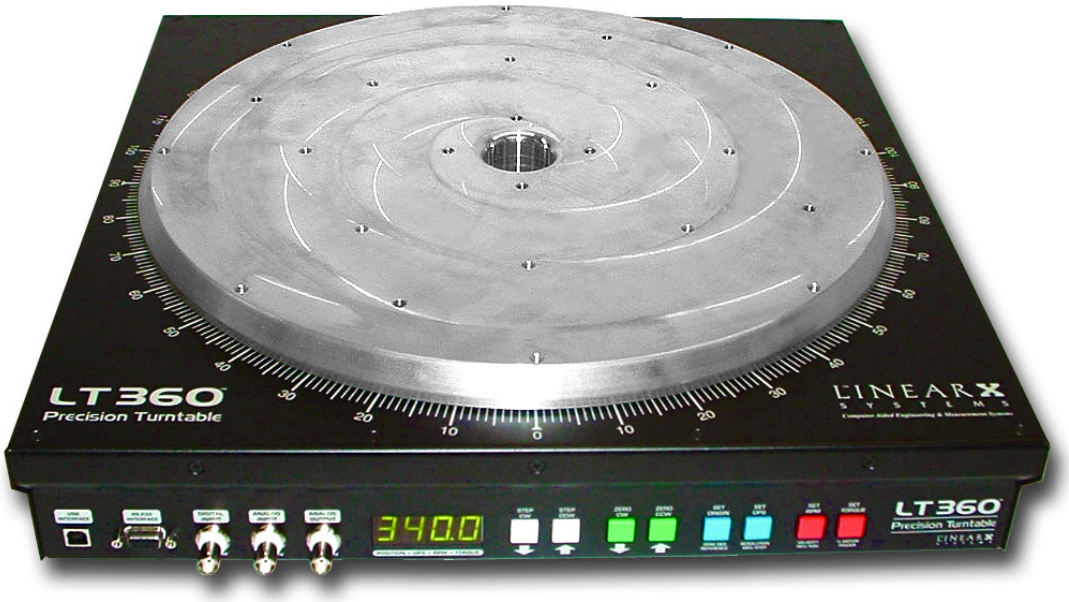
LT360™

Precision Turntable



DLL Programming Manual

LINEARX™
SYSTEMS

LT360 Precision Turntable
DLL Programming Manual

LT360LIB.DLL Win32 OS

Tel: (503) 612-9565
Fax: (503) 612-9344

Printed in the United States of America.
March 30, 2006.

This document was produced on a Pentium-4 / 2GHz PC with Win2K using Adobe PageMaker 7.0, Adobe Illustrator 10.0, Adobe PhotoShop 7.0, MathType 4.0 for mathematics typography, and SnagIt 5.2 for screen captures. Final masters were produced using an Xerox Docutech image setter. Help files were composed and compiled using Windows Help Designer 3.1.

License Agreement and Limited Warranty

Carefully read all of the following terms and conditions of this agreement before opening and using the contents of this package. The opening of this package indicates your acceptance of the terms and conditions of this license agreement. If you are not willing to accept the terms and conditions of this agreement, then you should return the entire product, with the package seal unbroken, to the place of purchase for a full refund of the purchase price.

■ Copyright Ownership

Both the program and the documentation are protected under applicable copyright laws. LinearX is the holder of this copyright. Your right to use the program and the documentation are limited to the terms and conditions described herein. Use of the software unless pursuant to the terms and conditions of this license, or as otherwise authorized by law, is an infringement of the copyright.

■ Limited Non-Exclusive License

You may: (a) use the enclosed program on a single computer, (b) physically transfer the program from one computer to another provided that the program is used on only one computer at a time, and that you remove any copies of the program from the computer from which the program is being transferred, (c) make copies of the program solely for backup or archival purposes. You must reproduce and include the copyright notice and label any backup copy.

You may not: (a) distribute copies of the program or the documentation to others, (b) lease, rent, grant sublicenses, or other rights to the program, (c) provide use of the program in a computer service business, network, time-sharing multiple CPU or multiple users arrangement without the prior written consent of LinearX, (d) translate or otherwise alter the program or related documentation without the prior written consent of LinearX.

■ Terms

Your license to use the program and the documentation will automatically terminate if you fail to comply with the terms of this agreement. Your license terminates in the event that you receive a license for an updated version of the product that replaces this product. If a license expiration date is printed on your documentation, or provided through other means such as a time limited electronic or software key, your license expires on the day as shown in the documentation, or on the day that the electronic or software key expires. If this license is terminated you agree to destroy all copies of the program and documentation.

■ Limited Warranty

LinearX warrants to the original licensee that the disk(s) and or electronic key(s) on which the program is recorded will be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of purchase as evidenced by a copy of your receipt. If failure of the product components has resulted from accident, abuse, or misapplication of the product, then LinearX or third party licensors shall have no responsibility to replace the disk(s) or key(s) under this limited warranty.

This limited warranty and right of replacement is in lieu of, and you hereby waive, any and all other warranties, both expressed and implied, including but not limited to warranties of merchantability and fitness for a particular purpose. The liability of LinearX or third party licensors pursuant to this limited warranty shall be limited to the replacement of the defective disk(s) or key(s), and in no event shall LinearX or third party licensors be liable for incidental, indirect, punitive, or consequential damages, including but not limited to loss of use, loss of profits, loss of data or data being rendered inaccurate, or losses sustained by third parties even if LinearX or third party licensors have been advised of the possibility of such damages. This warranty gives you specific legal rights which may vary from state to state. Some states do not allow the limitation or exclusion of liability for consequential damages, so the above limitation may not apply to you.

In addition to the foregoing, you should recognize that all complex software systems and their documentation contain errors and omissions. LinearX, its distributors, and dealers shall not be responsible under any circumstances for providing information on or corrections to errors and omissions discovered at any time in the product, whether or not they are aware of the errors or omissions. LinearX does not recommend the use of this product in applications in which errors or omissions could result in loss of life, injury, or other significant loss.

This license agreement shall be governed by the laws of the state of Oregon and shall inure to the benefit of LinearX, its successors, administrators, heirs and assigns or third party licensors.

■ United States Federal Government Restrictions

If this software is acquired by or on behalf of the U.S. Federal government or its agencies, this provision applies. Use, duplication, or disclosure of this software is subject to restrictions set forth in the appropriate FAR 52.227-19 and DFAR 252.227-7013 documents, as applicable. The software is "commercial computer software" and is licensed only with "Restricted Rights". Other Federal restrictions may also apply.

LinearX Systems Inc.
9500 SW Tualatin-Sherwood Rd.
Tualatin, OR 97062-8586 USA

TEL:(503) 612-9565 FAX:(503) 612-9344 WEB: www.linearx.com

Technical Support

LinearX provides detailed printed manuals and on-line help within the program as the primary source for user information and assistance regarding the use of this product. If these sources do not contain the answers to your questions, contact LinearX via any of the following methods:

Internet Forums: **www.linearx.com/forums**
Internet Email: **support@linearx.com**
Internet Web: **www.linearx.com**
Fax: **(503) 612-9344**
Tel: **(503) 612-9565**

Technical support is free and unlimited at this time, however we reserve the right to charge for this service in the future as conditions, overhead, and support personnel requirements dictate.

When contacting us regarding a technical support issue, PLEASE follow these steps to aid us in understanding and solving your problem:

(1) If your question involves specific details or parameters unique to your project and problem, please include a copy of your design files with the necessary data so that we can reproduce your problem. This is only possible if you are communicating via an electronic means such as Email or uploading files directly to our web site.

(2) If the issue regards error messages from the program, please include an exact description of the error message and/or address information that the program reports.

(3) If there are specific steps involved to reproduce the issue, please note these exact steps required so that we can reproduce the problem.

Note: Technical support *does not* include programming assistance. It is assumed that the reader has sufficient experience and knowledge to incorporate the DLL into their own application.

Technical support hours are: Monday-Friday 9:00AM to 5:00PM Pacific Standard Time.

Contents

Chapter 1: Overview 7

1.1 Overview of LT360LIB.DLL 7
1.2 Header / Include Files 8
1.3 Opening and Closing a Link 9
1.4 Error Messages 11
1.5 Using LT360 Commands 16

Chapter 2: Command Reference 17

2.1 Goto CCW 17
2.2 Goto CW 18
2.3 Step CCW 19
2.4 Step CW 20
2.5 Set Smart Torque 21
2.6 Set Baud Rate 22
2.7 Set Origin 23
2.8 Set Pulse Direction 24
2.9 Set Pulse Edge 25
2.10 Set Step Size 26
2.11 Set Velocity 27
2.12 Set Torque 28
2.13 Set Acceleration Function 29
2.14 Set Name 31
2.15 Set Pulse Input 32
2.16 Set Analog Input 33
2.17 Set Display Polarity 34
2.18 Set Input Polarity 35
2.19 Set Output Polarity 36
2.20 Set Move Abort 37
2.21 Set Motor Home Check 38
2.22 Set Output Mode 39
2.23 Get Name 41
2.24 Get Title 42
2.25 Get Firmware Version 43
2.26 Get Firmware Date 44
2.27 Get Production Date 45
2.28 Get Calibration Date 46

Contents

2.29 Get Calibration Due 47
2.30 Get Serial Number 48
2.31 Get Baud Rate 49
2.32 Get Position 50
2.33 Get Pulse Direction 51
2.34 Get Pulse Edge 52
2.35 Get Step Size 53
2.36 Get Velocity 54
2.37 Get Torque 55
2.38 Get Acceleration Function 56
2.39 Get Moving 58
2.40 Get Pulse Input 59
2.41 Get Analog Input 60
2.42 Get Smart Torque 61
2.43 Get Display Polarity 62
2.44 Get Input Polarity 63
2.45 Get Output Polarity 64
2.46 Get Motor Home Check 65
2.47 Get Revision Code 66
2.48 Get Output Mode 67

1.1 Overview of LT360LIB.DLL

A DLL (Dynamic Link Library) is provided for use by user application programs to allow direct control of the LT360 Precision Turntable. The DLL supports the Win32 OS environments. For other operating system environments, the universal RS-232 programming method should be used.

The DLL supports both methods of linking to LT360 turntables either via serial RS-232 connections or the USB (Universal Serial Bus). Multiple LT360 units can be managed by the DLL simultaneously, from 1 to 128 units.

The DLL handles all of the management details necessary for easy serial communications over either the PC's com ports or USB. The application program needs only to place calls to the DLL through 4 simple functions.

The LT360LIB.DLL file should be placed in a folder where the Windows system can find it. Typically placing a copy in the application folder is common, or in the main Windows folder.

The DLL functions are 32 bit code and can be called by any Win32 application written in any language. An appropriate header file is all that is required for the exported functions for linking.

1.2 Header / Include Files

Two example header / include files for linking and calling the DLL functions are provided for C/C++ or Delphi/Pascal:

Header / Include Files

■ LT360LIB.h	C/C++
■ LT360LIB.pas	Delphi/Pascal

Other header files for different languages can be translated from these examples. The header files contain all of the prototype exported functions along with the command identifier constants used to make the various function calls to the LT360 turntables.

The header files may require modification depending on the particular behavior of the compiler/linker being used.

1.3 Opening and Closing a Link

The procedure for opening a link is very easy. First connect an LT360 to either a serial COM port or USB port on the computer. Then call one of two functions to open the link:

```
LONGINT LT360LIB_OpenLinkUSB(void);

LONGINT LT360LIB_OpenLinkCOM(LONGINT ComNum, LONGINT BaudRate);
```

■ **LT360LIB_OpenLinkUSB**
This call is used if the turntable is connected to the USB bus. There are no parameters. Since there may be multiple LT360 units attached to the USB bus, this function can be called more than once.

The function returns a *Handle* to each unit found, which is simply a numeric index. The calling program should maintain and store these handles. If no more LT360 units are found, then the handle returned is 0.

Note: The USB driver for the LT360 must be installed in the Win32 system.

■ **LT360LIB_OpenLinkCOM**
This call is used if the turntable is connected to a serial COM port. There are two parameters: the COM port number (ie. 1,2,3,4...) and the baud rate. The default baud rate shipped with the LT360 is 9600. You must specify the COM port number in which the LT360 is connected. The function may be called multiple times with *different* com port numbers for LT360 units attached to the different com ports.

The function returns a *Handle* to the unit found, which is simply a numeric index. The calling program should maintain and store these handles. If no LT360 unit is found, then the handle is 0.

Note: The 9 pin serial cable should be straight-through, not null modem.

■ **LT360LIB_CloseLink**
To close a link to an LT360, call this function with the handle of the unit. The return value is True if successful, and false if an error.

```
LONGBOOL LT360LIB_CloseLink(LONGINT Handle);
```

■ Example with USB link

The following code fragment demonstrates the sequence of calls for a USB link to a LT360 unit.

```
// Open link via USB, save returned handle
H360 = LT360LIB_OpenLinkUSB();
...
// Call the LT360 to set the Step Size to 1 degree
LT360LIB_CmdValue(H360,lt_SetStepSize,'1.0');

// Call the LT360 to do a step CCW
LT360LIB_CmdValue(H360,lt_Step_CCW,null);
...
// Close the link when done
LT360LIB_CloseLink(H360);
```

■ Example with COM link

The following code fragment demonstrates the sequence of calls for a RS-232 serial port link to a LT360 unit.

```
// Open link via serial port, COM3 at 9600 baud, save returned handle
H360 = LT360LIB_OpenLinkCOM(3,9600);
...
// Call the LT360 to set the Step Size to 1 degree
LT360LIB_CmdValue(H360,lt_SetStepSize,'1.0');

// Call the LT360 to do a step CCW
LT360LIB_CmdValue(H360,lt_Step_CCW,null);
...
// Close the link when done
LT360LIB_CloseLink(H360);
```

1.4 Error Messages

The LT360 can display a variety of error codes on the front panel display. The error message will be displayed over a 4 second interval with a format such as *Err0* or *Er13*.

The following list describes some of the error codes which can appear:

```
//-------------------------------------------------
// Show Error Display Codes
//-------------------------------------------------
//       0      -       Motor Home Position Error
//       1      -       RS232-OE Overrun Error
//       2      -       RS232-PE Parity Error
//       3      -       RS232-FE Framing Error
//       4      -       RS232-BI Break Int Error
//       5      -       RS232 Unknown Command
//       6      -       RS232 Invalid Parameter
//       7      -       RS232 Invalid Return
//       8      -       USB   Unknown Command
//       9      -       RS232 Command Timeout
//      10      -
//      11      -       Display Cntrlr Adr Failure
//      12      -       Switch Cntrlr Adr Failure
//      13      -       Torque Cntrlr Adr Failure
//      14      -       Output Cntrlr Adr Failure
//      15      -
//      16      -
//      17      -
//      18      -
//      19      -
//      20      -
//-------------------------------------------------
```

Most errors produced by the LT360 will be RS-232 command or parameter errors, especially if the user is programming via RS-232. The USB processing routines also rely on the RS-232 command decoders, so it is very possible to see RS-232 error codes while using the USB interface.

The basic LT360 serial RS-232 communication parameters are: 9600 baud, 8 data bits, no parity, 1 stop bit, and no flow control.

Error - 0 : Motor Home Position Error

The stepper motor produces a specific number steps for a given amount of platter rotation. The LT360 utilizes microstepping for additional precision control. One particular phase of the motor is designated as the *Home* position. For any given platter rotation, the motor should always end in its Home position. This can be checked by the firmware continuously, if the *MotorHomeChk* option is enabled. If the motor position is not in the Home phase at the end of a movement, this error will be shown.

In normal operation this error should never occur. If it does, the unit must be powered off, or the *MotorHomeChk* option disabled, to clear the error. It is possible for this error to be produced by noise in the stepper motor controller, or by the load exceeding the torque capability thereby causing the motor to skip steps.

If this error is shown repeatedly, either there is a problem internal in the LT360 or the torque load is too great for the drive system.

Error - 1 : RS232-OE Overrun Error

This error indicates problems with the serial port communications. It can be caused by differences in the communication parameters between the computer and the LT360. Make sure that the computer serial port setup is 9600 baud, 8 data bits, no parity, and 1 stop bit. If the LT360 baud rate has been changed, then the PC must be set the same.

Error - 2 : RS232-PE Parity Error

This error indicates problems with the serial port communications. It can be caused by differences in the communication parameters between the computer and the LT360. Since the LT360 does not use parity, this error is unlikely. Make sure that the computer serial port setup is 9600 baud, 8 data bits, no parity, and 1 stop bit. If the LT360 baud rate has been changed, then the PC must be set the same.

Error - 3 : RS232-FE Framing Error

This error indicates problems with the serial port communications. It can be caused by differences in the communication parameters between the computer and the LT360, or invalid parameters in the computer UART of the computer. Make sure that the computer serial port setup is 9600 baud, 8 data bits, no parity, and 1 stop bit. If the LT360 baud rate has been changed, then the PC must be set the same. You may also need to reboot the computer to clear and reinitialize the UART in the computer.

Error - 4 : RS232-BI Break Int Error

This error indicates problems with the serial port communications. It can be caused by differences in the communication parameters between the computer and the LT360. Make sure that the computer serial port setup is 9600 baud, 8 data bits, no parity, and 1 stop bit. If the LT360 baud rate has been changed, then the PC must be set the same.

Error - 5 : RS232 Unknown Command

This error indicates that the character string sent to the LT360 did not contain a valid command. This error will generally occur if you are writing your own RS-232 program, and there are bugs in your program. Check the command strings being sent to verify that they contain valid LT360 commands. Case is not important, but spaces in the right or wrong places can make a difference.

