Soloist™

Position Controller and Servo Amplifier

Single axis digital servo controller with integral power supply and amplifier

Advanced software architecture shortens customer development time; use C#, VB.Net, LabVIEW® and MATLAB® combined with our full IDE and multitasking operating system

Host-mode operation allows you to send commands with your PC via Ethernet or USB for immediate execution

Ethernet or USB permits networked Soloists for multi-axis sequenced motion and IO passing

Ideal for simple applications with minimal setup or complex applications that use the full flexibility and scalability

Positioning control for brushless, DC brush-type, or stepping motors

Available in models up to 150 A peak current

Linear amplifier (HLe/CL) for low noise, ultra-high-performance applications

Allen-Bradley EtherNet/IP™ interface provides full integration with the Soloist; program the Soloist directly from RSLogix™ 5000

Introduction

Aerotech’s Soloist™ CP is a single-axis servo controller that combines a power supply, amplifier, and position controller in a single package. The Soloist can control up to five tasks simultaneously, as well as handle variables and manage IO, making it well-suited for demanding production applications. The Soloist has high-speed position latch inputs and advanced data logging capabilities, making it ideal for laboratory and test instrument applications. The advanced software architecture shortens customer development time, while including support for C#, VB.Net, LabVIEW®, and MATLAB®, combined with our full IDE and multitasking operating system. Host-mode operation allows you to send commands with your PC via Ethernet or USB for immediate execution, while Ethernet or USB also permits networked Soloists for multi-axis sequenced motion and IO passing.
Soloist DESCRIPTION

The Soloist MP offers the same advanced software as the CP but in a smaller package designed for OEMs that can supply bus power from existing power supplies.

The Soloist HPe can be used for larger systems requiring up to 150 A peak current. The Soloist has two linear-drive form factors (compact and standard) that are ideal for higher precision applications.

The Soloist HLe/CL, with linear power stage, is available for low noise and ultra-high-performance applications. This controller is ideal for high bandwidth requirements and maintains superb linearity and zero crossover distortion. For example, applications that have many motion reversals and that require high position accuracy will benefit from using the Soloist HLe/CL.

The IDE software shares a common theme with other Aerotech software products providing the user with a clear, easy to use platform. This Windows®-based interface provides powerful diagnostic, development, and analysis tools for OEMs and end-users alike.

Allen-Bradley Interface

Combine proven PLC with proven motion control for easier integration, startup, and maintenance of medium- and high-end automation projects. The Aerotech EtherNet/IP™ interface enables AB PLCs (MicroLogix, CompactLogix™, or ControlLogix) to be integrated directly with the Soloist. Motion can be directly programmed in the RSLogix 5000 environment or separate programs can be written on the controller and triggered from the AB PLC. Aerotech has two interfaces: ASCII and Register. Choose the PLC, motion controller, and interface that best fits your application needs.

Total Solution

The controllers are fully tested and ready to run right out of the box. Aerotech can integrate the Soloist into a complete motion system, removing the burden of parameter setup and axis tuning.

Practical Power

Each series is capable of driving a wide range of motors including brushless, DC servo, and microsteppers. Brushless motors are sinusoidally commutated to minimize torque ripple.

Using a digital servo loop with feedforward, the Soloist tightly tracks velocity and position trajectories with virtually zero error. On-board autotuning and built-in calculators make servo tuning simple.

Variables, Math and More

With variables and math capability, one program can be used to produce a variety of parts by simply prompting the user for new application data.

Application Versatility

The Soloist has other built-in features such as axis calibration and backlash compensation, so you can maximize your machine’s accuracy and precision. The “user units” feature makes it easy to customize the Soloist to your specific machine, allowing custom units for both linear and rotary applications.

The controller is equipped with dual encoder inputs, so you can tackle master-slave applications or achieve higher accuracies with dual-loop control. Precise registration-based moves are also possible because of the fast 0.1 microsecond acknowledge time of the Soloist. The Soloist easily handles complex functions such as output-on-the-fly and velocity profiling.
Allen-Bradley EtherNet/IP Interface to Aerotech Soloist

The Aerotech Ethernet/IP interface enables AB PLCs (MicroLogix, CompactLogix, and ControlLogix) to integrate directly with the Soloist motion control solution. Motion can be directly programmed in the RSLogix 5000 environment or separate programs can be written on the controller and triggered from the AB PLC. Aerotech has two Ethernet/IP interfaces: ASCII and Register. Choose the PLC, motion controller, and Ethernet/IP interface that best fits your application needs.

The Allen-Bradley code snippets provided here are written in the graphical “relay ladder logic” syntax. Allen-Bradley also supports function block and structured text programming languages.

**ASCII COMMAND INTERFACE**

The ASCII command interface can be used to send ASCII text strings to the Soloist and perform a set of actions such as commanding motion or retrieving diagnostic information. Our vendor-specific EtherNet/IP ASCII command interface object extends this functionality across EtherNet/IP.

The power of the ASCII command interface lies in its simplicity and ease of use. A text string is formed using an AeroBASIC™ command, followed by an End-Of-String (EOS) character. ASCII response data indicates whether or not the command was successfully executed. An ACK character is sent to indicate success, an NAK character is sent if there is a command error, and a FAULT character is sent if there is a task error. For commands that expect return data, the response character is followed by the return data, which is terminated by the EOS character. The EOS, ACK, NAK, and FAULT characters are configurable via Soloist drive parameters.

To send the ASCII command from the PLC (programmed using Rockwell RSLogix) to the Soloist, an MSG block is used (Figure 1). The EtherNet/IP message can be easily configured from within the RSLogix software as shown in Figure 2.

ASCII Text_String (specified in the red box in Figure 2) is an RSLogix “tag”. The value of this tag is the ASCII command that will be sent to the Soloist. The example in Figure 3 shows the command “HOME” as it would be configured in RSLogix.

When the MSG block is activated, the ASCII command is sent to the Soloist via the EtherNet/IP protocol. The Destination field of the message configuration can be used to specify a local tag for the ASCII command’s return data.

Soloist Code Snippet (AeroBASIC) for ASCII Interface

For this interface (see Figure 4), the user does not need to write any AeroBASIC code on the Soloist. The Soloist

EtherNet/IP™ is a trademark of ODVA.
CompactLogix™ and RSLogix™ are trademarks of Allen-Bradley.
automatically receives and processes the commands. This functionality is built into our EtherNet/IP module.

**REGISTER INTERFACE**

Our vendor-specific EtherNet/IP register interface object allows the Soloist’s built-in integer and double registers to be accessed via EtherNet/IP. This provides a flexible, general-purpose interface to the Soloist that can be adapted to many different applications. Data consistency is guaranteed internally, so there is no need for concern when accessing these registers simultaneously via EtherNet/IP and AeroBASIC programs on the controller.

The Rockwell software is configured in very much the same way as the ASCII command interface. In this case, the message source (shown in red in Figure 5) is a data array.

The message configuration in Figure 5 is for the “Write Single Register” service. That service requires two pieces of data: the number of the register to write, and the value to be written. Therefore, we configure the message source tag as shown in Figure 6.

The value 100 is the register to write to, and the value to be written is 7. When the MSG block is activated, the register query is sent to the Soloist via the EtherNet/IP protocol.

As you can see, this is a straightforward data interface from the PLC to the Soloist. In addition to writing a single register, the interface also supports writing multiple registers in one message and reading single or multiple registers. This example was interfacing with an integer (32-bit) register, but the interface supports the same functionality with double-precision (64-bit) floating point values.

**Soloist Code Snippet (AeroBASIC) for Register Interface**

To make use of the data that is being transferred to and from the PLC, the user can write an AeroBASIC program to respond to incoming register transfers, as well as write outbound register data. This combination of EtherNet/IP communication with the power and flexibility of AeroBASIC allows for the implementation of many unique applications. For example, the following AeroBASIC code snippet shows how the register interface can be used to control the Soloist timebase value (set by the AeroBASIC TimeScale command).

**HEADER**

```
' The PLC writes timebase values to IntegerRegister 101
Define timebaseRegisterIndex 101
```

**DECLARATIONS**

```
' Declare a global variable named “timebase”
Global timebase as Integer
```

**PROGRAM**

```
' initial the timebase register to 100% speed
RegS IntegerRegisters, timebaseRegisterIndex, 100
'
' ... do work in AeroBASIC ... 
'
' Change timebase to value sent from PLC
Call GetTimebase()
TimeScale timebase
'
' ... continue working ...
```

**FUNCTION GetTimebase() as void**

```
' Read the value from IntegerRegister 101 into the variable “timebase”
timebase = RegS(IntegerRegisters, timebaseRegisterIndex)
```

**Figure 5. Allen-Bradley code snippet (Ladder Logic) for Register interface.**

**Figure 6. Allen-Bradley code snippet (Ladder Logic) for Register interface.**