p/n YPM08128

Parker Automation

Acroloop Motion Control Systems Technical Brochure

Effective: October 7, 2002

This page intentionally left blank.

CHANGE NOTICE

ACROLOOP Technical Brochure Version Change:
From: Version 3.00
To: Version 4.00

The following changes have been incorporated into ACROLOOP Technical Brochure Version 4.00:

Miscellaneous	Added ACR1200 and ACR8000 information to brochure.
Introduction	Updated Acroloop Controller Highlights
Hardware Overview	Updated ACR1500, ACR2000, and ACR8010 drawings.
Controller Firmware	Updated Servo Loop Diagrams
	Changed “Motion Profile Group” to “Velocity Profile Group”
	Added Version 1.18 commands
Development Tools	Added Operating System information
	Updated “List of Library Functions”
Programming Examples	Replaced Program 9 C Interface example with a C++ Interface example
Controller Specifications	Updated specifications
ACR1500 Ordering Matrix	Updated information
ACR2000 Ordering Matrix	Updated information
ACR8010 Ordering Matrix	Updated information
Optional Accessories	Added chassis drawings
	Added Interconnecting Cable information
	Added 16-Bit A/D Board information
	Updated Expansion I/O information
Appendix	Updated Parameter Overview
	Updated Flag Overview

This page intentionally left blank.

TABLE OF CONTENTS

INTRODUCTION

Acroloop's Multi-Axis Motion Controllers

The Acroloop Controller Family continues to grow. Currently, there are five different Acroloop models available.

- ACR8010: 1-8 axis PC-Bus or Standalone Motion Controller
- ACR8000: 1-8 axis PC-Bus or Standalone Motion Controller
- ACR2000: 1-4 axis PC-Bus or Standalone Motion Controller
- ACR1500: 1-4 axis PC-Bus Motion Controller
- ACR1200: 1-2 axis Standalone Motion Controller

All of the controllers program identically and use the same firmware set. The main differences between the controllers are performance, features and cost. The ACR8010 and ACR2000 have processor speed advantages for interpreted programs in standalone applications. The ACR8010, ACR8000, ACR2000, and ACR1200 have on-board optically-isolated digital I/O for harsh environments. A basic summary is as follows:

Acroloop Product	Number of Axis/Board	Processor Frequency	Digital I/O # (Std/Exp)	Digital I/O Type
ACR8010	1-8	60	64/320	24VDC, Opto
ACR8000	1-8	27	64/320	24VDC, Opto
ACR2000	1-4	50	32/288	24VDC, Opto
ACR1500	1-4	40	48/--	5VDC, TTL
ACR1200	1-2	40	32/288	24VDC, Opto

Std = Standard
Exp = Expansion
Opto = Optically-Isolated

The Acroloop controllers are used in a wide variety of single and multi-axis applications. The applications for the Acroloop controller family are virtually unlimited, due to the flexible design. The main purpose of the controllers is to provide a flexible, yet easy to program environment, while insuring the motion profiles and digital I/O tasks are executed in real-time, independent of any operating system or operating station.

The Acroloop controllers are 32/64 bit floating-point Multi-Tasking DSP-based motion controllers. The Acroloop controller family is ideal for applications requiring a high degree of processing power and fast performance - eliminating the need or burden for a host processor.

The controllers can perform independent or coordinated axis moves including circular, linear, sinusoidal, elliptical, helical, and spherical interpolation in any combination of up to 4 and 8 axis. In addition, the user can define up to 8 coordinate systems for up to 24 programs (16 motion and 8 PLC programs). **In typical applications, the pre-emptive multi-tasker will execute motion profiles and execute a ladder logic PLC simultaneously.**

INTRODUCTION

In PC-based applications, the Acroloop motion controllers have the ability to perform the bulk of the calculations for motion control and command the I/O ports, while updating the graphical operator display is left for the CPU. This is especially important for Windows based applications where the engineer must insure the motion and PLC operations are executed in real time.

Acroloop Controller Highlights:

Hardware

- Texas Instruments™ TMS320C3X Series - 32/64 bit Floating Point DSP
- Pre-emptive multi-tasker for up to 16 simultaneous motion programs
- Pre-emptive multi-tasker for up to 8 simultaneous PLC programs
- Handles 1-2, 1-4, or 1-8 axis applications ("cascade-able" for additional axis)
- 32 bit Floating Point DSP @ 27/40/50/60 MHz
- 64 optically-isolated Digital I/O on ACR8000/ACR8010, expandable to 320 digital I/O
- 32 optically-isolated Digital I/O on ACR1200/ACR2000, expandable to 288 digital I/O
- 48 TTL Opto-22 Compatible Digital I/O on ACR1500
- On-board 24VDC @ 1A Watchdog Relay (SPDT) for ACR1200/ACR2000/ACR8000/ACR8010
- On-board Open-Collector Normally-Open and Normally-Closed Watchdog Outputs @ 30mA for ACR1500
- EPROM's for permanent firmware and program storage
- Flash Memory for permanent program storage
- Zero Wait State System RAM
- 1 Megabyte RAM memory on ACR2000 and ACR8010
- 256Kbytes RAM memory on ACR1200 and ACR1500
- 128Kbytes RAM memory on ACR8000
- Hardware Capture and Compare registers
- 8 User Programmable Status LED's on ACR8000 and ACR8010
- Handles both servo and stepper applications
- Optional 12-Bit or 16-Bit A/D (analog-digital) input boards for ACR1200/ACR1500/ACR8000/ACR8010
- Optional 12-Bit On-board A/D (analog-digital) input for ACR2000

Communications

- PC/ISA Bus available on ACR1500/ACR2000/ACR8000/ACR8010
- Serial Port and Parallel Port communications available on ACR1200/ACR2000/ACR8000/ACR8010
- RS-232 or RS-422 (2 COM ports with automatic baud detect to 38.4K)
- Parallel port (152Kb - 300Kb effective baud rate)
- PC-BUS port with 2 High Speed FIFO's (512 x 8)
- Simultaneous Communications on all ports
- Set up software (AcroVIEW) provided free of charge (RS-232 or PC-BUS)
- C++, Visual C++, and Visual Basic Libraries
- DOS, Windows, and Windows NT operating system Drivers

Firmware

- PID, Vff, Aff with Notch and Pass filtering (Bi-Quad Filtering)
- Closes Position and Velocity Loops
- Linear, Circular, Sinusoidal, Helical, & Elliptical interpolation
- 1-2 (ACR1200), 1-4 (ACR1500 or ACR2000), or 1-8 (ACR8000 or ACR8010) Axis Interpolation
- Mixing of Axis Combinations and Types of Interpolation
- Servo and Stepper (Closed or Open Loop) Combinations
- Closed Loop with Encoder or Analog Feedback
- Dual Encoder Feedback (Position and Velocity Loops)
- Ladder Logic PLC
- 8 Programmable Limit Switches
- Floating Point Electronic Gearing
- Ratchet Mode (Gearing similar to a ratchet wrench)
- Segmented Electronic CAM
- Incremental and Absolute Jogging (Independent Axis Control)
- External or Internal Time Base Programming
- S-Curve Acceleration and Deceleration
- Ballscrew and Backlash Compensation
- On-the-Fly Positioning and Trajectory Calculations
- Global and Local Arrays and Variables
- Direct Parameter and Flag Access (pre-programmed)
- 8 Channel Data Acquisition Commands
- Sinusoidal/Trapezoidal Commutation

Performance

- New Trajectory calculated every interrupt (200-500 microseconds)
- 50 microsecond servo update per axis
- 16-bit DAC output resolution (+/- 10VDC standard, programmable)
- 4 MHz stepper output; Pulse Width range: 125 nsec - 16 usec
- 100 nsec - 1 usec Position Capture Latency
- 100 nanosecond Position Compare Latency
- Data Acquisition sampling rate of 2 - 5KHz (depending on # of axis)
- 0.1Hz to 8 MHz Encoder Count Rate for ACR2000/ACR8000
- 0.1Hz to 20 MHz Encoder Count Rate for ACR1200/ACR1500/ACR8010
- 5 millisecond PLC scan rate (800 PLC instructions, 8 timers, and 8 counters)
- 200 - 500 microsecond PLS scan rate

CONTROLLER FEATURES

Floating Point DSP CPU

The Acroloop controllers use a **32/64 bit Floating Point Digital Signal Processor (DSP).** The Floating Point DSP used in the motion controllers has superior processing power and is specifically designed to handle motion control applications. Fixed Point DSP's and other non-DSP's cannot match the processing power and capability of the Acroloop controller's CPU. The main features of the processor are:

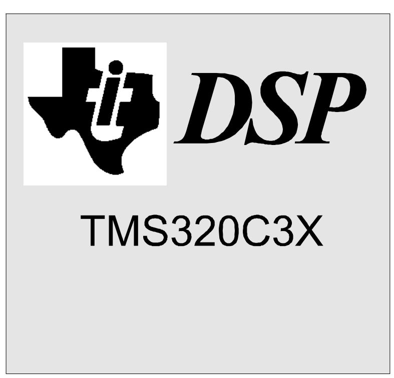
- 32/64 bit Floating Point DSP CPU
- 27/40/50/60 MHz (ACR8000/ACR1200/ACR1500/ACR2000/ACR8010)
- 27/40/50/60 MFLOPS (ACR8000/ACR1200/ACR1500/ACR2000/ACR8010)
- 13.5/20/25/30 MIPS (ACR8000/ACR1200/ACR1500/ACR2000/ACR8010)

The Texas Instruments TMS320C3X series processor used in the Acroloop motion controllers compares favorably in relation to other processors used in the market (see table below):

Processor Comparison Table

Processor	DSP	Bits	Frequency	MFLOPS	MIPS
TMS320C3X	Floating Pt	32	60 MHz	60	30
DSP56001FEXX	Fixed Pt	24	60 MHz	0	30
Motorola 68331	Non-DSP	32	16.8 MHz	0	10.0
ADSP-2105	Fixed Pt.	16	40 MHz	0	10.0

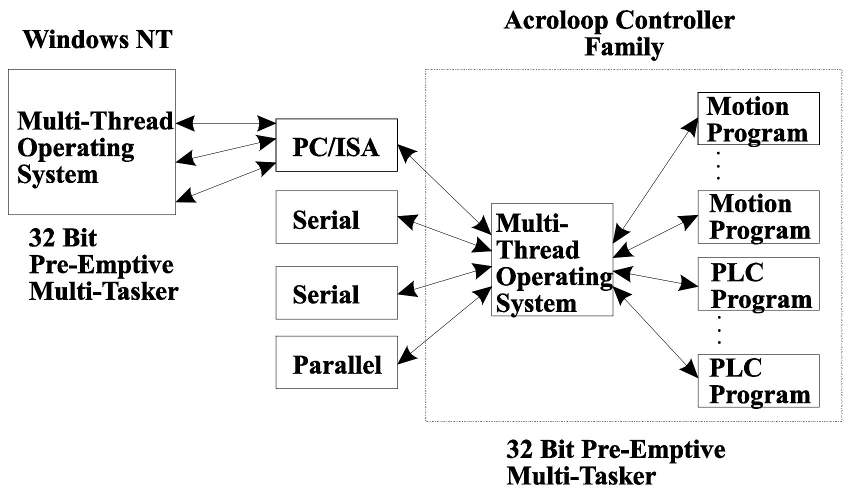
In comparison to other controllers on the market, the Acroloop controllers can compute algorithms, position, velocity, and acceleration terms faster and with more accuracy since it uses the 32/64 bit Floating Point DSP CPU and supporting architecture. This feature allows cost effective accuracy in position critical applications and offers time saving throughput since the settling time of the machine is reduced. Speed and accuracy equate to monetary value. Other non-floating point processors just can't compare which sacrifices response time and/or accuracy and precision. Furthermore, a floating point processor does not induce rounding errors whereas non-floating point processors will induce rounding errors in the motion controller.



Pre-Emptive Multi-Tasker

The Acroloop controllers are **true pre-emptive multi-taskers capable of performing multiple simultaneous tasks**. For example, the feedhold condition could be running on a PLC program (PLC0). Also, program number 1 is executing the motion profile for an XY system as well as monitoring the PC-BUS port for commanded positions from a PC. In addition, program 2 (PROG2) is sending sampled current position data and following error data serially to an off-line PC that is being used as a data collection terminal for statistical process control. The Acroloop controllers can perform all of these tasks simultaneously and using critical on-board processing power only when it is required.

The Acroloop motion controllers are unique since they are 32 bit pre-emptive multi-taskers. The architecture allows the motion controllers to be able to communicate over the PC-Bus, 2 serial ports, and 1 parallel port simultaneously. The controllers can also be provided with multi-threaded Windows NT drivers. The Acroloop controllers are one of the few if not the only board level motion controller that supports multi-threaded Windows NT applications.



The Acroloop controllers can perform the following with the pre-emptive multi-tasker:

- Perform 8 motion programs (50 microsecond/axis servo update rate)
- Perform an additional 8 programs (non-motion programs)
- Command up to 8 PLC programs (1.5 - 5 millisecond scan time)
- Command up to 4 communications ports simultaneously

Analog/Stepper Flexibility

The controllers can be set up to accommodate either servo drives or stepper drives. The controllers can be configured in any combination of pairs of servo and/or stepper outputs. For example, an ACR8010 4-axis board can be configured with 2 servo outputs and 2 stepper outputs on the same board. The controllers can operate the outputs independently or dependently depending on the application.

Analog Output Signals:
Output Resolution: 16 bits
Voltage Range: +/-10VDC, programmable

Stepper Output Signals:
Velocity Range: up to 4MHz
Pulse Width Range: 125 nanoseconds - 16 microseconds
Drive Enable: Both low current and normal operation

The stepper and analog outputs can be mixed and matched in any combination of profiles or programs. All axes could be run in a single profile or each individual axis could be run in a separate profile or any combination thereof.

Please see the Hardware Overview or the **Ordering Matrix** in this brochure for additional information.

Trajectory Calculation

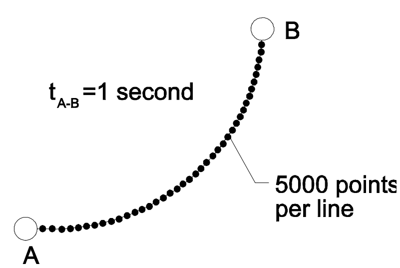
The Acroloop controllers can calculate new trajectories at extremely fast speeds. The Acroloop controllers are unique since they **calculate a new trajectory point every interrupt.** A new trajectory point is calculated every 200-500 microseconds. Comparatively, other motion controllers on the market calculate trajectories at a 5-10 millisecond rate. Thus, the accuracy of the Acroloop controllers is far superior.

Trajectory Calculation Table

Number Of Axes	Calculation Bits	(27/50 MHz) Points/Second
2	64	5,000
4	64	3,333
6	64	2,500
8	64	2,000

All of the trajectory calculations are performed on-board and calculated with a floating point processor. For example, based on a servo loop update time of 50 microseconds per axis, the servo update time for 2 axes is 100 microseconds. By adding an additional 100 microseconds for “overhead” tasks (PLC, calculations, etc.), the total interrupt period is 100usec + 100usec = 200usec. The trajectory calculation is then the inverse of 200 microseconds or 5,000 points per second.

Trajectory Calculation - Trajectories calculated every interrupt



Calculating a new trajectory every interrupt is the method of choice when the process consists of normal motion geometry’s (i.e. lines and arcs). The controllers can perform all trajectory calculations without host intervention. Other motion controllers require the host to perform all the lengthy calculations and must be programmed by the user.

The net result is the ability to “engineer” a complete solution without sacrificing either speed or precision. **The flexibility of the Acroloop controllers allow the designer to “engineer” the system by selecting data point spacing or programming the interrupt speed so that the desired speed and precision is maintained**.

Floating Point Mathematics Precision

The motion controllers are engineered around the Texas Instruments TMS320C3X Floating Point CPU. The controllers are also provided with 64-bit precision floating point math functions. Compare the 32-bit and 64-bit precision floating point variables to other motion controllers; you won't get rounding errors with the Acroloop controllers. For example, a simple electronic gearing ratio can be set to a floating point number unlike other controllers with limited fixed point or even integer gear ratio ranges.

The Acroloop controller family is supplied with a **64-bit Floating Point Math Library**. The Texas Instruments TMS320C3X uses 6 "decimal" digit numbers as standard to provide 32-bit accuracy. The ACR8000 uses a 16 "decimal" digit floating point number to provide superior accuracy. **For example, you can represent the distance between the Earth and the Sun to a precision of about 20 miles with a 6 "decimal" digit number. The same distance can be represented to a precision of 1/1000th of an inch with the 16 "decimal" digit number.**

The unique design of the processor allows the Floating Point Library to be installed on-board. The same Floating Point Library could not be installed on other processors due to their limited design and architectural features. With the Floating Point Library installed, a full complement of parametric evaluations is possible. The built-in floating point evaluator operates on **both global and local variables** programmable in the Acroloop controller family. Unlike other motion controllers, the intermediate calculations for trajectories, parametric evaluations, PLC operations, high speed position captures, and many other tasks can all be performed simultaneously on the Acroloop motion controller without host intervention. If a PC host is required, it only has to deal with updating the graphical display and feeding new operator information to the motion controller while the Acroloop board takes care of everything else.

Mathematical Expression List

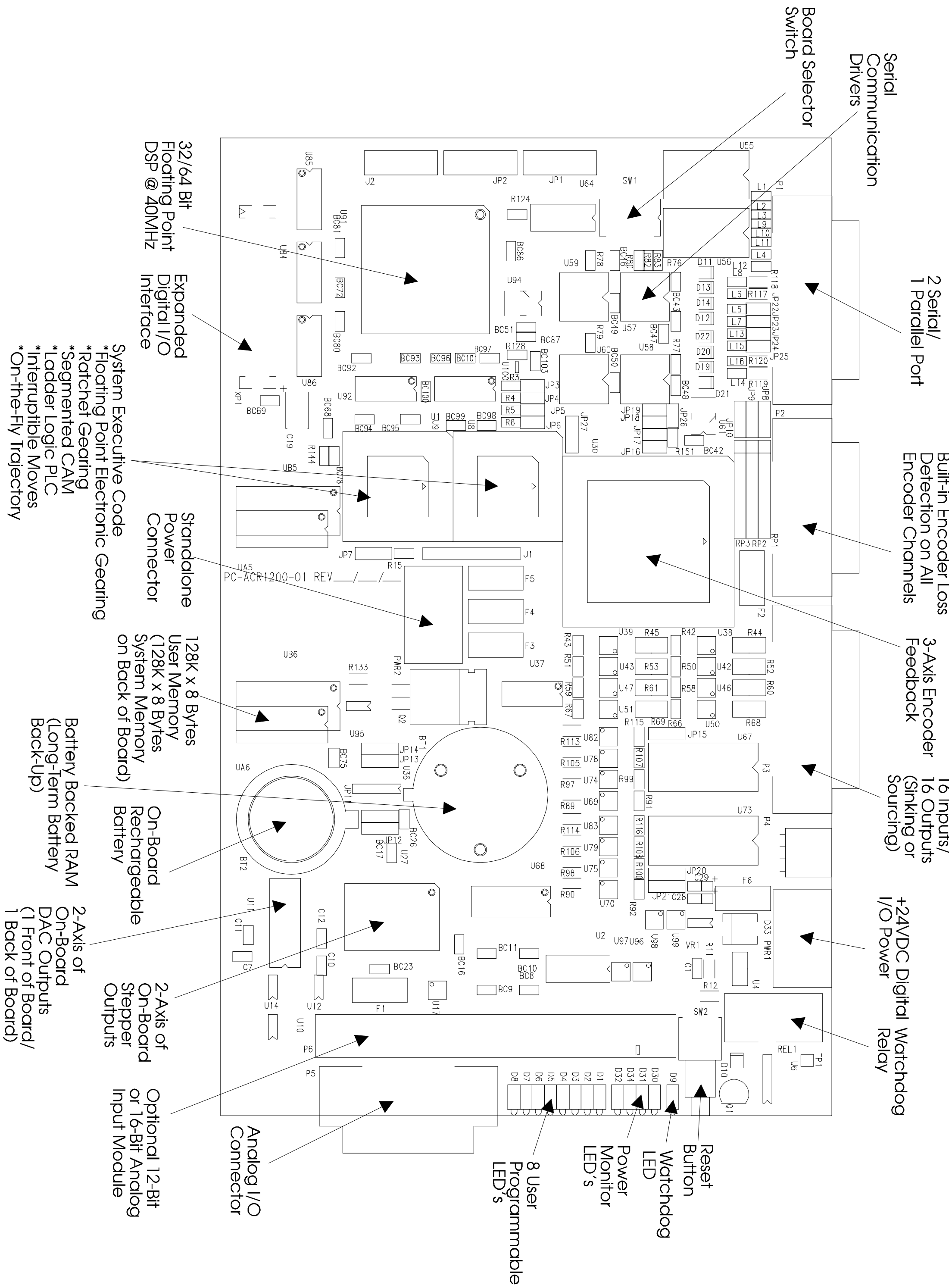
Type	Functions
Arithmetic	Addition, Subtraction, Multiplication, Division and Exponential
Trigonometric	SIN, COS, TAN, ATAN, ACOS, and ASIN
Hyperbolic	SINH, COSH, TANH, ATANH, ACOSH, and ASINH
Logarithmic	LOG, and LN
Comparison	=, <>, <, >, >=, and <=
Logical	AND, OR, XOR, NAND, NOR, XNOR, NOT, >>, and <<
Miscellaneous	Square Root, and Rounding

With the floating point evaluator on the ACR1200, ACR1500, ACR2000, ACR8000, and the ACR8010, a multitude of 32-bit and 64-bit parametric programming can be made by the motion controllers. In addition, the Acroloop controller family can use the built-in commands to perform virtually any interpolated move in any combination of up to 8 axis respectively.

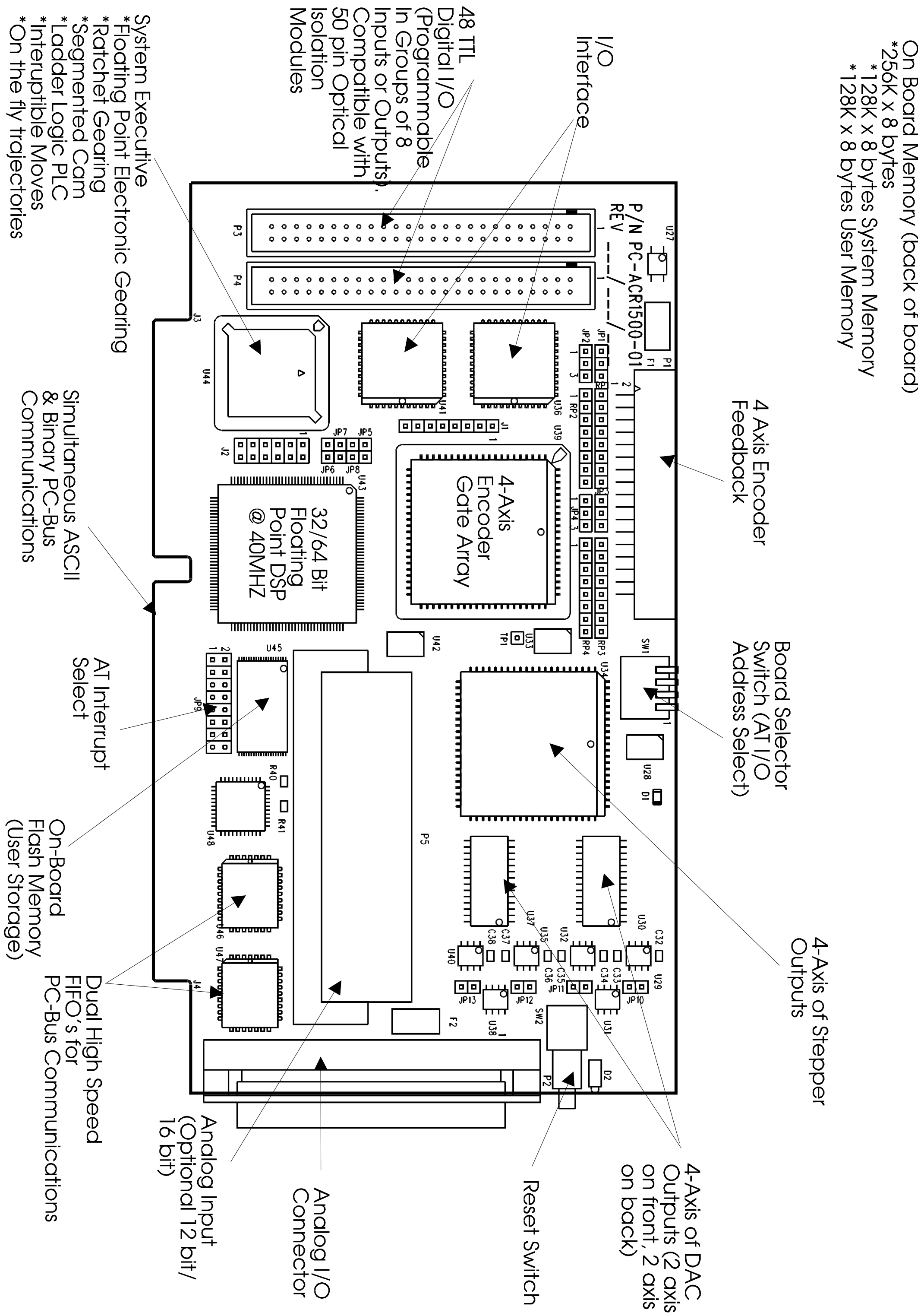
Hardware Overview

The following pages outline the hardware features on the ACR1200, ACR1500, ACR2000, ACR8000, and the ACR8010.

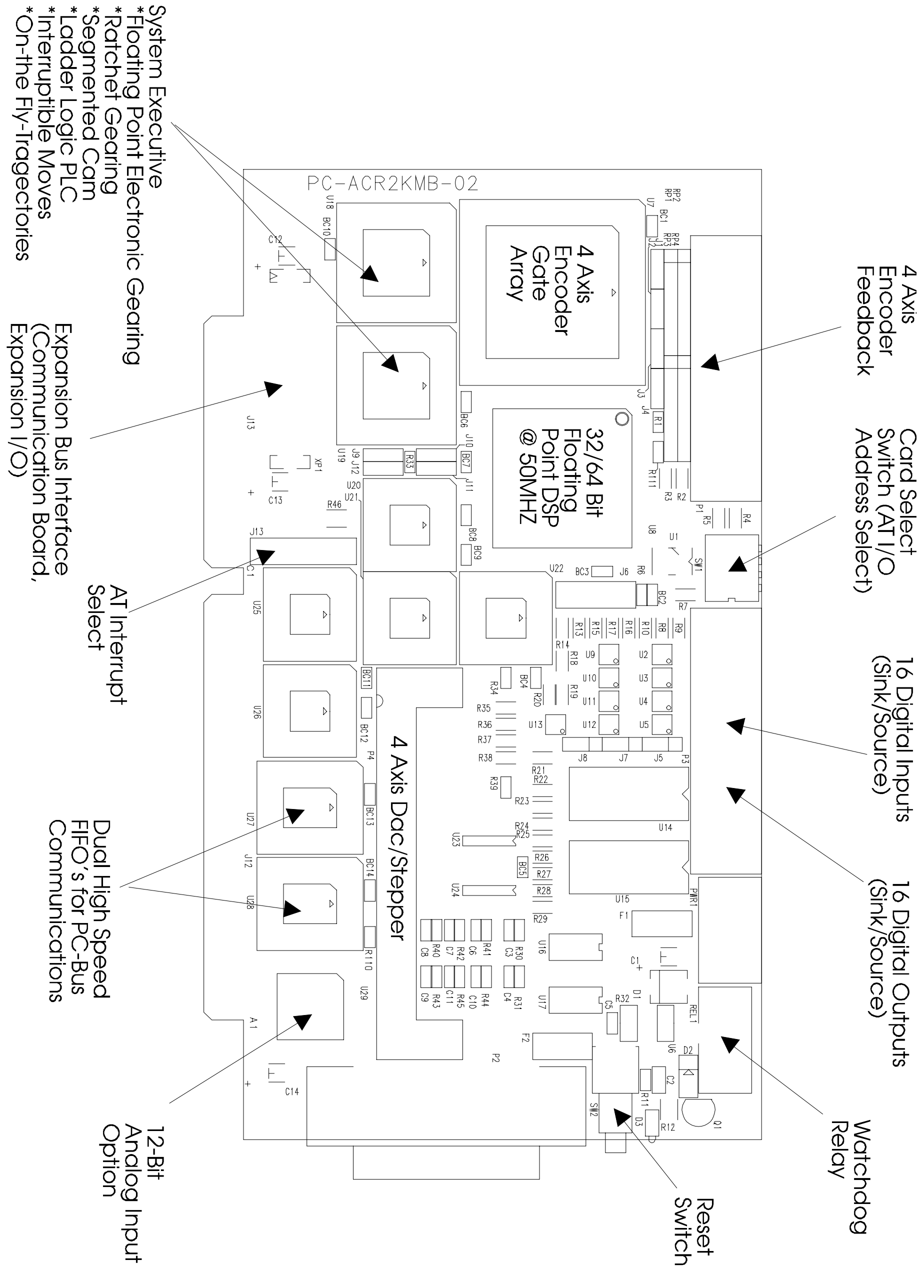
ACR1200 Hardware Overview (Front of ACR1200)



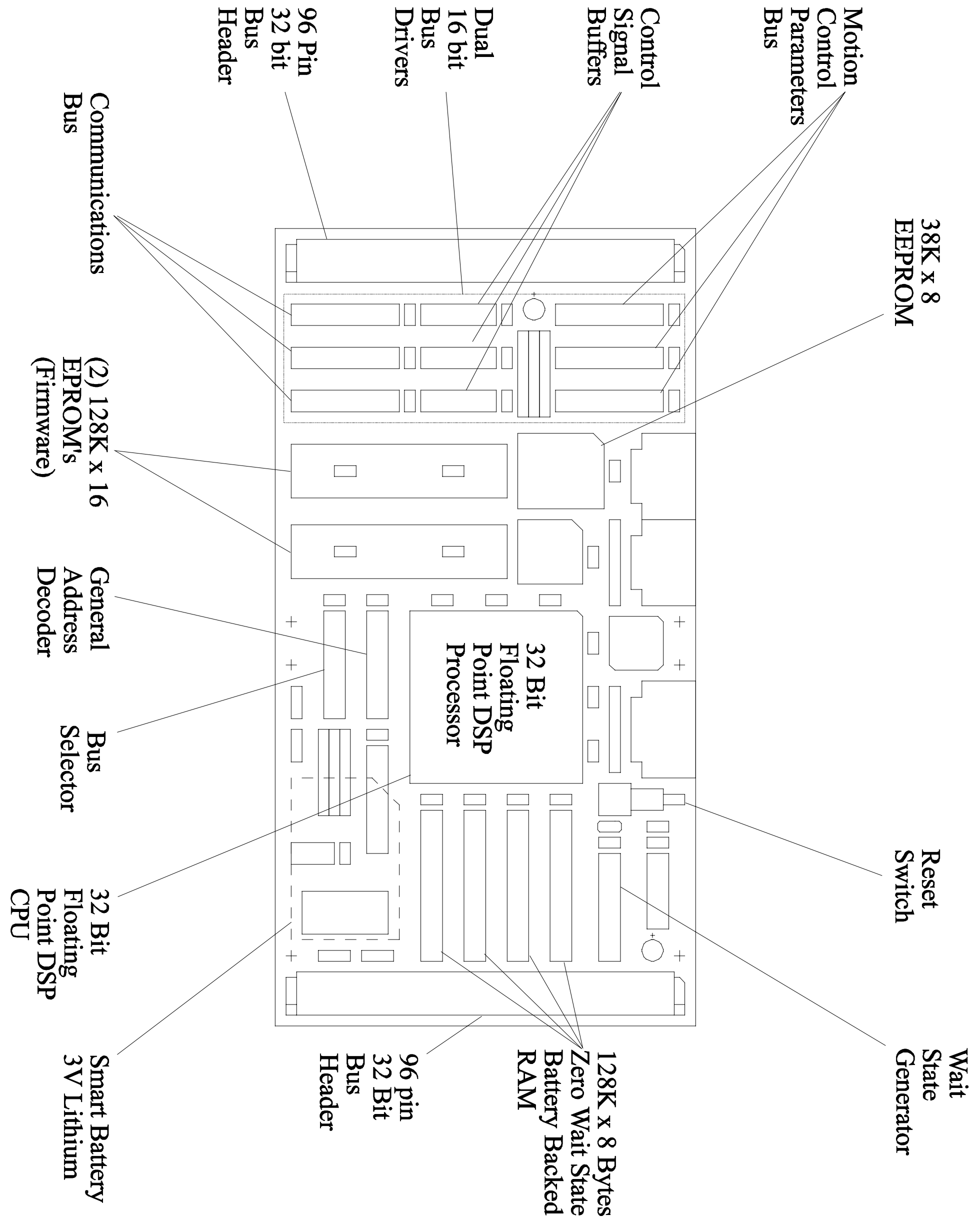
ACR1500 Hardware Overview (Front of ACR1500)



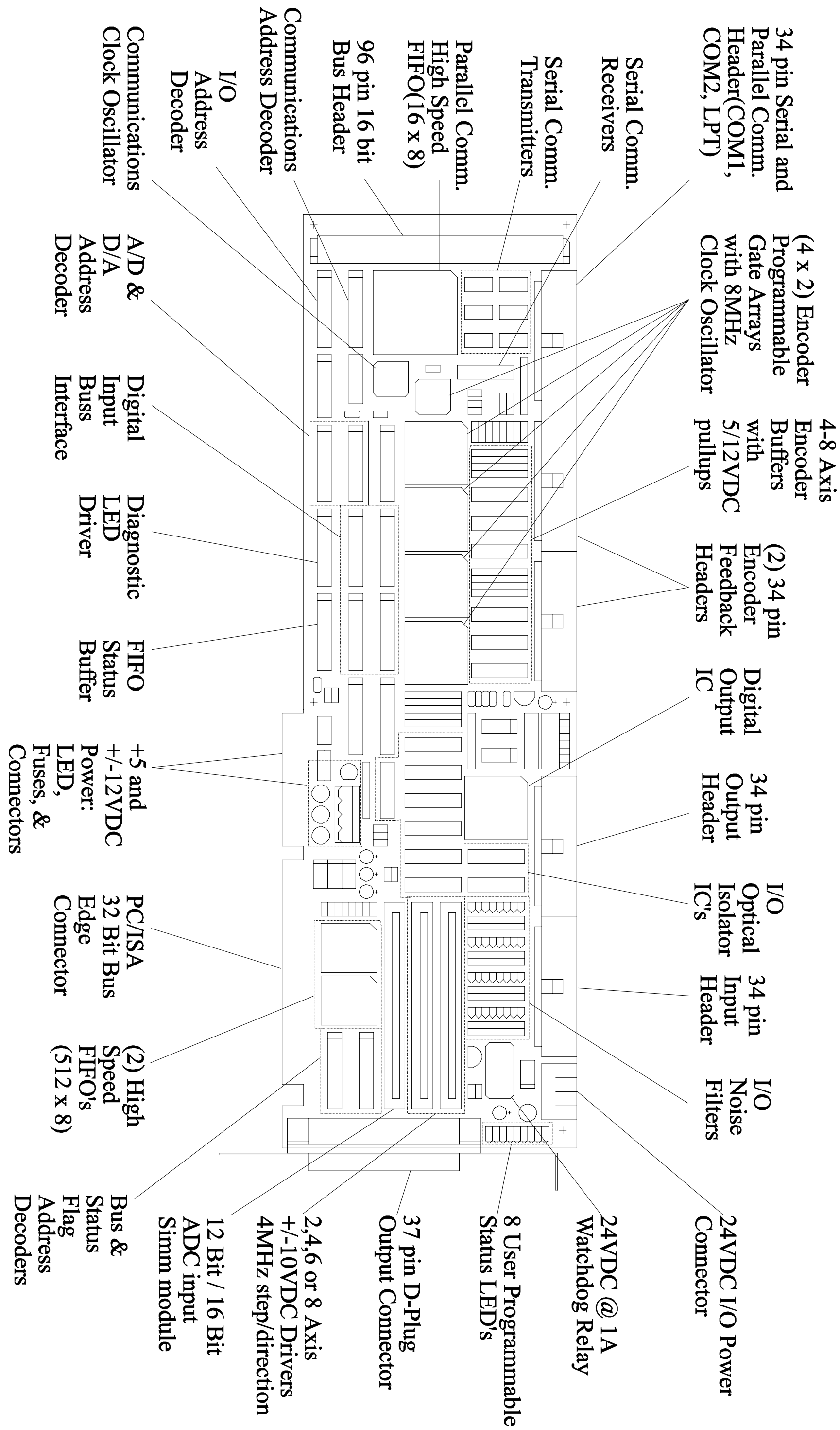
ACR2000 Hardware Overview (Front of ACR2000)



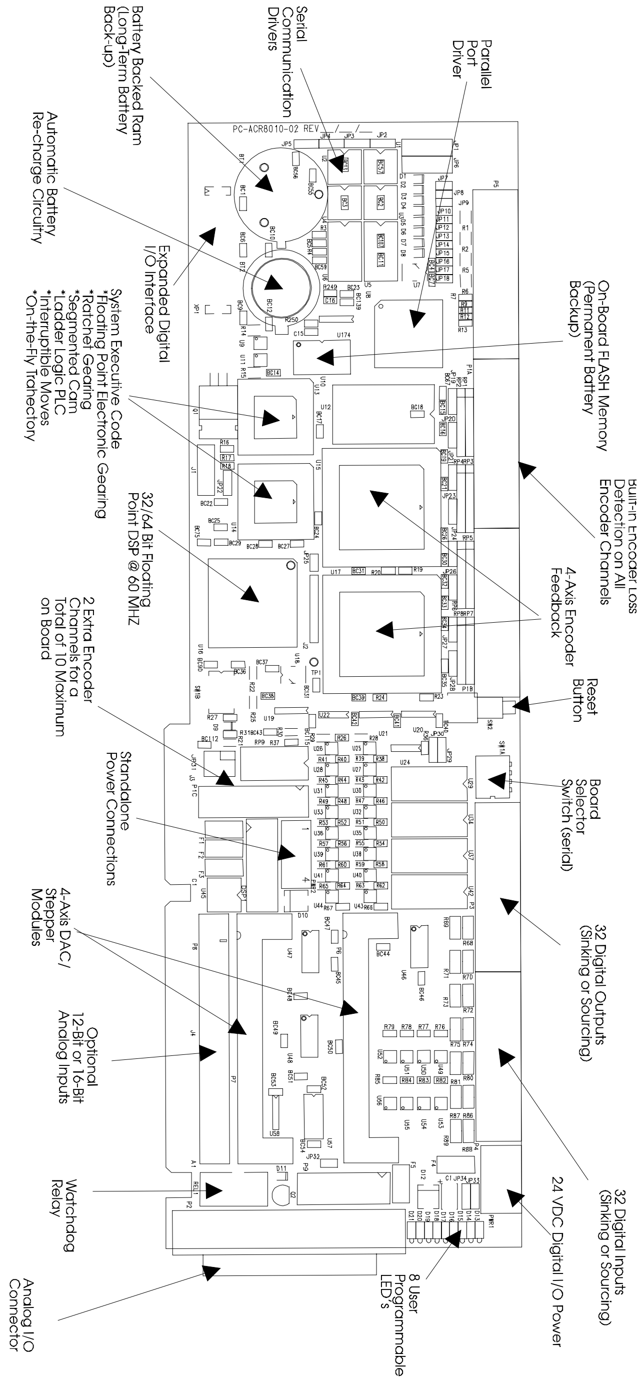
ACR8000 Daughterboard Hardware Overview (Front of ACR8000 Daughterboard)



ACR8000 Motherboard Hardware Overview (Front of ACR8000 Motherboard)



ACR8010 Hardware Overview (Front of ACR8010)



CONTROLLER FIRMWARE

Firmware Overview

The Acroloop controllers have the most complete and intuitive command set available on the market. The native firmware language is AcroBASIC. AcroBASIC is an intuitive language similar to the BASIC programming language.

The firmware is designed to work as a complete standalone package or in conjunction with a host processor (PC-BUS). Of course, with the processing power available on the controllers, a typical application would involve the host processor simply sending general commands to the card and updating the operator interface display. **Acroloop supports the DOS, Windows, and Windows NT operating systems.** The Acroloop controller family can be provided with the operating system drivers. **Acroloop also supports C++, Visual C, and Visual Basic software libraries to assist in applications where the user customizes the operator interface (See Development Tools Section).**

The following is an outline of AcroBASIC. AcroBASIC commands can be executed in either MDI (Manual Data Input) or program mode (from within a stored program). In the MDI mode, the commands will be executed immediately, while in the program mode the commands are stored in memory as they are entered. A combination of MDI mode and program mode is typically used in PC-Bus applications while only program mode is used in standalone applications. Any command preceded by a valid line number will be stored in memory for the selected program (Line numbers 1 to 65000).

There are two sets of memory types that are accessible by each program:

1. Global System Parameters
    a. Global System Variables (64 bit floating point or integers)
    b. Global Bit Flags
2. Local User Parameters
    a. Local Long (32 bit integer) Variables
    b. Local Long Arrays (user dimensionable)
    c. Local Double (64 bit floating point) Variables
    d. Local Double Arrays (user dimensionable)
    e. Local String (packed 8 bit) Variables
    f. Local String Arrays (user dimensionable)

Global system parameters can be accessed by any program space. Each program can dimension (allocate) local parameters. The Global user parameters are referenced by P0 to P4095 and can be used in any program by the user.

Firmware Overview (continued)

Local variables are referenced by their type, followed by a number. Local arrays are referenced by an array number and index.

LVnn = Local Long (32 bit integer) Variable
DVnn = Local Double (64 bit floating point) Variable
SVnn = Local Floating Point Variable
SAnn(index) = Local Floating Point Array
LAnn(index) = Local Long Array
DAnn(index) = Double Long Array
$Vnn = String Variable
$Ann(index) = String Array
Where nn is an arbitrary number

Only the amount of memory limits how many variables can be used. The Floating Point Math can be used on both global and local variables. Thus, the controllers are extremely flexible from a programming standpoint. The Acroloop controllers allow easy access to information through the use of unlimited variables, arrays, and the use of strings. The controllers also allow access to virtually every motion control parameter (see Appendix in this brochure).

Firmware Structure

The firmware is structured into groups. Each group consists of similar commands. The AcroBASIC firmware groups are:

1. Operating System Group
2. Program Control Group
3. Servo Control Group
4. Feedback Control Group
5. Setpoint Control Group
6. Axis Limits Group
7. Velocity Profile Group
8. Interpolation Group
9. Transformation Group
10. Logic Function Group
11. Program Flow Group
12. Memory Control Group
13. Character I/O Group
14. Global Objects Group
15. Non-Volatile Group

Main Firmware Groups

Each of the groups is discussed in detail in the Acroloop controller family programming manual. The important aspects of the controllers are:

1. Allows complete standalone operation.
2. Provides processing power to increase throughput, accuracy, and precision for PC-BUS applications. At the same time, minimizes host processor operations, thus reducing overhead to insure real time execution.
3. Provides flexibility for virtually any motion control application.

The main firmware groups are outlined on the next pages.

Main Firmware Group Definitions

Servo Control Group - The controllers are provided with full PID with Velocity and Acceleration Feed Forward control. In addition, controllers have built-in wave shaping within the control algorithm. This feature combined with the Graphical Tuning available with any Acroloop software package allows fine tuning of axes. The processing speed of the Acroloop controllers combined with the servo control algorithm provides positioning error accuracy typically within 1 encoder pulse.

The command set for the Servo Control Group includes:

PGAIN - Proportions output as a percentage from the desired setpoint.
IGAIN - Reduces steady state errors evaluating the error terms in closed loop control.
IDELAY - Delays the calculation of the error additions.
ILIMIT - Automatic “anti-windup” control.
DGAIN - Conditions the servo by analyzing the rate of change from desired setpoint.
DWIDTH - Sets the derivative bandwidth.
FFVEL - Reduces following error caused by differential gains.
FFACC - Reduces following errors caused by inertia.
NOTCH - Uses first bi-quad filter to implement a notch filter.
LOPASS - Uses second bi-quad filter to implement a lopass filter.
DUAL BI-QUAD FILTERS - Allows user to custom set poles and zero’s to implement specific filters for the application.
TORQUE LIMIT - Sets the torque limit output value for the servo loop.

The Acroloop controller family has the capability of programming the analog output signals. The analog output signals can be used to drive other “non-servo” devices. A simple example would be to implement a cyclic analog output signal to drive a thermoformer for a bag making machine. The thermoformer is used to preheat, heat, and cool the seal on the bags. The analog output signal is used to directly drive the heating device. A second example is driving a spindle on a CNC milling machine.

The Acroloop controller family can be provided with a stepper output. The stepper output frequency ranges up to 4MHz. The pulse width range for the stepper output is 125 nanoseconds to 16 microseconds. See **Ordering Matrix** for additional information.

Main Firmware Group Definitions (continued)

Setpoint Control Group - The setpoint control group provides the necessary tools in order to precisely control the outputs to the system. The ACR8000 provides the easiest, yet the most sophisticated, form of setpoint controls using built-in user definable arrays. The arrays are completely programmable, segmentable and linkable so that the resolution can be as coarse or as fine as desired.

The command set for the Setpoint Control Group includes:

- **Jog** - Independent Incremental and Absolute jogging modes and commands.
- **Floating Point Electronic Gearing**- Allows axes to be ratioed to a source encoder.
- **Handwheel** - Manual control from a source encoder input.
- **Segmented Electronic CAM** – Position and velocity matching table mechanism.
- **Ballscrew Compensation** - Compensation can be defined as coarse or fine as desired. User definable arrays allow segmentation of values. Includes built-in interpolation.
- **Backlash Compensation** - Compensates for errors introduced by hysteresis in mechanical gearboxes or rack and pinion systems.
- **Ratchet Mode** - Position or velocity follower mode; allows external time base to be set to servo clock, encoder register, or hardware parameter. 24 programmable ratchet modes allows treating of incoming pulses as normal, ignore, negate, or buffer the pulses.

Velocity Profile Group - The controller's Velocity Profile Group allows precise control of the servo control loop.

The command set for the Velocity Profile Group includes:

- **Velocity** - Sets the target velocity.
- **Acceleration** - Sets the trapezoidal acceleration ramp.
- **Deceleration** - Sets the trapezoidal deceleration ramp.
- **S-Curve** - Non-linear acceleration and deceleration motion profiles.
- **Feedrate Override** - On-the-fly feedrate override.
- **Automatic Vector Definition** - Defines vector calculations for axis combinations
- **Vector** - Manual override of automatic vector distance calculation.
- **Velocity/Acceleration/Jerk Limits** - Automatic profiling for multiple axes.
- **Virtual Master** - Master/Slaving to a simulated master internal to the controller.
- **Rotary Axis** - Automatic MARKER rollover for rotary axis application
- **Timebase -** Programmable source for external timebase for coordinated motion

PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

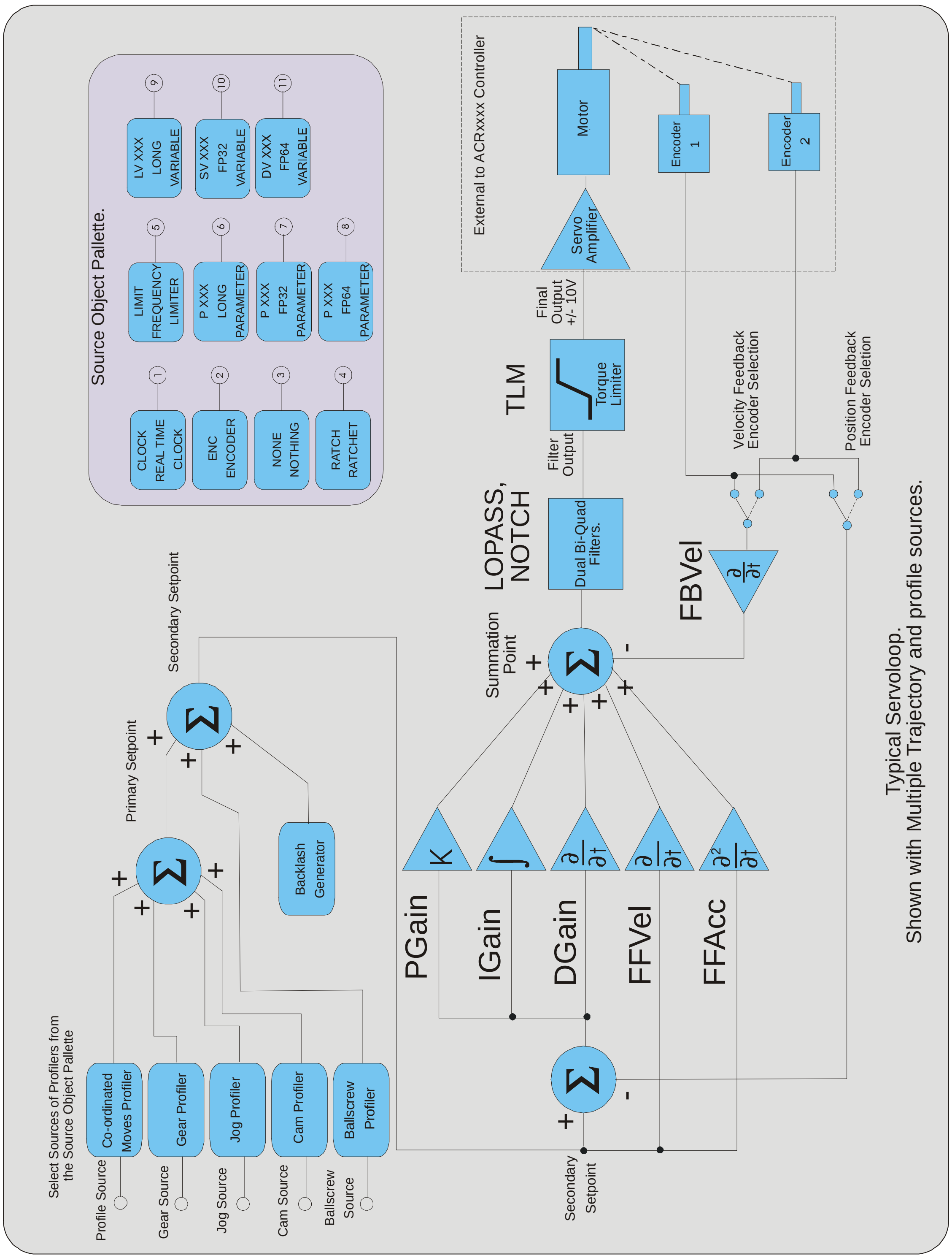
Servo Loop Design

Servo Loop Design - The Acroloop motion controller has several key features that the firmware implements. The features are:

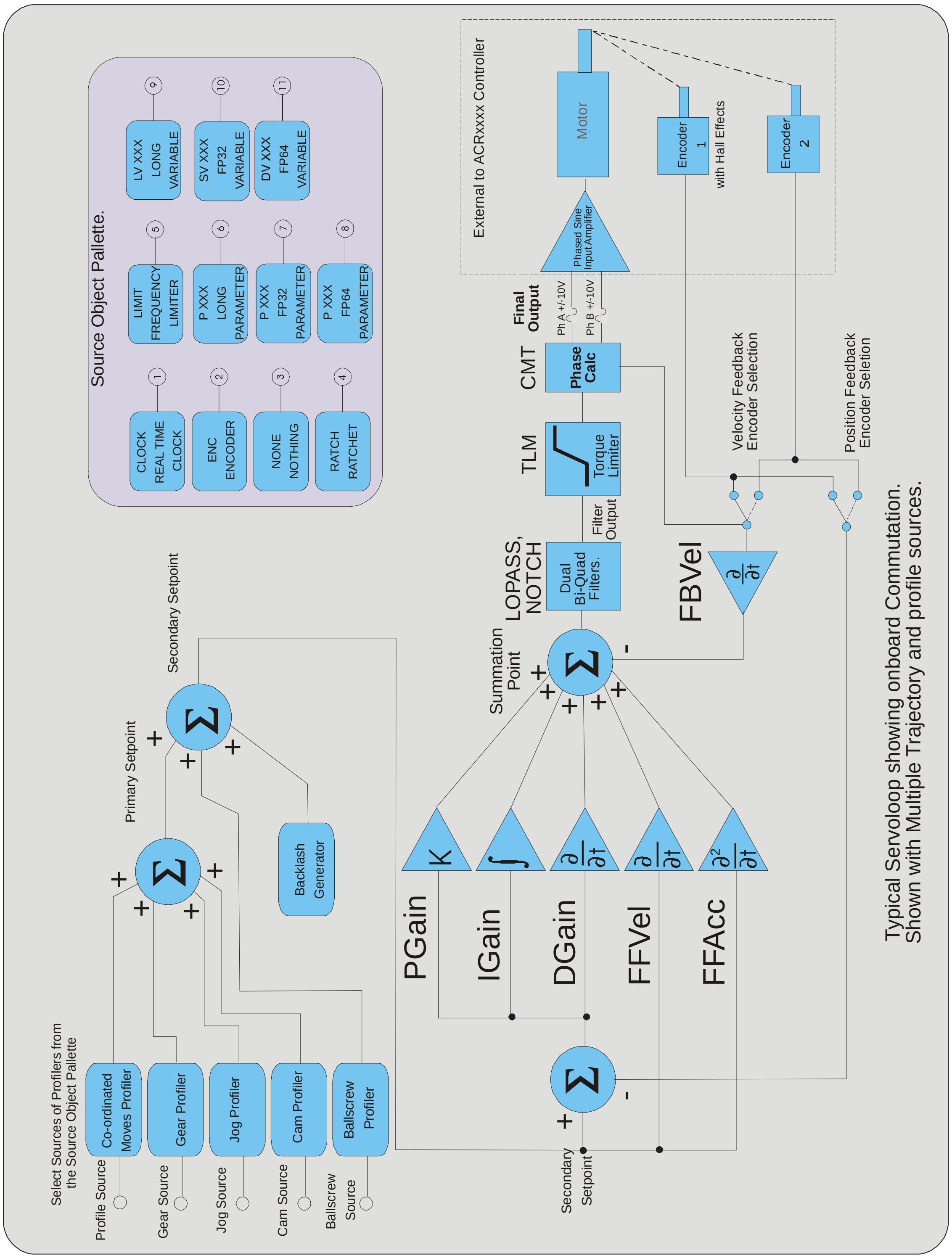
1. Servo Loop
2. Setpoint Summation
3. Servo Loop Core
4. Digital Filters

The following diagrams show a typical Acroloop motion controller servo loop with multiple trajectory and profile sources. The first diagram shows the standard +/- 10V final output available on all of the Acroloop Motion Control boards; the second diagram shows the on-board commutation output available on the ACR1200, ACR1500, ACR2000, and ACR8010 boards.

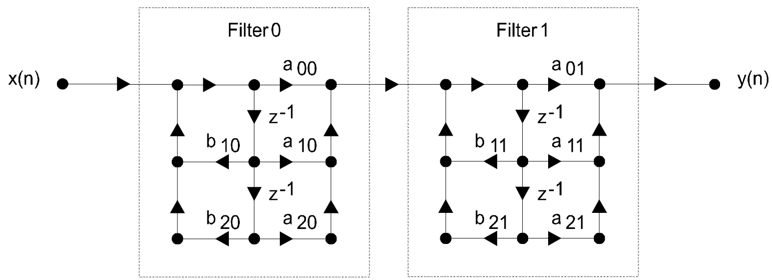
Servo Loop Design (continued)



Servo Loop Design (continued)



The Acroloop motion controllers are unique in that user specified poles and zeros can be implemented in the servo loop to customize the digital filter to the application.



$$H(z) = \left(\frac{a_{00} + a_{10} z^{-1} + a_{20} z^{-2}}{1 - b_{10} z^{-1} - b_{20} z^{-2}} \right) \left(\frac{a_{01} + a_{11} z^{-1} + a_{21} z^{-2}}{1 - b_{11} z^{-1} - b_{21} z^{-2}} \right)$$

Figure - Digital Filter Equations

The digital filters are directly accessible by the user through on-board built-in hardware parameters and flags. The parameters and flags are listed below.

Filter Parameters:

Filter 0	Axis Number							
	0	1	2	3	4	5	6	7
b2 Coefficient	12336	12592	12848	13104	13360	13616	13872	14128
a2 Coefficient	12337	12593	12849	13105	13361	13617	13873	14129
b1 Coefficient	12338	12594	12850	13106	13362	13618	13874	14130
a1 Coefficient	12339	12595	12851	13107	13363	13619	13875	14131
a0 Coefficient	12340	12596	12852	13108	13364	13620	13876	14132

Filter 1	Axis Number							
	0	1	2	3	4	5	6	7
b2 Coefficient	12341	12597	12853	13109	13365	13621	13877	14133
a2 Coefficient	12342	12598	12854	13110	13366	13622	13878	14134
b1 Coefficient	12343	12599	12855	13111	13367	13623	13879	14135
a1 Coefficient	12344	12600	12856	13112	13368	13624	13880	14136
a0 Coefficient	12345	12601	12857	13113	13369	13625	13881	14137

Filter Flags:

Control	Axis Number							
	0	1	2	3	4	5	6	7
Filter Activate	786	818	850	882	914	946	978	1010

Firmware Commands

Controller Family Command List (Version 1.12)

Operating System Group
ATTACH – Setup Axis Configuration
CPU – Display processor loading
DETACH – Cancel attachments
DIAG – Display system diagnostics
ECHO – Control character echoing
HELP – Display command list
MODE – Binary data formatting
PERIOD – Set base system timer period
PROG – Switch to a program prompt
SYS – Return to system prompt

Feedback Control Group
INTCAP – Encoder position capture
MSEEK – Marker seek operation
MULT - Set encoder multipliers
PPU - Set axis pulse/unit ratio
REN – Match position with encoder
RES – Reset or preload encoders
RES – Reset or preload encoders
ROTARY – Set rotary axis length

Setpoint Control Group
BKL - Set Backlash Compensation
BSC - Set Ballscrew Compensation
BSC DIM – Dimension memory
BSC SEG - Define ballscrew segments
BSC SRC - Redefine ballscrew source
BSC ON – Enable ballscrew compensation
BSC OFF – Disable ballscrew
BSC SCALE - Set ballscrew output scaling
BSC OFFSET - Set ballscrew output offset
BSC FLZ – Set ballscrew input offset
CAM – Electronic CAM
CAM DIM – Dimension memory
CAM SEG - Define cam segments
CAM SRC - Redefine cam source
CAM ON – Enable cam compensation
CAM OFF - Disable cam compensation
CAM SCALE - Set cam output scaling
CAM OFFSET - Set cam output offset
CAM FLZ – Set cam input offset
CAM RES - Transfer cam offset
CAM SHIFT - Set incremental cam shift

Program Control Group
HALT - Halt an executing program
LIST - List a stored program
LRUN - Run and listen to a program
NEW - Clear out a stored program
RUN - Run a stored program

Servo Control Group
DGAIN - Set derivative gain
DWIDTH - Set derivative sample period
FFACC - Set feedforward acceleration
FFVEL - Set feedforward velocity
IDELAY - Set integral time-out delay
IGAIN - Set integral gain
ILIMIT - Set integral anti-windup limit
LOPASS - Setup lopass filter
NOTCH - Setup notch filter
PGAIN - Set proportional gain

Axis Limits Group
ALM - Set Axis Limits A
BLM - Set Axis Limits B
EXC - Set excess error band
IPB - Set in-position band
ITB - Set in-torque band
TLM - Set torque limit

Velocity Profile Group
ACC - Set acceleration profile
DEC - Set deceleration profile
F - Set velocity in units/minute
FOV - Set feedrate override
FVEL - Set final velocity
IVEL - Set initial velocity
STP - Set stop ramp
VECDEF - Define automatic vector
VECTOR - Set manual vector
VEL - Set target velocity for a move

Interpolation Group
INT – Interruptible move
MOV - Define a linear move
SINE - Sinusoidal move
TRJ - Start new trajectory

PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

Controller Family Command List (Version 1.12, continued)

Setpoint Control Group(continued)
GEAR – Electronic Gearing
GEAR SRC -Set electronic gearing source
GEAR PPU - Scale electronic gearing
GEAR RATIO -Set electronic gearing ratio
GEAR RES - Reset or preload gearing
GEAR ON - Turn electronic gearing on
GEAR OFF - Turn electronic gearing off
HDW – Handwheel
JOG – Single axis velocity profile
JOG VEL – Set jog velocity
JOG ABS – Jog to absolute position
JOG ACC - Set jog acceleration
JOG DEC - Set jog deceleration
JOG INC – Jog an incremental distance
JOG RES - Reset or preload jog offset
JOG REN - Transfer current position
JOG FWD - Jog axis forward
JOG REV – Jog axis backward
JOG OFF – Stop jogging axis
JLM - Set jog limits

Memory Control Group
CLEAR – Clear dimension allocation
DIM – Allocate memory
MEM – Display program memory

Character I/O Group
CLOSE – Close a device
INPUT – Receive data from a device
LISTEN – Listen to program output
OPEN – Open a device
PRINT – Send data to a device

Non-Volatile Group
BRESET – Disable Battery Reset
ELOAD – Update from EEPROM
ERASE – Erase the EEPROM
ESAVE – Update to EEPROM
PBOOT – Auto-run a program
PROM – Dump burner image

Transformation Group
FLZ - Relative program path shift
OFFSET – Absolute program path shift
ROTATE - Rotate a programmed path
SCALE - Scale a programmed path

Logic Function Group
CLR - Clear a bit flag
DWL - Delay for a given period
IHPOS - Inhibit on position
INH - Inhibit on a bit high or low
SET - Set a bit flag
TRG - Start move on trigger

Program Flow Group
END - End of program execution
GOSUB - Branch to a subroutine
GOTO - Branch to a new line number
IF/THEN – Conditional execution
REM - Program comment
RETURN - Return from a subroutine

Global Objects Group
ADC POS - Select positive channel
ADC NEG – Select negative channel
ADC GAIN - Seta analog input gain
ADC OFFSET – Set analog input offset
ADC ON - Enable ADC update
ADC OFF – Disable ADC update
AXIS - Direct Axis Manipulation
DAC - Direct DAC manipulation
ENC - Direct Encoder manipulation
MASTER - Direct master manipulation
PLS – Programmable Limit Switch
PLS SRC - Set PLS source pointer
PLS DST - Set PLS destination pointer
PLS BASE - Set PLS array pointer
PLS RES - Reset internal counter
PLS ROTARY – Set PLS rotary length
PLS FLZ - Set PLS index offset
PLS MASK - Set PLS output bit mask
RATIO - Set PLS scaling ratio
PLS ON - Enable PLS update
PLS OFF – Disable PLS update
SAMP - Data sampling
SAMP SRC - Set sample source
SAMP BASE - Set sample base
SAMP CLEAR – Clear sample channels
SAMP TRG - Set sample trigger

Controller Family Command List (Version 1.12, continued)

Expressions/String Handling
ASC - ASCII value
BIT - Bit flag status
CHR$ - Character string
GETCH – Wait for a character
INKEY$ - Return a character
INSTR – String search
KBHIT – Check for waiting character
LCASE$ - Convert to lower case
LEFT$ - Left string
LEN - String length
MID$ - Middle string
RIGHT$ - Right string
SPACE$ - String of spaces
STR$ - Convert to numeric string
STRING$ - String of characters
UCASE$ - Convert to upper case
VAL – Convert string to numeric
CEIL – smallest integer >= expression
FLOOR – Largest integer <= expression
MOD – Modulus
RND – Random integer
TRUNC – Remove fractional part

Basic PLC Command Set
PLC - Switch to PLC program
PON - Turn on PLC scanning
POFF - Turn off PLC scanning
RUN - Run PLC program
HALT - Halt PLC program
LIST - List PLC program
MEM - Show PLC memory
PBOOT - PLC on power-up
LEFT$ - Left string

PLC Logic Commands
LOAD (index)
LOAD NOT (index)
AND (index)
AND NOT (index)
OR (index)
OR NOT (index)
AND LOAD (index)
OR LOAD (index)
OUT (index)

PLC Timer Commands
LOAD TIMER (index)
LOAD NOT TIMER (index)
AND TIMER (index)
AND NOT TIMER (index)
OR TIMER (index)
OR NOT TIMER (index)
TIMER (index) {preload}

PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

Controller Family Command List (Version 1.12, continued

PLC Counter Commands
LOAD COUNT (index)
LOAD NOT COUNT (index)
AND COUNT (index)
AND NOT COUNT (index)
OR COUNT (index)
OR NOT COUNT (index)
COUNT (index) {preload}

PLC Latch Commands
LOAD KR (index)
LOAD NOT KR (index)
AND KR (index)
AND NOT KR (index)
OR KR (index)
OR NOT KR (index)
KR (index)

Version 1.13 Commands
LOCK - Redirect primary setpoint (multiple dual axis control; i.e. X, X')
UNLOCK - Release primary setpoint redirection
AUT - Turn off block mode
BLK - Turn on block mode
PAUSE - Activate pause mode for program
RESUME - Resume program
TRON - Turn on trace mode(displays line number of program as it is executed)
TROFF - Turn off trace mode

Version 1.14 Commands
ATTACH AXIS axis ADC adc DAC dac - Analog feedback to close loop
CAM SRC - CAM source set to external timebase
CONFIG - Setup outputs configuration
GEAR SRC - GEAR source set to external timebase
GEAR ACC - Set gearing acceleration
GEAR DEC - Set gearing deceleration
JOG SRC - JOG source set to external timebase
PLS SRC - PLS source set to external timebase
RATCH SRC - Define ratchet source
RATCH MODE - Normal, ignore, negate, or buffer incoming source pulses
SRC - Select external timebase: NONE, servo clock, encoder, ratchet, or parameter
STEPPER GAIN - Set stepper output gain

Version 1.15 – 1.17 Commands
SCURVE - Define S-Curve velocity profiling capabilities
DUAL ENCODER FEEDBACK - Command set for position and velocity loops
HSINT - High speed interrupt for labeling, packaging, and bag making applications
ADC FEEDBACK – Closed loop analog feedback capability
FLASH – Set of flash memory commands for ACR2000
ROV – Simultaneous feedrate override and rapid override support
VER – Displays the card number and firmware version

Version 1.18 Commands
CMT – Command set to define sinusoidal/trapezoidal commutation
SYNC – Used to synchronize the moves of the masters
TMOV – Sets the time in seconds in which the moves will be completed

COMMUNICATIONS

Introduction

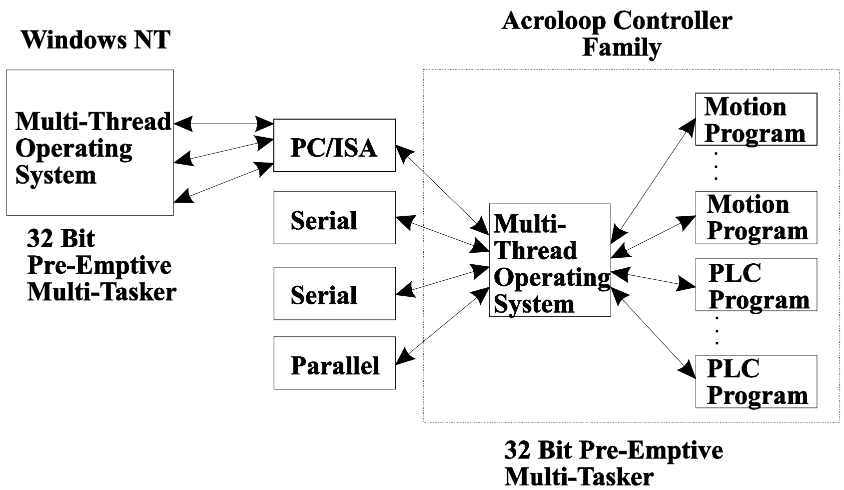
The ACR2000, ACR8000, and the ACR8010 are the only motion controllers available on the market with 4 communications ports. (The ACR2000 is a PC-Bus card with an optional communications card for serial communications sold separately). The ACR1200 is a standalone only design. The ACR1500 is a PC-Bus only design. Due to the unique architecture of the Acroloop controller family and supporting hardware, the controllers can receive and transmit data over any port simultaneously.

Communications Ports
Serial Ports - 2 serial ports as standard (COM1, COM2).
Parallel Port - 1 parallel port to communicate to another host or a printer (LPT).
PC-Bus - 2 high speed on-board FIFO's transfer data over the PC/ISA Bus.

The Acroloop controllers are also capable of **handling strings over any of the communications ports. The controllers can identify characters, search strings, handle upper and lower case strings simultaneously, identify lengths, and convert to and from ASCII code on-the-fly.**

Communication Channel Design

There are 4 communications ports available on the ACR2000, ACR8000, and the ACR8010. There are three ports available on the ACR1200 – 2 serial ports and 1 parallel port. There is one port available on the ACR1500, which is the PC-Bus port. Please reference the following example for a Windows NT PC-based system.



Communications Ports Table

Port	Interface
COM1	Serial RS-232, RS-422
COM2	Serial RS-232, RS-422
FIFO	PC/ISA bus
LPT	Parallel port access

Communication Channel Design (continued)

All of the ports can be operated simultaneously and attached to different programs. Programs can be running while others are being edited. All of the ports will “wake up” on power-up when data is detected. The serial ports have automatic baud detection at a maximum baud rate of 38.4K. The automatic baud detection is triggered by receiving two carriage returns (ASCII 13) after power-up.

The parallel port “LPT” is a high performance bi-directional port with high speed FIFO’s. The FIFO will function as a buffer and is very important for BURST sending and receiving. The effective baud rate is 152Kb - 300Kb. The parallel port will transfer data significantly faster than the maximum baud rate of the serial ports.

The communications protocol is identical for all of the ports (PC-BUS, COM1, COM2, and LPT). Thus, communications support software will function over each of the ports.

Communications ports are either at the “system” level or at a “program” level. The command prompt indicates the level of the communication channel.

System Level - The system level is the level that a communication channel is at upon power-up. The command prompt at this level is as follows:

SYS>

Only a limited set of commands can be issued from this level. From any other level, the SYS command will return the communication channel to the system level.

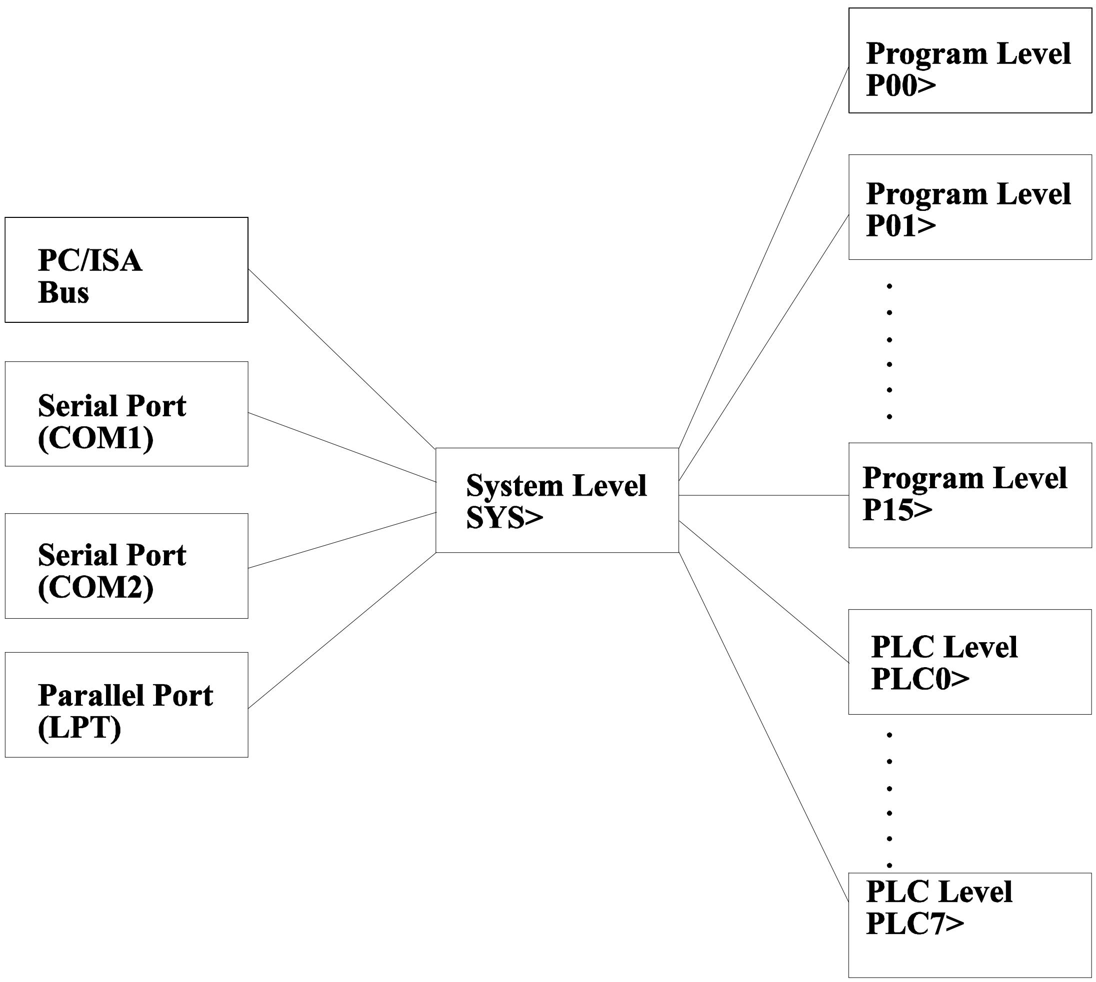
Program Level - The program level allows the editing and running of individual programs. The command prompt at the program level is as follows:

Pnn>

Where “nn” is the current active program number.

Communication Channel Design (continued)

The diagram below shows the communication channels for the ACR2000, ACR8000, and the ACR8010. The ACR1200 has two (2) serial ports and one (1) parallel port for access to communications. The ACR1500 has a single PC/ISA-Bus port for access to the communications. The advantage to having multiple ports is for multiple communications in an application and for development purposes. The advantage to a single PC-Bus port is cost effectiveness. The communication channels can be active simultaneously and operating at different levels. For example, COM1 could be editing program 3 while COM2 monitors user variables being modified by program 5. With the flexible communications structure, the controllers can also be used in a wide variety of applications where communications are made to multiple devices.



Communication Channel Design

LADDER LOGIC PLC

Ladder Logic PLC

The Acroloop controllers are provided with an on-board PLC and a free graphical PLC programming tool called AcroVIEW (see AcroVIEW section). Unlike other motion controllers that control I/O with slow high level code, the Acroloop PLC is programmed using standard logic commands compiled to machine code on-the-fly. This allows PLC control capability for external electrical devices to be operated in conjunction with normal motion programs without sacrificing PLC performance. By using the on-board PLC, high speed digital I/O execution can be accomplished in real time (i.e. limit switches, E-stop, feedhold, etc.) while slow PLC operation can be handled in the one of two areas:

- Remaining PLC spaces
- Other motion or non-motion program spaces

The controller's built-in PLC has the following features:

- **Developed graphically using the free AcroVIEW programming utility (see AcroVIEW section in this brochure).**
- Optically-isolated digital I/O (TTL I/O on ACR1500)
- Multi-tasking allows separate PLC programs simultaneously.
- Up to 8 PLC programs available (PLC0 - PLC7)
- 100 PLC instructions executed in 1.5 millisecond scan rate.
- Full 800 instructions/8 timers/8 counters executed 5 millisecond scan rate.
- PLC is automatically compiled to low level code on-the-fly.
- 8 built-in global PLC timers.
- 8 built-in global PLC counters.

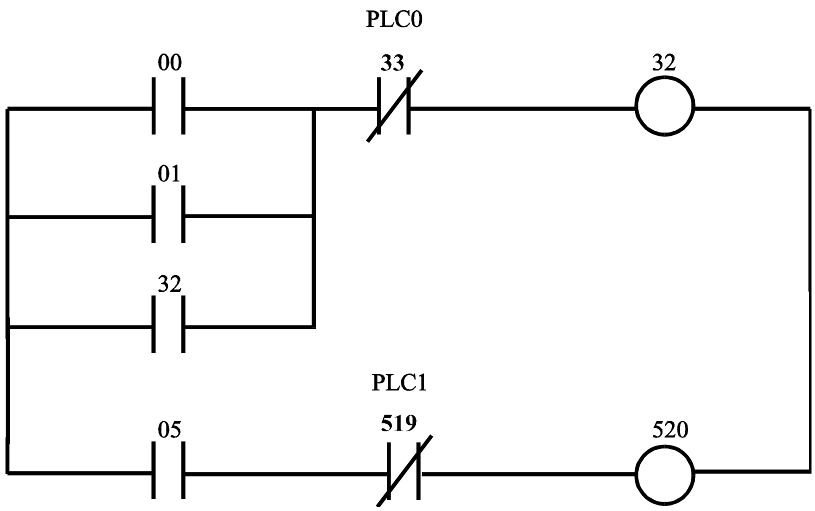
As previously mentioned, the PLC is programmed using standard PLC Logic commands. The Acroloop controller family firmware also implements Timer commands, Counter commands, and Latch commands, so that a complete PLC program can be executed by the appropriate controller. The following outline illustrates the standard PLC commands available in the Acroloop family of motion controllers.

PLC Logic Commands
LOAD (index)
LOAD NOT (index)
AND (index)
AND NOT (index)
OR (index)
OR NOT (index)
AND LOAD (index)
OR LOAD (index)
OUT (index)

PLC Timer Commands
LOAD TIMER (index)
LOAD NOT TIMER (index)
AND TIMER (index)
AND NOT TIMER (index)
OR TIMER (index)
OR NOT TIMER (index)
TIMER (index) {preload}

PLC Counter Commands	**PLC Latch Commands**
LOAD COUNT (index)	LOAD KR (index)
LOAD NOT COUNT (index)	LOAD NOT KR (index)
AND COUNT (index)	AND KR (index)
AND NOT COUNT (index)	AND NOT KR (index)
OR COUNT (index)	OR KR (index)
OR NOT COUNT (index)	OR NOT KR (index)
COUNT (index) {preload}	KR (index)

PLC Program Example



The graphical portion of the PLC program can be developed using the AcroVIEW programming utility. AcroVIEW also allows the program to be displayed in the TEXT view. The TEXT or AcroBASIC is the code that is downloaded to the Acroloop controller for execution and is compiled on-the-fly for deterministic execution times.

Command	Description
PLC0	Select PLC program 0.
10 LD 00	Load Output number 00
20 OR 01	Both input 00 AND input 01
30 OR 32	Load output number 02
40 AND NOT 33	Conditionally, output 03 must be closed
50 OUT 32	Tie both blocks together
	Either block, then set output 32.

Command	Description
PLC1	Select PLC program 1
10 LD 05	Detect feedhold push-button on input 4
20 AND NOT 519	And if not in feedhold (see sample flags in Appendix)
30 OUT 520	Set Master Profiler 0 in feedhold (see sample flags in Appendix)

Programming Notes: The ACR8000 and ACR8010 are provided with 64 optically isolated digital I/O. The inputs are numbered 0 through 31. The outputs are numbered 32-63. The ACR1200 and ACR2000 are provided with 32 digital I/O (16 inputs and 16 outputs standard). The ACR1500 is provided with 48 TTL digital I/O compatible with standard 50 pin optical-isolation boards.

PLC Operation

PLC programs are created in the same manner as user programs, but with a limited instruction set that is compiled into machine code for high-speed execution. Each PLC program can contain a maximum of 100 instructions. Memory for the PLC programs must be dimensioned from the system level using the DIM PLC command. On average, dimensioning 32 bytes per PLC instruction is sufficient.

PLC programs are linked into the PLC scanner, which is a list of events that are to be executed at the servo interrupt rate. During each servo interrupt, a single event from this list is executed. During the next interrupt, the next event is executed. This process is repeated after the last item in the list. In addition to PLC programs, the scanner event list also contains an input/output/clock scan and a timer/counter/latch scan.

As an example, two PLC programs running at the ACR1200/ACR2000/ACR8000/ACR8010 default 500 microsecond servo rate will be serviced every 2 milliseconds. One interrupt for the input/output/clock scan, one interrupt for the timer/counter/latch scan, and one interrupt for each of the PLC program scans. All eight PLC programs would be scanned every 5 milliseconds.

PLC Operation - Related Parameters:

Individual PLC programs may also be instructed to scan at a rate slower than the servo interrupt rate by setting system parameters. The "tick preload" parameter controls the scan rate in milliseconds and the "tick count" indicates the number of milliseconds remaining before the PLC program is scanned.

The following table outlines parameters related to PLC operation.

PLC Tick Parameters

PLC Number	Tick Preload	Tick Count
0	P6656	P6657
1	P6672	P6673
2	P6688	P6689
3	P6704	P6705
4	P6720	P6721
5	P6736	P6737
6	P6752	P6753
7	P6768	P6769

PLC Operation - Related Flags

The "PLC Running" flag is set when the RUN command is executed and cleared when the HALT command is executed. The "First PLC Scan" flag is set when a PLC program is RUN and cleared after the first PLC scan. The contact of this relay can be used for PLC reset logic as needed. The "RUN Request" and "HALT Request" flags cause the execution of RUN and HALT commands respectively.

The following table outlines bit flags related to PLC operation.

PLC Operation Flags

PLC Number	PLC Running	First PLC Scan	Run Request	Halt Request
0	BIT1536	BIT1537	BIT1538	BIT1539
1	BIT1568	BIT1569	BIT1570	BIT1571
2	BIT1600	BIT1601	BIT1602	BIT1603
3	BIT1632	BIT1633	BIT1634	BIT1635
4	BIT1664	BIT1665	BIT1666	BIT1667
5	BIT1696	BIT1697	BIT1698	BIT1699
6	BIT1728	BIT1729	BIT1730	BIT1731
7	BIT1760	BIT1761	BIT1762	BIT1763

PLC Operation Commands

PLC commands control the operation of PLC programs. The PLC, PON, and POFF commands can be executed from any prompt. The RUN, HALT, LIST, and MEM commands are similar to their user program counterparts, but they act slightly different when executed from the PLC prompt.

PLC Operation Command List

The following is a list of commands related to PLC programming:

PLC	Switch to PLC program
PON	Turn on PLC scanning
POFF	Turn off PLC scanning
RUN	Run PLC program
HALT	Halt PLC program
LIST	List PLC program
MEM	Show PLC memory

PROGRAMMABLE LIMIT SWITCH

The Acroloop Controller family is provided with 8 on-board Programmable Limit Switches (PLS0 - PLS7). **The PLC and PLS commands are designed to solve two different problems in the control of automated machinery.**

The PLC provides a programmable method of relating sensors, limit switches, selector switches, and the on-board “flags” or “variables (logic 0 or 1), and other input devices to output components such as solenoid valves, pneumatics, pop-up pins, emergency off circuitry, feedhold, cycle start, feedrate override buttons, and other electrical devices. Using the PLC, a programmer can easily change the way a machine operates by modifying the PLC’s logic program. There is no need to physically re-wire the machine’s components.

In contrast, the PLS command is designed to perform accurate control of repetitive high-speed machine functions. Because these functions are often correlated with a rotating shaft, the PLS is designed to turn ON and OFF based on a rotary position signal generated by a device (typically a differential encoder).

Many industrial control systems can benefit from including both a PLC and a PLS. The PLC coordinates low-speed processes while the PLS directly controls high-speed machine functions. In analyzing which machine processes are best handled by PLC’s or PLS’s, an understanding of scan time is important.

PLS/PLC Scan Time Table

Command	Scan Time
PLC	1.5 msec - 5 msec scan time
PLS	200 to 500 microseconds (every interrupt)

The PLS will update the digital outputs every interrupt. The interrupt of the controller is programmable using the PERIOD command. Thus, if the default interrupt is used (500 microseconds), the I/O will be updated every 0.5 msec.

Application Example:

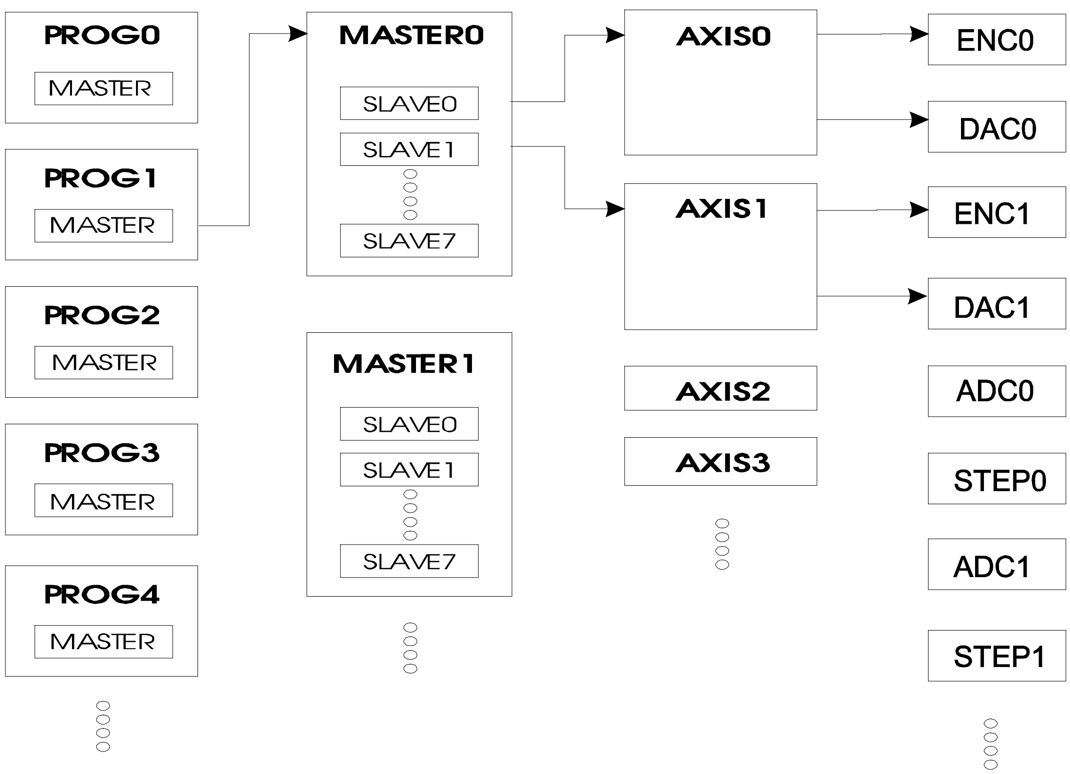
PLS is used in applications where inputs and outputs are triggered based on a rotary axis position. Examples are repetitive firing of batches of outputs for
1) Glue Guns or 2) Automated Paint Gun systems.

Please see the PLS programming example for additional information.

COORDINATE SYSTEMS

The Acroloop controller family is capable of handling up to 8 coordinate systems directly on the motion controller board. The coordinate systems are referred to as MASTER's. The **MASTER** is nothing more than a profiler. Up to 8 MASTER's are available. Each MASTER can have up to 8 SLAVE's. The **SLAVE** is dependent on the MASTER for the profile velocity, acceleration, and deceleration parameters. Each SLAVE can be setup to handle feedback and output capability for an AXIS. The feedback capability is **ENCODER's, or ANALOG feedback to close the loop**. **The Acroloop controllers support dual loop applications for separate position and velocity loops. The position and velocity loops can be either digital or analog feedback. The loop can also be an OPEN LOOP for STEPPER's**. The output of the AXIS could be ANALOG (DAC, +/-10VDC) or STEPPER output (pulse and direction).

The following diagram illustrates the overview of the system attachments for the Acroloop controllers. All of the controllers program identically.



Program Attachments

PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

The following program attachments are sample valid attachment statements that can be made with the Acroloop controller family.

```
PROG1                              REM:  Program number 1.
HALT                               REM:  Halt execution (Good practice).
DETACH                             REM:  Cancel any previous attachments.
ATTACH MASTER0                     REM:  Setup profiler 0.
ATTACH SLAVE0 AXIS0 “X”            REM:  Attach the X-axis.
ATTACH SLAVE1 AXIS1 “Y”            REM:  Attach the Y-axis
ATTACH AXIS0 ENC0 DAC0             REM:  Attach encoder and analog to axis0.
ATTACH AXIS1 ENC1 DAC1             REM:  Attach encoder and analog to axis1.

ATTACH MASTER1                     REM:  Setup profiler 1.
ATTACH SLAVE0 AXIS2 “LOAD”         REM:  Attach the Load axis to profiler 1.
ATTACH AXIS2 ADC0 DAC2             REM:  Attach analog feedback and output.


ATTACH MASTER2                     REM:  Setup profiler 2.
ATTACH SLAVE0 AXIS3 “STEP”         REM:  Setup the STEP axis.
ATTACH AXIS3 STEPPER0 STEPPER0     REM:  Setup up OPEN LOOP stepper


ATTACH MASTER3                     REM:  Setup profiler 3.
ATTACH SLAVE0 AXIS4 “PICK”         REM:  Setup the PICK axis.
ATTACH AXIS4 ENC2 STEPPER1         REM:  Setup a CLOSED LOOP stepper

ATTACH MASTER4                     REM:  Setup profiler 4.
ATTACH SLAVE0 AXIS0 “X”            REM:  Setup the X-axis
ATTACH SLAVE1 AXIS1 “Y”            REM:  Setup the Y-axis
ATTACH AXIS0 ENC0 DAC0 ENC1        REM:  Setup dual feedback for X-axis
ATTACH AXIS1 ENC2 DAC1 ENC3        REM:  Setup dual feedback for Y-axis
```

In general, any combination of encoder, analog, stepper, and DAC output/input combinations can be attached to an axis. The only restrictions are how the board is configured from a hardware standpoint (See Ordering Matrix).

PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

INTERFACE SCHEMES

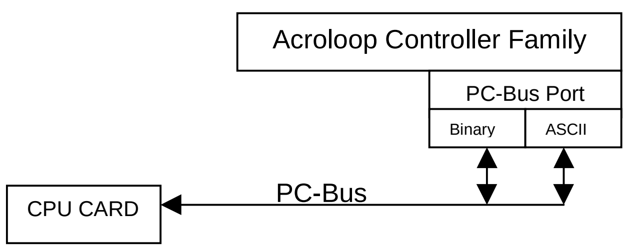
Introduction

The ACR2000, ACR8000, and ACR8010 are the only motion controllers available on the market with 4 communications ports all directly on the card. The ACR1200 is a standalone card only and the ACR1500 is a PC-Bus card only, but they can still communicate in a variety of methods. Due to the unique architecture of the Acroloop controller family and supporting hardware, the controllers can receive and transmit data over any port simultaneously.

The Acroloop controllers can also communicate in a variety of ways including: ASCII, Binary Interface, and String Handling. **The Acroloop interface scheme is unique since it allows ASCII (or String) and Binary information over the PC-Bus port to occur simultaneously. The simultaneous access ports allow unrestricted integration of front end software designs.**

ASCII Interface

The Acroloop controllers are unique since they can communicate both ASCII and Binary schemes simultaneously. Typically, the ASCII scheme is used for traditional serial communications in standalone situations. Simple ASCII codes can be transmitted to the board over any of the on-board communications ports (COM1, COM2, LPT, and the PC-BUS). However, the ASCII port on the PC-Bus is critical in PC-Bus applications. The Binary port is used to handles the coordinate system data while the ASCII port is left open for additional communication that may require bypassing the Binary interface port. The Acroloop scheme allows 2 access doors to the controllers, simultaneously. The second door is open for bi-directional communications to and from the controller to the CPU.



Acroloop Simultaneous Binary and ASCII Interface

Binary Interface

The Acroloop controller family can communicate through the use of a binary interface. A binary interface allows communications to occur in “coded” form. By using the “coded” form, communications can be increased dramatically to allow increased data throughput to and from the controllers. **The Acroloop controller family ACR1500, ACR2000, ACR8000, and ACR8010 controllers have 2 onboard high speed FIFO buffers to effectively transfer data over the PC-BUS from the host processor**.

PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

Typically, the host processor is used to update the graphical display in a PC-Based application and to also send information to and from the motion controller. Since the PC-Bus has a finite amount of "space" in which to transfer the data, a binary interface allows more data to be transferred over the PC-Bus to increase the block processing speed of the overall system. The binary interface coupled with the trajectory calculations of the Acroloop controller family allows time critical machine tool applications to be executed with ease.

The binary interface allows access to the controller's system parameters at any time. The binary interface consists of "data packets". The data packets are "coded" information. The packet is the quickest way to access information such as current position and following error to display on the operator interface in an application program. Binary Interface data packets are requested from the host processor by sending a four byte binary request record.

Binary Interface Data Packet Description Table

Byte #	Description
Byte 0:	Header ID code (0x00)
Byte 1:	Group Code
Byte 2:	Group Index
Byte 3:	Isolation Mask

The group code and group index work as a pair to select the data coming back in a data packet. The group code selects a general data grouping and the group index selects a set of eight fields within that group. The isolation mask then selects which of these eight fields is to compose the final data packet.

The isolation mask acts as a ***filter*** to select only the specific data required. For example, the isolation mask could be specified to retrieve the actual position for axis 2, axis 3, and axis 5. If a bit is set in this mask, the corresponding data filed is allowed to return the data packet. In order to return all eight fields, the isolation mask must be 0xFF. Mask BIT0 is used to isolate the first field in a group and BIT7 is used to isolate the last field.

The following table is a list of groups and what the isolation mask will isolate:

Binary Interface Isolation Mask Outline Table

Group	Description	Isolation Usage
0x10	Flag Parameters	Eight Consecutive parameters
0x18	Encoder Parameters	ENC0 – ENC7 (ENC0 – ENC9 for ACR8010)
0x19	DAC Parameters	DAC0 – DAC7
0x1A	PLC Parameters	PLC0 – PLC7
0x1B	Miscellaneous Parameters	Eight consecutive miscellaneous parameters
0x1C	Program Parameters	PROG0 – PROG15
0x20	Master Parameters	MASTER0 – MASTER7
0x30	Axis Parameters	AXIS0 – AXIS7
0x40	CMT Parameters	CMT0 – CMT7
0x50	Logging Parameters	Eight consecutive logging parameters

The Appendix of the Users Guide contains a complete list of groups and indexes that can be accessed by using binary data packets. The User's Guide contains complete information and also describes the details for data packet retrieval including the packet header and packet data.

String Handling

The Acroloop controller family is capable of handling string functions. String handling functions also can be sent over any of the communications ports. **Strings allow intuitive structure to programming assignments that otherwise may be limited to ASCII or HEX coded formats.**

As a simple example, consider the program attachments of an “X” and “Y” system, the axis be called “TOOL” and “ARM”. In addition, any hexadecimal, ASCII, or coded numbers could be identified as strings and sent to the board for decoding. The use of strings can greatly simplify programming exercises.

The Acroloop controller family is capable of handling strings over any of the communications ports. The controller string functions can:

1. Identify characters.
2. Search strings.
3. Handle upper and lower case strings simultaneously.
4. Identify characters and lengths.
5. Convert to and from ASCII code.
6. Convert to and from upper and lower case.
7. Convert strings to a numeric value.

Strings could also be sent from other host processors or peripherals to the controllers to provide additional ease of use and intuitiveness. **Since the strings are translated on-the-fly to ASCII codes, the use of strings does not slow down the operation.** The string handling capability offers superior flexibility to a wide variety of applications, See **Programming Example 19** for additional information.

DEVELOPMENT TOOLS

Introduction

There are several tools that can be provided with the Acroloop controller family. The controllers can be provided with AcroVIEW, C++ Libraries, Visual C++ libraries, or Visual Basic Libraries. With the development tools, the controllers can be integrated into customized application software. To aid the user in working with the tools, Acroloop provides a CD containing all of the software tools and documentation. The CD is shipped with every new order from Acroloop.

AcroVIEW Software

AcroVIEW is a development tool used to communicate to the motion controllers. The AcroVIEW utility has several key features including:

- Visual Ladder Logic PLC editor (IEC1131)
- Project programming environment
- Windows Menu driven parameter access
- Windows menu driven flag access
- Built-in debug mode (line by line execution of programs)
- Built-in 4 channel oscilloscope

AcroVIEW can talk to the controllers either over the PC-BUS or RS-232 communications ports (see table below). AcroVIEW is provided at no charge to help integrate, diagnose, and graphically tune the Acroloop motion controllers.

AcroVIEW Communications Table

Type	Communication Port	Operating System
Serial	RS-232	Windows 95/98/NT
Parallel	PC/ISA BUS	Windows NT

The designer can simply download the AcroVIEW communication utility from the Acroloop website free of charge at www.acroloop.com and under Transfer Zone/Public Files.

AcroVIEW - Initial Set Up

After receiving the controller, the user can set-up communications to the board immediately by using the supplied AcroVIEW communications tools supplied with the Acroloop CD (both serial and PC-Bus versions are provided in the same CD). This eliminates the need to develop communications to the controller over other communication tools (i.e. ProComm, Telex, Windows Terminal, etc.). In addition, a simple test program can be executed to insure that the controller is operating correctly (See Programming Example 1 and 2).

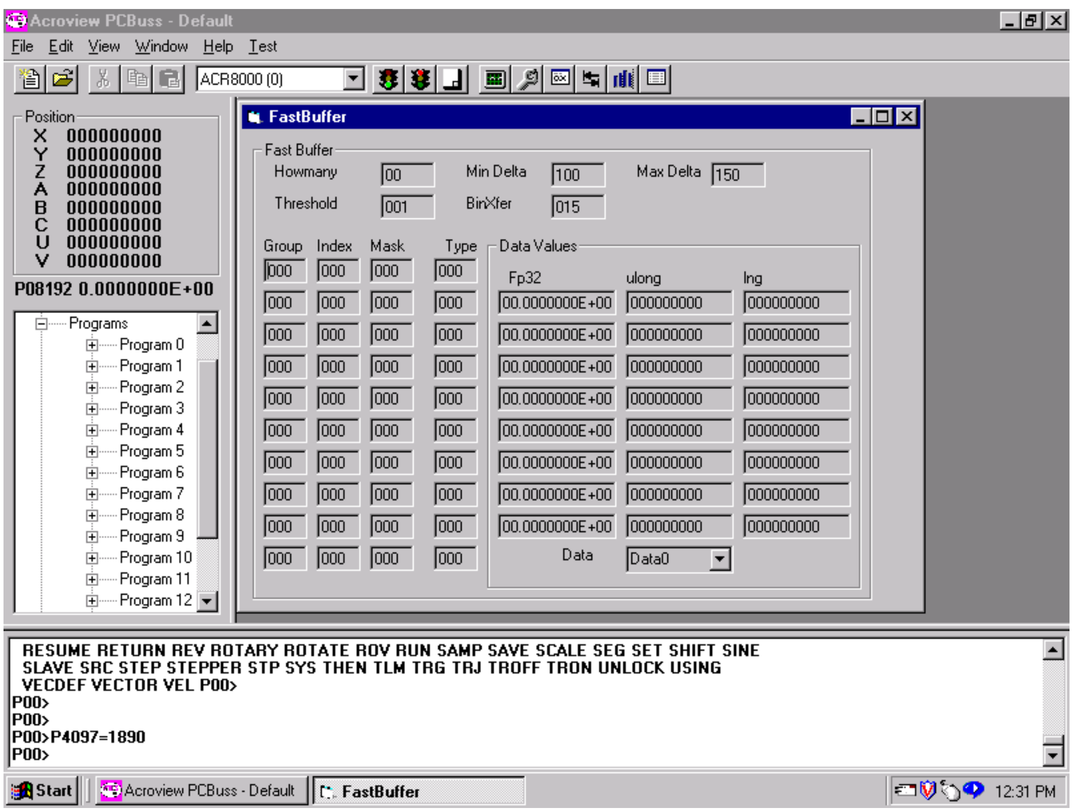
AcroVIEW - Diagnostics

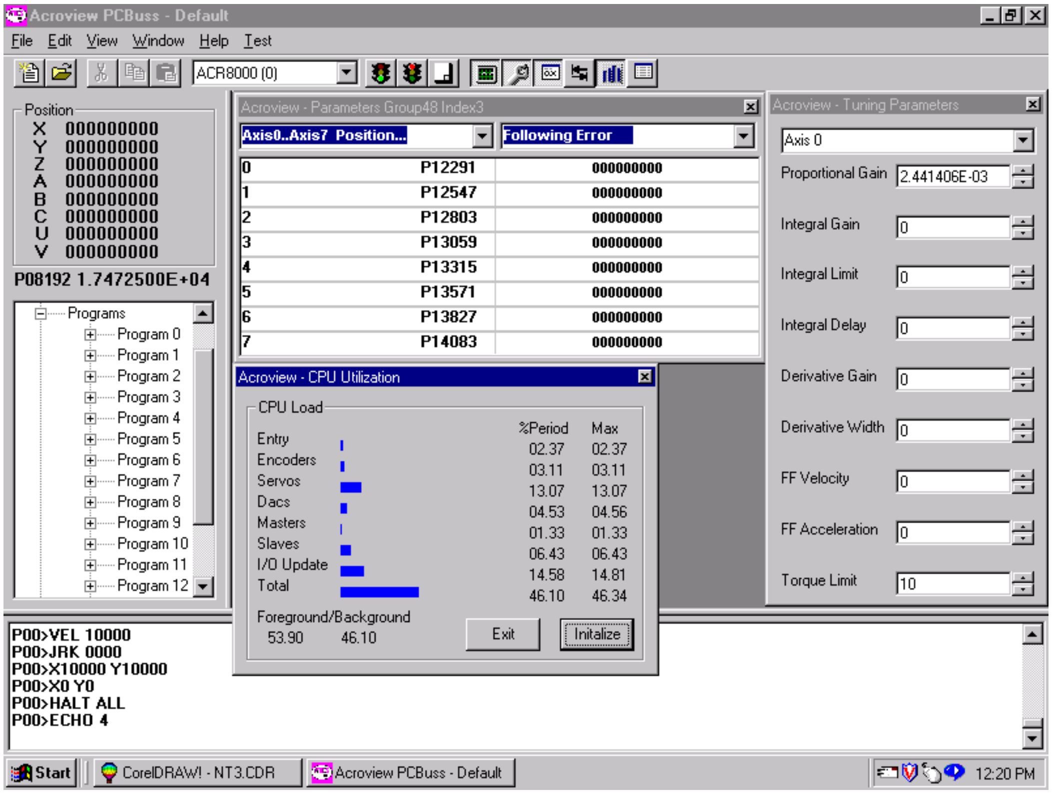
The AcroVIEW supplied software can also act as a valuable diagnostic tool. AcroVIEW can be used to enter in customized software programs and test the controllers on a bench before implementing the controller into a real world application. AcroVIEW allows the user to send and receive files to and from the controllers. In addition, **AcroVIEW provides an on-screen real time display of the input and outputs status while the desired program is being executed.**

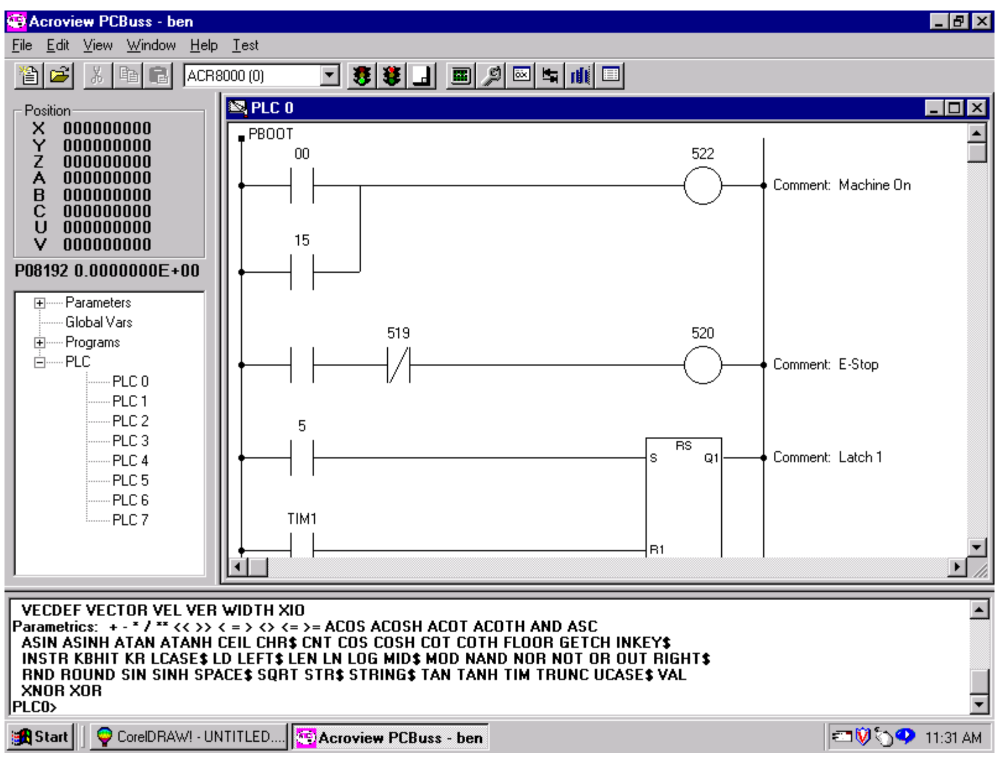
AcroVIEW - Graphical Tuning

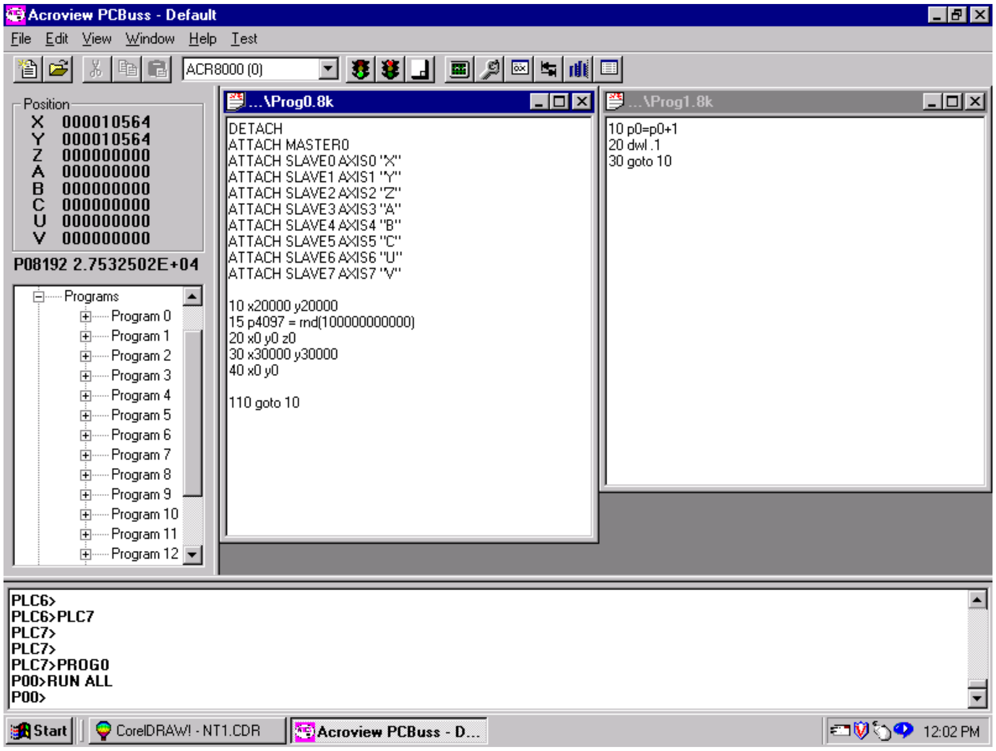
Possibly the most important aspect of AcroVIEW is to allow the user to graphically tune the servo system. AcroVIEW provides a built-in oscilloscope so that the servo system can be tuned on the computer screen. This could either be accomplished through the parallel port or via a laptop computer at the installation site. Even if other development tools are being used (i.e. Visual C++ or Visual Basic DLL's); AcroVIEW can be run separately to insure that the system is tuned properly.

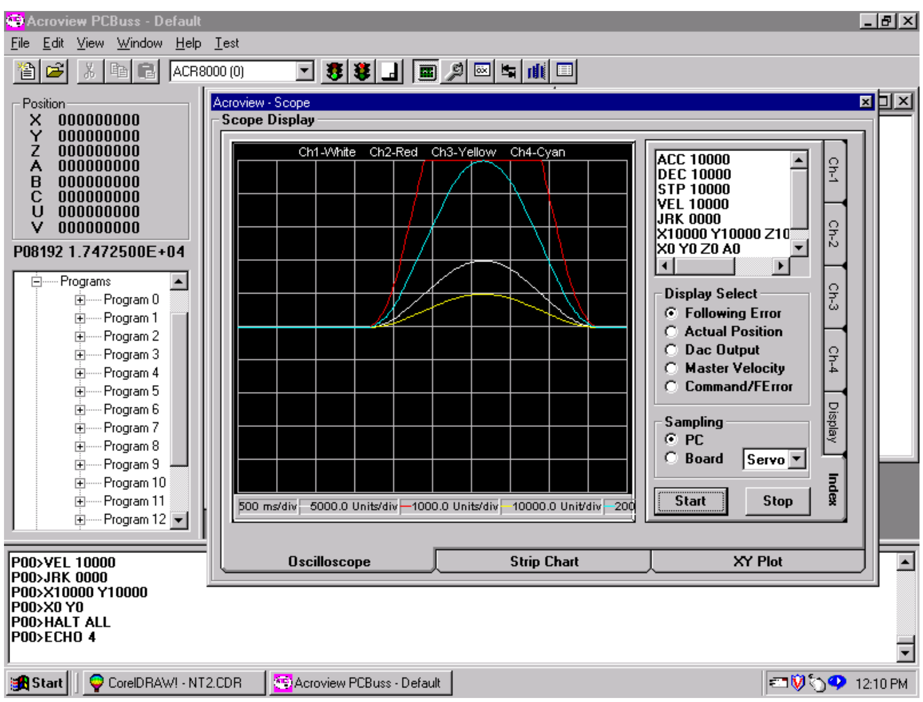
SAMPLE SCREENS FOR ACROVIEW




Acroview PCBuss - Default
File Edit View Window Help Test
ACR8000 (0)
Position
X 000000000
Y 000000000
Z 000000000
A 000000000
B 000000000
C 000000000
U 000000000
V 000000000
P08192 1.7472500E+04
Programs
Program 0
Program 1
Program 2
Program 3
Program 4
Program 5
Program 6
Program 7
Program 8
Program 9
Program 10
Program 11
Program 12
Acroview - Parameters Group48 Index3
Axis0..Axis7 Position...
Following Error
0 P12291 000000000
1 P12547 000000000
2 P12803 000000000
3 P13059 000000000
4 P13315 000000000
5 P13571 000000000
6 P13827 000000000
7 P14083 000000000
Acroview - CPU Utilization
CPU Load
%Period Max
Entry 02.37 02.37
Encoders 03.11 03.11
Servos 13.07 13.07
Dacs 04.53 04.56
Masters 01.33 01.33
Slaves 06.43 06.43
I/O Update 14.58 14.81
Total 46.10 46.34
Foreground/Background
53.90 46.10
Exit
Initalize
Acroview - Tuning Parameters
Axis 0
Proportional Gain 2.441406E-03
Integral Gain 0
Integral Limit 0
Integral Delay 0
Derivative Gain 0
Derivative Width 0
FF Velocity 0
FF Acceleration 0
Torque Limit 10
P00>VEL 10000
P00>JRK 0000
P00>X10000 Y10000
P00>X0 Y0
P00>HALT ALL
P00>ECHO 4
Start
CorelDRAW! - NT3.CDR
Acroview PCBuss - Default
12:20 PM


Acroview PCBuss - ben
File Edit View Window Help Test
ACR8000 (0)
Position
X 000000000
Y 000000000
Z 000000000
A 000000000
B 000000000
C 000000000
U 000000000
V 000000000
P08192 0.0000000E+00
Parameters
Global Vars
Programs
PLC
PLC 0
PLC 1
PLC 2
PLC 3
PLC 4
PLC 5
PLC 6
PLC 7
PLC 0
PBOOT
00
522
Comment: Machine On
15
519
520
Comment: E-Stop
5
RS
S
Q1
Comment: Latch 1
TIM1
R1
VECDEF VECTOR VEL VER WIDTH XIO
Parametrics: + - * / ** << >> < = > <> <= >= ACOS ACOSH ACOT ACOTH AND ASC
ASIN ASINH ATAN ATANH CEIL CHR$ CNT COS COSH COT COTH FLOOR GETCH INKEY$
INSTR KBHIT KR LCASE$ LD LEFT$ LEN LN LOG MID$ MOD NAND NOR NOT OR OUT RIGHT$
RND ROUND SIN SINH SPACE$ SQRT STR$ STRING$ TAN TANH TIM TRUNC UCASE$ VAL
XNOR XOR
PLC0>
Start
CorelDRAW! - UNTITLED...
Acroview PCBuss - ben
11:31 AM





PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

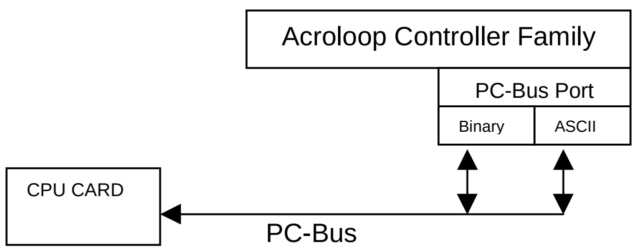
Acroloop Libraries

With the Acroloop Library, commands available on the Acroloop motion controller have a corresponding function call. In PC-based programming languages, the actual motion portion of the program is usually very small. Applications that are programmed on the PC simply use the function call provided in the library when the program needs to use the controller's motion control capabilities. The Acroloop controller family can be supplied with the following libraries:

1. C/C++ Library (DOS)
2. Visual C++ Library (Windows, Windows NT)
3. Visual Basic Library (Windows, Windows NT)

PC Operating Systems

The libraries can be used in DOS, Windows, and Windows NT operating systems. Since the libraries are generically designed, the libraries can be used for additional programming languages such as Delphi. The correct arguments must be programmed for this platform.



Acroloop Simultaneous Binary and ASCII Interface

Visual C++™ and Visual Basic™ DLL's allow the programmer to easily customize a graphically orientated operator interface to work in conjunction with the Acroloop controllers. The Acroloop controllers support simultaneous binary and ASCII interface schemes. The "double door" design allows complete access without worrying about congesting the port. Acroloop also supports Windows NT™ environments. The ***Windows NT™*** DLL's can support multi-thread applications running under ***Windows NT™***.

List of 32-Bit Library Functions

Version Retrieval Function
AcroGetVersion - get library version

Error Dialog Control Functions
AcroShowErrorDialogs
AcroHideErrorDialogs

Parameter Setting Functions
AcroSetInitialWait
AcroSetWaitBeforeTimeOut
AcroSetWaitBetweenStatusFetch
AcroSetMaxSizeofInputTextBuffer
AcroSetMaxSizeofInputBinaryBuffer

Buffer Routines for Text and Binary Input Buffers
AcroTextBufferEmpty
AcroBinaryBufferEmpty
AcroTextBufferNumberOfElements
AcroBinaryBufferNumberOfElements
AcroGetError
AcroSetError

String Routines
AcroReceiveTextString
AcroSendTextString

Initialization Routines
AcroCardPresent
AcroInitialize
AcroWait
AcroSendEscape

Direct Binary Access Commands
AcroLookFP
AcroLookLong
AcroSendProgramToCard Lib
AcroSendString
AcroSetpParameter

TimeOut Control Functions
AcroSetRetryBeforeReadTimeOut
AcroSetRetryBeforeWriteTimeOut
AcroSetRetryBeforeBinReadTimeOut
AcroSetRetryBeforeBinWriteTimeOut

List of 32-Bit Library Functions (continued)

Performance Monitoring Routines
AcroInitPerformance
AcroGetPerformance
AcroSetPerformance
AcroGetTransactionState
AcroSetTransactionState

Debugging Routines
AcroSimulateCard
AcroGetFastStatusTimerDelta
AcroSetFastStatusTimerDelta
AcroGetWriteTimerDelta
AcroSetWriteTimerDelta
AcroGetReadTimerDelta
AcroSetReadTimerDelta
AcroGetBinaryTimerDelta
AcroSetBinaryTimerDelta
AcroGetStatusTimerDelta
AcroSetStatusTimerDelta
AcroGetWriteTimeout
AcroSetWriteTimeout
AcroGetReadTimeout
AcroSetReadTimeout
AcroGetBinaryTimeout
AcroSetBinaryTimeout
AcroGetBinTransThreshold
AcroSetBinTransThreshold
AcroGetMaxFastStatusDelta
AcroSetMaxFastStatusDelta
AcroGetMaxBinTransPerFSInstr
AcroSetMaxBinTransPerFSInstr

Move Counter Functions
AcroGetMoveCounter
AcroSetMoveCounter

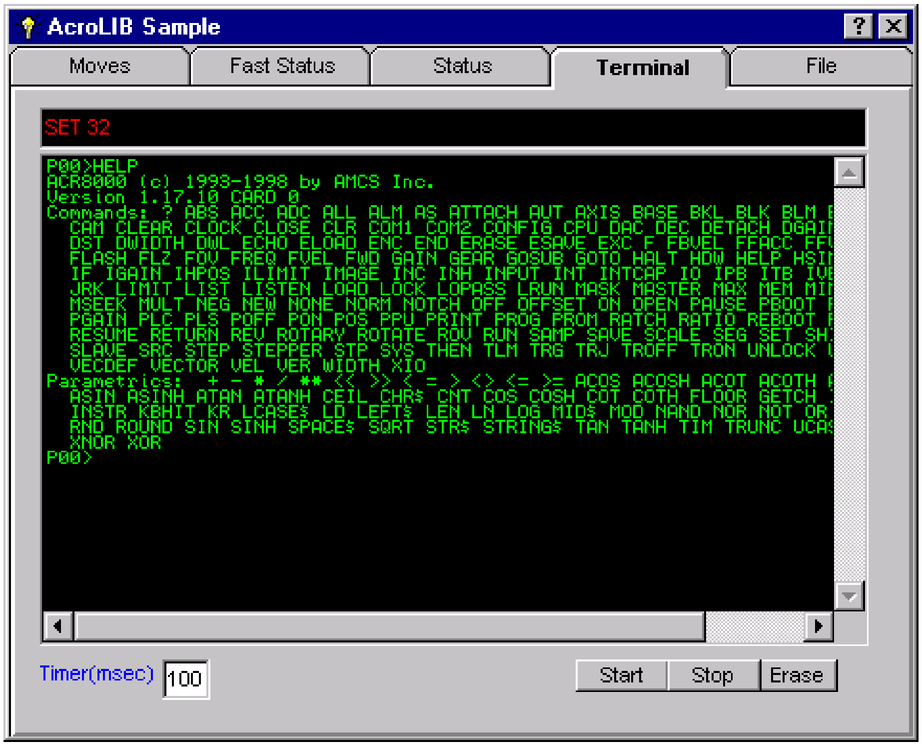
List of Library Functions (continued)

Direct Binary Set Commands
A8_BIN_SET
A8_BIN_CLR
A8_BIN_FOV
A8_BIN_ROV

Binary Parameter Access/Setting Commands
A8_BIN_GETLONG
A8_BIN_SETLONG
A8_BIN_GETIEEE
A8_BIN_SETIEEE
A8_BIN_ADDRESS
A8_BIN_PEEK_LONG
A8_BIN_PEEK_IEEE
A8_BIN_POKE_IEEE
A8_BIN_POKE_LONG
A8_BIN_INIT_STATUS
A8_BIN_GET_STATUS
A8_BIN_MOVE_LONG
A8_BIN_MOVE_IEEE
A8_BIN_GROUP_GETIEEE
A8_BIN_GROUP_GETLONG
A8_BIN_SYSADDRESS
A8_BIN_MASK
A8_BIN_SYS_MASK

General AcroLanguage Commands
All general AcroBASIC commands can be sent to the board using the "AcroSendString" command. This allows infinite flexibility in designing the board interaction to the front end operator interface.

Free Visual Basic sample application code is included on the Acroloop CD. The AcroLIB sample shows Visual Basic source code examples to program the Acroloop controllers.



Visit the Acroloop website to obtain a free copy of the Visual Basic library at www.acroloop.com.

PROGRAMMING EXAMPLES

Program 1 - Initial Test Program

The following program is a simple test program that will randomly move 2 axes for an infinite period of time until the program is stopped (HALT) by the user. The program could be expanded so that the test is completed on more than two axes.

