Parker Automation

p/n YPM018129

Acroloop Training Course for the ACR2000/8000 Motion Controller

Effective: October 7, 2002

COURSE OBJECTIVES

This three day training course will cover the information necessary to allow an individual to be able to setup and program a basic application for the ACR Multi-axes controller card. This course will also cover the specialized and advanced programming commands and features that make the ACR series the most flexible controller available.

Due to the flexibility and versatility of this Controller, there will not be sufficient time to cover all the AcroBasic programming commands available. Also there may be some material covered in this training manual that may not have coverage during class time.

A fundamental knowledge of basic programming techniques will aid the new user is using the AcroBasic programming.

To aid in making this manual more readable, the term ACR Card will be used to refer to the ACR2000/8000 Multi-axes Controller Card

ACR MULTI-AXES CONTROLLER BASIC OPERATION

The ACR Card is a DSP (Digital Signal Processor) based, 32-bit floating point controller, with onboard Multi-tasking executive allowing up to 24 Simultaneous Tasks. These tasks are executed at a default interrupt rate of 500 microseconds. The tasks can contain up to 16 motion programs, up to 8 PLC (Programmable Logic Controller) programs, and use up to 4 communication channels.

- Programs 0-7 are scanned at a fixed time period and are treated with equal priority. These are considered the primary motion programs.
- Program 8-15 share a time period that has the same priority as the first 8 programs and are scanned in sequence. These programs are considered non-motion programs because they have a lower priority due to their scan sequence.

NOTE: Care must be taken to avoid using commands that inhibits program operation such as an INH or DWELL in Programs 8-15. Use of these commands will cause the other programs that share that time period from operating when any program is inhibited.

Example: If there was an application using programs 0,1,2,8,9 & 10, the programs would be scanned in the following sequence.

0 - 1 - 2 - 8 - 0 - 1 - 2 - 9 - 0 - 1 - 2 - 10 - 0 - 1 - 2 - 8 ... etc.

- PLC 0-7 are run compiled and in their entirety each time they are executed. These are placed in the PLC scanner list and one event is run every servo interrupt.

The following diagram show the available tasks that can be run from the controller card.

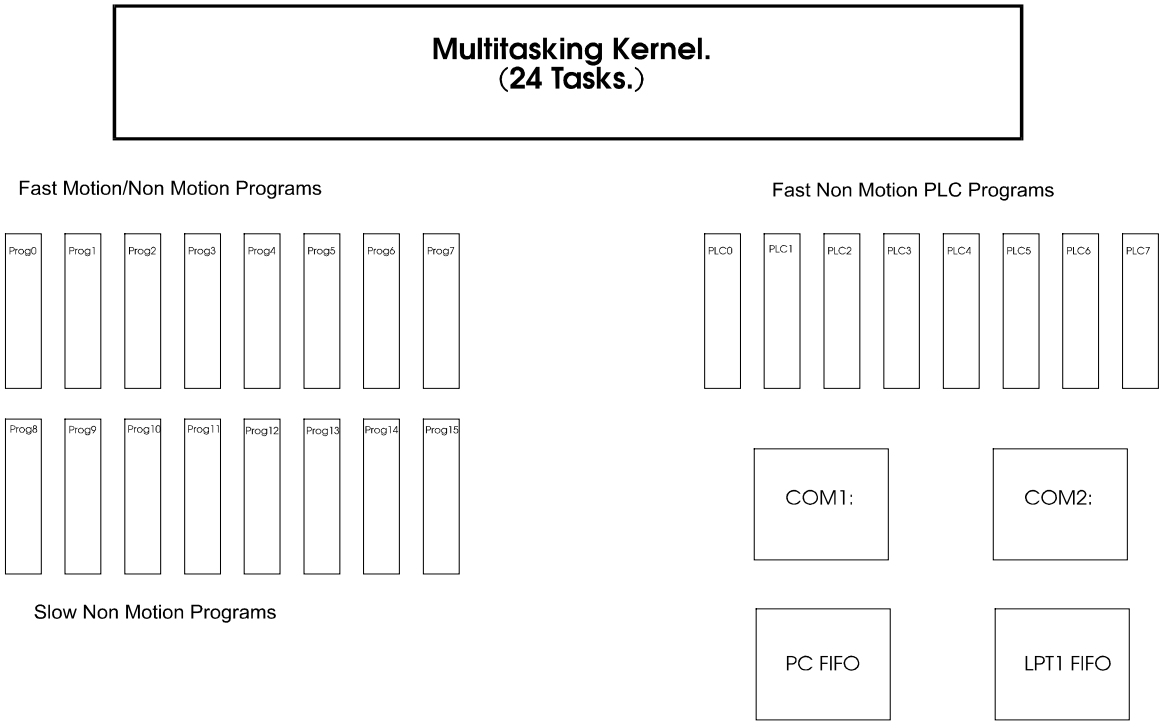


Figure 1

ACR SERIES CONTROLLER TIMING CHART

The following example gives a brief description and shows the timing used by the ACR card multi-tasking environment. This example has three programs and two PLCs active.

⇒ Active programs are scanned in sequence, each program has a fixed 1mSec scan time. If a command is not completed during the program scan in which it is initiated, it will be suspended until the next scan of that program. Moves that are started in a program are buffered and then controlled by the Servo interrupt.

⇒ PLC programs are linked into the PLC scanner, this 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. This event is executed in its entirety and the next event is executed during the next interrupt. This process is repeated after the last item in the list.

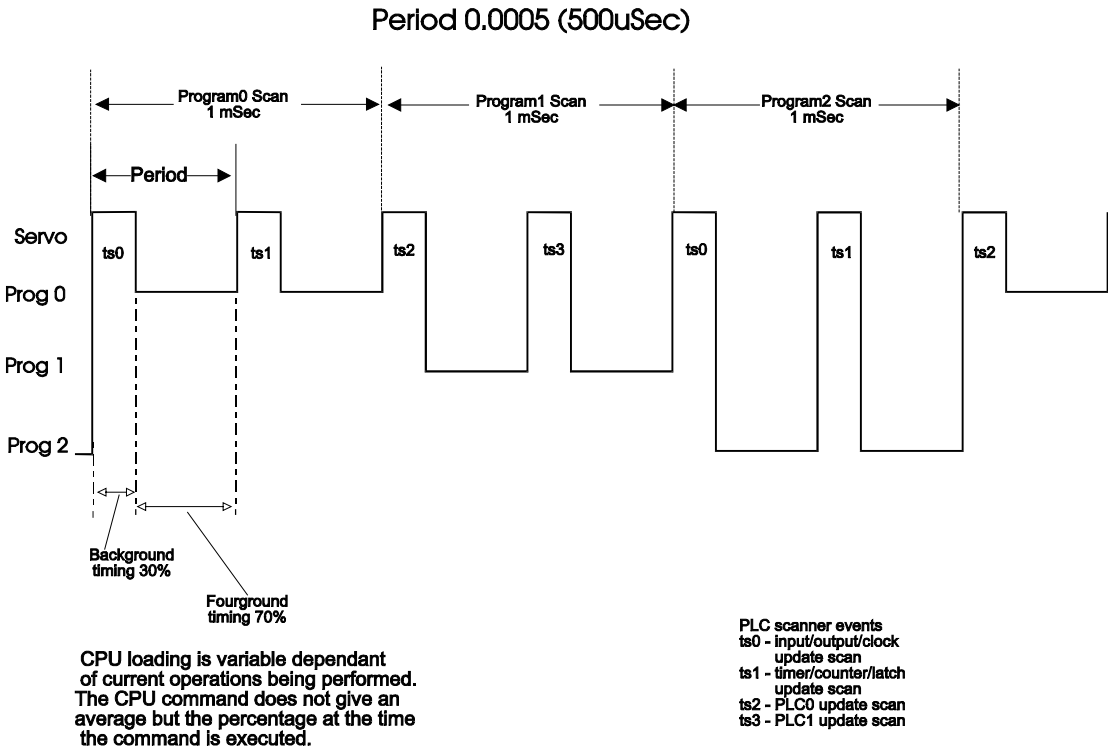


Figure 2

The following commands are used to determine program timing.

- PERIOD - This will display the current value of base system timer period. This value is displayed in seconds. The default value is 0.0005 Seconds (500 uSec)
- CPU - Displays the processor load as a percentage of the foreground and background timing. The background is the time used by servo loop updates, velocity profiles, and axis position interpolation. The foreground timing consists of the remaining time used for program operation.

The amount of time for program command execution in each program scan is determined by taking the 0.001 second (1 mSec) program scan time which is a fixed value and multiplying it by the foreground percentage. The CPU loading is not a fixed value so it may be difficult to get an accurate measurement of this time.

In addition to any PLC programs, the scanner event list also contains two other events.

The input/output/clock update scan.

This is always in the PLC scanner list, regardless as to whether there are any PLC programs in use. This event updates the Optoisolated digital I/O, the global system clock, and the clock tick flags.

The timer/counter/latch update scan.

This event updates the condition of the timers, counters, and latches that are available in the ACR Card.

There are several methods that you can use to reduce the background time and increase the operating speed of your programs.

- The ACR Card will update all active axes servo loops whether they are in use or not. If an axis is not to be used, turn off it's servo loop using the AXIS command such as AXIS7 OFF. You can also set the OPEN SERVO LOOP flag for that axis, such as SET 1009 for axis 7.

NOTE: More detailed examples of these commands will be discussed later in this manual.

- Removing print statements from programs that will not be connected to an open device in the final application.

- Removing REM (remark) statements from programs.

SERVO LOOP

The following is a block diagram of the servo loop used by the ACR Card. There is a similar loop produced for each of the 8 axes. An understanding of the basics of this closed loop operation will aid in learning the AcroBasic commands, parameters and bit flags used by the ACR Card.

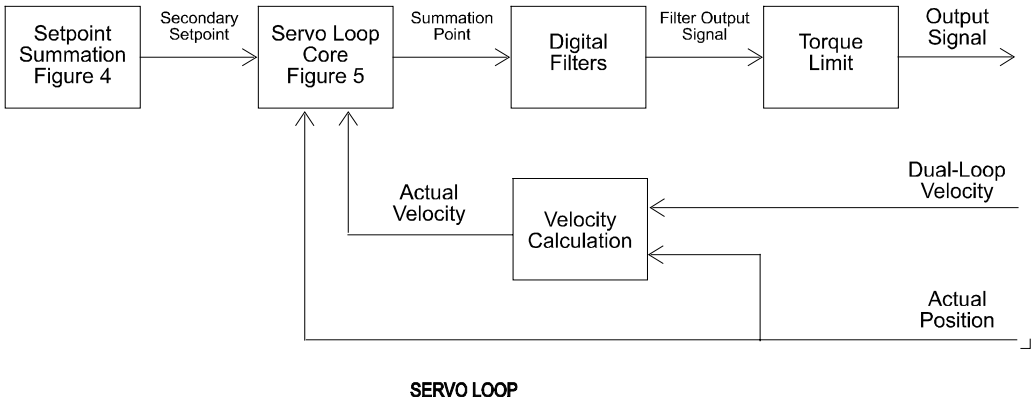
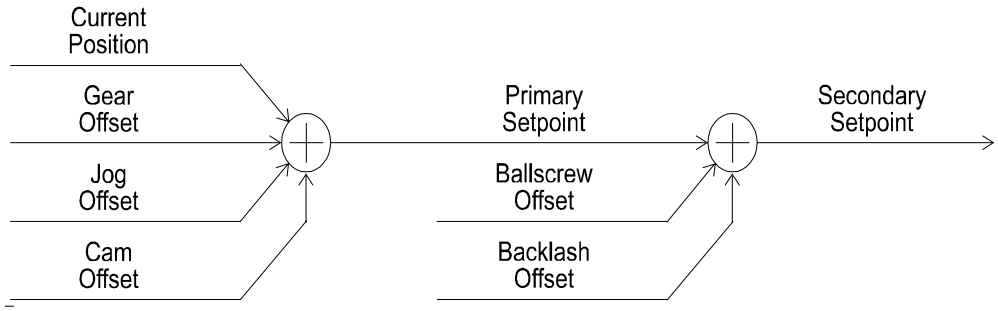


Figure 3

Setpoint Determination

There are four different methods used for generating axis motion, these are; MASTER PROFILE COMMANDED POSITION **(Current position)**, ELECTRONIC GEARING **(Gear Offset)**, ELECTRONIC CAM **(Cam Offset)**, and SINGLE AXIS MOTION PROFILE **(Jog offset)**. Each of these is determined independently by the ACR Card and combined to produce the Primary Setpoint.

The Primary Setpoint is then combined with two other factors BALLSCREW COMPENSATION **(Ballscrew Offset)** and BACKLASH COMPENSATION **(Backlash Offset)** to determine the Secondary Setpoint.



SETPOINT SUMMATION
Figure 4

- Primary Setpoint

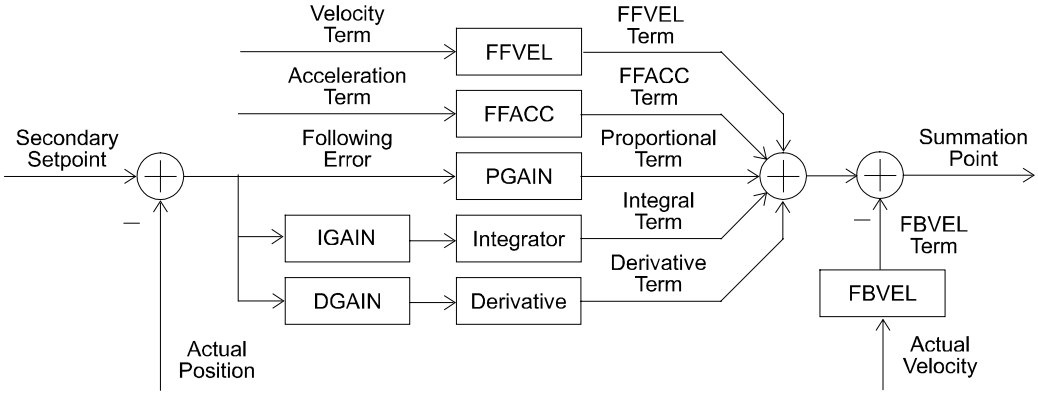
Motion calculations are performed for each axis, these produce offset values in encoder pulses. Each of the four methods produce an independent offset value that is not affected by the value of the other three.
As an example; an axis can be used in a motion profile to perform coordinated movement with another axis, and operating as an electronically geared axis following a selected source encoder simultaneously. Both of these operations will be generating separate offset values that are then used to produce the Primary Setpoint. These values are stored in independent registers and can be manipulated independently

- Secondary Setpoint

The output of the Primary Setpoint is then combined with the offsets generated by the two static compensations and this produces the Secondary Setpoint.

Servo Loop Core

The actual position register contains the value of the feedback object that is currently attached to the axis. This register is compared to the axis secondary setpoint register and determines the current value of following error. The value of following error is multiplied by the gain factors for that axis and produces the Proportional Term, Integral Term, and Derivative Term. Velocity and Acceleration terms are calculated for the axis and are multiplied by the FeedForward velocity and acceleration respectfully to produce the FFVEL Term and FFACC Term. These values are added together and compared to the FEEDBACK VELOCITY Term produced by the velocity encoder if the axis is being used in a dual-encoder configuration. This produces the axis Summation Point which is then used in the servo loop shown in figure 3.



