ESTUN-Trio Products & Solution's Technical

iranieq
02165612812
ESTUN Servo Systems
TRIO Motion Controller
PROMAX Srl CNC Controller

TRIO MOTION TECHNOLOGY
A MEMBER OF THE ESTUN GROUP

ESTUN AUTOMATION

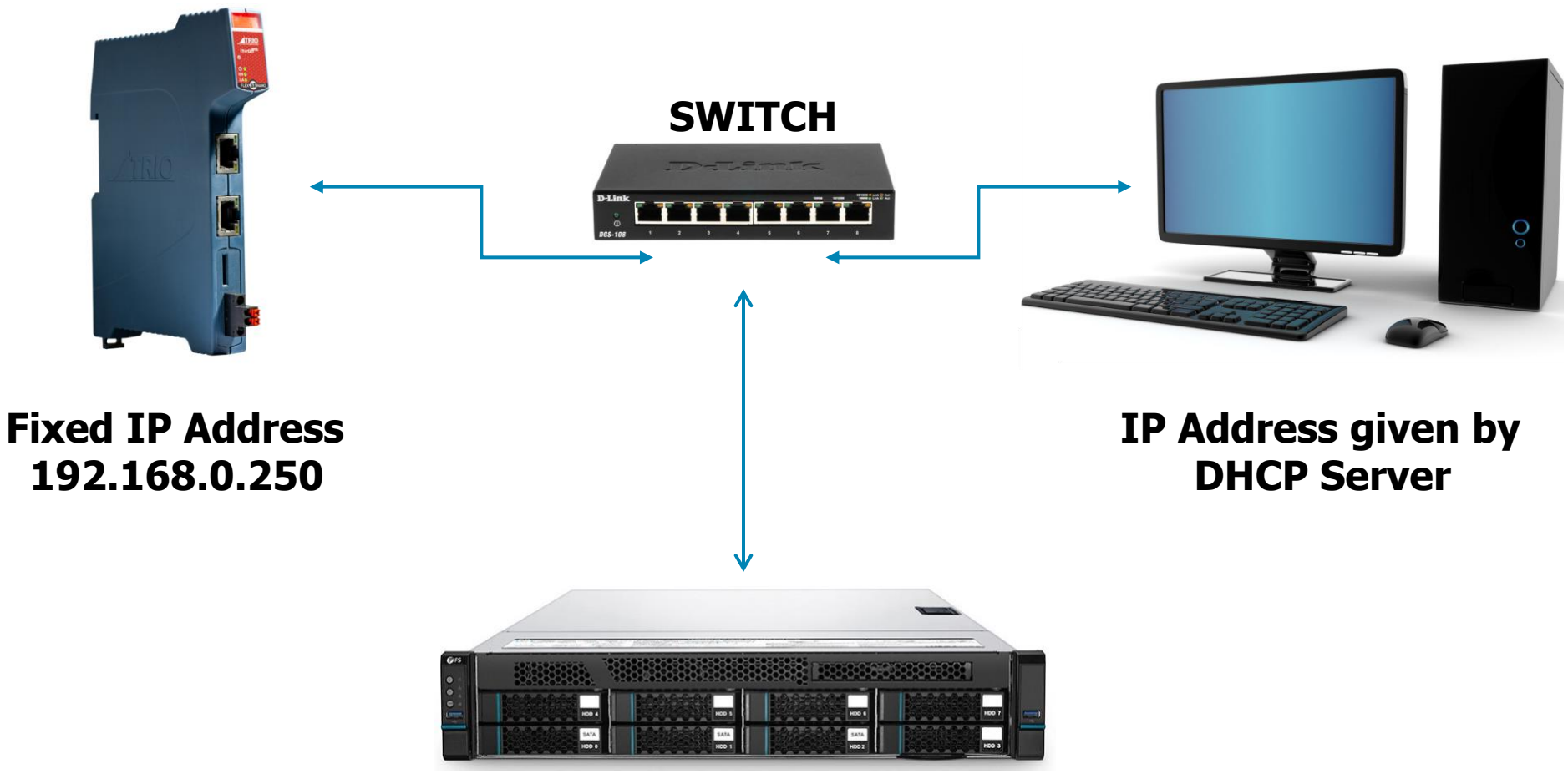
Motion Perfect Editor

Motion Perfect Editor V5

Design, Develop, Test, Deploy and Secure

***Motion* Perfect** provides the user with an easy-to-understand interface for rapid application development, including configuration and setup of drives, controllers and HMI.
A fully featured IDE for program development and debugging in all Motion-iX languages, multi-page HMI screen development and diagnostic tools for machine commission. *Motion* Perfect allows complete machine setup from a single software package.

Ethernet connection



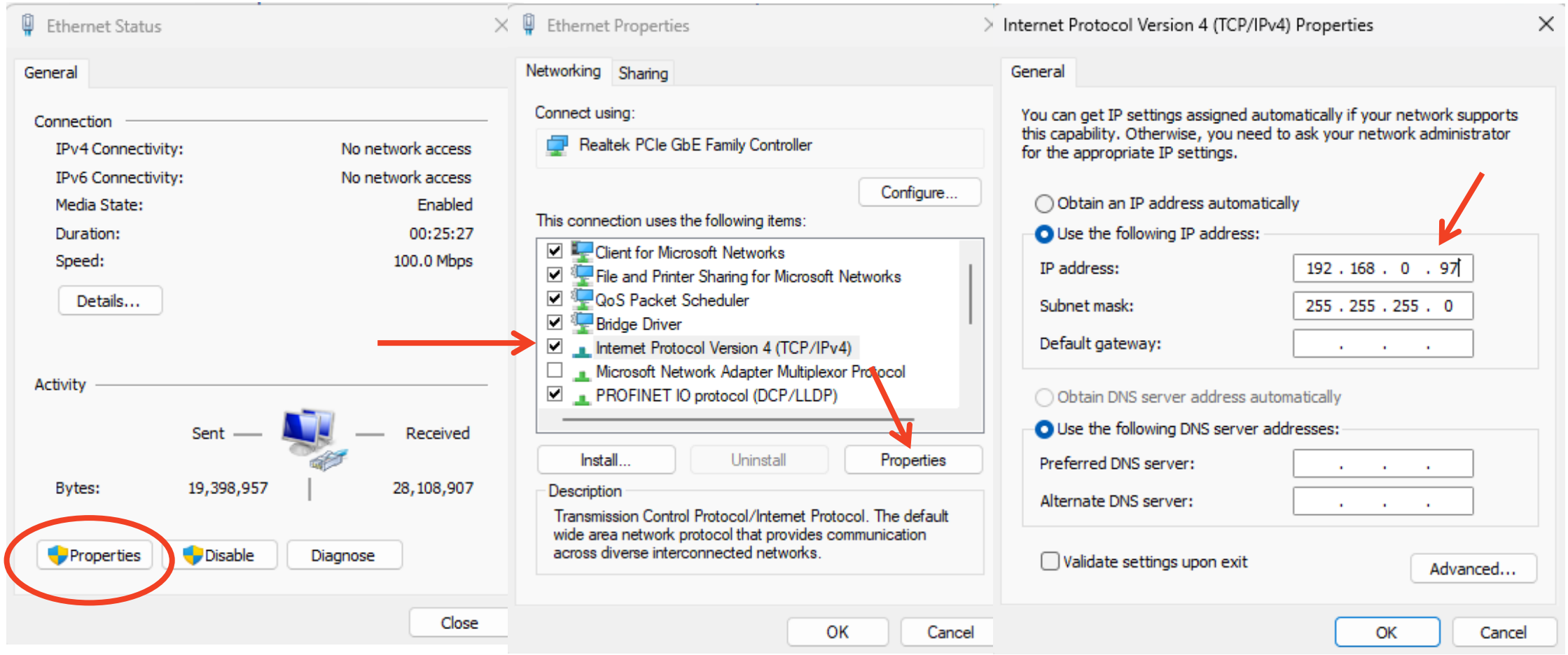
Ethernet connection

SWITCH

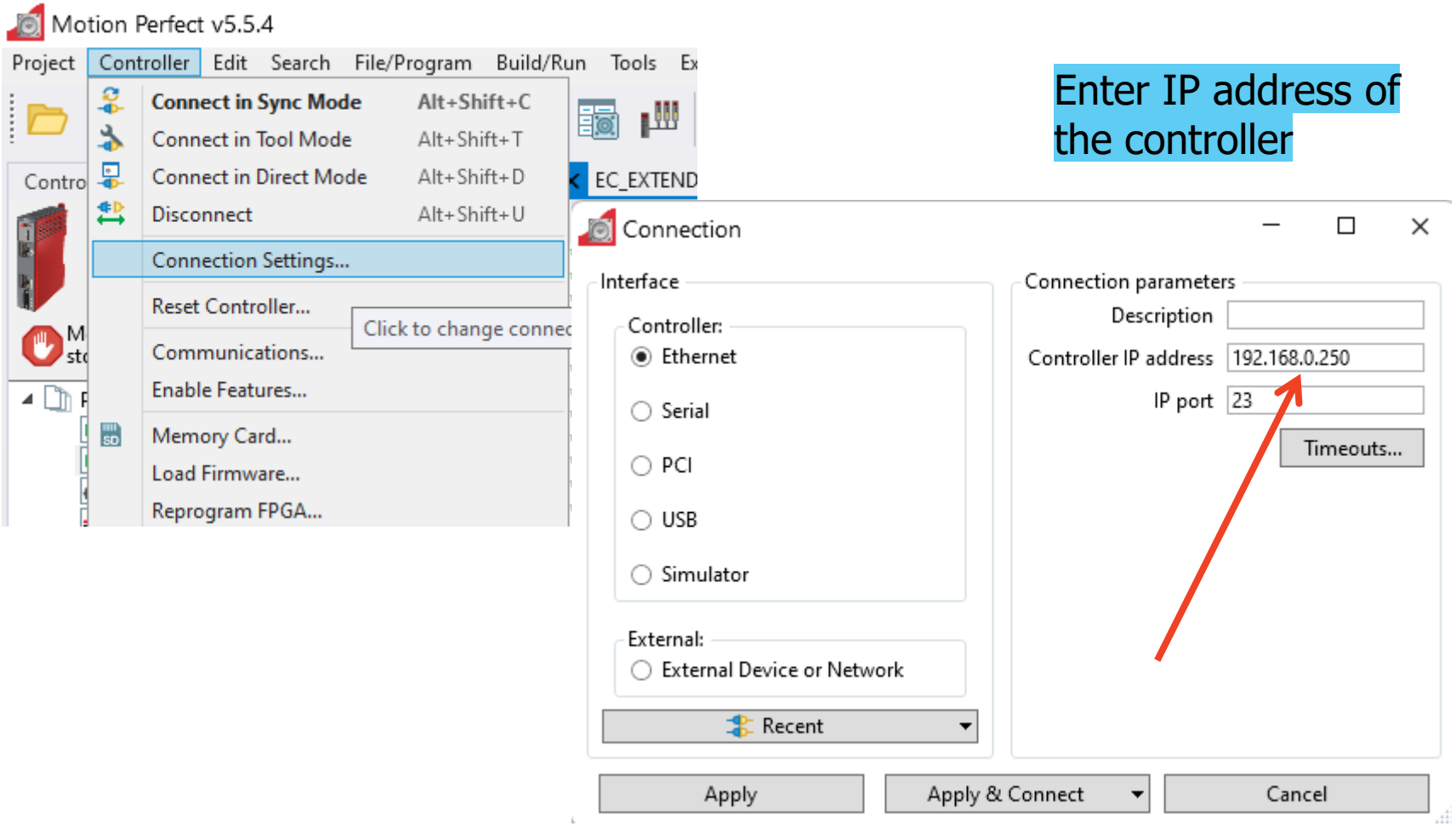
Fixed IP Address
192.168.0.250

Needs Fixed IP Address
e.g. 192.168.0.97

Set a Fixed IP Address



Motion Coordinator address



Enter IP address of the controller

Operation Modes

Disconnected

Not connected to a controller. All tools are closed, and no communications ports are open.

Direct Mode

A direct connection is made to a controller allowing a Terminal tool to be used for direct interaction with the command line on the controller.

Tool Mode

A multichannel connection is made to a controller allowing the monitoring tools within *Motion* Perfect to be used. This mode allows the user to see a list of the programs on the controller (so that they can be started and stopped) but does not allow editing of any of the programs.

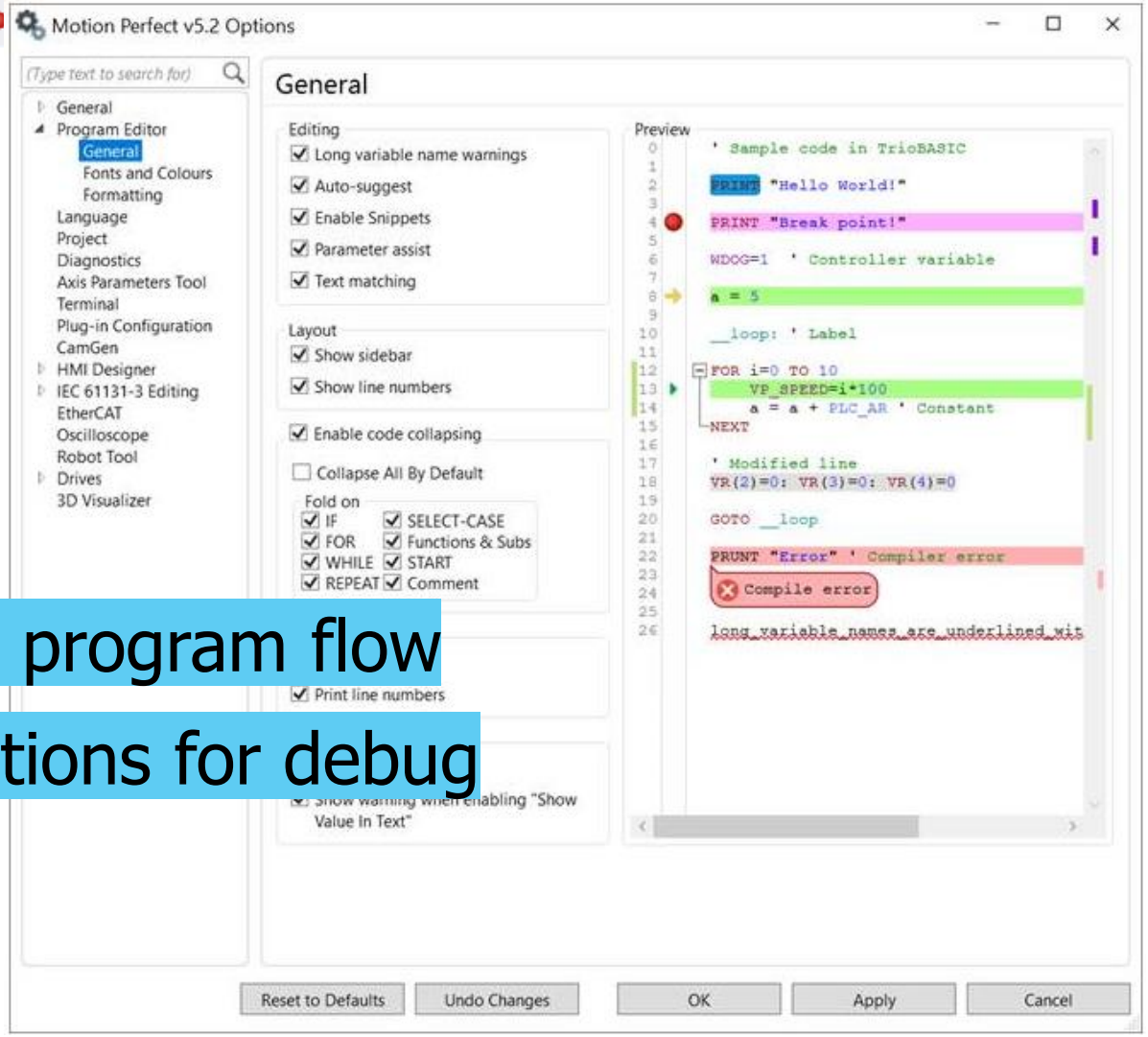
Sync Mode

A multichannel connection is made to a controller and a local project on the PC is opened. The contents of the controller and the project are synchronized so that the local copy of all programs matches those on the controller. All of *Motion* Perfect's tools are available and programs can be edited. The synchronization process can involve deleting programs or copying them from the controller to the PC of vice versa.

BASIC Program

STARTUP

```
'Start Ethernet port section
IP_ADDRESS = 192.168.0.250

WA(5000) ' 5 sec delay-warm up
ETHERCAT(0,0) 'ethercat running command

'RUN "CLEAR_UNIT_ERROR" 'calling sub routine

WA(500) ' 0.5  sec

'(1) - ADDRESS
IF CO_READ(0,1,$6860,0,2) <> 8 THEN
    CO_WRITE(0,1,$6860,0,2,-1,8)
ENDIF

WA(500) ' 0.5 sec
```



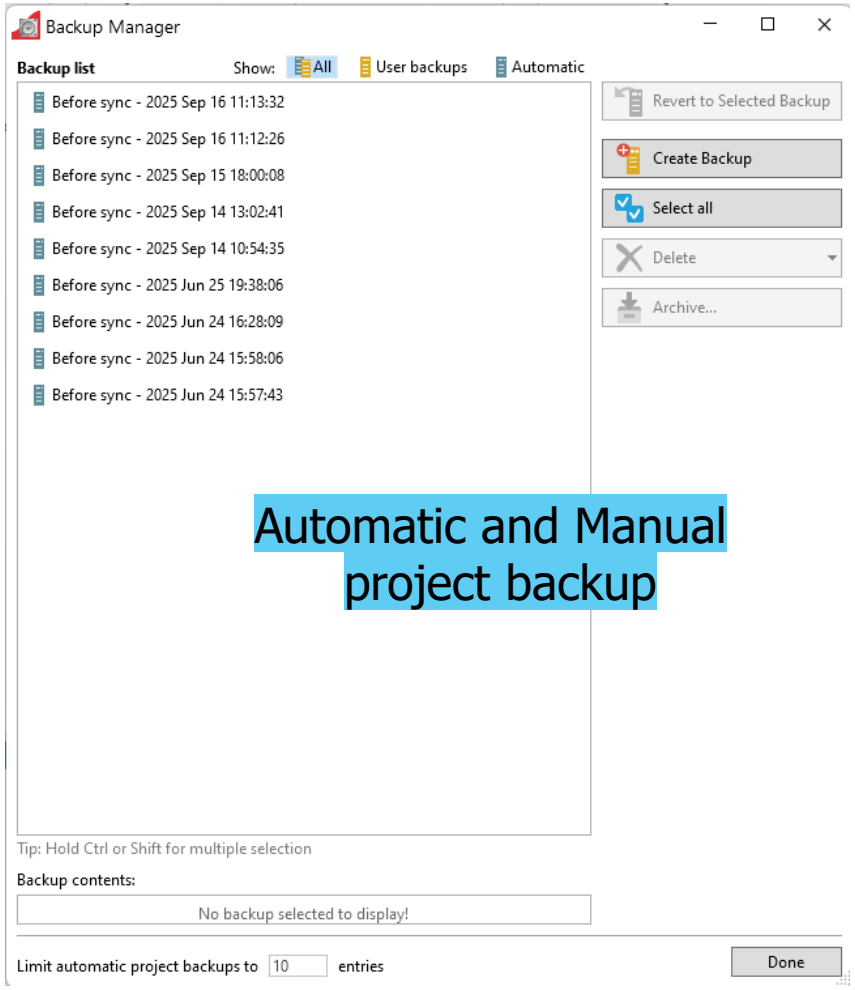
Full function program Editor

Auto-indent help to highlight program flow

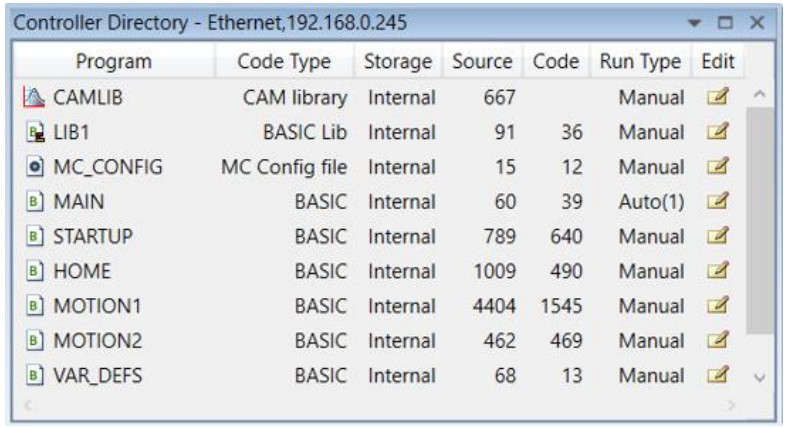
RUN, Step, Breakpoint operations for debug

Tailored color template

Backup manager



Automatic and Manual project backup



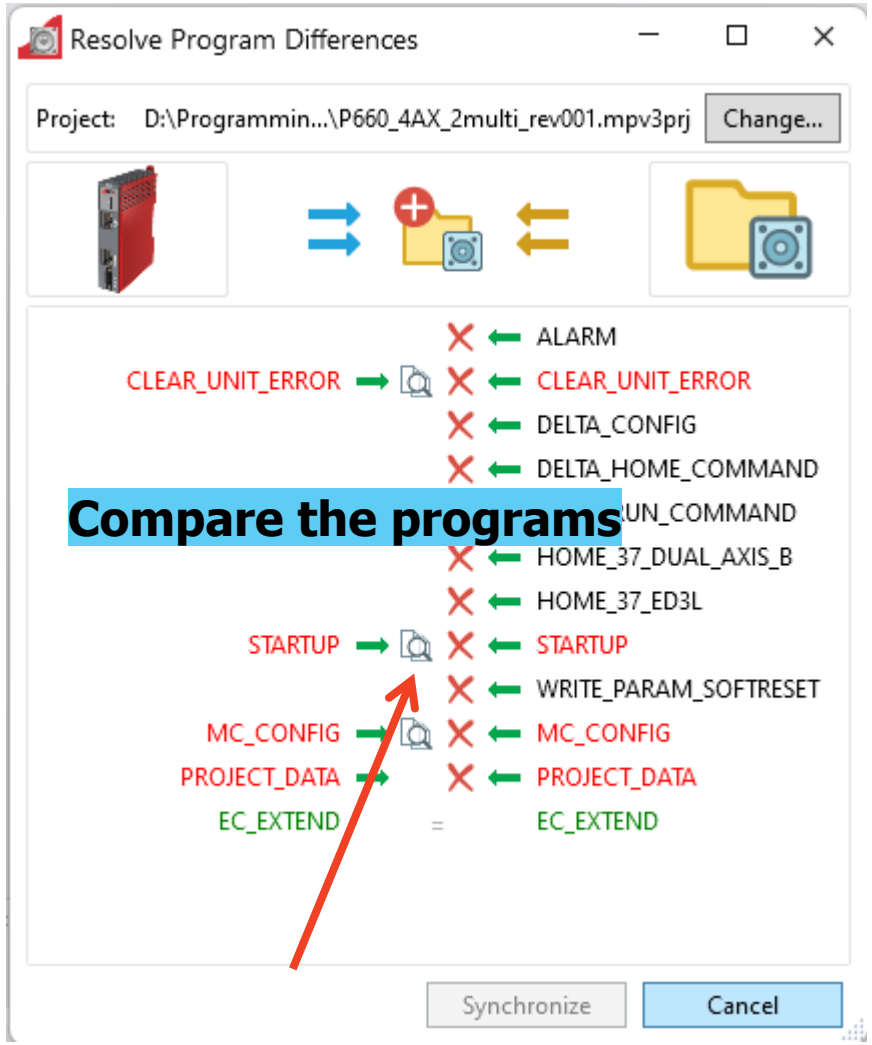
Controller Directory - Ethernet,192.168.0.245

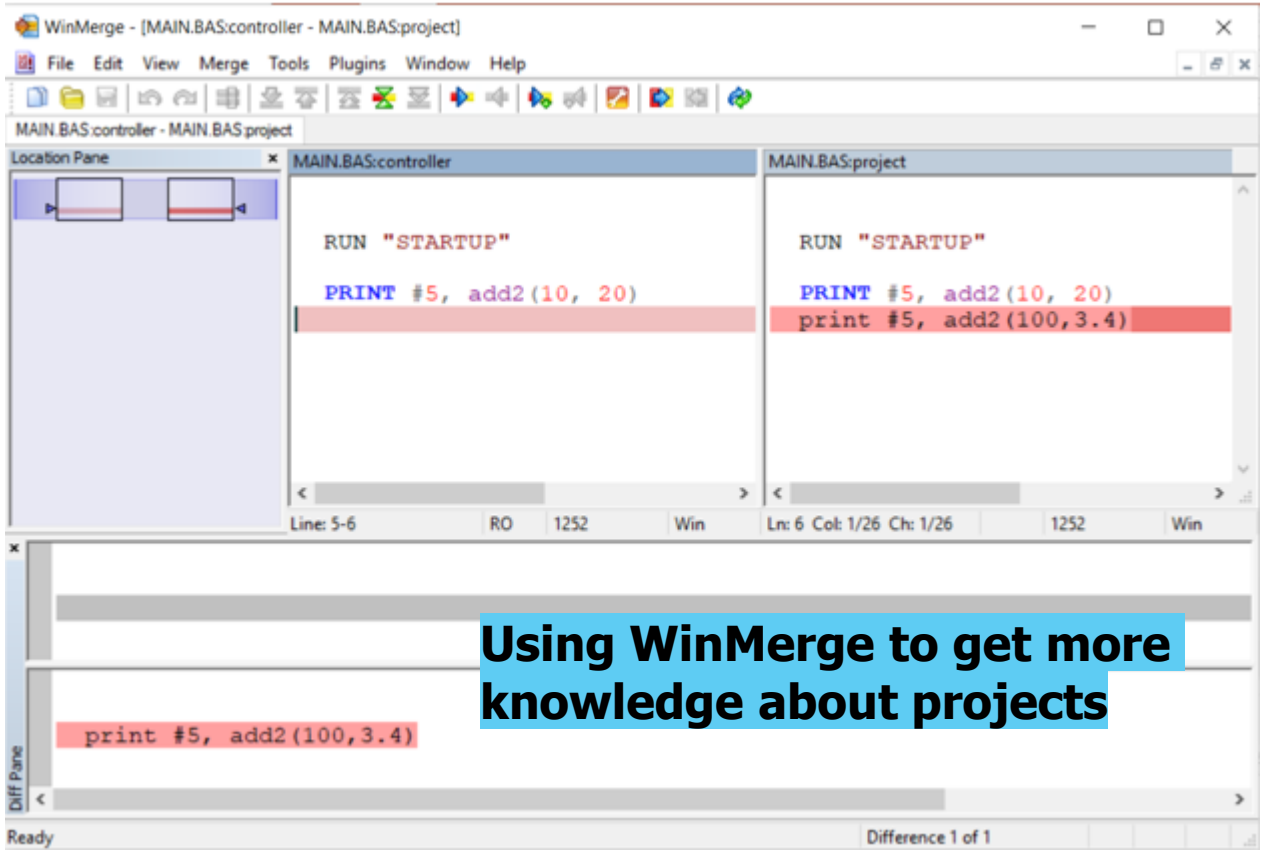
Program	Code Type	Storage	Source	Code	Run Type	Edit
CAMLIB	CAM library	Internal	667		Manual	
LIB1	BASIC Lib	Internal	91	36	Manual	
MC_CONFIG	MC Config file	Internal	15	12	Manual	
MAIN	BASIC	Internal	60	39	Auto(1)	
STARTUP	BASIC	Internal	789	640	Manual	
HOME	BASIC	Internal	1009	490	Manual	
MOTION1	BASIC	Internal	4404	1545	Manual	
MOTION2	BASIC	Internal	462	469	Manual	
VAR_DEFS	BASIC	Internal	68	13	Manual	

