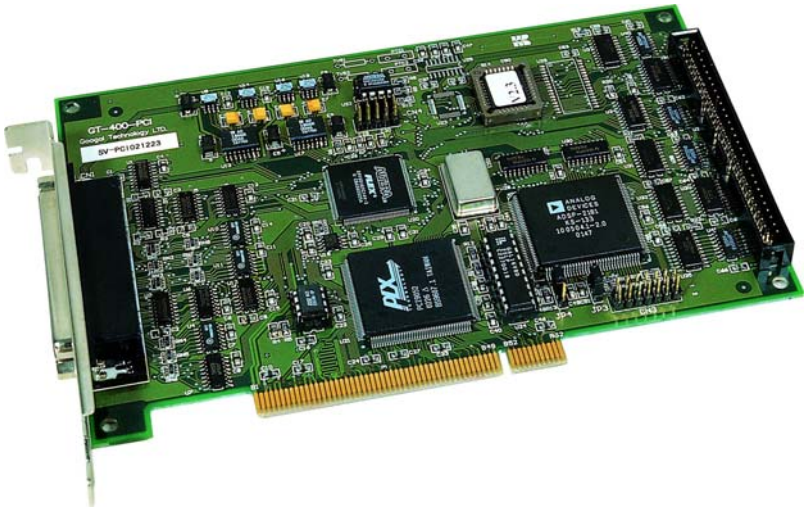
GOOGOL TECHNOLOGY (HK) LTD

Programming Manual

for

GE Series Motion Controller



Version 1.0 : 16 Dec, 2004

Part#: GE400-M-E-10-1216-002

Copyright Statement

Googol Technology Ltd.
All rights reserved.

Googol Technology Ltd. (Googoltech for short hereafter) reserves the right of modifying the products and product specifications described in this manual without notification in advance.

Googoltech is not submitted to any direct, indirect, special, incidental or consequent loss or liability caused by using this manual or product incorrectly.

Googoltech owns the patent, copyright or any other intellectual property right of this product and its software. No one shall directly or indirectly duplicate, produce, process or use this product and its relevant parts.

 **Notice**	***There is danger in a controller in motion! User has the duty to design an effective error treatment and safety protection system for the machine. Googoltech has no such obligation or liability to be responsible for any incidental or consequent loss caused accordingly.***

Foreword

Thank you for using Googoltech motion controller

To repay customers, we will help you establish your own control system with our first-class motion controller, perfect after-sale service, and high-efficiency technical support.

Technical Support and After-sale Service

You may get our technical support and after-sale service through the following approaches:

- E-mail: *support@googoltech.com*
- Tel.: (0755) 2697-0823, 2697-0835, 2697-0839
- Post: 518057

Use of this Programming Manual

By reading this manual, you will know the control functions of GE series motion controller, learn the usage of motion functions, and become familiar with the programming of specific control function. Finally, you can program your own user application for control according to your specific control system.

User of this Programming Manual

This manual is applicable to those engineering developers having the base knowledge of programming in C or using Dynamic Link Library (DLL) in Windows environment, and certain work experience in motion control and understanding of the basic architecture of servo or step control.

Main Contents of this Programming Manual

This manual is divided into two parts. The first part is programming specification, mainlyintroducing various control functions and corresponding programming approaches of GE series motion controller. The second part is interface library function specification, mainly introducing the function prototype, description and evoking of interface functions of GE series motion controller.

Usage Description of this Programming Manual

Except stated specially, among all the samples listed in this manual, all DOS samples are written in Borland C3.1 and all Windows samples in VC++.

Relevant Documents

For the installation and debugging of GE series motion controller, please refer to User’s Guide of GE series Motion Controller provided together with product.

目录

Chapter 1 Usage of Function Library of Motion Controller

The motion controller provides motion function library in DOS and DLL in Windows. Just by calling the command in library, the user can realize various functions of motion controller. The usage of library function in DOS and Windows will be described respectively as follow.

1.1 Function Library in DOS

The motion function library in DOS for continuous trajectory motion controller (GE-X00-SX) exists in the directory of DOS\UserLib in the CD provided with product, including seven files:

Ges.h	Head file
gest.lib	Function library of micro mode(Tiny)
gess.lib	Function library of small mode(Small)
gesm.lib	Function library of middle-sized mode(Medium)
gesc.lib	Function library of compact mode(Compact)
gesl.lib	Function library of large mode(Large)
gesh.lib	Function library of huge mode(Huge)

The motion function library in DOS for point-to-point motion controller (GE-X00-PX) exists in the directory of DOS\UserLib in the CD provided with product, including seven files:

Gep.h	Head file
gept.lib	Function library of micro mode(Tiny)
geps.lib	Function library of small mode(Small)
gepm.lib	Function library of middle-sized mode(Medium)
gepc.lib	Function library of compact mode(Compact)
gepl.lib	Function library of large mode(Large)
geph.lib	Function library of huge mode(Huge)

This library is compiled in Borland C3.1. Developers can link it in Borland C3.1 or higher versions of developing environment.

Methods of using library:

1. Make sure the compiling mode:

- Startup Borland C3.1.
- Select menu item “Project->Open Project”, open project file existed or create the new project file.
- Select menu item “Options->Compiler->Code-generation…”.
- Select corresponding compiling mode in “Model” section.

2. Add function library:

- Select the function library and head file of corresponding compiling mode, and copy them into the current project folder.
- Select menu item “Window->Project”, switch to project window.
- Select menu item “Project ->Add Item…”.
- Input “*.lib” in the “Name” section and enter.
- Select the function library required In “Files” section, and then click the button “Add”.

3. Add the source routine file:

- Select Menu “File->New”, and create a new “.cpp” or “.c” file.
- Add the statement of head file of function library in the new “.cpp” or “.c” file. For example: #include “ges.h”; ///state the head file of function library of continuous trajectory motion controller.
- Save this file, and then add the source routine file into current project.

-

1.2 DLL in Windows

The motion controller can be used after installing the driver. The driver exists in the directory of Windows\driver in the CD provided with product.

The DLL in Windows exists in the directory of Windows\VC in the CD provided with product. The DLL name of continuous trajectory motion controller is “ges.dll”, and the DLL name of point-to-point motion controller.

You can develop your own application program with any developing tool by calling the DLL, Regarding the advanced programming languages VC, VB and Delphi frequently used by programmers today, the usage of DLL in such languages will be described one by one as follows.

1.2.1 Usage in VC

1. Startup the VC++, and create a new project.
2. Copy the DLL (such as ges.dll) in the Windows\VC folder in the CD provided with the product into the project folder.
3. Copy the head file and lib file in the Windows\VC folder in the CD provided with the product into the project folder.
4. Select menu item “project ->settings…”.
5. Switch to Link tab-page, and input lib file name in “Object/library modules” section.
6. Add the head file statement of function library into the user program: ＃include “ges.h”.
7. Now, users can develop their application program by calling the function of DLL in Visual C++.

1.2.2 Usage in VB

1. Startup the VB, and create a project.
2. Copy the DLL (such as ges.dll) in the Windows\VB folder in the CD provided with the product into system folder.
3. Copy the function statement file (such as ges.bas) in the Windows\VB folder in the CD provided with the product into the project folder.
4. Select menu item “Project -> Add Module”.
5. Switch to “Existent” tab-page, and select function statement file (such as ges.bas), and add it into the project.
6. Now, users can develop their application program by calling the function of DLL in Visual Basic.

1.2.3 Usage in Delphi

1. Startup the Delphi and create a new project.
2. Copy the DLL (such as ges.dll) in the Windows\Delphi folder in the CD provided with the product into the project folder.
3. Copy function statement file (such as ges.pas) in the Windows\Delphi folder in the CD provided with the product into the project folder.
4. Select menu item “Add to Project…”.
5. Add function statement into current project file.
6. Switch to user unit file in the code editor window.
7. Select menu item “File->Use Unit…”, add the cite for function statement file.
8. Now, users can develop their application program by calling the function of DLL in Delphi.

Chapter 2 Command Return Values and their Meaning

2.1 Command Return Values

Command and Library Function: The motion controller works according to the ***motion controller commands*** sent by the host. It has C function library (in DOS) and DLL (in Windows). The user can call corresponding ***function in the library*** to operate the motion controller when he is programming in the host.

After receiving a command from the host, the motion controller will give a feedback after checking and verifying it. The definitions of return values are listed in Table 2-1.

Table 2-1 Definition of Return Values of GE-X00-PX

Value	Meaning	Processing Method
-1	Error in communication.	Check whether the controller works normally.
0	Command execution succeeded.	Operation can go on.
1	Error in command.	Call GT_GetCmdSts(). Find causes and correct them.
7	Command parameter error.	Check whether the command parameters are reasonable or exceed the limits. Send the command again after correction.
1024	Failure in opening the card.	Check whether the card has been opened in other program.

Table 2-2 Definition of Return Values of GE-X00-SX

Value	Meaning	Processing Method
-1	Error in communication.	Check whether the controller works normally.
0	Command execution succeeded.	Operation can go on.
1	Error in command.	Call GT_GetCmdSts(). Find causes and correct them.
2	Circular radius is zero or chord length is zero (except the whole circle).	This value can only be returned after GT_ArcXY, GT_ArcYZ or GT_ArcZX command is called. Check the command parameters and send the command again after correction.
3	Linear interpolation length is zero, negative or exceeds the motion bound of the controller.	This value can only be returned after GT_LnXY, GT_LnXYZ or GT_LnXYZA command is called. Check the command parameters and send the command again after correction.
4	In coordinated motion mode, the vector velocity (acceleration) is zero, negative or exceeds the motion bound of the controller.	This value can only be returned after GT_SetSynVel or GT_SetSynAcc command is called. Check the command parameters and send the command again after correction.
5	Chord length is more than the diameter.	This value can only be returned after GT_ArcXYP, GT_ArcYZPor or GT_ArcZXP command is called. Check the command parameters and send the command again after correction.
7	Command parameter error.	Check whether the command parameters are reasonable or exceed the limits. Send the command again after correction.

注意 Notice	*It is suggested to check each command return value in the main user program to confirm if the execution of the command is right, and establish necessary error treatment mechanism to assure the safety and reliability of the program.*

If the return value is -1 and it returns several times when being called repeatedly, it means that the communication of the motion controller has fault and the controller doesn't receive

the command from the host correctly, or the function library doesn't work normally, unable to process the command from the host correctly. At this time, user should check the matter according to the following step:

1. Please check whether the driver is installed correctly when users use the motion controller in Window system.

2. Check whether the connection of motion controller, connector and connect line is firmed.

3. Check whether the base address jumper is correct; please make sure the base address jumper of motion controller is matched with the specified base address of application routine (for the motion controller with ISA/PC104 bus).

4. Check whether the slot of PC can work normally, and try to change the slot or change the PC.

5. Check whether the JP4 jumper is correct, the default JP4 jumper is short circuit between pin 1 and pin 2.

2.2 Command Status Register

When the return value is 1, it means that the instruction is executed mistakenly. At this time, user
can call the command GT_GetCmdSts() to get the error causes. Command parameters can send
back the content of command status register, the detailed definition of command status register is
listed below, and when some bit is 1, it indicates corresponding error cause.

Table 2-3 Definition of Command Status Register of GE-X00-PX

Bit	Definition
Bit0	The command parameter is overflow.
Bit1	The command parameter is illegal.
Bit2	The host call GT_MltiUpdt (value), but the value=0.
Bit3	The illegal use of GT_DrvRst. When the current axis is in activating status, the host calls this command.
Bit4	Reserved
Bit5	Reserved
Bit6	When the specified axis is in motion, the host calls a command to change the motion mode of specified axis.
Bit7	When the sign of the Home or Index captured of the current axis status register is 1, the command GT_CaptIndex is called before the position is captured, or the host calls the command GT_CaptIndex again without

Bit	Definition
	clearing the status bit after the position is captured.
Bit8	When the sign of the Home or Index captured of the current axis status register is 1, the command GT_CaptHome is called before the position is captured, or the host calls the command GT_CaptHome again without clearing the status bit after the position is captured.
Bit9	When the driver alarm signal of the current axis status register is 1, the host calls the command GT_AxisOn.
Bit10	Reserved
Bit11	When the current axis is in motion, the host calls the command GT_ZeroPos() , or GT_SetAtlPos; or call the command GT_Update() (GT_MltiUpdt()) to modify some parameter which can not be changed during the current axis is in motion (For example, in the S-curve motion mode, when the motor is in motion, the host calls the command GT_SetVel and GT_Update or GT_MltiUpdt.)
Bit12	Reserved
Bit13	Reserved
Bit14	Reserved
Bit15	Reserved

Table 2-4 Definition of Command Status Register of GE series continuous trajectory motion controller

Bit	Definition
Bit0	Reserved.
Bit1	The command parameter is illegal.
Bit2	The host call GT_MltiUpdt (value), but the value=0.
Bit3	Motor status of control axis error: The illegal use of GT_DrvRst, GT_AlarmOn, GT_AlarmOff, GT_EncOff, GT_EncOn when the current axis is in activating status, the host calls this command. The host calls GT_AxisOn when serve alarm sign is generated.
Bit4	Reserved
Bit5	Set capture status command error: The capture mode has been set, but capture signal does not come, or although capture signal has come, the host calls the command of capture mode again without clearing this status bit.
Bit6	Status command of control axis error: Call those commands correlative to servo control, such as GT_CloseLp, GT_OpenLp, GT_SetKp when control axis stays in pulse output mode. Call GT_CtrlMode when control axis stays in activating status. Call GT_SepDir/GT_StepPulse when control axis stays in servo control

Bit	Definition
	mode; When the host calls GT_MapCnt to adjust the offset value, some axis is in motion, or the buffer is not closed, or the parameters exceed the limit.
Bit7	Reserved.
Bit8	Continuous trajectory motion command error: Directly call continuous trajectory commands (for example GT_LnXY) without sending the positioning command after the host call GT_StrtList to open buffer. Some interpolation command has not been over when the host calls direct-command. Call the positioning command again after the host closes the buffer, or call the positioning command many times; the velocity and acceleration parameters are not correct in positioning command. Parameters are negative or zero when GT_SetSynVel, GT_SetStrtVel are called. GT_SetSynAcc is called in the buffer mode. Interpolation motion has not been over when calling GT_SetStrtVel. The corresponding parameters of starting velocity, maximal velocity, velocity, acceleration, abrupt-stop acceleration and velocity override are negative
Bit9	Buffer status management command error: Interpolation command exists and the motion is not over when GT_StrtMtn is called. Continuous trajectory motion in buffer is not over or the buffer has been opened when GT_StrtList is called. GT_AddList is called when buffer is still in open status. Interpolation motion is not over or trajectory command in buffer is complete when GT_restoreMtn is called.
Bit10	Reserved
Bit11	When bit 10 of motion status register is 1, this bit is 1 when GT_ZeroPos or GT_SetAtlPos is called.
Bit12	Reserved
Bit13	Reserved
Bit14	Command error, this command is not among the continuous trajectory command sets.
Bit15	Motion command is not accepted by the controller because the buffer has been full; and the host need call the command repeatedly until the controller accepts this command.

※The following sample gives the method that error treatment is done through the command return value in standard C. When the return value is exceptional, the host can send buzzer sound and display the error information.

```
void error(short rtn)
{
        switch(rtn)
        {
                case    -1:
                        printf("\a\nCommunication Error !");          break;
                case     7:
                        printf("\a\nParameter Error !");              break;
                case     1:
                        printf("\a\nCommand Error !");                break;
                default   :
                        break;
        }
}
void main()
{
        short    rtn;
        rtn = GT_Reset();                   error(rtn);
        rtn = GT_ClrSts(1);                 error(rtn);

  }
```

2.3 Record the Command Executing Procedure

When user debugs the application, he often needs check the commands and their parameters. To make the matter simple, the command GT_HookCommand can be used.

User can call GT_HookCommand to hook his command treatment program to function library. When user calls the command of motion controller, the function library can automatically call the command treatment program hooked after the command has been executed, and send the current command name, the command parameters, and the command return value to the command treatment program. The command treatment program can record the information into file in order to analyze them.

User can also check the return value in the command treatment program, and do the error treatment concentrated to make the error treatment simple.

```
void __stdcall CommandHandle(char *command,short error)
```

2.3.1 The statement and hooking of the command treatment function in Visual C++

```
{
}
```

```
void main()
{
        GT_HookCommand(CommandHandle);
 }
```

The parameter command records the command name and parameters, and the parameter error records the return value of command.

2.3.2 The statement and hooking of the command treatment function in Visual Basic

```
Sub CommandHandle(ByVal command As String, ByVal error As Integer)
        Dim Msg As String
        Msg = StrConv(command, vbUnicode)                    ’ convert into Unicode codes
      End Sub
      Private Sub Command1_Click()
            GT_HookCommand AddressOf CommandHandle
      End Sub
```

The parameter command records the command name and parameters, and the parameter error records the return value of command.

In Visual Basic the command treatment program must be independently defined in the standard module, and StrConv() is called to convert string into Unicode codes.

2.3.3 The statement and hooking of the command treatment function in Delphi

```
procedure CommandHandle(command:PChar;error:SmallInt);stdcall
    begin
    end
    procedure TForm1.Button1Click(Sender: TObject);
    begin
    GT_HookCommand(CommandHandle);
    end;
```

command records the command name and parameters, error records the return value of command.

Chapter 3 Initialization of Control System

GE series motion controller includes continuous trajectory motion controller (GE-X00-SX) and point-to-point motion controller (GE-X00-PX). You must select and set it correctly before developing your application.

3.1 Initialization of Motion Controller

3.1.1 Command Summary

Table 3-1 Initialization Command Summary of Motion Controller

Function	Description
GT_Open()	Open the motion controller Need set base address for ISA bus Need not set base address for PCI bus
GT_Reset()	Reset the motion controller.

3.1.2 Notes of Main Point

Open motion controller

At first user must call GT_Open to open the motion controller and enable the communication between the host and motion controller.

The base address must be specified when GT_Open is called for those cards with ISA bus. The base address parameter should correspond to the “Base Address jumper” on the motion control board (See ***Set jumper on Motion Control Card*** in User’s Guide of GE Series Motion Controller.).

The base address need not be specified for PCI bus.

When user exits the application routine, he must call GT_Close to close the motion controller.

Reset the motion controller

GT_Reset is called to make all registers of motion controller come back the default status. The function is usually called after GT_Open is called.

3.2 Set Parameters for Dedicated Input Signal

3.2.1 Command Summary

Table 3-2 Command Summary of Specifying Parameters for Dedicated Input Signal

Function	Description
GT_LmtSns	Specify the effective electrical level for limit switch.
GT_HomeSns	Specify the triggering edge of Home signal of each axis.
GT_EncSns	Specify the direction of encoder (only for SV card).

3.2.2 Notes of Main Point

Set effective electrical level for limit switch.

Motion controller provides two (positive and negative) limit switches interfaces for each axis. The limit switch of each axis should connect the corresponding limit signal input interface and OGND on the terminal board. Please see “the connect mode of dedicated input and output Signal” in User’s Guide of GE Series Motion Controller. Once the limit switch is triggered, the controller will prohibit automatically the motion of control axis towards the limit. After leave the limit switch, the host must call GT_ClrSts to clear the limit status of this axis.

The corresponding bit of the parameter of GT_LmtSns specifies the triggering level of positive and negative limit switch. When some status bit of command parameter is set as 0, it means that the triggering level of corresponding limit switch is at high level. Whereas, if it is set to 1, it means that the input signal of limit switch is the triggering limit bit at low level. The corresponding relation between the status bit of parameter and limit switch is as follows:

Limit Switch	Axis #8		Axis #7		Axis #6		Axis #5		Axis #4		Axis #3		Axis #2		Axis #1	
	−	+	−	+	−	+	−	+	−	+	−	+	−	+	−	+
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Set triggering edge for limit switch.

Motion controller provides a Home signal input interface for each axis. The home switch of each axis should connect the corresponding home signal input interface and OGND on the terminal board. Please see “the connect mode of dedicated input and output Signal” in User’s Guide of GE Series Motion Controller.

The command GT_HomeSns can change the edge triggering mode of home signal. The default triggering edge of home signal is down-edge.

When some status bit of command parameter is set as 0, it means that the triggering edge of corresponding home switch is at down-edge. Whereas, if it is set to 1, it means that the triggering edge of home switch is at down-edge. The corresponding relation between the status bit of parameter and home switch is as follows:

Status Bit	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Home switch	Home7	Home6	Home5	Home4	Home3	Home2	Home1	Home0

Set the direction of coder (only for SV card).

When the positive direction of the rotating of the motor is consistent with the positive direction of the counting of coder, the correct feedback is established. But in actual application, it may be due to the unmatched set of motor and coder or wrong connection that the positive rotating of motor is opposite to that of coder, forming positive feedback and causing controller abnormal in work. Now, user can modify the counting direction of coder with a command GT_EncSns.

When some status bit of command parameter is set as 1, it means that the counting direction will be set as opposite direction. The definition of status bit is as follows:

Status Bit	7	6	5	4	3	2	1	0
Axis	Axis #7	Axis #6	Axis #5	Axis #4	Axis #3	Axis #2	Axis #1	Axis #0

Drive Alarm signal

Motion controller provides a drive alarm input signal for each axis. If the signal is not requisite, user can make alarm input signal pin and OGND connect to short circuit on terminal board, or call GT_AlarmOff to make alarm ineffective. Please see “the connect mode of dedicated input and output Signal” in User’s Guide of GE Series Motion Controller.

When control axis is in alarm status, motion controller will refuse to accept the motion commands for this axis. After the drive alarm of driver is settled, GT_ClrSts needs be called to clear the alarm status of the axis.

3.3 Initialization of Motion Control Axis

3.3.1 Command Summary

Table 3-3 Initialization Function List of Motion Controller

Function	Description
GT_AxisOn	Activate drive.
GT_ClrSts	Clear the current axis status.
GT_StepDir	Set the output mode of the current axis in the pulse output mode as “Pulse/Direction”.
GT_StepPulse	Set the output mode of the current axis in the pulse output mode as “Positive Pulse/Negative Pulse”.
GT_CtrlMode()	Set output analog amount/pulse amount. (GE-X00-XV)
GT_CloseLp()	Set as close loop control. (GE-X00-XV)
GT_OpenLp()	Set as open loop control. (GE-X00-XV)
GT_SetKp()	Set the servo filter percentage gain of the current axis. (GE-X00-XV)

GT_SetKi()	Set the servo filter integral gain of the current axis. (GE-X00-XV)
GT_SetKd()	Set the servo filter differential gain of the current axis. (GE-X00-XV)
GT_SetKvff()	Set the servo filter velocity feedforward gain of the current axis. (GE-X00-XV)
GT_SetKaff()	Set the servo filter acceleration feedforward gain of the current axis. (GE-X00-XV)
GT_SetILmt()	Set the servo filter error integral saturation of the current axis. (GE-X00-XV)
GT_SetMtrLmt()	Set the servo filter output saturation of the current axis. (GE-X00-XV)
GT_SetMtrBias()	Set the servo filter output zero point excursion of the current axis. (GE-X00-XV)

3.3.2 Example

Example 3-1 function which processes the return value of command

```
void error(short rtn)
{
    switch(rtn)
    {
        case  -1:
            printf("\a\nCommunication Error !"); break;
        case  1:
            printf("\a\nCommand Error !");    break;
        case  2:
            printf("\a\nRadius or chord is 0 !");   break;
        case  3:
            printf("\a\nLength is 0 or overflow !");    break;
        case  4:
            printf("\a\nVelocity or acceleration is less then 0 !");
                                                    break;
        case  5:
            printf("\a\nChord is greater than diameter !");
                                                    break;
        case  7:
            printf("\a\nParameter error !");    break;
        default:
            break;
    }
}
```

Example 3-2 the initialization of Control axis of GE-X00-XG

```
void AxisInitial(short axis_num,unsigned short limit,unsigned short encoder)
                {//Initialization function
                 short rtn;
                 rtn=GT_LmtSns(limit); //Set triggering level of limit switch
                 error(rtn);
                 rtn=GT_EncSns(encoder);    //Set the counting direction
                 error(rtn);
                 for(short i=1;i<=axis_num;++i)
                 {
                 rtn=GT_StepPulse(i);    // Set as " Positive and Negative Pulse "
                 error(rtn);
                 rtn=GT_AxisOn(i);       // Activate drive
                 error(rtn);
                 rtn=GT_ClrSts(i);  //Reset status register
                 error(rtn);
                 delay(200);    //Waiting for the servo ok
                 }
                 }
```

Example 3-3 the initialization of Control axis of GE-X00-XV (output pulse)

```
void AxisInitial(short axis_num,unsigned short limit,unsigned short encoder)
    {                                                   // Initialization function
        short rtn;
          rtn=GT_LmtSns(limit);                         //Set triggering level of limit
                                                          switch
          error(rtn);
          rtn=GT_EncSns(encoder);                       //Set the counting direction
          error(rtn);
        for(short i=1;i<=axis_num;++i)
        {
              rtn=GT_CtrlMode(i,1);                     //Set pulse output mode
              error(rtn);
              rtn=GT_StepPulse(i);                      // Set as "Positive and Negative
                                                          Pulse"
              error(rtn);
               rtn=GT_AxisOn(i);                        // Activate drive
               error(rtn);
              rtn=GT_ClrSts(i);                         //Reset status register
              error(rtn);
              delay(200);

        }
    }
```

Example 3-4 the initialization of Control axis of GE-X00-XV (output analog voltage)

```
    void AxisInitial(short axis_num,unsigned short limit,unsigned short
encoder，double kp)                    // Initialization function
    {
    short rtn;
    rtn=GT_LmtSns(limit);              //Set triggering level of limit
switch
    error(rtn);
    rtn=GT_EncSns(encoder);            //Set the counting direction
    error(rtn);
    for(short i=1;i<=axis_num;++i)
    {
    rtn=GT_SetKp(i,kp);                //Set Kp
    error(rtn);
    rtn=GT_Update(i);                  //Update the parameters
    error(rtn);
    rtn=GT_AxisOn(i);                  // Activate drive
    error(rtn);
    rtn=GT_ClrSts(i);                  //Reset status register
    error(rtn);
    delay(200);
    }
    }
```

3.3.3 Notes of Main Point

Set current axis output mode "analog voltage output" /"pulse output"

GE-X00-XG series motion controller can only perform pulse output, and it can control step motor and servo motor with position mode. GE-X00-XV series motion controller can carry out simultaneity pulse output and analog voltage output. The command GT_CtrlMode can be call to switch output mode of control axis.

