Smart bus communication protocol

Ⅰ Define of system communication command

Leading code	Length of data package	Original subnet ID	Original device ID	Original device type	Operate code	Targeted subnet ID	Targeted device ID	content	CRC H（higher then 8）	CRC L（ lower then 8 ）
16bit	8Bit	8Bit	8Bit	16Bit	16Bit	8Bit	8Bit	0-N	8Bit	8Bit
0xAAAA	13-128	0-254	0-254	0-0FFFF	0-0xFFFF	0-254	0-254	0-Nbyte		
1	2	3	4	5	6	7	8	9	10	

Pulse format as following

Leading code

Leading code is starting symbol of data package and fixed format is 0xAAAA，it will start to receive the whole package when the receiver get the fix format from the data and take a data as length of data package.

1. Length of data package：

11-78. content will be nil when length of data package are equal to 11. data content= length of data package-11

2. Original subnet ID

From 0-254

3. Original device ID

From 0-254

4. Original device type

From 0-65535

6. Operate code

From 0-65535, define all kind of operate command and info in the system.。

7. Targeted subnet ID

Two : one From 0-254 and another is 255, 0-254is real ID of the subnet and 255 is full network broadcast ID. Assigned data should sent to right subnet via gateway, when subnet ID is 255, the assigned data should sent to all device in the subnet.

8. Targeted device ID

Two : one From 0-254 and another is 255, 0-254 is real ID of the device and 255 is broadcast ID. Assigned data should be sent to right device , when ID is 255, the assigned data should sent to all device.

9. Content

Optional item.

10. CRC H-CRC L

CRC verification result from data package length to data content

What we need to care when use our device.

1、 Original subnet ID and Original device ID. It means ID of the controller, device ID overlapping is not allowed.

2、 Operate code. For example: scene open: 0x0002, single control is 0x0031, switch series is 0x001a

3、 Targeted subnet ID and Targeted device ID.

4、 Data content

This is related to operate code.

a. scene open(operate code is 0x0002), two bytes: one is for area No and second is scene No. returning command of scene open: operate code is 0x0003, targeted ID is 0xFF,0xFF(return as broadcast), content have two byte: one is for area No and second is scene No.

b. single control(operate code is 0x0031), four byte: first byte for channel No, second byte for intensity, the third one for high time and the fourth for low time. The two latter means that time (second)from present intensity to targeted intensity.

Returning command of single control: operate code is 0x0032, targeted ID is 0xFF,0xFF(return as broadcast), content have three byte: one is for channel No and second is outcome(0xF8—success , 0xF5—fail), the third is channel intensity.

c. Switch series(0x001a), content have three byte: one is for area No and second is series No.

Returning command of switch series: operate code is 0x001B, targeted ID is 0xFF,0xFF(return as broadcast), content have three byte: one is for area No and second is series.

d. status reading of present targeted lamp: operate code is 0x0033, NO CONTENT (CAN BE USEDTO READ LAMP STATUS)

