

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.

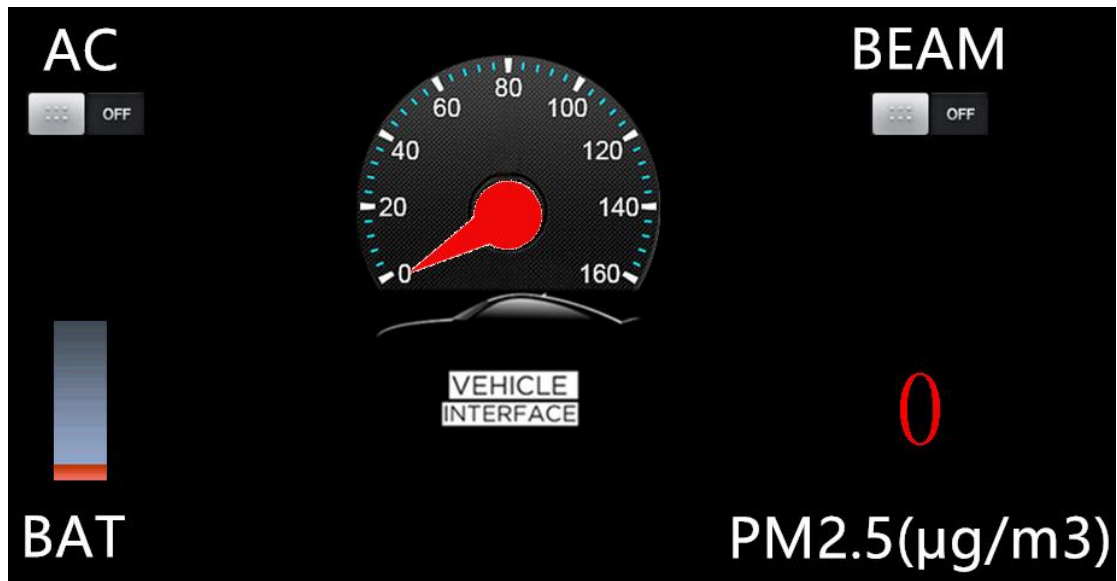
Introduction

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 stone 10.1-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++yvvueeXXsNbtAzA==:8c577f8b1f95e6e4517db93b91974d42&ali_trackid=1_8c577f8b1f95e6e4517db93b91974d42&spm=a230r.1.14.1&skuld=3916057122994

STVC101WT-01 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](#) 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

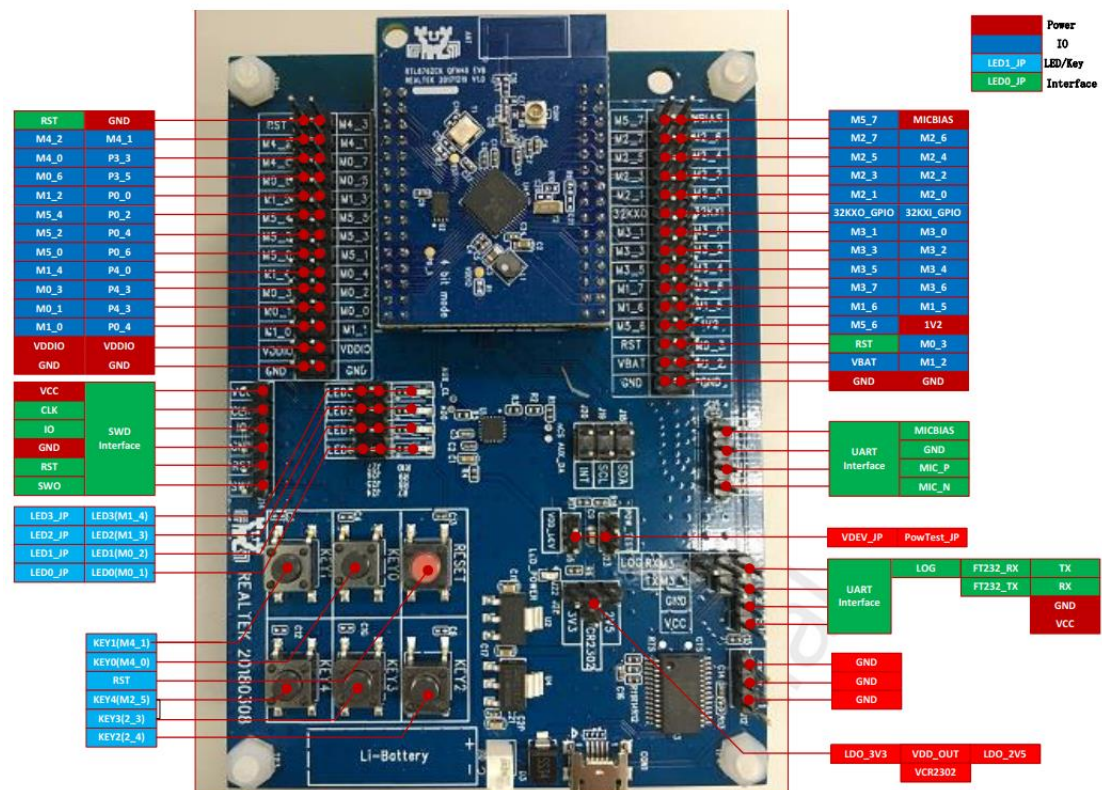
8762C evaluation board provides the hardware environment developed by the customer, including:

- 1) *Power conversion module;*
- 2) *6-axis motion sensor;*
- 3) *4 leds and 6 buttons;*
- 4) *Button battery and lithium battery holder;*
- 5) *USB to UART conversion chip, FT232RL.*

Evaluate board block and interface distribution

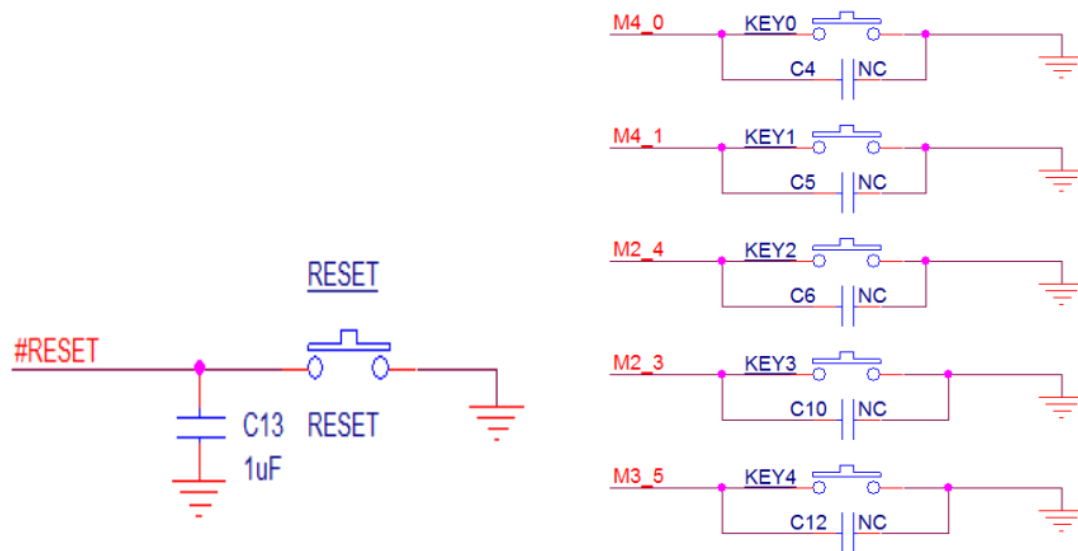
Detailed description of evaluation board block