Set pulse output mode "Pulse+Direction" and "Positive and Negative Pulse"

GE-X00-XG series motion controller can only perform pulse output. The default pulse output mode is "pulse+direction". The command GT_StepPulse can be called to set pulse output mode as "positive and negative pulse"; the command GT_StepDir can be called to set pulse output mode as "pulse+direction".

After GE-X00-XV series motion controller must call GT_CtrlMode(1) to switch to pulse output mode, pulse output mode of control axis can be changed really.

Set analog voltage output mode

The default output of GE-X00-XV series motion controller is analog voltage. The command GT_CtrlMode(0) can be called to switch back to analog voltage output mode from pulse output mode.

Each control axis may have zero point excursions, and user need call GT_SetMtrBias to set the compensation for zero point excursions, then precise positioning can be guaranteed.

3.4 Set actual position of Motion Control Axis

3.4.1 Command Summary

Table 3-4 Function List of setting actual position

Function	Description
GT_ZeroPos	Actual position and target position are set as zero
GT_SetAtlPos	Set the actual position of control axis

3.4.2 Notes of Main Point

Actual position and target position are clear to zero

The command GT_ZeroPos can be called to set the actual position and target position as zero. This command will be effective when the motion is stopped.

Set the actual position of the specified axis

The command GT_SetAtlPos can be called to set actual position and target position as specified value. This command will be effective when the motion is stopped.

3.5 Initialization example of motion controller

Example 3-5 the initialization of Control axis of GE-300-SG

```
#include <stdio.h>
#include “ges.h”

void CommandHandle(char *command,short error)
{
switch(error)
{
case   -1:
printf("\a\nCommunication Error !");   break;
case   1:
printf("\a\nCommand Error !");        break;
case   2:
printf("\a\nRadius or chord is 0 !");  break;
case   3:
printf("\a\nLength is 0 or overflow !"); break;
case   4:
printf("\a\nVelocity or acceleration is less then 0 !");
                                       break;
case   5:
printf("\a\nChord is greater than diameter !");
                                       break;
case   7:
printf("\a\nParameter error !");        break;
default:
break;
}
}
```

```
void main()
{
GT_HookCommand(CommandHandle);    //Hook the command treatment program
GT_Open(0x300);                   //Open motion controller for ISA bus
                                  //Base address is 0x300
GT_Reset();                       //Reset motion controller
GT_LmtSns(0);                     //Set triggering level of limit switch
GT_EncSns(0);                     //Set the counting direction of encoder
for(short i=1;i<=3;++i)
{
                                  GT_StepPulse(i);// Set as "Positive and Negative Pulse"
GT_AxisOn(i);                     // Activate drive
GT_ClrSts(i);                     //Reset status register
}
}
```

Example 3-5 the initialization of Control axis of GE-300-SG

```
#include <stdio.h>
#include "ges.h"
void error(short rtn)
{
switch(rtn)
{
case  -1:
printf("\a\nCommunication Error !");  break;
case  1:
printf("\a\nCommand Error !");      break;
case  2:
printf("\a\nRadius or chord is 0 !"); break;
case  3:
printf("\a\nLength is 0 or overflow !"); break;
case  4:
printf("\a\nVelocity or acceleration is less then 0 !");
                                  break;
case  5:
printf("\a\nChord is greater than diameter !");
                                  break;
case  7:
printf("\a\nParameter error !");      break;
default:
break;
}
}
void main()
{
short rtn;
rtn=GT_Open(0x300);
error(rtn);
rtn=GT_Reset();
error(rtn);
rtn=GT_LmtSns(0);
```

```
error(rtn);
rtn=GT_EncSns(0);
error(rtn);
for(short i=1;i<=3;++i)
{
rtn=GT_StepPulse(i);
error(rtn);
rtn=GT_AxisOn(i);
error(rtn);
rtn=GT_ClrSts(i);
error(rtn);
}
}
```

Chapter 4 Motion Controller Status Check

4.1 Command Summary

Table4-1 Function list of axis status checking

Function	Description
GT_GetSts	Get the status of specified axis
GT_SetCrdSts	Get continuous trajectory motion status (only for GE-X00-Sx)
GT_ClrSts	Clear the status of specified axis
GT_GetMode	Get the work mode of specified axis(only for GE-X00-PX)
GT_RstSts	Clear the specified status bit of specified axis

4.2 Notes of Main Point

For GE-X00-PX motion controller, GT_GetSts is called to get the control axis status in status register, and the bit10 in the status register specifies the motion status of control axis.

For GE-X00-SX motion controller, GT_GetCrdSts is called to read the continuous trajectory motion status register, the bit0 specifies the continuous trajectory motion status.

For GE-X00-SX motion controller, it is not suggested that bit10 is used to judge the trajectory motion (The preparation and pretreatment of trajectory motion need a period of time, and when start motion this bit has not been set as 1)

The motion status flag bit of control axis or continuous trajectory specifies the motion profile status not actual motion status. It is that whether the controller has completed an acceleration/deceleration profile procession and arrived at the target position. Whether the control axis has reached the target position is relative to the servo-delay, stability and other condition of actual system.

4.3 Control Axis Status Register

The motion controller provides a status register for each axis. The user can use command GT_GetSts() to get the status of current axis. The status register of GE-X00-SX is 16-bit, the definition of each bit is listed in table4-2; the status register of GE-X00-PX is 32-bit, the definition of each bit is listed in table4-3. Bit8-15 specifies the motion status and axis number of control axis, and user can not modify them, but bit0-7 specifies the event status

of control axis. Once the events occur, the corresponding bit will be set as 1, and will be kept until GT_ClrSts is called to clear the status bit.

Table 4-2 Definition of Continuous Trajectory Motion Status Register of GE-X00-SX

Bit	Definition
Bit0	Reserved.
Bit1	Alarm bit of motor servo driver. If the driver of control axis alarms, the bit = 1.
Bit2	Reserved
Bit3	Index/Home triggering bit. After setting a position capture command, when the controller detects required Index/Home capture conditions, this bit = 1.
Bit4	Reserved
Bit5	Triggering bit of positive limit switch. If the switch is triggered, this bit = 1.
Bit6	Triggering bit of negative limit switch. If the switch is triggered, this bit = 1.
Bit7	Reserved.
Bit8	Reserved.
Bit9	Motor servo activation/prohibition status (1 means activation and 0 means prohibition).
Bit10	Motion status symbol bit. It indicates continuously whether the control axis is in motion. If it is in motion, the bit = 1. If in static, the bit = 0.
Bit11	Limit switch activation/prohibit status (1 means activation and 0 means prohibition).
Bit12 Bit13	Indicate the number symbol of current axis. The coding of the current axis numbers is as follows. Bit13 Bit12 Axis 0 0 1 0 1 2 1 0 3
Bit14	Indicate Home switch signal capture status. (1 means activation and 0 means prohibition).
Bit15	Indicate Index signal capture status. (1 means activation and 0 means prohibition).

Table 4-3 Definition of Axis Status Register of GE-X00-PX

Bit	Definition
0	Reserved.
1	Alarm bit of motor servo driver. If the driver of control axis alarms, the bit = 1.
2	Reserved.
3	Index/Home triggering bit. After setting a position capture command, when the controller detects required Index/Home capture conditions, this bit = 1.
4	Motion error bit. If the position error exceeds the allowed scope (Refer to 2.3.1 Description.), the controller will set this bit as 1. Only when the controller is not in motion error status any more, this bit can be cleared.

Bit	Definition
5	Triggering bit of positive limit switch. If the switch is triggered, this bit = 1.
6	Triggering bit of negative limit switch. If the switch is triggered, this bit = 1.
7	Command error bit. If there is error command, the controller will set it as 1.
8	Open loop/close loop status of motor (1 means close loop and 0 means open loop.)
9	Motor servo activation/prohibition status (1 means activation and 0 means prohibition.)
10	Motion status symbol bit. It indicates continuously whether the control axis is in motion. If it is in motion, the bit = 1. If in static, the bit = 0.
11	Limit switch activation/prohibit status (1 means activation and 0 means prohibition.)
12~13	Reserved.
14	Indicate Home switch signal capture status. (1 means activation and 0 means prohibition.)
15	Indicate Index signal capture status. (1 means activation and 0 means prohibition.)
16~26	Reserved.
27	Coder error bit. If there is error, the bit will be set as 1.
28~30	Indicate the number symbol of current axis. The coding of the current axis numbers is as follows. Bit30 Bit29 Bit28 Axis 0 0 0 1 0 0 1 2 0 1 0 3 0 1 1 4 1 0 0 5 1 0 1 6 1 1 0 7 1 1 1 8
31	Reserved.

4.4 Control Axis Mode Register

GE-X00-PX motion controller provides a 16-bit mode register for each axis. The user call command GT_GetMode to enquire the work mode of the control axis. The definition of mode register of motion axis is listed as follows:

Table 4-4 Definition of Axis Mode Register of GE-X00-PX

Bit	Definition
0-6	Reserved.
7	Sign of allowing auto stop when position error overreach error band. The bit = 1 means that, when position error of specified axis overreach error band, the controller will auto stop this axis and inactivate the driver of this axis
8-10	Reserved.
11-13	The coding of specified axis motion mode: Bit13 Bit12 Bit11 Mode of Motion 0 0 0 T-curve of independent positioning 0 0 1 Independent jogging 0 1 0 S-curve of independent positioning
14-15	Reserved

4.5 Continuous Trajectory Motion Status Register

GE-X00-SX motion controller provides a register to record the continuous trajectory motion status. This register is managed by motion controller. The user can call GT_GetCrdSts to read the register, but the application routine can not directly clear or set the register status. The definition of continuous trajectory motion status register is listed as follows：

Table 4-5 Definition of Bits in Continuous Trajectory Status Register

Bit	Definition
0	Specify trajectory motion status, 0 motion; 1 no coordinated motion.
1	Whether the buffer is opened, 0 open; 1 close (default).
2	Whether the pretreatment is normal, 0 normal (default); 1 the sample time (interpolation period) is too short to cause motion error.
3	Reserved.
4	Whether the current segment path motion is completed. 0 motion; 1 completion (default).
5~6	Reserved.
7	Specify the command input status, 0 buffer command input or execution status; 1 status of single-segment path motion (default).
8	Reserved.
9	Whether the corresponding axis is abnormal (the triggering of limit switch), 0 normal (default); 1 abnormal and will automatically stop the trajectory motion.
10	Whether the pulse output is abnormal, 0 normal (default); 1 abnormal.
11～12	Reserved.
13	Whether the buffer is empty, 0 not empty; 1 empty (default).
14～15	Reserved.

Chapter 5 Independent Axis Motion (GE-X00-PX only)

GE-X00-PX motion controller provides several modes of motion: **independent positioning (including S-curve and T-curve) and independent jogging..**

In T-Curve mode, the target position and velocity can be modified on-the-fly. In S-Curve mode, only the target position can be modified during the motion.

In independent jogging mode, the target velocity and acceleration can be modified on-the-fly.

The motion between the specified axes is independent, and each axis follows its own profile. The description in details below will guide you to the appropriate mode and parameters of motion.

5.1 Independent Positioning: T-Curve

5.1.1 Command Summary

Table 5-1 Command Summary of T-Curve

Command	Description
GT_PrflT	Specify the motion mode of current axis as T-Curve
GT_SetAcc	Specify the acceleration of current axis
GT_SetVel	Specify the velocity of current axis
GT_SetPos	Specify the absolute position of current axis
GT_Update	Update the parameters of current axis

Span of the Parameters of T-Curve

Parameter	Value Scope	Unit
Absolute Position	-1,073,741,824 ～ 1,073,741,823	Pulse
Velocity	0～40,957	Pulse/ms
Acceleration	0～102,400	Pulse/ms2

5.1.2 Example

The servo motor of axis #1 is set as voltage mode, and the motion mode of this axis is T-curve mode.

acceleration = 4m/s2，velocity = 10m/min，absolute position = 200mm，1pulse for 1um .

$$\text{acceleration} = \frac{4m/s^2}{1um/pulse} = \frac{4*10^6 um/(10^3 ms)^2}{1um/pulse} = 4\,pulse/ms^2$$

$$\text{velocity} = \frac{10m/\min}{1um/pulse} = \frac{10*10^6 um/(60*10^3 ms)}{1um/pulse} = 166.7\,pulse/ms$$

$$\text{position} = \frac{200mm}{1um/pulse} = \frac{200*10^3 um}{1um/pulse} = 200000\,pulse$$

Example 5-1 Example of T-Curve

```
void AxisRunT(unsigned short axis, long pos, double vel, double acc)
{
    GT_PrflT(axis);                          //specify T-Curve
    GT_SetPos(axis,pos);                     //specify target position
    GT_SetVel(axis,vel);                     //specify target velocity
    GT_SetAcc(axis,acc);                     //specify acceleration
    GT_Update(axis);                         //update parameters
}

void CommandHandle(char *command,short error)
{
    switch(error)
    {
    case   -1:
    printf("\a\nCommunication Error !");    break;
    case    0:
    break;
    case    1:
    printf("\a\nCommand Error !");  break;
    case    7:
    printf("\a\nParameter Error !");  break;
    default   :
    printf("\a\nError Code = %d",error);     break;
    }
}

void main()
{
    GT_HookCommand(CommandHandle);           //hook command handler
    GT_Open(0x300)                           //open motion controller
    GT_Reset();                              //reset motion controller
```

```
    AxisInitial(1,0,10);                    //initialize axis
    AxisRunT(1,200000,166.7,4);             //run in T-Curve mode
}
```

5.1.3 Notes of Main Point

A typical velocity profile of T-curve is described as below (see Fig. 5-1).

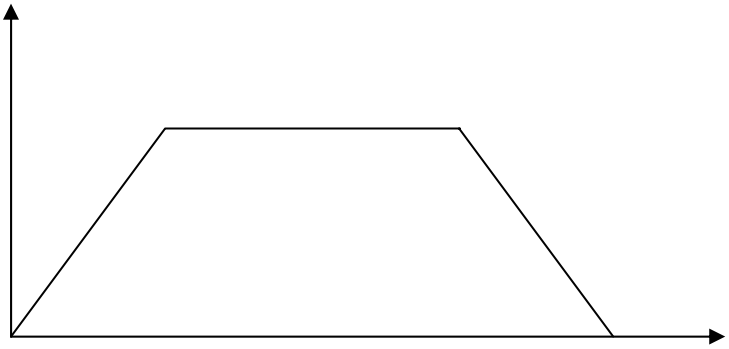


Fig5－1

1. Accelerate stage: the current axis accelerates from zero to the target velocity according to the specified acceleration.
2. Constant velocity stage: the current axis runs at target velocity.
3.Deceleration stage: the current axis decelerates from the target velocity to zero according to the specified acceleration.

In T-curve mode, the target velocity and position can be modified on-the-fly.

The motion controller will recalculate the decline point when a new target position is received on-the-fly. If the new decline point beyond the current position, the motion controller will begin to decelerate until the new decline point reaches. Otherwise, the motion controller will decelerate to zero immediately and then reverse to the new target position (see *Fig. 5*-2, *Fig. 5*-3 and *Fig. 5*-4).

5.2 Independent Positioning: S-Curve

5.2.1 Command Summary

Table 5-2　Command Summary of S-Curve

Command	Description
GT_PrflS	Specify the motion mode of current axis as S-Curve
GT_SetJerk	Specify the jerk of current axis
GT_SetMAcc	Specify the max acceleration of current axis
GT_SetVel	Specify the target velocity of current axis
GT_SetPos	Specify the target position of current axis

Command	Description
GT_Update	Update the parameters of current axis

Span of Parameters of S-Curve

Parameter	Value Scope	Unit
Target Position	-1,073,741,824 ～1,073,741,823	Pulse
Target Velocity	0～40,961	Pulse/ms
Max Acceleration	0～102,412	Pulse/ms^2
Jerk	0～256,054	Pulse/ms^3

5.2.2 Example

The servomotor of axis #1 is set as velocity mode, and the motion mode of this axis is S-Curve mode.

Jerk = 1m/s3, acceleration = 4m/s2, target velocity = 10m/min, target position = 200mm, 1 pulse for 1um.

$$\text{jerk} = \frac{1m/s^3}{1um/pulse} = \frac{10^6 um/(10^3 ms)^3}{1um/pulse} = 0.001 pulse/ms^3$$

$$\text{acceleration} = \frac{4m/s^2}{1um/pulse} = \frac{4*10^6 um/(10^3 ms)^2}{1um/pulse} = 4 pulse/ms^2$$

$$\text{velocity} = \frac{10m/\min}{1um/pulse} = \frac{10*10^6 um/(60*10^3 ms)}{1um/pulse} = 166.7 pulse/ms$$

$$\text{position} = \frac{200mm}{1um/pulse} = \frac{200*10^3 um}{1um/pulse} = 200000 pulse$$

Example 5-2 Example of S-Curve

```
void AxisRunS(unsigned short axis,long pos,double vel,double macc,double jerk)
{
    GT_PrflS(axis);                 //specify S-Curve
    GT_SetPos(axis,pos);            //specify target position
    GT_SetVel(axis,vel);            //specify target velocity
    GT_SetMAcc(axis,macc);          //specify acceleration
    GT_SetJerk(axis,jerk);          //specify jerk
```

```
    GT_Update(axis);                                    //update parameters
}

void CommandHandle(char *command,short error)
{
    switch(error)
    {
        case   -1:
                printf("\a\nCommunication Error !"); break;
        case    0:
                break;
        case    1:
                printf("\a\nCommand Error !");      break;
        case    7:
                printf("\a\nParameter Error !");     break;
        default  :
                printf("\a\nError Code=%d",error); break;
    }
}

void main()
{
    GT_HookCommand(CommandHandle);          //hook command handler
    GT_Open(0x300)                          //open motion controller
    GT_Reset();                             //reset motion controller

    AxisInitial(1,0,0,10);
    AxisRunS(1,200000,166.7,4,0.001);       //run in S-Curve mode
}
```

5.2.3 Notes of Main Point

The motion of S-Curve consists of 7 stages (see Fig. 5-6):

1. Acceleration increase stage: the acceleration of current axis increase from zero to max acceleration according to the specified jerk.

2. Constant acceleration stage: velocity increase according to max acceleration.

3. Acceleration decrease stage: the acceleration of current axis decrease from max acceleration to zero according to the specified jerk.

4. Constant velocity stage: the current axis runs at target velocity.

5. Deceleration increase stage: the deceleration of current axis increase from zero to max deceleration according to the specified jerk.

6. Constant deceleration stage: velocity decrease according to max deceleration.

7. Deceleration decrease stage: the deceleration of current axis decrease from max deceleration to zero according to the specified jerk.

In S-Curve mode, the target position can be modified on-the-fly.

The motion controller will recalculate the decline point when a new target position is received on-the-fly. If the new decline point beyond the current position, the motion controller will begin to decelerate until the new decline point reaches. Otherwise, the motion controller will decelerate to zero immediately and then reverse to the new target position.

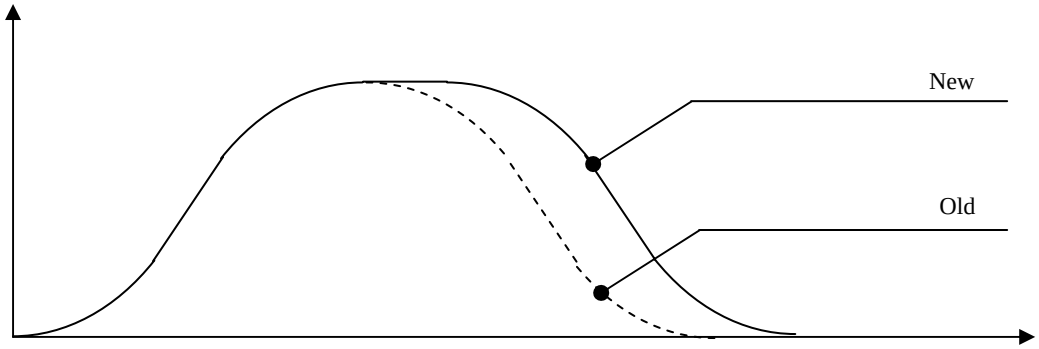


Fig 5-6 Increase target position

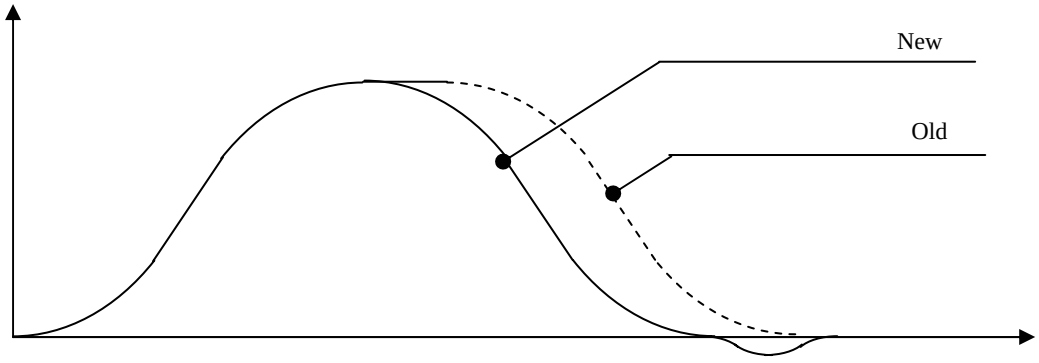


Fig 5-7 Decrease target position

5.3 Jogging Mode

5.3.1 Command Summary

Table 5-3 Command Summary of Jogging Mode

Command	Description
GT_SetAcc	Specify the acceleration of current axis
GT_SetVel	Specify the target velocity of current axis
GT_Update	Update the parameters of current axis

Span of Parameters of Jogging mode

Parameter	Value Scope	Unit
Target Velocity	－40963～40,961	Pulse/ms
Acceleration	0～102,412	Pulse/ms^2

5.3.2 Example

The servomotor of axis #1 is set as velocity mode, and the motion mode of this axis is set to jogging mode.
Acceleration = 4m/s^2, target velocity = 10m/min, 1 pulse for 1um.

$$\text{Acceleration} = \frac{4m/s^2}{1um/pulse} = \frac{4*10^6 um/(10^3 ms)^2}{1um/pulse} = 4\,pulse/ms^2$$

$$\text{Velocity} = \frac{10m/\min}{1um/pulse} = \frac{10*10^6 um/(60*10^3 ms)}{1um/pulse} = 166.7\,pulse/ms$$

Example 5-3 Example of jogging mode

```
void AxisRunV(unsigned short axis, double vel, double acc)
{
    GT_PrflV(axis);                    //specify jogging mode
    GT_SetVel(axis,vel);               //specify target velocity
    GT_SetAcc(axis,acc);               //specify acceleration
    GT_Update(axis);                   //update parameters
}

void CommandHandle(char *command, short error)
{
    switch(error)
    {
        case  -1:
```

```
                printf("\a\nCommunication Error !"); break;
        case    0:
                break;
        case    1:
                printf("\a\nCommand Error !");      break;
        case    7:
                printf("\a\nParameter Error !");     break;
        default  :
                printf("\a\nError Code=%d",error); break;
    }
}

void main()
{
    GT_HookCommand(CommandHandle);          //hook command handler
    GT_Open(0x300)                          //open motion controller
    GT_Reset();                             //reset motion controller
    AxisInitial(1,0,10);
    AxisRunV(1,166.7,4);                    //run in jogging mode
}
```

5.3.3 Notes of Main Point

In jogging mode, the velocity increases to the specified target velocity according to the specified acceleration. Then the velocity will be kept until the new target velocity command received. The target velocity can be modified on-the-fly.

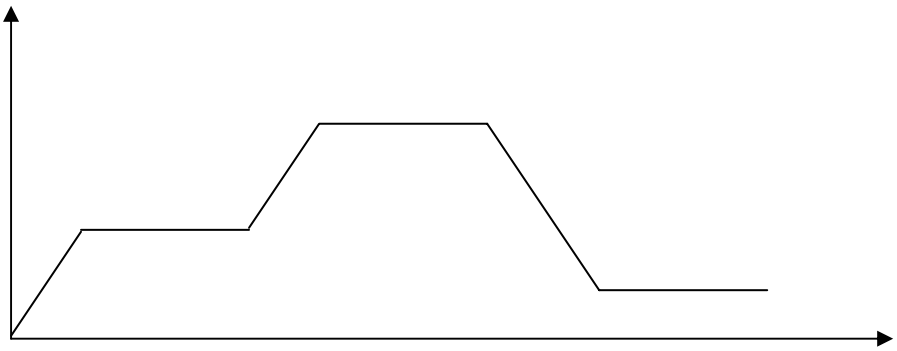


Fig5-8 Modify target velocity

5.4 Parameters Updating

The commands of GE-X00-PX motion controller can be divided into two groups: immediate commands and updating commands.

The immediate commands take effect immediately after the command is sent successfully. The updating commands will not take effect until the GT_Update or GT_MltiUpdt command is invoked. The parameters of current axis will take effective simultaneously after the GT_Update command is invoked. The parameters of several axes will take effective simultaneously after the GT_MltiUpdt command is invoked.

Table 5-4 Updating Commands Summary

指令	说明
GT_SetPos	Specify the target position of current axis
GT_SetVel	Specify the target velocity of current axis
GT_SetAcc	Specify the acceleration of current axis
GT_SetMAcc	Specify the max acceleration of current axis (S-Curve only)
GT_SetJerk	Specify the jerk of current axis (S-Curve only)
GT_SetKp	Specify kp of current axis
GT_SetKi	Specify ki of current axis
GT_SetKd	Specify kd of current axis
GT_SetKvff	Specify kvff of current axis
GT_SetKaff	Specify kaff of current axis
GT_SetMtrBias	Specify the motor bias of current axis
GT_SetMtrLmt	Specify the output voltage limit of current axis
GT_SetILmt	Specify the integral limit of current axis
GT_SetPosErr	Specify the limit of difference between profile position and actual position from encoder

5.5 Stop Motion

5.5.1 Command Summary

Table 5-5 Command Summary of Stop Motion

Command	Description
GT_SmthStp	Stop motion of current axis smoothly
GT_AbptStp	Stop motion of current axis immediately

5.5.2 Notes of Main Point

The command GT_SmthStp() stop the motion of current axis smoothly, i.e. decelerates the velocity to zero by specified acceleration (see Fig.5-9). The current axis will not start to move until the target velocity is received.

