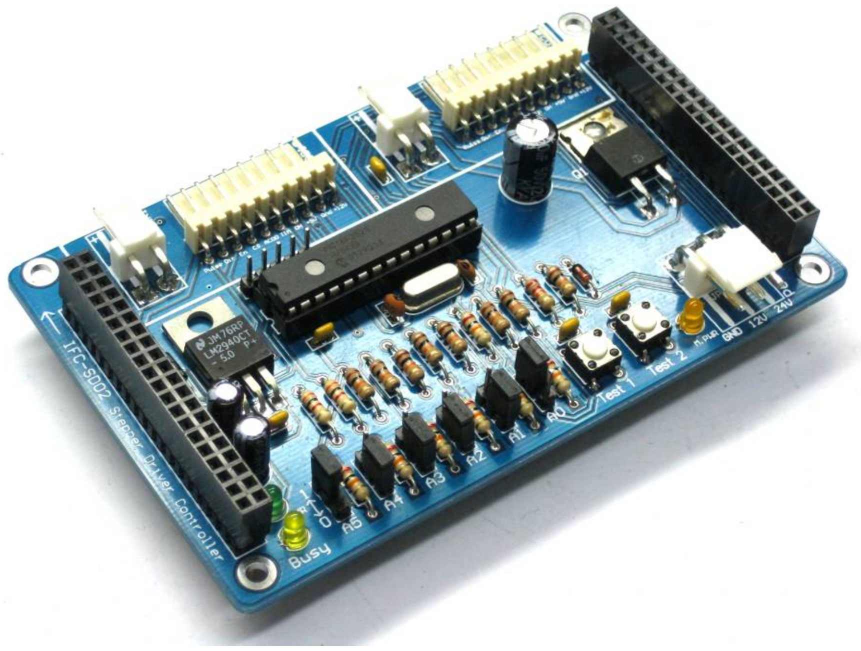
IFC-SI02 Interface Free Controller Dual Stepper Indexer Card



Card Library Functions

V1.0

Nov 2009

IFC-SI02 Card Technical Info

This document explains the function prototype for IFC-SI02 used in controlling 2 stepper drivers. The function prototype will be called in main program for MB00 in order to control/communicate with IFC-SI02. User can also find the explanation of function prototype in its header file, "iic_si.h". Table 1 shows the function prototype for IFC-SI02, the example card's address used in the function prototype is 'add_si'. Please make sure the address on IFC – SI02 is compatible to program. This technical info will explain on the function prototype of si_'x'('x' = 1 or 2). When x = 1, the function prototype is for channel 1 (SI1) on IFC-SI02 card and when x = 2, the function prototype is for channel 2 (SI2) on IFC-SI02.

Function Prototype	Remark	Parameter Description

void si_'x'_stop(unsigned char **add**)	si_1_stop(**add_si**)
	si_2_stop(**add_si**)
		add = card address.
		
		Stop and release motor connected at IFC-SI02 card.
		
		si_1 = stop channel 1
		si_2 = stop channel 2

void si_'x'_brake(unsigned char **add**)	si_1_brake(**add_si**)
	si_2_brake(**add_si**)
		add = card address.
		
		Brake motor connected at IFC-SI02 card.
		
		si_1 = brake channel 1
		si_2 = brake channel 2

void si_'x'_cw(unsigned char **add**)	si_1_cw(**add_si**)
	si_2_cw(**add_si**)
		add = card address.
		
		Set direction of motor connected at IFC-SI02 card to Clockwise (cw).
		
		si_1 = set channel 1 direction
		si_2 = set channel 2 direction

void si_'x'_ccw(unsigned char **add**)	si_1_ccw(**add_si**)
	si_2_ccw(**add_si**)
		add = card address.
		
		Set direction of motor connected at IFC-SI02 card to Counter-Clockwise (ccw).
		
		si_1 = set channel 1 direction
		si_2 = set channel 2 direction

void si_'x'_cson(unsigned char **add**)	si_1_cson(**add_si**)
	si_2_cson(**add_si**)
		add = card address.
		
		Set micro stepping of stepper motor driver **ON**.
		
		si_1=set channel 1 microstep on.
		si_2=set channel 2 microstep on.

void si_'x'_csoff(unsigned char **add**)	si_1_csoff(**add_si**
	si_2_csoff(**add_si**)
		add = card address.
		
		Set micro stepping of stepper motor driver **OFF**.
		si_1=set channel 1 microstep off.
		si_2=set channel 2 microstep off.

void si_'x'_acdon(unsigned char **add**)	si_1_acdon(**add_si**)
	si_2_acdon(**add_si**)
		add = card address.
		