The Source code is stored in Flash memory

A copy is automatically saved to the PC project folder

Project Synchronization





Local Variables

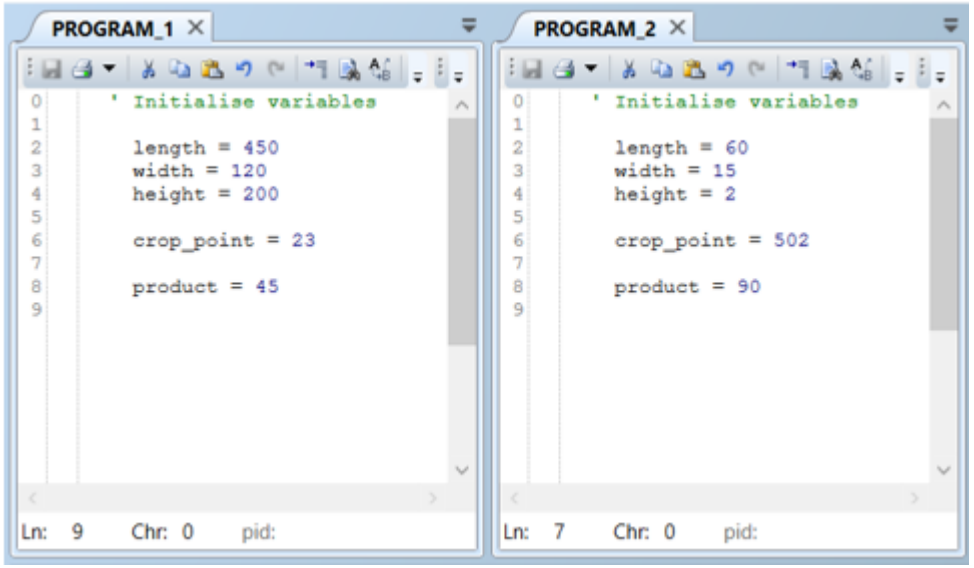
Simple case, just name the variables as you use them

X = 123 ‘create a variable called x and give it value
Offset = 12 ‘create another variable
Y = 30 * x + Offset

Or define your variable name and type before use

DIM flow_rate AS INTEGER
DIM calibration_factor AS FLOAT

Flow_rate = 24
Calibration_factor = 5.614 * PI

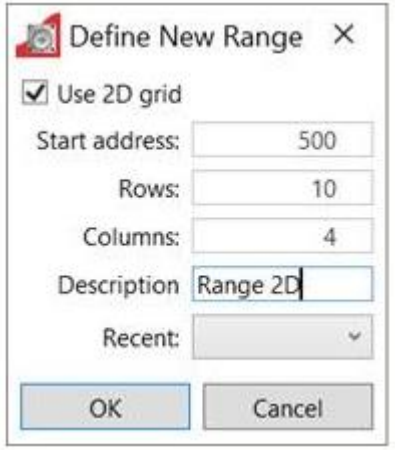


Each program keeps its own list of variables

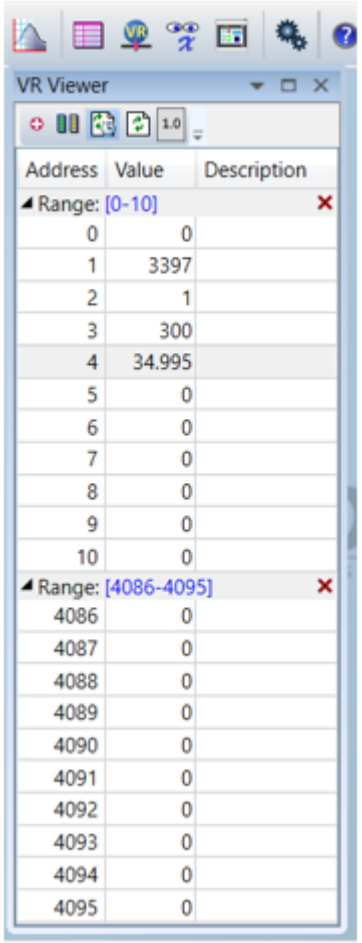
Local variables are not known to other programs
1024 local variables for each program

Global VR memory

- Motion PLC & MC403,MC404,MC405
  - VR(0) to VR(4095)
    - Except MCS50
      - VR(0) to VR(8191)
- Motion control & MC508, MC6N, Flex-X
  - VR(0) to VR(16383)
- Motion control (high performance) & MC664(X)
  - VR(0) to VR(65535)



Non-volatile; automatic Flash store.



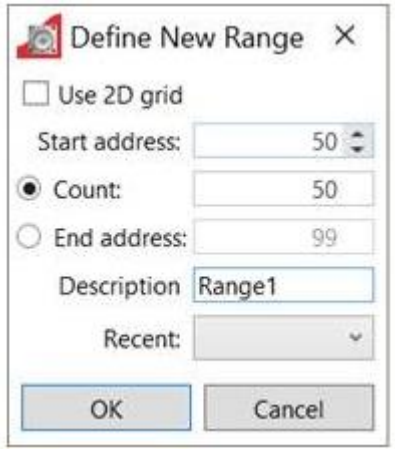

```
Syntax:        VR(index) = value
               value = VR(index)
```

Global TABLE memory

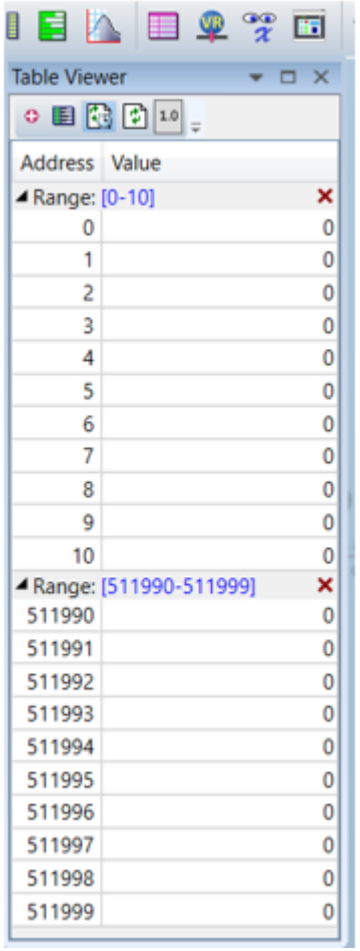
- All motion Coordinators:
  512.000 x 64 bit floating point values.

  TABLE(0) to TABLE(511999)

```
Syntax:
TABLE(index, value1, value2, value3, value4)

value_n = TABLE(index_n)
```



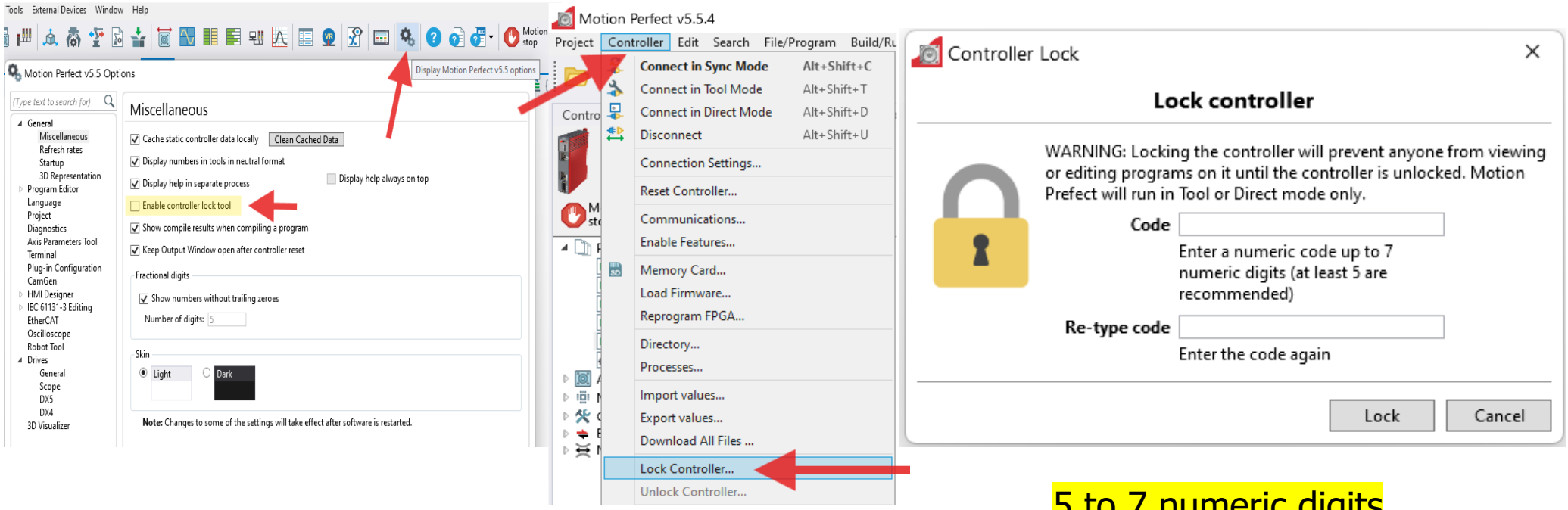
Volatile, except MC508 and MC664(X)



Lock Project

Locking the controller will prevent any unauthorized user from viewing or modifying the programs in memory, and also prevent *Motion* Perfect from connecting in Sync mode.
-To Lock the currently connected controller, select "Controller / Lock Controller" from the main menu. For this to be available it must be enabled in the Options Dialog (" General / Miscellaneous " page) and Motion Perfect must be connected to the controller in Sync mode.

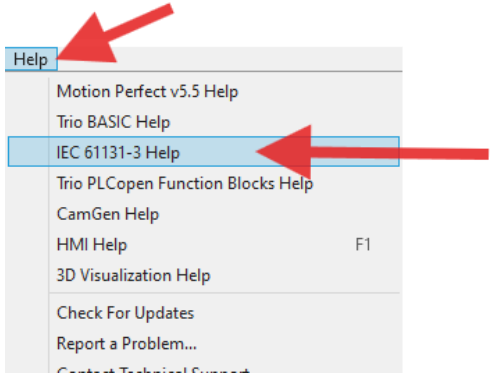
To Lock The Project

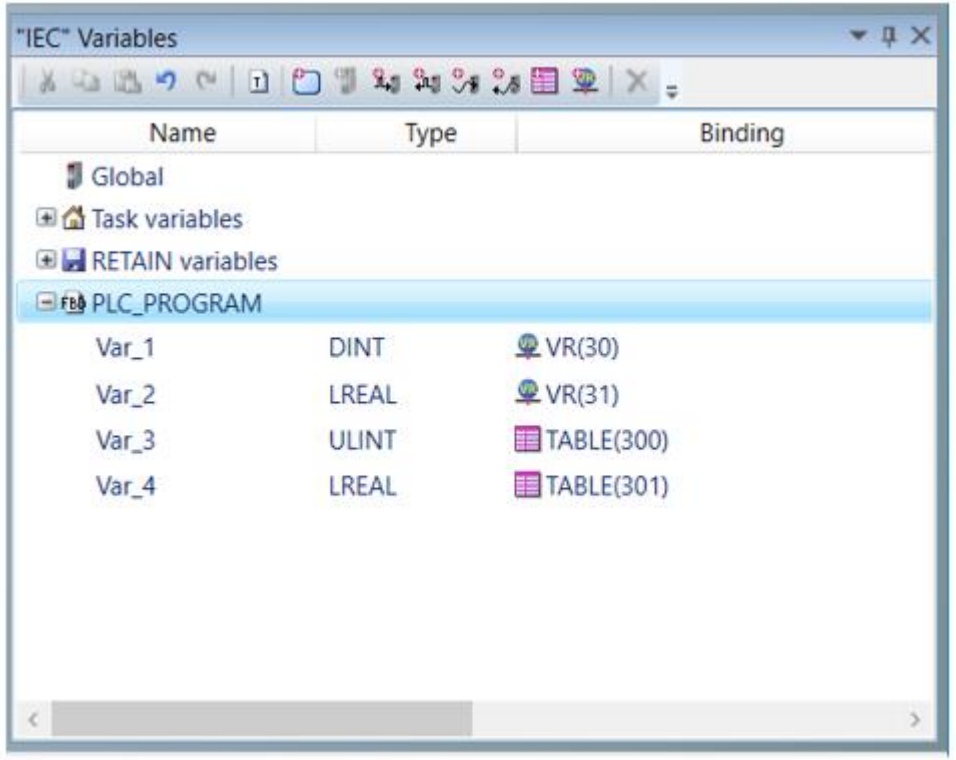


5 to 7 numeric digits

IEC 61131-3 Bound variables

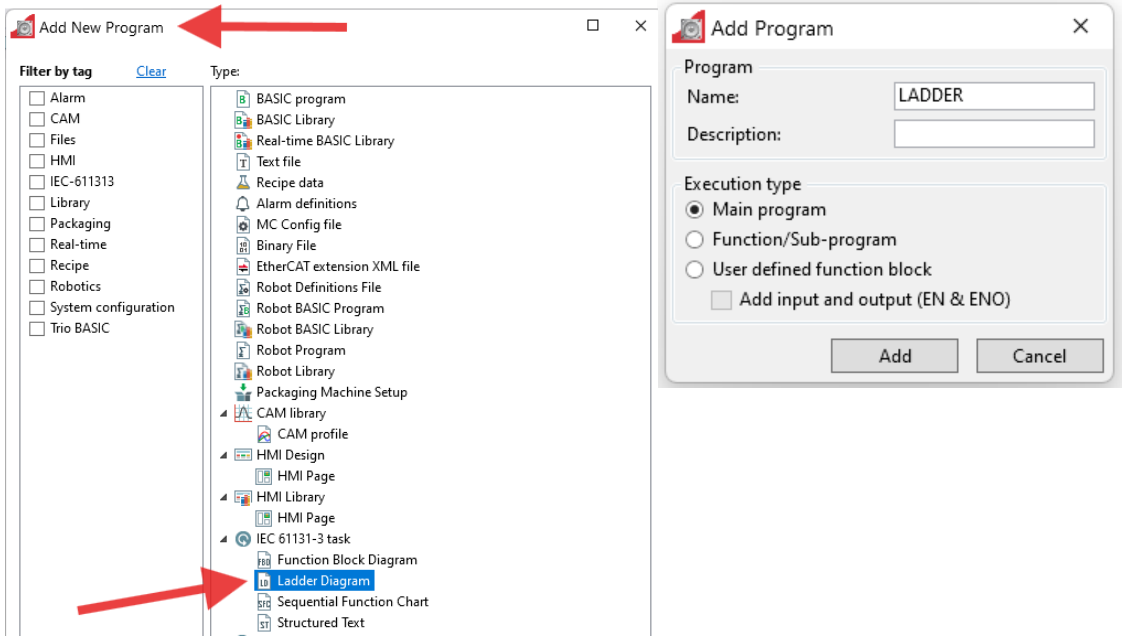
- IEC variables are Bound to either VR or TABLE
- Any valid data type in IEC program
- Mapped to VR or TABLE as Float 64
- Conversion is automatic

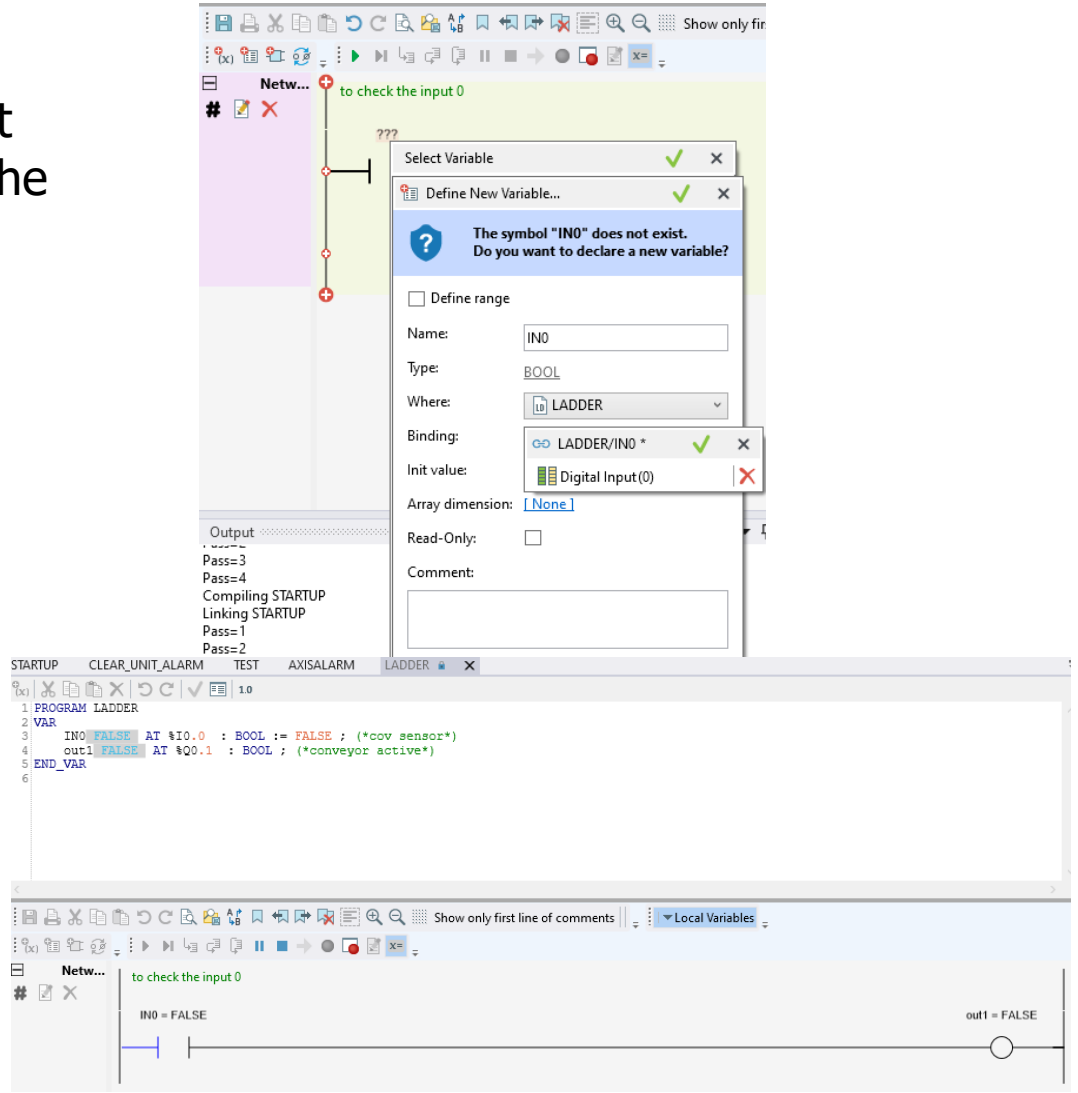




LADDER Programming

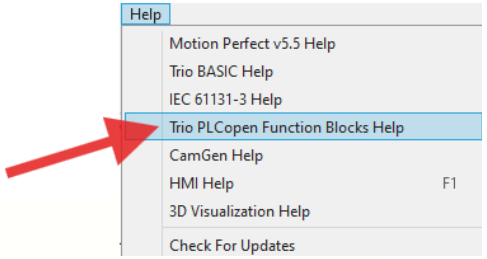
IEC 61131-3 LD language is a graphical programming language. **Ladder logic** is a programming language that represents a program by a graphical diagram based on the circuit diagrams of relay logic hardware.





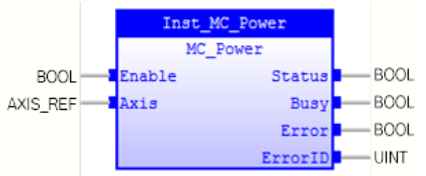
PLCopen

Arrows within the state diagram show the possible state transitions between the states. State transitions due to an issued command are shown by full arrows. Dashed arrows are used for state transitions that occur when a command of an axis has terminated or a system related transition (like error related). The motion commands which transit the axis to the corresponding motion state are listed above the states. These motion commands may also be issued when the axis is already in the according motion state.



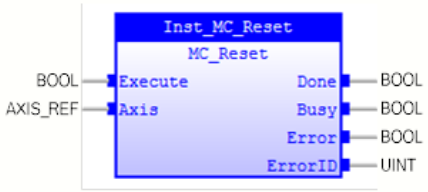
MC_Power

Controls the power stage (ON or OFF).



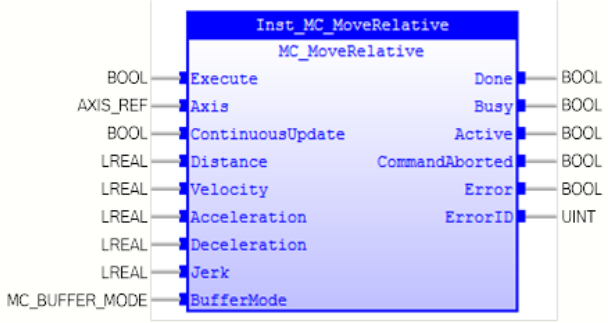
MC_Reset

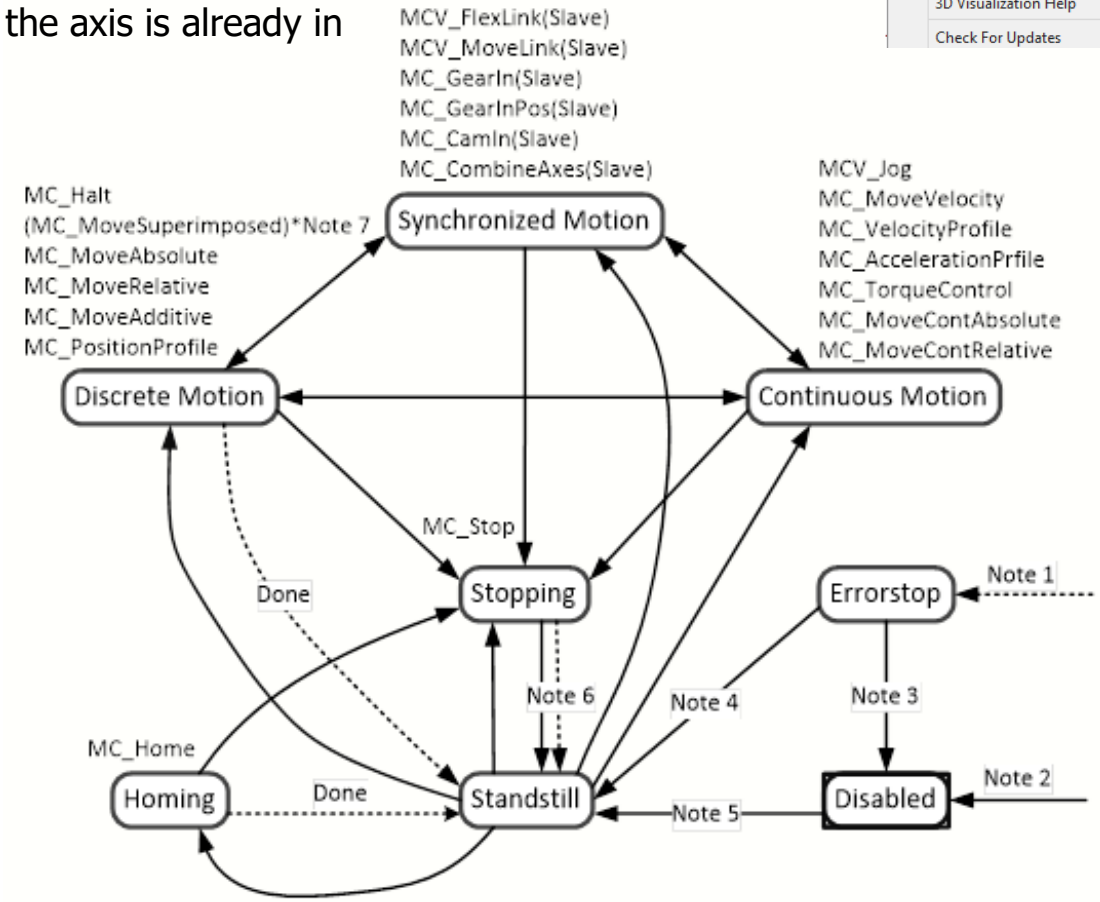
Resets all internal axis-related errors.



MC_MoveRelative

Commands a controlled motion of a specified distance relative to the position at time of the execution.

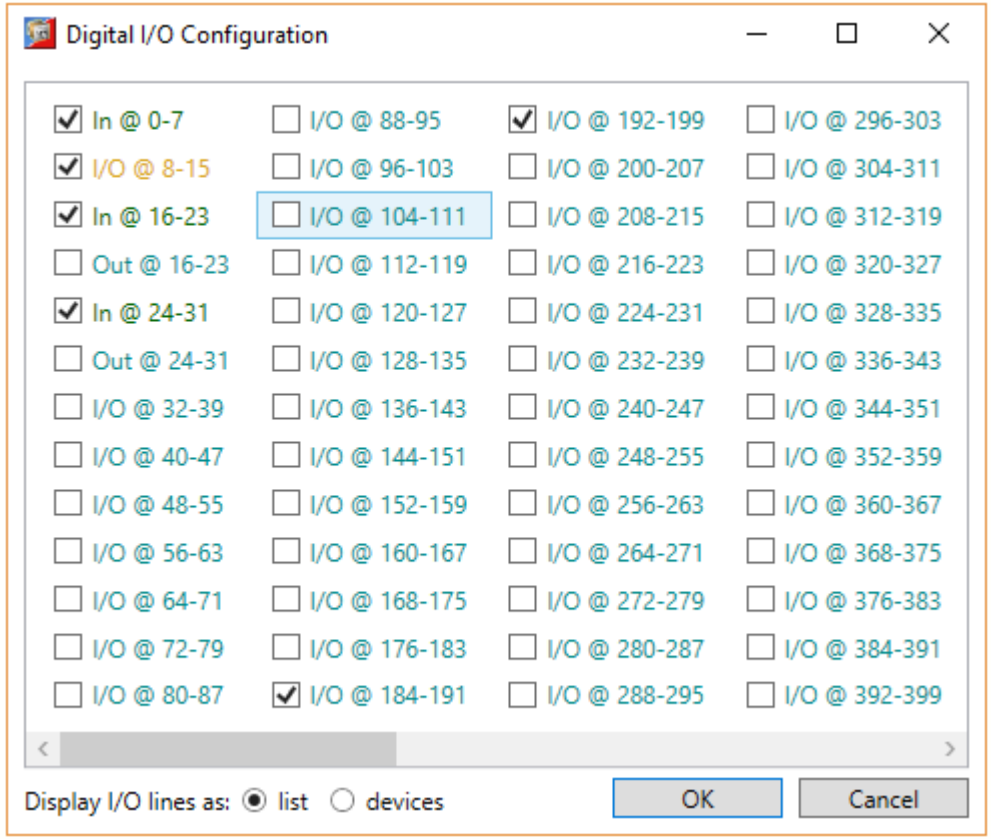


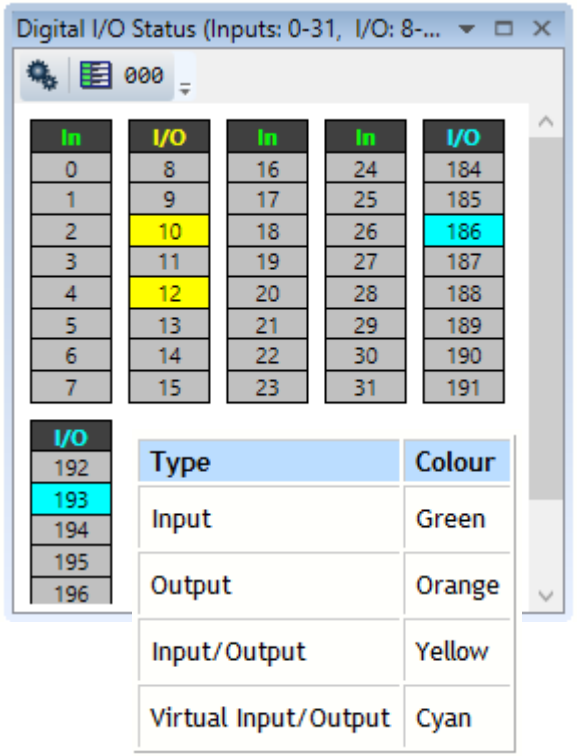


Digital I/O Viewer

The digital I/O viewer is used to show the states of the digital inputs and outputs of the controller (both local and remote).

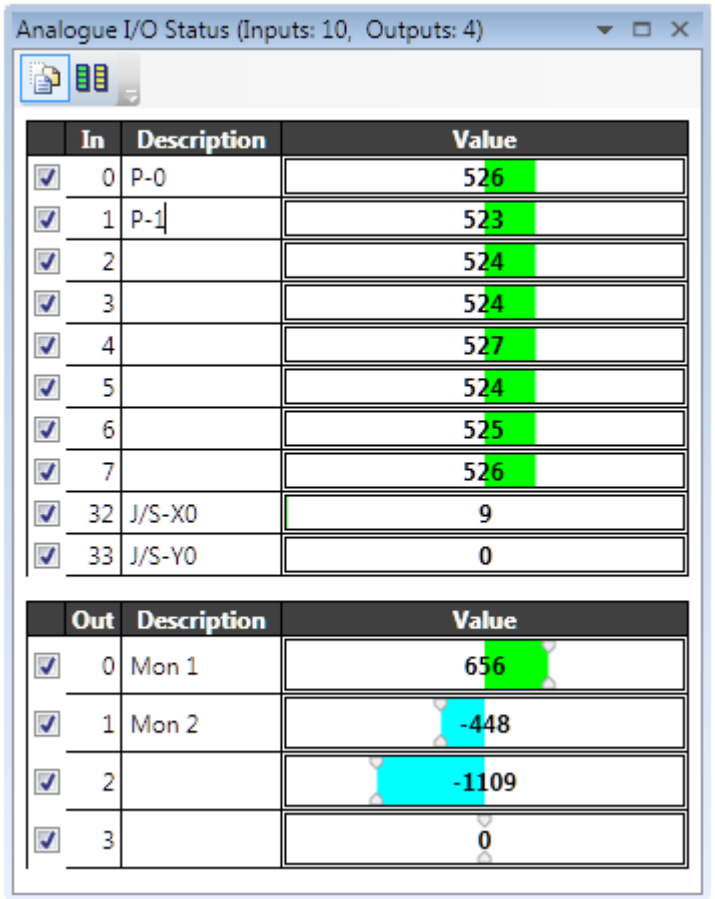
It is possible to change which banks are displayed by clicking on the "Configuration" button which then displays the configuration dialog.

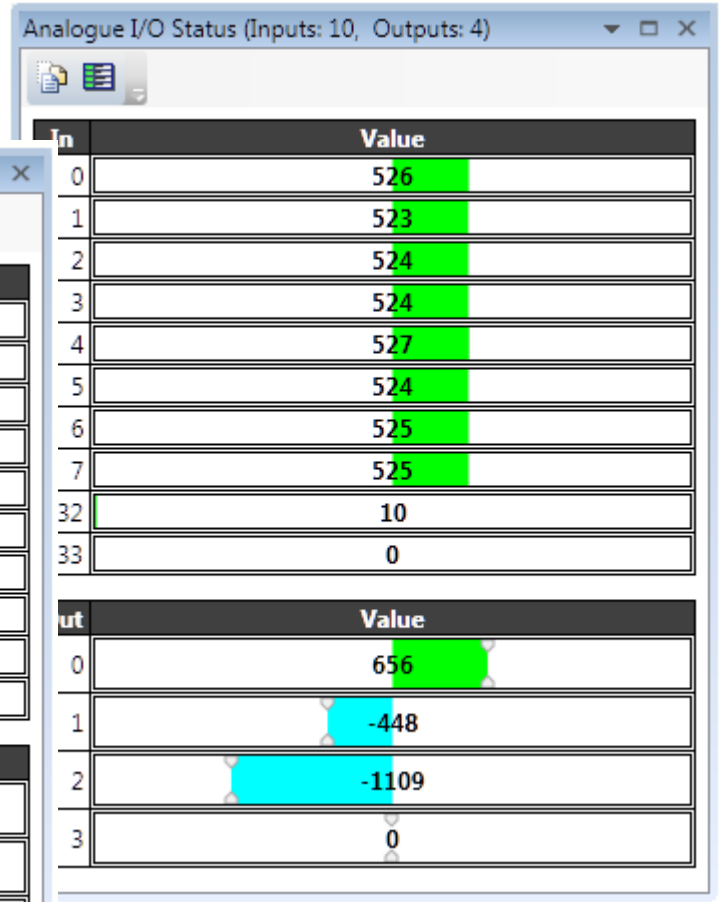




Analog I/O Viewer

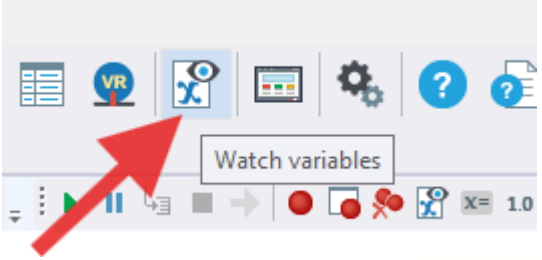
The analogue I/O viewer is used to show the values measured on the analogue inputs and set on the analogue outputs of the controller (both local and remote).





Watch Variables

The "Watch Variables" tool allows the user to look at the values of program internal variables and global variables while a program is running or stepping. It can be used to display variables of all types from all runnable program types.
Up to 4 different Watch Windows can be displayed if the Watch Tool is started from Motion Perfect's main menu. Other means of starting it will only display a single window.



Watch Variables 1

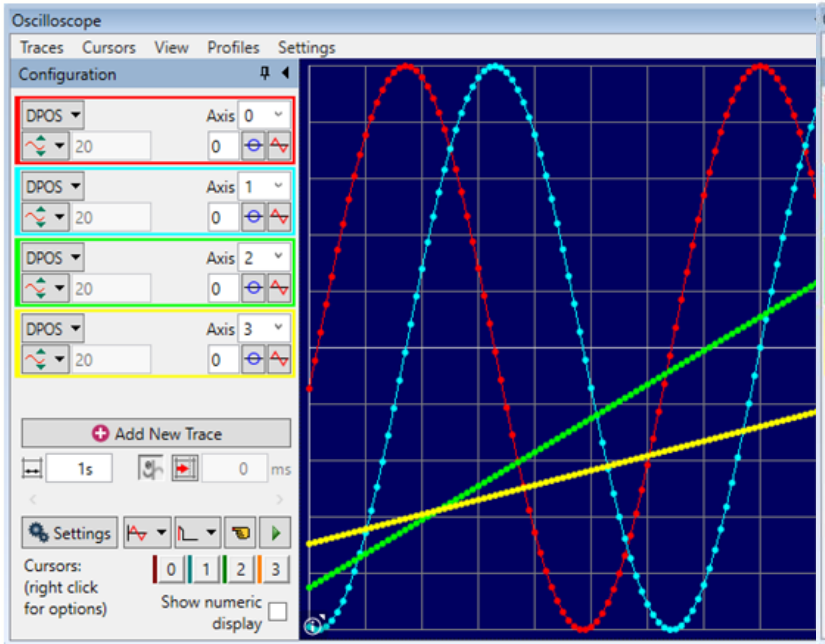
Name	Value	Context
ary(0,0,0)	1852	ARRY
ary(3,2,1)	1974	ARRY
j	32	OUTGEN
k	19	OUTGEN
m	79	ARRY
strng	/0123456789:;<=>?@ABCDEFGHIJKL	ARRY
bb	-1	ARRY,20
IEC_TASK1\IN10	TRUE	IEC_TASK1
IEC_TASK1\AIN32	TRUE	IEC_TASK1
mon1	5	D3D_1
mon2	3.1	D3D_1

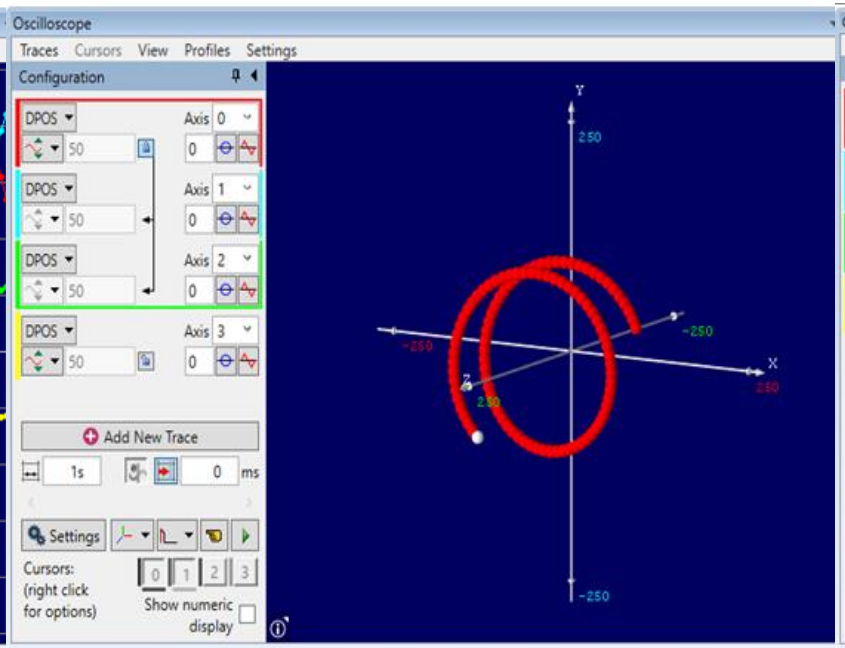
Integer Number Display Format

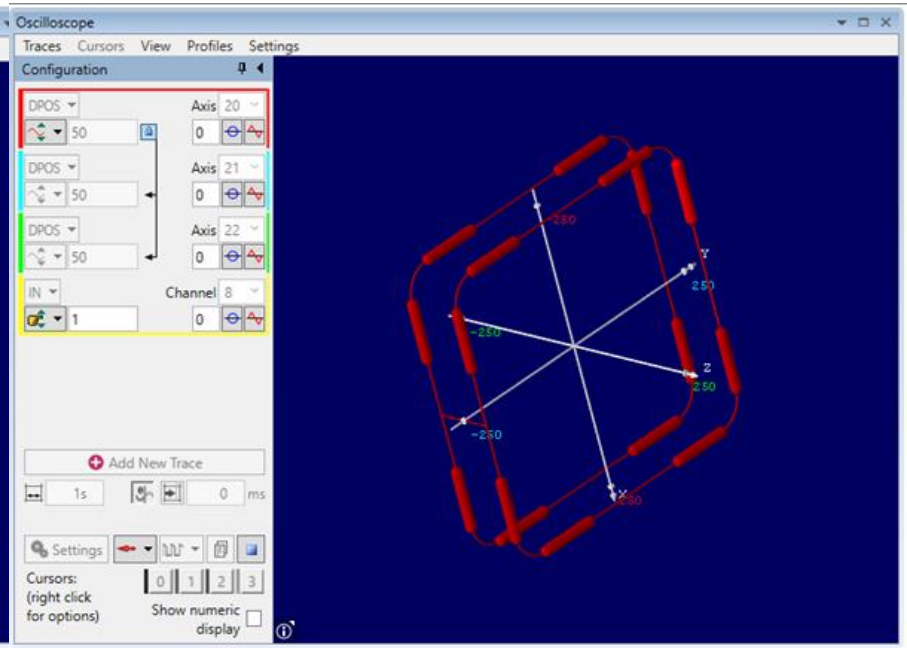
The 1.0 (decimal) or $1 (hexadecimal) button indicates the current format used to display integer numbers. Clicking on the button brings up a drop-down menu to allow the user to change the format.

OSCILLOSCOPE

-The software oscilloscope can be used to trace many of the variables and parameters on the *Motion* Coordinator, aiding program development and machine commissioning.
-There are up to 8 channels available, each capable of recording at up to 1000 samples/sec, with manual cycling or program linked triggering.

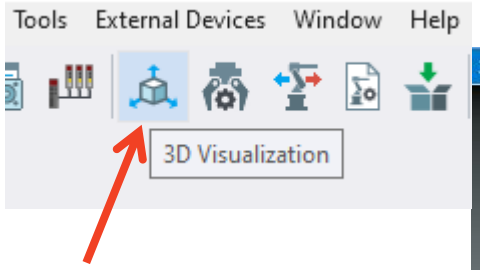


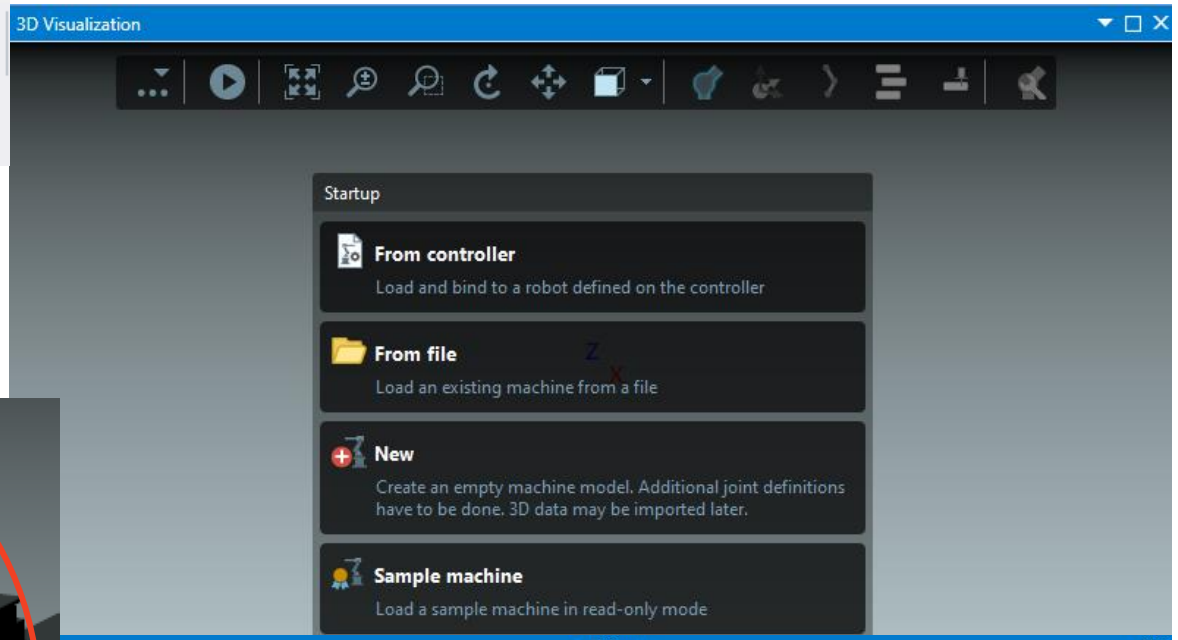


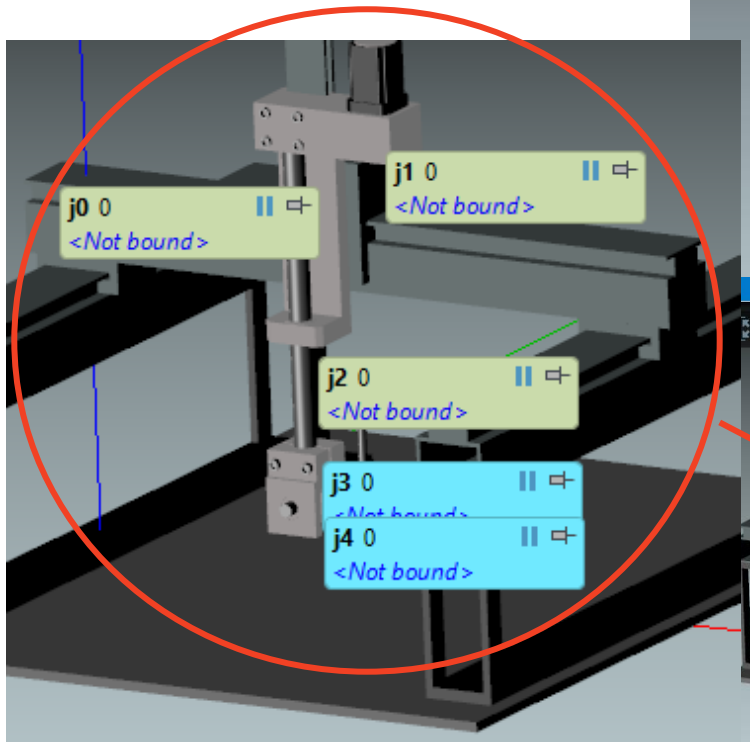


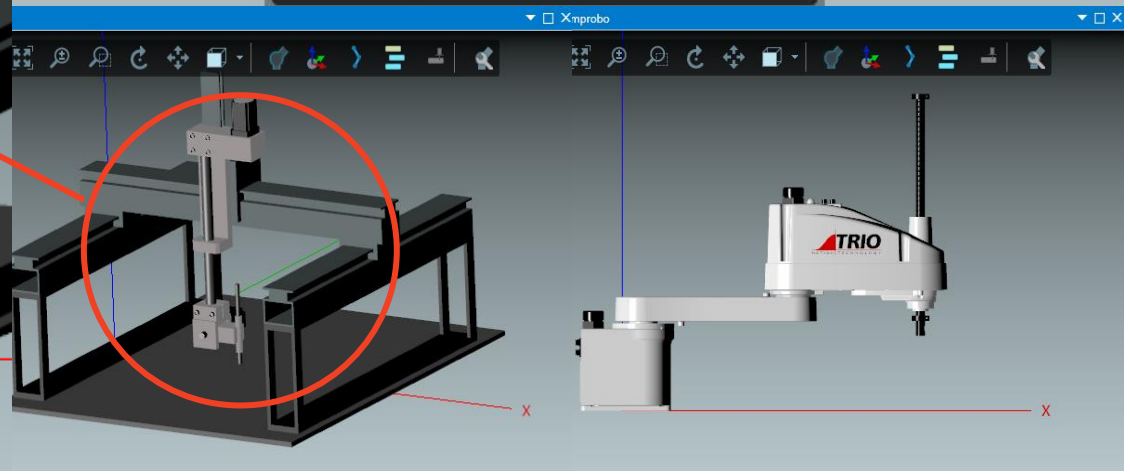
3D Visualization

This is done selecting " *Import 3D geometry...*" from the menu displayed by clicking the visualizer's " More " tool button. A machine model can be created from a 3D model exported by a third-party 3D modelling package. The 3D formats currently supported are OBJ and STL. OBJ is recommended.
Once the 3D data has been imported the machine joints need to be defined by their position in space and orientation (joint axis). The corresponding visual parts from the 3D model need to be bound to a joint so that its movement can be animated.







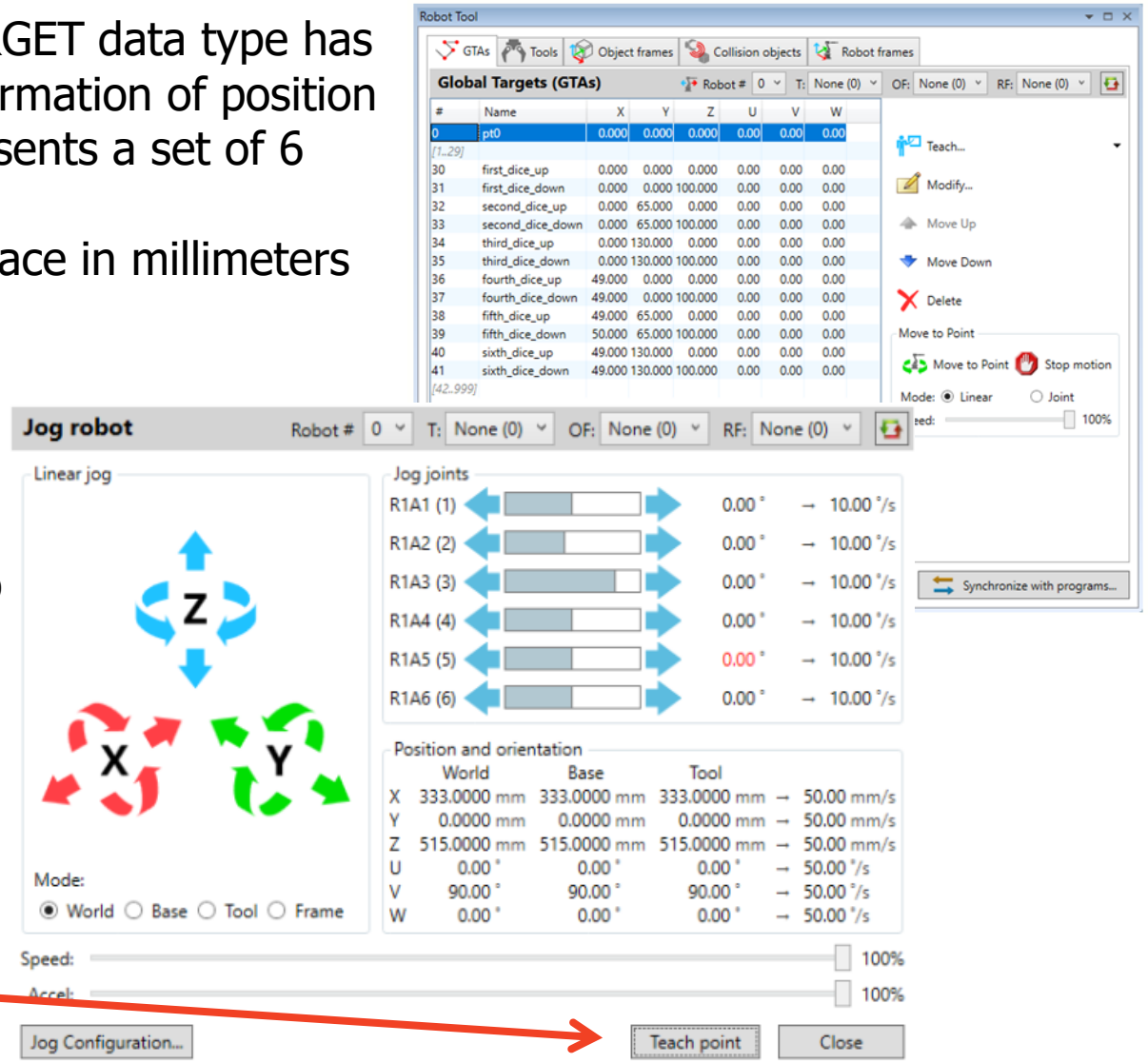


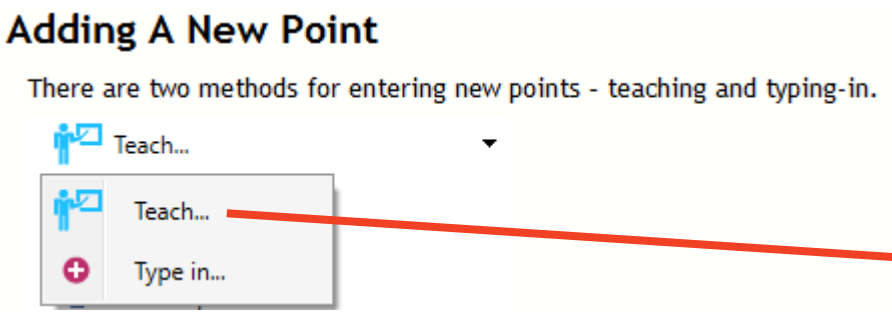
Robot Tool

In association with robot related development, a new TARGET data type has been introduced in the Trio BASIC language. It stores information of position and orientation in 3D space. The TARGET data type represents a set of 6 values:

- X, Y, Z – for the coordinates of the point in 3D space in millimeters
- U, V, W – for the angular orientation in degrees.