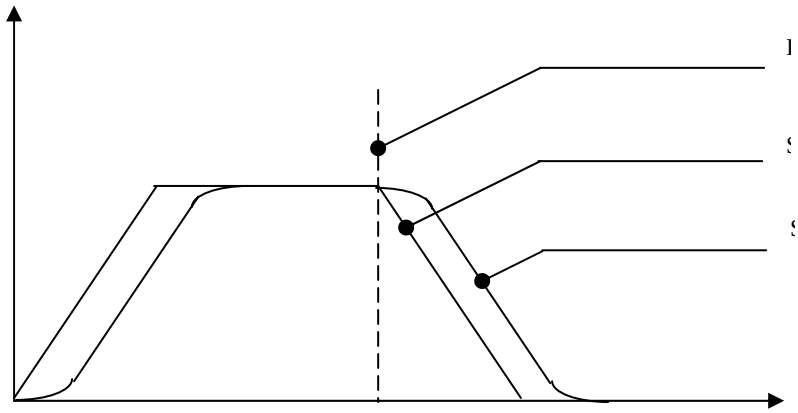


Fig 5-9 Smooth stop

The current axis will stop immediately without deceleration when the GT_AbptStp command is executed.

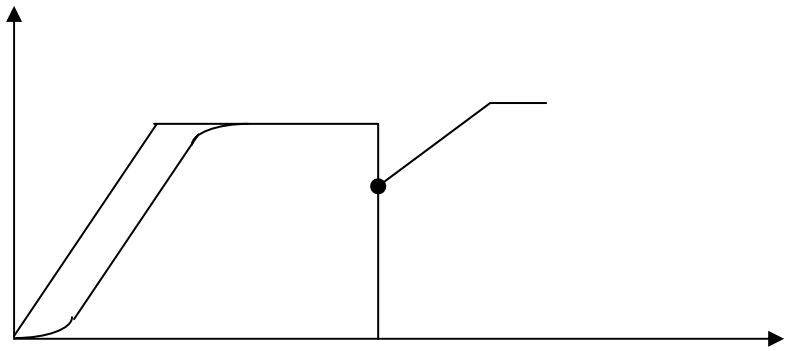


Fig 5-10 abrupt stop

Chapter 6 Interpolation Motion

GE-X00-SX provides interpolation motion, which consists of linear interpolation for 2 or more axes and circular interpolation for 2 axes. Motion along the path should be continuous and smooth at the prescribed path velocity with “Look Ahead” function.

6.1 Specifying path velocity and acceleration

6.1.1 Command Summary

Table 6-1 Command summary of Specifying path velocity and acceleration

Command	Description
GT_SetSynVel	Specify path velocity
GT_SetSynAcc	Specify path acceleration
GT_SetStrtVel	Specify start velocity
GT_SetStpAcc	Specify stop acceleration
GT_SetMaxVel	Specify maximum velocity

6.1.2 Example

Example 6-1 Specifying path velocity and acceleration

This Example shows how to specify path velocity and acceleration with specified unit meaning:

Suppose axis lead is 5mm per revolution, the motor move 10000 pulse per revolution (after quadrupling).

Path velocity (WorkVel) is 3m/min，acceleration (WorkAcc) is 0.9m/s^2，start velocity (StartVel) is 300 Hz (pulse/s)，maximum velocity (MaxVel) is 15m/min，stop acceleration (StopAcc) is 3m/s^2。

$$WorkVel = \frac{3000mm}{60*10^3 ms} \times \frac{10000\,pulse}{5mm} = 100\,pulse/ms$$

$$WorkAcc = \frac{900mm}{(10^3 ms)^2} \times \frac{10000\,pulse}{5mm} = 1.8\,pulse/ms^2$$

$$StartVel = \frac{300\,pulse}{10^3 ms} = 0.3\,pulse/ms$$

$$MaxVel = \frac{15000mm}{60*10^3 ms} \times \frac{10000\,pulse}{5mm} = 500\,pulse/ms$$

$$StopAcc = \frac{3000mm}{(10^3 ms)^2} \times \frac{10000\,pulse}{5mm} = 6\,pulse/ms^2$$

```
void MotionInitial()
{
    short rtn;
    double StartVel, MaxVel, StopAcc;
    StartVel=0.3;                          //start velocity: 300HZ
    MaxVel=500;                            //maximum velocity:
    15m/min
    StopAcc=6;                             //stop acceleration:
    3m/s2
    rtn=GT_SetStrtVel(StartVel);           //Specify start velocity
    error(rtn);
    rtn=GT_SetMaxVel(MaxVel);              // Specify maximum
    velocity
    error(rtn);
    rtn=GT_SetStpAcc(StopAcc);             // Specify stop
    acceleration
    error(rtn);

}

void main()
{
    short rtn;
    double WorkVel, WorkAcc;
    AxisInitial(3,0);                      //refer to example
    3-2~3-4
    MotionInitial();
    WorkVel=100;                           // path velocity:
    3m/min
    WorkAcc=1.8;                           // acceleration:
    0.9m/s2
    rtn=GT_SetSynAcc(WorkAcc);             // Specify
    acceleration
    error(rtn);
    rtn=GT_SetSynVel(WorkVel);             // Specify path
    velocity
    error(rtn);
}
```

6.1.3 Notes of Main Point

6.1.3.1 GT_SetStrtVel

This command specifies start velocity, unit: **pulse/ms.** Start velocity is always specified during initializing system parameter.

As to stepper motor, there is a start frequency which means no need to accelerate to path velocity from zero. Default value of start velocity in motion controller is: 500HZ.

6.1.3.2 GT_SetMaxVel

This command specifies maximum velocity, unit: **pulse/ms.** Maximum velocity is always specified during initializing system parameter.

According to the characteristic graph between moment of force and frequency for stepper motor, when working frequency is higher than nominal frequency, the moment of force should descend very quickly, user should specify maximum velocity during initializing based on the condition of loading and the characteristic of motor. Default value of maximum velocity in motion controller is: 256KHZ.

6.1.3.3GT_SetSynVel

This command specifies path velocity, unit: **pulse/ms**. Specified value decides all path's velocity followed (path velocity in single segment path motion mode and in multi-segment path continuous motion mode are different) till recalling this command.

6.1.3.4 GT_SetSynAcc

This command specifies path acceleration, unit: **pulse/ms^2**. If and only if GE-X00-SX is in single segment path motion, this command can be called.

6.1.3.5 GT_SetStpAcc

This command specifies stop acceleration, unit: **pulse/ms^2**. If some abnormal situation has happened during motion, like triggering limit switch or drive alarm, GE-X00-SX should stop motion by specified stop acceleration, to avoid damage motor with emergent stopping from high speed. Stop acceleration always be specified during initializing system parameter.

Notice: Stop acceleration should be larger than path acceleration.

6.2 Specifying Motion Path

6.2.1 Command Summary

表 6-2 Command summary of motion path

Command	Description
GT_LnXYG0	2D linear interpolation with symmetry profile of velocity planning
GT_LnXYZG0	3D linear interpolation with symmetry profile of velocity planning
GT_LnXY	2D linear interpolation
GT_LnXYZ	3D linear interpolation
GT_ArcXY	Circular interpolation in XY plane described with circle center position and angle
GT_ArcXYP	Circular interpolation in XY plane described with the end point position and radius
GT_ArcYZ	Circular interpolation in YZ plane described with circle center position and angle
GT_ArcYZP	Circular interpolation in YZ plane described with the end point position and radius
GT_ArcZX	Circular interpolation in ZX plane described with circle center position and angle
GT_ArcZXP	Circular interpolation in ZX plane described with the end point position and radius

6.2.2 Example

Example 6-2 realizing single segment path motion

This example call motion path description command based on example 6-1, realize single segment path motion.

```
void main()
{
    short rtn;
    double WorkVel, WorkAcc;

    AxisInitial(3,0);                          //reference to
    example3-2～3-4
    MotionInitial();                    // reference to example
    6-1
    WorkVel=100;                        //path velocity 3m/min
    WorkAcc=1.8;                        // path acceleration 0.9m/s2
    rtn=GT_SetSynAcc(WorkAcc);          //specify path acceleration
    error(rtn);
    rtn=GT_SetSynVel(WorkVel);          // specify path velocity
    error(rtn);
    rtn=GT_LnXY(10,10);                 //2D linear interpolation from
```

```
                                                    current
        error(rtn);                                 // position to（10，10）
    }
```

6.2.3 Notes of Main Point

The commands of motion path provided by GE-X00-SX are described in orthogonal coordinate system.

The unit of motion path is “pulse”, user should make sure that counts per revolution of every axis is matched with each other, to ensure match specified velocity and motion path.

Difference between GT_LnXYG0(GT_LnXYZG0) and GT_LnXY(GT_LnXYZ):

1. There is a symmetry profile of velocity planning during path motion described by GT_LnXYG0 or GT_LnXYZG0, it means GE-X00-SX should accelerate by specified acceleration from start velocity to path velocity, when arrive at decelerated point, GE-X00-SX should decelerate by same acceleration to start velocity and stop motion;
2. If and only if GE-X00-SX is in multi-segment path continuous motion mode, the velocity planning profile of path motion described by GT_LnXY or GT_LnXYZ depends on before and after motion path, their path velocity and end velocity.

According to the right-hand rotation rule, the rotating direction of circular interpolation is defined as that, from the “top” of 2-D coordinate plane (i.e. the positive direction of the third axis which is vertical to the 2-D coordinate plane), the counter-clockwise is positive (Fig. 6-1).

In short to remember: Extend the thumb of right hand, and make fist with the other four fingers; the thumb points to the positive direction of the third axis and the direction of the other four fingers is the positive rotating direction. When the coordinate system is a 2D system (X-Y), the positive direction of circular interpolation in XOY coordinate plane is defined as the same way.

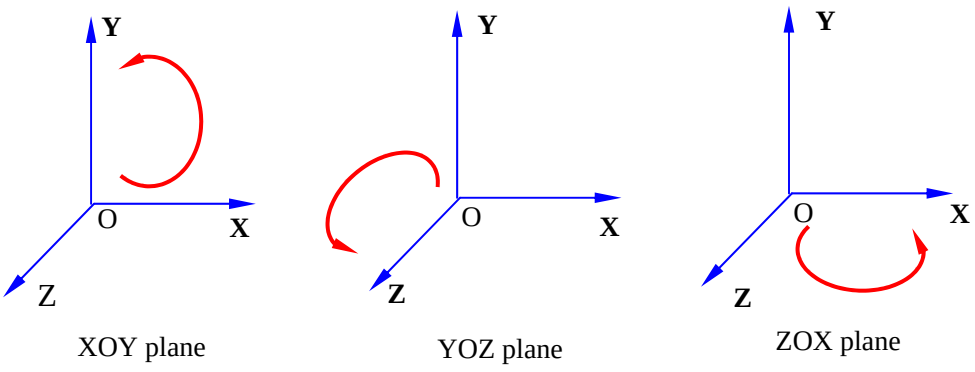


Figure 6-1 Positive Direction of Circular Interpolation

6.3 Multi-segment Continuous Path Motion

Example6-2 just shows how to realize single segment path motion. After sending the last command: rtn=GT_LnXY(10,10), if the machine tool connected with GE-X00-SX is normal (no limit switch triggered, no drive alarm signal), the machine tool should move to the point at (10pulse, 10pulse) immediately.

During the process of single segment path motion (bit0 of return status by calling GT_GetCrdSts is “0”), GE-X00-SX should not accept new path command.

If user wants to realize multi-segment continuous path motion, buffer management command listed below should be used.

GE-X00-SX has a buffer (size: 8k) to temporary store path description and parameter (such as velocity) commands. User can first push command into buffer till it is full, then start to execute motion. During motion, the buffer can accept commands on and on.

GE-X00-SX can realize smooth, continuous, high-speed and high-accuracy motion by enabling “Look-Ahead” function.

The following content will make out how to realize multi-segment continuous path motion by two parts:

- Push motion path and parameter commands into buffer;
- Start to execute multi-segment continuous path motion.

6.3.1 Push Motion Path and Parameters Commands into Buffer

6.3.1.1 Command Summary

Table 6-3 Command Summary of Buffer Management

command	Description
GT_StrtList	Open and clear the buffer
GT_MvXY	Specify the position of start point (2D), path velocity and acceleration in buffer
GT_MvXYZ	Specify the position of start point (3D), path velocity and acceleration in buffer
GT_EndList	Close buffer
GT_AddList	Reopen the buffer

The commands able to be pushed into buffer are listed in table 6-4:

Table 6-4 Summary of Commands able to be pushed into Buffer

command	Description
GT_SetSynVel	Specify the path velocity
GT_LnXYG0	2D linear interpolation (contains symmetry acceleration and deceleration profile)
GT_LnXYZG0	3D linear interpolation (contains symmetry acceleration and deceleration profile)
GT_LnXY	2D linear interpolation
GT_LnXYZ	3D linear interpolation
GT_ArcXY	circular interpolation in XY plane（The parameters are the circle center position and angle）
GT_ArcXYP	circular interpolation in XY plane（The parameters are the end point position and radius）
GT_ArcYZ	circular interpolation in YZ plane（The parameters are the circle center position and angle）
GT_ArcYZP	circular interpolation in YZ plane（The parameters are the end point position and radius）
GT_ArcZX	circular interpolation in ZX plane（The parameters are the circle center position and angle）
GT_ArcZXP	circular interpolation in ZX plane（The parameters are the end point position and radius）
GT_SetDccVel	specify the end velocity of path segment

6.3.1.2 Notes of main point

Open and clear the buffer

GT_StrtList： open and clear the buffer.

Before start to execute multi-segment path continuous motion, GT_StrtList should be called to let the buffer ready for receiving command. This command is valid when the buffer is closed and no multi-segment continuous path motion.

Specify the position of start point, path velocity and acceleration

On the neck of calling the command GT_StrtList() to open and clear the buffer . User **must** call the command (GT_MvXY or GT_MvXYZ) to specify the position of start point, path velocity and acceleration in buffer.

When GT_StrtMtn is called to execute buffer path continuous motion, GE-X00-SX should move from the current position to the position of start point specified along the straight line path, during this motion, it should accelerate from start velocity to path velocity and decelerate to start velocity. Then GE-X00-SX should execute the following path description commands in buffer in order.

Parameters of the commands GT_MvXY or GT_MvXYZ contain path velocity and

acceleration. The path velocity and acceleration specified affects on the following buffer path description commands until GT_SetSynVel is called to specify the path velocity. But path acceleration should not be changed (users are not allowed to modify the value of path acceleration in buffer)。

Close the buffer

GT_EndList：Close the buffer, this command is valid when the buffer is open.

Reopen the buffer

GT_AddList： Reopen closed buffer, then the motion path and parameters commands can be pushed into buffer again.

When no any other motion path and parameters commands should be pushed into the buffer, the command GT_EndList can be called again to close the buffer. User can call the command GT_AddList several times.

Notes of pushing motion path command into buffer

After calling GT_StrtList, user can call GT_EndList at any time to close the buffer. GT_AddList can be called to reopen buffer. Then GT_EndList can be called to close the buffer again. The combination of GT_AddList and GT_EndList can be used several times. When the whole description of motion paths is completed, user must call GT_EndList to close the buffer.

Mechanism of GE-X00-SX processing the path motion commands in buffer

User can push path and parameters commands into buffer continuously until it is full.

After the buffer is full, GE-X00-SX will refuse motion path and parameters commands and return a status shows that the buffer is full.

After the motion described in buffer is started, there should have new space in buffer, allowing more commands to be pushed into.

6.3.2 Start and stop multi-segment continuous path motion

6.3.2.1 Commands Summary

Table 6-5 Command Summary of Starting and Stopping Commands

Command	Description
GT_StrtMtn	Start executing multi-segment continuous path motion
GT_StpMtn	Stop multi-segment continuous path motion smoothly
GT_EstpMtn	Stop multi-segment continuous path motion abruptly
GT_RestoreMtn	Restore multi-segment continuous path motion after pause the

Command	Description
	motion

6.3.2.2 Example

Example 6-3 Realizing multi-segment continuous path motion

```
void main()
{
    short rtn;
    AxisInitial(3,0);                    //reference to example3-2 ~ 3-4
    rtn=GT_StrtList();
    error（rtn）;
    rtn=GT_MvXYZ(0, 0, 0, 100, 1.8);
    error（rtn）;
                                         //specify the start point position:
                                         //（0,0）.
                                         //path velocity 3m/min
                                         //path acceleration 0.9m/s²
    rtn=GT_LnXY(10,10);                  // Specify 2D linear interpolation in //XY plane
    error(rtn);
    rtn=GT_ArcXY(0,0,123);
    error(rtn);
                                         //Specify circular interpolation in //XY plane, the circle center position //is at (0, 0) and the circular angle
                                         //is 123 degree.
    rtn=GT_LnXYZ(0,0,10);                // linear interpolation motion to （0,0,10）//in XYZ plane
    error(rtn);
    rtn=GT_EndList();                    //close the buffer
    error(rtn);
    rtn=GT_StrtMtn();                    //start multi-segment continuous path motion
    error(rtn);
}
```

6.3.2.3 Notes of Main Point

Start multi-segment continuous path motion

The command GT_StrtMtn can be called to start multi-segment continuous path motion.

Before calling the command, user must confirm that there are motion path commands in buffer which can be executed.

After the command GT_StrtMtn is called, GE-X00-SX will execute the commands in buffer in order. During this process, user can push path and parameters commands into the buffer continuously if only the buffer has space until GT_EndList is called.

Stop motion

User can call GT_StpMtn and GT_EStpMtn to stop motion. When GT_StpMtn is called, GE-X00-SX will stop path motion with specified acceleration. When GT_EStpMtn is called, GE-X00-SX will stop path motion with stop acceleration.

When GT_StpMtn (GT_EStpMtn) is called successfully, the buffer should be closed immediately. Just like calling the command GT_EndList。While multi-segment continuous path motion stopped, GE-X00-SX will record the current position and segment number (named break point information). Once user restores motion, GE-X00-SX should return to break point precisely to continue the following path motion described in buffer.

After path motion has been stopped and the buffer is not reopened, the path command called by user will be executed immediately as single segment path motion.

GT_StpMtn and GT_EStpMtn can also stop single segment path motion. But GE-X00-SX should not save break point information for single segment path motion. That is, single segment path motion can not be restored after stopped. User must call path command again to start new single segment path motion.

Restore multi-segment continuous path motion

GT_RestoreMtn can be called to continue multi-segment continuous path motion stopped by GT_StpMtn (GT_EStpMtn) command. After calling GT_RestoreMtn, GE-X00-SX will move to the break point from the current position by linear interpolation and decelerate to start velocity at break point position, and then continue multi-segment continuous path motion in buffer.

Before calling GT_RestoreMtn, user must confirm that multi-segment continuous path

motion had been started and has been paused. GT_StartMtn is not allowed to restore multi-segment continuous path motion.

6.4 Velocity planning strategy for micro-segment continuous path motion

In multi-segment continuous path motion, the path velocity and acceleration is specified in GT_MxXY (GT_MxXYZ). GT_SetSynVel can modify path velocity. If the path velocity is greater than the maximum velocity, GE-X00-SX should set the maximum velocity as the path velocity.

In order to compromise the conflict between high-speed and high-accuracy, “Look Ahead’ strategy of GE-X00-SX should be applied. User can call corresponding command to tell GE-X00-SX about the technological parameters of machine tool, including counts per revolution, maximum velocity, acceleration limit, and maximum time of turning the corner. With Look Ahead function, GE-X00-SX can realize micro-segments continuous path motion in high-speed and high-accuracy.

Figure6-2 Profile of path velocity with and without Look Ahead function

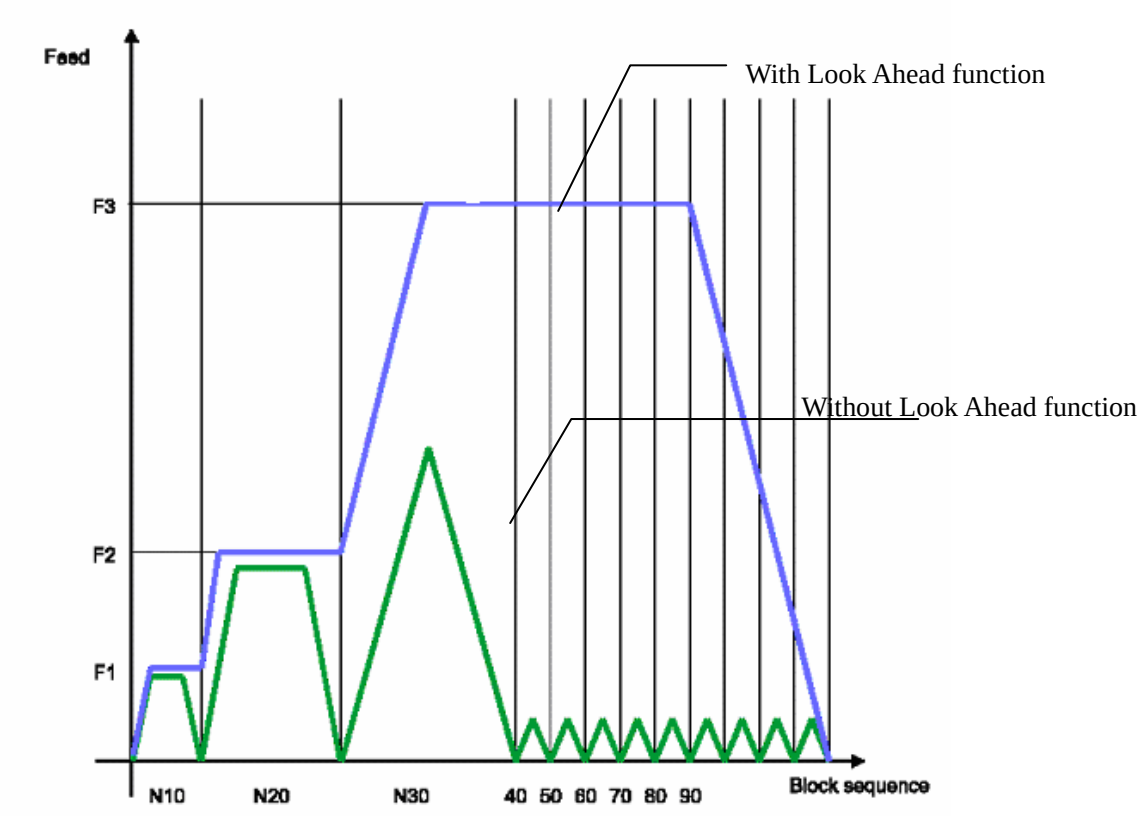


Figure 6-2 Profile of path velocity with and without Look Ahead function

6.4.1 Commands Summary

Table 6-6 Command Summary of Look Ahead function

command	description
GT_InitLook Ahead	Initialize the technological parameters of machine tool
GT_AddLookData	Push path parameters to Look Ahead block
GT_CalVel	Calculate segment end velocity
GT_SetDccVel	Specify segment end velocity

6.4.2 Notes of main points

6.4.2.1 Initialize the technological parameters of machine tool

GT_InitLook Ahead：Initialize the technological parameters of machine tool

Table 6-7 Parameter Summary of initialization command

Parameter	function	Description
T	Maximum time of turning the corner (unit: s)	Longer T, higher velocity, while lower accuracy. On the contrary, shorter T, lower velocity, while higher accuracy
acc_max	Maximum acceleration （unit：mm/s^2）	Related with machine tool and motor, experience value: 100~10000 mm/s^2
acc_run	Path acceleration (unit: mm/s^2)	Path acceleration must smaller than maximum acceleration
vel_run	Path velocity (unit: mm/s)	

Parameter **T** and **acc_max** will affect the position error on the corner

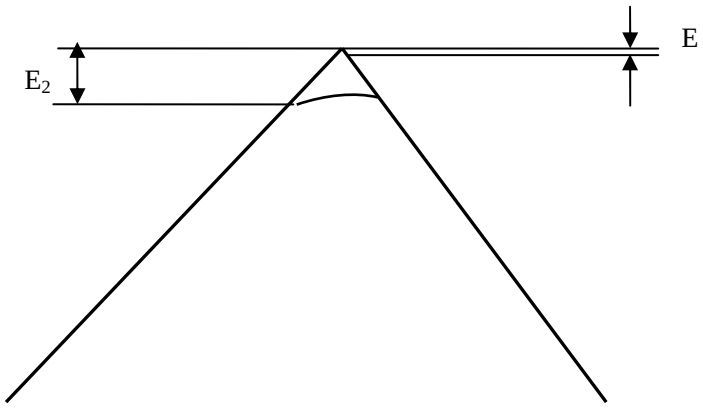


Figure 6-3 position error with different T and Acc_max

See figure 6-3，when **T** or **acc_max** is longer and larger, it will lose some position

accuracy on the corner (E_2). When **T** and **acc_max** are specified rationally，there should have a reasonable position accuracy (E).

6.4.2.2 Push path parameters into Look Ahead block

GT_AddLookData: Push path parameters into Look-Ahead block

Table 6-8 Parameter Summary of GT_AddLookData Command

Parameter	Definition	Description	
Code	Code of path curve	0	G00 Line
		1	G01 Line
		2	G02 Clockwise arc
		3	G03 Counter clockwise arc
PlaneGroup	Code of plane	17	XY Plane
		18	ZX Plane
		19	YZ Plane
R	Radius of arc	if the segment is not arc，R equal to be 0	
x，y，z	Segment end point position	Unit: mm	
F	Feed speed	segment feed speed can be different (unit：mm/s)	
cx，cy	Position of the arc center	if the segment is not arc，Specify **cx** and **cy** to be 0	

6.4.2.3 Calculate the end velocity

GT_Calvel： Calculate the end velocity of the current segment in the Look Ahead block. If the return value is 0, Look Ahead function should be going on. If the return value is 1, Look Ahead function is finished.

6.5 Feed Override of Interpolation Motion

6.5.1 Commands Summary

Table 6-8 Command Summary of feed override

command	description
GT_OverrideG0	Modify the feed override of GT_LnXYG0, GT_LnXYZG0, GT_MvXY or GT_MvXYZ
GT_Override	Modify the feed override of the other motion path description command

6.5.2 Notes of main points

GT_Override and **GT_OverrideG0** can be called to modify the feed override, which means feed speed can be modified on the fly. Once the override has been modified successfully, the override will affect the feed speed of the path motion followed until GT_Override or GT_OverrideG0 is called to modify the feed override again.

The default feed override is 100%.

The feed override can be modified between 0%~200%. If user modified the feed override and made the feed speed beyond maximum velocity. GE-X00-SX set the maximum velocity as the feed speed.

6.6 Machine Coordinate System and Working Coordinate System

6.6.1 Commands Summary

Table 6-9 command summary of specifying coordinate offset

Command	description
GT_MapCnt	Specify the coordinate offset of the specified axis

6.6.2 Notes of main points

Machine coordinate system set machine zero position as its base point. Working coordinate system is related to working reference point. In default, both coordinate systems are same. When user wants to set up working coordinate system, GT_MapCnt can be called to specify the coordinate offset.

6.7 Get Information of the Interpolation Motion

6.7.1 Commands Summary

Table6-10 Command Summary of the interpolation motion

Command	Description
GT_GetPrfPnt	Get the planning position of all axes
GT_GetAtlPos	Get the actual position of specified axis
GT_GetSegPnt	Get the end point position of current working segment
GT_GetPrfVel	Get the planning velocity

Command	Description
GT_GetCrdSts	Get the motion status of interpolation motion
GT_GetSts	Get the status of specified axis
GT_GetMtnNm	Get the working segment number
GT_GetBrkPnt	Get the break point position of all axes after stopping multi-segment continuous path motion

6.7.2 Notes of the main points

6.7.2.1 Get the planning position and actual position

GT_GetPrfPnt：Get the planning position of all axes in interpolation motion.

GT_GetAtlPos：Get the actual position of the specified axis

When GE-X00-SX is in the pulse output mode, it is in open loop control mode. In default, GE-X00-SX disables the encoder input.

User can call **GT_EncOn** to enable encoder input and call GT_GetAtlPos to get the actual position from encoder.

If the encoder input is disabled, GT_GetAtlPos returns planning position of specified axis.

6.7.2.2 Path segment number

GT_GetMtnNm: Get the working path segment number.

Rules of numbering path segment

1. When it is started to execute the path commands in buffer, the path segment number will increase by degrees with the path command executed one by one. The segment number of GT_MvXY and GT_MvXYZ is 0.
2. Segment number will count from 1 to 32767. When segment number reaches 32768, it will be overflow and recount from 1.
3. When the path commands in buffer has been finished, but user does not call GT_EndList to close buffer，the bit0 and bit1of interpolation motion status should be 0, path segment number should keep as the number of finished path.
4. If user calls GT_StpMtn or GT_EStpMtn to stop the multi-segment continuous path motion, Bit1 and Bit0 of interpolation motion status will be 1. The segment number should keep as the number of stopped path until the multi-segment continuous path motion is restored.
5. If GT_StrtList is called, the segment number should be set to be 0.

6.7.2.3 Interpolation motion status and other status

GT_GetCrdSts： Get the interpolation motion status. The definition of every bit in the status register is shown in table 6-11.

Table6-11 Definition of the interpolation motion status

BIT	Definition
0	Motion status. 0: in interpolation motion, 1: no interpolation motion (default).
1	Buffer opened or not. 0: opened, 1: closed (default)
2	Pre-handling is normal or not. 0: normal (default), 1: abnormal.
3	Reserved.
4	Status of current segment path motion. 0: current segment path motion is going on, 1: current segment path motion has finished（default）.
5~6	Reserved.
7	Command input mode. 0: pushing path and parameters commands into buffer. 1: path and parameters command should be executed immediately (default).
8	Reserved
9	Axis working status is abnormal or not. 0: axis working status is normal（default）. 1: axis working status is abnormal such as limit has been triggered, GE-X00-SX will stop interpolation motion.
10	pulse frequency is overflow or not. 0: normal (default), 1: abnormal.
11~12	Reserved
13	Buffer is empty or not. 0: not empty, 1: empty (default).
14~15	Reserved

GT_GetSts： This command can be called to get the specified axis status. If axis limit switch is triggered or there is a drive alarm signal for some axis, GE-X00-SX will automatically stop interpolation motion and set the corresponding bits of reference axes status to be 1.

Chapter 7 High Velocity Home/Index Capture

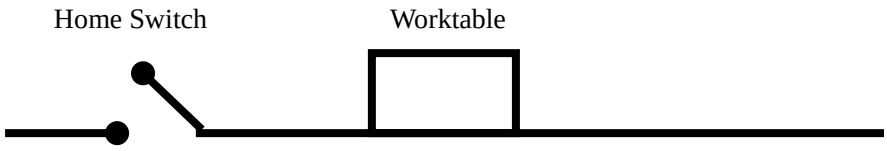
7.1 Command Summary

Table4-1 Function list of Home/Index capture

Function	Description
GT_CaptHome	Set the capture mode as Home capture.
GT_CaptIndex	Set the capture mode as Index capture.
GT_GetCapt	Get the position where the capture occurs.

7.2 Go to zero point by home mode

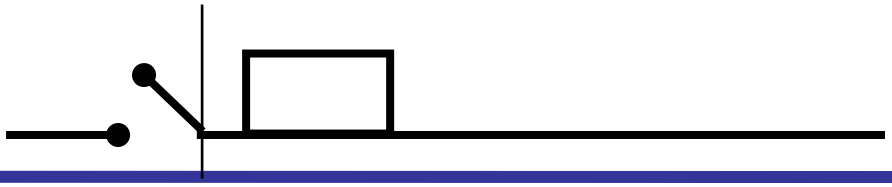
(1)The worktable moves toward home switch, and start the Home capture.



(2)When home signal comes, the worktable is stop smoothly.

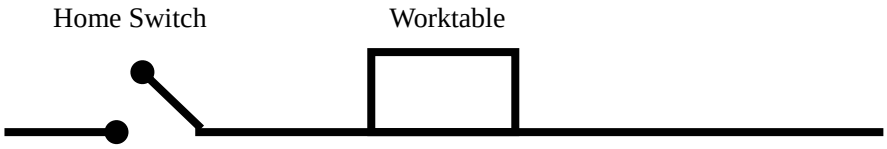
(3)User call command GT_GetCapt to get the actual position of worktable when home signal occurs, and then worktable will be moved to this position.

(4)Worktable is continuously moved toward opposite direction for a specified distance and leaves the limit switch. After the worktable is stopped firmly, User call GT_ZeroPos to clear the worktable position to zero, and setups the machine coordinates system.



7.3 Go to zero point by Home+Index mode

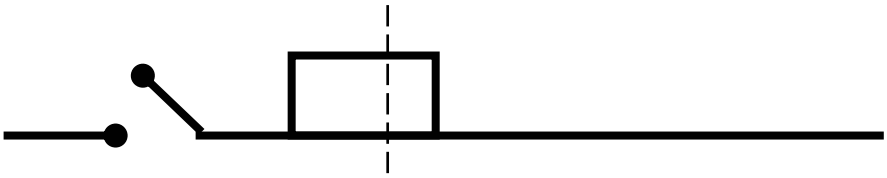
(1)The worktable moves toward home switch, and start the Home capture.



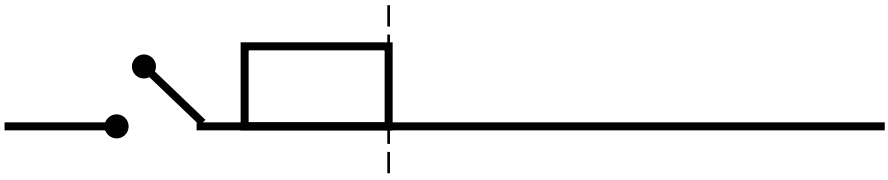
(2)When home signal comes, the worktable is stop smoothly.

(3)User call command GT_GetCapt to get the actual position of worktable when home signal occurs, and then worktable will be moved to this position.

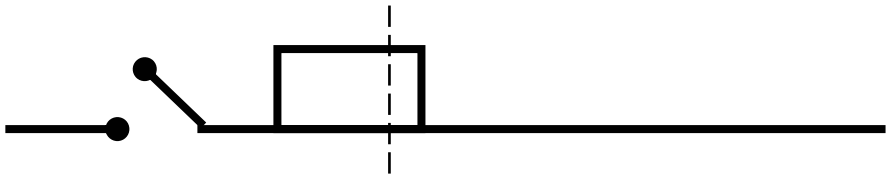
(4)Start Index capture, and move for one and half circle continuously. Stop the worktable smoothly when Index signal comes.



(5)User call command GT_GetCapt to get the actual position of worktable when Index signal occurs, and then worktable will be moved to this position.



(6)Move toward opposite direction for a specified distance and eliminate the gas of opposite direction of lead screw. After the worktable is stopped firmly, User call GT_ZeroPos to clear the worktable position to zero, and setups the machine coordinates system.



7.4 Example

Example7-1　Go to zero point by Home mode（For GE-300-SG motion controller）

```
void CommandHandle(char *command,short error)
{
    switch(error)
    {
        case  -1:
            printf("\a\nCommunication Error !");          break;
        case  0:
            break;
        case  1:
            printf("\a\nCommand Error !");                break;
        case  2:
            printf("\a\nRadius or chord is 0 !");         break;
        case  3:
            printf("\a\nLength is 0 or overflow !");       break;
        case  4:
            printf("\a\nVelocity or acceleration is less then 0 !");
                                                          break;
        case  5:
            printf("\a\nChord is greater than diameter !"); break;
        case  7:
            printf("\a\nParameter error !");                break;
        default:
            printf("\a\nError Code = %d",error);           break;
    }
}

short Home(unsigned short axis,long pos,long offset,double vel_high,double
vel_low)
{
    double prf_pos[4];
    unsigned short status,crd_status;
```

```
GT_ClrSts(axis);                          //Clear status.
GT_CaptHome(axis);                        //Start Home capture
GT_SetSynVel(vel_high);                   //Set velocity of going home
GT_SetSynAcc(0.01);                       //Set acceleration of going
                                            home
GT_GetPrfPnt(prf_pos);                    //Read the profile position
prf_pos[axis-1] = pos;                    //Set target position
GT_LnXYZ(prf_pos[0],prf_pos[1],prf_pos[2]);
do
{
      GT_GetCrdSts(&crd_status);          //Read status of coordinates
                                            system
      if(crd_status&1)                    //if motion has completed
      {                                   //Without triggering home
                                            signal
            return 1;                     //return error code1
      }
      GT_GetSts(axis,&status);            //Read axis status
}while(!(status&0x8));                    //wait for the home to trigger
GT_StpMtn();                              //Stop smoothly
do
{
      GT_GetCrdSts(&crd_status);
}while(!(crd_status&1));                  // wait for the motion to
                                            complete
GT_GetCapt(axis,&pos);                    //Read the capture position
GT_SetSynVel(vel_low);                    //Move to zero point at low
                                            speed
prf_pos[axis-1]= pos;
rtn=GT_LnXYZ(prf_pos[0],prf_pos[1],prf_pos[2]);
do
{
      GT_GetCrdSts(&crd_status);
}while(!(crd_status&1));
prf_pos[axis-1]+= offset;                 //Move a distance and leave
                                            Home switch
GT_LnXYZ(prf_pos[0],prf_pos[1],prf_pos[2]);
do
{
      GT_GetCrdSts(&crd_status);
}while(!(crd_status&1));
```

```
        delay(1000);                                    //Delay a period of time

        GT_ZeroPos(axis);                               //Clear the worktable position
                                                          to zero and setup the machine
                                                          coordinates system
        return 0;
}

void main()
{
        GT_HookCommand(CommandHandle);
        GT_Open();                                      //Open motion controller
        GT_Reset();                                     //Reset motion controller
        AxisInitial(1,0);                               //Cite example3-2 or 3-3
        if(0!=Home(1,-1000000,1000,20,2))               //Call the function of going
                                                          home
        {
              printf("\n Home Error !");
        }
}
```

example7-2 Going to zero point by Home＋Index mode（ for GE-800-PV）

```
void CommandHandle(char *command,short error)
{
        switch(error)
        {
                case   -1:
                        printf("\a\nCommunication Error !");          break;
                case   0:
                        break;
                case   1:
                        printf("\a\nCommand Error !");                break;
                case   7:
                        printf("\a\nParameter error !");              break;
                default:
                        printf("\a\nError Code = %d",error);          break;
        }
}

short Home(unsigned short axis,long pos,long offset,double vel_high,double
vel_low)
{
```

```
unsigned long status;

GT_ClrSts(axis);                        //Clear status
GT_CaptHome(axis);                      //Set Home capture
GT_SetPos(axis, pos);
GT_SetVel(axis,vel_high);               //Set high velocity of going
                                          home
GT_SetAcc(axis,0.01);
GT_Update(axis);

do
{
    GT_GetSts(axis,&status);            //Read axis status
    if(!(status&0x400))
    {                                   //Motion has stopped
        return 1;                       //Home signal has not
                                          triggered
    }
}while(!(status&0x8))                   //Wait for home to trigger

GT_SmthStp(axis);                       //Stop smoothly

do
{
    GT_GetSts(axis,&status);
}while(status&0x400);                   //Wait for motion to complete

GT_GetCapt(axis,&pos);                  //Read the capture position
GT_SetPos(axis,pos);                    //Move to capture position at
                                          low speed
GT_SetVel(axis,vel_low);
GT_Update(axis);

do
{
    GT_GetSts(axis,&status);
}while(status&0x400);                   //Wait for motion to complete

GT_ClrSts(axis);                        //Clear home capture symbol
                                          bit
GT_CaptIndex(axis);                     //Set Index capture
GT_SetPos(axis,pos+11000);              //Move one and half circle
```

```
    GT_Update(axis);

    do
    {
        GT_GetSts(axis,&status);
        if(!(status&x400))
        {                                   //Motion has completed
            return 2;                       //Index signal has not triggered

        }
    }while(!(status&0x8));                  //Wait for Index to trigger
    GT_SmthStp(axis);                       //Stop smoothly

    do
    {
        GT_GetSts(axis,&status);
    }while(status&0x400);                   //Wait for motion to complete

    GT_GetCapt(axis,&pos);                  //Read the Index capture position

    GT_SetPos(axis,pos);                    //Move to capture position
    GT_SetVel(axis,vel_low);
    GT_Update(axis);

    do
    {
        GT_GetSts(axis,&status);
    }while(status&0x400);                   //Wait for motion to complete
    GT_SetPos(axis,pos+offset);             //Move to the capture position
    GT_Update(axis);
    do
    {
        GT_GetSts(axis,&status);
    }while(status&0x400);                   //Wait for motion to complete
    delay(1000);                            //Delay a period of time

    GT_ZeroPos(axis);                       // Clear the worktable position to zero and setup the machine coordinates system

    return 0;
}

void main()
```

```
{
    GT_HookCommand(CommandHandle);
    GT_Open();                                  //Open motion controller
    GT_Reset();                                 //Reset motion controller
    AxisInitial(1,0);                           //Cite example 3-2 or 3-3
    if(0!=Home(1,-1000000,1000,20,2))           //Call the function of going
                                                  home
    {
        printf("\n Home Error !");
    }
}
```

7.5 Notes of Main Point

GE series motion controllers provide a high-speed position capture register for each axis to save the actual position when the triggering signal comes.

Home signal comes from zero point switch, User calls command GT_CaptHome to set the position capture mode as Home signal triggering. If specified axis has been Home capture mode or Index capture mode and the capture signal has not been triggered, this command can not be called again. If the signal has triggered, it is necessary that user will call GT_ClrSts to clear bit3, and then can call this command again.

Index signal comes from phase C of encoder (once each circle) or linear encoder; User calls command GT_CaptIndex to set the position capture mode as Home signal triggering. If specified axis has been Home capture mode or Index capture mode and the capture signal has not been triggered, this command can not be called again. If the signal has triggered, it is necessary that user will call GT_ClrSts to clear bit3, and then can call this command again.

Home and Index signals are all edge triggering mode. User can call GT_HomeSns to set that which edge (up-edge or down-edge) will be triggered, the default is down-edge. When Home or Index signal occurs, the motion controller will automatically save the actual position where Home or Index signal comes, and will set bit3 as 1, and closes Home or Index capture (bit14/bit15 will be clear to 0). User calls GT_GetCapt to read the capture position of specified axis.

Notes: when user adopts Home+Index mode to go to zero point, the Home switch will be fixed between two Index signals. The overlap of Home and Index signals may bring the error of Index signal triggering.

Chapter 8 Safety Mechanism

8.1 Set automatic stopping when error exceeds limit

8.1.1 Command Summary

Table8-1 Function list of setting automatic stopping

Function	Description
GT_SetPosErr	Set the position error limit.
GT_GetPosErr	Get the position error limit.
GT_AuStpOn	Enable automatic stopping when error motion. (for GE-X00-PX)
GT_AuStpOff	Disable automatic stopping when error motion. (for GE-X00-PX) (default)

8.1.2 Notes of main point

For the close loop servo control, sometimes the actual position of motor may be far from the target position. This situation usually means some dangers existing, such as motor fault, reversed connection or disconnection of A and B signals of encoder, stemming of motor caused by too large mechanical friction or mechanical fault. In order to detect this situation, to improve the safety of the system and prolong the service life of the equipment, motion controller sets programmable and modifiable position error limit of specified axis.

In each sample time, the motion controller compares the position error limit with the actual position error to check if a motion error occurs.

For GE-X00-PX motion controller, if position error overreaches the specified error limit, the motion controller will set bit4 as 1; if user has enabled automatic stopping, the motion controller will automatically abruptly stop motion of this axis.

For GE-X00-SX motion controller, if position error of control axis of continuous trajectory motion overreaches the specified error limit, the motion controller will abruptly stop continuous trajectory motion, and set bit4 of this axis as 1.

8.2 Set output voltage saturation limit

8.2.1 Function List

Table8-2 Function list of setting saturation limit

Function	Description
GT_SetMtrLmt	Set output voltage saturation limit of specified axis.

8.2.2 Notes of Main Point

The default output voltage range of motion controller is +/-10V. In some conditions, the output voltage of control axis need be limited, and user can call command GT_SetMtrLmt to set output voltage saturation value.

8.3 Treat Limit Status

The limit switch can be used to indicate automatically the motion bound of axes. The "safe" motion bound of axis with limit switch is illustrated in *Fig.8-1*.

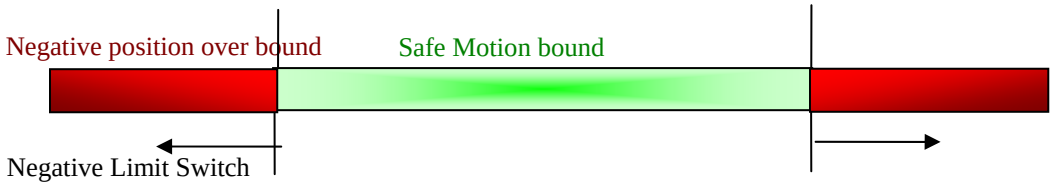


Fig.8-1 Definition of Motion bound of Specified Axis

Once a controlle axis triggers a limit switch, this axis will only be allowed to move towards the opposite direction. For example, suppose that the positive limit switch of a controlled axis be triggered, the controller will only allow the axis to move towards the negative direction, so as to return back to the safe motion range.

After the controlled axis returns back to the safe motion bound, the host must call the command GT_ClrSts() or GT_RstSts() to clear the relevant status bit, to restore the status of controlled axis from the status of over bound to normal status.

It is suggested that going zero point should not adopt limit switch, Home switch should be adopted to go zero point. Please refer to chapter 7.

8.4 Treat Axis Driver Alarm

For the purpose of safety and longer service life of motor and driver, some motor drivers set the detecting and protecting of system fault (such as input voltage limit of driver , protection of too large change for input signal). When the driver detects abnormal or fault

situation, it will trigger the fault protection function and output an alarm signal.

The controller provides a dedicated input of driver alarm signal. When it detects the input signal of motor driver alarm, the controller will set the sign bit of driver alarm and the sign bit of motion completed in the status register of specified axis as 1, and inactivate the motor driver of this axis (GE-X00-PX) or continuous trajectory motion (GE-X00-SX).

After the driver alarm signal comes, the user must take the following steps:

- Confirm the causes of the alarm signal and correct them.
- Use the command GT_DrvRst() to reset the driver
- Going the machine zero point again.

Chapter 9 Digital I/O

9.1 General Purposed I/O

9.1.1 Command Summary

Table9-1 Function list of General Purposed I/O

Function	Description
GT_ExInpt	Get the value of general purposed input port.
GT_ExOpt	Set the value of general purposed output port.

9.1.2 Notes of Main Point

The motion controller provides 16 channels of uncommitted opto-isolated digital input and 16 channels of uncommitted opto-isolated digital output.

16-bit Input Port: The host uses the command GT_ExInpt(&Data) to get the logic level of this port. The corresponding relation between the returned Data and the definition of EXI0-EXI15 bits is as follows.

Bit - Definition	Bit - Definition	Bit - Definition	Bit - Definition
Bit0----EXI0	Bit1----EXI1	Bit2----EXI2	Bit3----EXI3
Bit4----EXI4	Bit5----EXI5	Bit6----EXI6	Bit7----EXI7
Bit8----EXI8	Bit9----EXI9	Bit10----EXI10	Bit11----EXI11
Bit12----EXI12	Bit13----EXI13	Bit14----EXI14	Bit15----EXI15

16-bit Output Port: The host uses the command GT_ExOpt(Data) to set the value of this port. The corresponding relation between Data and the general purposed output port EXO0-EXO15 on the connector CN2 of the controller is as follows.

Bit - Definition	Bit - Definition	Bit - Definition	Bit - Definition
Bit0----EXO0	Bit1----EXO1	Bit2----EXO2	Bit3----EXO3
Bit4----EXO4	Bit5----EXO5	Bit6----EXO6	Bit7----EXO7
Bit8----EXO8	Bit9----EXO9	Bit10----EXO10	Bit11----EXO11
Bit12----EXO12	Bit13----EXO13	Bit14----EXO14	Bit15----EXO15

9.2 Dedicated Digital I/O

9.2.1 Function List

Table8-2 Function list of Dedicated Digital I/O

Function	Description
GT_GetLmtSwt	Get the status of limit switch.
GT_GetHomeSwt	Get the status of Home switch.

9.2.2 Notes of Main Point

The host uses the command GT_GetLmtSwt to get the level status of limit switch. The corresponding relation between status bit of parameters and limit signal input interface of terminal board is as follows.

Bit - Definition	Bit - Definition	Bit - Definition	Bit - Definition
Bit0----Limit0+	Bit1---- Limit0-	Bit2---- Limit1+	Bit3---- Limit1-
Bit4---- Limit2+	Bit5---- Limit2-	Bit6---- Limit3+	Bit7---- Limit3-
Bit8---- Limit4+	Bit9---- Limit4-	Bit10---- Limit5+	Bit11---- Limit5-
Bit12---- Limit6+	Bit13---- Limit6-	Bit14---- Limit7+	Bit15---- Limit7-

The host uses the command GT_GetHomeSwt to get the level status of limit switch. The corresponding relation between status bit of parameters and Home signal input interface of terminal board is as follows.

Bit - Definition	Bit - Definition	Bit - Definition	Bit - Definition
Bit0----Home0	Bit1---- Home1	Bit2---- Home2	Bit3---- Home3
Bit4---- Home4	Bit5---- Home5	Bit6---- Home6	Bit7---- Home7

Chapter 10 Optional Function of Motion Controller

10.1 Data Monitoring of Motion Controller

The user can sample and analyze corresponding data of each axis of motion controller, and adjust PID control parameters and motion parameters, and then the performance of servo system can be represented.

GE serial motion controller can monitor profile position, actual position, profile velocity, actual velocity, and control voltage of each axis. User can specify the monitoring parameters and monitoring frequency.

10.1.1 Function List

Table10-1 Function list of Data Monitoring

Function	Description
GT_SetWatch	Specify the monitoring parameters
GT_GetWatch	Get the monitoring data
GT_StartWatch	Specify the monitoring frequency, and start data monitoring
GT_StopWatch	Stop data monitoring

10.1.2 Example

For GE-800-PV motion controller, monitor profile position and actual position of each sampling period, and write data into files.

