



振動診断ユニット VD-unit

ユーザーズマニュアル

(PCI Express用サンプルプログラム作成解説編)

目次

1. 概要	4
2. 開発環境	4
2.1. システム構成イメージ	4
3. 処理フロー	4
4. ASLINK API	5
4.1. ASLINK API 入手方法	5
4.2. ASLINK API のセットアップについて	5
4.3. ASLINK API のヘルプ参照方法	5
5. サンプルプログラム	6
5.1. サンプルプログラムの共通内容	6
5.1.1. サンプルプログラムの前提条件	6
5.1.2. サンプルプログラムの共通処理	6
5.2. リモートユニット機器パラメータ読み出しサンプルプログラム	7
5.2.1. サンプルプログラム（ソースコード）	7
5.2.2. サンプルプログラム（フローチャート）	8
5.2.3. 使用 API 関数	8
5.2.4. 機器パラメータ読み出しについて	9
5.2.5. 機器パラメータ読み出しデータ（設定フラグ系）	10
5.2.5.1. 機器パラメータ(設定フラグ 1)	10
5.2.5.2. 機器パラメータ(設定フラグ 2)	12
5.2.5.3. 機器パラメータ(設定フラグ 3)	14
5.3. リモートユニット機器パラメータ書き込みサンプルプログラム	16
5.3.1. サンプルプログラム（ソースコード）	16
5.3.2. サンプルプログラム（フローチャート）	17
5.3.3. 使用 API 関数	17
5.3.4. 機器パラメータ書き込みについて	18
5.3.5. 機器パラメータ書き込みデータ（設定フラグ系）	19
5.3.5.1. 機器パラメータ(設定フラグ 1) 書き込みデータ(例)	19
5.3.5.2. 機器パラメータ(設定フラグ 2) 書き込みデータ(例)	19
5.3.5.3. 機器パラメータ(設定フラグ 3) 書き込みデータ(例)	20
5.4. センシングレベルデータ読み取りサンプルプログラム	21
5.4.1. サンプルプログラム（ソースコード）	21
5.4.2. サンプルプログラム（フローチャート）	22
5.4.3. 使用 API 関数	22
5.4.4. リモートユニット読み取りデータについて	23
5.4.4.1. 読み取りデータ（センシングレベル：加速度、速度、変位）	24
5.4.4.2. 読み取りデータ（センシングレベル：詳細状態）	25
5.4.4.3. 読み取りデータ（センシングレベル：状態 + 温度）	26
5.4.4.4. 読み取りデータ（ステータス）	27

5.4.5. センシングレベルデータ読取り処理について.....	27
5.5. I/O データ読み取りサンプルプログラム	28
5.5.1. サンプルプログラム（ソースコード）	28
5.5.2. サンプルプログラム（フローチャート）	29
5.5.3. 使用 API 関数	29
5.5.4. I/O データの読取りデータについて	30
5.5.4.1. I/O データの読取りデータ構造	30
5.5.4.2. 読取りデータ（I/O データ出力数：2bit）	31
5.5.4.3. 読取りデータ（I/O データ出力数：16bit）	31
5.5.4.4. 読取りデータ（I/O データ出力数：32bit）	32
5.5.4.5. 読取りデータ（I/O データ出力数：48bit）	33
5.5.5. I/O データのデータ読取り処理について.....	33

1. 概要

本書は、CP-9011-AL のサンプルプログラムに関する資料です。

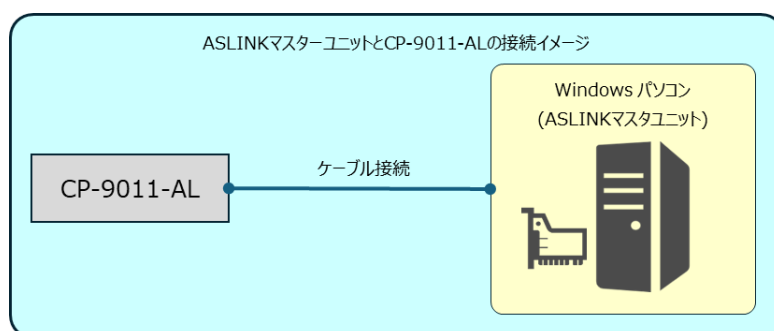
2. 開発環境

本書のサンプルプログラムについては、以下の開発環境で確認しています。

OS	Windows 10
統合開発環境	Visual Studio 2022
実行環境	.NET Framework 4.8.1
使用言語	C#
必要なファイル	casl.dll(ASLINK API), CAsl.cs(C# API インターフェースクラスライブラリ)
使用機器	PCI Express 用 AnyWireASLINK マスタ I/F(型式:B2P8-E01) ※以下、ASLINK マスタユニットと記載します。

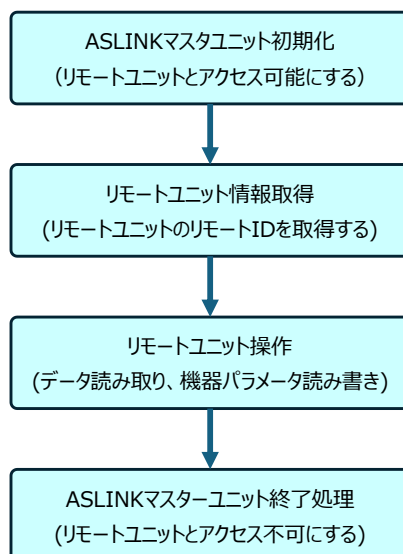
2.1. システム構成イメージ

下図は、ASLINK マスタユニットを搭載したパソコンと CP-9011-AL との接続イメージです。



3. 処理フロー

本章は、パソコンから CP-9011-AL(リモートユニット)を操作するまでの基本的な処理フローを記載しています。



4. ASLINK API

本章では、ASLINK API について記載します。

4.1. ASLINK API 入手方法

ASLINK API(本書ではドライバの C#インターフェースクラスライブラリを指す)の入手については、エニワイヤ社ホームページの下記リンク先から「B2P8-E01 用 PCI 診断ソフト及び Edgecross データコレクタ」をダウンロードしてください。ダウンロードファイルは、「B2P8-E01_Software_20211112.zip」となります。※

URL: <https://www.anywire.jp/products/anywireaslink/products/master04.php>

※本書に記載されている情報は、作成時点のものであり、今後変更される可能性があります

4.2. ASLINK API のセットアップについて

ASLINK API のセットアップについては、ダウンロードファイル「B2P8-E01_Software_20211112.zip」のファイル解凍をしていただき、ファイル解凍先フォルダ以下に含まれる「Readme.txt」ファイルをご確認ください。

解凍先フォルダ
| - B2P8-E01_DRIVER_Installer フォルダ
| - Readme フォルダ
| - ASLWDM フォルダ
| - JPN フォルダ
| - Readme.txt ファイル

4.3. ASLINK API のヘルプ参照方法

ASLINK API のヘルプ参照方法について記載します。

1. ダウンロードファイル「B2P8-E01_Software_20211112.zip」の解凍先フォルダ以下に含まれている「APITool.chm」ファイルをダブルクリックします。

解凍先フォルダ
| - B2P8-E01_DRIVER_Installer フォルダ
| - Help フォルダ
| - ASLWDM フォルダ
| - JPN フォルダ
| - APITool.chm ファイル

2. API ツールのヘルプ画面が起動します。ヘルプから API 関数の詳細についても参照できます。



5. サンプルプログラム

本章は、各種サンプルプログラムについて記載します。

5.1. サンプルプログラムの共通内容

5.1.1. サンプルプログラムの前提条件

本書のサンプルプログラムは、下記に示すように ASLINK API の名前空間を指定、及び Main 関数内に処理を記述することを前提としています。

コード 1 サンプルプログラムの共通

```
1 using System;
2 using CaslCs; //サンプルプログラムでは ASLINK API の名前空間を指定しています
3
4 class Program
5 {
6     static void Main(string[] args)
7     {
8         //サンプルプログラムでは、Main 関数内(ここ)に
9         //処理を記述する前提です
10    }
11 }
```

5.1.2. サンプルプログラムの共通処理

「[3. 処理フロー](#)」のうち、リモートユニット操作を除く処理はサンプルプログラム全てで実行する共通の処理となります。以降に示すサンプルプログラムでは、ここで示す共通処理の記載を省き、下記コード 50 行目に追加するリモートユニット操作処理のみを示します。

