ABH3 シリーズ　CAN-Bus 用
ROS2 対応ソフト説明書

履歴

日付	変更内容
2021/07/20	初版

目次

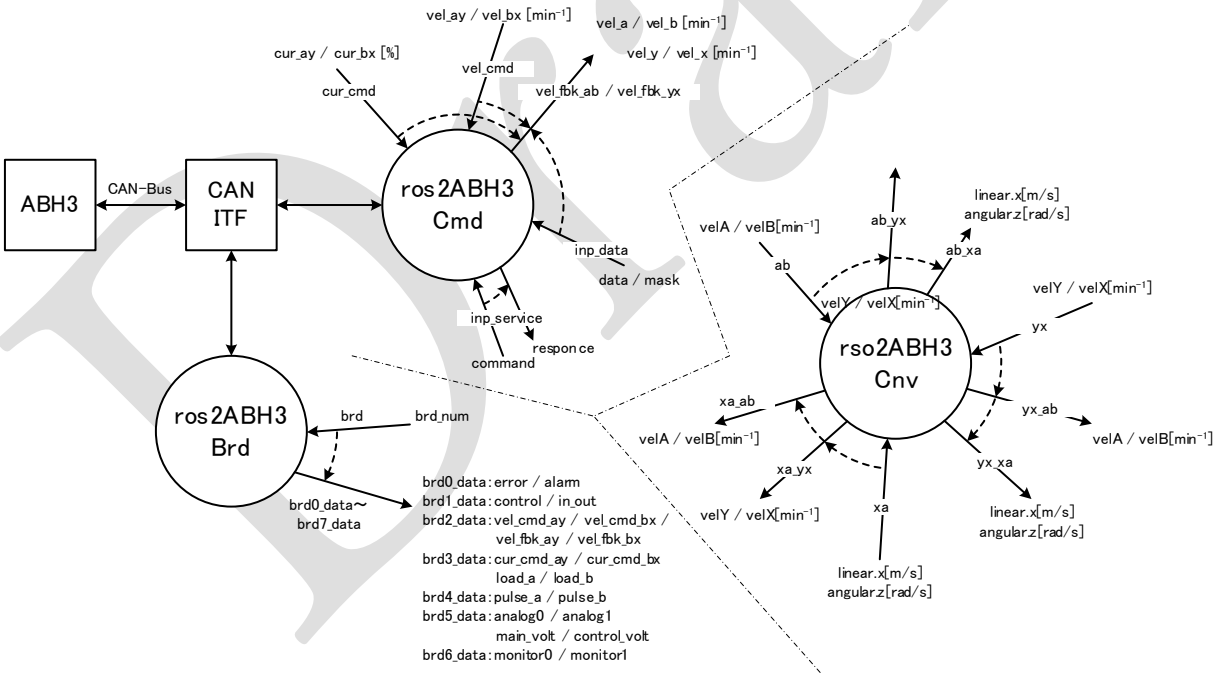
1　概要

ABH3 シリーズ・モータドライバの CAN-Bus 通信ポートを使用し、ROS2 を実装した UbuntuPC と接続・制御するためのサンプル・ソフトウェア・モジュールとなります。
ROS2 のノードとしては以下の 3 種類を実装します。

ABH3 指令ノード　：　ros2ABH3Cmd
PC と ABH3 との間で CAN-Bus による指令や制御 I/O の通信を行います。

ABH3 ブロードキャストノード　：　ros2ABH3Brd
PC と ABH3 との間で CAN-Bus による情報取得のための通信を行います。

速度値変換ノード　：　ros2ABH3Cnv
ABH3 の 2 つの速度単位モデルであるモータ軸モデル／走行軸モデルと、ROS2 の速度単位系との相互変換を行います。



1.1 インストール

動作確認は、ros2 のディストリビューション「Foxy Fitzroy」で行っています。

CAN-Bus へのアクセスのために、SocketCAN ドライバが動作する CAN インターフェースが必要になります。また、SocketCAN ドライバを使用したユーティリティ集である can-utils の一部ソースが必要となるため、以下の GitHub よりソース一式をダウンロードしてください。

https://github.com/linux-can/can-utils.git

以下のファイルがサンプル・ソフトでは必要になります。

lib.c
lib.h
include(フォルダ以下全て)