Example10-1 Program in Visual C++

```
#include “gep.h”                                    //Cite function library
#define BUFFER_SIZE 400                            //Define buffer size

void __stdcall CommandHandle(char * command,short error)
{                                                  //Error treatment function
    switch(error)
    {
        case  -1:
            printf("\n Communication Error : %s = %d",command,error);
            exit(3);
        case  0:
            return;
        case  1:
            printf("\n Command Error : %s = %d",command,error);
```

```
                exit(2);
        case  7:
                printf("\n Parameter Error : %s = %d",command,error);
                exit(1);
        default:
                printf("\n Error Code = %d ", error);
                exit(1);
    }
}

void main()
{
    FILE *fp;                                   //Create file
    if((fp=fopen("output.txt","wt"))==NULL)
    {
        printf("Can't Open output.txt\n");
        exit(0);
    }
    GT_HookCommand(CommandHandle);      //Hook error treatment function
    GT_Open(0x300);                             //Open motion controller
    GT_Reset();                                 //Reset motion controller
    unsigned short config[AXIS_NUM] = {0,3,0,0,0,0,0,0};
                                                //Refer to table10-2

     GT_SetWatch(config,BUFFER_SIZE);      //Monitor profile positionand
                                                 actual position of axis2
    long PrfPos[BUFFER_SIZE],AtlPos[BUFFER_SIZE];
                                               //Create data buffer
    TWatchBuffer buffer;                       //TWatchBuffer is defined in library
    buffer.Axis[1].PrfPos = PrfPos;            //Hook buffer of profile position
    buffer.Axis[1].AtlPos = AtlPos;            //Hook buffer of actual position
    GT_SetKp(2,10);                             //Set Kp of axis 2
    GT_Update(2);                               //Update parameter of axis 2
    GT_AxisOn(2);                               //Activate axis 2

    printf("\nPress any key to continue…");
    getch();

    GT_SetPos(2,100000);                        //Set target position
    GT_SetVel(2,10);                            //Set target velocity
    GT_SetAcc(2,1);                             //Set acceleration
    GT_StartWatch(0);                           //Start data monitoring
    GT_Update(2);                               //Update parameters

    unsigned short available;
```

```
    unsigned long status;
    do
    {
        GT_GetWatch(&buffer,&available);      //Whether buffer is full
        if(available)                         //If full, save data to file
            {
                for(short i=0;i<BUFFER_SIZE;++i)
                {
                    fprintf(fp,"%d\t%d\n",PrfPos[i],AtlPos[i]);
                }
            }
        GT_GetSts(2,&status);                 //Check status of axis 2
    }while(status&0x400);                     //Wait for axis 2 to stop

    do
                                              //After stopping, monitor some data
                                              again
    {
        GT_GetWatch(&buffer,&available);
        if(available)
        {
            for(short i=0;i<BUFFER_SIZE;++i)
            {
                fprintf(fp,"%d\t%d\n",PrfPos[i],AtlPos[i]);
            }
        }
    }while(!available);

    GT_StopWatch();                           //Stop data monitoring
    GT_Close();                               //Close motion controller
    fclose(fp);                               //Close data file
}
```

Example10-2 Program in Dephi

```
    const
        BUFFER_SIZE = 400;                    //Define buffer size
    procedure CommandHandle(Command:PChar;Error:short);stdcall;
    begin                                     //Error treatment function
```

```
        case error of
            -1:ShowMessageFmt('Communication Error %s = %d',[command,error]);
              0: exit;
              1: ShowMessageFmt('Command Error %s = %d',[command,error]);
              7 :    ShowMessageFmt('Param Error %s = %d',[command,error]);
        end;
end;

procedure TForm1.Button1Click(Sender: TObject);      //When click bottom 1
var
        config :TConfig;                              //TConfig is defined in library
        available : word;
        buffer : TWatchBuffer;                        //TWatchBuffer is defined in library
        status :LongWord;
        i : LongInt;
        list : TStringList;
begin
        list := TStringList.Create;                   //list is used to save data.
        GT_HookCommand(CommandHandle);  //Hook error treatment routine
        GT_Open($300);                                //Open motion controller
        GT_Reset;                                     //Reset motion controller
        config[0] := $0;
        config[1] := $3;                             //Monitor profile and actual position
        config[2] := $0;
        config[3] := $0;
        config[4] := $0;
        config[5] := $0;
        config[6] := $0;
        config[7] := $0;
        GT_SetWatch(config,BUFFER_SIZE);       //Set buffer size
        SetLength(buffer.Axis[1].PrfPos,BUFFER_SIZE);
                                                      //Create buffer of profile position
        SetLength(buffer.Axis[1].AtlPos,BUFFER_SIZE);
                                                     // Create buffer of profile position
        GT_SetKp(2,10);                               //Set Kp
        GT_Update(2);                                 //Update parameters
        GT_AxisOn(2);                                 //Activate axis 2
        ShowMessage('Click button to continue…');

        GT_SetPos(2,100000);                          //Set target position of axis 2
        GT_SetVel(2,10);                              //Set target velocity of axis 2
        GT_SetAcc(2,1);                               //Set acceleration
        GT_StartWatch(0);                             //Start data monitoring
        GT_Update(2);
```

```
    repeat
        GT_GetWatch(buffer,available);      //Whether buffer is full
        if 1 = available then               //If buffer is full
        begin                               //save data in list
            for i:=0 to BUFFER_SIZE - 1 do
            begin
                list.Add(Format('%d    %d',[buffer.Axis[1].PrfPos[i],
                                    buffer.Axis[1].AtlPos[i]]));
            end;
        end;
        GT_GetSts(2,status);                //Check status of axis 2
    until 0 = (status and $400);            //Wait for axis 2 to stop

    repeat                                  / /When Axis 2 stops, monitor some
                                              data again
        GT_GetWatch(buffer,available);
        if 1 = available then
        begin
            for i:=0 to BUFFER_SIZE - 1 do
            begin
                list.Add(Format('%d    %d',[buffer.Axis[1].PrfPos[i],
                                    buffer.Axis[1].AtlPos[i]]));
            end;
        end;
    until (1 = available);
    GT_StopWatch;                           //Stop data monitoring
    GT_Close;                               //Close motion controller
    list.SaveToFile('output.txt');          //Write data into file
    list.Free;
end;
```

Example 10-3 Program in Visual Basic

```
Const BUFFER_SIZE = 400                     //Define buffer size

Private Sub Command1_Click()                //Start monitoring when click bottom
    Dim fso As New FileSystemObject
    Dim ts As TextStream
    Set ts = fso.CreateTextFile("output.txt")  //Create file to save data
    GT_Open &H300                           //Open motion controller
    GT_Reset                                //Reset motion controller
    Dim Config(AXIS_NUM) As Integer         //AXIS_NUM is defined in library
    Config(0) = &H0
```

```
Config(1) = &H3                          //Monitor profile position and actual
                                           position of axis 2
Config(2) = &H0
Config(3) = &H0
Config(4) = &H0
Config(5) = &H0
Config(6) = &H0
Config(7) = &H0
GT_SetWatch Config(0), BUFFER_SIZE       //Set buffer size, and send first
                                           element to function library in citing
                                           mode
 Dim PrfPos(BUFFER_SIZE) As Long         //Create buffer of profile position
Dim AtlPos(BUFFER_SIZE) As Long          //Create buffer of actual position
Dim buffer As TWatchBuffer
buffer.Axis(1).PrfPos = VarPtr(PrfPos(0)) //Hook buffer of profile position
buffer.Axis(1).AtlPos = VarPtr(AtlPos(0))//Hook buffer of actual position
GT_SetKp 2, 10                           //Set Kp
GT_Update 2                              //Update parameters
GT_AxisOn 2                              //Activate axis 2
MsgBox “Click button to continue…”

GT_SetPos 2, 100000                      //Set target position of axis 2
GT_SetVel 2, 10                          //Set target velocity
GT_SetAcc 2, 1                           //Set acceleration
GT_StartWatch 0                          //Start data monitoring

GT_Update 2                              //Update parameters of axis 2
Dim i As Long
Dim status As Long
Dim available As Integer
Do
    GT_GetWatch buffer, available        //Check whether buffer is full
    If available = 1 Then                //if full
        For i = 0 To BUFFER_SIZE－1 //save data in file
            ts.WriteLine PrfPos(i) & "," & AtlPos(i)
        Next i
    End If
    GT_GetSts 2, status                  //Check status of axis 2
Loop Until (status And &H400) = 0        //Wait for axis 2 to stop
Do
                                         //After axis 2 stops, monitor some
                                           data again
    GT_GetWatch buffer, available
    If available = 1 Then
```

```
                    For i = 0 To BUFFER_SIZE－1
                        ts.WriteLine PrfPos(i) & "," & AtlPos(i)
                    Next i
                End If
            Loop Until 1 = available                    //Wait for buffer to be full
            GT_StopWatch                                //Stop data monitoring
            GT_Close                                    //close motion controller
            ts.Close
        End Sub
```

10.1.3 Notes of Main Point

10.1.3.1 Specify the control axis parameters monitored and data buffer size

User can call GT_SetWatch to specify the control axis parameters monitored and data buffer size. This command has two parameters, the first one specifies the control axis parameters, and the detailed definition is as followed table10-2:

When some bit is 1, it means that corresponding parameters will be monitored, and if 0, corresponding parameters will not be monitored.

The second one specifies data buffer size. If specified buffer size is too large, the return value of this command is 7.

The following example will monitor the actual positions of axis 1, 2, 3, and buffer size is 400.

```
unsigned short config[8]={0x2,0x2,0x2,0,0,0,0,0};//Monitor actual position of axis
                                                    1,2,3
short rtn;
rtn=GT_SetWatch(config,400);  //Set monitoring parameters and buffer size
if(rtn!=0)                                          //Check the return value
{
    cout<<"GT_SetWatch =\t"<<rtn<<endl;
    return;
```

Bit - Definition	Bit - Definition	Bit - Definition	Bit - Definition
Bit0----profile position	Bit1----actual position	Bit2----profile velocity	Bit3----actual velocity
Bit4---- reserved	Bit5----reserved	Bit6----control voltage	Bit7----reserved

10.1.3.2 Buffer data structure

Data structure of each axis is as follows：

```
typedef struct
{
    long    *PrfPos;
```

```
        long    *AtlPos;
        double  *PrfVel;
        double  *AtlVel;
        short   *Voltage;
    } TAxis;
```

Data structure of buffer is as follows:

```
  typedef struct
  {
      TAxis Axis[AXIS_NUM];
  } TWatchBuffer;
```

User data monitored must hook corresponding data buffer, or memory access error will be received.

The following routine hook actual positions of axis 1,2,3 to data buffer.

```
  long AtlPos[3][400];                    //Create data buffer
  TWatchBuffer buffer;
  buffer.Axis[0].AtlPos = AtlPos[0];      //Hook buffer
  buffer.Axis[1].AtlPos = AtlPos[1];
  buffer.Axis[2].AtlPos = AtlPos[2];
```

10.1.3.3 Read the buffer data

The command GT_GetWatch will be called to inquire buffer status, and read buffer data. The first parameter specifies the data buffer, and the second one specifies whether the buffer has been full. When the return value of the second parameter is 1, it shows that data buffer has been full, and user can access the data in buffer at this time.

```
  long AtlPos[3][400];                    //Create data buffer
  TWatchBuffer buffer;

  buffer.Axis[0].AtlPos = AtlPos[0];      //Hook actual position buffer of axis
                                            1
  buffer.Axis[1].AtlPos = AtlPos[1];      //Hook actual position  buffer of
                                            axis 1
  buffer.Axis[2].AtlPos = AtlPos[2];      //Hook actual position buffer of axis 1

  short available;
  int status;
  do{
      GT_GetWatch(&buffer,&available);
      If (1==available)                   //Check whether buffer is full
      {
          for(i=0;i<400;++i)              //Write data to file
          {
```

```
                    fprintf(fp,"%ld\t%ld\t%ld\r\n",AtlPos[0][i],AtlPos[1][i],AtlPos[2][i])
                ;
                }
            }
            GT_GetSts(1,&status);                    //Read status of axis 1
        }while(0x400==(status&0x400));               //Wait for axis1 to stop
```

10.1.3.4 “Start/Stop” data monitoring

The command GT_StartWatch can be called to start the data monitoring, and the parameter specifies the number of servo period. The motion controller will sample once every such time.

The command GT_StopWatch can be called to stop data monitoring.

10.2 Analog Voltage Output

10.2.1 Command Summary

Table10-3 Function list of Data Monitoring

Function	Description
GT_GetAdc	Read the analog voltage of specifying channel
GT_SetAdcChn	Set the sampling channel number (GE-X00-SX)

10.2.2 Example

Example 10-4 Read the analog voltage of channel 1

```
void main()
{
    short voltage;
    GT_Open();                                   //Open motion controller
    GT_Reset();                                  //Reset motion controller
    if(0==GT_GetAdc(1,&voltage))                 //Read analog voltage of channel 1
    {
        printf(“\n Channel 1 Voltage : %d”,voltage);
                                                 //Print the analog voltage value of
                                                 channel 1
    }
    else
    {
        printf(“\nCommand Error !”);
    }
    GT_Close();                                  //Close motion controller
}
```

10.2.3 Notes of Main Point

GE serial motion controllers provide 8 channels of independent 12-bit+/-10V analog input. The command GT_GetAdc is used to read the analog voltage value. The corresponding relation between analog voltage and digital is as follows:

Table 10-4 The relation between analog voltage and corresponding digital

Analog voltage (V)	Corresponding digital
+10	2047
0	0
-10	-2048

The sampling frequency of GE-X00-PX is 4KHZ, and that of GE-X00-SX is 770HZ.

GE-X00-SX motion controller provides command GT_SetAdcChn to set the number of channels in order to improve A/D sampling frequency. When there is 1 sampling channel, the sampling frequency will be up to 8×770HZ=6.16KHZ.

10.3 Manual pulse generator function

GE-X00-SX motion controller can directly control motor through manual pulse generator, and realize precise positioning of manual motion mode.

10.3.1 Command Summary

Table10-5 Function list of manual pulse generator function

Function	Description
GT_GetAdc	Read the analog voltage of specifying channel
GT_SetAdcChn	Set the sampling channel number (GE-X00-SX)

10.3.2 Example

Example 10-5 According to the input of general purposed digital IO, user can switch corresponding axis to manual pulse generator control mode.

```
short InMPG()
{
    short rtn;
    unsigned short axis=1;
    double rate=0.1;
    unsigned short CurInpt;
    rtn=GT_ExInpt(&CurInpt);          //Read IO value
    if(rtn!=0) return rtn;
    if(CurInpt & 0x1) axis=1;         //Bit0 is 1, axis 1 is selected
    if(CurInpt & 0x2) axis=2;         // Bit1 is 1, axis 2 is selected
    if(CurInpt & 0x4) axis=3;         //Bit2 is 1, axis 3 is selected
```

```
    if(CurInpt & 0x8) rate=0.001;          //Bit3 is 1, override is set
    if(CurInpt & 0x10) rate=0.01;          //Bit4 is 1, override is set
    if(CurInpt & 0x20) rate=0.1;           / / Bit5 is 1, override is set
    rtn=GT_HandWheel(axis,rate);           //Switch specified axis to
                                           manual pulse generator mode

    if(rtn!=0) return rtn;
}
```

10.3.3 Notes of Main Point

GE-X00-SX provides a channel of dedicated manual pulse generator input port (CN10), and this port may receive single-ended or differential pulse input signal.

Only one axis can stay in manual pulse generator mode. If some axis is switched to manual pulse generator mode, the prevenient axis stayed in manual pulse generator mode will automatically exit the manual pulse generator control mode. User can only switch some axis to the manual pulse generator control mode without any trajectory motion.

When control axis stays in the manual pulse generator control mode, the maximum velocity and acceleration must be set. The control axis will follow the manual pulse generator according to override. The control axis will start at starting velocity specified, and move to target following velocity at specified acceleration. The following velocity is calculated according to following override and the velocity of manual pulse generator. When manual pulse generator motion stops, the control axis will decelerate to starting velocity and stop. If the following velocity calculated exceeds the maximum velocity set, the following velocity will is set as maximum velocity.

For stepper motor, the following velocity of control axis should avoid in syntony section, or the more vibration may be appeared.

10.4 Rotate Speed Control of Spindle

10.4.1 Command Summary

Table10-6 Function list of rotate speed of spindle

Function	Description
GT_SetSpindleVel	Set rotate speed of spindle
GT_CloseSpindle	Close spindle

10.4.2 Example

Example 10-6 Set rotate speed of spindle

```
void main()
```

```
{
    GT_Open();                              //Open motion controller
    GT_Reset();                             //Reset motion controller
    if(0== GT_SetSpindleVel (32767)         //Set rotate speed of spindle
    {
        printf("\n GT_SetSpindleVel OK!");
    }
    delay(10000);                           //Delay 10s
    if(0== GT_CloseSpindle ()               //Close spindle
    {
        printf("\n GT_ CloseSpindle OK!");
    }
    GT_Close();                             //Close motion controller
}
```

10.4.3 Notes of Main Point

GE-X00-SX motion controller provides a channel of dedicated spindle control output port (CN8). The command GT_SetSpindleVel can be called to change output voltage of spindle control immediately to adjust the rotate speed of spindle.

Table 10-7 the relation between output voltage and corresponding digital

Output voltage (V)	Corresponding digital
+10	32767
0	0
-10	-32768

The command GT_CloseSpindle can be called to stop the spindle motion. Do not stop spindle with the command GT_SetSpindleVel with parameter 0, or the spindle cannot maybe stop completely because of voltage bias.

Chapter 11 Tables of Commands

Table 11-1 GE-X00-PX Functions

Function	Meaning
Set Motion Controller Group	
GT_Open	Open the motion controller
GT_Close	Close the motion controller
GT_Reset	Reset the motion controller
GT_HardRst	Reset the motion controller with hardware
GT_HookCommand	Hook command handler function defined by user
Set Control Axis Group	
GT_AlarmOn	Enable monitoring drive alarm signal of specified axis
GT_AlarmOff	Disable monitoring drive alarm signal of specified axis
GT_LmtsOn	Enable the limit switch
GT_LmtsOff	Disable effective the limit switch
GT_LmtSns	Set the effective level of the limit switch
GT_EncSns	Set counting direction of all axes encoder
GT_HomeSns	Set trigger edge of HOME signal
GT_IndexSns	Set trigger edge of INDEX signal
GT_CloseLp	Set the specified axis in closed loop control
GT_OpenLp	Set the specified axis in open loop control
GT_CtrlMode	Set the output mode of specified axis as analog voltage output or pulse output
GT_StepDir	Set the pulse output mode as “Pulse + Direction”
GT_StepPulse	Set the pulse output mode as “Positive and Negative Pulse”
GT_AuStpOn	Enable automatic stopping independent axis motion when motion error exceeds the error limit
GT_AuStpOff	Disable automatic stopping independent axis motion when motion error exceeds the error limit
GT_DrvRst	Reset the motor driver
GT_EncOn	Enable the encoder input
GT_EncOff	Disable the encoder input
Set axis motion mode group	
GT_PrflT	Set the independent axis motion mode of specified axis as T-curve independent positioning mode
GT_PrflS	Set the independent axis motion mode of specified axis as S-curve independent positioning mode
GT_PrflV	Set the independent axis motion mode of specified axis as independent jogging mode

Function	Meaning
Set axis motion parameter group	
GT_SetPos	Set the target position of the specified axis
GT_GetPos	Get the target position of the specified axis
GT_SetVel	Set the target velocity of the specified axis
GT_GetVel	Get the target velocity of the specified axis
GT_SetAcc	Set the acceleration of the specified axis
GT_GetAcc	Get the acceleration of the specified axis
GT_SetMAcc	Set the maximum acceleration of the specified axis
GT_GetMAcc	Get the maximum acceleration of the specified axis
GT_SetJerk	Set the jerk of the specified axis
GT_GetJerk	Get the jerk of the specified axis
GT_ZeroPos	Reset the actual and target position to zero of the specified axis
GT_SetAtlPos	Set the actual position of the specified axis
GT_GetAtlPos	Get the actual position of the specified axis
GT_GetPrfPos	Get the profile position of the specified axis
GT_GetAtlErr	Get the actual position error of the specified axis
GT_GetAtlVel	Get the actual velocity of the specified axis
GT_GetPrfVel	Get the planned velocity of the specified axis
Set axis control parameter group	
GT_SetKp	Set the percentage gain of the specified axis
GT_GetKp	Get the percentage gain of the specified axis
GT_SetKi	Set the integral gain of the specified axis
GT_GetKi	Get the integral gain of the specified axis
GT_SetKd	Set the differential gain of the specified axis
GT_GetKd	Get the differential gain of the specified axis
GT_SetKvff	Set the velocity feed forward of the specified axis
GT_GetKvff	Get the velocity feed forward of the specified axis
GT_SetKaff	Set the acceleration feed forward of the specified axis
GT_GetKaff	Get the acceleration feed forward of the specified axis
GT_SetMtrBias	Set the bias compensation of the specified axis
GT_GetMtrBias	Get the bias compensation of the specified axis
GT_SetMtrLmt	Set voltage limit of the specified axis
GT_GetMtrLmt	Get voltage limit of the specified axis
GT_SetILmt	Set the error integral limit of the specified axis
GT_GetILmt	Get the error integral limit of the specified axis
GT_GetIntgr	Get the integral error of the specified axis
GT_SetPosErr	Set the position error limit of the specified axis
GT_GetPosErr	Get the position error limit of the specified axis
Axis motion control group	
GT_AxisOn	Activate the specified axis and enable the driver

Function	Meaning
GT_AxisOff	Inactivates the specified axis and disable the driver
GT_Update	Update parameters of the specified axis
GT_MltiUpdt	Update parameters of specified several axes
GT_SmthStp	Stop the motion smoothly
GT_AbptStp	Stop the motion abruptly
Watch status group	
GT_GetSts	Get the status word of the specified axis
GT_GetCmdSts	Get the contents of the command status register of the controller
GT_ClrSts	Clear the status of the specified axis
GT_RstSts	Clear the status register of the specified axis according to the definition of mask
GT_GetMode	Get the control mode character of the specified axis
Home/Index capture group	
GT_CaptHome	Set HOME signal capture
GT_CaptIndex	Set INDEX signal capture
GT_GetCapt	Get the INDEX or HOME captured position
Digital I/O group	
GT_ExInpt	Get the status of general purposed input of the motion controller
GT_ExOpt	Set the status of general purposed output of the motion controller
GT_GetHomeSwt	Get the status of home switch
GT_GetLmtSwt	Get the status of limit switch
D/A group	
GT_SetMtrCmd	Set the motor control value in open loop mode
GT_GetMtrCmd	Get the motor control value in open loop mode
Multi card operation group	
GT_GetCurrentCardNo	Get the number of the current card
GT_SwitchtoCardNo	Set the specified card as the current card
A/D(optional)	
GT_GetAdc	Get the AD conversion result of the specified channel
Data watch(optional)	
GT_SetWatch	Set watch parameter
GT_GetWatch	Get watch parameter
GT_StartWatch	Start watch
GT_StopWatch	Stop watch

表 11-2　GE-X00-SX Commands

Function	Meaning
Set Motion Controller Group	
GT_Open	Open the motion controller
GT_Close	Close the motion controller
GT_Reset	Reset the motion controller
GT_HardRst	Reset the motion controller with hardware
GT_HookCommand	Hook command handler function defined by user
Set Control Axis Group	
GT_AlarmOn	Enable monitoring drive alarm signal of specified axis
GT_AlarmOff	Disable monitoring drive alarm signal of specified axis
GT_LmtsOn	Enable the limit switch
GT_LmtsOff	Disable effective the limit switch
GT_LmtSns	Set the effective level of the limit switch
GT_EncSns	Set counting direction of all axes encoder
GT_HomeSns	Set trigger edge of HOME signal
GT_CtrlMode	Set the output mode of specified axis as analog voltage output or pulse output
GT_StepDir	Set the pulse output mode as “Pulse + Direction”
GT_StepPulse	Set the pulse output mode as “Positive and Negative Pulse”
GT_DrvRst	Reset the motor driver
GT_EncOn	Enable the encoder input
GT_EncOff	Disable the encoder input
Set axis motion parameter group	
GT_ZeroPos	Reset the actual and target position to zero of the specified axis
GT_SetAtlPos	Set the actual position of the specified axis
GT_GetAtlPos	Get the actual position of the specified axis
GT_GetPrfPnt	Get the planning position of all axes in interpolation motion
GT_GetSegPnt	Get the end point position of the specified working segment
GT_GetBrkPnt	Get the break point position of all axes after stopping multi-segment continuous path motion
GT_GetPrfVel	Get the planned path velocity of the interpolation motion
Set axis control parameter group	
GT_SetKp	Set the percentage gain of the specified axis
GT_GetKp	Get the percentage gain of the specified axis
GT_SetKi	Set the integral gain of the specified axis
GT_GetKi	Get the integral gain of the specified axis
GT_SetKd	Set the differential gain of the specified axis

Function	Meaning
GT_GetKd	Get the differential gain of the specified axis
GT_SetKvff	Set the velocity feed forward of the specified axis
GT_GetKvff	Get the velocity feed forward of the specified axis
GT_SetKaff	Set the acceleration feed forward of the specified axis
GT_GetKaff	Get the acceleration feed forward of the specified axis
GT_SetMtrBias	Set the bias compensation of the specified axis
GT_GetMtrBias	Get the bias compensation of the specified axis
GT_SetMtrLmt	Set voltage limit of the specified axis
GT_GetMtrLmt	Get voltage limit of the specified axis
GT_SetILmt	Set the error integral limit of the specified axis
GT_GetILmt	Get the error integral limit of the specified axis
GT_SetPosErr	Set the position error limit of the specified axis
GT_GetPosErr	Get the position error limit of the specified axis
Axis motion control group	
GT_AxisOn	Activate the specified axis and enable the driver
GT_AxisOff	Inactivates the specified axis and disable the driver
GT_Update	Update parameters of the specified axis
GT_MltiUpdt	Update parameters of specified several axes
Path motion group	
GT_LnXY	2D linear interpolation motion
GT_LnXYG0	2D linear interpolation motion with symmetry profile of velocity planning
GT_LnXYZ	3D linear interpolation motion
GT_LnXYZG0	3D linear interpolation motion with symmetry profile of velocity planning
GT_ArcXY	circular interpolation motion in XOY plane, described by arc center position and angle
GT_ArcXYP	circular interpolation motion in XOY plane, described by radius and end position
GT_ArcYZ	circular interpolation motion in YOZ plane, described by arc center position and angle
GT_ArcYZP	circular interpolation motion in YOZ plane, described by radius and end position
GT_ArcZX	circular interpolation motion in ZOX plane, described by arc center position and angle
GT_ArcZXP	circular interpolation motion in ZOX plane, described by radius and end position
GT_SetSynVel	Set path velocity (feed speed)
GT_SetSynAcc	Set path acceleration
GT_SetStrtVel	Set start velocity
GT_SetStpAcc	Set stop acceleration

Function	Meaning
GT_SetMaxVel	Set maximum velocity
GT_Override	Specify the feed override of the motion path description command except command GT_LnXYG0, GT_LnXYZG0, GT_MvXY, GT_MvXYZ
GT_OverrideG0	Specify the feed override of GT_LnXYG0, GT_LnXYZG0, GT_MvXY or GT_MvXYZ
GT_StpMtn	Stop the interpolation motion with specified acceleration
GT_EStpMtn	Stop the interpolation motion with stop acceleration
GT_MapCnt	Specify the coordinate offset of the specified axis
Buffer group	
GT_StrtList	Open and clear the buffer
GT_MvXY	Specify start point position (2D), path velocity and acceleration in buffer.
GT_MvXYZ	Specify start point position (3D), path velocity and acceleration in buffer.
GT_EndList	Close opened buffer
GT_AddList	reopen closed buffer
GT_StrtMtn	Start multi-segment continuous path motion
GT_RestoreMtn	Restore multi-segment continuous path motion from the breakpoint after stop the motion
GT_GetMtnNm	Get the working path segment number
Watch status group	
GT_GetSts	Get the status word of the specified axis
GT_GetCmdSts	Get the contents of the command status register of the controller
GT_GetCrdSts	Get the motion status of interpolation motion
GT_ClrSts	Clear the status of the specified axis
GT_RstSts	Clear the status register of the specified axis according to the definition of mask
Home/Index capture group	
GT_CaptHome	Set HOME signal capture
GT_CaptIndex	Set INDEX signal capture
GT_GetCapt	Get the INDEX or HOME captured position
Digital I/O group	
GT_ExInpt	Get the status of general purposed input of the motion controller
GT_ExOpt	Set the status of general purposed output of the motion controller
GT_GetHomeSwt	Get the status of home switch
GT_GetLmtSwt	Get the status of limit switch
Velocity control	

Function	Meaning
GT_SetDccVel	Specify segment end velocity
Look ahead group	
GT_InitLookAhead	Initialize the technological parameters of machine tool
GT_AddLookData	Push path parameters to Look-Ahead block
GT_CalVel	Calculate the end velocity of the current segment in the Look-Ahead block
Auxilary encoder	
GT_EncPos	Get the actual position of auxiliary encoder
Multi card operation group	
GT_GetCurrentCard No	Get the number of the current card
GT_SwitchtoCardNo	Set the specified card as the current card
Hand wheel (optional)	
GT_HandWheel	Set specified axis into hand wheel control mode
GT_ExitHWheel	Let the axis which is in hand wheel control mode exit from hand wheel control mode
Spindle control (optional)	
GT_CloseSpindle	Close main spindle output
GT_SetSpindleVel	Specify speed of main spindle rotating and make spindle to rotate as specified speed
A/D(optional)	
GT_GetAdc	Get the AD conversion result of the specified channel
Data watch(optional)	
GT_SetWatch	Set watch parameter
GT_GetWatch	Get watch parameter
GT_StartWatch	Start watch
GT_StopWatch	Stop watch

Chapter 12 Description of Commands

 **Notice**	*Each command will be described as follows one by one in the alphabetic order, the prototypes of the commands are all written in C language environment.* *If you are learning to use the motion controller or at the stage of debugging the program, please don't connect it with mechanical system to avoid spoiling machine by mishandling.* *When you connect mechanical system at the first time, please check carefully if the status of limit switch and counting direction of encoder are correct, parameters of PID filter and motion planning are reasonable. In one word, please make sure the mechanical system is normal and safe.*

GT_AbptStp

Command Prototype: short GT_AbptStp (unsigned short axis)

Description of Command: This command is used to stop the axis motion of the specified axis abruptly, and set the parameter of target velocity and actual velocity as zero. After being executed, this command will become effective for abrupt stop. The description in details refers to Chapter 5.

