

WM2000 シリーズ API ユーザーズ・ガイド

Manual Number FOK-8440319B00

適用ユニット

WM2000TA

WM2000TB

WM2000SA

WM2000SB

WM2000ZA

WM2000ZB

WM2000ZC

権利について

無断転載禁止。本マニュアルに記載されている文章および図表は、すべて株式会社アドバンテストの著作物です。株式会社アドバンテストの書面による許可なしに無断で複製することは、いかなる形態においても禁じられています。本マニュアルに記載されている内容は、予告なしに変更されることがあります。

商標および登録商標

- ・ ADVANTEST および AirLogger™ は、（株）アドバンテストの、日本およびその他の国における登録商標または商標です。
- ・ Microsoft、Windows 7、Windows8、Windows10、Visual C++、および Visual Studio は、米国 Microsoft Corporation の、米国およびその他の国における登録商標または商標です。
- ・ その他の製品名は、それぞれの所有者の商標です。

改版履歴

版数	日付	備考
01	2017.11.16	
02	2019.7.29	WM2000SB 対応

はじめに

ここでは、本書を有効に活用していただくために、本書の内容と表記ルールについて説明します。

本書の対象製品

本書の内容は、以下の製品を対象に書かれています。

- WM2000TA
- WM2000TB
- WM2000SA
- WM2000SB
- WM2000ZA
- WM2000ZB
- WM2000ZC

本書の構成

本書の各章の内容は以下のとおりです。

1. 製品概要	本製品の製品概要について説明します。
2. API を使用したプログラムの開発方法	本製品がお手元に届いてからのセットアップおよび使用方法について説明します。
3. API 詳細	WM2000 シリーズ API の詳細な仕様について説明します。
4. サンプル・プログラム	WM2000 シリーズ API を使用したサンプル・プログラムを示します。

前提条件

測定ユニットの登録、通信周波数設定は PC アプリケーションにて事前に設定しておく必要があります。

詳細は、「無線データロガー WM2000 シリーズ ユーザーズ・ガイド」の「6. 測定ユニットの管理」を参照してください。

本書の読み方

GUI 操作の表記

GUI（Graphical User Interface）画面上で操作する部位の名称は、[]で囲んで表記します。
また、メニューの中にあるコマンドを選択する操作は、矢印記号（→）を使って表記します。

字体または記号	説明	例
[]	GUI 画面上で操作する部位の名称を表します。	[File]メニュー [OK]ボタン
[]→[]	メニューの中にあるコマンドを表します。	[File]→[Exit]

キー操作の表記

キーボード上のキーは、[]で囲んで表記します。また、複数のキーを同時に押すときの操作は、プラス記号(+)を使って表記し、複数のキーを順に押すときの操作は読点（、）を使って表記します。

字体または記号	説明	例
[]	キーボード上のキーを表します。	[Return]キー
[]+[]	キーをプラス記号(+)でつないでいるときは、最初のキーを押しながら 2 番目のキーを押すことを表します。	[Ctrl]+[c]

目次

1. 製品概要	1-1
2. API を使用したプログラムの開発方法	2-1
2.1 PC 環境の確認.....	2-1
2.2 ファイル構成	2-1
2.3 API の使用準備	2-2
2.4 API の使用	2-10
2.5 API を使用したプログラムの実行例	2-14
3. API 詳細	3-1
3.1 初期化処理.....	3-1
3.2 測定ユニット情報の取得	3-2
3.3 セットアップ情報の設定・取得	3-5
3.4 断線の確認.....	3-25
3.5 ゼロ点調整の開始	3-27
3.6 計測開始の準備	3-29
3.7 計測の開始.....	3-30
3.8 測定データの読み出し.....	3-31
3.9 ステータスの読み出し.....	3-34
3.10 計測の停止.....	3-35
3.11 LOST データの受信開始.....	3-39
3.12 LOST データの受信停止.....	3-40
3.13 通信周波数の設定	3-41

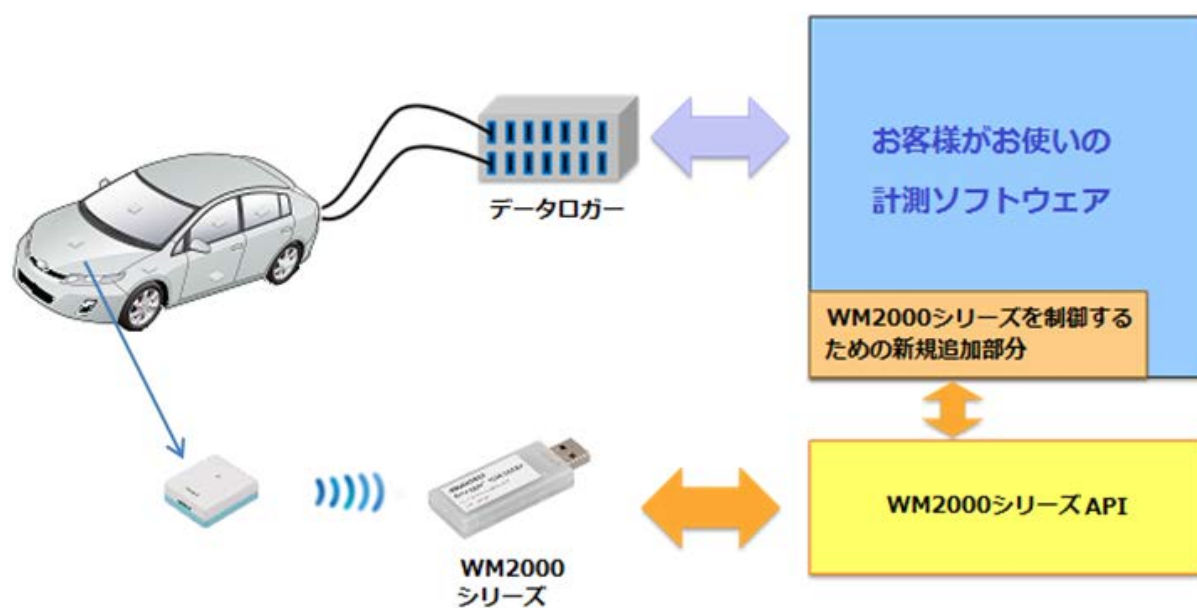
3.14 測定ユニットの強制終了	3-46
3.15 終了処理	3-47
4. サンプル・プログラム.....	4-1
販売お取引約款.....	1

1. 製品概要

WM2000 シリーズ API(以降、API と呼びます)は、WM2000 シリーズを制御するための関数群です。

API を使用することにより、お客様が作成したプログラムから WM2000 シリーズを制御することができます。

以下は、お客様が作成したプログラムに API を組み込む際のイメージを図で表したものです。



2. API を使用したプログラムの開発方法

この章では、API を使用したプログラムの開発方法について説明します。

なお、説明の内容や手順は、「Windows 7 Professional 64bit」がインストールされた PC において「Microsoft Visual Studio Express 2015 for Windows Desktop」を使用した場合のものとなります。

上記以外の開発環境においては、各開発環境のマニュアル等を参照の上、ご使用ください。

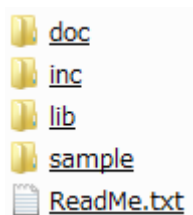
2.1 PC 環境の確認

API を使用したプログラムを開発する PC が、以下の条件を満たしていることを確認してください。

項目	条件
OS の種類	Windows 7 32bit または 64bit Windows 8 32bit または 64bit Windows 8.1 32bit または 64bit Windows 10 32bit または 64bit
ディスク空き容量	10MB 以上

2.2 ファイル構成

ここでは、ウェブサイトからダウンロードしたファイルの構成について説明します。



- **doc**
本書(WM2000_series_API_ug_j.pdf)のフォルダです。
- **inc**
プログラムから API を使用するためにインクルードするヘッダ・ファイル(AirLoggerAPI.h)のフォルダです。

- **lib**
プログラムから **API** を使用するためにリンクするライブラリ・ファイル (AirLoggerWM2000API.dll, AirLoggerWM2000API.lib) のフォルダです。
- **sample**
サンプル・プログラムのフォルダです。
本書の「4. サンプル・プログラム」に記載されているプログラムが含まれています。
- **ReadMe.txt**
ウェブサイトからダウンロードしたファイルの構成を記述したテキスト・ファイルです。

2.3 API の使用準備

プログラムから **API** を使用するには、ヘッダ・ファイルとライブラリ・ファイルをプロジェクトから参照できるように設定する必要があります。

ここでは、空のプロジェクトである "**SampleApplication**" を例として、**API** の使用準備手順を説明します。

プロジェクトを作成する手順については、**Visual Studio** のマニュアルを参照してください。

まずは、以下の手順でヘッダ・ファイルとライブラリ・ファイルをプロジェクトのフォルダにコピーします。

1-1. プロジェクトを作成したフォルダに "**inc**", "**lib**" というフォルダを作成します。

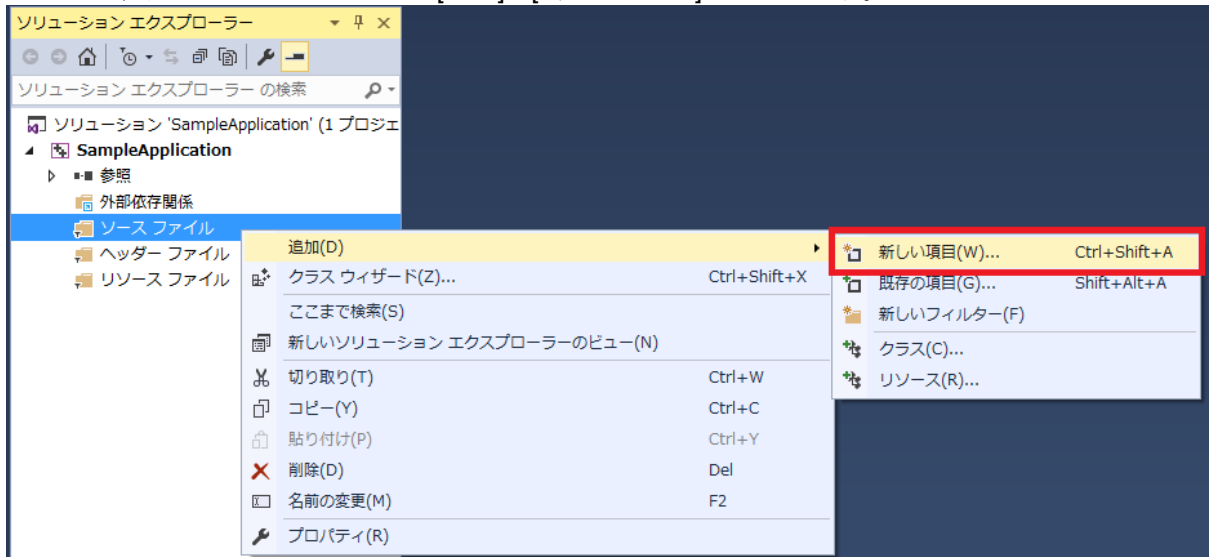
1-2. **AirLoggerAPI.h** を "**inc**" フォルダにコピーします。

1-3. **AirLoggerWM2000API.lib** を "**lib**" フォルダにコピーします。

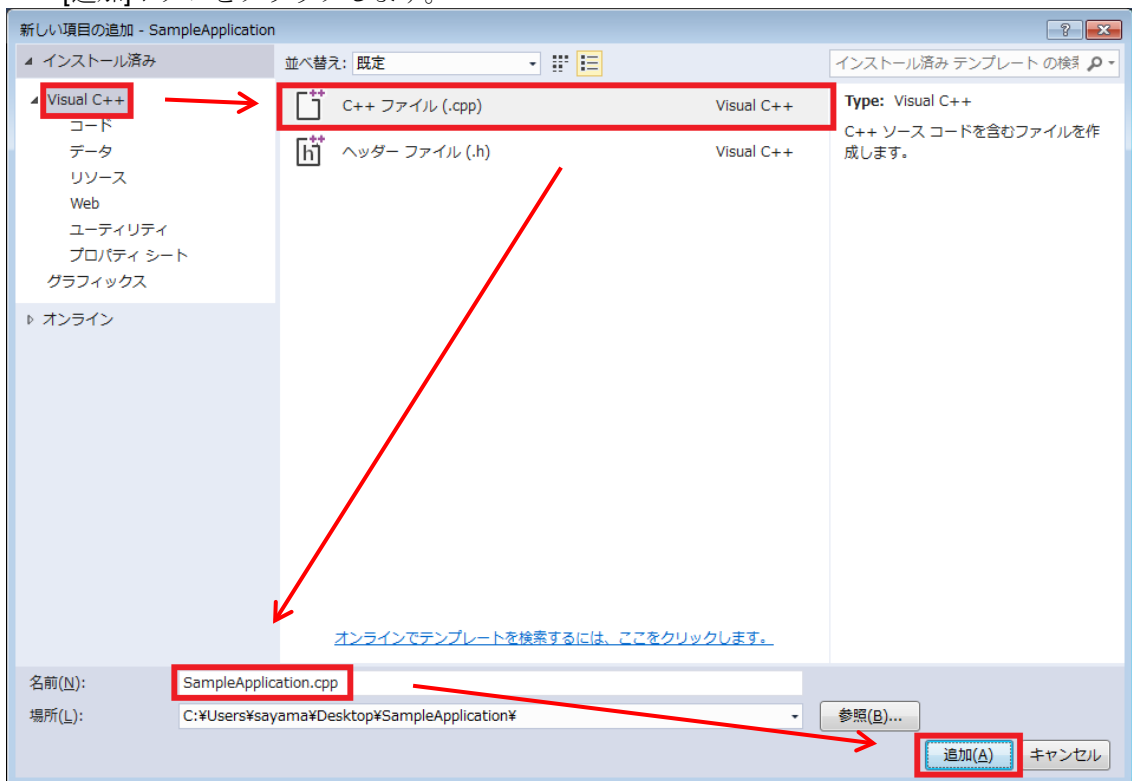
※ 必ず上記の場所にコピーしなければならない訳ではありません。ただし、別の場所にコピーした場合は、以降の手順の 2-4. と 3-4. で入力する内容を適宜変更してください。

ソース・ファイルを以下の手順で追加します。

- 1-4.[ソリューション エクスプローラー]の[SampleApplication]中の[ソース ファイル]を右クリックし、表示されるメニューから[追加]→[新しい項目...]を選択します。

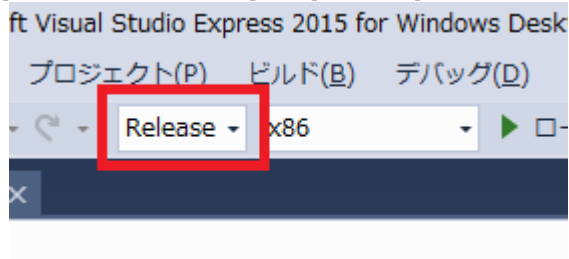


- 1-5.[Visual C++]→[C++ファイル(.cpp)]を選択し、名前に「SampleApplication.cpp」を入力して[追加]ボタンをクリックします。

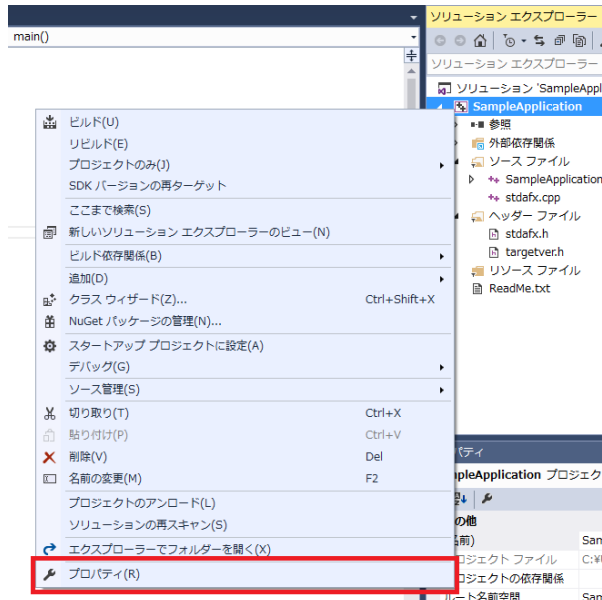


次に、以下の手順でヘッダ・ファイルを参照できるようにします。

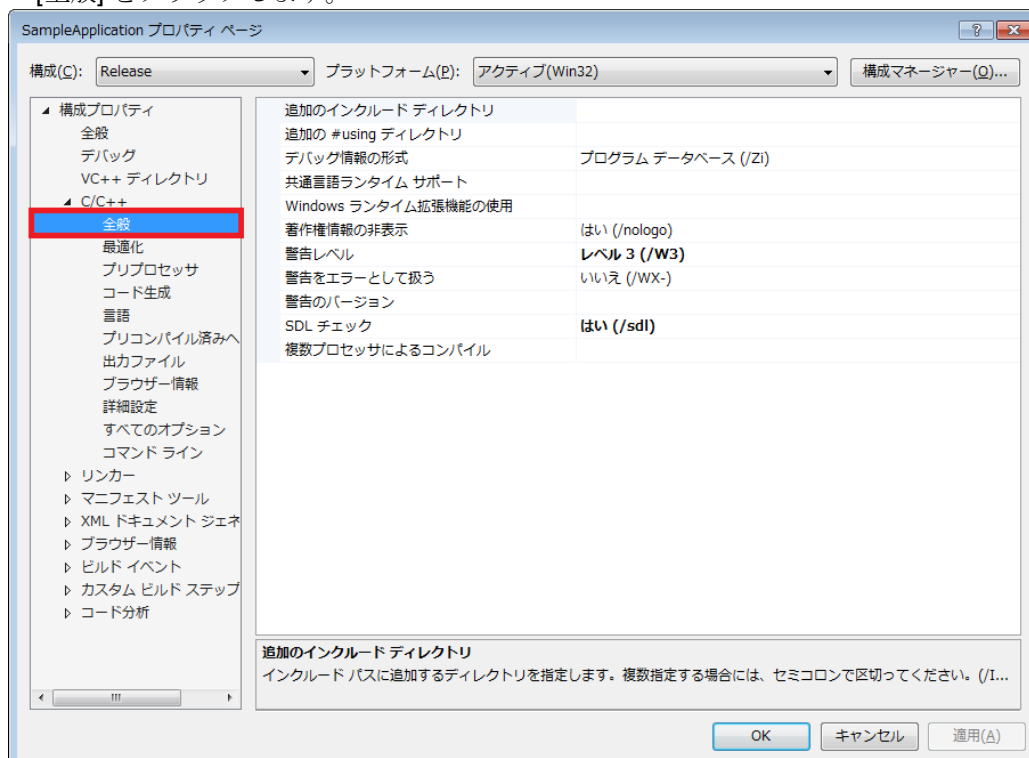
2-1.[ソリューション構成]から[Release]を選択します。



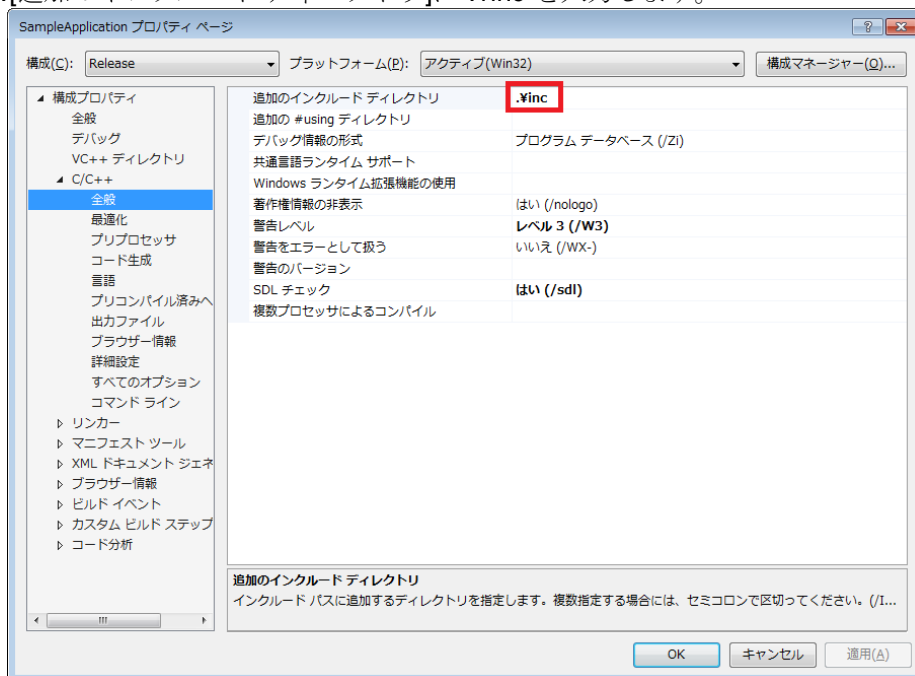
2-2. [ソリューション エクスプローラー]の[SampleApplication]を右クリックし、[プロパティ]をクリックします。



2-3.[プロパティ ページ] ダイアログ・ボックスが表示されますので、[構成プロパティ]→[C/C++]→[全般]をクリックします。

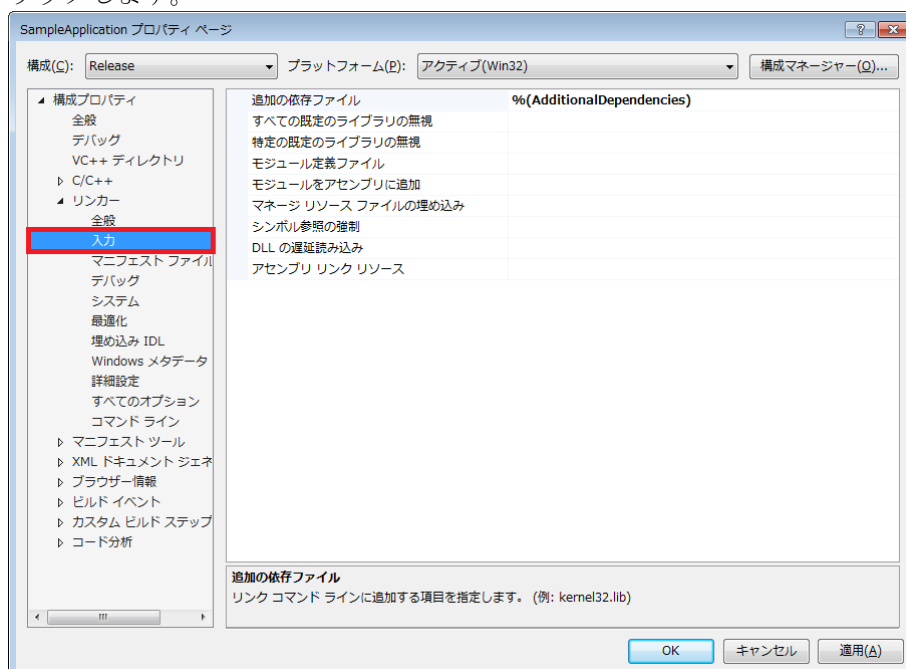


2-4.[追加のインクルード ディレクトリ]に".¥inc"を入力します。

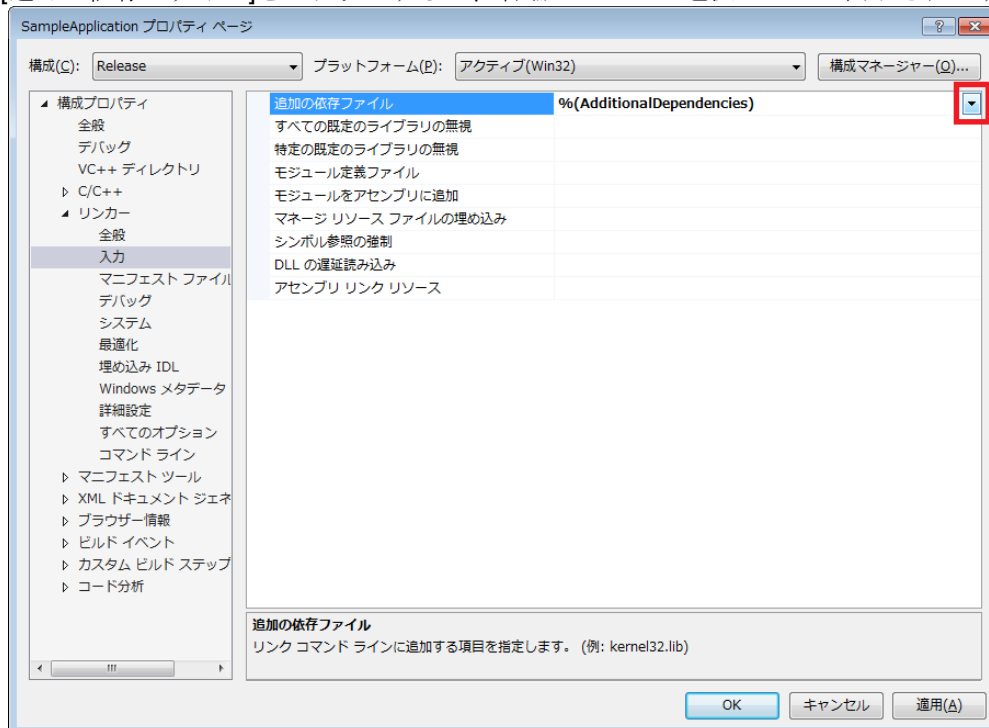


最後に、以下の手順でライブラリ・ファイルを参照できるようにします。

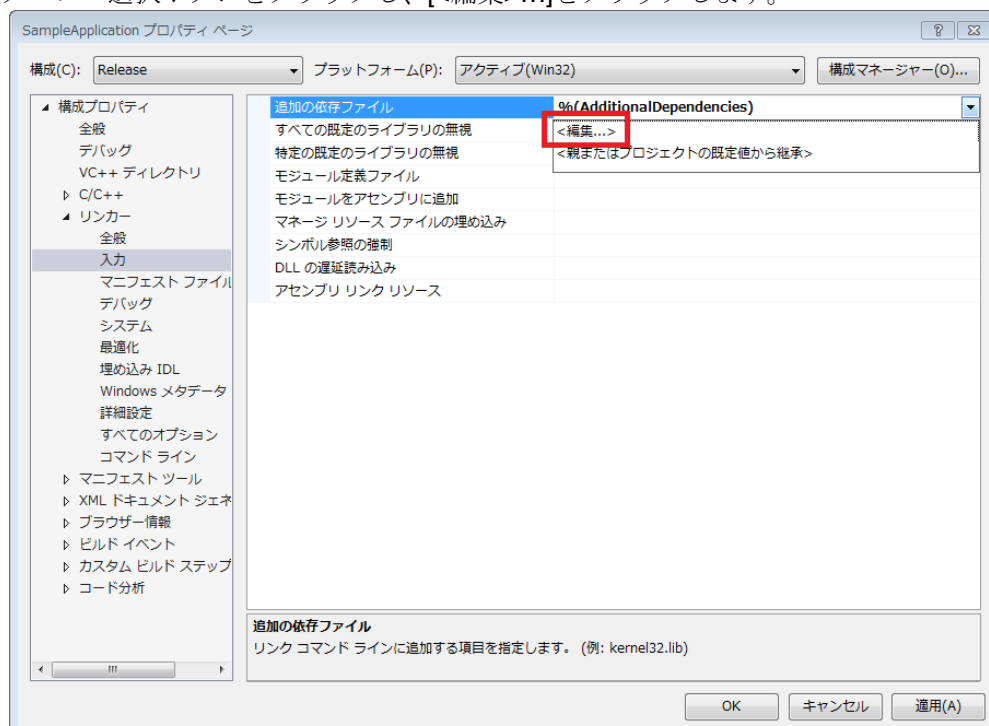
3-1.[プロパティ ページ] ダイアログ・ボックスから、[構成プロパティ]→[リンカー]→[入力]をクリックします。



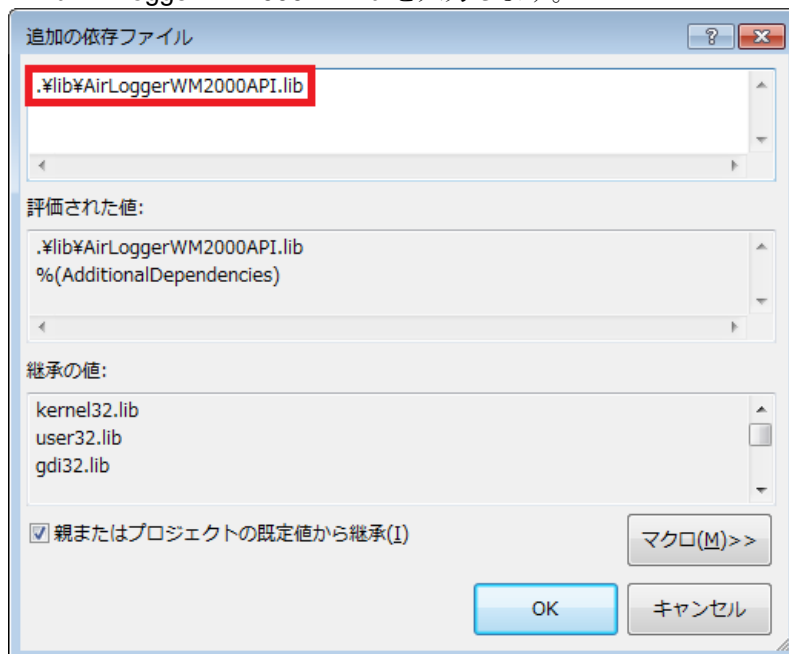
3-2.[追加の依存ファイル]をクリックすると、右端にメニュー選択ボタンが表示されます。



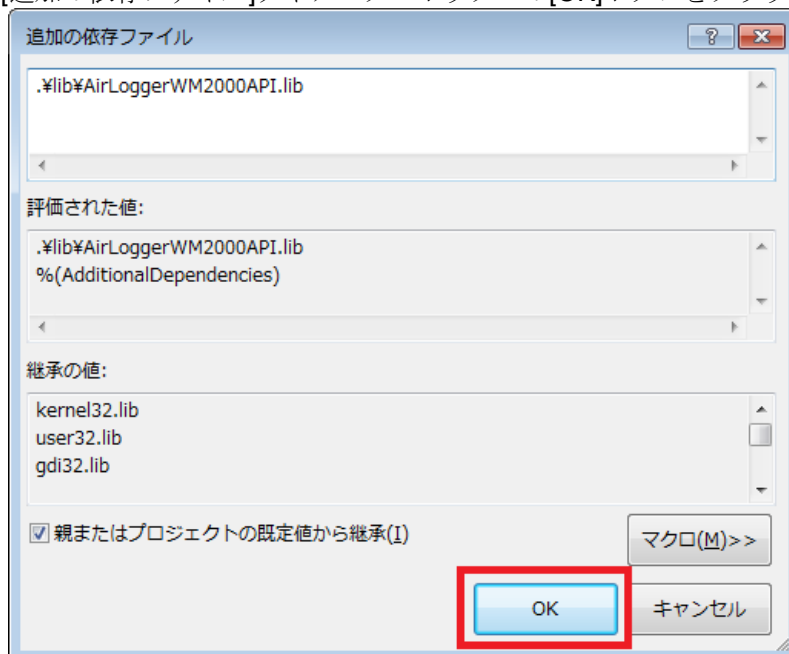
3-3. メニュー選択ボタンをクリックし、[<編集>...]をクリックします。



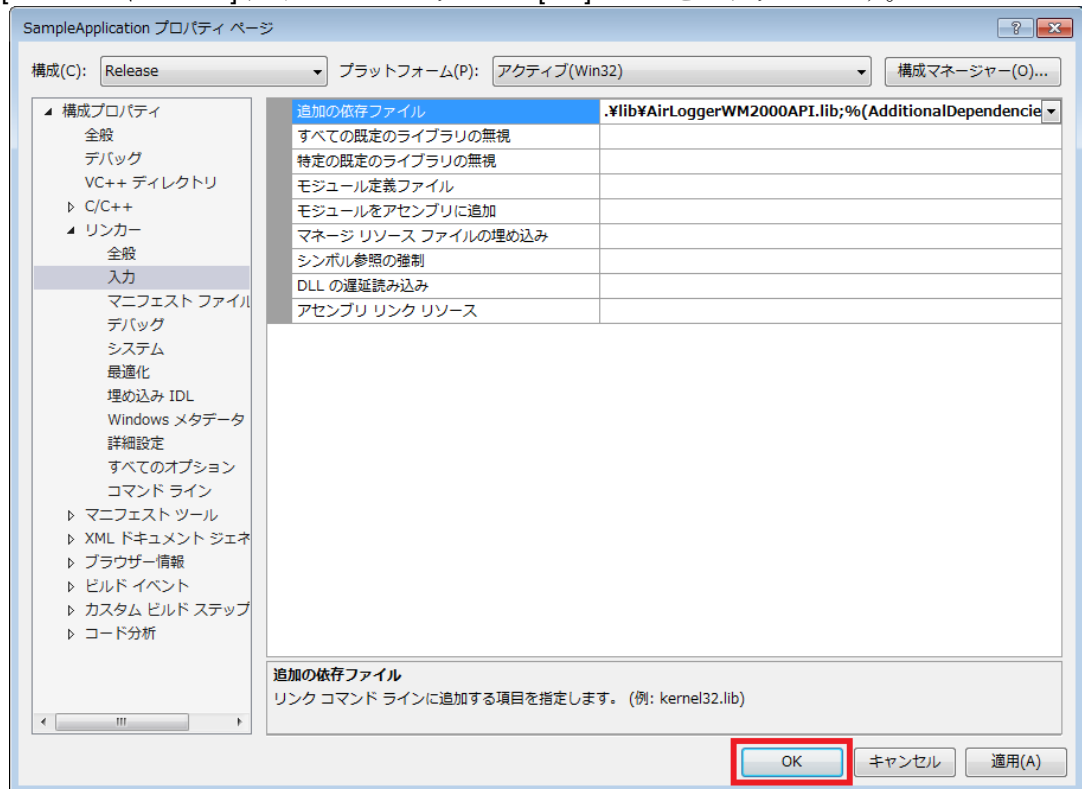
- 3-4.[追加の依存ファイル]ダイアログ・ボックスが表示されますので、一番上の領域に".¥lib¥AirLoggerWM2000API.lib"を入力します。



- 3-5.[追加の依存ファイル]ダイアログ・ボックスの[OK]ボタンをクリックします。



3-6.[プロパティ ページ] ダイアログ・ボックスの[OK]ボタンをクリックします。

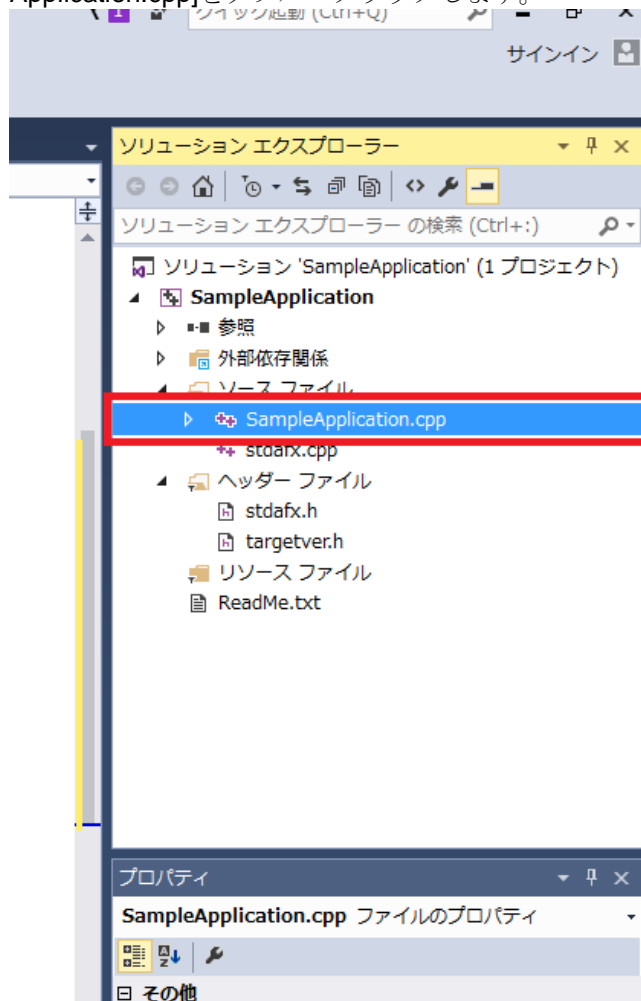


以上の手順で、API を使用するための準備が完了となります。

2.4 API の使用

ここでは、実際に API をソースコードに組み込む手順を、簡単な例を用いて説明します。
なお、前提として「2.3 使用準備」の手順が実施されているものとします。

1. [ソリューション エクスプローラー]の[SampleApplication]→[ソース ファイル]→[Sample Application.cpp]をダブル・クリックします。



2. [コードエディター]にソースコードが表示されますので、ソースコードを以下の内容に置き換えます。

```
#include <Windows.h>
#include "AirLoggerAPI.h"
#include <iostream>
#include <string>
using namespace std;

int setCondition = 0;

void _stdcall callback(int unitNo, int channelNo, unsigned long long
sequenceNo, int batteryLevel, int radioWave, BOOL isLost, BOOL
isDisconnect, int errorFlag, int dataCount, double* pData) {
    cout << endl;
    cout << "UNIT NO: " << unitNo << endl;
    cout << "CH NO: " << channelNo << endl;
    cout << "SEQ NO: " << sequenceNo << endl;
    cout << "BATTERY: " << batteryLevel << endl;
    cout << "RADIO WAVE: " << radioWave << endl;
    cout << "LOST: " << isLost << endl;
    cout << "DISCONNECT: " << isDisconnect << endl;
    cout << "ERROR: " << errorFlag << endl;
    cout << "DATA COUNT: " << dataCount << endl;
    for (int i = 0; i < dataCount; i++) {
        cout << "DATA" << i << ": " << pData[i] << endl;
    }
    cout << flush;
}

void _stdcall cond_callback(int unitNo) {
    if (unitNo == -1){
        cout << "Setting Condition ERROR" << endl;
        cout << flush;
    }
    else if (unitNo == 0){
        cout << "Setting Condition OK" << endl;
        cout << flush;
        setCondition = 1;
    }
}

void checkError(string functionName, int returnCode) {
```

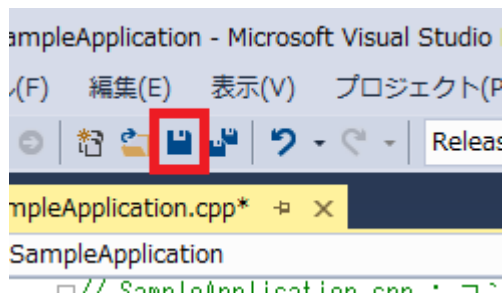
```

switch (returnCode) {
case AIRLOGGER_SUCCESS:
    return;
case AIRLOGGER_INVALID_PARAMETER:
    cout << functionName + ": AIRLOGGER_INVALID_PARAMETER" << endl;
    break;
case AIRLOGGER_INVALID_STATE:
    cout << functionName + ": AIRLOGGER_INVALID_STATE" << endl;
    break;
case AIRLOGGER_COMMUNICATION_ERROR:
    cout << functionName + ": AIRLOGGER_INVALID_STATE" << endl;
    break;
}
exit(returnCode);
}

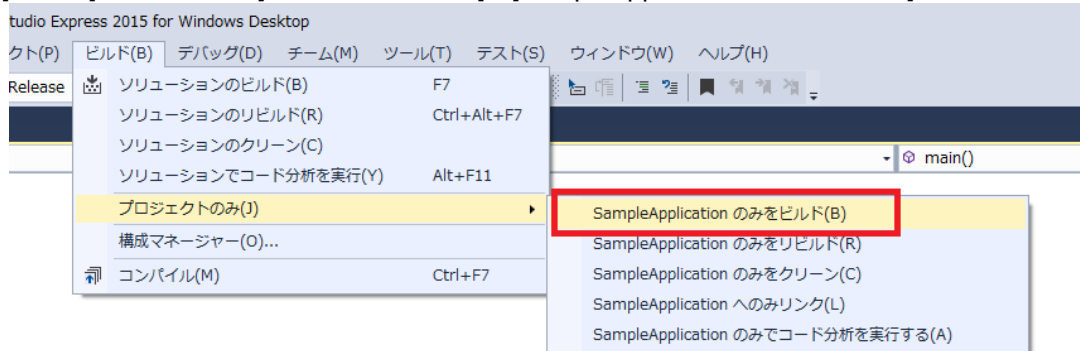
int main() {
    checkError("AirLogger_Initialize", AirLogger_Initialize());
    checkError("AirLogger_SetUnitSamplingRate",
AirLogger_SetUnitSamplingRate(1, AIRLOGGER_SAMPLE_1SEC));
    checkError("AirLogger_SetUnitSensorType",
AirLogger_SetUnitSensorType(1, 1, AIRLOGGER_SENSOR_THERMOCOUPLE_K));
    checkError("AirLogger_SetUnitSensorType",
AirLogger_SetUnitSensorType(1, 2, AIRLOGGER_SENSOR_THERMOCOUPLE_K));
    checkError("AirLogger_SendMeasurementCondition",
AirLogger_SendMeasurementCondition(cond_callback,30));
    while(setCondition == 0){
        Sleep(1000);
    }
    checkError("AirLogger_SetMeasurementStandby",AirLogger_SetMeasurements
tandby());
    checkError("AirLogger_WM2000_StartMeasurement",
AirLogger_WM2000_StartMeasurement(callback));
    Sleep(3000);
    checkError("AirLogger_StopMeasurement", AirLogger_StopMeasurement());
    checkError("AirLogger_Finalize", AirLogger_Finalize());
    return 0;
}

```

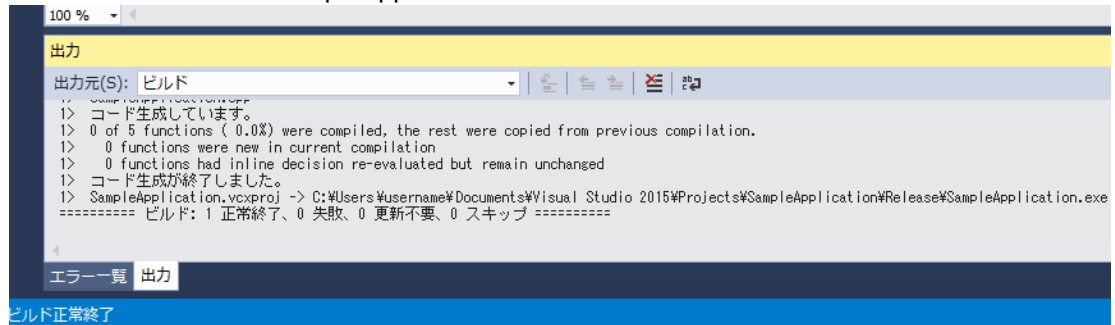
3. [保存]ボタンをクリックします。



4. [ビルド]メニューから[プロジェクトのみ]→[SampleApplication のみをビルド]を選択します。



5. ビルドが成功すると、[出力]に以下のような内容が表示され、プロジェクトのフォルダの "Release" フォルダに "SampleApplication.exe" が作成されます。



2.5 API を使用したプログラムの実行例

ここでは、API を使用したプログラムの実行例について、「2.4 API の使用」の項で作成したプログラムを用いて説明します。

1. プロジェクトのフォルダの "Release" フォルダに "AirLoggerWM2000API.dll" をコピーします。
2. [デバッグ]メニューから[デバッグなしで開始]を選択します。



3. 実行が成功すると、以下が表示されます。

```
UNIT NO: 1
CH NO: 1
SEQ NO: 0
BATTERY: 0
RADIO WAVE: 0
LOST: 0
DISCONNECT: 0
ERROR: 0
DATA COUNT: 1
DATA0: 24.2

UNIT NO: 1
CH NO: 2
SEQ NO: 0
BATTERY: 0
RADIO WAVE: 0
LOST: 0
DISCONNECT: 0
ERROR: 0
DATA COUNT: 1
DATA0: 24.1
```

なお、API を使用したプログラムには、以下の制限があります。

- API を使用したプログラムを、複数同時に実行することはできません。
- API を使用したプログラムと WM2000 シリーズの PC アプリケーションを、同時に実行することはできません。
- 同じプログラム内で WM1000 シリーズの API を使用することはできません。

3. API 詳細

この章では、API に含まれる関数の詳細について説明します。

3.1 初期化処理

◆ AirLogger_Initialize

(概要)

API を使用するための初期化処理を実行します。
 API で行う処理部分の最初に実行してください。
 AirLogger_Initialize()を実行していない状態で AirLogger_GetState() 以外の関数を実行すると、エラーとなります。

(書式)

```
int AirLogger_Initialize(void);
```

(パラメータ)

なし

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	初期化処理が正しく実行されました。
AIRLOGGER_INVALID_STATE	PC 側通信ユニットが接続されていません。
AIRLOGGER_MULTIPLE_PC_COMMUNICATION_UNIT	PC 側通信ユニットの初期化に失敗しました。
AIRLOGGER_UNREGISTERD_ROM	PC 側通信ユニットに測定ユニットが登録されていません。

3.2 測定ユニット情報の取得

◆ AirLogger_GetSystemUnitCount

(概要)

PC 側通信ユニットに登録している測定ユニットの個数を読み出します。

(書式)

```
int AirLogger_GetSystemUnitCount(int *count);
```

(パラメータ)

count

取得した測定ユニットの個数を格納する領域です。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	初期化処理が正しく実行されました。
AIRLOGGER_INVALID_PARAMETER	以下の状態のため、測定ユニット個数が取得できませんでした。 ・ count が NULL である。

◆ **AirLogger_GetSystemUnitInfomation**

(概要)

PC 側通信ユニットに登録している測定ユニットの番号、種類およびシリアル番号を読み出します。

(書式)

```
int AirLogger_GetSystemUnitInfomation(int *unitNumber, int *unitType, char
**serialNumber);
```

(パラメータ)

unitNumber

取得した測定ユニット番号を格納する領域です。

unitType

取得した測定ユニットの種類を格納する領域です。

定数名	説明
AIRLOGGER_UNIT_TYPE_WM2000TA	WM2000TA
AIRLOGGER_UNIT_TYPE_WM2000TB	WM2000TB
AIRLOGGER_UNIT_TYPE_WM2000SA	WM2000SA
AIRLOGGER_UNIT_TYPE_WM2000SB	WM2000SB

serialNumber

取得した測定ユニットのシリアル番号を格納する領域です。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	初期化処理が正しく実行されました。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていない、測定ユニット情報を取得できませんでした。

定数名	説明
AIRLOGGER_INVALID_PARAMETER	以下の状態のため、測定ユニット情報が取得できませんでした。 ・ unitNumber、unitType、serialNumber が NULL である。

3.3 セットアップ情報の設定・取得

◆ AirLogger_SetTemperatureUnit

(概要)

温度データの単位を設定します。

(書式)

```
int AirLogger_SetTemperatureUnit(int unit);
```

(パラメータ)

unit

温度データの単位を指定します。
以下のいずれかの定数を指定してください。

定数名	説明
AIRLOGGER_TEMPERATURE_UNIT_CELSIUS	温度データの単位を摂氏とします。
AIRLOGGER_TEMPERATURE_UNIT_FAHRENHEIT	温度データの単位を華氏とします。
AIRLOGGER_TEMPERATURE_UNIT_KELVIN	温度データの単位をケルビンとします。

初期状態は、AIRLOGGER_TEMPERATURE_UNIT_CELSIUS が設定されています。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	温度データの単位が正しく設定されました。
AIRLOGGER_INVALID_PARAMETER	不正な温度データの単位が指定されたため、温度データの単位を設定できませんでした。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていない、温度データの単位を設定できませんでした。

◆ **AirLogger_GetTemperatureUnit**

(概要)

温度データの単位を取得します。

(書式)

```
int AirLogger_GetTemperatureUnit(int* unit);
```

(パラメータ)

unit

取得した温度データの単位を格納する領域です。

温度データの単位については、AirLogger_SetTemperatureUnit() を参照してください。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	温度データの単位が正しく取得されました。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、温度データの単位を取得できませんでした。
AIRLOGGER_INVALID_PARAMETER	以下の状態のため、温度データの単位が取得できませんでした。 <ul style="list-style-type: none">unit が NULL である。

◆ **AirLogger_SetUnitSamplingRate**

(概要)

測定間隔を設定します。

(書式)

```
int AirLogger_SetUnitSamplingRate(int unitNumber, int rate);
```

(パラメータ)

unitNumber

ユニット番号を設定します。

指定可能な値は 1～100 です。

rate

測定間隔を指定します。

以下のいずれかの定数を指定してください。

定数名	説明
AIRLOGGER_SAMPLE_100USEC	測定間隔を 100 マイクロ秒とします。(*1)
AIRLOGGER_SAMPLE_200IUEC	測定間隔を 200 マイクロ秒とします。(*1)
AIRLOGGER_SAMPLE_500UEC	測定間隔を 500 マイクロ秒とします。(*1)
AIRLOGGER_SAMPLE_1MSEC	測定間隔を 1 ミリ秒とします。(*1)
AIRLOGGER_SAMPLE_2MSEC	測定間隔を 2 ミリ秒とします。(*1)
AIRLOGGER_SAMPLE_5MSEC	測定間隔を 5 ミリ秒とします。(*1)
AIRLOGGER_SAMPLE_10SEC	測定間隔を 10 ミリ秒とします。(*1)
AIRLOGGER_SAMPLE_20EC	測定間隔を 20 ミリ秒とします。(*1)
AIRLOGGER_SAMPLE_50EC	測定間隔を 50 ミリ秒とします。(*1)
AIRLOGGER_SAMPLE_100MSEC	測定間隔を 100 ミリ秒とします。
AIRLOGGER_SAMPLE_200MSEC	測定間隔を 200 ミリ秒とします。
AIRLOGGER_SAMPLE_500MSEC	測定間隔を 500 ミリ秒とします。

定数名	説明
AIRLOGGER_SAMPLE_1SEC	測定間隔を 1 秒とします。
AIRLOGGER_SAMPLE_2SEC	測定間隔を 2 秒とします。
AIRLOGGER_SAMPLE_10SEC	測定間隔を 10 秒とします。
AIRLOGGER_SAMPLE_1MIN	測定間隔を 1 分とします。
AIRLOGGER_SAMPLE_5MIN	測定間隔を 5 分とします。 (*2)
AIRLOGGER_SAMPLE_10MIN	測定間隔を 10 分とします。 (*2)

(*1) WM2000SB のみ使用可能です。

(*2) WM2000TA、WM2000TB、および WM2000SA で使用可能です。

サンプリング周期により、同時測定可能なユニット数が異なります。

- ① 100usec～50msec
・ 25 台
- ② 100msec～500msec
・ 30 台(WM2000TB 併用接続時は 15 台)
- ③ 1sec 以上
・ 100 台(②併用接続時は 40 台)

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	測定間隔が正しく設定されました。
AIRLOGGER_INVALID_PARAMETER	不正なユニット番号または、測定間隔が指定されたため、測定間隔を設定できませんでした。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、測定間隔を設定できませんでした。

◆ **AirLogger_GetUnitSamplingRate**

(概要)

測定間隔を取得します。

(書式)

```
int AirLogger_GetUnitSamplingRate(int unitNumber, int* rate);
```

(パラメータ)

unitNumber

ユニット番号を設定します。

指定可能な値は 1～100 です。

rate

取得した測定間隔を格納する領域です。

温度データの単位については、AirLogger_SetUnitSamplingRate()を参照してください。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	測定間隔が正しく取得されました。
AIRLOGGER_INVALID_PARAMETER	以下のいずれかの状態のため、測定間隔が取得できませんでした。 <ul style="list-style-type: none"> • rate が NULL である。 • 不正なユニット番号が指定されている。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、温度データの単位を取得できませんでした。

◆ AirLogger_SetUnitSensorType

(概要)

測定ユニットで使用するセンサのタイプを設定します。

(書式)

```
int AirLogger_SetUnitSensorType(int unitNumber, int channelNumber, int type);
```

(パラメータ)

unitNumber

ユニット番号を設定します。
指定可能な値は 1～100 です。

channelNumber

対象とする測定ユニットのチャンネル番号を指定します。
指定可能な値は以下のとおりです。

WM2000TA : 1～2
WM2000TB : 1～7
WM2000SA : 1
WM2000SB : 1～3

type

熱電対のタイプを指定します。
以下のいずれかの定数を指定してください。

定数名	説明
AIRLOGGER_SENSOR_NO_USE	測定ユニットを使用しません。計測の対象から除外します。
AIRLOGGER_SENSOR_THERMOCOUPLE_K	熱電対を K タイプとします。
AIRLOGGER_SENSOR_THERMOCOUPLE_T	熱電対を T タイプとします。
AIRLOGGER_SENSOR_THERMOCOUPLE_J	熱電対を J タイプとします。
AIRLOGGER_SENSOR_VOLTAGE	電圧測定とします。
AIRLOGGER_SENSOR_STRAIN	ひずみ測定とします。

初期状態は、AIRLOGGER_SENSOR_NO_USE が設定されています。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	センサのタイプが正しく設定されました。
AIRLOGGER_INVALID_PARAMETER	不正なユニット番号、チャンネル番号またはセンサのタイプが指定されたため、センサのタイプを設定できませんでした。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、センサのタイプを設定できませんでした。

◆ **AirLogger_GetUnitSensorType**

(概要)

測定ユニットに設定されているセンサのタイプを取得します。

(書式)

```
int AirLogger_GetUnitSensorType(int unitNumber, int channelNumber, int*
pType);
```

(パラメータ)

unitNumber

ユニット番号を設定します。
指定可能な値は 1～100 です。

channelNumber

対象とする測定ユニットのチャンネル番号を指定します。
指定可能な値は以下のとおりです。

WM2000TA : 1～2

WM2000TB : 1～7

WM2000SA : 1

WM2000SB : 1～3

pType

取得したセンサのタイプを格納する領域です。
センサのタイプについては、AirLogger_SetUnitSensorType() を参照してください。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	センサのタイプが正しく取得されました。
AIRLOGGER_INVALID_PARAMETER	以下のいずれかの状態のため、センサのタイプが取得できませんでした。 <ul style="list-style-type: none"> ・ pType が NULL である。 ・ 不正なユニット番号またはチャンネル番号が指定されている。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、センサのタイプを取得できませんでした。

◆ AirLogger_SetVoltageRange

(概要)

電圧の測定レンジを設定します。

(書式)

```
int AirLogger_SetVoltageRange(int unitNumber, int channelNumber, int
range);
```

(パラメータ)

unitNumber

ユニット番号を設定します。

指定可能な値は 1～100 です。

channelNumber

対象とする測定ユニットのチャネル番号を指定します。

指定可能な値は以下のとおりです。

WM2000TA : 1～2

WM2000TB : 1～7

range

測定レンジを設定します。

以下のいずれかの定数を指定してください。

定数名	説明
AIRLOGGER_VOLTAGE_RANGE_100MV	測定レンジを+/-100mV とします。
AIRLOGGER_VOLTAGE_RANGE_1V	測定レンジを+/-1V とします。
AIRLOGGER_VOLTAGE_RANGE_12V	測定レンジを+/-12V とします。

初期状態は、AIRLOGGER_VOLTAGE_RANGE_100MV が設定されています。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	測定レンジが正しく設定されました。

定数名	説明
AIRLOGGER_INVALID_PARAMETER	不正なユニット番号、チャンネル番号または測定レンジが指定されたため、測定レンジを設定できませんでした。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、測定レンジを設定できませんでした。

◆ AirLogger_GetVoltageRange

(概要)

測定ユニットに設定されている電圧の測定レンジを取得します。

(書式)

```
int AirLogger_GetVoltageRange(int unitNumber, int channelNumber, int*
pRange);
```

(パラメータ)

unitNumber

ユニット番号を設定します。
指定可能な値は 1～100 です。

channelNumber

対象とする測定ユニットのチャンネル番号を指定します。
指定可能な値は以下のとおりです。

WM2000TA : 1～2

WM2000TB : 1～7

pRange

取得した測定レンジを格納する領域です。
センサのタイプについては、AirLogger_SetVoltageRange() を参照してください。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	測定レンジが正しく取得されました。
AIRLOGGER_INVALID_PARAMETER	以下のいずれかの状態のため、測定レンジが取得できませんでした。 <ul style="list-style-type: none"> pRange が NULL である。 不正なユニット番号またはチャンネル番号が指定されている
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、測定レンジを取得できませんでした。

◆ AirLogger_SetStrainRange

(概要)

ひずみの測定レンジを設定します。

(書式)

```
int AirLogger_SetStrainRange(int unitNumber, int channelNumber, int
range);
```

(パラメータ)

unitNumber

ユニット番号を設定します。

指定可能な値は 1～100 です。

channelNumber

対象とする測定ユニットのチャネル番号を指定します。

指定可能な値は以下のとおりです。

WM2000SA : 1

WM2000SB : 1～3

range

測定レンジを設定します。

以下のいずれかの定数を指定してください。

定数名	説明
AIRLOGGER_STRAIN_RANGE_2000UST	測定レンジを+/-2000uST とします。
AIRLOGGER_STRAIN_RANGE_5000UST	測定レンジを+/-5000uST とします。
AIRLOGGER_STRAIN_RANGE_10000UST	測定レンジを+/-10000uST とします。
AIRLOGGER_STRAIN_RANGE_20000UST	測定レンジを+/-20000uST とします。

初期状態は、AIRLOGGER_STRAIN_RANGE_5000UST が設定されています。

AIRLOGGER_STRAIN_RANGE_2000UST は WM2000SB のみ使用できます。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	測定レンジが正しく設定されました。
AIRLOGGER_INVALID_PARAMETER	不正なユニット番号、チャンネル番号または測定レンジが指定されたため、測定レンジを設定できませんでした。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、測定レンジを設定できませんでした。

◆ **AirLogger_GetStrainRange**

(概要)

測定ユニットに設定されている電圧の測定レンジを取得します。

(書式)

```
int AirLogger_GetStrainRange(int unitNumber, int channelNo, int* pRange);
```

(パラメータ)

unitNumber

ユニット番号を設定します。
指定可能な値は 1～100 です。

channelNo

対象とする測定ユニットのチャネル番号を指定します。
指定可能な値は以下のとおりです。

WM2000SA : 1

WM2000SB : 1 ～ 3

pRange

取得した測定レンジを格納する領域です。
センサのタイプについては、AirLogger_SetStrainRange() を参照してください。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	測定レンジが正しく取得されました。
AIRLOGGER_INVALID_PARAMETER	以下のいずれかの状態のため、測定レンジが取得できませんでした。 <ul style="list-style-type: none"> ・ pRange が NULL である。 ・ 不正なユニット番号またはチャネル番号が指定されている
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、測定レンジを取得できませんでした。

◆ **AirLogger_WarmupSensorUnit**

(概要)

WM2000SB の測定値を安定させるためのウォームアップを実行するか否かを設定します。
AirLogger_SendMeasurementCondition()実行後、ウォームアップが開始します。
 ウォームアップを開始してから測定を開始するまでの推奨待ち時間は 30 分です。

(書式)

```
int AirLogger_WarmupSensorUnit(int unitNumber, BOOL OnOff)
```

(パラメータ)

unitNumber

ユニット番号を設定します。
 指定可能な値は 1～100 です。

OnOff

定数名	説明
TRUE	ウォームアップを実行します
FALSE	ウォームアップを実行しません。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	ウォームアップ情報が正しく設定されました。
AIRLOGGER_INVALID_PARAMETER	不正なユニット番号が指定されたため、ウォームアップ情報が設定されませんでした。

◆ **AirLogger_SendMeasurementCondition**

(概要)

測定条件を各測定ユニットに転送します。

(書式)

```
int AirLogger_SendMeasurementCondition(AirLogger_SendConditionCallback
pFunc, int time);
```

(パラメータ)

pFunc

測定ユニットから測定条件の転送結果を受信した際に呼び出されるコールバック関数のポインタを指定します。

コールバック関数の詳細は、「AirLogger_SendMeasurementConditionCallback()」を参照してください。

time

タイム・アウト時間を秒単位で指定します。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	測定条件の転送が正しく開始されました。
AIRLOGGER_INVALID_PARAMETER	以下が指定されたため、測定条件の転送を開始できませんでした。 <ul style="list-style-type: none"> pFunc が NULL である。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、測定条件の転送を開始できませんでした。
AIRLOGGER_NOT_FIND_ROUTER	高速データレシーバが接続されていません。
AIRLOGGER_DUPLICATE_ROUTER_IP	高速データレシーバの検出に失敗しました。PC の再起動をしてください。
AIRLOGGER_INVALID_TCP_ACK_FREQ	管理者権限で SettingDataReceiver.exe を実行をしてください。実行ファイルは PC アプリケーションをインストールしたフォルダにあります。

◆ AirLogger_SendMeasurementConditionCallback

(概要)

各ユニットに転送した測定条件の転送結果を読み出します。

各ユニットからの転送結果を受信した際に、メイン・スレッドとは異なる専用のスレッドでこの関数が実行されます。

(書式)

```
(*AirLogger_SendMeasurementConditonCallback)(int unitNumber)
```

(パラメータ)

unitNumber

1～100 : 測定条件に成功したユニット番号です。

0 : 全ユニットに測定条件の設定が成功しました。

-1 : AirLogger_SendMeasurementCondition()関数で指定したタイム・アウトの時間内にユニットの測定条件の設定が完了しませんでした。

(戻り値)

なし

◆ AirLogger_EnableHighSpeedConnection

(概要)

高速データレシーバの接続を行います。
測定条件の設定が完了後、実行してください。
本関数は、測定間隔 100msec 未満の測定ユニットのみ有効です。

(書式)

```
int
AirLogger_EnableHighSpeedConnection(AirLogger_HighSpeedConnectionCallback
pFunc, int waitTime)
```

(パラメータ)

pFunc

高速データレシーバの接続結果を受信した際に呼び出されるコールバック関数のポインタを指定します。
コールバック関数の詳細は、「AirLogger_HighSpeedConnectionCallback()」を参照してください。

waitTime

タイム・アウト時間を秒単位で指定します。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	高速データレシーバの接続ができました。
AIRLOGGER_INVALID_PARAMETER	以下が指定されたため、高速データレシーバの接続ができませんでした。 <ul style="list-style-type: none"> pFunc が NULL である。

◆ AirLogger_DisableHighSpeedConnection

(概要)

高速データレシーバの切断を行います。
本関数は、測定間隔 100msec 未満の測定ユニットのみ有効です。

(書式)

```
int  
AirLogger_DisableHighSpeedConnection(AirLogger_HighSpeedConnectionCallback  
pFunc, int waitTime)
```

(パラメータ)

pFunc

高速データレシーバの切断結果を受信した際に呼び出されるコールバック関数のポインタを指定します。
コールバック関数の詳細は、「AirLogger_HighSpeedConnectionCallback()」を参照してください。

waitTime

タイム・アウト時間を秒単位で指定します。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	高速データレシーバの切断ができました。
AIRLOGGER_INVALID_PARAMETER	以下が指定されたため、高速データレシーバの切断ができませんでした。 <ul style="list-style-type: none">pFunc が NULL である。

◆ AirLogger_HighSpeedConnectionCallback

(概要)

高速データレシーバの接続および切断の転送結果を読み出します。
各ユニットからの転送結果を受信した際に、メイン・スレッドとは異なる専用のスレッドでこの関数が実行されます。

(書式)

```
(*AirLogger_HighSpeedConnectionCallback)(int unitNumber)
```

(パラメータ)

unitNumber

- 1～100 : 測定条件に成功したユニット番号です。
- 0 : 全ユニットに測定条件の設定が成功しました。
- 1 : AirLogger_EnableHighSpeedConnection()関数または
AirLogger_DisableHighSpeedConnection()関数で指定したタイム・アウトの時間内に
高速データレシーバの接続および切断が完了しませんでした。

(戻り値)

なし

3.4 断線の確認

◆ AirLogger_CheckDisconnect

(概要)

断線の確認を行います。
本関数は、測定条件を設定した後に使用してください。
WM2000TA,WM2000TB のセンサが熱電対のみ有効です。

(書式)

```
int AirLogger_CheckDisconnect(int *unitNumber, int *isConnect, int *size);
```

(パラメータ)

unitNumber

断線確認を行ったユニット番号が配列で格納されます。
配列のサイズは、測定ユニット数分確保してください。

isConnect

断線確認の結果が以下のようにチャンネル毎に配列で格納されます。
配列のサイズは、測定ユニット数分確保してください。

			Bit6	bit5	bit4	bit3	bit2	bit1	LSB bit0
			CH7	CH6	CH5	CH4	CH3	CH2	CH1

定数名	説明
0	ビットのチャンネルは断線していません。
1	ビットのチャンネルは断線しています。

size

断線確認を行ったユニット数が格納されます。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	断線確認が正しく開始されました。

定数名	説明
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、断線確認を開始できませんでした。
AIRLOGGER_COMMUNICATION_ERROR	以下のいずれかのため、断線の確認ができませんでした。 <ul style="list-style-type: none">・PC 側通信ユニットとの通信に失敗した。・測定条件の設定が成功していない。

3.5 ゼロ点調整の開始

◆ AirLogger_ZeroPointAdjustment

(概要)

測定位置のゼロ点調整を開始します。
 本関数は、測定条件を設定した後に使用してください。
 また、WM2000SA および WM2000SB 以外のユニットは無視されます。

(書式)

```
int AirLogger_ZeroPointAdjustment(int *unitNumber, BOOL *isInit, int
*size);
```

(パラメータ)

unitNumber

ゼロ点調整を行ったユニット番号が配列で格納されます。
 配列のサイズは、測定ユニット数分確保してください。

isInit

ゼロ点調整の結果が配列で格納されます。
 配列のサイズは、測定ユニット数分確保してください。

定数名	説明
TRUE	ゼロ点調整が成功しました。
FALSE	ゼロ点調整が失敗しました。

size

ゼロ点調整を行ったユニット数が格納されます。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	ゼロ点調整が正しく開始されました。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、ゼロ点調整を開始できませんでした。

定数名	説明
AIRLOGGER_COMMUNICATION_ERROR	以下のいずれかのため、測定位置のゼロ点調整を開始できませんでした。 <ul style="list-style-type: none">・ PC 側通信ユニットとの通信に失敗した。・ 測定条件の設定が成功していない。

3.6 計測開始の準備

◆ AirLogger_SetMeasurementStandby

(概要)

測定データの計測準備を行います。

省略時は、AirLogger_WM2000_StartMeasurement()内で約 2 秒計測準備を行い、計測を開始します。

(書式)

```
int AirLogger_SetMeasurementStandby(void);
```

(パラメータ)

なし

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	測定データの計測準備が正しく開始されました。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、測定データの計測準備できませんでした。

3.7 計測の開始

◆ AirLogger_WM2000_StartMeasurement

(概要)

測定データの計測を開始します。

(書式)

```
int AirLogger_WM2000_StartMeasurement(AirLogger_WM2000_Callback pFunc);
```

(パラメータ)

pFunc

測定ユニットから測定データを受信した際に呼び出されるコールバック関数のポインタを指定します。

コールバック関数の詳細は、「3.8 測定データの読み出し」を参照してください。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	測定データの計測が正しく開始されました。
AIRLOGGER_INVALID_PARAMETER	以下のいずれかが指定されたため、測定データの計測を開始できませんでした。 ・ pFunc が NULL である。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、測定データの計測を開始できませんでした。
AIRLOGGER_COMMUNICATION_ERROR	PC 側通信ユニットとの通信に失敗したため、測定データの計測を開始できませんでした。 PC 側通信ユニットが PC の USB 端子に正しく接続されていることを確認してください。

3.8 測定データの読み出し

◆ AirLogger_WM2000_Callback

(概要)

測定データを読み出します。

測定ユニットから測定データを受信した際に、メイン・スレッドとは異なる専用のスレッドでこの関数が実行されます。

(書式)

```
(*AirLogger_WM2000_Callback)(it unitNumber, int channelNumber, unsigned
long long sequenceNumber, int batteryLevel, int radioWave, BOOL isLost,
BOOL isDisconnect, int error, int dataCount, double* pData)
```

(パラメータ)

unitNumber

測定データを読み出したユニット番号です。

channelNumber

測定データを読み出したユニットのチャンネル番号です。

sequenceNumber

シーケンス番号です。シーケンス番号は 0 から始まり、100 μ S ごとに+1 されます。

測定間隔を 100mS にした場合、シーケンス番号は、0→1000→2000...のように更新されます。

データ数が複数の場合(dataCount の値が 2 以上の場合は、一番古いデータのシーケンス番号が取得されます。

batteryLevel

電池残量です。

以下のいずれかの値が取得されます。

定数名	説明
AIRLOGGER_BATTERY_LEVEL_UNKNOWN	電池残量が不明であることを示します。
AIRLOGGER_BATTERY_LEVEL_NORMAL	電池残量が正常レベルであることを示します。
AIRLOGGER_BATTERY_LEVEL_ATTENTION	電池残量が注意レベルであることを示します。

定数名	説明
AIRLOGGER_BATTERY_LEVEL_WARNING	電池残量が警告レベルであることを示します。

radioWave

電波強度です。
以下のいずれかの値が取得されます。

定数名	説明
AIRLOGGER_RADIO_WAVE_FINE	通信状態が良好であることを示します。
AIRLOGGER_RADIO_WAVE_NORMAL	通信状態がやや悪いことを示します。
AIRLOGGER_RADIO_WAVE_BAD	通信状態が悪いことを示します。

isLost

LOST であるか否かが格納されます。

定数名	説明
TRUE	LOST しました。
FALSE	LOST していません。

isDisconnect

測定間隔が 10 秒以上の場合、断線検出結果が格納されます。

定数名	説明
TRUE	断線しています。
FALSE	断線していません。

error

測定値が測定可能範囲外か否かが格納されます。

値	説明
0	測定値は測定可能範囲内です。
1	測定値は上限値を超えています。

値	説明
2	測定値は下限値を超えています。

dataCount
pData のデータ数です。

pData
計測した測定データが格納されます。

(戻り値)
なし

3.9 ステータスの読み出し

◆ AirLogger_GetState

(概要)

API の状態を取得します。

(書式)

```
int AirLogger_GetState(void);
```

(パラメータ)

なし

(戻り値)

定数名	説明
AIRLOGGER_STATE_NOT_INITIALIZED	初期化処理が実行されていない状態を示します。 AirLogger_Initialize() を実行してください。
AIRLOGGER_STATE_NOT_READY	PC 側通信ユニットが差し込まれていない等の理由で、PC 側通信ユニットが認識できない状態を示します。
AIRLOGGER_STATE_READY	計測可能な状態を示します。
AIRLOGGER_STATE_MEASURING	測定をしている状態を示します。

3.10 計測の停止

◆ AirLogger_StopMeasurement

(概要)

測定データの計測を停止します。

(書式)

```
int AirLogger_StopMeasurement(void);
```

(パラメータ)

なし

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	測定データの計測が正しく停止されました。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、測定データの計測を停止できませんでした。
AIRLOGGER_COMMUNICATION_ERROR	PC 側通信ユニットとの通信に失敗したため、測定データの計測を停止できませんでした。 PC 側通信ユニットが PC の USB 端子に正しく接続されていることを確認してください。

◆ AirLogger_SetStopUnitCallback

(概要)

計測を停止したユニット番号を読み出します。

本関数は、測定間隔 100msec 未満の測定ユニットのみユニット番号が読めます。

(書式)

```
int AirLogger_SetStopUnitCallback(AirLogger_StopUnitCallback pFunc);
```

(パラメータ)

pFunc

測定ユニットから測定停止結果を受信した際に呼び出されるコールバック関数のポインタを指定します。

コールバック関数の詳細は、「AirLogger_SendMeasurementConditonCallback()」を参照してください。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	測定条件の転送が正しく開始されました。
AIRLOGGER_INVALID_PARAMETER	以下が指定されたため、停止したユニット番号が読めませんでした。 <ul style="list-style-type: none"> pFunc が NULL である。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、止したユニット番号が読めませんでした。

◆ AirLogger_StopUnitCallback

(概要)

計測停止したユニット番号を読み出します。

本関数は、測定間隔 100msec 未満の測定ユニットのみユニット番号が読めます。

(書式)

```
(*AirLogger_StopUnitCallback)(int unitNumber);
```

(パラメータ)

unitNumber

1～100 : 計測停止したユニット番号です。

0 : 全ユニットに測定条件の設定が成功しました。

-1 : タイム・アウトの時間内(2 分)に計測を停止しないユニットがありました。

(戻り値)

なし

◆ AirLogger_CancelStop

(概要)

計測を停止する処理を中止します。

本関数は、測定間隔 100msec 未満の測定ユニットのみ有効です。

(書式)

```
int AirLogger_CancelStop(void);
```

(パラメータ)

なし

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	計測を停止する処理が正しく実行されました。
AIRLOGGER_COMMUNICATION_ERROR	PC 側通信ユニットとの通信に失敗したため、中止できませんでした。

3.11 LOST データの受信開始

◆ **AirLogger_StartLoadingLostData**

(概要)

LOST データの受信を開始します。

(書式)

```
int AirLogger_StartLoadingLostData(AirLogger_WM2000_Callback pFunc, int
unitNumbre);
```

(パラメータ)

pFunc

測定ユニットから LOST データを受信した際に呼び出されるコールバック関数のポインタを指定します。

コールバック関数の詳細は、「3.6 測定データの読み出し」を参照してください。

unitNumber

ユニット番号を設定します。

指定可能な値は 1～100 です。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	LOST データの受信が正しく開始されました。
AIRLOGGER_INVALID_PARAMETER	以下のいずれかが指定されたため、LOST データの受信を開始できませんでした。 ・ pFunc が NULL である。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、LOST データの計測を開始できませんでした。
AIRLOGGER_COMMUNICATION_ERROR	PC 側通信ユニットとの通信に失敗したため、LOST データの受信を開始できませんでした。 PC 側通信ユニットが PC の USB 端子に正しく接続されていることを確認してください。

3.12 LOST データの受信停止

◆ **AirLogger_StopLoadingLostData**

(概要)

LOST データの受信を停止します。

(書式)

```
int AirLogger_StopLoadingLostData(void);
```

(パラメータ)

なし

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	LOST データの受信が正しく停止されました。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、LOST データの受信を停止できませんでした。
AIRLOGGER_COMMUNICATION_ERROR	PC 側通信ユニットとの通信に失敗したため、LOST データの受信を停止できませんでした。 PC 側通信ユニットが PC の USB 端子に正しく接続されていることを確認してください。

3.13 通信周波数の設定

◆ AirLogger_GetNoizeLevel

(概要)

PC 側通信ユニット周辺のノイズレベルを取得します。

(書式)

```
int AirLogger_GetNoizeLevel(int channelNumber, int *level);
```

(パラメータ)

channelNumber

対象とする通信チャネル番号を指定します。
指定可能な値は 11～24 です。

level

ノイズレベルが格納されます。
数値が小さいほど通信状態が良好になります。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	ノイズレベルが正しく設定されました。
AIRLOGGER_INVALID_PARAMETER	不正な通信チャネル番号が指定されたため、ノイズレベルを取得できませんでした。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、ノイズレベルを取得できませんでした。

◆ **AirLogger_SetAutoFrequencyNumber**

(概要)

通信チャネルを測定ユニットに設定します。

(書式)

```
int AirLogger_SetAutoFrequencyNumber(int channelNumber, int *failCount, int
*unitNumber);
```

(パラメータ)

channelNumber

設定する通信チャネル番号を指定します。
指定可能な値は 11～24 です。

failCount

設定に失敗したユニットの数が格納されます。

unitNumber

設定に失敗したユニット番号が配列で格納されます。
配列のサイズは、測定ユニット数分確保してください。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	通信チャネルが正しく設定されました。
AIRLOGGER_INVALID_PARAMETER	不正な通信チャネル番号が指定されたため、通信チャネルの設定ができませんでした。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、通信チャネルの設定ができませんでした。
AIRLOGGER_COMMUNICATION_ERROR	PC 側通信ユニットとの通信に失敗したため、通信チャネルの設定ができませんでした。 PC 側通信ユニットが PC の USB 端子に正しく接続されていることを確認してください。

◆ **AirLogger_SetDataReceiverFrequencyNumber**

(概要)

高速データレシーバの通信チャンネルを設定します。

(書式)

```
int AirLogger_DataReceiverFrequencyNumber(int channelNumber);
```

(パラメータ)

channelNumber

設定する通信チャンネルを指定します。

指定可能な値は 1～11 です。

PC 側通信ユニットの通信チャンネルと帯域が重なるチャンネルは、選択不可となります。

PC 側通信ユニットの通信チャンネルと高速データレシーバの通信チャンネルの組み合わせは以下になります。

PC 側通信ユニット	高速データレシーバ	PC 側通信ユニット	高速データレシーバ
11	2 ～ 11	18	1 ～ 4, 9 ～ 11
12	4 ～ 11	19	1 ～ 5, 10, 11
13	5 ～ 11	20	1 ～ 6, 11
14	6 ～ 11	21	1 ～ 7
15	1, 7 ～ 11	22	1 ～ 8
16	1, 2, 7 ～ 11	23	1 ～ 9
17	1 ～ 3, 8 ～ 11	24	1 ～ 10

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	通信チャンネルが正しく設定されました。
AIRLOGGER_INVALID_PARAMETER	不正な通信チャンネルが指定されたため、通信チャンネルの設定ができませんでした。

定数名	説明
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、通信チャネルの設定ができませんでした。
AIRLOGGER_NOT_FIND_ROUTER	高速データレシーバが見つかりませんでした。

◆ AirLogger_GetDataReceiverFrequencyNumber

(概要)

高速データレシーバの通信チャンネルを取得します。

(書式)

```
int AirLogger_DataReceiverFrequencyNumber(int* channelNumber);
```

(パラメータ)

channelNumber

取得した通信チャンネルを格納する領域です。

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	通信チャンネルが正しく取得されました。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、通信チャンネルの設定ができませんでした。
AIRLOGGER_NOT_FIND_ROUTER	高速データレシーバが見つかりませんでした。

3.14 測定ユニットの強制終了

◆ **AirLogger_StopAllUnit**

(概要)

測定ユニットを強制終了します。
測定ユニットのソフトウェア・スイッチが **OFF** 状態になります。

(書式)

```
int AirLogger_StopAllUnit();
```

(パラメータ)

なし

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	測定ユニットの強制終了が正しく実行されました。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、強制終了が正しく実行できませんでした。
AIRLOGGER_COMMUNICATION_ERROR	PC 側通信ユニットとの通信に失敗したため、強制終了が正しく実行できませんでした。 PC 側通信ユニットが PC の USB 端子に正しく接続されていることを確認してください。

3.15 終了処理

◆ AirLogger_Finalize

(概要)

API の終了処理を実行します。

API で行う処理部分の最後に実行してください。

尚、測定ユニットのソフトウェア・スイッチが **OFF** 状態になります。

(書式)

```
int AirLogger_Finalize(void);
```

(パラメータ)

なし

(戻り値)

定数名	説明
AIRLOGGER_SUCCESS	終了処理が正しく実行されました。
AIRLOGGER_INVALID_STATE	初期化処理が実行されていないため、終了処理を実行できませんでした。

4. サンプル・プログラム

この章では、API を使用し、測定を行うサンプル・プログラム(サンプル 1)と通信周波数を設定するサンプル・プログラム (サンプル 2) を示します。

下記のプログラムは、以下の設定で動作します。

- ユニット 1 : WM2000TA
- ユニット 2 : WM2000TB
- ユニット 3 : WM2000SA
- ユニット 4 : WM2000SB
- ユニット 1 の熱電対は K タイプを使用
- 測定間隔は、ユニット 1 とユニット 2 は 1S、ユニット 3 は 100mS、ユニット 4 は 50mS
- 標準出力に各データを 10 秒間出力

・ サンプル 1

```
#include <Windows.h>
#include "AirLoggerAPI.h"
#include <iostream>
#include <string>
using namespace std;

void _stdcall callback(int unitNo, int channelNo, unsigned long long
sequenceNumber, int batteryLevel, int radioWave, BOOL isLost, BOOL
isDisconnection, int error_flag, int data_size, double *Data) {

    cout << endl;
    cout << "UNIT No: " << unitNo << endl;
    cout << "CH NO: " << channelNo << endl;
    cout << "Seq No: " << sequenceNumber << endl;
    cout << "BATTERY:" << batteryLevel << endl;
    cout << "RADIO WAVE:" << radioWave << endl;
    cout << "LOST:" << isLost << endl;
    cout << "DISCONNECT:" << isDisconnection << endl;
    cout << "ERROR:" << error_flag << endl;
    cout << "DATA COUNT:" << data_size << endl;
    for (int i = 0; i < data_size; i++) {
// 本サンプルプログラムは 50mSec サンプリング時間ですので、sequenceNumber +500 になり
// ます
        unsigned long long currentSequenceNumber = sequenceNumber + i *
500;
        cout << "UNIT" << unitNo << " Seq No: " << currentSequenceNumber
<< " CH" << channelNo << " DATA " << Data[i] << endl;
    }
}
```

```

        cout << flush;
    }
    void _stdcall ConditionCallback(int unitNo) {
        if (unitNo == -1) {
            cout << "Setting Condition ERROR" << endl;
            cout << flush;
        }
        else if (unitNo == 0) {
            cout << "Setting Condition OK" << endl;
            cout << flush;
        }
    }
}

void _stdcall WLancallback(int unitNo) {
    if (unitNo == -1) {
        cout << "Setting HighSpeedConnection ERROR" << endl;
        cout << flush;
    }
    else if (unitNo == 0) {
        cout << "Setting HighSpeedConnection OK" << endl;
        cout << flush;
    }
}

void checkError(string functionName, int returnCode) {
    switch (returnCode) {
        case AIRLOGGER_SUCCESS:
            return;
        case AIRLOGGER_INVALID_PARAMETER:
            cout << functionName + ": AIRLOGGER_INVALID_PARAMETER" << endl;
            break;
        case AIRLOGGER_INVALID_STATE:
            cout << functionName + ": AIRLOGGER_INVALID_STATE" << endl;
            break;
        case AIRLOGGER_COMMUNICATION_ERROR:
            cout << functionName + ": AIRLOGGER_INVALID_STATE" << endl;
            break;
    }
    exit(returnCode);
}

void _stdcall StopUnitCallback(int unitNo) {
    if (unitNo == -1) {
        cout << "Stop Unit ERROR" << endl;
        cout << flush;
    }
}

```



```

    }
    else if (unitNo) {
        cout << "Stop Unit OK" << endl;
        cout << flush;
    }
}

int main()
{
    checkError("AirLogger_Initialize", AirLogger_Initialize());
    checkError("AirLogger_SetTemperatureUnit",
        AirLogger_SetTemperatureUnit(AIRLOGGER_TEMPERATURE_UNIT_CELSIUS));

    // UNIT1 の条件設定(WM2000TA)
    checkError("AirLogger_SetUnitSensorType", AirLogger_SetUnitSensorType(1,
    AIRLOGGER_SENSOR_THERMOCOUPLE_K));
    checkError("AirLogger_SetUnitSensorType", AirLogger_SetUnitSensorType(1,
    2, AIRLOGGER_SENSOR_THERMOCOUPLE_K));
    checkError("AirLogger_SetUnitSamplingRate",
    AirLogger_SetUnitSamplingRate(1, AIRLOGGER_SAMPLE_1SEC));

    // UNUT2 の条件設定(WM2000TB)
    for (int channel =1; channel <= 7; channel++){
        checkError("AirLogger_SetUnitSensorType",
    AirLogger_SetUnitSensorType(2, channel, AIRLOGGER_SENSOR_THERMOCOUPLE_K));
    }
    checkError("AirLogger_SetUnitSamplingRate",
    AirLogger_SetUnitSamplingRate(2,AIRLOGGER_SAMPLE_1SEC));

    // UNIT3 の条件設定(WM2000SA)
    checkError("AirLogger_SetUnitSensorType", AirLogger_SetUnitSensorType(3,
    1, AIRLOGGER_SENSOR_STRAIN));
    checkError("AirLogger_SetUnitSamplingRate",
    AirLogger_SetUnitSamplingRate(3, AIRLOGGER_SAMPLE_100MSEC));

    // UNIT4 の条件設定(WM2000SB)
    for (int channel = 1; channel <= 3; channel++) {
        checkError("AirLogger_SetSensorType", AirLogger_SetUnitSensorType(4,
    channel, AIRLOGGER_SENSOR_STRAIN));

        checkError("AirLogger_SetStrainRange", AirLogger_SetStrainRange(4,
    channel, AIRLOGGER_STRAIN_RANGE_2000UST));
    }
}

```

```

    checkError("AirLogger_SetSamplingRate", AirLogger_SetUnitSamplingRate(4,
AIRLOGGER_SAMPLE_50MSEC));

```

```

    // 条件設定を測定ユニットに設定
    checkError("AirLogger_SendMeasurementCondition",
AirLogger_SendMeasurementCondition(ConditionCallback, 60));

    // 高速データレシーバの設定(WM2000SB 以外は不要)
    checkError("AirLogger_EnableHighSpeedConnection",
AirLogger_EnableHighSpeedConnection(WLancallback, 60));

    int* unitNumber = new int[100];
    int* result = new int[100];
    int* size = new int[1];

    // UNIT3/UNIT4 (WM2000SA/WM2000SB のゼロ点調整)
    checkError("AirLogger_ZeroPointAdjustment",
AirLogger_ZeroPointAdjustment (unitNumber, result, size));

    // 測定準備
    checkError("AirLogger_SetMeasurementStandby",
AirLogger_SetMeasurementStandby());

    // 測定開始
    checkError("AirLogger_WM2000_StartMeasurement",
AirLogger_WM2000_StartMeasurement(callback));
    Sleep(10000);
    // 測定終了
    checkError("AirLogger_SetStopUnitCallback",
AirLogger_SetStopUnitCallback(StopUnitCallback));
    checkError("AirLogger_StopMeasurement", AirLogger_StopMeasurement());

    // LOST データの取得
    // 測定ユニットを PC 側通信ユニットの近くに移動
    cout << "0<CR>を入力してください" << endl;
    int lost = getchar();

    for(int unit = 1; unit <= 4; unit++) {
        checkError("AirLogger_StartLoadingLostData",
AirLogger_StartLoadingLostData(callback, unit));
    }

```

```

checkError("AirLogger_Finalize", AirLogger_Finalize());
}

```

・サンプル 2

```

void main()
{
    // 無線チャネルの変更
    checkError("AirLogger_Initialize", AirLogger_Initialize());

    int level = 0;
    int Freq = 0;
    int setCh = 0;
    for (int ch = 11; ch <= 24; ++ch) {
        checkError("AirLogger_GetNoizeLevel", AirLogger_GetNoizeLevel(ch,
&level));
        if (ch == 11) {
            Freq = level;
            setCh = ch;
        }
        else {
            if (level < Freq) { // ノイズの少ないチャネルを検索
                Freq = level;
                setCh = ch;
            }
        }
    }
    int errorCnt;
    int errorUnit[10] = { 0 };

    checkError("AirLogger_SetAutoFrequencyNumberAirLogger",
AirLogger_SetAutoFrequencyNumber(setCh , &errorCnt, errorUnit));

    // 高速データレシーバの通信チャネルを設定
    // PC 側通信ユニットの通信チャネルと帯域が重ならないチャネルを選択
    checkError("AirLogger_SetDataReceiverFrequencyNumber",
AirLogger_SetDataReceiverFrequencyNumber(8));

    // 終了処理
    checkError("AirLogger_Finalize", AirLogger_Finalize());
}

```

}

販売お取引約款

株式会社アドバンテスト(「当社」)の製品の販売およびソフトウェアの使用許諾には、本取引約款(「本約款」)ならびに見積書、注文請書に記載されるその他条件が適用されます。

1. 定義

- a. 「お取引条件」とは、現行のインコタームズで定義された条件のうち、お客様と合意し、かつ見積書に記載された条件を指します。
- b. 「消耗品」とは、製品の通常稼働中に寿命と故障形態が見られるスペアパーツを指します。
- c. 「特注品」とは、お客様の仕様とご要望に従い、当社が製造、開発または改造した製品を指します。
- d. 「納入日」とは、お取引条件に従い事前に取り決めた場所にて、当社がお客様またはお客様の代理人へ製品を引き渡す日を指します。
- e. 「パーツ」とは、製品に組み込まれるか製品と共に使用されるハードウェア付属品、パーツ、アッセンブリ、サブアッセンブリ、および製品とその周辺部の機器を指します。パーツには消耗品を含みません。
- f. 「製品」とは、お客様のご注文に基づき本約款のもとで販売または使用許諾される無線温度ロガー、ならびにその消耗品、パーツ、第三者製品、およびソフトウェアを指します。
- g. 「ソフトウェア」とは、製品に搭載されるか、単独で提供されるかに関わらず、本約款に基づきお客様に提供されるオブジェクトコード形式のコンピュータープログラムまたはファームウェア、および関連文書を指します。
- h. 「仕様」とは、当社が発行する製品に関連する技術情報のうち、お客様に製品を出荷する時点で有効な情報を指します。
- i. 「第三者製品」とは、当社が第三者から購入しお客様に第三者の商標を付して販売する製品を指します。

2. 価格

- a. 価格はお取引条件に従ってお見積りいたします。価格は見積書に記載された期間に限り有効です。前記にかかわらず、当社は予告なく価格および料金を調整することがあります。ただし、かかる調整の発効日前に当社がお請けしたご注文についてはこの適用を受けません。
- b. 価格には消費税、地方消費税や輸入関税を含みません。これらの租税等がお取引に適用される場合は別途申し受けます。お取引に租税等の減免措置が適用される場合は証明書類をご提供ください。

3. ご注文、取り消しおよび返品

- a. お客様の注文書を当社が受諾した時点でご注文が成立します。注文書に記載された納入日は、発注日から3ヶ月以内の日とします。

- b. 製品の返品には当社の承認が必要であり、別途手数料を申し受ける場合があります。
- c. 当社は、お客様固有の構成に応じて製品を生産致します。当初のご発注後に納入日または構成を変更する場合、見積書および価格の改定が必要となる場合があります。
- d. 本約款に別段の記載がない限り、お客様は当社に書面で通知することで製品のご注文を解約できます。当該解約が、ご注文が成立した日から納入予定日の15日前までの間の場合は製品価格の20%、納入予定日の14日前から8日前までの間の場合は製品価格の40%、納入予定日の8日前以降の場合は製品価格の60%を解約料金として申し受けます(ただし消耗品およびパーツは除きます)。前記にかかわらず、第三者製品および特注品のご注文の解約には、製品価格の100%の解約料金を申し受けます。
- e. お客様がご注文の納入日を変更しその後解約された場合でも、上記第3条d項に記載の解約料金を申し受けます。この場合、お客様が最初に納入日の変更を行った日を元に解約料金の算定を行います。お客様が納入日を変更した結果、当初の納入日から1ヶ月以内に製品が引き渡されない場合、ご注文は解約されたものとみなされ、上記第3条d項に記載の解約料金が適用されます。
- a. お客様は、当社製品の購入の際に、当社に対して、特定の統合業務パッケージ(「ERPパッケージ」)の使用を要請した場合、お客様はそのERPパッケージの当社による使用にかかる費用(年間使用料や継続使用料を含みます)を負担するものとします。当社は、その費用を、製品の購入の際の請求書に含めるか、または製品の購入の際の請求書とは別の請求書を発行することでお客様に課すことができます。

4. 出荷および危険負担

1. 当社は、納入日およびその他出荷要件を満たすべく商業上合理的な努力を尽くします。見積書またはその他の方法で提示される納入日は目安であり、当社が納入日およびその他出荷要件を満たすことができない場合には、お客様との合意により代替条件を定めるものとします。合意に達することができない場合、お客様の解決手段はご注文の取り消しに限られます。
- a. 損失および損害の危険負担は、お取引条件に従って当社が同意した所在地においてお客様に移転します。

2. 検収

- a. 製品の検収は、お取引条件に従った納入により完了とします。
- b. 特殊な検収条件は、お客様の注文書受領前に当社が書面で同意している場合に限り適用されます。

3. お支払い

- a. 支払条件は、当社の与信審査を満たすものとします。お客様は、製品の代金を、当社請求日から30日以内にお支払いください。お客様の財務状況または従前の支払記録によつ

ては、当社は与信または支払条件を変更することがあります。

- b. お支払いの遅延については、1ヶ月当たり1.5%の支払遅延料を申し受けることがあります。
- c. お支払いや本約款に基づく義務をお客様が履行せず、書面による通知から5日が経過してもかかる不履行が解消されない場合、当社は義務の履行を一時停止または中止することがあります。
- d. 本約款における別段の記載にかかわらず、法律により許容される範囲において、お客様は当社に対し、製品の購入価格のお支払いに対する担保として、お支払いが完了するまでの間、当該製品に対して譲渡担保権を付与するものとします。

4. 保証

- a. お客様と別段の合意が無い限り、当社は保証期間中、製品（消耗品および第三者製品を除きます）が材料および製造上の瑕疵が無く、仕様に適合することを保証します。保証期間は以下のとおりとします。

(i)製品:納入日から12ヶ月間

- b. 当社は、ソフトウェアが当社指定のハードウェア上で適切にインストールされ使用された場合には、材料および製造上の瑕疵を原因としてプログラムを実行できない事態が発生しないこと、ならびにソフトウェアが納入日時時点で存在する仕様およびその他の文書に実質的に適合することを保証します。保証期間は納入日から12ヶ月間とします。その他の保証の制限に加え、当社は、ソフトウェアがお客様の選択したハードウェアとソフトウェアの組み合わせにおいて稼働すること、また、お客様により個別に指定された要件を満たすことを保証しません。
- c. 本保証（およびかかる保証に関する当社の義務）は、当社の事前の書面による承諾なしに(i)製品が日本国外に移動された場合または(ii)製品がお客様から第三者に対して転売された場合には、終了し無効となります。
- d. 本約款に定められる保証はお客様のみに適用され、いかなる第三者にも適用されません。
- e. 本保証は、以下のいずれかの事由に起因して生じる製品またはその部品もしくは構成品の瑕疵または損傷に対しては適用されません。

(i)お客様または第三者(当社とその代理人を除きます)が、不適切または不十分な態様で、製品の保守、設置環境の準備、操作、改造、運搬または保管を行った場合

(ii) 製品を、仕様、取扱説明書、もしくは当社が書面で推奨するのとは異なる運用条件または環境において使用した場合。かかる事由には、(1)製品が許容範囲を超えた物理的ストレスまたは電圧にさらされた場合、および(2)腐食性ガスまたは埃の多い環境にさらされたことによって電気回路の腐食またはその他の劣化が加速した場合を含みます。

(iii) 製品を、当社が供給または書面で推奨したソフトウェア、インターフェイス、製品または部品以外のソフトウェア、インターフェイス、製品または部品とあわせて使用した場合

(iv) 製品に、(1)お客様が提供した部品もしくは構成部品、または(2)お客様の要請もしくは指示を受けて、あるいはお客様が提供した仕様もしくは設計に基づき第三者が提供した部品もしくは構成部品を組み込んだ場合(かかる部品または構成品の機能が低下した場合を含みます)

(v) お客様が提供した仕様又は設計を当社が組み込みまたは使用した場合

(vi) 第12条a項に記載の不可抗力事態が発生した場合

(vii) お客様または当社以外の第三者が過失による作為または不作為を行った場合

- f. ソフトウェア以外の製品について、保証期間中に第7条a項に記載の瑕疵または仕様への不適合の通知を当社が受けた場合、当該製品の交換に限り実施いたします。合理的な期間内に当該製品を交換することが不可能であると当社が判断した場合、速やかに返品を受けた上で、購入価格から合理的な減価償却費を差し引いた金額を返金いたします。ソフトウェアについて瑕疵の通知を受けた場合、当社は速やかに修正されたソフトウェアをご提供すべく商業上合理的な努力を尽くします。

- g. お客様と別段書面で合意した場合を除き、当社はお客様が購入する消耗品が納入時点で材料および製造上の瑕疵が無いことを保証します。当社は、納入時点で瑕疵が発見された場合、当該消耗品の交換に限り実施いたします。

- h. 当社は、製品の稼働が全く中断されないこと、またはエラーが皆無であることは保証しません。

- i. 法律により許容される限度において、当社は第三者製品について、当社商標が付された製品に組み込まれたものであっても、保証またはサポートしません。当社は第三者製品の一切を現状有姿で提供します。ただし、製造元または供給元が独自の保証を提供する場合があります。

- j. 保証サービスに際し、保証の対象とならない物品はお客様が取り除いてください。当社が取り除き追加作業が生じる場合、当該時点の標準的サービス料金で計算された追加料金をご請求する場合があります。

- k. 当社はお客様自身のファイル、データまたはプログラムの喪失や改変について責任を負わないものとし、これらを復元するための手順はお客様が当社製品とは別に維持管理してください。当社従業員または下請業者の健康に危害をもたらす可能性がある環境で製品が使用されている場合には、その旨を当社に通知するものとします。その場合、当社は保証サービスの提供を拒否、またはかかる製品を当社監督下に置くようお客様に要請する場合があります。

- l. 当社は、明示、黙示、または法律上のものであるかを問わず、本約款に定める事項以外の保証はいたしません。

- m. 当社製品は、核施設の運転、航空機の運航またはその通信システム、航空管制、軍事システム、直接的な救命装置

など、製品の不具合が死亡や深刻な人身傷害または重大な身体もしくは環境に対する損害につながるような無故障動作が要求される用途(「高リスク活動」)のために設計されており、これら用途での利用を意図していません。**当社は、高リスク活動に対する適合性について、明示、黙示を問わず保証しません。**

5. ライセンス

- a. 当社はお客様に対して、ソフトウェアと共に提供される文書に従ってお客様の内部使用目的のために当社の製品上でソフトウェアを使用するための非独占的なライセンス(ただし再許諾権を含みません)を供与します。ソフトウェア付随の文書、および当社見積書または注文請書には、当社または当社への製品供給者が提供する特定のソフトウェアに適用される追加的ライセンス条件を含める可能性があり、それらは、本項の一般的ソフトウェアライセンス条件より優先されます。ソフトウェアをインストール、コピーまたはその他の方法で使用するにより、お客様はかかる追加的ライセンス条件を読み、理解し、その条件に拘束されることに同意したものとみなされます。適用されるライセンスについて記載した文書が無い場合、お客様は、お客様が所有しているコンピューターに限りソフトウェアをコピーして使用すること、または見積書に記載のその他の方法でソフトウェアを使用する権利を許諾されます。
- b. 当社がお客様に対して永続的使用権を許諾するソフトウェアは、譲受人の氏名、名称、所在地をご連絡いただくことを条件として譲渡可能とします。譲受人は、ソフトウェアの譲渡前に当社のソフトウェアライセンス条件に同意するものとします。お客様のライセンス条件は合併または会社分割により譲り受けた譲受人に対しても拘束力を有し、ここにその旨の通知がなされたものとします。
- c. 本約款における別段の規定にかかわらず、当社が非永続的な使用権を許諾するソフトウェア(期間限定ライセンス等)は、譲渡不可とします。
- d. ソフトウェアは、当社または当社への製品供給者が所有権および著作権を有します。当社および当社への製品供給者がソフトウェアに対するすべての権利、権原および利益を留保します。本ライセンス条件に対する違反があった場合、当社への製品供給者はソフトウェアに対して有する権利を保護することができます。お客様は、ソフトウェア上またはソフトウェア内に含まれているあらゆる著作権の表示またはその他の専有財産に関する表示を、ソフトウェアのコピー(全体のコピーであるか一部のコピーであるかを問いません)に複製して付すものとします。
- e. 法律で認められている場合を除き、お客様は当社の書面による同意なしにソフトウェアを逆アセンブル、修正、または二次的著作物の作成をしてはなりません。お客様はソフトウェアを公的または分散的なネットワークに対してコピー、アップロードまたは頒布することはできません。
- f. 本ライセンス条件に対する違反があった場合、当社はその旨通知した上でお客様のライセンスを終了することができます。終了の通知を受けた場合、お客様はソフトウェアのすべてのコピーを破棄して書面でその旨を当社に証明するものとします。

g. ソフトウェアは「商用コンピューターソフトウェア」です。アメリカ合衆国政府に許諾されるソフトウェアおよび技術データに関する権利は、慣習的にエンドユーザーであるお客様に対して供与される権利に限られます。アメリカ合衆国政府による使用、複製、または開示には、本約款に記載の制限、ならびに連邦調達規則(「FAR」)12.211(技術データ)および12.212(コンピューターソフトウェア)、FAR 27.405(b)(2)、FAR 52.227-19、またはFAR 52.227-14(ALT III)、ならびに国防省調達規則(「DFARS」)252.227-7015(技術データ - 市販品)およびDFARS 227.7201乃至227.7202-4、またはそれらの後継的な規定が適用されます。

6. 知的財産権侵害に対する請求

- a. お客様が書面で速やかに当社に通知し、当社に協力し、当社に防御および解決に関するすべての権利を付与することを条件として、法律上認められている範囲で、当社はIP損失に関してお客様に対してなされる請求を防御し、これを解決するものとします。「IP損失」とは、本約款に基づいて納入された製品(特注品を除きます)が、お客様が使用する国または購入された国において、知的財産権を侵害しているとの請求(「IPに関する請求」)を受けることから生じる費用、和解金および裁判所の判決による損害賠償金をいいます。
- b. IPに関する請求が行われるか、行われる可能性が高い場合、当社は自らの選択において、権利を侵害していると主張されている製品を改修するか、必要なライセンスを調達するか、または製品を権利侵害していない代替品と交換することができます。これらいずれの選択肢でも合理的に解決できないと当社が判断した場合、お客様の購入価格から減価償却費(5年にわたる定額法減価償却)を差し引いた金額(ただし、無線温度ロガーに付属するソフトについてはUSD1,500)にて製品を買い戻します。
- c. 当社は、以下のいずれかの事由に起因するIP損失についてはお客様に対する補償義務を負いません。
 - (1) お客様の設計、仕様、指図または技術情報を当社が遵守または使用した場合
 - (2) お客様または第三者が製品の改造を行った場合
 - (3) 仕様で禁じられている方法で製品を使用した場合
 - (4) 当社が供給していないものと共に製品を使用した場合
- b. 本約款は知的財産権の侵害に関する主張に関して当社が負う責任をすべて記載したものです。

7. 免責、賠償責任

(ア) 当社、当社の関連会社、下請業者および製品供給者は、その予見の可能性の有無を問わず、いかなる間接損害、付随的損害、特別損害、結果的損害または懲罰的損害について、契約違反、保証違反、不法行為(過失を含みます)、その他の法的根拠に関わらず、また、本約款に記載された救済手段がその本質的な目的を達成することができない場合であっても、お客様または第三者に対して一切の責任を負いかねます。損害にはダウンタイムコスト、データ喪失、代替品調達費用、復旧費用、逸失利益を含みます。

a. 法律が免責を許容する範囲において、不法行為または契約に基づく請求かを問わず、すべての請求原因について、本約款に基づく当社の累積的な責任は、その責任を発生させた製品に対してお客様が支払った金額を限度とします。ただし、第7条の保証に基づき返金を行う当社の義務は、製品の購入価格を上限とします。本第10条に定める制限は、人身傷害または死亡に関する損害には適用されません。

b. 本約款に規定する責任が、当社のお客様に対する全責任とします。

8. 解除

倒産関連の法律で禁じられない限り、一方当事者が以下のいずれかの事由に該当する場合、他方当事者は、本約款に基づく未履行の義務を取り消すことができます。(i)支払停止、支払不能もしくは債務超過となるか、もしくは手形交換所による不渡処分を受けたとき、(ii)差押、仮差押、仮処分もしくは租税延滞処分その他これに準ずる処分を受けるか、会社更生手続や再生手続の開始、破産もしくは競売の申立を受けるか自らしたとき、(iii)営業の廃止もしくは会社の解散を決議するか、または継続的な営業を中止したとき、(iv)営業の全部または重要な一部を譲渡する決議をしたとき。

9. 一般条項

(イ) 当社は、不可抗力(労働争議、自然災害、供給者や下請業者もしくは運送業者による遅延、電力やその他のエネルギー不足、テロ行為、政府活動、火災、爆発、地質変動、暴風雨、洪水、地震、津波、疫病、放射能汚染、落雷、戦争行為、またはお客様の作為もしくは不作為(不払いや検収書類の不作成等)を含みますが、これらに限りません。)(i)に起因または関連して生じる本約款に基づく義務の履行遅延または不履行について、責任を負わないものとします。

a. 本約款に従った義務の履行前または履行中に、本約款に含まれる条件が不可抗力(経済環境の激変を含む)によって公正性を失った場合、両当事者は協議のうえ必要に応じて本約款の一部またはすべての条件を見直すものとします。

b. お客様は当社の事前の書面による承諾がない限り本約款により生じる一切の権利義務の全部または一部を譲渡することはできません。かかる譲渡を試みた場合、その譲渡は無効とします。

(ウ) お客様は、製品または技術情報を輸出、再輸出、譲渡または輸入する場合、お客様の責任で日本国およびアメリカ合衆国その他の国の関連法規を遵守し、必要な輸出許可を取得します。また、お客様は特定の最終需要者および仕向け地向け、または特定の使用目的向けの譲渡、輸出および再輸出を禁じる日本国およびアメリカ合衆国その他の国の関連法規を遵守し、これらの法規に基づき必要に応じて適切な政府からの許可を取得します。それらの法令に違反しているか、またはそのおそれがあると当社が判断する場合、当社は義務の履行を停止する場合があります。当社の輸出コンプライアンスプログラムの一環として、お客様が購入または使用許諾された製品に関連して、最終用途、最終需要者ならびにお客様の輸出法令遵守および輸出方針を明示した文書の提出をお願いする場合があります。

c. 本約款に関連して生じる紛争には日本法が適用されるものとし、両当事者は本約款に起因または関連して生じる請求または訴訟について、東京地方裁判所の専属的裁判管轄に服することに合意します。

d. 本約款の規定のうち、その性質上、製品の販売後も適用されるものは、それらが遂行されるまで有効に存続するものとします。

e. いずれかの当事者が本約款に基づく自身の権利を行使しなかった場合であっても、かかる権利の放棄または失権とはみなされないものとします。

f. お客様は、本約款の条項、あらゆるライセンス契約、その他当社がお客様に開示する非公開情報(技術情報ならびに製品または価格情報に関する文書、ならびに見積書、注文請書および請求書に記載されている諸条件を含みます)を機密として保持し、いかなる第三者にも開示しないものとします。当社とお客様の間では、当社が、本約款に起因または関連して、お客様に開示または提供した当社の機密情報に対するすべての知的財産権および所有権(設計データおよび製造情報に関するすべての権利を含みます)を有するものとします。当社の書面による事前承諾なしに情報が開示された場合、金銭的な損害賠償では十分な救済とならない、修復不能な損害および重大な被害がもたらされる可能性があります。当社は金銭的な損害賠償に加えて差止命令による救済または衡平法上の救済を受ける権利を付与される場合があります。当社の機密情報に対するいかなる権利も、お客様が当社から製品の購入を受けたことを理由にお客様に譲渡されるものではありません。

g. 本約款の規定が違法または強制執行不能と判断された場合でも、残りの規定は完全な効力を有します。

h. 本約款には、国際物品売買契約に関する国際連合条約は適用されません。

(エ) 本約款の規定は本お取引に関するお客様と当社の合意事項のすべてであり、口頭か書面かを問わず、本約款に関連して以前に当事者間でなされたすべての連絡、表明または合意事項に優先します。本約款に追加されたか、または本約款と異なるお客様所定のご注文条件(注文書に添付されたご注文条件を含みます)は適用されません。お客様が製品を購入またはその使用許諾を受けることによって、お客様は本約款を承諾したものとみなされます。本約款とお客様の注文書またはお客様のその他の文書との間で齟齬が存在する場合、本約款が優先します。本約款の規定に対する変更または修正は、書面で当事者が合意しない限り、効力または拘束力を有しません。

i. お客様は本約款に従って提供される技術、ソフトウェア、試作品およびその他の有形物について、直接または間接的にリバースエンジニアリング、逆アセンブル、または逆コンパイルを行ってはなりません。

(オ) 必要な通知は書面によるものとし、料金前払いかつ受領通知付きで書留郵便もしくは配達証明郵便にて、または配達証明付の宅配便にて、受領当事者の住所に宛てて送付するものとします。