Servo loop core

Figure 5

SYSTEM CONFIGURATION

PROMPT FROM THE ACR CARD.

When the ACR Card is powered up, the following prompt is given after communications has been established on any of the ports.
SYS> This represents the SYSTEM level.

⇒ To change to any program prompt; type the desired program using the command PROG***X*** where ***X*** is the program number.
Upon selecting PROG0.....PROG15 the prompt changes to
PO0> through PO15>.

⇒ To change to any PLC program prompt; type the desired PLC using the command PLC***X*** where ***X*** is the PLC number.
Upon selecting PLC0..... PLC7, the prompt changes to
PLC0> through PLC7>

HARDWARE CONFIGURATION

The base hardware installed on the system must be defined with the CONFIG command. This command contains four arguments, the argument definitions and valid values are shown in the following. To view the current configuration enter CONFIG with no arguments from the SYS prompt or any program prompt.

The default configuration is as follows - CONFIG ENC8 DAC4 DAC4 ADC8

* The first argument is the number of encoder channels installed on the motherboard.
  NONE, ENC2, ENC4, ENC6, ENC8
* The second argument is the type of module installed in the first SIMM socket.
  NONE, DAC2, DAC4, STEPPER2, STEPPER4
* The third argument is the type of module installed in the second SIMM socket.
  NONE, DAC2, DAC4, STEPPER2, STEPPER4
* The fourth argument is if an ADC module is installed in the third SIMM socket.
  NONE, ADC8

SOFTWARE ATTACHMENTS

The attach axis command defines the attachment of position feedback and signal output for a given axis. This command contains four arguments, the arguments and valid values are shown in the following. To view the current attachments enter ATTACH AXIS with no arguments from the SYS prompt or any program prompt.

* The first argument is the axis designation.
  AXIS0.....AXIS7
* The second argument is the position attachment for the axis, valid position feedback device attachments are
  Quadrature encoder feedback applied to the ACR Card encoder inputs. ENC0.....ENC7
  Analog position feedback applied to the ACR Card analog inputs. ADC0.....ADC7
  Open loop stepper feedback.
  STEPPER0.....STEPPER7
* The third argument is the signal attachment for the axis, valid signal output attachments are
  Analog voltage output
  DAC0.....DAC7
  Step and directions outputs.
  STEPPER0.....STEPPER7
* The fourth argument is the velocity attachment for the axis, this is used as a software tachometer based on encoder or analog signal input. Valid velocity feedback device attachments are
  Quadrature encoder feedback applied to the ACR Card encoder inputs. ENC0.....ENC7
  Analog position feedback applied to the ACR Card analog inputs. ADC0.....ADC7

The default configuration for each axis attaches that axis to it's matching encoder and DAC output.

```
ATTACH AXIS0 ENC0 DAC0 ENC0
ATTACH AXIS1 ENC1 DAC1 ENC1
ATTACH AXIS2 ENC2 DAC2 ENC2
ATTACH AXIS3 ENC3 DAC3 ENC3
ATTACH AXIS4 ENC4 DAC4 ENC4
ATTACH AXIS5 ENC5 DAC5 ENC5
ATTACH AXIS6 ENC6 DAC6 ENC6
ATTACH AXIS7 ENC7 DAC7 ENC7
```

MEMORY ALLOCATIONS

The amount of the total system memory available to the user for storing of programs, variables, arrays, and memory allocation is 512K (65525 X 8bit bytes). These bytes can be freely allocated towards PROG0.....PROG15, PLC0..... PLC7, as well as variables and arrays , the DIM command is used for this. Using the DIM command with no arguments will display the current memory allocation. Memory is allocated on two levels, system level and program level.

Free Memory can be displayed using the MEM command. When used from the SYS prompt, the MEM command displayed unallocated system memory. When used from any program or PLC prompt, the MEM command shows unused memory for that program.

The CLEAR command is used to free previously allocated memory.

NOTE: The system memory is stored in battery-backed SRAM. If this memory is reset using the BRESET command, memory allocation will return to default values. Hardware configuration information is also stored in the system memory.
CAUTION - Once the BRESET command has been issued, there is no way to return the battery to normal operation without removing and then restoring power.

System Level Formats

Memory allocation for the system level uses the DIM, CLEAR, and MEM commands previously explained. To use the CLEAR command, allocated programs space must be free from program statement lines. If an attempt is made to use the CLEAR command with programs still present, a **Programs not empty** error will be displayed. The NEW command is used to delete a program, such as, NEW PROG0. The NEW ALL command is used to delete all program at one time.

⇒ Allocate memory for program - DIM PROG***x***(***size***)
The system default is to divide the total memory into eight equal sections

⇒ Allocate memory for PLC - DIM PLC***x***(***size***)
The system default is no memory allocation.

⇒ Allocate memory for global variables - P(***count***)
The global variables range is P0.....P4095, these are 64 bit floating point values.

The system default is no memory allocation.

NOTE: Once memory has been allocated for a program, PLC, or for global variables; the allocation can NOT be changed with out CLEARing the memory. If an attempt to allocate memory to a previously defined memory block is attempted a **Redimensioned block** error message will be displayed.

Program Level Formats

Memory allocation for program level is used for variables and arrays. Program level also uses the DIM, CLEAR, and MEM commands. The use of the DIM command to allocate memory can be done from the program level or within a program using a program line.

The local variables consists of the following types:

- LV - Long (32 bit integers) - DIM LV(***count***)
- SV - Singles (32 bit floating point) - DIM SV(***count***)
- DV - Doubles (64 bit floating point) - DIM DV(***count***)
- $V - String Variables (8 bit character) - DIM $V(***count,length***)

The Following arrays are also handled. Array allocation is done in two parts. First the array reference is allocated, this specifies the type and number of arrays required. The second part is the number of elements in each array.

* LA - Long Arrays - DIM LA(***count***) - DIM LA***X***(***count***)
* SA - Singles Arrays - DIM SA(***count***) - DIM SA***X***(***count***)
* DA - Doubles Arrays - DIM DA(***count***) - DIM DA***X***(***count***)
* $A - String Arrays - DIM $A(***count***) - DIM $A***X***(***count,length***)

NOTE: Once memory has been allocated for a variable or array element; the allocation can **NOT** be changed with out CLEARing the memory. If any changes are made to a program by changing or adding a line, a CLEAR is performed on this program. The dimensioning must be re-entered or the dimensioning must be included in a program line. If an attempt to allocate memory to a previously defined memory block is attempted a **Redimensioned block** error message will be displayed.

PROGRAMS AND COMMANDS

To enter commands, change to the desired program prompt. From the program prompt you can enter immediate mode commands or program commands.

⇒ Immediate commands are executed when the enter key is pressed. This can be used to set operating characteristics, view the current setting or perform the command.

To view the current master velocity, type the VEL command with no value.

To change the current master velocity, type VEL and then the new value such as VEL 1000 or VEL1000.

To perform a command, such as turning on a the first of the digital outputs type SET 0.

⇒ To enter a line to be executed when the program is running; enter the line number, a space and the command.

Such as 10 VEL 1000

When writing programs with a text editor, both immediate commands and program lines can be included in the program. The immediate command will be executed when the program is downloaded to the ACR Card, the program lines will be stored in the program space.

Once a program has been entered, there are three ways to run the program.

⇒ Type the RUN command which will start program operation. If you wish to start a program at a location other then the first command use the format, RUN 100 to start the program from line number 100.

⇒ Use the LRUN command which echoes print commands and error messages to the current communication port. (Preferred method when troubleshooting program)

⇒ Enter the PBOOT command as the first line in a program. This will automatically start the program when power is applied to the ACR Card.

To stop a program, use the HALT command.

To enable echo on a running program use the LISTEN command. Pressing the Esc key will cancel the echo of the program.

To pause a currently running program use the PAUSE command, this will feedhold the current move and pause the program at the current command line. Use the RESUME command to continue program operation.

Program operation can be controlled from the SYS prompt or within another program location by specifying the program by number such as RUN PROG0, PAUSE PROG0, or HALT PROG0. To control all programs simultaneously, use the designator ALL in a command such as RESUME ALL, or HALT ALL.

STARTING FROM GROUND ZERO.

The following code initializes the card and clears all memory The following line can be entered from the system or any program prompt.
HALT ALL
NEW ALL
DETACH ALL
From the system prompt enter the following command to go to default memory allocation
CLEAR

⇒ Another way of achieving the same result is to take the following steps.
First enter the following command in the immediate mode
BRESET
Then turn the power off and wait for 20-30 seconds.
Then turn the power On. and do the following commands.
SYS
CLEAR

SAMPLE MEMORY ALLOCATION FOR PROGRAMS

The following commands entered from the system prompt will allocate 5000 bytes to each of PROG0.....PROG7 and PLC0.

```
DIM PROG0(5000)
DIM PROG1(5000)
DIM PROG2(5000)
DIM PROG3(5000)
DIM PROG4(5000)
DIM PROG5(5000)
DIM PROG6(5000)
DIM PROG7(5000)
DIM PLC0(5000)
```

The above commands are provided on disk in a file named DIMEQUAL.8K

(Before downloading this file to the ACR Card make sure all programs have been erased!)

ATTACHING AXES

Before any axis can be commanded to move, it must be attached to a MASTER. A MASTER is another name for a profile generator. There are 8 Masters available. Each Master can have 1 to 8 axes attached to it. Each of these axes are referred to as a SLAVE. Since the ACR Card can have up to 8 Axes on one board; there are, in total 8 possible slaves.
If the ACR Card is being used to control 8 independent axes, MASTER0 is attached to SLAVE0, MASTER1 to SLAVE1.....MASTER7 to SLAVE7.

If however the board is being used to control coordinated axes, MASTER0 can be attached to SLAVE0.....SLAVE7 To prepare a program area to control axis movement the ATTACH command is used.

The ATTACH MASTER command is used first. One of the 8 masters is attached to the program, each master may be used only once. Each master has eight internal slots which are used as attachment points for axes.

The ATTACH SLAVE command is used to attach individual axis to the internal slots of the master for that program.

The format for this command is ATTACH SLAVE *#* AXIS ***axis*** "***name***"

- The first argument is the internal slot in the master to attach the axis.
  SLAVE0.....SLAVE7
- The second argument is what axis of the ACR Card to use.
  AXIS0.....AXIS7
- The third argument is what ASCII alpha string is to be used in the program to designate this axis
  Use any alpha string from 1 to 4 characters Such as X, UP, ARM.

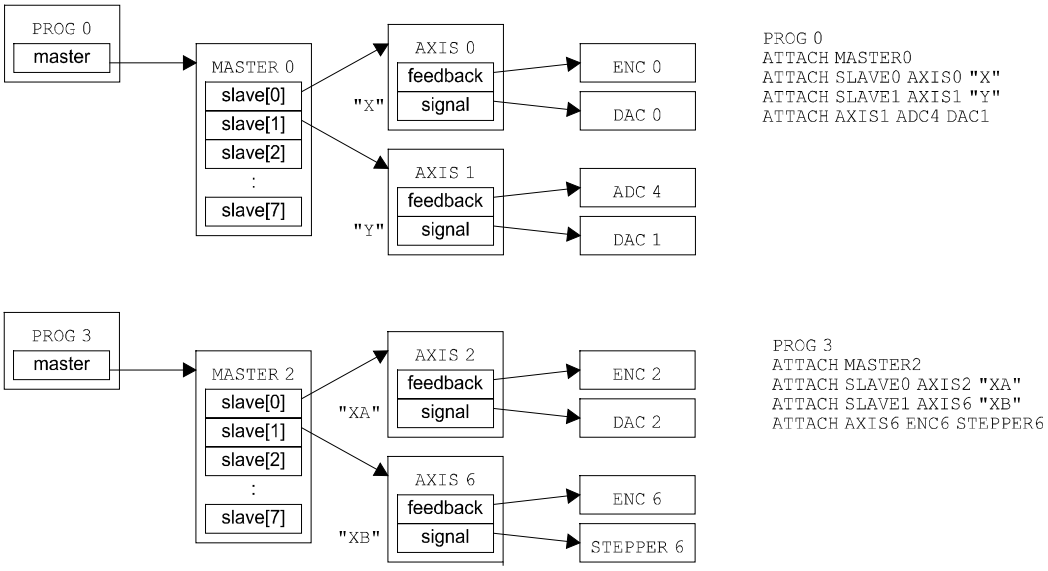


Figure 6

Because of it's use for global parameters and system parameters the character P should NOT be used to designate an axis. Because of it's use with the F(feedrate) command the character F should NOT be used to designate an axis.

NOTE: Each master and axis can only be attached once. An **Already attached** error message will be generated if the master is attached to another program, or the designated axis is attached to another master

In the following example, PROG0 will allow controlling a 2 Axis machine.
PROG0

```
HALT
DETACH
ATTACH MASTER0
ATTACH SLAVE0 AXIS0 “X”
ATTACH SLAVE1 AXIS1 “Y”
```

The above commands are provided on disk in a file named ATTCH2P0.8K

STORING SYSTEM PARAMETERS

System attachments, master and axis parameters, gains and offsets are stored into an EEPROM to be retrieved on power-up or by issuing an ELOAD command. After you have entered the desired information entering an ESAVE command will store the information. If an ESAVE command is not issued, system parameter, master profile, and attachment information will not `be retained and all information will revert to default value.

TAKING THE FIRST STEPS

MOVING AXES

The following program will move two axis to describe a SQUARE pattern. The linear move commands in this program will move the specified axis to the absolute position listed in the command. The MOV command is the default axis data input mode and is typically not specified. So lines 30 is formatted as shown and not as 30 MOV X2000.

```
PROG0
HALT
NEW
DETACH
ATTACH MASTER0
ATTACH SLAVE0 AXIS0 “X”
ATTACH SLAVE1 AXIS1 “Y”
MULT X4 Y4
10 ACC 100000 DEC 100000 STP 100000
20 VEL 4000
30 X2000
40 Y2000
50 X0
60 Y0
70 GOTO 30
```

The above commands are provided on disk in a file named SQUARE.8K

INCREMENTAL MOVES

To perform an incremental move instead of an absolute move, a forward slash (***/***) is placed in front of the distance in an axis move command.

```
X /2000
```

The above command will move axis X an incremental move of 2000 units. If the same move is used repeatedly, axis X will add 2000 units each execution.

IMMEDIATE MODE EXAMPLES

Doing feedrate override

To run the previous program, after either entering the program manually or downloading the file SQUARE.8K to the ACR Card, enter the following command.

```
RUN
```

Now both the motors should be running one after the other.

To see the affect of dynamic feedrate overriding on the fly issued the following commands and note the resulting effect on the motors. Wait 5 seconds after entering each command.
FOV 0.5
FOV .25
FOV 1
The above FOV commands will override the feedrate to 50%, 25% and back to 100% of the Programmed Feed.

Control from another program location

Immediate mode commands can be entered from a program other that the one you want to affect. Change to program 1 using the PROG1 command. The prompt will change to P01> and enter the following commands.
PAUSE PROG0
RESUME PROG0

Direct axis access

Direct access to an axis can be done by using the AXIS command. This does not require the use of the ASCII name assigned to that axis with the attach command
The format for usage of this command is to list the axis first, then the command and associated arguments. As an example, to change the encoder multiplier of the X and Y axis of the previous program you may use either of the following.
From program 0 - MULT X2 Y2
From any program location - AXIS0 MULT2 : AXIS1 MULT2

Other examples of direct axis control are - AXIS3 PGAIN0.001
AXIS0 PPU1

NOTE: Axis motion of an attached axis can **only** be done from the program location were it is attached.
EXAMPLE: X10000 Y12000 would be done from program 0 if their attachment was done in that program.

Direct Master Profile access

Direct access to master profile commands from a program other than the program where that master is attached can be done using the MASTER command.
The format for usage of this command is to list the master first, then the command and associated arguments. As an example, to change the velocity of master 0 of the previous program you may use either of the following
From program 0 - VEL 2000
From any program location - MASTER0 VEL2000

Other examples of direct master control are - MASTER0 STP0
MASTER2 FOV2.0

CONDITIONING THE MOTION PROFILE

SETTING THE VELOCITY PROFILE

Coordinated motion controlled by an attached master can be conditioned by use of the following velocity profile commands. These commands affect **ALL** axes attached to the master.
When a move is executed, the master controlling the move is actually making an imaginary move for a certain number of "units". The length of the master's move is called the "vector distance" and is either calculated automatically based on VECDEF and slave distances, or overridden manually by the VECTOR command. The characteristics of that move are controlled by the velocity profile commands.
Axis motion that is controlled by gear offset, cam offset, or jog offset are controlled by the commands associated with those offsets.
To check the current setting, enter the command with no argument. Setting any of these commands to a value of zero will disable the command.

- ACC - This command sets the acceleration rate used to ramp from lower to higher speeds.
  Default value 20000 units /second^2

- DEC - This command sets the deceleration ramp used to ramp from higher to lower speeds.
  Default value 20000 units/second^2

- STP - This command sets the deceleration ramp to be used at the end of the move. Setting STP to zero will end the move without ramping down, this allows back-to-back moves to be merged together. Setting STP to anything other than zero will cause the move to ramp down at the end of the move to the final velocity set by the FVEL command.
  Default value 20000 units/second^2
- FVEL - This command sets the final velocity value for a move profile. Final velocity is used as a target velocity when the STP ramp is active. This value is used to slow down, but not stop, between moves when the value is other than zero.
  Default value 0

- VEL - This command sets the target velocity for coordinated motion. Changes in velocity take place on the next move.
  Default value 10000 units/second

- F - This command is an alternative to using the VEL command. F translates the move velocity into units/minute
  Default value 10000 units/60 seconds = 166.67 units/minute.

The following chart will demonstrate how the ACC, DEC, and STP commands affect a typical trapezoidal profile

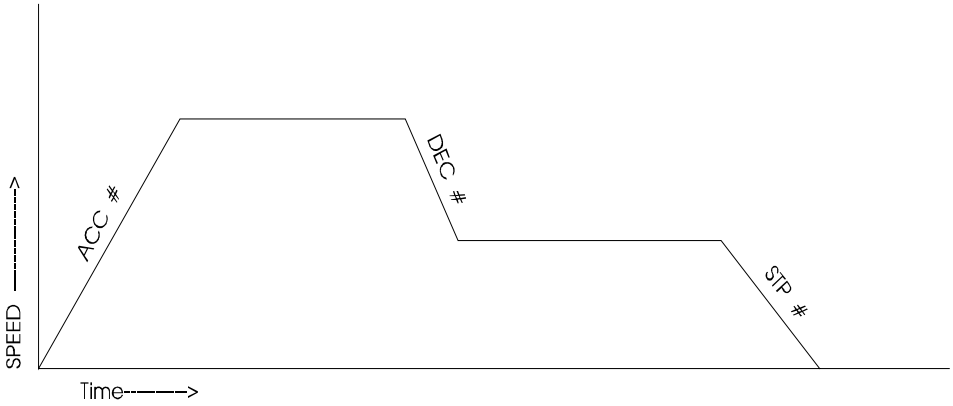


Figure 7

- FOV - This command sets the velocity override for the current master. This value is multiplied by the current VEL setting to determine the velocity to be used by master motion. Feedrate override takes place immediately, if a move is in progress the velocity will use the current ACC and DEC rate to change to the new velocity.
  Default value 1.0

- JRK - (scurve) This command controls the slope of the acceleration versus time profile, this produces a trapezoidal profile. If the jerk value is zero, the acceleration is rectangular.
  Default value 0 units/second^2

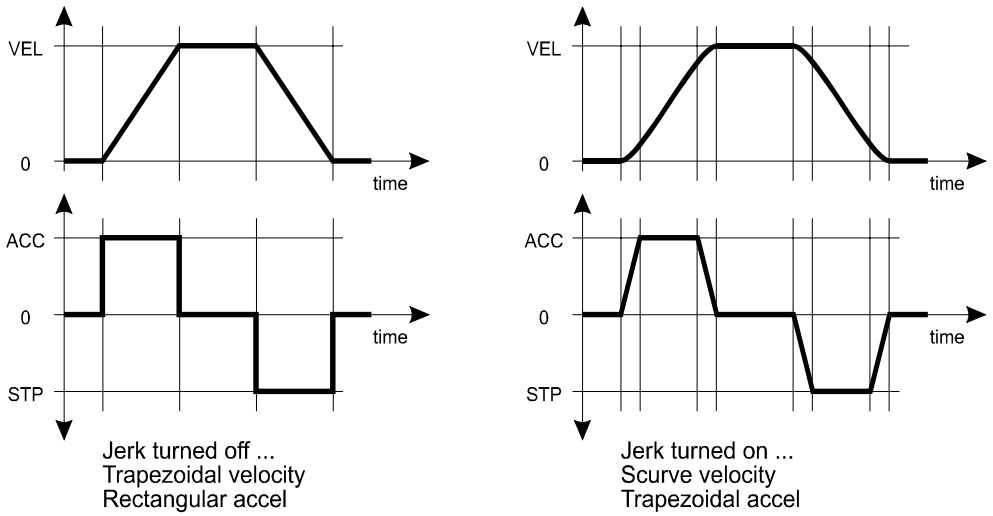


Figure 8

- VECDEF - This command controls how the master move vector is calculated. The argument passed to an axis determines how much the axis contributes to the vector calculation. In many multi-axis configurations, it is not necessary (nor desirable) to have all the axis contributing to this calculation. EXAMPLE: Setting an rotary axis value to zero will remove that axis from vector calculation. VECDEF X1 Y1 A0
  Default value is 1 for all axes.
- VECTOR - This command allows manual override of the default vector length calculations for moves. Entering a VECTOR command will make a move of a non-vectored axis as if the vectored axes made a move of that distance.
  Default value is 0.

The following example makes an X,Y move with A axis interpolation. The VECTOR command is then used to move the A axis by itself as if X and Y were moving along a vector that is 1200 units in length.

```
10 VECDEF X1 Y1 A0
20 X10000 Y10000 A270
30 VECTOR 1200
40 A0
50 VECTOR 0
```

SETTING FEEDBACK CONTROL COMMANDS

Feedback control commands are used to condition the encoder feedback used by the axes attached to the current program to control the command position. Each axis must have these factors set individually and the axis name must be used when using these commands. Multiple axis can be included when issuing these commands. Issuing the command without an argument will display the current setting.

NOTE: These commands affect velocity profile commands and are NOT scaled when the values are changed. Caution must be taken when these are changed to prevent unpredictable or erratic motion, or the possibility of motor runaway.

- MULT - This command sets up count direction and hardware multiplication for the encoder attached to a given axis. This command can be used with the DAC GAIN command to establish direction of rotation for an axis.
  The format for this command is MULT {***name mode***} {***name mode***}
  To set the multiplier for X axis to 4 use MULT X4

∗ The valid values for the first argument is
  The name specified in the ATTACH SLAVE command

∗ The valid values for the second argument are
  0 - encoder turned off, no quadrature counts.
  1 - count up on rising edge of A channel.
  2 - count up on both edges of A channel.

4 - count up on both edges of A channel and B channel.
-1 - count down on rising edge of A channel.
-2 - count down on both edges of A channel.
-4 - count down on both edges of A channel and B channel.
Default value for all axes is 1

- PPU - This command sets the pulses per programming unit for each axis attached to the program. This allows programming the motion profile in a convenient unit for the application such as inches, degrees, revolution, etc. Each axis has it's own independent PPU that may be set to any value and is not associated with any other axis. - Default value for all axes is 1

  The format for this command is PPU {***name ratio***} {***name ratio***}

  To set the pulses per unit of X axis at 10000 pulses per inch use PPU X10000

To set the pulses per unit of X axis to

- REN - This command loads the command position register with the actual encoder position. This command is used for the axis to learn the current axis position and zero the D/A command signal.

  The format for this command is REN {***name*** } {***name*** } To reset the command position registers of axis X and Y use REN X Y

- RES - This command zeros out the command position and the actual encoder position of the specified axes. An optional preload position is loaded into the registers.

  The format for this command is RES {***name preload***} {***name preload*** } To reset the command position and encoder position registers of axis X and Z and pre-load axis Z to 10 units use RES X Z10

- ROTARY - This command sets the rotary axis length used for shortest distance calculation. The resultant moves will never be longer than half the rotary axis length. Incremental moves are not affected by the rotary axis length.

  The format for this command is ROTARY {***axis length***} {***axis length***}

After you have entered the desired information entering an ESAVE command will store the information. If an ESAVE command is not issued, system parameter, master profile, and attachment information will not be retained and all information will revert to default value.

THE NEXT STEP

ACR CARD PARAMETERS AND FLAGS

Control of the characteristic of all aspects of operation and setting the many limits and setpoints is done by using these parameters and flags.

The system parameters are listed in Appendix A of the User's guide. A definition of these parameters can be found in the Appendixes at the end of this training manual.

* Flag parameters
* Object parameters
* Misc. parameters
* Program parameters
* Master parameters
* Axis parameters

These parameters can be viewed by using a PRINT command and the desired parameter. Such as PRINT P6144 or using the immediate mode shortcut ?P6144. To change the value of a parameter use Pxxxxx=yyyyy such as P12291=1000

NOTE: A parameter that is automatically changed by ACR Card operation will not retain a forced value. Care must be taken when changing a parameter value as changing the value of the wrong parameter can cause unpredictable results.

The flag references are listed in Appendix B of the User's guide. A definition of these flags can be found in the Appendixes at the end of this training manual.

* Optoisolated I /O flags
* Miscellaneous flags
* User flags
* Expansion I / O flags
* Master flags
* Axis flags
* Program flags
* PLC flags
* I / O stream flags
* User flags

These references are used as bits, they are either ON or OFF. The status of the bit can control an operation or indicate a wide variety of information. To view the status of a flag in immediate mode use the PRINT and BIT command and the desired flag. Such as PRINT BIT0 or ?BIT128.
If the flag is OFF (CLEAR) the response will be 0, if the flag is ON (SET) the response will be -1. One of the most common uses of bit status is in a program IF/THEN statement.

To change the status of a flag in program or immediate mode use the SET and CLR command. Such as SET 0 to turn ON a flag and CLR 0 to turn OFF a flag. You may also use the format BITx=Y such as BIT128=0.

NOTE: A flag that is automatically changed by ACR Card operation will not retain a forced condition. Care must be taken when changing a flag value as changing the status of the wrong flag can cause unpredictable results.

EXAMPLES USING PARAMETERS AND FLAGS

Checking flag parameter status

The following program uses typical examples of program operation controlled by bit status

```
PROG4
HALT
NEW
10 IF (BIT0 = -1) THEN GOTO 1000
20 IF (BIT0 = 0) THEN GOSUB 2000
30 DWL 1 :REM > Dwell (pause) for 1.0 seconds.  Minimum value 1 mSec.
40 GOTO 10
1000 PRINT “The switch is closed”
1010 DWL 1
1020 GOTO 10
2000 PRINT “The switch is open”
2010 DWL 1
2020 RETURN

LRUN
```

The above commands are provided on disk in a file named SWITCH.8K

Feedhold / CycleStart

While the Square pattern program is running, Setting the Feedhold and CycleStart bit flags will stop and restart the motion. Assuming the SQUARE pattern is being done with MASTER0 in PROG0,

```
SET 520
```

The above command will cause the axes to STOP using the DEC deceleration value.

```
SET 521
```

The above command will cause the axes to resume from the Feedhold state using the ACC acceleration value.

Checking encoder position

When the Square pattern program is in feedhold, check the location of axis X and axis Y by printing the encoder position parameter for the axis.

```
PRINT P6144
```

The above command will print the current value of the encoder position object parameter for axis X

? P6160
The above command will print the current value of the encoder position object parameter for axis y

PARAMETRIC EVALUATION

The on board evaluator can solve complex expressions in floating point or fixed point math. Arguments to most commands can be specified as littorals or expressions. The expression can be either numerical or string manipulations.
When an expression is used as an argument, it must be enclosed within parentheses. For example:

X(P0+P2*P30)
Assuming that the global variables P0..P30 have been pre-defined, the above command will cause the X axis to move to the resulting value of P0+P2*P30.

IF(P0=1234) then Print “OK”
The above command will result in the message “OK” to be printed if P0 is equal to 1234.

$V0=$V1+$V2
Assuming that String variables V0..V2 have been defined, the above command will concatenate $V1 and $V2 and put the result into $V0

Sample program using parametric evaluation

This program will generate a random number between 0 and 4000. If the random number is 1234 the program will halt. It will also print out how many loops it took to come up with that number.

```
PROG3
HALT
NEW
5 DIM LV(2)
10 LV0 = 0
20 LV1 = RND(4000)
30 LV0 = LV0 +1
40 IF (LV1 <> 1234) THEN GOTO 20
50 PRINT “DONE IN ”; LV0; “ TRIES”
LRUN
```

The above commands are provided on disk in a file named EVAL.8k

Simple Input / Output

This program will flash the first 30 outputs in a random manner.

```
PROG3
```

```
HALT
NEW
5  DIM DV(1)
10 DV0 = RND(4294967295)
20 P4097 = DV0
30 GOTO 10
RUN
```

The above commands are provided on disk in a file named IOTEST.8K

DIRECT AXIS ACCESS OF ANALOG OUTPUT

The closed loop operation of an axis can be controlled using two unique direct access commands, or with bit flags designated for that purpose.

AXIS OFF - This command will turn off the servo loop associated with that axis. Once the servo loop is open, the analog output for that axis can be directly controlled by entering a value in the object parameter that controls the DAC output or using the DAC command. The DAC output will remain at that value until it is changed by entering a new value into the parameter or turning the axis ON.
The format for this command is AXIS {***index*** } OFF To turn off servo loop for axis 7 - AXIS7 OFF

NOTE: Turning off unnecessary servo loops will reduce CPU load and improve system performance. The default powerup value for all 8 available axes is AXIS ON. It should become common practice for any experienced ACR Card user to actively turn off any axis not installed on the controller card in use.

AXIS ON - This command will turn on the servo loop associated with that axis. Once the servo loop is closed, axis operation will be determined by the servo loop core. DAC output will be determined by the value of following error generated by the secondary setpoint parameter compared to actual encoder position.
The format for this command is AXIS {***index*** } ON To turn off servo loop for axis 7 - AXIS7 ON

Another method of controlling the state of the servo loop is to use the axis control flag parameter Open Servo Loop, this flag has the save affect as AXIS OFF when the flag is SET.

DATA SAMPLING CONTROL (SAMP)

The Sample Command allows simultaneous sampling of 8 channels at a sampling rate that can be as fast as the Servo Interrupt. The sampling can also be set to occur at a fixed frequency
The base format for SAMP commands is SAMP {***channel***} ***command*** {***data***)

- The first argument selects one of the eight available sample channels.
- The second argument is what command is to be performed.

* The third argument is the data to be used for that command.

The following is a list of commands used for SAMP.

- SAMP SRC - Specifies the sample source for the given sample channel. This source can be any system parameter or user defined parameter. (See SRC command on Page 28 for the definition of the "*sourcedef* " argument.)
- SAMP BASE - Specifies storage array base for the given sample channel. This array can be a 32-bit long integer array (LA) or a 32-bit long floating point array (SA)

NOTE: The source and base must be of the same number type (integer or floating point) since no data conversion is done during the transfer.

- SAMP CLEAR - Clears sample channels of all system parameters and flags which are related to data sampling.
- SAMP TRG - Specifies the trigger condition to be monitored to start sampling when the sample trigger armed flag is set. A positive or un-signed index will trigger on an active state or rising edge condition. A negative index will trigger on an inactive state or failing edge condition.

There are associated parameters and bit flags that work in conjunction with the sample command. The definitions of all of these can be found in the User's Guide. The two of primary importance that need to be addressed when using any sampling are defined as follows.
BIT 104 - Enables monitoring of the data sample trigger which starts the sampling process. This flag is cleared when all of the sample channels have been filled, indicating that the sample has completed.
BIT 106 - Selects continuous mode sampling when this flag is cleared. In this mode, a sample is taken every servo interrupt until all sample channels have been filled. Selects edge trigger mode when this flag is set. A sample is taken during the transition of the selected trigger source.

The following program will illustrate the use of the sample command. It samples the command position parameter and the output voltage parameter of axis 0 using the 1 second clock pulse flag as the sample trigger. After the sample is completed, the contents of the base array can be displayed.

```
PROG0
HALT ALL
NEW ALL
DETACH ALL
ATTACH MASTER 0
ATTACH SLAVE0 AXIS0 "X"
VEL 5000 ACC 500 DEC 500 STP 500
MULT X4 Y4
PPU X1 Y1
10 DIM LV2
20 DIM LA1
30 DIM LA0(50)
40 DIM SA1
```

```
50 DIM SA0(50)
60 DIM $V(1,1)
100 SAMP CLEAR          - Clear out all system parameters and flags related
                          to sampling.
110 SAMP0 SRC P12288    - Select actual position of axis 0 for source of
                          sample 0
120 SAMP0 BASE LA0      - Select storage array base for sample 0
130 SAMP1 SRC P12319    - Select output voltage of axis 0 for source of
                          sample 1
140 SAMP1 BASE SA0      - Select storage array base for sample 1
150 SET 106             - Select edge trigger mode.
160 SAMP TRG +82        - Trigger the sample on the rising edge of the 1
                          Second clock pulse
170 SET 104             - Enable monitoring of the data sample trigger
180 X /190000
190 INH -104            - Inhibit until sampling complete
200 PRINT " "
210 PRINT "SAMPLE COMPLETED"
220 PRINT " "
230 PRINT "PRESS 0 TO VIEW SAMPLE 0"
240 PRINT "PRESS 1 TO VIEW SAMPLE 1"
250 INPUT "ANY OTHER KEY TO END  ", $V0
260 IF ($V0 = "0") THEN GOTO 300
270 IF ($V0 = "1") THEN GOTO 400
280 PRINT "PROGRAM TERMINATED"
290 END
300 LV0 = 0
310 PRINT LA0(LV0)
320 LV0 = LV0+1
340 IF (LV0 < 50) THEN GOTO 310
350 GOTO 220
400 LV0 = 0
410 PRINT SA0(LV0)
420 LV0 = LV0+1
440 IF (LV0 < 50) THEN GOTO 410
450 GOTO 220
500 END
```

The above commands are provided on disk in a file named SAMP.8K

SETTING AN EXTERNAL TIMEBASE (SRC)

The source command SRC sets the timebase for coordinated motion for the master attached to that program. By default, the velocity profile uses the CLOCK source, feeding a single time unit per interrupt. Redirecting the source allows an external timebase to be used. This will send a time unit per source pulse that is then multiplied times the period. The JOG SRC command controls the jog timebase in the same manner Source commands such as CAM SRC, or RATCH SRC condition the "input encoder" used with that function.

NOTE: Only input pulse in the positive direction will be used in calculating the change in source pulses when using the SRC commands that control timebase for master or jog.

The base format for SRC commands is SRC {***sourcedef***} The following is a list of the valid source definitions. These definition are valid for all source devices used with the ACR Card.

sourcedef	**description**
NONE	- Disconnect device from source
CLOCK	- Connect to servo clock (1 pulse per period)
ENC {***encoder***}	- Connect to encoder register
encoder	- Connect to encoder register
RATCH (***ratchet***}	- Connect to ratchet output
parameter	- Connect to user or system parameter
RES (***preload***)	- Reset or preload internal source count
REN	- Match internal source count to external input

The following commands will attach two axes to program 0 and set the source for the master to encoder 2. By using a velocity of 2000 pulses (period of 0.5 mSecs), the speed of coordinated motion will be equal to that of encoder 2. If the velocity is set at another value, the speed will be proportional. By setting the velocity to 1000, the speed will be 1/2 that of source encoder.

```
PROG0
HALT ALL
NEW ALL
DETACH ALL
ATTACH MASTER0
ATTACH SLAVE0 AXIS0 "X"
ATTACH SLAVE1 AXIS1 "Y"
MULT X4 Y4
PPU X1 Y1
SRC ENC2
ACC 100000  DEC 100000  STP 100000
VEL 2000
```

The above commands are provided on disk in a file named SOURCE.8K

Issue an immediate mode axis move of 10000 pulses to the two attached axes
X10000 Y10000
The axes do not move until the remote encoder is turned in the positive direction, the axes will move at the same speed as the encoder.
Repeat the following using different VEL settings to observe operation of external timebase.

SINGLE AXIS VELOCITY PROFILE (JOG)

Each axis of the ACR Card has its own single axis velocity profiling or "JOGGING" . Jogging sets up an individual velocity profile for each axis based on the current jog parameters for that axis. These parameters act independently of the parameters set by any master motion profile that controls the axis command position. Jog uses it own parameters and bit flags for each axis.
When a jog motion is started, the profile generates a jog offset. This offset is used during the summation of the primary setpoint.
The base format for jog commands issued in a program space where the axis is attached is JOG ***command*** {***name data***}. The alternate format for jog commands that can be used in any program space is AXIS***X*** JOG ***command*** {***data***}

The following is a list of the velocity profile commands used for JOG.

- JOG VEL - Set jog target velocity
- JOG ACC - Set jog acceleration
- JOG DEC - Set jog deceleration
- JOG SRC - Set external timebase (See SRC command on Page 28 for the definition of the "*sourcedef"* argument)
- JOG RES - Clear or preload the jog offset parameter of the axis and adds the difference to the current position of that axis.
- JOG REN - Clear or preload the current position of the axis and adds the difference to the jog offset parameter of that axis.

NOTE: The default values of JOG VEL, ACC, and DEC are zero, to initiate jog motion these values need to be set. Jog uses the feedback control settings that have been set for the axis

The following is a list of the motion commands used for JOG.

- JOG FWD - This command initiates a ramp to the velocity programmed by the JOG VEL command in the positive direction.
- JOG REV - This command initiates a ramp to the velocity programmed by the JOG VEL command in the negative direction.
- JOG OFF - This command initiates a ramp down to zero
- JOG ABS - This command will use the current jog settings to jog an axis to an absolute jog offset value.
- JOG INC - This command will use the current jog settings to jog an axis an incremental distance from the current jog offset value.

The following immediate commands will set the parameters for two axes attached to PROG0 to allow testing of jog operation.

```
PROG0
HALT
NEW
```

```
DETACH
ATTACH MASTER0
ATTACH SLAVE0 AXIS0 “X”
ATTACH SLAVE1 AXIS1 “Y”
MULT X4 Y4
PPU X1 Y1
ACC 100000 DEC 100000 STP 100000
VEL 4000
JOG VEL X1000 Y1000
JOG ACC X25000 Y25000 DEC X25000 Y25000
```

The above commands are provided on disk in a file named JOGTEST.8K

Check the current value of command position for axis X
?P12288
Check the current value of jog offset for axis X
?P12297

Move axis X with the MOV command, this changes the command position. Jog offset for axis X will not change.
X10000 - Axis X will move to the absolute command position of 10000
Check the value of command position and jog offset for axis X as previously shown.

Move axis X with the jog command, this changes the jog offset value. Command position for axis X will not change. Observe that the jog movement will be 1/4th the speed of the move because of the current jog setting.
JOG FWD X - Axis X will jog continuously in the positive direction.
Check the value of command position and jog offset for axis X as previously shown.

Stop the current jog move with the JOG OFF command. Use the axis format from a program location other then program 0.
AXIS0 JOG OFF - Axis X will ramp down to a stop.
Check the value of command position and jog offset for axis X as previously shown.

Try assorted combinations of move and jog formats to see the effect. If an axis has a commanded move and a jog move at the same time, the move will occur simultaneously. Try the REN, RES, JOG REN, and JOG RES commands.

STEPPING FARTHER

USING PROGRAM INHIBITS

The following commands are used to inhibit or interrupt a program by accessing specific parameters or flag..

INH -32
The above command will cause the program to halt further program execution until the specified bit is in the selected state. In this example, wait until Output 32 is de energized.

IHPOS P6144(10000,1.5)
The above command will cause the program to halt further program execution until the specified parameter passes the given 'Setpoint' or the 'time-out' parameter is reached. In this example; the command will cause the program to wait until ENC0 position becomes greater or equal to 10000 pulses or the 1.5 second time limit has been reached. The latency is within 1 interrupt.

INT -32 X(100000,2000)
The above command will cause the move to be interrupted if the specified bit condition is met and the incremental move is executed from that point. In this example; the command will cause the X axis to move towards absolute position of 100000 and stop. But before it reaches there, if Output 32 is de energized, it will go an additional 2000 pulses from where the output changed state and stop.

HARDWARE POSITION CAPTURE

10 INTCAP X2
20 INH 777
30 PRINT P12292
Assuming that the X axis is attached to Encoder0, the above commands will cause the program to wait at line 20 until the RISING edge of Input 24 is seen. This will latch the Encoder0 position into P12292 with a latency of only 1 Microsecond.
Using this command, an efficient WEB handling routine can be written that determines the number of pulses seen between the occurrences of two Triggers. Then the Gear ratio of the controlled axis can be manipulated to maintain a fixed distance between the web and the applicator/cutter axis.

The following program will illustrate the use of the INTCAP command. It counts how many pulses have gone by since the last time the marker pulse was seen in axis 0.
SYS
DIM P5

```
PROG0
HALT
NEW
DETACH
ATTACH MASTER0
ATTACH SLAVE0 AXIS0 “X”

 5  P0=0
10 VEL 500
20 X10000000000
30 INTCAP X0
40 INH 777
50 IF (P0=1) THEN GOTO 60
60 P1 = P12292  : P0 = (P0+1)  :GOTO 30   - SKIP THE FIRST TIME
70 P2 = P12292  : P3 = (P2 - P1)           - GET DIFFERENCE
80 PRINT P3                                - PRINT OUT THE
                                             DIFFERENCE
90 P1 = P2
100 GOTO 30

LRUN
```

The above commands are provided on disk in a file named INTCAP.8K

Now PROG0 will print out the number of pulses gone by between each marker pulse. Note that the velocity of the motor is kept slow to allow the printing to keep up with the motor speed. If the printing was not being done, the speed can be increased as needed.

Doing Arcs

The basic method for doing circles and arcs is the SINE command. Use the formulas shown in the User’s Guide and the following information to determine the values to use in the SINE command.

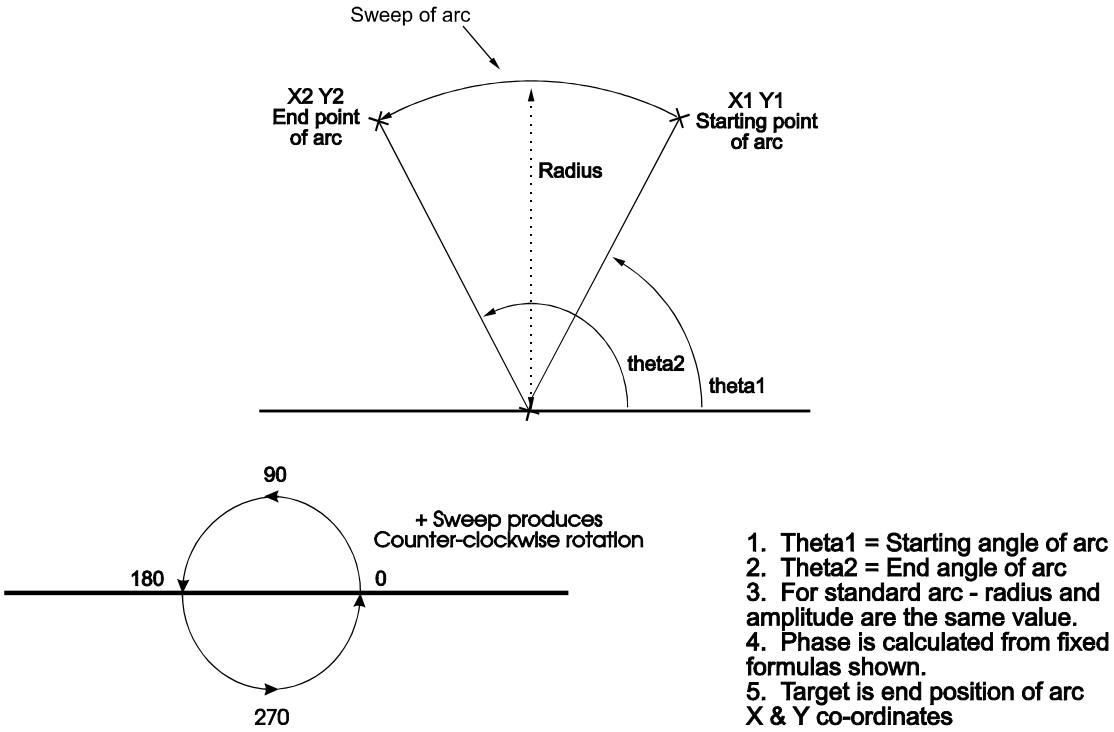


Figure 9

```
PROG0
HALT
NEW
DETACH
ATTACH MASTER0
ATTACH SLAVE0 AXIS0 “X”
ATTACH SLAVE1 AXIS1 “Y”

10 X0Y0
20 X10000 Y0
30 SINE X(0,90,90,10000) SINE Y(10000,0,90,10000)
40 GOTO 10
RUN
```

The above program will move from the center of the arc to 3 o’clock, then describe a counter clockwise arc to 12 o’clock and then go back the center of the circle.
Verify this arc by using the SET GAIN / MONITOR Screen in AcroView.

The above commands are provided on disk in a file named ARC.8K

You can verify this arc by using AcroView and the SET GAIN screen. To set the gain screen to show the X Y coordinated moves in the above program, the gain setup must be set to the following.

1. The axis selected must be Y axis **AXIS:Y** - Set with F1 (AXIS).

2. The horizontal resolution should be set to 1000 pulses per division. **HORI:1000U/Div** - Set by using F3 (SETUP) then F1 (HORIZ).

3. The vertical resolution should be set to 1000 pulses per division. **VERT:1000U/Div** - Set by using F3 (SETUP) then F2 (VERT).

4. The horizontal monitor type must be set to X Axis **HorMon:X** - Set by using F3 (SETUP) then F7 (HORIZTYPE).

5. The source should be set to current position. **SRC:CurPos** - Set with F4 (SOURCE).

The settings of the other parameters will have no effect on this operation. To see the arc being created by the above program, press the F7 (MONITOR).

Software ratchet command (RATCH)

The RATCH commands are used to setup software ratchets. There are eight available ratchets - RATCH0.....RATCH7. Ratchets are software sources that condition the input pulses applied to it from the designated source. The ratchet output pulses are then used as the source for another device.

The source pulses applied to a ratchet can be handled in one of four ways by the ratchet

1. Normal - All source pulses are passed through the ratchet.
2. Ignore - Source pulses in this direction make not change on ratchet output
3. Negate - Source pulses in this direction produce ratchet output pulses in the opposite direction.
4. Buffer - Source pulses in this direction are added to an internal count and cause no change on ratchet output. Pulses in the other direction are first used to unbuffer previously buffered pulses. When there are no more pulses to unbuffer, the ratchet tracks normally.

The following is a list of commands used for RATCH

- RATCH SRC - This command sets the input source for a ratchet. (See SRC command on Page 28 for the definition of the "*sourcedef*" argument)

The format for this command is RATCH ***X*** SRC ***sourcedef*** - The default value of sourcedef is none.

- RATCH MODE - This command sets the conversion mode for a ratchet.

The format for this command is RATCH ***X*** MODE ***mode*** - The default ratchet mode is zero.

The following is a table of ratchet modes and their affect on incoming source pulses. Not all modes have practical applications.

mode	positive pulses	negative pulses
0	normal	normal

1	normal	ignore
2	normal	negate
3	normal	buffer
4	ignore	normal
5	ignore	ignore
6	ignore	negate
7	ignore	buffer
8	negate	normal
9	negate	ignore
10	negate	negate
11	negate	buffer
12	buffer	normal
13	buffer	ignore
14	buffer	negate
15	buffer	buffer

The following commands will attach two axes to program 0 and set the source for the master to RATCH0. RATCH0 source will be set to use the remote encoder connected to ENC2. The ratchet will start with the default mode 0.

```
PROG0
HALT ALL
NEW ALL
DETACH ALL
ATTACH MASTER0
ATTACH SLAVE0 AXIS0 “X”
ATTACH SLAVE1 AXIS1 “Y”
MULT X4 Y4
PPU X1 Y1
SRC RATCH0
RATCH0 SRC ENC2
RATCH0 MODE0
ACC 100000  DEC 100000  STP 100000
VEL 2000
Change the ratchet mode to observe response.
```

The above commands are provided on disk in a file named RATCH.8K

Some further examples of the use of the software ratchet will be cover later when software gearing is introduced.

PLC OPERATION

The ACR Card can accommodate up to 8 PLC programs in it's multi-task environment. These programs are created in the same manner as user programs, but differ in several important ways.

1. PLC programs use a limited instruction set that is compiled into machine code for high-speed execution.
2. This instruction set does NOT contain any motion commands or reference to parameters.
3. Each program can contain a maximum of 100 instructions.
4. PLC programs can NOT contain remarks (REM statements)

PLC programs are linked into the PLC scanner, this 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. This event is executed in its entirety and the next event is executed during the next interrupt. This process is repeated after the last item in the list.

In addition to any PLC programs, the scanner event list also contains two other events.

⇒ The input/output/clock update scan.

This is always in the PLC scanner list, this event updates the Optoisolated digital I/O, the global system clock, and the clock tick flags.

NOTE: The input/output/clock update scan is <u>always</u> in the PLC scanner list and active.

⇒ The timer/counter/latch update scan.

This event updates the condition of the timers, counters, and latches that are available in the ACR Card.

PLC COMMANDS

The following is a list of commands related to PLC programming

- PLC - Switches the communication channel to the designated PLC prompt. PLC ***number***
- PON - Initializes the PLC scanner list previously discussed. Must be executed if the bit flags and parameters for timers, counters, or latches are to be used from normal user programs.
- POFF - Resets the PLC scanner list to contain only the input/output/clock update event.
- RUN - Command first "compiles" the PLC program source and then "links" the result into the PLC scanner list. Executes an implied PON command.
- HALT - Removes the current PLC program from the PLC scanner list. Does not affect other events in the PLC scanner list.
- LIST - This command lists the currently selected PLC program.

- MEM - This command displays the amount of free memory remaining in the current PLC space. The memory that is displayed only reflects the memory used by the source storage.

NOTE: On average, a total of 32 bytes of storage is required for each PLC instruction, 8 bytes for the source and 24 bytes for the "compiled" machine code. This memory usage should be considered when allocating memory space for PLC programs.

PLC INSTRUCTION SET

PLC instructions are combined to create PLC programs. Each instruction represents either the contact or coil of a relay on a traditional ladder logic diagram. In terms of this ladder diagram; a "relay" is any ACR Card bit flag, a "contact" is an instruction that monitors the state of a bit flag, and a "coil" is the instruction that controls the state of a bit flag.
For those not familiar with ladder logic diagrams, a logic block or "ladder rung" represents a complete Boolean logic equation. An example of this Boolean logic is (X AND Y = Z) which states that if X is a logic 1 AND Y is a logic 1 then Z is a logic 1. This would be represented in a ladder diagram as a NO (normally open) contract of relay X wired in series with a NO contact of relay Y connected to the coil of relay Z. The contacts of relay Z would then be used to represent the output of the Boolean equation.

The following is a list of instructions related to PLC programming

- LD - Start a logic block with a NO (normally open) contact.
- LD NOT - Start a logic block with a NC (normally closed) contact.
- AND - Add a NO contact in series to the logic block.
- AND NOT - Add a NC contact in series to the logic block.
- OR - Add a NO contact in parallel to the logic block.
- OR NOT - Add a NC contact in parallel to the logic block.
- AND LD - Connect the two most recent open (not connected to a "coil") logic blocks in series to create a new logic block.
- OR LD - Connect the two most recent open logic blocks in parallel to create a new logic block.
- OUT - Connect the current logic block to a bit flag ("coil") and close the block.
- TIM - Connect the current open logic block to the given "timer coil" Format - TIM ***timer*** There are eight global PLC timers TIM0.....TIM7
- CNT - Connect the two most recent open logic blocks and connects them to the given "counter". Format - CNT ***counter*** There are eight global PLC counters CNT0.....CNT7
- KR - Connects the two most recent open logic blocks and connects them to the given "latch" Format - KR ***latch*** There are eight global PLC latches KR0.....KR7
- PBOOT - This instruction will set the run request flag for that PLC on power-up. This instruction should be the first instruction of the PLC program.

- END - This instruction indicates the end of the current PLC program. Typically, the last instruction of a PLC program, it can be placed anywhere in the program to aid in troubleshooting a program.

SIMPLE PLC FUNCTIONS

PLC0..PLC7 allows direct Ladder logic programming to be accomplished.

The following program will turn on output 32 if Input 00 and Input 01 are turned on simultaneously

```
PLC0
HALT
NEW
10 LD 00
20 AND 01
30 OUT 32
RUN
```

This program is provided on the disk in a file called PLCTEST.8K.

ADVANCED FUNCTIONS

ELECTRONIC GEARING

Gearing allows slaving an axis to pulses from a selected gear source. The source pulses are scaled by a ratio that is equivalent to a gearbox ratio on a mechanical system. The rate at which the ratio changes is controlled by a ramping mechanism similar to a clutch or a variable speed gearbox. These pulsed are fed into the gear offset parameter of a slave axis. The commands can also be issued using the "handwheel" commands. There is only one internal mechanism for electronic gearing per axis, so either the HDW or GEAR commands can be used but not both.

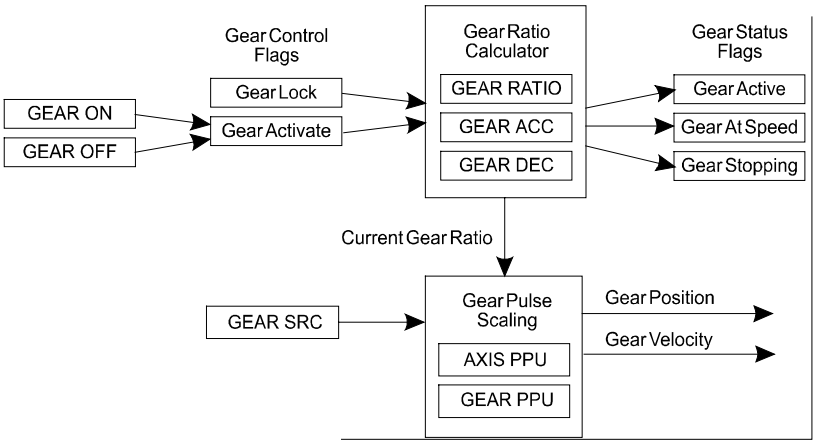


Figure 10

The base format for gear commands is GEAR ***command*** {***axis data***}
The base format for handwheel commands is HDW ***command*** {***axis data***}

The following is a list of commands used for GEAR

- GEAR SRC - Set electronic gearing source. (See SRC command on Page 28 for the definition of the *"sourcedef"* argument)

- GEAR PPU - This command establishes the relationship between the source encoder pulses and the "input shaft" of the electron gearbox.
  Default value is 1.0 pulses per unit.

- GEAR RATIO - This command sets the ratio between the "input shaft" and the "output shaft" of an electronic gearbox. The ratio format is expressed in (output units/input units).
  Default value is 1.0 pulses per unit. A value of GEAR RATIO ***X*** (1/10) or GEAR RATIO ***X*** 0.1 will supply one unit of "output shaft" motion for ten units of "input shaft" motion.

- GEAR ACC - Set acceleration ramp for the "geared" axis

- GEAR DEC - Set deceleration ramp for the “geared” axis

- GEAR RES - This command either clears or preloads the gear offset parameter for the given axis and adds the difference to the current position parameter of that axis. If the “offset” parameter is not used, the gearing offset parameter is set to zero.

- GEAR ON - This command enables electronic gearing for an axis. The optional “offset” parameter is preloaded to the gear offset parameter and adds the difference to the current position.

- GEAR OFF - This command disables electronic gearing for an axis. The optional “offset” parameter is preloaded to the gear offset parameter and adds the difference to the current position.

Sample electronic gearing example

The following commands will illustrate the use of electronic gearing. It will make an electronic gearbox that will attach Y axis as the “geared” axis to X axis as the “gear source” with a ration of 10/1. (One revolution of X axis will produce ten revolutions of Y axis)

```
PROG0
HALT
NEW
DETACH
ATTACH MASTER0
ATTACH SLAVE0 AXIS0 "X"
ATTACH SLAVE1 AXIS1 "Y"
PPU Y1            - Set PPU for axis Y
GEAR SRC Y0       - Set the electronic gearing source for axis Y (Use the X
                    axis encoder ENC 0)
GEAR PPU Y1       - Scale electronic gearing input
GEAR RATIO Y10 - Set gearing ratio at ten to one (Y axis will move 10 times
                    X axis move)
GEAR ON Y         - Turn on electronic gearing
```

The above commands are provided on disk in a file named GEAR.8K

The following command will make Y axis stop following X axis.

```
GEAR OFF Y
```

Effect of a software ratchet

The response of electronic gearing can be greatly modified with the use of the GEAR SRC and the software ratchet . To demonstrate the responses from a software ratchet used as an “input encoder”, the previous example will have the source changed and various ratchet modes will be used.

Enter the following to modify the current gear settings.

```
GEAR OFF Y            - Turn off software gearing
```

GEAR SRC Y RATCH1 - Set the gear input source to ratchet 1
RATCH1 SRC ENC0 - Set the software ratchet source to the encoder of axis X
GEAR RATIO Y1 - Set the gearing ratio to 1 to 1.
GEAR RES Y - Reset gear offset parameter to 0
RES X Y - Reset X and Y axis Command parameter and Actual encoder parameter to 0

⇒ Start with RATCH1 MODE0
Move X axis 10000 pulses and the Y axis will move the same distance. Return X axis to 0 and the Y axis will move the same distance. Move X axis 15000 pulses and the Y axis will follow.

⇒ Change to RATCH1 MODE1 - Reset gear and axes
Move X axis 10000 pulses and the Y axis will move the same distance. Return X axis to 0 and the Y axis will not move. Move X axis 15000 pulses and the Y axis will move 15000 pulses positive to a position of 25000 pulses.

⇒ Change to RATCH1 MODE2 - Reset gear and axes
Move X axis 10000 pulses and the Y axis will move the same distance. Return X axis to 0 and the Y axis will move 10000 pulses positive to 20000. Move X axis 15000 pulses and the Y axis will move 15000 pulses positive to a position of 35000.

⇒ Change to RATCH1 MODE3 - Reset gear and axes
Move X axis 10000 pulses and the Y axis will move the same distance. Return X axis to 0 and the Y axis will not move. Move X axis 15000 pulses and the Y axis will move 5000 pulses positive to 15000.

ELECTRONIC CAM AND BALLSCREW COMPENSATION.

The electronic cam feature is used to replicate mechanical cam systems that where a standard in industry. The physical cam features of the mechanical cam are replaced with entries stored in user defined arrays. The drive motor that was used to turn the physical cam and establish cam timing is replaced by the cam source to generate an index into the cam arrays.
As the cam source encoder moves, it generates an index into a table of offset values. If this index falls between to table entries, the cam offset is linearly interpolated between the entries. This offset value is then scaled, shifted by the output offset, and multiplied by the PPU for the given axis. This value is the cam offset parameter that affects the primary setpoint value.
A cam table can be composed of more than one segment with each segment having different distances between table entries. This allows some parts of the table to be defined coarsely and others to be defined in more detail. The cam table wraps around if it goes off either end of the table.

Ballscrew compensation is used to compensate for nonlinear position error introduced by the mechanical construction of ballscrews. The commands used by ballscrew (BSC) and the commands used by electronic cam (CAM) are identical. The main difference between the two are; the ballscrew source by default uses the primary setpoint for the axis and the value of the ballscrew offset parameter is used in calculating the secondary setpoint.

The base format for cam commands is CAM ***command*** {***axis data***}
The following is a list of valid commands used for CAM

- CAM DIM - This command allocates working space for a cam. A cam must be dimensioned before it can be initialized.

- CAM SEG - This command defines the allocated cam segments.

