

SW-PLC ライブラリ

API リファレンス

株式会社セイワ技研
2025 年 12 月 9 日

目次

ライブラリ仕様	2
SW-PLC ライブラリ	2
アドインライブラリ	2
サポート OS	3
サポート言語	3
マルチスレッド対応	3
ライブラリを使用する準備	4
Visual C++	4
Visual C	4
Visual Basic .NET	4
関数一覧	5
SwPlcLoadFile	8
SwPlcLoadFileEx	9
SwPlcInitPort	11
SwPlcLoadTag	18
SwPlcClosePort	19
SwPlcComStatus	20
SwPlcSetNotify	21
SwPlcExecute	23
SwPlcResponse	24
SwPlcClear	25
SwPlcAbort	26
SwPlcGetPlcErrorCode	27
SwPlcRead	28
SwPlcWrite	29
SwPlcWriteArray	30
SwPlcGet	31
SwPlcGetArray	32
SwPlcSetMonitorInterval	33
SwPlcMonitorStatus	34
SwPlcGetMonitorPlcErrorCode	35
SwPlcWriteLog	36
エラーコード一覧	37
サンプルプログラム	39
改訂履歴	42

ライブラリ仕様

PLC ライブラリは Windows ネイティブ DLL で提供するライブラリです。下記の環境で使用可能です。

SW-PLC ライブラリ

ファイル名	内容
SwPlcCom. dll	PLC 通信メイン DLL

アドインライブラリ

ファイル名	内容	対応プロトコル
McCom. dll	三菱 PLC 用アドイン DLL	MC プロトコル 1E/3E/4E/1C/3C/4C
OmronCom. dll	オムロン PLC 用アドイン DLL	C モード通信 FINS/TCP、FINS/UDP
KvCom. dll	キーエンス PLC 用アドイン DLL	上位リンク通信 TCP, UDP, シリアル
ModbusCom. dll	Modbus 通信用アドイン DLL	Modbus/ASCII、Modbus/RTU、 Modbus/TCP
ToyopucCom. dll	JTEKT TOYOPUC PLC 通信用アドイン DLL	コンピュータリンク通信
SwCom. dll	ダミーPLC 用アドイン DLL	専用プロトコル

PLC ライブラリ DLL は 32bit 版 DLL と 64bit 版 DLL を提供しております。

アプリケーションに必要な DLL を選択して配置してください。

サポート OS

- ・ Windows 7 (32bit/64bit)
- ・ Windows 8 (32bit/64bit)
- ・ Windows 8.1 (32bit/64bit)
- ・ Windows 10 (32bit/64bit)
- ・ Windows 11
- ・ Windows Server 2019 / 2022

サポート言語

- ・ Microsoft Visual C++
- ・ Microsoft Visual C#
- ・ Microsoft Visual Basic .NET
- ・ その他 Windows ネイティブ DLL を使用できる言語

マルチスレッド対応

PLC ライブラリの関数は全てスレッドセーフです。複数のスレッドから同時に関数を実行しても安全に実行されます。

ライブラリを使用する準備

PLC ライブラリの API を使用するには、言語ごとにヘッダファイル等を準備する必要があります。

Visual C++

下記のファイルをプロジェクトに追加して使用してください。

- ・ SwPlcCom.h
- ・ SwPlcCom.lib
- ・ SwPlcCom.dll
- ・ アドイン DLL ファイル

Visual C

下記のファイルをプロジェクトに追加して使用してください。

- ・ SwPlcComSdk.cs
- ・ SwPlcCom.dll
- ・ アドイン DLL ファイル

Visual Basic .NET

下記のファイルをプロジェクトに追加して使用してください。

- ・ SwPlcComSdk.vb
- ・ SwPlcCom.dll
- ・ アドイン DLL ファイル

※ アドイン DLL ファイルは通信対象の PLC に対応したファイルが必要です。

関数一覧

関数名	説明
SwPlcLoadFile	タグ定義ファイルからタグ定義と通信設定を読み込んで通信ポートを開きます。
SwPlcLoadFileEx	タグ定義ファイルからタグ定義のみ読み込んで通信ポートを開きます。
SwPlcInitPort	タグ定義と通信設定を引数で指定して通信ポートを開きます。
SwPlcLoadTag	タグ定義ファイルからタグ定義を読み込みます。
SwPlcClosePort	通信ポートを閉じます。
SwPlcComStatus	通信ポートの状態を取得します。
SwPlcSetNotify	コールバック関数を登録します。
SwPlcExecute	登録されたコマンドを実行します。
SwPlcResponse	実行中コマンドの応答を取得します。
SwPlcClear	キューに登録されたコマンドを全てクリアします。
SwPlcAbort	実行中のコマンドを中止します。
SwPlcGetPlcErrorCode	コマンド実行で PLC から受信した PLC のエラーコードを取得します。
SwPlcRead	データ読み込みコマンドをキューに追加します。
SwPlcWriteBit	ビット型データの書き込みコマンドをキューに追加します。
SwPlcWriteShort	符号付き 16 ビット整数型データの書き込みコマンドをキューに追加します。
SwPlcWriteUShort	符号無し 16 ビット整数型データの書き込みコマンドをキューに追加します。
SwPlcWriteInt	符号付き 32 ビット整数型データの書き込みコマンドをキューに追加します。
SwPlcWriteUInt	符号無し 32 ビット整数型データの書き込みコマンドをキューに追加します。
SwPlcWriteLong	符号付き 64 ビット整数型データの書き込みコマンドをキューに追加します。
SwPlcWriteULong	符号無し 64 ビット整数型データの書き込みコマンドをキューに追加します。
SwPlcWriteFloat	32 ビット浮動小数点数型データの書き込みコマンドをキューに追加します。
SwPlcWriteDouble	64 ビット浮動小数点数型データの書き込みコマンドをキューに追加します。