本サンプル・ソフトウェア・モジュールのダウンロードは、弊社ホームページ、および GitHub より行います。

https://github.com/wacogiken/abh3_CAN-Bus_ROS2.git

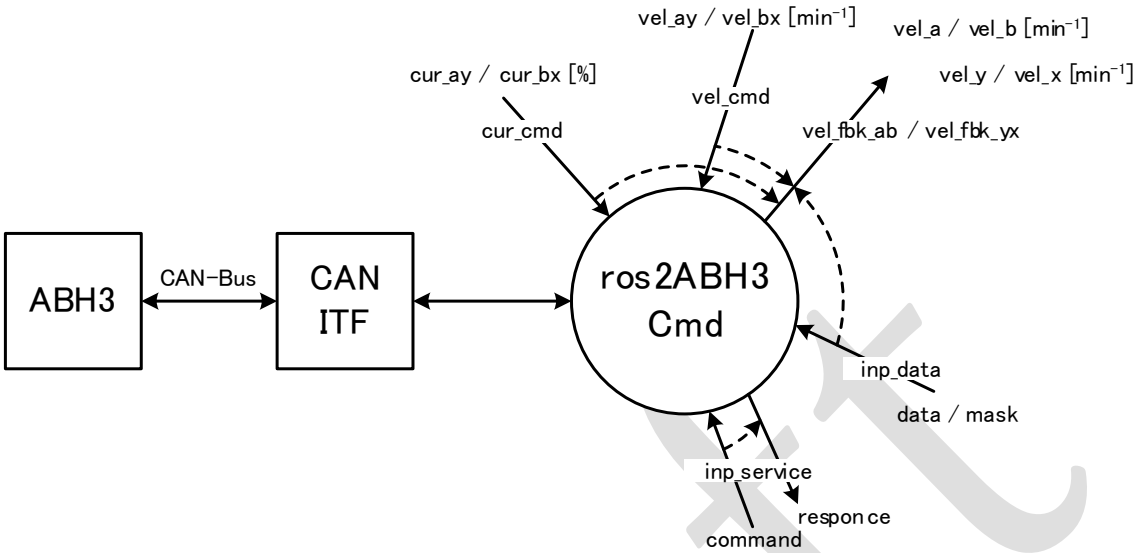
ダウンロードした以下の 3 個のフォルダを、ROS2 の src フォルダ以下にコピー(または移動)してください。

abh3_can_interface
abh3_can_launch
abh3_can_topic

また、上記に挙げた SocketCAN ドライバの 2 個のファイルと 1 個のフォルダを abh3_can_topic/src 以下にコピーしてください。

ros2 のトップフォルダにて「colcon build」によりコンパイルを行ってください。
エラーが無ければインストールは完了です。

2　ABH3 指令ノード　：　ros2ABH3Cmd（ノード名：abh3Cmd）



2.1　速度指令トピック

速度制御時に、ABH3 で設定された速度モデルの速度指令(vel_cmd)メッセージが Subscribe されると CAN-Bus により ABH3 へ速度指令値を送信します。その後、ABH3 から返された速度帰還(vel_fbk_ab / vel_fbk_yx)を Publish します。
※速度帰還は「2.4 速度帰還トピック」に記載します。

vel_cmd：速度指令[min^{-1}]
　メッセージタイプ：abh3_can_interface/msg/Abh3Velocity
　　float64　vel_ay　A 軸モータ回転速度(モータ軸モデル)、走行軸回転速度(走行軸モデル)
　　float64　vel_bx　B 軸モータ回転速度(モータ軸モデル)、旋回軸回転速度(走行軸モデル)

2.2　電流指令トピック

トルク制御時に、電流指令(cur_cmd)メッセージが Subscribe されると CAN-Bus により ABH3 へ電流指令値を送信します。その後、ABH3 から返された速度帰還(vel_fbk_ab / vel_fbk_yx)を Publish します。
※速度帰還は「2.4 速度帰還トピック」に記載します。

cur_cmd：電流指令[%]
　メッセージタイプ：abh3_can_interface/msg/Abh3Current
　　float64　cur_ay　A 軸モータ電流(モータ軸モデル)、走行軸電流(走行軸モデル)
　　float64　cur_bx　B 軸モータ電流(モータ軸モデル)、旋回軸電流(走行軸モデル)

2.3　入力トピック

制御入力の値とマスク値のメッセージが Subscribe されると CAN-Bus により ABH3 へ送信します。その後、ABH3 から返された速度帰還(vel_fbk_ab / vel_fbk_yx)を Publish します。
※速度帰還は「2.4 速度帰還トピック」に記載します。
ビット単位で機能が割り当てられています。マスクが 1 のビットが有効となります。

inp_data：入力
　メッセージタイプ：abh3_can_interface/msg/Abh3Input
　　int32　data　　入力データ
　　int32　mask　入力マスク

