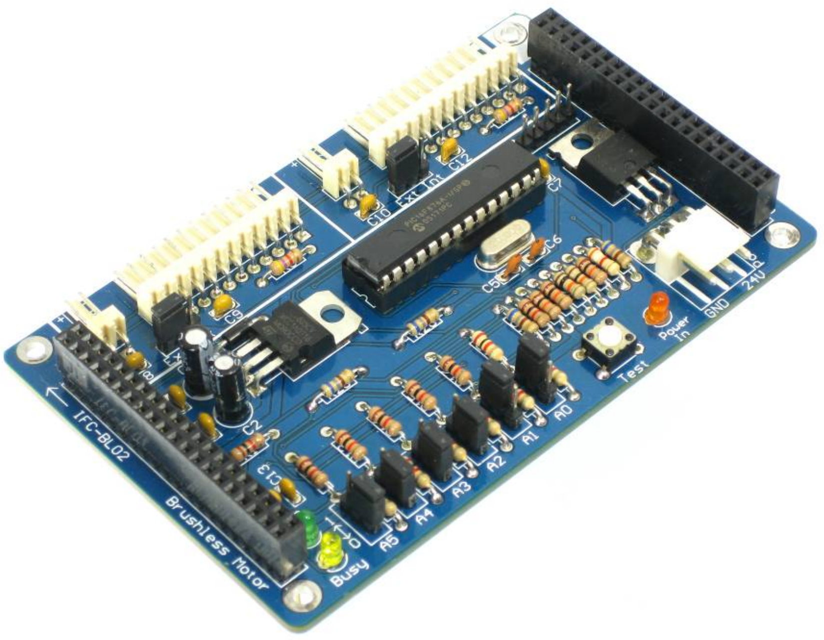
IFC-BL02
Interface Free Controller
Brushless Motor



Card Library Functions

V1.2

March 2010

IFC-BL02 Card Technical Info

This document explains the function prototype for BL02 needed in controlling 2 brushless motor in one time. The function prototype will be called in main program for MB00 in order to control/communicate with BL02. User can also find the explanation of function prototype in its header file, “iic_bl.h”. Table 1 shows function prototype for BL02, the example card address used in the function prototype is ‘add_bl1’. Please make sure the address on IFC-BL02 is compatible with program. This technical info will explain on the function prototype of bl_’x’. ‘x’ = 1 or 2. When x = 1, the function prototype is for brushless motor1 (BL1) on IFC-BL02 card and when x = 2, the function prototype is for brushless motor2 (BL2) on IFC-BL02 card.

Function Prototype	Remark	Parameter Description

void bl_’x’_stop(unsigned char **add**)	bl_1_stop(**add_bl1**)
	bl_2_stop(**add_bl1**)
		add = card address.
		Stop motor connected at BL02 card.
		bl_1 = stop BL1
		bl_2 = stop BL2

void bl_’x’_brake(unsigned char **add**)	bl_1_brake(**add_bl1**)
	bl_2_brake(**add_bl1**)
		add = card address.
		Brake motor connected at BL02 card.
		bl_1 = brake BL1
		bl_2 = brake BL2

void bl_’x’_cw(unsigned char **add**)	bl_1_cw(**add_bl1**)
	bl_2_cw(**add_bl1**)
		add = card address.
		Set direction of motor connected at BL02 card to Clockwise (cw).
		bl_1 = set BL1’s direction
		bl_2 = set BL2’s direction

void bl_’x’_ccw (unsigned char **add**)	bl_1_ccw(**add_bl1**)
	bl_2_ccw(**add_bl1**)
		add = card address.
		Set direction of motor connected at BL02 card to Counter-Clockwise (ccw).
		bl_1 = set BL1’s direction
		bl_2 = set BL2’s direction

void bl_’x’_speed(unsigned char **add**, unsigned char **speed**)	bl_1_speed(**add_bl1,100**)
	bl_2_speed(**add_bl1,100**)
		add = card address.
		speed = speed value,0-255.
		Set speed of motor connected at BL02 card.
		bl_1 = set speed for BL1
		bl_2 = set speed for BL2

void bl_1_encon(unsigned char **add**, unsigned short **enc_data**, unsigned char **action**, unsigned char **act_value1,**unsigned char **act_value2**)	bl_1_encon(**add_bl1, 2000,9,100,200**)	**add** = card address. **enc_data** = encoder’s data, 0-65536. **action** = action for BL1 and BL2, (0-11).

		act_value1 = value of BL1 action, 0-255(for speed, cw, ccw) **act_value2** = value of BL2 action, 0-255(for speed, cw, ccw) **action=0** none **action=1** BL1 stop **action=2** BL1 brake **action=3** BL1 CW **action=4** BL1 CCW **action=5** BL1 Speed **action=6** BL1,BL2 stop **action=7** BL1,BL2 brake **action=8** BL1,BL2 cw **action=9** BL1,BL2 ccw **action=10** BL1 cw,BL2 ccw **action=11** BL1 ccw,BL2 cw bl_1 = encoder's value for BL1 bl_2 = encoder's value for BL2
void bl_2_encon(unsigned char **add**, unsigned short **enc_data**, unsigned char **action**, unsigned char **act_value1,**unsigned char **act_value2**)	bl_2_encon(**add_bl1, 2000,7,0,0**)	**add** = card address. **enc_data** = encoder's data, 0-65536. **action** = action for BL1 and BL2, (0-11). **act_value1** = value of BL1 action, 0-255(for speed, cw, ccw). **act_value2** = value of BL2 action, 0-255(for speed, cw, ccw). **action=0** none **action=1** BL2 stop **action=2** BL2 brake **action=3** BL2 CW **action=4** BL2 CCW **action=5** BL2 Speed **action=6** BL1,BL2 stop **action=7** BL1,BL2 brake **action=8** BL1,BL2 cw **action=9** BL1,BL2 ccw **action=10** BL1 cw,BL2 ccw **action=11** BL1 ccw,BL2 cw bl_1 = encoder's value for BL1 bl_2 = encoder's value for BL2

void bl_'x'_enclr(unsigned char **add**)	bl_1_enclr(**add_bl1**)
	bl_2_enclr(**add_bl1**)
		add = card address.
		Clear encoder 16 bits register at BL02 card.
		bl_1 = clear BL1's encoder value
		bl_2 = clear BL2's encoder value

unsigned short bl_'x'_enval(unsigned char **add**)	bl_1_enval(**add_bl1**)
	bl_2_enval(**add_bl1**)
		add = card address.
		Read encoder 16 bits value (0-65536) on BL02 card.
		bl_1 = read BL1's encoder value
		bl_2 = read BL2's encoder value

unsigned char
bl_'x'_runstat(unsigned char **add**)
	bl_1_runstat(**add_bl1**)
	bl_2_runstat(**add_bl1**)
		add = card address.
		Read motor status connected at BL02 card.
		Return 1, when motor run,
		Return 0, when motor stop.
		bl_1 = check BL1's status
		bl_2 = check BL2's status

unsigned char
bl_'x'_enstat(unsigned char **add**)
	bl_1_enstat(**add_bl1**)
	bl_2_enstat(**add_bl1**)
		add = card address.
		Read encoder status connected at BL02 card.
		Return 1, when encoder in process,
		Return 0, when encoder NOT in process.
		bl_1 = check BL1's encoder status
		bl_2 = check BL2's encoder status.

unsigned char
bl_'x'_spval(unsigned char **add**)
	bl_1_spval(**add_bl1**)
	bl_2_spval(**add_bl1**)
		add = card address.
		Read speed value (0-255) for motor connected at BL02 card.
		bl_1 = read speed value for BL1
		bl_2 = read speed value for BL2

*** This function is upgraded for IFC-BL02 after March 2010 batch only**
void bl_'x'_alcon(unsigned char **add**, unsigned char **on_off**, unsigned char **autoreset**)
	bl_1_alcon(**add_bl1,1,1**)
	bl_2_alcon(**add_bl1,1,0**)
		add = card address.
		
		on_off=1 alarm enable
		on_off=0 alarm disable
		
		autoreset=1 alarm autoreset (alarm MUST be enabled for this condition, on_off=1)
		autoreset=0 alarm NOT autoreset.
		
		bl_1 = set BL1's alarm configuration
		bl_2 = set BL2's alarm configuration
		
		Note: By default alarm and alarm auto reset are disabled.

*** This function is upgraded for IFC-BL02 after March 2010 batch only**
void bl_'x'_alrst(unsigned char **add**)
	bl_1_alrst(**add_bl1**)
	bl_2_alrst(**add_bl1**)
		add = card address.
		Reset alarm at BL02 card.
		bl_1 = reset BL1's alarm
		bl_2 = reset BL2's alarm

*** This function is upgraded for IFC-BL02 after March 2010 batch only** unsigned char bl_'x'_alstat(unsigned char **add**)	bl_1_alstat (**add_bl1**) bl_2_alstat (**add_bl1**)	**add** = card address. Read alarm status connected at BL02 card but alarm MUST be enabled first. Return 1, when alarm detected, Return 0, when alarm no detected. bl_1 = check BL1's alarm status bl_2 = check BL2's alarm status

Table 1 Function Prototype for BL02

This document also comes with examples of function prototype. With these examples, hopefully user will understand more about BL02 function prototype. These examples will explain on the function prototype of bl_'x'. 'x' = 1 or 2. When x = 1, the function prototype is for brushless motor 1(BL1) on IFC-BL02 card and when x = 2, the function prototype is for brushless motor 2(BL2) on IFC-BL02 card.

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

This function is used to stop the brushless motor. When user calls this function, motor will stop but not immediately. A simple code for this function prototype is:

```
bl_1_stop(add_bl1);                // stop BL1
bl_2_stop(add_bl1);                // stop BL2
```

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

This function is also used to stop the brushless motor. User recommended using 'bl_'x'_brake' function compared to 'bl_'x'_stop' function if user wants to stop brushless motor. It is because when user calls this function to stop the motor, it will stop immediately. A sample code to call this function is:

```
bl_1_brake(add_bl1);               // brake BL1
bl_2_brake(add_bl1);               // brake BL2
```

void bl_1_cw(unsigned char **add**)

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

```
bl_1_cw(add_bl1);                  // Run BL1 in clockwise
bl_2_cw(add_bl1);                  // Run BL2 in clockwise
```

void bl_'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:

```
bl_1_ccw(add_bl1);                 // Run BL1 in counter-clockwise
bl_2_ccw(add_bl1);                 // Run BL2 in counter-clockwise
```

Note1: User needs to call function 'bl_'x'_speed' to give motor speed after set the direction of the motor. The method to call function 'bl_'x'_speed' will be explained later. x = 1 is for (BL1) and x = 2 is for (BL2).
Notes2: If BL02 is use to control **Linix Brushless motor**, after set the direction user needs to **delay** awhile before give motor speed. It is to avoid the motor run in previous direction that store in Linix motor's driver. Please refer sample source code for the example.

```
void bl_'x'_speed(unsigned char add, unsigned char speed)
```

This function is used to set the motor's speed. Unsigned char data is a value 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. A sample code below is to set BL1 and BL2 speed.

```
bl_1_speed(add_bl1,speed_bl1);          // set BL1's speed

 bl_2_speed(add_bl1,speed_bl2);         // set BL2's speed
```

void bl_'x'_encon(unsigned char **add**, unsigned short **enc_data**, unsigned char **action**, unsigned char **act_value1,**unsigned char **act_value2**)

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 brushless motor will do, which are : none(0), stop(1) , brake(2) , cw(3) , ccw(4) , speed(5) , BL1BL2 stop(6), BL1BL2 brake(7), BL1BL2 cw(8), BL1BL2 ccw(9), BL1cw BL2ccw(10) and BL1ccw BL2cw(11) is executed. The 'action' will execute after pulses from encoder reach the value set in 'enc_data'. 'act_value1'and act_value2 are the speed of BL1 and BL2 respectively when execute the 'action'. 'act_value' for 'action' cw(3), ccw(4), speed(5), BL1BL2 cw(8), BL1BL2 ccw(9), BL1cw BL2ccw(10) or BL1ccw BL2cw(11) is at the range of 0 to 255 while 'act_value' for 'action' none(0), stop(1), brake(2), BL1BL2 stop(6), BL1BL2 brake is 0. A sample code below is to show how to call this function prototype.

```
bl_1_encon(add_bl1, 1000, 9, 100,100);  // Activate BL2 and change BL1's speed and
                                        // direction when BL1's encoder value = 1000.

bl_2_encon(add_bl1, 500, 7, 0,0);         // Brake BL1 and BL2 when BL2's encoder
                                          // value = 500.
```

```
void bl_'x'_enclr(unsigned char add)
```

