AspinaMotorDrive node
取扱説明書

シナノケンシ株式会社

管理No. AE21-033-01
2022/06/14

目次

表 1. 改訂履歴

版	日付	内容
00	2021/10/21	初版
01	2022/06/14	rosparam対応バージョンアップに伴う修正

表 2. 参照ドキュメント

ドキュメント	解説
取扱説明書	対象製品の取扱説明書、動作仕様書、通信仕様書を参照の上ご使用ください。
動作仕様書	
通信仕様書	

1. はじめに

1.1. 本パッケージについて

PLEXMOTIONブランドにて販売しておりますモータドライバをROS上で制御するノードです。 本パッケージを使用する際は、必ず本取扱説明書の記載条件をご確認ください。本パッケージをご使用された場合は、本取扱説明書の記載条件に同意されたものとみなします。

1.2. 対象製品

- CBA-50FFF-T49
- CBA-JHWCシリーズ
- CAIWシリーズ

1.3. 動作確認環境

- OS Ubuntu(Linux)、Raspberry Pi OS
- ROSディストリビューション Melodic Noetic

1.4. RS485通信の接続

対象機種はRS485に対応した設計です。Linuxパソコンから通信を行うにはUSB-RS485コンバータをご用意ください。
接続についての詳細は各製品の取扱説明書を参照ください。
接続するモータドライバのIDは『1～7』となるよう設定してください。

本ROSノードはモータドライバのIDを『1～7』のいずれかで使用することを前提としており、それ以外の設定では動作しません。

1.5. 提供パッケージ

drive_motor.zip

```
scripts
  ├ AspinaCommands.cpp
  ├ AspinaCommands.h
  ├ AspinaMotorDrive.cpp
  ├ param.h
  └ UserSample.cpp
CMkeLists.txt
packeage.xml
```

各ファイルの内容を以下に示します

表 3. ファイル概要

ファイル名	説明
AspinaCommands.cpp	モータドライバへの入出力用ライブラリ
AspinaCommands.h	

ファイル名	説明
AspinaMotorDrive.cpp	AspinaMotorDrive node本体プログラム
param.h	AspinaCommandsおよびAspinaMotorDrive共通のパラメータ定義
UserSample.cpp	AspinaMotorDrive node利用のサンプルプログラム
CMakeLists.txt	catkin用のビルド設定
package.xml	パッケージの名前、バージョン、作者などの情報

2. ご注意事項

2.1. 免責事項

(1)お客様がご自身で選択される本パッケージを含むその他システムに適用される規格、それらに関する法令または規制、および下記(4)(5)のソフトウェアに関する条件に関しましては、お客様ご自身でのご確認および遵守をお願いいたします。

(2)(1)に基づき、お客様の本ソフトウェアの使用に起因して第三者の権利の侵害および損害が発生した場合は、シナノケンシ株式会社は一切の責任を負いません。

(3)本パッケージは、シナノケンシ株式会社によって現状有姿のまま提供され、動作不良、不具合、第三者権利侵害、商品性および特定の目的に対する適合性または有用性に関する明示または黙示の保証は一切含まれません。いかなる場合も、本パッケージを使用した結果または使用できなかったことによることについての直接的、間接的、偶発的、例示的、または結果的損害(代替商品またはサービスの購入、使用、データ、または利益の損失を含むがこれらに限定されない)に対して、そのような損害の可能性を知らされていた如何にかかわらずシナノケンシ株式会社は責任を負わないものとします。

(4)本パッケージは、BSD Licenseに基づきライセンスされるソフトウェアが含まれています。当該ソフトウェアに関する条件は、お客様ご自身でご確認ください。
https://opensource.org/licenses/bsd-license.php

(5)本パッケージは、GNU Lesser General Public License(LGPL)に基づきライセンスされるソフトウェアを使用しています。当該ソフトウェアに関する条件は、お客様ご自身でご確認ください。
https://www.gnu.org/licenses/lgpl-3.0.en.html

