関数リファレンス for CANa31.dll

|  |  |  |
| --- | --- | --- |
| Date | Auth | Notice |
| 2021/01/13 | Y.OGAWA | 1st release |
| 2021/03/01 | Y.OGAWA | Update |

# 目次

[目次 1](#_Toc65487655)

[注意点 2](#_Toc65487656)

[環境 3](#_Toc65487657)

[ファイル構成 3](#_Toc65487658)

[基本的な利用方法 5](#_Toc65487659)

[CANa31dll.cpp/hの利用方法 6](#_Toc65487660)

[構造体 8](#_Toc65487661)

[関数 10](#_Toc65487662)

[InitInstance 10](#_Toc65487663)

[ExitInstance 10](#_Toc65487664)

[GetInterfaceCount 11](#_Toc65487665)

[OpenInterface 11](#_Toc65487666)

[CloseInterface 11](#_Toc65487667)

[GetCurrentInterface 12](#_Toc65487668)

[IsOpenInterface 12](#_Toc65487669)

[SetOpenTimeout 13](#_Toc65487670)

[SetSendTimeout 13](#_Toc65487671)

[SetRecvTimeout 13](#_Toc65487672)

[SetTargetID 14](#_Toc65487673)

[GetTargetID 14](#_Toc65487674)

[SetHostID 14](#_Toc65487675)

[GetHostID 14](#_Toc65487676)

[SetBaudrate 15](#_Toc65487677)

[GetBaudrate 15](#_Toc65487678)

[GetTm 15](#_Toc65487679)

[abh3\_can\_init 16](#_Toc65487680)

[abh3\_can\_cmdAY 17](#_Toc65487681)

[abh3\_can\_cmdBX 17](#_Toc65487682)

[abh3\_can\_cmd 17](#_Toc65487683)

[abh3\_can\_inSet 18](#_Toc65487684)

[abh3\_can\_inBitSet 18](#_Toc65487685)

[abh3\_can\_reqPulse 19](#_Toc65487686)

[abh3\_can\_reqBRD 19](#_Toc65487687)

[abh3\_can\_trans 20](#_Toc65487688)

[cnvVel2CAN 21](#_Toc65487689)

[cnvCAN2Vel 21](#_Toc65487690)

[cnvCAN2Trq 21](#_Toc65487691)

[cnvCAN2Load 21](#_Toc65487692)

[cnvCAN2Analog 22](#_Toc65487693)

[cnvCAN2Volt 22](#_Toc65487694)

[値の単位 23](#_Toc65487695)

# 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2015又はそれ以降を想定しています

・本DLLの利用には、以下の知識がある事が前提となります

　DLLをWin32プログラムから利用する為の知識

・本DLLからHMS製の特定CANインターフェースが利用可能ですが、他社のCANインターフェースは

　利用不可です。又、HMS社のCANインターフェースによっては、動作環境でドライバのインストールが必要と

　なる場合があります

・高速にCAN通信を行いたい場合は、HMS社のUSB-to-CAN V2を御利用下さい。

　同社のsimplyCANはスレッドセーフなドライバでは無い為、動作速度が落ちます。

・本DLLが想定しているのは、32bitアプリケーション（Win32/MFC）のみです。

　標準DLLの形式な為、他のプロジェクトでも利用可能と思われますが未保障です。

・本DLLにデバッグを目的とした要素が含まれている場合が有りますが、当社用で有り、お客様で利用される事を

　想定していません。その為、その関係要素に関しては説明していない場合が有ります。

# 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 version 2004 | Windows10 version 2004 又はそれ以降 |
| CPU | Intel i7-3930K | Intel系CPU |
| コンパイラ | Microsoft Visual Studio 2015 pro | Microsoft Visual Studio 2015 又はそれ以降 |
| 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) |

