Logosol AC/DC Intelligent Servo Drive for Coordinated Motion Control LS-174

Doc # 712174001 / Rev. 1.07, 05/09/2002

Features

- **Motors supported:**
  - **Brushless 60/120° commutated (AC)**
  - **Brush-commutated (DC)**
- **12A peak, 8A continuous output current**
- **12 to 90V single power supply**
- **Path point buffer for coordinated motion control**
- **30/60/120 Hz rate between the points**
- **Absolute and relative moves**
- **32-bit position, velocity, acceleration, 16-bit PID filter gain values**
- **Comprehensive motor output short-circuit protection:**
  - **Output to output**
  - **Output to ground**
  - **Output to power**
- **Adjustable motor current limit**
- **Over/under voltage shutdown**
- **Overheating protection**
- **Emergency stop input**
- **Forward and reverse overtravel inputs**
- **Communication speed 19.2 - 115.2 KBps**
- **Servo rate 2 kHz**
- **PWM frequency 20 kHz**

Description

LS-174 is an enhanced version of the standard LS-173 Servo Drive, augmented with special features for supporting the coordinated motion of several motors. LS-174 is a single-axis motion controller with integrated servo amplifier designed for applications using brushless (AC) or brush-commutated (DC) motors up to 1 HP. No jumpers or setup is required to switch between both types of motors. Trapezoidal brushless motor commutation is performed automatically if hall sensors are connected to the unit.

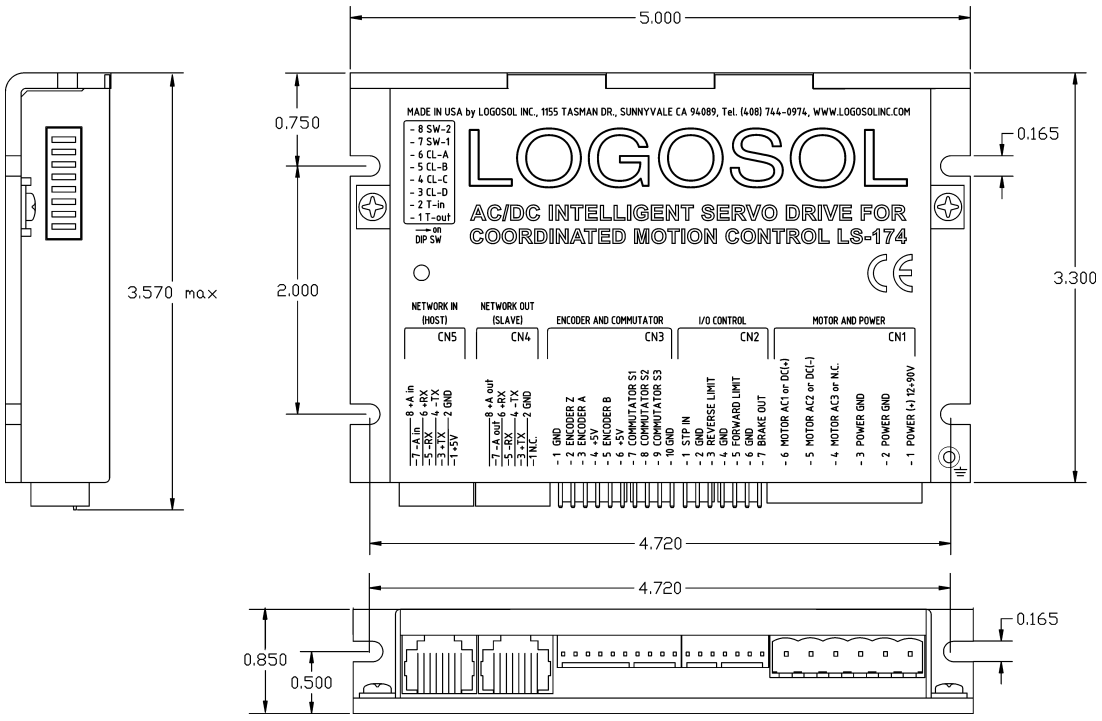
Up to 31 intelligent servo drives can be controlled over a multi-drop full duplex RS-485 network in a distributed motion control environment. Standard RJ-45 connectors and commercially available cables are used for daisy chaining of the modules.

LS-174 is equipped with various safety features such as short circuit protection for the motor and amplifier, over travel limit switch inputs, emergency stop input, over and under voltage shutdown. The maximum motor output current can be limited by setting of dipswitches.

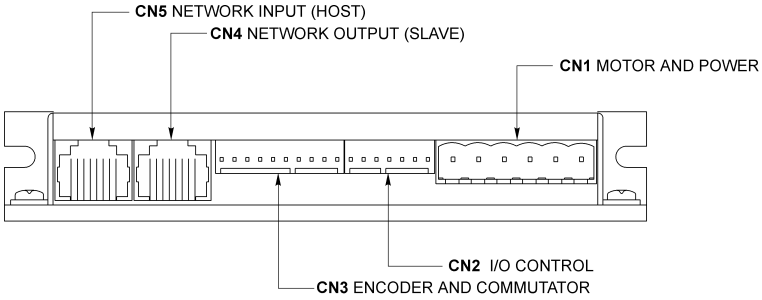
TECHNICAL SPECIFICATIONS rated at 25°C ambient, POWER (+)=60VDC, Load=250μH motor

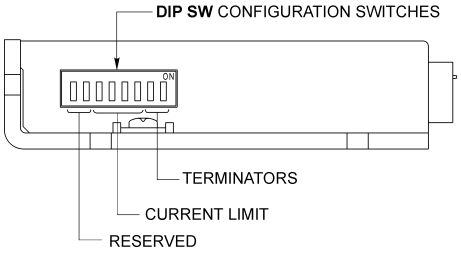
POWER SUPPLY VOLTAGE	12 to 90 V DC, 100V Absolute Maximum
MAX MOTOR OUTPUT CURRENT Peak Continuous	 12A 8A
MAX MOTOR OUTPUT VOLTAGE	V_{out}= 0.96(POWER (+)) – 0.17(I_{out})
MIN LOAD INDUCTANCE	200μH
PWM SWITCHING FREQUENCY	19,512 KHz
SERVO RATE	0.512 msec
SERIAL BAUD RATE	19.2 – 115.2 Kbps (faster communication rates are possible at lower servo rates)
OPTIONAL OPEN COLLECTOR BRAKE OUTPUT Max voltage applied to output Max current load	 48V 0.3A
INPUTS Encoder & Commutation Digital Inputs	 TTL with 1K pull-up to 5V LOmin=-1V, HImax=48V
ENCODER	Quadrature with index
COMMUTATION	Hall sensors 60/120°
INDICATORS Red LED (two intensity levels)	 Power 'ok' – low intensity Servo 'ok' – high intensity
PROTECTION Short circuit Overheating shut off	 Motor output to motor output Motor output to POWER GND Motor output to POWER (+) Activated at 80°C
FIRE-SAFETY Internal fuse	 10A Quick blow
POWER DISSIPATION (max)	30W
THERMAL REQUIREMENTS Storage temperature range Operating temperature range	 –30 to +85°C 0 to 45°C
MECHANICAL Size Weight	 L=5.00", H=3.30", D=0.85" 0.55lb. (250gr.)
MATING CONNECTORS Power & Motor Inputs & Outputs Encoder & Commutator Communication	 Magnum EM2565-06-VL or Phoenix MSTB 2.5/6-ST-5.08 Molex 22-01-3077 housing with 08-50-0114 pins (7 pcs.) Molex 22-01-3107 housing with 08-50-0114 pins (10 pcs.) 8 pin RJ-45

DIMENSIONAL DRAWING



SERVO DRIVE LAYOUT

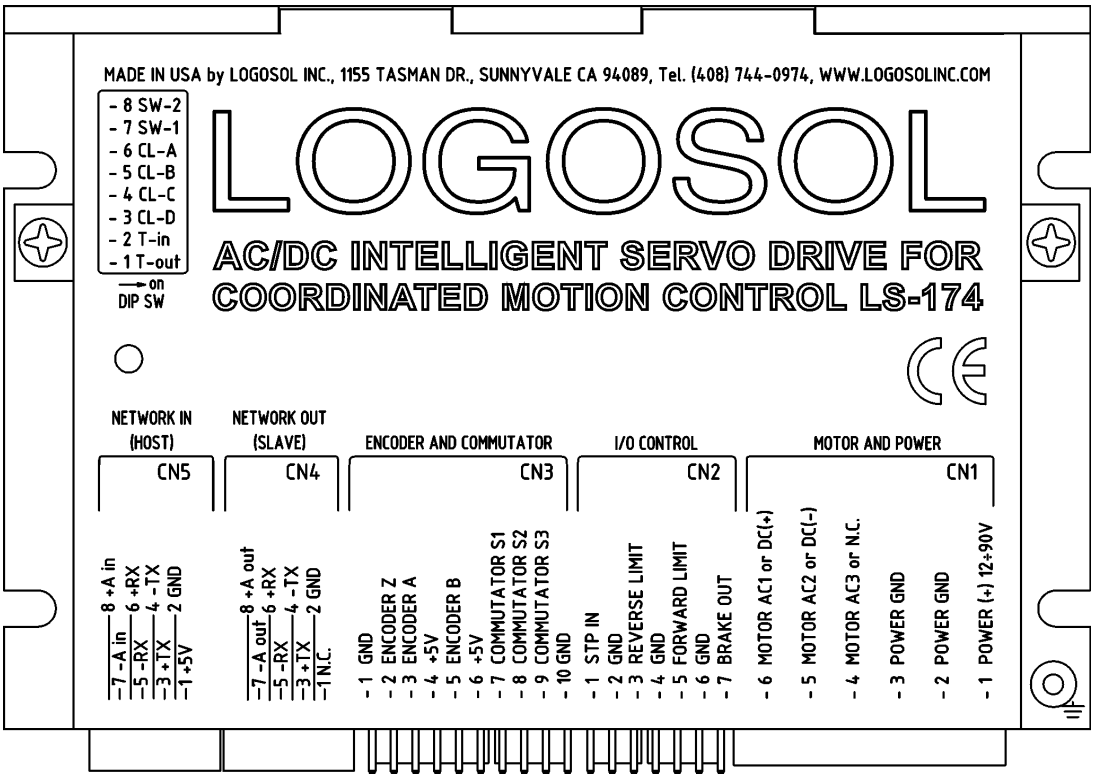




ORDERING GUIDE

PART NUMBER	MODEL	DESCRIPTION
912174001	LS-174	Intelligent Servo Drive with brake
230601004	LS-173-CN	Mating connector kit

CONNECTORS AND PINOUT



DIP SW – DIP SWITCH

SW	SIGNAL	DESCRIPTION	FACTORY SETTING
1	T-out	Transmit line terminator	OFF
2	T-in	Receive line terminator	OFF
3	CL-D	Current limit switch	OFF
4	CL-C	Current limit switch	ON
5	CL-B	Current limit switch	ON
6	CL-A	Current limit switch	ON
7	SW1	Reserved (must be set to ON)	ON
8	SW2	Reserved (must be set to ON)	ON

CN1 – POWER AND MOTOR CONNECTOR

PIN	SIGNAL	DESCRIPTION
1	POWER (+)	12 – 90V power supply, positive terminal
2	POWER GND*	Power supply ground
3	POWER GND*	Power supply ground
4	AC3 or NC	Output to motor Phase 3 terminal for brushless motor Not connected for brush motor
5	AC2 or DC (-)	Output to motor Phase 2 terminal for brushless motor Negative terminal for brush motor
6	AC1 or DC (+)	Output to motor Phase 1 terminal for brushless motor Positive terminal for brush motor

CN2 – I/O CONTROL

PIN	SIGNAL	DESCRIPTION
1	STP IN	Stop input (disable servo amplifier)
2	GND*	Signal ground
3	REVERSE LIMIT	Reverse limit input
4	GND*	Signal ground
5	FORWARD LIMIT	Forward limit input
6	GND*	Signal ground
7	BRAKE OUT	Brake output (optional). Open collector output 48V/0.3A

CN3 – ENCODER AND COMMUTATOR

PIN	SIGNAL	DESCRIPTION
1	GND*	Encoder ground
2	ENCODER Z	Encoder index
3	ENCODER A	Encoder phase A
4	+5V	Encoder power supply
5	ENCODER B	Encoder phase B
6	+5V	Commutator power supply
7	COMMUTATOR S1	Hall sensor #1
8	COMMUTATOR S2	Hall sensor #2
9	COMMUTATOR S3	Hall sensor #3
10	GND*	Commutator ground

CN4 – NETWORK OUT (SLAVE)

