# LCD Screen for Car Dashboard

Notes: The 10.1-inch LCD screen for car dashboard, and the STONE LCD screen combined with RTL8762CJF MCU is used to develop and make an on-board display dashboard.

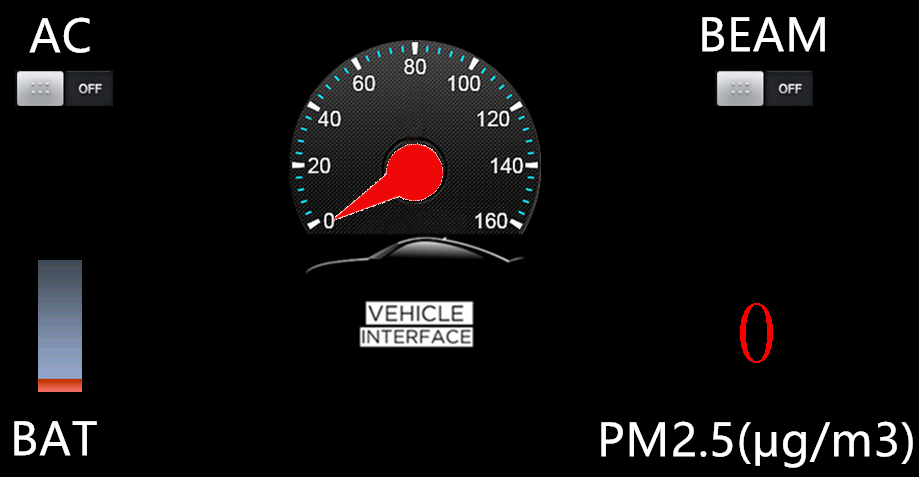
[**Introduction**](C:/Users/Administrator/AppData/Local/Youdao/dict/Application/8.9.0.0/resultui/html/index.html#/javascript:;)

With the rapid development of the economy and the gradual improvement of people's consumption power, cars have become the daily necessities of ordinary families, and everyone pays more attention to the comfort and safety of cars.

The automobile industry has developed for more than a hundred years now, and the automobile has become more and more intelligent with the change of time from the simple machinery at the beginning.How many parts does a car have?There is no specific figure yet.It is estimated that the average car is made up of more than 10,000 indivisible parts.Nowadays, the car has entered thousands of households and become an indispensable partner in daily travel.Therefore, in the process of daily use of the car, we need to always understand the state of their love car, to avoid causing damage to important parts of the car, but also to eliminate potential dangers.Generally, the information displayed on the dashboard is the way to know the status of the vehicle.

I have a stone10.1-inch TFTLCD screen, and this time I plan to make an on-board display dashboard. As we all know, the development of STONE intelligent TFTLCD module screen is convenient and quick, without too many tedious instructions.This is not only suitable for the vast number of learning enthusiasts, but also in the actual project to speed up the development speed, save development time, quickly occupy the market.

The effect picture is as follows:



LCD screen car dashboard

I use the more commonly used RTL8762CJF SCM to develop, through IIC or serial port to achieve the purpose of uploading data to the TFT LCD screen.This time will also use voice broadcast function, to give the driver a better simulation experience.

## lcd screen car dashboard Project function

Here we need to do a used car display project, the project mainly through touch regulation, microcontroller upload instructions manner, simulation with buttons, when MCU button press, through a serial port command to STVC101WT - 01 serial interface screen instructions to upload data, the screen will automatically data parsing, and displayed in the LCD screen.At the same time, there is also a button function on the screen to achieve the serial port instruction, so as to control the MCU.

### In summary, five functions:

1. The serial port screen realizes the bitmap display function;

(2) to achieve the dial rotation function;

(3) to achieve the touch command issued;

(4) to achieve voice broadcasting;

(5) to achieve data instruction upload.

### The function is determined, and then the module selection:

1. Model of touch screen;
2. what kind of MCU module to use;

(3) voice broadcast module.

## Hardware introduction and principle

### The horn

Because STONE serial port screen comes with Audio driver, and reserved the corresponding interface, so you can use the most common magnet loudspeaker, commonly known as horn.Loudspeaker is a kind of transducer which converts electrical signal into sound signal.Loudspeaker is one of the weakest components in sound equipment and one of the most important components for sound effects.There are many kinds of loudspeakers and the prices vary greatly.Audio electrical energy produces sound by making its paper basin or diaphragm vibrate and resonate (resonate) with the surrounding air through an electromagnetic, piezoelectric, or electrostatic effect.

Purchase link:

<https://detail.tmall.com/item.htm?id=529772120978&ali_refid=a3_430583_1006:1104520036:N:Xe0d++yvvuheeXXsNbtAzA==:8c577f8b1f95e6e4517db93b91974d42&ali_trackid=1_8c577f8b1f95e6e4517db93b91974d42&spm=a230r.1.14.1&skuId=3916057122994>

### [STVC101WT-01](https://www.stoneitech.com/product/by-application/civil-type/stvc101wt-01.html) serial LCD screen description

### 10.1-inch 1024x600 industrial-grade TFT panel and 4-wire resistive touch screen;

### Brightness 300cd/m2;

### LED backlight;

### RGB color 65 k;

### The visible area is 222.7mm \* 125.3mm;

### Visual Angle 70/70/50/60;

### Working life 20,000 hours.

### 32-bit cortex-m4 200Hz CPU;

### CPLD EPM240 tft-lcd controller;

### 128MB (or 1GB) of flash memory;

### USB port (U disk) download;

### Toolbox software for GUI design;

### Simple and powerful hexadecimal instruction.

### The basic function

### 8m-128m bytes Flash memory space, SDWe series 128M bytes, SDWa series 8M/16M bytes;

### Support hardware JPG decoding, storage more efficient, faster display;

### Support U disk offline batch download, effectively improve the efficiency of batch download, reduce the professional quality requirements of operators;

### 256-byte register space;

### 64K word (128K bytes) variable memory space, 8 channel curve storage, very fast (80ms) variable display

### Response speed;

### Support up to 128 display variables per page;

### Integrated real-time clock RTC, touch buzzer sound function;

### Support software 90 degree, 180 degree, 270 degree screen rotation, adjust the appropriate visual Angle;

### Support backlight brightness adjustment, auto standby screensaver function;

### Support external matrix keyboard;

### Support audio and video playback;

### Industry leading electromagnetic radiation index, help you easily deal with ClassB;

### The file name naming rule is simple, without corresponding to the Flash block number, also without tedious manual allocation Flash block

### Function;

### Support virtual serial screen function.

### [STONE STVC101WT - 01](https://www.stoneitech.com/product/by-application/civil-type/stvc101wt-01.html) display module is via a serial port communication with MCU, need to use it in this project, we need only through the PC to design good UI images through the menu bar options button, text box, background images, and logical page to add, then generate configuration files, download to the display screen can be run at last.

