Was linked MCU Program Description

category	content
Key words	**Was linked MCU program**
Summary	

revise history

version	date	the reason	prepared by	Examine
V1.0	2018/6/6	Create documents	Lin Qingtian	
V1.2	2018/7/12	Serial screen instructions issued to add Introduction P2	Lin Qingtian	
V1.4	2018/8/27	Time sliding interface UI change	Lin Qingtian	

Sales and Service

Guangzhou color Optoelectronics Technology Co., Ltd.

phone: 020-82186683

fax: 020-82187676 Email : hmi@gz-dc.com (Public

Service) website: http://www.gz-dc.com

Address: Guangzhou High-tech Industrial Development Zone, Yushu Industrial Park, Beverly West 8 number C Building 303 Housing

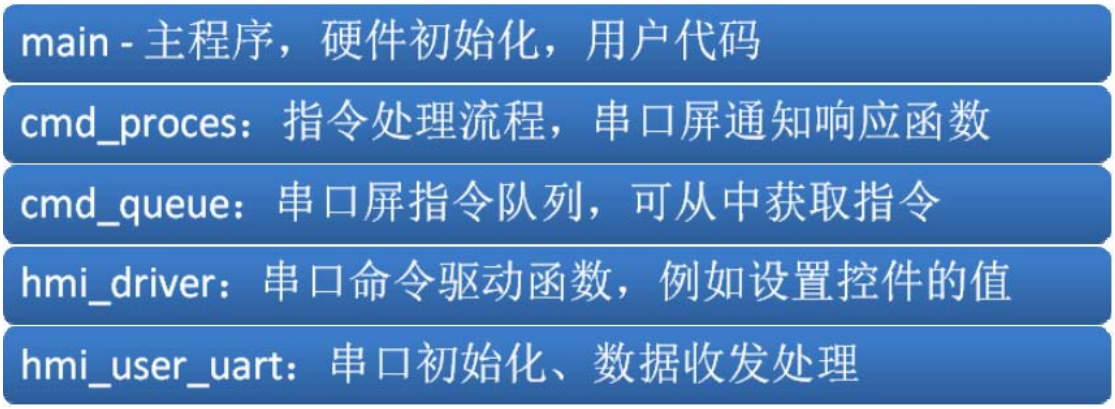
official website Taobao retail shop: https://gz-dc.taobao.com

table of Contents

1. Introduce procedural framework

Our screen provides a serial driver code and example programs, SCM currently supported platforms 51 , STM32 . On our test machine microcontroller is routine STM32F103VCT6 with STC89CX Series, the user can modify the example program, the function of the existing reference program, and add and modify their function code (e.g., acquisition of temperature and humidity, switch control, etc.). This document referenced by the program STM32 Routines, but 51 Realize the function code is the same, just different timers and serial port configuration.

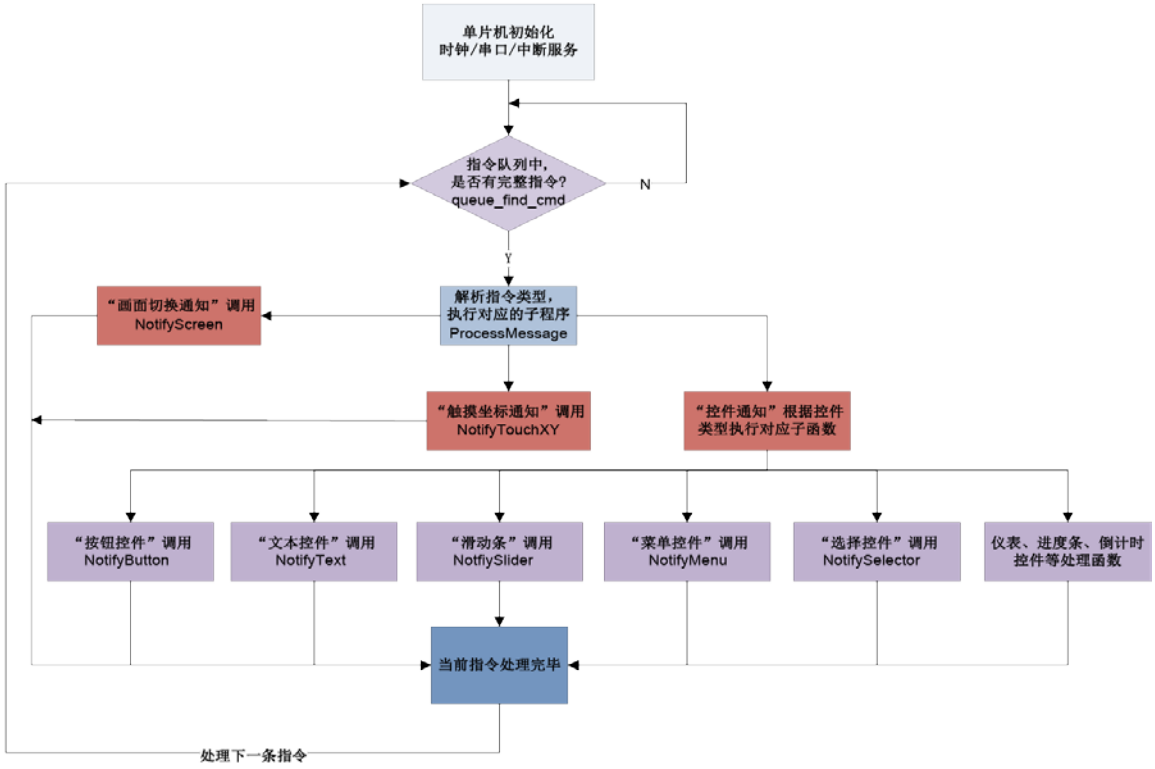
Parse and process flow is provided entirely by the drive instruction code. Examples program structure shown below 1-1 Below:



