

ABH3CAN 用 Arduino Host ソフト説明資料

株式会社 ワコー技研

2021.3.4 初版

目次

1. ハード構成	2
2. ライブラリのインストール	3
3. サンプルソフト	3
4. CAN 通信構造体	4
5. 関数	6
① CAN 基板の初期化	6
② CAN アドレスの設定	6
③ 指令の初期化	6
④ A/Y 軸指令の送信	7
⑤ B/X 軸指令の送信	7
⑥ A/Y,B/X 軸指令の同時送信	8
⑦ 制御フラグの一括送信	9
⑧ 制御フラグビット操作の送信	9
⑨ AB 軸積算値パルスのリクエスト	10
⑩ ブロードキャストバケットのリクエスト	10
⑪ 速度[min^{-1}]→CAN データ 変換関数	12
⑫ 速度 CAN データ→[min^{-1}] 変換関数	12
⑬ 電流[%]→CAN データ 変換関数	12
⑭ 電流 CAN データ→[%] 変換関数	12
⑮ 負荷率 CAN データ→[%] 変換関数	12
⑯ 主/制御電源電圧 CAN データ→[V] 変換関数	13
⑰ アナログ入力 CAN データ→[V] 変換関数	13
6. 各種フラグビット定義	14

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」

回路図　： CAN-BUS Shield v2.0.pdf

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

「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)

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「MCP_CAN_lib-master.zip」

https://github.com/Seeed-Studio/Seeed_Arduino_CAN

にて入手してください。

- ABH3CAN 通信ライブラリ「abh3can.zip」

インストール方法

Arduino IDE（動作確認バージョン：1.8.8）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。

その他

① CAN 初期化やエラー発生時のメッセージ消したい場合は、ドキュメントフォルダ内の Arduino¥libraries¥MCP_CAN_lib-master¥mcp_can_dfs.h 内にて
#define DEBUG_MODE 1 を 未定義にする。

② CAN 通信のデータ内容を表示したい場合は、ドキュメントフォルダ内の Arduino¥libraries¥abh3can¥ABH3CAN.h 内にて
DEBUG_PRINT を定義する。

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 →	int16_t	A/Y 速度指令[0.1min ⁻¹]	
	par->DP0S.io.bx_com	ABH3		B/X 速度指令[0.1min ⁻¹]	
	par->DP0S.io.in_port		uint32_t	制御フラグ	
	par->DP0R.io.avel_fb	ABH3 →	int16_t	A 軸速度帰還[0.1min ⁻¹]	
	par->DP0R.io.bvel_fb	HOST		B 軸速度帰還[0.1min ⁻¹]	
	par->DP0R.io.yvel_fb			Y 軸速度帰還[0.1min ⁻¹]	
	par->DP0R.io.xvel_fb			X 軸速度帰還[0.1min ⁻¹]	
シングルパケット DP1	par->DP1R.io.ay_com	ABH3 →	int32_t	A 軸積算パルス[pulse]	
	par->DP1R.io.bx_com	HOST		B 軸積算パルス[pulse]	
ブロードキャストパケット BR0	par->BR0.io.error_flg		uint32_t	異常フラグ	
	par->BR0.io.warningi_flg			警告フラグ	
ブロードキャストパケット BR1	par->BR1.io.control_flg	ABH3 →	uint32_t	制御フラグ	
	par->BR1.io.io_flg	HOST		I Oフラグ	
ブロードキャストパケット BR2	par->BR2.io.ayvel_com	ABH3 →	int16_t	AY 軸速度指令[0.1min ⁻¹]	
	par->BR2.io.bxvel_com	HOST		BX 軸速度指令[0.1min ⁻¹]	
	par->BR2.io.ayvel_fb			AY 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 ト packets BR3	par->BR3.io.ayctrl_com	ABH3 →	int16_t	AY 軸電流指令[0.01%]	
	par->BR3.io.bxcrt_com	HOST		BX 軸電流指令[0.01%]	
	par->BR3.io.aload			AY 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 ト packets BR4	par->BR4.io.apulse_integ	ABH3 →	int32_t	A 軸積算パルス[pulse]	
	par->BR4.io.bpulse_integ	HOST		B 軸積算パルス[pulse]	
ブロードキャスト ト packets BR5	par->BR5.io.analog_in0	ABH3 →	int16_t	アナログ入力 0[0.01V]	
	par->BR5.io.analog_in1	HOST		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 ト packets BR6	par->BR6.io.moni_data0	ABH3 →	float	モニタデータ 0	
	par->BR6.io.moni_data1	HOST		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_setadr(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.in_port	制御フラグ	0 に設定される。
par->DP0R.io.avel_fb	A 軸速度帰還	A 軸速度帰還が格納される。
par->DP0R.io.bvel_fb	B 軸速度帰還	B 軸速度帰還が格納される。
par->DP0R.io.yvel_fb	Y 軸速度帰還	Y 軸速度帰還が格納される。
par->DP0R.io.xvel_fb	X 軸速度帰還	X 軸速度帰還が格納される。

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.in_port	制御フラグ	変更なし。
par->DP0R.io.avel_fb	A 軸速度帰還	A 軸速度帰還が格納される。
par->DP0R.io.bvel_fb	B 軸速度帰還	B 軸速度帰還が格納される。
par->DP0R.io.yvel_fb	Y 軸速度帰還	Y 軸速度帰還が格納される。
par->DP0R.io.xvel_fb	X 軸速度帰還	X 軸速度帰還が格納される。

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.in_port	制御フラグ	変更なし。
par->DP0R.io.avel_fb	A 軸速度帰還	A 軸速度帰還が格納される。
par->DP0R.io.bvel_fb	B 軸速度帰還	B 軸速度帰還が格納される。
par->DP0R.io.yvel_fb	Y 軸速度帰還	Y 軸速度帰還が格納される。
par->DP0R.io.xvel_fb	X 軸速度帰還	X 軸速度帰還が格納される。

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.in_port	制御フラグ	変更なし。
par->DP0R.io.avel_fb	A 軸速度帰還	A 軸速度帰還が格納される。
par->DP0R.io.bvel_fb	B 軸速度帰還	B 軸速度帰還が格納される。
par->DP0R.io.yvel_fb	Y 軸速度帰還	Y 軸速度帰還が格納される。
par->DP0R.io.xvel_fb	X 軸速度帰還	X 軸速度帰還が格納される。

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.in_port	制御フラグ	引数の値に設定される。
par->DP0R.io.avel_fb	A 軸速度帰還	A 軸速度帰還が格納される。
par->DP0R.io.bvel_fb	B 軸速度帰還	B 軸速度帰還が格納される。
par->DP0R.io.yvel_fb	Y 軸速度帰還	Y 軸速度帰還が格納される。
par->DP0R.io.xvel_fb	X 軸速度帰還	X 軸速度帰還が格納される。

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.in_port	制御フラグ	引数の値に設定される。
par->DP0R.io.avel_fb	A 軸速度帰還	A 軸速度帰還が格納される。
par->DP0R.io.bvel_fb	B 軸速度帰還	B 軸速度帰還が格納される。
par->DP0R.io.yvel_fb	Y 軸速度帰還	Y 軸速度帰還が格納される。
par->DP0R.io.xvel_fb	X 軸速度帰還	X 軸速度帰還が格納される。

⑨ AB 軸積算値パルスのリクエスト

[関数] int abh3_can_reqPulse(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	シングルパケット DP1 を送信する。シングルパケット DP1 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->DP1R.io.ay_com	A 軸積算パルス	A 軸の積算パルス値が格納される。
par->DP1R.io.bx_com	B 軸積算パルス	B 軸の積算パルス値が格納される。

⑩ 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.warningi_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.control_flg	制御フラグ	制御フラグの値を格納する。
par->BR1.io.io_flg	I O フラグ	I O 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	AY 軸速度指令	AY 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	AY 軸速度帰還	AY 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	AY 軸電流指令	AY 軸電流指令が格納される。
par->BR3.io.bxcrtl_com	BX 軸電流指令	BX 軸電流指令が格納される。
par->BR3.io.aload	AY 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。

⑪ 速度[min^{-1}] \rightarrow 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

⑫ 速度 CAN データ \rightarrow [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

⑬ 電流[%] \rightarrow 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

⑭ 電流 CAN データ \rightarrow [%] 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

⑮ 負荷率 CAN データ \rightarrow [%] 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

⑩ 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

⑪ 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

制御フラグ ビット定義

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

異常フラグ ビット定義

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

警告フラグ ビット定義

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

I Oフラグ ビット定義

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