



## 比较v1.1修改内容

### 1. 增加节目页背景音乐添加删除函数

```
BOOL STDCALL CI_AddBackgroundMusic(wchar_t* wsPageName, wchar_t* pMusicPath);  
BOOL STDCALL CI_RemoveBackgroundMusic(wchar_t* wsPageName, wchar_t* wsMusicPath=0);
```

### 2. 删除节目页属性结构中的背景音乐

```
struct PageAttribute{  
    wchar_t wsPageName[ATT_BUFF_SIZE];//节目页唯一标识  
    int nWindowAmount;//窗口数目  
    wchar_t wsBackgroundPath[ATT_BUFF_SIZE];//背景图片  
    int nBackgroundColor;//背景色  
    //wchar_t wsBackgroundMusicPath[ATT_BUFF_SIZE];//背景音乐  
    int nPlayDuration;//播放时间  
    bool bWaitFinish;//等待节目播放完毕, true:该节目页的内容全部播放超过一次则节目页结束, false:按播放时间播放节目页,  
};
```

## 二次开发库说明文档v1.2

### 第一节、EncodeMaterial.dll动态库API说明（可参照目录TestEncodeMaterialDll源代码）

（编译生成素材和转换旧项目文件到新项目文件的动态库API函数）：

```
extern "C" __declspec(dllexport) void __stdcall StartupEncodeMaterial();  
extern "C" __declspec(dllexport) void __stdcall ShutDownEncodeMaterial();  
  
extern "C" __declspec(dllexport) int __stdcall LxdToXml(char *cLxdName, char *cXmlName);  
extern "C" __declspec(dllexport) int __stdcall CompilerTemperatureHumidity(char *cFileName,  
                                                                            char *cFontName,  
                                                                            int iFontSize,  
                                                                            int iFontColor,  
                                                                            int iBackColor,  
                                                                            char *cTemperateDescription,  
                                                                            char *cHumidityDescription);  
  
extern "C" __declspec(dllexport) int __stdcall CompilerDigitalClock(char *cFileName,  
                                                                      char *cFontName,  
                                                                      int iFontSize,  
                                                                      int iFontColor,
```



```
extern "C" __declspec(dllexport) int __stdcall  ComplierAnalogClock(char *cFileName, //素材文件路径名
                                                                    int iBackColor, //背景色
                                                                    int uiAnalogClockXingZhuang, //表盘形状, 0圆形, 1矩形
                                                                    int uiHourXingZhuang, //时标形状, //0圆形, 1矩形
                                                                    int uiHourPointSize, //时标大小
                                                                    int crHourPoint, //时标颜色
                                                                    int uiMinuteXingZhuang, //分标形状, //0圆形, 1矩形
                                                                    int uiMinutePointSize, //分标大小
                                                                    int crMinutePoint, //分标颜色
                                                                    int crHourGuidelines, //时针颜色
                                                                    int crMinuteGuidelines, //分针颜色
                                                                    int crSecondGuidelines, //秒针颜色
                                                                    char *cAddress, //时钟域内容, 如北京
                                                                    char *cAddressFontName, //时钟域字体
                                                                    int iAddressFontSize, //时钟域字体大小
                                                                    int iAddressFontColor, //时钟域字体颜色
                                                                    char *cDateFontName, //日期字体
                                                                    int iDateFontSize, //日期字体大小
                                                                    int iDateFontColor, //日期字体颜色
                                                                    char *cWeekFontName, //星期字体
                                                                    int iWeekFontSize, //星期字体大小
                                                                    int iWeekFontColor, //星期字体颜色
                                                                    int iWndWidth, //表盘宽
                                                                    int iWndHeight //表盘高
                                                                    );

extern "C" __declspec(dllexport) int __stdcall  ComplierTimer(char *cFileName,
                                                             char *cFontName,
                                                             int iFontSize,
                                                             int iFontColor,
                                                             int iBackColor,
                                                             char *cTextDescription
                                                             );
```

## 第二节、XiXunLedProgram.dll动态库API说明



**(操作熙讯xml项目文件的动态库API函数):**

```
// The following ifdef block is the standard way of creating macros which make exporting
// from a DLL simpler. All files within this DLL are compiled with the XIXUNLEDPROGRAM_EXPORTS
// symbol defined on the command line. This symbol should not be defined on any project
// that uses this DLL. This way any other project whose source files include this file see
// XIXUNLEDPROGRAM_API functions as being imported from a DLL, whereas this DLL sees symbols
// defined with this macro as being exported.
