Software Operational Manual

SMC6480G, G-Code, Network Contyrol

www.leadshine.com

SWMN-SMC6-R20120529

Leadshine reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Leadshine does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights of others.

Leadshine's general policy does not recommend the use of its products in life support or aircraft applications wherein a failure or malfunction of the product may directly threaten life or injury. According to Leadshine's terms and conditions of sales, the user of Leadshine's products in life support or aircraft applications assumes all risks of such use and indemnifies Leadshine against all damages.

Change Log

Revision Date	Changes	Version
2012-05-29	Origin Create	SWMN-SMC6-R20120529

Safety Items

Notice

Read this manual carefully before trying to install the motion controller into your system. The person who setups the controller should have a better understanding on electronics and mechanics. Contact Leadshine technical guys when you have questions on this document.

Warning

Before running execute motion program, make sure the axes will not impact anything. It is recommended to uncouple the motor from the load before you are familiar with Leashine motion controller. Otherwise, unexpected damage to the machine may occur.

Warning

Ensure that the power supply voltage dose not exceed the controller's input range. Double check the connections and make sure the power lead polarity is correct.

Table of Contents

Chapter 1 Introduction

1.1 Overview

Leadshine SMC6480 is 10/100M Ethernet-based general purpose motion controller. It provides 1 to 4 axes motion control to stepper/servo motors for various operations. It can work in standalone mode without PC or work as a slaver in an Ethernet network. The controller number in the network has no limitation.

Leadshine SMC6480 supports many programming language including Leadshine BASIC, Leadshine G-Code, Visual Basic and Visual C++. When it woks in standalone mode without PC, you can program in Leadshine BASIC or Leadshine G-code. The programming tool can be any text editor in the PC. Leadshine provides Motion6480 for you to download the program to SMC6480. If you need to control it on-line in a network, you can develop the motion program in Visual Basic and Visual C++. Leadshine provides you corresponding driver, DLL and liabrary.

This manual will illustrate how to use the Motion6480 to download/upload configuration data and G code motion program. It also introduces the Network control functions in Visual Basic and Visual C++. There is another manual for how to edit, compile and download the Leadshine BASIC.

1.2 Control Diagram

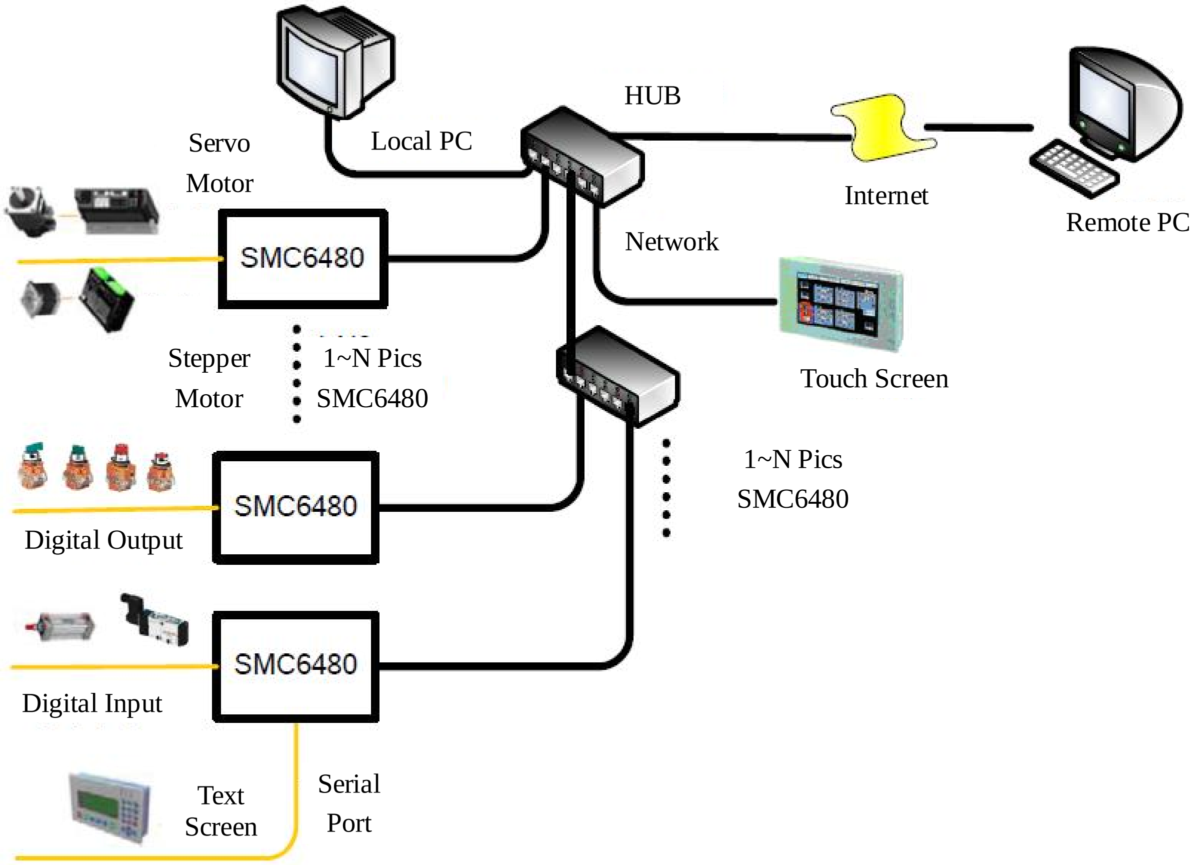


Figure 1-1: SMC6480 Connection Diagram

Chapter 2 Configuration Software Motion6480

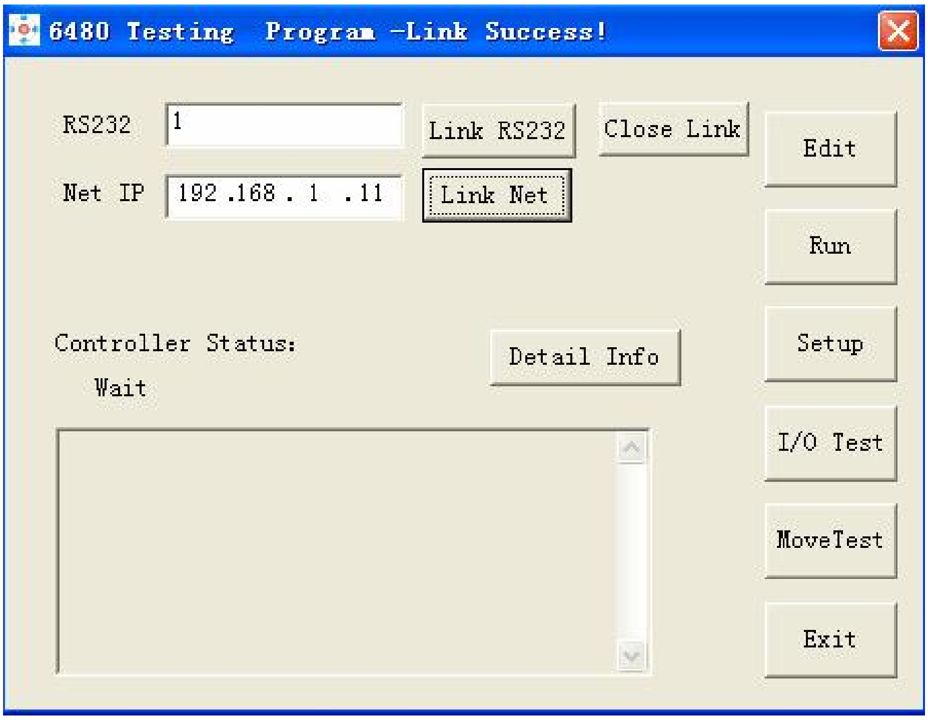
Leadshine Motion6480 is completely free configuration/programming software to all users. It helps you to understand the software and hardware functionality of SMC6480. You can make it perform all kind of motions and evaluate all features easily in this software. Motion6480 has mainly five function modules: Edit Program, Run Program, Parameters Setting, I/O Status and Motion Test You can perform basic operations such as simple point-point motion, checking I/O status, etc.

After you connect SMC6480 and its peripheral equipment, copy Motion6480 from Leadshine CD or download it from Leadshine website to your PC then run it.

For SMC6480's basic function and connections, please refer to its hard installation manual.

2.1 Main Window

The main window appears when you start Motion6480, as shown in figure 2-1. Make sure the controller has been connected to the PC's serial port or Ethernet port before clicking the corresponding "Link" button. If the communication is through Ethernet connection, the controller's IP address should be in the same sub-net with the PC's IP address.



6480 Testing Program -Link Success!

RS232: 1

Net IP: 192 .168 . 1 .11

Controller Status:
Wait

Figure 2-1: Main Window of Motion6480

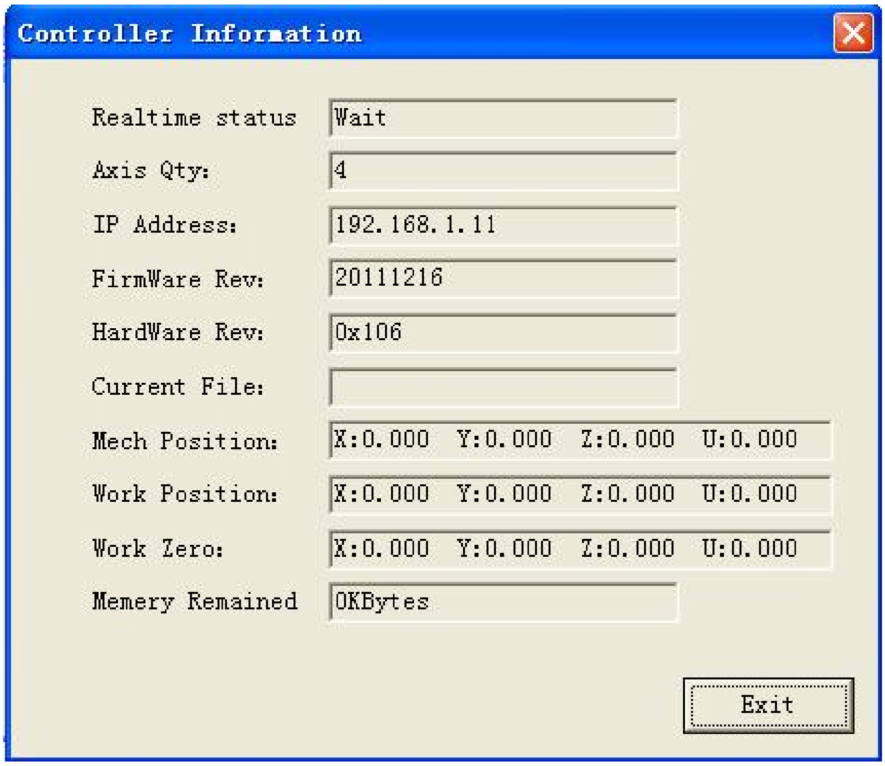
Notice

The default IP address of SMC6480 is 192.168.1.11.

Warning

It is recommended to connect the controller to the PC's Ethernet for the configuration.

If connection succeeded, the title of the main window will display "Link Success!". User can click "Detail Info" to view the controller's detailed information, shown in figure 2-2.


Controller Information

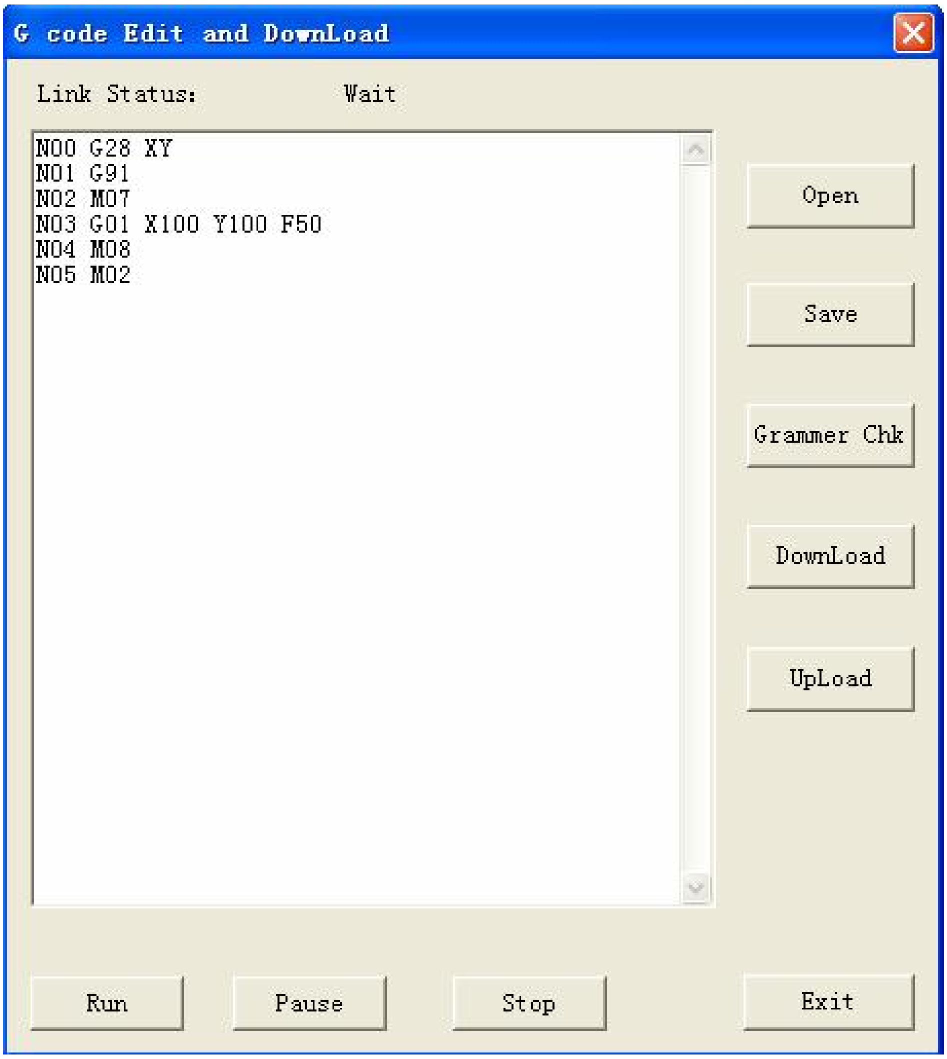
Realtime status: Wait
Axis Qty: 4
IP Address: 192.168.1.11
FirmWare Rev: 20111216
HardWare Rev: 0x106
Current File:
Mech Position: X:0.000 Y:0.000 Z:0.000 U:0.000
Work Position: X:0.000 Y:0.000 Z:0.000 U:0.000
Work Zero: X:0.000 Y:0.000 Z:0.000 U:0.000
Memery Remained: 0KBytes

Figure 2-2: Detail Information of Motion6480

2.2 Editing Program

Click "Edit" button to open the "G code edit and download" window, as shown in figure 2-3. You can type in the program and download it to the controller. The illustrations to the other button in this window are shown as follows.

G code Edit and Download Window	
Button	**Description**
Open	Open a G code motion program file in PC.
Save	Save G code motion program file to PC.
Grammer chk	Check the syntax of the current G code motion program
Download	Download G code motion program file to controller
Upload	Upload G code motion program file to Motion6480.
Run	Run G code motion program
Pause	Pause G code motion program
Stop	Stop G code motion program



```
N00 G28 XY
N01 G91
N02 M07
N03 G01 X100 Y100 F50
N04 M08
N05 M02
```

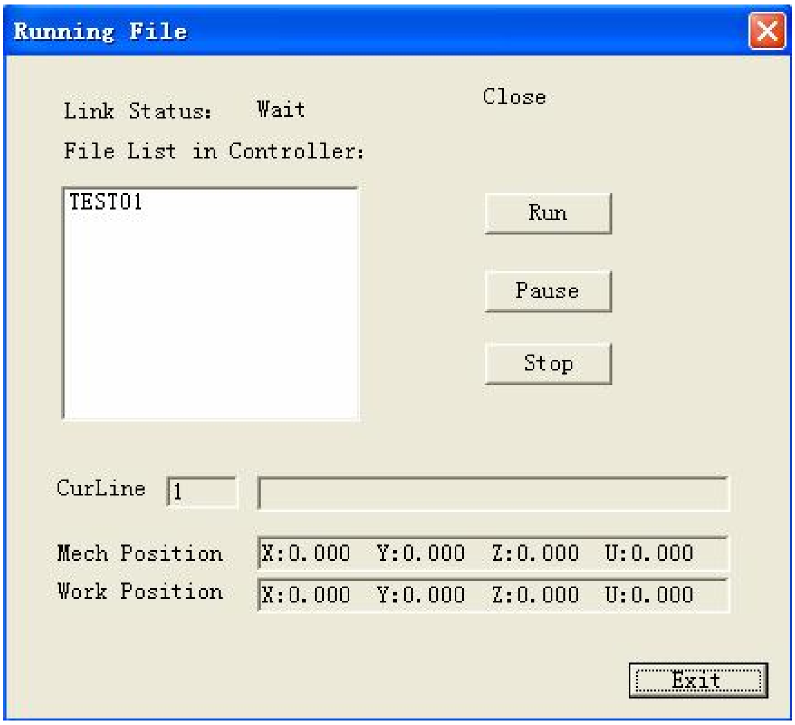
Figure 2-3: G code Edit and Download Window

2.3 Run Program

In the main window, click the “Run” button to open the “Running File” window, shown in figure 2-4. User can select a program in the left list to run. If the program stops, the up-right corner will display the status.

2.4 Setup

In the main window, click “Setup” button to open the “Setup’ window, shown in figure 2-5. User can upload, modify, download setup file in this window. Please see Chapter 3 for more information of setup file.



