ABH3CAN 用 Arduino Host ソフト説明資料

2023.02.10

株式会社　ワコー技研

目次

1. ハード構成

・マイコンボード：Arduino UNO

・CAN ボード：Seed Studio 社製「CAN-BUS Shield v2.0」

参考 URL： https://www.seeedstudio.com/CAN-BUS-Shield-V2.html

回路図 は上記 URL から入手できます（CAN-BUS Shield v2.0.pdf)。

「CAN-BUS Shield v2.0」のジャンパ・カット設定は次の通り

ジャンパ・カット名	定義	設定
P1	終端設定	ABH3 の通信ハード仕様に合わせて設定する。
J2	Dsub コネクタピン定義	
J3		
J4		
CS	MCP2515 の CS 接続	CS_A(デフォルト)
MISO	MCP2515 の SO 接続	MISO_A（デフォルト）
MOSI	MCP2515 の SI 接続	MOSI_A（デフォルト）
SCK	MCP2515 の SCK 接続	SCK_A（デフォルト）
INT	MCP2515 の INT 接続	INT_A（デフォルト）
CS_TF	TF カードスロットの CS に接続 ※未使用	IO4（デフォルトルト）

2.　ライブラリのインストール

・Seed Studio 社提供　CAN-BUS Shield v2.0 用ライブラリ
　「Seeed_Arduino_CAN-2.3.3.zip」を
　https://github.com/Seeed-Studio/Seeed_Arduino_CAN
から入手してください。

・ABH3CAN 通信ライブラリ
　「abh3can.zip」を
　https://github.com/wacogiken/abh3_CAN-Bus_Arduino
から入手してください。

インストール方法
　Arduino IDE（動作確認バージョン：2.0.3）のメニュー「スケッチ」→「ライブラリをインクルード」→「zip 形式をインストール」より各ファイルをインストールする。インストールされたライブラリはドキュメントフォルダ内の Arduino¥libraries 内に展開される。

3.　サンプルソフト

・sample1.ino
　スイッチでサーボ ON/OFF、START/STOP、アナログ入力で速度指令を設定する。

・sample2.ino
　　モータ運転と停止を繰り返す。ドライバ状態をブロードキャストで取得。今回用意した関数をほとんど使用している。

4. CAN 通信構造体

CAN 通信構造体“ABH3CAN” 定義

パケット形式	構造メンバ	方向	数値形式	定義	備考
シングルパケット DP0	par->DP0S.io.ay_com	HOST → ABH3	int16_t	A/Y 速度指令[0.1min^{-1}]	
	par->DP0S.io.bx_com			B/X 速度指令[0.1min^{-1}]	
	par->DP0S.io.op_flg		uint32_t	操作フラグ	
	par->DP0R.io.ayvel_fb	ABH3 → HOST	int16_t	A/Y 速度帰還[0.1min^{-1}]	
	par->DP0R.io.byvel_fb			B/Y 速度帰還[0.1min^{-1}]	
	par->DP0R.io.control_flg		uint32_t	制御フラグ	
ブロードキャストパケット BR0	par->BR0.io.error_flg	ABH3 → HOST	uint32_t	異常フラグ	
	par->BR0.io.warning_flg			警告フラグ	
ブロードキャストパケット BR1	par->BR1.io.io_flg	ABH3 → HOST	uint32_t	ＩＯフラグ	
	par->BR1.io.input_flg			入力フラグ	
ブロードキャストパケット BR2	par->BR2.io.ayvel_com	ABH3 → HOST	int16_t	A/Y 軸速度指令[0.1min^{-1}]	
	par->BR2.io.bxvel_com			BX 軸速度指令[0.1min^{-1}]	
	par->BR2.io.ayvel_fb			A/Y 軸速度帰還[0.1min^{-1}]	
	par->BR2.io.bxvel_fb			BX 軸速度帰還[0.1min^{-1}]	