(6)本説明書を、シナノケンシ株式会社の許可なしに複写、複製、再配布する事を禁じます。

(7)本パッケージの使用に対する許諾は、シナノケンシ株式会社の著作権およびその他の知的財産権の譲渡または実施権の許諾をともなうものではありません。

本資料はROSやLinuxの専門知識を持った方を対象としています。　ROSのインストール・使い方、Linuxについてのお問い合わせはサポート対象外となります。

ROSシステムで構成するシナノケンシ株式会社製品以外の機器についてはサポート対象外となります。

本資料の一部または全部を、シナノケンシ株式会社の許可なしに複写、複製、再配布することを禁じます。

本資料の記載内容は、2022/06/14時点のものです。本資料の記載内容は、改良のため予告なく変更されることがあります。

本資料は機器の通信接続確立までの手順について記載したものであって、機器個別の操作や設置および配線方法に関しては記載しておりません。通信接続手順以外の詳細に関しては、対象製品の取扱説明書を参照してください。

3. 準備

3.1. ハードウェアの準備

1. モータドライバの取扱説明書に記載されている仕様の電源を用意し、接続します。
2. ROSを動作させるPCにUSB-RS485コンバータを接続します。
   USB-RS485コンバータに、ドライバへ接続可能なハーネスを接続し、ドライバと接続します。
3. 接続を確認し電源を投入します。
4. PCからモータドライバへ接続する通信デバイス名を確認します。
   システムの /dev のデバイス一覧から、モータドライバと接続されている通信デバイスのデバイス名を確認します。
   例) /dev/ttyUSB0 など。

3.2. AspinaMotorDrive nodeのインストール方法

ご使用OSに対応したROSディストリビューションはインストール済みとし、新規にROS　ワークスペースを作成する手順からご説明いたします。

3.2.1. ユーザー・ワークスペースの作成

ワークスペース名をcatkin_wsとします。

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
$ cd ..
$ catkin_make
```

上記は以下の作業を行っています。

- ワークスペースのフォルダとパッケージを追加する src フォルダを作成
- src フォルダに移動
- ワークスペース初期化
- ワークスペースのルートフォルダに移動
- ワークスペース作成（ビルド）

3.2.2. AspinaMotorDrive nodeパッケージの追加とビルド

AspinaMotorDrive nodeパッケージが格納されたファイルdrive_motor.zipを/homeへ配置されている状態とします。

```
$ unzip /home/drive_motor.zip -d ~/catkin_ws/src
$ cd ~/catkin_ws
$ source devel/setup.bash
$ catkin_make
```

上記は以下の作業を行っています。

- AspinaMotorDrive nodeパッケージを展開
- ワークスペースのルートフォルダに移動
- ワークスペースの環境設定
- ビルド

以上でパッケージ（制御ノード、サンプルノード）のインストールが完了します。

3.3. 動作確認

AspinaMotorDrive　nodeパッケージにあるサンプルノードを使って動作確認します。 サンプルノードはモータを一定速度で10秒間回転させます。

1. モータドライバ基板のIDを『1』に設定します。サンプルノードはID1に対して命令を出すよう作成しています。
2. ROSコアの起動 端末（ターミナル）を起動し、以下の操作を行います。

```
$ roscore
```

1. AspinaMotorDrive nodeの起動 ROSコアとは別の端末（ターミナル）を起動し、以下の操作を行います。

```
$ cd ~/catkin_ws
$ source devel/setup.bash
$ rosrun drive_motor AspinaMotorDrive _com:="/dev/ttyUSB0"
```

ポート名指定のためのパラメータ"com"は省略不可です。 RS-485通信が可能なシリアルポートを"/dev/ttyUSB0"などデバイス名で指定してください。

1. サンプルノードの起動 新しい端末（ターミナル）を起動し、以下の操作を行います。

```
$ cd ~/catkin_ws
$ source devel/setup.bash
$ rosrun drive_motor UserSample
```

モータが10秒程回転し、停止します。UserSampleプログラムは終了します。AspinaMotorDrive　nodeはCtrlキーを押しながらCキーを押すことにより終了できます。

4. AspinaMotorDrive nodeの使用方法

4.1. ドライバに対して使用可能なコマンド

ドライバの動作仕様書・通信仕様書をご参照ください。

4.2. ノードの通信間隔

モータドライバは半2重通信をしていますが、AspinaMotorDrive　nodeからモータドライバへの送信はtopic受信の都度行うため、タイミングによってはモータドライバからの応答信号と干渉します。
メッセージ間は20ms以上空けて下さい。20msよりも狭い間隔でメッセージを配信した場合、そのメッセージは破棄され、エラーなどは発生しません。
指令用topicの送信には上記タイミングを留意してください。

4.3. 指令用topicの構成

std_msgs::String型にて、0から9、AからFの文字を組み合わせた16進数の文字列を使い構成します。
1文字目はID、その後2文字毎に1バイトのデータを表します。チェックサムは通信時に AspinaMotorDrive node ノードが自動で付加するため、計算・付加していただく必要はありません。

例