SwPlcWriteBitArray	ビット型配列データの書込みコマンドをキューに追加します。
SwPlcWriteShortArray	符号付き 16 ビット整数型配列データの書込みコマンドをキューに追加します。
SwPlcWriteUShortArray	符号無し 16 ビット整数型配列データの書込みコマンドをキューに追加します。
SwPlcWriteIntArray	符号付き 32 ビット整数型配列データの書込みコマンドをキューに追加します。
SwPlcWriteUIntArray	符号無し 32 ビット整数型配列データの書込みコマンドをキューに追加します。
SwPlcWriteLongArray	符号付き 64 ビット整数型配列データの書込みコマンドをキューに追加します。
SwPlcWriteULongArray	符号無し 64 ビット整数型配列データの書込みコマンドをキューに追加します。
SwPlcWriteFloatArray	32 ビット浮動小数点数型配列データの書込みコマンドをキューに追加します。
SwPlcWriteDoubleArray	64 ビット浮動小数点数型配列データの書込みコマンドをキューに追加します。
SwPlcGetBit	指定されたタグ ID からビット型データを取得します。
SwPlcGetShort	指定されたタグ ID から符号付き 16 ビット整数型データを取得します。
SwPlcGetUShort	指定されたタグ ID から符号無し 16 ビット整数型データを取得します。
SwPlcGetInt	指定されたタグ ID から符号付き 32 ビット整数型データを取得します。
SwPlcGetUInt	指定されたタグ ID から符号無し 32 ビット整数型データを取得します。
SwPlcGetLong	指定されたタグ ID から符号付き 64 ビット整数型データを取得します。
SwPlcGetULong	指定されたタグ ID から符号無し 64 ビット整数型データを取得します。
SwPlcGetFloat	指定されたタグ ID から 32 ビット浮動小数点数型データを取得します。
SwPlcGetDouble	指定されたタグ ID から 64 ビット浮動小数点数型データを取得します。
SwPlcGetBitArray	配列型タグのデータを取得します。
SwPlcGetShortArray	指定されたタグ ID から符号付き 16 ビット整数型配列データを取得します。
SwPlcGetUShortArray	指定されたタグ ID から符号無し 16 ビット整数型配列データを取得します。

SwPlcGetIntArray	指定されたタグ ID から符号付き 32 ビット整数型配列データを取得します。
SwPlcGetUIntArray	指定されたタグ ID から符号無し 32 ビット整数型配列データを取得します。
SwPlcGetLongArray	指定されたタグ ID から符号付き 64 ビット整数型配列データを取得します。
SwPlcGetULongArray	指定されたタグ ID から符号無し 64 ビット整数型配列データを取得します。
SwPlcGetFloatArray	指定されたタグ ID から 32 ビット浮動小数点数型配列データを取得します。
SwPlcGetDoubleArray	指定されたタグ ID から 64 ビット浮動小数点数型配列データを取得します。
SwPlcSetMonitorInterval	モニタ指定されたタグの読み込み周期を設定します。
SwPlcMonitorStatus	モニタの通信状態を取得します。
SwPlcGetMonitorPlcErrorCode	モニタ通信中に発生した最新の PLC エラーコードを取得します。
SwPlcWriteLog	通信ログファイルに文字列を出力します。

SwPlcLoadFile

```
int SwPlcLoadFile(  
    SWPLC_HANDLE* handle,  
    const char* fileName)
```

タグマネージャーで作成したタグ定義ファイルからタグ定義と通信設定を読み込んで通信ポートを開きます。関数の実行が成功すると handle に通信 ID が格納されます。

引数	説明
handle	通信 ID を取得する変数を指定します。
fileName	タグ定義ファイルのパスを指定します。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_ERROR	異常終了。作成した通信ポートが多すぎる場合に発生します。作成できる通信ポート数の上限は 300 です。
SWPLC_ERROR_LOADFILE	タグ定義ファイルを読み込みできませんでした。
SWPLC_ERROR_LOADLIBRARY	指定された通信アドインライブラリファイルが見つかりません。
SWPLC_ERROR_SHAREMEM	共有データメモリの初期化に失敗しました。すでに同じ名前の通信ポートがないか確認してください。
SWPLC_ERROR_TAG_DATACOUNT	タグのデータ数が不正です。データ数は 1 以上を設定してください。
SWPLC_ERROR_PARAM	通信パラメータが不正です。
SWPLC_ERROR_OPTIONPARAM	オプションパラメータが不正です。
SWPLC_ERROR_TAG_INVALIDTYPE	タグに無効なデータ型が設定されています。
SWPLC_ERROR_PORT	ポートのオープンに失敗しました。シリアル通信の場合、シリアル通信ポートのオープンに失敗。TCP または UDP 通信の場合、Socket ポートのオープンに失敗。

コード例

```
SWPLC_HANDLE handle;  
SwPlcLoadFile(&handle, "sample.xml");
```

SwPlcLoadFileEx

```
int SwPlcLoadFileEx(  
    SWPLC_HANDLE* handle,  
    const char* fileName,  
    const char* portName,  
    const char* libFileName,  
    int type,  
    const char* param,  
    const char* option)
```

タグマネージャーで作成したタグ定義ファイルからタグ定義のみ読み込んで通信ポートを開きます。関数の実行が成功すると handle に通信 ID が格納されます。

引数	説明
handle	通信 ID を取得する変数を指定します。
fileName	タグ定義ファイルのパスを指定します。
portName	通信ポート名を指定します。通信ポート名は英数文字で任意の名称を設定できますが、他の通信ポートと重複しない名称を設定してください。
libFileName	通信で使用するアドイン DLL 名を指定します。
type	通信タイプを指定します。SWPLC_TYPE_SERIAL: シリアル通信 SWPLC_TYPE_TCP: TCP 通信 SWPLC_TYPE_UDP: UDP 通信 SWPLC_TYPE_CUSTOM: カスタム通信
param	通信パラメータ文字列を指定します。パラメータ文字列は 1023 文字以内で設定してください。
option	通信オプション文字列を指定します。オプション文字列は 1023 文字以内で設定してください。

※param と option については SwPlcInitPort を参照してください。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_ERROR	異常終了。作成した通信ポートが多すぎる場合に発生します。作成できる通信ポート数の上限は 300 です。
SWPLC_ERROR_LOADFILE	タグ定義ファイルを読み込みできませんでした。
SWPLC_ERROR_LOADLIBRARY	指定された通信アドインライブラリファイルが見つかりません。
SWPLC_ERROR_SHAREMEM	共有データメモリの初期化に失敗しました。すでに同じ名前の通信ポートがないか確認してください。