# 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 |
| CANa31.cpp  CANa31.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 |
| CANa31.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 |
| CANa31.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 |
| CANa31dll.cpp  CANa31dll.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構築時に本コードは利用されません。 |
| 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  ユーザー側でアタッチ／デタッチ時に処理が必要な場合は、本ソースコードを改造して利用します |
| IxxatSimple.cpp  IxxatSimple.h | HMS社製、simplyCANインターフェースの制御クラス |
| 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構造体の定義 |
| resource.h  stdafx.cpp  stdafx.h  targetvar.h | Visual Studio利用時に自動作成されるファイル。  必要が有ればユーザー側で変更して下さい |
| readme.txt | 本プロジェクトの履歴 |
| CANa31.sln  CANa31.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 2015 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2種類のインターフェースに対応しています。

|  |  |  |
| --- | --- | --- |
| メーカー | HMS | |
| 名称 | USB-to-CAN V2 | |
| このインターフェースは、利用するだけでもインストール作業が必要です。 | |
| 準備手順 | No. | 手順 |
| 1 | インターネットブラウザでHMS社のサイト（以下URL）を開きます。  https://www.ixxat.com/ |
| 2 | USB-TO-CAN v2 のプロダクトから、以下のファイルをダウンロードします。  vci-v4-windows-10-8-7.zip  注意  　プロダクト内のDownloadを選んでもファイルが表示されない為、  　プロダクト画面の右の方にあるリンクからダウンロードします。 |
| 3 | 取得したファイルを解凍して以下のファイルを取り出し、実行してインストールします。  ixxat VCI Setup 4.0.939.0.exe  （ファイル名内の数字はバージョンの為、上記と多少異なる場合が有ります）  本DLLの再ビルドを行う場合、インストール先に必要なファイルが格納されています。  Visual Studioのプロジェクト設定は、ixxatV2.cppの先頭にある記述を確認して下さい。 |

|  |  |  |
| --- | --- | --- |
| メーカー | HMS | |
| 名称 | simplyCAN | |
|  | このインターフェースは、利用するだけなら何も作業は不要ですが、本DLLの再ビルド等を  行う場合には、準備が必要です。 | |
| 準備手順 | No. | 手順 |
| 1 | インターネットブラウザでHMS社のサイト（以下URL）を開きます。  https://www.ixxat.com/ |
| 2 | simplyCANのプロダクトから、以下のファイルをダウンロードします。  simplycan-driver-windows.zip  注意  　プロダクト内のDownloadを選んでもファイルが表示されない為、  　プロダクト画面の右の方にあるリンクからダウンロードします。 |
| 3 | 取得したファイルを解凍します。  本DLLの再ビルドを行う場合は、解凍先に必要なファイルが格納されています。.  Visual Studioのプロジェクト設定は、ixxatSimple.cppの先頭にある記述を確認して下さい。 |

# 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 2015 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CanA31dll.cpp/hを利用する事で、本DLLを楽に扱う事が可能です。  （「CANa31dll.cpp/hの利用方法」の項を参照）  処理の流れ等は、サンプルアプリケーションを御確認下さい。 |
| 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 2015 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CanA31dll.cpp/hを利用する事で、本DLLを楽に扱う事が可能です。  （「CANa31dll.cpp/hの利用方法」の項を参照）  処理の流れ等は、サンプルアプリケーションを御確認下さい。  注意点  　64bitアプリケーションからの利用は想定していません。  　32bitアプリケーションから御利用下さい。 |

# CANa31dll.cpp/hの利用方法

本ファイルは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 “CANa31dll.h”  static CAN\_FUNCLIST g\_func;  int test()  {  //DLLの読み込み  HANDLE hDLL = LoadLibrary(“CANa31.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(CANa31dll.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);  } |

サンプルアプリケーション

本DLLプロジェクトには以下のサンプルが付属します

|  |  |
| --- | --- |
| 要素 | 値 |
| プロジェクト名 | CanA31test |
| アプリケーション種類 | Win32 コンソールアプリケーション |
| 処理の流れ | ・DLL(CanA31.dll)を読み込みます  ・関数を楽に扱う為、CanA31dll.cpp/hを利用します  　（構造体に関数のエントリを入れます）  ・初期化の為、InitInstance関数を呼びます  ・接続されているインターフェース数を確認します  ・通信速度（ボーレート）を設定します  ・通信タイムアウト時間を設定します  ・通信元(PC)と通信先(ABH3)のIDを設定します  ・インターフェースを開きます  ・指令を初期化(abh3\_can\_init関数)します  　ソースコードではここに処理がコメント化された状態で記述されています  ・インタフェースを閉じます  ・終了処理の為、ExitInstance可数を呼びます  ・DLLを開放します |
| 備考 | 処理の流れを追うには、CanA31test.cppを見て下さい。  DLLの関数呼び出しに関しては、ソースコード上では最低限だけ残し、  残りをコメント化してあります。 |
| 注意点 | リンクしたアプリケーションの実行には、以下の要素が必要です   |  |  | | --- | --- | | ファイル名 | 説明 | | CanA31.dll | 本DLLプロジェクト出力 | | simplyCAN.dll  simplyCAN-64.dll | HMS社simplyCAN利用時に必要 | | USB-to-CAN V2のドライバ | HMS社USB-to-CAN V2利用時に必要  事前にインストールが必要 | |

