USB A/D コンバータボード

FAD-16H シリーズ

FAD-16HS / FAD-16HT

プログラミングガイド

Rev 1.1

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、**FAD-16H 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 **FAD-16H 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。

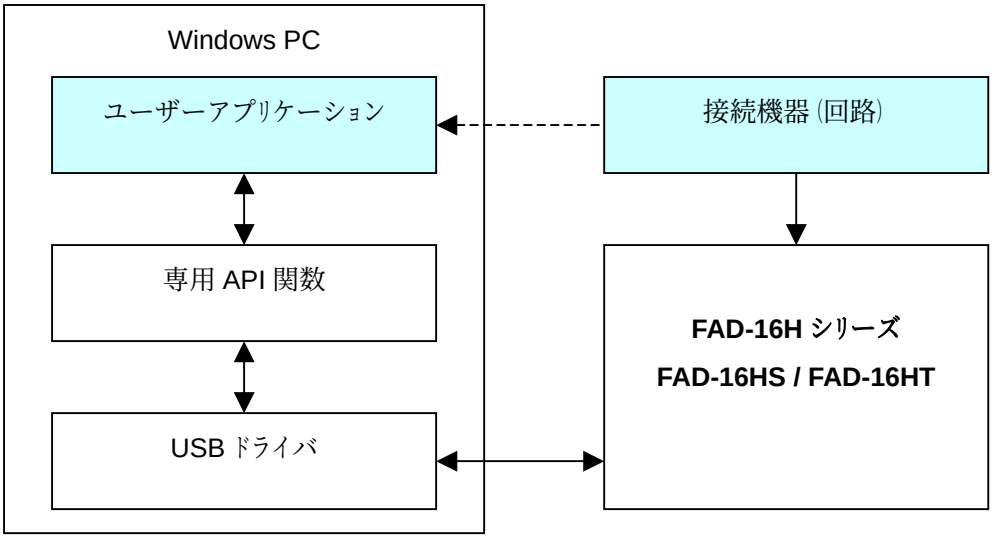


図 1.1 構成

1.2 関数について

FAD-16H シリーズ専用 API 関数は関数群をモジュール化した「FAD16H.dll」で提供されます。
「FAD16H.dll」ファイルは **FAD-16H 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

「FAD16H.h」、「FAD16H.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

「FAD16H.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

「FAD16H.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

「FAD16H.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
FAD16H_OK	0	正常終了
FAD16H_INVALID_ID	1	デバイスの ID が無効
FAD16H_DEVICE_NOT_FOUND	2	デバイスが見つからない
FAD16H_DEVICE_NOT_OPENED	3	デバイスがオープンできない
FAD16H_OTHER_ERROR	4	その他のエラーが発生した
FAD16H_COMMUNICATION_ERROR	5	通信エラーが発生した
FAD16H_INVALID_PARAMETER	6	パラメータが無効
FAD16H_MEMORY_ERROR	7	メモリ容量が不足している

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

FAD16H_GetNumberOfDevices

現在接続されているデバイスの数を取得します。

int **FAD16H_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 = FAD16H_GetNumberOfDevices(&number);
if (result == FAD16H_OK) {
	// FAD16H_GetNumberOfDevices 成功
}
else {
	// FAD16H_GetNumberOfDevices 失敗
}
```

FAD16H_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 **FAD16H_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 **FAD16H_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 = FAD16H_GetSerialNumber(index, &serialNumber);
if (result == FAD16H_OK) {
	// FAD16H_GetSerialNumber 成功
}
else {
	// FAD16H_GetSerialNumber 失敗
}
```

FAD16H_Open

デバイスをオープンし、デバイスの ID を取得します。

int **FAD16H_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 = FAD16H_Open(&id);
if (result == FAD16H_OK) {
	// FAD16H_Open 成功
}
else {
	// FAD16H_Open 失敗
}
```

FAD16H_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 **FAD16H_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 **FAD16H_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 = FAD16H_OpenBySerialNumber(serialNumber, &id);
if (result == FAD16H_OK) {
       // FAD16H_OpenBySerialNumber 成功
}
else {
       // FAD16H_OpenBySerialNumber 失敗
}
```

