MAI-2088 プログラミングガイド

Rev 2.3

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、「MAI-2088 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 **MAI-2088** の 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。

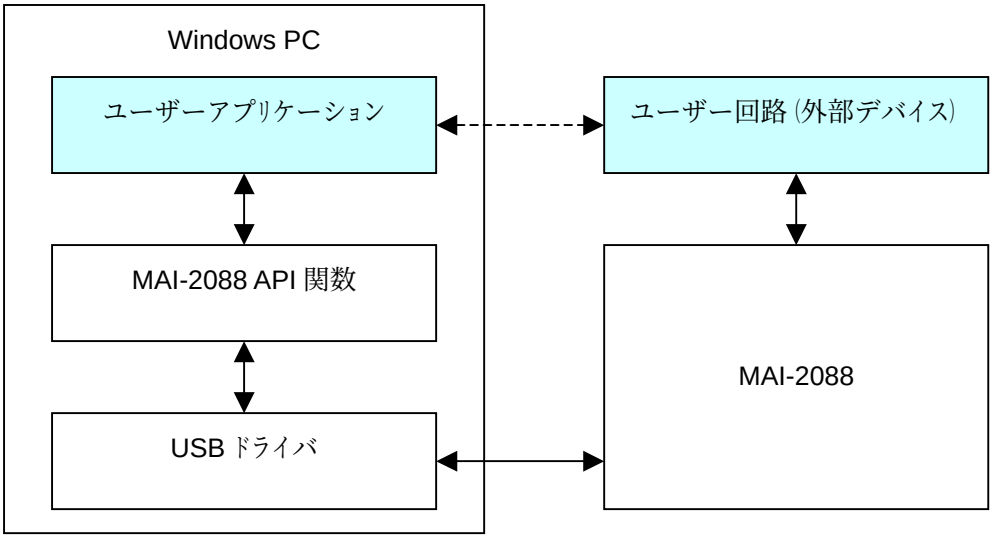


図 1.1 構成

1.2 関数について

「MAI-2088 API 関数」は関数群をモジュール化した「MAI2088.dll」で提供されます。
「MAI2088.dll」ファイルは **MAI-2088** 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**、**初期設定関数**、**I/O 制御関数**、**外部デバイス制御関数**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。
初期設定関数はデバイスを初期化する関数です。
I/O 制御関数は I/O 操作の基本となる入出力関数です。
外部デバイス制御関数はエンコーダ、LED 等を制御する関数です。
外部デバイスとの接続については「**MAI-2088 取扱説明書**」をご参照ください。

1.3 プログラミングの準備

・Visual C++の場合

「MAI2088.h」、「MAI2088.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

「MAI2088.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

「MAI2088.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

「MAI2088.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「¥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
MAI2088_OK	0	正常終了
MAI2088_INVALID_HANDLE	1	デバイスのハンドルが無効
MAI2088_DEVICE_NOT_FOUND	2	デバイスが見つからない
MAI2088_DEVICE_NOT_OPENED	3	デバイスがオープンできない
MAI2088_OTHER_ERROR	4	その他のエラーが発生した
MAI2088_COMMUNICATION_ERROR	5	通信エラーが発生した
MAI2088_OUT_OF_RANGE_PARAMETER	6	パラメータが範囲を超えた

表 1.3 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 **MAI-2088** 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

MAI2088_getNumberOfDevices

現在接続されているデバイスの数を取得します。

int **MAI2088_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 = MAI2088_getNumberOfDevices(&number);
if (result == MAI2088_OK) {
	// MAI2088_getNumberOfDevices 成功
}
else {
	// MAI2088_getNumberOfDevices 失敗
}
```

MAI2088_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 **MAI2088_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 **MAI2088_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裏面のシリアル番号と同一です。

Example

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

index = 0;
result = MAI2088_getSerialNumber(index, &serialNumber);
if (result == MAI2088_OK) {
	// MAI2088_getSerialNumber 成功
}
else {
	// MAI2088_getSerialNumber 失敗
}
```

MAI2088_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 **MAI2088_open** (MAI2088_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、制御関数等に引数として渡すことになります。

Example

```
int result;
MAI2088_HANDLE handle;

result = MAI2088_open(&handle);
if (result == MAI2088_OK) {
	// MAI2088_open 成功
}
else {
	// MAI2088_open 失敗
}
```

MAI2088_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 **MAI2088_openBySerialNumber** (int *serialNumber*, MAI2088_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 **MAI2088_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裏面のｼﾘｱﾙ番号と同一です。

取得したﾊﾝﾄﾞﾙは以後、制御関数等に引数として渡すことになります。

Example

```
int result;
MAI2088_HANDLE handle;
int serialNumber;    // 取得したデバイスのｼﾘｱﾙ番号

result = MAI2088_openBySerialNumber(serialNumber, &handle);
if (result == MAI2088_OK) {
       // MAI2088_openBySerialNumber 成功
}
else {
       // MAI2088_openBySerialNumber 失敗
}
```

MAI2088_close

デバイスをクローズします。

int **MAI2088_close** (MAI2088_HANDLE *handle*)

Parameters

handle　　デバイスのハンドル

Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル

result = MAI2088_close(handle);
if (result == MAI2088_OK) {
       // MAI2088_close 成功
}
else {
       // MAI2088_close 失敗
}
```

2.2 初期設定関数

MAI2088_initialize

デバイスを初期化し、電源投入時の状態に戻します。

int **MAI2088_initialize** (MAI2088_HANDLE *handle*)

Parameters

handle　　デバイスのハンドル

Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル

result = MAI2088_initialize(handle);
if (result == MAI2088_OK) {
       // MAI2088_initialize 成功
}
else {
       // MAI2088_initialize 失敗
}
```

2.3 I/O 制御関数

MAI2088_getAnalogInputValue

アナログ入力値を取得します。

int **MAI2088_getAnalogInputValue** (MAI2088_HANDLE *handle*, int *index*, int **value*)

Parameters

handle	デバイスのハンドル
index	0 から始まるアナログ入力 ch のインデックス
value	アナログ入力値の格納先へのポインタ

Remarks

インデックスの範囲は **0 ~ 1** です。
アナログ入力値の範囲は **0 ~ 1023** です。

Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル
int index;
int value;

index = 0;    // 入力 ch0
result = MAI2088_getAnalogInputValue(handle, index, &value);
if (result == MAI2088_OK) {
       // MAI2088_getAnalogInputValue 成功
}
else {
       // MAI2088_getAnalogInputValue 失敗
}
```

MAI2088_getDigitalInputState

デジタル入力状態を取得します。

int **MAI2088_getDigitalInputState** (MAI2088_HANDLE *handle*, int *index*, int **state*)

Parameters

handle	デバイスのハンドル
index	0 から始まるデジタル入力 ch のインデックス
state	デジタル入力状態の格納先へのポインタ

Remarks

インデックスの範囲は **0 ~ 7** です。

デジタル入力状態は MAI2088_LOW か **MAI2088_HIGH** です。

Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル
int index;
int state;

index = 0;    // 入力 ch0
result = MAI2088_getDigaitalInputState(handle, index, &state);
if (result == MAI2088_OK) {
       // MAI2088_getDigaitalInputState 成功
}
else {
       // MAI2088_getDigaitalInputState 失敗
}
```

MAI2088_getDigitalOutputState

デジタル出力状態を取得します。

int **MAI2088_getDigitalOutputState** (MAI2088_HANDLE *handle*, int *index*, int **state*)

Parameters

handle	デバイスのハンドル
index	0 から始まるデジタル出力 ch のインデックス
state	デジタル出力状態の格納先へのポインタ

Remarks

インデックスの範囲は **0 ~ 7 です。**

デジタル出力状態は **MAI2088_LOW** か **MAI2088_HIGH** です。

Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル
int index;
int state;

index = 0;    // 出力 ch0
result = MAI2088_getDigaitalOutputState(handle, index, &state);
if (result == MAI2088_OK) {
       // MAI2088_getDigaitalOutputState 成功
}
else {
       // MAI2088_getDigaitalOutputState 失敗
}
```

MAI2088_setDigitalOutputState

デジタル出力状態を設定します。

int **MAI2088_setDigitalOutputState** (MAI2088_HANDLE *handle*, int *index*, int *state*)

Parameters

handle	デバイスのハンドル
index	0 から始まるデジタル出力 ch のインデックス
state	デジタル出力状態

Remarks

インデックスの範囲は **0 ~ 7** です。
デジタル出力状態は **MAI2088_LOW** か **MAI2088_HIGH** です。

Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル
int index;
int state;

index = 0;    // 出力 ch0
state = MAI2088_HIGH;       // 出力状態 High
result = MAI2088_setDigaitalOutputState(handle, index, state);
if (result == MAI2088_OK) {
       // MAI2088_setDigaitalOutputState 成功
}
else {
       // MAI2088_setDigaitalOutputState 失敗
}
```

2.4 外部デバイス制御関数

MAI2088_getPotentiometerValue

ポテンショメータの値を取得します。

int **MAI2088_getPotentiometerValue** (MAI2088_HANDLE *handle*, int *index*, int **value*)

Parameters

handle	デバイスのハンドル
index	0 から始まるポテンショメータ ch のインデックス
value	ポテンショメータ入力値の格納先へのポインタ

Remarks

インデックスの範囲は **0 ~ 1** です。
ポテンショメータ入力値は **0 ~ 1023** です。
ポテンショメータの接続については「**MAI-2088 取扱説明書**」をご参照ください。

Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル
int index;
int value;

index = 0;    // ポテンショメータ ch0
result = MAI2088_getPotentiometerValue(handle, index, &value);
if (result == MAI2088_OK) {
       // MAI2088_getPotentiometerValue 成功
}
else {
       // MAI2088_getPotentiometerValue 失敗
}
```

MAI2088_setEncoderSensitivity

エンコーダの感度を設定します。

int **MAI2088_setEncoderSensitivity** (MAI2088_HANDLE *handle*, int *index*, int *level*)

Parameters

handle	デバイスのハンドル
index	0 から始まるエンコーダ ch のインデックス
level	エンコーダの感度レベル

Remarks

インデックスの範囲は **0 ~ 3** です。
エンコーダの感度レベルは **MAI2088_ENCODER_SENSITIVITY_LEVEL1 ~ 8** です。
デフォルトでは **MAI2088_ENCODER_SENSITIVITY_LEVEL5** が設定されています。
エンコーダの接続については「**MAI-2088 取扱説明書**」をご参照ください。

Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル
int index;
int level;

index = 0;    // エンコーダ ch0
level = MAI2088_ENCODER_SENSITIVITY_LEVEL8;      // 感度レベル 8
result = MAI2088_setEncoderSensitivity(handle, index, level);
if (result == MAI2088_OK) {
       // MAI2088_ setEncoderSensitivity 成功
}
else {
       // MAI2088_ setEncoderSensitivity 失敗
}
```

MAI2088_getEncoderPulse

エンコーダのパルス数を取得します。

int **MAI2088_getEncoderPulse** (MAI2088_HANDLE *handle*, int *index*, int **pulseCount*)

Parameters

handle	デバイスのハンドル
index	0 から始まるエンコーダ ch のインデックス
pulseCount	エンコーダパルス数の格納先へのポインタ

Remarks

インデックスの範囲は **0 ~ 3** です。
エンコーダパルス数の範囲は**-32768 ~ 32767** です。前回この関数で取得してからの相対パルス数になります。
エンコーダの接続については「**MAI-2088 取扱説明書**」をご参照ください。

Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル
static int totalPulseCount = 0;    // 絶対パルス数
int index;
int pulseCount;

index = 0;    // エンコーダ ch0
result = MAI2088_getEncoderPulse(handle, index, &pulseCount);
if (result == MAI2088_OK) {
       // MAI2088_getEncoderPulse 成功
       totalPulseCount += pulseCount;
}
else {
       // MAI2088_getEncoderPulse 失敗
}
```

MAI2088_getSwitchState

スイッチの状態を取得します。

int **MAI2088_getSwitchState** (MAI2088_HANDLE *handle*, int *index*, int **state*)

Parameters

handle	デバイスのハンドル
index	0 から始まるスイッチ ch のインデックス
state	スイッチ状態の格納先へのポインタ

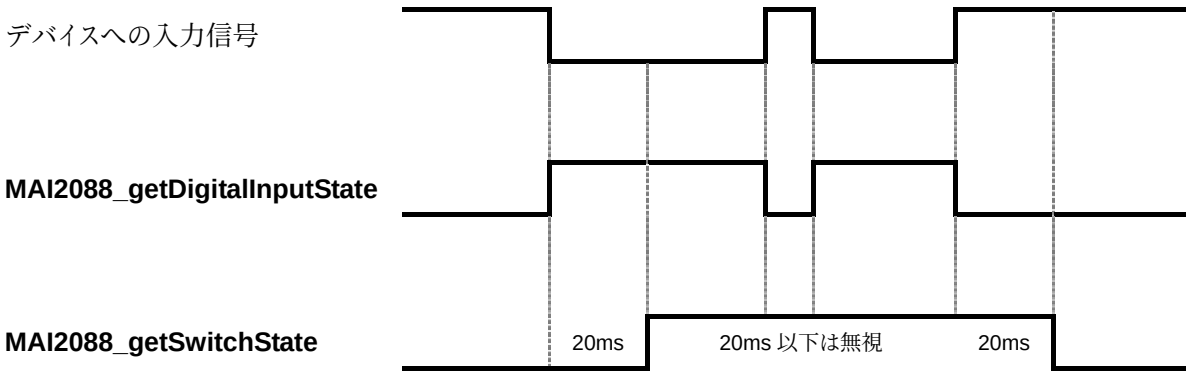
Remarks

インデックスの範囲は **0 ~ 7** です。

スイッチ状態は **MAI2088_OFF** か **MAI2088_ON** です。

スイッチの接続については「**MAI-2088 取扱説明書**」をご参照ください。

MAI2088_getDigitalInputState 関数との違いは下図のようになっています。



Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル
int index;
int state;

index = 0;    // スイッチ ch0
result = MAI2088_getSwitchState(handle, index, &state);
if (result == MAI2088_OK) {
       // MAI2088_getSwitchState 成功
}
else {
       // MAI2088_getSwitchState 失敗
}
```

MAI2088_setLEDState

LED を点灯（輝度 100%）または消灯します。

int **MAI2088_setLEDState** (MAI2088_HANDLE *handle*, int *index*, int *state*)

Parameters

handle	デバイスのハンドル
index	0 から始まる LED ch のインデックス
state	LED 状態

Remarks

インデックスの範囲は **0 ~ 7** です。

LED 状態は **MAI2088_OFF** か **MAI2088_ON** です。

LED の接続については「**MAI-2088 取扱説明書**」をご参照ください。

Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル
int index;
int state;

index = 0;    // LED ch0
state = MAI2088_ON;  // LED 点灯
result = MAI2088_setLEDState(handle, index, state);
if (result == MAI2088_OK) {
       // MAI2088_setLEDState 成功
}
else {
       // MAI2088_setLEDState 失敗
}
```

MAI2088_setLEDBrightness

LED を指定した明るさ(輝度)で点灯させます。

int **MAI2088_setLEDBrightness** (MAI2088_HANDLE *handle*, int *index*, int *brightness*)

Parameters

handle	デバイスのハンドル
index	0 から始まる LED ch のインデックス
brightness	LED 輝度

Remarks

インデックスの範囲は 0~3 です。

LED 輝度の範囲は 0~100 (%) で 5 (%) 刻みでの切り捨てになります。(例:74%→70%)

LED の接続については「**MAI-2088 取扱説明書**」をご参照ください。

Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル
int index;
int brightness;

index = 0;    // LED ch0
brightness = 80;     // 輝度 80%
result = MAI2088_setLEDBrightness(handle, index, brightness);
if (result == MAI2088_OK) {
       // MAI2088_setLEDBrightness 成功
}
else {
       // MAI2088_setLEDBrightness 失敗
}
```

MAI2088_setLEDBlink

LED を点滅させます。

int **MAI2088_setLEDBlink** (MAI2088_HANDLE *handle*, int *index*, double *frequency*,
int *dutyCycle*, int *blinkCount*)

Parameters

handle	デバイスのハンドル
index	0 から始まる LED ch のインデックス
frequency	点滅周波数
dutyCycle	点灯デューティ比
blinkCount	点滅回数

Remarks

インデックスの範囲は 0~3 です。
点滅周波数の範囲は 0.1~10 (Hz) です (小数点以下 1 桁有効)。
点灯デューティ比の範囲は 0~100 (%) です。
点滅回数の範囲は 0~65535 (**回**) です。
点滅周波数、または点灯デューティ比を 0 にすると消灯 (点滅停止) します。
点滅回数を 0 にすると点滅回数を無限大とし、連続で点滅します。
LED の接続については「**MAI-2088 取扱説明書**」をご参照ください。

Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル
int index;
double frequency;
int dutyCycle, blinkCount;

index = 0;    // LED ch0
frequency = 1.0;     // 点滅周波数 1Hz
dutyCycle = 40;      // 点灯デューティ比 40%
blinkCount = 100;    // 点滅回数 100 回
result = MAI2088_setLEDBlink(handle, index, frequency, dutyCycle, blinkCount);
if (result == MAI2088_OK) {
       // MAI2088_setLEDBlink 成功 (LED が 400ms 点灯 600ms 消灯を 100 回繰り返します)
}
else {
       // MAI2088_setLEDBlink 失敗
}
```

MAI2088_set7SegmentLED

7 セグメント LED に指定した文字を表示します。

int **MAI2088_set7SegmentLED** (MAI2088_HANDLE *handle*, int activeT*ype*,
char *displayCharactor*, int *dotPoint*)

Parameters

handle	デバイスのハンドル
activeType	7 セグメント LED タイプ（LOW アクティブまたは HIGH アクティブ）
displayCharactor	表示値
dotPoint	ドットポイントの表示

Remarks

7 セグメント LED タイプは **MAI2088_LOW** か **MAI2088_HIGH** です。
アノードコモンの 7 セグメント LED **を接続している**場合は **MAI2088_LOW**、カソードコモンの場合は **MAI2088_HIGH** を指定します。
表示値の範囲はアスキー文字の **0 ~ 9**、**A ~ Z**、**-**、**=**、_、**~**です。これら以外を指定すると表示がクリアされます。
ドットポイントの表示は **MAI2088_OFF** か **MAI2088_ON** です。
7 セグメント LED の接続、表示値については「**MAI-2088 取扱説明書**」をご参照ください。

Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル
int activeType;
char dispChar;
int dotPoint;

activeType = MAI2088_HIGH;  // High アクティブ（カソードコモン）
dispChar = '5';             // 表示文字'5'
dotPoint = MAI2088_OFF;     // ドットポイント消灯
result = MAI2088_set7SegmentLED(handle, activeType, dispChar, dotPoint);
if (result == MAI2088_OK) {
       // MAI2088_set7SegmentLED 成功
}
else {
       // MAI2088_set7SegmentLED 失敗
}
```

2.5 その他の関数

MAI2088_getVersion

デバイスのバージョン情報を取得します。

int **MAI2088_getVersion** (MAI2088_HANDLE *handle*, int **version*)

Parameters

handle	デバイスのハンドル
version	バージョン情報の格納先へのポインタ

Example

```
int result;
MAI2088_HANDLE handle;      // オープンしたデバイスのハンドル
int version;

result = MAI2088_getVersion(handle, &version);
if (result == MAI2088_OK) {
       // MAI2088_getVersion 成功
}
else {
       // MAI2088_getVersion 失敗
}
```