Error - 6 : RS232 Invalid Parameter

This error indicates that the LT360 was expecting a parameter value for this command, but did not find one in the command string. Some commands need parameters and some do not. This error will generally occur if you are writing your own RS-232 program, and there are bugs in your program. Check the command strings being sent to verify that they contain valid LT360 commands and parameters. Case is not important, but spaces in the right or wrong places can make a difference.

Error - 7 : RS232 Invalid Return

This error indicates that the LT360 was expecting to produce a return value for a Get command, but the return value was an empty string. This error will generally occur if you are writing your own RS-232 program, and there are bugs in your program. Check the command strings being sent to verify that they contain valid LT360 commands and/or parameters. Case is not important, but spaces in the right or wrong places can make a difference.

Error - 8 : USB Unknown Command

This error indicates that the LT360 did not understand the command index sent via USB. This error is very uncommon. It probably indicates faulty communication over USB, such as a cable being disconnected or dirty connections. Possibly some other unusual problem.

Error - 9 : RS232 Command Timeout

This error indicates that the LT360 received some characters, but never received a termination character [Null(0) or CR(13)] before the 10 second timeout expired. When the first character is sent to the LT360, a timer is started. If a termination character is not received within 10 seconds, then this error is produced and the command buffer is cleared. Receiving all of the characters for a command string should take very little time, on the order of micro or milli seconds, so if a termination charcater does not arrive within 10 seconds, something is wrong. The LT360 uses this timeout to clear the command buffer every 10 seconds if characters are received with no termination character. The termination character is mandatory to notify the LT360 that the entire command string has been sent and it can now process it.

Error - 11: Display Controller Address Failure

This error indicates that the CPU in the LT360 cannot communicate with the display controller. This error should never occur. If it does, a failure of a component in the LT360 is indicated.

Error - 12: Switch Controller Address Failure

This error indicates that the CPU in the LT360 cannot communicate with the switch controller. This error should never occur. If it does, a failure of a component in the LT360 is indicated.

Error - 13: Torque Controller Address Failure

This error indicates that the CPU in the LT360 cannot communicate with the stepper motor torque DAC controller. This error should never occur. If it does, a failure of a component in the LT360 is indicated.

Error - 14: Output Controller Address Failure

This error indicates that the CPU in the LT360 cannot communicate with the analog output DAC controller. This error should never occur. If it does, a failure of a component in the LT360 is indicated.

1.5 Using LT360 Commands

The commands which a typical user application needs can all be called using a single function:

```
LONGBOOL LT360LIB_CmdValue(LONGINT Handle, LONGINT CmdID, char *ParamValue);
```

This function has three parameters and returns a true/false result for success or failure. The first parameter is the *Handle* to the LT360 unit (returned from the OpenLink calls), the second parameter is the *Command Identifier*, and the third parameter is a *ParamValue* string (ASCIIZ string).

All user commands for the LT360 are sent using this function with various command identifiers. They are described in the header files, and on the following pages. The *ParamValue* string pointer is used both as an In/Out parameter to send data or receive data from the LT360.

For some *Set* commands that do not require a parameter, this value can be null/nil. For *Get* commands which return a value, the user must supply a string pointer to the user allocated character space. Generally 32 characters of space is adequate for all command value returns.

Commands fall into two categories: *Set* and *Get*. Set commands are Out commands which can send data to the LT360. Get commands are In commands which return data from the LT360. Some Set commands may or may not have a parameter value.

Numeric values sent as parameters to the LT360 must be converted into a string. They are also returned as a string.

If the command or data sent to an LT360 is incorrect, an error message will be displayed on the LT360 front panel. The function above will only reflect an error if there is a problem with the data transmission.

2.1 Goto CCW

- Class Set (out)
- Identifier lt_Goto_CCW
- Hex Value 0x1C
- Call LT360LIB_CmdValue(Handle, lt_Goto_CCW**, '**±*nnn.n*'**);**
- Parameter Position (degrees)

Description
This command causes the LT360 to move to the desired location specified by the parameter value using counter clockwise rotation. If the LT360 is already at this location, no movement occurs.

The parameter value should have tenths of a degree precision, and may be either unipolar (0..+360.0) or bipolar (0..±180.0).

The Goto 0 command can also be issued from the front panel of the unit, or the LR360 remote using the switches on the front panel.

2.2 Goto CW

- Class Set (out)
- Identifier lt_Goto_CW
- Hex Value 0x1D
- Call LT360LIB_CmdValue(Handle, lt_Goto_CW, '±*nnn.n*');
- Parameter Position (degrees)

Description
This command causes the LT360 to move to the desired location specified by the parameter value using clockwise rotation. If the LT360 is already at this location, no movement occurs.

The parameter value should have tenths of a degree precision, and may be either unipolar (0..+360.0) or bipolar (0..±180.0).

The Goto 0 command can also be issued from the front panel of the unit, or the LR360 remote using the switches on the front panel.

2.3 Step CCW

- Class Set (out)
- Identifier lt_Step_CCW
- Hex Value 0x1E
- Call LT360LIB_CmdValue(Handle, lt_Step_CCW, null);
- Parameter (none)

Description
This command causes the LT360 to move from its current position by the current step size using counter clockwise rotation. No parameter is required.

The step command can also be issued from the front panel of the unit, the LR360 remote, or by the TTL *Pulse* input on the front panel.

2.4 Step CW

- Class Set (out)
- Identifier lt_Step_CW
- Hex Value 0x1F
- Call LT360LIB_CmdValue(Handle, lt_Step_CW, null);
- Parameter (none)

Description
This command causes the LT360 to move from its current position by the current step size using clockwise rotation. No parameter is required.

The step command can also be issued from the front panel of the unit, the LR360 remote, or by the TTL *Pulse* input on the front panel.

The parameter value should have tenths of a degree precision, and may be either unipolar (0..360.0) or bipolar (0..±180.0).

2.5 Set Smart Torque

- Class Set (out)
- Identifier lt_SetSmartTorque
- Hex Value 0x20
- Call LT360LIB_CmdValue(Handle, lt_SetSmartTorque, 'xxx');
- Parameter 'OFF' or 'ON'

Description
This command controls the behavior of the stepper motor. When *SmartTorque* is set ON, the motor is powered down to half the driving power 2 seconds after each movement stops. Motor power is returned to full (as controlled by the Maximum Torque setting) when the motor starts again.

This feature takes advantage of the fact that the worm gear drive system employed by the LT360 is irreversible. Meaning, the platter is self-locking at its current position. The load cannot rotate the platter itself. Therefore, it is not necessary to maintain full power on the motor merely to hold its current position when the platter is not in movement.

The use of *SmartTorque* greatly reduces the thermal heating in the motor and driver circuitry, and produces a very efficient drive system. Under most conditions and use, this feature should always be kept ON.

If a particular application demands that the motor power be kept constant at all times, even when stationary (SmartTorque=OFF), then the maximum torque should be to 70% or less to prevent excessive heating. It is difficult to envision what kinds of applications would require this behavior, but the option is available if needed.

2.6 Set Baud Rate

- Class Set (out)
- Identifier lt_SetBaudRate
- Hex Value 0x21
- Call LT360LIB_CmdValue(Handle, lt_SetBaudRate, 'nnnnn');
- Parameter Baud rate, eg. '9600'

Description
This command sets the baud rate for RS-232 communications. Once the new baud rate is set, all further RS-232 communication with an LT360 must be at the new baud rate.

Legal baud rate values are:

9600
14400
19200
38400
57600

Note: Increasing the baud rate will not necessarily produce a direct effect on the speed of communication and commands with the LT360. The LT360 has a response time for commands which is dependent on many factors internally. The CPU must perform many other tasks so there can be a 1-25 mSec latency before the commands are processed regardless of the baud rate.

2.7 Set Origin

- Class Set (out)
- Identifier lt_SetOrigin
- Hex Value 0x22
- Call LT360LIB_CmdValue(Handle, lt_SetOrigin, null);
- Parameter (none)

Description
This command resets the rotational origin to that of the current platter position. No parameters are required, and the platter does not rotate. All further goto commands will be referenced to this new origin (0.0).

2.8 Set Pulse Direction

- Class Set (out)
- Identifier lt_SetPulseDir
- Hex Value 0x23
- Call LT360LIB_CmdValue(Handle, lt_SetPulseDir, 'xxx');
- Parameter 'CCW' or 'CW'

Description
This command controls the behavior to a TTL step pulse at the *Pulse* input BNC connector. The parameter string can be either CCW or CW, meaning counter clockwise or clockwise.

When a pulse arrives at the *Pulse* BNC input, the platter will rotate either CCW or CW as defined by this setting. The size of the step is controlled by the *Set StepSize* command.