ブロードキャストパケット BR3	par->BR3.io.aycrtl_com	ABH3 → HOST	int16_t	A/Y 軸電流指令[0.01%]	
	par->BR3.io.bxcrt_com			BX 軸電流指令[0.01%]	
	par->BR3.io.aload			A/Y 軸負荷率[0.1%]	
	par->BR3.io.bload			BX 軸負荷率[0.1%]	
ブロードキャストパケット BR4	par->BR4.io.apulse_integ	ABH3 → HOST	int32_t	A 軸積算パルス[pulse]	
	par->BR4.io.bpulse_integ			B 軸積算パルス[pulse]	
ブロードキャストパケット BR5	par->BR5.io.analog_in0	ABH3 → HOST	int16_t	アナログ入力 0[0.01V]	
	par->BR5.io.analog_in1			アナログ入力 1[0.01V]	
	par->BR5.io.main_volt			主電源電圧[0.1V]	
	par->BR5.io.ctrl_volt			制御電源電圧[0.1V]	
ブロードキャストパケット BR6	par->BR6.io.moni_data0	ABH3 → HOST	float	モニタデータ 0	
	par->BR6.io.moni_data1			モニタデータ 1	
CAN アドレス	par->SETTING.abh3_adrs	−	uint8_t	ABH3 の CAN アドレス	
	par->SETTING.host_adrs	−		HOST の CAN アドレス	

5. 関数

① CAN 基板の初期化

[関数]int can_init(int bps)

引数	int bps：CAN のボーレート設定 0: 250[kbps] 1: 500[kbps] 2: 1000[kbps]
戻り値	エラー状態（1：正常、0：異常）
内容	CAN 基板の初期化実行と CAN のボーレート設定を行う。
ソース	can.cpp

② CAN アドレスの設定

[関数]int can_setadrs(unsigned char abh3,unsigned char host,ABH3CAN *par)

引数	unsigned char abh3：ABH3 の CAN アドレス unsigned char host：HOST の CAN アドレス ABH3CAN*par：CAN 通信構造体ポインタ
戻り値	エラー状態（1：正常、0：異常）
内容	ABH3 と HOST の CAN アドレスを設定する。
ソース	can.cpp

ABH3CAN 構造体の変化

par->SETTING.abh3_adrs	ABH3 の CAN アドレス	引数の値に設定される。
par-> SETTING.host_adrs	HOST の CAN アドレス	引数の値に設定される。

③ 指令の初期化

[関数]int abh3_can_init(ABH3CAN *par)

引数	ABH3CAN*par：CAN 通信構造体ポインタ
戻り値	エラー状態（1：正常、0：異常）
内容	指令及び入力のすべてを 0 にし、シングルパケット DP0 を送信する。 シングルパケット DP0 を受信したら戻る。(タイムアウト判定時は異常終了)
ソース	can.cpp

ABH3CAN 構造体の変化

par->DP0S.io.ay_com	A/Y 速度指令	0 に設定される
par->DP0S.io.bx_com	B/X 速度指令	0 に設定される。
par->DP0S.io.op_flg	操作フラグ	0 に設定される。
par->DP0R.io.ayvel_fb	A/Y 速度帰還	A/Y 速度帰還が格納される。
par->DP0R.io.byvel_fb	B/Y 速度帰還	B/Y 速度帰還が格納される。
par->DP0R.io.control_flg	制御フラグ	制御フラグ状態が格納される。

④ A/Y 軸指令の送信

[関数]int abh3_can_cmdAY(short cmd, ABH3CAN *par)

引数	short cmd：指令値 ABH3CAN *par：CAN 通信構造体ポインタ
戻り値	エラー状態（1：正常、0：異常）
内容	A/Y 軸指令を設定し、シングルパケット DP0 を送信する。 シングルパケット DP0 を受信したら戻り、値を設定する。（タイムアウト判定時は異常終了）
ソース	single_packet.cpp

ABH3CAN 構造体の変化

par->DP0S.io.ay_com	A/Y 速度指令	引数の値に設定される。
par->DP0S.io.bx_com	B/X 速度指令	変更なし。
par->DP0S.io.op_flg	操作フラグ	変更なし。
par->DP0R.io.ayvel_fb	A/Y 速度帰還	A/Y 速度帰還が格納される。
par->DP0R.io.byvel_fb	B/Y 速度帰還	B/Y 速度帰還が格納される。
par->DP0R.io.control_flg	制御フラグ	制御フラグ状態が格納される。

