EPW-30

DLL インターフェース仕様書　1.00

~EPW-30U/EPW-30S/EPW-ISP 対応~

2017/02/03

SolitonWave Co.,Ltd.

■ 改版履歴

Ver 1.00　　2017 年 2 月 23 日

目次

EPW-30 インターフェース DLL について

EPW-30 には、EPW-30U/EPW-30S/EPW-ISP の 3 つの製品があります。この DLL では、この 3 つの製品すべてに対応しています。
EPW-30 の DLL は、.net 環境や VBA の環境にも対応していますので、C#や VisualBasic、マイクロソフト・エクセルの VBA からも容易にアクセスすることが可能です。*

API 仕様書で使われているデータ型について

本 API では、次のようなデータ型を使用しています。この表では C 言語のデータ型を参考用に記載しています。ほかの言語を使用する場合は、その言語の仕様書を確認して、適切な型を使用するようにしてください。

番号	型名	C 言語の型／値	内容
1	BOOL	int	0=FALSE,それ以外を TRUE とする
2	TRUE	0	数値＝0 の整数
3	FALSE	1	数値＝1 の整数
4	BYTE	unsigned char	8 ビットの符号なし整数
5	WORD	unsigned short	16 ビットの符号なし整数
6	int	int	32 ビットの符号付整数
7	VOID	void	引数や戻り値、型が無い事を示す

また、ポインタ渡しの引数は、使用する言語によっては、参照渡しに読み替える必要があります。

EPW30 インターフェース API

EPW インターフェース API には、次の物があります。

番号	API 名称	機能
1	EpwOpen	EPW-30 インターフェースをオープンします
2	EpwClose	EPW-30 インターフェースをクローズします
3	EpwMode	マイクロワイヤ/I2C/SPI のモードを切り替えます
4	EpwCheck	デバイスの接続の有無の確認
5	EpwRead	デバイスのデータを読み出します
6	EpwWrite	デバイスにデータを書き込みます
7	EpwErase	デバイスを消去します
8	EpwCompare	デバイスのデータを比較します
9	EpwCheksum	デバイスのチェックサムを取得します
10	EpwBlankChk	デバイスのブランクチェックを行います
11	EpwDllVer	DLL のバージョン番号を返します
12	EpwFrmVer	EPW-30 のファームウェアバージョンを返します
13	EpwSetFreq	クロック周波数を設定します
14	EpwGetFreq	クロック周波数を読み出します
15	EpwGetSwitch	スイッチの状態を読み出します(EPW-ISP 専用)
16	EpwSetEvent	スイッチイベントの設定(EPW-ISP 専用)
17	EpwReset	EPW-30 のハードウェア・リセットを行います
18	EpwVpp	VPP の電圧を取得します
19	EpwExtension	EPW-30 の拡張機能設定を行います

デバイスタイプ

デバイスに対するリード／ライトなどの操作は、デバイスタイプを指定する必要があります。
デバイスタイプは、0 から始まる番号で、次の表の様に番号が割り当てられています。

マイクロワイヤデバイスのタイプ番号

タイプ名	値	モード
D93C06	0	16 ビット
D93C46	1	16 ビット
D93C56	2	16 ビット
D93C57	3	16 ビット
D93C66	4	16 ビット
D93C76	5	16 ビット
D93C86	6	16 ビット
D93C06B	100	8 ビット
D93C46B	101	8 ビット
D93C56B	102	8 ビット
D93C57B	103	8 ビット
D93C66B	104	8 ビット
D93C76B	105	8 ビット
D93C86B	106	8 ビット
D9010	200	16 ビット
D9020	201	16 ビット
D9040	202	16 ビット
D9080	203	16 ビット
D9016	204	16 ビット

I2C デバイスのタイプ番号

タイプ名	値
D24C00	0
D24C01	1
D24C02	2
D24C04	3
D24C16	4
D24C32	5
D24C64	6
D24C128	7
D24C256	8
D24C512	9
D24C1024	10

SPI デバイスのタイプ番号

タイプ名	値
D25C010	0
D25C020	1
D25C040	2
D25C080	4
D25C160	16
D25C320	32
D25C640	64

※マイクロワイヤデバイスは、16 ビットモードと 8 ビットモードでタイプ名とタイプ番号が異なるので注意して下さい。また、マイクロワイヤデバイス、I2C デバイス、SPI デバイスの切り替えは、オープン時に指定するか、モード設定で行う必要があります。

※D90x0 タイプのデバイスを使用する場合は、オープン時に ExtMODE=1 とする必要があります。

EPW-30 インターフェース API のオープン（EpwOpen）

■書式

BOOL EpwOpen(int MODE,int ExtMODE,int writetime,BOOL ExtPower)

■引数

MODE　0=マイクロワイヤモード
　　1=I2C モード
　　2=SPI モード

ExtMODE

0=通常デバイス

1=BR90x0 タイプのデバイス（MODE=0 の時に有効）

writetime　書き込み時間（0-100ms）

ExtPower　外部電源を使用する場合は TRUE とする

■戻り値

TRUE　オープン成功

FALSE　オープン失敗

■機能

EPW-30 インターフェース API をオープンします。

EPW-30 の接続ポートは自動で検索され、最初に見つかった EPW-30 がオープンされます。

接続している EPW-30 が、引数で指定されたモードに対応していない場合は、オープン失敗となります。

オープンに成功すると TRUE が返り、失敗すると FALSE が返ります。

BR90x0 タイプのデバイスを使用する場合は、オープン時に ExtMODE として 1 を指定する必要があります。

writetime は、最大書き込み時間です。指定範囲は、0-100 で、単位は ms となります。

ExtPower は、通常 FALSE で、 外部電源を使用する場合は TRUE とします。

EPW-30 インターフェース API のオープン（EpwClose）

■書式
VOID EpwClose(VOID)

■引数
なし

■戻り値
なし

■機能
EPW-30 インターフェース API をクローズします。

EPW-30 インターフェース API のモードのセット（EpwMode）

■書式

BOOL EpwMode (int mode,int ExtMODE,int writetime)

■引数

mode=0　　マイクロワイヤモード

mode=1　　I2C モード

mode=2　　SPI モード

ExtMODE

0=通常デバイス

1=BR90x0 タイプのデバイス（MODE=0 の時に有効）

writetime　書き込み時間（0-100ms）

■戻り値

TRUE　成功

FALSE　失敗

■機能

EPW-30 インターフェースを指定したモードに切り替えて、ポジション LED を点灯します。対応していないモードを設定すると、戻り値が FALSE となり、モードの変更は行われません。

BR90x0 タイプのデバイスを使用する場合は、オープン時に ExtMODE として 1 を指定する必要があります。

writetime は、最大書き込み時間です。指定範囲は、0-100 で、単位は ms となります。

デバイスの接続確認（EpwCheck）

■書式

int EpwCheck (VIOD)

■引数

なし

■戻り値

0	デバイスは接続されていない
それ以外*	デバイスが接続されている

*マイクロワイヤモードの場合は、識別したアドレスビット数を返します。

■機能

EPW-30 にデバイスが接続されているかどうかを確認します。一部のデバイス(BR90x0 シリーズなど)では、デバイスチェックができないものもあります。この場合は、デバイスが接続されていても、0 となります。

デバイスの読み出し（EpwRead）

■書式

BOOL EpwRead(int TYPE,int adr,BYTE *dat int len,int psize)

■引数

TYPE　デバイスのタイプ番号*
adr　　読み出しを行う開始アドレス
dat　　読み出しデータの格納先
len　　読み出しデータのサイズ（バイト数）
psize　ページサイズ
*タイプ番号は、機能説明を参照

■戻り値

TRUE　成功
FALSE　失敗

■機能

デバイスの読み出しを行います。16 ビットモードデバイスの場合は、読出しデータはリトルエンディアンで格納されます。
psize には、1 回に読み出せるページサイズをバイト数で指定します。ページサイズが不明な場合は 0 を指定してください。

インターフェースをオープンしていない場合や、エラーの場合は、FALSE が返ります。

デバイスの書き込み（EpwWrite）

■書式

BOOL EpwWrite (int TYPE,int adr, BYTE *dat int len,int psize)

■引数

TYPE　デバイスのタイプ番号*
adr　　書き込みを行う開始アドレス
dat　　書き込みの格納先
len　　書き込みデータのサイズ（バイト数）
psize　ページサイズ
*タイプ番号は、機能説明を参照

■戻り値

TRUE　成功
FALSE　失敗

■機能

デバイスの書き込みを行います。インターフェースをオープンしていない場合や、エラーの場合は、FALSE が返ります。
psize には、1 回に書き込めるページサイズをバイト数で指定します。ページサイズが不明な場合は 0 を指定してください。
16 ビットモードの場合は、書き込みデータはリトルエンディアンで格納します。

デバイスの消去(EpwErase)

■書式

BOOL EpwErase (int TYPE,int adr, BYTE *dat int len,int psize)

■引数

TYPE　デバイスのタイプ番号*

adr　　書き込みを行う開始アドレス

dat　　書き込みの格納先

len　　書き込みデータのサイズ（バイト数）

psize　ページサイズ

*タイプ番号は、機能説明を参照

■戻り値

TRUE　成功

FALSE　失敗

■機能

デバイスの消去を行います。インターフェースをオープンしていない場合や、エラーの場合は、FALSE が返ります。

psize には、1 回に書き込めるページサイズをバイト数で指定します。ページサイズが不明な場合は 0 を指定してください。

デバイスの比較(EpwCompare)

■書式

BOOL EpwCompare (int TYPE,int adr, BYTE *dat int len,int psize)

■引数

TYPE　デバイスのタイプ番号*
adr　　比較を行う開始アドレス
dat　　比較するデータの格納先
len　　比較データのサイズ（バイト数）
psize　ページサイズ
*タイプ番号は、機能説明を参照

■戻り値

TRUE　成功
FALSE　失敗

■機能

デバイスの比較を行います。
psize にはページサイズをバイト数で指定します。ページサイズが不明な場合は 0 を指定してください。
インターフェースをオープンしていない場合や、エラーの場合は、FALSE が返ります。

デバイスのチェックサムの計算（EpwChecksum）

■書式

int EpwChecksum (int TYPE,int adr,int len,int psize)

■引数

TYPE　デバイスのタイプ番号*

adr　　開始アドレス

len　　サイズ（バイト数）

psize　ページサイズ

*タイプ番号は、機能説明を参照

■戻り値

チェックサム値

■機能

指定された範囲のデータのチェックサムを計算します。16 ビットモードでもチェックサムは 8 ビットの加算になることに注意してください。またエラーの場合は、0000h が返ります。

psize にはページサイズをバイト数で指定します。ページサイズが不明な場合は 0 を指定してください。

デバイスのブランクチェック（EpwBlankChk）

■書式

BOOL EpwBlankChk (int TYPE,int adr,int len,int psize)

■引数

TYPE　デバイスのタイプ番号*

adr　　開始アドレス

len　　サイズ（バイト数）

psize　ページサイズ

*タイプ番号は、機能説明を参照

■戻り値

TRUE　デバイスはブランク

FALSE　エラー

■機能

指定された範囲のデータのブランクチェックを行います。

psize にはページサイズをバイト数で指定します。ページサイズが不明な場合は 0 を指定してください。

エラーの場合は、FALSE が返ります。

DLL バージョンの取得(EpwDllVer)

■書式

int EpwDllVer(VOID)

■引数
なし

■戻り値
バージョン番号

■機能
DLL のバージョン番号を取得します。バージョン番号は、16 進コードで、0x100 倍された値として返されます。たとえば、バージョン 1.1A の場合は、戻り値は 0x11a となります。

ファームウェアバージョンの取得(EpwFrmVer)

■書式

int EpwFrmVer(VOID)

■引数
なし

■戻り値
ファームウェアバージョン

■機能
現在接続されている EPW-30 のファームウェアバージョンを返します。
バージョン番号は 16 進コードで、0x100 倍された値として返されます。たとえば、バージョン 1.1A の場合は、戻り値は 0x11a となります。

クロック周波数の設定 (EpwSetFreq)

■書式

BOOL EpwSetFreq(int freq)

■引数

freq　　クロック周波数(Hz)

■戻り値

TRUE　成功

FALSE　エラー

■機能

クロック周波数の設定を行います。

クロック周波数は、Hz 単位です。

設定できない周波数を指定した場合は、設定値を超えない周波数で設定可能な周波数に設定されます。

また、設定可能周波数以下の周波数を設定した場合は、設定可能な最低周波数に設定されます。

設定可能な周波数範囲は、次の通りです。

- デフォルト CLK 周波数　100KHz (MCW/SPI/I2C 共通)
- クロック周波数範囲　2KHz~12MHz(MCW/SPI)/46.9KHz~12MHz(I2C)

クロック周波数の読み出し (EpwGetFreq)

■書式

int　EpwGetFreq(VOID)

■引数

なし

■戻り値

クロック周波数(Hz)

■機能

実際に設定されている周波数を読み出します。

EpwSetFreq で設定した周波数は、ハードウェアの仕様により、設定可能な周波数に変更されます。

EpwGetFreq は、実際に設定されている周波数を読み出す時に利用されます。

スイッチの読み出し(EpwGetSwitch)

■書式

int　EpwGetSwitch(VOID)

■引数

なし

■戻り値

プログラム・スイッチの状態（１：スイッチが押されている、０：スイッチが押されていない）

■機能

EPW-ISP 専用の API です。

プログラム・スイッチの状態を所得します。

スイッチ・イベントの設定(EpwSetEvent)

■書式

void　EpwSetEvent (HANDLE evt)

■引数

evt　　イベントハンドル

■戻り値

なし

■機能

EPW-ISP 専用の API です。
引数には、CreateEvent で作成したイベントハンドルを指定します。
EPW-ISP のスイッチが押されると、設定したイベントがセットされます。
イベントを無効にする場合は、evt に NULL を指定してください。

ハードウェア・リセット(EpwReset)

■書式

void　EpwReset(VOID)

■引数

なし

■戻り値

なし

■機能

EPW-30 のハードウェア・リセットを行います。

VPP 電圧の取得(EpwVpp)

■書式

int　EpwVpp(VOID)

■引数

なし

■戻り値

VPP 電圧×100

■機能

VPP の電圧を取得します。
戻り値は、VPP を 100 倍した値です。
VPP=3.3V の場合は、330 となります。
EPW-30U および EPW-30S の場合は、内部スイッチで選択された電圧を調べることができます。
EPW-ISP の場合は、VPP 端子の電圧を出力します。

データ送信(EpwDWrite)

■書式

BOOL　EpwDWrite(int mode,BYTE *dat,int len)

■引数

mode　機能モード

dat　送信データ

len　送信データサイズ

■戻り値

TRUE　送信成功

FALSE　送信エラー

■機能

ターゲットデバイスに対して、データを送信します。

データ送信は、mode ﾋﾞｯﾄの値により、CS や Start/StopCondition の制御を行うことができます。

mode bit	機能
0	データ送信前に CS/CS#を有効にする(I2C デバイスの場合は StartCondition を発行)
1	データ送信後に CS/CS#を無効にする(I2C デバイスの場合は StopConditon を発行)
2	データ送信前に RepeatStart を発行する（I2C デバイスの場合のみ）
3	データ送信後に書き込み完了処理を行う
4-31	Reserved

この API を使用することで、標準でサポートしていないデバイスにも対応することが可能になります。

len=0 とすることで、CS/CS#や Start/StopConditon のみの制御を行うことができます。

使用例）SPI デバイスの書き込みイネーブル

```
BYTE dat[1];
EpwOpen(2,0,50);        //SPI モード、ExtMODE=0、書き込み時間=50ms でオープン
dat[0]=0x06;            //Write Enable コマンド
EpwDWrite(3,dat,1);     //0x06 を CS=ON/OFF 制御付きで送信
EpwClose();             //終了
```

データ受信(EpwDRead)

■書式

BOOL　EpwDRead(int mode,BYTE *dat,int len)

■引数

mode　　機能モード

dat　　送信データ

len　　送信データサイズ

■戻り値

TRUE　送信成功

FALSE　送信エラー

■機能

ターゲットデバイスにからデータを受信します。

データ送信は、mode ﾋﾞｯﾄの値により、CS や Start/StopCondition の制御を行うことができます。

mode bit	機能
0	データ送信前に CS/CS#を有効にする(I2C デバイスの場合は StartCondition を発行)
1	データ送信後に CS/CS#を無効にする(I2C デバイスの場合は StopConditon を発行)
2	データ送信前に RepeatStart を発行する（I2C デバイスの場合のみ）
3	データ受信後に最終データの ACK 処理を行う（I2C デバイスの場合のみ）
4-31	Reserved

この API を使用することで、標準でサポートしていないデバイスにも対応することが可能になります。

len=0 とすることで、CS/CS#や Start/StopConditon のみの制御を行うことができます。

使用例）SPI デバイスのステータス読み出し

```
BYTE dat[1];
EpwOpen(2,0,50);          //SPI モード、ExtMODE=0、書き込み時間=50ms でオープン
dat[0]=0x05;              //RDSR コマンド
EpwDWrite(1,dat,1);       //CS=ON にして RDSR を送信
EpwDRead(2,dat,1);        //1 バイトのステータスを受信して CS=OFF にする
EpwClose();               //終了
```

EPW-30 DLL インターフェース仕様書

2017 年 2 月 23 日　バージョン　1.0

発行者　株式会社ソリトンウェーブ

http://www.solitonwave.co.jp

info@solitonwave.co.jp