SWPLC_ERROR_TAG_DATACOUNT	タグのデータ数が不正です。データ数は1以上を設定してください。
SWPLC_ERROR_PARAM	通信パラメータが不正です。
SWPLC_ERROR_OPTIONPARAM	オプションパラメータが不正です。
SWPLC_ERROR_TAG_INVALIDTYPE	タグに無効なデータ型が設定されています。
SWPLC_ERROR_PORT	ポートのオープンに失敗しました。シリアル通信の場合、シリアル通信ポートのオープンに失敗。TCP または UDP 通信の場合、Socket ポートのオープンに失敗。

コード例

```
const char* param = "Address=192.168.0.1;"
                  "Port=5023;"
                  "ConnectionTimeout=2000;"
                  "ConnectionRetryTime=2000;"
                  "Timeout=1000;"
                  "Retry=1;"
                  "LogFilePath=C:¥¥log;"
                  "LogFileName=PlcLogFile;"
                  "LogSize=100000;"
                  "LogNum=3";
```

```
const char* option = "Code=BINARY;"
                   "Frame=3E;"
                   "MonitoringTimer=0005;"
                   "NetworkNo=00;"
                   "PcNo=FF;"
                   "ReqDestModIONo=03FF;"
                   "ReqDestModStationNo=00";
```

```
SWPLC_HANDLE handle;
```

```
SwPlcLoadFileEx(&handle, "sample.xml", "port1", "McCom", SWPLC_TYPE_TCP, param, option);
```

SwPlcInitPort

```
int SwPlcInitPort(  
    SWPLC_HANDLE* handle,  
    const char* portName,  
    const char* libFileName,  
    int type,  
    SwPlcTag tags[],  
    int tagCount,  
    const char* param,  
    const char* option)
```

タグ定義と通信設定を引数で指定して通信ポートを開きます。関数の実行が成功すると handle に通信 ID が格納されます。タグ定義ファイルを使用せずに通信ポートを開く場合は本関数を使用します。

引数	説明
handle	通信 ID を取得する変数を指定します。
portName	通信ポート名を指定します。通信ポート名は英数文字で任意の名称を設定できますが、他の通信ポートと重複しない名称を設定してください。
libFileName	通信で使用するアドイン DLL 名を指定します。
type	通信タイプを指定します。
tags	タグ定義配列を指定します。
tagCount	タグ定義配列の配列数を指定します。
param	通信パラメータ文字列を指定します。パラメータ文字列は 1023 文字以内で設定してください。
option	通信オプション文字列を指定します。オプション文字列は 1023 文字以内で設定してください。

通信タイプ

タイプ	説明
SWPLC_TYPE_SERIAL	シリアル通信
SWPLC_TYPE_TCP	TCP 通信
SWPLC_TYPE_UDP	UDP 通信

タグ定義構造体

```
#define DEVICENAME_LENGTH 32
```

```
typedef struct SwPlcTag_t{
    int    tagId;                /* タグ ID */
    char    device[DEVICENAME_LENGTH]; /* デバイス名 */
    int    address;              /* レジスタアドレス番号 */
    int    dataType;             /* データ型 */
    int    count;                /* データ数 */
    int    monitor;              /* モニタ登録 0:Off 1:On */
    int    notify;               /* 通知登録 0:Off 1:On */
} SwPlcTag;
```

変数名	説明
tagId	タグ ID を整数で設定します。
device	デバイス名を設定します。デバイス名は NULL 終端文字列で設定してください。使用できるデバイス名は各アドインの説明を参照してください。
address	レジスタのアドレス番号を設定します。アドレス番号の詳細は各アドインの説明を参照してください。
dataType	データ型を設定します。
count	このタグのデータ数を設定します。1 以上の数値で指定してください。2 以上の数値が設定された場合は配列型タグとなります。配列型タグをプログラムから操作する時は、SwPlcWriteShortArray や SwPlcGetShortArray などの配列操作関数の関数を使用してください。
monitor	モニタ登録の設定をします。モニタ登録する場合は 1 をモニタ登録をしない場合は 0 を設定してください。モニタ登録されたタグは定周期で PLC からデータが読み込まれ、データメモリが更新されます。
notify	通知登録の設定をします。通知登録する場合は 1 を通知登録をしない場合は 0 を設定してください。通知登録されたタグはデータが変化したときにコールバックにて変化が通知されます。

データ型

データ型	説明
SWPLC_DATATYPE_SHORT	ワードデバイスを符号付き 16 ビット整数として読み書きします。
SWPLC_DATATYPE_USHORT	ワードデバイスを符号無し 16 ビット整数として読み書きします。
SWPLC_DATATYPE_INT	2 ワードデバイスを符号付き 32 ビット整数として読み書きします。2 ワードデバイスは連続した 2 つのデータレジスタで構成されます。
SWPLC_DATATYPE_UINT	2 ワードデバイスを符号無し 32 ビット整数として読み書きします。2 ワードデバイスは連続した 2 つのデータレジスタで構成されます。
SWPLC_DATATYPE_LONG	4 ワードデバイスを符号無し 64 ビット整数として読み書きします。4 ワードデバイスは連続した 4 つのデータレジスタで構成されます。
SWPLC_DATATYPE_ULONG	4 ワードデバイスを符号無し 64 ビット整数として読み書きします。4 ワードデバイスは連続した 4 つのデータレジスタで構成されます。
SWPLC_DATATYPE_FLOAT	2 ワードデバイスを 32 ビット浮動小数点数として読み書きします。2 ワードデバイスは連続した 2 つのデータレジスタで構成されます。
SWPLC_DATATYPE_DOUBLE	4 ワードデバイスを 64 ビット浮動小数点数として読み書きします。4 ワードデバイスは連続した 4 つのデータレジスタで構成されます。
SWPLC_DATATYPE_BIT	ビットデバイスを指定するときに使用するデータ型です。
SWPLC_DATATYPE_BIT2	ビットデバイスを指定するときに使用するデータ型です。アドレス+ビット位置でビットデバイスを指定する PLC で使用します。ビットデバイスの指定方法は各アドインの説明を参照してください。ビットデバイスのアドレスは下 2 桁が 00~15 のビット位置、2 桁目以降がアドレス番号を示します。例: 10015 ... アドレス 100, ビット位置 15

param

通信パラメータは下記のキーと値の組み合わせをセミコロンで区切った文字列となります。

シリアル通信設定

キー	説明
ComPort	COM ポート番号 (COM1～)
Baudrate	ボーレート (1200/2400/4800/9600/19200/38400/57600/115200/230400)
DataLen	データ長 (7/8)
StopBit	ストップビット長 (1/2)
Parity	パリティ (NONE/ODD/EVEN)
Flow	フロー制御 (NONE/RTSCTS)