Running File

Link Status: Wait

File List in Controller:

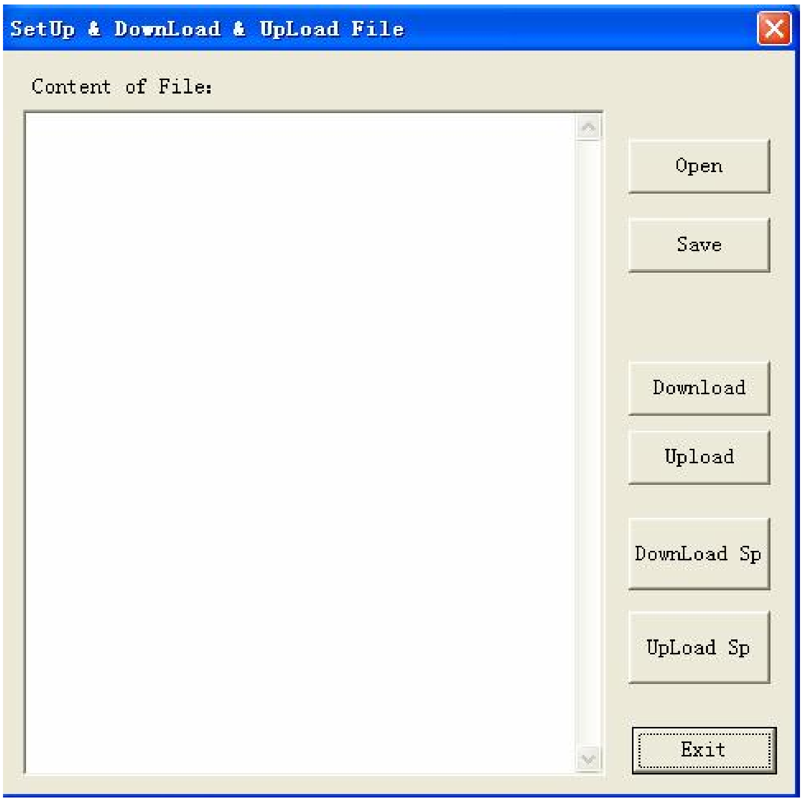
TEST01

CurLine 1

Mech Position X:0.000 Y:0.000 Z:0.000 U:0.000

Work Position X:0.000 Y:0.000 Z:0.000 U:0.000

Figure 2-4: Run G Code Motion Program Window



SetUp & DownLoad & UpLoad File

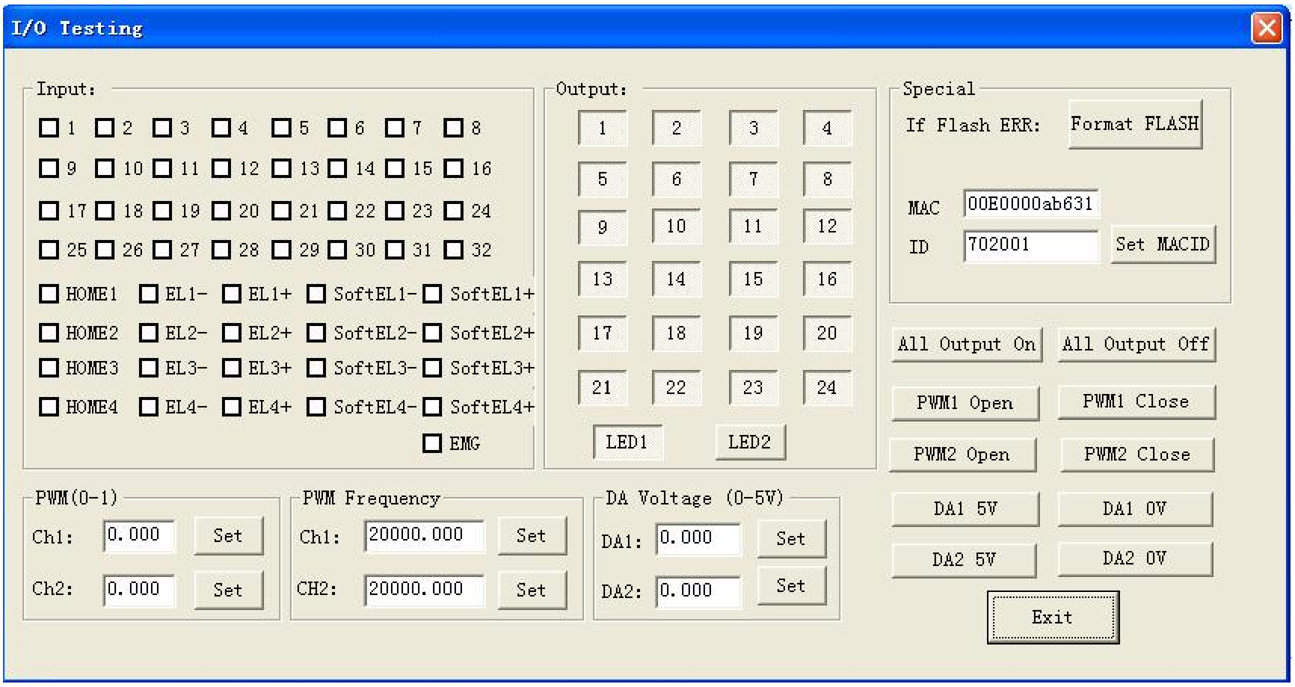
Content of File:

Figure 2-5: Setup (Download or Upload Setting) Window

2.5 I/O Test

Click the “I/O Test” button to open the “I/O Testing” window, shown in figure 2-6.

I/O Testing Window	
Region	**Description**
Input	Display digital input status. If digital input is on, the corresponding radio button will be checked.
Output	User can click the corresponding button to switch on or switch off the digital output.
PWM Output	Specifies the PWM duty-cycle (0-100%) and output frequency (Up to 1MHz)
DA Output	Specifies the DA output voltage(0.00-5.00V)



MAC 00E0000ab631

ID 702001

PWM(0-1)

Ch1: 0.000

Ch2: 0.000

PWM Frequency

Ch1: 20000.000

CH2: 20000.000

DA Voltage (0-5V)

DA1: 0.000

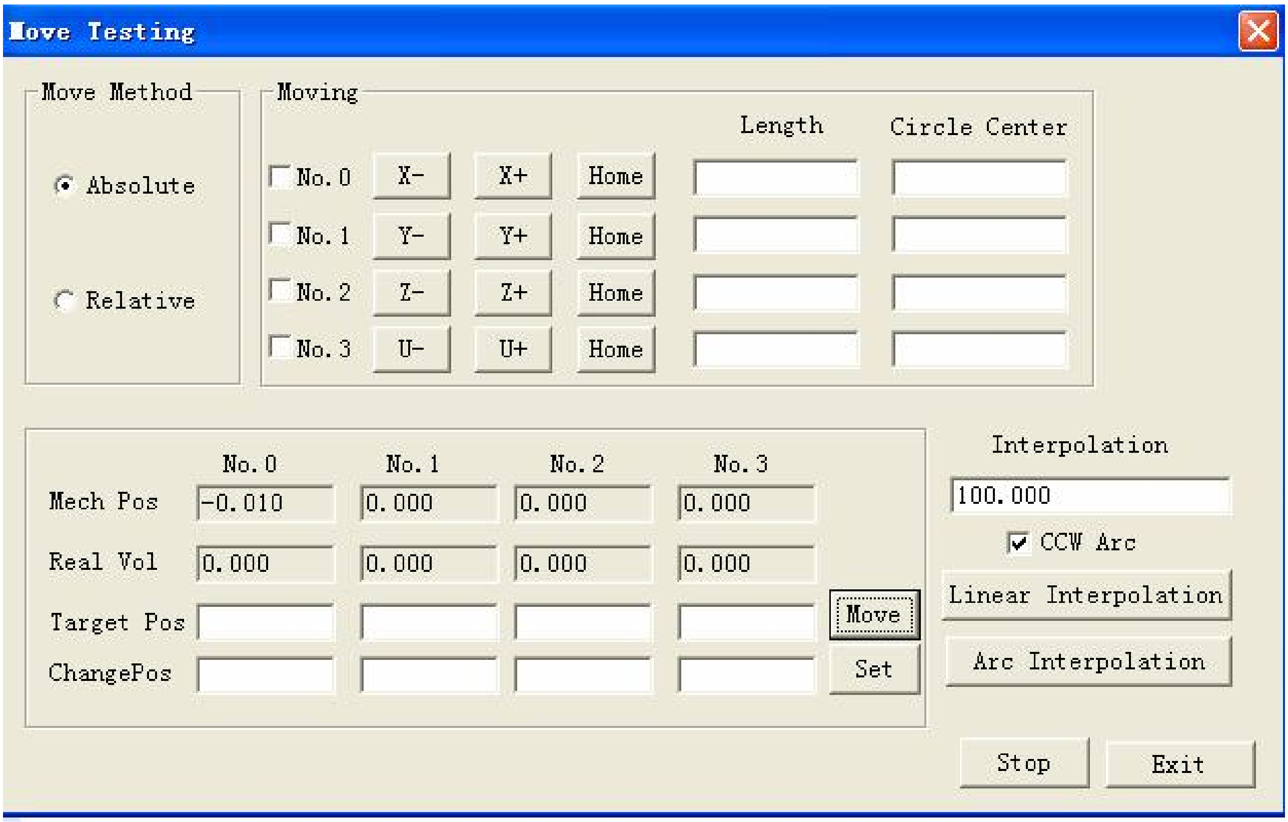
DA2: 0.000

Figure 2-6: I/O Testing Window

2.6 Move Test

Click the “Move Text” button to open the “Move Test” window, shown in figure 2-7. User can perform point-to-point move, homing move, linear interpolation, (arc) circular interpolation, changing of coordinates and manual move in this window.

Move Test Window	
Region	**Description**
Absolute	The controller selects absolute coordinates.
Relative	The controller selects relative coordinates.
Interpolation	Specifies the interpolation speed



Move Testing

Move Method

● Absolute

○ Relative

Moving

☐ No. 0

☐ No. 1

☐ No. 2

☐ No. 3

	No. 0	No. 1	No. 2	No. 3
Mech Pos	-0.010	0.000	0.000	0.000
Real Vol	0.000	0.000	0.000	0.000
Target Pos				
ChangePos				

Interpolation: 100.000

☑ CCW Arc

Figure 2-7: Move Test Window

Chapter 3 Controller Settings

All Leadshine SMC6480's settings are formatted into a INI file in the PC. Motion6480 can read and write this file. User can also read or write the setting via the Ethernet function **SMCCommand()**. See chapter 5 for more information.

The controller setting is protected by setting password. SMC6480 will ask you to input the password if it is set. Most of the parameters can be changed by the HMI or touch screen.

3.1 Parameter Overview

Parameter	Default Value	Description
ZeroSpeed1	20	Axis 1 homing speed
ZeroSpeed2	20	Axis 2 homing speed
ZeroSpeed3	20	Axis 3 homing speed
ZeroSpeed4	20	Axis 4 homing speed
LocateSpeed1	100	Axis 1 locating(move) speed
LocateSpeed2	100	Axis 2 locating(move) speed
LocateSpeed3	100	Axis 3 locating(move) speed
LocateSpeed4	100	Axis 4 locating(move) speed
ACC1	2000	Axis 1 acceleration
ACC2	2000	Axis 2 acceleration
ACC3	2000	Axis 3 acceleration
ACC4	2000	Axis 4 acceleration
DeACC1	0	Axis 1 deceleration
DeACC2	0	Axis 2 deceleration
DeACC3	0	Axis 3 deceleration
DeACC4	0	Axis 4 deceleration
UnitPulses1	100	Axis 1 user pulses per unit
UnitPulses2	100	Axis 2 user pulses per unit
UnitPulses3	100	Axis 3 user pulses per unit
UnitPulses4	100	Axis 4 user pulses per unit
VectSpeed	20	Interpolation
VectACC	200	Interpolation acceleration
VectDeACC	0	Interpolation deceleration
IfSoftLimit1	0	Enable or disable software limit of axis 1
IfSoftLimit2	0	Enable or disable software limit of axis 2
IfSoftLimit3	0	Enable or disable software limit of axis 3
IfSoftLimit4	0	Enable or disable software limit of axis 4
SoftLimitPlus1	200	Positive software limit of axis 1

SoftLimitPlus2	200	Positive software limit of axis 2
SoftLimitPlus3	200	Positive software limit of axis 3
SoftLimitPlus4	200	Positive software limit of axis 4
SoftLimitDec1	0	Negative software limit of axis 1
SoftLimitDec2	0	Negative software limit of axis 2
SoftLimitDec3	0	Negative software limit of axis 3
SoftLimitDec4	0	Negative software limit of axis 4
IfCompensate	0	Enable or disable backlash compensation
Compensate1	0	Axis 1 backlash
Compensate2	0	Axis 2 backlash
Compensate3	0	Axis 3 backlash
Compensate4	0	Axis 4 backlash
WorkZero1	0	Axis 1 default work piece origin
WorkZero2	0	Axis 2 default work piece origin
WorkZero3	0	Axis 3 default work piece origin
WorkZero4	0	Axis 4 default work piece origin
IfZeroPlus1	0	Axis 1 homing direction (1-positive)
IfZeroPlus2	0	Axis 2 homing direction (1-positive)
IfZeroPlus3	0	Axis 3 homing direction (1-positive)
IfZeroPlus4	0	Axis 4 homing direction (1-positive)
IfZero1	1	Enable or disable manual homing of axis 1
IfZero2	1	Enable or disable manual homing of axis 2
IfZero3	1	Enable or disable manual homing of axis 3
IfZero4	1	Enable or disable manual homing of axis 4
ZeroMode1	3	Axis 1 homing mode
ZeroMode2	3	Axis 2 homing mode
ZeroMode3	3	Axis 3 homing mode
ZeroMode4	3	Axis 3 homing mode
IfAutoZero	0	Enable or disable power-up auto homing
IfAutoStart	0	Enable or disable power-up auto running
IfDecWhenSoftLimit	0	Enable or disable decelerating stop of software limit
IfElAsHome	1	Enable or disable deceleration stop when taking HOME input as End Limit input
M07Delay	0	M07 delay time
M08Delay	0	M08 delay time
M09Delay	0	M09 delay time
M10Delay	0	M10 delay time
M11Delay	0	M11 delay time
M12Delay	0	M12 delay time
M08AheadDistance	0	M08 ahead interpolation distance

M10AheadDistance	0	M10 ahead interpolation distance
IfMoveWhenPause	0	Enable or disable manual move during program pause
IfGContinueLine	1	Enable or disable continue speed during program run
IfCornerDec	1	Enable or disable corner slow down
IfArcDec	1	Enable or disable arc speed limit
CornerDecSet	1	Decelerating rate of corner slow down
ArcDecSet	1	Limit rate for arc speed
JogLength1	1	Jogging step of axis 1
JogLength2	1	Jogging step of axis 2
JogLength3	1	Jogging step of axis 3
JogLength4	1	Jogging step of axis 4
JogLowSpeed	16	Jogging low speed
IfIoJog1	0	Enable or disable jogging switch input of axis 1
IfIoJog2	0	Enable or disable jogging switch input of axis 2
IfIoJog3	0	Enable or disable jogging switch input of axis 3
IfIoJog4	0	Enable or disable jogging switch input of axis 4
IfJogReverse1	0	Reverse jogging direction of axis 1
IfJogReverse2	0	Reverse jogging direction of axis 2
IfJogReverse3	0	Reverse jogging direction of axis 3
IfJogReverse4	0	Reverse jogging direction of axis 4
IfHandWheel1	0	Enable or disable hand wheel(MPG) input of axis 1
IfHandWheel2	0	Enable or disable hand wheel(MPG) input of axis 2
IfHandWheel3	0	Enable or disable hand wheel(MPG) input of axis 3
IfHandWheel4	0	Enable or disable hand wheel(MPG) input of axis 4
IfStartDownAsPause	0	Enable or disable pause function of Start switch input
IfStartUpAsPause	0	Pause program when Start switch input is released
IfStartIo	0	Enable or disable Start switch input
IfStopIo	0	Enable or disable Stop switch input
IfPauseIo	0	Enable or disable Pause switch input
IfZeroIo	0	Enable or disable homing switch input
ModbusID	8	MODBUS address
ZeroSequence1	1	Axis 1 homing sequence
ZeroSequence2	1	Axis 2 homing sequence
ZeroSequence3	1	Axis 3 homing sequence
ZeroSequence4	1	Axis 4 homing sequence
LocateSequence1	1	Axis 1 locating sequence
LocateSequence2	1	Axis 2 locating sequence
LocateSequence3	1	Axis 3 locating sequence
LocateSequence4	1	Axis 4 locating sequence
BeginSub	0	Begin teaching sub-program

EndSub	0	End teaching sub-program
IfCornerArc	0	Enable or disable fillets connection for corner
CornerArcRadius	2	Specify fillets radius
DefProgram		Default program name
IPAddr	192.168.1.11	Default IP address
IPGate	192.168.1.1	Default IP gate
IPMask	255.255.255.0	Default IP mask
IfDhcp	0	Enable or disable DHCP
IfHomeHighValid1	0	Axis 1 HOME input is active high or not
IfHomeHighValid2	0	Axis 2 HOME input is active high or not
IfHomeHighValid3	0	Axis 3 HOME input is active high or not
IfHomeHighValid4	0	Axis 4 HOME input is active high or not
IfSDValid1	0	Enable or disable SD (Start Deceleration) input of axis 1
IfSDValid2	0	Enable or disable SD (Start Deceleration) input of axis 2
IfSDValid3	0	Enable or disable SD (Start Deceleration) input of axis 3
IfSDValid4	0	Enable or disable SD (Start Deceleration) input of axis 4
IfSDHighValid1	0	Axis 1 SD (Start Deceleration) input is active high or not
IfSDHighValid2	0	Axis 2 SD (Start Deceleration) input is active high or not
IfSDHighValid3	0	Axis 3 SD (Start Deceleration) input is active high or not
IfSDHighValid4	0	Axis 4 SD (Start Deceleration) input is active high or not
IfINPValid1	0	Enable or disable INP(In-position) input of axis 1
IfINPValid2	0	Enable or disable INP(In-position) input of axis 2
IfINPValid3	0	Enable or disable INP(In-position) input of axis 3
IfINPValid4	0	Enable or disable INP(In-position) input of axis 4
IfINPHighValid1	0	Axis 1 INP (In-position) input is active high or not
IfINPHighValid2	0	Axis 2 INP (In-position) input is active high or not
IfINPHighValid3	0	Axis 3 INP (In-position) input is active high or not
IfINPHighValid4	0	Axis 4 INP (In-position) input is active high or not
IfERCOut1	0	Enable or disable ERC(Error-clear) output of axis 1
IfERCOut2	0	Enable or disable ERC(Error-clear) output of axis 2
IfERCOut3	0	Enable or disable ERC(Error-clear) output of axis 3
IfERCOut4	0	Enable or disable ERC(Error-clear) output of axis 4
IfERCHighValid1	0	Axis 1 ERC (Error-clear) output is active high or not
IfERCHighValid2	0	Axis 2 ERC (Error-clear) output is active high or not
IfERCHighValid3	0	Axis 3 ERC (Error-clear) output is active high or not
IfERCHighValid4	0	Axis 4 ERC (Error-clear) output is active high or not
InFilterSet	2	Filter configuration(0-99)
PulseSet1	0	Axis 1 pulse output mode
PulseSet2	0	Axis 2 pulse output mode
PulseSet3	0	Axis 3 pulse output mode

PulseSet4	0	Axis 4 pulse output mode
IfSCurve1	1	Enable or disable S-curve of axis 1
IfSCurve2	1	Enable or disable S-curve of axis 2
IfSCurve3	1	Enable or disable S-curve of axis 3
IfSCurve4	1	Enable or disable S-curve of axis 4
SCurveSet1	0	Axis 1 S-curve setting (0-1)
SCurveSet2	0	Axis 2 S-curve setting (0-1)
SCurveSet3	0	Axis 3 S-curve setting (0-1)
SCurveSet4	0	Axis 4 S-curve setting (0-1)
IfVectSCurve	0	Enable or disable S-curve for interpolation
VectSCurveSet	0	S-curve setting for interpolation
IfDecStopWhenEl	0	Enable or disable decelerating stop when end limit is activated
IfEMGHighValid	0	EMG(Emergency) input is active high or not
IfELHighValid1	0	Axis 1 end limit input is active high or not
IfELHighValid2	0	Axis 2 end limit input is active high or not
IfELHighValid3	0	Axis 3 end limit input is active high or not
IfELHighValid4	0	Axis 4 end limit input is active high or not

3.2 Parameter Detail

3.2.1. UnitPulses

UnitPulses is the equivalent pulses count per long measure unit. The long measure unit can be centimeter, millimeter, degree and revolution, etc. By default it is millimeter. User can calculate it as follows:

$$UnitPulses \;=\; Pulses\;\; Count\;\; of\;\; N\;\; \mathrm{Re}volutions\; /\; Move\;\; Dis\tan ce\;\; of\;\; N\;\; \mathrm{Re}voulutions$$

Note 1: This parameter can only be an integer. If the calculated value is not integer, try to use other long measure unit or modify the micro step resolution.

