screen is locked or not bx_sdk_dual.dll appendix file

Appendix 1 Enumeration Parameters

Screen E_ScreenColor_G56

```
public enum E_ScreenColor_G56
{
    eSCREEN_COLOR_SINGLE = 1,    //single color
    eSCREEN_COLOR_DOUBLE,       //dual colors
    eSCREEN_COLOR_THREE,        //seven colors
    eSCREEN_COLOR_FULLCOLOR,    //full color
}
```

Screen E_DoubleColorPixel_G56

```
public enum E_DoubleColorPixel_G56 : int
{
    eDOUBLE_COLOR_PIXTYPE_1 = 1, //dual color 1: G+R
    eDOUBLE_COLOR_PIXTYPE_2,    //dual color 2: R+G
}
```

Single/Multi E_arrMode

```
public enum E_arrMode : int
{
    eSINGLELINE,    //single line
    eMULTILINE,     //multi line
}
```

Date format E_DateStyle

```
public enum E_DateStyle : int
{
    eYYYY_MM_DD_MINUS,    //YYYY-MM-DD
    eYYYY_MM_DD_VIRGURE, //YYYY/MM/DD
    eDD_MM_YYYY_MINUS,    //DD-MM-YYYY
    eDD_MM_YYYY_VIRGURE, //DD/MM/YYYY
    eMM_DD_MINUS,         //MM-DD
    eMM_DD_VIRGURE,       //MM/DD
    eMM_DD_CHS,           //MM month DD day
    eYYYY_MM_DD_CHS,      //YYYY year MM month DD day
}
```

Time format E_TimeStyle

```
public enum E_TimeStyle : int
{
    eHH_MM_SS_COLON,  //HH:MM:SS
    eHH_MM_SS_CHS,    //HH hour MM minutes SS seconds
    eHH_MM_COLON,     //HH:MM
    eHH_MM_CHS,       //HH hour MM minutes
    eAM_HH_MM,        //AM HH:MM
    eHH_MM_AM,        //HH:MM AM
};
```

Week format E_WeekStyle

```
public enum E_WeekStyle : int
{
    eMonday = 1,    //Monday
    eMon,          //Mon.
    eMonday_CHS,  //Monday
}
```

Color value E_Color_G56

```
public enum E_Color_G56
{
    eBLACK,     //black
    eRED,       //red
    eGREEN,     //green
    eYELLOW,    //yellor
    eBLUE,      //blue
    eMAGENTA,   //magenta
    eCYAN,      //cyan
    eWHITE,     //white
}  //time zone of 5th generation supports only 4 colors
```

Clock style E_ClockStyle

```
public enum E_ClockStyle
{
    eLINE,     //line type
    eSQUARE,   //square
    eCIRCLE,   //circle
};//clock style
```

TEXT DIRECTION--DO NOT SUPPORT NOW E_txtDirection

```
public enum E_txtDirection
{
    pNORMAL,       //normal
    pROTATERIGHT,  //rotate right
    pMIRROR,       //mirror
    pROTATELEFT,   //rotate left
};//text direction--do not support now
```

Appendix 2 display stunts

```
0x00 – Random
0x01 – Static
0x02 – Quick type
0x03 – Move left
0x04 – Continuously move left
0x05 – Move up
0x06 – Continuously move up
0x07 – Flicker
0x08 – Floating snow
0x09 – Bubbling
0x0a – Middle out
0x0b – Move left and right
0x0c – Cross left and right
0x0d – Cross up and down
0x0e – Scroll closure
0x0f – Scroll open
0x10 – Stretch left
0x11 – Stretch right
0x12 – Stretch up
0x13 – Stretch down
0x14 – Laser left
0x15 – Laser right
0x16 – Laser up
0x17 – Laser down
0x18 – Left and right cross pull
0x19 – Up and down cross pull
0x1a – Scatter pull left
0x1b – Horizontal blinds
0x1c – Vertical blinds
0x1d – Pull left
0x1e – Pull right
0x1f – Pull up
0x20 – Pull down
0x21 – Left and right close
0x22 – Left and right split
0x23 – Up and down close
0x24 – Up and down split
0x25 – Move right
0x26 – Continuously move right
0x27 – Move down
0x28 – Continuously move down
```

Appendix 3 Construct parameters

ConfigFile

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct ConfigFile
    {
        public byte FileType;
        public byte[] ControllerName;
        ushort Address;
```

```
        public byte Baudrate;
        ushort ScreenWidth;
        ushort ScreenHeight;
        public byte Color;
        public byte MirrorMode;
        public byte OEPol;
        public byte DAPol;
        public byte RowOrder;
        public byte FreqPar;
        public byte OEWidth;
        public byte OEAngle;
        public byte FaultProcessMode;
        public byte CommTimeoutValue;
        public byte RunningMode;
        public byte LoggingMode;
        public byte GrayFlag;
        public byte CascadeMode;
        public byte Default_brightness;
        public byte HUBConfig;
        public byte Language;
        public byte Backup;
        ushort CRC16;
    }
```

Parameter	Specification
FileType	file type
ControllerName	controller name
Address	controller address default address is 0x0001(0x0000 will reserve address), 0xfffe is broadcast address
Baudrate	baud rate 0x00 –reserve the default baud rate 0x01 –set as 9600 0x02 –set as 57600
ScreenWidth	screen width
ScreenHeight	screen height
Color	customized screen color: Bit0 is red, bit1 is green, bits is blue, and for each Bit, 0 is off, 1 is bright

Parameter	Specification
MirrorMode	0x00 –no mirror 0x01 –mirror
OEPol	OE polarity 0x00–OE is low 0x01–OE is high
DAPol	data polarity 0x00 –data is low 0x01 –data is high
RowOrder	row order mode, this value range is 0-31 0-15 means positive sequence 0 means scan from 0 1 means scan from the first row..... 16-31 means negative sequence 0 means scan from 0 1 means scan from the first row
OEWidth	OE width
FaultProcessMode	Fault process mode of controller 0x00 – automatically 0x01 –manually(this mode is only for our testing person
CommTimeoutValue	serial port– 2S TCP/IP – 6S GPRS – 30S
RunningMode	Controller running mode, as below: 0x00 –normal mode 0x01 –adjusting mode
LoggingMode	log records mode 0x00 –no log 0x01 –only record controller error 0x02 –Record for all operations of controller, including: all commands which are received by controller, error and handle error
GrayFlag	gray flag(only support 5Q card) 0x00–no gray 0x01–gray
CascadeMode	cascade mode: (only support 5Q card) 0x00–not cascade mode 0x01–cascade mode
Default_brightness	Only for AX series controller, means the default brightness value when connect with power supply. It is different according to different screen
HUBConfig	HUB configuration (only support 6E series) 0x00–HUB512 default 0x01–HUB256

Parameter	Specification
Language	Multi language 0x00 ----simplified Chinese 0x01 ----Not Chinese, controller will show picture and English characters reserve other values
Backup	Back up bytes
CRC16	CRC16 verification

ConfigFile_G6

```
   [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct ConfigFile_G6
    {
        public byte FileType;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
        public byte[] ControllerName;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 48)]
        byte[] ScreenAddress;
        ushort Address;
        public byte Baudrate;
        ushort ScreenWidth;
        ushort ScreenHeight;
        public byte Color;
        public byte modeofdisp;
        public byte TipLanguage;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 5)]
        public byte[] Reserved;
        public byte FaultProcessMode;
        public byte CommTimeoutValue;
        public byte RunningMode;
        public byte LoggingMode;
        public byte DevideScreenMode;
        public byte Reserved2;
        public byte Default_brightness;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 5)]
        public byte[] Backup;
        public ushort CRC16;
    };
```

Parameters	Specification
FileType	file type
ControllerName	controller name
ScreenAddress	screen installing address is limited: 24 bytes
Address	controller address
Baudrate	baud rate 0x00 –reserve default baud rate 0x01 –set as 9600 0x02 –set as 57600
ScreenWidth	screen width
ScreenHeight	screen height
Color	customized screen color: Bit0 is red, bit1 is green, bits is blue, and for each Bit, 0 is off, 1 is bright
modeofdisp	6Q series display mode: 0 is 888, 1 is 565, for other controller, this byte is 0 6Q
TipLanguage	0 means upper computer is Chinese version, firmware needs to call inside Chinese warning informations when it needs to display warning information 1 means upper computer is English version, firmware needs to call inside English warning informations when it needs to display warning information 255 means upper computer is other language version, firmware needs to call customized warning informations when it needs to display warning information
Reserved	5 backup bytes
FaultProcessMode	fault process mode 0x00 – automatically 0x01 –manually(this mode is only for our testing person
CommTimeoutValue	communication time out value(unit is seconds) serial port– 2S TCP/IP – 6S GPRS – 30S

Parameters	Specification
RunningMode	controller running mode, as below 0x00 –normal mode 0x01 –adjusting mode
LoggingMode	log record mode 0x00 –no mode 0x01 –only record controller error 0x02 –Record for all operations of controller, including: all commands which are received by controller, error and handle error
DevideScreenMode	divide screen mode, only for 6Q2 other card is reserved byte
Default_brightness	Only for AX series controller, means the default brightness value when connect with power supply. It is different according to different screen
Backup	backup bytes
CRC16	CRC16 verification

Ping_data

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public struct Ping_data
{
    public ushort ControllerType;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]
    public byte[] FirmwareVersion;
    public byte ScreenParaStatus;
    public ushort uAddress;
    public byte Baudrate;
    public ushort ScreenWidth;
    public ushort ScreenHeight;
    public byte Color;
    public byte CurrentBrigtness;
    public byte CurrentOnOffStatus;
    public ushort ScanConfNumber;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 9)]
    public byte[] reversed;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 20)]
    public byte[] ipAdder;
}
```

Parameters	Specification
ControllerType	controller type For small end, low order is in front and high order is below, example: 0x254, low order means series, high order [0x54, 0x02] [series, numbers]
FirmwareVersion	firmware version
ScreenParaStatus	screen parameter status 0x00 there is no parameter configuration file in controller, what returned is default parameters of controller. Now, PC software should inform users to write screen parameters first. 0x01 –there is parameter configuration file
uAddress	controller address, screen number
Baudrate	baud rate
ScreenWidth	screen width
ScreenHeight	screen height
Color	customize screen color 1 single color screen 3 dual color screen 7 tri-color screen 255 full color
CurrentBrigtness	current brightness integer 1-16
CurrentOnOffStatus	controller on/off status 0 is off 1 is on
ScanConfNumber	scan configuration number
reversed	reserved
ipAdder	controller ip address

heartbeatData

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct heartbeatData
    {
        public string password;
        public string ip;
        public string subNetMask;
        public string gate;
        public short port;
        public string mac;
        public string netID;
    }
```

Parameters	Specification
password	passwords
ip	controller IP address
subNetMask	subnet mask
gate	gateway
port	port
mac	MAC address
netID	controller network ID

NetSearchCmdRet

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct NetSearchCmdRet
    {
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
        public byte[] Mac;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        public byte[] IP;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        public byte[] SubNetMask;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        public byte[] Gate;
        public ushort Port;
        public byte IPMode;
        public byte IPStatus;
        public byte ServerMode;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        public byte[] ServerIPAddress;
        public ushort ServerPort;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]
        public byte[] ServerAccessPassword;
        public ushort HeartBeatInterval;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 12)]
        public byte[] CustomID;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
```

```
        public byte[] BarCode;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
        public byte[] ControllerType;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]
        public byte[] FirmwareVersion;
        public byte ScreenParaStatus;
        public ushort Address;
        public byte Baudrate;
        public ushort ScreenWidth;
        public ushort ScreenHeight;
        public byte Color;
        public byte BrightnessAdjMode;
        public byte CurrentBrigtness;
        public byte TimingOnOff;
        public byte CurrentOnOffStatus;
        public ushort ScanConfNumber;
        public byte RowsPerChanel;
        public byte GrayFlag;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
        public byte[] UnitWidth;
        public byte modeofdisp;
        public byte NetTranMode;
        public byte PackageMode;
        public byte BarcodeFlag;
        public ushort ProgramNumber;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        public byte[] CurrentProgram;
        public byte ScreenLockStatus;
        public byte ProgramLockStatus;
        public byte RunningMode;
        public byte RTCStatus;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
        public byte[] RTCYear;
        public byte RTCMonth;
        public byte RTCDate;
        public byte RTCHour;
        public byte RTCMinute;
        public byte RTCSecond;
        public byte RTCWeek;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
        public byte[] Temperature1;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
        public byte[] Temperature2;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
        public byte[] Humidity;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
        public byte[] Noise;
        public byte Reserved;
        public byte LogoFlag;
        public ushort PowerOnDelay;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
        public byte[] WindSpeed;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
        public byte[] WindDirction;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
        public byte[] PM2_5;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
        public byte[] PM10;
```

```
        public ushort ExtendParaLen;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
        public byte[] ControllerName;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 44)]
        public byte[] ScreenLocation;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        public byte[] NameLocalationCRC32;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
        public byte[] PM100;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
        public byte[] AtmosphericPressure ;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        public byte[] illumination;
    }
```

Parameters	Specification
Mac	Mac address
IP	controller IP address
SubNetMask	subnet mask
Gate	gateway
Port	port
IPMode	1 means DHCP 2 means manually set
IPStatus	0 means IP set failed 1 means IP set successed

Parameters	Specification
ServerMode	Bit[0] means server mode can be used or not: 1 - can, 0 -cannot Bit[1] means server mode: 1 -web mode, 0 -normal mode
ServerIPAddress	server IP address
ServerPort	server port
ServerAccessPassword	server access password
HeartBeatInterval	default 20s, heardbeat interval (unit: seconds)
CustomID	customized ID, as first half part of network ID, convenient for users to identify controller
BarCode	barcode, as second half of network ID, it is uniqueness
ControllerType	low order byte means equipment series, high order means equipment numbers, example: BX-6Q2 is [0x66, 0x02]
FirmwareVersion	Firmware version
ScreenParaStatus	controller parameter file status 0x00 –there is no parameter configuration file in controller, what returned is default parameters of controller. So now, PC software should inform users to write screen parameters first. 0x01 –There is parameter configuration file in controller
Address	default factory address of controller is 0x0001(0x0000 address will be reserved) controller handle not only the data package which send to itself address, but also to the broadcast data package
Baudrate	baud rate 0x00 –reserve default baud rate 0x01 –set as 9600 0x02 –set as 57600
ScreenWidth	screen width
ScreenHeight	screen height
Color	For non gray system, it will return 1 for single color; return 3 for dual color; return 7 for tri color; For gray system, reutrn 255
BrightnessAdjMode	adjust brightness mode 0x00 –manually adjust brightness 0x01 –timing adjust brightness 0x02 –automatically adjust brightness
CurrentBrigtness	current brightness
TimingOnOff	Bit0 –timing on/off status 0 means non timing on/off 1 means timing on/off
CurrentOnOffStatus	ON status

Parameters	Specification
ScanConfNumber	scan configuration number
RowsPerChanel	row per channel
GrayFlag	for non gray system, return 0; for gray system
UnitWidth	minimum unit width
modeofdisp	6Q display mode: 0 is 888, 1 is 565 other card is 0
NetTranMode	when this byte is 0, lan communication uses old mode. UDP and TCP confirm package length according to PackageMode byte, and when communicated by UDP, divide big package into small packages. It will delay a little when send every small package when this byte is not 0, lan communication uses new mode.Package length of UDP=UDPPackageMode*8KBYTE, and will not be divided. TCP package length of TCP=PackageMode * 16KBYTE
PackageMode	package mode. 0 small package mode, divided package 600 byte. 1 big package mode, divided package 16K byte.
BarcodeFlag	set barcode: If ID set barcode, 0 of this byte will be 1, or it will be 0
ProgramNumber	program numbers of controller
CurrentProgram	current program name
ScreenLockStatus	Bit0 –screen is locked or not 1b’0 –no screen locked, 1b' -screen locked
ProgramLockStatus	Bit0 –program is locked or not 1b’0 –no program locked 1’b1 –program locked
RunningMode	controller running mode
RTCStatus	RTC status 0x00 – RTC unnormal 0x01 – RTC normal
RTCYear	year
RTCMonth	month
RTCDate	day
RTCHour	hour
RTCMinute	minute
RTCSecond	seconds
RTCWeek	week, range from 1~7, 7 means Sunday