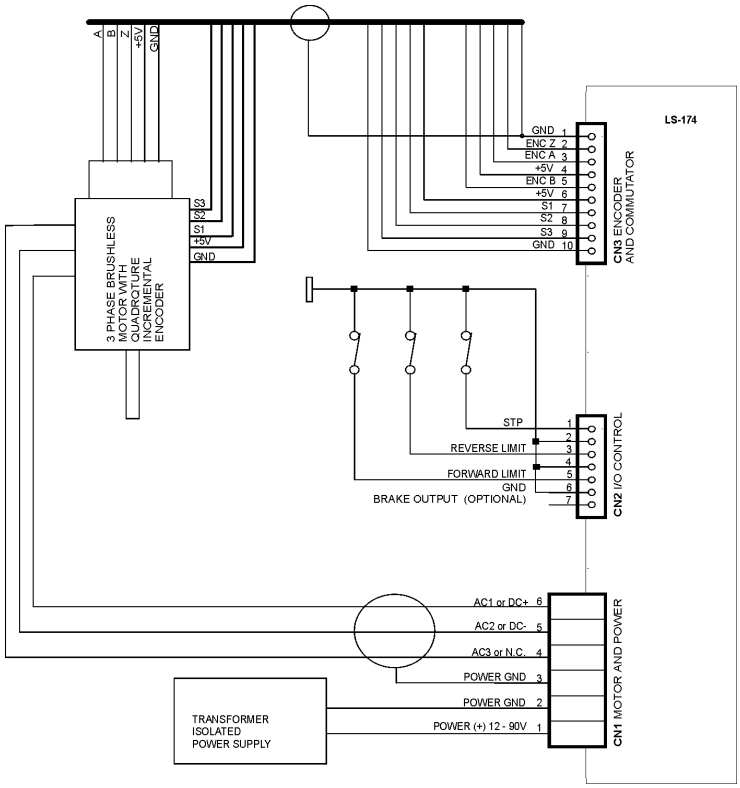
PIN	SIGNAL	DESCRIPTION
1	N.C.	Not connected
2	GND*	Interface ground
3	+TX	(+) Transmit data
4	-TX	(-) Transmit data
5	-RX	(-) Receive data
6	+RX	(+) Receive data
7	-A out	(-) Address output
8	+A out	(+) Address output

CN5 – NETWORK IN (HOST)

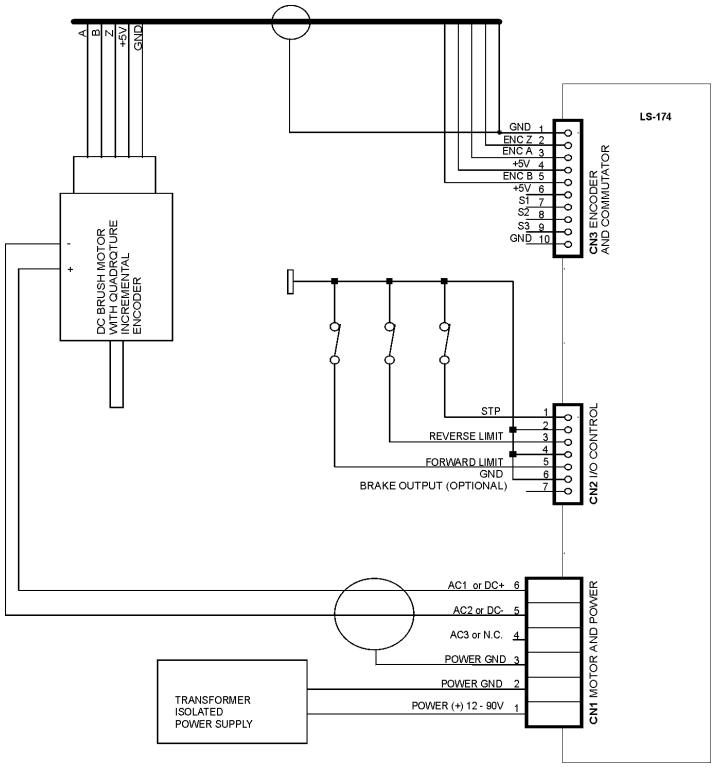
PIN	SIGNAL	DESCRIPTION
1	+5V	RS-232 adapter power supply
2	GND*	Interface ground
3	+TX	(+) Transmit data
4	-TX	(-) Transmit data
5	-RX	(-) Receive data
6	+RX	(+) Receive data
7	-A in	(-) Address input
8	+A in	(+) Address input

* *POWER GND and GND are electrically connected. Drive Case is isolated from drive circuitry and can be grounded externally.*

SAMPLE APPLICATION using AC (brushless) motor



SAMPLE APPLICATION using DC (brush) motor



LOGOSOL LS-174 QUICK START GUIDE

Hardware Setup

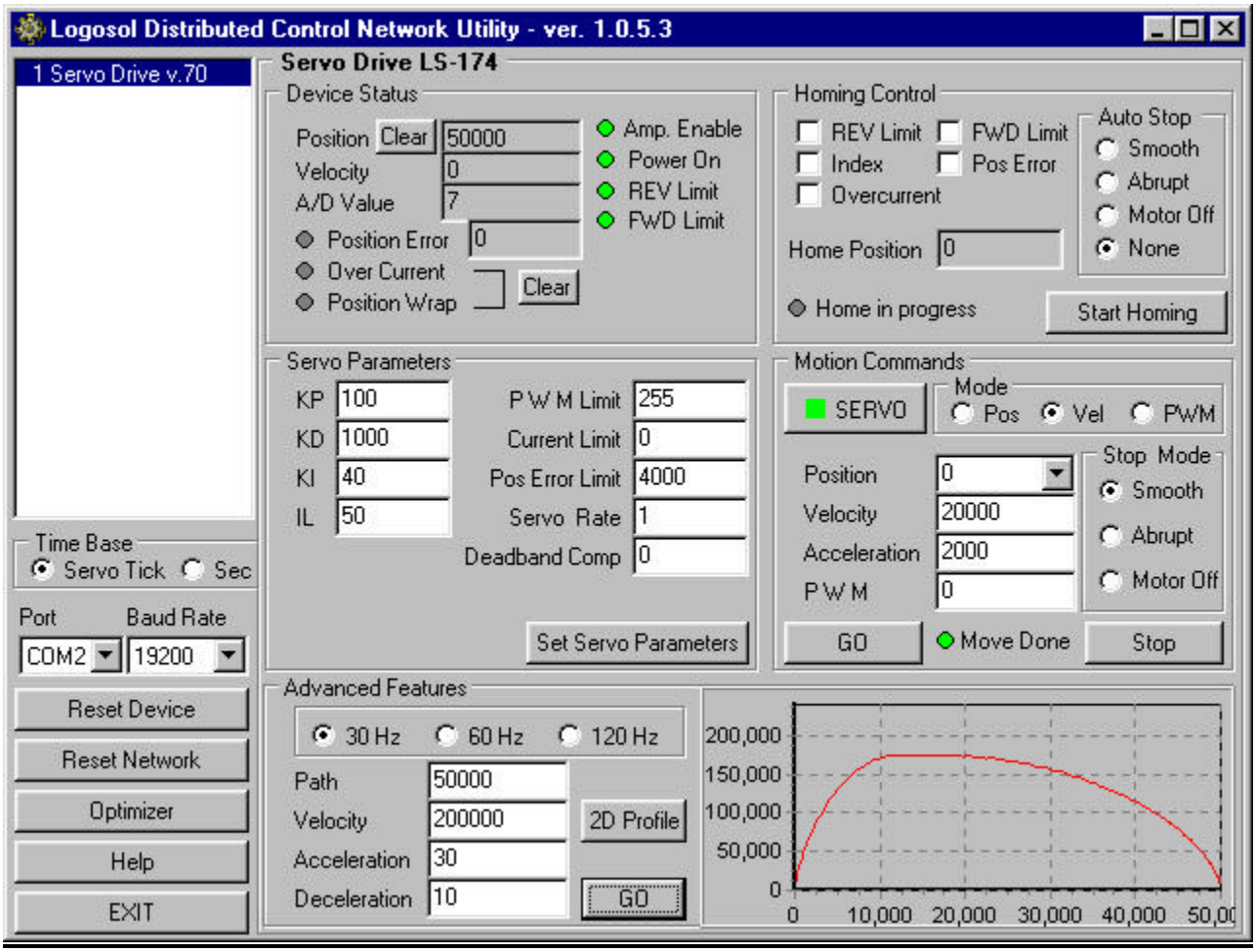
1. Connect power supply (12 to 90 V DC) to LS-174.
2. Connect your motor, encoder, Hall sensors and any other I/O you may have.
3. Connect RS-232 adapter and RJ-45 network cable between LS-174 and your host computer.

Software Installation

1. Installation and using Logosol Distributed Control Network Utility

A. Installation

1. Insert the Logosol Distributed Control Network Utility installation disk into the floppy drive.
2. Select Run from the Windows 95/98/NT Start menu.
3. Type a:\dcnsetup and then click OK (a: represents the drive letter).
4. The installation wizard will guide you through the setup process.



B. Initial Connection to the Host

1. Turn on the power supply.
2. Run the Logosol Distributed Control Network Utility.
3. Choose the proper COM port
4. Click “SERVO” button.
5. Click “GO” button. The motor should rotate slowly in positive direction. Click “Stop” to interrupt the motion. More information about using LDCN utility is available in LDCN Help.

2. Installation and using Logosol Motion Control Center

A. Installation

1. Insert the Logosol Motion Control Center installation disk into the floppy drive.
2. Select Run from the Windows 95/98/NT Start menu.
3. Type a:\mccsetup and then click OK (a: represents the drive letter).
4. The installation wizard will guide you through the setup process.

B. Initial Connection to the Host

1. Turn on the power supply.
2. Run the Logosol Motion Control Center software.
3. From the **Connection** menu select **Terminal**. This will open a terminal window. From the **Target** pull-down list select either RS-485-COM1 or RS-485-COM2 corresponding to the one used to communicate with LS-174. Press the Return key to verify that the connection is established and the command prompt (>) appears on the terminal window.
4. Type **INI** at the command prompt followed by Return to initialize the controller. It may take few seconds to complete the process.
5. Type **KP A1=20** to set the proportional coefficient, then set the velocity, acceleration and error limit by the following commands: **VEL A1=100**, **ACC A1=1000** and **MAX A1=8000**.
6. Type **SER** to close the servo loop, then **FOR A1** to instruct the controller to rotate the motor forward. Type **GO A1** to initiate the motion. The motor should rotate slowly in positive direction. Type **STO** to interrupt the motion and refer to the following MCL* Terminal Interpreter Command Set for executing various motion commands and parameters settings.

* The information about MCL Terminal Interpreter command set is also available in MCC Help Menu Index (Advanced Features /Command set for Distributed Servo Drive maintenance).

MCL TERMINAL INTERPRETER COMMAND SET

The following commands are available from the Terminal prompt:

POS – returns and sets current position:

1) "**POS**" returns the positions of all the axes;
2) "**POS A1**" returns the position of axis A1;
3) "**POS A1=Y**" sets the position of axis A1 to the specified value Y.

MAX – returns and sets maximal error (the difference between calculated and real positions):

1) "**MAX**" returns the maximal errors of all the axes;
2) "**MAX A1**" returns the maximal error of axis A1;
3) "**MAX A1=Y**" sets the error limit of axis A1 to the specified value 0<Y<16383.

VEL – returns and sets velocity in velocity mode or goal velocity in trapezoidal mode:

1) "**VEL**" returns the velocities of all the axes;
2) "**VEL A1**" returns the velocity of axis A1;
3) "**VEL A1=Y**" sets the velocity of axis A1 to the specified value Y.

ACC – returns and sets acceleration:

1) "**ACC**" returns the accelerations of all the axes;
2) "**ACC A1**" returns the acceleration of axis A1;
3) "**ACC A1=Y**" sets the acceleration of axis A1 to the specified value Y.

XST – displays status information:

1) "**XST**" displays status info for all the modules in the network;
2) "**XST A1**" displays status info for module A1.

Status information for module LS-174 includes:

1) Address
2) Status byte
3) Auxiliary status byte
4) Encoder position
5) Device ID
6) Version number
7) ADC value.

STA – displays MCL compatible status:

1) "**STA**" returns MCL compatible status of all the axes;
2) "**STA A1**" returns MCL compatible status of axis A1.

Note: Status 00000400 - position reached;
00000080 - servo off.

INI – resets the network and assigns individual and group addresses.

STO – stops abruptly:

1) "**STO**" stops the movement of all the axes;
2) "**STO A1**" stops the movement of axis A1.

HAL – stops smoothly with specified acceleration:

1) "**HAL**" stops the movement of all the axes;
2) "**HAL A1**" stops the movement of axis A1.

KP – returns and sets proportional PID filter parameter:
1) "**KP**" returns KP parameters for all the axes;
2) "**KP A1**" returns KP parameter for axis A1;
3) "**KP A1=Y**" sets KP of axis A1 to the specified value Y.

Note: KP is set to 0 after power-up.

KI – returns and sets integral PID filter parameter:
1) "**KI**" returns KI parameters for all the axes;
2) "**KI A1**" returns KI parameter for axis A1;
3) "**KI A1=Y**" sets KI of axis A1 to the specified value Y.

Note: KI is set to 0 after power-up.

KD – returns and sets deferential PID filter parameter:
1) "**KD**" returns KD parameters for all the axes;
2) "**KD A1**" returns KD parameter for axis A1;
3) "**KD A1=Y**" sets KD of axis A1 to the specified value Y.

Note: KD is set to 0 after power-up.