# 構造体

通信の結果取得に構造体を使用します。

各関数を利用した時、どの要素として格納されるのかは、その関数説明内に記載があります。

以下定義内容

typedef struct \_CANABH3\_RESULT

{

uint32\_t nID; //受信データのCAN-ID

struct \_DP0S

{

int16\_t nOrderAY; //送信データのA/Y指令値

int16\_t nOrderBX; //送信データのB/X指令値

int32\_t nInputBit; //送信データの入力(bit対応)値

} DP0S;

union

{

uint8\_t raw[8]; //一括アクセス用

struct \_DP0R

{

int16\_t nBackSpeedA; //A速度帰還

int16\_t nBackSpeedB; //B速度帰還

int16\_t nBackSpeedY; //Y速度帰還

int16\_t nBackSpeedX; //X速度帰還

} DP0R;

struct \_DP1R

{

int32\_t nInPulseA; //Aパルス積算値

int32\_t nInPulseB; //Bパルス積算値

} DP1R;

struct \_BR0

{

uint32\_t nErrorBit; //異常フラグ

uint32\_t nWarnBit; //警告フラグ

} BR0;

struct \_BR1

{

uint32\_t nCtrlBit; //制御フラグ

uint32\_t nIOflag; //IOフラグ

} BR1;

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;

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;

struct \_BR4

{

int32\_t nInPulseA; //Aパルス積算値

int32\_t nInPulseB; //Bパルス積算値

} BR4;

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;

struct \_BR6

{

float nMonitor0; //モニタ0データ

float nMonitor1; //モニタ1データ

} BR6;

struct \_BUF

{

uint8\_t nData[8]; //8バイトデータ

} BUF;

} u;

} CANABH3\_RESULT,\*pCANABH3\_RESULT;

# 関数

## InitInstance

|  |  |
| --- | --- |
| 概要 | インターフェースの利用開始 |
| 詳細 | 使用したいインターフェースを指定して、利用を開始します |
| 構文 | CANA31API void InitInstance(int32\_t nIFnum) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nIFnum | 使用したいインターフェースを指定します   |  |  | | --- | --- | | 値 | インターフェース | | 0 | USB-to-CAN v2 | | 1 | simplyCAN | | |
| 戻り値 | 無し |
| 注意点等 | この関数を呼び出す時点では、使用したいインターフェースが接続されていなくても問題有りません。（OpenInterfaceを呼び出す時点で接続されていれば良い）  一度開いたインターフェースを切り替えたい場合は、以下手順で行って下さい。   |  |  | | --- | --- | | No. | 手順 | | 1 | CloseInterfaceを呼び出す | | 2 | ExitInstanceを呼び出す | | 3 | InitInstanceで新しいインターフェースを指定する | |

## 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 | CANA31API void ExitInstance() |
| パラメータ | 無し |
| 戻り値 | 無し |
| 注意点等 | インターフェースを開いている場合は、先にCloseInterfaceを呼び出して下さい |

## 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 | CANA31API int32\_t GetInterfaceCount() |
| パラメータ | 無し |
| 戻り値 | 指定済みのインターフェースにより、値が異なります   |  |  | | --- | --- | | インターフェース | 戻り値 | | USB-to-CAN v2 | PCに接続されている本数が戻ります | | simplyCAN | 1本以上接続されている場合は、1  1本も接続されていない場合は、0  が戻ります | |
| 注意点等 | どちらのインターフェースも、2本以上接続する事は非推奨です。 |

