JetFileII API 文档 Ver2.10

更新记录表

版本	修改内容	修改人	修改日期
Ver1.0	第一个版本	oliny	2011-09-01
Ver1.1	修改 czReadPlayLogII 为 czReadDB	Lei	2011-11-07
Ver1.2	增加 Read/Write SysFile,FontFile,TextFile PictureFile,StringFile,ArrayPictureFile	oliny	2011-11-21
Ver1.3	整理操作接口	Lei	2013-4-3
Ver1.4	新加调用流程	Lei	2013-8-16
Ver1.5	修改 11,12 序号	Lei	2013-8-19
Ver1.6	增加 vpu 功能	Lei	2015-8-10
Ver1.7	增加 czEasyReadFile/czEasyWriteFile/cold	Lei	2016-4-5
Ver1.7	增加 Error 读取 API	Lei	2016-5-18
Ver1.9	增加切换 online/offline 功能	Lei	2016-5-23
Ver2.0	修改 czShowMsg 接口，增加一个默认参数	Lei	2016-9-2
Ver2.01	增加 czReadBrightInfo	Lei	2016-9-23
Ver2.02	增加生成 nmg 文件接口	Lei	2016-10-09
Ver2.03	EasyAPI 支持串口	Fm	2017-1-13
Ver2.04	增加 czReadStats 和 EasyAPI 增加 czShowFilesSpe	Fm	2017-10-16
Ver2.05	整理了 czGetPowerState 接口的描述	Fm	2017-12-12
Ver2.06	整理了部分接口的描述	Fm	2018-6-6
Ver2.07	增加了停车场屏的操作接口	Fm	2018-9-6
Ver2.08	增加了 czReadCurScreenshot 接口 增加了 czPlaySoundFile 接口 增加了 czAdjustVolume 接口	Fm	2018-9-24
Ver2.09	增加了 czSetEnableMulitZone 接口 增加了 czSetPageCount 接口 增加了 czDivideSpeZone 接口 增加了 czGetMulitZoneSetting 接口	Fm	2018-12-3
Ver2.10	增加了错误码的说明 增加了像点检测操作 增加了 czShowFilesII 接口	Fm	2019-6-5

一、简介：

JetFileII API 为与显示屏设备通信程序开发提供了一个强大，可扩充以及简单的接口。与显示屏通信的协议为 JetFileII 协议，它功能强大，可靠，灵活。JetFileII API 就是基于 JetFileII 协议，并封装了发送，接收数据，分包，组包等过程，也就是说，用户不必关注一些协议相关的细节和复杂的通信过程，就可以开发出功能强大的与显示屏通信的软件。

JetFileII API 函数封装在 DLL 里面，因此，必须把该 DLL 添加到相关的工程中，才可以调动 JetFileII API 函数。由于 DLL 与语言无关，所以，可以被任何支持动态链接库的编程语言调用，如 VC,VB,C#,Delphi,C++Builder,PB 等。开发平台为 WINDOWS。

排版说明：本文档以 0x 开头表十六进制数,如<0x30>,以" "或''表示 ASCII 字符。其它表示 10 进制数.

基本的数据类型定义：

INT8U: Unsigned 8 bit quantity
INT8S: Signed 8 bit quantity
INT16U: Unsigned 16 bit quantity
INT16S: Signed 16 bit quantity
INT32U: Unsigned 32 bit quantity
INT32S: Signed 32 bit quantity

显示屏路径说明：

大部分类型的显示屏目前只支持8.3文件格式,有部分能支持长文件名,请与support联系以确认您的显示屏是否支持长文件名.为了操作方便,显示里也有C,D,E,F盘的概念,一般C般是存放系统文件,播放的文件不应该放在这里,D,E,F是用于存放播放内容,E盘为RAM盘,F盘一般为扩展盘,如果没有扩展,该盘将存在.

显示屏数据盘里的一般格式如下

D,E,F
|- T :用于存放text file
|- S :用于存放string file
|- P :用于存放Picture file
|- A :用于存放Array picture file
|- F :用于存放动画,视频格式文件
|- Q :用于存放QST file
|- TEMP:用于存放临时的文件

二、API 调用流程

1.初始化接口(czInitAPI)

在调用基础操作接口之前,需要先初始 API,使用如下函数初始化 API.

函数原型

```
INT32U  czInitAPI(_czinterface* czif);
```

描述

该函数用于初始化 API. 如果有多线程的程序,每一个需要用到该 API 的线程,都需要调用一次该初始化函数.

参数

czinterface: **[in]**

控制结构,说明如下:

```
tpedef struct czinterface
{
  conn_type type;
  INT32U inited;
  INT8U  ip[32];
  INT8U   com[16];
  INT32U port;
  INT32U baudrate;
  void   *signPointer;
  void   *taskPointer;
  INT16U SrcAddr;
  INT16U DstAddr;
  INT32U Retry;
  INT32U Timeoutms;
  INT32U Stoped;
  INT16U totalSteps;
  INT16U curSteps;
  INT16U packetSize;
  INT8U  Rev[26];
  void (*pre_comm) (struct czinterface*);
  void (*post_comm) (struct czinterface*);
  void (*update_status) (struct czinterface*,INT32U code,INT32U precent);

  INT32U (*read) (void*,INT8U *buf, INT32U size, INT32U timeoutms);

  INT32U (*write)(void*,INT8U *buf, INT32U size);

  void   (*log)  (struct czinterface*,const char *file, int line, int
            level,const char *fmt, ...);
  void   (*curShowFilter) (struct czinterface*,const char *srcfile, const
                        char *dstfile,int frames,int delays);
}_czinterface;
```

conn_type type :

通信类型,定义如下

```
typedef enum
{
  COMM_UDP,
  COMM_TCP,
 COMM_COM
}conn_type;
```

inited:

API 内部使用

ip:

IP 地址,点分格式，如 169.254.10.49

com:

串口通信时用到的 com 字符，如 COM1

port:

网络通信时用到的端口号,一般的显示屏可以用 9520 端口进行通信.

baudrate:

串口通信时用到的 baudrate,数值型,如 115200

signPointer:

指针，调用者使用

taskPointer:
指针,调用者使用
SrcAddr:
源地址,对应 JetFileII 中的源地址
DstAddr
显示屏(目的)GGUU 地址,Unit Addr.在高位,Group Addr 在低位
Retry:
通信时，重试次数
Timeoutms:
通信时，超时时间，单位为 ms
Stoped:
停止通信
totalSteps:
写进度时，总进度个数,调用者使用,一般用于多个文件读写里,用这个表示总进度
curSteps:
当前写的进度所在部进度的步,调用都使用
packetSize:
通信时数据包的大小,单位为 Bytes,有效值为 64-1280
Rev[26]:
保留
void (*pre_comm) (struct czinterface*):
通信前调用的回调函数,API 在通信开始前,会回调 该函数,回调时,参数为该结构
void (*post_comm) (struct czinterface*):
通信后调用的回调函数
void (*update_status) (struct czinterface*,INT32U code,INT32U precent):
更新状态回调函数,在操作显示屏时,会用回调 这个函数更新进度,code 为 0 表示没有错误,precent 表示进度,有效值为 0-100.
INT32U (*read) (void*,INT8U *buf, INT32U size, INT32U timeoutms):
如果这个回调有设置,表示不使用 API 内部提供的网口/串口接口,需要用这个回调进行读取,如果为空，表示使用 DLL 内部的网络 / COM 类

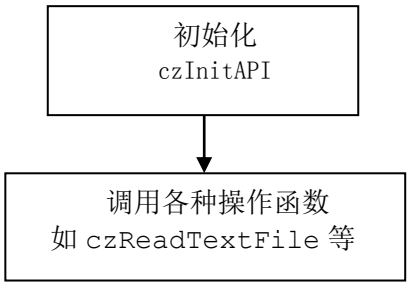
INT32U (*write)(void*,INT8U *buf, INT32U size):
如果这个回调有设置,表示不使用 API 内部提供的网口/串口接口,需要用这个回调进行写操作,如果为空，表示使用 DLL 内部的网络 / COM 类

void (*log) (struct czinterface*,const char *file, int line, int level,const char *fmt, ...):
日志回调，所有信息从这里输出
void (*curShowFilter) (struct czinterface*,const char *srcfile, const char *dstfile,int frames,int delays):
读播放内容时,会有用到该回调!

返回值
如果函数调用成功，返回 **0**，否则返回**错误代码**。
程序范例
参考应用示例节
相关函数

2.调用 API 流程

调用 API 流程如下图所示



对于 Easy API,直接调用函数即可,不需要初始化操作.

3.错误代码说明

调用 API 接口成功的时候返回 0，否则返回错误码，错误码对照表详见附录一。

程序范例

```
Char *errDesc;
INT32U errCode=0x5100;
errDesc = czErrorDesc(errCode);
```

三、基础操作接口定义

1.信息读取操作

1.1 读取系统文件(czReadSysFile)

函数原型

INT32U INT32U czReadSysFile(INT8U* systemFile,INT8U *PCPath)

描述

读回显示屏系统文件如 CONFIG.SYS,SEQUENT.SYS 等。

参数

systemFile: **[in]**

显示屏上系统文件名。

PCPath: **[in]**

读回的文件存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT8U SystemFile [64],path[256];
SystemFile = "CONFIG.SYS";
path = "C:\\CONFIG.SYS";
if(czReadSysFile(SystemFile,path)== 0)
{
    //OK
}
```

相关函数

1.2 读取显示屏字库(czReadFontFile)

函数原型

INT32U czReadFontFile (INT8U* fontName,INT8U* PCPath)

描述

读回显示屏上指定路径的文件。

参数

fontName: **[in]**

显示屏上字库名称。

PCPath: **[in]**

读回字库文件存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT8U Fontname[64],path[256];
name = "Normal11.fnt";
path = "C:\\Normal11.fnt";
if(czReadFontFile (Fontname,path)== 0)
{
    //OK
}
```

相关函数

1.3 读取指定盘符下的 Text 文件(czReadTextFile)

函数原型

INT32U czReadTextFile (INT8U Drive,INT8U* textName,INT8U* PCPath)

描述

读回显示屏上指定盘上的 Text 文件。

参数

Drive: **[in]**

读回文件在显示屏内所在盘。

textName: **[in]**

显示屏上指定 Text 文件名称。

PCPath: **[in]**

读回文件存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT8U Letter,TextName[64],path[256];
Letter = 'D';
TextName = "temp.pmg";
path = "C:\\temp.pmg";
if(czReadTextFile(Letter,TextName,path)== 0)
{
    //OK
}
```

相关函数

1.4 读取指定盘符下的 String 文件(czReadStringFile)

函数原型

INT32U czReadStringFile (INT8U Drive, INT8U* stringName, INT8U* PCPath);

描述

读回显示屏上指定盘上的 String 文件。

参数

Drive: **[in]**

读回文件在显示屏内所在盘。

stringName: **[in]**

显示屏上指定 String 文件名称。

PCPath: **[in]**

读回文件存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT8U Drive,StringName[64],path[256];
Drive = 'D';
TextName = "1";
path = "C:\\11.txt";
if(czReadStringFile (Drive, StringName,path)== 0)
{
    //OK
}
```

相关函数

1.5 读取指定盘符下的 Picture 文件(BMP 文件)(czReadPictureFile)

函数原型

INT32U czReadPictureFile (INT8U Drive, INT8U* pictureName, INT8U* PCPath);

描述

读回显示屏上指定盘符中的 picture 文件。

参数

Drive: **[in]**

读回文件在显示屏内所在盘。

pictureName: **[in]**

显示屏上指定 Text 文件名称。

PCPath: **[in]**

读回文件存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT8U Letter,PictureName[64],path[256];
Letter = 'D';
PictureName = "12.bmp";
path = "C:\\21.bmp";
if(czReadPictureFile (Letter,PictureName,path)== 0)
{
}
```

相关函数

1.6 读取指定盘符下的 ArrayPicture 文件(PMG 文件)(czReadArrPicFile)

函数原型

INT32U czReadArrPicFile (INT8U Drive, INT8U* arrPicName, INT8U* PCPath)

描述

读回显示屏上指定盘符中的 pmg 文件。

参数

Drive: **[in]**

读回文件在显示屏内所在盘。

arrPicName: **[in]**

显示屏上指定 Text 文件名称。

PCPath: **[in]**

读回文件存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT8U Letter,ArrPicName[64],path[256];
Letter = 'D';
ArrPicName = "temp.pmg";
path = "C:\\ temp.pmg";
if(czReadArrPicFile(Letter,ArrPicName,path)== 0)
{
}
```

相关函数

1.7 读取指定路径的文件(czReadSpecPathFile)

函数原型

```
INT32U czReadSpecPathFile (INT8U* SpecialFile,INT8U* PCPath);
```

描述

读回显示屏上指定路径的文件。

参数

SpecialFile: **[in]**

显示屏上指定文件和名称。

PCPath: **[in]**

读回文件存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT8U name[64],path[256];
name = "D:\\A\\temp.pmg";
path = "D:\\temp.pmg";
if(czReadSpecPathFile (name,path)== 0)
{
}
```

相关函数

1.8 系统参数读取(czReadSystemSet)

函数原型

```
INT32U czReadSystemSet (SYSTEM_SET* systemSet);
```

描述

读取显示屏参数。

参数

systemSet: **[out]**

读回的显示屏参数结构体。

typedef struct
{
  INT16U cpuv;
  INT16U tcpv;
  INT16U filesv;
  INT16U fpga;
  INT16U height;
  INT16U width;
  INT16U protocol;
  INT8U gg;
  INT8U uu;
}SYSTEM_SET;

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
SYSTEM_SET systemSet;
if(czReadSystemSet(&systemSet)== 0)
{
}
```

相关函数

1.9 系统当前状态读取(czReadCurrentState)

函数原型

```
INT32U czReadCurrentState (CURRENT_STATE* State void *ExtData, INT32U DataSize);
```

描述

读取系统当前所处的状态,如温度，播放模式等

参数

currentState: **[out]**

读回的当前屏状态。

结构体大小

typedef struct
{
  INT8U sysState;
  INT8U inTemp;
  INT8U outTemp;
  INT8U AutoPower;
  INT8U Humidity;
  INT8U Samples;
  INT8U BrightLevel;
}CURRENT_STATE;

sysState:

系统状态任

务调度状态 = 0
紧急消息播放状态 = 1
黑屏状态 = 2
遥控状态 = 3
测试状态 = 4
无限连接播放状态 = 5
头尾接龙播放状态 = 6

备份状态 = 7
关机状态 = 8
高温保护状态 = 9

inTemp:
屏内温度（℃），没有就为 0xFF
outTemp:
屏外温度（℃），没有就为 0xFF
AutoPower:
自动开关机状态,1=允许；0=禁止
Humidity:
湿度
Samples:
光控传感头采样值
BrightLevel:
当前所在的亮度级别[0-100],100 为最亮

ExtData: **[out]**
附加返回值

DataSize
附加返回值最大大小

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
CURRENT_STATE currentState;
if(czReadCurrentState(&currentState,NULL,0) != 0)
{
}
```

相关函数

1.10 读取系统 SN,MAC 信息(czGetSNMAC)

函数原型

INT32U czGetSNMAC (INT8U* SN,INT32U SNLen,INT8U* MAC,INT32U MACLen)

描述

读取显示屏的 SN 以及 MAC 信息。

参数

SN: [out]
显示屏序列号。
SNLen: **[in]**
序列号长度。
MAC:[out]
显示屏 MAC 地址。
MACLen: **[in]**
MAC 长度。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT8U SN[12],MAC[7];

if(czGetSNMAC(SN,12,MAC,7) == 0)
{
}
```

相关函数

1.11 读取默认显示样式(czReadDefDisplayStyle)

函数原型

```
INT32U czReadDefDisplayStyle(DEFAULT_SET* defaultSet)
```

描述

读取系统默认显示样式,这个样式主要是用来控制显示花样,颜色等.

参数

defaultSet : **[out]**

读回的系统默认显示样式结构。

```
Struct
{
  UWORD ID;            //0x55aa
  UBYTE PlayListLoc;  //列表保存位置
  UBYTE TimePre0En;   //前置 0
  UBYTE Ddrive;        //默认盘
  UBYTE Dback_color;  //默认背景色
  UBYTE Dfont_color;  //默认前景色
  UBYTE Dhor_just;    //水平对齐
  UBYTE Dver_just;    //垂直对齐
  UBYTE Dline_space; //行间距
  UBYTE Dfont;       //字体
  UBYTE Din_mode;   //入模式
  UBYTE Dout_mode;  //出模式
  UBYTE Dspeed;     //速度
  UBYTE Dstay_time;  //停留时间
  UBYTE Dwrap;      //自动换行
   INT32U Lstay_time;
   INT8U TimeFormat;
   INT8U rev[31];
}DEFAULT_SET
```

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
  DEFAULT_SET defaultSet;
  if(czReadDefDisplayStyle(&defaultSet) == 0)
  {
  }
```

相关函数

1.12 读取指定路径文件扩展(czReadSpecPathFileEx)

函数原型

```
INT32U czReadSpecPathFileEx (INT8U* SpecialFile,INT8U* PCPath)
```

描述

读取已知路径和文件名的文件，是 czReadSpecPathFile 命令的扩展,主要是用来弥补 czReadSpecPathFile 命令不能读大于 64M 文件的缺陷

参数

SpecialFile: **[in]**

显示屏上指定文件和名称。

PCPath: **[in]**

读回文件存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT8U name[64],path[256];
name = "D:\\A\\temp.pmg";
path = "D:\\temp.pmg";
if(czReadSpecPathFileEx(name,path)== 0)
{
}
```

相关函数

1.13 系统信息读取(czReadSystemInfo)

函数原型

INT32U czReadSystemInfo (INT8U* Info,INT32U BufLen,INT32U* size)

描述

读取系统信息,信息具体的内容由 Firmware 决定.