IL – returns and sets integral limit PID filter parameter:
1) "**IL**" returns IL parameters for all the axes;
2) "**IL A1**" returns IL parameter for axis A1;
3) "**IL A1=Y**" sets IL of axis A1 to the specified value Y.

Note: IL is set to 0 after power-up.

ABS – sets absolute motion position for specified axis – "**ABS A1=Y**".

REL – sets relative motion position for specified axis – "**REL A1=Y**".

PWM – returns and sets PWM value in range -255÷255:
1) "**PWM**" returns the PWM values of all modules.
2) "**PWM A1**" returns the PWM value of axis A1;
3) "**PWM A1=Y**" sets PWM of axis A1 to value Y;

FOR – sets forward motion in velocity mode:
1) "**FOR A1**" sets the motion of axis A1 with the current speed;
2) "**FOR A1=Y**" sets the motion of axis A1 with the specified speed Y.

REV – sets reverse motion in velocity mode:
1) "**REV A1**" sets the motion of axis A1 with the current speed;
2) "**REV A1=Y**" sets the motion of axis A1with the specified speed Y.

GO – starts motion in previously defined mode and parameters:
1) "**GO**" starts motion of all axes;
2) "**GO A1**" starts motion of axis A1.

CLI – returns and sets current limit:
1) "**CLI**" returns the current limits of all the axes;
2) "**CLI A1=0**" disables current limit of axis A1;
3) "**CLI A1=X**" sets continuous current limit of axis A1 to value X (range 0÷127).

Note: CL=2X+1. CL – current limit parameter of Set Gain command.

SER – enables servo:
1) "**SER**" enables servo of all connected modules;
2) "**SER A1**" enables servo of module A1.

NOS – disables servo:
1) "**NOS**" disables servo of all connected modules;
2) "**NOS A1**" disables servo of module A1.

INX – finds index:
1) "**INX A1 F**" finds index of axis A1 in forward direction;
2) "**INX A1 R**" finds index of axis A1 in reverse direction.

Note: if the motor makes more than one revolution to find index, decrease the speed.

FLS – finds limit switch:
1) "**FLS A1 F**" finds limit switch of axis A1 in forward direction;
2) "**FLS A1 R**" finds limit switch of axis A1 in reverse direction.

HEX – hex command mode – sends a low-level command written in hexadecimal format. For more information about command format refer to "Command Description" section in this document. Start byte (AA) and checksum byte are generated by the MCL interpreter:
"**HEX 01 05**" sends *Start Motion* command (code 0x05) for module with address 1.

BDR – sets baud rate. Possible baud rate values are 9600, 19200, 57600 and 115200:
"**BDR 115200**" – sets baud rate to 115.2 Kbps.

Note: baud rate is set to 19.2 Kbps after power-up.

HIS – shows the history of the recently used commands, their hexadecimal codes and the returned status packets.

EXE – executes a text file containing sequence of MCL commands:
"**EXE control.dat**" – executes the command sequence from "*control.dat*" file in current directory.

VER – returns MCL interpreter version.

NET – displays the number and types of all modules in the network and their addresses.

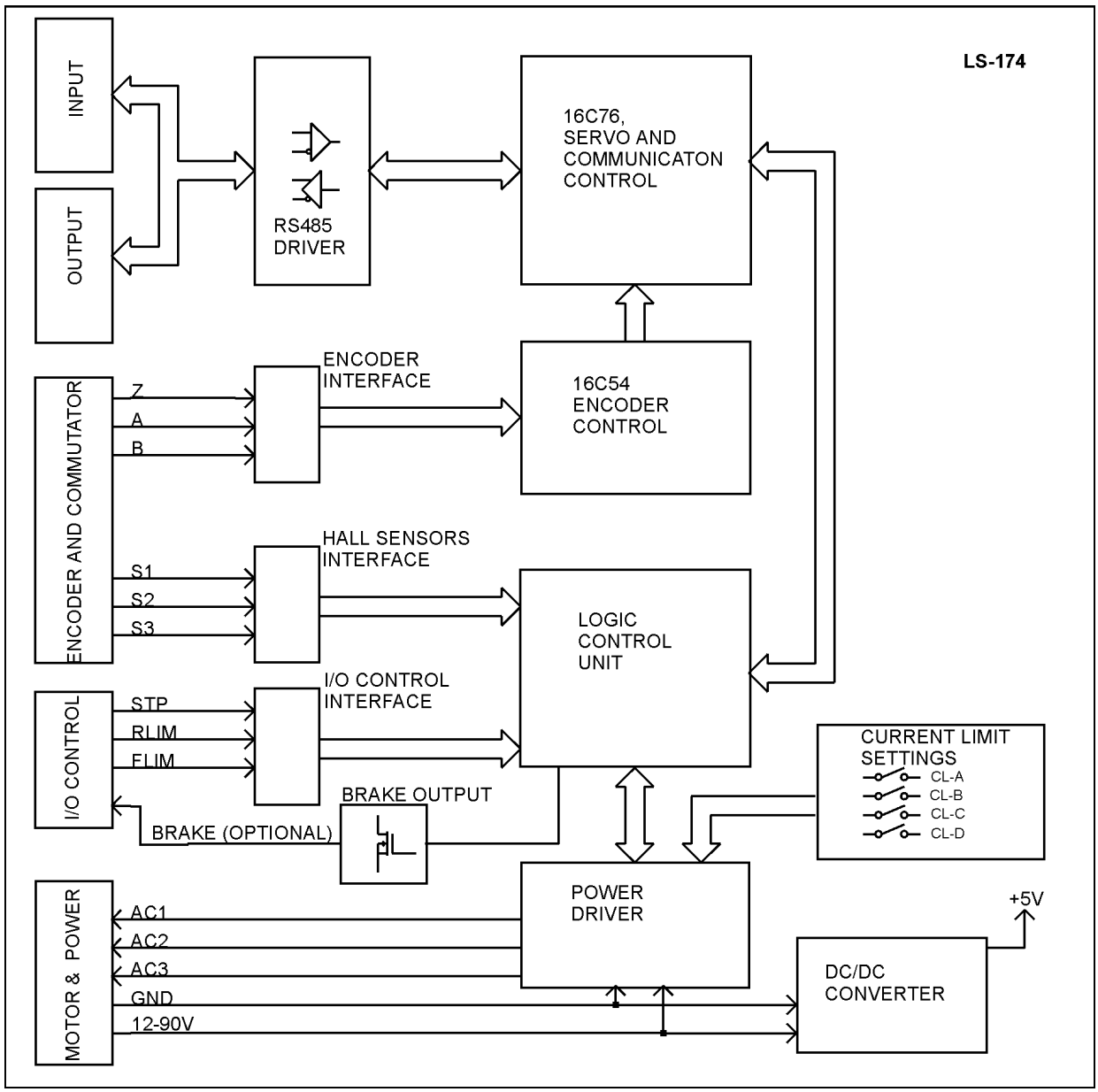
BRL – releases the motor brake to enable manually positioning:
1) "**BRL**" releases the brakes of all connected modules;
2) "**BRL A1**" releases the brake of module A1.

DBC – returns and sets amplifier deadband compensation value:
1) "**DBC**" returns the deadband compensation values of all the axes;
2) "**DBC A1**" returns the deadband compensation value of axis A1;
3) "**DBC A1**=**Y**" sets the deadband compensation value of axis A1 to the value Y.

LS-174 ARCHITECTURE

Overview

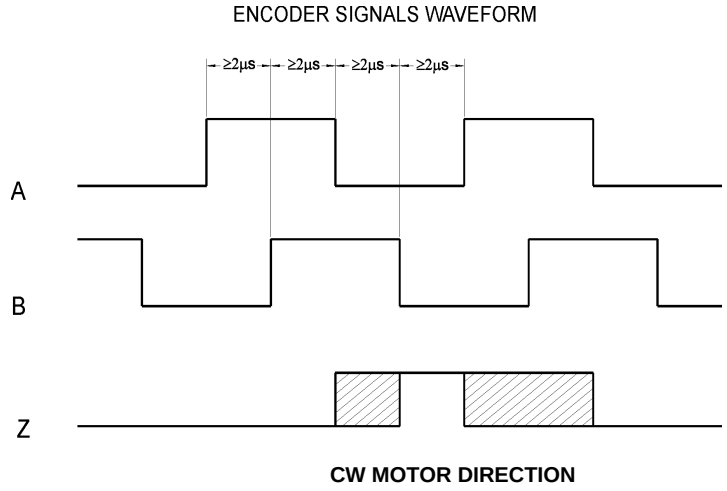
The LS-174 Intelligent Servo Drive is a highly integrated servo control module including a motion controller, servo amplifier, serial communication interface, optical encoder interface, limit switch inputs, and protection circuit (short circuit, under and overvoltage, overcurrent and software controlled current limit). The Servo Drive is designed so that up to 31 controllers can be daisy-chained and connected directly to a single standard serial port (RS-232 adapter may be necessary).



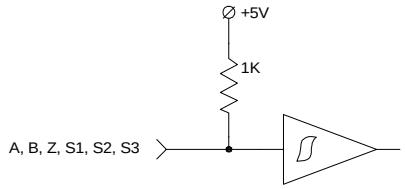
Functional Diagram

Encoder Input

The encoder interface accepts two square wave inputs, CH_A, CH_B and Index from an incremental encoder. Ideally, these square waves are 50% duty cycle and exactly +/-90 degrees out of phase. In any case, the time between encoder state transitions should be not less than 2 µsec. With ideally formed encoder pulses, this would correspond to a 500-line encoder (2000 counts/rev) rotating at 15,000 RPM.



All encoder inputs are with pull-up resistors 1K to +5V.



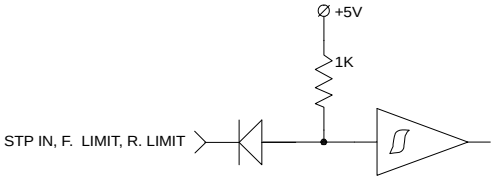
Encoder and Hall Inputs

Hall Inputs

Hall sensor inputs are placed on the same connector as encoder inputs. All hall sensors are with pull-up resistors 1K to +5V. 60°/120° hall sensors may be used.

Digital Inputs

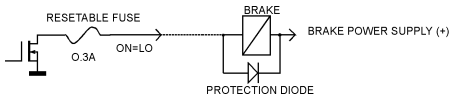
There are 3 digital inputs - STP IN, FORWARD LIMIT and REVERSE LIMIT. STP IN may be used only as “STOP” input. Limit inputs may be used as “HOME” switches, limit switches or as general-purpose inputs. (Refer to “I/O Control” and “Set Homing Mode” commands in the “Command Description” section in this document) All are with pull-up resistors 1K to +5V.



Limit Switches and Stop Input

Brake Output

Brake is released (brake output is “on”) when Power_on (bit3 of Status byte) and Pic_ae (bit0 of Stop command data byte) are set to 1.



Brake Output

Brake will be engaged (Brake output is “off”) if:

- STP IN is open;
- Overvoltage;
  Overcurrent;
- Motor short;
- Overheat;
- Position error exceeds the position error limit.

Note: For additional information refer to “Status bits and LED”, “Status byte and Auxiliary status byte” and “Stop” command description, sections of this document. If Power Driver is OK, brake will be released after Pic_ae 0 to 1 transition.

Dip Switch

Dip switch is used for setting overcurrent limit and terminator control (refer to “Overcurrent DIP Switch Setting” of “Safety Features” in this document) Two of switches, T-in and T-out, are used for connecting terminators to receive and transmit lines. SW-1 and SW-2 are factory reserved and must be set to ON.

Serial Command Interface

Serial communication with the LS-174 drives adheres to a full-duplex (4 wire) 8 bit asynchronous protocol with one start bit, followed by 8 data bits (lsb first), followed by a single stop bit.

The communication protocol of the LS-174 also supports a full-duplex multi-drop RS-485 interface that allows multiple LS-174 intelligent servo drives to be controlled over a single RS-485 port. In this case, the host sends commands over its RS-485 transmit line and receives all status data back over the shared RS-485 receive line.

The command protocol is a strict master/slave protocol in which the host master sends a command packet over the command line to a specific LS-174 slave. The data are stored in the buffer of the LS-174 until the end of the current servo cycle (0.512 msec max.) and then the command is executed. The servo drive then sends back a status packet. Typically, the host does not send another command until a status packet has been received to insure that it does not overwrite any previous command data still in use.

Each command packet consists of following:

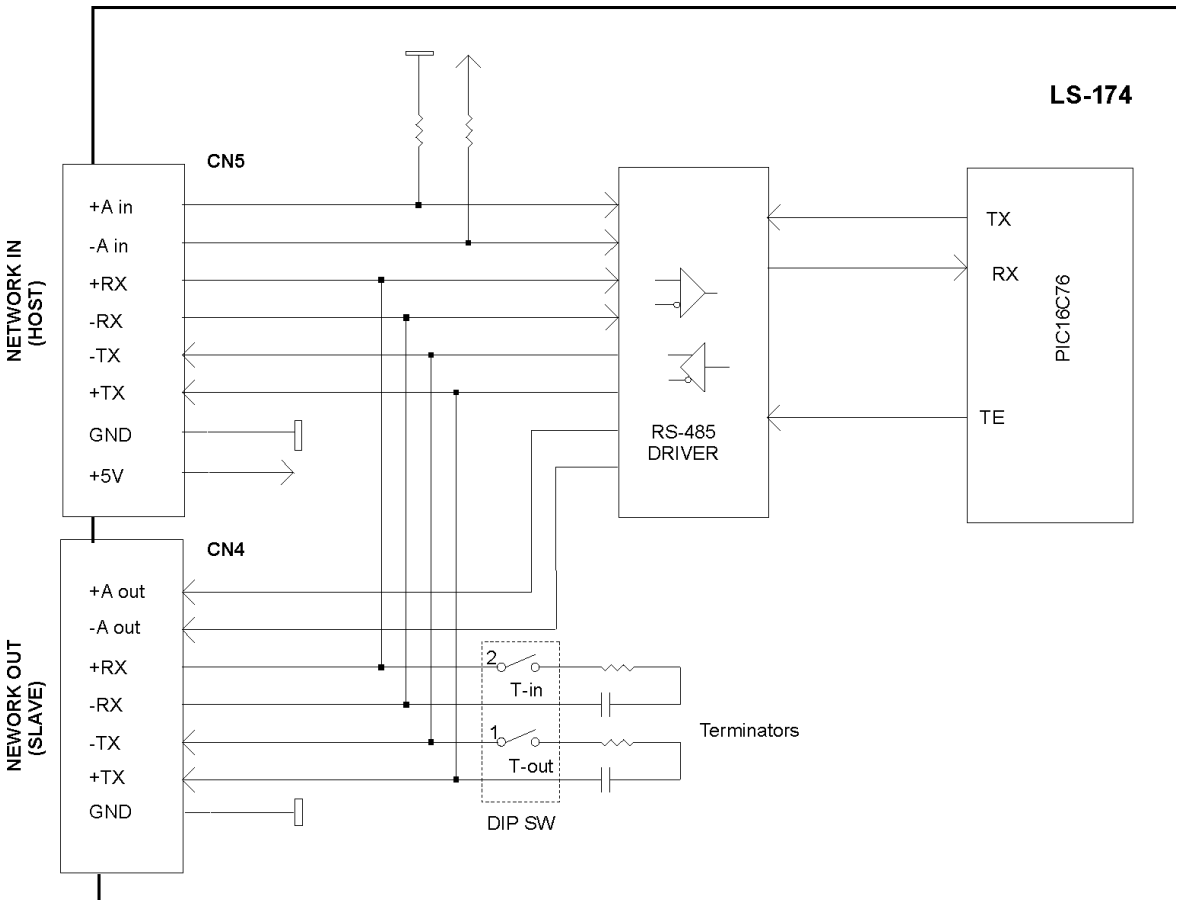
Header byte (0xAA)
Address byte - individual or group (0x00 - 0xFF)
Command byte
0 - 15 data bytes
Checksum byte

The command byte is divided into upper and lower nibbles: the lower nibble is the command value; the upper nibble is the number of additional data bytes, which will follow the command byte. The checksum byte is 8 bit sum of the address byte, the command byte and the data bytes.

The number of data bytes depends on the particular command chosen. After a command is issued, the corresponding controller will send back a status packet consisting of:

Status byte
0-16 optional bytes of status data
Checksum byte

The status byte contains basic status information about the LS-174, including a checksum error flag for the command just received. The optional data bytes may include data such as the position, velocity, etc. and are programmable by the host. The checksum byte is the 8 bit sum of the status byte and the additional optional status data bytes. All 16-bit and 32-bit data are sent with the least significant byte first.



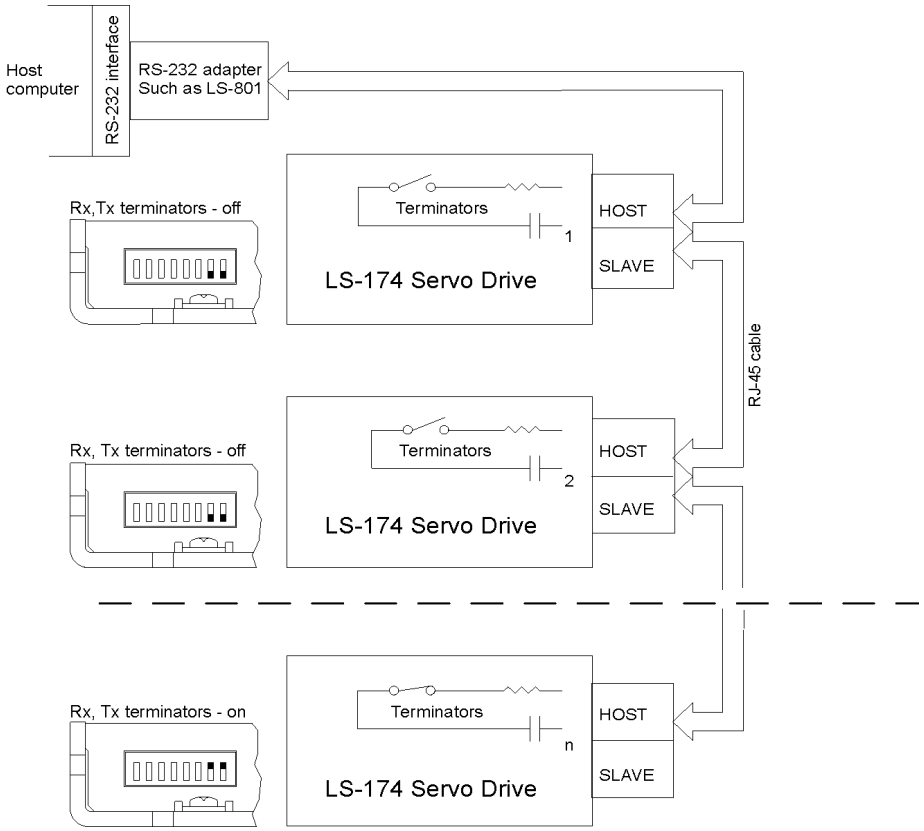
Servo Driver Serial Interface

Addressing

Rather than having to hard-wire or switch-select the address of each LS-174 servo drive, the host dynamically sets the address of each LS-174 with the aid of the daisy-chained “A in” and “A out” lines. This allows additional LS-174 controllers to be added to an RS-485 network with no hardware changes. On power-up, “A in” of the first LS-174 is pulled low, its communication is enabled and the default address is 0x00. When the *Set Address* command is issued to give this LS-174 new unique address, it will lower its “A out” pin. Connecting “A out” pin to the “A in” pin of the next servo drive in the network will enable its communication at default address of 0x00. Repeating this process allows a variable number of controllers present to be given unique addresses. See “Initializing procedure and programming examples for LS-174” later in this document.

Group Addresses

In addition to the individual address, each controller has a secondary group address. Several LS-174 controllers may share a common group address. This address is useful for sending commands, which must be performed simultaneously by a number of drivers (e.g. *Start motion, Set Baud Rate*, etc.). When a LS-174 receives a command sent to its group address, it will execute the command but will not send back a status packet. This prevents data collisions on the shared response line. When programming group addresses, however, the host can specify that one member of the group is the "group leader". The group leader will send back a status packet just like it would for a command sent to its individual address. The group address is programmed at the same time as the unique individual address using the *Set Address* command.



Multiple Controller Configuration

Communication Rate

The default baud rate after power-up is 19.2 Kbps. Baud rates up to 115.2 Kbps may be used at maximum servo rate. After communication has been established with all servo drives on a single network, the baud rate may be changed to a higher value with the *Set Baud Rate* command.

Servo Control

LS-174 uses a "proportional-integral-derivative", or PID filter. The PWM signal is a square wave with 51.2 µsec period and varying duty cycle. A PWM value of 255 corresponds to 100% and a value of 0 corresponds to 0%. Usually, PWM value greater than 250 is not recommended. The position, velocity and acceleration are programmed as 32-bit quantities in units of encoder counts for servo ticks. For example, a velocity of one revolution per second of a motor with a 500 line encoder (2000 counts/rev) at a tick time of 0.512 msec. would correspond to a velocity of 1.0240 counts/tick. Velocities and accelerations use the lower 16 bits as a fractional component so the actual programmed velocity would be 1.024 x 2^{16} or 67,109. An acceleration of 4 rev/sec/sec

(which would bring us up to the desired speed in ¼ sec) would be 0.0021 counts/tick/tick; with the lower 16 bits the fractional component, this would be programmed as 0.0021 x 2^{16} or 137. Position is programmed as a straight 32-bit quantity with no fractional component. Note that if the servo rate divisor is modified, the time dependent velocity and acceleration parameters will also have to be modified.

PWM Mode Operation

If the position servo is disabled, the motor is operated in a raw PWM output mode and no trapezoidal or velocity profiling is performed. In this mode, a user specified PWM value is outputted directly to the amplifier. Command position is continually updated to match the actual position of the motor and there will be no abrupt jump in the motor's position when position or velocity modes are entered. Also while the position servo is disabled, the command velocity is continually updated to match the actual velocity of motor. Thus, when velocity mode is entered, there will be no discontinuity in the motor's velocity. (Trapezoidal profile motions, however, will still force the motor to begin at zero velocity.).

Connecting Brushless or Brush Type Motor

LS-174 is capable of driving brushless commutated (AC) and brush (DC) type motors. No jumpers or other setting are required. If there are no Hall sensors connected to "ENCODER AND COMMUTATOR", LS-174 drives the motor as brush (DC) type. The positive motor lead should be connected to "AC1 or DC+" terminal and negative to "AC2 or DC-" terminal of "MOTOR AND POWER" connector. If Hall sensors are detected, LS-174 performs commutation according to their state.

Often, connecting the brushless motor phases is difficult because of the different terms and signal names, which different manufactures are using. Here is a simple procedure that may be used.

Connect the motor commutation sensors to LS-174 "ENCODER AND COMMUTATOR" connector according to the next table with most common manufacture signal names.

LS-174 Encoder & Sensor Connector signal	Motor manufacture signal name			
S1	R	U	A	S1
S2	S	V	B	S2
S3	T	W	C	S3

Connect the commutator power leads to GND and +5V. Connect the encoder and its power lines to the same connector. Connect the three motor leads to "AC1 or DC+", "AC2 or DC-", "AC3 or NC" of LS-174 "MOTOR AND POWER" connector using the same order as for the commutation sensors. Power on LS-174. Initialize the controller. Rotate motor shaft CW (ClockWise) by hand and check if the motor position is increasing. If motor position is not changing or it is decreasing, check encoder connection. Set the Drive in PWM mode. Start the motor with PWM for example 5 (this value might be enough or not depending on motor used) Set PWM to –5. If the phasing is correct the motor shaft should rotate CW (CCW) smoothly without any jerks. Otherwise try different motor leads connection. There are only six combinations and it is recommended to try all of them. Usually only one works fine. If you find more than one, try to run the motor at higher speed. Set the Drive in velocity mode and start the motor in CW direction. If the motor runs away, directions of motor and encoder are opposite. To change the motor direction exchange S1 with S3 and AC1 with AC2. To change the encoder direction exchange A and B phase wires.

Safety Features

To protect both the user device and the controller, LS-174 is equipped with various safety features.

STP IN – Stop Input

For normal operation STP signal must be “low”. If it is “high” it will disable the Power Driver and set status byte bit 3 (Power_on) to zero.

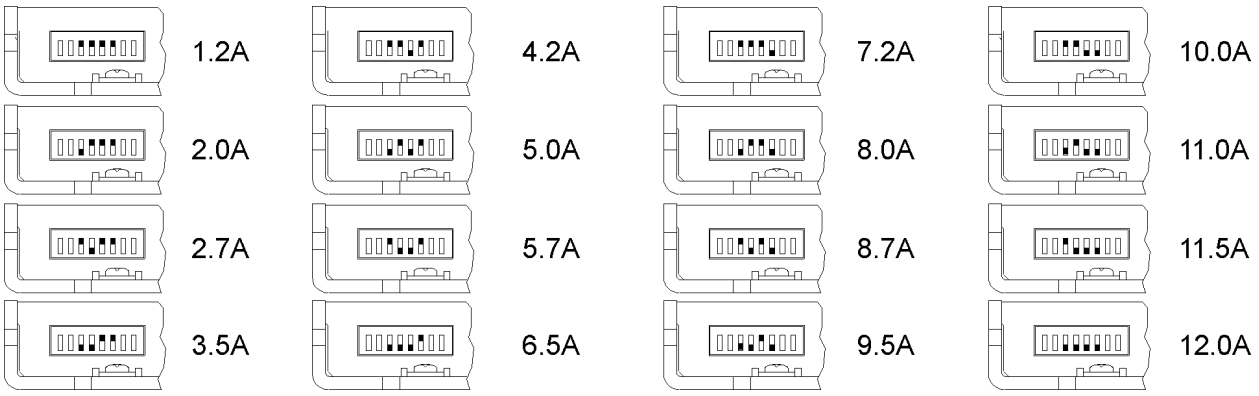
Undervoltage/Overvoltage Protection

LS-174 is protected against power supply under/overvoltage. In case the power supply is below 12V hardware reset is generated. More then 91V will disable the Power Driver and set status byte bit 3 (Power_on) to zero.

Overcurrent Protection

A protection circuit monitors the output current of the motor and limit to a value set by dipswitch. If the motor is overloaded for less than 100 ms, the output current is limited to the selected level. Power Driver will be disabled if the motor is overloaded for more then 100 ms. To set CL (current limit) use the table:

Overcurrent DIP Switch Settings



Motor current monitoring

Motor current can be monitored using *Read Status* command (refer to Command Description section of this document). A/D value is proportional to the motor current according to the following table:

A/D values as function of motor current

Motor Current	1A	2A	4A	6A	8A
A/D value	25	50	100	150	200

A/D value and CL (current limit parameter of *Set Gain* command) may be used for current limit control. CL is compared each servo tick with A/D value (proportional to the motor current). The actual PWM output value is:

$$PWM=PWMcalc - PWMadj$$

Where: PWM is output value; PWMcalc is motion command calculated value; PWMadj ($0<PWMadj\leq PWMcalc$) is internal parameter. If CL<A/D PWMadj is incremented by 1 each servo

tick. If CL>A/D PWMadj is decremented by 1 to 0. Bit 2 (Current_Limit) of status byte will be set. CL is in the range of 0 ÷ 255 and only odd values must be used.
If A/D>CL for more than 200ms the Power Driver will be disabled (refer to Status Bits and LED section of this document).
"Logosol MCL Interpreter for LS-174" CLI command (refer to *Logosol MCL Interpreter for LS-174 Intelligent Servo Drive* section in this document) is provided for current limit control. To disable the function set CL=0 of *Set Gain* command.

Status Bits and LED

Bit 3 (Power_on), bit 5 (Reverse Limit) and bit 6 (Forward Limit) of Status Byte* and bit 0 (Index) of Auxiliary Status byte* are used for reading input signals and driver diagnostics as shown on tables below.

Power Driver OFF condition (Stop command bit 0 (Pic_ae)=0)

Status byte diagnostic bits					
Bit 6 (Forward Limit)	**Bit 5 (Reverse Limit)**	**Bit 3 (Power_on)**	**Condition**		**LED intensity**
1	1	0	A	OVERVOLTAGE	Low
1	0	1	B	STP IN ACTIVATED	Low
0	1	1	C	OVERHEAT	Low
1	0	0	D	A+B	Low
0	1	0	E	A+C	Low
0	0	1	F	B+C	Low
0	0	0	G	A+B+C	Low
1	1	1	H	OK CONDITION	Low

Servo Drive diagnostic codes (Stop command bit 0 (Pic_ae)=0)

Status byte (hex)				
Pos_error =1	**Pos_error =0**	**Fault code description**		**Index**
71	61	A	OVERVOLTAGE	1
59	49	B	STP IN ACTIVATED	1
39	29	C	OVERHEAT	1
51	41	D	A+B	1
31	21	E	A+C	1
19	09	F	B+C	1
11	01	G	A+B+C	1
79	69	H	OK CONDITION	1

* *Refer to Status Byte and Auxiliary Status Byte description in "Command Description" section in this document.*

Power Driver ON condition (Pic_ae=1)

Status byte					
Bit 6 (Forward Limit)	**Bit 5 (Reverse Limit)**	**Bit 3 (Power_on)**	**Drive inputs CN2 (Control)**		**LED intensity**
0	1	1	A	FORWARD LIMIT=HIGH REVERSE LIMIT=LOW	High
1	0	1	B	FORWARD LIMIT=LOW REVERCE LIMIT=HIGH	High
0	0	1	C	FORWARD LIMIT=HIGH REVERSE LIMIT=HIGH	High
1	1	1	D	FORWARD LIMIT=LOW REVERSE LIMIT=LOW	High

Power Driver Fault condition (Pic_ae=1)

Diagnostic bits

Status byte					
Bit 6 (Forward Limit)	**Bit 5 (Reverse Limit)**	**Bit 3 (Power_on)**	**Condition**		**LED intensity**
1	0	0	A	STP IN (LATCHED) OR ENCODER ERR	Low
0	1	0	B	MTOR SHORT* OR OVERVOLTAGE	Low
0	0	0	C	OVERCURRENT	Low
1	1	0	D	OVERHEAT	Low

Diagnostic codes

Status byte		Fault code description	Index
51h	A	STP IN	1
		ENCODER ERROR	0
31h	B	MOTOR SHORT* OR OVERVOLTAGE	1
11h or 15h	C	OVERCURRENT	1
71h	D	OVERHEAT	1

Note: LED intensity follows the Brake status:

- brake - engaged <-> low intensity and brake - released <->high intensity.

To restore the normal operation, if Bit 3 (Power_on) is set to 0, Pic_ae must be cycled to 0 and 1. If there are no more fault conditions Power_on will be set to 1.

Power-up and Reset Conditions

On Power-up or reset, the following state is established:

Motor position is reset to zero;
Velocity and acceleration values are set to zero;
All gain parameters and limit values are set to zero;
The servo rate divisor is set to 1 (0.512msec servo rate);
The PWM value is set to zero;
The controller is placed in PWM mode;
The default status data is the status byte only;
The individual address is set to 0x00 and the group address to 0xFF (group leader not set);
Communications are disabled pending a low value of "A in";
The baud rate is set to 19.2 KBPS;
In the *status byte,* the *move_done* and *pos_error* flags will be set and the *current_limit* and *home_in_progress* flags will be clear;
In the *auxiliary status byte,* the *pos_wrap, servo_on, accel_done, slew_done* and *servo_overrun* flags will be cleared.

* *MOTOR SHORT – output to output, output to POWER (+) or output to POWER GND.*

THEORY OF OPERATION

LS-174 can operate in two different modes – normal and advanced. In normal mode it is fully compatible with LS-173 Servo Drive. Advanced mode is intended for coordinated motion control and includes different enhancements.

Relative Zero Position The position counter in the LS-174 can optionally be reset relative to the last position captured in the home position register. This relieves the user from having to calculate goal positions relative to the home position. Please refer to the Command Specification below.

Relative Moves The LS-174 can optionally be commanded to move relative to its current command position. This relieves the user from the additional calculations and bookkeeping otherwise required to implement relative moves. Please refer to the Command Specification below.

Improved Index Latching The LS-174 homing function will trigger on any index pulse longer than 120 nanoseconds. Note, however, that the positions are still only updated every 512 microseconds, and that the most accurate homing will require a motor speed of less than 1 encoder count per 512 microseconds.

Coordinated motion control

LS-174 contains a path point buffer with room for 96 entries. Each entry is a goal position for the motor. When the *Servo Drive* enters its special path mode, it will automatically move from one point to the next at a user selectable rate of either 30 or 60 Hz*. The Servo Drive moves the motor between goal points at a constant velocity such that it always arrives at the next path point in exactly 1/30th or 1/60th of a second. When sets of path points are downloaded into multiple controllers, and then the paths started simultaneously, the individual axes will execute their paths with exact** synchronization.

The path point buffer has room for about 3 seconds worth of motion for a 30 Hz path and about 1.5 seconds for a 60 Hz path. Typically, the host computer downloads the first part of a path to the LS-174 buffers and then starts the path mode. As the buffers becomes depleted, additional path points are dynamically added while the axes are still in motion, until the path is complete. The timing requirements for the host require that it be able to dynamically download new path points before the path point buffers empties completely. With a path point buffer size of 1.5 or 3 seconds, even a non-real time host, such as a PC running Windows, can easily keep up with the task of re-filling the path point buffers as needed.

The actual multi-axis paths, which are downloaded into the LS-174 path point buffers, are calculated by the host computer. In addition to creating the geometry desired (arcs, lines, etc.), the path should be smooth, adhering to the physical acceleration and velocity limits of the motors being controlled. Because the host computer actually creates the paths, any path the user can create can be executed, and paths can involve up to 31 axes. Most typically, coordinated straight-line motions, 2-axis circular motions, or S-curve profiling motions are created.
Note that motions created with the path mode are independent of any acceleration or velocity values loaded using the Load Trajectory command.

Path Accuracy

---

* *In "fast path" mode, the path rates can be selected as either 60 or 120 Hz. The default "slow path" mode is more than adequate for most applications, and requires less communications overhead.*
** *The exactness of the synchronization is subject to crystal frequency accuracy and other timing factors discussed later.*

The path accuracy of the LS-174 Servo Drive is more than adequate for most CNC machine control or robot control applications. For very high speed or very high accuracy applications, however, there are two types of path errors to consider: absolute path errors and timing errors.

Absolute Path Errors

Absolute path accuracy is the accuracy with which a series of calculated path points with straight line segments between them matches the actual curved path desired. For example, a circle, which is approximated by only 5 path points, will form a pentagon rather than a circle. The maximum error between the side of the pentagon and the circle may be quite large. A larger number of path points will produce a smaller error. In general, accuracy of an approximated path will be a function of the number of path points used, and the radius of the curve.

Because LS-174 uses a fixed number of points per second, moving more slowly will result in a more accurate path than moving quickly. Also, a 60 Hz path will be more accurate than a 30 Hz path. The main advantages of using a slower path, however, are that fewer path points need to be calculated, less data needs to be sent to the controllers, and the path point buffer will last twice as long.

The maximum absolute path error can be approximated by the formula:

$$\textbf{Error} = R \times \left(1 - \cos\left(V / \left(2 \times F \times R \right) \right) \right)^{*}$$

where **R** is the radius of the curve (in inches), **V** is the velocity of the motion (in inches/sec), and **F** is the path point frequency (30, 60 or 120 Hz). For example, a one-inch diameter circle with a velocity of 1 inch per second and a path frequency of 30 Hz would have a maximum error of 0.00028 inches. If a frequency of 60 Hz is used, the maximum error drops to 0.000069 inches.

Timing Errors

If the timing of multiple axes is not perfectly synchronized, there will be a deviation from the desired path from the fact that one axis will be ahead or behind in time. The exact deviation will depend on the path geometry.

The first type of timing error results from multiple axes not starting at exactly the same time. When a “start path” command is issued to a group of controllers, they will all start within +/- 0.00025 seconds of one another.

The second type of timing error results from inaccuracies in the frequencies of the oscillators running on each *LS-174* controller. (If all Servo Drives are timed from the same oscillator, this error is zero.) Typical oscillator variations (for the same operating temperature) are about 10 parts per million. Therefore, after running a path for 10 seconds, for example, the timing error would be about +/-0.0001 seconds.
By adding both of these timing errors together, and then multiplying by the path velocity, we get the total distance that one axis can be ahead of another axis. For a 10 second motion, while moving at 1 inch per second, we could have one axis moving ahead of another by at most 0.00035 inches. The actual worst case deviation (moving along a 45 degree angle) will produce an error from the ideal path of 0.00025 inches. Over a total distance of 10 inches traveled, this gives a basic accuracy of ±0.000025 inches per inch of travel. Other examples, of course, will produce different accuracy figures.

---

* *The cosine function should be executed for an angle in radians.*

Note that errors due to timing only accumulate during a coordinated motion and are, in essence, reset with each new move. Therefore, if errors due to timing do become a problem, the paths should be broken up into shorter moves.

COMMAND SPECIFICATION

List of Commands

Command	CMD Code	# Data bytes	Description	While Moving?
Reset position	0x0	0-1	Sets position counter to zero.	No
Set address	0x1	2	Sets the individual and group addresses	Yes
Define status	0x2	1	Defines which data should be sent in every status packet	Yes
Read status	0x3	1	Causes particular status data to be returned just once	Yes
Load trajectory	0x4	1-14	Loads motion trajectory parameters	Maybe*
Start motion	0x5	0	Executes the previously loaded trajectory	Maybe**
Set gain	0x6	14	Sets the PID gains and operating limits	Yes
Stop motor	0x7	1	Stops the motor in one of three manners	Yes
I/O control	0x8	1	Sets the direction and values of the LIMIT pins	Yes
Set home mode	0x9	1	Sets conditions for capturing the home position	Yes
Set baud rate	0xA	1	Sets the baud rate (group command only)	Yes
Clear bits	0xB	0	Clears the sticky status bits	Yes
Save as home	0xC	0	Saves the current position in the home position register	Yes
Nop	0xD	0	Simply causes the defined status data to be returned	Yes
Nop	0xE	0	Simply causes the defined status data to be returned	Yes
Hard reset	0xF	0	Resets the controller to its power-up state.	Yes

**Only allowed while moving if the "start motion now" bit of the trajectory control word is not set or if the "profile mode" bit is set for velocity mode.*

***Only allowed while moving if the previously loaded trajectory has the "profile mode" bit set for velocity mode.*

Command Description

Reset Position

Command value: *0x0*
Number of data bytes: *0 or 1*
Command byte: ***0x00*** *or* ***0x10***
Data bytes:
1.Control byte:
Bit 0: resets current position relative to home position
1-7 not used (clear to 0)

Description:
Resets the 32-bit encoder counter. If bit 0 in the control byte is set, current position will be set to the difference between old current position and the home position. Otherwise, new current position will be zero. Do not issue this command while executing a trapezoidal profile motion.

Set Address

Command value: *0x1*
Number of data bytes: *2*
Command byte: ***0x21***
Data bytes:

1. *Individual address: 0x01-0x7F (initial address 0x00)*
2. *Group Address: 0x80-0xFF (initial value 0xFF)*

Description:

Sets the individual address and group address. Group addresses are always interpreted as being between 0x80 and 0xFF. If a Drive is to be a group leader, clear bit 7 of the desired group address in the second data byte. The module will automatically set bit 7 internally after flagging the Drive as a group leader. (If bit 7 of the second data byte is set, the module will default to being a group member.) The first time this command is issued after power-up or reset, it will also enable communications for the next Drive in the network chain by lowering the it's "A out" signal.

Define Status

Command value: *0x2*
Number of data bytes: *1*
Command byte: ***0x12***
Data bytes:
1. Status items: (default: 0x00)

Bit 0: *send position (4 bytes)*
1: *send A/D value (1 byte)*
2: *send actual velocity (2 bytes - no fractional component)*
3: *send auxiliary status byte (1 byte)*
4: *send home position (4 bytes)*
5: *send device ID and version number (2 bytes)*
(motor controller device ID = 0, version number =70 or higher)
6: *send current position error (2 bytes)*
7: *send number of points in the path buffer*[*]

Description:

Defines what additional data will be sent in the status packet along with the status byte. Setting bits in the command's data byte will cause the corresponding additional data bytes to be sent after the status byte. The status data will always be sent in the order listed. For example if bits 0 and 3 are set, the status packet will consist of the status byte followed by four bytes of position data, followed by the aux. status byte, followed by the checksum. The status packet returned in response to this command will include the additional data bytes specified. On power-up or reset, the default status packet will include only the status byte and the checksum byte.
Note: The actual velocity is a positive number when moving in reverse direction and a negative number when moving in forward direction.

[*] *In advanced mode only.*

Read Status

Command value: *0x3*
Number of data bytes: *1*
Command byte: ***0x13***
Data bytes:
1.Status items:

Bit 0: *send position (4 bytes)*
1: *send A/D value (1 byte)*
2: *send actual velocity (2 bytes - no fractional component)*
3: *send auxiliary status byte (1 byte)*
4: *send home position (4 bytes)*
5: *send device ID, version number (2 bytes)*
(Motor controller device ID = 0, version number = 70 or higher)
6: *send current position error (2 bytes)*
7: *send number of points in the path buffer**

Description:

This is a non-permanent version of the *Define Status* command. The status packet returned in response to this command will incorporate the data bytes specified, but subsequent status packets will include only the data bytes previously specified with the *Define Status* command.
Note: The actual velocity is a positive number when moving in reverse direction and a negative number when moving in forward direction.
** In advanced mode only.*

Load Trajectory

Command value: *0x4*
Number of data bytes: *n = 1-14*
Command byte: ***0xn4***
Data bytes:
1.Control byte:

Bit 0: *load position data (n = n + 4 bytes)*
1: *load velocity data (n = n + 4 bytes)*
2: *load acceleration data (n = n + 4 bytes)*
3: *load PWM value (n = n + 1 bytes)*
4: *servo mode - 0 = PWM mode, 1 = position servo*
5: *profile mode - 0 = trapezoidal profile, 1 = velocity profile*
6: *in velocity/PWM mode* *direction flag* *0 = FWD, 1 = REV*
in trapezoidal mode *0 = absolute, 1 = relative**
7: *start motion now*

** In advanced mode only.*

Description:

All motion parameters are set with this command. Setting one of the first four bits in the control byte will require additional data bytes to be sent (as indicated) in the order listed. The position data (range* +/- 0x7FFFFFFF) is only used as the goal position in trapezoidal profile mode. The velocity data (range 0x00000000 to 0x7FFFFFFF) is used as the goal velocity in velocity profile mode or as the maximum velocity in trapezoidal profile mode. Velocity is given in encoder counts per servo tick, multiplied by 65536. The acceleration data (range 0x00000000 to 0x7FFFFFFF) is used in both trapezoidal and velocity profile mode. Acceleration is given in encoder counts per servo tick per servo tick, multiplied by 65536. The PWM value (range 0x00 - 0xFF), used only when the position servo is not operating, sends a raw PWM value directly to the amplifier. The

* *While the position may range from -0x7FFFFFFF to +0x7FFFFFFF, the goal position should not differ from the current position by more then 0x7FFFFFFF.*

PWM value is reset to 0 internally on any condition, which automatically disables the position servo.
Bit 4 of the control byte specifies whether the position servo should be used or if the PWM mode should be entered. Bit 5 specifies whether a trapezoidal profile motion should be initiated or if the velocity profiler is used. Trapezoidal profile motions should only be initialized when the motor velocity is 0. (Bit 0 of the status byte indicates when a trapezoidal profile motion has been completed, or in velocity mode, when the command velocity has been reached.) Bit 6 indicates the velocity or PWM direction. In trapezoidal profile mode, this bit indicates whether position data is absolute (if bit 6 = 0) or relative to the current position. This feature is available in advanced mode only. If bit 7 is set, the command will be executed immediately. If bit 7 is clear, the command data will be buffered and it will be executed when the *Start Motion* command is issued. For example to load only new position data and acceleration data but not to start the motion yet, the command byte would be 0x94, the control byte would be 0x15, followed by 4 bytes of position data (least significant byte first), followed by 4 bytes of acceleration data.
If in the middle of a trapezoidal position move, a new *Load Trajectory* command is issued with new position data downloaded, new position data will be used as a relative offset to modify the goal position. For example, if in the middle of a move to position 50,000, a new *Load Trajectory* command with new position data of 10,000 is loaded, the motor will stop at final position of 60,000. The relative offset can be either positive or negative. The new *Load Trajectory* command must be issued while the motor is running at a constant velocity – issuing the command while accelerating or decelerating will cause a position error to occur. If more than one *Load Trajectory* is issued before the end of move, the goal position will be modified by the sum of relative offsets.

Start Motion

Command value: *0x5*
Number of data bytes: *0*
Command byte: ***0x05***

Description:
Causes the trajectory information loaded with the most recent Load Trajectory command to execute. This is useful for loading several Drives with trajectory information and then starting them simultaneously with a group command.

Set Gain

Command value: *0x6*
Number of data bytes: *14*
Command byte: ***0xE6***
Data bytes:

1,2. Position gain KP (0 - 0x7FFF)
3,4. Velocity gain KD (0 - 0x7FFF)
5,6 Integral gain KI (0 - 0x7FFF)
7,8. Integration limit IL (0 - 0x7FFF)
9. Output limit OL (0 - 0xFF) (typically recommended 0xFA)
10. Current limit CL (0 - 0xFF) (only odd values)
11,12 Position error limit EL (0 - 0x3FFF)
13. Servo rate divisor SR (1 - 0xFF)
14. Amplifier deadband compensation (0 - 0xFF) (typical value is between 0x03 and 0x05)

Description:
Sets all parameters and limits governing the behavior of the position servo. KP, KD, KI and IL are PID filter parameters. OL limits the maximal PWM output value to 0<PWM≤OL in position servo

modes. In PWM mode OL is ignored. CL is used for motor current limitation (refer to "Motor current monitoring" in "Safety Features" for detailed information). Setting CL=0 effectively disables current limiting. The position error limit (EL) will cause the position servo to be disabled should the position error grow beyond the limit. The servo rate divisor sets the servo tick time to be a multiple of 0.512 msec (1.953 KHz). For example SR=3 gives a servo rate of 651 Hz. The servo tick rate is also used as the profiling timebase, although command processing and current limiting are always performed at the maximum tick rate. Sometimes it is necessary to compensate the deadband region around zero PWM output exhibited by some amplifier/motor combinations. The deadband compensation value will be added to the magnitude of the PWM output to force the amplifier into its active region.

Stop Motor

Command value: *0x7*
Number of data bytes: *1 or 5*
Command byte: ***0x17 or 0x57***
Data bytes:
1. Stop control byte
Bit 0: Pic_ae (Power Driver enable)
1: Turn motor off
2: Stop abruptly
3: Stop smoothly
4: Stop here
5: Enable advanced features
6-7: Clear all to 0
2-5. Stopping position (only required if bit 4 above is set)

Description:

Stops the motor in the specified manner. If bit 0 of the Stop Control Byte is set, Power Driver will be enabled. If bit 0 is cleared Power Driver will be disabled, regardless of the state of the other bits. Pic_ae also controls the meaning of bit 3 (Power_on), bit 5 (Reverse Limit), and bit 6 (Forward Limit) of status byte (refer to "Status Bits" section of "Safety Features" in this document). If bit 1 is set, the position servo will be disabled, the PWM output value will be set to 0, and bits 2, 3 and 4 are ignored. If bit 2 is set, the current command velocity and the goal velocity will be set to 0, the position servo will be enabled, and velocity mode will be entered. If the velocity servo was previously disabled, the motor will simply start servoing to its current position. If the motor was previously moving in one of the profiling modes, it will stop moving abruptly and servo to its current position. This stopping mode should only be used as an emergency stop where the motor position needs to be maintained. Setting bit 3 enters a more graceful stop mode - this sets the goal velocity to 0 and enters velocity mode, causing the motor to decelerate to a stop at the current acceleration rate. If bit 4 is set, the motor will move to the specified stopping position abruptly with no profiling. This mode can be used to cause the motor to track a continuous string of command positions. Note that if the stopping position is too far from the current position, a position error will be generated. Only one of the bits 1, 2, 3 or 4 should be set at the same time. The *Stop Motor* command must be issued initially to set Pic_ae before other motion commands are issued.

Bit 5 enables advanced features of the LS-174. Advanced features will stay enabled regardless of subsequent stop commands, until the drive is reset.

I/O Control

Command value: *0x8*
Number of data bytes: *1*
Command byte: ***0x18***
Data bytes:
1. I/O control byte
Bit 0: Output value of Reverse Limit (not used)
1: Output value of Forward Limit (not used)
2: Direction of Reverse Limit (must be set to 1 = input)
3: Direction of Forward Limit (must be set to 1 = input)
4-5: Unused (set to 0)
6: Fast path mode (0 = 30/60 Hz, 1 = 60/120 Hz)
7: Unused (set to 0)

Description:
Bit 6 controls whether the drive is in slow path mode (bit 6 = 0) or fast path mode when the drive is in advanced mode. After power-up Reverse Limit and Forward Limit are inputs.

Set Homing Mode

Command value: *0x9*
Number of data bytes: *1*
Command byte: ***0x19***
Data bytes:
1. Homing control byte
Bit 0: Capture home position on change of Reverse Limit
1: Capture home position on change of Forward Limit
2 Turn motor off on home
3: Capture home on change of Index
4: Stop abruptly on home
5: Stop smoothly on home
6: Capture home position when an excess position error occurs
7: Capture home position when current limiting occurs

Description:
Causes the Drive to monitor the specified conditions and capture the home position when any of the flagged conditions occur. The home_in_progress bit in the status byte is set when this command is issued and it is then lowered the home position has been found. Setting one (and only one) of bits 2, 4 or 5 will cause the motor to stop automatically in the specified manner once the home condition has been triggered. This feature can also be used as a safety shutoff.
Note: For homing with Index signal, use low velocities, which ensure the time of the Index pulse is at least one servo tick (0.512 msec). The maximum theoretical homing velocity is 65536 (one encoder count per servo tick). Depending of motor vibrations, the homing velocity should be less than 65536. A recommended homing velocity is 16384 (0.25 encoder counts per servo tick).

Set Baud Rate

Command value:	*0xA*	**sample values**:	
Number of data bytes:	*1*	9600	BRD = 0x81
Command byte:	***0x1A***	19200	BRD = 0x3F
Data bytes:		57600	BRD = 0x14
1. Baud rate divisor,	*BRD*	115200	BRD = 0x0A

Description:

Sets the communication baud rate. All drives on the network must have their baud rates changed at the same time; therefore this command should only be issued to a group including all of the controllers on the network. A status packet returned from this command would be at the new baud rate, so typically (unless the host's baud rate can be accurately synchronized) there should be no group leader when this command is issued.

Clear Sticky Bits

Command value:	*0xB*
Number of data bytes:	*0*
Command byte:	***0x0B***

Description:

The overcurrent and position error bits in the status byte and the position wrap and servo timer overrun bits in the auxiliary status byte will stay set unless cleared explicitly with this command.

Save Current Position as Home

Command value:	*0xC*
Number of data bytes:	*0*
Command byte:	***0x0C***

Description:

Causes the current position to be saved as the home position. This command is typically issued to a group of controllers to cause their current positions to be stored synchronously. The stored positions can then be read individually by reading the home position

Add path points

Command value:	*0xD*	
Number of data bytes:	*n = 0, 2, 4, 6, 8, 10, 12 or 14*	
Command byte:	***0xnD***	
Data bytes:		
1, 2:	*Incremental data for path point 1 (n >= 2)*	
3, 4:	*Incremental data for path point 2 (n >= 4)*	
…		
13, 14:	*Incremental data for path point 7 (n = 0xE)*	*or*
none	*Starts execution of path point mode (n = 0)*	

Description:

The Add Path Points command allow the user to add path points to the LS-174 internal path point buffer. Up to 7 path points may be added at with a single command, and multiple commands may be issued to load a total of 96 path points (Bit 7 of the Read Status or the Define Status control byte can be used to have the number of path points in the buffer returned in the status packet). Path point data is specified as a 16-bit integer, with 14 of those bits specifying the *absolute*

distance from the previous path point to the current path point[*]. The additional 2 bits are used to specify: 1) the direction of the incremental position data (*i.e.*, whether the path point is in front of or behind the previous point), and 2) if it should take 1/30th second or 1/60th second to get to the path point from the previous path point. Normally, just one timing mode is used in a path, but the timing can be mixed. The data format for each pair of data bytes is as follows:

30 Hz Path:	P_{13} P_{12} P_{11} P_{10} P_9 P_8 P_7 P_6 *most significant byte*	P_5 P_4 P_3 P_2 P_1 P_0 F D *least significant byte*
60 Hz Path:	P_{12} P_{11} P_{10} P_9 P_8 P_7 P_6 P_5 *most significant byte*	P_4 P_3 P_2 P_1 P_0 0 F D *least significant byte*

where P_0 - P_{13} are the 14 bits of incremental position data, F is the path frequency (0 = 60 Hz, 1 = 30 Hz) and D is the direction (0 = forward, 1 = reverse). Note that for a 60 Hz path, the position data bits are shifted to the left, and bit 2 is always zero. As with all other types of multi-byte data, the least significant byte is always sent first.

To actually start execution of the path in the path point buffer, an Add Path Points command is issued with no path point data included. To start several LS-174 controllers simultaneously, the Add Path Points command (with no path point data) should be sent to the group address of the group containing all the LS-174 controllers involved.

If "fast path" mode has been selected using the I/O Control command, bit F will be set to 0 for 120 Hz or 1 for 60 Hz, and the path point data will have the following format:

60 Hz Path (fast path mode):	P_{12} P_{11} P_{10} P_9 P_8 P_7 P_6 P_5 *most significant byte*	P_4 P_3 P_2 P_1 P_0 0 F D *least significant byte*
120 Hz Path (fast path mode):	P_{11} P_{10} P_9 P_8 P_7 P_6 $P_5$$P_4$ *most significant byte*	P_3 P_2 P_1 P_0 0 0 F D *least significant byte*

No Operation

Command value:	*0xE*
Number of data bytes:	*0*
Command byte:	***0x0E***

Description:
Does nothing except cause a status packet with the currently defined status data to be returned.

Hard Reset

Command value:	*0xF*
Number of data bytes:	*0*
Command byte:	***0x0F***

Description:
Resets the control module to its power-up state. No status will be returned. Typically, this command is issued to all the modules on the network, although if the baud rate is set at the default, it is possible to reset and re-initialize the addresses of a contiguous sub-chain of modules.

[*] *By using incremental position data rather than absolute position data, the amount of path data, which will fit in the LS-174 buffer is doubled.*

STATUS BYTE AND AUXILIARY STATUS BYTE DEFINITIONS

Status Byte

Bit	Name	Definition
0	Move_done	Clear when in the middle of a trapezoidal profile move or in velocity mode, when accelerating from one velocity to the next. This bit is set otherwise, including while the position servo is disabled
1	Cksum_error	Set if there was a checksum error in the just received command packet
2	Current_limit	Set if current limiting has exceeded (refer to "Motor Current Monitoring" section in this document). Must be cleared by user with *Clear Sticky Bits* command
3	Power_on/diag. bit	Refer to "Status Bits and LED" section in this document
4	Pos_error	Set if the position error has exceeded the position error limit. It is also set whenever the position servo is disabled. Must be cleared by user with *Clear Sticky Bits* command
5	Reverse Limit/ diag. bit	Reverse Limit or diagnostic bit (refer to "Status Bits and LED" section in this document).
6	Forward Limit/ diag. bit	Forward Limit or diagnostic bit (refer to "Status Bits and LED" section in this document).
7	Home_in_progress	Set while searching for a home position. Reset to zero once the home position has been captured

Auxiliary Status Byte

Bit	Name	Definition
0	Index/diag. bit	Compliment of the value of the index input or diagnostic bit (refer to "Status Bits and LED" section in this document).
1	Pos_wrap	Set if the 32-bit position counter wraps around. Must be cleared with the *Clear Sticky Bits* command
2	Servo_on	Set if the position servo is enabled, clear otherwise
3	Accel_done	Set when the initial acceleration phase of a trapezoidal profile move is completed. Cleared when the next move is started
4	Slew_done	Set when the slew portion of a trapezoidal profile move is complete. Cleared when the next move is started
5	Servo_overrun	At the highest baud rate and servo rate, certain combinations of calculations may cause the servo, profiling, and command processing to take longer than 0.512 msec, in which case, this bit will be set. This is typically not serious, only periodically introducing a small fraction of a millisecond delay to the servo tick time. Cleared with the *Clear Sticky Bits* command
6	Path mode	Set when the drive is currently executing a path. Cleared when buffer is emptied or Stop Motor or Load Trajectory command is sent.

INITIALIZING PROCEDURE AND PROGRAMMING EXAMPLES FOR SERVO DRIVES

To ensure a proper operation of all Servo drives connected to the network, the following initializing steps should be executed:

1. Reset all modules with *Hard Reset* command.
2. Set the addresses for all connected drives.
3. Set the individual gains (KP, KD, KI, IL, OL, CL, EL, SR and DB). Minimal requirements are: KP <> 0, EL <> 0 and SR <> 0.
4. Use *Load trajectory* command to set the target position, velocity acceleration with start motion now in trapezoidal mode. Minimal requirements are acceleration <> 0 and target position = 0. This command does not start any motion. It is necessary to initialize internal registers of the module.
5. Close the servo loop by using *Stop Motor* command (Pic_ae=1 and Stop abruptly=1).

Understanding the Serial Communication with Servo drives

The Serial Communication with Servo drives is strictly master-slave and matches repeatedly two elements:

- Sending a command to the specified drive's address;
- Receiving answer to the sent command – Status Byte(s).

Note: *During the communication all bytes are sent with LSB first.*

Commands

There are 16 commands managing Servo drives (refer to Command Description). Each command as shown in the following two tables includes header, address, command, data bytes and one checksum byte. Checksum does not include header byte.

Structure of *Read Status* command

Byte 1	Byte 2	Byte 3		Byte 4	Byte 5
Header	Address (Individual or Group)	Command Code		Data Byte	CheckSum = Byte 2 + Byte 3 + Data Byte
		High 4 bits No. of data bytes	Low 4 bits command code		
AA	01	1	3	01	15

Examples

Cmd. Bytes	Byte 1	Byte 2	Byte 3	Byte 4 – N	Byte N+1
Command	Header	Address	Cmd. Code	Data Byte(s)	Checksum
Reset position	AA	01	0 0		01
Define status	AA	05	1 2	05	1C
Set address	AA	01	2 1	07 FF	21
Load trajectory	AA	01	5 4	91 00 28 00 00	0E
Set gain	AA	01	E 6	64 00 00 04 00 00 00 00 FF 00 00 08 01 00	57

Status Data

The structure of the returned status information depends on *Define Status* or *Read Status* commands (refer to Command Description). By default only the Status byte and Checksum are returned to the host.

Examples

Byte 1	Optional Bytes 0-16	CheckSum
Status Byte	Additional Status Bytes as position, velocity, home position, A/D auxiliary byte, version and position error.	CheckSum = Byte 1+ Optional Bytes
09	no additional status bytes requested	09
09	00 28 00 00 – four additional status bytes	31

Addressing

Each drive in the daisy-chained network has two addresses:

- Individual - for individual control of each drive. Its range is from 01h to 7Fh.
- Group - for simultaneous control of all group members by sending a single command to their group address. It is in the range of 80h to FFh.

Both these addresses have to be set during the initialization process.

The group may have Group leader responsible to send status data. Its address is:

Group leader address = Group address - 80h.

If there is no group leader - no status data will be send after a group command.

Set Baud Rate command must be sent only as a group command with no group leader, otherwise communication problems may occur.

***Set Address* command format**

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Header	Preset Address	Command code	Individual Address	Group Address	Checksum
AA	00	21	01	FF	21

Setting the Addresses

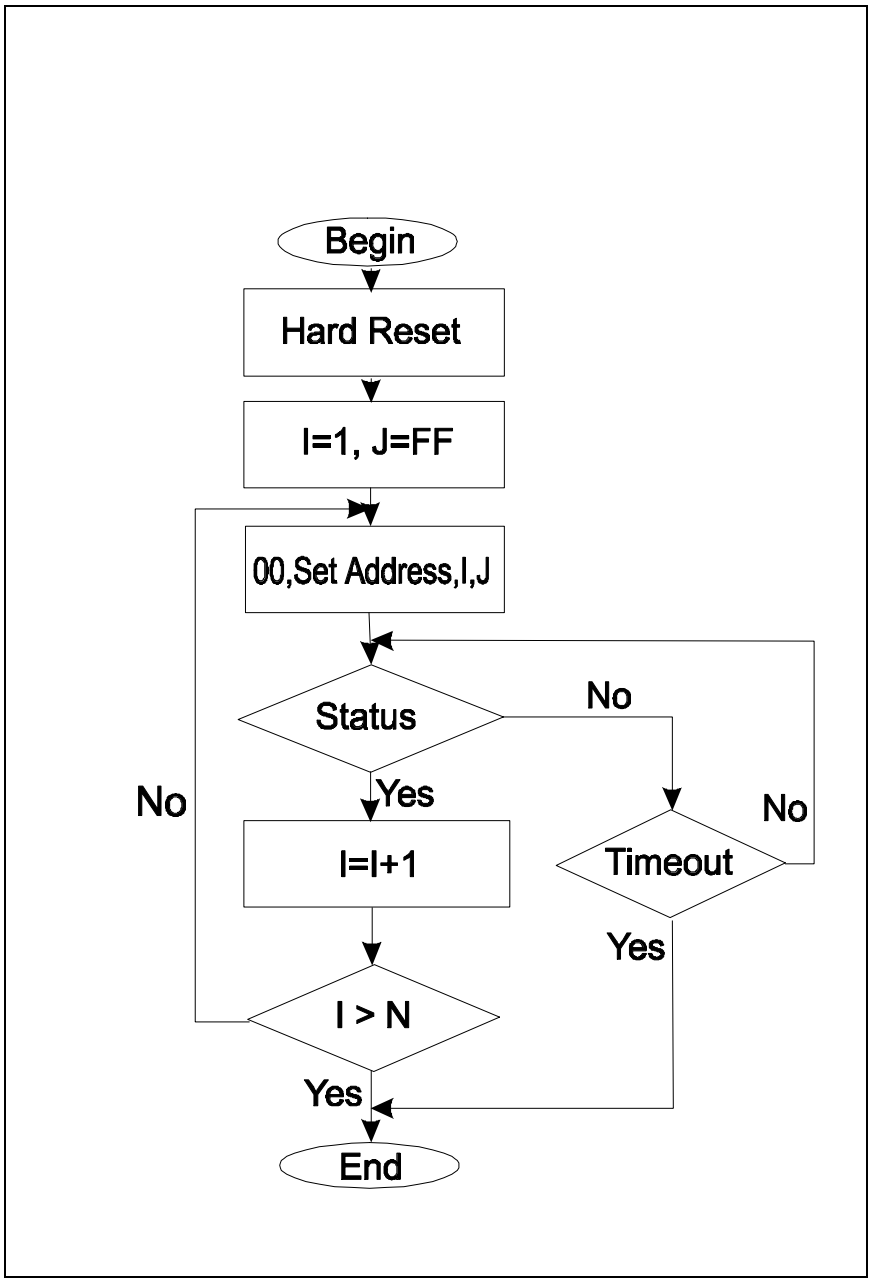
After power-up and *Hard Reset* command all drives have their address set to 00h and only the first drive (starting from the host) has its communication enabled. Consecutive *Set Address* commands are sent to address 00h until all drives are addressed. This procedure can be executed once after *Hard Reset*. The table below shows the steps to address 3-drives network.

Example of sequential addressing for three Servo drives

step	Command	Set address Hexadecimal Code	Drive 1		Drive 2		Drive 3	
			Individual address	Group address	Individual address	Group address	Individual address	Group address
0	Power-up							
1	Hard Reset	AA FF 0F 0E	address=00 communication *enabled*		address=00 communication *disabled*		address=00 communication *disabled*	
2	Set Address Drive1 = 01	AA 00 21 01 FF 21	01	FF	address=00 communication *enabled*		address=00 communication *disabled*	
3	Set Address Drive2 = 02	AA 00 21 02 FF 22	01	FF	02	FF	address=00 communication *enabled*	
4	Set Address Drive3 = 03	AA 00 21 03 FF 23	01	FF	02	FF	03	FF

Note: Before start addressing *Hard Reset* command must be issued.

The flowchart shows the addressing procedure of N drives network. There is no group leader and the group address is FF.



I - Individual Address; *J* - Group Address = FF;
Status - Status Data sent to the Host; *Timeout* - Greater than one servo circle.

Examples of Managing Two Servo-Drives

1 – Resets all modules with group command.
2 and # 3 - Set the addresses of drives 1 and 2.
4 and # 6 - Set PID parameters of drives 1 and 2.
6 and # 7 - Starts motion in trapezoidal mode with target position=0, velocity=0, acceleration=1 and PWM=0.
8 and # 9 - Close servo loops of drives 1 and 2. Initialization is complete at this point.
10 and # 10 - Load trajectories (positions, velocities and accelerations) for drives 1 and 2.
12 and # 13 - Load and execute new trajectory for drive 1.
14 and # 15 - Read additional status bytes from drives 1 and 2.
16, # 17 and #18 - Load new trajectories for drives 1 and 2 and execute them with one command sent to the drives' group address.

Examples

#	Hexadecimal code of command	Comments
1	AA FF 0F 0E	*Hard Reset*
2	AA 00 21 01 FF 21	*Set Address* 01h for drive 1. Group address=FFh.
3	AA 00 21 02 FF 22	*Set Address* 02h for drive 2. Group address=FFh.
4	AA 01 E6 64 00 00 04 00 00 00 00 FF 00 00 08 01 00 57	*Set Gains* of drive 1 – defines PID parameters: KP=64h, KI=400h, KI=00h, IL=00h, OL=FFh, CL=00h, EL=800h, SR=01h, DC=00h.
5	AA 02 E6 64 00 00 04 00 00 00 00 FF 00 00 08 01 00 58	*Set Gains* of drive 2 – defines PID parameters: KP=64h, KI=400h, KI=00h, IL=00h, OL=FFh, CL=00h, EL=800h, SR=01h, DC=00h.
6	AA 01 E4 9F 00 00 00 00 00 00 00 00 01 00 00 00 00 85	*Load trajectory* for drive 1 – target position=0, velocity=0, acceleration=1, PWM=0 and start motion now
7	AA 02 E4 9F 00 00 00 00 00 00 00 00 01 00 00 00 00 86	*Load trajectory* for drive 2 – target position=0, velocity=0, acceleration=1, PWM=0 and start motion now
8	AA 01 17 05 1D	*Stop Motor* - closes servo loop of drive 1 with *Power Driver enable* and *Stop Abruptly* in Command byte.
9	AA 02 17 05 1E	*Stop Motor* - closes servo loop of drive 2 with *Power Driver enable* and *Stop Abruptly* in Command byte.
10	AA 01 E4 9F 00 00 00 00 00 80 01 00 64 00 00 00 00 69	*Load Trajectory* of drive 1 with Pos=0000h, Vel=18000h, Acc=6400h, PWM=00h, servo mode=1.
11	AA 02 E4 9F 00 00 00 00 00 80 01 00 64 00 00 00 00 6A	*Load Trajectory* of drive 2 with Pos=0000h, Vel=18000h, Acc=6400h, PWM=00h, servo mode=1.
12	AA 01 54 11 00 28 00 00 8E	*Load Trajectory* of drive 1 with new position=2800h.
13	AA 01 05 06	*Start Motion* - executes previously loaded trajectory.
14	AA 01 13 05 19	*Read Status* from drive 1 (plus position and velocity).
15	AA 02 13 05 1A	*Read Status* from drive 2 (plus position and velocity).
16	AA 01 54 11 20 4E 00 00 D4	*Load Trajectory* of drive 1 with new position=4E20h.
17	AA 02 54 11 E0 B1 FF FF F6	*Load Trajectory* of drive 2 with new position=FFFFB1E0h (-4E20h).
18	AA FF 05 04	*Start Motion* – executes previously loaded trajectories. The command is sent to the drives' group address FFh.

Procedure Initialize

AA FF 0F 0E	Hard reset
AA 00 21 01 FF 21	Set address
AA 00 21 02 FF 22	Search for more modules until no response received
AA 01 13 20 34	Read Device ID and Version number
AA 01 13 FF 13	Read all status data
AA 01 E6 64 00 00 04 00 00 00 00 FF 00 00 08 01 00 57	Set Gain parameters
AA 01 E4 9F 00 00 00 00 00 00 00 00 01 00 00 00 00 85	Set Trajectory parameters
AA 01 17 05 1D	Close servo loop

Procedure FindHomePosition

AA 01 E6 C8 00 20 03 46 00 28 00 FF 00 40 1F 01 00 9F	Set gain parameters: KP=200, KD=800, KI=70, IL=40, Output limit=255, current limit =0, Position error limit=8000, Servo rate divisor=1 amplifier deadband compensation=0
AA 01 17 09 21	Close the servo loop (Stop smoothly and amplifier enable)
AA 01 94 37 25 06 01 00 58 01 00 00 51	Load trajectory: Velocity mode, Forward direction, Velocity=1 round per second (67109 programmed velocity for 500 line encoder), Acceleration = 10 round per second^2 (344 programmed acceleration for 500 line encoder)
AA 01 19 12 2C	Set home mode - capture home position on change of Forward Limit and stop abruptly
AA 01 05 06	Start motion
wait while home_in_progress bit=1	Home position is found on change of Forward Limit
AA 01 19 18 32	Set home mode - capture home position on change of Index and stop abruptly
AA 01 94 77 25 06 01 00 58 01 00 00 91	Load trajectory: Velocity mode, Reverse direction
AA 01 05 06	Start motion
wait while home_in_progress bit=1	Home position is found on change of Index

Calculation of programmed velocity and acceleration for servo rate divisor = 1:
Vel = (encoder counts per revolution) x (number of revolutions per second) x 33.554432
Acc = (encoder counts per revolution) x (number of revolutions per second^2) x 0.017179869184

For this example:

Vel	= 2000 x 1	x 33.554432	= 67109	= 00010625h
Acc	= 2000 x 10	x 0.017179869184	= 344	= 00000158h

Path Mode Example

The table below shows all the data for a smooth trapezoidal path for a single motor moving a carriage from a position of 0.0 to 2.0 inches. A path frequency of 30 Hz is used. (The path point numbers in boldface indicate the periods of acceleration or deceleration.) The last two columns of the table contain (in hex format) the exact position data with the frequency and direction bits set. Assuming that the address for the corresponding motor controller is 0x01, the command string to download the first seven path points would be as follows:

```
0xAA         Header byte
0x01         Address byte
0xED         Add Path Point Command byte (14 bytes of data)
0x5A 0x00    Path point 1
0xB6 0x00    Path point 2
0x0A 0x01    Path point 3
0x66 0x01    Path point 4
0xBE 0x01    Path point 5
0x1A 0x02    Path point 6
0x6E 0x02    Path point 7
0xBB         8-bit Checksum (does not include header)
```

Note that less than 7 path points can be added, as long as the upper nibble of the command byte, indicating the number of additional data bytes, is adjusted accordingly. After several path points have been added, you can actually start the path execution by issuing the command string:

```
0xAA         Header byte
0x01         Address byte
0x0D         Add Path Point Command byte (no additional data)
0x0E         8-bit Checksum (does not include header)
```

The path will continue to execute as long as new path points are added before the *Servo Drive* reaches the last point added.

Path Mode Example: Single Axis Trapezoidal Motion

Path Length (inches)	2.00
Encoder counts per inch	10000.00
Maximum Velocity (in/sec)	1.00
Acceleration (in/sec/sec)	2.00
Path Frequency (Hz)	30.00
Tick Time	0.033
Maximum Velocity (counts/tick)	333.333
Acceleration (counts/tick/tick)	22.22
Starting Position (counts)	0.00
Goal Position (counts)	20000.00

Path Point	*Current Velocity (counts/ tick)*	*Position (counts)*	*Integer Position (counts)*	*Distance from Prev. Point (counts)*	*Most Significant Byte (binary)*	*Least Significant Byte (binary)*	*Most Significant Byte (hex)*	*Least Significant Byte (hex)*
(start)	0.00	0	0					
1	**22.22**	22.22	22	22	00000000	01011010	00	5A
2	**44.44**	66.67	67	45	00000000	10110110	00	B6
3	**66.67**	133.33	133	66	00000001	00001010	01	0A
4	**88.89**	222.22	222	89	00000001	01100110	01	66
5	**111.11**	333.33	333	111	00000001	10111110	01	BE
6	**133.33**	466.67	467	134	00000010	00011010	02	1A
7	**155.56**	622.22	622	155	00000010	01101110	02	6E
8	**177.78**	800.00	800	178	00000010	11001010	02	CA
9	**200.00**	1000.00	1000	200	00000011	00100010	03	22
10	**222.22**	1222.22	1222	222	00000011	01111010	03	7A
11	**244.44**	1466.67	1467	245	00000011	11010110	03	D6
12	**266.67**	1733.33	1733	266	00000100	00101010	04	2A
13	**288.89**	2022.22	2022	289	00000100	10000110	04	86
14	**311.11**	2333.33	2333	311	00000100	11011110	04	DE
15	**333.33**	2666.67	2667	334	00000101	00111010	05	3A
16	333.33	3000.00	3000	333	00000101	00110110	05	36
17	333.33	3333.33	3333	333	00000101	00110110	05	36
18	333.33	3666.67	3667	334	00000101	00111010	05	3A
19	333.33	4000.00	4000	333	00000101	00110110	05	36
20	333.33	4333.33	4333	333	00000101	00110110	05	36
21	333.33	4666.67	4667	334	00000101	00111010	05	3A
22	333.33	5000.00	5000	333	00000101	00110110	05	36
23	333.33	5333.33	5333	333	00000101	00110110	05	36
24	333.33	5666.67	5667	334	00000101	00111010	05	3A
25	333.33	6000.00	6000	333	00000101	00110110	05	36
26	333.33	6333.33	6333	333	00000101	00110110	05	36
27	333.33	6666.67	6667	334	00000101	00111010	05	3A
28	333.33	7000.00	7000	333	00000101	00110110	05	36
29	333.33	7333.33	7333	333	00000101	00110110	05	36
30	333.33	7666.67	7667	334	00000101	00111010	05	3A
31	333.33	8000.00	8000	333	00000101	00110110	05	36
32	333.33	8333.33	8333	333	00000101	00110110	05	36
33	333.33	8666.67	8667	334	00000101	00111010	05	3A
34	333.33	9000.00	9000	333	00000101	00110110	05	36
35	333.33	9333.33	9333	333	00000101	00110110	05	36
36	333.33	9666.67	9667	334	00000101	00111010	05	3A
37	333.33	10000.00	10000	333	00000101	00110110	05	36
38	333.33	10333.33	10333	333	00000101	00110110	05	36
39	333.33	10666.67	10667	334	00000101	00111010	05	3A
40	333.33	11000.00	11000	333	00000101	00110110	05	36
41	333.33	11333.33	11333	333	00000101	00110110	05	36
42	333.33	11666.67	11667	334	00000101	00111010	05	3A

43	333.33	12000.00	12000	333	00000101	00110110	05	36
44	333.33	12333.33	12333	333	00000101	00110110	05	36
45	333.33	12666.67	12667	334	00000101	00111010	05	3A
46	333.33	13000.00	13000	333	00000101	00110110	05	36
47	333.33	13333.33	13333	333	00000101	00110110	05	36
48	333.33	13666.67	13667	334	00000101	00111010	05	3A
49	333.33	14000.00	14000	333	00000101	00110110	05	36
50	333.33	14333.33	14333	333	00000101	00110110	05	36
51	333.33	14666.67	14667	334	00000101	00111010	05	3A
52	333.33	15000.00	15000	333	00000101	00110110	05	36
53	333.33	15333.33	15333	333	00000101	00110110	05	36
54	333.33	15666.67	15667	334	00000101	00111010	05	3A
55	333.33	16000.00	16000	333	00000101	00110110	05	36
56	333.33	16333.33	16333	333	00000101	00110110	05	36
57	333.33	16666.67	16667	334	00000101	00111010	05	3A
58	333.33	17000.00	17000	333	00000101	00110110	05	36
59	333.33	17333.33	17333	333	00000101	00110110	05	36
60	333.33	17666.67	17667	334	00000101	00111010	05	3A
61	**311.11**	17977.78	17978	311	00000100	11011110	04	DE
62	**288.89**	18266.67	18267	289	00000100	10000110	04	86
63	**266.67**	18533.33	18533	266	00000100	00101010	04	2A
64	**244.44**	18777.78	18778	245	00000011	11010110	03	D6
65	**222.22**	19000.00	19000	222	00000011	01111010	03	7A
66	**200.00**	19200.00	19200	200	00000011	00100010	03	22
67	**177.78**	19377.78	19378	178	00000010	11001010	02	CA
68	**155.56**	19533.33	19533	155	00000010	01101110	02	6E
69	**133.33**	19666.67	19667	134	00000010	00011010	02	1A
70	**111.11**	19777.78	19778	111	00000001	10111110	01	BE
71	**88.89**	19866.67	19867	89	00000001	01100110	01	66
72	**66.67**	19933.33	19933	66	00000001	00001010	01	0A
73	**44.44**	19977.78	19978	45	00000000	10110110	00	B6
74	**22.22**	20000.00	20000	22	00000000	01011010	00	5A
75	**0.00**	20000.00	20000	0	00000000	00000010	00	02

Data for a Trapezoidal Motion