

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

For Data Generator 3000 and TravelData 3000

Version: 1.1

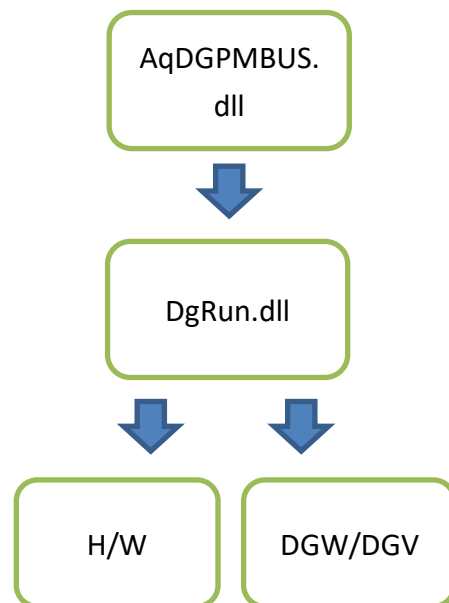
Publish: 2019/12/11

Contents

SDK Control Flow and simple introduction.....	3
SDK Function Definitions.....	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).....	6
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 Control Flow and simple introduction



This SDK provides an open interface for users to generate the PMBUS pattern; there are 2 ways to output the PMBUS pattern:

1. Output PMBUS pattern directly in a packet by packet way.
2. Output the PMBUS pattern directly in all the packets way.

SDK Function Definitions

```

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

bool InitProtocol(int iDGModel, bool fConnect)

Search and initialize the DG device connected on the computer.

Parameters

iDGModel[in]:

Type: **int**

Select the DG hardware model, enumerating these models as the following

```
enum DG_HW_MODEL
```

```
{
```

```
    DG3064B = 0x33064,
```

```
    DG3096B = 0x33096,
```

```
    DG3128B = 0x33128,
```

```
TD3008E = 0x23008,
TD3116B = 0x23116,
TD3216B = 0x23216,
};

fConnect[in]:
Type: bool
Set the connected mode, false is demo mode.
```

Return value

Return true if the function succeeded; false if the function failed.

Remarks

```
InitProtocol(TD3216B, false);
// Select the TD3216B model and demo mode
```

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

Initialize PMBus protocol parameters.

Parameters

iProtocolClockFreqInKHz[in]:

Type: **UINT32**
Set the PMBus frequency, unit: KHz.

iDGInitState[in]:

Type: **UINT32**
Set the beginning state of PMBus pattern, there are 3 states can be select:
DGINIT_STATE_LOW = 0,
DGINIT_STATE_HIGH = 1,
DGINIT_STATE_HI_Z = 2,

fSingleStep[in]:

Type: **bool**
Select **true**, DG will generate the PMBus pattern in a packet by packet way; select **false**, it will generate the PMBus pattern in all the packets way.

iSclNo, iSdaNo[in]:

Type: **int**
Set DG channel number for PMBus pin.

iFormat[in]:

Type: **int**
#define DGFMT_DGW 0x0001

```
#define DGFMT_DGV 0x0002
```

Select *.DGW or *.DGV file format.

Return value

Return the handle if the initial succeeded or -1 when failed.

Remarks

```
HDGPTL hDG = PMBus_Init (100, DGINIT_STATE_HIGH, true, 0, 1, DGFMT_DGW, true);  
// Set 100 KHz frequency for PMBus pattern, high state for the beginning of PMBus,  
// selecting the single step way to output PMBus pattern packet by packet,  
// set the DG CH-00 & CH-01 as PMBUS Scl & Sda pin, DGW file format, output repeatedly
```

bool CloseProtocol(HDGPTL hDGPTl)

Release the resource.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

Return value

Return true if the function succeeded; false if the function failed.

bool ClearProtocolPacket(HDGPTL hDGPTl)

Clear the protocol packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

Return value

Return true if the function succeeded; false if the function failed.

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

Save the protocol to the file.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

fFile[in]:

Type: **bool**

Save the protocol to the file(true) or to the buffer(false).

pPtr[in]:

Type: **char***

The file path when fFile = true or pointer of the buffer when fFile = false.

The function will return the buffer size when pPtr == NULL.

Return value

Return file/buffer size if the function succeeded; -1 if the function failed.

Remarks

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

// Save PMBus protocol to the file "PMBus.dgw"

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

Append the DG instruction to the PMBUS packet link list.

Parameters

hDGPtr[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

iInst[in]:

Type: **int**

The DG instructions, e.g. JP, LC, LP ... and so on.

#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]:

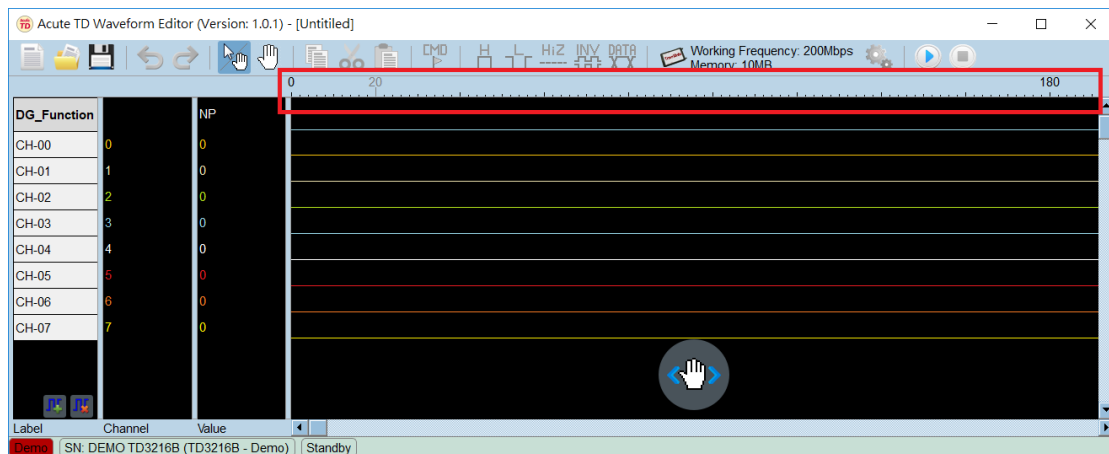
Type: **int**

The DG instruction parameter, e.g. JP 10, iParam = 10.

iDGAddr[in]:

Type: **DGADDR**

The address of DG, referring to the red area of the following snapshot.



Return value

Return true if the function succeeded; false if the function failed.

Remarks

AppendDGInstruction(hDG, CMD_JP, 0, 1000);

// Set DG instruction: Jump to New Address 0 at the DG address = 1000

int GetLastDGError()

Get Last Error.

Return value

Return the error code if the function failed, 0 if the function succeeded.

Error code

#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()

Get the numbers of pod.

Return value

Return 2 for model TD3216B, 6 for DG3064B.

bool SetOutputVolt(int imV, int iPodIndex)

Set the voltage for each pod.

Parameters

imV[in]:

Type: **int**

The voltage for each pod, unit: mV.

iPodIndex[in]:

Type: **int**

Zero-based index for pod.

Return value

Return true if the function succeeded; false if the function failed.

Remarks

```
SetOutputVolt(3000, 0);
```

```
// Set the Pod 0, 3V, Pod 0: CH0 ~ CH7
```

```
SetOutputVolt(3000, 1);
```

```
// Set the Pod 1, 3V, Pod 1: CH8 ~ CH15
```

bool OutputProtocol(HDGPTL hDGPTl)

Output the protocol.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

Return value

Return true if the function succeeded; false if the function failed.

int GetDGStatus()

Get the status of DG hardware.

Return value

Return DG_WAVEFORM_SENDING(0x80000000) status or other value for ready.

bool StopDG()

Stop the DG.

Return value

Return true if the function succeeded; false if the function failed.

bool ShutdownDG()

Shutdown the DG and disconnect DG with the computer.

Return value

Return true if the function succeeded; false if the function failed.

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

Get the protocol name & ID.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

pBuf[in]:

Type: **char***

The buffer to storage the protocol name.

iBufSize[in]:

Type: **int**

The buffer size.

Return value

Return the protocol ID.

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

Append the idle time to the PMBus packet link list.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

usecs[in]:

Type: **UINT32**

Idle time between the PMBus packets, unit: us

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

Remarks

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

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

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

Append group command protocol packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

gcp[in]:

Type: **GROUP_CMD_PTL***

The group command protocol packet 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**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

enum

```
{
    NOT_SET_HIZ = 0,
    SET_HIZ
};
```

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

Remarks

```
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)**

Append extended command read byte protocol packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

cExtCmd[in]:

Type: **UINT32**

Command extension code value.

cCmd[in]:

Type: **UINT32**

Command code value.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

Remarks

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)

Append extended command read byte protocol packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

cExtCmd[in]:

Type: **UINT32**

Command extension code value.

cCmd[in]:

Type: **UINT32**

Command code value.

cDat[in]:

Type: **UINT32**

Data byte value.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

Remarks

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)**

Append extended command read word protocol packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

cExtCmd[in]:

Type: **UINT32**

Command extension code value.

cCmd[in]:

Type: **UINT32**

Command code value.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

Remarks

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)

Append extended command write word protocol packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

cExtCmd[in]:

Type: **UINT32**

Command extension code value.

cCmd[in]:

Type: **UINT32**

Command code value.

wDat[in]:

Type: **UINT32**

Data word value.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

Remarks

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)

Append ZONE_READ with STATUS DATA response packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cCtlCmd[in]:

Type: **UINT32**

Command control code value.

cStatMask[in]:

Type: **UINT32**

Status mask value.

