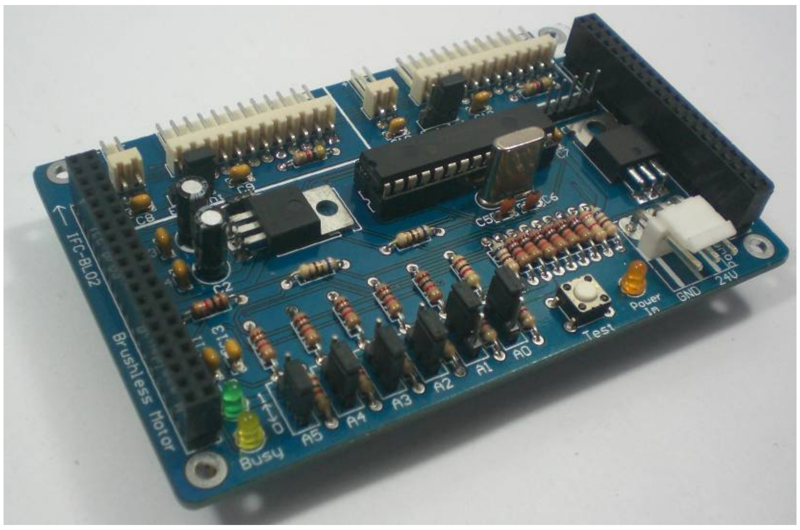
IFC-BL02
Interface Free Controller
Brushless Motor



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

V1.0

Apr 2009

Function Prototype for Brushless Motor (BL02)

This document explains the function prototype for controlling IFC-BL02 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-BL02 function prototype. User need to add reference ‘ifc_ci.dll’ and ‘ifc_bl.dll’ for IFC-CI00 and IFC-BL02 card in order to control/communicate IFC-BL02 using PC. Please note that before user start the programming, user need to initialize the ‘ifc.ifc_ci’ and ‘ifc.ifc_bl’ in order to use the functions to control IFC-BL02. Example of creating a ‘ifc.ifc_ci’ class called ‘ifc1’ and ‘ifc.ifc_bl’ class called bl1:

```
static ifc.ifc_ci ifc1 = new ifc.ifc_ci(74);
ifc.ifc_bl bl1 = new ifc.ifc_bl(ifc1, 5);
```

For ‘ifc.ifc_ci’ class, user need to specified the COM Port that is connected to IFC-CI00 and for ‘ifc.ifc_bl’ class, user need to specified the IFC-CI00 in use and also the address for IFC-BL02. 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 **bl_1_brake**()	**bl1.bl_1_brake**()	To brake BL1.		
void **bl_1_ccw**()	**bl1.bl_1_ccw**()	To change the BL1 direction to counter-clockwise.		
void **bl_1_cw**()	**bl1.bl_1_cw**()	To change the BL1 direction to clockwise.		
void **bl_1_enclr**()	**bl_1_enclr**()	To clear BL1 encoder value.		
bl_1_enstat()	**bl1.bl_1_enstat**()	To read BL1 stand alone encoding status.		Return true if BL1 encoder is in progress. (bool)
bl_1_enval()	**bl1.bl_1_enval**()	To read BL1 encoder value.		BL1 encoder value in 16-bit. (int)

bl_1_runstat()	**bl1.bl_1_runstat**()	To read BL1 run status.		Return false if BL1 is stopped or braked. (bool)
void bl_1_speed(int *speed*)	**bl1.bl_1_speed**(200)	To change the speed of BL1.	*speed*: Speed value of BL1 in range of 0 to 255. (int)	
void bl_1_speed(byte *speed*)	**bl1.bl_1_speed**(150)	To change the speed of BL1.	*speed*: Speed value of BL1 in range of 0 to 255. (byte)	
bl_1_spval()	**bl1.bl_1_spval**()	To read BL1 speed value.		BL1 speed value in 8-bit. (byte)
void bl_1_stop()	**bl1.bl_1_stop**()	To stop BL1.		
void bl_2_brake()	**bl1.bl_2_brake**()	To brake BL2.		
void bl_2_ccw()	**bl1.bl_2_ccw**()	To change the BL2 direction to counter-clockwise.		
void bl_2_cw()	**bl1.bl_2_cw**()	To change the BL2 direction to clockwise.		
void bl_2_enclr()	**bl1.bl_2_enclr**()	To clear BL2 encoder value.		
bl_2_enstat()	**bl1.bl_2_enstat** ()	To read BL2 stand alone encoding status.		Return true if BL2 encoder is in progress. (bool)
bl_2_enval()	**bl1.bl_2_enval**()	To read BL2 encoder value.		BL2 encoder value in 16-bit. (int)

bl_2_runstat()	**bl1.bl_2_runstat**()	To read BL2 run status.		Return false if BL2 is stopped or braked. (bool)
void bl_2_speed(int *speed*)	**bl1.bl_2_speed(255)**	To change the speed of BL2.	*speed*: Speed value of BL2 in range of 0 to 255. (int)	
void bl_2_speed(byte *speed*)	**bl1.bl_2_speed(180)**	To change the speed of BL2.	*speed*: Speed value of BL2 in range of 0 to 255. (byte)	
bl_2_spval()	**bl1.bl_2_spval**()	To read BL2 speed value.		BL2 speed value in 8-bit. (byte)
void bl_2_stop()	**bl1.bl_2_stop**()	To stop BL2.		
ifc_bl(ifc.ifc_ci *ifc_ci*, int *address*)	**ifc.ifc_bl**(ifc1, 5)	Initializes a new instance of the ifc.ifc_bl class using the specified ifc.ifc_ci and address for IFC-BL02.	*ifc_ci*: ifc.ifc_ci in use. *address*: Address for IFC-BL02, in range of 0 to 63. (int)	
ifc_bl(ifc.ifc_ci *ifc_ci*, byte *address*)	**ifc.ifc_bl**(ifc1, 5)	Initializes a new instance of the ifc.ifc_bl class using the specified ifc.ifc_ci and address for IFC-BL02.	*ifc_ci*: ifc.ifc_ci in use. *address*: Address for IFC-BL02, in range of 0 to 63. (byte)	
bl_1_encon		To enable stand alone encoding process for BL1.		
bl_2_encon		To enable stand alone encoding process for BL2.		

Table 1 Function Prototype for BL02

Table 2 is function prototype for sub function of ‘**bl_1_encon**’. These functions are to enable stand alone encoding process for BL1.

Function Prototype	Example	Summary	Parameter Description
void bl1_bl2_brake(int *enc_data*)	**bl1.bl_1_encon.bl1_bl2_brake(**2000)	BL1 and BL2 brake after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int)
void bl1_bl2_ccw(int *enc_data*, int *act_value1*, int *act_value2*)	**bl1.bl_1_encon.bl1_bl2_ccw(**2000, 100, 150)	BL1 and BL2 change direction to counter-clockwise after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int) *act_value1*: BL1 speed after targeted encoder value is reached. (int) *act_value2*: BL2 speed after targeted encoder value is reached. (int)
void bl1_bl2_cw(int *enc_data*, int *act_value1*, int *act_value2*)	**bl1.bl_1_encon.bl1_bl2_cw(**1500, 150, 255)	BL1 and BL2 change direction to clockwise after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int) *act_value1*: BL1 speed after targeted encoder value is reached. (int) *act_value2*: BL2 speed after targeted encoder value is reached. (int)
void bl1_bl2_stop(int *enc_data*)	**bl1.bl_1_encon.bl1_bl2_stop(**500)	BL1 and BL2 stop after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int)
void bl1_brake(int *enc_data*)	**bl1.bl_1_encon.bl1_brake(**1000)	BL1 brake after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int)
void bl1_ccw(int *enc_data*, int *act_value1*)	**bl1.bl_1_encon.bl1_ccw(**1000, 255)	BL1 change direction to counter-clockwise after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int) *act_value1*: BL1 speed after targeted encoder value is reached. (int)

void bl1_ccw_bl2_cw(int *enc_data*, **int** *act_value1*, **int** *act_value2*)	**bl1.bl_1_encon.bl1_ccw_bl2_cw(1000, 150, 150)**	BL1 change direction to counter-clockwise and BL2 change direction to clockwise after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int) *act_value1*: BL1 speed after targeted encoder value is reached. (int) *act_value2*: BL2 speed after targeted encoder value is reached. (int)
void bl1_cw(int *enc_data*, **int** *act_value1*)	**bl1.bl_1_encon.bl1_cw(1000, 250)**	BL1 change direction to clockwise after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int) *act_value1*: BL1 speed after targeted encoder value is reached. (int)
void bl1_cw_bl2_ccw(int *enc_data*, **int** *act_value1*, **int** *act_value2*)	**bl1.bl_1_encon.bl1_cw_bl2_ccw(2000, 150, 250)**	BL1 change direction to clockwise and BL2 change direction to counter-clockwise after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int) *act_value1*: BL1 speed after targeted encoder value is reached. (int) *act_value2*: BL2 speed after targeted encoder value is reached. (int)
void bl1_speed(int *enc_data*, **int** *act_value1*)	**bl1.bl_1_encon.bl1_speed(5000, 100)**	BL1 change speed after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int) *act_value1*: BL1 speed after targeted encoder value is reached. (int)
void bl1_stop(int *enc_data*)	**bl1.bl_1_encon.bl1_stop(1000)**	BL1 stop after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int)
void none(int *enc_data*)	**bl1.bl_1_encon.none(1000)**	No action 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 '**bl_2_encon**'. These functions are to enable stand alone encoding process for BL2.

Function Prototype	Example	Summary	Parameter Description
void bl1_bl2_brake(int *enc_data*)	**bl1.bl_2_encon.bl1_bl2_brake(2000)**	BL1 and BL2 brake after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int)
void bl1_bl2_ccw(int *enc_data*, **int** *act_value1*, **int** *act_value2*)	**bl1.bl_2_encon.bl1_bl2_ccw(2000, 150, 150)**	BL1 and BL2 change direction to counter-clockwise after targeted encoder value is reached.	
void bl1_bl2_cw(int *enc_data*, **int** *act_value1*, **int** *act_value2*)	**bl1.bl_2_encon.bl1_bl2_cw(2000, 200, 200)**	BL1 and BL2 change direction to clockwise after targeted encoder value is reached.	
void bl1_bl2_stop(int *enc_data*)	**bl1.bl_2_encon.bl1_bl2_stop(1500)**	BL1 and BL2 stop after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int)
void bl1_ccw_bl2_cw(int *enc_data*, **int** *act_value1*, **int** *act_value2*)	**bl1.bl_2_encon.bl1_ccw_bl2_cw(2000, 100, 150)**	BL1 change direction to counter-clockwise and BL2 change direction to clockwise after targeted encoder value is reached.	
void bl1_cw_bl2_ccw(int *enc_data*, **int** *act_value1*, **int** *act_value2*)	**bl1.bl_2_encon.bl1_cw_bl2_ccw(2000, 150, 150)**	BL1 change direction to clockwise and BL2 change direction to counter-clockwise after targeted encoder value is reached.	**void bl1_cw_bl2_ccw(int** *enc_data*, **int** *act_value1*, **int** *act_value2*)
void bl2_brake(int *enc_data*)	**bl1.bl_2_encon.bl2_brake(1500)**	BL2 brake after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int)

void bl2_ccw(int *enc_data*, int *act_value1*)	**bl1.bl_2_encon.bl2_ccw**(2000, 150)	BL2 change direction to counter-clockwise after targeted encoder value is reached.	
void bl2_cw(int *enc_data*, int *act_value1*)	**bl1**.**bl2_cw**(2000, 255)	BL2 change direction to clockwise after targeted encoder value is reached.	
void bl2_speed(int *enc_data*, int *act_value1*)	**bl1.bl_2_encon.bl2_speed**(2000, 180)	BL2 change speed after targeted encoder value is reached.	
void bl2_stop(int *enc_data*)	**bl1.bl_2_encon.bl2_stop**(2000)	BL2 stop after targeted encoder value is reached.	*enc_data*: Targeted encoder value in range of 0 to 65535. (int)
void none(int *enc_data*)	**bl1.bl_2_encon.none**(1500)	No action 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