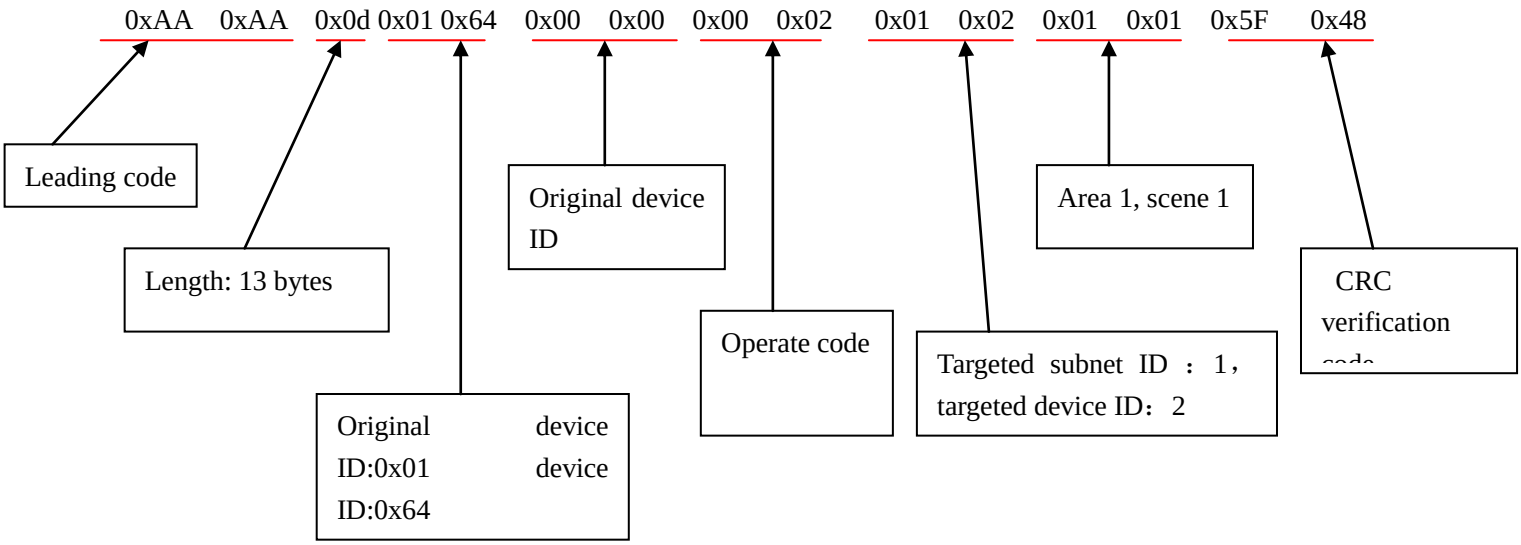
returning command: operate code is 0x0034,

first byte of additional data means total channel No

the latter is intensity of each channel.

Take MD1210 as a example:

Subnet ID: 1, device ID: 2, subnet of controller: 1 device ID of controller: 100, original device type is 0x0000, want to control first scene of first area of MD1210, FORMAT AS FOLLOWING:



Ⅱ RS485 commutation way

9600bps，11bit: initial , 8 data bit, efficacy bit, stop bit

CRC programming written by C language

```
unchar Check_crc(unchar *ptr, unchar len)
{
   unint crc;
   unchar dat;
   crc=0;
   while(len--!=0)
   {
      dat=crc>>8;                    /* */
      crc<<=8;                        /* */
      crc^=CRC_TAB[dat^*ptr];      /* */
      ptr++;
   }
   dat=crc;
   if((*ptr==(crc>>8))&&(*(ptr+1)==dat))
      return(TRUE);
   else
      return(FALSE);
}

void Pack_crc(unchar *ptr, unchar len)
{
   unint    crc;
   unchar dat;
   crc=0;
   while(len--!=0)
   {
```

```
        dat=crc>>8;                     /* */
        crc<<=8;                        /*    */
        crc^=CRC_TAB[dat^*ptr];         /* */
        ptr++;
    }
    *ptr=crc>>8;
     ptr++;
    *ptr=crc;
}
unsigned int    CRC_TAB[]={                     /* CRC tab */
        0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
        0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
        0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
        0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
        0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
        0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
        0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
        0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
        0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
        0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
        0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,
        0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
        0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,
        0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,
        0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
        0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,
        0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,
        0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
        0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
        0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
        0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
        0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
        0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,
        0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
        0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
        0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,
        0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
        0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,
        0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,
        0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
        0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
        0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0
    };
```

Ⅲ UDP communication protocol

Smart bus Ethernet communication protocol is based on winsock to communicate, following take ddelphi as an

example to explain socket.

1, initialed Socket: port: 6000

2, sending data package: using sendto function to send

3, receiving data package: using recvfrom function to receive

maraySendUDPBuf : datagroup of sending data

maraySendUDPBuf[0]:=mbytIPPart1;{eg. IP: 192.168.18.5, so mbytIPPart1=192}
maraySendUDPBuf[1]:=mbytIPPart2;(instead of the second local IP, here is 168)
maraySendUDPBuf[2]:=mbytIPPart3; (instead of the third local IP, here is 18)
maraySendUDPBuf[3]:=mbytIPPart4; (instead of the fourth local IP, here is 5)
maraySendUDPBuf[4]:=$48;{H}
maraySendUDPBuf[5]:=$44;{D}
maraySendUDPBuf[6]:=$4C;{L}
maraySendUDPBuf[7]:=$4D;{M}
maraySendUDPBuf[8]:=$49;{I}
maraySendUDPBuf[9]:=$52;{R}
maraySendUDPBuf[10]:=$41;{A}
maraySendUDPBuf[11]:=$43;{C}
maraySendUDPBuf[12]:=$4C;{L}
maraySendUDPBuf[13]:=$45;{E}

