Communication Protocol

1, Basic transmission protocol

Control software and display panel for serial, network communication, are based on a certain transmission protocol, the protocol can detect whether there is an error in the transmission, the display panel to receive the wrong data discarded.

The transport protocol defines the format of the packet as: Identifier + Packet Length + Packet Count + Data + Checksum

Identification word (2Byte): The beginning of a packet of data, set to "EB 90".

Packet Length (2Byte): The total length of a packet of data, including the identification word to the check word. Unsigned short type, low byte in the former high byte in the post.

Packet Count (2Byte): The identifier of a packet of data. It is used to check whether the command sent and the response received are corresponding. The packet received by the display panel will be returned to the packet, and the packet count will not be changed. Unsigned short type, low byte in the former high byte in the post.

Data (0-514Byte): the actual transmission of data, the length from 0 bytes to 514 bytes variable.

CRC16 (2Byte): CRC16 check value of data packet (not including 2 bytes here), unsigned short type, **high byte precedes low byte**. The CRC16 value for the entire packet (including 2 bytes here) should be zero.

For example, one packet of data is as follows: EB 90 0B 00 00 80 07 01 00 A3 AF

Identification word (2Byte): EB 90

Packet length (2Byte): 0B 00

Packet Count (2Byte): 00 80

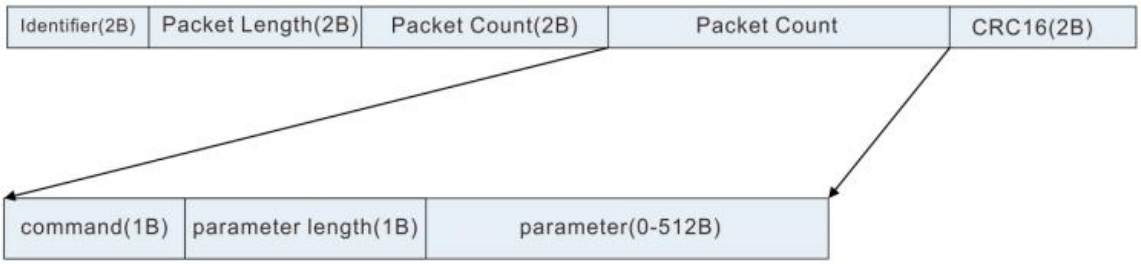
Data (0-514Byte): 07 01 00

CRC16（2Byte）： A3 AF

```
#define   ICE_MAX_DATA_SIZE      514
typedef struct kice
{
      unsigned short sync;
      unsigned short size;
      unsigned short id;
      unsigned char  data[ICE_MAX_DATA_SIZE + 2];
}kice_t;
```

```
typedef struct kice_command
{
    unsigned char  command;
    unsigned char  arg_size;
    unsigned char  arg[ICE_MAX_DATA_SIZE - 2];
}kice_command_t;
```

2. Command Protocol



Basic transmission protocol data, there is also a convention, the format is: command + parameter length + parameter.

The commands supported by the display panel and the return values are as follows:

```
enum
{
    GSM_CMD_CONFIG_PRINT        = 0x05,   // Print settings
    GSM_CMD_CONFIG_SAVE         = 0x06,   //Save the settings
    GSM_CMD_SCENE_PRINT         = 0x07,   // Print the scene
    GSM_CMD_SCENE_LOAD      = 0x08,   //Load the scene
    GSM_CMD_SCENE_SAVE      = 0x09,   //Save the scene
    GSM_CMD_PORT_INIT           = 0x0A,  //Port initialization
    GSM_CMD_PORT_SWITCH         = 0x0B,  //Port Switch
    GSM_CMD_PORT_MAP            = 0x0C,  //Port Map
    GSM_CMD_PORT_CONFIG         = 0x0D,  //Port Config

    GSM_CMD_ACK_SUCCESS         = 0x10,   //Respond executed successfully
    GSM_CMD_ACK_DATA                = 0x20,  (or command) / response succeeds, and returns the data

    GSM_CMD_ACK_INVALID         = 0x30,   // Respond    illegal command
    GSM_CMD_ACK_ERROR           = 0x40,   //Respond Parameter error
};
```

■Print the scene（GSM_CMD_SCENE_PRINT）

Command: 0x07

Participation Length: 1

Parameters: The number of the scene to be printed, 0-11 is valid

Display Panel Returns:
Command: 0x07 + 0x20
Length: sizeof (sw_state_t)
Parameters: sw_state_t

```
#define    GSM_MAX_TOTAL            96
typedef struct sw_state
{
      unsigned int    flag;       // 0xEB9055AA
      unsigned char  total;       // Total ports
      unsigned char  input;       // Input ports
      unsigned char  output;     // Output ports
      unsigned char group[GSM_MAX_TOTAL];// The configuration of each port group is described in ①。
}sw_state_t;
```

① Each group of values in the group represents the group information of the corresponding port. This information can be used to know the mapping between input and output ports.

Take 8 in 4 out as an example, then:
Group[0] = 1
Group[1] = 2
Group[2] = 3
Group[3] = 4
Group[4] = 5
Group[5] = 6
Group[6] = 7
Group[7] = 8
Group[8]。。。Group [11] can take the value 0-8.

Assuming Group [8] = 1, Group [9] = 1, Group [10] = 8, Group [11] = 0
Port 9, 10 and port 1 in the same group, ie 1 input video will be displayed in terminal 9;
Port 11 and port 8 in the same group, that is, 8 input video will be displayed in the terminal 11;
If the group number of port 12 is 0, it will not be displayed.

■Load the scene（GSM_CMD_SCENE_LOAD）
Command: 0x08
Participation Length: 2
Arguments: Load scene state (numbered Byte [1]) into scene (numbered Byte [0]). Byte [0] must be equal to 0, and Byte [1] must be 0-11.

Display Panel Returns:
Command: 0x08 + 0x10
Participation Length: 0

parameter:

■Save the scene（GSM_CMD_SCENE_SAVE）
Command: 0x09
Participation Length: 2
Arguments: Saves the scene (numbered Byte [1]) to a scene (numbered Byte [0]). Byte [1] must be 0 and Byte [0] is 0-11.

Display Panel Returns:
Command: 0x09 + 0x10
Participation Length: 0
parameter:

■Port initialization（GSM_CMD_PORT_INIT）
Command: 0x0A
Participation Length: 2
Parameters: Configured as Byte [0] inputs, Byte [1] outputs.

Display Panel Returns:
Command: 0x0A + 0x10
Participation Length: 0
parameter:

■Port Switch（GSM_CMD_PORT_SWITCH）
Command: 0x0B
Participation length: greater than or equal to 1
parameter:

Byte [0]: The input port to be switched (starting from 0). Valid values are: All input port numbers

Byte [1] .. Byte [n]: To switch to the output port of the input port (numbered starting from 0), valid values are for all output ports

Display Panel Returns:
Command: 0x0B + 0x10
Participation Length: 0
parameter:

■Port Map（GSM_CMD_PORT_ MAP）
Command: 0x0C
Participation length: greater than or equal to 1
parameter:
Byte [0]: Input port to be mapped (numbered from 0). Valid values are: All input port numbers
Byte [1] .. Byte [n]: The output port to be mapped to the input port (numbered from 0), valid values are all output port numbers

Display Panel Returns:
Command: 0x0C + 0x10
Participation Length: 0
parameter:

The difference between port switching and port mapping is that after the port mapping is performed, the original corresponding relationship of the input port is cleared. Assuming that the current 4 into 4, the state is as follows:
1 -> 5 6
2 -> 7
3 -> 8
4 ->
If you execute SWITCH 1 7, the status is as follows:
1 -> 5 6 7
2 ->
3 -> 8
4 ->
If MAP 1 7 is executed, the status is as follows:
1 -> 7
2 ->
3 -> 8
4 ->

The following commands are rarely used
■Print settings（GSM_CMD_CONFIG_PRINT）
Command: 0x05
Participation Length: 0
Parameters: None

Display Panel Returns:
Command: 0x05 + 0x20
Size: sizeof (global_config_t)
Parameters: global_config_t

```
typedef struct global_config
{
    unsigned int    flag;       // 0xEB9055AA
    unsigned char  hardware_version[2];     // hardware version
    unsigned char  firmware_version[2];     // Firmware version
    unsigned char  sn[4];      // The serial number is not guaranteed to be unique

    unsigned char         gateway[4];     // gateway
```

```
    unsigned char       subnet[4];      // subnet
    unsigned char       src_mac[6];     // MAC address
    unsigned char       src_ip[4];      // IP address
    unsigned short udp_port;// UDP port
    unsigned short tcp_port;        // TCP port
                                        // The above is read-only
    unsigned char  total;       // Total port
    unsigned char  input;       // Input port
    unsigned char  output;      // Outport
    unsigned char  rsv0;            // Reserved
}global_config_t;
```

■Save Settings（GSM_CMD_CONFIG_SAVE）
Command: 0x06
Size: sizeof (global_config_t)
Parameters: global_config_t

Display Panel Returns:
Command: 0x06 + 0x10
Participation Length: 0
parameter: