Programmer Handbook

(LedOK2.0) 2007-12

Catalogue

I.Chapter One Summarization

1.1.Control Card's Disposal Process and Programmer's Exploiture Procedure

1.2.DLL'S Transfering Methods and Attentions

1.3. Format Transfering Examples of C language, VB and Delphi

II.Chapter Two File Management

2.1. Storage Sequence of Item's Catalogue Management

2.1.1Disk Storage Catalogue Structure of Repertoire Files and Configuration Files

2.1.2 Configuration Files Format Analysis

2.1.3 TimeFomat Detailed Explanation

2.2. Subtitles Detailed Explanation

2.3.Detailed Explanation for Media Area Playing List

III.Chapeter Three Network Protocol Format

3.1. Situations for Protocol Application

3.2. Rules for Protocol Application

3.3. Protocol Structure

3.4. Common Response of Controller Protocol

3.5. Command Types Enumerating

IV.Chapeter Four Communication Funciton

4.1. Error-Checking Function

4.2. Control Card Checking Function

V. Chapeter Five I/O Pin Function

5.1.Pin Definition

5.2.I/O Pin Operating Protocol

5.3,I/O Pin State Checking and Function Setting

VI.Chapeter Six True-Time Insert Playing Function

6.1. Picture Function for Inserting bmp,jpg and gif format

6.2. Function for Screen Flickering

6.3.Inserting Multi-Pictures Stunt

VII.Chapeter Seven Audio Playing Protocol

7.1. The Contents of Protocol

VIII.Chapeter Eight Screen Parameter-Setting Protocol

8.1. Screen Command Parameter and the Corresponding Analysis

8.2.1 Main Function

8.2.2.The Corresponding Sub-functions of Testing Orders

8.2.3. (Non-testing type) the corresponding sub-text meaning to the main function

IX.Chapeter Nine Controlling System State-Checking and Basic Parameter Commands Setting

9.1. State Checking Commands

9.2. Loop Check Command

9.3. Controller Time Setting

Chapter One Summarization

The secondary development of LedoK control system can support 10M/100M network communication.

The playing datas of Ledok Cotrolling system has adopted yaffs's file management system. All pictures for playing will be stored as files in the flash disk of the control card.Computer's software making, items, screen, programmes and pictures will be stored in classification by file catalogue.The video playing method and settling time etc will be controlled by program list file. And all programs in the list must be config.sys.

The cotrol card can store as many as 100 sets programs.Different programs and the related pictues will be put into different catalogues according to programs' indexes.

1.1.Control Card's Disposal Process and Programmer's Exploiture Procedure

Just imagine the Ledok control system as a microcomputer, then it will be helpful to understand the controller's working process. All the programs/pictures for playing and their playing modes in Ledok controlling system will all be stored to the controller's disk in the form of files format.

1.2. DLL'S Transfering Methods and Attentions

(1) All DLL and Lib files can be put to the catalogue or folders under :windows/system32/.

(2)When transferring VC, it is necessary to set the LIB of link.(Project -> Settings -> Link)

(3) When transferring VB and Delphi, it is only necessary to link DLL files.

1.3. Format Transfering Examples of C language, VB and Delphi

ShowPicStunt

C Language DLL：ShowPicStunt(

char* cCardId, // character string point of control card's sequence No.

char* cCardIp, // character string point the appointed control card's IP address

nt iGroup, //team No., not use for the moment, to use the value zero is ok.

int iProtoType // Network communication protocol method, use the value 1,adopt TCP method

);

Function: After the function "AddPicAndStunt" sending and storaging the pictures and the corresponding stunts to the control card, use this function "ShowPicStunt" to display pictures and stunts on the screen.

Reture Value: 0: success 1: failed (At this moment, use "GetDllErr" to get the reasons.)

C language transferring format：extern "C" __declspec(dllexport) int __stdcall ShowPicStunt (int PicType, char* cCardId, char* cCardIp, int iGroup, int iProtoType);

VB language transferring format：Private Declare Function ShowPicStunt Lib "LwDll.dll" (ByVal PicType As Integer ， ByVal cCardId As String, ByVal cCardIp As String ， ByVal iGroup As Integer ， ByVal iProtoType As Integer) As Long

Delphi language transferring format：Function ShowPicStunt (PicType:integer，cCardId:string,cCardIp:string,iGroup:intege，iProtoType:integer):integer;stdcall;external 'LWDLL.dll';

Chapter Two File Management

2.1. Storage Sequence of Item's Catalogue Management

2.1.1Disk Storage Catalogue Structure of Repertoire Files and Configuration Files Detailed discriptions are listed as follows(* stands for multi-files of the same type)
All the configuration files are as follows:

```
|--use-defined folders ---|---Config.sys                  // Screen Configuration Files.
                          |---Upgrade.sys                 // Upgraded Configuration Files
                          |---/Area*----- |---Config.sys      //AreaConfigurationFiles,
which only can support /Area0
                                          |---tcmd.sys//TimingPlaying Configuration Files
                                          |---*.jpg  // area framed pictures
                                          |---*.jpg    *.bmp //Regional Backgroud Picture.
                                           |---/Audio/-----|---Config.sys     //      Background
music configuration files
                                                                |---*.mp3    //Background    music
files
                                           |---/Program*---|---Config.sys//     Playing     list
 configuration files
                                                                |---/Jpeg----|---/Folder*---*.jpg
                                                                |                |---*.jpg//Jpeg files.
                                                                |---/Bmp-----|---/Folder*---*.bmp
                                                                       |          |---*.bmp // Bmp file
                                                                       |---/Gif-----...     // Gif folder
                                                                       |---/Font----...// Font folder
                                                                  |---/Clock---...//Clock folder
                                                              |---/Flash---...//Flashfolder
                                                          |---/Video---...//Video folder
```

Attention: As there is differences between Chinese and English in LED controller recognition. Therefore, all the character arrays of files names in configuration-files' structure will have both Chine and English arrays.
For example:

CHAR pFileOrPathName[FILE_NAME_SIZE]; // the source file name displyed by user's software

CHAR pFileOrPathEnglishName[FILE_NAME_SIZE];//The file name stored on the hard disk

2.1.2 Configuration Files Format Analysis
① screen configuration files(SYTEM_CONFIG_FILE)
The parameters of this kind of files are regulated,therefore, it is ok just to use the fixed structure to define it,which is convenient to deal with.

```
typedef  struct     _AREA_FILL_COLOR                        // to color the area
{
    UINT    uiFillType;                  // FillType:Value,Rgb,Jpeg;
```

```
    FB_PIXEL_TYPE    uiColorValue;          //    Value.
    FB_PIXEL_TYPE    ucRed;                 // Rgb of Red.
    FB_PIXEL_TYPE    ucGreen;               // Rgb of Green.
    FB_PIXEL_TYPE    ucBlue;                // Rgb of Blue.
    CHAR             pFileName[FILE_NAME_SIZE];       // Jpeg File Name.
}AREA_FILL_COLOR,* P_AREA_FILL_COLOR;
typedef  struct    _AREA_PARAMS                        // area difinition.
{
    UINT             xStart;        // the starting point coordinatte of x
    UINT             yStart;        // the starting point coordinate of y
    UINT             xLength;           // x length
    UINT             yLength;           // y length
    //    all the parameters listed below are got through calculations
    UINT             xStop;         // the x ending point which equals xstart +xLength.
    UINT             yStop;         // the y ending point which equals yStart+yLength.
    UINT             xEnd;          // the x ending point coordinate ,equals xStop-1.
    UINT             yEnd;          // the y ending point coordinate ,equalsyStop-1.
    UINT             uiPixelSize;  // pixel size,equals xLength*yLength.
    UINT    uiFrameBufferSize;    //bytes    in    EMS    memory,    equals
uiPixelSize*sizeof(FB_PIXEL_TYPE).
}AREA_PARAMS,* P_AREA_PARAMS;

// Screen Configuration Files(SYTEM_CONFIG_FILE)The parameters of this kind of
files are regulated,therefore, it is ok just to use the fixed structure to define it,which is
convenient to deal with //the catalogue is in :/mnt/Project/Config.sys
typedef  struct    _PROJECT_AREA_PARAMS // Screen Area Definition
{    UINT    uiLoopbackNum;// circulated times,0: infinite loop.
    UINT    uiConfigNum;//total areas,the max area:AREA_MAX_NUM.
    CHAR    pAreaName[AREA_MAX_NUM][PATH_NAME_SIZE];//Area Name
}PROJECT_AREA_PARAMS,* P_PROJECT_AREA_PARAMS;
typedef  struct    _PLAY_SYSTEM // Screen Configuration Files Structure Definition
{    CHAR pStageTypeStart[PLAY_LIST_TYPE_SIZE]; // beginning mark: Please see
the definition of CONFIG_FILE_TYPE_START.
    UINT uiVersion;//Version number of system configuration files,as for the
    value,please see the definition in “ function requests of computer compilation
    software”.
    UINT    uiScreenType; // screen types, see its definition in "_SCREEN_TYPE".
    UINT    uiScreenBitsPerPiexl;// screen pixels BGR bits:16(5,6,5)/24(8,8,8).
    UINT    uiGray; //grayscale
    UINT    uiScreenWidth;   // screen width.
    UINT    uiScreenHeight;  // screen height.
    UINT    uiScreenRefreshFrequ;// screen refresh rate.
    float    fGamaColorModify;  // Gama color rectification value
    AREA_FILL_COLOR    screenBackround;// screen background definition
```

CHAR IP Address[32]; // card's IP address ,default: "192.168.0.200".
UINT uiIpPort; // card's port number,default "31296".
CHAR pFpgaObjName[FILE_NAME_SIZE]; //FPGA target code loading file name.
PROJECT_AREA_PARAMS projectAreaParams; // Need At Last.
UINT uiReserve[100];
CHAR pStageTypeEnd[PLAY_LIST_TYPE_SIZE]; // ending mark.
UINT uiCheckCode; // 32bits check sum from the beginning to the ending mark(including beginning and ending marks)
}PLAY_SYSTEM,* P_PLAY_SYSTEM;

② Area configuration files (SYTEM_CONFIG_FILE)

The parameters of this kind of files are regulated, therefore, it is ok just to use the fixed structure to define it,which is convenient to deal with.

typedef struct _PLAY_AREA // configuration files struncture definition
{ CHAR pStageTypeStart[PLAY_LIST_TYPE_SIZE]; //Beginning mark:see the difinition in: //CONFIG_FILE_TYPE_START.
AREA_PARAMS reaParams; // area size ,Only need the beginning coordinate and area size, nothing else.
UINT uiSolidWidth;// are's width (pixels),0:without solid effect.
UINT uiFrameWidth;// area frame width(pixels),0: without frames
UINT reserve[3];
AREA_FILL_COLOR rameParams;// Color definition of area's frames
AREA_FILL_COLOR creenBackround; //area background color definition.
UINT uiLoopbackNum; //circulated times,0:infinite loop.
UINT uiConfigNum; // advertisements' quantity, max: AD_MAX_NUM.
CHAR pPathName[AD_MAX_NUM][PATH_NAME_SIZE]; //All kinds of advertisements' folders name.
CHAR pPathEnglishName[AD_MAX_NUM][PATH_NAME_SIZE];
UINT uiReserve[100];
CHAR pStageTypeEnd[PLAY_LIST_TYPE_SIZE]; // ending mark.
UINT uiCheckCode;// 32 bits check sum from the beginning to the ending mark.(including the beginning and ending mark)
}PLAY_AREA,* P_PLAY_AREA;

③Playing list configuration files (SYTEM_CONFIG_FILE)

The parameters of this kind of files are not regulated,therefore, it can not be difined by the fixed structure. The parts are defined as different types of file structure.

{ CHAR pStageTypeStart[PLAY_LIST_TYPE_SIZE]; //beginning mark: see: CONFIG_FILE_TYPE_START.
UINT uiTimeAreaFlag; // Time Area Mark 0: without 1: with
UINT uiMediaAreaFlag; // media file area marks 0: without 1: with
UINT uiPmdAreaFlag; // Subtitles area's marks 0:without, 1: with
UINT uiAreaModType; // Area mode types 0: mode 1，1:mode 2，and by analogy

to others.

ST_TIMER st_Timer; //time structure

ST_FONTSCREEN st_Pmd; // Subtitles Light structure

。。。[media area playing list]。。。//the quantity of programs is not fixed.It can be saved according to cutomized sequences.

AREA_FILL_COLOR bkAreafillcolor;// Background-fill-color

UINT uiLoop_forever;//0 0:the program is permanent, other values: circulated times

CHAR pStageTypeEnd[PLAY_LIST_TYPE_SIZE]; // ending mark.

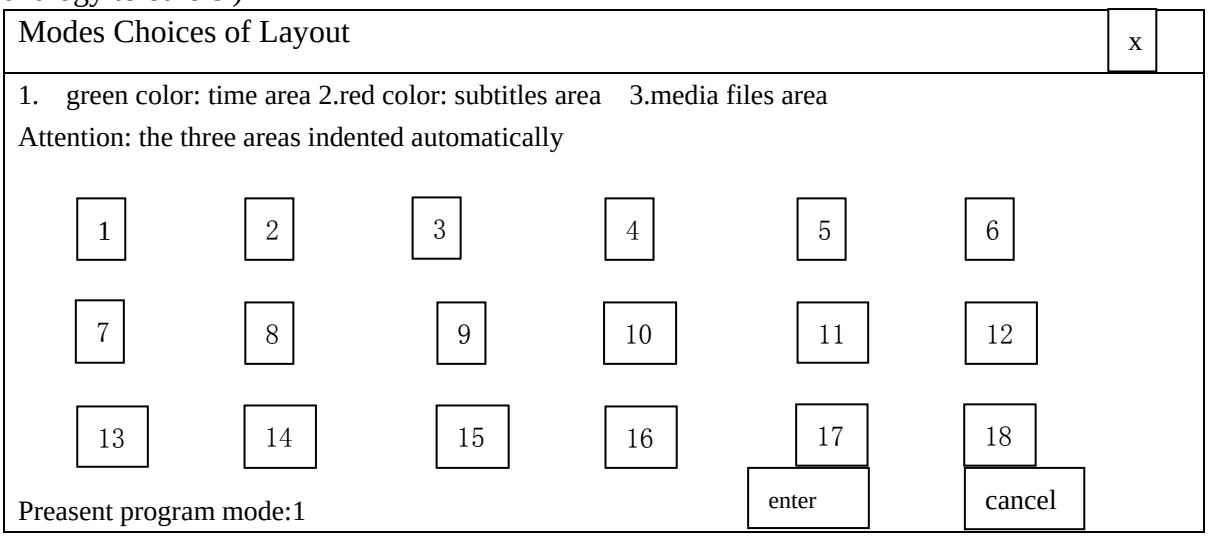
UINT uiCheckCode; // 32 bits check sum from the beginning to the ending mark.(including the beginning and the ending marks.)

}

```
typedef  struct     _AREA_FILL_COLOR                    // area-fill-color
{
          UINT          uiFillType;//area-fill-type:SCREEN_BACK_GROUND_TYPE
           UINT              uiColorValue;              //     Value.
           UINT              ucRed;                     // Rgb of Red.
           UINT              ucGreen;                   // Rgb of Green.
           UINT              ucBlue;                 // Rgb of Blue.
           CHAR                        pFileName[FILE_NAME_SIZE];  // Jpeg File Name.
     }AREA_FILL_COLOR,* P_AREA_FILL_COLOR;
enum SCREEN_BACK_GROUND_TYPE
{
           SCREEN_BACK_GROUND_NO,          //without background
           SCREEN_BACK_GROUND_VALUE,
           SCREEN_BACK_GROUND_RGB,
           SCREEN_BACK_GROUND_JPEG,
           SCREEN_BACK_GROUND_NUM

};
```

// area modes types are showed in the following picture(0: mode1,1: mode 2, and by analogy to others)



The corresponding Chinese one of the pictue above is as follows:



2.1.3 TimeFomat Detailed Explanation

```
// format modification of timepiece:
typedef struct _ST_TIMER//time format
{
    unsigned char ucType;
    unsigned char ucPos;
    short int ucFormate;
//  unsigned char ucBaoliu;
    UINT        uiDateType;     // date type
    UINT        uiWeekType;     // week type
    UINT        uiTimeType;     // time type
    UINT        uiTempratureType;// temperature type
    UINT        uiHumidity;
//20071022  UINT        uiReserved[5];   // reserved
    st_Wnd        st_Timer;  // PC reserved
    UINT uiReserved;

    ST_FONT m_st_font;
}ST_TIMER,*P_ST_TIMER;
struct st_Wnd
{
    UINT left;
    UINT top;
    UINT width;
    UINT height;
};
typedef struct _ST_FONT{
```

```
    char cFontName[FILE_NAME_SIZE];//Character name
//  int  type;//
    int size;//size
    int iB;//Boldface
    int iI;//italic
    int iU;//underline
    COLORREFbackcolor;//background color
    COLORREFcolor;//foreground color
}ST_FONT,*P_ST_FONT;

//ucType:
enum    _CLOCK_TYPE
{       CLOCK_TYPE_DIGITAL                      ,
        CLOCK_TYPE_ANALOG                       ,
        CLOCK_TYPE_NUM
};

//ucFormat:
enum    _CLOCK_DETAIL
{       CLOCK_DETAIL_TEMPRATURE                 ,
        CLOCK_DETAIL_WEEK                       ,
        CLOCK_DETAIL_SECOND                     ,
        CLOCK_DETAIL_MINUTE                     ,
        CLOCK_DETAIL_HOUR                       ,
        CLOCK_DETAIL_DAY                    ,
        CLOCK_DETAIL_MONTH                      ,
        CLOCK_DETAIL_YEAR                       ,
        CLOCK_DETAIL_NUM
};

//uiDateType(date type):
enum    _CLOCK_DATE_TYPE
{       CLOCK_DATE_TYPE_CN_YEAR_MONTH_DAY,  // 2006/Jan/1st
        CLOCK_DATE_TYPE_CN_SLANT        ,   // 2006/01/31.
        CLOCK_DATE_TYPE_CN_DEC          ,   // 2006-01-31.
        CLOCK_DATE_TYPE_CN_DOT          ,   // 2006.01.31.
        CLOCK_DATE_TYPE_EN_SLANT        ,   // 01/31/2006.
        CLOCK_DATE_TYPE_EN_DEC          ,   // 01-31-2006.
        CLOCK_DATE_TYPE_EN_DOT          ,   // 01.31.2006.
        CLOCK_DATE_TYPE_NUM
};

//uiWeekType(week type):
enum    _CLOCK_WEEK_TYPE
```

```
{       CLOCK_WEEK_TYPE_CN_WEEK                 ,   // Sunday.
        CLOCK_WEEK_TYPE_CN_ZHOU                 ,   // Sunday.
        CLOCK_WEEK_TYPE_EN_SUNDAY               ,   // Sunday.
        CLOCK_WEEK_TYPE_EN_SUN                  ,   // Sunday
        CLOCK_WEEK_TYPE_NUM
};

//uiTimeType(Time Type):
enum    _CLOCK_CLOCK_TYPE
{       CLOCK_CLOCK_TYPE_CN_HOUR_MINUTE     ,   //01:01:01 pm
        CLOCK_CLOCK_TYPE_COLON              ,   // 01:01:01.pm
        CLOCK_CLOCK_TYPE_12_HOUR_COLON      ,   // 01:01:01 pm.
        CLOCK_CLOCK_TYPE_NUM
};

//uiTempratureType(temperature type):
enum    _CLOCK_TEMPRATURE_TYPE
{       CLOCK_TEMPRATURE_TYPE_CENTIGRADE        ,   // 24'C.
        CLOCK_TEMPRATURE_TYPE_FAHRENHEIT        ,   // 24'F.
        CLOCK_TEMPRATURE_TYPE_CN_CENTIGRADE     ,   // 24'C.
        CLOCK_TEMPRATURE_TYPE_CN_FAHRENHEIT     ,   //24'F.
        CLOCK_TEMPRATURE_TYPE_NUM
};


// File name constant index:
enum    _CLOCK_PICTURE
{       CLOCK_PICTURE_TEMPRATURE            ,   // "'C".
        CLOCK_PICTURE_WEEK                  ,   // "week".
        CLOCK_PICTURE_SECOND                ,   // "second".
        CLOCK_PICTURE_MINUTE                ,   // "minute".
        CLOCK_PICTURE_HOUR                  ,   // "hour".
        CLOCK_PICTURE_DAY                   ,   // "day".
        CLOCK_PICTURE_MONTH                 ,   // "month".
        CLOCK_PICTURE_YEAR                  ,   // "year".
        CLOCK_PICTURE_SEVEN                 ,   // "Sunday".
        CLOCK_PICTURE_ONE                   ,   // "Monday".
        CLOCK_PICTURE_TWO                   ,   // "Tuesday".
        CLOCK_PICTURE_THREE                 ,   // "Wednesday".
        CLOCK_PICTURE_FOUR                  ,   // "Thursday".
        CLOCK_PICTURE_FIVE                  ,   // "Friday".
        CLOCK_PICTURE_SIX                   ,   // "Saturday".
        CLOCK_PICTURE_0                     ,   // "0".
        CLOCK_PICTURE_1                     ,   // "1".
```

```
    CLOCK_PICTURE_2                         ,   // "2".
    CLOCK_PICTURE_3                         ,   // "3".
    CLOCK_PICTURE_4                         ,   // "4".
    CLOCK_PICTURE_5                         ,   // "5".
    CLOCK_PICTURE_6                         ,   // "6".
    CLOCK_PICTURE_7                         ,   // "7".
    CLOCK_PICTURE_8                         ,   // "8".
    CLOCK_PICTURE_9                         ,   // "9".
    CLOCK_PICTURE_SPACE                     ,   // " ".
    CLOCK_PICTURE_SLANT                     ,   // "/".
    CLOCK_PICTURE_DEC                       ,   // "-".
    CLOCK_PICTURE_DOT                       ,   // ".".
    CLOCK_PICTURE_COLON                     ,   // ":".
    CLOCK_PICTURE_AM                    ,   // "AM".
    CLOCK_PICTURE_PM                    ,   // "PM".
    CLOCK_PICTURE_DU                    ,   // "Degree".
    CLOCK_PICTURE_CENTIGRADE                ,   // "Centigrade".
    CLOCK_PICTURE_FAHRENHEIT                ,   // "Fahrenheit".
    CLOCK_PICTURE_TEMPRATURE_FAHRENHEIT         ,   // "'F".
    CLOCK_PICTURE_ZHOU                      ,   // "week".
    CLOCK_PICTURE_SUN                       ,   // "SUN".
    CLOCK_PICTURE_MON                       ,   // "MON".
    CLOCK_PICTURE_TUE                       ,   // "TUE".
    CLOCK_PICTURE_WED                       ,   // "WED".
    CLOCK_PICTURE_THE                       ,   // "THUR".
    CLOCK_PICTURE_FRI                   ,   // "FRI".
    CLOCK_PICTURE_SAT                   ,   // "SAT".
    CLOCK_PICTURE_SUNDAY                    ,   // "Sunday".
    CLOCK_PICTURE_MONDAY                    ,   // "Monday".
    CLOCK_PICTURE_TUESDAY                   ,   // "Tuesday".
    CLOCK_PICTURE_WEDNESDAY                     ,   // "Wednesday".
    CLOCK_PICTURE_THURSDAY                  ,   // "ThursDay".
    CLOCK_PICTURE_FRIDAY                ,   // "Friday".
    CLOCK_PICTURE_SATURDAY                  ,   // "Saturday".

    CLOCK_PICTURE_NUM
};
```

