関数リファレンス for CANabh3.dll

|  |  |  |
| --- | --- | --- |
| Date | Auth | Notice |
| 2023/03/03 | Y.OGAWA | 1st release |
|  |  |  |
|  |  |  |
|  |  |  |

# 目次

[目次 1](#_Toc129269455)

[注意点 2](#_Toc129269456)

[環境 3](#_Toc129269457)

[ファイル構成 4](#_Toc129269458)

[基本的な利用方法 6](#_Toc129269459)

[CANabh3dll.cpp/hの利用方法 7](#_Toc129269460)

[列挙子 8](#_Toc129269461)

[構造体 9](#_Toc129269462)

[関数 11](#_Toc129269463)

[InitInstance 11](#_Toc129269464)

[ExitInstance 12](#_Toc129269465)

[GetInterfaceCount 13](#_Toc129269466)

[SetInterface 14](#_Toc129269467)

[GetCurrentInterface 15](#_Toc129269468)

[SetOpenTimeout 16](#_Toc129269469)

[SetSendTimeout 17](#_Toc129269470)

[SetRecvTimeout 18](#_Toc129269471)

[SetHostID 19](#_Toc129269472)

[GetHostID 20](#_Toc129269473)

[SetBaudrate 21](#_Toc129269474)

[GetBaudrate 22](#_Toc129269475)

[GetTm 23](#_Toc129269476)

[GetCounter 24](#_Toc129269477)

[abh3\_can\_init 25](#_Toc129269478)

[abh3\_can\_port\_init 26](#_Toc129269479)

[abh3\_cmd\_init 27](#_Toc129269480)

[abh3\_can\_cmdAY 28](#_Toc129269481)

[abh3\_can\_cmdBX 28](#_Toc129269482)

[abh3\_can\_cmd 29](#_Toc129269483)

[abh3\_can\_inSet 30](#_Toc129269484)

[abh3\_can\_inBitSet 31](#_Toc129269485)

[abh3\_can\_cmdAndopSet 32](#_Toc129269486)

[abh3\_can\_reqBRD 33](#_Toc129269487)

[abh3\_can\_trans 34](#_Toc129269488)

[abh3\_can\_copylastdata 35](#_Toc129269489)

[abh3\_can\_resetlastdata 36](#_Toc129269490)

[abh3\_can\_read 37](#_Toc129269491)

[abh3\_can\_flush 38](#_Toc129269492)

[abh3\_can\_finish 39](#_Toc129269493)

[cnvVel2CAN 40](#_Toc129269494)

[cnvCAN2Vel 41](#_Toc129269495)

[cnvCur2CAN 42](#_Toc129269496)

[cnvCAN2Cur 43](#_Toc129269497)

[cnvCAN2Load 44](#_Toc129269498)

[cnvCAN2Analog 45](#_Toc129269499)

[cnvCAN2Volt 46](#_Toc129269500)

[値の単位 47](#_Toc129269501)

[非対称通信について 48](#_Toc129269502)

[実装例 49](#_Toc129269503)

# 注意点

・本DLLプロジェクトはソースコードを含んだVisual Studio用のプロジェクトとして提供されます。

　利用するVisual Studioは、バージョン2019を想定しています。

・本DLLの利用には、以下の知識がある事が前提となります。

　標準DLLを利用する為の知識

・本DLLからHMS製のUSB-to-CAN V2インターフェースのみが利用可能です。

　HMS社からUSB-to-CAN V2インターフェースのデバイスドライバを取得し、本DLLを利用する

　アプリケーションの開発環境と実行環境へ、それぞれインストールする必要があります

・本DLLは、32bitアプリケーション用のDLLとして設計されていますが、

　実行環境は一般向けWindowsのx64バージョン(WOW64搭載のWindows10,11)を想定しています。

・本DLLは、前身である CANa31.dll に非対称通信を行う為の機能を追加して改造した物になります。

Visual Studio及びWindowsはMicrosoftの商標です。

# 環境

本DLLの作成環境と想定利用環境は以下の通りです

|  |  |  |
| --- | --- | --- |
| 要素 | 作成環境 | 想定利用環境 |
| OS | Windows10 pro 64bit  version 21H2 | Windows10 pro/home 64bit  version 21H2 又はそれ以降 |
| CPU | Intel i9-10900X | Intel系CPU(\*1) |
| コンパイラ | Microsoft Visual Studio 2019 pro  version 16.11.15 | Microsoft Visual Studio 2019 |
| DLL利用先 |  | 32bitアプリケーション  ・MFCアプリケーション(32bit)  ・Win32アプリケーション(32bit) |

