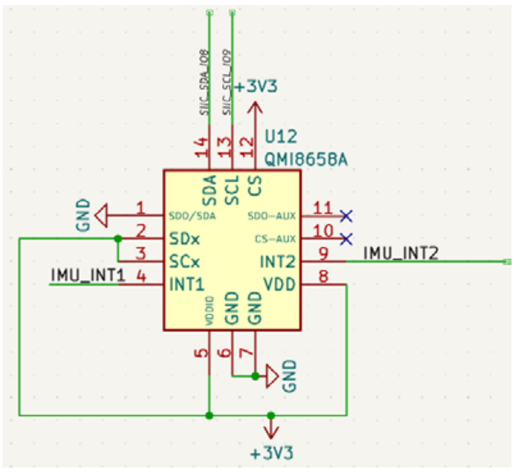
QMI8658It is a six-axis inertial measurement unit chip with three axesgyroscope, three-axisadd

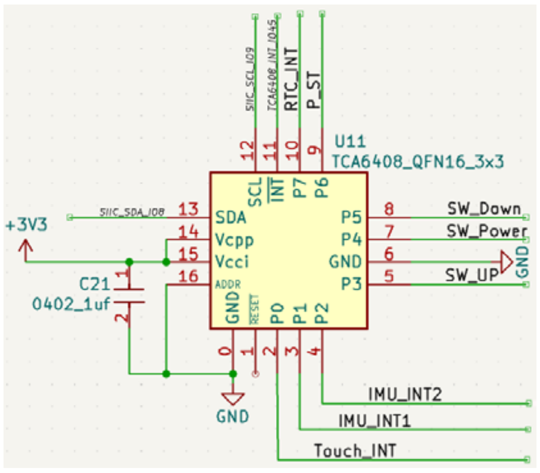
speedometer, enough for posture and motion state detection, throughIICESP32-S3

Line communication, with two interrupt pins, corresponding to the interrupt of gyroscope and accelerometer respectively

For user use.

Its IIC device address is0x6B, please refer to the following figure or table for connections on the development board





SCL	Clock pin	io9
SDA	Data pin	IO8
IMU_INT1	Interrupt pin 1	Connect to TCA6408 – P1
IMU_INT2	Interrupt pin 2	Connect to TCA6408 – P2

Driver code can refer to the engineering dataFullFunctionTestand chip manual

Read, users who are familiar with this type of chip can code the driver by themselves, or select the

The IMU part of this code is also transplanted from the Weixue example, saving

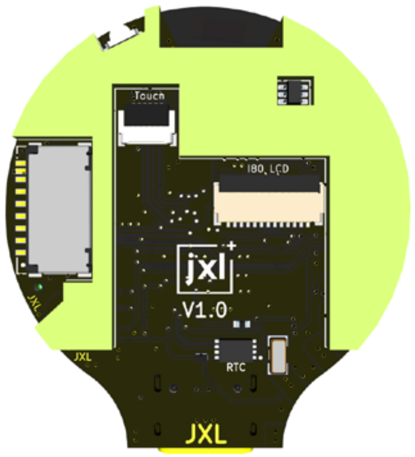
A lot of time, thanks to Weixue Electronics, and also suggested porting the verified code for use.

9. PCF85063 RTC Clock

The onboard RTC clock circuit is not on the same side as the ESP32-S3, so there is no need to

I was confused when I saw the related circuit on the development board.

The following figure shows the state when there is no screen on the back of the PCB.



PCF85063This RTC clock chip also usesIICCommunicate thanks to IIC

The characteristics of the bus, it shares a set of IIC pins with other IIC devices on the development board.

Used to set or read parameters such as time.

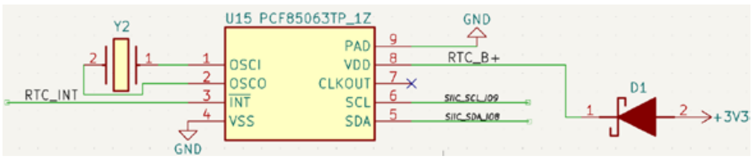
In addition, this chip hasINTpin, we connected it to the TCA6408 expansion

Chip ExhibitionP7Pin, INT pin settinghalf a minuteorOne minuteInterrupt, can be

ESP32-S3 enters low power state and wakes up through this interrupt state.

It can also be used for other interesting scenarios.

The connection between the chip and other components of the development board can refer to the following figure or table



SCL	Clock pin	io9
SDA	Data pin	IO8
INT	RTC interrupt pin	Connect to TCA6408 – P7

As for the power supply part, the power supply of RTC is obviously very important. There is an RTC battery on board.

The connection port is recommended.3VVoltagerechargeable batteries, when the RTC battery is not connected

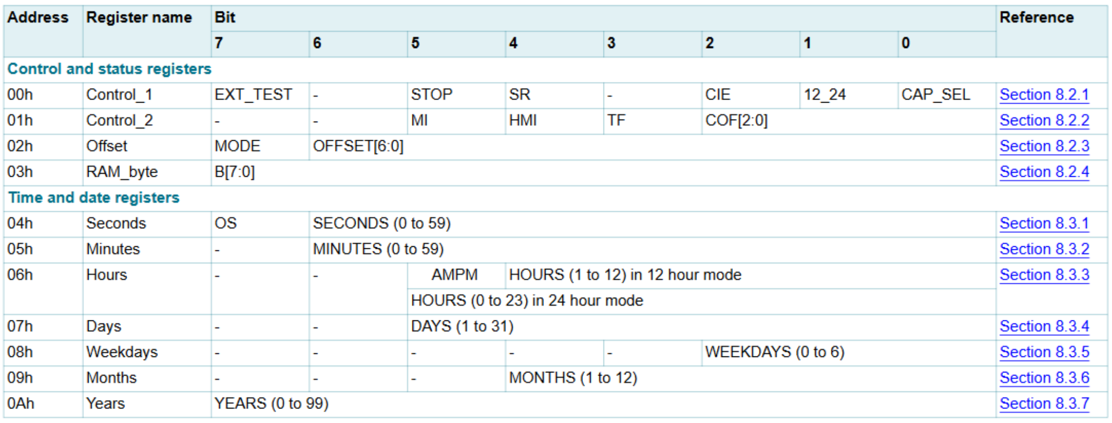
The chip gets its power from the onboard 3.3V voltage regulator. When the RTC battery is connected, the 3.3V will be

The RTC battery is charged.When the development board is turned off, the chip will be powered by the RTC battery.

For the quiescent current, please refer to the chip manual.

PCF85063 has 11 registers, each register has8The ones digit can be divided into two

Class, one isConfiguration parameters, the other isTime parameters.

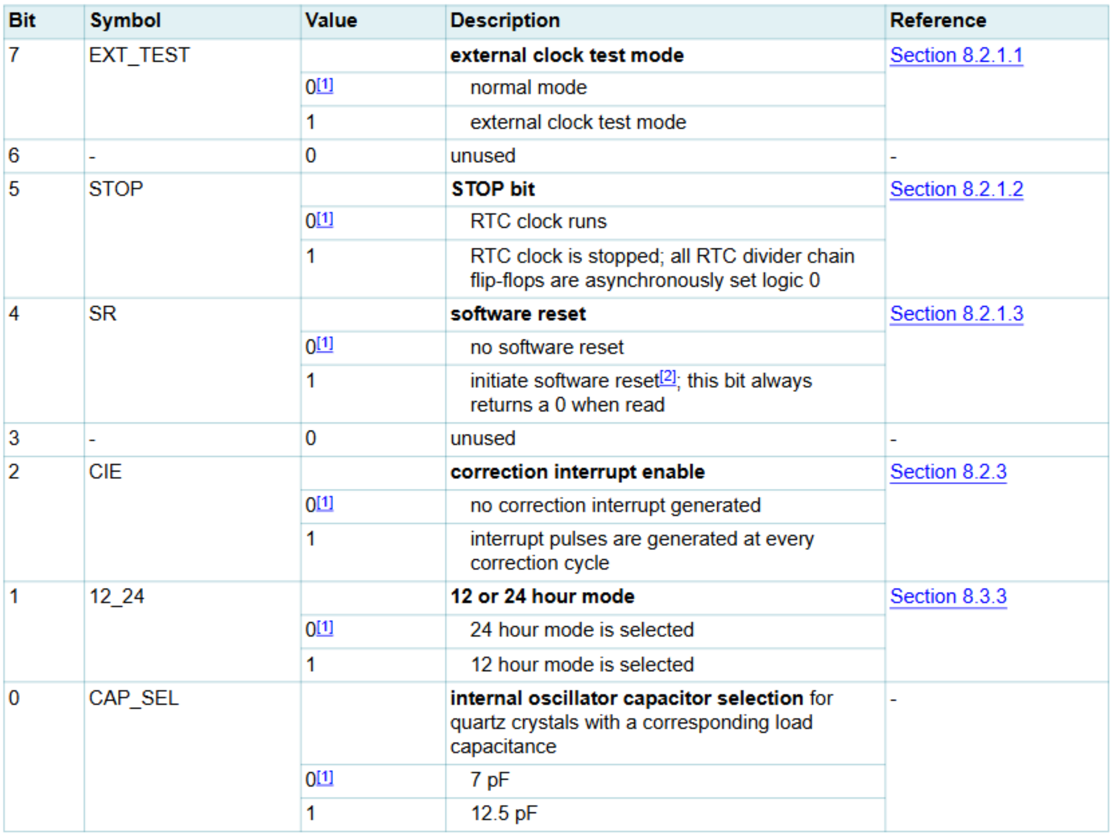


Address	Register name	Bit								Reference
		7	6	5	4	3	2	1	0	
Control and status registers										
00h	Control_1	EXT_TEST	-	STOP	SR	-	CIE	12_24	CAP_SEL	Section 8.2.1
01h	Control_2	-	-	MI	HMI	TF	COF[2:0]			Section 8.2.2
02h	Offset	MODE	OFFSET[6:0]							Section 8.2.3
03h	RAM_byte	B[7:0]								Section 8.2.4
Time and date registers										
04h	Seconds	OS	SECONDS (0 to 59)							Section 8.3.1
05h	Minutes	-	MINUTES (0 to 59)							Section 8.3.2
06h	Hours	-	-	AMPM	HOURS (1 to 12) in 12 hour mode					Section 8.3.3
				HOURS (0 to 23) in 24 hour mode						
07h	Days	-	-	DAYS (1 to 31)						Section 8.3.4
08h	Weekdays	-	-	-	-	-	WEEKDAYS (0 to 6)			Section 8.3.5
09h	Months	-	-	-	MONTHS (1 to 12)					Section 8.3.6
0Ah	Years	YEARS (0 to 99)								Section 8.3.7

0x00 – 0x03 registers store configuration parameters.

0x00Registers need attentionBit 1, used to set12 hoursor24 hoursMake, write

0 corresponds to 24-hour system, 1 corresponds to 12-hour system, and the default is 0.



Bit	Symbol	Value	Description	Reference
7	EXT_TEST		**external clock test mode**	Section 8.2.1.1
		0[1]	normal mode	
		1	external clock test mode	
6	-	0	unused	-
5	STOP		**STOP bit**	Section 8.2.1.2
		0[1]	RTC clock runs	
		1	RTC clock is stopped; all RTC divider chain flip-flops are asynchronously set logic 0	
4	SR		**software reset**	Section 8.2.1.3
		0[1]	no software reset	
		1	initiate software reset[2]; this bit always returns a 0 when read	
3	-	0	unused	-
2	CIE		**correction interrupt enable**	Section 8.2.3
		0[1]	no correction interrupt generated	
		1	interrupt pulses are generated at every correction cycle	
1	12_24		**12 or 24 hour mode**	Section 8.3.3
		0[1]	24 hour mode is selected	
		1	12 hour mode is selected	
0	CAP_SEL		**internal oscillator capacitor selection** for quartz crystals with a corresponding load capacitance	-
		0[1]	7 pF	
		1	12.5 pF	

0x01Registers are used to setMinute/half-minute interruptions, where digit 4 corresponds to half a minute

The default value is 0 to close. Bit 5 corresponds to minute interrupt, and the default value is 0 to close.