参数

Info: **[out]**

读回的 FirmWare 信息内容。

BufLen: **[in]**

缓冲区的长度。

size: **[out]**

信息内容长度。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT8U Buf[2048];
INT32U size;
if(czReadSystemInfo(Buf, 2048,&size) == 0)
{
}
```

相关函数

1.14 数据库内容读取(czReadDB)

函数原型

INT32U czReadDB(INT8U cmdType,INT8U* sql,INT8U* Field,INT32U FiledLen, INT8U* Record,INT32U RecordLen)

描述

读取系统中数据库的数据，可以用于读取播放日志.

参数

cmdType: **[in]**

开始箱体号(逻辑,从 0 开始). 1=SELECT 语句

sql: **[in]**

SQL 语句。

Field: **[out]**

返回的字段名集合，反回时分为三段，该字段前 2 字节表示 Record 中记录的条数,接着的两个字节表示字段个数,其它字节表示字段名集合，名称以“<>”包括起来.

FiledLen: **[in]**
Field 缓冲区大小。
Record: **[out]**
记录，每个字段以"<>"括起来,最大不超过 1200Bytes
RecordLen: **[in]**
Record 缓冲区大小。

返回值
如果函数调用成功，返回 **0**，否则返回**错误代码**。
程序范例

```
INT8U cmd;
INT8U sql[256],Field[1024],Record[4096];
cmd = 1;
sql = "select * from PlayLog limit 20 offset 1";
if(czReadDB(cmd,sql,Field,1024,Record,4096) == 0)
{
   printf("record number = %d, filed number =%d",Field[0]+Field[1]<<8,
   Field[3]+Field[4]<<8;
}
```

相关函数

1.15 亮度信息读取(czReadBrightInfo)

函数原型
INT32U czReadBrightInfo (INT8U* BrightType,INT8U* Percent,INT8U* ADValue)
描述
读取系统的亮度信息;
参数
BrightType: **[out]**
当前亮度类型。0=自动,1=手动,2=调度
Percent: **[out]**
亮度百分比，有效值[1-100]，100 表示最亮
ADValue: **[out]**
当前 AD 值，类型为自动时有效
返回值
如果函数调用成功，返回 **0**，否则返回**错误代码**。
程序范例

```
INT8U BrightType,Percent,ADValue;
if(czReadBrightInfo(&BrightType,&Percent,&ADValue) == 0)
{
}
```

相关函数

1.16 播放日志读取(czReadPlayLog)

函数原型
INT32U czReadPlayLog (INT8U* PCPath);
描述
读回播放日志到文件中，这个命令在 5800 主板上不支持，5800 的日志请用 czReadDB 读取.文件协议请参考 JetFileII 协议文档.
参数
PCPath: **[in]**

保存的文件在 PC 上的路径及文件名。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
if(czReadPlayLog ("C:\\plalog.log") == 0)
{
}
```

相关函数

1.17 显示屏属性初始化(czInitProperty)

函数原型

```
INT32U czInitProperty ();
```

描述

这个命令用于从屏体加载属性，并初始化在内存.需要用 czGetProperty 获取特定的属性.

参数

none

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
if(czInitProperty () == 0)
{
}
```

相关函数

czGetProperty

1.18 显示屏属性读取(czGetProperty)

函数原型

```
INT32U czGetProperty(void *buf, INT32U bufsize, INT32U type);
```

描述

读取保存在内存中的特殊，用这个函数前，请先调用 czInitProperty 初始化

参数

buf: **[in/out]**

获取属性的缓冲区。

bufsize:**[in]**

获取属性的缓冲区大小。

type: **[in]**

获取属性类型。定义如下:

```
#define PROPERTY_IS_F_DRIVE_EXIST     1        //查询显示屏是否存在F盘
#define PROPERTY_PLAYLIST_SAVE_TO     2        //查询playlist保存位置
#define PROPERTY_GET_DEFAULT          3        //获得默认设置结构
```

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT32U f_drive_exist;
czInitProperty();
if(CZ_ERROR_OK != czGetProperty(&f_drive_exist, 4, PROPERTY_IS_F_DRIVE_EXIST))
{
  //Error
}
```

相关函数

czInitProperty

1.19 显示屏警告信息 xml 文件读取(czReadWarning)

函数原型

```
INT32U czReadWarning (INT8U* pcpath)
```

描述

读取保存在屏中的警告信息 XML 文件,需要程序自己处理。

参数

pcpath: **[in/out]**

读取 waring xml 文件保存的路径和文件名

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
if(CZ_ERROR_OK != czReadWarning ((INT8U*)" f:\\WARING.XML" )
{
  //Error
}
```

相关函数

czReadWaringReport, czReadPixelErrorReport

1.20 显示屏警告信息读取(czReadWaringReport)

函数原型

```
INT32U czReadWaringReport(INT32U (*cb)(ERROR_INFO_STRUCT *errorInfo));
```

描述

读取显示屏中的警告信息，每条警告，会回调该 cb.

参数

cb: **[in/out]**

错误信息回调函数

```
//错误报告错误信息结构体
typedef struct
{
  error_type type;                                    //错误类型
  INT8U x;                                            //错误箱体所在列数，从0开始
  INT8U y;                                            //错误箱体所在行数，从0开始
  INT8U problem[64];                                      //问题
```

```
  INT8U description[256];                                    //问题描述
  INT8U time[64];                                            //时间
}ERROR_INFO_STRUCT;

//错误报告错误类型
typedef enum
{
  ERROR_MAINBOARD,
  ERROR_TILE,
  ERROR_PIXEL,
  ERROR_SIGNAL
}error_type;
```

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT32U cb(ERROR_INFO_STRUCT *errorInfo)
{
    //process err
 }

if(CZ_ERROR_OK != czReadWaringReport(cb))
{
  //Error
}
```

相关函数

czReadWaring, czReadPixelErrorReport

1.21 显示屏像点错误读取(czReadPixelErrorReport)

函数原型

```
INT32U czReadPixelErrorReport(INT32U (*cb)(PixelInfo *PI))
```

描述

读取显示屏中的像点检测信息，每条错误，会回调该 cb.

参数

cb: **[in/out]**

错误信息回调函数

```
//读取像点检测时
typedef struct
{
  error_type Type;          //错误类型：ERROR_PIXEL，ERROR_SIGNAL
  INT32U X;
  INT16U Y;
  INT8U  Result[100];
```

```
  INT8U  RecordDT[20];
}PixelInfo;
```

X:为X方向位置，从0开始，从左到右。

Y:为Y方向位置，从0开始，从上到下。

屏左上角是原点(0,0)，向右为X增加，现下是Y增加。对于像点错误来说，每个单位为一个像点，对信号来说，一个模块为一个单位。

Result:表示结果

像点错误表示为

R,G,B Error,其中R,G,B表示颜色，只有哪种颜色有错才会写

如只有Red错误，哪表示为：R Error

只有Red, Grenn有错,表示为：R,G Error

RecordDT:报告时间，格式为：YYYY-MM-DD HH:mm

```
//错误报告错误类型
typedef enum
{
  ERROR_MAINBOARD,
  ERROR_TILE,
  ERROR_PIXEL,
  ERROR_SIGNAL
}error_type;
```

回调结果示例说明:

假如屏有有以下几个位置像点错误

X	Y	Error	DateTime
14	68	R Error	2016-06-24 18:05
9	68	G Error	2016-06-24 18:05
0	65	B Error	2016-06-24 18:05

还有以下几个信号错误

Signal Error

Port	X	Y	Error	DateTime
0	0	9	R,G,B Error	2016-06-24 18:05
0	1	9	R,G,B Error	2016-06-24 18:05

回调函数会回调 5 次，每一次 PixelInfo 数据如下：

第一个点回调参数值（像点）

PI	0x010fcf88 {Type=ERROR_PIXEL X=0x0000000e Y=0x0044 ...}
Type	ERROR_PIXEL
X	0x0000000e
Y	0x0044
Result	0x010fcf92 "R Error"
RecordDT	0x010fcff6 "2016-06-24 18:05"

第二个点回调参数值 （像点）

PI	0x010fcf88 {Type=ERROR_PIXEL X=0x00000009 Y=0x0044 ...}
Type	ERROR_PIXEL
X	0x00000009
Y	0x0044
Result	0x010fcf92 "G Error"
RecordDT	0x010fcff6 "2016-06-24 18:05"

第 3 个点回调参数值 （像点）

PI	0x010fcf88 {Type=ERROR_PIXEL X=0x00000000 Y=0x0041 ...}
Type	ERROR_PIXEL
X	0x00000000
Y	0x0041
Result	0x010fcf92 "B Error"
RecordDT	0x010fcff6 "2016-06-24 18:05"

第 4 个点回调参数值 （信号错误）

PI	0x010fc1bc {Type=ERROR_SIGNAL X=0x00000000 Y=0x0009 ...}
Type	ERROR_SIGNAL
X	0x00000000
Y	0x0009
Result	0x010fc1c6 "R,G,B Error"
RecordDT	0x010fc22a "2016-06-24 18:05"

第 5 个点回调参数值 （信号错误）

PI	0x010fc1bc {Type=ERROR_SIGNAL X=0x00000001 Y=0x0009 ...}
Type	ERROR_SIGNAL
X	0x00000001
Y	0x0009
Result	0x010fc1c6 "R,G,B Error"
RecordDT	0x010fc22a "2016-06-24 18:05"

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT32U cb(PixelInfo *errorInfo)
{
    //process err
 }

if(CZ_ERROR_OK != czReadPixelErrorReport(cb))
{
  //Error
}
```

相关函数

czReadWaringReport, czReadWaring

1.22 亮度信息读取(czReadBrightInfoExt)

函数原型

```
INT32U czReadBrightInfoExt (czBrightInfoExt * BrightInfo)
```

描述

读取系统的亮度信息;可以读取 2 个亮度.

参数

BrightInfo: **[out]**

```
typedef struct
{
    INT8U  BrightType;
    INT8U  BrightPrecent1;
    INT16U BrightSensorAD1;
    INT8U  Rev;
    INT8U  BrightPrecent2;
    INT16U BrightSensorAD2;
}czBrightInfoExt;
```

BrightType: 当前亮度类型。0=自动,1=手动,2=调度

BrightPrecent1/ BrightPrecent2: 亮度百分比，有效值[1-100]，100 表示最亮

BrightSensorAD1/ BrightSensorAD2: 当前 AD 值，类型为自动时有效

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
czBrightInfoExt bri;
if(czReadBrightInfoExt(&bri) == 0)
{
}
```

相关函数

czReadBrightInfo

1.23 主板状态读取(czReadStats)

函数原型

INT32U czReadStats (INT8U statsType,INT8U* buf,INT32U bufSize)

描述

读取主板状态信息,信息具体的内容由 Firmware 决定.

参数

statsType: **[in]**

读取主板状态的类型。

0:请求所有状态	1:温度(内)	2:温度(外)	3:湿度(内)
4:湿度(外)	5:倾斜角度	6:风向	7:风速
8:亮度	9:开门状态	10:电源状态	11:firmware 版本号
12:像点错误个数			

buf: **[out]**

读回的 FirmWare 状态信息内容。

bufSize: **[in]**

缓冲区的长度。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT8U doorStat = 0;
if(czReadStats(9, &doorStat, 1) == 0)
{
}
```

相关函数

2．信息写入操作

2.1 写入系统文件到显示屏(czWriteSystemFile)

函数原型

```
INT32U czWriteSystemFile (INT8U* FileName,INT8U* PCPath)
```

描述

写文件到指定的路径。应用此函数写入 CONFIG.SYS 文件时需谨慎, 应在比较完全熟悉 CONFIG.SYS 文件组成的情况下才能使用.

参数

FileName：[in]

系统文件名。"CONFIG.SYS", "SEQUENT.SYS"

PCPath: [in]

文件在 PC 上的存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单 2.8
INT8U FileName[128],path[256];
FileName = "CONFIG.SYS";
Path = "c:\\CONFIG.SYS";
if(czWriteSystemFile(FileName,Path) == 0)
{
}
```

相关函数

2.2 写入字库到显示屏 (czWriteFontFile)

函数原型

```
INT32U czWriteFontFile(INT8U* FontName,INT8U* PCPath);
```

描述

写入字库或字体列表到显示屏。一般不建议调用,如果想改字库,请用 sigma3000 软件.

参数

FontName：[in]

要写入的字库文件在显示屏的名称。

PCPath: [in]

字库文件在 PC 上的存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单 2.8
INT8U FileName[128],path[256];
FileName = "Normal11.fnt";
Path = "c:\\Normal11.fnt";
if(czWriteFontFile(FileName,Path) == 0)
{
}
```

相关函数

2.3 写 Text 文件到显示屏 (czWriteTextFile)

函数原型

```
INT32U czWriteTextFile(INT8U Drive,INT8U* TextName,INT8U* PCPath);
```

描述

写 Text 文件到显示屏的指定盘下。

参数

Drive：[in]

指定盘。

TextName：[in]

要写入的 Text 文件名。

PCPath: [in]

文件在 PC 上的存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单 2.8
INT8U Letter,FileName[128],path[256];
Letter = 'D';
FileName = "Temp.NMG";
Path = "C:\\Temp.NMG";
if(czWriteTextFile(Letter,FileName,Path) == 0)
{
}
```

相关函数

2.4 写 String 文件到显示屏 (czWriteStringFile)

函数原型

```
INT32U czWriteStringFile(INT8U Drive,INT8U* StringName,INT8U* PCPath);
```

描述

写 String 文件到显示屏的指定盘下。

参数

Drive：[in]

指定盘。

StringName：[in]

要写入的 String 文件名。

PCPath: [in]

文件在 PC 上的存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单 2.8
INT8U Letter,FileName[128],path[256];
Letter = 'D';
FileName = "1";
Path = "C:\\11.txt";
if(czWriteStringFile(Letter,FileName,Path) == 0)
{
}
```

相关函数

2.5 写 Picture 文件到显示屏 (czWritePictureFile)

函数原型

INT32U czWritePictureFile(INT8U Drive,INT8U* PictureName,INT8U* PCPath);

描述

写 picture 文件到显示屏的指定盘下.

参数

Drive：[in]

指定盘。

PictureName：[in]

要写入的 Picture 文件名。

PCPath: [in]

文件在 PC 上的存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单 2.8
INT8U Letter,FileName[128],path[256];
Letter = 'D';
FileName = "12.bmp";
Path = "C:\\12.bmp";
if(czWritePictureFile(Letter,FileName,Path) == 0)
{
}
```

相关函数

2.6 写 ArrayPicture 文件(PMG 文件)到显示屏 (czWriteArrPicFile)

函数原型

INT32U czWriteArrPicFile(INT8U Drive,INT8U* ArrPicName,INT8U* PCPath);

描述

写 ArrayPicture 文件到显示屏的指定盘下。

参数

Drive：[in]

指定盘。

ArrPicName：[in]

要写入的 Text 文件名。

PCPath: [in]

文件在 PC 上的存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单 2.8
INT8U Letter,FileName[128],path[256];
Letter = 'D';
FileName = "Temp.PMG";
Path = "C:\\Temp.PMG";
if(czWriteArrPicFile(Letter,FileName,Path) == 0)
{
}
```

相关函数

2.7 写文件到指定的路径 (czWriteSpecFile)

函数原型

```
INT32U czWriteSpecFile(INT8U* SpecialFile,INT8U* PCPath)
```

描述

写文件到指定的路径。文件大小需<64MB.如果超过,请用扩展函数.

参数

SpecialFile：[in]

要写入的文件指定路径。

PCPath: [in]

文件在 PC 上的存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单 2.8
INT8U FileName[128],path[256];
FileName = "D:\\T\\Temp.NMG";
Path = "D:\\ Temp.NMG";
if(czWriteSpecFile(FileName,Path) == 0)
{
}
```

相关函数

2.8 紧急消息写入 (czWriteUrgentMsg)

函数原型

```
INT32U czWriteUrgentMsg(INT8U StayTime, INT8U SoundSwitch,INT8U*
TextMsg,INT32 MsgLen)
```

描述

紧急消息写入。

参数

StayTime：[in]

生存时间,0 表示永久播放,单位为秒。

SoundSwitch：[in]

声音开关。1=开，0 = 关

TextMsg: [in]

紧急消息数据(Text File 数据)，不能大于 1024 个字节。

MsgLen: [in]

消息长度。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单 2.9
INT8U StayTime,SoundSwitch,Msg[1024];
StayTime = 0;
SoundSwitch = 1;
Msg = "Welcome"
if(czWriteUrgentMsg(StayTime,SoundSwitch,Msg,sizeof(Msg)) == 0)
{
}
```

相关函数

2.9 亮度控制块写入 (czWriteBrightCtrlBlock)

函数原型

```
INT32U czWriteBrightCtrlBlock(BRIGHT_CTRL* brightCtrl)
```

描述

写入亮度控制块。对没有亮度控制的屏不起作用。

参数

brightCtrl：[in]

亮度控制块结构体信息。

```
typedef struct
{
 INT16U x;   //X 位置
 INT16U Y;   //Y 位置
 INT8U  Red;
 INT8U  Green;
 INT8U  Blue;
 INT16U Width;
 INT16U Height;
}BRIGHT_CTRL;
```

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单 2.10
BRIGHT_CTRL brightCtrl;
brightCtrl.x = 0;
brightCtrl.y = 0;
brightCtrl.Red = 0xFF;
brightCtrl.Green = 0xFF;
brightCtrl.Blue  = 0xFF;
brightCtrl.Width = 64;
brightCtrl.Height = 32;
if(czWriteBrightCtrlBlock(&brightCtrl) == 0)
{
}
```

相关函数

2.10 写入默认显示样式 (czWriteDefDisplayStyle)

函数原型

```
INT32U czWriteDefDisplayStyle(DEFAULT_SET* defaultSet)
```

描述

读取主板绝对地址信息。

参数

defaultSet：[in]

默认结构体信息。

```
Struct
{
  UWORD ID;              //55aa
  UWORD Rev;            //保留
  UBYTE Ddrive;        //默认盘
  UBYTE Dback_color;   //默认背景色
  UBYTE Dfont_color;   //默认前景色
  UBYTE Dhor_just;     //水平对齐
```

```
    UBYTE Dver_just;      //垂直对齐
    UBYTE Dline_space;    //行间距
    UBYTE Dfont;          //字体
    UBYTE Din_mode;       //入模式
    UBYTE Dout_mode;      //出模式
    UBYTE Dspeed;         //速度
    UBYTE Dstay_time;     //停留时间
    UBYTE Dwrap;          //自动换行
    INT32U Lstay_time;
    INT8U TimeFormat;
    INT8U rev[31];
  }DEFAULT_SET
```

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单 2.12
DEFAULT_SET defaultSet;
if(czReadDefDisplayStyle(&defaultSet) == 0)
{
   defaultSet.Ddrive = "D";
   if(czWriteDefDisplayStyle(defaultSet) == 0)
   {
   }
}
```

相关函数

2.11 写文件到指定的路径－扩展 (czWriteSpecFileEX)

函数原型

```
INT32U czWriteSpecFileEX(INT8U* SpecialFile,INT8U* PCPath)
```

描述

写文件到指定的路径,扩展函数,可以用于写>64MB , < 2GB 的文件.

参数

SpecialFile：[in]

要写入的文件指定路径。

PCPath: [in]

文件在 PC 上的存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单 2.13
INT8U FileName[128],path[256];
FileName = "D:\\T\\Temp.NMG";
Path = "D:\\ Temp.NMG";
if(czWriteSpecFileEX(FileName,Path) == 0)
{
}
```

相关函数

2.12 写文件 CRC 到屏中保存 (czWriteCRCForFile)

函数原型

```
INT32U czWriteCRCForFile(INT8U* FileName,INT32U FileSize,INT16U FileCRC)
```

描述

写一个文件的 CRC 到显示屏中保存，软件可以读取 D:\FILELST.SYS 文件，该文件记录了每个在屏中文件的和校验，如果和校验一样表示屏中已保存了该文件（不需要发送）。

注意：在先判断功能列表中是否支持”文件 CRC “项打开，否则 firmware 不支持该命令

参数

FileName：[in]
文件名。
FileSize：[in]
文件大小.
FileCRC: [in]
文件 CRC

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单 2.14
INT8U File[128] = “D:\\T\\TEMP.NMG”;
INT32U FileSize = GetFileSize(File);
INT16U FileCRC  = GetFileCRC(File);
if(czWriteCRCForFile(File,FileSize,FileCRC) == 0)
{
}
```

相关函数

3．测试功能

3.1 连接测试 (czConnectTest)

函数原型

INT32U czConnectTest(INT16U* FirmwareVer, INT16U* FPGAVer, INT32U* IPAddress)

描述

测试屏体有没有连接上，如果连上，就返回屏体的一些参数。

参数

FirmwareVer：[out]
返回程序版本。
FPGAVer：[out]
返回 FPGA 版本。
IPAddress：[out]
返回屏 IP 地址。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT16U FirmwareVer,FPGAVer;
INT32U ip;
if(czConnectTest(&FirmwareVer,&FPGAVer,&ip)==0)
{
}
```

相关函数

3.2 显示模式测试 (czPatternTest)

函数原型

INT32U czPatternTest(INT32U PatternMode)

描述

让显示进入特殊的测试模式，用于检测显示屏是否有问题。

参数

PatternMode：**[in]**

测试类型。

2=Auto
3=全亮
4=Red
5=Green
6=Blue
7=Horizontal
8=Vertical

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U TestType=2;
if(czPatternTest(TestType)==0)
{
}
```

相关函数

3.3 结束测试 (czStopTest)

函数原型

```
INT32U czStopTest()
```

描述

结束屏体测试状态。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czStopTest ()==0)
{
}
```

相关函数

3.4 灰度测试 (czCrayTest)

函数原型

```
INT32U czCrayTest(GRAY_TEST_PARAM*  grayParam)
```

描述

屏体灰度测试

参数

grayParam: **[in]**

灰度测试参数

```
typedef struct
{
  INT8U Type;    0=固定, 1=渐变
  INT8U Red;     0=没有 Red 分量, 1=有 Red 分量。
  INT8U Green;   0=没有 green 分量, 1=有 gree 分量。
```

```
    INT8U Blue;    0=没有 blue 分量, 1=有 blue 分量。
    INT16U Level;  1-256 的灰度级别
  } GRAY_TEST_PARAM;
```

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
GRAY_TEST_PARAM grayParam;
grayParam.Type = 1;
grayParam.Red  = 1;
grayParam.Green = 1;
grayParam.Blue = 1;
grayParam.Level = 256;
if(czCrayTest(&grayParam)==0)
{
}
```

相关函数

3.5 颜色测试 (czColorTest)

函数原型

```
INT32U czColorTest(INT8U red, INT8U green, INT8U blue)
```

描述

使屏体进入指定的颜色测试状态。

参数

red: **[in]**
Red 分量 [0-255]
green: **[in]**
green 分量 [0-255]
blue: **[in]**
blue 分量 [0-255]

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U r,g,b;
r=255;g=255;b=255;
if(czColorTest(r,g,b)==0)
{
}
```

相关函数

3.6 指定区域测试(czAreaTest)

函数原型

```
INT32U czAreaTest(AREA_STRUCT areaStruct)
```

描述

使屏体指定区域进入颜色测试状态。

参数

areaStruct: **[in]**
区域以及颜色指定。
typedef struct
{

```
    INT16U Beginx;
    INT16U Beginy;
    INT16U Endx;
    INT16U Endy;
    INT16U Intensity;
    INT8U Pattern;
    INT8U Red;
    INT8U Green;
    INT8U Blue;
    INT16U modWidth;
    INT16U modHeight;
}AREA_STRUCT;    // 各个参数含义请参考 JetFileII 协议。
```

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
AREA_STRUCT areaStruct;
Init struct …
if(czAreaTest(areaStruct)==0)
{
}
```

相关函数

4．系统操作

4.1 黑屏 (czBlackScreen)

函数原型

```
INT32U czBlackScreen()
```

描述

使屏体进入黑屏状态。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czBlackScreen()==0)
{
}
```

相关函数

4.2 结束黑屏 (czEndBlackScreen)

函数原型

```
INT32U czEndBlackScreen()
```

描述

使屏体结束黑屏状态。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czEndBlackScreen()==0)
{
}
```

相关函数

4.3 系统复位 (czResetSystem)

函数原型

```
INT32U czResetSystem()
```

描述

使显示屏重新启动。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czResetSystem()==0)
{
}
```

相关函数

czResetSystemCool

4.4 关闭/打开显示屏 (czPowerOnOff)

函数原型

```
INT32U czPowerOnOff(INT8U OnOFF)
```

描述

打开/关闭显示屏。

参数

OnOFF: **[in]**

0=关闭,1=打开

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U bDisplay;
bDisplay = 1;
if(czPowerOnOff(bDisplay)==0)
{
}
```

相关函数

4.5 读取显示屏开关机状态 (czGetPowerState)

函数原型

```
INT32U czGetPowerState(INT8U* MomState,INT8U* DriverState)
```

描述

读取屏体开关状态,可用于检查主板和驱动板开关状态情况。

参数

MomState: **[out]**

主板开关状态 0 = 打开状态 1= 关闭状态(测试时黑屏，需主板程序支持) 2=关闭状态(关屏)

DriverState: **[out]**
驱动板开关状态 1 = 关闭状态, 0 = 打开状态

返回值
如果函数调用成功，返回 **0**，否则返回**错误代码**。
程序范例

```
//代码清单
INT8U bFalg1,bFlag2;
if(czGetPowerState(&bFalg1,&bFlag2)==0)
{
}
```

相关函数

4.6 修改通信波特率 (czChangeBaudRate)

函数原型
INT32U czChangeBaudRate(INT8U BaudRate)
描述
动态更改显示参数，但此修改不会保存，掉电后就会丢失。部分主板支持.
参数
BaudRate: **[in]**
动态修改显示屏波特率。0 = 115200 1 = 57600 2 = 38400 3 = 19200 4 = 9600 5 = 4800 6 = 2400 7 = 1200 8 = 600

返回值
如果函数调用成功，返回 **0**，否则返回**错误代码**。
程序范例

```
//代码清单
INT8U BaudRate = 0;
if(czChangeBaudRate(BaudRate)==0)
{
}
```

相关函数

4.7 亮度调节 (czBrightAdjust)

函数原型
INT32U czBrightAdjust(INT8U bright)
描述
更改显示屏亮度。
参数
bright: **[in]**
亮度值。有效值为[0-100] 0 = 自动亮度 1-100 表示亮度级别，1 为最暗，100 为最亮

返回值
如果函数调用成功，返回 **0**，否则返回**错误代码**。
程序范例

```
//代码清单
INT8U bright = 0;
if(czBrightAdjust(BaudRate)==0)
{
}
```

相关函数

4.8 设置黄闪提示灯状态 (czSetBeacon)

函数原型

INT32U czSetBeacon(BEACON_ST* beacon)

描述

控制黄闪指示灯状态

参数

beacon: **[in]**

灯设置参数，如下定义。

```
typedef struct
{
  INT8U R;
  INT8U Y;
  INT8U G;
  INT8U Mode;
  INT8U Rev[4];
}BEACON_ST;
```

说明：R,Y,G 分别控制 Red,Yellow,Green 灯的开关，0xff 表示关闭,0x01 表示开,Mode 控制黄闪灯的模式，0x01=全闪,0x02=左右闪,0x03=上下闪,0x04=全亮,0x05=全灭

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
BEACON_ST  beacon;
beacon.R = 1;
beacon.G = 1;
beacon.Y = 1;
beacon.Mode = 0x03;

if(czSetBeacon(indiStatus)==0)
{
}
```

相关函数

4.9 读取指示灯状态 (czGetBeacon)

函数原型

INT32U czGetBeacon(BEACON_ST* beacon)

描述

读取指示灯状态。

参数

beacon: **[out]**

读回的指示灯状态参数。定义参见 czSetBeacon 函数。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
BEACON_ST beacon;
if(czGetBeacon(&beacon)==0)
{
```

```
}
```

相关函数

4.10 系统复位 (czResetSystemCool)

函数原型

```
INT32U czResetSystemCool()
```

描述

使显示屏重新启动。这个是冷启动，czResetSystem 是热启动

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czResetSystemCool()==0)
{
}
```

相关函数

czResetSystem

4.11 同步/脱机切换 (czSwitchOnlineOffline)

函数原型

```
INT32U czSwitchOnlineOffline(INT8U OnlineOffline)
```

描述

打开/关闭显示屏。

参数

OnlineOffline: **[in]**
0=Online,1=offline

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Online=0;
if(czSwitchOnlineOffline (Online)==0)
{
}
```

相关函数

5. 时间

5.1 时间读取 (czReadLEDTime)

函数原型

```
INT32U czReadLEDTime(INT16U *Y, INT8U *M, INT8U *D, INT8U *H, INT8U *MM, INT8U *S, INT8U *W, INT8U
                     *TZ)
```

描述

读取显示屏时间。

参数

Y: **[out]**
年
M:**[out]**
月
D:**[out]**
日
H:**[out]**
时
MM:**[out]**
分
S:**[out]**
秒
W:**[out]**
星期,0=星期天,1=星期一…6=星期六
TZ:**[out]**
时区,定义如下

+0x00	带时区的 HH：MIN	(-12)
+0x01	带时区的 HH：MIN	(-11)
+0x02	带时区的 HH：MIN	(-10)
+0x03	带时区的 HH：MIN	(-9)
+0x04	带时区的 HH：MIN	(-8)
+0x05	带时区的 HH：MIN	(-7)
+0x06	带时区的 HH：MIN	(-6)
+0x07	带时区的 HH：MIN	(-5)
+0x08	带时区的 HH：MIN	(-4)
+0x09	带时区的 HH：MIN	(-3)
+0x0a	带时区的 HH：MIN	(-2)
+0x0b	带时区的 HH：MIN	(-1)
+0x0c	带时区的 HH：MIN	(+0)
+0x0d	带时区的 HH：MIN	(+1)
+0x0e	带时区的 HH：MIN	(+2)
+0x0f	带时区的 HH：MIN	(+3)
+0x10	带时区的 HH：MIN	(+4)
+0x11	带时区的 HH：MIN	(+5)
+0x12	带时区的 HH：MIN	(+6)
+0x13	带时区的 HH：MIN	(+7)
+0x14	带时区的 HH：MIN	(+8)
+0x15	带时区的 HH：MIN	(+9)
+0x16	带时区的 HH：MIN	(+10)
+0x17	带时区的 HH：MIN	(+11)
+0x18	带时区的 HH：MIN	(+12)
+0x19	带时区的 HH：MIN	(+13)
+0x1a	带时区的 HH：MIN	(-3:30)
+0x1b	带时区的 HH：MIN	(+5:30)
+0x1c	带时区的 HH：MIN	(+5:45)
+0x1d	带时区的 HH：MIN	(+6:30)
+0x1e	带时区的 HH：MIN	(+9:30)
+0x1f	带时区的 HH：MIN	(+3:30)
+0x20	带时区的 HH：MIN	(+4:30)

+0x21　带时区的 HH：MIN　（-4:30）

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
 INT16U Y;
INT8U M,D,H,MM,S,W,TZ;
if(czReadLEDTime(&Y,&M,&D,&H,&MM,&S,&W,&TZ)==0)
{
}
```

相关函数

5.2 时间校正 (czAjustLEDTimeEx)

函数原型

INT32U czAjustLEDTimeEx(INT16U Y,INT8U M, INT8U D,INT8U H,INT8U MM, INT8U S,INT8U W,INT8U TZ);

描述

校正显示屏时间。

参数

参数意义与 czReadLEDTime 一样，请参数 czReadLEDTime 函数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
 INT16U Y=2013;
INT8U M=4,D=11,H=1,MM=1,S=0,W=4,TZ=1;
if(czAjustLEDTimeEx (Y,M,D,H,MM,S,W,TZ)==0)
{
}
```

相关函数

5.3 温/湿度通知 (czSendTempHumi)

函数原型

INT32U czSendTempHumi(INT8U Humidity, INT8S Temperature)

描述

命令的作用是用来发送温/湿度到 LED 主板上，即由温湿传感头主动发数据到主板中。

参数

Humidity: **[in]**
湿度,值为[0-100]。
Humidity: **[in]**
温度，值为[-128~+127]

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Humidity;
INT8S Temperature;
Humidity = 50;
```

```
Temperature = 30;
if(czSendTempHumi (Humidity, Temperature)==0)
{
}
```

相关函数

5.4 限速值写入 (czWriteSpeedLimit)

函数原型

```
INT32U czWriteSpeedLimit(INT16U limitSpeed,INT16U limitOffset)
```

描述

写入 RADAR 限速值及偏移值。部分主板支持。

参数

limitSpeed: **[in]**
限速值.
limitOffset: **[in]**
偏移值.

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT16U limitSpeed,INT16U limitOffset;
limitSpeed = 100;
limitOffset = 0;
if(czWriteSpeedLimit(limitSpeed,limitOffset)==0)
{
}
```

相关函数

6. 播放控制

6.1 重新播放文件列表 (czReplayList)

函数原型

```
INT32U czReplayList()
```

描述

重新播放文件列表。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czReplayList ()==0)
{
}
```

相关函数

6.2 重新播放当前文件 (czReplayCurrFile)

函数原型

```
INT32U czReplayCurrFile ()
```

描述

重新播放当前文件。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czReplayCurrFile ()==0)
{
}
```

相关函数

6.3 暂停播放 (czPlayPause)

函数原型

```
INT32U czPlayPause ()
```

描述

暂停播放。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czPlayPause ()==0)
{
}
```

相关函数

6.4 继续播放 (czPlayContinue)

函数原型

```
INT32U czPlayContinue()
```

描述

继续播放。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czPlayContinue()==0)
{
}
```

相关函数

6.5 播放下一个文件 (czPlayNext)

函数原型

```
INT32U czPlayNext()
```

描述

播放列表里下一个文件。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czPlayNext()==0)
{
}
```

相关函数

6.6 优先播放一个文件 (czPlayPriority)

函数原型

```
INT32U czPlayPriority (INT8U Drive, INT8U Type,INT8U* FileName);
```

描述

优先播放某个文件，播放完后，再重新开始列表的播放。

参数

Drive: **[in]**

分区号，"D" "E" "F"

Type: **[in]**

文件类型,'T'=Text File, P=Picture File, A=Array Picture file,F=movie

FileName: **[in]**

文件名

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Letter, INT8U Type
INT8U *FileName;
Letter = 'D';
Type   =  'T';
FileName = "Temp.NMG";
if(czPlayPriority(Letter,Type,FileName)==0)
{
}
```

相关函数

6.7 读取当前播放文件名 (czGetPlayingFileName)

函数原型

```
INT32U czGetPlayingFileName(INT8U* FileName,INT32U NameLen);
```

描述

读取当前播放文件名字。可以用读文件命令读取文件内容。

参数

FileName: **[out]**

读回的文件名

NameLen: **[in]**

FileName 最大长度。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U FileName[256];
if(czGetPlayingFileName (FileName,256)==0)
{
}
```

相关函数

6.8 读取下一个播放文件名(czGetNextPlayFileName)

函数原型

```
INT32U czGetNextPlayFileName(INT8U* FileName, INT32U NameLen);
```

描述

读取下一个播放文件名字。部分老的主板支持。

参数

FileName: **[out]**

读回的文件名

NameLen: **[in]**

FileName 最大长度

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U FileName[256]
if(czGetNextPlayFileName(FileName,256)==0)
{
}
```

相关函数

6.9 播放上一个文件 (czPlayPrevious)

函数原型

```
INT32U czPlayPrevious()
```

描述

播放上一个文件。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czPlayPrevious()==0)
{
}
```

相关函数

6.10 向前跳 (czPlayForword)

函数原型

```
INT32U czPlayForword()
```

描述

控制播放向前跳一格。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czPlayForword()==0)
{
}
```

相关函数

6.11 倒退 (czPlayBack)

函数原型

```
INT32U czPlayBack()
```

描述

控制播放往回跳一格。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czPlayBack()==0)
{
}
```

相关函数

6.12 下一帧 (czPlayNextFrame)

函数原型

```
INT32U czPlayNextFrame()
```

描述

控制播放下一帧。部分老的主板支持。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czPlayNextFrame()==0)
{
}
```

相关函数

6.13 声音控制 (czSoundCtrl)

函数原型

```
INT32U czSoundCtrl(INT8U SoundSwitch,INT8U mode,INT8U time);
```

描述

控制蜂鸣器。有蜂鸣器的主板才支持该命令

参数

switch: **[in]**

蜂鸣器开关 1 = 打开蜂鸣器, 0 = 关闭蜂鸣器。

mode: **[in]**

蜂鸣器模式,设置打开时才起作用 0 = 接收到新的文件时响 1 = 播放文件切换时响

time: **[in]**

蜂鸣器叫声长短 值为:'0' - '9' (Second) '0' = 不响 '1' = 响一秒。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U switch,INT8U mode,INT8U time;
switch = 1;
mode   = 0;
time   = 1;
if(czSoundCtrl(switch,mode,time)==0)
{
}
```

相关函数

6.14 开始倒/正计时 (czBeginTiming)

函数原型

```
INT32U czBeginTiming(INT8U day, INT8U hour,INT8U min,INT8U sec)
```

描述

开始倒/正计时,这个为可调节倒计时/正计时。部分老的主板支持。

参数

day: **[in]**

天

hour: **[in]**

小时

min: **[in]**

分钟

sec: **[in]**

秒 s

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U day=0,hour=0,min=5,sec=0;
if(czBeginTiming(day,hour,min,sec)==0)
{
}
```

相关函数

6.15 停止倒/正计时 (czStopTiming)

函数原型

```
INT32U czStopTiming()
```

描述

停止计时。部分老的主板支持。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czStopTiming()==0)
{
}
```

相关函数

6.16 暂停倒/正计时 (czPauseTiming)

函数原型

```
INT32U czPauseTiming()
```

描述

暂停计时。部分老的主板支持。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czPauseTiming()==0)
{
}
```

相关函数

6.17 继续倒/正计时 (czContinueTiming)

函数原型

```
INT32U czContinueTiming()
```

描述

继续计时。部分老的主板支持。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czContinueTiming()==0)
{
}
```

相关函数

6.18 获取当前的播放画面 (czReadCurScreenshot)

函数原型

```
INT32U czReadCurScreenshot(INT8U *pcpath)
```

描述

获取屏当前的实时播放画面的截图。

参数

pcPath: **[in]**

保存截图到本地的路径

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U pcpath[256];
if(czReadCurScreenshot(pcpath)==0)
{
}
```

相关函数

6.19 播放音频文件 (czPlaySoundFile)

函数原型

```
INT32U czPlaySoundFile(SoundST soundST,INT8U* soundPath)
```

描述

播放指定的音频文件。

参数

soundST: **[in]**

播放音频文件的参数，结构体如下定义：

```
typedef struct
{
  INT8U BuzzerSwitch;
  INT8U PlayTimes;
  INT8U Rev[6];
}SoundST;
```

说明：BuzzerSwitch 蜂鸣器开关(0:停止播放, 1:播放音频文件)，playTimes 播放次数(1:播放一次，2:播放两次，0xFF:循环播放)

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
SoundST soundST;
Memset(&soundST,0,sizeof(soundST));
soundST.BuzzerSwitch = 1;
soundST.PlayTimes = 1;
if(czPlaySoundFile(soundST,(INT8U*)"c://Test.mp3")==0)
{
}
```

相关函数

czAdjustVolume

6.20 调节音量 (czAdjustVolume)

函数原型

```
INT32U czAdjustVolume(INT8U soundVolume)
```

描述

调节播放音频文件的系统音量。

参数

soundVolume: **[in]**

音量值

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czAdjustVolume(80)==0)
{
}
```

相关函数

czPlaySoundFile

7．文件控制

7.1 格式化分区 (czFormatDrive)

函数原型

```
INT32U czFormatDrive(INT8U Drive);
```

描述

格式化指定分区。”C”, ”D”, ”E”, ”F”。

参数

Drive: **[in]**

指定分区.

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Letter = “D”;
if(czFormatLetter(Letter)==0)
{
}
```

相关函数

7.2 创建文件夹 (czCreateDir)

函数原型

```
INT32U czCreateDir(INT8U* Dir)
```

描述

创建文件夹。

参数

Dir: **[in]**

要创建的文件夹路径．如： 在 C 盘下创建 TEST 文件夹则为：“C:\TEST\”

注：不能一次创建多级文件夹,以 “\”+ NULL 结尾

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U DirPath = “C:\TEST\”;
if(czCreateDir(DirPath)==0)
{
}
```

相关函数

7.3 重命名 (czRename)

函数原型

```
INT32U czRename(INT8U* SourceName, INT8U* DestName)
```

描述

重命名指定文件。

参数

SourceName: **[in]**

源文件名。

DestName: **[in]**

目的文件名。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U *SourceName, *DestName;
SourceName = "D:\\T\\Temp.NMG";
DestName = "D:\\T\\1.NMG";
if(czRename(SourceName,DestName)==0)
{
}
```

相关函数

7.4 移动文件 (czMove)

函数原型

```
INT32U czMove(INT8U* SourceName, INT8U* DestName)
```

描述

移动文件。只支持在同一盘中移动.

参数

SourceName: **[in]**

源文件名。

DestName: **[in]**

目的文件名。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U SourceName[256], INT8U[256];
SourceName = "D:\\T\\Temp.NMG";
DestName = "F:\\T\\1.NMG";
if(czMove(SourceName,DestName)==0)
{
}
```

相关函数

7.5 删除文件 (czDelete)

函数原型

```
INT32U czDelete(INT8U* FileName)
```

描述

删除指定文件。

参数

FileName: **[in]**

要删除的文件路径以及文件名。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U *FileName;
FileName = "D:\\T\\Temp.NMG";
if(czDelete(FileName)==0)
{
}
```

相关函数

7.6 删除一个指定分区 Text 文件 (czDelTextFiles)

函数原型

```
INT32U czDelTextFiles(INT8U Drive);
```

描述

删除一个指定分区内的 Text 文件。

参数

Drive: **[in]**

指定分区。"D", "E", "F"

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Letter;
Letter = "D";
if(czDelTextFiles(Letter)==0)
{
}
```

相关函数

7.7 删除一个指定分区 String 文件 (czDelStringFInLetter)

函数原型

```
INT32U czDelStringFile(INT8U Drive);
```

描述

删除一个指定分区内的 String 文件。

参数

Drive: **[in]**

指定分区。"D", "E", "F"

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Letter;
Letter = "D";
if(czDelStringFile(Letter)==0)
{
}
```

相关函数

7.8 删除一个指定分区 Picture 文件 (czDelPictureFiles)

函数原型

```
INT32U czDelPictureFiles(INT8U Drive);
```

描述

删除一个指定分区内的 Picture 文件。

参数

Drive: **[in]**

指定分区。"D", "E", "F"

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Letter;
Letter = "D";
if(czDelPictureFiles (Letter)==0)
{
}
```

相关函数

7.9 删除一个指定分区 ArrayPicture 文件 (czDelArrpicFiles)

函数原型

```
INT32U czDelArrpicFiles(INT8U Drive);
```

描述

删除一个指定分区内的 ArrayPicture 文件。

参数

Drive: **[in]**

指定分区。"D", "E", "F"

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Letter;
Letter = "D";
if(czDelArrpicFiles (Letter)==0)
{
}
```

相关函数

7.10 获取文件夹文件项信息 (czGetDirFile)

函数原型

```
INT32U czGetDirFile(INT8U* path,INT8U* Num,DIRECTORY_ENTRY_STRUCT* dirEntry,INT32U size);
```

描述

获取文件夹里所有文件目录项。

参数

path: **[in]**

该参数指出要获取的文件夹路径。如："C:\TEST\"为显示 C 盘下 TEST 文件夹下面的文件

Num: **[out]**

返回该目录下文件目录项个数。

dirEntry: **[out]**

返回的文件目录项列表。

size: **[in]**

读回的最大的目录数。

typedef struct

```
    {
      UBYTE badir_name[11];       //文件名
      UBYTE bdir_attr;            //文件属性
      UBYTE bdir_rev;             //保留
      UBYTE bcrt_time_tecth;      //创建时间 1，秒的计数
      UWORD wcrt_time;            //创建时间 2，时分的计数
      UWORD wcrt_date;            //创建日期
      UWORD wlast_acc_time;       //最后访问时间，时分
      UWORD wfst_clus_hi;         //
      UWORD wwrt_time;            //最后修改时间
      UWORD wwrt_date;            //最后修改日期
      UWORD wfst_clus_lo;
      ULONG dwfile_size;          //文件大小
  }DIRECTORY_ENTRY_STRUCT;
```

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT32U Num;
DIRECTORY_ENTRY_STRUCT entry[10];
if(czGetDirFile("D:\\T\\",&Num, entry,100)==0)
{
}
```

相关函数

7.11 获取文件夹里所有文件(扩展) (czGetDirFileEX)

函数原型

```
INT32U czGetDirFileEx(INT8U* path,INT16U StarNo,INT16U RNum,INT32U* Num,
                      DIRECTORY_ENTRY_STRUCT* dirEntry, INT32U size);
```

描述

获取文件夹里指定的文件目录项,主要用于读取大的文件夹里的东西,因为 czGetDirFile 一次读取的大小有限。

参数

path: **[in]**

该参数指出要获取的文件夹路径。如："C:\TEST\"为显示 C 盘下 TEST 文件夹下面的文件

StartNo: **[in]**

开始目录项

RNum: **[in]**

读取目录项个数

Num: **[out]**

返回读回的文件目录项个数。

dirEntry: **[out]**

返回的文件目录项列表。

size:**[in]**

读回的最大目录数。

```
typedef struct
{
  UBYTE badir_name[11];       //文件名
  UBYTE bdir_attr;            //文件属性
  UBYTE bdir_rev;             //保留
  UBYTE bcrt_time_tecth;      //创建时间 1，秒的计数
```

```
    UWORD wcrt_time;            //创建时间 2，时分的计数
    UWORD wcrt_date;            //创建日期
    UWORD wlast_acc_time;       //最后访问时间，时分
    UWORD wfst_clus_hi;         //
    UWORD wwrt_time;            //最后修改时间
    UWORD wwrt_date;            //最后修改日期
    UWORD wfst_clus_lo;
    ULONG dwfile_size;          //文件大小
}DIRECTORY_ENTRY_STRUCT;
```

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT32U Num;
DIRECTORY_ENTRY_STRUCT entry[100];
if(czGetDirFileEx((INT8U*)"D:\\T\\",0,10,&Num, entry,100)==0)
{
}
```

相关函数

7.12 获取磁盘信息 (czGetDriveInfo)

函数原型

```
INT32U czGetDriveInfo(INT8U Drive, INT32U *totalSize, INT32U *remainSize, INT8U *driveName);
```

描述

获得指定盘的信息。

参数

Drive: **[in]**

指定分区。"D", "E", "F"

totalSize: **[out]**

返回的磁盘总大小信息,单位为 Bytes。

remainSize: **[out]**

返回磁盘剩余大小，单位为 Bytes。

driveName: **[out]**

返回磁盘名称。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT32U totalSize,remainSize;
INT8U  driveName[64];
memset(driveName,0,sizeof(driveName));
if(czGetDriveInfo('D',& totalSize, &remainSize, driveName )==0)
{
}
```

相关函数

7.13 查询指定文件是否存在 (czIsFileExist)

函数原型

```
INT32U czIsFileExist(INT8U* FileName)
```

描述

查询指定文件是否存在。

参数

FileName: **[in]**

指定文件路径以及名称.

返回值

如果函数调用成功并文件存在，返回 **0**，文件不存在，返回 0x7E01 错误码，否则返回**错误代码**。

程序范例

```
//代码清单
if(czIsFileExist((INT8U *)"D:\\T\\Temp.NMG")==0)
{
}
```

相关函数

7.14 清除屏体所有播放文件 (czClearAllPlayFile)

函数原型

```
INT32U czClearAllPlayFile(INT8U Operation,INT8U* Process)
```

描述

查询指定文件是否存在。

参数

Operation: **[in]**

操作. 0,清除,1 查询进度.

Process: **[out]**

清除进度.

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Operation ,Process;
Operation = 0;
if(czClearAllPlayFile(Operation,&Process)==0)
{
}
```

相关函数

7.15 获取文件夹里所有文件项(扩展－支持长文件名) (czGetDirLongFileEx)

函数原型

```
INT32U czGetDirLongFileEx(INT8U* path, INT8U* pcpath)
```

描述

获取文件夹里指定的文件目录项,这个函数支持长文件名.

参数

path: **[in]**

该参数指出要获取的文件夹路径。如："C:\TEST\"为显示 C 盘下 TEST 文件夹下面的文件

pcpath: **[in]**

电脑上要保存的文件文件全路径+文件名

读取成功后，文件 格式如下：

```
DIRECTORY_ENTRY_STRUCTExHead+ DIRECTORY_ENTRY_STRUCTEx*N
```

- 文件结构：

```
Typedef struct
```

```
{
    UWORD Flag;                 //"DL"文件标志  0x444C
    UWORD HeadLen;              //可变文件头大小
    ULONG Count;                //结构体个数
    ULONG  FileSize;            //文件大小
    ULONG  Recv;                //保留 4 字节
} DIRECTORY_ENTRY_STRUCTExHead;

 typedef struct  //DirectoryEntryStructure
 {
     UBYTE badir_name[255];  //文件名
     UBYTE bdir_attr;        //文件属性
     UBYTE bdir_rev;         //保留
     UBYTE bcrt_time_tecth;  //创建时间 1，秒的计数
     UWORD wcrt_time;     //创建时间 2，时分的计数
     UWORD wcrt_date;     //创建日期
     UWORD wlast_acc_time;   //最后访问时间，时分
     UWORD wfst_clus_hi;     //
     UWORD wwrt_time;     //最后修改时间
     UWORD wwrt_date;     //最后修改日期
     UWORD wfst_clus_lo;
     ULONG dwfile_size;      //文件大小
 }DIRECTORY_ENTRY_STRUCTEx;
```

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czGetDirLongFileEx ((INT8U*)"D:\\T\\", (INT8U*)"C:\\DList.dlt")==0)
{

}
```

相关函数

7.16 获取文件夹里所有文件项(扩展一支持长文件名)-回调 (czLstLongFolderCB)

函数原型

```
INT32U czLstLongFolderCB( INT8U* signpath,
                          INT32S (*cb)(DIRECTORY_ENTRY_STRUCTEx *dirEntry, void *cbObj), void *cbObj);
```

描述

获取文件夹里指定的文件目录项,这个函数支持长文件名.

参数

signpath: **[in]**

该参数指出要获取的文件夹路径。如："C:\TEST\"为显示 C 盘下 TEST 文件夹下面的文件

cb: **[in/out]**

回调函数

cbObj: **[in/out]**

回调函数用到的参数，用户自定义

读取成功后，文件 格式如下：

```
typedef struct  //DirectoryEntryStructure
{
    UBYTE badir_name[255];  //文件名
    UBYTE bdir_attr;        //文件属性
    UBYTE bdir_rev;         //保留
    UBYTE bcrt_time_tecth;  //创建时间 1，秒的计数
    UWORD wcrt_time;     //创建时间 2，时分的计数
    UWORD wcrt_date;     //创建日期
    UWORD wlast_acc_time;   //最后访问时间，时分
    UWORD wfst_clus_hi;     //
    UWORD wwrt_time;     //最后修改时间
    UWORD wwrt_date;     //最后修改日期
    UWORD wfst_clus_lo;
    ULONG dwfile_size;      //文件大小
}DIRECTORY_ENTRY_STRUCTEx;
```

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT32S cb(DIRECTORY_ENTRY_STRUCTEx *dirEntry, void *cbObj)
{

}

if(czLstLongFolderCB ((INT8U*)"D:\\T\\", cb, NULL)==0)
{

}
```

相关函数

8. OnLineTicker

8.1 开启 OnLine Ticker (czBeginUnlimited)

函数原型

```
INT32U czTickerStart(INT8U dir,INT8U speed, INT8U dataFormat);
```

描述

开启连接播放模式，这种模式下，所有显示内容都是以图片的形式下发到显示屏。部分老的主板支持。

参数

dir: **[in]**

方向 0x00 表示左移动,0x01 表示右移动.

speed: **[in]**

速度 0x00 最快 … 0x06 最慢 （7 个等级.

