

# SW-PLC ライブライ

## スクリプトリファレンス

株式会社セイワ技研  
2017 年 12 月 15 日

## 目次

概要	2
スクリプトテンプレート	3
機能	4
関数	4
イベント	4
クラス	4
プロパティ	4
SetTimer	5
KillTimer	6
PlcRead	7
PlcMultiRead	8
PlcWrite	9
PlcMultiWrite	10
OutputLog	11
PlcData クラス	12
TagItem クラス	13
PlcTagItems プロパティ	14
OnInit イベント	15
OnExit イベント	15
OnTimer イベント	15
OnDataChange イベント	15
サンプルスクリプト	16

## **概要**

SW-TagManager でスクリプトを利用すると PLC 接続時、またはダミーPLC を起動したときの動作をカスタマイズすることができます。スクリプトは C#で記述します。スクリプト内では通常の C#で使用できる機能に加えて以降に説明する機能が使用可能です。

※サポートされる C#の言語バージョンは C#5.0 です。

※C#の言語仕様については、マイクロソフトのサイトや C#について説明した書籍、サイトなどを参考にしてください。

## スクリプトテンプレート

スクリプトは PlcScriptBase クラスを継承したクラスとして作成します。スクリプトを作成する場合は下記テンプレートを元に作成してください。

```
using SwTagManager. Scripting;
using System;
using System. IO;
using System. Text;

class PlcScript : PlcScriptBase
{
    void OnInit()
    {
    }

    void OnExit()
    {
    }

    void OnTimer(int timerId)
    {
    }

    void OnDataChange(int tagId, object value)
    {
    }
}
```

## 機能

スクリプト内の PlcScriptBase クラスでは下記の機能が使用できます。

## 関数

関数名	説明
SetTimer	タイマーを作成します。
KillTimer	タイマーを破棄します。
PlcRead	PLC からデータを読み込みます。
PlcMultiRead	PLC から複数データを読み込みます。
PlcWrite	PLC へデータを書き込みます。
PlcMultiWrite	PLC へ複数データを書き込みます。
OutputLog	メッセージウィンドウにログメッセージを出力します。

## イベント

イベント名	説明
OnInit	スクリプトが開始された時に呼び出されるイベントです。
OnExit	スクリプトが終了された時に呼び出されるイベントです。
OnTimer	SetTimer で作成したタイマーで設定時間が経過したときに呼び出されるイベントです。
OnDataChange	通知設定されたタグの値が変化したときに呼び出されるイベントです。

## クラス

クラス名	説明
PlcData	PLC データを格納するクラスです。
TagItem	タグ定義が格納されるクラスです。

## プロパティ

プロパティ名	説明
PlcTagItems	TagItem クラスの配列として定義されたプロパティです。スクリプト内からタグ定義を参照できます。

## **SetTimer**

タイマーを作成します。タイマーを作成すると設定されたタイマー間隔で OnTimer イベントが呼び出されます。

```
void SetTimer(int timerId, int interval)
```

引数

timerId: タイマー識別番号

interval: タイマー間隔(ミリ秒)

コード例

```
// 3000 ミリ秒毎に OnTimer イベントが呼び出されます  
SetTimer(1, 3000);
```

## KillTimer

タイマーを破棄します。

```
void KillTimer(int timerId)
```

引数

timerId: タイマー識別番号

コード例

```
// タイマーID1 のタイマーを破棄  
KillTimer(1);
```

## PlcRead

PLC からデータを読み込みます。

```
bool PlcRead(PlcData data)
```

引数

data: 読込みデータ

戻り値

true: 成功 false: 失敗

### コード例

```
// タグ ID1(データ型: USHORT) を読み込み
var data = new PlcData { TagId = 1 };

if (PlcRead(data))
{
    OutputLog("Tag1={0}", (ushort)data.Value);
}

// 配列型タグの読み込み
// タグ ID2(データ型: USHORT, データ数: 3) を読み込み
var data = new PlcData { TagId = 2 };

if (PlcRead(data))
{
    ushort[] values = (ushort[])data.Value;
    OutputLog("Tag2[0]={0}", values[0]);
    OutputLog("Tag2[1]={0}", values[1]);
    OutputLog("Tag2[2]={0}", values[2]);
}
```

## PlcMultiRead

PLC から複数データを読み込みます。複数のデータを読み込む場合は、PlcRead 関数を複数回実行するよりも本関数を使用することで高速に読み込みを実行できます。

```
bool PlcMultiRead(PlcData[] data)
```

引数

data: 読込みデータを配列で指定

戻り値

true: 成功 false: 失敗

コード例

```
// タグ ID1～3(データ型: USHORT)を読み込み
var data = new PlcData[3];
data[0] = new PlcData { TagId = 1 };
data[1] = new PlcData { TagId = 2 };
data[2] = new PlcData { TagId = 3 };

if (PlcMultiRead(data))
{
    OutputLog("Tag1={0}", (ushort)data[0].Value);
    OutputLog("Tag2={0}", (ushort)data[1].Value);
    OutputLog("Tag3={0}", (ushort)data[2].Value);
}
```

## PlcWrite

PLC へデータを書き込みます。

```
bool PlcWrite(PlcData data)
```

引数

data: 書込みデータ

戻り値

true: 成功 false: 失敗

### コード例

```
// タグ ID1(データ型: USHORT)へ書き込み
var data = new PlcData { TagId = 1, Value = (ushort)100 };

if (PlcWrite(data))
{
    OutputLog("書き込み完了");
}

// 配列型タグへ書き込み
// タグ ID2(データ型: USHORT, データ数: 3)へ書き込み
ushort[] values = new ushort[] { 100, 200, 300 };
var data = new PlcData { TagId = 2, Value = values };

if (PlcWrite(data))
{
    OutputLog("書き込み完了");
}
```

## PlcMultiWrite

PLC へ複数データを書込みます。複数のデータを書込む場合は、PlcWrite 関数を複数回実行するよりも本関数を使用することで高速に書込みを実行できます。

```
bool PlcMultiWrite(PlcData[] data)
```

引数

data: 書込みデータを配列で指定

戻り値

true: 成功 false: 失敗

### コード例

```
// タグ ID1～3(データ型: USHORT)へ書込み
var data = new PlcData[3];
data[0] = new PlcData { TagId = 1, Value = (ushort)100 };
data[1] = new PlcData { TagId = 2, Value = (ushort)200 };
data[2] = new PlcData { TagId = 3, Value = (ushort)300 };

if (PlcMultiWrite(data))
{
    OutputLog("書込み完了");
}
```

## OutputLog

メッセージウィンドウにログメッセージを出力します。

```
// 文字列メッセージ出力  
void OutputLog(string message)
```

引数

message: 出力メッセージ

```
// 書式指定文字列によるメッセージ出力
```

```
void OutputLog(string format, params object[] args)
```

引数

message: 書式指定文字列

args: 書式指定対象の値を設定

コード例

```
// 文字列メッセージ出力  
OutputLog("Message");
```

```
// 書式指定 (data.TagId = 1, data.Value = 100, UShort 型)
```

```
OutputLog("TagId: {0}, Value: {1}", data.TagId, (ushort)data.Value);
```

```
// --> TagId: 1, Value: 100
```

```
// 書式指定 (data2.TagId = 2, data.Value = 1.2345, Double 型)
```

```
OutputLog("TagId: {0}, Value: {1:0.000}", data2.TagId, (double)data2.Value);
```

```
// --> TagId: 2, Value: 1.235
```

## PlcData クラス

PLC データを格納するクラスです。PlcRead, PlcMultiRead, PlcWrite, PlcMultiWrite の引数で使用します。

プロパティ	型	説明
TagId	int	読み込み、書き込み対象のタグ ID
Value	object	データの格納先です。データを設定または参照するときはタグ定義のデータ型に一致する C# のデータ型にキャストしてください。(*1)

(\*1) データ型

タグ定義	C#データ型
Short	short
UShort	ushort
Int	int
UInt	uint
Long	long
ULong	ulong
Floag	float
Double	double
Bit	byte
Bit2	byte

配列型タグの場合は上記 C#データ型の配列となります。

配列型タグの使用方法は PlcRead、PlcWrite のコード例を参照してください。

## TagItem クラス

タグ定義が格納されるクラスです。タグ定義の参照方法については、PicTagItems プロパティの説明を確認してください。

プロパティ	型	説明
TagId	int	タグ ID
Device	string	デバイス名
Address	int	アドレス
DataType	int	データ型(1: Short 2: UShort 3: Int 4: UInt 5: Long 6: ULong 7: Float 8: Double 9: Bit 10: Bit2)
Count	int	データ数
Name	string	タグ名
Comment	string	コメント

## PlcTagItems プロパティ

```
TagItem[] PlcTagItems;
```

TagItem クラスの配列として定義されたプロパティです。スクリプト内からタグ定義を参照できます。参照のみでタグ定義を書き換えることは出来ません。

### コード例

```
// Tag0～Tag2 のタグ定義を参照
for (int i = 0; i < 3; i++)
{
    TagItem item = PlcTagItems[i];
    OutputLog("TagId: {0} Name: {1} Device: {2} {3}", item.TagId, item.Name, item.Device,
    item.Address);
}
```

## **OnInit イベント**

`void OnInit()`

スクリプトが開始された時に呼び出されるイベントです。

## **OnExit イベント**

`void OnExit()`

スクリプトが終了された時に呼び出されるイベントです。

## **OnTimer イベント**

`void OnTimer(int timerId)`

`SetTimer` で作成したタイマーで設定時間が経過したときに呼び出されるイベントです。

- `timerId`  
`SetTimer` で設定した番号が格納されます。

## **OnDataChange イベント**

`void OnDataChange(int tagId, object value)`

通知設定されたタグの値が変化したときに呼び出されるイベントです。

- `tagId`  
値が変化したタグ ID が格納されます。
- `value`  
変化した値が格納されます。

## サンプルスクリプト

```
using SwTagManager. Scripting;
using System;
using System. IO;
using System. Text;

class PlcScript : PlcScriptBase
{
    void OnInit()
    {
        OutputLog("Script: Start");

        // タグ1～3を初期化
        var data = new PlcData[]
        {
            new PlcData { TagId = 1, Value = (ushort)100},
            new PlcData { TagId = 2, Value = (ushort)200},
            new PlcData { TagId = 3, Value = (ushort)300},
        };

        PlcMultiWrite(data);

        // 3秒のタイマーを作成
        SetTimer(1, 3000);
    }

    void OnExit()
    {
        OutputLog("Script: End");
        KillTimer(1);
    }

    void OnTimer(int timerId)
    {
        OutputLog("Script: timer id ={0}", timerId);

        // タグの読み込み
        var data = new PlcData { TagId = 1 };
        if (PlcRead(data))
        {
            OutputLog("Script: read Value={0}", (ushort)data.Value);
            // ファイルに出力
            string message = DateTime. Now + ",";
        }
    }
}
```

```

        + data.TagId + ", "
        + data.Value + "\r\n";
    File.AppendAllText(@"/d:/temp/sample.csv", message);
}
}

void OnDataChange(int tagId, object value)
{
    OutputLog("Script: OnDataChange tag = {0}, data = {1}", tagId, value);
    // タグ 0 の値が 1 に変化したときに処理(タグ 0 の通知を ON に設定しておく)
    if (tagId == 0 && (ushort)value == 1)
    {
        // タグの書き込み
        var data = new PlcData { TagId = 1, Value = (ushort)10 };
        if (PlcWrite(data))
        {
            OutputLog("Script: write tag = {0}, Value = {1}",
                      data.TagId, data.Value);
        }
    }
}
}

```