Bit	Symbol	Value	Description
7 to 6	-	00	unused
5	MI		**minute interrupt**
		0[1]	disabled
		1	enabled
4	HMI		**half minute interrupt**
		0[1]	disabled
		1	enabled
3	TF		**timer flag**
		0[1]	no timer interrupt generated
		1	flag set when timer interrupt generated
2 to 0	COF[2:0]	see Table 11	**CLKOUT control**

0X02Registers are used to setOffsetDuring operation, the chip will be negative due to the crystal oscillator accuracy.

It is difficult to achieve zero deviation due to factors such as load capacitance accuracy, aging, and temperature drift. This register is used to correct

Correct this deviation to achieve more accurate timekeeping.

Register bits 0 to 6 store the offset value, bit 7 - MODE is the offset mode setting,

Bit	Symbol	Value	Description
7	MODE		**offset mode**
		0[1]	normal mode: offset is made once every two hours
		1	course mode: offset is made every 4 minutes
6 to 0	OFFSET[6:0]	see Table 13	**offset value**

When MODE is set to 0, it is normal mode and the offset is adjusted every two hours.

The stored offset is 1 unit4.34ppm, that is to say, the specific offset needs to be

The value stored in 0-6*4.34, unit: ppm.

If MODE is set to 1, the chip will adjust the offset every 4 minutes.

The offset is 1 unit4.069ppm.

The values stored in bits 1 to 6 areTwo's ComplementThe value needs to be converted, please refer to the following table

Or see the chip manual for more detailed information.

OFFSET[6:0]	Offset value in decimal	Offset value in ppm	
		Normal mode MODE = 0	Fast mode MODE = 1
0111111	+63	+273.420	+256.347
0111110	+62	+269.080	+252.278
:	:	:	:
0000010	+2	+8.680	+8.138
0000001	+1	+4.340	+4.069
0000000[1]	0	0[1]	0[1]
1111111	–1	–4.340	–4.069
1111110	–2	–8.680	–8.138
:	:	:	:
1000001	–63	–273.420	–256.347
1000000	–64	–277.760	–260.416

0x03Registers are reserved for the chipIdle Bytes, there is an 8-bit space, which can

Used to store some information such as status data, which can be developed when connected to the RTC battery

The board saves data when it is powered off.

Bit	Symbol	Value	Description
7 to 0	B[7:0]	00000000[1] to 11111111	**RAM content**

Registers 0x04-0x0A are used to store time-related data. For more information, see Chip

The data sheet of the chip is as follows.

0x04Register storagesecondsdata.

Bit	Symbol	Value	Place value	Description
7	OS			**oscillator stop**
		0	-	clock integrity is guaranteed
		1[1]	-	clock integrity is not guaranteed; oscillator has stopped or has been interrupted
6 to 4	SECONDS	0[1] to 5	ten's place	**actual seconds** coded in BCD format, see Table 20
3 to 0		0[1] to 9	unit place	

0x05Register storageminutedata.

Bit	Symbol	Value	Place value	Description
7	-	0	-	unused
6 to 4	MINUTES	0[1] to 5	ten's place	**actual minutes** coded in BCD format
3 to 0		0[1] to 9	unit place	

0x06Register storageHourData is divided into 12-hour system and 24-hour system.

This data acquisition method sets the hour system in register 0x00.

Bit	Symbol	Value	Place value	Description
7 to 6	-	00	-	unused
12 hour mode[1]				
5	AMPM			**AM/PM indicator**
		0[2]	-	AM
		1	-	PM
4	HOURS	0[2] to 1	ten's place	**actual hours in 12 hour mode** coded in BCD format
3 to 0		0[2] to 9	unit place	
24 hour mode[1]				
5 to 4	HOURS	0[2] to 2	ten's place	**actual hours in 24 hour mode** coded in BCD format
3 to 0		0[2] to 9	unit place	

0x07Register storageDate/DayData, please note that the month and year are not set according to the

Instead of calculating the month as 30 days, it distinguishes between long and short months with 30/31 days, and includes automatic calculation for leap years.