コード 2 共通処理の Main 関数サンプルプログラム

```
1 static void Main(string[] args)
2 {
3     // ASLINK デバイス API 関数クラスのオブジェクトの生成
4     Casl caslObj = new Casl();
5
6     // マスタユニット(デバイス)名を設定(デフォルト値"ASL000")
7     string deviceName = "ASL000";
8
9     // マスタユニットの初期化(デバイス ID の取得し、デバイスへのアクセス可能にする)
10    int ret = caslObj.Init(deviceName, out short devId);
11    if (ret != Casl.ASL_ERR_SUCCESS)
12    {
13        // デバイス ID の取得に失敗した時
14        Console.Error.WriteLine($"デバイス{deviceName}の ID 取得に失敗。 戻り値={ret:X}");
15        return;
16    }
17
18    // マスタユニット(デバイス)のエラーステータスをリセット
19    ret = caslObj.ResetErrorStatus(devId);
20    if (ret != Casl.ASL_ERR_SUCCESS)
21    {
22        // エラーステータスのリセットに失敗
23        Console.Error.WriteLine($"リモートユニットの情報リスト取得に失敗。 戻り値={ret:X}");
24        return;
25    }
26
27    // リモートユニットの情報リストを取得(リモートユニット数、リモート ID リストなど)
28    short[] remoteIdList = new short[128];
29    short[] inPointNumList = new short[128];
30    short[] outPointNumList = new short[128];
31    ushort[] statelist = new ushort[128];
32    ushort[] senLvllist = new ushort[128];
33    ret = caslObj.GetSlaveInfo(devId, out short getUnitCnt, remoteIdList, statelist,
34        senLvllist, inPointNumList, outPointNumList, Casl.ASL_ADR_AUTO_REC_ON);
35    if (ret != Casl.ASL_ERR_SUCCESS)
36    {
37        // リモートユニットの情報リスト取得に失敗した時
38        Console.Error.WriteLine($"リモートユニットの情報リスト取得に失敗。 戻り値={ret:X}");
39    }
```

```

40     return;
41 }
42
43 // リモートユニット数とリモートユニット ID リストから有効なリモートユニット ID を抽出して変数に格納
44 short[] remoteIds = new short[getUnitCnt];
45 for (int i = 0; i < remoteIds.Length; i++)
46 {
47     remoteIds[i] = remoteIdList[i];
48 }
49
50 // ここにリモートユニットの操作内容を記述します
51
52 // マスタユニット(デバイス)の終了処理(デバイスへのアクセスを不能にする)
53 ret = caslObj.Exit(devId);
54 if (ret != Casl.ASL_ERR_SUCCESS)
55 {
56     Console.Error.WriteLine($"デバイス{devId}の終了処理に失敗。戻り値={ret}");
57 }
58
59 return;
60 }

```

5.2. リモートユニット機器パラメータ読み出しサンプルプログラム

リモートユニット（CP-9011-AL）の機器パラメータの読み出しサンプルプログラムについて記載します。

5.2.1. サンプルプログラム（ソースコード）

以下は、マスタユニット(デバイス名:ASL000)からリモートユニット（CP-9011-AL）にアクセスして、CP-9011-AL の機器パラメータの設定フラグ 1(パラメータ番号 1)を読み出すサンプルプログラムです。

コード 3 リモートユニットの機器パラメータを読み出すサンプルプログラム

```

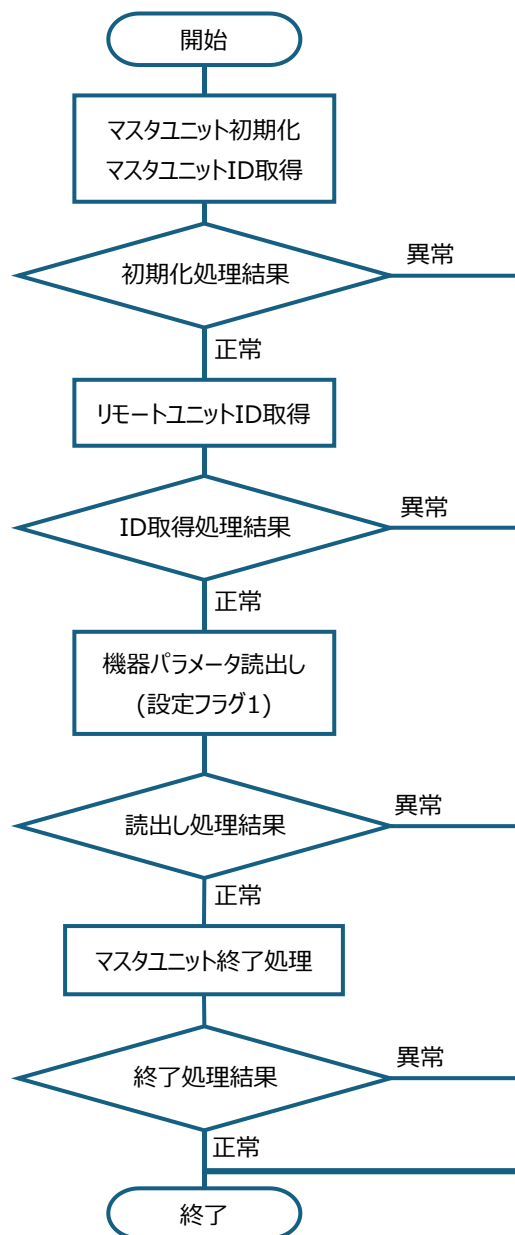
1 // 共通処理 コード 2 の 3～48 行目をここに記述する(マスタユニット初期化～リモートユニット情報 ID 取得)
2
3 // リモートユニットごとに機器パラメータを読み出す
4 // ここでは認識している全てのリモートユニットについて
5 // 機器パラメータ(設定フラグ 1)を読み出す場合を例として示す
6 short paramNo = 1;
7 foreach (var remoteId in remoteIds)
8 {
9     ret = caslObj.ReadSlaveParam(devId, remoteId, paramNo, out ushort paramVal);
10    if (ret != Casl.ASL_ERR_SUCCESS)
11    {
12        // 機器パラメータ(設定フラグ 1)の読み出しに失敗した時
13        Console.Error.WriteLine(
14            $"リモートユニット ID{remoteId}の機器パラメータ{paramNo}の取得に失敗。戻り値={ret}"
15        );
16    }
17    else
18    {
19        //機器パラメータ(設定フラグ 1)の読み出しに成功したので表示する
20        Console.WriteLine($"リモートユニット ID{remoteId} 機器パラメータ{paramNo}={paramVal}");
21    }
22 }
23
25 // 共通処理 コード 2 の 52～59 行目をここに記述する(マスタユニットの終了処理)

```

※リモートユニット ID の取得に失敗するときは ASLINK API のヘルプを参考にし、診断プログラムにて確認してください。

5.2.2. サンプルプログラム（フローチャート）

下図は、マスタユニット(デバイス名:ASL000)からリモートユニット（CP-9011-AL）にアクセスして、CP-9011-AL の機器パラメータの設定フラグ 1(設定フラグ 1)を読み出すプログラムのフローチャートです。



5.2.3. 使用 API 関数

読み出しサンプルプログラムで使用している ASLINK 製 API 関数について記載します。

API 関数の詳細については、「[4. ASLINK API](#)」をご参照ください。

API 関数※	概要
AslInit	マスタユニットからリモートユニットへのアクセスを可能にします。
AslResetErrorStatus	マスタユニットのエラーステータスをリセット
AslGetSlaveInfo	マスタユニットが認識しているリモートユニットの情報を取得します。
AslReadSlaveParam	リモートユニットの機器パラメータを読み出します。
AslExit	マスタユニットのリモートユニットへのアクセスを終了します。

※ASLINK API(ドライバの C# インターフェースライブラリ)ではプレフィックス"Asl"は除いた名称で使用してください

5.2.4. 機器パラメータ読出しについて

リモートユニット（CP-9011-AL）の機器パラメータの読み出しについては、マスタユニット ID とリモートユニット ID の他にパラメータ番号が必要になります。また機器パラメータの読出しデータは、整数値型データ(符号無 16bit)になります。

下図は、API ツールのヘルプ画面※から抜粋した API 関数(AslReadSlaveParam)の詳細です。

AslReadSlaveParam

機能

指定スレーブユニットに設定されている機器パラメータを取得します。

書式

Ret = **AslReadSlaveParam**(Id, SlaveId, ParamNo, Param)

引数

Id [VB.NET: Short] [C, C++: short *] [C#: out short]
AslInitで取得したデバイスIDを指定します。

SlaveId [VB.NET: Short] [C, C++: short] [C#: short]
スレーブユニットIDを指定します。

ParamNo [VB.NET: Short] [C, C++: short] [C#: short]
パラメータNoを指定します。

設定可能範囲: 1 - 19

Param [VB.NET: UShort] [C, C++: unsigned short *] [C#: out ushort]
スレーブユニットの機器パラメータを格納する変数のアドレスを指定します。