File name *.bmp index: (The corresponding picture names of the time elements stored on the disk)

```
static      P_STRING  ClockPictureName[CLOCK_PICTURE_NUM]=
                    {   "temprature",   // "'C".
                        "week"      ,   // "week".
                        "second"    ,   // "second".
                        "minute"    ,   // "minute".
```

```
"hour"        ,     // "hour".
"day"         ,     // "day".
"month"       ,     // "month".
"year"        ,     // "year".
"seven"       ,     // "sunday".
"one"         ,     // "Monday".
"two"         ,     // "Tuesday".
"three"       ,     // "Wednesday".
"four"        ,     // "Thursday".
"five"        ,     // "Friday".
"six"         ,     // "Saturday".
"0"           ,     // "0".
"1"           ,     // "1".
"2"           ,     // "2".
"3"           ,     // "3".
"4"           ,     // "4".
"5"           ,     // "5".
"6"           ,     // "6".
"7"           ,     // "7".
"8"           ,     // "8".
"9"           ,     // "9".
"space"       ,     // " ".
"slant"       ,     // "/".
"dec"         ,     // "-".
"dot"         ,     // ".".
"colon"       ,     // ":".
"AM"          ,     // "AM".
"PM"          ,     // "PM".
"du"          ,     // "degree".
"centigrade",// "centigrade".
"fahrenheit",// "Fahrenheit".
"tempratureF",    // "'F".
"zhou"        ,     // "week".
"SUN"         ,     // "SUN".
"MON"         ,     // "MON".
"TUE"         ,     // "TUE".
"WED"         ,     // "WED".
"THE"         ,     // "THE".
"FRI"         ,     // "FRI".
"SAT"         ,     // "SAT".
"Sunday"      ,     // "Sunday".
"Monday"      ,     // "Monday".
"Tuesday"     ,     // "Tuesday".
"Wednesday"       ,      // "Wednesday".
```

```
        "Thursday"   ,// "Thursday".
        "Friday",      // "Friday".
        "Saturday",// "Saturday".
};
```

2.2.Subtitles Detailed Explanation

The subtitle (which is a kind of dispay with words rolling and moving on it) is a pictue(the width of the picture is the width of characters, the height of the picture is the height of the charaters). The file name is: “FrontMessage.bmp tyeded struct_ST_FONTSCREEN”// subtitle structure.

```
{
    ST_PAOMADENG m_st_PaoMaDeng;
    char FontContentFileName[FILE_NAME_NODIR_LENGTH];
}ST_FONTSCREEN,*P_ST_FONTSCREEN;
typedef struct _ST_PAOMADENG
{
unsigned char ucType;//types 2:static，0:leftward，1: upward
unsigned char ucPos;//1:centered，0:leftward，2:rightward
unsigned char ucShuDu;//moving speed, millisecond as the unit
unsigned char ucShutPaoMaDeng;// shut off subtitle display while ending the media area
//20071022   UINT                uiReserve[10];//061121
    st_Wnd     st_PMD;//PC reserved
    UINT uiReserve[6];
    ST_FONT m_st_font;
}ST_PAOMADENG,*P_ST_PAOMADENG;
struct st_Wnd
{
    UINT left;
    UINT top;
    UINT width;
    UINT height;
};
typedef struct _ST_FONT{
    char cFontName[FILE_NAME_SIZE];//FrontName
//    int   type;//
    int size;//size
    int iB;//Boldface
    int iI;//Italic
    int iU;//underline
    COLORREFbackcolor;//background color
    COLORREFcolor;//foreground color
}ST_FONT,*P_ST_FONT;
```

2.3.Detailed Explanation for Playing List in Media Area

```
//Media Area Structure
//JPEG picture parameters in program playing list
        typedef  struct     _JPEG_STUNT
{ CHAR      pStageTypeStart[PLAY_LIST_TYPE_SIZE];   //beginning mark: see the
   definition of FILE_TYPE_JPEG.
  UINT uiLoopbackNum;      // circulated times,0: infinite loop.
  UINT uiStuntComeType;// pictures  will  enter  into  the  stunt  state.Please  see  the
 reference3.5.
  UINT uiStuntGoType;  // going to stunt type.see reference3.5
  UINT uiComeSpeed;//coming speed
  UINT uiGoSpeed;//going speed
  UINT uiSettleTime;              // the pictures settling time(second)
  UINT uiDispMode;                // the images displaying effect
  UINT uiLayer;                   // images layers.*
  UINT      uiReserve[10];//061121
  CHAR      pFileOrPathName[FILE_NAME_SIZE];//image file name or folder name
  CHAR      pFileOrPathEnglishName[FILE_NAME_SIZE];//ALAHOVER 1227
  CHAR      pStageTypeEnd[PLAY_LIST_TYPE_SIZE];     // ending mark.
    }JPEG_STUNT,* P_JPEG_STUNT;
    typedef  struct     _BMP_STUNT
    {
        CHAR  pStageTypeStart[PLAY_LIST_TYPE_SIZE];   //beginning mark: see
the definition of FILE_TYPE_JPEG.
        UINT    uiLoopbackNum;              // circulated times,0: infinite loop.
        UINT    uiStuntComeType;          // going to stunt type.
        UINT    uiStuntGoType;              // exit stunt type.
        UINT    uiComeSpeed;//coming speed
        UINT    uiGoSpeed;//going speed
        UINT    uiSettleTime;               //the pictures settling time(second)
        UINT    uiDispMode;                // the images displaying effect
        UINT    uiLayer;                  // images layers.*
        UINT      uiReserve[10];//061121
        CHAR  pFileOrPathName[FILE_NAME_SIZE];// image file name or folder
name

        CHAR  pFileOrPathEnglishName[FILE_NAME_SIZE];//ALAHOVER 1227
        CHAR  pStageTypeEnd[PLAY_LIST_TYPE_SIZE];     // ending mark.
    }BMP_STUNT,* P_BMP_STUNT;
    typedef  struct     _GIF_STUNT
    {
        CHAR  pStageTypeStart[PLAY_LIST_TYPE_SIZE];   //beginning mark: see
        the definition of FILE_TYPE_JPEG.
        UINT    uiLoopbackNum;              //circulated times,0:infinite loop.
```

```
        UINT    uiStuntComeType;            // coming to stunt type.
        UINT    uiStuntGoType;              // exit stunt type.
        UINT    uiComeSpeed;                //coming speed
        UINT    uiGoSpeed;                  // going speed
        UINT    uiSettleTime;               // the pictures settling time(second)


        UINT    uiDispMode;                     // the images displaying effect
        UINT    uiLayer;                // images layers.*
        UINT      uiReserve[10];//061121
        CHAR   pFileOrPathName[FILE_NAME_SIZE];// image file name or folder
name
        CHAR   pFileOrPathEnglishName[FILE_NAME_SIZE];//ALAHOVER 1227
        CHAR   pStageTypeEnd[PLAY_LIST_TYPE_SIZE];    // ending mark.
    }GIF_STUNT,* P_GIF_STUNT;
    typedef  struct    _VODIO_STUNT
    {
        CHAR   pStageTypeStart[PLAY_LIST_TYPE_SIZE];   // beginning mark: see
the definition of FILE_TYPE_JPEG.
        UINT    uiLoopbackNum;                  // circulated times,0: infinite loop.
        UINT    uiColorDepth;               // coming to stunt.
        UINT    reserve[6];             // exit stunt.
        UINT      uiReserve[10];//061121
        CHAR   pFileOrPathName[FILE_NAME_SIZE];// image file name or folder
name.
        CHAR   pFileOrPathEnglishName[FILE_NAME_SIZE];//ALAHOVER 1227
        CHAR   pStageTypeEnd[PLAY_LIST_TYPE_SIZE];    // ending mark.
    }VODIO_STUNT,* P_VODIO_STUNT;
    typedef  struct    _SWF_STUNT
    {
        CHAR   pStageTypeStart[PLAY_LIST_TYPE_SIZE];   // beginning mark:see
the definition of FILE_TYPE_JPEG.
        UINT    uiLoopbackNum;                  // circulated times,0: infinite loop.
        UINT    uiStuntComeType;                // coming to stunt.
        UINT    uiStuntGoType;              //exit stunt.
        UINT    uiComeSpeed;//coming speed
        UINT    uiGoSpeed;//going speed
        UINT    uiSettleTime;               //the pictures settling time(second)

        UINT    uiDispMode;                     // the images displaying effect.
        UINT    uiLayer;                // images layers.*
        UINT      uiReserve[10];//061121
        CHAR   pFileOrPathName[FILE_NAME_SIZE];// image file name or folder
name.
```

```
        CHAR   pFileOrPathEnglishName[FILE_NAME_SIZE];//ALAHOVER 1227
        CHAR   pStageTypeEnd[PLAY_LIST_TYPE_SIZE];     // ending mark.
    }SWF_STUNT,* P_SWF_STUNT;
    typedef  struct     _TXT_STUNT
    {    CHAR   pStageTypeStart[PLAY_LIST_TYPE_SIZE];   //beginning mark: see the definition of FILE_TYPE_JPEG.
        UINT    uiLoopbackNum;                //circulated times,0:infinite loop.
        UINT    uiColorDepth;                 // coming to stunt type.
        UINT    reserve[6];                   // exit stunt.
        UINT      uiReserve[10];//061121
        CHAR   pFileOrPathName[FILE_NAME_SIZE];//image file name or folder name.
        CHAR   pFileOrPathEnglishName[FILE_NAME_SIZE];//ALAHOVER 1227
        CHAR   pStageTypeEnd[PLAY_LIST_TYPE_SIZE];     // ending mark.
    }TXT_STUNT,* P_TXT_STUNT;
    typedef  struct     _RTF_STUNT
    {    CHAR   pStageTypeStart[PLAY_LIST_TYPE_SIZE];   // beginning mark: see the definition of FILE_TYPE_JPEG.
        UINT    uiLoopbackNum;                  // circulated times,0: infinite loop.
        UINT    uiStuntComeType;                // coming to stunt type.
UINT    uiStuntGoType;                  // exit stunt type.
        UINT    uiComeSpeed;//coming speed
        UINT    uiGoSpeed;//going speed
        UINT    uiSettleTime;           // the pictures settling time(second)
         UINT   uiDispMode;                     // the images displaying effect.
        UINT    uiLayer;                    // images layers.*
        UINT      uiReserve[10];//061121
        CHAR   pFileOrPathName[FILE_NAME_SIZE];// image file name or folder name.
        CHAR   pFileOrPathEnglishName[FILE_NAME_SIZE];//ALAHOVER 1227
        CHAR   pStageTypeEnd[PLAY_LIST_TYPE_SIZE];     // ending mark.
    }RTF_STUNT,* P_RTF_STUNT;
    typedef  struct     _GRID_STUNT
    {    CHAR   pStageTypeStart[PLAY_LIST_TYPE_SIZE];   //beginning mark: see the definition of FILE_TYPE_JPEG.
        UINT    uiLoopbackNum;                  // circulated times,0: infinite loop.
        UINT    uiStuntComeType;                // coming to stunt type.
UINT    uiStuntGoType;                    //exit stunt type.
        UINT    uiComeSpeed;                //coming speed
        UINT    uiGoSpeed;                  //going speed
        UINT    uiSettleTime;               //the pictures settling time(second).

        UINT    uiDispMode;                 // the images displaying effect.
        UINT    uiLayer;                    // images layers.*
```

```
        UINT    uiReserve[10];//061121
        CHAR  pFileOrPathName[FILE_NAME_SIZE];// image file name or folder name.
        CHAR  pFileOrPathEnglishName[FILE_NAME_SIZE];//ALAHOVER 1227
        CHAR  pStageTypeEnd[PLAY_LIST_TYPE_SIZE];    // ending mark.
    }GRID_STUNT,* P_GRID_STUNT;

    typedef struct _RTF_PLAY_STUNT
    {   CHAR  pStageTypeStart[PLAY_LIST_TYPE_SIZE];//ending mark: please see the definition of CONFIG_FILE_TYPE_START.
        COLORREF    crBackgroundColor;
        UINT    uiReserve[9];
        UINT    uiCount;
        RTF_STUNT    rtf_st[MAX_RTF_PIC_NUMBER];
        CHAR  pStageTypeEnd[PLAY_LIST_TYPE_SIZE];    // ending mark.
        UINT  uiCheckCode;  //The 32 bits check sum from the beginning to the end(including the beginning and the ending mark)
    }RTF_PLAY_STUNT,P_RTF_PLAY_STUNT;
④Timing Playing Configuration Files
 File Names: tcmd.sys   // in the catalogue:/Area0/tcmd.sys
 Timing Playing List
 UiTimerInstructEventType //   the corresponding value meaning
 enum ENUM_TIMER_INSTRUCTION
 {
    ENUM_TIMER_PANEL_POWER_ON=1, // turn on the display power
    ENUM_TIMER_PANEL_POWER_OFF, // turn off the display power
    ENUM_TIMER_PANEL_ADJUST_LIGHT, // adjust the display brightness
    ENUM_TIMER_PANEL_ADJUST_VOLUME, // adjust the volume
    ENUM_TIMER_PANEL_PLAY_PROGRAM=100// play program
 };

 typedef struct   _STRUCT_TIMER_INSTRUCTION_FORMAT
 {  CHAR       pStageTypeStart[PLAY_LIST_TYPE_SIZE];
     UINT m_uiTimerInstructEventType;// default:0xffffffff
    char m_cTimerProgramName[20]; //Character string
    char m_cTimerProgramEnglishName[20]; //Character string
    UINT m_uiValue;//default value:0xffffffff
    UINT m_uiStartTime;// default:0xffffffff
                                    //0xff0000/*hour the third byte of the unit*/
                                    //0xff00/*minute the second byte of the unit*/
                                    //0xff/*second the first byte of the unit*/
                                    //example:11pm 12 minutes 13 seconds，value:0x0B0C0D
    UINT m_uiEnableDataStart; //default:0xffffffff
```