FAD16H_Close

デバイスをクローズします。

int **FAD16H_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 = FAD16H_Close(id);
if (result == FAD16H_OK) {
       // FAD16H_Close 成功
}
else {
       // FAD16H_Close 失敗
}
```

FAD16H_GetDeviceType

デバイスタイプ（機種型番）を取得します。

int **FAD16H_GetDeviceType** (BYTE *ID*, BYTE **DeviceType*)

Parameters

ID　　デバイスの ID

DeviceType　　デバイスタイプ（機種型番）の格納先へのポインタ

Remarks

本関数が成功すると *DeviceType* にデバイスタイプ（機種型番）が格納されます。

デバイスタイプ（機種型番）の詳細については「**USB A/D コンバータボード FAD-16H シリーズ　取扱説明書**」をご参照ください。

DeviceType	説明
0	**FAD-16HS**
1	**FAD-16HT**

Example

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

result = FAD16H_GetDeviceType(id, &deviceType);
if (result == FAD16H_OK) {
       // FAD16H_GetDeviceType 成功
}
else {
       // FAD16H_GetDeviceType 失敗
}
```

2.2 A/D 制御関数

FAD16H_Start

指定したパラメータで A/D 変換を開始します。

int **FAD16H_Start** (BYTE *ID*, BYTE *SamplingMode*, int *SamplingRate*, int *SamplingCount*)

Parameters

ID	デバイスの ID
SamplingMode	サンプリングモード
SamplingRate	サンプリングレート
SamplingCount	サンプリング数

Remarks

SamplingMode にサンプリングするクロックを設定します。

SampligMode	説明
0	内部クロック
1	外部クロック信号立上り
2	外部クロック信号立下り

SamplingRate にサンプリングレートを設定します。

SamplingRate×50 がサンプリング周期 (nsec) となります。

SamplingRate の範囲は **FAD-16HS** が 10～2097152、**FAD-16HT** が 40～2097152 です。

SamplingMode が内部クロックのとき有効です。

SamplingCount に取得するサンプリング数を設定します。

SamplingCount の範囲は **FAD-16HS** が 1～2097152、**FAD-16HT** が 1～524288 です。

本関数を実行すると、設定されたサンプリングで A/D 変換を開始し、順次メモリへ書き込まれます。
所定のサンプリング数のデータを全てメモリに書き終えると A/D 変換が停止します。
そのときのイメージを以下のメモリマップに示します。

デバイス：**FAD-16HS**
サンプリング数：1600000
の場合

デバイス：**FAD-16HT**
サンプリング数：400000
の場合

Example

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

samplingMode = 0;    // 内部クロック
samplingRate = 40;   // サンプリング周期 2usec(500ksps)
samplingCount = 1000;
result = FAD16H_Start(id, samplingMode, samplingRate, samplingCount);
if (result == FAD16H_OK) {
       // FAD16H_Start 成功
}
else {
       // FAD16H_Start 失敗
}
```

FAD16H_TriggerStart

指定したパラメータで A/D 変換を開始します。

int **FAD16H_TriggerStart** (BYTE *ID*, BYTE *SamplingMode*, int *SamplingRate*,
int *PostTriggerCount*, BYTE *TriggerType*, double *TriggerThreshold*, int *TriggerOccurs*)

Parameters

ID	デバイスの ID
SamplingMode	サンプリングモード
SamplingRate	サンプリングレート
PostTriggerCount	ポストトリガデータ数
TriggerType	トリガ条件
TriggerThreshold	しきい値
TriggerOccurs	トリガ条件合致回数

Remarks

SamplingMode にサンプリングするクロックを設定します。

SampligMode	説明
0	内部クロック
1	外部クロック信号立上り
2	外部クロック信号立下り

SamplingRate にサンプリングレートを設定します。

SamplingRate×50 がサンプリング周期（nsec）となります。

SamplingRate の範囲は **FAD-16HS** が 10～2097152、**FAD-16HT** が 40～2097152 です。

SamplingMode が内部クロックのとき有効です。

PostTriggerCount にトリガ後に取得するデータ数を設定します。

PostTriggerCount の範囲は **FAD-16HS** が 1～2097152、**FAD-16HT** が 1～524288 です。

TriggerType にﾄﾘｶﾞ条件を設定します。

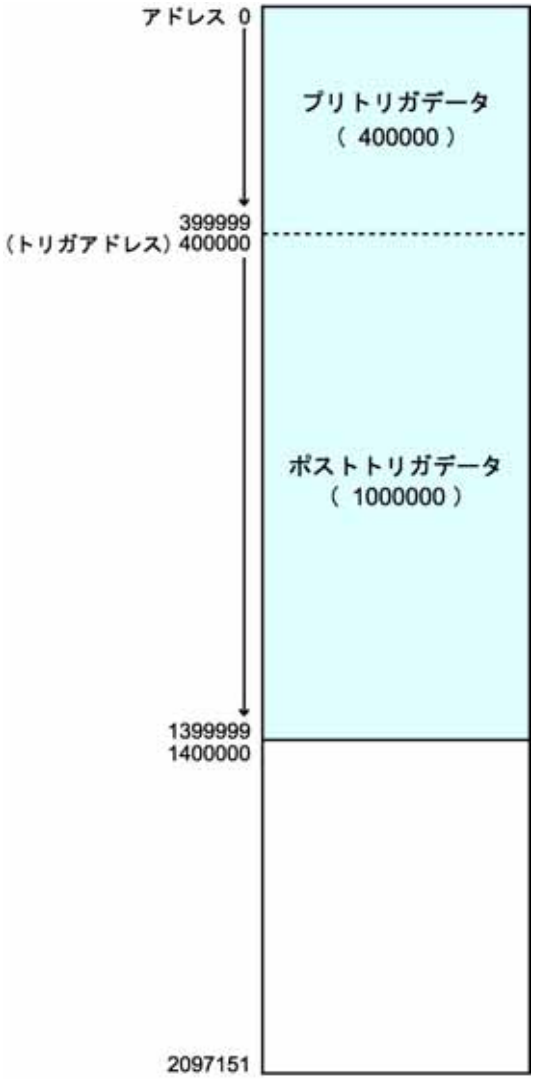
TriggerType	説明
0	設定したしきい値より電圧が高い（レベル）
1	設定したしきい値より電圧が低い（レベル）
2	設定したしきい値を低い方から横切る（立上り）
3	設定したしきい値を高い方から横切る（立下り）
4	外部同期信号が”Off”から”On”への変化点（On エッジ）
5	外部同期信号が”On”から”Off”への変化点（Off エッジ）

TriggerThreshold にﾄﾘｶﾞするしきい値を電圧値[V]で設定します。
TriggerType を、しきい値を使用する設定にしたとき有効です。

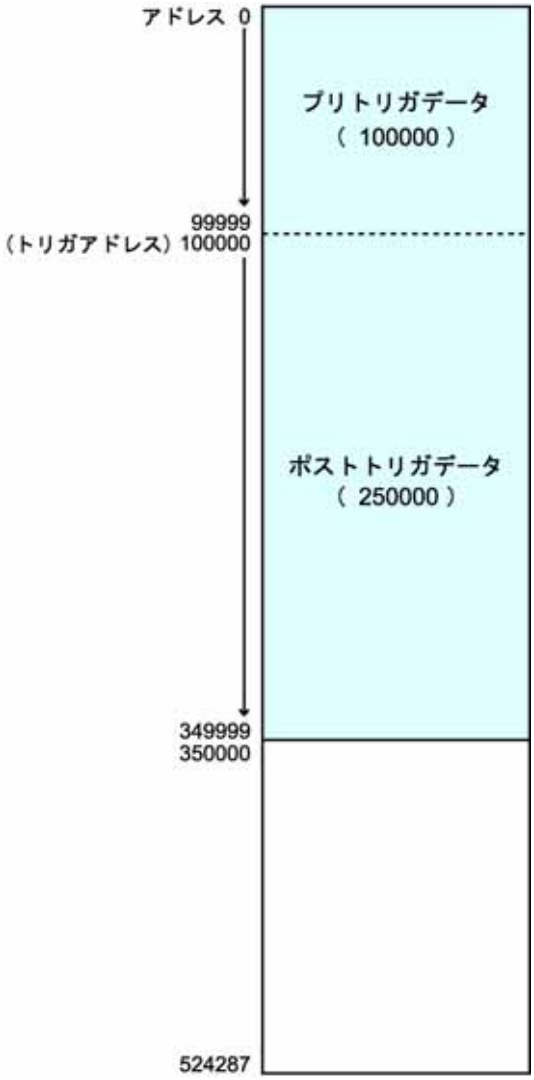
TriggerOccurs にﾄﾘｶﾞ条件合致回数を設定します。
TriggerOccurs の範囲は 1～65536 です。

本関数を実行すると、設定されたサンプリングで A/D 変換を開始し、順次メモリへ書き込まれます。
この時、トリガ条件に合致すると、そのアドレスをトリガアドレスとし、このアドレスからポストトリガデータ数のデータを全てメモリに書き終えると A/D 変換が停止します。
アドレス 0 からトリガアドレス以前までのデータがプリトリガデータになります。
トリガアドレス（プリトリガデータ数）は全メモリ容量からポストトリガデータ数を引いた数以上にはなりません。
そのときのイメージを以下のメモリマップに示します。

デバイス：**FAD-16HS**
ポストトリガデータ数：1000000
トリガアドレス：400000
の場合



デバイス：**FAD-16HT**
ポストトリガデータ数：250000
トリガアドレス：100000
の場合



Example

```
int result;
BYTE id;      // オープンしたデバイスの ID
BYTE samplingMode
int samplingRate
int postTriggerCount;
BYTE triggerType;
double triggerThreshold;
int triggerOccurs;

samplingMode = 0;    // 内部クロック
samplingRate = 40;   // サンプリング周期 2usec(500ksps)
postTriggerCount = 1000;
triggerType = 2;     // 立上り
triggerThreshold = 3.5;     // 3.5V
triggerOccurs = 1;
result = FAD16H_TriggerStart(id, samplingMode, samplingRate, postTriggerCount,
                             triggerType, triggerThreshold, triggerOccurs);
if (result == FAD16H_OK) {
       // FAD16H_TriggerStart 成功
}
else {
       // FAD16H_TriggerStart 失敗
}
```

FAD16H_Stop

A/D 変換を停止します。

int **FAD16H_Stop** (BYTE *ID*)

Parameters

ID　　デバイスの ID

Remarks

本関数が成功すると A/D 変換、及びﾒﾓﾘへのデータの書き込みを停止します。このとき、ﾄﾘｶﾞステータス、ﾄﾘｶﾞｱﾄﾞﾚｽ、及びﾒﾓﾘ書き込みデータ数は、停止時の状態が保持されます。
ﾄﾘｶﾞステータスは、**FAD16H_GetStatus** で取得することができます。
ﾄﾘｶﾞｱﾄﾞﾚｽは、**FAD16H_GetTriggerAddr** で取得することができます。
ﾒﾓﾘ書き込みデータ数は、**FAD16H_GetMemoryWriteCount** で取得することができます。
これらの関数の詳細については、後述にある各々の関数説明をご参照ください。

Example

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

result = FAD16H_Stop(id);

if (result == FAD16H_OK) {
       // FAD16H_Stop 成功
}
else {
       // FAD16H_Stop 失敗
}
```

FAD16H_GetStatus

ステータスを取得します。

int **FAD16H_GetStatus** (BYTE *ID*, BYTE **MemoryStatus*, BYTE **TriggerStatus*,
BYTE **ExtTriggerState*, BYTE * *ExtClockState*)

Parameters

ID	デバイスの ID
MemoryStatus	メモリ書込み動作状態の格納先へのポインタ
TriggerStatus	トリガ状態の格納先へのポインタ
ExtTriggerState	外部トリガ端子状態の格納先へのポインタ
ExtClockState	外部クロック端子状態の格納先へのポインタ

Remarks

本関数が成功すると *MemoryStatus* にメモリ書込み動作状態、*TriggerStatus* にトリガ状態、*ExtTriggerState* に外部トリガ端子状態、*ExtClockState* に外部クロック端子状態が格納されます。
外部トリガ端子、外部クロック端子の詳細については「**USB A/D コンバータボード FAD-16H シリーズ取扱説明書**」をご参照ください。

MemoryStatus	説明
0	メモリ書込み停止
1	メモリ書込み中

TriggerStatus	説明
0	トリガがかかっていない状態
1	トリガがかかっている状態

ExtTriggerState	説明
0	外部トリガ端子が”OFF”状態
1	外部トリガ端子が”ON”状態

ExtClockState	説明
0	外部クロック端子が”OFF”状態
1	外部クロック端子が”ON”状態

Example

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

result = FAD16H_GetStatus(id, &memoryStatus, &triggerStatus,
                                    &extTriggerState, &extClockState);

if (result == FAD16H_OK) {
       // FAD16H_GetStatus 成功
}
else {
       // FAD16H_GetStatus 失敗
}
```

FAD16H_GetTriggerAddr

ﾄﾘｶﾞｱﾄﾞﾚｽを取得します。

int **FAD16H_GetTriggerAddr** (BYTE *ID*, int **Addr*)

Parameters

ID	デバイスの ID
Addr	ﾄﾘｶﾞｱﾄﾞﾚｽの格納先へのポインタ

Remarks

本関数が成功すると *Addr* にﾄﾘｶﾞｱﾄﾞﾚｽが格納されます。
ﾄﾘｶﾞｱﾄﾞﾚｽとは、ﾄﾘｶﾞ点（ﾄﾘｶﾞ条件が合致した位置）のﾒﾓﾘｱﾄﾞﾚｽで、ﾌﾟﾘﾄﾘｶﾞﾃﾞｰﾀ及びﾎﾟｽﾄﾄﾘｶﾞﾃﾞｰﾀの先頭番地と等しい値です。

Example

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

result = FAD16H_GetTriggerAddr(id, &addr);

if (result == FAD16H_OK) {
       // FAD16H_GetTriggerAddr 成功
}
else {
       // FAD16H_GetTriggerAddr 失敗
}
```

FAD16H_GetMemoryWriteCount

ﾒﾓﾘに書き込まれたデータ数を取得します。

int **FAD16H_GetMemoryWriteCount** (BYTE *ID*, int **Count*)

Parameters

ID	デバイスの ID
Count	ﾒﾓﾘに書き込まれたデータ数の格納先へのポインタ

Remarks

本関数が成功すると *Count* にﾒﾓﾘに書き込まれたデータ数が格納されます。

Example

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

result = FAD16H_GetMemoryWriteCount(id, &count);

if (result == FAD16H_OK) {
       // FAD16H_GetMemoryWriteCount 成功
}
else {
       // FAD16H_GetMemoryWriteCount 失敗
}
```

FAD16H_ReadMemory

データをメモリから読み出します。

int **FAD16H_ReadMemory** (BYTE *ID*, double **Data*, int Addr, int *Count*)

Parameters

ID	デバイスの ID
Data	データの格納先配列へのポインタ
Addr	メモリアドレス
Count	データ数

Remarks

Addr に読み出し始めるメモリアドレス、*Count* にそのアドレスから読み出すデータ数を設定します。
本関数が成功すると *Data* 配列にデータが格納されます。
データはアナログ電圧値で、単位は[V]です。
メモリの読み出しは、いつでも（メモリ書き込み中であっても）読み出す事ができます。
但し、メモリ書き込みアドレス（メモリ書き込み数－1）以上のデータは無効なデータになります。

Example

```
int result;
BYTE id;      // オープンしたデバイスの ID
double data[1000];
int addr;
int count;

addr = 0;
count = 1000;
result = FAD16H_ReadMemory(id, data, addr, count);

if (result == FAD16H_OK) {
       // FAD16H_ReadMemory 成功
}
else {
       // FAD16H_ReadMemory 失敗
}
```

3　データ取得シーケンス

FAD-16H シリーズのデータ取得シーケンスの例を以下に示します。

・データ取得シーケンス①　(トリガ条件なし、データ一括取得)

① **FAD16H_Open**、または **FAD16H_OpenBySerialNumber** でデバイスをオープンします。
② **FAD16H_Start** で A/D 変換パラメータを設定し、A/D 変換を開始します。
③ **FAD16H_GetStatus** でメモリステータスを監視します。(サンプリングデータ書き込み終了待ち)
④ **FAD16H_ReadMemory** でサンプリングデータ (全数) をメモリから読み出します。
⑤ **FAD16H_Close** でデバイスをクローズします。

・データ取得シーケンス②　(トリガ条件あり、データ一括取得)

① **FAD16H_Open**、または **FAD16H_OpenBySerialNumber** でデバイスをオープンします。
② **FAD16H_TriggerStart** で A/D 変換パラメータを設定し、A/D 変換を開始します。
③ **FAD16H_GetStatus** でトリガステータスを監視します。(トリガ待ち)
④ **FAD16H_GetTriggerAddr** でトリガアドレスを取得します。(トリガアドレスが 0 なら⑥へ)
⑤ **FAD16H_ReadMemory** でプリトリガデータ (任意数) をメモリから読み出します。
⑥ **FAD16H_GetStatus** でメモリステータスを監視します。(ポストトリガデータ書き込み終了待ち)
⑦ **FAD16H_ReadMemory** でポストトリガデータ (全数) をメモリから読み出します。
⑧ **FAD16H_Close** でデバイスをクローズします。

・データ取得シーケンス③　(トリガ条件なし、データ分割取得)

① **FAD16H_Open**、または **FAD16H_OpenBySerialNumber** でデバイスをオープンします。
② **FAD16H_Start** で A/D 変換パラメータを設定し、A/D 変換を開始します。
③ **FAD16H_GetMemoryWriteCount** でメモリ書き込み数を取得します。
④ **FAD16H_ReadMemory** でメモリアドレス、データ数を指定しデータをメモリから読み出します。
(③、④を繰り返し、メモリに書き込まれたサンプリングデータを順次読み出します。)
⑤ **FAD16H_Close** でデバイスをクローズします。

上記のデータ取得シーケンスの詳細は、サンプルプログラムをご参照ください。