(\*1)Intel 第12世代（又はそれ以降）のCPU利用時、動作に問題が有る場合は、

　実行環境のUEFI-BIOS設定で、Eコアを無効化する事で改善する可能性が有ります。

　詳細は、実行環境のPCマニュアル又を参照願います。

# ファイル構成

本DLLは以下のファイルで構成されます

|  |  |
| --- | --- |
| ファイル名 | 内容 |
| ABH3.cpp  ABH3.h | ABH3ドライバ固有機能C++クラス |
| Can1939.h | CAN J1939仕様のデータ作成用C++クラス |
| CANabh3.cpp  CANabh3.h | 本DLLでエクスポートされる関数が格納されたCソースコード  ユーザーが利用する関数は、このコード内の関数が出入口となります |
| CANabh3.def | 本DLLでエクスポートされる関数を定義したファイル  本DLLでは、このファイルをプロジェクト設定で明示的に指定してあります |
| CANabh3.rc | DLLに内包されるリソース定義  バージョン情報等が含まれます |
| CANabh3dll.cpp  CANabh3dll.h | 本DLLをユーザー側でダイナミックロードする場合に、関数を楽に扱う為のコード。  ユーザー側の上位アプリケーションをMFCで作成する場合に利用可能。  本DLL構築時にこのソースは利用されません。 |
| CanIF.cpp  CanIF.h | 本DLLで扱うインターフェースの制御を行うC++クラス  利用可能なCANインターフェースは、本クラスから継承して実装します |
| dllmain.cpp | DLLエントリ  アタッチ／デタッチ時に処理が必要な場合は、本ソースコードを変更して御利用下さい |
| IxxatV2.cpp  IxxatV2.h | HMS社製、USB-to-CAN v2インターフェースの制御クラス |
| typedef.h | 本DLLでユーザーが使用する列挙子・構造体の定義 |
| packfloat.cpp  packfloat.h | 実数系にpackfloat（当社作成の特殊変数）を利用する場合に必要となるコード。 |
| resource.h  pch.cpp  pch.h  targetvar.h  framework.h | Visual Studio利用時に自動作成されるファイル。  一部変更済み。 |
| readme.md | Gitサーバ用表題ファイル（履歴記載有り） |
| CANabh3.sln  CANabh3.vcproj | Visual Studio用のソリューション/プロジェクトファイル。  本プロジェクトは、Visual Studio 2019 で構築しています。 |

インターフェースの利用準備

本DLLでは以下のインターフェースのみに対応しています。

|  |  |
| --- | --- |
| メーカー | HMS |
| メーカーURL | https://www.ixxat.com/ja/ |
| 名称 | USB-to-CAN V2 |
| 準備 | HMS社のサイトから、Windows用のファイル(zip)をダウンロードします  解凍後に出てくる実行ファイルを、開発環境及び実行環境で実行し、  上記名称用のデバイスドライバ（開発環境ならSDKを追加）を選択してインストール。 |
| 本書更新時の  デバイスドライバ | ダウンロードURL  https://www.ixxat.com/ja/technical-support/support/windows-driver-software  バージョン  4.0.1003.0 |

# 基本的な利用方法

本DLLの利用想定アプリケーションと利用方法は、以下となります

|  |  |  |
| --- | --- | --- |
| No. | アプリケーション種類 | 利用方法 |
| 1 | Win32アプリケーション | 本プロジェクトの生成物（DLL/LIB）をユーザー側のアプリケーションでスタティック又はダイナミックリンクで御利用下さい。  但し、Visual Studio 2019 以外のコンパイラを利用される場合は、  本DLLプロジェクトをお客様の環境で再ビルドしてから御利用下さい。 |
| 2 | MFCアプリケーション | 本プロジェクトの生成物（DLL/LIB）をユーザー側のアプリケーションでスタティック又はダイナミックリンクで御利用下さい。  但し、Visual Studio 2019 以外のコンパイラを利用される場合は、  本DLLプロジェクトをお客様の環境で再ビルドしてから御利用下さい。  DLLの動的ロードを行う場合、プロジェクト内のCanABH3dll.cpp/hを利用する事で、本DLLを楽に扱う事が可能です。  （「CANabh3dll.cpp/hの利用方法」の項を参照） |

# CANabh3dll.cpp/hの利用方法

本ファイルはMFCアプリケーションからDLLの関数を簡単に扱う為に用意されています。

ファイルをユーザーアプリケーションのプロジェクトにコピーして利用します。

コード例

|  |
| --- |
| #include “CANabh3dll.h”  static CAN\_FUNCLIST g\_func;  int test()  {  //DLLの読み込み  HANDLE hDLL = LoadLibrary(“CANabh3.dll”);  if(nDLL == NULL)  return(-1); //DLL読み込みエラー  //DLLに含まれる関数の取得(CANabh3dll.cpp内に関数実体)  GetFunctions(hDLL,&g\_func);  //DLL関数の使用例  g\_func.InitInstance(0); //初期化関数の呼び出し    g\_func.<各種関数> を使用して処理を行う  g\_func.ExitInstance(); //開放前の呼び出し  //DLL開放  FreeLibrary(hDLL);  return(0);  } |

# 列挙子

//パケット種類の指定用

typedef enum

{

UNKNOWN\_PACKET = 0, //内部用（ユーザーでは利用しない）

SINGLE\_PACKET = 1, //シングルパケット

BROADCAST\_PACKET = 2, //ブロードキャストパケット

ANY\_PACKET = 0, //パケット種類を問わない

} PACKETTYPE;

# 構造体

//受信に使用する構造体（受信情報＋受信データ8バイト）

typedef struct \_CANABH3\_RESULT

{

//受信データのCAN-ID

uint32\_t nID; //CANID

union

{

//内部用

uint8\_t raw[8];

//受信要素

//シングルパケット(DP0)

struct \_DP0R

{

int16\_t nBackAY; //A/Y帰還

int16\_t nBackBX; //B/X帰還

uint32\_t nCtrlBit; //制御フラグ

} DP0R;

//ブロードキャストパケット(0)

struct \_BR0

{

uint32\_t nErrorBit; //異常フラグ

uint32\_t nWarnBit; //警告フラグ

} BR0;

//ブロードキャストパケット(1)

struct \_BR1

{

uint32\_t nIoBit; //IOフラグ

uint32\_t nCtrlBit; //入力フラグ

} BR1;

//ブロードキャストパケット(2)

struct \_BR2

{

int16\_t nOrderSpeedAY; //A/Y速度指令

int16\_t nOrderSpeedBX; //B/X速度指令

int16\_t nBackSpeedAY; //A/Y速度帰還

int16\_t nBackSpeedBX; //B/X速度帰還

} BR2;

//ブロードキャストパケット(3)

struct \_BR3

{

int16\_t nOrderCurrentAY; //A/Y電流指令

int16\_t nOrderCurrentBX; //B/X電流指令

int16\_t nLoadA; //A負荷率

int16\_t nLoadB; //B負荷率

} BR3;

//ブロードキャストパケット(4)

struct \_BR4

{

int32\_t nCountPulseA; //Aパルス積算値

int32\_t nCountPulseB; //Bパルス積算値

} BR4;

//ブロードキャストパケット(5)

struct \_BR5

{

int16\_t nAnalog0; //アナログ入力0

int16\_t nAnalog1; //アナログ入力1

int16\_t nPowerMain; //主電源電圧

int16\_t nPowerCtrl; //制御電源電圧

} BR5;

//ブロードキャストパケット(6)

struct \_BR6

{

float nMonitor0; //モニタ0データ

float nMonitor1; //モニタ1データ

} BR6;

} u;

} CANABH3\_RESULT,\*pCANABH3\_RESULT;

//最終受信データ用構造体

typedef struct \_CANABH3\_LASTRECV

{

//シングルパケット(DP0)

struct \_DP0R

{

int16\_t nBackAY; //A/Y帰還

int16\_t nBackBX; //B/X帰還

uint32\_t nCtrlBit; //制御フラグ

} DP0R;

//ブロードキャストパケット(0)

struct \_BR0

{

uint32\_t nErrorBit; //異常フラグ

uint32\_t nWarnBit; //警告フラグ

} BR0;

//ブロードキャストパケット(1)

struct \_BR1

{

uint32\_t nIoBit; //IOフラグ

uint32\_t nCtrlBit; //入力フラグ

} BR1;

//ブロードキャストパケット(2)

struct \_BR2

{

int16\_t nOrderSpeedAY; //A/Y速度指令

int16\_t nOrderSpeedBX; //B/X速度指令

int16\_t nBackSpeedAY; //A/Y速度帰還

int16\_t nBackSpeedBX; //B/X速度帰還

} BR2;

//ブロードキャストパケット(3)

struct \_BR3

{

int16\_t nOrderCurrentAY; //A/Y電流指令

int16\_t nOrderCurrentBX; //B/X電流指令

int16\_t nLoadA; //A負荷率

int16\_t nLoadB; //B負荷率

} BR3;

//ブロードキャストパケット(4)

struct \_BR4

{

int32\_t nCountPulseA; //Aパルス積算値

int32\_t nCountPulseB; //Bパルス積算値

} BR4;

//ブロードキャストパケット(5)

struct \_BR5

{

int16\_t nAnalog0; //アナログ入力0

int16\_t nAnalog1; //アナログ入力1

int16\_t nPowerMain; //主電源電圧

int16\_t nPowerCtrl; //制御電源電圧

} BR5;

//ブロードキャストパケット(6)

struct \_BR6

{

float nMonitor0; //モニタ0データ

float nMonitor1; //モニタ1データ

} BR6;

//各格納場所の更新フラグ

struct \_UPDATE

{

uint8\_t nUpdate; //0以外で値が更新されている（受信している）

} update[8]; //0..DP0R 1..BR0 2..BR1 .... 7..BR6

} CANABH3\_LASTRECV,\*pCANABH3\_LASTRECV;

//PACK\_FLOAT用構造体

typedef struct \_PACK\_FLOAT

{

union

{

struct \_PACK\_FLOAT\_INFO

{

//LSB側から記載(処理系に注意)

int32\_t kasuu:21; // 仮数部(-999999～0～99999)

int32\_t shousuutenn:4; // 少数点(-8～0～7)

int32\_t shisuu:7; // 指数部(-64～0～63)

} info;

int32\_t nDirectData; // 直接アクセス用

} u;

} \*pPACK\_FLOAT,PACK\_FLOAT;

# 関数

## InitInstance

|  |  |
| --- | --- |
| 概要 | インターフェースの利用開始 |
| 詳細 | インターフェースの利用を開始します。 |
| 構文 | CANABH3API void InitInstance(int32\_t nIFnum) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nIFnum | 使用するインターフェースを指定して下さい   |  |  | | --- | --- | | 値 | インターフェース | | 0 | Ixxat USB-to-CAN V2 | | |
| 戻り値 | 無し |
| 注意点等 | この時点では、CAN回線にはまだ接続されていません。  本DLLでサポートされるインターフェースは、Ixxat USB-to-CAN V2のみとなります。 |

## ExitInstance

|  |  |
| --- | --- |
| 概要 | インターフェースの利用終了 |
| 詳細 | インターフェースの利用を終了し、DLLを開放可能な状態にします。 |
| 構文 | CANABH3API void ExitInstance() |
| パラメータ | 無し |
| 戻り値 | 無し |
| 注意点等 | CAN回線を切断してなかった場合、自動的に切断されます  既にインターフェースの利用を終了している場合は、何も処理しません。 |

## GetInterfaceCount

|  |  |
| --- | --- |
| 概要 | 使用可能なCANインターフェース数を取得 |
| 詳細 | InitInstanceで指定した「使用したいインターフェース」に対して、  現時点で利用可能な本数(PCに接続されているデバイス数)を取得します。 |
| 構文 | CANABH3API int32\_t GetInterfaceCount() |
| パラメータ | 無し |
| 戻り値 | PCに接続されている「本DLLで利用可能なCANインターフェース数」が戻ります。 |
| 注意点等 |  |

## SetInterface

|  |  |
| --- | --- |
| 概要 | CAN回線に接続するインターフェースを指定 |
| 詳細 | CAN回線に接続するインターフェースを指定。 |
| 構文 | CANABH3API int32\_t SetInterface(int32\_t nDeviceNum) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nDeviceNum | 開く対象のインターフェース番号として、  何本目のインターフェース（0開始）を指定します | |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | 上記以外 | 異常終了 | |
| 注意点等 | 複数本接続した場合、接続した順に番号が割り振られます。  但し、PCにインターフェースを接続した状態でPCを起動した場合、又は再起動した場合等、  常に同じ接続順になるとは限らない事に注意が必要です。  基本的には1本のみ利用する事を推奨します。 |