Parameters	Specification
Temperature1	current value of temperature sensor
Temperature2	current value of temperature sensor
Humidity	current value of humidity sensor
Noise	current value of noise sensor(current value is dividing 10 into it) for BX - ZS(485) 0xffff invalid
Reserved	reserve byte
LogoFlag	0：means do not set Logo program 1：means set Logo program
PowerOnDelay	0：do not set ON delay 1：set ON delay time
WindSpeed	wind speed(current value is dividing 10 into it) 0xfffff invalid
WindDirction	wind direction(current value) 0xfffff invalid
PM2_5	PM2.5 value(current value) 0xfffff invalid
PM10	PM10 value(current value 0xfffff invalid
ExtendParaLen	extend parameter length
ControllerName	controller name should be 16 byte length(means parameters are losed and invalid if all are 0x00, and upper system will be blank)
ScreenLocation	screen installation address should be 44 byte length(means parameters are losed and invalid if all are 0x00, and upper system will be blank)
NameLocalationCRC32	CRC32 verification value of controller and screen installation address (totally are 60 bytes),this value is for upper system to identify that this 64 bytes mean controller name or mean controller name & screen installation address, as to adopt different handling strategies. as to keep compatibility, lower system will not verify this value.
PM100	PM100(current value) 0xfffff invalid
AtmosphericPressure	atmospheric pressure value（KPa） 0xfffff invlid
illumination	illumination (current value) 0xffffffff invalid

NetSearchCmdRet_Web

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct NetSearchCmdRet_Web
    {
        byte CmdGroup;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
        byte Mac;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        byte IP;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        byte SubNetMask;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        byte Gate;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
        byte Port;
        byte IPMode;
        byte IPStatus;
        byte ServerMode;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        byte ServerIPAddress;
        ushort ServerPort;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]
        byte ServerAccessPassword;
        public ushort HeartBeatInterval;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 12)]
        byte CustomID;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 128)]
        byte WebUserID;
        public int GroupNum;
        byte DomainFlag;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 128)]
        byte DomainName;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 128)]
        byte WebControllerName;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
        byte BarCode;
        ushort ControllerType;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]
        byte FirmwareVersion;
        byte ScreenParaStatus;
        ushort Address;
        byte Baudrate;
        ushort ScreenWidth;
        ushort ScreenHeight;
        byte Color;
        byte BrightnessAdjMode;
        byte CurrentBrigtness;
        byte TimingOnOff;
        byte CurrentOnOffStatus;
        ushort ScanConfNumber;
        byte RowsPerChanel;
        byte GrayFlag;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
```

```
        byte UnitWidth;
        byte modeofdisp;
        byte NetTranMode;
        byte PackageMode;
        byte BarcodeFlag;
        ushort ProgramNumber;
        int CurrentProgram;
        byte ScreenLockStatus;
        byte ProgramLockStatus;
        byte RunningMode;
        byte RTCStatus;
        ushort RTCYear;
        byte RTCMonth;
        byte RTCDate;
        byte RTCHour;
        byte RTCMinute;
        byte RTCSecond;
        byte RTCWeek;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
        byte Temperature1;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
        byte Temperature2;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
        byte Humidity;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
        byte Noise;
        byte Reserved;
        byte LogoFlag;
        ushort PowerOnDelay;
        ushort WindSpeed;
        ushort WindDirction;
        ushort PM2_5;
        ushort PM10;

        ushort ExtendParaLen;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
        byte ControllerName;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 44)]
        byte ScreenLocation;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        byte NameLocalationCRC32;
    };
```

```
C#
```

Parameters	Specification
CmdGroup	0xA4 command group
Mac	Mac address
IP	controller IP address
SubNetMask	subnet mask
Gate	gateway
Port	port
IPMode	1 means DHCP 2 means manually setting
IPStatus	means set IP failed 1 means set IP successed
ServerMode	Bit[0] means server mode can be used or not 1 –can，0 – cannot Bit[1] means server mode 1 –web mode 0 –normal mode
ServerIPAddress	server IP address
ServerPort	server port
ServerAccessPassword	server access password
HeartBeatInterval	default 20S， heartbeat interval(unit: seconds)
CustomID	customized ID, as first half part of network ID, it's convenient for users to identify controller
WebUserID	WEB platform user id
GroupNum	screen group number
DomainFlag	domain mark 0 - no domain，1—domain
DomainName	domain name, send when DomainFlag is 1
WebControllerName	LED00001 WEB platform controller name
BarCode	barcode, as the second half part of network ID, as to be the uniqueness of network ID
ControllerType	low order byte means equipment series, high order byte means equipment numbers, example: BX-6Q2 should be [0x66,0x02]

Parameters	Specification
FirmwareVersion	Firmware version
ScreenParaStatus	controller parameter file status 0x00 –there is no parameter configuration file in controller, what returned is default parameters of controller. So now, PC software should inform users to write screen parameters first. 0x01 –there is parameters configuraiton file in controller
Address	default factory address of controller is 0x0001(0x0000 address will be reserved) controller handle not only the data package which send to itself address, but also to the broadcast data package
Baudrate	buad rate 0x00 –keep default baud rate 0x01 –set as 9600 0x02 –set as 57600
ScreenWidth	screen width
ScreenHeight	screen height
Color	For non gray system, it will return 1 for single color; return 3 for dual color; return 7 for tri color; For gray system, reutrn 255
BrightnessAdjMode	adjust brightness mode 0x00 – adjust brightness manually 0x01 – timing adjust brightness 0x02 – adjust brightness automatically
CurrentBrigtness	current brightness value
TimingOnOff	Bit0 –timing on/off status 0 means non timing on/off 1 means timing on/off
CurrentOnOffStatus	on/off status
ScanConfNumber	scan configuration number
RowsPerChanel	rows per channel
GrayFlag	for non gray system, return 0; for gray system
UnitWidth	minimum width
modeofdisp	6Q display mode: 0 is 888, 1 is 565 , other controllers are 0

Parameters	Specification
NetTranMode	PackageMode * 16KBYTE when this byte is 0, lan communication uses old mode. UDP and TCP confirm package length according to PackageMode byte, and when communicated by UDP, divide big package into small packages. It will delay a little when send every small package when this byte is not 0, lan communication uses new mode.Package length of UDP=UDPPackageMode*8KBYTE, and will not be divided. package length of TCP=PackageMode * 16KBYTE
PackageMode	package mode 0 small package mode, divided package 600 byte 1 big package mode, divided package 16K byte
BarcodeFlag	set barcode: If ID set barcode, 0 of this byte will be 1, or it will be 0
ProgramNumber	program numbers of controller
CurrentProgram	current program name
ScreenLockStatus	Bit0 –screen is locked or not 1b'0 –no screen locked
ProgramLockStatus	Bit0 –program is locked or not 1b'0 –no program locked 1'b1 –program locked
RunningMode	controller running mode
RTCStatus	RTC status 0x00 – RTC unnormal 0x01 – RTC normal
RTCYear	year
RTCMonth	month
RTCDate	day
RTCHour	hour
RTCMinute	minute
RTCSecond	seconds
RTCWeek	week, range from 1~7, 7 means Sunday
Temperature1	current value of temperature sensor
Temperature2	current value of temperature sensor
Humidity	current value of humidity sensor
Noise	current value of noise sensor(current value is dividing 10 into it) for BX - ZS(485) 0xffff invalid

Parameters	Specification
Reserved	reserve byte
LogoFlag	0：means do not set Logo program 1：means set Logo program
PowerOnDelay	0：do not set ON delay 1：set ON delay time
WindSpeed	wind speed(current value is dividing 10 into it) 0xfffff invalid
WindDirction	wind direction(current value) 0xfffff invalid
PM2_5	PM2.5 PM2.5 value(current value) 0xfffff invalid
PM10	PM10 PM10 value(current value) 0xfffff invalid
ExtendParaLen	extend parameter length
ControllerName	controller name should be 16 byte length(means parameters are losed and invalid if all are 0x00, and upper system will be blank)
ScreenLocation	screen installation address should be 44 byte length(means parameters are losed and invalid if all are 0x00, and upper system will be blank)
NameLocalationCRC32	verification value of controller and screen installation address (totally are 60 bytes),this value is for upper system to identify that this 64 bytes mean controller name or mean controller name & screen installation address, as to adopt different handling strategies. as to keep compatibility, lower system will not verify this value.

TimingOnOff

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct TimingOnOff
    {
        public byte onHour;
        public byte onMinute;
        public byte offHour;
        public byte offMinute;
    }
```

Parameters	Specification
onHour	On hour
onMinute	On minute
offHour	Off hour
offMinute	Off minute