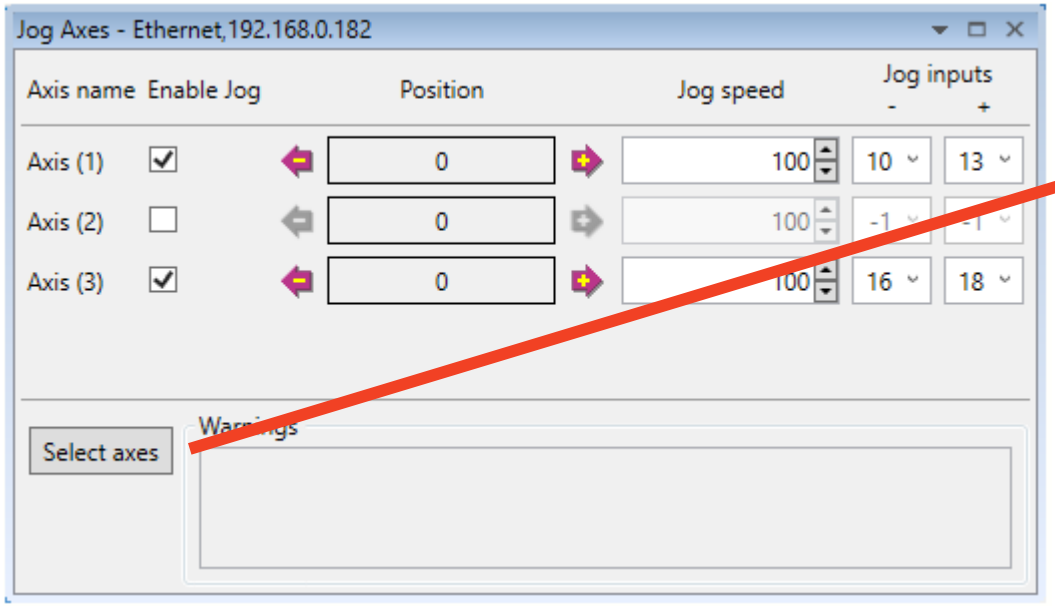
An array of 1000 global target (GTA) points is available for use in all programs. In addition to the 6 coordinates GTAs can have name assigned which can be used to reference them in programs. An Active state flag used to determine whether GTA point has already been defined and contains valid data

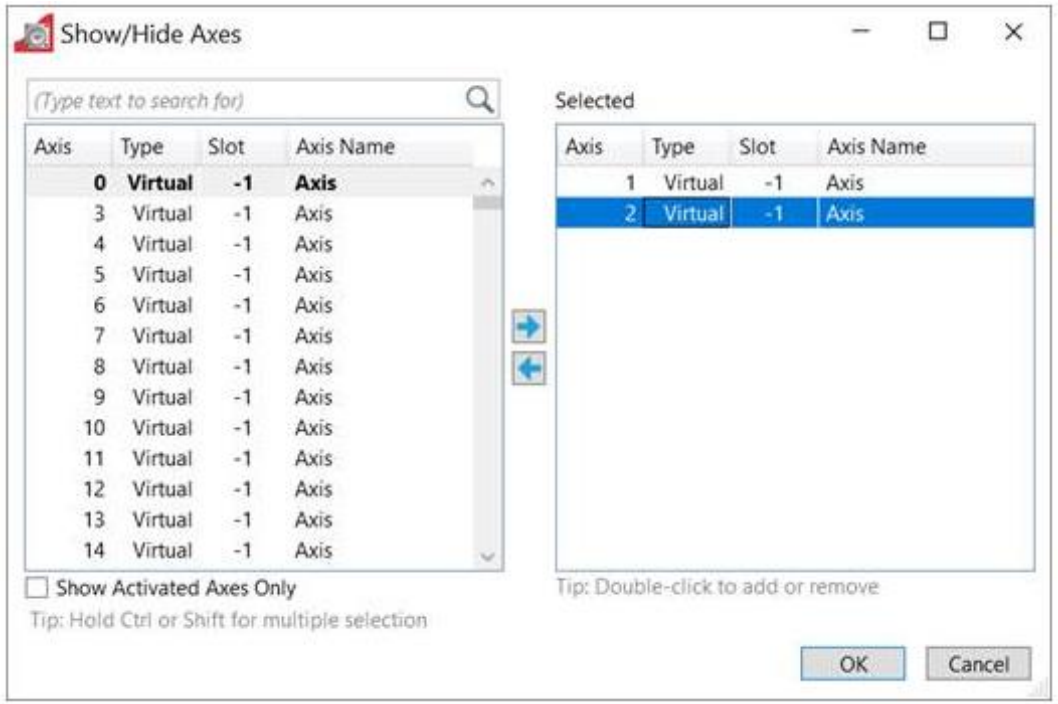




JOG AXIS

The Jog Axes tool allows the user to move the axes on the *Motion* Coordinator.

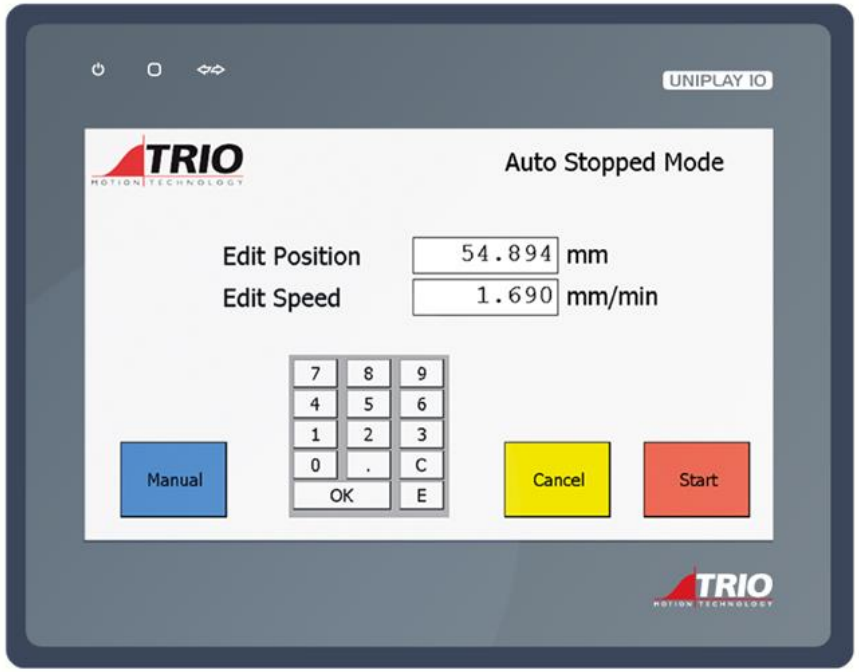


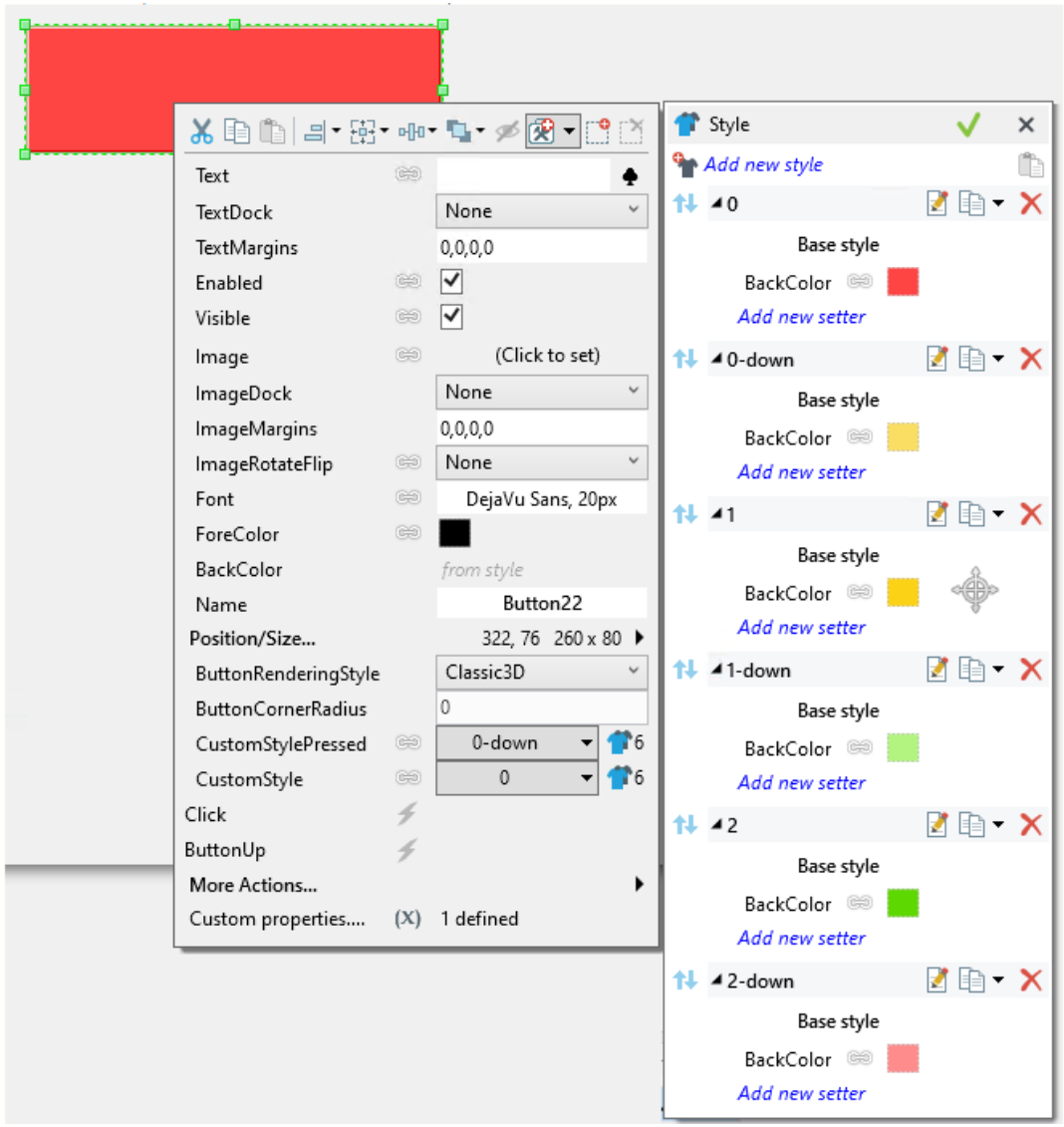


This displays an axis selector box which enables the user to select the axis to include in the jog axes display. By default, the physical axes fitted to the controller will be displayed.

HMI

Uniplay is Trio Motion Technology's HMI system for use in coordination with Trio's series 4 *Motion* Coordinators. It allows interaction with the *Motion* Coordinator using a fully customizable, touch-screen interface.

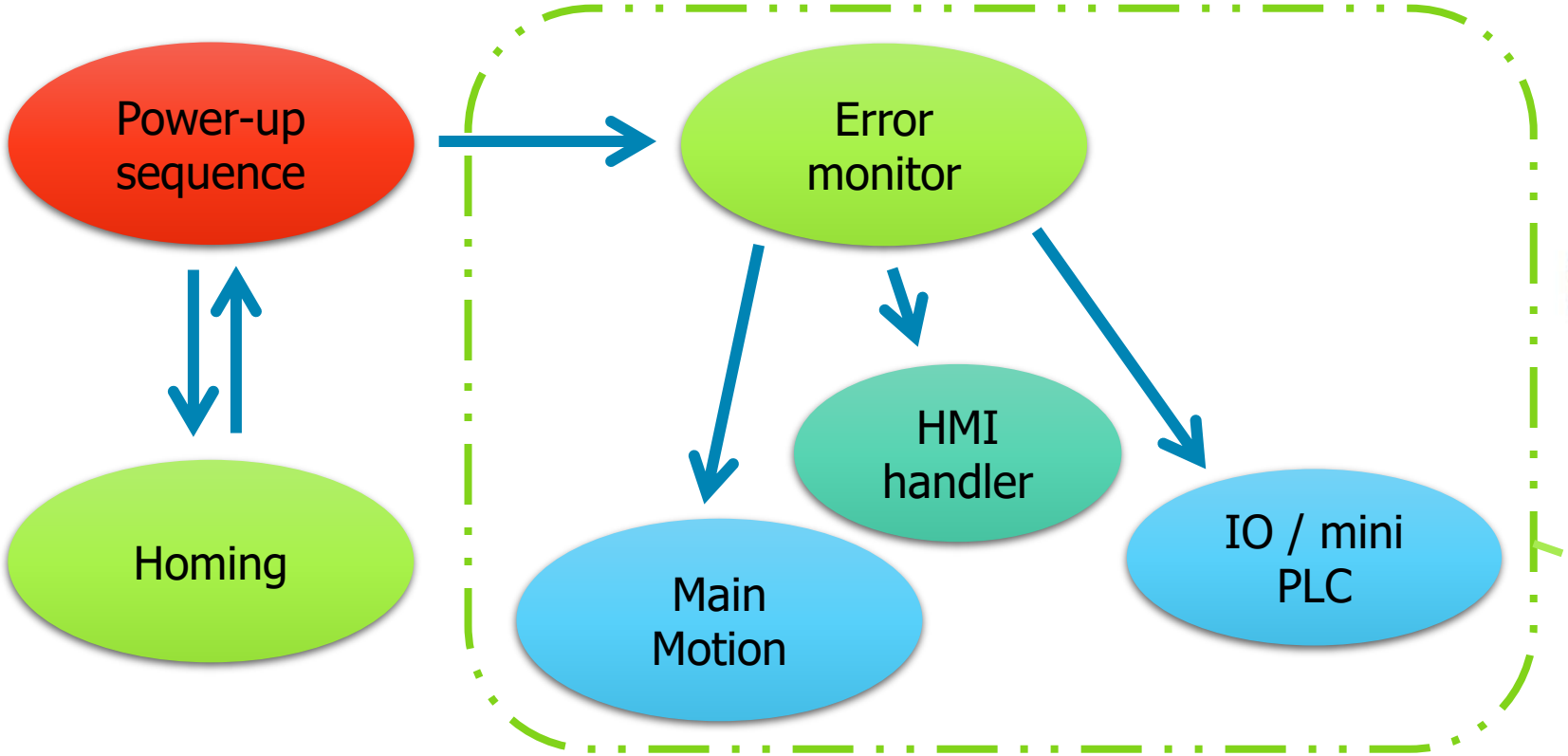




Why Multi-tasking?

Multiple functions

A machine has multiple functions, some operate sequentially and some concurrently.

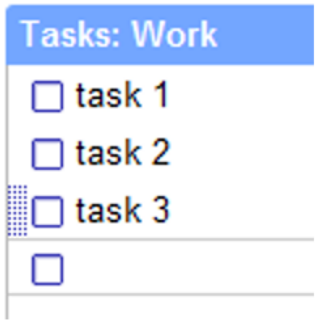


All running together!

Multi-Tasking: Benefit

Division of Tasks

Smaller, dedicated programs (processes) can be written to handle specific functions of the machine.

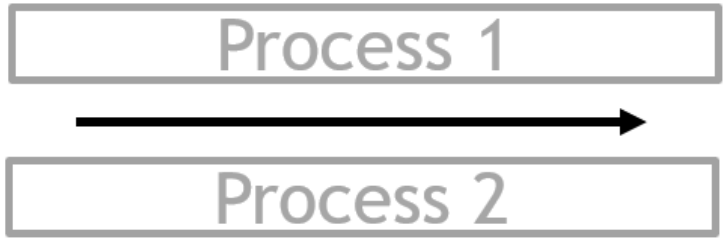


Error Handling

A process can be dedicated to error trapping.

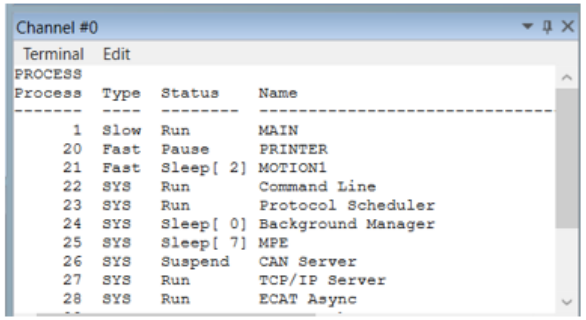
Concurrency

Each process acts like a virtual controller and operates independently of the others.



Command Line – Terminal #0

The command line / Motion Perfect Link is always available, even when programs are running.



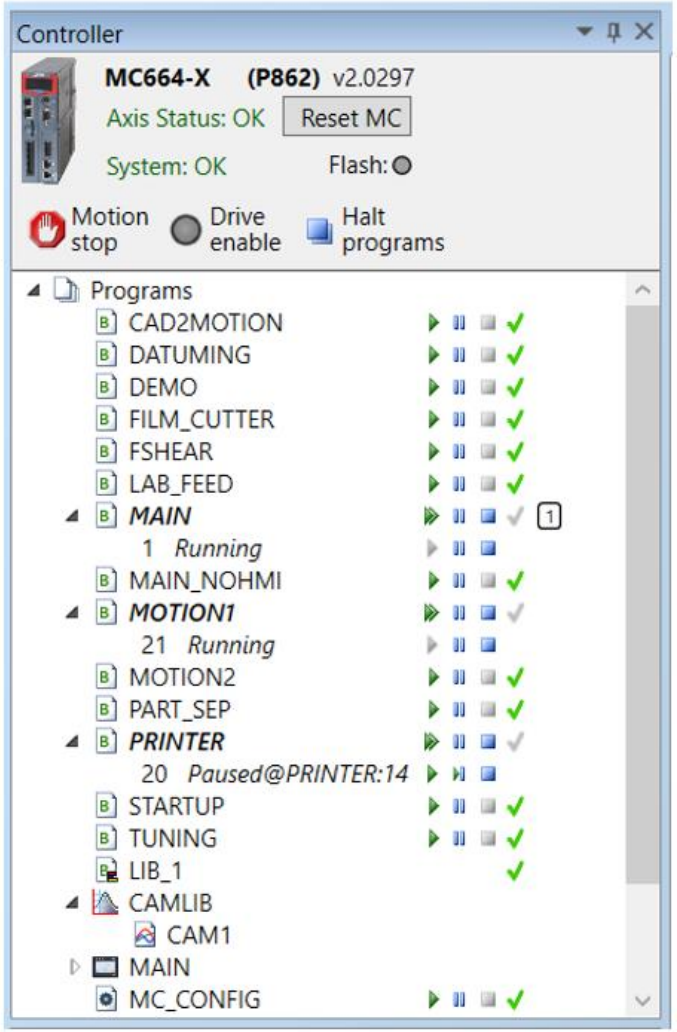
Processes

Process:
Processing slot that runs an IEC task or BASIC program

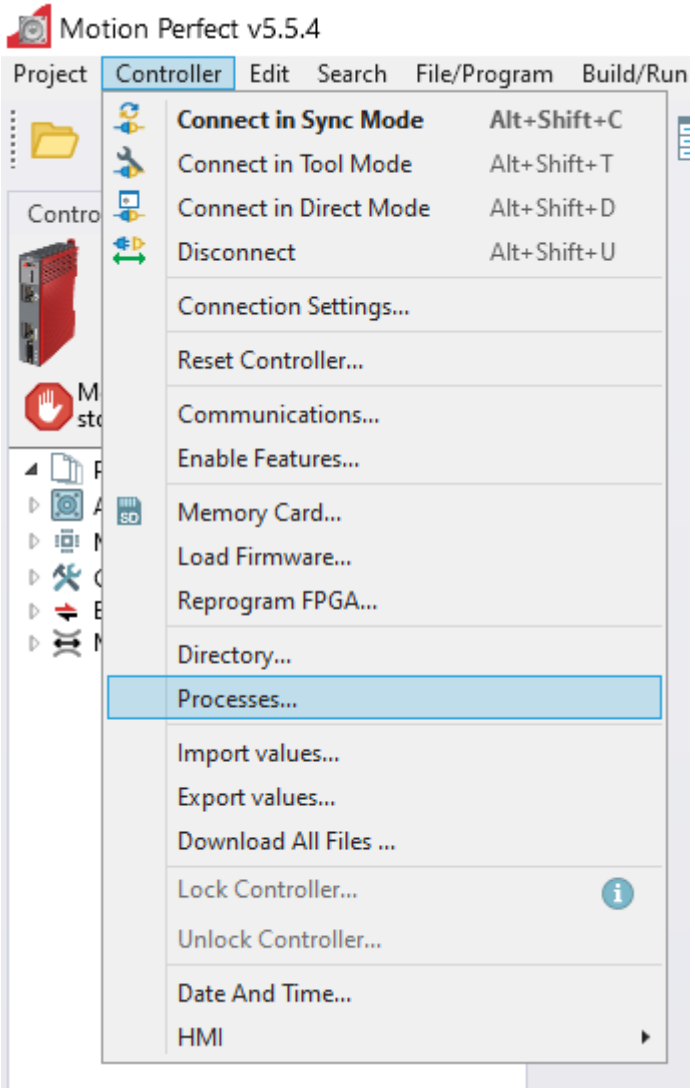
Each process is allocated a number between 0 and "max process"

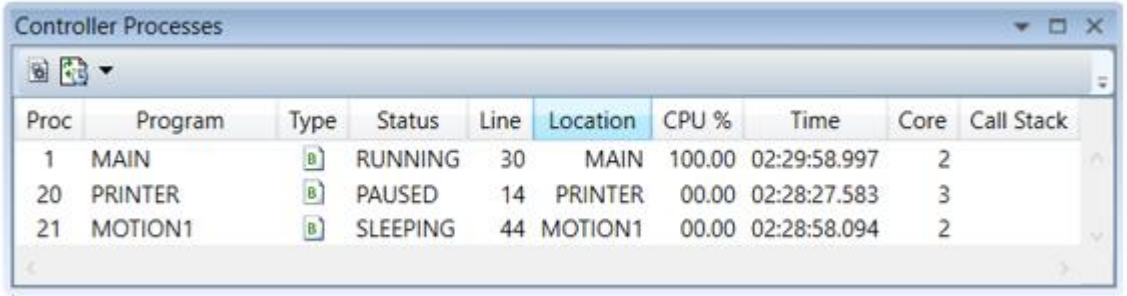
Motion Coordinator type determines the max number of processes

Controller	# Processes	
MC403 (Z)	6	0 .. 5
MC405 / Euro404	10	0 .. 9
MC508 / Euro408	22	0 .. 21
MC664(X)	22	0 .. 21
MC4N / MC6N	22	0 .. 9
Flex-6 Nano	22	0 .. 21
MotionPLC	4(12)	0 .. 3



Process Monitor

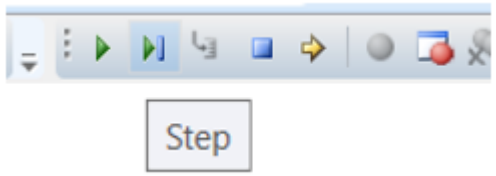




Proc	Program	Type	Status	Line	Location	CPU %	Time	Core	Call Stack
1	MAIN	B	RUNNING	30	MAIN	100.00	02:29:58.997	2	
20	PRINTER	B	PAUSED	14	PRINTER	00.00	02:28:27.583	3	
21	MOTION1	B	SLEEPING	44	MOTION1	00.00	02:28:58.094	2	

Running: running through the lines of BASIC at "full speed"
Paused: program has been paused by Motion Perfect.
Sleeping: program is in a wait state.

