This chapter deals with programming the ACR1000 for a few sample applications. Starting with a basic application, this chapter will also explain how to successfully combine a motion control program with a background PLC program.

Application Example 1

This example deals with a task that the axis has to perform over and over again. The operator has push buttons for RUN, CYCLE START, FEEDHOLD, HOME, JOG+ and JOG-. It is desired to have the system go through the HOME sequence after every cycle. Note that for most systems it is neither required or desirable to HOME more than once at power-up. The cycle consists of the following:

1. Seek "HOME" position.
2. Accelerate to a velocity of 2000 and while moving forward 20000 pulses.
3. Decelerate to a velocity of 1000 and continue forward, coming to a complete stop at 60000 pulses.
4. Wait at this point until a user defined switch closure is sensed as open on one of the 12 available inputs.
5. Accelerate to a velocity of 2000 and go back to absolute zero.
6. Check if another switch remains closed. If so, then go to step 1. Else terminate program.

A switch wired to Input #3 tells the system when to retract. A switch wired to Input #10 tells the system whether to repeat the cycle. Note that the following program uses the HOME sequence. This sequence requires setting of parameters P21 through P26, which are described in detail in chapter 6. The following program is suitable for v2.6.x software. If it is being run under v2.7.x, then make sure P28=0 and the velocities should be entered 10 times higher than shown in the example.

```
10 P21=10: REM THIS SETS HOME ACCELERATION
20 P22=10: REM THIS SETS HOME DECELERATION
30 P23=100: REM THIS SETS HOME VELOCITY TOWARDS THE SWITCH
40 P24=10: REM THIS SETS HOME VELOCITY AWAY FROM THE SWITCH
50 P25=0: REM THIS ASSIGN INPUT#0 AS THE HOME SWITCH
60 P26=1: REM THIS SETS DIRECTION TOWARDS HOME AS POSITIVE
70 HOME: REM ZERO SYSTEM
80 ACC 10
90 DEC 10
100 STP 0
110 VEL 1000
120 MOV 20000: REM THIS MOVE WILL STOP WITHOUT DECEL(STP 0)
130 STP 10
```

Sample application 1

This example deals with a task that the axis has to perform over and over again. The operator has push buttons for RUN, CYCLE START, FEEDHOLD, HOME, JOG+ and JOG-. It is desired to have the system go through the HOME sequence after every cycle. Note that for most systems it is neither required or desirable to HOME more than once at power-up. The cycle consists of the following:

1. Seek "HOME" position.
2. Accelerate to a velocity of 2000 and while moving forward 20000 pulses.
3. Decelerate to a velocity of 1000 and continue forward, coming to a complete stop at 60000 pulses.
4. Wait at this point until a user defined switch closure is sensed as open on one of the 12 available inputs.
5. Accelerate to a velocity of 2000 and go back to absolute zero.
6. Check if another switch remains closed. If so, then go to step 1. Else terminate program.

A switch wired to Input #3 tells the system when to retract. A switch wired to Input #10 tells the system whether to repeat the cycle. Note that the following program uses the HOME sequence. This sequence requires setting of parameters P21 through P26, which are described in detail in chapter 6. The following program is suitable for v2.6.x software. If it is being run under v2.7.x, then make sure P28=0 and the velocities should be entered 10 times higher than shown in the example.

```
10 P21=10: REM THIS SETS HOME ACCELERATION
20 P22=10: REM THIS SETS HOME DECELERATION
30 P23=100: REM THIS SETS HOME VELOCITY TOWARDS THE SWITCH
40 P24=10: REM THIS SETS HOME VELOCITY AWAY FROM THE SWITCH
50 P25=0: REM THIS ASSIGN INPUT#0 AS THE HOME SWITCH
60 P26=1: REM THIS SETS DIRECTION TOWARDS HOME AS POSITIVE
70 HOME: REM ZERO SYSTEM
80 ACC 10
90 DEC 10
100 STP 0
110 VEL 1000
120 MOV 20000: REM THIS MOVE WILL STOP WITHOUT DECEL(STP 0)
130 STP 10
```
140 VEL 1000
150 MOV 60000: REM THIS MOVE WILL DECEL TO A STOP (STP 10)
160 INH -3: REM WAIT TIL INPUT 3 OPENS
170 VEL 2000
180 MOV 0: REM THEN MOVE BACK TO 0
185 INH -96: REM WAIT FOR MOVE TO COMPLETE
190 IF 10 THEN 70: REM IF INPUT 10 IS CLOSED, DO IT AGAIN
200 END

Note that the above program will reside in the RAM memory of the controller and will be executed in the program mode. The program can be made to start running in one of several ways. The RUN command can be sent to the ACR1000; or a PBOOT command could be added as the first line in the program, i.e. 0 PBOOT, and the card reset or powered down/up; or perhaps an input could be assigned to control the Run Request relay. Note that a PLC program would have to make sure that the correct input gets "attached" to the Run Request relay. Chapter 10 describes how to do this.

Application Example 2—Using Thumbwheel Input THS16

1-axis control using thumbwheel input

AMCS provides an optional interface, the THS-16, which will allow reading up to 16 digits of thumbwheel data into parameters in the ACR1000. These digits may be read in any grouping. This application requires 1 axis to move by taking distance, velocity and number of passes from a thumbwheel input. Note that if it is required to command the distance...etc. from some other programmable controller, removing the thumb wheels and wiring the BCD lines directly to the host programmable controller will do the trick. For this application, it is assumed that when the operator pushes the START button, the system will read a 8-digit thumbwheel for absolute position, a 4-digit thumb wheel for velocity and a 2-digit thumb wheel for number of passes to move. After each pass the axis will go back to 0. The program to this follows:

0 PBOOT : REM This command causes the program to run on power-up
20 P21=10 : REM Set the home acceleration
30 P22=10 : REM Set the home deceleration
40 P23=100 : REM Set the home velocity towards the switch
50 P24=10 : REM Sets the home velocity away from the switch
60 P25=0 : REM Assign input#0 as the home switch
70 P26=1 : REM Set the direction towards home as positive
99 REM *** TELL OPERATOR MACHINE MUST BE ZEROED ***
100 SET 23 : REM Turn on light saying the machine must be homed
110 GSB 400 : REM Wait for start switch to home
120 HOME :REM Do HOME sequence
130 CLR 23 : REM TURN OFF HOME REQUIRED LIGHT

199 REM *** READ THUMBWHEELS, DO MOTION PROFILE ***
200 GOSUB 400 : REM Wait for start switch to start profile
210 GOSUB 500 : REM Read thumb switches. P25=VEL, P26=Distance, P24=Loops
240 IF P24=0 THEN GOTO 120 : REM HOMES system if 0 loops entered
242 ACC 10 : REM Set up rate controls
244 DEC 0
246 STP 10
250 VEL P25
270 MOV P27 : REM Move the distance in P27
280 MOV 0 : REM then retract
290 ADD -1,P24 : REM Subtracts 1 from loop count
300 IF P24 >0 THEN 270: REM Do while loop count > 0
300 GOTO 200 :REM Go back and read new parameters

399 REM *** WAIT FOR START TO BE PUSHED ***
400 IF 11 THEN 400 : REM input 11 is assigned to 'START'
410 IF 11=0 THEN 410
420 RET

499 REM *** READ THUMBWHEEL DATA ***
500 INP P27,T0,T7 : REM This reads 8 digit distance into P27
510 INP P25,T8,T11:REM This reads 4 digit velocity into P25
520 INP P24,T12,T13: REM This reads 2 digit #of loops into P24
530 RETURN

**Multiaxis System Commanded by a Host**

Let us consider a host computer using two ACR1000 cards to control 2-axes of motion commanded via the serial port. The program will be executed in the command mode. The reason for doing this in the command mode is that as two axes are being controlled, the synchronization is simpler. For this reason, the following sequence will not have line numbers. The host will be reading the In-Position flag from both cards to check if they have completed their moves. It is also desired to make board #2 go half as fast as board #1. The ctrl-A and ctrl-B commands may be terminated by a ASCII character representing the card number, i.e. "0" represents the first card and the zero key on a keyboard, or by a <cr>. A <cr> in this case means "all cards." Other commands are terminated by a <cr>.

```
ctrl-A<cr>
HOME<cr>
The host should fetch status and wait until both cards' run mode relay de-energizes
ACC 10<cr>
DEC 10<cr>
ctrl-B"1"
VEL 2000<cr>
ctrl-B"0"<cr>
ctrl-A"1"<cr>
VEL 1000<cr>
ctrl-A<cr>
MOV 20000<cr>
STP 10<cr>
VEL 1000<cr>
ctrl-B"0"<cr>
VEL 500<cr>
ctrl-A<cr>
MOV 60000<cr>
The host will fetch status and wait until "move-in-progress" relay from both cards gets de-energized.
VEL 2000<cr>
ctrl-B"0"
VEL 1000<cr>
ctrl-A<cr>
STP 10<cr>
MOV 0<cr>
The host will fetch status and wait until the move-in-progress relay from both cards gets de-energized and then repeat entire sequence as many times as needed.
```

As an alternative, the In-Position signals from both the cards could be wired in series so that the host will only have to sense one input to know if both axes are in position. See next section for details on how to do this.

**Multiaxis Application Using IBM PC BUSS (ISA bus)**

When the ACR1000 is plugged into the IBM PC BUSS (ISA bus), the communication is sped up significantly. There are two reasons for this. First, each card has its own unique address on the PC-port address space, so the cards do not have to be turned OFF and ON all the time. Second, because the data
transfer is taking place in an 8-bit parallel fashion, the effective baud rate is very high.

<table>
<thead>
<tr>
<th>TO CARD#1</th>
<th>TO CARD#2</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOME</td>
<td>HOME</td>
</tr>
</tbody>
</table>

The host will wait until both cards signal not Move-In-Progress

TO CARD#1
ACC 10
DEC 10
VEL 2000
STP 10
MOV 20000
VEL 1000
MOV 60000

TO CARD#2
ACC 10
DEC 10
VEL 1000
STP 10
MOV 20000
VEL 500
MOV 60000

The host will wait again for the Move-In-Progress from both cards to set and then clear.

TO CARD#1
VEL 2000
STP 10
MOV 0

TO CARD#2
VEL 1000
STP 10
MOV 0

As an alternative the following could be used.
1. Via the card#2 PLC, direct the card#2 Not-In-Position relay to an unused output.
2. Wire this output to an unused input of card#1.
3. Via the card#1 PLC, direct the card#2 Not-In-Position relay to an unused output ORed with the input that has the card#2 Not-In-Position.

The host could then poll the output status of card#1 and get the status of both Not-In-Position relays.
**Chapter 13 System Start Up and Tuning**

This chapter describes initial setup, setting the GAIN and OFFSET pots on the ACR1000 card. Also described is the setting of the GAIN on systems in which there is no tachometer on the motor. In such a case, the ACR1000 card can generate its own tachometer signal.

**Getting Started**

**Motor Runaway On Motors With A Tachometer**

When power is applied for the very first time on an untested system, the motor could run away. For systems that have a tachometer on the back of the motor, there are two possibilities for motor runaway. The following steps should be taken to resolve the problem:

**Motor/Tach Phasing, First Stage**

There are two stages of stabilizing the controller/amplifier/motor combination. First, with the open loop condition of the amplifier and the motor must be stable. This means that with no command signal going to the amplifier that is in the powered state, the motor shaft must be stationary (or moving very slowly). For the first stage phasing the following steps should be taken:

1. With all power OFF, disconnect the analog signal plug P2 from the ACR1000. This ensures that the servo amplifier is not getting any voltage from the ACR1000.

2. With the amplifier power on, note the motor shaft movement. It should be sitting still or creeping at a very slow rate. (The offset pot on the servo amplifier should be able to eliminate this creep.) If the motor is still, the first stage phasing is now complete.

3. If the motor runs away then the motor leads OR the tachometer leads must be swapped. Do not swap both as that will not cure the problem. The swapping is required because the servo amplifier thinks that the motor is turning in the wrong direction! Swapping either the tach wires or the motor wires will solve the problem. The decision of changing the motor or the tach leads will depend on the direction you want the motor to turn when commanding a positive move from the ACR1000.

**Controller/Amplifier Phasing, Second Stage**

The second stage is the closed loop stabilization of the controller and amplifier. This means that with a stable first stage, when the command signal is applied to the Amplifier and the feedback supplied to the controller, the motor shaft should be in the locked state. The following steps explain the setup.

1. With the first stage phasing complete and P2 plug still disconnected, turn power on and enter a RES command. This will result in the command voltage to be set to 0.0 Volts, then reconnect P2. If the motor sits still and is in the servo locked state, the second stage phasing is complete. If the motor runs away, continue to the next step.

2. Disable the servo amplifier such that the encoder shaft (or motor shaft if the encoder is mounted on the back of the motor) can be turned by hand with the ACR1000 still under power. Check if the encoder pulses are being recognized by the ACR1000 board:
   a. First make sure that parameter P2 is either 0, 1 or 2. If any different, it might disable encoder pulses coming into the board.
   b. Issue a ctrl-U status command. This command will return 8 digits of actual encoder position followed by 8 digits of target position (see chapter 7.) By repetitively doing a ctrl-U command while turning the encoder, the actual position will count up or down. If the encoder is set up for 1000 pulses/rev, moving it 1 revolution will show the actual position as 000003E8. If the encoder is then turned back 1 rev, the counts will go back to zero. If this does not happen, check the encoder connections. There is no sense in going any further until this problem is resolved.

3. Unplug P2, issue a RES command to zero the analog output, then plug P2 back on to the ACR1000 Board.
4. If the motor shaft stays still, the encoder is now in the proper phase. Skip to step 6.

5. If the motor now runs away, then the encoder is phased wrong. Interchange the channel A and channel A complement. These are on pins 1, 2 of plug P1. After this is done, the motor should stand still in a servo locked condition.

6. The motor and encoder are now phased properly and the gain and offset can now be set.

Motor Runaway On Motors Without A Tachometer

Before proceeding further, the first stage phasing explained previously must be accomplished. When this is done, the following steps should be taken.

1. With motor power OFF and P2 disconnected, enter a RES command to zero the analog output. Load P19 with 20, then reconnect P2. Turn motor power on. If the motor sits still or oscillates a little, the runaway problem is solved; skip to step 7. Else, continue to the next step.

2. With the first stage phasing complete and P2 plug still disconnected, turn power on and enter a RES command. This will result in the command voltage to be set to 0.0v then reconnect P2. If the motor sits still and is in the servo locked state, the second stage phasing is complete. If the motor runs away, continue to the next step.

3. Disable the servo amplifier such that the encoder shaft (or motor shaft if the encoder is mounted on the back of the motor) can be turned by hand with the ACR1000 still under power. Check if the encoder pulses are being recognized by the ACR1000 board:

   a. First make sure that parameter P2 is either 0, 1 or 2. If any different, it might disable encoder feedback.

   b. Issue a ctrl-U status command. This command will return 8 digits of actual encoder position followed by 8 digits of target position (see chapter 7.) By repetitively doing a ctrl-U command while turning the encoder, the actual position will count up or down. If the encoder is set up for 1000 pulses/rev, moving it 1 revolution will show the actual position as 00003BE8. If the encoder is then turned back 1 rev, the counts will go back to zero. If this does not happen, check the encoder connections. There is no sense in going any further until this problem is resolved.

4. Unplug P2, issue a RES command to zero the analog output, then plug P2 back on to the ACR1000 Board.

5. If the motor shaft stays still or oscillates back and forth, the encoder is in the proper phase and the system can now be fine tuned. The oscillations are probably due to P19 (Tach Gain) not being set properly. P19 should be increased to eliminate the oscillations. With everything stable, skip to step 7.

6. If the motor now runs away, then the encoder is phased wrong. Interchange the channel A and channel A complement. These are on PINS 1, 2 of PLUG P1. After this is done, the motor should stand still in a servo locked condition.

7. The motor and encoder are now phased properly and the gain and offset can now be set.

ACR1000/12 Gain/Offset Adjustment

There are two adjustment pots provided on the ACR1000 cards. One is a gain adjustment pot while the other is used to adjust the offset. The gain pot is used to adjust the overall feedback loop and software-tachometer gain. The offset pot is used to compensate for amplifier drift. These are set at the factory and should not need to be adjusted again, but in case a card needs to be readjusted, the procedure follows.

NOTE: The ACR1000/16 with v4.x Software has facility for software gain and offset adjustments. For this version, the offset and gain pots are factory adjusted. The gain adjustment can be done via the PID command.

To adjust offset and gain:
1. Disconnect plugs P1 and P2.
2. Turn on the servo amplifier and see if the motor drifts.

3. If there is excessive drift, adjust the servo amplifier’s offset for minimum drift.

   NOTE: If motor does not become almost stationary there is a problem with the servo amplifier.

4. Turn off the servo amplifier and reconnect P1 and P2.

5. Issue a RES command.

6. Connect a volt meter’s negative lead to test point #3 next to P2 on the ACR1000 card.

7. Connect the volt meter’s positive lead to test point #2 next to P2 on ACR1000 card.

8. Turn the offset pot all the way counter clockwise (up to 20 turns).

9. Adjust the gain pot until the volt meter displays +600mvdc.

10. Adjust the offset pot until the volt meter displays 0vdc.

11. Issue a MOV 4095 command.

12. Adjust the gain pot until volt meter displays -10vdc (±300mvdc).

13. Issue a MOV -4095 command. The volt meter should display +10vdc (±300mvdc).

14. Issue a RES command. The volt meter should display 0vdc (±300mvdc). If it does not, contact the factory for more instructions.

15. Turn on the servo amplifier.

16. Press ctrl-U and hold down to adjust offset pot until monitor displays all zeros. (For some communications software it may be necessary to repeatedly press ctrl-U instead of holding it down.)

   NOTE: The status information from a ctrl-U may vary ± one digit—this is normal.

Balancing The Gain And Offset Pots

When multiple cards will be interpolating together, it is necessary to have identical gains on all axes. This is necessary so that the dynamic characteristics of all the cards are the same. Failure to do so will result in, for example, flat spots while trying to interpolate a circle.

For example a system has 10000 pulses per inch (minimum resolution is 0.0001”). For overall gain of 1 inches per minute per thousandth:

1. Set the ACR1000 card gain such that with the motor disabled, when a 0.1 inch (1000 pulses) move is made, the analog output should show 4 Volts. This output will stay at 4 volts because the motor has been disabled. (Note that the motor can be disabled by simply unplugging the motor lead from the servo amplifier.)

2. After carefully making sure that both cards are set exactly the same, enable the motors. The gain pot on the ACR1000 card is now set. Do not touch this gain pot from now on. Now run each axis independently at 100 inches per minute while monitoring the analog signal to the servo amplifier. For a gain of 1, turn the gain pot on the servo amplifier until the analog reads 4 volts. For a gain of 2, this voltage should be at 2 volts. For a gain of 0.5, this voltage should be at 8 volts, etc.

3. It should be noted that in most cases, setting the gain can be done without having to go through the details explained above. For point to point applications it is sufficient to make the gain as high as possible to achieve fast response.

The offset pot should be set after the overall gain and tach gains have been set. To set the offset pot the following procedure should be followed.
Assuming that motor operation is now satisfactory (the motor starts and stops and positions properly):

1. Do a RES command in the immediate mode.

2. Next, do a ctrl-U status command that shows current position and actual shaft position.

3. The current position should be 00000000. The actual shaft position should also be 00000000 but might be off by a few pulses. While doing the ctrl-U command repetitively, and observing the actual position, turn the offset pot until the actual position becomes 00000000. Note that it is normal for the actual position to flicker a bit.

Tachless Operation

In some applications, there is no tachometer available on the motor. For such cases, the ACR1000 can generate the tach internally via software, and stabilize the system response. The main benefit is the cost savings of not having a tachometer on the motor. However, there is usually some degradation of the system response and performance. This is because the encoder is only used for position control but also velocity control. For applications that do not require high accuracy this is an acceptable way to go. The next section describes how to set the tach gain. If the motor has a tachometer, set P19 to 0 and skip this section entirely.

How To Set Software Tach Gain

Parameter P19 controls the Software Tach Gain (this feature not available in v2.7.x or v4.x). This parameter must be set with a value that will depend on the particulars of the mechanics and the motor/amplifier combination that the ACR1000 is trying to control. For instances where a tach is provided with the motor, this parameter must be set to 0. Typically, upon starting with an unknown system, begin with a P19 value of about 20 and increase or decrease it based on performance. The optimum value is such that the response is critically damped. Upon looking at the analog signal going to the amplifier, satisfactory operation will be achieved when the motor will start and stop without either undershooting or overshooting the target position. The following diagrams will illustrate.

![Correct Damping and Underdamped Signal](image)

**Software TACH adjustment**

With the analog signal disconnected from the servo amplifier, the motor shaft should be moving very slowly in one direction. This might be due to several reasons. First, with the analog command signal from the ACR1000 disconnected from the motor amplifier, it must be verified that the motor does not run away. If it does, then replace the motor amplifier. If the motor stands still with the signal disconnected but still runs away when connected, that indicates the encoder is phased backwards. To rectify this, simply change chA and chA complement wires from the encoder. These end up on PINS 1,2 on the encoder plug P1. If the motor does not run away but tends to oscillate badly, the solution is as follows:

Starting with a P19 value of about 5, observe the motor shaft oscillations. Increase P19 by 5 until the oscillations cease (the valid range of P19 is between 0 and 255). Then run a test program that indexes and delays in between the indexes.

```
10 ACC 0
20 DEC 0
30 STP 0
40 P20=0
50 VEL 3000
60 MOV /10000
```
The above program will index 10000 pulses. This distance should be changed to represent about 10 revolutions of the motor shaft. Note that the Feed-Forward parameter P20 is set to zero. Then observe the response of the motor. If an oscilloscope is available, the analog should be observed. If parameter P19 is too high or too low, or the ACR1000 gain is too high or too low, the motor will tend to oscillate when it stops. Once this parameter is set, it need not be changed.

**Setting the Feed-Forward parameter**

For some applications, it is desired to run a axis with little following error. By adjusting the system gain and the Feed-Forward parameter, this can be achieved. Before setting the Feed-Forward parameter, the gain must be established first. Make sure that the Feed-Forward parameter is zero at this time. If you wish you may use the "hunt-and-peck" method of finding the correct P20 value. To do this, jog the motor (under load) at a fairly high (but not atypical) speed. If the following error is too high, bring up P20, if it is less than zero, bring P20 down. If this is not possible, use the following method.

Make a trapezoidal index at the desired motor RPM. When the motor is at speed, note the following error by using the ctrl-U status. Set P20 according to the formula:

\[ P20 = \frac{(\text{Following Error} \times 128)}{\text{VELOCITY}} \]

For example, if the at-speed following error at a VEL of 2000 is noted to be 500 pulses then the Feed-Forward gain parameter should be loaded with

\[ 500 \times 128/2000 = 32 \]

Once this is set at a particular speed, the following error should track from minimum to maximum speed. Changing the load condition will affect the ideal P20 setting.

Note that making the Feed-Forward too large will cause the motor to over shoot the target destination if the ACC, DEC & STOP values are not set properly.

**2-Axis Control Using ACR1000/16 and v5.x**

v5.0 allows 2-axis control on the ACR1000/16/MS/AAO controller. The card must have the aux. D/A output and the aux. encoder input options. There are some differences that the user must be aware of to use this Version.

**Addressing Each Of Two Axes**

In this version, the CARD is still addressed like previous versions (0,1,2...A,B,C,D,E,F). The MAIN axis is now called "X" and the AUX axis is "Y"

Almost all the old commands will still pertain to the MAIN (X) axis. To specify the Y axis, simply use the prefix of "Y" before the pertinent command and the Y axis will see the command. For example VEL 1000 and XVEL 1000 will both set the velocity to 1000 for the X axis, while YVEL 1000 will set the velocity for the Y axis.

The following commands will cause both axes to start moving towards different destinations at different velocities and different ACC, STOP rates.

\[ \text{XVEL1000:YVEL2000:XACC10:XSTP10:YACC20:YSTP20} \]
\[ \text{XMOVE1000:YMOVE20000} \]

The following commands will support the prefix of X or Y to select the X or Y axis.

\[ \text{MOV VEL ACC DEC STP HOME IPB} \]
\[ \text{EXC INT JOG HDWON IPR FLZ} \]

**Re-Assignment Of Internal Relays**

Most all the internal relays are identical on the 1-Axis versions v2.6.x, v2.7.x, 3.x & 4.x. But in v5.x there are some differences. The new assignment for v5.x for these relays are as follows.

\[ 32/20h \]
\[ \text{Y Feed hld Pushed} \]
\[ \text{Energizing this will cause the Y axis to go into feed hold.} \]
<table>
<thead>
<tr>
<th>Relay Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>33/21h</td>
<td>Y +JOG pushed</td>
</tr>
<tr>
<td>34/22h</td>
<td>Y -JOG pushed</td>
</tr>
<tr>
<td>35/23h</td>
<td>Y Cycle Start Pushed</td>
</tr>
<tr>
<td>36/24h</td>
<td>Y Home Pushed</td>
</tr>
<tr>
<td>37/25h</td>
<td>Y In Accel</td>
</tr>
<tr>
<td>38/26h</td>
<td>Y In Decel</td>
</tr>
<tr>
<td>39/27h</td>
<td>Y Moving</td>
</tr>
<tr>
<td>40/28h</td>
<td>Y In Feed hid Cycle</td>
</tr>
<tr>
<td>41/29h</td>
<td>Y Feed hid Complete</td>
</tr>
<tr>
<td>42/2Ah</td>
<td>Y In JOG Mode</td>
</tr>
<tr>
<td>43/2Bh</td>
<td>Y Feed Direction</td>
</tr>
<tr>
<td>44/2Ch</td>
<td>Y Not In Position</td>
</tr>
<tr>
<td>45/2Dh</td>
<td>Y Not In Excess Error</td>
</tr>
<tr>
<td>46/2Eh</td>
<td>Y Cancel Move</td>
</tr>
</tbody>
</table>

Energizing this will cause Y to start jogging positive.
Energizing this will cause Y to start jogging negative.
Energizing this cancels a feed hold cycle.
Energizing this will cause Y to start homeing if X is not already in the home cycle.
This relay is energized during Y axis acceleration.
This relay is energized during Y axis deceleration.
This relay is energized during Y axis movement either via JOG or home or a YMOV command.
This relay is energized during Y axis feed hold cycle.
This relay is energized at the end of Y axis feed hold cycle.
This relay is energized while Y axis is jogging.
This relay is energized if Y axis is moving in the negative direction.
This relay is energized as long as the Y axis feedback is within the IPB (In Position Band) tolerance and the axis is not moving.
This relay is energized as long as the Y axis feedback is within the EXC (excess error) tolerance.
This relay can be set by the user to interrupt and cancel an ongoing Y axis move. This flag is modal and the user must detect that the Y moving relay is de-energized signalling that the move has been aborted and then clear the cancel Y move relay. If this is not done, all subsequent moves will be ignored.

The following relays are not to be used 47/2Fh - 55/37h.

Re-Assgnment Of Parameters

P30 used to be a SPARE general purpose parameter. It is now reassigned for the Y Floating Zero.

The WEB command is not available for version 5.x.

Home Parameters For X & Y

Both the MAIN (X) and the Y axes SHARE the same HOME parameters. This is not really a problem because in most cases the parameters are the same anyway. The exceptions are the INPUT number for the HOME SWITCH and perhaps the DIRECTION. So the problem is solved by assigning the common parameters once. Then before the individual XHOME or YHOME command, load the unique parameters with the proper value.

Cautions On Axes Interaction

Even though the X and Y axes are independent in the sense that they have their own ACC, DEC, STP, etc parameters, the board as a whole still has ONE interpreter. This limits the user to having ONE program to control both the axes.

For this reason the program flow must be carefully thought of at the beginning.

Several key points to consider.
1. Both axes have to be in either the run mode or the immediate mode.
2. On making indexes, care should be taken so as not to queue too many moves of the same axis before moving the other. Not being aware of this might result in the Y-axis for instance waiting for the queue to empty before it starts to move.

<table>
<thead>
<tr>
<th>Number</th>
<th>Command</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>XVEL</td>
<td>1000</td>
</tr>
<tr>
<td>20</td>
<td>YVEL</td>
<td>2000</td>
</tr>
<tr>
<td>30</td>
<td>XMOV</td>
<td>10000</td>
</tr>
<tr>
<td>40</td>
<td>YMOV</td>
<td>20000</td>
</tr>
<tr>
<td>50</td>
<td>XMOV</td>
<td>30000</td>
</tr>
<tr>
<td>60</td>
<td>YMOV</td>
<td>40000</td>
</tr>
<tr>
<td>70</td>
<td>XMOV</td>
<td>50000</td>
</tr>
<tr>
<td>80</td>
<td>XMOV</td>
<td>60000</td>
</tr>
<tr>
<td>90</td>
<td>XMOV</td>
<td>70000</td>
</tr>
</tbody>
</table>
In the above example, line 30 and 40 will get executed together thus resulting in both axes moving simultaneously. Line 50 will get queued and program control will wait there until the X axis reaches 10000. At this point X will start to move to 30000, then line 60 will get queued and control will wait until Y axis has reached 20000. At this time Y will start to move to 40000, then line 70 will get queued and program control will wait until X has reached 30000. At this time, X will start to move to 50000. The next three moves are all X axis moves. Even if by now the Y axis has reached 40000 pulse destination, line 100 will not start to get executed until X axis reaches 60000 pulses. This is a very workable method as long as the user is aware that there are limitations on how to move both the Axes.

3. One axis can be in hand wheel mode while the other is executing a MOVE command or is in JOG. Hand wheel mode is an option requiring encoder input daughter board.

4. Both the axes can be in the hand wheel mode with independent IPR ratios.

5. There are separate bit flags for In-Position, Move-En, Jogging, Excess_Error....etc. for both axes.

6. There are independent feed hold and cycle start modes for each axis, but there is no "status" command for the Y axis like there is for the X axis (ctrl-K). Likewise there is no Y axis Cycle-Start "status" command equivalent to the ctrl-C for the X axis.

7. Separate ACC/DEC and STOP parameters for each axes are provided.
Chapter 14 Expansion Options (Daughter Boards)

Expansion PLC Inputs and Outputs

v3.x and v5.x only.

One or two Expansion I/O boards may be added to an ACR1000/16 board. To read the Expansion I/O board(s), the firmware required is Version 3.24 (or later) or Version 5.10B (or later). In order to utilize the expansion boards, you will need to; use new values in the P10 parameter; use new arguments with the SET, CLR, INH, INT and IF commands and use new PLC commands.

P10 Values - Enabling the Expansion I/O Board

You must enable the expansion I/O boards by setting the P10 parameter to the correct values. Currently P10 defaults to 3 if no additional daughterboards are in use. Set P10 to 4 if you intend to use one expansion I/O board and 5 if you intend to use 2 expansion boards. If these values are changed after the boards are enabled, the inputs and outputs will remain in their last state.

SET, CLR, INH, INT and IF

The SET, CLR, INH, INT and IF commands are available for use with the expansion I/O boards. The relay numbers to access the addition I/O are:

<table>
<thead>
<tr>
<th>Board 1</th>
<th></th>
<th></th>
<th>Board 2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>Outputs</td>
<td>Inputs</td>
<td>Outputs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>address</td>
<td>name</td>
<td>address</td>
<td>name</td>
<td>address</td>
<td>name</td>
</tr>
<tr>
<td>128</td>
<td>input 0</td>
<td>140</td>
<td>output 0</td>
<td>152</td>
<td>input 0</td>
</tr>
<tr>
<td>129</td>
<td>input 1</td>
<td>141</td>
<td>output 1</td>
<td>153</td>
<td>input 1</td>
</tr>
<tr>
<td>130</td>
<td>input 2</td>
<td>142</td>
<td>output 2</td>
<td>154</td>
<td>input 2</td>
</tr>
<tr>
<td>131</td>
<td>input 3</td>
<td>143</td>
<td>output 3</td>
<td>155</td>
<td>input 3</td>
</tr>
<tr>
<td>132</td>
<td>input 4</td>
<td>144</td>
<td>output 4</td>
<td>156</td>
<td>input 4</td>
</tr>
<tr>
<td>133</td>
<td>input 5</td>
<td>145</td>
<td>output 5</td>
<td>157</td>
<td>input 5</td>
</tr>
<tr>
<td>134</td>
<td>input 6</td>
<td>146</td>
<td>output 6</td>
<td>158</td>
<td>input 6</td>
</tr>
<tr>
<td>135</td>
<td>input 7</td>
<td>147</td>
<td>output 7</td>
<td>159</td>
<td>input 7</td>
</tr>
<tr>
<td>136</td>
<td>input 8</td>
<td>148</td>
<td>output 8</td>
<td>160</td>
<td>input 8</td>
</tr>
<tr>
<td>137</td>
<td>input 9</td>
<td>149</td>
<td>output 9</td>
<td>161</td>
<td>input 9</td>
</tr>
<tr>
<td>138</td>
<td>input 10</td>
<td>150</td>
<td>output 10</td>
<td>162</td>
<td>input 10</td>
</tr>
<tr>
<td>139</td>
<td>input 11</td>
<td>151</td>
<td>output 11</td>
<td>163</td>
<td>input 11</td>
</tr>
</tbody>
</table>

For example: SET 165 sets output #1 on expansion I/O board #2.
INH 138 waits for input #10 on expansion I/O board #1.

PLC Commands

The new PLC commands to access expansion I/O are:

AND 90 46 XX E0 82 EY
AND NOT 90 46 XX E0 B0 EY
LOAD 90 46 XX E0 A2 EY
OR 90 46 XX E0 72 EY
OR NOT 90 46 XX E0 A0 EY
NOT B3 (SAME)
OUT 90 46 XX E0 92 EY F0
END 22 (SAME)

Where XX and Y are the address and bit reference of the desired relay. These can be determined from the following table:
Here is an example program:

```plaintext
POKE 4400 = 90, 46, 57, E0, A2, E3
POKE 4406 = 90, 46, 5A, E0, 82, E0
POKE 440C = 90, 46, 5B, E0, 92, E1, F0
POKE 4413 = 22

: REM LOAD board 1 input 3 (relay 131) = 57h bit ref 3
: REM AND board 2 input 8 (relay 160) = 5Ah bit ref 0
: REM OUT board 1 output 1 (relay 141) = 5Bh bit ref 1
: REM END of scan
```

New PLC commands can be used with old PLC commands to access the new and old I/O in the same PLC program. They can also be used with timers and counters.

**Expansion I/O Daughterboard Jumper Settings**

On the I/O daughter board, if the jumper on J2 is on the bottom two pins it's configured as Board 1. Otherwise if the jumper on J2 is on the top two pins it's configured as Board 2.

```
<table>
<thead>
<tr>
<th>J2 SET FOR BOARD 1</th>
<th>J2 SET FOR BOARD 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

The other selectors J3, J4 and J5 are reserved for future use and are not used at this time.

When installing the I/O Expansion board on the ACR1000 card please check the Expansion board spacing to assure that there is no shorting of the back of the board to any component on the ACR1000.
A/D Expansion Board Option

This option allows the signal from the main encoder or the auxiliary encoder (or both) to be replaced by an Analog Voltage signal. This added flexibility can be more convenient and has been used to support some very unique applications.

To use this board you must first determine the range of analog input voltages you will be using and set the jumpers for each channel accordingly.

A/D Expansion Board Jumper Settings

<table>
<thead>
<tr>
<th>CHANNEL #1</th>
<th>CHANNEL #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 to 10 Volts</td>
<td>1–2</td>
</tr>
<tr>
<td>-5 to +5 Volts</td>
<td>1–2</td>
</tr>
<tr>
<td>0 to 20 Volts</td>
<td>2–3</td>
</tr>
<tr>
<td>-10 to +10 Volts</td>
<td>2–3</td>
</tr>
</tbody>
</table>

Next you will need to connect your analog signal wires to the correct pins on JP6

<table>
<thead>
<tr>
<th>JP 6 Connector</th>
<th>pin# = Name of signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 = N/C</td>
<td>4 = N/C</td>
</tr>
<tr>
<td>1 = GND for #2</td>
<td>3 = input #2</td>
</tr>
</tbody>
</table>

If you already have the ribbon cable that converts JP6 to a D-type plug use the following pins on JP6D.

<table>
<thead>
<tr>
<th>JP 6 Converted to D-type</th>
<th>JP 6 D</th>
<th>pin# = Name of signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = GND for #2</td>
<td>2 = input #2</td>
<td>3 = input #1</td>
</tr>
<tr>
<td>6 = N/C</td>
<td>7 = N/C</td>
<td>8 = N/C</td>
</tr>
</tbody>
</table>

We recommend using twisted pair shielded cable for noise immunity. The system is sensitive to pick up 60 Hertz "hum" on a very small length of wire. The A/D is a full 12 bits with no filtering on board. Use external shielding and filtering as required.

Once the hardware is connected set the value of P2 to 4.5 or 6 to redirect the Main encoder to the A/D channel #1. Now the card will try to control the motor based on the values it recieves from the A/D card.

Set the value of P3 to 4.5 or 6 to redirect the Aux encoder to the A/D channel #2. Now the ACR-1000 will get the Handwheel values from the A/D instead of P9 the Aux encoder port.
Encoder Expansion Board Option

This option simply adds more encoder inputs to be used by the ACR-1000. On v5.x, J2 and J3 become the X-Hand wheel and Y-Hand wheel encoder ports. On v3.x and v5.x, RDPOS 3 and RDPOS 4 can be used to get the values available from J2 and J3.

To use this board, first determine what type of encoder you will be using. For TTL or Line Driver type encoders remove R1 for J2 or R2 for J3. For open collector types adjust J1 to select the proper voltage.

When you plug the daughter board in the P12 expansion connector you must also plug in the supplied jumper between J4 on the Daughter board and P7 the Master port on the ACR-1000 board.

### J1 Voltage Select Jumper

<table>
<thead>
<tr>
<th>pin# - pin#</th>
<th>selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 2</td>
<td>+12V</td>
</tr>
<tr>
<td>2 - 3</td>
<td>+5V</td>
</tr>
</tbody>
</table>

### J2 X-Axis Encoder Connector

<table>
<thead>
<tr>
<th>pin#</th>
<th>Name of signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Z Not</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>+5V</td>
</tr>
<tr>
<td>6</td>
<td>GND</td>
</tr>
<tr>
<td>8</td>
<td>N/C</td>
</tr>
<tr>
<td>10</td>
<td>N/C</td>
</tr>
</tbody>
</table>

### J3 Y-Axis Encoder Connector

<table>
<thead>
<tr>
<th>pin#</th>
<th>Name of signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Z Not</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>A Not</td>
</tr>
<tr>
<td>4</td>
<td>B Not</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
</tr>
<tr>
<td>7</td>
<td>B Not</td>
</tr>
<tr>
<td>9</td>
<td>Z</td>
</tr>
</tbody>
</table>

### J4 Acroloop Master/Slave Interface

<table>
<thead>
<tr>
<th>pin#</th>
<th>Name of signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Marker 1</td>
</tr>
<tr>
<td>1</td>
<td>GND</td>
</tr>
<tr>
<td>4</td>
<td>N/C</td>
</tr>
<tr>
<td>6</td>
<td>Marker 2</td>
</tr>
<tr>
<td>8</td>
<td>N/C</td>
</tr>
<tr>
<td>10</td>
<td>INT/CLK</td>
</tr>
<tr>
<td>3</td>
<td>GND</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
</tr>
<tr>
<td>7</td>
<td>GND</td>
</tr>
<tr>
<td>9</td>
<td>GND</td>
</tr>
</tbody>
</table>

If you have the ribbon cable that converts J2 or J3 to a D-type plug use the following pins on J2D or J3D.

### J2&J3 Converted to D-type Connector

<table>
<thead>
<tr>
<th>pin#</th>
<th>Name of signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>A Not</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
</tr>
<tr>
<td>4</td>
<td>B Not</td>
</tr>
<tr>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>6</td>
<td>Z Not</td>
</tr>
<tr>
<td>7</td>
<td>+5V</td>
</tr>
<tr>
<td>8</td>
<td>GND</td>
</tr>
<tr>
<td>9</td>
<td>N/C</td>
</tr>
</tbody>
</table>
### Appendix A: Acroloop ASCII Table

<table>
<thead>
<tr>
<th>DEC</th>
<th>HEX</th>
<th>symbol</th>
<th>Argument</th>
<th>Name/ACR-1000 function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00h</td>
<td>ctrl-@</td>
<td># or &lt;CR&gt;</td>
<td>Turn on card # or all cards</td>
</tr>
<tr>
<td>1</td>
<td>01h</td>
<td>ctrl-A</td>
<td></td>
<td>Turn off card # or all cards</td>
</tr>
<tr>
<td>2</td>
<td>02h</td>
<td>ctrl-B</td>
<td></td>
<td>Cycle Start command</td>
</tr>
<tr>
<td>3</td>
<td>03h</td>
<td>ctrl-C</td>
<td></td>
<td>Used by MOVE (see MOV command)</td>
</tr>
<tr>
<td>4</td>
<td>04h</td>
<td>ctrl-D</td>
<td></td>
<td>Feed and ramp values 2</td>
</tr>
<tr>
<td>5</td>
<td>05h</td>
<td>*ctrl-E</td>
<td></td>
<td>Forward Jog command</td>
</tr>
<tr>
<td>6</td>
<td>06h</td>
<td>ctrl-F</td>
<td></td>
<td>Used by FOV (v4.x)</td>
</tr>
<tr>
<td>7</td>
<td>07h</td>
<td>ctrl-G</td>
<td></td>
<td>&lt;backspace&gt;</td>
</tr>
<tr>
<td>8</td>
<td>08h</td>
<td>ctrl-H</td>
<td></td>
<td>&lt;tab&gt;</td>
</tr>
<tr>
<td>9</td>
<td>09h</td>
<td>ctrl-I</td>
<td></td>
<td>&lt;linefeed&gt;</td>
</tr>
<tr>
<td>10</td>
<td>0Ah</td>
<td>ctrl-J</td>
<td></td>
<td>Feed hld command</td>
</tr>
<tr>
<td>11</td>
<td>0Bh</td>
<td>ctrl-K</td>
<td></td>
<td>Move count (v4.x)</td>
</tr>
<tr>
<td>12</td>
<td>0Ch</td>
<td>ctrl-L</td>
<td></td>
<td>&lt;CR&gt;</td>
</tr>
<tr>
<td>13</td>
<td>0Dh</td>
<td>ctrl-M</td>
<td></td>
<td>Inputs 0 11.8 7.4 3.0</td>
</tr>
<tr>
<td>14</td>
<td>0 Eh</td>
<td>ctrl-N</td>
<td></td>
<td>Outputs 0 23.20 19.16 15.12</td>
</tr>
<tr>
<td>15</td>
<td>0Fh</td>
<td>ctrl-O</td>
<td></td>
<td>Following error</td>
</tr>
<tr>
<td>16</td>
<td>10h</td>
<td>*ctrl-P</td>
<td></td>
<td>XON</td>
</tr>
<tr>
<td>17</td>
<td>11h</td>
<td>ctrl-Q</td>
<td></td>
<td>Reverse Jog command</td>
</tr>
<tr>
<td>18</td>
<td>12h</td>
<td>ctrl-R</td>
<td></td>
<td>XOFF</td>
</tr>
<tr>
<td>19</td>
<td>13h</td>
<td>ctrl-S</td>
<td></td>
<td>Commanded Position TTTTTTTTTh</td>
</tr>
<tr>
<td>20</td>
<td>14h</td>
<td>*ctrl-T</td>
<td></td>
<td>Actual &amp; Target Position AAAAAAAATTTTTTTTTh</td>
</tr>
<tr>
<td>21</td>
<td>15h</td>
<td>*ctrl-U</td>
<td></td>
<td>Status bits 1</td>
</tr>
<tr>
<td>22</td>
<td>16h</td>
<td>ctrl-V</td>
<td></td>
<td>Status bits 2</td>
</tr>
<tr>
<td>23</td>
<td>17h</td>
<td>*ctrl-W</td>
<td></td>
<td>Current program line number</td>
</tr>
<tr>
<td>24</td>
<td>18h</td>
<td>ctrl-X</td>
<td></td>
<td>(reserved)</td>
</tr>
<tr>
<td>25</td>
<td>19h</td>
<td>*ctrl-Y</td>
<td></td>
<td>&lt;esc&gt;</td>
</tr>
<tr>
<td>26</td>
<td>1Ah</td>
<td>ctrl-Z</td>
<td></td>
<td>Feed and ramp values 1</td>
</tr>
<tr>
<td>27</td>
<td>1Bh</td>
<td>ctrl-[</td>
<td></td>
<td>SET bit corresponding to ASCII value of char</td>
</tr>
<tr>
<td>28</td>
<td>1Ch</td>
<td>ctrl-\</td>
<td>(char)</td>
<td>CLR bit corresponding to ASCII value of char</td>
</tr>
<tr>
<td>29</td>
<td>1Dh</td>
<td>ctrl-]</td>
<td>(char)</td>
<td>Print system parameter value.</td>
</tr>
<tr>
<td>30</td>
<td>1 Eh</td>
<td>ctrl-A</td>
<td>(char)</td>
<td>Y-axis status request (v5.x)</td>
</tr>
<tr>
<td>31</td>
<td>1Fh</td>
<td>ctrl-\</td>
<td>(any *)</td>
<td></td>
</tr>
</tbody>
</table>

Appendix A: Acroloop ASCII Table 95
<table>
<thead>
<tr>
<th>DEC</th>
<th>HEX</th>
<th>symbol</th>
<th>notes</th>
<th>DEC</th>
<th>HEX</th>
<th>symbol</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>20h</td>
<td>&quot;</td>
<td>(space)</td>
<td>80</td>
<td>50h</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>21h</td>
<td>!</td>
<td></td>
<td>81</td>
<td>51h</td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>22h</td>
<td>&quot;</td>
<td></td>
<td>82</td>
<td>52h</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>23h</td>
<td>#</td>
<td></td>
<td>83</td>
<td>53h</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>24h</td>
<td>$</td>
<td></td>
<td>84</td>
<td>54h</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>25h</td>
<td>%</td>
<td></td>
<td>85</td>
<td>55h</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>26h</td>
<td>&amp;</td>
<td></td>
<td>86</td>
<td>56h</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>27h</td>
<td>'</td>
<td>single-quote</td>
<td>87</td>
<td>57h</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>28h</td>
<td>(</td>
<td></td>
<td>88</td>
<td>58h</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>29h</td>
<td>)</td>
<td></td>
<td>89</td>
<td>59h</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>2Ah</td>
<td>*</td>
<td></td>
<td>90</td>
<td>5Ah</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>2Bh</td>
<td>+</td>
<td></td>
<td>91</td>
<td>5Bh</td>
<td>[</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>2Ch</td>
<td>,</td>
<td>comma</td>
<td>92</td>
<td>5Ch</td>
<td>\</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>2Dh</td>
<td>-</td>
<td>dash</td>
<td>93</td>
<td>5Dh</td>
<td>]</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>2Eh</td>
<td>.</td>
<td></td>
<td>94</td>
<td>5Eh</td>
<td>^</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>2Fh</td>
<td>/</td>
<td></td>
<td>95</td>
<td>5Fh</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>30h</td>
<td>0</td>
<td></td>
<td>96</td>
<td>60h</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>31h</td>
<td>1</td>
<td></td>
<td>97</td>
<td>61h</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>32h</td>
<td>2</td>
<td></td>
<td>98</td>
<td>62h</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>33h</td>
<td>3</td>
<td></td>
<td>99</td>
<td>63h</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>34h</td>
<td>4</td>
<td></td>
<td>100</td>
<td>64h</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>35h</td>
<td>5</td>
<td></td>
<td>101</td>
<td>65h</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>36h</td>
<td>6</td>
<td></td>
<td>102</td>
<td>66h</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>37h</td>
<td>7</td>
<td></td>
<td>103</td>
<td>67h</td>
<td>g</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>38h</td>
<td>8</td>
<td></td>
<td>104</td>
<td>68h</td>
<td>h</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>39h</td>
<td>9</td>
<td></td>
<td>105</td>
<td>69h</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>3Ah</td>
<td>:</td>
<td></td>
<td>106</td>
<td>6Ah</td>
<td>j</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>3Bh</td>
<td>;</td>
<td></td>
<td>107</td>
<td>6Bh</td>
<td>k</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>3Ch</td>
<td>&lt;</td>
<td></td>
<td>108</td>
<td>6Ch</td>
<td>l</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>3Dh</td>
<td>=</td>
<td>P1 for ctrl-^</td>
<td>109</td>
<td>6Dh</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>3 Eh</td>
<td>&gt;</td>
<td>P2</td>
<td>110</td>
<td>6 Eh</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>3Fh</td>
<td>?</td>
<td>P3</td>
<td>111</td>
<td>6Fh</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>40h</td>
<td>@</td>
<td>P4</td>
<td>112</td>
<td>70h</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>41h</td>
<td>A</td>
<td>P5</td>
<td>113</td>
<td>71h</td>
<td>q</td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>42h</td>
<td>B</td>
<td>P6</td>
<td>114</td>
<td>72h</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>43h</td>
<td>C</td>
<td>P7</td>
<td>115</td>
<td>73h</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>44h</td>
<td>D</td>
<td>P8</td>
<td>116</td>
<td>74h</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>45h</td>
<td>E</td>
<td>P9</td>
<td>117</td>
<td>75h</td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>46h</td>
<td>F</td>
<td>P10</td>
<td>118</td>
<td>76h</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>47h</td>
<td>G</td>
<td>P11</td>
<td>119</td>
<td>77h</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>48h</td>
<td>H</td>
<td>P12</td>
<td>120</td>
<td>78h</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>49h</td>
<td>I</td>
<td>P13</td>
<td>121</td>
<td>79h</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>4Ah</td>
<td>J</td>
<td>P14</td>
<td>122</td>
<td>7Ah</td>
<td>z</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>4Bh</td>
<td>K</td>
<td>P15</td>
<td>123</td>
<td>7Bh</td>
<td>{</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>4Ch</td>
<td>L</td>
<td>P16</td>
<td>124</td>
<td>7Ch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>4Dh</td>
<td>M</td>
<td>P17</td>
<td>125</td>
<td>7Dh</td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>4 Eh</td>
<td>N</td>
<td>P18</td>
<td>126</td>
<td>7 Eh</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>4Fh</td>
<td>O</td>
<td>P19</td>
<td>127</td>
<td>7Fh</td>
<td>&lt;bs&gt;</td>
<td></td>
</tr>
</tbody>
</table>
## INDEX

### A
- ACC, 27, 75, 79, 82, 86, 87, 88, 89
- ADD, 24, 27, 41, 76, 77, 81
- AND, 28
- AOUT, 27, 28
- array, 24, 36, 62
- ASCII, 58, 59, 81, 95
- AT, 1, 3, 4, 10, 17, 18
- AUT, 27, 28

### B
- Ballscrew, 28
- Battery, 4
- baud, 3, 8, 13, 15, 82
- binary, 37, 57, 58
- BLK, 27, 28, 68
- BSC, 27, 28, 29, 32, 44

### C
- CAM, 29, 30, 32, 38, 44
- CAPTURE, 31, 75, 76
- CHKJMP, 27, 32, 36, 47
- CIRCLE, 27, 32, 54, 55, 56, 57
- circular interpolation, 2, 53
- CLEAR, 27, 30, 32
- CLR, 23, 27, 31, 32, 50, 62, 64, 65, 67, 68, 69, 76, 77, 80, 95
- COS, 27, 32, 54
- counters, 34, 40, 49, 63, 65, 66, 68
- ctrl, 17, 24, 27, 28, 33, 37

### D
- D/A, 3, 51, 87
- DELREM, 27, 36, 47
- DIM, 24, 32, 36
- DIV, 27, 36, 77
- download, 15
- DWL, 36, 40, 65, 76, 87

### E
- echo, 13, 14, 17, 22, 58
- elliptical, 1, 54, 57
- encoder, 1, 2, 3, 8, 10, 20, 22, 23, 29, 30, 31, 38, 39, 41, 48, 75, 76, 83, 84, 86, 87, 89
- END, 27, 36, 42, 70, 71
- ESC, 33, 36, 41, 43, 68
- EXC, 27, 36, 69, 87, 88
- Excess Error, 36, 88

### F
- Fast Move, 58
- FED, 27, 37
- feed forward, 24, 45
- Floating Zero, 24, 37, 88
- FLZ, 24, 27, 30, 37, 87
- FOV, 37
- FTB, 27, 37, 38

### G
- gain pot, 46, 84, 85
- GOTO, 32, 36, 38, 40, 42, 61, 62, 76, 77, 80, 87
- GSB, 27, 32, 36, 38, 62, 80
H
HDWOFF, 27, 38, 49
HDWON, 18, 27, 38, 43, 75, 78
HELP, 27, 38
hexadecimal, 37, 46, 58, 59, 62, 70, 71
HOME, 21, 24, 27, 39, 40, 48, 68, 79, 80, 82, 87, 88
I
IF, 27, 40, 42, 62, 64, 65, 66, 67, 69, 76, 77, 80
IHPOS, 41
INH, 27, 40, 41, 44, 64, 67, 68, 69
INP, 41, 81
INPUT, 42
INT, 27, 40, 42, 67, 68, 87
interpolation, 1, 2, 8, 9, 21, 24, 50, 53
IPB, 27, 43, 69, 87, 88
IPR, 22, 27, 37, 38, 43, 75, 76, 77, 87, 89
J
JOG, 23, 33, 34, 41, 43, 68, 79, 87, 88, 89
L
LIST, 15, 27, 43, 47
M
MEM, 24, 27, 36, 43
MOVE, 4, 24, 27, 33, 37, 42, 43, 44, 50, 58, 61, 62, 75, 79, 80, 89, 95
MOVE RELAY, 34
MOVE-INC, 34
MUL, 27, 44, 77
multi-axis, 1, 9, 33, 53, 82
N
NEW, 15, 27, 44, 47
NOT, 44
O
offset, 27, 28, 30, 31, 37, 38, 43, 45, 49, 76, 77, 84, 85
offset pots, 46, 83, 85
OPS1000, 1
OPS8000, 1
OPS9000, 1
OR, 44
P
PBOOT, 27, 44, 47, 61, 76, 80
PC BUSS, 2, 8, 10, 13, 37, 57
PEEK, 45, 59, 70, 71
PID, 45
PLC, 2, 3, 4, 32, 33, 34, 39, 41, 45, 46, 50, 63
PLUG INTERFACE, 7
POF, 46
POKE, 27, 46, 59, 63, 64, 65, 67, 70, 71
PON, 27, 40, 46, 63, 64, 69, 70
PRINT, 22, 27, 41, 42, 46, 62, 64, 65
PROM, 27, 47
INDEX

R
RDFE, 47
RDPOS, 27, 41, 48, 77
REM, 27, 32, 36, 48, 61
REN, 2, 23, 27, 29, 48
REN 1, REN 2, 48
RES, 2, 23, 27, 29, 31, 48, 83, 84, 85, 86
RES 1, RES 2, RES 3, RES 4, 48
RET, 38, 40, 49, 62
RNM, 27, 49
RS232, 1, 8, 13
RUN, 27, 34, 49, 61, 68, 80

S
SAMP, 49
SET, 14, 23, 31, 32, 41, 50, 62, 65, 67, 68, 69, 76, 77, 80
SIN, 27, 50, 54
Software Tach, 21, 23, 86
SQRT, 27, 50
STP, 27, 35, 39, 40, 44, 50, 51, 53, 56, 62, 75, 79, 81, 82, 87, 88
SUB, 27, 51, 77
System Parameters, 2, 21, 22

T
Tach, 83, 84
TAN, 51
Thumbwheel, 1, 80
Timers, 34, 64, 67
TLM, 27, 31, 51
TROFF, 27, 51
TRON, 27, 51
Turbo C, 18

U
Unidirectional Approach, 23
upload, 15
User Relay, 47
user relay, 34, 35, 62

V
VEL, 27, 37, 39, 44, 51, 57, 75, 80, 81, 82, 87

W
Watchdog, 4, 8
WEB, 2, 21, 75, 76, 88

X
XOR, 27, 51