//0xffff00/*year*/
//0xff00/*month*/
//0xff/*day*/
//example:2007-12-13, then the value is: 0x07D70C0D

UINT m_uiEnableDataEnd; //default:0xffffffff
//0xffff00/*year*/
//0xff00/*month*/
//0xff/*day*/
//example:2007-12-13 then the value is:0x07D70C0D

UINT m_uiEnableWeek;//default:0xffffffff
//0x1000000/*Saturday*，
//0x100000/*Friday*/，
//0x10000/*Thursday*/，
//0x1000/*Wednesday*/，
//0x100/*Tuesday*/，
//0x10/*Monday*/，
//0x1/*Sunday*/
// Example: if Monday,Wednesday and Sunday is valid, then the value should be: 0x00001011

UINT m_uiReserve[8];
CHAR pStageTypeEnd[PLAY_LIST_TYPE_SIZE];//"END"
}

Attention：the value:0xffffffff stands for nullity

STRUCT_TIMER_INSTRUCTION_FORMAT,*P_STRUCT_TIMER_INSTRUCTION_FORMAT;

typedef struct _STRUCT_TIME_INSTRUCTION_TABLE
{
CHAR pStageTypeStart[PLAY_LIST_TYPE_SIZE];//"LED"
UINT uiUseFlag;//1/* start the timing clock*/，0/*not to start*/
UINT reserve[7];//reserved
STRUCT_TIMER_INSTRUCTION_FORMAT st_Timer_instruction_format[…]
CHAR pStageTypeEnd[PLAY_LIST_TYPE_SIZE];//"END"
UINT uiCheckCode;// verifying code
}STRUCT_TIME_INSTRUCTION_TABLE,*P_STRUCT_TIME_INSTRUCTION_TABLE;

⑤Background music configuration files

typedef struct _AUDIO_STUNT
{ CHAR pStageTypeStart[PLAY_LIST_TYPE_SIZE]; //Beginning mark: see the definition of FILE_TYPE_JPEG.
UINT uiLoopbackNum; // circulated times,0: infinite loop.
UINT reserve[7]; // exit stunt.
UINT uiReserve[10];//061121
CHAR pFileOrPathName[FILE_NAME_SIZE];// image file name or folder name.

```
    CHAR pFileOrPathEnglishName[FILE_NAME_SIZE];// image file name or
    folder name（English）
    CHAR pStageTypeEnd[PLAY_LIST_TYPE_SIZE];    // Ending Mark.
}AUDIO_STUNT,* P_AUDIO_STUNT;
typedef struct _STRUCT_AUDIO_LIST
{
    CHAR pStageTypeStart[PLAY_LIST_TYPE_SIZE];    // Starting Mark: see
    the definition of FILE_TYPE_JPEG.
    UINT uiCount;
    AUDIO_STUNT struct_Audio[32];
    CHAR pStageTypeEnd[PLAY_LIST_TYPE_SIZE];      // Ending Mark.
    UINT uiCheckCode;
}STRUCT_AUDIO_LIST;
```

2.4 Configuration Files Function

2.4.1 Screen Configuration Files Function

C languge DLL： int WriteScreenConfig（
char* cScreenConfigPath,//// The routes of screen configuration files
unsigned int uiAreaLoopNum, // it must be 0 for the moment now, can not halt
unsigned int uiAreaNum, ////it must be 1 for the moment now
char(&AreaNameArr)[16][100] //////it must be Area0 for the moment now
）

Function：
Store the files to the disk according to the formats of the screen configuration files.If the route didn't exist,function will create the routes by itself.
Return Value:

2.4.2 To Write Area Configuration Files

2.4.2 To Write Screen Configuration Files

C Language DLL： int WriteAreaConfig(
char* cAreaConfigPath, /// Area configuration files path
unsigned int xStartArea, //.The starting point coordinate of area Xward
unsigned int yStartArea, // The starting point coordinate of area Yward
unsigned int xLengthArea, // Length of the area xward
unsigned int yLengthArea, // Length of the area yward
unsigned int FrameWidth,// Area frame width（pixel），0：no frames
unsigned int SolidWidth, // Area solid width of vision effect（pixels），0：no solid effect
COLORREF colorFrame, // Area frame color definition

char* cPicJpgFrame, // Frame background pictures file name. If blank,then the area background should use “colorFrame” to fill in the color.
COLORREF colorBkGround, // Area background color definition.
char* cPicJpgBkGround,// Area background pictures file name. If blank, then the area background should use the colorBkGround to fill in the color.
int iLoopNum, // circulated times,0: infinite loop.
char (&cProjName)[20][100] // program name
)
Function：
Store the files to the disk according to the formats of the screen configuration files.If the route didn't exist,function will create the routes by itself.
Return Value:
Return Value:

2.4.3 To Add Programs

Before writing the configuration files functions of the playing list, the following functions should be used first to add programs, then the files of playing list can be written.

2.4.3.1 Add Picture Stunt

C Language DLL： int AddPicStuntList （
char* cPicName, // picture file name, the full path of the pictures on computer.
int iLoopbackNum, // the pictures’ displaying times, 0: infinite loop, which means the screen will display the pictures in circles all the time..
int uiStuntComeType, // Pictures coming to the stunt state,see the following stunt list.
int uiStuntGoType, // Stunt while pictures exiting the screen, see the following stunt list.
int uiComeSpeed, //coming speed, unit: second
int uiGoSpeed, // going speed, unit: second
int uiSettleTime, // the pictures settling time(unit: second)
int uiDispMode// pictues displaying effect, see the following displaying effect list.
）
Function：
Add the picture stunt to the playing list.
Reture Value：

2.4.3.2 Add Video

2.4.4 Write the Configuration Files of Playing List

C Language DLL： int WritePlayList（
char* cPlayListPath /// Playing list path
）

Function：
To writeh the programs which has been added already to the playing list of the configuration files.
Return Value：

2.5 Operation Function of Files and Folders

CreateFolder(char* cPath)
Function：
To create the appointed path and the corresponding folders according to “cPath”
CopyTreeOrFile(
char* cSourcePathOrFile,
char* cDestPathOrFile
)
Function：
Copy all the appointed documents of the files or folders to the appointed path.

MoveTreeOrFile(char* cSourcePathOrFile,char* cDestPathOrFile)
Function：
Move all the documents of the appointed files or folders to the appointed path.
DelTreeOrFile(char* cPathOrFile)
Function：
Delete all the documents of the appointed files or folders.

Three Network Protocol Format

3.1. Situations for Protocol Application
This protocol can be applicated to network communication occasions between Computer and LED Display Controller(hereafter call cotroller for short).It can realize the point to point communication, and can also realize the team network communication, which allows to connect into a local area network or other network equipments to allow one or multi-sets controllers to work together in a team communication.
3.2. Rules for Protocol Application
3.2.1 Computers use UDP/TCP Transport Layer Protocols to communicate with LED controller. Controller can be compatible to both protocols, both of which can receive the datas sent by computers. Users need to use a standard socket interface to programme: UDP port is 31296. It is a kind of general data communications, which can be broadcast, therefore,not a kind of reliable data transmission; TCP port is 31298,which can only be used through point-to-point communication and can not be broadcast. Then in the computer system,controller works as a service, a computer or other equipments as a client, more than one clients can be connected to communicaiton at the same time;

3.2.2 During the application of this protocol, when it is necessary to use hex datas, then the datas must begin with a mark “0x”, in order to tell the hex from the decimal datas. Eg: 0xFF in hex stands for 255 in algorism

3.2.3 Multibyte data type use little endian justified mode. Eg: as for the unsigned integer type “0x1234 5678”， the datas addresses in EMS memory from low to high are: 0x78、0x56、0x34、0x12，then in the communication data package, it will be also of the same order to transmit datas out, first to transmit 0x78, then 0x56, 0x34 and lastly 0x12;

3.2.4. To store datas of character string type must end up with the mark “0x00”,the effective length of character string should also plus one. Eg: as for the character string “S200-507-00888” ,the effective length shoud be 15, thelast byte is 0x00;

3.2.5.The contents whithin the inverted comma means the the character belonging to ASCII code. Eg: ‘A’ stands for ASCII code of the letter A, 65 in decimal system, 0x41 in hex.

3.2.6 In the detailed explanation of the protocol, the verification code of the demonstration use the mard “??” instead, which is also decided by the concrete protocol.

3.3 Protocol Structure

Protocal are composed by Sync head, command type, controller’s sequence number, Team number, length of the text, text and verification seven parts, which are listed as Follows:

Sync Head	Tata type	Controller Sequence No.	Team No.	Text Length	Text	Verification No.
3bytes	1byte	16bytes	4bytes	4bytes	（text length）	1byte

Sync Head: 3bytes; unsigned character type; Beginning of data package; Its value are three continuous fixed value: 0x7E 0x7E 0x7E;

Data Type: 1 byte; unsigned character type;The command type of data package; Please see the detailed explanation later in this handbook.

Controller Sequence No.:16 bytes; Character string type; The sequence No. can only be offered by the factory while products being dispatched from the factory. If less than 16 bytes, then the sequence number must use 0x00 to make up;It can be got from the network window of the computer program compilation software, or also can get it from the protocol responsion bag; If the command is broadcast type (computer send data commands to all controllers at the same time),then the sequence No. is “broadcast!”; Eg: to send point to point datas to the controller whose sequence number is “S200-507-00888”, then the controller’s responsion is also “S200-507-00888”.

Team No.: 4 bytes; Unsigned integer type ; When the data package is “broadcast type”, all the controllers then can be controlled in different teams; Only data package whose team number is the same with the controllers own sequence number, can controller accept it. But for the team No. “0x0000 0000”,it makes no difference between team numbers, all controllers can accept it. In the computer basic parameters of program compilation software, the team number of controllers can be settled.

Text Length: 4 bytes; unsigned integer type; (value range: 0x0000 0000~0x0000 07D0)(Applicated in Word2000). The value is the effective data bytes of text.

Text: the bytes number is decided by the text length; the types are introduced in a detailed way later in this handbook.

Verification No.:1 byte, unsigned character type; used to check the validity during receiving data package; The value is the sum of all the bytes from the beginning “data type” to the ending of the “text”.

