

# **Acute Data Generator – SPI/SIPI Protocol Software development kit (SDK) Programming guide**

For Data Generator 3000 and TravelData 3000

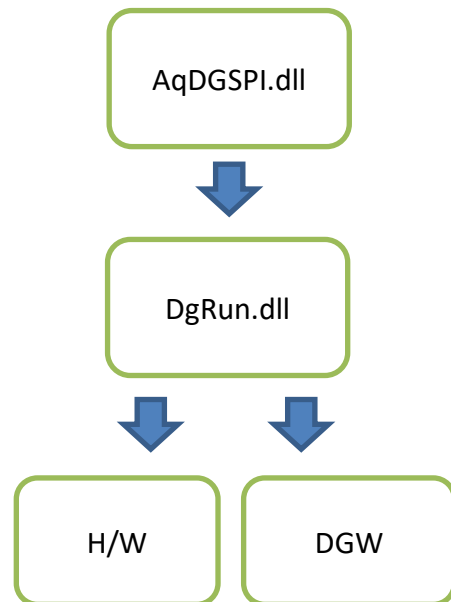
Version: 1.0

Publish: 2020/4/14

## 內容

|  |          |
|--|----------|
| <b>SDK 架構介紹</b> .....  | <b>3</b> |
| <b>SDK 函式說明</b> .....  | <b>3</b> |
| bool InitProtocol(int iDGModel, bool fConnect).....  | 3        |
| HDGPTL SPI_Init(UINT32 iProtocolClockFreqInKHz,UINT32 iDGInitState, bool fSingleStep, int*<br>piChBuf, int iFormat, int iProtocolType, int iWordSize, bool fRepeat)..... | 4        |
| bool CloseProtocol(HDGPTL hDGPTl).....   | 5        |
| 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().....   | 8        |
| bool SetOutputVolt(int imV, int iPodIndex).....  | 8        |
| bool OutputProtocol(HDGPTL hDGPTl).....  | 9        |
| int GetDGStatus().....   | 9        |
| bool StopDG().....   | 9        |
| bool ShutdownDG().....   | 9        |
| int GetProtocolName(HDGPTL hDGPTl, char* pBuf, int iBufSize).....  | 10       |
| DGADDR SPI_AppendIdle(HDGPTL hDGPTl, UINT32 nsecs).....  | 10       |
| DGADDR 4WireSPI_AppendPacket (HDGPTL hDGPTl, UINT64* pi64DatBuf, int iDatSize, int<br>iSlaveRespSet).....  | 11       |
| DGADDR 3WireSPI_AppendPacket(HDGPTL hDGPTl, UINT64* pi64DatBuf, int iDatSize, bool<br>fUseWrLatencyRd, int* piBitLengBuf, int iFrameGuardTime, int iSlaveRespSet).....   | 11       |
| DGADDR 2WireSPI_AppendPacket(HDGPTL hDGPTl, UINT64* pi64DatBuf, int iDatSize, bool<br>fUseWrLatencyRd, int* piBitLengBuf, int iFrameGuardTime, int iSlaveRespSet).....   | 12       |
| DGADDR SIPI_AppendPacket(HDGPTL hDGPTl, inti Type, int iClockNum, UINT64 i64Dat).....  | 13       |
| DGADDR SPI_InputFile(HDGPTL hDGPTl, char* pStrFile, UINT32 nsecs).....   | 14       |
| DGADDR SIPI_InputFile(HDGPTL hDGPTl, char* pStrFile).....  | 14       |

## SDK 架構介紹



此 SDK 提供一個開放介面讓使用者可以 2 種方式來發送 SPI/SIPI 波形。

1. 以一個一個 SPI/SIPI 封包的形式發送。
2. 一次發送所有 SPI/SIPI 封包的形式。

## SDK 函式說明

```
typedef unsigned int UINT32;  
typedef unsigned int HDGPTL;  
typedef unsigned int DGADDR;  
typedef __int64 I64;  
typedef unsigned __int64 UI64;
```

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

### 功能

尋找並啟動目前接在此電腦上的資料產生器。

### 參數

**iDGModel[in]:**

Type: **int**

選擇資料產生器機種，列舉如下：

enum DG\_HW\_MODEL

```
{  
    DG3064B = 0x33064,  
    DG3096B = 0x33096,  
    DG3128B = 0x33128,  
    TD3008E = 0x23008,  
    TD3116B = 0x23116,  
    TD3216B = 0x23216,  
};
```

**fConnect[in]:**

Type: **bool**

設定連接模式，false 表示 demo 模式。

回傳值

如果回傳值為 True，代表模式設定成功。如果回傳 False 值則代表設定失敗。

備註

```
InitProtocol(TD3216B, false);
```

```
// 選擇 TD3216B 機種並設定 demo 模式。
```

**HDGPTL SPI\_Init(UINT32 iProtocolClockFreqInKHz, UINT32 iDGInitState,  
bool fSingleStep, int\* piChBuf, int iFormat, int iProtocolType, int iWordSize,  
bool fRepeat)**