DataFormat: **[in]**

表示后面发送的数据格式. 具体参照 JetFileII 协议。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U direction,speed, DataFormat;
direction = 0x00;
speed = 0x00;
DataFormat = 1;
if(czTickerStart (direction,speed, DataFormat)==0)
{
}
```

相关函数

8.2 终止连接播放 (czTickerStop)

函数原型

```
INT32U czTickerStop()
```

描述

终止连接播放。部分老的主板支持。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czTickerStop()==0)
{
}
```

相关函数

8.3 查询接受状态 (czGetBufferStatus)

函数原型

```
INT32U czGetBufferStatus(INT16U* statusCode);
```

描述

查询连接播放的接收缓冲区状态。部分老的主板支持。

参数

StatusCode: **[out]**

返回状态码。定义如下:

状态码对照表

BufPost	缓冲区 1	缓冲区 2	空闲缓冲区	状态码
0	OLD_BUF	CUR_BUF	DISPLAY_BUF	0x8301
1	CUR_BUF	DISPLAY_BUF	OLD_BUF	0x8302
2	DISPLAY_BUF	OLD_BUF	CUR_BUF	0x8303

状态码抽象意义

①,②, ③分别表示缓冲区：1_BUF, 2_BUF, 3_BUF

0x8301：①,②缓冲区在显示，③缓冲区没有在显示。其中①在移出，②在移入;

0x8302：②,③缓冲区在显示，①缓冲区没有在显示。其中②在移出，③在移入

0x8303：③,①缓冲区在显示，②缓冲区没有在显示。其中③在移出，①在移入

如果系统不在无限连接播放状态，就返回 0x8305

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT16U StatusCode;
if(czCheckBufStatus(&StatusCode)==0)
{
}
```

相关函数

8.4 数据下载指令 (czUploadBuffer)

函数原型

```
INT32U czUploadBuffer(INT8U* Data, INT32U dataLen)
```

描述

把显示数据下载到屏体的缓冲区。部分老的主板支持。

参数

Data: **[in]**

一帧屏大小数据。

dataLen**[in]**

数据大小

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Data[12*1024];
//Set Data..
if(czUpdateBuf(Data, 12*1024)==0)
{
}
```

相关函数

9. 接龙播放(OFF Line Ticker)

9.1 开启接龙播放 (czOffLineTickerStart)

函数原型

```
INT32U czOffLineTickerStart (INT8U mode, INT8U dir,INT8U speed, INT8U sec);
```

描述

开启连接播放。显示会播放播放列表中的所有文件！

参数

mode: **[in]**

模式,0x00 = 不断移动,0x01=移动一屏暂停一下

dir: **[in]**

方向 0x00 表示左移动,0x01 表示右移动.

speed: **[in]**

速度 0x00 最快 … 0x06 最慢 （7 个等级.

sec: **[in]**

模式为 1 时，表示停留的时间,单位为秒。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U direction,speed, sec;
direction = 0x00;
speed = 0x00;
```

```
sec = 1;
if(czOffLineTickerStart (1,direction,speed, sec)==0)
{
}
```

相关函数

9.2 终止接龙播放 (czOffLineTickerStop)

函数原型

```
INT32U czOffLineTickerStop()
```

描述

终止连接播放。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czOffLineTickerStop ()==0)
{
}
```

相关函数

10. 登录

10.1 登录 (czLogin)

函数原型

```
INT32U czLogin (INT8U* UserName, INT8U* Password)
```

描述

登录到主控系统中。

参数

UserName: **[in]**

用户名。

Password: **[in]**

密码。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U UserName[16],Password[8];
memset(UserName,0,16);
memset(Password,0,8);
memcpy(UserName,"admin",strlen("admin"));
memcpy(Password,"1234",strlen("1234"));
if(czLogin (UserName,Passsword)==0)
{
}
```

相关函数

10.2 注销 (czLogout)

函数原型

INT32U czLogout()

描述

注销。

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czLogout()==0)
{
}
```

相关函数

10.3 更改密码 (czChangePSW)

函数原型

INT32U czChangePSW(INT8U* UserName, INT8U* Password, INT8U* NewPassword)

描述

登录到系统中。

参数

UserName: **[in]**

用户名。

Password: **[in]**

密码。

NewPassword: **[in]**

新密码。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U UserName[15],Password[7];
UserName = "admin";
Password = "1234";
NewPassword = "abcd";
if(czChangePSW(UserName,Passsword,NewPassword)==0)
{
}
```

相关函数

11. VPU3400 操作

11.1 选择视频输入通道 (czVPUSelChannel)

函数原型

INT32U czVPUSelChannel(INT8U ch)

描述

用于切换视频源。

参数

ch: **[in]**

通道选择。

0: YPbPr

1: S_Video

2: CVBS1
3: CVBS2
4: CVBS3
5: VGA
6: SDI
7: HDMI
8: Test

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U channel = 1;
if(czVPUSelChannel (channel)==0)
{
}
```

相关函数

11.2 设置显示模式 (czVPUSetMode)

函数原型

```
INT32U czVPUSetMode(INT8U mode, INT8U alpha)
```

描述

设置 DVI 窗口和视频窗口的叠加模式。

参数

Mode: **[in]**
0: 仅 DVI
1: 仅 Video
2: DVI 在上
3: Video 在上
4: 混合
5: 字幕机

alpha: **[in]**
设置混合模式下的透明度 [0-100]

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Mode,Transparency;
Mode = 4; Transparency = 50;
if(czVPUSetMode (Mode,Transparency)==0)
{
}
```

相关函数

11.3 设置视频窗口显示比例 (czVPUSetVideoRatio)

函数原型

```
INT32U czVPUSetVideoRatio(INT8U ratio);
```

描述

设置视频窗口的显示比例。

参数

Type: **[in]**

0:填充窗口
1: 16:10
2: 16:9
3: 5:4
4: 4:3
5: 3:2

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Type;
Type = 2;
if(czVPUSetVideoRatio(Type)==0)
{
}
```

相关函数

11.4 设置 DVI 窗口 (czVPUSetDVIWin)

函数原型

```
INT32U czVPUSetDVIWin(INT16U Type, INT16U value)
```

描述

设置 DVI 窗口的位置和大小。要设置一个完全，需要 4 次调用此函数.

参数

Type: **[in]**
0: 设置窗口的 X 坐标。
1: 设置窗口的 Y 坐标。
2: 设置窗口的宽度。
3: 设置窗口的高度。
value: **[in]**
设置的值

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT16U Type,value;
Type = 2; value = 0;
if(czVPUSetDVIWin(Type,value)==0)
{
}
```

相关函数

11.5 设置视频窗口 (czVPUSetVideoWin)

函数原型

```
INT32U czVPUSetVideoWin(INT16U Type, INT16U value)
```

描述

设置视频窗口的位置和大小。要设置一个完全，需要 4 次调用此函数.

参数

Type: **[in]**
0: 设置窗口的 X 坐标。
1: 设置窗口的 Y 坐标。

2：设置窗口的宽度。
3：设置窗口的高度。

value：**[in]**
设置的值

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT16U Type,value;
Type = 2; value = 0;
if(czVPUSetVideoWin(Type,value)==0)
{
}
```

相关函数

11.6 设置视频参数 (czVPUSetVideoArg)

函数原型

```
INT32U czVPUSetVideoArg(INT16U type, INT16U value);
```

描述

设置视频的亮度，对比度，饱和度等参数

参数

Type：**[in]**
0：设置亮度。有效值-50--50
1：设置对比度。有效值 10--100
2：设置色调。有效值-90--90
3：设置饱和度。有效值 0--100
4：设置锐利度。有效值 0--15
5：设置左右偏移量。有效值 0--512
6：设置上下偏移量。有效值 0--512
7：恢复为默认值。

value：**[in]**
设置的值

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT16U Type,value;
Type = 2; value = 0;
if(czVPUSetVideoArg (Type,value)==0)
{
}
```

相关函数

11.7 获取输入信号的状态 (czVPUGetSignalStatus)

函数原型

```
INT32U czVPUGetSignalStatus(INT16U *videoW, INT16U *vedioH, INT16U *dviW, INT16U *dviH);
```

描述

获取 DVI 输入信号和视频输入信号的状态.

参数

videoW：**[out]**

读回 Video 信号宽度。

vedioH: **[out]**

读回 Video 信号高度。

dviW: **[out]**

读回 DVI 信号宽度。

dviH: **[out]**

读回 DVI 信号高度。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT16U VideoWidth, VideoHeight, DVIWidth,DVIHeight;
if(czVPUGetSignalStatus (&VideoWidth, &VideoHeight, &DVIWidth,&DVIHeight)==0)
{
}
```

相关函数

11.8 设置系统类型 (czVPUType)

函数原型

```
INT32U czVPUType(INT8U type)
```

描述

设置系统类型，选择作为主机或从机.

参数

Type: **[in]**

0：设为主机。

1：设为从机。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Type = 1;
if(czVPUType(Type)==0)
{
}
```

相关函数

11.9 设置从机的起始行位置 (czVPUSlaverStartLine)

函数原型

```
INT32U czVPUSlaverStartLine(INT16U StartLine);
```

描述

设置从机起始行位置.

参数

StartLine: **[in]**

从机起始行位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT16U StartLine = 512;
if(czVPUSlaverStartLine(StartLine)==0)
{
}
```

相关函数

11.10 设置色温值 (czVPUSetColorTemp)

函数原型

```
INT32U czVPUSetColorTemp(INT8U type,INT8U R, INT8U G, INT8U B);
```

描述

设置色温值.

参数

Type: **[in]**
色温类型 0：6500 1：9300 2：自定义。

R: **[in]**
自定义色温时，红色分量的值

G: **[in]**
自定义色温时，绿色分量的值

B: **[in]**
自定义色温时，蓝色分量的值

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Type=0, Red=255, Green=220, Blue=220;
if(czVPUSetColorTemp(Type, Red, Green, Blue)==0)
{
}
```

相关函数

11.11 设置屏体亮度 (czVPUSetBright)

函数原型

```
INT32U czVPUSetBright(INT8U type,INT8U manualValue, INT8U autoMin, INT8U autoMax);
```

描述

设置显示屏的亮度.

参数

Type: **[in]**
0：手动亮度调节
1：自动亮度调节

manualValue: **[in]**
手动设置的亮度值

autoMin: **[in]**
自动亮度调节的最小值

autoMax: **[in]**
自动亮度调节的最大值

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Type, Bright, Max=100, Min=10;
Type = 0;
Bright = 50;
if(czVPUSetBright(Type, Bright, Max, Min)==0)
{
}
```

相关函数

11.12 获取亮度状态 (czVPUGetBright)

函数原型

```
INT32U czVPUGetBright(INT8U *type,INT8U *manualValue, INT8U *autoMin, INT8U *autoMax);
```

描述

获取亮度信息.

参数

Type: **[out]**

0：手动亮度调节 1：自动亮度调节

Bright: **[out]**

手动设置的亮度值

Max: **[out]**

自动亮度调节的最小值

Min: **[out]**

自动亮度调节的最大值

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Type, Bright, Max, Min;
if(czVPUGetBright(&Type,&Bright,&Max,&Min)==0)
{
}
```

相关函数

11.13 设置屏体 Gamma 值 (czVPUSetGamma)

函数原型

```
INT32U czVPUSetGamma(INT8U index);
```

描述

设置显示屏的 Gamma 校正值.

参数

GammaValue: **[in]**

Gamma 值。有效值为(0--3)

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U GammaValue = 0;
if(czSetGamma (GammaValue)==0)
{
}
```

相关函数

11.14 设置 LDU 数量 (czVPUSetLDUNums)

函数原型

```
INT32U czVPUSetLDUNums(INT8U LDUNums);
```

描述

设置显示屏的 LDU 数量.

参数

LDUNum: **[in]**

LDU 数量。有效值为(1--8)

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U LDUNum = 2;
if(czVPUSetLDUNums(LDUNum)==0)
{
}
```

相关函数

11.15 设置 LDU 坐标 (czVPUSetLDUPos)

函数原型

```
INT32U czVPUSetLDUPos(INT8U LDUID, INT16U x, INT16U y);
```

描述

设置显示屏的 LDU 坐标.

参数

LDUNo: **[in]**

要设置的 LDU 的序号。有效值为(1—8)

x: **[in]**

x 坐标

y: **[in]**

y 坐标

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U LDUNo = 2;
INT16U x = 512, y = 0;
if(czVPUSetLDUPos(LDUNo,x,y)==0)
{
}
```

相关函数

11.16 获取 VPU 版本信息 (czVPUGetInfo)

函数原型

```
INT32U czVPUGetInfo(VPUVerInfo *info);
```

描述

获取 VPU 的版本信息.

参数

info: **[out]**

取得的 LDU 版本信息结构体,结构定义如下

typedef struct

```
  {
    INT16U CPUVer;
    INT16U FPGA1Ver;
    INT16U FPGA2Ver;
    INT16U FPGA3Ver;
    INT8U  SN[12];
  }LDU_VERINFO;
```

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
LDU_VERINFO lduInfo
if(czVPUGetInfo(&lduInfo)==0)
{
}
```

相关函数

11.17 设置像素模式 (czVPUSetPixcelMode)

函数原型

```
INT32U czVPUSetPixcelMode(INT8U mode);
```

描述

设置工作模式.

参数

mode: **[in]**

0:实像素

1:虚拟像素

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U PixModel = 0;
if(czVPUSetPixcelMode(PixModel)==0)
{
}
```

相关函数

12 显示屏箱体操作

该命令主要用于读取多箱体的显示屏,每个箱体的都有控制板,这里的命令用于读取箱体里的状态

12.1 请求读控制板的状态

函数原型

```
INT32U czImposaGetTileStat(INT8U tileAddr, INT32U *rtCode);
```

描述

请求读取指定地址的箱体状态.

参数

tileAddr: **[in]**

箱体地址,有效值为 0 - 494

rtCode: **[Out**

返回代码,需要用这个代码查询读取的状态

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT32U retCode = 0;
if(czImposaGetTileStat(1, & retCode)==0)
{
}
```

相关函数

czImposaGetResult

12.2 对控制板的操作信息读回

函数原型

INT32U czImposaGetResult(INT32U rtCode, INT8U *result, INT32U size, INT16U *resultCode);

描述

读取上次请求的操作情况,rtCode 为 czImposaGetTileStat 返回的结果,结果保存在 result 中.

参数

rtCode: **[in]**

czImposaGetTileStat 函数的获得的值.

result : **[Out]**

保存结果,如果读取成功,结果里的意义如下:

地址偏移	数据名称	数据大小(字节)	说明
0x00000	箱体长	1	
0x00001	箱体高	1	
0x00002	箱体地址	1	地址范围 1---0xef ，最多可连接 239 箱体
0x00003	CPU 版本	2	（BCD 码）0x1000 表示 Ver1.0
0x00005	FPGA 版本	2	（BCD 码）0x1000 表示 Ver1.0
0x00007	当前箱体亮度	1	
0x00008	当前显示屏最大亮度	1	
0x00009	箱体温度	1	0x00 --- 0x7f 表示正的温度，范围 0 ---- 127 0x80 --- 0x99 表示负的温度，范围 -25 ---- 0 0xff 表示温度传感器故障
0x0000a	风扇开启温度	1	0---127
0x0000b	风扇开启状态	1	0x00 表示开；0x80 表示关
0x0000c	风扇的电流 0	2	AD 把原始值返回， 0 --- 1023
0x0000e	风扇的电流 1	2	AD 把原始值返回， 0 --- 1023
0x00010	电源电压 0	2	AD 把原始值返回， 0 --- 1023
0x00012	电源电压 1	2	AD 把原始值返回， 0 --- 1023
0x00014	箱体状态	2	D0 为 1 时表示 FPGA 加载失败 D1 为 1 时表示该箱体以太网连接错误
0x00016	以太网状态	2	
0x00018	通讯接收次数	2	控制板接收到发给自己的数据包就对该变量加 1
0x0001a	r 数据和校验	2	

0x0001c	驱动板数据和校验	2	
0x0001e	FPGA 数据和校验	2	
0x00020	配置数据和校验	2	
0x00022	X 坐标	2	读 FPGA 取得
0x00024	y 坐标	2	读 FPGA 取得
0x00026	该数据是否有效	1	标识该箱体的字体态是否有效 （由 HUB 控制，控制板不理会）
0x00027	该箱体属于的通道	1	0---15（由 HUB 控制）
0x00028	保留	4	
0x0002c	保留	1	
0x0002b	保留	1	
0x0002E	箱体高温工作时间	4	实际工作时间 ＝ 该值 x 5（分钟）
0x0002D	帧频	1	
0x0002F	像灯检查状态	1	
0x00030	SN 序列号	8	SN 序列号
0x0038	红色索引	1	
0x0039	驱动板类型	1	
0x003a	组地址	1	
0x003b	保留	1	

size: **[in]**
用于指示 result 的大小
resultCode:[Out]
箱体返回的代码

0x0C00	：请求的命令尚在缓冲中
0x0C 01	：任务成功
0x0C 02	：队列号错误
0x0C 03	：正在处理中
0x0C 04	：端点总线故障
0x0C 05	：重试次数大于设定次数
0x0C 06	：一般错误
0x0C 07	：命令 ID 错误
0x0C 08	：广播写数据部分不成功

操作结果等于 0 x01（任务成功），才回送实际数据，其他结果一律不回送数据

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//读取箱体地址为 1 的状态代码清单
INT32U retCode = 0;
if(czImposaGetTileStat(1, & retCode)==0)
{
    INT8U result[1024];
    INT16U resultCode;

    while(czImposaGetResult(retCode, result,1024,&resultCode) ==0)
    {
        if (resultCode == 0x0C01) //成功
        {
            //result[0] =箱体长
            //result[1] =箱体宽
```

```
                //…
                Break;
            }
            else if(resultCode == 0x0C00 || resultCode == 0x0C03)
            {
                //sleep(200) //停留点时间
            }
            else
            {
                Break;
            }
        }

    }
```

相关函数

czImposaGetTileStat

13 停车场屏操作

该命令主要用于停车场场屏的多区域控制

13.1 区域的划分设置

函数原型

```
INT32U czDivideZone(INT8U pageID, INT8U flag, ZoneSetHead zoneSetHead, ZoneSet* zoneSetArrs);
```

描述

设置停车场屏的多个区域.