Brightness

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct Brightness
    {
        public byte BrightnessMode;
        public byte HalfHourValue0;
        public byte HalfHourValue1;
        public byte HalfHourValue2;
        public byte HalfHourValue3;
        public byte HalfHourValue4;
        public byte HalfHourValue5;
        public byte HalfHourValue6;
        public byte HalfHourValue7;
        public byte HalfHourValue8;
        public byte HalfHourValue9;
        public byte HalfHourValue10;
        public byte HalfHourValue11;
        public byte HalfHourValue12;
        public byte HalfHourValue13;
        public byte HalfHourValue14;
        public byte HalfHourValue15;
        public byte HalfHourValue16;
        public byte HalfHourValue17;
        public byte HalfHourValue18;
        public byte HalfHourValue19;
        public byte HalfHourValue20;
        public byte HalfHourValue21;
        public byte HalfHourValue22;
        public byte HalfHourValue23;
        public byte HalfHourValue24;
        public byte HalfHourValue25;
        public byte HalfHourValue26;
        public byte HalfHourValue27;
        public byte HalfHourValue28;
        public byte HalfHourValue29;
        public byte HalfHourValue30;
        public byte HalfHourValue31;
        public byte HalfHourValue32;
        public byte HalfHourValue33;
        public byte HalfHourValue34;
        public byte HalfHourValue35;
        public byte HalfHourValue36;
        public byte HalfHourValue37;
```

```
        public byte HalfHourValue38;
        public byte HalfHourValue39;
        public byte HalfHourValue40;
        public byte HalfHourValue41;
        public byte HalfHourValue42;
        public byte HalfHourValue43;
        public byte HalfHourValue44;
        public byte HalfHourValue45;
        public byte HalfHourValue46;
        public byte HalfHourValue47;
    }
```

Parameters	Specification
BrightnessMode	0x00 –manually adjust brightness 0x01 –timing adjust brightness. Note: for brightness value list, only need to handle it when it's in timing adjust brightness and manually adjust brightness.
HalfHourValue0	00:00 – 00:29 brightness value 0x00 – 0x0f
HalfHourValue1	00:30-0:59 brightness value 0x00 – 0x0f
…….	……
HalfHourValue47	23:30-23:59 brightness value 0x00 – 0x0f

ControllerStatus_G56

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack = 1)]
    public struct ControllerStatus_G56
    {
        public byte onoffStatus;
        public byte timingOnOff;
        public byte brightnessAdjMode;
        public byte brightness;
        public short programeNumber;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        public byte[] currentProgram;
        public byte screenLockStatus;
        public byte programLockStatus;
        public byte runningMode;
        public byte RTCStatus;
        public short RTCYear;
        public byte RTCMonth;
        public byte RTCDate;
        public byte RTCHour;
        public byte RTCMinute;
        public byte RTCSecond;
        public byte RTCWeek;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
        public byte[] temperature1;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
```

```
        public byte[] temperature2;
        public short humidity;
        public short noise;
        public byte switchStatus;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 12)]
        public byte[] CustomID;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
        public byte[] BarCode;
    }
```

Parameters	Specification
onoffStatus	on/off status Bit 0 –on/off 0 means on 1 means off
timingOnOff	timing on/off status： 0 means no timing on/off， 1 means timing on/off
brightnessAdjMode	brightness mode 0x00 –manually adjust 0x01 –timing adjust brightness 0x02 –automatically adust brightness

Parameters	Specification
brightness	current brightness value
programeNumber	program numbers of controller
currentProgram	current program name
screenLockStatus	screen is locked or not，0 –no screen locked， 1 –screen locked
programLockStatus	program is locked or not 0 –no program locked，1 –program locked
runningMode	controller running mode
RTCStatus	RTC status 0x00 – RTC unnomral 0x01 – RTC normal
RTCYear	year
RTCMonth	month
RTCDate	day
RTCHour	time
RTCMinute	minute
RTCSecond	seconds
RTCWeek	week 1--7
temperature1	current value of temperature sensor 1
temperature2	current value of temperature sensor 2
humidity	current value of humidity sensor
noise	current value of noise sensor
switchStatus	test button status 0 -open 1 -close
CustomID	customized ID, as the first half part of network ID, convenient for users to identify controller
BarCode	barcode, as the second half part of network ID, it's uniqueness of network

TimingReset

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct TimingReset
    {
        public byte rstMode;
        uint RstInterval;
        public byte rstHour1;
        public byte rstMin1;
        public byte rstHour2;
        public byte rstMin2;
        public byte rstHour3;
        public byte rstMin3;
    }
```

Parameters	Specification
rstMode	reset mode 0x00 –cancel timing reset function 0x01 –period reset, now RstInterval is valid 0x02 –reset at specified time
RstInterval	reset period, unit: minute; if this is 0, will not be reset
rstHour1	hour 0Xff -means this group is invalid
rstMin1	minute
rstHour2	hour
rstMin2	minute
rstHour3	hour
rstMin3	minute

BattleTime

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct BattleTime
    {
        public short BattleRTCYear;
        public byte BattleRTCMonth;
        public byte BattleRTCDate;
        public byte BattleRTCHour;
        public byte BattleRTCMinute;
        public byte BattleRTCSecond;
        public byte BattleRTCWeek;
    }
```

Parameters	Specification
BattleRTCYear	year
BattleRTCMonth	month
BattleRTCDate	day
BattleRTCHour	hour
BattleRTCMinute	minute
BattleRTCSecond	seconds
BattleRTCWeek	week

EQprogrampTime_G56

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQprogrampTime_G56
    {
        public byte StartHour;
        public byte StartMinute;
        public byte StartSecond;
        public byte EndHour;
        public byte EndMinute;
        public byte EndSecond;
    };
```

Parameters	Specification
StartHour	start hour
StartMinute	start minute
StartSecond	start seconds
EndHour	start hour
EndMinute	start minute
EndSecond	start seconds

public struct EQprogramppGrp_G56

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQprogramppGrp_G56
    {
        public byte playTimeGrpNum;
        public EQprogrampTime_G56 timeGrp0;
        public EQprogrampTime_G56 timeGrp1;
        public EQprogrampTime_G56 timeGrp2;
        public EQprogrampTime_G56 timeGrp3;
        public EQprogrampTime_G56 timeGrp4;
        public EQprogrampTime_G56 timeGrp5;
        public EQprogrampTime_G56 timeGrp6;
        public EQprogrampTime_G56 timeGrp7;
    };
```

parameters	specification
playTimeGrpNum	valid groups of play time, 0 means no play time, will play whole day, maximum value is 8
timeGrp0	the first group play time EQprogrampTime_G56]()
timeGrp1	the second group play time
timeGrp2	the third group play time
timeGrp3	the fourth group play time
timeGrp4	play time
timeGrp5	play time
timeGrp6	play time
timeGrp7	play time

EQprogramHeader

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQprogramHeader
    {
        public byte FileType;
        public uint ProgramID;
        public byte ProgramStyle;
        public byte ProgramPriority;
        public byte ProgramPlayTimes;
        public ushort ProgramTimeSpan;
        public byte ProgramWeek;
        public ushort ProgramLifeSpan_sy;
        public byte ProgramLifeSpan_sm;
        public byte ProgramLifeSpan_sd;
        public ushort ProgramLifeSpan_ey;
        public byte ProgramLifeSpan_em;
        public byte ProgramLifeSpan_ed;
```

```
    }
```

parameters	specification
FileType	default file type：0x00LOGO file:0x08 scan configuration file:0x02 log file0x06 font file:0x05 prompt message file: 0x07
ProgramID	program ID
ProgramStyle	program style Bit0 –global program Bit1 –dynamic program Bit2 –screensavers program
ProgramPriority	program priority, program priority of play time is 1, program priority of without play time is 0
ProgramPlayTimes	program play times
ProgramTimeSpan	play method 0x0000 –play in orders other -play time(unit - seconds)
ProgramWeek	program week attribute Bit0 – 1 means play everyday in a week Bit0 is 0, set by bit1-bit7, and bit1-bit7 means from Monday to Sunday
ProgramLifeSpan_sy	0xffff–year, month and day are invalid, can play indefinitely 0xfffe- uear is invald others-play start and end year, range is 0x1900 ~ 0x2099，means from year 1900-2099
ProgramLifeSpan_sm	start and end month 0xfe--month is invalid
ProgramLifeSpan_sd	start and end day 0xfe—day is invalid
ProgramLifeSpan_ey	end year
ProgramLifeSpan_em	end date
ProgramLifeSpan_ed	end day

EQscreenframeHeader

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQscreenframeHeader
    {
        public byte FrameDispFlag;
        public byte FrameDispStyle;
        public byte FrameDispSpeed;
        public byte FrameMoveStep;
        public byte FrameWidth;
        public ushort FrameBackup;
    }
```