Note: The TTL pulse should be a minimum of 10uS, or longer. The triggering edge can be controlled by the Set PulseEdge command.

2.9 Set Pulse Edge

- Class Set (out)
- Identifier lt_SetPulseEdge
- Hex Value 0x24
- Call LT360LIB_CmdValue(Handle, lt_SetPulseEdge, 'xxxx');
- Parameter 'RISE' or 'FALL'

Description
This command controls the behavior to a TTL step pulse at the *Pulse* input BNC connector. The parameter string can be either RISE or FALL, meaning triggering occurs on the rising or falling edge of the pulse.

When a pulse arrives at the *Pulse* BNC input, the platter will rotate either at the rising or falling edge as defined by this setting. The size of the step is controlled by the *Set StepSize* command.

Note: The TTL pulse should be a minimum of 10uS, or longer. The direction can be controlled by the Set PulseDir command.

2.10 Set Step Size

- Class Set (out)
- Identifier lt_SetStepSize
- Hex Value 0x25
- Call LT360LIB_CmdValue(Handle, lt_SetStepSize, 'nn.n');
- Parameter Step Size (degrees)

Description
This command sets the angular step size by which the platter will rotate to either a step command or TTL pulse. The parameter string must contain a numeric value such as '15.0' with tenth of a degree precision.

The minimum step size is 0.1 degrees. Step size is always positive. Direction is controlled by the CCW/CW commands listed else where.

2.11 Set Velocity

- Class Set (out)
- Identifier lt_SetVelocity
- Hex Value 0x26
- Call LT360LIB_CmdValue(Handle, lt_SetVelocity, 'n.nn');
- Parameter Velocity (RPM)

Description
This command sets the angular velocity by which the platter will rotate. The value is set in RPM (revolutions per minute). The allowable range for the parameter is 0.01 to 3.00 RPM.

The parameter string must contain a numeric value such as '1.00' with hundreths precision.

2.12 Set Torque

- Class Set (out)
- Identifier lt_SetTorque
- Hex Value 0x27
- Call LT360LIB_CmdValue(Handle, lt_SetTorque, 'nnn.n');
- Parameter Torque (%)

Description
This command sets the stepper motor torque for the platter rotation. The value is set in percent of maximum torque. The allowable range for the parameter is 10.0 to 100.0 %.

The parameter string must contain a numeric value such as '100.0' with tenths of a degree precision.

For most typical applications your will want the torque set to 100%. If you intend to have the platter in constant motion, than you may wish to set the torque to a lower value such as 70%. This will reduce heating in the motor and drive circuitry.

The *SmartTorque* feature of the LT360 automatically reduces the motor torque to 1/2 its normal rotational power when it is not moving. Thus providing the advantages of maximum torque when in motion and reduced power consumption when stationary.

2.13 Set Acceleration Function

- Class Set (out)
- Identifier lt_SetAccelFunc
- Hex Value 0x28
- Call LT360LIB_CmdValue(Handle, lt_SetAccelFunc, 'n');
- Parameter Function Number 0..4

Description
This command sets the acceleration function which the LT360 will use during step or goto commands. The acceleration profile is an important feature of the LT360 which enables heavy loads to be moved with minimum disturbance.

The parameter string must contain a numeric value such as '1' with allowable values from 0..4. There are five acceleration functions numbered 0,1,2,3,4. The names of these profiles are:

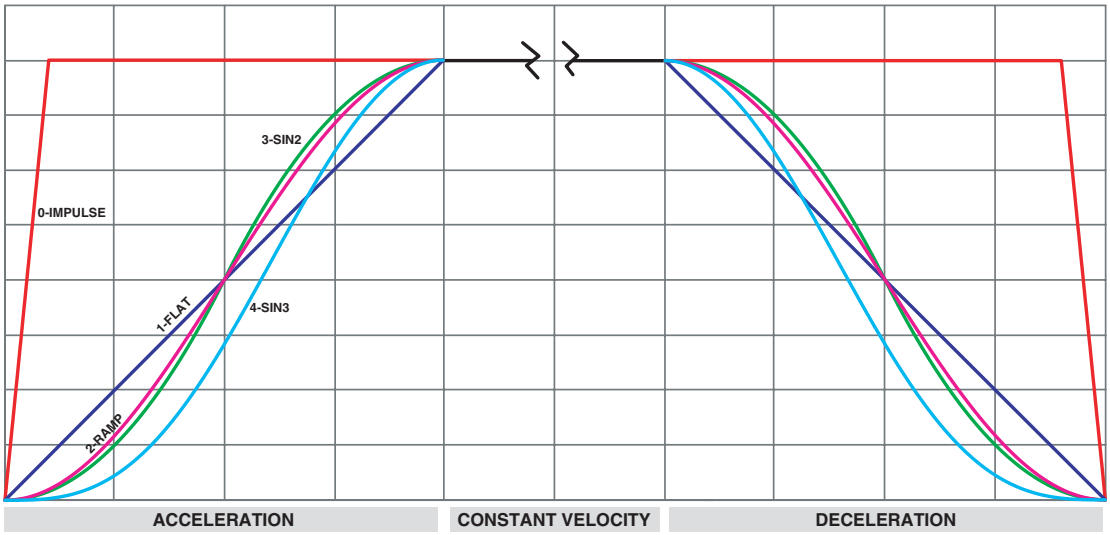
0 - Impulse
1 - Flat
2 - Ramp
3 - Sin2
4 - Sin3

For most typical applications acceleration function 1 - FLAT will provide excellent performance. Because the LT360 controls the acceleration and deceleration so precisely, very heavy loads can be rotated with ease.

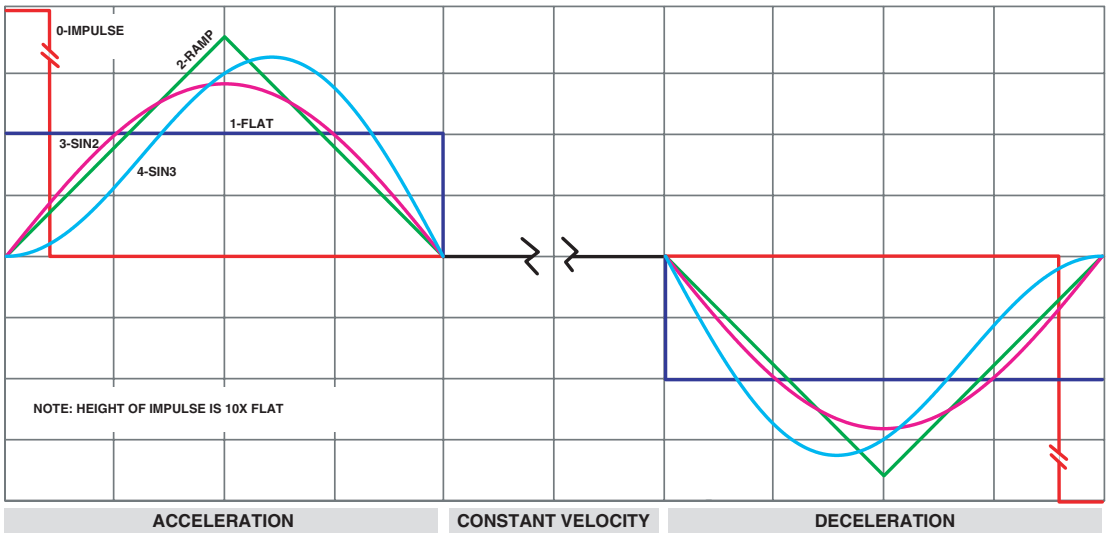
The 0 - IMPULSE function provides the fastest rise time from zero to constant velocity. However it also greatly reduces the maximum load which the LT360 can rotate, due to its higher acceleration. The acceleration of this function is about 10X that of the others.

The graphs on the following page show the velocity and acceleration profile curves for the five different LT360 functions.

LT360 VELOCITY PROFILES



LT360 ACCELERATION PROFILES



2.14 Set Name

- Class Set (out)
- Identifier lt_SetName
- Hex Value 0x29
- Call LT360LIB_CmdValue(Handle, lt_SetName, 'xx...xx');
- Parameter Name

Description
This command sets the Name for the LT360 unit. This is an arbitrary user name which can be assigned to the unit. This can be useful for applications which utilize multiple units to distinguish between the units, for example Horz, Vert, etc. This name is stored in the LT360 and is non volatile.

The Name parameter string has a maximum length of 21 characters.

2.15 Set Pulse Input

- Class Set (out)
- Identifier lt_SetPulseInput
- Hex Value 0x32
- Call LT360LIB_CmdValue(Handle, lt_SetPulseInput, 'xxx');
- Parameter 'OFF' or 'ON'

Description
This command enables the TTL Pulse Input feature. When *PulseInput* is set ON, the LT360 will respond to TTL pulses at the BNC connector. When it is set OFF, it will ignore any pulses.

