EPX-18QC 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

パソコンの USB ポートに接続して、デジタル入出力信号を制御する「EPX-18QC API 関数」をユーザーアプリケーションから呼び出すことで簡単に **EPX-18QC** のカウンタ機能、I/O 機能を制御することができます。下図は全体の構成です。

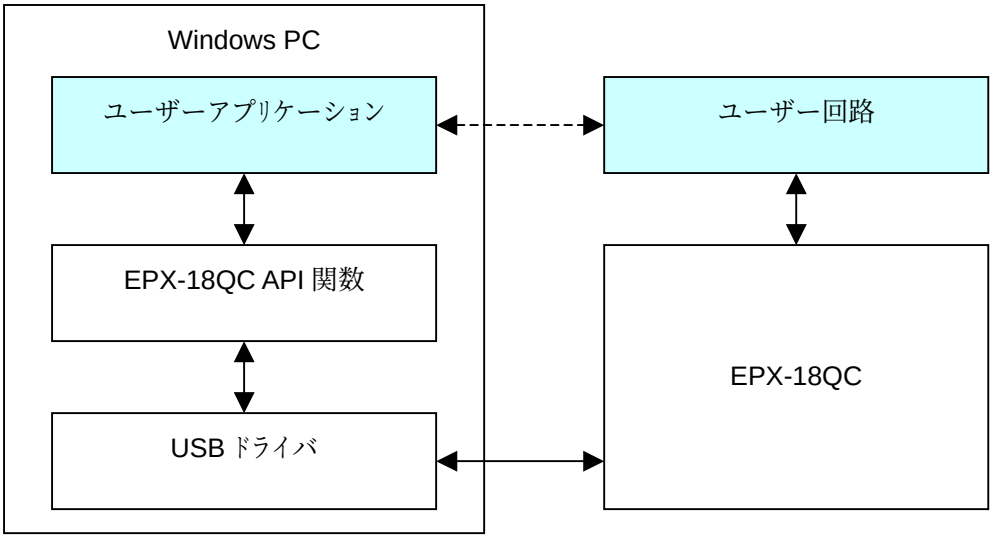

図 1.1 構成

1.2 関数について

「EPX-18QC API 関数」は関数群をモジュール化した「EPX18QC.dll」で提供されます。
「EPX18QC.dll」ファイルは **EPX-18QC** 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**、**カウンタ制御関数**、**I/O 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。
カウンタ制御関数はカウンタ機能を制御する関数です。
I/O 制御関数は I/O 機能を制御する関数です。

1.3 プログラミングの準備

・Visual C++の場合

「EPX18QC.h」、「EPX18QC.lib」ファイルをプロジェクトに追加してください。

・Visual Basic 6.0 の場合

「EPX18QC.bas」ファイルをプロジェクトの標準モジュールに追加してください。

・Visual Basic.NET の場合

「EPX18QC.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

「EPX18QC.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本製品に付属の CD-ROM「¥EPX-18QC¥library」フォルダにあります。

1.4 関数の戻り値について

関数の戻り値の説明は下表に示します。

定数	値	意味
EPX18QC_OK	0	正常終了
EPX18QC_INVALID_HANDLE	1	デバイスのハンドルが無効
EPX18QC_DEVICE_NOT_FOUND	2	デバイスが見つからない
EPX18QC_DEVICE_NOT_OPENED	3	デバイスがオープンできない
EPX18QC_OTHER_ERROR	4	その他のエラーが発生した
EPX18QC_COMMUNICATION_ERROR	5	通信エラーが発生した
EPX18QC_ INVALID _PARAMETER	6	パラメータが無効

表 1.4 関数の戻り値

1.5 注意事項

複数のアプリケーション、またはマルチスレッドによる **EPX-18QC** 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

EPX18QC_GetNumberOfDevices

現在接続されているデバイスの数を取得します。

int **EPX18QC_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 = EPX18QC_GetNumberOfDevices(&number);
if (result == EPX18QC_OK) {
	// EPX18QC_GetNumberOfDevices 成功
}
else {
	// EPX18QC_GetNumberOfDevices 失敗
}
```

EPX18QC_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 **EPX18QC_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 **EPX18QC_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基板記載のシリアル番号と同一です。

Example

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

index = 0;
result = EPX18QC_GetSerialNumber(index, &serialNumber);
if (result == EPX18QC_OK) {
	// EPX18QC_GetSerialNumber 成功
}
else {
	// EPX18QC_GetSerialNumber 失敗
}
```

EPX-18QC_Open

デバイスをオープンし、デバイスのハンドルを取得します。

Int **EPX18QC_Open** (EPX18QC_HANDLE **Handle*)

Parameters

Handle　　デバイスのハンドル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ハンドルを取得します。
取得したハンドルは以後、カウンタ及び I/O 制御関数等に引数として渡すことになります。

Example

```
int result;
EPX18QC_HANDLE handle;

result = EPX18QC_Open(&handle);
if (result == EPX18QC_OK) {
	// EPX18QC_Open 成功
}
else {
	// EPX18QC_Open 失敗
}
```

EPX18QC_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、デバイスのハンドルを取得します。

int **EPX18QC_OpenBySerialNumber** (int *SerialNumber*, EPX18QC_HANDLE **Handle*)

Parameters

SerialNumber　デバイスのｼﾘｱﾙ番号

Handle　デバイスのハンドルの格納先へのポインタ

Remarks

ｼﾘｱﾙ番号は **EPX18QC_GetSerialNumber** で取得することができます。

尚、このｼﾘｱﾙ番号は基板記載のｼﾘｱﾙ番号と同一です。

取得したﾊﾝﾄﾞﾙは以後、ｶｳﾝﾀ及び I/O 制御関数等に引数として渡すことになります。

Example

```
int result;
EPX18QC_HANDLE handle;
int serialNumber;    // 取得したデバイスのｼﾘｱﾙ番号

result = EPX18QC_OpenBySerialNumber(serialNumber, &handle);
if (result == EPX18QC_OK) {
       // EPX18QC_OpenBySerialNumber 成功
}
else {
       // EPX18QC_OpenBySerialNumber 失敗
}
```

EPX18QC_Close

デバイスをクローズします。

Int **EPX18QC_Close** (EPX18QC_HANDLE *Handle*)

Parameters

Handle　　デバイスのハンドル

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル

result = EPX18QC_Close(handle);
if (result == EPX18QC_OK) {
       // EPX18QC_Close 成功
}
else {
       // EPX18QC_Close 失敗
}
```

2.2 カウンタ制御関数

EPX18QC_SetCounterMode

カウンタの動作モードを設定します。

Int **EPX18QC_SetCounterMode** (EPX18QC_HANDLE *Handle*, BYTE *CH*, BYTE *Mode*)

Parameters

Handle　　デバイスのハンドル
CH　　カウンタの指定
Mode　　カウンタ動作モード

Remarks

Mode	説明
0	単相パルス入力モード　（初期値）
1	2 相パルス入力（位相差パルス入力）モード

単相パルス入力、2 相パルス入力については「**EPX-18QC 取扱説明書**」をご参照ください。

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE ch, mode;

ch = 0;       // CH0
mode = 0;     // 単相パルス入力モード
result = EPX18QC_SetCounterMode(handle, ch, mode);
if (result == EPX18QC_OK) {
      // EPX18QC_SetCounterMode 成功
}
else {
      // EPX18QC_SetCounterMode 失敗
}
```

EPX18QC_GetCounterMode

カウンタの動作モードを取得します。

Int **EPX18QC_GetCounterMode** (EPX18QC_HANDLE *Handle*, BYTE *CH*, BYTE **Mode*)

Parameters

Handle	デバイスのハンドル
CH	カウンタの指定
Mode	カウンタ動作モードの格納先へのポインタ

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE ch, mode;

ch = 0;       // CH0
result = EPX18QC_GetCounterMode(handle, ch, &mode);
if (result == EPX18QC_OK) {
       // EPX18QC_GetCounterMode 成功
}
else {
       // EPX18QC_GetCounterMode 失敗
}
```

EPX18QC_SetCounterDirection

カウンタのカウント方向を設定します。

Int **EPX18QC_SetCounterDirection** (EPX18QC_HANDLE *Handle*, BYTE *CH*, BYTE *Direction*)

Parameters

Handle　　デバイスのハンドル
CH　　カウンタの指定
Direction　　カウント方向

Remarks

Direction	説明
0	カウント方向はアップカウント　（初期値）
1	カウント方向はダウンカウント

2 相パルス入力時のカウント方向については「**EPX-18QC 取扱説明書**」をご参照ください。

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE ch, dir;

ch = 0;       // CH0
dir = 0;      // カウント方向はアップカウント
result = EPX18QC_SetCounterDirection(handle, ch, dir);
if (result == EPX18QC_OK) {
      // EPX18QC_SetCounterDirection 成功
}
else {
      // EPX18QC_SetCounterDirection 失敗
}
```

EPX18QC_GetCounterDirection

カウンタのカウント方向を取得します。

Int **EPX18QC_GetCounterDirection** (EPX18QC_HANDLE *Handle*, BYTE *CH*,
BYTE **Direction*)

Parameters

Handle	デバイスのハンドル
CH	カウンタの指定
Direction	カウント方向の格納先へのポインタ

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE ch, dir;

ch = 0;       // CH0
result = EPX18QC_GetCounterDirection(handle, ch, &dir);
if (result == EPX18QC_OK) {
       // EPX18QC_GetCounterDirection 成功
}
else {
       // EPX18QC_GetCounterDirection 失敗
}
```

EPX18QC_SetCounterControl

カウンタの動作を設定します。

int **EPX18QC_SetCounterControl** (EPX18QC_HANDLE *Handle*, BYTE *CH*, BYTE *Control*)

Parameters

Handle　　デバイスのハンドル
CH　　カウンタの指定
Control　　カウンタ動作

Remarks

Control	説明
0	カウンタのカウント動作は停止　（初期値）
1	カウンタはカウント動作

外部からのカウンタ制御入力信号（$\overline{\text{EX\_CTEN}}$ ）でも制御が可能です。
カウンタ制御入力信号については「**EPX-18QC 取扱説明書**」をご参照ください。

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE ch, control;

ch = 0;       // CH0
control = 1;  // カウント動作
result = EPX18QC_SetCounterControl(handle, ch, control);
if (result == EPX18QC_OK) {
       // EPX18QC_SetCounterControl 成功
}
else {
       // EPX18QC_SetCounterControl 失敗
}
```

EPX18QC_GetCounterControl

カウンタの動作状態を取得します。

int **EPX18QC_GetCounterControl** (EPX18QC_HANDLE *Handle*, BYTE *CH*, BYTE **Control*)

Parameters

Handle	デバイスのハンドル
CH	カウンタの指定
Control	カウンタ動作状態の格納先へのポインタ

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE ch, control;

ch = 0;       // CH0
result = EPX18QC_GetCounterControl(handle, ch, &control);
if (result == EPX18QC_OK) {
       // EPX18QC_GetCounterControl 成功
}
else {
       // EPX18QC_GetCounterControl 失敗
}
```

EPX18QC_ResetCounter

カウンタをﾘｾｯﾄ(0 ｸﾘｱ)します。

Int **EPX18QC_ResetCounter** (EPX18QC_HANDLE *Handle*, BYTE *CH*)

Parameters

Handle　　デバイスのハンドル

CH　　カウンタの指定

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE ch;

ch = 0;       // CH0
result = EPX18QC_ResetCounter(handle, ch);
if (result == EPX18QC_OK) {
       // EPX18QC_ResetCounter 成功
}
else {
       // EPX18QC_ResetCounter 失敗
}
```

EPX18QC_GetCounterValue

カウンタの値を取得します。

int **EPX18QC_GetCounterValue** (EPX18QC_HANDLE *Handle*, BYTE *CH*, int **Value*)

Parameters

Handle	デバイスのハンドル
CH	カウンタの指定
Value	カウンタ値の格納先へのポインタ

Remarks

カウンタ値は符号付き 24 ビット（-8,388,608 ~ 8,388,607）です。

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE ch;
int value;

ch = 0;       // CH0
result = EPX18QC_GetCounterValue(handle, ch, &value);
if (result == EPX18QC_OK) {
       // EPX18QC_GetCounterValue 成功
}
else {
       // EPX18QC_GetCounterValue 失敗
}
```

EPX18QC_GetCounterStatus

カウンタの状態を取得します。

int **EPX18QC_GetCounterStatus** (EPX18QC_HANDLE *Handle*, BYTE *CH*, BYTE **Status*)

Parameters

Handle	デバイスのハンドル
CH	カウンタの指定
Status	カウンタ動作状態の格納先へのポインタ

Remarks

Status	説明
0	カウンタは正常状態
1	カウンタはオーバーフロー（カウンタ停止状態）
2	カウンタはアンダーフロー（カウンタ停止状態）

オーバーフロー（カウンタ値：0x7FFFFF→0x800000）またはアンダーフロー（カウンタ値：0x800000→0x7FFFFF）が発生すると、カウンタは自動的に停止します。
再度、カウンタを動作させるには **EPX18QC_ResetCounter** でカウンタをリセットしてください。

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE ch, status;

ch = 0;       // CH0
result = EPX18QC_GetCounterStatus(handle, ch, &status);
if (result == EPX18QC_OK) {
       // EPX18QC_GetCounterStatus 成功
}
else {
       // EPX18QC_GetCounterStatus 失敗
}
```

EPX18QC_GetCounterExControl

外部からのカウンタ制御入力信号（$\overline{\text{EX\_CTEN}}$ ）の状態を取得します。

int **EPX18QC_GetCounterExControl** (EPX18QC_HANDLE *Handle*, BYTE *CH*,
BYTE * *Control*)

Parameters

Handle	デバイスのハンドル
CH	カウンタの指定
Control	カウンタ動作状態の格納先へのポインタ

Remarks

Control	説明
0	カウンタのカウント動作有効
1	カウンタはカウント動作無効

外部からのカウンタ制御入力信号（$\overline{\text{EX\_CTEN}}$ ）の状態を取得します。
カウンタ制御入力信号については「**EPX-18QC 取扱説明書**」をご参照ください。

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE ch, control;

ch = 0;       // CH0
result = EPX18QC_GetCounterExControl(handle, ch, &control);
if (result == EPX18QC_OK) {
       // EPX18QC_GetCounterExControl 成功
}
else {
       // EPX18QC_GetCounterExControl 失敗
}
```

EPX18QC_GetCounterLatchFlag

カウンタラッチ有効データの状態を取得します。

int **EPX18QC_GetCounterLatchFlag** (EPX18QC_HANDLE *Handle*, BYTE *CH*, BYTE **Flag*)

Parameters

Handle	デバイスのハンドル
CH	カウンタの指定
Flag	カウンタラッチ状態（有効データ有無）の格納先へのポインタ

Remarks

Flag	説明
0	カウンタラッチ有効データ無
1	カウンタラッチ有効データ有

外部からのカウンタラッチ入力信号（$\overline{\text{EX\_CTEN}}$ ）による有効データの有無状態を取得します。
カウンタ制御入力信号については「**EPX-18QC 取扱説明書**」をご参照ください。

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE ch, flag;

ch = 0;       // CH0
result = EPX18QC_GetCounterLatchFlag(handle, ch, &flag);
if (result == EPX18QC_OK) {
       // EPX18QC_GetCounterLatchFlag 成功
}
else {
       // EPX18QC_GetCounterLatchFlag 失敗
}
```

EPX18QC_GetCounterLatchValue

カウンタラッチの値を取得します。

int **EPX18QC_GetCounterLatchValue** (EPX18QC_HANDLE *Handle*, BYTE *CH*, int **Value*)

Parameters

Handle	デバイスのハンドル
CH	カウンタの指定
Value	カウンタラッチ値の格納先へのポインタ

Remarks

カウンタラッチ値は符号付き 24 ビット（-8,388,608 ~ 8,388,607）です。

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE ch;
int value;

ch = 0;       // CH0
result = EPX18QC_GetCounterLatchValue(handle, ch, &value);
if (result == EPX18QC_OK) {
       // EPX18QC_GetCounterLatchValue 成功
}
else {
       // EPX18QC_GetCounterLatchValue 失敗
}
```

2.3 I/O 制御関数

EPX18QC_SetPortDirection

ポートの入出力方向を設定します。

Int **EPX18QC_SetPortDirection** (EPX18QC_HANDLE *Handle*, BYTE *Port*, BYTE *Direction*)

Parameters

Handle　　デバイスのハンドル
Port　　ポートの指定
Direction　　ポートの入出力方向

Remarks

Port に入出力設定するポート番号を指定します。

Direction	説明
0	入力　（初期値）
1	出力

初期値（電源投入時）は全てのポートが入力に設定されています。

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE port, dir;

port = 0;     // Port0
dir = 1;      // 出力設定
result = EPX18QC_SetPortDirection(handle, port, dir);
if (result == EPX18QC_OK) {
       // EPX18QC_SetPortDirection 成功
}
else {
       // EPX18QC_SetPortDirection 失敗
}
```

EPX18QC_GetPortDirection

現在設定されているポートの入出力方向を取得します。

Int **EPX18QC_GetPortDirection** (EPX18QC_HANDLE *Handle*, BYTE *Port*, BYTE **Direction*)

Parameters

Handle	デバイスのハンドル
Port	ポートの指定
Direction	ポートの入出力方向の格納先へのポインタ

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE port, dir;

port = 2;     // Port2
result = EPX18QC_GetPortDirection(handle, port, &dir);
if (result == EPX18QC_OK) {
       // EPX18QC_GetPortDirection 成功
}
else {
       // EPX18QC_GetPortDirection 失敗
}
```

EPX18QC_OutputPort

ポートに出力します。

Int **EPX18QC_OutputPort** (EPX18QC_HANDLE *Handle*, BYTE *Port*, BYTE *Value*)

Parameters

Handle	デバイスのハンドル
Port	ポートの指定
Value	ポート出力値

Remarks

Port に出力するポート番号を指定します。ポートが入力に設定されている場合は出力されません。
Value の各ビットがポートの各ピンに対応しています。
ビットの値が **0** の場合は **LOW** レベル、**1** の場合は **HIGH** レベルになります。

例) *Port* = **1**, *Value* = **0xA5**

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Value	1	0	1	0	0	1	0	1
PIN	P17	P16	P15	P14	P13	P12	P11	P10
LEVEL	H	L	H	L	L	H	L	H

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE port, value;

port = 1;     // Port1
value = 0xA5; // P17,15,12,10 “HIGH”  P16,14,13,11 “LOW”
result = EPX18QC_OutputPort(handle, port, value);
if (result == EPX18QC_OK) {
       // EPX18QC_OutputPort 成功
}
else {
       // EPX18QC_OutputPort 失敗
}
```

EPX18QC_InputPort

ポートから入力します。

int **EPX18QC_InputPort** (EPX18QC_HANDLE *Handle*, BYTE *Port*, BYTE **Value*)

Parameters

Handle	デバイスのハンドル
Port	ポートの指定
Value	ポート入力値の格納先へのポインタ

Remarks

Port に入力するポート番号を指定します。

Value の各ビットがポートの各ピンに対応しています。

ビットの値が **0** の場合は **LOW** レベル、**1** の場合は **HIGH** レベルになります。

例) *Port* = **0**, *Value* = **0x3C**

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Value	0	0	1	1	1	1	0	0
PIN	P07	P06	P05	P04	P03	P02	P01	P00
LEVEL	L	L	H	H	H	H	L	L

Example

```
int result;
EPX18QC_HANDLE handle;      // オープンしたデバイスのハンドル
BYTE port, value;

port = 0;     // Port0
result = EPX18QC_InputPort(handle, port, &value);
if (result == EPX18QC_OK) {
       // EPX18QC_InputPort 成功
}
else {
       // EPX18QC_InputPort 失敗
}
```