Command Parameter: the number of the specified axis.

Applicable Card: GE-X00-PX.

Command Evoking: In the following example, when detecting that EXI15 port of external IO is at high level, stop the motion of the first axis abruptly.

```
void main()
{
    unsigned short exInpt;
    GT_HookCommand(CommandHandle);          //refer to example 5-1
    GT_Open();                              //open the motion controller
    GT_Reset();                             //reset the motion controller
    AxisInitial(1,0,0);                     //refer to example 3-2
    AxisRunT(1,1000000,1,0.1);              //refer to example 5-1
    do
    {
        GT_ExInpt(&exInpt);                 //read the status of the general
                                              digital
```

```
                                                        //input port
            if((exInpt&0x8000)==0x8000)
            {
                GT_AbptStp(1);                          //stop the first axis abruptly
            }
        }while((exInpt&0x8000)!=0x8000);
}
```

GT_AddList

Command Prototype: short GT_AddList(void)

Description of Command: **U**ser can call this command to reopen closed buffer, then the motion path and parameters commands can be pushed into buffer again. When the buffer is opened, this command is invalid.

Applicable Card: GE-X00-SX.

Command Evoking: In the following example, after using GT_EndList to close the buffer, open the buffer again.

```
short   AddList ()
{
    short rtn;
    rtn=GT_StrtList();                          //open the command buffer
    if(0!=rtn)   return   rtn;
    rtn=GT_MvXYZ(0,0,0,0,5,0.1);                //specify interpolation motion start
    point
    if(0!=rtn)   return   rtn;
    …                                           //continue to append commands
    rtn=GT_EndList();                           //close the buffer
    if(0!=rtn)   return   rtn;
    rtn=GT_AddList();                           //open the buffer again
    if(0!=rtn)   return   rtn;
    rtn=GT_LnXY(20000,30000);                   //add the path command to the
                                                //buffer
    if(0!=rtn)   return   rtn;
    …                                           //continue to append commands
```

GT_AddLookData

Command Prototype: short GT_AddLookData(char code, char plane_group, double r, double x, double y, double z, double vel, double cx, double cy, int i, long n, bool flag)

Description of Command：Push path parameters to Look-Ahead block. If the return value of this command is “0”, means Look-Ahead block receives path

parameters successfully; while it is “1”, the path parameters is same as the last one, you should push the following.

Command Parameter：

Parameter name	Parameter meaning	Explanation	
code	code of path curve	0	G00 line
		1	G01 line
		2	clockwise arc
		3	counter-clockwise arc
plane_group	code of plane	17	XY plane
		18	ZX plane
		19	YZ plane
r	arc radius	If the segment is not arc, R equal to be 0	
x，y，z	Segment end point position	unit：mm	
vel	feed speed	unit：mm/s	
cx，cy	Position of the arc center	If the segment is not arc, Specify **cx** and **cy** to be 0	
i	Specific memory location in Look-Ahead block		
n	segment mark	The segment number in code file from user	
flag	deceleration flag	In general situation, it is false. While if there is some technological request to stop the machine (such as tool change), it should be set as true.	

Applicable Card: GE-X00-SX.

GT_AlarmOff

Command Prototype：short GT_AlarmOff(unsigned short axis)
Description of Command：Disable monitoring drive alarm signal of specified axis.
Parameter of Command：Specified axis number.
Applicable Card：GE-X00-PX，GE-X00-SX.

GT_AlarmOn

Command Prototype：short GT_AlarmOn(unsigned short axis)
Description of Command：Enable monitoring drive alarm signal of specified axis.
Parameter of Command：Specified axis number.
Applicable Card：GE-X00-PX，GE-X00-SX。
Command Evoking：
short AlarmConfig(unsigned short axis,short enable)

```
{
    if(enable)
    {
        return GT_AlarmOn(axis);                    //enable monitoring drive alarm
        signal
    }
    else
    {
          return GT_AlarmOff(axis);                 //disable monitoring drive alarm
                                                        signal
    }
}
```

GT_ArcXY

Command Prototype: short GT_ArcXY(double x_center, double y_center, double angle)

Description of Command: This is circular interpolation command in XOY plane, described by arc center position and angle. The start positions are the end position of the last segment (in buffer motion mode) or current position (in immediate motion mode). If it is the first motion path segment in buffer, the start position is the position specified by GT_MvXY or GT_MvXYZ.

Command Parameter: **"**x_center" and "y_center" are the position of arc center, unit: pulse. "angle" is the rotating angle with a unit of degree. The positive or negative value represents the rotating direction. The value of rotating angle ranges from -360 to 360 degree. For the rotating direction, please refer to the description in Fig. 6-1.

Applicable Card: GE-X00-SX.

Command Evoking: The following example executes the command of circular interpolation in XOY plane immediately.

```
short   ArcXY ()
{
    short rtn;
    rtn=GT_SetSynVel(5);                            //set path velocity
    if(0!=rtn)   return   rtn;
    rtn=GT_SetSynAcc(0.1);                          //set acceleration
    if(0!=rtn)   return   rtn;
    rtn=GT_ArcXY(40000,30000,180);                  //start   circular   interpolation
    motion in
                                                        //XOY plane
    if(0!=rtn)   return   rtn;
    return 0;
}
```

GT_ArcXYP

Command Prototype: short GT_ArcXYP(double x_end, double x_end, double r, short dir)

Description of Command: This is circular interpolation command in XOY plane, described by radius and end position. The start positions are the end position of the last segment (in buffer motion mode) or current position (in immediate motion mode). If it is the first motion path segment in buffer, the start position is the position specified by GT_MvXY or GT_MvXYZ.

Command Parameter: “x_end” and “y_end” are the end position of arc. “r” is the arc radius with a symbol, which represents whether the arc is major or minor (positive: minor arc; negative: major arc). The units of position and radius are pulse. “dir” is the rotating direction of arc. “1” means the positive direction, “-1” means the negative direction. For the rotating direction, please refer to the description in Fig. 6-1.

Applicable Card: GE-X00-SX

Command Evoking: The following example pushes the command of circular interpolation in XOY plane into the buffer.

```
short    ArcXYP ()
{
        short rtn;
        rtn=GT_StrtList();
        if(0!=rtn)    return    rtn;
        rtn=GT_MvXY(0,0, 5,0.1);
        if(0!=rtn)    return    rtn;
        rtn=GT_ArcXYP(40000,0,20000,-1);
        if(0!=rtn)    return    rtn;
        rtn=GT_EndList();
        if(0!=rtn)    return    rtn;
        return 0;
}
```

GT_ArcYZ

Command Prototype: short GT_ArcYZ(double y_center, double z_center, double angle)

Description of Command: This is circular interpolation command in YOZ plane, described by radius and end position. The start positions are the end position of the last segment (in buffer motion mode) or current position (in immediate motion mode). If it is the first motion path segment in buffer, the start position is the position specified by GT_MvXYZ.

Command Parameter: **“**y_center” and “z_center” are the position of arc center, unit: pulse. “angle” is the rotating angle with a unit of degree. The positive or negative value represents the rotating direction. The value of rotating angle ranges from

-360 to 360 degree. For the rotating direction, please refer to the description in Fig. 6-1.

Applicable Card: GE-X00-SX.

Command Evoking: The following example executes the command of circular interpolation in YOZ plane immediately.

```
short    ArcYZ ()
{
        short rtn;
        rtn=GT_SetSynVel(5);
        if(0!=rtn)    return    rtn;
        rtn=GT_SetSynAcc(0.1);
        if(0!=rtn)    return    rtn;
        rtn=GT_ArcYZ(40000,30000,180);
        if(0!=rtn)    return    rtn;
        reurn 0;
}
```

GT_ArcYZP

Command Prototype: short GT_ArcYZP(double y_end, double y_end, double y, short dir);

Description of Command: This is circular interpolation command in YOZ plane, described by radius and end position. The start positions are the end position of the last segment (in buffer motion mode) or current position (in immediate motion mode). If it is the first motion path segment in buffer, the start position is the position specified by GT_MvXYZ.

Command Parameter: “y_end” and “z_end” are the end position of arc. “r” is the arc radius with a symbol, which represents whether the arc is major or minor (positive: minor arc; negative: major arc). The units of position and radius are pulse. “dir” is the rotating direction of arc. “1” means the positive direction, “-1” means the negative direction. For the rotating direction, please refer to the description in Fig. 6-1.

Applicable Card: GE-X00-SX

Command Evoking: The following example pushes the command of circular interpolation in YOZ plane into the buffer.

```
short    ArcYZP ()
{
        short rtn;
        rtn=GT_StrtList();
        if(0!=rtn)    return    rtn;
        rtn=GT_MvXYZ(0,0,0,5,0.1);
        if(0!=rtn)    return    rtn;
        rtn=GT_ArcYZP(40000,0,20000,-1);
```

```
    if(0!=rtn)  return  rtn;
    rtn=GT_EndList();
    if(0!=rtn)  return  rtn;
    return 0;
}
```

GT_ArcZX

Command Prototype: short GT_ArcZX(double z_center, double x_center, double angle)

Description of Command: This is circular interpolation command in ZOX plane, described by radius and end position. The start positions are the end position of the last segment (in buffer motion mode) or current position (in immediate motion mode). If it is the first motion path segment in buffer, the start position is the position specified by GT_MvXYZ.

Command Parameter: **"**z_center" and "x_center" are the position of arc center, unit: pulse. "angle" is the rotating angle with a unit of degree. The positive or negative value represents the rotating direction. The value of rotating angle ranges from -360 to 360 degree. For the rotating direction, please refer to the description in Fig. 6-1.

Applicable Card: GE-X00-SX.

Command Evoking: The following example executes the command of circular interpolation in YOZ plane immediately.

```
short ArcZX ()
{
    short rtn;
    rtn=GT_SetSynVel(5);
    if(0!=rtn)  return  rtn;
    rtn=GT_SetSynAcc(0.1);
    if(0!=rtn)  return  rtn;
    rtn=GT_ArcZX(40000,30000,180);
    if(0!=rtn)  return  rtn;
    return 0;
}
```

GT_ArcZXP

Command Prototype: short GT_ArcZXP(double z_end, double x_end, double r, short dir)

Description of Command: This is circular interpolation command in ZOX plane, described by radius and end position. The start positions are the end position of the last segment (in buffer motion mode) or current position (in immediate motion mode). If it is the first motion path segment in buffer, the start position is the position specified by GT_MvXYZ.

Command Parameter: “z_end” and “x_end” are the end position of arc. “r” is the arc radius with a symbol, which represents whether the arc is major or minor (positive: minor arc; negative: major arc). The units of position and radius are pulse. “dir” is the rotating direction of arc. “1” means the positive direction, “-1” means the negative direction. For the rotating direction, please refer to the description in Fig. 6-1.

Applicable Card: GE-X00-SX

Command Evoking: The following example pushes the command of circular interpolation in ZOX plane into the buffer.

```
short ArcZxp()
{
    short rtn;
    rtn=GT_StrtList();
    if(0!=rtn)  return  rtn;
    rtn=GT_MvXYZ(0,0,0,5,0.1);
    if(0!=rtn)  return  rtn;
    rtn=GT_ArcZXP(40000,0,20000,-1);
    if(0!=rtn)  return  rtn;
    rtn=GT_EndList();
    if(0!=rtn)  return  rtn;
    return 0;
}
```

GT_AuStpOff

Command Prototype: short GT_AuStpOff(unsigned short axis)

Description of Command： This command disables the function of automatic stop independent axis motion when the positional error of specified axis exceeds the error limit. After carrying out this command, if the error between the actual position and planning position exceed the error limit, the motion controller will not stop the independent motion of the specified axis.

Parameter of Command: Specified axis number.

Applicable Card: GE-X00-PV

GT_AuStpOn

Command Prototype：short GT_AuStpOn(unsigned short axis)

Description of Command： This command enables the function of automatic stop independent motion when the positional error of specified axis exceeds the error limit. After carrying out this command, if the error between the actual position and planning position exceed the error limit, the motion controller will stop the independent motion of the specified axis and set corresponding bit of the axis

status as 1. default value of the error limit is 32767，you may modify it by calling command GT_SetPosErr.

Parameter of Command: Specified axis number.

Applicable Card: GE-X00-PV.

Command Evoking:

```
short AutoStopConfig (unsigned short axis,short enable)
{
    if(enable)
    {
        return GT_AuStpOn (axis);        //enable automatic stop independent motion when the //positional error of specified axis exceeds the error //limit
    }
    else
    {
        return GT_AuStpOff (axis);       //disable the function of automatic stop independent motion

    }
}
```

GT_AxisOff

Command Prototype: short GT_AxisOff(unsigned short axis)

Description of Command: This command inactivates the specified axis and disables the driver.

Applicable Card: GE-X00-PX，GE-X00-SX

Relevant Command: GT_AxisOn

Command Evoking: refer to GT_AxisOn.

GT_AxisOn

Command Prototype: short GT_AxisOn(unsigned short axis)

Description of Command: This command activates the specified axis and enables the driver. If you hope that the system work in closed loop control, you should select servo motor, and let its driver receive analog voltage. In such situation, before call GT_AxisOn(), user should specify and validate the parameters of PID filter in motion controller. For safety of mechanical system, the motion controller will synchronize target position and planning position as actual position when this command is called.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: In the following example, when detecting that EXI15 port of external

IO is at high level, call GT_AxisOn for axis 1, when detecting that EXI15 port of external IO is at low level, call GT_AxisOff for axis 1.

```
short AxisEnable(unsigned short axis)
{
    unsigned short exInpt;
    rtn = GT_ExInpt(&exInpt);                    //read the status of general digital
    input
    if(0!=rtn)  return  rtn;
    if(exInpt&0x8000)
    {
        return GT_AxisOn(axis);                  //if EI15 is high level, enable the
        servo
    }
    else
    {
        return GT_AxisOff(axis);                 //if EI15 is low level, disable the
        servo
    }
}
```

GT_Calvel

Command Prototype: short GT_CalVel(double *vel, long *number)

Description of Command: Calculate the end velocity of the current segment in the Look Ahead block. If the return value is 0, Look Ahead function should be going on. If the return value is 1, Look Ahead function is finished.

Command Parameter: “vel” gets the end velocity, “number” is corresponding with the segment number in code file from user, that is parameter “n” in command GT_AddLookData.

Applicable Card: GE-X00-SX.

GT_CaptHome

Command Prototype: short GT_CaptHome(unsigned short axis)

Description of Command: This command sets the position capture event as home signal capture. After evoking this command, the position capture register will record the actual position exactly at that time home signal is triggered. GT_CaptHome just captures actual position one time. To capture another one, user must clear the corresponding bit (bit3) of axis status by calling the command of GT_ClrSts or GT_RstSts, and call GT_CaptHome again to set the position capture event as home signal capture. The process of capturing is done by hardware, besides the

motion velocity.

Applicable Card: GE-X00-PX，GE-X00-SX.

Command Evoking: refer to example 7-1.

GT_CaptIndex

Command Prototype: short GT_CaptIndex(unsigned short axis)

Description of Command: This command sets the position capture event as index signal capture. After evoking this command, the position capture register will record the actual position exactly at the time Index signal is triggered. GT_CaptIndexjust captures actual position one time. To capture another one, user must clear the corresponding bit (bit3) of axis status by calling the command of GT_ClrSts or GT_RstSts, and call GT_CaptIndex again to set the position capture event as index signal capture. The actual position captured by index signal can be used as the accurate zero return. The process of capturing is done by hardware, besides the motion velocity.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: refer to example 7-2

GT_Close

Command Prototype: short GT_Close(void)；

Description of Command: Close the motion controller. Generally this command is called at the end of user program.

Applicable Card: GE-X00-PX，GE-X00-SX

Relevant Command: GT_Open

GT_CloseLp

Command Prototype: short GT_CloseLp(unsigned short axis)

Description of Command: This command sets the specified axis in closed loop control. The default status of each axis is closed loop control. For safety of the system, please inactivate specified axis at first, call this command to set specified axis in closed loop control, then activate this axis again.

Command Parameter: Specified axis number

Applicable Card: GE-X00-PV

GT_CloseSpindle

Command Prototype: short GT_CloseSpindle(void)

Description of Command: Close main spindle output

Applicable Card：GE-X00-SV

Command Evoking: refer to example 10-6

GT_ClrSts

Command Prototype: short GT_ClrSts(unsigned short axis)

Description of Command: This command clears bit flag from bit0 to bit7 in status register of specified axis, the other bits in the specified axis status will not be affected. As to the definition of status bit, please refer to chapter 4.

Command Parameter：Specified axis number

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: refer to example 3-2, 3-3, 3-4.

GT_CtrlMode

Command Prototype: short GT_CtrlMode(unsigned short axis, unsigned short mode)

Description of Command: This command sets the output mode of specified axis as analog voltage output or pulse output.

If servo motor of specified axis is working in velocity loop mode, the output mode of specified axis should be analog, if servo motor of specified axis is working in position loop mode, or the specified axis connects with stepper motor, the output mode of specified axis should be pulse.

For GE-X00-XV, default mode of all axes is analog output. If the work mode of motor driver does not match the output mode of GE-X00-XV, the motor can't work.

Command Parameter: "axis" means specified axis number, "mode" means output mode, 0 means analog voltage output mode and 1 means pulse output mode.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking: In the following example, when detecting that EXI15 port of external IO is at high level, set the first axis as pulse output mode, when detecting that EXI15 port of external IO is at low level, set the first axis as analog output mode.

```
short ControlModeConfig(unsigned short axis)
{
    unsigned short exInpt;
    rtn = GT_ExInpt(&exInpt);                    //read the status of general digital inout
    if(0!=rtn)  return  rtn;
    if(exInpt&0x8000)
    {
        return GT_CtrlMode(axis,1);              //if EI15 is high level, setting pulse
        output mode
    }
    else
    {
        return GT_CtrlMode(axis,0);              //if EI15 is low level, setting voltage
```

```
            output mode
        }
    }
```

GT_DrvRst

Command Prototype: short GT_DrvRst(unsigned short axis)

Description of Command: When the driver of specified axis sends alarm signal to the motion controller, after evoking this command, the controller will send resetting signal to the axis driver to reset the driver. Before evoking this command, the specified axis has to be in drive prohibition status. Otherwise, the command is ineffective.

Command Parameter: specified axis.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: In the following example, first disable servo of the specified axis, then reset the driver.

```
short ClearAlarmSignal(unsigned short axis)
{
    short rtn;
    rtn = GT_AxisOff(axis);                    //disable servo of the specified axis
    if(0!=rtn)    return rtn;
    delay(10000);                    //delay 10 second
    rtn = GT_DrvRst(axis);      //reset the alarm of the specified axis
    return rtn;
}
```

GT_EncOff

Command Prototype：short GT_EncOff(unsigned short axis)

Description of Command：Close encoder input of specified axis, if specified axis is activated, calling this command is invalid.

Command Parameter：Specified axis number

Applicable Card：GE-X00-PX, GE-X00-SX

GT_EncOn

Command Prototype：short GT_EncOn(unsigned short axis)

Description of Command：Open encoder input of specified axis, if specified axis is activated, calling this command is invalid.

Command Parameter：Specified axis number

Applicable Card：GE-X00-PX, GE-X00-SX

GT_EncPos

Command Prototype: GT_EncPos(unsigned short axis, long *pos)

Description of Command: This command is to get the actual position of auxiliary encoder. GE-X00-SX has one auxiliary encoder.

Command Parameter: “axis” means the number of auxiliary encoder, “*pos” gets the actual position of specified auxiliary encoder.

Applicable Card: GE-X00-SX

Command Evoking: The following example gets and prints the position of auxiliary encoder.

```
short    ReadEncPos()
{
        short rtn;
        long pos
        rtn=GT_EncPos(1,&pos);
        if(0!=rtn) return rtn;
        printf(“pos:%ld”,pos);
}
```

GT_EncSns

Command Prototype: short GT_EncSns(unsigned short sense)

Description of Command: This command defines counting direction of all encoders. If and only if the positive direction of motor rotating agrees with the positive direction of encoder counting, the whole system can work normally. If the wrong connection or other reasons causes the two directions opposite to each other, user can evoke this command to change the positive direction of encoder counting.

Command Parameter：Specify positive direction of all encoder counting

Applicable Card: GE-X00-PX, GE-X00-SX

Command Evoking: refer to example 3-2

GT_EndList

Command Prototype: short GT_EndList(void);

Description of Command: When buffer is opened, user can call this command to close the buffer, then the motion path and parameters commands can’t be pushed into the buffer.

Applicable Card: GE-X00-SX

Command Evoking:

```
short    EndList()
{
        short rtn;
```

```
    rtn=GT_StrtList();                                    //Open and clear
    buffer
    if(0!=rtn) return rtn;
    rtn=GT_MvXYZ(0,0,0,16,3.7); //Specify the start point position, path velocity and
    acceleration
    if(0!=rtn) return rtn;
    rtn=GT_LnXYZ(1234,5678,9013);          //push 3D linear interpolation command
    into buffer
    if(0!=rtn) return rtn;
    rtn=GT_ArcXY(2345,6789,360); //push XOY plane circular interpolation command
    into buffer
    if(0!=rtn) return rtn;
    rtn=GT_EndList();
                                                                          //clos
    e buffer
    if(0!=rtn) return rtn;
}
```

GT_EStpMtn

Command Prototype: short GT_EStpMtn(void);

Description of Command: This command stops the interpolation motion with stop acceleration. During multi-segment continuous path motion, if this command is called successfully, the buffer will be closed immediately. Just like calling the command GT_EndList. After motion stops, GE-X00-SX will record break point information. So that once user restores motion, GE-X00-SX should return to break point precisely to continue the following path motion described in buffer. While during single-segment path motion, after motion stops, the information of current motion will be abandoned.

If user stops motion during executing GT_MvXY or GT_MvXYZ, maybe the multi-segment continuous path motion can't be restored correctly.

Applicable Card: GE-X00-SX

GT_ExInpt

Command Prototype: short GT_ExInpt(unsigned short *input)

Description of Command: This command gets the status of general digital input of the motion controller.

Command Parameter: "*input" returns the status. The corresponding bit to general digital input port is described in Chapter 9.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: See the GT_ExOpt

GT_ExitHWheel

Command Prototype: short GT_ExitHWheel(void)

Description of Command: This command lets the axis that is in hand wheel control mode exit from hand wheel control mode.

Applicable Card: GE-X00-SX

GT_ExOpt

Command Prototype: short GT_ExOpt(unsigned short output)

Description of Command: This command sets the status of general digital output of the motion controller.

Command Parameter: “output” is the status to be set. The corresponding bit to general digital output port is described in Chapter 9.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: The following example sets the general output EXO2 at high level if the general digital input EXI1 is at high level.

```
short DigitalInputOutput()
{
    short rtn;
    unsigned short exInpt;
    rtn = GT_ExInpt(&exInpt);                    //read the extended input
    if(0!=rtn) return rtn;
    if(exInpt&0x2)
    {
        return GT_ExOpt(0x4);                    //set the EXO2 is high level
    }
}
```

GT_GetAcc

Command Prototype: short GT_GetAcc(unsigned short axis, double *acc)

Description of Command: This command gets the acceleration of specified axis set by the command GT_SetAcc.

Command Parameter: “axis” is the specified axis number, “*acc” returns the acceleration.

Applicable Card: GE-X00-PX

Command Evoking: refer to the description of the command GT_GetPos.

GT_GetAdc

Command Prototype: short GT_GetAdc(unsigned short channel, short *voltage)

Description of Command: The motion controller provides 8 A/D channels with 12 bits

conversion precision. This command gets the A/D conversion result of the specified channel. The relation between the voltage and the gotten value is described in chapter 10.

Command Parameter: Channel is the number of the specified channel; *voltage is the gotten voltage.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: refer to example 10-4.

GT_GetAtlErr

Command Prototype: short GT_GetAtlErr(unsigned short axis,long *error)

Description of Command: This command gets the actual position error of specified axis, i.e. the difference between the current planned position and the current actual position. This command is usually used to monitor the servo control position error.

Command Parameter: “axis” is the specified axis number, “*error” returns the actual position error.

Applicable Card: GE-X00-PV

Command Evoking: The following example gets and prints the actual position error of the first axis.

```
void main()
{
    unsigned long status;
    long error;
    GT_HookCommand(CommandHandle);
    AxisInitial(1,0,0,1);                          //refer to example 3-4
    AxisRunT(1,10000,10,1);                        //refer to example 5-1
    do
    {
        GT_GetSts(1,&status);
    }while(status&0x400);                  //waiting for the end of the specified
    axis motion
    GT_GetAtlErr(1,&error);                        //read actual error of the
    specified axis
    printf(“\nError = %ld”,error);
}
```

GT_GetAtlPos

Command Prototype: short GT_GetAtlPos(unsigned short axis, long *pos)

Description of Command: This command gets the actual position of specified axis.

Command Parameter: “axis” is the specified axis number, “*pos” returns the actual position.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: The following example gets and prints the actual position of the first axis.

```
void main()
{
    unsigned long status;
    long atlPos, prfPos;
    GT_HookCommand(CommandHandle);
    AxisInitial(1,0,0,1);                          //reference to example 3-4
    AxisRunT(1,10000,10,1);                        //reference to example 5-1
    do
    {
        GT_GetSts(1, &status);
    }while(status&0x400);                  //waiting for the end of the specified axis motion
    GT_GetAtlPos(1, &atlPos) ;                     //read actual position of the specified axis
    GT_GetPrfPos(1, &prfPos);                      //read planned position of the specified axis
    printf("\nActual Position = %ld Profile Position = %ld", atlPos, prfPos);
}
```

GT_GetAtlVel

Command Prototype: short GT_GetAtlVel(unsigned short axis, double *vel)

Description of Command: Read actual velocity of specified axis

Command Parameter: "axis" is the specified axis number, "*vel" returns actual velocity of specified axis.

Applicable Card: GE-X00-PX

Command Evoking: When the planning of the first axis motion is over, read the actual velocity and planned velocity of the first axis.

```
void main()
{
    unsigned long status;
    double atlVel, prfVel;
    GT_HookCommand(CommandHandle);
    AxisInitial(1,0,0,1);                          // reference to example 3-4
    AxisRunT(1,10000,10,1);                        // reference to example 5-1
    do
    {
        GT_GetSts(1, &status);
    }while(status&0x400);                  //waiting for the end of the specified
```

```
        axis motion
        GT_GetAtlVel(1, &atlVel);                    //read actual velocity of the
        specified axis
        GT_GetPrfVel(1, &prfVel);                    //read planned velocity of the
        specified axis
        printf("\nActual Vel = %lf Profile Vel = %lf", atlVel, prfVel);
}
```

GT_GetBrkPnt

Command Prototype: short GT_GetBrkPnt(double *point)

Description of Command: Get the break point position of all axes after stopping multi-segment continuous path motion.

Command Parameter: "*point" returns break point position of all axes. The position are saved in the array in the order of X, Y, Z.

Applicable Card: GE-X00-SX

GT_GetCapt

Command Prototype: short GT_GetCapt(unsigned short axis, long *pos)

Description of Command: This command gets the actual position of specified axis captured by Index or Home signal triggered. This value keeps as same until the next capture event happens.

Command Parameter: "axis" is the specified axis number, "*pos" returns the captured position value by the specified axis.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: refer to example 7-1.

GT_GetCmdSts

Command Prototype: short GT_GetCmdSts(unsigned short *status)

Description of Command: This command gets the contents of the command status register of the controller. In main user program, detect the return value of each command to tell the execution status of communication and command. If the command return value is 1, it indicates that the command from the host is wrong, and the controller will neglect these illegal motion commands and report error reason in the command status register. User can evoke this command to get the reason for command error.

Command Parameter: "*status" returns the value of the register. This register is a 16-bit register. The definition of each bit is described in Chapter 2.

Applicable Card: GE-X00-PX，GE-X00-SX

GT_GetCrdSts

Command Prototype: short GT_GetCrdSts(unsigned short *status)

Description of Command: Get the motion status of interpolation motion.

Command Parameter: "*status" returns motion status. The definition of each bit in motion status are described in Chapter 4 and Chapter 6.

Applicable Card: GE-X00-SX

Command Evoking: The following example gets the motion status of interpolation motion, if find bit2 is set, inactivate all axes.

```
short CoordState()
{
    short rtn;
    unsigned short status;
    rtn=GT_GetCrdSts(&status);
    if(0==rtn) return rtn;
    if(status&0x4)
    {
        rtn=GT_AxisOff(1);
        if(0==rtn) return rtn;
        rtn=GT_AxisOff(2);
        if(0==rtn) return rtn;
        rtn=GT_AxisOff(3);
        if(0==rtn) return rtn;
    }
    return 0;
}
```

GT_GetCurrentCardNo

Command Prototype: short GT_GetCurrentCardNo(void)

Description of Command: This command gets the current motion control card number, ranged from 0 to 15.

Applicable Card: GE-X00-PX, GE-X00-SX

Command Evoking:

```
short main()
{
    long pos[3];
    short CardNo;
    rtn=GT_Open();                          //Open first card
    if(0!=rtn) return rtn;
    rtn=GT_Open();                          //Open second card
    if(0!=rtn) return rtn;
```

```
    CardNo = GT_GetCurrentCardNo ();                    //read current card number
    printf("Card No:%d",CardNo);
    rtn=GT_SwitchtoCardNo(1);                           //Specify the first card as
    current card
    if(0!=rtn) return rtn;
    rtn=GT_Close();                                     //Close current card
    if(0!=rtn) return rtn;

    rtn=GT_SwitchtoCardNo(2);                           //Specify the second card as
    current card
    if(0!=rtn) return rtn;
    CardNo = GT_GetCurrentCardNo ();                    //Read current card number
    if(0!=rtn) return rtn;
    printf("Card No:%d",CardNo);
    rtn=GT_Close();
    if(0!=rtn) return rtn;
    return 0;
}
```

GT_GetHomeSwt

Command Prototype：short GT_GetHomeSwt(unsigned short *home)

Description of Command：Get the home switch status. If some home switch is at high level, the corresponding bit in home switch status is 1. If some home switch is at low level, the corresponding bit is 0. Please refer to Chapter 9 to read the description in details.

Command Parameter: "*home" returns the home switch status of all axes

Applicable Card: GE-X00-PX, GT-X00-SX.

Command Evoking：

```
short DisplayHomeSwitch()
{
    unsigned short home,i,bit=1;
    short rtn;
    rtn = GT_GetHomeSwt(&home);
    if(0!=rtn) return rtn;
    for(i=0;i<8;++i)
    {
        if(home&bit)
        {
            printf("\nHome%d is high",i);
        }
        else
```

```
            {
                printf("\nHome%d is low",i);
            }
            bit<<=1;
        }
        return 0
}
```

GT_GetILmt

Command Prototype: short GT_GetILmt(unsigned short axis, unsigned short *limit)

Description of Command: This command gets the error integral limit of the specified axis set by the command GT_SetILmt .

Command Parameter: "axis" is the specified axis number, "*limit" returns the error integral limit.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking: refer to GT_GetKp

GT_GetIntgr

Command Prototype: short GT_GetIntgr(unsigned short axis, short *integral)

Description of Command: This command gets the integral error of the specified axis servo filter.

Command Parameter: "axis" is the specified axis number, "*integral" returns the integral error.

Applicable Card: GE-X00-PV，GE-X00-SVv

Command Evoking: refer to GT_GetKp

GT_GetJerk

Command Prototype: short GT_GetJerk(unsigned short axis, double *jerk)

Description of Command: This command gets the jerk of the specified axis set by the command GT_SetJerk .

Command Parameter: "axis" is the specified axis number, "*jerk" returns the jerk.

Applicable Card: GE-X00-PX

Command Evoking: refer to GT_GetPos。

GT_GetKaff

Command Prototype: short GT_GetKaff(unsigned short axis, double *kaff)

Description of Command: This command gets the acceleration feed-forward gain of the specified axis set by the command GT_SetKaff.

Command Parameter: "axis" is the specified axis number, "*kaff" returns the acceleration

feed-forward gain.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking: refer to GT_GetKp

GT_GetKd

Command Prototype: short GT_GetKd(unsigned short axis, double *kd)

Description of Command: This command gets the differential gain of the specified axis set by the command GT_SetKd.

Command Parameter: “axis” is the specified axis number, “*kd” returns the differential gain.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking: refer to GT_GetKp

GT_GetKi

Command Prototype: short GT_GetKi(unsigned short axis, double *ki)

Description of Command: This command gets the integral gain of the specified axis set by the command GT_SetKi .

Command Parameter: “axis” is the specified axis number, “*ki” returns the integral gain.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking: refer to GT_GetKp

GT_GetKp

Command Prototype: short GT_GetKp(unsigned short axis, double *kp)

Description of Command: This command gets the percentage gain of the specified axis set by the command GT_SetKp.

Command Parameter: “axis” is the specified axis number, “*kp” returns the percentage gain.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking:

```
short GetPid()
{
    short rtn;
    double kp,ki,kd,kaff,kvff;
    short bias;
    unsigned short motorLimit, intgrLimit,intgr,posErr;
    rtn = GT_GetKp(1,&kp);                    //gets the percentage gain of the
    first axis
    if(0!=rtn) return rtn;
    rtn = GT_GetKi(1,&ki);                    //gets the integral gain of the
    first axis
    if(0!=rtn) return rtn;
    rtn = GT_GetILmt(1,&intgrLimit);          //gets the error integral limit of the
```

```
        first axis
        if(0!=rtn) return rtn;
        rtn = GT_GetKd(1,&kd);                          //gets the differential gain of the
        first axis
        if(0!=rtn) return rtn;
        rtn = GT_GetKaff(1,&kaff);          //gets the acceleration feed-forward gain of
        the first axis
        if(0!=rtn) return rtn;
        rtn = GT_GetKvff(1,&kvff);            //gets the velocity feed-forward gain of
        the first axis
        if(0!=rtn) return rtn;
        rtn = GT_GetMtrBias(1,&bias);         //gets the net difference compensation of
        the first axis
        if(0!=rtn) return rtn;
        rtn = GT_GetMtrLmt(1,&motorLimit);                  //gets the output limit of the
        first axis
        if(0!=rtn) return rtn;
        rtn = GT_GetIntgr(1,&intgr);                        //gets the integral error of the
        first axis
        if(0!=rtn) return rtn;
        rtn = GT_GetPosErr(1,&posErr);              //gets the position error limit of the
        first axis
        if(0!=rtn) return rtn;
        printf("\n kp=%lf ki=%lf kd=%lf kvff=%lf kaff=%lf bias=%d IntgrLimit=%d
        MotorLimit=%d                                                    intgr=%d
        PosErr=%d",kp,ki,kd,kvff,kaff,bias,intgrLimit,motorLimit,intgr,posErr);
}
```

GT_GetKvff

Command Prototype: short GT_GetKvff(unsigned short axis, double *kvff)

Description of Command: This command gets the velocity feed-forward gain of the specified axis set by the command GT_SetKvff.

Command Parameter: "axis" is the specified axis number, "*kvff" returns the velocity feed-forward gain.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking: refer to GT_GetKp

GT_GetLmtSwt

Command Prototype: short GT_GetLmtSwt(unsigned short *limit)

Description of Command: This command gets the actual status of current limit switch.

Command Parameter: "*limt" returns the level of the limit switch input signal, as defined in the table 9-5. If a bit = 1, it means that the input signal of the corresponding switch is at high level. If a bit = 0, it means that the input signal is at low level. This command is not related with the command GT_LmtSns, and only shows the actual status of limit switch.

Applicable Card: GE-X00-PX, GE-X00-SX

Command Evoking:

```
short DisplayLimitSwitch()
{
    unsigned short limit, i, bit=1;
    short rtn;
    rtn = GT_GetLmtSwt(&limit);
    if(0!=rtn)  return  rtn;
    for(i=0;i<8;++i)
    {
        if(limit&bit)
        {
            printf("\nLimit%d＋ is high",i);
        }
        else
        {
            printf("\nLimit%d＋ is low",i);
        }
        bit<<=1;
        f(limit&bit)
        {
            printf("\nLimit%d－ is high",i);
        }
        else
        {
            printf("\nLimit%d－ is low",i);
        }
        bit<<=1;
    }
    return 0;
}
```

GT_GetMAcc

Command Prototype: short GT_GetMAcc(unsigned short axis,double *macc)

Description of Command: This command gets the maximum acceleration of the specified axis set by the command GT_SetMAcc.

Command Parameter: “axis” is the specified axis number, “*macc” indicates the maximum acceleration.

Applicable Card: GE-X00-PX

Command Evoking: refer to GT_GetPos

GT_GetMode

Command Prototype: short GT_GetMode(unsigned short axis,unsigned short *mode)

Description of Command: This command gets the control mode character of the specified axis.

Command Parameter: “axis” is the specified axis number, “*mode” returns the control mode character. Please refer to Chapter 4 to get the definition of each bit.

Applicable Card: GE-X00-PX

Command Evoking: refer to GT_GetPos

GT_GetMtnNm

Command Prototype: short GT_GetMtnNm(unsigned short *number)

Description of Command: Get the working path segment number, After user uses the commands GT_EStpMtn and GT_StpMtn to stop multi-segment continuous path motion, user can call this command to get the number of stopped segment.

Command Parameter: “*number” returns the working path segment number or the number of stopped segment.

Applicable Card: GE-X00-SX

GT_GetMtrBias

Command Prototype: short GT_GetMtrBias(unsigned short axis,short *bias)

Description of Command: This command gets the net difference compensation of the specified axis set by the command GT_SetMtrBias.

Command Parameter: “axis” is the specified axis number, “*bias” returns the net difference compensation.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking: refer to GT_GetKp.

GT_GetMtrCmd

Command Prototype: short GT_GetMtrCmd(unsigned short axis,short *voltage)

Description of Command: In open loop status, this command gets the motor control output value of the specified axis set by the command GT_SetMtrCmd.

Command Parameter: “axis” is the specified axis number; *voltage returns the motor control output value. When the specified axis is in close loop mode, the return value of parameter *voltage has no meaning.

Applicable Card: GE-X00-PV

GT_GetMtrLmt

Command Prototype: short GT_GetMtrLmt(unsigned short axis, unsigned short *limit)

Description of Command: This command gets the output limit of the specified axis set by the command GT_SetMtrLmt.

Command Parameter: “axis” is the specified axis number; “*limit” returns the output limit.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking: refer to GT_GetKp.

GT_GetPos

Command Prototype: short GT_GetPos(unsigned short axis, long *pos)

Description of Command: This command gets the position value of the specified axis set by the command GT_SetPos.

Command Parameter: “axis” is the specified axis number, “*pos” returns the target position value.

Applicable Card: GE-X00-PX

Command Evoking：

```
short GetMotionParameter()
{
    short rtn;
    long pos;
    double vel,acc,macc,jerk;
    unsigned long status;
    unsigned short mode;
    rtn = GT_GetPos(1,&pos);
    if(0!=rtn) return rtn;
    rtn = GT_GetVel(1,&vel);
    if(0!=rtn) return rtn;
    rtn = GT_GetAcc(1,&acc);
    if(0!=rtn) return rtn;
    rtn = GT_GetMAcc(1,&macc);
    if(0!=rtn) return rtn;
    rtn = GT_GetJerk(1,&jerk);
    if(0!=rtn) return rtn;
    rtn = GT_GetSts(1,&status);
    if(0!=rtn) return rtn;
    rtn = GT_GetMode(1,&mode);
    if(0!=rtn) return rtn;
    printf(“\n Pos = %ld Vel = %lf Acc = %lf MAcc = %lf Jerk = %lf Status = %lx Mode =
```

```
        %x",
        pos, vel, acc, macc, jerk, status, mode);
        return 0;
}
```

GT_GetPosErr

Command Prototype: short GT_GetPosErr(unsigned short axis,unsigned short *error)

Description of Command: This command gets the position error limit of the specified axis set by the command GT_SetPosErr.

Command Parameter: "axis" is the specified axis number, "*error" returns the position error limit.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking: refer to GT_GetKp.

GT_GetPrfPnt

Command Prototype: short GT_GetPrfPnt(double *point)

Description of Command: Get the planning position of all axes in interpolation motion.

Command Parameter: "*point" points to an array with a length of 3, and returns the planning position of all axes in interpolation motion in the order of Axis X, Y, Z.

When evoking this command, use should define a double-precision array with same length, and set the first address of the array as the physical parameter of the command.

Applicable Card: GE-X00-SX

Command Evoking: The following example gets and prints planning position of each axis.

```
short GetPrfPnt()
{
        short rtn;
        double crd_prf_pos[3];
        rtn=GT_GetPrfPnt(crd_prf_pos);
        if(0!=rtn) return rtn;
        printf("the coordinate pos are: %ld, %ld, %ld \n", crd_prf_pos[0], crd_prf_pos[1],
        crd_prf_pos[2]);
        return 0;
}
```

GT_GetPrfPos

Command Prototype: short GT_GetPrfPos(unsigned short axis,long *pos)

Description of Command: This command gets the planned position of the specified axis.

Command Parameter: "axis" is the specified axis number, "*pos" returns the planned position.

Applicable Card: GE-X00-PX。

Command Evoking: refer to GT_GetAtlPos。

GT_GetPrfVel

Command Prototype：short GT_GetPrfVel(unsigned short axis,double *vel)

Description of Command：This command gets the planned velocity of the specified axis.

Command Parameter: “axis” is the specified axis number, “*vel” returns the planned velocity.

Applicable Card：GE-X00-PX。

Command Evoking: refer to GT_GetAtlVel

GT_GetPrfVel

Command Prototype: short GT_GetPrfVel(double *vel)

Description of Command: This command gets the planned path velocity of the interpolation motion.

Command Parameter: “*vel” returns the current planned path velocity.

Applicable Card: GE-X00-SX

GT_GetSegPnt

Command Prototype: short GT_GetSegPnt(double *point)

Description of Command: This command gets the end point position of the specified working segment.

Command Parameter: “*point” points to an array with a length of 3, and returns the end point position of current working segment in the order of Axis X, Y, Z.

Applicable Card: GE-X00-SX

GT_GetSts

Command Prototype: short GT_GetSts(unsigned short axis, unsigned long *status)

Description of Command: This command gets the status character of the specified axis. Please refer to Chapter 4 to get the definition of each bit in axis status.

Command Parameter: “axis” is the specified axis number, “*status” returns the status character.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: refer to GT_GetPos

GT_GetVel

Command Prototype: short GT_GetVel(unsigned short axis, double *vel)

Description of Command: This command gets the velocity of the specified axis set by the

command GT_SetVel().

Command Parameter: “axis” is the specified axis number, “*vel” returns the velocity.

Applicable Card: GE-X00-PX

Command Evoking: refer to GT_GetPos.

GT_HardRst

Command Prototype: short GT_HardRst(void)

Description of Command: This command restores the motion controller. After evoking this command, the host will output a restoring signal. So, in any condition, GT_HardRst can restore the controller, and reset all the initial status of the control register, which are as follows.

1. Set all the motion parameters register to 0.
2. Set all the position capture register to 0.
3. Set the motion mode as T-curve independent positioning (GE-X00-PX).
4. Set the parameters of PID filter to 0 (GE-X00-XV).
5. Limit the integral of all axes to 32767 (GE-X00-XV).
6. The output voltage limit value of all axes is 32767 (GE-X00-XV).
7. Set the position error limit of all axes to 32767 (GE-X00-XV).
8. Prohibit the automatic stopping of four axes due to motion error (GE-X00-PV).
9. Set all axes as in closed loop mode (GE-X00-XV).
10. The control output of all axes is default mode.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: This command is evoked normally when the motion controller has error not restorable.

```
short HardReset()
{
    short rtn;
    unsigned short exInpt;
    rtn = GT_ExInpt(&exInpt);
    if(0!=rtn) return rtn;
    if(exInpt&1)
    {
        return GT_HardRst();
    }
}
```

GT_HomeSns

Command Prototype：short GT_HomeSns(unsigned short sense)

Description of Command：Set trigger edge of HOME signal. The detail references the chapter 3.

Command Parameter: The control word of home sense.

Applicable Card：GE-X00-PX，GE-X00-SX.

Command Evoking: while the EXI0, digital input, is high level, the home capture of axis 1 is triggered by rising edge. Or not, it is triggered by trailing edge.

```
short HomeConfig()
{
    short rtn;
    unsigned short exInpt;
    rtn = GT_ExInpt(&exInpt);
    if(0!=rtn) return rtn;
    if(exInpt&1)
    {
        return GT_HomeSns(1);
    }
    else
    {
        return GT_HomeSns(0);
    }
}
```

GT_HandWheel

Command Prototype：short GT_HandWheel(unsigned short axis, double ratio)

Description of Command：This command sets specified axis into hand wheel control mode. GE-300-SX provides an input port for hand wheel, it can receive the hand wheel (single-ended signal or differential signal) input or auxiliary encoder input.

When the system is not in motion, user can call this command to set specified axis into hand wheel control mode, and specify the feed override. After that, when user turn over the hand wheel, the specified axis will follow the hand wheel with specified override.

Once in hand wheel control mode, GE-X00-SX should not accept any motion path command or buffer management command. If some axis is in hand wheel control mode, while keeps no motion for several seconds, user can change axis number or feed override.

Command Parameter: “axis” is specified axis that follow the hand wheel, “ratio” specify the feed override.

Applicable Card: GE-X00-SX

Command Evoking: refer to example 10-5

GT_HookCommand

Command Prototype：short GT_HookCommand(TCommandHandle CommandHandle)

Description of Command: hook command handler. The detail refers to chapter 2.

Command Parameter：CommandHandle is the handler of the specified function to manager value by command returned. TcommandHandle is defined as following.

typedef void (__stdcall * TCommandHandle)(char * command,short error).

Applicable Card: GE-X00-PX，GE-X00-SX。

Command Evoking: refer to example 2-2, 2-3 and 2-4.

GT_IndexSns

Command Prototype: short GT_IndexSns(unsigned short sense)

Description of Command: Set trigger edge of INDEX signal.

Command Parameter: Set trigger edge of INDEX signal.

Applicable Card：GE-X00-PX。

GT_InitLookAhead

Command Prototype：short GT_InitLookAhead(double t，double acc_max，double acc，double vel，int n，double con)

Description of Command：Initialize the technological parameters of machine tool

Command Parameter：

t: Maximum time of turning the corner. unit: s, user must pay attention to the coincidence of the time, i.e. both the unit of feed speed and acceleration are second. The experience value range of t is: **0.001~0.01S**.

Longer T, higher velocity, while lower accuracy. On the contrary, shorter T, lower velocity, while higher accuracy. Please be sure to select reasonable value of t.

acc_max: Maximum acceleration, unit: mm/s^2. the value is related with machine tool and motor. experience value: 100~10000 mm/s^2.

Acc: Path acceleration, unit: mm/s^2

vel：Feed speed, unit: mm/s.

n：Length of Look-Ahead block.

con：Pulse per mm.

Applicable Card：GE-X00-SX

GT_LmtSns

Command Prototype: short GT_LmtSns(unsigned short sense)

Description of Command: This command sets the effective level of the limit switch. If the corresponding bit is set to 0, it means that the input signal of limit switch triggers limit at high level. If it is set to 1, it means that the input signal of limit switch triggers limit at low level. Default the high level triggers limit switch.

Command Parameter: the effective level of the limit switch.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: refer to example3-2, 3-3 and 3-4.

GT_LmtsOff

Command Prototype: short GT_LmtsOff(unsigned short axis)

Description of Command: This command prohibits the motion controller to monitor the limit switch status of specified axis. But the status is still reflected in the limit switch status register of the controller. User can evoke GT_GetLmtSwt to inquire the status of limit switch. The change of the status will not cause the controller to perform any operation.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: refer to GT_LmtsOn

GT_LmtsOn

Command Prototype: short GT_LmtsOn(unsigned short axis)

Description of Command: This command makes the motion controller to start monitoring the limit switch status of current axis. The default status of the controller is monitoring the limit.

Command Parameter: Specified axis.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: Disable limit monitor of axis 1 and enable limit monitor of axis 2.

```
short LimitConfig()
{
    short rtn;
    rtn = GT_LmtsOff(1);
    if(0!=rtn) return rtn;
    rtn = GT_LmtsOn(2);
    if(0!=rtn) return rtn;
    return GT_ClrSts(1);
}
```

GT_LnXY

Command Prototype: short GT_LnXY(double x, double y)

Description of Command: This command realizes the 2D linear interpolation motion.

Command Parameter: “x” and “y” are the end position of X and Y axis, unit: pulse. The start positions are the end position of the last segment (in buffer motion mode) or current position (in immediate motion mode). If it is the first motion path segment in buffer, the start position is the position specified by GT_MvXY or GT_MvXYZ.

Applicable Card: GE-X00-SX

Command Evoking: The following example executes the command of 2D linear interpolation immediately.

```
short LineXY()
{
    short rtn;
    rtn=GT_SetSynVel(3);
    if(0==rtn) retrun rtn;
    rtn=GT_SetSynAcc(1);
    if(0==rtn) retrun rtn;
    rtn=GT_LnXY(12345,67890);
    if(0==rtn) retrun rtn;
    return 0;
}
```

GT_LnXYZ

Command Prototype: short GT_LnXYZ(double x, double y, double z)

Description of Command: This command realizes the 3D linear interpolation motion.

Command Parameters: “x”, “y” and “z” are the end position of X, Y and Z axis, unit: pulse. The start positions are the end position of the last segment (in buffer motion mode) or current position (in immediate motion mode). If it is the first motion path segment in buffer, the start position is the position specified by GT_MvXY or GT_MvXYZ.

Applicable Card: GE-X00-SX

Command Evoking: The following example pushes the command of 3D linear interpolation into the buffer, start to execute.

```
short LineXYZ()
{
    short rtn;
    rtn=GT_StrtList();
    if(0==rtn) retrun rtn;
    rtn=GT_MvXYZ(0,0,0,6,0.3);
```

```
    if(0==rtn) retrun rtn;
    rtn=GT_LnXYZ(12945,58372,65473);
    if(0==rtn) retrun rtn;
    rtn=GT_StrtMtn();
    if(0==rtn) retrun rtn;
    …
}
```

GT_LnXYG0

Command Prototype: short GT_LnXYG0(double x, double y)

Description of Command: This command realizes the 2D linear interpolation motion with symmetry profile of velocity planning.

Command Parameters: “x” and “y” are the end position of X and Y axis, unit: pulse. The start positions are the end position of the last segment (in buffer motion mode) or current position (in immediate motion mode). If it is the first motion path segment in buffer, the start position is the position specified by GT_MvXY or GT_MvXYZ.

Applicable Card: GE-X00-SX。

Command Evoking: The following example executes the command of 2D linear interpolation with symmetry profile of velocity planning immediately.

```
short LineXYG0()
{
    short rtn;
    rtn=GT_SetSynVel(30);
    if(0==rtn) retrun rtn;
    rtn=GT_SetSynAcc(1);
    if(0==rtn) retrun rtn;
    rtn=GT_LnXYG0(12345,67890);
    if(0==rtn) retrun rtn;
    return 0;
}
```

GT_LnXYZG0

Command Prototype: short GT_LnXYZG0(double x, double y, double z)

Description of Command: This command realizes the 3D linear interpolation motion with symmetry profile of velocity planning.

Command Parameters: “x”, “y” and “z” are the end position of X, Y and Z axis, unit: pulse. The start positions are the end position of the last segment (in buffer motion mode) or current position (in immediate motion mode). If it is the first motion path segment in buffer, the start position is the position specified by GT_MvXY

or GT_MvXYZ.

Applicable Card：GE-X00-SX

Command Evoking：The following example pushes the command of 3D linear interpolation with symmetry profile of velocity planning into the buffer, and starts to execute.

```
short LineXYZG0()
{
    short rtn;
    rtn=GT_StrtList();
    if(0==rtn) retrun rtn;
    rtn=GT_MvXYZ(0,0,0,6,0.3);
    if(0==rtn) retrun rtn;
    rtn=GT_LnXYZG0(12945,58372,65473);
    if(0==rtn) retrun rtn;
    rtn=GT_StrtMtn();
    if(0==rtn) retrun rtn;
    …
}
```

GT_MapCnt

Command Prototype: short GT_MapCnt(unsigned short axis, double count)

Description of Command: This command Specify the coordinate offset of the specified axis. User can use this command to manage working coordinate system.

Command Parameter: “axis” is specified axis number, “count” specify the coordinate offset.

Applicable Card：GE-X00-SX

GT_MltiUpdt

Command Prototype: short GT_MltiUpdt(unsigned short mask)

Description of Command: This command becomes parameters of several axes effective immediately. The relation between the mask and the axes refers to following table.

status	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
bit	8	7	6	5	4	3	2	1

Command Parameter: “mask” specifies axes to update parameters.

Applicable Card: GE-X00-PX

Command Evoking: The following example updates the parameter Kp for the first and third axes at the same time.

```
void main()
{
    GT_HookCommand(CommandHandle);
```

```
    GT_Open();                                  //open motion controller
    GT_Reset();                                 //reset motion controller
    GT_SetKp(1,10);                             //set the percentage gain of axis
    1
    GT_SetKp(3,10);                             //set the percentage gain of axis
    3
    GT_MltiUpdt(5);                             //multi update axis 1 and axis 3
    GT_AxisOn(1);                               //axis 1 servo on
    GT_AxisOn(3);                               //axis 3 servo on
    printf("\nPress any key to continue…");
    getch();
    GT_PrflV(1);                                //set parameters for axis 1
    GT_SetVel(1,10);
    GT_SetAcc(1,0.1);
    GT_PrflV(3);                                //set parameters for axis 3
    GT_SetVel(3,10);
    GT_SetAcc(3,0.1);
    GT_MltiUpdt(5);                             //multi update axis 1 and axis 3
}
```

GT_MvXY

Command Prototype: short GT_MvXY(double x, double y,double vel,double acc)

Description of Command: For a 2D coordinate system, this command must be used as the first buffer command following the command GT_StrtList, to specify start point position (2D), path velocity and acceleration in buffer.

This command can only be evoked after the command GT_StrtList, and it can not be revoked if the first one is received successfully.

Command Parameter: "x" and "y" are the start point position, unit: pulse. "vel" is the path velocity, unit: pulse/ms. "acc" is the path acceleration, unit: pulse/ms^2. When the command GT_StrtMtn is evoked, GE-X00-SX will move all axes in coordinate system from current position to the start position specified by GT_MvXY in the linear interpolation mode, then execute path commands in buffer in order.

Applicable Card: GE-X00-SX

Relevant Command: GT_MvXYZ

Command Evoking: In the following example, After calls the command GT_StrtList, calls GT_MvXY to specify the start point of the motion in buffer at the point (1000, 2000), path velocity as 1 pulse/ms and path acceleration as 0.1 pulse/ ms^2.

```
short MoveXY()
{
    short rtn;
```

```
    rtn=GT_StrtList();
    if(0==rtn) retrun rtn;
    rtn=GT_MvXY (1000,2000, 1, 0.1);
    if(0==rtn) retrun rtn;
    …
    rtn=GT_StrtMtn();
    if(0==rtn) retrun rtn;
    return 0;
}
```

GT_MvXYZ

Command Prototype: short GT_MvXYZ (double x, double y, double z, double vel, double acc)

Description of Command: For a 3D coordinate system, this command must be used as the first buffer command following the command GT_StrtList, to specify start point position (3D), path velocity and acceleration in buffer.

This command can only be evoked after the command GT_StrtList, and it can not be revoked if the first one is received successfully.

Command Parameter: “x”, “y” and “z” are the start point position, unit: pulse. “vel” is the path velocity, unit: pulse/ms. “acc” is the path acceleration, unit: pulse/ms^2. When the command GT_StrtMtn is evoked, GE-X00-SX will move all axes in coordinate system from current position to the start position specified by GT_MvXYZ in the linear interpolation mode, then execute path commands in buffer in order..

Applicable Card: GE-X00-SX

Relevant Command: GT_MvXY

Command Evoking: refer to GT_ MvXY

GT_Open

Command Prototype: short GT_Open(unsigned long address=65535)

Description of Command: Evoking GT_Open to open the motion controller device. For ISA bus card, need to specify the base address. For PCI bus card, only to make sure the address equal to 65535.

Command Parameter: For ISA bus card, need to specify the base address. For PCI bus card, only to make sure the address equal to 65535.

Applicable Card: GE-X00-PX，GE-X00-SX。

Command Evoking: refer to example 5-1.

GT_OpenLp

Command Prototype: short GT_OpenLp(unsigned short axis)

Description of Command: This command sets the specified axis as in the servo open loop control status. It is mainly used in controlling the motor directly. Generally, the servo driver shall be in the close loop control status. In this control mode, GT_SetMtrCmd can be called to set the value of the voltage. The relation between the voltage output and the value refers to the following table.

Voltage output（V）	value
＋10	32767
0	0
－10	-32768

Applicable Card: GE-X00-PV

Command Evoking: refer to GT_SetMtrCmd

GT_Override

Command Prototype: short GT_Override(double override)

Description of Command: Specify the feed override of the motion path description command except command GT_LnXYG0, GT_LnXYZG0, GT_MvXY, GT_MvXYZ.

This command can be called to modify the feed override, which means feed speed can be modified on the fly. Once the override has been modified successfully, the override will affect the feed speed of the path moiton followed until GT_Override is called again.

Command Parameter: “override” is the specified feed override.

Applicable Card: GE-X00-SX

GT_OverrideG0

Command Prototype: short GT_OverrideG0(double override)

Description of Command: Specify the feed override of GT_LnXYG0, GT_LnXYZG0, GT_MvXY or GT_MvXYZ.

This command can be called to modify the feed override, which means feed speed can be modified on the fly. Once the override has been modified successfully, the override will affect the feed speed of the path moiton followed until GT_OverrideG0 is called again

Command Parameter: “override” is the specified feed override.

Applicable Card: GE-X00-SX。

GT_PrflS

Command Prototype: short GT_PrflS(unsigned short axis)

Description of Command: This command sets the independent axis motion mode of the

specified axis as S-curve independent positioning mode.

Command Parameter: Specified axis number

Applicable Card: GE-X00-PX

Command Evoking: refer to example 5-2

GT_PrflT

Command Prototype: short GT_PrflT(unsigned short axis)

Description of Command: This command sets the independent axis motion mode of specified axis as T-curve independent positioning mode.

Command Parameter: Specified axis number

Applicable Card: GE-X00-PX

Command Evoking: refer to example 5-1

GT_PrflV

Command Prototype: short GT_PrflV(unsigned short axis)

Description of Command: This command sets the motion control mode of specified axis as independent jogging mode.

Command Parameter: Specified axis number

Applicable Card: GE-X00-PX

Command Evoking: refer to example 5-3

GT_Reset

Command Prototype: short GT_Reset(void)

Description of Command: This command enables the host to reset the motion controller with a command. Its result is the same as that of the command GT_HardRst.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: refer to example 5-1

GT_RestoreMtn

Command Prototype：short GT_RestoreMtn (void)

Description of Command：Restore multi-segment continuous path motion from the breakpoint after stop the motion.

Before calling GT_RestoreMtn, user must confirm that multi-segment continuous path motion had been started and has been stopped. GT_StartMtn is not allowed to restore multi-segment continuous path motion.

Applicable Card: GE-X00-SX.

GT_RstSts

Command Prototype: short GT_RstSts(unsigned short axis, unsigned short mask)

Description of Command: This command clears the status register of the specified axis according to the definition of Mask.

Command Parameter: “axis” is the specified axis number; “mask” is consistent with the definition of Bit0 – Bit7 in the attached table of the command GT_RstIntr. When a bit of Mask is set to 1, this bit is allowed to be reserved. If it is set to 0, this event symbol is to be cleared.

Applicable Card: GE-X00-PX, GE-X00-SX

Command Evoking: refer to GT_SetAtlPos。

GT_SetAcc

Command Prototype: short GT_SetAcc(unsigned short axis, double acc)

Description of Command: This command sets the acceleration parameter of the specified axis. It is only effective in the T-curve velocity mode and velocity control mode.

Command Parameter: “axis” is the specified axis number; “acc” is the acceleration to be set, ranged from 0 to 102400. The unit of acceleration is pulse/ms^2. To make the new parameter set effective requires the evoking of the command GT_Update or GT_MltiUpdt.

Applicable Card: GE-X00-PX

Command Evoking: refer to example 5-1 and 5-3

GT_SetAdcChn

Command Prototype: short GT_SetAdcChn(unsigned short channel)

Description of Command: This command sets the number of AD convert channels.

Command Parameter: The parameter value sets the number of AD channels, ranged from 1 to 8.

Applicable Card: GE-X00-SX

GT_SetAtlPos

Command Prototype: short GT_SetAtlPos(unsigned short axis, long pos)

Description of Command: This command sets the actual, planning and target positions of the specified axis to be the specified value. This command is only effective when the specified axis is no motion. Otherwise, it will be considered as an illegal command with a motion error symbol generated.

Command Parameter: “axis” is the specified axis number; “pos” is the specified value of actual position.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: The following example sets the actual and planning position of the first axis to be zero.

```
void main()
{
    GT_HookCommand(CommandHandle);
    GT_Open();
    GT_Reset();
    AxisInitial(1,0,0);                          //reference to example 3-2
    AxisRunT(1,10000,1,0.1);                     // reference to example 5-1
    unsigned long status;
    do
    {
        GT_GetSts(1,&status);                    //read the status of the first
        axis
    }while(status&0x400);
    GT_SetAtlPos(1,0);                  //if the first axis is not in motion, call this
    command
    GT_RstSts(1,0xfe);
}
```

GT_SetDccVel

Command Prototype: short GT_SetDccVel (double vel)

Description of Command: Specify segment end velocity. This command should be called to specify the segment end velocity in buffer, GE-X00-SX will plan the working velocity as the specified path velocity and end velocity.

Command Parameter: “vel” is the value of end velocity.

Applicable Card: GE-X00-SX

GT_SetILmt

Command Prototype：short GT_SetILmt(unsigned short axis,unsigned short limit)

Description of Command：This function sets the error integral limit of the specified axis servo filter.

Command Parameter: “axis” is the specified axis number; limit is the error integral limit to be set, ranged from 0 to 32767. To make effective the newly set parameter requires to evoke the command GT_Update or GT_MltiUpdt

Applicable Card：GE-X00-PV，GE-X00-SV

Command Evoking: refer to GT_SetKp

GT_SetJerk

Command Prototype: short GT_SetJerk(unsigned short axis,double jerk)

Description of Command: This command is used to set the jerk of specified axis in S-curve mode. The unit of jerk is plus/ ms^3.

Command Parameter: “axis” is the specified axis number; Jerk is the jerk to be set, ranged from 0 to 256000. To make effective the newly set parameter requires to evoke the command GT_Update or GT_MltiUpdt first.

Applicable Card: GE-X00-PX

Command Evoking: refer to example 5-2

GT_SetKaff

Command Prototype: short GT_SetKaff(unsigned short axis,double kaff)

Description of Command: This command sets the acceleration feedback gain of the specified axis servo filter.

Command Parameter: “axis” is the specified axis number; “kaff” is the acceleration feedback gain to be set, ranged from 0 to 32767. To make effective the newly set parameter requires to evoke the command GT_Update or GT_MltiUpdt first.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking: refer to GT_SetKp

GT_SetKd

Command Prototype: short GT_SetKd(unsigned short axis,double kd)

Description of Command: This command sets the differential gain of the specified axis servo filter.

Command Parameter: “axis” is the specified axis number; “kd” is the differential gain to be set, ranged from 0 to 32767. To make effective the newly set parameter requires to evoke the command GT_Update or GT_MltiUpdt first.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking：refer to GT_SetKp.

GT_SetKi

Command Prototype: short GT_SetKi(unsigned short axis,double ki)

Description of Command: This command sets the integral gain of the specified axis servo filter.

Command Parameter: “axis” is the specified axis number; “ki” is the integral gain to be set, ranged from 0 to 32767. To make effective the newly set parameter requires to evoke the command GT_Update or GT_MltiUpdt first.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking: refer to GT_SetKp.

GT_SetKp

Command Prototype: short GT_SetKp(unsigned short axis,double kp)

Description of Command: This command sets the percentage gain of the specified axis servo filter.

Command Parameter: “axis” is the specified axis number; “kp” is the percentage gain to be set, ranged from 0 to 32767. To make effective the newly set parameter requires to evoke the command GT_Update or GT_MltiUpdt first.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking: for axis 1, set parameters of the servo filter.

```
short PidConfig()
{
    short rtn;
    rtn = GT_SetKp(1,10);                    //set the percentage gain
    if(0!=rtn) return rtn;
    rtn = GT_SetKi(1,20);                    //set the integral gain
    if(0!=rtn) return rtn;
    rtn = GT_SetILmt(1,20000);               //set the error integral limit
    if(0!=rtn) return rtn;
    rtn = GT_SetKd(1,5);                     // set the differential gain
    if(0!=rtn) return rtn;
    rtn = GT_SetKaff(1,10);                  // set the acceleration feedback
    gain
    if(0!=rtn) return rtn;
    rtn = GT_SetKvff(1,10);                  // set the velocity feedback gain
    if(0!=rtn) return rtn;
    rtn = GT_SetMtrBias(1,-200);             // set the bias compensation
    if(0!=rtn) return rtn;
    rtn = GT_SetMtrLmt(1,6*3276);            // set the voltage limit
    if(0!=rtn) return rtn;
    rtn = GT_SetPosErr(1,32767);             // set the position error limit
    if(0!=rtn) return rtn;
    return GT_Update(1);                     //the parameters become
    effective
}
```

GT_SetKvff

Command Prototype: short GT_SetKvff(unsigned short axis,double kvff)

Description of Command: This command sets the velocity feedback gain of the current axis servo filter.

Command Parameter: “axis” is the specified axis number; “kvff” is the velocity feedback

gain to be set, ranged from 0 to 32767. To make effective the newly set parameter requires to evoke the command GT_Update or GT_MltiUpdt first.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking: refer to GT_SetKp.

GT_SetMAcc

Command Prototype: short GT_SetMAcc(unsigned short axis,double macc)

Description of Command: When the control mode is S-curve mode, evoke this command to set the maximum acceleration of the specified axis.

Command Parameter: “axis” is the specified axis number; “macc” is the maximum acceleration to be set, ranged from 0 to 102400, with a unit of Pulse/ms^2. The detail refers to chapter 5. To make effective the newly set parameter requires to evoke the command GT_Update or GT_MltiUpdt first.

Applicable Card: GE-X00-PX

Command Evoking：refer to example.

GT_SetMaxVel

Command Prototype: short GT_SetMaxVel (double vel)

Description of Command: This command specifies maximum velocity, unit: pulse/ms. Maximum velocity is always specified during initializing system parameter.

Command Parameter: “vel” is the value of maximum velocity.

Applicable Card: GE-X00-SX

GT_SetMtrBias

Command Prototype: short GT_SetMtrBias(unsigned short axis, short bias)

Description of Command: This command sets the bias compensation of the control output of the specified axis servo filter.

Command Parameter: “axis” is the specified axis number; “bias” is the bias compensation to be set, ranged from -2^{15} to 2^{15}-1. To make effective the newly set parameter requires to voke the command GT_Update or GT_MltiUpdt first. This parameter is only effective for close loop control. The default value in the controller is 0.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking：refer to GT_SetKp.

GT_SetMtrCmd

Command Prototype: short GT_SetMtrCmd(unsigned short axis, short voltage)

Description of Command: This command sets the voltage output control value in the open loop control status.

Command Parameter: “axis” is the specified axis number; “voltage” is the motor output control value to be set, ranged from –32768 to 32767. The relation between voltage and setting value refers to GT_OpenLp.

Applicable Card: GE-X00-PV

Command Evoking：set the output voltage of axis 1 is 3V.

```
short OutputVoltage()
{
    short rtn;
    rtn = GT_OpenLp(1);                    //set axis 1 working in open
    loop mode
    if(0!=rtn) return rtn;
    return GT_SetMtrCmd(1, 3*3276);        //axis 1 output 3V voltage
}
```

GT_SetMtrLmt

Command Prototype: short GT_SetMtrLmt(unsigned short axis, unsigned short limit)

Description of Command: This command sets the voltage limit of the specified axis.

Command Parameter: “axis” is the specified axis number; “limit”is the limit value to be set, ranged from 0 to 32767. To make effective the newly set parameter requires to evoke the command GT_Update or GT_MltiUpdt first. This parameter is only effective for close loop control. The default value in the controller is 32767.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking: refer to GT_SetKp.

GT_SetPos

Command Prototype: short GT_SetPos(unsigned short axis, long pos)

Description of Command: This command sets the target position of the specified axis.

Command Parameter: “axis” is the specified axis number; “pos” is the target position, ranged from –1073741824 to 1073741823. The unit is pulse. To make effective the newly set parameter requires to evoke the command GT_Update or GT_MltiUpdt first.

Applicable Card: GE-X00-PX

Command Evoking：refer to example 5-1, 5-2 and 5-3.

GT_SetPosErr

Command Prototype: short GT_SetPosErr(unsigned short axis, unsigned short error)

Description of Command: This command sets the position error limit of the specified axis. To make effective the newly set parameter requires to evoke the command GT_Update or GT_MltiUpdt first. The detail refers to chapter 8.

Command Parameter: “axis” is the specified axis number; “error” is the position error limit

of the specified axis, ranged from 0 to 32767.

Applicable Card: GE-X00-PV，GE-X00-SV

Command Evoking：refer to GT_SetKp.

GT_SetSpindleVel

Command Prototype: short GT_SetSpindleVel(short vel)

Description of Command: This command specifies speed of main spindle rotating and make spindle to rotate as specified speed.

Command Parameter: “vel” is the speed value of main spindle rotating, ranged from –32768 to 32767, corresponding to analog output ranged from -10V to 10V

Applicable Card: GE-X00-SV

Command Evoking: refer to example 10-6.

GT_SetStpAcc

Command Prototype: short GT_SetStpAcc (double acc)

Description of Command: This command specifies stop acceleration, unit: pulse/ms^2. If some abnormal situation has happened during motion, like triggering limit switch or drive alarm, GE-X00-SX should stop motion by specified stop acceleration.

Command Parameter: “acc” is specified value of stop acceleration, experience value: double or treble value of normal acceleration.

Applicable Card: GE-X00-SX

GT_SetStrtVel

Command Prototype: short GT_SetStrtVel (double vel)

Description of Command: This command specifies start velocity, unit: pulse/ms. Start velocity is always specified during initializing system parameter. Default value of start velocity in motion controller is 500HZ.

Command Parameter: “vel” is the specified value of start velocity.

Applicable Card: GE-X00-SX

GT_SetSynAcc

Command Prototype: short GT_SetSynAcc(double acc)

Description of Command: This command specifies path acceleration, unit: pulse/ms^2. If and only if GE-X00-SX is in single-segment path motion, this command can be called.

Command Parameter: “acc” is the specified value of path acceleration.

Applicable Card: GE-X00-SX

GT_SetSynVel

Command Prototype: short GT_SetSynVel(double vel)

Description of Command: This command specifies path velocity (feed speed), unit: pulse/ms. Specified value decides all path's velocity followed (path velocity in single segment path motion mode and in multi-segment path continuous motion mode are different) till recalling this command.

Command Parameter: "vel" is the specified value of path velocity.

Applicable Card: GE-X00-SX

GT_SetVel

Command Prototype: short GT_SetVel(unsigned short axis, double vel)

Description of Command: This command sets the target velocity parameter of the specified axis.

Command Parameter: "axis" is the specified axis number; Vel is the target velocity value to be set. In the T-curve and S-curve modes, the velocity ranges from 0 to 40957. In the velocity control mode, the velocity ranges from –40958 to 40957. To make effective the newly set parameter requires to evoke the command GT_Update or GT_MltiUpdt first.

Applicable Card: GE-X00-PX

Command Evoking: Refer to example 5-1, 5-2 and 5-3.

GT_SmthStp

Command Prototype: short GT_SmthStp(unsigned short axis)

Description of Command: This command is to stop the motion of the specified axis. The difference from GT_AbptStp is that, GT_SmthStp will decelerate until stop the motion of current axis at the acceleration parameter set, rather than that GT_AbptStp stops the motion abruptly. This command can only be effective when being used together with the command GT_Update or GT_MltiUpdt. GT_SmthStp is used in the three motion control modes for the control axis, except the electrical gear mode.

Applicable Card: GE-X00-PX

Command Evoking: refer to example 7-1 and 7-2.

GT_StepDir

Command Prototype: short GT_StepDir(unsigned short axis)

Description of Command: If the control output mode of the specified axis is the pulse control mode, evoking this command can set the pulse output method as "Pulse + Direction".

Command Parameter: “axis” is the specified axis number.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking：If EXI0 is high level, the control output mode of axis 1 is Pulse + Direction. Or not, it is Positive Negative Pulse.

```
short PulseConfig()
{
    short rtn;
    unsigned short exInpt;
    rtn = GT_ExInpt(&exInpt);          //read general digital input
    if(0!=rtn) return rtn;
    if(exInpt&1)
    {
        return GT_StepDir(1);          //set control output mode of axis 1 is Pulse + Direction
    }
    else
    {
        return GT_StepPulse(1);        //set control output mode of axis 1 is Positive Negative Pulse
    }
}
```

GT_StepPulse

Command Prototype: short GT_StepPulse(unsigned short axis)

Description of Command: If the control output mode of the specified axis is the pulse output mode, evoking this command can set the pulse output mode as “Positive and Negative Pulse”.

Command Parameter: “axis” is the specified axis number.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: refer to GT_StepDir.

GT_StpMtn

Command Prototype: short GT_StpMtn (void)

Description of Command: This command stops the interpolation motion according to specified acceleration.

During multi-segment continuous path motion, if this command is called successfully, the buffer will be closed immediately. Just like calling the command GT_EndList. After motion stops, GE-X00-SX will record break point information. So that once user restores motion, GE-X00-SX should return to break point precisely to continue the following path motion described

in buffer. While during single-segment path motion, after motion stops, the information of current motion will be abandoned.

If user stops motion during executing GT_MvXY or GT_MvXYZ, maybe the multi-segment continuous path motion can not be restored correctly

Applicable Card: GE-X00-SX

GT_StrtList

Command Prototype: short GT_StrtList (void)

Description of Command: This command Open and clear the buffer. This command is valid when the buffer is closed and no multi-segment continuous path motion.

Applicable Card: GE-X00-SX

Command Evoking: refer to the description of the command GT_LnXYZ.

GT_StrtMtn

Command Prototype: short GT_StrtMtn (void)

Description of Command: This command starts multi-segment continuous path motion. Before calling the command, user must confirm that there are motion path commands in buffer which can be executed. After this command is called, GE-X00-SX will execute the commands in buffer in order.

Applicable Card: GE-X00-SX

Command Evoking: refer to the description of the command GT_LnXYZ.

GT_SwitchtoCardNo

Command Prototype: short GT_ SwitchtoCardNo (unsigned short number)

Description of Command: This command set the specified card as the current card. If user uses several control cards, this command can be used to specify the current control card. After the execution of this command succeeds, all the followed GT commands can only operate on the current card.

Command Parameter: “number” is the specified card number which will be set as the current card, ranged from 0 to 15.

Command Return: 0 means success, –1 means failure.

In a multiple-card system, each control card will be allocated with a card number (0-15) when the operating system starts. This card will keep effective before the system is restarted to differentiate each control card. The number allocation principle follows the PNP principle that, the first card recognized by the system will be the card 0 forever. Therefore, if there is no change in hardware configuration, the numbers allocated by the system each time will be consistent.

Applicable Card: GE-X00-SX

Command Evoking: refer to GT_GetCurrentCardNo.

GT_Update

Command Prototype: short GT_Update(unsigned short axis)

Description of Command: the parameters of the specified axis become effective after the command GT_Update is evoked.

Command Parameter：“axis” is the specified axis number.

Applicable Card: GE-X00-PX，GE-X00-SV

Command Evoking: refer to example 5-1 and 5-2.

GT_ZeroPos

Command Prototype: short GT_ZeroPos(unsigned short axis)

Description of Command: Evoke this command to set the actual position register and target position and the planned position register of the current control axis to 0. This command is only effective when the current axis stops. Otherwise, it will be considered as an illegal command and generates error symbol 1.

Command Parameter: “axis” is the specified axis number.

Applicable Card: GE-X00-PX，GE-X00-SX

Command Evoking: refer to the example 7-1 and 7-2.

Googoltech (Shenzhen) Co., Ltd.

Add.: W211, 2/F, West Building, SZHK Industry, Education and Research Base, South Area, High-tech Industry Park, Shenzhen

Tel.: (0755) 2697-0823, 2697-0817, 2697-0824

Fax: (0755) 2697-0846

E-mail: support@googoltech.com

Web: http://www.googoltech.com.cn/

Googol Technology (HK) Ltd

Add.: Room 3639, Annex Building
Hong Kong University of Science and Technology
Hong Kong

Tel: (852) 2358-1033

Fax: (852)2719-8399

E-mail: sales@googoltech.com

Web: http://www.googoltech.com/