Parameters	Specification
FrameDispFlag	show frame or not 0x00 –do not show 0x01 – show Note: if do not show frame, it will not send the below data
FrameDispStype	frame display method 0x00 – flicker 0x01 – consequent rotation 0x02 – reserve rotation 0x03 – flicker+consequent roration 0x04 – flicer+reserve rotation 0x05 – red and green colors alternating flicker 0x06 – red and green colors alternating rotation 0x07 – static
FrameDispSpeed	frame display speed
FrameMoveStep	frame move step, unit is pixels, range from 1~16
FrameWidth	frame width
FrameBackup	reserved words

EQareaframeHeader

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQareaframeHeader
    {
        public byte AreaFFlag;
        public byte AreaFDispStyle;
        public byte AreaFDispSpeed;
        public byte AreaFMoveStep;
        public byte AreaFWidth;
        public ushort AreaFBackup;
    }
```

Parameters	Specification
AreaFFlag	Area frame mark 0 without frame, 1 frame
AreaFDispStyle	frame display method 0x00 – flicker 0x01 – consequent rotation 0x02 – reserve rotation 0x03 – flicker+consequent roration 0x04 – flicer+reserve rotation 0x05 – red and green colors alternating flicker 0x06 – red and green colors alternating rotation 0x07 – static
AreaFDispSpeed	frame display speed
AreaFMoveStep	frame move step, unit is pixels, range from 1~8
AreaFWidth	frame length
AreaFBackup	frame width

EQareaHeader

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQareaHeader
    {
        public byte AreaType;
        public ushort AreaX;
        public ushort AreaY;
        public ushort AreaWidth;
        public ushort AreaHeight;
    }
```

Parameters	Specification
AreaType	area type text and graphic subtitle:0x00 font area:0x01 time area:0x02 temperature area：0x03 humidity area：0x04 noise area：0x05 transparent text：0x06 neon area：0x08 fighting time：0x09
AreaX	area top left horizontal coordinate, unit pixel
AreaY	area top left vertical coordinate, unit pixel
AreaWidth	area width, unit pixel
AreaHeight	area height, unit pixel

EQpageHeader

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQpageHeader
    {
        public byte PageStyle;
        public byte DisplayMode;
        public byte ClearMode;
        public byte Speed;
        public ushort StayTime;
        public byte RepeatTime;
        public ushort ValidLen;
        public E_arrMode arrMode;
        public ushort fontSize;
        public uint color;
        public byte fontBold;
        public byte fontItalic;
        public E_txtDirection tdirection;
        public ushort txtSpace;
        public byte Valign;
        public byte Halign;
    }
```

Parameters	Specification
PageStyle	data page style, fixed=0
DisplayMode	display mode
ClearMode	quit mode/clear mode, fixed=0
Speed	speed grade 1-64, 1 is fastest
StayTime	stay time, unit is 10ms
RepeatTime	repeat time, fixed=1
ValidLen	valid width, this is valid only in move left/right mode, default=area width
arrMode	arrow mode--single/multi E_arrMode]()
fontSize	font size
color	font color E_Color_G56 configure 7 colors by this value, if it's bigger than this range, use RGB888 mode
fontBold	it is bold or not, 0 is no, 1 is yes
fontItalic	it is italic or not, 0 is no, 1 is yes
tdirection	text direction E_txtDirection](), invalid
txtSpace	text interval, invalid
Valign	horizontal allign mode (0 is system mode, 1 is left allign, 2 is middle, 3 is right allign)
Halign	vertical allign mode (0 is system mode, 1 is upper allign, 2 is middle, 3 is lower allign)

EQprogram

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQprogram
    {
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        public byte[] fileName;
        public byte fileType;
        public uint fileLen;
        public IntPtr fileAddre;
        public uint fileCRC32;
    }
```

Parameters	Specification
fileName	program parameters file name
fileType	file type
fileLen	parameters file length
fileAddre	file address
fileCRC32	file CRC32 verification

getPageData【】

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct getPageData
    {
        ushort allPageNub;
        uint pageLen;
        public byte[] fileAddre;
    }
```

Parameters	Specification
allPageNub	
pageLen	
fileAddre	

EQunitHeader【】

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQunitHeader
    {
        ushort UnitX;
        ushort UnitY;
        public byte UnitType;
        public byte Align;
        public byte UnitColor;
        public byte UnitMode;
    }
```

Parameters	Specification
UnitX	
UnitY	
UnitType	
Align	
UnitColor	
UnitMode	

EQtimeAreaData_G56

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQtimeAreaData_G56
    {
        public E_arrMode linestyle;
        public uint color;
        public string fontName;
        public ushort fontSize;
        public byte fontBold;
        public byte fontItalic;
        public byte fontUnderline;
        public byte fontAlign;
        public byte date_enable;
        public E_DateStyle datestyle;
        public byte time_enable;
        public E_TimeStyle timestyle;
        public byte week_enable;
        public E_WeekStyle weekstyle;
    }
```

Parameters	Specification
linestyle	lines style E_arrMode]()
color	font color
fontName	font name
fontSize	font size
fontBold	bold or no 0 is no 1 is yes
fontItalic	italic 0 is no 1 is yes
fontUnderline	underline 0 is no 1 is yes
fontAlign	allign mode 0 is left, 1 is middle, 2 is right
date_enable	add date or not 0 is no, 1 is yes
datestyle	date style E_DateStyle]()
time_enable	add time or not 0 is no 1 is yes
timestyle	time format E_TimeStyle]()
week_enable	add week or not 0 is no 1 is yes
weekstyle	week style E_WeekStyle]()

EQAnalogClockHeader_G56

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQAnalogClockHeader_G56
    {
        public ushort OrignPointX;
        public ushort OrignPointY;
        public byte UnitMode;
        public byte HourHandWidth;
        public byte HourHandLen;
        public uint HourHandColor;
        public byte MinHandWidth;
        public byte MinHandLen;
        public uint MinHandColor;
        public byte SecHandWidth;
        public byte SecHandLen;
        public uint SecHandColor;
    }
```

Parameters	Specification
OrignPointX	point horizontal coordinate
OrignPointY	point vertical coordinat
UnitMode	unit mode
HourHandWidth	hour hand width
HourHandLen	hour hand length
HourHandColor	hour hand color
MinHandWidth	minute hand width
MinHandLen	minute hand length
MinHandColor	minute hand color
SecHandWidth	seconds hand width
SecHandLen	seconds hand length
SecHandColor	seconds hand color

EQprogramHeader_G6

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQprogramHeader_G6
    {
        public byte FileType;
        public uint ProgramID;
        public byte ProgramStyle;
        public byte ProgramPriority;
        public byte ProgramPlayTimes;
        public ushort ProgramTimeSpan;
        public byte SpecialFlag;
        public byte CommExtendParaLen;
        public ushort ScheduNum;
        public ushort LoopValue;
        public byte Intergrate;
        public byte TimeAttributeNum;
        public ushort TimeAttributeOOffset;
        public byte ProgramWeek;
        public ushort ProgramLifeSpan_sy;
        public byte ProgramLifeSpan_sm;
        public byte ProgramLifeSpan_sd;
        public ushort ProgramLifeSpan_ey;
        public byte ProgramLifeSpan_em;
        public byte ProgramLifeSpan_ed;
        public byte PlayPeriodGrpNum;
    }
```

Parameters	Specification
FileType	default file type： 0x00LOGO file:0x08 scan configuration file:0x02 log file:0x06 font file:0x05 prompt information file: 0x07
ProgramID	program ID
ProgramStyle	program type Bit0 – global program Bit1 –dynamic program Bit2 –screen savers program
ProgramPriority	program grade, program priority of play time is 1, without play time is 0
ProgramPlayTimes	program play times
ProgramTimeSpan	play mode
SpecialFlag	special program mark
CommExtendParaLen	extend parameters length, default is 0x00
ScheduNum	program scheduling
LoopValue	loop times of schedule
Intergrate	shedule
TimeAttributeNum	time attribute group
TimeAttribute0Offset	attribute offset of the first group time-- support only 1 group now
ProgramWeek	program week attribute
ProgramLifeSpan_sy	start year
ProgramLifeSpan_sm	start month
ProgramLifeSpan_sd	start date
ProgramLifeSpan_ey	end year
ProgramLifeSpan_em	end date
ProgramLifeSpan_ed	end day
PlayPeriodGrpNum	play time groups

EQscreenframeHeader_G6

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQscreenframeHeader_G6
    {
        public byte FrameDispStype;
        public byte FrameDispSpeed;
        public byte FrameMoveStep;
        public byte FrameUnitLength;
        public byte FrameUnitWidth;
        public byte FrameDirectDispBit;
    }
```