Note 2: The speed unit is based on user selected unit pulses.

3.2.2. PulsesSet

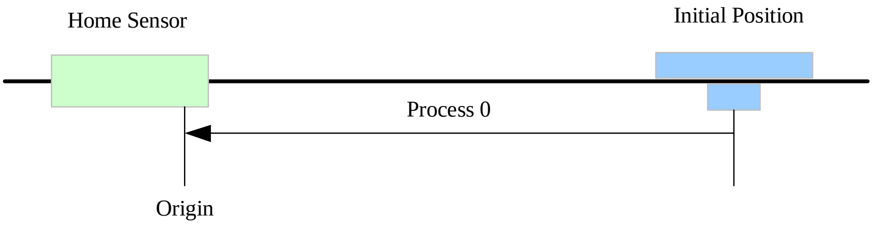
PulseSet is the pulse output mode. There are totally 6 pulse output modes as follows:

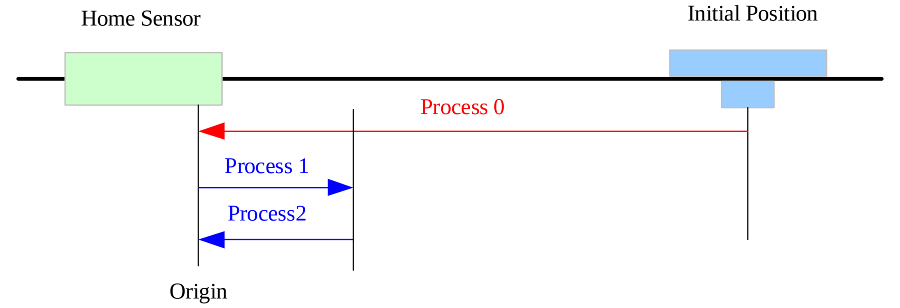
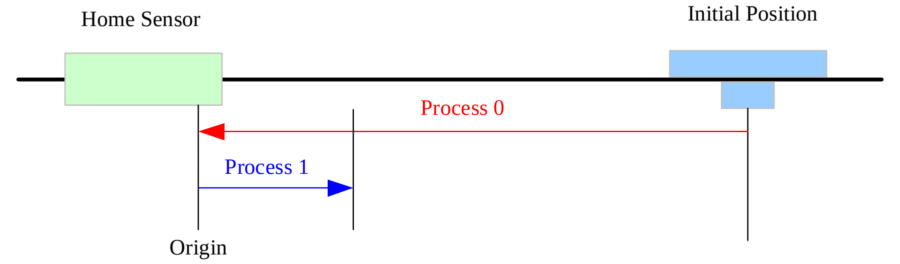
PulseSet	Pulse Mode	Description
0	Pulse + Direction	Direction signal is high level for positive direction. Pulse signal is normal high level
1	Pulse + Direction	Direction signal is high level for positive direction. Pulse signal is normal low level
2	Pulse + Direction	Direction signal is high level for negative direction. Pulse signal is normal high level
3	Pulse + Direction	Direction signal is high level for negative direction. Pulse signal is normal low level
4	Pulse + Pulse	Pulse signal is normal high level.
5	Pulse + Pulse	Pulse signal is normal low level.

PulseSet	Positive Move		Negative Move	
	Pulse Signal	Direction Signal	Pulse Signal	Direction Signal
0	‾|_|‾|_|‾	High	‾|_|‾|_|‾	Low
1	_|‾|_|‾|_	High	_|‾|_|‾|_	Low
2	‾|_|‾|_|‾	Low	‾|_|‾|_|‾	High
3	_|‾|_|‾|_	Low	_|‾|_|‾|_	High
4	‾|_|‾|_|‾	High	High	‾|_|‾|_|‾
5	_|‾|_|‾|_	Low	Low	_|‾|_|‾|_

3.2.3. ZeroMode

ZeroMode represents homing mode which can be 1, 2 and 3.

ZeroMode	Description
1	**Process 0:** Move the axis to the sensor and stop immediately when the sensor is activated. 

2	**Process 0:** Move axis to the sensor in setting velocity and stop when the sensor is activated. **Process 1:** Back off a little distance. **Process 2:** Move axis to the sensor in low velocity and stop when the sensor is activated.  Home Sensor Initial Position Process 0 Process 1 Process2 Origin
3	**Process 0:** Move axis to the sensor in setting velocity and stop when the sensor is activated. **Process 1:** Back off a little distance.  Home Sensor Initial Position Process 0 Process 1 Origin

3.2.4. WorkZero

WorkZero is the work piece origin which is based on user long measured. All locating coordinates in G code program are referenced to work piece zero. WorkZero can be modified by G53 and G92. G54 resets the work piece zero. (Note: User can set a special work piece origin in the G code program. See more information in HMI chapter. WorkZero will be ignored if a special work piece origin has been used.

3.2.5. IfElAsHome

Parameter IfElAsHome specifies whether the axis stops in decelerating way when the End Limit switch is activated during the homing process. If the End Limit switch is taken as the homing sensor, it is recommended to turn on this parameter. Thus there will be no big vibration that caused by immediate stop during homing.

Note: When IfElAsHome is on, you should leave enough distance after the End Limit switch. for decelerating stop.

3.2.6. MxxDelay

The M07, M09, M09, M10, M11 and M12 can control the special digital outputs. MxxDelay represents the delay time. The next statement is only executed after the delay time is out. This parameter has the same effect as G04.

3.2.7. MxxAheadDistance

M08AheadDistance and M10AheadDistance specify the advance distance to turn off the corresponding output before the continuous interpolation move is finished. It is based on user long measured unit. For example in the dispensing machine, it can be used to turn off dispensing ahead when it is necessary.
Note: It will be ignored when the continuous interpolation (IfGContinueLine) is turn off.

3.2.8. Corner Slow Down

Set **IfCornerDec** 1 turn on the corner slow down. Use **CornerDecSet** to change the slow down rate for continuous interpolation. The default value of **CornerDecSet** is 1.00 and increase **CornerDecSet** will also increase the speed. **CornerDecSet** will affect the transition speed between two curves.
Note: Modification of interpolation start speed also affects the corner slow down speed.

3.2.9. Fillet Settings

Set **IfCornerDec** 1 to turn on fillet for corner. The **CornerArcRadius** is the fillet radius. When it is turned on, corner will be replaced by arc.
Note: Fillet remove the corner thus the corner slow down speed will not be affected.

3.2.10. Arc Speed Limit

Set **IfCornerDec** 1 turn on the arc speed limit. Use Arc**DecSet** to limit the maximum arc speed rate. The default value of **CornerDecSet** is 1.00 and increase Arc**DecSet** will also increase the speed.
Note: Turn on the arc speed limit when there is small arc interpolation in motion program.

3.2.11. Jog Settings

User can jog the axis for manual move or teaching in touch screen or by digital inputs. The related settings are list as below.
JogLength: Jogging step distance when the button is clicked or the corresponding dedicated input is activated.
JogSpeed: Jogging speed in percentage related to the locating (move) speed.
JogLowSpeed: Jogging speed for the beginning 1 second.
IfIoJog: Enable or disable manual move function of the dedicated digital inputs (See dedicated input settings)
IfJogReverse: If it is 1**,** the jogging direction will be **i**nverted.

3.2.12. ZeroSequenceX

ZeroSequnce affects the manual homing sequence of each axis. The axis which has smaller **ZeroSequence** will home to origin first. If all axes have the same **ZeroSequence**, they will be homed at the same time.

Note: This parameter only take effects when it is manual homing.

3.2.13. Dedicated Digital Inputs and Outputs

The following parameters are used to enable or disable the dedicated IO:
IfStartIo: Enable or disable **Start** Input
IfStopIo: Enable or disable **Stop** Input
IfPauseIo: Enable or disable **Pause** Input
IfZeroIo: Enable or disable **Home** Input
IfSDValid: Enable or disable **SD (Slow Down)** Input
IfINPValid: Enable or disable INP(In-position) Input
IfIoJog: Enable or disable jogging Inputs
IfHandWheel: Enable or disable handwheel (MPG) Input

The following table lists the dedicated I/O of Leadshine SMC6480:

Dedicated IO	Function
IN1	Start (motion program)
IN2	Pause(motion program)
IN3	Stop (motion program)
IN4	Home (to origin)
IN5	X+(Axis 1 positive jogging)
IN 6	X-(Axis 1 negative jogging)
IN 7	Y+(Axis 2 positive jogging)
IN 8	Y-(Axis 2 negative jogging)
IN 9	Z+(Axis 3 positive jogging)
IN 10	Z-(Axis 3 negative jogging)
IN 11	U+(Axis 4 positive jogging)
IN 12	U-(Axis 4 negative jogging)
IN 13	INP1 (Axis 1 in-position signal)
IN 14	INP2 (Axis 2 in-position signal)
IN 15	INP3 (Axis 3 in-position signal)
IN 16	INP4 (Axis 4 in-position signal)
IN 21	ALARM1 (Axis 1 alarm signal)
IN 22	ALARM2 (Axis 2 alarm signal)
IN 23	ALARM3 (Axis 3 alarm signal)
IN 24	ALARM4 (Axis 4 alarm signal)
IN 25	Hand wheel (MPG) phase A input
IN 26	Hand wheel (MPG) phase B input
IN 27	10 times frequency selection for hand wheel (MPG)

IN 28	100 times frequency selection for hand wheel (MPG)
IN 29	Enable or disable hand wheel (MPG) input of axis 1
IN 30	Enable or disable hand wheel (MPG) input of axis 2
IN 31	Enable or disable hand wheel (MPG) input of axis 3
IN 32	Enable or disable hand wheel (MPG) input of axis 4
OUT1	Digital output controlled by M07 or M08. This output can output large current.
OUT 2	Digital output controlled by M09 or M10. This output can output large current.
OUT 3	Digital output controlled by M11 or M12.
OUT 13	ERC1 (Error clear output of axis 1)
OUT 14	ERC2 (Error clear output of axis 2)
OUT 15	ERC3 (Error clear output of axis 3)
OUT 16	ERC4 (Error clear output of axis 4)

Chapter 4 Motion Program Development

Leadshine SMC6480 supports development of motion program using C/C++, Basic language. If you are not familiar with them, you can also use the Leadshine G code too.

4.1 G Code Programming

Before starting your programming, it is strongly recommended you to study the Leadshine G code in Chapter 6 first. Make sure Leadshine G code can implement all the require functions in your application.

You can type in the G code line by line in the touch screen directly. However, it is recommended to compile your G code program in PC then download it to the controller.

The following procedure illustrates how to compose your G code program then download to the controller:

1) Compose your G code motion program. You can use **Motion6480** which is provided by Leadshine with no charge. You can also use notebook or other text editor you preferred.
2) Open **Motion6480** and connect it to the controller. Modify the controller's key parameter such as pulses count per user unit and work piece zero, etc. If the axes need to be home in your program, please click "I/O Test" then check whether the homing input is normal. Test other I/O as well if they will be used.
3) Click the "Editing Program" button in Motion6480 and import your G code program to the edit region. Then click the "Download" button. A dialog appears for you to select the G code program to be updated.
4) Return to the main window and click the "Move Test" button to go to the "Run Program" window. Select the G code program and run it to verify its function.

4.2 Programming in Visual Basic

Make sure the controller had been connected to PC. Motion6480 and Visual Basic had been installed correctly in the PC. Please walk through the following items before calling the SMC6480's API function:

1) Start Mitoin6480 and perform some basic test such as single axis move to make sure the SMC6480's firmware and hardware works well.
2) Create your own work directory such as D:\vbMotion.(Note: you can change the folder name).
3) Copy SMC6480.bas to the work directory. You can get it from Leadshine CD or Leadshine website at http://www.leadshine.com.
4) Run Visual Basic and create a new project then save it to the work directory.
5) Link your project to the motion library as follows:
   (1) Click "Project(P)" in Visual Basic and select "Add Module";
   (2) Click "Exiting";
   (3) Select "SMC6480.bas";
   (4) Click "OK".

Now the motion library has been linked to the project, you can now call the motion function like calling the API function. Please refer to chapter 5 for the detailed illustration of each motion function. You can also open the SMC6480.bas to study the detailed definition.

Leadshine provides the samples to the customer. You can download it from our website(http://www.leadshine.com) or copy them from Leadshine CD. If the SMC6480 has been connected to the PC and Visual Basic has been installed in the PC, theses samples can be run directly in VB.

4.3 Programming in Visual C++ 6.0

Make sure the controller had been connected to PC. Motion6480 and Visual C++ had been installed correctly in the PC. Please walk through the following items before calling the SMC6480's API function:

1) Start Mitoin6480 and perform some basic test such as single axis move to make sure the SMC6480's firmware and hardware work well.
2) Run Visual C++ and create a new project namely vcMotion in D:\vcMotion. Copy smc6x.lib, smc6x.dll and smc6480.h to this directory. You can get them from Leadshine CD or Leadshine website at http://www.leadshine.com.
3) Link to the motion library by adding the smc6x.lib to the project.
4) Type in "#include "smc6480.h" at the beginning of the file which calls the motion function.

Now the motion library has been linked to the project, you can now call the motion function like calling the API function. Please refer to chapter 5 for the detailed illustration of each motion function. You can also open the SMC6480.bas to study the detailed definition.

Leadshine provides the samples to the customer. You can download it from our website (http://www.leadshine.com) or copy them from Leadshine CD. If the SMC6480 has been connected to the PC and Visual Basic has been installed in the PC, theses samples can be run directly in VB.

Chapter 5 Ethernet Motion Control Functions

5.1 Functions Overview

5.1.1. Connection and Initialization

Connection and Initialization	
Function	**Description**
SMCOpen	Connect to controller
SMCOpenCom	Connect to controller via COM port
SMCOpenEth	Connect to controller via Ethernet port （IP address is string format)
SMCOpenEth2	Connect to controller via Ethernet port （IP address is socket format)
SMCClose	Disconnect from controller
SMCSetTimeOut	Set maximum response time out
SMCGetTimeOut	Read maximum response time out

5.1.2. Program File

Program File	
Function	**Description**
SMCDownProgram	Download motion program to controller’s FLASH memory
SMCDownMemProgram	Download motion program to controller’s FLASH memory
SMCDownProgramToTemp	Download motion program to controller’s temporary file
SMCRunProgramFile	Run motion program in FLASH
SMCRunTempFile	Run motion program in temporary file
SMCDownProgramToRamAndRun	Download motion program to RAM and run it
SMCDownMemProgramToRamAndRun	Download motion program to RAM and run it
SMCUpProgram	Upload motion program
SMCUpProgramToMem	Upload motion program
SMCPause	Pause motion program execution
SMCStop	Stop motion program execution
SMCCheckRemainProgramSpace	G remainder space of FLASH memory
SMCCheckProgramStopReason	Check stop reason of motion program
SMCGetCurRunningLine	Get current line number of executing motion program
SMCSetRunNoIO	Run motion program without I/O operation
SMCGetRunningOption	Get running options
SMCContinueRun	Continue executing after stop
SMCDeleteProgramFile	Delete motion program
SMCRemoveAllProgramFiles	Remove all motion programs
SMCCheckProgramFile	Get motion program information, i.e., size
SMCFindFirstProgramFile	Enumerate motion program

SMCFindNextProgramFile	Enumerate motion program
SMCCheckProgramSyntax	Check motion program syntax error

5.1.3. I/O Operation

I/O Operation	
Function	**Description**
SMCWriteLed	Turn on/off LED
SMCWriteOutBit	Write to output bit
SMCReadInBit	Read input bit
SMCReadOutBit	Read output bit
SMCWriteOutPort	Write output port
SMCReadInPort	Read input port
SMCReadOutPort	Read output port
SMCReadAlarmState	Get ALARM signal state (not supported by SMC6480)
SMCReadHomeState	Get HOME signal state
SMCReadEMGState	Get EMG signal state
SMCReadHandWheelStates	Get HandWheel(MPG) state
SMCReadElStates	Get EL(end limit) signal state
SMCReadSdStates	Get SD (start deceleration) signal state
SMCReadInpStates	Get INP(in-position) signal state
SMCReadAxisStates	Get axis status
SMCWritePwmDuty	Set PWM duty-cycle
SMCWritePwmFreqency	Set PWM frequency
SMCWriteDaOut	Set DA output
SMCReadPwmDuty	Get PWM ducy-cycle
SMCReadPwmFreqency	Get PWM frequency
SMCReadDaOut	Get DA Output

5.1.4. Motion Control

Program Operation	
Function	**Description**
SMCPMove	Specific distance move
SMCPMovePluses	Specific distance move in
SMCVMove	Constant speed move
SMCCheckDown	Check if axis has been stop
SMCHomeMove	Home axis
SMCIfHomeMoveing	Check if axis is homing
SMCDecelStop	Decelerating stop
SMCImdStop	Immediate stop
SMCEmgStop	Emergency stop

SMCChangeSpeed	Change speed on-the-fly
SMCGetPosition	Get current mechanical position
SMCGetWorkPosition	Get current work piece position
SMCGetPositionPulses	Get current mechanical position in pulse
SMCGetWorkOriginPosition	Get work piece origin
SMCSetPosition	Set position
SMCSetPositionPulses	Set position (pulse)
SMCWaitDown	Wait for motion done(not support now)
SMCHandWheelSet	Configure hand wheel (MPG)
SMCHandWheelMove	Hand wheel (MPG) move
SMCVectMoveStart	Start interpolation mode
SMCVectMoveEnd	End interpolation mode
SMCGetVectMoveState	Get interpolation state
SMCGetVectMoveRemainSpace	Get remainder space for interpolation
SMCVectMoveLine1	1-axis linear interpolation
SMCVectMoveLine2	2-axes linear interpolation
SMCVectMoveLineN	N-axes linear interpolation
SMCVectMoveMultiLine2	Multiple 2-axes linear interpolation
SMCVectMoveMultiLineN	Multiple N-axes linear interpolation
SMCVectMoveArc	2-axes circular interpolation
SMCVectMoveSetSpeedLimition	Set speed limit
SMCGetCurRunVectLength	Get current interpolation distance
SMCGetCurSpeed	Get current speed
SMCVectMovePause	Pause interpolation
SMCVectMoveStop	Stop interpolation

5.1.5. Setting File

Download / upload Setting File	
Function	**Description**
SMCDownSetting	Download setting from PC file
SMCDownMemSetting	Download setting from PC memory
SMCUpSetting	Upload setting to PC file
SMCUpSettingToMem	Upload setting to PC memory
SMCDownDefaultSetting	Download default setting form PC file
SMCDownMemDefaultSetting	Download default setting from PC memory
SMCUpDefaultSetting	Upload default setting to PC file
SMCUpDefaultSettingToMem	Upload default setting from to memory

5.1.6. Parameters

Configure Parameters	
Function	**Description**
SMCCommand	Modify setting via general string command
SMCBurnSetting	Burn to FLASH memory
SMCSetIpAddr	Modify IP address
SMCGetIpAddr	Get IP address
SMCGetCurIpAddr	Get current IP address
SMCSetZeroSpeed	Set homing speed
SMCGetZeroSpeed	Get homing speed
SMCSetLocateSpeed	Set single axis move speed
SMCGetLocateSpeed	Get single axis move speed
SMCSetLocateStartSpeed	Set single axis start speed
SMCGetLocateStartSpeed	Get single axis start speed
SMCSetLocateAcceleration	Set single axis acceleration
SMCGetLocateAcceleration	Get single axis acceleration
SMCSetLocateDeceleration	Set single axis deceleration
SMCGetLocateDeceleration	Get single axis deceleration
SMCSetUnitPulses	Set pulse count per unit
SMCGetUnitPulses	Get pulse count per unit
SMCSetVectStartSpeed	Set interpolation start speed
SMCGetVectStartSpeed	Get interpolation start speed
SMCSetVectSpeed	Set interpolation speed
SMCGetVectSpeed	Get interpolation speed
SMCSetVectAcceleration	Set interpolation acceleration
SMCGetVectAcceleration	Get interpolation acceleration
SMCSetVectDeceleration	Set interpolation deceleration
SMCGetVectDeceleration	Get interpolation deceleration

5.1.7. Password Operation

Password Operation	
Function	**Description**
SMCUpPasswordFile	Upload password file to PC file
SMCUpPasswordFileToMem	Upload password file to PC memory
SMCDownPasswordFile	Download password file from PC file
SMCDownMemPasswordFile	Download password file from memory
SMCEnterSetPassword	Input Parameters protection password
SMCEnterEditPassword	Input program protection password

SMCEnterSuperPassword	Input factory password
SMCEnterTimePassword	Input evaluation time-out password
SMCClearEnteredPassword	Clear time out password
SMCModifySetPassword	Set Parameters protection password
SMCModifyEditPassword	Set program protection password
SMCModifySuperPassword	Set factory password
SMCGetTrialCondition	Check if evaluation time is out

5.1.8. Other

Other	
Function	**Description**
SMCGetProgress	Get progress of downloading file
SMCGetState	Get controller state
SMCGetAxises	Get controllable axes
SMCGetSoftwareId	Get software type
SMCGetHardwareId	Get hardware version
SMCGetSoftwareVersion	Get firmware version
SMCModbus_Set0x	Set MODBUS bit register
SMCModbus_Get0x	Get MODBUS bit register
SMCModbus_Get4x	Set MODBUS word register
SMCModbus_Set4x	Get MODBUS word register
SMCGetErrcodeDescription	Get error code

5.2 Function Detail

5.2.1. Connection and Initialization

5.2.1.1. SMCOpen

Description:
Connect to controller

Prototype:
int32 SMCOpen(SMC6X_CONNECTION_TYPE type, char *pconnectstring , SMCHANDLE * phandle);

Parameters:

Type
Select connection Type.
SMC6X_CONNECTION_COM: Connect to controller via serial port
SMC6X_CONNECTION_ETH: Connect to controller via Ethernet.

Pconnectstring

IP address or Serial Port (com1 and com2) in string format

Phandle

Controller handle

Return Value:

Error code (See section 5.3 for the detail)

Example:

SMCHANDLE handle = NULL;
SMCOpen(SMC6X_CONNECTION_COM , “com2 ”, &handle);

5.2.1.2. SMCOpenCom

Description:

Connect to controller via COM port

Prototype:

int32 SMCOpenCom(uint comid, SMCHANDLE * phandle);

Parameters:

comid

COM port number from 1 to 255

Phandle

Controller handle

Return Value:

Error code (See section 5.3 for the detail)

Example:

SMCHANDLE handle = NULL;
SMCOpenCom(1 ,&handle);

5.2.1.3. SMCOpenEth

Description:

Connect to controller via Ethernet port (IP address is string format)

Prototype:

int32 SMCOpenEth(char *ipaddr, SMCHANDLE * phandle);

Parameters:

Ipaddr

IP address in string format

Phandle

Controller handle.

Return Value:

Error code (See section 5.3 for the detail)

Example:

5.2.1.4. SMCOpenEth2

Description:

Connect to controller via Ethernet port (IP address is socket format)

Prototype:

int32 SMCOpenEth2(struct in_addr straddr, SMCHANDLE * phandle);

Parameters:

straddr

IP address in socket

Phandle

Controller handle.

Return Value:

Error code (See section 5.3 for the detail)

Example:

```
struct in_addr straddr;
straddr.s_addr = inet_addr("192.168.1.11");
SMCOpenEth2(straddr, &handle);
```

5.2.1.5. SMCClose

Description:

Disconnect from controller

Prototype:

int32 SMCClose(SMCHANDLE handle);

Parameters:

handle

Controller handle.

Return Value:

Error code (See section 5.3 for the detail)

Example:

```
……
if (NULL != handle)
{
   SMCClose(handle);
   handle = NULL;
}
```

5.2.1.6. SMCSetTimeOut

Description:
Set maximum response time out

Prototype:
int32 SMCSetTimeOut(SMCHANDLE handle, uint32 timems);

Parameters:

handle
Controller handle.

timems
Time out in millisecond

Return Value:
Error code (See section 5.3 for the detail)

Example:
SMCSetTimeOut(handle, 10);

5.2.1.7. SMCGetTimeOut

Description:
Set maximum response time out

Prototype:
int32 SMCGetTimeOut(SMCHANDLE handle, uint32* ptimems);

Parameters:

handle
Controller handle.

timems
Time out in millisecond

Return Value:
Error code (See section 5.3 for the detail)

5.2.2. Program File

5.2.2.1. SMCDownProgram

Description:
Download motion program to controller's FLASH memory from PC file. It will be compiled once before downloading.

Prototype:
int32 SMCDownProgram(SMCHANDLE handle, const char* pfilename, const char* pfilenameinControl);

Parameters:

handle

Controller handle

pfilename

Name of PC file to be downloaded

pfilenameinControl

Name of file in controller FLASH

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.2. SMCDownMemProgram

Description:
Download motion program to controller's FLASH memory from PC memory. It will be compiled once before downloading.

Prototype:

int32 SMCDownMemProgram(SMCHANDLE handle, const char* pbuffer, uint32 buffsize, const char* filenameinControl);

Parameters:

handle

Controller handle

pbuffer

Buffer address in PC memory

buffsize

Buffer size

filenameinControl

Name of file in controller

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.3. SMCDownProgramToTemp

Description:
Download motion program in PC file to controller's temporary file

Prototype:
int32 SMCDownProgramToTemp(SMCHANDLE handle, const char* pfilename);

Parameters:

handle

Controller handle

pfilename

Name of PC file

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.4. SMCRunProgramFile

Description:
Run motion program in FLASH

Prototype:
int32 SMCRunProgramFile(SMCHANDLE handle, const char* pfilenameinControl);

Parameters:

handle

Controller handle

pfilenameinControl

Name of motion program file in controller

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.5. SMCRunTempFile

Description:
Run motion program in temporary file

Prototype:
int32 SMCRunTempFile(SMCHANDLE handle);

Parameters:

handle
Controller handle

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.6. SMCDownProgramToRamAndRun

Description:
Download motion program in PC file to controller RAM and run it

Prototype:

int32 SMCDownProgramToRamAndRun(SMCHANDLE handle, const char* pfilename);

Parameters:

handle
Contrller handle

pfilename
Name of motion program file in PC

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.7. SMCDownMemProgramToRamAndRun

Description:
Download motion program in PC memory to controller RAM and run it

Prototype:

int32 SMCDownMemProgramToRamAndRun(SMCHANDLE handle, const char* pbuffer, uint32 buffsize);

Parameters:

handle

Controller handle

pbuffer

Buffer address in PC memory

buffsize

Buffer size

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.8. SMCUpProgram

Description:
Upload motion program to PC file

Prototype:

int32 SMCUpProgram(SMCHANDLE handle, const char* pfilename, const char* pfilenameinControl);

Parameters:

handle

Controller handle

pfilename

PC file name

pfilenameinControl

Name of motion program file in controller

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.9. SMCUpProgramToMem

Description:
Upload motion program to PC memory

Prototype:

int32 SMCUpProgramToMem(SMCHANDLE handle, char* pbuffer, uint32 buffsize, char* pfilenameinControl, uint32* puifilesize);

Parameters:

handle

Controller handle

pbuffer

Buffer address in PC memory

buffsize

Buffer size

pfilenameinControl

Name of file in controller

puifilesize

Actual size uploaded

Return Value:

Error code (See section 5.3 for the detail)

5.2.2.10. SMCPause

Description:

Pause motion program execution

Prototype:

int32 SMCPause(SMCHANDLE handle);

Parameters:

handle

Controller handle

Return Value:

Error code (See section 5.3 for the detail)

5.2.2.11. SMCStop

Description:

Stop motion program execution

Prototype:

int32 SMCStop(SMCHANDLE handle);

Parameters:

handle

Controller handle

Return Value:

Error code (See section 5.3 for the detail)

5.2.2.12. SMCCheckRemainProgramSpace

Description:

Get remainder space in FLASH memory

Prototype:

int32 SMCCheckRemainProgramSpace(SMCHANDLE handle, uint32 * pRemainSpaceInKB);

Parameters:

handle

Controller handle

pRemainSpaceInKB

Pointer to a uint32 variable that receives remainder space in controller's flash memory (unit: KB).

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.13. SMCCheckProgramStopReason

Description:
Check the reason for motion program stop

Prototype:

int32 SMCCheckProgramStopReason(SMCHANDLE handle, uint32 * pStopReason);

Parameters:

handle

Controller handle

pStopReason

Pointer to a uint32 variable that receives a number which indicates the stop reason of motion program. See ERR_GSTOP_OFFSET for the detail.

Return Value:
Error code (See section 5.3 for the detail)

Example:

5.2.2.14. SMCGetCurRunningLine

Description:
Get current line number of executing motion program

Prototype:

int32 SMCGetCurRunningLine(SMCHANDLE handle, uint32 * pLineNum);

Parameters:

handle

Controller handle

pLineNum

Pointer to a uint32 variable that receives current line number of running motion program .

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.15. SMCSetRunNoIO

Description:
Enable or disable I/O operation when motion program is running

Prototype:

int32 SMCSetRunNoIO(SMCHANDLE handle, uint8 bifVainRun);

Parameters:

handle

Controller handle

bifVainRun

Specify whether motion program run without I/O operation or not

Return Value:
Error code (See section 5.3 for the detail)

Example:

SMCSetRunNoIO(handle, 1); // Set bifVainRun to Run motion program without I/O operation.

5.2.2.16. SMCGetRunningOption

Description:
Get running options

Prototype:

int32 SMCGetRunningOption(SMCHANDLE handle, uint8* bifStep, uint8* bifVainRun);

Parameters:

handle

Controller handle

bifStep

Pointer to a variable that indicates whether motion progrom runs step by step.

bifVainRun

Pointer to a variable that indicates whether motion progrom runs without I/O operation.

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.17. SMCContinueRun

Description:
Continue executing of motion program which is pause.

Prototype:

int32 SMCContinueRun(SMCHANDLE handle);

Parameters:

handle

Controller handle

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.18. SMCDeleteProgramFile

Description:
Delete motion program

Prototype:

int32 SMCDeleteProgramFile(SMCHANDLE handle, const char* pfilenameinControl);

Parameters:

handle

Controller handle

pfilenameinControl

Name of file in controller FLASH

Return Value:
Error code (See section 5.3 for the detail)

Example:

5.2.2.19. SMCRemoveAllProgramFiles

Description:
Remove all motion programs in controller FLASH

Prototype:

int32 SMCRemoveAllProgramFiles(SMCHANDLE handle);

Parameters:

handle

Controller handle

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.20. SMCCheckProgramFile

Description:
Get motion program information such as file size

Prototype:

int32 SMCCheckProgramFile(SMCHANDLE handle, const char* pfilenameinControl, uint8 *pbIfExist, uint32 *pFileSize);

Parameters:

handle

Controller handle

pfilenameinControl

Name of motion program file in controller FLASH

pbIfExist

Pointer to a variable that indicates whether motion program file is exist

pFileSize

Pointer to a variable that receives file size in controller. The actual uploading size may not be the same as file size in controller.

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.21. SMCFindFirstProgramFile

Description:
Fine the first motion program in FLASH

Prototype:

int32 SMCFindFirstProgramFile(SMCHANDLE handle, char* pfilenameinControl, uint32 *pFileSize);

Parameters:

handle

Controller handle

pfilenameinControl

Name of motion program file in controller FLASH *pFileSize*

Pointer to a variable that receives file size in controller. The actual uploading size may not be the same as file size in controller.

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.22. SMCFindNextProgramFile

Description:
Find next motion program in controller

Prototype:

int32 SMCFindNextProgramFile(SMCHANDLE handle, char* pfilenameinControl, uint32 *pFileSize);

Parameters:

handle

Controller handle

pfilenameinControl

Name of motion program file in controller FLASH

pFileSize

Pointer to a variable that receives file size in controller. The actual uploading size may not be the same as file size in controller.

Return Value:
Error code (See section 5.3 for the detail)

5.2.2.23. SMCCheckProgramSyntax

Description:
Check motion program syntax error

Prototype:

int32 SMCCheckProgramSyntax(const char *sin, char *sError);

Parameters:

sin

Pointer to a string that represents motion program

sError

Pointer to a variable that receives motion program error

Return Value:
Error code (See section 5.3 for the detail)

Example:

const char sin[10] = “G00”;

char sError [1024];

SMCCheckProgramSyntax(sin,sError);

5.2.3. Setting File

5.2.3.1. SMCDownSetting

Description:
Download settings to controller from PC file

Prototype:
int32 SMCDownSetting(SMCHANDLE handle, const char* pfilename);

Parameters:

handle
Controller handle

pfilename
PC file name in string format

Return Value:
Error code (See section 5.3 for the detail)

5.2.3.2. SMCDownMemSetting

Description:

Download settings to controller from PC memory

Prototype:
int32 SMCDownMemSetting(SMCHANDLE handle, const char* pbuffer, uint32 buffsize);

Parameters:

handle
Controller handle

pbuffer
Buffer address in PC memory

Buffsize
Buffer size

Return Value:

Error code (See section 5.3 for the detail)

5.2.3.3. SMCUpSetting

Description:

Upload settings to PC file

Prototype:
int32 SMCUpSetting(SMCHANDLE handle, const char* pfilename);

Parameters:

Phandle
Controller handle.

pfilename
Target file name in string format

Return Value:
Error code (See section 5.3 for the detail)

5.2.3.4. SMCUpSettingToMem

Description:

Upload settings to PC memory

Prototype:
int32 SMCUpSettingToMem(SMCHANDLE handle, char* pbuffer, uint32 buffsize, uint32* puifilesize);

Parameters:

Phandle
Controller handle.

Pbuffer

Buffer address in PC memory

Buffsize

Buffer size

Puifilesize

Actual size uploaded

Return Value:
Error code (See section 5.3 for the detail)

5.2.3.5. SMCDownDefaultSetting

Description:

Download default settings to controller form PC file

Prototype:
int32 SMCDownDefaultSetting(SMCHANDLE handle, const char* pfilename);

Parameters:

phandle
Controller handle.

pfilename

Name of PC file including default controller settings

Return Value:
Error code (See section 5.3 for the detail)

5.2.3.6. SMCDownMemDefaultSetting

Description:

Download default settings from PC memory

Prototype:
int32 SMCDownMemDefaultSetting(SMCHANDLE handle, const char* pbuffer, uint32 buffsize);

Parameters:

phandle
Controller handle

pbuffer

Buffer address in PC memory

buffsize

Buffer size

Return Value:
Error code (See section 5.3 for the detail)

5.2.3.7. SMCUpDefaultSetting

Description:

Upload default settings to PC file

Prototype:
int32 SMCUpDefaultSetting(SMCHANDLE handle, const char* pfilename);

Parameters:

phandle
Controller handle.

pfilename

PC file name in string format

Return Value:
Error code (See section 5.3 for the detail)

5.2.3.8. SMCUpDefaultSettingToMem

Description:

Upload default settings from controller to PC memory

Prototype:
int32 SMCUpDefaultSettingToMem(SMCHANDLE handle, char* pbuffer, uint32 buffsize, uint32* puifilesize);

Parameters:

phandle
Controller handle.

pbuffer

Buffer address in PC memory

buffsize

Buffer size

puifilesize

Actual size uploaded

Return Value:
Error code (See section 5.3 for the detail)

5.2.4. Motion Control

5.2.4.1. SMCPMove

Description:
Specific coordinate move in millimeter

Prototype:

int32 SMCPMove(SMCHANDLE handle, uint8 iaxis, double dlength, uint8 bIfAbs);
Parameters:

handle

Controller handle

iaxis

Specify which axis to be moved

dlength

Specifies the coordinate of the endpoint (default unit: mm)

bIfAbs

If *bIfAbs* is 1, *dlength* is absolute coordinate. If blfAbs is 0, *dlength* is relative coordinate.

Return Value:
Error code (See section 5.3 for the detail)

Example:

SMCPMove (handle, 0, 1000, 1); // Move axis-0 to 1000mm in absolute coordinate.

5.2.4.2. SMCPMovePluses

Description:
Specific coordinate move in pulse

Prototype:

int32 SMCPMovePluses(SMCHANDLE handle, uint8 iaxis, int32 ilength, uint8 bIfAbs);
Parameters:

handle

Controller handle

iaxis

Specify which axis to be moved

dlength

Specifies the coordinate of the endpoint (unit: pulses)

bIfAbs

If *bIfAbs* is 1, *dlength* is absolute coordinate. If blfAbs is 0, *dlength* is relative coordinate.

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.3. SMCVMove

Description:
Constant speed move

Prototype:

int32 SMCVMove(SMCHANDLE handle, uint8 iaxis, uint8 bIfPositiveDir);
Parameters:

handle

Controller handle

iaxis

Specify which axis to be moved

bIfPositiveDir

Specifies move direction. If bIfPositiveDir is 1, move direction is positive. If bIfPositiveDir is 0, move direction is negative.

Return Value:
Error code (See section 5.3 for the detail)

Example:

5.2.4.4. SMCCheckDown

Description:
Check if axis has been stopped

Prototype:

int32 SMCCheckDown(SMCHANDLE handle, uint8 iaxis, uint8* pbIfDown);
Parameters:

handle

Controller handle

iaxis

Specifies which axis to be checked.

pbIfDown

Pointer to a variable that indicates if axis has been stopped.

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.5. SMCHomeMove

Description:
Home axis

Prototype:

int32 SMCHomeMove(SMCHANDLE handle, uint8 iaxis);
Parameters:

handle

Controller handle

iaxis

Specifies which axis to be home

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.6. SMCIfHomeMoveing

Description:
Check if axis is at homing

Prototype:

int32 SMCIfHomeMoveing(SMCHANDLE handle, uint8 iaxis, uint8* pbIfHoming);
Parameters:

handle

Controller handle

iaxis

Specifies which axis to be checked

pbIfHoming

Pointer to a variable that indicates if axis is at homing.

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.7. SMCDecelStop

Description:
Decelerating stop

Prototype:

int32 SMCDecelStop(SMCHANDLE handle, uint8 iaxis);
Parameters:

handle

Controller handle

iaxis

Specifies which axis to be stopped in decelerating way

Return Value:
Error code (See section 5.3 for the detail)

Example:

5.2.4.8. SMCImdStop

Description:
Immediate stop

Prototype:

int32 SMCImdStop(SMCHANDLE handle,uint8 iaxis);
Parameters:

handle

Controller handle

iaxis

Specifies which axis to be stopped immediately

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.9. SMCEmgStop

Description:
Emergency stop

Prototype:
int32 SMCImdStop (SMCHANDLE handle, uint8 iaxis);

Parameters:

handle

Controller handle

iaxis

Specifies the emergency stop axis.

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.10. SMCChangeSpeed

Description:
Change speed on-the-fly

Prototype:

int32 SMCChangeSpeed(SMCHANDLE handle, uint8 iaxis, double dspeed);
Parameters:

handle

Controller handle

iaxis

Specifies the axis that needs to be changed speed on-the-fly

dspeed

Specifies the speed (unit: mm/s)

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.11. SMCGetPosition

Description:
Get current mechanical position

Prototype:

int32 SMCGetPosition(SMCHANDLE handle, uint8 iaxis, double* pposition);
Parameters:

handle

Controller handle

iaxis

Specifies the axis

pposition

Pointer to a variable that receives the mechanical position

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.12. SMCGetWorkPosition

Description:
Get current work piece position

Prototype:

int32 SMCGetWorkPosition(SMCHANDLE handle, uint8 iaxis, double* pposition);
Parameters:

handle

Controller handle

iaxis

Specifies the axis

pposition

Pointer to a variable that receives the work piece position

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.13. SMCGetPositionPulses

Description:
Get mechanical position in pulse

Prototype:

int32 SMCGetPositionPulses(SMCHANDLE handle, uint8 iaxis, int32* pposition);

Parameters:

handle

Controller handle

iaxis

Specifies the axis

pposition

Pointer to a variable that receives the mechanical position (unit: mm)

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.14. SMCGetWorkOriginPosition

Description:
Get work piece origin

Prototype:

int32 SMCGetWorkOriginPosition(SMCHANDLE handle, uint8 iaxis, double* pposition);

Parameters:

handle

Controller handle

iaxis

Specifies the axis

pposition

Pointer to a variable that receives the work piece origin position

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.15. SMCSetPosition

Description:
Set position

Prototype:

int32 SMCSetPosition(SMCHANDLE handle, uint8 iaxis, double dposition);
Parameters:

handle

Controller handle

iaxis

Specifies the axis

pposition

Specifies the position to be set

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.16. SMCSetPositionPulses

Description:
Set position (pulse)

Prototype:

int32 SMCSetPositionPulses(SMCHANDLE handle,uint8 iaxis, int32 iposition);
Parameters:

handle

Controller handle

iaxis

Specifies the axis

pposition

Specifies the position to be set (unit: pulses)

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.17. SMCHandWheelSet

Description:
Configure hand wheel (MPG)

Prototype:

int32 SMCHandWheelSet(SMCHANDLE handle,uint8 iaxis, uint16 imulti, uint8 bifDirReverse);
Parameters:

handle

Controller handle

iaxis

Specifies the axis

imulti

Specifies the factor that manual pulse frequency will be multiplied

bifDirReverse

Specifies if move direction of manual input pulse will be reversed. If *bifDirReverse* is 1, direction will be reversed. If *bifDirReverse* is 0, direction will not be reversed.

Return Value:
Error code (See section 5.3 for the detail)

Example:

SMCHandWheelSet(handle, 0, 2, 1);   // Configure manual pulse input of axis 0

5.2.4.18. SMCHandWheelMove

Description:
Enable or disable hand wheel (MPG) move

Prototype:

int32 SMCHandWheelMove(SMCHANDLE handle, uint8 iaxis, uint8 bifenable);
Parameters:

handle

Controller handle

iaxis

Specifies the axis

bifenable

Enable hand wheel (MPG) move if bifenable is 1. Disable hand wheel (MPG) move if bifenable is 0.

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.19. SMCVectMoveStart

Description:
Start interpolation mode

Prototype:

int32 SMCVectMoveStart(SMCHANDLE handle);
Parameters:

Handle

Controller handle

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.20. SMCVectMoveEnd

Description:
End interpolation mode

Prototype:

int32 SMCVectMoveEnd(SMCHANDLE handle);
Parameters:

Handle

Controller handle

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.21. SMCGetVectMoveState

Description:
Get interpolation state

Prototype:

int32 SMCGetVectMoveState(SMCHANDLE handle, uint8 *pState);

Parameters:

Handle

Controller handle

pState

Pointer to a variable that receives interpolation state:

VECTMOVE_STATE_RUNING = 1,

VECTMOVE_STATE_PAUSE = 2,

VECTMOVE_STATE_STOP = 3

Return Value:

Error code (See section 5.3 for the detail)

5.2.4.22. SMCGetVectMoveRemainSpace

Description:

Get remainder space for interpolation

Prototype:

int32 SMCGetVectMoveRemainSpace(SMCHANDLE handle, uint32 *pSpace);

Parameters:

Handle

Controller handle

pSpace

Pointer to a variable that receives the remainder space for interpolation

Return Value:

Error code (See section 5.3 for the detail)

5.2.4.23. SMCVectMoveLine1

Description:

1-axis linear interpolation

Prototype:

int32 SMCVectMoveLine1(SMCHANDLE handle, uint8 iaxis, double Distance, double dspeed, uint8 bIfAbs);

Parameters:

handle

Controller handle

iaxis

Specifies the axis

Distance

Specifies coordinate (unit: mm)

dspeed

Specifies interpolation speed (unit: mm/s)

bIfAbs

Specifies whether coordinate is absolute. If *bIfAbs* is 1, the coordinate is absolute. If *bIfAbs* is 0, the coordinate is relative.

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.24. SMCVectMoveLine2

Description:
2-axes linear interpolation

Prototype:

int32 SMCVectMoveLine2(SMCHANDLE handle, uint8 iaxis1, double Distance1, uint8 iaxis2, double Distance2, double dspeed, uint8 bIfAbs);
Parameters:

handle

Controller handle

iaxis1

Specifies the first axis

Distance1

Specifies the coordinate of the first axis (unit: mm)

iaxis2

Specifies the second axis

Distance2

Specifies the coordinate of the second axis (unit: mm)

dspeed

Specifies interpolation speed (unit: mm/s)

bIfAbs

Specifies whether coordinate is absolute. If *bIfAbs* is 1, the coordinate is absolute. If *bIfAbs* is 0, the coordinate is relative.

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.26. SMCVectMoveLineN

Description:
N-axes linear interpolation

Prototype:

int32 SMCVectMoveLineN(SMCHANDLE handle, uint8 itotalaxis, uint8* piaxisList, double* DistanceList, double dspeed, uint8 bIfAbs);
Parameters:

handle

Controller handle

itotalaxis

Specifies the axis quantity

piaxisList

Pointer to the axes list

DistanceList

Pointer to interpolation coordinate.

dspeed

Specifies interpolation speed (unit: mm/s)

bIfAbs

Specifies whether coordinate is absolute. If *bIfAbs* is 1, the coordinate is absolute. If *bIfAbs* is 0, the coordinate is relative.

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.25. SMCVectMoveMultiLine2

Description:
Multiple 2-axes linear interpolation

Prototype:
int32 SMCVectMoveMultiLine2(SMCHANDLE handle, uint8 iaxis1, uint8 iaxis2, uint16 uiSectes, double* DistanceList, double* dspeedList, uint8 bIfAbs);

Parameters:

handle

Controller handle

iaxis1

Specifies the first axis

iaxis2

Specifies the second axis

uiSectes

Specifies point quantity

DistanceList

Pointer to coordinate array whose size is two times of *uiSectes*. By default the coordinate is in millimeter.

dspeedList

Pointer to speed array whose size is *uiSectes*. By default the speed is in mm/s.

bIfAbs

Specifies whether coordinate is absolute. If *bIfAbs* is 1, the coordinate is absolute. If *bIfAbs* is 0, the coordinate is relative.

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.28. SMCVectMoveMultiLineN

Description:

Multiple N-axes linear interpolation

Prototype:

int32 SMCVectMoveMultiLineN(SMCHANDLE handle, uint8 itotalaxis, uint8* piaxisList, uint16 uiSectes, double*

DistanceList, double* dspeedList, uint8 bIfAbs);

Parameters:

handle

Controller handle

itotalaxis

Specifies axis quantity.

piaxisList

Pointer to axis array.

uiSectes

Specifies point quantity to be interpolated.

DistanceList

Pointer to coordinate array whose size is N times of *uiSectes*. By default the coordinate is in millimeter.

dspeedList

Pointer to speed array whose size is N times of *uiSectes*. By default the speed is in mm/s.

bIfAbs

Specifies whether coordinate is absolute. If *bIfAbs* is 1, the coordinate is absolute. If *bIfAbs* is 0, the coordinate is relative.

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.26. SMCVectMoveArc

Description:
2-axes circular interpolation

Prototype:
int32 SMCVectMoveArc(SMCHANDLE handle, uint8 iaxis1, uint8 iaxis2, double Distance1, double Distance2, double Center1, double Center2, uint8 bIfAnticlockwise, double dspeed, uint8 bIfAbs);

Parameters:

handle

Controller handle

iaxis1

Specifies the first axis

iaxis2

Specifies the second axis

Distance1

Specifies the coordinate of the first axis

Distance2

Specifies the coordinate of the second axis

Center1

Specifies the first axis coordinate of the center

Center2

Specifies the second axis coordinate of the center

bIfAnticlockwise

Specifies whether move direction is anti-clockwise. If *bIfAnticlockwise* is 1, the move direction is anti-clockwise. If *bIfAnticlockwise* is 0, the move direction is clockwise.

dspeed

Specifies interpolation speed

bIfAbs

Specifies whether coordinate is absolute. If *bIfAbs* is 1, the coordinate is absolute. If *bIfAbs* is 0, the coordinate is relative.

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.27. SMCVectMoveSetSpeedLimition

Description:
Set speed limit of the current point

Prototype:
int32 SMCVectMoveSetSpeedLimition(SMCHANDLE handle, double dspeed);

Parameters:

handle

Controller handle

dspeed

Specifies speed limitation. (unit: mm/s)

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.28. SMCGetCurRunVectLength

Description:
Get interpolation distance

Prototype:

int32 SMCGetCurRunVectLength(SMCHANDLE handle, double* pvectlength);
Parameters:

handle

Controller handle

pvectlength

Pointer to a variable that receives the interpolation distance (unit: mm)

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.29. SMCGetCurSpeed

Description:
Get current speed

Prototype:

int32 SMCGetCurSpeed(SMCHANDLE handle, uint8 iaxis, double* pspeed);
Parameters:

handle

Controller handle

iaxis

Specifies the axis

pspeed

Pointer to a variable that receives speed value

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.30. SMCVectMovePause

Description:
Pause interpolation

Prototype:

int32 SMCVectMovePause(SMCHANDLE handle);
Parameters:

handle

Controller handle

Return Value:
Error code (See section 5.3 for the detail)

5.2.4.31. SMCVectMoveStop

Description:
Stop interpolation

Prototype:

int32 SMCVectMoveStop(SMCHANDLE handle);
Parameters:

handle

Controller handle

Return Value:
Error code (See section 5.3 for the detail)

5.2.5. I/O Control

5.2.5.1. SMCWriteLed

Description:
Turn on/off LED

Prototype:

int32 SMCWriteLed(SMCHANDLE handle, uint16 iLedNum, uint8 bifLighten);

Parameters:

handle

Controller handle

iLedNum

Specifies the LED number which is 1 or 2.

bifLighten

Specifies whether turn on or turn off the LED. If ***bifLighten*** is 1, turn on the LED. If ***bifLighten*** is 1, turn off the LED.

Return Value:

Error code (See section 5.3 for the detail)

5.2.5.2. SMCWriteOutBit

Description:

Write to output bit

Prototype:

int32 SMCWriteOutBit(SMCHANDLE handle, uint16 ioNum, uint8 IoState);

Parameters:

handle

Controller handle

ioNum

Output bit address from 1 to 24.

IoState

Specifies the output state: 0 is on and 1 is off.

Return Value:

Error code (See section 5.3 for the detail)

5.2.5.3. SMCReadInBit

Description:

Read input bit

Prototype:

int32 SMCReadInBit(SMCHANDLE handle, uint16 ioNum, uint8* pIoState);

Parameters:

handle

Controller handle

ioNum

Input bit address from 1 to 24.

pIoState

Pointer to a variable that receives the state of the input bit: 0 is on and 1 is off.

Return Value:

Error code (See section 5.3 for the detail)

5.2.5.4. SMCReadOutBit

Description:

Read output bit

Prototype:

int32 SMCReadOutBit(SMCHANDLE handle, uint16 ioNum, uint8* pIoState);

Parameters:

handle

Controller handle

ioNum

Input bit address from 1 to 24.

pIoState

Pointer to a variable that receives the state of the output bit; 0 is on and 1 is off.

Return Value:

Error code (See section 5.3 for the detail)

5.2.5.5. SMCWriteOutPort

Description:

Write output port

Prototype:

int32 SMCWriteOutPort(SMCHANDLE handle, uint32 IoMask, uint32 IoState);

Parameters:

handle

Controller handle

IoMask

Specifies the bit mask that indicates which bit will be changed. Each bit of IoMask represents one digital output. Only the bit with value 0 will be changed..

IoState

Specifies the output state. Each bit of IoSate represents one digital output: 0 is on and 1 is off.

Return Value:

Error code (See section 5.3 for the detail)

5.2.5.6. SMCReadInPort

Description:

Read input port

Prototype:

int32 SMCReadInPort(SMCHANDLE handle, uint32* pIoState);

Parameters:

handle

Controller handle

pIoState

Pointer to a variable that receives the input port state. Each bit of this variable represents one digital input: 0 is on and 1 is off.

Return Value:

Error code (See section 5.3 for the detail)

5.2.5.7. SMCReadOutPort

Description:

Read output port

Prototype:

int32 SMCReadOutPort(SMCHANDLE handle, uint32* pIoState);

Parameters:

handle

Controller handle

pIoState

Pointer to a variable that receives the output port state. Each bit of this variable represents one digital input: 0 is on and 1 is off.

Return Value:

Error code (See section 5.3 for the detail)

5.2.5.8. SMCReadHomeState

Description:

Get HOME signal state

Prototype:

int32 SMCReadHomeState(SMCHANDLE handle, uint8 iaxis, uint8* pIoState);

Parameters:

handle

Controller handle

iaxis

Specifies the axis

pIoState

Pointer to a variable that receives HOME signal state: 0 is on and 1 is off.

Return Value:

Error code (See section 5.3 for the detail)

5.2.5.10. SMCReadEMGState

Description:

Get EMG signal state

Prototype:

int32 SMCReadEMGState(SMCHANDLE handle, uint8* pIoState);

Parameters:

handle

Controller handle

pIoState

Pointer to a variable that receives EMG(Emergency Stop) signal state: 0 is on and 1 is off.

Return Value:
Error code (See section 5.3 for the detail)

5.2.5.11. SMCReadHandWheelStates

Description:
Get HandWheel(MPG) state

Prototype:

int32 SMCReadHandWheelStates(SMCHANDLE handle, uint8 iaxis, uint8* pIoAState, uint8* pIoBState);
Parameters:

handle

Controller handle

iaxis

Specifies the axis.

pIoAState

Pointers to a variable that receives phase A state: 0 is on and 1 is off.

pIoBState

Pointers to a variable that receives phase B state: 0 is on and 1 is off.

Return Value:
Error code (See section 5.3 for the detail)

5.2.5.12. SMCReadElStates

Description:
Get EL(end limit) signal state

Prototype:

int32 SMCReadElStates(SMCHANDLE handle, uint8 iaxis, uint8* pElDecState, uint8* pElPlusState);
Parameters:

handle

Controller handle

iaxis

Specifies the axis

pElDecState

Pointer to a variable that receives the state of positive limit input: 0 is on and 1 is off.

pElPlusState

Pointer to a variable that receives the state of negitive limit input: 0 is on and 1 is off.

Return Value:
Error code (See section 5.3 for the detail)

5.2.5.13. SMCReadSdStates

Description:
Get SD (start deceleration) signal state

Prototype:

int32 SMCReadSdStates(SMCHANDLE handle, uint8 iaxis, uint8* pIoState);
Parameters:

handle

Controller handle

pIoState

Pointer to a variable that receives the SD (start deceleration) signal state: 0 is on and 1 is off.

Return Value:
Error code (See section 5.3 for the detail)

5.2.5.14. SMCReadInpStates

Description:
Get INP(in-position) signal state

Prototype:

int32 SMCReadInpStates(SMCHANDLE handle, uint8 iaxis, uint8* pIoState);
Parameters:

handle

Controller handle

pIoState

Pointer to a variable that receives the INP (in-position) signal state: 0 is on and 1 is off.

Return Value:
Error code (See section 5.3 for the detail)

5.2.5.15. SMCReadAxisStates

Description:
Get axis status

Prototype:

int32 SMCReadAxisStates(SMCHANDLE handle, uint8 iaxis, struct_AxisStates* pAxisState);
Parameters:

handle

Controller handle

iaxis

Specifies the axis

pAxisState

Pointer to a variable that receives axis state.

Return Value:
Error code (See section 5.3 for the detail)

5.2.5.16. SMCWritePwmDuty

Description:
Set PWM duty-cycle

Prototype:

int32 SMCWritePwmDuty(SMCHANDLE handle, uint8 ichannel, float fDuty);
Parameters:

handle

Controller handle

ichannel

Specifies the PWM output channel. It is from 1 to 2.

fDuty

Specifies the ducy-cycle which is from 0.0 to 1.0.

Return Value:
Error code (See section 5.3 for the detail)

5.2.5.17. SMCWritePwmFreqency

Description:
Set PWM frequency

Prototype:

int32 SMCWritePwmFreqency(SMCHANDLE handle, uint8 ichannel, float fFre);
Parameters:

handle

Controller handle

ichannel

Specifies the PWM output channel. It is from 1 to 2.

fDuty

Specifies the frequency which is from 1 to 10000000.