### 

**The data manual can be downloaded from the official website:**

<https://www.stoneitech.com/support/download>

## RTL8762C EVB Introduction

### 8762C evaluation board provides the hardware environment developed by the customer, including:

### *Power conversion module;*

### *6-axis motion sensor;*

### *4 leds and 6 buttons;*

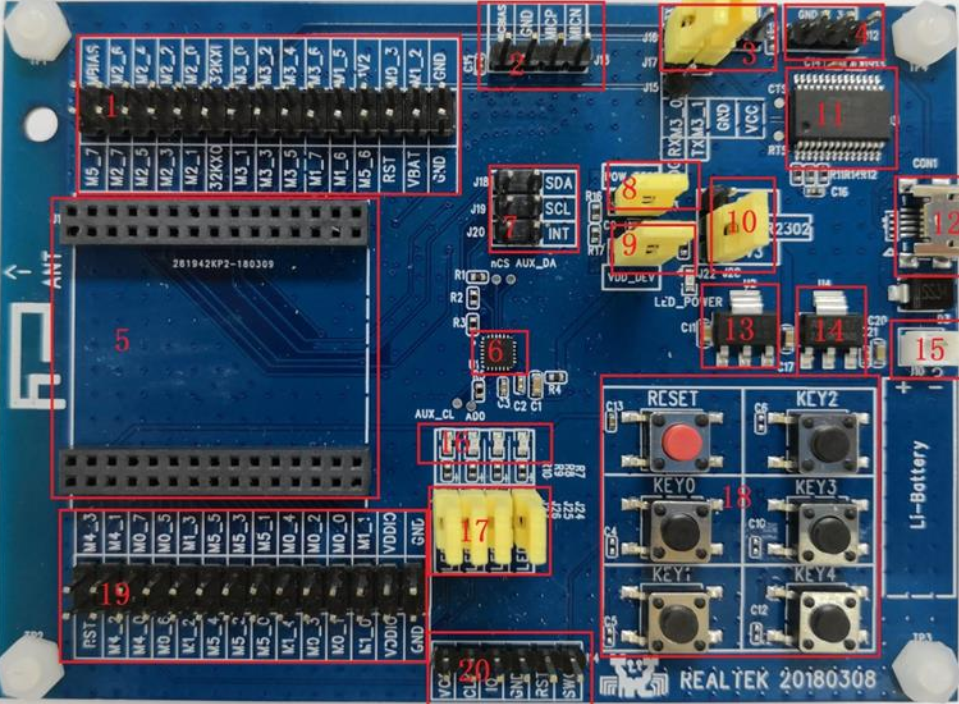
### *Button battery and lithium battery holder;*

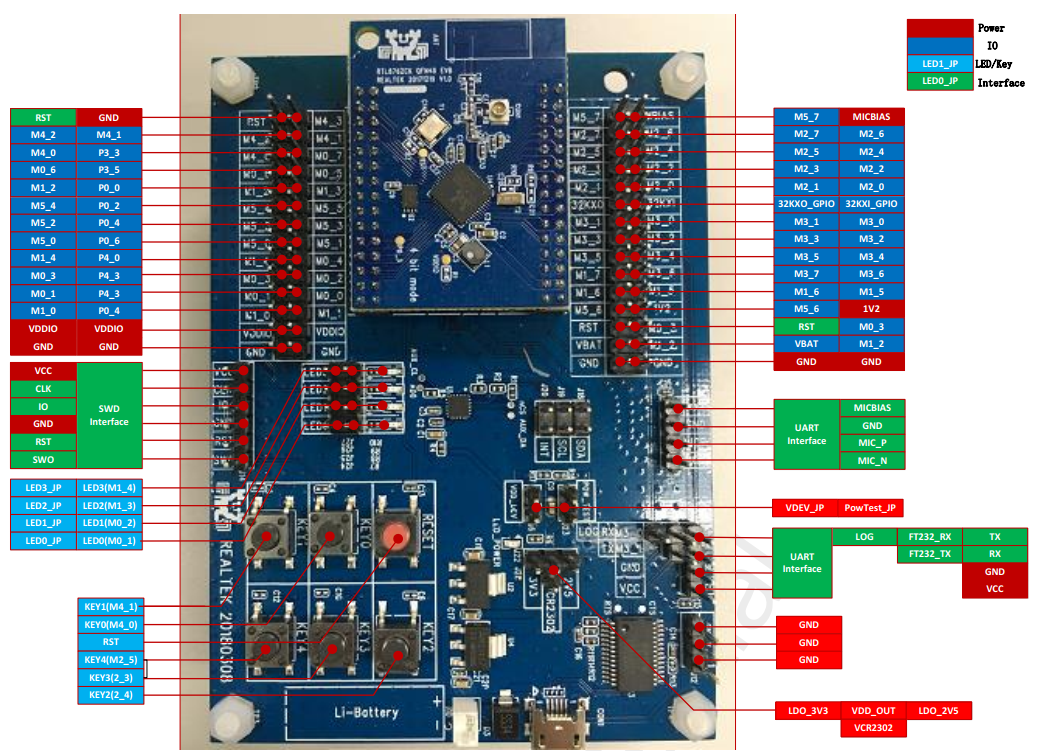
### *USB to UART conversion chip, FT232RL.*

##### Evaluate board block and interface distribution

Detailed description of evaluation board block

Evaluation board block and interface distribution, see the following figure:

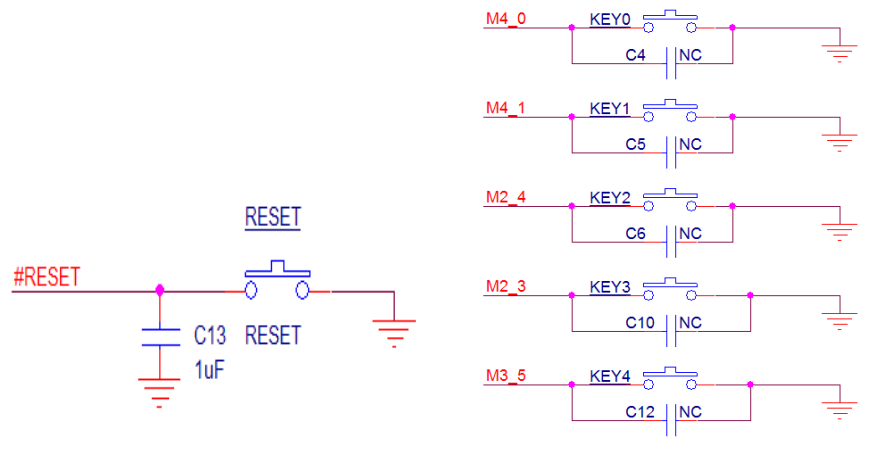




## The keys

There are a total of reset keys and 5 sets of independent keys,

as shown in the following figure:



## The main chip 8762 c

* Flexible GPIO design
* Hardware Keyscan and decoder
* Embedded IR transceiver
* Real-time counter (RTC)
* SPI master/from x two;Timer x 8;I2C x 2;PWM x 8;UART x 2
* 400ksps, 12bit, 8-channel AUXADC
* I2S interface for external audio codecs
* I8080 interface for LCD
* Internal 32K RCOSC keeps BLE links
* Embedded PGA and audio ADC with 5 band equalizer

## STONE TOOL Box Development steps

## In general, there are only three steps:

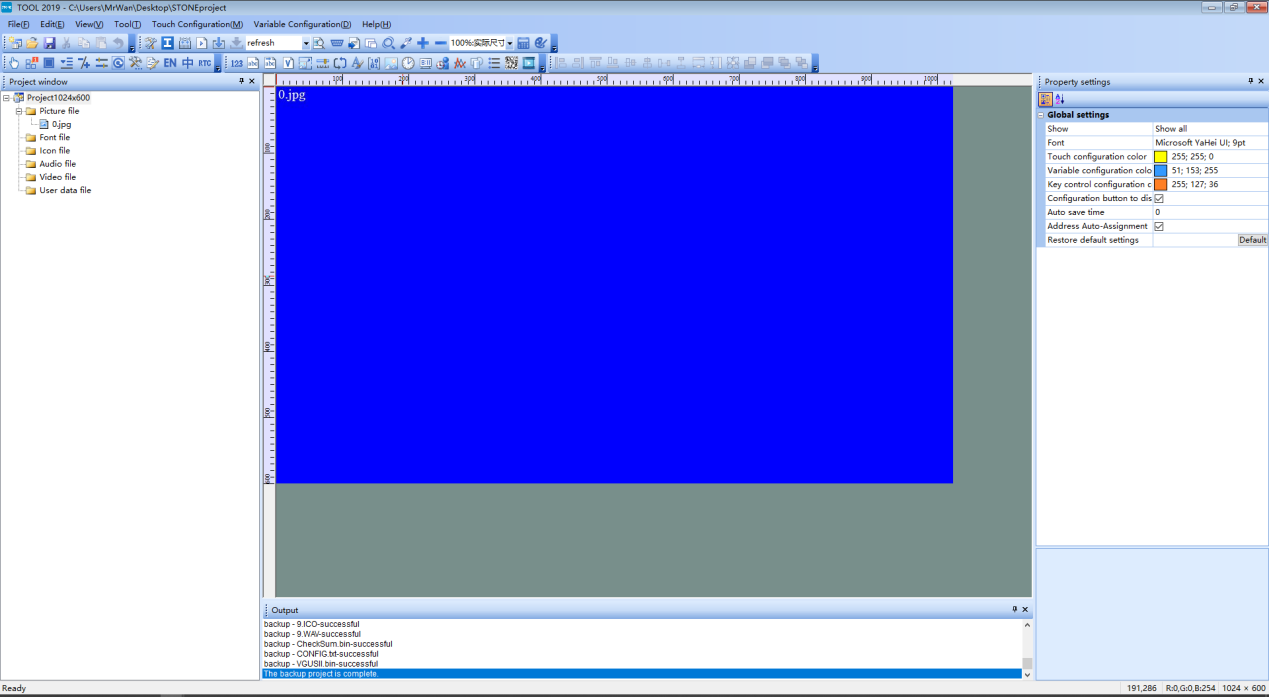
## (1) using TOOL2019 upper computer software design;

## (2) MCU and screen communication development;

## (3) audio file production and import.

## Installation of STONE TOOL

To the TOOL can be downloaded in the website <https://www.stoneitech.com>, as well as relevant USB serial driver. The software interface is as follows:



### The installation of the KEIL

1、Download link: <https://pan.baidu.com/s/1smropvXeXKXw413W_-rvtw>

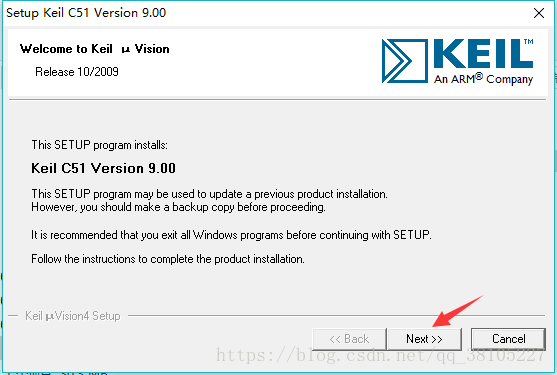
2、Download after decompression

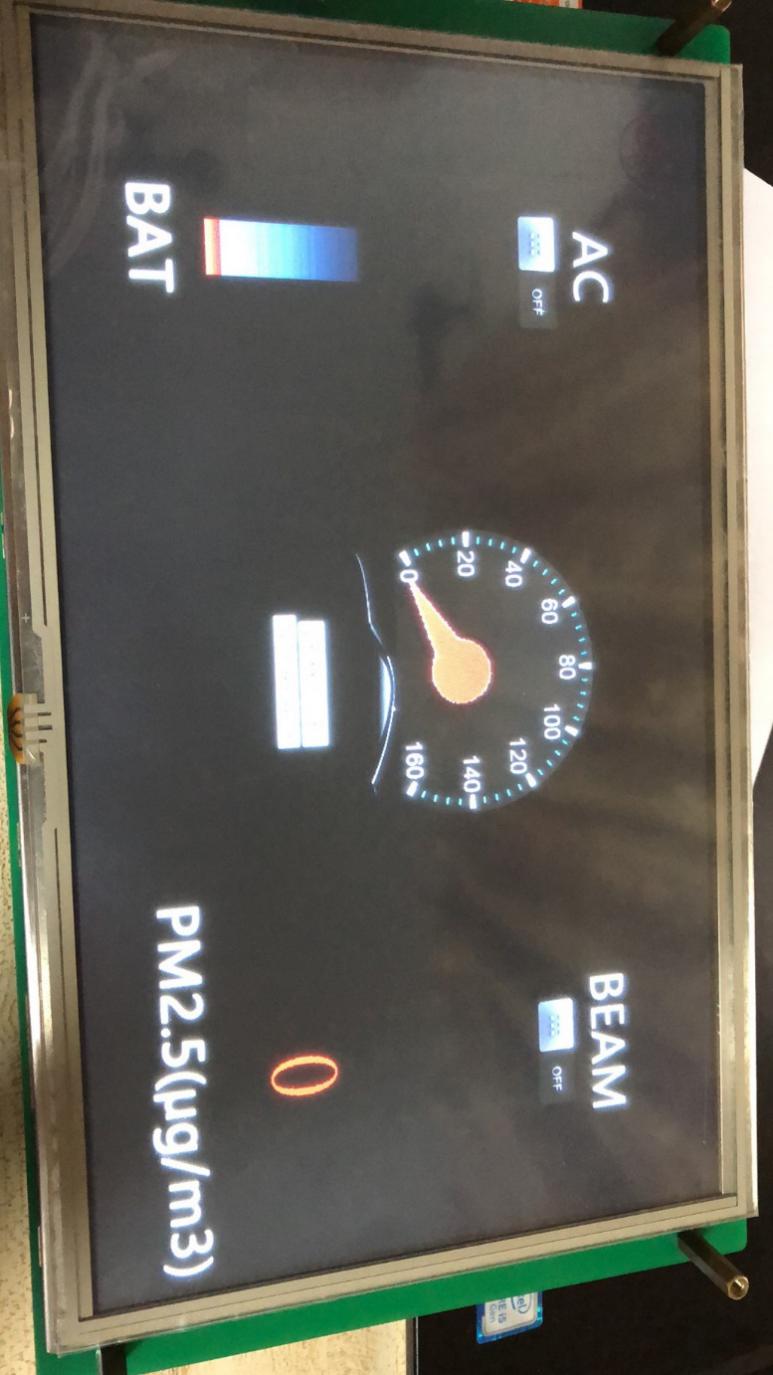


1. Open the folder after unzipping



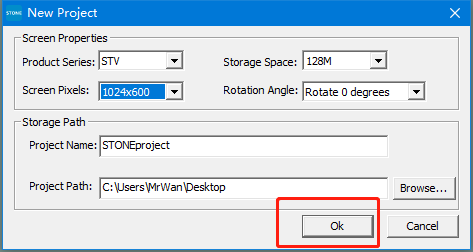
4、Double-click the file c51v900. exe, and click Next in the dialog box.



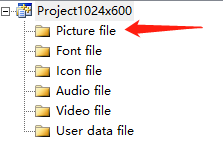


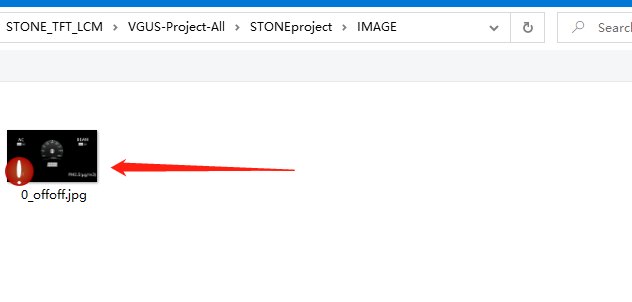
## STONE TOOL 2019 interface design

Using the installed TOOL 2019, click the new project in the upper left corner, and then click OK.



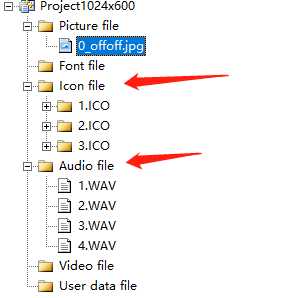
A default project is generated with a blue background by default.Select it, right-click, and select remove to remove the background.Next, right-click picture file and click add to add your own picture background, as:

follows:

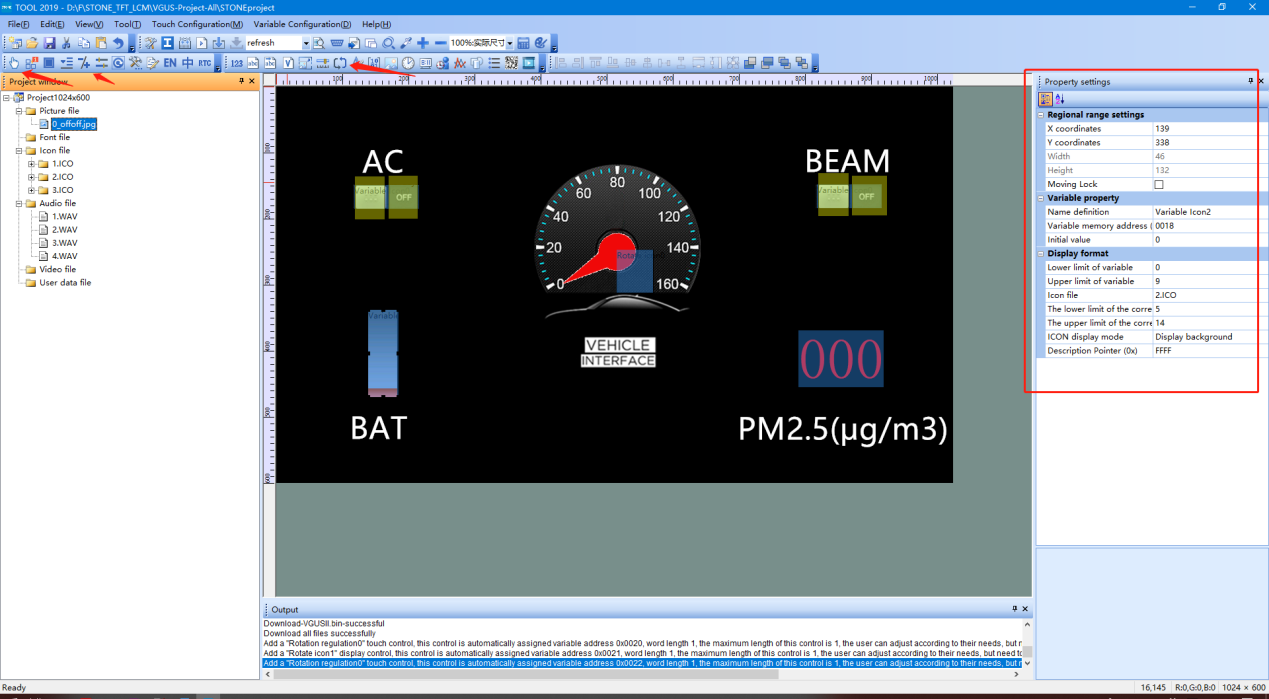


Select the corresponding background image.

In the same way, we add bitmap files and audio files to the project.



Then add the required controls, here is mainly the button control, numeric add and subtract control, data variable control.