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

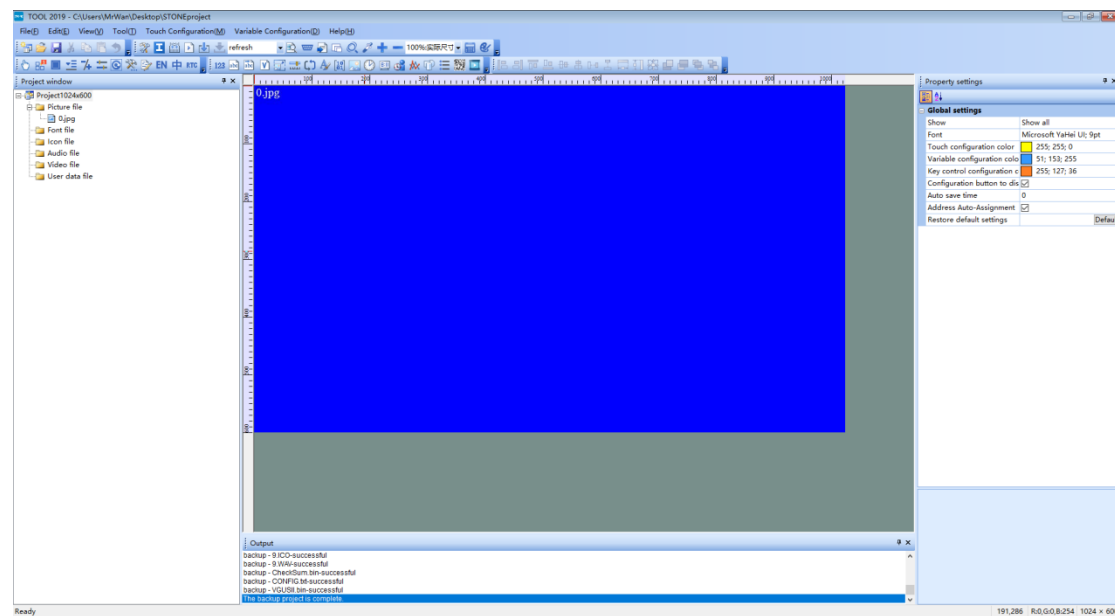
STONE TOOL Box Development steps

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> -rvrtw

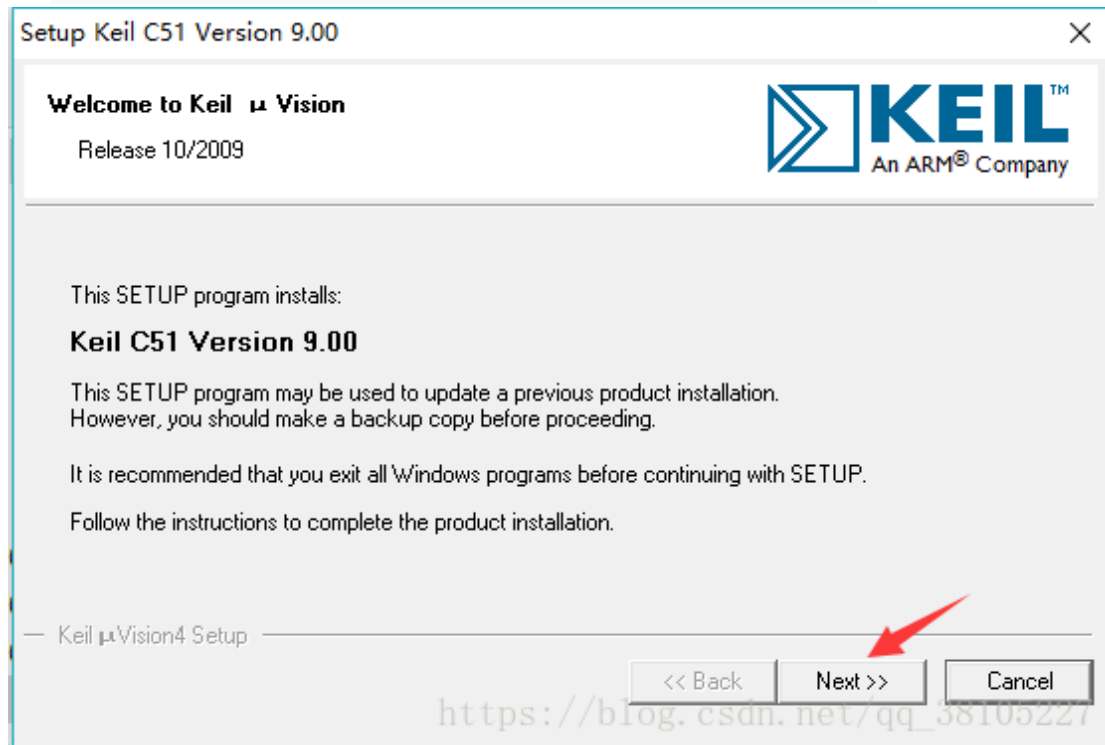
2、Download after decompression

C51+Keil4完美版.rar
keil4安装视频说明.zip

3、Open the folder after unzipping

注册机
C51V900.exe

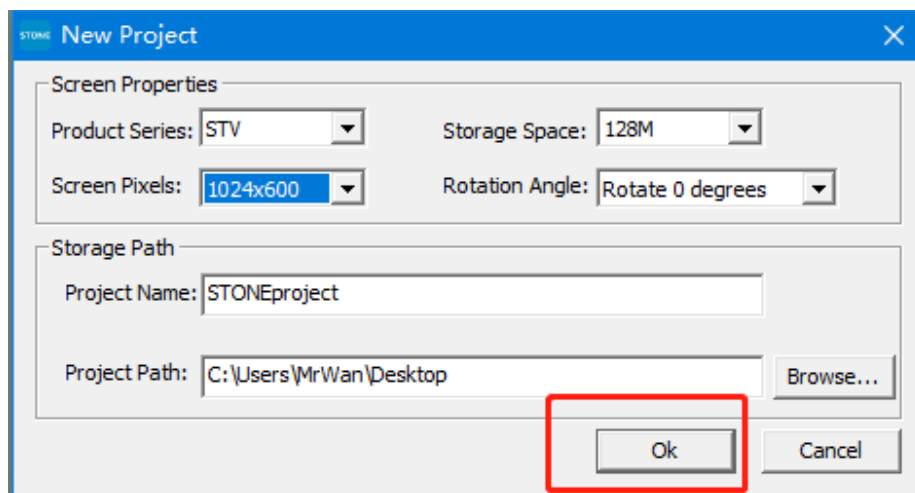