## GetCurrentInterface

|  |  |
| --- | --- |
| 概要 | CAN回線に接続したインターフェース番号を取得 |
| 詳細 | CAN回線に接続した時点で指定されていたインターフェース番号を取得。 |
| 構文 | CANABH3API int32\_t GetCurrentInterface() |
| パラメータ | 無し |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0未満 | CAN回線に接続していない | | 上記以外 | CAN回線に接続した時点で、SetInterface関数に  指定していた値 | |
| 注意点等 |  |

## SetOpenTimeout

|  |  |
| --- | --- |
| 概要 | CAN回線の接続タイムアウト時間を設定 |
| 詳細 | CAN回線に接続する時、接続要求から接続完了迄の許容時間を指定します。 |
| 構文 | CANABH3API void SetOpenTimeout(uint32\_t nTimeoutMS) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTimeoutMS | CAN回線へ接続する処理の許容時間[ms]  推奨値は1000 | |
| 戻り値 | 無し |
| 注意点等 | CAN回線に接続する前に必ず設定する必要が有ります。 |

## SetSendTimeout

|  |  |
| --- | --- |
| 概要 | CAN回線の送信タイムアウト時間を設定 |
| 詳細 | CAN回線へデータ送信する時、送信要求から送信完了迄の許容時間を指定します。 |
| 構文 | CANABH3API void SetSendTimeout(uint32\_t nTimeoutMS) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTimeoutMS | CAN回線へのデータ送信許容時間[ms]  推奨値は1000 | |
| 戻り値 | 無し |
| 注意点等 | CAN回線に接続する前に必ず設定する必要が有ります。 |

## SetRecvTimeout

|  |  |
| --- | --- |
| 概要 | CAN回線の受信タイムアウト時間を設定 |
| 詳細 | CAN回線からデータ受信する時、受信要求から受信完了迄の許容時間を指定します。 |
| 構文 | CANABH3API void SetRecvTimeout(uint32\_t nTimeoutMS) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTimeoutMS | CAN回線からの受信処理許容時間[ms]  推奨値は1000 | |
| 戻り値 | 無し |
| 注意点等 | CAN回線に接続する前に必ず設定する必要が有ります。 |

## SetHostID

|  |  |
| --- | --- |
| 概要 | CAN回線上の通信ホストアドレスを設定 |
| 詳細 | PCが使用する通信アドレスを指定します。 |
| 構文 | CANABH3API void SetHostID(uint8\_t nAdrs) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nAdrs | 通信ホスト(PC)のアドレスを指定します  通信ホストから送信する場合は、他の機器と重複する値を指定しないで下さい、又00hとFFhは利用不可です。 | |
| 戻り値 | 無し |
| 注意点等 | CAN回線に接続する前に必ず設定する必要が有ります。  他の機器の通信を傍受する為だけに利用する場合は、傍受対象の機器IDを指定して下さい。 |

## GetHostID

|  |  |
| --- | --- |
| 概要 | CAN回線上の通信ホストアドレスを取得 |
| 詳細 | PCが使用する通信アドレスを取得します。 |
| 構文 | CANABH3API uint8\_t GetHostID() |
| パラメータ | 無し |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 01h - FEh | SetHostID関数で指定された通信ホストアドレスが  戻ります | |
| 注意点等 |  |

## SetBaudrate

|  |  |
| --- | --- |
| 概要 | CAN回線で使用する通信速度を指定 |
| 詳細 | CAN回線で使用する通信速度を指定します。 |
| 構文 | CANABH3API void SetBaudrate(uint32\_t nBaudrateKbps) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nBaudrateKbps | 通信速度[Kbps]を以下から指定します   |  |  | | --- | --- | | 値 | 通信速度 | | 10 | 10[Kbps] | | 20 | 20[Kbps] | | 50 | 50[Kbps] | | 100 | 100[Kbps] | | 125 | 125[Kbps] | | 250 | 250[Kbps] | | 500 | 500[Kbps] | | 800 | 800[Kbps] | | 1000 | 1000[Kbps] | | |
| 戻り値 | 無し |
| 注意点等 | CAN回線に接続する前に必ず設定する必要が有ります。  パラメータで指定可能な値以外は、通信速度に設定出来ません。  ある程度高速な通信速度を指定した場合、ケーブル及び終端抵抗の品質によっては  正しく通信出来ない場合が有ります。  CAN回線接続後に通信速度を変更する場合は、一度CAN回線を切断(abh3\_can\_finish)  してから、本関数で新しい速度を設定し、再度CAN回線に接続(abh3\_can\_port\_open)して  下さい。 |

## GetBaudrate

|  |  |
| --- | --- |
| 概要 | 設定した通信速度を取得 |
| 詳細 | 設定済みの通信速度を取得します。 |
| 構文 | CANABH3API uint32\_t GetBaudrate() |
| パラメータ | 無し |
| 戻り値 | SetBaudrateで指定された通信速度[Kbps]が戻ります。 |
| 注意点等 | CAN回線接続中にSetBaudrateで通信速度を設定しても、実際の通信速度は変わりませんが、  本関数の戻り値は新しく設定した値が戻るので注意が必要です。 |

## GetTm

|  |  |
| --- | --- |
| 概要 | 時間を[ms]単位で取得 |
| 詳細 | PCが起動した時間を0として、現在迄の時間を[ms]単位で取得します。 |
| 構文 | CANABH3API uint32\_t GetTm() |
| パラメータ | 無し |
| 戻り値 | PCが起動してから現在迄の時間が[ms]単位で戻ります。 |
| 注意点等 | 32bitが最大の為、49.7日程度でオーバーフローして0に戻ります。  本関数は、連続稼働前提のシステムで利用する事は、非推奨です。  非力なPCでは、時間取得自体に処理時間が掛かる場合が有り、本関数を利用する事で、  アプリケーションのパフォーマンスに影響する場合が有ります。 |

## GetCounter

|  |  |
| --- | --- |
| 概要 | 送受信データ量をビット単位で取得 |
| 詳細 | 最後に本関数を利用してから、次に利用する迄に送受信を行ったCAN通信で、  CAN回線上で送受信したデータ量（ヘッダ含む）をビット単位で取得します。 |
| 構文 | CANABH3API uint32\_t GetCounter() |
| パラメータ | 無し |
| 戻り値 | 送受信量がビット単位で戻ります |
| 注意点等 | 本関数は、上位アプリケーションからCAN回線の使用帯域率を求める為に用意されましたが、  厳密な値では無い事に注意が必要です。（誤差が有ります） |

## abh3\_can\_init

|  |  |
| --- | --- |
| 概要 | 指令の初期化及びCAN回線へ接続 |
| 詳細 | 以下の順で処理が行われます。  (1)以下の要素を一括設定   |  |  | | --- | --- | | 要素名 | 設定値 | | A/Y指令 | 0 | | B/X指令 | 0 | | 入力(bit対応) | 0 |   (2)SetInterfaceで指定されたインターフェースを利用して、CAN回線を接続  (3)シングルパケットとして、上記要素を格納したパケットを送信  (4)(返信の格納領域が指定有りなら)返答が有る物として受信 |
| 構文 | CANABH3API int32\_t abh3\_can\_init(uint8\_t nTargetID,pCANABH3\_RESULT pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTargetID | 通信先のIDを指定 | | pPtr | 送信に対する返信の格納領域  NULL指定時は送信のみ行います  受信時は、pPtr->u.DP0Rに格納されます | |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | 0以外 | 異常終了時のエラーコード | |
| 注意点等 |  |

## abh3\_can\_port\_init

|  |  |
| --- | --- |
| 概要 | CAN回線に接続 |
| 詳細 | CAN回線に接続。 |
| 構文 | CANABH3API int32\_t abh3\_can\_port\_init(void) |
| パラメータ | 無し |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | 0以外 | 異常終了時のエラーコード | |
| 注意点等 | SetInterfaceで指定されたインターフェースを利用して、CAN回線に接続します。  CAN回線に接続するのみで、何も送信しません。  指令等も初期化されない事に注意が必要です。 |

## abh3\_cmd\_init

|  |  |
| --- | --- |
| 概要 | 指令の初期化と送信 |
| 詳細 | 以下の順で処理が行われます。  (1)以下の要素を一括設定   |  |  | | --- | --- | | 要素名 | 設定値 | | A/Y指令 | 0 | | B/X指令 | 0 | | 入力(bit対応) | 0 |   (2)シングルパケットとして、上記要素を格納したパケットを送信  (3)(返信の格納領域が指定有りなら)返答が有る物として受信 |
| 構文 | CANABH3API int32\_t abh3\_cmd\_init(uint8\_t nTargetID,pCANABH3\_RESULT pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTargetID | 通信先のIDを指定 | | pPtr | 送信に対する返信の格納領域  NULL指定時は送信のみ行います  受信時は、pPtr->u.DP0Rに格納されます | |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | 0以外 | 異常終了時のエラーコード | |
| 注意点等 | 本関数は、既にCAN回線へ接続している状態での利用が前提となります。 |

## abh3\_can\_cmdAY

## abh3\_can\_cmdBX

|  |  |
| --- | --- |
| 概要 | 指令の送信（軸別） |
| 詳細 | 指定を送信します。 |
| 構文 | CANABH3API int32\_t abh3\_can\_cmdAY(uint8\_t nTargetID,int16\_t cmd,pCANABH3\_RESULT pPtr)  CANABH3API int32\_t abh3\_can\_cmdBX(uint8\_t nTargetID,int16\_t cmd,pCANABH3\_RESULT pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTargetID | 送信先ID | | cmd | A/Y又はB/X指令値 | | pPtr | 送信に対する返信の格納領域  NULLを指定すると送信のみで戻ります  受信時は、pPtr->u.DP0Rに格納されます | |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | 0以外 | 異常終了時のエラーコード | |
| 注意点等 | A/Y指令送信では過去のB/X指令値が一緒に送信されます。  B/X指令送信では過去のA/Y指令値が一緒に送信されます。 |

## abh3\_can\_cmd

|  |  |
| --- | --- |
| 概要 | 指令の送信（同時） |
| 詳細 | A/Y指令値とB/X指令値を同時に送信します。 |
| 構文 | CANABH3API int32\_t abh3\_can\_cmd(uint8\_t nTargetID,int16\_t cmdAY,int16\_t cmdBX,  pCANABH3\_RESULT pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTargetID | 送信先ID | | cmdAY | A/Y指令値 | | cmdBX | B/X指令値 | | pPtr | 送信に対する返信の格納領域  NULLを指定すると送信のみで戻ります  受信時は、pPtr->u.DP0Rに格納されます | |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | 0以外 | 異常終了時のエラーコード | |
| 注意点等 |  |