戻り値

Ret [VB.NET: Integer] [C, C++: long] [C#: int]

<リモートユニット（CP-9011-AL）の機器パラメータとパラメータ番号>

パラメータ番号	リモートユニット機器パラメータ名
1	設定フラグ 1
2	設定フラグ 2
3	設定フラグ 3
4	振動ベースリミットレベル
5	振動アッパーリミットレベル
6	振動ロウワーリミットレベル
7	振動計測バンド下限値
8	振動計測バンド上限値
9	温度警告リミットレベル
10	振動計測フレーム時間

※API ツールのヘルプ画面では、リモートユニットはスレーブユニットと記載されています

5.2.5. 機器パラメータ読出しデータ（設定フラグ系）

機器パラメータの読出しデータ(設定フラグ系)については、読出しデータの整数値型データ(符号無 16bit)から更にビット単位でパラメータを設定値する必要があります。

5.2.5.1. 機器パラメータ(設定フラグ 1)

機器パラメータ(設定フラグ 1)のビット単位のパラメータについて記載します。

＜読出しデータ(設定フラグ 1)ビット単位＞

ビット	10 進表現	設定フラグ内パラメータ
0	1	センシングレベル振動レベル
1	2	
2	4	
3	8	I/O データワード 1 振動レベル
4	16	
5	32	I/O データワード 2 振動レベル
6	64	
7	128	I/O データワード 3 振動レベル
8	256	
9	512	ティーチング・アラート振動レベル
10	1024	
11	2048	ティーチング・アラート振動パラメータ
12	4096	
13	8192	ティーチングボタン操作
14	16384	
15	32768	

＜センシングレベル振動レベル＞

(bit1)	(bit0)	10 進表現	設定項目
0	0	0	RMS
0	1	1	0-P
1	0	2	P-P
1	1	3	C.F.

＜I/O データワード 1 振動レベル＞

(bit3)	(bit2)	10 進表現	設定項目
0	0	0	RMS
0	1	4	0-P
1	0	8	P-P
1	1	12	C.F.

＜I/O データワード 2 振動レベル＞

(bit5)	(bit4)	10 進表現	設定項目
0	0	0	RMS
0	1	16	0-P
1	0	32	P-P
1	1	48	C.F.

＜I/O データワード 3 振動レベル＞

(bit7)	(bit6)	10 進表現	設定項目
0	0	0	RMS
0	1	64	0-P
1	0	128	P-P
1	1	192	C.F.

＜ティーチング・アラート振動レベル＞

(bit9)	(bit8)	10 進表現	設定項目
0	0	0	RMS
0	1	256	0-P
1	0	512	P-P
1	1	768	C.F.

＜ティーチング・アラート振動パラメータ＞

(bit11)	(bit10)	10 進表現	設定項目
0	0	0	加速度
0	1	1024	速度

＜ティーチングボタン操作＞

(bit13)	(bit12)	10 進表現	設定項目
0	0	0	操作有効
0	1	4096	操作無効

＜動作モード＞

(bit15)	(bit14)	10 進表現	設定項目
0	0	0	計測
0	1	16384	ティーチング

<パラメータ設定値識別(例)>

- 機器パラメータ(設定フラグ 1)の読出しデータが設定値(768)のとき

<ビット単位のデータ状態>

bF	bE	bD	bC	bB	bA	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

<設定フラグ 1 パラメータ>

パラメータ	設定値	識別ビット	読出しデータ
センシングレベル振動レベル	RMS	b0, b1	0
I/O データワード 1 振動レベル	RMS	b2, b3	0
I/O データワード 2 振動レベル	RMS	b4, b5	0
I/O データワード 3 振動レベル	RMS	b6, b7	0
ティーチング・アラート振動レベル	C.F.	b8, b9	768
ティーチング・アラート振動パラメータ	加速度	bA, bB	0
ティーチングボタン操作	操作有効	bC, bD	0
動作モード	計測	bE, bF	0
		合計	768

5.2.5.2. 機器パラメータ(設定フラグ 2)

機器パラメータ(設定フラグ 2)のビット単位のパラメータについて記載します。

<読出しデータ(設定フラグ 2)ビット単位>

ビット	10 進表現	設定フラグ内パラメータ
0	1	センシングレベル出力タイプ
1	2	
2	4	
3	8	
4	16	I/O データワード 1 振動パラメータ
5	32	
6	64	I/O データワード 2 振動パラメータ
7	128	
8	256	I/O データワード 3 振動パラメータ
9	512	
10	1024	I/O データ 2bit 設定時 1bit 目
11	2048	
12	4096	I/O データ 2bit 設定時 2bit 目
13	8192	
14	16384	I/O データ点数
15	32768	

<I/O データワード 1 振動パラメータ>

(bit5)	(bit4)	10 進表現	設定項目
0	0	0	加速度
0	1	16	速度
1	0	32	変位

<I/O データワード 2 振動パラメータ>

(bit7)	(bit6)	10 進表現	設定項目
0	0	0	加速度
0	1	64	速度
1	0	128	変位

<I/O データワード 3 振動パラメータ>

(bit9)	(bit8)	10 進表現	設定項目
0	0	0	加速度
0	1	256	速度
1	0	512	変位

<センシングレベル出力タイプ>

(bit3)	(bit2)	(bit1)	(bit0)	10 進表現	設定項目
0	0	0	0	0	加速度
0	0	0	1	1	速度
0	0	1	0	2	変位
0	0	1	1	3	詳細状態
0	1	0	0	4	状態+温度

<I/O データ 2bit 設定時 1bit 目>

(bit11)	(bit10)	10 進表現	設定項目
0	0	0	動作状態
0	1	1024	ティーチング実施中
1	0	2048	振動レベルアラート
1	1	3072	温度レベルアラート

<I/O データ 2bit 設定時 2bit 目>

(bit13)	(bit12)	10 進表現	設定項目
0	0	0	動作状態
0	1	4096	ティーチング実施中
1	0	8192	振動レベルアラート
1	1	12288	温度レベルアラート

<I/O データ点数>

(bit15)	(bit14)	10 進表現	設定項目
0	0	0	2bit
0	1	16384	16bit
1	0	32768	32bit
1	1	49152	48bit

<パラメータ設定値識別(例)>

- 機器パラメータ(設定フラグ 2)の読出しデータが設定値(4672)のとき

<ビット単位のデータ状態>

bF	bE	bD	bC	bB	bA	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0

<設定フラグ 2 パラメータ>

パラメータ	設定値	識別ビット	読出しデータ
センシングレベル出力タイプ	加速度	b0, b1, b2, b3	0
I/O データワード 1 振動パラメータ	加速度	b4, b5	0
I/O データワード 2 振動パラメータ	速度	b6, b7	64
I/O データワード 3 振動パラメータ	変位	b8, b9	512
I/O I/O データ 2bit 設定時 1bit 目	動作状態	bA, bB	0
I/O I/O データ 2bit 設定時 2bit 目	ティーチング実施中	bC, bD	4096
I/O データ点数	2bit	bE, bF	0
合計			4672

5.2.5.3. 機器パラメータ(設定フラグ 3)

機器パラメータ(設定フラグ 3)のビット単位のパラメータについて記載します。

<読出しデータ(設定フラグ 3)ビット単位>

ビット	10 進表現	設定フラグ内パラメータ
0	1	リセット
1	2	
2	4	
3	8	接続センサ型式
4	16	
5	32	
6	64	リミットレベル調整率
7	128	
8	256	振動レベル更新間隔
9	512	予備
10	1024	
11	2048	
12	4096	
13	8192	ステータス異常解除
14	16384	
15	32768	ティーチング実施

<リセット>

(bit1)	(bit0)	10 進表現	設定項目
0	0	0	未実施
0	1	1	未実施
1	0	2	未実施
1	1	3	実施

<接続センサ型式>

(bit5)	(bit4)	(bit3)	(bit2)	10 進表現	設定項目
0	0	0	0	0	VP-8021
0	0	0	1	4	VP-7021

<リミットレベル調整率>

(bit7)	(bit6)	10 進表現	設定項目
0	0	0	0%
0	1	64	-5%
1	0	128	+5%
1	1	192	+10%

<振動レベル更新間隔>

(bit8)	10 進表現	設定項目
0	0	通常モード
1	256	高速モード

※通常モード時の更新間隔(時間)は、30 ミリ秒です。

※高速モード時の更新間隔(時間)は、機器パラメータ
(設定フラグ 2)の I/O データ点数によって異なります。

I/O データ点数	更新間隔(時間)
2bit	10 ミリ秒
16bit	
32bit	20 ミリ秒
48bit	

※高速モード時は、センシングレベル出力が停止します。

<ステータス異常解除>

(bit14)	10 進表現	設定項目
0	0	未実施
1	16384	実施

※bit14 を"1"に設定し、ステータス異常解除後
bit14 は"0"に戻ります。

<ティーチング実施>

(bit15)	10 進表現	設定項目
0	0	OFF
1	32768	ON

※bit15 を"1"に設定し、ティーチング実施後
bit15 は"0"に戻ります。

<パラメータ設定値識別(例)>

- 機器パラメータ(設定フラグ 3)の読出しデータが設定値(0)のとき

<ビット単位のデータ状態>

bF	bE	bD	bC	bB	bA	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

<設定フラグ 3 パラメータ>

パラメータ	設定値	識別ビット	読出しデータ
リセット	未実施	b0, b1	0
接続センサ型式	VP-8021	b2, b3, b4, b5	0
リミットレベル調整率	0%	b6, b7	0
振動レベル更新間隔	通常モード	b8	0
ステータス異常解除	未実施	bE	0
ティーチング実施	OFF	bF	0
合計			0

5.3. リモートユニット機器パラメータ書き込みサンプルプログラム

リモートユニット（CP-9011-AL）に対する機器パラメータの書き込みサンプルプログラムについて記載します。

5.3.1. サンプルプログラム（ソースコード）

以下は、マスタユニット(デバイス名:ASL000)からリモートユニット（CP-9011-AL）にアクセスして、CP-9011-AL の設定フラグの一部の値を変更した機器パラメータ(設定フラグ 2)を書き込むサンプルプログラムです。

コード 4 リモートユニットの機器パラメータを書き込むサンプルプログラム

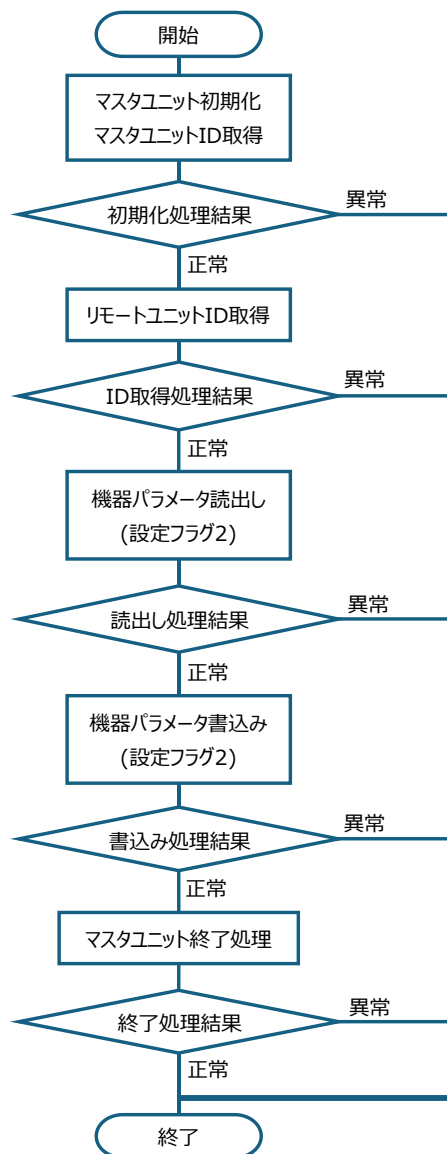
```
1 // 共通処理 コード 2 の 3~48 行目をここに記述する(マスタユニット初期化~リモートユニット情報 ID 取得)
2
3 // リモートユニットごとに機器パラメータを書き込む
4 // ここでは認識している全てのリモートユニットについて
5 // 機器パラメータ(設定フラグ 2)のセンシングレベル出力タイプを「速度」に変更する場合を例として示す
6 short paramNo = 2;
7 foreach (var remoteId in remoteIds)
8 {
9     // リモートユニット内の設定値を読み出す
10    ret = caslObj.ReadSlaveParam(devId, remoteId, paramNo, out ushort paramVal);
11    if (ret != Casl.ASL_ERR_SUCCESS)
12    {
13        // 機器パラメータ(設定フラグ 2)の読み出しに失敗した時
14        Console.Error.WriteLine(
15            $"リモートユニット ID{remoteId}の機器パラメータ{paramNo}の取得に失敗。戻り値={ret}"
16        );
17        continue;
18    }
19
20    // リモートユニットの設定フラグ 2 のセンシングレベル出力タイプを変更して書き込む
21    paramVal = (ushort)((((int)paramVal & 0xffff0) + 0x02); //下位 4 ビット[3:0]を 0b0010(変位)にする
22    ret = caslObj.WriteSlaveParam(devId, remoteId, paramNo, paramVal);
23    if (ret != Casl.ASL_ERR_SUCCESS)
24    {
25        // 機器パラメータの書き込みに失敗した時
26        Console.Error.WriteLine(
27            $"リモートユニット ID{remoteId}の機器パラメータ{paramNo}の更新に失敗。戻り値={ret}"
28        );
29        continue;
30    }
31 }
32 }
33
34 // 共通処理 コード 2 の 52~59 行目をここに記述する(マスタユニットの終了処理)
```

※リモート ID の取得に失敗するときは ASLINK API のヘルプを参考にし、診断プログラムにて確認してください。

5.3.2. サンプルプログラム（フローチャート）

下図は、マスタユニット(デバイス名:ASL000)からリモートユニット（CP-9011-AL）にアクセスして、CP-9011-AL へ機器パラメータの設定フラグ 1(パラメータ番号 1)を書き込むプログラムのフローチャートです。

※機器パラメータを部分的に変更する場合、サンプルプログラムのように書き込み処理を行う前に、同じ機器パラメータの読み出しが必要になります。



5.3.3. 使用 API 関数

書き込みサンプルプログラムで使用している ASLINK 製 API 関数について記載します。

API 関数の詳細については、「[4. ASLINK API](#)」をご参照ください。

API 関数※	概要
AslInit	マスタユニットからリモートユニットへのアクセスを可能にします。
AslResetErrorStatus	マスタユニットのエラーステータスをリセット
AslGetSlaveInfo	マスタユニットが認識しているリモートユニットの情報を取得します。
AslReadSlaveParam	リモートユニットの機器パラメータを読み出します。
AslWriteSlaveParam	リモートユニットへ機器パラメータを書き込みます。
AslExit	マスタユニットのリモートユニットへのアクセスを終了します。

※ASLINK API(ドライバの C#インターフェースライブラリ)ではプレフィックス"Asl"は除いた名称で使用してください

5.3.4. 機器パラメータ書込みについて

リモートユニット（CP-9011-AL）への機器パラメータの書き込みについては、機器パラメータの読出しと同様にマスタユニット ID とリモートユニット ID の他にパラメータ番号が必要になります。また機器パラメータは、整数値型データ(符号無 16bit)を書込みます。下図は、API ツールのヘルプ画面※から抜粋した API 関数(AslWriteSlaveParam)の詳細です。

AslWriteSlaveParam

機能

指定スレーブユニットに機器パラメータ設定を行います。

書式

Ret = **AslWriteSlaveParam**(Id, SlaveId, ParamNo, Param)

引数

Id [VB.NET: Short] [C, C++: short *] [C#: out short]
AslInitで取得したデバイスIDを指定します。

SlaveId [VB.NET: Short] [C, C++: short] [C#: short]
設定を行うスレーブユニットIDを指定します。

ParamNo [VB.NET: Short] [C, C++: short] [C#: short]
パラメータNoを指定します。

設定可能範囲: 1 - 19

Param [VB.NET: UShort] [C, C++: unsigned short] [C#: ushort]
設定する機器パラメータを指定します。

戻り値

Ret [VB.NET: Integer] [C, C++: long] [C#: int]

<リモートユニット（CP-9011-AL）の機器パラメータとパラメータ番号>

パラメータ番号	リモートユニット機器パラメータ
1	設定フラグ 1
2	設定フラグ 2
3	設定フラグ 3
4	振動ベースリミットレベル
5	振動アッパーリミットレベル
6	振動ロウワーリミットレベル
7	振動計測バンド下限値
8	振動計測バンド上限値
9	温度警告リミットレベル
10	振動計測フレーム時間

※API ツールのヘルプ画面では、リモートユニットはスレーブユニットと記載されています

5.3.5. 機器パラメータ書き込みデータ（設定フラグ系）

機器パラメータ(設定フラグ系)については、ビット単位のパラメータの設定値を整数値型データ(符号無 16bit)へまとめた状態にしてリモートユニット（CP-9011-AL）へ機器パラメータ(設定フラグ系)を書き込みます。

5.3.5.1. 機器パラメータ(設定フラグ 1) 書き込みデータ(例)

機器パラメータ(設定フラグ 1)の書き込みデータの例です。機器パラメータ(設定フラグ 1)のビット単位のパラメータについては、

「[5.2.5.1. 機器パラメータ（設定フラグ 1）](#)」をご参照ください。

＜機器パラメータ(設定フラグ 1)のパラメータ設定状態(例)＞

パラメータ	設定値	識別ビット	書き込みデータ
センシングレベル振動レベル	RMS	b0, b1	0
I/O データワード 1 振動レベル	RMS	b2, b3	0
I/O データワード 2 振動レベル	0-P	b4, b5	16
I/O データワード 3 振動レベル	P-P	b6, b7	128
ティーチング・アラート振動レベル	C.F.	b8, b9	768
ティーチング・アラート振動パラメータ	加速度	bA, bB	0
ティーチングボタン操作	操作有効	bC, bD	0
動作モード	計測	bE, bF	0
合計			912

リモートユニット(CP-9011-AL)の機器パラメータ(設定フラグ 1)には、書き込みデータの合計値“912”を書き込みます。

＜上記パラメータ設定状態のビット単位データ＞

bF	bE	bD	bC	bB	bA	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0

5.3.5.2. 機器パラメータ(設定フラグ 2) 書き込みデータ(例)

機器パラメータ(設定フラグ 2)の書き込みデータの例です。機器パラメータ(設定フラグ 2)のビット単位のパラメータについては、

「[5.2.5.2. 機器パラメータ（設定フラグ 2）](#)」をご参照ください。

＜機器パラメータ(設定フラグ 2)のパラメータ設定状態(例)＞

パラメータ	設定値	識別ビット	読出しデータ
センシングレベル出力タイプ	詳細状態	b0, b1, b2, b3	3
I/O データワード 1 振動パラメータ	加速度	b4, b5	0
I/O データワード 2 振動パラメータ	速度	b6, b7	64
I/O データワード 3 振動パラメータ	変位	b8, b9	512
I/O データ 2bit 設定時 1bit 目	動作状態	bA, bB	0
I/O データ 2bit 設定時 2bit 目	ティーチング実施中	bC, bD	4096
I/O データ点数	48bit	bE, bF	49152
合計			53827

リモートユニット(CP-9011-AL)の機器パラメータ(設定フラグ 2)には、書き込みデータの合計値“53827”を書き込みます。

＜上記パラメータ設定状態のビット単位データ＞

bF	bE	bD	bC	bB	bA	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
1	1	0	1	0	0	1	0	0	1	0	0	0	0	1	1

5.3.5.3. 機器パラメータ(設定フラグ 3) 書き込みデータ(例)

機器パラメータ(設定フラグ 3)の書き込みデータの例です。機器パラメータ(設定フラグ 3)のビット単位のパラメータについては、[「5.2.5.3. 機器パラメータ \(設定フラグ 3\)」](#)をご参照ください。

＜機器パラメータ(設定フラグ 3)のパラメータ設定状態(例)＞

パラメータ	設定値	識別ビット	読出しデータ
リセット	未実施	b0, b1	0
接続センサ型式	VP-8021	b2, b3, b4, b5	0
リミットレベル調整率	-5%	b6, b7	64
振動レベル更新間隔	通常モード	b8	0
ステータス異常解除	実施	bE	16384
ティーチング実施	ON	bF	32768
合計			49216

リモートユニット(CP-9011-AL)の機器パラメータ(設定フラグ 3)には、書き込みデータの合計値“49216”を書込みます。

＜上記パラメータ設定状態のビット単位データ＞

bF	bE	bD	bC	bB	bA	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0

5.4. センシングレベルデータ読み取りサンプルプログラム

リモートユニット（CP-9011-AL）のセンシングレベルのデータ読み取りサンプルプログラムについて記載します。

5.4.1. サンプルプログラム（ソースコード）

以下は、マスタユニット(デバイス名:ASL000)からリモートユニット（CP-9011-AL）にアクセスして、CP-9011-AL のセンシングレベル・ステータスのデータ読み取り、物理値に変換するサンプルプログラムです。

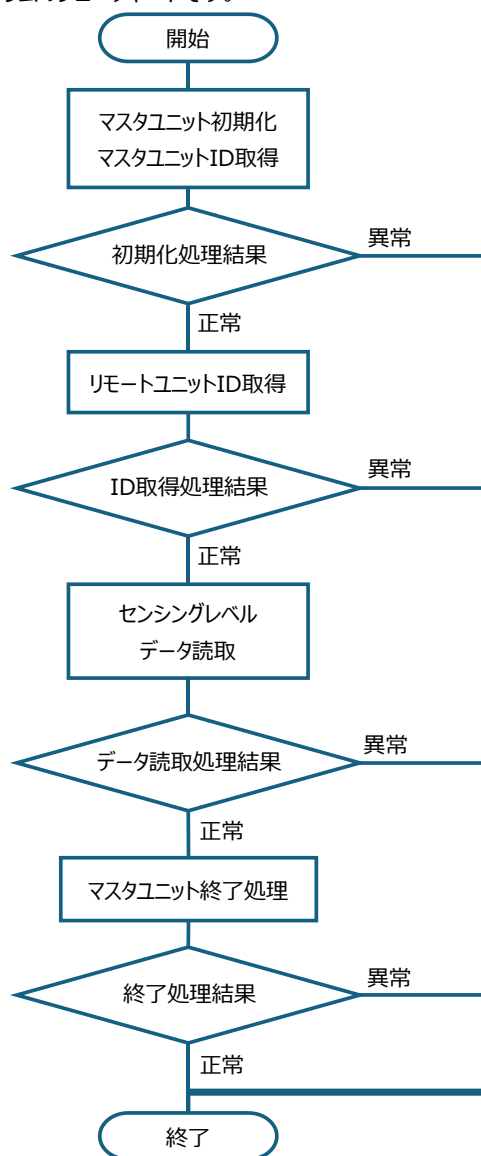
コード 5 リモートユニットのセンシングレベルデータを読み出すサンプルプログラム

```
1 // 共通処理 コード 2 の 3~48 行目をここに記述する(マスタユニット初期化〜リモートユニット情報 ID 取得)
2
3 // リモートユニットごとにセンシングレベルデータを読み取り、物理値に変換する
4 // ここでは機器パラメータ(設定フラグ 2)のセンシングレベル出力タイプを「変位」(=0b0010)
5 // に設定した場合の例を示す
6 foreach (var remoteId in remoteIds)
7 {
8     ushort[] paramVals = new ushort[9];
9     ret = caslObj.GetSlaveNetInfo(devId, remoteId, paramVals);
10    if (ret != Casl.ASL_ERR_SUCCESS)
11    {
12        // 機器パラメータのユニット情報の取得に失敗した時
13        Console.Error.WriteLine($"リモートユニット ID{remoteId}のユニット情報の取得に失敗。戻り値={ret}");
14        continue;
15    }
16
17    // ユニット情報からステータスとセンシングレベルを取り出す
18    ushort remoteState = paramVals[0];
19    short remoteSensLvl = (short)paramVals[1];
20
21    // センシングレベルを物理値に変換する
22    // 機器パラメータ(設定フラグ 2)のセンシングレベル出力タイプの設定によって
23    // 以下の数値で除算する
24    // 「加速度」(=0b0000) 100.0
25    // 「速度」(=0b0001) 100.0
26    // 「変位」(=0b0010) 10.0
27    // 「状態+温度」(=0b0100) 10.0 (センシングレベル[15:4]に対する除数)
28    double dispValue = (double)remoteSensLvl / 10.0;
29
30
31    // ステータスと物理値を表示
32    Console.WriteLine(
33        $"リモートユニット ID{remoteId} : ステータス=0x{remoteState:X4}, 変位値={dispValue}um"
34    );
35 }
36
37 // 共通処理 コード 2 の 52~59 行目をここに記述する(マスタユニットの終了処理)
```

※リモート ID の取得に失敗するときは ASLINK API のヘルプを参考にし、診断プログラムにて確認してください。

5.4.2. サンプルプログラム（フローチャート）

下図は、マスタユニット(デバイス名:ASL000)からリモートユニット（CP-9011-AL）にアクセスして、CP-9011-AL のセンシングレベル・ステータスのデータ読み取りプログラムのフローチャートです。



5.4.3. 使用 API 関数

データ読み取りサンプルプログラムで使用している ASLINK 製 API 関数について記載します。

API 関数の詳細については、「[4. ASLINK API](#)」をご参照ください。

API 関数※	概要
AslInit	マスタユニットからリモートユニットへのアクセスを可能にします。
AslResetErrorStatus	マスタユニットのエラーステータスをリセット
AslGetSlaveInfo	マスタユニットが認識しているリモートユニットの情報を取得します。
AslGetSlaveNetInfo	リモートユニット(CP-9011-AL)の情報を取得します。
AslExit	マスタユニットのリモートユニットへのアクセスを終了します。

※ASLINK API(ドライバの C# インターフェースライブラリ)ではプレフィックス"Asl"は除いた名称で使用してください

5.4.4. リモートユニット読取りデータについて

リモートユニット（CP-9011-AL）のセンシングレベルやステータスの読取りデータについては、整数型データ(符号無 16bit)として取得します。センシングレベルの読み取りデータは、センシングレベル出力タイプ(機器パラメータ)に対応したデータ加工が必要になります。下図は、API ツールのヘルプ画面※から抜粋した API 関数(AslGetSlaveNetInfo)の詳細です。

AslGetSlaveNetInfo

機能

指定したスレーブユニットの情報を取得します。

書式

Ret = **AslGetSlaveNetInfo**(Id, SlaveId, Param)

引数

Id [VB.NET: Short][C, C++: short][C#: short]
AslInitで取得したデバイスIDを指定します。

SlaveId [VB.NET: Short][C, C++: short][C#: short]
情報を取得するスレーブユニットIDを指定します。

Param [VB.NET: UShort][C, C++: unsigned short *][C#: out ushort]
データを格納する9データ分の配列を指定します。

配列添字		
シンボル	値	意味
ASL_SLAVE_PARAM_STATUS	0	ステータス詳細
ASL_SLAVE_PARAM_SENSING_LV	1	センシングレベル
予約	2	-
予約	3	-
予約	4	-
予約	5	-
予約	6	-
予約	7	-
予約	8	-

※API ツールのヘルプ画面では、リモートユニットはスレーブユニットと記載されています

5.4.4.1. 読取りデータ（センシングレベル：加速度、速度、変位）

リモートユニットのセンシングレベルの読取りデータを除算することによって、物理値に変換することができます。
機器パラメータ(設定フラグ 2)※のセンシングレベル出力タイプの設定値("加速度","速度","変位")によって、物理値を計算する除数が異なります。

<センシングレベル出力タイプと読取りデータ>

センシングレベル出力タイプ	読取りデータ(整数型)
加速度	加速度値の 100 倍
速度	速度値の 100 倍
変位	変位値の 10 倍

※機器パラメータ(設定フラグ 2)については、「[5.2.5.2. 機器パラメータ（設定フラグ 2）](#)」をご参照ください。

<計算手順>

- ① 整数型データ(符号付 16bit:short)から浮動小数点データ型(float, double)に変換します。
- ② 浮動小数点データ型(float, double)を適切な除数で除算して物理値を求めます。

<サンプルコード(C#)>

```
1 // ユニット情報からセンシングレベルを取り出す
2 short remoteSensLvl = (short)paramVals[1];
3
4 // ここでは、機器パラメータ(設定フラグ 2)のセンシングレベル出力タイプ設定値
5 // が符号無し整数変数 sensingLvlType に格納されているものとし、物理値を計算
6 // するための除数をセンシングレベル出力タイプ設定値から選択する
7 double divisor = 1.0;
8 switch (sensingLvlType)
9 {
10     // 加速度の場合
11     case 0:
12         divisor = 100.0;
13         break;
14     // 速度の場合
15     case 1:
16         divisor = 100.0;
17         break;
18     // 変位の場合
19     case 2:
20         divisor = 10.0;
21         break;
22 }
23
24 // センシングレベルデータと除数から物理値を求める
25 double physicVal = (double)remoteSensLvl / divisor;
```

5.4.4.2.読取りデータ（センシングレベル：詳細状態）

リモートユニットの機器パラメータ(設定フラグ 2)のセンシングレベル出力タイプに“詳細状態”を設定しているとき、センシングレベルの読取りデータの整数型データ(符号無 16bit)はビット単位でデータの意味が異なるため、読取りデータをビット単位で識別します。

※機器パラメータ(設定フラグ 2)については、「[5.2.5.2. 機器パラメータ（設定フラグ 2）](#)」をご参照ください。

＜読取りデータ(センシングレベル)ビット単位＞

ビット	10 進表現	データ説明
0	1	動作状態
1	2	振動アッパーリミットアラート
2	4	振動ベースリミットアラート
3	8	振動ロウワーリミットアラート
4	16	予備
5	32	予備
6	64	予備
7	128	接地面温度アラート状態
8	256	ティーチング状態
9	512	予備
10	1024	予備
11	2048	予備
12	4096	予備
13	8192	予備
14	16384	予備
15	32768	予備

※読取りデータ内の予備ビットは、固定値(0)になります。

＜動作状態＞

(bit0)	10 進表現	判定値
0	0	その他
1	1	計測時

＜振動アッパーリミットアラート＞

(bit1)	10 進表現	判定値
0	0	OK
1	2	NG

＜振動ベースリミットアラート＞

(bit2)	10 進表現	判定値
0	0	OK
1	4	NG

＜振動ロウワーリミットアラート＞

(bit3)	10 進表現	判定値
0	0	OK
1	8	NG

＜接地面温度アラート状態＞

(bit7)	10 進表現	判定値
0	0	OK
1	128	NG

＜ティーチング状態＞

(bit8)	10 進表現	判定値
0	0	ティーチング未実施
1	256	ティーチング実施中

5.4.4.3. 読取りデータ（センシングレベル：状態+温度）

リモートユニットの機器パラメータ(設定フラグ 2)のセンシングレベル出力タイプに“状態+温度”を設定しているとき、センシングレベルの読取りデータの整数型データ(符号無 16bit)はビット単位でデータの意味が異なるため、読取りデータをビット単位で識別します。また整数型データ(符号無 16bit)に含まれる温度値は 10 倍値になっているため、読取りデータの加工が必要になります。
※機器パラメータ(設定フラグ 2)については、「[5.2.5.2. 機器パラメータ（設定フラグ 2）](#)」をご参照ください。

<読取りデータ(センシングレベル)ビット単位>

ビット	10 進表現	データ説明
0	1	動作状態
1	2	振動レベルアラート状態
2	4	ティーチング状態
3	8	接地面温度アラート状態
4	16	温度値(10 倍) 12bit 固定小数点
5	32	
6	64	
7	128	
8	256	
9	512	
10	1024	
11	2048	
12	4096	
13	8192	
14	16384	
15	32768	

<動作状態>

(bit0)	10 進表現	判定値
0	0	その他
1	1	計測時

<振動レベルアラート状態>

(bit1)	10 進表現	判定値
0	0	OK
1	2	NG

<ティーチング状態>

(bit2)	10 進表現	判定値
0	0	ティーチング未実施
1	4	ティーチング実施中

<接地面温度アラート状態>

(bit3)	10 進表現	判定値
0	0	OK
1	8	NG

<計算手順>

- ① 整数型データ(符号付 16bit:short)を 4 ビット右にシフトします。
- ② 4 ビット右シフト後の整数型データを浮動小数点データ型(float, double)に変換し、除数 10 で除算して温度値を計算します

<サンプルコード(C#)>

```

1 // ユニット情報からセンシングレベルを取り出す
2 short remoteSensLvl = (short)paramVals[1];
3
4 // ここでは、機器パラメータ(設定フラグ 2)のセンシングレベル出力タイプ
5 // は「状態+温度」(4=0b0100)に設定されていることを前提とする
6 // 温度値の除数は 10 である
7 double divisor = 10.0;
8
9 // センシングレベルデータと除数から温度値を求める
10 double tempVal = (double)(remoteSensLvl >> 4) / divisor;
```

5.4.4.4. 読取りデータ（ステータス）

リモートユニット（CP-9011-AL）のステータスについては、読取りデータの整数型データ(符号無 16bit)がビット単位でデータの意味が異なるため、読取りデータをビット単位で識別します。

<読取りデータ(ステータス)ビット単位>

ビット	10 進表現	データ説明
0	1	DP,DN 電圧低下異常
1	2	予約
2	4	予約
3	8	予約
4	16	予約
5	32	予約
6	64	内部通信異常
7	128	
8	256	
9	512	設定異常
10	1024	センサ異常
11	2048	ハードウェア異常
12	4096	予約
13	8192	予約
14	16384	予約
15	32768	予約