参数

pageID: **[in]**

帧 ID

flag: **[in]**

0：修改区域或者增加区域

1：重置所有区域

zoneSetHead: **[in]**

区域划分头,结构体如下定义：

```
typedef struct
{
    INT8U BGColor_R;          //帧背景红色
    INT8U BGColor_G;          //帧背景绿色
    INT8U BGColor_B;          //帧背景蓝色
    INT8U ZoneNum;            //区域的数量
    INT16U PageStayTime;      //帧停留时间，单位：10ms,0为无限循环
    INT8U PortraitMode;       //显示模式 0:横向(默认)，1:竖向
    INT8U Rev;                //保留
}ZoneSetHead;
```

说明：BGColor_R, BGColor_G, BGColor_B 表示帧背景色 RGB，ZoneNum 区域的数量，PageStayTime 帧停留时间，单位：10ms，0 为无限循环显示，PortraitMode 显示模式(0：横向(默认)，1:竖向)

zoneSetArrs: **[in]**

要设置区域的数组，结构体如下定义：

```
typedef struct
{
  INT8U ZoneID;             //区域ID
  INT8U Rev[3];             //保留
  INT16U XPos;              //区域位置的横坐标
  INT16U YPos;              //区域位置的纵坐标
  INT16U ZoneWidth;         //区域的宽度
  INT16U ZoneHeight;        //区域的高度
}ZoneSet;
```

说明：ZoneID 区域 ID，XPos 区域的横坐标，YPos 区域的纵坐标，ZoneWidth 区域的宽度，ZoneHeight 区域的高度

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码(参见 13.20)**。

程序范例

```
//代码清单
ZoneSetHead zoneSetHead;
ZoneSet zoneSetEntry[3];
memset(&zoneSetHead, 0, sizeof(ZoneSetHead));
memset(zoneSetEntry, 0, sizeof(ZoneSe)*3);
zoneSetHead.ZoneNum = 1;
zoneSetEntry[0].ZoneID = 1;
zoneSetEntry[0].XPos = 0;
zoneSetEntry[0].YPos = 0;
zoneSetEntry[0].ZoneWidth = 0x40;
zoneSetEntry[0].ZoneHeight = 0x10;

if(czDivideZone(1, 1, zoneSetHead, zoneSetEntry)==0)
{
}
```

相关函数

czGetZone

13.2 获取当前划分的区域

函数原型

INT32U czGetZone(ZoneSetHead *zoneSetHead, ZoneSet *zoneSetArrs, INT8U maxEntryCount, INT8U pageID=1);

描述

获取停车场屏当前划分的多个区域的信息.

参数

zoneSetHead: **[out]**

返回的区域划分头

zoneSetArrs: **[out]**

返回区域划分的数组

maxEntryCount: **[in]**

读回最大区域的个数

pageID: **[in]**

获取页面 ID

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码(参见 13.20)**。

程序范例

```
//代码清单
ZoneSetHead zoneSetHead;
ZoneSet zoneSetEntry[3];
memset(&zoneSetHead, 0, sizeof(ZoneSetHead));
memset(zoneSetEntry, 0, sizeof(ZoneSe)*3);
if(czGetZone(&zoneSetHead, zoneSetEntry,3)==0)
{
}
```

相关函数

czDivideZone

13.3 带属性的显示设置

函数原型

INT32U czSetDisplay(INT8U pageID, INT8U zoneNum, INT8U setMode, ZoneDisplaySet parameter, INT8U* content);

描述

让屏显示带有属性参数的内容.

参数

pageID: **[in]**

帧 ID

zoneNum: **[in]**

区域的数量

setMode: **[in]**

1:保存为模版

parameter: **[in]**

内容的属性，结构体如下：

```
typedef struct
{
  INT16U ZoneSize;          //区域的分辨率大小(即ZoneWidth*ZoneHeight)
  INT8U ZoneID;             //区域ID
  INT8U ZoneType;           //内容类型，详见说明
  INT8U CodeType;           //编码类型（0：单字节ASCII编码， 1：双字节Unicode编码）
  INT8U InMode;             //入花样（0:跳入< 默认>，详见说明）
  INT8U OutMode;            //出花样（0:跳入< 默认>，详见说明）
  INT8U Align;              //水平垂直对齐,默认居中，详见说明
  INT8U FGColor_R;          //内容前景红色，默认0xFF
  INT8U FGColor_G;          //内容前景绿色，默认0x00
  INT8U FGColor_B;          //内容前景蓝色，默认0x00
  INT8U BGColor_R;          //内容背景红色，默认0x00
  INT8U BGColor_G;          //内容背景绿色，默认0x00
  INT8U BGColor_B;          //内容背景蓝色，默认0x00
  INT8U AutoLine;           //自动换行（0：不自动<默认>，1：自动换行）
  INT8U AutoWidth;          //变宽字体（0：等宽字体 1：非等宽字体<默认>）
  INT16U Speed;             //花样速度（n pixels per 10ms，默认1）
```

```
    INT8U StayTime;          //停留时间（单位秒，0：无限循环显示，默认1）
    INT8U Times;             //播放次数（0：无限循环显示，默认0）
    INT8U FontStyle;         //字体，详见说明
    INT8U LineSpace;         //行间距（默认1）
    INT8U CloumnSpace;       //列间距（默认1）
    INT8U Rev;               //保留
    INT16U BlinkOnTime;      //闪烁亮的时间（单位：10ms，默认50）
    INT16U BlinkOffTime;     //闪烁灭的时间（单位：10ms，默认50）
    INT16U ContentSize;      //内容数据的长度
    INT8U Rev2[2];           //保留
}ZoneDisplaySet;
```

说明：ZoneSize 区域分辨率大小（即 ZoneWidth*ZoneHeight），ZoneID 区域 ID，ZoneType 内容类型（0：文字，1：图片，2 特殊对象，由于有特殊控制符的配合使用，通常使用 0 即可），CodeType 编码类型（0：单字节 ASCII 编码， 1：双字节 Unicode 编码），InMode 入花样（0:跳入< 默认>，1：左移动，2：右移动，3：上移动，4：下移动，5：左拉动，6：右拉动，7：上拉动，8：下拉动），OutMode 出花样（0:跳入< 默认>，1：左移动，2：右移动，3：上移动，4：下移动，5：左拉动，6：右拉动，7：上拉动，8：下拉动。注意：如需要连续滚动，停留时间需要设置到 0），Align 水平垂直对齐,默认居中（[7:4]垂直方向<0:居中，1:上对齐，2:下对齐> ，[3:0]水平方向<0:居中，1:左对齐，2:右对齐>），FGColor_R,FGColor_G,FGColor_B 内容前景色（默认红色），
BGColor_R,BGColor_G,BGColor_B 内容的背景色（默认黑色）,AutoLine 自动换行（0：不自动<默认>，1：自动换行），AutoWidth 变宽字体（0：等宽字体 1：非等宽字体<默认>），Speed 花样速度（n pixels per 10ms，默认 1）,StayTIme 停留时间（单位秒，0：无限循环显示，默认 1），Times 播放次数(0:无限循环显示,默认 0),FontStyle 字体(参考 APIDemo 的示例,默认 0x32<即 Normal14，8 点宽，14 点高>），LineSpace 行间距（默认 1），CloumnSpace 列间距（默认 1），Rev 保留，
BlinkOnTime 闪烁亮的时间（单位：10ms，默认 50），BlinkOffTime 闪烁灭的时间（单位：10ms，默认 50），ContentSize 内容数据的长度，Rev2 保留

content: **[in]**
要显示的内容

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码(参见 13.20)**。

程序范例

```
//代码清单
   ZoneDisplaySet zoneDisplayEntry;
   memset(&zoneDisplayEntry, 0, sizeof(ZoneDisplaySet));
   zoneDisplayEntry.ZoneSize = zoneSetEntry[0].ZoneWidth *
zoneSetEntry[0].ZoneHeight;
   zoneDisplayEntry.ZoneID = zoneSetEntry[0].ZoneID;
   zoneDisplayEntry.ZoneType = 0;
   zoneDisplayEntry.CodeType = 0;
   zoneDisplayEntry.InMode = 3;
   zoneDisplayEntry.OutMode = 4;
   zoneDisplayEntry.Align = 0;
   zoneDisplayEntry.FGColor_R = 0xFF;
   zoneDisplayEntry.FGColor_G = 0xFF;
```

```
    zoneDisplayEntry.FGColor_B = 0xFF;
    zoneDisplayEntry.BGColor_R = 0;
    zoneDisplayEntry.BGColor_G = 0;
    zoneDisplayEntry.BGColor_B = 0;
    zoneDisplayEntry.AutoLine = 0;
    zoneDisplayEntry.AutoWidth = 1;
    zoneDisplayEntry.Speed = 5;
    zoneDisplayEntry.StayTime = 2;
    zoneDisplayEntry.Times = 0;
    zoneDisplayEntry.FontStyle = CZ_FONT_EN_14x8;
    zoneDisplayEntry.LineSpace = 1;
    zoneDisplayEntry.CloumnSpace = 1;
    zoneDisplayEntry.BlinkOnTime = 0x0A;
    zoneDisplayEntry.BlinkOffTime = 0x0A;
    content = "Welcome";
   zoneDisplayEntry.ContentSize = content.length();

  if(czSetDisplay(1, 1, 0, zoneDisplayEntry, (INT8U*)content.c_str()))==0)
  {
  }
```

相关函数

czSetContentDisplay

13.4 不带属性的显示设置

函数原型

INT32U czSetContentDisplay(INT8U pageID, INT8U zoneNum, ZoneDisplayContentSet parameter, INT8U* content);

描述

让屏显示内容

参数

pageID: **[in]**

帧 ID

zoneNum: **[in]**

区域的数量

parameter: **[in]**

显示内容的数据，结构体如下定义：

```
typedef struct
{
  INT8U ZoneSize;          //区域分辨率大小（即ZoneWidth*ZoneHeight）
  INT8U ZoneID;            //区域ID
  INT16U CodeType;         //编码类型（0：单字节ASCII编码， 1：双字节Unicode编码）
  INT16U ZoneType;         //内容类型，详见说明
  INT16U Rev;              //保留
  INT16U ContentSize;      //内容数据的长度
}ZoneDisplayContentSet;
```

说明：ZoneSize 区域分辨率大小（即 ZoneWidth*ZoneHeight），ZoneID 区域 ID，CodeType 编码类型（0：单字节 ASCII 编码， 1：双字节 Unicode 编码），ZoneType 内容类型（0：文字，1：图片，2 特殊对象，由于有特殊控制符的配合使用，通常使用 0 即可），Rev 保留，ContentSize 内容数据的长度

content: **[in]**
要显示的内容

返回值
如果函数调用成功，返回 **0**，否则返回**错误代码(参见 13.20)**。

程序范例

```
//代码清单
ZoneSetHead zoneSetHead;
ZoneSet zoneSetEntry[3];
memset(&zoneSetHead, 0, sizeof(ZoneSetHead));
memset(zoneSetEntry, 0, sizeof(ZoneSe)*3);
zoneSetHead.ZoneNum = 1;
zoneSetEntry[0].ZoneID = 1;
zoneSetEntry[0].XPos = 0;
zoneSetEntry[0].YPos = 0;
zoneSetEntry[0].ZoneWidth = 0x40;
zoneSetEntry[0].ZoneHeight = 0x10;

ZoneDisplayContentSet zoneDisplayContentEntry;
memset(&zoneDisplayContentEntry, 0, sizeof(ZoneDisplayContentSet));
zoneDisplayContentEntry.ZoneSize = zoneSetEntry[0].ZoneWidth *
zoneSetEntry[0].ZoneHeight;
zoneDisplayContentEntry.ZoneID = zoneSetEntry[0].ZoneID;
content = "Hello";
zoneDisplayContentEntry.ContentSize = content.length();

if(czSetContentDisplay(1, 1, zoneDisplayContentEntry,
(INT8U*)content.c_str()))==0)
{
}
```

相关函数
czSetDisplay

13.5 启用/停止多区域显示命令

函数原型
INT32U czSetEnableMulitZone(INT8U enabled);

描述
启用或者停止多区域显示的命令.

参数
enabled: **[in]**
启用的标志

返回值
如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czSetEnableMulitZone(1)==0)
{
}
```

相关函数

czGetMulitZoneSetting

13.6 多区域设置显示页数命令

函数原型

```
INT32U czSetPageCount(INT8U pageCount);
```

描述

多区域设置页数的命令.

参数

pageCount: **[in]**

设置总的页数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码(参见 13.20)**。

程序范例

```
//代码清单
if(czSetPageCount(1)==0)
{
}
```

相关函数

czGetMulitZoneSetting

13.7 特殊区域划分命令

函数原型

```
INT32U czDivideSpeZone(INT8U pageID, SpeZoneSet speZoneSet, SpeZoneXY* speZoneXYArrs);
```

描述

设置特殊区域划分的命令.

参数

pageID: **[in]**

要设置的页面 ID

speZoneSet: **[in]**

要设置的特殊区域划分头

```
typedef struct
{
  INT8U SpeZoneID;          //特殊区域ID
  INT8U PicID;              //图片ID
  INT8U FreshPattern;       //闪烁花样
  INT8U Rev;                //保留
  INT16U FlashHZ;           //闪烁的频率
  INT16U ZoneNum;           //区域的个数
  INT16U ZoneWidth;         //区域的宽度
  INT16U ZoneHeight;        //区域的高度
}SpeZoneSet;
```

说明：SpeZoneID 表示特殊区域 ID，PicID 表示图片 ID，FreshPattern 表示闪烁花样，Rev 保留，FlashHZ 表示闪烁的频率，ZoneNum 表示区域的个数，ZoneWidth 表示区域的宽度，

ZoneHeight 表示区域的高度

speZoneXYArrs: **[in]**
特殊区域划分的 XY 数组

```
typedef struct
{
  INT16U ZoneX;              //区域的X坐标
  INT16U ZoneY;              //区域的Y坐标
}SpeZoneXY;
```

说明：ZoneX 表示区域的 X 坐标,ZoneY 表示区域的 Y 坐标

返回值
如果函数调用成功，返回 **0**，否则返回**错误代码(参见 13.20)**。
程序范例

```
//代码清单
SpeZoneSet speZoneSet;
SpeZoneXY speZoneXYArrs[3];
memset(&speZoneSet, 0, sizeof(SpeZoneSet));
memset(speZoneXYArrs, 0, sizeof(SpeZoneXY)*3);
if(czDivideSpeZone(1, speZoneSet, SpeZoneXY)==0)
{
}
```

相关函数
czDivideZone

13.8 读取多区域使能开关和页数命令

函数原型
INT32U czGetMulitZoneSetting(INT8U* enabled, INT8U* pageCount);
描述
获取多区域的使能开关和页数的命令.
参数
enabled: **[out]**
返回的使能开关的标志
pageCount: **[out]**
返回总页数

返回值
如果函数调用成功，返回 **0**，否则返回**错误代码**。
程序范例