## abh3\_can\_inSet

|  |  |
| --- | --- |
| 概要 | 入力の送信（一括） |
| 詳細 | 入力(bit対応)の値をデータ値とマスク値で構築し、送信します。 |
| 構文 | CANABH3API int32\_t abh3\_can\_opSet(uint8\_t nTargetID,int32\_t data,int32\_t mask,  pCANABH3\_RESULT pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTargetID | 送信先ID | | data | データ値（各bitの割り当ては、機器側資料を参照） | | mask | マスク値 | | pPtr | 送信に対する返信の格納領域  NULLを指定すると送信のみで戻ります  受信時は、pPtr->u.DP0Rに格納されます | |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | 0以外 | 異常終了時のエラーコード | |
| 注意点等 | 実際に送信される値は、以下の様に算出されます。  入力(bit対応) = (入力(bit対応) & ~mask) | (data & mask)  dataの値をそのまま送りたい場合は、maskに0を指定して下さい。 |

## abh3\_can\_inBitSet

|  |  |
| --- | --- |
| 概要 | 入力の送信（ビット） |
| 詳細 | 現在の入力(bit対応)の特定ビットを操作し、送信します。 |
| 構文 | CANABH3API int32\_t abh3\_can\_inBitSet(uint8\_t nTargetID,int8\_t num,int8\_t data,  pCANABH3\_RESULT pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTargetID | 送信先ID | | num | ビット番号(0～31) | | data | 設定データ(0～1) | | pPtr | 送信に対する返信の格納領域  NULLを指定すると送信のみで戻ります  受信時は、pPtr->u.DP0Rに格納されます | |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | 0以外 | 異常終了時のエラーコード | |
| 注意点等 | 実際に送信される値は、以下の様に算出されます。  入力(bit対応) = 入力(bit対応) & ~(1 << num) | (data << num) |

## abh3\_can\_cmdAndopSet

|  |  |
| --- | --- |
| 概要 | 指令と入力の送信（一括） |
| 詳細 | 指令(A/Y,B/X)と、入力(bit対応)の値をデータ値とマスク値で構築し、送信します。 |
| 構文 | CANABH3API int32\_t abh3\_can\_cmdAndopSet(uint8\_t nTargetID,  int16\_t cmdAY,int16\_t cmdBX,  int32\_t data,int32\_t mask,  pCANABH3\_RESULT pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTargetID | 送信先ID | | cmdAY | A/Y指令値 | | cmdBX | B/X指令値 | | data | データ値 | | mask | マスク値 | | pPtr | 送信に対する返信の格納領域  NULLを指定すると送信のみで戻ります  受信時は、pPtr->u.DP0Rに格納されます | |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | 0以外 | 異常終了時のエラーコード | |
| 注意点等 | 本関数はabh3\_can\_cmd関数とabh3\_can\_opSet関数の処理を一度に行う為の関数です。  指令はabh3\_can\_cmd関数、入力はabh3\_can\_opSet関数に準拠する為、説明は各関数の  項目を参照願います。 |

## abh3\_can\_reqBRD

|  |  |
| --- | --- |
| 概要 | ブロードキャストパケットのリクエスト |
| 詳細 | 指定番号のブロードキャストパケットを送信し、指定番号に対する要素を取得します。 |
| 構文 | CANABH3API int32\_t abh3\_can\_reqBRD(uint8\_t nTargetID,uint8\_t num,  pCANABH3\_RESULT pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTargetID | 送信先ID | | num | 番号(0x00～0xff)  下位3bitには機器アドレス  上位5bitにはグループ番号が入ります | | pPtr | 送信に対する返信の格納領域  NULLを指定すると送信のみで戻ります  受信時の格納先は、本関数の注意点等を参照。 | |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | 0以外 | 異常終了時のエラーコード | |
| 注意点等 | パラメータのnumに対する通信結果を受け取る領域は以下の通り。  内容に関しては、構造体の説明を参照の事。   |  |  | | --- | --- | | 機器アドレス  (numの下位3bit) | 格納先 | | 0 | pPtr->u.BR0 | | 1 | pPtr->u.BR1 | | 2 | pPtr->u.BR2 | | 3 | pPtr->u.BR3 | | 4 | pPtr->u.BR4 | | 5 | pPtr->u.BR5 | | 6 | pPtr->u.BR6 | |

## abh3\_can\_trans

|  |  |
| --- | --- |
| 概要 | 非公開 |
| 詳細 | 非公開 |
| 構文 | CANABH3API int32\_t abh3\_can\_trans(uint8\_t nTargetID,char\* sbuf,char\* rbuf,  uint32\_t\* rbuflen) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTargetID | 送信先ID | | sbuf | 送信データバッファ | | rbuf | 受信データバッファ | | rbuflen | 受信データバッファサイズ[bytes] | |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | 0以外 | 異常終了時のエラーコード | |
| 注意点等 | 本関数は特殊なコマンドをマルチパケットで送る為の関数です。  パラメータで指定される受信データバッファは、十分な容量が必要です。 |

## abh3\_can\_copylastdata

|  |  |
| --- | --- |
| 概要 | 最終受信データをコピー |
| 詳細 | CAN回線から受信したデータをID毎に格納した領域をコピーします。 |
| 構文 | CANABH3API int32\_t abh3\_can\_copylastdata(uint8\_t nTargetID,pCANABH3\_LASTRECV pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTargetID | コピー対象のデータ発信元ID | | pPtr | 最終受信データを格納する領域 | |
| 戻り値 | 無し |
| 注意点等 | abh3\_can\_flush関数を呼び出した時点のデータが格納されます。  使用する構造体は、シングルパケット・ブロードキャストパケットが別々の領域として  用意されています。  詳細は、「非対称通信について」の項を参照願います。 |

