KOLLMORGEN®

Because Motion Matters™

This is a Discontinued Product

Contact Kollmorgen Customer Support at 1-540-633-3545 or email us at support.kollmorgen.com if assistance is required.

AMERICAN PRECISION INDUSTRIES

ICL Users Guide

Doc. #151-ICL Rev B

Intelli-Command Language

Software Reference

Controls Division

45 Hazelwood Drive
Amherst, New York 14228
(716) 691-9100 • Fax (716) 691-9181 • (800) 566-5274

Revision B
P/N 151-ICL
March 1996

TABLE OF CONTENTS

INTRODUCTION

This Intelli-Command Language (ICL) System User Guide is intended to be a software and applications guide for all indexers utilizing the Intelli-Command Language System. **Since the ICL System is operated on different indexers, all functions might not be available on the indexer that you may be using. Please refer to your indexer user guide for a list of exceptions on your device.**

Fundamentally, an indexer is a computer that is dedicated to motion control. Like every other computer it has its own operating system (ICL), data storage capabilities, data manipulation capabilities (mathematics, flags, logical operators and string variables) and interface for data communications. In addition, its built-in Inputs/Outputs allow for hard-wired connections to sensor switches to ensure motion that is "In-Sync" with a user's environment. All ICL Systems provide the user with the same basic components:

1) System software including commands, parameters and user variables that allow you to enter and manipulate data to perform motion in either immediate or stored program modes.

- Immediate mode refers to a method of control whereby you are commanding the indexer from a host processor or a dumb terminal via the communication port one command at a time.
- Stored program mode refers to a method of control where you create and store "motion programs" on your ICL system and execute the programs when required. Motion programs provide the user with the ability to perform repetitive functions without having to type each individual command line every time you wish to perform the same function. In this mode you may choose to initiate a program through your communications port or establish a monitor program to allow an input to "trigger" or begin execution of a particular sequence of commands (program).

2) Hardware interfaces for Travel Limits, Home, and User Programmable Inputs and Outputs that allow you to interface to your machinery/equipment.

3) Control signals to your motor drive. Note that since integrated indexer/drive units are connected internally, external connections will not be visible.

4) Communications interface to your host computer or terminal for programming or commanding the indexer.

5) Extended functions on some units include math capabilities for manipulating data, encoder feedback and interrupt inputs. Please refer to your indexer user guide to determine if your device has some or all of these extended functions.

At this time you might want to make mental note of an underlying design concept of an ICL System:

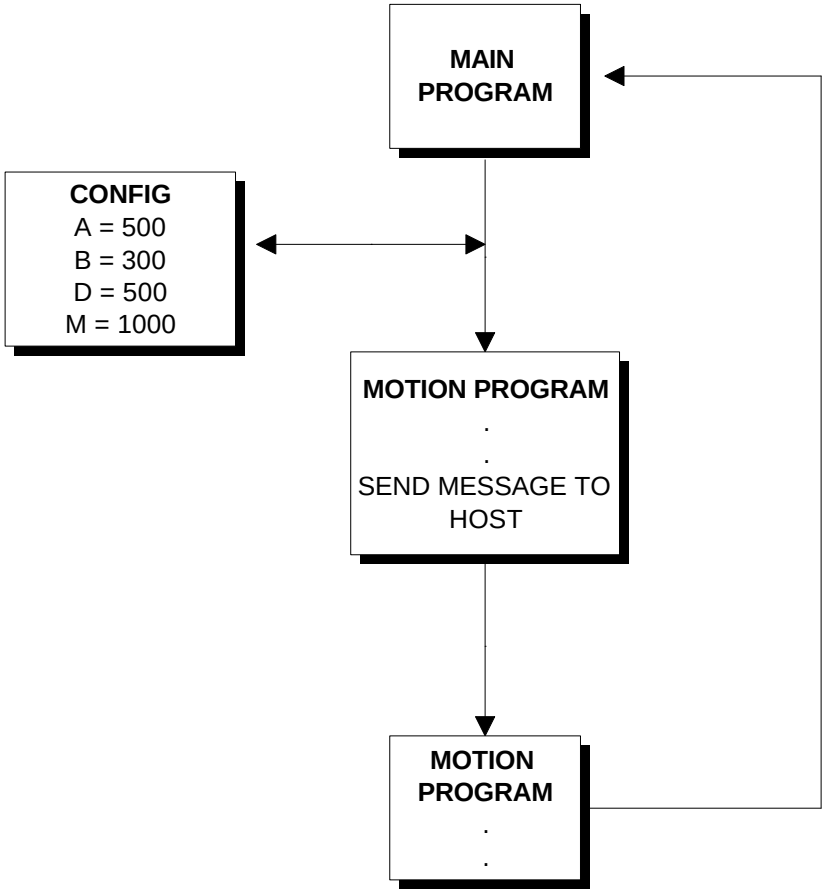
- ICL Commands tell the Indexer **"WHAT TO DO".**
- ICL Parameters and Variables tell the Indexer **"HOW TO DO IT."**

The Intelli-Command Language (ICL) is divided into three main categories: **ICL Parameters, ICL Variables and ICL Commands.** Each of these will be discussed in detail within the appropriate sections.

It should be noted that all parameters and commands may not be available for a specific indexer. Example: Since the **SAC-560** has six (6) programmable inputs while the **P315X** has thirteen (13), some parameters and commands may apply to the **P315X** but not to the **SAC-560**.

This ICL-System User Guide also contains a section on applications titled "Performing Motion". Where possible examples of actual ICL Programs and applications are used.

Six categories of additional suggestions are used throughout the manual in the form of a screened box with a drop shadow. The categories are: **Debugging Tool; Hardware Configuration; Hazard Note; Important Information; Programming Hints; and Software Bug.**



ICL PARAMETERS AND VARIABLES (GENERAL)

ICL Parameters and Variables contain user defined values to provide application specific information and define how the indexer will interface to its environment. Within an ICL System, Parameters and Variables tell the system “HOW TO DO IT” (i.e. how to perform motion or interface to your environment). Parameters and Variables are under user control and may be changed as required to meet the needs of the application. Note, that under certain conditions, you will not be allowed to change a specific parameter value. Those conditions are noted within the parameter definitions.

To simplify the use of this User Guide the section on Parameters and Variables is divided into General and Advanced Parameters. The section on ICL Parameters (Advanced) may be skipped if the you are not using an Encoder or Inputs/Outputs.

The ICL Parameters and Variables can be changed in program mode or in the immediate mode through use of the **p = *nn*** or **ENTER p *nn*** commands. The name of the parameter/variable is represented by **p** and its new value is represented by **nn**. Parameters and variables are maintained in nonvolatile battery-backed memory and may be changed as frequently as required by the user.

The **STATUS**, **SHOW** and **VERIFY** commands are used to display the current values of ICL Parameters and Variables. The **STATUS** command sends the table of all parameters to the host device. The **SHOW** command will display the values of all variables not cleared. The **VERIFY p** command is used to review the current value of one parameter or variable, where **p** is the name of the parameter or variable to be reviewed. When executed, the current value of that parameter or variable will be returned to the host device.

The listing below divides related parameters into categories to assist the user in identifying functionality.

ICL Parameters & Variables (General)

General Parameters and Motion Related Parameters

COMMAND	*DESCRIPTION*	*COMMAND*	*DESCRIPTION*
A	Acceleration Time	**R**	Reduce Current Time
B	Base Speed	**S**	Shutdown Current Time
C	Checksum	**SC**	Screen Characters
D	Deceleration Time	**T**	# of Lines to Transmit
F	Free Space	**U**	Unit Axis Designator
G	Auto-Start Program #	**UA**	Units Active
H	Highest Speed Limit	**UR**	User Resolution
J	Jog Speed	**W**	Current Program #
M	Maximum Velocity	**X**	Current Program Instruction Address
MR	Motor Resolution	**Y**	Loop Counter
N	Number of Pulses During Last Index	**Z**	Current Program Line Number
Q	Power-Up Time		

ICL Parameters (Advanced)

Encoder Related Parameters

COMMAND	*DESCRIPTION*
DW	Dead Band Window "Encoder"
CG	Position Correction Gain
CV	Correction Velocity
E	Encoder Position in Pulses
EF	Encoder Functions
ER	Encoder Resolution
ME	Motor Error
MR	Motor Resolution
P	Position of Motor in Pulses
PE	Maximum Position Error
SP	Stall Detect Program #

Input/Output Related Parameters

COMMAND	*DESCRIPTION*
HI	Hold Input #
I[n]	Status of Inputs
ID[n]	Input Definition
JM	Jog Minus Input #
JP	Jog Plus Input #
LA	Limit Action
O[n]	Status of Outputs
OD[n]	Output Definition
OF	Other Functions
SI	Stop Input #
SK	Soft Key

Parameter Definitions

Command Name or Parameter

Description Description of ICL Parameter or Command.

Format The Sample format for the ICL Parameter or Command.

<>	Characters enclosed in angle brackets designate keys or key strokes that can be found on the host keyboard.
[ ]	Optional characters are enclosed in brackets.
b	Bite
c	Denotes alphanumeric characters. Note labels must begin with ALPHA characters. Excluded characters are {<space>, <colon>, <comma> and <!>}.
f	Denotes an ICL flag name (F0 to F64).
fff	Denotes file name
n	Denotes numeric characters
p	Denotes an ICL parameter name
v	Denotes an ICL variable name (V0 to V99).
,	Command delimiter, has the same effect as a <space>. Each command must be separated from the next by a delimiter.

Mode

Immediate	Command is executed or changed in the immediate mode.
Read Only	Parameter is a Read Only
Stored	Command may be executed or changed within a program.

Range Denotes the minimum and maximum values allowed within a command or parameter.

Related Functions Listing of related parameters and commands.

Example Sample ICL command with comments.

```
A=1000 ^ Sets acceleration to 1 second
```

Sample Program

Example of ICL code.

Comments in program. Shown in lower case within this manual for purposes of readability.

The "Caret" denotes the start of a comment line. Note the "Caret" is followed by a blank space then the comment string.

> ***Programming Hint***
> The "!" (exclamations point) and ":" (colon) are reserved ASCII characters and must not/cannot be used in comments.

ACCELERATION TIME A

Description Acceleration time in milliseconds. This represents the maximum allowable time to accelerate from base speed **B** to highest speed limit **H**. This parameter is used to calculate the nonlinear acceleration profile utilized by the indexer.

Format **A** = *nn*

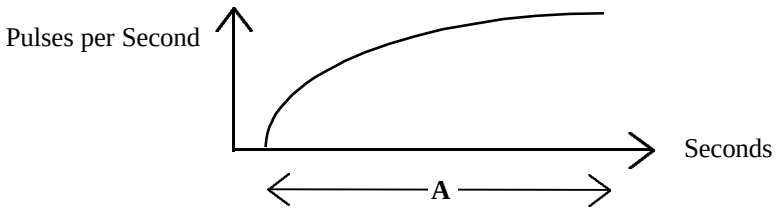
Mode Immediate/Stored

Range 400 ms minimum, 10,000 ms maximum values

Related Functions **B, D, H, M, GO, HOME, MOVE, POS, RUN, SLEW, TIME**

Example
```
A=1000 ^ Sets acceleration to 1 second
```

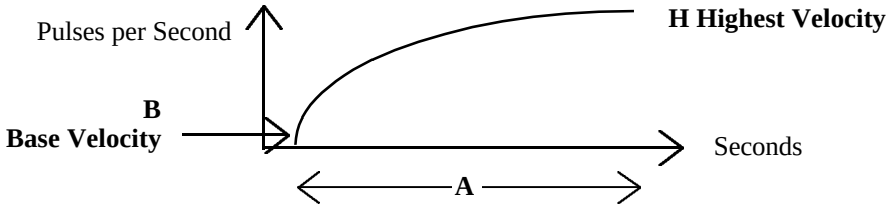
Sample Program



Programming Hints

Select an acceleration time as low as possible for your application. If the acceleration time is too long, the motor may become unstable while accelerating through its resonance velocities. A properly sized system should be able to accelerate to a speed of 20 revolutions per second in under one second, A=1000.

Below is a graphic showing the relationships between the motion parameters. These motion parameters are used to calculate the Optimal Non-Linear acceleration ramp tables utilized in the ICL indexer. The Optimal Non-Linear acceleration technique will reduce the acceleration time by 30-60% over linear acceleration techniques.



BASE SPEED B

Description The base speed is defined in motor pulses per second. The value of **B** should be selected low enough that the motor will stop and reverse direction instantaneously without missing steps. A low value of **B** will also reduce the damping time that occurs at the end of motion. This parameter is sometimes referred to as the error-free start-stop speed of the motor.

Format **B** = *nn*

Mode Immediate/Stored

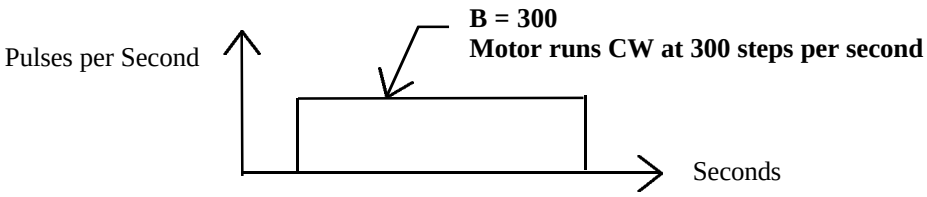
Range **B** is between 1 and **M** maximum velocity in motor pulses per second

Related Functions **B, D, H, M, GO, HOME, MOVE, POS, RUN, SLEW, TIME**

Example

```
B = 300    ^ Sets start speed at 300 pulses per second
B = MR / 5 ^ Set speed to 1/5 revolution per second
```

Sample Program



ICL COMMAND; + Run STOP

Programming Hints

Select a base velocity that is in the range of 1/5 revolution per second. If the selected base velocity is too high then there is not enough torque to allow for Optimal acceleration, (that is, all motor torque is used to start motion with little left to accelerate).

Select a base velocity and command the motor to RUN.

Change the direction of the motion by alternately typing a plus or minus sign to change the direction.

Use the <ESC> key to stop motion.

The motor must be able to instantly change direction without loss of synchronism.

Software Bug

Values of B less than 260 will result in erratic motion.

DECELERATION TIME D

Description Deceleration time in milliseconds. This represents the maximum allowable time to decelerate from highest motor speed limit **H** to base speed **B**. This parameter is used to calculate the nonlinear deceleration profile utilized by the indexer.

Format **D** = *nn*

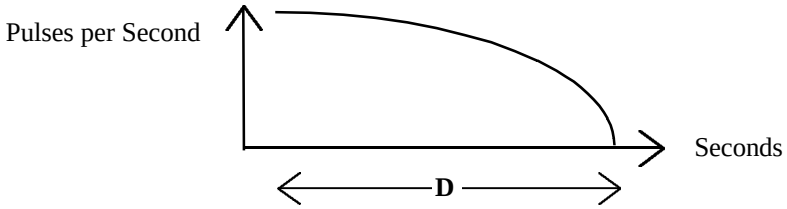
Mode Immediate/Stored

Range 400 ms minimum, 10,000 ms maximum values

Related Functions **B, D, H, M, GO, HOME, MOVE, POS, RUN, SLEW, TIME**

Example
```
D =2500 ^ Decelerate in 2.5 seconds or less
```

Sample Program



Programming Hints

Select a deceleration time as low as possible for your application. If the deceleration time is too long, the motor may become unstable while decelerating through its resonance velocities. A properly sized system should decelerate to a zero speed in less time than it took to accelerate to that speed, (i.e. if A = 500 then D can be less than 500).

Vertical Applications

For vertical applications the value of acceleration time must equal deceleration time. The motor must accelerate and decelerate in both the upward and downward motions, (i.e. A=1000, D=A).

FREE SPACE F

Description This parameter cannot be changed by the user. The user may verify the free space remaining in RAM in bytes.

Format **F** = *nn*

Mode Read only Immediate/Stored

Range Read only

Related Functions

Example

```
VERIFY F
F = 20152     (Tells the user 20,152 bytes of storage is still available)
```

Sample Program

AUTOSTART PROGRAM # G

Description Auto-start program number 0 to 87. The program number may be determined by obtaining a list of program numbers with the **DIRECTORY** command. If **G** = -1 is entered then no auto-start program is executed on power-up.

Format **G** = *nn*

Mode Immediate/Stored

Range -1 to 87

Related Functions **AUTOEXECUTE, DIRECTORY**

Example

```
The selected auto-start program is number 3
G = 3 <Return>
```

The next time the indexer is powered-up it will begin execution of the program listed as number 3 on the directory. The user may also execute the **AUTOSTART** command to begin execution of this program without a power-down and power-up cycle.

> ***Hazard Note***
> Do NOT assign an autoexecute program until you have fully debugged your programs. This will prevent you from accidentally creating an endless-loop routine and locking the controller. A badly corrupted operating system may require factory service.

HIGHEST SPEED LIMIT H

Description Highest speed limit in motor pulses per second, 750,000 pulses per second maximum value for microstep drives and 80,000 for full/half step or indexer dependent drives. Maximum velocity **M** cannot exceed this number. The highest speed parameter is used to calculate the acceleration and deceleration motion profiles.

When a stand alone API indexer is utilized to control a step motor drive, the user must make sure that the pulse width is large enough so that the selected drive will receive it. Consult the manufacturer's specifications on the selected drive for the required pulse width in microseconds. The pulse width for step is controlled by the following formula:

$$\text{WIDTH(microseconds)} = \frac{\text{MIN}\{255, \text{INTEGER}(4{,}000{,}000/\mathbf{H})\}}{8}$$

Format **H** = *nn*

Mode Immediate/Stored

Range **M** ≤ **H** ≤ value calculated from the above formula. (indexer dependent)

Related Functions **A, B, D, M, GO, HOME, MOVE, POS, RUN, SLEW, TIME**

Example

```
H=50000
^ Set the step pulse width for 10 micro seconds.

H = MR * 25 ^ Set Highest Velocity to 25 RPS
```

Sample Program

JOG SPEED J

Description Jog rate in motor pulses per second. This pulse rate is generated when **JP** or **JM** inputs are "ACTIVE". No acceleration occurs during jog. The current position parameter P is continuously updated whenever the JOG inputs are "ACTIVE".

Format **J** = *nn*

Mode Immediate/Stored

Range 1 ≤ **J** ≤ 3 rev per second pulse rate. (1 pulse per second to 3 rev per second).

Related Functions **JP, JM,** See also **Extended ASCII Commands**

Example Set the jog speed to 1000 motor pulses per second.

```
J=1000
```

Sample Program

```
J = MR/2
^ Set the jog velocity to 1/2 revolution per second

JP = 9
^ Activate input 9 to jog CW at 1/2 RPS

JM = 10
^ Activate input 10 to jog CCW at 1/2 RPS
```

JOG MINUS INPUT # JM

Description This parameter allows the user to select which programmable input is to be used as the **J**og **M**inus input. Set **JM = 0** to disable this feature. The jog speed is selected by parameter **J**.

Format **JM** = *nn*

Mode Immediate/Stored

Range 0 to 13 Indexer Dependent

Related Functions **J, JP,** See also **Extended ASCII Commands**

Example
```
To assign input #2 as jog minus,
JM=2 command is issued.
```

Sample Program
```
J = MR/2
^ Set the jog velocity to 1/2 revolution per second

JP = 9
^ Activate input 9 to jog CW at 1/2 RPS

JM = 10
^ Activate input 10 to jog CCW at 1/2 RPS
```

JOG PLUS INPUT # JP

Description This parameter allows the user to select which programmable input is to be used as the **J**og **P**lus input. Set **JP = 0** to disable this feature. The jog speed is selected by parameter **J**.

Format **JP** = *nn*

Mode Immediate/Stored

Range 0 to 13 Indexer Dependent

Related Functions **J, JM,** See also **Extended ASCII Commands**

Example
```
To assign input #3 as jog plus,
JP=3 command is issued.
```

Sample Program
```
J = MR/2
^ Set the jog velocity to 1/2 revolution per second

JP = 9
^ Activate input 9 to jog CW at 1/2 RPS

JM = 10
^ Activate input 10 to jog CCW at 1/2 RPS
```

MAXIMUM VELOCITY M

Description Maximum velocity in motor pulses per second. Minimum value must be greater than base speed **B** and cannot exceed highest speed limit **H**. The value of **M** is the maximum velocity that the user will allow his system to achieve.

Format **M** = *nn*

Mode Immediate/Stored

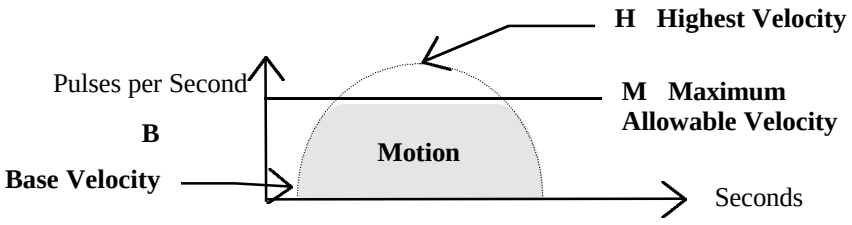
Range **B ≤ M ≤ H**

Related Functions **A, B, D, H, GO, HOME, MOVE, POS, RUN, SLEW, TIME**

Example
```
M=40000 ^ Set max. velocity to 40000 steps per second
```

Sample Program If you wish to achieve a maximum velocity of 600 RPM, running in the full step mode, a motion program with the appropriate settings could look as follows:

```
A=1000 ^ Accelerate in 1 second or less
B=270  ^ Start speed is 270 steps/second
D=500  ^ Decelerate in 0.5 seconds
M=2000 ^ Maximum velocity is 2000 steps/second (600 RPM)
H=2000 ^ Must be greater than or equal to M
SLEW  ^ Accelerate the motor and slew at 2000 steps/sec (600RPM)
WAIT 200 ^ Continue to slew for 2 seconds
STOP     ^ Decelerate the motor and stop
```



MOTOR RESOLUTION MR

Description The parameter **MR** is the number of pulses, (steps or microsteps) required for one revolution of the motor shaft. This value is used for encoder or user unit related commands.

Format **MR** = *nn*

Mode Immediate/Stored

Range 200, 400, 1000, 2000, 5000, 10000, 18000, 20000, 21600, 25000, 25400, 25600, 36000, 50000 and 50800 steps per revolution. Limited in firmware of the specific indexer.

Related Functions **ER, CG, CV, DW, EF, UR**

Example `MR=2000`

Establishes in software that 2,000 motor pulses will move the motor shaft one revolution. Make sure your drive's hardware is configured for 10 microsteps per each full step.

Sample Program

Hazard Note

The MR command must be executed in a program prior to generating motion. (P51X only). The P51X utilizes the MR command internally to reconfigure the drive. When this command is issued the P51X drive will de-energize the motor while reconfiguring, (vertical loads may fall!)

NUMBER OF MOTOR PULSES DURING LAST INDEX N

Description The number of motor pulses moved during the last indexed distance. This is always a positive integer value and is used when the **GO** command is executed. This parameter is not reset when a **MOVE 0** command is executed as a false move.

Format **N** = *nn*

Mode Immediate/Stored

Range ± 2.1 Billion

Related Functions **GO, MOVE, POSITION, RUN, SLEW**

Example

```
VERIFY N
"N = 1500"
```

The last move distance was 1500 motor pulses.

```
N = 1000  ^ Set the value prior to a GO command
GO        ^ Move distance N
```

Sample Program

```
MOVE 1000 ^ Move 1000 steps
WAIT 100  ^ Wait 1.0 seconds
GO        ^ Repeat last move distance
WAIT 100  ^ Wait 1.0 seconds
GO        ^ Repeat last move distance
WAIT 100  ^ Wait 1.0 seconds
```

OTHER FUNCTIONS OF

Description This parameter is used to activate "Other Functions" available within the ICL System and is represented as an eight (8) bit number. The purpose of the parameter **OF** is to place under user control a group of advanced functions for motion control. These advanced functions can be activated and deactivated as required within a motion program and will configure the indexer to perform a specific task utilizing a new set of "RULES" that correspond to the active bit. All eight bits must be entered or the command will not be accepted by the indexer. A "0" designates that the function is "INACTIVE". An "X" designates that the current definition is to be unchanged and the "1" designates that the function is "ACTIVE".

Format **OF =** *bbbbbbbb*

Mode Immediate/Stored

Range 0 = Deactivate (default), 1 = Activate, X = Do Not Change

Related Functions **CONTINUOUS, ECHO, GO, MOVE, PAUSE, POSITION, RUN, SLEW**

Additional Description Below is a listing of bits #1 through #8 and their assigned function.

Other Function BIT #	Function when active (1)
xxx00xxx	
	Pause/Continuous processing
	Echo (see also **ECHO** command)
	ICL unit designation display
	Reset outputs on PROGRAM FAULT
	Reset outputs on DRIVE FAULT
	Reserved

OF=1XXXXXXX
Bit #1 is a reserved bit. This bit must remain set to 0.

OF=X1XXXXXX
Bit #2 is set to 1, the indexer will reset all outputs when the ICL program encounters a command or motion error. The outputs are reset to their default OFF condition as defined by the Output Definition parameter **OD1**. This is a software interrupt that performs a similar function to **RESET 1 2 3 4 5 6 7 8** on a program error. If set to 0 this feature is disabled.

OF=XX1XXXXX
When bit #3 is set to 1, the indexer will reset all outputs when its Drive Fault input is ACTIVE, (internal on Power/Drive/Indexer Systems). If set to 0 this feature is disabled.

OF=XXXXX1XX
Bit #6. If value is 1 then the ICL unit designation will not be sent to the host device. If value is 0 the unit designation will be sent to the host device (example: 0 >).

OF=XXXXXX1X
Bit #7 displays if characters will be echoed to the host device or not, (0 = ECHO ON, 1 = ECHO OFF, also see command **ECHO**).

OF=XXXXXXX1
Bit #8. If value is 1 then pause processing mode is active. If value is 0 then continuous processing mode is active. This bit may be changed by the **PAUSE** or **CONTINUOUS** command.

Example

```
OF = 01100001
^ Reset outputs on drive or program fault
^ Execute program in pause mode
```

Sample Program

```
^ Auto execute program
OF = 01100001
^ Reset outputs on drive or program fault
^ Execute program in pause mode
SET 6
^ Output 6 is used by the PLC to determine if the
^ Indexer is up and running
^ If the drive or program faults then output 6 is
^ Turned off
```

Hazard Note

An output can be utilized to disengage a power off brake. Bits 2 & 3 will reset the output and allow brake engagement on; a power failure, program error or drive fault.

Other outputs affected may be to reset pumps, values or other external events that must be stopped on a controller fault.

POSITION OF MOTOR IN PULSES P

Description The parameter **P** is the current position in motor pulses sent to the drive. This parameter is continuously updated in 2 MS intervals during the execution of all motion commands and whenever the **JP** or **JM** inputs are active.

Format **P** = *nn*

Mode Immediate/Stored

Range ± 2.1 Billion

Related Functions **GO, HOME, MOVE, POSITION, RUN, SLEW, Jog inputs JP and JM**

Example

```
VERIFY P
"P = 2567"
```

Shows that the current motor position is 2,567 pulses in the positive direction.

Sample Program

```
+ RUN    ^ Move continuous in the CW direction
SYNC 1   ^ Poll input 1 every 100 microseconds
STOP     ^ Stop motion
VERIFY P ^ Display current position
IF (P <= 125000) UNDER
IF (P > 125000) OVER

(UNDER) SEND -1 "DIMENSION IS UNDER SIZE"
QUIT
(OVER) SEND -1 "DIMENSION IS WITHIN TOLERENCE"
QUIT
```

Programming Hint

The value of **P** is automatically set to zero after the **HOME** command is completed.

Hazard Note

Changing the value of **P** when encoder position maintenance mode is active will result in uncontrolled motion. Changing **P** causes a position error that the indexer will react to correct. Turn off encoder position maintenance prior to changing the position counter **P**.

POWER-UP TIME Q

Description **The parameter Q is the time delay required by the drive to return to full power from "reduce current" mode or the "shut down current" mode as set by parameters R and S respectively.** The time delay is entered in hundreds of seconds and may vary depending on the manufacture and type of drive. The minimum value is "0", the maximum value is 127 (1.27 seconds). After the indexer commands the drive into the "low power" or "no power" mode, any motion command encountered will cause a time delay of **Q** before sending pulses to the drive, thus allowing the drive to come up to "full power". **This time delay is required whenever the time between motion commands is greater than time periods defined by parameters R or S.** Consult the manufacturer's drive specifications for the required enable time.

Format **Q** = *nn*

Mode Immediate/Stored

Range 0 to 127

Related Functions **R, S, GO, HOME, MOVE, POSITION, RUN, SLEW, Jog inputs JP and JM**

Example

```
Q=50
```

Commands the ICL Indexer to wait one half second before generating pulses after returning the drive to full current.

Sample Program

> ***Programming Hint***
>
> The parameter **Q** is required in applications where the reduce power time, **R**; or no power time, **S** are non-zero values.

REDUCE CURRENT TIME R

Description This parameter sets the time delay utilized after each move command before the "Low Power" signal is sent to the drive. If 0, the controller assumes that the low power function will not be used. For a value greater than 0, the drive will be commanded to reduce power **R** hundreds of seconds after each move command is completed. When a new move command is executed the drive will be returned to full power and the pulses will begin **Q** hundreds of seconds later**. The parameter Q is used to allow the drive time to attain full power.**

Format **R** = *nn*

Mode Immediate/Stored

Range 0 to 32,767

Related Functions **Q, S, GO, HOME, MOVE, POSITION, RUN, SLEW, Jog inputs JP and JM**

Example R=200
The indexer will command the drive to reduce current to low-power two seconds after motion is complete.

Sample Program
```
^ Code from a autoexecute or setup program
S = 2000
R = 100
Q = 50
```

Hazard Note
Use of this command may be application dependent.
Use R = 0 for vertical applications where the load may be back driven !!!

SHUTDOWN CURRENT TIME S

Description This parameter sets the time delay utilized after each move command before the "No Power" signal is sent to the drive, (shutdown current). If 0, the indexer assumes that the no-power function will not be used. For a value greater than 0, the drive will be commanded to de-energize the motor **S** hundreds of seconds after each move command is completed. When a new move command is executed the drive will be returned to full power and the pulses will begin **Q** hundreds of seconds later. **The parameter Q is used to allow the drive time to attain full power.**

Format **S** = *nn*

Mode Immediate/Stored

Range 0 to 32,767

Related Functions **Q, R, GO, HOME, MOVE, POSITION, RUN, SLEW, Jog inputs JP and JM**

Example S=2000
The indexer will command the drive to reduce current to no power twenty seconds after motion is complete.

Sample Program Code from a autoexecute or setup program
```
S = 2000
R = 100
Q = 50
```

Hazard Note
Use of this command may be application dependent.
Use S = 0 for vertical applications where the load may be back driven !!!

SCREEN CHARACTERS SC

Description The screen characters parameter **SC** is used to define the number of characters displayed to the host device before a carriage return. This will allow the user to define a screen width from 20-80 characters as used by many dumb terminals.

Format **SC** = *nn*

Mode Immediate/Stored

Range 20 to 80

Related Functions **T, DIRECTORY, HELP, LIST, STATUS**

Example
```
SC=40 ^ Configure the indexer for 40 characters.
```

Note the parameter "T" is used to limit the number of display lines transmitted to the host device.

Sample Program Program used with a 4 line 20 character display
```
T = 4
SC = 20
```

TRANSMIT # OF LINES TO T

Description Allows the user to set the number of display lines transmitted to the host device before pausing. **DIRECTORY, LIST, STATUS** and **HELP** are some of the commands affected. A partial listing **T** lines long will be displayed to the host prior to pausing, and will scroll when any character is typed. Note the parameter **SC** is used to set the screen width sent to the host before a return is generated. At a pause point the escape **<ESC>** key can be utilized to terminate the listing being displayed.

Format **T** = *nn*

Mode Immediate/Stored

Range -1 to 256

Related Functions **SC, DIRECTORY, DUMP, HELP, LIST, STATUS**

Example
```
T=4 ^ Configure the indexer for 4 lines.
```

Note the parameter "SC" is used to limit the number of characters/lines transmitted to the host device.

Sample Program Program used with a 4 line 20 character display
```
T = 4
SC = 20
```

UNIT AXIS DESIGNATOR U

Description Axis designator assigned to a given indexer. **0-9** and **A-Z** are allowable for a total of 36 unique devices. Axis designators are required when daisy chaining devices. The axis designator will become part of the indexer's prompt thus the source of messages can be identified.

Format **U** = *c*

Mode Immediate/Stored

Range 0 - 9, A - Z

Related Functions **:c, OF, SEND**

Example When the command **U=X** is issued this device's address is "X".

The ICL System prompt will be "**X>**" and all error messages reported to the host will be preceded by an "**X**"

Sample Program

Programming Hints

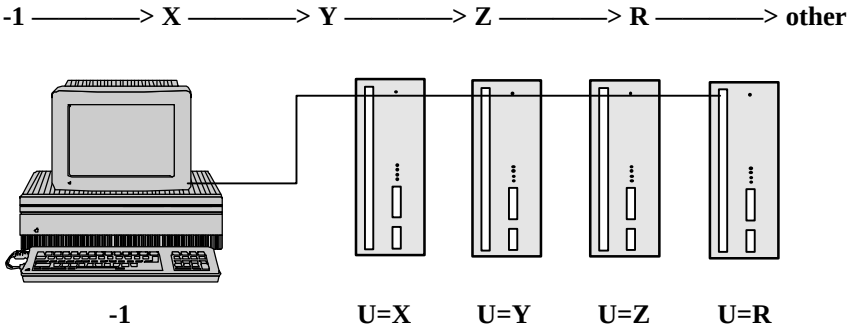
Other Functions is used to control display of the ICL unit designator. **OF=XXXXX1XX**, then the ICL unit designation will not be sent to the host device. If **OF=XXXXX0XX** the unit designation will be sent to the host device (example: "0>").

Use practical names when possible. For instance in a four axis system you may wish to use **U=X, U=Y, U=Z** and **U=R** for the rotary axis. This will simplify the programming of a multi-axis system.

When using RS-232 communications to command a multiple axis system, the unit designator will allow commands to be directed to a specific axis. The example below will command axis X to move 10 units, axis Y 15 units and axis Z to set output #1.

Transmitted strings are:

```
: X     MOVE 10
: Y     MOVE 15
: Z     SET 1
```



UNITS ACTIVE UA

Description This parameter displays the unit of measure that will be utilized during subsequent motion commands. The chart that follows indicates the ICL command names and the value of parameter **UA** after the desired command has been executed. It also indicates which parameter is used to establish the desired units of measure when performing motion.

Format **UA** = *n*

Mode Read only

Range

ICL Command	Units Active	Parameter **UA** Value
MUNITS	Motor units	**UA** = 0
EUNITS	Encoder units	**UA** = 1
UUNITS	User units	**UA** = 2

Related Functions **ER, MR, UR, EUNITS, MUNITS, UUNITS**

Example In the following example we wish to position the motor at 1º (degree) increments and a motor microstep resolution of 36000 steps/revolution has been selected. The encoder resolution after quadrature is 4000 counts/revolution. The motor is positioned at 120º positions utilizing user units, motor units and encoder units.

Sample Program

```
UR=360  ^ User resolution
ER=4000 ^ Encoder resolution
MR=36000^ Motor resolution

UUNITS  ^ Select user units
UP=0    ^ Set user position to zero
POS 120 WAIT 100
POS 240 WAIT 100
POS 360 WAIT 100
```

Code is easy to read & understand, distance is in degrees.

```
MUNITS  ^ Select motor units
P=0     ^ Set motor position to zero
POS 12000 WAIT 100
POS 24000 WAIT 100
POS 36000 WAIT 100
```

Code is in 1/100 degree increments.

```
EUNITS  ^ Select encoder units
E=0     ^ Set encoder position to zero
POS 1333 WAIT 100
POS 2667 WAIT 100
POS 4000 WAIT 100
```

Code is hard to read & understand.

Programming Hints

MR=5000, UR=10, **UUNITS**

The ICL indexer converts the requested motion by the following formula:

MOVE *n* equals *n**MR/UR equals *n**500 motor pulses

Use of the MOVE command when (*n**MR/UR) is a non-integer value will result in truncation of your requested motion. Try using the POSITION command when this situation exists. The ICL indexer will calculate its new position from zero or home, and account for the dither associated with truncation.

USER RESOLUTION UR

Description This value is used when commanding motion in user units. The parameter **UR** is an integer number of user units generated during one revolution of the motor shaft. If the value of **UR** is a real number, the user has a choice of selecting a smaller step resolution such as microstepping or by scaling (i.e. 2.54 may be expressed as 254 user units per revolution). Use the **UUNITS** command to utilize user units when commanding motion.

Format **UR** = *nn*

Mode Immediate/Stored

Range ± 2.1 Billion

Related Functions **P, ER, MR, UA, UP, EUNITS, MUNITS, UUNITS**

Example In the following example we wish to position the motor at 1º (degree) increments and a motor microstep resolution of 36000 steps/revolution has been selected. The encoder resolution after quadrature is 4000 counts/revolution. The motor is positioned at 120º positions utilizing user units, motor units and encoder units.

Sample Program

```
UR=360     ^ User resolution
ER=4000    ^ Encoder resolution
MR=36000   ^ Motor resolution
UUNITS     ^ Select user units
UP=0       ^ Set user position to zero
POS 120 WAIT 100
POS 240 WAIT 100
POS 360 WAIT 100
```

Example In the next example we wish to position a lead screw, P=5 (0.2 inch/rev) at increments of 0.001 inches.

Sample Program

```
ER=800     ^ Encoder resolution
MR=400     ^ Motor is in the half step mode
UR=2
UUNITS
MOVE 1000 ^ Move 1.000 inches
```

Programming Hints

MR=5000, UR=10, UUNITS

The ICL indexer converts the requested motion by the following formula;

MOVE *n* equals *n**MR/UR equals *n**500 motor pulses

Use of the MOVE command when (*n**MR/UR) is a non-integer value will result in truncation of your requested motion. Try using the POSITION command when this situation exists. The ICL indexer will calculate its new position from zero or home, and account for the dither associated with truncation. The user position parameter is updated at the end of a move, it is a calculated value. Be careful when using the "IF" command.

CURRENT PROGRAM # W

Description This parameter contains the program number of the current or last program executed. This parameter is utilized to locate a program or subroutine when debugging a motion routine. The **DIRECTORY** command is utilized to locate the program name associated with the program number.

Format **W** = *n*

Mode Read only

Range 0 - 87

Related Functions **X, Z, DIR, LIST**

Example

```
VERIFY W
"W = 4"
```

The last program executed was program number 4 on your directory. “W = 4”

Use the **DIR** command to determine the name of Program #4, Parameter **W**.

Sample Program

Sample Usage

```
0 | SETUP      1 | PROG1      2 | PROG2      3 | TEST
4 | SAMPLE1    5 | SAMPLE2    6 | SAMPLE3    7 | TEST1
8 | TEST2      9 | TEST3
                                82| PN1011     83| PN1012
84| PN1013     85| MACHINE1   86| MACHINE2   87| LAST
```

Programming Hints

The TRACE command as another powerful debugging tool.

CURRENT PROGRAM INSTRUCTION ADDRESS X

Description This parameter is utilized to locate the program instruction address of the current or last instruction executed in a motion program***. This parameter is utilized to help locate errors within a motion program.***

Format **X** = *n*

Mode Read only

Range 0 - 87

Related Functions **W, Z, DIR, LIST**

Example

```
VERIFY                                                    X
"X = 5258"
```

The last instruction executed was at address 5258 in program number "W" on your directory.

Use the **DIR** command to determine the name of Program #4, Parameter **W**.

View Program # 4 utilizing the **LIST** command. The listing of program 4, "SAMPLE1" shows line 20 and address 5258 is the location of the error.

Sample Program

```
LIST SAMPLE1
1 5136 A=1000
.
20 5249 (LABELTOLONG) UNTIL 2
```

... Label name is longer than eight characters
Address 5258, Parameter X

> ***Programming Hints***
> The TRACE command as another powerful debugging tool.

LOOP COUNTER Y

Description This parameter shows the current value of the loop counter in number of loops. In the first pass through a **LOOP** the value of **Y** is random as you may have nested loops. The value of **Y** is decrements by one with each pass through the loop sequence.

Format **Y** = *nn*

Mode Read only

Range Limit = 255

Related Functions **LOOP**

Example
```
VERIFY Y
"Y = 10"
```

Sample Program
```
(TOP) MOVE 1000
WAIT 10
V1 = Y  ^ Assign parameter y to a variable for display
SEND -1 "MAKING MOVE #" V1 " OF 5"
LOOP 4 TOP
QUIT
```

> ***Programming Hints***
> In the first pass through a loop the value of **Y** is random as you may have nested loops. We suggest you utilize a math variable to control program flow. See example below.

Sample Program
```
V1 = 0
V2 = 5
(TOP) MOVE 1000
WAIT 10
V1 = V1 + 1
SEND -1 "MAKING MOVE" V1 "OF" V2
IF (V1 < V2) TOP
QUIT
```

CURRENT PROGRAM LINE NUMBER Z

Description This parameter is utilized to locate the program line number of the current or last program line executed in a motion program***. This parameter is utilized to locate an error within a motion program***.

Format **Z** = *nn*

Mode Read only

Range

Related Functions **W, X, DIR, LIST**

Example While executing a program, the ICL system reports

```
"#17 LABEL TO LONG" error.

VERIFY W VERIFY X VERIFY Z
"W = 4"
"X = 5258"
"Z = 20"
```

Sample Program Use the **DIR** command to determine the name of Program #4, Parameter **W**.

```
0 | SETUP      1 | PROG1     2 | PROG2      3 | TEST
4 | SAMPLE1    5 | SAMPLE2   6 | SAMPLE3    7 | TEST1
8 | TEST2                    9 | TEST3
                             82| PN1011     83| PN1012
84| PN1013     85| MACHINE1  86| MACHINE2   87| LAST
```

View Program # 4 utilizing the **LIST** command. The listing of program 4, "SAMPLE1" shows line 20 and address 5258 is the location of the error.

```
LIST SAMPLE1
1 5136 A=1000

.
20 5249 (LABELTOLONG) UNTIL 2 CYCLE
.
```

... Label name is longer than eight characters Address 5258, Parameter X

Programming Hints

The TRACE command as another powerful debugging tool.

The value Z may become corrupted with errors associated with stack-overflow or memory errors.

ICL VARIABLES (V0 to V99)

Description ICL Variables contain user defined integer values or text strings that can be used in conjunction with ICL Commands to provide application specific information and define how the indexer will interface to its environment. Variables are maintained in nonvolatile battery backed memory and may be changed as frequently as required by the user. Variables may contain either integer values or string values. **There are 100 variables total which may be defined as either integer or string.**

Format V*n* = *nnn*

V*n* = "text string"

Mode Immediate/Stored

Range V0 - V99

Related Functions **CLEAR, CALL, IF, SEND, SHOW, UNTIL, $**

Example

```
V1=300 V55=77 V22=34000 $V3="ENTER MOVE DISTANCE"
SHOW
"V1 = 300"
"V3 = ENTER MOVE DISTANCE"
"V22 = 34000"
"V55 = 77"
```

Sample Program

Programming Hints

It is good programming practice to use variables wherever possible within your programs. Write your MAIN program to accept variables from subroutines which assigns data to variables for the selected process. This will reduce typographical errors, reduce the required memory storage area and increase speed of command execution.

The **SHOW** command is used to display variables that are not cleared.

INTEGER VARIABLES

Description ICL Integer Variables can be changed in program mode or in the immediate mode through use of the *v* = *nn*, or **PROMPT** "message" *v* commands. The variable name is represented by *v* and it's new numeric value is an integer represented by *nn*.

The **SHOW** and **VERIFY** commands are used to display the current values of the ICL Variables. The **SHOW** command sends the table of all active variables to the host device. The **VERIFY** *v* command is used to display the current value of one variable, where *v* is the name of the variable to be reviewed. When executed, the current value of that variable will be returned to the host device.

Format Vn = *nnn*

Mode Immediate/Stored

Range ± 2.1 billion

Related Functions **CLEAR, IF, PROMPT, SEND, SHOW, UNTIL, VERIFY**

Example

```
V1=300 V55=77 V22=34000
SHOW
"V1 = 300"
"V22 = 34000"
"V55 = 77"
```

Sample Program The user requires a program that will allow him/her to set a number of motions, the move distance and the wait time between motions. This sample program prompts the user to input the desired data. Input #1 begins execution of the sequence.

```
(A0) PROMPT "ENTER NUMBER OF MOTION CYCLES" V1
V1=V1-1 ^ Set v1 for use in loop command
PROMPT "ENTER MOTION DISTANCE " V2
PROMPT "THE DESIRED WAIT TIME IS " V3
SEND -1 "TRIGGER INPUT #1 WHEN READY TO BEGIN PROCESS"
PAUSE
(START) UNTIL 1 START
(A) MOVE V2
WAIT V3
LOOP V1 A
SEND -1 "PROCESS IS COMPLETE"
JUMP A0
```

Programming Hints

Variables can be cleared through use of the **CLEAR** [*v*] command. Integer variables are dynamically allocated and use four-bytes of user storage when defined. The **CLEAR** command will free this storage.

INTEGER VARIABLES/MATHEMATICAL FUNCTIONS

Description Mathematical functions such as multiplication, division, modulo, addition, and subtraction, (***, /, %,** + and -) may be performed on numeric variables. Note that when mathematical functions are performed on variables the result is an integer value where the remainder portion of the number is truncated.

Mathematical expressions are evaluated from left to right with the following order of precedence; multiplication, division and modulo then addition and subtraction, (*** / %** then **+ -**). The user may wish to direct the order in which the expression is evaluated by use of the "(" parenthesis commands.

Format Refer to the section on ICL COMMAND DESCRIPTIONS for examples of mathematical functions, (*, /, %, + and -)

Mode Immediate/Stored

Range ± 2.1 billion

Related Functions **IF, UNTIL**

- Mathematical expressions: ***, 1, %, +, -**
- Variable or parameter compares: **<>, =, >, <, >=, <=.**
- Boolean operations: **OR, AND**.

Example

```
V0 = 360      V1=500      V2=200
V3 = V1 / V2              V3 = 2
V4 = V1-V2                V4 = 300
V5 = V0/V1*V2             V5 = 0
V6 = V1*V2+V0             V6 = 100360
V7 = V1*(V2+V0)           V7 = 280000
V8 = V0%V2                V8 = 160
```

> ***Programming Hints***
>
> Note that when mathematical functions are performed on variables the result is an integer value where the remainder portion of the number is truncated. Multiply numbers before you divide them to reduce the truncation error. The modulo "%" command will allow you to work with truncated values.

Example Expressions comparing variables and/or parameters can also be used in decision making within the ICL System. The **IF** command is used to provide for conditional branching within a motion program. If the **IF** condition evaluates "TRUE", then control of the program branches to label **ccc**. If the **IF** condition evaluates "FALSE" then control passes to the next line in the program, (see **IF** command).

Evaluation conditions:
- Variable or parameter compares: **<>, =, >, <, >=, <=.**
- Boolean operations: **OR, AND**.

```
IF (Vn comparison operator Vn) ccc
IF (V1 > V2) LABEL1
IF ((P > V2) AND (V4 > V5)) LABEL3
IF (V1 > V2) APPLE ^ If true branch to label APPLE
IF ((V3 > V4) AND (V3 <> V5) BOTTOM
^ If True go to label "BOTTOM"
```

ASCII STRING VARIABLES

Description ICL string variables can be changed in program mode or immediate mode through use of ***$V*** **=** ***xxxx*** or **PROMPT "message"** ***$V*** commands.

Format **V*n*** = "ASCII text string enclosed in quotation marks"

Mode Immediate/Stored

Range The maximum string length is 80 ASCII characters, (0-9 and A-Z).

Related Functions **CALL**, **IF**, **JUMP**, **PROMPT**, **UNTIL, SEND**

Example

```
V1 = "ENTER YOUR NAME"
$V18 = "ENTER YOUR NAME"
```

Sample Program An example of a program using integer and string variables:

The user requires the ability to select a subroutine program. The name of the subroutine is the part number to be manufactured. A dynamic program shown below will prompt the user for the subroutine name (part number). Input #1 begins execution of the sequence.

```
$V18 = "ENTER YOUR NAME"
PROMPT $V18 $V0
^ Store last users name in as a string in var v0
(A0) PROMPT "ENTER PART NUMBER " $V1
PROMPT "ENTER NUMBER OF PARTS TO MANUFACTURE " V2
SEND -1 "TRIGGER INPUT #1 WHEN READY TO BEGIN PROCESS"
PAUSE
(START) UNTIL 1 START
(A) CALL $V1
V2=V2-1
IF (V2 = 0) END
JUMP A
(END) SEND -1 "PROCESS IS COMPLETE"
JUMP A0
```

Programming Hints

String variables are dynamically allocated to the user storage when defined. The **CLEAR** command will free this storage

A string variable must be enclosed in quotations when entered, except when using the **PROMPT** command. The variable must also be preceded by a "**$**" command to be distinguished from a label or program name.

Comparison of ASCII strings is not supported.

Description ICL Flags **F**1 to **F**64 are user defined flags and can be utilized in conjunction with ICL Commands to provide application specific information and define how the indexer will interface to its environment. A **SET** or **RESET** command is used to change a flag to "TRUE" or "FALSE". ICL Flags are maintained in nonvolatile battery backed memory and may be changed as frequently as required by the user. A flag name begins with the letter **F** and is followed by the numbers 1 to 64 (e.g. F1, F18).

ICL Flag **F**0 defines if motion is currently active on this indexer. A value of 1 indicates that the motion is moving, while 0 indicates that motion is complete.

Format **F*n*** = *nn*

Mode **F**1 to **F**64 are Immediate/Stored

Flag **F**0 is Read Only

Range F1 to F64, Flag **F**0 is reserved and "TRUE" when motion is being commanded.

Related Functions **IF, RESET**, **SET**, **UNTIL, VERIFY,** \

Example
```
SET F1 F4 RESET F2 F3
VERIFY F1 VERIFY F2 VERIFY F3 VERIFY F4
"F1 = 1"
"F2 = 0"
"F3 = 0"
"F4 = 1"
```

Additional Description The **IF** (Condition) **ccc** command is used to provide for conditional branching within a motion program. If the **IF** condition evaluates "TRUE", then control of the program branches to label **ccc**. If the **IF** condition evaluates "FALSE" then control passes to the next line in the program.

The ICL command **IF** (F*n*[\]) **ccc** tests the status of ICL flags defined by number *n* and conditionally branches to the line label **ccc** within the program on a "TRUE" condition. A backslash "\" after the Flag tests for a "FALSE" condition. The **IF** statement will result in branching to label **ccc** if comparisons are "TRUE". The **IF** statement will cause the next program line to be executed if the comparison is "FALSE". ***Restriction: Label ccc must start with an ALPHA character.***

Evaluation conditions:
- Input, Output and Flag status testing for "Active" or "Inactive"
- (Backslash " \ " after an I/O or Flag checks for an inactive condition).
- Boolean operations: **OR, AND**.

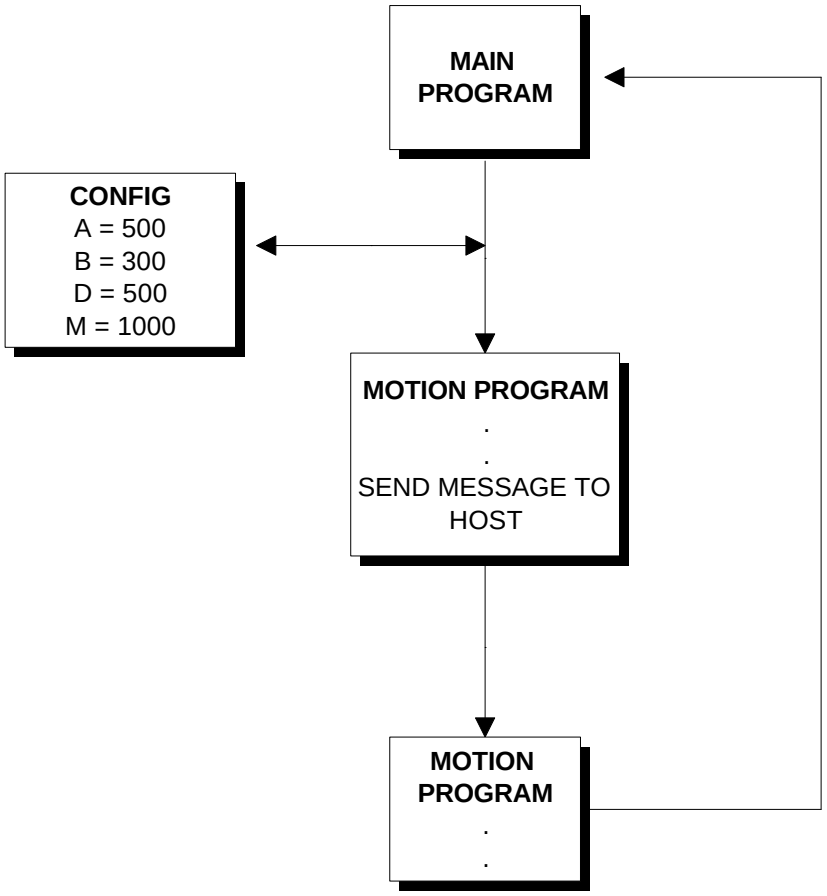
Example
```
(AA) IF (F1 AND F2) AA
^ If ICL flags 1 & 2 are TRUE branch to label AA
IF (F6) ACTIVE6
^ Test flag 6, if active branch to label ACTIVE6
QUIT           ^ If flag 6 is inactive quit program
(ACTIVE6) HOME ^ Execute home command
IF (F1\) LABEL2
IF ((V1 > V2) AND (V4 > V5) AND (F0\)) LABEL3
.
.
.
.
```

```
MOVE 1000
(HOLDING) IF (F0) HOLDING
^ Wait until moving flag is inactive
JUMP AA
QUIT
```

Programming Hints

ICL flag F0 is dedicated to the ICL System. Flag F0 is active while moving and is read only and cannot be set or reset.

FLOW CHART YOUR APPLICATION



Programming Hints

Maintain a list of all parameters in a configuration file on the ICL indexer to prevent loss of vital data necessary for interfacing the indexer to your environment.

Flow charting your application will clarify your thought process and streamline the implementation of your solution.

ICL PARAMETERS - (ADVANCED)

Parameters in the following section are related to users that will be utilizing input, outputs or closed loop encoder capabilities in their application.

If you will not be utilizing inputs, outputs or an encoder you may skip the following section.

Parameters related to inputs and outputs allow the user to define how the ICL indexer will react and interface to its environment. The ability to define inputs as active "HIGH" or "LOW" allow the user to use normally-open or normally-closed switches for inputs.

Limit Action modifiers allow the ability to define actions of the indexer to active limit inputs, abort program, stop motion immediately, etc. Outputs may also be defined as active "HIGH" or "LOW" to allow a wide variety of devices being controlled via the outputs.

Since the ICL System is operated on different indexers, all functions might not be available on the indexer that you may be using. Please refer to your indexer user guide for a list of exceptions on your device.

Parameters related to encoders allow definition of the encoder and selection of stall detection position maintenance or homing on the Z-channel of the encoder.

CORRECTION GAIN, POSITION CG

Description When the encoder position maintaining function is active, the correction gain times the motor error (**CG** * **ME**), is the velocity used to step the motor into the dead band window. The parameter **CV** sets the maximum correction velocity allowed during position correction.

Format **CG** = *nn*

Mode Immediate/Stored

Range (**CG** * **ME**) ≤ **CV**

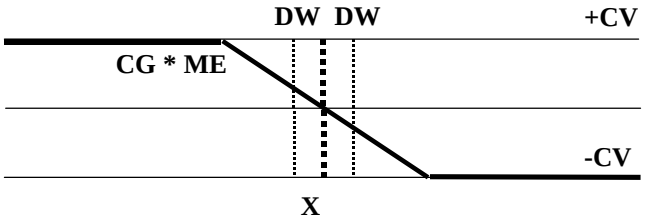
B (base speed) < **CG** ∗ **ME** ≤ **CV** (maximum correction velocity)

Related Functions **CV, DW, EF, ER, MR**

Example
```
CG=100 ^ Sets the correction gain to 100 sec-1
```

Sample Program

Graphic Example



X = Commanded Position
DW = Dead Band Window
CV = Correction Velocity Max
CG * ME = Correction Velocity

CORRECTION VELOCITY, MAXIMUM CV

Description When the encoder position maintaining function is active, the parameter **CV** sets the maximum correction velocity allowed during position correction. This parameter is entered in motor pulses per second. **CV** ≥ (**CG** * **ME**)

Format **CV** = *nn*

Mode Immediate/Stored

Range **B** (base speed) < **CV** (maximum correction velocity) < **M** (maximum velocity)

Related Functions **CG, DW, EF, ER, MR**

Example
```
CV=2000
^ Set the maximum correction velocity to 2000 motor
^ pulses per second
```

Sample Program See **CG** Correction Gain**, POSITION** Command

DEAD BAND WINDOW "ENCODER" DW

Description The dead band window **DW** is defined in encoder units after quadrature. Parameter **DW** is the window of encoder error associated with any commanded physical position. For the case **DW** = *10*, the motor is considered on position when the encoder is within ten counts of the commanded position.

Format **DW** = *nn*

Mode Immediate/Stored

Range This parameter has a minimum value when the motor resolution is less than three times the encoder resolution. The minimum value is calculated by the following:

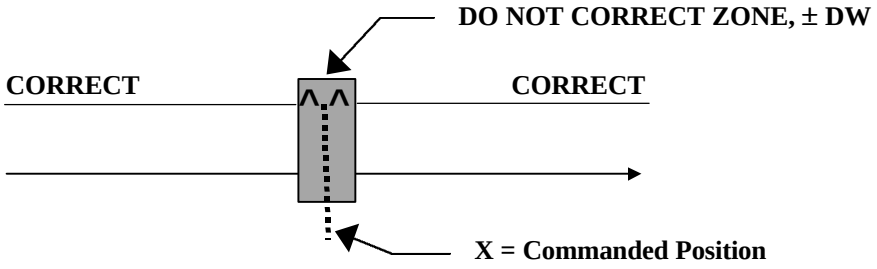
- For **MR** < **ER** **DW** = {Integer(**ER**/**MR**)} * 2 +1
- For **ER** < **MR** < **3*ER** **DW >= 3**

  Due to the mechanical variations in the motor.
- For 3***ER** < **MR** **DW >= 0**

Related Functions **CG, CV, EF, ER, MR**

Example `ER = 4000, MR = 1000, DW=9`

i.e. the motor is located within nine encoder counts of the commanded position).



ENCODER POSITION E

Description The parameter **E** reports the current encoder position in encoder pulses after quadrature. This parameter is continuously updated whenever the encoder shaft is displaced.

Format **E** = *nn*

Mode Immediate/Stored

Range ± 2.1 Billion

Related Functions **CG, CV, EF, ER, MR, PE**

Example
```
E=-4000
^ Define the current position as -4000 encoder counts
```

Sample Program Reset the position counter to correspond to the current encoder position.

```
EF = 00000000
P = E * MR / ER
EF = 11010000
^ Turn on stall detect, pos maint. and home functions
```

Programming Hints

Disable encoder functions EF = 00000000 prior to changing the encoder position. This will prevent accidental motion from occurring as the ICL system attempts to correct for the error.

In applications that are uni-directional, be sure to reset the counter to prevent mathematical overflow.

ENCODER RESOLUTION ER

Description The parameter **ER** is the number of encoder pulses generated during one revolution of the motor shaft. This value is used for encoder related commands. The number is post-quadrature (the number of encoder lines multiplied by 4). A 1000 line encoder produces 4000 pulses per revolution. Hence, **ER** = *4000* is entered. The **EUNITS** command sets parameter **UA** = 1. This tells the indexer to utilize encoder units when commanding motion.

Format **ER** = *nn*

Mode Immediate/Stored

Range ± 2.1 Billion

Related Functions **CG, CV, DW, EF, EUNITS, MR, PE, SP**

Example
```
ER=4000
^ Encoder resolution is 4000 pulses per revolution
```

Sample Program
```
MR=20000  ^ Motor resolution is 20000 steps per rev
ER=4000    ^ Encoder resolution is 4000 pulses per rev
EUNITS    ^ Set move units to encoder pulses
MOVE 4000
^ Move 4000 encoder counts or 1 shaft rev by
^ generating 20000 motor pulses
```

Programming Hints

MR=5000, ER=4000, EUNITS

The ICL indexer converts the requested motion by the following formula:

MOVE *n* equals *n**MR/ER equals *n**5000/4000 motor pulses

Use of the MOVE command when (*n**MR/ER) is a non-integer value will result in truncation of your requested motion. Try using the POSITION command when this situation exists. The ICL indexer will calculate its new position from zero or home, and account for the dither associated with truncation.

ENCODER FUNCTION EF

Description The encoder function parameter is an eight (8) bit number used to define how the indexer will utilize the encoder feedback. All eight bits must be entered or the command will not be accepted by the indexer. A "0" designates that the function is "INACTIVE" and "1" designates that the function is "ACTIVE". An "X" designates that the value is not to be changed when the new definition is entered.

Format **EF** = *bbbbbbbb*

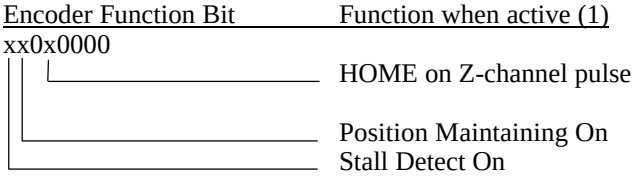
Mode Immediate/Stored**, Set to EF=00000000 on power-up.**

Range All eight bits must be entered as "1", "0" or "X".

Related Functions **CV, CG, DW, ER, MR, PE, SP**

Example
```
EF=11000000 ^ Turn on stall detect & maintenance
```

Additional Description The following table defines the ICL interpretation:



EF=1XXXXXXX
When "0", the stall detect feature is "INACTIVE". When "1" the stall detect feature is "ACTIVE". If the encoder position is greater than the Maximum Position Error, **PE**, for a given commanded position; the program execution will be interrupted and the indexer will begin execution of the program defined by the Stall Detect Program parameter **SP**.

EF=X1XXXXXX
When "0", the position maintaining feature is "INACTIVE". When "1" the position maintaining feature is "ACTIVE". The indexer will attempt to step the motor as required to stay within the Dead Band Window, **DW**. The parameters **CV, PE, ME** and **CG**, maximum correction velocity, position error, current motor error and correction gain respectively; will be utilized to determine the velocity during correction.

EF=XXX1XXXX
When "0", the home on Z-channel pulse feature is "INACTIVE". When "1" the encoder home on Z-channel pulse feature is "ACTIVE". When the **HOME** command is given; the ICL System will first locate on the leading edge of the home switch and then back over the home switch to the leading edge of the Z-channel pulse. The home switch and the Z-channel pulse must both be active or the ICL System will report that home was not found. (See the **HOME** command).

Programming Hints

Encoder Functions are disabled on power-up for reasons of safety. The user may re-enable Encoder Functions after verifying that the system is configured and wired properly. Sample programs ECKV and ECKH are shown below and may be used to verify functionality of the encoder.

NOTE: Encoder Functions are turned OFF when the HOME command is issued. Position maintenance and stall detect will not function. After HOME is found they are turned back ON.

Sample program for checking an encoder (ECKV)

Encoder Checking Program

```
EDIT ECKV
                ^ Vertical encoder check routine
EF=00000000     ^ Turn off all encoder functions
PAUSE           ^ Enable pause mode
R=0             ^ Disable power reduction
S=0             ^ Disable power off
MUNITS          ^ Enable motor units
MOV 1           ^ Command drive to powerup
MOV -1
WAIT 200        ^ Wait for motor to be stable
SET 1           ^ Set output to release brake
P=0             ^ Set current motor position to zero
E=0             ^ Set current encoder position to zero
DW=10 ^ Set dead band window to ten encoder counts
M=MR/2          ^ Set velocity to 1/2 rps
MOVE MR/10      ^ Move one tenth revolution cw
WAIT 100        ^ Wait for one second
IF (( E-ER/10 < DW) AND (ER/10-E < DW )) PASSCW
^ Verify encoder is within dead band window
^ Go to label pass-cw
JUMP ERROR
^ Go to label error if not in dead band window
(PASSCW) MOVE -MR/10 ^ move one tenth revolution ccw
WAIT 100        ^ Wait for one second
IF (( E < DW) AND ( E > -DW)) PASSCCW
^ Verify encoder is within dead band window
(ERROR)  RES 1 2 3 4 5 6 7 8
SEN -1 "THE SYSTEM DID NOT PASS PROGRAM ECK"
SEN -1 "**** PROGRAM ABORTED **** "
ABORT           ^ Abort program
(PASSCCW) SEN -1 "ENCODER HAS PASSED TEST"
                ^ Send message to host device
QUIT            ^ Last line of program

~
```

Sample program for checking an encoder (ECKH)

Encoder Checking Program

```
EDIT ECKH
                  ^ Horizontal encoder check routine
EF=00000000       ^ Turn off all encoder functions
PAUSE             ^ Enable pause mode
R=0               ^ Disable power reduction
S=0               ^ Disable power off
MUNITS            ^ Enable motor units
MOV 1             ^ Command drive to powerup
MOV -1
WAIT 200          ^ Wait for motor to be stable
P=0               ^ Set current motor position to zero
E=0               ^ Set current encoder position to zero
DW=10 ^ Set dead band window to ten encoder counts
M=MR/2            ^ Set velocity to 1/2 rps
MOVE MR/10        ^ Move one tenth revolution cw
WAIT 100          ^ Wait for one second
IF (( E-ER/10 < DW) AND (ER/10-E < DW )) PASSCW
^ Verify encoder is within dead band window
^ Go to label passcw
JUMP ERROR
^ Go to label error if not in dead band window
(PASSCW) MOVE -MR/10 ^ Move one tenth revolution ccw
WAIT 100          ^ Wait for one second
IF (( E < DW) AND ( E > -DW)) PASSCCW
^ Verify encoder is within dead band window
(ERROR) SEN -1 "THE SYSTEM DID NOT PASS PROGRAM ECK"
SEN -1 "**** PROGRAM ABORTED **** "
ABORT             ^ Abort program
(PASSCCW) SEN -1 "ENCODER HAS PASSED TEST"
                  ^ Send message to host device
QUIT              ^ Last line of program

~
```

MOTOR ERROR ME

Description The parameter **ME** is the current position error in motor pulses. The motor error times the correction gain, (**ME** * **CG**) is the velocity used for error correction during position maintaining. This parameter can also be utilized within the position correction routine to correct position to within the dead band window. In the case where the encoder is not directly connected to the motor shaft, the value of **ME** will include the motor pulses associated with compliance of the system, (couplings, backlash, etc.).

Format Motor Error is calculated by the Indexer utilizing the following formula:

$$\mathbf{ME} = \frac{\Delta \mathbf{E} * \mathbf{MR}}{\mathbf{ER}}$$

Where **ΔE** is the difference between the commanded position and the current encoder position.

Mode Read only

Range Internally calculated by the ICL updated on 2 MS intervals

Related Functions **CG, CV, EF, ER, MR, PE, SP**

Example
```
Verify ME
"ME = 1253"
```

Sample Program

POSITION ERROR, MAXIMUM PE

Description The parameter **PE** is the maximum position error in encoder pulses after quadrature. This parameter sets the maximum error in encoder pulses before the motor is considered stalled and the subroutine stall detect program **SP** is called, (see Encoder Functions parameter **EF**, to make the stall detect feature active). When a stall condition exists, no position correction will occur unless the indexer is directed to do so from within the stall detect program.

In the case where the encoder is not directly connected to the motor shaft, the value of **PE** must include the encoder counts associated with compliance of the system, (couplings, backlash, etc.).

Format **PE** = *nn*

Mode Immediate/Stored

Range ± 2.1 Billion

Related Functions **EF, ER, SP**

Example

```
ER=4000 ^ Encoder resolution is 4000 pulses per revolution
PE=1000 ^ The maximum position error is ±1/4 revolution
P=17
^ Sets stall detect program to program #17 in directory
EF=1XXXXXXX ^ Turn on stall detect feature
```

STALL DETECT PROGRAM # SP

Description This parameter is used by the indexer to locate the user's "Stall Detect Program". The program number 0 through 87 associated with the user's "Stall Detect Program" is entered as parameter **SP**. The parameter **EF** Encoder Functions is used to activate STALL DETECTION. If -1 is entered then no "Stall Detect Program" is available.

The parameter **EF**, Encoder Functions is used to make the stall detect feature active.
When **EF=0XXXXXXX** the stall detect feature is inactive. When **EF=1XXXXXXX** the stall detect feature is active. If the encoder position is greater than the Maximum Position Error, **PE**, for a given commanded position; the program execution will be interrupted and the indexer will begin execution of the program defined by the Stall Detect Program parameter **SP**

Format **SP** = *n*

Mode Immediate/Stored

Range -1 to 87

Related Functions **EF, ER, PE**

Example

```
ER=4000 ^ Encoder resolution is 4000 pulses per revolution
PE=1000 ^ The maximum position error is ±1/4 revolution
SP=17
^ Sets stall detect program to program # in directory
EF=1XXXXXXX ^ Turn on stall detect feature
```

HOLD INPUT # HI

Description This parameter allows the user to select which programmable input is to be used as the hold input. Example: To assign input #13 as hold, **HI** = 13 command is issued. When this hold input is active, the ICL System will decelerate the motor to a zero velocity and hold execution of the program until released. To deassign the hold input, **HI** = 0 command is issued.

Format **HI** = *nn*

Mode Immediate/Stored

Range 0 to 16 Indexer Dependent

Related Functions **ID1, ID2**

Example An example of how the active hold input effects execution of a program is illustrated below.

Sample Program

```
A=400, D=A, B=1000, MR=5000, M=10000, H=15000
ID2=XXXX0XXX
^ Define input 13 as active when taken low
PAUSE      ^ Set pause mode
HI=13      ^ Assign input 13 as the hold input
P=0        ^ Set current position to zero
POS 20000  ^ Move to position 20000
SET 1      ^ Set output #1 after the move
QUIT       ^ Last line of program
```

For our example we will assume that input #13 becomes active while the motor is moving toward the position 20000. When the input becomes active; the motor will be decelerated to a zero velocity. When the input becomes inactive the ICL System will make the balance of the interrupted move and then set output #1.

Hazard Note

Do NOT reverse define the Hold Input. Your ICL system will lock-up and not respond to commands, unless the HI input is wired and made INACTIVE. It will appear that the RS-232 communication is dead and the unit will not respond to any inputs until the HI input is made INACTIVE.

A service charge will apply to indexers returned due to this cause.

STATUS OF INPUTS I[n]

Description For purposes of simplicity, inputs are set up in banks of eight. Therefore if your indexer has 8 inputs the only parameter available would be **I** or **I1** if you prefer. If your system has 32 inputs then you would have four input parameters **I1** (1-8), **I2** (9-16), **I3** (17-24) and **I4** (25-32).

These are read only parameters that will update automatically based on the status of the inputs. A Status of "0" designates that the input is "INACTIVE" and a "1" designates that the input is "ACTIVE". The input definition parameter **ID** can be used to define the default or "INACTIVE" state of the input as "HIGH" or "LOW". **(Please refer to the user guide for your device to determine the actual number of inputs available.)**

Format **I[n]** = *bbbbbbbbb*

Mode Read Only in blocks of eight.

Range In blocks of 8, Indexer Dependent

Related Functions **ID[n], IO**

Example

```
VER I  ^ VERify status of input bank 1.
"I1 = 10110000" : Inputs 1,3,and 4 are active.

VER I2 ^ VERify status of input bank 2.
"I2 = 00110001" : Inputs 11, 12 and 16 are active.
```

Sample Program

Example An example of how 16 inputs assigned to two banks of **I[n]** could look:

I1, Input Block #1 Input Name	Input Number	I2, Input Block #2 Input Name	Input Number
Input 1	1	Input 9	9
Input 2	2	Input 10	10
Input 3	3	Input 11	11
Input 4	4	Input 12	12
Input 5	5	Input 13	13
Input 6	6	Input 14 (Home Limit)	14
Input 7	7	Input 15 (Clockwise Limit)	15
Input 8	8	Input 16 (Counter Clockwise Limit)	16

INPUT DEFINITION ID[n]

Description For purposes of simplicity, inputs are set up in banks of eight. Therefore if your indexer has 8 inputs the only parameter available would be **I** or **I1** if you prefer. If your system has 32 inputs then you would have four input parameters **I1** (1-8), **I2** (9-16), **I3** (17-24) and **I4** (25-32).

The input definition parameter, **ID**, defines the default state of inputs as either "HIGH" or "LOW". For each bank of inputs on your ICL indexer there is a corresponding bank of input definition bits. A "0" designates that the input will be active "LOW". An "X" designates that the current definition is to be unchanged and the "1" designates that the input is to be active "HIGH". All eight bits for the bank of an input definition must be entered or the command will not be accepted by the indexer. **(Please refer to the user guide for your device to determine the actual number of inputs available.)**

Format **ID[n]** = *bbbbbbbbb*

Mode Immediate/Stored.

Range In blocks of 8, Indexer Dependent.

All eight bits must be entered as "1", "0" or "X".

Related Functions **IO, SET, RESET I[n]**

Example

```
VER ID2      ^ VERify status of input definition bank 2
"ID2 = 00000000"
```
Inputs are all active low

```
ID2=XXXXX111 ^ Set inputs 14, 15 and 16 active high (open)
VER ID2      ^ VERify status of input definition bank 2
"ID2 = 00000111"
```
CWL, CCWL and HOME inputs are active high (open)

CWL and CCWL should be defined as active open for safety reasons. This will prevent motion if switches or wiring are damaged.

```
VER ID4      ^ VERify status of input definition bank 4
"ID4 = 10000001"
```
Inputs 25 and 32 are active high

```
ID4=X1XXXXXX
Set input 26 active high.
```

Sample Program

Example An example of how 16 inputs assigned to two banks of **I[n]** could look:

I1, Input Block #1 Input Name	Input Number	I2, Input Block #2 Input Name	Input Number
Input 1	1	Input 9	9
Input 2	2	Input 10	10
Input 3	3	Input 11	11
Input 4	4	Input 12	12
Input 5	5	Input 13	13
Input 6	6	Input 14 (Home Limit)	14
Input 7	7	Input 15 (Clockwise Limit)	15
Input 8	8	Input 16 (Counter Clockwise Limit)	16

LIMIT ACTION LA

Description The limit action parameter is an eight (8) bit number used to define how the indexer will interpret a limit input when it is active. All eight bits must be entered or the command will not be accepted by the indexer. A "0" designates that the function is "INACTIVE". An "X" designates that the current definition is to be unchanged and the "1" designates that the function is "ACTIVE".

Format **LA =** *bbbbbbbb*

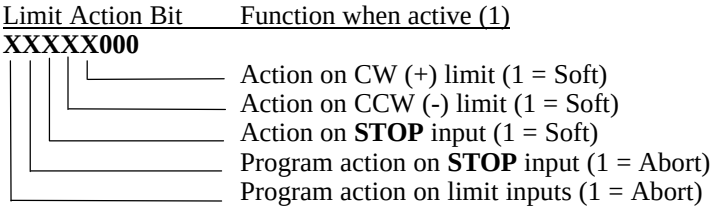
Mode Immediate/Stored

Range All eight bits must be entered as "1", "0" or "X".

Related Functions **SI,** CW and CCW limit inputs

Example `LA=00111000 ^ soft stop on limits`

Additional Description The following table defines the ICL interpretation:



LA=1XXXXXXX
Inactive "0" indicates that you will abort motion in the current direction and continue processing with the next command in your motion program if a limit is encountered. Your motion program will continue to execute skipping all motion commands in the direction while the input is active. Active "1" indicates that you will abort the program if a CW or CCW limit switch is encountered, the only exception is when the HOME command is being executed.

LA=X1XXXXXX
Inactive "0" indicates that you will abort motion in the current direction and continue processing with the next command in your motion program if a STOP input is encountered. Your motion program will continue to execute skipping all motion commands while the STOP input is active. Active "1" indicates that you will abort the program if a STOP input is encountered, preferred setting for safety reasons.

LA=XX1XXXXX
Inactive "0" indicates that you will stop the motor without decelerating when a STOP input is encountered. Active "1" indicates that a controlled deceleration will occur if a STOP input is encountered, (soft stop).

LA=XXX1XXXX
Inactive "0" indicates that you will stop the motor without decelerating when a CCW (-) input is encountered. Active "1" indicates that a controlled deceleration will occur when the CCW (-) input is encountered, (soft stop).

LA=XXXX1XXX
Inactive "0" indicates that you will stop the motor without decelerating when a CW (+) input is encountered. Active "1" indicates that a controlled deceleration will occur when the CW (+) input is encountered, (soft stop).

Programming Hints

A controlled deceleration from a high speed is usually faster and safer than a hard deceleration where the motor stalling will result, (use SOFT deceleration where possible).

STOP INPUT # SI

Description This parameter allows the user to select which programmable input is to be used as the stop input. The parameter **LA**, Limit Action is used to control the effect that the stop input will have when it is encountered. Polled at a 200µ sec interrupt.

The actions of bits 2 & 3 are shown below:

Limit Action Function **LA**

LA=X1XXXXXX
Inactive 0 indicates that you will continue processing in your motion program if a **STOP** input is encountered. Active 1 indicates that you will exit the program if a **STOP** input is encountered.

LA=XX1XXXXX
Inactive 0 indicates that you will stop the motor without decelerating when a **STOP** input is encountered. Active 1 indicates that a deceleration will occur if a **STOP** input is encountered.

Format **SI** = *nn*

Mode Immediate/Stored.

Range Device dependent.

Related Functions **LA**

Example Assign Input #3 as stop

```
SI=3
LA=01100000
^ Soft stop,exit program on active stop input
(TOP) IF 3 TOP
MOVE 1000
IF 3 TOP
.
.
.
SET 1, WAIT 50 RESET 1
IF 3 TOP
MOVE 1000
IF 3 TOP
SET 2
WAIT 50
RESET 2
IF 3 TOP
.
.
.
```

Hazard Note

If you allow the program to continue execution (LA = XOXXXXXX) when the SI input is active, you may experience machine or process malfunction. Your program will execute as normal, except no motion will occur. Note your outputs will set or reset as commanded. We recommend that you branch to a "safe location" within the program if the stop input becomes active. Remain there until the stop input becomes inactive.

The SI input is polled at a 200µ sec. interrupt. Poor wiring practices may result in false active input.

STATUS OF OUTPUTS O[n]

Description For purposes of simplicity, outputs are set up in banks of eight. Therefore if your indexer has 8 outputs the only parameter available would be **O** or **O1** if you prefer. If your system has 32 outputs then you would have four output parameters **O1** (1-8), **O2** (9-16), **O3** (17-24) and **O4** (25-32).

These are read only parameters that will update automatically based on the status of the outputs. A Status of "0" designates that the output is "INACTIVE" and a "1" designates that the output is "ACTIVE". The parameter **OD** can be used to define the default or "INACTIVE" state of the output as "HIGH" or "LOW". (Please refer to the user guide for your device to determine the actual type and number of outputs available.)

Format **O[n]** = *bbbbbbbb*

A "0" designates that the output is "INACTIVE"
A "1" designates that the output is "ACTIVE".

Mode Read only

Range Device dependent.

Related Functions **OD[n]**

Example VERIFY O1
"O1 = 10001100"
Shows that outputs 1, 5 and 6 are active.

Sample Program

OUTPUT DEFINITION OD[n]

Description For purposes of simplicity, outputs are set up in banks of eight. Therefore if your indexer has 8 outputs the only parameter available would be **OD** or **OD1** if you prefer. If your system has 32 outputs then you would have four output definition parameters **OD1** (1-8), **OD2** (9-16), **OD3** (17-24) and **OD4** (25-32).

The output definition parameter, **OD**, defines the default state of outputs as either "HIGH" or "LOW". For each bank of outputs on your ICL indexer there is a corresponding bank of output definition bits. A "0" designates that the output will be active "LOW". An "X" designates that the current definition is to be unchanged and the "1" designates that the output is to be active "HIGH". All eight bits of a bank of outputs must be entered or the command will not be accepted by the indexer. (Please refer to the user guide for your device to determine the actual type and number of outputs available.)

Format **OD[n]** = *bbbbbbbb*

Mode Immediate/Stored

Range Device dependent.

Related Functions **O[n], OF**

Example To have outputs #3 & #4 active "HIGH".
OD1=XX11XXXX

Sample Program

SOFT KEYS SK

Description Soft Keys are utilized as "software inputs" to control program flow similar to general purpose programmable inputs. The value of Soft Key is changed within a program by assigning **SK** its new value, and changed via the RS-232 input while a program is being executed. The Soft Key value is changed via RS-232 any time the indexer receives the reserved character **"!"** and a number. This powerful feature can be utilized in conjunction with ICL Commands to provide application specific information and define how the indexer will interface to its environment. SoftKeys are used in conjunction with the **IF**, **UNTIL** and **VERIFY** ICL Commands.

The **IF** (Condition) **ccc** command is used to provide for conditional branching within a motion program. If the **IF** condition evaluates "TRUE", then control of the program branches to label **ccc**. If the **IF** condition evaluates "FALSE" then control passes to the next line in the program.

Format

SK= *n*	Stored mode
!*n*	Immediate mode

Mode Immediate/Stored

Range 0 to 9.

Related Functions **IF, RESET, SET, UNTIL, VERIFY**

Example

```
SK = 8
```

Within a program

```
!8
```

Sent via RS-232 to set SK=8

Sample Program

```
(START) IF (SK=1) PGM1
IF (SK=2) PGM2
IF (SK=3) PGM3
IF (SK=4) NEW1
IF (SK=9) END
SK=0
WAIT 10
JUMP START
```

T-120 Data Terminal Key	T-120 Transmits via RS-232
<F1>	!1x08x08
<F2>	!2x08x08
<F3>	!3x08x08
<F4>	!4x08x08

Note the Hex commands "x08" following the Soft Keys Command. This is used to <backspace> the cursor two times to reposition the cursor on the screen. These functions are preprogrammed at the factory in the T-120 Data Terminals.

EXTENDED ASCII COMMANDS

Description An enhancement to all ICL indexers is the ability to receive via RS-232 while a program is executing. The list of commands shown below are received via RS-232 and updated in software every 10 milliseconds. These commands allow the user to Hex commands as if they were hard-wired IO, thus reducing wiring requirements in his application.

Format Shown in table below.

Mode Immediate/Stored

Range

Related Functions **HI, J, JM, JP, SK**

Example

Sample Program

T-120 Data Terminal Key	T-120 Transmits Via RS-232	Description
<F1>	!1x08x08	Set SK=1
<F2>	!2x08x08	Set SK=2
<F3>	!3x08x08	Set SK=3
<F4>	!4x08x08	Set SK=4
<HOLD>	x06	Software hold similar to the HI input.
<RESUME>	x05	Software resume, disables software hold.
<MODE>		Allows access to the Jog commands.
	x90x01	Jog CW
	x90x01	Jog CW
	x90x02	Jog CCW
	x90x02	Jog CCW
On release of a Jog key	x90x00	Commands the indexer to stop the jog.
AUTO	"~ AUTO"x0D	Global Stop, AUTOEXECUTE <return>
SETUP	"~ CALL SETUP"x0D	Global Stop, CALL SETUP <return>

Sample Program

```
EDIT SETUP
T=4, SC=20     ^ Must be first lines of program
G=1
V50 = 400*20   ^ Max move distance 20 revs
A=400, D=400, B=250,MR=400, H=MR*5, M=1000
ID1=00000000, ID2=00000000, OD1=00000000
J=1, JP=0, JM=0, $V10="                          "
^ v10 above is 20 spaces used to clear lines on t120
SEND -1|"1BS@" ^ Turn off scroll on t-120
SEND -1|"1BE"  ^ Clear screen
CURSOR 0 3
SEND -1 "API TEST EQPT CO"
CURSOR 3 0
SEND -1 "F1=HOME, F4 = QUIT"
SEND -1|"1BPT" ^ Make led1 flash
SEND -1|"1BPQ" ^ Make led4 flash
SK=0
(POLL14) WAIT 50
IF (SK=4) DONE
IF (SK=1) NEXT
JUMP POLL14
(NEXT)SEND -1|"1BPA" ^ Led4 off
SEND -1|"1BPD" ^ Led1 off
CURSOR 3 0
SEND -1 $V10
CURSOR 2 5
SEND -1 "HOMING"
- HOME
J=50   ^ 1/4 rev/sec
SEND -1|"1BE" ^ Clear screen
SEND -1|"07"  ^ Cause buzzer to beep
CURSOR 0 0
SEND -1 " JOG TO A POSITION"
CURSOR 1 0
SEND -1 "PRESS <F-KEY>"
CURSOR 2 0
SEND -1 "TO RECORD POSITION"
CURSOR 3 0
SEND -1 "ONE   TWO   THREE  EXIT"
(POLL1234) SK=0
WAIT 50
IF (P>V50) ERROR
IF (SK=1) ONE
IF (SK=2) TWO
IF (SK=3) THREE
IF (SK=4) DONE
JUMP POLL1234
(ONE) V1=P
JUMP POLL1234
(TWO) V2=P
JUMP POLL1234
(THREE) V3=P
JUMP POLL1234
(DONE)  J=1
SEND -1|"1BE" ^ Clear screen
```

```
SEND -1|"07"  ^ Cause buzzer to beep
CURSOR 0 3
SEND -1 "API TEST EQPT CO"
CURSOR 2 0
SEND -1 "PRESS START TO BEGIN"
QUIT
(ERROR) J=1
SEND -1|"1BE" ^ Clear screen
CURSOR 0 0
(E1) SEND -1 "***ERROR***"
SEND -1|"07"  ^ Cause buzzer to beep
WAIT 100
JUMP E1, QUIT
~

EDIT MAIN
J=1, T=4
SC=20 ^ Must be first lines of program
SEND -1|"1BS@" ^ Turn off scroll on t-120
SEND -1|"1BE"  ^ Clear screen
SEND -1|"07"   ^ Cause buzzer to beep
CURSOR 0 3
SEND -1 "API TEST EQPT CO"
CURSOR 1 5
SEND -1 "TEST STATION"
CURSOR 3 0
SEND -1 "ONE  TWO  THREE  NEW"
(POLL) SK=0
WAIT 50
IF (SK=1) ONE
IF (SK=2) TWO
IF (SK=3) THREE
IF (SK=4) EXIT
JUMP POLL
(ONE) POS V1
SEND -1|"07" ^ Cause buzzer to beep
JUMP POLL
(TWO) POS V2
SEND -1|"07" ^ Cause buzzer to beep
JUMP POLL
(THREE) POS V3
SEND -1|"07" ^ Cause buzzer to beep
JUMP POLL
(EXIT)
SEND -1|"07" ^ Cause buzzer to beep
SEND -1|"07" ^ Cause buzzer to beep
SEND -1|"07" ^ Cause buzzer to beep
CALL SETUP
QUIT

~
```

After loading programs you must enter:

```
V1=0, V2=0, V3=0
```

ICL COMMAND DEFINITIONS

Within the overall structure of an ICL System, ICL Commands allow the user to tell the indexer "WHAT TO DO". Each of the commands listed within this section will perform a specific function when issued to the indexer. Since the Intelli-Command Language (ICL) utilizes English-like wording, the task of learning and remembering the function of each command has been greatly simplified.

The alphabetical command listing includes an indication as to whether the command can be executed from command mode (Immediate), stored for repetitive execution as part of a motion program (Stored), or both. A shortened version of the ICL command can be used by typing the number of characters that makes the command unique as is denoted by the upper case characters in the command description. **To ensure command integrity within motion programs with future firm-ware updates, you should use a minimum of three (3) characters where applicable.**

Command Name

Description Description of ICL Command.

Format The Sample format for the ICL Command.

<>	Characters enclosed in angle brackets designate keys or key strokes that can be found on the host keyboard.
[ ]	Optional characters are enclosed in brackets.
b	Bite
c	Denotes alphanumeric characters. Note labels must begin with ALPHA characters. Excluded characters are {<space>, <colon>, <comma> and <!>}.
f	Denotes an ICL flag name (F0 to F64).
fff	Denotes file name
n	Denotes numeric characters
p	Denotes an ICL parameter name
v	Denotes an ICL variable name (V0 to V99).
,	Command delimiter, has the same effect as a <space>. Each command must be separated from the next by a delimiter.

Mode

Immediate	Command is executed or changed in the immediate mode from user's terminal.
Stored	Command may be executed or changed within a program.

Range Denotes the minimum and maximum values allowed within a command.

Related Functions Listing of related parameters and commands.

Example Sample ICL command with comments.

```
A=1000 ^ Sets acceleration to 1 second
```

Example of ICL code.

Comments in program. Shown in lower case within this manual for purposes of readability.

The "Caret" denotes the start of a comment line. Note the "Caret" is followed by a blank space then the comment string.

Programming Hint

The "!" (exclamations point) and ":" (colon) are reserved ASCII characters and must not/cannot be used in comments.

ABOrt

Description When abort is encountered in a motion program, the ICL System will stop pulses to the drive immediately without decelerating and the program execution will be terminated. All stored parameter values will remain valid, with the possible exception of motor position parameter **P**. The motors resulting position may be different from controller stored values if the motor was running above its stop/start speed range when this command was encountered. In the immediate mode the <ESC>, <~> or **ABORT** commands may be used to abort a **GO**, **MOVE**, **POSITION**, **RUN** or **SLEW** command.

Format **ABOrt**

Mode Immediate/Stored

Range

Related Functions **GO, HOME, MOVE, POS, RUN, SLEW**

Sample Program

```
^ MAIN PROGRAM
(TOP) IF 1 EXTEND
IF 2 RETRACT
IF 13 ERROR1
JUMP TOP
(EXTEND) MOVE 50000
JUMP TOP
(RETRACT) MOVE -50000
JUMP TOP
(ERROR1) SEND -1 "**INPUT 13 ACTIVE, PROGRAM ABORTED**"
ABORT
QUIT
```

AUTostart

Description This command begins execution of the auto-start program number defined by parameter **G**. To obtain the program number type the **DIRectory** command, find the name of the program you wish to run as the **AUTostart** program, and read its assigned program number. The program number may be from 0 to 87 as shown in the directory listing and may be loaded into parameter **G** by using the ENTER or <=> commands. If parameter **G** is defined as "-1", **G** = **-1**, then no auto-start program will be executed. Note that the auto-start program begins execution when power is first applied to the indexer.

The **AUTostart** command allows the user to begin execution of the auto-start program without powering the unit down. This command may be used in place of the **CALL ccc** command to start execution of the defined auto-start program.

Format **AUTostart**

Mode Immediate

Range

Related Functions **G, DIRECTORY, CALL**

Sample Program

> ***Programming Hint***
> Do **not** assign an **AUTOSTART** Program, **G=n**, until you have completely debugged your programs. This will prevent possible injury or a programming loops

BOOT

Description This command completely initializes the indexer, all user programs are erased and all ICL parameters and variables are reset to their default values, (see "ICL Parameters and Variables"). This command must be typed in full and the user will be prompted for verification to prevent an unintentional boot and the loss of data. An example of this message is displayed below:

Format **BOOT**

Mode Immediate

Range

Related Functions

Sample Program BOOT

```
#99 *** WARNING ***
      BOOT WILL ERASE ALL PROGRAMS AND PARAMETERS
      PROCEED? (Y/N)? -
```

CALl ccc

Description Begins execution of a program named *ccc* stored in RAM, where *ccc* is an alphanumeric label from 1 to 8 characters long. A directory of program names can be determined by issuing the **DIRECTORY** command.

If executed from within a calling program, control will be passed to the first line of the called program (subroutine) and will be passed back to the next line of the calling program when a **QUIT** or end of file is encountered.

When utilizing a string variable as the name of the called program the variable must be preceded with a <$>, (**CALL** $V45). *This command can be nested to four (4) levels.*

Format **CALl** *ccc*

Mode Immediate/Stored

Range

Related Functions

Sample Program

```
CALL START
      ^ Instructs the ICL system to call program named
      ^ "START" located in RAM.
$V10 = "PART5"
CALL $V10
      ^ Instructs the ICL system to call program name
      ^ Associated with string variable $V10
      ^ "PART5" located in RAM.
```

Programming Hints

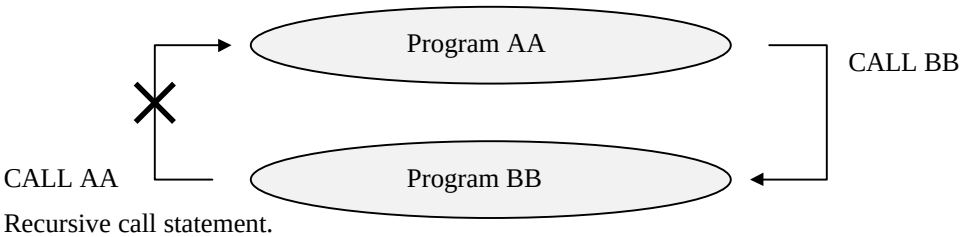
Error Message #20 CALL STACK OVERFLOW

The AUTOSTART program is a first level CALL command.

When using an Encoder Stall Program the maximum number of nested CALL statements is reduced to three. The Stall Program becomes the fourth CALL routine.

Avoid recursive CALL statements / programming.

Program A	CALL PG2
Program PG2	CALL A



Recursive call statement.

CLEAR VARIABLES CLEar [v]

Description Used to clear variables stored in RAM. If the variable name *v* is not supplied, the ICL System will clear all variables from *V0* to *V99*. Variables are dynamically allotted, when cleared the user storage space is increased.

Format **CLEar *[v]***

Mode Immediate/Stored

Range 0 - 99

Related Functions **SHOW, VERIFY**

Example

```
CLEAR V99 ^ Clears variable V99 from RAM
CLEAR     ^ Clears all variables from RAM
```

To prevent an unintentional loss of variables the following message is displayed.

```
#96 *** WARNING ***
CLEAR WILL ERASE ALL VARIABLES
PROCEED (Y/N)
```

CONtinue

Description The **CONTINUE** command instructs the ICL System to execute its instructions in the *continuous processing mode*. This mode allows other non-motion commands such as setting outputs and checking inputs to be executed while motion is occurring.

If you desire to wait until the current command is completed before executing the next command, the complement **PAUSE** command would be issued.

Format **CONtinue**

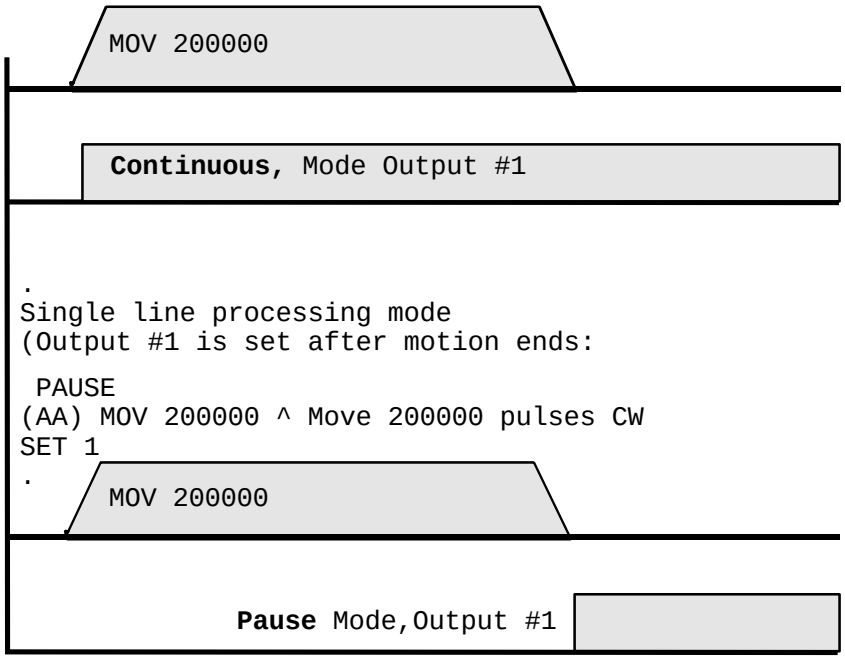
Mode Immediate/Stored

Range

Related Functions **MOVE, OF(Other Functions bit #8), PAUSE, POSITION**

Example Continuous processing mode:
(Output #1 is set after motion begins)

```
CONTINUE
(AA) MOV 200000 ^ Move 200000 pulses CW
SET 1
```




```
.
Single line processing mode
(Output #1 is set after motion ends:
 PAUSE
(AA) MOV 200000 ^ Move 200000 pulses CW
SET 1
.
```


Programming Hints

The PAUse program execution mode will remain in effect until changed by a CONtinuous mode command. The program execution mode command should appear only once in a program unless you are switching between modes.

The continuous motion commands RUN and SLEW will temporarily change the execution mode to CONtinuous to allow execution of additional commands until a STOP command is executed.

Bit eight of the parameter OF is also used to toggle between Pause and Continuous modes.

COPy “old” “new”

Description This command will allow the user to make a copy of an existing program to a new program name without the need for re-keying the entire program. The entire program named “old” is copied to a new location on the directory under the name “new”. The program names “old” and “new” are alphanumeric characters up to 8 characters in length. The ICL system will provide a warning if the “new” program name already exists so that you will not over-write files.

Format **COPy fff fff**

Mode Immediate

Range

Related Functions **EDIT, DELETE, LIST, PASSWORD, RENAME**

Example

```
COPY FIRST SECOND
COPY TEST MAIN
```

CURsor

Description This command allows the user to select the row and column for text to be printed to the host’s screen. **Used only with the T-120, T-121 and T-122 series data terminals sold by API.**

Format **CURsor** *nn nn*

Mode Stored

Range

Related Functions **T, SC, SEND -1, PROMPT**

Example

```
CURsor 2 0  ^ Locates cursor at row 2 column 0
SEND -1 "PROGRAM INITIATING"
            ^ Statement will be sent to host screen
            ^ starting at row 2 column 0
CURsor 3 0  ^ Locates cursor at row 3 column 0
PROMPT "ENTER MOVE DISTANCE" V1
            ^ Statement will be sent to host screen
            ^ starting at row 3 column 0
```

DELete ccc

Description Erases an entire program named *ccc* from your system Random Access Memory (RAM), where *ccc* is an alphanumeric label from 1 to 8 characters long. The program number associated with the deleted program name will not be reassigned until a new program name is created.

Format **DELete** *ccc*

Mode Immediate

Range

Related Functions **COPY, EDIT, LIST, PASSWORD, RENAME**

Example

```
DELETE PART1 - Deletes the program "PART1" from RAM.
```

DIRectory

Description Results in a listing of the program names and their associated numbers, stored in RAM, to be transmitted to the host device.

Format **DIRectory**

Mode Immediate

Range The program numbers range from 00 to 87.

Related Functions **G, SC, T, W, AUTOEXEC, CALL, LIST**

Sample Usage **DIR**ectory (ICL Parameter "SC" is set to 80 characters.)

```
0 | SETUP    1 | PROG1     2 | PROG2      3 | TEST
4 | SAMPLE1  5 | SAMPLE2   6 | SAMPLE3    7 | TEST1
8 | TEST2    9 | TEST3
                                82| PN1011     83| PN1012
84| PN1013   85| MACHINE1  86| MACHINE2   87| LAST
```

Programming Hints

Screen characters per line parameter "SC", is used to define the number of characters available per line on the host's display before a <Return> is transmitted. For a screen width of 40 characters, the ICL parameter "SC", is set to 40 characters, type **SC = 40 <Return>.** Parameter "SC" can be a value from 20 to 80 characters depending on the host's display width.

The number of lines to transmit parameter "T", is used to define the number of lines sent to the host device before pausing. The **T = 10 <Return>** command instructs the indexer to transmit 10 lines and pause until any character is typed. The listing may be terminated during a pause by the <ESC> command.

Sample Usage **DIR**ectory (ICL Parameter "SC" is set to 40 characters.)

```
0|SETUP      1|PROG1
2|PROG2      3|TEST
4|SAMPLE1    5|SAMPLE2
6|SAMPLE3    .......
```

DUMp fff n

Description Transmits the program named *fff* to the host device. *n* identifies the format of the transmission as follows. Assume program name to be *PART1* in the following

Format **DUMp** *fff n*

Mode Immediate

Range

Related Functions **EDIT, COPY, DELETE, LIST, PASSWORD, RENAME**

Example
```
DUMP PART1  0
```

The program named PART1 will be sent to the host device in a sequential format, program lines only, with no line numbers or memory addresses. This is the format recommended when down-loading programs to disk for purposes of backing up complex programs.

```
A=1000
D=500
+
MOV 12800
```

Example
```
DUMP PART1  1
```

Program lines are preceded by the line number.

```
0 A=1000
1 D=500
2 +
3 MOV 12800
```

Example
```
DUMP PART1  2
```

Program lines are preceded by the starting memory address of that program line.

```
4612 A=1000
4618 D=500
4623 +
4625 MOV 12800
```

Example
```
DUMP PART1 3
```

Program lines are preceded by both the number and the starting memory address of each program line.

```
0 4612 A=1000
1 4618 D=500
2 4623 +
3 4625 MOV 12800
```

Programming Hints

The number of lines to transmit parameter "T", is used to define the number of lines sent to the host device before pausing. The **T = 10 <Return>** command instructs the indexer to transmit 10 lines and pause until any character is typed. *The listing may be terminated during a pause by the <ESC> command.*

ECHo [ON/OFF]

Description This command allows the user to *toggle or set* the **ECHO** feature of the ICL indexer. When toggled ON, the indexer will echo characters to the host device, when OFF no characters will be echoed. This feature can be utilized to reduce the communication time between the indexer and the host device.

Format **ECHO ON, ECHO OFF**

Mode Immediate/Stored

Range On or Off

Related Functions **OF** (Other Functions, bit 7)

Example

```
ECHO     ^ Toggles the echo feature
ECHO ON  ^ Sets the echo feature on
ECHO OFF ^ Sets the echo feature off echo on
```

> ***Programming Hints***
> If your application requires the user or operator to input data, the command ECHO ON must executed so that the user can observe his data when entered.

EDIt ccc

Description This command puts the indexer into edit mode for the creating or editing of program named *ccc*. The program name *ccc* is either the name of an existing program to be edited, or the name of a new program that is created if it does not exist.

To exit from edit mode and return to command mode, use the **<ESC>** or <~> key on the host device.

Format **EDIt ccc**

Mode Immediate

Range

Related Functions **COPY, DELETE, PASSWORD, RENAME**

Example

> ***Programming Hints***
> See ICL Editor Commands on the next page.

ICL Editor Commands

Below is a listing of editor control characters available when editing an ICL program in the immediate mode, see **EDIT** *ccc* command.

For examples of creating and editing motion programs see the section on "CREATING PROGRAMS".

<Backspace> or <CTRL-H> Remove the preceding character from the edit line. This key has the same function in the command mode.

<ESC> or <CTRL-[> "Escape Key" on the host keyboard. **The <ESC> key is used to exit the ICL editor.** The last edit line must be entered into a program by pressing the <Return> key, before the <ESC> key will be recognized. Note: If the user's host device does not have an <ESC> then the <CTRL-[>, (control and left bracket sign) may be substituted.

<CTRL-X> Deletes the currently displayed program line. The editor then displays the next sequential program line.

<CTRL-J> Moves down one program line and displays the program line. The line number prompt character(s) will be echoed on the next display line for input of the desired changes.

<CTRL-K> Moves up one program line and displays the program line. The line number prompt character(s) will be echoed on the next display line for input of the desired changes.

<CTRL-N> Insert a single new program line before the current line displayed. The line number prompt character(s) will be echoed as a blank line. All succeeding program lines will be moved ahead by one line number.

<Return> Stores the current edit line within the program and moves down one program line.

=n Move to line number "n" and display its contents. ***Note that delimiters are not used in this editor command.*** The line number prompt character(s) will be echoed on the next display line for input of the desired changes.

ENTer p n

Description This command allows the user to change an ICL parameter or variable. *p* is the name of the parameter or variable you wish to change and *n* is the new value to be assigned. This command may be used in immediate mode or within a program. The value of a parameter can be checked through the use of the **VERIFY** *p* command.

(An alternate method is to use the "=" command to set values).

Format **ENTer** *p n*

Mode Immediate/Stored

Range

Related Functions All ICL parameters and variables.

Example

```
ENTER B 1000 ^ Set the base speed to 1000 PPS
ENT J 500    ^ Set the jog speed to 500 PPS
M=25000   ^ pertains the same function as ENTER M 25000
ENT B V1
B=V1
$V2 = "THIS IS A STRING"
ENTER $V2 "THIS IS A STRING"
```

Programming Hints

Use of the ENTER command will consume more RAM memory than the alternate form of this command. Any parameter or variable may be changed through use of the <=> command ("parameter" = "new value").

The command **A=1000** takes 4 bites less than the command **ENT A 1000.**

EUNits

Description This command configures the indexer to utilize encoder units for **MOVE** and **POSITION** commands.

The parameter **ER** is the number of encoder pulses generated during one revolution of the motor shaft. This value is also used for encoder related commands. The number is post-quadrature (the number of encoder lines multiplied by 4). A 1000 line encoder produces 4000 pulses per revolution. Hence, **ER**=4000 is entered. The parameter **UA**, Units Active is *1* when encoder units are active.

Format **EUNits**

Mode Immediate/Stored

Range

Related Functions **ER, MR, UR, MOVE, MUNITS, POSITION, UUNITS**

Example

Programming Hints

If the command EUNITS is executed the motion generated from the MOVE and POSITION commands is ***scaled*** by the following formula.

EUNIT MOVE 1000

$$1000 \text{ (Encoder Units)} * \frac{\text{MR (Motor Resolution)}}{\text{ER (Encoder Resolution)}}$$

or 1000*MR/ER (motor step pulses)

EUNITS does ***not*** imply that you have an encoder.

EUNITS does ***not*** enable encoder position maintenance or stall detection.

EXPert

Description When executed, the controller will send only the error message "number" instead of the full message text to the host device. The corresponding command **NOVICE**, which is the default during setup, results in transmitting the full message text to the host device.

Format **EXPert**

Mode Immediate/Stored

Range

Related Functions **System Error Messages**

Example

```
Expert ^ Sets message mode to send number only
Move 1000
Report ^ Send number only to host (#24)

Novice ^ Sets message mode to full text
Move 1000
Report ^ Send full text to host (#24 MOTOR IDLE)
```

GO

Description Causes the motor to move the distance specified by, number of motor pulses to index parameter **N**, utilizing the parameters; base speed **B**, maximum velocity **M**, acceleration time **A**, deceleration time **D**, and the currently set direction. If **N** is not set prior to use of this command, then **N** will be the absolute value of the last index distance. **N** is always a *positive* integer.

Format **GO**

Mode Immediate/Stored

Range

Related Functions **A, B, D, H, M, N**

Example

```
PAUSE
N=25600 ^ Enter an index distance of 25600 motor pulses
+       ^ Set the direction to CW
Go      ^ index the distance defined by parameter "N"
WAIT 200 ^ Wait for 2 seconds
-        ^ Set the direction to CCW
Go ^ Index the distance 25600 defined by parameter "N"
```

Programming Hints

NOTE: If a series of motion commands are issued in the immediate command mode while the ICL System is in the continuous processing mode, an error message will be displayed. The PAUSE command should be utilized to prevent an error since only one motion command can be executed at a time.

HELp

Description Results in a summary display on the host device of all the commands available within the Intelli-Command Language (ICL) set. A sample of this display is shown below where screen width parameter **SC**, equals 80 and the lines to display before pausing parameter **T**, equals 24:

Format **HELp**

Mode Immediate

Range This command may not be available in customized versions of firmware.

Related Functions

```
X> SC=80 T=24 HELP

ABORT                 AUTOSTART     BOOT
CALL ccc              CLEAR p       CONTINUE
DELETE ccc            DIRECTORY     DUMP ccc n
ECHO                  EDIT ccc      ENTER p n
EUNITS                EXPERT        GO
HOME                  IF [-]n ccc   IO [-n/n]
JUMP ccc              LIST ccc      LOOP n ccc
MOVE [-]n             MUNITS        NOVICE
PASSWORD "mmmmmm"     PAUSE         POSITION [-]n
PROMPT "mmmmm.."p     QUIT          REPORT
RESET n               RUN           SEND n "mmmmmm.."
SET n                 SHOW          SLEW
STATUS                STOP          SYNC n
TEST "mmmmm"          TIME          TRACE n
UNTIL [-]n ccc        UUNITS        VERIFY p
WAIT n                +             -
:c                    *             %
```

HOMe

Description This command initiates a sequence of moves to position the motor at a user defined home position, (home limit switch or home switch and index channel when bit #4 of the encoder function is enabled **EF**=XXX1XXXX).

The home position is defined by locating the home switch at the selected position. The home switch output is then hard wired to the indexer Home limit input.

Format **HOMe**

Mode Immediate/Stored

Range

Related Functions **+, -, A, B, D, E, H, M, E, EF, LA, ID2, P, UP**

Additional Description NOTE, to avoid damage to your system, first verify that the CWL and CCWL switches are wired correctly to their inputs, since these switch inputs establish the outer limits of motion. Also verify that the parameter **ID2**, Input Definition for bank #2, is properly set to define the logic as active "high" or active "low".

The user may select hard or soft limits for the end of travel limit inputs, parameter **LA**, Limit Action. Hard limits are selected when the user requires the motor to immediately stop when a limit switch is encountered. The motor position parameter **P** may lose sync due to the resulting abrupt stop associated with hard limits. Soft limits are selected when the user requires the motor have a controlled deceleration to stop utilizing the current deceleration time, **D**, when a limit switch is encountered.

The limit action parameter, LA is an eight (8) bit number used to define how the indexer will interpret a limit input when it is active.

The following table defines the ICL interpretation:

Limit Action Bit	Function when active (1)
XXXXX000	
	Action on CW (+) limit (1 = Soft)
	Action on CCW (-) limit (1 = Soft)
	Action on STOP input (1 = Soft)
	Program action on STOP input (1 = Abort)
	Program action on limit inputs (1 = Abort)

LA=1XXXXXXX
Inactive "0" indicates that you will abort motion in the current direction and continue processing with the next command in your motion program if a limit is encountered. Active "1" indicates that you will abort the program if a limit switch is encountered.

LA=X1XXXXXX
Inactive "0" indicates that you will abort motion in the current direction and continue processing with the next command in your motion program if a STOP input is encountered. Active "1" indicates that you will abort the program if a STOP input is encountered.

LA=XX1XXXXX
Inactive "0" indicates that you will stop the motor without decelerating when a **STOP** input is encountered. Active "1" indicates that a controlled deceleration will occur if a STOP input is encountered, (soft stop).

LA=XXX1XXXX
Inactive "0" indicates that you will stop the motor without decelerating when a CCW
(-) input is encountered. Active "1" indicates that a controlled deceleration will occur when the CCW (-) input is encountered, (soft stop).

LA=XXXX1XXX
Inactive "0" indicates that you will stop the motor without decelerating when a CW (+) input is encountered. Active "1" indicates that a controlled deceleration will occur when the CW (+) input is encountered, (soft stop).

When the **HOME** command is issued, the motor will accelerate to the maximum velocity parameter **M**, in the currently set direction in search of the home limit switch. If the home limit is encountered, the motor will then decelerate to a stop, reverse direction, and move at the base speed parameter **B**, back to the leading edge of the home limit switch and stop. The position parameter **P**, encoder position parameter **E** and user position parameter **UP** will then be set to "0".

If soft limit switches are encountered during a home limit search (ie. - the initial search direction was wrong), the motor will decelerate to a stop, reverse direction, accelerate to the maximum velocity parameter **M**, and continue the search in that direction. If a second limit switch is encountered before the home limit switch (ie. - The home limit switch is not connected), the motor will be decelerated to a stop and the error message **#23 HOME NOT FOUND** will be returned and program execution will be aborted.

If hard limit switches are encountered during a home limit search (ie. - The initial search direction was wrong), the motor will immediately stop, reverse direction, accelerate to the maximum velocity parameter **M**, and continue the search in that direction. If a second limit switch is encountered before the home limit switch (ie. - The home limit switch is not connected), the motor immediately stop and the error message **#23 HOME NOT FOUND** will be returned and program execution will be aborted. This procedure is the same as listed above for soft limits except the motor will be stopped immediately without deceleration when the limits are encountered.

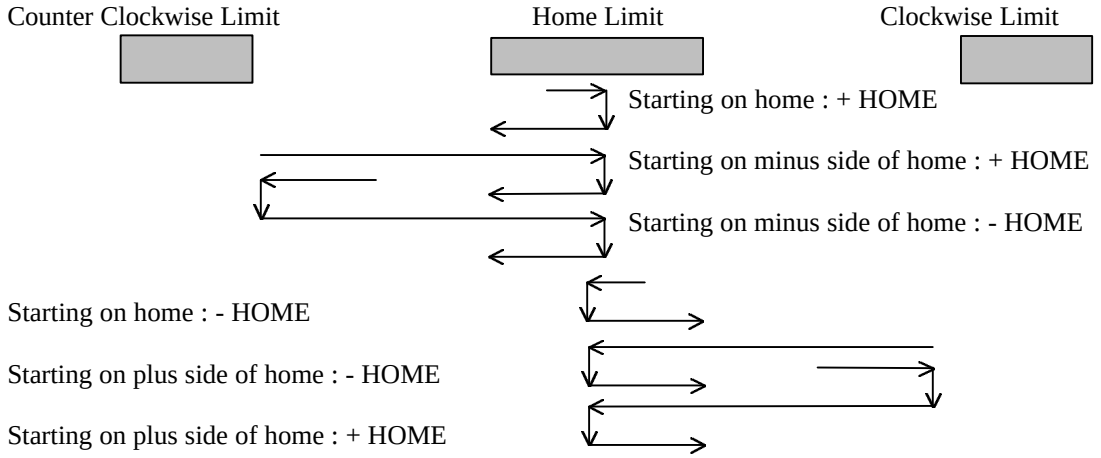
If the encoder function bit #4 is active, **EF=XXX1XXXX**, during the search for **HOME**, the indexer will continue over the home switch once it is found until the edge of the index or z-channel of the encoder. The position parameter **P**, encoder position parameter **E** and user position parameter **UP** will then be set to "0".

Graphic examples of the search for the **HOME** limit are shown on the next page along with examples of the search for the **HOME** and the **Z-CHANNEL**, (encoder index channel) with bit #4 of the encoder function enabled.

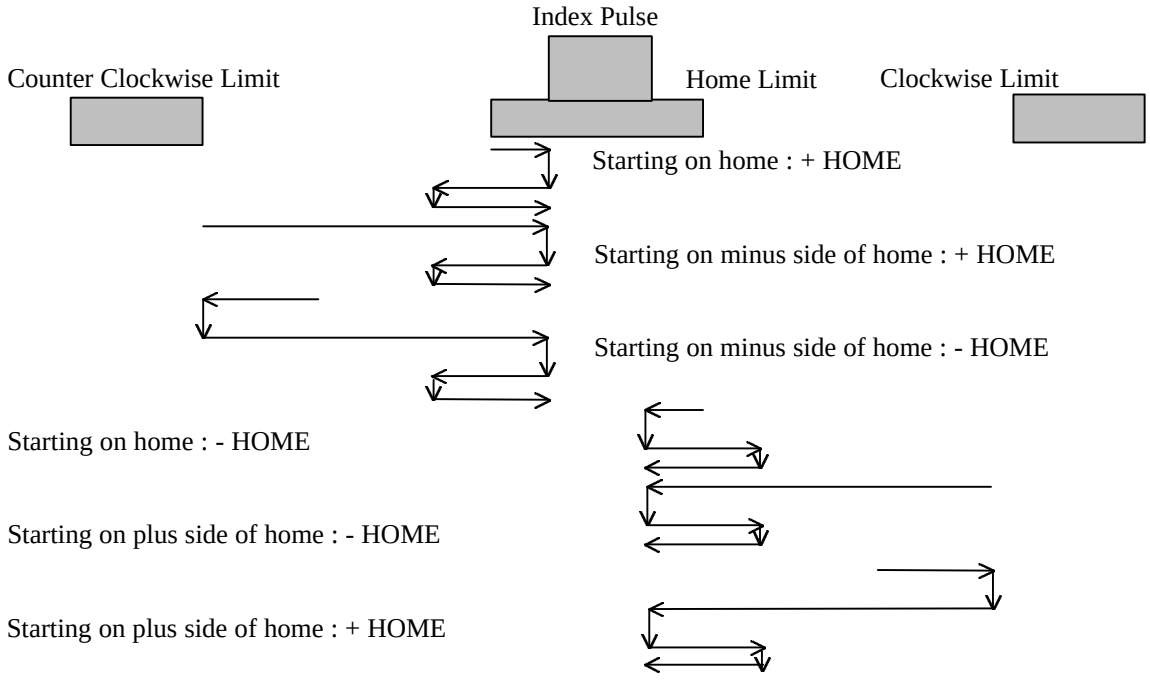
Programming Hints
Precede the HOME command with a + or - command to select the initial search direction.

> ***Hazard Note***
> Reduce the system speed "M" prior to issuing the HOME command.
>
> Encoder functions for position maintenance and stall detection are disabled during execution of the home command.

Graphic representation of the search for home, (encoder function OFF, "EF=00000000").



Graphic representation of the search for home, (encoder function ON, "EF=XXX1XXXX").



IF *condition* ccc

Description This command is used to provide for conditional branching within a motion program. If the **IF** condition evaluates "TRUE", then control of the program branches to label **ccc**. If the **IF** condition evaluates "FALSE" then control passes to the next line in the program.

When testing flags and/or compares parenthesis must be used to control the ICL interpretation of the command. When testing only inputs and outputs parenthesis are not required.

Format **IF *ccc***

Mode Immediate/Stored

Range

Related Functions **<>, =, >, <, >=, <=,. OR, AND, UNTIL**

Example

```
IF (V1 > V2) LABEL1
IF (F1) LABEL2
IF ((V1 > V2) AND (V4 > V5) AND F1\ OR F2) LABEL3
```

Sample Program Evaluation conditions:

-Input, Output and Flag status testing for "Active" or "Inactive" (Backslash (\) after an I/O or Flag checks for an inactive condition)

Programming Hints

The ICL system cannot perform string compares. The ICL system will not allow evaluation of input/outputs and any conditional. You must split your decision making into multiple statements.

Label ccc must start with an alpha character and should be two (2) characters in length.

IF [-]*n*[\] *ccc* (for I/O testing)

Description Tests the status of either input(s) or output(s) defined by number *n* and conditionally branches to the line label ***ccc*** within the program on a "TRUE" condition. When testing inputs and outputs parenthesis are not required. A backslash "\" after the I/O compares for an inactive condition. The **IF** statement will result in branching to label ***ccc*** if comparisons are "TRUE". The **IF** statement will cause the next program line to be executed if the comparison is "FALSE". Positive numbers represent inputs while negative numbers represent outputs, (see your indexer User Guide for the I/O name and its corresponding number). **Restriction: Label *ccc* must start with an ALPHA character.**

> ***Programming Hints***
> The parameters ID and OD are used to define the default states of the INPUT DEFINITION and OUTPUT DEFINITION respectively

Example

```
(A) IF 1 2 A
      ^ If both inputs 1 & 2 are active branch to label A
IF 6 ACTIVE6       ^ Test input 6, if active branch to
                   ^ label ACTIVE6
QUIT               ^ If input 6 is inactive quit program
(ACTIVE6) HOME     ^ Execute home command
IF -1 -2 NEXT
                   ^ Check outputs 1 & 2 to see if they
                   ^ are active
```

IF Fn[\] ccc (for testing ICL Flags)

Description Tests the status of ICL flags defined by number n and conditionally branches to the line label ccc within the program on a "TRUE" condition. A backslash "\" after the Flag compares for a "FALSE" condition. The IF statement will result in branching to label ccc if comparisons are "TRUE". The IF statement will cause the next program line to be executed if the comparison is "FALSE". Restriction: Label ccc must start with an ALPHA character.

When testing flags and/or compares parenthesis must be used to control the ICL interpretation of the command.

> ***Programming Hints***
> ICL flag F0 is dedicated to the ICL System. Flag F0 is active while moving and is read only and cannot be set or reset.

Example

```
(A) IF ((F1) AND (F2)) A
      ^ If ICL flags 1 & 2 are TRUE branch to label A
IF (F6) ACTIVE6
      ^ Test flag 6, if active branch to label ACTIVE6
QUIT  ^ If flag 6 is inactive quit program
(ACTIVE6) HOME     ^ Execute home command
```

IF (Vn comparison operator Vn) ccc (for testing ICL Variables

Description Does a comparison of the ICL variables defined by number n and conditionally branch to the line label ccc within the program on a "true" condition. The IF statement will result in branching to label ccc if comparisons are "TRUE". The IF statement will cause the next program line to be executed if the comparison is "FALSE".

When testing flags and/or compares parenthesis must be used to control the ICL interpretation of the command.

Example

```
IF (V1 > V2) A          ^ If true branch to label A
IF ((V3 > V4) AND (V3 <> V5)) B
                        ^ If True go to label B
```

IF (SK = n) ccc (for testing Soft Keys

Description Does a comparison of the ICL software inputs called Soft Keys used to control program flow similar to general purpose programmable inputs.

Example

```
SK = 0
(POLL) IF (SK = 1) FIRST
IF (SK = 2) SECOND
IF (SK = 3) UP
IF (SK = 4) DOWN
WAIT 10
JUMP POLL
(FIRST)
.
.
.
(SECOND)
.
.
.
```

IO [-n/n

Description This command displays the status, an input or output and returns the result of the test to the host device, where *n* is the number of the input or *-n* is the output to be tested. The indexer will report **1** if **ACTIVE** and **0** if **INACTIVE**. ***NOTE: The parameters ID and OD are used to define the default states of the inputs and outputs respectively, (see definition of parameters ID and OD).***

Format **IO [***-n/n***]**

Mode Immediate/Stored

Range 0 to 16 Indexer Dependent

Related Functions **ID1, ID2, OD1**

Example Positive numbers represent inputs.

```
IO 1        ^ Display status of input 1
IO 9        ^ Display status of input 9
```

Host displays:

```
"IO 1 = 0"
"IO 9 = 0"
```

Negative numbers represent outputs.

```
IO -1       ^ Display status of output 1
```

Host displays:

```
"IO -1 = 0"
```

If *n* is not specified then the default will be to list all I/O (inputs and outputs).

```
IO          ^ Display status of inputs and outputs
```

Host displays:

```
 "I1 = 00000000  I2 = 00000000   O1 = 11000000"
```

> ***Programming Hints***
>
> Refer to your indexer User Guide for the number and location of inputs and outputs available.

JUMp ccc

Description The **JUMP** command is used to unconditionally branch to line label *ccc* within the motion program. The **JUMP** command can be used to move backward or forward within a program and to exit a loop. Restriction: Label *ccc* must start with an ALPHA character and should be two characters in length.

Format **JUMp** *ccc*

Mode Stored

Range

Related Functions

Example

```
PAUSE
$V3="NEXTA"
(A) UNTIL 1 A
      ^ Test input 1, if inactive branch to label A
(B) SET 4 MOVE 5000 ^ Set output 4 and move 5000 pulses
RESET 4 WAIT 100    ^ Reset output 4 and wait 1 second
IF 2 A ^ Test input 2, if active branch to label A
JUMP NEXTA          ^ Go to label NEXTA
(C) MOVE 30000
JUMP $V3            ^ Go to label NEXTA
.
.
QUIT                ^ If input 6 is inactive quit program
(NEXTA) HOME        ^ Execute home command
JUMP A              ^ Go to label A.
```

LISt ccc

Description A listing of program named *ccc* will be transmitted to the host device in the same format as **DUMP filename 3**. Program names can be determined by using the **DIRECTORY** command. The returned listing will be in the same format as it was created in the editor preceded by both a line number and the starting memory location.

Format **LISt** *ccc*

Mode Immediate

Range

Related Functions **T, DIRECTORY, DUMP “filename”, PASSWORD**

Example

Programming Hints

The parameter T is used to define the number of lines displayed before the display is paused. The user can thus view convenient size blocks of his program. The <ESC> key will terminate this command at the pause points.

LOOp n ccc

Description Program execution jumps to label *ccc* in that program *n* number of times. The **LOOP** command must always follow the label *ccc* location in the program, forward loops are not allowed. See **JUMP** command for forward branching. If *n* is *0*, then the looping condition becomes continuous. The **LOOP** command may be nested to four (4) levels maximum. **Restriction: Label ccc must start with an ALPHA character.**

Format **LOOp** *n ccc*

Mode Stored

Range

Related Functions **JUMP**

Example

```
(A0) M 200 ^ Move 200 pulses CW
LOOP 4 A0  ^ Loop 4 times to label A0
```

> ***Programming Hints***
> Note that the move of 200 pulses CW will be executed 5 times. The move is executed once, then the program loops four more times to label A0.

MUNits

Description This command configures the indexer to utilize motor units for **MOVE** and **POSITION** commands. The parameter **MR** is the number of motor pulses, steps or microsteps, required for one revolution of the motor shaft. (This value is also used for encoder related commands). The parameter **UA** will be **0** when motor units are active.

Format **MUNits**

Mode Immediate /Stored

Range

Related Functions **MR, UA, MOVE, POSITION**

Example See MOVE command for sample programs.

MOVe [-]*n*

Description Causes the motor to move *n* number of units in the direction set by sign of the *n* value, (the move distance *n* may be in motor units, encoder units or user units as selected by the **MUNITS**, **EUNITS** and **UUNITS** commands). The value of **n** is assumed to be in the "**+**" direction unless it is preceded by an optional **"-"**. The move profile is determined by the parameters: base speed **B**, maximum velocity **M**, acceleration time **A** and deceleration time **D**. The **MOVE** is relative from the current motor position parameter **P**, and the resulting new motor position is stored in parameter **P**. The parameter **P** is continually updated while the motor is running.

Format **MOVe** *[-]n*

Mode Immediate /Stored

Range Maximum n = ±2.1 Billion.

Related Functions **A, B, D, E, H, M, P, UA, UP, CONTINUOUS, EUNITS, MUNITS, PAUSE, UUNITS,**

Example If the motor is currently in position "-100" and a **MOVE** 200 command is executed, the motor will move 200 steps clockwise to position P=100.

```
ICL Command     Units Active      Parameter "UA" Value
MUNITS          Motor units       "UA = 0"
EUNITS          Encoder units     "UA = 1"
UUNITS          User units        "UA = 2"
```

Example Another practical example is the use of user units in a coil winding application.

```
PAUSE  ^ Sets processing mode to single line execution
UUNITS ^ Establishes that motion will be in user units.
MR=5000
^ Sets motor resolution to 5000 steps per shaft
revolution.
UR=1
^ Sets user resolution to 1 shaft rev/user unit
MOVE 200   ^ Moves 200 shaft revolutions
WAIT 40    ^ Wait 1/40th of a second
MOVE 3     ^ Move 3 shaft revolutions.
```

Sample Program The portion of a program that performs motion in terms of "coil Turns" follows. Sample Move utilizing motor units.

```
PAUSE      ^ Sets processing mode to a single line.
MR=36000   ^ motor resolution
MUNITS     ^ Select motor units
P=0        ^ Set current motor position to zero
MOVE 12000 ^ Moves the motor shaft 120 degrees
WAIT 50    ^ Waits one half second
MOVE 12000 ^ Move the motor shaft 120 degrees
WAIT 50    ^ Waits one half second
MOVE 12000 ^ Move the motor shaft 120 degrees
WAIT 100   ^ Waits one second
MOVE 36000 ^ Returns to original position 0
```

Sample Move utilizing encoder units.

```
PAUSE      ^ Set processing mode to single line.
MR=36000   ^ Motor resolution per one shaft rev
ER=4000    ^ Encoder resolution per one shaft rev
EUNITS     ^ Select encoder units
P=0        ^ Set motor position to zero
MOVE 1333  ^ Move the motor shaft 120 degrees
WAIT 50    ^ Wait one-half second
MOVE 1334  ^ Move the motor shaft 120 degrees
WAIT 50    ^ Wait one-half second
MOVE 1333  ^ Move the motor shaft 120 degrees
WAIT 100   ^ Wait one second
MOVE 4000  ^ Return to original position 0
```

(continued) **MOVe**

Sample Program Sample a move utilizing integer variables.

```
PAUSE    ^ Set processing mode to single line.
MR=36000 ^ Motor resolution per one shaft rev
ER=4000  ^ Encoder resolution per one shaft rev
P=0      ^ Set motor position to zero
V1 = 36000
V2 = 500
V3 = 200
V4 = 720000
V6 = 300
MOVE V1
WAIT V2
MOVE V1
WAIT V3
MOVE V4
WAIT V6
MOVE -V2
QUIT
```

Programming Hints

When commanding motion in **UUNITS** or **EUNITS** with the **MOVE** command you may experience a cumulative-positioning-error due to truncation when the indexer is converting to motor units. This truncation will occur whenever **MR/UR** or **MR/ER** is a non-integer value. Use of the **POSITION** command will eliminate positioning errors due to truncation.

NOVice

Description The indexer will respond to the host device with full message text when an error is encountered (see section ICL SYSTEM MESSAGES). The default setting is **NOVICE**. The corresponding command **EXPERT** results in the sending of the error message "number" only. **Note the axis unit designator precedes the error message.**

Format **NOVICE**

Mode Immediate /Stored

Range

Related Functions **EXPERT**

Example In NOVICE mode a message would look like -
X #5 NOT ALLOWED IN PROGRAM

In the EXPERT mode the message would be -
X #5

Sample Program

PASsword

Description This command will prompt the user for an ASCII string, if already assigned the user must re-enter the password to clear this feature. If typed by accident the user must type a <Return> key and no password will be assigned. The password allows the programmer to lock-out the user from the following commands: **BOOT**, **CLEAR**, **DUMP**, **EDIT** and **LIST** but will allow the user to respond to **PROMPT** commands. Thus the user may direct the flow of the program but will not be able to alter it.

Format **PASsword**

Mode Immediate

Range

Related Functions **BOOT, CLEAR, DUMP, EDIT, LIST**

Example

Sample Program

Programming Hints

DO NOT FORGET YOUR PASSWORD!!!!!!! Contact an Applications Engineer at API-Controls Division if you have lost your password.

631-9800

PAUse

Description The **PAUSE** command instructs the ICL System to complete execution of current command prior to "attempting" to execute the next command. In this mode, commands such as setting outputs and checking inputs will not executed while motion is occurring. To continue processing before the current command is complete a **CONTINUOUS** command may be executed.

Format **PAUse**

Mode Immediate/Stored

Range

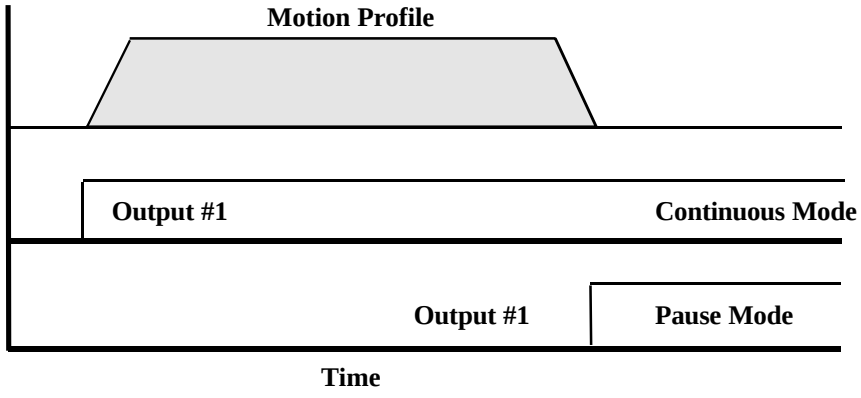
Related Functions **CONTINUOUS, GO, MOVE, OF (Other Functions Bit #8), POSITION**

Example In pause mode: Output #1 is set after motion ends.

```
PAUSE
(AA) MOV 200000          ^ Move 200000 pulses CW
SET 3
```

In continuous mode: Output #1 is set after motion begins.

```
CONTINUE
(AA) MOV 200000          ^ Move 200000 pulses CW
SET 3
```



POSition [-]n

Description Causes the motor to move to absolute position *n*, (the move distance *n* may be in motor units, encoder units or user units as selected by the **MUNITS**, **EUNITS** and **UUNITS** commands). The move is determined by the parameters: base speed **B**, maximum velocity **M**, acceleration time **A** and deceleration time **D**. Since **POSITION** is an absolute move, the position specified by **n** is assumed to be in the "**+**" direction unless preceded by a "**-**". This indicates which side of position 0 (or **HOME**) the motor is to be positioned. The location 0 is where the parameters **P** and or **E** is equal zero (or **HOME**).

The motor position parameter **P** is continuously updated during execution of this command and will equal the commanded position at the end of the move command.

When the **CONTINUOUS** mode is active other commands such as setting outputs and checking inputs can be executed while motion is occurring. To wait for motion to complete before an output is set, a **PAUSE** command may be executed.

Format **POSition** *[-]n*

Mode Immediate/Stored

Range Maximum n = ±2.1 Billion

Related Functions **A, B, D, E, H, M, P, UA, UP, CONTINUOUS, EUNITS, MUNITS, PAUSE, UUNITS,**

Example If the motor is currently in position "-100" and a **POSITION** 200 command is executed, the motor will actually move 300 steps clockwise.

```
ICL Command     Units Active      Parameter "UA" Value
MUNITS          Motor units       "UA = 0"
EUNITS          Encoder units     "UA = 1"
UUNITS          User units        "UA = 2"
```

Example In the following example we will position the motor at 1° increments utilizing a motor microstep resolution of 36000 steps/revolution. The encoder resolution after quadrature is 4000 counts/revolution. The motor is positioned at 120° positions utilizing user units, motor units and encoder units.

```
UR=360   ^ User resolution
ER=4000  ^ Encoder resolution
MR=36000 ^ Motor resolution
PAUSE    ^ Set pause mode active
UUNITS   ^ Select user units
UP=0     ^ Set current user position to zero
POS 120 WAIT 100
POS 240 WAIT 100
POS 360 WAIT 100
MUNITS   ^ Select motor units
P=0      ^ Set current position to zero
POS 12000 WAIT 100
POS 24000 WAIT 100
POS 36000 WAIT 100
EUNITS    ^ Select encoder units
E=0       ^ Set current encoder position to zero
POS 1333 WAIT 100
POS 2667 WAIT 100
POS 4000 WAIT 100
```

PROmpt ccc v

Description This command is used to change the value of an ICL parameter/variable from within the user's program. The ASCII string *ccc* must be enclosed in <"> quotes and may be up to 68 characters. The variable or parameter to be changed is *v*. The ICL prompt command will only accept integer values or string values.

Format **PROmpt** *ccc v*

Mode Stored

Range

Related Functions

Example

```
A) UNTIL 1 A
      ^ Test input 1, if inactive branch to label A
PROMPT "ENTER V1" V1
PROMPT "Enter maximum velocity " M
SEND -1 "TRIGGER INPUT #2 TO CONTINUE" ^ Send to host
B) UNTIL 2 B      ^ Hold here until input 2 is active
$V3 = "ENTER YOUR NAME "
PROMPT $V3 $V3
```

Sample Program

QUIt

Description Stops execution of the current program and returns to the command mode or to the calling program from a subroutine.

Format **QUIt**

Mode Stored

Range

Related Functions

Example

Sample Program

```
A) IF ((F1) AND (F2)) A
IF (F6) ACTIVE6
QUIT              ^ If flag 6 is inactive quit program
(ACTIVE6) HOME    ^ Execute home command
```

> ***Programming Hints***
>
> Must be used as the last command line of a program to prevent corruption of internal stacks and flags when a program error occurs. Used where necessary to exit a subroutine program and return to a calling program.

RENname “old” “new”

Description This command will allow the user to rename an existing program to a new program name without the need for re-keying the entire program. Renames a program named “old” to “new” on the directory. The program labels “old” and “new” are alphanumeric labels from 1 to 8 characters long. The ICL system will provide a warning if the “new” program name already exists so that you will not over-write files.

Format **RENname fff fff**

Mode Immediate

Range

Related Functions **DIRECTORY**

Example
```
RENAME PART1 PART22
```
Renames the program “PART1” to a new program name PART22

```
RENAME TEST MAIN
```
Renames the program “TEST” to new program name MAIN

RESet [F]n (Reset outputs or flags)

Description Changes the status of outputs or ICL Flags to **INACTIVE**, where *n* is the output number and **F***n* is the flag number to be set to **0** (INACTIVE). The **SET** command is used to change outputs or flags to **1** (ACTIVE).

The parameter **OD** is used to set the default states of the outputs. (See your indexer User Guide for the number and location of outputs).

Format **RESet [F]***n*

Mode Immediate/Stored

Range 0 to 13

Related Functions

Example To “turn off” outputs 1, 3 and 5 the following command would be issued.
```
RESET 1 3 5
```

To “turn off” ICL flags 1 and 64 the following command would be issued.
```
RESET F1 F64
```

Sample Program

Hazard Note

The **RESET 0** command allows the user to pulse the reset output to the drive. This command can be utilized when selecting different step resolutions or to clear a drive-fault condition. **THE DRIVE WILL LOOSE POWER TO THE MOTOR DURING A RESET 0 COMMAND, VERTICAL LOADS WILL FALL AND THE MOTOR WILL BE RELOCATED TO A ZERO PHASE LOCATION.**

REPort

Description Issuing a **REPORT** command from the command line mode results in the controller will returning a "#24 Motor Idle" message to the host device upon completion of any move in progress.

Format **REPort**

Mode Immediate and Stored

Range

Related Functions **HOME MOVE, POSITION, RUN, SLEW,**

Example The **REPORT** or **PAUSE** commands will cause the program execution to pause until the last motion is complete before execution of the next command. To wait for motion to complete before an output is set, a **REPORT** command may be executed as in the following example.

```
(AA) POSITION 2000
            ^ Move to position 2000 pulses CW of zero
REPORT
```
Cause the program to pause until last move is complete
```
SET 3       ^ Set output 3 after move is complete
```
Output 3 is set only after motion has stopped and "#24 Motor Idle" message is displayed.

Sample Program

RUN

Description This command will cause the motor to run continuously in the current direction at base speed as defined by base speed parameter **B**, until commanded to stop with the **STOP** or **ABORT** commands. An <ESC> key can also interrupt the move. The **RUN** command can only be issued when the motor is at standstill. The current motor position is maintained by the parameter **P** and all inputs and outputs will be functional. When used in a program, the controller will start moving at the base speed, then proceed to the next sequential program line as in the following example. (Pause mode has no effect when this command is active, the ICL System will assume **CONTINUE** mode.)

Format **RUN**

Mode Immediate/ Stored

Range

Related Functions **A, B, D, H, M, P, UP, ABORT, REPORT, STOP**

Example
```
(AA) POSITION 0 ^ Move to position zero
- RUN           ^ Begin to run in the CCW direction
SET 3           ^ Set output 3
WAIT 100        ^ Cause a 1 second program pause
RESET 3         ^ Reset output 3
(AB) UNTIL 1 AB ^ Poll input 1 until it becomes active
STOP            ^ Stop the motor when input 1 is active
REPORT          ^ Report when motor has stopped
WAIT 500        ^ Cause a 5 second program pause
LOOP 9 AA ^ Cause program to loop 9 times to label AA
```

SENd [-]c ccc [;]

Description This command is used to send an ASCII string to the host device or to other indexers. The ASCII string must be enclosed in <**“**> quotes or <**‘**> and may be up to 68 characters, (<**“**> and <**‘**> are treated the same, i.e. nested quotes are not allowed). The destination is defined as device **c** in the RS-232 daisy chain. An ICL Command string may only be sent down the daisy chain to a device with the unit designator **c**. A string may not be sent up the daisy chain except to the host device, the host device is defined as device **-1**.

The suffix character “;” can be utilized to suppress the <CR><LF>, (Carriage Return, Line Feed) when data is transmitted.

```
V1 = 99
Send -1 "Position Count = ";
Send -1 V1
```
Results in the following:
```
Position Count = 99
```

Format **SENd [-]c** *ccc* **[;]**

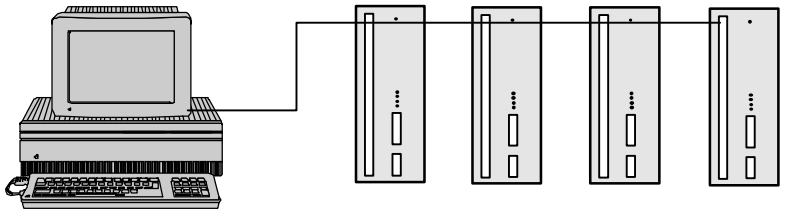
Mode Immediate/ Stored

Range

Related Functions

Example Host device indexer Daisy Chain axis designator

-1 ————> 1 ————> 2 ————> X ————> Y ————> other



Axis 1 can send to 2, X, Y, other.
Axis 2 can send to X, Y, other.
Axis X can send to Y, other.
Axis Y can send to any other devices down the daisy chain.
All axes can send to the host device via the unit designator “-1”.

Sample Program
```
(A) UNTIL 1 A
^ Test input 1, if inactive branch to label A
SEND 3 "MOV 5000" ^ Send to device 3 ICL Command MOV 5000
SEND 4 "CALL R54" ^ Send to device 4 ICL Command CALL R54
SEND -1 "TRIGGER INPUT #2 TO CONTINUE" ^ Send to host
(B) UNTIL 2 B
```

(continued) SENd -1|“1BE” (Send Non-Printable ASCII Strings

Description The “|” key is located on a standard 101 keyboard. This command tells the ICL System that the next two characters following the <”> are to be transmitted as extended ASCII. The balance of the string is transmitted as ASCII characters.

Format **SENd -1| “1BE”**

Mode

Range ASCII 128 to 256

Related Functions

Example

```
SEND -1|"07"
                    ^ Send <CONTROL G> to host
                    ^ Ring bell
SEND -1|"IB"
                    ^ Send <ESC> to host
SEND -1|"IBPA"
                    ^ Turn off LED 1 on T-120
                    ^ Date terminal
SEND -1|"IBPB"
                    ^ Turn off LED 2 on T-120
                    ^ Data terminal
SEND -1|"IBPA"|"IBPB"
                    ^ Turn off LEDS 1 & 2
```

Sample Program

Programming Hints

This command can be used to transmit codes to the host device to control its display, display location, color and many of the other features that may be available on you host device.

SET [F]n

Description Changes the status of outputs or ICL Flags to **ACTIVE**, where *n* is the output number or **F***n* is the flag number to be set to **1** (ACTIVE). The **RESET** command is used to change the output or ICL Flag to **0** (INACTIVE). The parameter **OD** is used to set the default states of the outputs. (See your indexer User Guide for the number and location of outputs).

Format **SET [F]***n*

Mode Immediate/Stored

Range Flag F1 to F64 and output 1 to 13 indexer dependent.

Related Functions **OD, RESET**

Example To "turn on" outputs 1, 3 and 5 the command structure is:
SET 1 3 5

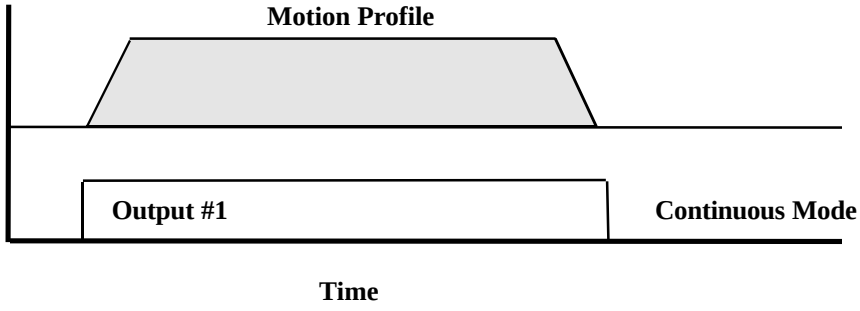
To "turn on" ICL Flags 1, 5 and 8 the command structure is:
SET F1 F5 F8

In pause mode : Output #1 is set after motion ends.

```
PAUSE
(AA) MOV 200000          ^ Move 200000 pulses CW
SET 1
```

Sample Program In continuous mode: Output #1 and flag F3 are set after motion begins and output 1 is reset after the move completes. Note that system-in-motion flag 0 (F0) is used to pause program execution until motion is complete.

```
CONTINUE
(AA) MOV 200000          ^ Move 200000 pulses CW
SET 1 F3
(DONE) IF (F0) DONE      ^ Wait here for motion done
RESET 1                  ^ Reset output 1
```



SHOw

Description This command directs the indexer to transmit a listing of all active ICL variables and their current assigned string or integer values to the host device. The **SHOW** command will list one variable per line, thus the screen characters parameter, **SC** will have no effect on the output. Variables can also be checked on an individual basis by use of the **VERIFY** command.

Format

Mode Immediate/Stored

Range

Related Functions **T, VERIFY**

Example Note: The lines to transmit parameter T, is used to define the number of lines sent to the host device before pausing. Set T=10 command will instruct the indexer to transmit 10 lines and pause until any character is typed. The listing may be terminated during a pause by the <ESC> command.

The **SHOW** command is used to display variables that are not cleared as in the example below.

```
V1=300 V55=77 V22=34000 $V3="ENTER MOVE DISTANCE"
SHOW
"V1 = 300"
"V3 = ENTER MOVE DISTANCE"
"V22 = 34000"
"V55 = 77"
```

Sample Program

SLEw

Description This command causes the motor to move utilizing the parameters; base speed **B**, acceleration time **A** to the maximum velocity **M**, then proceed to the next sequential program line and slew until the **STOP** or **ABORT** commands are issued. The motor direction is set prior to issuing the slew command. An <ESC> key can interrupt the move but the parameter **P** may loose sync due to the resulting abrupt stop. This command can only be executed when the motor is at standstill. The current motor position is maintained by the parameter **P**. Other commands such as setting outputs and checking inputs can still be executed while motion is occurring. (Pause mode has no effect when this command is active, the ICL System will assume **CONTINUE** mode.)

Format

Mode Immediate/Stored

Range

Related Functions **A, B, D, H, M, P, UP, +, -, ABORT, CONTINUE, STOP**

Example

```
(AA) POSITION 0    ^ Move to position zero
+ SLEW             ^ Begin to slew in the CCW direction
SET 3              ^ Set output 3
WAIT 100           ^ Cause a 1 second program pause
RESET 3            ^ Reset output 3
(AB) UNTIL 1 2 AB  ^ Poll inputs 1 and 2 until they
                   ^ become active
STOP               ^ Stop the motor when inputs 1 and 2
                   ^ are active
REPORT             ^ Report when motor has stopped
WAIT 500           ^ Cause a 5 second program pause
LOOP 9 AA          ^ Cause program to loop 9 times to
                   ^ label AA
```

Sample Program

> ***Programming Hints***
>
> The direction is set by a preceding motion and may result in improper motion. You should precede the slew command with a + or - direction command. This will make your programming easier and provide predictable motion.

STAtus

Description This command directs the indexer to transmit a listing of all ICL parameters and their current states or values to the host device. Please note that within this system, parameters provide information on "HOW TO DO IT", while commands tell the indexer "WHAT TO DO". The parameters can also be checked on an individual basis by use of the **VERIFY** command. A sample of the transmitted **STATUS** display is shown below:

Note: The screen characters parameter **SC** is used to define the number of characters available per line on the host's display. To configure the output screen width to 40 characters, set **SC=40**. The parameter **SC** can be a value from 20 to 80 characters depending on the host's display width.

Format

Mode Immediate/Stored

Range

Related Functions All ICL Parameters

Example Note: The lines to transmit parameter **T**, is used to define the number of lines sent to the host device before pausing. Set **T=10** command will instruct the indexer to transmit 10 lines and pause until any character is typed. The listing may be terminated during a pause by the **<ESC>** command.

```
0> STAT
   E   -ENCODER POSITION       =  0
   ER  -ENCODER RESOLUTION     =  4000
   P   -POSITION               =  0
   MR  -MOTOR RESOLUTION       =  400
   UP  -USER POSITION          =  0
   UR  -USER RESOLUTION        =  400
   A   -ACCELERATION TIME(ms)  =  10000
   D   -DECELERATION TIME(ms)  =  10000
   B   -BASE SPEED             =  200
   H   -MAXIMUM SPEED LIMIT    =  50000
   M   -MAXIMUM SLEW SPEED     =  6000
   J   -JOG SPEED              =  50
   N   -LAST INDEX DISTANCE    =  0
   RD  -USER RAMP DATA FILE #  =  -1
   PE  -MAX. POSITION ERR.(E)  =  400
   ME  -CURRENT MOTOR ERR.(M)  =  0
   DW  -DEADBAND WINDOW (E)    =  5
   CG  -POSITION CORRECT GAIN  =  100
   CV  -MAX CORRECT VELOCITY   =  1000
   EF  -ENCODER FUNCTIONS      =  00000000
   SP  -STALL DET. PROGRAM#    =  -1
   UA  -UNITS ACT(0=M,1=E,2=U) =  0
   I1  -STATES, INPUT SET 1    =  00000000
   ID  -INPUT DEFINITION 1     =  00000000
   I2  -STATES, INPUT SET 2    =  00000000
   ID2 -INPUT DEFINITION 2     =  00000000
   O1  -STATES, OUTPUT SET 1   =  00000000
   OD1 -OUTPUT DEFINITION 1    =  00000000
   JP  -JOG PLUS INPUT #       =  0
   JM  -JOG MINUS INPUT #      =  0
   SI  -STOP INPUT #           =  0
   LA  -LIMIT ACTION SETTINGS  =  10000000
```

```
OF -OTHER FUNCT. ENABLES   =  00000000
F  -FREE SPACE             =  20848
G  -AUTOSTART PROGRAM#     =  -1
Q  -POWER UP TIME (10ms)   =  0
R  -LOW POWER TIME (10ms)  =  0
S  -NO POWER TIME (10ms)   =  0
U  -UNIT DESIGNATION       =  0
W  -CURRENT PROGRAM #      =  0
X  -INSTRUCTION ADDR       =  0
Y  -LOOP COUNTER           =  0
Z  -CURRENT LINE NUMBER    =  0
T  -LINES TO DISPLAY       =  25
SC -SCREEN WIDTH           =  80
```

STOp

Description When executed this command will cause the motor to stop. This **STOP** command may be used to stop motion after a **MOVE**, **POSITION**, **RUN** or **SLEW** command has been issued. When used with a **MOVE**, **POSITION** or **SLEW**, the parameter **D** deceleration time, will be used to ramp the motor down to base speed as defined by parameter **B** before stopping. When used with **RUN** the indexer will immediately stop since it is already at the base speed **B**.

Format

Mode Immediate/Stored

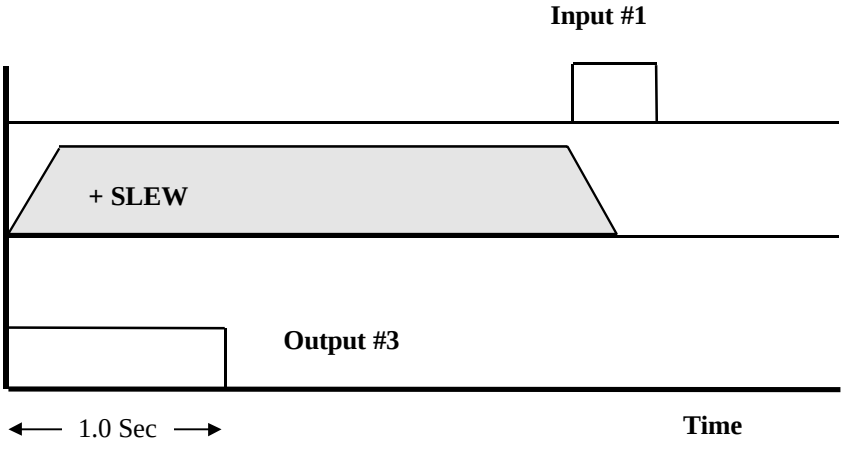
Range

Related Functions **MOVE, POSITION, RUN, SLEW**

Example

```
(AA) POSITION 0 ^ Move to position zero
+ SLEW           ^ Begin to slew in the CCW direction
SET 3           ^ Set output 3
WAIT 100        ^ Cause a 1 second program pause
RESET 3         ^ Reset output 3
(AB) UNTIL 1 AB ^ Poll input 1 until it becomes active
STOP            ^ Stop the motor when input 1 is active
REPORT          ^ Report when motor has stopped
WAIT 500        ^ Cause a 5 second program pause
LOOP 9 AA  ^ Cause program to loop 9 times to label AA
```

Sample Program



SYNc n

Description This command is utilized to poll input *n* at high speed, (300 - 500 micro-seconds). Program execution will be temporarily suspended until the input changes state, at which time the program execution will continue with the next command line.

This command could replace a command line **(START) UNTIL 6 START.**

Format Stored

Mode

Range Indexer dependent (Input #1 - 16)

Related Functions **ID1, ID2**

Example

```
(AA) SYNC 6\ ^ Trailing edge detect
SYNC 6       ^ Poll input 6 until it becomes active
- SLEW       ^ Begin to slew in the CCW direction
(AB) SYNC 6\ ^ Poll input 6 until it becomes inactive
STOP ^ Stop the motor when input 1 is active JUMP AA
POS V12      ^ Make a long move
SYNC 4       ^ Poll input 4 until it becomes active
V1=P         ^ Capture current position
STOP         ^ Stop current motion
POS V1+V6 ^ Move to offset distance V6 from input #4
```

Sample Program

Programming Hints

The SYNC command is usually used in pairs for the purpose of a "trailing edge then leading edge" program flow. This will make your programming easier and provide predictable motion.

Hazard Note

The **SYNC** command should only be utilized with debounced sensors such as solid state relays and photoeyes. If your program loop is short, the SYNC command may read contact bounce in mechanically switched inputs.

The SYNC command will cause the program execution to stop until the input is evaluated as TRUE. The stop input, SI should be utilized to terminate motion and prevent potentially unsafe situations.

TIMe

Description When executed this command will cause the indexer to report the time for **ACCEL**, **SLEW**, **DECEL** and **TOTAL** time of the last indexed distance, parameter **N**. The last indexed distance may be a **MOVE**, **POSITION**, **RUN** or **SLEW** command. The values reported for **ACCEL**, **SLEW**, **DECEL** and **TOTAL** times are in milliseconds (ms). This command is required as the indexer utilizes optimal non-linear mathematical functions for ramping. The mathematical function utilizes ICL parameters for acceleration time **A**, base speed **B**, deceleration time **D**, highest speed limit **H** and maximum velocity **M**.

Format

Mode Immediate/Stored

Range

Related Functions **A, B, D, H, M, MOVE, POSITION, RUN, SLEW**

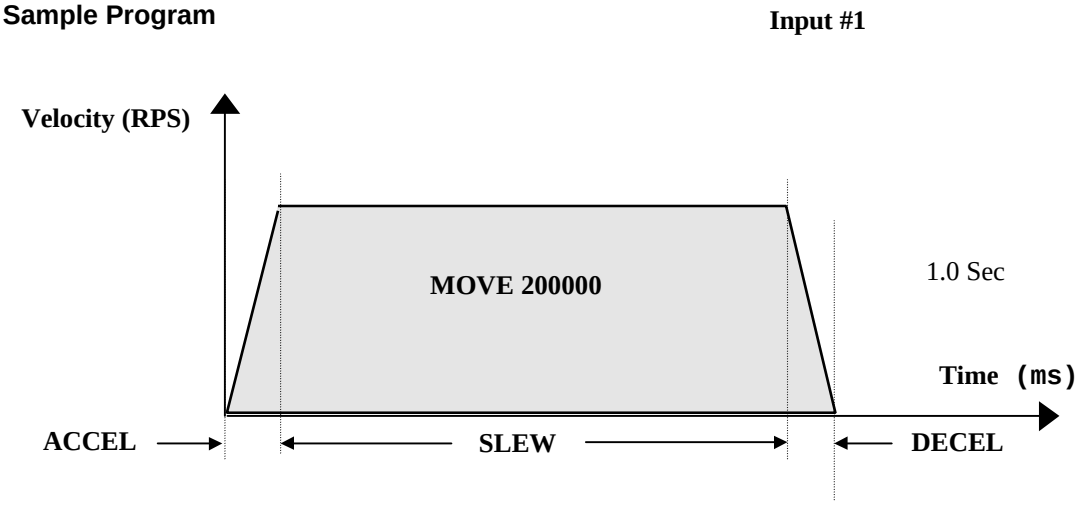
Example The **TIME** command and the results are shown in the following example:

```
A=400 B=50000 D=400
M=225000 H=225000
MOVE 200000
TIME
```

When the move is completed the ICL System will report:

```
"ACCEL = 348 SLEW = 327 DECEL = 344 TOTAL = 1019 "
```

Sample Program



TRAce n

Description This command is used as a debugging tool. **TRACE** will execute a number of program lines starting in the program defined by current program number parameter **W**, on instruction address defined by parameter **X** and current line number defined by parameter **Z**. The number of lines executed is specified by *n* before stopping in command mode. When used with a program containing **IF** and **UNTIL** commands the resulting position of trace may vary.

Format **TRAce** *n*

Mode Immediate

Range 1-127

Related Functions

Example

Sample Program

Debugging Tool

Edit your program and press <CR> until you move to the line of code you wish to begin execution on, and press<ESC>. You have now positioned an internal pointer to a location in memory within your program. You will now be in the immediate mode. Type: **Trace 10 <CR>.**

The ICL System will display a line of code and then execute that line of code. We suggest a value of 10 so that you can fit all the data on a standard terminal display. If an error occurs, you will see the line of code prior to your error. You will also be able to watch for loops, branches, etc.

Hazard Note

Be careful when executing the TRACE command on programs containing the RUN or SLEW commands. Remember the <ESC> key will stop motion.

UNTil [-]n[\] ccc

Description This command tests the status of an input(s) or output(s) *n* and if any are **INACTIVE** directs the program to branch to label *ccc*, where *ccc* is a location label specified in the motion program. If *n* is **ACTIVE**, the program execution will continue at the next sequential program line. If *n* is a positive value the corresponding input is tested. If negative the corresponding output is tested. **Restriction: Label *ccc* must start with an ALPHA character.** (See your indexer User Guide for the number of inputs and outputs).

Format **UNTIL [-]*n*[\] *ccc***

Mode Stored

Range

Related Functions **IF, ID1, ID2, OD1, **

Example This command may be used to cause a wait condition until an input condition is satisfied as shown in the example below:

```
(START) UNTIL 1 START
            ^ Loop to label start until input 1 is Active


(START) UNTIL 1 2 3 START
            ^ Loop to label start until inputs 1,
            ^ 2 and 3 are active.


(START) IF 1\ START


(START) IF 1\ 2\ 3\ START~
```

Sample Program

Programming Hints

NOTE: The parameters **ID** and **OD** are used to define the default states of the inputs and outputs respectively, (see your indexer User Guide for the number of inputs and outputs). **See IF command for additional examples.**

UUNits

Description This command configures the indexer to utilize user units during **MOVE** and **POSITION** commands. See **MOVE** and **POSITION** commands for additional examples.

Format **UUNits**

Mode Immediate/Stored

Range 1 to MR

Related Functions **ER, MR, UR, EUNITS, UUNITS,**

Example The parameter **UR** is an integer value that defines the number of user units desired to equal one revolution of the motor shaft. If the value of **UR** is a real number the user has a choice of selecting a smaller step resolution such as microstepping or by scaling (i.e. 2.54 may be expressed as 254 user units per revolution).

Sample Program

```
PAUSE    ^ Enable pause mode
UUNITS   ^ Configure ICL system for user units
MR=36000 ^ Motor resolution is 36,000 steps per shaft rev
UR=360   ^ User resolution is 360 positions per rev.
         ^ (1 user unit equals 100 motor steps)
P= 0     ^ Set motor position to zero
MOVE 120 ^ Move shaft 120 degrees (12,000 steps)
WAIT 50  ^ Wait one-half second
POS 240  ^ Move shaft to 240 degrees
Wait 50  ^ Wait one-half second
MOVE 120 ^ Move shaft to 360 degrees
WAIT 100 ^ Wait one second
POS 0    ^ Return to original position
```

Programming Hints

When commanding motion in **UUNITS** or **EUNITS** with the **MOVE** command you may experience a cumulative-positioning-error due to truncation when the indexer is converting to motor units. This truncation will occur whenever **MR/UR** or **MR/ER** is a non-integer value. Use of the **POSITION** command will eliminate positioning errors due to truncation.

VERify p

Description This command is used to review the current status of an individual ICL parameter, variable or flag, where **p** is the letter or name of the parameter, variable or flag to be reviewed. When executed, the current value of the selected parameter, variable or flag will be returned to the host device.

Format **VER P**

Mode Immediate/Stored

Range

Related Functions All parameters, variables and flags

Example

```
J=1500      ^ Set jog rate to 1500 pulses per second
VERIFY J    ^ Verify the jog rate
VERIFY F3 VERIFY V1
0 "J = 1500 "
0 "F3 = 1"
0 "V1 = 3652"
0>
```

Sample Program

WAIt n

Description Delays program execution for the specified period *n*, then proceed to the next sequential program line. The delay period *n* is specified in hundredths of a second. Thus, an *n* of 200 would result in a program delay of 2 seconds.

Format **WAIT n**

Mode Immediate/Stored

Range 1-32767

Related Functions **CONTINUOUS, OF (other functions Bit #8), PAUSE**

Example The **WAIT** command works in conjunction with the **PAUSE** and **CONTINUOUS** commands. **PAUSE** instructs the ICL System to complete execution of current command prior to "attempting" to execute the next command. In this mode Commands such as setting outputs and checking inputs will not execute while motion is occurring. To continue processing before the current command is complete a **CONTINUOUS** command may be executed.

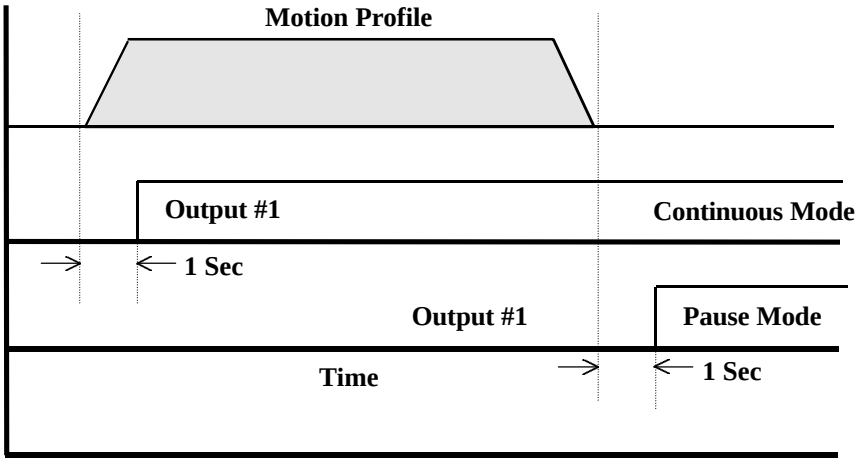
In pause mode: Output #1 is set one second after motion ends.

```
PAUSE
(AA) MOV 200000          ^ Move 200000 pulses CW
WAIT 100
SET 1
```

In continuous mode: Output #1 is set one second after motion begins.

```
CONTINUE
(AA) MOV 200000          ^ Move 200000 pulses CW
WAIT 100
SET 1
```

Sample Program



<Backspace> or <CTRL-H>

Description Used in either immediate command mode or while in edit mode to remove the preceding character from a command prior to the <**Return**>.

Format

Mode Immediate

Range

Related Functions

Example

Sample Program

<^> Caret

Description A **Caret** is used in motion programs to enter comments for clarification. The <^> command is followed by a delimiter character of <**,**> or <**Space**> and then text may be inserted, a <**Return**> is used to end the text.

Format ^ <SPACE> TEXT STRING, COMMENTS

Mode Stored

Range

Related Functions

Example Loop to START until input 1 is active

```
(START) UNTIL 1 START
```

Sample Program

```
(START) SYNC 1
+ SLEW
SYNC 1\
V0 = P
STOP
POS V0 + V1
WAIT 100
JUMP START
^ Input 1 starts process no switch
^ Position is captured as variable v0 when
^ Input 1 becomes inactive
^ Motor moves to offset position
^ Offset is variable v1
^ Wait for 1.00 seconds
^ Repeat cycle
QUIT
```

Programming Hints

A good programming practice is to comment your programs for yourself and others to understand these programs and your thought process.

Use of too many comments will reduce the speed of execution. If you require the fastest possible execution, we suggest that you place your comments at the bottom of the ICL program. Do not use () in your comment statements as the ICL command language may accidentally branch to these labels.

<:>c Colon Axis Designator

Description A **Colon** establishes that the following character is the **Axis Designator** of the indexer that you wish to communicate with through the RS-232 serial communication line. **c** is an alpha numeric character (0-9 or A-Z) that is assigned to the axis designator parameter **U** of the requested indexer, (see "ICL PARAMETER DESCRIPTION"). Only one indexer can be communicated with at a time by using it's unique axis designator. Each indexer will use the axis designator as part of the prompt in command mode.

Format : c

Mode Immediate

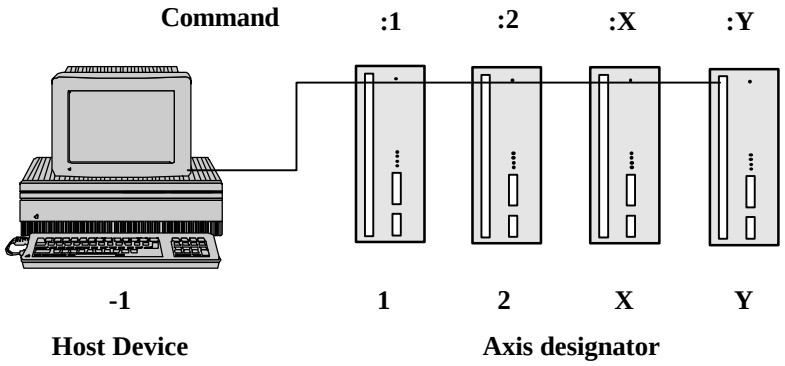
Range 0-9, A-Z

Related Functions **U, SEND**

Example For a controller with axis designator parameter **U** assigned the character **Z**, the prompt will be **Z>**.

When a number of indexers are first daisy chained in series they will all be designated by axis designator parameter **U** as axis **0**. The first indexer in the chain will be the first to sign on. The first axis can be renamed to axis **X** at this time by typing **U=X**, and the prompt will change to **X>**. To log to the next axis in the chain type **<:>0** and the prompt will change to **0>**, this axis can now be renamed as desired. The above process is repeated until all axes have been designated.

Example Host device indexer Daisy Chain axis designator



Sample Program

Programming Hints

NOTE: The Colon character is a reserved character and must not be typed as part of any name, label or comment. The indexer will interpret the Colon as a request to communicate with another axis. If you accidentally type a Colon and Axis Designator of an axis that does not exist and wish to return to the previous axis, type Colon and the Axis Designator of the previous axis. You will be returned to the location in the previous axis where the Colon was accidentally typed.

<,> Comma

Description A **Comma** can be used as a command delimiter and has the same effect as a blank space <**Space**>.

Format

Mode Immediate/Stored

Range

Related Functions

Example

```
A=680,B=900,MR=3600,UR=360,ER=4000,UUNITS
```

Sample Program

> ***Programming Hints***
> NOTE: A <,> or <Space> may not be used as part of any program name or label. The indexer will truncate the character string at the first delimiter and result in an error.

<;> Semicolon

Description A **<;>** is used in conjunction with the **SEND** command to suppress the <CR><LF> when transmitting to the host device. See the SEND command for more detail.

Format

Mode Stored

Range

Related Functions **SEND**

Example

```
V99=876
SEND -1 "COUNT IS ";
SEND -1 V99

COUNT IS 876
```

Sample Program

<CTRL-X>

Description Used to erase the entire current command line. This command can be used in the immediate command mode or edit mode.

Format

Mode Immediate

Range

Related Functions Current data line or program line when in EDIT mode.

Example

Sample Program

</> Divide sign

Description A **Divide sign** is used as an operator for mathematical expressions with the result being a truncated integer value. See the modulo command **%** for working with the truncated remainder of numbers that have been divided.

Format

Mode Immediate/Stored

Range

Related Functions ***, %, ( )**

Example

```
V1=300, V2=200, V0=V1/V2, VERIFY V0
"V0 = 1"
```

Sample Program

Programming Hints

The ICL System utilizes integer mathematics only. To prevent propagation of math errors you should execute multiplication before division. You may use ( ) to control the priority of calculation.

$$V0 = 5* 1000/3 = \frac{5000}{3} = 1666$$

$$V0 = 5/3 \times 1000 = 1 \times 1000 = 1000$$

<\> Backslash sign (NOT Operator)

Description A **Backslash sign** is used as a "not" operator for testing inputs, outputs, flags and comparison expressions.

Format

Mode Stored

Range

Related Functions **IF, UNTIL, SYNC**

Example Branch to label 1 if input 1 active and input 2 inactive
```
IF 1 2\ LABEL1
```

Branch to label 1 if output 1 active and output 2 inactive.
```
IF -1 -2\ LABEL1
```

Branch to label 1 if flag 1 is TRUE and flag 2 is FALSE.
```
IF ((F1) AND (F2\)) LABEL1
```

Sample Program
```
IF 1\ TOP
UNTIL 1 TOP
```
The above two statements are identical in function, they branch to label top when input 1 is inactive.

<%> Modulo Operator

Description A **Modulo Operator** is used in integer arithmetic. It gives the remainder that results when the integer to its left is divided by the integer to its right. For example, 13 % 5 (read as **13 modulo 5** has a value 3, since 5 goes into 13 twice with a remainder of 3.

Format V0 = V1 % V2

Mode Immediate/Stored

Range

Related Functions *, /

Example

```
V1=300, V2=155 V0=V1/V2 V3=V1%V2 VERIFY V0 VERIFY V3

                          (300/155) = 1.9354838
"V0 = 1"                            1
"V3 = 145" (145/155) = 0.9354838
```

(The truncated remainder)

For the case of **V1/V2*V3**

```
V1=500 V2=155 V3=10
V5=V1/V2*V3
VERIFY V5
V0=V1/V2*V3+(V1%V2)*V3/V2
VERIFY V0

"V5 = 30"
"V0 = 32"•••500/155*10 = 32.258065
```

Sample Program The example below uses the modulo function to determine the number of motor steps necessary to return the shaft to a zero degree location.

```
MUNITS          ^ Select motor units
MR=36000        ^ Motor resolution
(AO) P=O        ^ Set position at 0°
SYNC 2          ^ Poll input 2 until it becomes active
MOVE 225000     ^ Rotate 6.25 revolutions
V1 = P % MR     ^ Calculate modulo remainder
MOVE V1         ^ Rotate 0.75 rev to 0° position
JUMP AO         ^ Jump to label AO
QUIT
```

<*> Multiply sign

Description A **Multiply sign** is used as an operator for mathematical expressions with the result expressed as an integer value.

Format

Mode Immediate/Stored

Range

Related Functions

Example

```
V1=300, V2=200 V0=V1*V2 VERIFY V0
"V0 = 60000"
```

Sample Program

Programming Hints

The ICL System utilizes integer mathematics only. To prevent propagation of math errors you should execute multiplication before division. You may use ( ) to control the priority of calculation.

$$V0 = 5* 1000/3 = \frac{5000}{3} = 1666$$

$$V0 = 5/3 \times 1000 = 1 \times 1000 = 1000$$

$Vn Dollar sign

Description String variables must be preceded by a **$** command to be distinguished from a label, flag or program name. This command will allow string variables to be used in conjunction with the **CALL**, **PROMPT** and **SEND** ICL-Commands to direct the flow of your program.

Format $Vn = "string"

Mode Immediate/Stored

Range

Related Functions **CALL, PROMPT, SEND**

Example An example of a program using variables:

The user requires the ability to select a subroutine program. The name of the subroutine is the part number to be manufactured. A dynamic program shown below will prompt the user for the subroutine name (part number). Input #1 begins execution of the sequence.

```
(A0) PROMPT "ENTER PART NUMBER " $V1
$V99="ENTER NUMBER OF PARTS "
PROMPT $V99 V2
SEND -1 "TRIGGER INPUT #1 WHEN READY TO BEGIN PROCESS"
PAUSE V2=V2-1
(START) UNTIL 1 START
(A) CALL $V1
IF (V2=0) END
LOOP V2 A
(END) SEND -1 "PROCESS IS COMPLETE"
JUMP A0
```

Programming Hints

The ability to utilize string variables in conjunction with CALL command is a powerful tool that allows for dynamic programming techniques. We encourage you to take advantage of this capability where required.

NOTE: String compares are not allowed within the ICL System.

<ESC> or <CTRL-[>

Description **Escape Key** on the host key board**. The <ESC> key is used to terminate motion in the immediate mode, terminate execution of a current program or command and to exit the ICL editor.** In the ICL Editor the last edit line must be entered into a program by pressing the **<Return>** key, before the **<ESC>** key will be recognized.

Format

Mode Immediate

Range

Related Functions <~> Global Escape command.

Example

Sample Program

> ***Programming Hints***
> NOTE: If the users host device does not have an <ESC> key, the <CTRL-[>, (control and left bracket sign) may be substituted. The <~> key is used as a global <ESC> key for all axis on the RS-232 daisy chain.

<-> Minus sign

Description A **Minus sign** is used to set "counter clockwise" as the current direction of travel by setting the direction output to the step motor driver logically low. The **<->** is used in **MOVE** and **POSITION** commands to define direction or position. This command is also used as a unary-minus and an operator for mathematical expressions.

Format

Mode Immediate/Stored

Range

Related Functions **+, MOVE, POSITION, RUN, SLEW**

Example

```
V0=V1-V2-V3 POS V0 POS -5000 MOVE -2000 POS -V0
```

Sample Program

> ***Hardware Configuration***
> To reverse the default direction of the motor, you must switch the motor "A+" and "A-" motor leads. This may be helpful to have "positive" motion "up" and "negative" motion "down".

<+> Plus sign

Description A **Plus sign** is used to set "clockwise" as the current direction of travel by setting the direction output to the step motor driver logically high. The <+> is not used in **MOVE** or **POSITION** commands as the default number is assumed to be positive. This command is also used as an operator for mathematical expressions.

Format

Mode Immediate/Stored

Range

Related Functions **-, MOVE, POSITION, RUN, SLEW**

Example
```
V0=V1+V2+V3 POS V0 POS 5000 MOVE 2000 POS -V0
```

Sample Program

> ***Hardware Configuration***
> To reverse the default direction of the motor, you must switch the motor "A+" and "A-" motor leads. This may be helpful to have "positive" motion "up" and "negative" motion "down". You must switch the wiring of you limit inputs whenever you reverse the default direction of the motor.

<=> Equal sign

Description This command allows the user to change an ICL parameter or variable value. The format is **p = n**, where **p** is the name of the parameter or variable you wish to change and **n** is the new value to be assigned. This command may be used in immediate mode or within a program. The value of a parameter or variable can be checked through the use of the **VERIFY p** command. (An alternate method is to use the **ENTER** command to set values).

Format p = n, p = bbbbbbbb

Mode Immediate/Stored

Range

Related Functions All parameters and variables

Example
```
B=1000 ^ Set the base speed to 1000 PPS
J=500  ^ Set the jog speed to 500 PPS
$V10="THIS IS A STRING"
V0=V1*(V2+V3-V4)
```

The equal command is also used as a comparison operator to test variables. (See **IF** command.)

```
IF ((V1 = V2) AND (V4 > V5)) LABEL3
```

Sample Program

<, <=, =, <>, => and > Comparison Operators

Description Comparison operator commands **(<, <=, =, <>, => and >)** are used to compare variables and/or parameters to control branching within the ICL System.

Format

Mode Immediate/Stored

Range

Related Functions **IF, UNTIL**

Example

```
IF (V1 = V2) EQ2
            ^ Branch to EQ2 if V1 equals V2
IF (V1 < V2) LT2
            ^ Branch to LT2 if V1 is less than V2
IF (V1 <= V2) LTEQ2
      ^ Branch to LTEQ2 if V1 less than or equals V2
IF (V1 > V2) GT2
      ^ Branch to GT2 if V1 is greater than V2
IF (V1 <> V2) NE2
      ^ Branch to NE2 if V1 is not equal to V2
```

Sample Program

AND and OR Boolean Operators

Description Boolean operator commands such as **AND** and **OR** are used in conjunction with flags and compares to control branching within the ICL System. (See **IF** command.)

Format **((condition)) AND (condition))**

Mode Immediate/Stored

Range

Related Functions **IF, UNTIL**

Example

```
IF ((V1=V2) OR (F1)) EQ2
^ Branch to EQ2 if V1 equals V2 or flag 1 is active
IF ((V1<V2) AND (F1\)) LT2
^ Branch to LT2 if V1 is less than V2 and flag 1 is
^ Inactive
```

Sample Program

<(> Parentheses

Description Parentheses are used in the ICL System to define labels in programs, to direct the order in which mathematical expressions are evaluated and to delimit comparison operator expressions.

The **(** declares that the following characters are a label until **)** is found. The maximum label length is eight (8) alpha numeric characters. **A label must start with an alpha character.** The label must not contain any of the following characters **<,> Comma, <Space>, <~> Tilde** or the **<:> Colon.** ***Note that the*) *is followed by a delimiter* <Space> *or* <,> *to separate it from the next command.***

Format (LABEL)

Mode Immediate/Stored

Range

Related Functions Program names and labels, **IF, UNTIL**

Example
```
(IN1) IF 2 IN2
(IN2),SEND -1 "INPUT TWO IS ACTIVE"
```

Description **The user can elect to utilize parentheses to direct the order in which a mathematical expressions are evaluated.** Mathematical functions such as addition, subtraction, modulo, division and multiplication (+, -, %, / and *) may be performed on numeric variables. Note that when mathematical functions are performed on variables the result is an integer value where the remainder portion of the number is truncated. The order of precedence is: (unary -) then (%, * or /) then (+ or -) otherwise the expression is evaluated left to right.

Example
```
V0=360 V1=500 V2=200
V3=V1 / V2 VERIFY V3
V4 =V1-V2 VERIFY V4
V5=V0*V2/V3 VERIFY V5
V6=V0+V2*V1 VERIFY V6
V7=(V0+V1)*V2 VERIFY V7
```

The indexer will report:

```
"V3 = 2"
"V4 = 300"
"V5 = 900"
"V6 = 100360"
"V7 = 172000"
```

Description **The user must utilize parentheses to direct the order in which a comparison expressions are evaluated.** Comparison expressions such as **(>, <, <>, >=, <=, =)** and Boolean operations such as **(OR and AND)** may be performed on numeric variables and flags.

Example
```
(A) IF ((F1) AND (F2)) A
      ^ If ICL flags 1 & 2 are TRUE branch to label A
$V1="ACTIVE6"  ^ Assign V1 the label name
IF (F6) $V1    ^ Test flag 6, if active branch to label
ACTIVE6
QUIT           ^ If flag 6 is inactive quit program
(ACTIVE6) HOME ^ Execute home command
IF (F1) LABEL2
IF ((V1 > V2) AND (V4 > V5) AND F0\)) LABEL3
```

<~> Tilde

Description **Tilde Key** on the host key board. **The <~> key is used to terminate motion in the immediate mode, terminate execution of a current program or command and to exit the ICL editor. The <~> key is used as a global <ESC> key for all axis on the RS-232 daisy chain.** In the ICL Editor the last edit line must be entered into a program by pressing the **<Return>** key, before the <~> key will be recognized. This command may be inserted into a batch file on disk through use of an editing program, thus allowing the user to create a number of programs from a single file transmitted by the host.

Format

Mode Immediate

Range

Related Functions **EDIT, GO, HOME, MOVE, POSITION, RUN, SLEW**

Example

```
EDIT MAIN
HOME
SEND -1 "HOME FOUND"
.
~
EDIT 2NDPGM
.
.
.
.
QUIT
~
EDIT 3RDPGM
.
.
.
.
QUIT
~
```

Sample Program

> ***Hazard Note***
> The <~> key is used as a global <ESC> key to stop motion for all axes on the RS-232 daisy chain.

<Return>

Description **Return** key on host key board. Execute the current command line immediately when in the command mode, or enter the edit line to the program when in the edit mode.

Format

Mode Immediate

Range

Related Functions All ICL commands.

Example

Sample Program

<Space>

Description **Space Bar** on the host key board. Used as a delimiter character and has the same effect as a **<,>, Comma**.

Note that a **<,>** or **<Space>** may not be used as part of any program name or label. The indexer will truncate the character string at the first delimiter and result in an error.

Format

Mode Immediate/Stored

Range

Related Functions

Example

```
MOVE    10000
    SET 1
MOVE   -10000
QUIT
```

Sample Program

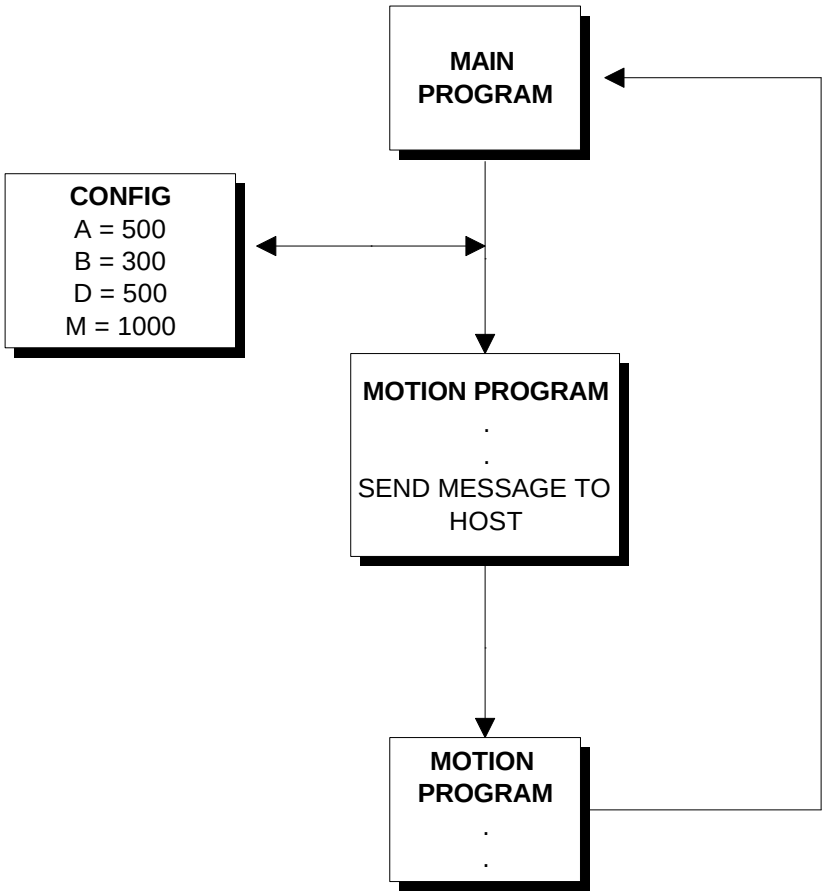
> ***Programming Hint***
> All ICL commands allow extra spaces to be entered with the exception of **(LABELS).** Extra spaces may be utilized to make your program more readable.

PERFORMING MOTION

Motion is performed on an ICL System by using English like ICL commands and parameters. Commands tell the indexer “WHAT TO DO” while parameters tell the system “HOW TO DO IT”. In this section the “WHAT TO DO” we will be discussing is performing motion. The parameters that are used to tell the system “HOW TO DO IT” are as follows:

A - Acceleration time in milliseconds.
B - Base speed, or starting speed, in pulses per second.
D - Deceleration time in milliseconds.
H - Highest velocity used in Optimal Non-Linear Ramping.
M - Maximum velocity in motor pulses per second.

By changing the value of these five parameters you will control how fast/slow you will accelerate, decelerate, and slew the motor. The methods and commands used to perform motion will be described in this section.



Move Types

The six basic motion commands on an ICL System include **MOVE**, **POSITION**, **GO**, **RUN**, **SLEW** and **HOME**. Controlled position type moves are performed by using the **MOVE**, **POSITION** and **GO** commands. Continuous type moves; **RUN** allows the user to run at base speed while **SLEW** and **HOME** commands motion at the maximum velocity until commanded to stop. Descriptions for each move command is listed below. ***NOTE: This user guide uses brackets [ ] to enclose optional characters in a command.***

Controlled Position Moves

Description When **MOVE [-]n** is executed it causes the motor to move **n** number of units, (the move distance **n** may be in motor units, encoder units or user units as selected by the **MUNITS**, **EUNITS** and **UUNITS** commands). The value of **n** is assumed to be in the "**+**" direction unless it is preceded by an optional **"-"**. The move will utilize the current parameters for base speed **B**, maximum velocity **M**, acceleration time **A** and deceleration time **D** The **MOV˜** is relative from the present position and the resulting position is stored in parameter **P**. The maximum move number **n** is [-]2.1 billion steps.

When **POSITION [-]n** is executed it causes the motor to move to absolute position **n**, (the move distance **n** may be in motor units, encoder units or user units as selected by the **MUNITS**, **EUNITS** and **UUNITS** commands). The indexer will utilize the current parameters for base speed **B**, maximum velocity **M**, acceleration time **A** and deceleration time **D**. Since **POSITION** is an absolute move, the position specified by **n** is assumed to be in the "**+**" direction unless preceded by a "**-**". This indicates which side of position **0** (or **HOME**) the motor is to be positioned to. The motor position parameter **P** is continuously updated during execution of this command and will equal the commanded position at the end of the move command.

Example If the motor is currently in position **P= -100** and a **POSITION 200** command is executed, the motor will actually move 300 steps clockwise.

ICL Command	Units Active	Parameter **UA** Value
MUNITS	Motor units	"UA = 0"
EUNITS	Encoder units	"UA = 1"
UUNITS	User units	"UA = 2"

In the following example we will position the motor at 1° increments utilizing a motor microstep resolution of 36000 microsteps/revolution. The encoder resolution after quadrature is 4000 counts/revolution. The motor is positioned at 120° positions utilizing user units, motor units and encoder units.

Example

```
PAUSE
UR=360   ^ User resolution
ER=4000  ^ Encoder resolution
MR=36000 ^ Motor resolution
UUNITS   ^ Select user units

UP=0     ^ Set user position to zero
POS 120 WAIT 100
POS 240 WAIT 100
POS 360 WAIT 100

MUNITS   ^ Select motor units
P=0      ^ Set motor position to zero
POS 12000 WAIT 100
POS 24000 WAIT 100
POS 36000 WAIT 100

EUNITS   ^ Select encoder units
E=0      ^ Set encoder position to zero
POS 1333 WAIT 100
POS 2667 WAIT 100
POS 4000 WAIT 100
```

Continuous Moves

Description The continuous type moves, **RUN** and **SLEW**, cause the motor to run at base speed **B** and at maximum velocity **M** respectively until commanded to stop. These commands are generally used in combination with the **IF, WAIT** and **STOP** commands.

The **RUN** command will cause the motor to run continuously at the base speed **B**, in the currently set direction until the **STOP** command is issued. An **<ESC>** key will also interrupt the move. The **RUN** command can only be issued when the motor is at a standstill. The position status is continuously updated in parameter **P**. All inputs and outputs will be functional.

When **SLEW** is executed, the motor will accelerate to maximum velocity **M**, in the currently set direction, until the **STOP** command is issued. An **<ESC>** key will also interrupt the move, but the parameter **P** may loose synchronism due to the resulting abrupt stop. This command can only be executed when the motor is at a standstill. The position status is continuously updated in parameter **P**. All inputs and outputs will be functional.

Example Example of the RUN command:

```
(AA) POSITION 0 ^ Move to position zero
- RUN           ^ Begin to run in the CCW direction
SET 3           ^ Set output 3
WAIT 100        ^ Cause a 1 second program pause
RESET 3         ^ Reset output 3
(AB) UNTIL 1 AB ^ Poll input 1 until it becomes active
STOP            ^ Stop the motor when input 1 is active
REPORT          ^ Report when motor has stopped
WAIT 500        ^ Cause a 5 second program pause
LOOP 9 AA ^ Cause program to loop 9 times to label AA
QUIT
```

Example Example of the SLEW command:

```
(AA) POSITION 0 ^ Move to position zero
- SLEW          ^ Begin to slew in the CCW direction
SET 3           ^ Set output 3
WAIT 100        ^ Cause a 1 second program pause
RESET 3         ^ Reset output 3
(AB) UNTIL 1 2 AB
^ Poll inputs 1 and 2 until they become active
STOP ^ Stop the motor when inputs 1 and 2 are active
REPORT          ^ Report when motor has stopped
WAIT 500        ^ Cause a 5 second program pause
LOOP 9 AA ^ Cause program to loop 9 times to label AA
QUIT
```

Encoder Capabilities

All ICL indexers interface with incremental encoders to provide the user with **Stall Detect**, **Position Maintenance** and **Home on Z-Channel** capabilities. Depending on your application, the use of encoders may be either required or desired. If you choose to use an encoder, you need to determine how it will be used and then follow the appropriate set-up configurations as described in this section.

If you will not be utilizing an encoder you may skip the following section.

Stall Detect

If this function is enabled the system will either provide you with a message that the motor has stalled and turn power off to the motor, or it will automatically branch and execute a predefined user program defined by stall program parameter **SP**.

The motor is considered **STALLED** when the instantaneous encoder error exceeds the value of ICL Parameter **PE**, position error.

Position Maintenance

With this option enabled, ICL indexers will correct for final positional accuracy after a move is completed. After a move is completed, the ICL System will correct for any positional error between the commanded position and the encoders actual position. This position correction will take place when ever the encoder position is outside the deadband window set by parameter **DW**. The ICL System will continue to send pulses to the drive until the motor is in position according to the encoder. Thus the ICL System will correct for any backlash that may be present between the motor and encoder location.

Once the motor is on a commanded position, any outside forces that cause the motor to move outside the deadband window will cause the ICL System to step the motor until it is within the deadband window.

With position maintenance it is necessary to define the correction gain **CG** and the maximum correction velocity **CV** that a position maintenance move will be allowed to achieve. If the correction gain value is too large, position correction may be too slow. If the correction velocity is to small, position correction may be too slow. These values are best determined while tuning the ICL System for your application.

Hazard Note

Use an encoder checking routine on each power-up to determine functionality of the encoder configuration prior to turning the encoder functions ON. This can prevent a potential run-away system. See the example of encoder checking program ECKH.ICL in this section. We suggest use of the STALL DETECT feature whenever using the POSITION MAINTENANCE feature. This will allow the user to control when position maintenance should be considered.

TERMINATION OF THE AC POWER IS THE ONLY SAFE WAY TO PREVENT UNWANTED MOTION WHEN POSITION MAINTENANCE IS ENABLED.

Home on Z-Channel

After locating the home switch during a search for home, the ICL System will back the motor up over the home switch to the leading edge of the first index or Z-channel of the encoder.

The home switch and Z-channel must both be active or the ICL System will report **"#23 HOME NOT FOUND"**. This feature provides the user with a more accurate home position with respect to the measurement system (i.e. the encoder). Home is now referenced with respect to the encoder rather than the home switch.

General Parameter Setup

Description When using any of the encoder functions, the ICL Parameters **ER** - Encoder Resolution and **MR** - Motor Resolution must be defined for the ICL System to operate correctly.

1. **Motor resolution is established by dip switch setting on the drive**. We will assume that the drive is configured for 20000 steps per revolution, thus we will set **MR=20000**.
2. The standard encoders supplied by the Controls Division are 1000 line encoders. After going through the quadrature detection circuit of the indexer it will have 4000 distinct locations per revolution, thus **ER=4000**. In cases where the encoder is not directly mounted to the motor, the user must determine the number of encoder pulses that will be received when the motor is displaced one shaft revolution.
3. Prior to performing a move we must also set motion parameters **A**, **B**, **D**, **H**, and **M**.

Example

EF=00000000	Sets all encoder functions off.
MUNITS	Sets move unit of measure to motor pulses.
A=1000	Sets acceleration from "B" speed to "H" speed to 1 second.
B=10000	Sets base speed at .5 revolutions per second (RPS).
D=1000	Sets decel rate to 1 second or less.
H=500000	Sets parabolic ramp high speed at 25 RPS.
M=400000	Sets maximum velocity to 20 RPS.
MR=20000	Sets motor resolution to 20,000 steps per rev.
ER=4000	Sets encoder resolution to 4,000 counts per rev.
E=0	Sets encoder position to zero.
P=0	Sets motor positions to zero.
PAUSE	Set processing mode to single line.
MOVE 20000	Move the motor 20,000 steps.
VER P VER E	Verify the motor and encoder position.
"P = 20000"	Motor position is 20,000.
"E = 4000"	Encoder position is 4,000.

Hazard Note

If "E = -4000" the encoder wires for "A+" and "A-" must be switched and the example re-run. The value of "E" must be positive when moving in a positive direction.

See the example of encoder checking program ECKH.ICL in this section.

Encoder Checking Program

Enter the ECKH program on your indexer and include the comments as shown. When executed the ECKH program will rotate the motor clockwise then counter clockwise 1/10 revolution and verify the positions of the encoder. If the encoder positions are not correct then the program will send a message top the host and ABORT. The program is executed via the CALL ECKH command and can be called from within the main program as shown in the example below.

Encoder Checking Program

```
EDIT ECKH
                    ^ Horizontal encoder check routine
EF=00000000         ^ Turn off all encoder functions
PAUSE               ^ Enable pause mode
R=0                 ^ Disable power reduction
S=0                 ^ Disable power off
MUNITS              ^ Enable motor units
MOV 1               ^ Command drive to powerup
MOV -1WAIT 200           ^ Wait for motor to be stable
P=0                 ^ Set current motor position to zero
E=0                 ^ Set current encoder position to zero
DW=10               ^ Set dead band window to ten encoder counts
M=MR/2              ^ Set velocity to 1/2 rps
MOVE MR/10          ^ Move one tenth revolution CW
WAIT 100            ^ Wait for one second
IF (( E-ER/10 < DW) AND (ER/10-E < DW )) PASSCW
^ Verify encoder is within dead band window
^ Go to label pass-cw
JUMP ERROR
^ Go to label error if not in dead band window
(PASSCW) MOVE -MR/10 ^ Move one tenth revolution CCW
WAIT 100            ^ Wait for one second
IF (( E < DW) AND ( E > -DW)) PASSCCW
^ Verify encoder is within dead band window
(ERROR) SEN -1 "THE SYSTEM DID NOT PASS PROGRAM ECK"
SEN -1 "**** PROGRAM ABORTED **** "
ABORT               ^ Abort program
(PASSCCW) SEN -1 "ENCODER HAS PASSED TEST"
                    ^ Send message to host device
QUIT                ^ Last line of program
```

<**ESC**> Terminate the ICL Editor and return to immediate mode.

Main Program

```
EDIT MAIN
CALL CONFIG          ^ Execute the configuration program
CALL ECKH             ^ Execute the encoder check program
DW=5                 ^ Reset encoder dead band window
EF=11010000          ^ Turn on desired encoder functions
^   Motion program starts
.
QUIT                 ^ Last line of program
```

<**ESC**> Terminate the ICL Editor and return to immediate mode.

> ***Hazard Note***
>
> Do not continue until you successfully pass the ECK program(s) in this Users Guide.
>
> Note: Personal injury and damage to you equipment may result by ignoring this warning!

Position Maintenance

Description To utilize the position maintenance capability of the ICL System the user must define parameters for the Dead Band Window - **DW**, the Maximum Correction Velocity - **CV**, and the Position Correction Gain - **CG**. The dead band window is defined as the encoder error associated with a commanded physical position. In the case where **DW=5**, the motor is considered on position when the encoder is within five counts of the commanded position. To prevent the ICL System from "hunting for position" the Dead Band Window has a pre-defined minimum value based upon Motor Resolution and Encoder Resolution.

For MR < ER
The minimum value is calculated by the following:
DW = {Integer(**ER**/**MR**)} * 2 +1

For MR > ER and MR < 3*ER
The minimum value is "3" due to the mechanical variations in the motor.

For MR > 3*ER
The minimum value is "0".

1. For our example **ER = 4000** and **MR = 20000**, thus the minimum value of the Dead Band Window is **DW = 0**.
2. Next we must define the Maximum Correction Velocity, **CV**, that will be allowed during error correction. For the case where **ER=4000**, and **CV=40000**, the Maximum Correction Velocity is ten revolutions per second, (**CV** is entered in encoder counts per second).
3. The Correction Gain parameter **CG** is utilized to automatically select a correction velocity based upon the motor position error, **ME**. The selected correction velocity will be less than or equal to **CV**, the Maximum Correction Velocity. Enter the following; **CG = 100**.
4. It is recommended that the user define his current position as zero before enabling position maintenance. Bit #2 of the Encoder Function parameter **EF=x1xxxxxx** is used to enable position maintenance. In our example we will move the motor a distance of three shaft revolutions and then manually move the motor shaft to simulate how the ICL System will return the motor to the original position. Note that because of the torque output of some motors you would need to put an inertia wheel on the motor shaft to allow you to turn the shaft.

Example

PAUSE, ER = 4000, MR = 20000

DW=5	Sets dead band window to five motor pulses.
CV=40000	Sets the maximum correction velocity to 10 RPS.
CG=100	Set the correction gain value.
MUNITS	Sets move unit of measure to motor pulses.
P=0 E=0	Sets motor and encoder positions to zero.
UP=0	Sets user position to zero.
EF=010XXXXX	stall detect and following off.
MOVE 60000	Move the motor shaft three revolutions.

At this point, if you were to move the motor shaft, you would see it correct to its intended position (P=60000).

Hazard Note

Disable position maintenance, **EF=x0xxxxxx** whenever you change the position register or encoder register. This will prevent the ICL system from auto-correcting or reading an error during the time it takes you to enter the new values.

Stall Detect

Description To utilize the Stall Detect capability of the ICL System we must define parameters for the Maximum Position Error - **PE**, and the Stall Detect Program Number - **SP**. The Maximum Position Error is defined as the encoder error associated with a commanded physical position. In the case where **PE=2000** the motor is considered stalled if the instantaneous encoder position differs by more than 2000 encoder counts of the commanded position. To prevent the ICL System from a false stall condition, the value of **PE** must be large enough to account for the mechanical compliance of the system, (couplings, backlash).

1. In our example the motor is considered stalled if the position error is one-half shaft revolution or greater. **PE=2000**.
2. Bit #1 of the Encoder Function parameter **EF** is used to enable the stall detect function, **EF = 1xxxxxxx**.
3. When a stall condition is detected, the ICL System will immediately transfer control to the program defined by the Stall Program Number parameter - **SP**. If you do not wish to execute a program, then parameter **SP** should be set to **-1**. In this case the system will respond to you with the message **#36 - MOTOR HAS STALLED**. For our example purposes we will not use a stall program and will default to the standard system message. Note that because of the torque output of some motors you would need to put an inertia wheel on the motor shaft to allow you to stall the motor.

Example

NOVICE
ER = 4000
MR = 20000

DW=5	Sets dead band window to five motor pulses.
CV=40000	Sets the maximum correction velocity to 10 RPS.
CG=100	Set the correction gain value.
P=0 E=0	Sets motor and encoder positions to zero.
UP=0	Sets user position to zero.
PE=2000	Sets position error to .5 shaft revolutions.
SP=-1	Sets system to send message only.
EF=100XXXXX	Sets stall detect on. Sets position maintenance off.
UR=1	Sets user resolution to 1 count per shaft revolution.
UUNITS	Selects user units as the move unit of measure.
MOVE 75	Move 75 shaft revolutions (1,500,000 pulses).

If you were to stall the motor, the following message should appear.

"#36 MOTOR HAS STALLED"

Main Program

```
EDIT MAIN
CALL CONFIG          ^ Execute the configuration program
CALL ECK             ^ Execute the encoder check program
DW=5                 ^ Reset encoder dead band window
SP=8                 ^ Program #8 is executed upon a stall condition
EF=11010000          ^ Turn on desired encoder functions
^   Motion program starts
.
QUIT                 ^ Last line of program
```

<**ESC**> Terminate the ICL Editor and return to immediate mode.

(continued) **Stall Detect**

STALL Program

```
EDIT STALLH
EF=00XXXXXX          ^ Turn stall detect and pos maint off
STOP                 ^ Stop motion
SEND -1 "STALLED"    ^ Send message to host device
F 13 ERROR           ^ Go to label error if stop input is active
^ The user may insert conditionals to decide to correct
^ Or go to label error
EF=X1XXXXXX          ^ Turn pos maint on, let it work
(A) UNT ((F0\) AND ( (ME<((DW*MR/ER)+1)) AND (ME>((-DW*MR/ER)-1)))) A
P=P                  ^ Set position counter to current position
EF=1XXXXXXX          ^ Turn encoder stall detect on
QUIT                 ^ Quit subroutine
(ERROR) SEND -1 "STOP INPUT ACTIVE, STALL & POS MAINT TURNED OFF"
ABORT                ^ Abort programs
QUIT                 ^ Last line of program technote # 1006
<ESC> Terminate the ICL Editor and return to immediate mode.
       Listed as program number 8 in the directory.
```

Programming Hints

Error Message #20 CALL STACK OVERFLOW

The AUTOSTART program is a first level CALL command.

When using an Encoder Stall Program the maximum number of nested CALL statements is reduced to three. The Stall Program becomes the fourth CALL routine.

Avoid recursive CALL statements / programming.

Program A	CALL PG2
Program PG2	CALL A

Home on Z-Channel

Description If bit #4 of the encoder function is active, **EF=xxx1xxxx**, during a search for home, the indexer will continue over the home limit switch once it is found until the edge of the index or Z-channel of the encoder. The position parameter **P**, encoder position parameter **E** and user position parameter **UP** would then be set to **0**. See the **HOME** command for graphic examples of how this function would work.

Both the Home input and the Z-Channel must be active or the following message will appear.

#23 HOME NOT FOUND

MOTION PROGRAMS

Stored motion programs provide the user with the ability to perform repetitive functions over and over again without having to type each individual command line every time you wish to perform the same function. A program can be defined as a series of commands or instructions and parameter values that tells the indexer "WHAT TO DO" and "HOW TO DO IT". Since the indexer is a computer, the stored command sequences can be run many times with predictable results. On the ICL System all the tools have been provided to allow the user to create, modify and store up to 88 motion programs within it's Random Access Memory (RAM). Programs on the indexer are stored and recalled by assigning a 1-8 character name to the program when created with the ICL Editor. You are not allowed to have duplicate program names. To obtain a listing of all previously created motion programs the **DIRECTORY** command is used.

An important feature found within the indexer includes the ability for one program to **CALL** another program. This feature, sometimes referred to as subroutines, provides the user with the ability to; 1) segment a large program into smaller more manageable segments, and 2) allows transfer of control to different subroutines based on conditions within the calling program.

Another feature called —**AUTOSTART**, allows the user to begin executing a pre-defined program when powered-up. This provides the user with the ability to use the indexer in the stand-alone mode without being tied into an RS-232 host or programming device.

All of these features will be explained in greater detail in subsequent sections.

Creating Programs

The ICL editor allows the user to create, modify, and delete motion programs on indexer. The program will be stored at the lowest available program number not currently being used within the indexer's memory, filling any holes that may be left open because of deleted programs.

Enter the ICL Editor by typing the **EDIT** *ccc* command, where *ccc* is an alphanumeric program name from 1 - 8 characters. If the program already exists, (the name entered exactly matches a program name in the indexer's memory), this program will be recalled to allow you to begin editing of the existing program. If you are creating a new program, (the name entered does not match a program name currently in the indexer's battery backed memory), you may now begin to create the new program by entering the desired commands.

<Backspace>
<CTRL-H> Removes one character from the end of the current edit line. This key has the same function in the command mode.

<Esc> **"Escape Key" on the host key board. The <ESC> key is used to exit the ICL editor.** In the ICL Editor the last edit line must be entered into a program by pressing the **<Return>** key, before the **<ESC>** key will be recognized. Note: If the users host device does not have an **<ESC>** then **<CTRL-[>**, (control and left bracket sign) may be substituted.

<Return> Enter the current edit line into the program.

<CTRL-X> Deletes the current line. The editor will display the next sequential line within the program. **<CTRL-J>** Moves down one line and display its contents. It will also prompt the user with an echoed line number for any requested changes.

<CTRL-K> Moves up one line and display its contents. It will also prompt the user with the echoed line number for any requested changes.

<CTRL-N> Inserts a new line before the current line currently being displayed. The line number will be echoed as a blank line and the rest of the motion program will be indexed down by one line number.

=n Moves to line number **n** and display its contents. The editor will then prompt the user with the echoed line number for any requested changes. Do not use a delimiter in this editor command.

<Backspace>
<CTRL-H> Removes one character from the end of the current edit line. This key has the same function in the command mode.

The SAMPLE1 program that we will create, will set the parameters for motion and wait for input #1 to become active. The cycle will consist of eight indexes of 19200 pulses clockwise with a six second pause between indexes for an outside operation to be completed. The outside operation will be triggered by output #1 and the program will abort if input #2 is active. The drive will also be commanded to switch to low-power mode three seconds after each move. The moves will start at 1/10 revolutions per second to a maximum speed of 10 revolutions per second. After the group of indexes are complete the motor will return to home position.

SAMPLE1

Sample Program

```
EDIT SAMPLE1 <Return>
0 :
0 :A=4000 D=4000 R=300 B=1280 H=20000 PAUSE <Return>
1 :
1 :M=20000 Q=10 ^ The icl parameters are now set <return>
2 :
2 :Move 0 ^ False move to force calculate ramp tables <return>
3 :
3 : ^ This prevents a delay when input 1 becomes active <return>
4 :
4 :(START) UNTIL 1 START ^ Wait until input 1 is active <return>
5 :
5 :(AA) MOVE 19200 ^ Move 1.5 revolutions clockwise <return>
6 :
6 :SET 1 ^ Make output <return>
7 :
7 :WAIT 100 ^ Wait 1 second for external device to recognize <ret>
8 :
8 :RESET 1 <Return>
9 :
9 :WAIT 500 ^ Wait balance of 6 seconds <return>
10 :
10 :IF 2 BB ^ Check input 2 if active go to label bb <return>
11 :
11 :LOOP 8 AA ^ Repeat cycle 8 times <return>
12 :
12 :POS 0 ^ Return to home or position zero <return>
13 :
13 :REPORT ^ Report motor idle and ready for input 1 <return>
14 :
14 :JUMP START ^ Branch to label start <return>
15 :
15 :(BB) ABORT <Return>
16 :
16 :<ESC>
0>
```

The display above is the same you will receive when entering this program. You will note that the ICL EDITOR will display what it currently has stored in line number and then asks what you want in that line number. This results in the notation as nn : and an echo of nn : for you to insert the replacement command string.

Begin program SAMPLE1 by typing, **CALL SAMPLE1 <Return>**.

Activate input #1 to begin cycle, Note the controller will report MOTOR IDLE when it is ready for the next trigger on input #1. Output 1 will trigger for one second after each 19200 pulse index. If input 2 is active at the end of the 6 second wait or the <ESC> character is received during any index the program will abort to the command mode.

Utilizing Subroutines

An important feature found within the indexer includes the ability for one program to **CALL** another program. This feature, sometimes referred to as subroutines, provides the user with the ability to; 1) segment a large program into smaller more manageable segments, and 2) allows transfer of control to different subroutines based on conditions within the calling program.

Upon completion of the called subroutines, control will return to the next line of the calling program. Although there is no limit on the number of subroutines used within a program, you are limited to a maximum nesting of 4 levels deep.

In the following example we will create a subroutine program called SETUP to initialize the ICL parameter values, edit SAMPLE1 to call this subroutine. This method of utilizing a SETUP program will reduce the number of times that you must reenter parameters.

SETUP Program

```
EDIT SETUP <Return>
0 :
0 :HOM ^ home axis <return>
1 :
1 :A=4000 <Return>
2 :
2 :D=4000 <Return>
3 :
3 :R=300 <Return>
4 :
4 :B=1280 <Return>
5 :
5 :H=20000 <Return>
6 :
6 :M=20000 <Return>
7 :
7 :Q=10 ^ The icl parameters are now set <return>
8 :
8 :MOV 0 ^ Make a false move, calculate ramp tables <return>
9 :
9 :<ESC>
0>
```

(continued) **Utilizing Subroutines**

We will now edit program SAMPLE1 to include the **CALL** to the **SETUP** program. We will use the ICL Editor commands **<CTRL-X>** to delete and **<CTRL-K>** to display what the new command string is for that line. Enter the new command line **CALL SETUP** to replace line "3 : ^ THIS PREVENTS A DELAY ....". Then **<ESC>** back to the command mode.

```
EDIT SAMPLE1 <Return>
0 :HOM A=4000 D=4000 R=300 B=1280 H=20000
0 :<CTRL-X>~M=20000 Q=10 ^ The icl parameters are now set
0 :<CTRL-X>~MOV 0 ^ False move to force calculate ramp tables
0 : ^ This prevents a calculation delay when input 1 becomes active
0 :CALL SETUP
1 :(START) UNTIL 1 START ^ Wait until input 1 is active
1 :<ESC>
0>
```

The **<CTRL-X>** character is typed when deleting a command line, in the example shown above the next line is now displayed overwriting the deleted one. If a listing of SAMPLE1 is done you will now see that we have deleted the ICL parameters and added a **CALL SETUP** subroutine. Type **CALL SAMPLE1**, the controller will now call SETUP to set the parameters then return to **SAMPLE1** to continue processing. Call and subroutines may be nested up to four (4) levels deep.

Programming Hints

Error Message #20 CALL STACK OVERFLOW

The AUTOSTART program is a first level CALL command.

When using an Encoder Stall Program the maximum number of nested CALL statements is reduced to three. The Stall Program becomes the fourth CALL routine.

Avoid recursive CALL statements / programming.

Program A	CALL PG2
Program PG2	CALL A

Using an Autostart Program

The autostart program feature on the indexer allows the user to designate the program number to be executed when the power is first applied to the unit. The program to be executed is defined by entering the program number into ICL Autostart Parameter **G**. To obtain the program numbers type the **DIRECTORY** command; find the name of the program you wish to run as the **AUTOSTART** program, and read it's assigned program number. The program number may be from 00 to 87 as shown in the directory listing and may be loaded into parameter **G** by using the **ENTER** command. If parameter **G** is defined as **-1**, then no autostart program will be executed.

This feature is particularly handy when you wish to use the ICL System indexer in a stand-alone mode, i.e. not interfaced to a host device on the RS-232 serial communications port. The sample programs that follow allows the user to use the ICL System indexer as a stand-alone device once it has been programmed via its RS-232 serial interface.

An application example follows:

For illustration purposes, let's assume that a manufacturer of tape wants to automatically wind three different spool sizes on the same piece of machinery. To accomplish this task we will use programmable inputs #1, #2 and #3 on the indexer to determine which spool to wind. We will create four programs with the names and functions of each defined as follows:

Program Function

MAIN Set parameter values, determine which spool to wind and to wait until done.

R25 Subroutine for winding spool #1

R35 Subroutine for winding spool #2

R50 Subroutine for winding spool #3

Our examples assume full step mode or 200 steps per shaft revolution. The program listings follows:

MAIN Program

```
EDIT MAIN
A=400          ^ Accel in .4 seconds
B=400          ^ Set base speed to 2 rev/second
D=400          ^ Decel in .4 seconds
H=10000        ^ Set highest speed for ramping
M=8000         ^ Set maximum velocity for 2400 rpm
PAUSE          ^ Set pause mode
MOVE 0         ^ Force calculation of ramp tables
(START) UNTIL 6 START ^ Poll cycle input 6 until active
IF 1 P1        ^ If input 1 active go to label p1
IF 2 P2        ^ If input 2 active go to label p2
IF 3 P3        ^ If input 3 active go to label p3
QUIT           ^ If inputs 1, 2 and 3 inactive then quit
(P1) CALL R25 ^ Call subroutine r25
JUMP START     ^ Branch to start for cycle input
(P2) CALL R35 ^ Call subroutine r35
JUMP START     ^ Branch to start for cycle input
(P3) CALL R50 ^ Call subroutine r50
JUMP START     ^ Branch to start for cycle input
QUIT           ^ Last line of main program
```

(continued) **Using an Autostart Program**

R25 Program

```
EDIT R25
MOVE 5000   ^ Move 25 revolutions (25*200)
SET 1       ^ Turn on output 1 for 50ms to notify an external
WAIT 5      ^ Device move has been completed
RESET 1     ^ Turn off output 1
QUIT        ^ Return to main program
```

R35 Program

```
EDIT R35
MOVE 7000   ^ Move 35 revolutions (35*200)
SET 1       ^ Turn on output 1 for 50ms to notify an external
WAIT 5      ^ Device move has been completed
RESET 1     ^ Turn off output 1
QUIT        ^ Return to main program
```

R50 Program

```
EDIT R50
MOVE 10000  ^ Move 50 revolutions (50*200)
SET 1       ^ Turn on output 1 for 50ms to notify an external
WAIT 5      ^ Device move has been completed
RESET 1     ^ Turn off output 1
QUIT        ^ Return to main program
```

In our example you can envision a panel with a rotary selector switch labeled for inputs #1, #2 and #3 on the machine control panel. Switch 1 is connected to input #1 on the indexer, switch 2 is connected to input #2 on the indexer and switch 3 is connected to input #3 on the indexer. Input #6 on the indexer is connected to a cycle start sensor internal to the winding machine and the output #1 may be connected to an external device or counter to record when a move has been completed and a spool of tape wound. The **STOP** input may also be connected to a internal machine switch to allow for program termination if any problems occur.

The **DIRECTORY** command is issued to determine the program number of MAIN.
DIRECTORY

```
0 |MAIN          1 |R25          2 |R35          3 |R50
4 |              5 |...
```

To establish "**MAIN**" as the **AUTOSTART** program, type **G = 0**

When the indexer is first powered-up the program "**MAIN**" will be executed and the controller will wait until input #6 is active before it polls input #1 if "**ACTIVE**" the user is prompted on instructions on spool selection. The operator will then manually input spool data. To begin the actual winding operation input #6 must be activated by the cycle start switch.

Hazard Note

Do NOT assign an autoexecute program until you have fully debugged your programs. This will prevent you from accidentally creating an endless-loop routine and locking the controller. A badly corrupted operating system may require factory service.

(continued) **Using an Autostart Program**

Try the following example using variables

```
EDIT MAIN
MR=200                  ^ Motor resolution
A=400                   ^ Accel
B=400                   ^ Set base speed to 2 rev/second
D=400                   ^ Decel
H=10000                 ^ Set highest speed for ramping
M=8000                  ^ Set maximum velocity for 2400 rpm
PAUSE                   ^ Set pause mode
MOVE 0                  ^ Force calculation of ramp tables
(START) UNTIL 6 START   ^ Poll cycle input 6 until active
IF 1 QA                 ^ If input 1 active go to label qa
(A) MOVE V1             ^ Move V1 distance
SET 1         ^ Turn on output 1 for V2 ms to notify an external
WAIT V2                 ^ Device move has been completed
RESET 1                 ^ Turn off output 1
JUMP START              ^ Branch to start for cycle input
(QA) PROMPT "ENTER MOVE DISTANCE" V1
PROMPT "ENTER TIME DELAY IN MILLISECONDS" V2
JUMP A
QUIT
<ESC>
```

Debugging a Motion Program

If a mistake is made in your motion program, such as a syntax error or bad command name, the ICL System will respond back to you at run time the error type that has been detected. Three ICL parameters have been assigned to facilitate quick and simple location of the command line that produced the error condition. See the **TRACE** command.

Current Program # W

This parameter cannot be changed by the user, (Read only). This parameter contains the program number of the current or last program executed. This parameter is utilized to locate a program or subroutine when debugging a motion routine. The **DIRECTORY** command is utilized to locate the program name associated with the program number.

EXAMPLE: **VERIFY W**

W = 4

The last program executed was program number 4 on your directory.

Current Program Instruction Address X

This parameter cannot be changed by the user, (Read only). This parameter is utilized to locate the program instruction address of the current or last instruction executed in a motion program. This parameter is utilized to help locate errors within a motion program.

EXAMPLE: **VERIFY X**

X = 5258
The last instruction executed was at address 5258 in program number **W** on your directory.

Current Program Line Number Z

This parameter cannot be changed by the user, (Read only). This parameter is utilized to locate the program line number of the current or last program line executed in a motion program. This parameter is utilized to locate an error within a motion program.

EXAMPLE:
While executing a program, the ICL system reports **"#17 LABEL TO LONG"** error.

VERIFY W VERIFY X VERIFY Z

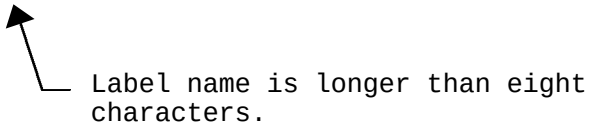
W = 4
X = 5258
Z = 20

DIRectory

```
 0|SETUP        1|PROG1      2|PROG2     3|TEST
 4|SAMPLE1  5|SAMPLE2     6|SAMPLE3 7|TEST1
 8|TEST2       9|TEST3 ...
```

The listing of program 4, "SAMPLE1" shows line 20 and address 5258 is the location of the error.

```
LIST SAMPLE1
1 5136 A=1000
.
20 5249 (LABELTOLONG) UNTIL 2 CYCLE
```



INTERFACING TO YOUR ENVIRONMENT

In their simplest form, the indexer's inputs and outputs (I/O's), provide the ability for interaction between the indexer and external devices. Types of external devices include other indexers, step motor drives, programmable controllers, limit switches, relays, status lights (LED'S), foot pedals, counters, optical sensors, pressure sensors, etc. The state of all I/O's can be defined as **ACTIVE** or **INACTIVE**. When the status of the inputs change, this change in state can be interpreted for taking appropriate action.

The inputs and outputs of the indexer can be separated into the following groups; **Driver Outputs**, **Programmable Inputs** and **Programmable Outputs**. A description of each group will follow.

Driver Outputs

The "step" and "direction" drive output signals generated by the indexer are the most common type of driver input signals. Other drive outputs may include "low-power", "no-power", "reset" and "+5VDC". If you are using a drive/indexer package you will find that the driver outputs are pre-wired internally, and are not visible from outside of the enclosures. If a stand alone ICL indexer is being utilized you must refer to the indexer User Guide for the proper wiring to the selected drive.

Note that the **RESET 0** command is utilized to command a reset signal to the users drive. This reset signal is required when the user has changed step resolution of the drive or if the drive is being asked to recover from a fault condition.

Limit Inputs

Limit inputs available on an ICL system include Jog+, Jog-, Home, Clockwise Limit (CWL) Counter Clockwise Limit (CCWL) and Stop. In this section we will define the input names, assigned numbers, the method to set the default state of these inputs, and how the inputs may be utilized in the users system. When utilizing limits the ICL parameter, limit action, **LA** is used to define "how they are used". Inputs that may be assigned are Jog minus input number **JM**, Jog plus input number **JP**, and stop input number **SI**. Detailed information on each of these parameters can be found in the ICL parameter definition section of this user guide.

When utilizing a home sensor switch you should review how the home command functions in the ICL command definition section of this user guide. Practical examples of using limits in an ICL system follow.

In this example the user requires that the CW, CCW and HOME limits be active **HIGH**, (i.e. limits will be active if they are not wired to a normal closed switch). Inputs 5, 6 and 7 are to be utilized as the JOG+, JOG- and STOP functions. The program is to exit to the immediate mode if limits occur at any time after the system has found home.

Example

```
LA=XXX00111 ^ Set SOFT action for limits
ID2=XXXXX111 ^ Set CW, CCW and HOME active high
A=1000 D=A B=2000 H=400000 J=4000
M=30000 ^ Set a maximum velocity to search for HOME
HOME ^ Search for HOME input
M=300000 ^ Set maximum system velocity
JP=5 JM=6 SI=7 ^ Assign inputs for JOG+, JOG- and STOP
LA=XXX11000 ^ Set to exit program on limits.
```

Programmable Inputs

Programmable inputs give the user the ability to interface the indexer with its environment. This allows the user to control motion on the indexer based on the external signals. The types of external devices include relays, foot pedals, optical sensors, pressure sensors, programmable controllers, counters etc. Inputs on an ICL indexer are grouped in banks of eight. If you have eight inputs you would have one bank, 9-16 inputs two banks, 17-24 inputs three banks etc. The user guide for specific device will define the number available on your unit. The ICL parameters that you need to be aware of are defined as follows:

Parameters **ID1** to **ID[x]** provides you with the ability to set default state.

The ICL commands that allow interrogation of programmable inputs are if, **IOSTATUS**, **UNTIL** and **VERIFY.**

Practical examples follow:

Special inputs **JOG+**, and **STOP** may be assigned as required by the application. To do this the input number must be assigned to the appropriate ICL parameter name as denoted: **JOG+ (JP), JOG- (JM)** and **STOP (SI)**.

Example `JP=11 JM=12 SI=13`

Assign **JOG+** to input # 11, **JOG-** to input # 12 and **STOP** to input # 13. If one or all of the special inputs is declined then the appropriate parameter value would be 0.

```
SI=0  ^ STOP Input is declined.
```

Input States

The **CW, CCW** and **HOME** limits are normally open with the pins internally pulled high at voltage OPTO. Limits are active when the normally open input switches are closed by taking them to the ground of the OPTO supply.

ICL Parameters **I1** and **I2** define the current status of the programmable and limit inputs while ICL parameter **ID1** and **ID2** define the default state of all inputs.

Each parameter is represented by an eight (8) bit number that defines the default state of the inputs. **ID1** defines the default date of programmable inputs 1-8 and **ID2** defines the defaults state of programmable inputs 9-13, CWL (14), CCWL (15) and HOME (16). In the following example the user has determined that an application requires a normally closed switch on inputs 2, 3 and 4. When the switches are opened, the ICL System will interpret them to be **ACTIVE**.

1) Check the current status of input definitions, **VERIFY ID1 <Return>.** The host device will display a string **ID1=00000000**, showing that the default states of inputs 1 to 8 is "open" or floating high, and must be taken low to become **ACTIVE**.

2) **ID1**, the input definition parameter, is used to set the selected default states. All eight values must be entered or the command will not be accepted by the controller. Type the following: **ID1=01110000 ID2=10000111 <Return>**

3) The ICL System will now interpret the inputs defaults as: 1, 5, 6, 7, 8, 10, 11, 12 and 13 as active low, and 2, 3, 4, 9, 14, 15 and 16 as active high. The definitions are retained in the battery backed RAM.

4) Check the new status of the inputs with the **VERIFY I** command. The host will now display **I=01110000**, **I2=10000111**. The input status now shows that #2, #3, #4, #9, #14, #15 and #16 are **ACTIVE**. Once the user attaches a normal closed switch to these inputs you may check the status again and all inputs will be **INACTIVE**.

Programmable Outputs

In this section we will define the output names, assigned number and the method by which the user can set the default states.

PROGRAMMABLE OUTPUTS:

NAME	NUMBER	TYPE
OUTPUT 1	1	sinking
OUTPUT 2	2	sinking
OUTPUT 3	3	sinking
OUTPUT 4	4	sinking
OUTPUT 5	5	sinking
OUTPUT 6	6	sinking
OUTPUT 7	7	sinking
OUTPUT 8	8	sinking

When outputs 1 to 8 are open the pins are high internally at voltage OPTO 2 and when "ACTIVE" the pins are sinking current to OUTPUT COMMON. The user may decide that the output default state needs to be changed. The example following describes how this is accomplished.

The parameter **OD**, Output Definition, is represented by an eight (8) bit number that defines the default state of the outputs. The user has determined that the application requires two sinking outputs to be **ACTIVE** low .

1) Check the current status of outputs, **VERIFY OD1 <Return**>. The host device will display a string **OD1 = 00000000**, showing that the default states of the outputs. Sinking outputs 1 to 9 are "open" and pulled high to voltage OPTO 2. When **ACTIVE** they will be sinking current to OUTPUT COMMON or "low".

2) **OD1**, the output definition parameter is used to set the selected default states. All eight values must be entered or the command will not be accepted by the controller. Type the following: **OD=x11xxxxx <Return>**

3) The ICL System will now interpret the output defaults as: 1, 4, 5, 6, 7 and 8 as active when sinking current, outputs 2 and 3 are active when open (not sinking current). The definitions are retained in the battery backed RAM.

4) Check the new status of the outputs with the VERIFY O1 command. The host will now display O=01100000.

INDEX

APPENDIX

INTELLI-COMMAND LANGUAGE SUMMARY

The following listing of commands are available on the indexer. The listing includes an indication as to weather the command responds to immediate execution from command mode “I”, stored in motion program “S”, or both.

ICL COMMAND DESCRIPTION

ABort (IS)..... Abort current program or move command.
AUtostart (IS) ..... Auto-start program number of parameter “G”.
BOOT (I) ..... Boot controller, erase all RAM memory.
Call ccc (IS) ..... Call program label “ccc”.
CLEar [v] (IS) ..... Clear variable “v”.
CONtinue (IS) ..... Enable Continuous processing mode.
DElete ccc (I) ..... Delete program label “ccc”.
DIrectory (IS) ..... Directory is listed to host device.
DUmp ccc n (I) ..... Dump program “ccc”, format “n” to host device.
ECHo [ON/OFF] (IS) ..... Enable/Disable echo mode.
EUNits (IS) ..... Enable encoder units for motion commands.
EDit ccc (I) ..... Edit program “ccc” or create program “ccc”.
ENter p n (IS) ..... Enter parameter “p” the value “n”.
EXpert (I) ..... Sends message numbers only to host device.
Go (IS) ..... Go last indexed distance.
HElp (I) ..... Display help screen.
HOme (IS) ..... Search until home input is active.
IF (Condition) (S) ..... Label “ccc” if “n” is active.
IO [-n/n] (IS) ..... List input or output states.
JUMp ccc (S) ..... Unconditional branch to label “ccc”.
LIst ccc (I) ..... List program “ccc” to host device.
LOop n ccc (S)..... Loop “n” times to label “ccc”.
Move [-]n (IS)..... Move relative “n” steps.
MUNits (IS) ..... Enable motor units for motion commands.
Novice (I) ..... Sends message and numbers to host device.
PAUse (IS) ..... Disable continuous processing mode.
PASsword (I) ..... Enable/Disable password protection of files.
Position [-]n (IS) ..... Move to absolute position “n”.
PROmpt “ccc” v (IS) ..... Prompt the user with message “ccc” and input answer into variable/parameter “v”.
Quit (S) ..... Quit subroutine program.
REPort (IS) ..... Report motor idle message to host device.
RESet [F]n (IS) ..... Reset flag Fn or output “n”.
RUN (IS) ..... Run continuously at base speed.
SENd [-]c “ccc” (IS) ..... Send string “ccc” by RS-232 to device “c”.
SHOw (IS) ..... Display list of assigned variables.
SET [F]n (IS) ..... Set flag Fn or output “n” active.
SLEw (IS) ..... Slew at maximum velocity with accel/decel.
STAtus (IS) ..... Send to host device the status display.
STOp (IS) ..... Stop current motion with accel/decel.
TIme (IS) ..... Reports times for last indexed distance.
Trace n (I) ..... Trace “n” lines of motion program.
Until [-]n ccc (S) ..... Until “n” is active go to label “ccc”.
UUNits (IS) ..... Enable user units for motion commands.
Verify p (IS) ..... Verify value of parameter “p” to host device.
Wait n (IS) ..... Wait “n” hundredths of a second.
<+>..... Set direction clockwise, math.
<->..... Set direction counter clockwise, math.
</> “Divide sign” (IS)..... Used in math to divide integers.
<\> “Backslash” (S) ..... Used as “not” operator.
<, <=, =, <>, => and > ..... Comparison Operators

AND and OR ..... Boolean Operators
<%> “Modulo Operator” (IS) Displays the remainder of integers that have been divided.
<*> “Multiply sign” (IS)..... Used in math to multiply integers.
$v “Dollar sign” (IS)..... Defines that variable “v” is a string.
<=> “Equal sign” (IS) ..... Math function.
<,> (IS) ..... Delimiter, same as a <space>.
(cccccccc) (S) ..... Label “ccc” up to 8 alpha numeric characters.
<^> Comment lines (S)..... The following lines are a comment.
<~> “Tilde” (I) ..... Global “ESCAPE” command for all axis on the RS-232 daisy chain.
<:> c (I) ..... Axis that host device will communicate to.
<CTRL-X> (I) ..... Erase current command line.
<Backspace> or <CTRL-H> (I) Remove character from command line.
<ESC> (I) ..... Exit editor or escape current motion program.
<Return> (I) ..... Return or ENTER key on host device.
<Space> (IS) ..... Delimiter between commands.

ICL SYSTEM MESSAGES

All messages will be preceded with an axis designator, in this example **Z**, so that customers with daisy chained systems will be able to identify the axis communicating. EXAMPLE: **Z "24 MOTOR IDLE"**

MESSAGE	MEANING
#0 American Precision Industries Smart Axis Controller, Copy .... Software Version x.x	Standard sign-on message preceded by unit designator
#1 COMMAND ERROR	Invalid instruction name.
#2 NUMBER FORMAT ERROR	Invalid parameter value.
#3 RANGE ERROR	Parameter value not within acceptable range.
#4 MUST BE IN PROGRAM	Instruction allowed in program mode only.
#5 NOT ALLOWED IN PROGRAM	Instruction allowed in immediate mode only.
#6 NO SUCH PARAMETER	Invalid parameter name.
#7 READ ONLY	Attempting to change a parameter that is read only.
#8 NOT ALLOWED WHILE MOVING	Instruction cannot be executed while motion is occurring
#9 REGISTER "H" OUT OF RANGE	Invalid entry for highest speed parameter "H".
#10 REGISTER "B" OUT OF RANGE	Invalid entry for base speed parameter "B".
#11 REGISTER "A" OUT OF RANGE	Invalid entry for acceleration time parameter "A"
#12 REGISTER "D" OUT OF RANGE	Invalid entry for deceleration time parameter "D".
#13 REGISTER "J" OUT OF RANGE	Invalid entry for jog speed parameter "J".
#14 REGISTER "M" OUT OF RANGE	Invalid entry for maximum speed parameter "M".
#15 DIRECTORY FULL	A maximum of 88 programs are allowed.
#16 MISSING PROGRAM NAME	Program name is required with edit command.
#17 LABEL TO LONG	A maximum of 8 characters is allowed for an address label.
#18 INSUFFICIENT MEMORY	You have exceeded the available memory.
#19 NO SUCH FILE FOUND	Attempting to access a file that does not exist.
#20 CALL STACK OVERFLOW	A maximum of 4 nested calls are allowed.
#21 UNDEFINED BRANCH TARGET	Attempting to branch to an undefined label.
#22 LOOP NESTED TOO DEEP	A maximum of 4 nested loops are allowed.
#23 HOME NOT FOUND	Unable to find home limit switch.
#24 MOTOR IDLE	Motion is complete.
#25 LIMIT SWITCH ENCOUNTERED	Attempting to move beyond a limit switch.
#26 NO RAMP DATA FILE SPECIFIED	Parameter "RD" Ramp data file number must be specified.
#27 SORRY - 32K RAM REQUIRED	32K memory option is required for this function.
#28 MISSING PARAMETER	Parameter name omitted in command
#29 STOP INPUT ACTIVE	Stop input has been activated.
#30 DRIVE FAULT	Drive fault is active.
#31 VARIABLE NUMBER OUT OF RANGE	Invalid entry for variable.
#32 WRONG VARIABLE TYPE	Invalid usage of a variable such as adding strings, etc.
#33 ***** CLEARED *****	Variable has not been assigned.
#34 ATTEMPT TO DIVIDE BY ZERO	Invalid math expression
#35 COMPUTATION OVERFLOW	Invalid math expression
#36 MOTOR HAS STALLED	Warning when "SP" not assigned

#37 DW CAN'T BE NEG, SETTING TO 0	Warning when a negative "DW" is entered
#38 DW SET TO MIN ONE MOTOR COUNT	Deadband window
#39 MISSING COMPARE OPERATOR	Invalid compare expression
#40 MISSING RIGHT PARENTHESIS	Invalid expression
#41 MISSING BOOLEAN OPERATOR	Invalid expression
#96 *** WARNING *** CLEAR WILL ERASE ALL VARIABLES PROCEED (Y/N) -•••-	Prevent an unintentional loss of variables
#97 *** WARNING *** MEMORY CHECKSUM HAS CHANGED! MEMORY IS NO LONGER VALID! DO YOU WISH TO CLEAR AND INITIALIZE MEMORY, WHICH WILL ERASE ALL PROGRAMS AND PARAMETERS? (Y/N)?_	Check sum error warning ICL System error message
#98 STACK ERROR	ICL System error message
#99 *** WARNING *** BOOT WILL ERASE ALL PROGRAMS AND PARAMETERS PROCEED? (Y/N)?	Prevent an unintentional boot and loss of data.

Programming Hints

NOTE: If a series of motion commands are issued in the immediate command mode while the ICL System is in the continuous processing mode, an error message will be displayed. The PAUSE command should be utilized to prevent an error since only one motion command can be executed at a time.

Notes