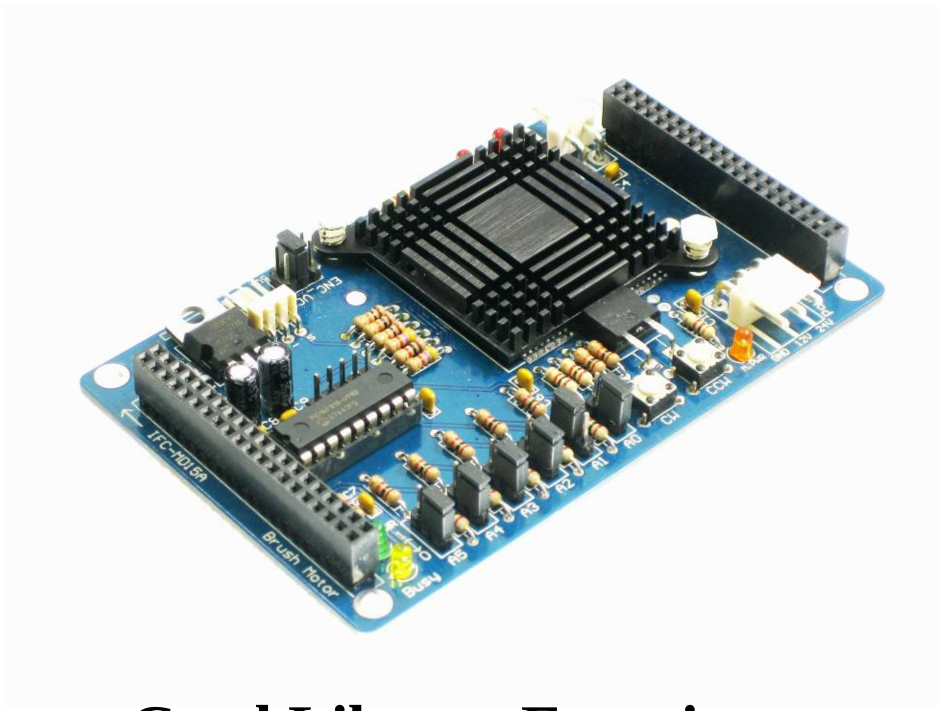
IFC-MD15A
Interface Free Controller
Brush Motor Card



Card Library Functions

V1.0

Oct 2008

IFC-MD15A Card Technical Info

This document explains the function prototype for MD15A needed in controlling a brush motor. The function prototype will be called in main program for MB00 in order to control/communicate with MD15A. User can also find the explanation of function prototype in its header file, "iic_md.h". Table 1 shows function prototype for MD15A, the example card address used in the function prototype is 'add_md1'.

Function Prototype	Remark	Parameter Description
void md_stop(unsigned char **add**)	md_stop(**add_md1**)	**add** = card address. Stop motor connected at MD15A card.
void md_brake(unsigned char **add**)	md_brake(**add_md1**)	**add** = card address. Brake motor connected at MD15A card.
void md_cw(unsigned char **add**)	md_cw(**add_md1**)	**add** = card address. Set direction of motor connected at MD15A card to Clockwise (cw).
void md_ccw(unsigned char **add**)	md_ccw(**add_md1**)	**add** = card address. Set direction of motor connected at MD15A card to Counter-Clockwise (ccw).
void md_speed(unsigned char **add**, unsigned char **speed**)	md_speed(**add_md1, 100**)	**add** = card address. **speed** = speed value,0-255. Set speed of motor connected at MD15A card.
void md_encon(unsigned char **add**, unsigned short **enc_data**, unsigned char **action**, unsigned char **act_value**)	md_encon(**add_md1, 50, 2,0**)	**add** = card address. **enc_data** = encoder's data, 0-65536. **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
void md_enclr(unsigned char **add**)	md_enclr(**add_md1**)	**add** = card address. Clear encoder 16 bits register at MD15A card.