### Then configure the variable address of each control, here we have the following configuration:

1. The air conditioning button address is configured as 0x000C;

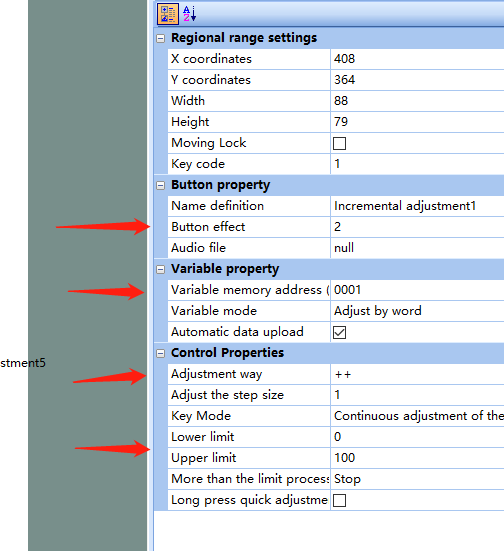
2. High beam button address is configured as 0x000D;

3. speed dial address is configured as 0x001B;

4. electricity icon address is configured as 0x0018;

5. the PM2.5 address is configured as 0x001C;

When the button is configured, the following figure shows once:



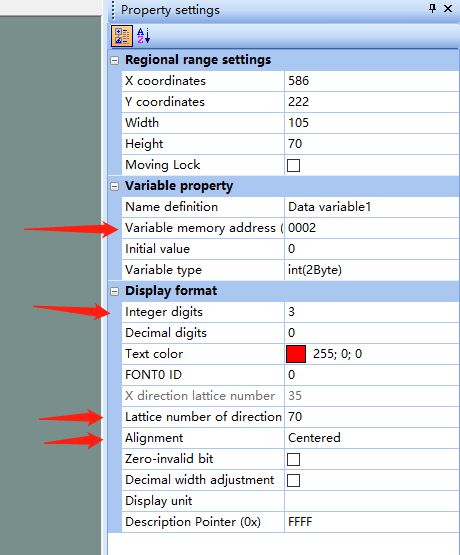
(1) the configuration button press effect;

(2) configure the control of the variable address, used to write its value;

(3) configuration plus or minus operations;

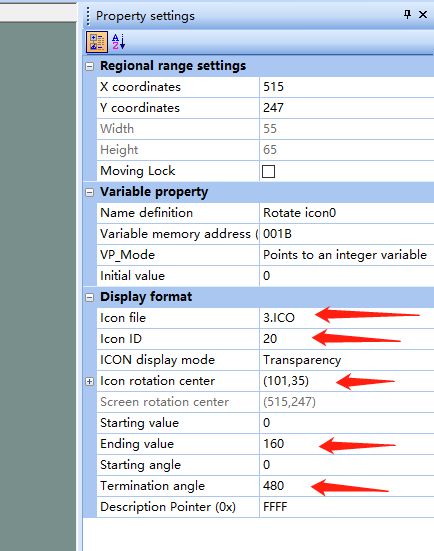
(4) configure the value range.

When configuring the digital text box, the following figure is shown in turn:



1. set the control variable address;
2. set the number of digits;
3. set the size of the number;
4. set the number of the alignment.

When configuring the speedometer, the following figure shows in turn:



1. Selected library file;
2. Which file to specify in the gallery file;
3. Set the center coordinates around the pointer icon;
4. Set the rotation range of the pointer.

Set the rotation Angle of the pointer.

Finally we click on the build configuration tool.

**Note:**

Control buttons are associated with their corresponding bitmaps via variable addresses, so consistency is required to achieve proper control.

#### Therefore, the serial port instruction is as follows:

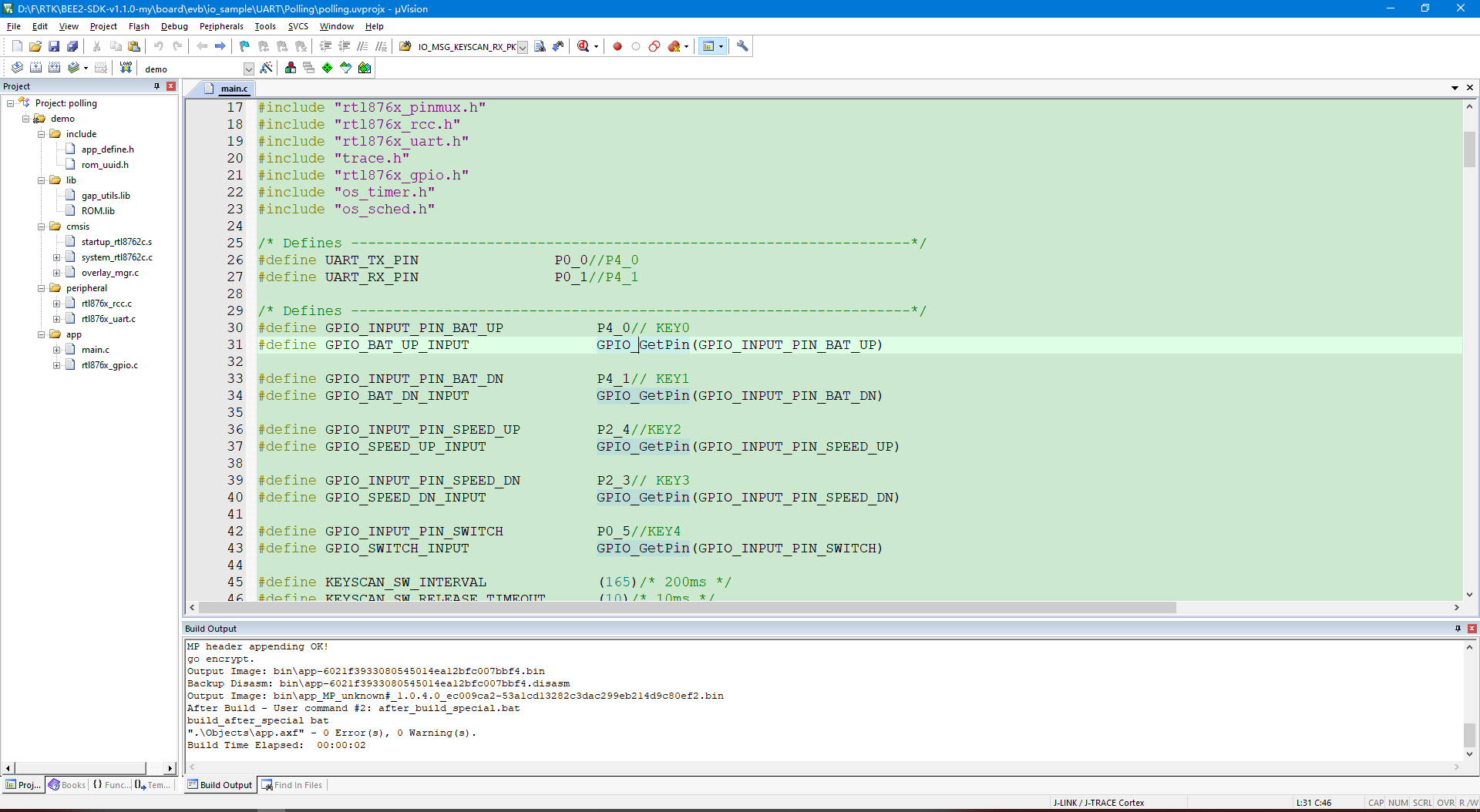
**Battery:** 0xA5, 0x5A, 0x05, 0x82, 0x00, 0x18, 0x00, 0x00

**Speed:** 0xA5, 0x5A, 0x05, 0x82, 0x00, 0x1B, 0x00, 0x00

**PM2.5:** 0xA5, 0x5A, 0x05, 0x82, 0x00, 0x1C, 0x00, 0x00

**The development of RTL8762C**

Open KEIL and import our project file, as shown in the following figure:



Since it is the first time to use, the FLASH algorithm needs to be adjusted accordingly:

Click the option button to go to the Flash Download configuration box and change the algorithm to look like the following figure.

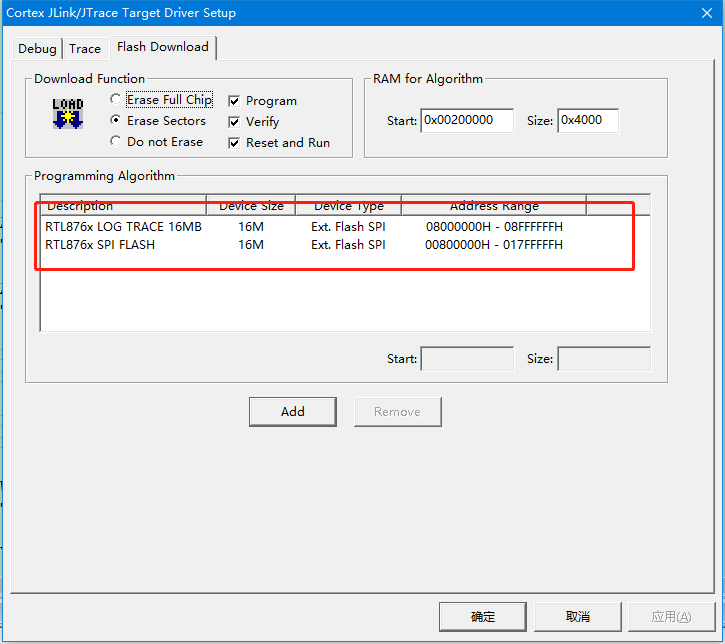


图22

Since the button control is used here, the following changes need to be made in the code：

/\*\*

\* @file main.c

\* @brief uart demo polling tx and rx.

\* @details

\* @author wangzex

\* @date 2018-06-28

\* @version v0.1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* Includes ------------------------------------------------------------------\*/

#include <string.h>

#include "rtl876x\_nvic.h"

#include "rtl876x\_pinmux.h"

#include "rtl876x\_rcc.h"

#include "rtl876x\_uart.h"

#include "trace.h"

#include "rtl876x\_gpio.h"

#include "os\_timer.h"

#include "os\_sched.h"

/\* Defines ------------------------------------------------------------------\*/

#define UART\_TX\_PIN P0\_0//P4\_0

#define UART\_RX\_PIN P0\_1//P4\_1

/\* Defines ------------------------------------------------------------------\*/

#define GPIO\_INPUT\_PIN\_BAT\_UP P4\_0// KEY0

#define GPIO\_BAT\_UP\_INPUT GPIO\_GetPin(GPIO\_INPUT\_PIN\_BAT\_UP)

#define GPIO\_INPUT\_PIN\_BAT\_DN P4\_1// KEY1

#define GPIO\_BAT\_DN\_INPUT GPIO\_GetPin(GPIO\_INPUT\_PIN\_BAT\_DN)

#define GPIO\_INPUT\_PIN\_SPEED\_UP P2\_4//KEY2

#define GPIO\_SPEED\_UP\_INPUT GPIO\_GetPin(GPIO\_INPUT\_PIN\_SPEED\_UP)

#define GPIO\_INPUT\_PIN\_SPEED\_DN P2\_3// KEY3

#define GPIO\_SPEED\_DN\_INPUT GPIO\_GetPin(GPIO\_INPUT\_PIN\_SPEED\_DN)

#define GPIO\_INPUT\_PIN\_SWITCH P0\_5//KEY4

#define GPIO\_SWITCH\_INPUT GPIO\_GetPin(GPIO\_INPUT\_PIN\_SWITCH)

#define KEYSCAN\_SW\_INTERVAL (165)/\* 200ms \*/

#define KEYSCAN\_SW\_RELEASE\_TIMEOUT (10)/\* 10ms \*/

uint8\_t data\_buf\_bat[] = {0xA5, 0x5A, 0x05, 0x82, 0x00, 0x18, 0x00, 0x00};

uint8\_t data\_buf\_speed[] = {0xA5, 0x5A, 0x05, 0x82, 0x00, 0x1B, 0x00, 0x00};

uint8\_t data\_buf\_pm25[] = {0xA5, 0x5A, 0x05, 0x82, 0x00, 0x1C, 0x00, 0x00};//the last byte is data

void \*KeyScan\_Timer\_Handle = NULL;

void timer\_keyscan\_callback(void \*p\_xTimer)

{

uint8\_t gpio\_input\_data\_level = 0;

gpio\_input\_data\_level = GPIO\_ReadInputDataBit(GPIO\_BAT\_UP\_INPUT);//KEY0

DBG\_DIRECT("GPIO\_BAT\_UP\_INPUT = %d\n", gpio\_input\_data\_level);

if(!gpio\_input\_data\_level)

{

gpio\_input\_data\_level = GPIO\_ReadInputDataBit(GPIO\_SWITCH\_INPUT);//KEY0

DBG\_DIRECT("GPIO\_BAT\_UP\_INPUT = %d\n", gpio\_input\_data\_level);

if(!gpio\_input\_data\_level)

{

data\_buf\_pm25[7] ++;

UART\_SendData(UART, data\_buf\_pm25, 8);

}

else

{

data\_buf\_bat[7] ++;

UART\_SendData(UART, data\_buf\_bat, 8);