Return Value:
Error code (See section 5.3 for the detail)

5.2.5.18. SMCWriteDaOut

Description:
Set DA output

Prototype:

int32 SMCWriteDaOut(SMCHANDLE handle, uint8 ichannel, float fLevel);
Parameters:

handle

Controller handle

ichannel

Specifies the PWM output channel. It is from 1 to 2.

fLevel

Specifies the DA output voltage which is from 0.0 to 5.0.

Return Value:
Error code (See section 5.3 for the detail)

5.2.5.19. SMCReadPwmDuty

Description:
Get PWM ducy-cycle

Prototype:

int32 SMCReadPwmDuty(SMCHANDLE handle, uint8 ichannel, float* fDuty);
Parameters:

handle

Controller handle

ichannel

Specifies the PWM output channel. It is from 1 to 2.

fDuty

Pointer to a variable which receives the PWM duty-cycle. The return value is from 0.0 to 1.0.

Return Value:
Error code (See section 5.3 for the detail)

5.2.5.20. SMCReadPwmFreqency

Description:
Get PWM frequency

Prototype:

int32 SMCReadPwmFreqency(SMCHANDLE handle, uint8 ichannel, float* fFre);
Parameters:

handle

Controller handle

ichannel

Specifies the PWM output channel. It is from 1 to 2.

fDuty

Pointer to a variable which receives the PWM frequency. The return value is from 1 to 10000000.

Return Value:
Error code (See section 5.3 for the detail)

5.2.5.21. SMCReadDaOut

Description:
Get DA Output

Prototype:

int32 SMCReadDaOut(SMCHANDLE handle, uint8 ichannel, float* fLevel);

Parameters:

handle

Controller handle

ichannel

Specifies the PWM output channel. It is from 1 to 2.

fLevel

Pointer to a variable that receives the DA output voltage which is from 0.0 to 5.0.

Return Value:
Error code (See section 5.3 for the detail)

5.2.6. Parameters

5.2.6.1. SMCCommand

Description:
Set parameter via string command

Prototype:

int32 SMCCommand(SMCHANDLE handle, const char* pszCommand, char* psResponse, uint32 uiResponseLength);

Parameters:

handle

Controller handle

pszCommand

Command string

psResponse

Pointer to a string that receives the response string.

uiResponseLength

Spcifies the maximum size of response string.

Return Value:
Error code (See section 5.3 for the detail)

Examples:

char stringresponse[1024];

SMCCommand(handle, “UnitPulses1=?”, stringresponse, 1024); // Get pulses count per one unit

5.2.6.2. SMCBurnSetting

Description:
Burn to FLASH memory

Prototype:

int32 SMCBurnSetting(SMCHANDLE handle);
Parameters:

handle

Controller handle

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.3. SMCSetIpAddr

Description:
Modify IP address

Prototype:

int32 SMCSetIpAddr(SMCHANDLE handle, const char* sIpAddr, const char* sGateAddr, const char* sMask, uint8 bifdhcp);

Parameters:

handle

Controller handle

sIpAddr

Pointer a char array that receives IP address

sGateAddr

Pointer a array that receives gate address.

sMask

Pointer a array that receives Ip Mask .

bifdhcp

If bifdhcp is 1, DHCP will be used.

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.4. SMCGetIpAddr

Description:
Get IP address

Prototype:

int32 SMCGetIpAddr(SMCHANDLE handle, char* sIpAddr, char* sGateAddr, char* sMask, uint8 *pbifdhcp);
Parameters:

handle

Controller handle

sIpAddr

Pointer to a char array that receives IP address.

sGateAddr

Pointer to a char array that receives gate address.

sMask

Pointer to an char array that receives IP mask.

bifdhcp

Pointer to a variable that indicates whether DHCP is used.

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.5. SMCGetCurIpAddr

Description:
Get current IP address The IP address may be changed if DHCP is activated.

Prototype:

int32 SMCGetCurIpAddr(SMCHANDLE handle, char* sIpAddr);
Parameters:

handle

Controller handle

sIpAddr

Pointer to a char array that receives IP address.

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.6. SMCSetZeroSpeed

Description:
Set homing speed

Prototype:

int32 SMCSetZeroSpeed(SMCHANDLE handle, uint8 iaxis, uint32 uiSpeed);
Parameters:

handle

Controller handle

iaxis

Specifies the axis.

uiSpeed

Specifies homing speed. (unit: mm/s)

Return Value:
Error code (See section 5.3 for the detail)

Example:

SMCSetZeroSpeed(handle, 0, 100); // Set homing speed of the first axis 100 mm/s.

5.2.6.7. SMCGetZeroSpeed

Description:
Get homing speed

Prototype:

int32 SMCGetZeroSpeed(SMCHANDLE handle, uint8 iaxis, uint32* puiSpeed);
Parameters:

handle

Controller handle

puiSpeed

Pointer to variable that receives homing speed. (unit: mm/s)

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.8. SMCSetLocateSpeed

Description:
Set single axis move speed

Prototype:

int32 SMCSetLocateSpeed(SMCHANDLE handle, uint8 iaxis, uint32 uiSpeed);
Parameters:

handle

Controller handle

iaxis

Specifies the axis.

uiSpeed

Specifies the speed. (unit: mm/s)

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.9.SMCGetLocateSpeed

Description:
Get single axis move speed

Prototype:

int32 SMCGetLocateSpeed(SMCHANDLE handle, uint8 iaxis, uint32* puiSpeed);
Parameters:

handle

Controller handle

iaxis

Specifies the axis.

uiSpeed

Specifies the speed. (unit: mm/s)

puiSpeed

Pointer to a variable that receives the move speed for single axis. (unit: mm/s)

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.10. SMCSetLocateStartSpeed

Description:
Set single axis start speed

Prototype:

int32 SMCSetLocateStartSpeed(SMCHANDLE handle, uint8 iaxis, uint32 uiSpeed);
Parameters:

handle

Controller handle

iaxis

Specifies the axis.

uiSpeed

Specifies the speed.

Return Value:

Error code (See section 5.3 for the detail)

5.2.6.11. SMCGetLocateStartSpeed

Description:
Get single axis start speed

Prototype:

int32 SMCGetLocateStartSpeed(SMCHANDLE handle, uint8 iaxis, uint32* puiSpeed);
Parameters:

handle

Controller handle

iaxis

Specifies the axis.

puiSpeed

Pointer to a variable that receives the speed.

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.12. SMCSetLocateAcceleration

Description:
Set single axis acceleration

Prototype:

int32 SMCSetLocateAcceleration(SMCHANDLE handle, uint8 iaxis, uint32 uiValue);
Parameters:

handle

Controller handle

iaxis

Specifies the axis.

uiValue

Specifies the acceleration for single axis move. (unit: mm/s^2)

Return Value:

Error code (See section 5.3 for the detail)

5.2.6.13. SMCGetLocateAcceleration

Description:
Get single axis acceleration

Prototype:

int32 SMCGetLocateAcceleration(SMCHANDLE handle, uint8 iaxis, uint32* puiValue);
Parameters:

handle

Controller handle

iaxis

Specifies the axis.

puiValue

Pointer to a variable that receives the acceleration for single axis acceleration. (unit: mm/s^2)

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.14. SMCSetLocateDeceleration

Description:
Set single axis deceleration

Prototype:

int32 SMCSetLocateDeceleration(SMCHANDLE handle, uint8 iaxis, uint32 uiValue);
Parameters:

handle

Controller handle

iaxis

Specifies the axis.

uiValue

Specifies the deceleration for single axis move. (unit: mm/s^2)

Return Value:

Error code (See section 5.3 for the detail)

5.2.6.15. SMCGetLocateDeceleration

Description:
Get single axis deceleration

Prototype:

int32 SMCGetLocateDeceleration(SMCHANDLE handle, uint8 iaxis, uint32* puiValue);
Parameters:

handle

Controller handle

iaxis

Specifies the axis.

puiValue

Pointer to a variable that receives the deceleration for single axis acceleration. (unit: mm/s^2)

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.16. SMCSetUnitPulses

Description:
Set pulse count per unit

Prototype:

int32 SMCSetUnitPulses(SMCHANDLE handle, uint8 iaxis, uint32 uiValue);
Parameters:

handle

Controller handle

iaxis

Specifies the axis.

uiValue

Specifies the pulse count per one user unit.

Return Value:

Error code (See section 5.3 for the detail)

5.2.6.17. SMCGetUnitPulses

Description:
Get pulse count per unit

Prototype:

int32 SMCGetUnitPulses(SMCHANDLE handle, uint8 iaxis, uint32* puiValue);
Parameters:

handle

Controller handle

iaxis

Specifies the axis.

puiValue

Pointer a variable that receives pulses count per one user unit. .

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.18. SMCSetVectStartSpeed

Description:
Set interpolation start speed

Prototype:

int32 SMCSetVectStartSpeed(SMCHANDLE handle, uint32 uiValue);
Parameters:

handle

Controller handle

uiValue

Specifies the start speed for interpolation.

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.19. SMCGetVectStartSpeed

Description:
Get interpolation start speed

int32 SMCGetVectStartSpeed(SMCHANDLE handle, uint32* puiValue);

puiValue

Pointer to a variable that receives the start speed for interpolation.

5.2.6.20. SMCSetVectSpeed

Description:
Set interpolation speed

Prototype:

int32 SMCSetVectSpeed(SMCHANDLE handle, uint32 uiValue);
Parameters:

handle

Controller handle

uiValue

Specifies the interpolation speed. (unit: mm/s)

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.21. SMCGetVectSpeed

Description:
Get interpolation speed

Prototype:

int32 SMCGetVectSpeed(SMCHANDLE handle, uint32* puiValue);
Parameters:

handle

Controller handle

puiValue

Pointer to a variable that receives the interpolation speed.(mm/s)

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.22. SMCSetVectAcceleration

Description:
Set interpolation acceleration

Prototype:

int32 SMCSetVectAcceleration(SMCHANDLE handle, uint32 uiValue);
Parameters:

handle

Controller handle

uiValue

Specifies the acceleration for interpolation. (mm/s^2)

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.23. SMCGetVectAcceleration

Description:
Get interpolation acceleration

Prototype:

int32 SMCGetVectAcceleration(SMCHANDLE handle, uint32* puiValue);
Parameters:

handle

Controller handle

puiValue

Pointer a variable that receives the interpolation acceleration. (mm/s^2)

Return Value:
Error code (See section 5.3 for the detail)

5.2.6.24. SMCSetVectDeceleration

Description:
Set interpolation deceleration

Prototype:

int32 SMCSetVectDeceleration(SMCHANDLE handle, uint32 uiValue);

Parameters:

handle

Controller handle

uiValue

Specifies the interpolation deceleration. (unit: mm/s^2)

Return Value:

Error code (See section 5.3 for the detail)

5.2.6.25. SMCGetVectDeceleration

Description:

Get interpolation deceleration

Prototype:

int32 SMCGetVectDeceleration(SMCHANDLE handle, uint32* puiValue);

Parameters:

handle

Controller handle

puiValue

Pointers a variable that receives the deceleration for interpolation.(unit: mm/s^2)

Return Value:

Error code (See section 5.3 for the detail)

5.2.7. Password

5.2.7.1. SMCUpPasswordFile

Description:

Upload password file to PC file

Prototype:

int32 SMCUpPasswordFile(SMCHANDLE handle, const char* pfilename);

Parameters:

handle

Controller handle

pfilename

Pointer a char array that the name of PC file.

Return Value:
Error code (See section 5.3 for the detail)

5.2.7.2. SMCUpPasswordFileToMem

Description:

Upload password file to PC memory

Prototype:
int32 SMCUpPasswordFileToMem(SMCHANDLE handle, char* pbuffer, uint32 buffsize, uint32* puifilesize);

Parameters:

handle

Controller handle

pbuffer

Buffer address in PC memory

buffsize

Maximum buffer size.

puifilesize

Pointer to variable that receives the password file size.

Return Value:
Error code (See section 5.3 for the detail)

5.2.7.3. SMCDownPasswordFile

Description:

Download password file from PC file

Prototype:
int32 SMCDownPasswordFile(SMCHANDLE handle, const char* pfilename);

Parameters:

handle

Controller handle

pfilename

Specifies the PC file name to be downloaded.

Return Value:
Error code (See section 5.3 for the detail)

5.2.7.4. SMCDownMemPasswordFile

Description:

Download password file from memory

Prototype:
int32 SMCDownMemPasswordFile(SMCHANDLE handle, const char* pbuffer, uint32 buffsize);

Parameters:

handle

Controller handle

pbuffer

Buffer address in PC memory

buffsize

Maximum buffer size.

Return Value:
Error code (See section 5.3 for the detail)

5.2.7.5. SMCEnterSetPassword

Description:

Input parameter password before changing the settings

Prototype:
int32 SMCEnterSetPassword(SMCHANDLE handle, uint32 uipassword);

Parameters:

handle

Controller handle

uipassword

Specifies the password.

Return Value:
Error code (See section 5.3 for the detail)

5.2.7.6. SMCEnterEditPassword

Description:

Input edit password before editing the program file

Prototype:
int32 SMCEnterEditPassword(SMCHANDLE handle, uint32 uipassword);

Parameters:

handle

Controller handle

uipassword

Specifies the password.

Return Value:
Error code (See section 5.3 for the detail)

5.2.7.7. SMCEnterSuperPassword

Description:

Input supper password

Prototype:
int32 SMCEnterSuperPassword(SMCHANDLE handle, uint32 uipassword);

Parameters:

handle

Controller handle

uipassword

Specifies the supper password

Return Value:
Error code (See section 5.3 for the detail)

5.2.7.8. SMCEnterTimePassword

Description:

Input evaluation time-out password

Prototype:
int32 SMCEnterTimePassword(SMCHANDLE handle, uint32 uipassword);

Parameters:

handle

Controller handle

uipassword

Specifies the evaluation time out password.

Return Value:
Error code (See section 5.3 for the detail)

5.2.7.9. SMCClearEnteredPassword

Description:

Clear password**.** Need to input the password again.

Prototype:
int32 SMCClearEnteredPassword(SMCHANDLE handle);

Parameters:

handle

Controller handle

Return Value:
Error code (See section 5.3 for the detail)

5.2.7.10. SMCModifySetPassword

Description:

Modify password for changing settings

Prototype:
int32 SMCModifySetPassword(SMCHANDLE handle, uint32 uipassword);

Parameters:

handle

Controller handle

uipassword

Specifies the setting password

Return Value:
Error code (See section 5.3 for the detail)

5.2.7.11. SMCModifyEditPassword

Description:

Modify editing password of program file

Prototype:
int32 SMCModifyEditPassword(SMCHANDLE handle, uint32 uipassword);

Parameters:

handle

Controller handle

uipassword

Specifies the password

Return Value:
Error code (See section 5.3 for the detail)

5.2.7.12. SMCModifySuperPassword

Description:

Modify supper password

Prototype:
int32 SMCModifySuperPassword(SMCHANDLE handle, uint32 uipassword);

Parameters:

handle

Controller handle

uipassword

Specifies the password.

Return Value:
Error code (See section 5.3 for the detail)

5.2.7.13. SMCGetTrialCondition

Description:

Check if evaluation time is out

Prototype:
int32 SMCGetTrialCondition(SMCHANDLE handle, uint32* pRunHours, uint16* pbifTimeLocked, uint16* pAlreadyEnterdTimePasswordNum);

Parameters:

handle

Controller handle

pRunHours

Pointer to a variable that receives the system escape time. (unit: hour)

pbifTimeLocked

Pointer a variable that indicates whether time is out. The controller will be locked and it is required to input the password again.

pAlreadyEnterdTimePasswordNum

Pointer to a variable that indicates how many passwords you have input.

Return Value:

Error code (See section 5.3 for the detail)

5.2.8. Other

5.2.8.1. SMCGetProgress

Description:

Get progress of downloading file

Prototype:
float SMCGetProgress(SMCHANDLE handle);

Parameters:

handle

Controller handle

Return Value:

A float value from 0.0 to 1.0 that indicates the current progress of downloading file to controller

5.2.8.2. SMCGetState

Description:

Get controller state

Prototype:
int32 SMCGetState(SMCHANDLE handle,uint8 *pstate);

Parameters:

handle

Controller handle

pstate

Pointer a unsigned char variable that receives the controller state shown as follows:

```
#define SYS_STATE_IDLE                1     //  Standby
#define SYS_STATE_GRUNNING            3     //  Running
#define SYS_STATE_MANUALING           4     //  Manual
#define SYS_STATE_PAUSE               5     //  Pause
#define SYS_STATE_GEDIT               6     //  Editing
#define SYS_STATE_SETTING             7     //  Configuring
#define SYS_STATE_TEST                8     //  Testing
#define SYS_STATE_GFILEREVIEW         9     //  Viewing File
#define SYS_STATE_UDISK               10    //  U-disk Operation
#define SYS_STATE_GTEACHING           11    //  Teaching
#define SYS_STATE_CANNOT_CONNECT      50    //  No Connection
```

Return Value:

Error code (See section 5.3 for the detail)

5.2.8.3. SMCGetAxises

Description:

Get controllable axes

Prototype:
uint8 SMCGetAxises(SMCHANDLE handle);

Parameters:

handle

Controller handle

Return Value:

This function returns the controllable axes. If the return value is 0, it indicates there is error happended.

5.2.8.4. SMCGetSoftwareId

Description:

Get controller software(firmware) ID.

Prototype:
int32 SMCGetSoftwareId(SMCHANDLE handle, uint16 *pId);

Parameters:

handle

Controller handle

pId

Pointer to a variable that receives controller software (firmware) ID.

Return Value:

Error code (See section 5.3 for the detail)

5.2.8.5. SMCGetHardwareId

Description:

Get controller hardware version

Prototype:
int32 SMCGetHardwareId(SMCHANDLE handle,uint16 *pId);

Parameters:

handle

Controller handle

pId

Pointer to a variable that receives controller hardware ID.

Return Value:

Error code (See section 5.3 for the detail)

5.2.8.6. SMCGetSoftwareVersion

Description:

Get controller software(firmware) version

Prototype:
int32 SMCGetSoftwareVersion(SMCHANDLE handle,uint32 *pVersion);

Parameters:

handle

Controller handle

pVersion

Pointer to a variable that receives controller software(firmware) version.

Return Value:

Error code (See section 5.3 for the detail)

5.2.8.7. SMCModbus_Set0x

Description:

Set MODBUS bit register

Prototype:
uint32 SMCModbus_Set0x(SMCHANDLE handle, uint16 start, uint16 inum, uint8* pdata);

Parameters:

handle

Controller handle

start

0-based Starting address of bit register. Note: it is 1-based addressing for touch screen.

inum

Specifies the bit register number to be set.

pdata

Pointer to buffer that specifies the bit data. A logic “1” in the bit position turns on the corresponding register. A logic “0” in the bit position turns off the corresponding register.

