C-Power5200 SDK API Manual

Version: V1.6

2015.05.28

Recension Log:

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Changes | Executor |
| 2009-8-11 | V1.0 | The first version |  |
| 2010-1-28 | V1.1 | 1. Add multi-window protocol data packing API 2. Add multi-window protocol serial and network simple use API |  |
| 2010-5-22 | V1.2 | Increase the following functions:：   1. CP5200\_Program\_AddLafPict 2. CP5200\_Program\_AddLafVideo 3. CP5200\_Program\_AddVariable 4. CP5200\_MakeGetTypeInfoData 5. CP5200\_ParseGetTypeInfoRet 6. CP5200\_MakeGetTempHumiData 7. CP5200\_ParseGetTempHumiRet 8. CP5200\_MakeReadConfigData 9. CP5200\_ParseReadConfigRet 10. CP5200\_MakeWriteConfigData 11. CP5200\_ParseWriteConfigRet 12. CP5200\_RS232\_GetTemperature 13. CP5200\_RS232\_GetTypeInfo 14. CP5200\_Net\_GetTemperature 15. CP5200\_Net\_GetTypeInfo |  |
| 2011-02-24 | V1.3 | Increase the following functions:   1. CP5200\_MakeReadHWSettingData 2. CP5200\_ParseReadHWSettingRet 3. CP5200\_MakeWriteHWSettingData 4. CP5200\_ParseWriteHWSettingRet 5. CP5200\_RS232\_ReadHWSetting 6. CP5200\_RS232\_WriteHWSetting 7. CP5200\_Net\_ReadHWSetting 8. CP5200\_Net\_WriteHWSetting |  |
| 2012-03-16 |  | Increase the following functions:   1. CP5200\_RS232\_RemoveFile 2. CP5200\_Net\_RemoveFile |  |
| 2012.08.04 | V1.4 | 1. Increase the following functions:  CP5200\_CommData\_SetParam |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | CP5200\_MakeReadRunningInfoData  CP5200\_ParseReadRunningInfoRet  CP5200\_MakeScreenTestData  CP5200\_ParseScreenTestRet  CP5200\_MakeInstantMessageData1  CP5200\_MakeOpenFileData  CP5200\_ParseOpenFileRet  CP5200\_MakeGetDirentryData CP5200\_ParseGetDirentryRet  CP5200\_MakeReadFileNoData  CP5200\_ParseReadFileNoRet  CP5200\_MakeCloseFileNoData CP5200\_ParseCloseFileNoRet  CP5200\_CmmPacket\_SetParam  CP5200\_MakePlaySelectedPrgData1  CP5200\_MakeSetZoneAndVariableData  CP5200\_ParseSetZoneAndVariableRet  CP5200\_MakeSendPureTextData  CP5200\_ParseSendPureTextRet  CP5200\_RS232\_SetZoneAndVariable  CP5200\_RS232\_SendPureText  CP5200\_Net\_SetZoneAndVariable  CP5200\_Net\_SendPureText  CPowerBox\_MakeSendClockOrTemperatureData  CPowerBox\_ParseSendClockOrTemperatureRet  CPowerBox\_MakeSetAloneProgramData  CPowerBox\_ParseSetAloneProgramRet  CPowerBox\_MakeQueryProgramData CPowerBox\_ParseQueryProgramRet CPowerBox\_MakeSetScheduleData  CPowerBox\_ParseSetScheduleRet  CPowerBox\_MakeDeleteScheduleData  CPowerBox\_ParseDeleteScheduleRet  CPowerBox\_MakeGetScheduleData  CPowerBox\_ParseGetScheduleRet  CPowerBox\_RS232\_SendClockOrTemperature  CPowerBox\_RS232\_SetAloneProgram  CPowerBox\_RS232\_QueryProgram  CPowerBox\_RS232\_SetSchedule  CPowerBox\_RS232\_DeleteSchedule  CPowerBox\_RS232\_GetSchedule  CPowerBox\_Net\_SendClockOrTemperature  CPowerBox\_Net\_SetAloneProgram CPowerBox\_Net\_QueryProgram |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | CPowerBox\_Net\_SetSchedule  CPowerBox\_Net\_DeleteSchedule  CPowerBox\_Net\_GetSchedule  CP5200\_Net\_SetBindParam  2. Perfect interface parameters |  |
| 2012-05-08 |  | Increase the following functions:  CP5200\_Program\_AddFormattedText  CP5200\_Program\_AddFormattedTextW  CP5200\_TextToImage  CP5200\_TextToImageW |  |
| 2013/8/5 |  | Increase the following functions:  CP5200\_MakeReadSoftwareSwitchInfoData  CP5200\_ParseReadSoftwareSwitchInfoRet  CP5200\_MakeWriteSoftwareSwitchInfoData  CP5200\_ParseWriteSoftwareSwitchInfoRet  CP5200\_RS232\_ReadSoftwareSwitchInfo  CP5200\_RS232\_WriteSoftwareSwitchInfo  CP5200\_Net\_ReadSoftwareSwitchInfo  CP5200\_Net\_WriteSoftwareSwitchInfo |  |
| 2014/1/23 |  | Increase the following functions:  CP5200\_Program\_AddFormattedTextEx  CP5200\_TextToImageEx |  |
| 2015/3/17 |  | Increase the following functions:  CP5200\_MakeQueryControllerInfo  CP5200\_ParseQueryControllerInfoRet  CP5200\_Net\_QueryControllerInfo |  |
| 2015/4/1 |  | Increase the following functions:  CP5200\_RS232\_ReadNetworkParam  CP5200\_RS232\_WriteNetworkParam CP5200\_RS232\_Upgrade  CP5200\_Net\_ReadNetworkParam  CP5200\_Net\_WriteNetworkParam  CP5200\_Net\_Upgrade |  |
| 2015/5/18 |  | Increase the following functions:  CP5200\_MakeSendMultiProtocol  CP5200\_ParseSendMultiProtocoltRet  CP5200\_Net\_SendMultiProtocol  CP5200\_RS232\_SendMultiProtocol |  |
| 2015/5/27 |  | Modify the following functions:  CP5200\_MakeSendPictureData  CP5200\_RS232\_SendPicture  CP5200\_Net\_SendPicture  Increase the following functions: CP5200\_MakeSendSimpleImageData  CP5200\_ParseSendSimpleImageRet  CP5200\_RS232\_SendSimpleImageData |  |
|  |  | CP5200\_Net\_SendSimpleImageData |  |

# Basic definition

1.1. Data type

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Definition |
| Object Handle | HOBJECT | void\* |

1.2. Classification of API function

 Creating program file API function

 Create playbill file API function

 Communication data API function

1.3. Common operating steps

1. Create program file
2. Create playbill file
3. Use communication data API to generate command data, then send the data to controller and receive return data, also use communication data API to parse the return data and get the result。

Note:control card only search program files “playbill.lpp” when it starts, if the generated is saved as other file name, when the program single-file(“.lpp”) sent to the card ,you need to change

the file name “playbill.lpp”.

1.4. Communication protocol

C-Power5200 controller support RS232/485 and network communication

mode.

1.4.1. RS232/485

Communication start code is 0xA5, end code is 0xAE。Between start code and end code, if there is 0xA5, 0xAA or 0xAE, it should be converted to two code。

When PC send data to controller, convert one code to two code:

0xA5  0xAA 0x05

0xAA  0xAA 0x0A

0xAE  0xAA 0x0E

When PC receive data from controller, convert two code to one code:

0xAA 0x05  0xA5

0xAA 0x0A  0xAA

0xAA 0xAE  0xAE

1.4.2. Network

Need 4 bytes controller network ID code at the beginning of data

to be sent to controller.

1.5. Special effect for text and picture

|  |  |
| --- | --- |
| Code | Effect |
| 0 | Draw |
| 1 | Open from left |
| 2 | Open from right |
| 3 | Open from center(Horizontal) |
| 4 | Open from center(Vertical) |
| 5 | Shutter(vertical) |
| 6 | Move to left |

|  |  |
| --- | --- |
| 7 | Move to right |
| 8 | Move up |
| 9 | Move down |
| 10 | Scroll up |
| 11 | Scroll to left |
| 12 | Scroll to right |
| 13 | Flicker |
| 14 | Continuous scroll to left |
| 15 | Continuous scroll to right |
| 16 | Shutter(horizontal) |
| 17 | Clockwise open out |
| 18 | Anticlockwise open out |
| 9 | Windmill |
| 20 | Windmill（anticlockwise） |
| 21 | Rectangle forth |
| 22 | Rectangle entad |
| 23 | Quadrangle forth |
| 24 | Quadrangle endtad |
| 25 | Circle forth |
| 26 | Circle endtad |
| 27 | Open out from left up corner |
| 28 | Open out from right up corner |
| 29 | Open out from left bottom corner |
| 30 | Open out from right bottom corner |
| 31 | Bevel open out |
| 32 | AntiBevel open out |
| 33 | Enter into from left up corner |
| 34 | Enter into from right up corner |
| 35 | Enter into from left bottom corner |

|  |  |
| --- | --- |
| 36 | Enter into from lower right corner |
| 37 | Bevel enter into |
| 38 | AntiBevel enter into |
| 39 | Horizontal zebra crossing |
| 40 | Vertical zebra crossing |
| 41 | Mosaic(big) |
| 42 | Mosaic(small) |
| 43 | Radiation(up) |
| 44 | Radiation(downwards) |
| 45 | Amass |
| 46 | Drop |
| 47 | Combination(Horizontal) |
| 48 | Combination(Vertical) |
| 49 | Backout |
| 50 | Screwing in |
| 51 | Chessboard(horizontal) |
| 52 | Chessboard(vertical) |
| 53 | Continuous scroll up |
| 54 | Continuous scroll down |
| 55 | Reserved |
| 56 | Reserved |
| 57 | Gradual bigger(up) |
| 58 | Gradual smaller(down) |
| 59 | Reserved |
| 60 | Gradual bigger(vertical) |
| 61 | Flicker(horizontal) |
| 62 | Flicker(vertical) |
| 63 | Snow |
| 64 | Scroll down |
| 65 | Scroll from left to right |
| 66 | Open out from top to bottom |
| 67 | Sector expand |
| 68 | Reserved |
| 69 | Zebra crossing (horizontal) |
| 70 | Zebra crossing (Vertical) |
| 32768 | Random effect |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. 6.Text extend tag

The Text of contain extend tags may contain extend tags as below and all extend tag must be write in lowercase letters.

|  |  |
| --- | --- |
| Extend sign | Description |
| <size> | Designate the size of letters , must append attribute value,otherwise it will be ignore,if the attribute  value is inoperative,it will be ignore  also.Attribute value is the size of letter,virtual  value as below：  <size=8> : 8 lattice letter  <size=16> : 16 lattice letter  <size=24> : 24 lattice letter  <size=32> : 32 lattice letter |
| <color> | Designate the color of letters , must append  attribute value,otherwise it will be ignore,if the attribute value is inoperative,it will be ignore |
|  | also.Attribute value is the color of RGB hex  value ,for example：  <color=#ff0000> : Red  <color=#00ff00> : Green  <color=#0000ff> : Blue |
| <p> | Newline |
| <align> | The level of alignment , must append attribute value, otherwise it will be ignore,if the attribute value is inoperative,it will be ignore also.Virtual value  as below：  <align=left> : left Alignment  <align=center> ：centerAlignment  <align=right> ：right Alignment |
| <font> | Designate the font style of letters , must append attribute value,otherwise it will be ignore,if the attribute value is inoperative,it will be ignore also.Attribute value is the size of letter,virtual  value as below：  <font=0> : Default font <font=1> : Font 1  ……  <font=7> : Font 7 |

* 1. Font size code and font style
     1. Font size code

|  |  |
| --- | --- |
| Code | Font Size(lattice) |
| 0 | 8 |
| 1 | 12 |
| 2 | 16 |
| 3 | 24 |
| 4 | 32 |
| 5 | 40 |
| 6 | 48 |
| 7 | 56 |

* + 1. Font style code

|  |  |
| --- | --- |
| Code | Font style |
| 0 | Font 0 defaule font |
| 1 | Font 1 |
| 2 | Font 2 |
| 3 | Font 3 |
| 4 | Font 4 |
| 5 | Font 5 |
| 6 | Font 6 |
| 7 | Font 7 |

Note：If no special instructions, the parameter of the function in this document called

"nFontSize" was defined in the following format:

Byte 0~1：font size（lattice），如 8、12、24、32、40、48、56

Byte 2：Bit 0~2，font style code；

Bit 3，Whether the specified font to use for each character (0 default font, 1 specify the font with each character), and add tagtext program is set to 1, others it is set to 0;

Bit4~7，Resvered.

Byte 3：Resvered

* 1. Font color code

One-byte font color code:

It can express 8 kinds of color. Use each one bit to represent red、green、blue.

The lowest stands for red

The last but one stands for green The antepenultimate stands for blue 3 One-byte font color code:

It can express all kinds of color. Use each one byte to represent red、green、blue.

* 1. Picture effect code

|  |  |
| --- | --- |
| Code | Picture effect |
| 0 | Center |
| 1 | Zoom |
| 2 | Stretch |
| 3 | tile |

* 1. Clock format and display content

Clock format： Represent by one byte：

bit 0: Signal timing(0: 12signal timing；1: 24 signal timing) bit 1: Year by bit(0: 4 bit；1: 2 bit) bit 2: Line folding(0: single-row；1: multi-row) bit 3~5: Reserved(set to 0) bit 6: Show time scale ”Hour scale、Minute scale” bit 7: Reserved(set to 0)

Clock display content：

Represent by one byte： Ascertain the display content by bit:

bit 7: pointer bit 6: week bit 5: second bit 4: minute bit 3: hour bit 2: day bit 1: month bit 0: year

1.11. Simple picture data format

Data composition：

|  |  |  |  |
| --- | --- | --- | --- |
| Data head | Red data(optional) | Green data(optional) | Blue data(optional) |

Data head description：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0x00 | identify | | width | | height | | property | Reserved |

Description：

|  |  |  |
| --- | --- | --- |
| Data name | Data size(byte) | Description |
| Identify | 2 | Set to “I1”。 |
| Width | 2 | The width of the picture, low byte previous(little endian) |
| Height | 2 | The height of the picture,low byte previous(little endian) |
| Property | 1 | The gray-scale and color of the picture  Bit0: Whether exist red data, exist when 1.  Bit1: Whether exist green data, exist when 1.  Bit2: Whether exist blue data, exist when 1.  Bit3: Reserved, set to 0.  Bit4~7: Gray-scale, only support 0 and 7 now.  0: 2 levels gray, Each lattic data use 1 bit.  7: 256 levels gray, Each lattic data use 8 bit.  Each line of the picture data is aligned by byte. As for 2 levels gray picture, when the line data is not enough for 8 bit, add 0. |
| Reserved | 1 | Set 0 |

Data description：

The color of the data is order by red、green、blue. If the identify bit of the property is 0, the color data is not exist.

For one color data, order by ” left to right, top to bottom”. First put the data to the first line, then second line and so on.

1.12. Global zone message format

Each zone take 16 bytes, the format as below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0x00 | Type | Mode | x |  | y |  | cx |  | cy |  | ItemPropData | | | |  |  |

Description：

|  |  |  |
| --- | --- | --- |
| Data name | Data size(byte) | Description |
| Type | 1 | 1：Show text  2：Show specify picture file  3：Clock  4：Temperature  5：Humidity  6：Hint text(After the ’\n’)  7：Time |
| Mode | 1 | Have different meanings according to the window mode.  When the window mode is 1 or 6 means text align mode. |
|  |  | When window mode is 2 means the picture show effect:  0: Center 1:zoom 2:stretch 3:tile  Ignore the other window mode right now,set to 0 default. |
| X | 2 | Start point X, high byte previous(big endian) |
| y | 2 | Start point Y, high byte previous(big endian) |
| cx | 2 | Zone width, high byte previous(big endian) |
| cy | 2 | Zone height, high byte previous(big endian) |
| ItemPropData | 6 | The property value of the zone, it depends on the window mode. |

ItemPropData particular description

1.Show text

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F |
| 0x00 | start | end | Stay | | Font color | Reserved |

Description：

|  |  |  |
| --- | --- | --- |
| Data Item | Data  Size(BYTE) | Description |
| Start | 1 | Start number,valid value: 1～100 |
| End | 1 | End number,valid value: 1～100 |
| Stay | 2 | Stay time of each valid variable in second. High byte previous(big endian) |
| Font color | 1 | Bit0~3：Font size  0：8 lattic  1：12 lattic  2：16 lattic  3：24 lattic  4：32 lattic  5：40 lattic  6：48 lattic  7：56 lattic  8：64 lattic    Bit4~6：Color  0：Black  1：Red  2：Green  3：Yellow  4：Blue  5：Mauve |
|  |  | 6：Cyan  7：White    Bit7：Whether invert color，1 is yes |

2.Show specify picture file

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F |
| 0x00 | Start | End | Stay | | Reserved | Reserved |

Description：

|  |  |  |
| --- | --- | --- |
| Data Item | Data  Size(BYTE) | Description |
| Start | 1 | Start number,valid value: 1～100 |
| End | 1 | End number,valid value: 1～100 |
| Stay | 2 | Stay time of each valid variable in second. High byte previous(big endian) |

3.Clock

4.Temperature

5.Humidity

6.Hint text

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F |
| 0x00 | Hint window number | | Stay | | Font color | Reserved |

Description：

|  |  |  |
| --- | --- | --- |
| Data Item | Data  Size(BYTE) | Description |
| The window number  of hint | 2 | Mark which window need this hint by bit,1 is hint, 0 is not  Bit 0：window number 1  Bit 1：window number 2  ……  Bit 15: window number 16 |
| Stay | 2 | Stay time of each valid variable in second. High byte previous(big endian) |
| Font color | 1 | See the “Font color” in “1. Show text” |

Note：

1. It will ignore when the window number of hint is set to “6.hint text”.
2. When the variable data of the window number of hint doesn’t have’\n’ or have none data

after the ‘\n’, this variable will not take part in the hint.

3.When synchronization stay time equals the max time of switch window div the number of the hint window.

7. Time

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F |
| 0x00 | Time number | Format | Stay | | Font color | Reserved |

Description：

|  |  |  |
| --- | --- | --- |
| Data Item | Font size(BYTE) | Description |
| Time number | 1 | Time number |
| Format | 1 | 0: “mm:ss”  1: “mm:ss:nn” 2: “hh:mm:ss”  3: “hh:mm:ss:nn” |
| Stay | 2 | Stay time of each valid variable in second. High byte previous(big endian) |
| Font color | 1 | See the “Font color” in “1. Show text” |

1.13. Window position and property

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0x00 | x |  | y |  | cx |  | cy |  | Window property | | | |  |  |  |  |

Description：

|  |  |  |
| --- | --- | --- |
| Data Item | Data  Size(BYTE) | Description |
| x | 2 | Window start point x, high byte previous (big endian). |
| y | 2 | Window start point Y, high byte previous (big endian). |
| cx | 2 | Window width.High byte previous (big endian). |
| Cy | 2 | Window height.High byte previous (big endian). |
| Window property | 8 | Window default type and parameters. |

Window property is represent by 8 bytes：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0x00 | mode | Parameter | | |  |  |  |  |

The definition of the window mode：

|  |  |
| --- | --- |
| Mode value | Description |
| 0 | Blank(Show nothing) |
| 1 | Text |
| 2 | Clock、calendar |
| 3 | Temperature、Humidity |
| 4 | Picture、Reference of the picture |
| Other | Reserved |

The parameter has different values according to the the mode.There are all the modes’s parameters. All reserved position should be set 0x00.

Blank type

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0x00 | 0 | Reserved | | |  |  |  |  |

Text type

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0x00 | 1 | mode | Font size | Font color | Speed | Stay time | | Reserved |

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Length(BYTE) | Description |
| mode | 1 | 1 | See in ‘1.7’ |
| Font size |  | 1 | Bit0~2: Font size  0x00: 8 lattice（Only english）  0x01: 12 lattice（Only english）  0x02: 16 lattice  0x03: 24 lattice  0x04: 32 lattice |
| Font color |  | 1 | Bit0~2: Font color  0x01: Red  0x02: Green  0x03: Yellow  0x04: Blue  Red、green、blue can make up another color by bit. |

Clock/Calendar type

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0x00 | 2 | Font size | Font color | Stay time | | calendar | Format | Content |

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Length(BYTE) | Description |
| Font size |  | 1 | Bit0~2: Font size  0x00: 8 lattice（Only english）  0x01: 12 lattice（Only english）  0x02: 16 lattice  0x03: 24 lattice  0x04: 32 lattice |
| Font color |  | 1 | Bit0~2: Font color  0x01: Red  0x02: Green  0x03: Yellow  0x04: Blue  Red、green、blue can make up another color by bit. |
| Stay time | 0x0000~0xffff | 2 | High byte previous(big endian), in seconds. |
| Calendar |  | 1 | 0: The gregorian calendar |
| Format |  | 1 | bit 0: Signal timing(0: 12 signal timing；1: 24 signal timing) bit 1: Year by bit(0: 4 bit；1: 2 bit) bit 2: Line folding(0: single-row；1:  multi-row)  bit 3~5: Reserved(set to 0)  bit 6: Show time scale ”Hour scale、 Minute scale” bit 7: Reserved(set to 0) |
| Content |  | 1 | Ascertain the display content by bit:  bit 7: pointer bit 6: week bit 5: second bit 4: minute bit 3: hour bit 2: day bit 1: month bit 0: year |

Temperature and Humidity type

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0x00 | 3 | Font size | Font color | Stay time | | Format | Reserved | |

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Length(BYTE) | Description |
| Font size |  | 1 | Bit0~2: Font size  0x00: 8 lattice（Only english）  0x01: 12 lattice（Only english）  0x02: 16 lattice  0x03: 24 lattice  0x04: 32 lattice |
| Font color |  |  | Bit0~2: Font color  0x01: Red  0x02: Green  0x03: Yellow  0x04: Blue  Red、green、blue can make up another color by bit. |
| Stay time | 0x0000~0xffff | 2 | High byte previous (big endian), in seconds. |
| Format |  | 1 | 0: Celsius  1: Fahrenheit  2: Humidity |

Picture and reference to the picture

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0x00 | 4 | Mode | Speed | Stay time | | Reserved | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Length(BYTE) | Description |
| Mode |  | 1 | See in “1.7” |
| Speed | 0～9 | 1 | The smaller of the value, the faster. Invalid when display immediately. |
| Stay time | 0x0000~0xffff | 2 | High byte previous (big endian). In seconds. |

1.14、The meaning of each byte of the scan parameters

A total of 16 bytes of scan parameters, set the scanning parameters and read the scan parameters to be used, the meaning of each byte is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Byte | Byte  Meaning | CPower3200/2200/1200 Value Description | CPower5200/4200 Value  Description |
| 0x00 | Column order | 0:Positive(+),1:Negvtive(-) | 0:Positive(+),1:Negvtive(-) |
| 0x01 | Data polarity | 0:Positive(+),1:Negvtive(-) | 0:Positive(+),1:Negvtive(-) |
| 0x02 | OE polarity | CPower3200/2200:This parameter does not exist , is set to 0.  CPower1200:0:Positive(+),1:Negvtive(-) | 0:Positive(+),1:Negvtive(-) |
| 0x03 | Line adjust | 0:-1,1:0,2:1,3:2 | 0:0,1:1,2:2,3:-1 |
| 0x04 | Hide scan | 0:No，1：Yes | 0:No hide，1:Hide front，  2:Hide back，3:Hide both |
| 0x05 | Color order | 0：Red-Green，1：Green-Red | 0:Red-Green-Blue ，  1:Red-Blue-Green，  2: Green-Red-Blue,  3：Green-Blue-Red，  4：Blue-Red-Green，  5：Blue-Green-Red |
| 0x06 | Color mode | 0：6Mhz，1：12Mhz | 0~15：Mode 1~Mode 16 |
| 0x07 | Timing trimming | This parameter does not exist , is set to 0. | 0：1,1：2,2：3,3：4 |
| 0x08 | Pulse trimming | This parameter does not exist , is set to 0. | 0：1,1：2,2：3,3：4 |
| 0x09 | Scan mode | 0：1/16，1：1/8，2：1/4，3：1/2，4：Static | 0：1/16，1：1/8，2：1/4，3：  1/2，4：Static |
| 0x0A | Module size | 0：16-Line，1：8-Line，2：4-Line，3：2-Line，  4：1-Line | 0：16-Line，1：8-Line，2： 4-Line，3：2-Line，4：1-Line |
| 0x0B | Line change space | 0：Every 4，1：Every 8，2：Every 16，3：  Every 32 | 0：Every 8，1：Every 4，2：  Every 16，3：Every 32 |
| 0x0C | Line change direction | 0:Positive(+),1:Negvtive(-) | 0:Positive(+),1:Negvtive(-) |
| 0x0D | Signal reverse | 0:None，  1:Odd line reverse，  2:Even linereverse，  3:All | 0：None，1：Reverse 8-pixel，  2：Reverse 4-pixel，  3：Reverse 16-pixel，  4：Reverse 32-pixel， |
| 0x0E | Output board | 0:Normal，1:Extend | 0：Type 1,1：Type 2,  2：Type 3,3：Type 4 |
| 0x0F | Line reverse | This parameter does not exist , is set to 0. | 0 ： None,2 ： Even line reverse,3: Odd line reverse |

1.15、The meaning of each byte of the parameters of formatted text item

Formatted text control data

|  |  |  |
| --- | --- | --- |
| Data Item | Length(  BYTE) | Description |
| Data length | 2 | Formatted text control data length(little endian)  (except data item “Length”) |
| Font point size | 2 | Font point size(little endian) |
| Flag | 1 | Bit0-7:  Bit 0：Bold  Bit 1：Italic  Bit 2：Underline |
| Align mode | 1 | Bit0~1: Horizontal( 0:Left, 1:horizontal center, 2:right )  Bit2~3: Vertical(0:Top, 1: vertical center, 2: down ）  Others: Reserved |
| Paging | 1 | Bit0~1: Paging mode（0：No，1：Horizontal，2：Vertical）  Bit 2：Break word between pages( 1: Yes; 0: No )  Bit 3：Image Cover whole page or not( 1: Yes; 0: No ) |
| Font forground color | 4 | BYTE 0：RED；1：GREEN；2：BLUE；3：0 |
| Font background color | 4 | BYTE 0：RED；1：GREEN；2：BLUE；3：0 |
| Page background color | 4 | BYTE 0：RED；1：GREEN；2：BLUE；3：0 |
| Row height | 1 | Row height of every row of string( by pixel ) |
| Y-Offset | 1 | Offset of vertical direction( by pixel ) |

Screen data

|  |  |  |
| --- | --- | --- |
| Data Item | Length(  BYTE) | Description |
| Data length | 2 | Screen data length( little endian )  (except data item “Length”) |
| Screen width | 2 | Pixel( little endian ) |
| Screen height | 2 | Pixel( little endian ) |
| Color | 1 | The color data and data format that contain in the Image data .  The lower 4 bit show what color data of exist. Can be a combination of the following values |
|  |  | 0x01: Red data is exist.  0x02: Green data is exist  0x04: Blue data is exist  The hign 4 bit show dormat of data(Gray-level), support two formats  0x0: Binary image. Line in accordance with the level of images, each of the eight data points to form a byte, the less than 8 points at the end to make up a byte 0; the amount of a color data (unit: bytes) is: ((picture width + 7)  / 8) \* picture height.  0x7: 256 gray-scale data. Each point expressed by 1 byte; the amount of a color data (unit: bytes) is: image width \* image height.    For example：0x71 show 256 gray picture, only exist red data |

1.16、The meaning of each byte of the parameters of extent formatted text item Extent formatted text content

|  |  |  |
| --- | --- | --- |
| Data Item | Length(  BYTE) | Description |
| Data length | 2 | Screen data length( little endian )  (except data item “Length”) |
| Text encoding | 1 | 0x00：multibyte  0x01：widechar |
| Text segment count | 1 | Text’s segment count(one line string represents one segment) |
| Text segment | Variable-l ength | Reference to Text segment format definition |

Text segment format

|  |  |  |
| --- | --- | --- |
| Data Item | Length(  BYTE) | Description |
| Data length | 2 | Screen data length( little endian )  (except data item “Length”) |
| Align mode | 1 | Bit0~1: Horizontal( 0:Left, 1:horizontal center,  2:right ) |
| Substring count | 1 | Text segment’s substring count |
| Substring | Variable-l ength | Reference to Substring format definition |

Substring format

|  |  |  |
| --- | --- | --- |
| Data Item | Length(  BYTE) | Description |
| Data length | 2 | Screen data length( little endian )  (except data item “Length”) |
| Flag | 1 | Bit0-7:  Bit 0：Bold  Bit 1：Italic  Bit 2：Underline |
| Font forground color | 4 | BYTE 0：RED；1：GREEN；2：BLUE；3：0 |
| Font background color | 4 | BYTE 0：RED；1：GREEN；2：BLUE；3：0 |
| Font point size | 2 | Font point size(little endian) |
| Font facename length | 1 | Font facename length |
| Font facename | Variable-l ength | Font facename string, end with 0x00 |
| String length | 2 | String length(little endian) |
| String | Variable-l ength | String, end with 0x00 |

Extent formatted text control data

|  |  |  |
| --- | --- | --- |
| Data Item | Length(  BYTE) | Description |
| Data length | 2 | Formatted text control data length(little endian)  (except data item “Length”) |
| Align mode | 1 | Bit0~1: Vertical(0:Top, 1: vertical center, 2: down ） |
|  |  | Others: Reserved |
| Paging | 1 | Bit0~1: Paging mode（0：No，1：Horizontal，2：  Vertical）  Bit 2：Break word between pages( 1: Yes; 0: No )  Bit 3：Image Cover whole page or not( 1: Yes; 0:  No ) |
| Page background color | 4 | BYTE 0：RED；1：GREEN；2：BLUE；3：0 |
| Row height | 1 | Row height of every row of string( by pixel ) |
| Y-Offset | 1 | Offset of vertical direction( by pixel ) |

Extent screen data

|  |  |  |
| --- | --- | --- |
| Data Item | Length(  BYTE) | Description |
| Data length | 2 | Screen data length( little endian )  (except data item “Length”) |
| Screen width | 2 | Pixel( little endian ) |
| Screen height | 2 | Pixel( little endian ) |
| Color | 1 | The color data and data format that contain in the Image data .  The lower 4 bit show what color data of exist. Can be a combination of the following values  0x01: Red data is exist.  0x02: Green data is exist  0x04: Blue data is exist  The hign 4 bit show dormat of data(Gray-level), support two formats  0x0: Binary image. Line in accordance with the level of images, each of the eight data points to form a byte, the less than 8 points at the end to make up a byte 0; the amount of a color data (unit: bytes) is: ((picture width + 7) / 8) \* picture height.  0x7: 256 gray-scale data. Each point expressed by 1 byte; the amount of a color data (unit: bytes) is: image width \* image height. |
|  |  | For example：0x71 show 256 gray picture, only exist red data |

# API function for creating program file

2.1. Overview of program creating API functions

|  |  |  |
| --- | --- | --- |
| No. | Function name | Description |
| 1 | CP5200\_Program\_Create | Create program object |
| 2 | CP5200\_Program\_Destroy | Destroy program object |
| 3 | CP5200\_Program\_SetProperty | Set the attribute value of program object |
| 4 | CP5200\_Program\_SetBackgndImage | Set the background image of program object |
| 5 | CP5200\_Program\_AddPlayWindow | Add play window to program |
| 6 | CP5200\_Program\_SetWindowProperty | Set window property |
| 7 | CP5200\_Program\_SetItemProperty | Set play item property |
| 8 | CP5200\_Program\_AddText  CP5200\_Program\_AddText1 | Add text item to play window |
| 9 | CP5200\_Program\_AddTagText  CP5200\_Program\_AddTagText1 | Add text item of contain extend tag to play window |
| 10 | CP5200\_Program\_AddFormattedText | Add formatted text item to play window |
| 11 | CP5200\_Program\_AddFormattedTextW | Add formatted text item to play window(wide character) |
| 12 | CP5200\_Program\_AddFormattedTextEx | Add extent formatted text item to play window |
| 13 | CP5200\_Program\_AddPicture | Add picture item to play window |
| 14 | CP5200\_Program\_AddImage | Add image item to play window |
| 15 | CP5200\_Program\_AddLafPict | Add Laf picture item to play window |
| 16 | CP5200\_Program\_AddLafVideo | Add Laf animator item to play window |
| 17 | CP5200\_Program\_AddAnimator | Add animator item to play window |
| 18 | CP5200\_Program\_AddClock | Add clock item to play window |
| 19 | CP5200\_Program\_AddTemperature | Add temperature item to play window |
| 20 | CP5200\_Program\_AddVariable | Add custom variable data to play window |
| 21 | CP5200\_Program\_AddTimeCounter | Add time counter data to play window |
| 22 | CP5200\_Program\_SaveToFile | Save program to file |

Usage:

Step 1: Create program object

Step 2: Add play window

Step 3: Add play item to play window

Step 4: Save program to file

Step 5: Destroy program object

2.2. Detail of creating program file API functions

CP5200\_Program\_Create

|  |  |
| --- | --- |
| HOBJECT CP5200\_Program\_Create(WORD width, WORD height, BYTE color) | |
| Description | Create program object |
| Parameter | width: Width of the screen, unit is pixel |
| height: Height of the screen, unit is pixel |
| color: Color and gray-scale.  Bit0~2: 1 red color, 3 red & green color, 7 red , green and blue color  Bit4~6: gray scale. 0 (white or black), 7(256 grayscale)  For Example: 0x01(red color no gray), 0x77 full color, 256 gray scale) |
| Return | Handle of program object, all these kind of API functions use this handle  Return NULL if fail |
| Note | When an application no longer requires a given object, it should be destroyed to free the resource. |

CP5200\_Program\_Destroy

|  |  |
| --- | --- |
| int CP5200\_Program\_Destroy(HOBJECT hObj) | |
| Description | Destroy program object |
| Parameter | hObj: Handle of program object to be destroyed |
| Return | 0: No error  -1: Invalid program object handle |
| Note |  |

CP5200\_Program\_SetProperty

|  |  |
| --- | --- |
| int CP5200\_Program\_SetProperty(HOBJECT hObj, int nPropertyValue, DWORD nPropertyID) | |
| Description | Set the attribute value of program object |
| Parameter | hObj: Handle of program object |
| nPropertyValue: Attribute value ，depend on parameter“nPropertyID”have different meanings program repetition play times’s range is1~65535 program play time’s unit is second and range is 1~65535  Code conversion: 0: no conversion;1: simplified Chinese = > traditional Chinese;2: traditional Chinese = > simplified Chinese; |
| nPropertyID: Attribute identify，must be the one as below:  1: program repetition play times  2: program play time  3: Code conversion |
| Return | -1: Wrong handle of program object  0: Unacquainted Attribute identify  >0: Setting success |
| Note | “program repetition play times”and “program play time”,only one is virtuous and the lastly setting is virtuous。 |

CP5200\_Program\_SetBackgndImage

|  |  |
| --- | --- |
| int CP5200\_Program\_SetBackgndImage(HOBJECT hObj, const BYTE\* pImgDat, WORD wImgWidth,  WORD wImgHeight, BYTE color, int nMode, int nCompress) | |
| Description | Set the background image of program object |
| Parameter | hObj: Handle of program objec |

|  |  |
| --- | --- |
|  | pImgDat: Image data buffer。Determine the data of the color and data  format by the value of parameters “color”. Multi-color data exist, the first red data, add the green data, and finally the blue data.  Data for each color, to put the data of line one frist, add then line second , the data of each pixel based on parameters “color” high  4 bit to determine。 |
| wImgWidth: Picture width points |
| wImgHeight:Picture height points |
|  | color: The color data and data format that contain in the Image data . The lower 4 bit show what color data of exist. Can be a combination of the following values  0x01: Red data is exist.  0x02: Green data is exist  0x04: Blue data is exist  The hign 4 bit show dormat of data(Gray-level), support two formats  0x0: Binary image. Line in accordance with the level of images,  each of the eight data points to form a byte, the less than 8 points at the end to make up a byte 0; the amount of a color data (unit: bytes) is: ((picture width + 7) / 8) \* picture height.  0x7: 256 gray-scale data. Each point expressed by 1 byte; the  amount of a color data (unit: bytes) is: image width \* image height.  For example：0x71 show 256 gray picture, only exist red data |
|  | nMode: Display dispose mode  0: center  1: by scaling  2: Stretch  3: flat |
| nCompress: Compressed image data. Only support the non-compressed now  0：non-compressed |
| Return | -1: Invalid program object handle  -4 : Memory not enough |
| Note |  |