Calculate the number of days in February.

Bit	Symbol	Value	Place value	Description
7 to 6	-	00	-	unused
5 to 4	DAYS[1]	0[2] to 3	ten's place	**actual day** coded in BCD format
3 to 0		0[3] to 9	unit place	

0x08Register storageWeek/weekData, note that data 0 is Sunday/day, and Saturday is not

Non-corresponding data 7

Bit	Symbol	Value	Description
7 to 3	-	00000	unused
2 to 0	WEEKDAYS	0 to 6	**actual weekday** values, see Table 25

0x09Register storagemoondata

Bit	Symbol	Value	Place value	Description
7 to 5	-	000	-	unused
4	MONTHS	0 to 1	ten's place	**actual month** coded in BCD format, see Table 27
3 to 0		0 to 9	unit place	

0x0ARegister storageYeardata

Bit	Symbol	Value	Place value	Description
7 to 4	YEARS	0[1] to 9	ten's place	**actual year** coded in BCD format
3 to 0		0[1] to 9	unit place	

For the use of RTC, please pay attention to setting the year and month information, even if it is not used, for the offset

The setting requires running a fixed time and comparing the deviation of the standard time to calculate the required

The offset value cannot be set directly without testing. For details, please refer to the chip

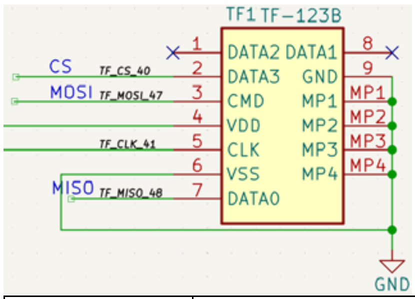
The manual explains it in more detail.

10. TF Card

TF card holder adoptsSPIThe connection method is to connect to ESP32-S3, so you need to use

Use TF card that supports SPI protocol. Most of the cards on the market support it, but some of them do not.

For TF cards that do not support the SPI protocol, please refer to the following figure or table for IO connection.



CS	SPI chip select pin	IO40
CLK	SPI data pin	IO41
MOSI	SPI data pin	IO47
MISO	SPI data pin	IO48

TF card can be used with #include <SD.h> library files andFile vFile;

To operate the file system, please refer to the section on playing TF card video in the sample code.

Initialize and determine whether the TF card exists

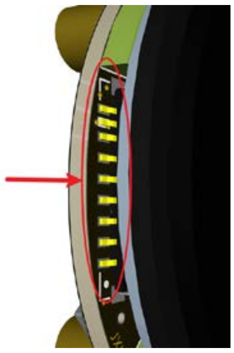
```
pinMode(SD_CS,INPUT_PULLUP);
delay(10);
SPI.begin(SCK, MISO, MOSI, SD_CS);
delay(10);
if(!SD.begin(SD_CS, SPI, 80000000))           //1-bit SD bus mode
```

Play Video

```
{
 uint8_t*mjpeg_buf=(uint8_t*)ps_malloc(MJPEG_BUFFER_SIZE);
 Start:
 vFile=SD.open(MJPEG_FILENAME);
 if(!vFile || vFile.isDirectory())
 {
  Serial.println(F("ERROR: Failed to open " MJPEG_FILENAME " file for reading"));
  vTaskDelayUntil(&lastWakeTime, pdMS_TO_TICKS(1000));
  esp_restart();
  }
 else
 {
  mjpeg.setup(&vFile, mjpeg_buf, displayBack, true, 0, 0, gfx->width()/*widthLimit*/, gfx->height());
  Serial.println(F("MJPEG video start"));
  while(vFile.available() && mjpeg.readMjpegBuf())
  {
   //Play video
   mjpeg.drawJpg();
   //Serial.printf("\r\nmainTask: %d", uxTaskGetStackHighWaterMark(NULL));
   vTaskDelayUntil(&lastWakeTime, pdMS_TO_TICKS(33));
   }
  vTaskDelayUntil(&lastWakeTime, pdMS_TO_TICKS(10));
  Serial.println(F("MJPEG video end"));
  vFile.close();
  goto Start;
  }
}
```

The TF card is located between the PCB and the screen, so when designing the housing, you need to pay attention to the opening because

It is not easily visible on the front of the PCB and can be easily missed during design.