Map 1-1 Program Structure

2. Routine analysis

2.1 Work program flow chart



Map 2-1 Work program flow chart

2.1 Issued instructions serial interface screen

Whenever the operating screen on the serial port, the serial interface will send the screen corresponding to the instruction, such as switching the screen, text input control, the progress value change operation and the like will be issued corresponding to the control command value change notifications MCU .

2.2 instruction

When the serial port and screen MCU Through the serial port connection, the communication between them depend on the instruction. The instructions are for the convenience of our specifications and data exchange protocol-specific instructions can refer to various functions specific instruction set documentation.

2.2.1. Command parsing process

MCU After receiving the interrupt handler instruction in the instruction stored in the instruction buffer, the main Function while Cycle, queue_find_cmd It would have been detected instruction buffer, if the buffer command is received immediately after extraction; then call ProcessMessage Instruction parsing function and then calls the corresponding function of the type of data and command instruction as a parameter into a function call.

2.2.2 Receive instructions

MCU Interrupt receiving serial commands transmitted to the screen buffer

```
void USART1_IRQHandler (void) {
```

```
        if (USART_GetITStatus (USART1, USART_IT_RXNE)! = RESET) {

            uint8_t data = USART_ReceiveData (USART1); queue_push
            (data);                                          // The reception data buffer
        }}
```

2.2.3 cmd_queue Serial screen instruction queue

Instructions may also be understood as a data frame, the data frame is generally divided into three parts: header, a data portion, the end of the frame. The instruction queue is defined as a structure QUEUE, the queue structure member comprises a head end of the queue, and the data buffer queue. Call queue function is turned on when the MCU queue_reset Empty the buffer queue, to prevent errors in the data in the buffer. Screen send instructions to the serial port MCU Rear, MCU The serial interrupt function calls queue_push Extracting the data into the buffer queue, the queue after the present data buffer, queue_find_cmd The data buffer queue function calls queue_pop Then one by one put forward spliced into a complete instruction.

```
#define CMD_HEAD 0XEE                                    // Header
#define CMD_TAIL 0XFFFCFFFF                              // End of frame
typedef struct _QUEUE {

    qsize _head;                                         // Head of the queue
    qsize _tail;                                         // End of the queue
    qdata _data [QUEUE_MAX_SIZE];                        // Queue data buffer
} QUEUE;
static QUEUE que = {0,0,0};                              // Instruction queue
static uint32 cmd_state = 0;                             // End of frame detection state queue
static qsize cmd_pos = 0;                                // The current instruction pointer position
void queue_reset () {

    que._head = que._tail = 0;
  cmd_pos = cmd_state = 0;}

void queue_push (qdata _data)                            // Call to get instruction in the interrupt data
{
    qsize pos = (que._head + 1)% QUEUE_MAX_SIZE; if (! pos =
    que._tail)                                           // It did not reach the upper limit of the buffer zone
    {
        que._data [que._head] = _data;
    que._head = pos;}}


// A fetch data from the queue buffer
static void queue_pop (qdata * _data) {

    if (que._tail! = que._head)                          // Non-empty state
    {
        * _data = que._data [que._tail];
```

```
                que._tail = (que._tail + 1)% QUEUE_MAX_SIZE;}}


// Get the number of valid data queue
static qsize queue_size () {

        return ((que._head + QUEUE_MAX_SIZE-que._tail)% QUEUE_MAX_SIZE);}

qsize queue_find_cmd (qdata * buffer, qsize buf_len) {

        qsize cmd_size = 0;                                     // qsize : typedef unsigned char qdata;
        qdata _data = 0;                                        // qdata : typedef unsigned short qsize;
        while (queue_size ()> 0) {

                // A data fetch
                queue_pop (& _ data);
                 if (cmd_pos == 0 && _ data! = CMD_HEAD)        // The first instruction byte header must be, otherwise jump
                 {
                      continue;}

                if (cmd_pos <buf_len)                           // Prevent buffer overflow
                buffer [cmd_pos ++] = _data; cmd_state =
                ((cmd_state << 8) | _data);                     // Final stitching 4 Bytes, consisting of 32 Bit integer
                // At last 4 Byte comparison with the frame end, obtain the full frame
                 if (cmd_state == CMD_TAIL) {

                      cmd_size = cmd_pos;                       // Byte length instructions
                      cmd_state = 0;                            // Frame end redetection
                      cmd_pos = 0;                              // Reset instruction pointer
#if (CRC16_ENABLE)
                // Instructions to remove head and tail EE ,tail FFFCFFFF total 5 Bytes, only the data portion is calculated CRC
                      if (! CheckCRC16 (buffer + 1, cmd_size-5))        // CRC check
                            return 0; cmd_size - = 2 ; // Remove CRC16 ( 2 byte)

# Endif
                      return cmd_size;}}
            return 0;

                                                                // Did not form a complete frame
 }
```

Find out when a complete instruction frame, the size of the instruction will be returned, and then calls the function ProcessMessage Parse command:

```
size = queue_find_cmd (cmd_buffer, CMD_MAX_SIZE);
// size Greater than zero proved to be complete instruction
```

```
if (size> 0) {

        ProcessMessage ((PCTRL_MSG) cmd_buffer, size);                    // Resolution instructions
}
```

2.2.4. ProcessMessage Parsing command type

```
/ *!
*     \ Brief Message Handling Process
*     \ Param msg Pending instructions
*     \ Param size Instruction length
*     Macro definitions or msg See the instruction type structure main Header function # include "cmd_process.h"
*     This function is the classification of the data acquisition instructions, then switch Selecting a corresponding call control function and the type of instruction data passed as parameters
      of the function call.
* /
void ProcessMessage (PCTRL_MSG msg, uint16 size) {

      uint8cmd_type = msg-> cmd_type;                                     // Instruction type instruction
      uint8 ctrl_msg = msg-> ctrl_msg;                                    // Type instruction message
      uint8 control_type = msg-> control_type;                            // Instruction control type
      uint16 screen_id = PTR2U16 (& msg-> screen_id);                     // Instruction screen ID
      uint16 control_id = PTR2U16 (& msg-> control_id);                   // Directive controls ID
      uint32 value = PTR2U32 (msg-> param);                               // Numerical instructions
      switch (cmd_type) {

         case NOTIFY_TOUCH_PRESS:                                         // Touch screen Press
         case NOTIFY_TOUCH_RELEASE:                                       // Touch-screen release
               NotifyTouchXY (cmd_buffer [1], PTR2U16 (cmd_buffer + 2), PTR2U16 (cmd_buffer + 4)); break;

         case NOTIFY_WRITE_FLASH_OK:                                      // write FLASH success
               NotifyWriteFlash (1); break;

         case NOTIFY_WRITE_FLASH_FAILD:                                   // write FLASH failure
               NotifyWriteFlash (0); break;

         case NOTIFY_READ_FLASH_OK:                                       // Read FLASH success
               NotifyReadFlash (1, cmd_buffer + 2, size-6);               // Header removed frame end
               break;
         case NOTIFY_READ_FLASH_FAILD:                                    // Read FLASH failure
               NotifyReadFlash (0,0,0); break;

         case NOTIFY_READ_RTC:                                            // Read RTC time
               NotifyReadRTC (cmd_buffer [2], cmd_buffer [3], cmd_buffer [4], cmd_buffer [5], cmd_buffer [6]
                              cmd_buffer [7], cmd_buffer [8]);
               break;
         case NOTIFY_CONTROL:
```

```
{
    if (ctrl_msg == MSG_GET_CURRENT_SCREEN)                    // Picture ID Change Notification
    {
        NotifyScreen (screen_id);                              // Screen switching function mobilized
    }
  Else {

      switch (control_type) {

        case kCtrlButton:                                      // Button control
            NotifyButton (screen_id, control_id, msg-> param [1]); break; case
          kCtrlText:
                                                               // Text Control
            NotifyText (screen_id, control_id, msg-> param); break; case
          kCtrlProgress:
                                                               // Progress bar control
            NotifyProgress (screen_id, control_id, value); break; case
          kCtrlSlider:
                                                               // Slider controls
            NotifySlider (screen_id, control_id, value); break; case
          kCtrlMeter:
                                                               // Instrument Control
            NotifyMeter (screen_id, control_id, value); break; case
          kCtrlMenu:
                                                               // Menu control
            NotifyMenu (screen_id, control_id, msg-> param [0], msg-> param [1]); break; case
          kCtrlSelector:
                                                               // Picker
            NotifySelector (screen_id, control_id, msg-> param [0]); break; case
          kCtrlRTC:
                                                               // Countdown Controls
            NotifyTimer (screen_id, control_id); break;
      default: break;}} break;} default:

break;}}
```

2.2.5 Get the picture ID Interface and data update

MCU The interrupt function after receiving the "screen switch" instruction, the store instruction to the instruction buffer; in main Function while Cycle, queue_find_cmd Detects the instruction buffer command is received, and extracted; then called ProcessMessage Parsing function and an instruction type and calls the function corresponding to the instruction type data as a parameter; command functions received by the "screen switch", the processing will be called "screen switch" message function NotifyScreen To get the current picture in the function ID .

in main Function while Cycle UpdateUI Function, the screen used for timing the serial data update cycle.

```
// The main function of while cycle
while (1) {

    size = queue_find_cmd (cmd_buffer, CMD_MAX_SIZE);                    // Obtaining an instruction from the buffer
      if (size> 0) {

          ProcessMessage ((PCTRL_MSG) cmd_buffer, size);                 // Instruction processing
      }
     // timing 20 Ms updating data
     // Into an interrupt 10ms , timer_tick_count Value + 1,100 * timer_tick_count = 1s
    if (timer_tick_count% 2 == 0) {

        UpdateUI ();}}


void NotifyScreen (screen_id) {

    current_screen_id = screen_id;          // Open Screen switch notification in engineering configuration, record the current picture ID
  ......}.

void UpdateUI ()                                                         // Regularly updated data
{
    ......
    // Picture 6 timing 20ms Refresh data
    if (current_screen_id == 6) {

        Set_picMeterValue (6,2, test_value);          // Set the rotation angle of the current picture pointer control screen
        test_value + = 1; if
        (test_value> = 260)                           // Pointer from 0 To 260 Degrees rotation of the pointer co 260 degree
        {
            test_value = 0;}} ......}
```

2.3 MCU Routine functions to achieve

2.3.1. Serial routine main interface screen factory

Map 2-2 A main interface

Map 2-3 The main interface two

2.3.2. Set button is pressed

Map 2-4 Button interface

Click on the map 2-2 The main interface is a button to switch to button interface, as shown in 2-4 Shown, while the serial to screen MCU Two transport instructions, an instruction is a click of a button, a screen switching instruction is. MCU After receiving the instruction for

Resolution, please review the specific process directory 2.2.1. Command parsing process.

MCU After switching the screen resolution instructions, the call process "screen switch" function of the message NotifyScreen To get the current picture in the function ID . And calls the driver function SetButtonValue Set button control status values:

```
void NotifyScreen (screen_id) {

    current_screen_id = screen_id;                    // Open Screen switch notification in engineering configuration, record the current picture ID
    if (screen_id == 3)                               // Into the picture 3 After pressing a button
    {
        // Set the current screen controls 1 Value button, the button is 1 . Button value 0 To bounce, 1 For the press
        SetButtonValue (3,1,1);} .....}
```

2.3.3. Regularly updated text data

Map 2-5 Text interface

Click on the map 2-2 After the main interface of a text into the text interface, as shown in 2-5 , The transmitting microcontroller and the screen and click on instruction button control switch, MCU Will be resolved after receiving the instruction, the specific process, please see the directory 2.2.1. Command parsing process.

MCU After switching the screen resolution instructions, the call process "screen switch" function of the message NotifyScreen To get the current picture in the function ID Rear. in while Timing loop 20ms Call function update screen UpdateUI And then UpdateUI Driver function call SetTextInt32 , SetTextValue , SetControlBackColor Update text data interface:

```
void UpdateUI () {

    .....
  // Setting the display timing and text 20ms Refresh.
    if (current_screen_id == 4) {

        // Present current, temperature from 0 To 1000 Display cycle, from WordArt 0-999 Cycle through
        SetTextInt32 (4,6, test_value% 1000,1,1);          // Send data to the serial port to modify the current screen controls 6 The value
        SetTextInt32 (4,7, test_value% 1000,1,1);          // Send data to the serial port to modify the current screen controls 7 The value
```

```
                SetTextValue (4,1, " engine room 10 ");                    // Send data to the serial port to modify the current screen controls 1 The value
                test_value ++; if (test_value>
                = 1000) {

                    test_value = 0;}

                if (test_value> 0 && test_value <500)                      // more than the 0 Less than 500 Text displayed in red
                {
                    SetControlBackColor (4,6,0xF800);                      // Send data to modify the control panel serial port 6 Background color
                }
                else if (test_value> = 500)                                // more than the 500 Text was blue
                {
                    SetControlBackColor (4,6,0x00if);                      // Send data to modify the control panel serial port 6 Background color
                }}

        .....
    }
```

2.3.4 Regularly updated meter data

Map 2-6 Instrument Interface

Click on the map 2-2 A main interface of the meter into the meter interface, in FIG. 2-6 , The transmitting microcontroller and the screen and click on instruction button control switch. MCU Will be resolved after receiving the instruction, the specific process, please see the directory 2.2.1. Command parsing process.

MCU After switching the screen resolution instructions, the call process "screen switch" function of the message NotifyScreen To get the current picture in the function ID . in while Timing loop 20ms Call function update screen UpdateUI And then

UpdateUI Driver function call Set_picMeterValue Update instrument rotation angle:

```
    void UpdateUI () {

        .....
        // Instrument interface and timing 20ms Refresh rotation angle
        if (current_screen_id == 6)
```

```
{
    Set_picMeterValue (6,2, test_value);          // Set the rotation angle of the current picture pointer control screen
    test_value + = 1; if
    (test_value> = 260)                           // Instrumentation value 0 to 140 , Co-rotation of the pointer 260 degree
    {
        test_value = 0;}} ......}
```

2.3.5 The value of the text associated with the progress bar

Map 2-7 Slider interface

Click on the map 2-2 After a slider in the main interface, the interface into the slider, FIG. 2-7 Shown, when the drag the slider, the slider value is changed, while the MCU Transmitting the change instruction value slider. MCU Will be resolved after receiving the instruction, the specific process, please see the directory 2.2.1. Command parsing process.

