Game keyboard communication protocol

table of Contents

Page 3

2

Page 4

Chapter 1ROM Space Allocation

The ROM space is mainly divided into 5 areas, including button definition area (A area), basic information area (B area), LED special effect area (C area), game Mode area (D area) and macro definition area (E area), in which game mode area, button definition area, backlight definition area, and macro definition area (E area) are changing Corresponding updates are also required during Profile. The memory block address allocation is shown in the following table. The AP only needs to be partitioned CMD and the AP is 1 PAGE (Ex: EEPROM 64Bytes/1Page) is used as the minimum data amount for relative address update, and the absolute address in ROM is processed by FW.

	Area allocation		**KB Table Size (Bytes)**
Zone A	Basic information area (128 Bytes)		128
Zone B	Game mode area	Profile	64
Zone C	Button definition area	Profile	640 (Note 1)
Zone D	LED special effect area	Profile	960 (Note 2)
E area macro definition data area			400 ~ 128K

Note 1: The button definition area is divided into: For details, please refer to (button definition area)

KB	576 Bytes (programmable key 144keys * 4Bytes)	64Bytes reserved

Note 2: The LED special effect area is divided into: For details, please refer to (LED special effect area)

KB	320 Bytes (non-customized 20 types Special effects* 16Bytes)	576 Bytes (custom 144 LEDs * 4Bytes)	64Bytes LED current lighting parameters

3

Page 5

1.1 Area **A --** Basic equipment information **(128 Bytes)**

The basic equipment information mainly includes setting the current working scenario mode, PID, VID, product production information, etc., to facilitate later management, control and maintenance, a
The 16-31 bytes are the serial ID number, which is only filled in by a specific AP when the device is produced (Ex: Create this Table in the FW and update it with USB ISP).
No longer changes, representing the factory information of the device.

I. MCU F/W update, in addition to the original VID/PID judgment of the USB ISP, it is also necessary to read the basic information of the device for judgment and confirm the product status.

II. The size of macro data that can be stored by the Device upload determines how much macro size the AP can store.

Address [Byte]	Content	Description
0	Device ROM size	Total space of the device (unit: KB)
1	Macro space size_L	The size of the space that can store the macro data (64 byte is 1 unit)
2	Macro space size_H	The size of the space that can store the macro data (64 byte is 1 unit) (Not including checking Page)
3	Reserve	
4	VID_L	Device VID_L
5	VID_H	Device VID_H
6	PID_L	Device PID_L
7	PID_H	Device PID_H
8	VERSION_L	Device firmware version_L
9	VERSION_H	Device firmware version_H
10~128	Reserve	

4

Page 6

1.2 Area **B -** Game Mode Definition Area (**64 Bytes**)

The game mode area has a total of 64 Bytes, which records the function status of the Game Profile.

1.2.1 Game Profile

position	Features	range	Device description	Initial value
0x00	hot key	0x00-0x0A KB	FN+0~9 number keys and disable, a total of 10 shortcut keys to switch the emotion Environment mode 0~9 (This function is only valid when AP is connected)	0x00 (Disabled)
0x01	Game mode function 1	0x00-0x01 KB	Lock Windows key function; On: 0x01, Off: 0x00	0x00
0x02~0x3F reserved				

1.3 Area **C -** Button Definition Area

The main purpose of the key definition area is to let all keys have the flexibility to change their functions (any key of EX: KB is set as the left mouse button), and each key is set to 4 Bytes defines its key function. The format of 4 Bytes storage is as follows:

【Page + Page Description 1 + Page Description 2 + Page Description 3 】---------- Key definition format

Notice: Keyboard/mouse, there are different SIZE plans in the button definition area. (ROM space configuration)

keyboard
The key definition area is 576 Bytes, and the function of a key is defined by 4 Bytes. 576 Bytes can support up to 144 Keys for key customization.
The key addresses are arranged in order according to RC Table 0.0, 0.1...7.17, a total of 144 (RC Table: Row.Column).
Ex: Address 0x00~0x03 stores the button definition of Game Profile button 0.0.

1.3.1 Key definition format description **:**

The key definition format on KB/MS is as follows:

[Page + page description 1 + page description 2 + page description 3]

Page can be divided into the following description settings. Different categories have different definitions in page descriptions 1~3:

Page classification	Value	Description
Default Page	0x00	Defined as default
Mouse Page	0x01	Defined as L/M/R/4/5/roll up/roll down
Keyboard Page	0x02	Defined as keyboard Standard and Modifier keys
ConsumerKey Page	0x03	Defined as Consumer Key
SystemKey Page	0x04	Defined as System Key
Extra Function Page	0x05	Defined as special function keys

Macro Page	0x06	Defined as Macro Key
Multi-Key Tap Page	0x07	Keystroke

Page 8

1.3.2 Mouse Page Definition

Byte Map	Descriptor	Value
Byte [0]	Page	(0x01) Mouse Page
Byte [1]	Section	(0x01) Type L/M/R/4/5
		(0x03) Wheels
Byte [2]	Mouse behavior 1	(0x01) L
		(0x04) M
		(0x02) R
		(0x08) Page Back/Key 4
		(0x10) Page forward/Key 5
		(0xFF) roll down
		(0x01) roll up
Byte [3]	Keep	Keep

1.3.3 Keyboard Page Definition

Note: The second and third bytes store hot keys (ctrl/shift/alt/win) and normal keys respectively, only one byte has content representing a single key, and both bytes have The content is a combination key.

Byte Map	Descriptor	Value
Byte [0]	Page	(0X02) Keyboard Page
Byte [1]	Modifier Key code	0x01/0x02/0x04/0x08/0x10/0x20/0x40/0x80
Byte [2]	Standard key code	KEY_CODE reference attachment
Byte [3]	Keep	Keep

See Appendix 1 for the KEY CODE of the keyboard page.

1.3.4 ConsumerKey Page Definition

Byte Map	Descriptor	Value
Byte [0]	Page	(0X03) Consumer Key Page
Byte [1]	Key Code Lowbyte	KEYCODE reference attachment
Byte [2]	Key Code Highbyte	KEYCODE reference attachment
Byte [3]	Keep	Keep

See Appendix 1 for KEY CODE on ConsumerKey page.

7

Page 9

1.3.5 SystemKey Page Definition

Byte Map	Descriptor	Value
Byte [0]	page	(0X04) System Key Page
Byte [1]	Key Code Low byte	KEYCODE reference attachment
Byte [2]	Key Code High byte	KEYCODE reference attachment
Byte [3]	Keep	Keep

The buttons on the SystemKey page are mainly power buttons, including Power, Sleep, and Wakeup. For details, see Appendix 1 KEY CODE.

1.3.6 Extra Function Page Definition

Byte Map	Descriptor	Value
Byte [0]	Page	(0x05) Extra Function Page
Byte [1]	Function item	0x01 -HW switch context profile 0x02-start OS program 0x03-disabled 0x04-Joystick mode 1 0x05-Joystick mode 2 0x06-Joystick mode 3 0x07-Joystick mode 4
Byte [2]	Functional data	1. Corresponding function item 0x01 (HW switching situation configuration File) of the context configuration file Value definition: Ex: 0x01-0x0A-Profile 1-10 2. Corresponding function item 0x02 (start OS program) Definition of Value: Start program number: (0x00~0x8F)
Byte [3]	Keep	Keep

The AP receives the data 0x05 0x00 0x0M 0x00 sent by the device Endpoint2, where the first byte is the report ID (0x05), and M is the startup program sequence number.

When AP receives the data sent by the device, it starts the application with the corresponding serial number. If the corresponding application does not exist, the system does not respond.

8

Page 10

1.3.7 Macro Key Page Definition

Byte Map	Descriptor	Value

Byte [0]	Page	(0x06) Macro Page
Byte [1]	Macro number	0x00~ 0x64 (supports 100 sets of Macro Key)
Byte [2]	Macro retransmission mode	0x00: play once
		0x01: fixed number of times
		0x02: Press again to end
Byte [3]	Macro retransmission times	0x00 ~ 0xFF

1.3.8 Multi-Key Tap Page Definition

I. KB supports Standard Key + Modifier Key (For Key Code, please refer to Appendix 1 "Key Combo and Macro Key Data")

II. KB supports up to 3 consecutive keys (different keys are available), and the AP places the keys in the order set by the user.

III. MS supports MS Key N-burst.

KB:

Byte Map	Descriptor	Value
Byte [0]	Page	(0x07) Multi-Key Tap Page
Byte [1]	Key Code 1	0x00~ 0xFF (Key Code)
Byte [2]	Key Code 2	0x00~ 0xFF (Key Code)
Byte [3]	Key Code 3	0x00~ 0xFF (Key Code)

MS:

Byte Map	Descriptor	Value
Byte [0]	Page	(0x07) Multi-Key Tap Page
Byte [1]	Number of combos	0x02~ 0xFF
Byte [2]	MS Key Code	0x01/0x02/0x04/0x08/0x10
		(L/R/M/4/5 Key code)
Byte [3]	Reserve	0x00

1.4 Area **D- LED** special effect definition area

1. The LED special effect definition area is divided into different storage contents of KB/MS/MS PAD:

KB: 16 Bytes *20 Non-customized special effect Page+ define one RGB LED status every 4 Bytes* N LEDs+64 Bytes LED Data Info. Page.

LED special effect unit format:

【LED Effect Page of KB (Note 1) + LED custom state + LED Data Information Page】 ------LED special effect unit format

Note 1: LED Effect Page of KB is the keyboard only data storage area, this data area does not exist in other devices. .

keyboard	Each Game Profile LED Effect area is 16 Bytes *20 Non-customized special effects Page+ Define an RGB LED state every 4 Bytes* 144 LEDs+ 64 Bytes LED Data Info. Page, LED addresses are based on RC Table 0.0, 0.1...7.17, a total of 144 (RC Table: Row.Column) are arranged in order. Ex: Address 0x280~0x283 is stored in the RC Table of Game Profile, the 0th LED is customized.

1.4.1 Format description of **LED** special effect unit:

The format of KB/MS/MS PADLED special effect unit is as follows:

【LEDEffect Page of KB (16 Bytes * 20 Page)+ LED custom status (4Bytes * N keys)+ Bytes LED Data Info. Page】

Note: LED Effect Page of KB is Keyboard only, this data area does not exist in other devices.

Each special effect 16 Bytes, each Flash Page describes 4 special effect data, the specific content is as follows:

Byte Map	content	Value
Byte[0]	Special effects mode	1-32
Byte[1]	colour	Full color: 0x00 Monochrome: 0x01
Byte[2]	R color ratio	0x00-0xFF (full color is 0, invalid)
Byte[3]	G color ratio	0x00-0xFF (full color is 0, invalid)
Byte[4]	B color ratio	0x00-0xFF (full color is 0, invalid)
Byte[5]	Dynamic direction	Left to right: 0x00, right to left: 0x01 Bottom to top: 0x02, top to bottom: 0x03
Byte[6]	Brightness control	0x00-0x0F (0x0F is the brightest)
Byte[7]	Cycle control	0x00-0x0F (0x0F has the longest period)
Byte[8:13]	Reserve	
Byte[14]	Check code_L	0xAA

Byte[15]	Check code_H	0x55

1.4.2 LED Effect Pages of KB

This storage area is only available in KB, and the purpose is to sort and store the special effects from AP. When AP is offline, KB can perform AP offline setting The corresponding special effects. In the same way, the HW operation content when offline can also be stored.

Special effects mode	colour	R, G, B color ratio	Dynamic direction brightness, period control	
Static constant bright	Monochrome/full color		no	UI maintains 0-15 levels
Single light up	Monochrome/full color		no	Same as above
Single off	Monochrome/full color		no	Same as above
Starry	Monochrome/full color		no	Same as above
All over the sky	Monochrome/full color		no	Same as above
A hundred flowers contend for beauty	Full color only	When it is monochrome, the 3 Byte	no	Same as above
Dynamic breathing	Monochrome/full color	Respectively represent the Value of RGB,	no	Same as above
Spectral cycle	Full color only	Range 0~0xFF, when it is full color	no	Same as above
Colorful spring surging	Monochrome/full color	The 3 Byte data is invalid,	no	Same as above
Colorful aspect	Monochrome/full color	Is 0x00.	up and down	Same as above
Go with the flow	Monochrome/full color		about	Same as above
Turn around	Monochrome/full color		about	Same as above
Immediately	Monochrome/full color		no	Same as above
one stone two bird	Monochrome/full color		no	Same as above
Ripple spread	Monochrome/full color		no	Same as above

Constant stream	Monochrome/full color	about	Same as above
Heavy mountain range	Monochrome/full color	no	Same as above
Stormy	Monochrome/full color	no	Same as above
Shuttle back and forth	Monochrome/full color	about	Same as above
Lights off	invalid	no	Same as above

1.4.3 LED custom mode

Send 576(KB) Bytes data(4Bytes * N keys), the format is as follows.

Byte Map	content	Value	Description
Byte [0]	LED custom mode	0x80 (custom)	
Byte [1]	R color ratio	0x00- 0xFF	
Byte [2]	G color ratio	0x00- 0xFF	
Byte [3]	B color ratio	0x00- 0xFF	

1.5 E area - macro data area

This area is completely controlled by the AP, and the KB is not modified. After the AP arranges the data, it is sent to the Device through the write macro area command. Macro space Small is dynamic. KB supports up to 100 sets of Macro Keys.

The macro needs to have Over size protection. When connecting, the AP first reads the basic message area to confirm the supported MACRO capacity. When the user recorded the total MACRO If the storage capacity of the device is exceeded, when you press OK, an error message will be displayed and the number of actions set by Macro will be displayed.

1.5.1 Macro recording method

Support MS/KB: Execute the macro recording function on the AP interface, the recording result is temporarily stored on the AP, and then downloaded to the MCU for storage.

Before recording the Macro Key, the AP needs to download the "Turn off the keyboard customization function" CMD, and when the recording is over, it needs to click the "Start the keyboard customiz

1.5.2 Macro format

[Macro number address table + macro data] ---------- Macro cell format

I. Macro number address table format:

The address table has a total of 400 Bytes (support 4Bytes*100 groups of macros) 0x00000000 ~ 0x0000018F, each macro number address is represented by 4 Bytes Pointer information, the purpose is to point to the corresponding macro row address in the macro data area.

The schematic diagram of the architecture is as follows (for detailed control instructions, please refer to the Macro example):

	ADDRESS OFFSET	BYTE MAP	Description	Value
Macro format				
	0x00-0x03	Byte [0:3]	Store the macro data address of the first key	0x00000190
	0x04-0x07	Byte [4:7]	Store the macro data address of the second key	0x000001C4
Macro number				
Address table	0x08-0x0B	Byte [8:11]	Store the macro data address of the 3rd key	0x000001F4
	…….	.....	.....	
	0x18C- 0x18F	Byte [396:399]	Store the macro data address of the 100th key	0x00000000

Page 13

			(Unused)
Macro data	0x190-0x1C3	Byte[400:451]	The macro data of the first key, Contains macro header (8 Bytes) and 11 macro actions Make (4 Byte * 11) = 52 Bytes
	0x1C4-0x1F3	Byte[452:499]	The macro data of the second key, Contains macro header (8 Bytes) and 10 macro actions Make (4 Byte * 10) = 48 Bytes
	0x1F4-0x223	Byte[500:547]	The macro data of the 3rd key, Contains macro header (8 Bytes) and 10 macro actions Make (4 Byte * 10) = 48 Bytes

Page 14

II. Macro data:

The macro header plus the macro actions it carries form a complete macro data:

Macro data = [Macro header (8 Bytes) + Macro action (4 Bytes * N macro actions)]-------- Macro data format

Macro header format:

The information in the macro header is the total number of macro actions it carries, that is, N macro actions and the value of N are stored in the macro header.

Byte Map	content
Byte [0]	The total number of macro keystrokes Low Byte
Byte [1]	Total number of macro keystrokes High Byte
Byte [2:7]	Keep

Macro action content:

Divided into two data formats, button and delay time.

Key format

Byte Map	Bit Map	content
Byte[3]	Bit[7]	Button state 1b: make (press) 0b: break (bounce)
	Bit[6:4]	Macro action classification 001b: Mouse button (L/M/R/4/5) 010b: Mouse wheel (roll up, scroll down) 011b: Keyboard (Modifier + Standard) 101b: Delay time
	Bit[3:0]	Keep
Byte[2]	Bit[7:0]	Key Code (a) 1. Mouse button (L/M/R/4/5) Mouse Key: Key L(0x01), Key M(0x04), Key R(0x02), Key 4(0x08), Key 5(0x10) 2. Mouse wheel Roll down: 0xFF, Roll up: 0x01 3. Keyboard class (Modifier: Ctrl, Alt, Shift...) Modifier Key Code 4. Keyboard (Standard) Standard Key Code
Byte[1]	Keep	Keep
Byte[0]	Keep	Keep

Delay time format

Byte Map	Bit Map	content
Byte[3]	Bit[7]	Keep
	Bit[6:4]	Macro action classification 101b: Delay time (b)
	Bit[3:0]	Bit of delay time[27:24]
Byte[2]		Delay time bit[23:16]
Byte[1]		Delay time bit[15:8]
Byte[0]		Delay time bit[7:0]

Note: (a) Refer to Appendix 1 for Key Code (keystroke and Macro Key data)

(b) Action delay, the minimum unit is 1ms, that is, 1 means 1ms.

The maximum unit only supports up to 99999999ms

1.5.3 Macro area check code

The length of the macro data varies according to the length of the recording, so the macro area check code is placed after the content of the macro data that is updated each time, on the next page (+64Bytes) check code Page. EX: Macro data is 452 Bytes, so the macro data actually occupies 452/64=7.... The remaining 4Bytes, so the actual number of macros According to the data, 8 Pages are occupied, and a check code is added after the 8th Page (+ 64 Bytes)

The macro data check code is as follows.

Byte Map	content	Value
Byte [0-61] Reserve		0x00
Byte [62]	Check code_L	0xAA
Byte [63]	Check code_H	0x55

1.5.4 **Macro** Examples

I. Added "MouseTest" macro with 11 mouse actions. The "MouseTest" macro is Macros are created according to the macro data format
The first macro data storage starts from 0x190

Macro number address table: 90 01 00 0000 00 00 00…(Total 400 Bytes)
Macro header: 0B 00 00 00 00 00 00 00 (length 8Bytes, 0x0B represents a total of 11 mouse actions)
Macro action content: 11 actions * 4 Bytes (length of each action) = 44 Bytes
A total of 400 + 8+44 = 452 Bytes.

II. Added "KeyTest" macro, which contains 10 keyboard actions
After storing the first macro (52 Bytes), the second macro address starts from 0x190 + 0x34 = 0x1C4

Macro number address table: 90 01 00 00C4 01 00 0000 00 00 00… (total 400 Bytes)
The first macro header + macro action content = 52 Bytes
Macro header: 0A 00 00 00 00 00 00 00 (length 8Bytes, 0x0A represents a total of 10 keyboard actions)
Macro action content: 10 actions * 4 Bytes (length of each action) = 40 Bytes
A total of 400 + 52 + 8 + 40 = 500 Bytes.

III. Added "KeyTest2" macro with 10 keyboard actions

After storing the first and second macros (100Bytes), the third macro address starts from 0x1C4+0x30 = 0x1F4

Macro number address table: 90 01 00 00C4 01 00 00F4 01 00 00 00 00 00 00 …(total 400 Bytes)
The first macro header + macro action content = 52 Bytes
The second macro header + macro action content = 48 Bytes
Macro header: 0A 00 00 00 00 00 00 00 (length 8Bytes, 0x0A represents a total of 10 keyboard actions)
Macro action content: 10 actions * 4 Bytes (length of each action) = 40 Bytes
A total of 400 + 52 +48 + 8 + 40 = 548 Bytes.

Notice: Please refer to 3.4 Macro Edit Interface for APP setting operation process.

IV. Macro output process:
1. Press the button to get the macro number address table of the corresponding button
2. Get the macro header through the macro number address table
3. Get a string of Macro actions through the macro header
4. Analyze the action description and output

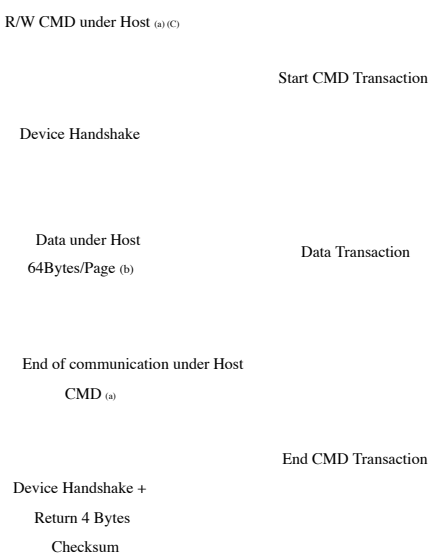
15

Page 17

Chapter 2 Communication Protocol Definition

2.1 Basic communication behavior

The input and output of a custom device is adopted, and it is realized by Set/Get Report Feature (EP0). The length of all input and output data is 64 Bytes.
The communication command establishment process is:



Note:
(a) The interval of command transmission is defined as 10ms
(b) The interval of data transmission is defined as 10ms

16

Page 18

2.2 Communication command format

All communication lengths are transmitted in 64 Bytes:

Byte Map	Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]
Data definition	Header	CMD	Keep	Device Status	CheckSum			
Data content	0x04	Communication command		Equipment back Should state	CheckSum [0]	CheckSum [1]	CheckSum [2]	CheckSum [3]

Byte Map	Byte[8]	Byte[9]	Byte[10]	Byte[11]	Byte[12]	Byte[13]	Byte[14]	Byte[15]
Data definition	DataLength Low Byte (Page)	Data Length High Byte (Page)	Keep	Keep	Keep	Keep	Keep	Keep
Data content	Send/receive Data length [0]	Send/receive Data length [1]	-	-	-	-	-	-

Byte Map	Byte[16-63]
Data definition	Zero (reserved)

I. Byte[3]: Device response status

Device Status	Value
ACK	0x01
NACK	0XFF

II. Byte[1]: Communication command definition:

CMD	Value	support	Description
End of communication	0X02	After KB/MS/MSPD read/write is completed, send the end command and the keyboard after the MCU reply command Resume normal work	
Read basic device information	0x05	KB/MS/MSPD read the basic information of the device 128 Bytes	
Read key definition area	0X10	KB	Read matrix situation
Write key definition area	0X11	KB	Change the matrix and redefine the button function
Read LED special effects area	0X12	KB	Read LED special effect area information
Write LED special effect area	0X13	KB	Write LED special effect area information and change LED special effect.
Read macro definition area	0x14	KB	Read the macro definition
Write macro definition area	0x15	KB	Change the macro data, complete the definition of the macro button
Read game mode area	0X16	KB	Read game mode area information
Write game mode area	0X17	KB	Write game mode area information
Start the keyboard custom function	0X18	KB	Start keyboard customization
Turn off the keyboard customization function	0X19	KB	Turn off keyboard customization
LED Effect Start	0xF0	KB/MS/MSPD LED special effect start command	
LED Sync Initial	0xF1	Before KB/MS/MSPD LED SYNC, let all devices do LED SYNC Initial	
LED Sync	0xF2	KB/MS/MSPD LED SYNC CMD, updated every time when the system runs the SW synchronization method After the data needs to bring this CMD, then download Data	
LED Sync End	0xF3	KB/MS/MSPD turns off Sync CMD in AP connection status	

Page 19

III. EP2 response command definition:

CMD	Value	support	Description
Sync Initial Ready CMD	0x05, 0xAA, 0xEE, 0x00	KB/MS/MSPD	The device responds to whether the Host currently completes the Effect Initial.
Start the program CMD	0x05, 0x00, 0x0M, 0x00	KB/MS/MSPD	The first byte is report ID (0x05), M is the sequence number of the startup program, the AP receives the device The data sent is to start the application corresponding to the serial number Sequence, if the corresponding application does not exist then The system is unresponsive.

Page 20

2.4 Different **ROM TYPE** of **APP** corresponding to set

Add MemoryType setting in .Ini file in APP file:

ROM TYPE	MemoryType Value	Description
EEPROM	0	Data Transaction is written at 10ms intervals
FLASH	1	Because it needs to be erased, Data Transaction needs to wait for the first time Erase time of 30ms, and 5ms interval writing starts (1k takes 35ms)

Writing example:

A total of 1792 Bytes (28 Pages) when updating the LED (Zone C).

1. LED write command 04 0A 00 00 00 00 00 00 1C 00 00 00...
2. MCU response 04 0A 00 01 00 00 00 00 1C 00 00 00...
3. Download data each time 64 Bytes C zone length is 1792 Bytes total 28 times, data time interval is 10 ms.
4. End command 04 02 00 00 00 00 00 00...
5. MCU response 04 02 00 01 xx xx xx xx (4 Bytes CheckSum) 00 00 00 00...
6. The AP confirms that CheckSum successfully ended the update.

Data transaction Recovery mechanism (Checksum error)

7. If Checksum is wrong, start the Recovery mechanism from 1 again.
8. After three attempts, the Checksum is still wrong, the AP changes and returns to the Default state CMD.
9. The error window is displayed.

When updating the Macro (zone E). The data length increases or decreases according to the downloaded data.

1. LED write command 04 0C 00 00 00 00 00 00 LL HH 00 00...
2. MCU response 04 0C 00 01 00 00 00 00 LL HH 00 00...
3. Download data every 64 Bytes, and the data interval is 10 ms.
4. End command 04 02 00 00 00 00 00 00...
5. MCU response 04 02 00 01 xx xx xx xx (4 Bytes CheckSum) 00 00 00 00...
6. The AP confirms that CheckSum successfully ended the update.

Read example:

The basic information of the device is 128 Bytes. (2 Pages)

1. Device (read) command 04 05 00 00 00 00 00 00 02 00 00 00...

2. MCU response 04 05 00 01 00 00 00 00 02 00 00 00...
3. The data is uploaded twice each time with 64 Bytes, and the data interval is 10 ms. Get Report Feature command under AP.
4. End command 04 02 00 00 00 00 00 00...
5. MCU response 04 02 00 01 xx xx xx xx (4 Bytes CheckSum) 00 00 00 00...
6. The AP confirms that CheckSum successfully ended the update.

Data transaction Recovery mechanism (Checksum error)

7. If Checksum is wrong, start the Recovery mechanism from 1 again.
8. Checksum is still wrong after three attempts.
9. The error window is displayed.

Page 21

4.2 Examples

The AP uses the 0xAB command to randomly download 64 bytes of random data information

0~15bytes 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xAD, 0xAE,0xAF

16~31bytes 0xB0, 0xB1, 0xB2, 0xB3, 0xB4, 0xB5, 0xB6, 0xB7, 0xB8, 0xB9, 0xBA, 0xBB, 0xBC, 0xBD, 0xBE,0xBF

32~47bytes 0xC0, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7, 0xC8, 0xC9, 0xCA, 0xCB, 0xCC, 0xCD, 0xCE,0xCF

48~63bytes 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xDE,0xDF

Take the data 3 th is 0xA3, 6 th is 0xA6, 36 th is 0xC4, 63 th is 0xDF, and the calculation is as follows

```
bData_Temp1 = 0xA3;                          //*Byte[3]
bData_Temp1 += 0xAC;                         //*Byte[12]
bData_Temp2 = 0xA6;                          //*Byte[6]
bData_Temp2 -= 0xAD;                         //*Byte[13]
bData_Temp3 = 0xC4;                          //*Byte[36]
bData_Temp3 &= 0xA5;                         //*Fixed value
bData_Temp4 = 0xDF;                          //*Byte[63]
bData_Temp4 <<= 3;
wData_Temp0 = *((uint32_t *)0x1FFF01FC);     //*0x54711688 This is the fixed data inside the MCU, and the AP part is filled with fixed data
                                             0x54711688

wData_Temp1 = (bData_Temp1 << 24) |          //*(0x4FF8F984) Combine the four Bytes after the operation
(bData_Temp4 << 16) | (bData_Temp2 << 8) |
(bData_Temp3);

wData_Out = (wData_Temp0)^( wData_Temp1);    //*(0x4FF8F984)XOR(0x54711688) =0x1B89EF0C
```

//*Finally, if wData_Temp0 and wData_Temp1 are mutually exclusive OR, the MCU will return wData_Out to the AP through the checksum in the End command

Page 22

Appendix 1: KEY CODE

```
//=========================================== ====================
//Mouse KEY_CODE
Left button  mouse_key_left          0x01
Right click  mouse_key_right         0x02
Middle button mouse_key_middle       0x04
Back key mouse_key_4                 0x08
Forward key mouse_key_5              0x10
Swing left   mouse_rock_left         0xff
Right swing mouse_rock_right         0x01

//=========================================== ====================
//Keyboard hotkey (each key function occupies 1bit)
#define key_ctrl_l                   0x01
#define key_shift_l                  0x02
#define key_alt_l                    0x04
#define key_win_l                    0x08
#define key_ctrl_r                   0x10
#define key_shift_r                  0x20
#define key_alt_r                    0x40
#define key_win_r                    0x80

//The AP will give Device data to the device when the key is double-clicked and the Macro Key
#define key_ctrl_l                   0xe0
#define key_shift_l                  0xe1
#define key_alt_l                    0xe2
#define key_win_l                    0xe3
#define key_ctrl_r                   0xe4
#define key_shift_r                  0xe5
#define key_alt_r                    0xe6
#define key_win_r                    0xe7

//Ordinary keyboard keys
#define key_a                        0x04
#define key_b                        0x05
#define key_c                        0x06
#define key_d                        0x07
#define key_e                        0x08
#define key_f                        0x09
#define key_g                        0x0a
#define key_h                        0x0b
#define key_i                        0x0c
#define key_j                        0x0d
#define key_k                        0x0e
#define key_l                        0x0f
```