TCP 通信設定

キー	説明
Address	IP アドレスまたはホスト名
Port	接続先ポート番号
ConnectionTimeout	接続待ちタイムアウト値(ミリ秒) (省略可) デフォルト値: 2000
ConnectionRetryTime	TCP 接続が切断された時に再接続リトライする間隔(ミリ秒) (省略可) 0 の場合リトライは実行されません。デフォルト値: 2000

UDP 通信設定

キー	説明
Address	IP アドレスまたはホスト名
Port	相手先ポート番号
LocalPort	ローカルポート番号 (省略可) 0 の場合は自動割り当て。デフォルト値: 0

共通パラメータ (省略可)

キー	説明
LogFilePath	通信ログ保存パスデフォルト値: 指定無し
LogFileName	通信ログファイル名ファイル名が指定されていない場合、通信ログは出力されません。デフォルト値: 指定無し
LogSize	通信ログファイル最大サイズ (byte) デフォルト値: 100000
LogNum	通信ログファイル数通信ログファイルは指定された数だけ保存されます。ファイル数が指定数を超えた場合、古いファイルから削除されます。デフォルト値: 0
Timeout	タイムアウト値 (ミリ秒) デフォルト値: 2000
Retry	通信コマンドが失敗した場合の通信リトライ回数 0 の場合リトライは実行されません。デフォルト値: 0

例 1

```
"ComPort=COM1;Baudrate=19200;DataLen=8;StopBit=1;Parity=EVEN;Flow=NONE;Timeout=1000;Retry=1;LogFilePath=C:;LogFileName=PlcLogFile;LogSize=100000;LogNum=3"
```

例 2

```
"Address=192.168.0.1;Port=5023;ConnectionTimeout=2000;ConnectionRetryTime=2000;Timeout=1000;Retry=1;LogFilePath=C:;LogFileName=PlcLogFile;LogSize=100000;LogNum=3"
```

option

通信オプションは各アドインライブラリのマニュアルを参照してください。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_ERROR	異常終了。作成した通信ポートが多すぎる場合に発生します。作成できる通信ポート数の上限は 300 です。
SWPLC_ERROR_LOADFILE	タグ定義ファイルを読み込みできませんでした。
SWPLC_ERROR_LOADLIBRARY	指定された通信アドインライブラリファイルが見つかりません。
SWPLC_ERROR_SHAREMEM	共有データメモリの初期化に失敗しました。すでに同じ名前の通信ポートがないか確認してください。
SWPLC_ERROR_TAG_DATACOUNT	タグのデータ数が不正です。データ数は 1 以上を設定してください。

SWPLC_ERROR_PARAM	通信パラメータが不正です。
SWPLC_ERROR_OPTIONPARAM	オプションパラメータが不正です。
SWPLC_ERROR_TAG_INVALIDTYPE	タグに無効なデータ型が設定されています。
SWPLC_ERROR_PORT	ポートのオープンに失敗しました。シリアル通信の場合、シリアル通信ポートのオープンに失敗。TCP または UDP 通信の場合、Socket ポートのオープンに失敗。

コード例

```
SwPlcTag tags[3];
```

```
// タグ 0 ヘデータレジスタ D1000 を short 型として登録
```

```
tags[0].tagId = 0;
strcpy(tags[0].device, "D");
tags[0].address = 1000;
tags[0].dataType = SWPLC_DATATYPE_SHORT;
tags[0].count = 1;
tags[0].monitor = 0;
tags[0].notify = 0;
```

```
// タグ 1 ヘデータレジスタ D2000 を int 型として登録
```

```
// (int 型なのでデータレジスタ D2000 と D2001 の 2 つが使用される)
```

```
tags[1].tagId = 1;
strcpy(tags[1].device, "D");
tags[1].address = 2000;
tags[1].dataType = SWPLC_DATATYPE_INT;
tags[1].count = 1;
tags[1].monitor = 0;
tags[1].notify = 0;
```

```
// タグ 2 ヘビットデバイス M100 を登録
```

```
tags[2].tagId = 1;
strcpy(tags[2].device, "M");
tags[2].address = 100;
tags[2].dataType = SWPLC_DATATYPE_BIT;
tags[2].count = 1;
tags[2].monitor = 0;
tags[2].notify = 0;
```

```
const char* param = "Address=192.168.0.1;"
                    "Port=5023;"
                    "ConnectionTimeout=2000;"
                    "ConnectionRetryTime=2000;"
                    "Timeout=1000;"
                    "Retry=1;"
                    "LogFilePath=C:¥¥log;"
```

```
"LogFileName=PlcLogFile;"  
"LogSize=100000;"  
"LogNum=3";
```

```
const char* option = "Code=BINARY;"  
"Frame=3E;"  
"MonitoringTimer=0005;"  
"NetworkNo=00;"  
"PcNo=FF;"  
"ReqDestModIONo=03FF;"  
"ReqDestModStationNo=00";
```

```
SWPLC_HANDLE handle;  
int ret = SwPlcInitPort(&handle, "TESTPORT", "McCom", SWPLC_TYPE_TCP, tags, 3, param,  
option);
```

SwPlcLoadTag

```
int SwPlcLoadTag(  
    const char* fileName,  
    SwPlcTag tags[],  
    int* tagCount)
```

タグ定義ファイルからタグ定義を読み込みます。

引数	説明
fileName	タグ定義ファイルのパスを指定します。
tags	タグ構造体配列を指定します。タグ定義はこの配列に格納されます。tags に NULL を指定した場合、タグ定義は取得せずに tagCount にタグ数のみ取得します。
tagCount	タグ構造体配列数を指定します。関数実行後は実際に取得したタグ数が格納されます。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_ERROR_LOADFILE	タグ定義ファイルを読み込みできませんでした。

コード例

```
const char* fileName = ".¥¥sample.xml";  
int tagCount;  
  
// タグ数を取得  
SwPlcLoadTag( fileName, NULL, &tagCount );  
  
// タグ定義を取得  
SwPlcTag *tags = new SwPlcTag[tagCount];  
SwPlcLoadTag( fileName, tags, &tagCount );  
  
delete[] tags;
```

SwPlcClosePort

int SwPlcClosePort(SWPLC_HANDLE handle)