<DP,DN 電圧低下異常>

(bit0)	10 進表現	判定値
0	0	正常
1	1	異常検出

<内部通信異常>

(bit8)	(bit7)	(bit6)	10 進表現	判定値
0	0	0	0	正常
0	0	1	64	異常検出
0	1	0	128	異常検出

<設定異常>

(bit9)	10 進表現	判定値
0	0	正常
1	512	異常検出

<センサ異常>

(bit10)	10 進表現	判定値
0	0	正常
1	1024	異常検出

<ハードウェア異常>

(bit11)	10 進表現	判定値
0	0	正常
1	2048	異常検出

5.4.5. センシングレベルデータ読取り処理について

本章のサンプルプログラムではセンシングレベルのデータ読取りを 1 回実行するのみですが、ループ処理内やタイマーイベント処理等でセンシングレベルのデータ読取り(API 関数:AslGetSlaveNetInfo)を繰り返し行うことで、モニタリングの様なプログラムも出来ます。

5.5. I/O データ読み取りサンプルプログラム

リモートユニット（CP-9011-AL）に対する I/O データのデータ読み取りサンプルプログラムについて記載します。

5.5.1. サンプルプログラム（ソースコード）

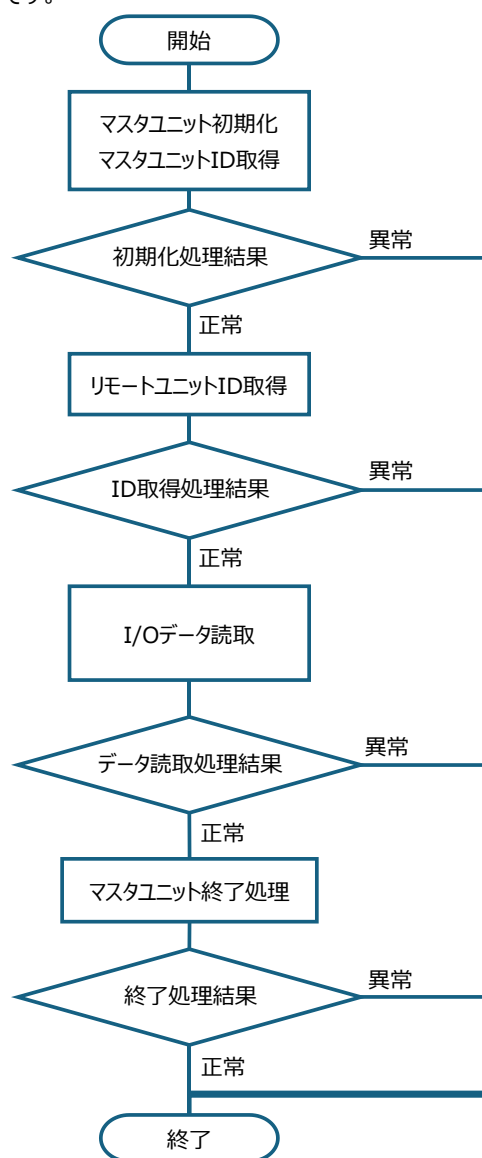
以下は、マスタユニット(デバイス名:ASL000)からリモートユニット（CP-9011-AL）にアクセスして、CP-9011-AL の I/O データのデータ読み取りプログラムのサンプルプログラムです。

```
1 // 共通処理 コード 2 の 3~48 行目をここに記述する(マスタユニット初期化〜リモートユニット情報 ID 取得)
2
3 // リモートユニットの I/O データを取得
4 // ここでは、機器パラメータ(設定フラグ 2)をデフォルト(25156)から
5 // I/O データ点数[15:14]を 48bit(b[11])とした"57924"とした場合の例を示す
6 // この時、I/O データ 1~3 の割当は以下となる
7 // I/O データワード 1 : 加速度 m/s^2(RMS※)
8 // I/O データワード 2 : 速度 mm/s(RMS※)
9 // I/O データワード 3 : 変位 μm(P-P※)
10 // ※各振動レベルの種別は機器パラメータ(設定フラグ 1)に寄る
11 foreach (var remoteId in remoteIds)
12 {
13     uint[] readIoDatas = new uint[2];
14     ret = caslObj.ReadData(devId, remoteId, readIoDatas);
15     if (ret != Casl.ASL_ERR_SUCCESS)
16     {
17         // リモートユニットの I/O データ取得に失敗した時
18         Console.WriteLine($"リモートユニット{remoteId}の I/O データ取得に失敗しました。 戻り値={ret}");
19         continue;
20     }
21
22     // I/O データの読み取りを連結
23     ulong remoteIoData = ((ulong)readIoDatas[1] << 32) | (ulong)readIoDatas[0];
24     // I/O データワード 1[15:0]より加速度を算出
25     double accValue = (double)((short)(remoteIoData & 0xffff)) / 100.0;
26     // I/O データワード 2[31:16]より速度を算出
27     double velValue = (double)((short)((remoteIoData >> 16) & 0xffff)) / 100.0;
28     // I/O データワード 3[47:32]より変位を算出
29     double dispValue = (double)((short)((remoteIoData >> 32) & 0xffff)) / 10.0;
30
31     // I/O データの物理値を表示
32     Console.WriteLine(
33         $"リモートユニット ID{remoteId}:加速度値={accValue}m/s^2, 速度値={velValue}cm/s, 変位値
34         ={dispValue}um"
35     );
36 }
37
38 // 共通処理 コード 2 の 52~59 行目をここに記述する(マスタユニットの終了処理)
```

※リモート ID の取得に失敗するときは ASLINK API のヘルプを参考にし、診断プログラムにて確認してください。

5.5.2. サンプルプログラム（フローチャート）

下図は、マスタユニット(デバイス名:ASL000)からリモートユニット（CP-9011-AL）にアクセスして、CP-9011-AL の I/O データのデータ読み取りプログラムのフローチャートです。



5.5.3. 使用 API 関数

データ読み取りサンプルプログラムで使用している ASLINK 製 API 関数について記載します。

API 関数の詳細については、「[4. ASLINK API](#)」をご参照ください。

API 関数※	概要
AslInit	マスタユニットからリモートユニットへのアクセスを可能にします。
AslResetErrorStatus	マスタユニットのエラーステータスをリセット
AslGetSlaveInfo	マスタユニットが認識しているリモートユニットの情報を取得します。
AslReadData	リモートユニット(CP-9011-AL)の入力データを取得します。
AslExit	マスタユニットのリモートユニットへのアクセスを終了します。

※ASLINK API(ドライバの C# インターフェースライブラリ)ではプレフィックス"Asl"は除いた名称で使用してください

5.5.4. I/O データの読取りデータについて

リモートユニット（CP-9011-AL）の I/O データの読み取りについては、整数型データ(符号無 32bit)の配列を取得します。
読み取りデータは、機器パラメータ(I/O データ出力数, I/O データの振動パラメータなど)に対応したデータ加工が必要になります。
下図は、API ツールのヘルプ画面※から抜粋した API 関数(AslReadData)の詳細です。本節のサンプルプログラムは、整数型データ(符号無 32bit)の 2 データを連結して整数型データ(符号無 64bit)にしています。

AslReadData

機能

指定スレーブユニットから入力を行います。

書式

Ret = AslReadData(Id, SlaveId, Data)

引数

Id [VB.NET: Short][C, C++: short *][C#: out short]
AslInitで取得したデバイスIDを指定します。

SlaveId [VB.NET: Short][C, C++: short *][C#: out short]
入力対象のスレーブユニットIDを指定します。

Data [VB.NET: UInteger][C, C++: unsigned long *][C#: out uint]
データ入力に使用する2データ分の配列を指定します。

1スレーブユニットに割り当てられる最大データ長は、64bitとなり32bitの変数を2データ使用します。

※API ツールのヘルプ画面では、リモートユニットはスレーブユニットと記載されています

5.5.4.1.I/O データの読取りデータ構造

I/O データの読取りデータのデータ構造について記載します。

下図は、整数型データ(符号無 32bit)の 2 データを連結した状態のデータ構造(ビット状態)です。

<全体のデータ構造> ※データサイズは、ワードデータ単位に分けています。

ビット番号	63	...	48	47	...	32	31	...	16	15	...	0
データサイズ	16 ビット			16 ビット			16 ビット			16 ビット		

<データ構造(I/O データ出力数:2bit)>

ビット番号	63	...	2	1	0
データサイズ	62 ビット			1 ビット	1 ビット
データ説明	不使用			2 ビット出力 2 ビット目	2 ビット出力 1 ビット目

<データ構造(I/O データ出力数:16bit)>

ビット番号	63	...	48	47	...	32	31	...	16	15	...	0
データサイズ	16 ビット			16 ビット			16 ビット			16 ビット		
データ説明	不使用			不使用			不使用			I/O ワードデータ 1		