//熙讯LED节目XML格式接口
//日期: 2012-11-0
//更改: 2012-12-03
//更新内容:
//1, 所有ItemName归属唯一的windowName和PageName <ITEM_INFO结构> 存放在list_item链表中
//2, 所有windowName归属唯一的pageName <WINDOW_INFO结构> 存放在list_window链表中
//3, 所有pageName必须唯一, 存放在list_page链表中
//4, CreateProgram时将上述三个链表清空
//5, LoadProgram时将xml中的pageName, windowName和itemName装入上述三个链表中

//更改: 2012-12-11
//更新内容:
//1, 新增函数CI_AddBackgroundMusic和CI_RemoveBackgroundMusic向页面添加背景音乐
//2, PageAttribute中的nWindowAmount属性启用, 新增窗口和删除窗口后自动更新该属性值
//3, PageAttribute中wsBackgroundMusicPath弃用<xsd中保留并表现为一组属性>

//更改: 2012-12-13
//更新内容:
//TextFileAttribute中增加nFontWeight和nTextColor属性

//更改: 2012-12-18
//更新内容:
//1, AnalogClockWindow增加nAnalogClockShape表盘形状属性
//2, AnalogClockWindow增加bShowWeek有无星期标志
//3, PeroidWindow增加计时类型nTimerType
//4, PeroidWindow的DateTime结构中增加星期属性

//更改: 2012-12-24
//更新内容:
```



```
//1, PageAttribute增加别名 (wsPageVariantName) 属性
//2, PeriodWindow增加只显示最大单位 (bMaxUnitOnly), 多窗口组合显示 (bMultiView)
// 和显示时间单位 (bShowTimeUnit) 三个属性
//3, PageAttribute中的BackgroundMusic增加ItemName属性

//接口初始化和释放-----//
void STDCALL CI_InitDLL();
void STDCALL CI_UnInitDLL();

//---- Proram -----//
//通过xml文件路径加载一个已有的节目
BOOL STDCALL CI_LoadProgram(wchar_t* xmlFile);
void STDCALL CI_SaveProgram(wchar_t* xmlFile);
//新建一个节目, 填充节目属性(节目页的数量根据创建的page数量自动更新)
void STDCALL CI_CreateProgram(struct ProgramAttribute* pAttr);
//获得和设置节目属性
BOOL STDCALL CI_GetProgramAttribute(struct ProgramAttribute* pAttr);
BOOL STDCALL CI_SetProgramAttribute(struct ProgramAttribute* pAttr);
//-----//

//---- Pages + Page -----//
//新建一个节目页, 填充节目页属性, 返回节目页名称. (函数自动为每个节目生成windows节点)
wchar_t* STDCALL CI_CreatePage(struct PageAttribute* pAttr);
BOOL STDCALL CI_RemovePage(wchar_t* wsPageName);
//获得和设置节目页属性
BOOL STDCALL CI_GetPageAttribute(wchar_t* wsPageName, PageAttribute* pAttr);
BOOL STDCALL CI_SetPageAttribute(wchar_t* wsPageName, PageAttribute* pAttr);
BOOL STDCALL CI_AddBackgroundMusic(wchar_t* wsPageName, struct BackgroundMusic* pMusic);
BOOL STDCALL CI_RemoveBackgroundMusic(wchar_t* wsPageName, wchar_t* wsItemName=0);
//新建一个全局节目页, 填充节目页属性, 返回节目页名称 (该函数只允许调用一次)
wchar_t* STDCALL CI_CreateGlobalPage(struct PageAttribute* pAttr);
//移除全局节目页可以调用Removepage ()
//获得和设置属性调用CI_GetPageAttribute和CI_SetPageAttribute
//获取和设置节目页限制
BOOL STDCALL CI_AddConstraint(wchar_t* wsPageName, wchar_t* wsItemName,
    struct DateConstraint* ds, struct TimeConstraint* ts, struct WeekConstraint* ws);
//移除一个节目页播放限制
```



```
BOOL STDCALL CI_RemoveConstraint(wchar_t* wsPageName, wchar_t* wsItemName);
//-----//

//---- Windows + FileWindow -----//
//新建一个文件窗，需要两个输入参数：属主节目页名称和文件窗属性
wchar_t* STDCALL CI_CreateFileWindow(wchar_t* wsPageName, struct FileWindowAttribute* pAttr);
//获得和设置文件窗口属性
BOOL STDCALL CI_GetFileWindowAttribute(wchar_t* wsPageName, wchar_t* wsWindowName, struct FileWindowAttribute* pAttr);
BOOL STDCALL CI_SetFileWindowAttribute(wchar_t* wsPageName, wchar_t* wsWindowName, struct FileWindowAttribute* pAttr);
//移除文件窗口
BOOL STDCALL CI_RemoveWindow(wchar_t* wsPageName, wchar_t* wsWindowName);
//向文件窗中添加素材
BOOL STDCALL CI_AddImageToFileWindow(wchar_t* wsPageName, wchar_t* wsWindowName, struct PictureAttribute* pAttr);
BOOL STDCALL CI_AddTextToFileWindow(wchar_t* wsPageName, wchar_t* wsWindowName, struct TextFileAttribute* pAttr);
BOOL STDCALL CI_AddVideoToFileWindow(wchar_t* wsPageName, wchar_t* wsWindowName, struct VideoAttribute* pAttr);
BOOL STDCALL CI_AddGIFToFileWindow(wchar_t* wsPageName, wchar_t* wsWindowName, struct GIFAttribute* pAttr);
BOOL STDCALL CI_AddRTFTToFileWindow(wchar_t* wsPageName, wchar_t* wsWindowName, struct RTFAttribute* pAttr);