maraySendUDPBuf[14]=$AA; { leading code }
maraySendUDPBuf[15]=$AA; { leading code}
maraySendUDPBuf[16]=$0F; {data package length}
maraySendUDPBuf[17]=$01; {original subnet ID}
maraySendUDPBuf[18]=$FA; {original device ID}
maraySendUDPBuf[19]=$FF; {original device type: higher then 8 }
maraySendUDPBuf[20]=$FE; { original device type: lower then 8 }
maraySendUDPBuf[21]=$00; { operate code: higher then 8 }
maraySendUDPBuf[22]=$31; { operate code: lower then 8}
maraySendUDPBuf[23]=$01; {subnet ID of targeted device }
maraySendUDPBuf[24]=$44 { device ID of targeted device }
maraySendUDPBuf[25]=$01; { additional, channel No }
maraySendUDPBuf[26]=$46; { additional, intensity }
maraySendUDPBuf[27]=$00 { additional, channel running time, higher then 8 }
maraySendUDPBuf[28]=$00; { additional, channel running time, lower then 8 }

maraySendUDPBuf[29]= $1D; {CRC, higher then 8 }
maraySendUDPBuf[30]= $A3; {CRC, lower then 8}

and then we can
sendto (moSocket,maraySendUDPBuf,intUDPBufLen,0,moSockAddrIn,sizeof(moSockAddrIn)); send

out data package of maraySendUDPBuf.

Ⅳ code for CRC verification（for delphi）

1．CRC table

{----------CRC table----------}

```
constCRCTab:array[0..255] of word=(
    $0000, $1021, $2042, $3063, $4084, $50a5, $60c6, $70e7,
    $8108, $9129, $a14a, $b16b, $c18c, $d1ad, $e1ce, $f1ef,
    $1231, $0210, $3273, $2252, $52b5, $4294, $72f7, $62d6,
    $9339, $8318, $b37b, $a35a, $d3bd, $c39c, $f3ff, $e3de,
    $2462, $3443, $0420, $1401, $64e6, $74c7, $44a4, $5485,
    $a56a, $b54b, $8528, $9509, $e5ee, $f5cf, $c5ac, $d58d,
    $3653, $2672, $1611, $0630, $76d7, $66f6, $5695, $46b4,
    $b75b, $a77a, $9719, $8738, $f7df, $e7fe, $d79d, $c7bc,
    $48c4, $58e5, $6886, $78a7, $0840, $1861, $2802, $3823,
    $c9cc, $d9ed, $e98e, $f9af, $8948, $9969, $a90a, $b92b,
    $5af5, $4ad4, $7ab7, $6a96, $1a71, $0a50, $3a33, $2a12,
    $dbfd, $cbdc, $fbbf, $eb9e, $9b79, $8b58, $bb3b, $ab1a,
    $6ca6, $7c87, $4ce4, $5cc5, $2c22, $3c03, $0c60, $1c41,
    $edae, $fd8f, $cdec, $ddcd, $ad2a, $bd0b, $8d68, $9d49,
    $7e97, $6eb6, $5ed5, $4ef4, $3e13, $2e32, $1e51, $0e70,
    $ff9f, $efbe, $dfdd, $cffc, $bf1b, $af3a, $9f59, $8f78,
    $9188, $81a9, $b1ca, $a1eb, $d10c, $c12d, $f14e, $e16f,
    $1080, $00a1, $30c2, $20e3, $5004, $4025, $7046, $6067,
    $83b9, $9398, $a3fb, $b3da, $c33d, $d31c, $e37f, $f35e,
    $02b1, $1290, $22f3, $32d2, $4235, $5214, $6277, $7256,
    $b5ea, $a5cb, $95a8, $8589, $f56e, $e54f, $d52c, $c50d,
    $34e2, $24c3, $14a0, $0481, $7466, $6447, $5424, $4405,
    $a7db, $b7fa, $8799, $97b8, $e75f, $f77e, $c71d, $d73c,
    $26d3, $36f2, $0691, $16b0, $6657, $7676, $4615, $5634,
    $d94c, $c96d, $f90e, $e92f, $99c8, $89e9, $b98a, $a9ab,
    $5844, $4865, $7806, $6827, $18c0, $08e1, $3882, $28a3,
    $cb7d, $db5c, $eb3f, $fb1e, $8bf9, $9bd8, $abbb, $bb9a,
    $4a75, $5a54, $6a37, $7a16, $0af1, $1ad0, $2ab3, $3a92,
    $fd2e, $ed0f, $dd6c, $cd4d, $bdaa, $ad8b, $9de8, $8dc9,
    $7c26, $6c07, $5c64, $4c45, $3ca2, $2c83, $1ce0, $0cc1,
    $ef1f, $ff3e, $cf5d, $df7c, $af9b, $bfba, $8fd9, $9ff8,
    $6e17, $7e36, $4e55, $5e74, $2e93, $3eb2, $0ed1, $1ef0);
```

