Intelligent Actuator, Inc. Product Training

SEL Notes and Topics

Table of Contents – Part One

Product Line	3
System Interface	3
I/O, Wiring, BCD	4
Specifications / Architecture	6
Encoder Operation	8
Programming Semantics	10
Practice Programming I	11
Command Structure	13
Numbers and Variables	13
Programming Logic	14
Teaching Pendant	16

Table of Contents - Part Two

Maintenance, Service, & Repair	17
Cables	18
Controllers	18
More Variables	19
Pointers	20
More SEL Commands	20
Programming Examples	23
Special Commands	24
Cycle Time	27
Program Flow	29
Troubleshooting	29
Parameters	31
Palletizing	34

Product Line Overview (from bottom up)

- RC: "Point Mode" only. No programming. Control serially or via I/O. Priced from ~$900.

- DS: Programmable, typically single-axis. Supplanted SA series. Priced from ~$1250. Two-axis systems lack interpolated motion. May also be used with the X-SEL or SEL-GDS multi-axis controllers.

- Super SEL A&B: 20W-200W. The predecessor to the Super SEL E&G, the Super SEL A&B are available in AC or DC versions. Super SEL B can multitask up to 16 programs with expandable I/O (up to 288). No option for 2ch Serial Communication.

- Super SEL E&G: 20W-400W, controls actuators up to 2500mm (ballscrew), 3000mm (belt) in length. AC or DC versions.

- X-SEL: 20W-750W, latest & greatest SEL controller. Features improved memory, processing, drivers, troubleshooting capability, as well as various network communication options. AC motor version only.

- SCARA: Newer unit, cheaper than competitors, with 80% of competitor functionality. Units run about $20,000 complete (list). 24 I/O standard, expandable up to 96. 0.76 sec. cycle time. X-SEL based SCARA in development.

System Interface

- No tuning needed. Controller, actuator(s), brackets, cables & software included.

- DS has 15 user inputs, 6 user outputs. 8 dedicated inputs, 2 dedicated outputs.

- Super SEL-E (1 axis) has 24 I/O standard, expandable up to 96.

- Super SEL-G (2-8 axes) has 24 I/O standard, expandable up to 288.

- X-SEL (1-4 axes) has 32 inputs and 16 outputs, expandable to 4 I/O slots.

- Systems are compact, flexible, easy to use, and powerful.

- PC Software or Teaching Pendant may be used to interface.

- Teach pendant great for teaching points and portability.
- Some commands unavailable on teach pendant.
- Serial protocol used in both teaching pendant and software.
- RS232 option allows for greater networking flexibility, at nominal additional cost for the option and cable ($200).

I/O & Wiring

- 24VDC optically isolated I/O (except DS).
- Surge protection is present on the DS, but not on Super SEL.
- 20mA continuous output, 100mA / point maximum rated, depending on duty cycle.
- If more power is needed, external relays or PLCs must be used.
- NPN is the standard I/O setup; PNP is available upon request.
    - NPN – "Sinking". 0V turns on input, "on" output is 0V. Goes "low".
    - PNP – "Sourcing". 24V turns on input, "on" output is 24VDC "high".
- Must supply 24VDC to I/O on SEL E/G, IH-SEL, and expansion SEL A/B.
- No ground loops, and dedicated controller power supply.
- Installation and noise concerns. (Consult p. 7-11, etc.)
    - Surge killers for AC inductive loads.
    - Clamping diode for DC inductive loads.
    - Precautions taken for noisy environments will solve common noise related problems.
    - Noise introduced to the controller via power supply and I/O.

- Wiring diagrams for I/O. Input / output introduction. (Consult p. 29)
- DS allows for 1000 programming steps, 32 programs, 500 points, and 8 multitasks.
- Super SEL allows for 3000 programming steps, 64 programs, 2000 points, and 16 multitasks.
- X-SEL allows for 6000 programming steps, 64 programs, 3000 points, and 16 multitasks.

BCD Inputs

- Used for external start of programs. Thumbwheel usually used, IA I/O can also be used to dial in program numbers.
- Dedicated inputs for BCD program entry.
- Can only re-use these inputs (7-13) on X-SEL after program has begun.
- Example: we want to "launch program #15".

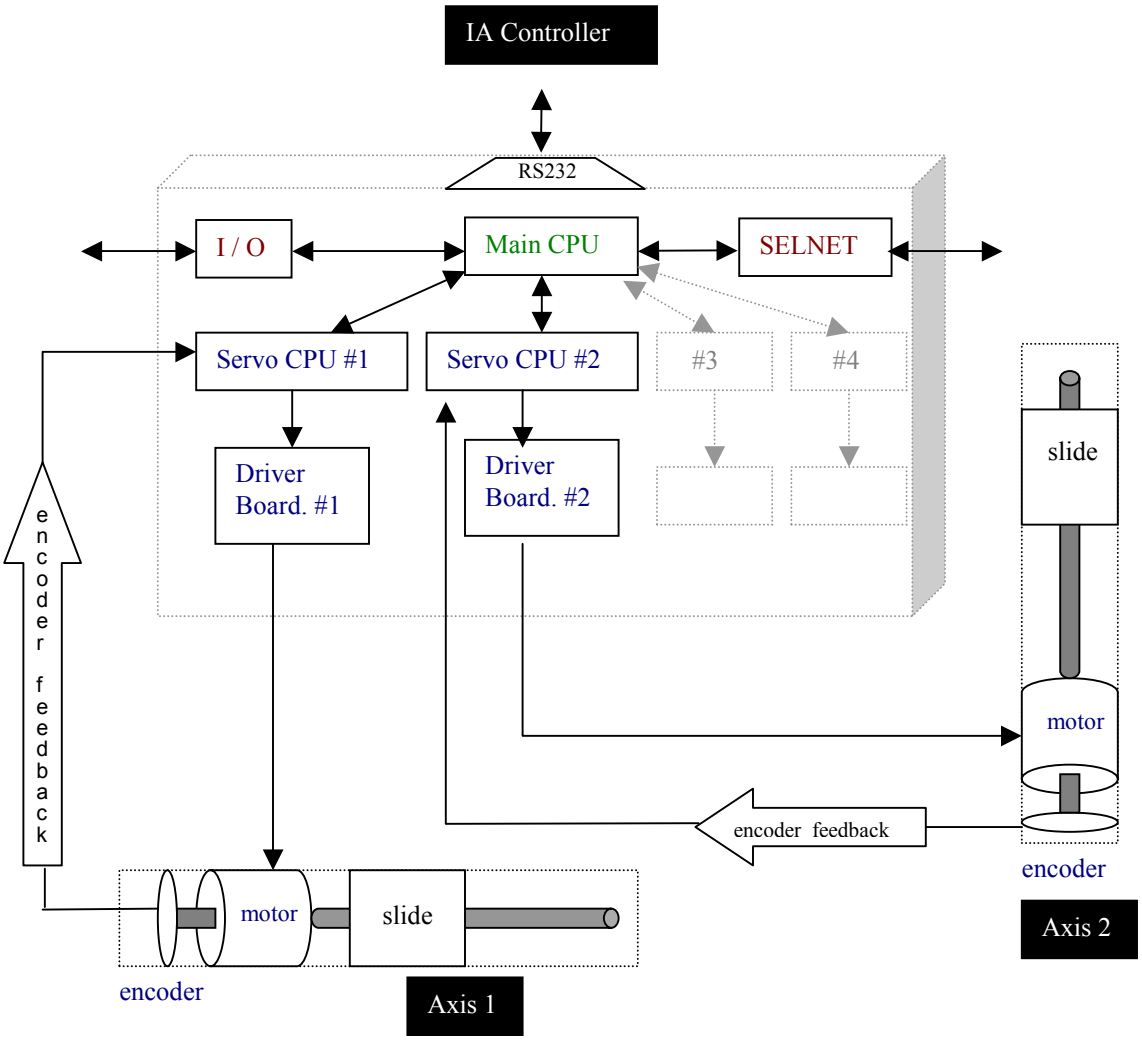
			1				5
	0	***0***	***1***	***0***	***1***	***0***	***1***
PRG#	40	20	10	8	4	2	1
BCD INP#	13	12	11	10	9	8	7