## abh3\_can\_resetlastdata

|  |  |
| --- | --- |
| 概要 | 最終受信データの指定更新フラグを解除 |
| 詳細 | 最終受信データ指定箇所の更新フラグを解除します。 |
| 構文 | CANABH3API int32\_t abh3\_can\_resetlastdata(uint8\_t nTargetID,int32\_t nAdrs) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTargetID | 対象の発信元ID | | nAdrs | 最終受信データの更新フラグを解除するアドレス   |  |  | | --- | --- | | 値 | 解除場所 | | 0 | DP0R | | 1 | BR0 | | 2 | BR1 | | 3 | BR2 | | 4 | BR3 | | 5 | BR4 | | 6 | BR5 | | 7 | BR6 | | |
| 戻り値 | 無し |
| 注意点等 | 利用法や詳細は、「非対称通信について」の項を参照願います。 |

## abh3\_can\_read

|  |  |
| --- | --- |
| 概要 | 指定種類のパケットを受信する |
| 詳細 | 指定相手先が送信した指定種類のパケットを受信する。 |
| 構文 | CANABH3API int32\_t abh3\_can\_read(uint8\_t nTargetID,pCANABH3\_RESULT pPtr,  PACKETTYPE nType,uint8\_t nAdrs) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTargetID | 発信元ID | | pPtr | 送信に対する返信の格納領域  NULL指定不可 | | nType | 受信するパケットタイプを指定する   |  |  | | --- | --- | | 値 | パケットタイプ | | SINGLE\_PACKET | シングルパケット | | BROADCAST\_PACKET | ブロードキャスト  パケット | | ANY\_PACKET | パケット種類問わない | | | nAdrs | nTypeがブロードキャストパケット指定時に受信対象の  機器アドレスを指定する。  他のパケット種類では本パラメータは無視される。   |  |  | | --- | --- | | 値 | 機器アドレス | | 0 | 機器アドレス0 | | 1 | 機器アドレス1 | | 2 | 機器アドレス2 | | 3 | 機器アドレス3 | | 4 | 機器アドレス4 | | 5 | 機器アドレス5 | | 6 | 機器アドレス6 | | |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | 0以外 | 異常終了時のエラーコード | |
| 注意点等 | 本関数を呼び出すと、  ・指定発信元から指定パケットに合致する物を受信  ・受信タイムアウト  どちらかの状態になるまで処理が呼び出し元へ戻らないので注意。  条件に合う/合わない関係なく、パケットを受信した場合は、最終受信データに反映される。 |

## abh3\_can\_flush

|  |  |
| --- | --- |
| 概要 | 受信バッファを空にします |
| 詳細 | 受信バッファにある内容を全て受信し、最終受信データに反映させます。  受信により上書きされた最終受信データ領域は、更新フラグが成立します。 |
| 構文 | CANABH3API int32\_t abh3\_can\_flush() |
| パラメータ | 無し |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | 0以外 | 異常終了時のエラーコード | |
| 注意点等 | 利用法や詳細は、「非対称通信について」の項を参照願います。 |

## abh3\_can\_finish

|  |  |
| --- | --- |
| 概要 | CAN回線の切断 |
| 詳細 | CAN回線から切断します。 |
| 構文 | CANABH3API int32\_t abh3\_can\_finish() |
| パラメータ | 無し |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | 0以外 | 異常終了時のエラーコード | |
| 注意点等 |  |

## cnvVel2CAN

|  |  |
| --- | --- |
| 概要 | 速度を「ABH3の速度」に変換 |
| 詳細 | ユーザーが扱う[min-1]の速度値を、「ABH3で扱う速度」に変換。 |
| 構文 | CANABH3API int16\_t cnvVel2CAN(float vel) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | vel | 変換元の速度[min-1] | |
| 戻り値 | 変換された速度が戻ります。 |
| 注意点等 | ユーザー側で扱う値とABH3で扱う値の関係は、「値の単位」の項を参照の事。 |

## cnvCAN2Vel

|  |  |
| --- | --- |
| 概要 | 「ABH3の速度」を変換 |
| 詳細 | 「ABH3で扱う速度」の値を、ユーザーが扱う速度[min-1]に変換。 |
| 構文 | CANABH3API float cnvCAN2Vel(int16\_t vel) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | vel | 変換元の値 | |
| 戻り値 | 変換された速度[min-1]が戻ります。 |
| 注意点等 | ユーザー側で扱う値とABH3で扱う値の関係は、「値の単位」の項を参照の事。 |

## cnvCur2CAN

|  |  |
| --- | --- |
| 概要 | 電流値を「ABH3の電流値」に変換 |
| 詳細 | ユーザーが扱う電流値[%]を、「ABH3で扱う電流値」に変換。 |
| 構文 | CANABH3API float cnvCur2CAN(float cur) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | cur | 変換元の値 | |
| 戻り値 | 変換された電流値[%]が戻ります。 |
| 注意点等 | ユーザー側で扱う値とABH3で扱う値の関係は、「値の単位」の項を参照の事。 |

## cnvCAN2Cur

|  |  |
| --- | --- |
| 概要 | 「ABH3の電流値」を変換 |
| 詳細 | 「ABH3で扱う電流値」の値を、ユーザーが扱う電流値[%]に変換。 |
| 構文 | CANABH3API float cnvCAN2Cur(int16\_t cur) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | cur | 変換元の値 | |
| 戻り値 | 変換された電流値が戻ります。 |
| 注意点等 | ユーザー側で扱う値とABH3で扱う値の関係は、「値の単位」の項を参照の事。 |

## cnvCAN2Load

|  |  |
| --- | --- |
| 概要 | 「ABH3の負荷率」を変換 |
| 詳細 | 「ABH3で扱う負荷率」の値を、ユーザーが扱う負荷率[%]に変換。 |
| 構文 | CANABH3API float cnvCAN2Load(int16\_t load) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | load | 変換元の値 | |
| 戻り値 | 変換された負荷率[%]が戻ります。 |
| 注意点等 | ユーザー側で扱う値とABH3で扱う値の関係は、「値の単位」の項を参照の事。 |