3.4. Common Response of Controller Protocol

If without extra explanation,controller will give out “success” or ”failed” as its response. The concrete definition is listed as follows:

Data type: all the data type of commands sent out by computer should plus “0x40”. Eg: if the computer sent out the data type command “0x80”, then the controller should response 0xC0. (reference: 0x80+0x40=0xc0)
Text Length: it is decided by the total number of Chinese bytes.
Text: character string type; successful operation: “success!”; defeated operation: “Error!”.

3.5. Command Types Enumerating

```
enum _NET_COMMU_TYPE
{
    COMMAND_UPGRADE = 0x80,// upgraded command
    COMMAND_SIGNLE_FILE_UPGRADE ,// single file upgraded command
    COMMAND_LOOPCHECK ,// control card loopcheck command
    COMMAND_VERIFY_CLOCK ,// clock verifying command
    COMMAND_SCREEN_OPERATOR,// screen parameter operational command
    COMMAND_SET_SERIAL_NUM,// com number command setting
    NET_COMMU_TYPE_ RESERVE1,//reserved
    NET_COMMU_TYPE_ RESERVE2,//reserved
    NET_COMMU_TYPE_ RESERVE3,//reserved
    NET_COMMU_TYPE_ RESERVE4,//reserved
    SCREEN_OPERATOR_COMM_PARAMS_SAVE,// parameter saveing command
    NET_COMMU_TYPE_COMM_EXIT,// system exit command
    NET_COMMU_TYPE_AUDIO_PLAY,//0X8C // audio playing command
    NET_COMMU_TYPE_IO,// IO operational command
    NET_COMMU_TYPE_IO_CHECK,//IO state checking
    NET_COMMU_TYPE_LIMITE_TIME,// restricting command
    NET_COMMU_TYPE_SYSTEM_BROADCAST,// Broadcasting short package state checking commands
    NET_COMMU_TYPE_REALTIME_INSERT_PLAY,// Realtime inserted playing command
    NET_COMMU_TYPE_SHANSUO_INSERT_PLAY,// flickering area commands setting
    NET_COMMU_TYPE_RESERVE5,//reserved
    NET_COMMU_TYPE_RESERVE6,//reserved
    NET_COMMU_TYPE_NUM,
    //response，command type: +0x40
```

COMMAND_UPGRADE_ANSWER =0xC0,
COMMAND_SIGNLE_FILE_UPGRADE_ANSWER ,
COMMAND_LOOPCHECK_ANSWER ,
COMMAND_VERIFY_CLOCK_ANSWER ,
COMMAND_SCREEN_OPERATOR_ANSWER,
COMMAND_SET_SERIAL_NUM_ANSWER,
NET_COMMU_TYPE_RESERVE5_ANSWER1, // Network synchronizational playing.
NET_COMMU_TYPE_RESERVE5_ANSWER2, // Network online preview
NET_COMMU_TYPE_RESERVE5_ANSWER3, // System upgraded
NET_COMMU_TYPE_RESERVE5_ANSWER4, // The right way of start-up to revert to the last time.
SCREEN_OPERATOR_COMM_PARAMS_SAVE_ANSWER,
NET_COMMU_TYPE_COMM_EXIT_ANSWER,
NET_COMMU_TYPE_AUDIO_PLAY_ANSWER,//0XcC
NET_COMMU_TYPE_IO_ANSWER,
NET_COMMU_TYPE_IO_CHECK_ANSWER,
NET_COMMU_TYPE_LIMITE_TIME_ANSWER,
NET_COMMU_TYPE_SYSTEM_BROADCAST_ANSWER,
NET_COMMU_TYPE_REALTIME_INSERT_PLAY_ANSWER,
NET_COMMU_TYPE_SHANSUO_INSERT_PLAY_ANSWER,
NET_COMMU_TYPE_RESERVE5_ANSWER,//reserved
NET_COMMU_TYPE_RESERVE6_ANSWER,//reserved
NET_COMMU_TYPE_NUM_ANSWER,
COMMAND_CONNECT_COLSE_ANSWER

IV.Chapeter Four Communication Controlling Funciton

§4.0 Error Checkout Function

int **GetDLLErr**(CString& strError);

Function:

Put the recent ten-error responses into the parameter strError.

Return Value:

1:already put the error responses into the parameter “strError”, and already clear empty the error responses in the DLL.

0:No error responses in the DLL recently.

§4.1 Control Card Checking Function

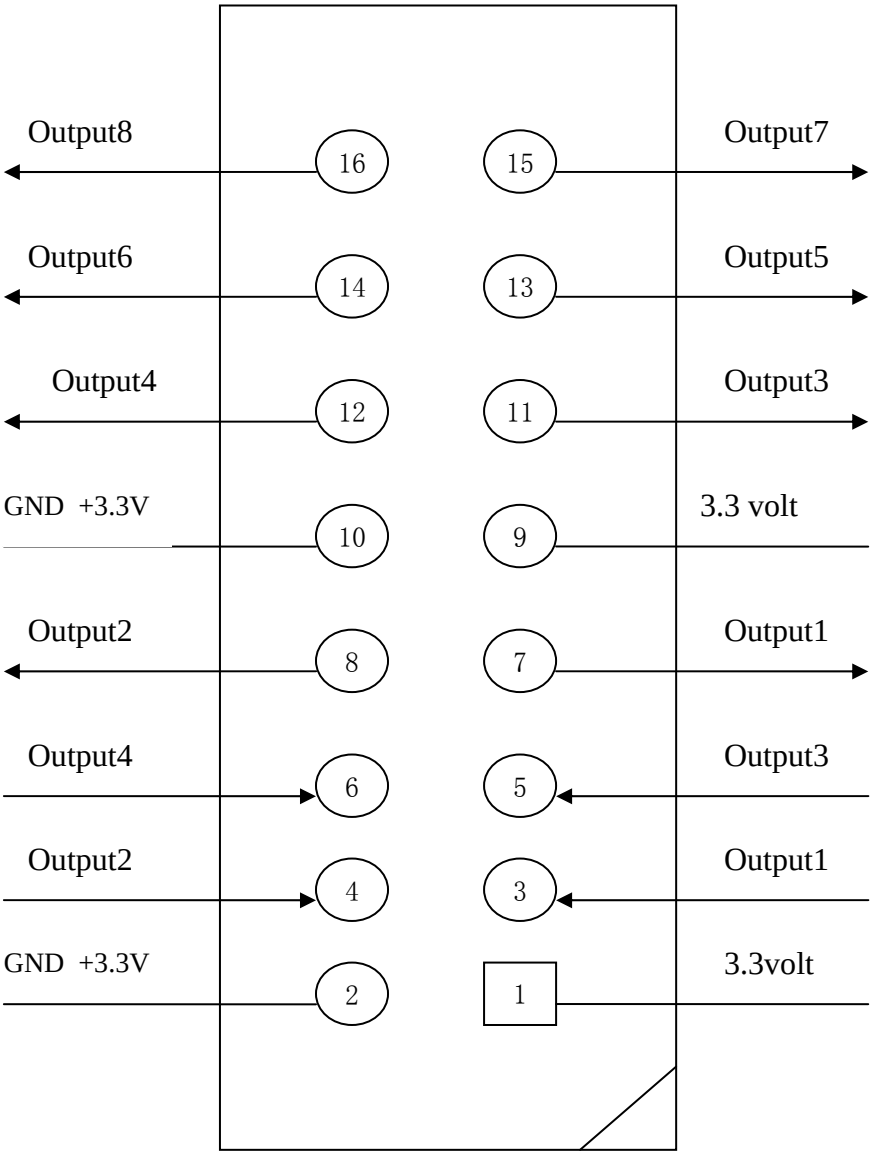
This function is used to checkout all control cards in LAN and to get back the sequence number and IP address of the control cards. It is composed by “LoopCheck” and “LoadCardInfor” two functions.

C Language DLL： int **LoopCheck**（）
Function：
This function is used to send checkout commands to all control cards in LAN for the preparation to get back the sequence number and IP address of controller cards.
Reture Value:1: already transferring the function “LoopCheck” and still working for the moment. This time of transferring should exit at once.
0:Transferring normally.
C Language DLL： int **LoadCardInfor**(
char* cArrCardId, //introducing parameters, the sequence number for returning to control cards, the first address of array[20];
char* cArrCardIp, // introduction parameters, IP address for returning to IP address of control card, the first address of array[20];
int iNum ， // sending out parameters, appointed to get the sequence number and IP address of iNum control card.
)
Function: To ge the sequence number and IP address of control card in LAN. Firstly it is a must to transfer “LoopCheck” before using “LoadCardInfor”.
Return Value:
-1: still checking now, and the iNum is smaller than or equal to the total number of the already checked control cards, success to get the information of the iNum control card.
-2:still checking now, and the iNum is bigger than the total number of the already checked control cards, failure to get the information of the iNum control card.
1: finished checking, and iNum is smaller than or equal to the total number of the already checked controller cards, success to get the information of the iNum control card.
2: not finished checking, and the iNum is smaller than or equal to the total number of the already checked controller cards, failed to get the information of the iNum control card.

Chapeter Five　I/O Pin Function

5.1 5.1.Pin Definition

Circuit Board Face，J6



§5.2 I/O Pin Oprating Protocol

Protocol to set I/O is listed as follows:

Head Tip	Command Types	Controller Card Sequence No.	Text Length	Text	Verification
0x7e0x7e0x7e（3Byte）	(0x8D)1Byte	20Byte	2*sizeof(UINT)		1Byte（The bytes sum from the

					command type to the end of the text.

Text

I/O No.	I/O Value
1-8	0 :low electrical level，1: high electrical level

Protocol for checking I/O state is listed as follows:

Head Tip	I/O Command Types Checking	Controller Card Sequence No.	Text Length	Text	Verification
0x7e0x7e0x7e（3Byte）	(0x8D)1Byte	20Byte	1*sizeof(UINT)		1Byte (The bytes sum from the command type to the end of the text.)

Text

I/O Sequence No.
1-8

Response Checking

Head Tip	I/O Commands Types Checking	Controller Card Sequence No.	Text Length	Text	Verification(1Byte)
0x7e0x7e0x7e（3Byte）	0xCD (1Byte)	20Byte	2*sizeof(UINT)		(The bytes sum from the command type to the end of the text.)

Text

I/O NO.(size of (UINT))	I/O Value (sizeof(UINT))

1-8	0 :low electrical level，1: high electrical level

5.3,I/O Pin State Checking and Function Setting

C language DLL：int **GetIOState**(
char* cCardId,//Control card's sequence number such as the point of character string: **"**P100-707-00092"
char* cCardIp, //The appointed control card's IP address, such as the character string point "192.168.0.92".
int iTransferType,// //Network communication protocol method, it has 1 and 2 two values, whose meanings are: 1:TCP, one to one reliable transmission.2: UDP, one to one transmission, the speed is really fast but it is possible to lose datas.

int iGroup, //Team No.; Not used ,reserved for the moment, to use 0 is ok.
unsigned int IOpin,//// Pin number needs for checking
int& iMaxPinSetNum,/// Return the total pin number which can be set.
int& iMaxPinCheckNum// Return the total pin number which can be checked.
)
Function: To check the eclectrical level of the appointed pin of the control card.
Return Value: 0、1 ： the eclectrical level of the appointed pin
<0：error in transferring

C Language DLL：int **SetIOState**(
char* cCardId, //Control card's sequence number, such as the character string point "P100-707-00092".
char* cCardIp, //The IP address of the appointed control card, such as the character string point: "192.168.0.92".
int iTransferType, // Network communication protocol method, it has 1 and 2 two values, whose meanings are: 1:TCP,one to one reliable transmission.2: UDP, one to one transmission, the speed is really fast but it is possible to lose datas.
int iGroup, // Team No.; Not used ,reserved for the moment, to use 0 is ok.
unsigned int IOpin,// Pin number needs for setting
int iState ///// if>0,the value is 1;If<=0,the value is 0
)
Function: set the appointed pin's electrical level of controller card.
Return Value: 0: successful setting
1: error in transferring

Chapeter Six True-Time Insert Playing Function

§6.1 Picture functions for playing bmp, jpg, gif formats pictures.