　データ／マスクビット

bit	内容	bit	内容	bit	内容	bit	内容
7	A/Y 補正極性	15	B/X 補正加算	23	B/X 補正極性	31	エラーリセット
6	A/Y データ選択 2	14	B/X 指令極性	22	－	30	－
5	A/Y データ選択 1	13	B/X スタート	21	－	29	－
4	A/Y データ選択 0	12	B/X サーボ ON	20	－	28	ブレーキ
3	A/Y 補正加算	11	－	19	B軸積算クリア	27	－
2	A/Y 指令極性	10	－	18	B/X データ選択 2	26	マスタ / スレーブ
1	A/Y スタート	9	－	17	B/X データ選択 1	25	B/X 速度 / トルク
0	A/Y サーボ ON	8	A軸積算クリア	16	B/X データ選択 0	24	A/Y 速度 / トルク

2.4　速度帰還トピック

速度指令、電流指令または入力トピックを受けると、ABH3 から現在の回転速度を受信し、速度帰還(vel_fbk_ab / vel_fbk_yx)として Publish します。

vel_fbk_ab：モータ軸モデル速度帰還[min^{-1}]
　メッセージタイプ：abh3_can_interface/msg/Abh3Velocity
　　float64　vel_ay　A 軸モータ回転速度
　　float64　vel_bx　B 軸モータ回転速度

vel_fbk_yx：走行軸モデル速度帰還[min^{-1}]
　メッセージタイプ：abh3_can_interface/msg/Abh3Velocity
　　float64　vel_ay　進行軸回転速度
　　float64　vel_bx　旋回軸回転速度

2.5　入力サービス

制御入力に対し、ROS2 のサービスを利用して個別に ON/OFF 制御を行う事ができます。

inp_service：入力

サービスタイプ：abh3_can_interface/srv/Abh3Input

string　command　命令文字列

---

string　responce　応答文字列

命令文字列

命令	内容
SERVO␣A␣B	サーボの ON/OFF を行う。A/B は ON/OFF/1/0 の何れかを指定する。
START␣A␣B	スタートの ON/OFF を行う。A/B は ON/OFF/1/0 の何れかを指定する。
DIR␣A␣B	指令極性の ON/OFF を行う。A/B は ON/OFF/1/0 の何れかを指定する。
CADD␣A␣B	補正加算の ON/OFF を行う。A/B は ON/OFF/1/0 の何れかを指定する。
CDIR␣A␣B	補正極性の ON/OFF を行う。A/B は ON/OFF/1/0 の何れかを指定する。
PCLR␣A␣B	積算クリアの ON/OFF を行う。A/B は ON/OFF/1/0 の何れかを指定する。 OFF から ON のタイミングでクリアされる。通常は OFF にしておく。
TORQUE␣A␣B	速度／トルク制御の切り替えを行う。A/B は ON/OFF/1/0 の何れかを指定する。 ON または 1 でトルク制御、OFF または 0 で速度制御となる。
M/S␣A␣B	マスタ・スレーブ動作の切り替えを行う。A/B は ON/OFF/1/0 の何れかを指定し、同じ値にする。 ON または 1 でマスタ・スレーブ動作、OFF または 0 で通常動作となる。
BRAKE␣A␣B	ブレーキ制御の切り替えを行う。A/B は ON/OFF/1/0 の何れかを指定し、同じ値にする。 サーボ OFF 時、ON または 1 でブレーキ開放、OFF または 0 でブレーキ動作となる。
RESET␣A␣B	異常リセットを行う。A/B は ON/OFF/1/0 の何れかを指定し、同じ値にする。 OFF から ON のタイミングでリセットされる。通常は OFF にしておく。
SELECT␣A␣B	指令データの選択を行う。A/B は 0～7 の数値で指定する。 指定 / 内部データ 0 / #7 & STOP 1 / #6 2 / #5 3 / #4 4 / #3 5 / #2 6 / #1 7 / #0

応答文字列

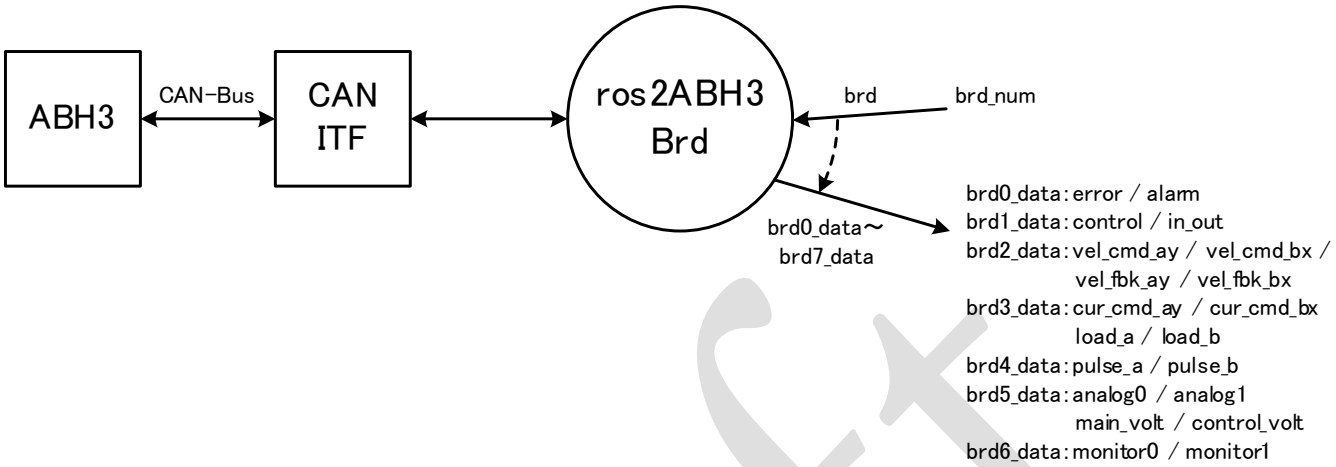
応答	内容
（空文字列）	正常応答
Packet Error: (数値)	CAN-Bus 通信エラー　※数値はエラー内容
Invalid Command.	命令文字列異常

2.6 パラメータ

ノード毎に持つパラメータで、ローンチファイル等で指定することにより、ノード毎に動作条件を変更できます。

パラメータ名称	初期値	内容
device	can0	CAN-Bus のデバイス名称 Socket-CAN ドライバ互換デバイスの場合、ifconfig/ip 命令により表示される名称。
abh3_id	1	通信対象となる ABH3 の ID 設定(0～253)
host_id	2	PC の ID 設定(0～253)
priority	0	通信プライオリティの設定(0～7)
timeout	1000	受信のタイムアウト設定[ms]

3　ABH3 ブロードキャストノード　：　ros2ABH3Brd（ノード名：abh3Brd）



3.1　状態指定トピック

ABH3 から取得する番号が Subscribe されると CAN-Bus により ABH3 へブロードキャスト送信を行います。その後、ABH3 から返された各種状態(brd0_data～brd6_data)を Publish します。
※各種状態トピックについては 3.2 以降に記載します。

brd_num：取得番号
　メッセージタイプ：std_msgs/msg/Int32
　　int32　data　　取得番号

3.2 状態 0 トピック

取得番号 0 により取得し Publish します。
データは異常フラグと警告フラグであり、ビット毎に割当てられています。

brd0_data：状態 0 データ
メッセージタイプ：abh3_can_interface/msg/Abh3Brd0
int32 error 異常フラグ
int32 alarm 警告フラグ

異常フラグ／警告フラグ

bit	内容	bit	内容	bit	内容	bit	内容
7	B 軸 過電流	15	制御電源 過電圧 主電源 過電圧	23	CAN 通信 トラフィック過大	31	－
6	A 軸 過電流	14	主電源 電圧低下	22	CAN 通信異常	30	－
5	B 軸 レゾルバ	13	B 軸 電子サーマル	21	B 軸 電流リミット	29	－
4	A 軸 レゾルバ	12	A 軸 電子サーマル	20	A 軸 電流リミット	28	－
3	ブレーキ異常	11	B 軸 PDU	19	B 軸 速度リミット	27	－
2	ドライバ過熱	10	A 軸 PDU	18	A 軸 速度リミット	26	－
1	B 軸 メカロック	9	パラメータ	17	B 軸 過速度	25	－
0	A 軸 メカロック	8	制御電源 電圧低下	16	A 軸 過速度	24	－

3.3 状態 1 トピック

取得番号 1 により取得し Publish します。
データは制御フラグと入出力フラグであり、ビット毎に割当てられています。

brd1_data：状態 1 データ
メッセージタイプ：abh3_can_interface/msg/Abh3Brd1
int32 control 制御フラグ
int32 in_out 入出力フラグ

制御フラグ

bit	内容	bit	内容	bit	内容	bit	内容
7	A/Y 補正極性	15	B/X 補正加算	23	B/X 補正極性	31	エラーリセット
6	A/Y データ選択 2	14	B/X 指令極性	22	－	30	－
5	A/Y データ選択 1	13	B/X スタート	21	－	29	－
4	A/Y データ選択 0	12	B/X サーボ ON	20	－	28	ブレーキ
3	A/Y 補正加算	11	－	19	B軸積算クリア	27	モータ軸モデル/ 走行軸モデル
2	A/Y 指令極性	10	－	18	B/X データ選択 2	26	マスタ/スレーブ
1	A/Y スタート	9	－	17	B/X データ選択 1	25	B/X 速度/トルク
0	A/Y サーボ ON	8	A軸積算クリア	16	B/X データ選択 0	24	A/Y 速度/トルク

入出力フラグ

bit	内容	bit	内容	bit	内容	bit	内容
7	エラーコード 1	15	42pin : デジタル入力 #12	23	32pin : デジタル入力 #4	31	20pin : エラーリセット入力
6	エラーコード 0	14	31pin : デジタル入力 #3	22	49pin : デジタル入力 #19	30	41pin : デジタル入力 #11
5	B/X 軸 ビジー	13	30pin : デジタル入力 #2	21	48pin : デジタル入力 #18	29	40pin : デジタル入力 #10
4	B/X 軸 レディ	12	29pin : デジタル入力 #1	20	47pin : デジタル入力 #17	28	37pin : デジタル入力 #9
3	A/Y 軸 ビジー	11	28pin : デジタル入力 #0	19	46pin : デジタル入力 #16	27	36pin : デジタル入力 #8
2	A/Y 軸 レディ	10	ブレーキ解放	18	45pin : デジタル入力 #15	26	35pin : デジタル入力 #7
1	アラーム発生	9	エラーコード 3	17	44pin : デジタル入力 #14	25	34pin : デジタル入力 #6
0	エラー発生	8	エラーコード 2	16	43pin : デジタル入力 #13	24	33pin : デジタル入力 #5

3.4 状態 2 トピック

取得番号 2 により取得し Publish します。
データは速度指令[min-1]と速度帰還[min-1]になります。

brd2_data：状態 2 データ
メッセージタイプ：abh3_can_interface/msg/Abh3Brd2

float64	vel_cmd_ay	A 軸モータ速度指令(モータ軸モデル)、走行軸速度指令(走行軸モデル)
float64	vel_cmd_bx	B 軸モータ速度指令(モータ軸モデル)、旋回軸速度指令(走行軸モデル)
float64	vel_fbk_ay	A 軸モータ速度帰還(モータ軸モデル)、走行軸速度帰還(走行軸モデル)
float64	vel_fbk_bx	B 軸モータ速度帰還(モータ軸モデル)、旋回軸速度帰還(走行軸モデル)

3.5 状態 3 トピック

取得番号 3 により取得し Publish します。
データは電流指令[%]と負荷率[%]になります。

brd3_data：状態 3 データ
メッセージタイプ：abh3_can_interface/msg/Abh3Brd3

float64	cur_cmd_ay	A 軸モータ電流指令(モータ軸モデル)、走行軸電流指令(走行軸モデル)
float64	cur_cmd_bx	B 軸モータ電流指令(モータ軸モデル)、旋回軸電流指令(走行軸モデル)
float64	load_a	A 軸モータ負荷率
float64	load_b	B 軸モータ負荷率

3.6 状態 4 トピック

取得番号 4 により取得し Publish します。
データはパルス積算値[pulse]になります。

brd4_data：状態 4 データ
メッセージタイプ：abh3_can_interface/msg/Abh3Brd4

int32	pulse_a	A 軸モータパルス積算値
int32	pulse_b	B 軸モータパルス積算値

3.7 状態 5 トピック

取得番号 5 により取得し Publish します。
データはアナログ入力電圧[V]と電源入力電圧[V]になります。

brd5_data：状態 5 データ
メッセージタイプ：abh3_can_interface/msg/Abh3Brd5
float64 analog0 アナログ入力 0 電圧
float64 analog1 アナログ入力 1 電圧
float64 main_volt 主電源電圧
float64 control_volt 制御電源電圧

3.8 状態 6 トピック

取得番号 6 により取得し Publish します。
データはモニタ出力電圧[V]になります。

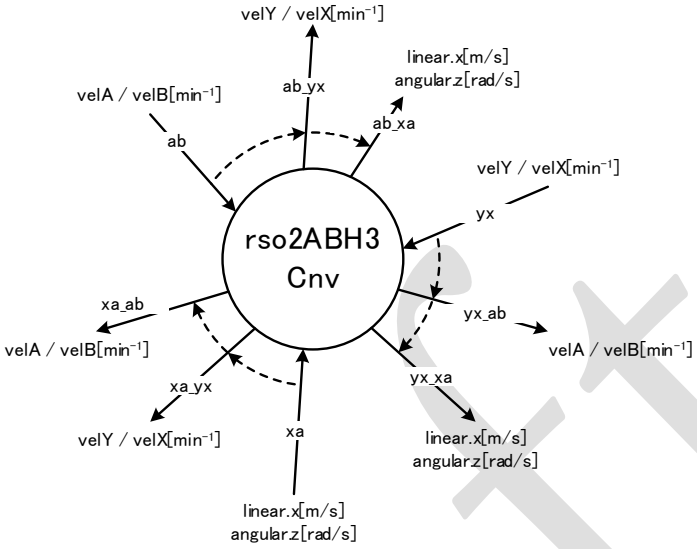
brd6_data：状態 6 データ
メッセージタイプ：abh3_can_interface/msg/Abh3Brd6
float64 monitor0 モニタ出力 0 電圧
float64 monitor1 モニタ出力 1 電圧

3.9 パラメータ

ノード毎に持つパラメータで、ローンチファイル等で指定することにより、ノード毎に動作条件を変更できます。

パラメータ名称	初期値	内容
device	can0	CAN-Bus のデバイス名称 Socket-CAN ドライバ互換デバイスの場合、ifconfig/ip 命令により表示される名称。
abh3_id	1	通信対象となる ABH3 の ID 設定(0～253)
host_id	2	PC の ID 設定(0～253)
priority	0	通信プライオリティの設定(0～7)
brd_group	5	ABH3 のブロードキャストグループ設定(0～31)
timeout	1000	受信のタイムアウト設定[ms]
mask	0x7f(127)	状態 0(bit0)～状態 6(bit6)のマスクであり、該当 bit を 1 にすると、パブリッシャーを生成し、状態取得を受け付けます。0 にするとパブリッシャーは生成せず、状態取得も受け付けません。

4　ABH3 変換ノード　：　ros2ABH3Cnv（ノード名：abh3Cnv）



4.1　ロボット座標系からの変換トピック

ロボット座標系の速度データが Subscribe されると、ABH3 座標系：走行・旋回モデルと ABH3 座標系：モータ軸モデルの速度データに変換して Publish します。

xa：ロボット座標系速度入力
　メッセージタイプ：geometry_msgs/msg/Twist
　　Vector3　linear
　　　float64　x　走行速度[m/s]
　　Vector3　angular
　　　float64　z　旋回速度[rad/s]

xa_yx：ABH3 座標系：走行・旋回モデル速度出力
　メッセージタイプ：abh3_can_interface/msg/Abh3Velocity
　　float64　vel_ay　走行軸回転速度[min-1]
　　float64　vel_bx　旋回軸回転速度[min-1]

xa_ab：ABH3 座標系：モータ軸モデル速度出力
　メッセージタイプ：abh3_can_interface/msg/Abh3Velocity
　　float64　vel_ay　A 軸モータ回転速度[min-1]
　　float64　vel_bx　B 軸モータ回転速度[min-1]

4.2　ABH3 座標系：走行・旋回モデルからの変換トピック

ABH3 座標系：走行・旋回モデルの速度データが Subscribe されると、ロボット座標系と ABH3 座標系：モータ軸モデルの速度データに変換して Publish します。

yx：ABH3 座標系：走行・旋回モデル速度入力
　メッセージタイプ：abh3_can_interface/msg/Abh3Velocity
　　float64　vel_ay　走行軸回転速度[min^{-1}]
　　float64　vel_bx　旋回軸回転速度[min^{-1}]

yx_xa：ロボット座標系速度出力
　メッセージタイプ：geometry_msgs/msg/Twist
　　Vector3　linear
　　　float64　x　走行速度[m/s]
　　Vector3　angular
　　　float64　z　旋回速度[rad/s]

yx_ab：ABH3 座標系：モータ軸モデル速度出力
　メッセージタイプ：abh3_can_interface/msg/Abh3Velocity
　　float64　vel_ay　A 軸モータ回転速度[min^{-1}]
　　float64　vel_bx　B 軸モータ回転速度[min^{-1}]

4.3 ABH3 座標系：モータ軸モデルからの変換

ABH3 座標系：モータ軸モデルの速度データが Subscribe されると、ロボット座標系と ABH3 座標系：走行・旋回モデルの速度データに変換して Publish します。

ab：ABH3 座標系：モータ軸モデル速度入力
　メッセージタイプ：abh3_can_interface/msg/Abh3Velocity
　　float64　vel_ay　A 軸モータ回転速度[min^{-1}]
　　float64　vel_bx　B 軸モータ回転速度[min^{-1}]

ab_yx：ABH3 座標系：走行・旋回モデル速度出力
　メッセージタイプ：abh3_can_interface/msg/Abh3Velocity
　　float64　vel_ay　走行軸回転速度[min^{-1}]
　　float64　vel_bx　旋回軸回転速度[min^{-1}]

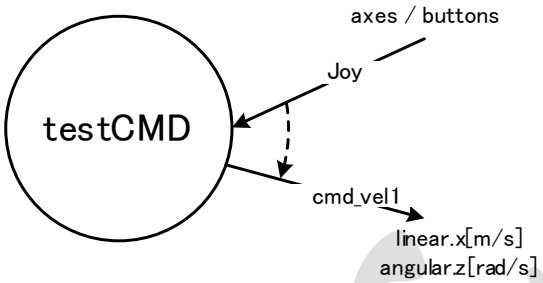
ab_xa：ロボット座標系速度出力
　メッセージタイプ：geometry_msgs/msg/Twist
　　Vector3　linear
　　　float64　x　走行速度[m/s]
　　Vector3　angular
　　　float64　z　旋回速度[rad/s]

4.4 パラメータ

ノード毎に持つパラメータで、ローンチファイル等で指定することにより、ノード毎に動作条件を変更できます。

パラメータ名称	初期値	内容
rateNum	1.0	減速比：分子
rateDen	10.0	減速比：分母
wheel	0.1	車輪径[m]
width	0.5	車輪間隔[m]

5　指令テストノード　：　testCMD（ノード名：testCMD）



5.1　ジョイスティックによる速度指令生成トピック

PS3 等のジョイスティックを用い、ABH3 に使用できるロボット座標系の速度データを Publish します。

joy：ジョイスティック入力
　メッセージタイプ：sensor_msgs/msg/Joy
　　float32[]　axes
　　　float32　axes[4]　走行指令用（PS3 右スティック前後）
　　　float32　axes[0]　旋回指令用（PS3 左スティック左右）
　　int32[]　buttons　未使用

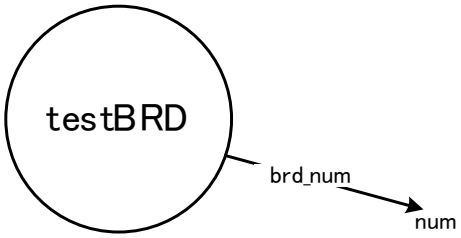
cmd_vel1：速度指令出力
　メッセージタイプ：geometry_msgs/msg/Twist
　　Vector3　linear
　　　float64　x　走行速度[m/s]
　　Vector3　angular
　　　float64　z　旋回速度[rad/s]

5.2　パラメータ

ノード毎に持つパラメータで、ローンチファイル等で指定することにより、ノード毎に動作条件を変更できます。

パラメータ名称	初期値	内容
gainX	1.0	走行指令ゲイン
gainA	1.0	旋回指令ゲイン

6　状態取得テストノード　：　testBRD（ノード名：testBRD）



6.1　状態取得番号生成トピック

パラメータで指定されたマスクと周期設定により、状態取得用の番号を生成して Publish します。

brd_num：取得番号
　メッセージタイプ：std_msgs/msg/Int32
　　int32　data　　取得番号

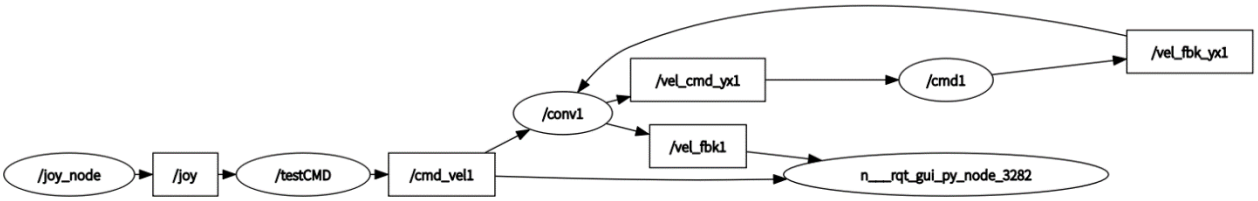
6.2　パラメータ

ノード毎に持つパラメータで、ローンチファイル等で指定することにより、ノード毎に動作条件を変更できます。

パラメータ名称	初期値	内容
freq	10	取得番号を Publish する周期（周波数）を設定する。
gainA	0x7f(127)	状態 0(bit0)～状態 6(bit6)のマスクであり、該当 bit を 1 にしたばあいはその番号が Publish される。

7　ローンチファイルサンプル

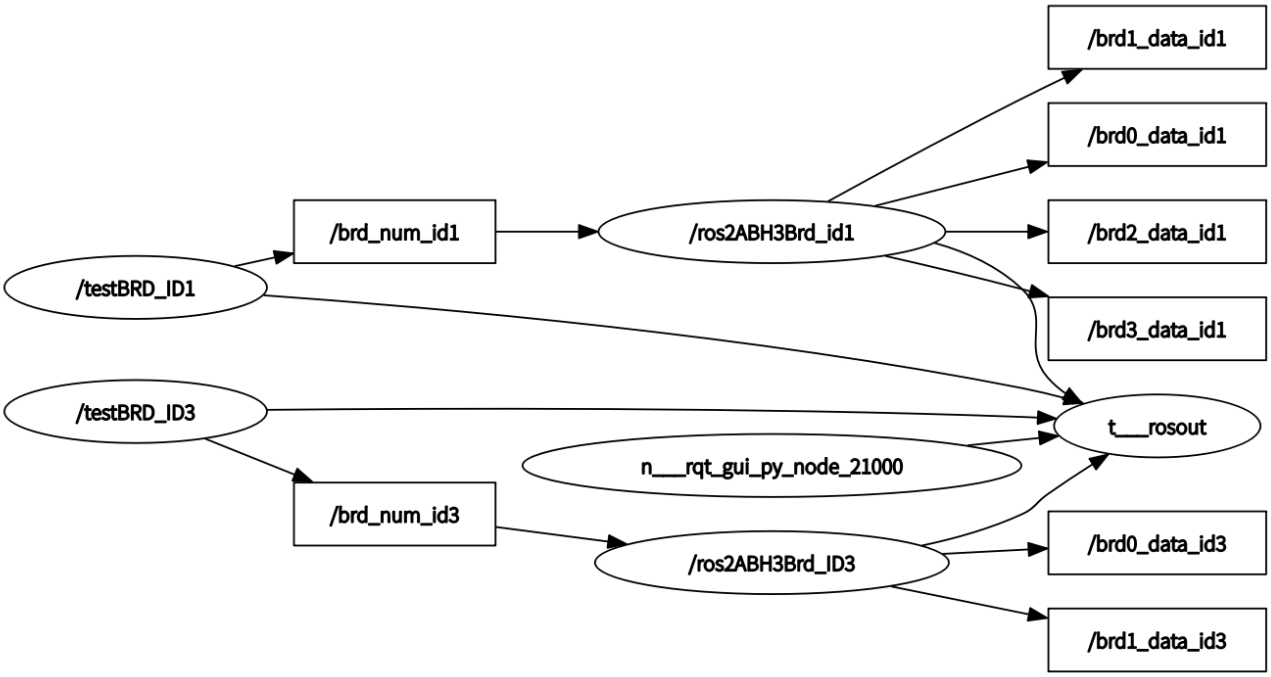
7.1　指令サンプル　：　abh3_cmd_launch.py



① USB 接続された PS3 のジョイスティック値を[/joy_node]で読み取り Publish します。
② [/testCMD]でジョイスティックの値を Subscribe し、ロボット座標系の速度指令に変換して Publish します。
③ [/conv1]ノードでロボット座標系を Subscribe し、ABH3 座標系：走行・旋回モデルに変換して速度指令を Publish します。
④ [/cmd1]ノードにより走行・旋回モデル速度指令を Subscribe し、CAN-Bus 通信で ABH3 に指令値を送信します。ABH3 は速度帰還を返信し、[/cmd1]ノードは帰還速度として Publish します。
⑤ [/conv1]ノードで走行・旋回モデル速度帰還を Subscribe し、ロボット座標系を Publish します。

①は「ros2 launch joy joy-launch.py」で起動します。
②～⑤は「ros2 launch abh3_can_launch abh3_cmd_launch.py」で起動します。

7.2　状態取得サンプル　：　abh3_brd_launch.py



① [/testBRD_ID1]で brd0～brd3(マスク値 0xf)の番号を生成し Publish します。
② [/ros2ABH3Brd_id1]で番号を Subscribe し、ID=1 の ABH3 に対しブロードキャスト通信で状態値の要求を行います。ABH3 から返信された状態値を Publish します。
③ 同様に、brd0 から brd3 を繰り返します。
④ [/testBRD_ID3]で brd0～brd1(マスク値 3)の番号を生成し Publish します。
⑤ [/ros2ABH3Brd_id3]で番号を Subscribe し、ID=3 の ABH3 に対しブロードキャスト通信で状態値の要求を行います。ABH3 から返信された状態値を Publish します。
⑥ 同様に、brd0 と brd1 を繰り返します。

①～⑥は「ros2 launch abh3_can_launch abh3_brd_launch.py」で起動します。

以上