BOOL STDCALL CI_AddFlashToFileWindow(wchar_t* wsPageName, wchar_t* wsWindowName, struct FlashAttribute* pAttr);
BOOL STDCALL CI_RemoveItemFromFileWindow(wchar_t* wsPageName, wchar_t* wsWindowName, wchar_t* wsItemName);
//-----//

//---- windows + SingLineTextWindow -----//
//创建一个单行文本窗, 返回窗口名称
wchar_t* STDCALL CI_CreateSinglineTextWindow(wchar_t* wsPageName,
                                             struct TextWindowAttribute* pWindow,
                                             struct SingLineTextAttribute* pAttr,
                                             struct TextAttribute* pText);
//移除一个SinglineTextWindow, 同CI_RemoveWindow()
//XIXUNLEDPROGRAM_API bool CI_RemoveWindow(wchar_t* wsWindowName);
//获得和设置TextWindow的属性和文字
BOOL STDCALL CI_GetSinglineTextAttribute(wchar_t* wsPageName,
                                         wchar_t* wsWindowName,
                                         struct TextWindowAttribute* pWindow,
                                         struct SingLineTextAttribute* pAttr,
                                         struct TextAttribute* pText=0);

BOOL STDCALL CI_SetSinglineTextAttribute(wchar_t* wsPageName, wchar_t* wsWindowName,
```



```
        struct TextWindowAttribute* pWindow,
        struct SingLineTextAttribute* pAttr,
        struct TextAttribute* pText=0);
BOOL STDCALL CI_GetText(wchar_t* wsPageName, wchar_t* wsWindowName, wchar_t* wsItemName, struct TextAttribute* pText);
BOOL STDCALL CI_SetText(wchar_t* wsPageName, wchar_t* wsWindowName, wchar_t* wsItemName, struct TextAttribute* pText);
//-----//

BOOL STDCALL CI_GetTextWindowAttribute(wchar_t* wsPageName, wchar_t* wsWindowName, struct TextWindowAttribute* pAttr);
BOOL STDCALL CI_SetTextWindowAttribute(wchar_t* wsPageName, wchar_t* wsWindowName, struct TextWindowAttribute* pAttr);
//---- windows + MultiLineTextwindow -----//
//创建一个多行文本窗,返回窗口名称
wchar_t* STDCALL CI_CreateMultilineTextWindow(wchar_t* wsPageName,
        struct TextWindowAttribute* pWindow);
//移除一MultilineTextWindow,同CI_RemoveWindow()
//新增一个文本到多行文本窗中
BOOL STDCALL CI_AddMultiTextToWindow(wchar_t* wsPageName, wchar_t* wsWindowName,
        struct MultiLineTextAttribute* pAttr, struct TextAttribute* pText);
//获得和设置
BOOL STDCALL CI_GetMultilineTextAttribute(wchar_t* wsPageName, wchar_t* wsWindowName,
        wchar_t* wsItemName, struct TextWindowAttribute* pWindow,
        struct MultiLineTextAttribute* pAttr, struct TextAttribute* pText=0);
BOOL STDCALL CI_SetMultilineTextAttribute(wchar_t* wsPageName, wchar_t* wsWindowName, wchar_t* wsItemName,
        struct MultiLineTextAttribute* pAttr, struct TextAttribute* pText);
//获得和设置Text同CI_GetText() and CI_SetText()
//移除一行文字
BOOL STDCALL CI_RemoveText(wchar_t* wsPageName, wchar_t* wsWindowName, wchar_t* wsItemName);
//-----//

//---- windows + TableWindow -----//
//创建一个表格窗
wchar_t* STDCALL CI_CreateTableWindow(wchar_t* wsPageName, struct TableWindowAttribute* pAttr);
//移除同CI_RemoveWindow()
//获得和设置TableWindow的属性
BOOL STDCALL CI_GetTableWindowAttribute(wchar_t* wsPageName, wchar_t* wsWindowName,
        struct TableWindowAttribute* pAttr);
BOOL STDCALL CI_SetTableWindowAttribute(wchar_t* wsPageName, wchar_t* wsWindowName,
        struct TableWindowAttribute* pAttr);
```



```
//新增table
BOOL STDCALL CI_AddTable(wchar_t* wsPageName, wchar_t* wsTableWindowName, struct TableAttribute* pAttr,
    struct TextAttribute* pTitle=0);
BOOL STDCALL CI_RemoveTable(wchar_t* wsPageName, wchar_t* wsWidnowName, wchar_t* wsTableName);
BOOL STDCALL CI_GetTableAttribute(wchar_t* wsPageName, wchar_t* wsWindowName, wchar_t* wsTableName, struct TableAttribute* pAttr);
BOOL STDCALL CI_SetTableAttribute(wchar_t* wsPageName, wchar_t* wsWindowName, wchar_t* wsTableName, struct TableAttribute* pAttr);
//-----//

//---- windows + PeroidWindow -----//
//创建一个计时窗
wchar_t* STDCALL CI_CreatePeroidWindow(wchar_t* wsPageName,
    struct PeroidWindowAttribute* pAttr,
    struct DATE_TIME * pDt,
    struct TextAttribute* pTitle=0);

//移除同CI_RemoveWindow ()
//获得和设置PeroidWindow的属性
BOOL STDCALL CI_GetPeroidWindowAttribute(wchar_t* wsPageName, wchar_t* wsWindowName,
    struct PeroidWindowAttribute* pAttr);
BOOL STDCALL CI_SetPeroidWindowAttribute(wchar_t* wsPageName, wchar_t* wsWindowName,
    struct PeroidWindowAttribute* pAttr);

//获得和设置title 同CI_GetText() and CI_SetText()
//-----//

//---- windows + ClockWindow -----//
//创建一个数字时钟窗
wchar_t* STDCALL CI_CreateDigitalClockWindow(wchar_t* wsPageName,
    struct DigitalClockWindowAttribute* pAttr, struct TextAttribute* pTitle=0);
//移除同CI_RemoveWindow ()

//获得和设置数字时钟窗口的属性
BOOL STDCALL CI_GetDigitalClockWindowAttribute(wchar_t* wsPageName, wchar_t* wsWindowName,
    struct DigitalClockWindowAttribute* pAttr);
BOOL STDCALL CI_SetDigitalClockWindowAttribute(wchar_t* wsPageName, wchar_t* wsWindowName,
    struct DigitalClockWindowAttribute* pAttr);

//创建一个模拟时钟窗
wchar_t* STDCALL CI_CreateAnalogClockWindow(wchar_t* wsPageName,
```



```
    struct AnalogClockWindowAttribute* pAttr, struct TextAttribute* pTitle=0,
    struct TextAttribute* pDateTitle=0, struct TextAttribute* pWeekTitle=0);
//移除同CI_RemoveWindow ()

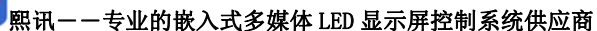
//获得和设置模拟时钟窗口的属性
BOOL STDCALL CI_GetAnalogClockWindowAttribute(wchar_t* wsPageName, wchar_t* wsWindowName,
                                              struct AnalogClockWindowAttribute* pAttr);
BOOL STDCALL CI_SetAnalogClockWindowAttribute(wchar_t* wsPageName, wchar_t* wsWindowName,
                                              struct AnalogClockWindowAttribute* pAttr);

//获得和设置title 同CI_GetText() and CI_SetText()
//-----//

//---- windows + TempHumiWindow -----//
//创建一个温湿度窗
wchar_t* STDCALL CI_CreateTempHumiWindow(wchar_t* wsPageName,
                                         struct TemperatureHumidityWindowAttribute* pAttr);
//移除同CI_RemoveWindow()
//获得和设置TempHumiWindow的属性
BOOL STDCALL CI_GetTempHumiWindowAttribute(wchar_t* wsPageName, wchar_t* wsWindowName,
                                           struct TemperatureHumidityWindowAttribute* pAttr);
BOOL STDCALL CI_SetTempHumiWindowAttribute(wchar_t* wsPageName, wchar_t* wsWindowName,
                                           struct TemperatureHumidityWindowAttribute* pAttr);
//-----//

//初始化属性-----//
void STDCALL CI_InitTextFile(struct TextFileAttribute* pAttr);
void STDCALL CI_InitText(struct TextAttribute* pAttr);
void STDCALL CI_InitPicture(struct PictureAttribute* pAttr);
void STDCALL CI_InitVideo(struct VideoAttribute* pAttr);
void STDCALL CI_InitGIF(struct GIFAttribute* pAttr);
void STDCALL CI_InitRTF(struct RTFAttribute* pAttr);
void STDCALL CI_InitFlash(struct FlashAttribute* pAttr);
void STDCALL CI_InitProgram(struct ProgramAttribute* pAttr);
void STDCALL CI_InitPage(struct PageAttribute* pAttr);
void STDCALL CI_InitFileWindow(struct FileWindowAttribute* pAttr);
void STDCALL CI_InitSinglineText(struct SingLineTextAttribute* pAttr);
```





Shanghai Xixun Electronic Technology Company Ltd. Website: <http://www.xixunled.com>



```
    int nFontWeigth;
    int nTextColor;