功能

初始化 SPI/SIPI 匯流排設定。

參數

**iProtocolClockFreqInKHz[in]:**

Type: **UINT32**

設定 SPI/SIPI 頻率，單位: KHz。

**iDGInitState[in]:**

Type: **UINT32**

設定 SPI/SIPI 匯流排起始狀態，共有下列 3 種:

DGINIT\_STATE\_LOW = 0,

DGINIT\_STATE\_HIGH = 1,

DGINIT\_STATE\_HI\_Z = 2,

**fSingleStep[in]:**

Type: **bool**

設定發送模式，選擇 true 時，資料產生器將以一個接一個封包的方式來發送

SPI/SIPI 封包，反之則是一次全部發送。

**piChBuf [in]:**

Type: **int\***

設定 SPI/SIPI 發送通道編號。

該通道沒有使用時，設為 -1。

piChBuf[0]: CS

piChBuf[1]: SCK; SIPI-CLK

piChBuf[2]: SDI(SDA); SIPI-DATA

piChBuf[3]: SDO

**iFormat[in]:**

Type: **int**

選擇發送的檔案格式。

#define DGFMT\_DGW 0x0001

**iProtocolType[in]:**

Type: **int**

0 = SPI

1 = SIPI

**iWordSize[in]:**

Type: **int**

4~40 bit

**fRepeat[in]:**

Type: **bool**

設定是否重複發送。

回傳值

回傳 handle 數值，代表模式設定成功。如果回傳 -1 則代表設定失敗。

備註

```
int iChNoBuf[] = {0, 1, 2, 3};
```

```
HDGPTL hDG = m_pSPIInit(1000, DGINIT_STATE_LOW, true, iChNoBuf, DGFMT_DGW,
```

```
    SPI_PROTOCOL, 8, false);
```

```
// 設定 1 MHz 頻率 SPI，波形初始階段為low，
```

```
// 選擇一個封包接著一個封包發送形式，
```

```
// 設定 CH-00 / CH-01 / CH-02 / CH-03為SPI CS / SCK / SDI / SDO通道，DGW 格式，
```

```
// 非重複發送。
```