4、Double-click the file c51v900. exe, and click Next in the dialog box.



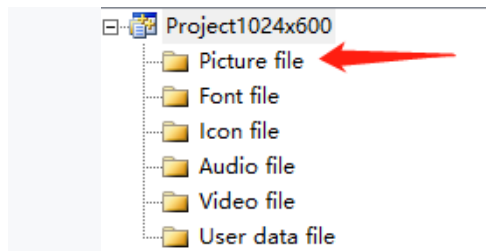


STONE TOOL 2019 interface design

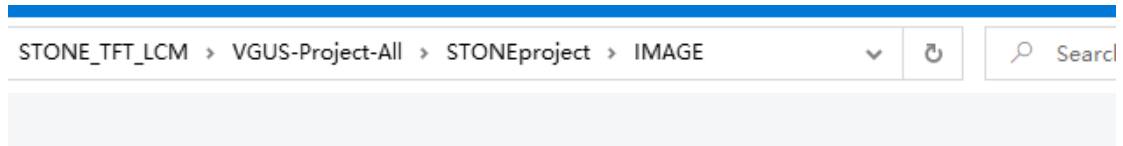
Using the installed TOOL 2019, click the new project in the upper left corner, and then click OK.



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:

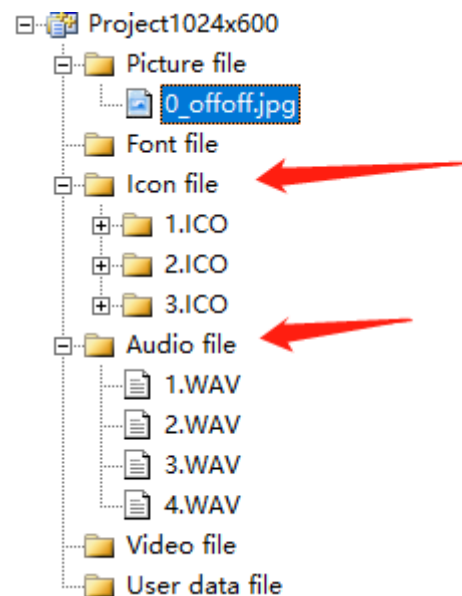


follows:

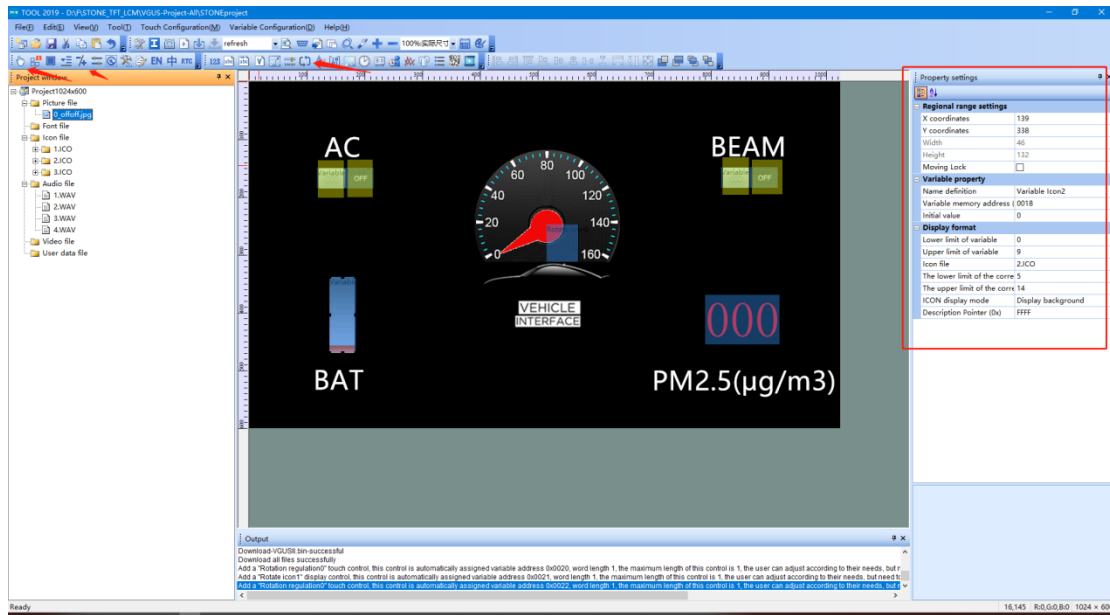


Select the corresponding background image.

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



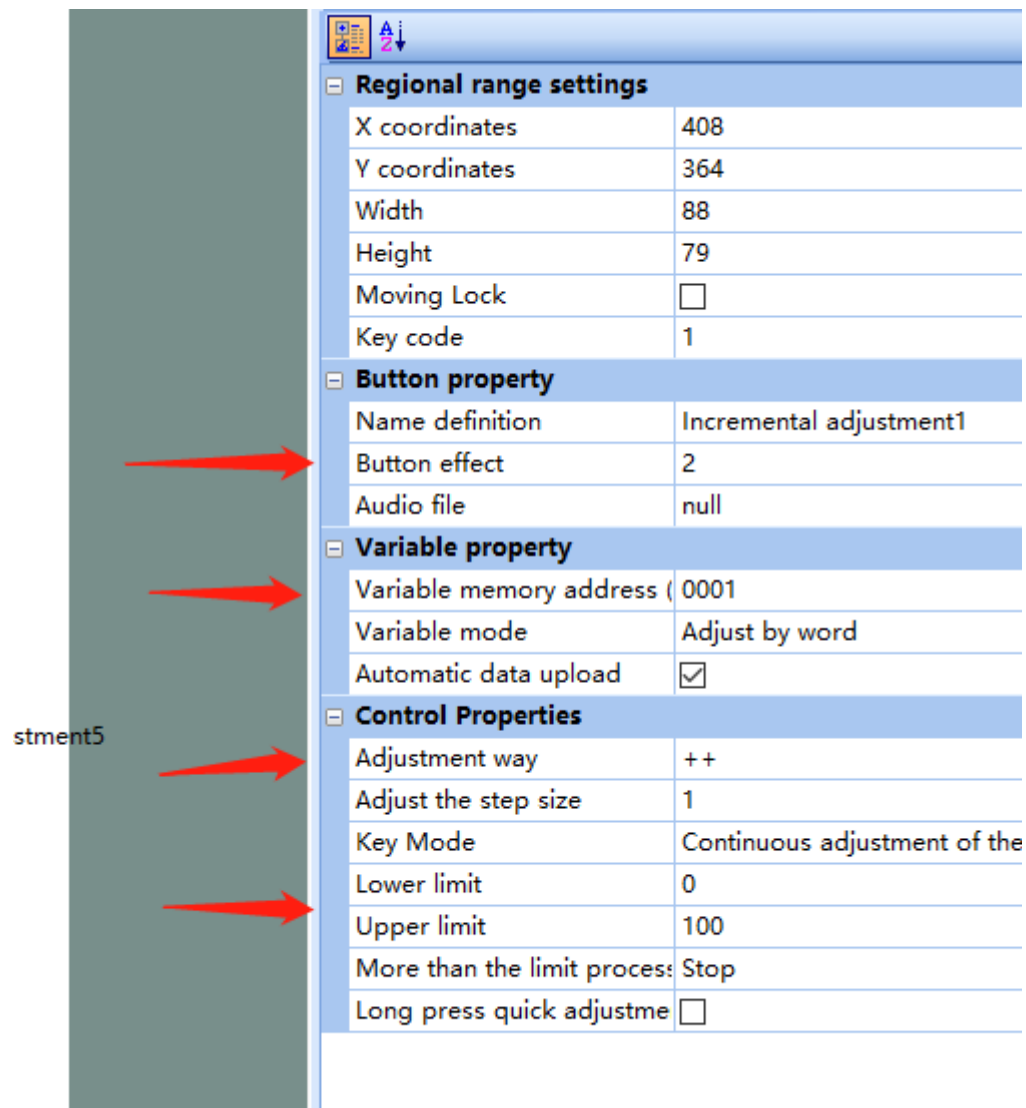
Then add the required controls, here is mainly the button control, numeric add and subtract control, data variable control.