通信ポートを閉じます。

引数	説明
handle	通信 ID を指定します。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。

SwPlcComStatus

int SwPlcComStatus(SWPLC_HANDLE handle)

通信ポートの状態を取得します。

引数	説明
handle	通信 ID を指定します。

戻り値

状態コードを返します。

シリアル通信または UDP 通信の場合、通信ポートが正常に開かれた後は常に SWPLC_PORT_OPEN を返します。SwPlcClosePort で通信ポートを閉じた後は SWPLC_INVALID_ID を返します。

コード	説明
SWPLC_INVALID_ID	無効な通信 ID が指定されました。
SWPLC_PORT_OPEN	ポートオープン

TCP 通信の場合、通信ポートが正常に開かれた後は SWPLC_PORT_OPEN を返します。通信ポートが開かれた後、TCP 接続が試行され接続が確立すると SWPLC_PORT_CONNECTED を返します。TCP 接続が切断された場合、一旦ポートが閉じられます。ポートが閉じられた状態のときは SWPLC_PORT_CLOSE を返します。ポートが閉じられた後は自動でポートが再度開かれ、TCP 接続が試行されます。TCP 接続の試行は SwPlcClosePort で通信ポートが閉じられるまで繰り返し実行されます。SwPlcClosePort で通信ポートを閉じた後は SWPLC_INVALID_ID を返します。

コード	値
SWPLC_INVALID_ID	無効な通信 ID が指定されました。
SWPLC_PORT_CLOSE	ポートクローズ
SWPLC_PORT_OPEN	ポートオープン
SWPLC_PORT_CONNECTED	TCP コネクション確立

SwPlcSetNotify

```
int SwPlcSetNotify(  
    SWPLC_HANDLE handle,  
    SwPlcNotifyCallbackFunc callbackFunc)
```

コールバック関数を登録します。コールバック関数を登録した場合、各種通知を受け取ることができます。

コールバック関数はライブラリ内の通信スレッドから呼び出されます。コールバック関数内の処理は別スレッドで実行されていることを考慮する必要があります。

コールバック関数内で時間のかかる処理を実行すると通信に影響します。コールバック関数内は速やかに処理を完了するようにしてください。

引数	説明
handle	通信 ID を指定します。
callbackFunc	コールバック関数を指定します。

コールバック関数

```
typedef void (CALLBACK *SwPlcNotifyCallbackFunc)(  
    UINT msg,  
    SWPLC_HANDLE handle,  
    int param);
```

引数	説明
msg	通知メッセージ
handle	通知元の通信 ID
param	通知データ

通知メッセージ

通知メッセージ	値	説明
SWPLC_MSG_RESPONSE	0	コマンドの実行が完了したときに通知されます。param はコマンド実行の結果コードが格納されます。詳細は SwPlcResponse の戻り値を参照してください。
SWPLC_MSG_DATACHANGED	1	通知登録されたタグの値が変化した時に通知されます。param はタグ ID が格納されます。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。

コード例

```
SwPlcSetNotify(handle, SwPlcNotifyCallback);  
void CALLBACK SwPlcNotifyCallback(UINT msg, SWPLC_HANDLE handle, int param)  
{  
    if (msg == SWPLC_MSG_DATACHANGED) {  
        short data;  
        SwPlcGetShort(handle, param, &data);  
        TRACE("TagId=%d Value=%d\r\n", param, data);  
    }  
}
```

SwPlcExecute

int SwPlcExecute(SWPLC_HANDLE handle)

実行キューに登録されたコマンドを順次実行します。命令は非同期で実行され、実行結果は SwPlcResponse で取得します。実行キューに登録されたコマンドは SwPlcExecute 実行後に全てクリアされます。SwPlcExecute 実行後は、次に実行するコマンドを実行キューに登録することができますが、前回実行したコマンド処理が完了していない状態で次の SwPlcExecute を実行した場合、SWPLC_ERROR_BUSY となり実行できません。

引数	説明
handle	通信 ID を指定します。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。
SWPLC_ERROR_BUSY	実行中のコマンドがあります。前回実行したコマンドが全て完了していません。
SWPLC_ERROR_NOCOMMAND	実行するコマンドがありません。実行するコマンドを SwPlcRead、SwPlcWrite で登録してから実行してください。

SwPlcResponse

```
int WINAPI SwPlcResponse(  
    SWPLC_HANDLE handle,  
    int wait)
```

SwPlcExecute の実行結果を取得します。

引数	説明
handle	通信 ID を指定します。
wait	待ち時間(ミリ秒)0: 待機なし-1: コマンド実行が完了するまで待機

戻り値

コマンド実行中は SWPLC_RESPONSE_BUSY を返します。

コマンド実行が正常完了した場合は下記のコードを返します。

コード	説明
SWPLC_RESPONSE_IDLE	コマンドは実行されていません。(実行キューが空の状態 SwPlcExecute が実行された)
SWPLC_RESPONSE_DONE	コマンド実行完了。
SWPLC_RESPONSE_ABORT	コマンド実行中断。

コマンド実行が異常完了した場合は下記のコードを返します。

コード	説明
SWPLC_INVALID_ID	無効な通信 ID が指定されました。
SWPLC_ERROR	実行エラー。(通信ポートが通信出来ない状態)
SWPLC_ERROR_OUTOFMEMORY	送受信メモリが足りません。問合せデータ数を少なく してください。
SWPLC_RESPONSE_TIMEOUT	コマンドの応答がタイムアウトしました。
SWPLC_RESPONSE_ERROR_PLG	PLC からエラー応答を受信しました。 SwPlcGetPlcErrorCode にて PLC から受信したエラーコード を取得できます。

SwPlcClear

int SwPlcClear(SWPLC_HANDLE handle)

実行キューに登録されたコマンドを全てクリアします。SwPlcExecute を実行した後の実行中のキューには影響しません。

引数	説明
handle	通信 ID を指定します。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。

SwPlcAbort

int SwPlcAbort(SWPLC_HANDLE handle)

実行中のコマンドを中止します。

引数	説明
handle	通信 ID を指定します。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。

SwPlcGetPlcErrorCode

```
int SwPlcGetPlcErrorCode(  
    SWPLC_HANDLE handle,  
    int *pErrorCode)
```

コマンド実行で PLC から受信した PLC のエラーコードを取得します。

引数	説明
handle	通信 ID を指定します。
pErrorCode	PLC からのエラーコードを格納する変数を指定します。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。

SwPlcRead

```
int SwPlcRead(  
    SWPLC_HANDLE handle,  
    int tagId)
```

実行キューにデータ読み込みコマンドを追加します。読み込みコマンドが成功すると、タグデータメモリが更新されます。タグデータは SwPlcGet 関数で取得できます。

引数	説明
handle	通信 ID を指定します。
tagId	対象のタグ ID を指定します。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。
SWPLC_ERROR_TAG_RANGE	指定されたタグ ID が無効です。タグ ID は 0 以上で指定してください。
SWPLC_ERROR_TAG_NOTFOUND	指定されたタグ ID が見つかりません。タグ ID とタグの定義を確認してください。

SwPlcWrite

```
int SwPlcWriteBit(SWPLC_HANDLE handle, int tagId, unsigned char data)
int SwPlcWriteShort(SWPLC_HANDLE handle, int tagId, short data)
int SwPlcWriteUShort(SWPLC_HANDLE handle, int tagId, unsigned short data)
int SwPlcWriteInt(SWPLC_HANDLE handle, int tagId, int data)
int SwPlcWriteUInt(SWPLC_HANDLE handle, int tagId, unsigned int data);
int SwPlcWriteLong(SWPLC_HANDLE handle, int tagId, __int64 data);
int SwPlcWriteULong(SWPLC_HANDLE handle, int tagId, unsigned __int64 data);
int SwPlcWriteFloat(SWPLC_HANDLE handle, int tagId, float data);
int SwPlcWriteDouble(SWPLC_HANDLE handle, int tagId, double data);
```

実行キューにデータ書き込みコマンドを追加します。タグ定義で設定した型と同じ型名の関数を実行してください。

SwPlcWriteBit の data は 0 か 1 を格納します。0 はビットデバイスの OFF、1 はビットデバイスの ON に対応します。

引数	説明
handle	通信 ID を指定します。
tagId	対象のタグ ID を指定します。
data	書き込むデータを指定します。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。
SWPLC_ERROR_TAG_RANGE	指定されたタグ ID が無効です。タグ ID は 0 以上で指定してください。
SWPLC_ERROR_TAG_NOTFOUND	指定されたタグ ID が見つかりません。タグ ID とタグの定義を確認してください。
SWPLC_ERROR_DATATYPE_MISMATCH	タグに定義されたデータ型と実行された関数のデータ型が一致しません。タグ定義と一致するデータ型の関数を実行してください。

SwPlcWriteArray

```
int SwPlcWriteBitArray(SWPLC_HANDLE handle, int tagId, unsigned char data[], int count)
int SwPlcWriteShortArray(SWPLC_HANDLE handle, int tagId, short data[], int count)
int SwPlcWriteUShortArray(SWPLC_HANDLE handle, int tagId, unsigned short data[], int count)
int SwPlcWriteIntArray(SWPLC_HANDLE handle, int tagId, int data[], int count)
int SwPlcWriteUIntArray(SWPLC_HANDLE handle, int tagId, unsigned int data[], int count)
int SwPlcWriteLongArray(SWPLC_HANDLE handle, int tagId, __int64 data[], int count)
int SwPlcWriteULongArray(SWPLC_HANDLE handle, int tagId, unsigned __int64 data[], int count)
int SwPlcWriteFloatArray(SWPLC_HANDLE handle, int tagId, float data[], int count)
int SwPlcWriteDoubleArray(SWPLC_HANDLE handle, int tagId, double data[], int count)
```

実行キューにデータ書き込みコマンドを追加します。配列型タグのデータ書き込みで使います。
タグ定義で設定した型と同じ型名の関数を実行してください。

SwPlcWriteBitArray の data は 0 か 1 を格納します。0 はビットデバイスの OFF、1 はビットデバイスの ON に対応します。

引数	説明
handle	通信 ID を指定します。
tagId	対象のタグ ID を指定します。
data	書き込む配列データを指定します。
count	配列データ数を指定します。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。
SWPLC_ERROR_TAG_RANGE	指定されたタグ ID が無効です。タグ ID は 0 以上で指定してください。
SWPLC_ERROR_TAG_NOTFOUND	指定されたタグ ID が見つかりません。タグ ID とタグの定義を確認してください。
SWPLC_ERROR_DATATYPE_MISMATCH	タグに定義されたデータ型と実行された関数のデータ型が一致しません。タグ定義と一致するデータ型の関数を実行してください。

SwPlcGet

```
int SwPlcGetBit(SWPLC_HANDLE handle, int tagId, unsigned char* data)
int SwPlcGetShort(SWPLC_HANDLE handle, int tagId, short* data)
int SwPlcGetUShort(SWPLC_HANDLE handle, int tagId, unsigned short* data)
int SwPlcGetInt(SWPLC_HANDLE handle, int tagId, int* data)
int SwPlcGetUInt(SWPLC_HANDLE handle, int tagId, unsigned int* data);
int SwPlcGetLong(SWPLC_HANDLE handle, int tagId, __int64* data);
int SwPlcGetULong(SWPLC_HANDLE handle, int tagId, unsigned __int64* data);
int SwPlcGetFloat(SWPLC_HANDLE handle, int tagId, float* data);
int SwPlcGetDouble(SWPLC_HANDLE handle, int tagId, double* data);
```

指定されたタグ ID のデータをタグデータメモリから取得します。タグ定義で設定した型と同じ型名の関数を実行してください。タグデータは SwPlcRead または、SwPlcWrite の実行で更新されます。

SwPlcGetBit の data はビットデバイスが ON の時は 1、OFF の時は 0 が格納されます。

引数	説明
handle	通信 ID を指定します。
tagId	対象のタグ ID を指定します。
data	データを格納する変数を指定します。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。
SWPLC_ERROR_TAG_RANGE	指定されたタグ ID が無効です。タグ ID は 0 以上で指定してください。
SWPLC_ERROR_TAG_NOTFOUND	指定されたタグ ID が見つかりません。タグ ID とタグの定義を確認してください。
SWPLC_ERROR_DATATYPE_MISMATCH	タグに定義されたデータ型と実行された関数のデータ型が一致しません。タグ定義と一致するデータ型の関数を実行してください。
SWPLC_TAG_STATUS_NONE	タグのが不明です。通信ポートが開かれてから読み込みまたは書き込みが一回も実行されていません。
SWPLC_TAG_STATUS_ERROR	タグの読み込みまたは書き込みに失敗した状態です。

SwPlcGetArray

```
int SwPlcGetBitArray(SWPLC_HANDLE handle, int tagId, short data[], int count)
int SwPlcGetShortArray(SWPLC_HANDLE handle, int tagId, short data[], int count)
int SwPlcGetUShortArray(SWPLC_HANDLE handle, int tagId, unsigned short data[], int count)
int SwPlcGetIntArray(SWPLC_HANDLE handle, int tagId, int data[], int count)
int SwPlcGetUIntArray(SWPLC_HANDLE handle, int tagId, unsigned int data[], int count)
int SwPlcGetLongArray(SWPLC_HANDLE handle, int tagId, __int64 data[], int count)
int SwPlcGetULongArray(SWPLC_HANDLE handle, int tagId, unsigned __int64 data[], int count)
int SwPlcGetFloatArray(SWPLC_HANDLE handle, int tagId, float data[], int count)
int SwPlcGetDoubleArray(SWPLC_HANDLE handle, int tagId, double data[], int count)
```

指定されたタグ ID のデータをタグデータメモリから取得します。配列型タグのデータ取得で使
用します。タグ定義で設定した型と同じ型名の関数を実行してください。タグデータは
SwPlcRead または、SwPlcWrite の実行で更新されます。

SwPlcGetBitArray の data はビットデバイスが ON の時は 1、OFF の時は 0 が格納されます。

引数	説明
handle	通信 ID を指定します。
tagId	対象のタグ ID を指定します。
data	データを格納する配列変数を指定します。
count	配列データ数を指定します。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。
SWPLC_ERROR_TAG_RANGE	指定されたタグ ID が無効です。タグ ID は 0 以上で指定してください。
SWPLC_ERROR_TAG_NOTFOUND	指定されたタグ ID が見つかりません。タグ ID とタグの定義を確認してください。
SWPLC_ERROR_DATATYPE_MISMATCH	タグに定義されたデータ型と実行された関数のデータ型が一致しません。タグ定義と一致するデータ型の関数を実行してください。
SWPLC_TAG_STATUS_NONE	タグの状態が不明です。通信ポートが開かれてから読み込みまたは書き込みが一回も実行されていません。
SWPLC_TAG_STATUS_ERROR	タグの読み込みまたは書き込みに失敗した状態です。

SwPlcSetMonitorInterval

```
int SwPlcSetMonitorInterval(  
    SWPLC_HANDLE handle,  
    int ms)
```

モニタ周期をミリ秒単位で設定します。モニタ登録されたタグはこの関数で設定されたモニタ周期で読み込みが実行されます。正常に読み込みが来ているかは SwPlcGet 関数の戻り値が SWPLC_OK かどうかで判断します。SwPlcGet 関数の戻り値が SWPLC_TAG_STATUS_ERROR である場合、通信エラーなどで正常に読み込みできていない状態です。0 でモニタが停止します。モニタ登録されたタグが多すぎると、通信負荷の増大により設定周期でモニタできない場合がありますのでご注意ください。

引数	説明
handle	通信 ID を指定します。
ms	モニタ周期時間(ミリ秒単位)0 指定でモニタ停止

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。

SwPlcMonitorStatus

int SwPlcMonitorStatus(SWPLC_HANDLE handle)

モニタ通信の状態を取得します。

引数	説明
handle	通信 ID を指定します。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。
SWPLC_ERROR	モニタ通信中に異常が発生しました。
SWPLC_RESPONSE_ERROR_PLG	PLC からエラー応答を受信しました。 SwPlcGetPlcErrorCode にて PLC から受信したエラーコードを取得できます。

SwPlcGetMonitorPlcErrorCode

```
int SwPlcGetMonitorPlcErrorCode(  
    SWPLC_HANDLE handle,  
    int *pErrorCode)
```

SwPlcMonitorStatus の応答が SWPLC_RESPONSE_ERROR_PLC の場合、本関数を実行して PLC から受信したエラーコードを取得できます。

引数	説明
handle	通信 ID を指定します。
pErrorCode	PLC から受信したエラーコードを格納する変数を指定します。

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。

SwPlcWriteLog

```
int SwPlcWriteLog(  
    SWPLC_HANDLE handle,  
    const char* logString)
```

通信ログファイルに文字列を出力します。

引数	説明
handle	通信 ID を指定します。
logString	出力する文字列

戻り値

正常の場合、SWPLC_OK を返します。異常の場合、SWPLC_OK 以外のコードを返します。

戻り値	説明
SWPLC_OK	正常終了。
SWPLC_INVALID_ID	無効な通信 ID が指定されました。

エラーコード一覧

コード	値	説明
SWPLC_ERROR_TRIAL_TIMEOVER	-2	試用時間が超過しました。
SWPLC_INVALID_ID	-1	通信 ID が不正です。
SWPLC_OK	0	正常。
SWPLC_ERROR	1	実行エラー。通信ポートが使用できない状態でコマンド実行した時に発生します。
SWPLC_ERROR_LOADFILE	2	ファイルのロードに失敗しました。
SWPLC_ERROR_LOADLIBRARY	3	通信ライブラリのロードに失敗しました。
SWPLC_ERROR_PARAM	4	通信パラメータが不正です。
SWPLC_ERROR_OPTIONPARAM	5	オプションパラメータが不正です。
SWPLC_ERROR_PORT	6	通信ポートの初期化に失敗しました。
SWPLC_ERROR_BUSY	7	前回実行したコマンドが完了していません。
SWPLC_ERROR_NOCOMMAND	8	実行コマンドが登録されていません。
SWPLC_ERROR_OUTOFMEMORY	9	送受信メモリが足りません。1 度の要求で実行されるコマンドサイズが大きくなりすぎたためコマンド作成用のメモリが足りない状態です。1 度の要求で問合せするデータ数を少なくしてください。
SWPLC_ERROR_TAG_RANGE	10	タグ ID が範囲外です。
SWPLC_ERROR_TAG_NOTFOUND	11	タグ ID が登録されていません。
SWPLC_ERROR_TAG_INVALIDTYPE	12	タグのデータ型が不正です。
SWPLC_ERROR_DATATYPE_MISMATCH	13	データ型が一致しません。
SWPLC_ERROR_SHAREMEM	14	共有データメモリの初期化に失敗しました。ポート名称に使用できない文字が指定されたか、ポート名称が他のポートと重複している可能性があります。
SWPLC_ERROR_TAG_DATACOUNT	15	タグのデータ数が不正です。
SWPLC_PORT_CLOSE	100	ポートクローズ。TCP 通信の場合に、通信相手先からコネクションが切断された場合に発生します。
SWPLC_PORT_OPEN	101	ポートオープン。シリアルポートまたはソケットポートが開かれた状態で

		す。
SWPLC_PORT_CONNECTED	102	TCP コネクション確立。ソケットポートが開かれた後、通信先と TCP コネクションが確立した状態です。TCP 通信の場合、TCP コネクションが確立した後に通信が可能となります。
SWPLC_TAG_STATUS_NONE	200	タグの状態が不明です。通信ポートが開かれてから読み込みまたは書き込みが一回も実行されていません。
SWPLC_TAG_STATUS_ERROR	201	タグの読み込みまたは書き込みに失敗した状態です。
SWPLC_RESPONSE_IDLE	1000	コマンドは実行されていません。コマンドがキューに登録されていない状態で SwPlcExecute が実行された状態です。
SWPLC_RESPONSE_DONE	1001	コマンド実行完了。
SWPLC_RESPONSE_BUSY	1002	コマンド実行中。
SWPLC_RESPONSE_ABORT	1003	コマンド実行中断。SwPlcAbort により実行が中断されました。
SWPLC_RESPONSE_TIMEOUT	1004	コマンド応答タイムアウト。通信コマンドの PLC からの応答待ちでタイムアウトが発生した状態です。
SWPLC_RESPONSE_ERROR_PLG	1005	PLC エラー。PLC からエラー応答を受信。

サンプルプログラム

Visual C++のプログラム例です。

例 1. タグ定義ファイルから設定を読み込んで通信を行う

```
SWPLC_HANDLE handle;

// タグ定義ファイルを使用してポートを開く
if (SwPlcLoadFile(&handle, "sample.xml") != SWPLC_OK) {
    // ポートオープン失敗
    return;
}

// Tag0～Tag2 に書込み
// 書込みコマンドを実行キューに登録
SwPlcWriteShort(handle, 0, 100);
SwPlcWriteShort(handle, 1, 200);
SwPlcWriteShort(handle, 2, 300);

// 実行
if (SwPlcExecute(handle) != SWPLC_OK) {
    // 実行エラー
    SwPlcClosePort(handle);
    return;
}

// コマンド実行の結果を取得(-1 指定でコマンド実行が完了するまで待つ)
if (SwPlcResponse(handle, -1) == SWPLC_RESPONSE_DONE) {
    // 書込み正常完了
} else {
    // 書込みエラー
}

// Tag0～Tag2 を読込み
// 読込みコマンドを実行キューに登録
SwPlcRead(handle, 0);
SwPlcRead(handle, 1);
SwPlcRead(handle, 2);

// 実行
if (SwPlcExecute(handle) != SWPLC_OK) {
    // 実行エラー
    SwPlcClosePort(handle);
    return;
}

// コマンド実行の結果を取得(-1 指定でコマンド実行が完了するまで待つ)
```



```

if (SwPlcResponse(handle, -1) == SWPLC_RESPONSE_DONE) {
    // 読み込み正常完了
    // 読み込んだ値を取得
    short data0;
    short data1;
    short data2;
    SwPlcGetShort(handle, 0, &data0);
    SwPlcGetShort(handle, 1, &data1);
    SwPlcGetShort(handle, 2, &data2);
} else {
    // 読み込みエラー
}

```

```

// ポートを閉じる
SwPlcClosePort(handle);

```

例 2. タグ定義ファイルを使用せずに通信ポートを作成し通信を行う
SwPlcTag tags[3];

```

// タグ 0 ヘデータレジスタ D1000 を short 型として登録
tags[0].tagId = 0;
strcpy(tags[0].device, "D");
tags[0].device[1] = '¥0'; // デバイス名は NULL 終端文字列で設定
tags[0].address = 1000;
tags[0].dataType = SWPLC_DATATYPE_SHORT;
tags[0].count = 1;
tags[0].monitor = 0;
tags[0].notify = 0;

```

```

// タグ 1 ヘデータレジスタ D2000 を int 型として登録
// (int 型なのでデータレジスタ D2000 と D2001 の 2 つが使用される)
tags[1].tagId = 1;
strcpy(tags[1].device, "D");
tags[1].address = 2000;
tags[1].dataType = SWPLC_DATATYPE_INT;
tags[1].count = 1;
tags[1].monitor = 0;
tags[1].notify = 0;

```

```

// タグ 2 ヘビットデバイス M100 を登録
tags[2].tagId = 1;
strcpy(tags[2].device, "M");
tags[2].address = 100;
tags[2].dataType = SWPLC_DATATYPE_BIT;
tags[2].count = 1;
tags[2].monitor = 0;
tags[2].notify = 0;

```

```

const char* param = "Address=192.168.0.1;"
                    "Port=5023;"
                    "ConnectionTimeout=2000;"
                    "ConnectionRetryTime=2000;"
                    "Timeout=1000;"
                    "Retry=1;"
                    "LogFilePath=C:\\¥¥log;"
                    "LogFileName=PlcLogFile;"
                    "LogSize=100000;"
                    "LogNum=3";

const char* option = "Code=BINARY;"
                    "Frame=3E;"
                    "MonitoringTimer=0005;"
                    "NetworkNo=00;"
                    "PcNo=FF;"
                    "ReqDestModIONo=03FF;"
                    "ReqDestModStationNo=00";

SWPLC_HANDLE handle;
int ret = SwPlcInitPort(&handle, "TESTPORT", "McCom", SWPLC_TYPE_TCP, tags, 3, param,
option);

```

改訂履歴

改訂日	改定内容
2023/08/30	サポート OS に Windows11 を追加。
2023/12/12	サポート OS に Windows Server2019 / 2022 を追加。 アドインライブラリに ModbusCom.dll を追加。
2024/12/06	アドインライブラリに ToyopucCom.dll を追加。 UDP 通信パラメータにローカルポート番号を追加。（※SwPlcInitPort の UDP 通信欄を参照）
2025/12/09	作成できるポート数上限を 256 から 300 に変更。