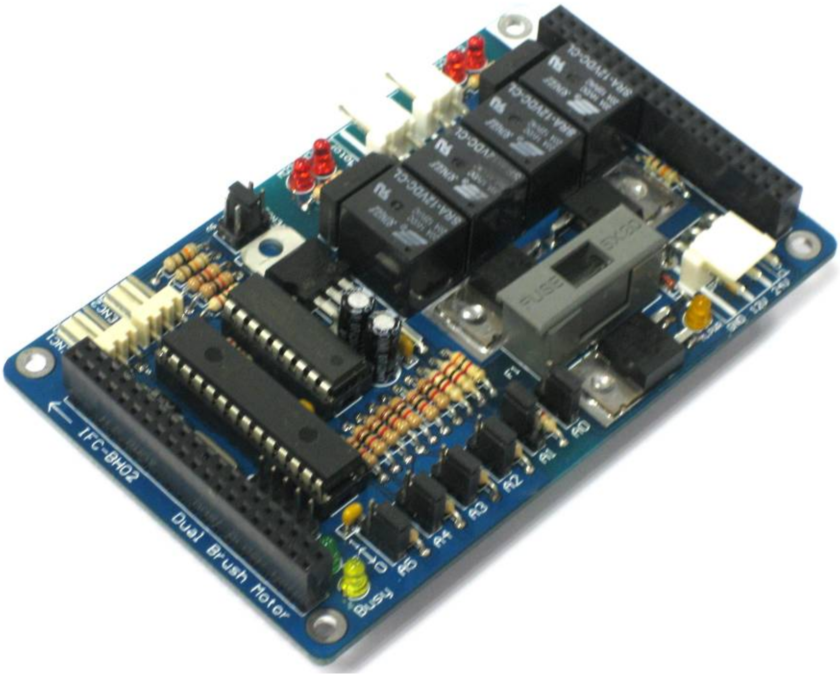
IFC-BH02
Interface Free Controller
Dual Brush Motor Card



Card Library Functions for Visual C# Express and Visual Basic Express

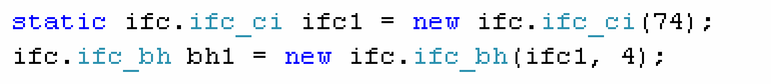
V1.0

Apr 2009

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Cytron Technologies Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Cytron Technologies's products as critical components in life support systems is not authorized except with express written approval by Cytron Technologies. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Function Prototype for Dual Brush motor (BH02)

This document explains the function prototype for controlling IFC-BH02 using PC through IFC-CI00. User may also use ‘object browser’ under Microsoft Visual C# to view the summary, parameter and return value description of IFC-BH02 function prototype. User need to add reference ‘ifc_ci.dll’ and ‘ifc_bh.dll’ for IFC-CI00 and IFC-BH02 card in order to control/communicate IFC-BH02 using PC. Please note that before user start the programming, user need to initialize the ‘ifc.ifc_ci’ and ‘ifc.ifc_bh’ in order to use the functions to control IFC-BH02. Example of creating a ‘ifc.ifc_ci’ class called ‘ifc1’ and ‘ifc.ifc_bh’ class called bh1:



```
static ifc.ifc_ci ifc1 = new ifc.ifc_ci(74);
ifc.ifc_bh bh1 = new ifc.ifc_bh(ifc1, 4);
```

For ‘ifc.ifc_ci’ class, user need to specified the COM Port that is connected to IFC-CI00 and for ‘ifc.ifc_bh’ class, user need to specified the IFC-CI00 in use and also the address for IFC-BH02. Please make sure that the address must be unique and different with other IFC card in the IFC system.

Function Prototype	Example	Summary	Parameter Description	Return Value
void bh_1_brake()	**bh1.bh_1_brake**()	To brake Motor 1.		
void bh_1_ccw()	**bh1.bh_1_ccw**()	To change the Motor 1 direction to counter-clockwise.		
void bh_1_cw()	**bh1.bh_1_cw**()	To change the Motor 1 direction to clockwise.		
void bh_1_enclr()	**bh1.bh_1_enclr**()	To clear Encoder 1 value.		
bh_1_enstat()	**bh1.bh_1_enstat**()	To read Encoder 1 stand alone encoding status.		Return true if Encoder 1 is in progress. (bool)
bh_1_enval()	**bh1.bh_1_enval**()	To read Encoder 1 value.		Encoder 1 value in 16-bit. (int)

bh_1_runstat()	**bh1.bh_1_runstat**()	To read Motor 1 run status.		Return false if Motor 1 is stopped or braked. (bool)
void bh_1_speed(int *pwm*)	**bh1.bh_1_speed(255**)	To change the speed of Motor 1.	*pwm*: Speed value of Motor 1 in range of 0 to 255. (int)	
void bh_1_speed(byte *pwm*)	**bh1.bh_1_speed(**255)	To change the speed of Motor 1.	*pwm*: Speed value of Motor 1 in range of 0 to 255. (byte)	
bh_1_spval()	**bh1.bh_1_spval**()	To read Motor 1 speed value.		Motor 1 speed value in 8-bit. (byte)
void bh_1_stop()	**bh1.bh_1_stop**()	To stop Motor 1.		
void bh_2_brake()	**bh1.bh_2_brake**()	To brake Motor 2.		
void bh_2_ccw()	**bh1.bh_2_ccw**()	To change the Motor 2 direction to counter-clockwise.		
void bh_2_cw()	**bh1.bh_2_cw**()	To change the Motor 2 direction to clockwise.		
void bh_2_enclr()	**bh1.bh_2_enclr**()	To clear Encoder 2 value.		
bh_2_enstat()	**bh1.bh_2_enstat**()	To read Encoder 2 stand alone encoding status.		Return true if Encoder 2 is in progress. (bool)
bh_2_enval()	**bh1.bh_2_enval**()	To read Encoder 2 value.		Encoder 2 value in 16-bit. (int)

bh_2_runstat()	**bh1.bh_2_runstat**()	To read Motor 2 run status.		Return false if Motor 2 is stopped or braked. (bool)
void bh_2_speed(int *pwm*)	**bh1.bh_2_speed**(180)	To change the speed of Motor 2.	*pwm*: Speed value of Motor 2 in range of 0 to 255. (int)	
void bh_2_speed(byte *pwm*)	**bh1.bh_2_speed**(180)	To change the speed of Motor 2.	*pwm*: Speed value of Motor 2 in range of 0 to 255. (byte)	
bh_2_spval()	**bh1.bh1.bh_2_spval**()	To read Motor 2 speed value.		Motor 2 speed value in 8-bit. (byte)
void bh_2_stop()	**bh1.bh_2_stop**()	To stop Motor 2.		
ifc_bh(ifc.ifc_ci *ifc_ci*, int *address*)	**ifc.ifc_bh**(ifc1, 4)	Initializes a new instance of the ifc.ifc_bh class using the specified ifc.ifc_ci and address for IFC-BH02.	*ifc_ci*: ifc.ifc_ci in use. *address*: Address for IFC-BH02, in range of 0 to 63. (int)	
ifc_bh(ifc.ifc_ci *ifc_ci*, byte *address*)	**ifc.ifc_bh**(ifc1, 4)	Initializes a new instance of the ifc.ifc_bh class using the specified ifc.ifc_ci and address for IFC-BH02.	*ifc_ci*: ifc.ifc_ci in use. *address*: Address for IFC-BH02, in range of 0 to 63. (byte)	
bh_1_encon		To enable stand alone encoding process for Encoder 1. (please refer to Table 2 for sub functions)		
bh_2_encon		To enable stand alone encoding process for Encoder 2. (please refer to Table 3 for sub functions)		

Table 1 Function Prototype for BH02

Table 2 is function prototype for sub function of '**bh_1_encon**'. These functions are to enable stand alone encoding process for Encoder 1.

Function Prototype	Example	Summary	Parameter Description
void brake(int *enc_data*)	**bh1.bh_1_encon.brake**(2000)	Motor 1 brake after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int)
void ccw(int *enc_data*, int *act_val*)	**bh1.bh_1_encon.ccw**(2000, 150)	Motor 1 change direction to counter-clockwise after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int) *act_val*: Motor 1 speed after targeted encoder value is reached. (int)
void cw(int *enc_data*, int *act_val*)	**bh1.bh_1_encon.cw**(1500, 250)	Motor 1 change direction to clockwise after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int) *act_val*: Motor 1 speed after targeted encoder value is reached. (int)
void none(int *enc_data*)	**bh1.bh_1_encon.none**(1500)	No action after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int)
void speed(int *enc_data*, int *act_val*)	**bh1.bh_1_encon.speed**(1500, 150)	Motor 1 change speed after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int) *act_val*: Motor 1 speed after targeted encoder value is reached. (int)
void stop(int *enc_data*)	**bh1.bh_1_encon.stop**(2000)	Motor 1 stop after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int)

Table 2

Table 3 is function prototype for sub function of ‘**bh_2_encon**’. These functions are to enable stand alone encoding process for Encoder 2.

Function Prototype	Example	Summary	Parameter Description
void brake(int *enc_data*)	**bh1.bh_2_encon.brake**(2000)	Motor 2 brake after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int)
void ccw(int *enc_data*, int *act_val*)	**bh1.bh_2_encon.ccw**(2000, 150)	Motor 2 change direction to counter-clockwise after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int) *act_val*: Motor 2 speed after targeted encoder value is reached. (int)
void cw(int *enc_data*, int *act_val*)	**bh1.bh_2_encon.cw**(1500, 250)	Motor 2 change direction to clockwise after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int) *act_val*: Motor 2 speed after targeted encoder value is reached. (int)
void none(int *enc_data*)	**bh1.bh_2_encon.none**(1500)	No action after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int)
void speed(int *enc_data*, int *act_val*)	**bh1.bh_2_encon.speed**(1500, 150)	Motor 2 change speed after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int) *act_val*: Motor 2 speed after targeted encoder value is reached. (int)
void stop(int *enc_data*)	**bh1.bh_2_encon.stop**(2000)	Motor 2 stop after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int)

Table 3

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