2．Get two value of CRC: higher then 8 and lower then 8, and distribute to mbytCRCHighData and mbytCRCLowData

Parameter 1：ArayPtrBuf is from data length of data package (not including 0xAA,0xAA）

eg. Data package is （170，170，13, 1, 250, 255, 254, 0, 2, 1, 2, 1, 1, 0, 0)，170 means Hex 0xAA, so parameter 1 ArayPtrBuf is （13, 1, 250, 255, 254, 0, 2, 1, 2, 1, 1, 0, 0) (not including 0xAA,0xAA）

parameter 2：intBufLen deduct 2 byte from this data package(because CRC take 2 byte)

eg. Length of above data package is 13 byte , so intBufLen =13-2=11

finally, it can get CRC verification code from above two parameters that input following hdlPackCRC function.

```
procedure hdlPackCRC(arayPtrBuf:array of byte;intBufLen:integer);
var
  wdCRC:word;
  wdPtrCount:word;
  bytDat:byte;
begin
  try
    wdCRC:=0;
    wdPtrCount:=0;
    while intBufLen<>0 do
    begin
      bytdat:=wdCRC shr 8;
      wdCRC:=wdCRC shl 8;

      wdCRC:=wdCRC xor constCRCTab[bytdat xor arayPtrBuf[wdPtrCount]];
      wdPtrCount:=wdPtrCount+1;
      intBufLen:=intBufLen-1;
    end;

    arayPtrBuf[wdPtrCount]:=wdCRC shr 8;
    mbytCRCHighData:=arayPtrBuf[wdPtrCount];
    wdPtrCount:=wdPtrCount+1;
    arayPtrBuf[wdPtrCount]:=wdCRC and $FF;
    mbytCRCLowData:=arayPtrBuf[wdPtrCount];

  except
    on ex:Exception do
    begin
      MessageDlg(ex.Message+'(hdlPackCRC)',mtError,[mbOK],0);
    end;
  end;
end;
```

3. Use hdlCheckCRC function for CRC verification when receiving data package,

HdlCheckCRC function

Parameter 1: arayPtrBuf also not include two oxAA data package

eg. Data package (170，170，11, 1, 241, 255, 254, 0, 51, 1, 59, 88, 44)，
so arayPtrBuf is (11, 1, 241, 255, 254, 0, 51, 1, 59, 88, 44), exclude 170

parameter 2: intBufLen deduct 2 byte from this data package(because CRC take 2 byte)