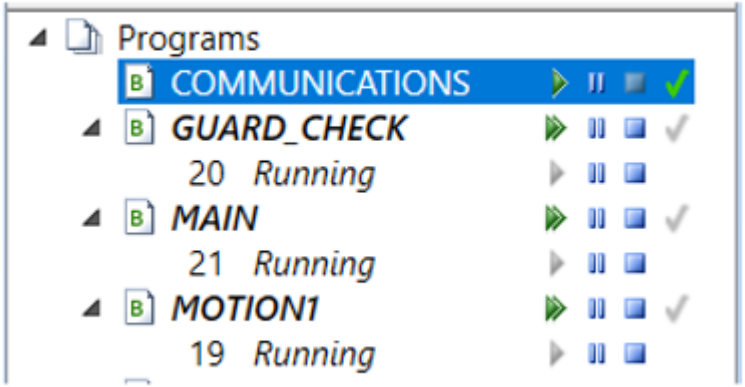
```
WA(time_in_msecs)
WAIT IDLE
WAIT LOADED
```

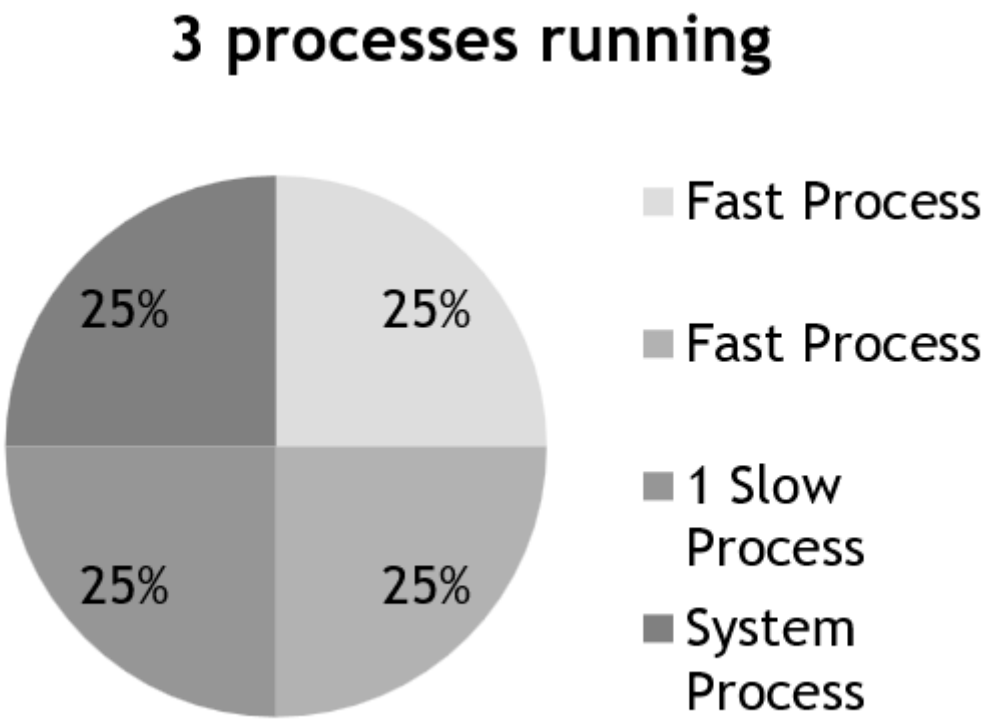


Multi-Tasking: Program priority

Process priority is defined by process number
The 2 highest process numbers are FAST
All others are SLOW

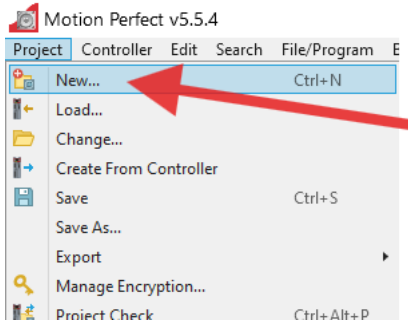
Example: MC664 (P861)

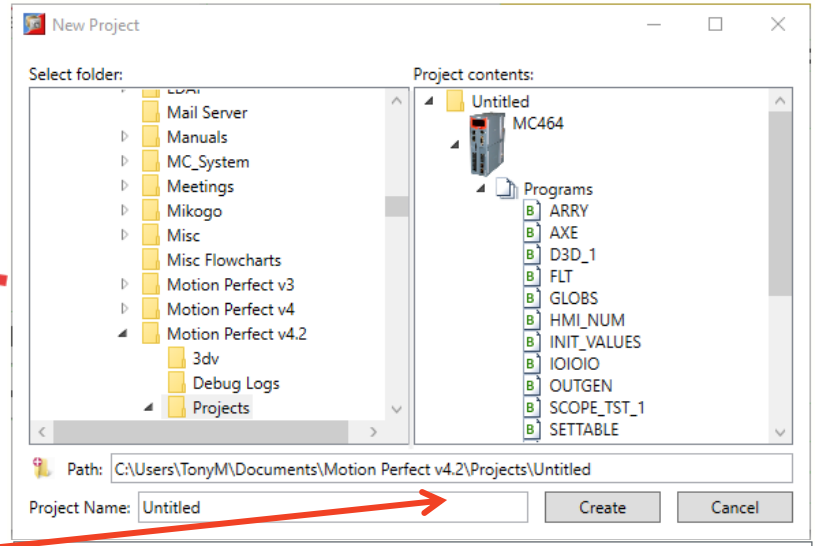




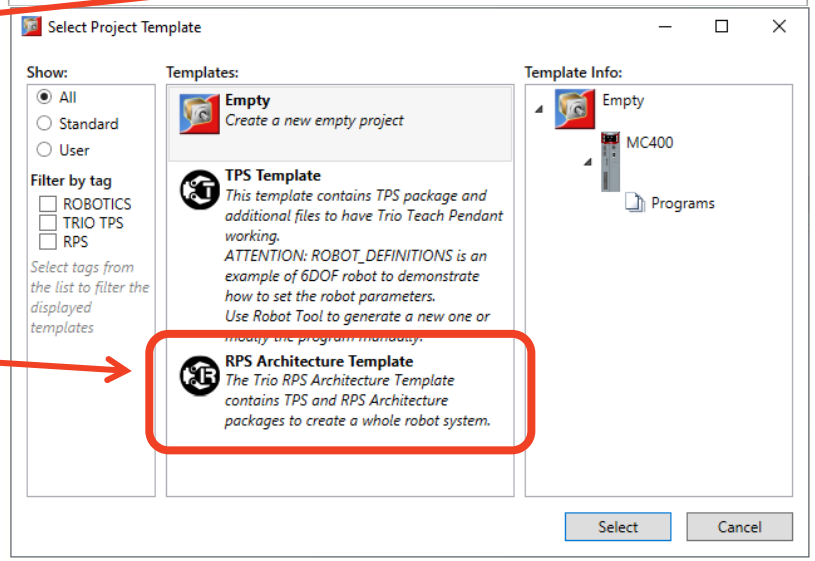
New Project

A new project can be created either as a new project (either empty or from a template) or from the contents of a connected controller.



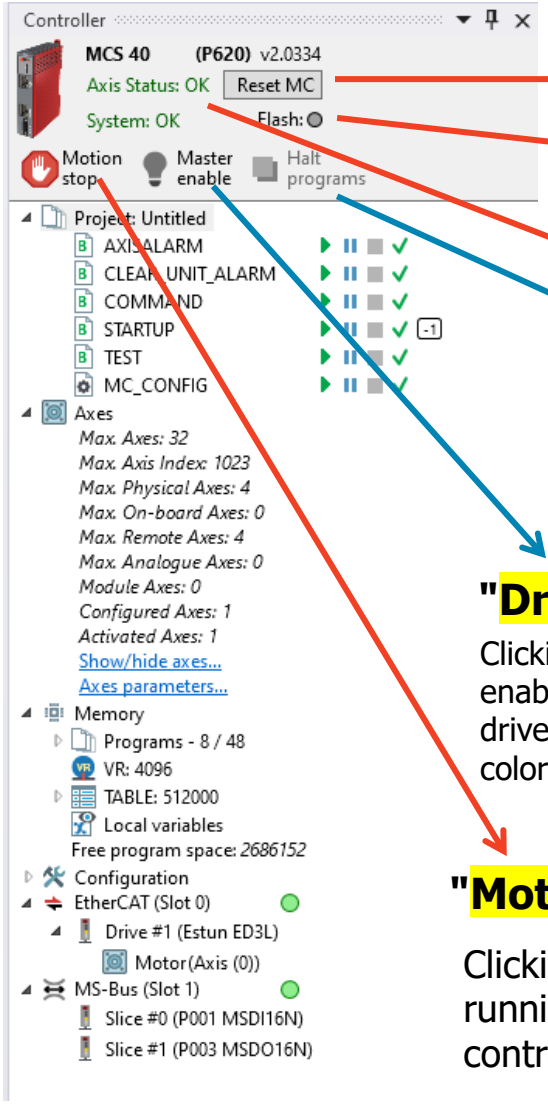


Select the directory in which to store the project, enter a name for the project then click on the "Create" button. The "Select Project Template" dialog is then displayed.

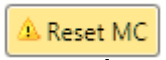


The RPS Template (for robotic features) just available in Motion Controller
The RPS features Not supported by Motion PLCs.

Project Tree



"Reset MC" Control

This control acts as a status indicator and a reset button. If the MC_CONFIG program on the controller has been created, changed or deleted since the controller was last reset, the control is highlighted, 

That means indicating that the controller needs to be reset before programs can be run again. Otherwise, the control appears as a normal button. When in either state, clicking on the control will reset the controller.

"Flash" Status Indicator

This indicates, by turning red, when the controller is writing to its "flash" EEPROM memory.

"Axis Status" Indicator
"System" Status Indicator

"Drives Enable" Button

Clicking on this button toggles the state of the drive enable (watchdog output) on the controller. When drives are enabled the background of the button is colored yellow.

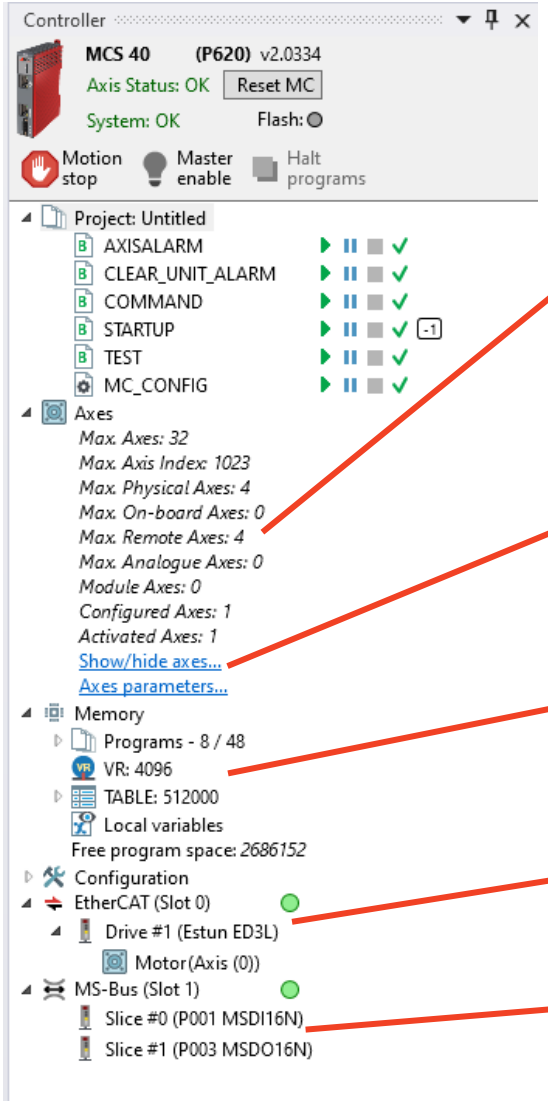
"Halt Programs" Button

Clicking on this button halts all currently running programs but does not stop current or buffered moves.

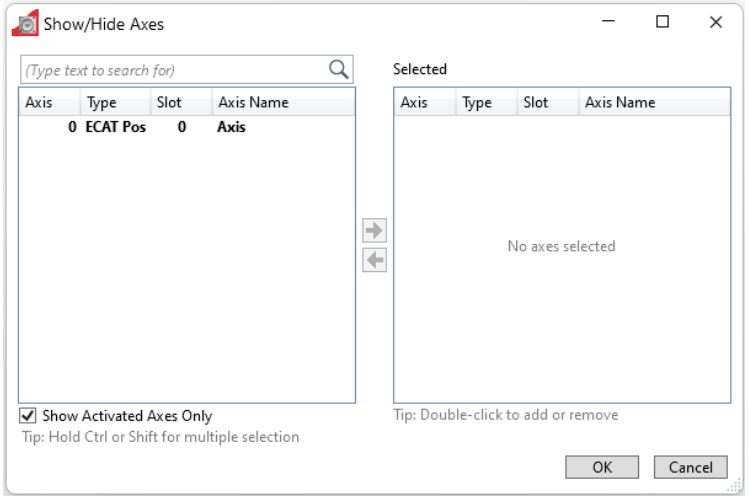
"Motion Stop" Button

Clicking on the "Motion Stop" button stops all currently running programs and empties all the move buffers on the controller causing all motion to stop.

Project Tree



Summary of AXES
(Configured and Virtual)



Main Unit Memory Capacity
(According to Hardware)

Connected Devices on EtherCAT network
(Slot 0)

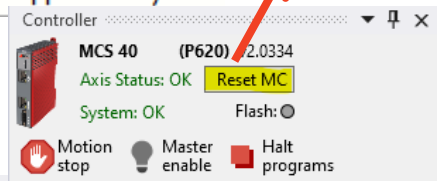
Connected IO Expansions Cards on MS-Bus
(Slot 1)

MC_CONFIG

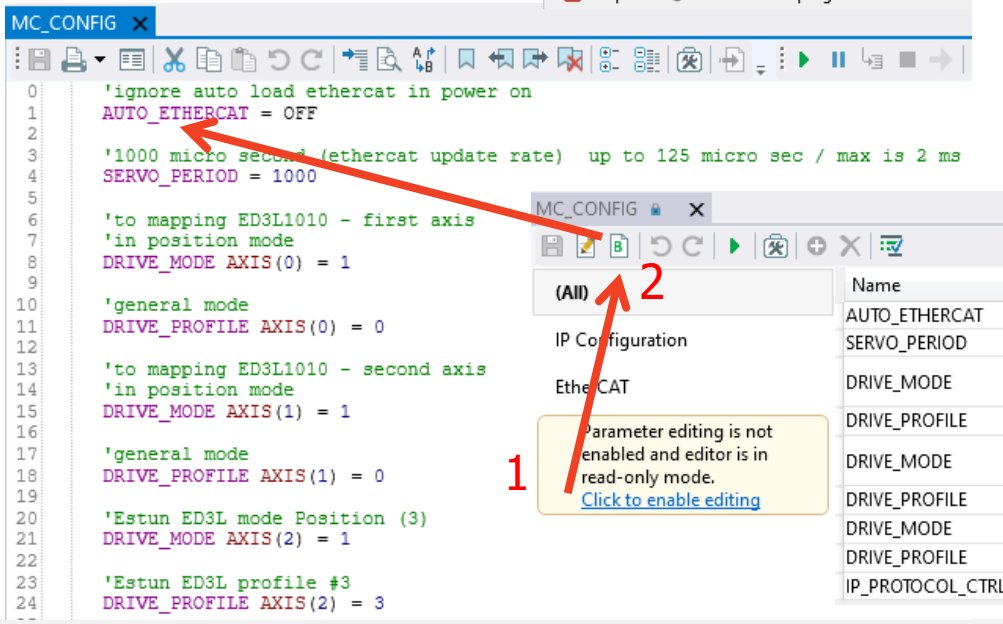
DRIVE_MODE

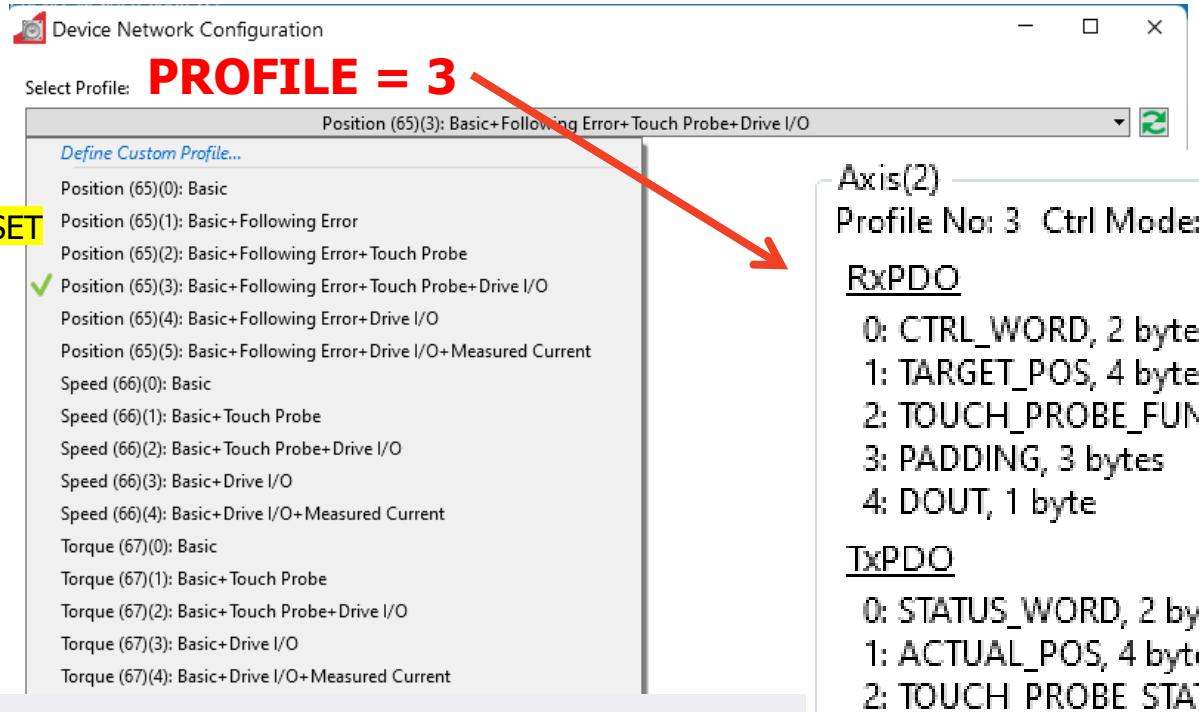
value:	1: Cyclic Synchronous Position mode (CSP)
	2: Cyclic Synchronous Velocity mode (CSV)
	3: Cyclic Synchronous Torque mode (CST)
	4: Open-loop speed (Inverter drive)
	5: Reference Encoder
	6: Open loop position (Stepper drive)

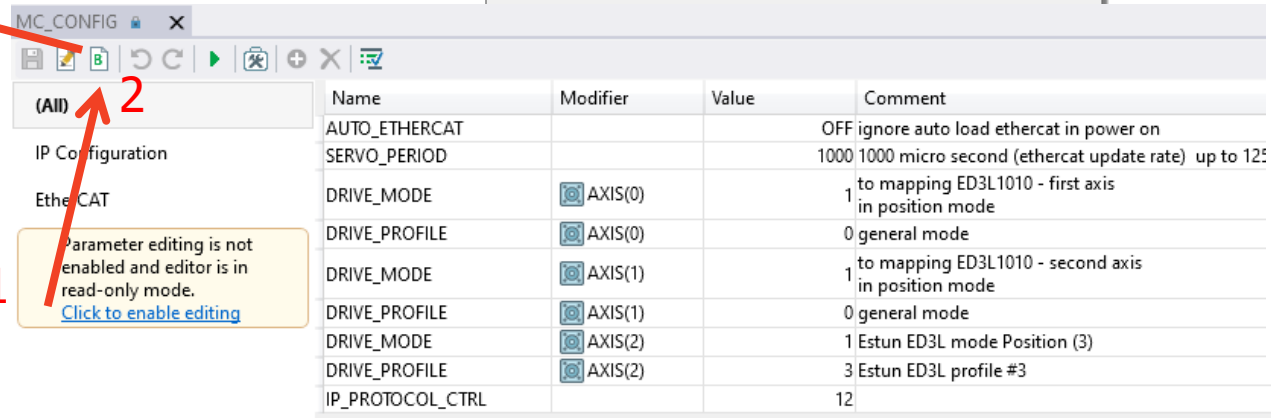
MUST BE RESET AFTER EDIT



DRIVE_PROFILE







Axis(2)

Profile No: 3 Ctrl Mode: Position (65)

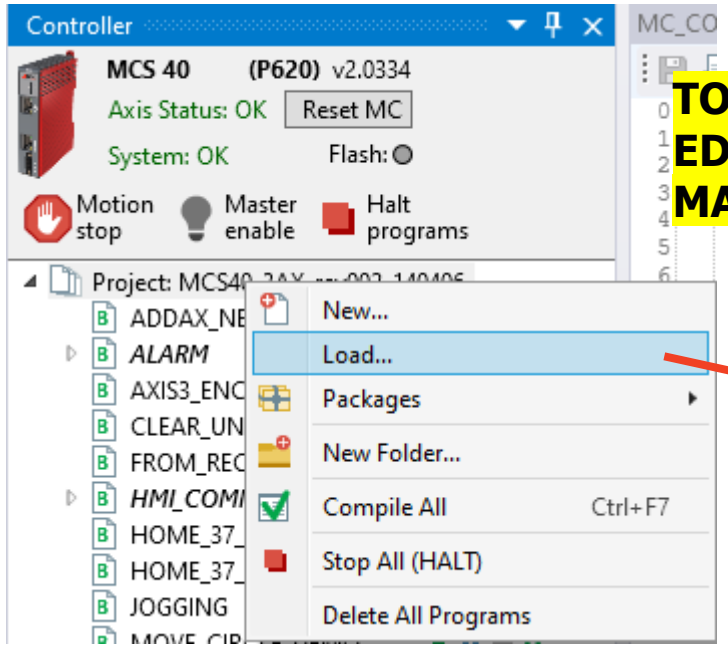
RxPDO

0: CTRL_WORD, 2 bytes
1: TARGET_POS, 4 bytes
2: TOUCH_PROBE_FUNCTION, 2 bytes
3: PADDING, 3 bytes
4: DOUT, 1 byte

TxPDO

0: STATUS_WORD, 2 bytes
1: ACTUAL_POS, 4 bytes
2: TOUCH_PROBE_STATUS, 2 bytes
3: TOUCH_PROBE1_POS1, 4 bytes
4: TOUCH_PROBE1_POS2, 4 bytes
5: TOUCH_PROBE2_POS1, 4 bytes
6: TOUCH_PROBE2_POS2, 4 bytes
7: ACTUAL_FE, 4 bytes
8: PADDING, 2 bytes
9: DIN, 1 byte
10: PADDING, 1 byte

EC_EXTEND



TO GET ED3L-1010AEA EtherCAT MAPPING FILE

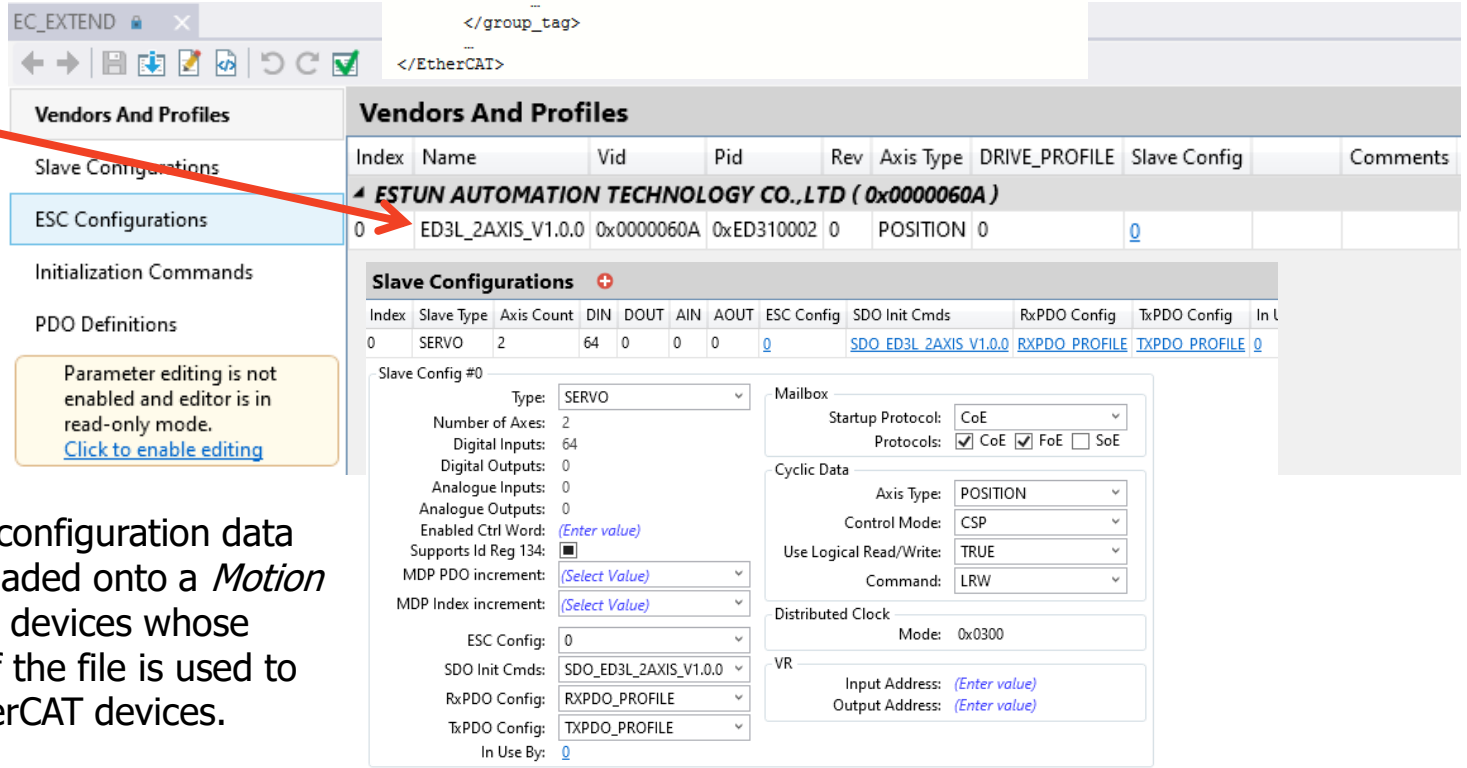
GET FILE

File structure

The XML format has tags to delineate the various objects that require description.

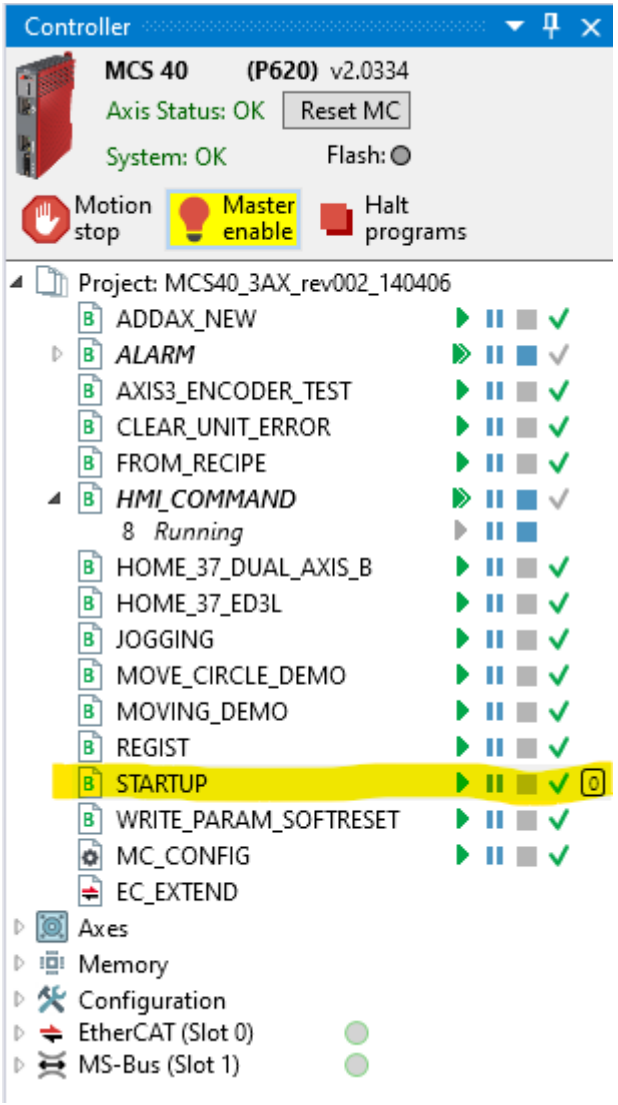
The Tags' hierarchy is as shown below:

```
<EtherCAT>
        <group_tag>
                <block_tag>
                        <Entry_tag>
                                <Attribute_tag>Value</Attribute_tag>
                                …
                        </Entry_tag>
                        …
                </block_tag>
                …
        </group_tag>
        …
</EtherCAT>
```



The EC_EXTEND file is an XML format text file containing configuration data for one or more EtherCAT slave devices. The file can be loaded onto a *Motion* Coordinator so that it is able to interact with the EtherCAT devices whose configuration data in contained in the file. The contents of the file is used to extend the *Motion* Coordinator's in-build dictionary of EtherCAT devices.

