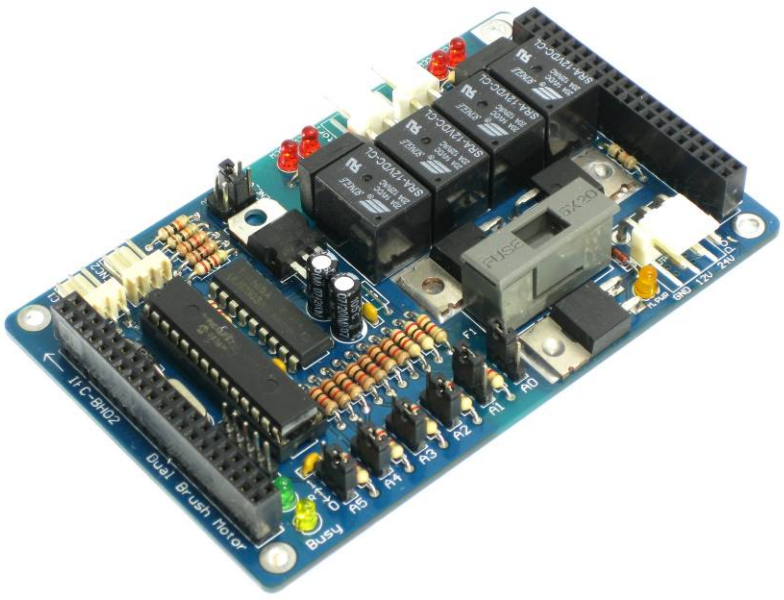
IFC-BH02
Interface Free Controller
Dual Brush Motor Card



Card Library Functions

V1.0

Jan 2009

IFC-BH02 Card Technical Info

This document explains the function prototype for BH02 used in controlling 2 brush motor. The function prototype will be called in main program for MB00 in order to control/communicate with BH02. User can also find the explanation of function prototype in its header file, "iic_bh.h". Table 1 shows the function prototype for BH02, the example card's address used in the function prototype is 'add_bh1'. Please make sure the address on IFC – BH02 is compatible with program. This technical info will explain on the function prototype of bh_'x'. 'x' = 1 or 2. When x = 1, the function prototype is for brush motor 1 (BH1) on IFC-BL02 card and when x = 2, the function prototype is for brush motor 2 (BH2) on IFC-BH02.

Function Prototype	Remark	Parameter Description

void bh_'x'_stop(unsigned char **add**)	bh_1_stop(**add_bh1**)
	bh_2_stop(**add_bh1**)
		add = card address.
		
		Stop motor connected at BH02 card.
		
		bh_1 = stop motor 1
		bh_2 = stop motor 2

void bh_'x'_brake(unsigned char **add**)	bh_1_brake(**add_bh1**)
	bh_2_brake(**add_bh1**)
		add = card address.
		
		Brake motor connected at BH02card.
		
		bh_1 = brake motor 1
		bh_2 = brake motor 2

void bh_'x'_cw(unsigned char **add**)	bh_1_cw(**add_bh1**)
	bh_2_cw(**add_bh1**)
		add = card address.
		
		Set direction of motor connected at BH02 card to Clockwise (cw).
		
		bh_1 = set motor 1 direction
		bh_2 = set motor 2 direction

void bh_'x'_ccw(unsigned char **add**)	bh_1_ccw(**add_bh1**)
	bh_2_ccw(**add_bh1**)
		add = card address.
		
		Set direction of motor connected at BH02 card to Counter-Clockwise (ccw).
		
		bh_1 = set motor 1 direction
		bh_2 = set motor 2 direction

void bh_'x'_speed(unsigned char **add**, unsigned char **pwm**)	bh_1_speed(**add_bh1**,**150**)
	bh_2_speed(**add_bh1**,**200**)
		add = card address.
		speed = speed value,0-255.
		
		Set speed of motor connected at BH02 card.
		
		bh_1 = set motor 1 speed
		bh_2 = set motor 2 speed

void bh_'x'_encon(unsigned char **add**, unsigned short **enc_data**, unsigned char **action**, unsigned char **act_val**)	bh_1_encon(**add_bh1**, **500**, **2**, **0**)
	bh_2_encon(**add_bh1**, **500**, **4**, **100**)
		add = card address.
		enc_data = encoder's data, 0-65535.
		action = action for motor, 0-5.
		act_value = speed of action, 0-255(for speed, cw, ccw)
		
		action=0 none
		action=1 stop
		action=2 brake
		action=3 cw
		action=4 ccw
		action=5 speed
		
		bh_1 = set encoder value and action for motor 1.
		bh_2 = set encoder value and action for motor 2.

void bh_'x'_enclr(unsigned char **add**)	bh_1_enclr(**add_bh1**)
	bh_2_enclr(**add_bh1**)
		add = card address.
		
		Clear encoder, 16 bits register at BH02 card.
		
		bh_1 = clear encoder 1
		bh_2 = clear encoder 2

unsigned short bh_'x'_enval(unsigned char **add**)	bh_1_enval(**add_bh1**)
	bh_2_enval(**add_bh1**)
		add = card address.
		
		Read encoder, 16 bits value (0-65535) on BH02 card.
		
		bh_1 = read encoder 1
		bh_2 = read encoder 2

unsigned char bh_'x'_runstat(unsigned char **add**)	bh_1_runstat(**add_bh1**)
	bh_2_runstat(**add_bh1**)
		add = card address.
		
		Read motor status connected at BH02 card.
		Return 1, when motor run,
		Return 0, when motor stop.
		
		bh_1 = read status for motor1
		bh_2 = read status for motor2

unsigned char bh_'x'enstat(unsigned char **add**)	bh_1_enstat(**add_bh1**)
	bh_2_enstat(**add_bh1**)
		add = card address.
		
		Read encoder status connected at BH02 card.
		Return 1, encoder in process,
		Return 0, encoder NOT in process.
		
		bh_1 = read encoder 1 status
		bh_2 = read encoder 2 status

unsigned char bh_'x'_spval(unsigned char **add**)	bh_1_spval(**add_bh1**)
	bh_2_spval(**add_bh1**)
		add = card address.
		
		Read speed value (0-255) for motor connected at BH02 card.
		
		bh_1 = read speed value for motor 1.
		bh_2 = read speed value for motor 2.

Table 1 Function Prototype for BH02

This document also shows the examples of function prototype usage. With these examples, user can understand further on the usage of BH02 function prototype. These examples will explain on the function prototype of bh_'x'. 'x' = 1 or 2. When x = 1, the function prototype is for brush motor 1(BH1) on IFC-BH02 card and when x = 2, the function prototype is for brush motor 2 (BH2) on IFC-BH02 card.

void bh_'x'_stop(unsigned char **add**)

This function is used to stop brush motor. When user calls this function, motor will stop but not immediately. User can call this function as below:

```
bh_1_stop(add_bh1);          // stop motor 1
bh_2_stop(add_bh1);          // stop motor 2
```

void bh_'x'_brake(unsigned char **add**)

This function is also used to stop brush motor. User is recommended to use 'bh_'x'_brake' function compared to 'bh_'x'_stop' function if user wants to stop brush motor immediately. A sample code to call this function is:

```
bh_1_brake(add_bh1);         // brake motor 1
bh_2_brake(add_bh1);         // brake motor 2
```

void bh_'x'_cw(unsigned char **add**)

This function is used to set the direction of the motor. When user calls this function, motor will change the direction to clockwise direction (cw). Below is a simple example for user to call this function:

```
bh_1_cw(add_bh1);            // Run motor 1 in clockwise
bh_2_cw(add_bh1);            // Run motor 2 in clockwise
```

void bh_'x'_ccw(unsigned char **add**)

This function is also used to set the direction of the motor. But this function is to change the direction to counter clockwise direction (ccw). Below is a simple example for user to call this function:

```
bh_1_ccw(add_bh1);           // Run motor 1 in counter-clockwise
bh_2_ccw(add_bh1);           // Run motor 2 in counter-clockwise
```

Note: User needs to call function 'bh_'x'_speed' to give motor speed after set the direction of the motor. The method to call function 'bh_'x'_speed' will be explained later.

void bh_'x'_speed(unsigned char **add**, unsigned char **pwm**)

This function is used to set the motor's speed. Unsigned char pwm is a value for the motor's speed. Maximum speed for motor is 255 while the minimum is 0. User also can use this function to increase or decrease the motor speed. To run the motor with speed of 150, user can call the function as below:

```
bh_1_speed(add_bh1,150);     // motor 1 speed = 150
bh_2_speed(add_bh1,250);     // motor 2 speed = 250
```

void bh_'x'_encon(unsigned char **add**, unsigned short **enc_data**, unsigned char **action**, unsigned char **act_val**)

This function is used to set the motor's action based on the encoder's value. 'enc_data' is the value to compare with the pulses from encoder which can count up to 65,535. 'action' is what the brush motor will do, which are : none (0), stop (1), brake (2), cw (3), ccw (4) or speed (5). The 'action' will be executed after pulses from encoder reach the value set in 'enc_data'. 'act_value' is the speed of motor when 'action' cw (3), ccw (4) or speed (5) is executed. The range for 'act_value' is 0 to 255. For action 0-2, 'act_value' is not used, however, user needs to put an dummy value from 0 to 255 to avoid error during compilation, the value of 'act_value' will not give any effect for the execution of action 0-2. For example, user can call the function if user wants to change the motor direction to ccw (4) and the speed of the motor to 100 ('act_value') after encoder counted 1000 ('enc_data') pulses.

```
bh_1_encon(add_bh1, 1000, 4, 100);  // change motor 1 direction
                                     // to ccw with speed 100 when
                                     // encoder 1 value = 1000.
bh_2_encon(add_bh1, 500, 1, 0);     // motor 2 stop when encoder
                                     // 2 value = 500.
```

void bh_'x'_enclr(unsigned char **add**)

This function is used to clear the value for previous pulses counted by encoder. If user wants encoder to start counting again from zero, user needs to clear the previous pulses. Use can call this function as below:

```
bh_1_enclr(add_bh1);     // clear encoder value for motor 1
bh_2_enclr(add_bh1);     // clear encoder value for motor 2
```

unsigned short bh_'x'_enval(unsigned char **add**)

bh_'x'_enval stores the value counted by the encoder. This function is used to read the encoder value and display on LCD at IFC-CP04. The encoder can count up to 65,535 pulses. A simple example to call this function is:

```
cp1_dec(bh_1_enval(add_bh1),5);     // send decimal to be displayed
                                    // by calling function cp1_dec
cp1_dec(bh_2_enval(add_bh1),5);     // send decimal to be displayed
                                    // by calling function cp1_dec
```

unsigned char bh_'x'_runstat(unsigned char **add**)

This function is used to read motor status which will return '1' if brush motor is set to run or return '0' if brush motor is set to stop in the program. This function is useful when user wants to check the motor's status set by the program. To use this function, user can write as below:

```
led4=bh_1_runstat(add_bh1); // turn OFF led4 when motor 1 brake
led5=bh_2_runstat(add_bh1); // turn ON led5 when motor 2 run
```

unsigned char bh_'x'_enstat(unsigned char **add**)

This function is used to read the encoder status. Encoder is used as a counter. Encoder status is '1' if encoder is in process (function 'bh_'x'_encon()' is called and main program is checking value of encoder and compare it with 'act_value') or return '0' if encoder is NOT in process (function 'bh_'x'_encon()' not being called). User can call this function as below:

```
led6=bh_1_enstat(add_bh1);      // turn ON led6 when encoder 1 in process
led7=bh_2_enstat(add_bh1);      // turn ON led7 when encoder 2 in process
```

unsigned char bh_'x'_spval(unsigned char **add**)

This function is used to read the speed value. Maximum value for speed is 255 while minimum is 0. User can call this function to display speed value on IFC-CP04's LCD.

```
cp1_dec(bh_1_spval(add_bh1),3);     // send decimal to be displayed
                                    // by calling function cp1_dec
cp1_dec(bh_2_spval(add_bh1),3);     // send decimal to be displayed
                                    // by calling function cp1_dec
```

Note: User needs to add header file (iic.h and iic_bh.h) and object file (iic.o and iic_bh.o) for IFC-MB00 and IFC-BH02 each time user opens a new project for IFC. User also needs to include card's header file at the beginning of the program. Please refer sample source code for the example to include card's header file.

Prepared by
Cytron Technologies Sdn. Bhd.
19, Jalan Kebudayaan 1A,
Taman Universiti,
81300 Skudai,
Johor, Malaysia.

Tel: *+607-521 3178*
Fax: *+607-521 1861*

URL: *www.cytron.com.my*
Email: *support@cytron.com.my*
sales@cytron.com.my