```
バージョン取得コマンド
1バイト目 ID 1
2バイト目 Get指令 7
3バイト目 データ長 1
4バイト目 コマンドID(バージョン取得) 1
5バイト目 チェックサム

以上の内容から、チェックサムを除いた下のような4バイトの列になります。
1, 7, 1, 1

それぞれ16進に変換します。
"01" "07" "01" "01"

これらを文字列としてつなげます。
"01070101"

最初の"0"を削除します。
"1070101"

このように作成した以下の文字列をメッセージとして送信することにより、バージョン情報の取得を行うことができます。

rostopicコマンドで発行する場合は以下のようになります。
$> rostopic pub /AspinaOrder std_msgs/String "data: '1070101'"

『AspinaResponse』として応答があり、ドライバ基板が応答したバージョンを取得できます。
```

4.4. 応答topicの構成

std_msgs::String 型にて、0から9、AからFの文字を組み合わせた16進数の文字列を使い構成されています。
文字列は2文字毎に動作仕様書・通信仕様書で定義される1バイトのデータを表します。
内容については動作仕様書・通信仕様書の返信データの説明を参照してください。

4.5. AspinaCommandsクラス

指令用topicのメッセージ用文字列作成をサポートするため、AspinaCommandsクラスを用意しています。
サンプルプログラムを例にご活用ください。

4.6. サンプルプログラム

4.6.1. ソースコード

UserSample.cpp

```
/**
 *  @file
 *      UserSample.cpp
 *  @brief
 *      Plexmotion Standard Link Test Node
 *  @attention
 *      Copyright (C) 2021 - SHINANO-KENSHI Co,Ltd
 */
/*
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *   * Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 *   * Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the distribution.
 *   * Neither the name, the copyright and the trademark of the SHINANO-KENSHI
 *     CO.,LTD. nor the names of its contributors may be used to endorse or
 *     promote products derived from this software without specific prior
 *     written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 */
```

```
35
36 #include <ros/ros.h>
37 #include "AspinaCommands.h"
38
39 AspinaCommands Aspina;  /** Aspinaノードを扱うためのクラス定義 */
40
41 /**
42  * @brief 購読時のコールバック
43  * @param [in] 購読したメッセージ
44  */
45 void SubscribeCallback(const std_msgs::String& responseMsg)
46 {
47     /**
48      * AspinaMotorDriveノードから発行されたメッセージを
49      *
   AspinaクラスのCallback関数へ入力することにより、データの読み取りが容易になります。
50      */
51     Aspina.Callback(responseMsg.data.c_str());
52 }
53
54 /**
55  * @brief メイン関数
56  */
57 int main(int argc, char **argv)
58 {
59     // ROSノードとしての初期化
60     ros::init(argc, argv, "userSample");
61
62     /* ノードハンドラ作成 */
63     ros::NodeHandle n;
64
65     /* パブリッシャ・サブスクライバ作成 */
66     ros::Publisher serialPub = n.advertise<std_msgs::String>("AspinaOrder", 1000);
67     ros::Subscriber responseSub = n.subscribe("AspinaResponse", 20,
   SubscribeCallback);
68
69     /* AspinaCommandsへpublisherのポインタを設定 */
70     Aspina.SetPublisherHandle(&serialPub);
71
72     /* モータ初期設定 */
73     uint8_t id = 1;
74     Aspina.Handshake(id);
75
76     ROS_INFO("ServoOff");
77     Aspina.PublishCommand(ASPINA_SET_SERVO_ONOFF, ASPINA_SERVO_OFF, id);
78
79     ROS_INFO("SetServoMode");
80     Aspina.PublishCommand(ASPINA_SET_SERVO_MODE, ASPINA_SERVOMODE_SPEED, id);
81
82     ROS_INFO("OrderSpeed");
83     Aspina.PublishCommand(ASPINA_SET_ORDER_SPEED, 2000, id);
```