Page 23

```
#define key_m                        0x10
#define key_n                        0x11
#define key_o                        0x12
#define key_p                        0x13
#define key_q                        0x14
#define key_r                        0x15
#define key_s                        0x16
#define key_t                        0x17
#define key_u                        0x18
#define key_v                        0x19
#define key_w                        0x1a
#define key_x                        0x1b
#define key_y                        0x1c
#define key_z                        0x1d

#define key_1                        0x1e
#define key_2                        0x1f
```

```
#define key_3                   0x20
#define key_4                   0x21
#define key_5                   0x22
#define key_6                   0x23
#define key_7                   0x24
#define key_8                   0x25
#define key_9                   0x26
#define key_0                   0x27
#define key_enter               0x28
#define key_esc                 0x29
#define key_backspace           0x2a
#define key_tab                 0x2b
#define key_space               0x2c
#define key_mis                 0x2d    //-and _
#define key_equ                 0x2e    // = and +
#define key_oqo                 0x2f    // [and {
#define key_eqo                 0x30    //] and}
#define key_bsl                 0x31    // \ and |
#define key_COL                 0x33    //; and:
#define key_CC                  0x34    // 'and "
#define key_GAT                 0x35    // `and ~
#define key_CMA                 0x36    //, and <
#define key_DOT                 0x37    //. and>
#define key_SL                  0x38    // / and?
#define key_cap                 0x39
#define key_f1                  0x3a
#define key_f2                  0x3b
#define key_f3                  0x3c
#define key_f4                  0x3d
#define key_f5                  0x3e
```


Page 24

```
#define key_f6                  0x3f
#define key_f7                  0x40
#define key_f8                  0x41
#define key_f9                  0x42
#define key_f10                 0x43
#define key_f11                 0x44
#define key_f12                 0x45
#define key_print               0x46
#define key_scroll              0x47
#define key_pause               0x48
#define key_insert              0x49
#define key_home                0x4a
#define key_pgup            0x4b
#define key_delete              0x4c
#define key_end                 0x4d
#define key_pgdn                0x4e
#define key_right               0x4f
#define key_left                0x50
#define key_down                0x51
#define key_up                  0x52
#define key_numlock             0x53
#define key_num_PADNSL 0x54             // keypad /
#define key_num_PADNMU 0x55             // keypad *
#define key_num_PADNMI 0x56             // keypad-
#define key_num_PADNPL 0x57             // keypad +
#define key_num_enter           0x58
#define key_num_1               0x59
#define key_num_2               0x5a
#define key_num_3               0x5b
#define key_num_4               0x5c
#define key_num_5               0x5d
#define key_num_6               0x5e
#define key_num_7               0x5f
#define key_num_8               0x60
```

```
#define key_num_9                   0x61
#define key_num_0                   0x62
#define key_num_del                 0x63
#define key_app                     0x65
;============================================= ============================
; Standard USB usage codes for different languages
;============================================= ============================
#define key_k29                     0x31
#define key_k42                     0x32
#define key_k45                     0x64
#define key_k107                    0x85
#define key_k56                     0x87
#define key_k14                     0x89
```


Page 25

```
#define key_k132                    0x8a
#define key_k131                    0x8b
#define key_k151                    0x90
#define key_k150                    0x91
//============================
//ConsumerKey button
#define key_calculator_l                0x92
#define key_calculator_h                0x01
/Calculator

#define key_web_l                   0x23
#define key_web_h                   0x02
/Web page

#define key_play_l                  0xCD
#define key_play_h                  0x00
/play / Pause

#define key_search_l                0x21
#define key_search_h                0x02
/search for

#define key_mail_l                  0x8a
#define key_mail_h                  0x01
/mailbox

#define key_vol_add_l               0xe9
#define key_vol_add_h               0x00
/Volume+

#define key_mute_l                  0xe2
#define key_mute_h                  0x00
/Mute

#define key_stop_l                  0xb7
#define key_stop_h                  0x00
/stop

#define key_vol_dec_l               0xea
#define key_vol_dec_h               0x00
/volume-

#define key_favorite_l              0x2a
#define key_favorite_h              0x02
/Collection

#define key_forward_l               0x25
```

Page 26

```
#define key_forward_h 0x02
/Page forward

#define key_back_l 0x24
#define key_back_h 0x02
/Page back

#define key_prev_l 0xb6
#define key_prev_h 0x00
/Previous

#define key_next_l 0xb5
#define key_next_h 0x00
/next song

#define key_computer_l 0x94
#define key_computer_h 0x01
/My computer

#define key_refresh_l 0x27
#define key_refresh_h 0x02
/Refresh

#define key_media_l 0x83
#define key_media_h 0x01
/music player

//SystemKey button
#define key_power_l 0x01
#define key_power_h 0x00

#define key_sleep_l 0x02
#define key_sleep_h 0x00

#define key_wakeup_l 0x04
#define key_wakeup_h 0x00
```