<u>Declaration and Set-up</u>

```
PROG0                      REM:  Select program number 0.
HALT                       REM:  Halt execution (Good practice).
DETACH                     REM:  Cancel any attachments.
ATTACH MASTER0             REM:  Attach the master to program.
ATTACH SLAVE0 AXIS0 X      REM:  Define X-axis and slave to master.
ATTACH SLAVE1 AXIS1 Y      REM:  Define Y-axis and slave to master.
NEW                        REM:  Start a new program (Good practice).
```

After the declarations are made, we can begin to write the actual program. Any command that is preceded by a valid line number will get stored into the memory of the current selected program number. Valid line numbers are in the range of 1 to 65000. The line numbers are also used as destination address numbers. In this case, the program and corresponding line numbers will be stored in memory attached to program number 0.

```
10 PBOOT                   REM:  Runs program automatically on power-up.
20 DV1 = RND(12)*30        REM:  DV1 equals a random integer between 1 and 12
                                 multiplied by 30.
30 DV2 = 30000             REM:  DV2 equals 30000.
40 DV3 = DV2*SIN(DV1)      REM:  Assign DV3 to a random value.
50 DV4 = DV2*COS(DV1)      REM:  Assign DV4 to a random value.
60 X(DV3) Y(DV4)           REM:  Move the X-axis DV3 pulses and move the Y-axis
                                 DV4 pulses.
70 GOTO 20                 REM:  Continue random test.

DIM DV(4)                  REM:  Dimension memory for 4 variables.
```

The test program will run indefinitely with the RUN command until stopped by the operator. To stop the program, enter the following commands:

```
HALT                       REM:  Stop test.
```

Program 2 - Input/Output Test Program

The inputs and outputs of the controllers can easily be tested to insure proper operation. If the controller is ordered with the optional interconnect board, the I/O can be observed directly on the face of the enclosure with the RED and GREEN I/O status LED's. The status LED's are important for diagnostic and troubleshooting purposes. If the optional interconnect board was not ordered with the controller board, the input and output status conditions can be observed through AcroVIEW (See Development Tools).

Declaration and Set-up

```
PROG1                              REM:  Select program number 1.
HALT                               REM:  Stop execution (Good practice).
NEW                                REM:  Start a new program (Good practice).
```

To observe the inputs and outputs cycling on and off, the inputs and outputs should be shorted together so that when the output is SET, the corresponding input is also observed.

```
10 DV1 = 32                        REM:  Initialize DV1 to 32.
20 IF (DV1 = 64) THEN GOTO 100     REM:  Goto clear outputs.
30 SET(DV1)                        REM:  Turn on Output number DV1.
40 DWL 0.1                         REM:  Dwell for 0.1 seconds.
50 DV1 = DV1 + 1                   REM:  Select next output.
60 GOTO 20                         REM:  Repeat next output.

100 DV1 = 32                       REM:  Initialize DV1.
110 IF (DV1 =64) THEN GOTO 10      REM:  After 32 clears repeat outputs.
120 CLR (DV1)                      REM:  Clear output number DV1.
130 DWL 0.1                        REM:  Dwell for 0.1 seconds.
140 DV1 = DV1 + 1                  REM:  Increment DV1 by 1.
150 GOTO 110                       REM:  Repeat next output clear.

CLEAR                              REM:  Clears memory allocated to local  variables and
                                         arrays.
DIM DV(1)                          REM:  Dimension memory for 1 variable.
```

To stop the input/output test, type "HALT" at the P00> prompt at the main terminal screen:

```
HALT                               REM:  Stop test; outputs held at last value.
P4097 = 0                          REM:  Clears all outputs to zero.
```

Program 3 - Sample HOME Test Program

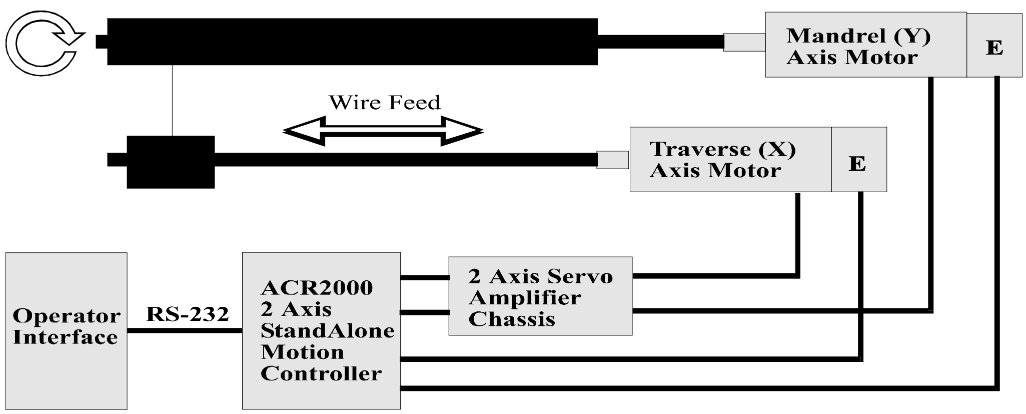
A HOME test routine can also be easily accomplished. The HOME test allows an arbitrary axis to move rapidly to the HOME switch, back off the switch and then seek the marker pulse on the encoder for the axis. In this example, the home switch has been connected to input 1.

PROG0	REM:	Select program number 0.
HALT	REM:	Stop execution (Good practice).
DETACH	REM:	Cancel any attachments.
ATTACH MASTER0	REM:	Attach the master to program.
ATTACH SLAVE0 AXIS0 “X”	REM:	Define X-axis and slave to master.
NEW	REM:	Start a new program.
100 ACC 20000	REM:	Set profiler 0 acceleration ramp to 20000 pulses/sec/sec.
110 DEC 20000	REM:	Set profiler deceleration ramp.
120 STP 20000	REM:	Set stop ramp(ending decel).
130 IF (BIT1) THEN GOTO 300	REM:	Skip switch approach routine if already at the home switch.
200 VEL 10000	REM:	Rapid home profiler velocity.
210 X/10000000	REM:	Move X-axis incrementally (/) toward the home switch. 10000000 pulses is selected to insure reaching home switch.
220 INH 1	REM:	Wait until home switch is closed.
230 SET 523	REM:	Set the Stop All Moves Req (for master profiler 0**). A Kill All Moves Req is automatically generated at the end of the Stop All Moves Req.**
240 INH –516 **INH -523**	REM:	Wait until motor has stopped (516 is the “in motion” flag for master 0). **Wait for the INH –523 (Stop All Moves Req) to clear itself.**
250 CLR 222	REM:	Clear the Kill All Moves Req for master profiler 0 (flag was set in statement 230).
300 VEL 1000	REM:	Set velocity to move off home switch.
310 X/-10000000	REM:	Move X-axis incrementally (/) off the closed home switch.
320 INH –1	REM:	Wait until home switch is open.
330 SET 523	REM:	Set the Stop All Moves Req (for master profiler 0**). A Kill All Moves Req is automatically generated at the end of the Stop All Moves Req.**
340 INH –516 **INH -523**	REM:	Wait until motor has stopped(516 is the “in motion” flag for master 0). **Wait for the INH –523 (Stop All Moves Req) to clear itself.**
350 CLR 222	REM:	Clear the Kill All Moves Req for master profiler 0 (flag was set in statement 330).
400 VEL 1000	REM:	Set velocity to look for marker.
410 MSEEK X (10000,0)	REM:	Find the X-axis encoder marker pulse. <u>Can add an offset value here if desired.</u>
420 END	REM:	End of program.

Boldface is applicable to ver 17.05 & above. Remove boldface statements for ver. 17.04 <u>and below.</u>

Program 4 - Coil Winding

The Acroloop controllers can be easily implemented to perform a typical 2-axis coil winding application. The coil winding application makes use of the standard command set for the controllers and illustrates the ease of use for the product. A typical coil winding application requires turning the mandrel motor synchronously with the wire feed or traverse motor (see diagram). In this example we will select the mandrel motor as the Y-axis and the traverse motor as the X-axis.



Coil Winding Machine Diagram

As in the previous examples, the controller must be properly set up to perform the application.

PROG0	REM: Select program number 0.
HALT	REM: Stop execution (Good practice).
DETACH	REM: Cancel any attachments.
ATTACH MASTER0	REM: Attach the master to program.
ATTACH SLAVE0 AXIS0 “X”	REM: Define X-axis and slave to master.
ATTACH SLAVE1 AXIS1 “Y”	REM: Define Y-axis and slave to master.
PPU X2000 Y1000	REM: X-axis is 2000 **pulses/inch**, Y is 1000 **pulses/rev**.
VECDEF X1	REM: Board calculates vector distance using the X-axis
VECDEF Y0	REM: Board does not use Y-axis to calculate vector.
NEW	REM: Start new program.

Program 4 - Coil Winding (continued)

We will assume that the system requires 500 revolutions (Y-axis) for every 10 inches of travel (X-axis) at 0.5 inches per second velocity. Let's also assume that the system will be allowed to ramp up to speed and down from speed in 1 second. From physics, velocity = (acceleration * time). Thus, to ramp up to 0.5 inches/second in 0.5 seconds, the acceleration is 1 inch/second/second.

10 VEL 0.5 ACC 1.0 STP 1.0	REM:	Set velocity, acceleration, and stop (ending decel) values.
20 X10 Y500	REM:	Move X 10 inches and Y 500 revolutions.

Additionally, we could easily modify the program to include a changing pitch requirement. If we assume that the pitch doubles halfway through the program, then the program would be as follows:

10 VEL 0.5 ACC 1.0 DEC 1.0	REM:	Set velocity, acceleration, and deceleration values.
20 X5 Y/250	REM:	Move X 5 inches and Y incrementally (/) 250 revolutions.
30 X10 Y/500	REM:	Move X 5 inches and Y incrementally (/) 500revolutions(double pitch)

Note: The Acroloop controllers can easily perform coil winding applications involving additional axes. **The Acroloop controllers are currently used in coil winding involving multiple axes and complex shapes**. The applications are easily accomplished by implementing the additional hardware and firmware capabilities.

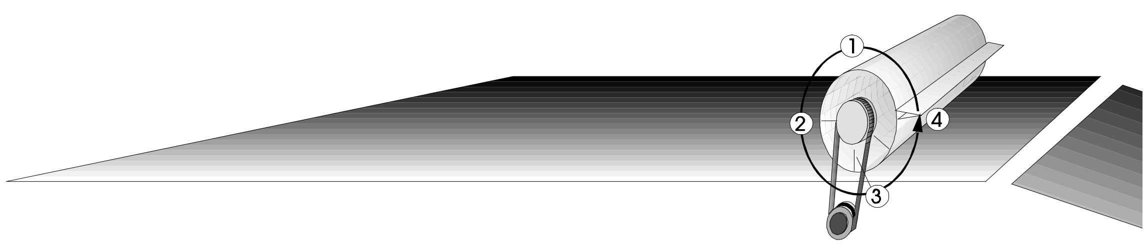
Program 5 - Segmented Electronic CAM

The Acroloop controller family has a multi-axis Electronic CAM. The Acroloop Electronic CAM is unique in that the CAM can be segmented into any number of user definable parts. The segments then are linked together to form any shape. The CAM automatically linearly interpolates between segments.

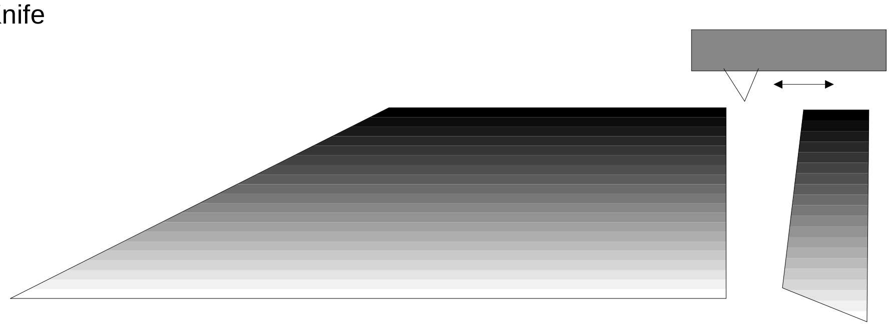
The Segmented electronic cam feature is possibly one of the most powerful command sets on the controllers. Since you can define the segments in a position matching table and control the distance between segments, then the designer can control both the change in position and the change in time. Thus, the segmented cam feature is both a position matching and velocity matching mode. **Some applications include applying labels, sewing patterns, packaging equipment, and inspection stations. The applications for segmented electronic cam are endless and this is merely one example. Any application that requires either position or velocity matching or both is a candidate for segmented electronic cam.**

One example for segmented electronic cam is a form fill machine in a packaging machine application. The idea is to create a profile that minimizes the spillage of the product being dispensed into a carton while maximizing the speed of filling the carton. Other examples include a rotary flying knife or a linear flying knife.

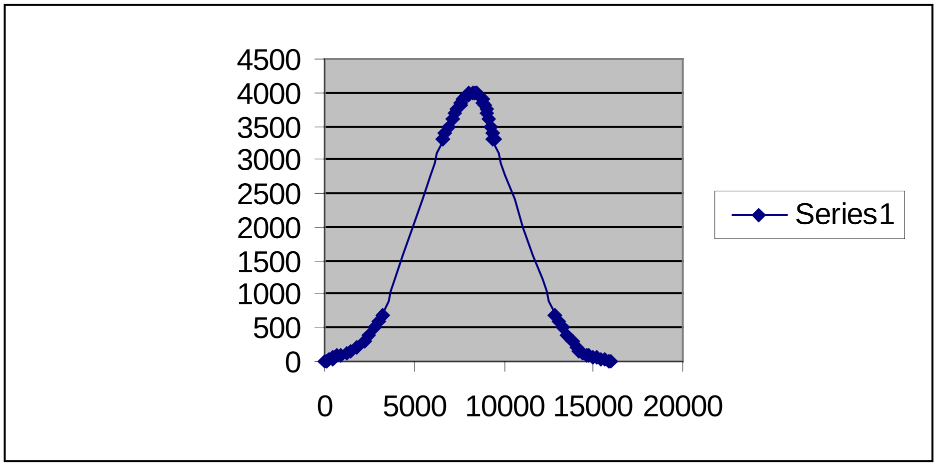
Rotary Flying Knife



Linear Flying Knife



Program 5 - Segmented Electronic CAM (continued)



The development of the Flying Linear Knife (which directly relates to the rotary flying knife) is as follows:

```
PROG0                        REM:  Select program number 0.
HALT                         REM:  Stop execution(Good practice).
DETACH                       REM:  Cancel any attachments.
ATTACH MASTER0               REM:  Attach the master to program.
ATTACH SLAVE0 AXIS0 “X”      REM:  Define X-axis and slave to master.
ATTACH SLAVE1 AXIS1 “Y”      REM:  Define Y-axis and slave to master.

MULT X4 Y4                   REM:  Setup multipliers (quadrature)
RES X Y                      REM:  Reset encoder registers to zero for start-up

DIM LA(7)                    REM:  Dimension 7 arrays
DIM LA0(2)                   REM:  Dimension 2 elements in array 0
DIM LA1(15)                  REM:  Dimension 15 elements in array 1
DIM LA2(2)                   REM:  Dimension 2 elements in array 2
DIM LA3(29)                  REM:  Dimension 29 elements in array 3
DIM LA4(2)                   REM:  Dimension 2 elements in array 4
DIM LA5(15)                  REM:  Dimension15 elements in array 5
DIM LA6(2)                   REM:  Dimension 2 element in array 6

LA0(0) = 0                   REM:  Elements for array 0
LA0(1) = 0

LA1(0) = 0                   REM:  Elements for array 1
LA1(1) = 25
LA1(2) = 35
LA1(3) = 45
LA1(4) = 55
LA1(5) = 75
LA1(6) = 95
```

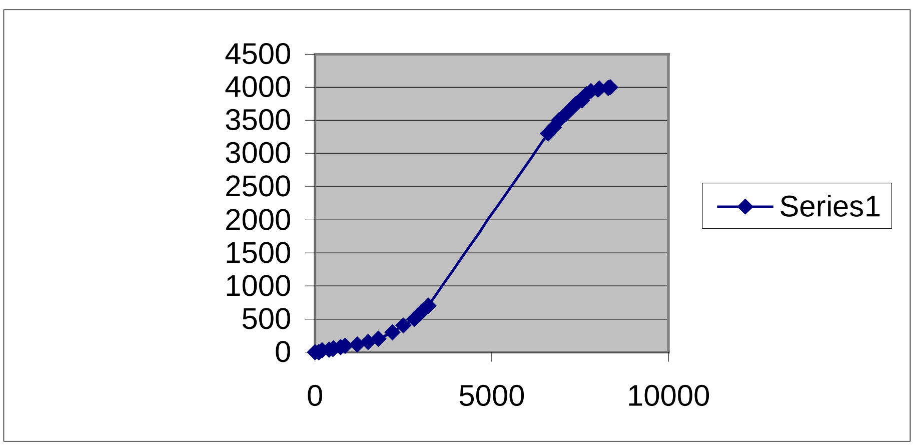
Program 5 - Segmented Electronic CAM (continued)

```
LA1(7) = 115
LA1(8) = 150
LA1(9) = 200
LA1(10) = 300
LA1(11) = 400
LA1(12) = 500
LA1(13) = 600
LA1(14) = 700

LA2(0) = 700                        REM:   Elements for array 2
LA2(1) = 3300

LA3(0) = 3300                       REM:   Elements for array 3
LA3(1) = 3400
LA3(2) = 3500
LA3(3) = 3600
LA3(4) = 3700
LA3(5) = 3750
LA3(6) = 3800
LA3(7) = 3850
LA3(8) = 3900
LA3(9) = 3920
LA3(10) = 3940
LA3(11) = 3960
LA3(12) = 3980
LA3(13) = 3990
LA3(14) = 4000
LA3(15) = 3990
LA3(16) = 3980
LA3(17) = 3960                      REM:   Remaining elements for array 3
LA3(18) = 3940
LA3(19) = 3920
LA3(20) = 3900
LA3(21) = 3850
LA3(22) = 3800
LA3(23) = 3750
LA3(24) = 3700
LA3(25) = 3600
LA3(26) = 3500
LA3(27) = 3400
LA3(28) = 3300

LA4(0) = 3300                       REM:   Elements for array 4
LA4(1) = 700

LA5(0) = 700                        REM:   Elements for array 5
LA5(1) = 600
LA5(2) = 500
```

Program 5 - Segmented Electronic CAM (continued)

```
LA5(3) = 400
LA5(4) = 300
LA5(5) = 200
LA5(6) = 150
LA5(7) = 115
LA5(8) = 95
LA5(9) = 75
LA5(10) = 55
LA5(11) = 45
LA5(12) = 35
LA5(13) = 25
LA5(14) = 0

LA6(0) = 0                    REM:  Elements for array 6
LA6(1) = 0


CAM DIM Y7                    REM:  Dimension 7 segments for Y axis
CAM SEG Y(0,100,LA0)          REM:  Segment 0 is 100 pulses long, use data in LA0
CAM SEG Y(1, 3100, LA1)       REM   Segment 1 is 3100 pulses long, use data in LA1
CAM SEG Y(2, 3400, LA2)       REM:  Segment 2 is 3400 pulses long, use data in LA2
CAM SEG Y(3, 2800, LA3)       REM:  Segment 3 is 2800 pulses long, use data in LA3
CAM SEG Y(4, 3400, LA4)       REM:  Segment 4 is 3400 pulses long, use data in LA4
CAM SEG Y(5, 3100, LA5)       REM:  Segment 5 is 3100 pulses long, use data in LA5
CAM SEG Y(6, 100, LA6)        REM:  Segment 6 is 100 pulses long, use data in LA6

CAM SRC Y ENC0                REM:  Cam source for Y axis is encoder 0

CAM ON Y                      REM:  Turn on the cam
```

A similar cam profile can be developed for a rotary flying knife. The rotary flying knife profile would be:

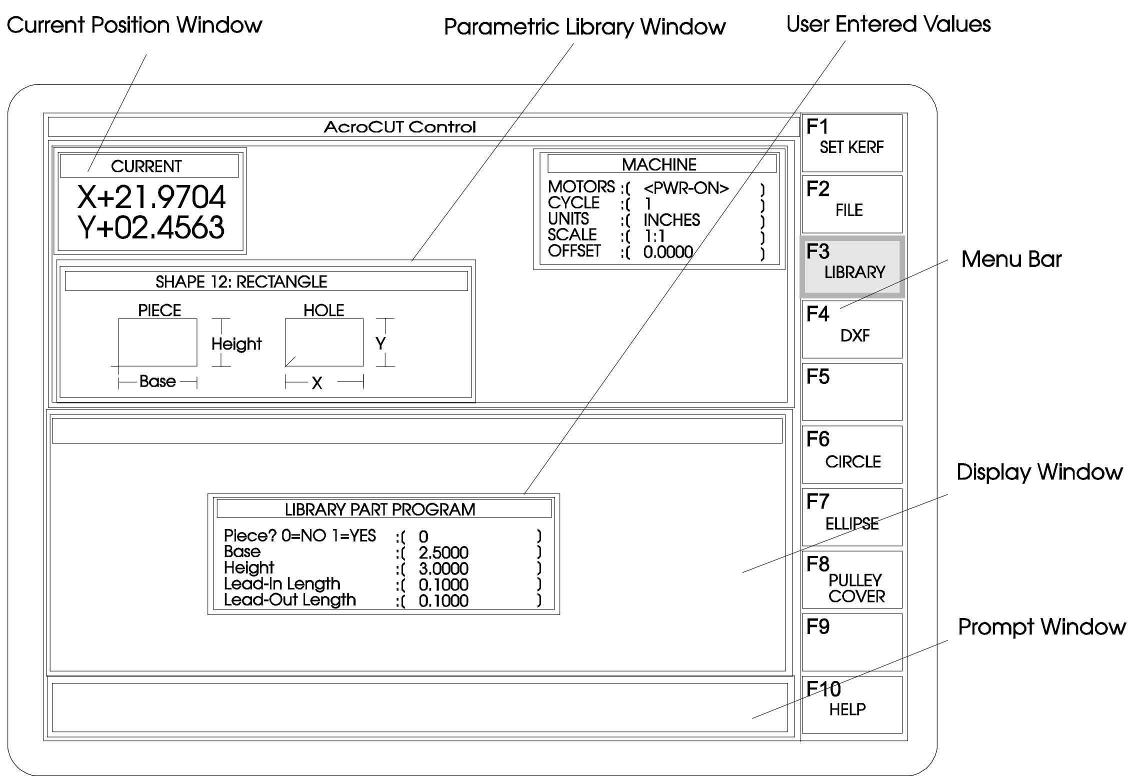


Important Notes on Electronic CAM:

1. Electronic CAM links the output position to the encoder source via a CAM table stored in the controller. This eliminates lengthy calculations to be performed thus reducing the time to match the slave position(s). Therefore, CAM is fast and with proper tuning realizes zero following error.
2. Electronic CAM automatically linearly interpolates between entries.
3. Electronic CAM can be segmented into any number of subsets. This allows as coarse or as fine of resolution between entries as desired.
4. Electronic CAM will function for any number of axes matched to the encoder source.
5. With Electronic CAM it doesn't matter how many times the encoder source has turned, just the current rotational position matters.
6. Electronic CAM also incorporates scaling, offsets, and floating zeros.

Program 6 - Plasma, Oxy-Fuel, Laser, and Water Jet Cutting

Machine Tool Cutting is accomplished by implementing Acroloop's off-the-shelf AcroCUT software package. AcroCUT contains all of the vital software functions that are necessary to perform machine tool cutting applications. AcroCUT is an Acroloop engineered product that works in conjunction with the Acroloop motion controllers. For example, we will assume that we want to plasma cut a quantity of 100 - 2.5"x3" rectangles from a sheet of sheetmetal on the machine. **30 standard shapes and parts are pre-programmed and included in the AcroCUT Part**; to process the part, we simply pull up the library part and enter in the appropriate values (see example screen below).



AcroCUT Library Part Screen

After the rectangle size values are entered, the 100 parts can be programmed onto the sheet by using the **"Shape Repeat"** function built into the AcroCUT controller. AcroCUT provides a pop-up window to enter in the size of the sheet and the number of parts to be cut. After the sizes are entered, AcroCUT provides a graphical display of the sheet of parts. The graphical display will also shows a ***cursor*** that traces the parts as they are cut on the machine.

Plasma, Oxy-Fuel, Laser, and Water Jet Cutting (continued)

AcroCUT Features Table

PC Based - PC Based program, works on DOS or Windows™.
Programmable - The software is programmable using RS-274D M and G codes.
Retrace - Allows "retracing" of the cut path to "back-up" on the cut path.
Programmable - Implement the built-in functions to perform tasks such as ignition, preheat, pierce, and creep times.
Dripfeed - Allows oversized large part files to be executed without cutting interruption; there is no limit on the length of the program size.
Dry Run - A dry run with graphical display.
Run - Executes the loaded program from the hard disk.
Feedrate - Allows increase and decrease of feedrate on-the-fly.
Pause and Resume - On and Off path pause and resume feature. Machine can move in the retrace, skip, resume, jog and home functions after the pause.
Kerf Compensation - Automatic kerf compensation ensures accurate cuts.
Cornering - Automatic corner slowdown based on acute angle threshold; acute angle threshold is user programmable.
Plate Alignment - Automatic plate alignment feature.
Software Limits - Automatic deceleration of cutter before software limits.
Parameters - Sets system parameters including program, offset, compensation, feedrates, limits, assignments, format, and other "protectable" parameters. A special access code is required to change any system parameters.
Graphical Tuning - A graphical tuning display is provided to allow the tuning of the servo system.
Help Screens - User programmable/editable help screens on any display.

AcroCUT Part Manipulation

Built-in Part Library - User customized parametric shape libraries.
Step and Repeat Function - User programmable sheet size as well as a built-in nesting routine for multiple shapes.
Upload - Allows uploading of CAD programs to a host computer via the RS-232 interface or from a resident hard drive.
Download - Allows downloading of CAD programs from a host computer via the RS-232 interface.
Graphics - Displays the cutter path graphically after the program is downloaded and during part execution.
Scaling and Rotation - Allows scaling and rotation of part.
Isometric - Programmable isometric part view.
DXF Translator – Built-in DXF file translator to automatically convert CAD files to G code files.

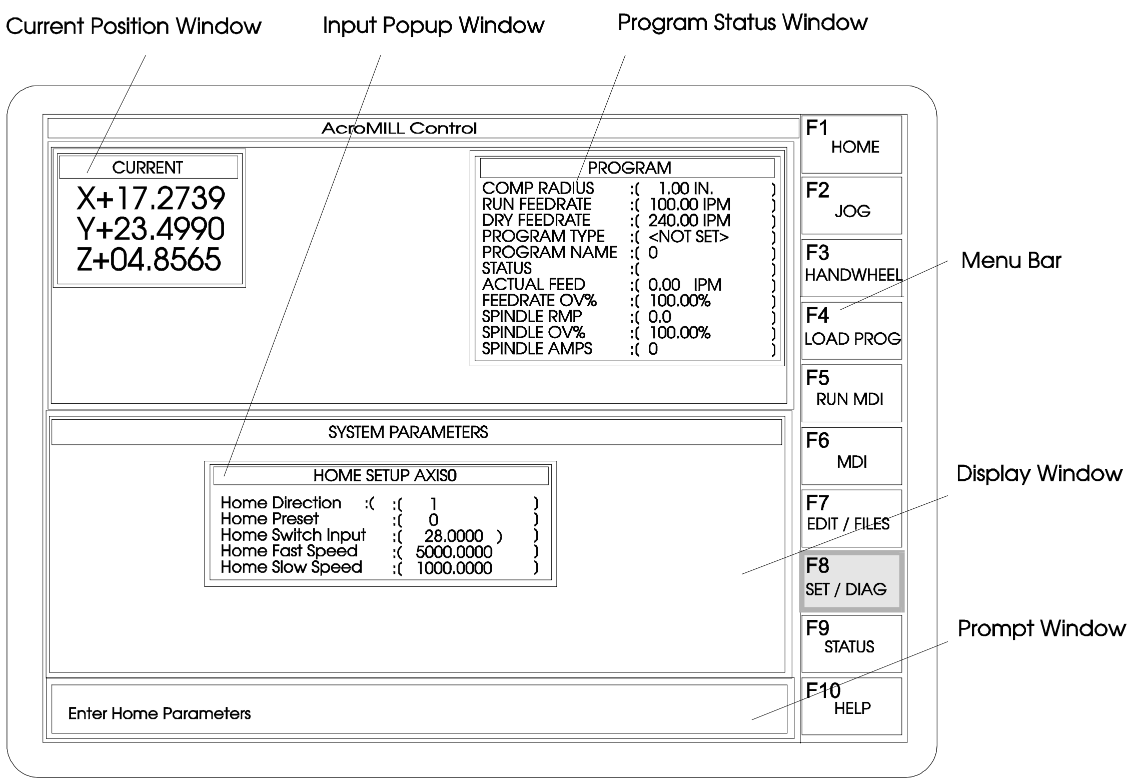
AcroCUT Basic Machine Moves

Home - Allows homing of the machine.
Jog - Incremental or absolute jogging.
MDI - Allows manual data entry for immediate executable commands.

Program 7 - CNC Milling

The Acroloop controller family can be implemented to perform a typical 3-8 axis CNC Milling application. The Acroloop controllers are combined with the Acroloop hardware and the Acroloop CNC Milling software for a complete off-the-shelf turnkey package.

The AcroMILL software sends the appropriate commands to the controllers by implementing the binary interface scheme. The binary interface allows the milling machine to have a much faster block processing time than by using the ASCII interface scheme. For a typical 3 axis application, the block processing time is 2 milliseconds. The 2 millisecond block processing time is based on *continuous* throughput; the binary interface data throughput will withstand short “bursts” of data for faster throughput since the controllers implement high speed FIFO’s on the PC/ISA bus interface. The high speed FIFO’s are important for handling burst data throughput.



AcroMILL Main Screen

Above is an outline of the AcroMILL Main Menu screen. A definition of the various screen menus is outlined on the following page.

Program 7 - CNC Milling (continued)

Outline of AcroMILL Screen Menus

Current Position Window shows you a real-time display of the current position of each of the attached axes.

Input Pop-up Window is used by AcroMill to ask you for values of various parameters.

Program Status Window shows you status of the current program including a real-time display of feedrate and spindle parameters.

Menu Bar shows you the currently active menus. You can select a menu by hitting a function key (for example, F1) attached to it. Hitting *ESCAPE* key will take you back to the previous menu.

Display Window is used by AcroMill to show you pop-up windows for additional screens or to graphically display the part(s) being processed.

Prompt Window is used by AcroMill to show you errors or to occasionally ask for input.

Main Features of AcroMILL:

- PC-Based CNC Machine Tool Software
- Programmable from 2 to 8 axes applications
- Programs in standard M and G code (EIA RS-274D)
- All M code functions are completely programmable
- 32 H codes for offset functions (multi-dimensional)
- S codes for spindle control
- Programmable Tool Radius Compensation
- Programmable Ballscrew and Backlash Compensation
- Built-In programmable feedrate and spindle override hard inputs.
- Built-in PLC for customized circuitry or Automatic Tool Changer
- 32 Tool Offsets (standard)
- Graphical Display of part being processed
- Built-in programmable part libraries (30 shapes provided free)
- User programmable help screens edited by any word processor
- Parametric Programming for 100 variables
- Shape repeat, Scale, and Rotate parts functions
- Graphical Tuning (Built-in Oscilloscope)
- DXF File translator to automatically convert CAD drawings
- RS-232 or Hard Disk File Transfer capability

Program 8 - PC String Interface

Several libraries are available for the Acroloop motion controllers. The standard library support set includes C++ for DOS, Visual C and Visual Basic for Windows and Windows NT Drivers. Since the libraries are generic, they will also support additional platforms such as Borland's Delphi. A list of the function calls is located in the Development Tools section of this brochure.

At the simplest level, all of the interpreted commands can be sent directly to the Acroloop controller. The Acroloop function call "AcroSendString" can be used to send AcroBASIC commands to the controller.

The following program can be sent from Visual Basic using the Windows NT operating system and the Acroloop Visual Basic library for Windows NT. The program simply sets up the Acroloop controller for a single coordinate system consisting of 2 axes (X and Y). The program then moves the axes 1000 units and prints "DONE" when completed.

```
AcroSendString “SYS”, 0
AcroSendString “HALT ALL”, 0
AcroSendString “NEW ALL”, 0
AcroSendString “DETACH ALL”, 0
AcroSendString “CLEAR”, 0
AcroSendString “ERASE ALL”, 0
AcroSendString “DIM PROG0(5000)”, 0
AcroSendString “PROG0”, 0
AcroSendString “ATTACH MASTER0”, 0
AcroSendString “ATTACH SLAVE0 AXIS0 “X” ”, 0
AcroSendString “ATTACH SLAVE1 AXIS1 “Y” ”, 0
AcroSendString “10 X 10000 Y 10000”, 0
AcroSendString “20 PRINT “DONE” ”, 0
AcroSendString “LRUN”, 0
```

In general, any AcroBASIC command can be sent to the Acroloop controllers using the AcroSendString command. This allows general motion functionality to be sent directly to the board while still operating in a PC platform.

For further C++ and Visual Basic programming examples, please reference the low level interface code in Program 10 of this brochure and the Visual Basic example for retrieving current position status from the controllers (Program 10).

Program 9 - Low Level C++ Interface

The Acroloop controllers can be provided with C++ Libraries. As a low level example, the following program communicates to the Acroloop controllers with PC-Bus interfaces (ACR1500, ACR2000, ACR8000, and ACR8010). The program is a low level routine that can be incorporated into higher level routines.

All of the Acroloop libraries and documentation are *free of charge* from the Acroloop website. The Acroloop website address is *www.acroloop.com*. Look under Transfer Zone, and then under the public files listing, for a complete file listing of all the files available for the Acroloop controllers including the API User Guide.

The following C++ sample shows how to construct a simple terminal to the card using AcroLIB API.

Terminal.h:

```
#include "..\..\..\inc\acrolib.h"
#include <stdio.h>
#include <iostream.h>
#include <time.h>
```

Terminal.cpp:

```
/*++

Copyright (c) 1996-1997, Acroloop Motion Control Systems Inc.

Module Name:

       term.cpp

Abstract:

       Sample showing how to construct a simple terminal to the card
       using AcroLIB API. ReadThread reads characters from the card
       and displays them to stdout. WriteThread reads characters from
       stdin and writes them to the card.

Date:

       April 28, 1997.

Author:

       IZ.

--*/


#include "terminal.h"
#include <windows.h>
#include <wincon.h>
```

Program 9 - Low Level C++ Interface (continued)

```
//save the old colors
WORD wOldColorAttrs;

VOID MyErrorExit(char * ErrorMessage);

VOID MyErrorExit(char * ErrorMessage)
{
// write your error function here.
};

//size the console

void SetConsoleSize(HANDLE hConsole)
{

 SMALL_RECT lpConsoleWindow[1];
 CONSOLE_SCREEN_BUFFER_INFO lpConsoleScreenBufferInfo[1];
 COORD LargestWindowSize;

 CONSOLE_SCREEN_BUFFER_INFO csbi; /* to get buffer info */

 GetConsoleScreenBufferInfo( hConsole, &csbi );

 GetConsoleScreenBufferInfo(
    hConsole,                    // handle of console screen buffer
    lpConsoleScreenBufferInfo    // address of screen buffer info.
   );

 LargestWindowSize = GetLargestConsoleWindowSize(hConsole);

 *lpConsoleWindow = lpConsoleScreenBufferInfo->srWindow;

 lpConsoleWindow->Left=0;
 lpConsoleWindow->Top=0;
 lpConsoleWindow->Right=75;
 lpConsoleWindow->Bottom=30;

 if (!SetConsoleWindowInfo(
    hConsole,       // handle of console screen buffer
    TRUE,                  // coordinate type flag
    lpConsoleWindow        // address of new window rectangle
    ))
 {

        printf("Could not set the size.\n");
        printf("GetLastError=%lx\n:", GetLastError());

        while(1);

 }

}
```

Program 9 - Low Level C++ Interface (continued)

```
//clear the console.

void cls( HANDLE hConsole )
{
   COORD coordScreen = { 0, 0 };    /* here’s where we’ll home the
                                       cursor */
   BOOL bSuccess;
   DWORD cCharsWritten;
   CONSOLE_SCREEN_BUFFER_INFO csbi; /* to get buffer info */
   DWORD dwConSize;                 /* number of character cells in
                                       the current buffer */

   /* get the number of character cells in the current buffer */

   bSuccess = GetConsoleScreenBufferInfo( hConsole, &csbi );
   dwConSize = csbi.dwSize.X * csbi.dwSize.Y;

   /* fill the entire screen with blanks */

   bSuccess = FillConsoleOutputCharacter( hConsole, (TCHAR) ’ ’,
      dwConSize, coordScreen, &cCharsWritten );

   /* get the current text attribute */

   bSuccess = GetConsoleScreenBufferInfo( hConsole, &csbi );

   /* now set the buffer’s attributes accordingly */

   bSuccess = FillConsoleOutputAttribute( hConsole, csbi.wAttributes,
      dwConSize, coordScreen, &cCharsWritten );

   /* put the cursor at (0, 0) */

   bSuccess = SetConsoleCursorPosition( hConsole, coordScreen );
   return;
}
```

Program 9 - Low Level C++ Interface (continued)

```
// what to do on ctrl-break

BOOL CleanUpAndExit()
{

   HANDLE hStdout;

       // grab a handle to the stdout

       hStdout = GetStdHandle(STD_OUTPUT_HANDLE);

    /* Restore the original text colors. */

    if (! SetConsoleTextAttribute(hStdout, wOldColorAttrs))
       MyErrorExit("SetConsoleTextAttribute");

       cls(hStdout);

       return TRUE;

}// control-break handler

BOOL CtrlHandler(DWORD fdwCtrlType) {
    switch (fdwCtrlType) {

        /* Handle the CTRL+C signal. */

        case CTRL_C_EVENT:

            Beep(1000, 1000);
            return TRUE;

        /* CTRL+CLOSE: confirm that the user wants to exit. */

        case CTRL_CLOSE_EVENT:

            return TRUE;

        /* Pass other signals to the next handler. */

        case CTRL_BREAK_EVENT:

                    CleanUpAndExit();

        case CTRL_LOGOFF_EVENT:

        case CTRL_SHUTDOWN_EVENT:

        default:

            return FALSE;

    }

}
```

Program 9 - Low Level C++ Interface (continued)

```
// this thread reads from the card and displays on
// stdout.


DWORD WINAPI ReadThread (LPVOID lpvThreadParm) {


LPSTR lpszPrompt1 = "";
LPSTR lpszPrompt2 = "ACR8000 Parallel Terminal\n";

HANDLE hStdout;


long cRead;
unsigned long cPresent;
DWORD cWritten;
CONSOLE_SCREEN_BUFFER_INFO csbi;


unsigned char chInBuffer[4096];

// grab a handle to the stdout

hStdout = GetStdHandle(STD_OUTPUT_HANDLE);

if (hStdout == INVALID_HANDLE_VALUE)
MyErrorExit("GetStdHandle");

// read and save the color information for stdout.

GetConsoleScreenBufferInfo(hStdout, &csbi);
wOldColorAttrs = csbi.wAttributes;

/* Set the text attr. to draw white on black background. */

if (! SetConsoleTextAttribute(hStdout, FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE ))
        MyErrorExit("SetConsoleTextAttribute");
```

Program 9 - Low Level C++ Interface (continued)

```
// clear the screen

cls(hStdout);

if (! WriteFile(
        hStdout,              /* output handle  */
        lpszPrompt2,          /* prompt string  */
        lstrlen(lpszPrompt2), /* string length  */
        &cWritten,            /* bytes written  */
        NULL) )               /* not overlapped */
    MyErrorExit("WriteFile");

while (1)
{

       // see if there are any elements in the read buffer.

       AcroGetDrReadBuffSize(&cPresent, 0);

       //If there are any elements, then read them into the text buffer

        if (cPresent > 0)
        {

                 // read a string from the card
                 AcroReceiveTextString(chInBuffer, cPresent, (long*)&cRead, 0);

                 // display on stdout
                 WriteFile(hStdout, (char *)chInBuffer, cRead,
                        &cWritten, NULL);

     }

        Sleep(0);


}

return(0);

}
```

Program 9 - Low Level C++ Interface (continued)

```
//this thread reads the stdin and writes it to the card.

DWORD WINAPI WriteThread (LPVOID lpvThreadParm) {


CHAR chBuffer[2];
DWORD cRead, cWritten, fdwMode, fdwOldMode;

HANDLE hStdin;
hStdin = GetStdHandle(STD_INPUT_HANDLE);

if (hStdin == INVALID_HANDLE_VALUE)
MyErrorExit("GetStdHandle");

// Set the Input Mode
// by disabling all controls

if (! GetConsoleMode(hStdin, &fdwOldMode))
    MyErrorExit("GetConsoleMode");


// turn off all preprocessing.

fdwMode = fdwOldMode &
         ~ENABLE_LINE_INPUT;

fdwMode = fdwMode &
          ~ENABLE_ECHO_INPUT;

fdwMode = fdwMode &
          ~ENABLE_PROCESSED_INPUT;


if (! SetConsoleMode(hStdin, fdwMode))
    MyErrorExit("SetConsoleMode");
```

Program 9 - Low Level C++ Interface (continued)

```
while(1)
{
        // read from stdin

        ReadFile(hStdin, chBuffer, 1, &cRead, NULL);

        if (cRead>0)
        {
          // write to the card
          AcroSendTextString((unsigned char*)chBuffer,1,(long *)&cWritten,0);
        }


       Sleep(0);

}

return(0);

}


void main()
{

  DWORD  ReadThreadId;
  DWORD  WriteThreadId;

  HANDLE hThreads[2];

// initialize the card.

  AcroInitialize(0);

  if(AcroGetError()!=ACRO_SUCCESS)
  {
  printf("Could not find the card.\n");
  return;
  }
```


PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

Program 9 - Low Level C++ Interface (continued)

```
//install console handler function

  BOOL fSuccess;

  fSuccess = SetConsoleCtrlHandler(
    (PHANDLER_ROUTINE) CtrlHandler,  /* handler function */
    TRUE);                           /* add to list      */
  if (! fSuccess)
    MyErrorExit("Could not set control handler");

  //card was found, create read and write threads.

  hThreads[0] = CreateThread(
                    NULL,               // LPSECURITY_ATTRIBUTES lpsa,
                    0,                  // DWORD cbStack,
                    ReadThread,         // LPTHREAD_START_ROUTINE lpStartAddr,
                    NULL,               // LPVOID lpvThreadParm,
                    0,                  // DWORD fdwCreate,
                    ReadThreadId        // LPDWORD lpIDThread
                             );

 if (hThreads[0]==NULL)
 {
       printf("Could not create Read Thread %i.\n",0);
 }

  hThreads[1] = CreateThread(
                    NULL,               // LPSECURITY_ATTRIBUTES lpsa,
                    0,                  // DWORD cbStack,
                    WriteThread,        // LPTHREAD_START_ROUTINE lpStartAddr,
                    NULL,               // LPVOID lpvThreadParm,
                    0,                  // DWORD fdwCreate,
                    &WriteThreadId      // LPDWORD lpIDThread
                            );

if (hThreads[1]==NULL)
 {
       printf("Could not create Write Thread %i.\n",1);
 }

 // wait for both threads to exit

 WaitForMultipleObjects(2, hThreads, TRUE, INFINITE);

}
```

Program 9 - Low Level C++ Interface (continued)

Terminal.mak:

```
# Microsoft Developer Studio Generated NMAKE File, Format Version 4.10
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Console Application" 0x0103

!IF "$(CFG)" == ""
CFG=terminal - Win32 Debug
!MESSAGE No configuration specified.  Defaulting to terminal - Win32 Debug.
!ENDIF

!IF "$(CFG)" != "terminal - Win32 Release" && "$(CFG)" !=\
 "terminal - Win32 Debug"
!MESSAGE Invalid configuration "$(CFG)" specified.
!MESSAGE You can specify a configuration when running NMAKE on this makefile
!MESSAGE by defining the macro CFG on the command line.  For example:
!MESSAGE
!MESSAGE NMAKE /f "terminal.mak" CFG="terminal - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "terminal - Win32 Release" (based on\
 "Win32 (x86) Console Application")
!MESSAGE "terminal - Win32 Debug" (based on "Win32 (x86) Console Application")
!MESSAGE
!ERROR An invalid configuration is specified.
!ENDIF

!IF "$(OS)" == "Windows_NT"
NULL=
!ELSE
NULL=nul
!ENDIF
################################################################################
# Begin Project
# PROP Target_Last_Scanned "terminal - Win32 Debug"
RSC=rc.exe
CPP=cl.exe

!IF  "$(CFG)" == "terminal - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Target_Dir ""
OUTDIR=.\Release
INTDIR=.\Release
```

Program 9 - Low Level C++ Interface (continued)

```
ALL : "$(OUTDIR)\terminal.exe"

CLEAN :
       -@erase "$(INTDIR)\terminal.obj"
       -@erase "$(OUTDIR)\terminal.exe"
       -@erase "$(OUTDIR)\terminal.pdb"

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

# ADD BASE CPP /nologo /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CONSOLE" /YX /c
# ADD CPP /nologo /Zp4 /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CONSOLE" /c
# SUBTRACT CPP /YX
CPP_PROJ=/nologo /Zp4 /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CONSOLE"\
 /Fo"$(INTDIR)/" /c
CPP_OBJS=.\Release/
CPP_SBRS=.\.
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
BSC32_FLAGS=/nologo /o"$(OUTDIR)/terminal.bsc"
BSC32_SBRS= \

LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib
/nologo /subsystem:console /machine:I386
# ADD LINK32 ..\..\..\lib\Release\acrownt.lib kernel32.lib user32.lib gdi32.lib
winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:console /debug /machine:I386
# SUBTRACT LINK32 /incremental:yes
LINK32_FLAGS=..\..\..\lib\Release\acrownt.lib kernel32.lib user32.lib gdi32.lib\
 winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib\
 uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:console /incremental:no\
 /pdb:"$(OUTDIR)/terminal.pdb" /debug /machine:I386\
 /out:"$(OUTDIR)/terminal.exe"
LINK32_OBJS= \
       "$(INTDIR)\terminal.obj"

"$(OUTDIR)\terminal.exe" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ELSEIF  "$(CFG)" == "terminal - Win32 Debug"
```

Program 9 - Low Level C++ Interface (continued)

```
# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Target_Dir ""
OUTDIR=.\Debug
INTDIR=.\Debug

ALL : "$(OUTDIR)\terminal.exe"

CLEAN :
	-@erase "$(INTDIR)\terminal.obj"
	-@erase "$(INTDIR)\vc40.idb"
	-@erase "$(INTDIR)\vc40.pdb"
	-@erase "$(OUTDIR)\terminal.exe"
	-@erase "$(OUTDIR)\terminal.pdb"

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

# ADD BASE CPP /nologo /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D "_CONSOLE" /YX /c
# ADD CPP /nologo /Zp4 /MT /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D "_CONSOLE" /c
# SUBTRACT CPP /YX
CPP_PROJ=/nologo /Zp4 /MT /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D\
 "_CONSOLE" /Fo"$(INTDIR)/" /Fd"$(INTDIR)/" /c
CPP_OBJS=.\Debug/
CPP_SBRS=.\.
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
BSC32_FLAGS=/nologo /o"$(OUTDIR)/terminal.bsc"
BSC32_SBRS= \

LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib
/nologo /subsystem:console /debug /machine:I386
# ADD LINK32 ..\..\..\lib\Debug\acrownt.lib kernel32.lib user32.lib gdi32.lib
winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:console /incremental:no /debug
/machine:I386
LINK32_FLAGS=..\..\..\lib\Debug\acrownt.lib kernel32.lib user32.lib gdi32.lib\
 winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib\
 uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:console /incremental:no\
 /pdb:"$(OUTDIR)/terminal.pdb" /debug /machine:I386\
 /out:"$(OUTDIR)/terminal.exe"
LINK32_OBJS= \
	"$(INTDIR)\terminal.obj"
```

Program 9 - Low Level C++ Interface (continued)

```
"$(OUTDIR)\terminal.exe" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ENDIF

.c{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.cpp{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.cxx{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.c{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

.cpp{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

.cxx{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

################################################################################
# Begin Target

# Name "terminal - Win32 Release"
# Name "terminal - Win32 Debug"

!IF  "$(CFG)" == "terminal - Win32 Release"

!ELSEIF  "$(CFG)" == "terminal - Win32 Debug"

!ENDIF

################################################################################
# Begin Source File

SOURCE=.\terminal.cpp
DEP_CPP_TERMI=\
       "..\..\..\inc\acrolib.h"\
       ".\terminal.h"\


"$(INTDIR)\terminal.obj" : $(SOURCE) $(DEP_CPP_TERMI) "$(INTDIR)"


# End Source File
# End Target
# End Project
################################################################################
```

Example 10 - Visual Basic Programming

The Acroloop controllers can also communicate to the PC using a Visual Basic interface. In general, the Acroloop controllers are able to send and receive information to and from the board. The information is sent and received using both the binary and ASCII communication interface schemes(See Interface Schemes).

The following program sends the current position to the Visual Basic front end to be displayed. The program is based on a Visual Basic timer. In this case, the timer can be set up to retrieve current position every 100 milliseconds. 100 milliseconds is used since the human eye cannot detect faster changes.

```
Private Sub Timer1_Timer()

Static ElementsPresent As Long
Static Temp2 As String
Static PPUnit As Double
Static VelArray(8) As Single
Static Number As Single
Static ClipText As String
Static Group As Long
Static Index As Long
Static BitParm As Long
PPUnit = 10000


'Display Position for 8 Axes
AcroLookLong CLng(&H30), CLng(&H6), CLng(&HFF), Acpos(0), 0
For i = 0 To 7
Temp = Format(Acpos(i) / PPUnit, "0000.0000")
If Form5.label1(i).Caption <> Temp Then Form5.label1(i).Caption = Temp
Next i


'Display Parameter Status for 8 Parameters
Group = ParamGroupCodes(GroupSelect + 1)
Index = ParamGroupIndexStart(GroupSelect + 1) + ParamSelect
mask = &HFF


If Not ParameterDisplayHold Then

Select Case ParamTypes(GroupSelect + 1)
```

Example 10 - Visual Basic Programming (continued)

```
Case LongSingle, LongFlag, LongSet

   AcroLookLong CLng(Group), CLng(Index), CLng(mask), LongParm(0), 0

   For i = 0 To 7
      Temp = Format(LongParm(i), "00000000")
      If Form2.Text1(i).Text <> Temp Then
         Form2.Text1(i).Text = Temp
      End If
   Next i

   'Display Bit Status for 1 Parameter
   BitParm = LongParm(BitSelect)
   For i = 0 To 31
      If (BitParm And 1) Then
         Form4.Shape4(i).FillStyle = 0
         Else: Form4.Shape4(i).FillStyle = 1
         End If
      BitParm = BitParm \ 2
   Next i

Case FP32

   AcroLookFP CLng(Group), CLng(Index), CLng(mask), FPParm(0), 0
   For i = 0 To 7
   Temp = Format(FPParm(i), "0.0000000E+00")
   If Form2.Text1(i).Text <> Temp Then
     Form2.Text1(i).Text = Temp
   End If
   Next i
   BitParm = FPParm(BitSelect)

End Select

End If

'Check if there are any data needing to be displayed
```

Example 10 - Visual Basic Programming (continued)

```
While (AcroTextBufferNumberOfElements(0) > 0)

   ElementsPresent = AcroTextBufferNumberOfElements(0)
   MyError = AcroReceiveTextString(Temp, ElementsPresent, ReceivedElements, 0)

   If Left(Temp, 1) <> Chr(8) Then

     ClipText = Left(Temp, ReceivedElements)

    Form3.Terminal.SelStart = 1 + Len(Form3.Terminal.Text)
    Form3.Terminal.SelLength = 1
    Form3.Terminal.SelText = ClipText

   Else

    ClipText = ""
    If Form3.Enabled Then
       '6/26/96
        If Len(Form3.Terminal.Text) > 1 Then
        Form3.Terminal.SelStart = Len(Form3.Terminal.Text) - 1
        End If

        Form3.Terminal.SelLength = 1
        Form3.Terminal.SelText = ClipText
    End If
   End If

  Form3.Terminal.SelStart = 1 + Len(Form3.Terminal.Text)
  Form3.Terminal.SelLength = 1

Wend

End Sub
```

The Visual Basic program is included as part of the Visual Basic sample program available from Acroloop free of charge. The sample program also includes subroutines for the following:

1. Terminal Emulator
2. Bit Flag Status
3. Parameter Status
4. Current Position Display

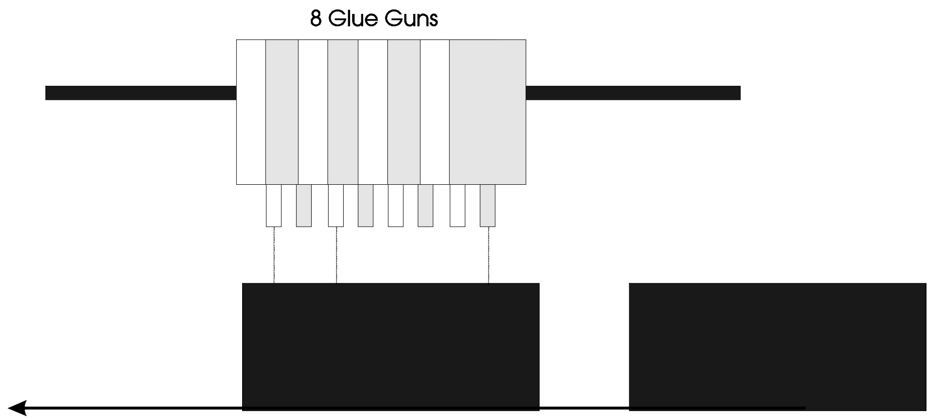
The sample program source code is included so that the programmer can modify the code for customized API programming.

Program 11 - Programmable Limit Switch

The Acroloop controllers are ideal for high speed glue gun, paint gun, or rotary cam applications. **The controllers are supplied with up to 8 Programmable Limit Switch (PLS) function command sets**. The 8 PLS's allow the digital I/O to be updated every interrupt of the board. The controller interrupt period is programmable from 200 - 500 microseconds. Thus, the update time (scan rate) of the digital I/O is 0.2 to 0.5 milliseconds. The default interrupt speed is 0.5 milliseconds.

Consider a high speed Glue Gun application. The applications consists of controlling 8 glue guns to apply glue to boxes traveling past the glue guns. The 8 Glue Guns will be controlled by 1 of the 8 PLS functions(i.e. PLS0).

Assume that the velocity of the boxes is 1 meter/second or 1000 mm/sec. The default interrupt of the controller is 0.5 milliseconds; the digital outputs are updated every 0.5 milliseconds. Thus, the accuracy of the firing of the digital outputs is
1000mm/sec * 0.0005 sec = 0.5 mm.



Glue Gun PLS Application

The are several factors to consider in programming the PLS.

1. How may entries for the PLS table. The number of entries in the PLS table determine the size of the array to dimension in memory. The array is used to change the state of the outputs. In this example, assume that there are 8 different combinations of glue guns to be fired; the 8 glue guns will change state in 9 different intervals **(DIM LA0(9)).**

2. What is the maximum count of the PLS source (PLS SRC). Typically, the source will be an encoder. In this example, assume that the number of pulses between the **beginning on one box to the beginning of the next box is 2000 pulses.**

Program 11 - Programmable Limit Switch (continued)

3. Calculate the scaling ratio. The **PLS RATIO** is calculated as: ((Entries - 1)/ maximum pulse count). In this example, the PLS RATIO is (9-1) / 2000 pulses = 0.004.

4. Determine which outputs will be controlled by the PLS. By default, the PLS will control the standard 32 digital outputs on the controller. In this example, the PLS will control only 8 outputs; the controller will be programmed to control the specific 8 outputs by using the **PLS MASK** command.

   The digital outputs can be represented in binary or hexadecimal form. The binary representation for the first 8 outputs would be 00000000000000000000000011111111 or hex 000000FF. The binary or hex form corresponds to the decimal number 255.

5. Determine the combination of outputs to be triggered. We have determined that 8 outputs will be fired based on a maximum encoder count of 2000 pulses. Within, the 2000 pulses, we want to fire specific outputs. The binary form of the 32 outputs is 11111111111111111111111111111111 or in hex FFFFFFFF. To fire specific outputs, represent the outputs in binary or hex form and calculate the decimal value. For example, to fire the first 4 outputs of the total of 8 outputs used, the binary form is 00001111; the decimal value is then 15. The decimal values are the array values **(LA0(n)).**

6. Determine the destination pointer for the PLS. The PLS points to an array. To specify which array is to be used in the PLS, the **PLS BASE** command identifies which array is associated with the PLS.

7. Determine the range of the source encoder. The **PLS ROTARY** command wraps the encoder count back to zero once the maximum encoder count is reached. In this example, the PLS ROTARY is equal to 2000 pulses.

   Now that the PLS is set up, the commands can be written to the controller using the AcroVIEW communication tool or any other terminal device.

   The PLS commands will control the 8 glue guns fired in combination based on the encoder pulse position and the combination of glue guns desired. The commands will be made from the program 0 (P00>) prompt:

```
CLEAR                  REM: Clears memory allocated for local variables/arrays.
DIM LA(1)              REM: Allocate memory for one long integer array
DIM LA0(9)             REM: Allocate memory for 9 entries in the array LA0.
PLS0 BASE LA0          REM: Use the data in LA0 for the PLS.
PLS0 SRC ENC1          REM: Set encoder input 1 as the source encoder.
PLS0 MASK 255          REM: The first 8 outputs will be used for the PLS.
PLS0 RATIO 0.004       REM: Encoder counts per firing of outputs ratio
PLS0 ROTARY 2000       REM: After 2000 counts, wrap pointer to the beginning.
```

Program 11 - Programmable Limit Switch (continued)

The array values (specific outputs to be fired) are as follows:

```
LA0(0) = 1        REM: Fire first glue gun
LA0(1) = 15       REM: Fire first 4 glue guns
LA0(2) = 63       REM: Fire first 6 glue guns
LA0(3) = 255      REM: Fire all 8 glue guns
LA0(4) = 252      REM: Fire last 6 glue guns
LA0(5) = 240      REM: Fire last 4 glue guns
LA0(6) = 128      REM: Fire last glue gun only
LA0(7) = 0        REM: Wait for next box
LA0(8) = 0        REM: Wait for next box
```

The following table helps to clarify the specific outputs fired at the specific encoder positions:

Encoder Position	Array Value	Outputs Fired
0	1	00000001
250	15	00001111
500	63	00111111
750	255	11111111
1000	252	11111100
1250	240	11110000
1500	128	10000000
1750	0	00000000
2000	0	00000000

PLS Output Firing Sequence

Finally, the command to energize the programmable limit switch would be executed.

PLS0 ON

Program 11 - Programmable Limit Switch (continued)

The Acroloop controller family has the ability to have up to 8 PLS functions active simultaneously. **The Programmable Limit Switch function can also be set up to look at a “dumb” axis.**

In this example, the PLS will be activated from an internal source and not an encoder source. Then, the updating of the outputs of the PLS is dependent on the jog offset parameter. In other words, as the “dumb” axis is jogged forward, no actual motion is observed from the system, thus creating a jog offset. As the jog offset is increased, the PLS updates the outputs per the array LA0. **The purpose of this example is to be able to update the outputs at high speeds without actually having actual motion in the system.**

The following example illustrates the internal source for the PLS.

```
CLEAR                          REM:  Clears memory allocated for local variables/arrays.
DIM LA(1)                      REM:  Allocate memory for one long integer array
DIM LA0(9)                     REM:  Allocate memory for 8 entries in the array LA0.
PLS7 BASE LA0                  REM:  Use the data in LA0 for the PLS.
PLS7 SRC P13321                REM:  Source for PLS is the JOG OFFSET for AXIS4
PLS7 MASK 255                  REM:  The first 8 outputs will be used for the PLS.
PLS7 RATIO 0.004               REM:  Encoder counts per firing of outputs ratio
PLS7 ROTARY 2000               REM:  After 2000 counts, wrap pointer to the beginning.
PLS7 ON                        REM:  Turn on the PLS.

PROG7                          REM:  Set up program space number 7.
HALT                           REM:  Halt any running programs.
NEW                            REM:  Declare a new program.
DETACH                         REM:  Cancel any previous attachments.
CLEAR                          REM:  Clear the PLS channels.

ATTACH MASTER7                 REM:  Set up profiler number 7.
ATTACH SLAVE0 AXIS4 “DUMB”     REM:  Set up the DUMB axis.

JOG VEL DUMB 1000              REM:  To change speed of PLS, set jog velocity
JOG FWD DUMB                   REM:  Activate the PLS, jog the axis forward
```

Theoretically, the source of the PLS can be tied to any of the hardware parameters on the Acroloop controller family.