```
//代码清单
INT8U enabled,pageCount;
if(czGetMulitZoneSetting(&enabled, &pageCount)==0)
{
}
```

相关函数
czSetEnableMulitZone, czSetPageCount

13.9 显示特殊字符解析

对于特殊的对象，采用特殊的编码来表示。可在文本中添加

编码	特殊对象说明
时间插入符	
0x0800	HH:MM (24H)
0x0801	HH:MM:SS (24H)
0x0802	HH:MM (12H)
0x0803	HH:MM:SS (12H)
0x0804	HH:MM X(A/P)M (12H)
0x0805	HH:MM:SS X(A/P)M (12H)
0x0806	HH (24H)
0x0807	HH (12H)
0x0808	MM
0x0809	SS
0x080A	X(A/P)M
0x080B	GTM+X:00 (时区)
日期插入符	
0x0810	月/日/年 (数字)
0x0811	日/月/年 (数字)
0x0812	月.日.年 (数字)
0x0813	年-月-日 (数字)
0x0814	年 (数字)
0x0815	月 (数字)
0x0816	月 (英文)
0x0817	月 (英文缩写)
0x0818	日 (数字)
0x0819	星期 (数字)
0x081A	星期 (英文)
0x081B	星期 (英文缩写)
传感器值插入符	
0x0820	温度(摄氏度)
0x0821	温度(华氏)
0x0822	湿度
0x0823	雷达速度 KPH
0x0824	雷达速度 MPH
0x0825	限速 KPH
特殊控制符	
0x0830	字体颜色，后面跟 3 个 byte 表示 RGB，如 0x080x30 0xFF0x000x00 ，表示红色 0x080x30 0x00xFF0x00，表示绿色 0x080x20 0x000x000xFF，表示蓝色
0x0831	更换字体，后面跟 1 个 byte 表示字体编号，如 0x080x31 0x31，编号为 0x31 的字体 0x080x31 0x32，编号为 0x32 的字体

0x0832	闪烁控制符，必须成对使用，夹在中间的为闪烁的内容。 针对 CZ9270，data 中用该字符 全部都会在停留的时候闪烁。
0x0833	图片插入符，后面跟 1 个 byte 表示图片编号，如 0x080x33 0x01，编号为 1 的图片 0x080x33 0x02，编号为 2 的图片 图库路径： F：\P\x.bmp 默认： 图片大小：16X16 编号 1 的图片为：向上绿箭头 编号 2 的图片为：向下绿箭头 编号 3 的图片为：向左绿箭头 编号 4 的图片为：向右绿箭头 编号 5 的图片为：向左上绿箭头 编号 6 的图片为：向右上绿箭头 编号 7 的图片为：向左下绿箭头 编号 8 的图片为：向右下绿箭头 编号 9 的图片为：红叉 编号 10 的图片为：用户自用 ….

13.20 多区域错误码特别说明

错误码	描述
0x36E1	PageID 无效(可能是 PageID 超出实际页数，或者 PageID 为 0)
0x36E2	PageID 有效，但该页还未被配置
0x36E3	设置总页数失败，超出最大值，当前最大值 12 页
0x36E4	ZoneID 无效(可能是 ZoneID 超出该页实际区域数，或者 ZoneID 为 0)
0x36E5	ZoneID 有效，但该区域还未被配置
0x36E6	ZoneCount 无效(命令中的总区域数与实际总区域数不匹配，或者超出最大值，当前区域最多 20 个)

14 播放列表操作

该命令主要用于播放列表的操作

14.1 初始化播放列表工作目录(czPLInit)

函数原型

```
INT32U czPLInit (INT8U* workPath,INT8U* playListName)
```

描述

打开播放列表并加载，如果不存在，就创建一个空的。此函数是最开始调用的。

参数

workPath：[in]

播放列表工作目录，播放列表及播放文件放于此处。

playListName: [in]

播放列表文件名

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U workPath[128],playListName[256];
workPath = "\\";
playListName = "DEMO.LST";
if(czPLInit(workPath,playListName) == 0)
{
}
```

相关函数

czLoadSYSFromXML, czPLSpeSendToLED, czReadSpePlayListIndex, czPlaySpePlaylist

14.2 加载播放列表文件(czLoadSYSFromXML)

函数原型

```
INT32U czLoadSYSFromXML (char* fileName)
```

描述

加载 XML 格式的播放列表文件

参数

fileName: [in]

播放列表文件名

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U playListName[256];
playListName = "DEMO.LST";
if(czLoadSYSFromXML(playListName) == 0)
{
}
```

相关函数

czPLInit, czPLSpeSendToLED, czReadSpePlayListIndex, czPlaySpePlaylist

14.3 发送预定义播放列表(czPLSpeSendToLED)

函数原型

```
INT32U czPLSpeSendToLED(INT8U playListIndex,INT8U isSendChangedFiles)
```

描述

发送预定义播放列表到显示屏，如果文件内容更新了需要发送，就先发送文件。

参数

playListIndex：[in]

预播放列表索引,范围：[1--255]

isSendChangedFiles: [in]

是否发送内容更新后的文件，0：不发送，1：发送

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U workPath[128],playListName[256];
workPath = ".\\";
playListName = "DEMO.LST";
czPLInit(workpath, playListName);
if(czPLSpeSendToLED(1,1) == 0)
{
}
```

相关函数

czLoadSYSFromXML, czPLInit, czReadSpePlayListIndex, czPlaySpePlaylist

14.4 获取当前在播的预定义播放列表的索引(czReadSpePlayListIndex)

函数原型

```
INT32U czReadSpePlayListIndex(INT8U* playListIndex)
```

描述

获取当前在播的预定义播放列表的索引。返回 0xFF 表示当前没有播放预定义播放列表。

参数

playListIndex：[out]

播放列表索引。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U curPLIndex;
if(czReadSpePlayListIndex(&curPLIndex) == 0)
{
}
```

相关函数

czLoadSYSFromXML, czPLSpeSendToLED, czPLInit, czPlaySpePlaylist

14.5 指定播放预定义播放列表(czPlaySpePlaylist)

函数原型

```
INT32U czPlaySpePlaylist(INT8U playListIndex)
```

描述

指定播放预定义播放列表。

参数

playListIndex：[in]

播放列表索引

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czPlaySpePlaylist(1) == 0)
{
}
```

相关函数

czLoadSYSFromXML, czPLSpeSendToLED, czReadSpePlayListIndex, czPLInit

15 像点检测操作

该命令主要用于像点检测的操作

15.1 开始进行像点检测(czBeginPixCheck)

函数原型

```
INT32U czBeginPixCheck()
```

描述

开始进行像点检测

参数

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czBeginPixCheck() == 0)
{
}
```

相关函数

czPixProgress, czReadPixResult, czReadSignalResult, czPxlChk

15.2 查询像点检测的进度(czPixProgress)

函数原型

```
INT32U czPixProgress(INT8U* Progress)
```

描述

查询像点检测的进度

参数

Progress: **[out]**

进度值，有效值[0-100]，100 表示完成

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
INT8U Progress;
if(czPixProgress(&Progress) == 0)
{
}
```

相关函数

czBeginPixCheck, czReadPixResult, czReadSignalResult, czPxlChk

15.3 获取像素点检测结果(czReadPixResult)

函数原型

```
INT32U czReadPixResult(INT8U* PCPath)
```

描述

获取像点检测的像素点检测结果

参数

PCPath: **[in]**

读回文件存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czReadPixResult("D:\\pxl.csv") == 0)
{
}
```

相关函数

czBeginPixCheck, czPixProgress, czReadSignalResult, czPxlChk

15.4 获取信号检测结果(czReadSignalResult)

函数原型

```
INT32U czReadSignalResult(INT8U* PCPath)
```

描述

获取像点检测的信号检测结果

参数

PCPath: **[in]**

读回文件存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czReadSignalResult("D:\\signal.csv") == 0)
{
}
```

相关函数

czBeginPixCheck, czPixProgress, czReadPixResult, czPxlChk

15.5 像点检测(czPxlCheck)

函数原型

```
INT32U czPxlCheck(INT8U* PCPxlPath, INT8U* PCSignalPath)
```

描述

执行阻塞式像点检测，可以通过初始化接口中的 update_status 的回调获得进度。

具体操作参见 API Demo 工程中的范例。

参数

PCPxlPath: **[in]**

读回像素点结果文件存储位置。

PCSignalPath: **[in]**

读回信号结果文件存储位置。

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
//代码清单
if(czPxlCheck("D:\\pxl.csv","D:\\signal.csv") == 0)
{
}
```

相关函数

czBeginPixCheck, czPixProgress, czReadPixResult, czReadSignalResult

四、Easy API 接口定义

Easy API 是对基础操作接口一种封装,对普通的应用实现简单的操作接口,目前 Easy API 只支持网络操作.

1. 在显示屏上显示 Text 信息(czShowMsg)

函数原型

```
INT32U czShowMsg(char* msg, INT32U msg_size, INT32U font,INT32U color,INT32U mode,
                 INT32U stay_time_sec, char* sign_ip,INT32U sign_port,INT32U is_store_ram=0,
                 INT32U is_send_playlist=1);
```

描述

发送一条信息到显示屏显示.

参数

msg: **[in]**
需要显示的信息。

msg_size: **[in]**
显示信息大小。

font :**[in]**
字体,参考API Demo里的示例

color :**[in]**
颜色,参考 color define,如果最低字节定义为'/'表示自定义颜色,排列为"RGB/"

mode :**[in]**
花样,参考 mode define

stay_time_sec :**[in]**
停留时间,0-9999 sec

sign_ip :**[in]**
显示屏 IP 地址

sign_port :**[in]**
显示屏端口

is_store_ram :**[in]**
是否保存了 RAM 中,1=表示保存在 RAM 中，0 表示保存在 F 盘或 D 盘

is_send_playlist :**[in]**
是否发送播放列表,1=表示更新信息后再发播放列表，0 表示只更新播放内容，不发列表

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
if(czShowMsg("show demo msg", 13, CZ_FONT_EN_14x8, CZ_COLOR_AMBER ,
              CZ_MODE_STATIC, 0, TEST_SIGN_IP, 9520,1)== 0)
{
    //OK
}
```

相关函数

czShowPic

2．在显示屏上显示 picture 信息(czShowPic)

函数原型

```
INT32U czShowPic(char* bitmap, INT32U bmp_size, INT32U mode, INT32U stay_time_sec,
                 char* sign_ip,INT32U sign_port,INT32U is_store_ram=0);
```

描述

发送一张 bmp 图片到显示屏显示.

参数

bitmap:**[in]**
  bmp 图片
bmp_size:**[in]**
  信息大小
mode:**[in]**
  花样,参考 mode define
stay_time_sec:**[in]**
  停留时间,0-9999 sec
sign_ip:**[in]**
  显示屏 IP 地址
sign_port:**[in]**
  显示屏端口
is_store_ram:**[in]**
  是否保存了 RAM 中,1=表示保存在 RAM 中，0 表示保存在 F 盘或 D 盘

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
FILE * fp = fopen(".\\demo.bmp", "rb");
if(fp)
{
    long size = get_file_size(fp);
    char *buf = (char*)malloc(size);
    if(buf)
    {
        fread(buf, 1, size, fp);
        czShowPic(buf, size, CZ_MODE_RAND,3, TEST_SIGN_IP, 9520);
    }
    free(buf);
    fclose(fp);
}
```

相关函数

czShowMsg

3．在显示屏上显示一组文件 (czShowFiles)

函数原型

```
INT32U czShowFiles(char* files[], INT32U numfiles, INT32U mode, INT32U stay_time_sec,
                   char* sign_ip,INT32U sign_port,INT32U is_store_ram=0);
```

描述

控制显示屏显示一组文件，显示屏循环显示这些文件.文件类型可以是 bmp,pmg,nmg,qst,flw 格式,
如果发送 F 盘成功,就把内容存放到 F 盘,否则就存放到 D 盘

参数

files **:[in]**
文件在本机 PC 的路径，2 维数组结构
numfiles **:[in]**
文件个数,即 1 维的长度
mode **:[in]**
花样,参考 mode define,只对 bmp 有效,其它文件里文件格式里控制,详情请参考 JetFileII 里的文件格式章节.
stay_time_sec **:[in]**
停留时间,0-9999 sec,只对 bmp 有效,
sign_ip **:[in]**
显示屏 IP 地址
sign_port **:[in]**
显示屏端口
is_store_ram **:[in]**
是否保存了 RAM 中,1=表示保存在 RAM 中，0 表示保存在 F 盘或 D 盘

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
char *file[3] =
{
    {".\\1.bmp"},
    {".\\2.bmp"},
    {".\\3.bmp"}
};
INT32U RT = czShowFiles((char**)file, 3, CZ_MODE_MOVEUP, 2,TEST_SIGN_IP, 9520);
```

相关函数

czShowMsg

4．从显示屏读取文件 (czEasyReadFile)

函数原型

```
INT32U czEasyReadFile(char* pc_file_path, char* sign_file_path,char* sign_ip,INT32U sign_port);
```

描述

从显示屏读取文件到 PC 上

参数

pc_file_path **:[in]**
文件在本机 PC 的路径和文件名
sign_file_path **:[in]**
文件在本机 Sign 的路径和文件名
sign_ip **:[in]**
显示屏 IP 地址
sign_port **:[in]**
显示屏端口

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT32U RT = czEasyReadFile ((char*)"E:\\1", (char*)"C:\\TEST.TXT",
(char*)"169.254.10.49", 9520);
```

相关函数

czEasyWriteFile, czReadSpecPathFile

5．写文件到显示屏 (czEasyWriteFile)

函数原型

```
INT32U czEasyWriteFile(char* pc_file_path, char* sign_file_path,char* sign_ip,INT32U sign_port);
```

描述

写文件到显示屏上

参数

pc_file_path **:[in]**

文件在本机 PC 的路径和文件名

sign_file_path **:[in]**

文件在本机 Sign 的路径和文件名

sign_ip **:[in]**

显示屏 IP 地址

sign_port **:[in]**

显示屏端口

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
    INT32U RT = czEasyWriteFile ((char*)”E:\\1”, (char*)”C:\\TEST.TXT”,
(char*)”169.254.10.49”, 9520);
```

相关函数

czEasyReadFile, czWriteSpecFile

6．从生成 nmg 文件

函数原型

```
INT32U czShowMsgToNmg(char* msg, INT32U msg_size, char* nmg_path_name,INT32U font,
              INT32U color, INT32U mode, INT32U stay_time_sec);
INT32U czShowBmpToNmg(char* bmp_path_name, char* nmg_path_name, INT32U mode, INT32U stay_time_sec);
```

描述

把 message/bmp 转成 nmg 文件

参数

msg **:[in]**

message

msg_size **:[in]**

信息大小

nmg_path_name**:[in]**

保存 nmg 文件的位置

font **:[in]**

字体

color **:[in]**

颜色

mode **:[in]**

显示花样

stay_time_sec**:[in]**

停留时间，单位为秒

bmp_path_name:**[in]**

bmp 文件路径和文件名

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
    INT32U RT = czShowMsgToNmg("Demo show",9,nmgfile[2],CZ_FONT_EN_14x8, CZ_COLOR_AMBER ,
CZ_MODE_STATIC,6);
```

相关函数

czShowBmpToNmg, czShowMsgToNmg

7．在显示屏上显示 Text 信息(czShowMsgEx)(支持串口)

函数原型

```
INT32U czShowMsgEx(char* msg, INT32U msg_size, INT32U font,INT32U color,INT32U mode,
                   INT32U stay_time_sec, char* sign_ip,INT32U sign_port,INT32U is_store_ram=0,
                   INT32U is_send_playlist=1,
                   conn_type type=COMM_UDP,char* comStr="COM3",INT32U baudrate=9600,INT16U
                   dstAddr=0x0101);
```

描述

发送一条信息到显示屏显示.

参数

msg: **[in]**

需要显示的信息。

msg_size: **[in]**

显示信息大小。

font :**[in]**

字体,参考API Demo里的示例

color :**[in]**

颜色,参考 color define,如果最低字节定义为'/'表示自定义颜色,排列为"RGB/"

mode :**[in]**

花样,参考 mode define

stay_time_sec :**[in]**

停留时间,0-9999 sec

sign_ip :**[in]**

显示屏 IP 地址

sign_port :**[in]**

显示屏端口

is_store_ram :**[in]**

是否保存了 RAM 中,1=表示保存在 RAM 中，0 表示保存在 F 盘或 D 盘

is_send_playlist :**[in]**

是否发送播放列表,1=表示更新信息后再发播放列表，0 表示只更新播放内容，不发列表

type :**[in]**

通信类型,定义如下

```
typedef enum
{
  COMM_UDP,
  COMM_TCP,
 COMM_COM
```

}conn_type;

comStr **:[in]**

串口通信时用到的 com 字符，如 COM1

baudrate **:[in]**

串口通信时用到的 baudrate,数值型,如 115200

dstAddr **:[in]**

显示屏(目的)GGUU 地址,Unit Addr.在高位,Group Addr 在低位

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
if(czShowMsgsEx((char**)file, 3, CZ_MODE_MOVEUP, 2,TEST_SIGN_IP, 9520,
              0,1,COMM_COM,TEST_SIGN_COM,TEST_SIGN_COM_BAUDRATE,TEST_SI
              GN_COM_GGUU)== 0)
{
    //OK
}
```

相关函数

czShowPicEx

8．在显示屏上显示 picture 信息(czShowPicEx)（支持串口）

函数原型

```
INT32U czShowPicEx(char* bitmap, INT32U bmp_size, INT32U mode, INT32U stay_time_sec,
                   char* sign_ip,INT32U sign_port,INT32U is_store_ram=0,
                   conn_type type=COMM_UDP,char* comStr="COM3",INT32U baudrate=9600,INT16U
                   dstAddr=0x0101);
```

描述

发送一张 bmp 图片到显示屏显示.