}

}

gpio\_input\_data\_level = GPIO\_ReadInputDataBit(GPIO\_BAT\_DN\_INPUT);//KEY0

DBG\_DIRECT("GPIO\_BAT\_UP\_INPUT = %d\n", gpio\_input\_data\_level);

if(!gpio\_input\_data\_level)

{

gpio\_input\_data\_level = GPIO\_ReadInputDataBit(GPIO\_SWITCH\_INPUT);//KEY0

DBG\_DIRECT("GPIO\_BAT\_UP\_INPUT = %d\n", gpio\_input\_data\_level);

if(!gpio\_input\_data\_level)

{

data\_buf\_pm25[7] --;

UART\_SendData(UART, data\_buf\_pm25, 8);

}

else

{

data\_buf\_bat[7] --;

UART\_SendData(UART, data\_buf\_bat, 8);

}

}

gpio\_input\_data\_level = GPIO\_ReadInputDataBit(GPIO\_SPEED\_UP\_INPUT);//KEY0

DBG\_DIRECT("GPIO\_BAT\_UP\_INPUT = %d\n", gpio\_input\_data\_level);

if(!gpio\_input\_data\_level)

{

data\_buf\_speed[7] ++;

UART\_SendData(UART, data\_buf\_speed, 8);

}

gpio\_input\_data\_level = GPIO\_ReadInputDataBit(GPIO\_SPEED\_DN\_INPUT);//KEY0

DBG\_DIRECT("GPIO\_BAT\_UP\_INPUT = %d\n", gpio\_input\_data\_level);

if(!gpio\_input\_data\_level)

{

data\_buf\_speed[7] --;

UART\_SendData(UART, data\_buf\_speed, 8);

}

// gpio\_input\_data\_level = GPIO\_ReadInputDataBit(GPIO\_PM25\_INPUT);//KEY0

// DBG\_DIRECT("GPIO\_BAT\_UP\_INPUT = %d\n", gpio\_input\_data\_level);

// if(!gpio\_input\_data\_level)

// {

// data\_buf\_pm25[7] --;

// UART\_SendData(UART, data\_buf\_pm25, 8);

// }

os\_timer\_restart(&KeyScan\_Timer\_Handle, KEYSCAN\_SW\_INTERVAL);

}

void timer\_keyscan\_init(void)

{

DBG\_DIRECT("[io\_keyscan] timer\_keyscan\_init: keyscan timer init");

if (false == os\_timer\_create(&KeyScan\_Timer\_Handle, "keyscan\_timer", 1, \

KEYSCAN\_SW\_INTERVAL, false, timer\_keyscan\_callback))

{

DBG\_DIRECT("[io\_keyscan] timer\_keyscan\_init: timer creat failed!");

}

os\_timer\_start(&KeyScan\_Timer\_Handle);

}

/\*\* @brief UART\_BaudRate\_Table

\* div ovsr ovsr\_adj :These three parameters set the baud rate calibration parameters of UART.

baudrate | div | ovsr | ovsr\_adj

--------------------------------------------------------

1200Hz | 2589 | 7 | 0x7F7

9600Hz | 271 | 10 | 0x24A

14400Hz | 271 | 5 | 0x222

19200Hz | 165 | 7 | 0x5AD

28800Hz | 110 | 7 | 0x5AD

38400Hz | 85 | 7 | 0x222

57600Hz | 55 | 7 | 0x5AD

76800Hz | 35 | 9 | 0x7EF

115200Hz | 20 | 12 | 0x252

128000Hz | 25 | 7 | 0x555

153600Hz | 15 | 12 | 0x252

230400Hz | 10 | 12 | 0x252

460800Hz | 5 | 12 | 0x252

500000Hz | 8 | 5 | 0

921600Hz | 4 | 5 | 0x3F7

1000000Hz | 4 | 5 | 0

1382400Hz | 2 | 9 | 0x2AA

1444400Hz | 2 | 8 | 0x5F7

1500000Hz | 2 | 8 | 0x492

1843200Hz | 2 | 5 | 0x3F7

2000000Hz | 2 | 5 | 0

2100000Hz | 1 | 14 | 0x400

2764800Hz | 1 | 9 | 0x2AA

3000000Hz | 1 | 8 | 0x492

3250000Hz | 1 | 7 | 0x112

3692300Hz | 1 | 5 | 0x5F7

3750000Hz | 1 | 5 | 0x36D

4000000Hz | 1 | 5 | 0

6000000Hz | 1 | 1 | 0x36D

-----------------------------------------------------

\*/ /\* End of UART\_BaudRate\_Table \*/

/\* Globals ------------------------------------------------------------------\*/

typedef struct

{

uint16\_t div;

uint16\_t ovsr;

uint16\_t ovsr\_adj;

} UART\_BaudRate\_TypeDef;

const UART\_BaudRate\_TypeDef BaudRate\_Table[10] =

{

{271, 10, 0x24A}, // BAUD\_RATE\_9600

{165, 7, 0x5AD}, // BAUD\_RATE\_19200

{20, 12, 0x252}, // BAUD\_RATE\_115200

{10, 12, 0x252}, // BAUD\_RATE\_230400

{5, 12, 0x252}, // BAUD\_RATE\_460800

{4, 5, 0x3F7}, // BAUD\_RATE\_921600

{2, 5, 0}, // BAUD\_RATE\_2000000

{1, 8, 0x492}, // BAUD\_RATE\_3000000

{1, 5, 0}, // BAUD\_RATE\_4000000

{1, 1, 0x36D}, // BAUD\_RATE\_6000000

};

void board\_gpio\_init(void)

{

Pad\_Config(GPIO\_INPUT\_PIN\_BAT\_UP, PAD\_PINMUX\_MODE, PAD\_IS\_PWRON, PAD\_PULL\_UP, PAD\_OUT\_DISABLE,

PAD\_OUT\_HIGH);

Pinmux\_Config(GPIO\_INPUT\_PIN\_BAT\_UP, DWGPIO);

Pad\_Config(GPIO\_INPUT\_PIN\_BAT\_DN, PAD\_PINMUX\_MODE, PAD\_IS\_PWRON, PAD\_PULL\_UP, PAD\_OUT\_DISABLE,

PAD\_OUT\_HIGH);

Pinmux\_Config(GPIO\_INPUT\_PIN\_BAT\_DN, DWGPIO);

Pad\_Config(GPIO\_INPUT\_PIN\_SPEED\_UP, PAD\_PINMUX\_MODE, PAD\_IS\_PWRON, PAD\_PULL\_UP, PAD\_OUT\_DISABLE,

PAD\_OUT\_HIGH);

Pinmux\_Config(GPIO\_INPUT\_PIN\_SPEED\_UP, DWGPIO);

Pad\_Config(GPIO\_INPUT\_PIN\_SPEED\_DN, PAD\_PINMUX\_MODE, PAD\_IS\_PWRON, PAD\_PULL\_UP, PAD\_OUT\_DISABLE,

PAD\_OUT\_HIGH);

Pinmux\_Config(GPIO\_INPUT\_PIN\_SPEED\_DN, DWGPIO);

Pad\_Config(GPIO\_INPUT\_PIN\_SWITCH, PAD\_PINMUX\_MODE, PAD\_IS\_PWRON, PAD\_PULL\_UP, PAD\_OUT\_DISABLE,

PAD\_OUT\_HIGH);

Pinmux\_Config(GPIO\_INPUT\_PIN\_SWITCH, DWGPIO);

}

uint8\_t String\_Buf[100];

/\*\*

\* @brief Initialization of pinmux settings and pad settings.

\* @param No parameter.

\* @return void

\*/

void board\_uart\_init(void)

{

Pad\_Config(UART\_TX\_PIN, PAD\_PINMUX\_MODE, PAD\_IS\_PWRON, PAD\_PULL\_UP, PAD\_OUT\_DISABLE, PAD\_OUT\_HIGH);

Pad\_Config(UART\_RX\_PIN, PAD\_PINMUX\_MODE, PAD\_IS\_PWRON, PAD\_PULL\_UP, PAD\_OUT\_DISABLE, PAD\_OUT\_HIGH);

Pinmux\_Config(UART\_TX\_PIN, UART0\_TX);

Pinmux\_Config(UART\_RX\_PIN, UART0\_RX);

}

/\*\*

\* @brief Initialize GPIO peripheral.

\* @param No parameter.

\* @return void

\*/

void driver\_gpio\_init(void)

{

/\* Initialize GPIO peripheral \*/

RCC\_PeriphClockCmd(APBPeriph\_GPIO, APBPeriph\_GPIO\_CLOCK, ENABLE);

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_StructInit(&GPIO\_InitStruct);

GPIO\_InitStruct.GPIO\_Pin = GPIO\_INPUT\_PIN\_BAT\_UP;

GPIO\_InitStruct.GPIO\_Mode = GPIO\_Mode\_IN;

GPIO\_InitStruct.GPIO\_ITCmd = DISABLE;

GPIO\_Init(&GPIO\_InitStruct);

GPIO\_InitStruct.GPIO\_Pin = GPIO\_INPUT\_PIN\_BAT\_DN;

GPIO\_Init(&GPIO\_InitStruct);

GPIO\_InitStruct.GPIO\_Pin = GPIO\_INPUT\_PIN\_SPEED\_UP;

GPIO\_Init(&GPIO\_InitStruct);

GPIO\_InitStruct.GPIO\_Pin = GPIO\_INPUT\_PIN\_SPEED\_DN;

GPIO\_Init(&GPIO\_InitStruct);

GPIO\_InitStruct.GPIO\_Pin = GPIO\_INPUT\_PIN\_SWITCH;

GPIO\_Init(&GPIO\_InitStruct);

}

/\*\*

\* @brief Demo code of operation about gpio.

\* @param No parameter.

\* @return void

\*/

void gpio\_demo(void)

{

/\* Configure pad and pinmux firstly! \*/

board\_gpio\_init();

/\* Initialize gpio peripheral \*/

driver\_gpio\_init();

}

/\*\*

\* @brief Initialize uart peripheral.

\* @param No parameter.

\* @return void

\*/

void driver\_uart\_init(void)

{

UART\_DeInit(UART);

RCC\_PeriphClockCmd(APBPeriph\_UART0, APBPeriph\_UART0\_CLOCK, ENABLE);

/\* uart init \*/

UART\_InitTypeDef UART\_InitStruct;

UART\_StructInit(&UART\_InitStruct);

/\* Config uart baudrate \*/

UART\_InitStruct.div = BaudRate\_Table[BAUD\_RATE\_115200].div;

UART\_InitStruct.ovsr = BaudRate\_Table[BAUD\_RATE\_115200].ovsr;

UART\_InitStruct.ovsr\_adj = BaudRate\_Table[BAUD\_RATE\_115200].ovsr\_adj;

UART\_InitStruct.parity = UART\_PARITY\_NO\_PARTY;

UART\_InitStruct.stopBits = UART\_STOP\_BITS\_1;

UART\_InitStruct.wordLen = UART\_WROD\_LENGTH\_8BIT;

UART\_InitStruct.rxTriggerLevel = 16; //1~29

UART\_InitStruct.idle\_time = UART\_RX\_IDLE\_2BYTE; //idle interrupt wait time

UART\_Init(UART, &UART\_InitStruct);

}

/\*\*

\* @brief UARt send data continuous.

\* @param No parameter.

\* @return void

\*/

void uart\_senddata\_continuous(UART\_TypeDef \*UARTx, const uint8\_t \*pSend\_Buf, uint16\_t vCount)

{

uint8\_t count;

while (vCount / UART\_TX\_FIFO\_SIZE > 0)

{

while (UART\_GetFlagState(UARTx, UART\_FLAG\_THR\_EMPTY) == 0);

for (count = UART\_TX\_FIFO\_SIZE; count > 0; count--)

{

UARTx->RB\_THR = \*pSend\_Buf++;

}

vCount -= UART\_TX\_FIFO\_SIZE;

}

while (UART\_GetFlagState(UARTx, UART\_FLAG\_THR\_EMPTY) == 0);

while (vCount--)

{

UARTx->RB\_THR = \*pSend\_Buf++;

}

}

/\*\*

\* @brief Demo code of uart.

\* @param No parameter.

\* @return void

\*/

void uart\_demo(void)

{

uint16\_t demo\_str\_len = 0;

// uint8\_t rx\_byte = 0;

board\_uart\_init();

driver\_uart\_init();

char \*demo\_str = "### Uart demo polling read uart data ###\r\n";

demo\_str\_len = strlen(demo\_str);

memcpy(String\_Buf, demo\_str, demo\_str\_len);

/\* Send demo tips \*/

uart\_senddata\_continuous(UART, String\_Buf, demo\_str\_len);

/\* Loop rx and tx \*/

// while (1)

// {

// if (UART\_GetFlagState(UART, UART\_FLAG\_RX\_DATA\_RDY) == SET)

// {

// rx\_byte = UART\_ReceiveByte(UART);

// UART\_SendByte(UART, rx\_byte);

// }

// }

}

/\*\*

\* @brief Entry of app code

\* @return int (To avoid compile warning)

\*/

int main(void)

{

uart\_demo();

gpio\_demo();

timer\_keyscan\_init();

os\_sched\_start();

return 0;

}

Finally, just connect the MCU to the serial port LCD screen lcd for car dashboard

and connect the speaker to demonstrate.