## cnvCAN2Analog

|  |  |
| --- | --- |
| 概要 | 「ABH3のアナログ入力」を変換 |
| 詳細 | 「ABH3で扱うアナログ入力」の値を、ユーザーが扱うアナログ入力値[V]に変換。 |
| 構文 | CANABH3API float cnvCAN2Analog(int16\_t analog) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | analog | 変換元の値 | |
| 戻り値 | 変換されたアナログ入力値[V]が戻ります。 |
| 注意点等 | ユーザー側で扱う値とABH3で扱う値の関係は、「値の単位」の項を参照の事。 |

## cnvCAN2Volt

|  |  |
| --- | --- |
| 概要 | 「ABH3の電源電圧値」を変換 |
| 詳細 | 「ABH3で扱う電源電圧値」の値を、ユーザーが扱う電源電圧値[V]に変換。 |
| 構文 | CANABH3API float cnvCAN2Volt(int16\_t volt) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | volt | 変換元の値 | |
| 戻り値 | 変換された電源電圧値[V]が戻ります。 |
| 注意点等 | ユーザー側で扱う値とABH3で扱う値の関係は、「値の単位」の項を参照の事。 |

# 値の単位

ユーザー側で使用する単位とABH3側に指定する値の関係は以下の通り

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 要素 | ユーザー側単位 | ABH3分解能 | 変換関数 | |
| ユーザー -> ABH3 | ABH3 -> ユーザー |
| 速度（指令・帰還） | [min] | 0.2[min] | cnvVel2CAN | cnvCAN2Vel |
| 電流（指令・帰還） | [%] | 0.01[%] | cnvCur2CAN | cnvCAN2Cur |
| パルス積算値 | [Pulse] | 1[Pulse] | - | - |
| 負荷率 | [%] | 1[%] | - | cnvCAN2Load |
| 主電源／制御電源電圧 | [V] | 0.1[V] | - | cnvCAN2Volt |
| アナログ入力 | [V] | 0.01[V] | - | cnvCAN2Analog |
| モニタデータ | 単位無し | 単位無し | - | - |

# 非対称通信について

本DLLが想定している接続先との通信は、以下の様な方法があり、条件により非対称(\*1)の場合が有ります。

|  |  |  |
| --- | --- | --- |
| No. | 挙動 | 通信方式 |
| 1 | PC側から送信すると、必ず送信先から返信が届く。 | 通常 |
| 2 | PC側から送信しても、送信先から返信は来ない。 | 非対称1 |
| 3 | PC側から送信しなくても、返信に相当する物が届く。  PC側から送信した場合の返答は、有り無しどちらにも対応が必要な場合。 | 非対称2 |

通信方式により本DLLを以下の様に使用する事が推奨されます。

|  |  |
| --- | --- |
| 通信方式 | 通信方法と利用条件等 |
| 通常 | 通信用関数(abh3\_can\_\*)では、受信領域を指定して利用する。  送信と受信をセットで扱う事になる。 |
| 非対称1 | 通信用関数(abh3\_can\_\*)では、送信処理のみとする為、受信領域にNULL指定。  送信だけ行う事になる。 |
| 非対称2 | 通信用関数(abh3\_can\_\*)では、送信処理のみとする為、受信領域にNULL指定。  受信は勝手に送られてくる分も含めて処理する為、一定間隔でabh3\_can\_flush関数を  呼び出して最終受信データとして内部に取り込む必要が有る。  それと同じタイミング、又は別のタイミングで、受信した内容の処理を行う為に  (1)abh3\_can\_copylastdata関数で最終受信データを取得  (2)取得した構造体のupdateメンバが成立(0以外の値)している所を確認  (3)該当箇所を指定して、abh3\_can\_resetylastdata関数を呼び出す  (4)該当箇所のデータを元に、必要な処理が有れば処理する。  (5)これを全てのupdateメンバに対して行う（(2)に戻る）  この様な処理となる。 |

(\*1)PC(ホスト)が送信しても、機器側が無返信、又は機器側が無条件でPC(ホスト)に何か送ってくる場合。

　主に送信と受信が1:1になっていない関係を指す。

## 実装例

以下前提条件に対する実装の例

前提条件

|  |  |
| --- | --- |
| No. | 条件 |
| (1) | PC側は一定周期毎に指令を送信する必要が有る |
| (2) | PC側はブロードキャストで得られる値（各種ステータス類）を一定周期で表示する必要が有る |
| (3) | ABH3側がブロードキャスト(0～6)で全ての項目を自動配信する設定にしてある |
| (4) | ABH3側はシングルパケットに対して必ず返答する |

上記条件に合う実装の例

|  |  |
| --- | --- |
| No. | 元条件と実装された処理 |
| 1 | 元条件  　(1)PC側は一定周期毎に指令を送信する必要が有る  　(4)ABH3側はシングルパケットに対して必ず返答する  解説  　指令を送るシングルパケットで必ず返答がある条件の為、送信->受信の処理をセットで行う。  実装例  　一定周期毎に指令と入力の一括送信(abh3\_can\_cmdAndopSet関数)を受信領域指定有りで使用 |
| 2 | 元条件  　(2)PC側はブロードキャストで得られる値（各種ステータス類）を一定周期で表示する必要が有る  　(3)ABH3側がブロードキャスト(0～6)で全ての項目を自動配信する設定にしてある  解説  　PC側から要求しなくても勝手に送ってくるので、PC側は受信したデータを格納する処理を  　表示周期で行い、格納結果を表示に利用するだけで良い処理となる。  実装例  　表示周期毎に以下処理を実行する  　(1)abh3\_can\_flush関数を呼び出して受信バッファ内の物を処理させる  　(2)abh3\_can\_copylastdata関数で最終受信データを取得する  　(3)取得した構造体に含まれる値を表示する（要変換） |