

Acute Data Generator – PMBus protocol Software development kit (SDK) Programming guide

For Data Generator 3000 and TravelData 3000

Version: 1.1

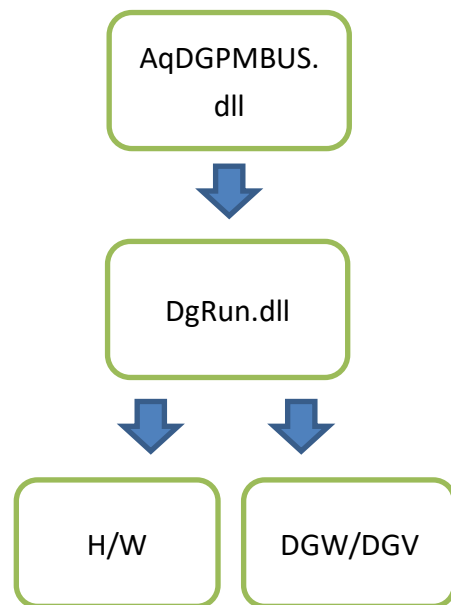
Publish: 2019/12/11

内容

SDK 架构介绍	3
SDK 函式说明	3
bool InitProtocol(int iDGModel, bool fConnect).....	4
HDGPTL PMBus_Init(UINT32 iProtocolClockFreqInKHz, UINT32 iDGInitState, bool fSingleStep, int iSclkNo, int iSdataNo, int iFormat, bool fRepeat).....	5
bool CloseProtocol(HDGPTL hDGPTl).....	6
bool ClearProtocolPacket(HDGPTL hDGPTl).....	6
int SaveProtocolList(HDGPTL hDGPTl, bool fFile, char* pPtr).....	7
bool AppendDGInstruction(HDGPTL hDGPTl, int iInst, int iParam, DGADDR iDGAddr).....	7
int GetLastDGError().....	8
int GetPodNum().....	9
bool SetOutputVolt(int imV, int iPodIndex).....	9
bool OutputProtocol(HDGPTL hDGPTl).....	9
int GetDGStatus().....	10
bool StopDG().....	10
bool ShutdownDG().....	10
int GetProtocolName(HDGPTL hDGPTl, char* pBuf, int iBufSize).....	10
DGADDR PMBus_AppendIdle(HDGPTL hDGPTl, UINT32 usecs).....	11
DGADDR PMBus_AppendGroupCmdPtl(HDGPTL hDGPTl, GROUP_CMD_PTL* gcp, bool fUsedPEC, int iSlaveRespSet).....	11
DGADDR PMBus_AppendExtCmdRdBytePtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cExtCmd, UINT32 cCmd, bool fUsedPEC, int iSlaveRespSet).....	12
DGADDR PMBus_AppendExtCmdWrBytePtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cExtCmd, UINT32 cCmd, UINT32 cDat, bool fUsedPEC, int iSlaveRespSet).....	13
DGADDR PMBus_AppendExtCmdRdWordPtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cExtCmd, UINT32 cCmd, bool fUsedPEC, int iSlaveRespSet).....	14
DGADDR PMBus_AppendExtCmdWrWordPtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cExtCmd, UINT32 cCmd, UINT32 wDat, bool fUsedPEC, int iSlaveRespSet).....	15
DGADDR PMBus_AppendZoneRdStatDatResp(HDGPTL hDGPTl, UINT32 cCtlCmd, UINT32 cStatMask, int iSlaveRespSet, int iPEC1, int iPEC2).....	16
DGADDR PMBus_AppendZoneRdPMBusCmd(HDGPTL hDGPTl, UINT32 cCtlCmd, UINT32 cPMBusCmd, int iSlaveRespSet, int iPEC1, int iPEC2).....	16
DGADDR PMBus_AppendZoneWrPMBusCmd_TwoDatByte(HDGPTL hDGPTl, UINT32 cPMBusCmd, UINT32 wDat, int iSlaveRespSet).....	17
DGADDR PMBus_AppendAlertRespAddr(HDGPTL hDGPTl, bool fUsedPEC, int iSlaveRespSet).....	18