AXIS INITIALIZE




```
FOR i = 0 TO 2
    BASE(i)
    'to reverse motor direction (CCW)
    ENCODER_RATIO(-1,1) 'This command allows the incoming encoder count to be scaled by a non integer ratio
    STEP_RATIO(-1,1) 'This command sets up an integer ratio for the axis' demanded positon

    UNITS = 2 ^ 20
    SPEED = 10 '600RPM
    JOGSPEED = 5 '300RPM
    ACCEL = 100
    DECEL = ACCEL
    'CREEP = 0.1
    'REP_DIST = 360
    'REP_OPTION = 0
    DRIVE_FE_LIMIT = 100 '100turn - per revolution
    FE_LIMIT = 100 'the maximum following error
    FE_RANGE = 70 ' following error report range - shows in status
    FS_LIMIT = 400000000000
    RS_LIMIT = -400000000000
    AXIS_ENABLE = ON ' enable the axis in controller
    SERVO AXIS(i) = ON 'run the axis in controller
    'DEFPOS(0) ' get zero value
NEXT i
```

Axis Parameters

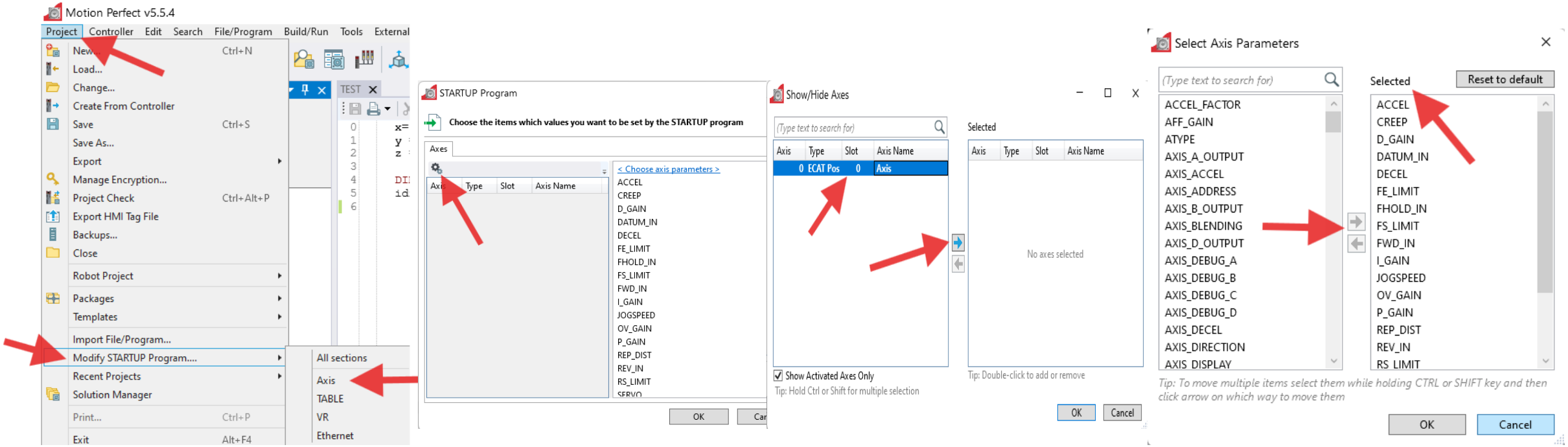
(Type text to search for)

Parameter	Axis (0)	Axis (1)	Axis (2)
ATYPE	ECAT Pos	ECAT Pos	ECAT Pos
UNITS	1048576	1048576	1048576
Velocity profile			
ACCEL	100	100	100
DECEL	100	100	100
SPEED	10	10	10
CREEP	9.53674e-05	9.53674e-05	9.53674e-05
JOGSPEED	5	5	5
MERGE	0	0	0
SRAMP	0	0	0
MSPEED	0	0.00095	0
VP_SPEED	0	0	0
AXIS_SPEED	1000	1000	1000
Limits			
DATUM_IN	-1	-1	-1
DRIVE_FE_LIMIT	100	100	100
FE_LIMIT	100	100	100
FE_RANGE	70	70	70
FHOLD_IN	-1	-1	-1
FWD_IN	-1	-1	-1
REP_DIST	190734.86328	190734.86328	190734.86328
REP_OPTION	0	0	0
REV_IN	-1	-1	-1
FS_LIMIT	400000000000	400000000000	400000000000
RS_LIMIT	-400000000000	-400000000000	-400000000000
Positions			
DPOS	9.25064e-05	-0.00074	-0.00235
ENCODER	99	-780	-2465
ENDMOVE	9.25064e-05	-0.00074	-0.00235

Axis parameters need to be mapped only once at controller startup (MC power on).

Modify STARTUP Program

The STARTUP program is a user run TrioBASIC program used to initialize the system on power-up. It is commonly used to set up Axis Parameters, TABLE Areas, VR Variables, some Communications Parameters and Drive Parameters (when intelligent drive support is available). Adding Drive Parameters is handled by the appropriate drive configuration tool, all other data is handled by a common dialog set.
When "Modify STARTUP Program…" is selected from the "Project" item in the main menu the user is presented with a list of things which can be added to the STARTUP program.



ED3L Alarm Handler

AXISSTATUS

Bit	Description	Value	char
0	Speed limit active	1	l
1	Following error warning range	2	w
2	Communications error to remote drive	4	a
3	Remote drive error	8	m
29	MOVEADD is being cancelled (CANCEL(5))	536870912	
30	EtherCAT emergency message received from remote drive	1073741824	
31	Not used	2147483648	

ED3L Error Code Address

Error Code (603Fh)

Index	Subindex	Name	Data Type	Access	PDO Mapping	Value
603Fh	0	Error Code	UINT16	RO	TxPDO	0x0000

```
IF AXISSTATUS AXIS(2).30=1 OR AXISSTATUS AXIS(2).3=1 THEN
    VR(2000)=0 ' check if there is any remote Drive Error then read the relevant ADDRESS.
    CO_READ(0,2,$603f,0,6,1000) ' Read the Error Code from Object $603F subindex0 of  - in HEX format
    'ED3M drive's (2) Axis.
    VR(1001)= VR(1000) AND $00FF 'separate the low byte to read the Alarm Code and
    'convert it TO decimal TO read in VR(1001).
    WA(50)
    TABLE(90, VR(1001)) 'table 90 to get single axis alarm on HMI in word
```

IF TABLE(90) = 80 ‘decimal
It means A.50 ‘hex

ED3L PROFILE (GET/SET)

According to the Drive Profile,
Using include mapped properties!

MOVE_CIRCLE_DEMO

```
BASE(0,1) 'select X and Y for circular interpolati
SPEED AXIS(0) = 5
SPEED AXIS(1) = 5

WA(2000)
MOVECIRC(10,10,-10,10,1)
WAIT IDLE

MOVE(0, 12) 'Move A -> B
MOVECIRC(3, 3, 3, 0, 1) 'Move B -> C
MOVE(4, 0) 'Move C -> D
MOVECIRC(3, -3, 0, -3, 1) 'Move D -> E
MOVE(0, -12) 'Move E -> F
MOVECIRC(-3, -3, -3, 0, 1) 'Move F -> G
MOVE(-4, 0) 'Move G -> H
MOVECIRC(-3, 3, 0, 3, 1) 'Move H -> A
WAIT IDLE

SPEED AXIS(0) = 10
SPEED AXIS(1) = 10
DRIVE_C
```

DRIVE_CLEAR
DRIVE_CONTROL
DRIVE_CONTROLWORD
DRIVE_CURRENT
DRIVE_CW_MODE
drive_c

Axis Parameters

Parameter	Axis (2)
SYNC_PAUSE	0
SYNC_WITHDRAW	0
SYNC_TIMER	0
Drive	
AXIS_ADDRESS	-1
AXIS_ERROR_COUNT	0
DRIVE_CONTROL	0
DRIVE_CONTROLWORD	0
DRIVE_CW_MODE	0
DRIVE_ENABLE	0
DRIVE_INDEX	0
DRIVE_MODE	1
DRIVE_PARAMETER	0
DRIVE_PROFILE	3
DRIVE_SET_VAL	0
DRIVE_TYPE	0
DRIVE_VALUE	0
DRIVE_CURRENT	0
DRIVE_INPUTS	31
DRIVE_MONITOR	0
DRIVE_STATUS	592
DRIVE_TORQUE	0
DRIVE_VELOCITY	0

Intelligent drives

Slot 0 - EtherCAT | Slot 1 - MS-Bus

Diagram
Master state: Operational — Diagnostics Disabled
Address: 1 2

Details
Drives

Axis	Ctrl Mode	Model	Pos	Alias
0	ECAT Pos		0	0
1	ECAT Pos		0	0
2	ECAT Pos	Estun ED3L	1	0

Modify STARTUP Program | Browse database...

EtherCAT Profile Display
Model: Mentor MP (0x00050007)
Vendor: Control Techniques (0x000000F9)
Ctrl Mode: Position (65) Profile No 0

Display Profile
Position (65) #0 #1 #2
Speed (66) #0 #1 #2
Torque (67) #0 #1 #2

PDO Config
RxPDO
0: Status Word, 2 bytes
1: Actual Position, 4 bytes
TxPDO
0: Control Word, 2 bytes
1: Target Position, 4 bytes

SDO Config
CoE written on Pre-Op to Safe-Op
0x1C12:00 (1) = 0x00
0x1C13:00 (1) = 0x00
0x1600:00 (1) = 0x00
0x1600:01 (4) = 0x60400010
0x1600:02 (4) = 0x607A0020
0x1600:00 (1) = 0x02
0x1A00:00 (1) = 0x00
0x1A00:01 (4) = 0x60410010
0x1A00:02 (4) = 0x60640020
0x1A00:00 (1) = 0x02
0x1C12:01 (2) = 0x1600

Add to MC_CONFIG... | Close

Motion 1

BASE(n)

The axis group consists of:
Axes n, n+1, n+2 etc.

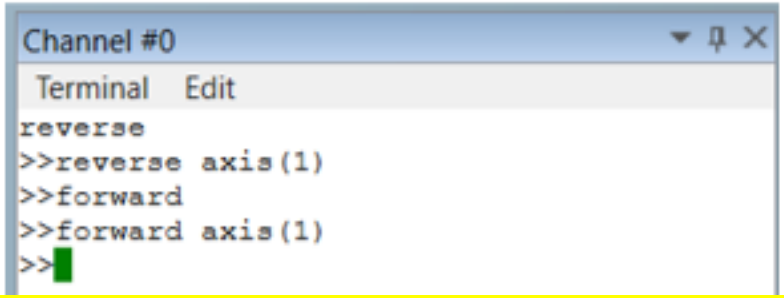
Example on MC405 set:
BASE(2)
Axis group is 2, 3, 4, 5, 6, 7 etc.

BASE(x,y,z)

This specifies the axis group as:
Axes x, y, and z

Example on MC508 set:
BASE(1,2,5)
Axis group is 1, 2, 5, 0, 3, 4, 6, 7 etc.

Each Process can specify a different axis group.

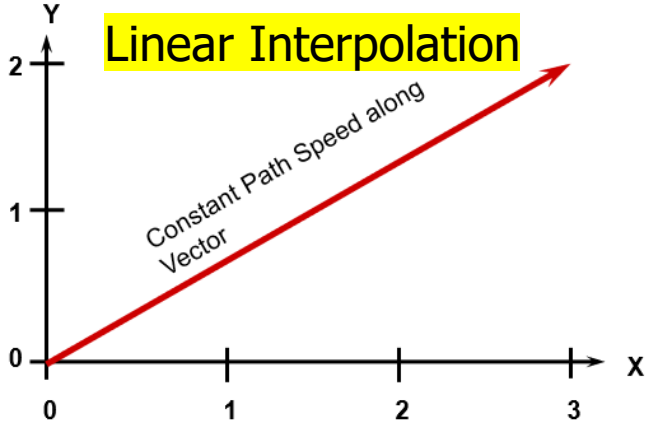


Type commands into terminal channel #0

'MOTION1' Main routine

```
FOR x=0 TO 1
   BASE(x)
   SPEED=200
   ACCEL=500
   DECEL=500
NEXT x
```

Linear Interpolation



MOVE(3,2)

Single axis continuous moves:

- Starts movement with the programmed speeds and accelerations
- Multiple axis can be run simultaneously

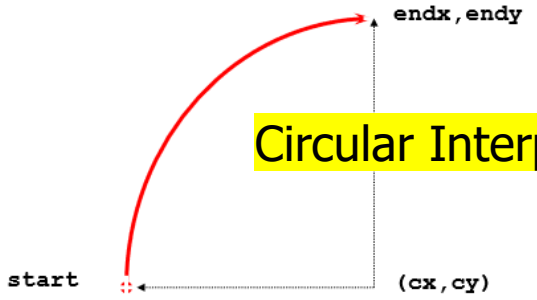
FORWARD

- Move the base axis in a forward direction

REVERSE

- Move the base axis in a reverse direction

Circular Interpolation



MOVECIRC(endx,endy,cx,cy,dir)

NOTE: Endpoint & Centre values are RELATIVE to start position.

BASIC Programming language

WHILE

```
WHILE IN(8) = 1
  OP(10,ON) ' output 10 ON
  WA(1000) ' wait 1000ms
  OP(10,OFF) ' output 10 OFF
  WA(1000) ' wait 1000ms
WEND
```

Create a slowly flashing output

Infinity WHILE

The main loop for a machine program

```
WHILE TRUE

  .. Main program code
  ..

WEND
```

REPEAT

```
REPEAT
  OP(10,ON)
  WA(1000)
  OP(10,OFF)
  WA(1000)
UNTIL IN(8) = ON
```

The same flashing output - will always flash at least once.

Infinity REPEAT

Infinite loop

```
REPEAT

  .. Code in continuously
  .. repeating loop

UNTIL FALSE
```

FOR

```
FOR a = 8 TO 15
    OP(a,ON)
    WA(1000)
    OP(a,OFF)
NEXT a
```

FOR & STEP

```
FOR a = 14 TO 8 STEP -2
    OP(a,ON)
    WA(1000)
    OP(a,OFF)
NEXT a
```

Counting down

FOR & Condition

```
FOR x = 1 TO 2047
  IF x > AIN(0) THEN
    VR(10) = x
    EXIT

  ELSEIF IN(1) = ON THEN
    CONTINUE

  ENDIF
  a = 2 * x
  AOUT(2) = a - 512
NEXT x
```

IF

Multiple Actions

```
IF IN(8) = ON THEN
  OP(10, ON)
  x = 10.5
  distance = SPEED ^ 2 / (2 * DECEL)
ENDIF
```

BASIC Programming language

SWITCH CASE

```
Var_1 = 45

SELECT_CASE var_1
  CASE 1
    OP(8, ON)
  CASE 2, 3, 4
    OP(9,ON)
  CASE 30, 50
    OP(10, ON)
    OP(15, ON)
  CASE 45
    OP(11, ON)
  CASE ELSE
    OP(8, 15, 0)
END_CASE
```

GOTO

```
x = 1
y = 2
z = x + y
GOTO label1

a = x + 2
b = y * z
label1:
a = x + 10
b = y / z
```

```
  GOSUB sub1
  z = x * y
STOP


sub1:
  x = 2
  y = y + 3
RETURN
```

FUNCTION

```
FUNCTION set_axis(v, axis_number AS INTEGER)
  SPEED AXIS(axis_number) = v
  ACCEL AXIS(axis_number) = v * 10
  DECEL AXIS(axis_number) = v * 10
ENDFUNC
```

Functions must be placed in a BASIC Library.

WHILE .. WEND


```
WHILE IN(8) = 1
  OP(10,ON) ' output 10 ON
  WA(1000) ' wait 1000ms
  OP(10,OFF) ' output 10 OFF
  WA(1000) ' wait 1000ms
WEND
```

Create a slowly flashing output

Syntax:

```
WHILE condition
  Commands
WEND
```

Parameters:

condition:	Any valid logical TrioBASIC expression
commands:	TrioBASIC statements that you wish to execute

The main loop for a machine program

```
WHILE TRUE

  .. Main program code
  ..

WEND
```

REPEAT .. UNTIL


```
REPEAT
   OP(10,ON)
   WA(1000)
   OP(10,OFF)
   WA(1000)
UNTIL IN(8) = ON
```

The same flashing output -
will always flash at least once.

Infinite loop

```
REPEAT

   .. Code in continuously
   .. repeating loop

UNTIL FALSE
```

Syntax:

```
REPEAT
  commands
UNTIL expression
```

Parameters:

expression:	Any valid TrioBASIC expression
commands:	TrioBASIC statements that you wish to execute

FOR .. NEXT .. LOOPS


```
FOR a = 8 TO 15
    OP(a,ON)
    WA(1000)
    OP(a,OFF)
NEXT a
```

Flashes all 8 outputs in sequence

```
FOR a = 8 TO 14 STEP 2
    OP(a,ON)
    WA(1000)
    OP(a,OFF)
NEXT a
```

Flashes only even numbered outputs

```
FOR a = 14 TO 8 STEP -2
    OP(a,ON)
    WA(1000)
    OP(a,OFF)
NEXT a
```

Counting down

Syntax:

```
FOR variable = start TO end [STEP increment]
  commands
NEXT variable
```

Parameters:

commands:	Trio BASIC statements that you wish to execute
variable:	A valid Trio BASIC variable. Either a global VR variable, or a local variable may be used.
start:	The initial value for the variable
end:	The final value for the variable
increment:	The value that the variable is incremented by, which may be positive or negative

EXIT and CONTINUE


```
FOR x = 1 TO 2047
  IF x > AIN(0) THEN
    VR(10) = x
    EXIT _ LOOP

  ELSEIF IN(1) = ON THEN
    CONTINUE

  ENDIF
  a = 2 * x
  AOUT(2) = a - 512
NEXT x
```

Syntax:

```
EXIT_LOOP
```

Syntax:

```
CONTINUE
```

FOR .. NEXT loop must normally complete to the last value of x.

To come out of the FOR .. NEXT loop early, use EXIT.

The program then continues on the line after NEXT x.

CONTINUE forces the program to go back to the top of the loop.

AOUT(2) is then not set on that pass round the loop.

IF .. THEN .. ENDIF

Single Action

```
IF IN(8) = ON THEN OP(10, ON)
```

Multiple Actions

```
IF IN(8) = ON THEN
  OP(10, ON)
  x = 10.5
  distance = SPEED ^ 2 / (2 * DECEL)
ENDIF
```

This form is clearer to read

```
IF IN(8) = ON THEN
  OP(10, ON)
ENDIF
```

Syntax:

```
IF condition THEN
  commands
ELSEIF expression THEN
  commands
ELSE
  commands
ENDIF
```

Parameters:

condition:	Any valid logical TrioBASIC expression
commands:	TrioBASIC statements that you wish to execute

IF .. ELSE .. ELSEIF


```
IF IN(8)= ON THEN
  OP(10,ON)
  OP(11,ON)
ELSE
  OP(10,OFF)
  OP(11,ON)
ENDIF
```

```
IF IN(8)= ON THEN
  OP(10,ON)
ELSEIF IN(9)=ON THEN
  stop_all()
ELSEIF data = 22 THEN
  value_test(1, 30)
ELSE
  PRINT #5, "Not found"
ENDIF
```

Use an IF structure to report the operating state of a machine.

```
IF operating_state = 0 THEN
  PRINT#5, "Machine Running"
ELSEIF operating_state = 1 THEN
  PRINT#5, "Machine Idle"
ELSEIF operating_state = 2 THEN
  PRINT#5, "Machine Jammed"
ELSE
  PRINT#5, "Machine in unknown state"
ENDIF
```

SELECT .. CASE


```
Var_1 = 45

SELECT_CASE var_1
  CASE 1
    OP(8, ON)
  CASE 2, 3, 4
    OP(9,ON)
  CASE 30, 50
    OP(10, ON)
    OP(15, ON)
  CASE 45
    OP(11, ON)
  CASE ELSE
    OP(8, 15, 0)
END_CASE
```

Syntax:

```
SELECT "program"[, program_type]
```

1) SELECT_CASE checks <u>one</u> variable.

2) Each CASE that numerically equals the value causes execution of code between 2 CASE statements.

3) The program then continues after END_CASE.

4) This form is ideal for several actions dependant on the value of a variable.

FUNCTIONS


```
FUNCTION set_axis(v, axis_number AS INTEGER)
  SPEED AXIS(axis_number) = v
  ACCEL AXIS(axis_number) = v * 10
  DECEL AXIS(axis_number) = v * 10
ENDFUNC
```

Functions must be placed in a BASIC Library.

A Function can have values passed to it.

In the calling program:

```
set_axis(25, 0)
set_axis(30, 1)
```

A Function can return a value

The returned value can be a single value or an array containing multiple values.

Syntax:

```
FUNCTION name([param1 AS type[, param2 AS type[, param3 AS type .... ]]])
....
RETURN x
ENDFUNC
```

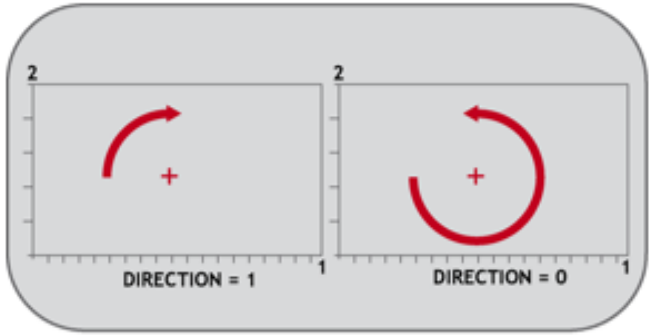
Parameters:

param1:	First optional parameter to be passed into the function
param2:	The second parameter if required
param3:	The third parameter if required
paramN:	Up to 16 parameters may be passed

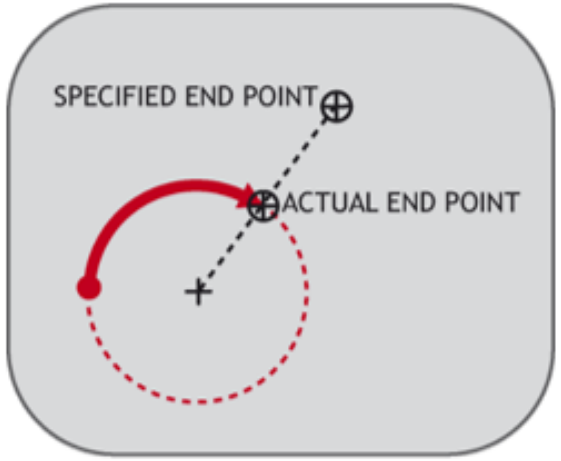
Circular interpolated moves

MOVECIRC(*end1*, *end2*,*centre1*, *centre2*, ***direction***)

1 .. CW
0 .. CCW



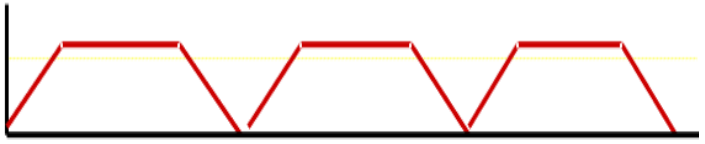
If the end point is not on the circular arc, the arc will end at the angle specified between the centre and the endpoint



Contoured Motion-MERGE Command

Merged moves:

- Enables moves to be joined without having to ramp speed down to zero
- The programmer is responsible for sensible merges
  - Ex: do not merge moves that reverse direction



MERGE = OFF

MERGE = *state*

- Set MERGE to be either ON or OFF

MERGE = ON

```
merge_move:
  BASE(0,1) 'set base array
  MERGE=ON 'set MERGE state
  MOVEABS(0,50) 'run a sequence of moves
  MOVE(0,100)
  MOVECIRC(50,50,50,0,1)
  MOVE(100,0)
  MOVECIRC(50,-50,0,-50,1)
  MOVE(0,-100)
  MOVECIRC(-50,-50,-50,0,1)
  MOVE(-100,0)
  MOVECIRC(-50,50,0,50,1)
  WAIT IDLE
  MERGE=OFF
RETURN
```

AXIS ENABLE & Servo State & WDOG

AXIS_ENABLE

Can be used to independently disable an axis. ON by default,
can be set to OFF to disable the axis.
The axis is enabled if AXIS_ENABLE = ON and WDOG = ON .
On stepper axis AXIS_ENABLE will turn on the hardware enable outputs.

```
AXIS_ENABLE = ON  'Enable the axis
```

SERVO_ENABLE

Enable axis 1 to run under closed loop control and axis 1 as open loop.

```
SERVO AXIS(0)=ON   'Axis 0 is under servo control-Closed loop
SERVO AXIS(1)=OFF  'Axis 1 is run Open loop
```

WDOG

Controls the WDOG relay contact used for enabling external drives.
The WDOG= ON command MUST be issued in a program prior to executing moves.

Lookahead

LIMIT_BUFFERED:

- Increases buffered moves from 2 (MTYPE and NTYPE) to up to 64
- Enables many small moves to be MERGED together
- Increases path speed
- Useful when generating move profiles from CAD data

LIMIT_BUFFERED = value

MOVES_BUFFERED:

- Parameter to read the number of buffered motion commands

Channel #0

Terminal Edit Search

```
>>?buffersize
64
>>
```

MCS40 Buffer Size

In AXIS PARAMETERS

Axis Parameters

Parameter	Axis (0)	Axis
Control		
LOOKAHEAD_FACTOR	1	
Drive		
DRIVE_ENABLE	0	

This sets the maximum number of moves that the controller can buffer.

You can increase the machine speed when using MERGE or CORNER_MODE by increasing the number of moves buffered.

Each controller has its own maximum buffer size, indicated by BUFFERSIZE .

Running and Stopping programs

RUN "*program_name*" [, process]

- Runs a program on a particular process
- The process can be left off and it will run on the highest process number available
- Program must exist else an error occurs at runtime

```
IF TABLE(27) = 1 AND PROC_STATUS PROC(2) = 0 THEN
 RUN "REGIST",2
```

STOP

- Stops the current program

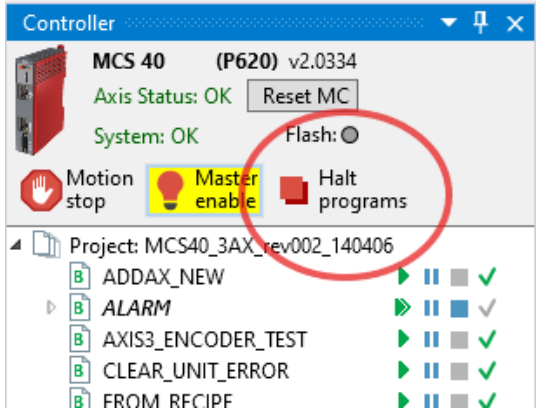
```
'### This Example will set some esse
'for Axis#2 and perform a Soft Reset
 STOP "ALARM"
 STOP "HMI_COMMAND"
```

STOP "*program_name*" [, process]

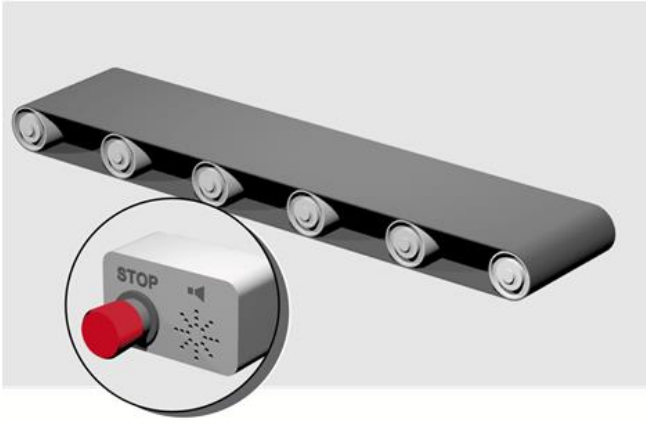
- Stops a specified program
- The process can be left off and it will stop all instances of the program
- Program must exist else an error occurs at runtime

HALT

- Stops all programs



Stop Motion methods




```
CONNECT(1, 0) AXIS(1) 'Axis 1 follows axis 0
BASE(0)
REPAEAT
  MOVE(1000) AXIS (0)
  MOVE(-100000) AXIS (0)
  MOVE(100000) AXIS (0)
UNTIL IN (2) = OFF 'Stop button pressed?
RAPIDSTOP(2)
```

Motion can be stopped either across all axis or on a single axis

CANCEL

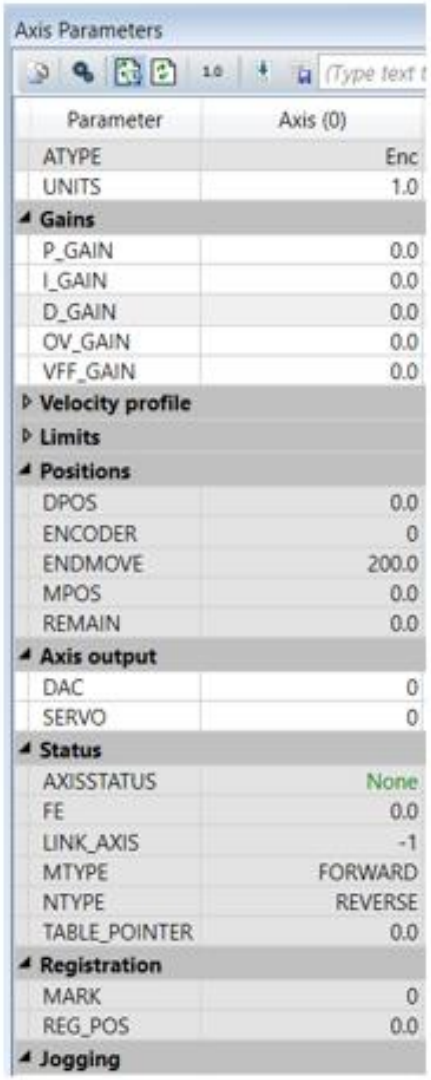
- Stops the running move on the base axis

RAPIDSTOP

- Stops the motion on all axis

CANCEL(2) and RAPIDSTOP(2)

- Stops the motion, clears buffered moves and the PMOVE



Axis Parameters

Parameter	Axis (0)
ATYPE	Enc
UNITS	1.0
Gains	
P_GAIN	0.0
I_GAIN	0.0
D_GAIN	0.0
OV_GAIN	0.0
VFF_GAIN	0.0
Velocity profile	
Limits	
Positions	
DPOS	0.0
ENCODER	0
ENDMOVE	200.0
MPOS	0.0
REMAIN	0.0
Axis output	
DAC	0
SERVO	0
Status	
AXISSTATUS	None
FE	0.0
LINK_AXIS	-1
MTYPE	FORWARD
NTYPE	REVERSE
TABLE_POINTER	0.0
Registration	
MARK	0
REG_POS	0.0
Jogging	

Printing

You can print information to terminal windows in Motion perfect, the serial ports or to a file.

PRINT "*text*"

- Prints text to channel 0 on the coordinator

PRINT#*channel*, "*text*"

- Prints text to specified channel
- Channel 5 is a terminal window in Motion Perfect
- Other channels are shown in the manual

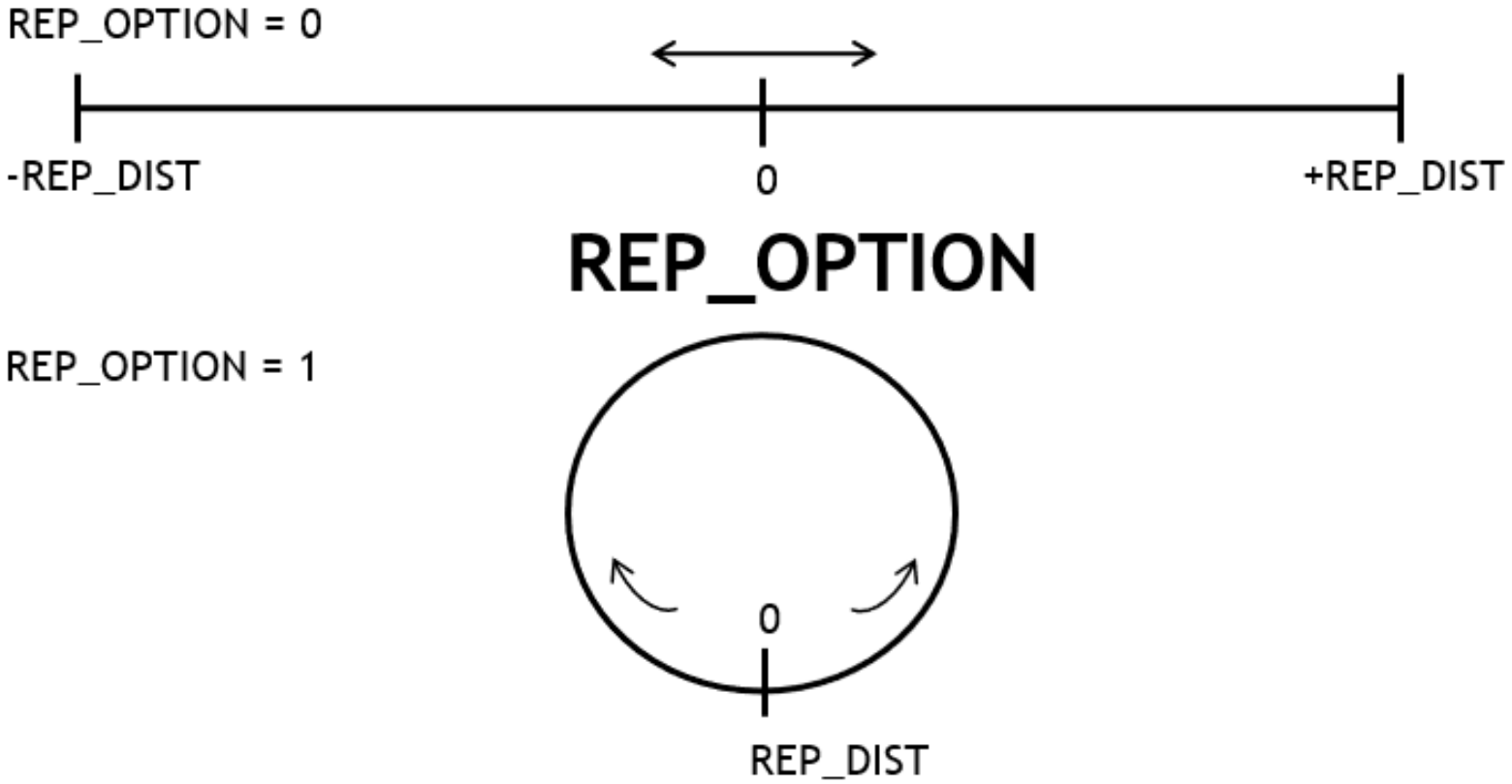
```
'Need to print double escape code (ASCII 27) to output a single escape
PRINT #5, CHR(27); CHR(27);

'Move cursor to row 14, column 10
PRINT #5, "[14;10H";

'Display the temperature`
PRINT #5, "Temperature: "; AIN(1);
```

MOTION 2

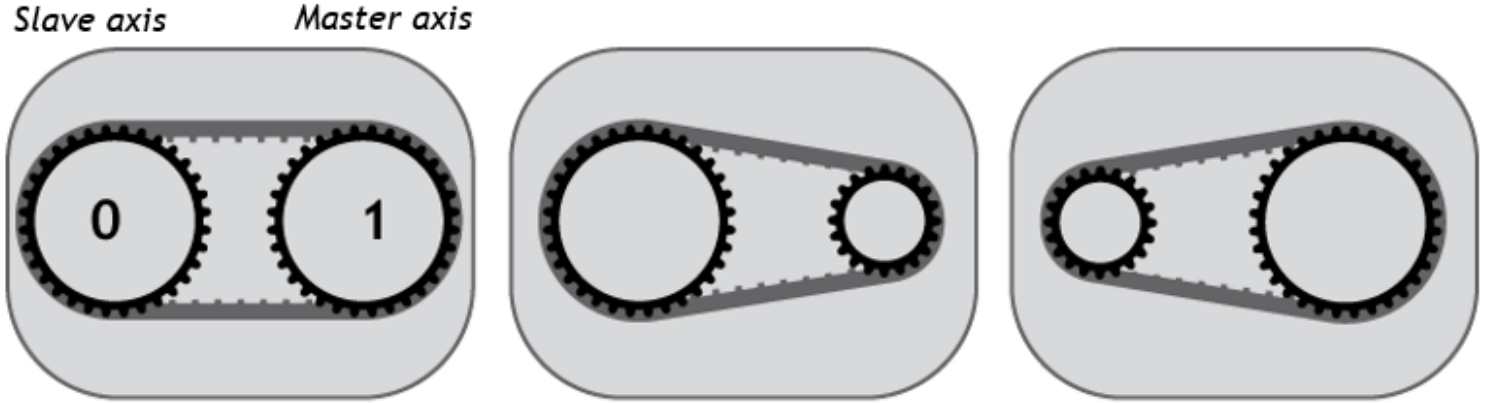
Continuously running axes need a different "wrap around" than axes with end limits.

An axis has 4000 counts per revolution, configure REP_DIST and REP_OPTION so that it wraps from 0 to 4000.

```
REP_OPTION = 1
REP_DIST = 4000
```



Multiple Axis Linked Continuous Moves



CONNECT(1, 1) CONNECT(0.5, 1) CONNECT(2, 1)

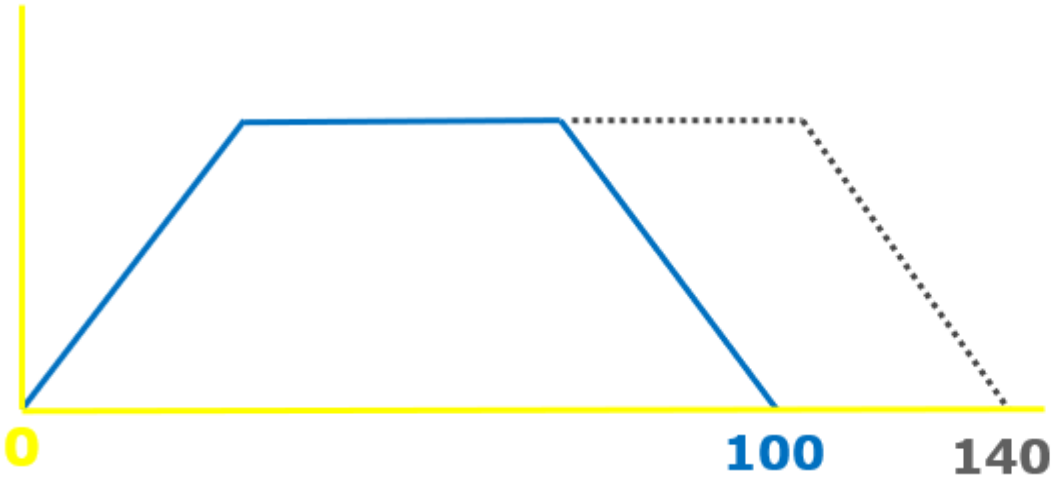
CONNECT(0,1) - Connection remains but slave axis does not move

```
connectaxis:
  VR(10) = 1.0
  BASE(0)
  TICKS=5000
  WHILE TICKS>0
    CONNECT(VR(10), 2)
  WEND
  RAPIDSTOP
  RETURN
```

CONNECT does not buffer in NTYPE

Each new CONNECT simply changes the ratio

MOVE Modify



MOVEABS (100)

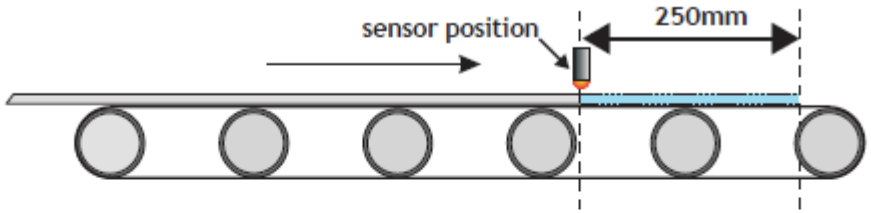
MOVEMODIFY (140)

MOVEMODIFY:

- Starts movement with the programmed speeds and accelerations
- Multiple axis can be run simultaneously
- No new command issued, updates end position

MOVEMODIFY(*position*)

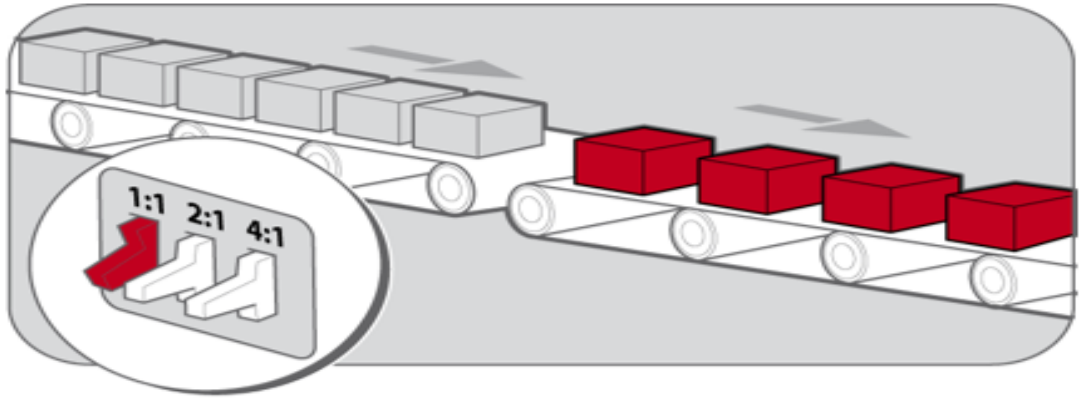
- Set the final position to a point in the workspace




```
MOVE(10000)' Start a long move on conveyor
REGIST(3)'   set up registration
WAIT UNTIL MARK 'MARK will be true when proximity seen
OFFPOS=-REG_POS'set position where mark seen to 0
MOVEMODIFY(250)'change move to stop at 250mm
```

Multiple Axis Linked Continuous Moves


```
part_seperator:
  GOSUB table_val
  BASE(1)
  CONNECT(0,0)
  REVERSE AXIS(0)
  conn_index=0
  CLUTCH_RATE=5
  WHILE IN(1)=ON
    IF IN(4,6)<>conn_index THEN
      conn_index=IN(4,6)
      CONNECT(TABLE(conn_index),0)
    ENDIF
  WEND
  RAPIDSTOP
  RETURN
table_val:
  TABLE(0,0,0.25,0.4,0.55,0.7,1,2,2.2)
  RETURN
```



Clutching a change in ratio:

- A change in connection ratio, acceleration can be defined
- This is the change in connection ratio per second
- Default is very high, attempts a step change in speed

CLUTCH_RATE=*value*

- Sets the value to the CLUTCH_RATE parameter

This affects operation of CONNECT by changing the connection ratio at the specified rate/second.

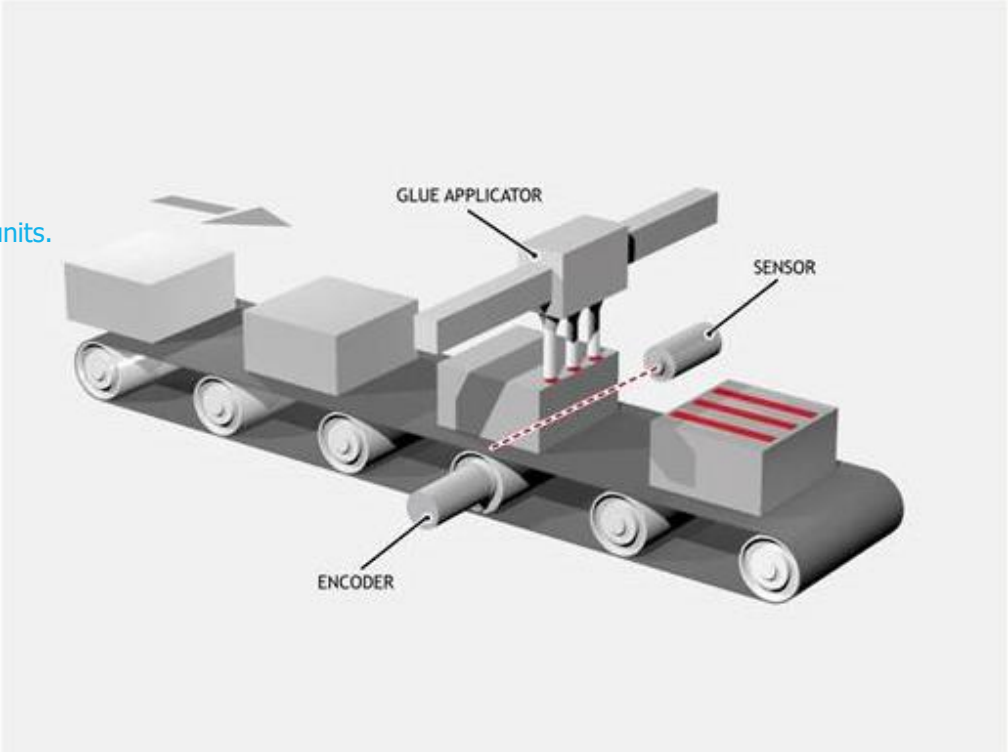
The connection ratio will be changed from 0 to 6 when an input is set.
It is required to take 2 second to accelerate
the linked axis so the ratio must change at 3 per second.

```
CLUTCH_RATE = 3
CONNECT(0,0)
WAIT UNTIL IN(1)=ON
CONNECT(6,0)
```

Registration

A machine adds glue to the top of a box by switching output 8. It must detect the rising edge (appearance) of and the falling edge (end) of a box. Additionally, it is required that the MPOS be reset to zero on the detection of the Z position.

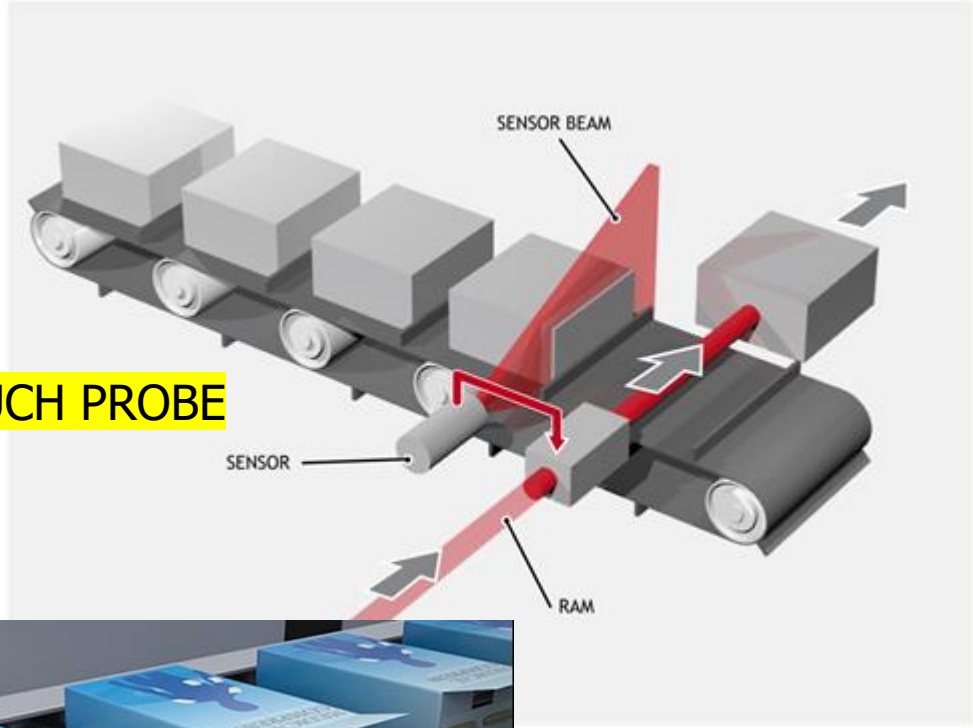
```
reg = 6 'Select registration mode 6 (rising edge R, rising edge Z)
REGIST(reg)
FORWARD
WHILE IN(2) = OFF
  IF MARKB THEN  'On a Z mark MPOS is reset to zero
    OFFPOS = -REG_POSB 'Stores the latest position at which a registration mark was seen on each axis in user units.
    REGIST(reg)
  ELSEIF MARK THEN  'On R input output 8 is toggled
    IF reg = 6 THEN
      'Select registration mode 8 (falling edge R, rising edge Z)
      reg = 8
      OP(8, ON)
    ELSE
      reg = 6
      OP(8, OFF)
    ENDIF
    REGIST(reg)
  ENDIF
WEND
CANCEL
```



Registration & Windowing

It is required to detect if a component is placed on a flighted belt, so windowing is used to avoid sensing the flights. The flights are at a pitch of 120 mm and the component will be found between 30 and 90 mm. If a component is found, then an actuator is fired to push it off the belt.

```
REP_DIST = 120          'sets repeat distance to pitch of belt flights
REP_OPTION = ON
OPEN_WIN = 30           'sets window open position
CLOSE_WIN = 90          'sets window close position
REGIST(17 + 256)        'RB input registration with windowing
FORWARD                 'start the belt
box_seen = 0
REPEAT
  WAIT UNTIL MPOS < 60 'wait for center point between flights
  WAIT UNTIL MPOS > 60 'so that actuator is fired between flights
  IF box_seen = 1 THEN 'was a box seen on the previous cycle?
    OP(8, ON)           'fire actuator
    WA(100)
    OP(8, OFF)          'retract actuator
    box_seen = 0
  ENDIF
  IF MARKB THEN box_seen = 1 'set "box seen" flag
  REGIST(17 + 256)
UNTIL IN(2) = OFF
CANCEL                  'stop the belt
WAIT IDLE
```

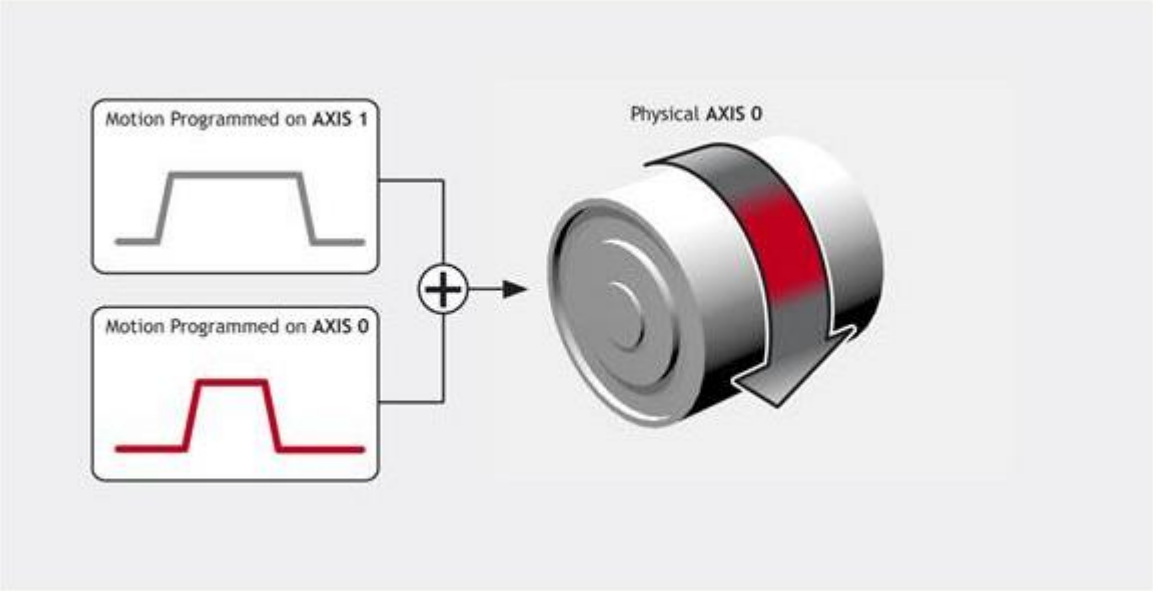


ADDAX

Parameter:

axis:	Axis to superimpose. -1 breaks the link with the other axis.