## OpenInterface

|  |  |
| --- | --- |
| 概要 | 指定インターフェースを開く。 |
| 詳細 | インターフェースを指定して、その回線を開きます |
| 構文 | CANA31API int32\_t OpenInterface(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を指定します  指定する値は、使用インターフェース毎に異なりますので注意が必要です   |  |  | | --- | --- | | インターフェース | 値 | | USB-to-CAN v2 | 何本目か（開始：0） | | simplyCAN | COMポート番号（開始：1） | | |
| 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 | インターフェースにsimplyCANを使用する場合、本関数の実行前にどこのCOMポートに接続されているか、デバイスマネージャ等を利用して調べる必要が有ります。  インターフェースを既に開いている時に本関数を呼びだした場合、開いているインターフェースを閉じてから、新しい設定で開きなおします。 |

## CloseInterface

|  |  |
| --- | --- |
| 概要 | 開いたインターフェースを閉じる |
| 詳細 | OpenInterfaceで開いたインターフェースを閉じます |
| 構文 | CANA31API void CloseInterface() |
| パラメータ | 無し |
| 戻り値 | 無し |
| 注意点等 | インターフェースを開いていない場合は、何もしません。 |

## GetCurrentInterface

|  |  |
| --- | --- |
| 概要 | 現在開いているインターフェース番号を取得 |
| 詳細 | OpenInterfaceを実行した時に指定したインターフェース番号を取得します |
| 構文 | CANA31API int32\_t GetCurrentInterface() |
| パラメータ | 無し |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0以上 | OpenInterface関数に指定した値が戻ります | | 上記以外 | インターフェースを開いていません | |
| 注意点等 |  |

## IsOpenInterface

|  |  |
| --- | --- |
| 概要 | 現在インターフェースを開いているか？ |
| 詳細 | 現在、インターフェースを開いた状態かどうか判断します |
| 構文 | CANA31API int32\_t IsOpenInterface() |
| パラメータ | 無し |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0以外 | インターフェースを開いています | | 0 | インターフェースを開いていません | |
| 注意点等 |  |

## SetOpenTimeout

|  |  |
| --- | --- |
| 概要 | インターフェースを開くタイムアウト時間を設定 |
| 詳細 | OpenInterfaceを実行する時、回線が開く迄待つ時間を[ms]単位で指定します |
| 構文 | CANA31API void SetOpenTimeout(uint32\_t nTimeoutMS) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTimeoutMS | インターフェースを開く処理に許容する時間[ms]  推奨値は3000 | |
| 戻り値 | 無し |
| 注意点等 | インターフェースにsimplyCANを使用している場合、本関数の指定は無視されます。  インターフェースにUSB-to-CAN v2を使用している場合は、OpenInterfaceを呼び出す前に必ず設定する必要が有ります。 |

## SetSendTimeout

|  |  |
| --- | --- |
| 概要 | 送信タイムアウト時間を設定 |
| 詳細 | インターフェースへデータ送信する場合の、許容時間を[ms]単位で指定します |
| 構文 | CANA31API void SetSendTimeout(uint32\_t nTimeoutMS) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTimeoutMS | 送信処理に許容する時間[ms]  送信処理実行時、この設定値以上の時間が掛かった場合は、異常と判定されます  推奨値は1000 | |
| 戻り値 | 無し |
| 注意点等 | インターフェースにsimplyCANを使用している場合、本関数の指定は無視されます。  インターフェースにUSB-to-CAN v2を使用している場合は、OpenInterfaceを呼び出す前に必ず設定する必要が有ります。 |

## SetRecvTimeout

|  |  |
| --- | --- |
| 概要 | 受信タイムアウト時間を設定 |
| 詳細 | インターフェースからデータ受信する場合の、許容時間を[ms]単位で指定します |
| 構文 | CANA31API void SetRecvTimeout(uint32\_t nTimeoutMS) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nTimeoutMS | 受信処理に許容する時間[ms]  受信処理実行時、この時間内に何も受信出来なかった場合は、異常と判定されます  推奨値は1000ですが、異常に対して早くリカバリする為には、小さい値を指定して下さい | |
| 戻り値 | 無し |
| 注意点等 | 本関数は、OpenInterfaceを呼び出す前に、必ず設定する必要が有ります。 |