CP5200\_Program\_AddPlayWindow

|  |  |
| --- | --- |
| int CP5200\_Program\_AddPlayWindow(HOBJECT hObj, WORD x, WORD y, WORD cx, WORD cy) | |
| Description | Add play window to program |
| Parameter | hObj: Handle of program object |
| x: Start X of the play window |
| y: Start X of the play window |
| cx: Width of play window |
| cy: Height of play window |
| Return | >=0: Number of play window  -1: Invalid program object handle  -3: Argument error |
| Note |  |

CP5200\_Program\_SetWindowProperty

int CP5200\_Program\_SetWindowProperty(HOBJECT hObj, int nWinNo, int nPropertyValue, int nPropertyID);

|  |  |
| --- | --- |
| Description | Set window property |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window,base on 0 |
| nPropertyValue: property value  When the attribute ID is 1, meaning its value is as follows:  bit0 ~ 1: frame speed (the smaller the faster) bit2: border background color (1 green, 0 for black) bit3: Border foreground color (1 red, 0 for white) bit4 ~ 6: Border action mode (0 No border,1 Single point , 2 Dash, 3 Cross, 4 chase) bit7: Border rolling direction (0 clockwise, 1 counterclockwise)  When the attribute ID is 2, meaning its value is as follows:  0: still, 1: loop 2: Hide |
| nPropertyID: attribute ID, can be one of the following values:  1: Set the Properties window frame  2: Set the window waiting for the type |
| Return | 0: No error  -1: Invalid program object handle  -3: Invalid play window number  -4: Memory not enough |
| Note |  |

CP5200\_Program\_SetItemProperty

|  |  |
| --- | --- |
| int CP5200\_Program\_SetItemProperty(HOBJECT hObj, int nWinNo, int nItem , int nPropertyValue, int nPropertyID); | |
| Description | Set play item property |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window,base on 0 |
| nItem:Play item no |
|  | nPropertyValue: Color，RGB value same as windows macro RGB(r,g,b), Cancel transparent color when it is -1. |
| nPropertyID: Value of 1 means set the transparent color |
| Return | 0: success  -1: Invalid program object handle  -3: Invalid play window number |
| Note |  |

CP5200\_Program\_AddText (CP5200\_Program\_AddText1)

|  |  |
| --- | --- |
| int CP5200\_Program\_AddText(HOBJECT hObj, int nWinNo, const char\* pText, int nFontSize,  COLORREF crColor, int nEffect, int nSpeed, int nStay) | |
| Description | Add text item to play window |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window, base on 0 |
| pText: Text to be added |
| nFontSize: font size and style，see 1.7. Font size code and font style |
| crColor: Text color |
| nEffect: Show effect |
| nSpeed: Effect speed |
| nStay: Stay time in second |
| Return | >=0: Play item no  -1: Invalid program object handle  -3: Invalid play window number  -4: Memory not enough |
| Note | CP5200\_Program\_AddText1 is for single byte characters, ASCII and extended ASCII. |

CP5200\_Program\_AddTagText

(CP5200\_Program\_AddTagText1)

|  |  |
| --- | --- |
| int CP5200\_Program\_AddTagText(HOBJECT hObj, int nWinNo, const char\* pText, int nFontSize, COLORREF crColor, int nEffect, int nSpeed, int nStay) | |
| Description | Add text item of contain extend tag to play window |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window, base on 0 |
| pText: Text to be added |
| nFontSize: font size and style，see 1.7. Font size code and font style |
| crColor: Text color |
| nEffect: Show effect |
| nSpeed: Effect speed |
| nStay: Stay time in second |
| Return | >=0: Play item no  -1: Invalid program object handle  -3: Invalid play window number  -4: Memory not enough |
| Note | Default use the parameters value of the text size, color and other attribute values, if the text contains the extensible tag, from extensible tag, use the value of extensible tag specified.  CP5200\_Program\_AddTagText1 is for single byte characters, ASCII and extended ASCII. |

CP5200\_Program\_AddFormattedText

int CP5200\_Program\_AddFormattedText( HOBJECT hObj, int nWinNo, const char \*pText, const char \*pFontFaceName, const byte \*pFormatData, const byte \*pScreenData, int nMode, int nEffect, int nSpeed, int nStay, int nCompress )

|  |  |
| --- | --- |
| Description | Add formatted text item to play window |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window, base on 0 |
| pText: Text string |
| pFontFaceName：Font face name |
| pformatData：Formatted text control data |
| pScreenData：Screen data |
| nMode: Render mode  0: Center  1: Zoom to fit the window  2: Stretch to fit the window  3: Tile  4: Lefttop  8: Vertical multipage  11: Horizontal multi page (Align left)  12: Horizontal multi page (Align right) |
| nEffect: Show effect |
| nSpeed: Effect speed. 0 fastest |
| nStay: Stay time in second |
| nCompress: Compress picture data  0：Do not compress it  1：Convert to 256 color and compress the data |
| Return | >=0: Play item no  -1: Invalid program object handle  -3: Invalid play window number  -4: Memory not enough  -5: Invalid params related to format text data |
| Note |  |

CP5200\_Program\_AddFormattedTextW

|  |  |
| --- | --- |
| int CP5200\_Program\_AddFormattedTextW( HOBJECT hObj, int nWinNo, const wchar\_t \*pText, const char \*pFontFaceName, const byte \*pFormatData, const byte \*pScreenData, int nMode, int nEffect, int nSpeed, int nStay, int nCompress ) | |
| Description | Add formatted text item to play window(wide character) |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window, base on 0 |
| pText: Text string |
| pFontFaceName：Font face name |
| pformatData：Formatted text control data |
| pScreenData：Screen data |
| nMode: Render mode  0: Center  1: Zoom to fit the window  2: Stretch to fit the window  3: Tile  4: Lefttop  8: Vertical multipage  11: Horizontal multi page (Align left)  12: Horizontal multi page (Align right) |
| nEffect: Show effect |
| nSpeed: Effect speed. 0 fastest |
| nStay: Stay time in second |
| nCompress: Compress picture data  0：Do not compress it  1：Convert to 256 color and compress the data |
| Return | >=0: Play item no  -1: Invalid program object handle |
|  | -3: Invalid play window number  -4: Memory not enough  -5: Invalid params related to format text data |
| Note |  |

CP5200\_Program\_AddFormattedTextEx

|  |  |
| --- | --- |
| int CP5200\_Program\_AddFormattedTextEx( HOBJECT hObj, int nWinNo, const byte  \*pTextContent, const byte \*pFormatData, const byte \*pScreenData, int nMode, int nEffect, int nSpeed, int nStay, int nCompress ); | |
| Description | Add extent formatted text item to play window |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window, base on 0 |
| pTextContent: Extent formatted text content |
| pFormatData：Extent formatted text control data |
| pScreenData：Extent screen data |
| nMode: Render mode  0: Center  1: Zoom to fit the window  2: Stretch to fit the window  3: Tile  4: Lefttop  8: Vertical multipage  11: Horizontal multi page (Align left)  12: Horizontal multi page (Align right) |
| nEffect: Show effect |
| nSpeed: Effect speed. 0 fastest |
| nStay: Stay time in second |
|  | nCompress: Compress picture data  0：Do not compress it  1：Convert to 256 color and compress the data |
| Return | >=0: Play item no  -1: Invalid program object handle  -3: Invalid play window number  -4: Memory not enough  -5: Invalid params related to format text data |
| Note |  |

CP5200\_Program\_AddPicture

|  |  |
| --- | --- |
| int CP5200\_Program\_AddPicture(HOBJECT hObj, int nWinNo, const char\* pPictFile, int nMode, int nEffect, int nSpeed, int nStay, int nCompress) | |
| Description | Add picture item to play window |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window, base on 0 |
| pPictFile: Path and file name of the picture file |
| nMode: Render mode  0: Center  1: Zoom to fit the window  2: Stretch to fit the window  3: Tile  4: Lefttop  8: Vertical multipage  11: Horizontal multi page (Align left)  12: Horizontal multi page (Align right) |
| nEffect: Show effect |
| nSpeed: Effect speed. 0 fastest |
| nStay: Stay time in second |
|  | nCompress: Compress picture data  0：Do not compress it  1：Convert to 256 color and compress the data |
| Return | >=0: Play item no  -1: Invalid program object handle  -3: Invalid play window number  -4: Memory not enough |
| Note |  |

CP5200\_Program\_AddImage

|  |  |
| --- | --- |
| int CP5200\_Program\_AddImage(HOBJECT hObj, int nWinNo, const BYTE\* pImgDat, WORD wImgWidth, WORD wImgHeight, BYTE color, int nMode, int nEffect, int nSpeed, int nStay, int nCompress, int nPageCount) | |
| Description | Add image item to the play window |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window, base on 0 |
| pImgDat: Image data buffer。Determine the data of the color and data  format by the value of parameters “color”. Multi-color data exist, the first red data, add the green data, and finally the blue data.  Data for each color, to put the data of line one frist, add then line second ,  the data of each pixel based on parameters “color” high 4 bit to determine。 |
| wImgWidth: Picture width points. |
| wImgHeight: Picture height points. |

|  |  |
| --- | --- |
|  | color: The color data and data format that contain in the Image data . The lower 4 bit show what color data of exist. Can be a combination of the following values  0x01: Red data is exist.  0x02: Green data is exist  0x04: Blue data is exist  The hign 4 bit show dormat of data(Gray-level), support two formats  0x0: Binary image. Line in accordance with the level of images,  each of the eight data points to form a byte, the less than 8 points at the end to make up a byte 0; the amount of a color data (unit: bytes) is: ((picture width + 7) / 8) \* picture height.  0x7: 256 gray-scale data. Each point expressed by 1 byte; the  amount of a color data (unit: bytes) is: image width \* image height.  For example：0x71 show 256 gray picture, only exist red data |
| nMode: Render Moder  0: Center  1: Zoom to fit the window  2: Stretch to fit the window  3: Tile |
| nEffect: Show effect |
|  | nSpeed: Effect speed. 0 fastest |
| nStay: Stay time in second. |
| nCompress: Compress picture data  0：Do not compress it  1：Convert to 256 color and compress the data |
| nPageCount: The page count |
| Return | >=0: Play item no |
|  | -1: Invalid program object handle  -3: Invalid play window number  -4: Memory not enough |
| Note | The size of the image must fit the window size |

CP5200\_Program\_AddLafPict

|  |  |
| --- | --- |
| int CP5200\_Program\_AddLafPict(HOBJECT hObj, int nWinNo, const char\* pLafFile, int nMode, int nEffect, int nSpeed, int nStay, int nCompress) | |
| Description | Add Laf picture item to play window |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window, base on 0 |
| pLafFile: Path and file name of the laf picture file |
| nMode: Render mode  0: Center  1: Zoom to fit the window  2: Stretch to fit the window  3: Tile |
| nEffect: Show effect |
| nSpeed: Effect speed. 0 fastest |
| nStay: Stay time in second |
| nCompress: Compress picture data  0：Do not compress it  1：Convert to 256 color and compress the data |
| Return | >=0: Play item no  -1: Invalid program object handle  -3: Invalid play window number  -4: Memory not enough |
| Note |  |

CP5200\_Program\_AddLafVideo

|  |  |
| --- | --- |
| int CP5200\_Program\_AddLafVideo(HOBJECT hObj, int nWinNo, const char\* pLafFile, int nMode, int nRepeat) | |
| Description | Add Laf animator item to play window |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window, base on 0 |
| pLafFile: Path and file name of laf video file |
| nMode: Render mode  0: Center  1: Zoom to fit the window  2: Stretch to fit the window  3: Tile |
| nRepeat: Repeat time. 1 time, 2 times, … |
| Return | >=0: Play item no  -1: Invalid program object handle  -3: Invalid play window number  -4: Memory not enough |
| Note |  |

CP5200\_Program\_AddAnimator

|  |  |
| --- | --- |
| int CP5200\_Program\_AddAnimator(HOBJECT hObj, int nWinNo, const char\* pAniFile, int nMode, int nRepeat) | |
| Description | Add animator item to play window |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window, base on 0 |
| pAniFile: Path and file name of gif file |
|  | nMode: Render mode  0: Center  1: Zoom to fit the window  2: Stretch to fit the window  3: Tile |
| nRepeat: Repeat time. 1 time, 2 times, … |
| Return | >=0: Play item no  -1: Invalid program object handle  -3: Invalid play window number  -4: Memory not enough |
| Note |  |

CP5200\_Program\_AddClock

|  |  |
| --- | --- |
| int CP5200\_Program\_AddClock(HOBJECT hObj, int nWinNo, const char\* pText, int nFontSize,  COLORREF crColor, int nStay, WORD wAttr) | |
| Description | Add clock item to play window |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window, base on 0 |
| pText: Text string |
| nFontSize: font size and style，see 1.7. Font size code and font style |
| crColor: Text color |
| nStay: Stay time in second |
|  | wAttrib: Clock attribute. By bit to determine the content to display.  bit 0: Show year bit 1: Show month bit 2: Show day bit 3:Show hour bit 4:Show minute bit 5:Show second bit 6:Show week bit 7:Show clock hand bit 8: when the system (0: 12 hour; 1: 24 hours system) bit 9: Year digit (0: 4; 1: 2) bit 10: Branch (0: single; 1: multi-line) bit 11~13: Format control, such as the November 12, 2010 Friday , according to diffenert values expressed as:  0: 2010/11/12 Friday 16:20:30  1: Fri，12/11/2010 16:20:30  2: 2010-11-12 Fri. 16:20:30  3: Friday，12 November 2010 16:20:30 4: Fri，Nov 12,2010 16:20:30  5: Friday，November 12 2010 16:20:30  6: Fri，11/12/2010 16:20:30  7: 2010/11/12，Fri.16:20:30 bit 14: show hands,marks bit 15 Transparent |
| Return | >=0: Play item no  -1: Invalid program object handle  -3: Invalid play window number  -4: Memory not enough |
| Note |  |

CP5200\_Program\_AddTemperature

int CP5200\_Program\_AddTemperature(HOBJECT hObj, int nWinNo, const char\* pText, int nFontSize, COLORREF crColor, int nStay, WORD wAttr)

|  |  |
| --- | --- |
| Description | Add temperature item to play window |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window, base on 0 |
| pText: Text string |
| nFontSize: font size and style，see 1.7. Font size code and font style |
| crColor: Text color |
| nStay: Stay time in second |
| wAttrib: Temperature attribute  0: Celsius degree  1: Fahrenheit degree |
| Return | >=0: Play item no  -1: Invalid program object handle  -3: Invalid play window number  -4: Memory not enough |
| Note |  |

CP5200\_Program\_AddVariable

|  |  |
| --- | --- |
| int CP5200\_Program\_AddVariable(HOBJECT hObj, int nWinNo, int nFontSize, COLORREF crColor, int nStay, WORD wAttr) | |
| Description | Add custom variable data to play window |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window, base on 0 |
| nFontSize: font size and style，see 1.7. Font size code and font style |
| crColor: Font Color |
| nStay: StayTime |
|  | wAttrib: Custom data attributes  High-byte format: the following: 0: text data.Direct display variable text 1：picture data.Variable text shows the low byte of the specified image as a variable number, valid values 1 to 100, specify which user variables |
| Return | >=0: Play item no  -1: Invalid program object handle  -3: Invalid play window number  -4: Memory not enough |
| Note |  |

CP5200\_Program\_AddTimeCounter

|  |  |
| --- | --- |
| int CP5200\_Program\_AddTimeCounter(HOBJECT hObj, int nWinNo, int nFontSize, COLORREF crColor, int nStay, int nOption , const int\* pBaseTime ,const char\* pContent) | |
| Description | Add time counter data to play window |
| Parameter | hObj: Handle of program object |
| nWinNo: Number of play window, base on 0 |
| nFontSize: font size and style，see 1.7. Font size code and font style |
| crColor: Font Color |
| nStay: StayTime |
| nOption: Display properties  Byte 1：Format，  Bit0: time counter type。0:up timer，1 countdown Bit1~7: reserved  Byte 2 ：Align,  Bit0~1:horizontal alignment;Bit2~3:vertical alignment |
| pBaseTime: Integer array Pointer , the array Length is 6,  store year,month,day,hour,minute,second |
| pContent: |
| Return | >=0: Play item no  -1: Invalid program object handle |
|  | -3: Invalid play window number  -4: Memory not enough |
| Note |  |

CP5200\_Program\_SaveToFile

|  |  |
| --- | --- |
| int CP5200\_Program\_SaveToFile(HOBJECT hObj, const char\* pFilename) | |
| Description | Save program to file |
| Parameter | hObj: Handle of program object |
| pFilename: Path and file name |
| Return | 0: No error  -1: Invalid program object handle  -3: File create error |
| Note |  |

# API function for creating playbill file

3.1. Overview of playbill creating API function

|  |  |  |
| --- | --- | --- |
| No. | Function name | Description |
| 1 | CP5200\_Playbill\_Create | Create playbill object |
| 2 | CP5200\_Playbill\_Destroy | Destroy playbill object |
| 3 | CP5200\_Playbill\_SetProperty | Set the attribute value of playbill object |
| 4 | CP5200\_Playbill\_AddFile | Add program file to playbill |
| 5 | CP5200\_Playbill\_DelFile | Delete program file from playbill |
| 6 | CP5200\_Playbill\_SaveToFile | Save playbill to file |

Usage:

Step 1: Create playbill object

Step 2: Add program file to playbill

Step 3: Save playbill to file

Step 4: Destroy playbill object

3.2. Detail of creating playbill file functions

CP5200\_Playbill\_Create

|  |  |
| --- | --- |
| HOBJECT CP5200\_Playbill\_Create(WORD width, WORD height, BYTE color) | |
| Description | Create playbill object |
| Parameter | width: Screen width, unit is pixel |
| height: Screen height, unit is pixel |
| color: Color and gray-scale.  Bit0~2: 1 red color, 3 red & green color, 7 red , green and blue color Bit4~6: gray scale. 0 (white or black), 7(256 grayscale)  For Example: 0x01(red color no gray), 0x77 full color, 256 gray scale) |
| Return | Handle of playbill object, all these kind of API functions use this handle  Return NULL if fail |
| Note | When an application no longer requires a given object, it should be destroyed to free the resource. |

CP5200\_Playbill\_Destroy

|  |  |
| --- | --- |
| int CP5200\_Playbill\_Destroy(HOBJECT hObj) | |
| Description | Destroy playbill object |
| Parameter | hObj: Handle of playbill object to be destroyed |
| Return | 0: No error  -1: Invalid playbill object handle |
| Note |  |

CP5200\_Playbill\_SetProperty

|  |  |
| --- | --- |
| int CP5200\_Playbill\_SetProperty(HOBJECT hObj, int nPropertyValue, DWORD nPropertyID) | |
| Description | Set the attribute value of playbill object |
| Parameter | hObj: Handle of program object |
| nPropertyValue: Attribute value ，depend on parameter“nPropertyID”have different meanings rotation : 0: Don’t rotate ; 1: Rotate 90 degrees |
| nPropertyID: Attribute identify，must be the one as below:  1: rotation |
| Return | -1: Wrong handle of program object  0: Unacquainted Attribute identify  >0: Setting success |
| Note |  |

CP5200\_Playbill\_AddFile

|  |  |
| --- | --- |
| int CP5200\_Playbill\_AddFile(HOBJECT hObj, const char\* pFilename) | |
| Description | Add program file to playbill |
| Parameter | hObj: Handle of playbill object |
| pFilename: Path and file name of program file |
| Return | >=0: Add file success  -1: Invalid playbill object handle  -3: Not short file name |
| Note |  |

CP5200\_Playbill\_DelFile

|  |  |
| --- | --- |
| int CP5200\_Playbill\_DelFile(HOBJECT hObj, const char\* pFilename) | |
| Description | Delete program file from play bill |
| Parameter | hObj: Handle of playbill object |
| pFilename: Path and file name |
| Return | 0: Delete success  -1: Invalid playbill object handle |
|  | -3: File create error |
| Note |  |

CP5200\_Playbill\_SaveToFile

|  |  |
| --- | --- |
| int CP5200\_Playbill\_SaveToFile(HOBJECT hObj, const char\* pFilename) | |
| Description | Save playbill to file |
| Parameter | hObj: Handle of playbill object |
| pFilename: Path and file name |
| Return | 0: No error  -1: Invalid playbill object handle  -3: File create error |
| Note |  |

# API function for data communication

4.1. Overview of data communication API function

|  |  |  |
| --- | --- | --- |
| No. | Function name | Description |
| 1 | CP5200\_CommData\_Create | Create communication data object |
| 2 | CP5200\_CommData\_Destroy | Destroy communication data object |
| 3 | CP5200\_CommData\_SetParam | Set data packet parameter |
| 4 | CP5200\_MakeCreateFileData | Make create file command data |
| 5 | CP5200\_ParseCreateFileRet | Parse return data of create file |
| 6 | CP5200\_MakeWriteFileData | Make write file command data |
| 7 | CP5200\_ParseWriteFileRet | Parse return data of write file |
| 8 | CP5200\_MakeCloseFileData | Make close file command data |
| 9 | CP5200\_ParseCloseFileRet | Parse return data of close file |
| 10 | CP5200\_MakeDeleteFileNoData | Make delete file command data (By file number) |
| 11 | CP5200\_ParseDeleteFileNoRet | Parse return data of delete file by file number |
| 12 | CP5200\_MakeDeleteFileNameData | Make delete file command data (By file name) |

|  |  |  |
| --- | --- | --- |
| 13 | CP5200\_ParseDeleteFileNameRet | Parse return data of delete file by file name |
| 14 | CP5200\_MakeReadTimeData | Make query controller time command data |
| 15 | CP5200\_ParseReadTimeRet | Parse return data of query controller time |
| 16 | CP5200\_MakeWriteTimeData | Make set controller time command data |
| 17 | CP5200\_ParseWriteTimeRet | Parse return data of set controller time |
| 18 | CP5200\_MakeReadBrightnessData | Make query brightness setting command data |
| 19 | CP5200\_ParseReadBrightnessRet | Parse return data of set brightness |
| 20 | CP5200\_MakeWriteBrightnessData | Make set brightness command data |
| 21 | CP5200\_ParseWriteBrightnessRet | Parse return data of set brightness |
| 22 | CP5200\_MakeWriteIOOnOffTimeData | Make set IO timing control command data |
| 23 | CP5200\_ParseWriteIOOnOffTimeRet | Parse return data of set IO timing control |
| 24 | CP5200\_MakeReadIOOnOffTimeData | Make query IO timing control information |
| 25 | CP5200\_ParseReadIOOnOffTimeRet | Parse query IO timing control information |
| 26 | CP5200\_MakeWriteOnOffTimeData | Make set auto ONOFF control command data |
| 27 | CP5200\_ParseWriteOnOffTimeRet | Parse return data of set auto ONOFF control |
| 28 | CP5200\_MakeReadOnOffTimeData | Make query auto ONOFF control  information command data |
| 29 | CP5200\_ParseReadOnOffTimeRet | Parse return data of query auto ONOFF control information |
| 30 | CP5200\_MakeReadVersionData | Make query version information command data |
| 31 | CP5200\_ParseReadVersionRet | Parse return data of query version information |
| 32 | CP5200\_MakeFormatData | Make format controller file system command data |
| 33 | CP5200\_ParseFormatRet | Parse return data of format controller file system |
| 34 | CP5200\_MakeRestartAppData | Make restart Appcommand data |
| 35 | CP5200\_ParseRestartAppRet | Parse return data of restart App |
| 36 | CP5200\_MakeRestartSysData | Make restart controller command data |
| 37 | CP5200\_ParseRestartSysRet | Parse return data of restart controller |
| 38 | CP5200\_MakeGetFreeSpaceData | Make query free space in controller command data |
| 39 | CP5200\_ParseGetFreeSpaceRet | Parse return data of query free space in controller |
| 40 | CP5200\_MakeGetFileInfoData | Make query file information command data |
| 41 | CP5200\_ParseGetFileInfoRet | Parse return data of query file information |
| 42 | CP5200\_ParseGetFirstFileInfoRet | Parse return data of query file information and get first file information |
| 43 | CP5200\_ParseGetNextFileInfoRet | Parse return data of query file information |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | and get next file information | |
| 44 | CP5200\_MakeBeginFileUploadData | Make start upload file command data | |
| 45 | CP5200\_ParseBeginFileUploadRet | Parse return data of start upload file command | |
| 46 | CP5200\_MakeFileUploadData | Make upload file command data | |
| 47 | CP5200\_ParseFileUploadRet | Parse return data of upload file command | |
| 48 | CP5200\_MakeEndFileUploadData | Make finish upload file command data | |
| 49 | CP5200\_ParseEndFileUploadRet | Parse return data of finish upload file command | |
| 50 | CP5200\_MakeGetTypeInfoData | Make query type information command data | |
| 51 | CP5200\_ParseGetTypeInfoRet | Parse return data of query type information | |
| 52 | CP5200\_MakeGetTempHumiData | Make query temperature and humidity information command data | |
| 53 | CP5200\_ParseGetTempHumiRet | Parse return data of query temperature information | |
| 54 | CP5200\_MakeReadConfigData | Make read configuration information command data | |
| 55 | CP5200\_ParseReadConfigRet | Parse return data of read configuration information | |
| 56 | CP5200\_MakeWriteConfigData | Make write configuration information command data | |
| 57 | CP5200\_ParseWriteConfigRet | Parse return data of write configuration information | |
| 58 | CP5200\_MakeReadRunningInfoData | Make query running info data | |
| 59 | CP5200\_ParseReadRunningInfoRet | Parse return value of query running info command | |
| 60 | CP5200\_MakeScreenTestData | Make show test pattern data | |
| 61 | CP5200\_ParseScreenTestRet | Parse return value of q show test pattern command | |
| 62 | CP5200\_MakeInstantMessageData  CP5200\_MakeInstantMessageData1 | Make instant message data | |
| 63 | CP5200\_MakeSendInstantMessageData | Make send instant message data | |
| 64 | CP5200\_ParseSendInstantMessageRet | Parse return value of send instant message command | |
| 65 | CP5200\_MakeReadHWSettingData | Make read scan param command data | |
| 66 | CP5200\_ParseReadHWSettingRet | Parse return data of read scan param | |
| 67 | CP5200\_MakeWriteHWSettingData | Make write scan param command data | |
| 68 | CP5200\_ParseWriteHWSettingRet | Parse return data of write scan param | |
| 69 | CP5200\_MakeReadSoftwareSwitchInfoData | Make read software switch info data | |
| 70 | CP5200\_ParseReadSoftwareSwitchInfoRet | Parse return data of read software switch info data | |
| 71 | CP5200\_MakeWriteSoftwareSwitchInfoData | Make write software switch info data | |
| 72 | CP5200\_ParseWriteSoftwareSwitchInfoRet | Parse return data of write software switch info data | |
| 73 | CP5200\_MakeQueryControllerInfo | Make query controller information data |  |
| 74 | CP5200\_ParseQueryControllerInfoRet | Parse return data of query controller information data |
| 75 | CP5200\_MakeOpenFileData | Make open file data | |
| 76 | CP5200\_ParseOpenFileRet | Parse return value of open file command | |
| 77 | CP5200\_MakeGetDirentryData | Make get file info | |
| 78 | CP5200\_ParseGetDirentryRet | Parse return value of get file info command | |
| 79 | CP5200\_MakeReadFileNoData | Make read file data | |
| 80 | CP5200\_ParseReadFileNoRet | Parse return value of read file command | |
| 81 | CP5200\_MakeCloseFileNoData | Make close file data | |
| 82 | CP5200\_ParseCloseFileNoRet | Parse return value of close file command | |
|  |  |  | |

Usage:

Step 1: Create data object

Step 2: Make communication data, include RS232/485’s code convert

(0xa5 => 0xaa 0x05, …), or network ID code

Step 3: Send communication data to the controller

Step 4: Receive data from controller, and process code convert (0xaa

0x05 => 0xa5, …)

Step 5: Parse the return data and get the result

Step 6: Destroy data object

4.2. Detail of data communication API functions

CP5200\_CommData\_Create

|  |  |
| --- | --- |
| HOBJECT CP5200\_CommData\_Create(int nCommType, BYTE byCardID, DWORD dwIDCode) | |
| Description | Create communication data object |
| Parameter | nCommType: RS232/485 or network communication type  0: RS232/485  1: Network |
| byCardID: Controller ID |
| dwIDCode: Network ID code of the controller. RS232 ignore it. |
| Return | Handle of communication data object, all these kind of API functions use |
|  | this handle  Return NULL if fail |
| Note | When an application no longer requires a given object, it should be destroyed to free the resource. |

CP5200\_CommData\_Destroy

|  |  |
| --- | --- |
| HOBJECT CP5200\_CommData\_Destroy(HOBJECT hObj) | |
| Description | Destroy communication data object |
| Parameter | hObj: Handle of communication data object to de destroyed |
| Return | 0: No error  -1: Invalid data object handle |
| Note |  |

CP5200\_CommData\_SetParam

|  |  |
| --- | --- |
| HOBJECT CP5200\_CommData\_SetParam (HOBJECT hObj, int nParamType, const char \*pParamString) | |
| Description | Set data packet parameter |
| Parameter | hObj: Handle of communication data |
| nParamType：Parameter type，valid value 1. |
| pParamString：Parameter string  When parameter type is 1，pParamString is controller’s device ID |
| Return | 1: No error  0: Parameter type is wrong  -1: Invalid data object handle  -2: pParamString is wrong |
| Note |  |

CP5200\_MakeCreateFileData

|  |  |
| --- | --- |
| int CP5200\_MakeCreateFileData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize, const char\* pFilename, long lFilesize, const BYTE\* pTimeBuffer) | |
| Description | Make create file command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| pFilename: File name to be created. Must be short name |
| lFilesize: Size of the new file (BYTE) |
| pTimeBuffer: File time information, the length is 6  Byte 0: Year-2000 (00~99), the year value plus 2000 is the real year value  Byte 1: Month (1~12)  Byte 2: Day (1~31)  Byte 3: Hour(0~23)  Byte 4: Minute (0~59)  Byte 5: Second (0~59) |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note | If these is an old file in the controller, it will be overwritten  The maximum file size is 1.5M byte  Only ONE file can be read or write at the same time |

CP5200\_ParseCreateFileRet

|  |  |
| --- | --- |
| int CP5200\_ParseCreateFileRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of create file |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
|  | nLength: Length of the return data |
| Return | 1: File is created successfully  0: Can not create file  -2: Incorrect return data  -3: Return data length is not enough |
| Note |  |

CP5200\_MakeWriteFileData

|  |  |
| --- | --- |
| int CP5200\_MakeWriteFileData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize, const BYTE \*pData,  WORD wDatLen, WORD \*pwChksum) | |
| Description | Make write file command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| pData: File data buffer |
| wDataLen: Data length |
| pwChksum: WORD type pointer to a checksum variable, input old checksum value and return new checksum value |
| Return | >0: Length of the output data  -2: Incorrect return data  -3: Return data length is not enough |
| Note | If a file is large, it should be split to blocks and write one block each time, each block no more than 1000 bytes |

CP5200\_ParseWriteFileRet

|  |  |
| --- | --- |
| int CP5200\_ParseWriteFileRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of write file |
| Parameter | hObj: Handle of communication data |
|  | pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1: Success  0: Fail  -2: Incorrect return data  -3: Return data length is not enough |
| Note | Controller saved the received file data in a temporary buffer, and the data is written to file when file closing |

CP5200\_MakeCloseFileData

|  |  |
| --- | --- |
| int CP5200\_MakeCloseFileData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize, WORD wChksum) | |
| Description | Make close file command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| wChksum: Checksum of file data |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note | Closing file need to write all file data to the disk, it need sometime to do this, the time is about：  ((file size / 4096) + 1) \* 200 + 100 ms |

CP5200\_ParseCloseFileRet

|  |  |
| --- | --- |
| int CP5200\_ParseCloseFileRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of close file |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1: Success  0: Fail  255: Checksum error  -2: Incorrect return data  -3: Return data length is not enough |
| Note |  |

CP5200\_MakeDeleteFileNoData

|  |  |
| --- | --- |
| int CP5200\_MakeDeleteFileNoData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize, int fno) | |
| Description | Make delete file command data (By file number) |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| fno: File number |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note |  |

CP5200\_ParseDeleteFileNoRet

|  |  |
| --- | --- |
| int CP5200\_ParseDeleteFileNoRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of delete file by file number |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1: Success  0: Fail  -2: Incorrect return data |
|  | -3: Return data length is not enough |
| Note |  |

CP5200\_MakeDeleteFileNameData

|  |  |
| --- | --- |
| int CP5200\_MakeDeleteFileNameData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize, const char  \*pFilename) | |
| Description | Make delete file command data (By file name) |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| pFilename: File name |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note |  |

CP5200\_ParseDeleteFileNameRet

|  |  |
| --- | --- |
| int CP5200\_ParseDeleteFileNameRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of delete file by file name |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1: Success  0: Fail  -2: Return wrong data  -3: The length of return data is not enough |
| Note |  |

CP5200\_MakeReadTimeData

|  |  |
| --- | --- |
| int CP5200\_MakeReadTimeData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize) | |
| Description | Make query controller time command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note |  |

CP5200\_ParseReadTimeRet

|  |  |
| --- | --- |
| int CP5200\_ParseReadTimeRet (HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pTimeBuffer, int nTimeBufSize) | |
| Description | Parse return data of query controller time |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
|  | pTimeBuffer: Time information buffer，the meaning is:  Byte 0: Second  Byte 1: Minute  Byte 2: Hour  Byte 3: Week day  Byte 4: Day  Byte 5: Month  Byte 6: Year(2-year, plus 2000 is real year value) |
| nTimeBufSize: Length of time information buffer, at least 7 bytes |
| Return | 1: Success  0: Fail  -2: Return wrong data  -3: The length of return data is not enough |
| Note |  |

CP5200\_MakeWriteTimeData

|  |  |
| --- | --- |
| int CP5200\_MakeWriteTimeData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize, const BYTE\* pTimeBuffer) | |
| Description | Make set controller time command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
|  | pTimeBuffer: Time information data buffer, ，the meaning is:  Byte 0: Second  Byte 1: Minute  Byte 2: Hour  Byte 3: Week day  Byte 4: Day  Byte 5: Month  Byte 6: Year(2-year, plus 2000 is real year value) |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note |  |

CP5200\_ParseWriteTimeRet

|  |  |
| --- | --- |
| int CP5200\_ParseWriteTimeRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of set controller time |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1: Success  0: Fail  -2: Return wrong data  -3: The length of return data is not enough |
| Note |  |

CP5200\_MakeReadBrightnessData

|  |  |
| --- | --- |
| int CP5200\_MakeReadBrightnessData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize) | |
| Description | Make query brightness setting command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note |  |

CP5200\_ParseReadBrightnessRet

|  |  |
| --- | --- |
| int CP5200\_ParseReadBrightnessRet (HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pBrightnessBuffer, int nBrightBufSize) | |
| Description | Parse return data of query brightness setting |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| pBrightnessBuffer: Brightness information buffer, one byte one hour’s brightness, total 24 bytes. Each byte has the meaning:  Value 0~31: Brightness level  Value >31: Auto brightness by light sensor |
| nBrightBufSize: Brightness information buffer size, at least 24 bytes |
| Return | 1: Success  0: Fail  -2: Return wrong data  -3: The length of return data is not enough |
| Note |  |

CP5200\_MakeWriteBrightnessData

int CP5200\_MakeWriteBrightnessData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize, const

|  |  |
| --- | --- |
| BYTE \*pBrightnessBuffer) | |
| Description | Make set brightness command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| pBrightnessBuffer: Brightness setting information buffer, one byte one hour’s brightness, total 24 bytes. Each byte has the meaning:  Value 0~31: Brightness level  Value >31: Auto brightness by light sensor |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note |  |

CP5200\_ParseWriteBrightnessRet

|  |  |
| --- | --- |
| int CP5200\_ParseWriteBrightnessRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of set brightness |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1: Success  0: Fail  -2: Return wrong data  -3: The length of return data is not enough |
| Note |  |

