

## **Communication protocol between LED controller and PC or other devices**

### **1. Occasions**

This protocol works for led controller communicates with PC or other devices, which will realize the point-to-point communication and also could support the PC or other devices communicate with one or more led controllers via network.

### **2. How to build communicate with led controller?**

PC or other devices are the Server side while the led controller is the customer side.

The server side adopts TCP communication protocol to build up a socket connection, default monitor port is 31297.

The led controller will build up a reliable TCP transport interface channel with server side automatically via Domain name or defined IP address.

Connect success and report its identification information then it will enter Receiving mode, Server side should send a heart beating package to led controller per certain time to maintain this interface channel. If the led controller does not get heart beating package three times continuously then will be time out and will re-connect.

Command Content of report and heart beating package, please refer to Definition part.

# Communication protocol between Led controller and PC or other devices

Domain name fits for 3G, Internet or not-static IP address.

## 3. Rules

3.1 When PC or other devices communicates with led controller via UDP/TCP transport layer protocol, the led controller can compatible with both and will receive data from PC or other devices and it adopts standard socket interface programming.

UDP use 31296 port, transport data not very stable, PC or other devices adopts announcement call mode to check one or more led controller status under the same network segment.

TCP use 31297 port, transport data point-to-point reliable, announcement call is not allowed, led controller as customer side while PC or other devices as server side. Server side will communicate with multi-customer side at the same time.

TCP use 31299 port, transport data point-to-point reliable, announcement call is not allowed, led controller as Server side while PC or other devices as customer side. Customer side will communicate with multi-server side at the same time.

3.2 If the protocol needs hexadecimal data then should add "0x" to distinguish decimal data. For example, hexadecimal 0xFF stands for 255 decimal.

3.3 Multi-byte data type adopts little-endian data alignment mode. For example, unsigned integer 0x87654321, it takes up 4 bytes in storage, so the order of data address from low to high like this 0x21、0x43、0x65、0x87, besides, the communication data package will send data according to this order, sending low byte 0x21 then 0x43 then 0x65 and last one 0x87.

3.4 Character string data must stored and close with 0x00('0') , add 1 for valid length. For example, valid length of character string "M30-507-00888" is 14, the last bytes should be 0x00.

3.5 Single quotes information in this document means the ASCII code. For example, 'A' means A's ASCII code, 65 decimal, 0x41 hexadecimal.

3.6 Check code of example part in this document was replaced by "??", decided by specific algorithm checkout.

# Communication protocol between Led controller and PC or other devices

## 4. Structure

The protocol contains seven parts: Sync-Header, command type, reserved field, main body length, main body content, checkout, please see the table in below:

Sync-header	Command type	Reserved field	Main body length	Main body content	Checkout
3 bytes	1 bytes	4 bytes	4 bytes	(main body length)	1 byte

Sync-header: 3 bytes, unsigned character type

To identify the start of data package

Value: three continuous fixed value sequence: 0x7E 0X7E 0X55

Command type: 1 byte, unsigned character type

Data package's command type

Please refer to following detailed explanation

Reserved field: 4 bytes

Main body length: 4 bytes, unsigned integer (number range: 0x0000 0000-0x0000 07D0) (within 2048)

Valid data bytes of “main body content”

Main body content: its byte decided by main body length, data content refers to following detailed explanation

Checkout: 1 byte, unsigned character type

Be used for the receiver side check whole data package correctly.

Add all of data (starting from “command type” field to the end of “main body”) in bytes and then get the negation.

# Communication protocol between Led controller and PC or other devices

## 5. Detailed explanation

### 5.1 General response of led controller side protocol

Led controller side will response command receiving, operating success or operating failed without specific annotations, detailed definition in below:

#### 5.1.1 The first type of general response, it is used for command setup success or failed

Data type: PC or other devices send command's "data type" add 0x40. For example, PC or other devices side send out 0x80 "data type" then led controller side response 0xC0

Main body length: 0x0000 0004

Main body content: unsigned integer: operating result

Operating success: 0x00000000

Operating failed: 0x00000001

#### 5.1.2 The second type of general response, it is used for command operating success or failed with subcommand.

Data type: PC or other devices send command's "data type" add 0x40. For example, PC or other devices side send out 0x80 "data type" then led controller side response 0xC0

Main body length: 0x00000008

Main body content: segment one unsigned integer: subcommand content:

It's same with the first segment unsigned integer of command main body data sending from Sender side.