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

iPEC1[in]:

Type: **int**

PEC byte data, -1 if none PEC.

iPEC2[in]:

Type: **int**

PEC byte data, -1 if none PEC.

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

Remarks

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

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

Append ZONE_READ with STATUS DATA response packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cCtlCmd[in]:

Type: **UINT32**

Command control code value.

cPMBusCmd[in]:

Type: **UINT32**

PMBus command code value.

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

iPEC1[in]:

Type: **int**

PEC byte data, -1 if none PEC.

iPEC2[in]:

Type: **int**

PEC byte data, -1 if none PEC.

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

Remarks

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

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

Append ZONE_WRITE operation with PMBus command and two data bytes packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cPMBusCmd[in]:

Type: **UINT32**

PMBus command code value.

wDat[in]:

Type: **UINT32**

Data word value.

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

Remarks

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

DGADDR PMBus_AppendAlertRespAddr(HDGPTL hDGPTl, bool fUsedPEC, int

iSlaveRespSet)

Append Alert Response Address packet.

Parameters

hDGPtr[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

Remarks

PMBus_AppendAlertRespAddr(hDG, true, SET_HIZ);

DLLEXDGADDR PMBus_AppendQuickCmdPtr(HDGPTL** hDGPtr, **UINT32** cSlaveAddr, **UINT32** cRW, **int** iSlaveRespSet)**

Append Quick Command packet.

Parameters

hDGPtr[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

cRW[in]:

Type: **UINT32**

Wr: 0; Rd: 1.

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

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

Append Send Byte packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

cDat[in]:

Type: **UINT32**

Data byte value.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

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

Append Receive Byte packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

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

Append Write Byte packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

cCmd[in]:

Type: **UINT32**

Command code value.

cDat[in]:

Type: **UINT32**

Data byte value.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

DLLEXDGADDR PMBus_AppendReadBytePtl(HDGPTL** hDGPTl, **UINT32** cSlaveAddr,**

UINT32 cCmd, bool fUsedPEC, int iSlaveRespSet)

Append Read Byte packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

cCmd[in]:

Type: **UINT32**

Command code value.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

DLLEXDGADDR PMBus_AppendWriteWordPtl(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cCmd, UINT32 wDat, bool fUsedPEC, int iSlaveRespSet)

Append Write Word packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

cCmd[in]:

Type: **UINT32**

Command code value.

wDat[in]:

Type: **UINT32**

Data word value.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

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

Append Read Word packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

cCmd[in]:

Type: **UINT32**

Command code value.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

DLLEXDGADDR PMBus_AppendWrite32Ptl(HDGPTL** hDGPTl, **UINT32** cSlaveAddr,**

UINT32 cCmd, UINT32 dwDat, bool fUsedPEC, int iSlaveRespSet)

Append Write 32 packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

cCmd[in]:

Type: **UINT32**

Command code value.

dwDat[in]:

Type: **UINT32**

Double Data word value.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

DLLEXDGADDR PMBus_AppendRead32PtI(HDGPTL hDGPTl, UINT32 cSlaveAddr, UINT32 cCmd, bool fUsedPEC, int iSlaveRespSet)

Append Read 32 packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

cCmd[in]:

Type: **UINT32**

Command code value.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

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

Append Block Write packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

cCmd[in]:

Type: **UINT32**

Command code value.

cByteCnt[in]:

Type: **UINT32**

Data byte count.

pDatBuf[in]:

Type: **UINT32***

Block data buffer.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.

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

Append Block Read packet.

Parameters

hDGPTl[in]:

Type: **HDGPTL**

The handle for PMBus packet list.

cSlaveAddr[in]:

Type: **UINT32**

Slave address value.

cCmd[in]:

Type: **UINT32**

Command code value.

cByteCnt[in]:

Type: **UINT32**

Data byte count.

fUsedPEC[in]:

Type: **bool**

Use Packet Error Checking (PEC).

iSlaveResp[in]:

Type: **int**

Set the slave response as Hi-Z (SET_HIZ) or High (NOT_SET_HIZ).

Return value

Return the address of DG if the function succeeded; -1 if the function failed.