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

```
bl_1_enclr(add_bl1);            // Clear BL1 encoder value
bl_2_enclr(add_bl1);            // Clear BL2 encoder value
```

```
unsigned short bl_'x'_enval(unsigned char add)
```

bl_’x’_enval stores the value counted by the encoder. This function is used to read the encoder value. Encoder can count up to 65,535 pulses. A simple example to call this function is:

```
cp1_dec(bl_1_enval(add_bl1),5); // Read BL1 encoder's value to be displayed
cp1_dec(bl_2_enval(add_bl1),5); //  Read BL2 encoder's value to be displayed
```

```
unsigned char bl_'x'_runstat(unsigned char add)
```

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

```
if(bl_1_runstat(add_bl1)==1)    //Turn on led1 if motor1 run
{
    led1=1;
}
if(bl_2_runstat(add_bl1)==0)    //turn off led1 if motor2 stop
{
    led1=0;
}
```

unsigned char bl_'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 in process or return '0' if encoder is NOT in process. Encoder in process means when function 'bl_x_encon()' is called, the encoder will count the pulses and compare it with the 'enc_data' before execute the 'action'. While encoder NOT in process means encoder still counting but DO NOT compare the value with 'en_data' in function 'bl_x_encon()'. User can call this function as below:

```
while(bl_1_enstat(add_bl1)==1)          // check status of BL1's encoder
{
    led5=1;                             // turn ON LED5 when BL1's encoder in process
}

while(bl_2_enstat(add_bl1)==1)          // check status of BL2's encoder
{
    led6=1;                             // turn ON LED6 when BL2's encoder in process
}
```

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

bl_'x'_speed stores speed value for BL1 and BL2. This function is used to read speed value. Maximum value for speed is 255 while minimum is 0. User can call this function as below:

```
speed_bl1=bl_1_spval(add_bl1);       // store BL1's speed value to speed_bl1
speed_bl2=bl_2_spval(add_bl1);       // store BL2's speed value to speed_bl2
```

*** This function is upgraded for IFC-BL02 after March 2010 batch only**

void bl_'x'_alcon(unsigned char **add**, unsigned char **on_off**, unsigned char **autoreset**)

This function is used to configure the alarm for both channels. 'on_off' is the alarm enable bit, 1 for alarm enable and 0 for alarm disable. If user need the alarm auto reset function, 'autoreset' should be set to 1 and make sure the alarm is enabled ('on_off'=1). Please do take note that by default alarm and alarm auto reset are DISABLED. A sample code below is to show how to call this function.

```
bl_1_alcon(add_bl1,1,1);     // BL1's alarm and alarm auto reser on
bl_2_alcon(add_bl1,1,0);     // BL2's alarm on and alarm auto reser off
```

*** This function is upgraded for IFC-BL02 after March 2010 batch only**

void bl_'x'_alrst(unsigned char **add**)

This function is used to reset the alarm by user manually. For example, if alarm was detected, user can reset it by calling this function. To call this function, user may write as below:

```
if(bl_1_alstat(add_bl1)==1)      // test for BL1's alarm status
{
    bl_1_alrst(add_bl1);         // reset BL1's alarm if return '1'
}                                // means BL1's alarm detected

if(bl_2_alstat(add_bl1)==1)      // test for BL2's alarm status
{
    bl_2_alrst(add_bl1);         // reset BL2's alarm if return '1'
}                                // means BL2's alarm detected
```

*** This function is upgraded for IFC-BL02 after March 2010 batch only**

unsigned char bl_'x'_alstat(unsigned char **add**)

This function is used to read alarm status which will return '1' if alarm detected or return '0' if no alarm detected. But make sure the alarm was enabled (using 'bl_'x'_alcon' function) before using this function. To use this function, user can write as below:

```
if(bl_1_alstat(add_bl1)==1)      // test for BL1's alarm status
{
    bl_1_alrst(add_bl1);         // reset BL1's alarm if return '1'
}                                // means BL1's alarm detected

if(bl_2_alstat(add_bl1)==1)      // test for BL2's alarm status
{
    bl_2_alrst(add_bl1);         // reset BL2's alarm if return '1'
}                                // means BL2's alarm detected
```

Note: User is reminded to add header file (iic.h and iic_bl.h) and object file (iic.o and iic_bl.o) for IFC-MB00 and IFC-BL02 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