C Language DLL： int **ShowPicture**(
char * cCardId, //Controller's sequence number, such as the character string point "P100-707-00092".
char * cCardIp, //The IP address of the appointed control card, such as the character

string point “192.168.0.92”.
int iProtoType, //Network communication protocol method, it has 1, 2 and 3 three values,whose meanings are:1:TCP, one to one reliable transmission. 2: UDP, one to one transmission, the speed is really fast but it is possible to lose datas. 3: UDP: one to many transmission, the speed is also really fast but it is possible to lose datas also.When using this method, “cCardId” can be filled with “Broadcast!” and “cCardIp” can be filled with “0.0.0.0”.
int iGroup,// Team No.; Not used ,reserved for the moment, to use 0 is ok.
char* cPicFileName//Send out the character string point of picture file namme.Attention: must be absolute path
);

Function:

Insert playing the Bmp format pictures.It is used to send the bmp pictures from computer to the appointed LED control card.

Return Value:

1:successful sending.
0: failed sending. Can use the function “GetDLLErr” to find the reasons of failure.

Examples:It is supposed to send pictures from the computer “ C:\Picture\ScreenCatch.bmp ” to the appointed control card. If the sequence number of the control card were already known: P100-707-00092, and the IP address is 192.168.0.70, then the programme should be as follows:

```
Char cardIp[]=”P100-707-00092”;
Char cardIp[]=”192.168.0.70”;
Char PictureName[]=” C:\Picture\ScreenCatch.bmp”;
Int    SendMethod=11 ////// To adopt TCP one to one transmitting method.
Int Result= ShowPicture (cardId,cardIp, SendMethod,0，PictureName);//Sending out pictures. If successful sending then the result=1, if the picture volume is too large to be sent out then the result=0;
```

C Language Transferring Format： extern “C” __declspec(dllimport) int __stdcall ShowPicture(char * cCardId, char * cCardIp, int iProtoType, int iGroup, char* cPicFileName);

VB Language Transferring Format： Private Declare Function ShowPicture Lib "lwdll.dll" (ByVal cCardId As String, ByVal cCardIp As String, ByVal iProtoType As Integer, ByVal iGroup As Integer ,ByVal cPicFileName As String) As Integer

Delphi Language Transferring Format: Function Show Picture (cCardId:string;cCardIp:string;iProtoType:integer;iGroup:integer;cPicFileName:string):integer;stdcall;external 'LWDLL.dll';

§6.2 Screen Flickering Function

C LanguageDLL： int **CreateWindowRegin** (
COLORREF BackGroundColor, // The flickering area color, if the value is 0x80000000, then the area color should use the contrary colour.

int iLeft, //X coordinate of the top left corner of the square area
int iTop, // Y coordinate of the top left corner of the square area
int iWidth, // Square area width(the totall pixels of the area width)
int iHeigth // Square area height(the total pixels of the area height)
);

Function: Build the flickering areas on LED screen. This function can store the information such as coordinates, color, etc of the flickering areas in advance. It is necessary then to use the function “RealTimeInsertGlintWindow” to realize the flickering on screen(see the next function), therefore the two functions should work cooperatively.

Return Value:

Return the quantity of created areas.

Examples: see below.

C Language Transferring Format： extern "C" __declspec(dllexport) int __stdcall CreateWindowRegin (COLORREF BackGroundColor, int iLeft, int iTop, int iWidth, int iHeigth)

VB Language Transferring Format： Private Declare Function CreateWindowRegin Lib "LwDll.dll" (ByVal BackGroundColor As Long, ByVal iLeft As Integer, ByVal iTop As Integer, ByVal iWidth As Integer, ByVal iHeigth As Integer) As Long

Delphi Language Transferring Format ： Function CreateWindowRegin (BackGroundColor: COLORREF; iLeft: integer; iTop: integer; iWidth:integer; iHeigth: integer):integer;stdcall;external 'LWDLL.dll';

C Language DLL：RealTimeInsertGlintWindow(

char* cCardId, // The character string point of the control card’s sequence number
char* cCardIp, // the character string point of the appointed control card’s IP address.
int iGroup, // Team No.; Not used ,reserved for the moment, to use 0 is ok.
int iProtoType，//Network Communication Protocol Methods, 1,2 and 3 three values, the meaning of each one is listed as follows:

1：TCP: one to one reliable transmitting

2: One to one broadcasting transmitting, the speed is really fast but with the danger to lose datas.

3. UDP one to many broadcasting transmitting, the speed is really fast but also with the danger to lose datas

char* cPicFileName, //The character string point of the pictures in computer, must be full file name and full path, working as the background picture when flickering.
int iGlintTime // Flickering time, unit is second, the smaller the value is, the faster the flickering speed will be.
) ;

Function:after creating one or more areas by using the function “CreateWindowRegin”, the flicker can be realized by using this function and also the already built flickering areas must be cleared away.

Return Value: 1: Command sending successfully

2: failed transmitting, can use the function “GetDllErr” to find out the reasons.

Example：

Send the flickering commands to the control card whose sequence number is “P100-707-00092”, IP address is “192.168.0.92” to buid three flikering areas on D:\\flower.bmp. The first area is on the top left corner of the screen whose width is 20 pixels, height is 20 pixels, color is red; The sencond one: the top left corner coordinate (20,20), width is 30 pixels,height is 30 pixels, the color is green;The third one:(50,50), width is 20 pixels, height is 20 pixels, color is contrary to the backgound’s.Then the program is listed as follows:

///// Build area flickering information separately;

CreateWindowRegin（RGB(255,0,0),0,0,20,20）; // Create the first area information

CreateWindowRegin（RGB(0,255,0),20,20,30,30）/ Create the second area information

Int rectCount=CreateWindowRegin（0x80000000,50,50,20,20）/ Create the third area information,the value of rectCount is 3

////// Realize the flickering on screen with interval of 500 millisecond

Char FileName[]=”D:\\flower.bmp”;

Char cardId[]=”P100-707-00092”;

Char cardIp[]=”192.168.0.92”;

Int Result=RealTimeInsertGlintWindow(cardId,cardIp,1,0,FileName, 500);

If the return value of the above program is 0, then area flickering can be realized and the area information will be cleared away.

C Language Transferring Format：extern "C" __declspec(dllexport) int __stdcall RealTimeInsertGlintWindow(char* cCardId, char* cCardIp, int iGroup, int iProtoType, char* cPicFileName，int iGlintTime);

VB Language Transferring Format ： Private Declare Function RealTimeInsertGlintWindow Lib "LwDll.dll" (ByVal iGlintTime As Integer, ByVal cFileName As String, ByVal cCardId As String, ByVal cCardIp As String, ByVal iGroup As Integer, ByVal iProtoType As Integer) As Long

Delphi Language Transferring Format： Function RealTimeInsertGlintWindow (iGlintTime:

integer;cCardId:string;cCardIp:string;iGroup:integer;iProtoType:integer):integer;stdcall; external 'LWDLL.dll';

§6.3 Multi-Pictures Stunt Displaying

It is necessary to use the two functions “AddPicAndStunt” and “ShowPicStunt” to realize the multi-pictures stunt on screen. The function of “AddPicAndStunt” is to send and store one or more bmp or jpg format pictures and the the pictures’ displaying stunt as well into the control card. After “AddPicAndStunt” finishing one or more pictures(less then 50) sending and saving, then the function “AddPicAndStunt”will realize the stunt on screen. This two functions are introduced as follows:

C Language DLL：int **AddPicAndStunt**(

char* cCardId, // the character string point of the control card’s sequence number

char* cCardIp, // the character string point of the appointed control card’s IP address

int iProtoType, // Network Communication Protocol methodes, value=1, TCP method

int iGroup, // Group No.; Not used ,reserved for the moment, to use 0 is ok

char* cPicFileName, // Picture file name, the full route of the pictuers on computer
int LoopbackNum,//Displaying times, 0: infinite loop(displaying the picture all the time)
int StuntComeType, // Pictures entering the stunt state, see below the stunt list.
int StuntGoType， // . Pictures coming stunt state, see below the stunt list.
int ComeSpeed , // stunt coming speed, unit is millisecond
int GoSpeed , // stunt going speed, unit is millisecond
int SettleTime , // pictures settling time, unit is millisecond
int DisplayMode // pictures displaying effect, see below the displaying effect list
）；

Function: Send the pictures and their displaying stunt as well to the control card. If successful sending, the pictures and stunt will not be displayed on the screen at once. It is necessary to use the function “ShowPicStunt”(see below the explanation of this function) to realize the pictures stunt on screen.

Return Value: 0: successful operation. 1: failed operation, can use the function “GetDllErr” to get the reasons.

Pictures playing stunt: the values can use the following enumerating constants
Pictures displaying stunt:

```
enum _STUNT
{
STUNT_NONE_GO_NO_CLEAN_SCREEN ,// without stunt，（ recommend）
STUNT_RANDOM ,// random stunt;
STUNT_COVER_UP ,// upwards covering stunt;
STUNT_COVER_DOWN ,// downwards covering stunt;
STUNT_COVER_LEFT ,// leftwards covering stunt;
STUNT_COVER_RIGHT ,// rightwards covering stunt;
STUNT_COVER_BOTTOM_RIGHT ,// from diagonal line to the bottom-rightwards covering stunt;
STUNT_COVER_BOTTOM_LEFT,// from diagonal line to the bottom-lefttwards covering stunt;
STUNT_COVER_TOP_RIGHT,// from the diagonal line to the top-rightwards covering stunt;
STUNT_COVER_TOP_LEFT ,// from the diagonal line to the top-leftwards covering stunt
STUNT_COVER_HORIZONE_IN ,// topward and bottomward coming to center-horizontal-line covering stunt;
STUNT_COVER_VERTICAL_IN ,// rightward and leftward coming into center-vertical-line covering stunt;
STUNT_COVER_HORIZONE_OUT ,// center-horizontal-line coming out to topwards and bottomwards covering stunt;
STUNT_COVER_VERTICAL_OUT ,// center-vertical-line coming out to rightwards and leftwards covering stunt;
STUNT_COVER_CROSS_IN ,// cross-in covering stunt;
```

```
    STUNT_COVER_CROSS_OUT                ,// cross-out covering stunt;
    STUNT_COVER_RECTANGLE_IN             ,// middle-in covering stunt;
    STUNT_COVER_RECTANGLE_OUT            ,// middle-out covering
stunt;
    STUNT_INSERT_UP                      ,// upwards moving stunt;
    STUNT_INSERT_DOWN                    ,// downwards moving stunt;
    STUNT_INSERT_LEFT                    ,// leftwards moving stunt;
    STUNT_INSERT_RIGHT                   ,// rightwards moving stunt;
    STUNT_INSERT_BOTTOM_RIGHT,// bottom-rightwards moving stunt;
    STUNT_INSERT_BOTTOM_LEFT  ,// bottom-leftwards moving stunt;
    STUNT_INSERT_TOP_RIGHT               ,// top-rightwards moving stunt;
    STUNT_INSERT_TOP_LEFT                ,// top-leftwards moving stunt;
    STUNT_NUM
    };
```

Pictures displaying effect: the values can use the following enumerating constants

```
    enum   _DISP_MODE
    {
    DISP_MODE_FULL_AREA          ,//Extend to the full area(recommended)
    DISP_MODE_CENTER             ,// area centered;
    DISP_MODE_LEFT               ,// area flush left;
    DISP_MODE_RIGHT              ,// area flush right ;
    DISP_MODE_UP                 ,// area top;
    DISP_MODE_DOWN               ,// area down;
    DISP_MODE_TOP_LEFT           ,// area top-left;
    DISP_MODE_BOTTOM_LEFT        ,// area bottom-left;
    DISP_MODE_TOP_RIGHT          ,// area top-right;
    DISP_MODE_BOTTOM_RIGHT       ,// area bottom-right;
    DISP_MODE_SHRINK_CENTER      ,//area shrinked and centered;
    DISP_MODE_SHRINK_LEFT        ,// area shrinked and left;
    DISP_MODE_SHRINK_RIGHT       ,//area shrinked and right;
    DISP_MODE_SHRINK_UP          ,// area shrinked and up;
    DISP_MODE_SHRINK_DOWN        ,// area shrinked and down;
    DISP_MODE_TILE               ,// spreading
    DISP_MODE_NUM
    };
```