    //int nPlayLoopCount;
};

//文本类型 -- 6 --
struct TextAttribute{
    wchar_t wsItemName[ATT_BUFF_SIZE]; //文本自定义名称
    wchar_t wsText[ATT_BUFF_SIZE]; //文本内容
    wchar_t wsFont[ATT_BUFF_SIZE]; //字体
    int nFontWeight; //字体大小
    int nTextColor; //字体颜色
    int nTextStyle; //字体风格(粗体0x4, 斜体0x2, 下划线0x1)
};

//图片类型 --- 9 ---
struct PictureAttribute{
    wchar_t wsItemName[ATT_BUFF_SIZE]; //唯一标识名称
    wchar_t wsFilePath[ATT_BUFF_SIZE]; //文件路径名
    int nViewMode; //显示模式, 0: 普通居中 1: 缩放 2: 拉伸 3: 平铺
    int nBackColor; //背景色
    int nSpecialEffects; //进场特技
    int nSpecialEffectSpeed; //进场特技速度
    int nSpecialEffectDuration; //停留时间
    int nFinishSpecialEffects; //出场特技
    int nFinishSpecialEffectSpeed; //出场特技速度
};

//视频类型-- 6
struct VideoAttribute{
    wchar_t wsItemName[ATT_BUFF_SIZE]; //唯一标识名称
    wchar_t wsFilePath[ATT_BUFF_SIZE]; //文件路径名
    int nPlayLoopCount; //循环次数
    int nBackColor; //背景色
    int nVideoWidth; //视频宽度
    int nVideoHeight; //视频高度
};
```



```
//GIF类型 ---5
struct GIFAttribute{
    wchar_t wsItemName[ATT_BUFF_SIZE];//唯一标识名称
    wchar_t wsFilePath[ATT_BUFF_SIZE];//文件路径名
    int nBackColor;//循环次数
    int nSpecialEffects;//显示模式, 0: 普通居中1: 缩放2: 拉伸3: 平铺
    int nPlayLoopCount;//循环次数
};

//RTF类型 -- 10
struct RTFAttribute{
    wchar_t wsItemName[ATT_BUFF_SIZE];//唯一标识名称
    wchar_t wsFilePath[ATT_BUFF_SIZE];//文件路径名
    wchar_t wsBackground[ATT_BUFF_SIZE];//背景图片
    int nBackColor;//背景色
    int nSpecialEffects;//特技类型
    int nSpecialEffectSpeed;//特技速度
    int nSpecialEffectDuration;//停留时间
    int nLineSpace;//行距
    int nVerPosType;//0垂直居中, 1垂直靠上, 2垂直靠下
    int nDisplayType;//显示类型, 保留
};

//Flash类型 -- 9
struct FlashAttribute{
    wchar_t wsItemName[ATT_BUFF_SIZE];//唯一标识名称
    wchar_t wsFilePath[ATT_BUFF_SIZE];//文件路径名
    int nBackColor;//背景色
    int nFlashWidth;//宽度
    int nFlashHeight;//高度
    int nFrameRate;//帧率
    int nStartFrame;//起始帧
    int nFinishFrame;//结束帧
    int nPlayLoopCount;//循环次数
};
// -----//
```



//节目属性 -- 13

```
struct ProgramAttribute{
    wchar_t wsVersion[ATT_BUFF_SIZE]; //版本
    int nDirectionX; //节目在计算机屏幕上预览窗口的起始x坐标
    int nDirectionY; //节目在计算机屏幕上预览窗口的起始y坐标
    int nScreenWidth; //节目宽度
    int nScreenHeight; //节目高度
    int nCommunicationType; //通讯类型, 0:网络, 1:串口, 2:U盘, 3:GPRS, 4:数据中心服务器, 5:FTP方式
    wchar_t wsIPAddress[ATT_BUFF_SIZE]; //IP地址
    wchar_t wsLanguage[ATT_BUFF_SIZE]; //语言类型
    int nPort; //网络端口或串口号
    int nBaudrate; //波特率
    int nUSBMode; //U盘使用模式, 0:更新节目, 1:扩展存储
    bool bFullScreen; //是否全屏, 保留
    bool bTextSmooth; //是否字体光滑, 保留
};
```

//Page属性 -- 7

```
struct PageAttribute{
    wchar_t wsPageName[ATT_BUFF_SIZE]; //节目页唯一标识
    wchar_t wsBackgroundPath[ATT_BUFF_SIZE]; //背景图片
    wchar_t wsPageVariantName[ATT_BUFF_SIZE];
    int nWindowAmount; //窗口数目
    int nBackgroundColor; //背景色
    //wchar_t wsBackgroundMusicPath[ATT_BUFF_SIZE]; //背景音乐
    int nPlayDuration; //播放时间
    bool bWaitFinish; //等待节目播放完毕, true:该节目页的内容全部播放超过一次则节目页结束, false:按播放时间播放节目页,
};
```

//FileWindow属性 -- 8

```
struct FileWindowAttribute{
    wchar_t wsWindowName[ATT_BUFF_SIZE]; //窗口唯一标识
    wchar_t wsBackgroundPath[ATT_BUFF_SIZE]; //背景图片
    int nWindowFrame; //边框
    int nWindowFrameColor; //边框颜色
    int nDirectionX; //窗口的起始x坐标
```



```
int nDirectionY;//窗口的起始y坐标
int nWindowWidth;//窗口宽度
int nWindowHeight;//窗口高度
};

//文本窗属性 --8--
struct TextWindowAttribute{
    wchar_t wsWindowName[ATT_BUFF_SIZE];//窗口唯一标识
    int nWindowBackColor;//背景色
    int nWindowFrame;//边框
    int nWindowFrameColor;//边框颜色
    int nDirectionX;//窗口的起始x坐标
    int nDirectionY;//窗口的起始y坐标
    int nWindowWidth;//窗口宽度
    int nWindowHeight;//窗口高度
};

//单行文本-- 10
struct SingLineTextAttribute{
    int nSpecialEffects;//特技
    int nSpecialEffectSpeed;//特技速度
    int nSpecialEffectDuration;//停留时间
    wchar_t wsBackgroundPath[ATT_BUFF_SIZE];//背景图片
    bool bPictureUseasIcon;//true:背景图片作为logo, false:背景图片作为背景
    int nVerticalOffset;//垂直偏移
    int nTextSpace;//字间距
    int nTextRotaion;//0:不旋转, 1:旋转90度, 2:旋转180度, 3:旋转270度
    int nTextStyle;//字体风格(粗体0x4, 斜体0x2, 下划线0x1)
    int nTextStyleColor;//字体颜色
};

//多行文本-- 9
struct MultiLineTextAttribute{
    wchar_t wsTextName[ATT_BUFF_SIZE];//文本唯一标识
    int nTextHorizonAligning;//0:左对齐, 1:右对齐, 2:居中对齐, 3两端对齐
    int nTextVerticalAligning;//0垂直居中, 1垂直靠上, 2垂直靠下
    int nSpecialEffects;//特技
```



```
int nSpecialEffectDuration;//停留时间
int nSpecialEffectSpeed;//特技速度
int nBackgroundColor;//背景色
wchar_t wsBackgroundPicturePath[ATT_BUFF_SIZE];//背景图片
int nTextSpace;//字间距
int nLineSpace;//行距
};

//表格窗属性 // -- 8
struct TableWindowAttribute{
    wchar_t wsWindowName[ATT_BUFF_SIZE];//窗口唯一标识
    int nWindowFrame;//边框
    int nWindowFrameColor;//边框颜色
    int nWindowBackColor;//背景色
    int nDirectionX;//窗口的起始x坐标
    int nDirectionY;//窗口的起始y坐标
    int nWindowWidth;//窗口宽度
    int nWindowHeight;//窗口高度
};

//表格属性 -- 10
struct TableAttribute{
    wchar_t wsTableName[ATT_BUFF_SIZE];//表格唯一标识
    wchar_t wsFilePath[ATT_BUFF_SIZE];//文件路径名
    int nRowAmount;//行数
    int nColumnAmount;//列数
    int nFixedRowAmount;//固定行数
    int nFixedColumnAmount;//固定列数
    int nGridLineColor;//表格颜色
    int nSpecialEffects;//特技
    int nSpecialEffectSpeed;//特技速度
    int nSpecialEffectDuration;//停留时间
};

struct DATE_TIME{
    int nYear;//年
```



```
int nMonth;//月
int nDay;//日
int nHour;//时
int nMinute;//分
int nSecond;//秒
int nWeek; //周
};
//计时窗属性 -- 16 ---
struct PeroidWindowAttribute{
    wchar_t wsWindowName[ATT_BUFF_SIZE];//窗口唯一标识
    wchar_t wsBackgroundPath[ATT_BUFF_SIZE];//背景图
    int nWindowFrame;//边框
    int nWindowFrameColor;//边框颜色
    int nWindowBackColor;//背景色
    int nDirectionX;//窗口的起始x坐标
    int nDirectionY;//窗口的起始y坐标
    int nWindowWidth;//窗口宽度
    int nWindowHeight;//窗口高度
    struct DATE_TIME definiteTime;//日期时间结构
    bool bShowDays;//是否显示天
    bool bShowHours;//是否显示小时
    bool bShowMinutes;//是否显示分钟
    bool bShowSeconds;//是否显示秒
    bool bShowMultiLine;//true:多行显示, false:单行显示
    bool bUnsignedNumber;//是否显示负数
    int nTimerType;        //正计时或者倒计时
    bool bMaxUnitOnly;
    bool bMultiView;
    bool bShowTimeUnit;
};

//数字时钟窗口属性 --29--
struct DigitalClockWindowAttribute{
    wchar_t wsWindowName[ATT_BUFF_SIZE];//窗口唯一标识
    wchar_t wsBackgroundPath[ATT_BUFF_SIZE];//背景图
    int nWindowFrame;//边框
    int nWindowFrameColor;//边框颜色
```



```
int nWindowBackColor;//背景色
int nDirectionX;//窗口的起始x坐标
int nDirectionY;//窗口的起始y坐标
int nWindowWidth;//窗口宽度
int nWindowHeight;//窗口高度
int nShowStyle;//显示类型, 1:[2003-05-30], 2:[05-30-2003], 3:[30-05-2003], 4:[2003/05/30], 5:[05/30/2003], 6:[30/05/2003], 7:[2003年5月30日]
int nHourStyle;//0:12小时制, 1:24小时制
int nYearStyle;//0:4位年, 1:2位年
bool bShowYear;//是否显示年
bool bShowMonth;//是否显示月
bool bShowDay;//是否显示日
bool bShowLunarYear;//是否显示农历年
bool bShowLunarMonth;//是否显示农历月
bool bShowLunarDay;//是否显示农历日
bool bShowWeek;//是否显示星期
bool bShowHalfDay;//是否显示半日标记
bool bShowHour;//是否显示小时
bool bShowMinute;//是否显示分钟
bool bShowSecond;//是否显示秒
bool bHourrateBefore;//true超前, false滞后
int nHourrateDays;//超前滞后天数
int nHourrateHours;//超前滞后小时
int nHourrateMinutes;//超前滞后分
int nHourrateSeconds;//超前滞后秒
int nMultiLineStyle;//0:单行显示, 1:多行显示
};

//模拟时钟窗口属性 ---25---
struct AnalogClockWindowAttribute{
    wchar_t wsWindowName[ATT_BUFF_SIZE];//窗口唯一标识
    wchar_t wsBackgroundPath[ATT_BUFF_SIZE];//背景图
    int nWindowFrame;//边框
    int nWindowFrameColor;//边框颜色
    int nWindowBackColor;//背景色
    int nDirectionX;//窗口的起始x坐标
    int nDirectionY;//窗口的起始y坐标
    int nWindowWidth;//窗口宽度
```





```
int nWindowHeight; //窗口高度
int nHourScaleColor; //时标颜色
int nHourScaleSize; //时标大小
int nHourScaleShape; //时标形状, 0: 圆形, 1: 方形, 2: 数字
int nMinuteScaleColor; //分标颜色
int nMinuteScaleSize; //分标大小
int nMinuteScaleShape; //分标形状, 0: 圆形, 1: 方形
bool bShowDate; //是否显示日期
bool bLunarCalendar; //是否显示农历
int nHourhandColor; //时针颜色
int nMinutehandColor; //分针颜色
int nSecondhandColor; //秒针颜色
int nHourrateDays; //超前滞后天数
int nHourrateHours; //超前滞后小时
int nHourrateMinutes; //超前滞后分
int nHourrateSeconds; //超前滞后秒
bool bHourrateBefore; //true超前, false滞后
int nAnalogClockShape; //表盘形状
bool bShowWeek; //有无星期标志
};

struct TemperatureHumidityWindowAttribute{
    wchar_t wsWindowName[ATT_BUFF_SIZE]; //窗口唯一标识
    int nWindowFrame; //边框
    int nWindowFrameColor; //边框颜色
    int nWindowBackColor; //背景色
    int nDirectionX; //窗口的起始x坐标
    int nDirectionY; //窗口的起始y坐标
    int nWindowWidth; //窗口宽度
    int nWindowHeight; //窗口高度
    int nFontWeight; //字体大小
    int nTextColor; //字体颜色
    int nFontStyle; //字体风格(粗体0x4, 斜体0x2, 下划线0x1)
    int nDegreeType; //温度类型, 0: 摄氏度, 1: 华氏度
    int nTempRegulate; //温度校正
    int nHumiRegulate; //湿度校正
    bool bMultiLine; //0: 单行显示, 1: 多行显示
```



```
bool bShowTemperature;//是否显示温度
bool bShowHumidity;//是否显示湿度
bool bShowTemperatureUnit;//是否显示温度单位
bool bShowHumidityUnit;//是否显示湿度单位
wchar_t wsFont[ATT_BUFF_SIZE];//字体
wchar_t wsTempTitle[ATT_BUFF_SIZE];//温度描述
wchar_t wsHumiTitle[ATT_BUFF_SIZE];//湿度描述
wchar_t wsHumidityUnit[ATT_BUFF_SIZE];//湿度单位
wchar_t wsTemperatureUnit[ATT_BUFF_SIZE];//温度单位
wchar_t wsBackgroundPath[ATT_BUFF_SIZE];//背景图片路径
};

//页面约束
struct DateConstraint{
    int nStartYear;//开始年
    int nStartMonth;//开始月
    int nStartDay;//开始日
    int nEndYear;//结束年
    int nEndMonth;//结束月
    int nEndDay;//结束日
};

struct TimeConstraint{
    int nStartHour;//开始时
    int nStartMinute;//开始分
    int nStartSecond;//开始秒
    int nEndHour;//结束时
    int nEndMinute;//结束分
    int nEndSecond;//结束秒
};

struct WeekConstraint{
    bool bMonday;//星期一
    bool bTuesday;//星期二
    bool bWednesday;//星期三
    bool bThursday;//星期四
    bool bFriday;//星期五
```



```
bool bSaturday;//星期六
bool bSunday;//星期日
};

struct BackgroundMusic{
    wchar_t wsItemName[ATT_BUFF_SIZE];
    wchar_t wsMusicFilePath[ATT_BUFF_SIZE];
};

enum COMMUNICATION_TYPE{NETWORK=1, COMPORT, USBMODE};
enum TEXT_STYLE{STANDARD=0, BOLD=0x4, ITALIC=0x2, UNDERLINE=0x1};
enum VIEW_MODE{CENTER, ZOOM, STRETCH, TILING};
enum TEXT_ROTATION{ROTATE_0, ROTATE_90, ROTATE_180, ROTATE_270};
enum TEXT_ALIGNING{ALIGNING_NONE, ALIGNING_LEFT, ALIGNING_CENTER, ALIGNING_RIGHT, ALIGNING_UP, ALIGNING_DOWN};
enum SHAPE{SHAPE_ROUND, SHAPE_RECT};
enum TIMER_STYLE{COUNT, COUNTDOWN};
enum ERROR_VALUES{ERROR_PROGRAM_EXIST = 1, ERROR_PROGRAM_ATTR, ERROR_IN_PARAM, ERROR_OUT_PARAM,
    ERROR_PAGE_NAME, ERROR_EXIST_GLOBALPAGE, ERROR_WINDOW_NAME,
    ERROR_WINDOW_NAME_NOT_FOUND, ERROR_PAGE_NAME_NOT_FOUND, ERROR_ITEM_NAME,
    ERROR_PAGE_NAME_EXIST, ERROR_WINDOW_NAME_EXIST, ERROR_TABLE_NAME_EXIST,
    ERROR_ITEM_NAME_EXIST, ERROR_ITEM_NAME_NOT_FOUND};
enum _ENUM_ACTION
{
    ENUM_TEXIAO_NO=0,
    ENUM_TEXIAO_RADIO,
    ENUM_TEXIAO_FG_UP,
    ENUM_TEXIAO_FG_DOWN,
    ENUM_TEXIAO_FG_LEFT,
    ENUM_TEXIAO_FG_RIGHT,