DGADDR PMBus_AppendQuickCmdPtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cRW, Int iSlaveRespSet).....	18
DGADDR PMBus_AppendSendBytePtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cDat, bool fUsedPEC, int iSlaveRespSet).....	19
DGADDR PMBus_AppendReceiveBytePtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, bool fUsedPEC, int iSlaveRespSet).....	19
DGADDR PMBus_AppendWriteBytePtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cCmd, UINT32 cDat, bool fUsedPEC, int iSlaveRespSet).....	20
DGADDR PMBus_AppendReadBytePtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cCmd, bool fUsedPEC, int iSlaveRespSet).....	21
DGADDR PMBus_AppendWriteWordPtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cCmd, UINT32 wDat, bool fUsedPEC, int iSlaveRespSet).....	21
DGADDR PMBus_AppendReadWordPtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cCmd, bool fUsedPEC, int iSlaveRespSet).....	22
DGADDR PMBus_AppendWrite32Ptl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cCmd, UINT32 dwDat, bool fUsedPEC, int iSlaveRespSet).....	23
DGADDR PMBus_AppendRead32Ptl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cCmd, bool fUsedPEC, int iSlaveRespSet).....	23
DGADDR PMBus_AppendBlockWrPtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cCmd , UINT32 cByteCnt,UINT32* pDatBuf, bool fUsedPEC, int iSlaveRespSet).....	24
DGADDR PMBus_AppendBlockRdPtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cCmd, UINT32 cByteCnt, bool fUsedPEC, int iSlaveRespSet).....	25

SDK 架构介绍



此 SDK 提供一个开放接口让用户可以 2 种方式来发送 PMBus 波形。

1. 以一个一个 PMBus 封包的形式发送。
2. 一次发送所有 PMBus 封包的形式。

SDK 函式说明

```

typedef unsigned int UINT32;
typedef unsigned int HDGPTL;
typedef unsigned int DGADDR;
  
```

```
bool InitProtocol(int iDGModel, bool fConnect)
```

功能

寻找并启动目前接在此计算机上的数据产生器。

参数

iDGModel[in]:

Type: **int**

选择数据产生器机种，列举如下：

```
enum DG_HW_MODEL
```

```
{
```

```
    DG3064B = 0x33064,
```

```
DG3096B = 0x33096,  
DG3128B = 0x33128,  
TD3008E = 0x23008,  
TD3116B = 0x23116,  
TD3216B = 0x23216,  
};
```

fConnect[in]:

Type: **bool**

设置连接模式，false 表示 demo 模式。

回传值

如果回传值为 True，代表模式设置成功。如果回传 False 值则代表设置失败。

备注

InitProtocol(TD3216B, false);

// 选择 TD3216B 机种并设置 demo 模式。

**HDGPTL PMBus_Init(UINT32 iProtocolClockFreqInKHz,UINT32 iDGInitState,
bool fSingleStep, int iSclNo, int iSdaNo, int iFormat, bool fRepeat)**

功能

初始化 SPI/SIPI 总线设置。

参数

iProtocolClockFreqInKHz[in]:

Type: **UINT32**

设置 PMBus 频率，单位: KHz。

iDGInitState[in]:

Type: **UINT32**

设置 PMBus 总线起始状态，共有下列 3 种:

DGINIT_STATE_LOW = 0,

DGINIT_STATE_HIGH = 1,

DGINIT_STATE_HI_Z = 2,

fSingleStep[in]:

Type: **bool**

设置发送模式，选择 true 时，数据产生器将以一个接一个封包的方式来发送 PMBus 封包，反之则是一次全部发送。

iSclNo, iSdaNo[in]:

Type: **int**

设置 PMBus 发送通道编号。

iFormat[in]:

Type: **int**

选择发送的文件格式。

```
#define DGFMT_DGW 0x0001
```

```
#define DGFMT_DGV 0x0002
```

fRepeat[in]:

Type: **bool**

设置是否重复发送。

回传值

回传 handle 数值，代表模式设置成功。如果回传 -1 则代表设置失败。

备注

```
HDGPTL hDG = PMBus_Init (100, DGINIT_STATE_HIGH, true, 0, 1, DGFMT_DGW, true);  
// 设置 100 KHz 频率 PMBus 波形，波形初始阶段为 high。  
// 选择一个封包接着一个封包发送形式，  
// 设置 CH-00 & CH-01 为 Scl & Sda ， DGW 文件格式，重复发送。
```

bool CloseProtocol(HDGPTL hDGPTl)

功能

释放 SDK 所占用的资源。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

回传值

如果回传值为 True，代表模式设置成功。如果回传 False 值则代表设置失败。

bool ClearProtocolPacket(HDGPTL hDGPTl)

功能

清除 PMBus 封包序列。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

回传值

如果回传值为 True，代表模式设置成功。如果回传 False 值则代表设置失败。

int SaveProtocolList(HDGPTL hDGPtr, bool fFile, char* pPtr)

功能

将 PMBus 封包序列存档。

参数

hDGPtr[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

fFile[in]:

Type: **bool**

存入档案(true) 或存入缓冲(false)。

pPtr[in]:

Type: **char***

当 fFile = true 表示档案路径反之则为缓冲区指标。

回传值

回传档案或缓冲大小，代表模式设置成功。如果回传 -1 则代表设置失败。

备注

SaveProtocolList(hDG, true, "PMBus.dgw");

// 储存 "PMBus.dgw" 档案。

**bool AppendDGInstruction(HDGPTL hDGPtr, int iInst, int iParam, DGADDR
iDGAddr)**

功能

加入数据产生器指令。

参数

hDGPtr[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

iInst[in]:

Type: **int**

设置 DG 指令, e.g. JP, LC, LP。

#define CMD_NP	0	// No Operation
#define CMD_LC	1	// Loop Count
#define CMD_LP	2	// Loop to New Address
#define CMD_JP	3	// Jump to New Address
#define CMD_WE	5	// Wait Event

```
#define CMD_HD
```

```
7 // Hold Count
```

iParam[in]:

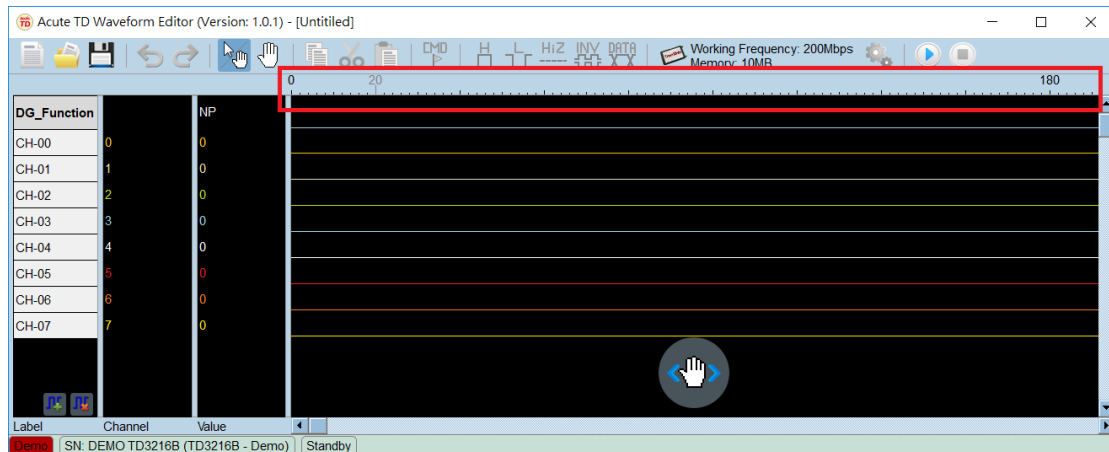
Type: **int**

设置 DG 指令参数, e.g. JP 10, iParam = 10.