Parameters	Specification
FrameDispStype	display frame or not 0x00 –not display 0x01 –display
FrameDispSpeed	frame display speed
FrameMoveStep	frame move steps, unit is pixel, range from 1~16
FrameUnitLength	frame length
FrameUnitWidth	frame width
FrameDirectDispBit	display direction of frame this byte lower 4 (from Bit3 to Bit0), each means the right, left, upon, low frames 0 – display 1 – not display (only support 6QX-M)

EQSound_6G

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQSound_6G
    {
        public byte SoundFlag;
        public byte SoundPerson;
        public byte SoundVolum;
        public byte SoundSpeed;
        public byte SoundDataMode;
        public int SoundReplayTimes;
        public int SoundReplayDelay;
        public byte SoundReservedParaLen;
        public byte Soundnumdeal;
        public byte Soundlanguages;
        public byte Soundwordstyle;
        public int SoundDataLen;
        public IntPtr SoundData;
    }
```

Parameters	Specification
SoundFlag	voice play: 0 means cannot play by voice; 1 means to play SoundDate part
SoundPerson	pronounce person, from 0-5, totally 6 choices SoundFlag（send this byte only when SoundFlag is 1, or do not send, default is 0
SoundVolum	volum, from 0~10, 0 is mute send this byte only when SoundFlag is 1, or do not send, default is 5
SoundSpeed	sound speed, from 0~10 send this byte only when SoundFlag is 1, or do not send, default is 5
SoundDataMode	format of SoundData： 0x00 GB2312; 0x01 GBK; 0x02 BIG5; 0x03 UNICODE send this byte only when SoundFlag is 1, or do not send
SoundReplayTimes	replay times, this value is 0, means play once and this value is 1, play twice this value is 0xffffffff, means play till the end. send this byte only when SoundFlag is 1, or do not send, default is 0
SoundReplayDelay	replay time interval, unit 10ms send this byte only when SoundFlag is 1, or do not send, default is 0
SoundReservedParaLen	reserve voice parameter length, fixed=3
Soundnumdeal	0：judge automatically 1：as number 2：as numerical value send this parameter when SoundFlag is 1 and SoundReservedParaLen is not 0
Soundlanguages	0：identify language automatically 1：arabic numbers, unit of measure, special symbol, Chinese 2：arabic numbers, unit of measure, special symbol, English send this paramter when SoundFlag is 1 and SoundReservedParaLen is not 0 (only support Chinese and English)
Soundwordstyle	0：identify pronounce mode 1：pronounce by characters 2：pronounce by words send this parameter when SoundFlag is 1 and SoundReservedParaLen is not 0
SoundDataLen	length of sound data length send this byte only when SoundFlag is 1, or do not send
SoundData	sound data send this byte only when SoundFlag is 1, or do not send

ClockColor_G56

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct ClockColor_G56
    {
        public uint Color369;
        public uint ColorDot;
        public uint ColorBG;
    }
```

Parameters	Specification
Color369	369 dot color
ColorDot	dot color
ColorBG	Clock frame color

EQareaHeader_G6

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQareaHeader_G6
    {
        public byte AreaType;
        public ushort AreaX;
        public ushort AreaY;
        public ushort AreaWidth;
        public ushort AreaHeight;
        public byte BackGroundFlag;
        public byte Transparency;
        public byte AreaEqual;
        public EQSound_6G stSoundData;
    }
```

Parameters	Specification
AreaType	area type graphic and text subtitle:0x00 font area:0x01 time area:0x02 temperature area：0x03 humidity area：0x04 noise area：0x05 transparent text：0x06 neon area：0x08 firghting time：0x09
AreaX	area top left horizontal coordinate, unit pixel
AreaY	area top left vertical coordinate, unit pixel
AreaWidth	area width, unit pixel
AreaHeight	area height, unit pixel
BackGroundFlag	background, do not support now, fixed 0
Transparency	transparency, do not support now, fixed 101
AreaEqual	prospect, background areas are same or not, do not support now, fixed 0
stSoundData	sound content EQSound_6G]()

EQPicAreaSoundHeader_G6

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack = 1)]
    public struct EQPicAreaSoundHeader_G6
    {
        public byte SoundPerson;
        public byte SoundVolum;
        public byte SoundSpeed;
        public byte SoundDataMode;
        public uint SoundReplayTimes;
        public uint SoundReplayDelay;
        public byte SoundReservedParaLen;
        public byte Soundnumdeal;
        public byte Soundlanguages;
        public byte Soundwordstyle;
    }//图文分区播放语音
```

Parameters	Specification
SoundPerson	sound person, from 0-5 send this byte only when SoundFlag is 1, or do not send, default is 0
SoundVolum	volum, from 0~10, 0 is mute send this byte only when SoundFlag is 1, or do not send, default is 5
SoundSpeed	sound speed,from 0~10 send this byte only when SoundFlag is 1, or do not send, default is 5
SoundDataMode	format of SoundData： 0x00 GB2312; 0x01 GBK; 0x02 BIG5; 0x03 UNICODE send this byte only when SoundFlag is 1, or do not send
SoundReplayTimes	replay times, this value is 0, means play once and this value is 1, play twice this value is 0xffffffff, means play till the end. send this byte only when SoundFlag is 1, or do not send, default is 0
SoundReplayDelay	replay time interval, unit 10ms send this byte only when SoundFlag is 1, or do not send, default is 0
SoundReservedParaLen	reserve voice parameter length, fixed=3
Soundnumdeal	0：judge automatically 1：as number 2：as numerical value send this parameter when SoundFlag is 1 and SoundReservedParaLen is not 0
Soundlanguages	0：identify language automatically 1：arabic numbers, unit of measure, special symbol, Chinese 2：arabic numbers, unit of measure, special symbol, English send this paramter when SoundFlag is 1 and SoundReservedParaLen is not 0 (only support Chinese and English)
Soundwordstyle	0：identify pronounce mode 1：pronounced by characters 2：pronounced by characters send this parameter when SoundFlag is 1 and SoundReservedParaLen is not 0

EQTimeAreaBattle_G6

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQTimeAreaBattle_G6
    {
        public ushort BattleStartYear;
        public byte BattleStartMonth;
        public byte BattleStartDate;
        public byte BattleStartHour;
        public byte BattleStartMinute;
        public byte BattleStartSecond;
        public byte BattleStartWeek;
        public byte StartUpMode;
    }
```

Parameters	Specification
BattleStartYear	start and end year(BCD format, below is same)
BattleStartMonth	start and end month
BattleStartDate	start and end date
BattleStartHour	start and end hour
BattleStartMinute	start and end minute
BattleStartSecond	start and end seconds
BattleStartWeek	start and end week value
StartUpMode	start mode

EQpageHeader_G6

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQpageHeader_G6
    {
        public byte PageStyle;
        public byte DisplayMode;
        public byte ClearMode;
        public byte Speed;
        public ushort StayTime;
        public byte RepeatTime;
        public ushort ValidLen;
        public byte CartoonFrameRate;
        public byte BackNotValidFlag;
        public E_arrMode arrMode;
        public ushort fontSize;
        public uint color;
        public byte fontBold;
        public byte fontItalic;
```

```
        public E_txtDirection tdirection;
        public ushort txtSpace;
        public byte Valign;
        public byte Halign;
    }
```