    ENUM_TEXIAO_FG_CORNER_RD,
    ENUM_TEXIAO_FG_CORNER_LD,
    ENUM_TEXIAO_FG_CORNER_RU,
    ENUM_TEXIAO_FG_CORNER_LU,
    ENUM_TEXIAO_FG_UD_IN,
    ENUM_TEXIAO_FG_LR_IN,
    ENUM_TEXIAO_FG_UD_OUT,
```



```
ENUM_TEXIAO_FG_LR_OUT,
ENUM_TEXIAO_FG_SZ_IN,
ENUM_TEXIAO_FG_SZ_OUT,
ENUM_TEXIAO_FG_OUT_INT,
ENUM_TEXIAO_FG_INT_OUT,
ENUM_TEXIAO_MOVEUP,
ENUM_TEXIAO_MOVEDOWN,
ENUM_TEXIAO_MOVELEFT,
ENUM_TEXIAO_MOVERIGHT,
ENUM_TEXIAO_MOVERIGHTDOWN,
ENUM_TEXIAO_MOVELEFTDOWN,
ENUM_TEXIAO_MOVERIGHTUP,
ENUM_TEXIAO_MOVELEFTUP,
STUNT_COVER_SLANT_TOP_LEFT      , // 左上角覆盖（斜）；
STUNT_COVER_SLANT_TOP_RIGHT     , // 右上角覆盖（斜）；
STUNT_COVER_SLANT_BOTTOM_LEFT   , // 左下角覆盖（斜）；
STUNT_COVER_SLANT_BOTTOM_RIGHT  , // 右下角覆盖（斜）；
STUNT_BLIND_HORIZONE            , // 水平百叶窗；每个窗口按8个页计算每页的宽度。
STUNT_BLIND_VERTICAL            , // 垂直百叶窗；每个窗口按8个页计算每页的宽度。
STUNT_MASAIC                    , // 马赛克；每个马赛克块大小按8*8像素计算。
STUNT_CIRCLE_OUT                , // 旋出。
STUNT_CIRCLE_IN                 , // 旋入。
STUNT_TURN_CIRCLE_LEFT_360      , // 左旋360度；
STUNT_TURN_CIRCLE_RIGHT_360     , // 右旋360度；
STUNT_TURN_CIRCLE_LEFT_180      , // 左旋180度；
STUNT_TURN_CIRCLE_RIGHT_180     , // 右旋180度；
STUNT_TURN_CIRCLE_LEFT_90       , // 左旋90度；
STUNT_TURN_CIRCLE_RIGHT_90      , // 右旋90度；
STUNT_SECTOR                    , // 扇形。
STUNT_ZOOM_UP                   , // 由小变大（上）；
STUNT_ZOOM_DOWN                 , // 由小变大（下）；
STUNT_ZOOM_LEFT                 , // 由小变大（左）；
STUNT_ZOOM_RIGHT                , // 由小变大（右）；
STUNT_ZOOM_CENTROL              , // 由小变大（中间）；
STUNT_ZOOM_TOP_LEFT             , // 由小变大（左上角）；
STUNT_ZOOM_TOP_RIGHT            , // 由小变大（右上角）；
STUNT_ZOOM_BOTTOM_LEFT          , // 由小变大（左下角）；
```



```
STUNT_ZOOM_BOTTOM_RIGHT      , // 由小变大（左下角）；
STUNT_STUNT_COVER_RHOMBUS_IN  , // 四周向中间（菱形）；
STUNT_STUNT_COVER_RHOMBUS_OUT , // 中间向四周（菱形）；
STUNT_STUNT_COVER_CIRCLE_IN   , // 四周向中间（圆形）；
STUNT_STUNT_COVER_CIRCLE_OUT  , // 中间向四周（圆形）；
STUNT_MERGE_HORIZONE         , // 水平合并。
STUNT_MERGE_VERTICAL         , // 垂直合并。
STUNT_CHEQUIRE_HORIZONE     , // 水平棋盘。
STUNT_CHEQUIRE_VERTICAL     , // 垂直棋盘。
STUNT_GLINT_HORIZONE         , // 水平闪动。
STUNT_GLINT_VERTICAL         , // 垂直闪动。
STUNT_INTERSECT_HORIZONE     , // 水平交叉。
STUNT_INTERSECT_VERTICAL     , // 垂直交叉。
STUNT_SNOW                   , // 飘雪。
STUNT_FALLOW_DOWN           , // 下落
STUNT_GILNT                  , // 闪烁，闪烁次数可以定义，计算机软件默认3次，0为永久闪烁。
STUNT_AGGLOMERTION_UP        , // 向上聚结。
STUNT_AGGLOMERTION_DOWN      , // 向下聚结。
STUNT_AGGLOMERTION_LEFT      , // 向左聚结。
STUNT_AGGLOMERTION_RIGHT     , // 向右聚结。
STUNT_LASER_UP               , // 向上镭射。
STUNT_LASER_DOWN             , // 向下镭射。
STUNT_LASER_LEFT             , // 向左镭射。
STUNT_LASER_RIGHT            , // 向右镭射。
STUNT_STRETCH_UP             , // 向上拉伸。
STUNT_STRETCH_DOWN           , // 向下拉伸。
STUNT_STRETCH_LEFT           , // 向左拉伸。
STUNT_STRETCH_RIGHT          , // 向右拉伸。
STUNT_CENTROL_MOVE_OUT       , // 中间移出。
STUNT_GRADUAL_CHANGE         , // 渐变效果；图像亮度从0~255按16步进变化。无灰度卡时无此选项。
STUNT_BANMATIAO_HORIZONE     , // 斑马条水平
STUNT_BANMATIAO_VERTICAL     , // 斑马条竖直
STUNT_MASAIK_SMALL           , // 马赛克；每个马赛克块大小按8*8像素计算。
STUNT_STUNT_SCROLL_LEFT      , // 向左连续移动；
STUNT_STUNT_SCROLL_UP        , // 向上连续移动；；
STUNT_STUNT_SCROLL_RIGHT     , // 向右连续移动；
STUNT_STUNT_SCROLL_DOWN      , // 向下连续移动；；
```



```
        ENUM_TEXIAO_NUMBER,  
    };  
#endif  
  
#define FONT_WEIGHT          12  
#define WINDOW_FRAME        1  
#define WINDOW_FRAME_COLOR   (RGB(0, 0, 255))  
#define WINDOW_BACK_COLOR    (RGB(0, 0, 0))  
#define FONT_COLOR           (RGB(255, 255, 255))
```