When installing the TF card, the metal contacts should face the bottom (opposite to the screen display).

eleven, IIS Audio Output & MIC

Audio output and MIC audio collection are bothIISFor drive connection, use

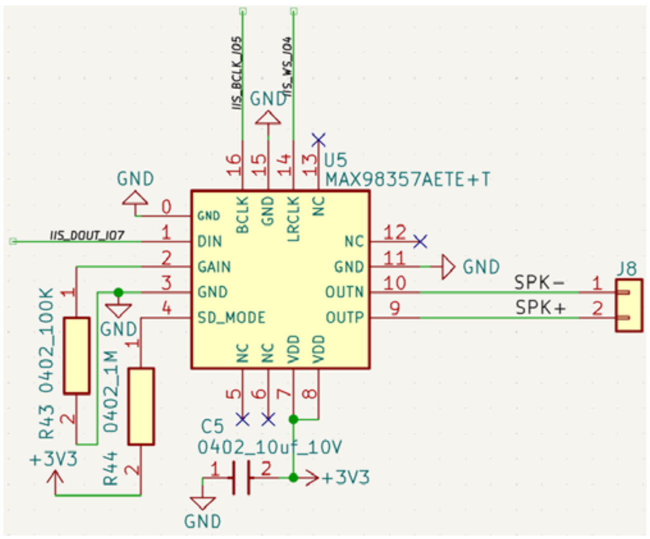
Duplex I2SMode is connected to ESP32, that is, the clock and left and right channels of IIS are shared

pin.

IIS audio output usesMAX98357Chip, maximum output power3W, according to actual use

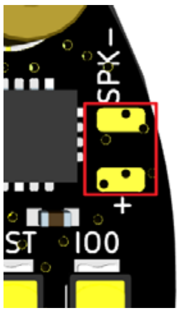
In terms of the effect, it is recommended to use an 8Ω speaker.

For connection with ESP32-S3, please refer to the following diagram or table.



WS	Left and right channels	IO4
BCLK	Clock pin	IO5
DIN	Data pin	IO7

The speaker connection position is shown in the figure below. The speaker wire needs to be soldered to the corresponding pad.



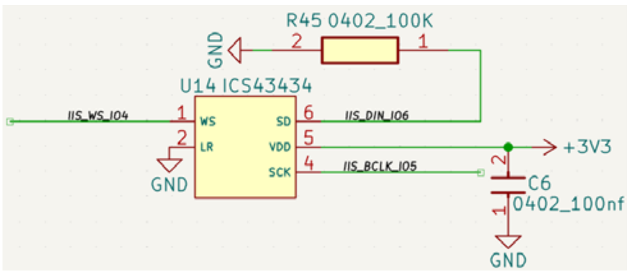
If you need to keep the volume small, it is recommended to use the speaker model used on mobile phones and watches.

When the shell is made into a cavity, a higher volume output can be achieved.

The cavity is very important and will greatly affect the volume of the output sound.

Audio capture usedICS43434, a commonly usedIIS Microphone, with ESP32-

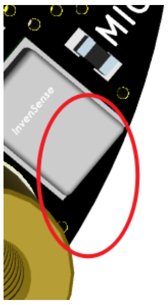
For S3 connections, please refer to the following figure or table.

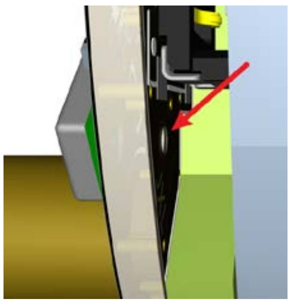


WS	Left and right channels	IO4
BCLK	Clock pin	IO5
DOUT	Data pin	IO6

There is a gap on the side of the development board for collecting sound. Please do not block this gap.

Even when making the shell, a hole needs to be drilled at this position.