The ADDAX command sums the movements in encoder edge units.



Using ADDAX on axis with different UNITS ,
axis 0 will move 1 * 1000 + 2 * 20 = 1040 edges.

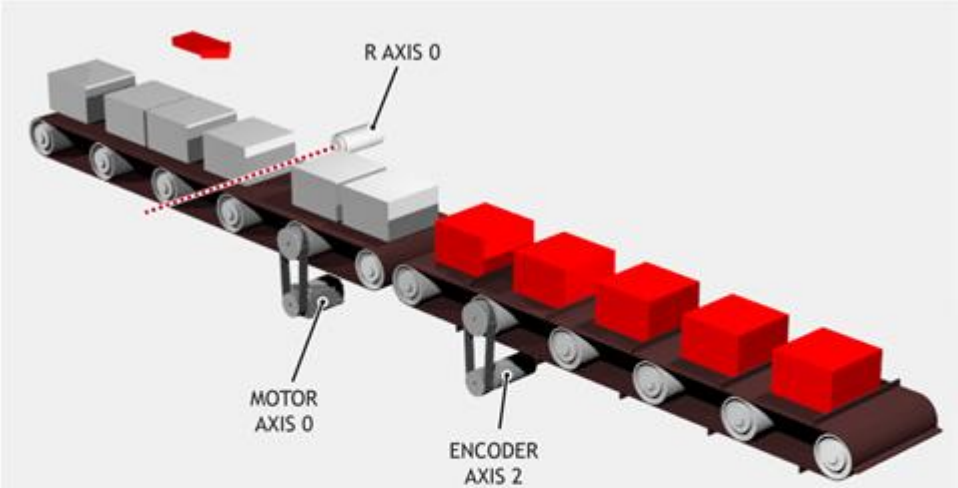
```
UNITS AXIS(0) = 1000
UNITS AXIS(1) = 20

'Superimpose axis 1 on axis 0
ADDAX(1) AXIS(0)
MOVE(1) AXIS(0)
MOVE(2) AXIS(1)
```

ADDAX

The ADDAX command is used to superimpose 2 or more movements to build up a more complex movement profile

Pieces are placed randomly onto a continuously moving belt and further along the line are transferred to a second flighted belt. A detection system gives an indication as to whether a piece is in front of or behind its nominal position, and how far.

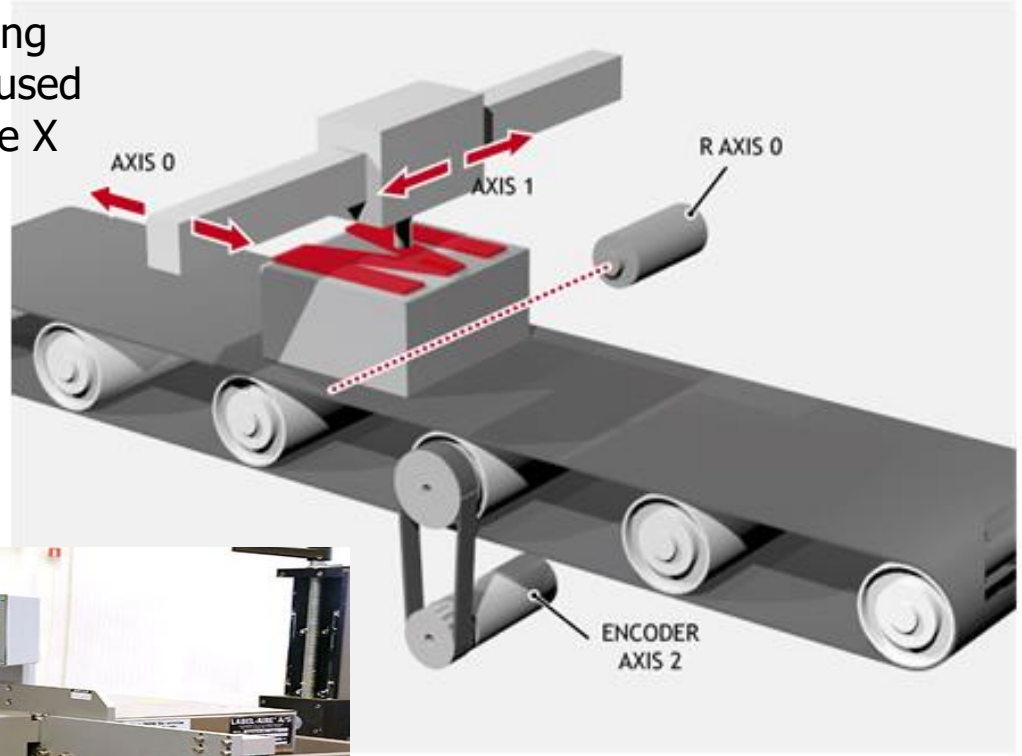
```
expected = 2000 'Sets expected position
  BASE(0)
  ADDAX(1)
  CONNECT(1, 2) 'Continuous geared connection to flighted belt
  REPEAT
    GOSUB getoffset 'Get offset to apply
    MOVE(offset) AXIS(1) 'Make correcting move on virtual axis
  UNTIL IN(2) = OFF 'Repeat until stop signal on input 2
  RAPIDSTOP
  ADDAX(-1) 'Clear ADDAX connection
  STOP

Getoffset: 'Subroutine to register the position of the
           'piece and calculate the offset
  BASE(0)
  REGIST(3)
  WAIT UNTIL MARK
  seenat = REG_POS
  offset = expected - seenat
  RETURN
```



ADDAX + RGIST

An XY marking machine must mark boxes as they move along a conveyor. Using CONNECT enables the X marking axis to follow the conveyor. A virtual axis is used to program the marking absolute positions; this is then superimposed onto the X axis using ADDAX.

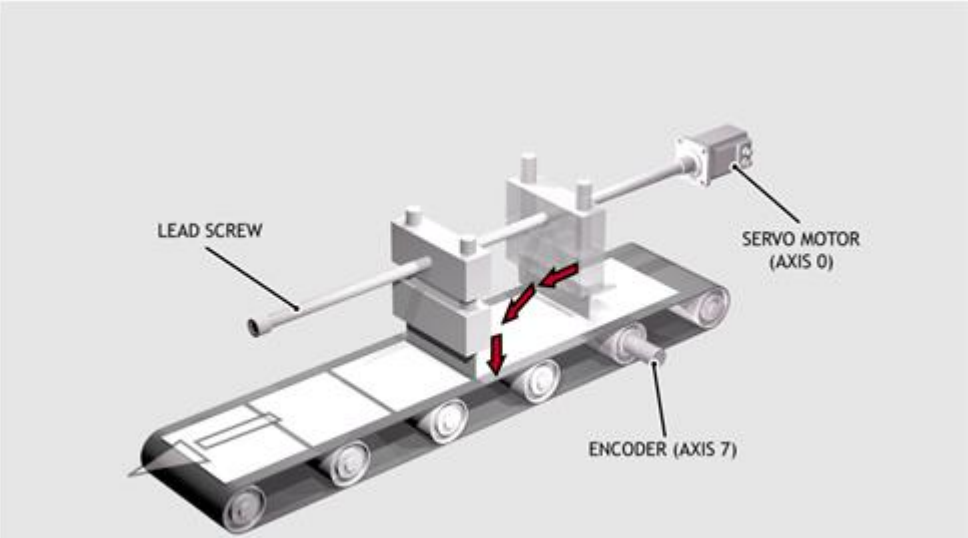
```
ATYPE AXIS(3) = 0 'Set axis 3 as virtual axis
SERVO AXIS(3) = ON
DEFPOS(0) AXIS(3)
ADDAX (3) AXIS(0) 'Connect axis 3 requirement to axis 0
WHILE IN(2) = ON
  REGIST(3) 'Registration input detects a box on the conveyor
  WAIT UNTIL MARK OR IN(2) = OFF
  IF MARK THEN
    CONNECT(1, 2) AXIS(0) 'Connect axis 0 to the moving belt
    BASE(3, 1) 'Set the drawing motion to axis 3 and 1
    'Draw the M
    MOVEABS(1200, 0) 'Move A > B
    MOVEABS(600, 1500) 'Move B > C
    MOVEABS(1200, 3000) 'Move C > D
    MOVEABS(0, 0) 'Move D > E
    WAIT IDLE
    BASE(0)
    CANCEL 'Stop axis 0 from following the belt
    WAIT IDLE
    MOVEABS(0) 'Move axis 0 to home position
  ENDIF
WEND
CANCEL
```



MOVELINK

A flying shear cuts a long sheet of paper into cards every 160 m whilst moving at the speed of the material. The shear can travel up to 1.2 m of which 1 m is used in this example. The paper distance is measured by an encoder; the unit conversion factor being set to give units of meters on both axes: (Note that axis 7 is the link axis)

```
WHILE IN(2) = ON
  MOVELINK(0, 150, 0, 0, 7)        'Dwell (no movement) for 150m
  MOVELINK(0.3, 0.6, 0.6, 0, 7)    'Accelerate to paper speed
  MOVELINK(0.7, 1.0, 0, 0.6, 7)    'Track the paper then decelerate
  WAIT LOADED 'Wait until acceleration movelink is finished
  OP(8, ON)   'Activate cutter
  MOVELINK(-1.0, 8.4, 0.5, 0.5, 7) 'Retract cutter back to start
  WAIT LOADED
  OP(8, OFF)  'Deactivate cutter at end of outward stroke
WEND
```



Syntax:

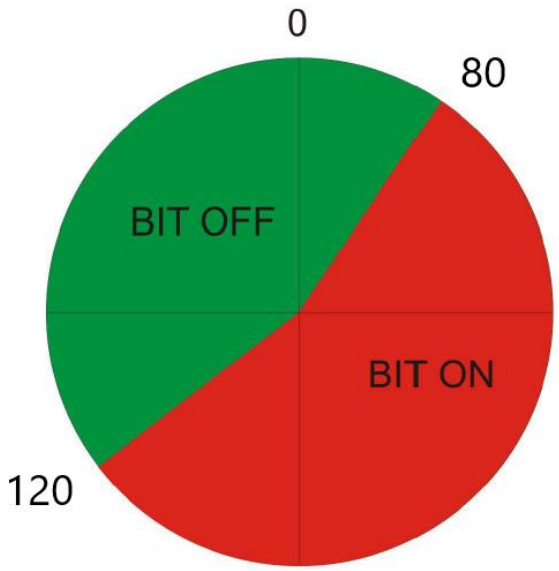
```
MOVELINK (distance, link_dist, link_acc, link_dec, link_axis[, link_options[, link_pos[, base_dist]]]).
```

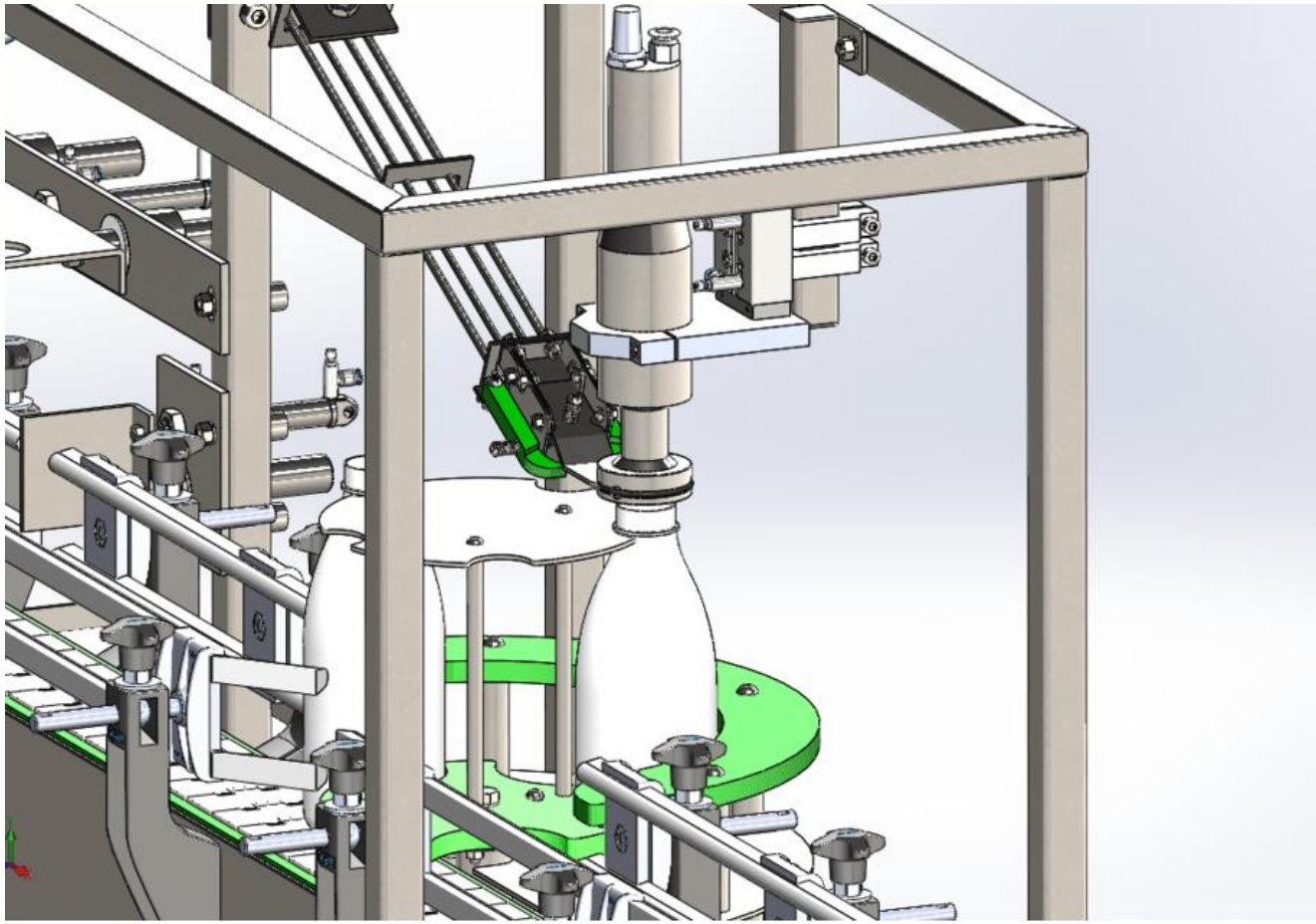
PSWITCH

Imagine the same situation as in Example 1, where output 11 must give a falling edge signal to the "cam switch" system, so that the output must be off from 80° after TDC for a period of 120°.

The values for all the PSWITCH parameters:

```
axis_number = 0
op_number = 11
op_state = OFF
setpos = 80 ' degrees after TDC
reset_pos = setpos + 120 ' degrees

PSWITCH(24,ON,axis_number,op_number,op_state,setpos,reset_pos)
```

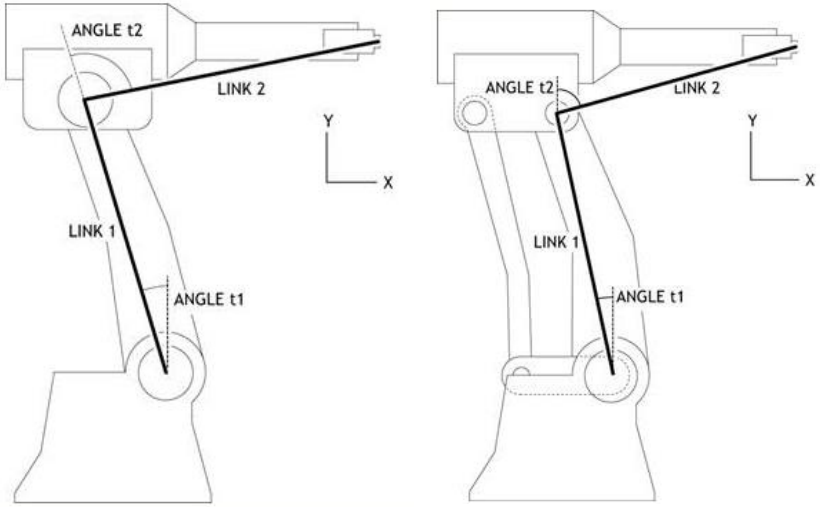




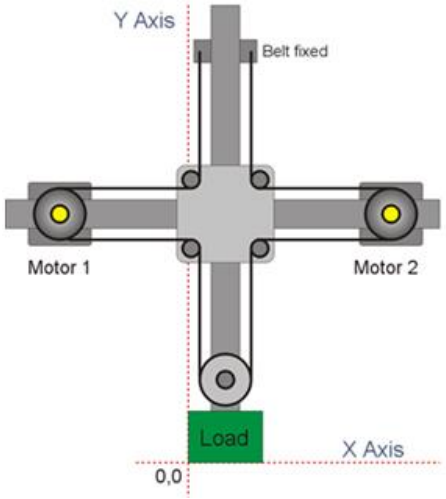
FRAME - ROBOT

A FRAME is a transformation which enables the user to program in one coordinate system when the machine or robot does not have a direct or one-to-one mechanical connection to this coordinate system.

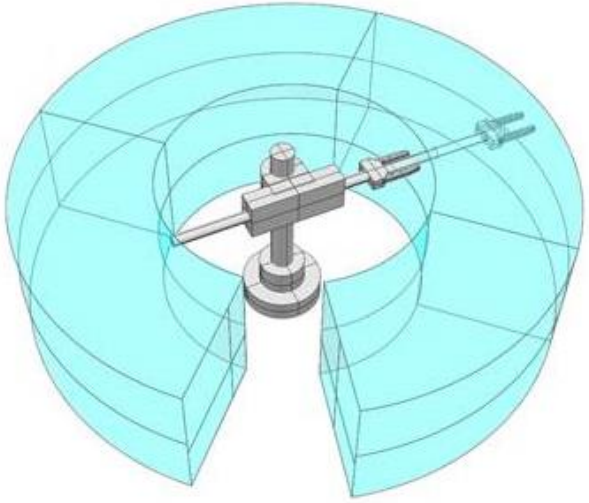
FRAME=1, 2 axis SCARA



FRAME=2, XY single belt

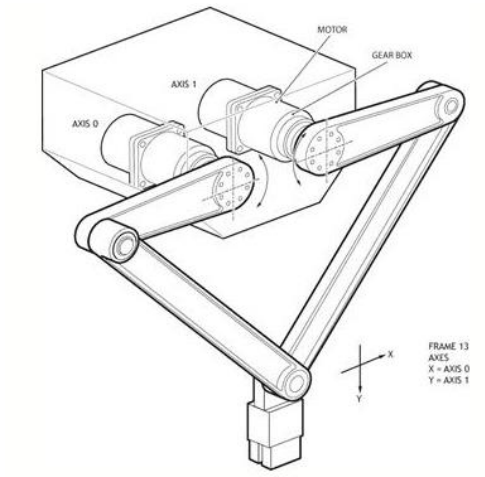


FRAME=10, Cartesian to Polar

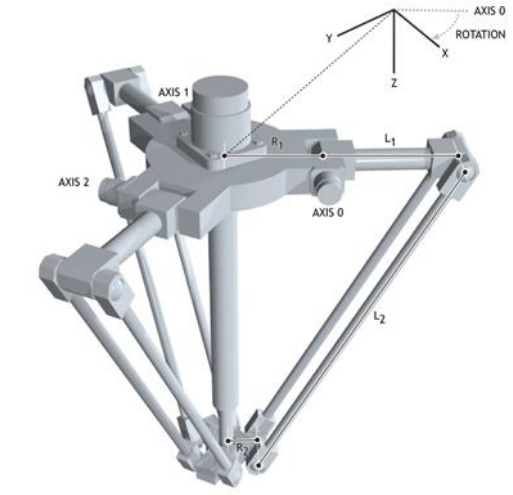


FRAME - ROBOT

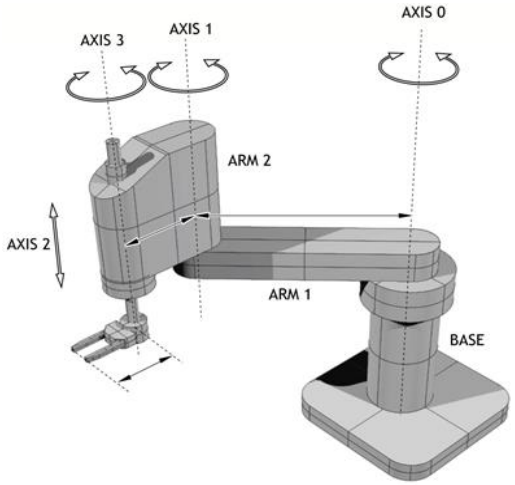
FRAME=13, Dual arm Parallel



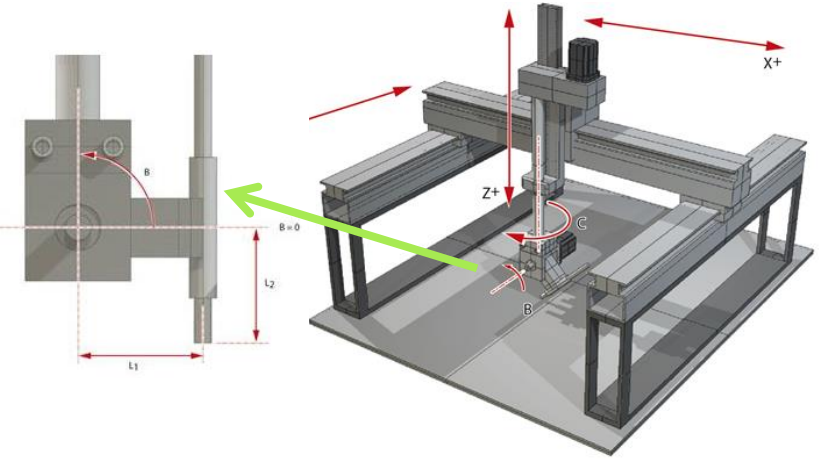
FRAME=14, Delta Robot



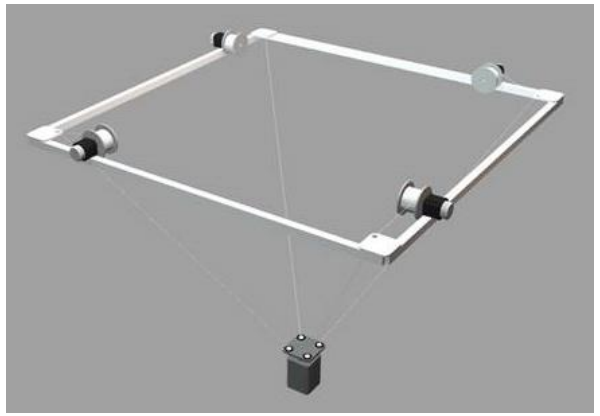
FRAME=15, 4 axis SCARA



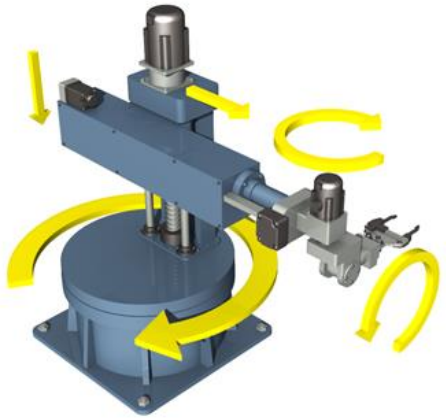
FRAME = 16, 3 Axis Robot with 2 Axis Wrist



FRAME=17, Multi-wire camera



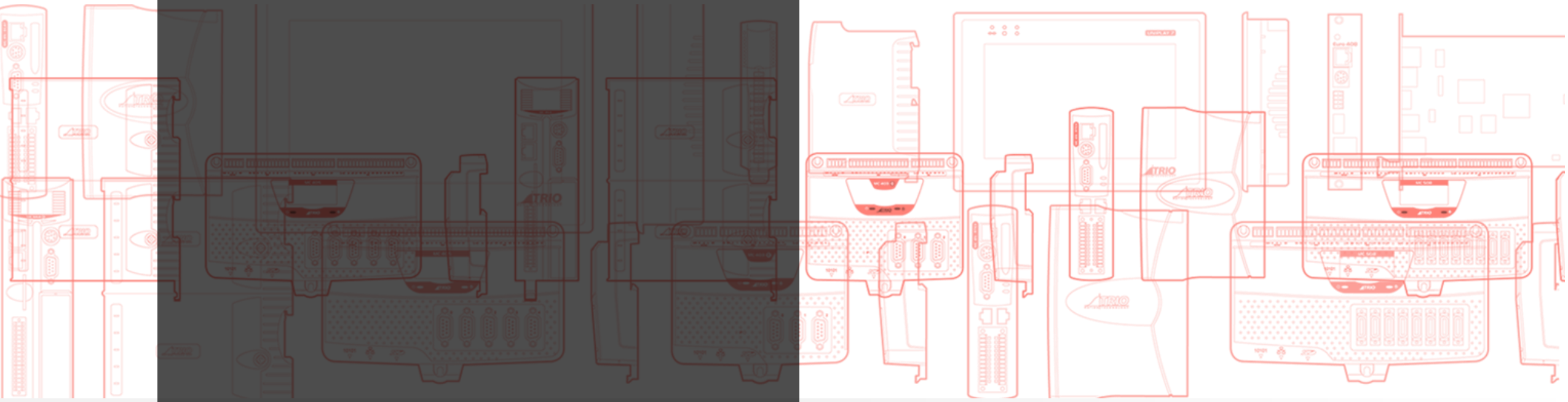
FRAME 119 (Draft)



Companies Overview

“TRIO and ESTUN”

ESTUN AUTOMATION



Thank You

Reza Marzban

Business Development Specialist

Iranieq Co.

www.iranieq.com