void md_alcon(unsigned char **add**, unsigned char **autoreset,** unsigned short **response**)	md_alcon(**add_md1,1,0**)	**add** = card address. **autoreset** = alarm configuration. **1** = alarm autoreset, **0** = alarm NOT autoreset. **response** = acceleration after autoresetting, value **1 – 1023**. **0** = default (by default response = 820)
void md_alrst(unsigned char **add**)	md_alrst(**add_md1**)	**add** = card address. Manually reset alarm for motor connected at MD15A card.
unsigned short md_enval(unsigned char **add**)	md_enval(**add_md1**)	**add** = card address. Read encoder 16 bits value (0-65536) on MD15A card.
Unsigned char md_runstat(unsigned char **add**)	md_runstat(**add_md1**)	**add** = card address. Read motor status connected at MD15A card. Return 1, when motor run, Return 0, when motor stop.
unsigned char md_enstat(unsigned char **add**)	md_enstat(**add_md1**)	**add** = card address. Read encoder status connected at MD15A card. Return 1, when encoder in process, Return 0, when encoder NOT in process.
unsigned char md_spval(unsigned char **add**)	md_spval(**add_md1**)	**add** = card address. Read speed value (0-255) for motor connected at MD15A card.
unsigned char md_alstat(unsigned char **add**)	md_alstat(**add_md1**)	**add** = card address. Read alarm status for motor connected at MD15A card. Return 0, when NO alarm notification Return 1, when has alarm notification Return 2, over temperature (thermal shutdown)

Table 1 Function Prototype for MD15A

This document also comes with example of function prototype. With this example, hopefully user will understand more about MD15A function prototype.

void md_stop(unsigned char **add**)

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

```
md_stop(add_md1);              // stop motor
```

void md_brake(unsigned char **add**)

This function is also used to stop the brush motor. User is recommended to use 'md_brake' function compared to 'md_stop' function if user wants to stop the brush motor. It is because when user calls this function to stop the brush motor, it will stop immediately. A sample code to call this function is:

```
md_brake(add_md1);             // brake motor
```

void md_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:

```
md_cw(add_md1);                // Run motor in clockwise
```

void md_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:

```
md_ccw(add_md1);               // Run motor in counter-clockwise
```

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

void md_speed(unsigned char **add**, unsigned char **speed**)

This function is used to set the motor's speed. Unsigned char speed 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:

```
md_speed(add_md1,150);      // motor's speed = 150
```

void md_encon(unsigned char **add**, unsigned short **enc_data**, unsigned char **action**, unsigned char **act_value**)

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 execute after pulses from encoder reach the value set in 'enc_data'. 'act_value' is the speed of motor when 'action' cw (3), ccw (4) and speed (5) is executed. The range for 'act_value' is 0 to 255. For action 0-2, user needs to put a value from 0 to 255 for 'act_value' 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 50000 ('enc_data') pulses.

```
md_encon(add_md1, 50000, 4, 100);    //when encoder value = 50000.
                                     //change motor direction to
                                     //counter clockwise with speed 100
```

void md_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. To call this function, user may write as below:

```
md_enclr(add_md1);   //clear encoder
```

Caution: The motor driver on MD15A will limit current to motor, but in some cases if the **direction changes rapidly**, it will cause the motor to **draw very high current** which will **activate the alarm**. The **motor** will **stop** when the alarm is activated. It that case, user needs to **reset alarm** in motor driver either manually (use function 'md_alrst()') or enable autoreset in function 'md_alcon()' (set 'autoreset' to '1' in function 'md_alcon()'). It is to make sure the motor will run smoothly after direction changed.

void md_alcon(unsigned char **add**, unsigned char **autoreset,** unsigned short **response**)

This function is used to set alarm configuration at brush motor, whether it is auto reset or manually reset. If user wants alarm to set as autoreset, then set ‘autoreset’ to ‘1’. Otherwise, user can set ‘autoreset’ to ‘0’ if user wants to disable alarm autoreset function and set it as manually reset. User can manually reset the alarm by calling function ‘md_alrst( )’, the method to call function ‘md_alrst’ will be explained later. ‘Response’ equivalent to the acceleration of the motor after auto resetting and the value can be set from 1-1023 (depend on the motor). It is because this motor driver can’t drive some high ampere brush motors in high acceleration (high current protected). When user set response to a big value, motor will fast go to high speed compare a small value. By default the response is 820. To use this function, user can call it as below:

```
md_alcon(add_md1,1,542);  // alarm is set to autoreset
                          // motor response is 542
```

void md_alrst(unsigned char **add**)

This function is used to manually reset the alarm. User needs to call this function if user had disable alarm autoreset. User can call this function like this:

```
md_alrst(add_md1);  //Manually reset alarm for motor
```

unsigned short md_enval(unsigned char **add**)

md_enval stores the value counted by the encoder. This function is used to read the encoder value. The encoder can count up to 65,535 pulses. A simple example to call this function is:

```
md_enval(add_md1);      // read encoder value
```

unsigned char md_runstat(unsigned char **add**)