⑤ B/X 軸指令の送信

[関数]int abh3_can_cmdBX(short cmd, ABH3CAN *par)

引数	short cmd： 指令値 ABH3CAN *par：CAN 通信構造体ポインタ
戻り値	エラー状態（1：正常、0：異常）
内容	B/X 軸指令を設定し、シングルパケット DP0 を送信する。 シングルパケット DP0 を受信したら戻り、値を設定する。（タイムアウト判定時は異常終了）

ソース	single_packet.cpp

ABH3CAN 構造体の変化

par->DP0S.io.ay_com	A/Y 速度指令	変更なし。
par->DP0S.io.bx_com	B/X 速度指令	引数の値に設定される。
par->DP0S.io.op_flg	操作フラグ	変更なし。
par->DP0R.io.ayvel_fb	A/Y 速度帰還	A/Y 速度帰還が格納される。
par->DP0R.io.byvel_fb	B/Y 速度帰還	B/Y 速度帰還が格納される。
par->DP0R.io.control_flg	制御フラグ	制御フラグ状態が格納される。

⑥ A/Y,B/X 軸指令の同時送信

[関数]int abh3_can_cmd(short cmdAY、short cmdBX, ABH3CAN *par)

引数	short cmdAY： A/Y 軸指令値 short cmdBX： B/X 軸指令値 ABH3CAN *par：CAN 通信構造体ポインタ
戻り値	エラー状態（1：正常、0：異常）
内容	A/Y，B/X 軸軸指令を設定し、シングルパケット DP0 を送信する。 シングルパケット DP0 を受信したら戻り、値を設定する。（タイムアウト判定時は異常終了）
ソース	single_packet.cpp

ABH3CAN 構造体の変化

par->DP0S.io.ay_com	A/Y 速度指令	引数の値に設定される。
par->DP0S.io.bx_com	B/X 速度指令	引数の値に設定される。
par->DP0S.io.op_flg	操作フラグ	変更なし。
par->DP0R.io.ayvel_fb	A/Y 速度帰還	A/Y 速度帰還が格納される。
par->DP0R.io.byvel_fb	B/Y 速度帰還	B/Y 速度帰還が格納される。
par->DP0R.io.control_flg	制御フラグ	制御フラグ状態が格納される。

⑦ 制御フラグの一括送信

[関数] int abh3_can_inSet(long data, long mask, ABH3CAN *par)

引数	long data： データ値 long mask： マスク値

	ABH3CAN *par：CAN 通信構造体ポインタ
戻り値	エラー状態（1：正常、0：異常）
内容	マスク値が 1 のデータを入力に設定し、シングルパケット DP0 を送信する。シングルパケット DP0 を受信したら戻り、値を設定する。（タイムアウト判定時は異常終了）
ソース	single_packet.cpp

ABH3CAN 構造体の変化

par->DP0S.io.ay_com	A/Y 速度指令	変更なし。
par->DP0S.io.bx_com	B/X 速度指令	変更なし。
par->DP0S.io.op_flg	操作フラグ	引数の値に設定される。
par->DP0R.io.ayvel_fb	A/Y 速度帰還	A/Y 速度帰還が格納される。
par->DP0R.io.byvel_fb	B/Y 速度帰還	B/Y 速度帰還が格納される。
par->DP0R.io.control_flg	制御フラグ	制御フラグ状態が格納される。

⑧　制御フラグビット操作の送信

[関数] int abh3_can_inBitSet(char num, char data, ABH3CAN *par)

引数	char num：ビット番号(0～31) char data：設定データ(0～1) ABH3CAN *par：CAN 通信構造体ポインタ
戻り値	エラー状態（1：正常、0：異常）
内容	ビット番号で指定された入力をデータの値とし、シングルパケット DP0 を送信する。シングルパケット DP0 を受信したら戻り、値を設定する。（タイムアウト判定時は異常終了）
ソース	single_packet.cpp