iDGAddr[in]:

Type: **DGADDR**

设置欲插入指令位置, 请参考下方截图。



回传值

如果回传值为 **True**, 代表模式设置成功。如果回传 **False** 值则代表设置失败。

备注

```
AppendDGInstruction(hDG, CMD_JP, 0, 1000);
```

```
// 设置 DG 指令:在 DG address = 1000 插入 JP 0 指令。
```

int GetLastDGError()

功能

取得错误代码。

回传错误码或是 0 表示无错误。

错误码

#define ERR_MSG_FILE_NOT_FOUND	0x0001
#define ERR_MSG_CANT_FIND_DLL	0x1001
#define ERR_MSG_EMPTY_SLOT	0x1002
#define ERR_MSG_NO_HARDWARE	0x1004
#define ERR_MSG_INVALID_WORK_FREQ	0x1005
#define ERR_MSG_DUPLICATED_CH_NO	0x1006
#define ERR_MSG_CONFLICTED_HIZ_CH_NO	0x1007
#define ERR_MSG_INVALID_STATUS	0x1008
#define ERR_MSG_NOT_UNDER_CAPTURE	0x1009


```
#define ERR_MSG_NONEXISTENT_HANDLE      0x100A
#define ERR_MSG_INVALID_IDLE_TIME      0x100B
#define ERR_MSG_OVER_DATA_BUFF_SIZE    0x100C
```

int GetPodNum()

功能

取得 POD 数量。

回传值

TD3216B = 2, DG3064B = 6。

bool SetOutputVolt(int imV, int iPodIndex)

功能

设置输出电压。

参数

imV[in]:

Type: **int**

设置单一 pod 输出电压，单位: mV。

iPodIndex[in]:

Type: **int**

设置 pod 索引，从 0 开始。

回传值

如果回传值为 True，代表模式设置成功。如果回传 False 值则代表设置失败。

备注

```
SetOutputVolt(2500, 0);
// 设置 Pod 0, 2.5V, Pod 0: CH0 ~ CH7
SetOutputVolt(2500, 1);
// 设置 Pod 1, 2.5V, Pod 1: CH8 ~ CH15
```

bool OutputProtocol(HDGPTL hDGPTl)

功能

输出 protocol。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

回传值

如果回传值为 **True**，代表模式设置成功。如果回传 **False** 值则代表设置失败。

int GetDGStatus()

功能

取得资料产生器目前状态。

回传值

回传 **DG_WAVEFORM_SENDING(0x80000000)** 表示 DG 在发送状态，其他数值则为准备状态。

bool StopDG()

功能

停止发送。

回传值

如果回传值为 **True**，代表模式设置成功。如果回传 **False** 值则代表设置失败。

bool ShutdownDG()

功能

关闭数据产生器定中断与计算机的 USB 连接。

回传值

如果回传值为 **True**，代表模式设置成功。如果回传 **False** 值则代表设置失败。

int GetProtocolName(HDGPTL** hDGPTl, **char*** pBuf, **int** iBufSize)**

功能

取得目前总线名称与 ID。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

pBuf[in]:

Type: **char***

设置总线名称缓冲区。

iBufSize[in]:

Type: **int**

设置缓冲区大小。

回传值

回传总线 ID。

DGADDR PMBus_AppendIdle(**HDGPTL** hDGPTl, **UINT32** usecs)

功能

加入 PMBus Idle 。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

nsecs[in]:

Type: **UINT32**

设置 Idle 时间，单位: us。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

备注

```
PMBus_AppendDelay(hDG, 1000);
```

```
// Set the idle time 1 ms
```

DGADDR PMBus_AppendGroupCmdPtl(**HDGPTL** hDGPTl, **GROUP_CMD_PTL*** gcp, **bool** fUsedPEC, **int** iSlaveRespSet)

功能

加入 group command protocol 封包。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

gcp[in]:

Type: **GROUP_CMD_PTL***

group command protocol struct.

```
#define MAX_DEVICE_ADDR_SIZE 64
```

```
#define MAX_WORD_DATA_SIZE 64
```

```
typedef struct _GROUPCMDPTL
```

```
{
    UINT32 cDevAddr[MAX_DEVICE_ADDR_SIZE]; // Device address
    UINT32 cCmdCode[MAX_DEVICE_ADDR_SIZE]; // Command code
    UINT32 cDatSize[MAX_DEVICE_ADDR_SIZE]; // Data Size
    UINT32 wWordDat[MAX_WORD_DATA_SIZE]; // Data value
    UINT32 iDeviceSize; // Device numbers
} GROUP_CMD_PTL;

fUsedPEC[in]:
    Type: bool
    使用 Packet Error Checking (PEC).

iSlaveResp[in]:
    Type: int
    设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。
    enum
    {
        NOT_SET_HIZ = 0,
        SET_HIZ
    };
};
```

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

备注

```
GROUP_CMD_PTL gcp;
memset(&gcp, INVALID_DAT_VALUE, sizeof(GROUP_CMD_PTL));
gcp.iDeviceSize = 1; // 1 device
gcp.cDevAddr[0] = 0x10; // device address = 0x10
gcp.cCmdCode[0] = 0x16; // command code = 0x16
gcp.cDatSize[0] = 1; // 1 data size
gcp.wWordDat[0] = 0x4020; // data word = 0x4020
PMBus_AppendGroupCmdPtl(hDG, &gcp, true, SET_HIZ);
```

DGADDR PMBus_AppendExtCmdRdBytePtl(hDGPTL hDGPTl, **UINT32 cSlaveAddr, **UINT32** cExtCmd, **UINT32** cCmd, **bool** fUsedPEC, **int** iSlaveRespSet)**

功能

加入 extended command read byte protocol 封包。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address。

cExtCmd[in]:

Type: **UINT32**

设置 Command extension。

cCmd[in]:

Type: **UINT32**

设置 Command。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC)。

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

备注

PMBus_AppendExtCmdRdBytePtl(hDG, 0x10, 0xFF, 0x16, true, SET_HIZ);

DGADDR PMBus_AppendExtCmdWrBytePtl(HDGPTL** hDGptl, **UINT32** cSlaveAddr, **UINT32** cExtCmd, **UINT32** cCmd, **UINT32** cDat , **bool** fUsedPEC, **int** iSlaveRespSet)**

功能

加入 extended command read byte protocol 封包。

参数

hDGptl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address。

cExtCmd[in]:

Type: **UINT32**

设置 Command extension。

cCmd[in]:

Type: **UINT32**

设置 Command。

cDat[in]:

Type: **UINT32**

设置数据。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC)。

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

备注

PMBus_AppendExtCmdWrBytePtl(hDG, 0x10, 0xFF, 0x16, 0x20, true, SET_HIZ);

DGADDR PMBus_AppendExtCmdRdWordPtl(HDGPTL** hDGPTl, **UINT32** cSlaveAddr, **UINT32** cExtCmd, **UINT32** cCmd, **bool** fUsedPEC, **int** iSlaveRespSet)**

功能

加入 extended command read word protocol 封包。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address。

cExtCmd[in]:

Type: **UINT32**

设置 Command extension。

cCmd[in]:

Type: **UINT32**

设置 Command。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

备注

PMBus_AppendExtCmdRdWordPtl(hDG, 0x10, 0xFF, 0x16, true, SET_HIZ);

DGADDR PMBus_AppendExtCmdWrWordPtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cExtCmd, UINT32 cCmd, UINT32 wDat, bool fUsedPEC, int iSlaveRespSet)

功能

加入 extended command write word protocol 封包。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address。

cExtCmd[in]:

Type: **UINT32**

设置 Command extension。

cCmd[in]:

Type: **UINT32**

设置 Command。

wDat[in]:

Type: **UINT32**

设置数据。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

备注

PMBus_AppendExtCmdWrWordPtl(hDG, 0x10, 0xFF, 0x16, 0x4020, true, SET_HIZ);

DGADDR PMBus_AppendZoneRdStatDatResp(HDGPTL hDGPTl, UINT32 cCtlCmd, UINT32 cStatMask, int iSlaveRespSet, int iPEC1, int iPEC2)

功能

加入 ZONE_READ with STATUS DATA response 封包。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

cCtlCmd[in]:

Type: **UINT32**

设置 Command control。

cStatMask[in]:

Type: **UINT32**

设置 Status mask。

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

iPEC1[in]:

Type: **int**

设置 PEC, 没使用PEC时设为-1。

iPEC2[in]:

Type: **int**

设置 PEC, 没使用 PEC 时设为-1。

回传值

回传 DG address, 代表模式设置成功。如果回传 -1 则代表设置失败。

备注

PMBus_AppendZoneRdStatDatResp(hDG, 0x10, 0xFF, SET_HIZ, -1, -1);

DGADDR PMBus_AppendZoneRdPMBusCmd(HDGPTL hDGPTl, UINT32 cCtlCmd, UINT32 cPMBusCmd, int iSlaveRespSet, int iPEC1, int iPEC2)

功能

加入 ZONE_READ with STATUS DATA response 封包。

参数

hDGpTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

cCtlCmd[in]:

Type: **UINT32**

设置 Command control。

cPMBusCmd[in]:

Type: **UINT32**

设置 PMBus command。

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

iPEC1[in]:

Type: **int**

设置 PEC, 没使用PEC时设为-1。

iPEC2[in]:

Type: **int**

设置 PEC, 没使用 PEC 时设为-1。

回传值

回传 DG address, 代表模式设置成功。如果回传 -1 则代表设置失败。

备注

PMBus_AppendZoneRdPMBusCmd(hDG, 0x10, 0xFF, SET_HIZ, 0x22, 0x44);

DGADDR PMBus_AppendZoneWrPMBusCmd_TwoDatByte(HDGPTL hDGpTl, UINT32 cPMBusCmd, UINT32 wDat, int iSlaveRespSet)

功能

加入 ZONE_WRITE operation with PMBus command and two data bytes 封包。

参数

hDGpTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

cPMBusCmd[in]:

Type: **UINT32**

设置 PMBus command。

wDat[in]:

Type: **UINT32**

设置数据。

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

备注

PMBus_AppendZoneWrPMBusCmd_TwoDatByte(hDG, 0x20, 0x2266, SET_HIZ);

DGADDR PMBus_AppendAlertRespAddr(HDGPtL hDGpTl, bool fUsedPEC, int iSlaveRespSet)

功能

加入 Alert Response Address 封包。

参数

hDGpTl[in]:

Type: **HDGPtL**

PMBus 封包序列 handle。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC)。

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

备注

PMBus_AppendAlertRespAddr(hDG, true, SET_HIZ);

DLLEXDGADDR PMBus_AppendQuickCmdPtl(HDGPtL hDGpTl, UINT32 cSlaveAddr, UINT32 cRW, int iSlaveRespSet)

功能

加入 Quick Command 封包。

参数

hDGpTl[in]:

Type: **HDGPtL**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address 。

cRW[in]:

Type: **UINT32**

Wr: 0; Rd: 1.

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

DLLEXDGADDR PMBus_AppendSendBytePtl(HDGPTL** hDGPTl, **UINT32** cSlaveAddr, **UINT32** cDat, **bool** fUsedPEC, **int** iSlaveRespSet)**

功能

加入 Send Byte 封包。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address。

cDat[in]:

Type: **UINT32**

设置数据。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

DLLEXDGADDR PMBus_AppendReceiveBytePtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, bool fUsedPEC, int iSlaveRespSet)

功能

加入 Receive Byte packet 封包。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC)。

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

DLLEXDGADDR PMBus_AppendWriteBytePtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cCmd, UINT32 cDat, bool fUsedPEC, int iSlaveRespSet)