Parameters	Specification
PageStyle	data page style, fixed=0
DisplayMode	display mode
ClearMode	quit mode/clear mode, fixed=0
Speed	speed grade 1-64, 1 is fastest
StayTime	stay time, unit is 10ms
RepeatTime	repeat time, fixed=1
ValidLen	valid width, this is valid only in move left/right mode, default=area width
CartoonFrameRate	cartoon stunt, this value is frame, fixed=0
BackNotValidFlag	background invalid mark, fixed=0
arrMode	arrow mode--single/multi E_arrMode]()
fontSize	font size
color	font color E_Color_G56 configure 7 colors by this value, if it's bigger than this range, use RGB888 modefontBoldit is bold or not, 0 is no, 1 is yes
fontBold	it is bold or not, 0 is no, 1 is yes
fontItalic	it is italic or not, 0 is no, 1 is yes
tdirection	text direction E_txtDirection](), invalid
txtSpace	text interval, invalid
Valign	horizontal allign mode (0 is system mode, 1 is left allign, 2 is middle, 3 is right allign)
Halign	vertical allign mode (0 is system mode, 1 is upper allign, 2 is middle, 3 is lower allign)

EQprogram_G6

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack = 1)]
    public struct EQprogram_G6
    {
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
```

```
        public byte[] fileName;
        public byte fileType;
        public uint fileLen;
        public IntPtr fileAddre;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        public byte[] dfileName;
        public byte dfileType;
        public uint dfileLen;
        public IntPtr dfileAddre;
    }
```

Parameters	Specification
fileName	program parameter file name
fileType	file type
fileLen	parameter file length
fileAddre	file address
dfileName	program date file name
dfileType	program date file type
dfileLen	data file length
dfileAddre	data file address

GetDirBlock_G56

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack = 1)]
    public struct GetDirBlock_G56
    {
        public byte fileType;
        public ushort fileNumber;
        public IntPtr dataAddre;
    }
```

Parameters	Specification
fileType	obtain file type
fileNumber	return file numbers
dataAddre	return file list address

FileAttribute_G56

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct FileAttribute_G56
    {
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
        public byte[] fileName;
        public byte fileType;
        public int fileLen;
        public int fileCRC;
    }
```

Parameters	Specification
fileName	file name
fileType	file type
fileLen	file length
fileCRC	file CRD verification

EQdynamicHeader

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQdynamicHeader
    {
        public byte RunMode;
        ushort Timeout;
        public byte ImmePlay;
        public byte AreaType;
        ushort AreaX;
        ushort AreaY;
        ushort AreaWidth;
        ushort AreaHeight;
    }
```

Parameters	Specification
RunMode	dynamic area running mode 0— dynamic area data (loop display) 1— display the last page data statically when dynamic area finished display 2— loop display, it will not display if data is not updated after set time 3— loop display, display Logo information (Logo information is the last page of dynamic area) if data is not updated after set time 4— dynamic area data will display in order, and it will not display after the last page
Timeout	time out , unit is seconds
ImmePlay	play immediately or not byte is 0, dynamic area will play together with asynchronous program byte is 1, asynchronous program stops play, only play dynamic area Note: byte is 0, RelateAllPro to RelateProSerialN-1 parameters are valid, or invalid parameter is 1, since it is not play together with asynchronous program, and as to stop play dynamic area immediately, users cam select RunMode parameter as 2 or 4, also, you can delete this dynamic area
AreaType	area type 0x10
AreaX	area top left horizontal coordinate, unit is pixel
AreaY	area top left vertical coordinate, unit is pixel
AreaWidth	area width, unit is pixel
AreaHeight	area height, unit is pixel

EQSoundDepend_6G

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQSoundDepend_6G
    {
        public byte VoiceID;
        public EQSound_6G stSound;
    }
```

Parameters	Specification
VoiceID	ID of each voice, start from 0
stSound	language data EQSound_6G]()

FileCRC16_G56

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct FileCRC16_G56
    {
        IntPtr fileAddre;
        ushort fileLen;
        ushort fileCRC16;
    }
```

Parameters	Specification
fileAddre	file address
fileLen	file length
fileCRC16	file CRC16

###

FileCRC32_G56

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct FileCRC32_G56
    {
        IntPtr fileAddre;
        ushort fileLen;
    }
```

Parameters	Specification
fileAddre	file address
fileLen	file length
fileCRC32	file CRC32

DynamicAreaParams

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct DynamicAreaParams
    {
        public byte uAreaId;
        public bxdualsdk.EQareaHeader_G6 oAreaHeader_G6;
        public bxdualsdk.EQpageHeader_G6 stPageHeader;
        public IntPtr fontName;
        public IntPtr strAreaTxtContent;
    }
```

Parameters	Specification
uAreaId	area ID
oAreaHeader_G6	area attributes
stPageHeader	display data attribute EQareaHeader_G6]()
fontName	font name EQpageHeader_G6]()
strAreaTxtContent	display content data

BxAreaFrmae_Dynamic_G6

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct BxAreaFrmae_Dynamic_G6
    {
    public byte AreaFFlag;       // 1 0x00 area frame mark;
    public EQscreenframeHeader_G6 oAreaFrame;
    public byte[] pStrFramePathFile;
};
```

Parameters	Specification
AreaFFlag	area frame mark 0 is without frame, 1 is frame
oAreaFrame	frame attributes EQscreenframeHeader_G6]()
pStrFramePathFile	frame picture file

BXSound_6G

- Note: this sound construct BXSound_6G is only for dynamic area; for graphic and text area, please use: EQPicAreaSoundHeader_G6

```
[StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
public struct BXSound_6G
{
    public byte SoundFlag;
    public byte SoundPerson;
    public byte SoundVolum;
    public byte SoundSpeed;
    public byte SoundDataMode;
    public int SoundReplayTimes;
    public int SoundReplayDelay;
```

```
    public byte SoundReservedParaLen;
    public byte Soundnumdeal;
    public byte Soundlanguages;
    public byte Soundwordstyle;
    public int SoundDataLen;
    public IntPtr SoundData;
};
```

Parameters	Specification
SoundFlag	sound play; 0 means cannot use sound; 1 means play SoundData part in below content.

Parameters	Specification
SoundPerson	sound person, from 0-5 send this byte only when SoundFlag is 1, or do not send, default is 0
SoundVolum	volum, from 0~10, 0 is mute send this byte only when SoundFlag is 1, or do not send, default is 5
SoundSpeed	sound speed,from 0~10 send this byte only when SoundFlag is 1, or do not send, default is 5
SoundDataMode	format of SoundData： 0x00 GB2312; 0x01 GBK; 0x02 BIG5; 0x03 UNICODE send this byte only when SoundFlag is 1, or do not send
SoundReplayTimes	replay times, this value is 0, means play once and this value is 1, play twice this value is 0xffffffff, means play till the end. send this byte only when SoundFlag is 1, or do not send, default is 0
SoundReplayDelay	replay time interval, unit 10ms send this byte only when SoundFlag is 1, or do not send, default is 0
SoundReservedParaLen	reserve voice parameter length, fixed=3
Soundnumdeal	0：judge automatically 1：as number 2：as numerical value send this parameter when SoundFlag is 1 and SoundReservedParaLen is not 0
Soundlanguages	0：identify language automatically 1：arabic numbers, unit of measure, special symbol, Chinese 2：arabic numbers, unit of measure, special symbol, English send this paramter when SoundFlag is 1 and SoundReservedParaLen is not 0 (only support Chinese and English)
Soundwordstyle	0：identify pronounce mode 1：pronounced by characters 2：pronounced by words send this parameter when SoundFlag is 1 and SoundReservedParaLen is not 0
SoundDataLen	sound data length send this byte only when SoundFlag is 1, or do not send
SoundData	sound data send this byte only when SoundFlag is 1, or do not send

DynamicAreaBaseInfo_5G

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct DynamicAreaBaseInfo_5G
    {
        public byte nType;
        public byte DisplayMode;
        public byte ClearMode;
        public byte Speed;
        public ushort StayTime;
        public byte RepeatTime;
        public EQfontData oFont;
        public IntPtr fontName;
        public IntPtr strAreaTxtContent;
        public IntPtr filePath;
    }
```

Parameters	Specification
nType	nType=1: text；　nType=2: graphic；
DisplayMode	display mode
ClearMode	quite/clear mode, fixed is 0
Speed	speed grade 1-64, 1 is fastest
StayTime	stay time, unit is 10ms
RepeatTime	repeat times, fixed is 1
oFont	font attribute EQfontData]()
fontName	font name
strAreaTxtContent	display text