CP5200\_MakeWriteIOOnOffTimeData

int CP5200\_MakeWriteIOOnOffTimeData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize, const

|  |  |
| --- | --- |
| BYTE \*pOnOffBuffer) | |
| Description | Make set IO timing control command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| pOnOffBuffer: IO timing control information buffer, at least 4 bytes  Byte 0~1: Hour and minute of “ON”  Byte 2~3: Hour and minute of “OFF”  If “ON” time and ”OFF” time is same, it will always “ON”; if hour large than 23 or minute large than 59, the time invalid |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note | IO signal is out put from J3 pin5 and pin6 |

CP5200\_ParseWriteIOOnOffTimeRet

|  |  |
| --- | --- |
| int CP5200\_ParseWriteIOOnOffTimeRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of set IO timing control |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1: Success  0: Fail  -2: Return wrong data  -3: The length of return data is not enough |
| Note |  |

CP5200\_MakeReadIOOnOffTimeData

|  |  |
| --- | --- |
| int CP5200\_MakeReadIOOnOffData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize) | |
| Description | Make query IO timing control information |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note |  |

CP5200\_ParseReadIOOnOffTimeRet

|  |  |
| --- | --- |
| int CP5200\_ParseReadIOOnOffTimeRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength,  BYTE\* pOnOffBuffer, int nOnOffBufSize) | |
| Description | Parse query IO timing control information |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| pOnOffBuffer: IO timing control information buffer, total 4 bytes  Byte 0~1: Hour and minute of “ON”  Byte 2~3: Hour and minute of “OFF”  If “ON” time and ”OFF” time is same, it will always “ON”; if hour large than 23 or minute large than 59, the time invalid |
| nOnOffBufSize: IO timing control information buffer size, at least 4 bytes |
| Return | 1: Success  0: Fail  -2: Return wrong data  -3: The length of return data is not enough |
|  | -4: The size of information buffer is too small  -5: Checksum error |
| Note |  |

CP5200\_MakeWriteOnOffTimeData

|  |  |
| --- | --- |
| int CP5200\_MakeWriteOnOffTimeData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize, const BYTE  \*pOnOffBuffer) | |
| Description | Make set auto ONOFF control command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| pOnOffBuffer: Auto ONOFF control information buffer， total 6 bytes  Byte 0~1: Hour and minute of “ON”  Byte 2~3: Hour and minute of “OFF”  Byte 4~5: Reserve, set to 0  If “ON” time and ”OFF” time is same, it will always “ON”; if hour large than 23 or minute large than 59, the time invalid |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note |  |

CP5200\_ParseWriteOnOffTimeRet

|  |  |
| --- | --- |
| int CP5200\_ParseWriteOnOffTimeRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of set auto ONOFF control |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1: Success  0: Fail  -2: Return wrong data  -3: The length of return data is not enough |
| Note |  |

CP5200\_MakeReadOnOffTimeData

|  |  |
| --- | --- |
| int CP5200\_MakeReadOnOffTimeData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize) | |
| Description | Make query auto ONOFF control information command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note |  |

CP5200\_ParseReadOnOffTimeRet

|  |  |
| --- | --- |
| int CP5200\_ParseReadOnOffTimeRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pOnOffBuffer, int nOnOffBufSize) | |
| Description | Parse return data of query auto ONOFF control information |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
|  | pOnOffBuffer: Auto ONOFF control information buffer, total 6 bytes  Byte 0~1: Hour and minute of “ON”  Byte 2~3: Hour and minute of “OFF”  Byte 4~5: Reserve, default is 0  If “ON” time and ”OFF” time is same, it will always “ON”; if hour large than 23 or minute large than 59, the time invalid |
| nOnOffBufSize: Auto ONOFF control information buffer size, at least  6bytes |
| Return | 1: Success  0: Fail  -2: Return wrong data  -3: The length of return data is not enough  -4: The size of information buffer is too small  -5: Checksum error |
| Note |  |

CP5200\_MakeReadVersionData

|  |  |
| --- | --- |
| int CP5200\_MakeReadVersionData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize) | |
| Description | Make query version information command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note |  |

CP5200\_ParseReadVersionRet

|  |  |
| --- | --- |
| int CP5200\_ParseReadVersionRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pInfoBuffer, int nInfoBufSize) | |
| Description | Parse return data of query version information |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| pInfoBuffer: Version information buffer, version information include 3 bytes, in each byte the high 4 bits is major version and the low 4 bit is miner version (0x10 = V1.0)  Byte 0: Bios version  Byte 1: Logic version  Byte 2: Software version |
| nInfoBufSize: version information buffer size, at least 3 bytes |
| Return | 1: Success  0: Fail  -2: Return wrong data  -3: The length of return data is not enough  -4: The size of version information buffer is too small |
| Note |  |

CP5200\_ParseReadVersionRet2

|  |  |
| --- | --- |
| int CP5200\_ParseReadVersionRet2(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pInfoBuffer, int nInfoBufSize) | |
| Description | Parse return data of query version information, include card type number, each version info represented by 2 bytes |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
|  | nLength: Length of the return data |
| pInfoBuffer: Version information buffer, return value were defined:  Byte 0: effective data len, include this byte  Byte 1: control card type  Byte 2~3: Bios version  Byte 4~5: Logic version  Byte 6~7: APP(program)version  BIOS、Logic、APP version info represented by 2 bytes |
| nInfoBufSize: version information buffer size |
| Return | 1: Success  0: Fail  -2: Return wrong data  -3: The length of return data is not enough  -4: The size of version information buffer is too small |
| Note |  |

CP5200\_MakeFormatData

|  |  |
| --- | --- |
| int CP5200\_MakeFormatData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize) | |
| Description | Make format controller file system command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note | Format will delete all files in the controller!  For formatting disk need some time, so after sending the format command to controller, need to wait about 1 second before respond data can be received. |

CP5200\_ParseFormatRet

|  |  |
| --- | --- |
| int CP5200\_ParseFormatRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of format controller file system |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1: Success  0: Fail  -2: Return wrong data  -3: The length of return data is not enough |
| Note |  |

CP5200\_MakeRestartAppData

|  |  |
| --- | --- |
| int CP5200\_MakeRestartAppData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize) | |
| Description | Make restart App command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note |  |

CP5200\_ParseRestartAppRet

|  |  |
| --- | --- |
| int CP5200\_ParseRestartAppRet (HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of restart App |
| Parameter | hObj: Handle of communication data |
|  | pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1: Success  0: Fail  -2: Return wrong data  -3: The length of return data is not enough |
| Note |  |

CP5200\_MakeRestartSysData

|  |  |
| --- | --- |
| int CP5200\_MakeRestartSysData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize) | |
| Description | Make restart controller command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note |  |

CP5200\_ParseRestartSysRet

|  |  |
| --- | --- |
| int CP5200\_ParseRestartSysRet (HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of restart controller |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1: Success  0: Fail  -2: Return wrong data |
|  | -3: The length of return data is not enough |
| Note |  |

CP5200\_MakeGetFreeSpaceData

|  |  |
| --- | --- |
| int CP5200\_MakeGetFreeSpaceData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize) | |
| Description | Make query free space in controller command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note |  |

CP5200\_ParseGetFreeSpaceRet

|  |  |
| --- | --- |
| int CP5200\_ParseGetFreeSpaceRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of query free space in controller |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | >=0: Size of free space（Byte）  -2: Return wrong data  -3: The length of return data is not enough |
| Note |  |

CP5200\_MakeGetFileInfoData

|  |  |
| --- | --- |
| int CP5200\_MakeGetFileInfoData(HOBJECT hObj, BYTE\* pBuffer, int nLength) | |
| Description | Make query file information command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| Return | >0: Length of the output data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note | It gets all files’ information in the controller, so the buffer for return data should be large enough for all files. It needs:  (file quantity) \* 32 + 10 |

CP5200\_ParseGetFileInfoRet

|  |  |
| --- | --- |
| int CP5200\_ParseGetFileInfoRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, int pos,  BYTE\* pInfoBuffer, int nInfoBufSize) | |
| Description | Parse return data of query file information and get next file information |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| pos: Current file sequence number |
|  | pInfoBuffer: File information buffer  Byte 0~44: File name and extension, including partition point ‘.’ between them ,for example:a1.txt  45~45:The two high of year. For example: in 2009, the byte value is  20; in 1999, the byte value is 19  46~46: The two low of year . For example: in 2009, the byte value is 9; in 1999, the byte value is 99  47~47:Month  48~48: Day  49~49: Hour  50~50: Minute  51~51: Second  52~55: File size, lower byte in the front |
| nInfoBufSize: File information buffer size, at least 64 bytes |
| Return | >0:File information number in the “pBuffer”  0: No required information, no next file  -1: Invalid data object handle  -2: Return wrong data  -3: The length of return data is not enough  -4: The size of information buffer is too small |
| Note | File sequence number base on 0, if pos=0 this function get file information of the second file.  Use CP5200\_ParseGetFirstFileInfoRet and this function to get file information one by one. |

CP5200\_ParseGetFirstFileInfoRet

|  |  |
| --- | --- |
| int CP5200\_ParseGetFirstFileInfoRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength,  BYTE\* pInfoBuffer, int nInfoBufSize) | |
| Description | Parse return data of query file information and get first file information |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| pInfoBuffer: File information buffer  Byte 0~11: File name  12~15 : File extend name  16~18:File Data( Year,Month,Day) , A value for each byte,the value  range of year is 0~99, the year value plus 2000 is the real year value  19~21: File time( Hour,Minute,Second) , A value for each byte,  22~25: File size, lower byte in the front  26~31: Reserve |
| nInfoBufSize: File information buffer size, at least 32 bytes |
| Return | 1: Success  0: No required information, no any file in the controller  -2: Return wrong data  -3: The length of return data is not enough  -4: The size of information buffer is too small |
| Note | Use this function and CP5200\_ParseGetNextFileInfoRet to get file information one by one |

CP5200\_ParseGetNextFileInfoRet

|  |  |
| --- | --- |
| int CP5200\_ParseGetNextFileInfoRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, int pos, BYTE\* pInfoBuffer, int nInfoBufSize) | |
| Description | Parse return data of query file information and get next file information |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| pos: Current file sequence number |
|  | pInfoBuffer: File information buffer  Byte 0~11: File name  12~15 : File extend name  16~18:File Data( Year,Month,Day) , A value for each byte,the value  range of year is 0~99, the year value plus 2000 is the real year value  19~21: File time( Hour,Minute,Second) , A value for each byte,  22~25: File size, lower byte in the front  26~31: Reserve |
| nInfoBufSize: File information buffer size, at least 32 bytes |
| Return | 1: Success  0: No required information, no next file  -2: Return wrong data  -3: The length of return data is not enough  -4: The size of information buffer is too small |
| Note | File sequence number base on 0, if pos=0 this function get file information of the second file.  Use CP5200\_ParseGetFirstFileInfoRet and this function to get file information one by one.  If the return value is less than 4,it show get all information of the controller already.  If the return value is equal to 4,you can get more information by called the function of “CP5200\_MakeGetFileInfoData()” again |

CP5200\_MakeBeginFileUploadData

|  |  |
| --- | --- |
| int CP5200\_MakeBeginFileUploadData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize, const char\* pFilename, long lFilesize, const BYTE\* pTimeBuffer) | |
| Description | Make start upload file command data,to inform the controller prepare to |
|  | receive data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| pFilename:The file name to be create,must bu short file name |
| lFilesize: The file size(byte) to be create |
| pTimeBuffer: file time message，6 bytes length byte0: year(00~99), real year-2000 byte 1: month(1~12) byte 2: day(1~31) byte 3: hour (0~23) byte 4: minute(0~56) byte 5: second(0~59) |
| Return | >0: Length of create data  -1: Invalid data object handle  -4: The size of buffer is too small |
| Note | If the controller have a same name file,the old file will be cover with.  The file’s max size is 1.5M byte  Controller can operate only one file each time |

CP5200\_ParseBeginFileUploadRet

|  |  |
| --- | --- |
| int CP5200\_ParseBeginFileUploadRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of start upload file command |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1: Success  0: Fail  -2: Return wrong data |
|  | -3: The length of return data is not enough |
| Note |  |

CP5200\_MakeFileUploadData

|  |  |
| --- | --- |
| int CP5200\_MakeFileUploadData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize, const BYTE  \*pData, WORD wDatLen, WORD wSegNo, WORD wSegLen, int nWantRet) | |
| Description | Make upload file command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| pData: The uploaded file data buffer |
| wDataLen: length of the actual upload data. Can not be greater than wSegLen. |
| wSegNo: Data segment number, from 0 to start |
| wSegLen: Data segment length of not more than 1024, it can be set to 512  generally. For each upload, the parameters need to be consistent. |
| nWantRet: Whether or not to return immediately to confirm the information。   1. .No return 2. .Ruturn |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note | If the file is too big ,must upload data in multiple  The time between two intervals upload to be not less than 50 milliseconds. Must be based on the value of “nWantRet” to determine whether or not to deal with the return of information. |

CP5200\_ParseFileUploadRet

|  |  |
| --- | --- |
| int CP5200\_ParseFileUploadRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of upload file command |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1: Success  0: Fail  -2: Return wrong data  -3: The length of return data is not enough |
| Note | Only need to use this function while passed to the parameter “nWantRet” of the function “CP5200\_MakeWriteFileData” is non-0 |

CP5200\_MakeEndFileUploadData

|  |  |
| --- | --- |
| int CP5200\_MakeEndFileUploadData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize, WORD wTotalSeg) | |
| Description | Make finish upload file command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| wTotalSeg: The total number of data segment |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseEndFileUploadRet

int CP5200\_ParseEndFileUploadRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\*

|  |  |
| --- | --- |
| pInfoBuffer, int nInfoBufSize) | |
| Description | Parse return data of finish upload file command |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| pInfoBuffer: Upload result information buffer, to record data segment unsuccessful. Every 2 bytes of data on behalf of one data segment number.  Low byte first. |
| nInfoBufSize: Upload result information buffer size, at least 72 bytes |
| Return | 0: Success  >0 And <=36: Wrong data segment amount  255: No file to be closed  -2: Return wrong data  -3: The length of return data is not enough |
| Note | Return the number of errors in the data above does not mean that all of the wrong data segment , re-issued the above known data errors, and then the result of new information, know that there is no error so far. |

CP5200\_MakeGetTypeInfoData

|  |  |
| --- | --- |
| int CP5200\_MakeGetTypeInfoData(HOBJECT hObj, BYTE\* pBuffer, int nBufSize) | |
| Description | Make query type information command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseGetTypeInfoRet

|  |  |
| --- | --- |
| int CP5200\_ParseGetTypeInfoRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pInfoBuffer, int nInfoBufSize) | |
| Description | Parse return data of query type information |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| pInfoBuffer: type of information buffer (10 bytes) byte 0: Control Card Type byte 1: FPGA version bytes 2-5: BIOS version bytes 6-9: APP version |
| nInfoBufSize: type result information buffer size, at least 10 bytes |
| Return | 0: Success  -1: Incorrect data object handle  -2: return data error  -3: the returned data length less than  -5: checksum error |
| Note |  |

CP5200\_MakeGetTempHumiData

|  |  |
| --- | --- |
| int CP5200\_MakeGetTempHumiData(HOBJECT hObj, BYTE\* pBuffer, int nBufSize ,byte byFlag) | |
| Description | Make query temperature and humidifier information command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
|  | byFlag:Query flag  Bit0: Is query temperature (0 No,1Yes)  Bit1: Is query humidifier (0 No,1Yes) |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseGetTempHumiRet

|  |  |
| --- | --- |
| int CP5200\_ParseGetTempHumiRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pInfoBuffer, int nInfoBufSize) | |
| Description | Parse return data of query temperature and humidity information |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |

|  |  |
| --- | --- |
|  | pInfoBuffer: temperature and humidity information buffer, length is 8 bytes , the meanings :  byte 0：Query flag. The same as send package byte 1~2：temperature (degress Celsius)：  Byte 1：Bit7: numeric symbols。1 negative，0 positive。 Bit6~0: the high 7 bit of the integer part of temperature absolute  Byte 2：Bit7~4: the lower 4 bit of the integer part of  temperature absolute  Bit3~0: fractional part ，unit is 1/16(0.0625)  byte 3~4：temperature (degress Fahrenheit)： byte 5：temperature adjustment value，  Bit7: 1 degress Fahrenheit，0 degress Celsius  Bit6: 1 negative，0 positive  Bit5~0: The absolute value of the temperature adjustment  byte 6：humidity。Valid values 0～100 byte 7：humidity adjustment value  Bit7: reserved  Bit6: 1 negative，0 positive  Bit5~0: The absolute value of the humidity adjustment |
| nInfoBufSize: temperature result information buffer size |
| Return | -1 |
| Note |  |

CP5200\_MakeReadConfigData

|  |  |
| --- | --- |
| int CP5200\_MakeReadConfigData(HOBJECT hObj, BYTE\* pBuffer, int nBufSize, int nFlag) | |
| Description | Make read configuration information command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| nFlag: Read the configuration information of the tag |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseReadConfigRet

|  |  |
| --- | --- |
| int CP5200\_ParseReadConfigRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pInfoBuffer, int nInfoBufSize) | |
| Description | Parse return data of read configuration information |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| pInfoBuffer: configuration of information buffer |
| nInfoBufSize: configuration result information buffer size ,at least 10 bytes |
| Return | 6: Success  -1: Incorrect data object handle  -2: return data error  -3: the returned data length less than  -5: checksum error |
| Note |  |

CP5200\_MakeWriteConfigData

|  |  |
| --- | --- |
| int CP5200\_MakeWriteConfigData(HOBJECT hObj, BYTE\* pBuffer, int nBufSize, const BYTE\* pConfig, int nCfgLength) | |
| Description | Make write configuration information command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| pConfig: Configuration information content pointers |
| nCfgLength: Configuration information length |
| Return | >0: Length of the output data |
|  | -1: Invalid data object handle |
| Note |  |

CP5200\_ParseWriteConfigRet

|  |  |
| --- | --- |
| int CP5200\_ParseWriteConfigRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of write configuration information |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Incorrect data object handle  -2: return data error  -3: the returned data length less than  -5: checksum error |
| Note |  |

CP5200\_MakeReadRunningInfoData

|  |  |
| --- | --- |
| int CP5200\_MakeReadRunningInfoData(HOBJECT hObj, BYTE\* pBuffer, int nBufSize, int nFlag) | |
| Description | Make read running info data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| nFlag: to read the run mark  1: The current playing program number  2: Read the current font information  Other: Reserved |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseReadRunningInfoRet

|  |  |
| --- | --- |
| int CP5200\_ParseReadRunningInfoRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pInfoBuffer, int nInfoBufSize) | |
| Description | Parse the return data of read running info |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| pInfoBuffer: The running info buffer byte 0：Confirmation message，0 failed；>0 success  If the flag is 1：The current playing program number，Byte1~Byte5 as follows：  Byte 1：Paly type  Byte 2~3：Program total. High byte first  Byte 4~5：Program number. High byte first  If the flag is 2：Read the current font information，Byte1~Byte6 as follows：  Byte 1：Font type  Byte 2：Reserved  Byte 3~4: ASCII character available size. High byte first Byte 5~6: Extended font available size. High byte first |
| nInfoBufSize: Running info buffer size ,at least 7 bytes |
| Return | >=6. success  -1: Incorrect data object handle  -2: return data error  -3: the returned data length less than  -5: checksum error |
| Note |  |

CP5200\_MakeScreenTestData

|  |  |
| --- | --- |
| int CP5200\_MakeScreenTestData(HOBJECT hObj, BYTE\* pBuffer, int nBufSize, BYTE\* pInfoBuffer, int nInfLength) | |
| Description | Make show the test pattern data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| pInfoBuffer: Test pattern info buffer，Details are as follows： byte 0：Option：Bit7: 0 to cancel the test, 1 immediate access to the test.  Bit6: 1 automatic return, 0 don’t automatically return  Bit0~5: automatically returns the number of previous tests.  When cancel the test, the latter parameter is invalid.  byte 1~2：Screen width. High byte first  0：defaule，>0 Screen width  byte 3~4：Screen height. High byte first  0：defaule，>0 Screen height  byte 5：Pattern color：Bit0~2：base color  Bit3：Whether the combination of the basic color  Bit4~7：Resvered, fill 0.  byte 6: Pattern gray: Bit0~3：Gray  Bit4~7：Resvered, fill 0.  byte 7: Test pattern：0: the entire screen  1: Single slash left  2: oblique grid line to the left other：Resvered, fill 0..  byte 8~9：Switching time, High byte first. Units of 10 milliseconds.  0 is the default time (3 seconds the entire screen, move 20 ms) |
| nInfLength: The test pattern data length, and now is 10 bytes. |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseScreenTestRet

|  |  |
| --- | --- |
| int CP5200\_ParseScreenTestRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse the return data of show test pattern command |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1. success  0: failed  -1: Incorrect data object handle  -2: return data error  -3: the returned data length less than  -5: checksum error |
| Note |  |

CP5200\_MakeInstantMessageData

|  |  |
| --- | --- |
| int CP5200\_MakeInstantMessageData( BYTE\* pBuffer, int nBufSize, byte byPlayTimes , int x , int y , int cx , int cy , byte byFontSizeColor , int nEffect , byte nSpeed , byte byStayTime ,const char\* pText ); | |
| Description | Make instant message data |
| Parameter | pBuffer:Output data buffer |
| nBufSize: The size of the output data buffer (BYTE) |
| byPlayTimes: Play times, from 0 to 255. 0 means continue play until new commands arrive. |
| x: Display start point x,the upper left corner of the abscissa. |
| y: Display start point y, the upper left corner of the ordinate. |
| Cx: Display width. 0 means set to maximum width. |
| Cy: Display height. 0 means set to maximum height. |
|  | byFontSizeColor: Font size and color.  Bit0~3: Font size.  Bit4: The weight of the red color  Bit5: The weight of the green color  Bit6: The weight of the blue color  Bit 7: Reserved |
| nEffect: Display effect. |
| nSpeed: Display speed,0~255.The smaller the faster. Invalid when set to display immediately. |
|  | byStayTime: Stay time. High byte previous(big endian). |
| pText: The text data. |
| Return | >0: The length of the make data  <=0: The result buffer is not big enough |
| Note | This function only packs the message but not send. You should use it with the functions: CP5200\_MakeSendInstantMessageData and  CP5200\_ParseSendInstantMessageRet |

CP5200\_MakeInstantMessageData1

|  |  |
| --- | --- |
| int MakeInstantMessageData1( BYTE\* pBuffer, int nBufSize, BYTE byPlayTimes , int x , int y , int cx , int cy , int nFontSize ,byte byColorAlign , int nEffect , BYTE nSpeed , BYTE byStayTime ,const char\* pText ); | |
| Description | Make instant message data |
| Parameter | pBuffer:Output data buffer |
| nBufSize: The size of the output data buffer (BYTE) |
| byPlayTimes: Play times, from 0 to 255. 0 means continue play until new commands arrive. |
| x: Display start point x,the upper left corner of the abscissa. |
| y: Display start point y, the upper left corner of the ordinate. |
| Cx: Display width. 0 means set to maximum width. |
|  | Cy: Display height. 0 means set to maximum height. |
| nFontSize: font size and style，see 1.7. Font size code and font style |
| byColorAlign：color and alignment  Bit0: Red flag  Bit1: Green flag  Bit2: Blue flag  Bit3: Resvered  Bit4~5: Horizontal alignment. 0 Left, 1 Middle, 2 right  Bit6~7: Vertical alignment. 0 Top , 1 Middle , 2 Bottom |
| nEffect: Display effect. |
| nSpeed: Display speed,0~255.The smaller the faster. Invalid when set to display immediately. |
| byStayTime: Stay time. High byte previous(big endian). |
| pText: The text data. |
| Return | >0: The length of the make data  <=0: The result buffer is not big enough |
| Note | This function only packs the message but not send. You should use it with the functions: CP5200\_MakeSendInstantMessageData and  CP5200\_ParseSendInstantMessageRet |

CP5200\_MakeSendInstantMessageData

|  |  |
| --- | --- |
| int CP5200\_MakeSendInstantMessageData(HOBJECT hObj, BYTE\* pBuffer, int nBufSize, const BYTE\* pData, int nDataLen , byte byLastPacket , long lDataOffset); | |
| Description | Make send instant message data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| pData: The instant message data which is return of the function CP5200\_MakeInstantMessageDat. |
| nDataLen: The length of data. Low byte previous (little endian). |
|  | byLastPacket: Whether is the last packet.  1: YES  0: NO |
| lDataOffset: The data offset, Low byte previous (little endian). |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note | This function sends the packet which is made by  CP5200\_MakeInstantMessageData function. The length of each packet can not bigger than 1024 bytes, 200 is proposition. |

CP5200\_ParseSendInstantMessageRet

|  |  |
| --- | --- |
| int CP5200\_ParseSendInstantMessageRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pInfoBuffer, int nInfoBufSize); | |
| Description | Parse the return data of send instant message command |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nLength: The length of the data buffer |
| pInfoBuffer: The return data buffer |
| nInfoBufSize: 5 bytes.  Byte 1: 0x00:Failure. Not 0: Success  Other: Low byte previous (little endian). |
| Return | 0: Success  -1: Incorrect data object handle  -2: return data error  -3: the returned data length less than  -5: checksum error |
| Note |  |

CP5200\_MakeReadHWSettingData

int CP5200\_MakeReadHwSettingData(HOBJECT hObj, BYTE\* pBuffer, int nBufSize)

|  |  |
| --- | --- |
| Description | Make read scan param command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseReadHWSettingRet

|  |  |
| --- | --- |
| int CP5200\_ParseReadHWSettingRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pInfoBuffer, int nInfoBufSize , int nPassword) | |
| Description | Parse return data of read scan param |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| pInfoBuffer: Scan param buffer，at least 16 bytes，see the meaning of each byte 1.14、The meaning of each byte of the scan parameters |
| nInfoBufSize: The size of Scan param buffer，at least 16 bytes |
| nPassword: Parsing code, depending on the control card filled with different passwords, or not to accept |
| Return | 16: Success  -1: Incorrect data object handle  -2: return data error  -3: the returned data length less than  -5: checksum error |
| Note |  |

CP5200\_MakeWriteHWSettingData

|  |  |
| --- | --- |
| int CP5200\_MakeWriteHWSettingData(HOBJECT hObj, BYTE\* pBuffer, int nBufSize, const BYTE\* pSetting, int nPassword) | |
| Description | Make write scan param command data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| pSetting: Scan param buffer，16 bytes，see the meaning of each byte 1.14、  The meaning of each byte of the scan parameters |
| nPassword: Parsing code, depending on the control card filled with different passwords, or not to accept |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseWriteHWSettingRet

|  |  |
| --- | --- |
| int CP5200\_ParseWriteConfigRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of write scan param |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | >0: Success  -1: Incorrect data object handle  -2: return data error  -3: the returned data length less than  -5: checksum error |
| Note |  |

CP5200\_MakeReadSoftwareSwitchInfoData

|  |  |
| --- | --- |
| int CP5200\_MakeReadSoftwareSwitchInfoData(HOBJECT hObj, BYTE \*pBuffer, int nBufSize) | |
| Description | Make read software switch info data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| Return | >0: Length of the output data(BYTE)  -1: Invalid data object handle  -4: Buffer len not enough |
| Note |  |

CP5200\_ParseReadSoftwareSwitchInfoRet

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| int CP5200\_ParseReadSoftwareSwitchInfoRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pInfoBuffer, int nInfoBufSize) | | | | | |
| Description | Parse return data of read software switch info data | | | | |
| Parameter | hObj: Handle of communication data | | | | |
| pBuffer: The return data | | | | |
| nLength: Length of the return data | | | | |
| pInfoBuffer: Software switch info buffer, software switch info include 9  BYTEs  Data Length Description | | | | |
|  | Switch info | 1 | 0: off  1: on |  |
| Value | 8 | BYTE 1~2: Turn on hour, minute  BYTE 3~4: Turn off hour, minute  BYTE 5~8: Reserved |
| nInfoBufSize: The size of Software switch info buffer，at least 9 bytes | | | | |
| Return | 1: Success | | | | |
|  | 0: Fail  -1: Incorrect data object handle  -2: Returned data type error  -3: Returned data len not enough  -4: Buffer size not enough  -5: Checksum error | | | | |
| Note |  | | | | |

CP5200\_MakeWriteSoftwareSwitchInfoData

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| int CP5200\_MakeWriteSoftwareSwitchInfoData(HOBJECT hObj, BYTE\* pBuffer, int nBufSize, const BYTE \*pSoftwareSwitchInfoBuf) | | | | | |
| Description | Make write software switch info data | | | | |
| Parameter | hObj: Handle of communication data | | | | |
| pBuffer: Output data buffer | | | | |
| nBufSize: Size of output data buffer (BYTE) | | | | |
| pSoftwareSwitchInfoBuf: Software switch info | | | | |
|  | Data | Length | Description |  |
| Switch info | 1 | 0: Turn off immediately  1: Turn on immediately |
| Value | 8 | Default: 0 |
| Return | >0: Length of the output data(BYTE)  -1: Invalid data object handle  -4: Buffer len not enough | | | | |
| Note |  | | | | |

CP5200\_ParseWriteSoftwareSwitchInfoRet

int CP5200\_ParseWriteSoftwareSwitchInfoRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pInfoBuffer, int nInfoBufSize)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description | Parse return data of write software switch info data | | | | |
| Parameter | hObj: Handle of communication data | | | | |
| pBuffer: The return data | | | | |
| nLength: Length of the return data | | | | |
| pInfoBuffer: Software switch info buffer, Software switch info include at least 1 byte | | | | |
|  | Data | Length | Description |  |
| Status | >=1 | First byte：0 turn off screen，1 turn on screen  Others: ignored |
| nInfoBufSize: The size of Software switch info buffer，at least 1 bytes | | | | |
| Return | 1: Success  0: Fail  -1: Incorrect data object handle  -2: Returned data type error  -3: Returned data len not enough  -4: Buffer size not enough  -5: Checksum error | | | | |
| Note |  | | | | |

CP5200\_MakeQueryControllerInfo

|  |  |
| --- | --- |
| int CP5200\_MakeQueryControllerInfo( HOBJECT hObj, BYTE\* pBuffer, int nBufSize, byte byInfoFlag, const BYTE \*pAppendBuf, int nAppendLen ) | |
| Description | Make query controller information data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
|  | byInfoFlag：Query flag , currently only support 0x0b  0x00：（invalid）  0x01：The current playing program number  0x02：Read current font information  0x0a: Query the check result in the program playing  0x0b: Query playing status and data others：retention |
| pAppendBuf：Append data  When query flag is 0x01 or 0x02, no append data.  When query flag is 0x0a, the append data only one byte, is 0x31. When query flag is 0x0b, the append data length >=1 bytes, to read the information and parameters  The first byte =0 returns the screenshot data identification, program number, program has been playing time.  The first byte =1 returns the screenshot data identification, program number, program has been playing time, in addition to return in the broadcast on the screen picture data. The second byte multiplied by 8 is the number of bytes per packet data of a desired picture contains the 0 representation is determined by the control card.  The first byte=2 returns the actual image data. Second and three bytes for the image datapacket sequence number, the high byte in the front. |
| nAppendLen：Append data length |
| Return | >0: Length of the output data(BYTE)  -1: Invalid data object handle  -4: Buffer len not enough |
| Note |  |

CP5200\_ParseQueryControllerInfoRet

|  |  |
| --- | --- |
| int CP5200\_ParseQueryControllerInfoRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, byte byInfoFlag, byte byAppendFlag, byte \*pInfoBuf, int nInfoBufLen ) | |
| Description | Parse return data of query controller information |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
|  | byInfoFlag：Query flag , currently only support 0x0b  0x00：（invalid）  0x01：The current playing program number  0x02：Read current font information  0x0a: Query the check result in the program playing  0x0b: Query playing status and data others：retention |
| byAppendFlag： |
| pInfoBuf：See below result cache description |
| nInfoBufLen：结果缓存长度 |
| Return | 1: Valid data length  0: The AppendFlag is invalid  -1: Incorrect data object handle  -2: Returned data type error  -3: Returned data len not enough  -4: Buffer size not enough  -5: Checksum error  -6: The return data error  -7: The “byInfoFlag” invalid. |
| Note |  |

Result cache description

|  |  |
| --- | --- |
| Query flag | Return data description |
| 0x01：Query current play program number | 5 bytes。  Byte 0: Play type.  Bit0~3: 0 general program,  Bit4: 0 the first set of programs, 1 second sets of programs Bit5~7: reserves  Byte 1~2: The total number of bytes of program. High byte in the front  Byte 3~4: Program number. High byte in the front |
| 0x02：Query current font | 6 bytes。  Byte 0: font type  Byte 1: reserves  Byte 2~3: ASCII font available size. High byte in the front  Byte 4~5: extended fonts available size. High byte in the front |
| 0x0a ： Query the check result in the program playing， the append data | Variable length, no more than 150 bytes.  Byte 0: Append value (with the transmitted value)  Byte 1~2: The total program numbers in play list. High byte in the front |
| value must to be set to  0x31 | Byte 3~4: checked program number. When it is broadcast program, this value may be in error. High byte in the front  Byte 5~6: Program error message data length. High byte in the front  Byte 7~: Program error message data. According to the program sequence, each of the 8 program information is represented by 1 bytes (1 bits each program), 0 id not found error (alsomay not have to check), 1 identity is wrong. |
| 0x0b：Query playing status and data    The return value is determined according to the append value | Append value is 0 and 1：  The append value of 0 to return to 0~17 bytes of information; the append value of 1, also returns 18 bytes and the later data:  Byte 0~5: current screenshot data identifies  Byte 6~7: program number. High byte in the front, the first program from the beginning of 1, 0 indicates no programs in play or play the temporary information  Byte 8~9: playing item no..  Byte 10~13: program has broadcast time, the unit is 1/10 seconds, high byte in the front  Byte 14~17: play item has the playing time, the unit is 1/10 seconds, high byte in the front  Following  Byte 18~19: image width, high byte in the front  Byte 20~21: image height, high byte in the front  Byte 22: color and gray  Byte 23: multiplied by 8 for an image data packet length  Byte 24~27: image data length, high byte in the front Append value is 2:  Byte 0~5: current screenshot data identifies  Byte 6~7: the total number of image data packet, the high byte in the front  Byte 8~9: image data packet sequence number, the high byte in the front  Byte 10~13: image data offset, high byte in the front  Byte 14~15: image data length, high byte in the front  Byte 16~: image data |

CP5200\_MakeOpenFileData

|  |  |
| --- | --- |
| int CP5200\_MakeOpenFileData(HOBJECT hObj, BYTE\* pBuffer, int nBufSize, const char\* chFileName) | |
| Description | Make open file data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
|  | nBufSize: Size of output data buffer (BYTE) |
| chFileName: The file name will to be open.If the file in the system disk, name needs coupled with the “S:” |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseOpenFileRet

|  |  |
| --- | --- |
| int CP5200\_ParseOpenFileRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pInfoBuffer, int nInfoBufSize) | |
| Description | Parse the return data of open file |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| pInfoBuffer: file info buffer byte 0~1：file number，Low byte first. |
| nInfoBufSize: the file info buffer size ，require more than 2 bytes. |
| Return | >=0: Success , the file number will to be open  -1: Incorrect data object handle  -2: return data error  -3: the returned data length less than  -4: the info buffer length less than  -5: checksum error |
| Note |  |

CP5200\_MakeGetDirentryData

int CP5200\_MakeGetDirentryData(HOBJECT hObj, BYTE\* pBuffer, int nBufSize, WORD dno, int nPath)

|  |  |
| --- | --- |
| Description | Make get file info data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| dno: file number，Obtained by CP5200\_ParseOpenFileRet |
| nPath：Path infp，user disk is 1，system disk is 0. |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseGetDirentryRet

|  |  |
| --- | --- |
| int CP5200\_ParseGetDirentryRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pInfoBuffer, int nInfoBufSize) | |
| Description | Parse the return data of get file info |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| pInfoBuffer: file info buffer  Byte 0 to 31: File Name  Bytes 32 to 43: the file name extension  Bytes 44 to 45: file attributes  Bytes 46 to 47: File checksum  Bytes 48 to 49: Reserved  Bytes 50 to 53: file generation time  Bytes 54 to 57: Date of file generation  Bytes 58 to 59: meaning unknown  Bytes 60 to 63: File Size  All data are low byte first. |
| nInfoBufSize: the file info buffer size ，require more than 64 bytes. |
| Return | >=0: the file number  -1: Incorrect data object handle  -2: return data error  -3: the returned data length less than  -4: the info buffer length less than  -5: checksum error |
| Note |  |

CP5200\_MakeReadFileNoData

|  |  |
| --- | --- |
| int CP5200\_MakeReadFileNoData(HOBJECT hObj, BYTE\* pBuffer, int nBufSize,WORD wdCount, byte fno) | |
| Description | Make read file data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| wdCount: the data length will to be read |
| fno：file number，Obtained by CP5200\_ParseOpenFileRet |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseReadFileNoRet

|  |  |
| --- | --- |
| int CP5200\_ParseReadFileNoRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pInfoBuffer, int nInfoBufSize) | |
| Description | Parse the return data read file data |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
|  | pInfoBuffer: The file data buffer |
| nInfoBufSize: The file data buffer size, require more than 512 bytes. |
| Return | >=0: The length of the data has been read  -1: Incorrect data object handle  -2: return data error  -3: the returned data length less than  -4: the info buffer length less than  -5: checksum error |
| Note |  |

CP5200\_MakeCloseFileNoData

|  |  |
| --- | --- |
| int CP5200\_MakeCloseFileNoData(HOBJECT hObj, BYTE\* pBuffer, int nBufSize, byte fno) | |
| Description | Make close file data |
| Parameter | hObj: Handle of communication data |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| fno：file number，Obtained by CP5200\_ParseOpenFileRet |
| Return | >0: Length of the output data  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseCloseFileNoRet

|  |  |
| --- | --- |
| int CP5200\_ParseCloseFileNoRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\*) | |
| Description | Parse the return data of close file |
| Parameter | hObj: Handle of communication data |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 1: Success |
|  | -1: Incorrect data object handle  -2: return data error  -3: the returned data length less than  -4: the info buffer length less than  -5: checksum error |
| Note |  |

# API function for multi-window protocal data communication

5.1、Overview of data communication API function

|  |  |  |
| --- | --- | --- |
| No. | Function name | Description |
| 1 | CP5200\_CmmPacker\_Create | Create multi-window communication data object |
| 2 | CP5200\_CmmPacker\_Destroy | Destroy multi-window communication data object |
| 3 | CP5200\_CmmPacket\_SetParam | Set communication data packet parameter |
| 4 | CP5200\_CmmPacker\_Count | Get the number of packets in the object |
| 5 | CP5200\_CmmPacker\_Data | Get the data of packet in the object |
| 6 | CP5200\_MakeSplitScreenData | Make split window command data |
| 7 | CP5200\_ParseSplitScreenRet | Parse return data of split window command |
| 8 | CP5200\_MakeSendTextData  CP5200\_MakeSendTextData1 | Make send text command data |
| 9 | CP5200\_ParseSendTextRet | Parse return data of send text command |
| 10 | CP5200\_MakeSendTagTextData  CP5200\_MakeSendTagTextData1 | Make send tag text command data. Font, color , etc… can be controlled by tag text |
| 11 | CP5200\_ParseSendTagTextRet | Parse return data of send tag text command |
| 12 | CP5200\_MakeSendPictureData | Make send picture command data |
| 13 | CP5200\_ParseSendPictureRet | Parse return data of send picture command |
| 14 | CP5200\_MakeSendStaticData | Make send static text command data |
| 15 | CP5200\_ParseSendStaticRet | Parse return data of send static text command |
| 16 | CP5200\_MakeSendClockData | Make send clock command data |
| 17 | CP5200\_ParseSendClockRet | Parse return data of send clock command |
| 18 | CP5200\_MakeExitSplitScreenData | Make exit split window command data |
| 19 | CP5200\_ParseExitSplitScreenRet | Parse return data of exit split window |
|  |  | command |
| 20 | CP5200\_MakeSaveClearWndData | Make save or clear window data command data |
| 21 | CP5200\_ParseSaveClearWndRet | Parse return data of save or clear window data command |
| 22 | CP5200\_MakePlaySelectedPrgData  CP5200\_MakePlaySelectedPrgData1 | Make select play program command data |
| 23 | CP5200\_ParsePlaySelectedPrgRet | Parse return data of select play program command |
| 24 | CP5200\_MakeSetUserVarData | Make set user variable command data |
| 25 | CP5200\_ParseSetUserVarRet | Parse return data of set user variable command |
| 26 | CP5200\_MakeSelectedAndUserVarData | Make selected and user variable data |
| 27 | CP5200\_\_ParseSelectedAndUserVarRet | Parse the return data of selected and user variable command |
| 28 | CP5200\_MakeSetGlobalZoneData | Make set global zone data |
| 29 | CP5200\_ParseSetGlobalZoneRet | Parse the return data of set global zone command |
| 30 | CP5200\_MakePushUserVarData | Make push user variable data |
| 31 | CP5200\_ParsePushUserVarRet | Parse the return data of push user variable data |
| 32 | CP5200\_MakeTimerCtrlData | Make the timer ctronl data |
| 33 | CP5200\_ParseTimerCtrlRet | Parse the return data of timer contrl command |
| 34 | CP5200\_MakeSetZoneAndVariableData | Make set global zone and user variable value data |
| 35 | CP5200\_ParseSetZoneAndVariableRet | Parse the return data of set global zone and user variable value |
| 36 | CP5200\_MakeSendPureTextData | Make send pure text data |
| 37 | CP5200\_ParseSendPureTextRet | Parse the return data of send pure text |

Usage：

Step 1: Create multi-window communication data object

Step 2: Make communication data, include RS232/485’s code convert

(0xa5 => 0xaa 0x05, …), or network ID code

Step 3: Get the number of packets in the object

Step 4: One by one to handle each packet of data by the following manner：

1. Send the packet data to the controller
2. Receive data from controller, and process code convert

(0xaa 0x05 => 0xa5, …)

1. Parse the return data and get the result

Step 5: Destroy multi-window communication data object

5.2 、 Detail of multi-window protocal data

communication API functions

CP5200\_CmmPacker\_Create

|  |  |
| --- | --- |
| HOBJECT CP5200\_ CmmPacker\_Create(int nCommType, BYTE byCardID, DWORD dwIDCode) | |
| Description | Create multi-window communication data object |
| Parameter | nCommType: RS232/485 or network communication type  0: RS232/485  1: Network |
| byCardID: Controller ID |
| dwIDCode: Network ID code of the controller. RS232 ignore it. |
| Return | Handle of multi-window communication data object, all these kind of API functions use this handle  Return NULL if fail |
| Note | When an application no longer requires a given object, it should be destroyed to free the resource. |

CP5200\_CmmPacker\_Destroy

|  |  |
| --- | --- |
| int CP5200\_CmmPacker\_Destroy(HOBJECT hObj) | |
| Description | Destroy multi-window communication data object |
| Parameter | hObj: Handle of multi-window communication data object to de destroyed |
| Return | 0: No error  -1: Invalid data object handle |
| Note |  |

CP5200\_CmmPacket\_SetParam

HOBJECT CP5200\_CmmPacket\_SetParam (HOBJECT hObj, int nParamType, const char

|  |  |
| --- | --- |
| \*pParamString) | |
| Description | Set data packet communication parameter |
| Parameter | hObj: Handle of communication data object |
| nParamType：Parameter type，valid value 1. |
| pParamString：Parameter string  When parameter type is 1，pParamString is controller’s device ID |
| Return | 1: No error  0: Parameter type is wrong  -1: Invalid data object handle  -2: pParamString is wrong |
| Note |  |

CP5200\_CmmPacker\_Count

|  |  |
| --- | --- |
| int CP5200\_CmmPacker\_Count(HOBJECT hObj) | |
| Description | Get the number of packets in the object |
| Parameter | hObj: Handle of communication data object |
| Return | >=0: the number of packets  -1: Invalid data object handle |
| Note |  |

CP5200\_CmmPacker\_Data

|  |  |
| --- | --- |
| int CP5200\_CmmPacker\_Data(HOBJECT hObj , BYTE \*pBuffer, int nBufSize, int nPackIndex ) | |
| Description | Get the data of packet in the object |
| Parameter | hObj: Handle of communication data object |
| pBuffer: Output data buffer |
| nBufSize: Size of output data buffer (BYTE) |
| nPackIndex: Pack index,starting from 0. |
| Return | >=0: the length of packet data |
|  | -1: Invalid data object handle |
| Note |  |

CP5200\_MakeSplitScreenData

|  |  |
| --- | --- |
| int CP5200\_MakeSplitScreenData(HOBJECT hObj, int nScrWidth, int nScrHeight, int nWndCnt, const int \*pWndRects); | |
| Description | Make split window command data |
| Parameter | hObj: Handle of communication data object |
| nScrWidth: the width of screen |
| nScrHeight: the height of screen |
| nWndCnt: the split window number ，RMS 1~8。 |
| pWndRects: Window coordinates, each window with four integer said the  "left, up,right,down” coordinates,ave the same data structure with the  "RECT"of windows。 |
| Return | >=0: the number of packets  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseSplitScreenRet

|  |  |
| --- | --- |
| int CP5200\_ParseSplitScreenRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of split window command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeSendTextData

(CP5200\_MakeSendTextData1)

|  |  |
| --- | --- |
| int CP5200\_MakeSendTextData(HOBJECT hObj, int nWndNo, const char \*pText, COLORREF crColor, int nFontSize, int nSpeed, int nEffect, int nStayTime, int nAlignment); | |
| Description | Make send text command data |
| Parameter | hObj: Handle of communication data object |
| nWndNo: Window sequence number, valid values 0 to 7 |
| pText: Text to be send |
| crColor: Text color |
| nFontSize: font size and style，see 1.7. Font size code and font style，this parameter only support the font size, does not support multiple font |
| nSpeed: Effect speed  0～100：The fastest value of 0。 |
| nEffect: Show effect。See the "1.5" section. |
| nStayTime: Stay time in second |
| nAlignment: The level of alignment  0: left Alignment  1: center Alignment  2: right Alignment |
| Return | >=0: the number of packets  -1: Invalid data object handle |
| Note | CP5200\_MakeSendTextData1 is for single byte characters, ASCII and extended ASCII. |

CP5200\_ParseSendTextRet

int CP5200\_ParseSendTextRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength)

|  |  |
| --- | --- |
| Description | Parse return data of send text command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeSendTagTextData (CP5200\_MakeSendTagTextData1)

|  |  |
| --- | --- |
| int CP5200\_MakeSendTagTextData(HOBJECT hObj, int nWndNo, const char \*pText, COLORREF crColor, int nFontSize, int nSpeed, int nEffect, int nStayTime, int nAlignment) | |
| Description | Make send tag text data |
| Parameter | hObj: Handle of communication data object |
| nWndNo: indow sequence number, valid values 0 to 7 |
| pText: Text to be send |
| crColor:Text color |
| nFontSize: font size and style，see 1.7. Font size code and font style，this parameter only support the font size, does not support multiple font |
| nSpeed:Show speed |
| nEffect:Render effect。See the "1.5" section. |
| nStayTime: Stay time in second. |
| nAlignment: The level of alignment。  0: Left alignment  1: Center alignment  2: Right alignment |
| Return | >=0: The number of packets |
|  | -1: Invalid data object handle |
| Note | CP5200\_MakeSendTagTextData1 is for single byte characters, ASCII and extended ASCII. |

CP5200\_ParseSendTagTextRet

|  |  |
| --- | --- |
| int CP5200\_ParseSendTagTextRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse the return data of send tag text command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength The length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeSendPictureData

|  |  |
| --- | --- |
| int CP5200\_MakeSendPictureData(HOBJECT hObj, int nWndNo, int nPosX, int nPosY, int nCx, int nCy, const char \*pPictureFile, int nSpeed, int nEffect, int nStayTime, int nPictRef); | |
| Description | Make send picture command data |
| Parameter | hObj: Handle of communication data object |
| nWndNo: Window sequence number, valid values 0 to 7 |
| nPosX: Began to show the location of X coordinate. Relative upper-left corner the window.。 |
| nPosY: Began to show the location of Y coordinate. Relative upper-left corner the window. |
| nCx: The width of picture |
| nCy: The heigth of picture |

|  |  |
| --- | --- |
|  | pPictureFile: Path and file name of the picture file ,this is based on the value of nPictRef.  When the value of nPictRef is 0: pPictureFile is the Path and file name of the file on the computer.  When the value of nPictRef is 1: pPictureFile is the Path and file name of the GIF file on the controller card.  When the value of nPictRef is 2: pPictureFile is the Path and file name of the file on the computer.  When the value of nPictRef is 3: pPictureFile is the Path and file name of picture packages and the serial number of the picture on the controller card.  Packages name followed by is separated by a space. For example:  “images.rpk 1” |
| nSpeed: Effect speed  0～100：The fastest value of 0。 |
| nEffect: Show effect。See the "1.5" section. |
|  | nStayTime: Stay time in second |
| nPictRef: the way to send picture and meaning.  0：display the local picture that will be converted into the format of GIF to send.  1：display the gif picture that on the controller card.  2：display the local picture that will be converted into the format of simple to send.  3：display the picture in the picture packages that on the controller card.  Other values: deal with 0. |
| Return | >=0: the number of packets  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseSendPictureRet

|  |  |
| --- | --- |
| int CP5200\_ParseSendPictureRet (HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of send picture command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeSendSimpleImageData

|  |  |
| --- | --- |
| int CP5200\_MakeSendSimpleImageData(HOBJECT hObj, int nWndNo, int nPosX, int nPosY, int nSpeed, int nEffect, int nStayTime, BYTE\* pPicData , long lPicDataLen ); | |
| Description | Parse return data of send simple image command |
| Parameter | hObj: Handle of communication data object |
| nWndNo: Window sequence number, valid values 0 to 7 |
| nPosX: Began to show the location of X coordinate. Relative upper-left corner the window.。 |
| nPosY: Began to show the location of Y coordinate. Relative upper-left corner the window. |
| nSpeed: Effect speed  0～100：The fastest value of 0。 |
| nEffect: Show effect。See the "1.5" section. |
| nStayTime: Stay time in second |
| pPicData: simple picture data,see the 1.11 simple picture data fomart. |
| lPicDataLen:the length of simple picture data. |
| Return | >=0: the number of packets  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseSendSimpleImageRet

|  |  |
| --- | --- |
| int CP5200\_ParseSendSimpleImageRet (HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of send simple picture command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeSendStaticData

|  |  |
| --- | --- |
| int CP5200\_MakeSendStaticData(HOBJECT hObj, int nWndNo, const char \*pText, COLORREF crColor, int nFontSize, int nAlignment, int x, int y, int cx, int cy); | |
| Description | Make send static text command data |
| Parameter | hObj: Handle of communication data object |
| nWndNo: Window sequence number, valid values 0 to 7 |
| pText: Text to be send |
| crColor: Text color。 |
| nFontSize: font size and style，see 1.7. Font size code and font style |
|  | nAlignment: The level of alignment  0: left Alignment  1: center Alignment  2: right Alignment |
| x: Start X of the play window |
| y: Start Y of the play window |
|  | cx: The width of play window |
| cy: The height of play window. |
| Return | >=0: the number of packets  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseSendStaticRet

|  |  |
| --- | --- |
| int CP5200\_ParseSendStaticRet (HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of send static text command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeSendClockData

|  |  |
| --- | --- |
| int CP5200\_MakeSendClockData(HOBJECT hObj, int nWinNo , int nStayTime , int nCalendar , int nFormat , int nContent , int nFont , int nRed , int nGreen , int nBlue , LPCSTR pTxt); | |
| Description | Make send clock command data |

|  |  |
| --- | --- |
| Parameter | hObj: Handle of communication data object |
| nWndNo: Window sequence number, valid values 0 to 7 |
| nStayTime: Stay time in second。 |
| nCalendar: Calendar 0: Gregorian calendar date and time  1: Lunar date and time  2: Chinese lunar solar terms  3: Lunar time and date + Solar Terms |
| nFormat: Format bit 0: when the system (0: 12 hour; 1: 24 hours system) bit 1: Year digit (0: 4; 1: 2) bit 2: Branch (0: single; 1: multi-line)  bit 3~5: Format control, such as the November 12, 2010 Friday , according to diffenert values expressed as: 0: 2010/11/12 Friday 16:20:30  1: Fri，12/11/2010 16:20:30  2: 2010-11-12 Fri. 16:20:30  3: Friday，12 November 2010 16:20:30 4: Fri，Nov 12,2010 16:20:30  5: Friday，November 12 2010 16:20:30  6: Fri，11/12/2010 16:20:30 7: 2010/11/12，Fri.16:20:30 bit 6: show hands,marks bit 7: Transparent |
| nContent: Content  By bit to determine the content to display.  bit 7: Pointer bit 6: weeks bit 5: seconds bit 4: minute bit 3: hour bit 2: day bit 1: month bit 0: year |
| nFont: Font，Bit0~3: font size |
| nRed: The red color component |
| nGreen: The red green component |
| nBlue: The red blue component |
| pTxt: Text string to the end of 0x00. |
| Return | >=0: the number of packets  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseSendClockRet

|  |  |
| --- | --- |
| int CP5200\_ParseSendClockRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of send clock command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeExitSplitScreenData

|  |  |
| --- | --- |
| CP5200\_MakeExitSplitScreenData(HOBJECT hObj) | |
| Description | Make exit split window command data |
| Parameter | hObj: Handle of communication data object |
| Return | >=0: the number of packets  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseExitSplitScreenRet

|  |  |
| --- | --- |
| int CP5200\_ParseExitSplitScreenRet (HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of exit split window command |
| Parameter | hObj: Handle of communication data object |
|  | pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeSaveClearWndData

|  |  |
| --- | --- |
| CP5200\_MakeSaveClearWndData(HOBJECT hObj, int nSavaOrClear); | |
| Description | Make save or clear window data command data |
| Parameter | hObj: Handle of communication data object |
| nSavaOrClear：Save or clear data 0: Save data to the flash.  1: Clear data from the flash. |
| Return | >=0: the number of packets  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseSaveClearWndRet

|  |  |
| --- | --- |
| int CP5200\_ParseSaveClearWndRet (HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of save or clear window data command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakePlaySelectedPrgData

|  |  |
| --- | --- |
| int CP5200\_MakePlaySelectedPrgData(HOBJECT hObj, const WORD \*pSelected, int nSelCnt, int nOption) | |
| Description | Make select play program command data |
| Parameter | hObj: Handle of communication data object |
| pSelected：The program number array of be selected to play |
| nSelCnt：The program count of be selected |
| nOption：Whether to save select message to the flash  0：No save  1：Save |
| Return | >=0: the number of packets  -1: Invalid data object handle |
| Note |  |

CP5200\_MakePlaySelectedPrgData1

|  |  |
| --- | --- |
| int CP5200\_MakePlaySelectedPrgData1(HOBJECT hObj, const WORD \*pSelected, int nSelCnt, int nOption , int nScrWidth, int nScrHeight , byte byColorGray , byte nWndCnt ) | |
| Description | Make select play program command data |
| Parameter | hObj: Handle of communication data object |
| pSelected：The program number array of be selected to play |
| nSelCnt：The program count of be selected |
| nOption：Whether to save select message to the flash  0：No save  1：Save |
| nScrWidth：Screen width |
| nScrHeight：Screen height |
| byColorGray：color gray |
| nWndCnt：window count |
| Return | >=0: the number of packets |
|  | -1: Invalid data object handle |
| Note |  |

CP5200\_ParsePlaySelectedPrgRet

|  |  |
| --- | --- |
| int CP5200\_ParsePlaySelectedPrgRet (HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of select play program command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeSetUserVarData

|  |  |
| --- | --- |
| int CP5200\_MakeSetUserVarData(HOBJECT hObj, int bSave , int nVarNum , int bAstride , int\* nVarLen , byte\* byNoData ); | |
| Description | Make set user variable command data |
| Parameter | hObj: Handle of communication data object |
| bSave: Bit0:Whether to save all variables to the flash 0:No save，1:Save。  Bit1~7: Reserved,set to 0 |
| nVarNum: Variable number |
| bAstride: Whether to allow cross-variable zone setting. 0 is not permitted; 1 is permit |
| nVarLen：Bytes of data specified for each variable. |
| byNoData：Specified number of variables and variable data for each variable, the first byte of each variable is the variable number, followed by a specified length of variable data. |
| Return | >=0: the number of packets |
|  | -1: Invalid data object handle |
| Note | Corresponds to a variable number of each variable area size of each variable region is 32 bytes. Multiple continuous variables can be linked to a variable area used,occupied area of the variable number of variables can not be used。  When does not allow cross-variable area, more than 32 bytes of data are discarded；When allow cross-variable area,calculate the length of the data area to use the number of variables.  Valid values for the variable number is 1~100。Number of variables corresponding to each variable area can store 32 bytes of data, a number of continuous variable area can be used together for a variable, the variable area occupied number of variables can not be used。  When variable values are not updated and just save the variable value to the FLASH, it can set the " nVarNum " of the value of 0, set the " bSave " to save |

CP5200\_ParseSetUserVarRet

|  |  |
| --- | --- |
| int CP5200\_ParseSetUserVarRet (HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of set user variable command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeSelectedAndUserVarData

|  |  |
| --- | --- |
| int CP5200\_MakeSelectedAndUserVarData(HOBJECT hObj, int nOption , int nVarNum , int bAstride , int\* nVarLen , byte\* byNoData, int nSelPrg) | |
| 作用 | Make selected and user variable command data |
| 参数 | hObj: Handle of communication data object |
|  | nOption:  Bit0: Whether to save the program number to the FLASH  0:Not save, 1: Save  Bit1: Whether to save all the variables to the FLASH  0:Not save, 1: Save  Bit2: Whether to clear the old variables  0:Not clear, 1:Clear  Bit3~7: Reserved,set to 0 |
| nVarNum: Variable Number。Bit0~6：The variable number which to be set |
| bAstride: Whether to allow cross-variable zone setting. 0 is not permitted; 1 is permit |
| nVarLen: Variable data length.Sort every variable byte data in alphabet order.The total length of variable number and data is (1+n)byte. |
|  | byNoData: Variable No and data. The first byte is variable No, followed by a specify length data. |
| nSelPrg: Program No.The List of the selected program No.Each program  No has 2 bytes,high byte is previous.The overflow program will be ignored. |
| 返回值 | >=0: the number of packets  -1: Invalid data object handle |
| 其它说明 |  |

CP5200\_ParseSelectedAndUserVarRet

|  |  |
| --- | --- |
| int CP5200\_ParseSelectedAndUserVarRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| 作用 | Parse return data of selected and user variable command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle |
|  | -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeSetGlobalZoneData

|  |  |
| --- | --- |
| int CP5200\_MakeSetGlobalZoneData(HOBJECT hObj, byte byConfig , byte bySynchro , byte byZoneNum , byte \*byZoneMsg) | |
| 作用 | Make set global messge command data |
| 参数 | hObj: Handle of communication data object |
| byConfig:  Bit0: Whether to save to FLASH  0:Not save, 1:Save  Bit1~7:Reserved, set to 0 |
| bySynchro: Synchronization。  Bit0: Whether to synchronization, 0 Not synchronous，1 synchronous。  Bit1~7: Reserved |
| byZoneNum: Zone number.The golobal display zone number which to be set.Cancel all the zone when zone number is 0. |
| byZoneMsg: Zone definition, the size of the zone is zone count multiply 16 bytes. See in “1.7 global zone message format”. |
| 返回值 | >=0: The number of the packets  -1: Invalid data object handle |
| 其它说明 |  |

CP5200\_ParseSetGlobalZoneRet

|  |  |
| --- | --- |
| int CP5200\_ParseSetGlobalZoneRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of set global message command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakePushUserVarData

|  |  |
| --- | --- |
| int CP5200\_MakePushUserVarData( HOBJECT hObj, byte byOption , byte byVarZoonNum , byte byVarDataLen , byte\* pVarNoAndData ) | |
| Description | Make push and user variable command data |
| Parameter | hObj: Handle of communication data object |
| byOption:  Bit0:Whether to save all the variable to the FLASH  0:Not Save 1:Save  Bit1: Push direction. 0:push back 1:push forward Bit2~3: Reserved, set to 0.  Bit4~7: Push count. +1 is the push of zoon number. |
| byVarZoonNum: Zoon number.  Bit0~6:the zoon numbe which to be pushed:1~100  Bit7: Reserved, please set 0. |
| byVarDataLen: Variable data length.Sort every variable byte data in alphabet order.The total length of variable number and data is (1+n)byte. |
| pVarNoAndData: Variable No and data. The first byte is variable No, followed by a specify length data. |
| Return | >=0: The number of the packets  -1: Invalid data object handle |
| Note |  |

CP5200\_ParsePushUserVarRet

|  |  |
| --- | --- |
| int CP5200\_ParsePushUserVarRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of push and user variable command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeTimerCtrlData

|  |  |
| --- | --- |
| int CP5200\_MakeTimerCtrlData( HOBJECT hObj, byte byTimerNo , byte byCmd , byte byProp ,  DWORD dwValue ) | |
| Description | Make timer ctrlon command data |
| Parameter | hObj: Handle of communication data object |
| byTimerNo: Timer no,set the Timer by byte,1 is activity Bit0: Timer 1.  Bit1: Timer 2  Bit3: Timer 3  Bit4: Timer 4  Bit5: Timer 5  Bit6: Timer 6.  Bit7: Timer 7. |
| byCmd: Action。 1： Initializtion Timer  2： Reset Timer  3： Start Timer 4： Puse Timer  Other：Reserved |
|  | byProp: Property. Have different meaning according to the action.  When the action is initialize the time:  Bit0: 0 Time, 1 count down  Bit1: 0 pause, 1 start immediately  Bit2~3: Reserved  Bit4~7: time count  Set to 0 when the action is other. |
| dwValue: Value. Have different meaning according to the action.  When the action is initialize the time:  The initialization value when count down, in seconds.  High byte previous.  Set to 0 when timing.  Set to 0 when the action is other. |
| Return | >=0: The numbe of the packets.  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseTimerCtrlRet

|  |  |
| --- | --- |
| int CP5200\_ParseTimerCtrlRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of timer crtlon command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeSetZoneAndVariableData

|  |  |
| --- | --- |
| int CP5200\_MakeSetZoneAndVariableData(HOBJECT hObj, const BYTE\* pZoneData, int nZoneLen, const BYTE\* pVariableData, int nVarLen, WORD wCtrl, WORD wReserved) | |
| Description | Make set global zone and user variable value data |
| Parameter | hObj: Handle of communication data object |
| pZoneData：The global zone data. Including the zone Options, the number of zone, zone number, the zone defined. |
| nZoneLen：The global zone data length |
| pVariableData：Variable data, including variable options, variable data and  cross-district allows ,the length of the variable data table, the variable number and data |
| nVarLen：The variable data length |
| wCtrl：Effective control parameters play times, high byte first.  The value of 0 has been effective .  Bit15: Resvered, fill 0.  Bit0~14: Display times. |
| wReserved：resvered |
| Return | >=0: The numbe of the packets.  -1: Invalid data object handle |
| Note | After use this conmand, the global zone to be automatic into synchronous display. |

CP5200\_ParseSetZoneAndVariableRet

|  |  |
| --- | --- |
| int CP5200\_ParseSetZoneAndVariableRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse the return data of set global zone and user variable value |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success |
|  | -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeSendPureTextData

|  |  |
| --- | --- |
| int CP5200\_MakeSendPureTextData(HOBJECT hObj, int nWndNo, const char \*pText, COLORREF crColor, int nFontSize, int nSpeed, int nEffect, int nStayTime, int nAlignment); | |
| Description | Make send pure text data |
| Parameter | hObj: Handle of communication data object |
| nWndNo: Window sequence number, valid values 0 to 7 |
| pText: Text to be send |
| crColor: Text color |
| nFontSize: font size and style，see 1.7. Font size code and font style |
| nSpeed: Effect speed  0～100：The fastest value of 0。 |
| nEffect: Show effect。See the "1.5" section. |
| nStayTime: Stay time in second |
| nAlignment: The level of alignment  0: left Alignment  1: center Alignment  2: right Alignment |
| Return | >=0: The numbe of the packets.  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseSendPureTextRet

int CP5200\_ParseSendPureTextRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength)

|  |  |
| --- | --- |
| Description | Parse the return data of send pure text data |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CP5200\_MakeSendMultiProtocol

|  |  |
| --- | --- |
| int CP5200\_MakeSendMultiProtocol(HOBJECT hObj, int nItem, const BYTE \*pText, int nLength) | |
| Description | Make send multi protocol data |
| Parameter | hObj: Handle of communication data object |
| nItem: Items of multi protocol |
| pText: Datas of multi protocol，see《 C-Power external calls communication protocol》send multi protocol data CC=0x60 Data item |
| nLength:Length of datas |
| Return | >=0: The numbe of the packets.  -1: Invalid data object handle |
| Note |  |

CP5200\_ParseSendMultiProtocolRet

|  |  |
| --- | --- |
| int CP5200\_ParseSendMultiProtocoltRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse the return data of send multi protocol data |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

# Template data communication API function

6.1、Overview of template data communication API functions

|  |  |  |
| --- | --- | --- |
| No | API function name | Description |
| 1 | CPowerBox\_MakeSetProgramTemplateData CPowerBox\_MakeSetProgramTemplateData1 | Make set program template command data |
| 2 | CPowerBox\_ParseSetProgramTemplateRet | Pare return data of set program template command |
| 3 | CPowerBox\_MakeInOutProgramTemplateData | Make the in or out program template command data |
| 4 | CPowerBox\_ParseInOutProgramTemplateRet | Parse return data of in or out program template command |
| 5 | CPowerBox\_MakeQueryProgramTemplateData CPowerBox\_MakeQueryProgramTemplateData1 | Make the query program  template data |
| 6 | CPowerBox\_ParseQueryProgramTemplateRet | Parse the return data of query program template data. |
| 7 | CPowerBox\_MakeDeleteProgramData | Make delete program command data. |
| 8 | CPowerBox\_ParseDeleteProgramRet | Parse return data of delete program command |
| 9 | CPowerBox\_MakeSendTextData | Make send text command data |
| 10 | CPowerBox\_ParseSendTextRet | Parse return data of send text command |
| 11 | CPowerBox\_MakeSendPictureData | Make send picture command data |
| 12 | CPowerBox\_ParseSendPictureRet | Parse return data of send picture |
|  |  | command |
| 13 | CPowerBox\_MakeSendClockOrTemperatureData | Make send clock and temperature command data |
| 14 | CPowerBox\_ParseSendClockOrTemperatureRet | Parse return data of send clock and temperature command |
| 15 | CPowerBox\_MakeSetAloneProgramData | Make set alone program command data |
| 16 | CPowerBox\_ParseSetAloneProgramRet | Parse return data of set alone program command |
| 17 | CPowerBox\_MakeQueryProgramData | Make query program command data |
| 18 | CPowerBox\_ParseQueryProgramRet | Parse return data of query program command |
| 19 | CPowerBox\_MakeSetProgramPropertyData | Make set program property command data |
| 20 | CPowerBox\_ParseSetProgramPropertyRet | Parse return data of set program property command |
| 21 | CPowerBox\_MakeSetScheduleData | Make set schedule command data |
| 22 | CPowerBox\_ParseSetScheduleRet | Parse return data of set set schedule command |
| 23 | CPowerBox\_MakeDeleteScheduleData | Make delete schedule command data |
| 24 | CPowerBox\_ParseDeleteScheduleRet | Parse return data of delete schedule command |
| 25 | CPowerBox\_MakeGetScheduleData | Make get schedule command data |
| 26 | CPowerBox\_ParseGetScheduleRet | Parse return data of get schedule command |

Usage：

Step 1: Create template communication data object

Step 2: Make communication data, include RS232/485’s code convert

(0xa5 => 0xaa 0x05, …), or network ID code Step 3: Get the number of packets in the object

Step 4: One by one to handle each packet of data by the following manner：

1. Send the packet data to the controller
2. Receive data from controller, and process code convert

(0xaa 0x05 => 0xa5, …)

1. Parse the return data and get the result

Step 5: Destroy template communication data object

6.2、Detail of template data communication base API functions

CPowerBox\_MakeSetProgramTemplateData

|  |  |
| --- | --- |
| int CPowerBox\_MakeSetProgramTemplateData(HOBJECT hObj, byte byColor ,USHORT nWidth ,  USHORT nHeight , byte nWndNum , byte \*byDefParam , byte\* pWndParam); | |
| Description | Make set program template command data |
| Parameter | hObj: Handle of communication data object |
| byColor: Bit0: Red mark  Bit1: Green mark  Bit2: Blue mark  Bit3: Reserved  Bit4～6: Gray level  0: 2 level gray，7: 256 level gray  Bit7: Reserved |
| nWidth: The width of the screen，high byte previous |
| nHeight: The height of the screen，high byte previous |
| nWndNum: The display window number,the maximum number is 10 |
| byDefParam: Default parameter。  Byte0~1: Stay time in second. High byte previous.  Byte2: Speed。The smaller the faster.  Byte3: Font size. See “Font size code”  Byte4: Font color. See “Font color code”  Byte5: Show effect See”Show effect code”  Byte6: Picture type. See”Picture type code”  Byte7: Clock Format. See “Clock format and content”  Byte8: Clock content. See “Clock format and content” |
| pWndParam: Window parameter. Each window has a 16 bytes length parameter. The total length of the data is: the number of the window\*16. You can see the detail at “appendix:1 window position and property” |
| Return | >=0: The number of the packets  -1: Invalid data object handle |
| Note |  |

CPowerBox\_MakeSetProgramTemplateData1

|  |  |
| --- | --- |
| int CPowerBox\_MakeSetProgramTemplateData(HOBJECT hObj, byte byColor ,USHORT nWidth ,  USHORT nHeight , byte nWndNum , BYTE byOption, byte \*byDefParam , byte\* pWndParam); | |
| Description | Make set program template command data |
| Parameter | hObj: Handle of communication data object |
| byColor: Bit0: Red mark  Bit1: Green mark  Bit2: Blue mark  Bit3: Reserved  Bit4～6: Gray level  0: 2 level gray，7: 256 level gray  Bit7: Reserved |
| nWidth: The width of the screen，high byte previous |
| nHeight: The height of the screen，high byte previous |
| nWndNum: The display window number,the maximum number is 10 |
| byOption:  Bit0: Forced into the program template run  Bit1: Save the template position. 0: user disk, 1: system disk.  If the template is saved to the system tray, the original template of the user tray is cleared; if the template is saved to the user's disk, the original template of the system disk is cleared。  Bit2~7: Reserved |
| byDefParam: Default parameter。  Byte0~1: Stay time in second. High byte previous.  Byte2: Speed。The smaller the faster.  Byte3: Font size. See “Font size code”  Byte4: Font color. See “Font color code”  Byte5: Show effect See”Show effect code”  Byte6: Picture type. See”Picture type code”  Byte7: Clock Format. See “Clock format and content”  Byte8: Clock content. See “Clock format and content” |
| pWndParam: Window parameter. Each window has a 16 bytes length parameter. The total length of the data is: the number of the window\*16. You can see the detail at “appendix:1 window position and property” |
| Return | >=0: The number of the packets  -1: Invalid data object handle |
| Note |  |

CPowerBox\_ParseSetProgramTemplateRet

|  |  |
| --- | --- |
| int CPowerBox\_ParseSetProgramTemplateRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength); | |
| Description | Parse return data of set program template command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CPowerBox\_MakeInOutProgramTemplateData

|  |  |
| --- | --- |
| int CPowerBox\_MakeInOutProgramTemplateData(HOBJECT hObj,byte byInOrOut ); | |
| Description | Make in or out program template command data |
| Parameter | hObj: Handle of communication data object |
| byInOrOut: In or Out。 1: In the program template  0: Out the program template |
| Return | >=0: The number of the packets  -1: Invalid data object handle |
| Note |  |