ABH3CAN 構造体の変化

par->DP0S.io.ay_com	A/Y 速度指令	変更なし。
par->DP0S.io.bx_com	B/X 速度指令	変更なし。
par->DP0S.io.op_flg	操作フラグ	引数の値に設定される。
par->DP0R.io.ayvel_fb	A/Y 速度帰還	A/Y 速度帰還が格納される。
par->DP0R.io.byvel_fb	B/Y 速度帰還	B/Y 速度帰還が格納される。
par->DP0R.io.control_flg	制御フラグ	制御フラグ状態が格納される。

⑨ ブロードキャストパケットのリクエスト

[関数] int abh3_can_reqBRD(int num, ABH3CAN *par)

引数	int num：番号(0x00～0xff) ABH3CAN *par：CAN 通信構造体ポインタ
戻り値	エラー状態（1：正常、0：異常）
内容	引数の番号のブロードキャストリクエストを送信する。 指定したブロードキャストを受信したら戻り、値を設定する。(タイムアウト判定時は異常終了)
ソース	broadcast.cpp

ブロードキャスト 0（num=0x28 ）時の ABH3CAN 構造体の変化

par->BR0.io.error_flg	異常フラグ	異常フラグの値を格納する。
par->BR0.io.warning_flg	警告フラグ	警告フラグの値を格納する。

ブロードキャスト 1（num=0x29 ）時の ABH3CAN 構造体の変化

par->BR1.io.io_flg	ＩＯフラグ	ＩＯフラグの値を格納する。
par->BR1.io.input_flg	入力フラグ	入力フラグの値を格納する。

ブロードキャスト 2（num=0x2a ）時の ABH3CAN 構造体の変化

par->BR2.io.ayvel_com	A/Y 軸速度指令	A/Y 軸速度指令が格納される。
par->BR2.io.bxvel_com	BX 軸速度指令	BX 軸速度指令が格納される。
par->BR2.io.ayvel_fb	A/Y 軸速度帰還	A/Y 軸速度帰還が格納される。
par->BR2.io.bxvel_fb	BX 軸速度帰還	BX 軸速度帰還が格納される。

ブロードキャスト 3（num=0x2b ）時の ABH3CAN 構造体の変化

par->BR3.io.aycrtl_com	A/Y 軸電流指令	A/Y 軸電流指令が格納される。
par->BR3.io.bxcrt_com	BX 軸電流指令	BX 軸電流指令が格納される。
par->BR3.io.aload	A/Y 軸負荷率	A 軸負荷率が格納される。
par->BR3.io.bload	BX 軸負荷率	B 軸負荷率が格納される。

ブロードキャスト 4（num=0x2c ）時の ABH3CAN 構造体の変化

par->BR4.io.apulse_integ	A 軸積算パルス	A 軸の積算パルス値が格納される。
par->BR4.io.bpulse_integ	B 軸積算パルス	B 軸の積算パルス値が格納される。

ブロードキャスト 5 （num=0x2d ）時の ABH3CAN 構造体の変化

par->BR5.io.analog_in0	アナログ入力 0	アナログ入力 0 の値が格納される。
par->BR5.io.analog_in1	アナログ入力 1	アナログ入力 1 の値が格納される。
par->BR5.io.main_volt	主電源電圧	主電源電圧の値が格納される。
par->BR5.io.ctrl_volt	制御電源電圧	制御電源電圧の値が格納される。

ブロードキャスト 6（num=0x2e ）時の ABH3CAN 構造体の変化

par->BR6.io.moni_data0	モニタデータ 0	モニタ出力 0 の値が格納される。
par->BR6.io.moni_data1	モニタデータ 1	モニタ出力 1 の値が格納される。

⑩　速度[min^{-1}]→CAN データ　変換関数

[関数] short cnvVel2CAN(float vel)

引数	float vel：速度[min^{-1}]
戻り値	CAN 形式の速度データ
内容	速度[min^{-1}]を CAN データ形式（単位 0.2[min^{-1}]）に変換する
ソース	cnv.cpp