Parameters	Specification
filePath	picture path

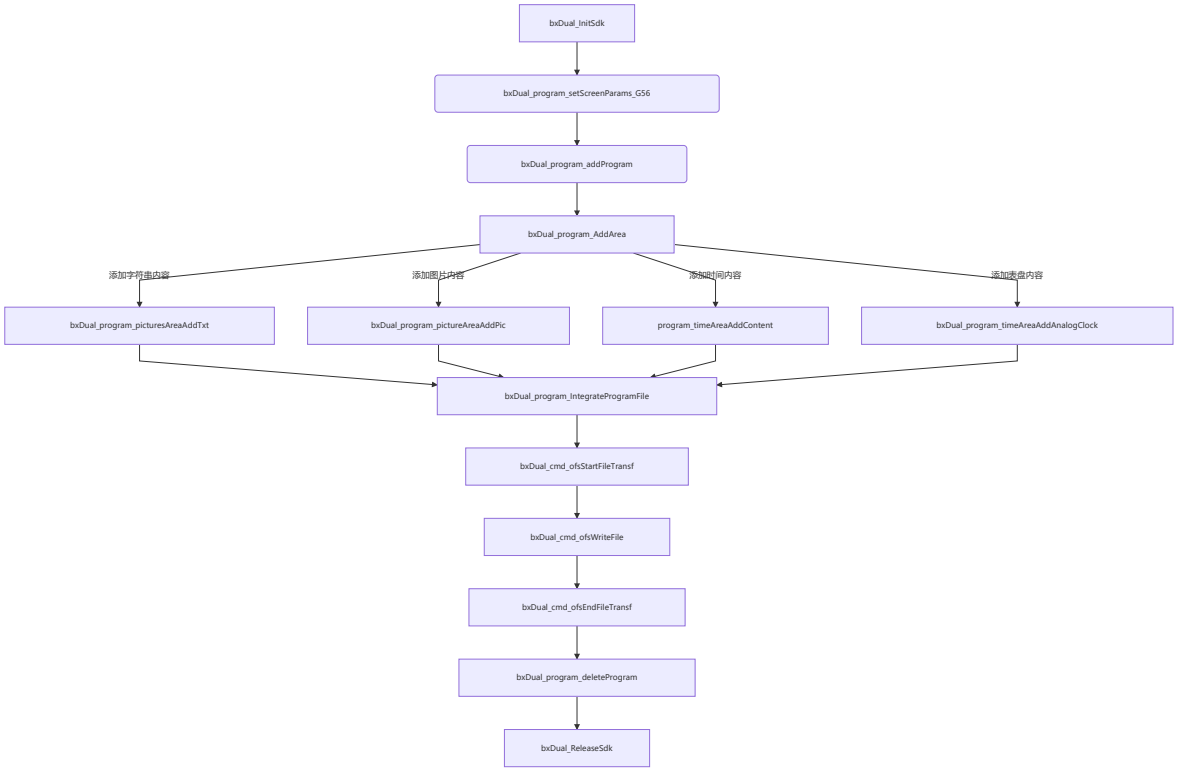
EQfontData

```
    [StructLayoutAttribute(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack =
1)]
    public struct EQfontData
    {
        public E_arrMode arrMode;
        public ushort fontSize;
        public uint color;
        public byte fontBold;
        public byte fontItalic;
        public E_txtDirection tdirection;
        public ushort txtSpace;
        public byte Halign;
        public byte Valign;
    }
```

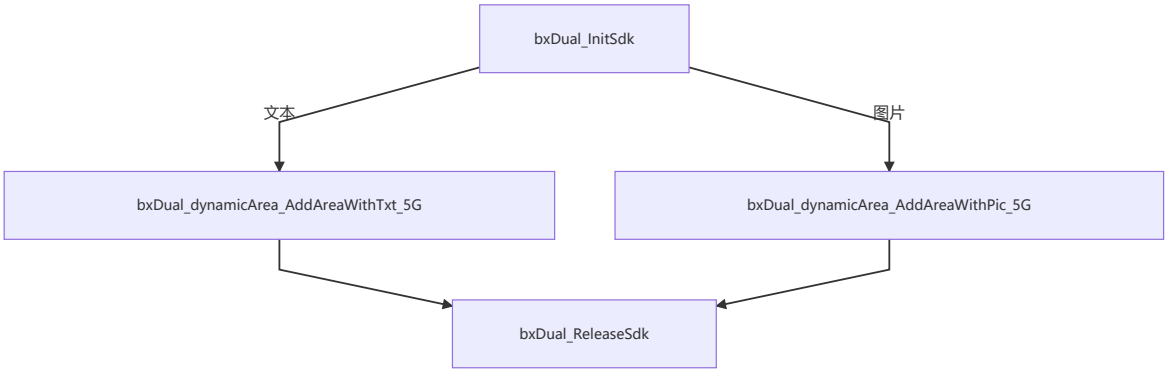
Parameters	Specification
arrMode	arrow mode--single/multi E_arrMode
fontSize	font size
color	font color E_Color_G56 configure 7 colors by this value, if it's bigger than this range, use RGB888 mode
fontBold	it is bold or not, 0 is no, 1 is yes
fontItalic	it is italic or not, 0 is no, 1 is yes
tdirection	text direction E_txtDirection](), invalid
txtSpace	words interval, invalid
Halign	horizontal allign mode (0 is system mode, 1 is left allign, 2 is middle, 3 is right allign)
Valign	vertical allign mode (0 is system mode, 1 is upper allign, 2 is middle, 3 is lower allign)

Appedix 4 send process

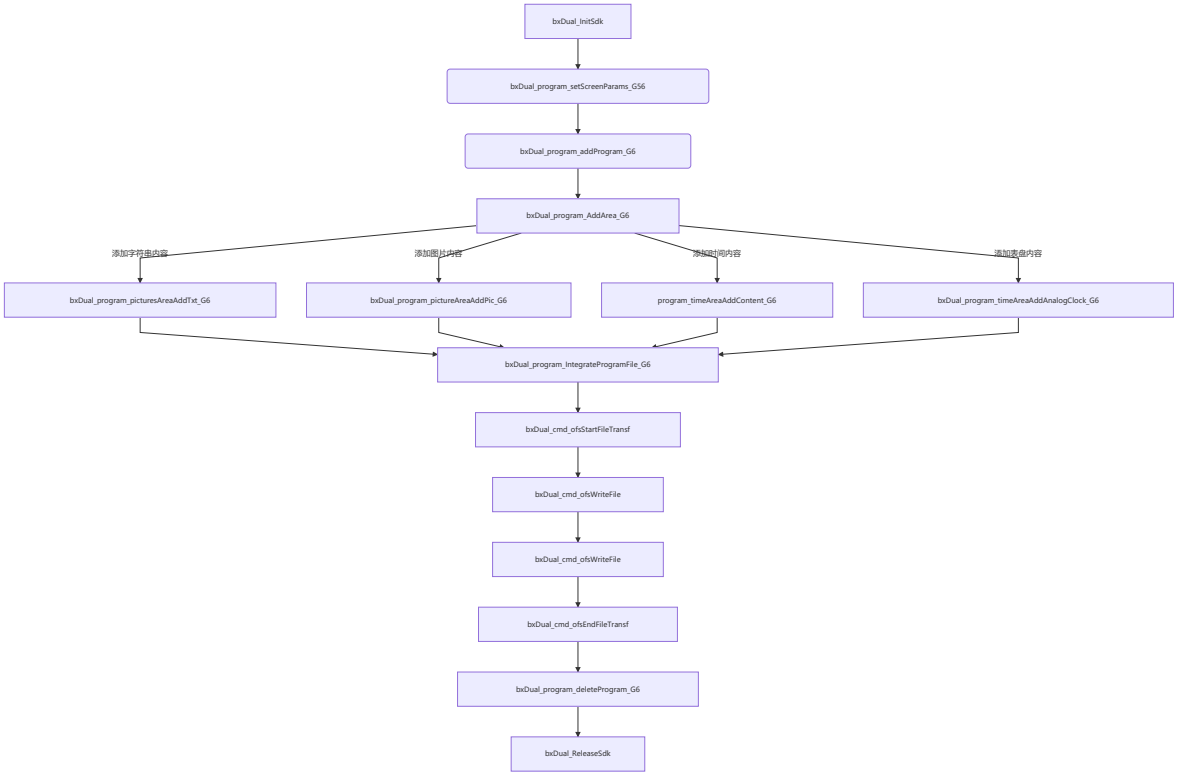
BX-5 series (send program)



BX-5 series (send dynamic area BX-5E)



BX-6 series (send program)



BX-6 series (send dynamic area BX-6E/6EX)