Program 12 - Data Acquisition

In many cases, status information must be gathered from the board. The Acroloop controller family is supplied with **built-in hardware registers to store the on-board data**. For example, the current position of Axis 0 is stored in hardware register P12288 (See appendix for parameters). The Acroloop controllers are the only motion controllers that have pre-programmed hardware registers that the user can access directly. Each of the hardware registers is updated every 0.5 milliseconds.

The data stored in the hardware registers can be sampled by the user. In most applications, the data is captured in the hardware registers and then transferred to the front end operator interface over the PC-BUS. Capturing the data in hardware is the preferred method when the data is time critical. Data can be captured on the board at extremely high rates; **the default data sampling rate is 2KHz or every 0.5 milliseconds.**

Data sampling allows the monitoring of system parameters at the servo interrupt rate (0.5 milliseconds). Optionally, data may be also sampled at a fixed frequency programmable to 25 milliseconds or the rising or falling edge of a bit flag (see appendix for flags).

A total of eight channels can be simultaneously filled from different parameter sources. For example, one channel can monitor actual position while another is monitoring output voltage for a given axis. The resulting **information can be stored in on-board arrays** and transferred to an off line system for graphical plotting or analysis. The data can be transferred over any communication port on the board.

The sample command is combined with other commands to prepare the system for data sampling. The following is a list of valid sample command combinations:

Commands	Description
SAMP SRC	Set sample source
SAMP BASE	Set sample base array
SAMP CLEAR	Clear sample channels
SAMP TRG	Set sample trigger(start)

The sample command includes a set of system parameters for data sampling:

Parameter	Description
P6912	Sample array index
P6913	Sample trigger index
P6914	Sample timer clock
P6915	Sample timer period

Sample Command Related Flags

Flag	Description
BIT104	Sample trigger armed
BIT105	Sample in progress
BIT106	Sample mode select
BIT107	Sample trigger latched

Program 12 - Data Acquisition (continued)

The following example takes 500 samples of axis 0 current position and the output voltage at the default servo interrupt rate of 2KHz(250 milliseconds total sampling time).

```
PROG0                        REM:  Set up program 0
HALT                         REM:  Halt program (Good practice)
DETACH                       REM:  Detach any attachments (Good practice)
ATTACH MASTER0               REM:  Set up master profiler 0
ATTACH SLAVE0 AXIS0 “X”      REM:  Set up the X-axis

CLEAR                        REM:  Clear memory of any variables/arrays
DIM LA(1)                    REM:  Dimension 32 bit integer array
DIM LA0(500)                 REM:  Dimension 500 index values
DIM SA(1)                    REM:  Dimension 32 bit floating point array
DIM SA0(500)                 REM:  Dimension 500 index values

SAMP CLEAR                   REM:  Clear the sample channels

SAMP0 SRC P12290             REM:  Sample source is X-axis current position
SAMP0 BASE LA0               REM:  Store values in array LA0
SAMP1 SRC P12319             REM:  Sample source is output signal
SAMP1 BASE SA0               REM:  Store values in array SA0

SAMP TRG 516                 REM:  Sample trigger is profiler 0 in motion flag
SET 104                      REM:  Arm the sample trigger flag
X1000                        REM:  Move the X-axis 1000 units
INH -104                     REM:  Inhibit program until sample complete
```

The Acroloop controller family has the hardware parameters and flags pre-programmed into hardware registers for direct use by the controller(i.e. Access to hardware parameters for data sampling). The controller can then transfer the information over any one of the 4 communication ports (PC-BUS, COM1, COM2, and LPT).

Program 13 - CNC Tapping

The Acroloop controllers are easily programmed to handle tapping applications. A typical tapping application would involve a **CNC machine tool in which a drilling and tapping operation is performed.**

In this example, we will assume that a simple drilled hole already exists at X-Y coordinate position (10,7). The implementation of the tapping requirement can be accomplished by using the board level GEAR command. **The GEAR command is then sourced to the Z spindle encoder to be able to tap the hole while the Z-axis is indexed**. We will also assume that the hole is drilled and tapped through a piece of metal; additional program statements could be added to take care of the boundary conditions if the hole were of a fixed depth into a piece of metal (i.e. To include inertia and mechanical offset conditions to stop the spindle and reverse the direction in exactly the same path).

```
PROG7                             REM:  Set up program 7
HALT                              REM:  Halt program (Good practice)
DETACH                            REM:  Detach any attachments (Good practice)
ATTACH MASTER7                    REM:  Set up master profiler 7
ATTACH SLAVE0 AXIS0 “X”           REM:  Set up the X-axis
ATTACH SLAVE1 AXIS1 “Y”           REM:  Set up the Y-axis
ATTACH SLAVE2 AXIS2 “Z”           REM:  Set up the Z-axis
ATTACH SLAVE3 AXIS3 “S”           REM:  Set up the spindle axis
ESAVE ALL                         REM:  Save attachments to EEPROM

NEW                               REM:  Start a new program
PPU X10000 Y10000 Z10000          REM:  All axes are 10000 pulses per inch
X10 Y7                            REM:  Move axes to target tapping position
Z0                                REM:  Move Z-axis to tapping height

GEAR SRC Z3                       REM:  Gear source is encoder 2(Z-axis)
GEAR PPU Z1000                    REM:  Master axis is 1000 pulses per revolution
GEAR RATIO Z 0.1                  REM:  Set gear ratio to 1:10 (0.1 inches per revolution)
GEAR ON Z                         REM:  Turn on the gear

SET 32                            REM:  Turn on the spindle CW.
IHPOS -P6192(10000, 2)            REM:  Inhibit program until Z encoder parameter gets to
                                        10000 pulses or times out at 2 seconds.
CLR 32                            REM:  Turn off the spindle
SET 33                            REM:  Turn on the spindle CCW
IHPOS P6192(0, 2)                 REM:  Back out off the tapped hole
```


PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

Program 14 - Circular and Helical Interpolation

The Acroloop controller family can **perform interpolated moves in any combination of up to 8-axis.** In this example, we will **demonstrate the circular and helical interpolation** capabilities of the motion controller. It is important to note that all of the interpolation calculations are performed on the motion controller by the floating point DSP. If the application requires interpolated moves and is a PC-BUS application, the host processor only needs to send the coordinate information and the type of move to the Acroloop motion controller, the controller takes care of everything else.

```
PROG5                          REM:  Set up program 5
HALT                           REM:  Halt program (Good practice)
DETACH                         REM:  Detach any attachments (Good practice)
ATTACH MASTER5                 REM:  Set up master profiler 5
ATTACH SLAVE0 AXIS0 “X”        REM:  Set up the X-axis
ATTACH SLAVE1 AXIS1 “Y”        REM:  Set up the Y-axis
ATTACH SLAVE2 AXIS2 “Z”        REM:  Set up the Z-axis
ESAVE ALL                      REM:  Save the attachments to EEPROM

NEW                            REM:  Start a new program
```

The command to perform circular(2-axis), elliptical(2-axis) and helical(3-axis) is the SINE command. The format for arc commands is as follows:

SINE {axis(target, phase, sweep, amplitude)}.

To generate a simple 90 degree circle, the commands are:

```
10  X0 Y0                      REM:  Linear move to 0,0
20  X10000                     REM:  Linear move to 10000,0
30  SINE X(0,90,90,10000)
    SINE Y(10000,0,90,10000)   REM:  Circular move to 0,10000
40  X0 Y0                      REM:  Linear to 0,0
```

To generate a simple helical move, the commands are:

```
10  X10000 Y0 Z0               REM: Move the axes to 10000,0,0
20  SINE X(10000,90,360,10000)
    SINE Y(0,90,360,10000)
    Z 20000                    REM: Perform a helical move to 10000,0,20000
30  X0 Y0 Z0                   REM: Linearly interpolated move to 0,0,0
```

Program 15 - Electronic Gearing

The Acroloop controller family is capable of electronic gearing. The electronic gearing function can attach up to 8 slave axes to a GEAR SOURCE on one board. The GEAR SOURCE can be any of the following:

1. Encoder number or encoder hardware register
2. Internal Servo Clock
3. Hardware parameter (See Appendix for list of hardware parameters)
4. Ratchet - Gearing mode similar to a socket wrench.

In this example the X and Y axes will follow encoder 4 at a 1:4 and 1:10 ratio. The slave axes (X, Y) will move 0.25 and 0.10 inches per revolution (IPR) of the electronic gearing source encoder.

```
PROG4                       REM:  Set up program 4
HALT                        REM:  Halt program (Good practice)
DETACH                      REM:  Detach any attachments (Good practice)
ATTACH MASTER5              REM:  Set up master profiler 5
ATTACH SLAVE0 AXIS0 “X”     REM:  Set up the X-axis
ATTACH SLAVE1 AXIS1 “Y”     REM:  Set up the Y-axis
ESAVE ALL                   REM:  Save attachments to EEPROM

NEW                         REM:  Start a new program

PPU X10000                  REM:  X-axis is 10000 pulses per inch
PPU Y10000                  REM:  Y-axis is 10000 pulses per inch

GEAR SRC X4 Y7              REM:  Source is encoder 4 for X, 7 for Y
GEAR PPU X1000 Y2500        REM:  Encoder 4 is 1000 PPR, 7 is 2500 PPR
GEAR RATIO X0.25 Y0.10      REM:  Set gear ratio at 1:4 and 1:10 respectively
GEAR ON X Y                 REM:  Turn on electronic gearing
```

The electronic gearing function is extremely flexible and can handle a wide variety of master/slave applications.

PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

Program 16 - Position Registration

The Acroloop controller family is capable of performing high speed position registration. This operation is important when precise information is required to know the encoder position when a proximity switch or other device is detected by the controller. The motion controllers high speed position registration occurs in hardware registers on the board. **Hardware registration is preferred when specific position registration is required; thus eliminating slow software execution and host CPU involvement.**

The controllers are capable of high speed registration initiated by either an encoder marker pulse or an external input device. The controllers are provided with on-board programmable gate arrays; the gate arrays are capable of handling hardware position registration. The on-board command INTCAP enables hardware position capture triggered from one of several different sources. **The latency for position registration using the encoder marker is less than 100 nanoseconds(1 microsecond latency using external input signal through the optical isolation. If the optical isolation on the digital inputs is bypassed, the latency is 100 nanoseconds.**

The INTCAP command can be operated in several different modes. The valid modes are:

0 - Trigger on Rising Edge of Primary Marker Pulse
1 - Trigger on Rising Edge of Secondary Marker Pulse
2 - Trigger on Rising Edge of External Primary Input
3 - Trigger on Rising Edge of Secondary External Input
4 - Trigger on Falling Edge of Primary Marker Pulse
5 - Trigger on Falling Edge of Secondary Marker Pulse
6 - Trigger on Falling Edge of Primary External Input
7 - Trigger on Falling Edge of Secondary External Input

The following example captures the X, Y, and Z-axis position from a proximity switch:

```
PROG6                         REM:  Set up program 6
HALT                          REM:  Halt program (Good practice)
DETACH                        REM:  Detach any attachments (Good practice)
ATTACH MASTER6                REM:  Set up master profiler 6
ATTACH SLAVE0 AXIS0 “X”       REM:  Set up the X-axis
ATTACH SLAVE1 AXIS1 “Y”       REM:  Set up the Y-axis
ATTACH SLAVE2 AXIS2 “Z”       REM:  Set up the Z-axis
ESAVE ALL                     REM:  Save attachments to EEPROM

NEW                           REM:  Start a new program
INTCAP X2 Y2 Z2               REM:  Hardware position capture of X, Y, Z-axis on rising
                                    edge of external input(Assume inputs 24, 25, and 26
                                    are all tied together for registration to proximity
                                    switch).
```

Please see the Users Guide for additional information.

Program 17 - External Timebase

The Acroloop controller family is capable of specifying the timebase for coordinated motion. For example, you may want to control the velocity of an XY system based on the speed of a trackball. The faster the trackball is moved by the operator, the faster the velocity of the XY-axis.

The SRC(source) command can be defined in any of the following formats:

Source Definition	Description
NONE	Disconnect device from source
CLOCK	Connect to servo clock(1 pulse per period)
ENC	Connect to encoder
RATCH	Connect to ratchet output
parameter	Connect to user or system parameter(hardware register)

During each servo interrupt, the change in source pulses is multiplied by the servo period and the resulting delta time is fed into the velocity profile mechanism. By default, the velocity profile is sourced off the CLOCK, feeding a single time unit per interrupt. Redirecting the source allows an external timebase to be used for coordinated motion.

```
PROG6                        REM:  Set up program 6
HALT                         REM:  Halt program (Good practice)
DETACH                       REM:  Detach any attachments (Good practice)
ATTACH MASTER6               REM:  Set up master profiler 6
ATTACH SLAVE0 AXIS0 “X”      REM:  Set up the X-axis
ATTACH SLAVE1 AXIS1 “Y”      REM:  Set up the Y-axis
MULT X4 Y4                   REM:  Multiplier is 4(quadrature encoder)
PPU X2000 Y3500              REM:  Setup encoder multipliers

10 SRC ENC3                  REM:  Source is encoder input 3, which is a trackball
20 X10 Y20                   REM:  XY coordinated motion based on trackball velocity
```

Program 18 - System Setup

The Acroloop controller family is easily setup to perform motion control. Provided with the Acroloop controller, is the AcroVIEW communication utility program. With the AcroVIEW disk are setup programs, which are **prewritten programs that monitor excess error, overtravel limit switches, amplifier faults, and, in general, configures the board**. An example of a 2-axis setup program is provided below.

```
CONFIG ENC2 DAC2 NONE NONE        REM:  Configure the board for 2 servo axes.
HALT ALL                          REM:  Halt all programs
NEW ALL                           REM:  New all programs
DETACH ALL                        REM:  Detach any program attachments

CLEAR                             REM:  Clear memory(all programs must be empty)

DIM PROG0(5000)                   REM:  Dimension 40K memory (5K x 8) for program 0
DIM PROG1(5000)                   REM:  Dimension 40K memory for program1
DIM PROG2(3000)                   REM:  Dimension 24K memory (3K x 8) for program 2
DIM PROG3(3000)                   REM:  Dimension 24K for program 3
DIM PLC0(5000)                    REM:  Dimension 40K for PLC program 0
DIM PLC1(5000)                    REM:  Dimension 40K for PLC program 1
DIM PLC2(1000)                    REM:  Dimension 8K for PLC program 2
DIM PLC3(1000)                    REM:  Dimension 8K for PLC program 3
DIM P(10)                         REM:  Dimension 10 global variable(P1..P10)

ATTACH AXIS0 ENC0 DAC0            REM:  Setup axis0
ATTACH AXIS1 ENC1 DAC1            REM:  Setup axis1
AXIS2 OFF                         REM:  Turn off unused axis
AXIS3 OFF                         REM:  Turn off unused axis
AXIS4 OFF                         REM:  Turn off unused axis
AXIS5 OFF                         REM:  Turn off unused axis
AXIS6 OFF                         REM:  Turn off unused axis
AXIS7 OFF                         REM:  Turn off unused axis

PROG0                             REM:  Switch to program 0
ATTACH MASTER0                    REM:  Setup profiler 0
ATTACH SLAVE0 AXIS0 "X"           REM:  Setup the X-axis
ATTACH SLAVE1 AXIS1 "Y"           REM:  Setup the Y-axis

MULT X4 Y4                        REM:  Set the encoder multipliers for X and Y
PPU X2000 Y2000                   REM:  Set pulses per inch for X and Y
F300                              REM:  Set the feedrate(alternative is VEL command)
ACC10 DEC10 STP10                 REM:  Set the acceleration, deceleration & stop ramps
JOG VEL X10 Y10                   REM:  Set the jog velocity
JOG ACC X20 Y20                   REM:  Set the jog acceleration
JOG DEC X20 Y20                   REM:  Set the jog deceleration
JOG STP X20 Y20                   REM:  Set the jog stop ramp
EXC X+1.0,-1.0 Y1.0 Y-1.0         REM:  Set the excess error band
```

Program 18 - System Setup (continued)

```
PLC0                 REM:  Switch to PLC program 0
10 LD 30             REM:  Look at input 30 for E-Stop button
20 OUT 62            REM:  If E-Stop, the set output 62

100 LD NOT 769       REM:  Load excess error flag for X-axis
110 OR NOT 801       REM:  Or excess error flag for Y-axis
120 OUT 61           REM:  Set output 61 if excess error flags tripped

200 LD 8             REM:  Load input 8 as overtravel limit switch for X-axis
210 OR 9             REM:  Or input 9 as overtravel limit switch for Y-axis
220 OUT 59           REM:  Set output 59 if overtravel

400 LD 16            REM:  Load input 16 as drive fault input X-axis
410 OR 17            REM:  Or input 17 as drive fault input Y-axis
420 OUT 60           REM:  Set output 60 if in drive fault

600 LD 59            REM:  Load overtravel condition
610 OR 60            REM:  Or drive fault condition
620 OR 61            REM:  Or Excess Error limit condition
630 OR 62            REM:  Or E-Stop condition
640 OUT 520          REM:  Issue a feedhold request for Master profiler 0
650 OUT 522          REM:  Issue a kill move request for Master profiler 0
660 OUT 63           REM:  Set hard output 63(To E-Stop Relay wiring)

RUN
```

Programming Note: The inputs and outputs used in this program are completely programmable and must be connected and programmed properly for correct operation.

PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

Program 19 - String Handling

The Acroloop controller family is provided with a powerful string handler. The controllers are also capable of communicating over the PC-Bus and serial ports simultaneously. This allows the **flexibility to be running a PC-Bus application and simultaneously monitoring a serial port(s) for input data. With the string handling functions on the controller a host of applications are possible.**

A possible application is to monitor the serial port for changing velocity on the fly. This sample program communicates to a dumb terminal with a separate multi-tasked program monitoring 1 of the serial ports. When a string is detected, the program converts the string to its ASCII equivalent and then could change the velocity of another multi-tasked program on-the-fly.

```
SYS
PROG4
HALT
NEW

10 OPEN "COM1:9600,O,7,1" AS #1        REM:   Open COM1 and setup
20 PRINT #1, CHR$(12)                  REM:   Enter/Return key

50 PRINT
60 PRINT "   Waiting   ";              REM:   Wait for "wake-up"
70 DWL 5                               REM:   Dwell for 5 seconds

100 PRINT #1, "HELLO THERE";           REM:   Print to dumb terminal
120 DWL 1                              REM:   Dwell for 1 second
130 PRINT #1, CHR$(12)                 REM:   Enter/Return key

150 PRINT
160 PRINT "Enter New Velocity  ";      REM:   Look for velocity
180 INPUT #1, "Velocity = ", $V0       REM:   Input velocity as a string variable

190 DV0 = VAL($V0)                     REM:   DV0 equals value of string
200 IF (DV0 = 0) THEN GOTO 400         REM:   Check for zero velocity
210 DV1 = DV0                          REM:   DV1 = DV0

400 PRINT "New Velocity is "; DV1      REM:   Echo the velocity
410 DWL 1                              REM:   Dwell for 1 second
420 GOTO 180                           REM:   Watch for new velocity

DIM $V(1,32)                           REM:   Dimension string variables
DIM DV(2)                              REM:   Dimension double variables
```

Program 20 – Sewing Setup Programming

The Acroloop controller family can easily handle high-speed sewing applications using unique on-board features. Consider an XY sewing machine table with a Z-axis sewing head. In addition, the Z-axis sewing head must be synchronized with the sewing loop while maintaining consistent sewing stitch length.

The Acroloop controller would be setup as follows:

```
SYS
PROG0
HALT
NEW

ATTACH MASTER0                  REM:  Setup profiler number 0.
ATTACH SLAVE0 AXIS0 “X”         REM:  Setup the X-axis.
ATTACH SLAVE1 AXIS1 “Y”         REM:  Setup the Y--axis.
PROG2
HALT
NEW
ATTACH MASTER1
ATTACH SLAVE2 AXIS2 “Z”         REM:  Setup the Z-axis.
ATTACH SLAVE3 AXIS3 “A”         REM:  Setup the slave sewing loop axis.

LOCK ZA                         REM:  Lock the Z and A-axis together.  The LOCK
                                      command re-directs the primary setpoint from the
                                      Z-axis to the A-axis.

JOG VEL Z P8193                 REM:  Set the jog velocity of the Z-axis and subsequently
                                      the A-axis to equal the vector velocity of the XY
                                      coordinate system.  Parameter number P8193 is the
                                      vector velocity of MASTER0 and is a built-in
                                      parameter on the Acroloop board.  P8193 is
                                      calculated on-the-fly every interrupt of the board.

JOG FWD Z                       REM:  As the vector velocity of the XY coordinate system
                                      becomes a non-zero value, move the Z-axis forward.
                                      The A-axis will follow automatically.
```

Program 21 – Packaging Machine Application

The Acroloop controller family is provided with a high speed interrupt command. The command is specifically designed for applications involving detection of a registration mark on-the-fly and then indexing a programmable distance on the appropriate axis. The high speed interrupt command is ideal for packaging, labeling, roll feed and indexing style applications.

The HSINT command initiates a high speed interruptible move. The latency of the move is 1 interrupt or < 0.5 msec. The HSINT sequence consists of an incremental or absolute move with a capture window in the middle of it. Within the capture window, an internal hardware capture is initiated and monitored. If a capture occurs within the window, the current move is killed and a second move is started.

```
SYS
PROG7
HALT
NEW
DETACH

ATTACH MASTER0
ATTACH SLAVE0 AXIS0 “FEED”
```

The following example starts an incremental HSINT sequence with a rising primary external capture input, a total move distance of 100000 units, a move after capture of 50000 units, a capture window with a width of 20000 units starting 10000 units into the move, and monitoring input 9 for an external abort signal. All of this is accomplished with a single command line.

```
10 HSINT FEED/(2, 100000, 50000, 20000, 10000, 9)
```


PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

Program 22 – Pick and Place Machines

Typical pick and place applications may involve vision systems. The goal of the machine is to pick up a part and with a commanded move pass the part over a vision inspection station and continue to travel on the initial commanded path. The vision system then inspects the part to evaluate the exact orientation of the part. An offset is calculated by the vision system and the corresponding offset is sent to the motion controller. The commanded position (current position) and the offset position (jog offset) can be handled simultaneously on-the-fly with the Acroloop controller. The current position and jog offset are independent hardware registers that are summed to form the primary setpoint. The final result is a pick and place machine with increased throughput and accuracy. Other controllers would have to perform 2 moves to accomplish the task and therefore cannot match the efficiency of the Acroloop system.

Setpoint Summation (Simultaneous Position, CAM, Gear, and Jog Commands)

The Acroloop controller would be setup as follows:

```
PROG0
HALT
NEW

ATTACH MASTER0                  REM:  Setup profiler number 0.
ATTACH SLAVE0 AXIS0 "X"         REM:  Setup the X-axis.
ATTACH SLAVE1 AXIS1 "Y"         REM:  Setup the Y-axis.
ATTACH SLAVE2 AXIS2 "Z"         REM:  Setup the Z-axis.

PPU X10000 Y10000 Z2000         REM:  Setup the pulses per mm for each axis

10 X0 Y0 Z0                     REM:  Move to initial pickup point
20 Z-50                         REM:  Move Z-axis to pick-up part
30 SET 32                       REM:  Activate air pressure to pick up part
40 X207 Y205 Z0                 REM:  Move to target position passing vision system
50 JOG FWD X(P1) Y(P2)          REM:  Adjust X & Y-axis movement on-the-fly
60 Z-50                         REM:  Place the part
70 CLR 32                       REM:  Release air pressure
80 X0 Y0 Z0                     REM:  Move back to initial position
```

Note: Line 50 could be replaced with a JOG REV (Jog Reverse) command. The system could be programmed to perform the correct sub-routine based on whether the initial position is forward or backward to target position.

Program 23 – Automatic Corner Velocity Control

In many applications it is desirable to automatically control the vector velocity at a corner. Applications include laser, waterjet, plasma, and CNC milling applications. The final velocity command implements automatic corner slowdown for the Acroloop controllers.

The final velocity command controls the "slowdown" velocity between 2 moves. The following example generates a path using the final velocity (FVEL) command.

```
PROG0
HALT
NEW
DETACH

ATTACH MASTER0
ATTACH SLAVE0 AXIS0 "X"
ATTACH SLAVE1 AXIS1 "Y"
```

10 ACC 1000 DEC 1000 VEL 3000	REM:	Setup velocity, acceleration, and deceleration settings
20 FVEL 2000 STP500 X/19000	REM:	Move the X-axis incrementally (/) 19000 units with a slowdown ramp (STP) of 500 units/second^2. The final velocity is 2000 pulses/second after the slowdown.
30 Y/20000	REM:	Move the Y-axis incrementally 20000 units. The move would ramp back to the velocity of 3000 units/second using the master acceleration of 1000 pulses/second^2.

The above example simulates a corner in the XY plane. The Acroloop controllers also support the use of the final velocity command with the binary interface. Thus, binary commands can be sent to the board with the appropriate final velocity or corner slowdown vector velocities.

CONTROLLER SPECIFICATIONS

Item	Specification
CPU:	32/64 bit Floating Point DSP @ 60 MHz (ACR8010) ACR2000 is 50MHz, ACR1500 is 40MHz, ACR1200 is 40 MHz, ACR8000 is 27MHz
Processor Type:	Texas Instruments TMS320C3X Family
Servo Update Rate:	50 microseconds per axis
Size:	1.5 ISA slots (ACR8000) 1 ISA slot (ACR8010) 0.5 ISA slots(ACR2000, ACR1500) 8"w x 5"H (ACR1200)
Axis Configurations:	2, 4, 6, or 8-axis configurations (ACR8010/ACR8000) 2 or 4-axis configurations (ACR2000/ACR1500) 1 or 2-axis configurations (ACR1200)
Operating Temperature:	0ºC to 55ºC (32ºF to 132ºF)
Humidity:	0% to 95% non-condensing
Power Consumption:	+5 VDC @ 2.3 A (ACR8000/ACR8010) +5 VDC @ 1.5A (ACR1200/ACR1500/ACR2000) -12 VDC @ 0.13 A (ACR8000) -12 VDC @ 0.15 A (ACR1200/ACR1500/ACR2000/ACR8010) +12 VDC @ 0.13 A (ACR8000) +12 VDC @ 0.15 A (ACR1200/ACR1500/ACR2000/ACR8010)
External I/O Power Supply Requirements:	+24 VDC @ 2.0A (TTL digital I/O for ACR1500)
Battery Backed RAM:	1000mA-Hr Smart Battery (ACR8000) 1000mA-Hr Lithium Battery (ACR1200/ ACR2000/ACR8010) plus Automatic Recharge Circuitry for 50mA-Hr Vanadium-Lithium Battery (ACR1200/ACR2000/ACR8010)
EPROM (firmware):	Two 128K x 16 EPROM's (ACR1200/ACR2000/ACR8000/ACR8010) One 256K x 16 EPROM (ACR1500)
Critical Memory:	32K x 8 EEPROM Memory (ACR8000) 512K x 8 Flash Memory (ACR1500/ACR2000) 1024K x 8 Flash Memory (ACR1200/ACR8010)

PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

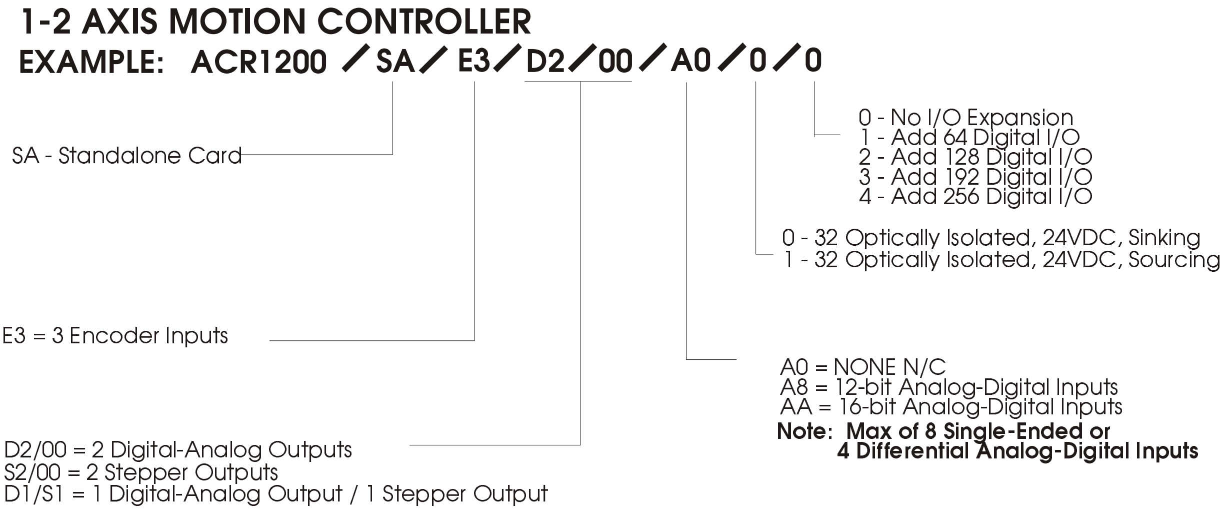
Item	Specification
Encoder Inputs:	Differential Quadrature Encoder Open-Collector or Line Driver 0.1 Hz to 8 MHz Frequency Range (ACR2000/ACR8000) 0.1 Hz to 20 MHz Frequency Range (ACR1200/ACR1500/ACR8010) 100mA maximum power source per channel Encoder Input Check Circuitry (ACR1200/ACR8010)
Analog:	+/- 10 VDC @ 5mA maximum, 16 bit resolution Programmable Output (DAC GAIN, DAC OFFSET)
Stepper:	4MHz maximum velocity 50% duty cycle (125 nsec - 16 usec pulse width range) Low current enable
Feedback Types:	Any Differential 5VDC or 12 VDC, including: - Quadrature Encoder - Glass Scales - Interferometer
Watchdog Relay:	+24 VDC @ 1.0 A (ACR1200/ACR2000/ACR8000/ACR8010) Open Collector Output @ 30mA, max. (ACR1500)
Digital Outputs:	24VDC, optically-isolated (ACR1200/ACR2000/ACR8000/ACR8010) - 50mA full load (maximum) - 125mA – up to 13 outputs (ACR8000/ACR8010) - 125mA – up to 6 outputs (ACR1200/ACR2000) - Sink Only (ACR8000) - Sink or Source (ACR1200/ACR2000/ACR8010) TTL I/O, compatible with 50-pin optical isolation boards (ACR1500)
Digital Inputs:	24VDC, optically-isolated (ACR1200/ACR2000/ACR8000/ACR8010) - Diode protected - Activates on 10mA per input - Sink Only (ACR8000) - Sink or Source (ACR1200/ACR2000/ACR8010) TTL I/O, compatible with 50-pin optical isolation boards (ACR1500)
A/D Inputs:	Up to 8 single-ended or 4 differential Programmable configuration (+/- 10VDC range) 12-Bit A/D Option - ACR1200/ACR1500/ACR2000/ACR8000/ACR8010 - 1.2 microseconds conversion time (per input) 16-Bit A/D Option - ACR1200/ACR1500/ACR8000/ACR8010 - 14 microseconds conversion time (per input)

CONTROLLER SPECIFICATIONS

Item	Specification
Communications:	PC-Bus (ACR1500/ACR2000/ACR8000/ACR8010) COM1, COM2, LPT (ACR1200/ACR2000/ACR8000/ACR8010) Simultaneous Communications on all ports on board
Serial Communications	2 ports (COM1, COM2) (ACR1200/ACR2000/ACR8000/ACR8010) Configurable as RS-232 or RS-422 Automatic Baud Detect (300Hz – 38.4 kHz)
Position Capture Latency:	100 nanoseconds (using encoder marker) < 1 microsecond (using digital input)
Position Compare Latency:	100 nanoseconds
Positioning Accuracy:	< 1 count (0.5 count, typical)
Data Sampling Rate:	Hardware, 2KHz – 5KHz
Construction	8 layer construction, Through-Hole Design (ACR8000) 6 layer construction, SMT Design (ACR1500) 8 layer construction, SMT Design (ACR1200/ACR2000/ACR8010)
MTBF:	MIL-HDBK-217F G_B @ 25C 50,000 Hours

ACR1200 ORDERING MATRIX

EXAMPLE: 2-Axis Standalone Servo Controller

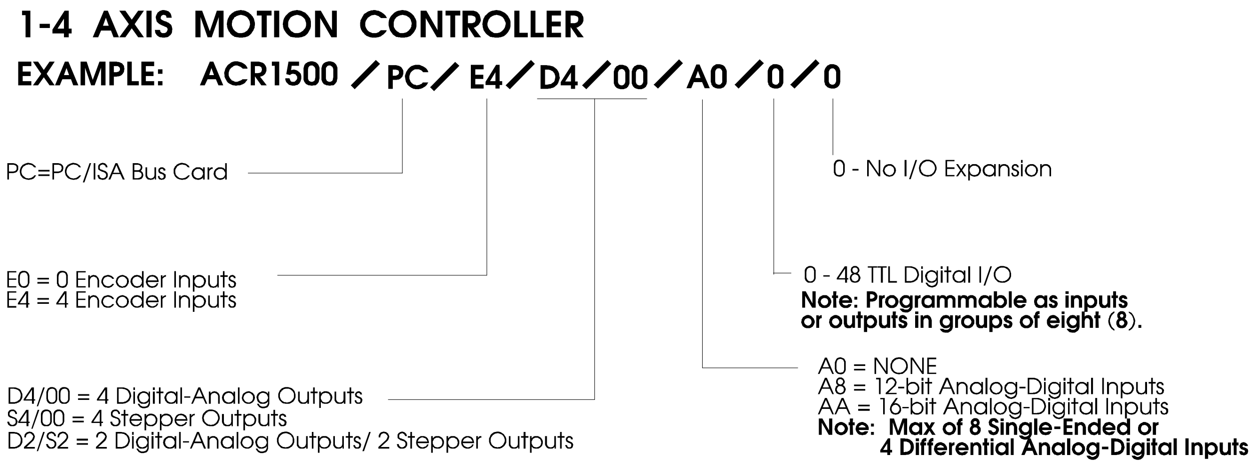


Standard ACR1200 Includes:

1. 32/64 bit Floating Point DSP CPU @ 40MHz
2. Pre-Emptive Multi-Tasker
3. Quadrature Encoder channels (0.1 Hz to 20MHz)
4. 16 bit DAC resolution (+/- 10 VDC output standard, programmable)
5. 4 MHz maximum stepper velocity
6. 25 nsec - 16 usec stepper pulse width range
7. Zero Wait State 128K x 8 bytes System Memory
8. Battery Backed-up 128K x 8 bytes User Memory
9. Standalone configuration
10. 32 Optically isolated 24VDC I/O (16 inputs, 16 outputs)
11. 1024K x 8 Flash Memory (Stores critical parameters and User Programs)
12. Two 128K x 16 EPROM’s
13. Free Support Tools including AcroVIEW, C++, Visual Basic, Visual C, and DOS, Windows, or Windows NT Drivers from Acroloop website.
14. User’s Guide, Hardware Manual, and Training Manuals

ACR1500 ORDERING MATRIX

EXAMPLE: 4-Axis PC-Bus Servo Controller

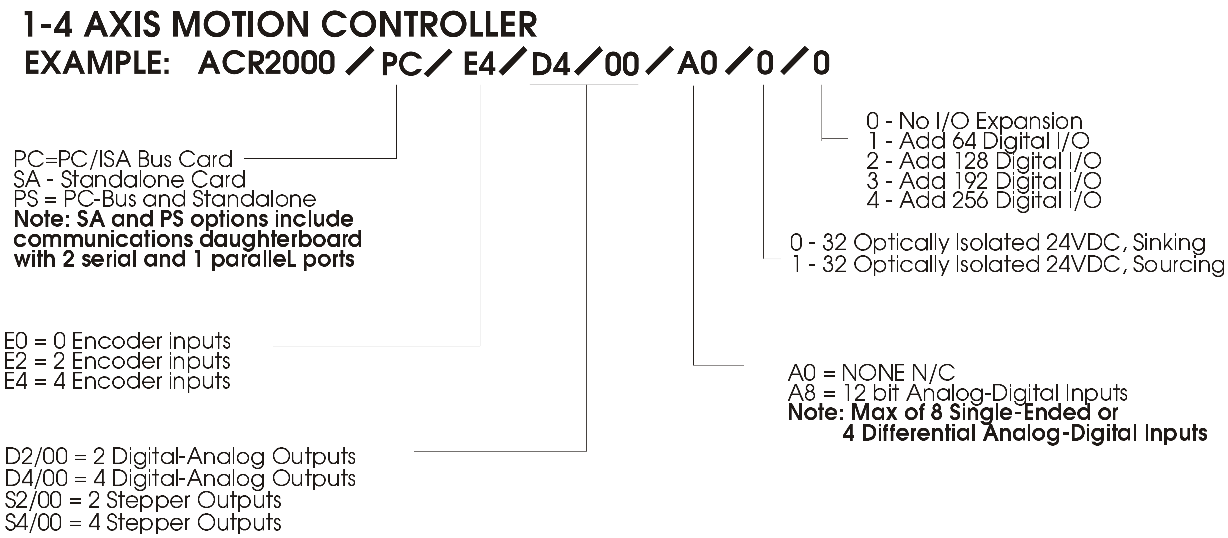


Standard ACR1500 Includes:

1. 32/64 bit Floating Point DSP CPU @ 40MHz
2. Pre-Emptive Multi-Tasker
3. Quadrature Encoder channels (0.1 Hz to 20MHz)
4. 16-bit DAC resolution (+/- 10 VDC output standard, programmable)
5. 4 MHz maximum stepper velocity
6. 125 nsec - 16 usec stepper pulse width range
7. Zero Wait State 256K x 8 bytes RAM (128K x 8 System Memory / 128K x 8 User Memory)
8. PC-Bus Configuration
9. 48 TTL Digital I/O (programmable as inputs or outputs in groups of eight (8))
10. 512K x 8 Flash Memory (Stores critical parameters and User Programs)
11. One 256K x 16 EPROM
12. Two High Speed Communications FIFO's (512 x 8)
13. Free Support Tools including AcroVIEW, C++, Visual Basic, Visual C, and DOS, Windows, or Windows NT Drivers from Acroloop website.
14. Users Guide, Hardware Manual and Training Manual.

ACR2000 ORDERING MATRIX

EXAMPLE: 4-Axis PC-Bus Servo Controller



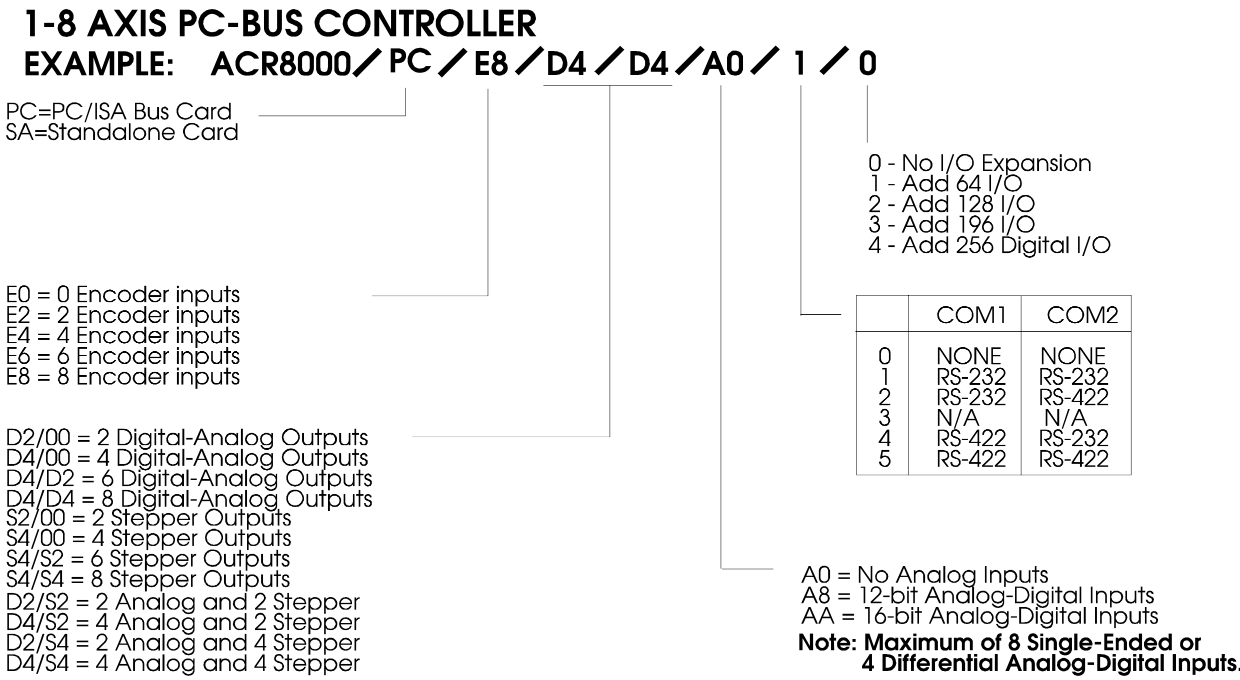
Standard ACR2000 Includes:

1. 32/64 bit Floating Point DSP CPU @ 50MHz
2. Pre-Emptive Multi-Tasker
3. Quadrature Encoder channels (0.1 Hz to 8MHz)
4. 16-bit DAC resolution (+/- 10 VDC output standard, programmable)
5. 4 MHz maximum stepper velocity
6. 125 nsec - 16 usec stepper pulse width range
7. Zero Wait State 512K x 8 bytes System Memory
8. 512K x 8 bytes User Memory (PS and SA Options – User Memory is Battery Backed-Up)
9. PC-Bus or Standalone configuration
10. 32 Optically isolated 24VDC I/O (16 inputs, 16 outputs)
11. 512K x 8 Flash Memory (Stores critical parameters and User Programs)
12. Two 128K x 16 EPROM's
13. Two High Speed Communications FIFO's(512 x 8)
14. Free Support Tools including AcroVIEW, C++, Visual Basic, Visual C, and DOS, Windows, or Windows NT Drivers from Acroloop website.
15. User's Guide, Hardware Manual, and Training Manuals

Note: Standalone ACR2000 boards (PS and SA Options) have 2 serial ports and 1 LPT port. Both serial ports are programmable (RS-232 or RS-422).

ACR8000 ORDERING MATRIX

EXAMPLE: 8-Axis PC-Bus Servo Controller with Serial Ports



Standard ACR8000 Includes:

1. 32/64 bit Floating Point DSP CPU @ 27MHz
2. Pre-Emptive Multi-Tasker
3. Quadrature Encoder channels (0.1 Hz to 8MHz)
4. 16-bit DAC resolution(+/- 10 VDC output standard, programmable)
5. 4 MHz maximum stepper velocity
6. 125 nsec - 16 usec stepper pulse width range
7. 128K x 8 Zero Wait State RAM (64K x 8 System Memory / 64K x 8 bytes User Memory)
8. PC-Bus or Standalone configuration
9. 64 Optically Isolated 24VDC I/O (32 inputs, 32 outputs)
10. 32K x 8 EEPROM (Stores critical parameters)
11. Two 128K x 16 EPROM's
12. Two High Speed Communications FIFO's (512 x 8)
13. Free Support Tools including AcroVIEW, C++, Visual Basic, Visual C, and DOS, Windows, or Windows NT Drivers from Acroloop website.
14. User's Guide, Hardware, and Training Manuals

PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

ACR8010 ORDERING MATRIX

EXAMPLE: 8-Axis PC-Bus Servo Controller with Serial Ports

1-8 AXIS PC-BUS CONTROLLER

EXAMPLE: ACR8010/ PS / E8 / D4 / D4 / A0 / 0 / 0

PC=PC/ISA Bus Card
SA=Standalone Card
PS=PC & Standalone
Note: SA and PS options include 2 serial and 1 parallel port. Serial ports are programmable for RS-232 or RS-422 communications.

E0 = 0 Encoder Inputs
E4 = 4 Encoder Inputs
E5 = 5 Encoder Inputs
E8 = 8 Encoder Inputs
E10 = 10 Encoder Inputs

D2/00 = 2 Digital-Analog Outputs
D4/00 = 4 Digital-Analog Outputs
D4/D2 = 6 Digital-Analog Outputs
D4/D4 = 8 Digital-Analog Outputs
S2/00 = 2 Stepper Outputs
S4/00 = 4 Stepper Outputs
S4/S2 = 6 Stepper Outputs
S4/S4 = 8 Stepper Outputs
D2/S2 = 2 Analog and 2 Stepper
D4/S2 = 4 Analog and 2 Stepper
D2/S4 = 2 Analog and 4 Stepper
D4/S4 = 4 Analog and 4 Stepper

A0 = No Analog Inputs
A8 = 12-bit Analog-Digital Inputs
AA = 16-bit Analog-Digital Inputs
Note: Maximum of 8 Single-Ended or 4 Differential Analog-Digital Inputs.

0 - 64 optically isolated, 24VDC, SINKING
1 - 64 optically isolated, 24VDC, SOURCING

0 - No I/O Expansion
1 - Add 64 I/O
2 - Add 128 I/O
3 - Add 196 I/O
4 - Add 256 Digital I/O

Standard ACR8010 Includes:

1. 32/64 bit Floating Point DSP CPU @ 60MHz
2. Pre-Emptive Multi-Tasker
3. Quadrature Encoder channels (0.1 Hz to 20MHz)
4. 16-bit DAC resolution(+/- 10 VDC output standard, programmable)
5. 4 MHz maximum stepper velocity
6. 125 nsec - 16 usec stepper pulse width range
7. 512K x 8 bytes Zero Wait State System Memory
8. 512K x 8 bytes User Memory (PS and SA Options – User Memory is Battery Backed-Up)
9. PC-Bus or Standalone configuration
10. 64 Optically isolated 24VDC I/O (32 inputs, 32 outputs)
11. 1024K x 8 Flash Memory (Stores critical parameters and User Programs)
12. Two 128K x 16 EPROM's
13. Two High Speed Communications FIFO's (512 x 8)
14. Free Support Tools including AcroVIEW, C++, Visual Basic, Visual C, and DOS, Windows, or Windows NT Drivers from Acroloop website.
15. User's Guide, Hardware, and Training Manuals

OPTIONAL ACCESSORIES

Controller Chassis

The Acroloop controllers can be mounted into an industrial chassis. A number of industrial enclosure options are available. The chassis' provide an easy way to integrate the motion controller into any industrial application. The chassis' allows the following connections:

- 1 to 8 axes of encoder feedback
- Up to 64 digital I/O
- 1 to 8-axis of +/-10VDC analog output
- External I/O Power Connection
- 115VAC Power Connection

The industrial chassis' are complete with an integral power supply and line filter. The chassis' are designed to be mounted into an industrial enclosure typically next to the servo amplifier chassis.

The Acroloop chassis' can be configured with:

- CPU and VGA Cards
- 3.5" 1.44M Floppy Drive
- Hard Disk Drive
- 2 Serial and 1 Parallel Ports

The following figures show a sample of the chassis available for the different Acroloop Motion Control boards.

Controller Chassis (continued)

ACH3120 Chassis for the ACR1200 Standalone Controller Board

ACH3200 Chassis for the ACR2000 Standalone Controller Board

Controller Chassis (continued)

ACH4800 Chassis for the ACR1500/ACR2000/ACR8000/ACR8010 PC or Standalone Controller Boards

Standalone Brackets

As an option, the Acroloop controllers can also be mounted with a standalone bracket. The bracket is ideal for applications requiring a simple rigid bracket to secure the controllers into an industrial enclosure. The standalone mounting bracket should be used in conjunction with the Acroloop Interconnecting Cables.

Interconnecting Cables

Acroloop Interconnecting Cables are available for the Acroloop Motion Control boards. These include Analog I/O cables with flying leads for all of the various P2 board connector types; miscellaneous cables with flying leads for the P1, P3, and P5 headers on the ACR1200 boards; and ribbon cables to mate to the P1, P3, P4 and P5 headers on the ACR1500, ACR2000, ACR8000, and ACR8010 boards.

PH: 612-448-9800 • FAX: 612-448-9321
E-Mail: sales@acroloop.com • URL: www.acroloop.com

Breakout Box

The Acroloop Controllers may also be interconnected using simple interconnect board. The interconnect board/enclosure allow the I/O and encoder feedback signals and digital I/O to be connected to screw terminations. The following table lists the connection cables available for the Acroloop motion control family.

Breakout Box Terminations
- 1-8-axis of encoder feedback -Screw terminations
- Communications Ports – DB-style connectors (COM1, COM2, LPT)

Please note that the 2' ribbon cables are supplied with the interconnect board/enclosure. The interconnect board is also supplied with diagnostic LED's on all of the encoder feedback and digital I/O lines.

Analog-Digital (A/D) Boards

A 12-bit or 16-bit A/D board is available for the ACR1200/ACR1500/ACR8000/ACR8010 boards. Only the 12-bit A/D option is available on-board the ACR2000. The A/D board selects a wide variety of analog inputs as feedback to the controller. Using the analog inputs, the Acroloop Controllers can monitor external devices or use the analog signals to close the loop for the servo control loop.

12-bit A/D Option	**(Use with ACR1200/ACR1500/ACR2000/ACR8000/ACR8010)**
Resolution:	12 bits
Inputs:	Up to 4 Differential or 8 Single-Ended Inputs
Conversion Time:	1.2 microseconds per input

16-bit A/D Option	**(Use with ACR1200/ACR1500/ACR8000/ACR8010)**
Resolution:	16 bits
Inputs:	Up to 4 Differential or 8 Single-Ended Inputs
Conversion Time:	14 microseconds per input

The A/D board option can close the servo loop in place of the encoder feedback.

ACR1200/ACR2000/ACR8000/ACR8010 Expanded I/O Board

The optically isolated digital I/O for the ACR1200/ACR2000/ACR8000/ACR8010 may be expanded. As standard, the ACR8000 and ACR8010 boards are provided with 64 optically isolated I/O (32 inputs and 32 outputs). The ACR1200 and ACR2000 boards are provided with 32 optically isolated I/O (16 inputs and 16 outputs).

The digital I/O can be expanded. Each Expanded I/O board adds 64 optically isolated I/O (32 inputs and 32 outputs) to the ACR1200, ACR2000, ACR8000, or the ACR8010. The maximum number of digital I/O for the ACR1200 and ACR2000 is 288 (144 inputs and 144 outputs). The maximum number of digital I/O for the ACR8000 and ACR8010 is 320 (160 inputs and 160 outputs). Up to 4 digital I/O expansion cards are supported on the ACR1200, ACR2000, ACR8000, and the ACR8010. Specifications for the Expanded I/O board are as follows:

Item	Specification
Size:	4.1” x 6.5” (ACR8000 I/O Expansion Board) 3.85” x 7.0” (ACR1200/ACR2000/ACR8010 I/O Expansion Board)
External Power:	+24VDC @ 2A required per board
Digital Outputs:	24VDC, optically-isolated - 50mA full load (maximum) - 125mA – up to 13 outputs - Sink Only (ACR8000 I/O Expansion Board) - Sink or Source (ACR1200/ACR2000/ACR8010 I/O Expansion Board)
Digital Inputs:	24VDC, optically-isolated - Diode protected - Activates on 10mA per input - Sink Only (ACR8000 I/O Expansion Board) - Sink or Source (ACR1200/ACR2000/ACR8010 I/O Expansion Board)

APPENDIX
PARAMETER/FLAG OVERVIEW

The Acroloop controller family is designed to be able to access all of the pertinent motion control parameters. The parameters (or variables) can then be used or displayed in motion control and PLC programs. This appendix is provided as a simple outline of only a fraction of the parameters that are provided on the Acroloop motion controller boards Also listed is the type of value the variable can have. For example, a variable could be a **“FP32” which stands for a 32 bit floating point number.** The variable could also be a **“LONG” variable, which stands for a 32 bit long integer.**

In general, the Acroloop controller family allows access to every variable. The variables allow infinite flexibility to motion control application design.

Parameter Overview

Description:

The User's Guide appendix provides a list of all system parameters.

P4096-P4175	Flag Parameters
P6144-P6527	Object Parameters
P6656-P6775	PLC Parameters
P6912-P7046	Misc Parameters
P7168-P7408	Program Parameters
P8192-P8216	Master 0 Parameters
P8448-P8472	Master 1 Parameters
P8704-P8728	Master 2 Parameters
P8960-P8984	Master 3 Parameters
P9216-P9240	Master 4 Parameters
P9472-P9496	Master 5 Parameters
P9728-P9752	Master 6 Parameters
P9984-P10008	Master 7 Parameters
P12288-P12355	Axis 0 Parameters
P12544-P12611	Axis 1 Parameters
P12800-P12867	Axis 2 Parameters
P13056-P13123	Axis 3 Parameters
P13312-P13379	Axis 4 Parameters
P13568-P13635	Axis 5 Parameters
P13824-P13891	Axis 6 Parameters
P14080-P14147	Axis 7 Parameters
P16384-P16423	CMT 0 Parameters
P16640-P16679	CMT 1 Parameters
P16896-P16935	CMT 2 Parameters
P17512-P17191	CMT 3 Parameters
P17408-P17447	CMT 4 Parameters
P17664-P17703	CMT 5 Parameters
P17920-P17959	CMT 6 Parameters
P18176-P18215	CMT 7 Parameters
P20480-P20487	Logging Parameters

The following page is an example of the parameters for the Acroloop controllers.

P12288-P14147

Axis Parameters

Position Parameters		Axis Number							
		0	1	2	3	4	5	6	7
Current Position	LONG	12288	12544	12800	13056	13312	13568	13824	14080
Target Position	LONG	12289	12545	12801	13057	13313	13569	13825	14081
Actual Position	LONG	12290	12546	12802	13058	13314	13570	13826	14082
Following Error	LONG	12291	12547	12803	13059	13315	13571	13827	14083
Hardware Capture	LONG	12292	12548	12804	13060	13316	13572	13828	14084
Software Capture	LONG	12293	12549	12805	13061	13317	13573	13829	14085
Primary Setpoint	LONG	12294	12550	12806	13062	13318	13574	13830	14086
Secondary Setpoint	LONG	12295	12551	12807	13063	13319	13575	13831	14087

Offset Parameters		Axis Number							
		0	1	2	3	4	5	6	7
Gear Offset	LONG	12296	12552	12808	13064	13320	13576	13832	14088
Jog Offset	LONG	12297	12553	12809	13065	13321	13577	13833	14089
Cam Offset	LONG	12298	12554	12810	13066	13322	13578	13834	14090
Ballscrew Offset	LONG	12299	12555	12811	13067	13323	13579	13835	14091
Backlash Offset	LONG	12300	12556	12812	13068	13324	13580	13836	14092
Reserved	LONG	12301	12557	12813	13069	13325	13581	13837	14093
Reserved	LONG	12302	12558	12814	13070	13326	13582	13838	14094
Reserved	LONG	12303	12559	12815	13071	13327	13583	13839	14095

Servo Parameters		Axis Number							
		0	1	2	3	4	5	6	7
Proportional Gain	FP32	12304	12560	12816	13072	13328	13584	13840	14096
Integral Gain	FP32	12305	12561	12817	13073	13329	13585	13841	14097
Integral Limit	FP32	12306	12562	12818	13074	13330	13586	13842	14098
Integral Delay	FP32	12307	12563	12819	13075	13331	13587	13843	14099
Derivative Gain	FP32	12308	12564	12820	13076	13332	13588	13844	14100
Derivative Width	FP32	12309	12565	12821	13077	13333	13589	13845	14101
Feedforward Velocity	FP32	12310	12566	12822	13078	13334	13590	13846	14102
Feedforward Accel	FP32	12311	12567	12823	13079	13335	13591	13847	14103

Monitor Parameters		Axis Number							
		0	1	2	3	4	5	6	7
Proportional Term	FP32	12312	12568	12824	13080	13336	13592	13848	14104
Integral Term	FP32	12313	12569	12825	13081	13337	13593	13849	14105
Derivative Term	FP32	12314	12570	12826	13082	13338	13594	13850	14106
Velocity Term	FP32	12315	12571	12827	13083	13339	13595	13851	14107
Acceleration Term	FP32	12316	12572	12828	13084	13340	13596	13852	14108
Summation Point	FP32	12317	12573	12829	13085	13341	13597	13853	14109
Filter Output Signal	FP32	12318	12574	12830	13086	13342	13598	13854	14110
Output Signal	FP32	12319	12575	12831	13087	13343	13599	13855	14111

Flag Overview

Description:

The following table provides a list of all system flags. The following page is an example of the flags for Master Flags BIT512 to BIT767:

Description	***First Flag***	***Last Flag***
Optoisolated Inputs	BIT0	BIT31
Optoisolated Outputs	BIT32	BIT63
Miscellaneous Inputs	BIT64	BIT95
Miscellaneous Outputs	BIT96	BIT127
User Flags Group 1-4	BIT128	BIT256
Expansion I/O Flags	BIT256	BIT511
Master Flags	BIT512	BIT767
Axis Flags	BIT768	BIT1023
Program Flags	BIT1024	BIT1535
PLC Flags	BIT1536	BIT1791
FIFO Stream Flags	BIT1792	BIT1823
LPT1 Stream Flags	BIT1824	BIT1855
COM1 Stream Flags	BIT1856	BIT1887
COM2 Stream Flags	BIT1888	BIT1919
User Flags Group 5-8	BIT1920	BIT2047
Secondary Master Flags	BIT2048	BIT2303
Secondary Slave Flags	BIT2304	BIT2559
Encoder Flags	BIT2560	BIT3071
Stepper Flags	BIT3072	BIT3583
Commutator Flags	BIT3584	BIT4095

BIT512-BIT767

Master Flags

See also: Secondary Master Flags

Flag Parameter	4112	4113	4114	4115	4116	4117	4118	4119

Status Flags	Bit	MASTER Number							
	Index	0	1	2	3	4	5	6	7
Accelerating	0	512	544	576	608	640	672	704	736
Decelerating	1	513	545	577	609	641	673	705	737
Stopping	2	514	546	578	610	642	674	706	738
Jerking	3	515	547	579	611	643	675	707	739
In Motion	4	516	548	580	612	644	676	708	740
Move Buffered	5	517	549	581	613	645	677	709	741
Feedholding	6	518	550	582	614	646	678	710	742
In Feedhold	7	519	551	583	615	647	679	711	743

Control Flags	Bit	MASTER Number							
	Index	0	1	2	3	4	5	6	7
Feedhold Request	8	520	552	584	616	648	680	712	744
Cycle Start Request	9	521	553	585	617	649	681	713	745
Kill All Moves Request ☠	10	522	554	586	618	650	682	714	746
Stop All Moves Request ☠	11	523	555	587	619	651	683	715	747
FVEL Zero Pending	12	524	556	588	620	652	684	716	748
FVEL Zero Active	13	525	557	589	621	653	685	717	749
FOV/ROV Lock Pending	14	526	558	590	622	654	686	718	750
FOV/ROV Lock Active	15	527	559	591	623	655	687	719	751

Limit Flags	Bit	MASTER Number							
	Index	0	1	2	3	4	5	6	7
Not In Position	16	528	560	592	624	656	688	720	752
Not Excess Error	17	529	561	593	625	657	689	721	753
Within A Limit	18	530	562	594	626	658	690	722	754
Not Within B Limit	19	531	563	595	627	669	691	723	755
Not Torque Limit	20	532	564	596	628	660	692	724	756
Not In Torque Band	21	533	565	597	629	661	693	725	757
Reserved	22	534	566	598	630	662	694	726	758
Reserved	23	535	567	599	631	663	695	727	759

Sequence Flags	Bit	MASTER Number							
	Index	0	1	2	3	4	5	6	7
Decrement Count	24	536	568	600	632	664	696	728	760
Increment Count	25	537	569	601	633	665	697	729	761
Interrupt On Move	26	538	570	602	634	666	698	730	762
TRG Pending	27	539	571	603	635	667	699	731	763
Start Move Inhibit	28	540	572	604	636	668	700	732	764
REN Request Flag	29	541	573	605	637	669	701	733	765
Cycle Start Lockout	30	542	574	606	638	670	702	734	766
Reserved	31	543	575	607	639	671	703	735	767