CPowerBox\_ParseInOutProgramTemplateRet

int CPowerBox\_ParseInOutProgramTemplateRet(HOBJECT hObj, const BYTE\* pBuffer, int

|  |  |
| --- | --- |
| nLength); | |
| Description | Parse return data of in or out program template command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CPowerBox\_MakeQueryProgramTemplateData

|  |  |
| --- | --- |
| int CPowerBox\_MakeQueryProgramTemplateData(HOBJECT hObj ); | |
| Description | Make query program template command data |
| Parameter | hObj: Handle of template communication data object to de destroyed |
| Return | >=0: The number of the packets  -1: Invalid data object handle |
| Note |  |

CPowerBox\_MakeQueryProgramTemplateData1

|  |  |
| --- | --- |
| int CPowerBox\_MakeQueryProgramTemplateData(HOBJECT hObj , byte byFlag ); | |
| Description | Make query program template command data |
| Parameter | hObj: Handle of communication data object |
| byFlag:  Bit0: Whether to query program template status parameter  Bit1:Whether to return the template definition color gray, screen size information  Bit2~7: Reserved |
| Return | >=0: The number of the packets |
|  | -1: Invalid data object handle |
| Note |  |

CPowerBox\_ParseQueryProgramTemplateRet

|  |  |
| --- | --- |
| int CPowerBox\_ParseQueryProgramTemplateRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength ,BYTE\* pInfoBuffer, int nInfoBufSize); | |
| Description | Parse return data of query program template command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| pInfoBuffer: The return information |
| nInfoBufSize:Length of the return information |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

“pInfoBuffer” have the following meanings:

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x83 | 1 | Describe the package is the return data of query program template status parameter. |
| Options |  | 1 | The same value with send value of “Options”. |
| Template mode |  | 1 | 0: Not program template  1: program template |
| Template status |  | 1 | Bit0~1: template availability  0: the template is not available  1: the template can be used others: Reserved  Bit2~7: Reserved |
| Color gray |  | 1 | Color and gray。  Same with define“set program template” |
| Screen width |  | 2 | High byte first |
| Screen height |  | 2 | High byte first |
| Window count |  | 1 | Play window count。  Supports up to 10 play windows |

CPowerBox\_MakeDeleteProgramData

|  |  |
| --- | --- |
| int CPowerBox\_MakeDeleteProgramData(HOBJECT hObj,byte byConfig , byte byProNum , byte\* pDelPro ); | |
| Description | Make delete program command data |
| Parameter | hObj: Handle of communication data object |
| byConfig:  Bit0: The range of the delete program  0：Delete all program  1：Delete the specify program  Other: Reserved |
| byProNum: The program number. Do not need this item when delete all the programs. |
| pDelPro: The delete program list. Each program is represent by 1 byte, start from 1. |
| Return | >=0: The number of the packets  -1: Invalid data object handle |
| Note |  |

CPowerBox\_ParseDeleteProgramRet

|  |  |
| --- | --- |
| int CPowerBox\_ParseDeleteProgramRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength); | |
| Description | Parse return data of delete program command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CPowerBox\_MakeSendTextData

|  |  |
| --- | --- |
| int CPowerBox\_MakeSendTextData(HOBJECT hObj, DWORD dwAppendCode , byte byProNo , byte byWndNo , byte byProp , byte \*byShowFormat , char\* pText); | |
| Description | Make send text command data |
| Parameter | hObj: Handle of communication data object |
| dwAppendCode: The user’s append code, high byte previous. |
| byProNo: Program No.,Valid value:1~255 |
| byWndNo: Window No. Valid value:1～10 , Invalid when out of program template definition. |
| byProp: Property，Bit0~3: Text type 0：Common Text  Bit4: Display format. 0: default format 1:specify format  Bit5~7: Reserved |
| byShowFormat: Show format. Do not need this item when the property’s display format is 0.  Byte0~1: Stay time,High byte previous.  Byte2: Speed. The smaller the faster.  Byte3: Font size. See “Font size code”  Byte4: Font color. See “Font color code”  Byte5: Show effect See”Show effect code”  Byte6: Reserved  Byte7: Reserved |
| pText: Text data, end with ‘0x00’ |
| Return | >=0: The number of the packets  -1: Invalid data object handle |
| Note |  |

CPowerBox\_ParseSendTextRet

|  |  |
| --- | --- |
| Int CPowerBox\_ParseSendTextRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength); | |
| Description | Parse the return data of send text command |
| Parameter | hObj: Handle of communication data object |
|  | pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CPowerBox\_MakeSendPictureData

|  |  |
| --- | --- |
| int CPowerBox\_MakeSendPictureData(HOBJECT hObj,DWORD dwAppendCode , byte byProNo , byte byWndNo , byte byPicType , byte \*byShowFormat , byte\* pPicData , long lPicDataLen); | |
| Description | Make send picture command data |
| Parameter | hObj: Handle of communication data object |
| dwAppendCode: The user’s append code, high byte previous. |
| byProNo: Program No.,Valid value:1~255 |
| byWndNo: Window No. Valid value:1～10 , Invalid when out of program template definition. |
| byPicType: Picture type. Bit0~3: Picture type  1: Data of GIF picture file which include the information of the picture’s width and height so on.  2: The stored GIF filename in the contrl card.  4. Simple picture data, Check the format information at ”Simple Picture data format”  Bit4: Show format. 0 default format,1 specify format  Bit5~7: Reserved |
|  | byShowFormat: Show format.  Do not need this item when the property’s display format is 0.  Byte0~1: Stay time, High byte previous.  Byte2: Speed. The smaller the faster.  Byte3: Show effect See”Show effect code”  Byte4: Picture style(zoom、tile), see “Picture style code”  Byte5: Reserved  Byte6: Reserved  Byte7: Reserved |
| pPicData: Picture data. |
| lPicDataLen: Picture data length. |
| Return | >=0: The number of the packets -1: Invalid data object handle. |
| Note |  |

CPowerBox\_ParseSendPictureRet

|  |  |
| --- | --- |
| int CPowerBox\_ParseSendPictureRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength); | |
| Description | Parse return data of send picture command |
| Parameter | hObj: Handle of communication data object |
| Return | pBuffer: The return data |
| nLength: Length of the return data |
| 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CPowerBox\_MakeSendClockOrTemperatureData

int CPowerBox\_MakeSendClockOrTemperatureData(HOBJECT hObj,DWORD dwAppendCode , BYTE byProNo , BYTE byWndNo , BYTE byProgramType , UINT nPropLen , BYTE\* pProgramProp )

|  |  |
| --- | --- |
| Description | Make send clock and temperature command data |
| Parameter | hObj: Handle of communication data object |
| dwAppendCode: The user’s append code, high byte first. |
| byProNo: Program No.,Valid value:1~255 |
| byWndNo: Window No. Valid value:1～10 , Invalid when out of program template definition. |
| byProgramType: Program type Bit0~3: Type  2：Clock ; 3：Temperature Bit4: Display format.  0: default format 1:specify format  Bit5~7: Reserved, fill in 0 |
| nPropLen: Property length |
| pProgramProp：Program property  The meaning of the attribute data according to different types  Type = 2 , see Clock/Calendar type proprtey  Type = 3 , see Temperature and Humidity type proprtey |
| Return | >=0: The number of the packets -1: Invalid data object handle. |
| Note |  |

CPowerBox\_ParseSendClockOrTemperatureRet

|  |  |
| --- | --- |
| int CPowerBox\_ParseSendClockOrTemperatureRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength , BYTE\* pInfoBuffer, int nInfoBufSize ) | |
| Description | Parse return data of send clock and temperature command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| pInfoBuffer: The return information |
| nInfoBufSize:Length of the return information |
| Return | 0: Success |
|  | -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

“pInfoBuffer” have the following meanings:

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x87 | 1 | Describe the package is the return data which to show clock/temperature in the specified window of the specified program |
| Append code |  | 4 | The user’s append code, high byte previous. |
| Program No |  | 1 | The same value with send value “Program no”.  Valid value:1~100 |
| Window No |  | 1 | The same value with send value “Window no”. Valid value:1~10,Invalid when out of program template definition. |
| Packet loss number |  | 1 | The number of packets that have not yet received. Sends the first packet loss number is the total number of packets minus one. |
| The packet number of the packet loss |  | Variable-length | Packet loss packet number. Always in accordance with small to large; the first packet packet number is 0. Each package a byte. |

CPowerBox\_MakeSetAloneProgramData

|  |  |
| --- | --- |
| int CPowerBox\_MakeSetAloneProgramData(HOBJECT hObj, DWORD dwAppendCode , BYTE byProgramNo , BYTE byWindowCnt ,BYTE\* pWndParam, BYTE\* pWndData) | |
| Description | Make set alone program command data |
| Parameter | hObj: Handle of communication data object |
| dwAppendCode: The user’s append code, high byte first. |
| byProNo: Program No.,Valid value:1~255 |
| byWindowCnt: Window count. Valid value:1～10 |
|  | pWndParam：windows parameter  Every window information table has a 22 bytes length parameter. The 1~16 bytes are window position and property, You can see the detail at 1.13. Window position and property; The 17~19 bytes are window data offset; The 20~22 bytes are window data length. High byte first.  If no data ,then window data offset and window data length all are 0.  The total length of the data is: the number of the window\*22. |
| pWndData：Window play data:“Text”、“Picture”…  Byte 1：Data Type(1 Text；4 Picture)  Byte 2：Data Format（Like “Text type” in command 0x85 and “Picture type” in command 0x86）  Byte 3：Text data or picture data。 |
| Return | >=0: The number of the packets -1: Invalid data object handle. |
| Note |  |

CPowerBox\_ParseSetAloneProgramRet

|  |  |
| --- | --- |
| int CPowerBox\_ParseSetAloneProgramRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength ,  BYTE\* pInfoBuffer, int nInfoBufSize ) | |
| Description | Parse return data of set alone program command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| pInfoBuffer: The return information |
| nInfoBufSize:Length of the return information |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data  0x01 program template is invalid  0x11 program number is out of range  0x12 window number out of range  0x13 The definition of the window outside the screen size of the |
|  | program template definition  0x80 currently is not program template way |
| Note |  |

“pInfoBuffer” have the following meanings:

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x88 | 1 | Describe the package is the return data which to send alone program |
| Append code |  | 4 | The user’s append code, high byte previous. |
| Program No |  | 1 | Valid value:1~100 |
| Reserved |  | 1 | Reserved, fill in 0. |
| Packet loss number |  | 1 | The number of packets that have not yet received. Sends the first packet loss number is the total number of packets minus one. |
| The packet number of the packet loss |  | Variable-length | Packet loss packet number. Always in accordance with small to large; the first packet packet number is 0. Each package a byte. |

\*Must first send the first packet. Best to confirm the first packet sent successfully, and then send subsequent packets.

* The meaning of "return value" in the return packet:

0x01 program template is invalid

0x11 program number is out of range

0x12 window number out of range

0x13 The definition of the window outside the screen size of the program template definition 0x80 currently is not program template way

CPowerBox\_MakeQueryProgramData

|  |  |
| --- | --- |
| int CPowerBox\_MakeQueryProgramData(HOBJECT hObj , byte byFlag , byte\* pParam ) | |
| Description | Make query program command data |
| Parameter | hObj: Handle of communication data object |
| byFlag:Special which program info will to be query 1: Query valid programs count and program number 2: Query specifies program information.  Other: Reserved |
| pParam:  If “byFlag” is 1：byte1~5，resvered，fill 0  If “byFlag” is 2：：byte1，program number；byte2~5，resvered，fill 0 |
| Return | >=0: The number of the packets -1: Invalid data object handle. |
| Note |  |

CPowerBox\_ParseQueryProgramRet

|  |  |
| --- | --- |
| int CPowerBox\_ParseQueryProgramRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength ,BYTE\* pInfoBuffer, int nInfoBufSize ) | |
| Description | Parse return data of query program command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| pInfoBuffer: The return information |
| nInfoBufSize:Length of the return information |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

“pInfoBuffer” have the following meanings:

 Query“valid program count and program number”

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x89 | 1 | Describe the package is the return data packet of query program info |
| Info flag |  | 1 | Same with send value ”info flag” |
| parameters |  | 5 | Same with send value ”parameters” |
| Valid program count |  | 1 | Valid program count |
| Valid program number |  | Variable-length | Each byte identifies an effective program。  Valid value 1～100。 |

* The meaning of "return value" in the return packet:

0x01 Controller not running in program template mode

0x10 Unknown info flag

 Query specifies program information

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x89 | 1 | Describe the package is the return data packet of query program info |
| Info flag |  | 1 | Same with send value ”info flag” |
| parameters |  | 5 | Same with send value ”parameters” |
| Information count |  | 1 | Now only return one information |
| Program number |  | 1 | Program number |
| User append  code |  | 4 | User append code |

* The meaning of "return value" in the return packet:

0x01 Controller not running in program template mode

0x10 Unknown info flag

0x11 Invalid programs

0x12 Can’t get program information

CPowerBox\_MakeSetProgramPropertyData

|  |  |
| --- | --- |
| int CPowerBox\_MakeSetProgramPropertyData(HOBJECT hObj, byte byOption , byte byProgramCnt , byte\* pPrograms , byte byPropertyID1 , byte byPropertyID2 , byte byProgramLevel , USHORT nLoopCnt , USHORT nTime , byte\* pDuetime , byte\* pTimeInterval); | |
| Description | Make set program property command data |
| Parameter | hObj: Handle of communication data object |
| byOption:  Bit0: Set the range of the program property  0: All programes  1: Specify program  Other: Reserved |
| byProgramCnt:The count of the program |
| pPrograms: The list of the programes |
|  | ByPropertyID1：Property ID 1, marked which property you want to set by  byte, set 0 if the data not exist.  Bit0: The level of the program.  Bit1: The cycle count.  Bit2: Valid time. How long will the program be valid from now on.  Bit3: Interval time  Bit4~7: Reserved |
| ByPropertyID2: Property ID 2。Bit0~4: valid time. >0 the count of the valid time.<=4  Bit5~7: Reserved |
| byProgramLevel: The program level. 1～3 level, The high level of the program is priority. |
|  | nLoopCnt: Loop count, High byte previous(big-endian).  0: Do not play the program, use to shield program temporarily.  1~255: The loop count of the program. |
| nTime: Valid time. High byte previous (big-endian). In minute.  0: Not limit play time  >0: Specify play time in minute. |
| pDuetime: time limit |
| pTimeInterval:The interval time. The start tag “Hour/Minute/Second”and the end tag “Hour/Minute/Second” both represent by one byte. |
| Return | >=0: The number of the packets -1: Invalid data object handle. |
| Note |  |

CPowerBox\_ParseSetProgramPropertyRet

|  |  |
| --- | --- |
| int CPowerBox\_ParseSetProgramPropertyRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength); | |
| Description | Parse the return data of set program property command |
| Parameter | hObj: Handle of communication data object |
|  | pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data |
| Note |  |

CPowerBox\_MakeSetScheduleData

|  |  |
| --- | --- |
| int CPowerBox\_MakeSetScheduleData(HOBJECT hObj, DWORD dwAppendCode, BYTE byScheduleNo, const BYTE\* pProperty, const BYTE\* pBoxes, BYTE byBoxCnt); | |
| Description | Make set schedule command data |
| Parameter | hObj: Handle of communication data object |
| dwAppendCode: The user’s append code, high byte first. |
| byScheduleNo: Schedule number，Valid value 1~100。Total support 100 plans, For each plan No, the new data cover the old data |
| pProperty: play property，total 14 bytes： byte 0：Format and level： Bit0~3: Data format，fill in 0x01  Bit4~7: Indicates the priority level. The priority level the greater the value, the more priority to play, 0 is the lowest priority.。。  byte 1：Weekday：Bit0~6: 7-bit logo Sunday to Saturday byte 2~4 ： Begin date ， 3 bytes: Byte1:Year,Valid value0~99,means  2000~2999; Byte2:Month ;Byte3:Day  byte 5~7：End date，3 bytes: Byte1:Year,Valid value0~99,means 2000~2999;  Byte2:Month ;Byte3:Day  byte 8~10：Begin time, 3 bytes:Byte1:Hour；Byte2:Minute；Byte3:Second byte 11~13：End time, 3 bytes:Byte1:Hour；Byte2:Minute；Byte3:Second |
| pBoxes: program number , each byte represents a program. Numbered in  ascending order, do not repeat..... |
| byBoxCnt:program number count, Valid value:1～100, |

CPowerBox\_ParseSetScheduleRet

|  |  |
| --- | --- |
| int CPowerBox\_ParseSetScheduleRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of set set schedule command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data  0x01 program template is invalid  0x80 currently is not program template way |
| Note |  |

CPowerBox\_MakeDeleteScheduleData

|  |  |
| --- | --- |
| int CPowerBox\_MakeDeleteScheduleData(HOBJECT hObj, DWORD dwAppendCode, const BYTE\* pSchs, BYTE bySchCnt) | |
| Description | Make delete schedule command data |
| Parameter | hObj: Handle of communication data object |
| dwAppendCode: The user’s append code, high byte first. |
| pSchs: schedule number, Valid value 1~100。Each byte represents a play schedule。  When delete all play schedule, the length of this data is one , value is 0xff. |
| bySchCnt: The number of play schedule will to be delete。0 means delete all play plans. |

CPowerBox\_ParseDeleteScheduleRet

|  |  |
| --- | --- |
| int CPowerBox\_ParseDeleteScheduleRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength) | |
| Description | Parse return data of delete schedule command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data  0x01 program template is invalid  0x11 The number of play plan will to be delete is 0. 0x80 currently is not program template way |
| Note |  |

CPowerBox\_MakeGetScheduleData

|  |  |
| --- | --- |
| int CPowerBox\_MakeGetScheduleData(HOBJECT hObj, DWORD dwAppendCode, BYTE byType, BYTE byScheduleNo) | |
| Description | Make get schedule command data |
| Parameter | hObj: Handle of communication data object |
| dwAppendCode: The user’s append code, high byte first. |
| byType：0: Query all valid play plan.  1: Query specified play plan no  Other: Reserved |
| byScheduleNo: Valid value:1~100。When query type is 0，this data fill in 0。 |

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x8d | 1 | Describe the package is the return data which to query play plan |
| Append code |  | 4 | The user’s append code, high byte previous. |
| Query type |  | 1 | 0: Query all valid play plan.  1: Query specified play plan no  Other: Reserved |
| Count /Number |  | 1 | When query type is 0，this value is valid play schedule count  When query type is 1，this value is play schedule number. |

CPowerBox\_ParseGetScheduleRet

|  |  |
| --- | --- |
| int CPowerBox\_ParseGetScheduleRet(HOBJECT hObj, const BYTE\* pBuffer, int nLength, BYTE\* pInfoBuffer, int nInfoBufSize ) | |
| Description | Parse return data of get schedule command |
| Parameter | hObj: Handle of communication data object |
| pBuffer: The return data |
| nLength: Length of the return data |
| pInfoBuffer: The return information |
| nInfoBufSize:Length of the return information |
| Return | 0: Success  -1: Invalid data object handle  -2: Incorrect return data  -3: Incorrect length of return data 0x01 program template is invalid 0x11 Don’t support the query type.  0x12 Invalid play plan no.  0x80 currently is not program template way |
| Note |  |

“pInfoBuffer” have the following meanings:

|  |  |  |  |
| --- | --- | --- | --- |
| Play schedule number table/ play schedule content |  | Variable-length | When query type is 0，this value is valid play schedule number table  When query type is 1，this value is play schedule content. Data format like command 0x8B. |

You must deal with the return data according to the different query type.

The meaning of "return value" in the return packet:

0x01 program template is invalid 0x11 Don’t support the query type.

0x12 Invalid play plan no.

0x80 currently is not program template way

# Communication base API function

7.1、 Overview of RS232 communication base API

functions

|  |  |  |
| --- | --- | --- |
| No | API function name | Description |
| 1 | CP5200\_RS232\_Init | Initialize serial port parameters |
| 2 | CP5200\_RS232\_InitEx | Initialize serial port parameters and set timeout |
| 3 | CP5200\_RS232\_Open | Open serial port |
| 4 | CP5200\_RS232\_OpenEx | Open serial port，assigned reading and writing timeout |
| 5 | CP5200\_RS232\_Close | Close serial port |
| 6 | CP5200\_RS232\_IsOpened | Test whether the serial port has been opened |
| 7 | CP5200\_RS232\_Write | Write data to serial port |
| 8 | CP5200\_RS232\_Read | Read data from serial port |
| 9 | CP5200\_RS232\_WriteEx | Write data to serial port，and processing for transcoding |
| 10 | CP5200\_RS232\_ReadEx | Read data from serial port，and processing for transcoding |

Usage:

Step 1: Initialize serial port parameters

Step 2: Open serial port

Step 3: Read and write operations on the serial

Step 4: Close serial port

7.2、Detail of RS232 communication base API functions

CP5200\_RS232\_Init

|  |  |
| --- | --- |
| int CP5200\_RS232\_Init(const char \*fName, int nBaudrate) | |
| Description | Initialize serial port parameters |
| Parameter | fName: RS232 serial port name，for example:“COM1”、“COM2”、… |
| nBaudrate: baud rate，for example :115200、57600、... |
| Return | 1: success  0: fail |
| Note | Other serial port parameters are fixed:  Parity: No parity  Data bits: 8  Stop bits: 1  Flow Control: None |

CP5200\_RS232\_InitEx

|  |  |
| --- | --- |
| int CP5200\_RS232\_InitEx(const char \*fName, int nBaudrate, DWORD dwTimeout); | |
| Description | Initialize serial port parameters and set timeout |
| Parameter | fName: RS232 serial port name，for example:“COM1”、“COM2”、… |
| nBaudrate: baud rate，for example :115200、57600、... |
| dwTimeout; time of timeout |
| Return | 1: success  0: fail |
| Note | Other serial port parameters are fixed:  Parity: No parity  Data bits: 8  Stop bits: 1  Flow Control: None |

CP5200\_RS232\_Open

|  |  |
| --- | --- |
| int CP5200\_RS232\_Open(void) | |
| Description | Open serial port |
| Parameter | None |
| Return | 1: success  0: fail |
| Note | After using the serial port, need to call CP5200\_RS232\_Close () to close  Read, write, timeouts are set to 600 ms |

CP5200\_RS232\_OpenEx

|  |  |
| --- | --- |
| int CP5200\_RS232\_OpenEx(DWORD dwReadTimeout, DWORD dwWriteTimeout) | |
| Description | Open serial port，assigned reading and writing timeout |
| Parameter | dwReadTimeout: Reading timeout. Units ms |
| dwWriteTimeout: Writing timeout. Units ms |
| Return | 1: success  0: fail |
| Note | After using the serial port, need to call CP5200\_RS232\_Close () to close |

CP5200\_RS232\_Close

|  |  |
| --- | --- |
| int CP5200\_RS232\_Close(void) | |
| Description | Close serial port |
| Parameter | None |
| Return | 1: success  0: fail or the serial port is the closed state |
| Note |  |

CP5200\_RS232\_IsOpened

|  |  |
| --- | --- |
| int CP5200\_RS232\_IsOpened(void) | |
| 作用 | Test whether the serial port has been opened |
| 参数 | No |
| 返回值 | 1: Has been opened  0: No open |
| 其它说明 |  |

CP5200\_RS232\_Write

|  |  |
| --- | --- |
| int CP5200\_RS232\_Write(const void\* pBuf, int nLength) | |
| Description | Write data to serial port |
| Parameter | pBuf: Data buffer pointer |
| nLength: Data length |
| Return | 1: success  0: fail or the serial port is the closed state |
| Note |  |

CP5200\_RS232\_Read

|  |  |
| --- | --- |
| int CP5200\_RS232\_Read(void\* pBuf, int nBufSize) | |
| Description | Read data from serial port |
| Parameter | pBuf: Data buffer pointer, stored data of reading |
| nBufSize: Data Buffer size |
| Return | Data length |
| Note |  |

CP5200\_RS232\_WriteEx

int CP5200\_RS232\_WriteEx(const void\* pBuf, int nLength)

|  |  |
| --- | --- |
| Description | Write data from serial port，and processing for transcoding |
| Parameter | pBuf: Data buffer pointer |
| nLength: Data length |
| Return | 1: success  0: fail or the serial port is the closed state |
| Note | Add code “0XA5” at the beginning of the data and add code “0XAE” at the end of the data, send data of processing for transcoding  0xa5 => 0xaa 0x05  0xaa => 0xaa 0x0a  0xae => 0xaa 0x0e |

CP5200\_RS232\_ReadEx

|  |  |
| --- | --- |
| int CP5200\_RS232\_ReadEx(void\* pBuf, int nBufSize) | |
| Description | Read data from serial port，and processing for transcoding |
| Parameter | pBuf: pBuf: Data buffer pointer, stored data of reading |
| nBufSize: Data Buffer size |
| Return | Data length |
| Note | Read the data between beginning code “0xa5” and ending code  “0xae”,the return data not contain beginning code “0xa5” and ending code  “0xae”,and processing for transcoding the data of between beginning code  “0xa5” and ending code “0xae”  0xaa 0x05 => 0xa5  0xaa 0x0a => 0xaa  0xaa 0x0e => 0xae |

7.3、Overview of Network communication base API functions

|  |  |  |
| --- | --- | --- |
| No | API function name | Description |
| 1 | CP5200\_Net\_Init | Initialize network parameters |
| 2 | CP5200\_Net\_SetBindParam | Bind client IP and port |
| 3 | CP5200\_Net\_Connect | Open network connections |
| 4 | CP5200\_Net\_IsConnected | Test whether the network has been connected |
| 5 | CP5200\_Net\_Disconnect | Close network connections |
| 6 | CP5200\_Net\_Write | Write data to network |
| 7 | CP5200\_Net\_Read | Read data from network |

Usage:

Step 1: Initialize network parameters

Step 2: Open network connections

Step 3: Read and write operations network

Step 4: Close network connections t

7.4 、 Detail of network communication base API

functions

CP5200\_Net\_Init

|  |  |
| --- | --- |
| int CP5200\_Net\_Init(DWORD dwIP, int nIPPort, DWORD dwIDCode, int nTimeOut) | |
| Description | Initialize network parameters |
| Parameter | dwIP:IP address. For example: 192.168.1.100 is 0xc0a80164 |
| nIPPort:Port |
| dwIDCode:ID |
| nTimeOut:timeout |
| Return | 0 |
| Note |  |

CP5200\_Net\_SetBindParam

|  |  |
| --- | --- |
| int CP5200\_Net\_SetBindParam( DWORD dwClientIP , int nClientPort ) | |
| Description | Bind client IP and port |
| Parameter | dwClientIP: Bind client IP。For example： 192.168.1.100 is 0xc0a80164 |
| nClientPort: Bind client port |
| Return | 0 |
| Note |  |

CP5200\_Net\_Connect

|  |  |
| --- | --- |
| int CP5200\_Net\_Connect(void) | |
| Description | Open network connections |
| Parameter | No |
| Return | 1: Success  0: Failure  -1: IP is not valid |
| Note |  |

CP5200\_Net\_IsConnected

|  |  |
| --- | --- |
| int CP5200\_Net\_IsConnected(void) | |
| Description | Test whether the network has been connected |
| Parameter | No |
| Return | 1: Has been connected  0: No connect |
| Note |  |

CP5200\_Net\_Disconnect

|  |  |
| --- | --- |
| int CP5200\_Net\_Disconnect(void) | |
| Description | Close network connections |
| Parameter | No |
| Return | 1: Success  0: Failure or network has been turned off. |
| Note |  |

CP5200\_Net\_Write

|  |  |
| --- | --- |
| int CP5200\_Net\_Write(const BYTE\* pBuf, int nLength); | |
| Description | Write data to network |
| Parameter | pBuf: Data buffer pointer |
| nLength: Data length |
| Return | 1: Success  0: Failure  -1: Network is the closed state. |
| Note |  |

CP5200\_Net\_Read

|  |  |
| --- | --- |
| int CP5200\_Net\_Read(BYTE\* pBuf, int nSize) | |
| Description | Read data from network |
| Parameter | pBuf: Data buffer pointer, stored data of reading |
| nBufSize: Data Buffer size |
| Return | >0: Data length  0: Failure  -1: Network is the closed state. |
| Note |  |

# Running plan API function

C-Power5200 controller control running program by date ,week. Running plan is saved as file int the controller,the file name is “playbill.rsf” and it can’t change.

Running program API function in order to create “playbill.rsf” file.

8.1、Overview of running plan API functions

|  |  |  |
| --- | --- | --- |
| No | API function name | Description |
| 1 | CP5200\_Runsch\_Create | Create running plan object |
| 2 | CP5200\_Runsch\_Destroy | Destroy running plan object |
| 3 | CP5200\_Runsch\_AddItem | Add running plan item |
| 4 | CP5200\_Runsch\_SaveToFile | Save running plan to file |

Usage:

Step 1: Create running plan object

Step 2: Add running plan item

Step 3: Save running plan to file

Step 4: Destroy running plan object

8.2、Detail of running plan API functions

CP5200\_Runsch\_Create

|  |  |
| --- | --- |
| HOBJECT CP5200\_Runsch\_Create(int nPrgSum, int nAttrib) | |
| Description | Create running plan object |
| Parameter | nPrgSum: total number of program |
| nAttrib: Attribute  0: Default time period is not playing any program  1: Default time period is not playing all program |
| Return | Running plan object handle,called by this type of API  Return “NULL” is said failure to create |
| Note | Object to create successful and no longer in use, the object must be destroyed |

CP5200\_Runsch\_Destroy

|  |  |
| --- | --- |
| int CP5200\_Runsch\_Destroy(HOBJECT hObj) | |
| Description | Destroy running plan object |
| Parameter | hObj: The running plan object handle to be destroy |
| Return | 0: no error  -1: object handle is null  -2: wrong object handle |
| Note |  |

CP5200\_Runsch\_AddItem

|  |  |
| --- | --- |
| int CP5200\_Runsch\_AddItem(HOBJECT hObj, int nGrade, int nWeekDateRelative, int nWeeks, const int\* pBeginDate, const int\* pEndDate, const int\* pBeginTime, const int\* pEndTime, int nItemCnt, const int \*pItems) | |
| Description | Add running plan item |
| Parameter | hObj: The running plan object handle |
| nGrade: plan item level，0~9level, More higher-level priority。 |
| nWeekDateRelative: the relationship of date and week  0: Execute this plan must that all of week and date are satisfy  1: Execute this plan must that one of week and date are satisfy |
| nWeeks: week tag, value can be one or more of the following combination of values  1: Sunday  2: Monday  4: Tuesday  8: Wednesday  16: Thursday  32: Friday  64: Saturday |
| pBeginDate: Start date. Three integer values denote "Year" "Month"  "Day" respectively |
| pEndDate: End date. Three integer values denote "Year" "Month" "Day" respectively |
|  | pBeginTime: Start time. Three integer values denote "hour" "minute"  "second" respectively |
| pEndTime: End time. Three integer values denote "hour" "minute"  "second" respectively |
| nItemCnt:The number of program want to play。The number can’t greater than the number of the first parameter of the function  “CP5200\_Runsch\_Create” specified |
|  | pItems: The program number that will to be play.length is ”nItemCnt”  integer.Every integer is the number of program to be play.Program number started from 0, and the number less than the first parameter of the function “CP5200\_Runsch\_Create” |
| Return | >=0: success , plan item number  -1: Invalid object handle  -2: Error parameter  -3: Memory not enough  -4: Memory wrong |
| Note |  |

CP5200\_Runsch\_SaveToFile

|  |  |
| --- | --- |
| int CP5200\_Runsch\_SaveToFile(HOBJECT hObj, const char\* pFilename) | |
| Description | Save running plan to file |
| Parameter | hObj: The running plan object handle |
| pFilename: File path and name |
| Return | 0: no error  -1: Invalid object handle  -3: File operater fail |
| Note |  |

# Time-limite play information by week

C-Power5200 controller support play by period of time, Time-limite informatin is saved as file and its name is “playbill.lpt” , the file format as below:

|  |
| --- |
| File head |
| The frist record |
| … |
| The n record |

File head’s length is 7 bytes and every recor’s length is 7 bytes two.

The file only record the time-limite information of time-limite program.Program of always played do not need to record any information in this file.

9.1、Detail of file head

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0x00 | File  ID | | Format version number | | Record number | | Reservations |

Description：

|  |  |  |
| --- | --- | --- |
| Date name | Data size(byte) | Description |
| File ID | 2 | Fixed for the "LT"。 |
| Format version numbe | 2 | 0x0100(the frist byte is 0x00,the second byte is 0x01) |
| Record number | 2 | number of time-limite players recorded information， Low byte first。 |

# 9.2、Detail definition of time-limite play information by week

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0x00 | Program number | | week | Begin minute | Begin hour | End minute | End hour |

Description：

|  |  |  |
| --- | --- | --- |
| Data name | Data size(byte) | Description |
| Program number | 2 | Program number, started from 0. |
| week | 1 | Limited by week,use 7 bits, Each bit denote one day. If the day need to play,set corresponding bit to 1 . |
|  |  | Sunday：0x01  Monday：0x02  Tuesday：0x04  Wednesday：0x08  Thursday：0x10  Friday：0x20  Saturday：0x40 |
| Begin minute | 1 | Begin play time: minute(0~59) |
| Begin hour | 1 | Begin play time: hour (0~23) |
| End minute | 1 | End play time: minute (0~59) |
| End hour | 1 | End play time: hour (0~23) |

# Multi-window control API function

10.1、Overview of RS232 multi-window control API function

|  |  |  |  |
| --- | --- | --- | --- |
| No | API function name | Description | |
| 1 | CP5200\_RS232\_SplitScreen | | Send split window command |
| 2 | CP5200\_RS232\_SendText  CP5200\_RS232\_SendText1 | | Send text to special window |
| 3 | CP5200\_RS232\_SendTagText  CP5200\_RS232\_SendTagText1 | | Send tag text to special window |
| 4 | CP5200\_RS232\_SendPicture | | Send picture to special window |
| 5 | CP5200\_RS232\_SendStatic | | Send static text to special window |
| 6 | CP5200\_RS232\_SendClock | | Send clock to special window |
| 7 | CP5200\_RS232\_ExitSplitScreen | | Exit split window command |
| 8 | CP5200\_RS232\_SaveClearWndData | | Save or clear split window mesage |
| 9 | CP5200\_RS232\_PlaySelectedPrg  CP5200\_RS232\_PlaySelectedPrg1 | | Select play stored program |
| 10 | CP5200\_RS232\_SetUserVarData | | Set user variable |
| 11 | CP5200\_RS232\_SetSelectedAndUserVarData | | Set selected and user var command |
| 12 | CP5200\_RS232\_SetGlobalZone | | Set global message command |
| 13 | CP5200\_RS232\_PushUserVarData | | Push user data command |
| 14 | CP5200\_RS232\_TimerCtrl | | Set Timer contrl command |
| 15 | CP5200\_RS232\_SetZoneAndVariable | | Set global zone and user variable |
| 16 | CP5200\_RS232\_SendPureText | | Send pure text to special window |

Usage:

Step 1: Initialize serial port parameters

Only record the serial parameter initialization parameter information, not the actual serial port operation。

Step 2: Send split window command，

If the window has been divided and have been met requirements, this step can be dispensed with, or to send the split window command。 Step 3: Send text or picture to window。

Note: This category interface need not to consider whether the serial port has been opened , as long as the serial port parameters have been initialized.。

10.2 、 Detail of RS232 multi-window control API

function

CP5200\_RS232\_SplitScreen

|  |  |
| --- | --- |
| int CP5200\_RS232\_SplitScreen(int nCardID, int nScrWidth, int nScrHeight, int nWndCnt, const int \*pWndRects) | |
| Description | Send split window command |
| Parameter | nCardID: Controller ID |
| nScrWidth: the width of screen |
| nScrHeight: the height of screen |
| nWndCnt: The window number of the screen will be splitted , valid values 1~8。 |
| pWndRects: Window coordinates, each window with four integer said the  "left, up,right,down” coordinates,ave the same data structure with the  "RECT"of windows。 |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port |
|  | -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error  -9: The window number was too much |
| Note | This function sets sub-windows information and sends split-screen commad. |

CP5200\_RS232\_SendText

(CP5200\_RS232\_SendText1)

|  |  |
| --- | --- |
| int CP5200\_RS232\_SendText(int nCardID, int nWndNo, const char \*pText, COLORREF crColor, int nFontSize, int nSpeed, int nEffect, int nStayTime, int nAlignment); | |
| Description | Send text to special window |
| Parameter | nCardID: Controller ID |
| nWndNo: Window sequence number, valid values 0 to 7 |
| pText: The text will to be sent |
| crColor: Text color。 |
| nFontSize: font size and style，see 1.7. Font size code and font style，this parameter only support the font size, does not support multiple font |
| nSpeed: Effect speed  0～100：The fastest value of 0。 |
| nEffect: Show effect。See the "1.5" section. |
| nStayTime: Stay time in second |
| nAlignment: The level of alignment  0: left Alignment  1: center Alignment  2: right Alignment |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note | CP5200\_RS232\_SendText1 is for single byte characters, ASCII and extended ASCII. |

CP5200\_RS232\_SendTagText (CP5200\_RS232\_SendTagText1)

|  |  |
| --- | --- |
| int CP5200\_RS232\_SendTagText(int nCardID, int nWndNo, const char \*pText, COLORREF crColor, int nFontSize, int nSpeed, int nEffect, int nStayTime, int nAlignment) | |
| Description | Send tag text to specify window |
| Parameter | nCardID: Controller ID |
| nWndNo: Window sequence number, valid values 0 to 7 |
| pText: The text which to be sent |
| crColor: Text color。 |
| nFontSize: font size and style，see 1.7. Font size code and font style，this parameter only support the font size, does not support multiple font |
| nSpeed: Effect speed  0～100：The fastest value of 0。 |
| nEffect: Show effect。See the "1.5" section. |
| nStayTime: Stay time in second |
|  | nAlignment: The level of alignment  0: left Alignment 1: center Alignment 2: right Alignment: |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note | CP5200\_RS232\_SendTagText1 is for single byte characters, ASCII and extended ASCII. |

CP5200\_RS232\_SendPicture

|  |  |
| --- | --- |
| int CP5200\_RS232\_SendPicture(int nCardID, int nWndNo, int nPosX, int nPosY, int nCx, int nCy, const char \*pPictureFile, int nSpeed, int nEffect, int nStayTime, int nPictRef) | |
| Description | Send picture to special window |
| Parameter | nCardID: Controller ID |
| nWndNo: Window sequence number, valid values 0 to 7 |
| nPosX: Began to show the location of X coordinate. Relative upper-left corner the window.。 |
| nPosY: Began to show the location of Y coordinate. Relative upper-left corner the window. |
| nCx: The width of picture |
| nCy: The heigth of picture |

|  |  |
| --- | --- |
|  | pPictureFile: Path and file name of the picture file ,this is based on the value of nPictRef.  When the value of nPictRef is 0: pPictureFile is the Path and file name of the file on the computer.  When the value of nPictRef is 1: pPictureFile is the Path and file name of the GIF file on the controller card.  When the value of nPictRef is 2: pPictureFile is the Path and file name of the file on the computer.  When the value of nPictRef is 3: pPictureFile is the Path and file name of picture packages and the serial number of the picture on the controller card.  Packages name followed by is separated by a space. For example:  “images.rpk 1” |
| nSpeed: Effect speed  0～100：The fastest value of 0。 |
| nEffect: Show effect。See the "1.5" section. |
|  | nStayTime: Stay time in second |
| nPictRef: the way to send picture and meaning.  0：display the local picture that will be converted into the format of GIF to send.  1：display the gif picture that on the controller card.  2：display the local picture that will be converted into the format of simple to send.  3：display the picture in the picture packages that on the controller card.  Other values: deal with 0. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract |
|  | -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error  -9: Param “nWndNo” is wrong  -10: Image file does not exist  -11: The specified file is not available to support the image file |
| Note | Final image is converted to 256 color pictures to send, if given a true color image, there may be color changes.  Image size will be stretched or compressed to fit the size of the specified  window。 |

CP5200\_RS232\_SendSimpleImageData

|  |  |
| --- | --- |
| int CP5200\_RS232\_SendSimpleImageData(int nCardID, int nWndNo, int nPosX, int nPosY, const char \*pPictureFile, int nSpeed, int nEffect, int nStayTime, BYTE\* pPicData , long lPicDataLen) | |
| Description | Send simple picture to special window |
| Parameter | nCardID: Controller ID |
| nWndNo: Window sequence number, valid values 0 to 7 |
| nPosX: Began to show the location of X coordinate. Relative upper-left corner the window.。 |
| nPosY: Began to show the location of Y coordinate. Relative upper-left corner the window. |
| nSpeed: Effect speed  0～100：The fastest value of 0。 |
| nEffect: Show effect。See the "1.5" section. |
| nStayTime: Stay time in second |
| pPicData: simple picture data,see the 1.11 simple picture data fomart. |
| lPicDataLen:the length of simple picture data. |
| Return | 0: Success |
|  | -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error  -9: Param “nWndNo” is wrong  -10: Image file does not exist  -11: The specified file is not available to support the image file |
| Note | Final image is converted to 256 color pictures to send, if given a true color image, there may be color changes.  Image size will be stretched or compressed to fit the size of the specified  window。 |

CP5200\_RS232\_SendStatic

|  |  |
| --- | --- |
| int CP5200\_RS232\_SendStatic(int nCardID, int nWndNo, const char \*pText, COLORREF crColor, int nFontSize, int nAlignment, int x, int y, int cx, int cy) | |
| Description | Send static text to special window |
| Parameter | nCardID: Controller ID |
| nWndNo: Window sequence number, valid values 0 to 7 |
| pText: Text to be send |
| crColor: Text color。 |
| nFontSize: font size and style，see 1.7. Font size code and font style |
| nAlignment: The level of alignment  0: left Alignment  1: center Alignment  2: right Alignment |
|  | x: Start X of the play window |
| y: Start Y of the play window |
| cx: The width of play window |
|  | cy: The height of play window. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note | Content outside the region remain unchanged |

CP5200\_RS232\_SendClock

|  |  |
| --- | --- |
| CP5200\_RS232\_SendClock( int nCardID, int nWinNo , int nStayTime , int nCalendar , int nFormat , int nContent , int nFont , int nRed , int nGreen , int nBlue , LPCSTR pTxt ); | |
| Description | Send clock to special window |
| Parameter | nCardID: Controller ID |
| nWndNo: Window sequence number, valid values 0 to 7 |
| nStayTime: Stay time in second。 |
| nCalendar: Calendar 0: Gregorian calendar date and time  1: Lunar date and time  2: Chinese lunar solar terms  3: Lunar time and date + Solar Terms |

|  |  |
| --- | --- |
|  | nFormat: Format bit 0: when the system (0: 12 hour; 1: 24 hours system) bit 1: Year digit (0: 4; 1: 2) bit 2: Branch (0: single; 1: multi-line) bit 3~5: Format control, such as the November 12, 2010 Friday , according to diffenert values expressed as:  0: 2010/11/12 Friday 16:20:30  1: Fri，12/11/2010 16:20:30  2: 2010-11-12 Fri. 16:20:30  3: Friday，12 November 2010 16:20:30 4: Fri，Nov 12,2010 16:20:30  5: Friday，November 12 2010 16:20:30  6: Fri，11/12/2010 16:20:30 7: 2010/11/12，Fri.16:20:30 bit 6: show hands,marks bit 7: Transparent |
| nContent: Content  By bit to determine the content to display.  bit 7: Pointer bit 6: weeks bit 5: seconds bit 4: minute bit 3: hour bit 2: day bit 1: month bit 0: year |
| nFont: Font，Bit0~3: font size |
|  | nRed: The red color component |
| nGreen: The red green component |
| nBlue: The red blue component |
| pTxt: Text string to the end of 0x00. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data |
|  | -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CP5200\_RS232\_ExitSplitScreen

|  |  |
| --- | --- |
| int CP5200\_RS232\_ExitSplitScreen( int nCardID ); | |
| Description | Exit split window command |
| Parameter | nCardID: Controller ID |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| 其它说明 |  |

CP5200\_RS232\_SaveClearWndData

|  |  |
| --- | --- |
| int CP5200\_RS232\_SaveClearWndData( int nCardID , int nSavaOrClear ); | |
| Description | Save or clear split window mesage |
| Parameter | nCardID: Controller ID |
| nSavaOrClear：Save or clear data 0: Save data to the flash.  1: Clear data from the flash. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port |
|  | -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CP5200\_RS232\_PlaySelectedPrg

|  |  |
| --- | --- |
| int CP5200\_RS232\_PlaySelectedPrg(int nCardID, const WORD \*pSelected, int nSelCnt, int nOption) | |
| Description | Select play stored program |
| Parameter | nCardID: Controller ID |
| pSelected：The program number array of be selected to play |
| nSelCnt：The program count of be selected |
| nOption：Whether to save select message to the flash  0：No save  1：Save |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CP5200\_RS232\_PlaySelectedPrg1

int CP5200\_RS232\_PlaySelectedPrg1(int nCardID, const WORD \*pSelected, int nSelCnt, int

|  |  |
| --- | --- |
| nOption, int nScrWidth , int nScrHeight , byte byColorGray , byte nWndCnt) | |
| Description | Select play stored program |
| Parameter | nCardID: Controller ID |
| pSelected：The program number array of be selected to play |
| nSelCnt：The program count of be selected |
| nOption：Whether to save select message to the flash  0：No save  1：Save |
| nScrWidth：Screen width |
| nScrHeight：Screen height |
| byColorGray：color gray |
| nWndCnt：window count |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CP5200\_RS232\_SetUserVarData

|  |  |
| --- | --- |
| int CP5200\_Rs232\_SetUserVarData(int nCardID, int bSave , int nVarNum , int bAstride , int\* nWarLen , byte\* byNoData ); | |
| Description | Set user variable |
| Parameter | nCardID: Controller ID |
| bSave: Bit0:Whether to save all variables to the flash 0:No save，1:Save。  Bit1~7: Reserved,set to 0 |
| nVarNum: Variable number |
|  | bAstride: Whether to allow cross-variable zone setting. 0 is not permitted;  1 is permit |
| nVarLen：Bytes of data specified for each variable. |
| byNoData：Specified number of variables and variable data for each variable, the first byte of each variable is the variable number, followed by a specified length of variable data. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note | Corresponds to a variable number of each variable area size of each variable region is 32 bytes. Multiple continuous variables can be linked to a variable area used,occupied area of the variable number of variables can not be used。  When does not allow cross-variable area, more than 32 bytes of data are discarded；When allow cross-variable area,calculate the length of the data area to use the number of variables.  Valid values for the variable number is 1~100。Number of variables corresponding to each variable area can store 32 bytes of data, a number of continuous variable area can be used together for a variable, the variable area occupied number of variables can not be used。  When variable values are not updated and just save the variable value to the FLASH, it can set the " nVarNum " of the value of 0, set the " bSave " to save |

CP5200\_RS232\_SetSelectedAndUserVarData

|  |  |
| --- | --- |
| int CP5200\_RS232\_SetSelectedAndUserVarData(int nCardID, int bSave , int nVarNum , int bAstride , int\* nWarLen , byte\* byNoData, int nSelPrg ) | |
| Description | Set selected and user variable data |
| Parameter | nCardID: Controller ID |

|  |  |
| --- | --- |
|  | bSave: Bit0:Whether to save all variables to the flash 0:No save，1:Save。  Bit1~7: Reserved,set to 0 |
| nVarNum: Variable number |
| bAstride: Whether to allow cross-variable zone setting. 0 is not permitted;  1 is permit |
|  | nVarLen：Bytes of data specified for each variable. |
| byNoData：Specified number of variables and variable data for each variable, the first byte of each variable is the variable number, followed by a specified length of variable data. |
|  |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note | Corresponds to a variable number of each variable area size of each variable region is 32 bytes. Multiple continuous variables can be linked to a variable area used,occupied area of the variable number of variables can not be used。  When does not allow cross-variable area, more than 32 bytes of data are discarded；When allow cross-variable area,calculate the length of the data area to use the number of variables.  Valid values for the variable number is 1~100。Number of variables corresponding to each variable area can store 32 bytes of data, a number of continuous variable area can be used together for a variable, the variable area occupied number of variables can not be used。  When variable values are not updated and just save the variable value to the FLASH, it can set the " nVarNum " of the value of 0, set the " bSave " to save |

CP5200\_RS232\_SetGlobalZone

|  |  |
| --- | --- |
| int CP5200\_RS232\_SetGlobalZone(int nCardID, byte byConfig , byte bySynchro , byte byZoneNum , byte \*byZoneMsg ) | |
| Description | Set global display zoneControl the internal timer |
| Parameter | nCardID: Controller ID |
| byConfig: Bit0: save the setting to FLASH or not 0 not to save，1 save。  Bit1~7: Reserved, set value 0 |
| bySynchro: Synchronous display. 0 not synchronous, 1 synchronous.  Bit1~7: Reserved |
| byZoneNum: Zone count to be set. Normal 1~8, 0 clear all zones. |
| pZoneMsg: Zone definition data. 16 bytes for each zone. See the following table for detail. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CP5200\_RS232\_PushUserVarData

|  |  |  |  |
| --- | --- | --- | --- |
| int CP5200\_RS232\_PushUserVarData(int nCardID, byte byOption byVarZoonNum , byte byVarDataLen , byte\* pVarNoAndData ) | | , | byte |
| Description | Push user variable data |  |  |
| Parameter | nCardID: Control Card ID |  |  |
|  | byOption:  Bit0:Whether to save all the variable to the FLASH  0:Not Save 1:Save  Bit1: Push direction. 0:push back 1:push forward Bit2~3: Reserved, set to 0.  Bit4~7: Push count. +1 is the push of zoon number. | | |
| byVarZoonNum: Zoon number.  Bit0~6:the zoon numbe which to be pushed:1~100  Bit7: Reserved, please set 0. | | |
| byVarDataLen: Variable data length.Sort every variable byte data in alphabet order.The total length of variable number and data is (1+n)byte. | | |
|  | pVarNoAndData: Variable No and data. The first byte is variable No, followed by a specify length data. | | |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error | | |
| Note |  | | |

CP5200\_RS232\_TimerCtrl

|  |  |
| --- | --- |
| int CP5200\_RS232\_TimerCtrl(int nCardID, byte byTimerNo , byte byCmd , byte byProp , DWORD dwValue ); | |
| Description | Set timer control |
| Parameter | nCardID: Control card ID. |

|  |  |
| --- | --- |
|  | byTimerNo: Timer no,set the Timer by byte,1 is activity Bit0: Timer 1.  Bit1: Timer 2  Bit3: Timer 3  Bit4: Timer 4  Bit5: Timer 5  Bit6: Timer 6.  Bit7: Timer 7. |
| byCmd: Action。 1： Initializtion Timer  2： Reset Timer  3： Start Timer 4： Puse Timer  Other：Reserved |
| byProp: Property. Have different meaning according to the action.  When the action is initialize the time:  Bit0: 0 Time, 1 count down  Bit1: 0 pause, 1 start immediately  Bit2~3: Reserved  Bit4~7: time count  Set to 0 when the action is other. |
|  | dwValue: Value. Have different meaning according to the action.  When the action is initialize the time:  The initialization value when count down, in seconds.  High byte previous.  Set to 0 when timing.  Set to 0 when the action is other. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data |
|  | -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

The description of all Actions and the correspondence property and value

|  |  |  |  |
| --- | --- | --- | --- |
| Action | Description | Property | Value |
| Initialize  Timer |  | Bit0: 0 count up, 1 Count down  Bit1: 0 Pause, 1 start immediately  Bit2~3: reserved  Bit4~7: step distance | High byte previous.  The initialization value of countdown, measure time by millisecond. The value reserved when time, set to 0 |
| Reset Timer |  | Bit0: 0 Use old value，1 Use new value  Bit1: 0 Pause，1 start immediately Bit2~3: reserved | High byte previous. Countdown timer: Use as a new initialization value when the property is set to use new value. Ignore when the property is set to use the old value. Count up timer: reserved, set to 0. |
| Start Timer |  | reserved, set to 0 | reserved, set to 0 |
| Pause Timer |  | reserved, set to 0 | reserved, set to 0 |
| Save the timer setting to flash |  | reserved, set to 0 | reserved, set to 0 |

CP5200\_RS232\_SetZoneAndVariable

|  |  |
| --- | --- |
| int CP5200\_RS232\_SetZoneAndVariable(int nCardID, const BYTE\* pZoneData, int nZoneLen, const BYTE\* pVariableData, int nVarLen, WORD wCtrl, WORD wReserved) | |
| Description | Set global zone and user variable |
| Parameter | nCardID: Controller ID |
| pZoneData：The global zone data. Including the zone Options, the number of zone, zone number, the zone defined. |
| nZoneLen：The global zone data length |
|  | pVariableData：Variable data, including variable options, variable data and  cross-district allows ,the length of the variable data table, the variable number and data |
| nVarLen：The variable data length |
| wCtrl：Effective control parameters play times, high byte first.  The value of 0 has been effective .  Bit15: Resvered, fill 0.  Bit0~14: Display times. |
|  | wReserved：resvered |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note | After use this conmand, the global zone to be automatic into synchronous display. |

CP5200\_RS232\_SendPureText

|  |  |
| --- | --- |
| int CP5200\_RS232\_SendPureText(int nCardID, int nWndNo, const char \*pText, COLORREF crColor, int nFontSize, int nSpeed, int nEffect, int nStayTime, int nAlignment) | |
| Description | Send pure text to special window |
| Parameter | nCardID: Controller ID |
| nWndNo: Window sequence number, valid values 0 to 7 |
| pText: Text to be send |
| crColor: Text color |
| nFontSize: font size and style，see 1.7. Font size code and font style |
|  | nSpeed: Effect speed  0～100：The fastest value of 0。 |
| nEffect: Show effect。See the "1.5" section. |
| nStayTime: Stay time in second |
| nAlignment: The level of alignment  0: left Alignment  1: center Alignment  2: right Alignment |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CP5200\_RS232\_SendMultiProtocol

|  |  |
| --- | --- |
| int CP5200\_Net\_SendMultiProtocol(int nCardID, int nItem, const BYTE \*pText, int nLength) | |
| Description | Send multi protocol data |
| Parameter | nCardID: Controller ID |
| nItem: Items of multi protocol |
| pText: Datas of multi protocol，see《 C-Power external calls communication protocol》send multi protocol data CC=0x60 Data item |
| nLength:Length of datas |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error |
|  | -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

10.3、Overview of network multi-window control API function

|  |  |  |  |
| --- | --- | --- | --- |
| No | API function name | Description | |
| 1 | CP5200\_Net\_SplitScreen | | Send split window command |
| 2 | CP5200\_Net\_SendText  CP5200\_Net\_SendText1 | | Send text to special window |
| 3 | CP5200\_Net\_SendTagText  CP5200\_Net\_SendTagText1 | | Send tag text to special window |
| 4 | CP5200\_Net\_SendPicture | | Send picture to special window |
| 5 | CP5200\_Net\_SendStatic | | Send static text to special window |
| 6 | CP5200\_Net\_SendClock | | Send clock to special window |
| 7 | CP5200\_Net\_ExitSplitScreen | | Exit split window command |
| 8 | CP5200\_Net\_SaveClearWndData | | Save or clear split window mesage |
| 9 | CP5200\_Net\_PlaySelectedPrg  CP5200\_Net\_PlaySelectedPrg1 | | Select play stored program |
| 10 | CP5200\_Net\_SetUserVarData | | Set user variable |
| 11 | CP5200\_Net\_SetSelectedAndUserVarData | | Set selected and user variable |
| 12 | CP5200\_Net\_SetGlobalZone | | Set global message |
| 13 | CP5200\_Net\_PushUserVarData | | Push and use variable |
| 14 | CP5200\_Net\_TimerCtrl | | Set timer control |
| 15 | CP5200\_RS232\_SetZoneAndVariable | | Set global zone and user variable |
| 16 | CP5200\_RS232\_SendPureText | | Send pure text to special window |

Usage:

Step 1: Initialize network parameters

Only record the network parameter initialization parameter information, not the actual network operation。

Step 2: Send split window command，

If the window has been divided and have been met requirements, this step can be dispensed with, or to send the split window command。 Step 3: Send text or picture to window。

Note: This category interface need not to consider whether the network has been connected , as long as the network parameters have been initialized.。

10.4 、 Detail of network multi-window control API function

CP5200\_Net\_SplitScreen

|  |  |
| --- | --- |
| int CP5200\_Net\_SplitScreen(int nCardID, int nScrWidth, int nScrHeight, int nWndCnt, const int \*pWndRects) | |
| Description | Send split window command |
| Parameter | nCardID: Controller ID |
| nScrWidth: the width of screen |
| nScrHeight: the height of screen |
| nWndCnt: The window number of the screen will be splitted , valid values 1~8。 |
| pWndRects: Window coordinates, each window with four integer said the  "left, up,right,down” coordinates,ave the same data structure with the  "RECT"of windows。 |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error  -9: The window number was too much |
| Note | This function sets sub-windows information and sends split-screen commad. |

CP5200\_Net\_SendText (CP5200\_Net\_SendText1)

|  |  |
| --- | --- |
| int CP5200\_Net\_SendText(int nCardID, int nWndNo, const char \*pText, COLORREF crColor, int nFontSize, int nSpeed, int nEffect, int nStayTime, int nAlignment); | |
| Description | Send text to special window |
| Parameter | nCardID: Controller ID |
| nWndNo: Window sequence number, valid values 0 to 7 |
| pText: The text will to be sent |
| crColor: Text color。 |
| nFontSize: font size and style，see 1.7. Font size code and font style，this parameter only support the font size, does not support multiple font |
| nSpeed: Effect speed  0～100：The fastest value of 0。 |
| nEffect: Show effect。See the "1.5" section. |
| nStayTime: Stay time in second |
| nAlignment: The level of alignment  0: left Alignment  1: center Alignment  2: right Alignment |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified |
|  | -7: Data validation error |
| Note | CP5200\_Net\_SendText1 is for single byte characters, ASCII and extended ASCII. |

CP5200\_Net\_SendTagText (CP5200\_Net\_SendTagText1)

|  |  |
| --- | --- |
| int CP5200\_Net\_SendTagText(int nCardID, int nWndNo, const char \*pText, COLORREF crColor, int nFontSize,b int nSpeed, int nEffect, int nStayTime, int nAlignment) | |
| Description | Send tag text to specify window |
| Parameter | nCardID: Controller ID |
| nWndNo: Window sequence number, valid values 0 to 7 |
| pText: The text which to be sent |
| crColor: Text color。 |
| nFontSize: font size and style，see 1.7. Font size code and font style，this parameter only support the font size, does not support multiple font |
| nSpeed: Effect speed  0～100：The fastest value of 0。 |
| nEffect: Show effect。See the "1.5" section. |
| nStayTime: Stay time in second |
| nAlignment: The level of alignment  0: left Alignment 1: center Alignment 2: right Alignment: |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data |
|  | -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note | CP5200\_Net\_SendTagText1 is for single byte characters, ASCII and extended ASCII. |

CP5200\_Net\_SendPicture

|  |  |
| --- | --- |
| int CP5200\_Net\_SendPicture(int nCardID, int nWndNo, int nPosX, int nPosY, int nCx, int nCy, const char \*pPictureFile, int nSpeed, int nEffect, int nStayTime, int nPictRef) | |
| Description | Send picture to special window |
| Parameter | nCardID: Controller ID |
| nWndNo: Window sequence number, valid values 0 to 7 |
| nPosX: Began to show the location of X coordinate. Relative upper-left corner the window.。 |
| nPosY: Began to show the location of Y coordinate. Relative upper-left corner the window. |
| nCx: The width of picture |
| nCy: The heigth of picture |

|  |  |
| --- | --- |
|  | pPictureFile: Path and file name of the picture file ,this is based on the value of nPictRef.  When the value of nPictRef is 0: pPictureFile is the Path and file name of the file on the computer.  When the value of nPictRef is 1: pPictureFile is the Path and file name of the GIF file on the controller card.  When the value of nPictRef is 2: pPictureFile is the Path and file name of the file on the computer.  When the value of nPictRef is 3: pPictureFile is the Path and file name of picture packages and the serial number of the picture on the controller card.  Packages name followed by is separated by a space. For example:  “images.rpk 1” |
| nSpeed: Effect speed  0～100：The fastest value of 0。 |
| nEffect: Show effect。See the "1.5" section. |
|  | nStayTime: Stay time in second |
| nPictRef: the way to send picture and meaning.  0：display the local picture that will be converted into the format of GIF to send.  1：display the gif picture that on the controller card.  2：display the local picture that will be converted into the format of simple to send.  3：display the picture in the picture packages that on the controller card.  Other values: deal with 0. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract |
|  | -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error  -9: Param “nWndNo” is wrong  -10: Image file does not exist  -11: The specified file is not available to support the image file |
| Note | Final image is converted to 256 color pictures to send, if given a true color image, there may be color changes.  Image size will be stretched or compressed to fit the size of the specified  window。 |

CP5200\_Net\_SendSimpleImageData

|  |  |
| --- | --- |
| int CP5200\_Net\_SendSimpleImageData(int nCardID, int nWndNo, int nPosX, int nPosY, const char \*pPictureFile, int nSpeed, int nEffect, int nStayTime, BYTE\* pPicData , long lPicDataLen) | |
| Description | Send simple picture to special window |
| Parameter | nCardID: Controller ID |
| nWndNo: Window sequence number, valid values 0 to 7 |
| nPosX: Began to show the location of X coordinate. Relative upper-left corner the window.。 |
| nPosY: Began to show the location of Y coordinate. Relative upper-left corner the window. |
| nSpeed: Effect speed  0～100：The fastest value of 0。 |
| nEffect: Show effect。See the "1.5" section. |
| nStayTime: Stay time in second |
| pPicData: simple picture data,see 1.11 simple picture data fomart. |
| lPicDataLen:the length of simple picture data. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error  -9: Param “nWndNo” is wrong  -10: Image file does not exist  -11: The specified file is not available to support the image file |
| Note | Final image is converted to 256 color pictures to send, if given a true color image, there may be color changes.  Image size will be stretched or compressed to fit the size of the specified  window。 |

CP5200\_Net\_SendStatic

|  |  |
| --- | --- |
| int CP5200\_Net\_SendStatic(int nCardID, int nWndNo, const char \*pText, COLORREF crColor, int nFontSize, int nAlignment, int x, int y, int cx, int cy) | |
| Description | Send static text to special window |
| Parameter | nCardID: Controller ID |
| nWndNo: Window sequence number, valid values 0 to 7 |
| pText: Text to be send |
| crColor: Text color。 |
| nFontSize: font size and style，see 1.7. Font size code and font style |
| nAlignment: The level of alignment  0: left Alignment  1: center Alignment  2: right Alignment |
|  | x: Start X of the play window |
| y: Start Y of the play window |
| cx: The width of play window |
|  | cy: The height of play window. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note | Content outside the region remain unchanged |

CP5200\_Net\_SendClock

|  |  |
| --- | --- |
| CP5200\_Net\_SendClock( int nCardID, int nWinNo , int nStayTime , int nCalendar , int nFormat , int nContent , int nFont , int nRed , int nGreen , int nBlue , LPCSTR pTxt ); | |
| Description | Send clock to special window |
| Parameter | nCardID: Controller ID |
| nWndNo: Window sequence number, valid values 0 to 7 |
| nStayTime: Stay time in second。 |
| nCalendar: Calendar 0: Gregorian calendar date and time  1: Lunar date and time  2: Chinese lunar solar terms  3: Lunar time and date + Solar Terms |

|  |  |
| --- | --- |
|  | nFormat: Format bit 0: when the system (0: 12 hour; 1: 24 hours system) bit 1: Year digit (0: 4; 1: 2) bit 2: Branch (0: single; 1: multi-line) bit 3~5: Format control, such as the November 12, 2010 Friday , according to diffenert values expressed as:  0: 2010/11/12 Friday 16:20:30  1: Fri，12/11/2010 16:20:30  2: 2010-11-12 Fri. 16:20:30  3: Friday，12 November 2010 16:20:30 4: Fri，Nov 12,2010 16:20:30  5: Friday，November 12 2010 16:20:30  6: Fri，11/12/2010 16:20:30 7: 2010/11/12，Fri.16:20:30 bit 6: show hands,marks bit 7: Transparent |
| nContent: Content  By bit to determine the content to display.  bit 7: Pointer bit 6: weeks bit 5: seconds bit 4: minute bit 3: hour bit 2: day bit 1: month bit 0: year |
| nFont: Font，Bit0~3: font size |
|  | nRed: The red color component |
| nGreen: The red green component |
| nBlue: The red blue component |
| pTxt: Text string to the end of 0x00. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data |
|  | -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CP5200\_Net\_ExitSplitScreen

|  |  |
| --- | --- |
| int CP5200\_Net\_ExitSplitScreen( int nCardID ); | |
| Description | Exit split window command |
| Parameter | nCardID: Controller ID |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| 其它说明 |  |

CP5200\_Net\_SaveClearWndData

|  |  |
| --- | --- |
| int CP5200\_Net\_SaveClearWndData( int nCardID , int nSavaOrClear ); | |
| Description | Save or clear split window mesage |
| Parameter | nCardID: Controller ID |
| nSavaOrClear：Save or clear data 0: Save data to the flash.  1: Clear data from the flash. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller |
|  | -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CP5200\_Net\_PlaySelectedPrg

|  |  |
| --- | --- |
| int CP5200\_Net\_PlaySelectedPrg(int nCardID, const WORD \*pSelected, int nSelCnt, int nOption) | |
| Description | Select play stored program |
| Parameter | nCardID: Controller ID |
| pSelected：The program number array of be selected to play |
| nSelCnt：The program count of be selected |
| nOption：Whether to save select message to the flash  0：No save  1：Save |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CP5200\_Net\_PlaySelectedPrg1

int CP5200\_Net\_PlaySelectedPrg1(int nCardID, const WORD \*pSelected, int nSelCnt, int

|  |  |
| --- | --- |
| nOption, int nScrWidth , int nScrHeight , byte byColorGray , byte nWndCnt) | |
| Description | Select play stored program |
| Parameter | nCardID: Controller ID |
| pSelected：The program number array of be selected to play |
| nSelCnt：The program count of be selected |
| nOption：Whether to save select message to the flash  0：No save  1：Save |
| nScrWidth：Screen width |
| nScrHeight：Screen height |
| byColorGray：color gray |
| nWndCnt：window count |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CP5200\_Net\_SetUserVarData

|  |  |
| --- | --- |
| int CP5200\_Net\_SetUserVarData(int nCardID, int bSave , int nVarNum , int bAstride , int\* nWarLen , byte\* byNoData ); | |
| Description | Set user variable |
| Parameter | nCardID: Controller ID |
| bSave: Bit0:Whether to save all variables to the flash 0:No save，1:Save。  Bit1~7: Reserved,set to 0 |
| nVarNum: Variable number |
|  | bAstride: Whether to allow cross-variable zone setting. 0 is not permitted;  1 is permit |
| nVarLen：Bytes of data specified for each variable. |
| byNoData：Specified number of variables and variable data for each variable, the first byte of each variable is the variable number, followed by a specified length of variable data. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note | Corresponds to a variable number of each variable area size of each variable region is 32 bytes. Multiple continuous variables can be linked to a variable area used,occupied area of the variable number of variables can not be used。  When does not allow cross-variable area, more than 32 bytes of data are discarded；When allow cross-variable area,calculate the length of the data area to use the number of variables.  Valid values for the variable number is 1~100。Number of variables corresponding to each variable area can store 32 bytes of data, a number of continuous variable area can be used together for a variable, the variable area occupied number of variables can not be used。  When variable values are not updated and just save the variable value to the FLASH, it can set the " nVarNum " of the value of 0, set the " bSave " to save |

CP5200\_Net\_SetSelectedAndUserVarData

|  |  |
| --- | --- |
| int CP5200\_Net\_SetSelectedAndUserVarData(int nCardID, int bSave , int nVarNum , int bAstride , int\* nWarLen , byte\* byNoData, int nSelPrg ) | |
| Description | Set selected and user variable data |
| Parameter | nCardID: Controller ID |

|  |  |
| --- | --- |
|  | bSave: Bit0:Whether to save all variables to the flash 0:No save，1:Save。  Bit1~7: Reserved,set to 0 |
| nVarNum: Variable number |
| bAstride: Whether to allow cross-variable zone setting. 0 is not permitted;  1 is permit |
|  | nVarLen：Bytes of data specified for each variable. |
| byNoData：Specified number of variables and variable data for each variable, the first byte of each variable is the variable number, followed by a specified length of variable data. |
|  |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note | Corresponds to a variable number of each variable area size of each variable region is 32 bytes. Multiple continuous variables can be linked to a variable area used,occupied area of the variable number of variables can not be used。  When does not allow cross-variable area, more than 32 bytes of data are discarded；When allow cross-variable area,calculate the length of the data area to use the number of variables.  Valid values for the variable number is 1~100。Number of variables corresponding to each variable area can store 32 bytes of data, a number of continuous variable area can be used together for a variable, the variable area occupied number of variables can not be used。  When variable values are not updated and just save the variable value to the FLASH, it can set the " nVarNum " of the value of 0, set the " bSave " to save |

CP5200\_Net\_SetGlobalZone

|  |  |
| --- | --- |
| int CP5200\_Net\_SetGlobalZone(int nCardID, byte byConfig , byte bySynchro , byte byZoneNum , byte \*byZoneMsg ) | |
| Description | Set global display message |
| Parameter | nCardID: contrl card ID |
| byConfig:  Bit0: Whether to save to FLASH  0:Not save, 1:Save  Bit1~7:Reserved, set to 0 |
| bySynchro: Synchronization。  Bit0: Whether to synchronization, 0 Not synchronous，1 synchronous。  Bit1~7: Reserved |
| byZoneNum: Zone number.The golobal display zone number which to be set.Cancel all the zone when zone number is 0. |
| byZoneMsg: zone message.The specify message of global display zone. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CP5200\_Net\_PushUserVarData

|  |  |
| --- | --- |
| int CP5200\_Net\_PushUserVarData(int nCardID, byte byOption , byte byVarZoonNum , byte byVarDataLen , byte\* pVarNoAndData ) | |
| Description | Push user variable data |
| Parameter | nCardID: Control Card ID |
|  | byOption:  Bit0:Whether to save all the variable to the FLASH  0:Not Save 1:Save  Bit1: Push direction. 0:push back 1:push forward Bit2~3: Reserved, set to 0.  Bit4~7: Push count. +1 is the push of zoon number. |
| byVarZoonNum: Zoon number.  Bit0~6:the zoon numbe which to be pushed:1~100  Bit7: Reserved, please set 0. |
| byVarDataLen: Variable data length.Sort every variable byte data in alphabet order.The total length of variable number and data is (1+n)byte. |
|  | pVarNoAndData: Variable No and data. The first byte is variable No, followed by a specify length data. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CP5200\_Net\_TimerCtrl

|  |  |
| --- | --- |
| int CP5200\_Net\_TimerCtrl(int nCardID, byte byTimerNo , byte byCmd , byte byProp ,  DWORD dwValue); | |
| Description | Set timer control |
| Parameter | nCardID: Control card ID. |

|  |  |
| --- | --- |
|  | byTimerNo: Timer no,set the Timer by byte,1 is activity Bit0: Timer 1.  Bit1: Timer 2  Bit3: Timer 3  Bit4: Timer 4  Bit5: Timer 5  Bit6: Timer 6.  Bit7: Timer 7. |
| byCmd: Action。 1： Initializtion Timer  2： Reset Timer  3： Start Timer 4： Puse Timer  Other：Reserved |
| byProp: Property. Have different meaning according to the action.  When the action is initialize the time:  Bit0: 0 Time, 1 count down  Bit1: 0 pause, 1 start immediately  Bit2~3: Reserved  Bit4~7: time count  Set to 0 when the action is other. |
|  | dwValue: Value. Have different meaning according to the action.  When the action is initialize the time:  The initialization value when count down, in seconds.  High byte previous.  Set to 0 when timing.  Set to 0 when the action is other. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data |
|  | -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CP5200\_Net\_SetZoneAndVariable

|  |  |
| --- | --- |
| int CP5200\_Net\_SetZoneAndVariable(int nCardID, const BYTE\* pZoneData, int nZoneLen, const  BYTE\* pVariableData, int nVarLen, WORD wCtrl, WORD wReserved) | |
| Description | Set global zone and user variable |
| Parameter | nCardID: Controller ID |
| pZoneData：The global zone data. Including the zone Options, the number of zone, zone number, the zone defined. |
| nZoneLen：The global zone data length |
| pVariableData：Variable data, including variable options, variable data and  cross-district allows ,the length of the variable data table, the variable number and data |
| nVarLen：The variable data length |
| wCtrl：Effective control parameters play times, high byte first.  The value of 0 has been effective .  Bit15: Resvered, fill 0.  Bit0~14: Display times. |
| wReserved：resvered |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note | After use this conmand, the global zone to be automatic into synchronous |
|  | display. |