Segment two unsigned integer: operating result:

# Communication protocol between Led controller and PC or other devices

---

Operating success: 0x00000000

Operating failed: 0x00000001

5.1.3 The third type of general response, this response type is not very much with current products, it mainly for compatible with some old command response functions

Data type: PC or other devices send command's "data type" add 0x40. For example, PC or other devices side send out 0x80 "data type" then led controller side response 0xC0

Main body length: decided by bytes number of main body content

Main body content: character string type

Operating success: "Success!"

Operating failed:"Error!"

# Communication protocol between Led controller and PC or other devices

## 5.2 Get working status of led controller side

PC sends: data type: 0x17  
Main body length: 0x0000 0000  
Main body content: no

Example: check the display status of led controller side in below:

7E 7E 55	// sync-header
17	// data type
00 00 00 00	// reserved
00 00 00 00	// main body length
??	// check code

Led controller side response: data type: 0x57

Main body length: it will change according to main body content length

Main body content: segment one unsigned integer: led controller side length (programs size that will send should be exactly same as that of the led controller side)

Segment two unsigned integer: led controller side height (programs size that will send should be exactly same as that of the led controller side)

Segment three unsigned integer: Boolean type working status: to identify each working status

according to bit, details in below:

```
Enum    _WORK_STATE_BIT_MAP
{
    WORK_STATE_PALY_ERROR,           // D0: 0—play normal, 1—program abnormal.
    WORK_STATE_DISPLAY_ERROR,        // D1: 0—display normal, 1—hardware display abnormal.
    WORK_STATE_USB_DICK_PLAY,        // D2: 1—display from U-disk.
    WORK_STATE_PROGRAM_UPGRADING,    // D3: 1—program updating.
```

# Communication protocol between Led controller and PC or other devices

```
WORK_STATE_SYSTEM_UPGRADING, // D4: 1—system upgrading.
WORK_STATE_TESTING,          // D5: 1—system testing.
WORK_STATE_POWER_OFF,        // D6: 1—power off.
WORK_STATE_PROGRAM_SIZE,     // D7: 1—program size is not the same with controller side.
WORK_STATE_TIMER_POWER,      // D8: 1—has timer switch power
WORK_STATE_TIMER_LIGHT,      // D9: 1—has timer light
WORK_STATE_TIMER_PROGRAM,    // D10:1—has timer program
ACK_VALUE_WORK_STATE_INTERNAL_PROGRAM, // D11:1—default internal program will be displayed
only when customer's program abnormal
WORK_STATE_NUM                // Max:D31.
};
```

Segment four unsigned integer: disk size (KB) (programs that will send should smaller than disk size)

Segment five unsigned integer: version number (programs that will send should keep with this version)

Segment six unsigned integer: system reserved (some bytes)

Structure definition:

```
typedef struct _SYSTEM_WORK_STATE
{
    UINT    uiScreenWidth;        // led controller side length
    UINT    uiScreenHeight;       // led controller side height
    UINT    uiWorkState; // Boolean type working status; refer to enum definition “_WORK_STATE_BIT_MAP”;
    UINT    uiDiskSize_KB;        // disk size (KB);
    UINT    uiVersion;            // version number
    UINT    Reserved[n];         // reserved
}SYSTEM_WORK_STATE,* P_SYSTEM_WORK_STATE;
```

Example: response of checking display status of led controller side in below:

7E 7E 55	// sync-header
57	// data type
00 00 00 00	// reserved
?? 00 00 00	// main body length, decided by main body content

# Communication protocol between Led controller and PC or other devices

40 01 00 00	// led controller side length 320.	// main body content
A0 00 00 00	// led controller side height 160.	
?? ?0 00 00	// Boolean type working status, refer to enum definition”_WORK_STATE_BIT_MAP”;	
95 0F 07 00	// disk size 462741KB, nearly 451MB.	
80 00 00 00	// current led controller side version is 8.	
...	// system reserved several bytes	
??		// check code

## 5.3 Setup led controller side size command

PC sends      data type: 0x1C

Main body length: 0x0000 0008 (8 bytes)

Main body content: first segment unsigned integer: led controller side length

Second segment unsigned integer: led controller side height

Structure definition:    typedef struct    \_COMMAND\_PARAMS\_SCREEN\_SIZE  
                         {            UINT    uiScreenWidth;                            // led controller side length  
                                    UINT    uiScreenHeight;                        // led controller side height  
                         }COMMAND\_PARAMS\_SCREEN\_SIZE,\* P\_COMMAND\_PARAMS\_SCREEN\_SIZE;

Example:

7E 7E 55		// sync-header
1C		// data type
00 00 00 00		// reserved
08 00 00 00		// main body length
80 00 00 00	// led controller side length 128.	// main body content
40 00 00 00	// led controller side height 64.	
??		// check code

Led controller side response:    data type: 0x5C      the first type of general response



# Communication protocol between Led controller and PC or other devices

## 5.4 Select single sub program to display

PC or other devices will upload content of program steps to led controller side, without this command then led controller side will display all programs automatically. After receiving the command, the led controller side will display sub programs of the current one that has been selected in loop. Also could cancel selection display mode via this command and restore to normal display mode. This status will not be saved after power off. It will restore to normal display mode after power on.

PC sends      data type: 0x97

                Main body length: 0x0000 0004 (4 bytes)

                Main body content:   first paragraph unsigned integer: unsigned integer, select the program number that will be display

  0x00000000: cancel the selected one and restore to normal display

  Others: program number that need to be displayed

Example: command of led controller choosing program3 in below:

7E 7E 55

97

00 00 00 00

04 00 00 00

03 00 ... 00

??

// choose and display program 3.

// sync-header

// data type

// reserved

// main body length

// main body content

// check code

Led controller side response:   data type: 0xD7; third type of general response;

# Communication protocol between Led controller and PC or other devices

## 5.5 Turn/off led controller side command

To turn on/off led sign via this command, it will be black when turn off while other modems keep working normally. The led controller system need about 20 seconds to restore working normally after turn on.

PC sends: data type: 0x10

Main body length: 0x0000 0004 (4 bytes)

Main body content: first segment unsigned integer: parameter is 0 for turn on/off led sign. 0---turn off, 1—turn on

Led controller side response: data type: 0x50 first type of general response

# Communication protocol between Led controller and PC or other devices

## 5.6 Setup system time command

PC sends            data type: 0x11  
Main body length: 0x0000 0004 (4 bytes)  
Main body content: segment one unsigned integer: year, 1900-2100  
                     segment two unsigned integer: month, 1-12  
                     segment three unsigned integer: date, 1-31  
                     segment four unsigned integer: hour, 0-23  
                     segment five unsigned integer: minute, 0-59  
                     segment six unsigned integer: second, 0-59  
                     segment seven unsigned integer: week, 1-7, Sunday is 7

Structure definition: typedef struct \_DATE\_TIME\_CLOCK  
                     {        UINT        uiYear;        //year, 1900~2100.  
                              UINT        uiMonth;     //month, 1~12.  
                              UINT        uiDay;        // date, 1~31.  
                              UINT        uiHour;       // hour, 0~23.  
                              UINT        uiMinute;    //minute, 0~59.  
                              UINT        uiSecond;    // second, 0~59.  
                              UINT        uiWeek;     // week, 1~7 Sunday is 7  
                     }DATE\_TIME\_CLOCK,\* P\_DATE\_TIME\_CLOCK;

Led controller side response: data type: 0x51; first type of general response

# Communication protocol between Led controller and PC or other devices

## 5.7 Control controller side light command

PC sends:     data type: 0x12  
              Main body length: 0x000 0004 (4 bytes)  
              Main body content: segment 1 unsigned integer: light value, 1 (dimmest)—8 (lightest)

Led controller side response:   data type: 0x52,   first type of general response.

# Communication protocol between Led controller and PC or other devices

## **6. Send program**

Xixun Company doesn't provide protocol for sending program, as it is too complicated. We provide FTP solution for sending program.

# Communication protocol between Led controller and PC or other devices

## 7. Command of sending program via FTP

Company or other devices download the content of multi-program pages into led controller side, if no this command, then led controller will auto display all programs. But after getting this command, led controller will display the sub-program of the current selected program page in loop. Also can cancel this display mode via this command and restore to normal display.

This status will not restore after power off, it will display programs normally after restart.

PC sends: data type: 0x5;

Main body length: 0x0000 0004 (4 bytes)

Main body content: segment one unsigned integer: unsigned integer, to select the program numbers that want to display

0x00000000: cancel the selected program and restore to normal display

Other: program number that needs to display

Example: command of display program 3 :

7E 7E 55

05

00 00 00 00

0C 19 00 00

00 00 00 00

// FTP type, must be 0

00 00 00 00

// 324bytes, must be 0

// sync-head

// data type

// reserved

// main body length 412

// main body content

// main body content

# Communication protocol between Led controller and PC or other devices

00 00 00 00	//program size	// main body content
09 00 00 00	//main version	// main body content
37 B7 32 01	//sub version	// main body content
?? 00 00 00	//program's width pixels	// main body content
?? 00 00 00	//program's height pixels	// main body content
?? ??.... ?? ??	//program name ,64bytes	
??		// check code

Response of led controller: data type: 0x45; the first type of general response