**bool CloseProtocol(**HDGPTL** hDGptl)**

功能

釋放 SDK 所占用的資源。

#### 參數

**hDGPtr[in]:**

Type: **HDGPTL**

SPI/SIPI 封包序列 handle。

#### 回傳值

如果回傳值為 **True**，代表模式設定成功。如果回傳 **False** 值則代表設定失敗。

### **bool ClearProtocolPacket(**HDGPTL** hDGPtr)**

#### 功能

清除 SPI/SIPI 封包序列。

#### 參數

**hDGPtr[in]:**

Type: **HDGPTL**

SPI/SIPI 封包序列 handle。

#### 回傳值

如果回傳值為 **True**，代表模式設定成功。如果回傳 **False** 值則代表設定失敗。

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

#### 功能

將 SPI/SIPI 封包序列存檔。

#### 參數

**hDGPtr[in]:**

Type: **HDGPTL**

SPI/SIPI 封包序列 handle。

**fFile[in]:**

Type: **bool**

存入檔案(true) 或存入緩衝(false)。

**pPtr[in]:**

Type: **char\***

當 **fFile = true** 表示檔案路徑反之則為緩衝區指標。

#### 回傳值

回傳檔案或緩衝大小，代表模式設定成功。如果回傳 **-1** 則代表設定失敗。

#### 備註

**SaveProtocolList(hDG, true, "SPI.dgw");**

// 儲存 “SPI.dgw” 檔案

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

功能

加入資料產生器指令。

參數

**hDGPTl[in]:**

Type: **HDGPTL**

SPI/SIPI 封包序列 handle。

**iInst[in]:**

Type: **int**

設定 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]:**

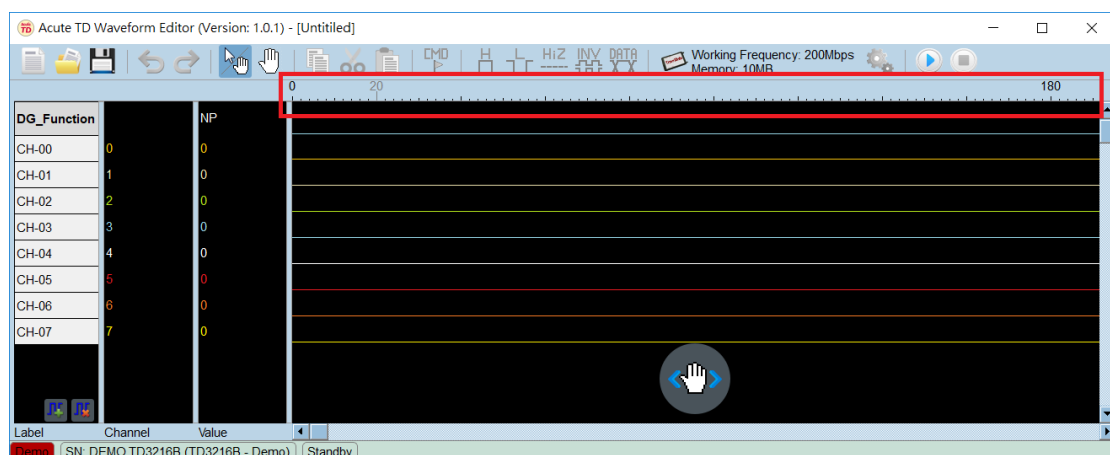
Type: **int**

設定 DG 指令參數, e.g. JP 10, iParam = 10.

**iDGAddr[in]:**

Type: **DGADDR**

設定欲插入指令位置，請參考下方擷圖。



回傳值

如果回傳值為 **True**，代表模式設定成功。如果回傳 **False** 值則代表設定失敗。

備註

AppendDGInstruction(hDG, CMD\_JP, 0, 1000);

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

## int GetLastDGError()

功能

取得錯誤代碼。

回傳錯誤碼或是 0 表示無錯誤。

錯誤碼

|                                      |        |
|--------------------------------------|--------|
| #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 數量。

回傳值

TD3216B = 2, DG3064B = 6。

## bool SetOutputVolt(int imV, int iPodIndex)

功能

設定輸出電壓。

參數

**imV[in]:**



Type: **int**

設定單一 pod 輸出電壓，單位: mV。

**iPodIndex[in]:**

Type: **int**

設定 pod 索引，從 0 開始。

回傳值

如果回傳值為 **True**，代表模式設定成功。如果回傳 **False** 值則代表設定失敗。

備註

```
SetOutputVolt(2500, 0);
```

```
// 設定 Pod 0, 2.5V, Pod 0: CH0 ~ CH7
```

```
SetOutputVolt(2500, 1);
```

```
// 設定 Pod 1, 2.5V, Pod 1: CH8 ~ CH15
```

**bool OutputProtocol(**HDGPTL** hDGPTl)**

功能

輸出 protocol。

參數

**hDGPTl[in]:**

Type: **HDGPTL**

SPI/SIPI 封包序列 handle。

回傳值

如果回傳值為 **True**，代表模式設定成功。如果回傳 **False** 值則代表設定失敗。

**int GetDGStatus()**

功能

取得資料產生器目前狀態。

回傳值

回傳 **DG\_WAVEFORM\_SENDING(0x80000000)** 表示 DG 在發送狀態，其他數值則為準備狀態。

**bool StopDG()**

功能

停止發送。

回傳值

如果回傳值為 **True**，代表模式設定成功。如果回傳 **False** 值則代表設定失敗。

### **bool ShutdownDG()**

功能

關閉資料產生器定中斷與電腦的 USB 連接。

回傳值

如果回傳值為 **True**，代表模式設定成功。如果回傳 **False** 值則代表設定失敗。

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

功能

取得目前匯流排名稱與 ID。

參數

**hDGPTl[in]:**

Type: **HDGPTL**

SPI/SIPI 封包序列 handle。

**pBuf[in]:**

Type: **char\***

設定匯流排名稱緩衝區。

**iBufSize[in]:**

Type: **int**

設定緩衝區大小。

回傳值

回傳匯流排 ID。

### **DGADDR SPI\_AppendIdle(HDGPTL hDGPTl, UINT32 nsecs)**

功能

加入 SPI/SIPI Idle。

參數

**hDGPTl[in]:**

Type: **HDGPTL**

SPI/SIPI 封包序列 handle。

**nsecs[in]:**

Type: **UINT32**

設定 Idle 時間，單位: ns。

回傳值

回傳 DG address，代表模式設定成功。如果回傳 -1 則代表設定失敗。

備註

```
SPI_AppendIdle(hDGPTl, 1000000);  
// set the idle time 1 ms
```

**DGADDR 4WireSPI\_AppendPacket (HDGPTL hDGPTl, UINT64\* pi64DatBuf, int iDatSize, int iSlaveRespSet)**

功能

加入 4-Wire SPI 封包。

參數

**hDGPTl[in]:**

Type: **HDGPTL**

SPI 封包序列 handle。

**pi64DatBuf[in]:**

Type: **UINT64\***

SPI 數據 buffer。

**iDatSize[in]:**

Type: **int**

設定 SPI 數據寬度: 8 ~ 40 bit。

**iSlaveResp[in]:**

Type: **int**

設定 slave 回應狀態 Hi-Z (SET\_HIZ) 或 High (NOT\_SET\_HIZ)。

回傳值

回傳 DG address，代表模式設定成功。如果回傳 -1 則代表設定失敗。

備註

```
I64 i64DatBuf[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};  
4WireSPI_AppendPacket(hDGPTl, i64DatBuf, 8, SET_HIZ);  
// 加入 4-Wire SPI 封包  
// 設定 SDO 為 Hi-Z 狀態。
```

**DGADDR 3WireSPI\_AppendPacket(HDGPTL hDGPTl, UINT64\* pi64DatBuf, int iDatSize, bool fUseWrLatencyRd, int\* piBitLengBuf, int iFrameGuardTime, int iSlaveRespSet)**

功能

加入 3-Wire SPI 封包。

參數

**hDGPI[in]:**

Type: **HDGPTL**

SPI 封包序列 handle。

**pi64DatBuf[in]:**

Type: **UINT64\***

SPI 數據 buffer。

**iDatSize[in]:**

Type: **int**

設定 SPI 數據寬度: 8 ~ 40 bit。

**fUseWrLatencyRd[in]:**

Type: **bool**

設定 SDI(Write)-Latency-SDO(Read)。

**piBitLengBuf[in]:**

Type: **int**

Write/Read/Latency 長度 (Bits)

piBitLengBuf[0]: Write Length Bits

piBitLengBuf[1]: Read Length Bits

piBitLengBuf[2]: Latency Length Bits

**iFrameGuardTime[in]:**

Type: **int**

Frame guard time, 單位: ns

**iSlaveResp[in]:**

Type: **int**

設定 slave 回應狀態 Hi-Z (SET\_HIZ) 或 High (NOT\_SET\_HIZ)。

回傳值

回傳 DG address，代表模式設定成功。如果回傳 -1 則代表設定失敗。

**DGADDR 2WireSPI\_AppendPacket (HDGPTL hDGPI, UINT64\* pi64DatBuf, int iDatSize, bool fUseWrLatencyRd, int\* piBitLengBuf, int iFrameGuardTime, int iSlaveRespSet)**

功能

加入 2-Wire SPI 封包。

參數

**hDGPI[in]:**

Type: **HDGPTL**

SPI 封包序列 handle。

**pi64DatBuf[in]:**

Type: **UINT64\***

SPI 數據 buffer。

**iDatSize[in]:**

Type: **int**

設定 SPI 數據寬度: 8 ~ 40 bit。

**fUseWrLatencyRd[in]:**

Type: **bool**

設定 SDI(Write)-Latency-SDO(Read)。

**piBitLengBuf[in]:**

Type: **int**

Write/Read/Latency 長度 (Bits)

piBitLengBuf[0]: Write Length Bits

piBitLengBuf[1]: Read Length Bits

piBitLengBuf[2]: Latency Length Bits

**iFrameGuardTime[in]:**

Type: **int**

Frame guard time, 單位: ns

**iSlaveResp[in]:**

Type: **int**

設定 slave 回應狀態 Hi-Z (SET\_HIZ) 或 High (NOT\_SET\_HIZ)。

回傳值

回傳 DG address，代表模式設定成功。如果回傳 -1 則代表設定失敗。

**DGADDR SIPI\_AppendPacket (HDGPTL hDGPTl, int iType, int iClockNum, UINT64 i64Dat)**

功能

加入 SIPI 封包。

參數

**hDGPTl[in]:**

Type: **HDGPTL**

SIPI 封包序列 handle。

**iClockNum[in]:**

Type: **int**

設定 clock 數量。

**i64Dat[in]:**

Type: **UINT64**

設定 SIPI 數據。

回傳值

回傳 DG address，代表模式設定成功。如果回傳 -1 則代表設定失敗。

**DGADDR SPI\_InputFile(**HDGPTL** hDGPTl, **char\*** pStrFile, **UINT32** nsecs)**

功能

載入 \*.txt 或 \*.bin 檔案。

參數

**hDGPTl[in]:**

Type: **HDGPTL**

SPI 封包序列 handle。

**pStrFile[in]:**

Type: **char\***

設定 \*.txt or \*.bin 檔案路徑。

**nsecs[in]:**

Type: **UINT32**

設定 Idle, 單位: ns

回傳值

回傳 DG address，代表模式設定成功。如果回傳 -1 則代表設定失敗。

**DGADDR SIPI\_InputFile(**HDGPTL** hDGPTl, **char\*** pStrFile)**

功能

載入 SIPI 文字檔。

**Parameters**

**hDGPTl[in]:**

Type: **HDGPTL**

SIPI 封包序列 handle。

**pStrFile[in]:**

Type: **char\***

設定 SIPI 文字檔案路徑。

回傳值

回傳 DG address，代表模式設定成功。如果回傳 -1 則代表設定失敗。