eg. Length of above data package is 11 byte , so intBufLen =11-2=9

finally, it can be verified from above two parameters that input following hdlPackCRC function.

```
function hdlCheckCRC(arayPtrBuf:array of byte;intBufLen:integer):boolean;
var
   wdCRC:word;              // dual type variable
   bytDat:byte;
   bytPtrCount:byte;
   blnIsRight:boolean;      // true or false
begin
   wdCRC:=0;
   bytPtrCount:=0;

   try
      while intBufLen<>0 do
      begin
         bytDat:=wdCRC shr 8;
         wdCRC:=wdCRC shl 8;

         wdCRC:=wdCRC xor constCRCTab[bytDat xor arayPtrBuf[bytPtrCount]];
         bytPtrCount:=bytPtrCount+1;
         intBufLen:=intBufLen-1;
      end;

      if(arayPtrBuf[bytPtrCount]=(wdCRC shr 8)) and (arayPtrBuf[bytPtrCount+1]=wdCRC and $ff) then
      begin
         blnIsRight:=true;
      end
      else
      begin
         blnIsRight:=false;
      end;

   except
      on ex:Exception do
      begin
         blnIsRight:=false;
         MessageDlg(ex.Message+'(hdlCheckCRC)',mtError,[mbOK],0);
      end;
   end;
   Result:=blnIsRight;
end;
```

Ⅴ Command on search on-line device(example)

Subnet ID of targeted device: subnet ID from 0 to 255

Device ID of targeted device: device ID255, broadcast

Operate code :0x000E

Example 1: assigned subnet ID :1 search on-line device, device ID:FF
AA AA 0B 01 FA FF FE 00 0E 01 255 0D 3F

Totally 14 byte if send data package via Ethernet.

{take local IP for example， eg.IP : 192.168.18.5, so mbytIPPart1=192}
maraySendUDPBuf[0]:=mbytIPPart1;
maraySendUDPBuf[1]:=mbytIPPart2; (instead of the second local IP, here is 168)
maraySendUDPBuf[2]:=mbytIPPart3; (instead of the third local IP, here is 18)
maraySendUDPBuf[3]:=mbytIPPart4; (instead of the fourth local IP, here is 5)

maraySendUDPBuf[4]:=$48;{H}
maraySendUDPBuf[5]:=$44;{D}
maraySendUDPBuf[6]:=$4C;{L}
maraySendUDPBuf[7]:=$4D;{M}
maraySendUDPBuf[8]:=$49;{I}
maraySendUDPBuf[9]:=$52;{R}
maraySendUDPBuf[10]:=$41;{A}
maraySendUDPBuf[11]:=$43;{C}
maraySendUDPBuf[12]:=$4C;{L}
maraySendUDPBuf[13]:=$45;{E}

example 2 broadcast ID :FF search online, device ID:FF
AA AA 0B 01 FA FF FE 00 0E 01 255 3D F1

Ⅵ device type table

defDeviceType			
DeviceType	Model	DescribeInChn	DescribeInEng
1	HDL-MD0610		6 channels 10A dimmable scene controller
2	HDL-MD1210A		12 channels 10A dimmable scene controller (with load status feedback)
3	HDL-MD2405A		24 channels 5A dimmable scene controller (with load status feedback)
4	HDL-MD0620		6 channels 20A dimmable scene controller
5	HDL-MD1210		12 channels 10A dimmable scene controller
6	HDL-MD2405		24 channels 5A dimmable scene controller
7	HDL-MDH1210A		12 channels 10A dimmable scene controller(with load status feedback)