## SetTargetID

|  |  |
| --- | --- |
| 概要 | 通信対象ABH3のアドレスを設定 |
| 詳細 | 通信を行う場合に、通信相手となるABH3を指定します |
| 構文 | CANA31API void SetTargetID(uint8\_t nAdrs) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nAdrs | 通信対象ABH3のアドレスを指定します | |
| 戻り値 | 無し |
| 注意点等 | 本関数は、OpenInterfaceを呼び出す前に、必ず設定する必要が有ります。 |

## GetTargetID

|  |  |
| --- | --- |
| 概要 | 通信対象ABH3のアドレスを取得 |
| 詳細 | 現在の通信相手となるABH3のアドレスを取得します |
| 構文 | CANA31API uint8\_t GetTargetID() |
| パラメータ | 無し |
| 戻り値 | SetTargetIDで指定された通信対象ABH3のアドレスが戻ります |
| 注意点等 |  |

## SetHostID

|  |  |
| --- | --- |
| 概要 | 通信ホストのアドレスを設定 |
| 詳細 | PCが使用する通信アドレスを指定します |
| 構文 | CANA31API 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 | |
| 戻り値 | 無し |
| 注意点等 | 本関数は、OpenInterfaceを呼び出す前に、必ず設定する必要が有ります。 |

## GetHostID

|  |  |
| --- | --- |
| 概要 | 通信ホストのアドレスを取得 |
| 詳細 | 現在、PCが使用する通信アドレスを取得します |
| 構文 | CANA31API uint8\_t GetHostID() |
| パラメータ | 無し |
| 戻り値 | SetHostIDで指定された通信ホストのアドレスが戻ります |
| 注意点等 |  |

## SetBaudrate

|  |  |
| --- | --- |
| 概要 | 通信速度を指定 |
| 詳細 | 通信速度を指定します |
| 構文 | CANA31API void SetBaudrate(uint32\_t nBaudrateKbps) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | nBaudrateKbps | HMS社製インターフェス（USB-TO-CAN v2及びsimplyCAN）が利用可能な通信速度[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 | 本関数は、OpenInterfaceを呼び出す前に、必ず設定する必要が有ります。  インターフェースを開いた後で通信速度を変更する場合は、インターフェースを  一度閉じてから本関数で新しい速度を設定し、再度インターフェースを開き直して下さい。 |

## 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 | CANA31API 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 |  |

## 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 | CANA31API 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 |

## abh3\_can\_init

|  |  |
| --- | --- |
| 概要 | 指令の初期化 |
| 詳細 | 以下の要素を一括設定します   |  |  | | --- | --- | | 要素名 | 設定値 | | A/Y指令 | 0 | | B/X指令 | 0 | | 入力(bit対応) | 0 | |
| 構文 | CANA31API int32\_t abh3\_can\_init() |
| 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 |  |

## 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 | CANA31API int32\_t abh3\_can\_cmdAY(int16\_t cmd,pCANABH3\_RESULT pPtr)  CANA31API int32\_t abh3\_can\_cmdBX(int16\_t cmd,pCANABH3\_RESULT pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | cmd | A/Y又はB/X指令値 | | pPtr | 通信結果を受け取る領域へのポインタ  （pPtr->u.DP0Rに格納される） | |
| 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 | 指定値以外に必要な値が有る場合、過去の値を使用します |

## 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 | CANA31API int32\_t abh3\_can\_cmd(int16\_t cmdAY,int16\_t cmdBX,pCANABH3\_RESULT pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | cmdAY | A/Y指令値 | | cmdBX | B/X指令値 | | pPtr | 通信結果を受け取る領域へのポインタ  （pPtr->u.DP0Rに格納される） | |
| 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 | 指定値以外に必要な値が有る場合、過去の値を使用します |