  - Dial in, pulse external start (000), program starts.
  - Dedicated e-stop terminal, and must be a normally closed contact. (See X-SEL "7. Other Parameters" for e-stop recovery options)

Specifications / Architecture (X-SEL)

- X-SEL uses a Hitachi SH-4 32-bit RISC (reduced instruction set chip) processor.
- 150 Million Instructions/Second (MIPS), Servo uses 240000 samples/sec, or every 4μsec.
- All work on encoder feedback for positioning. "Checks and balances".
- Servo CPU for servo control and Main CPU for calculating, running programs, memory, I/O.
- Allows for 16 programs to be run at one time, in parallel (multitasking).

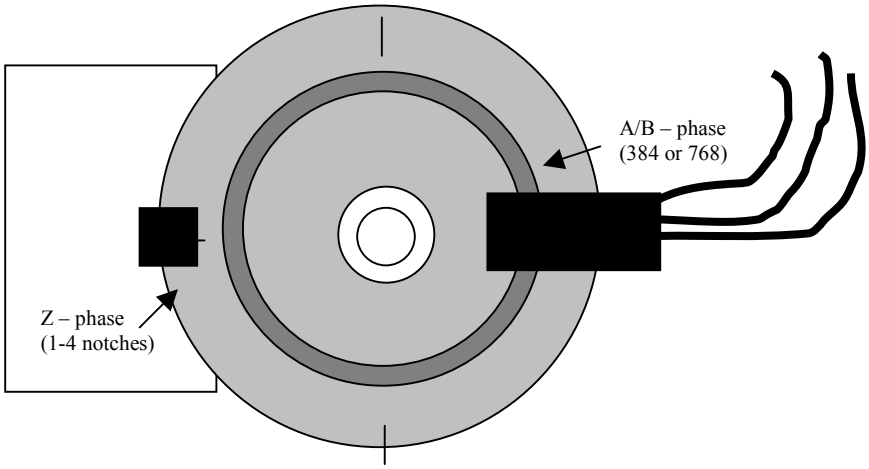
- 1 msec time slice maximum for each program running in parallel.
- Program control does not wait for certain commands to finish executing before jumping to the next program's command.
- Faster time slice for the following: WTON/WTOF, MOVP, MOVL, PATH, ARC, CIR, TIMW.



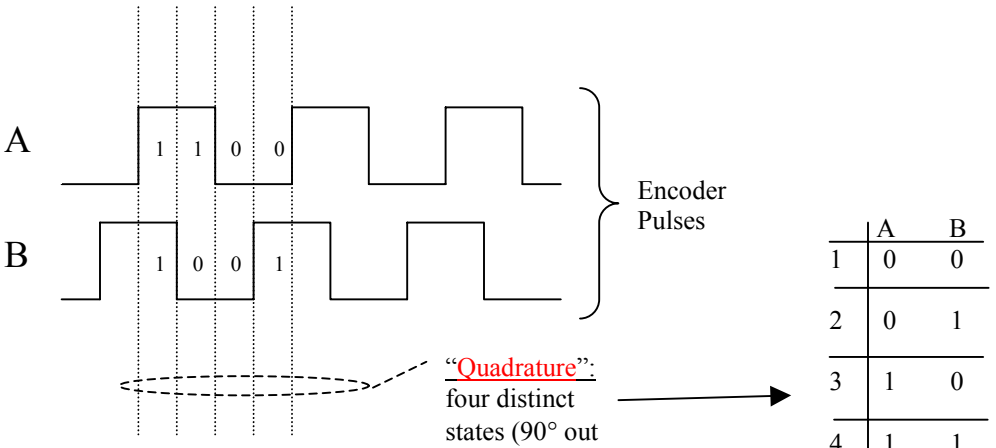
IA System Architecture Diagram

Encoder Operation (SEL-G)

- X-SEL - 17 bit incremental encoder, 14 bit control resolution, 16384 pulses/rev
- "3-phase, Optical Incremental Encoder" allows for relative positioning (loses position when power is cut, cable is damaged; good because home value is based off of encoder).
- Z-phase is based on aligning motor coils with Z-pulses.
- A/B-phase sensors are used for position feedback. Controller must know where the actuator is while moving in order to control drivers.
- After over current limit is sensed, the motor rotates until the z-pulse is sensed to "zero" position (0.000 mm).
- Motion then occurs using the A/B pulses, feedback, and servo drivers.
- Quadrature (see diagram) allows for an effective resolution 4 times that of the encoder count (384 or 768 pulses per revolution).

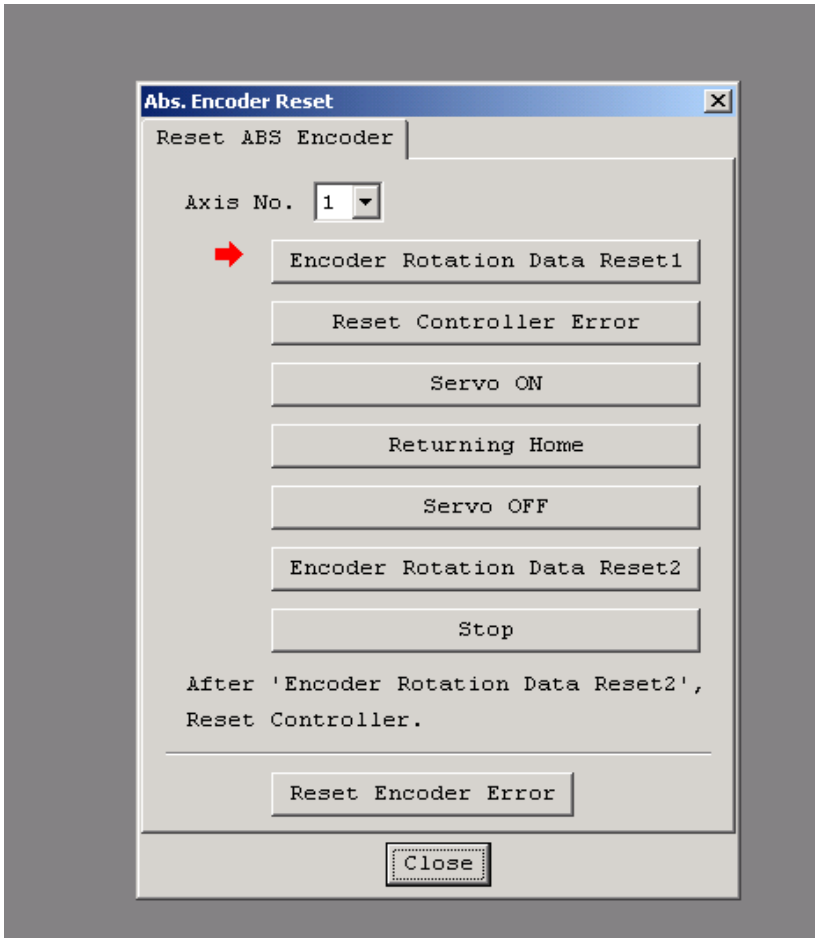


Example encoder assembly: IS unit w/ 16 pitch ballscrew



	A	B
1	0	0
2	0	1
3	1	0
4	1	1

Absolute Encoder Homing



The absolute encoder homing menu

- This menu, "Abs. Encoder reset", is located under the "Controller" menu. This menu will take you through the homing procedure for each axis. Press each button once and the red arrow will indicate the next step. Please note the "Reset controller" text. Once the homing procedure is complete, reset the controller with the "Software Reset" under the "Controller" menu.

- The controller now knows the location of the actuators until the actuators are unplugged from the controller.

- The X-SEL demo is set up for use with absolute encoders, which are located on the actuator. The battery is conveniently located on the front of the controller (one battery for each axis). This means that whenever the actuator is disconnected from the controller, the absolute encoders lose power. This will require you to home the actuators.

Programming semantics

- SEL is a mnemonic language in which all commands hint at the actual process.
- "HOME" homes the actuator(s), "MOVP" performs a point to point move, etc.
- Over 100 commands for programming flexibility.
- Can perform numerous tasks using only a handful of commands:

  HOME – used to send the actuator to its home position.

  VEL – sets the velocity within the program for point-to-point moves, etc.

  MOVP – moves each axis involved at a designated velocity to the chosen point. The axes are not related in movement (may end at different times).

  MOVL – moves the actuator to the specified position number by adjusting acceleration, velocity, and axes to arrive in the shortest path (linear interpolation). Both axes complete the move at the same time.

  BTON – turns output / flag on.

  BTOF – turns output / flag off.

  WTON – waits for input/flag to turn on.

  WTOF – waits for input/flag to turn off.

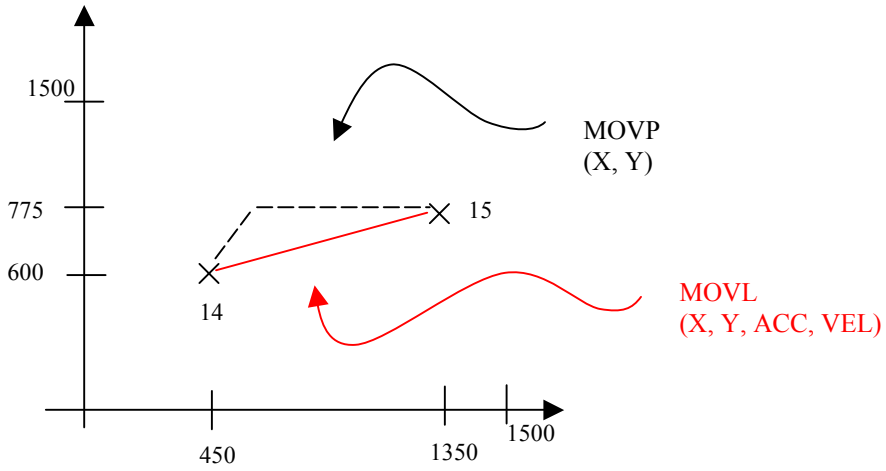
  TIMW – sets a timer that pauses the program for a specified time.

  TAG – sets a placeholder for a GOTO loop.

  GOTO – returns the program control to the tag number specified, thus creating a simple loop.

- "Axis Pattern" is used for certain commands (i.e. HOME 10011100). Works from right to left in increasing axis number (p. 114, 115).
- (Example program on board)

- Program is based on the following diagram:



Example (plotted version)

Point #	Acc	Vel	Axis 1	Axis 2	…Axis n
14			25.686	125.1	
15			120	76.55	

Example Point Table Data

Software Lab I

Introduction

- Check out simple 6-line program (uses very few commands).
- Open software (RS-232 cable must be connected to be "on-line")
- FILE menu works with files on the hard drive or floppy disk.
- Some menus work with the controller itself (note: not available off-line). When unavailable the menus will be grayed out.
- Demonstrate the PROGRAM control (play, pause, step-by-step and break).

Point Table

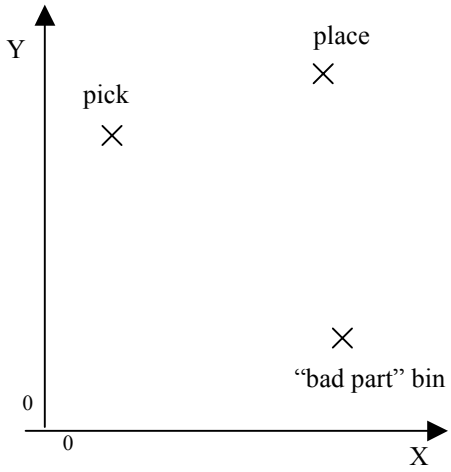
- Enter points in the point table (POINT, Edit, then specify range)
- Can also set ACCeleration and VELocity values in the point table. These values will override those settings in the program for moves made to that point.
- These points can either be typed in directly, or taught by jogging/manual move.
  (Note: must turn servo on, home once, then turn servo off to manually teach points. Controller needs to know where the actuator is!)
- There is also a "Safety Velocity Specified" menu on the "on-line menu".

Programming

- Enter sample program, write to controller, then run.
- Actuator should home, then move back and forth between the two points entered.
- Point data can be changed "on the fly" while the program is running. This is helpful for fine-tuning points or alterations to your system while running code.
- Simply open up point table (if you have closed it). Highlight the point you wish to change, enter the new value, and write to controller.
  (Note: point(s) must be within the range of the actuator or an error will occur.)