参数

bitmap:**[in]**

bmp 图片

bmp_size:**[in]**

信息大小

mode:**[in]**

花样,参考 mode define

stay_time_sec:**[in]**

停留时间,0-9999 sec

sign_ip:**[in]**

显示屏 IP 地址

sign_port:**[in]**

显示屏端口

is_store_ram:**[in]**

是否保存了 RAM 中,1=表示保存在 RAM 中，0 表示保存在 F 盘或 D 盘

type **:[in]**

通信类型,定义如下

```
typedef enum
 {
```

```
        COMM_UDP,
        COMM_TCP,
       COMM_COM
    }conn_type;
```

comStr **:[in]**

串口通信时用到的 com 字符，如 COM1

baudrate **:[in]**

串口通信时用到的 baudrate,数值型,如 115200

dstAddr **:[in]**

显示屏(目的)GGUU 地址,Unit Addr.在高位,Group Addr 在低位

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
FILE * fp = fopen(".\\demo.bmp", "rb");
if(fp)
{
    long size = get_file_size(fp);
    char *buf = (char*)malloc(size);
    if(buf)
    {
        fread(buf, 1, size, fp);
        czShowPicEx(buf, size, CZ_MODE_RAND,3, TEST_SIGN_IP, 9520,0
        ,COMM_COM,TEST_SIGN_COM,TEST_SIGN_COM_BAUDRATE,TEST_SIGN_COM_GGUU);
    }
    free(buf);
    fclose(fp);
}
```

相关函数

czShowMsgEx

9．在显示屏上显示一组文件 (czShowFilesEx)（支持串口）

函数原型

```
INT32U czShowFilesEx(char* files[], INT32U numfiles, INT32U mode, INT32U stay_time_sec,
                char* sign_ip,INT32U sign_port,INT32U is_store_ram=0,
                conn_type type=COMM_UDP,char* comStr="COM3",INT32U baudrate=9600,INT16U
                dstAddr=0x0101);
```

描述

控制显示屏显示一组文件，显示屏循环显示这些文件.文件类型可以是 bmp,pmg,nmg,qst,flw 格式,如果发送 F 盘成功,就把内容存放到 F 盘,否则就存放到 D 盘

参数

files **:[in]**

文件在本机 PC 的路径，2 维数组结构

numfiles **:[in]**

文件个数,即 1 维的长度

mode **:[in]**

花样,参考 mode define,只对 bmp 有效,其它文件里文件格式里控制,详情请参考 JetFileII 里的文件格式章节.

stay_time_sec **:[in]**

停留时间,0-9999 sec,只对 bmp 有效,

sign_ip **:[in]**

显示屏 IP 地址

sign_port **:[in]**

显示屏端口

is_store_ram **:[in]**

是否保存了 RAM 中,1=表示保存在 RAM 中, 0 表示保存在 F 盘或 D 盘

type **:[in]**

通信类型,定义如下

```
typedef enum
{
  COMM_UDP,
  COMM_TCP,
  COMM_COM
}conn_type;
```

comStr **:[in]**

串口通信时用到的 com 字符, 如 COM1

baudrate **:[in]**

串口通信时用到的 baudrate,数值型,如 115200

dstAddr **:[in]**

显示屏(目的)GGUU 地址,Unit Addr.在高位,Group Addr 在低位

返回值

如果函数调用成功, 返回 **0**, 否则返回**错误代码**。

程序范例

```
char *file[3] =
{
    {".\\1.bmp"},
    {".\\2.bmp"},
    {".\\3.bmp"}
};
INT32U RT = czShowFilesEx((char**)nmgfile, 3, CZ_MODE_MOVEDOWN, 2,TEST_SIGN_IP, 9520, 0,
                          COMM_COM,TEST_SIGN_COM,TEST_SIGN_COM_BAUDRATE,TEST_SIGN_COM_GGUU);
```

相关函数

czShowMsgEx

10. 从显示屏读取文件 (czEasyReadFileEx) (支持串口)

函数原型

```
INT32U czEasyReadFileEx(char* pc_file_path, char* sign_file_path,char* sign_ip,INT32U sign_port,
                        conn_type type=COMM_UDP,char* comStr="COM3",INT32U baudrate=9600,INT16U
                        dstAddr=0x0101);
```

描述

从显示屏读取文件到 PC 上

参数

pc_file_path **:[in]**
文件在本机 PC 的路径和文件名
sign_file_path **:[in]**
文件在本机 Sign 的路径和文件名
sign_ip **:[in]**
显示屏 IP 地址
sign_port **:[in]**
显示屏端口
type **:[in]**
通信类型,定义如下

```
typedef enum
{
  COMM_UDP,
  COMM_TCP,
  COMM_COM
}conn_type;
```

comStr **:[in]**
串口通信时用到的 com 字符，如 COM1
baudrate **:[in]**
串口通信时用到的 baudrate,数值型,如 115200
dstAddr **:[in]**
显示屏(目的)GGUU 地址,Unit Addr.在高位,Group Addr 在低位

返回值
如果函数调用成功，返回 **0**，否则返回**错误代码**。
程序范例

```
INT32U RT = czEasyReadFileEx ((char*)" E:\\1", (char*)" C:\\TEST.TXT", (char*)" 169.254.10.49",
9520, COMM_COM,TEST_SIGN_COM,TEST_SIGN_COM_BAUDRATE,TEST_SIGN_COM_GGUU);
```

相关函数
czEasyWriteFile, czReadSpecPathFile

11．写文件到显示屏 (czEasyWriteFileEx)（支持串口）

函数原型

```
INT32U czEasyWriteFileEx(char* pc_file_path, char* sign_file_path,char* sign_ip,INT32U sign_port,
                         conn_type type=COMM_UDP,char* comStr="COM3",INT32U baudrate=9600,INT16U
                         dstAddr=0x0101);
```

描述
写文件到显示屏上
参数
pc_file_path **:[in]**
文件在本机 PC 的路径和文件名
sign_file_path **:[in]**
文件在本机 Sign 的路径和文件名
sign_ip **:[in]**
显示屏 IP 地址
sign_port **:[in]**
显示屏端口
type **:[in]**

通信类型,定义如下

```
typedef enum
{
  COMM_UDP,
  COMM_TCP,
  COMM_COM
}conn_type;
```

comStr **:[in]**

串口通信时用到的 com 字符，如 COM1

baudrate **:[in]**

串口通信时用到的 baudrate,数值型,如 115200

dstAddr **:[in]**

显示屏(目的)GGUU 地址,Unit Addr.在高位,Group Addr 在低位

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
INT32U RT = czEasyWriteFileEx((char*)”E:\\1”, (char*)”C:\\TEST.TXT”, char*)”169.254.10.49”, 9520, COMM_COM,TEST_SIGN_COM,TEST_SIGN_COM_BAUDRATE,TEST_SIGN_COM_GGUU);
```

相关函数

czEasyReadFile, czWriteSpecFile

12．在显示屏上显示一组文件 (czShowFilesSpe)（支持串口）

函数原型

```
INT32U czShowFilesSpe(char* files[], INT32U numfiles, INT32U mode, INT32U stay_time_sec,
                      char* sign_ip,INT32U sign_port,INT32U is_store_ram=0,
                      conn_type type=COMM_UDP,char* comStr="COM3",INT32U baudrate=9600,INT16U
                      dstAddr=0x0101,INT8U sendType=0);
```

描述

控制显示屏显示一组文件，显示屏循环显示这些文件.文件类型可以是 bmp,pmg,nmg,qst,flw 格式,如果发送 F 盘成功,就把内容存放到 F 盘,否则就存放到 D 盘,将发送文件和发送播放列表分开,多个屏同步更新内容时用到。

参数

files **:[in]**

文件在本机 PC 的路径，2 维数组结构

numfiles **:[in]**

文件个数,即 1 维的长度

mode **:[in]**

花样,参考 mode define,只对 bmp 有效,其它文件里文件格式里控制,详情请参考 JetFileII 里的文件格式章节.

stay_time_sec **:[in]**

停留时间,0-9999 sec,只对 bmp 有效，

sign_ip **:[in]**

显示屏 IP 地址

sign_port **:[in]**

显示屏端口

is_store_ram **:[in]**

是否保存了 RAM 中,1=表示保存在 RAM 中，0 表示保存在 F 盘或 D 盘

type **:[in]**

通信类型,定义如下

```
typedef enum
{
  COMM_UDP,
  COMM_TCP,
  COMM_COM
}conn_type;
```

comStr **:[in]**

串口通信时用到的 com 字符，如 COM1

baudrate **:[in]**

串口通信时用到的 baudrate,数值型,如 115200

dstAddr **:[in]**

显示屏(目的)GGUU 地址,Unit Addr.在高位,Group Addr 在低位

sendType **:[in]**

发送方式 0：发送文件+播放列表
1：仅发送文件
2：仅发送播放列表

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
char *file[3] =
{
    {".\\1.bmp"},
    {".\\2.bmp"},
    {".\\3.bmp"}
};
//only send files
INT32U RT = czShowFilesSpe((char**)nmgfile, 3, CZ_MODE_MOVEDOWN, 2,IPArr[i], 9520, 0,
            COMM_COM,TEST_SIGN_COM,TEST_SIGN_COM_BAUDRATE,TEST_SIGN_COM_GGUU,1);

//do other things

//play files
INT32U RT = czShowFilesSpe((char**)nmgfile, 3, CZ_MODE_MOVEDOWN, 2,TEST_SIGN_IP, 9520, 0,
            COMM_COM,TEST_SIGN_COM,TEST_SIGN_COM_BAUDRATE,TEST_SIGN_COM_GGUU,2);
```

相关函数

czShowMsgEx

13．在显示屏上显示一组文件,每个 bmp 文件停留时间可不同 (czShowFilesII)（支持串口）

函数原型

```
INT32U czShowFilesII(char* files[], INT32U numfiles, INT32U stay_time_sec[], INT32U mode,
                     char* sign_ip,INT32U sign_port,INT32U is_store_ram=0,
                     conn_type type=COMM_UDP,char* comStr="COM3",INT32U baudrate=9600,INT16U
```

dstAddr=0x0101,INT8U sendType=0);

描述

控制显示屏显示一组文件，显示屏循环显示这些文件.文件类型可以是 bmp,pmg,nmg,qst,flw 格式,如果发送 F 盘成功,就把内容存放到 F 盘,否则就存放到 D 盘,将发送文件和发送播放列表分开,多个屏同步更新内容时用到, 如果文件是 bmp，每个 bmp 的停留时间可以不同。

参数

files **:[in]**

文件在本机 PC 的路径，2 维数组结构

numfiles **:[in]**

文件个数,即 1 维的长度

stay_time_sec **:[in]**

多张 bmp 停留时间,0-9999 sec,只对 bmp 有效

mode **:[in]**

花样,参考 mode define,只对 bmp 有效,其它文件里文件格式里控制,详情请参考 JetFileII 里的文件格式章节.

sign_ip **:[in]**

显示屏 IP 地址

sign_port **:[in]**

显示屏端口

is_store_ram **:[in]**

是否保存了 RAM 中,1=表示保存在 RAM 中，0 表示保存在 F 盘或 D 盘

type **:[in]**

通信类型,定义如下

```
typedef enum
{
  COMM_UDP,
  COMM_TCP,
  COMM_COM
}conn_type;
```

comStr **:[in]**

串口通信时用到的 com 字符，如 COM1

baudrate **:[in]**

串口通信时用到的 baudrate,数值型,如 115200

dstAddr **:[in]**

显示屏(目的)GGUU 地址,Unit Addr.在高位,Group Addr 在低位

sendType **:[in]**

发送方式 0：发送文件+播放列表
1：仅发送文件
2：仅发送播放列表

返回值

如果函数调用成功，返回 **0**，否则返回**错误代码**。

程序范例

```
char *file[3] =
{
    {".\\1.bmp"},
    {".\\2.bmp"},
    {".\\3.bmp"}
};
```

```
    //only send files
    INT8U stayTimes[3] = {1,5,8};
    INT32U RT = czShowFilesII((char**)nmgfile, 3, (INT8U*)stayTimes, CZ_MODE_MOVEDOWN,IPArr[i],
9520, 0,COMM_COM,TEST_SIGN_COM,TEST_SIGN_COM_BAUDRATE,TEST_SIGN_COM_GGUU,1);

    //do other things

    //play files
    RT = czShowFilesII((char**)nmgfile, 3, (INT8U*)stayTimes,(INT8U*)CZ_MODE_MOVEDOWN,
TEST_SIGN_IP, 9520, 0,COMM_COM,TEST_SIGN_COM,TEST_SIGN_COM_BAUDRATE,TEST_SIGN_COM_GGUU,2);
```

相关函数

czShowMsgEx

五、应用示例

参考 API Demo 工程

六、附录一

错误码对照表

错误码	描述
DLL 普通错误码	
0x00	成功
0x01	SNMP 读写失败
0x02	调用函数的参数非法
0x03	获取 czinterface 错误，可能是没有调用 init 函数
0x04	调用函数超时
0x0100	调用 DLL 时失败
0x0101	内存分配失败
0x0102	文件格式错误
0x0103	请求修改，但读取的结果不在修改状态下，NTCIP 使用
0x0104	任务不存在
0x0105	DLL 没有该功能，或显示屏不支持该功能
JetFileII 错误码	
0x4B4F	正确处理完一个命令的回送码
0x5245	不正确的回送码
0x9000	正确处理完一个命令的回送码
0x9001	通信同步头错误
0x9002	和校验错误
0x9003	地址错误
0x9004	大类命令非法
0x9005	小类命令非法
0x9006	包长度不对
0x9008	文件不存在
0x9009	文件已到文件尾，再读文件时，包序号过大

0x9010	打开文件失败
0x9011	小类命令在这个系统中不支持
0x9012	文件写入失败
0x9013	文件写入时，包大小大于 1500 或数据大小多余分包大小
0x9014	文件写入时，没有按顺序发送
0x9015	文件删除时，这个文件正在打开
0x9030	请先登录，设置了密码管理，没有登录而进行的操作时返回
0x9031	密码不对
0x9032	用户名不对
0x9033	旧密码不对
0x9035	在别处已经登录了
0x9036	用户非法，或没有权限
0x1101	绝对地址读时，要读取的数量太大
0x1102	绝对地址读写时，硬盘读写错误，可能是读取地址不正确
0x1F01	CPU 升级不成功
0x1F02	正在进行升级操作
0x1F03	没有进行升级操作
0x2101	写文件时，写入的数据量超过了 320K
0x2102	写文件时，磁盘空间不足
0x2103	写文件时，C 盘空间不足
0x2104	写文件时，D 盘空间不足
0x2105	写文件时，E 盘空间不足
0x2106	写文件时，F 盘空间不足
0x2107	写文件时，G 盘空间不足
0x2000	写入文件时，管理内存块申请失败
0x2901	写紧急消息时，不能大于 1024 个字节
0x3A01	系统不支持灰度测试
0x5201	时间设置不成功
0x6601	无效的 license
0x6701	当前没有显示文件
0x6702	当前显示文件打开失败
0x6703	当前显示文件太大，请用扩展读取命令
0x7201	格式化失败，可能原因是指定盘符不存在
0x7301	文件夹创建失败
0x7401	文件改名失败
0x7402	改名时，路径格式不对
0x7501	移动文件失败
0x7601	删除文件失败
0x7B01	打开文件夹失败
0x7D01	读取磁盘信息失败(磁盘不存在，磁盘名称不对)
0x7E01	指定的文件不存在
多区域错误码	
0x36E1	PageID 无效(可能是 PageID 超出实际页数，或者 PageID 为 0)
0x36E2	PageID 有效，但该页还未被配置
0x36E3	设置页总数失败，超出最大值，当前最多 12 页
0x36E4	ZoneID 无效(可能是 ZoneID 超出该页实际区域数，或者 ZoneID 为 0)
0x36E5	ZoneID 有效，但该区域还未被配置
0x36E6	ZoneCount 无效(命令中的总区域数与实际总区域数不匹配，或者超

	出最大值，当前最多 20)
PC 软件错误码	
0x5000	用户错误
0x5001	打开 Com 错误
0x5002	连接超时
0x5003	操作失败
0x5004	头错误
0x5005	源地址错误
0x5006	目的地址错误
0x5007	包序号错误
0x5008	主命令错误
0x5009	子命令错误
0x5010	和校验错误
0x5011	操作放弃
0x5012	没有播放列表
0x5013	
0x5100	通信错误
0x5101	打开 PC 文件失败
0x5102	文件为空
0x8301	OLD_BUF CUR_BUF 在播放，DISPLAY_BUF 可以接收
0x8302	CUR_BUF DISPLAY_BUFz 在播放，OLD_BUF 可以接收
0x8303	DISPLAY_BUF OLD_BUF 在播放，CUR_BUF 可以接收
0x8305	系统不在无线连接播放状态
0x8306	发进去的数据超出范围，接收的缓冲区超出
0x8307	发送的数据格式不对，屏处于 8:8:8:n 数据位 1:1 格式时出现
控制板操作错误错误码	
0x0C00	请求的命令尚在缓冲中
0x0C01	任务成功
0x0C02	队列号错误
0x0C03	正在处理中
0x0C04	端点总线故障
0x0C05	重试次数大于设定次数
0x0C06	一般错误
0x0C07	命令 ID 错误
0x0C08	广播写数据部分不成功
0x0C09	LDU 控制
0x0C11	LDU 控制地址失败
0x0C12	TWI 数据错误
0x0C13	像点检测错误
0xFFFF	返回队列满的错误信息