<データ構造(I/O データ出力数:32bit)>

ビット番号	63	...	48	47	...	32	31	...	16	15	...	0
データサイズ	16 ビット			16 ビット			16 ビット			16 ビット		
データ説明	不使用			不使用			I/O ワードデータ 2			I/O ワードデータ 1		

<データ構造(I/O データ出力数:48bit)>

ビット番号	63	...	48	47	...	32	31	...	16	15	...	0
データサイズ	16 ビット			16 ビット			16 ビット			16 ビット		
データ説明	不使用			I/O ワードデータ 3			I/O ワードデータ 2			I/O ワードデータ 1		

5.5.4.2. 読取りデータ (I/O データ出力数 : 2bit)

機器パラメータ(設定フラグ 2)の I/O データ出力数に“2bit”を設定しているとき、読取りデータから 1 ビット目(ビット番号:0)と 2 ビット目(ビット番号:1)のデータを読み取ります。また各ビットのデータの意味については、機器パラメータ(設定フラグ 2)の I/O データ 2 ビット出力の 1 ビット目出力タイプ、2 ビット目出力タイプの設定状態でデータの内容が異なります。

※機器パラメータ(設定フラグ 2)については、「[5.2.5.2. 機器パラメータ \(設定フラグ 2\)](#)」をご参照ください。

＜読取りデータ(I/O データ)ビット単位＞

ビット	データ説明
0	2 ビット出力 1 ビット目
1	2 ビット出力 2 ビット目
2	不使用
...	
63	

＜2 ビット出力時の出力タイプ＞

出力タイプ	ビットデータ(0)	ビットデータ(1)
動作状態	その他	計測時
ティーチング実施	ティーチング未実施	ティーチング実施中
振動レベルアラート状態	OK	NG
温度レベルアラート状態	OK	NG

5.5.4.3. 読取りデータ (I/O データ出力数 : 16bit)

機器パラメータ(設定フラグ 2)の I/O データ出力数に“16bit”を設定しているとき、読取りデータから I/O データワード 1 を抽出します。抽出したワードデータは整数型データ(符号無 16bit)になります。この抽出データを機器パラメータ(設定フラグ 2)の I/O データ用の振動パラメータに対応するデータへのデータ加工が必要になります。

＜I/O データの振動パラメータと抽出ワードデータ＞

振動パラメータ	抽出ワードデータ(整数型)
加速度	加速度値の 100 倍値
速度	速度値の 100 倍値
変位	変位値の 10 倍値

※機器パラメータ(設定フラグ 2)については、「[5.2.5.2. 機器パラメータ \(設定フラグ 2\)](#)」をご参照ください。

＜計算手順＞

- ① 整数型データ(符号無 64bit:ulong)から整数型データ(符号無 16bit:ushort)を抽出します。
- ② 整数型データ(符号無 16bit:ushort)から整数型データ(符号無 16bit:short)に変換します。
- ③ 整数型データ(符号付 16bit:short)を浮動小数点データ型(double 等)に変換し、振動パラメータの適切な除数で除算して物理値を計算します。

＜サンプルコード(C#)＞

```
1 // リモートユニットの I/O データを取得
2 // ここでは、I/O データ 1～3 の割当は以下を前提とする
3 // I/O データワード 1 : 加速度 m/s^2
4 // I/O データワード 2 : 速度 mm/s
5 // I/O データワード 3 : 変位 μm
6 uint[] readIoDatas = new uint[2];
7 int ret = cas10bj.ReadData(devId, remoteId, readIoDatas);
8
9 // I/O データの読み取りを連結
10 ulong remoteIoData = ((ulong)readIoDatas[1] << 32) | (ulong)readIoDatas[0];
11
12 // I/O データワード 1[15:0]より加速度を算出
13 double accValue = (double)((short)(remoteIoData & 0xffff)) / 100.0;
```

5.5.4.4. 読取りデータ (I/O データ出力数 : 32bit)

機器パラメータ(設定フラグ 2)の I/O データ出力数に“32bit”を設定しているとき、読取りデータから I/O データワード 1 と I/O データワード 2 を抽出します。抽出した各ワードデータは整数型データ(符号無 16bit)になります。

この抽出データを機器パラメータ(設定フラグ 2)の I/O データ用の振動パラメータに対応するデータへのデータ加工が必要になります。

※機器パラメータ(設定フラグ 2)については、「[5.2.5.2. 機器パラメータ \(設定フラグ 2\)](#)」をご参照ください。

<計算手順>

- ④ 整数型データ(符号無 64bit:ulong)から整数型データ(符号無 16bit:ushort)を抽出します。
- ⑤ 整数型データ(符号無 16bit:ushort)から整数型データ(符号無 16bit:short)に変換します。
- ⑥ 整数型データ(符号付 16bit:short)を浮動小数点データ型(double 等)に変換し、振動パラメータの適切な除数で除算して物理値を計算します。

<サンプルコード(C#)>

```
1 // リモートユニットの I/O データを取得
2 // ここでは、I/O データ 1~3 の割当は以下を前提とする
3 // I/O データワード 1 : 加速度 m/s^2
4 // I/O データワード 2 : 速度 mm/s
5 // I/O データワード 3 : 変位 μm
6 uint[] readIoDatas = new uint[2];
7 int ret = cas10bj.ReadData(devId, remoteId, readIoDatas);
8
9 // I/O データの読み取りを連結
10 ulong remoteIoData = ((ulong)readIoDatas[1] << 32) | (ulong)readIoDatas[0];
11
12 // I/O データワード 1[15:0]より加速度を算出
13 double accValue = (double)((short)(remoteIoData & 0xffff)) / 100.0;
14 // I/O データワード 2[31:16]より速度を算出
15 double velValue = (double)((short)((remoteIoData >> 16) & 0xffff)) / 100.0;
```

5.5.4.5. 読取りデータ (I/O データ出力数 : 48bit)

機器パラメータ(設定フラグ 2)の I/O データ出力数に“48bit”を設定しているとき、読取りデータから I/O データワード 1、I/O データワード 2、I/O データワード 3 を抽出します。抽出した各ワードデータは整数型データ(符号無 16bit)になります。

この抽出データを機器パラメータ(設定フラグ 2)の I/O データ用の振動パラメータに対応するデータへのデータ加工が必要になります。

※機器パラメータ(設定フラグ 2)については、「[5.2.5.2. 機器パラメータ \(設定フラグ 2\)](#)」をご参照ください。

＜計算手順＞

- ① 整数型データ(符号無 64bit:ulong)から整数型データ(符号無 16bit:ushort)を抽出します。
- ② 整数型データ(符号無 16bit:ushort)から整数型データ(符号付 16bit:short)に変換します。
- ③ 整数型データ(符号付 16bit:short)を浮動小数点データ型(double 等)に変換し、振動パラメータの適切な除数で除算して物理値を計算します。

＜サンプルコード(C#)＞

```
1 // リモートユニットの I/O データを取得
2 // ここでは、I/O データ 1～3 の割当は以下を前提とする
3 // I/O データワード 1 : 加速度 m/s^2
4 // I/O データワード 2 : 速度 mm/s
5 // I/O データワード 3 : 変位 μm
6 uint[] readIoDatas = new uint[2];
7 int ret = cas10bj.ReadData(devId, remoteId, readIoDatas);
8
9 // I/O データの読み取りを連結
10 ulong remoteIoData = ((ulong)readIoDatas[1] << 32) | (ulong)readIoDatas[0];
11
12 // I/O データワード 1[15:0]より加速度を算出
13 double accValue = (double)((short)(remoteIoData & 0xffff)) / 100.0;
14 // I/O データワード 2[31:16]より速度を算出
15 double velValue = (double)((short)((remoteIoData >> 16) & 0xffff)) / 100.0;
16 // I/O データワード 3[47:32]より変位を算出
17 double dispValue = (double)((short)((remoteIoData >> 32) & 0xffff)) / 10.0;
```

5.5.5. I/O データのデータ読取り処理について

本章のサンプルプログラムは、I/O データのデータ読取りを 1 回実行する処理となっております。ループ処理内やタイマーイベント処理等で I/O データのデータ読取り(API 関数:AslReadData)を繰り返し行うことで、モニタリングをイメージしたプログラムを作成する事も出来ます。

改訂履歴

バージョン	改訂日	改訂内容
第1版	2025年 2月	初版
第2版	2026年 9月	・ 巻末のお問い合わせ窓口（事業所情報）を更新

著 作 者	IMV株式会社		
発 行 者	IMV株式会社		
文 章 番 号	TVE-6-4275		
発 行 年 月 日	2026年 9月	第2版	発行
MES事業本部	〒555-0011 大阪市西淀川区竹島2-6-10		
予 知 保 全 B U	TEL: 050-1745-6779		