## 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  入力(bit対応) = (入力(bit対応) & ~mask) | (data & mask) |
| 構文 | CANA31API int32\_t abh3\_can\_inSet(int32\_t data,int32\_t mask,pCANABH3\_RESULT pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | data | データ値 | | mask | マスク値 | | pPtr | 通信結果を受け取る領域へのポインタ  （pPtr->u.DP0Rに格納される） | |
| 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\_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  入力(bit対応) = 入力(bit対応) & ~(1 << num) | (data << num) |
| 構文 | CANA31API int32\_t abh3\_can\_inBitSet(int8\_t num,int8\_t data,pCANABH3\_RESULT pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | num | ビット番号(0～31) | | data | 設定データ(0～1) | | pPtr | 通信結果を受け取る領域へのポインタ  （pPtr->u.DP0Rに格納される） | |
| 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 | 指定値以外に必要な値が有る場合、過去の値を使用します |

## abh3\_can\_reqPulse

|  |  |
| --- | --- |
| 概要 | 積算値のリクエスト |
| 詳細 | 以下の要素を取得します   |  | | --- | | 要素名 | | Aパルス積算値 | | Bパルス積算値 | |
| 構文 | CANA31API int32\_t abh3\_can\_reqPulse(pCANABH3\_RESULT pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | pPtr | 通信結果を受け取る領域へのポインタ  （pPtr->u.DP1Rに格納される） | |
| 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\_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 | CANA31API int32\_t abh3\_can\_reqBRD(uint8\_t num,pCANABH3\_RESULT pPtr) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | num | 番号(0x00～0xff) | | pPtr | 通信結果を受け取る領域へのポインタ  格納先は、「注意点等」を参照の事 | |
| 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、構造体の項を参照の事   |  |  | | --- | --- | | num | 格納先 | | 0x28 | pPtr->u.BR0 | | 0x29 | pPtr->u.BR1 | | 0x2a | pPtr->u.BR2 | | 0x2b | pPtr->u.BR3 | | 0x2c | pPtr->u.BR4 | | 0x2d | pPtr->u.BR5 | | 0x2e | pPtr->u.BR6 | |

## abh3\_can\_trans

|  |  |
| --- | --- |
| 概要 | マルチパケットによるTelABH3パケットの送受信 |
| 詳細 | TelABH3互換コマンドを送信し、結果を取得します |
| 構文 | CANA31API int32\_t abh3\_can\_trans(char\* sbuf,char\* rbuf,uint32\_t\* rbuflen) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | sbuf | 送信対象のTelABH3コマンド(ANSI文字列)が格納された領域へのポインタ。 | | rbuf | 送信に対する返答(ANSI文字列)を格納する領域へのポインタ。 | | rbuflen | rbufのバッファ長[bytes]を指定する変数へのポインタ。  正常通信時は、rbufに返答を格納するのに必要なサイズ[bytes]が格納される。 | |
| 戻り値 | |  |  | | --- | --- | | 戻り値 | 内容 | | 0 | 正常終了 | | -2 | 正常終了したが、返答を格納する領域が足りない。（rbufのサイズが足りない） | | -11 | 通信先からABORTを受信 | | -12 | 通信先から重複セッション要求 | | -13 | PC側送信失敗 | | -14 | 受信タイムアウト | |
| 注意点等 | rbufの領域が不十分（戻り値:-2）と判断された場合、コマンド自体は実行されている事に  注意が必要。 |

## 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の速度」に変換 |
| 構文 | CANA31API 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の速度」の値を、ユーザーが扱う速度[min-1]に変換 |
| 構文 | CANA31API 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 |

## cnvCAN2Trq

|  |  |
| --- | --- |
| 概要 | 「ABH3のトルク」を変換 |
| 詳細 | 「ABH3のトルク」の値を、ユーザーが扱うトルク[%]に変換 |
| 構文 | CANA31API float cnvCAN2Trq(int16\_t trq) |
| パラメータ | |  |  | | --- | --- | | 変数名 | 内容 | | trq | 変換元の値 | |
| 戻り値 | 変換されたトルク[%]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の負荷率」の値を、ユーザーが扱う負荷率[%]に変換 |
| 構文 | CANA31API 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のアナログ入力」の値を、ユーザーが扱うアナログ入力値[V]に変換 |
| 構文 | CANA31API 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の電源電圧値」の値を、ユーザーが扱う電源電圧値[V]に変換 |
| 構文 | CANA31API 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[%] | - | cnvCAN2Trq |
| 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 | - | - |