Each pulse will trigger a step. The width of the pulse should be at least 10uSec, and the triggering can be set on the rising or falling edge. Any pulses which occur while the LT360 is moving will be ignored.

If you are not using the TTL pulse input, it is probably best to keep this feature OFF to prevent false triggering due to any noise pickup at the open connector.

2.16 Set Analog Input

- Class Set (out)
- Identifier lt_SetAnalogInput
- Hex Value 0x33
- Call LT360LIB_CmdValue(Handle, lt_SetAnalogInput, 'xxx');
- Parameter 'OFF' or 'ON'

Description
This command enables the Analog Input feature. When *AnalogInput* is set ON, the LT360 will respond to the DC voltage at the analog input BNC connector. When it is set OFF, it will ignore that input.

The LT360 has a scale factor of 10mV/deg for the analog input. The range can be either unipolar (0 ... +3.600V) or bipolar (0 ... ±1.800V). The input impedance of the analog input is 1M Ohm. Although the input is an unbalanced BNC connector, the actually circuitry is differential. This helps to reject ground noise.

The input source must be very stable and of low noise. The LT360 has a resolution of 0.1 degrees and therefore responds to changes of 1mV at the analog input. When this input is enabled, the LT360 will sample the input and move to the location as indicated by the voltage. After that move is completed, it then samples the analog input again and moves to the new position. Monitoring of the analog input is real time.

If you are not using the analog input, this input *must* be set OFF. Otherwise the LT360 will be controlled by any noise received at the input.

2.17 Set Display Polarity

- Class Set (out)
- Identifier lt_SetDisplayPolarity
- Hex Value 0x34
- Call LT360LIB_CmdValue(Handle, lt_SetDisplayPolarity, 'xxxxxxx');
- Parameter 'UNIPOLAR' or 'BIPOLAR'

Description
This command changes how the LT360 represents the degree position on the front display of the chassis, and in the Win32 software. When *DisplayPolarity* is set to UNIPOLAR, the LT360 will display the position in the range of 0 to 360 degrees. All position values are positive. When it is set to BIPOLAR, the range will be 0 to ±180 degrees. Both positive and negative values will be shown.

2.18 Set Input Polarity

- Class Set (out)
- Identifier lt_SetInputPolarity
- Hex Value 0x35
- Call LT360LIB_CmdValue(Handle, lt_SetInputPolarity, 'xxxxxxx');
- Parameter 'UNIPOLAR' or 'BIPOLAR'

Description
This command changes how the LT360 interprets the DC voltage received at the Analog Input connector. When *InputPolarity* is set to UNIPOLAR, the LT360 will accept voltages in the range of 0.000 to +3.600 volts. All position values are positive. When it is set to BIPOLAR, the range will be 0.000 to ±1.800 volts. Both positive and negative values are used.

Voltages higher than the allowable range for the selected mode are invalid.

The LT360 has a scale factor of 10mV/deg for the analog input. The range can be either unipolar (0 ... +3.600V) or bipolar (0 ... ±1.800V). The input impedance of the analog input is 1M Ohm. Although the input is an unbalanced BNC connector, the actually circuitry is differential. This helps to reject ground noise.

The input source must be very stable and of low noise. The LT360 has a resolution of 0.1 degrees and therefore responds to changes of 1mV at the analog input. When this input is enabled, the LT360 will sample the input and move to the location as indicated by the voltage. After that move is completed, it then samples the analog input again and moves to the new position. Monitoring of the analog input is real time.

If you are not using the analog input, this input *must* be set OFF. Otherwise the LT360 will be controlled by any noise received at the input.

2.19 Set Output Polarity

- Class Set (out)
- Identifier lt_SetOutputPolarity
- Hex Value 0x36
- Call LT360LIB_CmdValue(Handle, lt_SetOutputPolarity, 'xxxxxxx');
- Parameter 'UNIPOLAR' or 'BIPOLAR'

Description
This command changes how the LT360 generates the DC voltage output at the Analog Output connector. When *OutputPolarity* is set to UNIPOLAR, the LT360 will produce voltages in the range of 0.000 to +3.600 volts. All position values are positive. When it is set to BIPOLAR, the range will be 0.000 to ±1.800 volts. Both positive and negative values are produced.

The LT360 has a scale factor of 10mV/deg for the analog output. The range can be either unipolar (0 ... +3.600V) or bipolar (0 ... ±1.800V). The output impedance of the analog output is <200 Ohms. Although the output is an unbalanced BNC connector, it is highly recommended that the input on the other device be differential to reject ground noise. Since the smallest resolution of the LT360 is 0.1 degree, the equivalent resolution in the output is 1mV. The input impedance of the other device should be at least 100K Ohm to prevent loading errors.

2.20 Set Move Abort

- Class Set (out)
- Identifier lt_SetMoveAbort
- Hex Value 0x38
- Call LT360LIB_CmdValue(Handle, lt_SetMoveAbort, null);
- Parameter (none)

Description
This command can be used to terminate the rotation in progress. The LT360 will stop rotating as quickly as possible at any current location. This command can be used as an emergency stop feature if needed.

2.21 Set Motor Home Check

- Class Set (out)
- Identifier lt_SetMotorHomeChk
- Hex Value 0x39
- Call LT360LIB_CmdValue(Handle, lt_SetMotorHomeChk, 'xxx');
- Parameter 'OFF' or 'ON'

Description
This command controls checking of the stepper motor synchronization . When *MotorHomeChk* is set ON, the firmware checks that the motor returns to its home position after each movement. If disabled, no checking is performed. If checking is enabled and the motor does not end its operation in the home position, an error message is shown on the front panel display.

The stepper motor produces a specific number steps for a given amount of platter rotation. The LT360 utilizes microstepping for additional precision control. One particular phase of the motor is designated as the *Home* position. For any given platter rotation, the motor should always end in its Home position.

In normal operation this error should never occur. If it does, the unit must be powered off to clear the error, or this feature disabled. It is possible for this error to be produced by noise in the stepper motor controller, or by the load exceeding the torque capability thereby causing the motor to skip steps.

If this error is shown repeatedly, either there is a problem internal in the LT360 or the torque load is too great for the drive system.

2.22 Set Output Mode

- Class Set (out)
- Identifier lt_SetOutputMode
- Hex Value 0x3B
- Call LT360LIB_CmdValue(Handle, lt_SetOutputMode, 'xxxx');
- Parameter 'CONT' or 'START' or 'STOP'

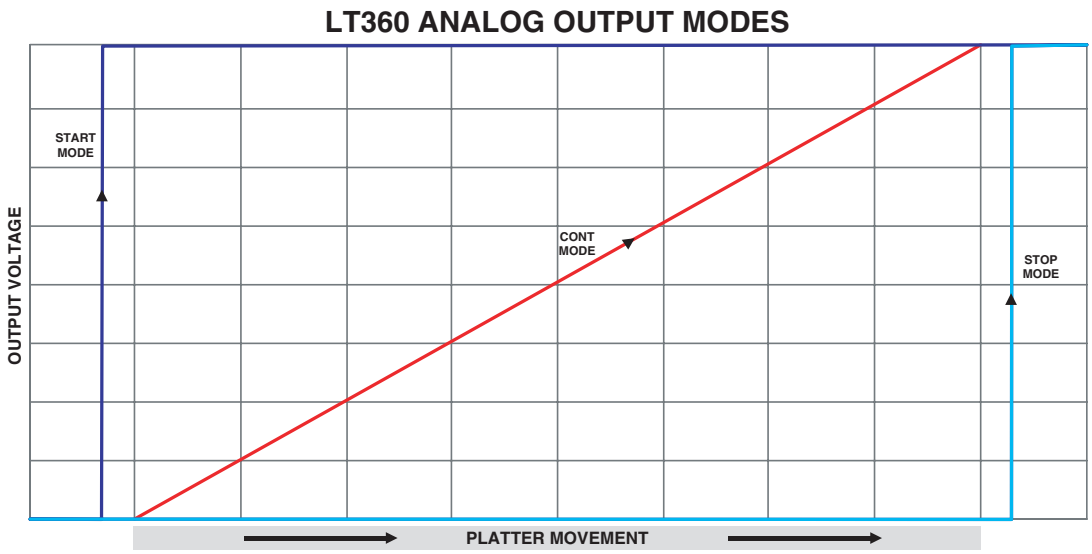
Description
This command selects the operating mode for the Analog Output. The parameter string value is either CONT or START or STOP.

When *CONT* is the mode, the Analog Output will continuously follow the platter position as it is moving. This is a real time output mode where the analog output voltage always represents the current platter position.

When *START* is the mode, the Analog Output will only produce a single step change output voltage representing the final destination position at the start of the movement. For example, if the current position is 0.0 degrees and the destination is 100.0 degrees, the Analog Output voltage will change from 0.000V to 1.000V at the start of the movement.