For IIS audio acquisition and output, the sample code is mainly used to play MJPEG files.

There is no audio signal. Currently testing is open source on github.Xiaozhi AIAfter code migration

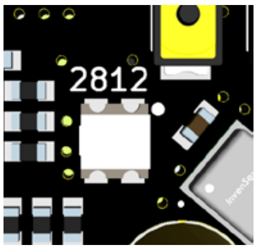
Test, you can also directly burn our transplanted ones, thanks to Xiaozhi AI author for open source such an excellent

Quality project, open source address is as follows https://github.com/78/xiaozhi-esp32

twelve, WS2812 & Expansion IO

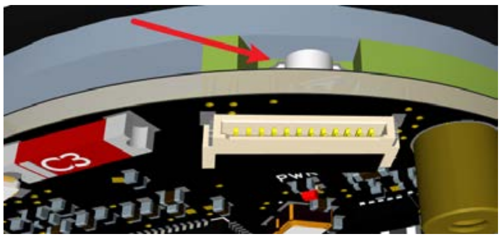
The development board has twoWS2812-RGB lights, connected via a single bus, through ESP32-

S346The first pin is next to the microphone and is located as follows



The second one is connected in series with the first one and is located between the PCB and the screen (the reverse side of the PCB).

Surface), as shown below



The example code uses the #include <Adafruit_NeoPixel.h> library for driving.

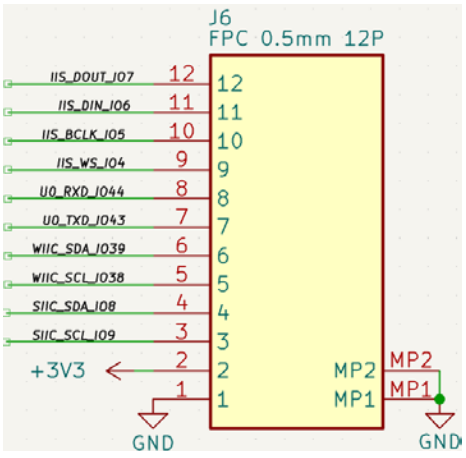
The following code adds the header file and sets the pins and quantities used by WS2812

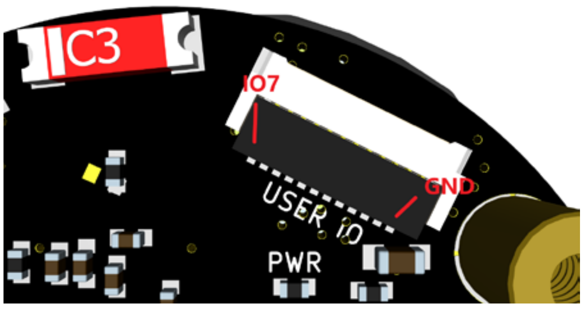
```
# include <Adafruit_NeoPixel.h>

#define WS2812Pin 46
#define WS2812_Count 2
Adafruit_NeoPixel strip(WS2812_Count, WS2812Pin, NEO_GRB+NEO_KHZ800);
```

Initialize and set the initial color

```
void WS2812Init(){
  strip.begin();       //Initializing the WS2812
  strip.clear();       //Clear All LEDs
  strip.show();      //Turn off all lights
  //Set the first light to red (R, G, B)
  strip.setPixelColor(0, strip.Color(20, 0, 0));
  //Set the second light to green
  strip.setPixelColor(1, strip.Color(0, 20, 0));
  strip.show();//Send data to update the lamp beads
  }
```

The development board has an expansion interface, and the pin order is as follows





The reference plan is as follows

IO7/6/5/4	IIS usage	If the audio function is not available or not needed, it can be used as an IO port
IO44/43	UART0	44–RXD 43-TXD external GPS/4G etc.
IO39/38	IIC2/IO*2	Normal IO or second group IIC function enable
IO8/9	IIC1	8–SDA 9-SCL Pay attention to address conflicts with development board sensors

At this point, the basic functions of the development board are introduced in the above documents. You are welcome to submit bugs to us.

So that we can fix or make the product better.