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:

Property settings


Regional range settings

X coordinates	586
Y coordinates	222
Width	105
Height	70
Moving Lock	<input type="checkbox"/>

Variable property

Name definition	Data variable1
Variable memory address (0002
Initial value	0
Variable type	int(2Byte)

Display format

Integer digits	3
Decimal digits	0
Text color	 255; 0; 0
FONT0 ID	0
X direction lattice number	35
Lattice number of direction	70
Alignment	Centered
Zero-invalid bit	<input type="checkbox"/>
Decimal width adjustment	<input type="checkbox"/>
Display unit	
Description Pointer (0x)	FFFF

- 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:

Property settings	
Regional range settings	
X coordinates	515
Y coordinates	247
Width	55
Height	65
Moving Lock	<input type="checkbox"/>
Variable property	
Name definition	Rotate icon0
Variable memory address (001B
VP_Mode	Points to an integer variable
Initial value	0
Display format	
Icon file	3.ICO
Icon ID	20
ICON display mode	Transparency
Icon rotation center	(101,35)
Screen rotation center	(515,247)
Starting value	0
Ending value	160
Starting angle	0
Termination angle	480
Description Pointer (0x)	FFFF

- ⑤ Selected library file;
- ⑥ Which file to specify in the gallery file;
- ⑦ Set the center coordinates around the pointer icon;
- ⑧ 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:

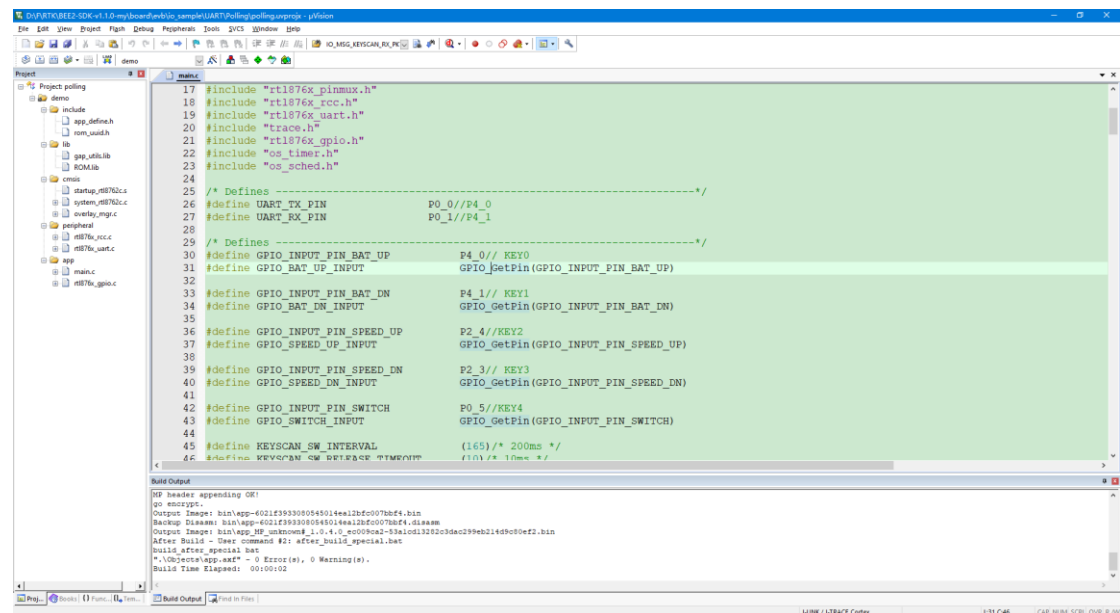
Battery: 0xA5, 0x5A, 0x05, 0x82, 0x00, 0x18, 0x00, 0x00

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.