CP5200\_Net\_SendPureText

|  |  |
| --- | --- |
| int CP5200\_Net\_SendPureText(int nCardID, int nWndNo, const char \*pText, COLORREF crColor, int nFontSize, int nSpeed, int nEffect, int nStayTime, int nAlignment) | |
| Description | Send pure text to special window |
| Parameter | nCardID: Controller ID |
| nWndNo: Window sequence number, valid values 0 to 7 |
| pText: Text to be send |
| crColor: Text color |
| nFontSize: font size and style，see 1.7. Font size code and font style |
| nSpeed: Effect speed  0～100：The fastest value of 0。 |
| nEffect: Show effect。See the "1.5" section. |
| nStayTime: Stay time in second |
| nAlignment: The level of alignment  0: left Alignment  1: center Alignment  2: right Alignment |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CP5200\_Net\_SendMultiProtocol

|  |  |
| --- | --- |
| int CP5200\_Net\_SendMultiProtocol(int nCardID, int nItem, const BYTE \*pText, int nLength) | |
| Description | Send multi protocol data |
| Parameter | nCardID: Controller ID |
| nItem: Items of multi protocol |
| pText: Datas of multi protocol，see《 C-Power external calls communication protocol》send multi protocol data CC=0x60 Data item |
| nLength:Length of datas |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

# Program template API function

11.1 、 Overview of RS232 program template API

function

|  |  |  |
| --- | --- | --- |
| No. | Function name | Description |
| 1 | CPowerBox\_RS232\_SetProgramTemplate  CPowerBox\_RS232\_SetProgramTemplate1 | Set program template command |
| 2 | CPowerBox\_RS232\_InOutProgramTemplate | In or out program template command |
| 3 | CPowerBox\_RS232\_QueryProgramTemplate | Query program template command |
|  | CPowerBox\_RS232\_QueryProgramTemplate1 |  |
| 4 | CPowerBox\_RS232\_DeleteProgram | Delete program command |
| 5 | CPowerBox\_RS232\_SendText | Send text to special window |
| 6 | CPowerBox\_RS232\_SendPicture | Send picture to special window |
| 7 | CPowerBox\_RS232\_SendClockOrTemperature | Send clock and temperature to special window |
| 8 | CPowerBox\_RS232\_SetAloneProgram | Set alone program |
| 9 | CPowerBox\_RS232\_QueryProgram | Query program information |
| 10 | CPowerBox\_RS232\_SetProgramProperty | Set program property |
| 11 | CPowerBox\_RS232\_SetSchedule | Set play schedule |
| 12 | CPowerBox\_RS232\_DeleteSchedule | Delete play schedule |
| 13 | CPowerBox\_RS232\_GetSchedule | Get play schedule |

11.2、Detail of RS232 program template API function

CPowerBox\_RS232\_SetProgramTemplate

|  |  |
| --- | --- |
| int CPowerBox\_RS232\_SetProgramTemplate(int nCardID, byte byColor ,USHORT nWidth , USHORT nHeight , byte nWndNum , byte \*byDefParam , byte\* pWndParam) | |
| Description | Set program template |
| Parameter | nCardID: Control card ID. |
| byColor: Bit0: Red mark  Bit1: Green mark  Bit2: Blue mark  Bit3: Reserved  Bit4～6: Gray level  0: 2 level gray，7: 256 level gray  Bit7: Reserved |
| nWidth: The width of the screen，high byte previous |
| nHeight: The height of the screen，high byte previous |
| nWndNum: The display window number,the maximum number is 10 |
|  | byDefParam: Default parameter。  Byte0~1: Stay time in second. High byte previous.  Byte2: Speed。The smaller the faster.  Byte3: Font size. See “Font size code”  Byte4: Font color. See “Font color code”  Byte5: Show effect See”Show effect code”  Byte6: Picture type. See”Picture type code”  Byte7: Clock Format. See “Clock format and content”  Byte8: Clock content. See “Clock format and content” |
| pWndParam: Window parameter. Each window has a 16 bytes length parameter. The total length of the data is: the number of the window\*16. You can see the detail at “appendix:1 window position and property” |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_RS232\_SetProgramTemplate1

|  |  |
| --- | --- |
| int CPowerBox\_RS232\_SetProgramTemplate1(int nCardID, BYTE byColor ,USHORT nWidth , USHORT nHeight , BYTE nWndNum , BYTE byOption, BYTE\* pDefParam , BYTE\* pWndParam) | |
| Description | Set program template |
| Parameter | nCardID: Control card ID. |
| byColor: Bit0: Red mark  Bit1: Green mark  Bit2: Blue mark  Bit3: Reserved  Bit4～6: Gray level  0: 2 level gray，7: 256 level gray  Bit7: Reserved |
| nWidth: The width of the screen，high byte previous |
|  | nHeight: The height of the screen，high byte previous |
| nWndNum: The display window number,the maximum number is 10 |
| byOption:  Bit0: Forced into the program template run  Bit1: Save the template position. 0: user disk, 1: system disk.  If the template is saved to the system tray, the original template of the user tray is cleared; if the template is saved to the user's disk, the original template of the system disk is cleared。  Bit2~7: Reserved |
|  | byDefParam: Default parameter。  Byte0~1: Stay time in second. High byte previous.  Byte2: Speed。The smaller the faster.  Byte3: Font size. See “Font size code”  Byte4: Font color. See “Font color code”  Byte5: Show effect See”Show effect code”  Byte6: Picture type. See”Picture type code”  Byte7: Clock Format. See “Clock format and content”  Byte8: Clock content. See “Clock format and content” |
| pWndParam: Window parameter. Each window has a 16 bytes length parameter. The total length of the data is: the number of the window\*16. You can see the detail at “appendix:1 window position and property” |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_RS232\_InOutProgramTemplate

|  |  |
| --- | --- |
| int CPowerBox\_RS232\_InOutProgramTemplate( int nCardID,byte byInOrOut ) | |
| Description | Set in or out program template |
| Parameter | nCardID: Control card ID |
|  | byInOrOut: In or Out  1: In program template style.  0: Out program template style. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_RS232\_QueryProgramTemplate

|  |  |
| --- | --- |
| int CPowerBox\_RS232\_QueryProgramTemplate(int nCardID , byte\* pState ); | |
| Description | Set query program template |
| Parameter | nCardID: Card ID |
| pState: template status, 1 is template mode and 0 is not template mode |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_RS232\_QueryProgramTemplate1

|  |  |
| --- | --- |
| int CPowerBox\_RS232\_QueryProgramTemplate1(int nCardID , byte byFlag , BYTE\* pStateBuf , int nBufSize ) | |
| Description | Set query program template |
| Parameter | nCardID: Card ID |
| byFlag:  Bit0: Whether to query program template status parameter  Bit1:Whether to return the template definition color gray, screen size information  Bit2~7: Reserved |
| pStateBuf: The results data buffer |
| nBufSize: The size of results data buffer |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |
|  |  |

“pStateBuf” have the following meanings:：

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x83 | 1 | Describe the package is the return data of query program template status parameter. |
| Options |  | 1 | The same value with send value of “Options”. |
| Template mode |  | 1 | 0: Not program template  1: program template |
| Template status |  | 1 | Bit0~1: template availability  0: the template is not available  1: the template can be used others: Reserved |
|  |  |  | Bit2~7: Reserved |
| Color gray |  | 1 | Color and gray。  Same with define“set program template” |
| Screen width |  | 2 | High byte first |
| Screen height |  | 2 | High byte first |
| Window count |  | 1 | Play window count。  Supports up to 10 play windows |

CPowerBox\_RS232\_DeleteProgram

|  |  |
| --- | --- |
| int CPowerBox\_RS232\_DeleteProgram( int nCardID,byte byConfig , byte byProNum , byte\* pDelPro ); | |
| Description | Delete program |
| Parameter | nCardID: Control card ID. |
| byConfig: Bit0: The range of the delete program  0：Delete all the program  1：Delete the specify program  Other: Reserved |
| byProNum: Program number. Do not need this item when delete all the program. |
| pDelPro: The list of the program need to be delete. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_RS232\_SendText

|  |  |
| --- | --- |
| int CPowerBox\_RS232\_SendText( int nCardID, DWORD dwAppendCode , byte byProNo , byte byWndNo , byte byProp , byte \*byShowFormat , char\* pText); | |
| Description | Send text to the specify window |
| Parameter | nCardID: Card ID |
| dwAppendCode: The user’s append code, high byte previous. |
| byProNo: Program No.,Valid value:1~255 |
| byWndNo: Window No. Valid value:1～10 , Invalid when out of program template definition. |
| byProp: Property，Bit0~3: Text type 0：Common Text  Bit4: Display format. 0: default format 1:specify format  Bit5~7: Reserved |
| byShowFormat: Show format. Do not need this item when the property’s display format is 0.  Byte0~1: Stay time,High byte previous.  Byte2: Speed. The smaller the faster.  Byte3: Font size. See “Font size code”  Byte4: Font color. See “Font color code”  Byte5: Show effect See”Show effect code”  Byte6: Reserved  Byte7: Reserved |
| pText: Text data, end with ‘0x00’ |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_RS232\_SendPicture

|  |  |
| --- | --- |
| int CPowerBox\_RS232\_SendPicture( int nCardID, DWORD dwAppendCode , byte byProNo , byte byWndNo , byte byPicType , byte \*byShowFormat , byte\* pPicData , long lPicDataLen); | |
| Description | Send picture to the specify picture |
| Parameter | nCardID: Control card ID |
| dwAppendCode: The user’s append code, high byte previous. |
| byProNo: Program No.,Valid value:1~255 |
| byWndNo: Window No. Valid value:1～10 , Invalid when out of program template definition. |
| byPicType: Picture type. Bit0~3: Picture type  1: Data of GIF picture file which include the information of the picture’s width and height so on.  2: The stored GIF filename in the contrl card.  4. Simple picture data, Check the format information at ”Simple Picture data format”  Bit4: Show format. 0 default format,1 specify format  Bit5~7: Reserved |
| byShowFormat: Show format.  Do not need this item when the property’s display format is 0.  Byte0~1: Stay time, High byte previous.  Byte2: Speed. The smaller the faster.  Byte3: Show effect See”Show effect code”  Byte4: Picture style(zoom、tile), see “Picture style code”  Byte5: Reserved  Byte6: Reserved  Byte7: Reserved |
| pPicData: Picture data. |
| lPicDataLen:Picture data length. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract |
|  | -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_RS232\_SendClockOrTemperature

|  |  |
| --- | --- |
| CPowerBox\_RS232\_SendClockOrTemperature( int nCardID,DWORD dwAppendCode , BYTE byProNo , BYTE byWndNo , BYTE byProgramType , UINT nPropLen , BYTE\* pProgramProp ,byte\* pBuf , int nBufSize ) | |
| Description | Send clock and temperature to special window |
| Parameter | nCardID: Controller ID |
| dwAppendCode: The user’s append code, high byte first. |
| byProNo: Program No.,Valid value:1~255 |
| byWndNo: Window No. Valid value:1～10 , Invalid when out of program template definition. |
| byProgramType: Program type Bit0~3: Type  2：Clock ; 3：Temperature Bit4: Display format.  0: default format 1:specify format  Bit5~7: Reserved, fill in 0 |
| nPropLen: Property length |
| pProgramProp：Program property  The meaning of the attribute data according to different types  Type = 2 , see Clock/Calendar type proprtey  Type = 3 , see Temperature and Humidity type proprtey |
| pBuf: The results data buffer |
| nBufSize：The size of results data buffer |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port |
|  | -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

“pBuf” have the following meanings:

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x87 | 1 | Describe the package is the return data which to show clock/temperature in the specified window of the specified program |
| Append code |  | 4 | The user’s append code, high byte previous. |
| Program No |  | 1 | The same value with send value “Program no”.  Valid value:1~100 |
| Window No |  | 1 | The same value with send value “Window no”. Valid value:1~10,Invalid when out of program template definition. |
| Packet loss number |  | 1 | The number of packets that have not yet received. Sends the first packet loss number is the total number of packets minus one. |
| The packet number of the packet loss |  | Variable-length | Packet loss packet number. Always in accordance with small to large; the first packet packet number is 0. Each package a byte. |

CPowerBox\_RS232\_SetAloneProgram

|  |  |
| --- | --- |
| int CPowerBox\_RS232\_SetAloneProgram(int nCardID,DWORD dwAppendCode , BYTE byProgramNo ,  BYTE byWindowCnt ,BYTE\* pWndParam, BYTE\* pWndData) | |
| Description | Set alone program |
| Parameter | nCardID: Controller ID |
| dwAppendCode: The user’s append code, high byte first. |
| byProNo: Program No.,Valid value:1~255 |
| byWindowCnt: Window count. Valid value:1～10 |
|  | pWndParam：windows parameter  Every window information table has a 22 bytes length parameter. The 1~16 bytes are window position and property, You can see the detail at 1.13. Window position and property; The 17~19 bytes are window data offset; The 20~22 bytes are window data length. High byte first.  If no data ,then window data offset and window data length all are 0.  The total length of the data is: the number of the window\*22. |
| pWndData：Window play data:“Text”、“Picture”…  Byte 1：Data Type(1 Text；4 Picture)  Byte 2：Data Format（Like “Text type” in command 0x85 and “Picture type” in command 0x86）  Byte 3：Text data or picture data。 |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_RS232\_QueryProgram

|  |  |
| --- | --- |
| Int CPowerBox\_RS232\_QueryProgram( int nCardID ,byte byFlag , byte\* pParam , BYTE\* pBuf , int nBufSize ); | |
| Description | Query program information |
| Parameter | nCardID:Controller ID |
| byFlag:Special which program info will to be query 1: Query valid programs count and program number 2: Query specifies program information.  Other: Reserved |
|  | pParam:  If “byFlag” is 1：byte1~5，resvered，fill 0  If “byFlag” is 2：：byte1，program number；byte2~5，resvered，fill 0 |
| pBuf: The results data buffer |
| nBufSize：The size of results data buffer |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

“pBuf” have the following meanings：

 Query“valid program count and program number”

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x89 | 1 | Describe the package is the return data packet of query program info |
| Info flag |  | 1 | Same with send value ”info flag” |
| parameters |  | 5 | Same with send value ”parameters” |
| Valid program count |  | 1 | Valid program count |
| Valid program number |  | Variable-length | Each byte identifies an effective program。  Valid value 1～100。 |

* The meaning of "return value" in the return packet:

0x01 Controller not running in program template mode

0x10 Unknown info flag

 Query specifies program information

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x89 | 1 | Describe the package is the return data packet of query program info |
| Info flag |  | 1 | Same with send value ”info flag” |
| parameters |  | 5 | Same with send value ”parameters” |
| Information |  | 1 | Now only return one information |
| count |  |  |  |
| Program number |  | 1 | Program number |
| User append  code |  | 4 | User append code |

* The meaning of "return value" in the return packet:

0x01 Controller not running in program template mode

0x10 Unknown info flag

0x11 Invalid programs

0x12 Can’t get program information

CPowerBox\_RS232\_SetProgramProperty

|  |  |
| --- | --- |
| int CPowerBox\_RS232\_SetProgramProperty( int nCardID, byte byOption , byte byProgramCnt , byte\* pPrograms , byte byPropertyID1 , byte byPropertyID2 , byte byProgramLevel , USHORT nLoopCnt , USHORT nTime , byte\* pDuetime , byte\* pTimeInterval); | |
| Description | Set program property |
| Parameter | nCardID: The control card ID |
| byOption:  Bit0: Set the range of the program property  0: All programes  1: Specify program  Other: Reserved |
| byProgramCnt:The count of the program |
| pPrograms: The list of the programes |
| ByPropertyID1：Property ID 1, marked which property you want to set by  byte, set 0 if the data not exist.  Bit0: The level of the program.  Bit1: The cycle count.  Bit2: Valid time. How long will the program be valid from now on.  Bit3: Interval time  Bit4~7: Reserved |
|  | ByPropertyID2: Property ID 2。Bit0~4: valid time. >0 the count of the valid time.<=4  Bit5~7: Reserved |
| byProgramLevel: The program level. 1～3 level, The high level of the program is priority. |
| nLoopCnt: Loop count, High byte previous(big-endian).  0: Do not play the program, use to shield program temporarily.  1~255: The loop count of the program. |
|  | nTime: Valid time. High byte previous (big-endian). In minute.  0: Not limit play time  >0: Specify play time in minute. |
| pDuetime: time limit |
| pTimeInterval:The interval time. The start tag “Hour/Minute/Second”and the end tag “Hour/Minute/Second” both represent by one byte. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_RS232\_SetSchedule

|  |  |
| --- | --- |
| int CPowerBox\_RS232\_SetSchedule(int nCardID, DWORD dwAppendCode, BYTE byScheduleNo, const BYTE\* pProperty, const BYTE\* pBoxes, BYTE byBoxCnt) | |
| Description | Set play schedule |
| Parameter | nCardID: Controller ID |
| dwAppendCode: The user’s append code, high byte first. |
|  | byScheduleNo: Schedule number，Valid value 1~100。Total support 100 plans, For each plan No, the new data cover the old data |
| pProperty: play property，total 14 bytes： byte 0：Format and level： Bit0~3: Data format，fill in 0x01  Bit4~7: Indicates the priority level. The priority level the greater the value, the more priority to play, 0 is the lowest priority.。。  byte 1：Weekday：Bit0~6: 7-bit logo Sunday to Saturday byte 2~4： Begin date ， 3 bytes: Byte1:Year,Valid value0~99,means  2000~2999; Byte2:Month ;Byte3:Day  byte 5~7：End date，3 bytes: Byte1:Year,Valid value0~99,means 2000~2999;  Byte2:Month ;Byte3:Day  byte 8~10：Begin time, 3 bytes:Byte1:Hour；Byte2:Minute；Byte3:Second byte 11~13：End time, 3 bytes:Byte1:Hour；Byte2:Minute；Byte3:Second |
| pBoxes: program number , each byte represents a program. Numbered in  ascending order, do not repeat..... |
|  | byBoxCnt:program number count, Valid value:1～100, |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_RS232\_DeleteSchedule

|  |  |
| --- | --- |
| int CPowerBox\_RS232\_DeleteSchedule(int nCardID, DWORD dwAppendCode, const BYTE\* pSchs,  BYTE bySchCnt) | |
| Description | Delete play schedule |
| Parameter | nCardID: Controller ID |
| dwAppendCode: The user’s append code, high byte first. |
| pSchs: schedule number, Valid value 1~100。Each byte represents a play schedule。  When delete all play schedule, the length of this data is one , value is 0xff. |
| bySchCnt: The number of play schedule will to be delete。0 means delete all play plans. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_RS232\_GetSchedule

|  |  |
| --- | --- |
| int CPowerBox\_RS232\_GetSchedule(int nCardID, DWORD dwAppendCode, BYTE byType, BYTE byScheduleNo , byte\* pBuf , int nBufSize ) | |
| Description | Get play schedule |
| Parameter | nCardID: Controller ID |
| dwAppendCode: The user’s append code, high byte first. |
| byType：0: Query all valid play plan.  1: Query specified play plan no  Other: Reserved |
| byScheduleNo: Valid value:1~100。When query type is 0，this data fill in 0。 |
| pBuf: The results data buffer |
| nBufSize：The size of results data buffer |
| Return | 0: Success  -1: Can not generate command data |
|  | -2: The command data package error  -3: Can not open serial port  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

“pBuf” have the following meanings:

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x8d | 1 | Describe the package is the return data which to query play plan |
| Append code |  | 4 | The user’s append code, high byte previous. |
| Query type |  | 1 | 0: Query all valid play plan.  1: Query specified play plan no  Other: Reserved |
| Count /Number |  | 1 | When query type is 0，this value is valid play schedule count  When query type is 1，this value is play schedule number. |
| Play schedule number table/ play schedule content |  | Variable-length | When query type is 0，this value is valid play schedule number table  When query type is 1，this value is play schedule content. Data format like command  0x8B. |

You must deal with the return data according to the different query type.

The meaning of "return value" in the return packet:

0x01 program template is invalid 0x11 Don’t support the query type.

0x12 Invalid play plan no.

0x80 currently is not program template way

11.3 、 Overview of Network program template API

function

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Function name | Description | |
| 1 | CPowerBox\_Net\_SetProgramTemplate | Set program template | |
| 2 | CPowerBox\_Net\_InOutProgramTemplate | Set in or out program template | |
| 3 | CPowerBox\_Net\_QueryProgramTemplate | Query program template | |
| 4 | CPowerBox\_Net\_DeleteProgram | Delete program | |
| 5 | CPowerBox\_Net\_SendText | Send text to special window | |
| 6 | CPowerBox\_Net\_SendPicture | Send picture to special window | |
| 7 | CPowerBox\_Net\_SendClockOrTemperature | | Send clock and temperature to special window |
| 8 | CPowerBox\_Net\_SetAloneProgram | | Set alone program |
| 9 | CPowerBox\_Net\_QueryProgram | | Query program information |
| 10 | CPowerBox\_Net\_SetProgramProperty | | Set program property |
| 11 | CPowerBox\_Net\_SetSchedule | | Set play schedule |
| 12 | CPowerBox\_Net\_DeleteSchedule | | Delete play schedule |
| 13 | CPowerBox\_Net\_GetSchedule | | Get play schedule |

11.4、Detail of Network program template API function

CPowerBox\_Net\_SetProgramTemplate

|  |  |
| --- | --- |
| int CPowerBox\_Net\_SetProgramTemplate(int nCardID, byte byColor ,USHORT nWidth , USHORT nHeight , byte nWndNum , byte \*byDefParam , byte\* pWndParam) | |
| Description | Set program template |
| Parameter | nCardID: Control card ID. |
| byColor: Bit0: Red mark  Bit1: Green mark  Bit2: Blue mark  Bit3: Reserved  Bit4～6: Gray level  0: 2 level gray，7: 256 level gray  Bit7: Reserved |
| nWidth: The width of the screen，high byte previous |
| nHeight: The height of the screen，high byte previous |
| nWndNum: The display window number,the maximum number is 10 |
|  | byDefParam: Default parameter。  Byte0~1: Stay time in second. High byte previous.  Byte2: Speed。The smaller the faster.  Byte3: Font size. See “Font size code”  Byte4: Font color. See “Font color code”  Byte5: Show effect See”Show effect code”  Byte6: Picture type. See”Picture type code”  Byte7: Clock Format. See “Clock format and content”  Byte8: Clock content. See “Clock format and content” |
| pWndParam: Window parameter. Each window has a 16 bytes length parameter. The total length of the data is: the number of the window\*16. You can see the detail at “appendix:1 window position and property” |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_Net\_SetProgramTemplate1

|  |  |
| --- | --- |
| int CPowerBox\_Net\_SetProgramTemplate1(int nCardID, BYTE byColor ,USHORT nWidth , USHORT nHeight , BYTE nWndNum , BYTE byOption, BYTE\* pDefParam , BYTE\* pWndParam) | |
| Description | Set program template |
| Parameter | nCardID: Control card ID. |
| byColor: Bit0: Red mark  Bit1: Green mark  Bit2: Blue mark  Bit3: Reserved  Bit4～6: Gray level  0: 2 level gray，7: 256 level gray  Bit7: Reserved |
| nWidth: The width of the screen，high byte previous |
|  | nHeight: The height of the screen，high byte previous |
| nWndNum: The display window number,the maximum number is 10 |
| byOption:  Bit0: Forced into the program template run  Bit1: Save the template position. 0: user disk, 1: system disk.  If the template is saved to the system tray, the original template of the user tray is cleared; if the template is saved to the user's disk, the original template of the system disk is cleared。  Bit2~7: Reserved |
|  | byDefParam: Default parameter。  Byte0~1: Stay time in second. High byte previous.  Byte2: Speed。The smaller the faster.  Byte3: Font size. See “Font size code”  Byte4: Font color. See “Font color code”  Byte5: Show effect See”Show effect code”  Byte6: Picture type. See”Picture type code”  Byte7: Clock Format. See “Clock format and content”  Byte8: Clock content. See “Clock format and content” |
| pWndParam: Window parameter. Each window has a 16 bytes length parameter. The total length of the data is: the number of the window\*16. You can see the detail at “appendix:1 window position and property” |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_Net\_InOutProgramTemplate

|  |  |
| --- | --- |
| int CPowerBox\_Net\_InOutProgramTemplate( int nCardID,byte byInOrOut ) | |
| Description | Set in or out program template |
| Parameter | nCardID: Control card ID |
|  | byInOrOut: In or Out  1: In program template style.  0: Out program template style. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_Net\_QueryProgramTemplate

|  |  |
| --- | --- |
| int CPowerBox\_Net\_QueryProgramTemplate(int nCardID , byte\* pState ); | |
| Description | Set query program template |
| Parameter | nCardID: Card ID |
| pState: Reserved |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_Net\_QueryProgramTemplate1

|  |  |
| --- | --- |
| int CPowerBox\_Net\_QueryProgramTemplate1(int nCardID , byte byFlag , BYTE\* pStateBuf , int nBufSize ) | |
| Description | Set query program template |
| Parameter | nCardID: Card ID |
| byFlag:  Bit0: Whether to query program template status parameter  Bit1:Whether to return the template definition color gray, screen size information  Bit2~7: Reserved |
| pStateBuf: The results data buffer |
| nBufSize: The size of results data buffer |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |
|  |  |

“pStateBuf” have the following meanings:：

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x83 | 1 | Describe the package is the return data of query program template status parameter. |
| Options |  | 1 | The same value with send value of “Options”. |
| Template mode |  | 1 | 0: Not program template  1: program template |
| Template status |  | 1 | Bit0~1: template availability  0: the template is not available  1: the template can be used others: Reserved |
|  |  |  | Bit2~7: Reserved |
| Color gray |  | 1 | Color and gray。  Same with define“set program template” |
| Screen width |  | 2 | High byte first |
| Screen height |  | 2 | High byte first |
| Window count |  | 1 | Play window count。  Supports up to 10 play windows |

CPowerBox\_Net\_DeleteProgram

|  |  |
| --- | --- |
| int CPowerBox\_Net\_DeleteProgram( int nCardID,byte byConfig , byte byProNum , byte\* pDelPro ); | |
| Description | Delete program |
| Parameter | nCardID: Control card ID. |
| byConfig: Bit0: The range of the delete program  0：Delete all the program  1：Delete the specify program  Other: Reserved |
| byProNum: Program number. Do not need this item when delete all the program. |
| pDelPro: The list of the program need to be delete. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_Net\_SendText

|  |  |
| --- | --- |
| int CPowerBox\_Net\_SendText( int nCardID, DWORD dwAppendCode , byte byProNo , byte byWndNo , byte byProp , byte \*byShowFormat, char\* pText); | |
| Description | Send text to the specify window |
| Parameter | nCardID: Card ID |
| dwAppendCode: The user’s append code, high byte previous. |
| byProNo: Program No.,Valid value:1~255 |
| byWndNo: Window No. Valid value:1～10 , Invalid when out of program template definition. |
| byProp: Property，Bit0~3: Text type 0：Common Text  Bit4: Display format. 0: default format 1:specify format  Bit5~7: Reserved |
| byShowFormat: Show format. Do not need this item when the property’s display format is 0.  Byte0~1: Stay time,High byte previous.  Byte2: Speed. The smaller the faster.  Byte3: Font size. See “Font size code”  Byte4: Font color. See “Font color code”  Byte5: Show effect See”Show effect code”  Byte6: Reserved  Byte7: Reserved |
| pText: Text data, end with ‘0x00’ |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_Net\_SendPicture

|  |  |
| --- | --- |
| int CPowerBox\_Net\_SendPicture( int nCardID, DWORD dwAppendCode , byte byProNo , byte byWndNo , byte byPicType , byte \*byShowFormat , byte\* pPicData , long lPicDataLen); | |
| Description | Send picture to the specify window |
| Parameter | nCardID: Control card ID |
| dwAppendCode: The user’s append code, high byte previous. |
| byProNo: Program No.,Valid value:1~255 |
| byWndNo: Window No. Valid value:1～10 , Invalid when out of program template definition. |
| byPicType: Picture type. Bit0~3: Picture type  1: Data of GIF picture file which include the information of the picture’s width and height so on.  2: The stored GIF filename in the contrl card.  4. Simple picture data, Check the format information at ”Simple Picture data format”  Bit4: Show format. 0 default format,1 specify format  Bit5~7: Reserved |
| byShowFormat: Show format.  Do not need this item when the property’s display format is 0.  Byte0~1: Stay time, High byte previous.  Byte2: Speed. The smaller the faster.  Byte3: Show effect See”Show effect code”  Byte4: Picture style(zoom、tile), see “Picture style code”  Byte5: Reserved  Byte6: Reserved  Byte7: Reserved |
| pPicData: Picture data. |
| lPicDataLen:Picture data length. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract |
|  | -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_Net\_SendClockOrTemperature

|  |  |
| --- | --- |
| CPowerBox\_Net\_SendClockOrTemperature( int nCardID,DWORD dwAppendCode , BYTE byProNo , BYTE byWndNo , BYTE byProgramType , UINT nPropLen , BYTE\* pProgramProp ,byte\* pBuf , int nBufSize ) | |
| Description | Send clock and temperature to special window |
| Parameter | nCardID: Controller ID |
| dwAppendCode: The user’s append code, high byte first. |
| byProNo: Program No.,Valid value:1~255 |
| byWndNo: Window No. Valid value:1～10 , Invalid when out of program template definition. |
| byProgramType: Program type Bit0~3: Type  2：Clock ; 3：Temperature Bit4: Display format.  0: default format 1:specify format  Bit5~7: Reserved, fill in 0 |
| nPropLen: Property length |
| pProgramProp：Program property  The meaning of the attribute data according to different types  Type = 2 , see Clock/Calendar type proprtey  Type = 3 , see Temperature and Humidity type proprtey |
| pBuf: The results data buffer |
| nBufSize：The size of results data buffer |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error |
|  | -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

“pBuf” have the following meanings:

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x87 | 1 | Describe the package is the return data which to show clock/temperature in the specified window of the specified program |
| Append code |  | 4 | The user’s append code, high byte previous. |
| Program No |  | 1 | The same value with send value “Program no”.  Valid value:1~100 |
| Window No |  | 1 | The same value with send value “Window no”. Valid value:1~10,Invalid when out of program template definition. |
| Packet loss number |  | 1 | The number of packets that have not yet received. Sends the first packet loss number is the total number of packets minus one. |
| The packet number of the packet loss |  | Variable-length | Packet loss packet number. Always in accordance with small to large; the first packet packet number is 0. Each package a byte. |

CPowerBox\_Net\_SetAloneProgram

|  |  |
| --- | --- |
| int CPowerBox\_Net\_SetAloneProgram(int nCardID,DWORD dwAppendCode , BYTE byProgramNo ,  BYTE byWindowCnt ,BYTE\* pWndParam, BYTE\* pWndData) | |
| Description | Set alone program |
| Parameter | nCardID: Controller ID |
| dwAppendCode: The user’s append code, high byte first. |
| byProNo: Program No.,Valid value:1~255 |
| byWindowCnt: Window count. Valid value:1～10 |
|  | pWndParam：windows parameter  Every window information table has a 22 bytes length parameter. The 1~16 bytes are window position and property, You can see the detail at 1.13. Window position and property; The 17~19 bytes are window data offset; The 20~22 bytes are window data length. High byte first.  If no data ,then window data offset and window data length all are 0.  The total length of the data is: the number of the window\*22. |
| pWndData：Window play data:“Text”、“Picture”…  Byte 1：Data Type(1 Text；4 Picture)  Byte 2：Data Format（Like “Text type” in command 0x85 and “Picture type” in command 0x86）  Byte 3：Text data or picture data。 |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_Net\_QueryProgram

|  |  |
| --- | --- |
| Int CPowerBox\_Net\_QueryProgram( int nCardID ,byte byFlag , byte\* pParam , BYTE\* pBuf , int nBufSize ); | |
| Description | Query program information |
| Parameter | nCardID:Controller ID |
| byFlag:Special which program info will to be query 1: Query valid programs count and program number 2: Query specifies program information.  Other: Reserved |
|  | pParam:  If “byFlag” is 1：byte1~5，resvered，fill 0  If “byFlag” is 2：：byte1，program number；byte2~5，resvered，fill 0 |
| pBuf: The results data buffer |
| nBufSize：The size of results data buffer |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

“pBuf” have the following meanings：

 Query“valid program count and program number”

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x89 | 1 | Describe the package is the return data packet of query program info |
| Info flag |  | 1 | Same with send value ”info flag” |
| parameters |  | 5 | Same with send value ”parameters” |
| Valid program count |  | 1 | Valid program count |
| Valid program number |  | Variable-length | Each byte identifies an effective program。  Valid value 1～100。 |

* The meaning of "return value" in the return packet:

0x01 Controller not running in program template mode

0x10 Unknown info flag

 Query specifies program information

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x89 | 1 | Describe the package is the return data packet of query program info |
| Info flag |  | 1 | Same with send value ”info flag” |
| parameters |  | 5 | Same with send value ”parameters” |
| Information |  | 1 | Now only return one information |
| count |  |  |  |
| Program number |  | 1 | Program number |
| User append  code |  | 4 | User append code |

* The meaning of "return value" in the return packet:

0x01 Controller not running in program template mode

0x10 Unknown info flag

0x11 Invalid programs

0x12 Can’t get program information

CPowerBox\_Net\_SetProgramProperty

|  |  |
| --- | --- |
| int CPowerBox\_Net\_SetProgramProperty( int nCardID, byte byOption , byte byProgramCnt , byte\* pPrograms , byte byPropertyID1 , byte byPropertyID2 , byte byProgramLevel , USHORT nLoopCnt , USHORT nTime , byte\* pDuetime , byte\* pTimeInterval); | |
| Description | Set program property |
| Parameter | nCardID: The control card ID |
| byOption:  Bit0: Set the range of the program property  0: All programes  1: Specify program  Other: Reserved |
| byProgramCnt:The count of the program |
| pPrograms: The list of the programes |
| ByPropertyID1：Property ID 1, marked which property you want to set by  byte, set 0 if the data not exist.  Bit0: The level of the program.  Bit1: The cycle count.  Bit2: Valid time. How long will the program be valid from now on.  Bit3: Interval time  Bit4~7: Reserved |
|  | ByPropertyID2: Property ID 2。Bit0~4: valid time. >0 the count of the valid time.<=4  Bit5~7: Reserved |
| byProgramLevel: The program level. 1～3 level, The high level of the program is priority. |
| nLoopCnt: Loop count, High byte previous(big-endian).  0: Do not play the program, use to shield program temporarily.  1~255: The loop count of the program. |
|  | nTime: Valid time. High byte previous (big-endian). In minute.  0: Not limit play time  >0: Specify play time in minute. |
| pDuetime: time limit |
| pTimeInterval:The interval time. The start tag “Hour/Minute/Second”and the end tag “Hour/Minute/Second” both represent by one byte. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_Net\_SetSchedule

|  |  |
| --- | --- |
| int CPowerBox\_Net\_SetSchedule(int nCardID, DWORD dwAppendCode, BYTE byScheduleNo, const  BYTE\* pProperty, const BYTE\* pBoxes, BYTE byBoxCnt) | |
| Description | Set play schedule |
| Parameter | nCardID: Controller ID |
| dwAppendCode: The user’s append code, high byte first. |
|  | byScheduleNo: Schedule number，Valid value 1~100。Total support 100 plans, For each plan No, the new data cover the old data |
| pProperty: play property，total 14 bytes： byte 0：Format and level： Bit0~3: Data format，fill in 0x01  Bit4~7: Indicates the priority level. The priority level the greater the value, the more priority to play, 0 is the lowest priority.。。  byte 1：Weekday：Bit0~6: 7-bit logo Sunday to Saturday byte 2~4： Begin date ， 3 bytes: Byte1:Year,Valid value0~99,means  2000~2999; Byte2:Month ;Byte3:Day  byte 5~7：End date，3 bytes: Byte1:Year,Valid value0~99,means 2000~2999;  Byte2:Month ;Byte3:Day  byte 8~10：Begin time, 3 bytes:Byte1:Hour；Byte2:Minute；Byte3:Second byte 11~13：End time, 3 bytes:Byte1:Hour；Byte2:Minute；Byte3:Second |
| pBoxes: program number , each byte represents a program. Numbered in  ascending order, do not repeat..... |
|  | byBoxCnt:program number count, Valid value:1～100, |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_Net\_DeleteSchedule

|  |  |
| --- | --- |
| int CPowerBox\_Net\_DeleteSchedule(int nCardID, DWORD dwAppendCode, const BYTE\* pSchs,  BYTE bySchCnt) | |
| Description | Delete play schedule |
| Parameter | nCardID: Controller ID |
| dwAppendCode: The user’s append code, high byte first. |
| pSchs: schedule number, Valid value 1~100。Each byte represents a play schedule。  When delete all play schedule, the length of this data is one , value is 0xff. |
| bySchCnt: The number of play schedule will to be delete。0 means delete all play plans. |
| Return | 0: Success  -1: Can not generate command data  -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

CPowerBox\_Net\_GetSchedule

|  |  |
| --- | --- |
| int CPowerBox\_Net\_GetSchedule(int nCardID, DWORD dwAppendCode, BYTE byType, BYTE byScheduleNo , byte\* pBuf , int nBufSize ) | |
| Description | Get play schedule |
| Parameter | nCardID: Controller ID |
| dwAppendCode: The user’s append code, high byte first. |
| byType：0: Query all valid play plan.  1: Query specified play plan no  Other: Reserved |
| byScheduleNo: Valid value:1~100。When query type is 0，this data fill in 0。 |
| pBuf: The results data buffer |
| nBufSize：The size of results data buffer |
| Return | 0: Success  -1: Can not generate command data |
|  | -2: The command data package error  -3: Can not connect controller  -4: Wrong data subcontract  -5: Timeout not receive the return data  -6: The length of return data is not enough, or wrong data identified  -7: Data validation error |
| Note |  |

“pBuf” have the following meanings:

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Value | Lenght(byte) | Description |
| CC | 0x8d | 1 | Describe the package is the return data which to query play plan |
| Append code |  | 4 | The user’s append code, high byte previous. |
| Query type |  | 1 | 0: Query all valid play plan.  1: Query specified play plan no  Other: Reserved |
| Count /Number |  | 1 | When query type is 0，this value is valid play schedule count  When query type is 1，this value is play schedule number. |
| Play schedule number table/ play schedule content |  | Variable-length | When query type is 0，this value is valid play schedule number table  When query type is 1，this value is play schedule content. Data format like command  0x8B. |

You must deal with the return data according to the different query type.

The meaning of "return value" in the return packet:

0x01 program template is invalid 0x11 Don’t support the query type.

0x12 Invalid play plan no.

0x80 currently is not program template way

# Simple use API function

12.1、Overview of RS232 simple use API function

|  |  |  |
| --- | --- | --- |
| No. | Function name | Description |
| 1 | CP5200\_RS232\_UploadFile | Upload file to controller |
| 2 | CP5200\_RS232\_DownloadFile | Download file from controller |
| 3 | CP5200\_RS232\_RemoveFile | Delete controller file |
| 4 | CP5200\_RS232\_TestController | Test whether controller has connected to PC |
| 5 | CP5200\_RS232\_TestCommunication | Test whether controller communication is normal |
| 6 | CP5200\_RS232\_GetTime | Get controller time |
| 7 | CP5200\_RS232\_SetTime | Set controller time |
| 8 | CP5200\_RS232\_GetTempHumi | Get controller temperature and humidity |
| 9 | CP5200\_RS232\_RestartApp | Restart controller app |
| 10 | CP5200\_RS232\_RestartSys | Restart controller system |
| 11 | CP5200\_RS232\_GetTypeInfo | Get controller type information |
| 12 | CP5200\_RS232\_SendInstantMessage  CP5200\_RS232\_SendInstantMessage1 | Send Instant Message |
| 13 | CP5200\_RS232\_ReadHWSetting | Read scan param |
| 14 | CP5200\_RS232\_WriteHWSetting( | Write scan param |
| 15 | CP5200\_RS232\_ReadSoftwareSwitchInfo | Read software switch info |
| 16 | CP5200\_RS232\_WriteSoftwareSwitchInfo | Write software switch info |
|  |  |  |
| 18 | CP5200\_RS232\_ReadNetworkParam | Read network parameter |
| 19 | CP5200\_RS232\_WriteNetworkParam | Write network parameter |
| 20 | CP5200\_RS232\_Upgrade | Upgrade controller |
|  |  |  |

Usage:

Step 1: Initialize serial port parameters

Only record the serial parameter initialization parameter information, not the actual serial port operation。 Step 2: Use the simple use API function

Note: This category interface need't to consider whether the serial port has been open , as long as the serial port parameters have been initialized.。

12.2、Detail of RS232 simple use API function

CP5200\_RS232\_UploadFile

int CP5200\_RS232\_UploadFile(int nCardID, const char\* pSourceFilename, const char

\*pTargetFilename);

|  |  |
| --- | --- |
| Description | Upload file to controller |
| Parameter | nCardID: Controller ID |
| pSourceFilename: Sourse file name |
| pTargetFilename: Purpose file name |
| Return | 0: Success  -1: Error reading source file  -2: Can not generate the command data  -3: Production start file upload data error or the return data of start file upload errors  -5: Can not open the serial port  -7: Return data of file upload error  -8: File upload does not return data  -9: Production end file upload data errors  -10: Start or end file upload does not return data  -11: Return data of the end file upload errors |
| Note |  |

CP5200\_RS232\_DownloadFile

|  |  |
| --- | --- |
| int CP5200\_RS232\_DownloadFile(int nCardID, const char\* pSourceFilename, const char  \*pTargetFilename); | |
| Description | Download file from controller |
| Parameter | nCardID: Controller ID |
| pSourceFilename: Sourse file name，If the file in the system disk, name needs coupled with the “S:” |
| pTargetFilename: Purpose file name |
| Return | 0: Success  -1: failed  -2: Can not generate the command data  -3: Can’t Open controller file  -4: Can’t get controller file information |
|  | -5: Can not open the serial port  -6: Allocation file buffer failed  -7: Read controller file data error -8: Save file error |
| Note |  |

CP5200\_RS232\_RemoveFile

|  |  |
| --- | --- |
| int CP5200\_RS232\_RemoveFile(int nCardID, const char\* pFilename ); | |
| Description | Delete controller file |
| Parameter | nCardID: Controller ID |
| pFilename : file name，If the file in the system disk, name needs coupled with the “S:” |
| Return | 1: Success  0: Can’t delete file  -1: Incorrect data object handle  -2: Return data type error  -3: Return data length not enough  -4: The buffer length not enough  -5: Can not open the serial port  -6: Can not generate the command data  -7: Can’t get controller file information |
| Note |  |

CP5200\_RS232\_TestController

|  |  |
| --- | --- |
| int CP5200\_RS232\_TestController(int nCardID); | |
| Description | Test whether controller has connected to PC |
| Parameter | nCardID: Controller ID |
| Return | >0: The controller has been connected. |
| Note |  |

CP5200\_RS232\_TestCommunication

|  |  |
| --- | --- |
| int CP5200\_RS232\_TestCommunication(int nCardID); | |
| Description | Test whether controller communication is normal |
| Parameter | nCardID: Controller ID |
| Return | 1: The communication is normal.  0: The communication is not normal. |
| Note | This function is not responsible the opening and closure for the serial port , can be used as test whether the port is turned on.. |

CP5200\_RS232\_GetTime

|  |  |
| --- | --- |
| int CP5200\_RS232\_GetTime(int nCardID, BYTE \*pBuf, int nBufSize); | |
| Description | Get controller time |
| Parameter | nCardID: Controller ID |
| pBuf: time information buffer, the meaning is   1. byte: second 2. byte: minute 3. byte: time 4. bytes: week 5. bytes: day 6. bytes: month 7. bytes: year (2 digits, together with 2000 is the actual year value) |
| nBufSize: The length of time information buffer to require no less than 7 bytes |
| Return | 1: Success |
|  | 0: Failure |
| Note |  |

CP5200\_RS232\_SetTime

|  |  |
| --- | --- |
| int CP5200\_RS232\_SetTime(byte nCardID, const BYTE \*pInfo); | |
| Description | Set controller time |
| Parameter | nCardID: Controller ID |
| pInfo: time information buffer, the meaning is   1. byte: second 2. byte: minute 3. byte: time 4. bytes: week 5. bytes: day 6. bytes: month 7. bytes: year (2 digits, together with 2000 is the actual year value) |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_RS232\_GetTempHumi

|  |  |
| --- | --- |
| int CP5200\_RS232\_GetTempHumi(int nCardID, BYTE \* pBuf, int nBufSize , byte byFlag) | |
| Description | Get controller temperature and humidity |
| Parameter | nCardID: Controller ID |
|  | pBuf: temperature and humidity information buffer, length is 8 bytes , the meanings :  byte 0：Query flag. The same as send package byte 1~2：temperature (degress Celsius)：  Byte 1：Bit7: numeric symbols。1 negative，0 positive。  Bit6~0: the high 7 bit of the integer part of temperature absolute  Byte 2：Bit7~4: the lower 4 bit of the integer part of  temperature absolute  Bit3~0: fractional part ，unit is 1/16(0.0625)  byte 3~4：temperature (degress Fahrenheit)： byte 5：temperature adjustment value，  Bit7: 1 degress Fahrenheit，0 degress Celsius  Bit6: 1 negative，0 positive  Bit5~0: The absolute value of the temperature adjustment  byte 6：humidity。Valid values 0～100 byte 7：humidity adjustment value  Bit7: reserved  Bit6: 1 negative，0 positive  Bit5~0: The absolute value of the humidity adjustment |
| nBufSize: Temperature information buffer length |
| byFlag:Query flag  Bit0: Is query temperature (0 No,1Yes)  Bit1: Is query humidifier (0 No,1Yes) |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_RS232\_RestartApp

|  |  |
| --- | --- |
| int CP5200\_RS232\_RestartApp(byte nCardID); | |
| Description | Restart controller app |
| Parameter | nCardID: Controller ID |
| Return | 1: Success |
|  | 0: Failure |
| Note |  |

CP5200\_RS232\_RestartSys

|  |  |
| --- | --- |
| int CP5200\_RS232\_RestartSys(byte nCardID); | |
| Description | Restart controller system |
| Parameter | nCardID: Controller ID |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_RS232\_GetTypeInfo

|  |  |
| --- | --- |
| int CP5200\_RS232\_GetTypeInfo(byte nCardID, BYTE \*pBuf, int nBufSize); | |
| Description | Get controller type information |
| Parameter | nCardID: controller ID |
| pBuf: control card type information, information means the following:  byte 0: Control Card Type byte 1: FPGA version bytes 2-5: BIOS version bytes 6-9: APP version |
| nBufSize: control card type information length, At least 10 bytes |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_RS232\_SendInstantMessage

int CP5200\_RS232\_SendInstantMessage( byte nCardID, byte byPlayTimes , int x , int

|  |  |
| --- | --- |
| y , int cx , int cy , byte byFontSizeColor , int nEffect , byte nSpeed , byte byStayTime ,const char\* pText ); | |
| Description | Send instant message |
| Parameter | nCardID: Controller ID |
| byPlayTimes: Play times, from 0 to 255. 0 means continue play until new commands arrive. |
| x: Display start point x,the upper left corner of the abscissa. |
| y: Display start point y, the upper left corner of the ordinate. |
| Cx: Display width. 0 means set to maximum width. |
| Cy: Display height. 0 means set to maximum height. |
| byFontSizeColor: Font size and color.  Bit0~3: Font size.  Bit4: The weight of the red color  Bit5: The weight of the green color  Bit6: The weight of the blue color  Bit 7: Reserved |
| nEffect: Display effect. |
| nSpeed: Display speed,0~255.The smaller the faster. Invalid when set to display immediately. |
| byStayTime: Stay time. High byte previous(big endian). |
| pText: The text data. |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_RS232\_SendInstantMessage1

CP5200\_RS232\_SendInstantMessage1( BYTE nCardID, BYTE byPlayTimes , int x , int y , int cx , int cy , int nFontSize , byte byColorAlign , int nEffect , BYTE nSpeed , BYTE byStayTime ,const char\* pText )

|  |  |
| --- | --- |
| Description | Send instant message |
| Parameter | nCardID: Controller ID |
| byPlayTimes: Play times, from 0 to 255. 0 means continue play until new commands arrive. |
| x: Display start point x,the upper left corner of the abscissa. |
| y: Display start point y, the upper left corner of the ordinate. |
| Cx: Display width. 0 means set to maximum width. |
| Cy: Display height. 0 means set to maximum height. |
| nFontSize: font size and style，see 1.7. Font size code and font style |
| byColorAlign：color and alignment  Bit0: Red flag  Bit1: Green flag  Bit2: Blue flag  Bit3: Resvered  Bit4~5: Horizontal alignment. 0 Left, 1 Middle, 2 right  Bit6~7: Vertical alignment. 0 Top , 1 Middle , 2 Bottom |
| nEffect: Display effect. |
| nSpeed: Display speed,0~255.The smaller the faster. Invalid when set to display immediately. |
| byStayTime: Stay time. High byte previous(big endian). |
| pText: The text data. |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_RS232\_ReadHWSetting

|  |  |
| --- | --- |
| int CP5200\_RS232\_ReadHWSetting(byte nCardID, BYTE \*pBuf, int nBufSize , int nPassword); | |
| Description | Read controller scan param |
| Parameter | nCardID: Controller ID |
|  | pBuf: Scan param buffer，at least 16 bytes ,see the meaning of each byte  1.14、The meaning of each byte of the scan parameters |
| nBufSize: Scan param buffer，at least 16 bytes |
| nPassword: Parsing code, depending on the control card filled with different passwords, or not to accept |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_RS232\_WriteHWSetting

|  |  |
| --- | --- |
| int CP5200\_RS232\_WriteHWSetting(byte nCardID, BYTE \*pSetting, int nPassword); | |
| Description | Write controller scan param |
| Parameter | nCardID: Controller ID |
| pSetting: Scan param buffer，16 bytes，see the meaning of each byte  1.14、The meaning of each byte of the scan parameters |
| nPassword: Parsing code, depending on the control card filled with different passwords, or not to accept |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_RS232\_ReadSoftwareSwitchInfo

|  |  |
| --- | --- |
| int CP5200\_RS232\_ReadSoftwareSwitchInfo(BYTE nCardID, BYTE \* pBuf, int nBufSize ) | |
| Description | Read software switch info |
| Parameter | nCardID: control card ID |
| pBuf: software switch info buffer, refrence to:  CP5200\_ParseReadSoftwareSwitchInfoRet pSoftwareSwitchInfoBuf’s description |
|  | nBufSize: software switch info’s len, at least 9 bytes |
| Return | 1: Success  0: Fail |
| Note |  |

CP5200\_RS232\_WriteSoftwareSwitchInfo

|  |  |
| --- | --- |
| int CP5200\_RS232\_WriteSoftwareSwitchInfo(BYTE nCardID,const BYTE \*pBuf ) | |
| Description | Write software switch info |
| Parameter | nCardID: control card ID |
| pBuf: software switch info buffer, refrence to:  CP5200\_MakeWriteSoftwareSwitchInfoData pSoftwareSwitchInfoBuf’s description |
| Return | 1: Success  0: Fail |
| Note |  |

CP5200\_RS232\_ReadNetworkParam

|  |  |
| --- | --- |
| CP5200\_RS232\_ReadNetworkParam(BYTE nCardID, BYTE \*pBuf, int nBufSize ) | |
| Description | Read network connection parameters |
| Parameter | nCardID: control card ID |
| pBuf: The network connection parameters buffer, meaning each byte is as follows:  Byte 0 ~ 3: IP address  Byte 4 ~ 7: gateway  Byte 9 ~ 11: the subnet mask  Byte 12 ~ 13: IP port number  Byte 14 ~ 17: network identification code |
|  | nBufSize: The length of the network connection parameters information, for not less than 18 bytes |
| Return | 1: Success  0: Fail |
| Note |  |

CP5200\_RS232\_WriteNetWorkParam

|  |  |
| --- | --- |
| CP5200\_RS232\_WriteNetworkParam(BYTE nCardID,DWORD dwIP , DWORD dwGateway , DWORD dwIPMast , WORD nPort , DWORD dwIDCode ) | |
| Description | Setting up the network connection parameters |
| Parameter | nCardID: control card ID |
| dwIP：IP address |
| dwGateway：gateway |
| dwIPMast：the subnet mask |
| nPort：IP port number |
| dwIDCode：network identification code |
| Return | 1: Success  0: Fail |
| Note |  |

CP5200\_RS232\_Upgrade

|  |  |
| --- | --- |
| CP5200\_RS232\_Upgrade(int nCardID, int nProgramType , const char\* pProgramFilename) | |
| Description | Upgrade controller program |
| Parameter | nCardID: control card ID |
|  | nProgramType: Upgrad program type  3：BIOS  4：APP  5：SCAN  6：NET  8：BAS  9：GRAPH |
| pProgramFilename: Upgrade program file path name |
| Return | 0: Success  -1: Error reading source file or program type error  -2: Can not generate the command data  -3: Production start file upload data error or the return data of start file upload errors  -4: Make upload file data error  -5: Can not open the serial port  -7: Return data of file upload error  -8: File upload does not return data  -9: Production end file upload data errors  -10: Start or end file upload does not return data  -11: Return data of the end file upload errors |
| Note |  |

12.3、Overview of network simple use API function

|  |  |  |
| --- | --- | --- |
| No. | Function name | Description |
| 1 | CP5200\_Net\_UploadFile | Upload file to controller |
| 2 | CP5200\_RS232\_DownloadFile | Download file from controller |
| 3 | CP5200\_RS232\_RemoveFile | Delete controller file |
| 4 | CP5200\_Net\_TestController | Test whether controller is connected to the PC |
| 5 | CP5200\_Net\_TestCommunication | Test whether controller communication is |
|  |  | normal |
| 6 | CP5200\_Net\_GetTime | Get controller time |
| 7 | CP5200\_Net\_SetTime | Set controller time |
| 8 | CP5200\_Net\_GetTempHumi | Get controller temperature and humidity |
| 9 | CP5200\_Net\_RestartApp | Restart controller app |
| 10 | CP5200\_Net\_RestartSys | Restart controller system |
| 11 | CP5200\_Net\_GetTypeInfo | Get controller type information |
| 12 | CP5200\_Net\_SendInstantMessage  CP5200\_Net\_SendInstantMessage1 | Send instant message |
| 13 | CP5200\_Net\_ReadHWSetting | Read scan param |
| 14 | CP5200\_Net\_WriteHWSetting( | Write scan param |
| 15 | CP5200\_Net\_ReadSoftwareSwitchInfo | Read software switch info |
| 16 | CP5200\_Net\_WriteSoftwareSwitchInfo | Write software switch info |
| 17 | CP5200\_Net\_QueryControllerInfo | Query controller information |
| 18 | CP5200\_Net\_ReadNetworkParam | Read network parameter |
| 19 | CP5200\_Net\_WriteNetworkParam | Write network parameter |
| 20 | CP5200\_Net\_Upgrade | Upgrade controller |
|  |  |  |

Usage:

Step 1: Initialize network parameters

Only record the network parameter initialization parameter information, not the actual network operation。 Step 2: Use the simple use API function

Note: This category interface need't to consider whether the network has been connected, as long as the network parameters have been initialized.。

12.4、Detail of network simple use API function

CP5200\_Net\_UploadFile

|  |  |
| --- | --- |
| int CP5200\_Net\_UploadFile(int nCardID, const char\* pSourceFilename, const char  \*pTargetFilename); | |
| Description | Upload file to controller |
| Parameter | nCardID: Controller ID |
| pSourceFilename: Sourse file name |
| pTargetFilename: Purpose file name |
| Return | 0: Success  -1: Error reading source file  -2: Can not generate the command data  -3: Production start file upload data error or the return data of start file upload errors  -5: Can not connect controller  -7: Return data of file upload error  -8: File upload does not return data  -9: Production end file upload data errors  -10: Start or end file upload does not return data  -11: Return data of the end file upload errors |
| Note |  |

CP5200\_Net\_DownloadFile

|  |  |
| --- | --- |
| int CP5200\_Net\_DownloadFile(int nCardID, const char\* pSourceFilename, const char  \*pTargetFilename); | |
| Description | Download file from controller |
| Parameter | nCardID: Controller ID |
| pSourceFilename: Sourse file name，If the file in the system disk, name needs coupled with the “S:” |
| pTargetFilename: Purpose file name |
| Return | 0: Success  -1: failed  -2: Can not generate the command data  -3: Can’t Open controller file  -4: Can’t get controller file information  -5: Can not connect controller  -6: Allocation file buffer failed  -7: Read controller file data error -8: Save file error |
| Note |  |

CP5200\_Net\_RemoveFile

|  |  |
| --- | --- |
| int CP5200\_Net\_RemoveFile(int nCardID, const char\* pFilename ); | |
| Description | Delete controller file |
| Parameter | nCardID: Controller ID |
| pFilename : file name，If the file in the system disk, name needs coupled with the “S:” |
| Return | 1: Success  0: Can’t delete file  -1: Incorrect data object handle  -2: Return data type error  -3: Return data length not enough  -4: The buffer length not enough  -5: Can not connect controller  -6: Can not generate the command data  -7: Can’t get controller file information |
| Note |  |

CP5200\_Net\_TestController

|  |  |
| --- | --- |
| int CP5200\_Net\_TestController(int nCardID); | |
| Description | Test whether controller is connected to the PC |
| Parameter | nCardID: Controller ID |
| Return | >0: The controller has been connected. |
| Note |  |

CP5200\_Net\_TestCommunication

|  |  |
| --- | --- |
| int CP5200\_Net\_TestCommunication(int nCardID); | |
| Description | Test whether controller communication is normal |
| Parameter | nCardID: Controller ID |
| Return | 1: The communication is normal.  0: The communication is not normal. |
| Note | This function is not responsible the opening and closure for the serial port , can be used as test whether the port is turned on.. |

CP5200\_Net\_GetTime

|  |  |
| --- | --- |
| int CP5200\_Net\_GetTime(int nCardID, BYTE \*pBuf, int nBufSize); | |
| Description | Get controller time |
| Parameter | nCardID: Controller ID |
| pBuf: time information buffer, the meaning is   1. byte: second 2. byte: minute 3. byte: time 4. bytes: week 5. bytes: day 6. bytes: month 7. bytes: year (2 digits, together with 2000 is the actual year value) |
| nBufSize: The length of time information buffer to require no less than 7 bytes |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_Net\_SetTime

|  |  |
| --- | --- |
| int CP5200\_Net\_SetTime(byte nCardID, const BYTE \*pInfo); | |
| Description | Set controller time |
| Parameter | nCardID: Controller ID |
| pInfo: time information buffer, the meaning is   1. byte: second 2. byte: minute 3. byte: time 4. bytes: week 5. bytes: day 6. bytes: month 7. bytes: year (2 digits, together with 2000 is the actual year value) |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_Net\_GetTemperature

|  |  |
| --- | --- |
| int CP5200\_Net\_GetTemperature(int nCardID, BYTE \*pBuf, int nBufSize , byte byFlag) | |
| Description | Get controller temperature and humidity |
| Parameter | nCardID: Controller ID |
|  | pBuf: temperature and humidity information buffer, length is 8 bytes , the meanings :  byte 0：Query flag. The same as send package byte 1~2：temperature (degress Celsius)：  Byte 1：Bit7: numeric symbols。1 negative，0 positive。  Bit6~0: the high 7 bit of the integer part of temperature absolute  Byte 2：Bit7~4: the lower 4 bit of the integer part of  temperature absolute  Bit3~0: fractional part ，unit is 1/16(0.0625)  byte 3~4：temperature (degress Fahrenheit)： byte 5：temperature adjustment value，  Bit7: 1 degress Fahrenheit，0 degress Celsius  Bit6: 1 negative，0 positive  Bit5~0: The absolute value of the temperature adjustment  byte 6：humidity。Valid values 0～100 byte 7：humidity adjustment value  Bit7: reserved  Bit6: 1 negative，0 positive  Bit5~0: The absolute value of the humidity adjustment |
| nBufSize: Temperature information buffer length |
| byFlag:Query flag  Bit0: Is query temperature (0 No,1Yes)  Bit1: Is query humidifier (0 No,1Yes) |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_Net\_RestartApp

|  |  |
| --- | --- |
| int CP5200\_Net\_RestartApp(byte nCardID); | |
| Description | Restart controller app |
| Parameter | nCardID: Controller ID |
| Return | 1: Success |
|  | 0: Failure |
| Note |  |

CP5200\_Net\_RestartSys

|  |  |
| --- | --- |
| int CP5200\_Net\_RestartSys(byte nCardID); | |
| Description | Restart controller system |
| Parameter | nCardID: Controller ID |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_Net\_GetTypeInfo

|  |  |
| --- | --- |
| int CP5200\_Net\_GetTypeInfo(byte nCardID, BYTE \*pBuf, int nBufSize); | |
| Description | Get controller type information |
| Parameter | nCardID: controller ID |
| pBuf: control card type information, information means the following:  byte 0: Control Card Type byte 1: FPGA version bytes 2-5: BIOS version bytes 6-9: APP version |
| nBufSize: control card type information length, At least 10 bytes |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_Net\_SendInstantMessage

int CP5200\_Net\_SendInstantMessage( byte nCardID, byte byPlayTimes , int x , int y , int cx , int cy , byte byFontSizeColor , int nEffect , byte nSpeed , byte byStayTime ,const char\* pText );

|  |  |
| --- | --- |
| Description | Send instant message |
| Parameter | nCardID: Controller ID |
| byPlayTimes: Play times, from 0 to 255. 0 means continue play until new commands arrive. |
| x: Display start point x,the upper left corner of the abscissa. |
| y: Display start point y, the upper left corner of the ordinate. |
| Cx: Display width. 0 means set to maximum width. |
| Cy: Display height. 0 means set to maximum height. |
| byFontSizeColor: Font size and color.  Bit0~3: Font size.  Bit4: The weight of the red color  Bit5: The weight of the green color  Bit6: The weight of the blue color  Bit 7: Reserved |
| nEffect: Display effect. |
| nSpeed: Display speed,0~255.The smaller the faster. Invalid when set to display immediately. |
| byStayTime: Stay time. High byte previous(big endian). |
| pText: The text data. |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_Net\_SendInstantMessage1

|  |  |
| --- | --- |
| CP5200\_Net\_SendInstantMessage1( BYTE nCardID, BYTE byPlayTimes , int x , int y , int cx , int cy , int nFontSize , byte byColorAlign , int nEffect , BYTE nSpeed , BYTE byStayTime ,const char\* pText ) | |
| Description | Send instant message |
| Parameter | nCardID: Controller ID |
|  | byPlayTimes: Play times, from 0 to 255. 0 means continue play until new commands arrive. |
| x: Display start point x,the upper left corner of the abscissa. |
| y: Display start point y, the upper left corner of the ordinate. |
| Cx: Display width. 0 means set to maximum width. |
| Cy: Display height. 0 means set to maximum height. |
| nFontSize: font size and style，see 1.7. Font size code and font style |
| byColorAlign：color and alignment  Bit0: Red flag  Bit1: Green flag  Bit2: Blue flag  Bit3: Resvered  Bit4~5: Horizontal alignment. 0 Left, 1 Middle, 2 right  Bit6~7: Vertical alignment. 0 Top , 1 Middle , 2 Bottom |
| nEffect: Display effect. |
|  | nSpeed: Display speed,0~255.The smaller the faster. Invalid when set to display immediately. |
| byStayTime: Stay time. High byte previous(big endian). |
| pText: The text data. |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_Net\_ReadHWSetting

|  |  |
| --- | --- |
| int CP5200\_Net\_ReadHWSetting(byte nCardID, BYTE \*pBuf, int nBufSize , int nPassword); | |
| Description | Read controller scan param |
| Parameter | nCardID: Controller ID |
| pBuf: Scan param buffer，at least 16 bytes ,see the meaning of each byte  1.14、The meaning of each byte of the scan parameters |
| nBufSize: Scan param buffer，at least 16 bytes |
|  | nPassword: Parsing code, depending on the control card filled with different passwords, or not to accept |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_Net\_WriteHWSetting

|  |  |
| --- | --- |
| int CP5200\_Net\_WriteHWSetting(byte nCardID, BYTE \*pSetting, int nPassword); | |
| Description | Write controller scan param |
| Parameter | nCardID: Controller ID |
| pSetting: Scan param buffer，16 bytes，see the meaning of each byte  1.14、The meaning of each byte of the scan parameters |
| nPassword: Parsing code, depending on the control card filled with different passwords, or not to accept |
| Return | 1: Success  0: Failure |
| Note |  |

CP5200\_Net\_ReadSoftwareSwitchInfo

|  |  |
| --- | --- |
| int CP5200\_Net\_ReadSoftwareSwitchInfo(BYTE nCardID, BYTE \* pBuf, int nBufSize ) | |
| Description | Read software switch info |
| Parameter | nCardID: control card ID |
| pBuf: software switch info buffer, refrence to:  CP5200\_ParseReadSoftwareSwitchInfoRet pSoftwareSwitchInfoBuf’s description |
| nBufSize: software switch info’s len, at least 9 bytes |
| Return | 1: Success  0: Fail |
| Note |  |

CP5200\_Net\_WriteSoftwareSwitchInfo

|  |  |
| --- | --- |
| int CP5200\_Net\_WriteSoftwareSwitchInfo(BYTE nCardID,const BYTE \*pBuf ) | |
| Description | Write software switch info |
| Parameter | nCardID: control card ID |
| pBuf: software switch info buffer, refrence to:  CP5200\_MakeWriteSoftwareSwitchInfoData pSoftwareSwitchInfoBuf’s description |
| Return | 1: Success  0: Fail |
| Note |  |

CP5200\_Net\_QueryControllerInfo

|  |  |
| --- | --- |
| int CP5200\_Net\_QueryControllerInfo(BYTE nCardID, byte byInfoFlag, byte \*pInfoBuf, int nInfoBufLen, const char \*szSavePath ) | |
| Description | Query controller information |
| Parameter | nCardID: controller ID |
| byInfoFlag：Query flag , currently only support 0x0b |
|  | pInfoBuf：Query result buffer byte 0~1: program number. High byte in the front, the first program starting from 1, 0 means no program in play or broadcast information temporarily byte 2~3: Play item number.  Byte 4~7: The program has been broadcast time, the unit is 1/10 of a second, high byte in the front.  Byte 8~11: Play the item have play time, unit is 1/10 of a second, high byte in the front.  Byte 12~13: image width, high byte in the front  Byte 14~15: image height, high byte in the front  Byte 16: color and gray level  Byte 17~20: image data length, high byte in the front |
| nInfoBufLen：Query result buffer lenrth，must bigger than 21 bytes |
| szSavePath：The path of save the display screen images. |
| Return | 1: Successful  0: Failed |
| Note |  |

CP5200\_Net\_ReadNetworkParam

|  |  |
| --- | --- |
| CP5200\_Net\_ReadNetworkParam(BYTE nCardID, BYTE \*pBuf, int nBufSize ) | |
| Description | Read network connection parameters |
| Parameter | nCardID: control card ID |
| pBuf: The network connection parameters buffer, meaning each byte is as follows:  Byte 0 ~ 3: IP address  Byte 4 ~ 7: gateway  Byte 9 ~ 11: the subnet mask  Byte 12 ~ 13: IP port number  Byte 14 ~ 17: network identification code |
|  | nBufSize: The length of the network connection parameters information, for not less than 18 bytes |
| Return | 1: Success  0: Fail |
| Note |  |

CP5200\_Net\_WriteNetWorkParam

|  |  |
| --- | --- |
| CP5200\_Net\_WriteNetworkParam(BYTE nCardID,DWORD dwIP , DWORD dwGateway , DWORD dwIPMast , WORD nPort , DWORD dwIDCode ) | |
| Description | Setting up the network connection parameters |
| Parameter | nCardID: control card ID |
| dwIP：IP address |
| dwGateway：gateway |
| dwIPMast：the subnet mask |
| nPort：IP port number |
| dwIDCode：network identification code |
| Return | 1: Success  0: Fail |
| Note |  |

CP5200\_Net\_Upgrade

|  |  |
| --- | --- |
| CP5200\_Net\_Upgrade(int nCardID, int nProgramType , const char\* pProgramFilename) | |
| Description | Upgrade controller program |
| Parameter | nCardID: control card ID |
|  | nProgramType: Upgrad program type  3：BIOS  4：APP  5：SCAN  6：NET  8：BAS  9：GRAPH |
| pProgramFilename: Upgrade program file path name |
| Return | 0: Success  -1: Error reading source file or program type error  -2: Can not generate the command data  -3: Production start file upload data error or the return data of start file upload errors  -4: Make upload file data error  -5: Can not connect controller  -7: Return data of file upload error  -8: File upload does not return data  -9: Production end file upload data errors  -10: Start or end file upload does not return data  -11: Return data of the end file upload errors |
| Note |  |

# Other API

13.1、Overview of other API

|  |  |  |
| --- | --- | --- |
| No. | Function name | Description |
| 1 | CP5200\_CalcImageDataSize | Image data size calculation |
| 2 | CP5200\_MakeImageDataFromFile | Image data obtained from the image file |
| 3 | CP5200\_TextToImage | Image file generates from formatted text |
| 4 | CP5200\_TextToImageW | Image file generates from formatted text(wide character) |
| 5 | CP5200\_TextToImageEx | Image file generates from extent formatted text |
|  |  |  |

13.2、Detail of other API

CP5200\_CalcImageDataSize

|  |  |
| --- | --- |
| int CP5200\_CalcImageDataSize(WORD imgw, WORD imgh, BYTE color) | |
| Description | Image data size calculation |
| Parameter | imgw: Image width |
| imgh: Image height |
| color: Image color |
| Return | >=0: Image data size |
| Note |  |

CP5200\_MakeImageDataFromFile

|  |  |
| --- | --- |
| int CP5200\_MakeImageDataFromFile(WORD imgw, WORD imgh, BYTE color, BYTE  \*pDatBuf, int nBufSize, const char\* pFilename, int nMode) | |
| Description | Image data obtained from the image file |
| Parameter | imgw: Image width |
| imgh: Image height |
| color: Image color |
| pDatBuf：Image data buffer |
| nBufSize：Image data buffer size |
| pFilename：Picture file path name |
| nMode:Picture mode,see1.9. Picture effect code |
| Return | >=0: Image data size |
|  | -1: Image file not found or load failed  -2: Image conversion failed  -3: Picture mode is wrong  -4: Image data buffer length is not enough |
| Note |  |

CP5200\_TextToImage

|  |  |
| --- | --- |
| int CP5200\_TextToImage( const char \*pSavePath, const char \*pText, const char  \*pFontFaceName, const byte \*pFormatData, const byte \*pScreenData ) | |
| Description | Image file generates from formatted text |
| Parameter | pSavePath：Path to save |
| pText: Text string |
| pFontFaceName：Font face name |
| pformatData：Formatted text control data |
| pScreenData：Screen data |
| Return | 0: Successful  -1：Failed |
| Note |  |

CP5200\_TextToImageW

|  |  |
| --- | --- |
| int CP5200\_TextToImageW( const char \*pSavePath, const wchar\_t \*pText, const char \*pFontFaceName, const byte \*pFormatData, const byte \*pScreenData ) | |
| Description | Image file generates from formatted text(wide character) |
| Parameter | pSavePath：Path to save |
| pText: Text string |
| pFontFaceName：Font face name |
| pformatData：Formatted text control data |
| pScreenData：Screen data |
| Return | 0: Successful  -1：Failed |
| Note |  |

CP5200\_TextToImageEx

|  |  |
| --- | --- |
| int CP5200\_TextToImageEx(const char \*pSavePath, const byte \*pTextContent, const byte \*pFormatData, const byte \*pScreenData ); | |
| Description | Image file generates from extent formatted text |
| Parameter | pSavePath：Path to save |
| pTextContent: Extent formatted text content |
| pFormatData：Extent formatted text control data |
| pScreenData：Extent screen data |
| Return | 0: Successful  -1：Failed |
| Note |  |