```
84
85      ROS_INFO("ServoOn");
86      Aspina.PublishCommand(ASPINA_SET_SERVO_ONOFF, ASPINA_SERVO_ON, id);
87
88      // 回転速度の設定と監視
89      for (uint8_t iLoop = 0; iLoop < 100; iLoop++) {
90          int32_t orderSpeed = 2000 + iLoop * 10;
91          Aspina.PublishCommand(ASPINA_SET_ORDER_SPEED, orderSpeed, id);
92          Aspina.PublishCommand(ASPINA_GET_CURRENT_SPEED, id);
93          ROS_INFO("Speed %d", Aspina.GetCurSpeed(id));
94      }
95
96      // モータ停止
97      ROS_INFO("ServoOff");
98      Aspina.PublishCommand(ASPINA_SET_SERVO_ONOFF, ASPINA_SERVO_OFF, id);
99
100     return 0;
101 }
```

4.6.2. ソースコード説明

モータドライバを操作するための準備として、AspinaCommandsクラスを利用できるようにしてください。(ヘッダのインクルード、CMakeList.txtへの記述)

```
#include "AspinaCommands.h"
AspinaCommands Aspina;  /* Aspinaノードを扱うためのクラス定義 */
```

購読時のコールバック関数ではAspinaMotorDrive nodeがpublishしているtopicを受信し、内容の文字列をAspinaCommandsクラスのCallback関数へ渡してください。

```
void SubscribeCallback(const std_msgs::String& responseMsg)
{
  Aspina.Callback(responseMsg.data.c_str());
}
```

モータを動かすための命令をAspinaOrderとしてpublishし、モータからの応答をAspinaResponseとして購読します。

```
  ros::Publisher serialPub = n.advertise<std_msgs::String>("AspinaOrder", 1000);
  ros::Subscriber responseSub = n.subscribe("AspinaResponse", 20, SubscribeCallback);
```

AspinaCommandsクラスは命令用文字列を作成し、そのままpublishする機能を持っています。これを利用するため、publisherのハンドラをわたしておきます。

```
  Aspina.SetPublisherHandle(&serialPub);
```

AspinaMotorDrive nodeとモータドライバ間の通信を確立させるため、Handshake関数を実行します。

```
  Aspina.Handshake(id); // Aspinaノード、モータとのハンドシェイク
```

モータドライバへモータ回転のための初期パラメータを送信します。
最初のサーボOFF指令はすでに回転していた場合に止めるための処理です。
サーボモードはトルク制御モードと速度制御モードがあり、ここでは速度制御モードを指定しています。
次に指令速度を指定しています。単位はppsとなります。
最後にサーボON指令をしています。これによりモータの回転がスタートします。

```
  ROS_INFO("ServoOff");
  Aspina.PublishCommand(ASPINA_SET_SERVO_ONOFF, ASPINA_SERVO_OFF, id);  // サーボオフ

  ROS_INFO("SetServoMode");
  Aspina.PublishCommand(ASPINA_SET_SERVO_MODE, ASPINA_SERVOMODE_SPEED, id); //
サーボモードを速度制御モードに設定

  ROS_INFO("orderfSpeed");
  Aspina.PublishCommand(ASPINA_SET_ORDER_SPEED, 1000, id);  // 指令速度を設定

  ROS_INFO("ServoOn");
  Aspina.PublishCommand(ASPINA_SET_SERVO_ONOFF, ASPINA_SERVO_ON, id);   //
モータ運転開始
```

for文部分では、ループ毎に現在速度の取得をし表示しています。

```
  for (int iLoop = 0; iLoop < 100; iLoop++) {
    serialMsg.data = Aspina.CreateCommand(ASPINA_GET_CURRENT_SPEED, id);
    serialPub.publish(serialMsg);

    ROS_INFO("Speed %d", Aspina.GetCurSpeed(id));
    ...
  }
```

サーボOFF指令にてモータが停止します。