C Language Transferring Format： extern "C" __declspec(dllexport) int __stdcall AddPicAndStunt(char* cCardId, char* cCardIp, int iProtoType,int iGroup,char* cPicFileName, int LoopbackNum, int StuntComeType, int StuntGoType，int ComeSpeed , int GoSpeed ,int SettleTime , int DisplayMode）;

VB Language Transferring Format： Private Declare Function AddPicAndStunt Lib "LwDll.dll" (ByVal cCardId As String, ByVal cCardIp As String, ByVal iProtoType As Integer ，ByVal iGroup As Integer， ByVal cPicFileName As String, ByVal LoopbackNum As Integer, ByVal StuntComeType As Integer，ByVal StuntGoType As

Integer，ByVal ComeSpeed As Integer，ByVal GoSpeed As Integer，ByVal SettleTime As Integer，ByVal DisplayMode As Integer) As Long

Delphi Language Transferring Format：Function AddPicAndStunt (cCardId:string，cCardIp:string， iProtoType:integer， iGroup:integer， cPicFileName:string ， LoopbackNum: integer，StuntComeType: integer，StuntGoType: integer， ComeSpeed: integer ， GoSpeed: integer ， SettleTime: integer ， DisplayMode: integer):integer;stdcall;external 'LWDLL.dll';

ShowPicStunt

C Language DLL：**ShowPicStunt**(

char* cCardId, // the character string point of the control card’s sequence number.

char* cCardIp, // the character string point of the appointed control card’s IP address

nt iGroup, // Team No.. Reserved and not used for the moment, the value is 0.

int iProtoType // Network communication protocol method, use the value 1 only, adopting TCP method

);

Function: After the function “AddPicAndStunt” sending and saving all the pictures and their corresponding stunts as well to the controller cards, the function “ShowPicStunt” then can realize the stunt on screen.

Return Value:0: successful operation 1: failed operation. This time can use the function “GetDllErr” to find out the reasons.

C Language Transferring Format：extern "C" __declspec(dllexport) int __stdcall ShowPicStunt (int PicType, char* cCardId, char* cCardIp, int iGroup, int iProtoType);

VB Language Transferring Format：Private Declare Function ShowPicStunt Lib "LwDll.dll" (ByVal PicType As Integer ， ByVal cCardId As String, ByVal cCardIp As String ， ByVal iGroup As Integer ， ByVal iProtoType As Integer) As Long

Delphi Language Transferring Format：Function ShowPicStunt (PicType:integer ， cCardId:string ， cCardIp:string ， iGroup:integer ， iProtoType:integer):integer;stdcall;external 'LWDLL.dll';

Chapeter Seven Audio Playing Protocol

§7.1 Protocol Contents are listed as follows:

Firstly all the audio materials should be put to the catalogue: /Area0/Audio/

Head Tip	Command Type	Controller Card No.	Text Length	Text	Verification
0x7e0x7e0x7e（3Byte）	(0x8C)1Byte	20Byte	Sizeof(int)	STRUCT_AUDIO _PLAY	1Byte (The bytes sum from the command type to the end of the text.)

Text STRUCT_ AUDIO _PLAY

Playing Mark: 0	MP3 File Names（ Sequence No.+filename extension of the added MP3 files “.mp3”）
Stop Mark: 1	——

Chapeter Eight Screen Parameter-Setting Protocol

Sync Head	Data Type	Controller Sequence No.	Team NO.	Text Length	Text	Verifying Code
3bytes	1bytes	16bytes	4bytes	4bytes	（Text length）	1byte

8.1Screen Operation-Cammands ParametersCOMMAND_SCREEN_OPERATOR

The corresponding analysis.

Text		
Main Function(Non tesing)	sub-text	
Main-function (testing)	sub-function	sub-text

```
typedef struct _SCREEN_OPERATOR_PARAMS
{       UINT            uiMainFunc;// main function.
        UINT            uiSubFunc;  // sub-function.
        UINT
    DataBuffer[SCREEN_OPERATOR_PARAMS_SIZE];//parameters.
}SCREEN_OPERATOR_PARAMS,* P_SCREEN_OPERATOR_PARAMS;
```

§8.2.1 uiMainFunc(Main Function):

```
enum    _SCREEN_OPERATOR_COMM_TYPE
{       SCREEN_OPERATOR_COMM_NULL, // empty.
        SCREEN_OPERATOR_COMM_TEST,  //  Checking  commands,
        (withsub-function)
        SCREEN_OPERATOR_COMM_SET_LIGHT,//brightness setting.
        SCREEN_OPERATOR_COMM_SET_GAMMA,// Gamma setting.
        SCREEN_OPERATOR_COMM_SET_DEFAULT_PARAMS,//Parameters-set
        in the factory.

        SCREEN_OPERATOR_COMM_SET_GRAY  , // grayscale setting.
        SCREEN_OPERATOR_COMM_SET_OE  ,// OE signal setting.
        SCREEN_OPERATOR_COMM_SET_CLOCK_FREQ,//Clock  frequency
        setting.
        SCREEN_OPERATOR_COMM_SET_PLUSE_WIDTH_RATE,//  Duty  Cycle
        setting.
        SCREEN_OPERATOR_COMM_SET_RGB_EXCHANGE,//RGB
        color-exchange setting.
        SCREEN_OPERATOR_COMM_SET_REFRESH_FREQU, //Refresh rate.
        SCREEN_OPERATOR_COMM_SET_DATA_POLARITY,  //  data  polarity
        setting
        SCREEN_OPERATOR_COMM_SET_SCAN_TYPE,// scanning type setting.
        SCREEN_OPERATOR_COMM_SET_RETRACE_TIMES  ,//  retracing  times
        setting.
        SCREEN_OPERATOR_COMM_SET_LOCK_WIDTH ,// Lock width setting.
        SCREEN_OPERATOR_COMM_SET_LINE_START,  //  Line  starting
        sequence No.:0~4,windows default :0.
        SCREEN_OPERATOR_COMM_SET_ROW_EXCHANGE ,//RowsExchanges
        SCREEN_OPERATOR_COMM_SET_COLUME_EXCHANGE,//  Columes
        Exchanges
        SCREEN_OPERATOR_COMM_SET_AREA_LINE_NUMBER,//Each  area
        linages.
        SCREEN_OPERATOR_COMM_SET_PICTURE_MOVE_TO_RIGHT,  //
        Picture rightwards.
        SCREEN_OPERATOR_COMM_SET_LINE_HEIGHT  ,  //  row  height
        setting:0~31,default16.
          SCREEN_OPERATOR_COMM_SET_RETRACE_TYPE  ,  //
          Retracing type setting.
          SCREEN_OPERATOR_COMM_PARAMS_SAVE=0xF0  ,  //
          Parameters saving
        SCREEN_OPERATOR_COMM_SET_SCREEN_LOCK  ,//screen
        locking or unlocking.
         SCREEN_OPERATOR_COMM_PARAMS_NO_SAVE_EXIT,//  exit  screen
```

operation without parameters unsaveing
SCREEN_OPERATOR_COMM_NUM1
};

§8.2.2uiSubFunc(sub-function) the corresponding sub-functions to the testing commands

SCREEN_OPERATOR_COMM_TEST(testing commands)
enum _SCREEN_OPERATOR_COMM_TEST
{ SCREEN_OPERATOR_COMM_TEST_NULL, // empty.
SCREEN_OPERATOR_COMM_TEST_AUTO,// NetGrid Testing
SCREEN_OPERATOR_COMM_TEST_NET_GRID,// NetGrid Testing
SCREEN_OPERATOR_COMM_TEST_GRAY,// grayscale testing.
SCREEN_OPERATOR_COMM_TEST_SNOW_FLAKE_DOTS,//Dots testing
SCREEN_OPERATOR_COMM_TEST_NUM ,
SCREEN_OPERATOR_COMM_TEST_RESERVED=0xF0, // reserved.
SCREEN_OPERATOR_COMM_TEST_EXIT, // exit testing.
SCREEN_OPERATOR_COMM_NUM1
};

SCREEN_OPERATOR_COMM_TEST_AUTO(automaticTesting)DataBuf-fer (parameters analyzing)
SCREEN_OPERATOR_TEST_PARAMS screenOperatorTestParams;

See below the parameters definition, will test all the testing items automatically in circles.

SCREEN_OPERATOR_COMM_TEST_NET_GRID(NetGridTesting) DataBuffer(Parameter analysis)
SCREEN_OPERATOR_TEST_NET_GRID_PARAMS netGridParams;

SCREEN_OPERATOR_COMM_TEST_GRAY (Grayscale testing) DataBuffer(parameters analysis)
SCREEN_OPERATOR_TEST_GRAY_PARAMS grayParams;

SCREEN_OPERATOR_COMM_TEST_SNOW_FLAKE_DOTS(Dots Testing) DataBuffer(parameters analysis)
SCREEN_OPERATOR_TEST_SNOW_FLAKE_DOTS_PARAMS snowFlakeDotsParams;
typedef struct _SCREEN_OPERATOR_TEST_PARAMS
{ SCREEN_OPERATOR_TEST_NET_GRID_PARAMS netGridParams;
SCREEN_OPERATOR_TEST_GRAY_PARAMS grayParams;

```
        SCREEN_OPERATOR_TEST_SNOW_FLAKE_DOTS_PARAMS
    snowFlakeDotsParams;
        UINT Reserved[100]; // used for software upgrading, the added parameters
        should then minus this reserved value.
}SCREEN_OPERATOR_TEST_PARAMS,*
P_SCREEN_OPERATOR_TEST_PARAMS;

typedef struct _SCREEN_OPERATOR_TEST_NET_GRID_PARAMS
{       UINT            uiSettleTime;//  Unit settling time(ms,1000~).
        UINT            uiInterval;  // intervals between lines(,32,...).
        UINT            bHorizonalLine;// horizonal line testing
        UINT            bVerticalLine;// vertical line testing.
        UINT            bTopLeftSlantLine;// Top-Left-Slant line testing
        UINT            bTopRightSlantLine;//Top-Right-Slant line testing
        UINT            uiFrontColorNumber;// Foreground color changing times.
        COLORREF        frontColor[MAX_FRONT_COLOR_NUMBER];
//foreground-color array value.
        COLORREF        backgroundColor;        //  background-color value.
}SCREEN_OPERATOR_TEST_NET_GRID_PARAMS,*
P_SCREEN_OPERATOR_TEST_NET_GRID_PARAMS;

typedef struct _SCREEN_OPERATOR_TEST_GRAY_PARAMS
{       UINT    uiSettleTime;       // SettleTime (ms,100~).
        UINT    uiInterval;         // GrayScale changing ranges and step (1~255).
        UINT    bIntersect;         //  0: all colors keep continuous testing,1: color
of the same value keep testing alternately.
        UINT    bTestRed;           //  red testing
        UINT    bTestGreen;         //  green testing
        UINT    bTestBlue;          // blue testing
}SCREEN_OPERATOR_TEST_GRAY_PARAMS,*
P_SCREEN_OPERATOR_TEST_GRAY_PARAMS;

typedef struct _SCREEN_OPERATOR_TEST_SNOW_FLAKE_DOTS_PARAMS
{       UINT    uiSnowFlakeDotsType;    //Dots testing, see below
        UINT    uiInterval;             // intervals between dots or lines.
        COLORREFfrontColor;             // dot foreground color
        COLORREFbackgroundColor;        // background color.
}SCREEN_OPERATOR_TEST_SNOW_FLAKE_DOTS_PARAMS,*
P_SCREEN_OPERATOR_TEST_SNOW_FLAKE_DOTS_PARAMS;

uiSnowFlakeDotsType(dots-testing type)
enum    _SCREEN_OPERATOR_SNOW_FLAKE_DOTS_TYPE
{       SNOW_FLAKE_DOTS_TYPE_DOTS               ,   // dot.
        SNOW_FLAKE_DOTS_TYPE_HORIZONTAL_LINE ,  //horizontal line.
```

```
        SNOW_FLAKE_DOTS_TYPE_VERTICAL       ,   // vertical line.
        SNOW_FLAKE_DOTS_TYPE_NUM
};
```

§8.2.3 (Non-testing) the corresponding sub-text meaning to the main function

```
SCREEN_OPERATOR_COMM_SET_LIGHT(brightness setting)
        UINT    uiLight;            // brightness value:0~3.

SCREEN_OPERATOR_COMM_SET_GAMMA(Gamma Setting)
typedef struct _RGB_GAMMA               // (0~4.0)
{       FLOAT                   fRed;       // red.
        FLOAT                   fGreen;     // green.
        FLOAT                   fBlue;      // blue.
}RGB_GAMMA,* P_RGB_GAMMA;

SCREEN_OPERATOR_COMM_SET_DEFAULT_PARAMS ( parameters set in factory)
    no parameters, come to the windows default parameters.
SCREEN_OPERATOR_COMM_SET_GRAY(grayscale setting)        **** the value set for the moment: 4096.
        UINT    uiGray;     // grayscale value:64,256,1024,4096.
        UINT    uiGrayNumber;//Grayscale Parameter1:0~1023, windows default:19.
        UINT    uiOE_Number;//Grayscale Parameter 2:0~15, windows default:4.

SCREEN_OPERATOR_COMM_SET_OE(OE Polarity Setting)
        UINT    uiOE;       // 0,1.
SCREEN_OPERATOR_COMM_SET_CLOCK_FREQ(clock frequency setting)
        UINT    uiClockFreq;// 0~3.
SCREEN_OPERATOR_COMM_SET_PLUSE_WIDTH_RATE(space-ocuppied rate setting)
        UINT    uiPluseWidthRate; // 0~2.
SCREEN_OPERATOR_COMM_SET_RGB_EXCHANGE ( RGB color exchanges setting)
        UINT    uiRgbExchange;      // RGB color exchanges,0~7,undetermined???
SCREEN_OPERATOR_COMM_SET_REFRESH_FREQU(refresh rate)
        UINT    uiScreenRefreshFrequ;// refresh rate:60~4000Hz,undetermined???
SCREEN_OPERATOR_COMM_SET_DATA_POLARITY(data polarity setting)
        UINT    uiDataPolarity;  //1: datas reverse-outputing 0:datas outputing normally.
SCREEN_OPERATOR_COMM_SET_SCAN_TYPE(scanning type setting)
        UINT    uiScanType;         // scanning type:0: 1/1 scanning, 1:1/4 scanning , 2: 1/8 scanning, 3: 1/16 scanning;
SCREEN_OPERATOR_COMM_SET_RETRACE_TIMES(retracing times setting)
        UINT    uiRetraceTimes; //  retracing  times:0:no  retrace,1:1retrace,3:3
```

retrace,7:7retrace,15:15 retrace.
SCREEN_OPERATOR_COMM_SET_LOCK_WIDTH**(**width of Latch signal**)**
UINT uiLockWidth; //LockWidth:0~1023,Windows default is 128.
SCREEN_OPERATOR_COMM_SET_LINE_START(Row starting sequence number)
UINT uiLineStart; // Row starting sequence number:0~4,windows default:0.
SCREEN_OPERATOR_COMM_SET_ROW_EXCHANGE(Rows Exchange)
UINT uiRowExchange; // rows exchange.
SCREEN_OPERATOR_COMM_SET_COLUME_EXCHANGE(Columes Exchanges)
UINT uiColumeExchange; // Columes Exchanges.
SCREEN_OPERATOR_COMM_SET_AREA_LINE_NUMBER(area total rows)
UINT uiAreaLineNumber; // area total rows:???
SCREEN_OPERATOR_COMM_SET_PICTURE_MOVE_TO_RIGHT(picture moving)
UINT uiPictureMoveToRight; // picture rightwards:0~65535.
SCREEN_OPERATOR_COMM_SET_LINE_HEIGHT(row-height setting:0~31,default 16)
UINT uiLineHeight; // RowHeight:0~31,Default=16.
SCREEN_OPERATOR_COMM_SET_RETRACE_TYPE(retracing-type setting)
UINT RetraceType[3]; // retracing type.

SCREEN_OPERATOR_COMM_PARAMS_SAVE(0xF0, parameters saving)
SCREEN_OPERATOR_BASE_PARAMS screenOperatorBaseParams;
SCREEN_OPERATOR_COMM_SET_SCREEN_LOCK(screen lock or unlock)
No parameters,lock the current screen (pause playing programs)or unlock(continue playing programs)
SCREEN_OPERATOR_COMM_PARAMS_NO_SAVE_EXIT(screen parameters no-save-exit)
No parameters, no-save for the setting values of the debugging operations.

ChapeterNine.Controlling System State-Checking and Basic Parameter Commands Setting

§9.1 State-Checking Commands（COMMAND_LOOPCHECK）

From Computer： data type： **0x82**；
Text length：0；（0 bytes）；
Text Contents： empty；
Controller Response：data type **0xc3**
Text Length: sizeof(BOARD_REPORT_INFO)
Text contents：BOARD_REPORT_INFO
typedef struct _BOARD_REPORT_INFO
{
charpSerialNumber[SERIAL_NUMBER_SIZE]; //sequence number.// Must At

```
Head.
    SCREEN_OPERATOR_BASE_PARAMS     screenOperatorBaseParams;
    SCREEN_OPERATOR_HIGH_PARAMS     screenOperatorHighParams;
     UINT  uiRunOrPauseState;// 0: programs playing normally,1: playing pause,2:
     testiing pause
    UINT   bProgramUpgrading;// 1: Control card downloading new programs
    UINT   uiPlayMode;        // 0: normal palying mode, 1: synchronized playing
    mode, 2:   online preview playing mode
    UINT   uiConfigNum;      // Total programs playing for the moment (0:none,
remind users to download).
    UINT   bTestingOperator; // 1: Control card being tested
    UINT   uiSoftwareVersion;// software version number
    UINT   uiHardwareVersion;// hardware version number
    UINT     uiDiskSize_KB;       // disk space size
    UINT     uiDiskUsedSize_KB;// already used disk space size
    UINT     uiReserve[7];
    UINT     uiMapTableSpecialFlag;
    UINT     uiReserve1[2];
}BOARD_REPORT_INFO,* P_BOARD_REPORT_INFO;
typedef struct _SCREEN_OPERATOR_BASE_PARAMS
{
    charpCardName[CARD_NAME_LENGTH];   //current controller name.
    charMacAddress[IP_MAC_ADDRESS_SIZE]; //Mac Address.
    charIpAddress[IP_ADDRESS_SIZE];  //Mac IP Address.
    charIpMask[IP_ADDRESS_SIZE];     //IP Mask
    UINT   uiIpPortBase;                  // network port number
    UINT   uiScreenType;              //1:homochromy,2:double color,3:full color.
    UINT   uiCardScreenWidth;                 // Screen Width.
    UINT   uiCardScreenHeight;                // Screen Height
    UINT   uiLanguage; // language value, see definition in the setting process .
    UINT   uiMaxWidth;
    UINT     uiMaxHeight;
    UINT     uiOpenRetrace;
    UINT     uiScanOrStatic;
    UINT     uiGroupNumber;
    char      NetGateWayAddress[IP_ADDRESS_SIZE];
    UINT     uiReserve[95-(IP_ADDRESS_SIZE/4)];

}SCREEN_OPERATOR_BASE_PARAMS,*
P_SCREEN_OPERATOR_BASE_PARAMS;

typedef struct _SCREEN_OPERATOR_HIGH_PARAMS
{
    UINT   uiOE;                  // OE Polarity:0,1.
```

```
    UINT    uiClockFreq;            // clock frequency:0~3.
    UINT    uiScreenRefreshFrequ;// refresh rate:60~4000Hz,undetermined???
    UINT    uiPluseWidthRate;    // Space-occupied rate:0~2.
    UINT    uiRgbExchange;       // RGB color exchange,0~7,undetermined???
   UINT    uiDataPolarity;       // 0: datas reverse-outpiting, 1: datas outputing
   normally
    UINT    uiLight;              // brightness:0~3.
    UINT    uiGray;               // grayscale:64,256,1024,4096.
    RGB_GAMMA  gamma;             // Gamma:0~4.0.
    UINT    uiScanType; // scanning type:???undetermined. 0:1/1 scanning ,1:1/4
    scanning, 0:1/8 scanning, 0:1/16 scanning;
   UINT      uiRetraceTimes;// retrace times:0:no retraces,1:1retraces,3:3 retraces,7:7
   retraces,15:15 retraces.
    UINT    uiGrayNumber;  // Grayscale Parameter 1:0~1023,windows default 为 19.
    UINT    uiOE_Number;   // Grayscale Parameter 2:0~15,windows default 4.
    UINT      uiSensorLightPercent;
    UINT    uiLineStart;        // row starting sequence number:0~7,windows default 0.
    UINT      uiLineExchange;
    UINT      uiColumeExchange;
    UINT    uiAreaLineNumber;  // total area rows:???
    UINT    uiPictureMoveToRight; // picture moving rightwards:0~65535.
    UINT    uiLineHeight;         // row height:0~31,Default=16.
    UINT    RetraceType[3];  // retrace type.
    UINT      uiRotate;
    UINT      uiHubType;
     UINT  Reserved[90]; // Used for the software upgrading, the added parameters
     then should be minus the reserved value.
 }SCREEN_OPERATOR_HIGH_PARAMS,*
P_SCREEN_OPERATOR_HIGH_PARAMS;
```

9.2 Loop-Checking Commands（NET_COMMU_TYPE_SYSTEM_BROADCAST）

From Computer： data types：**0x90**；
Text Length：0；（0 bytes）；
Text Contents： empty；

Controller Response：data types **0xd0**
Text Length: sizeof(BOARD_REPORT_INFO)
Text Contents：BOARD_REPORT_INFO

```
typedef struct _BOARD_REPORT_SIMPLIFY_INFO
{       UINT    uiRunOrPauseState;// 0:programs palying normally,1:program playing
pause,2:testing pause
        UINT    bProgramUpgrading;// 1: Control card downloading new programs
        UINT    uiPlayMode;// 0: normal playing mode,1: synchronized playing mode,
         2: Online preview playing mode.
```

```
    UINT uiConfigNum;// Total programs playing for the moment(0:no programs,
    remind users to download programs).
    UINT   bTestingOperator; // 1: Controller being tested .
    UINT   uiSoftwareVersion;// software version number.
    UINT   uiHardwareVersion;// hardware version number.
    UINT      uiDiskSize_KB;       // Disk space size.
    UINT      uiDiskUsedSize_KB;// Already used disk space size.
    charpCardName[CARD_NAME_LENGTH];// The current controller name.
    UINT   Reserved[2];
}BOARD_REPORT_SIMPLIFY_INFO,* P_BOARD_REPORT_SIMPLIFY_INFO;
```

§9.3 Controller Time Setting

From Computer： data type： **0x83**；
Text Length： 0x0000 001C；（28 bytes）；
Text Contents：First Line unsigned integer type ： year， 1900～2100；
Second line unsigned integer type：month， 1～12；
Third line unsigned integer type: day, 1～31；
Fourth line unsigned integer type: hour, 0～23；
Fifth line unsigned integer type: minutes, 0～59；
Sixth line unsigned integer type: second, 0～59；
Seventh line unsigned integer type: week, 1～7；
Controller's Responsion:data type:0xC3; the same as the explanation of "Common Response of Controller Protocol".