defDeviceType			
DeviceType	Model	DescribeInChn	DescribeInEng
8	HDL-MD0620A		6 channels 20A dimmable scene controller(with load status feedback)
9	HDL-MDH0610		6 channels 10A dimmable scene controller(with load status feedback)
11	HDL-MR1220		12 channels 20A relay
12	HDL-MR2420		24 channels 20A relay
13	HDL-MC48IP		48 channels scene controller bus (up to 8 unit 6 channels power modules in which is extendible,standard t-shaped tunnel installation with Ethernet channels)
14	HDL-MC48		48 channels scene controller bus (up to 8 unit 6 channel power modules in which is extendible,standard t-shaped tunnel installation)
15	HDL-MD0602		6 channels 2A dimmable scene controller
16	HDL-MC48IPDMX		48 channels scene controller bus (standard t-shaped tunnel installation with Ethernet channels,with DMX)
17	HDL-MRDA06		6 channels,0-10V output,dimmable scene controller of fluorescent lamp
18	HDL-MC48DMX		48 channels scene controller bus (standard t-shaped tunnel installation with DMX)
19	HDL-MR1210		12 channels 10A relay
20	HDL-MC240IP		240 channels show controller
21	HDL-MC512IP		512 channels show controller
22	HDL-MR1205		12 channels 5A relay
23	HDL-MR0810		8 channels 10A relay
25	HDL-MD0403		4 channels 3A dimmable scene controller
26	HDL-MDH2405		24 channels 5A dimmable scene controller
27	HDL-MDH0620		6 channels 20A dimmable scene controller
28	HDL-MD0304		3 channels 4A dimmable scene controller
40	HDL-MC48DALI		48 channels DALI scene controller
50	HDL-MP8RM		8 keys multi-functional panel (with infrared control)
51	HDL-MP8M		8 keys multi-functional panel
52	HDL-MP4RM		4 keys multi-functional panel (with infrared control)
53	HDL-MP4M		4 keys multi-functional panel
54	HDL-MP6R		6 keys scene panel (with infrared control, scene intensity temporary adjustable)

defDeviceType			
DeviceType	Model	DescribeInChn	DescribeInEng
55	HDL-MP6		6 keys scene panel (without infrared control, with scene intensity temporary adjustable)
56	HDL-MP2R		2 keys scene panel (with infrared control, scene intensity temporary adjustable)
57	HDL-MP2		2 keys scene panel (without infrared control, with scene intensity temporary adjustable)
60	HDL-MP8RM		8 keys multi-functional panel (with infrared control)
61	HDL-MP4RM		4 keys multi-functional panel (with infrared control)
62	HDL-MWL16		Wireless receiver
63	HDL-MP4RM		4 keys multi-functional panel (with infrared control)
70	HDL-MSR04		Room partition
80	HDL-MS01R		Motion sensor
81	HDL-MS01L		Linear sensor
90	HDL-MS01R		Motion sensor
91	HDL-MS01L		Linear sensor
92	HDL-MS12		12 channels sensor
93	HDL-MS04		Sensor Input Module
95	HDL-MHS01		Roof-Mounting Infrared Dual-Technology Motion Sensor
96	HDL-MPE01		Ambient Intensity Monitor
97	HDL-MHS02		Wide Field Infrared Dual-Technology Motion Sensor
98	HDL-MHS02		Wide Field Infrared Dual-Technology Motion Sensor
99	HDL-MHS01		Roof-Mounting Infrared Dual-Technology Motion Sensor
100	HDL-MT12IP		12 channels timer (standard t-shaped tunnel installation with Ethernet port)
101	HDL-MT12IP		12 channels timer (standard t-shaped tunnel installation with Ethernet port)
102	HDL-MT01		1 channels Event timer
103	HDL-MT12IP		12 channels timer (standard t-shaped tunnel installation with Ethernet port)
128	HDL-MBUS-RS232		HDL-BUS/RS232 converter
150	HDL-MR1220A		12 channels 20A relay
151	HDL-MR2420A		24 channels 20A relay/ without status feedback
152	HDL-MR0610		6 channels 10A relay
153	HDL-MR0416A		4 channels 16A relay

defDeviceType			
DeviceType	Model	DescribeInChn	DescribeInEng
236	HDL-MBUS01IP		1 port switchboard
237	HDL-MBUS04IP		4 port switchboard
238	HDL-MBUS08IP		8 port switchboard
300	HDL-MIR01		Infrared signal emission,remote receiving module
400	HDL-MAIR01		Air-Condition controller
500	HDL-MTS7000		7″ Ture Color Touch Screen
700	HDL-MW02		Curtain controller
701	HDL-MW02		Curtain controller
800	HDL-MDMXI08		8 channels DMX input module
850	HDL-MC96IPDMX		96 channels scene controller bus (standard t-shaped tunnelinstallation with DMX)
900	HDL-MPM01		Digital electric meter
1000	HDL-MBUS-EIB		EIB/HDL-BUS converter
1005	HDL-MBUS-SAMSUNG		SAMSUNG touch screen convert to HDL-BUS
65534			PC