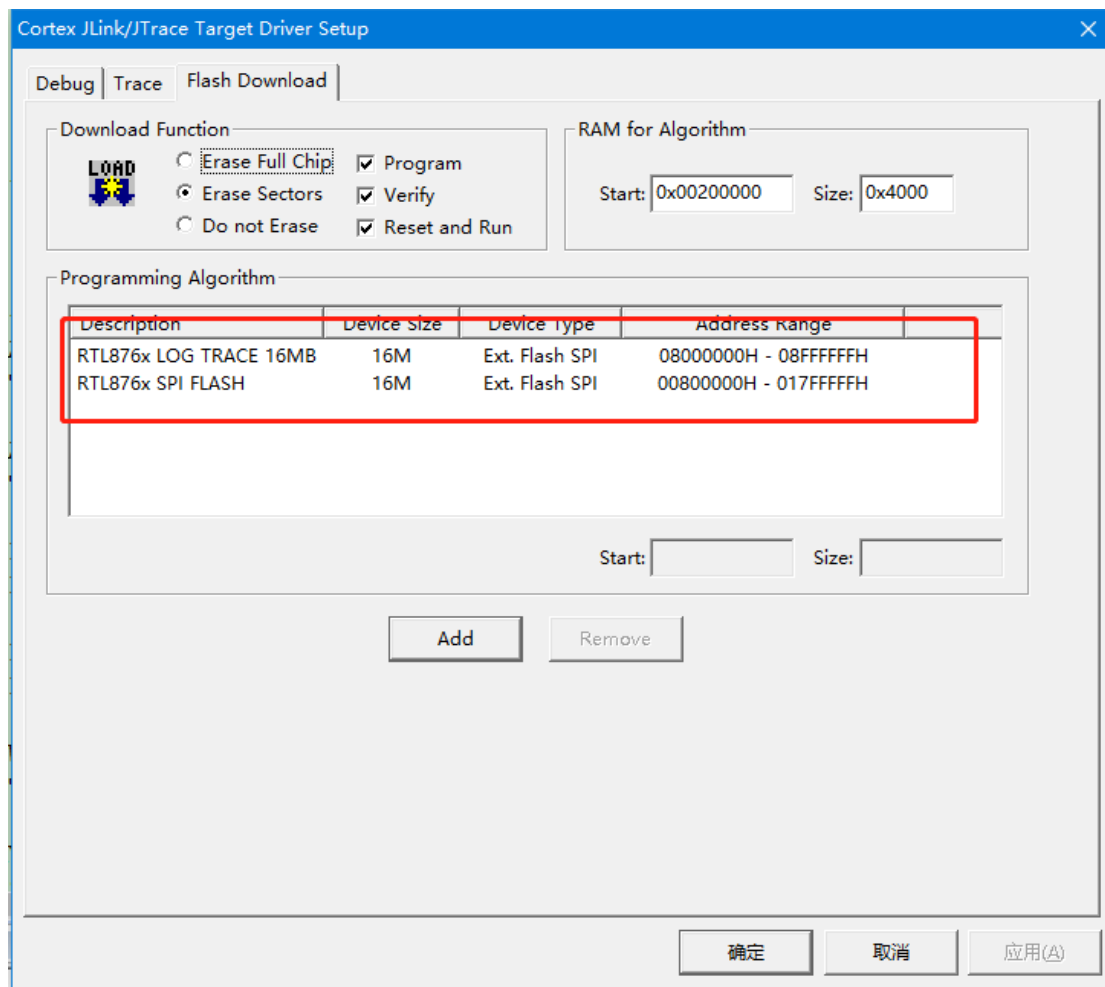


图 22

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 KEYSKAN_SW_INTERVAL        (165)* 200ms */
#define KEYSKAN_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_InitStructure;
    GPIO_StructInit(&GPIO_InitStructure);
    GPIO_InitStructure.GPIO_Pin    = GPIO_INPUT_PIN_BAT_UP;
    GPIO_InitStructure.GPIO_Mode   = GPIO_Mode_IN;
    GPIO_InitStructure.GPIO_ITCmd  = DISABLE;
    GPIO_Init(&GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin    = GPIO_INPUT_PIN_BAT_DN;
    GPIO_Init(&GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin    = GPIO_INPUT_PIN_SPEED_UP;
    GPIO_Init(&GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin    = GPIO_INPUT_PIN_SPEED_DN;
    GPIO_Init(&GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin    = GPIO_INPUT_PIN_SWITCH;
    GPIO_Init(&GPIO_InitStructure);
}

/**
 * @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.

