AWG-10K2 プログラミングガイド

Rev 1.0

株式会社エルモス

目次

1　はじめに

1.1 概要

製品付属の波形編集ソフトウェアやユーザーが作成した波形データをユーザーアプリケーションから「AWG-10K2 API 関数」を介して本体に転送することができます。
本体の動作モードが「リモート（PC）」時は、波形出力制御（スタート／ストップ）ができます。
また、汎用 I/O ポートを制御することもできます。
動作モード、出力パラメータ設定の詳細は**「AWG-10K2 取扱説明書」**をご参照ください。

1.2 動作環境

対応 OS	Windows 10／8（8.1）／7／Vista／XP
対応開発言語	Visual C++／Visual Basic.NET／Visual C#.NET
必要メモリ空き容量	10M バイト以上

1.3 関数について

「AWG-10K2 API 関数」は関数群をモジュール化した「AWG10K2.dll」で提供されます。
「AWG10K2.dll」ファイルは **AWG-10K2** をインストールする時にシステムフォルダに入ります。

関数は**デバイス関数**、**制御関数**に分類されます。
デバイス関数はデバイスの検索、デバイスとの接続、切断等の関数です。
制御関数は波形データ転送、波形出力、汎用 I/O 操作等の関数です。

1.4 プログラミングの準備

「AWG10K2.dll」ファイルをシステムフォルダまたは実行フォルダにコピーしてください。

・Visual C++の場合
「AWG10K2.h」、「AWG10K2.lib」ファイルをプロジェクトに追加してください。

・Visual Basic.NET の場合
「AWG10K2.vb」ファイルをプロジェクトに追加してください。

・Visual C#.NET の場合
「AWG10K2.cs」ファイルをプロジェクトに追加してください。

※これらのファイルは製品に付属の CD-ROM「¥library」フォルダにあります。

1.5 関数の戻り値について

関数の戻り値の説明を下表に示します。

定数	値	意味
AWG10K2_OK	0	正常終了
AWG10K2_INVALID_ID	1	デバイスの ID が無効
AWG10K2_DEVICE_NOT_FOUND	2	デバイスが見つからない
AWG10K2_DEVICE_NOT_OPENED	3	デバイスがオープンできない
AWG10K2_MEMORY_ERROR	4	メモリが不足している
AWG10K2_INVALID_PARAMETER	5	パラメータが無効
AWG10K2_COMMUNICATION_ERROR	6	通信エラーが発生した
AWG10K2_EXECUTION_ERROR	7	実行エラーが発生した
AWG10K2_OTHER_ERROR	8	その他のエラーが発生した

表 1.5 関数の戻り値

1.6 注意事項

複数のアプリケーション（製品付属の波形編集ソフトウェア含）、またはマルチスレッドによる **AWG-10K2** への同時アクセスはできませんので注意してください。

2　関数リファレンス

2.1 デバイス関数

AWG10K2_GetNumberOfDevices

現在接続されているデバイスの数を取得します。

int **AWG10K2_GetNumberOfDevices** (int **Number*)

Parameters

Number　　接続デバイス数の格納先へのポインタ

Remarks

デバイスが接続されていない場合は *Number* には 0 が格納されます。

Example

```
int result;
int number;

result = AWG10K2_GetNumberOfDevices(&number);
if (result == AWG10K2_OK) {
	// AWG10K2_GetNumberOfDevices 成功
}
else {
	// AWG10K2_GetNumberOfDevices 失敗
}
```

AWG10K2_GetSerialNumber

現在接続されているデバイスのシリアル番号を取得します。

int **AWG10K2_GetSerialNumber** (int *Index*, int **SerialNumber*)

Parameters

Index 0 から始まる接続デバイスのインデックス

SerialNumber シリアル番号の格納先へのポインタ

Remarks

取得したシリアル番号を引数として **AWG10K2_OpenBySerialNumber** でオープンすることができます。尚、このシリアル番号は本体記載のシリアル番号と同一です。

Example

```
int result;
int index;
int serialNumber;

index = 0;
result = AWG10K2_GetSerialNumber(index, &serialNumber);
if (result == AWG10K2_OK) {
	// AWG10K2_GetSerialNumber 成功
}
else {
	// AWG10K2_GetSerialNumber 失敗
}
```

AWG10K2_Open

デバイスをオープンし、デバイスの ID を取得します。

int **AWG10K2_Open** (BYTE **ID*)

Parameters

ID　　デバイスの ID の格納先へのポインタ

Remarks

デバイスが複数接続されている場合は接続できた最初のデバイスの ID を取得します。
取得した ID は以後、制御関数等に引数として渡すことになります。

Example

```
int result;
BYTE id;

result = AWG10K2_Open(&id);
if (result == AWG10K2_OK) {
	// AWG10K2_Open 成功
}
else {
	// AWG10K2_Open 失敗
}
```

AWG10K2_OpenBySerialNumber

指定したシリアル番号と一致するデバイスをオープンし、デバイスの ID を取得します。

int **AWG10K2_OpenBySerialNumber** (int *SerialNumber*, BYTE **ID*)

Parameters

SerialNumber　デバイスのシリアル番号

ID　デバイスの ID の格納先へのポインタ

Remarks

シリアル番号は **AWG10K2_GetSerialNumber** で取得することができます。

尚、このシリアル番号は本体記載のシリアル番号と同一です。

取得した ID は以後、制御関数等に引数として渡すことになります。

Example

```
int result;
BYTE id;
int serialNumber;     // 取得したデバイスのシリアル番号

result = AWG10K2_OpenBySerialNumber(serialNumber, &id);
if (result == AWG10K2_OK) {
       // AWG10K2_OpenBySerialNumber 成功
}
else {
       // AWG10K2_OpenBySerialNumber 失敗
}
```

AWG10K2_Close

デバイスをクローズします。

int **AWG10K2_Close** (BYTE *ID*)

Parameters

ID　　デバイスの ID

Example

```
int result;
BYTE id;       // オープンしたデバイスの ID

result = AWG10K2_Close(id);
if (result == AWG10K2_OK) {
       // AWG10K2_Close 成功
}
else {
       // AWG10K2_Close 失敗
}
```

2.2 制御関数

AWG10K2_SetFile

波形ファイルを転送します。

int **AWG10K2_SetFile** (BYTE *ID*, const char **FilePath*)

Parameters

ID	デバイスの ID
FilePath	ファイルパスを保持している文字列へのポインタ

Remarks

ファイルパスは終端が NULL で終わる絶対パス(フルパス)をマルチバイト文字列で指定してください。
尚、波形編集ソフトウェアで作成したファイル(拡張子*.twg)以外は指定できません。
波形ファイルの転送は波形出力中にはできません。波形出力停止時に転送してください。

Example

```
int result;
BYTE id;       // オープンしたデバイスの ID
char filePath[] = "C:¥¥temp¥¥test.twg";

result = AWG10K2_SetFile(id, filePath);
if (result == AWG10K2_OK) {
       // AWG10K2_SetFile 成功
}
else {
       // AWG10K2_SetFile 失敗
}
```

AWG10K2_SetParametersAndWaveData

パラメータと波形データを転送します。

int **AWG10K2_SetParametersAndWaveData** (BYTE *ID*,
BYTE *ControlMode*, BYTE *OutputMode*,
int *RepeatCount*, BYTE *TriggerType*,
int *SamplingRate*, int *SamplingCount*,
double **WaveData*)

Parameters

ID	デバイスの ID
ControlMode	制御モード
OutputMode	出力モード
RepeatCount	繰り返し回数
TriggerType	トリガータイプ
SamplingRate	サンプリングレート
SamplingCount	サンプリング数
AnalogData	波形データ配列へのポインタ

Remarks

パラメータと波形データを **AWG-10K2** 本体に転送します。
転送は波形出力中にはできません。波形出力停止時に転送してください。

ControlMode に制御モードを設定します。

ControlMode	説明
0	リモート(PC)
1	外部トリガー

①リモート(PC)

「リモート(PC)」に設定すると、**AWG10K2_Start**、**AWG10K2_Stop** で信号出力の開始・停止が行えます。

②外部トリガー

「外部トリガー」に設定すると、本体に入力する外部トリガー信号による信号出力の開始が可能になります。後述のトリガータイプで立上り(正)エッジか立下り(負)エッジが設定できます。

OutputMode に出力モードを設定します。

OutputMode	説明
0	連続
1	繰り返し

①連続出力

信号出力が開始されると、停止されるまで波形データ（1周期信号）が連続的に出力されます。
「連続」に設定した場合、繰り返し回数は無効です。

②繰り返し出力

設定された繰り返し回数分の波形データ（1周期信号）が繰り返し出力されます。
繰り返し回数は *RepeatCount* に設定します。範囲は 1～256 です。

TriggerType にトリガータイプを設定します。
制御モードを「リモート（PC）」に設定した場合は無効になります。

TriggerType	説明
0	立上りエッジ
1	立下りエッジ

SamplingRate、*SamplingCount* にサンプリング設定をします。
SamplingRate×100 がサンプリング周期（nSec）となります。範囲は 1～4194304 です。
SamplingCount に波形データのサンプリング数を設定します。範囲は 1～262144 です。

WaveData に波形データが格納された配列のポインタを指定します。
波形データの単位は[V]です。

Example

```
int result;
BYTE id;       // オープンしたデバイスの ID
BYTE controlMode, outputMode, triggerType;
int repeatCount, samplingRate, samplingCount;
double waveData[10000];      // 波形データが格納されたバッファ

controlMode = 1;      // 外部トリガー
outputMode = 1;       // 繰り返し
repeatCount = 100;    // 100 回
triggerType = 0;      // 立上りエッジ
samplingRate = 10;    // 1uSec
samplingCount = 10000;

result = AWG10K2_SetParametersAndWaveData(id, controlMode, outputMode,
              repeatCount, triggerType, samplingRate, samplingCount, waveData);
if (result == AWG10K2_OK) {
       // AWG10K2_SetParametersAndWaveData 成功
}
else {
       // AWG10K2_SetParametersAndWaveData 失敗
}
```

AWG10K2_Start

波形出力をスタートします。

int **AWG10K2_Start** (BYTE *ID*)

Parameters

ID　　デバイスの ID

Remarks

波形データが本体に設定されていない(ファイル転送していない)場合、波形出力は開始できません。
本体の動作モードが「外部トリガー」に設定されている場合、本関数では波形出力は開始できません。
本体の動作モードが「リモート(PC)」の時のみ有効です。
「外部トリガー」時の波形出力制御については**「AWG-10K2 取扱説明書」**をご参照ください。

Example

```
int result;
BYTE id;       // オープンしたデバイスの ID

result = AWG10K2_Start(id);
if (result == AWG10K2_OK) {
       // AWG10K2_Start 成功
}
else {
       // AWG10K2_Start 失敗
}
```

AWG10K2_Stop

波形出力をストップします。

int **AWG10K2_Stop** (BYTE *ID*)

Parameters

ID　　デバイスの ID

Remarks

本体の動作モードが「外部トリガー」に設定されている場合、本関数では波形出力は停止できません。
本体の動作モードが「リモート(PC)」の時のみ有効です。
「外部トリガー」時の波形出力制御については**「AWG-10K2 取扱説明書」**をご参照ください。

Example

```
int result;
BYTE id;       // オープンしたデバイスの ID

result = AWG10K2_Stop(id);
if (result == AWG10K2_OK) {
       // AWG10K2_Stop 成功
}
else {
       // AWG10K2_Stop 失敗
}
```

AWG10K2_GetStatus

動作ステータスを取得します。

int **AWG10K2_GetStatus** (BYTE *ID*, BYTE **Status*)

Parameters

ID	デバイスの ID
Status	動作ステータスの格納先へのポインタ

Remarks

本関数が成功すると *Status* には動作ステータスが下記の値で格納されます。

定数	値	意味
AWG10K2_STATUS_NO_DATA	0	波形出力停止中(波形データ未設定)
AWG10K2_STATUS_READY	1	波形出力停止中(波形データ設定済)
AWG10K2_STATUS_RUNNING	2	波形出力中

Example

```
int result;
BYTE id;       // オープンしたデバイスの ID
BYTE status;

result = AWG10K2_GetStatus(id, &status);
if (result == AWG10K2_OK) {
       // AWG10K2_GetStatus 成功
}
else {
       // AWG10K2_GetStatus 失敗
}
```

AWG10K2_OutputPort

汎用出力端子に出力します。

int **AWG10K2_OutputPort** (BYTE *ID*, BYTE *Port*)

Parameters

ID デバイスの ID

Port 汎用出力端子に出力する値

Remarks

本関数が成功すると汎用出力端子に出力されます。

値	状態
0	汎用出力端子が Low レベル
1	汎用出力端子が High レベル

Example

```
int result;
BYTE id;        // オープンしたデバイスの ID
BYTE port;

port = 1;       // High レベル出力
result = AWG10K2_OutputPort(id, port);
if (result == AWG10K2_OK) {
        // AWG10K2_OutputPort 成功
}
else {
        // AWG10K2_OutputPort 失敗
}
```

AWG10K2_InputPort

汎用入力端子の状態を取得します。

int **AWG10K2_InputPort** (BYTE *ID*, BYTE **Port*)

Parameters

ID	デバイスの ID
Port	汎用入力端子の状態の格納先へのポインタ

Remarks

本関数が成功すると *Port* には汎用入力端子の状態が下記の値が格納されます。

値	状態
0	汎用入力端子が Low レベル
1	汎用入力端子が High レベル

Example

```
int result;
BYTE id;        // オープンしたデバイスの ID
BYTE port;

result = AWG10K2_InputPort(id, &port);
if (result == AWG10K2_OK) {
        // AWG10K2_InputPort 成功
}
else {
        // AWG10K2_InputPort 失敗
}
```