When *STOP* is the mode, the Analog Output will only produce a single step change output voltage representing the final destination position after the movement stops. For example, if the current position is 0.0 degrees and the destination is 100.0 degrees, the Analog Output voltage will remain at 0.000V during the movement and then change to 1.000V when the movement stops. The final output voltage changes 2 seconds after the platter movemet stops.

If the movement is aborted before it reaches the destination, the output voltage will be updated to the correct final value in all modes.


LT360 ANALOG OUTPUT MODES
START MODE
CONT MODE
STOP MODE
OUTPUT VOLTAGE
PLATTER MOVEMENT

2.23 Get Name

- Class Get (in)
- Identifier lt_GetName
- Hex Value 0x91
- Call LT360LIB_CmdValue(Handle, lt_GetName, RtnStr);
- Returns *Name* string

Description
This command returns the Name for the LT360 unit. This is an arbitrary user name which can be assigned to the unit. This can be useful for applications which utilize multiple units to distinguish between the units, for example Horz, Vert, etc. This name is stored in the LT360 and is non volatile.

The Name parameter string has a maximum length of 21 characters.

2.24 Get Title

- Class Get (in)
- Identifier lt_GetTitle
- Hex Value 0x92
- Call LT360LIB_CmdValue(Handle, lt_GetTitle, RtnStr);
- Returns *'LT360 Precision Turntable'*

Description
This command returns the Title of the LT360 unit, which always is *LT360 Precision Turntable*. This command can be useful if you are writing your own RS-232 program to control the LT360. This command can be used as a test to verify that you are communicating with an LT360 unit.

2.25 Get Firmware Version

- Class Get (in)
- Identifier lt_GetVerCode
- Hex Value 0x95
- Call LT360LIB_CmdValue(Handle, lt_GetVerCode, RtnStr);
- Returns *'n.nn'*

Description
This command returns the version code of the firmware within the LT360. For example: *1.50* The numeric value is returned as a string.

2.26 Get Firmware Date

- Class Get (in)
- Identifier lt_GetVerDate
- Hex Value 0x96
- Call LT360LIB_CmdValue(Handle, lt_GetVerDate, RtnStr);
- Returns *'mmm-dd-yyyy'*

Description
This command returns the version date of the firmware within the LT360. For example: *JAN-01-2006* The date value is returned as a string.

2.27 Get Production Date

- Class Get (in)
- Identifier lt_GetProdDate
- Hex Value 0x97
- Call LT360LIB_CmdValue(Handle, lt_GetProdDate, RtnStr);
- Returns *'mmm-dd-yyyy'*

Description
This command returns the date when the LT360 was manufactured. For example: *JAN-01-2006* The date value is returned as a string.

2.28 Get Calibration Date

- Class Get (in)
- Identifier lt_GetCalDate
- Hex Value 0x98
- Call LT360LIB_CmdValue(Handle, lt_GetCalDate, RtnStr);
- Returns *'mmm-dd-yyyy'*

Description
This command returns the date when the LT360 was last calibrated. For example: *JAN-01-2006* The date value is returned as a string.

Calibration is performed using the Win32 application software. Calibration is performed originally at the factory, and in most cases will be permanent and not be required again. It is possible for a customer to performed the calibration, but only if the proper equipment is available. The computer must have a GPIB bus, and two 6-1/2 digit DMMs are required themselves with accurate recent calibration. See the LT360 Manual for further details.

2.29 Get Calibration Due

- Class Get (in)
- Identifier lt_GetDueDate
- Hex Value 0x99
- Call LT360LIB_CmdValue(Handle, lt_GetDueDate, RtnStr);
- Returns *'mmm-dd-yyyy'*

Description
This command returns the date when the LT360 should be re-calibrated. For example: *JAN-01-2007* The date value is returned as a string.

Calibration is performed using the Win32 application software. Calibration is performed originally at the factory, and in most cases will be permanent and not be required again. However if your application requires NIST compliance, this is the date when the LT360 should be rechecked. It is possible for a customer or other lab to performed the calibration, but only if the proper equipment is available. The computer must have a GPIB bus, and two 6-1/2 digit DMMs are required themselves with accurate recent calibration. See the LT360 Manual for further details.

2.30 Get Serial Number

- Class Get (in)
- Identifier lt_GetSerialNum
- Hex Value 0x9A
- Call LT360LIB_CmdValue(Handle, lt_GetSerialNum, RtnStr);
- Returns *'nnnnnn'*

Description
This command returns the serial number of the LT360 unit. The number is returned as a string value of 6 digits.

2.31 Get Baud Rate

- Class Get (in)
- Identifier lt_GetBaudRate
- Hex Value 0x9B
- Call LT360LIB_CmdValue(Handle, lt_GetBaudRate, RtnStr);
- Returns *'nnnnn'*

Description
This command returns the current baud rate the LT360 is using. The number is returned as a string value of 4-6 digits.

2.32 Get Position

- Class Get (in)
- Identifier lt_GetPosition
- Hex Value 0x9C
- Call LT360LIB_CmdValue(Handle, lt_GetPosition, RtnStr);
- Returns *'±nnn.n'*

Description
This command returns the current position of the LT360 in degrees. The value is returned as a string with tenths of degree precision. The value may be 0 to +360.0 or 0 to ±180.0 depending on the settings for the display polarity.

This command is very useful when you want to know the current position of the LT360, or if the LT360 has completed a movement. It can be used in a polling loop to see when the LT360 step or goto command has finished. Another alternative method would be to call the *Get Moving* command until the *NO* value is returned.

When using either method in a polling loop, the frequency of calls should be reasonable. Typically, testing for movement completion once every second is more than adequate.

2.33 Get Pulse Direction

- Class Get (in)
- Identifier lt_GetPulseDir
- Hex Value 0x9D
- Call LT360LIB_CmdValue(Handle, lt_GetPulseDir, RtnStr);
- Returns *'CCW'* or *'CW'*

Description
This command returns the setting for the Pulse Direction. The value is returned as either CCW (counter clockwise) or CW (clockwise). This parameter controls the behavior to a TTL step pulse at the *Pulse* input BNC connector.

When a pulse arrives at the *Pulse* BNC input, the platter will rotate either CCW or CW as defined by this setting. The size of the step is controlled by the *Set StepSize* command.

Note: The TTL pulse should be a minimum of 10uS, or longer. The triggering edge can be controlled by the Set PulseEdge command.

2.34 Get Pulse Edge

- Class Get (in)
- Identifier lt_GetPulseEdge
- Hex Value 0x9E
- Call LT360LIB_CmdValue(Handle, lt_GetPulseEdge, RtnStr);
- Returns *'RISE'* or *'FALL'*

Description
This command returns the setting for the Pulse Edge triggering. The value is returned as either RISE (rising edge) or FALL (falling edge). This parameter controls the behavior to a TTL step pulse at the *Pulse* input BNC connector.

When a pulse arrives at the *Pulse* BNC input, the platter will rotate either at the rising or falling edge as defined by this setting. The size of the step is controlled by the *Set StepSize* command.

Note: The TTL pulse should be a minimum of 10uS, or longer. The direction can be controlled by the Set PulseDir command.

2.35 Get Step Size

- Class Get (in)
- Identifier lt_GetStepSize
- Hex Value 0x9F
- Call LT360LIB_CmdValue(Handle, lt_GetStepSize, RtnStr);
- Returns *'nnn.n'*

Description
This command returns the step size value. The value is returned as a string with tenths of a degree precision, and is always positive. This value sets the angular step size by which the platter will rotate to either a step command or TTL pulse.

The minimum step size is 0.1 degrees. Step size is always positive. Direction is controlled by the CCW/CW commands listed else where.

2.36 Get Velocity

- Class Get (in)
- Identifier lt_GetVelocity
- Hex Value 0xA0
- Call LT360LIB_CmdValue(Handle, lt_GetVelocity, RtnStr);
- Returns *'n.nn'*

Description
This command returns the velocity in RPM (revolutions per minute). The value is returned as a string with hundreths precision, and is always positive. The allowable range for the parameter is 0.01 to 3.00 RPM.

2.37 Get Torque

- Class Get (in)
- Identifier lt_GetTorque
- Hex Value 0xA1
- Call LT360LIB_CmdValue(Handle, lt_GetTorque, RtnStr);
- Returns *'nnn.n'*

Description
This command returns the torque in percent (%). The value is returned as a string with tenths precision, and is always positive. The allowable range for the parameter is 10.0 to 100.0 %.

This value sets the stepper motor torque for the platter rotation. The value is set in percent of maximum torque. For most typical applications your will want the torque set to 100%. If you intend to have the platter in constant motion, than you may wish to set the torque to a lower value such as 70%. This will reduce heating in the motor and drive circuitry.

The *SmartTorque* feature of the LT360 automatically reduces the motor torque to 1/2 its normal rotational power when it is not moving. Thus providing the advantages of maximum torque when in motion and reduced power consumption when stationary.

2.38 Get Acceleration Function

- Class Get (in)
- Identifier lt_GetAccelFunc
- Hex Value 0xA2
- Call LT360LIB_CmdValue(Handle, lt_GetAccelFunc, RtnStr);
- Returns *'n'*

Description
This command returns the acceleration function number (0 to 4). This value gives the acceleration function which the LT360 will use during step or goto commands. The acceleration profile is an important feature of the LT360 which enables heavy loads to be moved with minimum disturbance.

The parameter string contains a numeric value such as '1' with allowable values from 0..4. There are five acceleration functions numbered 0,1,2,3,4. The names of these profiles are:

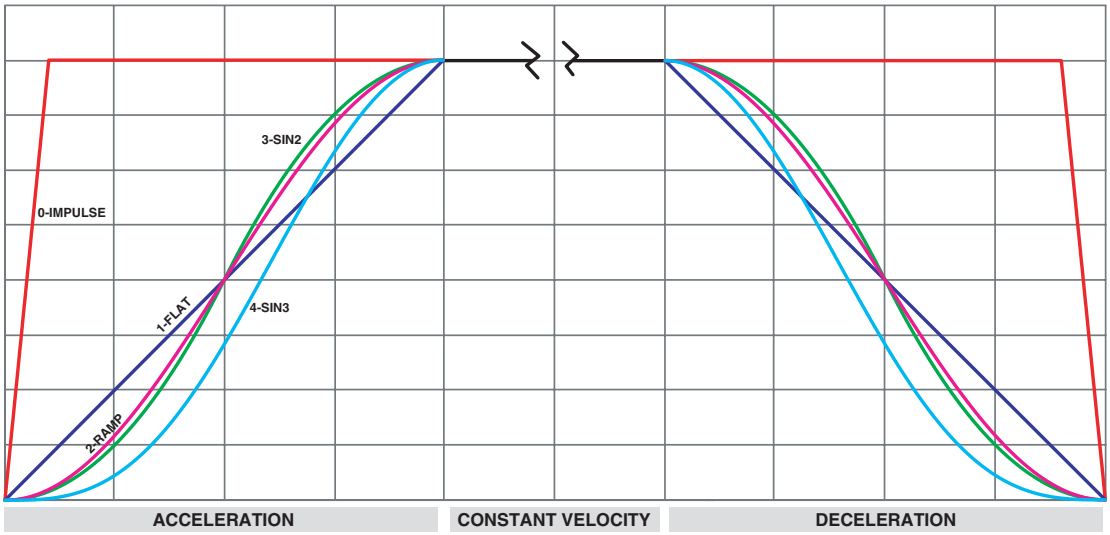
0 - Impulse
1 - Flat
2 - Ramp
3 - Sin2
4 - Sin3

For most typical applications acceleration function 1 - FLAT will provide excellent performance. Because the LT360 controls the acceleration and deceleration so precisely, very heavy loads can be rotated with ease.

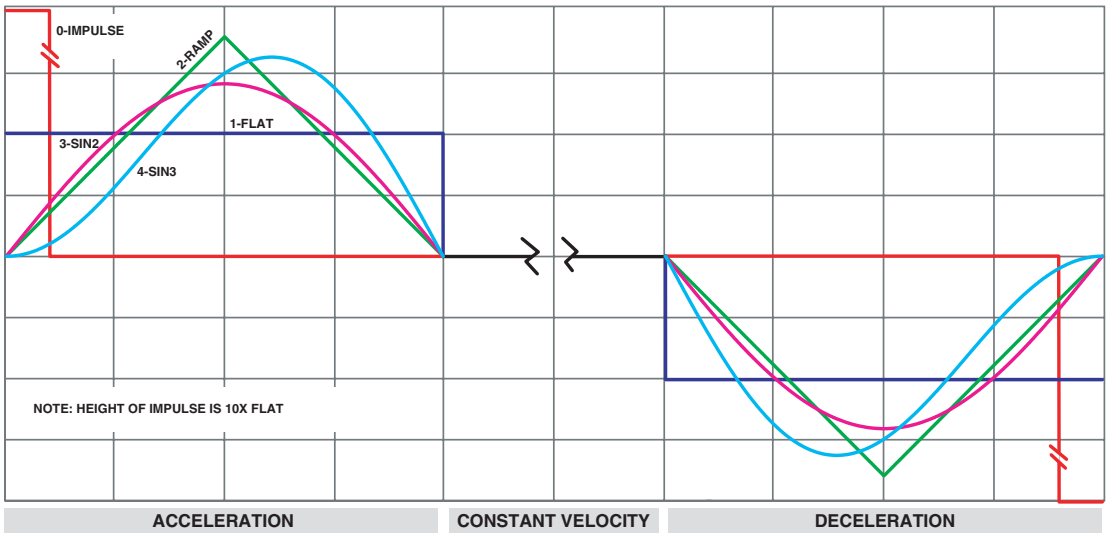
The 0 - IMPULSE function provides the fastest rise time from zero to constant velocity. However it also greatly reduces the maximum load which the LT360 can rotate, due to its higher acceleration. The acceleration of this function is about 10X that of the others.

The graphs on the following page show the velocity and acceleration profile curves for the five different LT360 functions.

LT360 VELOCITY PROFILES



LT360 ACCELERATION PROFILES



2.39 Get Moving

- Class Get (in)
- Identifier lt_GetMoving
- Hex Value 0xA3
- Call LT360LIB_CmdValue(Handle, lt_GetMoving, RtnStr);
- Returns *'CCW' or 'CW' or 'NO'*

Description
This command returns the rotational status of the LT360. Three possible string values can be returned:

CCW - rotating counter clockwise
CW - rotating clockwise
NO - not rotating

This command is very useful when you want to know if the LT360 has completed a movement. It can be used in a polling loop to see when the LT360 step or goto command has finished. Another alternative method would be to call the *Get Position* command until the desired location is returned.

When using either method in a polling loop, the frequency of calls should be reasonable. Typically, testing for movement completion once every second is more than adequate.

2.40 Get Pulse Input

- Class Get (in)
- Identifier lt_GetPulseInput
- Hex Value 0xA9
- Call LT360LIB_CmdValue(Handle, lt_GetPulseInput, RtnStr);
- Returns *'OFF' or 'ON'*

Description
This command returns the status for the Pulse Input BNC function of the LT360. The returned string value is either OFF or ON.

This parameter enables the TTL Pulse Input feature. When *PulseInput* is set ON, the LT360 will respond to TTL pulses at the BNC connector. When it is set OFF, it will ignore any pulses.

Each pulse will trigger a step. The width of the pulse should be at least 10uSec, and the triggering can be set on the rising or falling edge. Any pulses which occur while the LT360 is moving will be ignored.

If you are not using the TTL pulse input, it is probably best to keep this feature OFF to prevent false triggering due to any noise pickup at the open connector.

2.41 Get Analog Input

- Class Get (in)
- Identifier lt_GetAnalogInput
- Hex Value 0xAA
- Call LT360LIB_CmdValue(Handle, lt_GetAnalogInput, RtnStr);
- Returns *'OFF' or 'ON'*

Description
This command returns the status for the Analog Input BNC function of the LT360. The returned string value is either OFF or ON.

This parameter enables the Analog Input feature. When *AnalogInput* is set ON, the LT360 will respond to the DC voltage at the analog input BNC connector. When it is set OFF, it will ignore that input.

The LT360 has a scale factor of 10mV/deg for the analog input. The range can be either unipolar (0 ... +3.600V) or bipolar (0 ... ±1.800V). The input impedance of the analog input is 1M Ohm. Although the input is an unbalanced BNC connector, the actually circuitry is differential. This helps to reject ground noise.

The input source must be very stable and of low noise. The LT360 has a resolution of 0.1 degrees and therefore responds to changes of 1mV at the analog input. When this input is enabled, the LT360 will sample the input and move to the location as indicated by the voltage. After that move is completed, it then samples the analog input again and moves to the new position. Monitoring of the analog input is real time.

If you are not using the analog input, this input *must* be set OFF. Otherwise the LT360 will be controlled by any noise received at the input.

2.42 Get Smart Torque

- Class Get (in)
- Identifier lt_GetSmartTorque
- Hex Value 0xAB
- Call LT360LIB_CmdValue(Handle, lt_GetSmartTorque, RtnStr);
- Returns *'OFF' or 'ON'*

Description
This command returns the status for the Smart Torque feature of the LT360. The returned string value is either OFF or ON.

This parameter controls the behavior of the stepper motor. When *SmartTorque* is set ON, the motor is powered down to half the driving power 2 seconds after each movement stops. Motor power is returned to full (as controlled by the Maximum Torque setting) when the motor starts again.

This feature takes advantage of the fact that the worm gear drive system employed by the LT360 is irreversible. Meaning, the platter is self-locking at its current position. The load cannot rotate the platter itself. Therefore, it is not necessary to maintain full power on the motor merely to hold its current position when the platter is not in movement.

The use of *SmartTorque* greatly reduces the thermal heating in the motor and driver circuitry, and produces a very efficient drive system. Under most conditions and use, this feature should always be kept ON.

If a particular application demands that the motor power be kept constant at all times, even when stationary (SmartTorque=OFF), then the maximum torque should be to 70% or less to prevent excessive heating. It is difficult to envision what kinds of applications would require this behavior, but the option is available if needed.

2.43 Get Display Polarity

- Class Get (in)
- Identifier lt_GetDisplayPolarity
- Hex Value 0xAC
- Call LT360LIB_CmdValue(Handle, lt_GetDisplayPolarity, RtnStr);
- Returns *'UNIPOLAR' or 'BIPOLAR'*

Description
This command returns the mode for the Display Polarity of the LT360. The returned string value is either UNIPOLAR or BIPOLAR.

This parameter changes how the LT360 represents the degree position on the front display of the chassis, and in the Win32 software. When *DisplayPolarity* is set to UNIPOLAR, the LT360 will display the position in the range of 0 to 360 degrees. All position values are positive. When it is set to BIPOLAR, the range will be 0 to ±180 degrees. Both positive and negative values will be shown.

2.44 Get Input Polarity

- Class Get (in)
- Identifier lt_GetInputPolarity
- Hex Value 0xAD
- Call LT360LIB_CmdValue(Handle, lt_GetInputPolarity, RtnStr);
- Returns *'UNIPOLAR' or 'BIPOLAR'*

Description
This command returns the mode for the Analog Input Polarity of the LT360. The returned string value is either UNIPOLAR or BIPOLAR.

This parameter changes how the LT360 interprets the DC voltage received at the Analog Input connector. When *InputPolarity* is set to UNIPOLAR, the LT360 will accept voltages in the range of 0.000 to +3.600 volts. All position values are positive. When it is set to BIPOLAR, the range will be 0.000 to ±1.800 volts. Both positive and negative values are used.

Voltages higher than the allowable range for the selected mode are invalid.

The LT360 has a scale factor of 10mV/deg for the analog input. The range can be either unipolar (0 ... +3.600V) or bipolar (0 ... ±1.800V). The input impedance of the analog input is 1M Ohm. Although the input is an unbalanced BNC connector, the actually circuitry is differential. This helps to reject ground noise.

The input source must be very stable and of low noise. The LT360 has a resolution of 0.1 degrees and therefore responds to changes of 1mV at the analog input. When this input is enabled, the LT360 will sample the input and move to the location as indicated by the voltage. After that move is completed, it then samples the analog input again and moves to the new position. Monitoring of the analog input is real time.

If you are not using the analog input, this input *must* be set OFF. Otherwise the LT360 will be controlled by any noise received at the input.

2.45 Get Output Polarity

- Class Get (in)
- Identifier lt_GetOutputPolarity
- Hex Value 0xAE
- Call LT360LIB_CmdValue(Handle, lt_GetOutputPolarity, RtnStr);
- Returns *'UNIPOLAR' or 'BIPOLAR'*

Description
This command returns the mode for the Analog Output Polarity of the LT360. The returned string value is either UNIPOLAR or BIPOLAR.

This parameter changes how the LT360 generates the DC voltage output at the Analog Output connector. When *OutputPolarity* is set to UNIPOLAR, the LT360 will produce voltages in the range of 0.000 to +3.600 volts. All position values are positive. When it is set to BIPOLAR, the range will be 0.000 to ±1.800 volts. Both positive and negative values are produced.

The LT360 has a scale factor of 10mV/deg for the analog output. The range can be either unipolar (0 ... +3.600V) or bipolar (0 ... ±1.800V). The output impedance of the analog output is <200 Ohms. Although the output is an unbalanced BNC connector, it is highly recommended that the input on the other device be differential to reject ground noise. Since the smallest resolution of the LT360 is 0.1 degree, the equivalent resolution in the output is 1mV. The input impedance of the other device should be at least 100K Ohm to prevent loading errors.

2.46 Get Motor Home Check

- Class Get (in)
- Identifier lt_GetMotorHomeChk
- Hex Value 0xC1
- Call LT360LIB_CmdValue(Handle, lt_GetMotorHomeChk, RtnStr);
- Returns *'OFF' or 'ON'*

Description
This command returns the status for checking stepper motor synchronization. The returned string value is either OFF or ON.

When *MotorHomeChk* is set ON, the firmware checks that the motor returns to its home position after each movement. If disabled, no checking is performed. If checking is enabled and the motor does not end its operation in the Home position, an error message is shown on the front panel display.

The stepper motor produces a specific number steps for a given amount of platter rotation. The LT360 utilizes microstepping for additional precision control. One particular phase of the motor is designated as the *Home* position. For any given platter rotation, the motor should always end in its Home position.

In normal operation this error should never occur. If it does, the unit must be powered off to clear the error, or this feature disabled. It is possible for this error to be produced by noise in the stepper motor controller, or by the load exceeding the torque capability thereby causing the motor to skip steps.

If this error is shown repeatedly, either there is a problem internal in the LT360 or the torque load is too great for the drive system.

2.47 Get Revision Code

- Class Get (in)
- Identifier lt_GetRevCode
- Hex Value 0xC2
- Call LT360LIB_CmdValue(Handle, lt_GetRevCode, RtnStr);
- Returns *'nn'* ASCII code of chr

Description
This command returns the revision code of the controller PC board, as an ASCII code for the revision letter. For example, '65' = A.

2.48 Get Output Mode

- Class Get (in)
- Identifier lt_GetOutputMode
- Hex Value 0xC3
- Call LT360LIB_CmdValue(Handle, lt_GetOutputMode, RtnStr);
- Returns *'CONT'* or *'START'* or *'STOP'*

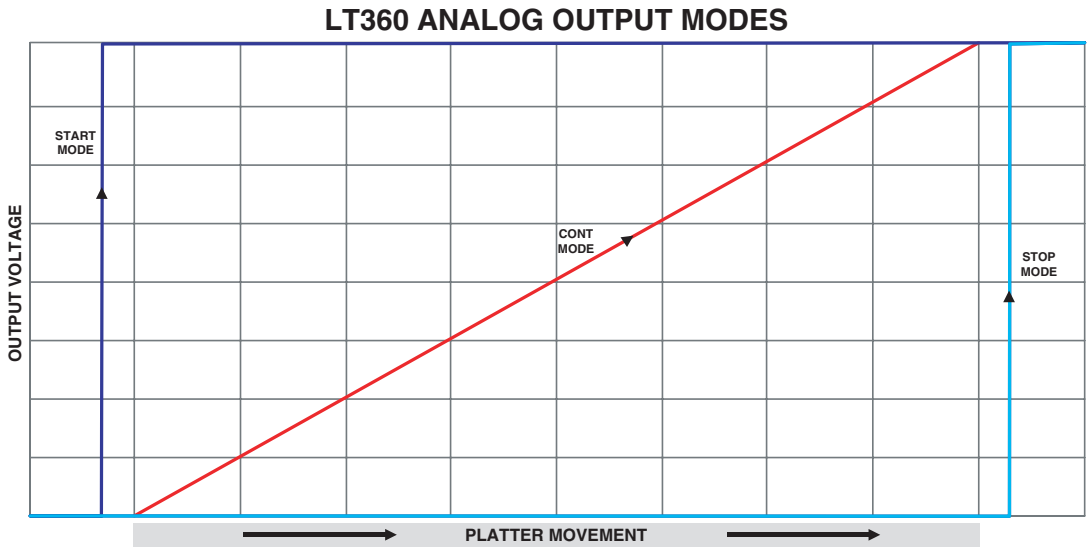
Description
This command returns the operating mode for the Analog Output. The returned string value is either CONT or START or STOP.

When *CONT* is the mode, the Analog Output will continuously follow the platter position as it is moving. This is a real time output mode where the analog output voltage always represents the current platter position.

When *START* is the mode, the Analog Output will only produce a single step change output voltage representing the final destination position at the start of the movement. For example, if the current position is 0.0 degrees and the destination is 100.0 degrees, the Analog Output voltage will change from 0.000V to 1.000V at the start of the movement.

When *STOP* is the mode, the Analog Output will only produce a single step change output voltage representing the final destination position after the movement stops. For example, if the current position is 0.0 degrees and the destination is 100.0 degrees, the Analog Output voltage will remain at 0.000V during the movement and then change to 1.000V when the movement stops. The final output voltage changes 2 seconds after the platter movemet stops.

If the movement is aborted before it reaches the destination, the output voltage will be updated to the correct final value in all modes.


LT360 ANALOG OUTPUT MODES
START MODE
CONT MODE
STOP MODE
OUTPUT VOLTAGE
PLATTER MOVEMENT