Page 27

Appendix 2: Chinese and English comparison table of UI interface

Chinese	English
Scene mode	Profile
Scene mode editing	Profile Edit
Left button	Left Click
Right click	Right Click
Middle button	Center Click
go ahead	Forward
Back	Backward
Scroll up	Scroll Up
Scroll down	Scroll Down
Macro editing	Macro Edit

Restore default settings	Default settings
brightness	Brightness
speed	Speed
USB report rate	USB Report Rate
determine	Confirm
cancel	Cancel
application	Apply
Macro/Macro	Macro
Move up to top	Move Top
Move up	Move Up
Move down	Move Down
Move down to the end	Move Bottom
Record	Record
Empty	Clear all items
insert	Insert
Insert a mouse event	Insert a mouse event
Fixed delay	Fixed Delay
Cycles	Loop count
Mouse pointer moving speed	Mouse move Speed
Activate mouse X/Y pointer sensitivity	
Mouse scroll speed	Mouse Scroll Speed
Scroll one screen at a time	Scoll one Screen
Mouse double click speed	Mouse Click Speed
Adjust the silent height of the mouse	
default	Default
Keyboard function	Keyboard Function
Mouse function	Mouse Function
Switch profile	Switch Profile
starting program	Run Program
Windows shortcut keys	Windows Hotkey
Combination keys	Combo Key

Page 28

Disable	Disable
next song	Next Track
Previous song	Previous Track
stop	Stop
play / Pause	Play/Pause
Mute	Mute
Volume+	Volume +
volume-	Volume-
Cut	Cut
copy	Copy
Paste/paste	Paste
delete	Delete
select all	Selete All
Find	Find
New	New
save	Save

27

Page 29

Appendix III. LED address and key correspondence number (transmission sequence)

#define	LED_ESCAPE	0	
#define	LED_F1	1	
#define	LED_F2	2	
#define	LED_F3	3	
#define	LED_F4	4	
#define	LED_F5	5	
#define	LED_F6	6	
#define	LED_F7	7	
#define	LED_F8	8	
#define	LED_F9	9	
#define	LED_F10	10	
#define	LED_F11	11	
#define	LED_F12	12	
#define	LED_PRINTSCREEN	13	
#define	LED_SCROLLLOCK	14	
#define	LED_PAUSE	15	
#define	LED_GAT	16	
#define	LED_1	17	
#define	LED_2	18	
#define	LED_3	19	
#define	LED_4	20	
#define	LED_5	twenty one	
#define	LED_6	twenty two	
#define	LED_7	twenty three	
#define	LED_8	twenty four	
#define	LED_9	25	
#define	LED_0	26	
#define	LED_MIS	27	//-_
#define	LED_EQU	28	//=+
#define	LED_BACKSPACE	29	
#define	LED_INSERT	30	
#define	LED_HOME	31	
#define	LED_PAGEUP	32	
#define	LED_PADNUMLOCK	33	
#define	LED_PADNSL	34	//Pad/
#define	LED_PADNMU	35	//Pad*
#define	LED_PADNMI	36	//Pad-
#define	LED_TAB	37	
#define	LED_Q	38	
#define	LED_W	39	
#define	LED_E	40	
#define	LED_R	41	

28

Page 30

```
#define LED_T 42
#define LED_Y 43
#define LED_U 44
#define LED_I 45
#define LED_O 46
#define LED_P 47
#define LED_OQO 48 //[{
#define LED_EQO 49 //]}
#define LED_BSL 50 /\|
#define LED_DELETE 51
#define LED_END 52
#define LED_PAGEDOWN 53
#define LED_PAD7 54
#define LED_PAD8 55
#define LED_PAD9 56
#define LED_PADNPL 57 //Pad+

#define LED_CAPSLOCK 58
#define LED_A 59
#define LED_S 60
#define LED_D 61
#define LED_F 62
#define LED_G 63
#define LED_H 64
#define LED_J 65
#define LED_K 66
#define LED_L 67
#define LED_COL 68 //:;
#define LED_CC 69 //'"
#define LED_ENTER 70
#define LED_PAD4 71
#define LED_PAD5 72
#define LED_PAD6 73

#define LED_LEFTSHIFT 74
#define LED_Z 75
#define LED_X 76
#define LED_C 77
#define LED_V 78
#define LED_B 79
#define LED_N 80
#define LED_M 81
#define LED_CMA 82 //,<
#define LED_DOT 83 //.>
#define LED_SL 84 ///?
#define LED_RIGHTSHIFT 85
#define LED_UP 86
```


Page 31

```
#define LED_PAD1 87
#define LED_PAD2 88
#define LED_PAD3 89

#define LED_LEFTCTRL 90
#define LED_LEFTGUI 91
#define LED_LEFTALT 92
#define LED_SPACEBAR 93
#define LED_PADENTER 94
#define LED_RIGHTALT 95
#define LED_FN 96
#define LED_APPLICATION 97
#define LED_RIGHTCTRL 98
```

```
#define LED_LEFT 99
#define LED_DOWN 100
#define LED_RIGHT 101
#define LED_PAD0 102
#define LED_PADDEL 103
```