

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *ACS Motion Control / Tech80 5912-4* at our website: *Click HERE*

ACS Tech80

REDEFINING MOTION CONTROL

Engineering Handbook

Version 3.00

ACS-Tech80 Part Number 700037
Document version no. 3.00

Information deemed to be correct at time of publishing. ACS-Tech80 reserves the right to change specifications without notice.
ACS-Tech80 is not responsible for incidental, consequential, or special damages of any kind in connection with this document.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Intentionally Left Blank

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Document version no.: 3.00 (July 2001)
Part number: 700037

COPYRIGHT
Copyright © 1999 - 2001 ACS-Tech80. Changes are periodically made to the information contained in this document. ACS-Tech80 Inc. reserves the right to change specifications without notice. This document cannot be reproduced in any form, without permission in writing from ACS-Tech80. All Rights Reserved.

TRADEMARKS
PEG, SPii, and ACS-Tech80 are trademarks of ACS-Tech80.
Microsoft Windows is a registered trademark of Microsoft Corporation. LabVIEW is a trademark of National Instruments



Internet: http://www.acs-tech80.com
E-mail: info@acs-tech80.com
support@acs-tech80.com

ACS-Tech80 Inc.
7351 Kirkwood Lane North, Suite 130
Maple Grove, MN 55369
USA
Tel: (+1) 763-493-4080 (800-545-2980 in USA)
Fax: (+1) 763-493-4089

ACS-Tech80 BV
Antonie van Leeuwenhoekstraat 18
3261 LT Oud-Beijerland
THE NETHERLANDS
Tel: (+31) 186-623518
Fax: (+31) 186-624462

ACS-Tech80 Ltd.
Ramat Gabriel Industrial Park
POB 5668
Migdal Ha'Emek, 10500
ISRAEL
Tel: (+972) 4-6546440
Fax: (+972) 4-6546443

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Intentionally Left Blank

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Changes to this Guide

Version	Date	Section	Changes	ECR
3.00	July 2001	SPiiPlus Series, SB Series, Library Programmable	Addition of new SPiiPlus PCI product & SPiiPlus section updated. SB Series programming examples updated. Addition of new SPii PCI product. Deletion of obsolete SPiiPlus SA products.	N/A
2.01	March 2001	SB Series	SB1291 references replaced with SB1381. Model A deleted leaving models B & C.	N/A
2.00	January 2001	all	New Handbook with updated product list and new technical support data. New sample programs from Applications.	N/A

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Intentionally Left Blank

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TABLE OF CONTENTS

Changes to this Guide 5

INTRODUCTION 9

Engineering Handbook Organization 10
Section 1: SPiiPlus Series Products Overview 10
Section 2: SB Series Products Overview 10
Section 3: Library Programmable Products Overview 10

SPiiPlus Series 11

Introduction 11
ACS-Tech80 SPiiPlus PCI Highlights 16
Hardware 16
Communications 16
Firmware Highlights 16
SPiiPlus Series: CONTROLLER FEATURES 20
Overview 20
SPiiPlus Series: FIRMWARE 32
Firmware Overview 32
Connect & Mapping Function Examples 57
Master Function Examples 59
ACSPL+ Programming Examples 62
Multi-Programming Synchronization example 62
Other Example Programs 63
SPiiPlus Series: DEVELOPMENT TOOLS 69
Development Tools 69
SPiiPlus Series: CONTROLLER SPECIFICATIONS 81
SPiiPlus Series: ORDERING MATRIX 83
SPiiPlus PCI Ordering Matrix 83

SB Series 84

Introduction 84
SB Series: CONTROLLER FEATURES 89
SB Series: FIRMWARE 95
Modes of Motion 106
SB Series: DEVELOPMENT TOOLS 140
Multi-Threaded Application written in C using Visual C++ for Windows NT 143
SB Series: ORDERING MATRIX 144
SB214PC Ordering Matrix 144
SB214SA Ordering Matrix 145
SB1381 Ordering Matrix 146
SB1292 Ordering Matrix 147
SB1391 Ordering Matrix 148
Unilite Ordering Matrix 149

LIBRARY PROGRAMMABLE 150

INTRODUCTION 150
SPii PCI Multi-Axis Motion Controller 151
Axis Options 154
Inputs & Outputs 155
Software Tools 156
SPii PCI Scope 160
SPii PCI Functions by Group 161
SPiiPCI General Specifications 166

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Environmental 167
Humidity: 90% RH, non-condensingEncoder Interfaces 167
Model 5912 168
Model 5312B 168
Stepper Controller 171
Model 5000 171
Multi-Axis Motion Controllers 171
Model 5650A 171
Model 5950B 172
LIBRARY PROGRAMMABLE: FEATURES 173
Multi-Axis Motion Controllers 173
Visual Basic Demo 177
COMPILING AND LINKING 178
Linking to Dynamic Linking Libraries 179
Routines By Group 180
LIBRARY PROGRAMMABLE: ORDERING MATRIX 185
SPii PCI Ordering Matrix 185
5000 Ordering Matrix 186
5650A Ordering Matrix 187
5950B Ordering Matrix 188
ENCODER INTERFACE: ORDERING MATRIX 189
5912 Ordering Matrix 189
5312B Ordering Matrix 190

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

INTRODUCTION

ACS-Tech80 Background

ACS-Tech80 is the combination of two successful motion control companies. ACS was formed in 1985 and is located in Migdal Ha'Emek, Israel. Technology 80 was created in 1980 and is located in Minneapolis, Minnesota. The two companies merged on September 30, 1999, to form ACS-Tech80. Technology 80's primary market was the United States while ACS' primary market was the rest of the world (ROW). Technically, Technology 80 is a wholly owned subsidiary of ACS Electronics. ACS-Tech80 is a publicly held organization that is traded under the symbol ACSEF.

ACS-Tech80 maintains two strategic world locations in Israel and Minnesota. Israel is well known for developing technological products. Technology companies that have facilities in Israel include:

- General Electric
- Compaq
- KLA Tencor
- Seimens
- Samsung
- Motorola
- Applied Materials
- Microsoft
- Intel
- Philips
- Brother
- Anorad
- AT&T
- America On Line (AOL)

Israel is one of the premier locations for developing new technology, especially semi-conductor, hardware, and software technology. The Minneapolis facility is located in Minneapolis, Minnesota. Minneapolis is the home of numerous technology companies and specifically has generated a worldwide presence in motion control. There are more than a dozen organizations that focus on manufacturing motion control products in the Minneapolis area. Minneapolis is located in the heart of the mid-western United States to serve the U.S. market.

There are several goals for ACS-Tech80. The merger takes advantage of economies of scale and economies of scope. Economies of scale are the result of increased volume of production. Economies of scope are the result of two locations to better serve a worldwide market with experienced engineering, sales, applications, marketing, and production personnel in two strategic locations. This leads to better efficiencies and takes advantage of reducing the cost associated with overhead activities. Technology 80 brings vast expertise and Library Programmable controllers while ACS brings stand-alone and control modules (integrated motion controllers and digital amplifiers). The goal of ACS-Tech80 is to be able to market high performance motion controllers and control modules worldwide with two locations in Israel and Minnesota. The products are stocked in both locations to provide quick delivery on manufactured components.

ACS-Tech80 employs more than 85 people. Sixty people are located in the Israel facility and 25 people are located in the Minnesota facility. The core of ACS-Tech80 is research and development (R&D). R&D is maintained in both locations, concentrating core efforts where the best expertise is located. This allows invaluable R&D and customer service and support to reside in both locations. There are approximately 30 full-time engineers, including 12 full-time software engineers. The multi-disciplinary R&D staff provides expertise in real time software, motion control, digital, and analog electronics, ASIC design, and power electronics. Thus, the primary expertise for ACS-Tech80 is both hardware and software development for machine controllers. ACS-Tech80 R&D is supported by managing activities associated with marketing and production effectively.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Engineering Handbook Organization

This Handbook is organized into three distinct sections:

- **Section 1: SPiiPlus Series Products (ACSPL+ Programming Language)**
- **Section 2: SB Series Products (Control Modules and Multi-Axis PC / Stand-alone Motion Controllers with ACSPL Programming Language)**
- **Section 3: LIBRARY Programmable Multi-Axis Motion Controllers and Encoder Interfaces**

Section 1: SPiiPlus Series Products Overview

The SPiiPlus Series products feature the ACSPL+ programming language. The ACSPL+ programming language is an advanced motion control language developed on the Intel Pentium programming platform with distributed RISC architecture. The multi-axis controllers using ACSPL+ support the following drive technologies:

- +/-10V analog drives
- Sinusoidal Commutation

The SPiiPlus Series products have the ability to close the position and velocity loops for motion control applications. The SPiiPlus Series products feature optional SPiiPlus C LIB Libraries. The SPiiPlus C LIB Libraries are comprehensive C, C++, and Visual Basic libraries for Windows 98/NT/2000 operating systems. Full multi-thread support is provided for Windows 98/NT/2000.

Section 2: SB Series Products Overview

The SB Series products feature the ACSPL programming language. The ACSPL programming language is a motion control language developed on the Intel 196 with distributed RISC architecture. The ACSPL programmed products also include control modules (integrated motion controllers and digital amplifiers). Thus, the ACSPL products are sophisticated motion controllers and highly intelligent single and multi-axis control modules. The control modules are universal with programmable digital drives. The control modules and supporting development and diagnostic software supports the following motor technologies:

- Linear Brushless Motors
- AC/DC Brushless Servo Motors
- DC Brushed Servo Motors
- AC Induction Motors

The ACSPL products feature the ACSLIB (C Libraries) as a cost option. The libraries are comprehensive C, C++, and Visual Basic libraries for DOS, Windows 3.x/95/98/NT. Full multi-thread support is provided for Windows 95/98/NT.

Section 3: Library Programmable Products Overview

The Library Programmable, multi-axis motion controllers are board level products for the PCI, PC/ISA, PC/PCI, and PC/104 architectures. A host PC is needed to operate the motion controllers and encoder interface boards. The Library Programmable controllers support servo, stepper, and commutation. The Open Architecture Software Library provides support for C, C++, BASIC, and Visual Basic. The 16- and 32-bit libraries support DOS, Windows 3.x/95/98/NT.

The Library Programmable motion controllers and encoder interfaces are the most cost-effective solutions on the market today, which can be implemented in high volume applications.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SPiiPlus Series

Introduction

The SPiiPlus Series of multi-axis motion controller are specifically designed to answer the most demanding needs of present and future equipment, such as semiconductor manufacturing and inspection, electronic assembly, pick & place, complex digital printing, medical scanners, packaging, and material handling.

The SPiiPlus controllers are targeted at applications that require:



- Multi-axis interpolation
- Tight synchronization between trajectory, position, I/O, and time
- The ultimate in accuracy and throughput
- Sin-Cos encoder multipliers and sub-nanometer resolution without speed compromises
- 1/T sub-count interpolation
- Flexible motion generation
- Gantry control
- Fast time to market and future growth while maintaining investments made in existing software

The SPiiPlus Series controllers are based on the most advanced, most powerful, and easy to use software and hardware technologies available. The SPiiPlus series itself is named after the second generation of the Servo Processor (SPii) developed by ACS-Tech80.

The SPiiPlus PCI controllers can be used either as a PCI card in a desktop PC or as a stand-alone controller using Serial communication or an optional 10 Base T Ethernet module.

The controllers use multi processor distributed architecture, with an Intel x86 / Pentium based Motion Processor Unit (MPU) and a dedicated Servo Processor per two axes for servo control algorithm execution. As part of the distributed architecture, the X86 engine distributes instructions to the Servo Processor, which perform the real time motion control. The SPiiPlus motion controllers are fully on-board programmable. They are programmed using the ACSPL+ programming language.

ACSPL+ is an extremely flexible and powerful programming language developed on the X86 Intel architecture and is specifically designed for motion control. ACSPL+ is a C-like high-level, multi-tasking compiled language. It enables you to implement complex motion / time / event sequences, shifting the processing load from the host computer to the controller. The knowledgeable user can modify even the control algorithm, safely and easily.

Special Features For Special Needs:

- Sin-Cos encoder multipliers with x4 – x8192 multiplication factor (Optional)
- Position Event Capture (PEG) hardware position capture with extended capability
- 1/T measurement for smooth and precise low speed motion
- Software commutation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Advanced Software Development Tools – The Surest Way to Minimize Time to Market

To ensure ease of development and fast time to market, the SPiiPlus Series features a comprehensive suite of software tools for programming and debugging applications. The tools suite allows for software development, feedback tuning, modifying the control algorithm, monitoring, diagnostics, and debugging. A comprehensive DLL / static library is available for host application programming in C/C++ and Visual Basic, with support for the most popular operating systems such as Windows 98 / 2000 / NT.

A unique feature of the software is the built-in SPiiPlus Simulator, which simulates the behavior of the SPiiPlus controller by closing the loop with zero following error. The Simulator allows applications to be developed and logically developed and debugged without attaching any hardware. This SPiiPlus Simulator vastly improves time to market.

Standard software:

- SPiiPlus MMI (Motion Machine Interface): A tool for tuning the axes and analyzing motion and motor performance. Includes the SPiiPlus Simulator for designing and troubleshooting programs without the need for a controller (see page 66).

Optional software:

- SPiiPlus MultiDebugger for Windows: Multi-threaded programs can be written, compiled, and executed with built-in diagnostics for parameters and I/O. The MultiDebugger supports up to 9 different programming strings in separate buffers for line-by-line sychronization. Includes the SPiiPlus Simulator for running and debugging programs without hardware attached.

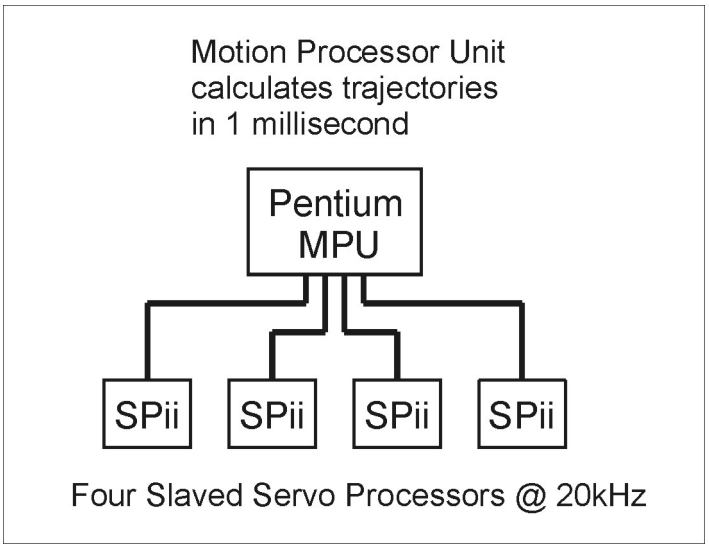
SPiiPlus Series Products Table

Product	Controller Type	# of Axes	Discrete Digital I/O	Feedback Types	Drive Command Output Types
SPiiPlus PCI-4	PCI Bus	1-4	8in / 8 out	Encoder Sin-Cos	+/-10VDC (16-bit) Sinusoidal Commutation
SPiiPlus PCI-8	PCI Bus	1-8	8in / 8 out	Encoder Sin-Cos	+/-10VDC (16-bit) Sinusoidal Commutation
SPiiPlus PCI-ETR	Ethernet (10BaseT)	1-4	8in / 8 out	Encoder Sin-Cos	+/-10VDC (16-bit) Sinusoidal Commutation
SPiiPlus PCI-ETR	Ethernet (10BaseT)	1-8	8in / 8 out	Encoder Sin-Cos	+/-10VDC (16-bit) Sinusoidal Commutation
SPiiPlus PCI-SA	Serial RS-232	1-4	8in / 8 out	Encoder Sin-Cos	+/-10VDC (16-bit) Sinusoidal Commutation
SPiiPlus PCI-SA	Serial RS-232	1-8	8in / 8 out	Encoder Sin-Cos	+/-10VDC (16-bit) Sinusoidal Commutation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The SPiiPlus controllers have significant speed and performance advantages over other motion controllers. The multi-processor and distributed architecture allows motion control tasks to be executed efficiently and seamlessly. The X86 processor is referred to as the Motion Processor Unit (MPU). The MPU performs the high level tasks including the trajectory calculations, pre-emptive multi-tasking programs, communications, and I/O management tasks. The MPU is a 133 MHz Pentium-compatible 586, which performs a deterministic 1 millisecond MPU cycle time.

By using a distributed architecture, the servo loop performance is executed in 50 microseconds independent of the number of axes controlled. The slaved processors are supplied with a PIV (Position Integration Voltage) control algorithm with a notch filter and a second-order low-pass filter. The Servo Processors implement the control loops and hard real time control.



Distributed Architecture in the SPiiPlus Product Family (8 Axis Layout)

There are two slaved Servo Processors supplied with the 4 axis SPiiPlus controllers and four servo processors with the 8 axis controllers. (Each Servo Processor controls two axes). Each distributed Servo Processor performs the real time servo loop calculations. This architecture gives the SPiiPlus motion controllers extremely high performance. (See matrix on next page).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

FEATURE	ACS-TECH80 SPiiPlus	BRAND D	BRAND G	BRAND M
Architecture				
Distributed / single cpu	1MPU+4 SPii	Single CPU	Single CPU	Single CPU
Main Processor	Pentium®compatible	Motorola 56000	Motorola 68000	Analog Devices SHARC
Servo Processor	Yes - SPii, 24 bit RISC (120 MHz each)	No	No	No
MPU Operating Frequency	133MHz	40MHz	40 MHz	150 MHz
Real-Time OS	Yes	No	No	No
Communications	PCI, Serial and Ethernet from same controller to ease development for different applications	Varies	Varies	varies
Servo performance				
Servo Update Rate (8 axes)	50 µsec., 20 kHz	144 µsec, 7 kHz	1 msec. or more, 1 kHz	10KHz
Standard control algorithm	PIV, with automatic Velocity & Acceleration Feed Forward	PID, VFF, AFF	PID, VFF, AFF	PID, VFF, AFF
User's customized algorithm	Yes, powerful dvelopment tool – SPii Debugger	Yes, in firmware	Yes, in firmware	Yes, in firmware
Sinusoidal commutation	Yes, at 20kHz update rate	Yes	Yes	Yes
Trajectory generation				
Trajectory Update Rate (8 axes)	1kHz	100 - 200 Hz	500 Hz Maximum	Dependent on OS
Trajectory Calculation	64 bit, floating point	56 bit, fixed point	32 bit, fixed point	64 bit, floating point
Motion Variable Rate	Guaranteed in 1 msec.	5 msec.	10 msec.	Dependent on OS
On board application high level language	Yes, ACSPL+	Yes, Complex	Yes, 2-letter commands	No
Programming	Easy	Difficult	Easy	Easy
Multi-tasking	Pre-Emptive, programmable	Time Share	Time Share	On PC
Program Execution	All programs compiled	Some PLC's Compiled	Interpreted Language	Compiled on PC
Programming Flexibility	Flexible & Powerful	Flexible, but difficult	Limited	Flexible, but time consuming
High Speed PLC	Yes, 1 msec.	Yes	No	No
Additional PLC programs	Yes, programmable scan time	Yes	No	Yes, must be developed on PC
User Variable Update Rate	Guaranteed in 1 msec.	5 msec.	Interpreted, 1 msec / variable	Dependent on OS
Variable Definitions	User defined names	User defined numbers	User defined numbers	User Defined Names
User units	Yes	Yes	Yes	Yes
User Memory	8 Mb	384 K (approximate)	256 K (approximate)	On PC
C/C++/Visual Basic Programming	Yes, Windows 98/NT/2000	Yes	Yes	Yes

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

FEATURE	ACS-TECH80 SPiiPlus PCI	BRAND D	BRAND G	BRAND M
Communication channels				
PCI	FIFO and Dual Port RAM (DPR) standard	FIFO (DPR Optional)	FIFO	FIFO
RS232	2 channels, 115,000 baud, standard	Optional	Optional	No
Ethernet	10Based T	No	Optional product	Optional product
Simultaneous Communications	Yes, through all channels, preemptive	No	No	No
Advanced features				
Position Registration	<0.1 µsec., hardware	<1 µsec, hardware	Software regulated	<1µsec
Position Compare	0.2 µsec, hardware (PEG™) 25 nanosecond minimum pulse width, 4 MHz frequency Incremental mode Random position, table based mode	1 µsec, hardware	Software compare	1 µsec
Sin / Cos Encoders	Yes, 4 – 8192 programmable multiplier. (On board). Maximum encoder frequency: 500kHz	Yes, 4096 multiplier (separate module)	No	Yes, 4096 multiplier (separate module)
Dynamic error mapping	Yes, two dimensional	Yes	No	No
Electronic CAM	Yes, segmented	Yes	Yes	Yes
Electronic GEAR	Yes, floating point	Yes, fixed point	Yes, integer derived	Yes
Inverse Kinematics	Yes, user programmable	Yes	No	Yes
I/O				
User Digital I/O	8/8, RS422/485 type	24VDC, opto-isolated	5VDC, TTL	5VDC opto-isolated
Safety, drive enable, fault, E-stop	5 or 24Vdc, opto-isolated	Yes	Yes	Yes
D to A	16 bits resolution	16 bits	16 bits	16 bits
A to D	14 bit resolution, standard, 16 single-ended channels	Optional	Optional	Optional
I/O extension bus	High-speed Serial Synchronous interface (HSSI), 256 input bits, 256 output bits. All 256/256 are sampled and updated every µsec. Open to the user. 5-24VDC, optically-isolated	No	No	No
Simulator				
Full simulator of the controller.	Yes. Full simulator of the controller. Available with all tools – C library, Mmi, Multi-debugger. Develop on board and host programs without hardware	No	No	No

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ACS-Tech80 SPiiPlus PCI Highlights

Hardware

- MPU is 586 Pentium-compatible @ 133 MHz
- 24-Bit RISC SB2700 Servo Processors (SPii's) for Distributed Architecture
- Real Time Operating System provided on MPU for deterministic control
- MPU Handles Trajectory Generation, Digital I/O, Communications, and Programs
- Distributed Control Servo Processor Executes Real Time Control Algorithms.
- Dedicated Optically-Isolated Digital I/O for Home, Limits, and E-Stop, drive enable, and drive fault, 5 or 24VDC and Source or Sink Programmable
- 4 Megabytes Flash Memory for Firmware Storage
- 1 Megabyte Flash Memory for User Programmable Functions
- 8 Megabytes RAM Memory (Expandable)
- Hardware Capture and Compare Registers
- 20 MHz Encoder Feedback (Built-in Encoder Loss and Error Detection)
- Sixteen 14-bit resolution Analog Inputs
- Sin-Cos Encoder feedback with programmable x4 to x8192 Multiplication (Optional)
- Drive Command Signals for Analog Drives
- SA and Ethernet options can be provided with a stand-alone mounting bracket for remote installations

Communications

- PCI bus
- Optional Ethernet 10BaseT
- RS-232 Serial Communications to 115K Baud
- Simultaneous Communications on all communication channels
- Software Development Tools for Diagnostics, Tuning, and Debugging
- Dynamic Link Libraries for Windows 98/NT/2000 Operating Systems (Multi-Threaded)

Firmware Highlights

Execution

- ACSPL+ Motion Multi-tasking Programming Language
- Label Based Programming Language with On-Board Compiled Execution
- 10 Simultaneously Executed Program Buffers (Programmable Prioritization)
- Deterministic Execution of Program Buffers (Single Line-By-Line Synchronization)
- Multiple Command Execution on a Single Program Line
- Automatic Routines sampled on each MPU cycle (1 millisecond)
- Built-in PLC Implementing Parameters, Flags, and Mathematical Expressions

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Variables and Program Flow

- 32/64-Bit Variables and 64-Bit Trajectory Calculations (64-Bit Floating Point Calculations)
- Real and Integer Variables
- Scalar, Local, and Global Variables
- Local and Global One- and Two-Dimensional Arrays
- Arbitrary Names of Variables
- Full Set of Arithmetical, Logical, and Comparison Operators
- Pre-Programmed Hardware Registers and Flags
- Programmable Engineering Units
- If-Then-Else, Case, Repeat/Until, Do/While, Goto, Till, and Label Subroutines

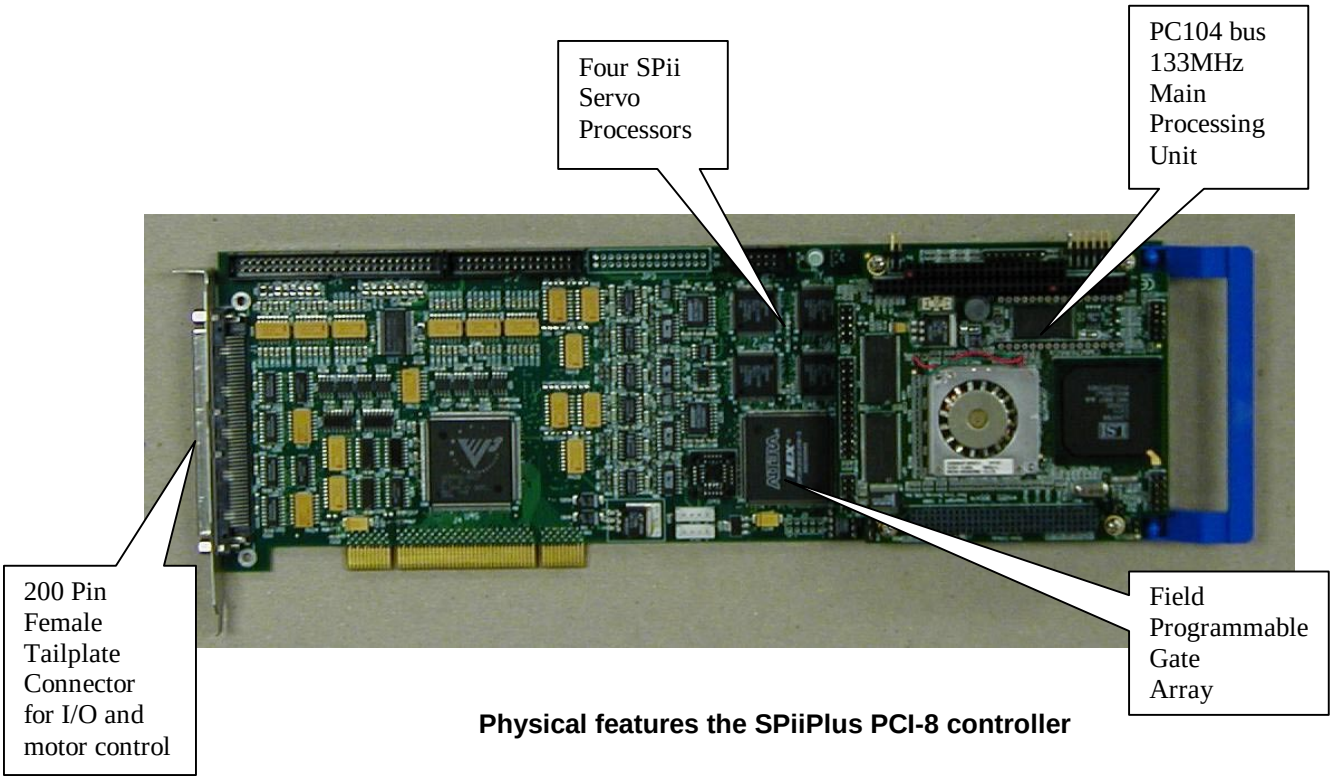
Motion Profiling

- Programmable Electronic Cam and Electronic Gearing
- On-the-Fly Velocity and Position Matching
- Position Event Generation (Hardware Position Compare), incremental interval and random position, table based modes
- Up to 8 Groups (Coordinate Systems)
- Complete Virtual Axes and Phantom Axis Programming
- Linear, Circular, Sinusoidal, Elliptical, and Helical Interpolation
- Programmable Coordinated Paths Including Inverse Kinematics
- Dual Loop Feedback Control (Special Order)
- Multiple Coordinate Systems and Types of Interpolation
- Incremental and Absolute Jogging
- S-Curve Acceleration and Deceleration
- Teach and Learn Functions
- Programmable Dynamic Two-dimentional Error Mapping

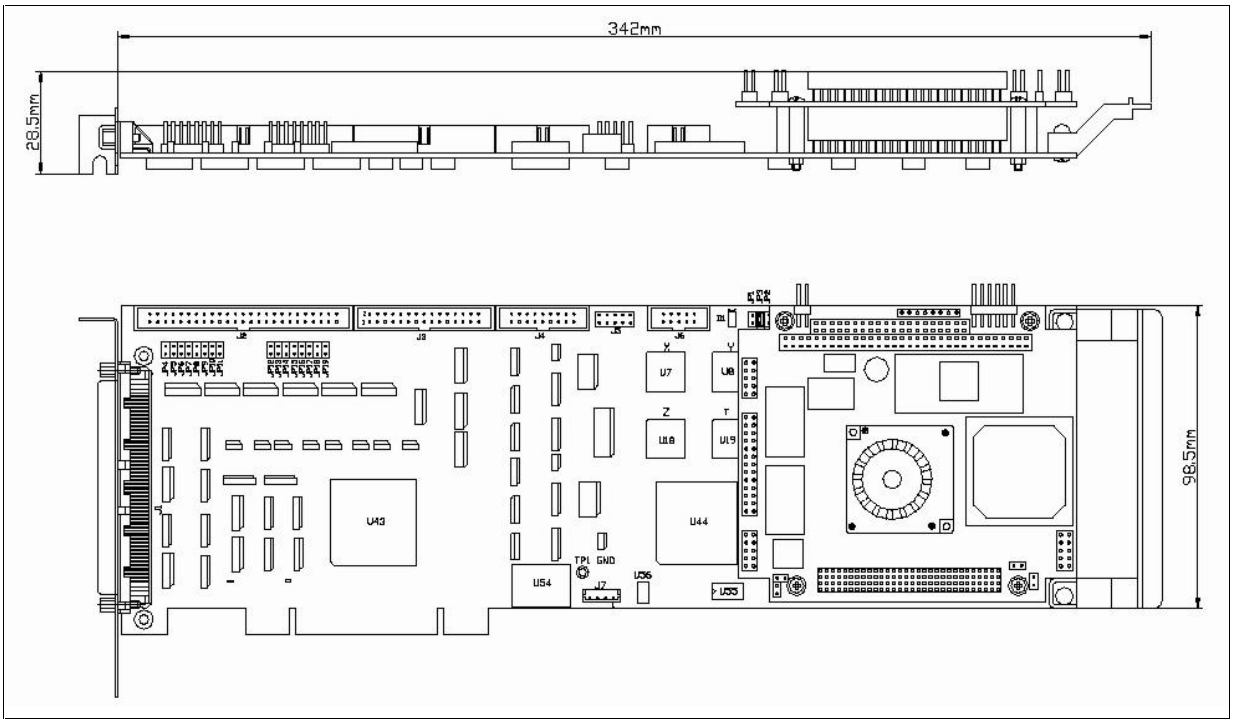
Motion Control Performance

- New Trajectory Calculated Every MPU cycle (1 Millisecond)
- Servo Update Rate of 20kHz (50 microseconds, Independent of number of Axes)
- Real Time Operating System with Deterministic 1 Millisecond Scan Rate
- 16-bit DAC Drive Commands, two per axis for software sinusoidal commutation of AC servo motors
- Hardware Position Event Generation (PEG) to 4 MHz
- Positioning Accuracy to 1 Encoder count
- Velocity Resolution to 1 Count/Second
- Optional Sin-Cos encoder interface. One per axis. Programmable multipier x4 to x8192 counts per encoder line. (Note: Each Sin-Cos axis takes two analog inputs.)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Physical features the SPiiPlus PCI-8 controller



Dimensions of the SPiiPlus PCI controller

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SPiiPlus Tools Suite Highlights

1 SPiiPlus MMI for Windows: User interface to the controller. Multi-purpose interactive support tool for setting up, tuning, and viewing parameters. Features include a four channel soft oscilloscope with FFT capabilities. The MMI also includes the SPiiPlus Simulator for operating the MMI without hardware attached. The Simulator can be used to write and debug motion programs and host applications before any hardware is purchased. The Simulator simulates the motion profile with zero closed-loop following error. See SPiiPlus Development Tools (page 66) for details.

2 SPiiPlus Upgrader: A utility program that simplifies upgrading or downgrading of the SPiiPlus PCI controller firmware. The Upgrader can be run as a command line console mode as well as a Windows application.

3 SPiiPlus Emergency Wizard: Allows users to recover the contents of the firmware in the controller if the firmware is damaged. The firmware can then be upgraded with the SPiiPlus Upgrader.

SPiiPlus Application Development Kit (Optional)

1 SPiiPlus MultiDebugger for Windows: Development environment for ACSPL+ (ACS Multi-Programming Language) applications. Multiple programs can be developed, compiled, and executed with built-in diagnostics for parameters and I/O. The MultiDebugger includes the SPiiPlus Simulator for operating the MMI without hardware attached. The Simulator can be used to write and debug motion programs and host applications.

2 SPiiPlus C LIB Library: Comprehensive Dynamic Link Library (DLL) that supports host application programming in C/C++ and Visual Basic. Supports Windows 98/NT/2000. Full multi-threaded support in Windows 98/NT/2000.

Note:
For a limited time, the Applications Development Kit will be included with the standard software tools suite at no additional charge.

ACS-Tech 80 reserves the right to withdraw this offer at any time without advance notice.

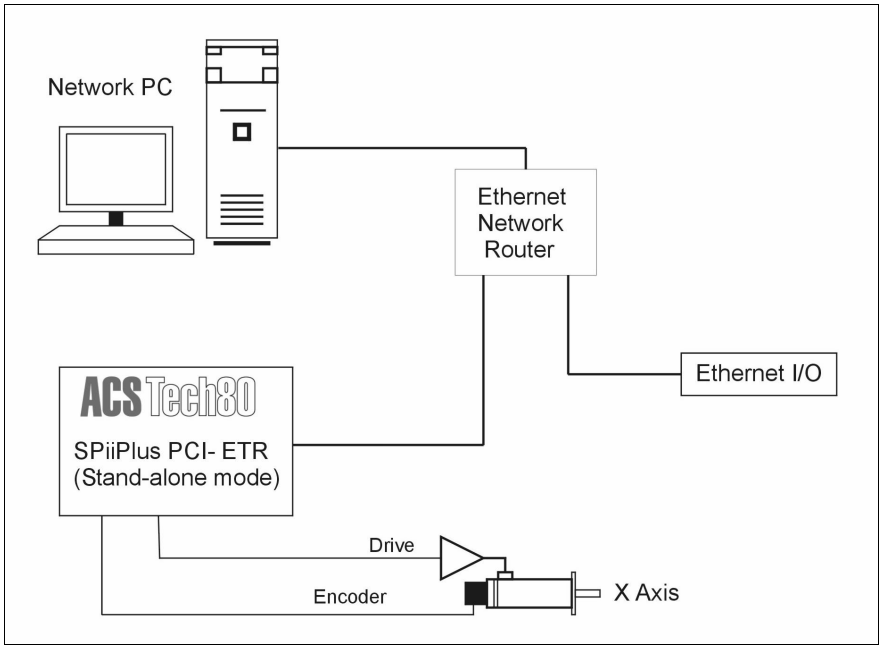
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SPiiPlus Series: CONTROLLER FEATURES

Overview

The ACS-Tech80 SPiiPlus PCI, Serial, and Ethernet controllers are 64-bit multi-tasking and multi-processor based motion controllers. The controllers are ideal for applications that require a high degree of processing power and real time control performance. The controller's Motion Processor Unit (MPU) is provided with a built-in Real Time Operating System (RTOS). The RTOS installed on the MPU along with the dedicated servo processors eliminates the need for a host processor. The SPiiPlus PCI controllers can communicate back to a host PC for implementing an Application Programming Interface (API).

The SPiiPlus PCI can function as a stand-alone controller when mounted on an optional mounting bracket and interface kit. This allows the controller to be located remotely from the PC chassis.
The controllers can communicate back to a host PC for implementing an Application Programming Interface (API). With the Ethernet communications, serial data communications can occur to the host PC while locating the controller outside of the PC chassis.



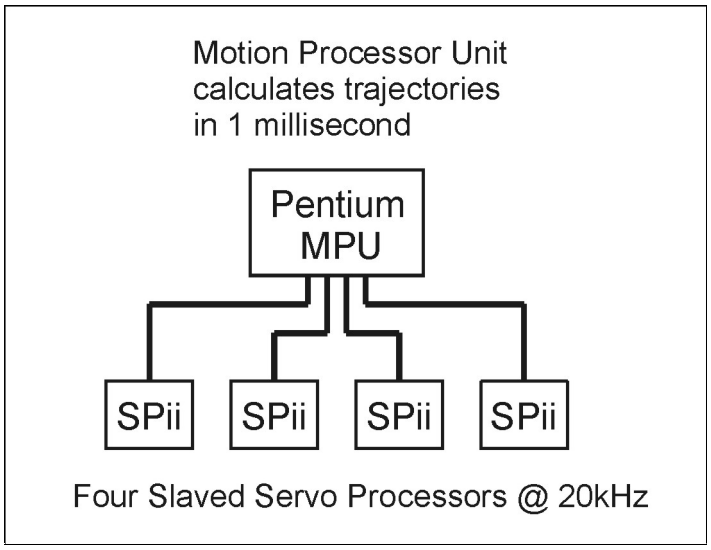
This method allows locating the controller in the equipment panel near the amplifiers and interconnection devices. This simplifies interconnectivity and dramatically reduces wiring connections costs.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Motion Processor Unit

The MPU is an X86 based processor (see picture below). The standard configuration of the MPU is a Pentium-compatable 586 processor operating @ 133 MHz. With the standard MPU, the controller executes compiled programs in a deterministic 1 millisecond cycle time. The MPU has several functions, including:

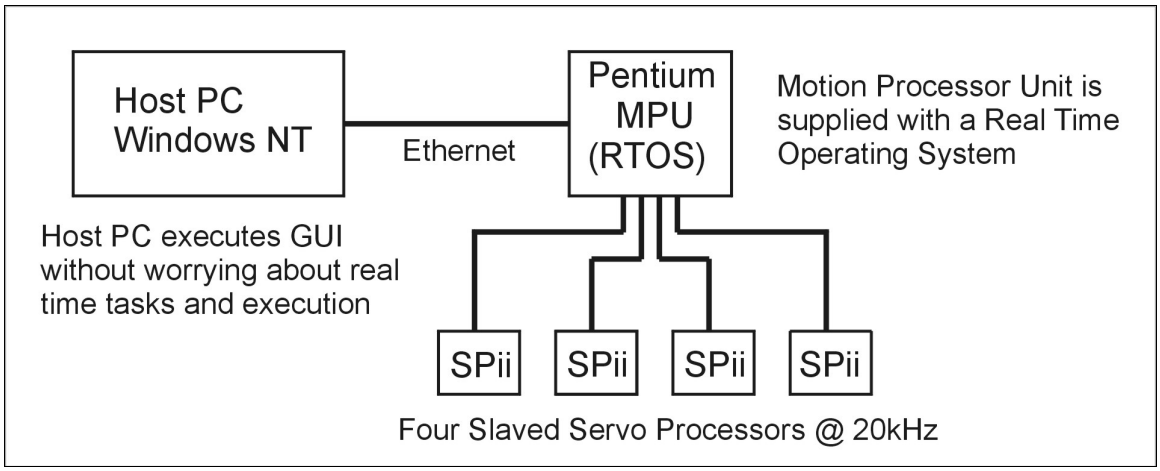
- 64-bit trajectory calculation (52 bits mantissa, 12 bit exponent)
- Feeds coordinated profiles to slaved servo processors
- Executes the firmware commands (multi-threaded buffers)
- Executes the communications interface
- Real Time Operating System for safety controls



Distributed Architecture in the SPiiPlus Product Family

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

An additional feature of the MPU is the status updates. Every parameter and flag, including user programmable parameters and flags, is updated at the MPU cycle rate of 1 millisecond. Thus, critical parameters and flags are always updated at the scan rate. Other motion controllers with lower performance DSP's do not achieve this update rate. For example, let us assume that a semi-conductor machine needs to stop when the machine is within the "in position flag." The in-position flag is updated at the MPU cycle rate. Thus, the controller will know and execute the appropriate logic within the default 1 millisecond scan rate. Other motion controllers do not update critical flags and parameters at every cycle. The ACS-Tech80 implementation dramatically improves overall performance and increases throughput in demanding applications.



As previously noted, the MPU is provided with a Real Time Operating System (RTOS) to guarantee real time performance. The RTOS is important to allow deterministic control and the fast execution of motion programming (i.e., execution of programs, and updating of flags and parameters). With the RTOS on-board in the ACS-Tech80 controller, the real time program execution can be passed to the controller. This frees up the host processor to perform the graphical user interface (GUI). Thus, the host processor does not require a RTOS and the host can simply run standard operating systems, the most popular being Windows NT. Here again, the programmer can implement off-the-shelf programming tools for C and Visual Basic more easily. This can greatly reduce the engineering development time (i.e., time to market).

PC-Based Control

The ACS-Tech80 solution is an open architecture PC control, based on the open standard of the personal computer industry. This philosophy allows superior future designs to be achieved by default due to the enhancement of using standard PC hardware. The engineer can be confident that the ACS-Tech80 solution will be a valuable controller both now and in the future as future ACS-Tech80 designs can be ported easily to the PC platform. This is true since the PC industry has a distinct economy of scale advantage. The PC industry consumes $200 billion while the industrial control market consumes $2 billion annually (i.e., two orders of magnitude larger). PC technology is the reason PC-based motion control exists. Thus, it makes sense to base the selected design on a PC topology.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Trajectory Calculation / Servo Loop Update Rate

The ACS-Tech80 SPiiPlus PCI controllers are provided with high bandwidth trajectory and servo loop calculations. At ACS-Tech80, the true measure of a motion controller is the combination of both the trajectory calculation and the servo loop calculation speeds. These speeds determine the overall performance of the motion controller. The highest performance motion controllers will have fast trajectory and servo loop updates (see the Update Rate Table below).

Trajectory Calculation/Servo Loop Update Rate Table

# Axes	Calculation Bits	Trajectory Update Rate	Servo Update Rate	Parameter/Flag Update Rate[1]
1	64 bits	1 millisecond	50 microseconds	1 microsecond
2	64 bits	1 millisecond	50 microseconds	1 microsecond
3	64 bits	1 millisecond	50 microseconds	1 microsecond
4	64 bits	1 millisecond	50 microseconds	1 microsecond
5	64 bits	1 millisecond	50 microseconds	1 microsecond
6	64 bits	1 millisecond	50 microseconds	1 microsecond
7	64 bits	1 millisecond	50 microseconds	1 microsecond
8	64 bits	1 millisecond	50 microseconds	1 microsecond

Note 1: Every parameter and flag, including user defined parameters and flags, are updated every interrupt of the Motion Processing Unit (MPU) real time operating system.

The MPU performs the trajectory calculations. The default trajectory calculation for all axes is 1000 points per second (1000 Hz). The trajectory calculation is a full 64-bit floating point calculation (52 bit mantissa and 12 bit exponent). The engineer will not get rounding errors with the 64 bit precision trajectory calculations provided in the SPiiPlus controllers.

Pre-Emptive Multi-Tasker / Multi-Programming

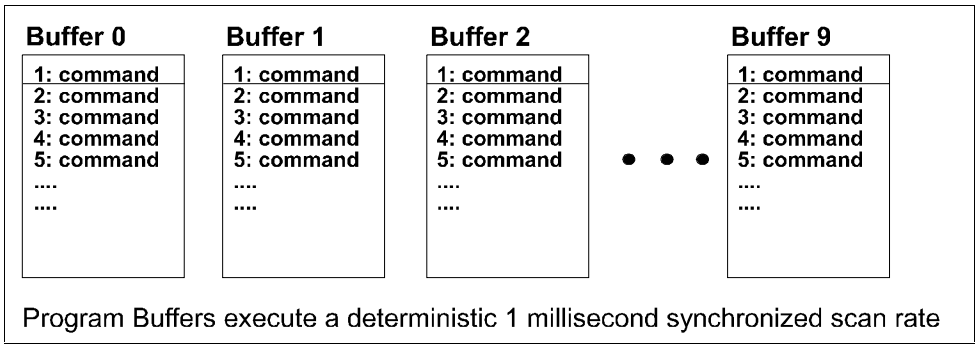
The ACS-Tech80 controllers are true pre-emptive multi-taskers capable of performing simultaneous tasks. The controllers are provided with 10 program buffers. Each buffer can execute motion and PLC programming. Each of the buffers can interact with other buffers through global variables and synchronized with the MPU scan rate.

For example, one program buffer can be executing a motion profile on the X-axis while a second program buffer is executing a motion profile on the Y-axis. The two program buffers can be synchronized by a global variable (SYNC) so they start at exactly the same time. The two axes can be synchronized even though their motion profiles are different (i.e., different position, velocity, and acceleration profiles). In addition, additional motion programs, high speed and normal PLC programs, and other tasks can all be running simultaneously. Program buffers can interact with the communications ports to send and receive data intelligently. A program buffer can be programmed to capture current position data and send the data to a serial port for off-line storage.

The design is a true pre-emptive multi-tasking design. The program buffers do not time share but act independently within the same MPU scan. Other motion controllers will slow down and cannot interact with the same time cycle like the ACS-Tech80 controllers can. The result is a complete design controlled by the design engineer so that time critical events can be maintained. The performance and accuracy is maintained with the distributed control architecture.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Buffer Synchronization

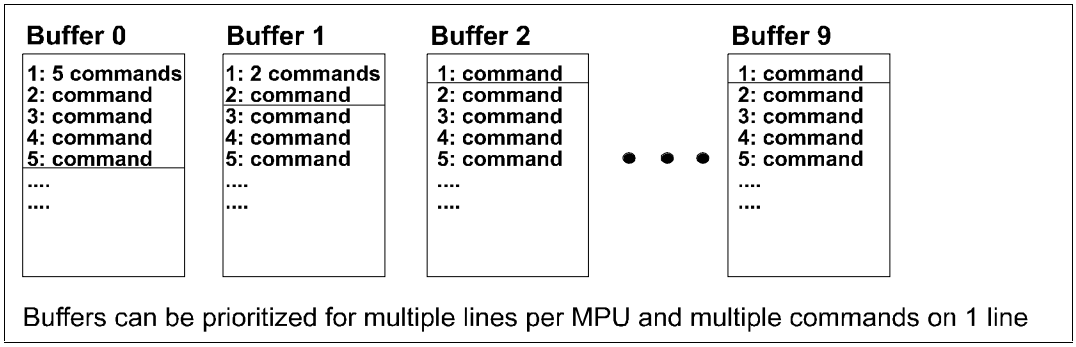


The on-board firmware command set is highly dynamic and flexible. Virtually any number of variables or mathematical expressions can be programmed and solved at the MPU cycle rate of 1 millisecond). The ACSPL+ programming language is compiled by the MPU before execution. It guarantees fast execution and reduces processing overhead. There are 10 multi-tasked program buffers. The programmer is guaranteed that one line of every program will be executed in the same MPU cycle (see Buffer Synchronization picture above). Each line can contain multiple commands. Although there is no formal limit, in typical programming practice up to 10 ACSPL+ commands can reside on the same program buffer line (or up to 1000 characters).

The program buffers can be prioritized (see Buffer Command Prioritization picture below). For example, a program buffer can execute multiple lines of commands in the same MPU cycle.

Buffer Command Prioritization

The priority of the buffer is controller with a 10-element array. Each element of the array controls the number of lines executed in 1 MPU cycle. The default configuration is 1 line per MPU cycle. The parameter name of the buffer priority is the PRATE (Program Rate) parameter. The priority (execution rate) of each buffer can be individually controlled by the PRATE parameter. PRATE determines how many lines are executed each MPU cycle.

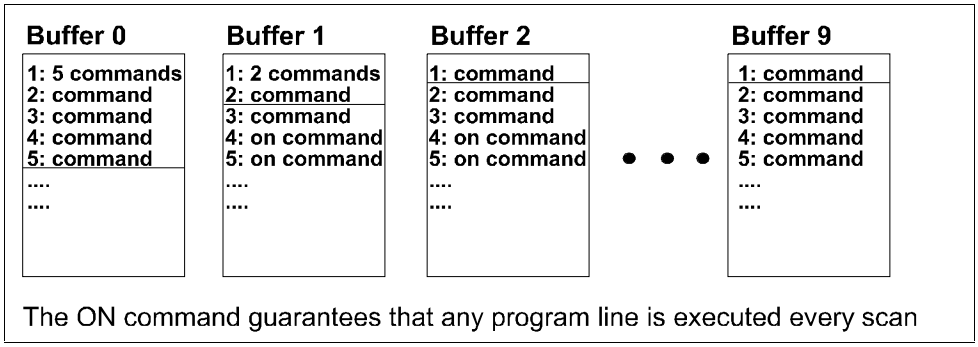


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PLC Programming

The ACSPL+ programming language includes PLC programs. PLC programs can reside in any program buffer. PLC programs can contain digital and analog I/O logic, variables or any mathematical expression. There are two types of PLC programs. PLC programs can be executed at high-speed or normal scan rates. High-speed PLC programs are guaranteed a 1-millisecond scan rate by using autoroutines (ON command, see High Speed PLC Execution Picture). Normal PLC programs run at a programmable rate. For normal PLC programs, the number of program lines multiplied by the MPU cycle rate determines the scan rate. Both high speed and normal PLC programs are deterministic due to the RTOS supplied on the MPU. There is no practical limit to the number of high-speed or normal PLC lines. Only the memory size and processing speed of the MPU limit the PLC size.

High Speed PLC Execution



PLC programs are directly related to autoroutines. Autoroutines are programs that are executed based on an event. Autoroutines are typically high-speed PLC programs that are executed when a critical event occurs. For example, an autoroutine can be executed when a limit switch is triggered. After the motion profile is stopped by the limit, error recovery can be initiated with autoroutines

Similar to the program rate priority (PRATE), the autoroutines can also be prioritized to execute a user-specified number of lines in 1 MPU cycle. The priority is programmed using a 10-element array. The 10-element array parameter is the ONRATE parameter. The maximum number of autoroutine lines executed in 1 MPU cycle (1 msec) is 10.

PLC Implementation

The ACSPL+ programming language can be used to implement powerful PLC programs. PLC programs can reside in any program buffer. PLC programs can contain digital I/O logic, variables, or any mathematical expression. There are two types of PLC programs. PLC programs can be executed at high-speed or normal scan rates. The high-speed PLC programs are similar to autoroutines. High-speed PLC programs are guaranteed a 1-millisecond scan rate. Normal PLC programs run at a programmable rate. The number of program lines multiplied by the MPU cycle rate determines the normal PLC program rate. Both high speed and normal PLC programs are deterministic due to the RTOS supplied on the MPU. There is no practical limit to the number of high-speed or normal PLC lines. Only the memory size and processing speed of the MPU limit the PLC size.

The PLC is often used in order to handle digital inputs/outputs. Similar to the techniques used for autoroutines, implementation of PLC functionality by the controller is very flexible. This approach provides an easy integration of PLC program with motion control. For example, a motion can be started when a condition calculated by the PLC program is satisfied, and an output can be activated when a motion starts or terminates.

Unlike other motion controllers, the on-board PLC is not limited to simple bit manipulation. The ACSPL+ PLC can control digital I/O, user and standard bit flags, and expressions.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ACSPL+ PLC programs can implement several philosophies:

- Separate buffer(s) can be used to implement a PLC program.
- Mix motion programs and PLC program in the same buffers.
- Divide PLC programs into multiple buffers.

Multiple PLC programs are suitable when it is needed to have PLC programs with different scan times. In this case, the user can assign different PLC's greater priority using the **PRATE** and **ONRATE** variables. This allows concurrent execution of a slow (scanning) and fast PLC. Implement a part of the PLC program as a set of autoroutines. Because the autoroutine condition is checked each controller cycle, this provides very fast and interrupt-like response to critical conditions.

Example High-Speed PLC Program:

```
on X_FAULT.#LL; disp "X Left Limit"; ret   Display message on Left Limit fault
on abs(X_FVEL) > 20000; OUT0.5 = 1; ret   Activate output 5 when feedback velocity
                                          exceeds 20000
on IN0.9; killall; ret                    Kill all motion when input 9 is active
on X_GRTIME = 70; OUT0.3 = 1; ret         Pre-ignition: activate output 3 when 70
                                          milliseconds remain to the motion end
```

By using the ON command, the PLC is scanned every MPU cycle of the controller.

Master and Connect Functions

The master and connect functions are a unique feature of the SPiiPlus controllers. The controllers can perform independent or coordinated axes moves in any combination of up to 4 and 8 axes. Coordinate vector controlled motion includes linear, circular, sinusoidal, elliptical, helical, and inverse kinematics. The master and connect functions can be programmed to relate any mathematical expression to other axes. The master function is used for relating master positions to axis, reference, and feedback position by any mathematical expression updated at the MPU rate. The connect function is any mathematical expression that relates the logical axes to the physical axes. The master and connect function along with the programmable variables on the controllers, provide unlimited flexibility in virtually any programming application. An example of the connect function is listed below.

ACSPL+ Connect Function

Logical Axes	Connect Function	Physical Axes
x	*connect X = f(any variable)*	X
y	*connect Y = f(any variable)*	Y
z	*connect Z = f(any variable)*	Z
t	*connect T = f(any variable)*	T
a	*connect A = f(any variable)*	A
b	*connect B = f(any variable)*	B
c	*connect C = f(any variable)*	C
d	*connect D = f(any variable)*	D

The connect function allows the engineer complete flexibility in defining the characteristics of the motion profile *(the default is X = x)*. The engineer can implement a custom wave shaping algorithm, inverse kinematics or any mathematical expression.
For example, the X-axis could be programmed to follow a signal. By implementing *connect X = sin x*, The x axis will follow a sinusoidal waveform. The connect function is updated at the MPU cycle rate to ensure real time execution. In fact, all of the parameters and flags are updated every scan to guarantee deterministic performance.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Distributed Control Architecture

The SPiiPlus controllers use distributed control architecture. The MPU has one slaved Servo Processor for each set of two axes.

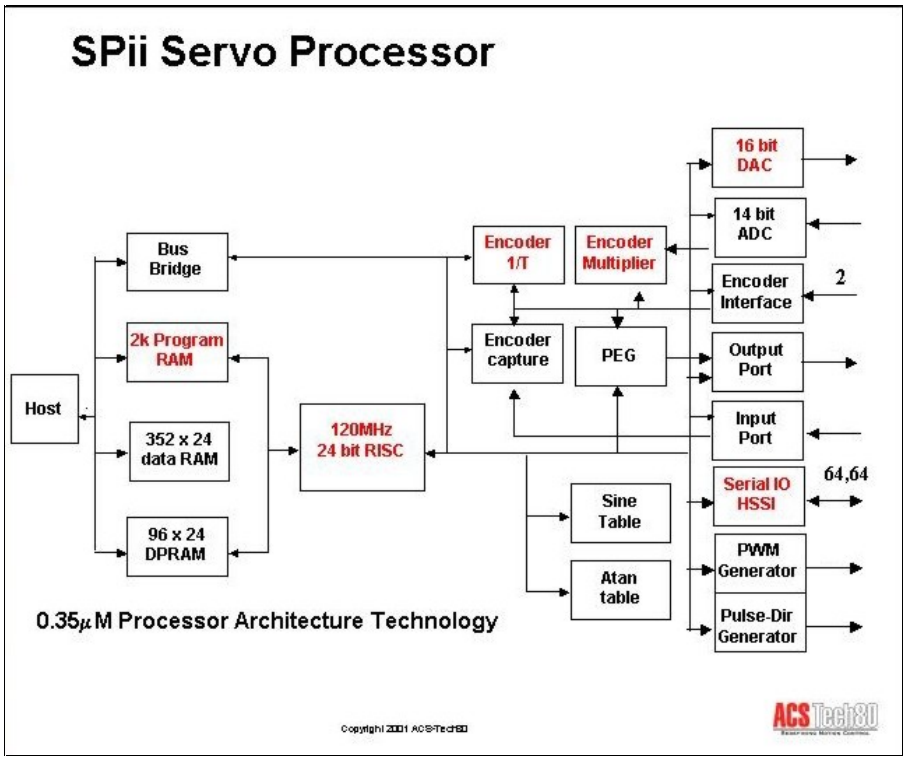
Each one of the slaved servo processors is a 24-bit fixed point RISC. The RISC guarantees that the real time control algorithms are executed in 50 microseconds regardless of the number of axes that are being processed. In addition to the servo loop, the RISC passes data to and from the MPU as required for motion profiling and data reporting. The slaved servo processor performs the real time control for the servo loop. The distributed control architecture allows the optimal performance for both the trajectory calculation and the servo loop calculation. Distributed control architecture also puts the intelligence where the most benefits can be achieved and optimized for performance. ACS-Tech80 SPiiPlus products implement a distributed architecture so the control algorithms can be optimized while maintaining an open system design.

A good distributed design minimizes data flow between the host and the processor.
The principle of the Servo Processor is to minimize the amount of data transfer to increase performance.

See following sections for details on the SPii Servo Processor.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SB2700 Servo Processor (featured in the SPiiPlus PCI)



The SPii is a state-of-the-art, second generation Servo Processor. It incorporates and a powerful 120MHz 24-bit processing unit (RISC) that is optimized for motion control applications and all the peripherals necessary to implement a full motion control system, including a digital amplifier.
The SPii is designed and optimized to act as a slave processor, in charge of the real time calculations.

SPiiPlus PCI Servo Processor Features:

Hardware Design:

- Application Specific Integrated Circuit (ASIC) design to optimize motion control performance.
- RISC and all peripherals integrated together eliminating software overhead.
- Math processing is handled by a 48 bit Multiply and Accumulate Unit (MAC) and a 24-bit Arithmetic Logic Unit (ALU)
- 2k program memory
- 96 x 24-bit dual port RAM is used to reduce communications lag time
- 352 x 24-bit RAM

Real time high order filters at sampling rates of 20kHz:

- Position loop filter (P-type, 48 bits accuracy)
- Velocity loop filter (PI-type with a second-order low pass filter and a notch filter, 48 bits)
- Velocity and acceleration feed-forward
- Anti-reset windup (integral limit)

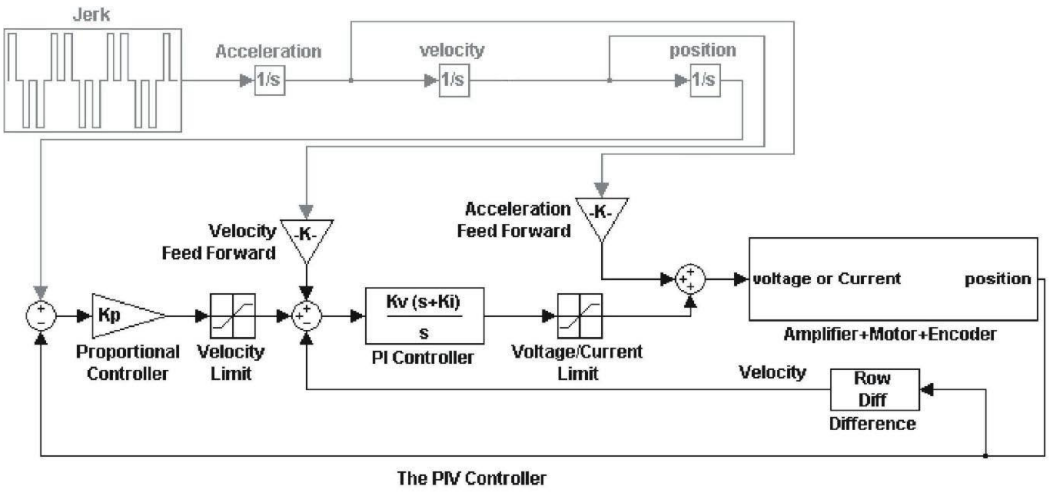
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Robust Feedback Interfaces:

- A quad B, clock-direction, up-down formats
- 3rd order input filter
- Built-in encoder error detection
- High resolution encoders with higher physical machine speeds
- 20 million counts per second for linear motor based applications
- Dual encoder feedback control (independent position and velocity loops)
- Optional Closed or Open loop analog feedback (on-the-fly changes)
- Position Event Generation (PEG). Hardware position compare within < 0.1 microsecond
- A 10-bit analog output (Scope monitoring of parameters, such as following error)
- 16-bit analog outputs for interface drives
- 14 bit analog to digital converter interface
- Program, Data, and Dual Port RAM interface to MPU (built-in host)

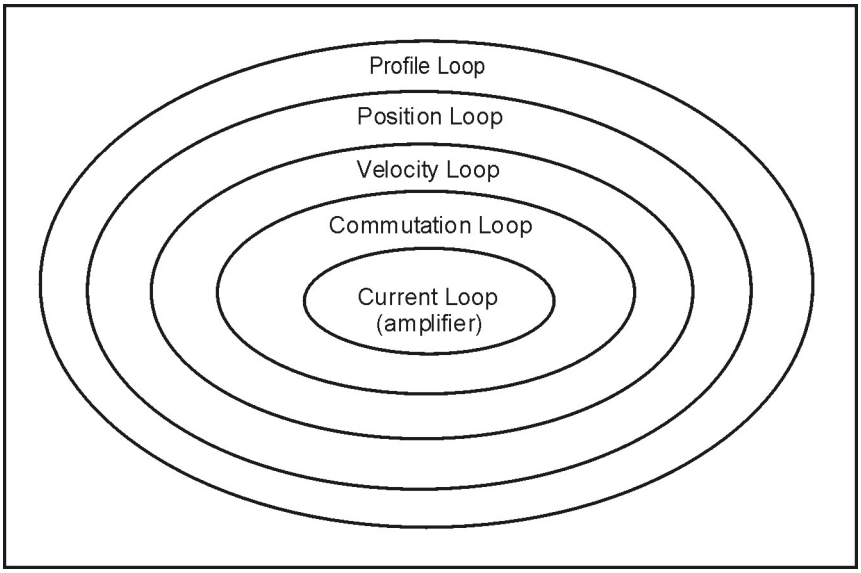
Control Algorithm

The SB2700 Servo Processor implements the servo control algorithm. The Servo Processor has been optimized for motion control applications. The algorithm is programmed into the servo processor in assembly language and optimized by the RISC to execute up to two loops at a 20kHz-sampling rate. The servo update rate is independent of the number of axes being controlled since there is one servo processor for every two axes. The servo processor algorithm can be customized in special OEM applications for support and engineering charges. However, in most applications, the standard servo control algorithm is implemented since it is already optimized for motion control. The figure below depicts a block diagram of the control algorithm.



SPiiPlus Series Control Algorithm Block Diagram

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



SPiiPlus Control Loops

With the SPiiPlus controllers, the amplifier of choice closes the current loop. The current is proportional to the torque delivered by the servomotor. Amplifiers with built-in current limiting are preferred. The adjustment method of the system control algorithm is based on working from the inner loop to the outer loop. The steps are as follows:

1 Tuning of the Current loop on the amplifier of choice
2 Tuning of the Commutation loop
3 Tuning of the Velocity loop on the controller
4 Tuning of the Position loop on the controller
5 Polishing to improve the overall performance with the Profile loop.

Features of the Control Algorithm:

- High bandwidth digital controller for speed and accuracy.
- Fully digital position and velocity control at 20kHz sampling rate independent of the number of axes, for excellent dynamic and tracking performance.
- Dual loop control with two encoders (one mounted on the motor and one on the load for accurate belt-driven and lead-screw based applications).
- Tight control and automatic velocity feedforward control
- Programmable automatic routines for fault, error, and exception.
- Real-time data collection. Programmable sampling rate up to 1kHz.
- Analog inputs with 14-bit resolution are sampled at 20kHz rate and can be used as feedback.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

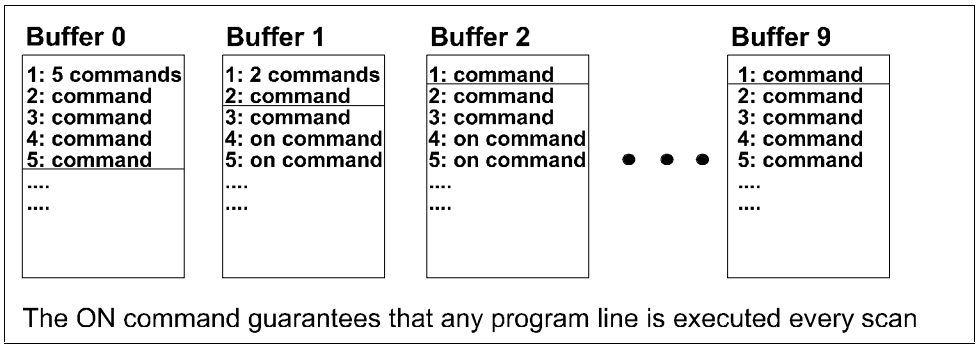
Autoroutines

Autoroutines are directly related to safety controls since they allow high-speed PLC execution of conditional responses. Autoroutines do not have to be designated to error handling but typically are used in exception handling conditions. Autoroutines are automatically implemented based on a programmable event trigger. Autoroutine event triggers include logical bits or expressions. Inputs, outputs, local, global, user, and standard variables can be used in the expression. The expression in the autoroutine header is verified each MPU cycle. Therefore, autoroutines are interruptible responses. Autoroutines are designated by the "on" command.

Autoroutine Example:

```
ON X_FAULT.#DRIVE          REM: ACTIVATE X AXIS DRIVE FAULT FLAG
Disp "X Driver Alarm"      REM: Display to the screen
ret                        REM: Return
```

Autoroutine Execution



Autoroutine Digital I/O Condition Examples:

on IN0.1 Start autoroutine when IN0.1 changes from 0 to 1. To start the autoroutine again, IN0.1 must return to 0 and then change to 1.

on IN0.0 & IN0.8 Start autoroutine when both IN0.0 and IN0.8 become 1. To start the autoroutine again, at least one of IN0.0 and IN0.8 must return to 0 and then both them must change to 1.

on IN0.0 | IN0.8 Start autoroutine when at least one of IN0.0 and IN0.8 becomes 1. To start the autoroutine again, both IN0.0 and IN0.8 must return to 0 and then at least one of them must change to 1.

on IN0 & 0x0100 The same as above. Start autoroutine when at least one of IN0.0 and IN0.8 becomes 1. To start the autoroutine again, both IN0.0 and IN0.8 must return to 0 and then at least one of them must change to 1.

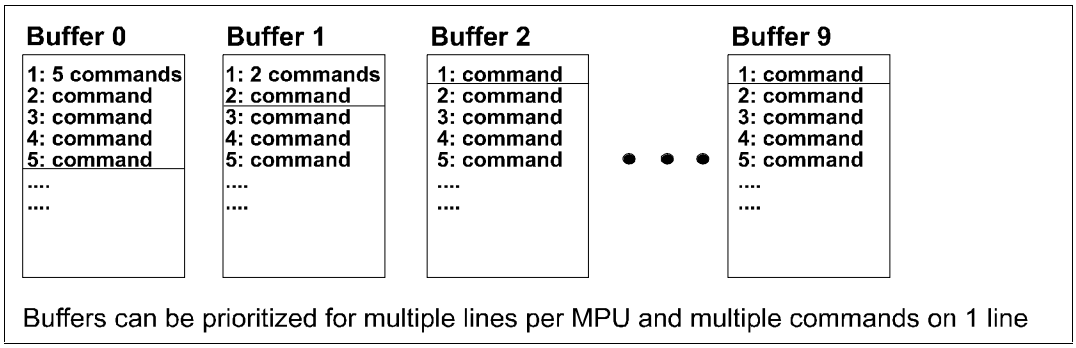
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SPiiPlus Series: FIRMWARE

Firmware Overview

The on-board firmware command set is highly dynamic and flexible. Virtually any number of variables or mathematical expressions can be programmed and solved at the MPU cycle rate (MPU cycle is 1 millisecond). The ACSPL+ programming language is compiled by the MPU before execution. It guarantees fast execution and reduced processing overhead. There are 10 multi-tasked program buffers. The programmer is guaranteed that one line of every program will be executed in the same MPU cycle. The program buffers can be prioritized. For example, a program buffer can be programmed to execute five lines in one MPU and one line of every other program. Also, up to 10 ACSPL+ commands can reside on the same program line (see the following diagram).

Buffer Command Prioritization



Firmware Expressions

ACSPL+ programming is similar to the C programming language. There is a full set of arithmetical, logical, and comparison operators. The programmer has complete control and flexibility with ACSPL+. ACSPL+ supports parametric programming. Any axis can be defined and programmed as a global variable. The axis can be controlled from any program or even multiple programs. Expressions are practically unlimited in length (1000 characters) and complexity.

ACSPL+ Expressions can define:

1. Numerical values (i.e., target point coordinates)
2. Logical values (i.e., condition for branching)
3. Axis or axes groups for motion

For example, the feedback position can be monitored with the standard variable FPOS. The FPOS parameter is an eight-element array for the eight axes. A simple program could monitor all the axes feedback positions:

```
Global real variable1              REM: declare real variable1
Variable1 = 0                      REM: Initialize variable1
    Loop 8                         REM: Loop 8 times
        Print FPOS(variable1)      REM: Print Feedback Position
        Variable1 = variable1 +1   REM: Increment counter
    End                            REM: End of Loop
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ACSPL+ Expression List

Expression types
- Automatic type conversion
- Using arithmetical expression as logical

Arithmetical operators +, -, *, /
Compare operators =, <>, >, >=, <, <=
Logical operators &, |, ^
Bitwise operators &, |, ^, ~
Bit selection operator (dot)
Complex expressions
- Operators precedence
- Brackets

Usage of expressions
- Assignment
- Function arguments
- Condition in commands if, while, till
- Autoroutines
- Axis specification using expression

Syntax summary
Assignment
- Assignment to scalar variable
- Assignment to array element
- Assignment to bit

Program flow commands
- if, else, elseif, end
- while
- loop, end
- goto
- call, ret

Synchronization commands
- till
- wait

PLC (Autoroutines)
- on CONDITION
- ret

Program management commands
- start
- stop
- stopall
- pause, resume
- enableon, disableon

User interaction commands
- disp
- input

Axes and motors management
- set
- connect
- group, ungroup
- enable, disable

Motion commands
- Axis specification
- Ptp (point-to-point)
- jog
- mptp, point, ends
- master, slave (gearing)
- mseg (camming), projection, line, arc1, arc2, stopper

Motion management commands
- go
- halt
- kill, killall
- break
- imm

Data collection commands
- dc, stopdc

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In general, the ACSPL+ programming language will support virtually any expression. Important notes on the determination of expressions are:

- Integer and Real calculations
- Automatic type conversion:
  - Left-side terms and right-side expressions may be of different types: bit (logical), arithmetical, integer, or real in any combination.
  - If the types differ, the type of calculated right-side expression is automatically converted to the type of left-side term.
  - Integer to Real conversions are exact.
  - Real to Integer conversions are rounded to the closest integer.

Valid ACSPL+ Expression Examples:

```
V0 = (V1 + V2/2) * sin (V4)                    REM: Expression in assignment
if (V0 > 0) & IN0.5; ptp X,2000; end           REM: Expression in IF-condition
till (X_FPOS > 10000) & (X_FPOS < 12000)       REM: Expression in TILL-condition
V2 = 1; loop V1/2; V2 = V2 * 2; end            REM: Expression in LOOP condition
on (IN0.3 & S_FAULT.#ES); disableall; ret      REM: Expression in autoroutine
```

Firmware Functions

The unique design of the SPiiPlus controllers allows a full set of arithmetical functions. The functions can be included in any valid expression, compiled, and updated at the MPU cycle rate (default is 1 millisecond). A full set of functions is included with the ACSPL+ programming language.

ACSPL+ Valid Arithmetical Functions

- abs
- acos
- asin
- atan
- atan2
- ceil
- cos
- exp
- floor
- hypot
- ldexp
- log
- log10
- pow
- sin
- sqrt
- tan
- sign

Signal processing functions

- deadzone
- edge
- delay

Statistical functions

- min
- max
- avg

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Variables and Flags

With ACSPL+, virtually any number of variables or mathematical expressions can be programmed and solved at the MPU cycle rate (MPU cycle is 1 millisecond). There are user defined variables and standard variables.

User Variables Features:

- Variable count only limited to the size of memory (practically unlimited)
- Arbitrary length of variable name. The user can select any sequence of letters and digits
- Integer or Real (64 bit floating point)
- Scalar, one- and two-dimensional arrays

- **Local User Variable Examples:**

```
int LIScalar                REM: integer variable LIScalar
int LIArray1(20)            REM: integer array of 20 elements
int LIArray2(10)(100)       REM: integer two-dimensional array
real LRScalar               REM: real variable LRScalar
real LRArray1(20)           REM: real array of 20 elements
real LRArray2(10)(100)      REM: real two-dimensional array
```

Local variables are only visible within the program (buffer) in which it is defined

- **Global User Variables Examples:**

```
global int GIScalar             REM: global variable GIScalar
global int GIArray1(20)         REM: global array of 20 elements
global int GIArray2(10)(100)    REM: global two-dimensional array
global real GRScalar            REM: global real variable GIScalar
global real GRArray1(20)        REM: global real 20 element array
global real GRArray2(10)(100)   REM: global real two dimensional array
```

Global variables are common for all programs in the controller

Important notes regarding ACSPL+ variables:

- Local and global user variables must be declared before they can be used
- If a user variable is one-dimensional array, it requires one index
- If a user variable is two-dimensional array, it requires two indices.

Standard Variables

Standard variables are the variables predefined in the controller. A standard variable can be used in any ACSPL+ program without explicit definition. Redefinition of standard variable name is prohibited and causes program error. All standard variables have global scope (i.e. all references to a global variable in any program buffer refer to the same variable). A list of the pre-defined standard variables and flags is in the appendix of this section. (i.e. Feedback Position = FPOS; Velocity = VEL).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Flow Programming

The ACSPL+ programming language includes a full set of flow programming controls. This allows the programmer complete flexibility in designing motion control programs. In normal program flow, the compiled command lines are executed sequentially in the order of appearance. Program flow commands change the order of the command execution. The following program flow commands are available:

Program Flow List:

if *expression*
elseif *expression*
else
end
while *expression*
loop *expression*
call *label or number*
ret
goto *label or number*

The commands if, while and loop open a structure that may include a number of commands. The structure may be in the following format:

structure:
if *structure*
while *structure*
loop *structure*

In general, the flow programming tools allow a complete structure to be engineered by the programmer. The flow programming tools help the designer implement a logical label based program using a modular design. For additional details on the flow programming architecture, please consult the ACSPL+ programming manual.

Safety Controls

The ACSPL+ has extensive built-in safety controls. The safety controls are important to avoid costly component failures. The controller has a calculated MTBF of >50,000 hours and therefore will not typically be the component that fails. Other high power or mechanically operated components will fail several times before the controller will fail. In the unlikely event that the controller does fail, the controllers are supplied with a watchdog signal that should be interlocked with the emergency stop circuitry on the machine.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The safety controls provide built-in protection so that a fail-safe condition can be achieved. The safety controls include the following:

- User errors in programming
- Loss of feedback
- Power amplifier fault
- Motor malfunction, temperature or current
- Safety limits
- E-stop

Summary of built in faults:

Fault	Fault category	Fault explanation
#LL	Motor	Left Limit. The controller sets the fault bit if the left limit switch is activated.
#RL	Motor	Right Limit. The controller raises the fault bit if the right limit switch is activated.
#LL2	Motor	Left Limit 2. The controller raises the fault bit when the preliminary left limit switch is activated.
#RL2	Motor	Right Limit 2. The controller raises the fault bit when the preliminary right limit switch is activated.
#SLL	Motor	Software Left Limit. The controller raises the fault bit when the motor reference position (**RPOS**) is smaller than the software left limit margin (**SLLIMIT**).
#SRL	Motor	Software Right Limit. The controller raises the fault bit when the motor reference position (**RPOS**) is greater than the software right limit margin (**SRLIMIT**).
#PE	Motor	Position Error. The controller raises the fault bit when the position error (**PE**) exceeds the limit. The limit depends on the motor state and is defined by the following variables: **ERRI** if the motor is idle (not moving) **ERRV** if the motor is moving with constant velocity **ERRA** if the motor is accelerating or decelerating
#CPE		Critical Position Error. The controller raises the fault bit when the position error (**PE**) exceeds the value of the critical limit. The limit depends on the motor state and is defined by: **CERRI** if the motor is idle (not moving) **CERRV** if the motor is moving with constant velocity **CERRA** if the motor is accelerating or decelerating

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Summary of built in faults: (Continued)

Fault	Fault category	Fault explanation
#ENC	Motor	Encoder Error. The controller raises the fault bit when the primary encoder does not work properly.
#ENCNC	Motor	Encoder Not Connected. The controller raises the fault bit when the primary encoder is not connected.
#ENC2	Motor	Encoder 2 Error. The controller raises the fault bit when the second encoder does not work properly.
#ENC2NC	Motor	Encoder 2 Not Connected. The controller raises the fault bit when the second encoder is not connected.
#DRIVE	Motor	Driver Alarm. The controller raises the fault bit when the signal from the driver reports a failure.
#HOT	Motor	Motor Overheat. The controller raises the fault bit when the motor's temperature sensor indicates overheat.
#VL	Motor	Velocity Limit. The controller raises the fault bit when the absolute value of the reference velocity (**RVEL**) exceeds the limit defined by the **XVEL** parameter.
#AL	Motor	Acceleration Limit. The controller raises the fault bit when the absolute value of the reference acceleration (**RACC**) exceeds the limit defined by the **XACC** parameter.
#CL	Motor	Current Limit. The controller raises the fault bit when the RMS current exceeds the limit value defined by the **XRMS** parameter.
#SP	Motor	Servo Processor Alarm. The controller raises the fault bit when the axis Servo Processor loses synchronization with the main processor. The fault indicates a fatal problem in the controller.
#ES	System	Emergency Stop. The controller raises the fault bit when the ES signal is activated.
#PROG	System	Program Fault. The controller raises the fault bit when a run time error occurs in one of the executed ACSPL+ programs.
#MEM	System	Memory Fault. The user application requires too much memory.
#TIME	System	Time Fault. The user application consumes too much time in the controller cycle. Execution rate of ACSPL+ program and/or the quality of the motion control might deteriorate.
#INT	System	Servo Interrupt. The servo interrupt that defines the controller cycle is not generated. The fault indicates a fatal problem in the controller.

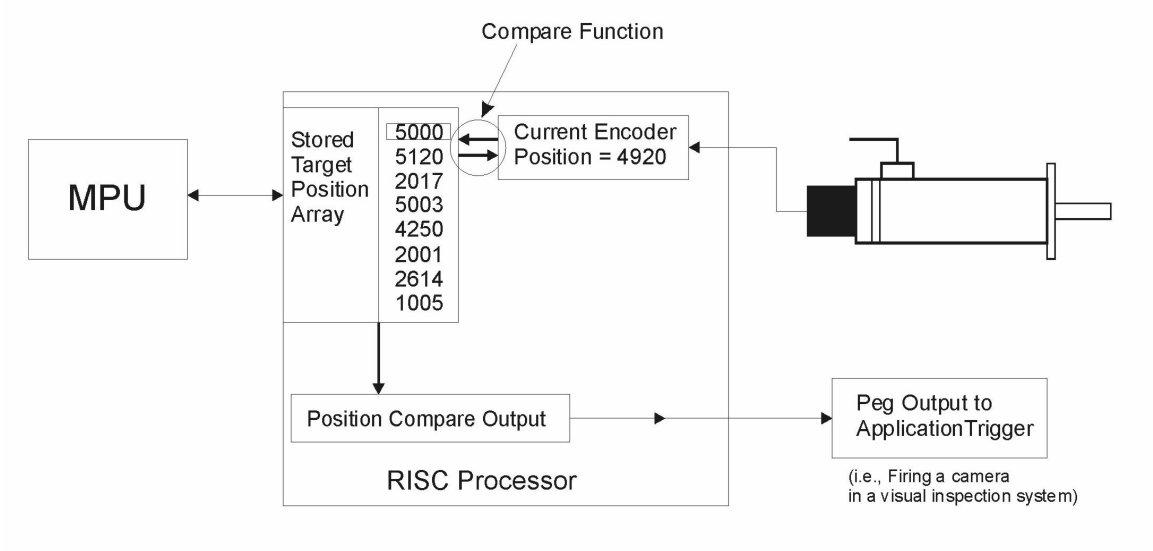
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Real Time Position Event Generator (PEG)

The purpose of the Position Event Generator (PEG) is to generate accurate position based events. It provides the ability to generate a pulse whenever a fixed position interval has passed (Incremental mode), and to generate an event (state of selected outputs plus a pulse) at a set of pre-defined positions (random mode).

In both modes, it is possible to create a stream of pulses when a match occurs with a programmable time based interval. This feature facilitates the use of high-resolution Sin-Cos encoders in high performance inspection systems. The Sin-Cos encoders can be used instead of laser interferometers without degrading the performance of the system.

Random Mode

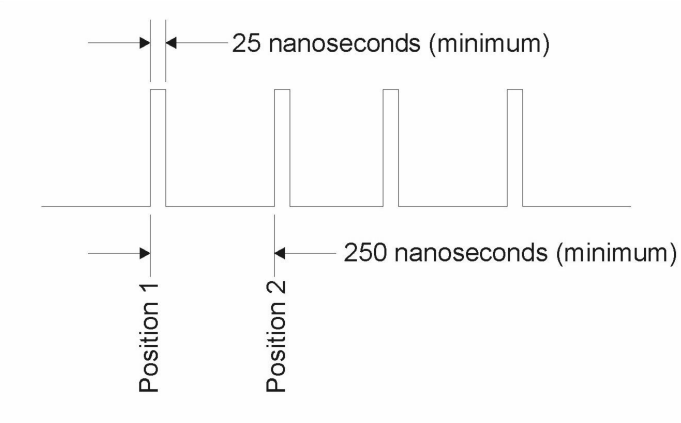


Each target position can be defined at up to 10,000 points. Once the axis reaches a trigger point, the state of the output port changes instantaneously (electronic gate delay). In addition, a pulse with a programmable period is generated at each trigger point by an additional output. Eight events can be defined per 0.001 second via an 8-deep FIFO buffer in each SPii. The minimum time interval between two adjacent events is less than 0.25 microseconds.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Incremental Mode

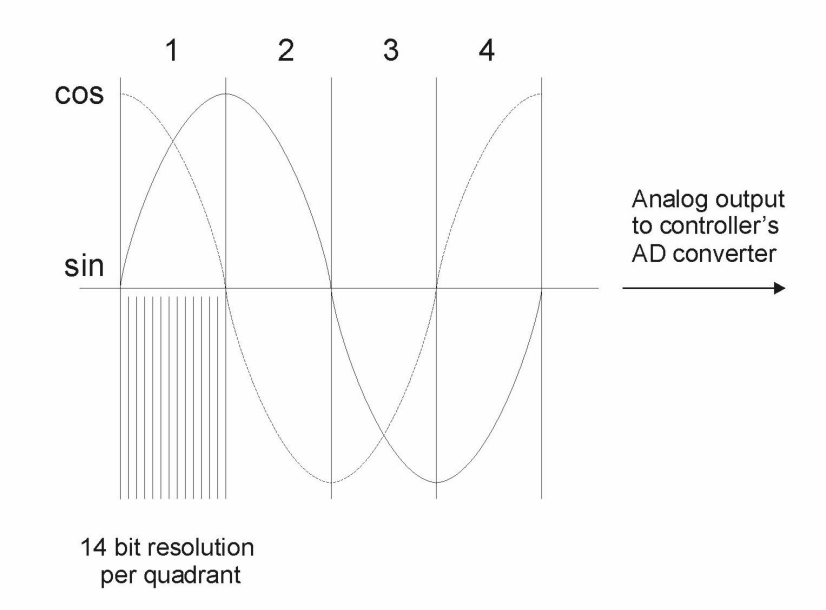
In this mode, a programmable pulse is generated whenever a predefined distance is passed. The start point and the distance between the points are programmable. The minimum time interval between two adjacent events is less than 0.25 microseconds.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Sin-Cos Encoders

An optional built-in interface for Sin-Cos encoders (1v ptp) is available for each encoder channel. The advantage of analog encoders is their very high positional resolution. The following diagrams give a general explanation of how the Sin-Cos encoders achieve this.



The encoder produces analog sine and cosine waves for each encoder line. The analog signals are sampled with high-speed 14-bit analog to digital converters (ADC). As shown above, each quadrant of the waveform has a resolution of 14 bits, for a total resolution of 16 bits per each encoder line:

14 bit resolution X 4 quadrants = 16 bit resolution

or

$2^{14} \times 2^{2} = 2^{16}$ bits (65,536 possible positions)

This is a theoretical multiplier. In reality, a system like this is limited to 13-bit resolution by line noise, signal attenuation, etc. Thirteen-bit resolution yields a maximum real-world multiplier of 8,192 bits. No matter what resolution is selected, there are no limits imposed on the physical speed by the controller.

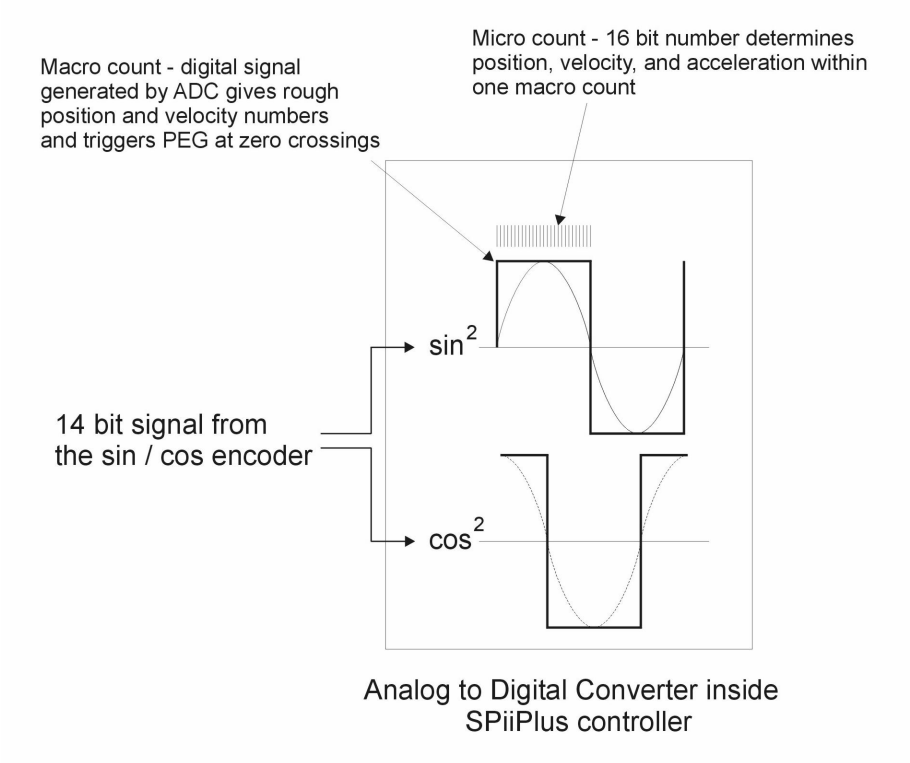
The SPiiPlus controllers support 1Vp-p Sin-Cos encoders. The least significant bit of the ADC represents 0.0001V. In most cases the noise component of the sampled signal is higher than that and therefore it introduces jitter to the position at the high resolution.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In tests we have conducted, it was possible to use multiplication of 8192 and to stabilize the system to +/- 1 count. With a 2000 line rotary encoder it translates to 16,384,000 counts per revolution. The stability achieved was 0.08 arc-second. With a linear encoder with 250 lines per mm, the resolution is ~0.5 nanometer!

The frequency of the Sin-Cos encoder is limited to 500kHz at any resolution. So the maximum speed for the 2000 line rotary encoder is 15,000 rpm [(500,000/2000)*60]. The maximum speed of the 250 lines/mm (4 micrometers per line) linear encoder is limited to 2 meter/second [(4*500,000/1000,000)].

Motion data is determined by two factors, the Macro count and the Micro count:



Analog to Digital Converter inside SPiiPlus controller

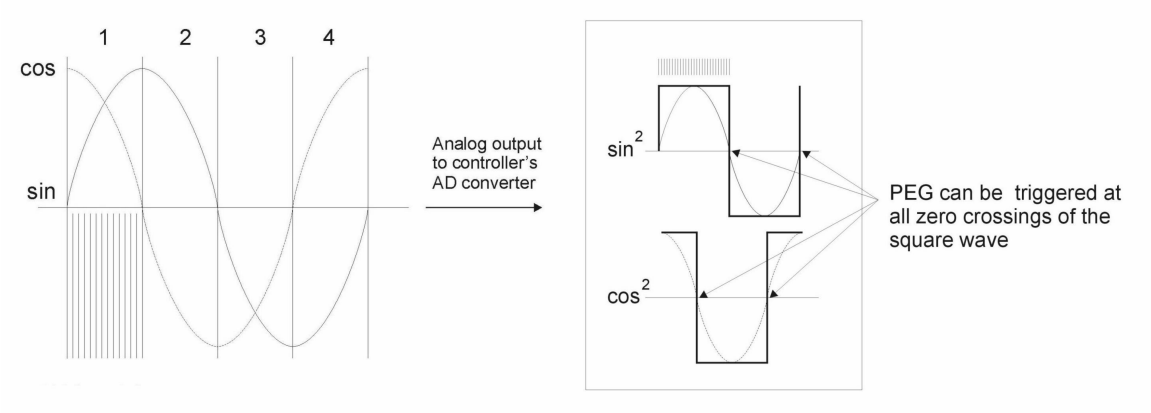
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Sin-Cos Encoder and PEG

Since the waveforms of the Sin-Cos encoders are not ideal sine and cosine, any multiplication cannot achieve a uniform size of each count. Activating the PEG based on the position that is generated based on the output of the multiplier might generate inaccurate spacing of events. Activating the PEG should be based on full number of encoder lines. Then the event spacing is as accurate as is the spacing of these lines. This is the theoretical limit. As a result, high accuracy inspection systems were using laser interferometers as position feedback instead of encoders. Laser interferometers are at least 10 times more expensive than linear optical encoders. The SPiiPlus PEG overcomes this limitation and provides the ability to use encoders instead of the high cost interferometers in high accuracy applications, such as wafer inspection. The SPiiPlus PEG provides the ability to generate a stream of pulses at each specified position with a programmable time interval. So when a linear stage is moving at a tight controlled constant velocity, and a PEG stream of pulses are generated based on each encoder line the position spacing of the pulses is very accurate and with sufficient resolution.

Example:
A linear stage with 500 lines/mm Sin-Cos encoder (with 8192 multiplication) moves at a speed of 62.5 mm/sec with constant velocity of +/-0.25% (line to line spacing). It is required to generate a 0.1 µsec pulse at each position increment of 62.5 nanometer. The velocity command will be 256,000,000 counts/second.

The PEG is programmed to generate an event at intervals of 8192 counts (= 1 encoder line) Each event consists of a stream of 256 pulses with time interval of 1 µsec. The result will be a pulse each 62.5 nanometer with position accuracy of each pulse of 5 (2000*0.25/100) nanometer.



PEG generation can happen at any of the leading or trailing edges of the macro count pulse where there is a positive to negative transition.

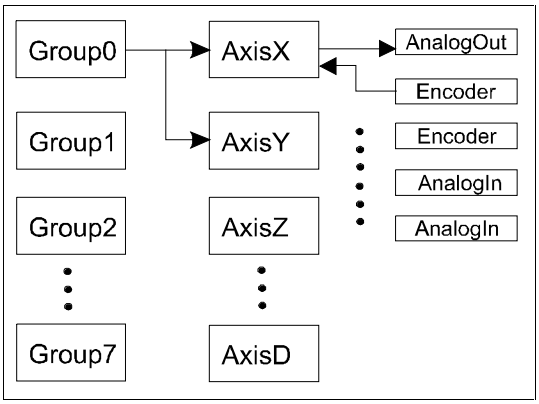
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Coordinate Systems

The SPiiPlus Series are capable of handling up to eight coordinate systems. The coordinate systems are referred to as groups. The group is the motion profiler and has built-in parameters and flags to aid the programmer in controlling the desired motion profile.

The main purpose of the group is to define a multi-axis coordinate system. Per the diagram below, up to eight groups can be programmed. Any combination of axes can be allocated to a group definition.

The default definition of the axis is an analog output and an encoder input. No programming is required to set up the axis in the default condition. However, the axis can be defined to have dual loop encoder feedback (position and velocity), or analog feedback and combinations thereof.



The following examples are valid group definitions for ACSPL+

group X, group Y, group Z, ….
group XY, group XZ, group XT, ….
group XYZ, group XYT, group XYA, …..
group XYZT, group ABCD, ….
group XYZTA, group XYZAB, ….
group XYZTAB, group XYZABC, ….
group XYZTABC, group XYZTACD, ….
group XYZTABCD

A group in any combination of up to eight axes can be defined. Multiple groups can be defined for up to eight different coordinate systems. An axis can only belong to one group at a time. However, a group can be split and group definitions can be changed on the fly.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Axis/Motor Management

Axis and motor management commands prepare the controller for motion and arrange the controller environment to correspond to the user system.

Axis-motor-commands:

enable *command*	**REM: activates one or more drives for physical motion.**
disable *command*	**REM: disables one or more drives from physical motion.**
set *command*	**REM: appoints a current value of the feedback, reference or master position.**
group *command*	**REM: defines a coordinate system.**
split *command*	**REM: splits a coordinate system.**
splitall *command*	**REM: splits all coordinate systems.**
connect *command*	**REM: defines a relationship between motors and axes.**
master *command*	**REM: defines a master value for an axis.**

Motion Commands

Motion management commands comprise the commands that start, stop or modify a motion profile. Commands in this group are not motion specific and manage any type of motion mode.

Motion-management-commands:

Go *command*	**REM: starts a motion that was created using the w suffix**
Halt *command*	**REM: terminates a motion using a deceleration profile**
Kill *command*	**REM: terminates a motion using a kill deceleration profile**
Killall *command*	**REM: terminates all currently executed profiles**
Break *command*	**REM: terminates a motion immediately with no deceleration**
Imm *command*	**REM: provides on-the-fly change of motion parameters**

The motion-management commands are invaluable to creating motion profiles. The commands are used in conjunction with the motion modes (see next sections). The motion-management and axis-motor commands can be used in any mode, group, or axis.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Point-to-Point Motion

Point-to-point (PTP) motion provides positioning to specified target point. The **ptp** command is used to create point-to-point motion control. By using the Group command, we can generate a simple 3-axis coordinate system and perform a multi axis point-to-point motion.

Point-to-Point Examples

```
group XYZ                       REM: Create a 3 axis coordinate system
ptp XYZ, 1000, 2000, 3000       REM: Move XYZ to 1000, 2000 and 3000 units
```

The **imm, go, halt, kill, break** can be applied to point-to-point motion control to customize the profile to the desired conditions. The **ptp** command accepts the following suffixes:

r	**The target point is specified relative to the start point**
w	**Create the motion and wait for the go command**
v	**Use the specified velocity instead of the default velocity**

All suffixes are valid for multi-axis motion as well as for single axis motion. If the **v** suffix is used, the velocity value is specified after the last target value:

```
ptp/v X1000, 15000              REM: Velocity is 15000 pulses/second for ptp
```

More than one suffix can be used in a ptp move profile.

```
group XYZ                       REM: Create coordinate system
ptp/rv XYZ, 1000, 2000, 3000, 15000  REM: Relative move @ 15000 pulses/second
```

Multi-Point Motion

Multi-point motion provides sequential positioning to a set of points, optionally with dwell at each point. The following commands control multipoint motion:

mptp	**Create multi-point motion**
point	**Add the next point to the point sequence**
ends	**Terminate the point sequence**

Again, the commands **imm, go, halt, kill, break** can be applied to multi-point mode.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The **mptp** command accepts the following suffixes:

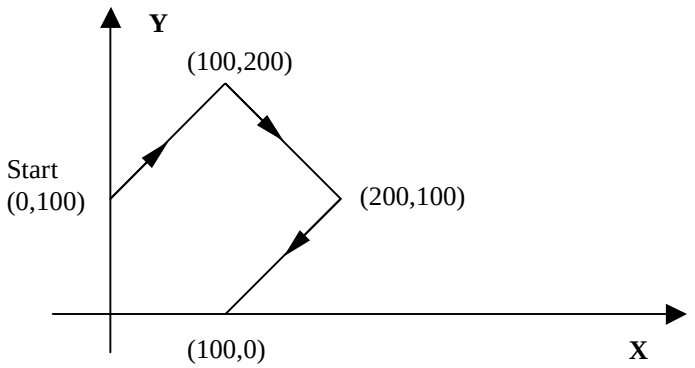
r	**The coordinates of each point are relative to the previous point.**
w	**Create the motion, but wait for the go command.**
v	**Use the velocity specified for each point instead of the default velocity.**
c	**Use the last point sequence as a cyclic array.**

The **mptp** command specifies axis and dwell time: The command **mptp x, 1000** creates a multi-point motion of the X-axis and specifies dwell of 1000 msec in each point. If dwell is not required, the second argument along with comma may be omitted. The **mptp** command itself does not specify any point, so the created motion never starts immediately. Any number of the **point** commands must follow the **mptp** command in order to specify the point sequence.
To execute multi-axis motion, first a suitable axis group must be created. Then the **mptp** command can specify all or several from the group's axes.

Multi-point Example

```
group XY                    REM: Create XY coordinate system
mptp XY                     REM: Create multi-point motion for XY group
point XY, 0, 1000           REM: Add first point
point XY, 100, 200          REM: Add second point
point XY, 200, 100          REM: Add third point
point XY, 100, 0            REM: Add fourth point
ends XY                     REM: End the point sequence
```

The **mptp** command creates the multi-point motion. The motion does not start, because no point is defined yet. After the first **point** command the motion can start, and really starts if the axis group is idle (not involved in some previous motion). The four **point** commands specify the following sequence:



The controller performs sequential positioning to each point. The **ends** command informs the controller that no more points will be specified for the former motion. **Similar to the ptp mode, the mptp suffixes can be used in any combination along with the mptp command. The mptp mode also supports cyclic motion profiles.** Suffix **c** provides cyclic execution of multipoint motion. The **mptp** command creates a motion that after positioning to the last point of sequence does positioning to the first point and continues through the same sequence. Cyclic multipoint motion does not finish automatically. One of the commands **halt, kill, break** must be used in order to stop cyclic motion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Jogging

Jog motion is a motion with constant velocity and no end point. The motion continues until the next motion command stops it or the motion fails because of limit switch activation or other condition. The following command initiates jog motion:

Jog move at a constant velocity

The commands **imm, go, halt, kill, break** apply to jog motion as well as to any other motion type. The **jog** command accepts the following suffixes:

w Create the motion, but do not start until the go command
v Use the velocity specified in the command instead of the default velocity

The jog command follows the exact same format as the ptp and mptp modes.

Valid Jog Command Examples:

1) **jog x (jog, x+)** **REM: Jog the X axis forward using the X_VEL velocity**
jog, x- **REM: Jog the x axis backward using the X_VEL velocity**
jog/v x, 30000, + **REM: Jog the x axis forward using 30000 pulses/second**

2) **GROUP XYZ** **REM: CREATE AN XYZ COORDINATE SYSTEM**
jog xyz, -++ **REM: Jog the axes in the appropriate direction using the X_VEL as the multi-axis vector velocity**

As any other motion, jog motion may be terminated by the halt, kill, break commands. Unlike any other motion, jog motion also terminates when the next motion command for the same axis executes.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Slaved Motion

Slaved motion is both a position and velocity lock mode. It is similar to other motion controllers electronic gearing and electronic cam functions, but with advanced features. Slaved motion causes the axis to follow a master value. Slaved motion is a simple form, more sophisticated forms are presented by the Slaved mode of the Segmented and Spline motion. The following command initiates slaved motion:

master	**Define master value for the axis**
slave	**Create multi-point motion**

The commands **go, halt, kill, break** apply to slaved motion as well as to any other motion type.

The **slave** command accepts the following suffixes:

w	**Create the motion but wait until the go command**
p	**Use position lock instead of velocity lock**
t	**Stall in the extreme points**

The **master** command defines a formula for calculating the axis master position (**MPOS**). In the simplest case the master position must follow the feedback position of another axis:

```
master X MPOS = Y FPOS      REM: Gearing ratio of 1:1
```

When the command executes, the controller stores the formula, specified right to the equal sign, and then calculates the master position **X_MPOS** according to this formula (in this example, simply assigns the current **Y_FPOS** value to **X_MPOS**). The controller provides calculation each controller period, independently on the program execution. Even if the program, that executed the **master** command, terminates, the controller continues calculating the last specified master expression, until the next **master** command for the same axis is executed.

A more sophisticated example of the **master** command connects the axis to another axis feedback with a scale factor:

```
master X MPOS = 2.3 * Y FPOS        REM: Gearing ratio of 2.3 : 1
```

There are no restrictions on the assignment of the master position. The master may be assigned to any parameter in the controller, including user parameters. The master value may be connected to the sum of two or more axes, to an analog input, etc.

The **slave** command creates a motion slaved to the master value of the specified axis. Only an individual axis, not a group, is allowed in the **slave** command. The created motion starts immediately if the axis is idle; otherwise, the motion wait in the motion queue until all motions created before for the axis terminate. This command creates a slaved motion of the X-axis:

```
slave X
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Slaved Motion

The motion is always slaved to the master value of the specified axis. Therefore, the **master** command must precede the **slave** command for the same axis. Except the transient process, when the motion is asynchronous, in slaved motion the **APOS (actual position)** axis reference follows the **MPOS (master position)** axis master value strictly or with a constant offset:

APOS = MPOS + C

The **C** is constant in velocity lock mode and is zero in position lock mode. Once started, slaved motion terminates only if a failure occurs or one of the commands **halt, kill, break** is executed. The **halt** and **kill** commands provide deceleration to zero and then the next motion starts. If no next motion was created, the axis becomes idle. The **break** command provides smooth transition to the next motion without stop, if the next motion was created beforehand.

Slaved Motion Synchronization

Normally, in slaved motion the **APOS** axis reference follows the **MPOS** master value strictly or with a constant offset (i.e. the slave is synchronized to the master). However, there are two cases when synchronism is not attainable: The slaved motion starts, and positions (position lock) or velocities (velocity lock) of the master and slave differ. The motion starts as asynchronous. The motion was synchronized, but the acceleration of the master exceeds allowed limit (the **XSACC** variable of the axis). The slave comes out from synchronism.

In both cases, the motion continues asynchronously; i.e., the correspondence between the **APOS** and **MPOS** appears broken. The controller tries to regain synchronism by moving the slave after the master with the maximal allowed motion parameters. When the slave overtakes the master, the synchronism is reestablished and then the motion continues as synchronous.

The slave dynamics is governed by the variables of the slaved axis. The following variables have effect:

XSACC	**Maximum acceleration of the synchronous motion**
SYNV	**Allowed difference in master and slave velocities**
JERK	**Default electronic jerk value**
ACC	**Default electronic gearing acceleration value**
VEL	**Default electronic gearing velocity**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Slaved Motion: Velocity Lock vs. Position Lock

The **slave** command without **p** suffix activates a velocity-lock mode of slaved motion. When synchronized, the **APOS** axis reference follows the **MPOS** with a constant offset (APOS = MPOS +C). The **p** suffix, attached to the **mseg** command, activates a position lock mode of slaved motion. When synchronized, the **APOS** axis reference follows the **MPOS** strictly (APOS = MPOS).

When the motion is asynchronous for any reason (see above), the controller tries to regain synchronism by moving the slave after the master with the maximal allowed motion parameters. Distinction between position lock and velocity lock manifests at the moment of regaining synchronism:

Velocity lock motion switches in synchronism when the slave velocity reaches the master velocity (with allowed error defined by the **SYNV** variable of the slaved axis). At this moment the difference between the master position and the slave position is latched as the constant offset **C**, which then remains unchanged as long as the motion is synchronous.

Position lock motion switches in synchronism when the slave position overtakes the master position, i.e. when **APOS = MPOS**. Please note: Each time the motion looses and regains synchronism, the velocity lock offset **C** may latch a different value. Under the same conditions, the position lock motion reestablishes the strict equality **APOS = MPOS** each time.

Slaved Motion: Stalled

Without the **t** suffix, the **slave** command applies no limits on the slaved axis. The axis follows the master everywhere, unless some failure occurs, e.g. the limit switch is activated.

The **slave** command with the **t** suffix requires two additional parameters. These parameters define a permitted interval of the slaved axis motion. For example:

slave/t X, -1000, 2000

The command allows the **X** axis motion only within the interval (-1000, 2000). When the **APOS** axis reference approaches any of the two extreme points, the slave comes out from synchronism and stalls at that point until the **MPOS** master value allows it to regain synchronism. For velocity lock, synchronism regains when the **MPOS** changes its direction; after regaining the offset **C** may have a different value than before approaching the extreme point. For position lock, synchronism regains when the **MPOS** comes back into the permitted interval. The controller ensures a smooth approach to the extreme points and smooth return to synchronism.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Segmented Motion

Segmented motion moves axes along a continuous path. The path is defined as a sequence of linear and arc segments on the plane. Even though the segmented motion follows a flat path, it may involve any number of axes, because the motion plane can be connected to the axes with an arbitrary projection transform.

The following commands control segmented motion:

mseg	**Create segmented motion**
projection	**Set projection matrix**
line	**Add linear segment**
arc1	**Add arc segment**
arc2	**Add arc segment**
stopper	**Pause the motion in the intermediate point**
ends	**Terminate the point sequence**

The commands **imm, go, halt, kill, break** apply to segmented motion as well as to any other motion type. The **mseg** command accepts the following suffixes:

w	**Create the motion, but wait until the go command**
v	**Use velocity specified for each segment instead of the default velocity**
c	**Use the segment sequence as a cyclic array**
s	**Velocity Lock (slaved motion) to the leading master velocity**
p	**Position Lock conforming to master position**
e	**Extrapolated: if a master value travels beyond the specified path, the last or the first segment is extrapolated.**
t	**Stalled: if a master value travels beyond the specified path, the motion stalls at the last or first point.**

The **e** and **t** suffixes are relevant only for slaved motion and must be used with **s** or **p** suffix. Segmented motion can be executed in axis group with any number of axes.

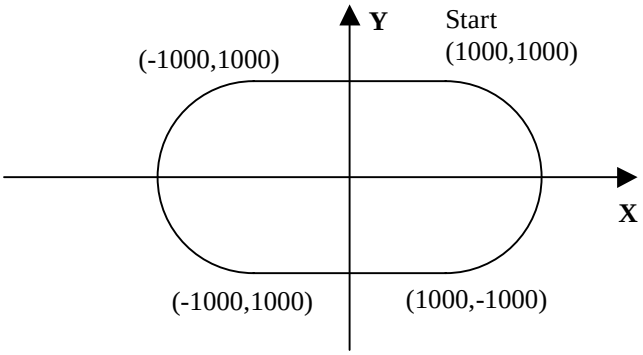
Segmented Motion Example:

```
GROUP XY                        REM: CREATE AXIS GROUP X
    mseg XY,1000,1000           REM: Create segmented motion in group X,
                                coordinates of the initial point are (1000,1000)
    arc1 XY, 1000,0,1000,–1000,– REM: Add arc segment with center (1000,0), final
                                point (1000,-1000), clockwise rotation
    arc2 XY,–1000,0,–3.141529   REM: Add arc segment with center (-1000,0) and
                                rotation angle –π
line XY,1000,1000               REM: Add line segment with final point (1000,1000)
ends XY                         REM: End the segment sequence
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Segmented Motion (continued)

The **mseg** command creates the segmented motion. The motion does not start, because no segment is defined yet. After the first **arc1** command the motion can start, and really starts if the axis group is idle (not involved in some previous motion). The four segment commands specify the following path:



The **arc1** and **arc2** commands differ by the required arguments. The **arc1** command specifies the coordinates of the center point, the coordinates of the final point and the direction of rotation (positive for counterclockwise, negative for clockwise rotation). The **arc2** command specifies coordinates of the center point and rotation angle (positive for counterclockwise, negative for clockwise rotation). The **arc1** and **arc2** may produce the same result, so the user may select one that suits the available data. The **ends** command informs the controller that no more segments will be specified for the former motion. The motion cannot finish until the **ends** executes. If the **ends** command is omitted, the motion will stop in the last point of the sequence waiting for the next point. No transition to the next motion in the queue will occur until the **ends** command is executed.

Normally, the segmented motion starts with the first segment command (**line**, **arc1**, or **arc2**). Therefore, the next segment command executes while the motion is already in progress. This is not a problem, because the program execution rate is much quicker than typical motion time.

However, it is sometimes desirable to delay the start of motion until all segments are defined. In this example, the **w** suffix is appended to the command to prevent the motion from starting:

mseg/w XY, 1000, 1000

The motion will not start until the **go XY** command is executed. The **r** suffix is not allowed with segmented motion. All coordinates in the **line**, **arc1,** and **arc2** commands are absolute in the plane. However, the whole path is relative to the point, where the axes are located when the segmented motion starts. In other words, all coordinates are absolute in the plane, but the plane is relative to the starting point. Please note the initial point specified in the **mseg** command is also absolute in the plane. Therefore, the initial point does not cause any motion to that point, but only supplies starting coordinates for the first segment. Normally, the motion program should provide a motion to the desired initial point before executing the **mseg** command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Segmented Motion Projection

As mentioned before, all coordinate systems specified in the segment commands, are absolute in the working plane. Projection is a matrix that connects the plane coordinates and the axis values in the axis group. If the axis group contains two axes and no **projection** command is specified, the controller provides a default projection that corresponds to the matrix 2x2:

```
1       0
0       1
```

The matrix directly connects the first coordinate of the working plane to the first axis of the axis group and the second coordinate to the second axis. In fact, the matrix can define rotation and scaling. The full transform includes also an implicit offset. The controller calculates the offset automatically in such a way that the initial coordinates specified in the **mseg** command match the desired axis values at the moment when the motion starts. The offset provides the full path to be relative to the starting point. If an axis group contains N axes, the controller extends the default matrix to N lines. The additional lines are filled by zeros:

```
1       0
0       1
0       0
…       …
0       0
```

The matrix connects only the first two axes to the plane coordinates. Therefore, the segmented motion will involve only the first two axes in the group. If N = 1, i.e. the **mseg** command applies to single axis, the matrix contains only the first line

```
1       0
```

In this case, the axis will follow the first coordinate of the working plane.

The user can replace the default matrix with the **projection** command. The following program fragment:

```
group XYZ
real M(3)(2)
M(0)(0) = 1; M(0)(1) = 0
M(1)(0) = 0; M(1)(1) = 1.41421
M(2)(0) = 0; M(2)(1) = 1.41421
projection XYZ, M
```

provides the working plane to be inclined by 45°. In the same way, the matrix can define arbitrary linear transform. Note: if the group contains N axis, the matrix in the **projection** command must be of size Nx2.

Segmented Motion Smoothness and Stoppers

The controller builds the motion so the vector velocity follows the smooth velocity diagram. The segments define projecting the vector velocity to axis velocities. If all segments are connected smoothly, axis velocity is also smooth. However, if the user defined a path with inflection point, axis velocity has a jump in this point. The jump might cause a motion failure due to the acceleration limit. Even if the failure does not occur, the abrupt change in velocity impairs accuracy and may be harmful to the machine.

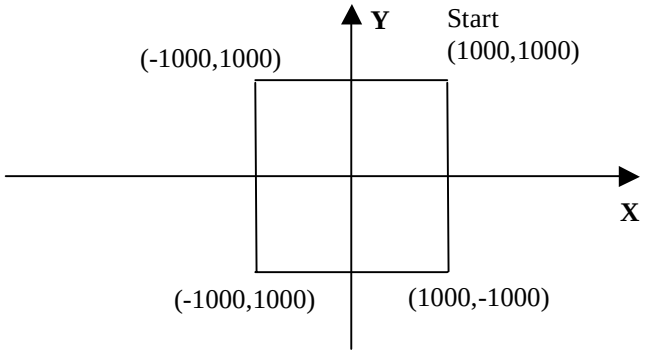
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Segmented Motion (continued)

The **stopper** command must be used in order to avoid velocity jump in the inflection points. If a stopper is specified between two segments, the controller provides smooth deceleration to zero before the stopper and smooth acceleration to specified velocity after the stopper. Consider the following program fragment:

```
group XY                REM: Create coordinate system XY
ptp XY, 1000, 1000      REM: Linear move to coordinates
mseg XY, 1000, 1000     REM: Start Segmented Motion
Line XY, 1000, –1000    REM: Insert line
stopper XY              REM: Insert stopper to zero velocity
Line XY, –1000, –1000   REM: Insert line
stopper XY              REM: Insert stopper to zero velocity
Line XY, –1000, 1000    REM: Insert line
stopper XY              REM: Insert stopper to zero velocity
Line XY, 1000, 1000     REM: Insert line
ends X                  REM: End Segmented Motion
```

The program provides a rectangular path without velocity jumps:



Segmented Cyclic Motion

Suffix **c** provides cyclic execution of segmented motion. The **mseg/c** command creates a motion that after executing the last segment executes the first one and continues through the same sequence. In this case, the final point of the last segment should match the starting point of the first segment.

Cyclic segmented motion does not finish automatically. One of the commands **halt, kill, break** must be used in order to stop cyclic motion.

Similar to the slaved motion mode, the segmented motion mode implements the **XSACC, SYNV, JERK, ACC, and VEL parameters to control smooth motion.** In addition, velocity locking, position locking, stalled motion, and boundary condition mechanisms can be implemented in ACSPL+.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Virtual Axes

There is no limitation to the number of virtual axes that can be implemented in ACSPL+. Since all variables are updated at the MPU and trajectory update rate, any user variable can be a virtual axis. The user variables can be slaved to create the motion profiles desired. Virtual axes are typically used in applications that can control master motion in software versus adding additional encoder hardware to the applications.

The maximum number of axes available on the SPii Series products is eight axes per controller. All of the axes can be used as virtual axes inside the controller. If you are using a four axis SPiiPlus PCI, there are an additional four full motion axes that can be used as virtual axes. However, any user variable can also be used as a virtual axis.

Connect Function

As previously discussed in the control features section, the controllers are supplied with an on-board "connect function". The connect functions related the logical axes to the physical axes. The logical axis "x" controls the physical axis "X". The reference for the logical axis can be sourced by any variable or expression. Coordinate vector controlled motion includes linear, circular, sinusoidal, elliptical, helical, and inverse kinematics. The connect functions related the logical axes to the physical axes.

ACSPL+ Connect Function

Logical Axes	Connect Function	Physical Axes
X	*connect X = f(any variable)*	X
Y	*connect Y = f(any variable)*	Y
Z	*connect Z = f(any variable)*	Z
T	*connect T = f(any variable)*	T
A	*connect A = f(any variable)*	A
B	*connect B = f(any variable)*	B
C	*connect C = f(any variable)*	C
D	*connect D = f(any variable)*	D

The connect function allows the engineer complete flexibility in defining the characteristics of the motion profile. The engineer can implement a custom wave shaping algorithm, inverse kinematics, or any mathematical expression. For example, the x axis could be programmed to produce an output signal. By implementing ***connect X = sin x***, the resulting output waveform would be a sinusoidal output. The connect function is updated at the Motion Processor Unit cycle rate to ensure real time execution. In fact, all of the parameters and flags are updated every scan to guarantee deterministic performance.

The connect functions provides the ability to implement:

- Inverse kinematics
- Dynamic error compensation
- Gantry control where both X motor and Y motor, each with its own encoder, are acting as a single moving axis. The implementation is as simple as:

```
X_RPOS= connect X_APOS
Y_RPOS = connect X_APOS
```

Afterwards, the two motors will act as one, responding to each X axis move command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Connect & Mapping Function Examples

Encoder error compensation with constant step

Assume an axis was calibrated with a laser interferometer. The calibration process includes positioning to the points equally spaced according to the encoder feedback, and measuring the exact positions by the laser interferometer. The calibration points start from the coordinate 10000 and follow each 1000 counts. The last calibration point is 20000.

The calibration produced the following table:

Feedback	10000	11000	12000	13000	14000	15000	16000	17000	18000	19000	20000
Actual	10012	10998	11985	12981	13997	15007	16013	17023	18005	18993	19991
Error	+12	-2	-15	-19	-3	+7	+13	+23	+5	-7	-9

Only the third row is entered to a controller variable and is stored in flash file X_ERROR. Details of the calibration routine, which can be implemented on ACSPL+, are not discussed here.
An application that uses the file may provide an initialization routine like this:

```
real X_ERR(11)
AUTOEXEC:                                   Standard label of initialization routine
read X_ERR,X_ERROR                          Read calibration table from the flash
connect X_RPOS = X_APOS - map (X_APOS, X_ERR, 10000, 1000)
stop                                        Finish initialization
```

The connect function specifies calculating the reference position as subtraction of the interpolated error from the desired position.

Encoder error compensation with arbitrary step

Assume in the above example the calibration routine does not calculates an error, but writes to flash file X_CALIBR the first two lines from the above table. The table is stored as an array with 11 rows and 2 columns.

In this case the application can implement an initialization routine like this:

```
real X_CAL(11)(2)
AUTOEXEC:                                   Standard label of initialization routine
read X_CAL,X_CALIBR                         Read calibration table from the flash
connect X_RPOS = mapby2(X_APOS, X_CAL)
stop                                        Finish initialization
```

In this case file X_CALIBR can contain also a table with non-uniform points.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Two-dimensional encoder error compensation with constant step

Assume an XY table was calibrated with a laser interferometer. The calibration process includes positioning to the points equally spaced in the nodes of XY grid according to the encoder feedback, and measuring the exact positions by the laser interferometer. In each point two errors are calculated: by X and by Y. The results are stored in two matrixes:

X error matrix:

Y \ X	10000	11000	12000	13000	14000
5000	+6	0	-3	-4	+1
7000	+12	-2	-15	-19	-3
9000	+10	+1	-9	-9	-10

Y error matrix:

Y \ X	10000	11000	12000	13000	14000
5000	-12	-7	-1	+4	+9
7000	-3	+1	+5	+7	+11
9000	+1	+3	+6	+5	+7

The calibration program stores in the flash files X_ERROR and Y_ERROR. Details of the calibration routine, which can be implemented on ACSPL+, are not discussed here.
An application that uses the file may provide an initialization routine like this:

```
real X_ERR(3)(5),Y_ERROR(3)(5)
AUTOEXEC:                                   Standard label of initialization routine
read X_ERR,X_ERROR                          Read X calibration table from the flash
read Y_ERR,Y_ERROR                          Read Y calibration table from the flash
connect X_RPOS=X_APOS-map2(X_APOS,Y_APOS,X_ERR,10000,1000,5000,1000)
connect Y_RPOS=Y_APOS-map2(X_APOS,Y_APOS,Y_ERR,10000,1000,5000,1000)
stop                                        Finish initialization
```

The **connect** functions specify calculating the reference position as subtraction of the interpolated error from the desired position.

Backlash compensation

Assume, X axis has a backlash of 20 counts. The following **connect** command compensates the backlash:

```
connect X_RPOS = X_APOS + 10*dsign(X_RVEL, 0, 0)
```

By this **connect** the value added to desired position changes immediately when the direction of motion changes. In many cases such jumps in the desired position are harmful. In this case the third parameter in the **dsign** function can be used in order to introduce the backlash

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

compensation gradually. In the following example the backlash compensation is introduced by small steps, so that the compensation growth to the entire value by 20 milliseconds:

```
connect X_RPOS = X_APOS + 10*dsign(X_RVEL, 0, 20)
```

If the X axis executes master-slave motion slaved to some physical value like encoder feedback, the RVEL value contains a noise that can cause undesirable switching of the backlash sign. In this case the second parameter of the **dsign** function can be used in order to introduce antibouncing effect. In the following example the backlash compensation changes its sign only if the RVEL holds its new sign for more than 10 milliseconds:

```
connect X_RPOS = X_APOS + 10*dsign(X_RVEL, 10, 20)
```

Combined compensation of encoder error and backlash

An arbitrary expression can be used as argument in the **dsign** function and in the **map**- functions. It ensures combining different compensation function and other required transformations in one **connect** command. The following example combines error and backlash compensations from the above examples:

```
real X_CAL(11)(2)
AUTOEXEC:                                   Standard label of initialization routine
read X_CAL,X_CALIBR                         Read calibration table from the flash
connect X_RPOS = mapby2(X_APOS+10*dsign(X_RVEL, 10, 20), X_CAL)
stop                                        Finish initialization
```

Master Function Examples

Cam motion

Assume the Y axis must provide a cam motion following the X axis. File CAMTABLE in the flash contains 1000 rows and 2 columns. Each row contains an X coordinate in the first column and the corresponding Y coordinate in the second column. The X coordinates in the first column can be spaced either equally or non-equally.

The following fragment initializes the cam motion:

```
real CAMTABLE(1000)(2)
read CAMTABLE,CAMTABLE                      Read calibration table from the flash
master Y_MPOS = mapby1(X_FPOS, CAMTABLE)    Define master value via cam table
slave Y                                     Start master-slave motion
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Joystick

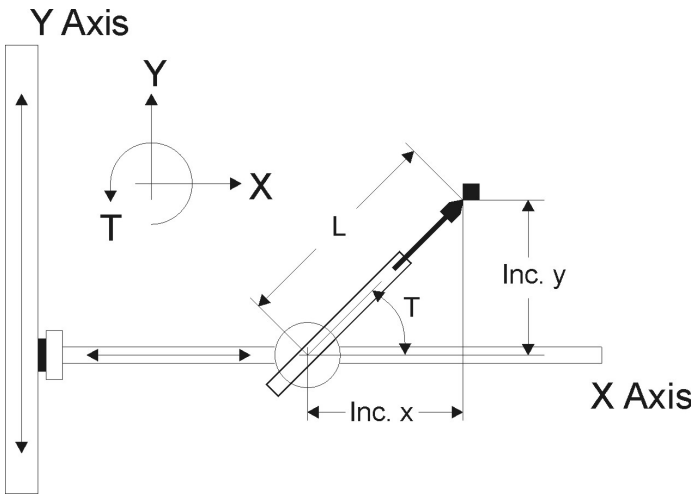
Assume the joystick is connected to analog inputs AIN0, AIN1. The joystick must control the motion of XY table. Velocity of each coordinate must be proportional to the corresponding analog input. Analog input lower than 20 counts must be ignored to avoid motion due to analog drift or bias. The X motion is limited in the range from –500 to 100000 counts. The X motion is limited in the range from –500 to 500000 counts.

The following fragment initializes the joystick motion:

```
real JK
JK = 10                                                    Joystick factor
master X_MPOS = intgr(AIN0*JK, 20, -500, 100000)           Define X master
master Y_MPOS = intgr(AIN1*JK, 20, -500, 500000)           Define Y master
slave Y; slave Y                                           Start master-slave motion
```

Inverse kinematics

Assume the controlled object has the following kinematics:



The tool is located on the carriage that moves in Cartesian coordinates XY. The tool can be rotated on the carriage (coordinate T). The user needs to specify motion of the tool's tip and the orientation of the tool.

The kinematics of the object is described by the following equations:

X = x + L*cos(T)

Y = y + L*sin(T)

Where X,Y are the tip coordinates,
x,y are the carriage coordinates,
L is a distance between the tip and the center of rotation,
T is a rotation angle.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following connect statements implement the inverse kinematics:

```
connect X_RPOS = X_APOS – TLEN*cos(T_APOS)

connect Y_RPOS = Y_APOS – TLEN*sin(T_APOS)

connect T_RPOS = T_APOS
```

The formulae assume that the user units of the T axis are radians. If other user units are used, an additional coefficient is required. The TLEN variables store the tool's length.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ACSPL+ Programming Examples

Multi-Programming Synchronization example

Program 1	**Program 2**
global int SYNC enable X ST: till SYNC ptp X, 2000 ptp X, 0 goto ST	global int SYNC enable Y ST: till SYNC ptp Y, 4000 ptp Y, 0 goto ST

- Program 1 controls X axis
- Program 2 controls Y axis
- As long as global variable SYNC equals one, the two programs run asynchronously
- Another program or operator can set the SYNC variable to zero and then to one to in order to start X and Y synchronously

Hardware Registration

The controller has high speed index mark registration latching. In typical packaging applications, it is desirable to move an axes until a registration mark is detected. At the moment of the registration, the existing move is cancelled and a new point-to-point move is initiated. This time-critical motion is easily implemented with ACSPL+.

Example of marker registration:

- Move to the mark position.
- Latch the exact mark position and continue moving.
- Stop the motion smoothly at the position calculated as mark position plus 15,000 units.

The following program fragment performs the task:

X_IST.#MARK = 0	Enable X axis index state for marking (registration)
jog X, +	Start jogging in positive direction
till X_IST.#MARK	Wait for MARK (registration) signal
ptp X, X_MARK + 15000	Seamless move to the point **X_MARK+15000**

When the MARK1 becomes active, the instantaneous position is automatically latched into the **MARK** variable. At the same moment, the corresponding bit **IST.#MARK** (index state) is set indicating that variable **MARK** contains a valid value. Additional occurrences of the MARK1 input are not latched as long as the corresponding bit **IST.#MARK** remains set. The **MARK** variable retains the position of the first occurrence. In order to enable MARK1 latching, the application needs to clear the **IST.#MARK** bit. The latched position accuracy is ±1 quadrature encoder count at any velocity up to 5,000,000 counts/sec. Assignment **X_IST.#MARK = 0** resets the status bit of MARK1.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Other Example Programs

Homing an Axis

```
#/ Controller version  = 1.81-03u(40)
#/ Date = 03/07/2001 16:05
#/ User remarks =
#0
! Program 6_1   Homing example
!   The program executes the following sequence:
!     - Move to the left limit switch.
!     - Move to the encoder index.
!     - Set the axis origin to the position of index.
!     - Move to the origin.

enable X

jog/v X,-5000            ! Move to the left limit switch
till X_FAULT.#LL         ! Wait for the left limit switch activation
jog/v X,5000             ! Move to the encoder index
till ^X_FAULT.#LL        ! Wait for the left limit release
X_IST.#IND=0             ! Reset the index flag - activate index
circuit
till X_IST.#IND          ! Wait for crossing the index
set X_FPOS=X_FPOS-X_IND  ! Set axis origin to the position of index
ptp X,0                  ! Move to the origin
stop                     ! Stop the program
```

Point to Point Motion on an Ellipse

```
#/ Controller version  = 1.81-03u(40)
#/ Date = 03/07/2001 17:17
#/ User remarks =
#0
! Program 8_2  Points on Ellipse, PTP motion
!   The program executes positioning to 30 points spread equally
!      along the ellipse with half-axes 4000 and 2000.
!   The program repeats positioning to the points cyclically.

int J              ! Local variable J - index
J = 0              ! Start from zero

enable XY
mptp/c XY          ! Create cyclic MPTP motion in XY axes
loop 30            ! Execute the loop 30 times
                   ! Specify points for MPTP motion
   point XY,4000*cos(2*3.14159/30*J),2000*sin(2*3.14159/30*J)
   J = J + 1       ! Increment index
end                ! End of loop
ends XY            ! End MPTP motion

stop               ! Stop the program
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Fault Response

```
#/ Controller version  = 1.81-03u(40)
#/ Date = 03/07/2001 18:25
#/ User remarks =
#0
! Program 11_3  Globalizing response to the fault.
!   The program kills all executed motions and terminates all
!     executed programs once any limit switch fault is active.

on S_FAULT.#LL | S_FAULT.#RL   ! Once any Limit fault is active…
   killall; stopall            ! Kill all motions, terminate all
program
ret
```

Inspection

```
#/ Controller version  = 1.81-03v02
#/ Date = 04/01/2001 14:15
#/ User remarks =
#0
! Program 12_1  Tool Inspection.
!   The program validates the lathe tool by touching the contact sensor
!     with the tool edge.

real ALARMTIME              ! Local variable for alarm indication
ALARMTIME=0; OUT0.2=0       ! Reset alarm indication

enable XY

ptp XY,0,0                  ! Move to initial point.
ptp XY,10000,30000          ! Move to the point where the contact
search begins.
X_IST.#MARK=0               ! Reset MARK flag, enable MARK operation
till ^X_MST.#MOVE           ! Wait for the motion termination.
ptp X,11000                 ! Move over the contact point.
  ! The  next line is here only to simulate the actual Mark signal.
  ! Must be omitted in actual application.
OUT0.3=1; till X_RPOS>10900; OUT0.3=0

till ^X_MST.#MOVE           ! Wait for the motion termination.
if ^X_IST.#MARK | (abs(X_MARK-10900)>100) ! Check fitness condition.
    ALARMTIME=TIME+200      ! Activate alarm indication
end
stop

on (ALARMTIME <> 0) & (TIME > ALARMTIME)  ! Alarm indication
    OUT0.2=^OUT0.2; ALARMTIME=TIME+200    ! Blink the output
ret
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Pick and Place

```
#/ Controller version  = 1.81-03u(40)
#/ Date = 03/07/2001 18:59
#/ User remarks =
#0
! Program 14_1 Pick-and-Place robot.
!   The robot picks a part in position -1000 and places in position
9000.
!   The robot uses suction to kepp the part.
!   The application processes the Mark signal from a photo sensor
!     to compensate inaccuracy of the part location on the suction.

real DIS            ! Local variable DIS. Used in displacement
calculation.

enable XZ

PnPcycle:           ! Label. Start of Pick-and-Place cycle.
ptp X,-1000         ! Move to pick position.
till ^X_MST.#MOVE   ! To start Z motion only after the end of X motion.
ptp Z,-2000         ! Expand manipulator down to the storage.
OUT0.2=1            ! Activate suction.
wait 50             ! Wait for suction operation.
ptp Z,2000          ! Move manipulator up.
till ^Z_MST.#MOVE   ! To start X motion only after the end of Z motion.
ptp X,9000          ! Move to preliminary place position.
X_IST.#MARK=0       ! Reset MARK flag - enable MARK operation.

  ! The next line is here only to emulate the actual Mark signal.
  ! Must be omitted in actual application.
till X_RPOS>1000; OUT0.4=1

till X_IST.#MARK    ! Wait for photo sensor pulse.
DIS=X_MARK-1000     ! Calculate displacement.
  ! 1000 is an expected point of sensor activation assuming ideal part
location.

break X             ! The current motion will be terminated immediately
when the next appears.
ptp X,9000+DIS      ! Move to the corrected place position.
till ^X_MST.#MOVE   ! To start Z motion only after the end of X motion.
ptp Z,200           ! Expand manipulator to the place point
OUT0.2=0            ! Deactivate suction.
wait 50             ! Wait for suction operation.
ptp Z,2000          ! Move manipulator up.
till ^Z_MST.#MOVE   ! To start X motion only after the end of Z motion.
goto PnPcycle       ! Repeat
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Master Slave Motion

```
#/ Controller version  = 1.81-03v03
#/ Date = 04/02/2001 13:27
#/ User remarks =
#0
! Program 15_2    Paper-cutting machine
!   The knife edge circumference is not equal to required sheet length.
!   The application implements master-slave motion with
!      poisition lock and each cycle corrects the master position.

global real Factor, Offset      ! Variable definitions.
  ! The variables must be global because the MASTER command can
  ! refer to global variables only.
real Length

Factor=4000/(2*3.14159*94.5*12)  ! Gear factor ratio.
Offset=0

enable X
set X_FPOS=0, T_FPOS=0

CutCycle:            ! Label. Start of cutting cycle.
Offset=Offset+4000-Factor*Length ! Correct the master offset
master X_MPOS=Factor*T_FPOS + Offset  ! Define master.
till X_MPOS>=X_FPOS  ! Wait to prevent motion in negative direction
slave/p X            ! Start master-slave motion, position lock.
X_IST.#IND=0; till ^X_IST.#IND  ! Move up to encoder index
halt X               ! Terminate motion
goto CutCycle        ! Repeat cutting cycle
```

Thread Cutting

```
#/ Controller version  = 1.81-03v03
#/ Date = 03/30/2001 17:49
#/ User remarks =
#0
! Program 16_1    Thread-cutting with variable pitch.
!   The pitch starts from 1mm and rises 0.5mm per each rotation.

enable XY

ptp XY,10000,40000  ! Move to the beginning of the thread.
set Z_FPOS=0        ! Set Z origin to the current location.

master Y_MPOS=(1000+Z_FPOS*500/4096)*Z_FPOS/4096 ! Define master
formula.
slave Y             ! Start master-slave motion.

OUT0.4=1            ! Start the spindle rotation.
ptp/rv X,-2000,1000 ! Run-in.
 ! Relative PTP by .2000 counts with velocity 1000 counts/sec

till Z_FPOS>4096*10 ! 10 threads.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
halt Y              ! Terminate master-slave motion.

OUT0.4=0            ! Stop the spindle.
stop                ! Stop the program
```

PLC Control

```
#/ Controller version  = 1.81-03u(40)
#/ Date = 03/07/2001 20:04
#/ User remarks =
#0
! Program 18_1    PLC example

real T1          ! Local variable. Used to calculate time.

PLCcycle:        ! Beginning of PLC cycle
OUT0.0 = ^IN0.5  ! OUT0.0 follows inverted IN0.5.
OUT0.1 = IN0.6 ~ IN0.7  ! OUT0.1 follows exclusive OR of IN0.6 and
IN0.7.
OUT0.2 = (IN0.6 ~ IN0.7) & IN0.8
   ! OUT0.2 is the same as OUT0.1 while enabled by IN0.8.
if (T1 <= TIME); OUT0.3 = ^OUT0.3; T1 = TIME + 500; end
   ! OUT0.3 blinks each 500msec.
goto PLCcycle     ! Execute PLC cycle forever

! Autoroutines provide one-controller-cycle time resolution (fast PLC)
!  OUT0.4 follows raising edge of IN0.0 with 2sec delay,
!        falling edge - immediately.
on lag(IN0.0, 2000) ; OUT0.4 = 1; ret
on ^IN0.0; OUT0.4 = 0; ret
```

Teaching Motion

```
#/ Controller version  = 1.81-03v03
#/ Date = 03/30/2001 17:54
#/ User remarks =
#0
! Program 19_1    Teach-and-Go example
!     The operator must do the following:
!         1. Activate switch IN0 when he is ready to start TEACH phase
!         2. Move the X motor by hand while OUT0 is blinking (Teach
phase)
!         3. Deactivate switch IN0 when OUT0 is off and he is ready to
start GO phase
!         4. Watch the reproduced motion

global real Trace(10000) ! Array for TEACH data.
      ! Array for data collection must be declared global.
disable X     ! Release X motor to be moved by hand.
till IN0.0    ! Wait for user to activate IN0.
dc Trace,10000,1,X_FPOS   !Do TEACH data collection.
   ! Collect X_FPOS to Trace array, 10000 samples with 1msec period.
while S_ST.#DC !Indicate TEACH phase.
   ! S_ST.#DC=1 while the data collection is in progress.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
   OUT0.0=^OUT0.0  ! Blink LED.
   wait 200        ! Blink period is 200msec.
end
OUT0.0=0     ! Indicate end of TEACH phase.
till ^IN0.0  ! Wait for user to deactivate IN0.
OUT0.0=1     ! Indicate GO phase.

enable X
ptp X,Trace(0) ! Positioning to the first point
path X,1     ! Arbitrary path motion, points follow with 1 ms interval.
mpoint X,Trace,10000  ! Add 10000 points from the Trace array.
ends X       ! No more points will be added.
till ^X_AST.#MOVE     ! Wait for motion termination.
OUT0.0=0     ! Indicate end of GO phase
stop
```

Incremental PEG

```
#/ Controller version  = 1.81-03u(40)
#/ Date = 03/07/2001 20:12
#/ User remarks =
#0
! Program 20_11  Incremental PEG example.
! Inspection machine.
!   The machine provides scanning move along Y axis.
!   Moving in positive direction, the controller must activate the
!   camera in the positions from 10000 to 20000 with interval 1000
counts.

setconf(205, 1, 0b100000000)  ! Assign OUT7 to Y PEG output
enable Y
Cycle:      ! Label.
ptp Y,0     ! Move to the beginning of scan.
peg_i/s Y,0.01,10000,1000,20000
   ! Activate incremental PEG:
   !    /s - synchronously with motion, to operate with the next
motion,
   !    0.01 – PEG pulse width of 0.01msec=10microsecond,
   !    10000 – first point,
   !    1000 – increment,
   !    20000 – last point.
ptp Y,30000 ! Scan motion
goto Cycle  ! Repeat
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SPiiPlus Series: DEVELOPMENT TOOLS

Development Tools

SPiiPlus Series products are provided with comprehensive support tools. Since the controllers are based on the X86 processing platform, the development tools can be created in parallel with the firmware development. This allows the highest quality and most cost-effective solutions to be implemented. The tools suite provided with the controllers contains:

SPiiPlus MMI for Windows: User interface to the controller. The Motion Machine Interface (MMI) is a interactive support tool for setting up, tuning, and viewing parameters. Features a four-channel soft oscilloscope for feedback tuning. Also includes the SPiiPlus Simulator, which allows the user to demo the other tools using only a single PC.
SPiiPlus MultiDebugger for Windows (Optional): Development environment for ACSPL+ (ACS MultiProgramming Language) applications. Includes the SPiiPlus Simulator.
SPiiPlus C LIB Libraries (Optional): Comprehensive Dynamic Link Library (DLL) that supports host application programming in C/C++, and Visual Basic with full multithreading support.

SPiiPlus Simulator

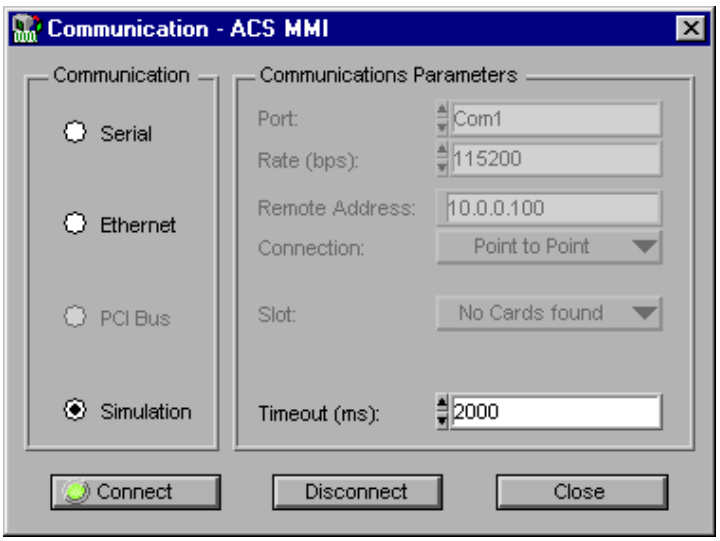
A unique feature of the software is the SPiiPlus Simulator. The Simulator is a full-function software module that exactly simulates the movements of the controller's axes. The SPiiPlus Simulator is run on the same PC as the MultiDebugger and the MMI and communicates directly with them. There is no controller necessary. This allows development and design of an application before a controller or any hardware is introduced.

The Simulator :

- Allows use of the MMI and MultiDebugger without being connected to an actual controller.
- Allows users to test applications they have written as if a controller was attached.
- Powerful demonstration tool for selling your product.

The simulator is easily implemented by starting the MMI software. The simulator assumes zero closed loop following error. Intelligent ACSPL+ programs can be developed on the desktop without time-consuming hardware purchases.





Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The Motion Machine Interface (MMI) for Windows is a powerful tool that simplifies the setup and tuning process of the motion control system.

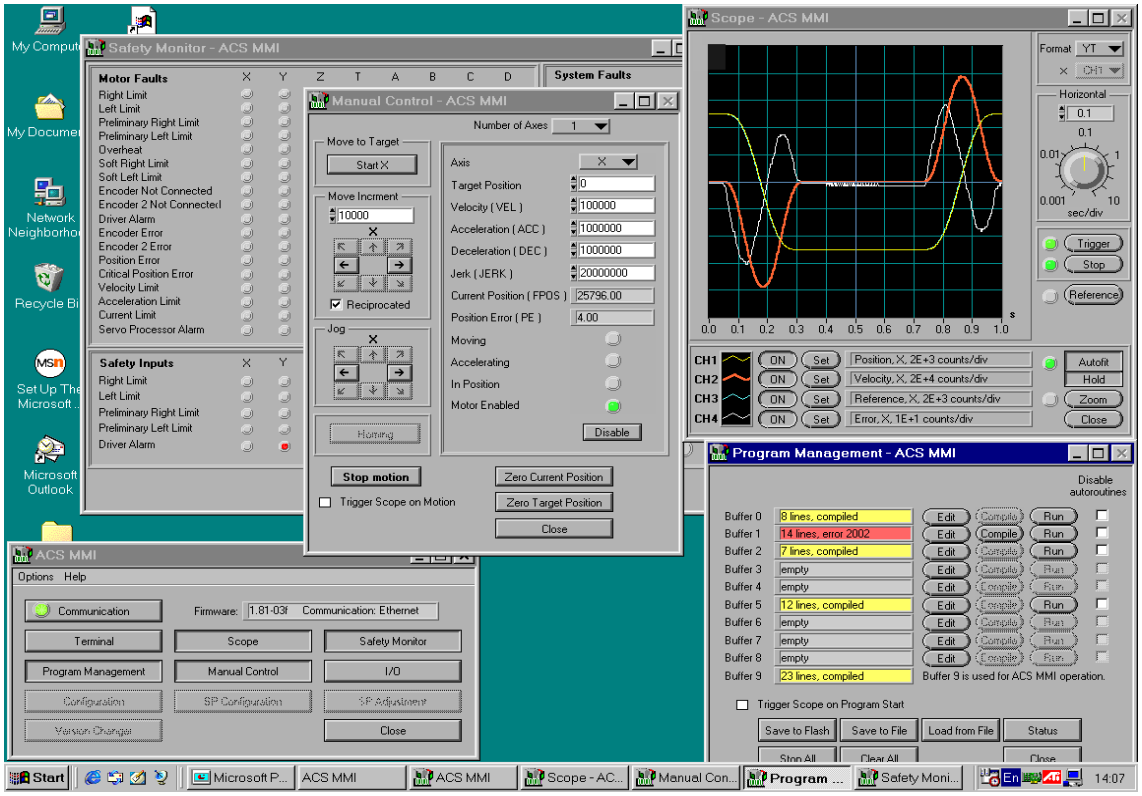
Motion Machine Interface (MMI) Features

Features	Benefits
Powerful Scope	Scope Signals: • Up to four signals simultaneously • Predefined signals i.e., following error, feedback, velocity, etc. • Any standard or user variables • Any internal variable of Servo Processor • Any address in the controller memory • One bit of any integer variable • FFT signal analysis Scope Triggering: • Normal, Auto, and Single triggering modes • Triggering by rising or falling edge • Adjustable trigger level
Communication Terminal	• User communication to the controller Background mode: • The Communication Terminal displays messages issued by other components if ACS MMI Direct Mode: • The user communicates the controller solely, all other components are disabled • Commands can be sent to the controller and cause scope to trigger
Program Management	• Watch state of all program buffers • Load programs to the controller • Save programs from the controller • Compile, Edit, Run, Stop program • Disable/Enable autoroutines
Motion Management	• Scaleable from 1 to 8 axis • Axis state display • Enable/Disable axis • Zero current position • Absolute, Incremental, and Jog moves • Monitor axes state • Homing, etc.
Safety Signals Monitor	Current state of all Safety Signals: • Limit Switches • Temperature Sensor • Driver Alarm • Emergency Stop • Current state of Motor and System Faults
I/O Monitor	• Indication for current state of General Purpose Inputs/Outputs • Ability to change state of General Purpose Outputs
Tuning	• Password protected • Window to set maximum tolerable values • Set of specific adjusters and powerful tuning tools • Tuning in parallel with scope • Configurator for setting motor and encoder values • Adjuster for tuning motor feedback circuit

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MMI Features (continued)

Features	Benefits
Flash Manager (Firmware Upgrade)	• Password Protected • No hassle with EPROM changes • Firmware is upgraded serially or through Ethernet • No external hardware required
Application Saver/Loader	• Password Protected • The ACS Saver Module reads data stored in the controller's flash memory and saves it to a file. • The ACS Loader module loads data stored in a file to a controller. It is possible to load a file to a different controller from which it was saved.



Overall view of MMI Interface showing multiple task windows open

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

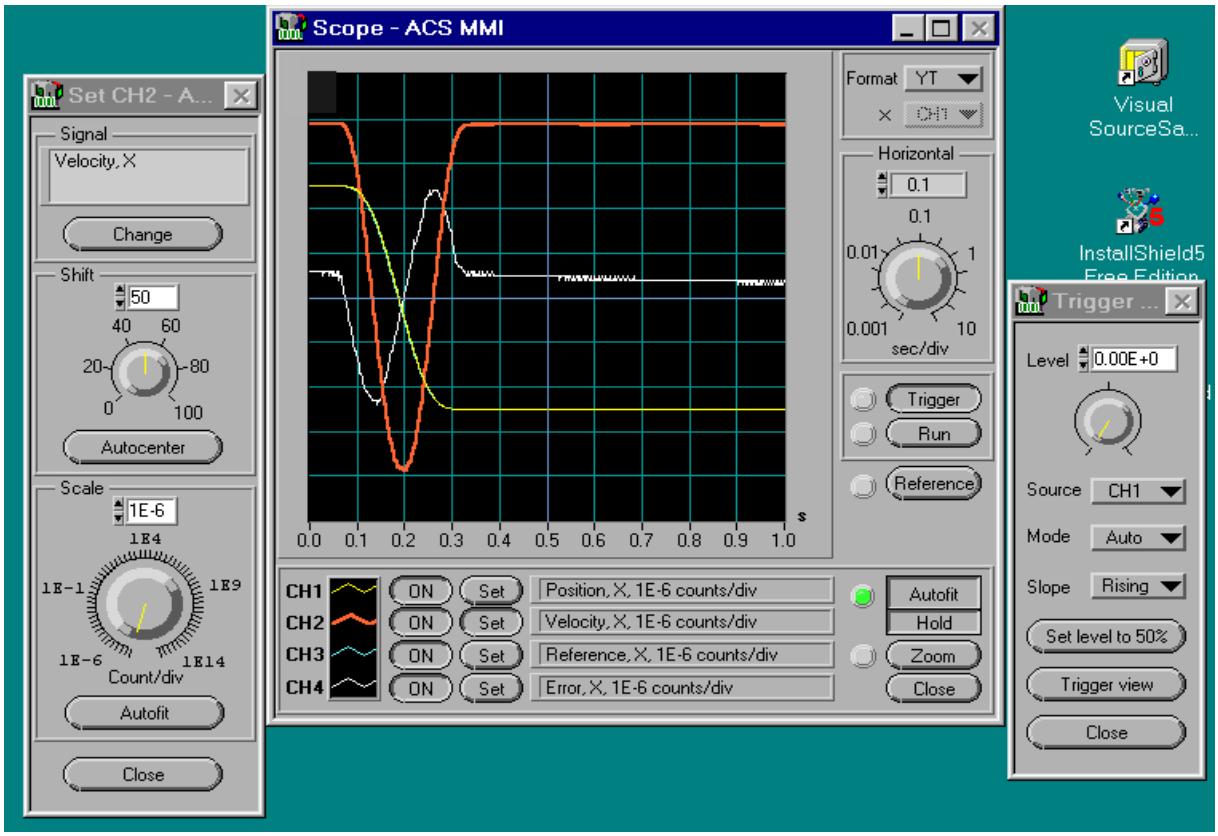
MMI Features (continued)

Scope Signals

- Up to four signals simultaneously
- Predefined signals i.e., following error, feedback, velocity etc.
- Any standard or user variables
- Any internal variable of Servo Processor
- Any address in the controller memory
- One bit of any integer variable
- FFT signal analysis

Scope Triggering

- Normal, Auto, and Single triggering modes
- Triggering by rising or falling edge
- Adjustable trigger level



Communications window

- User communication to the controller

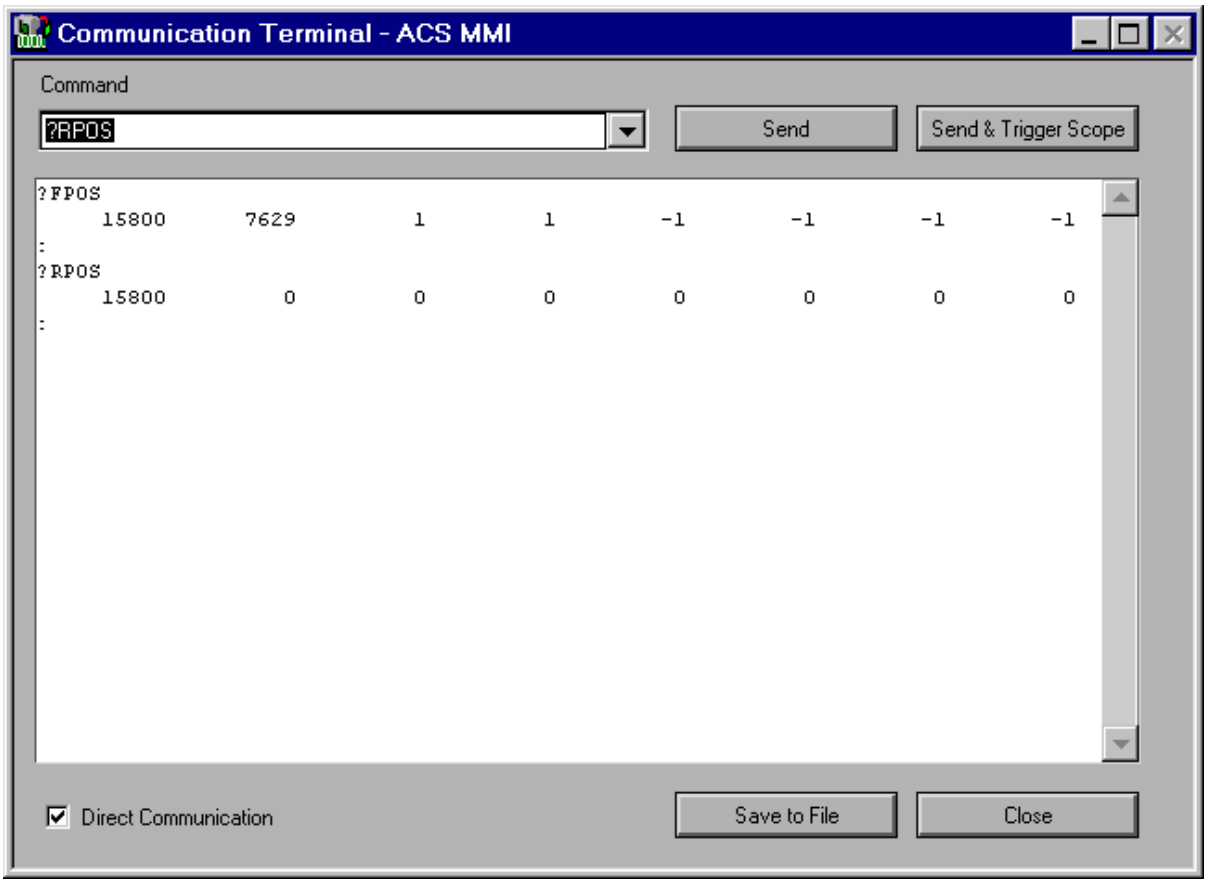
Background mode

- The Communication Terminal displays messages issued by other components if ACS MMI is not in direct communication mode.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Direct Mode

- The user communicates the controller solely; all other components are disabled
- Commands can be sent to the controller and cause scope to trigger

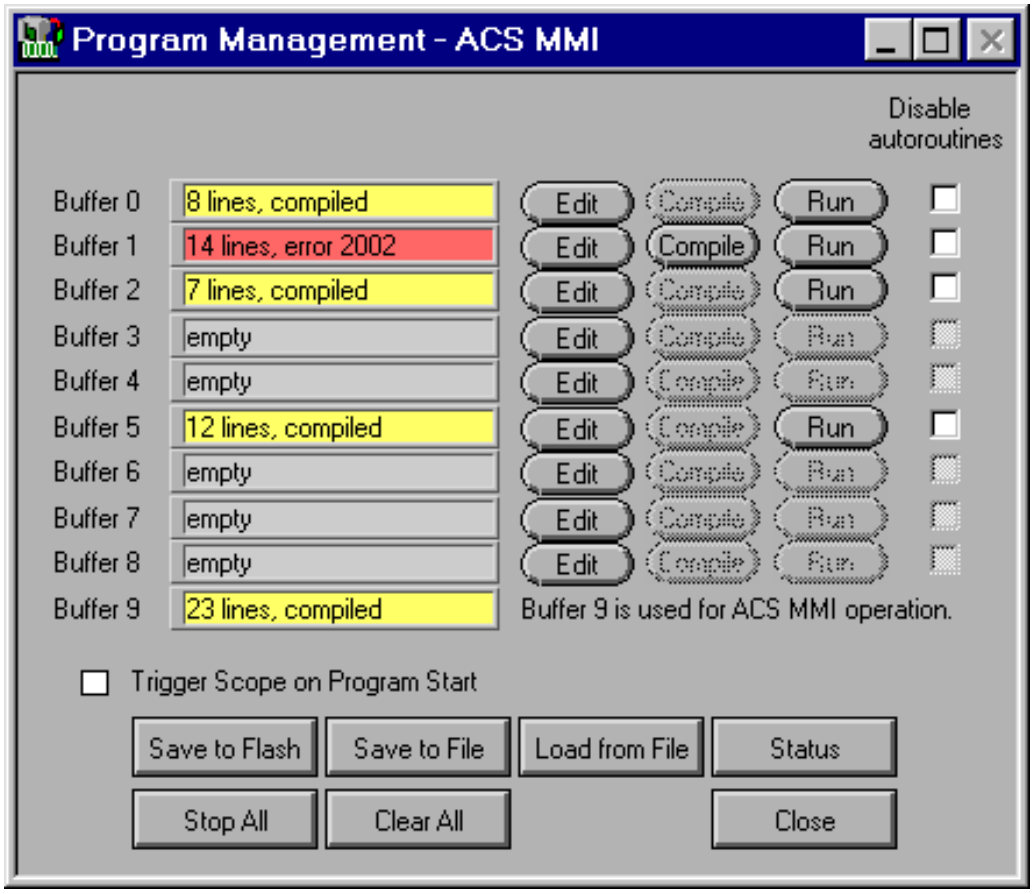


Example of Feedback Position and Reference Position

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MMI Program Management

- Watch state of all program buffers
- Load programs to the controller
- Save programs from the controller
- Compile, Edit, Run, Stop program
- Disable/Enable autoroutines

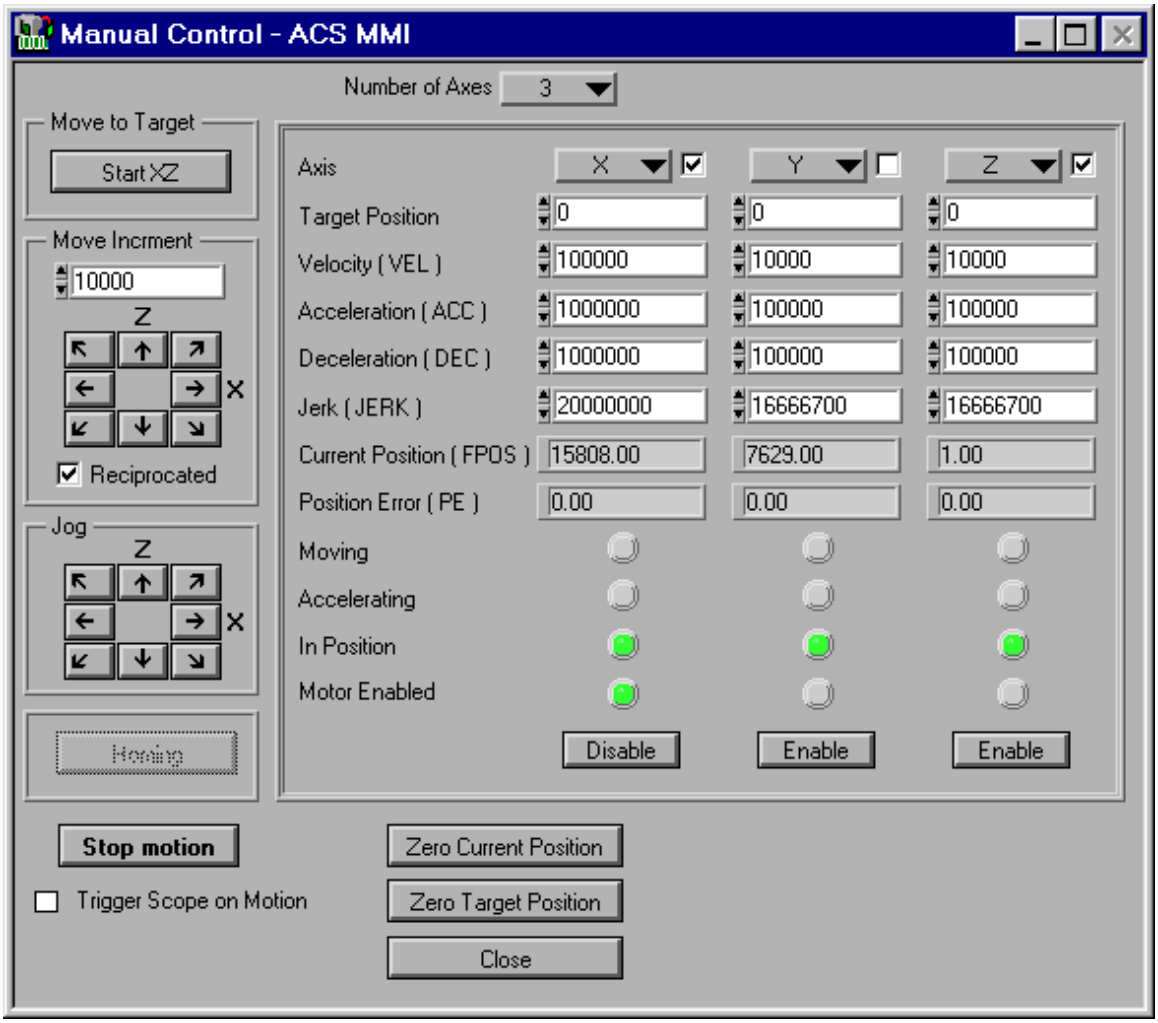


Program Management View

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MMI Manual Control

- Scaleable from 1 to 8 axis
- Axis state display
- Enable / Disable axis
- Zero current position
- Absolute, Incremental and Jog moves
- Monitor axes state
- Homing, etc.

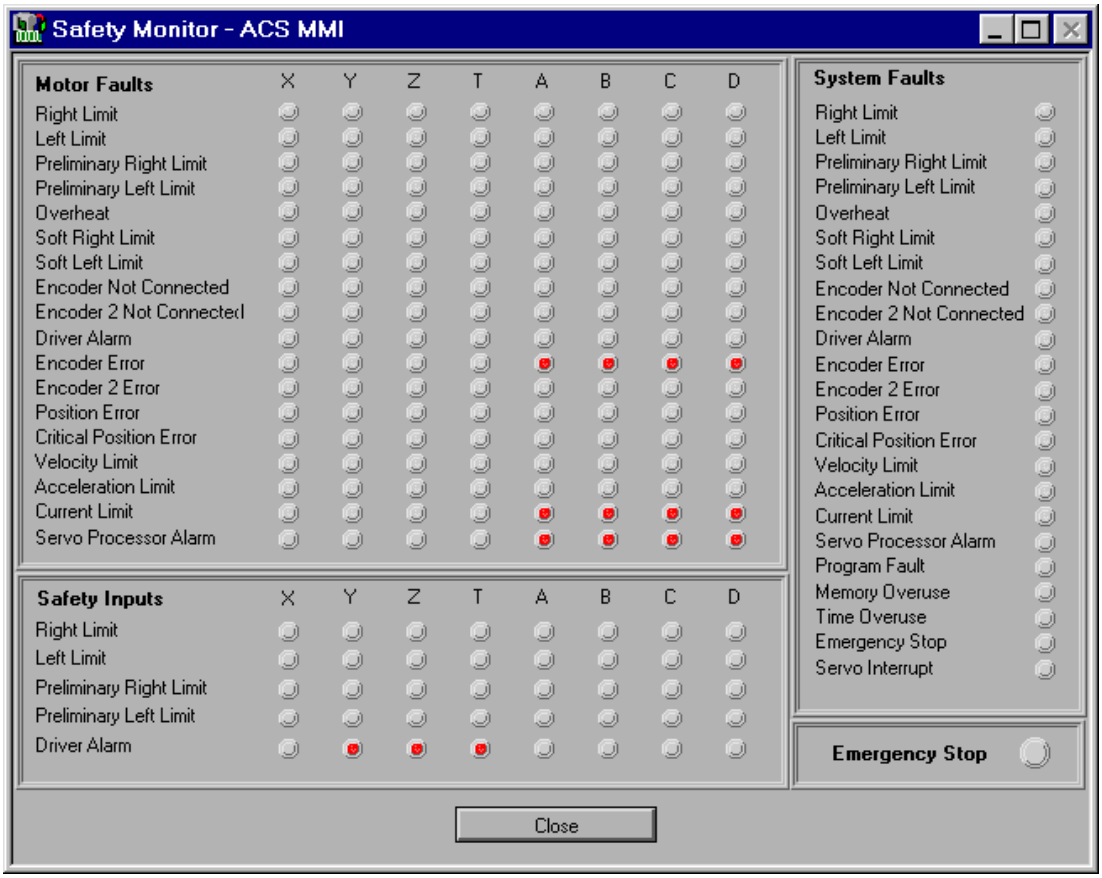


Motion Management View

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MMI Safety Signals Monitor

- Limit Switches
- Temperature Sensor
- Driver Alarm
- Emergency Stop
- Current state of Motor and System Faults



Safety Monitor View

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

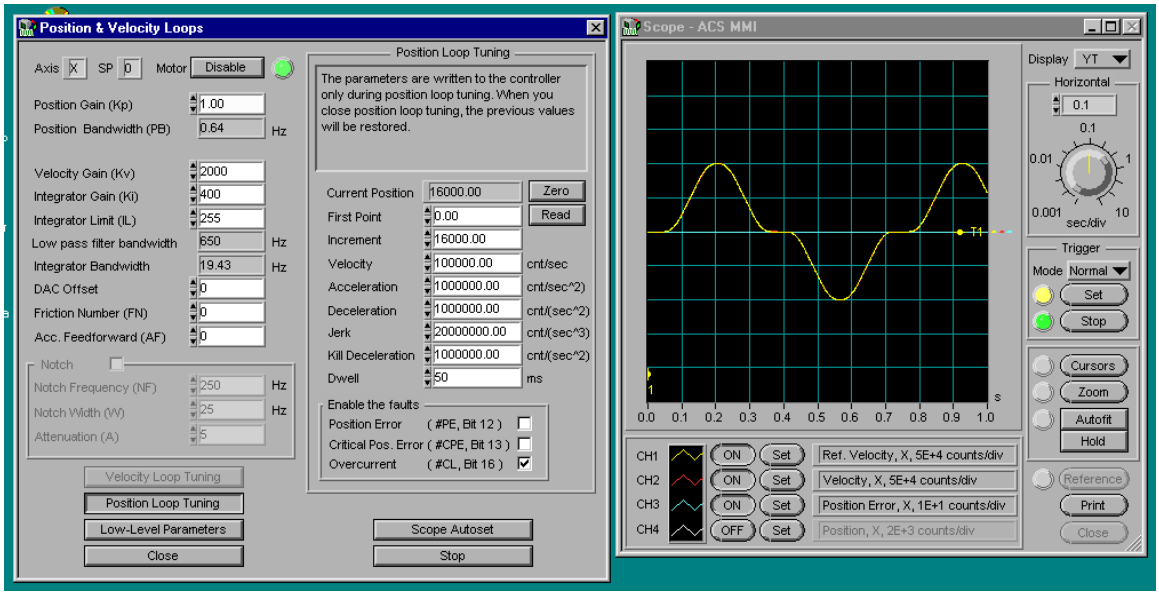
Tuning

The SPiiPlus MMI contains the Adjuster, a powerful tuning tool for tuning position and velocity loops. The Adjuster records all parameter values in an application database, which contains all the values entered or changed during an adjustment session. An adjustment session can be saved as a project, which can be used later to quickly set up other controllers for the same application. (There is also a function for open loop tuning.)

A typical adjustment session involves the following 10 steps:

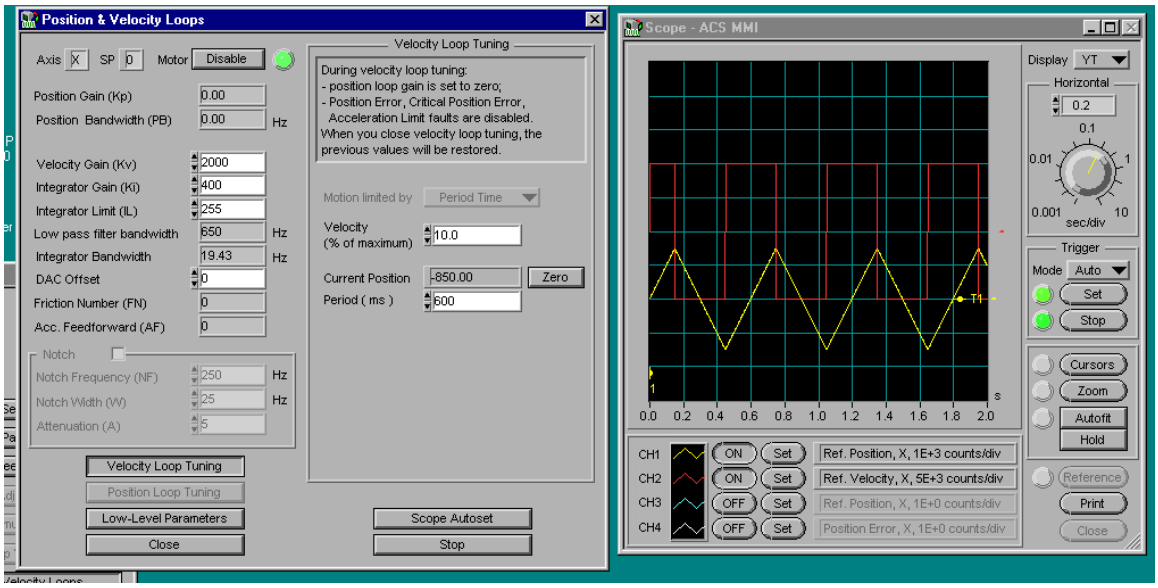
1. Amplifier parameters
2. Motor/feedback parameters
3. Protection parameters
4. Feedback verification
5. Current loop adjustment
6. Commutation adjustment
7. Velocity loop adjustment
8. Position loop adjustment
9. Polishing (feed-forward acceleration, slip constant, field current, etc.)
10. Review Parameters

The Adjuster links to the SPiiPlus Scope, a powerful software oscilloscope that enables you to monitor and print all real time variables, such as position error, velocity, phase current, etc.



SPiiPlus MMI Position Loop Tuning

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

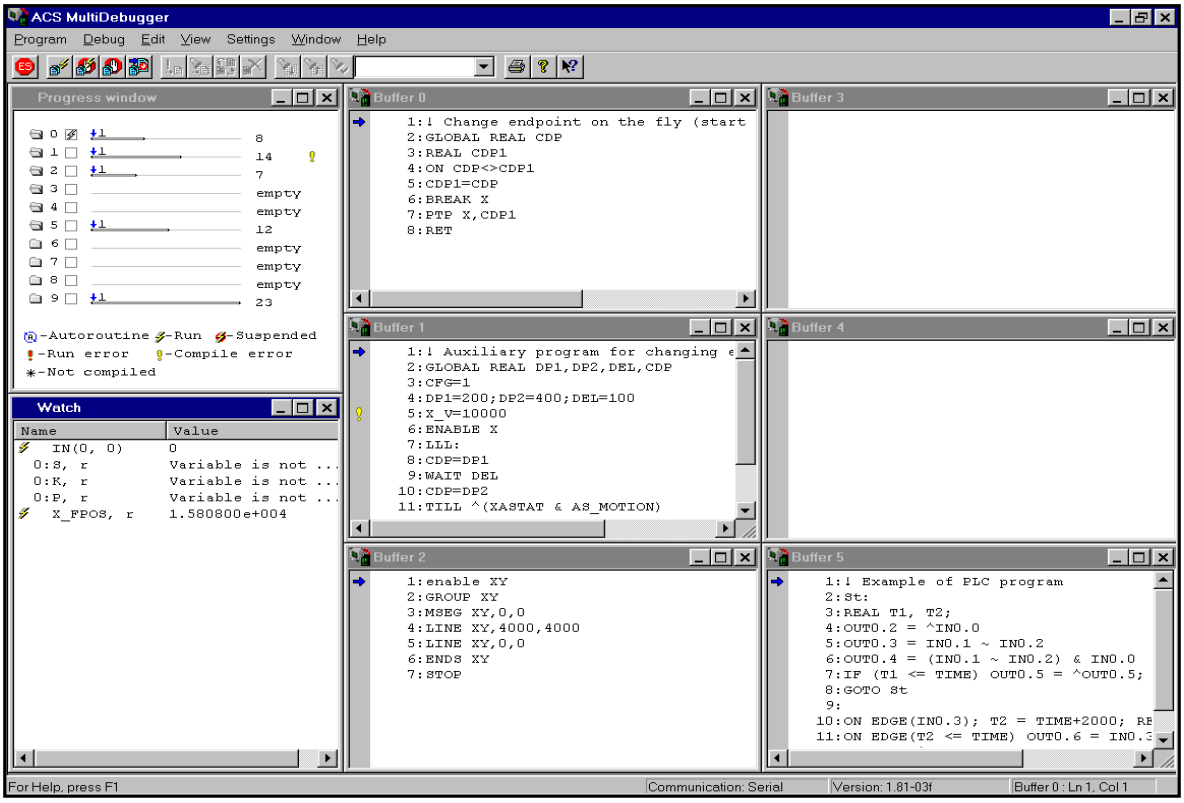


SPiiPlus MMI Velocity Loop Tuning

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MultiDebugger for Windows (optional) is a powerful aid to programming in ASCPL+.

- Interactive tool for ACSPL+ MultiProgramming
- 10 Parallel programs (buffers)
- Progress window for imaging concurrent execution
- Normal and step-by-step execution
- Breakpoints
- Watches for standard user variables
- Multiple program windows, watches, etc. can be viewed simultaneously



SPiiPlus MultiDebugger

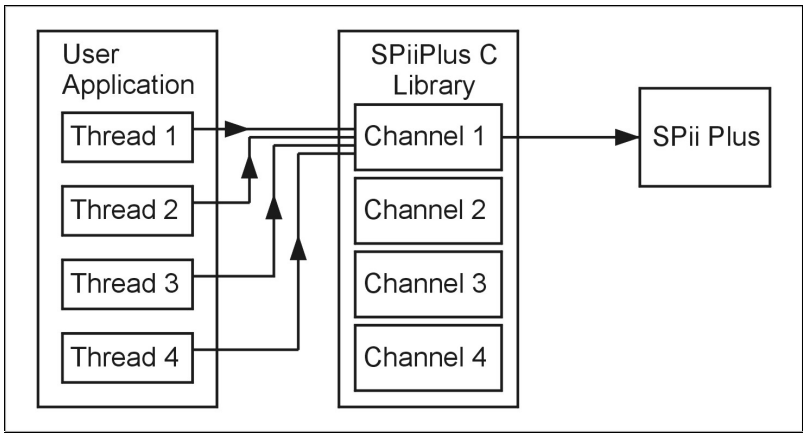
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SPiiPlus C LIB Library is a Dynamic Link Library (DLL) that supports host application programming in C/C++, and Visual Basic. The DLL works for Windows 98/NT/2000 operating systems.

- Hides the complexity of low-level communication from the user
- Makes totally transparent difference between communication channels
- Includes Simulator, so the user application can be operated without a real controller

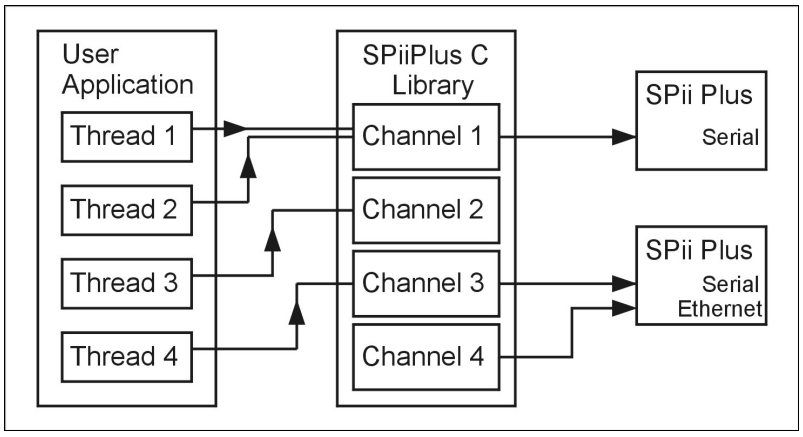
SPiiPlus C LIB Multithreading

- The library supports single- and multi-threaded applications.
- After a communications channel is open, any thread in the application can call the library function.
- Synchronization is transparent for the application (i.e., the library itself resolves collisions).



SPiiPlus C LIB Multiple Channels

- The Library supports up to 10 channels of communication simultaneously.
- Typical connection: one channel – one SPiiPlus controller.
- Also two or more channels can be connected to one SPiiPlus controller. For example, the first channel to a serial port and the second to Ethernet.
- Any thread in the application can work with any open channel.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SPiiPlus Series: CONTROLLER SPECIFICATIONS

Position Control
Sampling Rate: 20kHz
Control Algorithms: P Type, automatic velocity feed-forward, acceleration feed-forward, integrator anti-reset windup
Trajectory Calculation Rate: 1kHz
Range: +/-4x10^{15} counts.
Accuracy: +/-1 encoder count
Registration Mark (Position capture):
Inputs: up to one per main axis (X, Y, Z, T), through the general purpose digital inputs
Delay: <0.1µs

Velocity Control
Sampling Rate: 20kHz.
Control Algorithm: PI Type, second order low pass and notch filters
Range: Up to 160x10^{9} count/second
Resolution: 1 count/second
Velocity Accuracy:
Long term: 0.005%
Short term: 0.01% - 0.5% (system-dependent)
Acceleration Range:
Up to 4x10^{15} count/second^2
Position and Velocity Feedback
Types: Incremental digital encoder, optional SIN-COS encoders
Incremental Encoder Type: Incremental, 3 channel, differential, RS-422.
A&B, I; UP-DN, I; CLK-DIR, I. Maximum frequency of 60MHz theoretical, 20MHz practical
SIN-COS Encoder Interface (optional): Up to eight (one per axis)
Programmable multiplication factor: x4 – x8192
SIN-COS Encoder Type: Incremental, 3 channel, differential, 1Vptp. Maximum frequency of 500kHz.

Drive Interface
Servo/DC Motor Commands: Two commands per axis (for commutation), differential, +/-10V, 16-bit DAC resolution. All unused drive commands can be used as general purpose, uncommitted analog outputs.
Offset compensation: software compensated with 0.3mV resolution.
Stepper Motor Commands: Four pairs of clock-dir, differential, RS422, 6 MHz maximum frequency
Drive Enable Output: One per axis.
Type: Open collector/open emitter.
Collector Emitter Voltage Range: 5Vdc to 30Vdc
Output Current: 50mA
Propagation Delay: <1ms
Drive Fault Input: 1 per axis
Type: Source/sink
Input Voltage Range (External Supply): 5Vdc (+/-10%) or 24Vdc (+/-20%), automatic detection
Propagation delay: <1ms

I/O
Safety Inputs: One dedicated E-stop. Left limit and right limit per axis.
Type: source/sink, opto-isolated
Input Safety Voltage Range (External Supply): 5Vdc (+/-10%) or 24Vdc (+/-20%), automatic detection
Propagation Delay: <1ms

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

General Purpose and Registration Mark (Position Capture)
Inputs: Eight. Can be used as general-purpose inputs or as Registration Mark inputs (two per X, Y, Z, and T axis)
Type: differential, RS-422
Propagation Delay: <0.1µs
General Purpose and PEG™ Outputs: 8, can be used as general-purpose outputs or as PEG (position compare) outputs. 4 additional outputs are dedicated for PEG only.
Type: differential, RS-422
Propagation Delay: <0.1µs
PEG position compare accuracy: ±1 count up to 5,000,000 counts/second
PEG generated pulse width range: 25nsec to 1.6msec
Edge separation between two PEG events: minimum 200nsec
Number of PEG pulses in Random (table based) mode: up to10,000
Number of PEG events in Incremental mode: Unlimited
Analog Inputs: Eight / sixteen, differential, +/- 1.25V, 14-bit resolution. Each SIN-COS encoder consumes two analog inputs.
Analog Outputs: Eight / sixteen. Shared with the drive commands. Differential, +/-10V, 16-bit resolution

Communication
PCI-Bus: 33MHz for host interface, 32 bits
Bi-directional FIFO: 2 x 512x8
Uni-directional FIFO (Controller to host): 1,024x8
Dual Port Ram (DPR): 256x16
Comprehensive interrupt support
RS-232: 2 ports. 115,000 Baud.
Ethernet port (optional): 10BaseT; 10MHz
Simultaneous communications through all channels is supported.

Controller
Multi-Processor Architecture:
Motion Processor Unit (MPU): 133 MHz PC104+ single board computer
Real-time controllers: 120MHz SPii Servo Processor, one per two axes
Memory:
RAM: 8M
Non-Volatile Memory: 4M
User Memory: 1M
Power Consumption
5VDC @ 3.5A, +/-12VDC @ 0.25A
When used outside the PC, the 5V and +/-12V must be supplied through a dedicated power connector

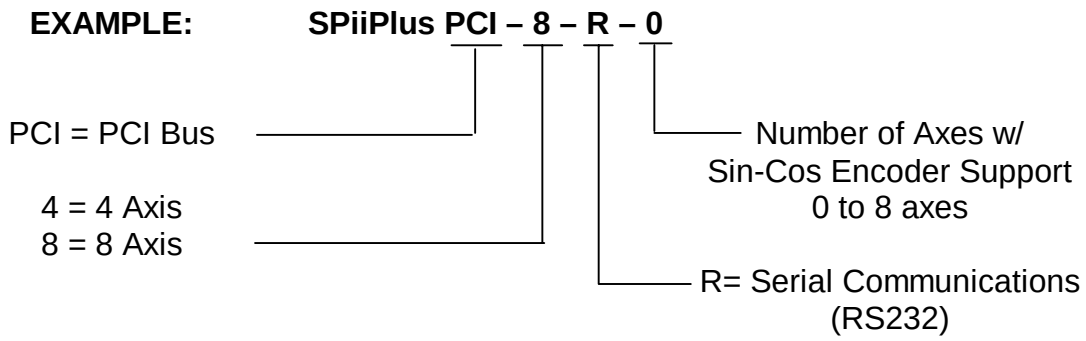
Environmental
Operating Temperature: 0 to 70 degrees C
Storage Temperature: -40 to 85 degrees C
Humidity: 90% RH, non-condensing

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SPiiPlus Series: ORDERING MATRIX

SPiiPlus PCI Ordering Matrix

- 1-4 Axis Ethernet or Serial Communications Motion Controller
- 1-8 Axis Ethernet or Serial Communications Motion Controller



Standard SPiiPlus PCI 4 / 8 Includes:

- Pre-Emptive Multi-Tasker (Multi-Programming)
- Motion Processor Unit (MPU) is Pentium-compatible 586 @ 133 MHz
- Two/Four 24-bit RISC Servo Processors for Distributed Real Time Servo Loop Control
- Real Time Operating System provided on MPU for deterministic control
- MPU Handles Trajectory Generation, Digital I/O, Communications, and Programs
- Distributed Control Servo Processor Executes Real Time Control Algorithms
- Dedicated Optically Isolated Digital I/O for Limits, Registration, Faults & Enable)
- 8 Digital Inputs (Hardware registration)
- 8 Digital Outputs (Position Event Generation outputs)
- 3 Megabytes Flash Memory for Firmware Storage
- 1 Megabyte Flash Memory for User Programmable Functions
- 8 Megabytes RAM Memory
- Hardware Capture and Compare Registers
- 20 MHz Encoder Feedback, with Encoder Loss Detection
- Eight/Sixteen 14-bit Analog Input Interface
- Drive Command Signals (16-bit resolution)
- Software Support Tools
- Electronic Hardware and Software Manuals and Documentation

SPiiPlus PCI 4/8 Options:

- SPiiPlus PCI-INT – Internal mounting kit containing ribbon cables, terminal blocks, and RS232 serial cable for mounting the controller inside a PC.
- SPiiPlus PCI-SA – Stand-alone mounting kit containing a stand-alone mounting bracket, power cable, an additional RS232 serial cable, and everything contained in the INT kit, listed above.
- SPiiPlus PCI-ETR – Stand-alone Ethernet kit containing a 10BaseT daughterboard, an Ethernet communications cable, and everything contained in the SA kit, listed above.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SB Series

Introduction

The SB Series product line includes Multi-Axis Motion Controllers, as well as Universal Intelligent Motion Control Digital Drives. The current SB Series product line supports a one to four axis stand alone Motion Controller, a one to four axis PC based Motion Controller, and six one or two axis Universal Intelligent Motion Control Digital Drives from 100W to 7.5KW. All SB Series products are developed around a unique architecture in which each product has a main CPU and a state-of-the-art proprietary Servo Processor developed by ACS-Tech80. The main processor is an Intel 80196 and the Servo Processor is the SB2500.

All the SB Series products program identically and use the ACSPL programming language.

SB Series Products

ACS-Tech80 Products	Configuration	# of Axis	Communications	Digital I/O	Analog I/O
SB214SA	Controller	1-4	RS232/422	16 In / 16 Out Opto-isolated Sink type +5Vdc or +24Vdc	2 Inputs, 10-bit 4 Inputs, 12-bit 1 Output, 8-bit 4 Outputs, 10-bit
SB214PC	Controller	1-4	RS232/422 PCISA Bus	16 In / 16 Out Opto-isolated Sink type +5Vdc or +24Vdc	2 Inputs, 10-bit 4 Inputs, 12-bit 1 Output, 8-bit 4 Outputs, 10-bit
Unilite-1	Controller/Drive	1	RS232/422/485	4 In / 2 Out Opto-isolated Source type 5mA/output	None
Unilite-2	Controller/Drive	2	RS232/422/485	8 In / 8 Out Opto-isolated Source type 5mA/output	1 differential or 2 Single-ended Inputs, 12-bit 1 Output, 10-bit
SB1381	Controller/Drive	1	RS232/422/485	8 In / 8 Out Opto-isolated Sink or Source +5Vdc or +24Vdc	1 Input, 12-bit 1 Output, 10-bit
SB1292	Controller/Drive	2	RS232/422/485	16 In / 16 Out Opto-isolated Sink or Source +5Vdc or +24Vdc	2 differential Inputs, 12-bit 2 Single-ended Outputs, 10-bit
SB1391	Controller/Drive	1	RS232/422/485	8 In / 8 Out Opto-isolated Sink or Source +5Vdc or +24Vdc	1 differential Input, 12-bit 1 Single-ended Output, 10-bit

- All products have dedicated opto-isolated safety inputs for L & R limit per axis and E-stop.
- All models except SB214PC and Unilite have a 7-segment display for error, status, and programmable messages.
- All products take differential line driver encoders.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ACSPL Multi-Axis Motion Controllers

ACS-Tech80 Products	Power requirements	Output	Encoder Input
SB214SA	24Vdc @ 1.5A	+/- 10 VDC or Sinusoidal Commutation (Two +/- 10 VDC)	Differential line driver
SB214PC	5Vdc @ 1.5A +/-12Vdc @ .25A	+/- 10 VDC or Sinusoidal Commutation (Two +/- 10 VDC)	Differential line driver

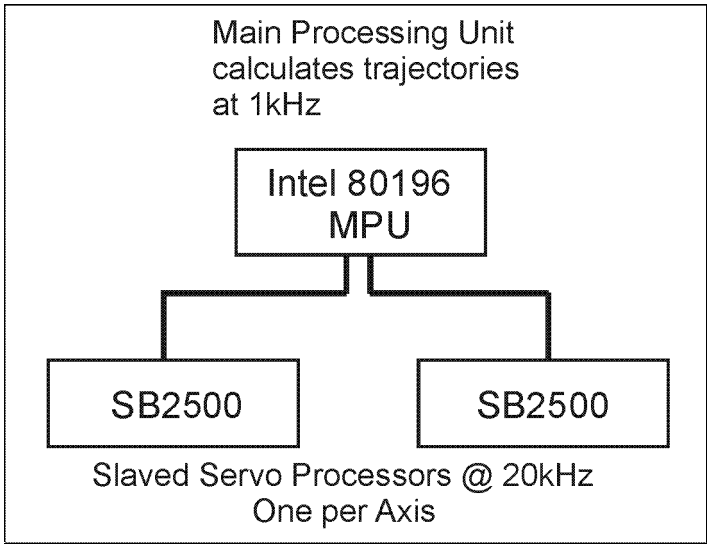
ACSPL Universal Intelligent Motion Control Modules

ACS-Tech80 Products	Power requirements	Output	Motors Supported
Unilite-1	15-36Vdc, 3A	100W, 3A-5Apeak	DC Brushed only
Unilite-2	15-36Vdc, 3A	2 @ 100W, 3A-5Apeak	DC Brushed only
SB1381B	43Vac or 60Vdc, 7.5A	450W, 7.5A-15Apeak	DC Brushed DC Brushless AC Induction
SB1381C	85Vac or 120Vdc, 5A	600W, 5.0A-10Apeak	DC Brushed DC Brushless AC Induction
SB1292B	24-60Vdc or 18-45Vac	2 @ 450W, 7.5A-15Apeak	DC Brushed DC Brushless AC Induction
SB1391A	85-265Vac, 5Ap-p	1KW, 5Ap-p	DC Brushed DC Brushless AC Induction
SB1391B	85-265Vac, 10Ap-p	3KW, 10Ap-p	DC Brushed DC Brushless AC Induction
SB1391C	85-265Vac, 15Ap-p	5KW, 15Ap-p	DC Brushed DC Brushless AC Induction

All drives are 20 kHz PWM digital voltage control

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

All SB Series products have significant speed and performance advantages over other motion controllers and intelligent drives. The multi-processor and distributed architecture allows the motion control tasks to be executed efficiently and seamlessly. The 80196 processor is referred to as the Motion Processor Unit (MPU). The MPU performs the high-level tasks including the trajectory calculations, communications, and I/O management tasks. The MPU performs a deterministic 1 millisecond MPU cycle time.



There is a slaved servo processor for each axis. Each distributed servo processor performs the real time servo loop calculations. By using a distributed architecture the servo loop performance is executed in 50 microseconds independent of the number of axes controlled. The slaved processors are supplied with a servo control algorithm. The servo control algorithm implements the control loops and hard real time control.

SB Series Products Highlights:

Hardware

- MPU is Intel 80196
- One 16-bit RISC Servo Processor SB2500 per axis
- Real Time Operating System provided on MPU for deterministic control
- MPU Handles Trajectory Generation, Digital I/O, Communications, and Programs
- Slaved Servo Processor Executes Real Time Control Algorithms
  - Dedicated Optically-Isolated Digital I/O for Home, Limits, and E-Stop
  - Hardware Capture and Compare Registers
- 20-MHz Encoder Feedback (Encoder Loss Detection)
- Drive Command Signals for Analog Drives

Communications

- RS-232/422/485 Serial Communications, PC Bus, (Reference chart above for product specifics)
- Software Development Tools for Diagnostics, Tuning, and Debugging
- Dynamic Link Libraries for Windows 95/98/NT Operating Systems (multi-threaded)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Firmware Highlights Execution

- Direct mode – commands are sent via serial or bus link and are executed immediately.
- Program mode – a sequence of commands (Program) is written. The program is compiled and saved in the controller's memory and can be invoked and executed at any time.

Variables and Program Flow

- Pre-assigned arrays for setting and reporting
- Automatic line numbering
- Full support of arithmetic, relational, logical, and mathematical operators
- Variable support for internal, program, set, report, and various arrays
- Variables can be set using the let command or using the "=" operator
- String support
- Ability to display messages
- If-Then-Else, Do-Else, Wait, Goto, and Call subroutine
- Built-in timers
- Built-in automatic subroutines (AUTOEXEC) for Emergency Stop, Position error, limit switches, inputs, timers, software limits, registration, etc.
- Function key assignment
- Step mode for debugging

Motion Profiling

Modes of Motion:

0. Linear Point to Point, smoothed trapezoid or parabolic profile
1. Repetitive Linear Point to Point
3. Move by Sequence, step through an array and wait at each point
4. Circular Point to Point
8. Point to Point with multiple velocities
9. Search for Contact, ideal for pick and place applications
10. Jogging (Continuous Velocity)
12. Master-Slave, ideal for flying cutoff
15. Continuous Path Following
16. Arbitrary Path Generation and Electronic CAM
21. Manual Control by Joystick, speed is proportional to an analog input
22. Position Tracking

Advanced Functions

- Error mapping for enhanced accuracy
- Search for Index
- Homing
- Registration Mark Movement
- Real Time Data Collection
- Velocity/Smoothness Measurement
- Position Event Generator, PEG function generates real time, position triggered events

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Motion Control Performance

- New Trajectory calculated every 1 millisecond
- Servo Update Rate of 20kHz (50 microseconds, independent of # of axes)
- Hardware Position Event Generation to 1 MHz
- Positioning Accuracy to 1 Encoder Pulse
- Velocity Accuracy to 1 Count/Second

Software Tools

- ACS Adjuster for Windows: Interactive tool for setting up, tuning, and viewing parameters. Includes soft oscilloscope.
- ACS Debugger for Windows: Development environment for ACSPL (ACS Programming Language) applications.
- ACS Saver/Loader for Windows: Tool for copying system setup and application data from one controller to another. Enables rapid setup of identical controller installations.
- ACSLIB Libraries (option): Comprehensive C/C++, and Visual Basic libraries for DOS, and Windows 3.11/95/98/NT. Multithreading support for Windows NT.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SB Series: CONTROLLER FEATURES

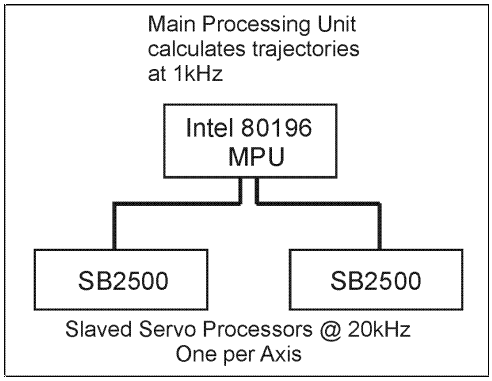
Overview

The SB Series controllers and intelligent drives are ideal for applications requiring a high degree of processing power and real time control performance. The controllers Motion Processor Unit (MPU) is provided with a built-in Real Time Operating System (RTOS). The RTOS installed on the MPU along with the dedicated servo processors eliminates the need or burden for a host processor. The controllers can function completely stand-alone and communicate over serial or PC Bus communications ports. The controllers can communicate back to a host PC for implementing an Application Programming Interface (API).

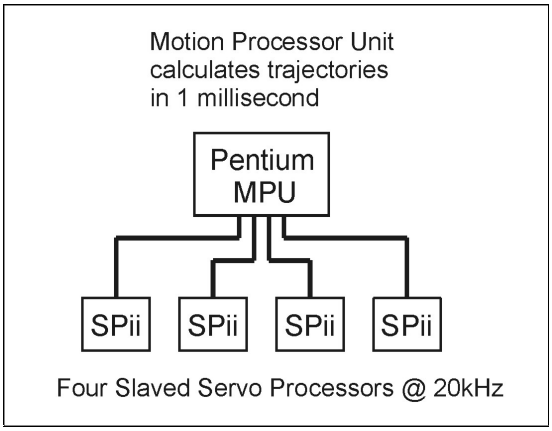
Motion Processor Unit (MPU)

The MPU is an Intel 80196 based processor running at 80Mhz. With the standard MPU, the controller executes programs on a 1 millisecond cycle time. The MPU has several functions, including:

- Feeds coordinated profiles to slaved servo processors
- Executes the firmware commands
- Executes the communications interface



The MPU is provided with a real time operating system to guarantee real time performance. An additional feature of the MPU is the status updates. Every parameter and flag, including user programmable parameters and flags, is updated at the MPU cycle rate. Thus, critical parameters and flags are always updated at the next scan rate. Other motion controllers with lower performing DSP's do not achieve this update rate.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The RTOS is important to allow deterministic control and the fast execution of motion programming. With the RTOS on-board in the ACS-Tech80 controller, the real time program execution can be passed to the controller. This frees up the host processor to perform the graphical user interface (GUI). Thus, the host processor does not require a real time operating system and the host can simply run standard operating systems. The most popular PC operating system is the Windows operating system. Here again, the programmer can implement off-the-shelf programming tools for C and Visual Basic more easily. This can greatly reduce the engineering development time (time to market) without having to implement a real time operating system in the host PC.

PC Based Control

The ACS-Tech80 solution is a true PC-based control based on open architecture. Unlike other motion controller designs, the design is not limited by the enhancement of RISC control. The design is based on the open standard of the personal computer industry. This philosophy allows superior future designs to be achieved merely by the default enhancement of using standard hardware. Clearly, the investments made in the standard PC hardware and software tools are further enhanced with the shear volume invested into this technology. The engineer can be assured that future designs can be more easily achieved without costly re-engineering to different platforms. The engineer can be confident that the ACS-Tech80 solution will be a valuable controller both now and in the future as future ACS-Tech80 designs can be ported easily to the PC platform. This is true since the PC industry has a distinct economy of scale advantage. The PC industry consumes $200 billion while the industrial control market consumes $2 billion annually. The PC technology is the reason that PC based motion control exists. Thus, it makes sense to base the selected design on a PC topology.

Trajectory Calculation/Servo Loop Update Rate

The ACSPL products are provided with high bandwidth trajectory and servo loop calculations. At ACS-Tech80, the true measure of a motion controller is the combination of both the trajectory calculation and the servo loop calculation speeds. These speeds determine the overall performance of the motion profile. The highest performance motion controllers will have fast trajectory and servo loop updates. The total performance measure would then include the ease of programming and development tools. The ACS-Tech80 products are superior in all cases.

The MPU performs the trajectory calculations. The default trajectory calculation is 1000 points per second (1000 Hz). For most applications, one millisecond trajectory calculation is sufficient.

Distributed Control Architecture

ACSPL products use distributed control architecture. This is the fundamental design concept of the controllers. The MPU has one slaved servo processor per axis of control. Each one of the slaved servo processors is a 16-bit fixed point RISC. The RISC guarantees that the real time control algorithms are executed in 50 microseconds regardless of the number of axes that are being processed. In addition to the servo loop, the RISC passes data to and from the MPU as required for motion profiling and data reporting. The slaved servo processors perform the real time control for the servo loop. The distributed control architecture allows the optimal performance for both the trajectory calculation and the servo loop calculation. Distributed control architecture also puts the intelligence where the most benefits can be achieved and optimized for performance. Thus, the ACS-Tech80 ACSPL products implement a distributed architecture so that the control algorithms can be optimized while maintaining an open system design.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SB2500 Servo Processor

The SB2500 Servo Processor is a state-of-the-art Servo Processor specifically designed and developed by ACS-Tech80 in order to provide better solutions to the high-demand needs of present and future machinery and industrial automation. This Servo Processor (SB2500) incorporates an 80 MHz 16-bit RISC core that is optimized for control algorithms and all the peripherals that are needed in order to implement a complete high performance servo system.

Each axis has a dedicated Servo Processor responsible for real time high-speed tasks. It is capable of implementing the standard position control filer and sinusoidal brushless current commutation and control in less than 25 microseconds.

- Real time high order filters at sampling rates of 20KHz:
  a) Position loop filter (P type, 32 bits accuracy)
  b) Velocity loop filter (PI + second LPF + notch, 32 bits)
  c) Velocity and acceleration feed-forwards
  d) Anti-reset windup (integral limit)

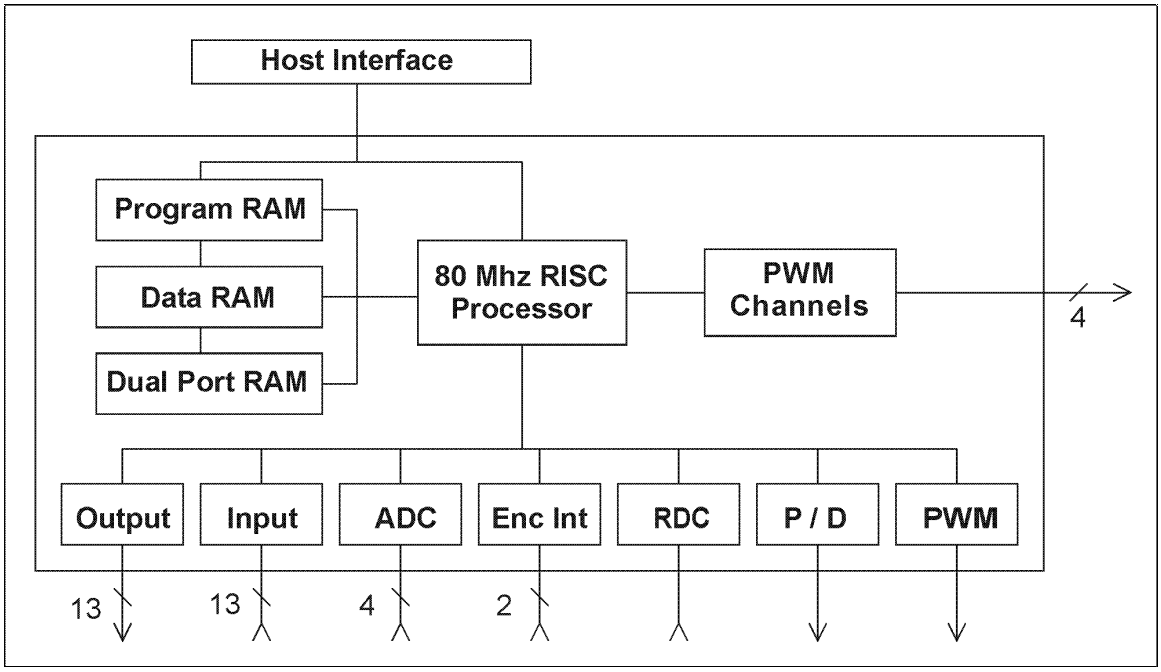
- Fully Digital Current Control (Control Modules Only):
  a) 4 x 12 bits fast ADC interface
  b) 4 PWM, 20 kHz pairs for direct FET/IGBT gate control
  c) Easy to optimize per motor
  d) Sine table for SW sinusoidal commutation
  e) Three current loops (PI type, 32 bits)
  f) Vector transformation for AC induction and brushless motors

- Robust and fast Encoder Interface:
  a) A quad B, clock-direction, up-down formats
  b) 3rd order input filter
  c) Encoder error detection
  d) 20KHz resolution
  e) No limitation for linear motor based applications
  f) Dual encoder feedback control (two position feedback loops)
  g) Optional Resolver feedback interface
  h) Optional Closed or Open loop analog feedback (on-the-fly changes)

- Programmable Event Generation (Hardware position compare with < 0.01 microsecond delay)
- A 10-bit analog output (Scope monitoring of parameters like following error)
- 16-bit analog outputs for interface to digital and analog drives
- Program, Data, and Dual Port RAM interface to Motion Processor Unit (built-in host)
- See diagram on following page

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

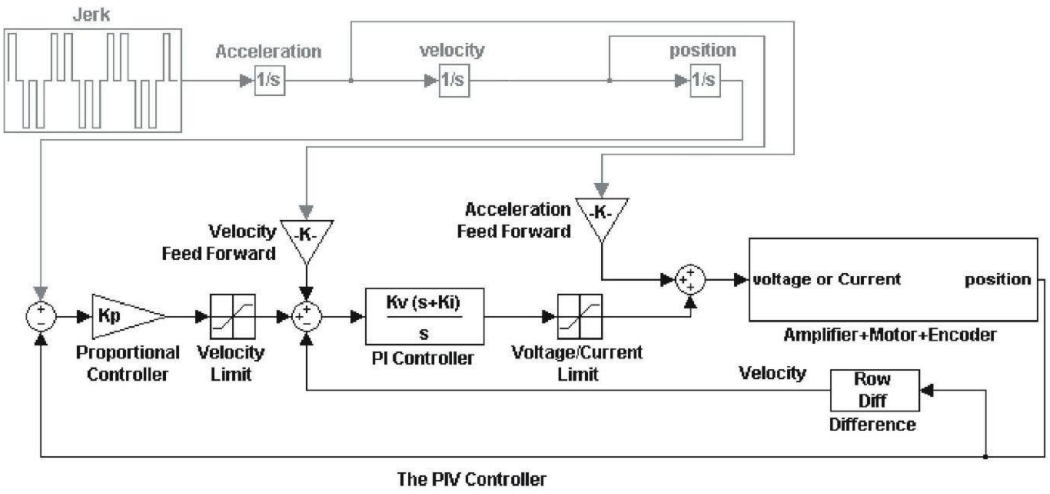
The SB2500 Servo Processor



ACS-Tech80 developed the SB2500 Servo Processor in 1994, using .6-micron semiconductor technology. It consists of a core 80MHz 16-bit RISC processor and a 32-bit MAC (Multiply Accumulate Unit).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

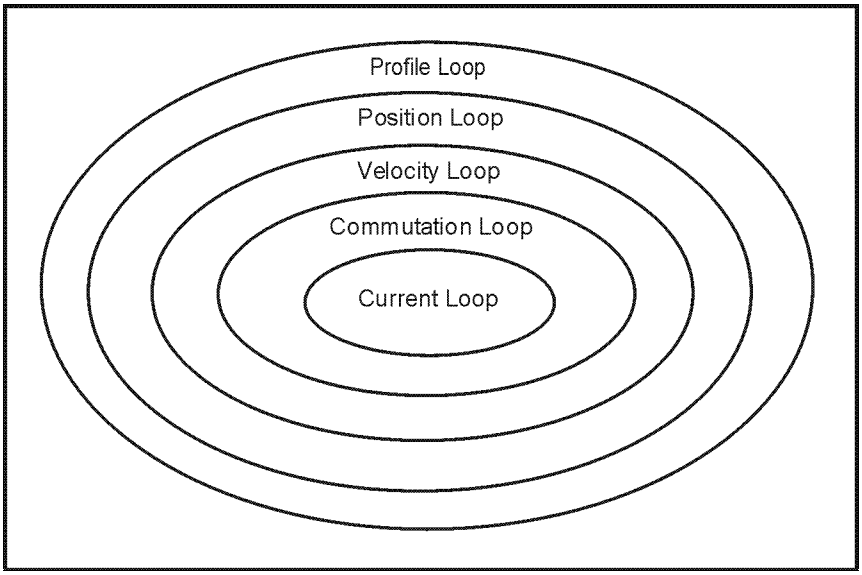
The SB2500 servo processor implements the servo control algorithm. The servo processor has been optimized for motion control applications. The algorithm is programmed into the servo processor in assembly language and optimized by the RISC to execute loops for current, velocity, position, and commutation; all at 20kHz sampling rate. The servo update rate is independent of the number of axes being controlled since there is one servo processor for each axis. In most applications the standard servo control algorithm is implemented. However, it can be customized in special OEM applications for support and engineering charges. The figure below depicts a block diagram of the control algorithm.



Control Algorithm Block Diagram

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Control Algorithm Loops



The ACSPL products adjustment method of the control algorithm is based on working from the inner loop out:

1. Current loop adjustment
2. Commutation adjustment
3. Velocity loop adjustment
4. Position loop adjustment
5. Polishing to improve the overall performance (acceleration feed forward, phase advance, slip constant, field current, etc.)

The Control Algorithm Features:

- Software configurable for DC brushless, AC induction, or DC brushed motor.
- High bandwidth digital controller for speed and accuracy.
- Sinusoidal commutation with automatic setup for three-phase motors.
- Fully digital position and velocity control at 20kHz sampling rate independent of the number of axes, for excellent dynamic and tracking performance.
- Dual loop control with two encoders - one mounted on the motor and one on the load for accurate belt-driven and lead-screw based applications.
- Tight control and automatic velocity feedforward control.
- Programmable automatic routines for fault, error, and exception.
- Real-time data collection. Programmable sampling rate up to 1kHz.
- Analog input can be used as feedback.
- State-of-the-art 20kHz/40kHz PWM power bridge with optimized current ripple and efficiency.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SB Series: FIRMWARE

Firmware Overview

The on-board firmware command set is highly dynamic and flexible. A number of variables or mathematical expressions can be programmed and interpreted at the MPU cycle rate (default MPU cycle rate is 1 millisecond). The ACSPL programming language is interpreted by the MPU before execution. It guarantees fast execution and reduced processing overhead. Built-in autoroutines simplify programming. (See page 98.)

Firmware Commands

ACSPL allows communications with the controller in two modes:

- **Direct Mode -** In this mode, the host computer communicates with the controller via a serial communication or bus link. Each command is interpreted and executed immediately after it is typed.
- **Programming Mode -** In this mode, a program is written. The program is then saved and compiled in the controller's memory and can be activated at any time.

Communication with the controller is done using the following **M**aster **C**ommands (**MC**):

Direct mode commands

S	**S**etting parameters and outputs
R	**R**eporting parameters, inputs and outputs status
G	Defining the **G**roup reports' structure
T	**T**elling and changing status
A	Setting and reporting the various **A**rrays
B	Motion execution **B**egin
M	Motion execution **M**ove
GO	Motion execution **G**o
E	Motion execution **E**nd
K	Motion execution **K**ill
KM	Motion execution **M**aster
W	**W**aking up another controller in a multi axis environment
C	a**C**knowledge ACSPL display messages
RESET	Memory management command
CLEAR	Memory management command
SAVE	Memory management command
LOAD	Memory management command
HWRES	Memory management command

A master command may be followed by a **S**ub**C**ommand (**SC**), axis definition (**axis**), and/or data (**data**). It is terminated with a **C**arriage **R**eturn (**cr**).

- **M**aster **C**ommands must be typed in uppercase letters.
- Axes (**axis**) must be typed in uppercase letters.
- **S**ub **C**ommands are typed according to the dictionary.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Firmware Commands (continued)

Tell Master Commands

T0 - Tell Motor And Motion Status
T1 - Tell Result Of Last 'Begin Motion' Command
T2 - Tell The Cause Of Motion's End
T3 - Tell Program Status (For Program Mode Only)
T4 - Tell Last Program Message (For Program Mode Only)
T5 - Tell Servo Processors Status
T6 - Tell Firmware Memory Status
T8 - Tell Voltages Status
T9 - Tell Memory Status

Program Mode Commands

Editing Commands

Program
Insert
Delete
List
List Labels
Compile
Execute
Stop
Saving The Program

Program Flow Commands

Till
Goto
Call
Loop
If
Do
Case
Break
Disable
Enable

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Operators

ACSPL supports the following operators, which are classified into four major groups:

- Arithmetic
- Relational
- Logical
- Mathematical

Arithmetic Operators

+	**Addition**	V0= V1+100
-	**Subtraction**	V0= V1-100
*	**Multiplication**	V0= V1*100
/	**Division**	V0= V1/100

Relational Operators

=	**Equals**	if XRP=10000
<	**Smaller than**	till XCP<2000
>	**Greater than**	till YCP>5000
<=	**Smaller than or equal to**	if XCP<=1000
>=	**Greater than or equal to**	till XLV>=7500
<>	**Not equal**	if XLA<>500000

Logical Operators

&	**And**	V1=V2 & V3
|	**Or**	V2=V4 | V5
^	**Not**	V3=^V4
~	**XOR**	V1=V0 ~ IP

Mathematical Operators

abs	**Absolute of a variable**	V1=abs(V0)
min	**Minimum value of an array**	V1=min(TX)
max	**Maximum value of an array**	V1=max(TX)
avg	**Average value of an array**	V3=avg(TY)
sqr	**Square root of a variable**	V6=sqr(V10)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Variables, Flags and Parameters

A variable is a name of a location in memory that holds information.
The following types of variables are supported:

- Internal
- Program
- Set
- Report
- Target points arrays
- Continuous path arrays
- Sine tables and offsets
- Servo processor's arrays

Internal

An internal variable holds information that describes motion status.
These variables are read only and cannot be changed by the program.

Program

The program variables are pre-defined locations in the memory of the controller for use by a program. Each variable has its own valid range of values and may be read or written by a program. Included are indices, timers, and user variables.

Set and Report Variables

Set variables can be altered by the user by using the **SET** command in direct mode or by a **let** assignment within a program.

Report variables can be accessed through the **REPORT** command direct mode or by using their mnemonics within a program. (A full list of the set/report parameters will follow.)

Target Points Arrays

An array of 1024 elements per axis. This array is used by Motion Modes 3 and 16 and by **PEG** and **Data Collection** functions.

Continuous path arrays

The segment of a path used in Motion Mode 15 **Continuous Path**.

Sine tables and offsets

These variables are used for commutation of AC and DC brushless motors.

Servo processor's arrays

These arrays are used for communication between the main CPU and the Servo Processor(s).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ASCPL SET and REPORT Parameters

Servo Control Parameters

Sub Com.	Description	Axis	Group
AF	Acc. Feed forward	X/Y/Z/T	S/R
D4	D4 analog output	X/Y/Z/T	S/R
DA	DAc offset	X/Y/Z/T	S/R
DO	Dac Output	X/Y/Z/T	R
DT	Drive Type	X/Y/Z/T	S/R
FT	Filter Type	X/Y/Z/T	S/R
GA	GAin	X/Y/Z/T	S/R
GF	Gain Factor	X/Y/Z/T	S/R
KZ	coeff.(K) 2nd Zero	X/Y/Z/T	S/R
PF	Pwm Frequency	X/Y/Z/T	S/R
PO	POle	X/Y/Z/T	S/R
SA	SAmpling rate-kHz	X/Y/Z/T	S/R
ZE	ZEro	X/Y/Z/T	S/R

Stepper Control Parameters

Sub Com.	Description	Axis	Group
CL	Current Level (%)	X/Y/Z/T	S/R
EC	ver. Enc. Constant	X/Y/Z/T	S/R
EF	ver. Enc. Factor	X/Y/Z/T	S/R
P0	P0 Idle Current	X/Y/Z/T	S/R
P1	P1 Jerk Current	X/Y/Z/T	S/R
P2	P2 Acceleration Current	X/Y/Z/T	S/R
P3	P3 Velocity Current	X/Y/Z/T	S/R
P4	P4 Deceleration Current	X/Y/Z/T	S/R
PW	Pulse Width (stepper)	X/Y/Z/T	S/R
VE	Verify Enable/Disable	X/Y/Z/T	S/R

Commutation Parameters

Sub Com.	Description	Axis	Group
_h	Hall Offset	X/Y/Z/T	S/R
_m	Index Offset	X/Y/Z/T	S/R
CO	Commutation Offset	X/Y/Z/T	S/R
CU	Commutation Usage	X/Y/Z/T	S/R
FC	Field Current	X/Y/Z/T	S/R
HA	HAll effect reading	X/Y/Z/T	R
LF	Lines Factor	X/Y/Z/T	S/R
LR	Lines per Rev.	X/Y/Z/T	S/R
NP	Number of motor's Poles	X/Y/Z/T	S/R
PA	Phase Advance	X/Y/Z/T	S/R
RS	ReSolver reading	X/Y/Z/T	R
SK	Slip constant	X/Y/Z/T	S/R
XT	aXis Type	X/Y/Z/T	S/R

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ASCPL SET and REPORT Parameters (continued)

Motion Parameters

Sub Com.	Description	Axis	Group
AO	Array Offset	X/Y/Z/T	S/R
AP	next trgt Abs Pos	X/Y/Z/T	S/R
AV	Actual Velocity	X/Y/Z/T	R
BP	Begin Position	X/Y/Z/T	S/R
BV	Begin Velocity	X/Y/Z/T	S/R
CA	Current Angle		R
CC	Center Coordinates	X/Y/Z/T	S/R
CM	Current Motion Mode	X/Y/Z/T	R
CP	Current Position	X/Y/Z/T	R
D1	DAC 1 command	X/Y/Z/T	R
D2	DAC 2 command	X/Y/Z/T	R
DP	Desired Position	X/Y/Z/T	R
E2	Encoder #2 position	X/Y/Z/T	R
ED	Enable Delay time	X/Y/Z/T	S/R
ES	Executed Step	X/Y/Z/T	R
ET	Encoder Type	X/Y/Z/T	S/R
Et	2nd Encoder type	X/Y/Z/T	S/R
IF	Interval Factor	X/Y/Z/T	S/R
KD	Kill Deceleration	X/Y/Z/T	S/R
la	lower acceleration	X/Y/Z/T	S/R
LA	Linear Acceleration	X/Y/Z/T	S/R
ld	lower deceleration	X/Y/Z/T	S/R
LD	Linear Deceleration	X/Y/Z/T	S/R
LI	Array's Low Index	X/Y/Z/T	S/R
LV	Linear Velocity	X/Y/Z/T	S/R
M2	Mark 2 position	X/Y/Z/T	R
MA	MAster slave configuration	X/Y/Z/T	S/R
MD	Master Distance	X/Y/Z/T	S
MM	next Motion Mode	X/Y/Z/T	S/R
MO	MOtor enabled	X/Y/Z/T	S/R
MP	Master Position	X/Y/Z/T	R
MR	Master Relative position	X/Y/Z/T	R
MS	Master Synchronization	X/Y/Z/T	R
MV	Master Velocity	X/Y/Z/T	R
MZ	Master Zero position	X/Y/Z/T	S
NF	Next Full step	X/Y/Z/T	R
PE	Position Error	X/Y/Z/T	R
PG	Path Generation mode	X/Y/Z/T	S/R
PT	Path Time	X/Y/Z/T	S/R
RA	Rotation Angle		S/R
RD	Registration Distribution	X/Y/Z/T	S/R
RP	next target Relative Position	X/Y/Z/T	S/R

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ASCPL SET and REPORT Parameters (continued)

Motion Parameters (continued)

Sub Com.	Description	Axis	Group
SD	Search Distance	X/Y/Z/T	S/R
SE	SEnsor input source	X/Y/Z/T	S/R
SF	Smooth Factor	X/Y/Z/T	S/R
SH	Sensor High time	X/Y/Z/T	S/R
SO	ShOrt time	X/Y/Z/T	S/R
SP	Sensor Point	X/Y/Z/T	S/R
SQ	SeQuence type	X/Y/Z/T	S/R
SS	Start Sensor position	X/Y/Z/T	R
SV	Search Velocity	X/Y/Z/T	S/R
TC	Tracking Constant	X/Y/Z/T	S/R
TD	Target rad. Delay	X/Y/Z/T	S/R
TE	Tracking Error	X/Y/Z/T	R
TF	Tracking Factor	X/Y/Z/T	S/R
TR	Target Radius	X/Y/Z/T	S/R
TS	Track Sync. Range	X/Y/Z/T	S/R
TV	Tracking Velocity	X/Y/Z/T	R
UI	array's Upper Index	X/Y/Z/T	S/R
VA	Vector Acceleration		S/R
VD	Vector Deceleration		S/R
VK	Vector Kill deceleration		S/R
VM	Velocity Minimal	X/Y/Z/T	R
VO	search Velocity Output	X/Y/Z/T	S/R
VS	Vector Smooth factor		S/R
VV	Vector Velocity		S/R
WT	Wait Time	X/Y/Z/T	S/R
zp	zero position of 2nd encoder	X/Y/Z/T	S
ZP	Zero Position	X/Y/Z/T	S

Safety & Protection Parameters

Sub Com.	Description	Axis	Group
BO	Break Open	X/Y/Z/T	S/R
CL	Current Level	X/Y/Z/T	S/R
EA	MAXIMUM ERROR IN ACCELERATION	X/Y/Z/T	S/R
ER	Maximum Error	X/Y/Z/T	S/R
IL	safety Inputs Logic		S/R
PH	Position High	X/Y/Z/T	S/R
PL	Position Low	X/Y/Z/T	S/R
PP	Prog. Protect line		S/R
QP	Qarea Protect line	X/Y/Z/T	S/R
TL	Torque Limit	X/Y/Z/T	S/R
TO	Maximum TOrque	X/Y/Z/T	S/R

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ASCPL SET and REPORT Parameters (continued)

Special Functions Parameters

Sub Com.	Description	Axis	Group
AL	Array Low	X/Y/Z/T	S/R
AS	Array Step	X/Y/Z/T	R
AU	Array Upper	X/Y/Z/T	S/R
BG	Begin on input/GO	X/Y/Z/T	S/R
CG	Current position-Grid coordinate	X/Y/Z/T	R
DC	Data Collection mask	X/Y/Z/T	S/R
DL	Data coll. Low index		S/R
DS	Data coll. exec. Step		R
DU	Data coll. Upper index		S/R
EG	Event Gen. mode	X/Y/Z/T	S/R
EL	End Location	X/Y/Z/T	S/R
EM	Error Mapping		S/R
FA	Functions Avail.	X/Y/Z/T	S/R
FE	Function Enable		S/R
GC	Grid Correction	X/Y/Z/T	R
GD	Grid Delta	X/Y/Z/T	S/R
GO	Grid Offset	X/Y/Z/T	S/R
GS	Grid Size	X/Y/Z/T	S/R
IX	IndeX position	X/Y/Z/T	R
M1	Mark 1 position	X/Y/Z/T	R
MI	Master Index	X/Y/Z/T	R
MT	Match Time (peg)	X/Y/Z/T	S/R
PI	Position Interval	X/Y/Z/T	S/R
PM	Peg Mask		S/R
SL	Start Location	X/Y/Z/T	S/R

System Parameters

Sub Com.	Description	Axis	Group
A0	Analog input 0	X/Y/Z/T	R
A1	Analog input 1	X/Y/Z/T	R
A2	Analog input 2	X/Y/Z/T	R
A3	Analog input 3	X/Y/Z/T	R
A4	Analog input 4	X/Y/Z/T	R
A5	Analog input 5	X/Y/Z/T	R
AN	Acknowledge Number		R
at	acknowledge time		S/R
BE	Begin/End masking		S/R
BR	Baud Rate		S/R
D0	D0 analog output		S/R
d1	display constant 1	X/Y/Z	S/R
d2	display constant 2	X/Y/Z	S/R
d3	display constant 3	X/Y/Z	S/R

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ASCPL SET and REPORT Parameters (continued)

System Parameters (continued)

Sub Com.	Description	Axis	Group
df	DISPLAY FUNCTION	X/Y/Z	S/R
dh	display hysteresis	X/Y/Z	S/R
dm	display maximum value	X/Y/Z	S/R
dr	display rate	X/Y/Z	S/R
dv	display value	X/Y/Z	S/R
EI	Extended Input (32)		R
FG	Flag reGister		R
FL	Field Length		S/R
FM	Function Mapping		S/R
FR	Flash eRror counter		R
G#	Group #	X/Y/Z/T	R
HI	HIgh output #		S
HP	Host Protocol		S/R
HT	Host / Terminal		S/R
IC	Initiate Comms.		S/R
ID	controller's ID		S/R
IN	INput port logic		S/R
IP	Input Port		R
IS	Input Source	X/Y/Z/T	S/R
KB	Keyboard Input		R
LO	LOw output #		S
LT	Lower Threshold	X/Y/Z/T	S/R
MF	Monitor Factor	X/Y/Z/T	S/R
MN	Motion moNitor	X/Y/Z/T	S/R
N1	extended iNput 1		R
N2	extended iNput 2		R
N3	extended iNput 3		R
N4	extended iNput 4		R
O1	extended Output 1		S/R
O2	extended Output 2		S/R
OL	Output Logic		S/R
OM	Output Mask		S/R
OP	Output Port		S/R
SG	Status reGister		R
SI	System Input		R
SN	Serial Number		R
st	program step mode		S/R
ti	time interval		S/R
TN	Transmission Number		R
tt	transmission time		S/R
UT	Upper Threshold	X/Y/Z/T	S/R
VN	Vector Number		S/R
VR	software VeRsion		R

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Autoroutines

An Autoroutine is a subroutine which is invoked automatically when a certain condition is met. Autoroutines must follow these rules:

- Its label must be one of the reserved labels
- It should be terminated with a **ret** or **stop** command

If a condition is met while the controller is not busy with another program, the autoroutine starts running in less than one millisecond. If a condition is met while a program is running, the current statement is completed, and then the relevant autoroutine is invoked. Upon a **ret** command, the controller continues with the next statement of the interrupted program (unless the INT_LINE variable was altered by the autoroutine). If a **stop** command is encountered in the autoroutine, the program stops. Autoroutines may also be activated by a **call** command.

AUTOEXEC When a program is labeled AUTOEXEC, it is executed automatically two seconds after the system's power-up. The delay enables the user to interrupt (via the communication link) and prevent the execution. If any character(s) is received within the first two seconds, the execution of the AUTOEXEC is prevented. When partial communication is used, AUTOEXEC is executed without any delay. The AUTOEXEC is not executed automatically if both limit switches of the X-axis are ON on power up. This procedure should be used to prevent the execution of the AUTOEXEC subroutine.

AUTO_ES Invoked when the motor is enabled and one of these conditions is met:

- **E**mergency **S**top signal is active. (Error 22)
- Drive fault signal is detected. (Error 26)

AUTO_ERR Invoked when the absolute **P**osition **E**rror (of either X , Y, Z or T):

- Exceeds the **ER**ror limit (**ER**), while acceleration is zero
- **E**rror during **A**cceleration (**EA**) when accelerating/decelerating (end message 25)

AUTO_LL$ Invoked if **L**eft **L**imit switch is activated

AUTO_RL$ Invoked if **R**ight **L**imit switch is activated

AUTO_IN# Invoked when **INPUT # (1 TO 5)** is activated
The polarity of this signal is defined by the **IN**put logic (**IN**) parameter

AUTO_T1 Invoked when **TIMER1** is decreased from 1 to 0

AUTO_T2 Invoked when **TIMER2** is decreased from 1 to 0

AUTO_FLT is a general purpose handling subroutine that is invoked when the causing error condition has not invoked the specific error related automatic subroutine. For example, when the Emergency Stop input is activated, generating End error message 22, **AUTO_FLT** is invoked if **AUTO_ES** does not exist.

AUTO_SL$ Invoked when a left or right software limit is reached. Useful when using master/slave mode

AUTO_M1$ Invoked when the axis registration mark is activated and the $_MARK1 was false. The purpose of this routine is to stop a motor very accurately relative to a sensor that is connected to the registration mark input.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Autoroutines (continued)

AUTO_EX$** When the profile generation is done, in modes 0, 1, and 3, the controller starts an internal clock. This internal clock is initialized to a user-defined parameter: **T**arget radius **D**elay (**TD**). The parameter defines the time period (in milliseconds) in which the motor is allowed to be out of the **T**arget **R**adius without any notice. However, if this time period has elapsed and the motor is out of the **T**arget **R**adius for at least one millisecond, **AUTO_EX$ is invoked. The user can tailor the content of **AUTO_EX$** to the needs of the application (send a message, operate an output, etc.). The **AUTO_EX$** routine is invoked only once per motion. If a **Kill** command was issued before the routine has been invoked, it is not invoked during the current motion.

AUTO_EN$** Invoked when either **$_ENER (encoder error), or **$_ENCNC** (encoder not connected) for the $ axis becomes true (End message 21)

AUTO_TE$** Invoked when **abs($PE) > $TE while the motor is not accelerating or not decelerating. The purpose of this routine is to detect if during motion the instantaneous position error exceeds the Tracking Error (TE) parameter.

AUTO_HT$** Invoked when **$_HOT becomes true indicating that the motor is hot. This feature is available for products with built-in drives.

FKEY_0# Invoked when one of the function keys is pressed. The function key must have been defined earlier in the program.

Real Time Position Event Generator (PEG)

The controllers are provided with a hardware Position Event Generator. The PEG is controlled by the Servo Processor to ensure accuracy. The PEG is ideal for applications where position is synchronized with the digital outputs. The PEG is for high-accuracy position marking with accuracy of +/-1 count at up to 5,000,000 counts/second. Two modes of operation are supported:

Random Mode

The pattern of seven outputs can be predefined at up to 512 points. Once the axis reaches a trigger point, the state of the output port changes instantaneously (electronic gate delay). In addition, a pulse with a programmable period is generated at each trigger point by an additional output. Five events can be defined per 0.001 second. The minimum time interval between two adjacent events is less than 0.25 microseconds.

Incremental Mode

A programmable pulse is generated whenever a predefined distant is passed. The start point and the distance between the points are programmable. The minimum time interval between two adjacent events is less than 0.25 microseconds.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Modes of Motion

The Motion Mode is set by Set Motion Mode (S$MM#) Command. When the mode is set, a Begin Command must be sent in order to start motion. Before sending a Begin Motion Command each motor must be enabled by S$MO1 Command.

Changing motion parameters (velocity, acceleration) while in motion (on-the-fly) does not affect the present motion, with an exception: In Jogging Mode (mode 10) and Joystick Mode (mode 21), changing the Linear Velocity (LV) after a Begin Command has been issued causes an immediate change.

Unless the mode requires a coordinated motion, each axis may be operated in a different mode, with its own set of motion parameters. For example, X motor is running in jogging mode and Y motor is running in linear point-to-point mode. For a multi-axis motion (linear, circular, etc.), both axes must be in the same mode.

Example:

SXLA20000 Sets the Linear Acceleration to 20000
SXLD10000 Sets the Linear Deceleration to 10000
SXLV500 Sets the Linear Velocity for the X-axis to 500
SXMM10 Sets motion mode for X to 10 (jogging)
SXMO1 Enables the X-axis motor
SXLV-1000 Changes the Linear Velocity of X to –1000

When the motor is enabled (SXMO1) it will accelerate from 0 to 500 (encoder pulses per second) at the Linear Acceleration rate of 20000.

Upon execution of the SXLV-1000, the motor will decelerate from 500 to 0 at the Linear Deceleration rate of 1000, It will then reverse directions and accelerate to –1000 at the Linear Acceleration rate of 20000.

Modes of Motion

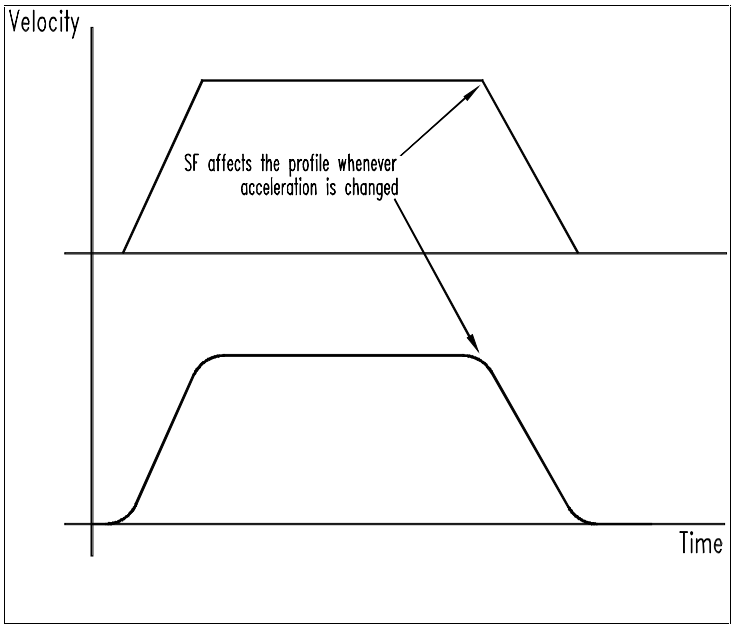
Mode 0	**Linear Point to Point - smoothed trapezoid or parabolic profile**
Mode 1	**Linear Point to Point - Repetitive**
Mode 3	**Move by Sequence, step through an array and wait at each point**
Mode 4	**Circular Interpolation - Point to Point**
Mode 8	**Multiple Velocities - Point to Point**
Mode 9	**Search for Contact - ideal for pick and place applications**
Mode 10	**Jogging (Continuous Velocity)**
Mode 12	**Master-Slave - ideal for flying cutoff**
Mode 15	**Continuous Path Following**
Mode 16	**Arbitrary Path Generation and Electronic CAM**
Mode 21	**Manual Control by Joystick, speed is proportional to an analog input**
Mode 22	**Position Tracking**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Modes of Motion (continued)

Mode 0 - Linear Point-To-Point (PTP)

In this mode, each motor moves to the next desired Absolute Position, using a smoothed trapezoidal velocity profile. This mode can be used for non-interpolated and interpolated move commands. The motion can be activated by a Begin command or through an external input.



Smoothed Trapezoidal Profile

While the motor moves, the target position for the next move can be specified and a Begin Command can be issued. The controller calculates the motion parameters of the next move, and the new move is executed upon completion of the present motion. Using this method, the (calculated) delay between two successive motions can be minimized.

Program Example:

```
EXMM00_A:
rem Filename: MM00_Basic_X-Axis_v110.prg
rem VERSION: 1.10   DATE: 01-30-01   AUTHOR: Douglas A. Wright

! PURPOSE:ACSPL Training example Mode 0 X axis Linear point to point
! Simple point to point move.  3 revolutions cloclwise.


CALC:
! SYSTEM: Motor max speed = 5000 RPM = 83.3 RPS
!         Encoder 1000 lines/rev = 4000 counts/rev
!         Max speed = 333,333 Counts/Second
!         4000 cps = 1 RPS

let XMM = 0
   ! X axis Motion Mode = 0 = point to point
let XLA = 10000
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
   ! X axis Linear acceleration = 10,000 counts/sec^2
let XLV = 4000
   ! X axis Linear Velocity = 4,000 counts/sec
let XLD = 20000
   ! X axis Linear deceleration = 20,000 counts/sec^2

let XMO = 1
   ! X axis MOtor Enable = 1 = ON
wait 50
   ! Wait for settling, then proceed.
let XZP = 0
   ! Zero X primary encoder position.


let XAP = 12000
   ! X axis Relative Position, target distance.  (Must follow MOtor
Enable).

exec BX
   ! Begin X axis move. (Immediate Mode command)

till ^ X_MOVE
   ! Wait until X axis profile stops.

let XMO = 0
   ! X axis MOtor Enable = 0 = OFF
stop
   ! End of program
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Modes of Motion (continued)

Mode 1 - Repetitive Linear Point-To-Point

In this mode, each axis moves repetitively between the last two target points, stopping at each point for a period defined by the Wait Time parameter.

Program Example:

```
EXMM01_A:
rem FILENAME: MM01_Basic_v110.prg
rem VERSION: 1.10   DATE: 02-01-01   AUTHOR: Douglas A. Wright

! REQUIRES: ACS-Tech80 Classroom SB1292 motor fixture.

! PURPOSE: ACSPL Example Code.  Mode 1 Linear point to point,
repetitive.
!          From "ACS Motion Controllers Software Guide" 1st Edition
section 3.2
!          Motor automatically goes back and forth.

! USAGE: ! Input # 2 stops program.


PROMPT:
disp " "
disp "Axis moves back and forthe automatically.  IN2 stops program."
wait 1000
   ! Delay 1000 mSec


IMITAXIS:
let XMM = 1
   ! X axis motion mode = 1 = point to point, repetitive.
let XMO = 1
   ! X axis MOtor Enable = 1 = ON
wait 50
   ! Wait for settling, then proceed.
let XZP = 0
   ! Zero X primary encoder position.


PROFILE:
let XLA = 30000
   ! X axis Linear Acceleration = 30,000 counts/sec^2
let XLV = 8000
   ! X axis Linear Velocity = 8,000 counts/sec
let XLD = 30000
   ! X axis Linear deceleration = 30,000 counts/sec^2
let XWT = 500
   ! X axis Wait Time.  Delay between all moves, both directions.
let XRP = 4000
   ! X axis Relative Position, target distance.


MAIN:
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
exec BX
   ! Begin X axis move. (Immediate Mode command)
disp "IN2 stops program"
till IN2
   ! Wait until IN2 is ON.
till ^ X_MOVE
   ! Wait until X axis profile is completed.


COMPLETE:
wait 1000
   ! Delay 1000 mSec
let XMO = 0
   ! X axis MOtor Enable = 0 = OFF
stop
   ! End of program.
```

Mode 3 - Move By Sequence

In this mode, the axis (axes) steps through an array of pre-defined position coordinates, stopping at each point for a period defined by the Wait Time ($WT) parameter. If more than one axis is involved, then the largest of the individual WT is used. The maximum array size is 1024 for each axis.

Program Example:

```
EXMM03_A:
rem FILENAME: MM03_Basic_v110.prg
rem VERSION: 1.10   DATE: 02-01-01   AUTHOR: Douglas A. Wright

! REQUIRES: ACS-Tech80 Classroom SB1292 motor fixture.

! PURPOSE: ACSPL Example Code.  Mode 3 Move by sequence.
!          From "ACS Motion Controllers Software Guide" 1st Edition
section 3.3
!          Axis one and two move through positions in the axis target
array.
!          Table is shifted left from the one in the book. Now it
starts @ 0,0


PROMPT:
disp " "
disp "Profile moves out through array and then back.
wait 1000
   ! Delay 1000 mSec


INITVARS:
let OM = 0
   ! no dedicated IO reassignments
let PM = 0
   ! no PEG IO reassignments
let IN = 0
   ! no input port inversions assigned
let IL = 0
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
   ! no safety input inversions assigned
let OL = 0
   ! No drive enable inversions.
let OP = 0
   ! Clear all digital outputs.


INIT_X:
let XMM = 3
   ! X axis motion mode = 3 = Move by Sequence.
let XMO = 1
   ! X axis MOtor Enable = 1 = ON
wait 50
   ! Wait for settling, then proceed.
let XZP = 0
   ! Zero X primary encoder position.
let Xzp = 0
   ! Zero X auxiliary encoder (not required in this program)
let XLI = 0
   ! X axis Upper Index.  Target array pointer.
let XUI = 4
   ! X axis Lower Index.  Target array pointer.
let XSQ = 0
   ! X axis SeQuence type.  0 = start at LI and go to UI.
let XWT = 250
   ! X axis Wait Time.  mSec How long to pause between all moves.


INIT_Y:
let YMM = 3
   ! X axis motion mode = 3 = Move by Sequence.
let YMO = 1
   ! X axis MOtor Enable = 1 = ON
wait 50
   ! Wait for settling, then proceed.
let YZP = 0
   ! Zero X primary encoder position.
let Yzp = 0
   ! Zero X auxiliary encoder (not required in this program)
let YLI = 0
   ! Y axis Lower Index.  Target array pointer.
let YUI = 4
   ! Y axis Upper Index.  Target array pointer.
let YSQ = 0
   ! Y axis SeQuence type.  0 = start at LI and go to UI.
let YWT = 250
   ! X axis Wait Time.  mSec How long to pause between all moves.


PROFILE:
let XLA = 30000
   ! X axis Linear Acceleration = 30,000 counts/sec^2
let XLV = 8000
   ! X axis Linear Velocity = 8,000 counts/sec
let XLD = 30000
   ! X axis Linear Velocity = 8,000 counts/sec
let XKD = 30000
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
   ! X axis Kill Acceleration = 30,000 counts/sec^2

let YLA = 30000
   ! Y axis Linear Acceleration = 30,000 counts/sec^2
let YLV = 8000
   ! Y axis Linear Velocity = 8,000 counts/sec
let YLD = 30000
   ! Y axis Linear deceleration = 30,000 counts/sec^2
let YKD = 30000
   ! Y axis Kill Acceleration = 30,000 counts/sec^2


ARRAY:
exec AXSE0 0 1000 0 -1000 0
exec AYSE0 0 1000 2000 1000 0


MAIN:
exec BXY
   ! Begin XY Vector move. (Immediate Mode command)
till X_STEP =  XUI
   ! Wait until segment pointer is at the UI end of the table.
till ^ X_MOVE
   ! Wait for last segment to complete.

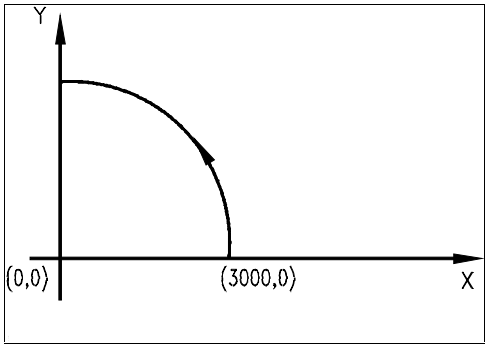

COMPLETE:
wait 1000
   ! Delay 1000 mSec
let XMO = 0
   ! X axis MOtor Enable = 0 = OFF
let YMO = 0
   ! X axis MOtor Enable = 0 = OFF
stop
   ! End of program.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Modes of Motion (continued)

Mode 4 - Circular Point-To-Point

In this mode (assuming that X and Y motors are connected to an XY table), a table can be moved on an arc (circular interpolation). The movement starts at the present (if at rest) XY location along an arc defined by Rotation Angle (RA) and the Center Coordinates (CC). The tangent velocity and acceleration are determined by the Vector Velocity and Vector Acceleration parameters.



Circular point to point

Programming Example:

```
MM04_90:
rem FILENAME: MM04_90-Degrees_v110.prg
rem VERSION: 1.10   DATE: 02-01-01   AUTHOR: Douglas A. Wright

! REQUIRES: ACS-Tech80 Classroom SB1292 motor fixture.

! PURPOSE: ACSPL Example Code.  Mode 4 Circular Interpolation. 90
degree circle.
!          From "ACS Motion Controllers Software Guide" 1st Edition
section 3.4

! ALGORITHM:  Simple version form the book.  See diagram section 3.4.
!  1. Preset location of X axie with XZP.
!  2. Then X + Y axes do a 90 degree arc.
!  3. Then X axis is NOT returned to the start, or 0,0.


PROMPT:
disp " "
disp "Set parameters to pre position the X axis, linear point to point"
disp "Pre position X axis before arc"


INITVARS:
let OM = 0
   ! no dedicated IO reassignments
let PM = 0
   ! no PEG IO reassignments
let IN = 0
   ! no input port inversions assigned
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
let IL = 0
   ! no safety input inversions assigned
let OL = 0
   ! No drive enable inversions.
let OP = 0
   ! Clear all digital outputs.


INIT_X:
let XMM = 4
   ! X axis motion mode = 4 = circular.
let XMO = 1
   ! X axis MOtor Enable = 1 = ON
wait 50
   ! Wait for settling, then proceed.
let XZP = 2000
   ! Set X primary encoder position at 2000.  Avoid a pre-move.


INIT_Y:
let YMM = 4
   ! Y axis motion mode = 4 = circular.
let YMO = 1
   ! Y axis MOtor Enable = 1 = ON
wait 50
   ! Wait for settling, then proceed.
let YZP = 0
   ! Zero Y primary encoder position.


CIRCLE:
disp "Prepare vector circular move.  Wait 1 second"
wait  1000
   ! Delay 1000 mSec
let VA = 30000
   ! X Vector Acceleration = 30,000 counts/sec^2
let VV = 2000
   ! X Vector Velocity = 2,000 counts/sec
let VD = 30000
   ! X Vector Deceleration = 30,000 counts/sec^2
let XCC = 0
   ! X axis Circle Center position.
let YCC = 0
   ! Y axis Circle Center position.
let RA =90000
   ! Rotation angle = 90000 (1/1000th) of a degree = 90 degrees.


MAIN:
disp "Starting 90 degree arc"
exec BXY
   ! Begin XY Vector move. (Immediate Mode command)
till ^ X_MOVE
   ! Wait until X axis profile is done.
disp "Arc Completed - wait one second"
```

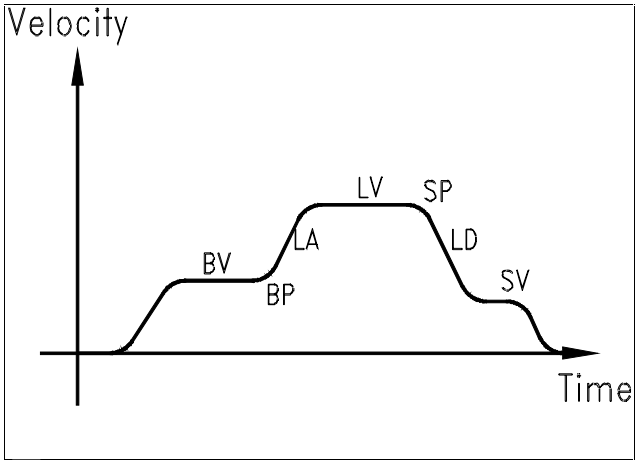

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
COMPLETE:
wait  1000
   ! Delay 1000 mSec
let XMO = 0
   ! X axis MOtor Enable = 0 = OFF
let YMO = 0
   ! X axis MOtor Enable = 0 = OFF
stop
   ! end of program
```

Mode 8 – Point-To-Point With Multiple Velocities

This motion mode was developed for dicing saw types of applications where it is necessary to enter the diced specimen and exit it in different speeds.

The motor moves toward the specified target point using three different velocities. Each velocity has its own associated position where it begins building.



Multiple velocities point-to-point

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Modes of Motion (continued)

Programming Example:

```
EXMM08_A:
rem FILENAME: MM08_Multi-Vel_v110.prg
rem VERSION: 1.10   DATE: 02-02-01   AUTHOR: Douglas A. Wright

! REQUIRES: ACS-Tech80 Classroom SB1292 motor fixture.

! PURPOSE: ACSPL Example Code.
!     Mode 8 Point to point, with multiple velocities.
!          Axis moves 5 revolutions during move.
!          "ACS Motion Controllers Software Guide" section 3.5.


PROMPT:
disp "X moves with two velocities, IN1 ON to abort, amp off when IN1
OFF."


INITVARS:
let OM = 0
   ! no dedicated IO reassignments
let PM = 0
   ! no PEG IO reassignments
let IN = 0
   ! no input port inversions assigned
let IL = 0
   ! no safety input inversions assigned
let OL = 0
   ! No drive enable inversions.
let OP = 0
   ! Clear all digital outputs.


INITAXIS:
let XMM = 8
   ! X axis motion mode = 8 = Multiple Velocities.
let XMO = 1
   ! X axis MOtor Enable = 1 = ON
wait 50
   ! Wait for settling, then proceed.
let XZP = 0
   ! Zero X primary encoder position.
let Xzp = 0
   ! Zero X auxiliary encoder (not required in this program)

let XLA = 30000
   ! X axis Linear Acceleration = 30,000 counts/sec^2
let XLV = 8000
   ! X axis Linear Velocity = 8,000 counts/sec
let XLD = 30000
   ! X axis Linear deceleration = 30,000 counts/sec^2
let XKD = 30000
   ! X axis Kill Acceleration = 30,000 counts/sec^2
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
REM  Special mode 8 parameters
let XBV = 3000
   ! X axis Begin Velocity.  The first, lower,  dwell velocity.
let XBP = 8000
   ! X axis Begin Position.
   ! Where to start the accel to Linear Velocity plateau.
let XSP = 16000
   ! X axis Search Point.
   ! Where to start the decel to Search Velocity slow move.
let XSV = 1000
   ! X axis Search Velocity.  Velocity used to look for the Sensor
Input.


MAIN:
let XRP = 20000
   ! X axis Relative Position, target distance
exec BX
   ! Begin X axis move. (Immediate Mode command)


CHECK_1:

if IN1 goto COMPLETE
   ! If IN1 on, exit program.
if X_MOVE goto CHECK_1
   ! loop until move done


COMPLETE:
if IN1 do
   ! If move is running and operator set IN1
  exec KX
   ! Kill X axis move.  (Immediate Mode command)
  till ^X_MOVE
   ! Wait until X axis profile stops.
  end
    else do
   ! If move finished
    wait 1000
   ! Delay 1000 mSec before disabling amp
    end


CHECK_2:
if IN1 goto CHECK_2
   ! Loop until IN1 set OFF by operator
let XMO = 0
   ! X axis MOtor Enable = 0 = OFF
stop
   ! End of program
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Modes of Motion (continued)

Mode 9 - Search For Contact

The Search for Contact mode was specifically developed for Pick & Place applications. In these applications, a part holder must descend at high speed reaching a predefined (programmable) point above a surface at low search velocity, continuing at that velocity until a sensor or contact is activated (signaling that the surface has been reached), and then continues for a predefined distance beyond the location of the contact and stops (applying force).

Programming Example:

```
MM09_X:
rem FILENAME: MM09_X__v110.prg
rem VERSION: 1.10   DATE: 02-02-01   AUTHOR: Douglas A. Wright

! REQUIRES: ACS-Tech80 Classroom SB1292 motor fixture.

! PURPOSE: ACSPL Example Code.   X axis search for contact.
! ACSPL Training example   Mode 9 End move with a search for contact,
IN1.
! continues to move 1/4 rev after contact.
! From "ACS Motion Controllers Software Guide" section 3.6

! USAGE: 1. Start move.
!        2. Wait until OUT2 has turned on, then off.
!        3. Turn IN1 on to stop in a quarter rev.


PROMPT:
disp "X moves with two velocities, IN1 ON to abort, amp off when IN1
OFF."

INITVARS:
let OM = 0
   ! Clear IO dedicated assignmnets.  Set OM again below with the MM9
data.
let PM = 0
   ! no PEG IO reassignments.  Also scleared in MM9 parameters below.
let IN = 0
   ! no input port inversions assigned
let IL = 0
   ! no safety input inversions assigned
let OL = 0
   ! No drive enable inversions.
let OP = 0
   ! Clear all digital outputs.


INITAXIS:
let XMM = 9
   ! X axis motion mode = 9 = Search for contact (registration).
let XMO = 1
   ! X axis MOtor Enable = 1 = ON
wait 50
   ! Wait for settling, then proceed.
let XZP = 0
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
   ! Zero X primary encoder position.
let Xzp = 0
   ! Zero X auxiliary encoder (not required in this program)

let XLA = 2000
   ! X axis Linear Acceleration = 2,000 counts/sec^2
   ! NOTE: Used for both first accel and decel to search velocity in
MM9.
   !       Linear Deceleration not used in MM9.
let XLV = 8000
   ! X axis Linear Velocity = 8,000 counts/sec
let XKD = 30000
   ! X axis Kill Acceleration = 30,000 counts/sec^2
   ! Used for final decel after sensor is found.
let XSF = 25
   ! X axis Smooth Factor. Jerk time in mSec. (Default)

REM  Special mode 9 parameters
let XVO = 2
   ! X axis search Velocity Output = 2 = OUT2.
   ! Output 2 will be on during decel to search vel. Must be 1-4 to
work!!!
let OM = 2
   ! 2d = 0002h = 0000000000000010b  out2 for VO
   ! Must set bit used for VO output in the Output Mask.
let PM = 0
   ! The ouput used for VO must be cleared in PEG mask.
let XSE = 1
   ! X axis SEnsor input source, look for sensor on input 1
let XSH = 5
   ! X axis Sensor High time, debounce in mSec
let XSV = 1000
   ! X axis Search Velocity.  Velocity used to look for the Sensor
Input.
let XSP = 48000
   ! X axis Search Position.  Point where Search Velocity is reached.
   ! 12 revs out, approximate 4 revs to accel/decel
let XSD = 100000
   ! X axis Search Distance.  Distance to look for Sensor before
default stop.
   ! 25 revs out
let XRD = 1000
   ! X axis Registration Distance.  Distance to keep moving beyond
Sensor.


MAIN:
exec BX
   ! Begin X axis move. (Immediate Mode command)
till X_MOTION = 3
   ! Wait until X axis motion profile is constant velocity (LV).
let HI = 1
   ! Turn OUT1 on to show vel stable.
till X_MOTION = 4
   ! Wait until X asis motion profile is decelerating.
let LO = 1
   ! Turn OUT1 off.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
till ^ X_MOVE
   ! Wait until X axis profile completes.

  if XSS > 0 do
   ! If sensor value changed from init value.
    disp "Sensor input found.  XSS position = " XSS
  end
    else do
      disp "Move completed without finding the Sensor.  XSS = " XSS
    end


COMPLETE:
wait  1000
   ! Delay 1000 mSec
let XMO = 0
   ! X axis Motor Enable = 0 = OFF
stop
   ! End of program
```

Mode 10 - Jogging (Continuous Velocity)
Upon a Begin command, the motor accelerates, using Linear Acceleration ($LA), to the Linear Velocity ($LV), and runs continuously at that speed until a new velocity command is issued, or an End/Kill command is entered. Both the velocity and acceleration can be changed while the motor is running. The direction is specified by the sign of the Linear Velocity parameter ($LV).

When a Kill or End command is issued, the motor decelerates using the Kill Deceleration ($KD) and Linear Acceleration parameters, respectively. While moving in this mode, it is possible to introduce an instantaneous change in the position (phase correction) by issuing a S$RP command. A large value of RP should be avoided.

Programming Example:

```
MM10_JOG:
rem FILENAME: MM10_Jog_v110.prg
rem VERSION: 1.10   DATE: 02-02-01   AUTHOR: Douglas A. Wright

! REQUIRES: ACS-Tech80 Classroom SB1292 motor fixture.

! PURPOSE: ACSPL Example Code.   Mode 10 Jog mode.  Move until IN1
turned on.
! From "ACS Motion Controllers Software Guide" section 3.7


PROMPT:
disp "Set new velocities from Terminal."
disp "Set IN1 ON to exit"


INITAXIS:
let XMM = 10
   ! X axis motion mode = point to point
let XMO = 1
   ! X axis MOtor Enable = 1 = ON
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
wait 50
   ! Wait for settling, then proceed.
let XZP = 0
   ! Zero X primary encoder position.


PROFILE:
let XLA = 30000
   ! X axis Linear Acceleration = 30,000 counts/sec^2
let XLV = 8000
   ! X axis Linear Velocity = 8,000 counts/sec


MAIN:
exec BX
   ! Begin X axis move. (Immediate Mode command)
till IN1
   ! Wait until IN1 is ON.  Jog continues.
exec KX
   ! Kill X axis move.  (Immediate Mode command)
till ^ X_MOVE
   ! Wait until X axis profile stops.


COMPLETE:
wait 1000
   ! Delay 1000 mSec
let XMO = 0
   ! X axis MOtor Enable = 0 = OFF
stop
   ! End of program
```

Mode 12 - Master-Slave

The Master-Slave Mode slaves the velocity and position of the motor under control to an external Master encoder and develops a profiled move that is super-imposed on top of that velocity.

The Master Slave mode is used in the following applications:

- Electronic Gearing
- Flying Shears
- Tension Control
- Coil winding
- High Accuracy Laser Plotting
- Web Processing
- High Speed Printing

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Modes of Motion (continued)

The main features of this mode are:

- Two master encoders and up to four slaves
- High resolution gear ratio - 1/8,000,000
- High speed (within 0.002 second), 'on the fly' changing of gear ratio
- Super-imposed smoothed trapezoidal profile move, phase correction
- 'On the fly' switching from Master-Slave Mode to non-slaved Velocity
- The Y axis can be controlled as a "standard' axis, while the X axis is slaved to it
- 'On the fly' index finding of both the Master and of the Slave
- The following Master encoder formats are supported:
  1. Quadrature pair
  2. UP/DOWN pair
  3. CLOCK/DIRECTION pair
  4. C1/C0 two adjacent bits of a binary up-down counter pair
- No position or time limits on the Master operation
- Unmatched following error between the Slave and the Master - down to ±1 count, independent of the velocity
- 'On the fly' super imposed registration move
- Full support by the ACSPL high-level language, for simple application implementation

Programming Example:

```
EXMM12_A:
rem FILENAME: MM12_Jog-Mode_Master_v110.prg
rem VERSION: 1.10   DATE: 02-02-01   AUTHOR: Douglas A. Wright

! REQUIRES: ACS-Tech80 Classroom SB1292 motor fixture.

! PURPOSE: ACSPL Example Code.  Mode 12 Master-Slave. Master turned by
hand.
! The Y axis encoder is plugged into the Y axis auxiliary encoder input
with a
! special 15 to 9 pin adaptor X axis is run as a slave of the Y aux
master encoder.
! The Y axis is run in jog mode as the master for X which will follow.
! You can play with the speed and accel of the Y master with immediate
! mode commnds through the terminal.


! USAGE:  The Y axis Master can can be turned by hand and the X axis
slave will follow.
!         Motion continues until input 1 is toggled


PROMPT:
disp "Plug Y axis into Y Aux Encoder input with adaptor"
disp "Turn the Y axis Master by hand and the X axis slave will follow"
disp "Toggle IN 1 to stop program"


INITVARS:
let OM = 0
   ! no dedicated IO reassignments
let PM = 0
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
   ! no PEG IO reassignments
let IN = 0
   ! no input port inversions assigned
let IL = 0
   ! no safety input inversions assigned
let OL = 0
   ! No drive enable inversions.
let OP = 0
   ! Clear all digital outputs.


INIT_X:
let XMM = 12
   ! X axis motion mode = 12 = Master - Slave.
let XMO = 1
   ! X axis MOtor Enable = 1 = ON
wait 50
   ! Wait for settling, then proceed.
let XZP = 0
   ! Zero X Slave primary encoder position.
let Xzp = 0
   ! Zero X Slave auxiliary encoder (not required in this program)
REM  Special mode 12 parameters
let XMA = 2
   ! X axis Master Axis = 2 = Y axis auxiliary encoder is the
   ! master encoder input for the X axis.


INIT_Y:
let YMM = 10
   ! Y axis motion mode = 10 = Jog mode.
let YMO = 1
   ! Y axis MOtor Enable = 1 = ON
wait 50
   ! Wait for settling, then proceed.
let YZP = 0
   ! Zero Y primary encoder position.
let Yzp = 0
   ! Zero Y auxiliary encoder.  MASTER ENCODER INPUT!  DO NOT FORGET!


PROF_Y:
   ! Set profile for master axis jog.
let YLV = 4000
   ! Y axis Linear Velocity = 4,000 counts/sec
let YLA = 4000
   ! Y axis Linear Acceleration = 4,000 counts/sec^2


MAIN:
exec BX
   ! Begin X axis move. Master-Slave mode on.(Immediate Mode command)
exec BY
   ! Begin Y axis move. Start Jogging master.(Immediate Mode command)
till IN1
   ! Wait until IN1 is ON.  Master-Slave keeps running.
exec KX
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

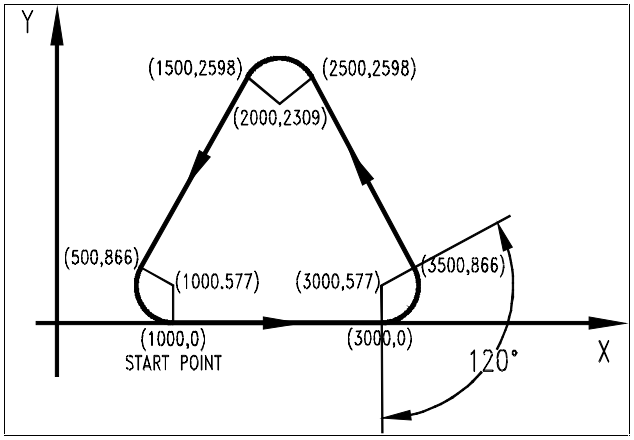
```
   ! Kill X axis move. Disable Master-Slave lock. (Immediate Mode
command)
exec KY
   ! Kill Y axis move. Disable Jog of master. (Immediate Mode command)


COMPLETE:
wait 1000
   ! Delay 1000 mSec
let XMO = 0
   ! X axis MOtor Enable = 0 = OFF
let YMO = 0
   ! Y axis MOtor Enable = 0 = OFF
stop
   ! End of program
```

Mode 15 - Continuous Path Following

In Continuous Path Mode, the X-Y axes follow a pre-defined path. The path is defined by an array of up to 32 mixed straight-line segments and arc segments. (Only XY moves are supported.)

The motion starts at the present position, accelerates to the specified vector velocity, keeps that velocity until the last segment, and then decelerates toward the end point of the path.



Continuous path

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Modes of Motion (continued)

Programming Example:

```
EXMM15_A:
rem Filename: MM15_Book_Example_v110.prg
rem VERSION: 1.10   DATE: 02-01-01   AUTHOR: Douglas A. Wright

! REQUIRES: ACS-Tech80 Classroom SB1292 motor fixture.

! PURPOSE: ACSPL Example Code.
!           Demonstrate MM15 Continuous Path Following.
!           Mixes lines and circles for multiaxis vectors.


INITVARS:
let OM = 0
   ! no dedicated IO reassignments
let PM = 0
   ! no PEG IO reassignments
let IN = 0
   ! no input port inversions assigned
let IL = 0
   ! no safety input inversions assigned
let OP = 0
   ! No drive enable inversion


INITAXIS:
let XMM = 15
   ! X axis motion mode = 15 = Continuous path following
let YMM = 15
   ! Y axis motion mode = 15 = Continuous path following
let XMO = 1
   ! X axis MOtor Enable = 1 = ON
wait 50
   ! Wait for settling, then proceed.
let XZP = 1000
   ! X axis primary encoder position = 1000.
let Xzp = 0
   ! X axis auxiliary encoder position = 0. (not required in this
program)

let YMO = 1
   ! Y axis MOtor Enable = 1 = ON
wait 50
   ! Wait for settling, then proceed.
let YZP = 0
   ! Y axis primary encoder position = 0.
let Yzp = 0
   ! Y axis auxiliary encoder position = 0.  (not required in this
program)


PROFILE:
let VA = 90000
   ! Vector Acceleration = 90,000 counts/sec^2
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
let VV = 2000
   ! Vector Velocity = 2,000 counts/sec
let XLD = 90000
   ! Vector Deceleration = 90,000 counts/sec^2
let VK = 90000
   ! Vector Kill Deceleration = 90,000 counts/sec^2
let XLI = 0
   ! X axis Lower Index, pointer for Target array
let XUI = 5
   ! X axis Upper Index, pointer for Target array


SEGMENTS:
exec ACSE0 L 3000 0
   ! "exec"    Immediate mode command.
   ! "ACSE0"   Load Common array element 0.
   ! "L"       Line segment.
   ! "3000"    X axis end point od this vector.
   ! "0"       Y axis end point of this vector.
exec ACSE1 C 3000 577 120000
   ! "exec"    Immediate mode command.
   ! "ACSE1"   Load Common array element 0.
   ! "C"       Arc Segment.
   ! "3000"    X axis position of circle center.
   ! "577"     Y axis position of circle center.
   ! "120000"  Rotation Angle in 1/1000 of a degree.  120 degrees.
exec ACSE2 L 2500 2598
   ! Line segment. etc
exec ACSE3 C 2000 2309 120000
   ! Arc segment. etc
exec ACSE4 L 500 866
   ! Line segment. etc
exec ACSE5 C 1000 577 120000
   ! Arc segment. etc


MAIN:
exec BXY
   ! Begin XY Vector move. (Immediate Mode command)

if ^ Y_MOVE do
  disp Y_STEP
  end

till ^ X_MOVE
   ! Wait until X axis profile is completed

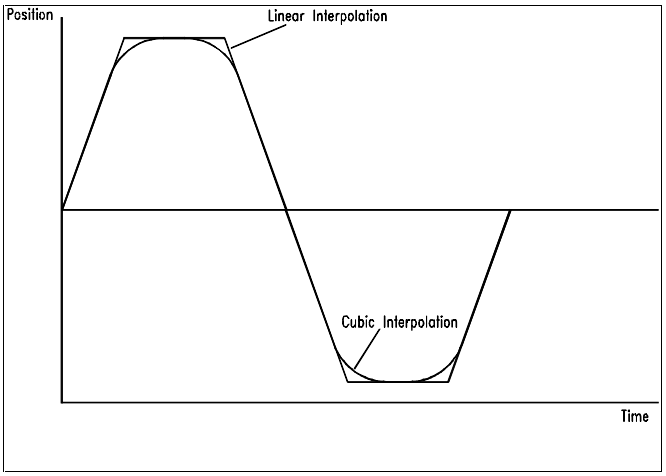

COMPLETE:
wait 1000
   ! Delay 1000 mSec
let XMO = 0
   ! X axis MOtor Enable = 0 = OFF
let YMO = 0
   ! Y axis MOtor Enable = 0 = OFF
stop
   ! End of program
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

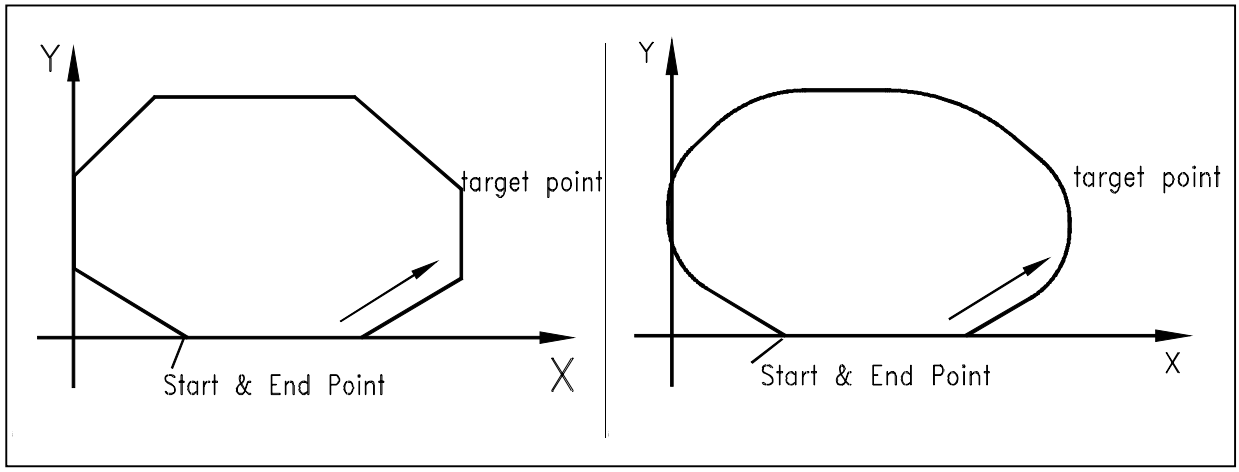
Modes of Motion (continued)

Mode 16 - Arbitrary Path Generation And Electronic Cam

The path is defined by a table of the coordinates of the desired motion. The table is stored in TX(i), TY(i), TZ(i), TT(i) - the same arrays that are used by Mode 3, data collection and PEG functions. The controller generates a path by piece-wise linear interpolation between the specified points, or by spline or cubic interpolation. The path can also be defined 'on the fly' while being executed. The path can be defined by absolute or relative coordinates, and by fixed or random time interval between adjacent points.



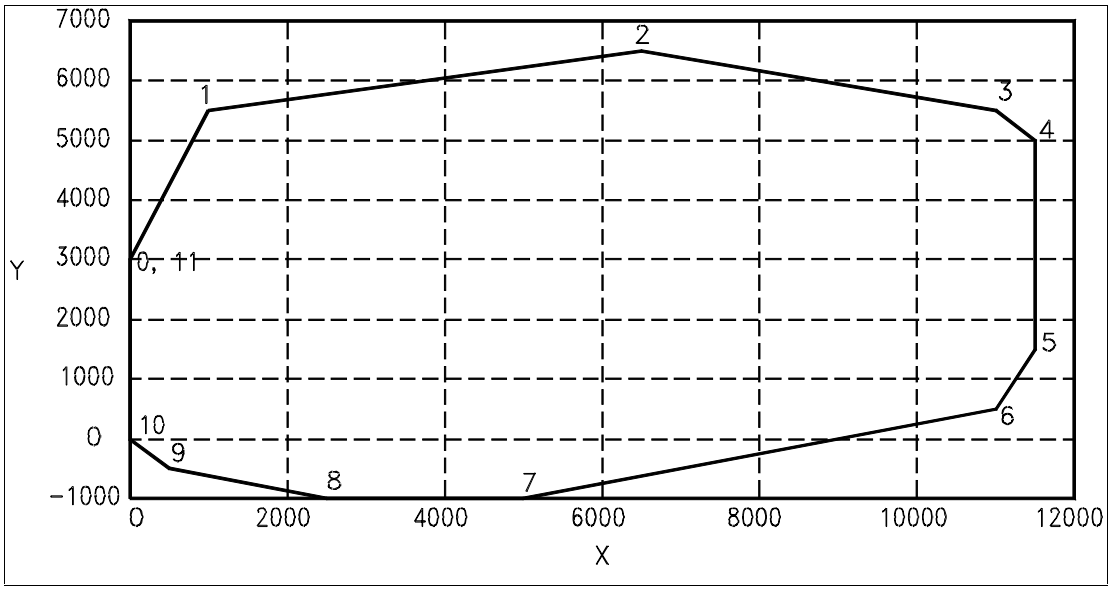
Linear vs. cubic interpolation



Linear vs. Cubic Interpolation in Profile Generation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Modes of Motion (continued)



An eleven segment arbitrary path

Programming Example:

```
MM16EXAC:
rem Filename: MM16_Arb_Exact-Spline_v110.prg
rem VERSION: 1.10   DATE: 02-01-01   AUTHOR: Douglas A. Wright

! REQUIRES: ACS-Tech80 Classroom SB1292 motor fixture.

! PURPOSE: ACSPL Example Code.
! ACSPL Training example   Mode 16 Arbitrary Path
! From "ACS Motion Controllers Software Guide" 1st Edition section 3.10
! Move through Target Arrays for X+Y
! XPG = 1 for Linear, 33 for Cubic, 97 for Exact Spline
! Change XPT to change feedrate.


PROMPT:
disp "Arbitrary Path - Exact Spline"
disp " Pre Move axis Y to starting position 0,3000"


INITAXES:
let XMM = 0
   ! X axis Motion Mode = 0 = point to point
let YMM = 0
   ! Y axis Motion Mode = 0 = point to point
let YLA = 30000
   ! Y axis Linear acceleration = 30,000 counts/sec^2
let YLV = 8000
   ! X axis Linear Velocity = 8,000 counts/sec
let YLD = 30000
   ! X axis Linear deceleration = 30,000 counts/sec^2
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
let XMO = 1
   ! X axis MOtor Enable = 1 = ON
let YMO = 1
   ! Y axis MOtor Enable = 1 = ON
wait    50
   ! Wait for settling.
let XZP = 0
   ! Zero X primary encoder position.
let YZP = 0
   ! Zero Y primary encoder position.
let Xzp = 0
   ! Zero X auxiliary encoder position.
let Yzp = 0
   ! Zero Y auxiliary encoder position.


PREMOVE:
let YAP = 3000
   ! X axis Absolute Position, target distance.  (Must follow MOtor
Enable).

exec BY
   ! Begin Y axis move. (Immediate Mode command)
till ^ Y_MOVE
   ! Wait until Y axis profile is completed.
disp "Delay 1 second"
wait  1000
   ! Delay 1000 mSec


ARB:
disp "Start Arbitrary Mode"
let XMM = 16
   ! X axis Motion Mode = 16 = Arbitrary Path following
let YMM = 16
   ! Y axis Motion Mode = 16 = Arbitrary Path following
   ! NOTE: All motion parameters for MM16 follow the lowest axis, here
that is X.
let XPG = 97
   ! X axis Path Generation mode, 97d = 61h = 01100001 = Exact Spline,
Absolute
let XPT = 1000
   ! X axis Path Time interval parameter = 1000 mSec per segment
let XLI = 0
   ! X axis Lower Index, start at beginning of the array
let XUI = 11
   ! X axis Upper Index, stop at the end of the data in the array
let XSQ = 0
   ! X axis SeQuence type = 0 = start at LI and goto UI
let XAO = 0
   ! X axis Array Offset = 0


ARRAY:
exec AXSE0 0 1000 6500 11000 11500 11500 11000 5000 2500 500 0 0
exec AYSE0 3000 5500 6500 5500 5000 1500 500 -1000 -1000 -500 0 3000
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
MAIN:
exec BXY
   ! Begin XY Arb move.   (Immediate Mode Command)
till ^ X_MOVE
   ! Wait until X axis profile is done.

disp "Arb move complete."
wait  1000
   ! Delay 1000 mSec


RETURN:
disp " Post Move axis Y back to home 0,0"
let XMM = 0
   ! X axis Motion Mode = 0 = point to point
let YMM = 0
   ! Y axis Motion Mode = 0 = point to point
let YLA = 30000
   ! Y axis Linear acceleration = 30,000 counts/sec^2
let YLV = 8000
   ! Y axis Linear Velocity = 8,000 counts/sec
let YLD = 30000
   ! Y axis Linear deceleration = 30,000 counts/sec^2
let YAP = 0
   ! X axis Absolute Position, target distance.
exec BY
   ! Begin Y axis move. (Immediate Mode command)
till ^ Y_MOVE
   ! Wait until Y axis profile is done.

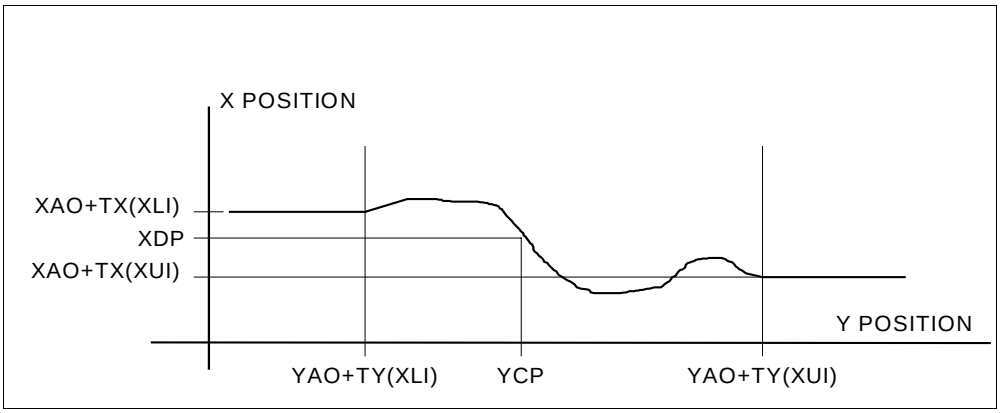

COMPLETE:
disp "Now at home, disable drives"
wait 1000
   ! Delay 1000 mSec
let XMO = 0
   ! X axis MOtor Enable = 0 = OFF
let YMO = 0
   ! Y axis MOtor Enable = 0 = OFF
stop
   ! End of program.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Modes of Motion (continued)

Electronic Cam

The position of an axis (the slave) is determined by the position of the other axis (the master). The exact relation between the slave and the master is defined by a table stored in the target arrays of these axes (master slave mode with a variable gear ratio). This type of Cam is referred to as 'Position Lock'. Another type of electronic Cam is a 'Time Lock'. This Cam is a private case of the arbitrary path for a single axis.



Cam – non-cyclic mode

Programming Example:

```
rem FILENAME: MM16_CAM-Cyclic_V110.prg
rem VERSION: 1.10   DATE: 02-01-01   AUTHOR: Douglas A. Wright

! PURPOSE: Blue line baseline functionality testing project.
!   Non-Cyclic CAM example, based on application by Jacob Sirkis. Blue
book section 3.10.2.
!   Axis X CAMs off of the master position of Y, which is turned by
hand.
!   The Y AXIS encoder is paralleled into both the Y Primary + Y Aux
encoder inputs.

! REQUIRES: ACS-Tech80 Classroom SB1292 motor fixture.

!CALC:
! SYSTEM: Motor max speed = 5000 RPM = 83.3 RPS
!         Encoder 1000 lines/rev = 4000 counts/rev
!         Max speed = 333,333 Counts/Second


INITVARS:
I1 = 0
   ! init index for input port data
let OM = 0
   ! no dedicated IO reassignments
let PM = 0
   ! no PEG IO reassignments
let IN = 0
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
   ! no input port inversions assigned
let IL = 0
   ! no safety input inversions assigned
let OP = 0
   ! Clear all outputs.


SLAVE:
   ! Load Slave target array
exec AXSE0 0 112 224 336 448 562 707 891 1112 1371 1669 2004 2377 2789
3238 3725
exec AXSE16 4236 4747 5258 5769 6280 6792 7303 7814 8325 8835 9317 9762
10169 10537 10868 11160
exec AXSE32 11415 11631 11810 11950 12064 12176 12288

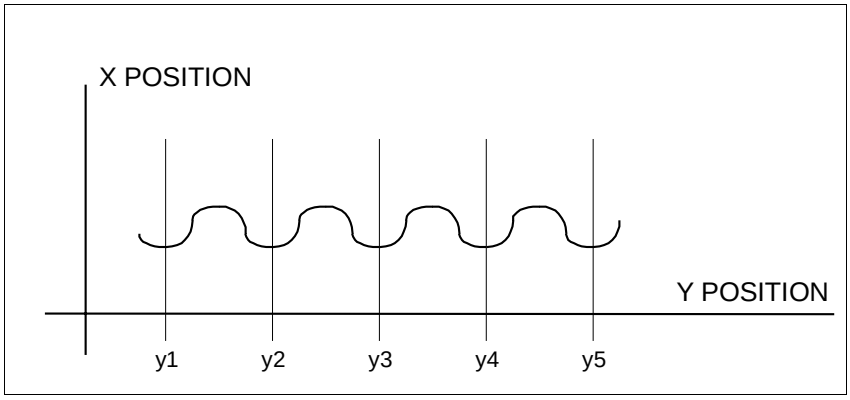

MASTER:
   ! Load Master target array
exec AYSE0 0 289 578 867 1156 1443 1730 2017 2304 2591 2878 3165 3452
3739 4026 4313
exec AYSE16 4600 4887 5174 5461 5748 6035 6322 6609 6896 7183 7470 7757
8044 8331 8618
exec AYSE31 8905 9192 9479 9766 10055 10344 10633 10923


INITAXES:
let XMM = 16
   ! X axis Motion Mode = 16 = Arb + CAM
let XMA = 2
   ! Set Master Axis for CAM mode = Y axis aux encoder input for SB1292
let XPG = 17
   ! Set X axis (Slave) Path Generation parameter
   ! 17d = 11H = 0001 0001 = , CAM, Non-Cyclic, Absolute
let XLI = 0
   ! Set X axis Lower Index for target array
let XUI = 38
   ! Set X axis Upper Index for target array
let XAO = 0
   ! Set X target array offset.
let YAO = 0
   ! Set Y target array offset.
let YMM = 0
   ! Set Y axis Motion Mode to joystick
let XMO = 1
   ! Turn X amplifier on.   DO NOT ENABLE Y, HAND TURN AXIS.  SAFETY!
let XZP = 0
   ! Zero the slave primary encoder
let YZP = 0
   ! Zero the master primary encoder
let Yzp = 0
   ! Zero the master auxillary encoder, CAM master input channel


exec BX
   ! Begin X axis move.  Must be separate from Y start.  BXY nogo.
till IN1
   ! Wait until IN1 is ON, then exit program.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
let XMO = 0
   ! X axis MOtor Enable = 0 = OFF
stop
   ! End of program
```



Cam - cyclic mode

Programming Example:

```
rem FILENAME: MM16_CAM-Cyclic_V110.prg
rem VERSION: 1.10   DATE: 02-01-01   AUTHOR: Douglas A. Wright

! PURPOSE: Blue line baseline functionality testing project.
!   Cyclic CAM example, based on application by Jacob Sirkis.  Blue
book section 3.10.2.
!   Axis X CAMs off of the master position of Y, which is running in
Joystick mode 21.
!   The Y AXIS encoder is paralleled into both the Y Primary + Y Aux
encoder inputs.

! REQUIRES: ACS-Tech80 Classroom SB1292 motor fixture.

!CALC:
! SYSTEM: Motor max speed = 5000 RPM = 83.3 RPS
!         Encoder 1000 lines/rev = 4000 counts/rev
!         Max speed = 333,333 Counts/Second


INITVARS:
I1 = 0
   ! init index for input port data
let OM = 0
   ! no dedicated IO reassignments
let PM = 0
   ! no PEG IO reassignments
let IN = 0
   ! no input port inversions assigned
let IL = 0
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
   ! no safety input inversions assigned
let OP = 0
   ! Clear all outputs.


SLAVE:
   ! Load Slave target array
exec AXSE0 0 112 224 336 448 562 707 891 1112 1371 1669 2004 2377 2789
3238 3725
exec AXSE16 4236 4747 5258 5769 6280 6792 7303 7814 8325 8835 9317 9762
10169 10537 10868 11160
exec AXSE32 11415 11631 11810 11950 12064 12176 12288


MASTER:
   ! Load Master target array
exec AYSE0 0 289 578 867 1156 1443 1730 2017 2304 2591 2878 3165 3452
3739 4026 4313
exec AYSE16 4600 4887 5174 5461 5748 6035 6322 6609 6896 7183 7470 7757
8044 8331 8618
exec AYSE31 8905 9192 9479 9766 10055 10344 10633 10923

INITAXES:
let XMM = 16
   ! X axis Motion Mode = 16 = Arb + CAM
let XMA = 2
   ! Set Master Axis for CAM mode = 2 = Y Axis aux encoder input for
SB1292
let XPG = 19
   ! Set X axis (Slave) Path Generation parameter
   ! 19d = 13H = 0001 0011 = , CAM, Cyclic, Absolute
let XLI = 0
   ! Set X axis Lower Index for target array
let XUI = 38
   ! Set X axis Upper Index for target array
let XAO = 0
   ! Set X target array offset.
let YAO = 0
   ! Set Y target array offset.
let YMM = 21
   ! Set Y axis Motion Mode to joystick
let YLV = 16000
   ! Set Y axis Linear Velocity, max vel,  AIN1 scales it
let XMO = 1
   ! Turn X amplifier on.
let YMO = 1
   ! Turn Y amplifier on.
let XZP = 0
   ! Zero the slave primary encoder
let YZP = 0
   ! Zero the master primary encoder
let Yzp = 0
   ! Zero the master auxillary encoder, CAM master input channel


exec BX
   ! Begin X axis move.  Must be separate from Y start.  BXY nogo.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

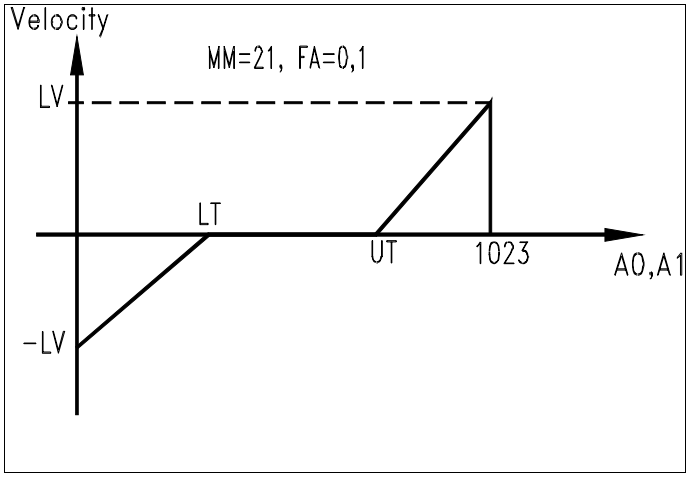
```
exec BY
   ! Begin Y axis move.
till IN1
   ! Wait until IN1 is ON, then exit program.


let XMO = 0
   ! X axis MOtor Enable = 0 = OFF
let YMO = 0
   ! Y axis MOtor Enable = 0 = OFF
stop
   ! End of program
```

Mode 21 - Manual Joystick Mode

In this mode the motor moves in a speed which is proportional to the voltage output of a joystick, which is connected to the proper analog input.

FA.1=0 , MM=21
This is the standard joystick motion mode. Every 1 millisecond, the analog input (A0 for X and A1 for Y) is sampled and a velocity command is generated according to the transfer function. Note that changing the sign of **LV** will change the direction of the move.

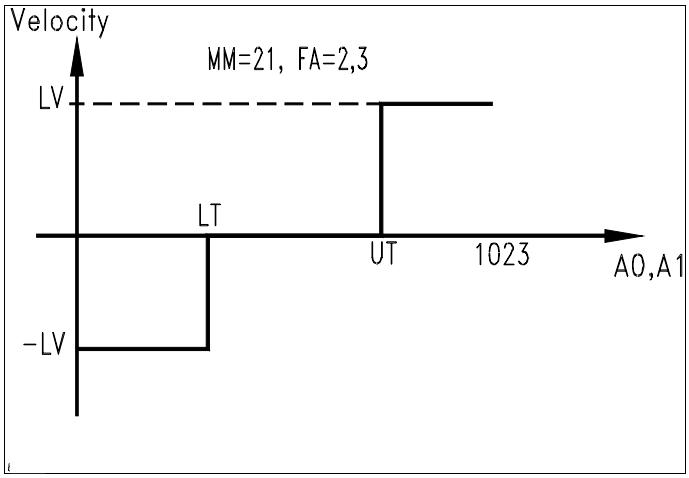


Joystick method of operation FA.1=0

Every 1 millisecond, the analog input is sampled. A velocity command is generated according to the transfer function. Note that changing the sign of **LV** will change the direction of the move.

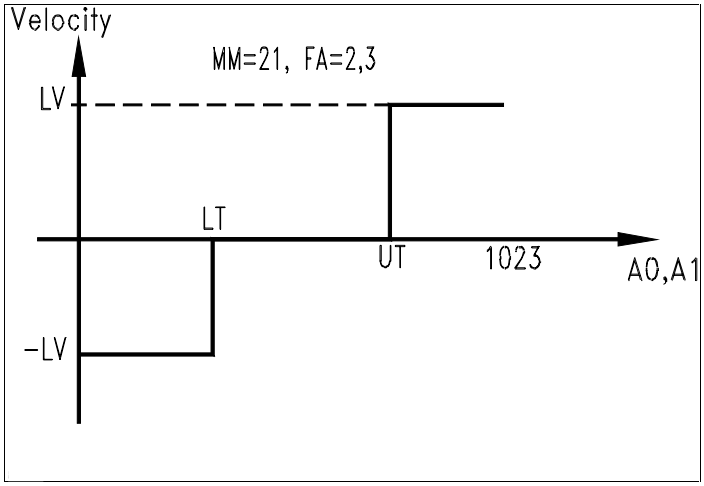
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Modes of Motion (continued)



Joystick method of operation FA.1=1

FA.1=1 , MM=21
Every 1 millisecond, the analog input is sampled. A velocity command is generated according to the transfer function. Note that changing the sign of **LV** will change the direction of the move.



Joystick method of operation FA.1=1

Programming Example:

```
EXMM21_A:
rem MM21_X_Joystick_v110.PRG

rem VERSION: 1.10   DATE: 02-01-01   AUTHOR: Douglas A. Wright
rem REQUIRES: Motor and IO Sim fixture.

! PURPOSE: Blue line baseline functionality testing project.
! ACSPL Training example   Mode 21 Manual joystick control.
! From "ACS Motion Controllers Software Guide" 1st Edition section 3.11
! Analog input from IO Simulation box simulates joystick.
! Analog voltage commands velocity.  X axis.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
! Negative polarity LV will turn the other direction.

! USAGE: Run program, watch motor move.

CALC:
! SYSTEM: Motor max speed = 5000 RPM = 83.3 RPS
!         Encoder 1000 lines/rev = 4000 counts/rev
!         Max speed = 333,333 Counts/Second


INITVARS:
let OM = 0
   ! no dedicated IO reassignments
let PM = 0
   ! no PEG IO reassignments
let IN = 0
   ! no input port inversions assigned
let IL = 0
   ! no safety input inversions assigned
let OL = 0
   ! No inversion of drive enable


INITAXIS:
   ! Use software commutation procedure, not hall senors
let XMM = 21
   ! X axis motion mode = 21 = Joystick, velocity follows voltage
input.
let XMO = 1
   ! Enable axis
wait 50
   ! Wait for settling, then proceed.
let XZP = 0
   ! Zero X primary encoder position.
let Xzp = 0
   ! Zero X auxiliary encoder (not required in this program)


PROFILE:
let XLV = 48000
   ! X axis Linear acceleration = 48,000 counts/sec^2.  6 RPS, 360 RPM
   ! NOTE: No acceleration parameter is used.  Motor can fault if
siganl is extreme.
let XLT = -80
   ! Set Lower Threshold to default
let XUT = 80
   ! Set Upper Threshold to default
let XMO = 1
   ! X axis MOtor Enable = 1 = ON
wait 50
   ! Let axis settle
let XZP = 0
   ! Zero X primary encoder position.
exec BX
   ! Begin X axis move. (Immediate Mode command)

disp "Turn TOP pot = Analog Input 0 and watch X motor"
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
disp " Toggle IO #1 up top stop program"

till IN1
   ! Wait until IN1 is ON
let XMO = 0
   ! X axis MOtor Enable = 0 = OFF
stop
   ! End of program.
```

Mode 22 - Position Tracking

Changing XAP or XRP causes an instantaneous change (within 5 milliseconds) in the desired command, without developing a profiled motion toward a new target position.

Programming Example:

```
XMM22_A:
rem Filename: MM22_Moving_till_IN2_v110.PRG
rem VERSION: 1.10   DATE: 02-01-01   AUTHOR: Douglas A. Wright
! REQUIRES: ACS-Tech80 Classroom SB1292 motor fixture.

! PURPOSE: ACSPL Example Code.
! ACSPL Training example   Mode 22 Postion Tracking. Continuous move.
! From "ACS Motion Controllers Software Guide" 1st Edition section 3.12
! A longer distance per "jump" will cause a system faults.

! USAGE: Programm runs until IN2 is set.


let XMM = 22
   ! X axis Motion Mode = 22 = Position Tracking
let XMO = 1
   ! X axis MOtor Enable = 1 = ON
wait 50
   ! Wait for settling, then proceed.
let XZP = 0
   ! Zero X primary encoder position.

disp "Start a series of Jumps.  Set IN2 ON to stop program"

V1 = 25
   ! Distance for each jump
exec BX
   ! Begin X axis move.  (Immediate Mode command.)
   ! Can now input new XAP's until EX given, all will be done


MOVE:
let XAP = V1
   ! X axis Absolute Position, target position.  (Must follow MOtor
Enable).
wait 10
   ! Delay 10 mSec.
V1 = V1 + 25
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
   ! Calulate next target position.
IF ^ IN2 goto MOVE
   ! Loop and keepo jumping until IN2 is ON.


disp "Jumps Completed"
exec EX
   ! End X move, close move.  Disable move. (Immediate Mode command.)
wait 100
   ! Delay 100 mSec
let XMO = 0
   ! X axis MOtor Enable = 0 = OFF
stop
   ! End of program
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

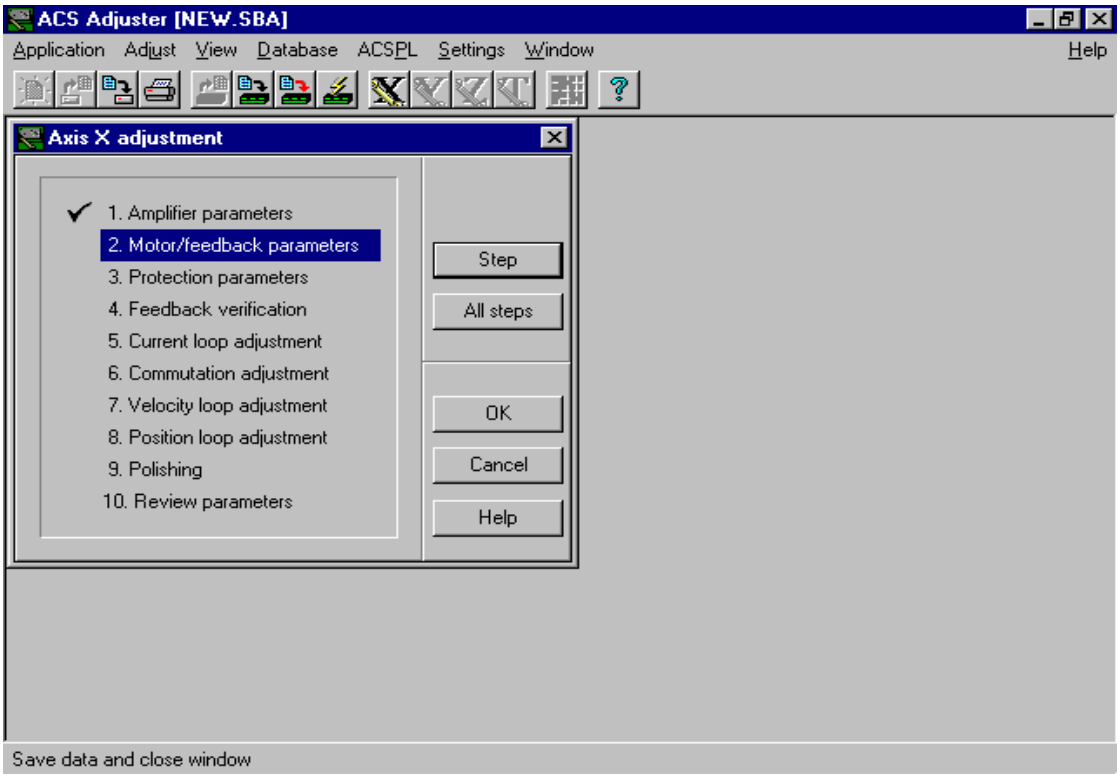
SB Series: DEVELOPMENT TOOLS

All of the SB Series products use the same Powerful Programming and Support Tools:

1) ACS Adjuster for Windows: Interactive tool for setting up, tuning, and viewing parameters. Includes soft oscilloscope.
2) ACS Debugger for Windows: Development environment for ACSPL (ACS Programming Language) applications.
3) ACS Saver / Loader for Windows: Tool for copying system setup and application data from one controller to another. Enables rapid setup of identical controller installations.
4) ACSLIB Libraries (optional): Comprehensive C, C++, and Visual Basic libraries for DOS and Windows 3.11/95/98/NT. Multithreading support under Windows NT

ACS Adjuster for Windows is a powerful tool that simplifies the setup and tuning process of the motion control system. An adjustment session involves the following steps:

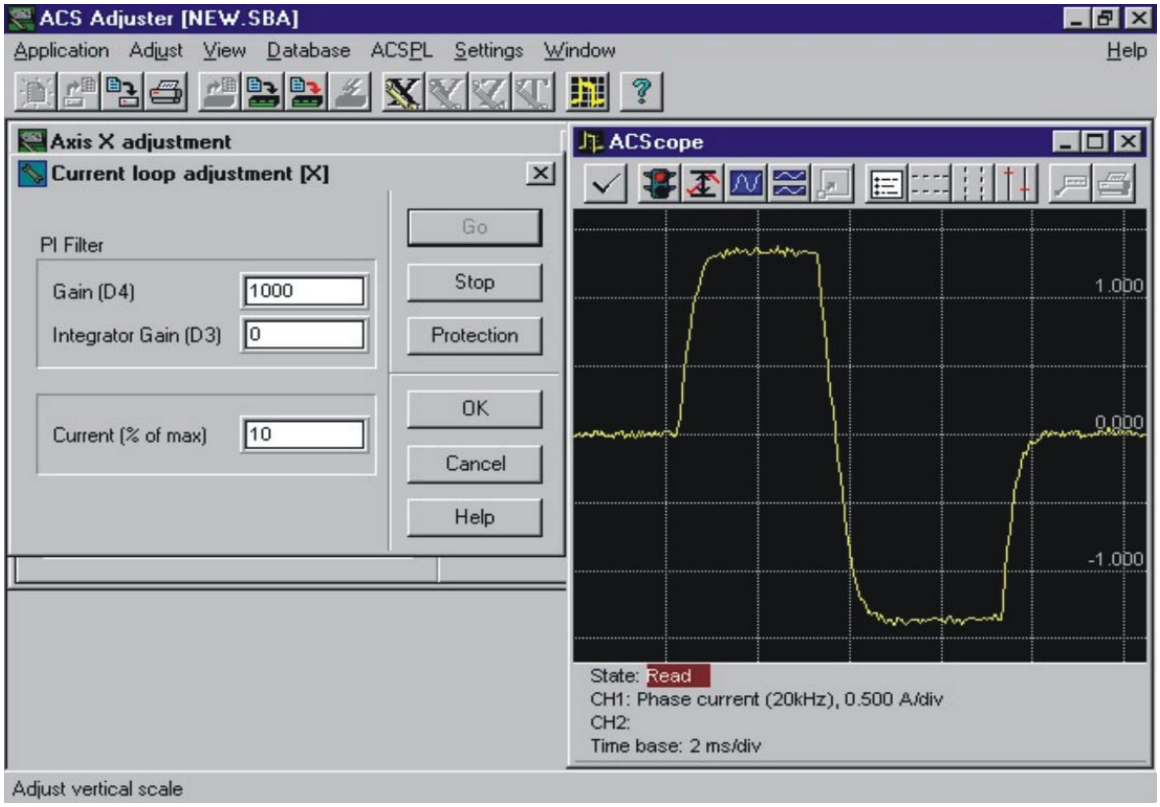
1 Amplifier parameters
2 Motor/feedback parameters
3 Protection parameters
4 Feedback verification
5 Current loop adjustment
6 Commutation adjustment
7 Velocity loop adjustment
8 Position loop adjustment
9 Polishing (feed-forward acceleration, slip constant, field current, etc.)
10 Review Parameters



Adjustment Screen

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Adjuster includes ACScope, a powerful software oscilloscope that enables you to monitor and print all real time variables, such as position error, velocity, phase current, etc.



ACScope used during current loop adjustment.

The ACS Adjuster allows viewing of:

1. The application database
2. The communications terminal window
3. Current Position
4. Secondary Position
5. Position Error
6. Safety Signals
7. Input and Output ports

Adjuster records all parameter values in an application database. An application file is a database that contains all the values entered or changed during an adjustment session. An adjustment session can be saved as a project, which can be used later to quickly set up other controllers for the same application (see ACS Saver/Loader for Windows).

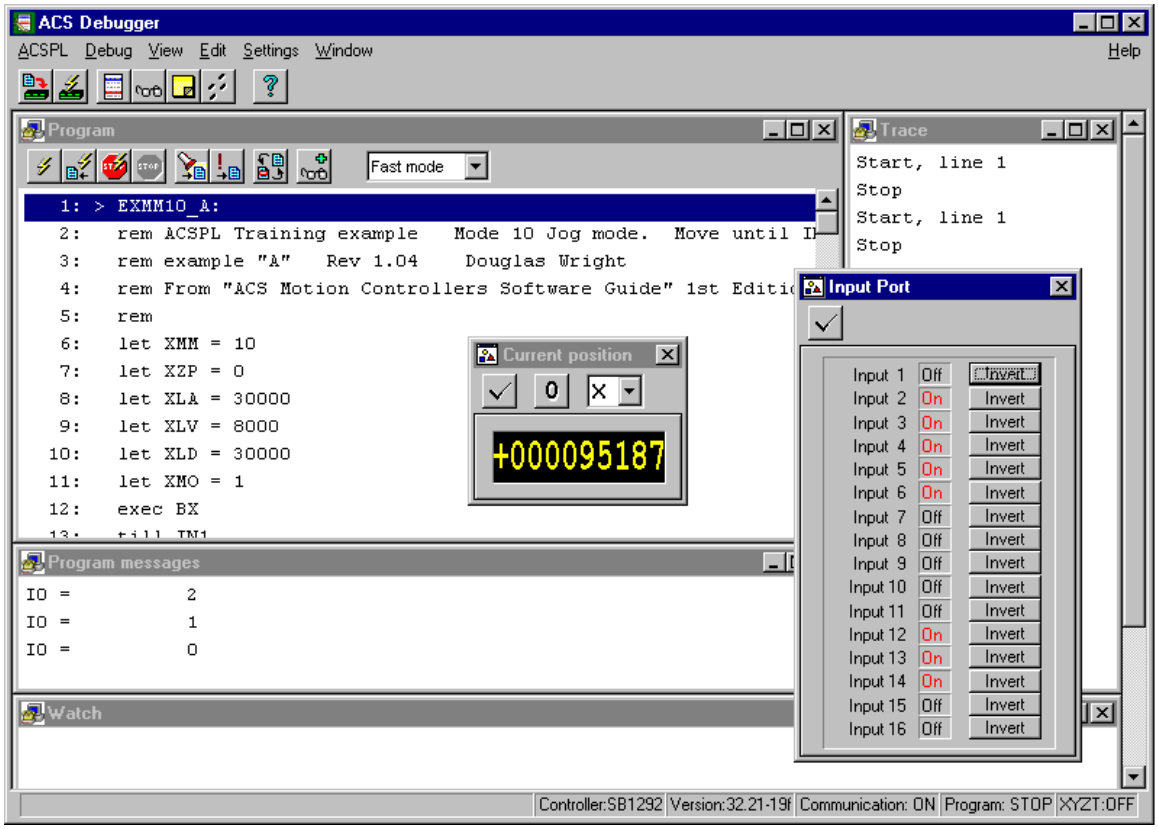
The adjuster allows you to compile and run simple programs; The ACS Debugger is used for more complex programs.

If support for the controller is needed, save the application and make a printout of ACScope. Send the application file and the printout to the ACS-Tech80 applications department. This will help the factory analyze the problem and offer a solution.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1) ACS Debugger for Windows is a powerful aid to programming in ASCPL.

- Normal and step-by-step execution
- Breakpoints
- Watches for standard user variables



ACS Debugger programming aid

2) ACS Saver & Loader for Windows is a utility designed to enable the user to save data stored in the controller's flash memory to file, and subsequently load the data from that file to a controller. This utility comprises two modules:

The ACS Saver module reads data stored in the controller's flash memory and saves it to file.

The ACS Loader module loads data stored in a file to a controller. It is possible to load a file to a different controller from which it was saved.

3) ACS C Library The ACS Application Library suite provides two C-language application programming interfaces (APIs) to allow end-user programs to communicate with ACS-Tech80 controllers. The two Dynamic Linked Libraries (DLLs) and their respective functions are:

ACSLIB0 - the 'lowest-level' functions used to interface to the ACS-Tech80 equipment; these are the raw function calls. An application programmer must add logic for error checking and state checking. Only programmers with intimate knowledge of the ACS-Tech80 controllers' language (ACSPL) should directly use this library.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ACSLIB2 - the 'highest-level' functions; they are built upon ACSLIB0 functions, but hide the details of the ACS-Tech80 controllers' language. ACSLIB2 operates in synchronous mode, in which a function call will only return after all necessary processing has completed. In addition, another set of functions commonly called ASCLIB1, is included in this library. The functions that comprise ACSLIB1 provide a higher degree of communications integrity for noisy operating environments.

Multi-Threaded Application written in C using Visual C++ for Windows NT

ACS-Tech80 can supply an example showing all the elements of a multi-threaded application written in C using Microsoft's Visual C++ compiler in the Windows NT environment. This application has been simplified for sample purposes, but shows how one might write a multithreaded process to control the motion portion of a much more complex software system.

In this example, the motion process consists of two threads. The main thread processes commands received from an upstream system process that has more information about the total application. This process communicates with the motion process via Inter-process communications using a named pipe. When motion is required, the higher-level process sends a request to the command thread of the motion process. In a full application, the command set between the processes would be much larger. For this sample, there are simply commands to set the velocity and acceleration, home the system, move the X axis, move the Y axis and move the X and Y axes together (although in a non-interpolated motion).

The second thread, the position thread, simply requests position for one axis every 250 milliseconds. In a full application, there might be functions that are deterministic in nature and require limiting the length of messages sent to and received from the controller. If this is the case, you would want to keep low priority messages such as requests for position as short as possible. With this in mind, the position thread is set up to request position for only one axis at a time. When position has been collected for both axes, the information is sent up to a higher-level process for display to the user.

There are two program files and two header files included in the project used to create the application. One set of files (multi-threaded.c and mtmotion.h) handles the motion functions, while the other (mt_ipc.c and mt_ipc.h) handles the inter-process communications with the other processes in the system.

As with the single threaded application, it should be noted that the Visual C++ environment must be modified to include the directories containing the ACS lib and include files. This can be done by adding the appropriate directories to the Directories tab in the Options window off the Tools menu in the Visual C++ IDE. In addition, the full path to the acslibm0.lib and acslibm2.lib files must be added to the Object/library modules line on the link tab of the Project settings. Finally, on the C/C++ tab of the Project settings, add /D "_MT" to the Project Options.

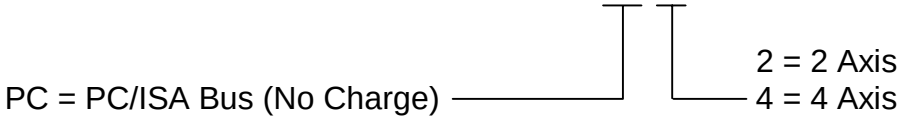
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SB Series: ORDERING MATRIX

SB214PC Ordering Matrix

- 1-4 Axis PC/ISA or Stand-alone Motion Controller
- PC Bus and Serial Communications
- ACSPL Programming Language

EXAMPLE: **SB214 PC 4**



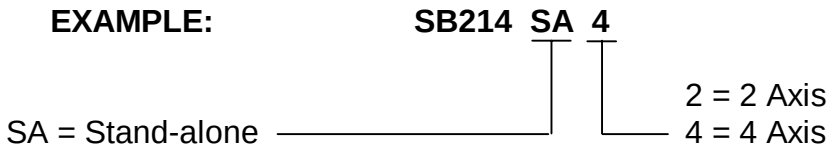
Standard SB214PC Includes:

- Motion Processor Unit (MPU) is Intel 80196
- One 16-bit RISC Servo Processors SB2500 per axis
- MPU Handles Trajectory Generation, Digital I/O, Communications, and Programs
- Slaved Servo Processor Executes Real Time Control Algorithms
- Dedicated Optically-Isolated Digital I/O for Limits, Faults & Enable
- Hardware Capture and Compare Registers
- 20-MHz Encoder Feedback (Encoder Loss Detection)
- Drive Command Signals for Analog Drives (+/- 10 VDC or Sinusoidal Commutation)
- Support Tools including ACS Adjuster & ACS Debugger
- Hardware and Software Manuals and Documentation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SB214SA Ordering Matrix

- 1-4 Axis Stand-alone Motion Controller
- Serial Communications
- ACSPL Programming Language



Note: SB214SA is supplied with an integral chassis for stand-alone communications.

Standard SB214SA Includes:

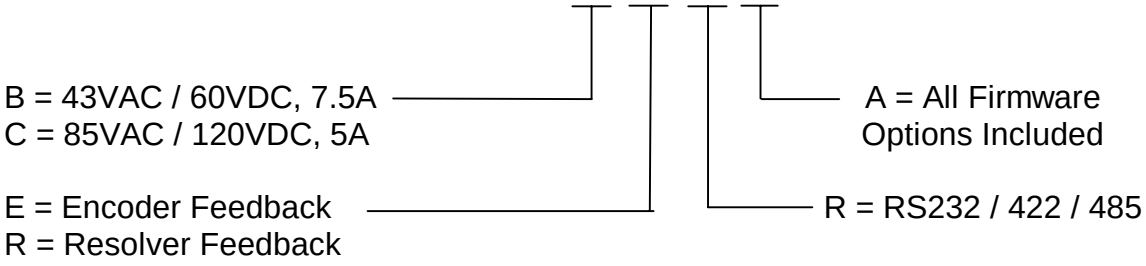
- Motion Processor Unit (MPU) is Intel 80196
- One 16-bit RISC Servo Processors SB2500 per axis
- MPU Handles Trajectory Generation, Digital I/O, Communications, and Programs
- Slaved Servo Processor Executes Real Time Control Algorithms
- Dedicated Optically-Isolated Digital I/O for Limits, Faults & Enable
- Hardware Capture and Compare Registers
- 20-MHz Encoder Feedback (Encoder Loss Detection)
- Drive Command Signals for Analog Drives (+/- 10 VDC or Sinusoidal Commutation)
- Support Tools including ACS Adjuster & ACS Debugger
- Hardware and Software Manuals and Documentation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SB1381 Ordering Matrix

- 1 Axis Stand-alone Control Module
- Brushed, Brushless and AC Induction Motors, 108-600W
- ACSPL Programming Language

EXAMPLE: **SB1381 – B – E – R – A**



Standard SB1381 Includes:

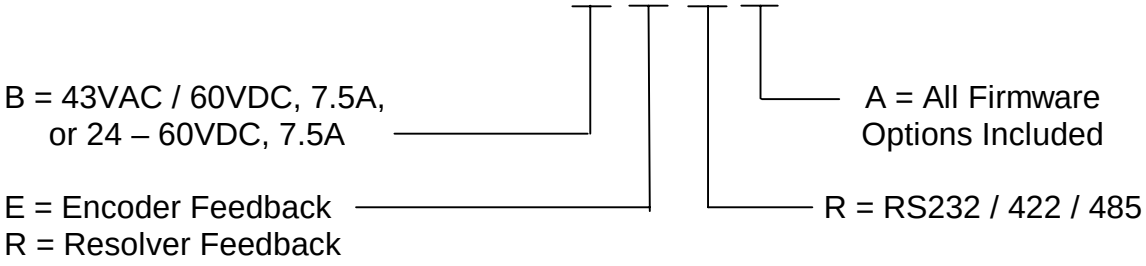
- Motion Processor Unit (MPU) is Intel 80196
- One 16-bit RISC Servo Processors SB2500
- MPU Handles Trajectory Generation, Digital I/O, Communications, and Programs
- Slaved Servo Processor Executes Real Time Control Algorithms
- Dedicated Optically-Isolated Digital I/O for Limits, Faults & Enable
- Hardware Capture and Compare Registers
- 20-MHz Encoder Feedback (Encoder Loss Detection)
- Drive Command Signals for PWM and Analog Drives
- Support Tools including ACS Adjuster & ACS Debugger
- Hardware and Software Manuals and Documentation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SB1292 Ordering Matrix

- 1-2 Axis Stand-alone Control Module
- Brushed, Brushless and AC Induction Motors, 500W
- ACSPL Programming Language

EXAMPLE: **SB1292 – B – E – R – A**



Standard SB1292 Includes:

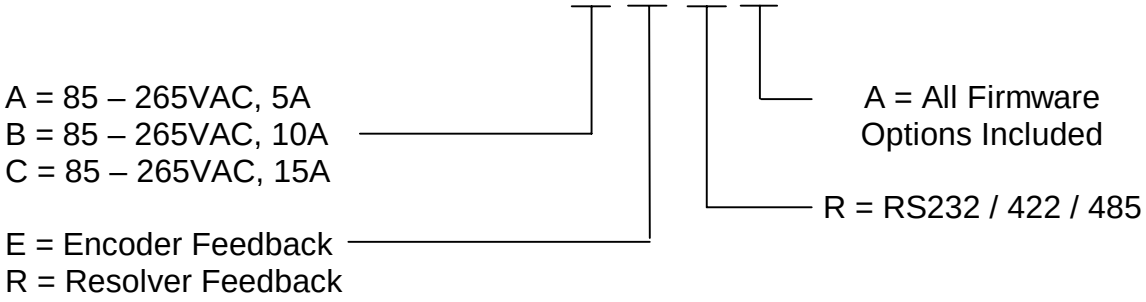
- Motion Processor Unit (MPU) is Intel 80196
- One 16-bit RISC Servo Processors SB2500 per axis
- MPU Handles Trajectory Generation, Digital I/O, Communications, and Programs
- Slaved Servo Processor Executes Real Time Control Algorithms
- Dedicated Optically-Isolated Digital I/O for Limits, Faults & Enable
- Hardware Capture and Compare Registers
- 20-MHz Encoder Feedback (Encoder Loss Detection)
- Drive Command Signals for PWM and Analog Drives
- Support Tools including ACS Adjuster & ACS Debugger
- Hardware and Software Manuals and Documentation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SB1391 Ordering Matrix

- 1 Axis Stand-alone Control Module
- Brushed, Brushless and AC Induction Motors, 1-5kW
- ACSPL Programming Language

EXAMPLE: **SB1391 – B – E – R – A**



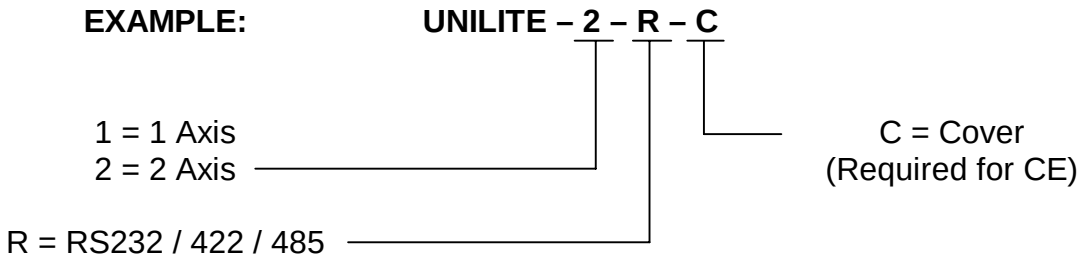
Standard SB1391 Includes:

- Motion Processor Unit (MPU) is Intel 80196
- One 16-bit RISC Servo Processors SB2500
- MPU Handles Trajectory Generation, Digital I/O, Communications, and Programs
- Slaved Servo Processor Executes Real Time Control Algorithms
- Dedicated Optically-Isolated Digital I/O for Limits, Faults & Enable
- Hardware Capture and Compare Registers
- 20-MHz Encoder Feedback (Encoder Loss Detection)
- Drive Command Signals for PWM and Analog Drives
- Support Tools including ACS Adjuster & ACS Debugger
- Hardware and Software Manuals and Documentation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Unilite Ordering Matrix

- 1-2 Axis Stand-alone Control Module
- Brushed Motors only, 100W
- ACSPL Programming Language



Note: Unilite does not support resolvers, current loop, dual loop, or position event generation.

Standard Unilite Includes:

- Motion Processor Unit (MPU) is Intel 80196
- One 16-bit RISC Servo Processors SB2500 per axis
- MPU Handles Trajectory Generation, Digital I/O, Communications, and Programs
- Slaved Servo Processor Executes Real Time Control Algorithms
- Dedicated Optically-Isolated Digital I/O for Limits, Faults & Enable
- Hardware Capture and Compare Registers
- 20-MHz Encoder Feedback (Encoder Loss Detection)
- Drive Command Signals for PWM and Analog Drives
- Support Tools including ACS Adjuster & ACS Debugger
- Hardware and Software Manuals and Documentation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

LIBRARY PROGRAMMABLE

INTRODUCTION

The Library Programmable Motion Controllers and Encoder Interface products were created with ACS-Tech80's commitment to open-architecture system design. With industry-standard language support, the programmer is free to use familiar programming tools. The Library Programmable encoder interface boards and multi-axis motion controllers are board level products for the PC/ISA and PC/104 architectures. A host PC is needed to operate the motion controllers and encoder interface boards. The Library Programmable controllers support servo, stepper, and commutation. The open architecture library supports drivers for DOS (16-bit), Win95/98/NT (32-bit DLL's); for C, C++, and Visual Basic. Motion specific function calls enhance standard languages, which enable ease of programming.

Library Programmable Motion Controller Products

Product	Controller Type	# of Axes	Discrete Digital I/O[1]	Encoder Count Rate[2]	Drive Command / Output Type
SPii PCI	PC/PCI Servo / Stepper Controller or Encoder Only Interface	4-8	8/8 (Opto-22 compatible)	20 MHz	±10VDC (16-bit)
5000	PC/ISA Bus Stepper Controller	1-3	5 in 3 out	N/A	Differential Pulse / Direction
5650A	PC/ISA Bus Servo Controller	1-4	8	1.0 MHz	±10VDC (16-bit) Analog Or 0-5VDC (10 bit) PWM
5651A	PC/ISA Bus Servo Controller	1-2	8	1.75 MHz	±10VDC (16-bit) 2 Phase Sinusoidal Analog
5950B	PC/104 Bus Servo Controller	4	4	1.0 MHz	±10VDC (16-bit) Analog
5950BP	PC/104 Servo Controller	4	4	1.0 MHz	0-5VDC (10-bit) PWM
5951B	PC/104 Bus Servo Controller	2	4	1.3 MHz	±10VDC (16-bit) 2 Phase Sinusoidal Analog
5954BP	PC/104 Bus Stepper Controller	4	4	1.25 MHz	Differential Pulse / Direction
5912	PC/104 Bus Encoder Interface	4	N/A	4.8 MHz	N/A
5312B	PC/ISA Bus Encoder Interface	4	N/A	4.8 MHz	N/A

1. All digital I/O is TTL compatible and bi-directional. The Model 5000 has optically isolated I/O. In addition to the General Purpose I/O all other controllers have dedicated digital I/O per axis:
- Home Input
- ± Limit Inputs
- Drive Fault Input
- Drive Enable Output (2 total on 5650)

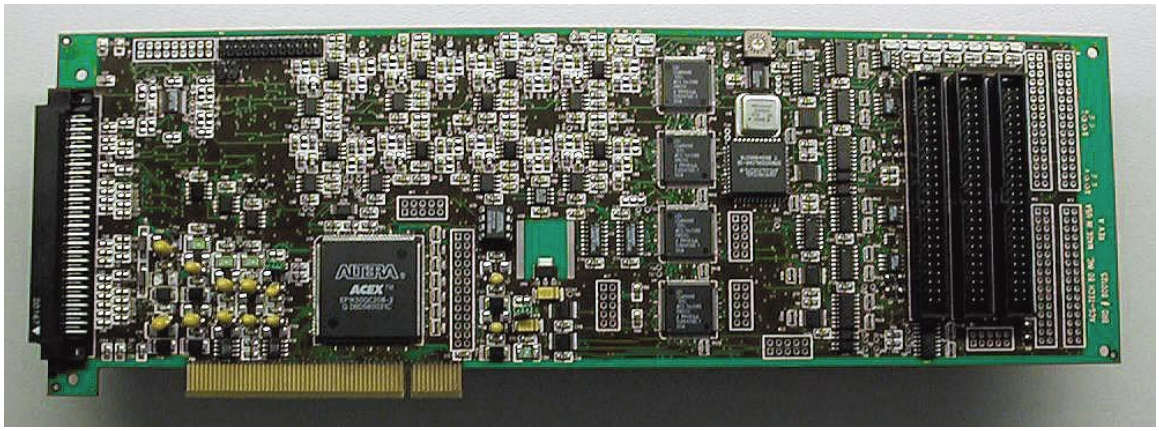
2. All encoder inputs accept square-wave Quadrature (Phase A, Phase B, and Index) incremental encoders. Inputs are configurable for Single-Ended or Differential.

ACS-Tech80's Library Programmable Motion Controllers and Encoder Interfaces are the most cost effective solution on the market today. They are the designer's choice for low-cost, host-dependent motion control. Their compact, low cost designs make them ideally suited for PC-

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

based OEM applications such as embedded machine control, packaging and semiconductor processing. They are best implemented in high volume applications where the engineer can invest resources to program the controllers entirely in C. Typically these controllers are used in high volume applications where the C programming environment and open architecture can be leveraged to integrate all needed elements in a cost effective system.

SPii PCI Multi-Axis Motion Controller



The ACS-Tech80 SPii PCI is a high-performance, low-cost motion controller for PC based control applications that require point-to-point and / or jogging motion capabilities.

The SPii controller module is a PCI bus circuit card that holds ACS-Tech80's proprietary ASIC Servo Processor, Mark II (SPii) and the necessary peripheral components. The SPii chips provide the profile generation and control functions.

The four-axis version has two SPii processors and the eight-axis model uses four. All the processors on the card share a common set of I/O connections.

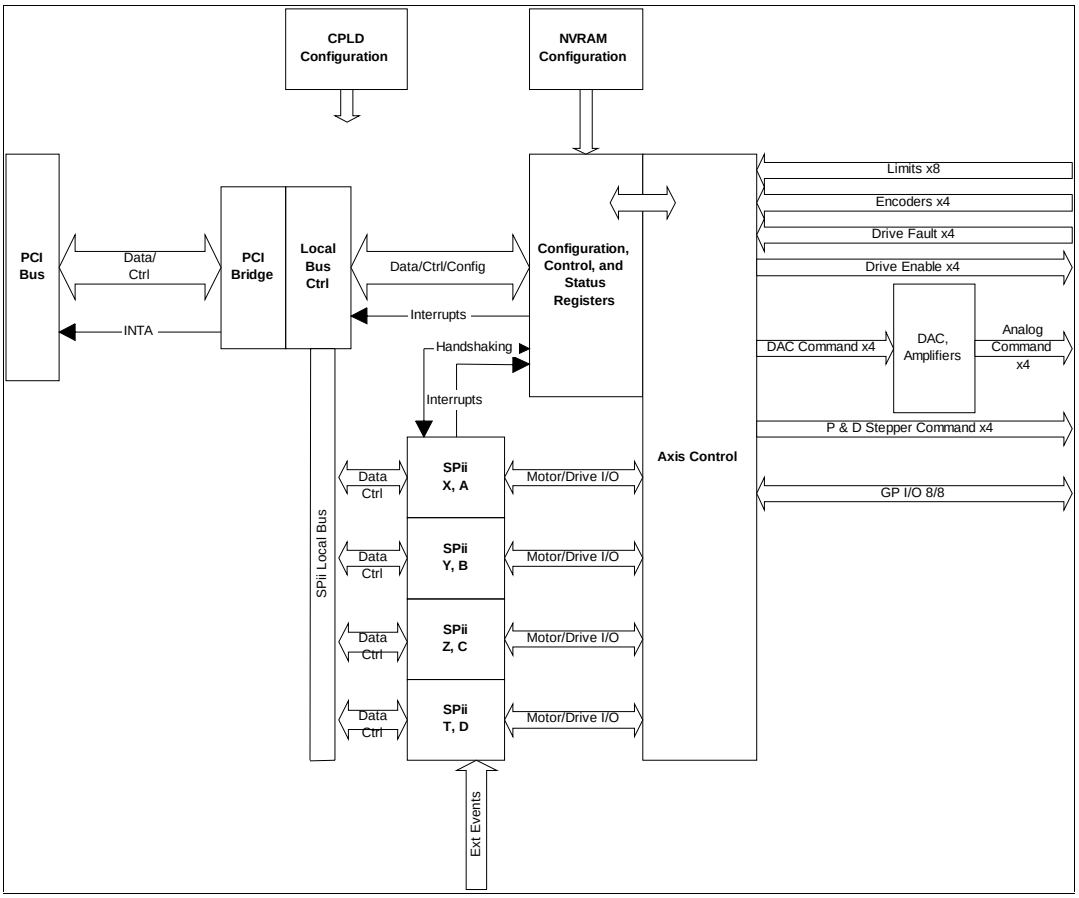
The SPii processors and the PC's CPU exchange data through dual-ported RAM on the SPii PCI card. This is random access memory, which the CPU and the SPii's share for more efficient communication. Data transfer on the card is accomplished through a 40 MHz local bus.

The SPii PCI is designed for use in a PC with a Pentium or Athlon class processor and communicates with the PC via a 33 MHz PCI bus through any standard PCI slot. The board interrupts are Plug-and-Play.

Once the SPii PCI controller is installed in the PC, no additional hardware is required. The module contains Digital to Analog Converters (DACs), which deliver analog signals directly to the amplifiers.

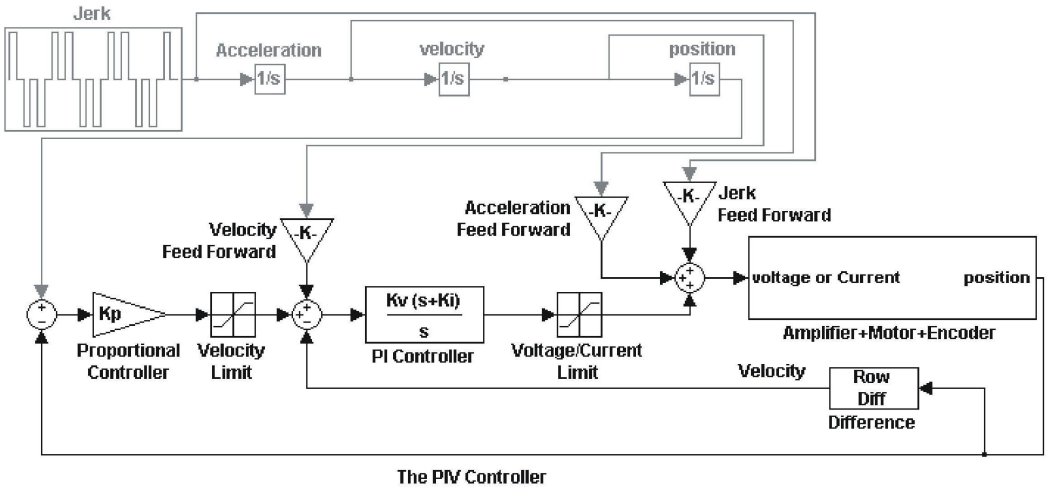
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Basic Operational Diagram of the SPii PCI Controller



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Basic Diagram of the SPii PCI PIV Control Loops



An advanced feature of the SPii PCI controller is the PIV filter and addition of a second order low pass filter. Most other controllers use a traditional PID filter. PIV filters are proven to easier to tune and more accurate than PID filters in servo based applications. In addition, the SPiiPCI includes a second order low pass filter. Other controllers use first order low pass filters. A second order low pass filter is an order of magnitude better at filtering out high frequency distrurbances. Thus, a more stable system is achieved.

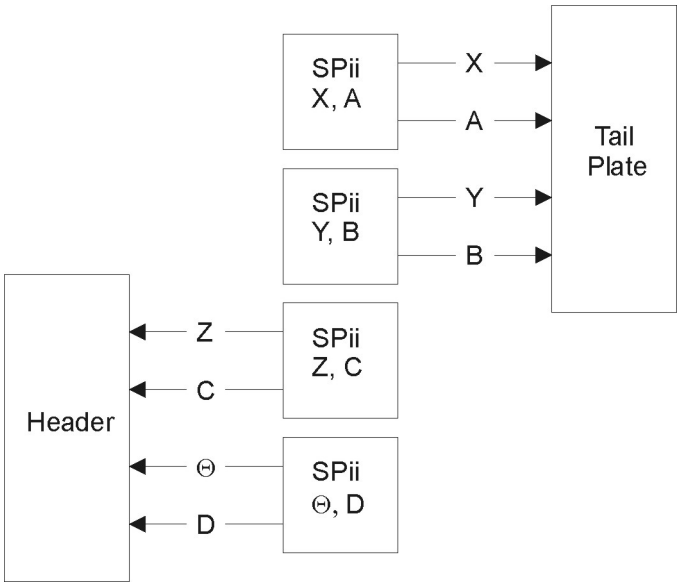
SPii Axis Designation and Support

Four SPii processors support eight (X, Y, Z, Θ, A, B, C, and D) axes of analog servo command or four (X, Y, Z, and Θ) axes of pulse and direction stepper, depending on configuration. The primary axes are designated X, Y, Z, and Θ, while the secondary axes are designated A, B, C, and D.

The SPii PCI can directly support one primary and one secondary axis per SPii processor. SPii X and Y axes connect through the board tail plate, and SPii Z and Θ axes are brought out to the header connector. From the header, ribbon cables connect to an auxiliary tail-plate, where a second axis connector is mounted.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Axis Options



Direct Port Selections (Primary & Secondary.)

Servomotor Drives

The axis combinations for servomotor drive support are:

- Primary only—X, Y, Z, and Θ
- Secondary only—A, B, C, and D
- Primaries and secondaries—X, Y, Z, Θ, A, B, C, and D

Stepper Motor Drives

Stepper motors are supported on primary axes only:

- Primary only—X, Y, Z, and Θ

Motor Drive Support

The SPii PCI supports analog command servo drives and pulse and direction stepper motor drives. Brushless motors are supported only if the motors can be commutated through the motor drive unit. *PWM servomotors are not supported.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

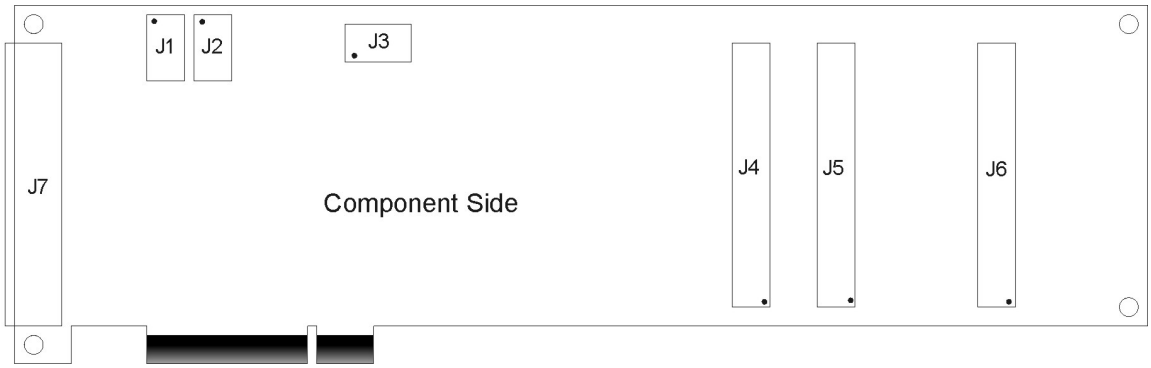
Inputs & Outputs

I / O for two primary and two secondary axes is handled by a 100-pin tail plate connector, with conductors in 25 pin groups for each axis.

Other external connections include—*two header sites* (supporting Z, C, Θ, and D), *general purpose I/O* (8 inputs and 8 outputs), and *External Board Sync* (1 in and 1 out).

To support axes Z, C, Θ, or D, an additional tail plate and slot are required. To support general purpose I/O, an additional tail plate and slot are required. For an 8-axis system using general purpose I/O, three tail plates and slots are required.

Support for input voltage rails other than +5V requires an additional interface board.



SPii PCI Connector Locations.

Inputs

- PCI Bus
- Encoders
- Drive Faults
- Limit Switches
- Mark Registration
- Emergency Stop
- General Purpose Inputs

Outputs

- Analog Motor Commands
- Drive Enable
- P & D Stepper Command
- General Purpose Outputs
- PEG (Position Event Generator, not used)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

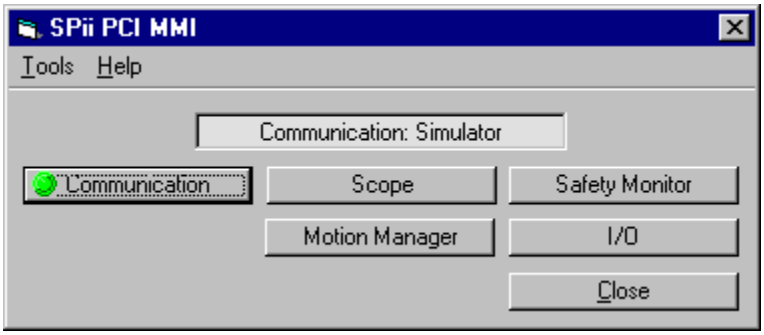
Software Tools

Tools Suite



The SPii PCI controller is controlled and programmed through a comprehensive set of software tools. The tools suite shares the look and feel of ACS-Tech80's other product lines.

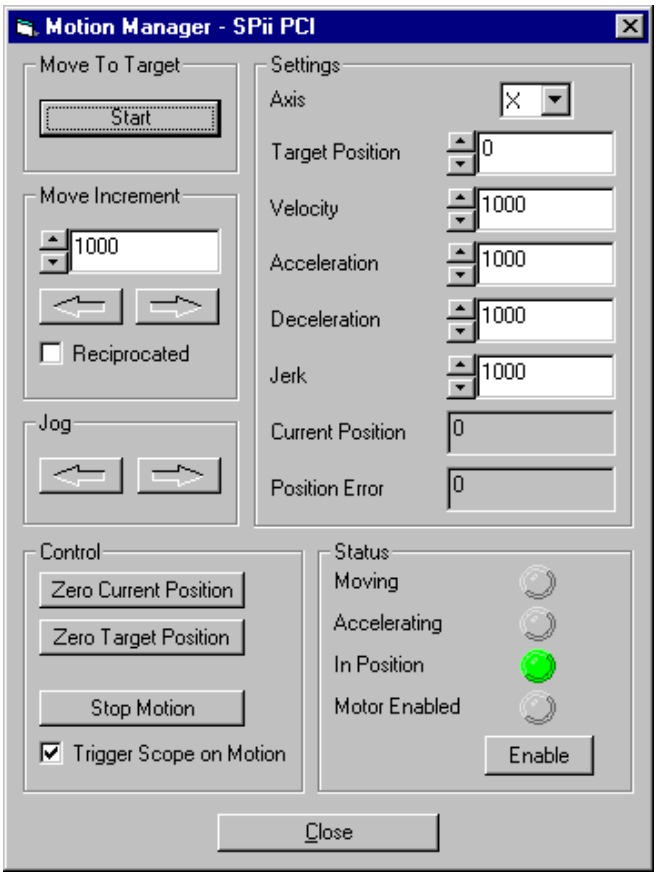
SPiiPlus Simulator



As with the SPiiPlus series, the SPii PCI MMI includes a Simulator, which allows applications to be designed and tested without a controller. The Simulator can be run on any PC and responds in exactly the same way as the SPii PCI controller card will.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Motion Manager

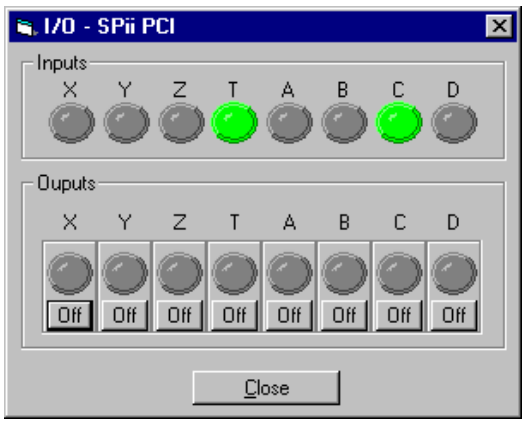


This window controls basic motion. It is normally used to manually move a given axis for testing or adjustment purposes.

The Motion Manager can move the axis in incremental, absolute, or jogging modes.

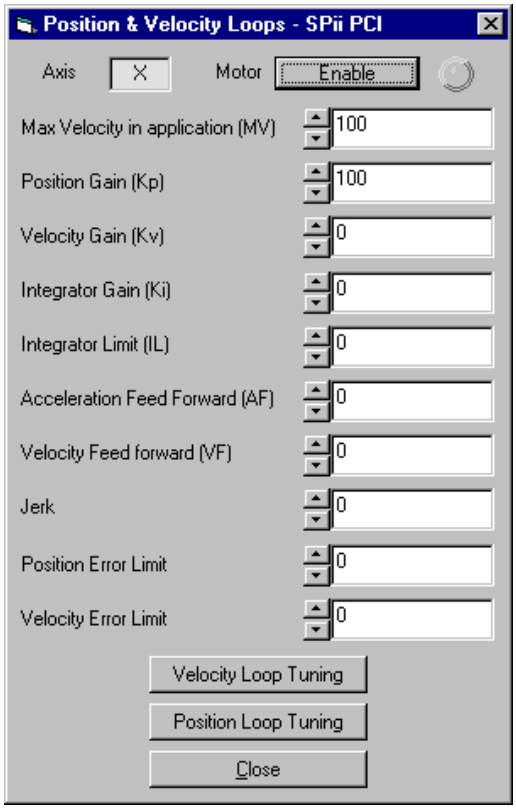
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

I / O Control



The I / O window monitors inputs and contains on / off switches for the outputs. While connected to the simulator the inputs randomly flash on and off.

Tuning



The MMI contains the Adjuster, a powerful tuning tool for tuning position and velocity loops. The Adjuster records all parameter values in an application database, which contains all the values entered or changed during an adjustment session. An adjustment session can be saved as a project, which can be used later to quickly set up other controllers for the same application.

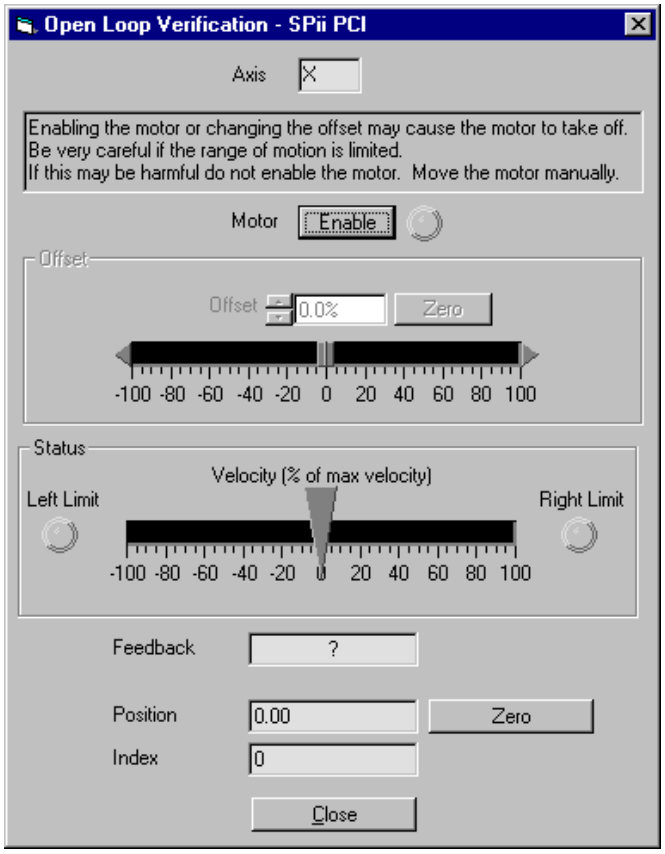
A typical adjustment session involves the following steps:

- Amplifier parameters
- Motor/feedback parameters
- Protection parameters
- Feedback verification
- Current loop adjustment
- Commutation adjustment
- Velocity loop adjustment
- Position loop adjustment
- Polishing (feed-forward acceleration, slip constant, field current, etc.)
- Review Parameters

The Adjuster links to the SPii PCI Scope, a software oscilloscope that enables you to monitor and print all real time variables, such as position error, velocity, phase current, etc.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Open Loop Verification

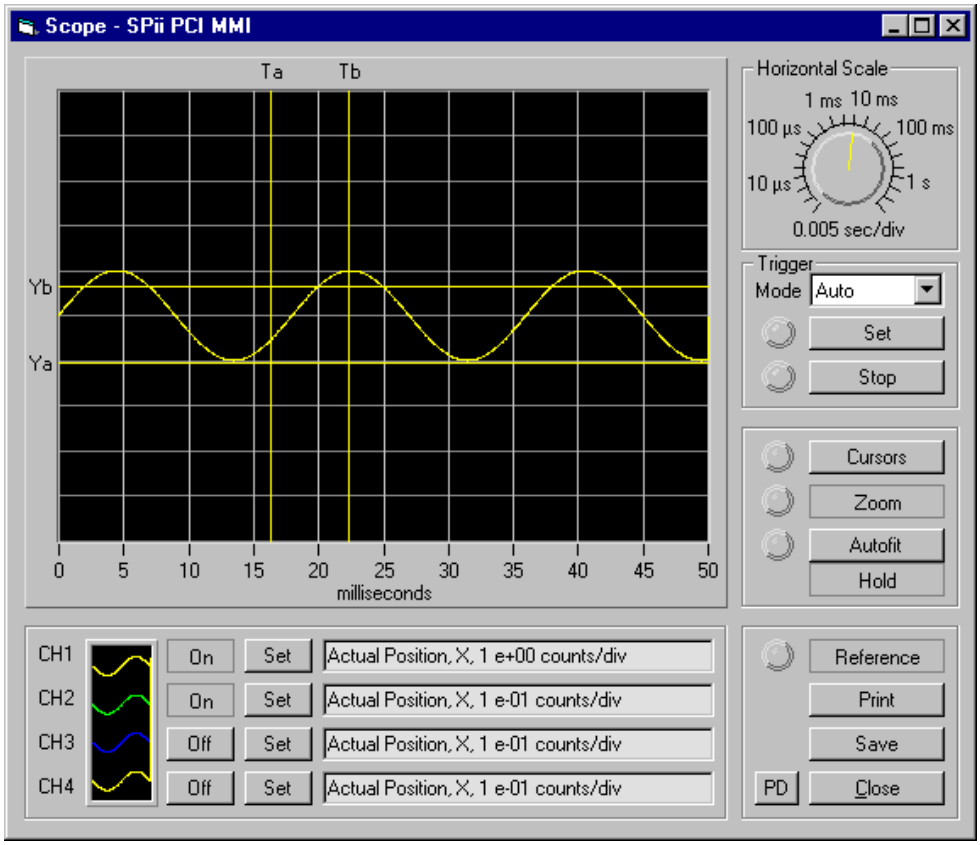


This window is a basic diagnostic tool to test the integrity of the wiring to the amplifiers and motors. This testing will also ensure each motor is correctly wired in-phase.

The Offset slider is moved to supply a positive or negative voltage to the analog outputs to run the motor. The Status meter displays the direction and relative speed of the motor. While the motor is running Feedback will display Positive or Negative depending on direction. Position from zero (in encoder counts) is also shown. The encoder value can be zeroed for reference.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SPii PCI Scope



Scope Signals

- Up to four signals simultaneously
- Predefined signals i.e., following error, feedback, velocity etc.
- Any standard or user variables
- Any internal variable of Servo Processor
- Any address in the controller memory
- One bit of any integer variable
- FFT signal analysis

Scope Triggering

- Normal, Auto, and Single triggering modes
- Triggering by rising or falling edge
- Adjustable trigger level

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SPii PCI Functions by Group

Communications Commands

spii_Open	The function opens a channel with the identified motion control card.
spii_Close	The function closes a channel with the identified motion control card.

Setup Commands

spii_InitializeAxis	The function initializes the given axis.
spii_SetScaleFactor	The function stores the scale factor for user-defined units.
spii_GetScaleFactor	The function retrieves the scale factor for user-defined units.
spii_SetStepRatio	The function stores the ratio of encoder counts to step pulses.
spii_GetStep Ratio	The function retrieves the ratio of encoder counts to step pulses.
spii_SetLoop	The function sets the state of the control loop.
spii_GetLoop	The function retrieves the state of the control loop.

Motion Control Commands

spii_Move	The function commands motion for the given axis.
spii_Halt	The function stops motion using the normal deceleration profile.
spii_Kill	The function stops motion using the kill deceleration profile.
spii_MoveGroup	The function commands motion for the given group of axes.
spii_HaltGroup	The function stops motion on all axes in the group using the normal deceleration profile.
spii_KillGroup	The function stops motion on all axes in the group using the kill deceleration profile.

Motion Event Commands

spii_SetCallback	The function registers the user defined callback function.
spii_GetEventStatus	The function returns the status of the events.
spii_SetActiveEvents	The function sets the state for each event.
spii_GetActiveEvents	The function gets the state for each event.

Buffered Motion Control Commands

spii_SetTriggers	The function sets the triggers that will cause an update of buffered data on the axis.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

spii_GetTriggers	The function retrieves the triggers.
spii_SetTriggerPos	The function sets where a position trigger will occur.
spii_GetTriggerPos	The function retrieves the triggers position value.
spii_SetTriggerTime	The function sets the time a trigger will occur at.
spii_GetTriggerTime	The function retrieves the trigger time.

Profile Control Commands

spii_SetAccel	The function stores the profile acceleration value.
spii_GetAccel	The function retrieves the acceleration value.
spii_SetTargetPos	The function stores the target (desired) position.
spii_GetTargetPos	The function retrieves the target (desired) position.
spii_SetDecel	The function stores the profile deceleration value.
spii_GetDecel	The function retrieves the deceleration value.
spii_SetJerk	The function stores the profile jerk value.
spii_GetJerk	The function retrieves the jerk value.
spii_SetVel	The function stores the profile velocity value.
spii_GetVel	The function retrieves the velocity value.
spii_SetKillDecel	The function stores the profile kill deceleration value.
spii_GetKillDecel	The function retrieves the kill deceleration value.
spii_GetRefPos	The function retrieves the current value of position from the profile generator.
spii_GetRefVel	The function retrieves the current value of velocity from the profile generator.
spii_GetRefAccel	The function retrieves the current value of acceleration from the profile generator.
spii_GetRefJerk	The function retrieves the current value of jerk from the profile generator.
spii_SetDwellTime	The function stores the dwell time for given motion.
spii_GetDwellTime	The function retrieves the dwell time for given motion.
spii_SetMoveMode	The function stores the mode of operation for a move.
spii_GetMoveMode	The function retrieves the mode of operation for a move.
spii_SetMoveType	The function stores the move type for a move.
spii_GetMoveType	The function retrieves the type for a move.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Controller Commands

spii_SetII	The function stores the velocity integration limit.
spii_GetII	The function retrieves the velocity integration limit.
spii_SetKaff	The function stores the acceleration feed forward gain.
spii_GetKaff	The function retrieves the acceleration feed forward gain.
spii_SetKi	The function stores the velocity integration gain.
spii_GetKi	The function retrieves the velocity integration gain.
spii_SetKp	The function stores the position loop gain.
spii_GetKp	The function retrieves the position loop gain.
spii_SetKv	The function stores the velocity loop gain.
spii_GetKv	The function retrieves the velocity loop gain.
spii_SetKvff	The function stores the velocity feed forward gain.
spii_GetKvff	The function retrieves the velocity feed forward gain.
spii_SetPosLoopLimit	The function stores the position loop output limit.
spii_GetPosLoopLimit	The function retrieves the position loop limit.
spii_SetCmdLimit	The function stores the controller output command limit.
spii_GetCmdLimit	The function retrieves the controller output command limit.
spii_SetCmdOffset	The function stores the offset for the controller output command.
spii_GetCmdOffset	The function retrieves the offset for the controller output command.
spii_SetSecondOrderFilter	The function stores the parameters for the second order filter.
spii_GetSecondOrderFilter	The function retrieves the constants for the second order filter.
spii_SetMaxPosError	The function stores the maximum allowed position error.
spii_GetMaxPosError	The function retrieves the maximum allowed position error.
spii_SetMaxVelError	The function stores the maximum allowed velocity error.
spii_GetMaxVelError	The function retrieves the maximum allowed velocity error.
spii_SetPosError	The function retrieves the current value of position error.
spii_GetVelError	The function retrieves the current value of velocity error.
spii_SetActual Pos	The function stores the actual position.
spii_GetActualPos	The function retrieves the current value of actual position.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

spii_GetActualVel	The function retrieves the current value of actual velocity.
spii_GetActualAccel	The function retrieves the current value of actual acceleration.
spii_GetActualJerk	The function retrieves the current value of actual jerk.

Safety Commands

spii_SetStopOnEstop	The function sets how a axis will respond to motion faults.
spii_GetStopOnEstop	The function retrieves the state for response to limit faults.
spii_ClearEstop	The function clears the indicated faults.
spii_EstopStatus	The function retrieves the state of the emergency stop.
spii_SetStopOnFault	The function sets how a axis will respond to motion faults.
spii_GetStopOnFault	The function retrieves the state for response to limit faults.
spii_ClearFault	The function clears the indicated faults.
spii_FaultStatus	The function retrieves the set of all faults that have occurred.

Axis I/O Commands

spii_SetEnable	The function allows the given axis to be enabled or disabled.
spii_GetEnable	The function retrieves the state of the axis.
spii_SetInputActiveState	The function retrieves the polarity of a given input.
spii_GetInputActiveState	The function retrieves the polarity of the inputs.
spii_GetInputState	The function retrieves the current state of the input.

Open Loop Control Commands

spii_SetDacLimit	The function stores the open loop DAC limit.
spii_GetDacLimit	The function retrieves the open loop DAC limit.
spii_SetDacOffset	The function stores the open loop DAC output offset.
spii_GetDacOffset	The function retrieves the open loop DAC output offset.
spii_SetDacValue	The function stores the open loop DAC output value.
spii_GetDacValue	The function retrieves the open loop DAC output.
spii_SetRateLimit	The function stores the open loop pulse generator rate limit.
spii_GetRateLimit	The function retrieves the open loop pulse generator rate limit.
spii_SetRateValue	The function stores the open loop pulse generator value.
spii_GetRateValue	The function retrieves the open loop pulse generator value.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Encoder Commands

spii_SetEncoderValue	The function sets the position of the encoder.
spii_GetEncoderValue	The function gets the position of the encoder.
spii_SetCapture	The function clears the indicated faults.
spii_GetCapturePosition	The function retrieves the set of all faults that have occurred.

General Purpose I/O Commands

spii_GetInputs	The function retrieves the current state of the general-purpose inputs.
spii_SetOutputs	The function sets the current state of the general-purpose outputs.
spii_GetOutputs	The function retrieves the current state of the general-purpose outputs.

Data Collection Commands

spii_SetupDataCollection	The function sets up data collection functionality.
spii_SetupDataCollectionIds	The function sets up the parameter to be collected.
spii_SetDataCollection	The function enables or disables data collection.
spii_GetDataCollection	The function returns the state of data collection.
spii_GetData	The function returns the collected data samples.

Error Commands

spii_GetLastError	The function returns the last error from a driver call.
spii_GetLastErrorString	The function returns a string that describes an error.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SPiiPCI General Specifications

Motion Control Profiles
Types: Servo, Stepper and Encoder Interface
Profiles: Trapzoidal or S-Curve

Position Control
Sampling Rate: 10kHz
Control Algorithms: P Type, automatic velocity feed-forward, acceleration feed-forward, integrator anti-reset windup, torque limits, DAC offset adjustment
Trajectory Calculation Rate: 10kHz on the SPii, New targets fed by host PC (system dependent)
Range: +/-140x10^{12} counts.
Accuracy: +/-1 encoder count
Registration Mark (Position capture):
Inputs: Four, 24-bit value
Trigger: Index or home signal
Delay: 0.1µs

Velocity Control
Sampling Rate: 10kHz.
Control Algorithm: PI Type, second order low pass filter
Range: Up to 8x10^{6} count/second
Resolution: 1 count/second
Velocity Accuracy: 0.01% - 0.5% (system-dependent)
Acceleration Range:
Up to 100x10^{6} count/second^{2}
Position and Velocity Feedback
Types: Incremental digital encoder
Incremental Encoder Type: Incremental, 3-channel, differential, RS-422.
A, A-, B, B-, I, I-, 0-5VDC, Maximum frequency of 20 MHz

Drive Interface
Servo/DC Motor Commands: Differential, +/-10V, 16-bit DAC resolution. Two per SPii processor.
Stepper Motor Commands: Four pairs of clock-dir, differential, RS422, One per SPii processor, 1MHz maximum frequency

Axis I/O
Limits: +/- limit inputs per axis
Home: home input per axis
Drive Enable Output: One per axis.
Type: Open collector/open emitter.
Collector Emitter Voltage Range: 5Vdc to 30Vdc
Output Current: 50mA
Propagation Delay: <1ms
Drive Fault Input: One per axis
Type: Source/sink
Input Voltage Range (External Supply): 5Vdc (+/-10%) or 24Vdc (+/-20%), automatic detection
Propagation delay: <1ms
Emergency Stop Input: One dedicated E-stop.
Type: source/sink, opto-isolated
Input Safety Voltage Range (External Supply): 5Vdc (+/-10%) or 24Vdc (+/-20%), automatic detection
Propagation Delay: <1ms

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

General Purpose I/O
Digital Inputs: Eight.
Type: TTL, Opto-22 compatible
Digital Outputs: Eight.
Type: TTL, Opto-22 compatible

Communication
PCI-Bus: 33MHz for host interface, 32 bits
Comprehensive interrupt support

Controller
Multi-Processor Architecture:
Host CPU required
Real-time controllers: 120MHz SPii Servo Processor
One SPii per two axes of servo/encoder control
One SPii per one axis of stepper control

Power Consumption
3.3VDC +/- 0.3VDC @ 2.2A, 5VDC +/-0.25VDC @ 120mA, +/-12V +/-0.6DVC @ 200mA

Environmental
Operating Temperature: 0 to 70 degrees C
Storage Temperature: -40 to 85 degrees C
Humidity: 90% RH, non-condensing

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Encoder Interfaces

Model 5912

1-4 axis PC/104 Encoder Interface

The Model 5912 provides decoding for up to four incremental quadrature encoders or pulse sources. For each encoder, Phase 0, Phase 90, and Index pulse inputs are available; these may be single-ended or differential and are conditioned by a four-stage digital filter and have sockets for terminator resister packs. The four-stage digital filter includes seven software-selectable sample clock frequencies, ranging up to 10 MHz. By selecting the lowest frequency compatible with the highest input rate expected, the noise immunity may be maximized. The board's conditioned inputs go through quadrature decoding and are then sent to the 24-bit counter. Pulse and Direction go directly to the counter. By issuing commands to a single global port, the independent counters may be commanded simultaneously. The 5912 meets current industry standards and easily plugs directly into a PC/104-compatible CPU or ACS-Tech 80's Model 5641 carrier board.

- PC/104 architecture
- Provides decoding for up to four incremental quadrature encoders or pulse
- 1.2 MHz quadrature count rate per axis (filtered)
- MHz quadrature count rate per axis (unfiltered)
- MHz pulse rate per axis lines (filtered)
- 15 MHz pulse rate per axis lines (unfiltered)
- 24-bit pre-settable counters, or chain 2 together for 48 bits
- Software configurable timer internal capture mode, or timestamp mode
- Digitally filtered inputs with selectable sampling rates
- Interrupts on index pulse, overflow/underflow, or compare value match
- Index actions: load counter from preset, capture
- Counter to output latch, gate counter on or off, reset counter
- Open architecture software library; drivers for DOS, Win3.1 and Win95 (16-bit); For C, C++, BASIC & Pascal.

Model 5312B

1-4 axis PC/ISA Encoder Interface

The Model 5312B quadrature encoder card is PC/ISA bus compatible. It provides inputs and decoding for up to four incremental quadrature encoders, depending on the model purchased. You may also use the card as a high-speed pulse counter (up/down and pulse/direction) for general counting applications.

- PC/ISA architecture
- The board provides decoding for up to four incremental quadrature encoders
- It can be programmed to react to index with reset, capture, pregate and gate
- The 5312B offers an affordable solution for applications needing positional feedback, digital readout, and event counting for high-input signal occurrences
- Inputs on the 5312B may be single-ended or differential and are conditioned by a four-phase digital filter
- Power (+5V) and grounding connections are provided for encoder use
- There are five jumper-selectable filter clock frequencies, ranging up to 10 MHz

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- The board applies the signal inputs to a 24-bit decoder/counter with programmable modes
- The counter may be used either for quadrature decoding or as a counter for pulse and direction or count up/down inputs
- Counters may be cascaded in 24-bit groups
- The counters are independent, but may be initialized simultaneously by issuing commands to a single port
- When an index pulse occurs, the count may be latched or a new count value may be loaded
- Open architecture software library; drivers for DOS (16-bit), Win95/98/NT (32-bit DLL's); for C, C++, and Visual Basic.

For each encoder circuit, Phase A (Phase 0), Phase B (Phase 90), and Index pulse inputs are provided. Jumper options on board allow you to configure the inputs as single-ended TTL or differential (the recommended connection method). Individual connectors for each encoder provide power (+5V) and ground for the encoder if needed.

You can also use the 5312B as pulse counter for up to four independent events, or you can cascade the counters to provide high-speed pulse counting over an extended count range. A 4-stage digital filter conditions inputs. The filter clock is one of five jumper-selectable sampling frequencies ranging up to 10 MHz. Selecting the lowest frequency compatible with the highest expected input rate will maximize noise immunity. The maximum input rate per phase in quadrature decode mode is approximately 333 kHz. The maximum input rate in count mode is approximately 1.25 MHz.

The conditioned inputs are applied to a 24-bit counter provided for each encoder. You can use the counters for quadrature decoding, pulse and direction input counting, or as a pulse input up/down counter. Count output is available for the PC Bus in binary or binary coded decimal (BCD) form. The count value may be latched on command, latched on an index pulse, or latched with a new count value when an index pulse occurs. The 5312B is capable of generating interrupts. Maskable interrupts may come from a valid index pulse, counter overflow/underflow, or on count value match with a preset compare value.

The Model 5312B can be configured for up to four independent 24-bit multimode counters, depending on the model purchased. All models are PC/XT/AT compatible. For applications requiring more than 24 bits of count range, the counters can be cascaded together to form various 24-bit counter configurations. For example, you can cascade counters to obtain one 48-bit counter, a 24-bit counter, and a 72-bit counter, or even one 96-bit counter.
Each counter is capable of numerous modes. Examples include:

- A/B quadrature with a maximum input frequency of .333 MHz
- Up/down count with a maximum input frequency of 1.25 MHz
- Count/direction with a maximum input frequency of 1.25 MHz
- Divide by n mode with a maximum input frequency of 1.25 MHz.

All three inputs to each counter (Phase A, Phase B, and Index) can be connected single-ended TTL or differential for greater noise immunity. The 5312B provides sockets in differential mode to allow you to easily insert termination resistors. The input lines do not need to be terminated, but for best noise immunity, terminate them at the characteristic impedance of the input line. Each input is tied through a buffer to an LED that is turned on when the respective input is LOW. Each input is digitally filtered (see section 2.2.10) using a sample clock rate that can be optimized for your signal input rate.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The index input can be used to generate an interrupt or any one of the following:

- Resetting the counter
- Enabling the gate for phase A and B inputs
- Transferring the 24-bit count value to the count latch

The counter can generate an interrupt on an overflow/underflow or on a compare match condition between the counter and the preset register. An 8259 Programmable Interrupt Controller (PIC) handles interrupts, and the PIC can be polled to determine the cause of the interrupt.

Description of Interrupt Control:
The 5312B uses an 8259A Programmable Interrupt Controller (PIC) to handle the interrupt sources on the board, and must be programmed prior to operation. Each axis can generate two interrupts: the first is a carry/borrow, signaling that an overflow or underflow has occurred, and the second interrupt is caused when a valid index pulse has occurred.

The PIC may only be used in a non-vectored mode (polled). When a board-level interrupt is generated, the PIC must be polled to determine which interrupt was triggered. The PIC may be disabled by removing jumper W8.

Only the features of the 8259A PIC that are used by 5312B are discussed below. For complete information on the 8259A PIC refer to the Peripheral Components Manual or the Microsystem Components Handbook, Volume 1, published by Intel.

Description of function calls:

te5312DisableIRQO Restore old interrupt vectors and disables PC IRQ lines.
te5312EnableIRQ(wBoardNum,Set up interrupt vector and enable PC IRQ line
sIRQLevel)on the bus.
te5312IndexAlertOff(sAxisNum)Disable index interrupt
te5312IndexAlertOn(wAxisNum)Enable index interrupt
te5312InitBoard(wBoardAddr,Initialize 5312 boardwNumAxes)
te5312InitEncoder(sAxisNum, sMCR, Initialize encoder.sICR, sOCCR, sQR)
te5312InitSw( ) Initialize software.
te5312InterruptHooks(*WrapAroundHook, *IndexHook) Define hooks to user functions
called on interrupts (not usable in BASIC).
te5312LoadCntr(sAxisNum, lValue)Load encoder counter.
te5312LoadPr(sAxisNum, lValue)Load encoder preset register.
te5312ReadCntr(sAxisNum)Read encoder counter.
te5312ReadOL(sAxisNum)Read encoder output latch.
te5312ReadSts(sAxisNum)Read encoder status.
te5312WrapAroundAlertOff(sAxisNum)Disable borrow/carry interrupt
te5312WrapAroundAlertOn(sAxisNum)Enable borrow/carry interrupt
te5312Write Cmd(sAxisNum, sCmd)Write encoder command

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Stepper Controller

Model 5000

1-3 axis stepper controller

The Model 5000 is a 1-3 axis stepper controller. The Model 5000 can control up to three independent step motor drivers, allowing for the performance of complex motion control profiling routines. Each axis provides five control inputs and three outputs. All inputs and outputs are opto-isolated to minimize noise at the card cage and protect the board from induced transients.

- Nippon pulse chipset
- PC/ISA architecture
- 1-3 Axes of stepper control
- 240 KHz step rate
- Programmable acceleration/deceleration
- Programmable velocity profiling
- OPTO-isolated inputs and outputs
- Each axis has dedicated control inputs – two stop limits, two deceleration inputs and one home input
- Each axis has outputs for pulse, direction and hold
- Four programming modes and eight operating modes
- Open Architecture Software Library; drivers for DOS, Win3.1 and Win95 (16-bit); For C, C++, BASIC & Pascal.

Multi-Axis Motion Controllers

The 5650A and 5950B library programmable motion controllers use a PMD RISC (Reduced Instruction Set Computer). The RISC handles the servo algorithms with PID and velocity feedforward filtering for all axes. It performs the intensive computational tasks required for high performance applications such as digital velocity, torque and position control. The dedicated RISC frees the host CPU for other tasks and permits motion control activities to run independently.

Systems that have high jerk values and steep acceleration curves can adjust phase lag at high speeds. The controller compensates for the motor's physical limitations and keeps the system in phase at all times. Its external sinusoidal commutation capabilities create stable systems under conditions that would cause standard amplified external commutation systems to fault. Sinusoidal commutation also significantly reduces torque ripple better than standard amplified external commutation.

Model 5650A

4 axis motion controller

- PC/ISA architecture
- Eight digital, general purpose TTL I/O can be configured as either inputs or outputs
- Each of the four axes has individual home and ±limits
- There are two drive enable outputs per board, 1& 2 are shared, 3 & 4 are shared

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- All limit inputs are capable of stopping motion in either direction
- A 1.0-MHz incremental encoder input provides position feedback
- The 5650 accepts single-ended or differential quadrature signals, and index pulses
- A high-speed (1 usec) **position capture feature** allows the latching of position values during a move in progress
- Built in **electronic gearing** allows for tight control between axes without taking up host processor time. The host specifies one parameter, the gear ratio. Applying the ratio to the current position of the master encoder generates the target position for the slaved axis
- Three forms of output are supported: ±10VDC DAC, PWM, and two Phase Sinusoidal commutation
- Open architecture software library; drivers for DOS (16-bit), Win95/98/NT (32-bit DLL's); for C, C++, and Visual Basic.

Model 5950B

The Model 5950B is industry's first full featured four axis motion controller that fits the PC/104 industry-required form factor.

- PC/104 architecture
- The heart of the Model 5950B is its PMD RISC with a custom ACS-Tech80 I/O chip
- Four digital, general purpose I/O can be configured as either inputs or outputs
- Each of the four axes has individual home, ±limit, and drive fault inputs
- Each axis has an individual drive enable output
- All limit inputs are capable of stopping motion in either direction
- The home input is configurable as either differential or single-ended
- The home and limit inputs contain a digital, software selectable debounce circuit
- A 1.0-MHz incremental encoder input provides position feedback
- The 5950 accepts single-ended or differential quadrature signals, and index pulses
- Built in **electronic gearing** allows for tight control between axes without taking up host processor time. The host specifies one parameter for the gear ratio. Applying the ratio to the current position of the master encoder generates the target position for the slaved axis.
- Four forms of output are supported: ±10VDC DAC, PWM, two phase sinusoidal commutation, and differential pulse/direction for stepper
- Open architecture software library; drivers for DOS (16-bit), Win95/98/NT (32-bit DLL's); for C, C++, and Visual Basic.

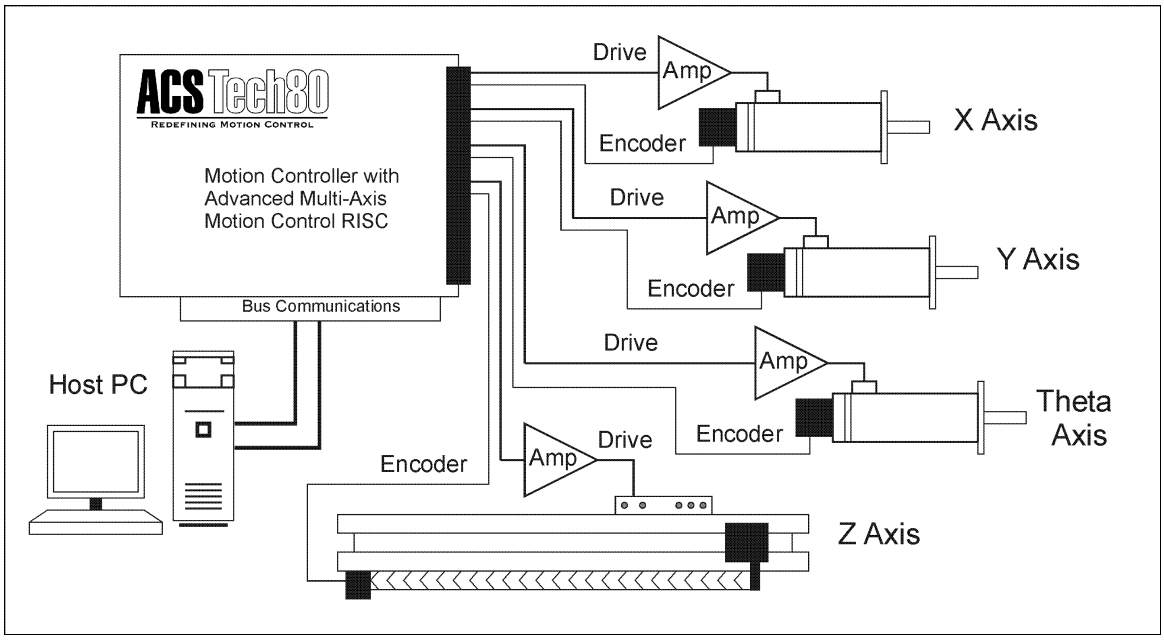
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

LIBRARY PROGRAMMABLE: FEATURES

Multi-Axis Motion Controllers

Models 5650A and 5950B Series

The model 5650 and 5950 ACS-Tech80 library programmable motion controllers use a PMD advanced multi-axis motion control RISC. The RISC (Reduced Instruction Set Computer) computation unit contains special on-board hardware such as a multiply instruction that makes it well-suited for the task of motion control. The RISC provides trajectory generation and closed-loop digital servo control for a large variety of servo and stepper motors. It uses incremental encoder position feedback signals, and a DAC, PWM, or Pulse/ Direction output drive. Axes can be programmed independently or synchronously for advanced multi-axis motion such as circular and continuous-path profiles.



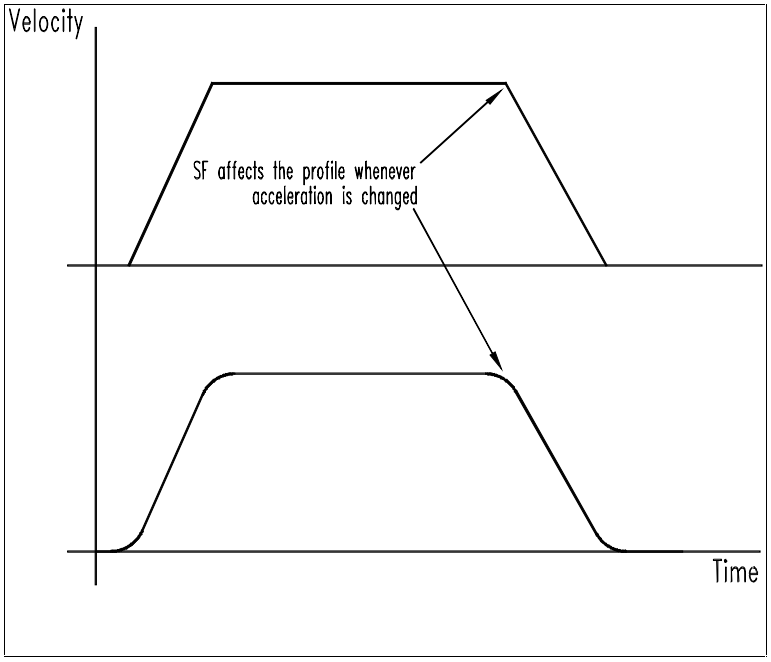
A host computer via an 8-bit, bi-directional port controls the RISC. Communications to and from the RISC consist of packet-oriented messages. A host interrupt line is provided so that the RISC can signal the host when special conditions such as encoder index pulse occur.

Each axis interfaces with a quadrature encoder with an optional index pulse. The incoming A and B quadrature data stream is digitally filtered, and then passed on to a high-speed up/down counter. Using the parallel-word interface, a direct binary-encoded position of up to 16 bits is read by the chipset. The position information is then used to maintain a 32-bit actual axis position counter. An instantaneous position for each axis can be captured upon an external trigger source; the host processor may then retrieve the captured value.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Depending on the type of motor controller used, the RISC provides servo loop closure, on-board commutation for brushless motors, and high-speed pulse and direction outputs for stepper motors. The actual position is combined with the target position generated by the trajectory profile generator to calculate a position error, which is passed through a PID filter. The resulting value is then output by the chipset to an external amplifier using either PWM or DAC signals.

Advanced features such as S-curve profile generation, bi-directional motion motion-travel limit switches, and separate home and index position capture signals.



S-curve (jerk) profile generation

Trajectory Profile Generation

The trajectory profile generator performs calculations to determine the target position, velocity, and acceleration at each servo loop update. These calculations take into account the current profile mode, as well as the current profile parameters set by the host.
Four trajectory modes are supported:

- S-curve point to point
- Trapezoidal point to point
- Velocity contouring
- Electronic gear

The profiles mode may be programmed independently for each axis. For example axis 1 may be in trapezoidal point-to-point while axis 2 is in S-curve point-to-point.

Generally, the axis should be at rest when switching profile modes. Under certain conditions, however, switching into certain modes “on the fly” is allowed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Trapezoidal point-to-point

- The program must start this mode when the axis is not already moving.
- Filter parameters can all be changed "on the fly."
- Trajectory complete status bit is set when the profile's destination desired position is reached.
- Stop and SmoothStop function normally.
- The program can change these profile parameters on the fly:
    - Target position
    - Velocity
- The program can change profile mode while moving to:
    - Velocity
    - Gearing

S-curve point to point

- The program must start this mode when the axis is not already moving.
- Filter parameters can all be changed on the fly.
- Trajectory Complete status bit is set when the profile's destination desired position is reached.
- Stop and SmoothStop function normally.
- The program cannot change the profile parameters "on the fly" under any conditions.
- The program can change profile mode while moving to:
    - Velocity
    - Gearing

Velocity contouring

- The program can start this mode while the axis is moving. Violent motion can occur if it is moving in the negative direction.
- Filter parameters can all be changed "on the fly".
- Trajectory complete status bit is set when command velocity (target) equals zero and desired velocity (instantaneous) is zero
- Stop and SmoothStop function normally.
- The program can change these profile parameters "on the fly":
    - Velocity
    - Acceleration
- The program can change profile mode while moving to:
    - Gearing

Electronic Gear

- The program can start this mode while the axis is moving.
- Filter parameters can all be changed "on the fly".
- NO Trajectory Complete status bit is used.
- Stop functions normally.
- SmoothStop does NOT work in gear mode.
- The program can change these profile parameters "on the fly":
    - Gear Ratio
- The program can change profile mode while moving to:
    - Velocity

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Mode Change Options and Adjustable Parameters

MODE: From/To	**S-Curve**	**Trapezoid**	**Velocity**	**Gear**
S-Curve	None	NA	Yes Positive direction only	Yes
Trapezoid	NA	Velocity Position	Yes Pos dir only	Yes
Velocity	NA	NA	Velocity Acceleration	Yes
Gear	NA	NA	Yes	Ratio

- NA = Not Allowed
- Only switch to velocity mode while moving in the positive direction!
- Switching to gear mode may cause problems.
- Left column is the starting mode.
- Right four columns is new mode.
- For no change cells (such as Trap to Trap) the variables that can be changed "on the fly" are shown.
- For Trapezoid mode, velocity and position can be changed "on the fly".

"On–The–Fly" and Stopped Mode Changing:

TRAPEZOID Mode
Stopped → Trapezoid
Trapezoid → Velocity
Trapezoid → Gear

S-CURVE Mode
Stopped → S-Curve
S-Curve → Velocity
S-Curve → Gear

VELOCITY Mode
Stopped → Velocity
Trapezoid → Velocity
S-Curve → Velocity
Gear → Velocity
Velocity → Gear

GEAR Mode
Stopped → Gear
Trapezoid → Gear
S-Curve → Gear
Velocity → Gear
Gear → Velocity

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Visual Basic Demo

ACS-Tech80 has created a suite of tools that allows programmers to access the 5950, 5951, and 5954 controller boards. The software will configure the board as well as monitor the operation of the controller from a programming standpoint. Windows are available to control and monitor the Status, Events, and Inputs of the 5950.

The Status window allows programmers to set and alter the distance, velocity, acceleration, etc. of a move and to observer the bit behavior during operation.

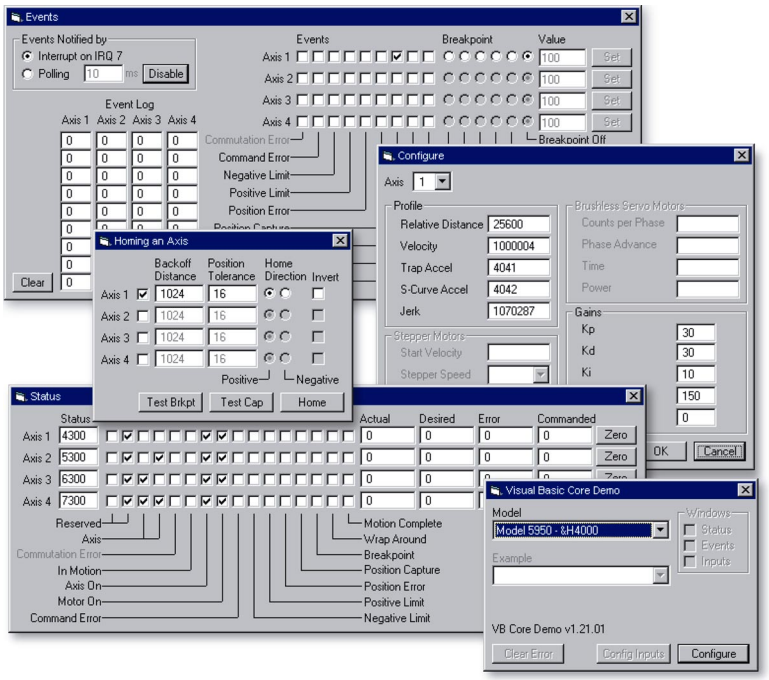
The Events window allows individual events to be handled by interrupts or polling. This window also offers the option of switching on a Breakpoint function, which can be defined as position (in number of encoder counts) or as time (in units of sample times.) The Event Log records a count of what events have occurred on which axis.

The Inputs window is a read-only display showing the state of the Limits, Drive Faults, Encoder inputs, etc. for the 5950, 5951, 5954.

The Visual Basic Demo includes three example programs:

- Making Moves – Allows the user to move any combination of axes in any of the allowable modes. The mode and some parameters can be changed on the fly.
- Homing an Axis – Features routines for homing servo and stepper motors. Direction, Position Tolerance, and Backoff Distance are all settable.
- Limit Recovery – Allows recovery from a fault position by moving a selected axis until it hits a limit switch and then zeros the axis.

The demo software can either be run from the Visual Basic 5.0 / 6.0 environment or as an executable in Windows.



Visual Basic Programming Suite

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

COMPILING AND LINKING

Programs can be compiled with the library driver routines simply by linking the library at compile time. In each case, the assumed source file is named DEMO.C.

Microsoft C

To compile and link on the command line:

cl/Ax/Gs/demo.c tcore_x_.Lib

where x is:
- S Small Model
- M Medium Model
- C Compact Model
- L Large Model
- H Huge Model

Stack checking must be turned off with the /Gs option if interrupts are used. For Codeview compatibility include the / Zi option.

To use the PMD Family Software Library Reference driver in the environment, perform the following steps:

1. In the **Make** menu select the **Set Program List** option.
2. After naming the **Make** file, select **Edit Program List**, and enter the names of the source file (demo.c) and the appropriate library (e.g. tcore_s_.lib for small model).
3. In the **Option / Make** menu select the **Compiler Flags** option and set the appropriate memory model (this model must match the library in the make list—S, M, C, L, H).



If you plan to use interrupts in a Microsoft environment, Stack Checking must be turned off. Interrupts will not work if Stack Checking is turned on.

Borland

To compile and link on the command line:

bcc -mx demo.c tcore_6_.Lib

where x is:
- S Small Model
- M Medium Model
- C Compact Model
- L Large Model
- H Huge Model

For Turbo Debugger compatibility include the -V option.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To use the PMD Family Software Library Reference driver in the environment, take the following steps:

1 In the **Project/Open Project** menu, type in the name of the project file you want to create.

2 In the **Project/Add Item** menu, enter the names of the source file (demo.c) and the appropriate library (e.g. tcore_s_.lib for small model).

3 In the **Options/Compiler/Code Generation** menu, set the appropriate memory model (this model must match the library in the Make list). If you are going to be using interrupts make sure the **Stack Checking** option is turned off. This is the default in Borland.

Linking to Dynamic Linking Libraries

If you plan to compile, link and work in the Windows environment, you will need to use a Windows Dynamic Linking Library (DLL). If you try to operate under Windows, the operating system will automatically search for the DLL. It will search first in the Systems directory, then in the current working directory for the DLL. If it doesn't find the appropriate DLL in either directory, it will give you a stop dialogue box, forcing you to quit the application. If it finds the appropriate DLL, it will automatically generate a def file and call this file where applicable.

There are two key features of DLLs from the operator's perspective. First, you need to copy the DLLs into the system directory of your host computer. Second, the DLL you copy needs to fit your Windows platform. On the CD we provide both 16- and 32-bit DLLs for both Visual C and Visual BASIC. The 32-bit DLLs are for Windows 95, and Windows NT.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Routines By Group

Initialization

tePmdAssignAxis()

tePmdInitSw()

tePmdInitDevice()

Control

tePmdMoveUpdate()

tePmdMultiUpdate()

tePmdUpdate()

Miscellaneous

teErrorCode()

teErrorCreate()

teErrorModel()

tePmdCloseAllDevices()

tePmdGetAxisHandle()

tePmdGetDeviceHandle()

tePmdGetModelNumber()

tePmdReturnAxisMask()

tePmdReturnDeviceHandle()

tePmdReturnDeviceHandleByAddr()

tePmdSetupCheckSum()

Status

tePmdCheckAxisEnabled()

tePmdCheckDeviceReady()

tePmdCheckInMotion()

tePmdCheckLoopClosed()

tePmdGetAllLimitInputs()

tePmdGetDeviceType()

tePmdGetHomeInputs()

tePmdGetModeStatus()

tePmdGetPMDVersion()

tePmdGetSampleTimeCount()

tePmdGetStatus()

Servo Loop

tePmdGetServoOutputBias()

tePmdGetServoOutputCmd()

tePmdGetServoOutputCmdMode()

tePmdGetServoOutputLimit()

tePmdSetupLoopMode()

tePmdSetServoOutputBias()

tePmdSetServoOutputCmd()

tePmdSetupServoOutputCmdMode()

tePmdSetServoOutputLimit()

Filter

tePmdGetBufFilterIL()

tePmdGetBufFilterKD()

tePmdGetBufFilterKI()

tePmdGetBufFilterKP()

tePmdGetBufFilterKVFF()

tePmdGetBufFilterPIDVFF()

tePmdGetIFilterSum()

tePmdGetSampleTime()

tePmdSetBufFilterIL()

tePmdSetBufFilterKD()

tePmdSetBufFilterKI()

tePmdSetBufFilterKP()

tePmdSetBufFilterKVFF()

tePmdSetBufFilterPIDVFF()

tePmdSetBufPosnErrZero()

tePmdSetSampleTime()

Encoder

tePmdGetnRearmCapPos()

tePmdGetStepRatio()

tePmdResetCapture()

tePmdSetStepRatio()

tePmdSetupCaptureMode()

tePmdSetupPosErrLimit()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Profile

tePmdSetupAxisEnable()
tePmdSetupLimitEnable()
tePmdSetupLimitPolarity()
tePmdSetBufGearRatio()
tePmdGetBufGearRatio()
tePmdSetBufStopAbrupt()
tePmdSetBufStopSmooth()
tePmdStopAbrupt()
tePmdStopSmooth()
tePmdRecover()
tePmdSetBufProfileSCurve()
tePmdSetBufProfileTrap()
tePmdSetBufProfileVel()
tePmdSetProfileVel()
tePmdSetProfileMode()
tePmdSetActPos()
tePmdSetBufRelEndPos()
tePmdGetActPos()
tePmdSetBufEndPos()
tePmdGetBufEndPos()
tePmdGetDesPos()
tePmdGetPosErr()
tePmdSetBufVel()
tePmdGetBufVel()
tePmdSetStdStepperRate()
tePmdSetHighStepperRate()
tePmdSetStepperMinVel()
tePmdGetStepperMinVel()
tePmdGetDesVel()
tePmdCheckDesVelZero()
tePmdSetBufAcc()
tePmdGetBufAcc()
tePmdGetBufJerk()
tePmdSetBufJerk()

Events

tePmdResetEvents()
tePmdSetupAutoUpdate()
tePmdSetupEventBP()
tePmdSetSingleHomeBkPt()
tePmdSetSingleMtnCmpltBkPt()
tePmdGetBkPtCmpVal()
tePmdGetPosErrLimit()
tePmdChecknClrBreakPoint()
tePmdChecknClrCapture()
tePmdChecknClrCmdErr()
tePmdChecknClrLimits()
tePmdChecknClrMtnCmplt()
tePmdChecknClrNegLimit()
tePmdChecknClrPosErr()
tePmdChecknClrPosLimit()
tePmdChecknClrWrapAround()

Registers

teHwaSetupConfigRegBit()
teHwaReadConfigReg()
teHwaResetDevice()
teHwaEnableDrives()
teHwaDisableDrives()
teHwaEnableRegWrite()
teHwaDisableRegWrite()
teHwaReadEncoderInputs()
teHwaReadHome()
teHwaSetupEncoderABSwap()
teHwaSetupEncoderPolarity()
teHwaSetupHomePolarity()
teHwbResetDevice()
teHwbDisableDrives()
teHwbEnableDrives()
teHwbReadDriveEnables()
teHwaWriteReg()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

teHwbWrite16()

teHwbWrite32()

teHwaReadReg()

teHwbRead16()

teHwbRead32()

teHwbReadInputs()

teHwbReadAuxInputs()

teHwbSetAuxOutputs()

teHwbClearAuxOutputs()

teHwbReadAuxOutputs()

teHwbReadDriveFaults()

teHwbWriteAxisCfg()

teHwbWriteDeviceCfg()

teHwbReadAxisCfg()

teHwbReadDeviceCfg()

teHwbReadIDSpace()

teHwbCalDacOffsets()

teHwbStoreEEConfig()

teHwbLoadEEConfig()

teHwbReadWriteCycles()

te565XCheckInputBit()

te565XClearOutputs()

te565XReadInputs()

te565XReadOutputs()

te565XSetOutputs()

te565XWriteOutputs()

Interrupts

tePmdGetEventsMask()

tePmdEnableEvents()

tePmdDisableEvents()

tePmdPollEvents()

tePmdPollAllEvents()

tePmdDisableEventProcessing()

tePmdEnableEventProcessing()

teBufGetEventsMask()

Internal

tePmdSendCmd()

tePmdSendCmdOut()

tePmdSetAxis()

Commutation

tePmdInitPhaseAlgor()

tePmdSetupPhaseCommMode()

tePmdSetupPhaseInitMode()

tePmdSetupPhaseType()

tePmdSetupPrescale()

tePmdSetPhaseCounts()

tePmdGetPhaseCounts()

tePmdSetBufServoOutputCmd()

tePmdGetBufServoOutputCmd()

tePmdSetKPhase()

tePmdGetKPhase()

tePmdSetPhasePos()

tePmdGetPhase()

tePmdGetPhaseInfo()

tePmdSetPhaseOffset()

tePmdGetPhaseOffset()

tePmdSetPhaseTime()

tePmdGetPhaseTime()

tePmdInitPhase()

tePmdSetHallSense()

tePmdGetHallSense()

tePmdGetHall()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example 5312B Program in C

```
#include "te5312.h"
#include <stdio.h>
#include <conio.h>

#define BOARD  0
#define AXIS_A 0
#define AXIS_B 1
#define GLOBAL -1

// interrupt hook prototypes
static void te5312IndexAlert(short *psAxisNum);
static void te5312WrapAroundAlert(short *psAxisNum);

// interrupt counters
static unsigned short wCarryA, wCarryB;
static unsigned short wIndexA, wIndexB;

void main()
{
  unsigned short wBoardAddr;
  long lCntA, lCntB;
  short sStatA, sStatB;
  short sIRQNum;
  // get the address
  printf("\nEnter the base address the 5312 is strapped at in hexadecimal - ");
  scanf("%x", &wBoardAddr);
  // get the IRQ number
  do{
    printf("\nEnter the interrupt request line used (2 to 7) - ");
    scanf("%u", &sIRQNum);
  }while((sIRQNum < 2) || (sIRQNum > 7));
  // initialize the software
  te5312InitSw();
  // initialize the board
  te5312InitBoard(wBoardAddr,4);
  // zero the counters
  te5312LoadCntr(GLOBAL, 0L);
  // initialize interrupts
  te5312InterruptHooks(te5312WrapAroundAlert, te5312IndexAlert);
  te5312EnableIRQ(BOARD, sIRQNum);
  te5312IndexAlertOn(GLOBAL);
  te5312WrapAroundAlertOn(GLOBAL);
// print column headers
  printf("\nPress any key to exit\n\n"
    "              Axis A                  "
    "              Axis B\n"
    "                  Index    WrapAround  "
    "                  Index    WrapAround\n"
    "  Count  Status Interrupts Interrupts  "
    "  Count  Status Interrupts Interrupts\n");
  // display counter values and status until key pressed
  while(!kbhit()){
    lCntA = te5312ReadCntr(AXIS_A); lCntB = te5312ReadCntr(AXIS_B);
    sStatA = te5312ReadSts(AXIS_A); sStatB = te5312ReadSts(AXIS_B);
    printf("\r%8ld   %2X     %5u      %5u     ", lCntA, sStatA,
      wIndexA, wCarryA);
    printf("%8ld   %2X     %5u      %5u", lCntB, sStatB,
      wIndexB, wCarryB);
  }
  if (!getch())
    (void)getch();
  printf("\n");
  // disable interrupts before exiting program
  te5312DisableIRQ();
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example 5312B Program in C: (continued)

```
void te5312WrapAroundAlert(short *psAxisNum)
{
  switch(*psAxisNum){
  case AXIS_A: wCarryA++; break;
  case AXIS_B: wCarryB++; break;
  }
}

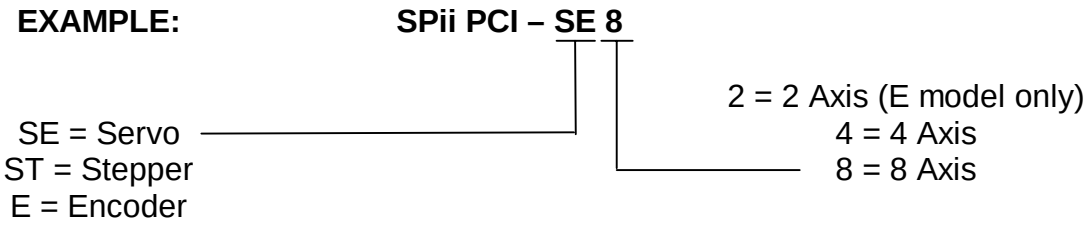
void te5312IndexAlert(short *psAxisNum)
{
  switch(*psAxisNum){
  case AXIS_A: wIndexA++; break;
  case AXIS_B: wIndexB++; break;
  }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

LIBRARY PROGRAMMABLE: ORDERING MATRIX

SPii PCI Ordering Matrix

- 4-8 Axis PC/PCI Bus Motion Controller
- High Performance SPii Servo Processor motion control
- Supports Servo & Stepper Motors



Standard SPii PCI Includes:

- Second-generation custom designed Servo Processors (SPii)
- Servo and Stepper Control on a Single PCI Card
- Distributed Architecture Control
- Programmable Acceleration/Deceleration
- Programmable Velocity Profiling (Third-order S-Curve profile generation)
- Opto-Isolated Inputs & Outputs
- Each axis has dedicated control inputs – Two Stop Limits, two Deceleration inputs and one Home Input
- Four programming modes and eight operating modes
- Open Architecture Software Library; drivers for Windows NT / 98 / 2000
- Documentation includes Hardware, Software, and Application Guides

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5000 Ordering Matrix

- 1-3 Axis PC/ISA Bus Motion Controller
- Library Programmable Motion Controller
- Supports Stepper Motors

EXAMPLE: **5000 – 3**

1 = 1 Axis Pulse & Direction
2 = 2 Axis Pulse & Direction
3 = 3 Axis Pulse & Direction

Standard Model 5000 Includes:

- Nippon Pulse chipset
- 1-3 Axes of Stepper Control
- 240 KHz Step Rate
- Programmable Acceleration/Deceleration
- Programmable Velocity Profiling
- Opto-Isolated Inputs & Outputs
- Each axis has dedicated control inputs – Two Stop Limits, two Deceleration inputs and one Home Input
- Each axis has outputs for Pulse, Direction and Hold
- Four programming modes and eight operating modes
- Open Architecture Software Library; drivers for DOS, Win3.1 and Win95 (16-bit); For C, C++, BASIC & Pascal (Part of the Development Kit).
- Hardware and Software Manuals and Documentation (Part of the Development Kit)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5650A Ordering Matrix

- 1-4 Axis PC/ISA Bus Motion Controller
- Library Programmable Motion Controller
- Supports Servo Motors

EXAMPLE: **5650A**

50A = 1 – 4 axis Position & Velocity Controller
51A = 1 – 2 axis with Sinusoidal Commutation

Standard Model 5650A Includes:

- Eight digital, general purpose TTL I/O can be configured as either inputs or outputs
- Each of the four axes has individual home, and ±limit
- There are two drive enable outputs per board, 1& 2 are shared, 3 & 4 are shared
- All limit inputs are capable of stopping motion in either direction
- A 1.0-MHz incremental encoder input provides position feedback
- The 5650 accepts single-ended or differential quadrature signals, and index pulses
- A high-speed (1 usec) **position capture feature** allows the latching of position values during a move in progress
- Built in **Electronic gearing** allows for tight control between axes without taking up host processor time. The host specifies one parameter, the gear ratio. Applying the ratio to the current position of the master encoder generates the target position for the slaved axis
- Three forms of output are supported, ±10VDC DAC, PWM, and two Phase Sinusoidal commutation
- Open Architecture Software Library; drivers for DOS (16-bit), Win95/98/NT (32-bit DLL's); for C, C++, and Visual Basic (Part of the Development Kit).
- Hardware and Software Manuals and Documentation (Part of the Development Kit).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5950B Ordering Matrix

- 1-4 Axis PC/104 Bus Motion Controller
- Library Programmable Motion Controller
- Supports Servo and Stepper Motors

EXAMPLE: **5954B**

50BP =1 – 4 Axis PWM Outputs
50B = 1 – 4 Axis Analog Outputs
51B = 1 – 2 Axis Analog
Sinusoidal Commutation Outputs
54BP = 1 – 4 Axis Pulse & Direction

Standard Model 5950B Includes:

- The heart of the Model 5950B is its PMD RISC with a custom ACS-Tech80 I/O chip
- Four digital, general purpose I/O can be configured as either inputs or outputs
- Each of the four axes has individual home, ±limit, and drive fault inputs
- Each axis has an individual drive enable output
- All limit inputs are capable of stopping motion in either direction
- The home input is configurable as either differential or single ended
- The home and limit inputs contain a digital, software selectable debounce circuit
- A 1.0-MHz incremental encoder input provides position feedback
- The 595X accepts single-ended or differential quadrature signals, and index pulses
- Built in **Electronic Gearing** allows for tight control between axes without taking up host processor time. The host specifies one parameter, the gear ratio. Applying the ratio to the current position of the master encoder generates the target position for the slaved axis
- Four forms of output are supported, ±10VDC DAC, PWM, two Phase Sinusoidal commutation, and Differential Pulse / Direction for Stepper
- Open Architecture Software Library; drivers for DOS (16-bit), Win95/98/NT (32-bit DLL's); for C, C++, and Visual Basic (Part of the Development Kit).
- Hardware and Software Manuals and Documentation (Part of Development Kit).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENCODER INTERFACE: ORDERING MATRIX

5912 Ordering Matrix

- 1-4 Axis PC/104 Bus Encoder Interface Board
- Library Programmable Encoder Interface Board
- Hardware Position Capture and Compare

EXAMPLE: **5912 – 4**

1 = 1 Axis Encoder Inputs
2 = 2 Axis Encoder Inputs
3 = 3 Axis Encoder Inputs
4 = 4 Axis Encoder Inputs

Standard Model 5912 Includes:

- Provides decoding for up to four incremental quadrature encoders or pulse
- 1.2 MHz Quadrature Count Rate Per Axis (Filtered)
- MHz Quadrature Count Rate Per Axis (Unfiltered)
- MHz Pulse Rate Per Axis Lines (Filtered)
- 15 MHz Pulse Rate Per Axis Lines (Unfiltered)
- 24-bit Pre-Settable Counters, Or Chain 2 Together for 48 Bits
- Software Configurable Timer Internal Capture Mode, or Timestamp Mode
- Digitally Filtered Inputs With Selectable Sampling Rates
- Interrupts On Index Pulse, Overflow/Underflow, or Compare Value Match
- Index Actions: Load Counter From Preset, Capture
- Counter to Output Latch, Gate Counter On or Off, Reset Counter
- Open Architecture Software Library; drivers for DOS, Win3.1 and Win95 (16-bit); For C, C++, BASIC & Pascal (Part of the Development Kit).
- Hardware and Software Manuals and Documentation (Part of the Development Kit).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5312B Ordering Matrix

- 1-4 Axis PC/ISA Bus Encoder Interface Board
- Library Programmable Encoder Interface Board
- Hardware Position Capture and Compare

EXAMPLE: **5312B – 4**

1 = 1 Axis Encoder Inputs
2 = 2 Axis Encoder Inputs
3 = 3 Axis Encoder Inputs
4 = 4 Axis Encoder Inputs

Standard Model 5312B Includes:

- The board provides decoding for up to four incremental quadrature encoders
- It can be programmed to react to index with reset, capture, pregate and gate
- The 5312B offers an affordable solution for applications that need positional feedback, digital readout, and event counting for high input signal occurrences
- Inputs on the 5312B may be single-ended or differential and are conditioned by a four-phase digital filter
- Power (+5V) and grounding connections are provided for encoder use
- There are five jumper-selectable filter clock frequencies, ranging up to 10 MHz
- The board applies the signal inputs to a 24-bit decoder / counter with programmable modes
- The counter may be used either for quadrature decoding or as a counter for pulse and direction or count up/down inputs
- Counters may be cascaded in 24-bit groups
- The counters are independent, but may be initialized simultaneously by issuing commands to a single port
- When an index pulse occurs, the count may be latched or a new count value may be loaded
- Open architecture software library; drivers for DOS (16-bit), Win95/98/NT (32-bit DLL's); for C, C++, and Visual Basic (Part of the Development Kit).
- Hardware and Software Manuals and Documentation (Part of the Development Kit).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP