USB A/D ボード

RAI シリーズ

RAI-12 / RAI-16

プログラミングガイド

Rev 2.1

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、**RAI シリーズ**専用の 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 **RAI シリーズ**の A/D 機能を制御することができます。
下図は全体の構成です。

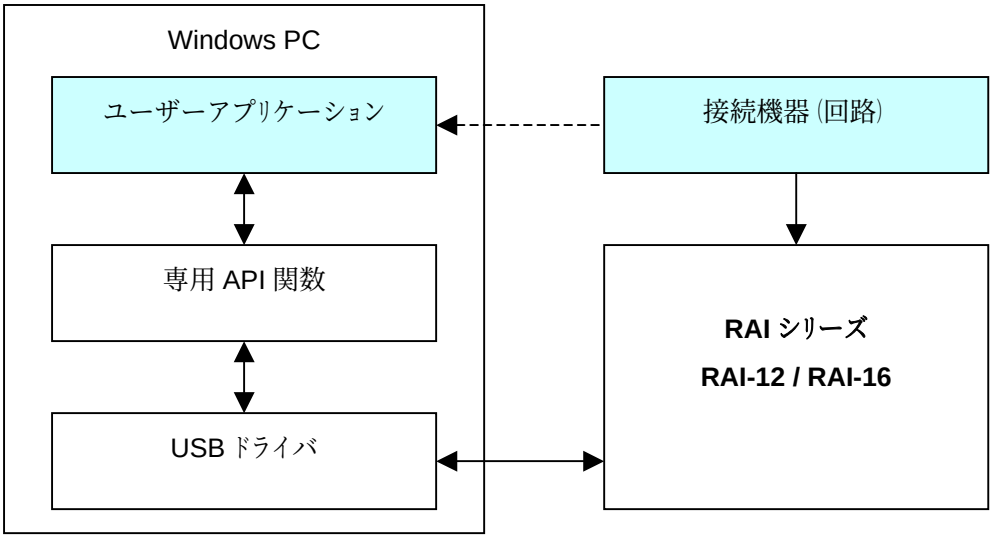

図 1.1 構成

1.2 関数について

RAI シリーズ専用 API 関数は関数群をモジュール化した「RAIxx.dll」で提供されます。
「RAIxx.dll」ファイルは **RAI シリーズ**をインストールするときにシステムフォルダに入ります。

関数は**デバイス関数**、**A/D 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。
A/D 制御関数はアナログ電圧値を取得する関数です。

1.3 プログラミングの準備

・Visual C++の場合

「RAIxx.h」、「RAIxx.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

「RAIxx.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

「RAIxx.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

「RAIxx.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
RAI_OK	0	正常終了
RAI _INVALID_ID	1	デバイスの ID が無効
RAI _DEVICE_NOT_FOUND	2	デバイスが見つからない
RAI _DEVICE_NOT_OPENED	3	デバイスがオープンできない
RAI _OTHER_ERROR	4	その他のエラーが発生した
RAI _COMMUNICATION_ERROR	5	通信エラーが発生した
RAI _INVALID_PARAMETER	6	パラメータが無効
RAI _MEMORY_ERROR	7	メモリが不足している

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同じ基板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

RAI_GetNumberOfDevices

現在接続されているデバイスの数を取得します。

int **RAI_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 = RAI_GetNumberOfDevices(&number);
if (result == RAI_OK) {
	// RAI_GetNumberOfDevices 成功
}
else {
	// RAI_GetNumberOfDevices 失敗
}
```

RAI_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 **RAI_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 **RAI_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本体記載のシリアル番号と同一です。

Example

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

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

RAI_Open

デバイスをオープンし、デバイスの ID を取得します。

int **RAI_Open** (BYTE **ID*)

Parameters

ID	デバイスの ID 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 ID を取得します。
取得した ID は以後、A/D 制御関数等に引数として渡すことになります。

Example

```
int result;
BYTE id;

result = RAI_Open(&id);
if (result == RAI_OK) {
	// RAI_Open 成功
}
else {
	// RAI_Open 失敗
}
```

RAI_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 ID を取得します。

int **RAI_OpenBySerialNumber** (int *SerialNumber*, BYTE **ID*)

Parameters

SerialNumber　デバイスのｼﾘｱﾙ番号

ID　デバイスの ID の格納先へのポインタ

Remarks

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

尚、このｼﾘｱﾙ番号は本体記載のｼﾘｱﾙ番号と同一です。

取得した ID は以後、A/D 制御関数等に引数として渡すことになります。

Example

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

result = RAI_OpenBySerialNumber(serialNumber, &id);
if (result == RAI_OK) {
       // RAI_OpenBySerialNumber 成功
}
else {
       // RAI_OpenBySerialNumber 失敗
}
```

RAI_Close

デバイスをクローズします。

int **RAI_Close** (BYTE *ID*)

Parameters

ID　　デバイスの ID

Example

```
int result;
BYTE id;      // オープンしたデバイスの ID

result = RAI_Close(id);
if (result == RAI_OK) {
       // RAI_Close 成功
}
else {
       // RAI_Close 失敗
}
```

2.2 A/D 制御関数

RAI_GetAnalogVoltage

サンプリング周期 100usec でアナログ電圧値を取得します。

int **RAI_GetAnalogVoltage** (BYTE *ID*, double **CH0Value*, double **CH1Value*,
double **CH2Value*, double **CH3Value*, int *SamplingCount*)

Parameters

ID	デバイスの ID
CH0Value	CH0 アナログ電圧値の格納先配列へのポインタ
CH1Value	CH1 アナログ電圧値の格納先配列へのポインタ
CH2Value	CH2 アナログ電圧値の格納先配列へのポインタ
CH3Value	CH3 アナログ電圧値の格納先配列へのポインタ
SamplingCount	取得するアナログ電圧値のサンプリング数

Remarks

本関数が成功すると *CH0~3Value* の配列にアナログ電圧値が格納されます。単位は[V]です。
取得しないチャネルには NULL を指定することが可能です。
サンプリング数の範囲は 1～1000000 です。
アナログ電圧値が入力測定レンジ外の場合のアナログ電圧値は保証できません。

Example

```
int result;
BYTE id;      // オープンしたデバイスの ID
double ch0val[10000], ch1val[10000], ch2val[10000], ch3val[10000];
int samplingCount;

samplingCount = 10000;      // サンプリング数 10000 (100usec×10000＝1sec)
result = RAI_GetAnalogVoltage(id, ch0val, ch1val, ch2val, ch3val, samplingCount);
if (result == RAI_OK) {
       // RAI_GetAnalogVoltage 成功
}
else {
       // RAI_GetAnalogVoltage 失敗
}
```

RAI_GetAnalogVoltageEx

指定したサンプリングレートでアナログ電圧値を取得します。

int **RAI_GetAnalogVoltageEx** (BYTE *ID*, double **CH0Value*, double **CH1Value*,
double **CH2Value*, double **CH3Value*, int *SamplingRate*, int *SamplingCount*)

Parameters

ID	デバイスの ID
CH0Value	CH0 アナログ電圧値の格納先配列へのポインタ
CH1Value	CH1 アナログ電圧値の格納先配列へのポインタ
CH2Value	CH2 アナログ電圧値の格納先配列へのポインタ
CH3Value	CH3 アナログ電圧値の格納先配列へのポインタ
SamplingRate	取得するアナログ電圧値のサンプリングレート
SamplingCount	取得するアナログ電圧値のサンプリング数

Remarks

本関数は **RAI_GetAnalogVoltage** にサンプリングレートを指定できるように拡張した関数です。
SamplingRate、*SamplingCount* にサンプリング設定をします。
SamplingRate×100 がサンプリング周期（usec）となります。

Example

```
int result;
BYTE id;      // オープンしたデバイスの ID
double ch0val[10000], ch1val[10000], ch2val[10000], ch3val[10000];
int samplingRate, samplingCount;

samplingRate = 10;   // サンプリング周期 1msec
samplingCount = 1000;
result = RAI_GetAnalogVoltageEx(id, ch0val, ch1val, ch2val, ch3val,
                                samplingRate, samplingCount);
if (result == RAI_OK) {
       // RAI_GetAnalogVoltageEx 成功
}
else {
       // RAI_GetAnalogVoltageEx 失敗
}
```

RAI_GetAnalogVoltageExEx

指定したﾄﾘｶﾞｰ条件でアナログ電圧値を取得します。

int **RAI_GetAnalogVoltageExEx** (BYTE *ID*, double **CH0Value*, double **CH1Value*,
double **CH2Value*, double **CH3Value*, int *SamplingRate*, int *SamplingCount*,
int **ReadCount*, BYTE *TriggerCH*, BYTE *TriggerType*,
double *TriggerThreshold*, int *TriggerWaitCount*)

Parameters

ID	デバイスの ID
CH0Value	CH0 アナログ電圧値の格納先配列へのポインタ
CH1Value	CH1 アナログ電圧値の格納先配列へのポインタ
CH2Value	CH2 アナログ電圧値の格納先配列へのポインタ
CH3Value	CH3 アナログ電圧値の格納先配列へのポインタ
SamplingRate	取得するアナログ電圧値のサンプﾘﾝｸﾞレート
SamplingCount	取得するアナログ電圧値のサンプﾘﾝｸﾞ数
ReadCount	取得したアナログ電圧値のカウﾝﾄ数のポインタ
TriggerCH	ﾄﾘｶﾞｰチャネル
TriggerType	ﾄﾘｶﾞｰタイプ
TriggerThreshold	しきい値
TriggerWaitCount	ﾄﾘｶﾞｰ待ちカウﾝﾄ数

Remarks

本関数は **RAI_GetAnalogVoltageEx** にﾄﾘｶﾞｰ条件を指定できるように拡張した関数です。
TriggerCH、*TriggerType*、*TriggerThreshold*、*TriggerWaitCount* にﾄﾘｶﾞｰ条件を設定します。
TriggerCH はﾄﾘｶﾞｰするチャネルを 0（CH0）～3（CH3）で設定します。
TriggerThreshold はﾄﾘｶﾞｰするしきい値を電圧値[V]で設定します。
TriggerWaitCount はﾄﾘｶﾞｰ条件が満たされるまでの待ちカウﾝﾄ数です。
このカウﾝﾄ数は設定したサンプﾘﾝｸﾞ周期と同じ間隔でのカウﾝﾄとなります。
ﾄﾘｶﾞｰ条件が満たされなかった場合は、データを取得しないで関数から戻ってきます。
このとき、*ReadCount* は 0 になります。

TriggerType	説明
0	設定したしきい値より電圧が高い（レベル）
1	設定したしきい値より電圧が低い（レベル）
2	設定したしきい値を低い方から横切る（立上り）
3	設定したしきい値を高い方から横切る（立下り）

Example

```
int result;
BYTE id;      // オープンしたデバイスの ID
double ch0val[10000], ch1val[10000], ch2val[10000], ch3val[10000];
int samplingRate, samplingCount, readCount;
BYTE triggerCH, triggerType;
double triggerThreshold;
int triggerWaitCount;

samplingRate = 100;
samplingCount = 100;
readCount = 0;
triggerCH = 1;       // CH1
triggerType = 2;     // 立上り
triggerThreshold = 3.5;     // 3.5V
triggerWaitCount = 1000;    // 10msec×1000＝10sec
result = RAI_GetAnalogVoltageExEx(id, ch0val, ch1val, ch2val, ch3val,
                                  samplingRate, samplingCount, &readCount,
                                  triggerCH, triggerType, triggerThreshold,
                                  triggerWaitCount);

if (result == RAI_OK) {
       // RAI_GetAnalogVoltageExEx 成功
       if (readCount == 0) {
              // トリガー条件満たされず
       }
}
else {
       // RAI_GetAnalogVoltageExEx 失敗
}
```