Return Value:

Error code (See section 5.3 for the detail)

5.2.8.8. SMCModbus_Get0x

Description:

Get MODBUS bit register

Prototype:
uint32 SMCModbus Get0x(SMCHANDLE handle, uint16 start, uint16 inum, uint8* pdata);

Parameters:

handle

Controller handle

start

0-based Starting address of bit register. Note: it is 1-based addressing for touch screen.

inum

Specifies bit register number.

pdata

Pointer to buffer that receives the bit data. A logic “1” in the bit position indicates the corresponding register is on. A logic “0” in the bit position indicates the corresponding register is off.

Return Value:

Error code (See section 5.3 for the detail)

5.2.8.9. SMCModbus_Get4x

Description:

Set MODBUS word register

Prototype:
uint32 SMCModbus_Get4x(SMCHANDLE handle, uint16 start, uint16 inum, uint16* pdata);

Parameters:

handle

Controller handle

start

0-based Starting address of word register. Note: it is 1-based addressing for touch screen.

inum

Specifies word register number.

pdata

Pointer to buffer that receives the word data. Each register contains two bytes or 16 bits data.

Return Value:

Error code (See section 5.3 for the detail)

5.2.8.10. SMCModbus_Set4x

Description:

Get MODBUS word register

Prototype:
uint32 SMCModbus_Set4x(SMCHANDLE handle, uint16 start, uint16 inum, uint16* pdata);

Parameters:

handle

Controller handle

start

0-based Starting address of word register. Note: it is 1-based addressing for touch screen.

inum

Specifies the word register number to be set.

pdata

Pointer to buffer that specifies the word data. Each register contains two bytes or 16 bits data.

Return Value:

Error code (See section 5.3 for the detail)

5.2.8.11. SMCGetErrcodeDescription

Description:

Get description for error code.

Prototype:
char* SMCGetErrcodeDescription(int32 ierrcode);

Parameters:

handle

Controller handle

Return Value:

Pointer to a char array that receives the description for error code. A NULL value indicates there is no description.

5.3 Error Code

Error Code	Name	Description
0	ERR_NOERR	No error.
1	ERRCODE_UNKNOWN	Unknown error.
2	ERRCODE_PARAERR	Parameter error.
3	ERRCODE_TIMEOUT	Time out.
4	ERRCODE_CONTROLLERBUSY	Controller is busy.
5	ERRCODE_CONNECT_TOOMANY	Too many user connecting to controller.
6	ERRCODE_OS_ERR	Operating system error.
7	ERRCODE_CANNOT_OPEN_COM	Can not open COM port.
8	ERRCODE_CANNOT_CONNECTETH	Can not connect to Ethernet.
9	ERRCODE_HANDLEERR	Invalid controller handle.
10	ERRCODE_SENDERR	Error in sending.
11	ERRCODE_GFILE_ERR	Syntax error in G code program file

12	ERRCODE_FIRMWAREERR	Invalid firmware file
13	ERRCODE_FILENAME_TOOLONG	File name is too long
14	ERRCODE_FIRMWAR_MISMATCH	Unmatched firmware version
15	ERRCODE_BUFFER_TOO_SMALL	Buffer size is too small
16	ERRCODE_NEED_PASSWORD	Need to input password first
17	ERRCODE_PASSWORD_ENTER_TOOFAST	Input of password is too fast
100	ERRCODE_GET_LENGTH_ERR	Wrong data length of receiving package
1000	ERRCODE_COMPILE_OFFSET	Error in compiling G code program file
10000	ERRCODE_CONTROLLERERR_OFFSET	Error in the controller which is based on this offset

Chapter 6 Leadshine G Code Detail

G code is the common name of the CNC (computer numerical control) program language. Leadshine G code coincides with ISO-1056-1975E and adds some special G code and M code such as condition execution, loop control, subprogram and multi-task in order to increase the flexibility of programming.

6.1 Coordinates System

SMC6480 adopts the right-handed rectangular Cartesian coordinate system for the G-code programming. See Figure 6-1 for the illustration of right-handed Cartesian Coordinates. The system coordinates is the exclusive coordinates inside SMC6480. It is usually attached to the machine shelf. The origin is usually the home switch in the machine.

6.2 Absolute and Relative Coordinates

The motion in SMC6480 can be in absolute or relative mode, see figure 6-2. In relative mode, a curve is defined by a couple of points and changing of one point will affect al the points after it. In absolute mode, a curve is defined by a couple points and changing of one point will not affect the points after it.

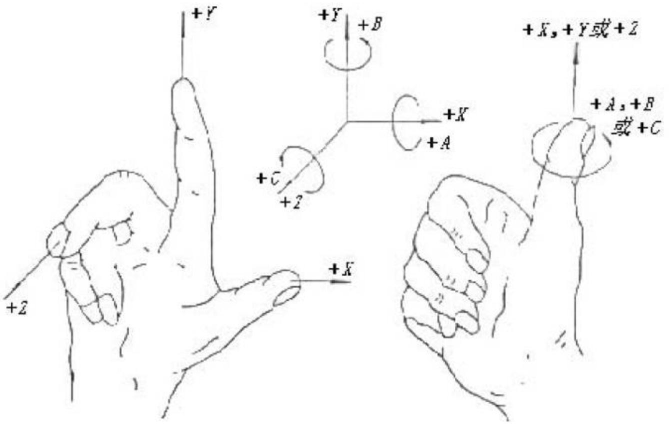


Figure 6-1: Right-handed Cartesian Coordinates

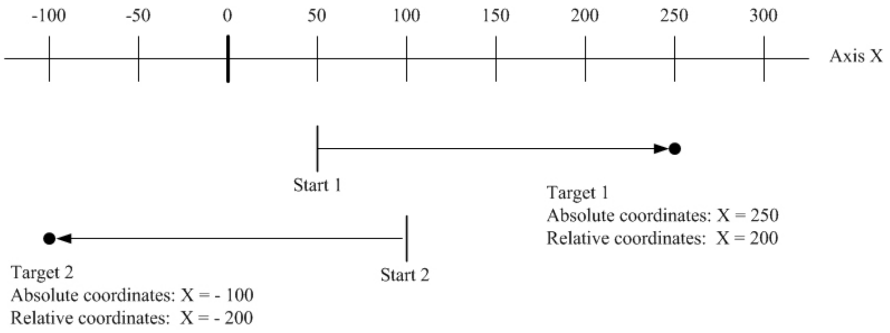

-100
-50
0
50
100
150
200
250
300
Axis X
Start 1
Target 1
Absolute coordinates: X = 250
Relative coordinates: X = 200
Start 2
Target 2
Absolute coordinates: X = - 100
Relative coordinates: X = - 200

Figure 6-2: Absolute and relative coordinates

6.3 Leadshine G Code List

No.	G-code	Description
1	G00	Rapid Positioning
2	G01	Linear Interpolation
3	G02	Clockwise Circular Interpolation
4	G03	Counter Clockwise Circular Interpolation
5	G04	Delay(Unit: ms)
6	G05	Pass point of Circular Interpolation
7	G06	End point of Circular Interpolation
8	G26	Home Move
9	G28	Move to Workpiece Zero Point
10	G53	Change to Mechanical Coordinates
11	G54	Change to Workpiece Coordinates
12	G90	Start Absolute Coordinates
13	G91	Start Relative Coordinates
14	G92	Reposition Coordinates
15	F	Velocity Percent
16	M00	Program Pause
17	M02	Program End.
18	M07	Output 1 ON
19	M08	Output 1 OFF
20	M09	Output 2 ON
21	M10	Output 2 OFF
22	M11	Output 3 ON
23	M12	Output 3 OFF
24	M30	Program End and Loops Continuously
25	M80	Set Output On
26	M81	Set Output OFF
27	M82	Pauses until Input ON
28	M83	Pauses until Input OFF
29	M84	Start Continuous Movement
30	M85	Stop Continuous Movement
31	M86	Increase Variable Value
32	M87	Set variable Value
33	M89	Pause until Pass the Point
34	M90	End Sub-loop
35	M91	Start sub-loop
36	M92	Modify the work piece coordinates
37	M94	Jump Depends on Conditional Variable
38	M95	Unconditional Jump to line No.

39	M96	Call sub-program depends on Conditional Variable
40	M97	Simultaneous Start of Multiple Tasks
41	M98	Go to sub-program
42	M99	Return to Main Program

6.4 Leadshine G Code Detail

6.4.1. G00 – Rapid Positioning

Parameters: Xm Yn Zi Uj (You can select any one of them)

Functions: Rapid position

Parameters Detail:

Xm	Optional. Move axis X to coordinate m. Unit: mm
Yn	Optional. Move axis Y to coordinate n. Unit: mm
Zi	Optional. Move axis Z to coordinate i. Unit: mm
Uj	Optional. Move axis U to coordinate j Unit: mm

Example: N00 G00 X100 Y100 Z100 U100

6.4.2. G01 – Linear Interpolation

Parameters: Xm Yn Zi Uj Fy

Functions: Linear interpolation from the current point to the target.

Parameters Detail:

Xm	Optional. Target position **m** of axis X. Unit: mm
Yn	Optional. Target position **n** of axis X. Unit: mm
Zi	Optional. Target position **i** of axis X. Unit: mm
Uj	Optional. Target position j of axis X. Unit: mm
Fy	Optional. Specifies the interpolation rate. Unit: %

Example: N00 G01 X100 Y100

6.4.3. G02 – Clockwise Circular Interpolation

Parameters: Xm Yn Zi Uj Rx Fy

Note 1: Only two parameters are accepted when doing circular interpolations. If three parameters are specified, the first and second axis does circular interpolation, while the third axis make linear move.

Note 2: The arc angle is greater than 180 degree when Rx<0 while the arc angle is less than 180 degree when Rx>0.
Note 3: One G02/G03 code can not draw a full circle which requires two G02/G03 line.

Functions: Clockwise circular interpolation

Parameters Detail:

Xm	Optional, Target position m of axis X, unit: mm.
Yn	Optional, Target position n of axis Y, unit: mm.
Zi	Optional, Target position i of axis Z, unit: mm.
Uj	Optional, Target position j of axis U, unit: mm.
Rx	Necessary, Circular Radius, unit: mm.
Fy	Optional, Velocity percentage, unit:%

Example: N00 G02 X100 Y0 R50 F50 ; Draw an arc from the current point to (100, 0) with radius 50 and 50% interpolation speed

6.4.4. G03 - Counter Clockwise Circular Interpolation

Parameters: Xm Yn Zi Uj

Note 1: Only two parameters are accepted when doing circular interpolations. If three parameters are specified, the first and second axis does circular interpolation, while the third axis make linear move.
Note 2: The arc angle is greater than 180 degree when Rx<0 while the arc angle is less than 180 degree when Rx>0.
Note 3: One G02/G03 code can not draw a full circle which requires two G02/G03 line.

Functions: Counter Clockwise Circular Interpolation

Parameters Detail:

Xm	Optional, Target position m of axis X, unit: mm.
Yn	Optional, Target position n of axis Y, unit: mm.
Zi	Optional, Target position i of axis Z, unit: mm.
Uj	Optional, Target position j of axis U, unit: mm.
Rx	Necessary, Circular Radius, unit: mm.
Fy	Optional, Velocity percentage, unit:%

Example: N00 G03 X0 Y-100 R50 F50 ; Draw an arc from the current point to (0, -100) with radius 50 and 50% interpolation speed

6.4.5. G04 - Delay

Parameters: Pm (Pm must be specified)

Functions: Delay (Unit: ms)

Parameters Detail:

Pm	Necessary, Delay time, unit: ms, 0~9999.

Example: N00 G04 P500 ; Delay 500ms

6.4.6. G05 - Pass point of Circular Interpolation

Parameters: Xm Yn Zi Uj

Note: Only two parameters are accepted when doing circular interpolations. If three parameters are specified, the first and second axis does circular interpolation, while the third axis make linear move.

Functions:

Pass point of circular interpolation

Parameters Detail:

Xm	Optional, pass point position m of axis X, unit: mm.
Yn	Optional, pass point position n of axis Y, unit: mm.
Zi	Optional, pass point position i of axis Z, unit: mm.
Uj	Optional, pass point position j of axis U, unit: mm.

Example: N00 G05 X45 Y-63

6.4.7. G06 - End point of Circular Interpolation

Parameters: Xm Yn Zi Uj

Functions: End point of circular interpolation

Note1: Only two parameters are accepted when doing circular interpolations. If three parameters are specified, the first and second axis does circular interpolation, while the third axis make linear move.

Parameters Detail:

Xm	Optional, end point position m of axis X, unit: mm.
Yn	Optional, end point position n of axis Y, unit: mm.
Zi	Optional, end point position i of axis Z, unit: mm.
Uj	Optional, end point position j of axis U, unit: mm.

6.4.8. G26 - Home Move

Parameters: X Y Z U

Functions: Home move

Parameters Detail: You can choose any one of X, Y, Z and U.

Example: N00 G26 X Y Z U ; Home axis X, axis Y, axis Z and axis U

6.4.9. G28 - Move to Workpiece Zero Point

Parameters: Xm Yn Zi Uj

Functions: Move to workpiece zero point

Parameters Detail: You can choose any one of X, Y, Z and U.

Example: N00 G28 X Y Z U

6.4.10. G53 - Change to Mechanical Coordinates

Functions: Change to mechanical coordinates

Example: N00 G53

6.4.11. G54 - Change to Workpiece Coordinates

Functions: Change to workpiece coordinates

Example: N00 G54

6.4.12. G90 - Start Absolute Coordinates

Functions: Start absolute coordinates

Example: N00 G90

6.4.13. G91 - Start Relative Coordinates

Functions: Start relative coordinates

Example: N00 G91

6.4.14. G92 - Reposition Coordinates

Parameters: Xm Yn Zi Uj

Functions: Reposition coordinates

Parameters Detail:

Xm	Optional, new coordinates m of axis X, unit: mm.
Yn	Optional, new coordinates n of axis Y, unit: mm
Zi	Optional, new coordinates i of axis Z, unit: mm
Uj	Optional, new coordinates j of axis U, unit: mm

Example: G92 X100 Y500 Z1000 ; Set the current coordinates to (100, 500, 1000)

6.4.15. F - Velocity Percent

Parameters: Fm

Functions: Set velocity Percentage

Parameters Detail:

Fm	Necessary, velocity percentage.

Example: N00 F52 ; Set 52% velocity

6.4.16. M00 - Program Pause

Functions: Pause program

Example: N00 M00

6.4.17. M02 - Program End

Functions: End program

Example: N100 M02

6.4.18. M07 - Output 1 ON

Functions: Turn on digital output 1

Example: N00 M07

6.4.19. M08 - Output 1 OFF

Functions: Turn off digital output 1

Example: N00 M08

6.4.20. M09 - Output 2 ON

Functions: Turn on digital output 2

Example: N00 M09

6.4.21. M10 - Output 2 OFF

Functions: Turn off digital output 2

Example: N00 M10

:

6.4.22. M11 - Output 3 ON

Functions: Turn on digital output 3

Example: N00 M11

6.4.23. M12 - Output 3 OFF

Functions: Turn off digital output 3

Example: N00 M12

6.4.24. M30 - Program End and Loops Continuously

Functions: Program end and loops continuously. You can only stop the execution by emergency stop.

Example: N00 M30

6.4.25. M80 - Set Output On

Parameters: Sm

Functions: Set digital output Sm on.

Parameters Detail:

Sm	Necessary, specify the digital output to be turned on.

Example: N00 M80 S5 ; Turn on digital output S5

6.4.26. M81 - Set Output OFF

Parameters: Sm

Functions: Set digital output Sm off.

Parameters Detail:

Sm	Necessary, specify the digital output to be turned off.

Example: N00 M81 S5 ; Turn off digital output 5

6.4.27. M82 - Pauses until Input ON

Parameters: Sm

Functions: Pauses program until the specified input is on

Parameters Detail:

Sm	Necessary, specify the digital input

Example: N00 M82 S5 ; Wait until the digital input 5 is on

6.4.28. M83 - Pauses until Input OFF

Parameters: Sm

Functions: Pauses program until the specified input is off

Parameters Detail:

Sm	Necessary, specify the digital input

Example: N00 M82 S5 ; Wait until the digital input 5 is off

6.4.29. M84 - Start Continuous Movement

Parameters: Xm Yn Zi Uj

Functions: Start continuous movement. You can select any one of Xm, Yn, Zi and Uj. The program does not wait the movement to finish but start the next statement

Parameters Detail:

Xm	Optional, m specifies move direction of axis X. If **m** is positive, the move direction is positive, If **m** is negative, the move direction is negative.
Yn	Optional, m specifies move direction of axis X. If **n** is positive, the move direction is positive, If **n** is negative, the move direction is negative.
Zi	Optional, m specifies move direction of axis X. If **i** is positive, the move direction is positive, If **i** is negative, the move direction is negative.
Uj	Optional, m specifies move direction of axis X. If **j** is positive, the move direction is positive, If **j** is negative, the move direction is negative.

Example: N00 M84 X1 Y -1 ; Axis X move continuously in positive direction
; Axis Y move continuously in negative direction

6.4.30. M85 - Stop Continuous Movement

Parameters: X Y Z U

Functions: Stop continuous movement. You can select any one of X, Y, Z and U.

Example: N00 M85 X Y ; Stop continuous movement of axis X and axis Y

6.4.31. M86 - Increase Variable Value

Parameters: Sm Vn

Functions: Increase variable value

Parameters Detail:

Sm	Variable to be increased or modified. **m** is a number which has the following definition. **S1-S24**: General-purpose Digital Outputs **S41-S42**: Duty-cycle of PWM Output 1 and PWM Output 2, from 0 to 1. **S51-S52**: Output voltage of DA Output 1 and DA Output 2, from 0.7V to 5V. **S61-S62**: They are used to toggle on/off the LED on the controller's case. **S71-S72**: Frequency of PWM Output 1 and PWM Output 2. **S99**: Set whether run program by ignoring the digital input and output operation

	S100: Run counter **S101-S119**: Internal variables which corresponded to Modbus register 457, 459, 461, …
Vn	Necessary. If Sm is digital output or S99, V1 toggles the variable, V0 keeps the variable. Otherwise V1 increase the variable value.

Example:
(Suppose digital output 5 is high level)
N10 M86 S5 V1 ; Change the digital output 5 to low level
N20 M02

(Note: If the above states are executed again, the digital output 5 resumes to high level. If the statement is M86 S5 V0, the digital output 5 keeps the same.)

6.4.32. M87 - Set variable Value

Parameters: Sm Vn

Functions: Set variable value

Parameters Detail:

Sm	Variable to be increased or modified. **m** is a number which has the following definition. **S1-S24**: General-purpose Digital Outputs **S41-S42**: Duty-cycle of PWM Output 1 and PWM Output 2, from 0 to 1. **S51-S52**: Output voltage of DA Output 1 and DA Output 2, from 0.7V to 5V. **S61-S62**: They are used to toggle on/off the LED on the controller's case. **S71-S72**: Frequency of PWM Output 1 and PWM Output 2. **S99**: Set whether run program by ignoring the digital input and output operation **S100**: Run counter **S101-S119**: Internal variables which corresponded to Modbus register 457, 459, 461, …
Vn	Necessary. If Sm is digital output or S99, V1 toggles the variable, V0 keeps the variable. Otherwise n is set to the corresponding variable..

Example:
Example:
(Suppose digital output 5 is high level)
N10 M86 S5 V1 ; Change the digital output 5 to low level
N20 M02

(Note: If the above states are executed again, the digital output 5 resumes to high level. If the statement is M86 S5

V0, the digital output 5 keeps the same. This feature can be used to debug the program if any problem.)

6.4.33. M89 - Pause until Pass the Point

Parameters: Xm Yn Zi Uj

Functions: Pause the program until a specific point is passed.

Parameters Detail:

Xm	Optional, Axis X coordinates m of the pass point, unit: mm.
Yn	Optional, Axis Y coordinates n of the pass point, unit: mm
Zi	Optional, Axis Z coordinates i of the pass point, unit: mm
Uj	Optional, Axis U coordinates j of the pass point, unit: mm

Note1: Only one axis takes effect. If more than one axis is specified, the first axis takes effect.

Note2: M89 can only be used in multi-task.

Example: N10 M89 X100 ; Wait until axis x pass 100

6.4.34. M90 - End Sub-loop

Functions: End sub-loop running

6.4.35. M91 - Start sub-loop

Parameters: Cm

Functions: Start sub-loop

Parameters Detail:

Cm	Necessary, specify the loop count

Example: N10 M91 C10 ; Repeats the statements between M91 and M90 10 times

6.4.36. M92 - Modify the work piece coordinates

Parameters: Xm Yn Zi Uj

Functions:

Parameters Detail:

Xm	Optional, Axis X coordinates m of the modified coordinates, unit: mm.
Yn	Optional, Axis Y coordinates n of the modified coordinates, unit: mm
Zi	Optional, Axis Z coordinates i of the modified coordinates unit: mm
Uj	Optional, Axis U coordinates j of the modified coordinates, unit: mm

Example: N00 M92 X0

6.4.37. M94 - Jump Depends on Conditional Variable

Parameters: Sm Vn Ni

Functions: Jump depending on conditional variable

Parameters Detail:

Sm	Conditional Variable. **m** is a number which has the following definition. **S1-S28**: General-purpose Digital Inputs **S99**: Set whether run program by ignoring the digital input and output operation **S100**: Run counter **S101-S119**: Internal variables which corresponded to Modbus register 457, 459, 461, …
Vn	Necessary. If Sm is digital input or S99, V1 means it is active, V0 means it is not active. Otherwise **n** is the variable value.
Ni	Line number

Example: N10 M94 S3 V1 N50 ; Goto N50 when digital input 3 is active

6.4.38. M95 - Unconditional Jump to line No.

Parameters: Nm

Functions: Unconditional jump to line No

Parameters Detail:

Nm	Necessary, specify the line No.

Example: N10 M95 N100 ; Goto line N100

6.4.39. M96 - Call sub-program depends on Conditional Variable

Parameters: Sm Vn Ni

Functions: Call sub-program depends on conditional variable

Note: The sub-program must be ended with M99.

Parameters Detail:

Sm	Conditional Variable. **m** is a number which has the following definition. **S1-S28**: General-purpose Digital Inputs **S99**: Set whether run program by ignoring the digital input and output operation **S100**: Run counter **S101-S119**: Internal variables which corresponded to Modbus register 457, 459, 461, …
Vn	Necessary. If Sm is digital input or S99, V1 means it is active, V0 means it is not active. Otherwise **n** is the variable value.
Ni	Line number

Example: N10 M96 S5 V1 N100 ;Call N100 if digital input 5 is active

6.4.40. M97 - Simultaneous Start of Multiple Tasks

Parameters: Nm

Functions: Simultaneous start of multiple tasks

Note1: In the sub-program called by M97, you can use G00, G01, G02, G03, M84 and M85. Continuous interpolation is not allowed.]

Note2: You can call sub-program and use sub-loop in the simultaneous multi-tasks.

Note3: M99 will terminate the main task as well as all the sub-tasks.

Note4 M02 in the sub-task will terminate the execution of G code program.

Note5: There are maximum 3 sub-tasks allowed.

Note6: Once there is exception appeared in the sub-task, the coordinates can not be displayed. Be careful.

Parameters Detail:

Nm	Necessary, specify the line No.

Example: Please refer to section 6.5.6.

6.4.41. M98 - Go to sub-program

Parameters: Nm

Functions: Go to sub-program

Note: The sub-program must be ended with M99.

Parameters Detail:

Nm	Necessary, specify the line No.

Example: N00 M98 N20 ; Call sub-program N20

6.4.42. M99 - Return to Main Program

Functions: Return to main program

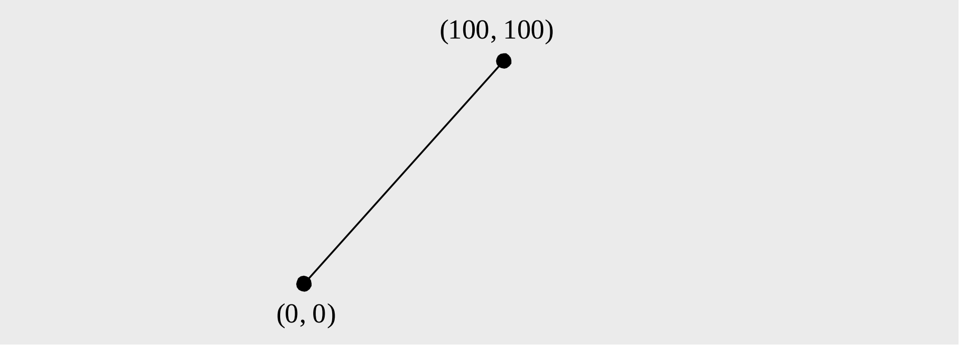
Example: N100 M99

6.5 Leadshine G Code Examples

6.5.1. Line

The following G-code welds a work piece along a line from (0, 0) to (100, 100).

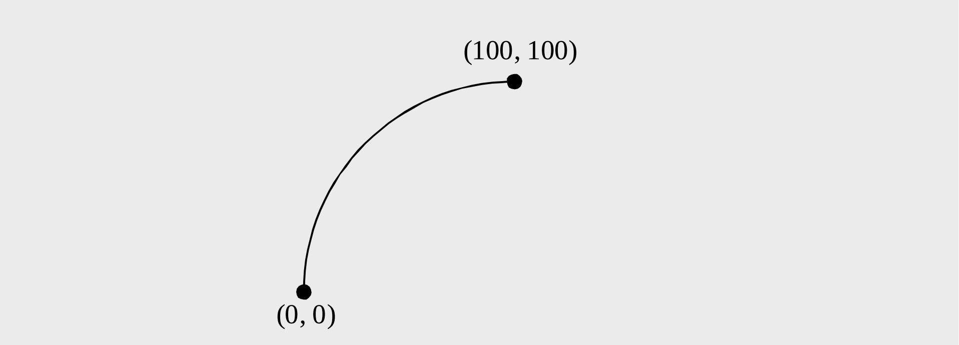
```
N00 G28 X Y               ; Home to (0, 0) of the work piece
N01 G91                   ; Use relative coordinates
N02 M07                   ; Turn on the laser（Output 1）
N03 G01 X100 Y100 F50     ; Linear interpolation at 50% feedrate
N04 M08                   ; Turn off the laser（Output 1）
N05 M02                   ; End
```



6.5.2. Circular interpolation

The following G-code welds a work piece along an arc from (0, 0) to (100, 100).

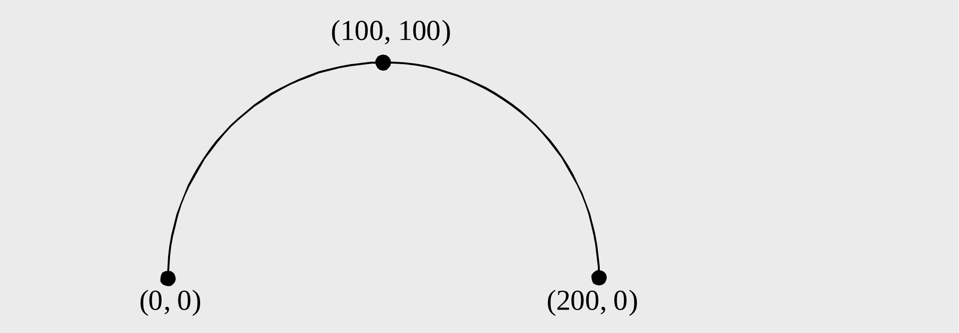
```
N00 G28 XY                ; Home to (0, 0) of the work piece
N02 M07                   ; Turn on the laser（Output 1）
N03 G02 X100 Y100 R100    ; Clockwise circular interpolation
N04 M08                   ; Turn off the laser（Output 1）
N10 M02                   ; End
```



6.5.3. Another Circular interpolation

The following G-code welds a work piece along an arc from (0, 0) to (200, 0).

```
N00 G28 XY              ; Home to (0, 0) of work piece
N02 M07                 ; Turn on the laser（Output 1）
N03 G05 X100Y100        ; Set midpoint (or point on same arc) of the arc
N04 G05 X200Y100        ; Set endpoint of the arc
N05 M08                 ; Turn off the laser（Output 1）
N10 M02                 ; End
```

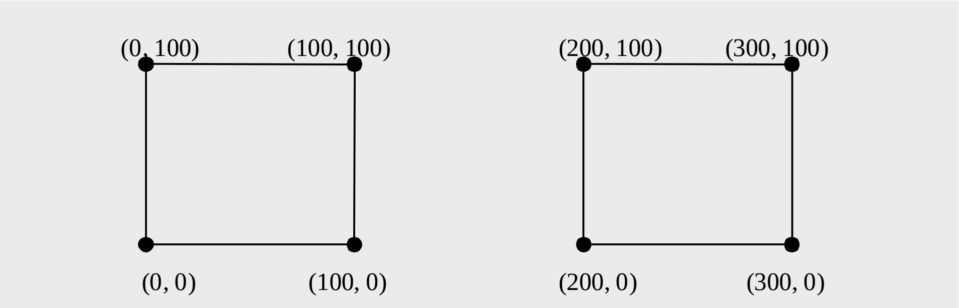


6.5.4. G92

6.5.4.1. Sub-program

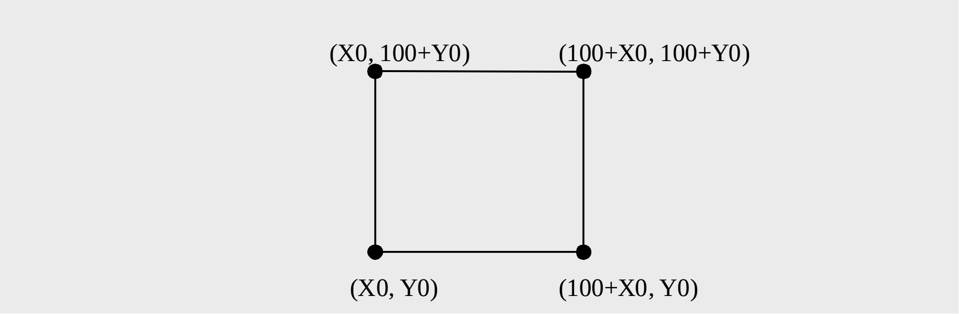
```
N01 G28 XY              ; Home to (0, 0) of work piece
N05 M98 N25             ; Call Sub-program at N25
N07 G00 X200            ; Move 200mm rightwards
N08 G92 X0Y0            ; Reset current coordinates as (0, 0)
N10 M98 N25             ; Call Sub-program at N25
N15 M02                 ; End

; Draw a square of 100*100
N25 M07                 ; Turn on the laser
N30 G01 X100
N31 G01 Y100
N32 G01 X0
N33 G01 Y0
N34 M08                 ; Turn off the laser
N40 M99                 ; Return
```

The above G-code draws two square of 100*100 by 200mm horizontal distance



6.5.4.2. Relative move using absolute coordinates

```
N10 G92 X0Y0            ; Reset the current coordinates as (0, 0)
N25 M07                 ; Turn on the laser（Output 1）
N30 G01 X100
N31 G01 Y100
N32 G01 X0
N33 G01 Y0
N34 M08                 ; Turn off the laser（Output 1）
N40 M02                 ; End
```



6.5.5. Jump and Repeat

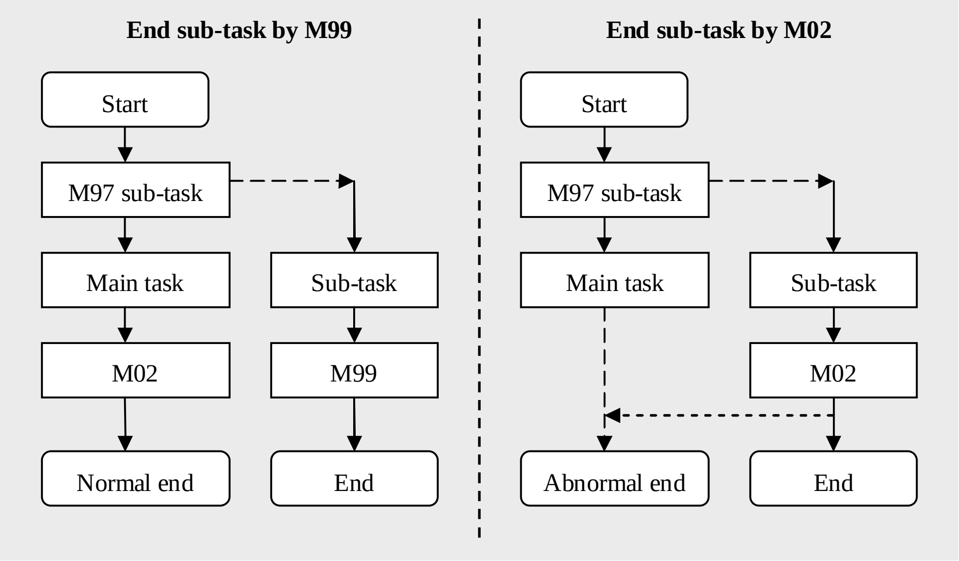
```
N01 G28 XY              ; Home to (0, 0) of the work piece
N10 M91 C100            ; Repeat 100 times
N20 M96 S10V1N100       ; Call Sub-program N100 if input10 effective,or continue
N30 G04 P2000           ; Delay 2000 millisecond
N40 M90                 ; End repeat

; Sub-program that draw a square of 100*100
N100 M07                ; Turn on the laser
N110 G01 X100
N120 G01 Y100
N130 G01 X0
N140 G01 X0
```

```
N150 M08                 ; Turn on the laser（Output 1）
N160 M99                 ; Return
```

6.5.6. Multi-task

```
N01 G28 X Y              ; Home to (0, 0) of the work piece
N10 M97 N200             ; Call multi-task and start the sub-task at N200
; Draw a square of 100*100
N100 M07                 ; Turn on the laser（Output 1）
N110 G01 X100
N120 G10 Y100
N130 G01 X0
N140 G01 Y0
N150 M08                 ; Turn on the laser（Output 1）
N160 M02                 ; End
; Check the digital input in sub-task
N200 M82 S10             ; Wait until input 10 effective
N210 M02                 ; Abnormal end
```

In the above G-code, the main task and the sub-task are parallel. If the sub-task is ended by M99, the main task continues until M02. When the sub-task ended by M02, both the main task and sub-task would stop. The life time of the sub-task is show as the following figure.



6.5.7. M89

Note: M89 can only be used in multi-task.
The following example turns on the valve when X axis move to a specific position.

```
N01 G28 X Y              ; Home to (0, 0) of the work piece
N10 M97 N100             ; Call multi-task and start the sub-task at N100
N20 M07                  ; Turn on the laser
N30 G01 X100 Y200
N40 M08                  ; Turn on the laser
N50 M02                  ; End
N100 M89 X100            ; Wait until axis X reach to 100
N110 M09                 ; Turn on the valve
N120 G04 P1000           ; Delay 1000 millisecond
N130 M10                 ; Turn off the valve
N140 M99                 ; End sub-program and return
```

6.5.8. Example of battery welding

The following G-code is a practical example of battery welding for a manufacturer of cell phone battery.

Pin assignment:
Input:
Digital input 8: Check whether the battery is in-position

Output:
Digital output 1: Control to clamp the battery in vertical direction
Digital output 2: Control to clamp the battery in horizontal direction
Digital output 4: Switch for pushing the battery
Digital output 5: Another switch for pushing the battery
Digital output 6: Switch for nitrogen
Digital output 7: Switch for laser

Axis functions (Move the head to start point before welding then reset the coordinates by G92)
Axis X: Move the battery
Axis Y: Rotate the battery.

Process:
Firstly push the battery to the camp then check whether the battery is in position. End the program if no battery is detected or clamp the battery and begin welding.

```
N010 G92 X0 Y0              ; Set current coordinates (0, 0)
N020 G80 S4                 : Push the battery to camp
N030 G04 P800               ; Delay 800 millisecond
N040 M80 S5                 ; Push the battery in horizontal direction
N050 G04 P500
```

```
N060 M94 S8 V0 N300          ; Jump to N300 if no battery detected
N070 M80 S2
N080 G04 P500
N090 M80 S1                  ; Camp the battery
N100 M81 S5
N110 M81 S4                  ; Finish pushing the battery
N120 G04 P500
N130 G00 X34                 ; Fast position to the start point
N140 M80 S6                  ; Turn on the nitrogen
N150 G04 P300
N160 80 S7                   ; Turn on the laser
N170 G01 X-0.5 F100          ; First welding
N180 M81 S7                  ; Turn off laser
N190 G01 X7 Y90 F300         ; Rotates the clamp 90 degree
N200 M80 S7
N210 G04 P80
N220 G01 X-0.5 F100          ; Second welding
N230 M81 S7                  ; Turn off laser
N240 M81 S6                  ; Turn off the nitrogen
N250 G01 Y135                ; Rotates the clamp 135 degree
N260 M81 S2                  ; Release the battery
N270 M81 S1
N280 G04 P800
N290 G00 X0 Y0               ; Return to (0, 0) when finish welding
N300 M81 S5
N310 M81 S4
N320 M02                     ; End
```

Contact Us

China Headquarters

Address: 3/F, Block 2, Nanyou Tianan Industrial Park, Nanshan District Shenzhen, China
Web: http://www.leadshine.com

Sales Hot Line:
Tel: 86-755-2641-7674 (for Asia, Australia, Africa areas)
86-755-2640-9254 (for Europe areas)
86-755-2641-7617 (for America areas)
Fax: 86-755-2640-2718
Email: sales@leadshine.com.

Technical Support:
Tel: 86-755-2641-8447, 86-755-2641-8774, 86-755-2641-0546
Fax: 86-755-2640-2718
Email: tech@leadshine.com(for All)

Leadshine U.S.A
Address: 25 Mauchly, Suite 318 Irvine, California 92618
Tel: 1-949-608-7270
Fax: 1-949-608-7298
Web: http://www.leadshineUSA.com
Email: sales@leadshineUSA.com and support@leadshineUSA.com.