⑪　速度 CAN データ→[min^{-1}]　変換関数

[関数] float cnvCAN2Vel(short vel)

引数	short vel：CAN 形式の速度データ
戻り値	速度[min^{-1}]
内容	CAN データ形式（単位 0.2[min^{-1}]）を速度[min^{-1}]に変換する
ソース	cnv.cpp

⑫　電流[%]→CAN データ　変換関数

[関数] short cnvCur2CAN (float trq)

引数	float trq：電流[%]
戻り値	CAN データ形式の電流データ
内容	電流[%]を CAN データ形式（単位 0.01[%]）に変換する
ソース	cnv.cpp

⑬　電流 CAN データ→[%]　変換関数

[関数] float cnvCAN2Cur(short trq)

引数	short trq：CAN 形式の電流データ
戻り値	電流[%]
内容	CAN データ形式（単位 0.01[%]）を電流[%]に変換する
ソース	cnv.cpp

⑭　負荷率 CAN データ→[%]　変換関数

[関数] float cnvCAN2Load(short load)

引数	short load：CAN 形式の負荷率データ
戻り値	負荷率[%]
内容	CAN データ形式（単位 0.1[%]）を負荷率[%]に変換する
ソース	cnv.cpp

⑮　主／制御電源電圧 CAN データ→[V]　変換関数

[関数] float cnvCAN2Volt(short volt)

引数	short volt：CAN 形式の負荷率データ
戻り値	主／制御電源電圧[V]
内容	CAN データ形式（単位 0.1[V]）を主／制御電源電圧[V]に変換する
ソース	cnv.cpp

⑯　アナログ入力 CAN データ→[V]　変換関数

[関数] float cnvCAN2Analog(short analog)

引数	short analog：CAN 形式のアナログ入力データ
戻り値	アナログ入力電圧[V]
内容	CAN データ形式（単位 0.01[V]）をアナログ入力電圧[V]に変換する
ソース	cnv.cpp

6. 各種フラグビット定義

① 操作フラグ、入力フラグ、制御フラグ

CAN 通信による操作フラグを設定することで ABH3 を操作する。入力フラグは ABH3 の外部操作入力または ABH3 の内部設定値を反映する。制御フラグは操作フラグと入力フラグの論理演算の結果が反映される。論理演算の種類は論理和と論理積があり ABH3 のパラメータで選択する。

bit	操作フラグ または 入力フラグ	制御フラグ		
		機能名	0 状態	1 状態
31		エラー発生	異常フラグ=ALL「0」	異常フラグ≠ALL「0」
30		ブレーキ解放動作	ブレーキ保持	ブレーキ解放
29		制御モデル	モータ軸	走行軸
28		B/X 軸 ビジー	指令＝0	指令≠0　条件:サーボ ON
27		B/X 軸 レディ	サーボ OFF 状態	サーボ ON 状態
26		A/Y 軸 ビジー	指令＝0	指令≠0　条件:サーボ ON
25		A/Y 軸 レディ	サーボ OFF 状態	サーボ ON 状態
24				
23				
22	エラーリセット	エラーリセット	非動作 サーボ ON 有効	0→1 遷移でエラークリア サーボ ON 無効
21	B 軸積算クリア	B 軸積算クリア	非動作	0→1 遷移でクリア
20	A 軸積算クリア	A 軸積算クリア	非動作	0→1 遷移でクリア
19	ブレーキ	ブレーキ操作	解放操作オフ	解放操作オン
18	マスタ / スレーブ	マスタ/スレーブ	M/S 無効	M/S 有効
17	B/X 速度 / トルク	B/X 速度/トルク	速度	トルク
16	A/Y 速度 / トルク	A/Y 速度/トルク	速度または M/S 有効時マスタ	トルクまたは M/S 有効時スレーブ
15	B/X 補正極性	B/X 補正極性	加算	減算
14	B/X データ選択 2	B/X データ選択 2	データ選択は内部データ選択番号の負論理 3bit コードを表します。	
13	B/X データ選択 1	B/X データ選択 1		
12	B/X データ選択 0	B/X データ選択 0		
11	B/X 補正加算	B/X 補正加算	オフ	オン
10	B/X 指令極性	B/X 指令極性	スルー	反転
9	B/X スタート	B/X スタート	指令無効	指令有効
8	B/X サーボ	B/X サーボ	サーボオフ	サーボオン
7	A/Y 補正極性	A/Y 補正極性	加算	減算
6	A/Y データ選択 2	A/Y データ選択 2	データ選択は内部データ選択番号の負論理 3bit コードを表します。	
5	A/Y データ選択 1	A/Y データ選択 1		
4	A/Y データ選択 0	A/Y データ選択 0		
3	A/Y 補正加算	A/Y 補正加算	オフ	オン
2	A/Y 指令極性	A/Y 指令極性	スルー	反転
1	A/Y スタート	A/Y スタート	指令無効	指令有効
0	A/Y サーボ ON	A/Y サーボ ON	サーボオフ	サーボオン