功能

加入 Write Byte 封包。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address。

cCmd[in]:

Type: **UINT32**

设置 Command。

cDat[in]:

Type: **UINT32**

设置数据。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

DLLEXDGADDR PMBus_AppendReadBytePtl(HDGPTL** hDGPTl, **UINT32** cSlaveAddr, **UINT32** cCmd, **bool** fUsedPEC, **int** iSlaveRespSet)**

功能

加入 Read Byte 封包。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address。

cCmd[in]:

Type: **UINT32**

设置 Command。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

DLLEXDGADDR PMBus_AppendWriteWordPtl(HDGPtI hDGpTl, **UINT32 cSlaveAddr, **UINT32** cCmd, **UINT32** wDat, **bool** fUsedPEC, **int** iSlaveRespSet)**

功能

加入 Write Word 封包。

参数

hDGpTl[in]:

Type: **HDGPtI**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address。

cCmd[in]:

Type: **UINT32**

设置 Command。

wDat[in]:

Type: **UINT32**

设置数据。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC)。

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

DLLEXDGADDR PMBus_AppendReadWordPtl(HDGPtI hDGpTl, **UINT32 cSlaveAddr, **UINT32** cCmd, **bool** fUsedPEC, **int** iSlaveRespSet)**

功能

加入 Read Word 封包。

参数

hDGpTl[in]:

Type: **HDGPtI**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address。

cCmd[in]:

Type: **UINT32**

设置 Command。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC)。

iSlaveResp[in]:

Type: **int**

设置 slave 回应状态 Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

DLLEXDGADDR PMBus_AppendWrite32Ptl(HDGPTL** hDGPTl, **UINT32** cSlaveAddr, **UINT32** cCmd, **UINT32** dwDat, **bool** fUsedPEC, **int** iSlaveRespSet)**

功能

加入 Write 32 封包。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address。

cCmd[in]:

Type: **UINT32**

设置 Command。

dwDat[in]:

Type: **UINT32**

设置数据。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC)。

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

DLLEXDGADDR PMBus_AppendRead32Ptl(HDGPtI hDGPtI, UINT32 cSlaveAddr, UINT32 cCmd, bool fUsedPEC, int iSlaveRespSet)

功能

加入 Read 32 封包。

参数

hDGPtI[in]:

Type: **HDGPtI**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address。

cCmd[in]:

Type: **UINT32**

设置 Command。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC)。

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

DLLEXDGADDR PMBus_AppendBlockWrPtl(HDGPtI hDGPtI, UINT32 cSlaveAddr, UINT32 cCmd, UINT32 cByteCnt, UINT32* pDatBuf, bool fUsedPEC, int iSlaveRespSet)

功能

加入 Block Write 封包。

参数

hDGPtI[in]:

Type: **HDGPtI**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address。

cCmd[in]:

Type: **UINT32**

设置 Command。

cByteCnt[in]:

Type: **UINT32**

设置 Byte Count。

pDatBuf[in]:

Type: **UINT32***

设置数据 buffer。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC)。

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。

DLLEXDGADDR PMBus_AppendBlockRdPtl(HDGPTL** hDGPTl, **UINT32** cSlaveAddr, **UINT32** cCmd, **UINT32** cByteCnt, **bool** fUsedPEC, **int** iSlaveRespSet)**

功能

加入 Block Read 封包。

参数

hDGPTl[in]:

Type: **HDGPTL**

PMBus 封包序列 handle。

cSlaveAddr[in]:

Type: **UINT32**

设置 Slave address。

cCmd[in]:

Type: **UINT32**

设置 Command。

cByteCnt[in]:

Type: **UINT32**

设置 Byte Count。

fUsedPEC[in]:

Type: **bool**

使用 Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

设置slave 回应状态Hi-Z (SET_HIZ) 或 High (NOT_SET_HIZ)。

回传值

回传 DG address，代表模式设置成功。如果回传 -1 则代表设置失败。