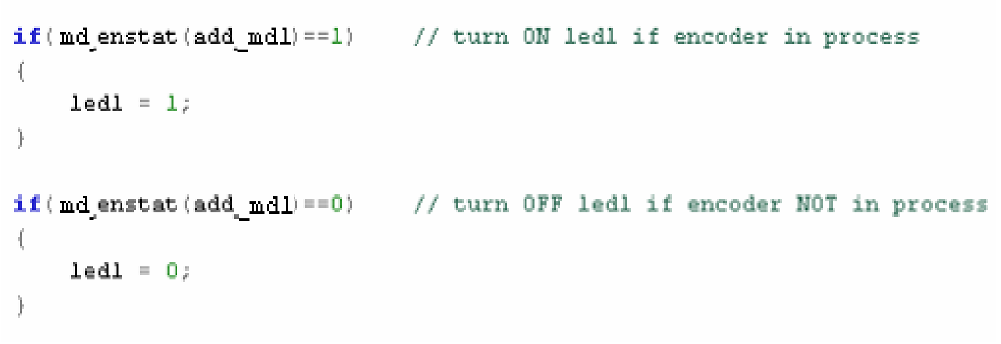
This function is used to read motor status which will return ‘1’ if brush motor is set to run or return ‘0’ if brush motor is set to stop in the program. This function is useful when user wants to check the motor’s status set by the program. To use this function, user can write as below:

```
if(md_runstat(add_md1)==1)   // turn ON led1 if brush motor Run
{
    led1 = 1;
}

if(md_runstat(add_md1)==0)   // turn OFF led1 if brush motor stop
{
    led1 = 0;
}
```

unsigned char md_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 ‘md_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 ‘md_encon()’ not being called). User can call this function as below:

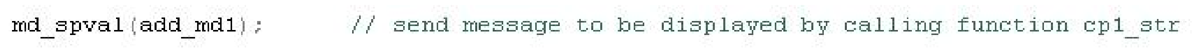


```
if(md_enstat(add_md1)==1)     // turn ON led1 if encoder in process
{
    led1 = 1;
}

if(md_enstat(add_md1)==0)     // turn OFF led1 if encoder NOT in process
{
    led1 = 0;
}
```

unsigned char md_spval(unsigned char **add**)

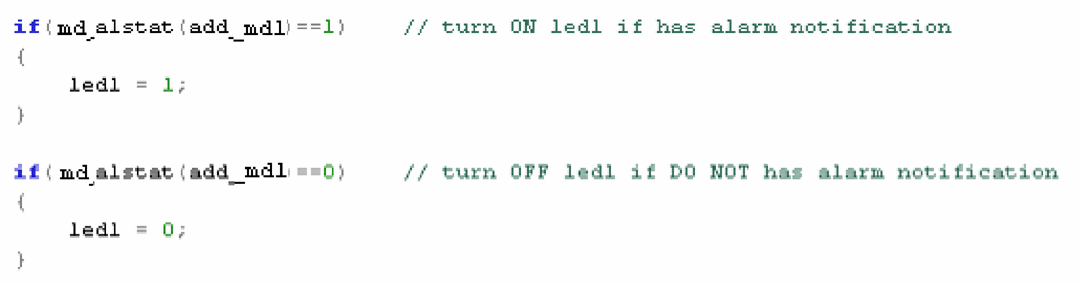
This function is used to read the speed value. Maximum value for speed is 255 while minimum is 0. User can call this function like:



```
md_spval(add_md1);      // send message to be displayed by calling function cp1_str
```

unsigned char md_alstat(unsigned char **add**)

This function is used to read alarm status. The status is ‘1’ if it has alarm notification and ‘0’ if it does NOT have alarm notification. A sample code is shown as below:



```
if(md_alstat(add_md1)==1)     // turn ON led1 if has alarm notification
{
    led1 = 1;
}

if(md_alstat(add_md1)==0)     // turn OFF led1 if DO NOT has alarm notification
{
    led1 = 0;
}
```

Note: User is reminded to add header file (iic.h and iic_md.h) and object file (iic.o and iic_md.o) for IFC-MB00 and IFC-MD15A each time user opens a new project for IFC. User also needs to include card h file at the beginning of the program. Please refer sample source code for the example to include card h 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