- CAM SRC - Specifies the source for the input “cam pointer” of a cam. (See SRC command on Page 28 for the definition of the “*sourcedef*” argument)

NOTE: Cam source command must be entered **AFTER** the cam segments have been defined. Improper operation may result from designating the cam source first

- CAM ON - Enable cam output for the designated axes

- CAM OFF - Disables cam output for the designated axes.

- CAM SCALE - Sets the cam output scaling for an axis. When an cam offset value is calculated, it is multiplied by the cam output scaling factor and then multiplied by the PPU of the axis.

- CAM OFFSET - Sets the cam output offset of an axis. After the cam output offset value is calculated, this value is added before it is multiplied by the cam scaling factor. This shifts the cam table output by the offset value.

- CAM FLZ - Sets the cam input offset of an axis. The cam input offset is added to the cam table index before it is used to calculate the actual table index. This shifts the cam table index by the FLZ value.

- CAM SHIFT - Sets the incremental cam shift. Whenever an incremental cam crosses a segment boundary, the difference between the two entries is used to adjust the cam shift.

- CAM RES - This command is used to clear or preload the cam offset and clear out any cam shift that has built up for a given axis. The cam offset parameter value is transferred into the current position parameter.

Sample Cam Program

```
HALT
NEW
CLEAR
REM  -    Dimension two longint arrays
DIM LA(2)

REM  - Data for CAM Segments are entered in PULSES not units
REM  - If the PPU for the axis is not 1 use CAM SCALE value of 1/PPU

REM  -  Enter data for array 0
DIM LA0(9)
LA0(00) = 0
LA0(01) = 73
LA0(02) = 250
LA0(03) = 427
LA0(04) = 500
LA0(05) = 427
LA0(06) = 250
LA0(07) = 73
LA0(08) = 0

REM  -  Enter data for array 1
DIM LA1(5)
LA1(00) = 0
LA1(01) = 0
LA1(02) = -500
LA1(03) = -500
LA1(04) = 0

REM  -  Allocate two cam segments for X axis
```

```
CAM DIM X2

REM  -  Define CAM segment 0 for 500 units long and use LA0 for its data
CAM SEG X(0,500,LA0)

REM  -  Define CAM segment 1 for 1000 units long and use LA1 for its data
CAM SEG X(1,1000,LA1)

REM  -  Set pointer to the memory area for encoder 1
CAM SRC X1

REM  To activate cam type CAM ON X. When axis Y is moved, axis X motion
REM  will follow the CAM data
```

The above commands are provided on disk in a file named CAM.8K

PROGRAMMABLE LIMIT SWITCH

The programmable limit switch feature is used to replicate mechanical drum or cam timer systems that where a standard in industry. The drum timer used a series of physical switches that were activated by placement of physical high points on a revolving "drum". The timing of the activation of these switches was controlled by the speed of the "timing" motor used to rotate the "drum" These functions are implemented by the PLS object as follows. The physical switches are replaced by the destination pointer which can be any long integer parameter. The physical high and low points that determined the state of the switches is replaced by a user defined array. The "timing" motor is replaced with an encoder source input.

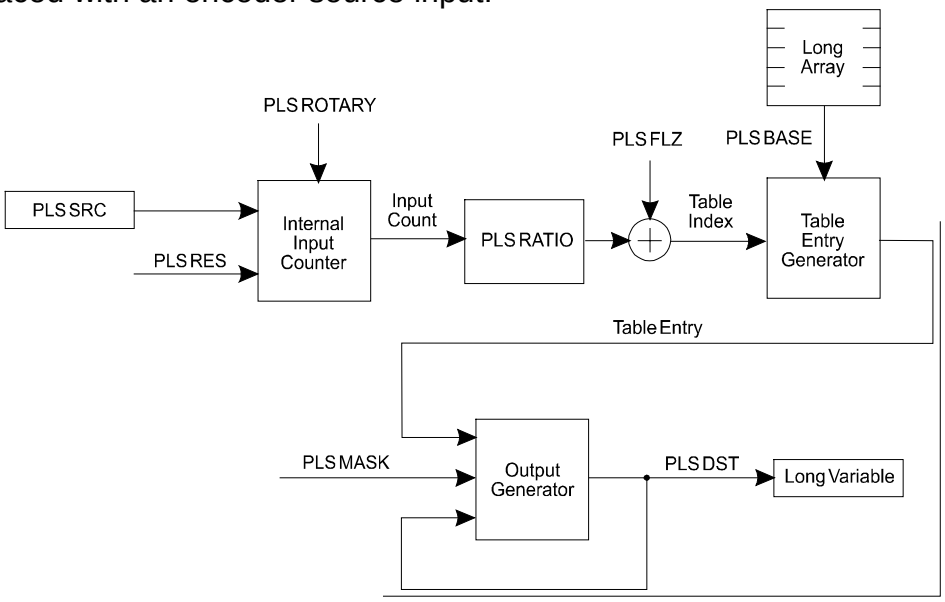


Figure 11

There are a total of eight programmable limit switches available on the ARC8000 controller - PLS0.....PLS7. By default the PLS affects the selected subset of Outputs 32....63 (digital output parameter - P4097). The state of the outputs is stored in a user defined array. The pointer into this array is the actual position of the selected encoder. The latency of the PLS switching the outputs is 1 Servo Interrupt cycle with the PLC off.

The base format for pls commands is PLS ***index*** COMMAND {***data***}
The following is a list of valid commands used for PLS

- PLS SRC - Specifies the source for the input of a PLS. (See SRC command on Page 28 for the definition of the "*sourcedef*" argument)

- PLS DST - Sets the PLS destination pointer. Any long integer parameter can be used, the default is P4097.

- PLS BASE - Sets the PLS array pointer. The array must first be allocated since there is no default array.

- PLS RES - This command resets or preloads the PLS internal input counter. This command is only used if the PLS is set to rotary operation.

- PLS ROTARY - Sets the PLS rotary length.
  The default rotary length is zero counts, this sets the PLS to linear operation. In linear operation, any table index that is generated that is outside the boundaries of the PLS array will zero the PLS output.
  If the rotary length is set to any other value the PLS will operate in rotary mode. The source parameter is used to generate an input count that "wraps-around" by the given length before it is used to generate a table index.

- PLS FLZ - Sets the index offset in array entries. The default offset is 0 array entries.

- PLS MASK - Sets the PLS output bit mask. This determines which destination output bits will be transferred. The default mask setting is -1, this allows all bits to be transferred.

- PLS RATIO - Set the scaling ratio. If rotary mode is used, the formula to determine scaling is (# of array entries - 1) / rotary length.

- PLS ON - This command enables PLS update.

- PLS OFF - This command disables PLS update.

When setting up a Programmable Limit Switch (PLS) there are some factors that need to be determined.

1. How many entries are to be in the lookup table
- Used to determine the number of base array entries

This is the equivalent of the number of high and low points on a mechanical drum or cam assembly. This number determines the size of the array that will be dimensioned for the base array

2. What is the maximum count of the PLS source
- Used with the array entries to determine scaling

Determine what is the amount of motion that the PLS source will travel and if the count will return to that range or continue, if it is to continue then PLS rotary must be selected.

3. Calculate the scaling ratio to use
- (Number of array entries/maximum source encoder length in pulses)

Example of calculating scaling ratio. If you have 8 entries in the lookup table and the source traveled 2000 pulses the ratio would be 8/2000=0.004 The PLS ratio entry will be 0.004

4. Determine which outputs that are to controlled by the PLS
- Used to determine what value to use for the destination pointer

Determine the values to be placed in the base array. This determines what the state of the bits of the output parameter will be when within that lookup table entry. the 32 bit binary number can represented in binary as 11111111111111111111111111111111 or in hexadecimal as FFFFFFFF. Now determine which outputs you require ON in the lookup entry that you are calculating, as an example: 00000000000000000000000000001111 in binary, 0000000E in hexadecimal, 15 in decimal. the value in the array is the decimal number 15.

5. Which outputs of the destination pointer are to be controlled by the PLS and which will not be affected by the PLS

- Used to determine the MASK value to be used.

By default, all 32 bits of the destination pointer are controlled by a PLS when it is ON. To limit the number of bits controlled by a PLS, a PLS MASK is used.

To limit the PLS to the first eight bits of the destination pointer the MASK decimal value 255 is used, the positions affected is shown in the binary value 00000000000000000000000011111111 shown in hexadecimal 000000FF

To limit the PLS to the second eight bits of the destination pointer the MASK decimal value 65280 is used, the positions affected is shown in the binary value 00000000000000001111111100000000 shown in hexadecimal 0000FF00

6. Determine what destination pointer to use for PLS

- Any long integer parameter can be used.

7. Will the source encoder stay within the PLS range or continue to move outside of the PLS encoder range

- This is used to determine whether this PLS will be rotary. If the source encoder goes negative or greater than the maximum count used to determine the scaling, all bits in the pointer will be held at zero.

By selecting PLS ROTARY the encoder count is wrapped back to zero when the maximum number is reached.

NOTE: If rotary mode is to be used the formula changes to
(# of entries - 1) /length

Sample PLS Program

Sample programmable limit switch with a lookup table using the first eight bits of default destination parameter P4097 All entries can be made from any program prompt, no line number are required to test operation.

DIM LA(1) - Allocate space for one long integer array

DIM LA0(9) - Allocate space for nine entries in the array

PLS0 BASE LA0 - Use the data in LA0 for the lookup table for PLS0

PLS0 SRC ENC2 - Set encoder 2 as the source pointer

PLS0 MASK 255 - The first eight outputs will be controlled by PLS0, the upper 24 will not be effected by PLS0 operation
Outputs controlled by PLS0 - 32,33,34,35,36,37,38,39

PLS0 Ratio 0.0040 - (# of lookup entries -1) / length (9-1) / 2000 = 0.0040

PLS0 ROTARY 2000 - The encoder pointer will continue above the number used in the scaling calculation.

LA0(0) = 1
LA0(1) = 15
LA0(2) = 63
LA0(3) = 255
LA0(4) = 252
LA0(5) = 240
LA0(6) = 128
LA0(7) = 0
LA0(8) = 0

Encoder position	Array Value	Outputs
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

PLS0 ON
To see PLS operation slowly move the remote encoder and ENC2 will supply the input for the PLS.

The above commands are provided on disk in a file named PLS.8K

THE FINAL STEPS

BOOLEAN LOGIC EQUATION WITHIN A PROGRAM

ACR Card bit flags can be used to perform logical expressions within programs besides there use in PLC programs. These logical expressions will occur at a slower rate so the PLC method is preferred.
The following logic commands are available for use

- AND - Bit in the result will be set if both expression bits are set.
- BIT - Bit flag status
- NAND - Bit in the result will clear if both expression bits are set.
- NOR - Bit in the result will clear if either expression bit is set.
- NOT - Invert status of bit
- OR - Bit in the result will be set if either expression bit is set.
- XNOR - Bit in the result will be set if both expression bits are set or clear.
- XOR - Bit in the result will set if only one expression bit is set.

IF (BIT0 AND BIT1) THEN JOG FWD X
IF (BIT516 OR BIT548) THEN PRINT “System is ready”

SERIAL PORT CONTROL FROM WITHIN A PROGRAM

For standalone applications, the ACR Card has built in commands to facilitate talking to a “dumb” terminal and getting keystrokes.
The following commands are available for use

- OPEN - Opens a communication port for use in a program
- KBHIT - Checks if there is a character in the buffer waiting to be read
- INKEY$ - Returns a waiting Key in the buffer. Returns a null if buffer is empty.
- PRINT - Prints literal or an expression to the port.
- GETCH - Returns a character from the buffer. Waits if buffer is empty.
- CLOSE - Closes the open communication port.

- INSTR - Allows string manipulation.
- LCASE$ - Converts string expression to lower case.
- LEFT$ - Returns Leftmost “n” characters from a string.
- LEN - Returns the length of a string.
- MID$ - Returns characters from the middle of a string.
- RIGHT$ - Returns rightmost “n” characters from a string.
- SPACE$ - Returns a string of “n” spaces.
- STR$ - Converts a numerical expression to a string.
- STRING$ - Returns a string of specified characters.
- UCASE$ - Returns an upper case image of a string.
- VAL - Converts a string expression into a number.

The following program illustrates the use of a serial communication prodigal to enter information into an ACR Card program.

```
SYS
PROG0
HALT
NEW

10 PBOOT
20 DIM $V(1,10)
30 OPEN "COM1:19200,N,8,1" AS #1
40 PRINT #1,
50 PRINT #1, "What kind of fruit do you want?"
60 PRINT #1, "(A)pple, (B)anana, (C)oconut"
70 PRINT "I would like to have a(an) ";

100 $V0 = UCASE$(INKEY$(1))
110 IF ($V0 = "A") PRINT  "APPLE" : GOTO 50
120 IF ($V0 = "B") PRINT  "BANANA" : GOTO 50
130 IF ($V0 = "C") PRINT  "COCONUT" : GOTO 50
140 IF ($V0 = CHR$(27)) GOTO 200
150 IF ($V0 = "X") GOTO 200
160 IF ($V0 = "" ) GOTO 100

170 PRINT #1, " " : PRINT #1, "Not a valid choice "
180 PRINT " " : PRINT " INVALID SELECTION"
```

```
190 GOTO 50

200 PRINT #1, "PROGRAM TERMINATED"
210 CLOSE #1
```

The above program is provided on disk in a file named INKEY.8K

GLOBAL OBJECTS

Analog input control (ADC)

The ACR Card has the capability to perform Analog to Digital Conversion using the optional ADC input module. The analog input module normally converts eight single-ended ±10 volt signals to a 12-bit signed fixed point number reserved for that ADC input. The channels can be optionally redirected to be read as differential pairs.

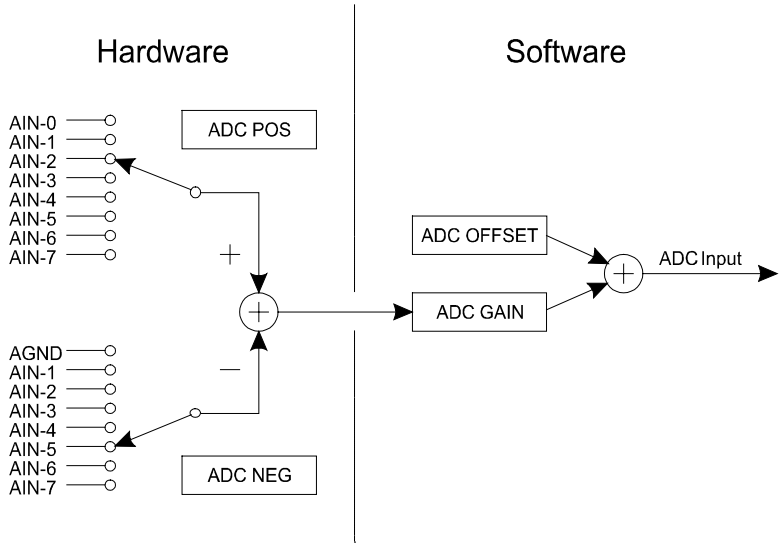


Figure 12

When ADC updating is enabled, the readings from the analog input module are internally scaled to generate a base number of ±1.0 input units. This number is then multiplied by the ADC GAIN setting and then the ADC OFFSET value is added. The result is then stored in the ADC input object parameter as a 32-bit floating point number. When the ADC input is used as the feedback device for an axis, the number is converted into an longint (Long Integer)

The base format for ADC commands is ADC {***index***} ***command*** {***data***)

- The first argument selects one of the eight available ADC inputs.
- The second argument is what command is to be performed.
- The third argument is the data to be used for that command.

The following example will select ADC channel 2 as the positive input for a differential pair for ADC 4 - ADC4 POS2

The following is a list of the commands used for ADC

- ADC POS - This command sets the positive input for differential analog conversion.
- ADC NEG - This command sets the negative input for differential analog conversion.
- ADC GAIN - This command sets the analog gain for software conversion.
- ADC OFFSET - This command sets the software offset for analog conversion. This command is typically used to offset an ambient analog value to remove it from having any effect on digital output.
- ADC ON - This command enables the update of the analog input module. This command does not have an ***index*** or ***data*** argument.
- ADC OFF - This command disables the update of the analog input module. This command does not have an ***index*** or ***data*** argument.

ADC0 GAIN4095

The above command will take the ±1.0 input unit generated by ADC input 0 and multiply that by 4095 add the value of ADC0 OFFSET and store that number in ADC0 input object parameter. This will give ADC0 a range of -4095 for a -10 volt input to +4095 for a +10 volt input with the default value of ADC0 OFFSET 0.0.

Analog output control (DAC)

The DAC commands give direct access to the D/A converter software adjustments. The output of the ACR Card D/A converter use the 32-bit floating point DAC Output Signal parameter and produces a 15-bit signed fixed point number that is sent to the DAC module. The value of that number is controlled by DAC parameters.

The base format for DAC commands is DAC {***index***} ***command*** {***data***)

- The first argument selects one of the eight available DAC outputs.
- The second argument is what command is to be performed.
- The third argument is the data to be used for that command.

The following is a list of the commands used for DAC

- DAC GAIN - The default gain is 3276.8 dac_units / volt. With this value, the actual value of the voltage out of P2 will be equal to the DAC output signal parameter. There is an inversion produced in the DAC module so a DAC output signal parameter value of 1.0 will actually produce a -1.0 volts at P2 with default gain.
- DAC OFFSET - This offset is added to the output signal parameter before it is multiplied by the DAC GAIN. This value will be added to the output continuously and is typically used to offset error in the servo amplifier If a DAC OFFSET value of 0.5 is entered, the actual voltage produced at P2 will be offset by a value of -0.5 volts from the value of the DAC output signal parameter.

EXAMPLE - An application for this would be if the output to be used was a single ended controller that uses +5.0 VDC as the stationary position and 0.0 VDC was full speed counterclockwise rotation and +10.0 VDC was full speed clockwise rotation. By setting the DAC OFFSET of that axis at -5.0 then with the output signal parameter at 0.00 the DAC output voltage would be +5.0 VDC. When the output signal parameter changed to +2.5, the DAC output voltage would be +2.5 VDC. When the output signal parameter changed to -5.0, the DAC output voltage would be +10.0 VDC.
Use a TLM setting of TLM X(0,-10) (Covered in next section)
NOTE: THERE IS AN INVERSION BETWEEN THE OUTPUT SIGNAL PARAMETER AND THE DAC OUTPUT VOLTAGE.

Quadrature input control (ENC)

The ENC commands give direct access to the encoder reset and multiplier setup.

- ENC RES - This command will set the encoder position object parameter for the axis to zero. It will not affect any other parameters.

NOTE: Care must be taken with this command when used with a closed servo loop axis. This command will only reset the encoder position object parameter and not change the command position. This could generate a large following error which can cause axis runaway.

- ENC MULT - This command will set the encoder multiplier in the same manner as the MULT command. The advantage to using this command is the encoder need not be attached. This is useful for encoders used as master encoder or handwheel encoders.

AXIS LIMITS

There are multiple software limit commands that are provided with the ACR Card.
The following is a list of available axis limits

- ALM - Sets the command position limits monitored by the “A limit” flags. When the command position of a given axis is within these limits, the appropriate flag is set. For masters, the flag is set if all of its slaves are within their limits. Two values are entered for these limits, using the format (+limit, -limit). If the limits are of equal value, one number can be used. Default value is (0,0) for all axes.
  Examples - ALM X10 Y(30,-20)

- BLM - Sets the command position limits monitored by the “not B limit” flags. When the command position of a given axis is outside of these limits, the appropriate flag is set. For masters, the flag is set if any of its slaves are outside of their limits. Two values are entered for these limits, using the format (+limit, -limit). If the limits are of equal value, one number can be used. Default value is (0,0) for all axes.
  Examples - BLM X(10,-5) Y15

- EXC - Sets the following error limits monitored by the “not excess error” flags. When the following error of a given axis is within its excess error band, the appropriate flag is set. For masters, the flag is set if all of its slaves are within their excess error bands. Two values are entered for these limits, using the format (+limit, -limit). If the limits are of equal value, one number can be used. Default value is (0,0) for all axes.

- IPB - Sets the following error limits monitored by the “not in-position” flags. When the following error of a given axis is outside of its in-position band, the appropriate flag is set. For masters, the flag is set if any of its slaves are outside of their in-position bands. Two values are entered for these limits, using the format (+limit, -limit). If the limits are of equal value, one number can be used. Default value is (0,0) for all axes.

- ITB - Sets the voltage limits monitored by the “not-in-torque band” flags. When the output voltage of a given axis is outside of its in-torque band, the appropriate flag is set. For masters, the flag is set if any of its slaves are outside of their in-torque bands. Two values are entered for these limits, using the format (+limit, -limit). If the limits are of equal value, one number can be used. Default value is (0,0) for all axes.
- JLM - This command sets the jog limits for an axis. The jog limits are only checked when the “jog limit check” bit is set and the JOG FWD or JOG REV commands are in use. When the “jog limit check” bit is set, the JOG FWD or REV command will jog to the value of jog limit and stop. Two values are entered for these limits, using the format (+limit, -limit). If the limits are of equal value, one number can be used. Default value is (0,0) for all axes.

- TLM - This command sets the voltage limits monitored by the “not torque limit” flags. When the output voltage of a given axis in not being torque limited, the appropriate flag is set. For masters, the flag is set if none of its slaves are being torque limited. Two values are entered for these limits, using the format (+limit, -limit). If the limits are of equal value, one number can be used. Default value is (0,0) for all axes.

NOTE: The limits set by the TLM command cause the output of the servo loop to be clipped at the given values.

APPLICATIONS

APPLICATION #1

IMPLEMENTING HOME CYCLE

This program performs a homing routine as follow:
1. Move toward home switch
2. When home switch is closed, motor will reverse direction and move away from home switch until switch is cleared
3. Motor will reverse direction and move toward home switch until encoder marker is found
4. Encoder position is reset.

Commands used:
ACC, AXIS, DEC, GOTO, HALT, IF/THEN, INH, LRUN, MASTER, MSEEK, NEW, PBOOT, PRINT, PROG, SET, SLAVE, STP, VEL

Master flags used:
Bit 516 - MASTER 0 IN MOTION
Bit 523 - MASTER 0 STOP MOVE REQUEST

Hardware required:
Servo motor using DAC 0 (P2-1,20)
Servo motor encoder using ENC 0 (P1A-1,2,3,4,5,6,7,8)
Home position switch connected to Optoisolated input 0 (P4-1,Ext.Gnd)

```
REM   -> Activate system prompt level
SYS

REM   -> Switch the communication channel to program 0
PROG 0

REM    -> Stop operation of any running program and kill any motion profile
REM         activated by that program
HALT

REM   -> Erase the current program
NEW

REM   -> Cancel master and slave attachments to program 0
DETACH

REM   -> Attach Master Profile 0 for use by PROG 0
ATTACH MASTER0
```

```
REM   -> Attach axis 0 to the first internal slot for master profile 0 and
refer to the axis with the string X
ATTACH SLAVE0 AXIS0 "X"


80  PRINT " Starting Home cycle for AXIS X "
90  PRINT " "

100 ACC 20000        : REM Set Acceleration ramp value
110 DEC 20000        : REM Set Deceleration ramp value
120 STP 20000        : REM Set stop ramp value
140 IF (BIT 0) THEN GOTO 300      : REM If ON Home Switch then
move away from switch

200 PRINT " -> Seeking home switch"
210 VEL 10000        : REM Set velocity toward home switch
220 X /10000000      : REM Do an incremental move toward home switch
230 INH 0            : REM Wait until home switch is closed
240 SET 523          : REM Issue a Stop Move Request
250 INH -516         : REM Wait for motor motion to stop

300 PRINT " -> ->  Clearing home switch"
310 VEL 1000         : REM Set velocity away from home switch
320 X /-10000000                          : REM Do an incremental move
away from home switch
330 INH -0           : REM Wait until home switch is open
340 SET 523          : REM Issue a Stop Move Request
350 INH -516         : REM Wait for motor motion to stop

400 PRINT " -> -> ->  Seeking marker"
410 VEL 1000         : REM Set velocity for marker seek
420 MSEEK X(10000,0)     : REM Initiate a marker seek operation

500 PRINT "   Home Cycle Complete  "
```

Start program with LRUN command to run program and have communication channel linked to the programs output so that print commands will be displayed.

To return to command prompt issue a ESCAPE character

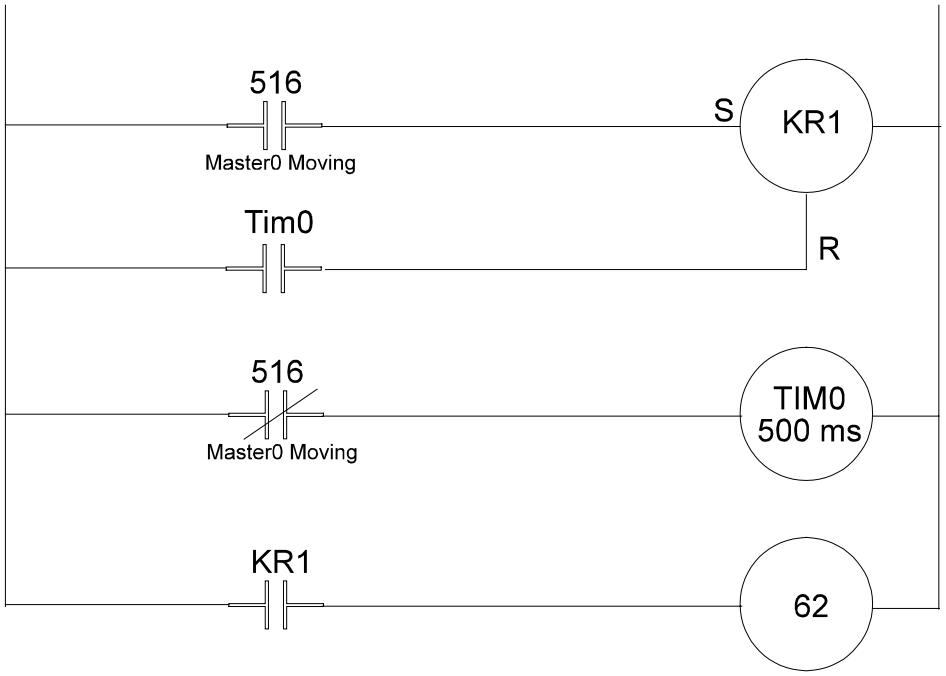
Incremental moves used should be of sufficient length to assure that the home switch will change states BEFORE the move is complete

The Stop Move Request flag bit 523 will clear when the move has been stopped

APPLICATION #2

IMPLEMENTING A PLC PROGRAM

Implement a TIMER that turns an output on if each axis is moving and turns off that output if no movement is detected for 1/2 second.



Note that as soon as the X axis starts to move KR1 sets. Then when the axis stops moving, TIMER0 starts to time 1/2 seconds. When the timer counts out, then KR1 will reset. So actually KR1's output is the desired result for us. The last rung of the ladder links KR1 to a physical Output

PLC0
HALT
NEW
10 LD 516
20 LD TIM0
30 KR1
40 LD NOT 516
50 TIM0 500
60 LD KR1 OUT 62
70 END
RUN

PROG0
HALT
NEW
10 ACC 100000:DEC100000:STP100000: VEL 3000

```
20 X1000
30 DWL 5
40.X0
50 DWL 5
60 GOTO 10
```

Prog0 moves the X axis motor back and forth. While this is happening, Bring up the I/O Status up on the screen and watch what happens to Output 62.

APPLICATION #3

MULTITASKING PROGRAM.

This program shows an example of multiple programs that use global parameters and program variables to determine operation of sections of other programs. The programs operate as follows:

⇒ PLC 0 - Monitors condition of input 0 and input 1 to see if they are both ON at the same time.
⇒ Program 1 - Sets the velocity for the motion of axis 1 and monitors input 30 to increase velocity by 5% and input 31 to decrease velocity by 5% . Holds velocity between set minimum and maximum values. Also monitors user flag set by PLC 0 to determine an error condition.
⇒ Program 2 - Contains motion profile and performs the actual motion of axis 1 using the velocity set in program 1. Sets the status of the user flags used by program 3.
⇒ Program 3 - Displays the velocity set by program 1, the encoder position of axis 1 after the moves controlled by program 2, and the error condition monitored by PLC 0. Clears the status of all flags in user group 0

- Commands used:

ACC, AND, ATTACH, AXIS, CEIL, CLEAR, DEC, DETACH, DIM, DWL, END, HALT, IF/THEN, INH, LD, LRUN, MASTER, NEW, OUT, PBOOT, PLC, PRINT, RUN, SET, SLAVE, STP, SYS, USING, VEL

- Master flags used:

Bit 548 - Master 1 IN MOTION

- General flag Parameters:

P4100 - User group 0

- User flags used:

Bit 128 - User flag 0
Bit 129 - User flag 1
Bit 130 - User flag 2
Bit 131 - User flag 3

- Global Parameters:

P0,P1,P2

- Programs used:

PLC 0
PROG 1

PROG 2
PROG 3

- Miscellaneous Outputs:

LED bank output 0
LED bank output 1

- Hardware required:

Servo motor using DAC 1 (P2-2,21)
Servo motor encoder using ENC 1 (P1A-9,10,11,12,13,14,15,16)
Velocity increase switch connected to Optoisolated input 0 (P4-1, Ext.Gnd)
Velocity decrease switch connected to Optoisolated input 1 (P4-2, Ext.Gnd)

Use of ALL in the following program statements used to clear entire system without the need of changing to each program. Care should be taken when using this technique because programs and attachments can NOT be recovered

```
REM   -> Activate system prompt level
SYS

REM   -> Stop operation of any running programs and kill any motion profile
REM      activated by those programs
HALT ALL

REM   -> Erase all current programs
NEW ALL

REM   -> Cancel master and slave attachments to all programs
DETACH ALL

REM   -> Clear memory allocations
CLEAR

REM   -> Allocate memory for PLC program 0
DIM PLC0(5000)

REM   -> Allocate memory for programs 1,2, & 3
DIM PROG1(5000)
DIM PROG2(5000)
DIM PROG3(5000)

REM   -> Allocate memory for 3 global variables
DIM P(3)

Remarks are not allowed in a PLC program
so the REM statements for this program will
be done before switching to PLC0
REM   10 Program will self-start when power is applied
REM   20 Start a block with the status of Optoisolated input 30
```

```
REM   30 AND the status of Optoisolated input 31
REM   40 Connect block to User flag 0
REM   50 End of PLC program

PLC 0
HALT
NEW

10 PBOOT
20 LD 0
30 AND 1
40 OUT 128
50 END

RUN

REM   -> Switch the communication channel to program 1
PROG 1

10 PBOOT
REM Program will self-start when power is applied

20 P0 = CEIL(2500)
REM Set initial velocity using global variable P0  CEIL command returns the
smallest integer equal to or greater than expression

30 P1 = CEIL(1000)
REM Set minimum velocity using global variable P1

40 P2 = CEIL(15000)
REM Set Maximum velocity using global variable P2

50 P4100 = 0
REM Clear flags in User group 0

100 IF (BIT128) THEN GOTO 100
REM Check status of User flag 0 used by PLC 0

110 IF (BIT131) THEN GOTO 110
REM Check status of User flag 3 used by PROG 1

200 IF (BIT0) THEN P0 = CEIL(P0 * 1.05)
REM Increase velocity by 5% when INPUT 0 is ON

210 IF (BIT1) THEN P0 = CEIL(P0 * 0.95)
REM Decrease velocity by 5% when INPUT 1 is ON

300 IF (P0 < P1) THEN P0 = P1
REM Check for minimum velocity
```

```
310 IF (P0 > P2) THEN P0 = P2
REM Check for maximum velocity

400 BIT40 = NOT BIT40
REM Change state of program heartbeat OUTPUT 40

410 SET 131
500 GOTO 100

RUN

PROG 2

REM   -> Attach Master Profile 1 for use by PROG 2
ATTACH MASTER 1

REM   -> Attach axis 1 to the first internal slot for master profile 1
REM      and refer to the axis with the string BERT
ATTACH SLAVE 0 AXIS 1 "BERT"

10 PBOOT
20 ACC 20000          : REM Set Acceleration ramp value
30 DEC 20000          : REM Set Deceleration ramp value
40 STP 20000          : REM Set stop ramp value
100 IF (BIT128) THEN GOTO 100
110 IF (BIT130) THEN GOTO 110

120 VEL (P0)
REM Velocity for master profile 1 is stored in global variable 0

200 SET 96
REM Set LED bank output 0

210 BERT 10000
REM Move motor BERT to absolute position of 10000 pulses

220 INH -548
REM Inhibit program until IN MOTION master flag for Master 1 is clear.

230 DWL 1
REM Dwell program for 1 second

240 SET 129
REM Set User flag 1

300 SET 97
REM Set LED bank output 1
```

```
310 BERT 0
REM Move motor BERT to absolute position of 0 pulses

320 INH -548
REM Inhibit program until IN MOTION master flag for Master 1 is clear.

330 DWL 1           : REM Dwell program for 1 second
340 SET 130         : REM Set User flag 2

400 BIT41 = NOT BIT41
REM Change state of program heartbeat OUTPUT 41

410 CLR 96          : REM Clear LED bank output 0
420 CLR 97          : REM Clear LED bank output 1
500 GOTO 100
RUN

PROG3
10 PBOOT
20 IF (BIT128) THEN GOTO 400
REM Check status of User flag 0

30 PRINT USING "####";"Velocity is "; P0
REM Print value of P0 using format string to determine format of numeric
output

100 INH 129
REM Inhibit program until User flag 1 is Set.

120 PRINT USING "####";"  Position is ";P6160
REM Print encoder 1 position stored in object parameter P6160

200 INH 130
REM Inhibit program until User flag 2 is Set.

210 PRINT USING "####";"   Position is now ";P6160
220 PRINT " "

300 P4100 = 0
REM Clear flags in User group 0

310 GOTO 20
400 PRINT "         -+-+-+- ERROR -+-+-+-"
410 PRINT " Velocity increase and decrease are both set"
420 PRINT " "
430 DWL 1
440 GOTO 20
LRUN
```

Flag Reference Definitions

APPENDIX B
ACR Card Version 1.13.03 USER'S GUIDE

*** NOTE** - The speed reference for the flag parameters uses the following legends

F - Indicates fast flags which have an sample rate of once each servo interrupt.

S - Indicates slow flags which have a sample rate of approximately 50 milliseconds. This rate is independent of servo interrupt rate.

**** NOTE** - Under most conditions, there are flags that should only have their status read and not changed. Changing the status of these flags can have unpredictable and undesired results. The following legends are used to indicate the difference between the types.

R - Indicates flags that should only be read.

W - Indicates flags which can have their status written to or read.

P4112 - P4119 (Master Flag Parameters)

Status Flags	*	**	
Accelerating	F	R	Flag will be set if the master is currently accelerating using the ACC ramp.
Decelerating	F	R	Flag will be set if the master is currently decelerating using the DEC ramp
Stopping	F	R	Flag will be set if the master is currently decelerating to a complete stop using STP ramp
Reserved			Currently not used
In Motion	F	R	Flag set if the master is currently in a move, this Flag will also stay activated when the master is in Feedhold.
Move Buffered	F	R	Flag set if the master currently has a move pending (buffered) while a previous move is in progress. This Flag will clear when there are no moves pending.
Feedholding	F	R	Flag set if the master has received a Feedhold request. The attached axes are decelerating or stopped. This Flag will clear when a Cycle Start request is received.
In Feedhold	F	R	Flag set if the master has received a Feedhold request and the attached axes have stopped. This Flag will clear when a Cycle Start request is received.

Control Flags	*	**	
Feedhold Request	S	W	This Flag will cause the current move to decelerate to zero and set the Feedholding status flag. Processor acknowledgment clears the Flag.
Cycle Start Request	S	W	This Flag will cause the move in progress when a Feedhold Request was processed to accelerate and continue the moves. Processor acknowledgment clears the Flag.
Kill Move Request	S	W	Setting this Flag will stop current move without using any acceleration or deceleration ramps.
Stop Move Request	S	W	Setting this Flag will cause the master to respond the same as receiving a Feedhold Request, wait for "In Feedhold" flag, then follow with a Kill Move Request
FVEL Zero Pending	F	W	State of "FVEL zero" flag at start of next move.
FVEL Zero Active	F	W	When this flag is set, use 0 for FVEL instead of FVEL setting.
FOV Lock Pending	F	W	State of "FOV lock" flag at start of next move.
FOV Lock Active	F	W	When this flag is set, use 1.0 for FOV instead of FOV setting

Limit Flags	*	**	
Not In Position	S	R	Flag will be set whenever any attached axes has a following error that is outside of its In-Position Band (IPB)
Not Excess Error	S	R	Flag will be set if all of the attached axes has a following error that is within their Excess Error bands (EXC)
Within A Limit	S	R	Flag will be set if all attached axes have a command position that is within the values set by Stroke Limit ‘A‘ (ALM)
Not Within B Limit	S	R	Flag will be set if any attached axes has a command position that is outside of values set by Stroke Limit ‘B‘ (BLM)
Not Torque Limit	S	R	Flag will be set if none of the attached DAC outputs is being torque limited. This is set by the value of Torque Limit. (TLM)
Not in Torque Band	S	R	Flag will be set if any attached DAC output is outside of the torque band. This is set by the value of In-torque Band. (ITB)
Reserved			Currently not used.
Reserved			Currently not used.

Sequence Flags	*	**	
Decrement Count	F	W	When this flag is set, the move counter decrements when a move starts.
Increment Count	F	W	When this flag is set, the move counter increments when a move starts.
Interrupt On Move	F	W	When this flag is set, an interrupt is sent to the AT Bus when a move starts.
TRG Pending	F	W	This flag is set when the master is waiting for TRG condition to become true.
Start Move Inhibit	F	W	When this flag is set, buffered moves are prevented from starting. This flag is used internally by the BLK/STEP sequence.
REN Request Flag	S	W	When This flag is set, the attached axes will perform a REN command. Flag is cleared on operation.
Cycle Start Lockout	F	W	When this flag is set, the master will ignore cycle start requests.
Reserved			Not currently used.

P4120 - P4127 (Axis Flag Parameters)

Limit Flags	*	**	
Not in Position	S	R	Flag will be set when following error is outside of In-Position Band (IPB)
Not Excess Error	S	R	Flag will be set when following error is within Excess Error band (EXC)
Within A Limit	S	R	Flag will be set if command position is within the values set by Stroke Limit ‘A‘ (ALM)
Not Within B Limit	S	R	Flag will be set if command position is outside of values set by Stroke Limit ‘B‘ (BLM)
Not Torque Limit	S	R	Flag will be set if attached DAC output is being torque limited. This is set by the value of Torque Limit. (TLM)
Not in Torque Band	S	R	Flag will be set if attached DAC output is outside of the torque band. This is set by the value of In-torque Band. (ITB)
Reserved			Not currently used.
Reserved			Not currently used.

Status Flags	*	**	
Not Marker	S	R	Flag will be set when encoder marker of attached encoder is not present
Capture Complete	F	R	Flag will be set on completion of a encoder capture using the INTCAP command. Flag will clear when the next INTCAP is issued.
Reserved			Not currently used.
Reserved			Not currently used.
Sinusoidal Mode	F	W	When this bit is set, linear moves will be converted into SINE commands. (SEE SINE COMMAND FOR DETAILS)
Reserved			Not currently used.
Reserved			Not currently used.
Reserved			Not currently used.

Control Flags	*	**	
Clamp Output Signal	F	W	When this flag is set, the axis "Output Signal" parameter is forced to zero.
Open Servo Loop	F	W	When this flag is set, the axis does not execute servo loop.
Biquad Filter Activate	F	W	When this flag is set, the axis executes digital filters.
REN Request Flag	S	W	When this flag is set, the axis executes a REN command. Flag is cleared on operation
Gearing Active	F	W	Flag is set when a GEAR ON command is issued and gear is active for this axis.
Reserved			Not currently active
Cam Active	F	R	Flag is set when a CAM ON command is issued and cam is active for this axis.
Ballscrew Active	F	R	Flag is set when a BSC ON command is issued and ballscrew is active for this axis.

Jog Flags	*	**	
Jog Active	F	R	Flag is set when the axis is in any jog mode.
Jog Direction	F	R	Flag is set when the axis is jogging negative and off when jogging positive or not jogging.
Jog at Speed	F	R	Flag is set when the axis speed is at the velocity set by JOG VEL command.
Jog Stopping	F	R	Flag is set when axis is ramping to zero.
Jog Forward	F	W	Flag is set when axis is jogging forward. Can be used to start jog forward by setting the flag or stopping jog forward by clearing the flag.
Jog Reverse	F	W	Flag is set when axis is jogging in reverse. Can be used to start jog reverse by setting the flag or stopping jog reverse by clearing the flag.
Jog Limit Check	F	W	When this flag is set the jog limits are checked when the JOG FWD and JOG REV commands are used. The value of jog limits is set using the JLM command.
Jog Lockout	F	W	When this flag is set a JOG FWD or JOG REV command will set the appropriate flag but not produce motion. If this flag is cleared and the Jog Forward or Jog Reverse flag is set the Jog will start. If this flag is set while in a jog mode it will not take effect until the present jog is stopped.

P4128 - P4143 (Program Flag Parameters)

Status Flags	*	**	
Program Running	F	R	Flag is set when the program is running.
Program Dwelling	F	R	Flag is set when a program operation is executing a DWL command.
Program Inhibited	F	R	Flag is set when the program operation is suspended during an INH or IHPOS command.
Move Pending		R	INTERNAL USE ONLY
Program Time-out	F	R	Flag is set if IHPOS command times out Cleared by IHPOS command.
Program Modified	F	R	This flag is set when program is modified (add, delete, or edit program lines) Flag is cleared when program is run.
Reserved			Not currently used.
Reserved			Not currently used.

Control Flags	*	**	
Run Request Flag	S	W	If this flag is set, program is RUN when detected. Flag is cleared automatically
Halt Request Flag	S	W	If this flag is set, program is Halted when detected. Flag is cleared automatically
Reserved			Not currently used.
Reserved			Not currently used.
Trace Mode Enable	F	W	Flag is set by TRON command, cleared with TROFF command. Can be used to enable or disable trace mode.
Reserved			Not currently used.
Reserved			Not currently used.
Reserved			Not currently used.

Block Flags	*	**	
Block Control	S	W	Flag is set when a block command is issued for this program. Flag is cleared by AUT command. Can be used to initiate or cancel block mode by setting or clearing the flag.
Block Edge Detect	S	R	This flag reflects previous state of "Block Control" flag. It is used for detecting edges.
Block Mode	F	R	Flag is set as soon as block control is detected if no master is attached. If attached to a master the program will Feedhold and flag will be set when the master "In Feedhold" flag is detected.
Block Mode Waiting	F	R	Flag is set when program is waiting for a STEP command.
Step First Block	F	R	Flag is set by entering block mode operation. Flag is cleared after first step.
Step Request Flag	S	W	This flag is set when a STEP command is issued, this executes the next line of a program is block mode.
Step Control	S	W	Rising edge of this flag causes the "Step Request" flag to be set.
Step Edge Detect	S	R	This flag reflects previous state of "Step Control" flag. It is used for detecting edges.

Pause Flags	*	**	
Pause Control	S	W	Flag is set when a pause command is issued for this program. Flag is cleared by RESUME command. Can be used to initiate or cancel pause mode by setting or clearing the flag.
Pause Edge Detect	S	R	This flag reflects previous state of "Pause Control" flag. It is used for detecting edges.
Pause Mode	F	R	Flag is set as soon as pause control is detected if no master is attached. If attached to a master the program will Feedhold and flag will be set when the master "In Feedhold" flag is detected.
Pause Mode Waiting	F	R	This flag is set when the program is waiting for RESUME command.
Reserved			Not currently used.
Reserved			Not currently used.
Reserved			Not currently used.
Reserved			Not currently used.

P4144 - P4151 (PLC Flag Parameters)

PLC Flags	*	**	
PLC Running	F	R	Flag is set when the PLC is running a program.
First Scan	F	R	Flag is set when the PLC program is run, clears after first scan.
Run Request	S	W	If this flag is set, PLC is RUN when detected. Flag is cleared automatically
Halt Request	S	W	If this flag is set, PLC is Halted when detected. Flag is cleared automatically

Timer Flags	*	**	
Timer Output	F	R	Flag is set when the timer count is zero. The flag will remain set until the Timer input is cleared.
Timer Input	F	W	Setting the flag will cause the timer count to decrement. When the flag is cleared the timer is reset to it's preset value and the timer output flag is cleared.

Counter Flags	*	**	
Counter Output	F	R	Flag is set when the count is zero. The flag will remain set until the counter reset flag is set
Counter Clock	F	W	Counter will decrement once on each rising edge of this flag.
Counter Reset	F	W	When flag is set the counter output is cleared and counter clock inputs are ignored.

Latch Flags	*	**	
Latch Output	F	R	Flag will be set when the latch set flag is set. Flag will remain set after latch set is removed. Flag will clear when latch reset flag is set.
Latch Set	F	W	Setting this flag will set latch output flag.
Latch Reset	F	W	Setting this flag will clear latch output flag, latch set will not function as long as this flag is set.

ACROLOOP MOTION CONTROL SYSTEMS, INC.

ACR8000 Version 1.11

ACR8000 AXIS CONTROL DIAGRAM