MCU After parsing instructions, call ProcessMessage Instruction parsing function and calls the corresponding function of instruction type instructions and data carried as a parameter; Function herein as a "value change slider", the function calls

NotifySlider deal with:

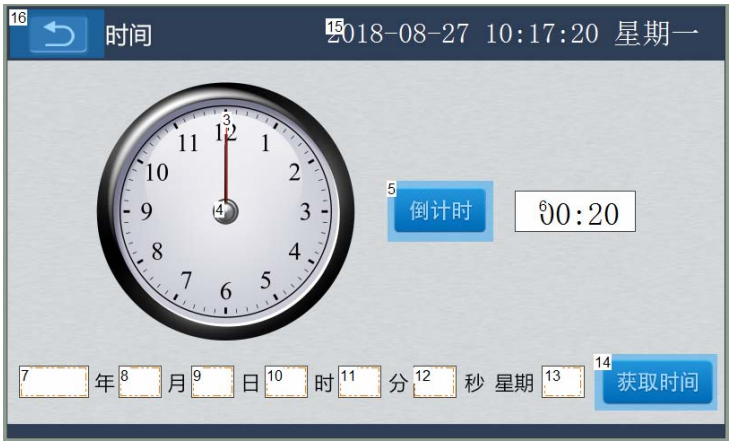
```
/ *!
*     \ Details When the slider to change (or call GetControlValue) When performing this function
*     \ Paramscreen_id Picture ID
*     \ Paramcontrol_id Controls ID
*     \ Param value value
* /
void NotifySlider (uint16 screen_id, uint16 control_id, uint32 value) {

    if (screen_id == 7 && control_id == 5)          // Slider Control
   {
        if (value <100 || value> 0) {
```

```
                SetProgressValue (7,4, value);          // Update the current screen progress bar control 4 The value
                SetTextValueInt32 (7,6, value);         // Update the current screen text control 6 The value
            }}

}
```

2.3.6 Acquisition time and time countdown



Map 2-8 Time Interface

Click on the map 2-2 The main interface after a time, into the interface time, as shown in 2-8 , The transmitting microcontroller and the screen and click on instruction button control switch, MCU Will be resolved after receiving the instruction, the specific process, please see the directory 2.2.1. Command parsing process.

MCU After switching the screen resolution instructions, the call process "screen switch" function of the message NotifyScreen To get the current picture in the function ID . Then while Cycle timing function calls UpdateUI Update the current screen ID Data, the function UpdateUI Each timing 500 Ms calls the driver function ReadRTC Send read screen within the screen to the serial port RTC Clock instruction, read the received serial screen RTC After the command to MCU When the serial transmission time BCD code. Instruction here is " RTC Time ", so call processing RTC The function NotifyReadRTC Press the button acquisition time, will MCU Acquired RTC Time displayed:

```
// in UpdateUI Add the following function in the code
// Time into the screen, the timing of each 500 Ms transmission time read RTC Instruction time, and each 10s The buzzer rang again
void UpdateUI () {

    .....
    if (current_screen_id == 8 && timer_tick_count% 50 == 0)
    {
        ReadRTC ();                                     // Send obtain RTC Time Directive
        if (sec% 10 == 0)                               // when 10s, 20s, ..... 60s When the buzzer sounds
        {
            SetBuzzer (250);                            // Buzzer
          }}}
  .....


void NotifyButton (uint16 screen_id, uint16 control_id, uint8 state)
```

```
{
        ......
    if (screen_id == 8) {

        if (control_id == 14) {

            SetTextInt32 (8,7, years, 1,1);                    // On display
            SetTextInt32 (8,8, months, 1,1);                   // Display month
            SetTextInt32 (8,9, days, 1,1);                     // Japanese display
            SetTextInt32 (8,10, hours, 1,1);                   // Display
            SetTextInt32 (8,11, minutes,                       // Display division
            SetTextInt32 (8,12, sec, 1,1);                     // Display seconds
            if (weeks == 1) {

                SetTextValue (8,13, " One");                   // Show Monday
            }}}


        ......
// Serial screen is sent to the MCU The serial port RTC Time BCD code
void NotifyReadRTC (uint8 year, uint8 month, uint8 week, uint8 day, uint8 hour, uint8 minute, uint8
                    second)
{
    sec = (0xff & (second >> 4)) * 10 + (0xf & second);            // BCD Decimal code turn, converts the second value
    years       = (0xff & (year >> 4)) * 10 + (0xf & year); months = (0xff &
    (month >> 4)) * 10 + (0xf & month); weeks
                = (0xff & (week >> 4)) * 10 + (0xf & week); days
                = (0xff & (day >> 4)) * 10 + (0xf & day); hours
                = (0xff & (hour >> 4)) * 10 + (0xf & hour); minutes = (0xff &
    (minute >> 4)) * 10 + (0xf & minute);
}
```

2.3.7 Play movies and play music

Map 2-9 Animated interface

Map 2-10 Audio Interface

Click on the map 2-2 After the main screen in a movie or music into animation 2-9 Animation or auto-play music after interface 2-10 Auto-play music after interface. MCU Is parsed after receiving the instruction to switch the screen, please see the specific directory 2.2.1. Command parsing process.

MCU After switching the screen resolution instructions, the call process "screen switch" function of the message NotifyScreen And calls the play the animation in the function driver functions AnimationStart Or call the driver function for playing music PlayMusic :

```
void NotifyScreen (screen_id) {

    current_screen_id = screen_id;                    // (Open Screen switch notification in engineering configuration) Record the current screen ID
    // Proceeds to screen AutoPlay GIF
    if (current_screen_id == 9) {

        AnimationStart (9,1);                         // Animation starts
    }
    // Into the music playback screen automatically
```

```
if (current_screen_id == 18) {

    // Hex audio paths and audio name suggests, you can use software visual TFT The conversion instruction assistant
    uint8 buffer [19] = {0x94,0x61, 0x3A, 0x2F, 0x73, 0x6F, 0x75, 0x6E, 0x64, 0x73, 0x2F,
                         0x30, 0x31, 0x2E, 0x6D, 0x70, 0x33};
    SetButtonValue (18,3,1);                    // Set the audio screen button controls 3 The value
    PlayMusic (buffer);                         // play music
}}
```

2.3.8 Display icon

Map 2-11 Icon interface

Click on the map 2-2 A main interface after an icon, the icon into the interface, as shown in 2-11 , The transmitting microcontroller and the screen and click on instruction button control switch, MCU Will be resolved after receiving the instruction, the specific process, please see the directory 2.2.1. Command parsing process.

MCU After switching the screen resolution instructions, the call process "screen switch" function of the message NotifyScreen And obtain the current screen function ID . in while A timing loop update screen function calls UpdateUI And then UpdateUI

Then call the driver function SetButtonValue Settings button 2,3,4,5 State, and calls the driver function AnimationPlayFrame Set to play the specified frame icon, the icon is displayed to achieve a circulating effect:

```
void UpdateUI () {

    .....
    // When the current picture is an icon interface, icon 40ms Turns display
  if (current_screen_id == 10) {

        if (timer_tick_count% 40 == 0 && icon_flag == 0)
          {
            SetButtonValue (9,5,0);                 // Settings button 5 State value
            SetButtonValue (9,2,1);                 // Settings button 1 State value
            AnimationPlayFrame (9,1,0);             // Displays the frame icon controls
            icon_flag = 1;                          // Flag
```

```
                    }
                    else if (timer_tick_count% 40 == 0 && icon_flag == 1) {

                        SetButtonValue (9,2,0);                    // Settings button 2 State value
                        SetButtonValue (9,3,1);                    // Settings button 3 State value
                        AnimationPlayFrame (9,1,1);                // Displays the frame icon controls
                        icon_flag = 2;}

                    else if (timer_tick_count% 40 == 0 && icon_flag == 2) {

                        SetButtonValue (9,3,0); SetButtonValue
                      (9,4,1); AnimationPlayFrame (9,1,2);
                      icon_flag = 3;}

                    else if (timer_tick_count% 40 == 0 && icon_flag == 3) {

                        SetButtonValue (9,4,0); SetButtonValue
                      (9,5,1); AnimationPlayFrame (9,1,3);
                      icon_flag = 0;}

}
.....}
```

2.3.9 Regularly updated curve data

Map 2-12 Sine wave

Map 2-13 Sawtooth

Click on the map 2-3 After the main interface of two curves, the curve into the interface, and the microcontroller and the screen instruction sending click button control switch, MCU Will be resolved after receiving the instruction, the specific process, please see the directory 2.2.1. Command parsing process.

MCU After switching the screen resolution instructions, the call process "screen switch" function of the message NotifyScreen And obtain the current screen function ID . Press button interface sine wave, MCU Receiving an instruction button is pressed, the button function is called NotifyButton And change the process to sine curve type, and a timing period 20 Ms update screen function UpdateUI , Call the driver function GraphChannelDataAdd Updated value of a sinusoidal curve, sawtooth if the button is pressed, then the following values sawtooth update:

```
void NotifyButton (uint16 screen_id, uint16 control_id, uint8 state) {

    if (screen_id == 11) {

        if (control_id == 2)                                    // Sine wave control
        {
            curves_type = 0;                                    // Sine curve type
        }
        else if (control_id == 3)                               // Sawtooth control
        {
            curves_type = 1;                                    // Sine curve type
        }}}


void UpdateUI () {

    // Real-time curve, sine wave array. timing 50ms Refresh data
    if (current_screen_id == 11 && timer_tick_count% 5 == 0) {

        if (curves _type == 0) {
```

```
// Sine array
uint8sin [256] = {} ;                                   // Refer to the specific code array
// Add data to a curve of a control data of the current picture
GraphChannelDataAdd (11,1,0, & sin (num), 1); num ++; if
(num> = 255) {


    num = 0;}}


else if (curves_type == 1) {

// Sawtooth array
  uint8 sawtooth [180] = {}                             // Refer to the specific code array
/ / Add data to a curve of a control data of the current picture
  GraphChannelDataAdd (11,1,0, & sawtooth (num), 1); num ++; if
  (num> = 180) {


    num = 0;}}}}
```

2.3.10 Picker

Map 2-14 Picker

Click on the map 2-3 Two main interface selection control, selection control to enter the interface, as shown in 2-14 Fig. Elected after a certain time, click the Save button MUC We will return the corresponding time period:

If selected 10 Point, the screen will Serial MUC Transmission instruction (13 No screen, select the control 1 ,select 10 point), MCU

After extraction instruction, call ProcessMessage Instruction parsing function and calls the corresponding function data and instruction type as a parameter; Here, because the received instruction is "change the control value selection", the function call processing selection control NotifySelector Gets the value of the selected control, then press the button on the acquired time value is determined and shows a selected time period:

```
/ * Brief Select control notification
*      \ Details When you select a control change, perform this function
*      \ Paramscreen_id Picture ID
*      \ Paramcontrol_id Controls ID
*      \ Param item Current options
* /
void NotifySelector (uint16 screen_id, uint16 control_id, uint8 item) {

        if (screen_id == 13 && control_id == 1) {

                Select_H = item;                                    // Gets Picker 1 The value
        }
        if (screen_id == 13 && control_id == 2) {

                Select_M = item;                                    // Gets Picker 2 The value
        }}

void NotifyButton (uint16 screen_id, uint16 control_id, uint8 state) {

        .....
        if (screen_id == 13 && control_id == 4) {

          if (Select_H> = 0 && Select_H <= 6)                       // 0 To 6 The period of time is the early morning hours
        {
                SetSelectorValue (13,3,0);}

        else if (Select_H> = 7 && Select_H <= 12)                   // 7 To 12 The period of time is the early morning hours
        {
                SetSelectorValue (13,3,1);}

        else if (Select_H> = 13 && Select_H <= 18)                  // 13 To 18 The period of time is the early morning hours
        {
                SetSelectorValue (13,3,2);}


        else if (Select_H> 18 && Select_H <= 23)                    // 18 To twenty three The period of time is the early morning hours
        {
                SetSelectorValue (13,3,3);}}
```

```
}
```

2.3.11 Trigger the warning and warning lifted

Map 2-15 data record

Click on the map 2-3 After the main interface of two data recording, data recording into the FIG. 2-15 After the interface, and single-chip instruction and click send button control to switch the screen, MCU Will be resolved after receiving the instruction, the specific process, please see the directory 2.2.1. Command parsing process.

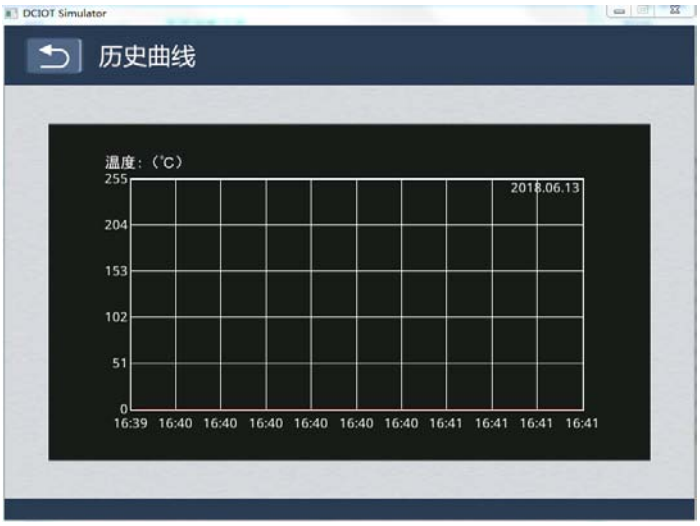
MCU After parsing the instruction, call processing "screen switch" function of the message NotifyScreen And it calls the driver function in the function Record_SetEvent Set trigger an alert, calls the driver function after a delay of two seconds Record_ResetEvent Issued a warning released:

```
void NotifyScreen (screen_id) {

    current_screen_id = screen_id;                  // Record the current picture ID , current_screen_id Global variables
    ......
    if (current_screen_id == 15)                    // Data records show
    {
        // Record_SetEvent (15,1,0,0); The third parameter is the value of the warning data record, a fourth parameter is releasing the warning time is 0 // When using the serial port screen time
        Record_SetEvent (15,1,0,0);
        Record_SetEvent (15,1,1,0);
        Record_SetEvent (15,1,2,0);                 // Setting data logging control, set the warning value
        Record_SetEvent (15,1,3,0);
        Record_SetEvent (15,1,4,0);
        Record_SetEvent (15,1,5,0);
        Record_SetEvent (15,1,6,0);
        Record_SetEvent (15,1, 7,0);
        delay_ms (2000);                            // Delay two seconds
        Record_ResetEvent (15,1,0,0);
        Record_ResetEvent (15,1,1,0);               // Setting data logging control, set to lift a warning
```

```
Record_ResetEvent (15,1,2,0);} .....}
```

2.3.12 Historical curve



Map 2-16 History curve interface

Click on the map 2-3 After the main interface II history button control curve, the curve entry history screen, FIG. 2-16 , The transmitting microcontroller and the screen and click on instruction button control switch, MCU Will be resolved after receiving the instruction, the specific process, please see the directory 2.2.1. Command parsing process.

MCU After switching the screen resolution instructions, the call process "screen switch" function of the message NotifyScreen And obtain the current screen function ID . in while A timing loop update screen function calls UpdateUI And then UpdateUI

Timing 1 Second drive function is called once HistoryGraph_SetValueInt8 Update History curve values:

```
void UpdateUI () {

    .....
    // Duration curve, sine wave array
    if (current_screen_id == 16)                                    // Historical curve controls the sampling period 1s one point.
    {
        if (curves == 0) {

            uint8i, Sendsin [1] = {0};                              // Initialized or cleared 0
            HistoryGraph_SetValueInt8 (16,1, & sin [num], 1); // Adding historical data curve
            num ++; if (num> =
            255) {

                num = 0;}}


        if (curves == 1) {
```

```
                uint8sawtooth [180] = {};                                   // Sawtooth array Refer to the program
                // Adding historical data add one curve
                HistoryGraph_SetValueInt8 (16,1, [num] & sawtooth, 1); num ++; if
                (num> = 180) {



                    }}}} .....;
num = 0.
```

3. Configuration

3.1 Visual TFT Configuration

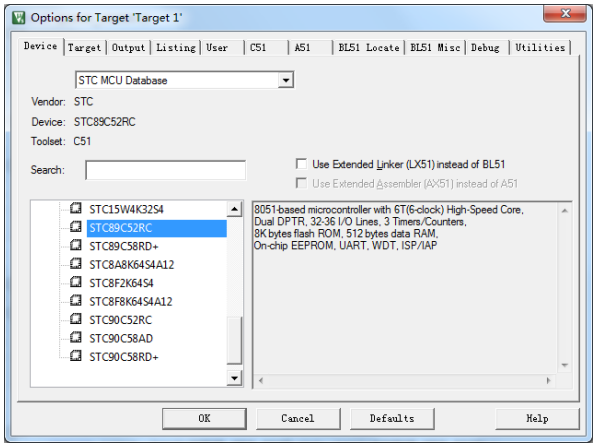
VisualTFT in CRC16 Respective settings, as FIG. 3-1 As shown :(Note: You can not open)

Map 3-1 TFT Configuration

VisualTFT The baud rate is set to be consistent with the microcontroller initialization of engineering.

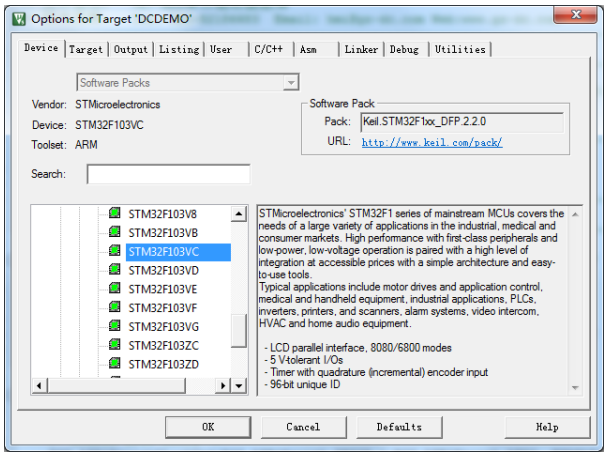
3.2 KEIL develop software

Keil C51, Compatible microcontroller C Language software development system. Keil C51 of STC Libraries need STC-ISP The software adds. (In order to avoid unnecessary BUG , So try to choose the right type).



Map 3-2 C51

Keil MDK , MDK-ARM Software-based Cortex-M , Cortex-R4 , ARM7 , ARM9 The processor device provides a complete development environment. FIG configured as follows 3-3 (Corresponding to the selected model)



Map 3-3 STM32

4. How to migrate to other MCU series

Our routine is used by the microcontroller STM32F103VCT6 with STC89CX Series, other series if the customer needs the transplant works, it is recommended to copy all the source files, and then modify the following several places as needed:

1. SCM need to modify the corresponding model for their own header files such as:

STC89CX Series of headers ------ # include <reg52.h> STC15F2K60S2 The header file

---- # include <STC15F2K60S2.h>

2. **hmi_user_uart.c - serial port initialization and data transmission (according to the user's own MCU Data set)**

```
void UartInit ();                    // Initialization of a communication port, set the baud rate and other
void SendChar (char ch);             // Transmitting a byte through the serial port
```

3. Serial data reception interrupt processing (Here are examples only, mainly to see the user's MCU)

```
// 8051 Serial data receiving process under the following internet
// Serial data received by queue_push End of the queue of instructions added
void serial () interrupt 4 {

    if (RI)                          // Window data received
    {
        RI = 0;
        queue_push (SBUF); // Pressed into the instruction buffer
     }}
```

4. Serial data reception process shown in the following STM32 platform:

```
void USART1_IRQHandler (void) {

    if (USART_GetITStatus (USART1, USART_IT_RXNE)! = RESET) {

        uint8_t data = USART_ReceiveData (USART1); queue_push
   (data);}}
```