② 警告フラグ、異常フラグ

ドライバの警告と異常状態を示すフラグ。

bit	警告フラグ	異常フラグ	備考
31	アラームコード 3	エラーコード 3	本体数字表示に該当する BIN コードを示す
30	アラームコード 2	エラーコード 2	
29	アラームコード 1	エラーコード 1	
28	アラームコード 0	エラーコード 0	
27	未定義	未定義	
26	未定義	未定義	
25	未定義	未定義	
24	未定義	未定義	
23	CAN 通信トラフィック過大	CAN 通信トラフィック過大	CAN 通信異常
22	CAN 通信タイムアウト	CAN 通信タイムアウト	CAN 通信異常
21	B 軸 電流リミット	B 軸 電流リミット	
20	A 軸 電流リミット	A 軸 電流リミット	
19	B 軸 速度リミット	B 軸 速度リミット	
18	A 軸 速度リミット	A 軸 速度リミット	
17	B 軸 過速度	B 軸 過速度	
16	A 軸 過速度	A 軸 過速度	
15	制御電源 過電圧・主電源 過電圧	制御電源 過電圧・主電源 過電圧	
14	主電源 電圧低下	主電源 電圧低下	
13	B 軸 電子サーマル	B 軸 電子サーマル	警告判定値は パラメータ設定
12	A 軸 電子サーマル	A 軸 電子サーマル	
11	B 軸 PDU	B 軸 PDU	
10	A 軸 PDU	A 軸 PDU	
9	パラメータ	パラメータ	
8	制御電源 電圧低下	制御電源 電圧低下	
7	B 軸 過電流	B 軸 過電流	
6	A 軸 過電流	A 軸 過電流	
5	B 軸 レゾルバ	B 軸 レゾルバ	
4	A 軸 レゾルバ	A 軸 レゾルバ	
3	ブレーキ異常	ブレーキ異常	
2	ドライバ過熱	ドライバ過熱	
1	B 軸 メカロック	B 軸 メカロック	
0	A 軸 メカロック	A 軸 メカロック	

灰文字は拡張用予約フラグ

③　ＩＯフラグ

ドライバの IO コネクタの状態を反映する。

bit	IO フラグ
31	エラーリセット入力
30	
29	
28	
27	デジタル入力 #19
26	デジタル入力 #18
25	デジタル入力 #17
24	デジタル入力 #16
23	デジタル入力 #15
22	デジタル入力 #14
21	デジタル入力 #13
20	デジタル入力 #12
19	デジタル入力 #11
18	デジタル入力 #10
17	デジタル入力 #9
16	デジタル入力 #8
15	デジタル入力 #7
14	デジタル入力 #6
13	デジタル入力 #5
12	デジタル入力 #4
11	デジタル入力 #3
10	デジタル入力 #2
9	デジタル入力 #1
8	デジタル入力 #0
7	デジタル出力 #7
6	デジタル出力 #6
5	デジタル出力 #5
4	デジタル出力 #4
3	デジタル出力 #3
2	デジタル出力 #2
1	デジタル出力 #1
0	デジタル出力 #0