Command Structure

- Can get more complex from our 6-line example.
- Pick and place example (three points).




```
                HOME    1       1
                VEL     100
                TAG     1
                MOVL    1
                BTON    310
          15    MOVL    2
        N 15    MOVL    3
                BTOF    310
                GOTO    1
```

Conditions:
15 "on" = good part
15 "off" = bad part

Numerals and Symbols

- Most information needed is on p. 106 of SEL E/G manual.
- All on/off items within the SEL controller have unique numbers.
- INPUTS: 000-299
- OUTPUTS: 300-599
- FLAGS: 600-999
- Flags are "internal I/O"; turned on or off.
- Some flags are program-only; some can be shared between programs.
- VARIABLES: 1-399 &1001-1399 (½ integer, ½ real)
- Real variables and all calculations use floating-point methodology (p.107).

- COLUMNS: used for storing ASCII strings for SELNET communications.
- "LOCAL" flag/variable/column: only used within a given program and reset to 0 or blank at end of program, E-stop, or power-off.
- "GLOBAL" flag/variable/column: can be read by any other program running in parallel (multitasking) and these global items maintain their value upon end of program, E-stop, and power off (as long as the backup battery is still good!).
- " VIRTUAL INPUTS & OUTPUTS" 7000-7299 & 7300-7599
- "SYMBOLS" allow all addresses in the controller to be named for easier programming and troubleshooting. The 1000 possible defined symbols can be used 5000 times. Can also name/define constants.

Programming Logic

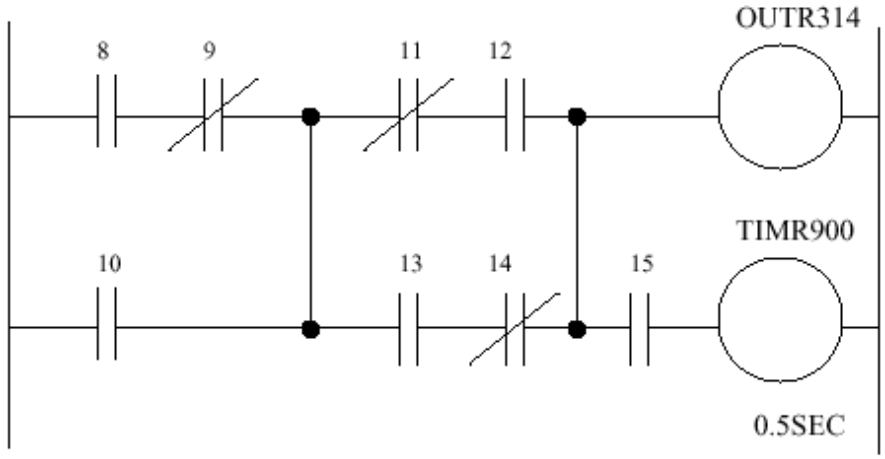
- The first 3 (or 2) columns in the command structure can be utilized to create conditional lines of code.
- "Conditional Statement": If the condition specified is true, that line of code will be executed. If not, the program will skip to the next line – it does not wait for the condition to become true.
- This functionality makes the SEL language a truly powerful language.
- The condition may be an input, output, or flag.
- The "N" column is used to specify "NOT" or an off (logic 0) condition.
- The "E" or "A/O" column is used for expanding the condition set with logical AND/OR statements with the ability for AND/OR blocks.
- The result can be "ladder-like" as shown. Similar to ladder logic.
- The command column for all but the last expansion condition must be blank.

Step	A/O	Cond	Comd	Operand 1	Operand 2	Post	Comments
1		N 900					If 900 isn't on
2	A	N 901	GOTO	1			And 901 isn't on, goto 1
3		902					If 902 is on,
4	O	N 903	GOTO	2			Or 903 is not, goto 2.
5			EXIT				(Above conditions weren't met
6							So, exit the program)

- By combining the different conditional statements, code can be compressed greatly.

Conditions Example

5. Input Examples



Expansion Condition	N	Input Condition	Command	Operand 1	Operand 2	Post
E	N	Cnd	Cmnd	Operand 1	Operand 2	Pst
LD		7001	CHPR	1		
			TPCD	1		
			TAG	1		
LD		8				
A	N	9				
O		10				
LD	N	11				
A		12				
LD		13				
A	N	14				
OB						
AB			OUTR	314		
A		15	TIMR	0.5		
LD		7001	TSLP	3		
LD		7001	GOTO	1		
LD		7001	EXIT			

Teaching Pendant

- Name comes from the ability to teach points while standing above the work area.
- Rarely used, but affordable and portable.
- Works on same RS-232 port as software does, using same protocols.

Maintenance

- SA: Ballscrew goes through motor assembly. Can replace DS encoder assembly, but SA is a "throwaway" actuator. Greasing is necessary for DS & SA.

- IS / ISP: Ballscrew goes through motor assembly with IS Series as well. Encoder located on end. Grease gun used on fittings, lithium grease used on ballscrew.

- AS: Motor/Encoder assembly is all enclosed. Coupling mechanism used between motor and ballscrew. Homing is done differently.

- AS homing involves the coupling as well. Must tighten down motor, loosen coupling to obtain difference between home and 1st position, then re-tighten the coupling.

- B Grease used in maintenance, C class used in clean room units from the factory. THK AFB grease crosses over specs for both ballscrew and linear guide grease, therefore, this grease can be used on both, in non-clean room situations.

- Substitutions can be made for THK AFB grease. Please ask us if you can substitute for a specific brand (i.e. Shell, Mobil, etc.).

- Spray lubricant is acceptable when the ballscrew cover cannot be removed. Careful! Too much lubricant will run down to the bottom and pool on the brake assembly causing damage to the brake.

- Ballscrew grease is applied by hand to the ballscrew. Even distribution of the ballscrew grease is accomplished by moving the slide back and forth. This should be applied until the ballscrew is greasy on all sides.

- Grease fittings are inserted to lubricate the guide rails. This grease is applied until grease appears out of the sides.

Service and repairs

- Both the Chicago and LA offices have replacement parts and perform service. Call for part availability, pricing and troubleshooting questions.

- Common electrical parts: Main CPU, Motor CPU, Drivers, I/O, boards, and backup lithium batteries.
- Common actuator parts: Motor & Encoder, Brushes, Pulleys, Ballscrews, Belts, Linear Guides, and Couplings.
- Besides greasing, re-alignment of the actuator may need to be done if the slide is "binding".
- The more information provided upon sending in a repair, the quicker the turnaround on a replacement or repaired unit.

Cables

- Many different options for cables and cable assemblies.
- The motor/encoder cables are now robot cables.
- The correct cable will vary from system to system based on different factors.
- Thorough specifications of cabling needs must be done when ordering additional cables or replacements.
- Other miscellaneous cables available are the RS-232 cable replacements, RS-232 with E-stop, brake and limit switch cables, SCARA accessory cabling, and so forth.

Controllers

- Controllers are configurable to meet the specific needs of the system.
- The main adaptable controller features are:

  -Amount of I/O

  -Wattage of drivers

  -Number of axes able to be controlled

  -Limit Switch option

-RS232 option

-High speed I/O

- The combinations are numerous, but check for your system's feasibility.
- Also, changes to your system may induce noise that was not present before.

More variables

- As before, "local" variables are used only within the program.
- And "global" variables can be shared between programs.
- The same goes for flags and columns.
- An SEL "REAL" variable has the range of –99,999.99 to 999,999.99; with the decimal point able to "float" within the 8-digit number (floating point math).
- An SEL "INTEGER" variable has the range of –99,999,999 to 99,999,999 and cannot accommodate decimal points.
- Real used often in point table manipulation.
- Using these variables with math and trig functions allow for a large amount of calculation possibilities.

Pointers

- Pointers are used for "indirect addressing" between variables.
- Pointers and counters used in combination with simple loops simplifies code greatly:

```
LET     1   10                  /*  let variable 1 = 10
```

```
TAG     1                 /*  set placeholder 1
ADD     1   5                   /*  add 5 to variable 1
MOVL  *1                  /*  move to point determined by
                          /*  the value present in variable 1.
GOTO    1                 /*  jump back to tag 1 and repeat
```

- The program above will increment the point that the actuator(s) move to by 5 each time through the loop, and move to that point.
- The use of the pointer *1 cuts down on programming lines, which saves space, time, and confusion.
- More advanced pointers will be used later on in training.

More SEL Commands

- Walkthrough of the additional commands available.
- Standard commands are grouped by purpose on p.119-120.
- Expansion commands are grouped by purpose on p.162.
- Some popular additional commands are as follows:

OFST – adds an offset value to the original value when before the actuator moves (p. 122).

HOLD – decelerate and stops motion when a designated input turns on, continues motion when input turns back off again (p. 124). Not OSHA compliant.

CIR – actuator executes a circular motion starting at the current position and passing points 1 and 2 using interpolation (p. 128).

PATH – continuous move between the starting point and ending point using B-spline interpolation through a number of points. Actuator will try to "reach" points in-between, but not necessarily intersect them (p.129).

ARC – uses similar calculations methods as CIR and PATH to create an arc through two points.

JFWN – jogs the actuator forward upon the activation of a chosen input (p. 130).

JFWF – jogs the actuator forward upon the deactivation of a chosen input (p.131).

IN – read in binary numbers through I/O or flag (p. 136).

OUT – output binary numbers through I/O or flag (p. 137).

EXPG / ABPG – start other program / abandon other program (p. 140).

BGSR / EDSR – begin subroutine / end subroutine (p. 141).

PGET – read determined position data from the point table location into local variable (p. 157).

PRED – read current coordinates of the axis specified into point table location (p. 159).
-Additional commands for math functions, subroutines, point table, IF-THEN-ELSE loops, DO-WHILE loops, bit masking and SELNET.

More X-SEL Commands

OUTR – PLC output turns off when condition turn off.

BTPN – Similar as BTON but is pulsed on.(op1 = flag/output, op2 = time)

TIMR – PLC timer relay turn on local flag with delay. (op1 = flag, op2 = time)

SSPG – Effectively holds another program. OP1 = program to hold

RSPG – Releases the hold on another program

CHVL – Specifies a velocity for specific axis. OP1=Axis Pattern, OP2=Velocity

PSPL – Similar to path but will actually hit the points with better velocity control

PUSH – The Push movement can only work on a single axis at a time. There is no spec data on the push force yet. Notice that the push can be pushed back without error. OP1=Point

PAPR – To be used before the PUSH command. OP1=Distance from end point to start pushing. OP2=Velocity

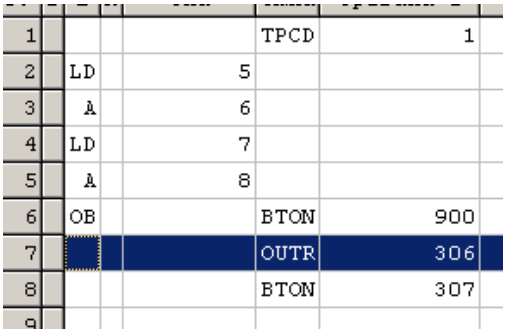
CIR2 – Same and CIR on the Sel-G

ARC2 – Same as ARC on the Sel-G

ARCD – Arc based on angle. Operand 1 is the end point, operand 2 the angle(degrees)

ARCC – Arc based on radius. Operand 1 is the radius, operand 2 the angle(degrees)

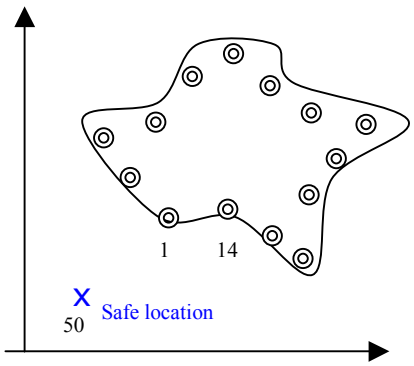
The X-SEL has additional commands that were not available on earlier versions of the Super SEL controllers.



1				TPCD	1
2		LD	5		
3		A	6		
4		LD	7		
5		A	8		
6		OB		BTON	900
7				OUTR	306
8				BTON	307

Pointer Example

- Have a part on a pallet that needs 14 different holes drilled.
- A "safe" position is designated for the actuator to wait while the part is being swapped out.
- Input "16" is wired to an external cycle start button.
- Input "15" is wired to the drill activation switch.
- Output "311" is wired to an external lamp that indicates when the cycle is done.
- Since we have 14 points to move to, and external devices attached to I/O, we need to use a combination of comparison statements, loops, pointers and output for this system to properly work.



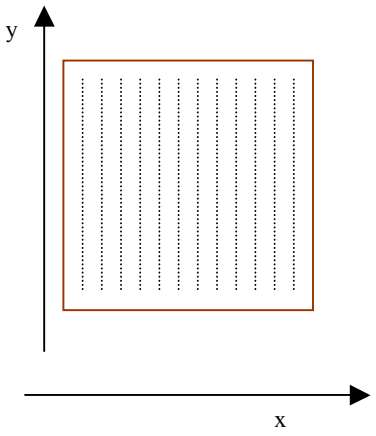

```
                HOME  11
                VEL   100
                LET   20    1

                TAG   1
                MOVP  *20
                BTON  310
                WTON  15
                BTOF  310
                ADD   20    1
                CPEQ  20    15    900
N 900           GOTO  1

                BTON  311
                MOVP  50
                WTON  16
                BTOF  311
                LET   20    1
                GOTO  1
```

- Could also use a global variable in so we know where we left off in case of a broken drill bit, power loss, etc.
- This would save time and money in having to re-do a part or restart the entire cycle.

- This global variable could be used within our present loop and only be reset when the part is complete. Would increment and "keep track" as we drill holes.

OFST (offset) Example:

- We have 144 points on a pallet, but don't want to teach all the points.
- Best plan of attack: teach one row (or column), and offset the rest.



-2.5 mm offset between columns.

-Teach first row (far left)

-Offset the rest before performing task on points.

IN/OUT Example:

- We have five stations in a row working off a conveyor belt that is indexing parts. The first station is the inspection station and the last is the sorting station (discards bad parts).
- Best plan of attack: use flags to set up a shift register.

Multi-tasking Example:

- We have an inkjet head (single axis) to dispense within a certain length of a sheet that is indexed.
- Best Plan of attack: Multi-task two programs, a motion program and a motion-monitoring program.

Special Commands

- **MOVD / MVDI**: Move position / Incremental move (DS only).
- **MVPI / MVLI**: Incremental PTP move / Incremental interpolated move.
- **VELS**: Sets max X,Y velocity as a % of rated during PTP move. (SCARA)
- **ACCS**: Sets max. X, Y acceleration as a % of rated during PTP move. (SCARA)
- **JUMP**: Executes arch motion move between two points (SCARA)
- **LEFT**: Moves SCARA into left arm work envelope. (SCARA)
- **RIGH**: Moves SCARA into right arm work envelope. (SCARA)
- **PTPD**: Forbid the SCARA to make reversed arm move. (SCARA)
- **PTPE**: Permit the SCARA to make a reversed arm move. *(SCARA)*

Symbols

- A new feature is the ability to name just about everything in the program (points, programs, inputs, outputs, virtual inputs, virtual outputs, flags, tags, subroutines …etc-9 characters max). This can make troubleshooting or altering a program much easier. To place a name, open the "Edit Symbol" under the "Symbol" menu. Once you have labeled your points or variables you can set up a program that may look similar to this:

			TAG	1		
			MOVP	Pick		
			TIMW	0.25		
			OFST	1	*xoffset	
			OFST	10	*yoffset	
			MOVP	Place		
			TIMW	0.25		
			ADD	Pallets	1	
			ADD	xoffset	20	
			CPEQ	xoffset	140	900
	N	900	GOTO	1		
			LET	xoffset	0	
			ADD	yoffset	25	
			CPEQ	yoffset	75	901
	N	901	GOTO	1		
			LET	xoffset	0	
			LET	yoffset	0	
		10	LET	Pallets	0	
			GOTO	1		

- Remember that to make a name or change, it must be done in the Symbols-edit menu. On the current versions of the software the names can only be seen in Symbol-edit screen. The software will not automatically

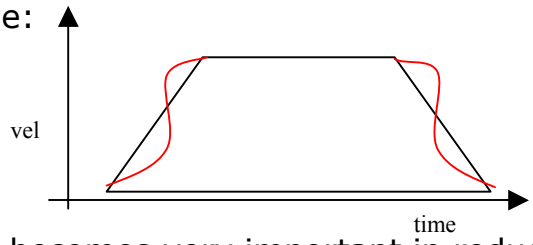
show you the names in any other menus. Also, note that the names are case sensitive, allowing you to use the same name with different casing.

Cycle Time

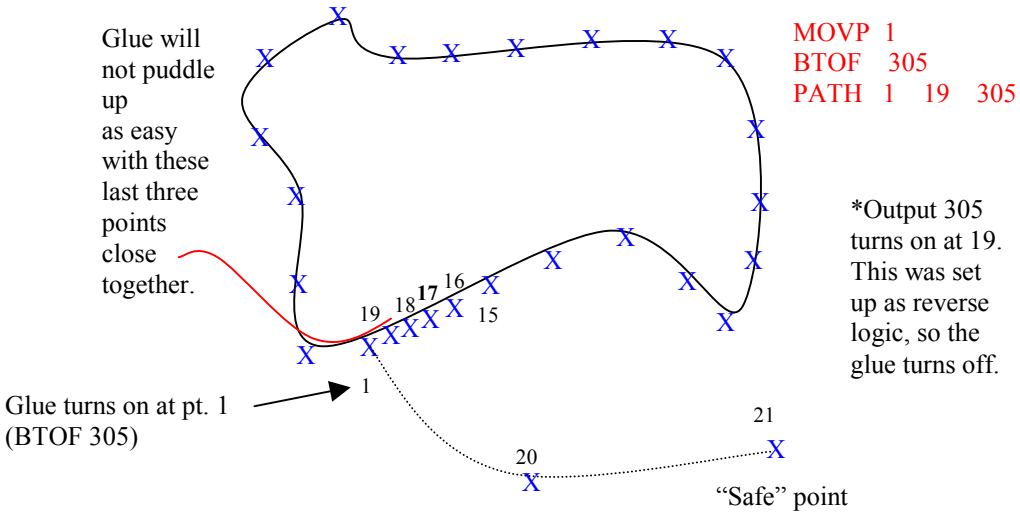
- Back of the SA catalog has good equations for determining cycle time.
- Excel spreadsheet with timing is available.
- Settling time is 0.15 sec for ballscrews, 0.20 for belts.
- This settling time is required in order to maintain the specified 0.2mm repeatability.
- If this tight repeatability is not needed, the "position end band" parameter can be increased to allow slightly sloppier positioning. The use of commands PBND, QRTN can also shorten cycle time.
- This method has been known to decrease settling time to as low as 0.05 sec.
- In cases where there are multiple movements, the decrease in individual settling time adds up dramatically.
- Multitasking programs also save time by executing non-motion dependent functions in parallel such as turning on/off outputs, grippers, etc.
- GTTM is a command used for determining cycle time by taking an initial "snapshot" of the system timer at both the beginning and end of the cycle.
- By subtracting the beginning sample from the end sample, the final cycle time is determined.
- Can use WTON, WTOF, and WTNT with a time added to only allow the program to wait for an input or flag for a set amount of time.

ACC and VEL vs. cycle time

- For short moves, acceleration is the most important characteristic.
- For long moves, velocity is the most important.
- S-curve, as seen below, "rounds" the edges for smoother motion. (SCRV).
- High acceleration does not necessarily mean faster cycle! The overshoot involved with a large acceleration value will cause increased settling time and longer overall cycle times
- Typical motion profile:



- The PATH command becomes very important in reducing cycle time because there is no complete stop in the move through the different points, only deceleration.
- Interpolated moves look 3 points ahead (PATH, CIR, ARC) which is beneficial for dispensing applications. "Back to Back" PATH or CIR or ARC, allows for the output to turn on at the end of the path. In example below, the glue turns on at point 1 and off at point 19. The outputs are reverse logic in that when output 305 turns on, the glue turns off, and when 305 turns off, the glue turns on.



Program Flow

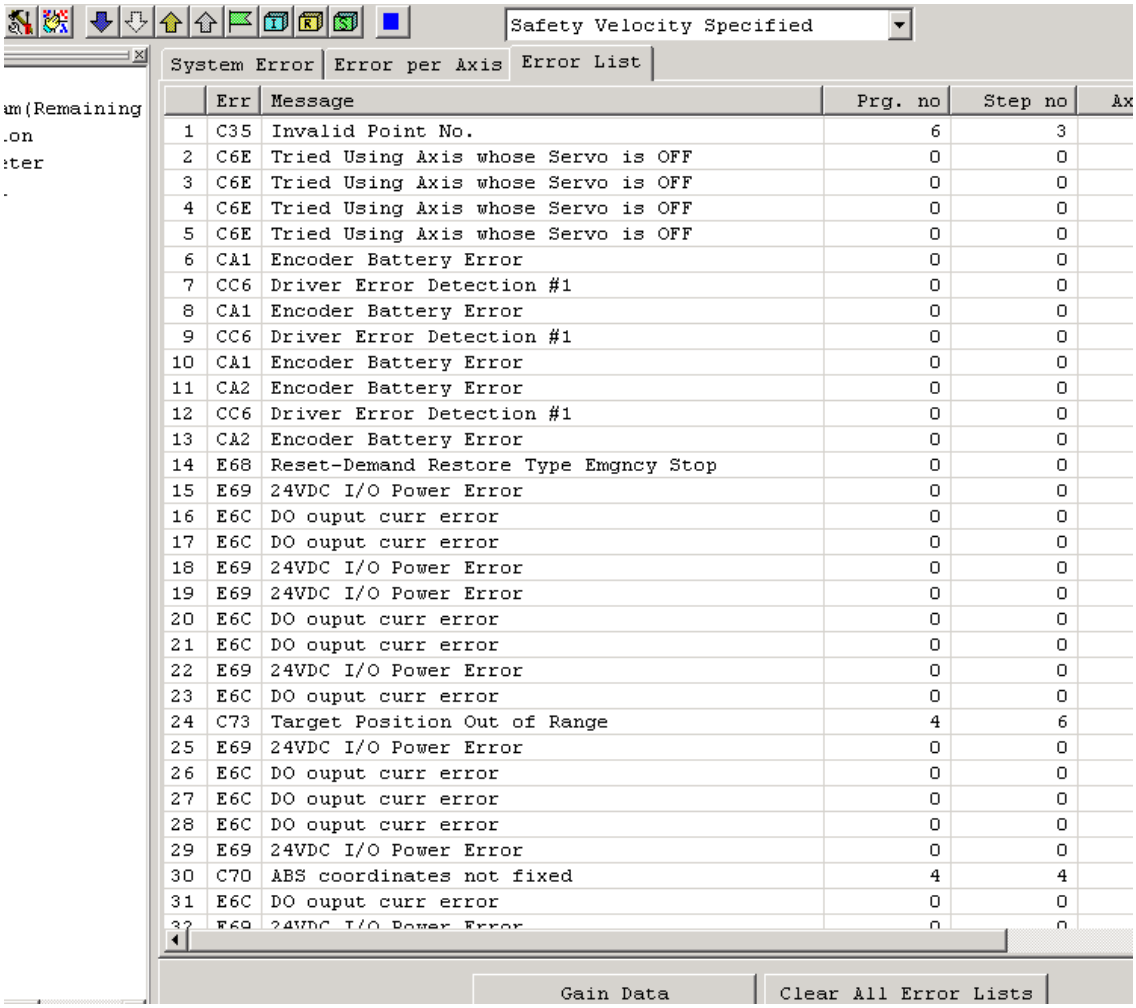
- Subroutines are "functions" within a program.
- Multitask programs are completely separate programs that can be made to run in parallel.
- Each separate program running in parallel can have its own subroutines that are executed when certain events occur or certain conditions are met.
- To control program flow, flags are used in conjunction with conditional statements, logical comparisons, and so forth.
- The most simple method used to control program execution is using a logical comparison with a simple loop, as seen before.

Troubleshooting

- **A**: communication/programming error codes
- **B**: CPU instruction processing error
- **C**: programming error
- **D**: axis errors (driver or encoder)
- **E**: CPU/IO error.
- Other common problems:
- Positioning off due to changes in environment affecting hard stop.
- Positioning off a fixed amount due to bad encoder or connection.
- Screen blank or memory problems from noise interference.
- If software related errors occur, check the programming or I/O.
- If hardware errors occur, check the lubrication, workspace or payload to see if anything has been changed or needs maintenance.
- There is an extensive error list that is very useful

- There is a list of errors that have occurred on the controller as shown below:

Error History Menu



	Err	Message	Prg. no	Step no	Ax
1	C35	Invalid Point No.	6	3	
2	C6E	Tried Using Axis whose Servo is OFF	0	0	
3	C6E	Tried Using Axis whose Servo is OFF	0	0	
4	C6E	Tried Using Axis whose Servo is OFF	0	0	
5	C6E	Tried Using Axis whose Servo is OFF	0	0	
6	CA1	Encoder Battery Error	0	0	
7	CC6	Driver Error Detection #1	0	0	
8	CA1	Encoder Battery Error	0	0	
9	CC6	Driver Error Detection #1	0	0	
10	CA1	Encoder Battery Error	0	0	
11	CA2	Encoder Battery Error	0	0	
12	CC6	Driver Error Detection #1	0	0	
13	CA2	Encoder Battery Error	0	0	
14	E68	Reset-Demand Restore Type Emgncy Stop	0	0	
15	E69	24VDC I/O Power Error	0	0	
16	E6C	DO ouput curr error	0	0	
17	E6C	DO ouput curr error	0	0	
18	E69	24VDC I/O Power Error	0	0	
19	E69	24VDC I/O Power Error	0	0	
20	E6C	DO ouput curr error	0	0	
21	E6C	DO ouput curr error	0	0	
22	E69	24VDC I/O Power Error	0	0	
23	E6C	DO ouput curr error	0	0	
24	C73	Target Position Out of Range	4	6	
25	E69	24VDC I/O Power Error	0	0	
26	E6C	DO ouput curr error	0	0	
27	E6C	DO ouput curr error	0	0	
28	E6C	DO ouput curr error	0	0	
29	E69	24VDC I/O Power Error	0	0	
30	C70	ABS coordinates not fixed	4	4	
31	E6C	DO ouput curr error	0	0	

Parameters

- Step through parameter list and explanations. Also listed in the manual.
- Changing many parameters on list will void warranty (results in pushing system beyond rated limits).
- Changing parameters correctly can benefit system, as mentioned with position end band vs. cycle time.
- Gain settings are often the culprit where noisy actuators are concerned.
- The parameters are also used to alter the characteristics of I/O, fault recovery and other program control features.

Fault Recovery Examples

Emergency stop options (used with parameter "OTHER #10" set to 0,1,2)

- The default e-stop situation.

  If the e-stop is triggered all programs will stop immediately and turn outputs off. Once the e-stop is released the controller will return to the "ARDY" state (automatic mode ready state).

- E-stop requiring reset.

  If the e-stop is triggered all programs will stop immediately and turn outputs off. Once the e-stop is released the controller will go to an alarm state indicating the controller must be reset. Using the soft rest input is useful in this case (Input #1 see parameter "I/0 #31). Using this input will require it to be on for one second. Please note, if this parameter is used the controller will always reset when the input #1 is on for more than one second.

- E-stop with program recovery.

  If the e-stop is triggered all MOTION programs will pause and outputs remain on. Once the e-stop is released the controller will display "- rS" indicating input #5 will continue program where it last left off. This will require the parameter "I/0 #35 to be set to 1.

There are some notes to keep in mind. If the system is moved during e-stop, the system will be moved to where it left off (one axis at a time) before continuing with the program. Also, the e-stop will only stop motion programs, not non-motion programs. The last note is that the controller can run a non-motion program during e-stop. This program will continue to run after e-stop recovery.

Enable options (used with parameter "OTHER #11" set to 0,1,2)

- The default enable situation.

  If the disable is triggered all programs will stop immediately and turn outputs off. Once the e-stop is released the controller will return to the "ARDY" state (automatic mode ready state).

- Enable requiring reset.

  If the disable is triggered all programs will stop immediately and turn outputs off. Once the disable is released the controller will go to an alarm state indicating the controller must be reset. Using the soft rest input is useful in this case (Input #1 see parameter "I/0 #31"). Using this input will require it to be on for one second. Please note, if this parameter is used the controller will always reset when the input #1 is on for more than one second.

- Enable with program recovery.

  If the disable is triggered all MOTION programs will pause and outputs remain on. Once the e-stop is released the controller will display "- rS" indicating input #5 will continue program where it last left off. This will require the parameter "I/0 #35 to be set to 1.

  There are some notes to keep in mind. If the system is moved during disable, the system will be moved to where it left off (one axis at a time) before continuing with the program. Also, the disable will only stop motion programs, not non-motion programs. The last note is that the controller can run a non-motion program during disable. This program will continue to run after enable recovery.

Fault Recovery Related Parameters

- Outputs not affected by e-stop or enable. Using the parameters "I/O #70 & 71", the controller can specify a range of outputs that will not turn off during e-stop or enable.

- These parameters can come in useful when the controller starts running a program during e-stop ("Other #2" set to 1~64) or enable ("Other #3" set to 1~64).

- These parameters all work after a program has been triggered. In some cases triggering a program at e-stop or disable may be necessary, whether or not a program was running before. The parameter "Other #5" will allow you to start a program once the e-stop or disable has been triggered.

- AutoStart is a parameter (Other #1) that is used to start a program automatically after the controller is reset or power up. In this case the program will always start on it's own (in the AUTO mode) without a signal from a PLC or a serial interface. Input Function Selection 003 (I/O Parameters #3) should also be set to 1. Input Function Selection 003 (I/O Parameters #3) if set to 2 will start the program on the rising edge of and stop the program on the falling edge.

Also, a few parameters need to be changed.

First, in the parameter menu under Other,

Parameter 3 = Program number of the E-Stop program

Parameter 10 = 2

Under I/O Parameters,

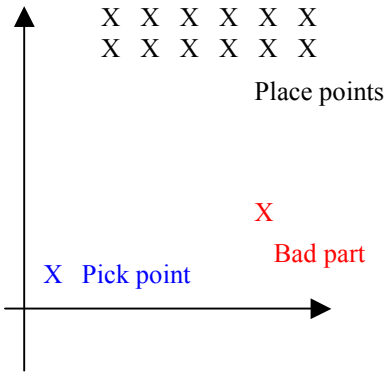
Parameter 35 = 1

In the E-Stop program, a program is aborted and re-executed. The program number should be the number of the AutoStart program. Right now it is set to 1.

Finally, when the e-stop is released, the controller will require Input 5 to be pulsed before returning to normal operation.

Programming Lab II

Part 1



REQUIREMENTS

-Z-axis system w/ pneumatic gripper.

-SOL for Z-axis: 310 on = down
310 off = up

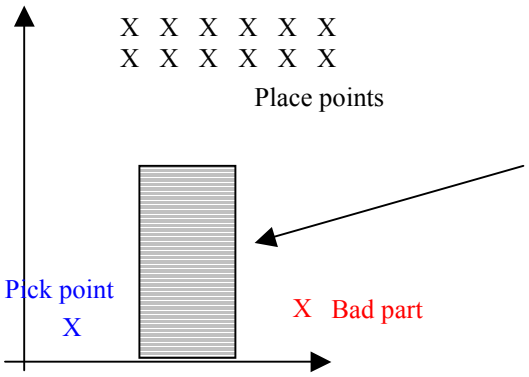
-SOL for gripper: 311 on = close
311 off = open

-Z-axis limit sw. up = 15 on

-Z-axis limit sw. dn. = 16 on

-Good part? 20 on.

Part 2



Object blocking path to bad part bin and other points.

Palletizing Examples

	B	E	N	Cnd	Cmnd	Operand 1	Operand 2	Pst
1		*Pallet one						
2					SVON	11		
3					VEL	400		
4								
5					BGPA	1		
6					PAPI	4	3	
7					PAPN	1		
8					PAPS	2992		
9					EDPA			
10								
11					TAG	2		
12					PSET	1	1	
13					TAG	1		
14					MOVL	1		
15					PMVL	1		
16					PINC	1		900
17					TIMW	1		
18				900	GOTO	1		
19					GOTO	2		
20								

4 x 3 Standard Pallet defined by 3 positions

PAPI defines the size of the pallet. Here is an example of the 4x3 pallet. (4 on X-axis, 3 on Y axis)
PAPN defines the pattern that is executed (0 or 1).
PAPS defines the size of the pallet by three points that start in 2992 of the point table. The first point starts at the first point of the pallet, the second is at the end of the x-axis (primary), the third at the end of the y-axis (secondary)
PSET starts Pallet one at it's first point.
PMVL moves to the pallet.
PINC sets the next point in the pallet.
Flag 900 turns on when the pallet is complete.

The points change the shape/angle of the pallet.

.	B	E	N	Cnd	Cmnd	Operand 1	Operand 2	Pst
1					SVON	11		
2					VEL	400		
3								
4					BGPA	1		
5					PASE	2	1	
6					PAPN	0		
7					PAPI	4	3	
8					PAPT	15	30	
9					PAST	2		
10					EDPA			
11								
12					TAG	2		
13					PSET	1	1	
14					TAG	1		
15					MOVL	1		
16					TIMW	0.5		
17					PMVL	1		
18					PINC	1		900
19					TIMW	0.5		
20				900	GOTO	1		
21					GOTO	2		
22								

Example of a 4x3 pallets defined by pitch (distance between pallet points)

PASE sets the primary axis
PAPN sets the pattern
PAPI sets the size of pallets
PAPT defines the offset of the pallet (the pitch for your axis)
PAST set the origin off your pallet
PSET starts your pallet at position one

.	B	E	N	Cnd	Cmnd	Operand 1	Operand 2	Pst
1		*Pallet two						
2					SVON	11		
3					VEL	400		
4								
5					BGPA	1		
6					PAPI	4	8	
7					PAPN	1		
8					PAPS	2992		
9					PSLI	20	2	
10					EDPA			
11								
12					TAG	2		
13					PSET	1	1	
14					TAG	1		
15					MOVL	1		
16					PMVL	1		
17					PINC	1		900
18					TIMW	1		
19				900	GOTO	1		
20					GOTO	2		
21								

Example of a Zig-Zag Pattern

PAPI - 4 (points on the primary axis) x 8 (points on the secondary axis)
PAPN – Pattern to visit the pallet points
PAPS – the three points in the point table that define the pallet
PSLI – the first point is indented (along the primary axis-20mm) and the number if pallet points on that row