		Set automatic current cut back of stepper motor driver **ON**.
		
		si_1=set channel 1 automatic current cut back on
		si_2= set channel 2 automatic current cut back on

void si_'x'_acdoff(unsigned char **add**)	si_1_acdoff(**add_si**)
	si_2_acdoff(**add_si**)
		add = card address.
		
		Set automatic current cut back of stepper motor driver **OFF**.
		
		si_1=set channel 1 automatic current cut back off
		si_2= set channel 2 automatic current cut back off

void si_'x'_setacc(unsigned char **add**, unsigned char **acc**)	si_1_setacc(**add_si**, **150**)
	si_2_setacc(**add_si**, **255**)
		add = card address
		acc = acceleration value, 1-255.
		
		Set the acceleration value for all the functions that involve acceleration.
		
		si_1 = set channel 1 acceleration value
		si_2 = set channel 2 acceleration value

void si_'x'_speed(unsigned char **add**, unsigned char **speed**)	si_1_speed (**add_si**, 255) si_2_speed(**add_si**, 50)	**add** = card address **speed** = final speed value, 0-255. Set the channel's motor to start accelerate or decelerate at the previous preset acceleration value. Motors will start accelerate or decelerate until the desired speed. si_1 = set channel 1 to accelerate si_2 = set channel 2 to accelerate

void si_maxfreq(unsigned char **add**, unsigned char **maxfreq**)	si_maxfreq(**add_si**, 150)	**add** = card address **maxfreq** = maximum pulse frequency generated when speed is at maximum (255) for both channel. 25-200(2.5kHz-20kHz).

void si_'x'_encon(unsigned char **add**, unsigned long **enc_data**, unsigned char **action**, unsigned char **act_value1**, unsigned char **act_value2**);
 	
there is only one act value need in current function
	si_1_encon(**add_si**, **500**, **2**, **0**, **0**)
	si_2_encon(**add_si**, **500**, **4**, **100**, **200**)
		add = card address.
		enc_data = encoder's data, 0-16777215.
		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
		action=6 SI1,SI2 stop
		action=7 SI1,SI2 brake
		action=8 SI1,SI2 cw
		action=9 SI1,SI2 ccw
		action=10 SI1 cw,SI2 ccw
		action=11 SI1 ccw,SI2 cw
		si_1 = encoder's value for SI1
		si_2 = encoder's value for SI2
		
		si_1 = set encoder value and action for motor 1.
		si_2 = set encoder value and action for motor 2.

void si_'x'_econclr(unsigned char **add**)	si_1_econclr(**add_si**)
	si_2_econclr(**add_si**)
		add = card address.
		
		Clear encoder function.
		
		si_1 = clear encoder 1 function.
		si_2 = clear encoder 2 function.

void si_'x'_enclr(unsigned char **add**)	si_1_ enclr((**add_si**)
	si_2_ enclr ((**add_si**)
		add = card address.
		
		Clear encoder, 24 bits register at IFC-SI02 card.
		
		si_1 = clear encoder channel 1
		si_2 = clear encoder channel 2

unsigned long si_'x'_enval(unsigned char **add**)	si_1_enval(**add_si**)
	si_2_enval(**add_si**)
		add = card address.
		
		Read encoder, 24 bits value (0-16777216) on IFC-SI02 card.
		
		si_1 = read encoder 1
		si_2 = read encoder 2

unsigned char si_'x'_runstat(unsigned char **add**)	si_1_runstat(**add_si**)
	si_2_runstat(**add_si**)
		add = card address.
		
		Read motor status connected at IFC-SI02 card.
		
		Return 1, when motor run,
		Return 0, when motor stop.
		
		si_1 = check SI1's status
		si_2 = check SI2's status

unsigned char si_'x'_spval(unsigned char **add**)	si_1_spval(**add_si**)
	si_2_spval(**add_si**)
		add = card address.
		
		Read speed value (0-255) of the channels of IFC-SI02 card.
		
		si_1 = read speed value for channel 1.
		si_2 = read speed value for channel 2.

unsigned char sd_'x'_OH(unsigned char **add**)	si_1_OH(**add_si**)
	si_2_OH(**add_si**)
		add = card address.
		
		Read the overheat signal (if available) from the stepper driver card.
		
		si_1 = read overheat signal of channel 1
		si_2 = read overheat signal of channel 2

unsigned char si_'x'_dirstat(unsigned char **add**)	si_1_dirstat(**add_si**)
	si_2_dirstat(**add_si**)
		add = card address.
		
		Read the motor direction(cw/ccw) of the channels. Return 0 (cw) or 1(ccw) depending on direction
		
		si_1 = read direction of channel 1
		si_2 = read direction of channel 2

Table 1 Function Prototype for IFC-SI02

This document also shows the examples of function prototype usage. With these examples, hopefully user can get more in-depth understanding of IFC-SI02's function prototype. These examples will explain on the function prototype of si_'x'. 'x' = 1 or 2. When x = 1, the function prototype is for channel 1(SI1) on IFC-SI02 card and when x = 2, the function prototype is for channel 2 (SI2) on IFC-SI02 card.

```
void si_'x'_stop(unsigned char add)
```

This function is used to off and release the stepper motor of each channel. When user calls this function, motor will stop or release (Enable signal to the stepper driver card is grounded). User can call this function as below:

```
si_1_stop(add_si);          //release channel 1 motor
si_2_stop(add_si);          //release channel 2 motor
```

```
void si_'x'_brake(unsigned char add)
```

This function is also used to enable the stepper motor; the motor will in brake condition. User is recommended to use 'si_'x'_brake' function compared to 'si_'x'_stop' function if user wants to stop stepper motor immediately. A sample code to call this function is:

```
si_1_brake(add_si);          //brake channel 1 motor
si_2_brake(add_si);          //brake channel 2 motor
```

```
void si_'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). User is advised to decelerate the motor before change the direction to avoid step-slipping. Below is a simple example for user to call this function:

```
si_1_cw(add_si)          //change direction of channel 1 to clockwise
si_2_cw(add_si)          // change direction of channel 2 to clockwise
```

```
void si_'x'_ccw(unsigned char add)
```

This function is used to set the direction of the motor to counter clockwise (ccw). User is advised to decelerate the motor before change the direction to avoid step-slipping. Below is a simple example for user to call this function:

```
si_1_ccw(add_si);          //change direction of channel 1 to counter clockwise
si_2_ccw(add_si);          // change direction of channel 2 to counter clockwise
```

Note 1: User needs to call function to run or to accelerate motor after changing the direction.
Note 2: There are some different signal timing requirements for different types of stepper motor driver when changing direction. Please refer respective usual manual for the knowledge to minimize the chance of signal lost.

void si_'x'_cson(unsigned char **add**)
void si_'x'_csoff(unsigned char **add**)

These functions are to set the microstepping (if available) of the stepper driver card. si_'x'_cson will set the CS pin high while si_'x'_csoff will set the CS pin low.

```
si_1 _cson (add_si);        //on channel 1 microstepping
si_2_csoff (add_si);        //off channel 2 microstepping
```

void si_'x'_acdon(unsigned char **add**)
void si_'x'_acdoff(unsigned char **add**)

These functions are to set the automatic current cut back release (if available) of the stepper driver. si_'x'_acdon will set the automatic current cut back ON, while si_'x'_acdoff will set OFF the automatic current cut back.

```
si_1 _acdon (add_si);       //on channel 1 automatic current cut back
si_2_acdoff (add_si);       //off channel 2 automatic current cut back
```

void si_'x'_setacc(unsigned char **add**, unsigned char **acc**)

This function is used to set the acceleration value of each channel. The value of acceleration, 'acc' is **1-255**. This function need to be called prior to calling any other functions that involve acceleration (si_'x'_accelerate and si_'x'_encon)

```
si_1 _setacc (add_si, 255);     //set channel 1 acceleration value to be 255 (maximum).
si_2_setacc(add_si, 50);        //set channel 2 acceleration value to be 50.
```

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

This function is to set the channels to start accelerate or decelerate at the acceleration value set by si_'x'_setacc function to a desired speed value (**speed).**

```
si_1_speed (add_si, 255);        //set channel 1 accelerate to the speed of 255.
si_2_speed (add_si,100);         //set channel 2 accelerate or decelerate to speed of 100.
```

```
void si_maxfreq(unsigned char add, unsigned char maxfreq)
```

This function set the maximum frequency, **maxfreq** of both channels. This maximum frequency is also representing the pulse frequency of the maximum speed in each channel. The frequency range is from 25-200 (25 is 2.5kHz and 200 is 20kHz). The maximum frequency will be defaulted to 5kHz if user do not set it. User is advised to set the maximum frequency at the beginning of the program.

Selectable maximum frequency are 2.5kHz, 3kHz, 4kHz, 5kHz, 6kHz, 7kHz, 8kHz, 9kHz, 10kHz, 11kHz, 12kHz, 13kHz, 14kHz, 15kHz, 16kHz, 17kHz, 18kHz, 19kHz, 20kHz. Maximum frequency will directly affect the resolution of acceleration; higher maximum frequency will give lower acceleration resolution.

```
si_maxfreq(add_si, 50);          //the maximum frequency is set to 5kHz.
```

Note: There are some frequency tolerances depending on what is the maximum frequency set by the user.

```
void si_'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 16777215. 'action' is what the stepper motor will do, which are : none (0), release (1), brake (2), cw (3), ccw (4), speed (5), SI1SI2 release(6), SI1SI2 brake(7), SI1SI2 cw(8), SI1SI2 ccw(9), SI1cw SI2ccw(10) and SI1ccw SI2cw(11). The 'action' will be executed after pulses from encoder reach the value set in 'enc_data'. 'act_value1'and act_value2 are the speed of SI1 and SI2 respectively when execute the 'action'. 'act_value' for 'action' cw(3), ccw(4), speed(5), SI1SI2 cw(8), SI1SI2 ccw(9), SI1cw SI2ccw(10) or SI1ccw SI2cw(11) is at the range of 0 to 255 while 'act_value' for 'action' none(0), stop(1), brake(2), SI1SI2 stop(6), SI1SI2 brake is 0. For the convenient use of this encoder function, user can call up to 8 encoder functions at once to do different jobs at different encoder values. These 8 encoder functions will be stacked and will activate once the respective encoder value reached. Calling more than 8 encoder functions at once will cause the excessive one to be ignored. However, if one or more previous called function had done its process, user may call new encoder function to replace in.

Encoder functions are stacked:

1st enconder function
2nd enconder function
3rd enconder function
4th enconder function
5th enconder function
6th enconder function
7th enconder function
8th enconder function

9th encoder function
(will be ignored if no more space in stack but if previous function had done, it will replace it.)

A sample code below is to show how to call this function prototype.

```
si_1_encon(add_si,80000, 5, 1,0);

si_1_encon(add_si,196990, 2, 1,0);

// calling the above functions at once will cause them to be stacked as tasks need to be done when
                                     the encoder value reached
// channel one will start decelerate at encoder value of 80000 until speed value 1, and brake at
                                     encoder value of 19690.
```

Note: User may need to set the acceleration before calling the encoder function.

```
void si_'x'_econclr(unsigned char add)
```

This function is used to clear the previously called encoder functions. **All** encoder functions which were called by si_'x'_encon( ) will be cleared at once.

```
si_1_enconclr(add_si);          // clear channel 1 encoder functionS.

si_2_enconclr(add_si);          // clear channel 2 encoder functionS
```

```
void si_'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:

```
si_1_enclr(add_si);          //clear channel 1 encoder
si_2_enclr(add_si);          //clear channel 2 encoder
```

```
unsigned long si_'x'_enval(unsigned char add)
```

This function is used to read the encoder value stored each channel. The function will return 24 bits encoder data, but is store in 32 bits variable. User may call this function to get the encoder value to be displayed on LCD of IFC-CP04. The encoder can count up to 16777215 pulses. Examples to call this function are:

```
si_1_enval(add_si);          //retrieve channel 1 encoder value
si_2_enval(add_si);          //retrieve channel 2 encoder value
```

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

This function is used to read motor status which will return '1' if channel is set to run
or return '0' if channel is set to stop or brake 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 (si_1_runstatl(add_si)==1)          //Turn on led1 if motor1 run
{
    led1=1;
}
if (si_2_runstatl(add_si)==0)          //Turn off led1 if motor2 stop

{
    led1=0;
}
```

```
unsigned char si_'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(si_1_spval(add_si), 3);          //display channel 1 speed on cp1
cp1_dec(si_2_spval(add_si), 3);          //display channel 2 speed on cp1
```

```
unsigned char si_'x'_OH(unsigned char add)
```

This function is to read the overheat signal (if available) from the stepper driver card.

```
si_1_OH(add_si);          //channel 1 overheat signal
si_2_OH(add_si);          //channel 2 overheat signal
```

unsigned char si_'x'_dirstat(unsigned char **add**)

This function is to read the direction of the motor of each channel respectively.

```
si_1_dirstat(add_si);          //channel 1 direction
si_2_dirstat(add_si);          //channel 2 direction
```

Note: User needs to add header file (iic.h and iic_si.h) and object file (iic.o and iic_si.o) for IFC-MB00 and IFC-SI02 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