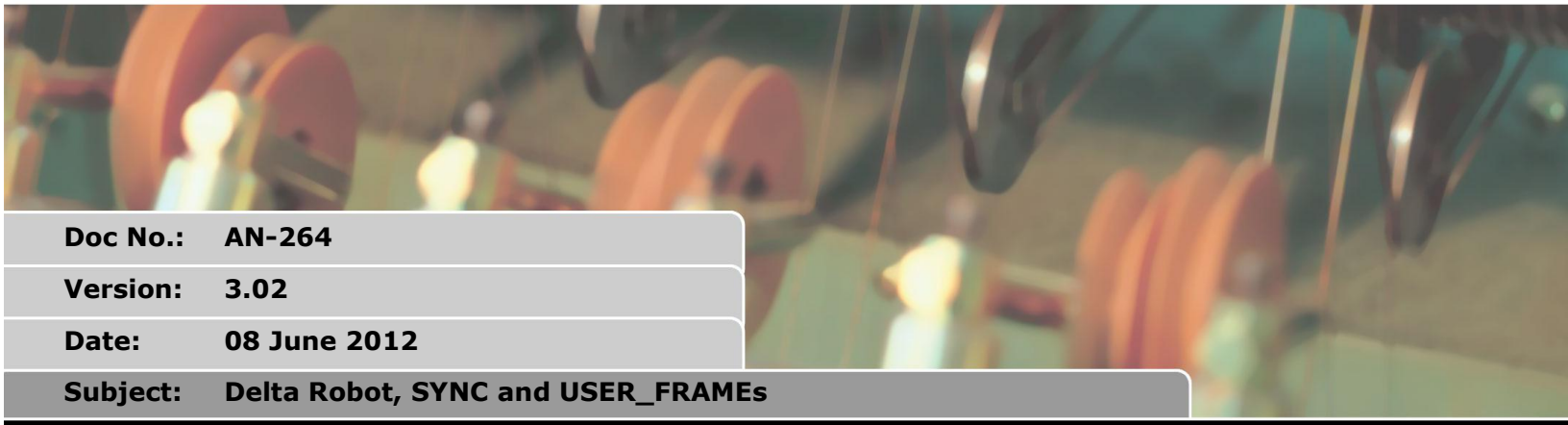
Trio Motion Technology Ltd.
Shannon Way, Tewkesbury,
Gloucestershire. GL20 8ND
United Kingdom
Tel: +44 (0)1684 292 333
Fax: +44 (0)1684 297 929

1000 Gamma Drive
Suite 206
Pittsburgh, PA 15238
United States of America
Tel: +1 412 968 9744
Fax: +1 412 968 9746

B1602 Tomson Centre
188 Zhang Yang Rd.,
Pudong New Area, Shanghai,
Postal code: 200122
CHINA
Tel/Fax: +86 21 587 97659



Doc No.: AN-264

Version: 3.02

Date: 08 June 2012

Subject: Delta Robot, SYNC and USER_FRAMEs

APPLICATION NOTE

1. Change History

3.00	-	Release version
3.01	-	Updated FRAME_TRANS Ex1
		Updated TOOL_OFFSET Ex
3.02	-	Corrected USER_FRAME parameter list, SYNTAX was correct.

2. Required Version

2.0149 or newer

3. Introduction

The 3 arm delta is a commonly used robot for pick and place applications. Multiple robots can used with conveyors and vision systems to form the full system. Using the functionality detailed in this application note it is possible to program these systems using the standard TrioBASIC commands plus a few new ones.

One key new feature is the ability to synchronise to a point on a conveyor. The new command SYNC will enable the user to easily synchronise to a conveyor in all 3 axis. Once synchronised he user can then perform operations on the synchronised position before cancelling synchronisation or re-synchronising to a different point or a different conveyor.

The following new features have been implemented to simplify programming of delta robots but of course can be used on any application*.

- Define up to 31 tool offsets for use with a multi tool wrist.
- Change your reference position by defining up to 31 user coordinate systems.
- Translate positions from your vision system to any defined user coordinate system.
- Track items on a conveyor and move from tracking on one conveyor to tracking another in just a couple of lines of BASIC.
- Run up to 8 FRAMEs on one controller.
- Define which axes to use for a robot.
- Enable software limits to the motor axis or the robots coordinate system.
- Enable a cone/cylindrical working volume limit.
- Use any of the above features on any FRAME including 2 arm delta and, SCARA XY belt transformations

All of these features are included in the standard MC464 firmware however some of the functions require the kinematic runtime feature enable code. On the MC464 this is FEC number 22.

4. AXIS_DPOS

4.1. Type:

Axis Parameter (Read Only)

4.2. Alternative Format

TRANS_DPOS

4.3. Description:

AXIS_DPOS is the axis demand position at output of FRAME transformation.

AXIS_DPOS is normally equal to DPOS on each axis. The frame transformation is therefore equivalent to 1:1 for each axis (FRAME = 0). For some machinery configurations it can be useful to install a frame transformation which is not 1:1, these are typically machines such as robotic arms or machines with parasitic motions on the axes. In this situation when FRAME is not zero AXIS_DPOS returns the demand position for the actual motor.

Note:

AXIS_DPOS can be scaled using AXIS_UNITS.

4.4. Parameters:

value: The axis demand position at the output of the FRAME transformation.

4.5. Example:

Return the axis position in user UNITS using the command line.

```
>>PRINT AXIS_DPOS
125.22
>>
```

4.6. See also:

AXIS_UNITS, FRAME

5. AXIS_FS_LIMIT

5.1. Type:

Axis Parameter

5.2. Description:

An axis end of travel limit may be set up in software thus allowing the program control of the working range of the axis. This parameter holds the absolute position of the forward travel limit in user AXIS_UNITS.

Bit 16 of the AXISSTATUS register is set when the axis position is greater than the AXIS_FS_LIMIT.

Axis software limits are only enabled when FRAME<>0 so that the user can limit the range of motion of the motor/ joint.

Note:

When AXIS_DPOS reaches AXIS_FS_LIMIT the controller will CANCEL all moves on the FRAME_GROUP, the axis will decelerate at DECEL or FAST_DEC. Any SYNC is also stopped. As this software limit uses AXIS_DPOS it will require a negative change in AXIS_DPOS to move off the limit. This may not be a negative movement on DPOS due to the selected FRAME transformation.

Tip:

AXIS_FS_LIMIT is disabled when it has a value greater than REP_DIST or when FRAME=0.

5.3. Parameters:

value:	The absolute position of the software forward travel limit in user units. (default = 200000000000)

5.4. Example:

Set up an axis software limit so that the axis operates between 180 degrees and 270 degrees. The encoder returns 4000 counts per revolution.

```
AXIS_UNITS=4000/360
AXIS_FS_LIMIT=270
AXIS_RS_LIMIT=180
```

5.5. See Also:

AXIS_DPOS, AXIS_RS_LIMIT, AXIS_UNITS

6. AXIS_RS_LIMIT

6.1. Type:

Axis Parameter

6.2. Description:

An axis end of travel limit may be set up in software thus allowing the program control of the working range of the axis. This parameter holds the absolute position of the reverse travel limit in user AXIS_UNITS.

Bit 17 of the AXISSTATUS register is set when the axis position is greater than the AXIS_RS_LIMIT.

Axis software limits are only enabled when FRAME<>0 so that the user can limit the range of motion of the motor/ joint.

Note:

When AXIS_DPOS reaches AXIS_RS_LIMIT the controller will CANCEL all moves on the FRAME_GROUP, the axis will decelerate at DECEL or FAST_DEC. Any SYNC is also stopped. As this software limit uses AXIS_DPOS it will require a positive change in AXIS_DPOS to move off the limit. This may not be a positive movement on DPOS due to the selected FRAME transformation.

Tip:

AXIS_RS_LIMIT is disabled when it has a value greater than REP_DIST or when FRAME=0.

6.3. Parameters:

value:	The absolute position of the software forward travel limit in user units. (default = -200000000000)

6.4. Example:

An arm on a robots joint can move 90degrees. The encoder returns 400 counts per revolution and there is a 50:1 gearbox

```
AXIS_UNITS=4000*50/360
AXIS_FS_LIMIT=0
AXIS_RS_LIMIT=90
```

6.5. See Also:

AXIS_DPOS, AXIS_RS_LIMIT, AXIS_UNITS

7. AXIS_UNITS

7.1. Type:

Axis Parameter

7.2. Description:

AXIS_UNITS is a conversion factor that allows the user to scale the edges/ stepper pulses to a more convenient scale. AXIS_UNITS is only used when a FRAME is active and only applies to the parameters in the axis coordinate system (after the FRAME). This includes AXIS_DPOS, AXIS_FS_LIMIT, AXIS_RS_LIMIT and MPOS.

Warning:

MPOS will use UNITS when FRAME=0 and AXIS_UNITS when FRAME <> 0

7.3. Parameters:

value: The number of counts per required units (default =1).

7.4. Example:

A motor on a robot has a 18bit encoder and uses an 18bit encoder and 31:1 ratio gearbox. To simplify reading AXIS_DPOS the user wants to use radians.

```
encoder_bits = 2^10
gearbox_ratio = 31
radians_conversion=2*PI
AXIS_UNITS=( encoder_bits * gearbox_ratio)/ radians_conversion
```

7.5. See Also:

AXIS_DPOS, UNITS

8. FRAME

8.1. Type:

Axis Parameter

8.2. Description:

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

The FRAME command selects which transformation to use on axes in a FRAME_GROUP. Applying a FRAME to an axis in a FRAME_GROUP will apply that frame to all the axes in the group. To make this compatible with older firmware, if no FRAME_GROUPS have been configured then a default group is generated using the lowest axes, regardless of what axis the FRAME parameter was issued on.

Most transformations require configuration data to specify the lengths of mechanical links or operating modes. This is stored in the table with offsets detailed below in the parameters list. These table positions are offset by the 'table_offset' parameter in FRAME_GROUP. For a default FRAME_GROUP table_offset is 0.

Note:

The kinematic runtime feature enable code is required to run FRAME 14 and higher

Axis scaling

When a FRAME is enabled UNITS applies the scaling to the world (Cartesian) coordinate system and AXIS_UNITS applies scaling to the axis coordinate system.

Warning:

When frame is enabled MPOS is scaled by AXIS_UNITS, when frame is disabled MPOS is scaled by UNITS.

Position and following errors

When a FRAME is active MPOS is the motor position and DPOS is in the world coordinate system. AXIS_DPOS can be read to find the demand position in the motor coordinate system.

The following error is calculated between MPOS and AXIS_DPOS and so is the following error of the motor.

Hardware and Software limits

As FS_LIMIT and RS_LIMIT use DPOS they are both active in the world coordinate system. VOLUME_LIMIT also uses DPOS so is also in the world coordinate system. FWD_IN and REV_IN, AXIS_FS_LIMIT and AXIS_RS_LIMIT use AXIS_DPOS as so act on the forward and reverse limit of the motor.

Note:

When moving off FWD_IN and AXIS_FS_LIMIT the motor must move in a reverse direction. Due to the FRAME transformation this may not be a reverse movement in the FRAME coordinates. When moving off a REV_IN and AXIS_RS_LIMIT the motor must move in a forward direction. Due to the FRAME transformation this may not be a forward movement in the FRAME coordinates.

Power on sequence and Homing

Most FRAME transformations require the machine to be homed and/ or moved to a position before the FRAME is enabled. This can be done using standard DATUM commands.

When a FRAME is enabled DPOS is adjusted to the world coordinates which are calculated from the current MPOS.

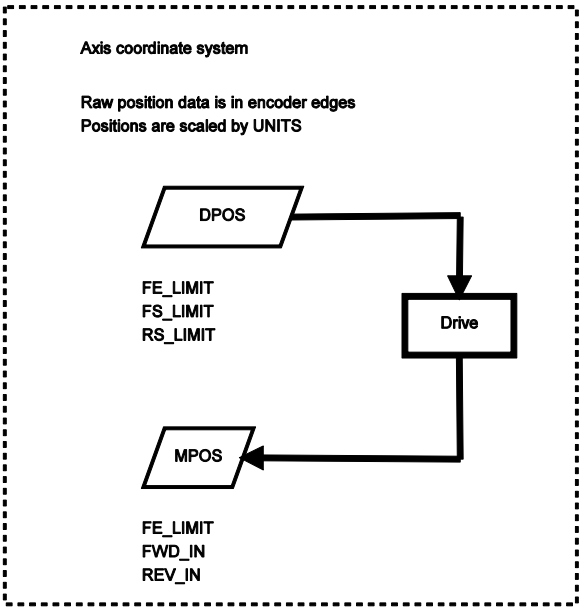
Tip:

When using multiple frames or if you wish to group your axis you can use DISABLE_GROUP so that a MOTION_ERROR on one axis does not affect all.

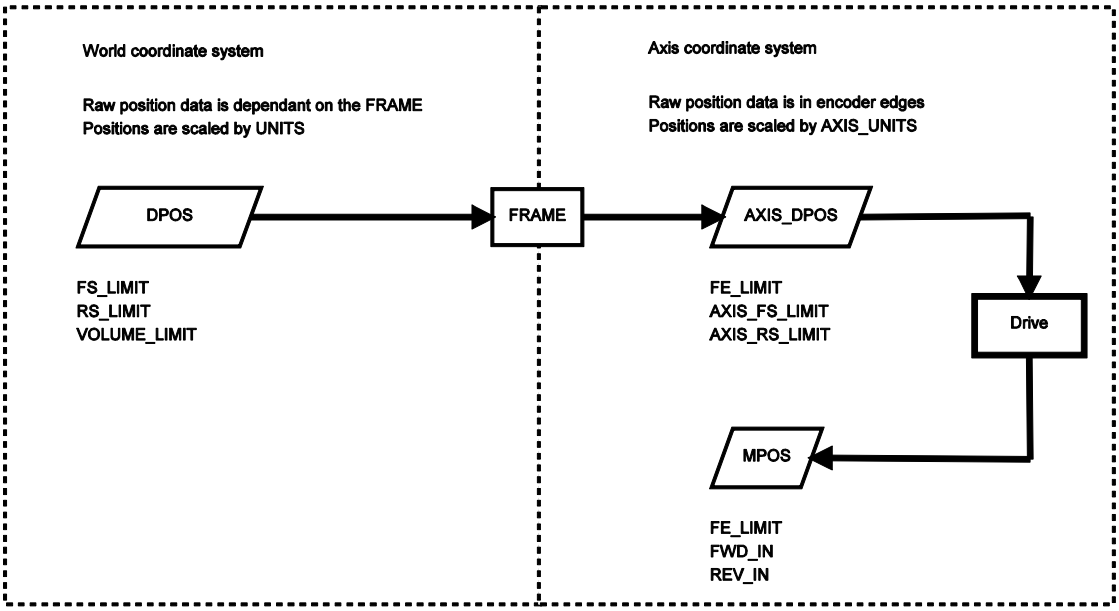
Offsetting positions

When a FRAME is enabled OFFPOS and DEFPOS must not be used. You can use USER_FRAME to define a different origin to program from.

System with FRAME=0



System with FRAME<>0



8.3. Parameters:

value:

0 = No transform
1 = 2 axis SCARA robot
2 = XY single belt
3 = Double XY single belt
4 = 2 axis pick and place
5 = 2x2 Matrix transform
6 = Polar to Cartesian transformation
10 = Cartesian to polar transformation
13 = Dual arm robot transformation
14 = 3 arm delta robot.
15 = 4 axis SCARA

Note:

Only FRAME=14 is considered in this application note, please see other application notes or the manual for the other FRAMEs

---

8.4. Syntax:

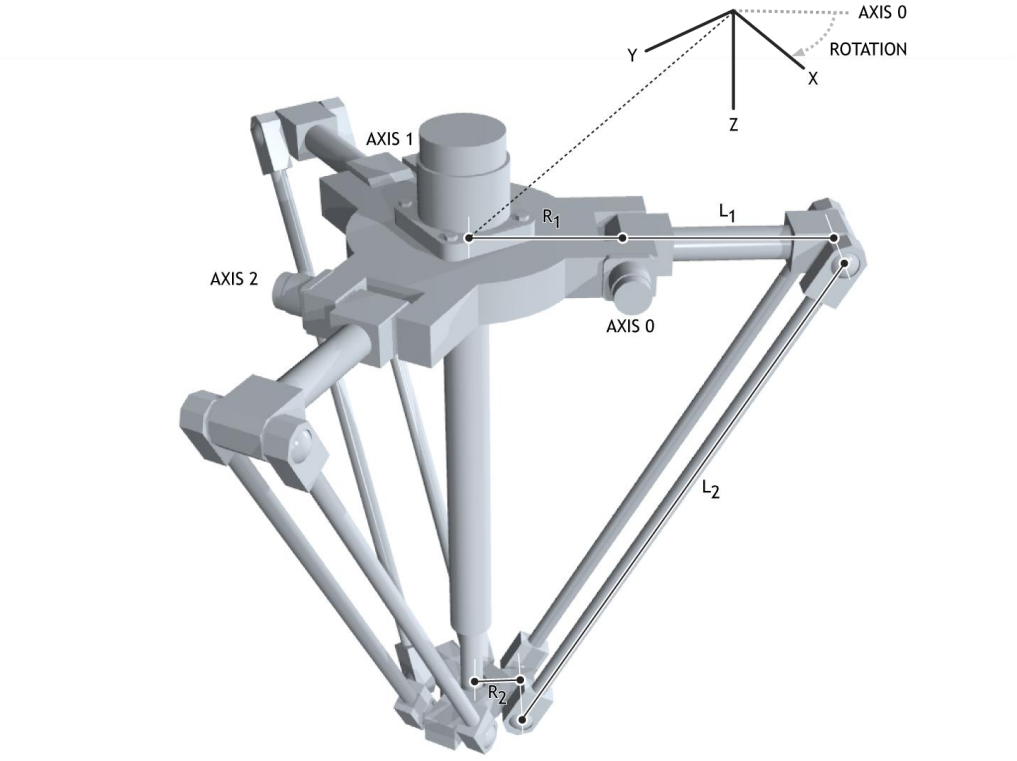
FRAME=14

8.5. Description:

FRAME=14 enables the transformation for a 3 arm 'delta' or 'parallel' robot. It transforms 3 axes from the mechanical configuration to Cartesian coordinates using the right hand rule.

Note:

FRAME=14 requires the kinematic runtime FEC



Once the frame is enabled the raw position data (UNITS=1) is measured in Micrometres, UNITS can then be set to a convenient scale. The origin for the robot is the centre of the top plate with the X direction following the first axis. This can be adjusted using the rotation parameter.

Homing

Before enabling FRAME=14 the position must be defined so that when the upper arms are horizontal the axis position is 0. You do not need to start in this position, just ensure that it has been defined.

8.6. Parameters:

value: 0 = No transform
14 = 3 arm delta robot.

The Table data defines the mechanical configuration of the robot, the values in brackets refer to the labels in the above diagram.

Table data 0 = Top radius to joint in Micrometres (R1)
1 = Wrist radius to joint in Micrometres (R2)
2 = Upper arm length in Micrometres (L1)
3 = Lower arm length in Micrometres (L2)
4 = Edges per radian
5 = Angle of rotation in radians (Rotation)

8.7. Example

Start-up sequence for a 3 arm delta robot using the default FRAME_GROUP. Homing is completed using a sensor that detects when the upper arms are level.

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Define Link Lengths for 3 arm delta:
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

  TABLE(0,200000)' Top radius to joint
  TABLE(1,50000)'  Wrist radius to joint
  TABLE(2,320000)' Upper arm length
  TABLE(3,850000)' Lower arm length

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Define encoder edges/radian
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
  '18bit encoder and 31:1 ratio gearbox
  resolution = 262144 * 31 / (2 * PI)
  TABLE(4,resolution)

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Define rotation of robot relative to global frame
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
  rotation = 30 'degrees
  TABLE(5, (rotation*2*PI )/360)

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Configure axis
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
  FOR axis_number=0 TO 2
    BASE(axis_number)
    'World coordinate system to operate in mm
    UNITS=1000
    SERVO=ON
  NEXT axis_number
```

```
  WDOG=ON
  BASE(0)

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Home and initialise frame
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
  'Arms MUST be horizontal in home position
  ' before frame is initialised.
  FOR axis_number=0 TO 2
    DATUM(4)
    WAIT IDLE
  NEXT axis_number

  'Enable Frame
  FRAME=14
```

8.8. See Also:

FRAME_GROUP, USER_FRAME

9. FRAME_GROUP

9.1. Type:

System Command

9.2. Syntax:

FRAME_GROUP(group, [table_offset, [axis0, axis1 …axisn]])

9.3. Description:

FRAME_GROUP is used to define the group of axes and the table offset which are used in a FRAME or USER_FRAME transformation. There are 8 groups available meaning that you can run a maximum of 8 FRAMEs on the controller.

Note:

FRAME_GROUP requires the kinematic runtime FEC

Warning:

Although 8 FRAME's can be initialised on a controller it may not be possible to process all 8 at a given SERVO_PERIOD. The number that can be run depends on many factors including, which FRAME is selected, drive connection method, if USER_FRAME and TOOL_OFFSET are enabled and additional factory communications.

The number of axes in the group must match the number of axes used by the FRAME. The axes must also be ascending order though they do not have to be contiguous. If a group is deleted FRAME and USER_FRAME are set to 0 for those axes.

To maintain backward compatibility if the FRAME command is used on an axis that is not in a group, or no groups are configured then a default group is created using the lowest axes and table_offset=0. In this situation if FRAME_GROUP(0) is already configured it is overwritten.

9.4. Parameters:

group:	The group number, 0-7. When used as the only parameter FRAME_GROUP prints the FRAME_GROUP, the active USER_FRAME and TOOL_OFFSET information to the currently selected output channel (default channel 0)
table_offset:	-1 = Delete group data 0+ = The start position in the table to store the FRAME configuration.
axis0:	The first axis in the group
axis1:	The second axis in the group
axisn:	The last axis in the group

9.5. Example:

Configure a FRAME_GROUP for axes 1,2 and 5 using table offset 100.

```
'Initialise the FRAME_GROUP
FRAME_GROUP(0,100, 1,2,5)

'Configure the axes, FRAME table data and home the robot
 GOSUB configure_frame

'PRINT the FRAME_GROUP information to the command line
 FRAME_GROUP(0)
```

```
'Enable the frame
FRAME AXIS(1)=14
```

10. FRAME_TRANS

10.1. Type:

Mathematical Function

10.2. Syntax:

FRAME_TRANS(frame, table_in, table_out, direction [,table_offset])

10.3. Description:

This function enables you to perform both the forward and inverse transformation calculations of a FRAME. One particular use is to check following errors in user units or to calculate positions outside of the FRAME working area.

Note:

FRAME_TRANS requires the kinematic runtime FEC to use a FRAME 14 and higher.

Tip:

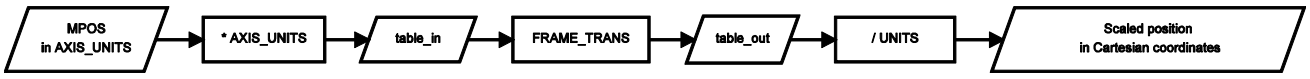
The FRAME calculations are performed on raw position data. When using a FRAME typically the raw position data for DPOS is micrometres and the raw position data for MPOS is encoder counts but this can vary depending on which FRAME you select.

10.4. Parameters:

frame:	The FRAME number to run
table_in	The start position in the TABLE of the input positions
table_out	The start position in the TABLE of the generated positions
direction	1 = AXIS_DPOS to DPOS (Forward Kinematics)
	0 = DPOS to AXIS_DPOS (Inverse Kinematics)
table_offset	The first position in the table where the frame configuration is found (default 0)

10.5. Example 1:

Using MPOS calculate the Cartesian values so you can compare them to DPOS. This can be used to check the following error in the world coordinate system. The frame configuration is stored in the table starting at position 100.




```
'Load positions into the table
FOR x=0 TO 3
BASE(x)
TABLE(1000+x,MPOS AXIS(x)*UNITS AXIS(x))
NEXT x
'Calculate forward transform to see MPOS is Cartesian coordinates
FRAME_TRANS(15, 1000,2000,1,100)

TABLE(3000, TABLE(2000)/ UNITS AXIS(0))
```

```
TABLE(3001, TABLE(2001)/ UNITS AXIS(1))
TABLE(3002, TABLE(2002)/ UNITS AXIS(2))
PRINT "DPOS IN ENCODER COUNTS",TABLE(2000),TABLE(2001),TABLE(2002)
PRINT "DPOS IN MM",TABLE(3000),TABLE(3001),TABLE(3002)
PRINT "FE in world x = ", TABLE(3000) - DPOS AXIS(0)
PRINT "FE in world y = ", TABLE(3001) - DPOS AXIS(1)
PRINT "FE in world z = ", TABLE(3002) - DPOS AXIS(2)
```

10.6. Example 2

Use the inverse kinematics to confirm that a demand position will result in an axis position that the motors can achieve.


```
'Load positions into the table
TABLE(5000,100*UNITS AXIS(0),200*UNITS AXIS(1),400*UNITS AXIS(2))

'Calculate reverse transform to see
FRAME_TRANS(14, 5000,6000,0)

'Divide the result by the AXIS_UNITS to get
'the MPOS in degrees
TABLE(7000, TABLE(6000)/ AXIS_UNITS)
TABLE(7001, TABLE(6001)/ AXIS_UNITS)
TABLE(7002, TABLE(6002)/ AXIS_UNITS)

PRINT "MPOS RAW ENCODER COUNTS", TABLE(6000),TABLE(6001),TABLE(6002)
PRINT "MPOS degrees", TABLE(7000),TABLE(7001),TABLE(7002)

WEND
```

11. INTERP_FACTOR

11.1. Type:

Axis Parameter

11.2. Description:

This parameter excludes the axis from the interpolated motion calculations so that it will become a following axis. This means that you can create an interpolated x,y move with z completing its movement over the same time period. The interpolated speed is calculated using any axes that have INTERP_FACTOR enabled. This means that at least one axis must be enabled and have a distance in the motion command otherwise the calculated speed will be zero and the command will complete immediately with no movement.

INTERP_FACTOR only operates with MOVE, MOVEABS and MHELICAL (on the 3rd axis) including the SP versions. All other motion commands require interpolated axes and so ignore this parameter.

11.3. Example:

It is required to move a 'z' axis interpolated with x and y however we want the interpolated speed to only be active on the x,y move. We disable the z axis from the interpolation group using INTERP_FACTOR. Remember when the movement is complete you must enable INTERP_FACTOR again.

```
BASE(2)
INTERP_FACTOR=0

'Perform movement
BASE(0,1,2)
MOVEABS(x_offset, y_offset, z_offset)

WAIT IDLE
INTERP_FACTOR AXIS(12) = 1
```

12. SYNC

12.1. Type:

Axis command

12.2. Description:

The SYNC command is used to synchronise one axis with a moving position on another axis. This can be used to synchronise a robot to a point on a conveyor. The user can define a time to synchronise and de-synchronise.

The synchronising movement on the base axis is the sum of two parts:

1. The conveyor movement from the 'sync_pos', this is the movement of the demand point along the conveyor.
2. The movement to 'pos1', this is the position in the current coordinate system where the sync_pos was captured on the slave axis.

When the axis is synchronised it will follow the movements on the 'sync_axis'. As the SYNC does not fill the MTYPE buffer you can perform movements while synchronised.

Note:

To synchronise to a new USER_FRAME using SYNC(20) requires the kinematic runtime FEC

Warning:

As SYNC does not get loaded in to the move buffer it is not cancelled by CANCEL or RAPIDSTOP, you have to perform SYNC(4). When a software or hardware limit is reached the SYNC is immediately stopped with no deceleration.

Tip:

Typically you can use the captured position REG_POS, or a position from a vision system for the 'sync_position'. The pos1, pos2 and pos3 are typically the position of the sensor/ vision system in the current USER_FRAME.

12.3. Syntax:

SYNC(control, sync_time, [sync_position, sync_axis, pos1[, pos2 [,pos3]]])

12.4. Parameters:

control:	1 = Start synchronisation, requires minimum first 5 parameters
	4 = Stop synchronisation, requires minimum first 2 parameters
	10 = Re-synchronise to another axis, requires minimum first 5 parameters
	20 = Re-synchronise to USER_FRAMEB, requires minimum first 5 parameters
sync_time:	Time to complete the synchronisation movement in milliseconds
sync_position:	The captured position on the sync_axis.
sync_axis:	The axis to synchronise with.
pos1:	Absolute position on the first axis on the base array
pos2:	Absolute position on the second axis on the base array
pos3:	Absolute position on the third axis on the base array

12.5. Example:

The robot must pick up the components from one conveyor and place them at 100mm pitch on the second. The registration sensor is at 385mm from the robots origin and the start of the second conveyor is 400mm from the robots origin.

```
'axis(0) - robot axis x
'axis(1) - robot axis y
'axis(2) - robot axis z
'axis(3) - robot wrist rotate
'These are the actual robot axis, FRAME=14 can be applied to these

'axis(10) - conveyor axis
'axis(11) - conveyor axis
'These are the real conveyors that you wish to link to

  'Sensor and conveyor offsets
  sen_xpos = 385
  conv1_yoff = 200
  conv2_yoff = -250
  conv2_xoff = 40
  place_pos = 0

  BASE(0,1)
  'Move to home position.
  MOVEABS(200,50)
  'start conveyors
  DEFPOS(0) AXIS(11) ' reset conveyor position for place
  FORWARD AXIS(10)
  FORWARD AXIS(11)
  WAIT IDLE

  WHILE(running)
    REGIST(20,0,0,0,0) AXIS(10)
    WAIT UNTIL MARK AXIS(10)

    SYNC(1, 1000, REG_POS, 10, sen_xpos , conv1_yoff)
    WAIT UNTIL SYNC_CONTROL AXIS(0)=3
    'Now synchronised
    GOSUB pick

    SYNC(10, 1000, place_pos, 11, conv2_xoff, conv2_yoff)
    WAIT UNTIL SYNC_CONTROL AXIS(0)=3
    'Now synchronised
    GOSUB place

    SYNC(4, 500)
    place_pos = place_pos + 100
  WEND
```

12.6. See Also:

SYNC_CONTROL, SYNC_TIMER, USER_FRAME, USER_FRAMEB

13. SYNC_CONTROL

13.1. Type:

Axis parameter (Read Only)

13.2. Description:

SYNC_CONTROL returns the current SYNC state of the axis

13.3. Parameters:

value: 0 = No synchronisation

1 = Starting synchronisation

2 = Performing synchronisation movement

3 = Synchronised

4 = Stopping synchronisation

5 = Starting interpolated movement on second or third axis

6 = Performing interpolated movement on second or third axis

10 = Starting re- synchronisation

11 = Performing re- synchronisation

20 = Starting re-synchronisation to a different USER_FRAME

21 = Performing re-synchronisation to a different USER_FRAME

13.4. Example:

Synchronise to a conveyor linking to a position defined from registration, then wait until synchronisation before picking a part

```
'Set up start position and link to conveyor
  SYNC(10, 500, REG_POS AXIS(5), 5) AXIS(0)
  WAIT UNTIL SYNC_CONTROL AXIS(0)= 3
  GOSUB pick_part
```

13.5. See Also:

SYNC

14. SYNC_TIMER

14.1. Type:

Axis parameter (Read Only)

14.2. Description:

SYNC_TIMER returns the elapsed time of the synchronisation or re-synchronisation phase. Once the synchronisation is complete the SYNC_TIMER will return the completed synchronisation time.

14.3. Parameters:

value: The elapsed time of the synchronisation phase in milliseconds

14.4. Example:

Synchronise to a conveyor linking to a position defined from registration, then wait until synchronisation before picking a part

```
'Set up start position and link to conveyor
  SYNC(10, 500, REG_POS AXIS(5), 5) AXIS(0)
  WAIT UNTIL SYNC_TIMER AXIS(0)= 500
  GOSUB pick_part
```

14.5. See Also:

SYNC

15. TOOL_OFFSET

15.1. Type:

Axis Parameter

15.2. Syntax

TOOL_OFFSET(identity, x_offset, y_offset, z_offset)

15.3. Description:

TOOL_OFFSET can be used to adjust the position of a coordinate system to align with a tool point. Multiple tool points can be assigned and the user can switch between points on the fly.

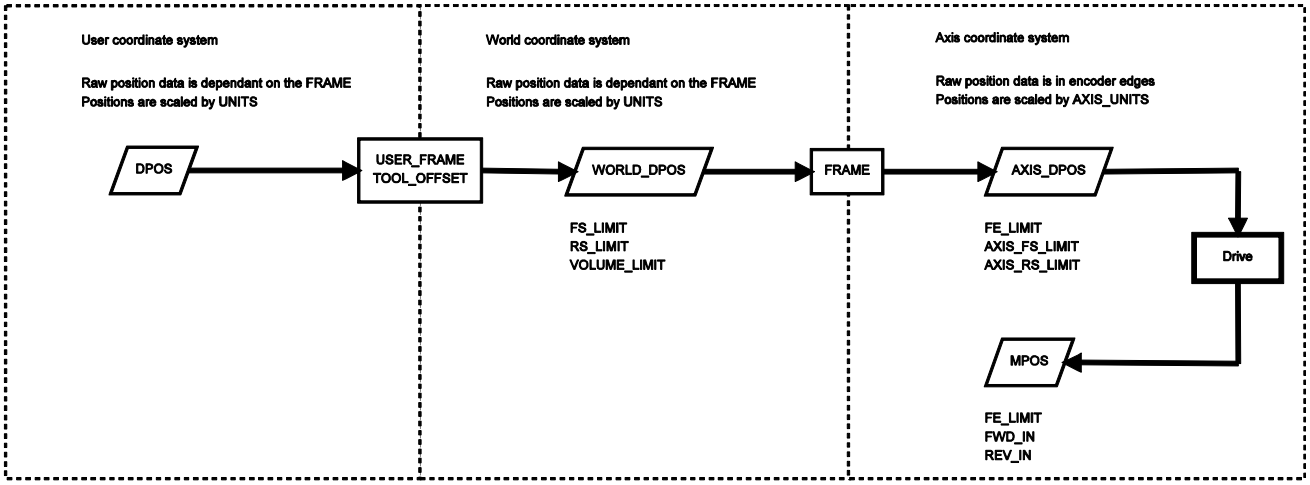
Note:

TOOL_OFFSET requires the kinematic runtime FEC

The default TOOL_OFFSET has the identity 0 and is equal to the world coordinate system origin, this cannot be modified. If you wish to disable the TOOL_OFFSET select TOOL_OFFSET(0).

TOOL_OFFSETs are applied on the axis FRAME_GROUP. If no FRAME_GROUP is defined then a runtime error will be generated.

Movements are loaded with the selected TOOL_OFFSET. This means that you can buffer a sequence of movements on different tools. The active TOOL_OFFSET is the one associated with the movement in the MTYPE. If the FRAME_GROUP is IDLE then the active TOOL_OFFSET is the selected TOOL_OFFSET.



Tip:

If you wish to check which USER_FRAME, TOOL_OFFSET and VOLUME_LIMIT are active you can print the details using FRAME_GROUP(group).

15.4. Parameters

identity:	0 = default group which is set to the world coordinate system
	1 to 31 = Identification number for the user defined tool offset.
x_offset:	Offset in the x axis from the world origin to the user origin.
y_offset:	Offset in the y axis from the world origin to the user origin.
z_offset:	Offset in the z axis from the world origin to the user origin.

15.5. Example

A tool is rotated 45degrees about the y axis and has an offset of 20mm in the x direction, 30mm in the y direction and 300mm in the z direction. The programmer wants to move the tool forward on its axis so a TOOL_OFFSET is applied to adjust the position to the tool tip, then a USER_FRAME is applied to allow programming about the tool axis.

```
'Configure USER_FRAME and TOOL_OFFSET
FRAME_GROUP(0,0,0,1,2)
USER_FRAME(1, 20, 30, 300, 0, PI/4, 0)
TOOL_OFFSET(1, 20, 30, 300)
'Select tool and frame and start motion.
USER_FRAME(1)
TOOL_OFFSET(1)
BASE(2)
FORWARD
```

16. USER_FRAME

16.1. Type:

Axis Parameter

16.2. Syntax

USER_FRAME(identity [, x_offset, y_offset, z_offset [, x_rotation [, y_rotation [, z_rotation]]]])

16.3. Description:

The USER_FRAME allows the user to program in a different coordinate system. The USER_FRAME can be defined up to a 3-axis translation and rotation from the world coordinate origin. The rotations are applied using the Euler ZYX convention. This means that the z rotation is applied first, then the y is applied on the new coordinate system and finally the x is applied. The coordinate system is defined using the 'right hand rule' and the rotation of the origin is defined using the 'right hand turn'.

Note:

USER_FRAME requires the kinematic runtime FEC

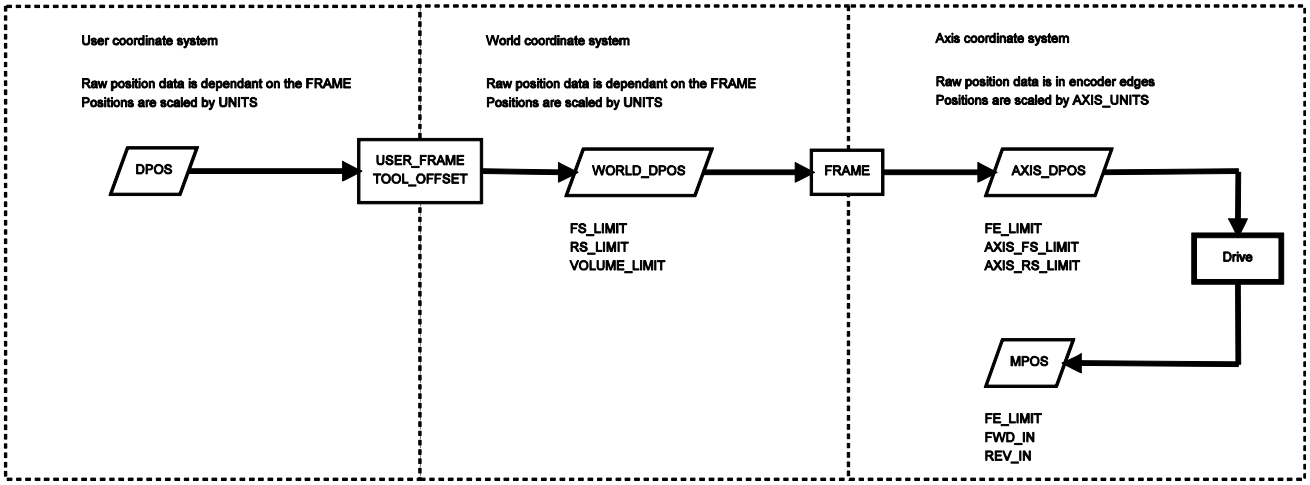
The default coordinate system has the identity 0 and is equal to the world coordinate system, this cannot be modified. If you wish to disable the USER_FRAME select USER_FRAME(0).

USER_FRAMEs are applied on the axis FRAME_GROUP. If no FRAME_GROUP is defined then a runtime error will be generated.

Movements are loaded with the selected USER_FRAME. This means that you can buffer a sequence of movements on different USER_FRAMES. The active USER_FRAME is the one associated with the movement in the MTYPE. If the FRAME_GROUP is IDLE then the active USER_FRAME is the selected USER_FRAME.

Note:

The USER_FRAME is applied to all the axes in the FRAME_GROUP. This can be the same group as used by FRAME. The FRAME_GROUP does not have to be 3 axis, however the USER_FRAME will only process position for the axes in the FRAME_GROUP. It can be useful in a 2 axes FRAME_GROUP to perform a USER_FRAME rotation about the third axis.



Tip:

If you wish to check which USER_FRAME, TOOL_OFFSET and VOLUME_LIMIT are active you can print

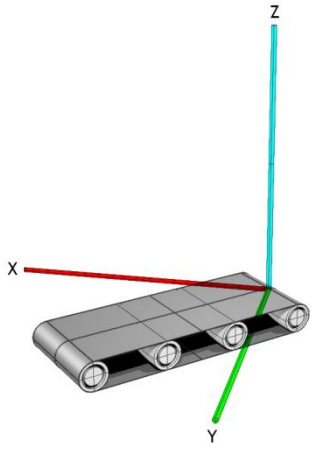
the details using FRAME_GROUP(group).

16.4. Parameters

identity:	0 = default group which is set to the world coordinate system
	1 to 31 = Identification number for the user defined frame.
x_offset:	Offset in the x axis from the world origin to the user origin.
y_offset:	Offset in the y axis from the world origin to the user origin.
z_offset:	Offset in the z axis from the world origin to the user origin.
x_rot:	Rotation about the items x axis in radians.
y_rot:	Rotation about the items y axis in radians.
z_rot:	Rotation about the items z axis in radians.

16.5. Example 1:

A conveyors origin is at 45degrees to the world coordinate (robots) origin, as shown in the image. To ease programming a USER_FRAME is assigned to align the x axis with the conveyor so that it is possible to program in the conveyor coordinate system.




```
FRAME_GROUP(0,0,0,1,2)
USER_FRAME(1,0,0,0,PI/4)
```

16.6. Example 2

Initialise a user coordinate system then perform a movement on the world coordinate system before starting a FORWARD on the first user coordinate system.

```
FRAME_GROUP(0,0,0,1,2)
BASE(0,1,2)
DEFPOS(10,20,30)
USER_FRAME(1,10,20,30,PI/2)
USER_FRAME(0)
MOVEABS(100,100,50)
WAIT IDLE
USER_FRAME(1)
FORWARD
```

17. USER_FRAMEB

17.1. Type:

Axis Parameter

17.2. Syntax

USER_FRAME(identity)

17.3. Description:

USER_FRAMEB is only used with SYNC. It defines the new USER_FRAME to resynchronise to when performing the SYNC(20) operation. When the resynchronisation is complete USER_FRAMEB is the active USER_FRAME. USER_FRAMEB selects one of the defined USER_FRAMEs.

17.4. Example:

The robot must pick up the components from one conveyor and place them on a second conveyor which is in a different USER_FRAME.

```
WHILE(running)
  USER_FRAMEB(conv1)
  REGIST(20,0,0,0,0) AXIS(10)
  WAIT UNTIL MARK AXIS(10)

  SYNC(1, 1000, REG_POS, 10, sen_xpos , conv1_yoff)
  WAIT UNTIL SYNC_CONTROL AXIS(0)=3
  'Now synchronised
  GOSUB pick

  USER_FRAMEB(conv2)
  SYNC(20, 1000, place_pos, 11, conv2_xoff, conv2_yoff)
  WAIT UNTIL SYNC_CONTROL AXIS(0)=3
  'Now synchronised
  GOSUB place

  SYNC(4, 500)
  place_pos = place_pos + 100
WEND
```

17.5. See Also:

SYNC, USER_FRAME

18. USER_FRAME_TRANS

18.1. Type:

Mathematical Function

18.2. Syntax:

USER_FRAME_TRANS(user_frame_in, user_frame_out, tool_offset_in, tool_offset_out, table_in, table_out, [scale])

18.3. Description:

This function enables you to transform a set of positions from one frame to another. This could be used to take a set of positions from a vision system and transform them so that they are a set of positions relative to a conveyor.

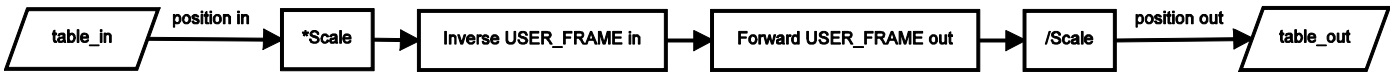
Note:

USER_FRAME_TRANS requires the kinematic runtime FEC

It is required to set-up a FRAME_GROUP and USER_FRAME to use this function. If you do not wish to set up a USER_FRAME with real axis you can use virtual.

Tip:

The USER_FRAME calculations are performed on raw position data which are integers. The table data is scaled by the scale parameter, for optimal resolution scale should be set to the UNITS of the robot.



Note:

As all the USER_FRAME transformations use the same coordinate scale it does not matter if the positions are supplied as raw positions or scaled by UNITS.

18.4. Parameters:

user_frame_in:	The USER_FRAME identity that the points are supplied in
user_frame_out:	The USER_FRAME identity that the points are transformed to
tool_offset_in:	The TOOL_OFFSET identity that the points are supplied in
tool_offset_out:	The TOOL_OFFSET identity that the points are transformed to
table_in:	The start of the input positions
table_out:	The start of the generated positions
scale:	This parameter allows you to scale the table values (default 1000)

18.5. Example:

USER_FRAME(vision) has been configured to the vision system relative to the robot origin. The conveyor has been configures in USER_FRAME(conveyor). To use the vision system positions on the conveyor USER_FRAME they must be transformed through USER_FRAME_TRANS.

```
USER_FRAME_TRANS(vision, conveyor, 0, 0, 200,300)
```

19. VOLUME_LIMIT

19.1. Type:

Axis Function

19.2. Syntax:

VOLUME_LIMIT(mode, [,table_offset])

19.3. Description:

VOLUME_LIMIT enables a software limit that restricts the motion into a defined three dimensional shape. The calculations are performed on DPOS and so it can be used in addition to a FRAME. The limit applies to axes defined in a FRAME_GROUP.

Note:

VOLUME_LIMIT requires the kinematic runtime FEC

Warning:

If no FRAME_GROUP is defined then a 'parameter out of range' run time error will be returned when VOLUME_LIMIT is called.

All axes in the FRAME_GROUP must have the same UNITS

When the limit is active moves on all axes in the FRAME_GROUP are cancelled and so will stop with the programmed DECEL or FAST_DEC. Any active SYNC is also stopped. AXISSTATUS bit 15 is also set. This means you should set your VOLUME_LIMIT smaller than the absolute operating limits of the robot.

19.4. Parameters:

mode:	0 = VOLUME_LIMIT is disabled 1 = Cylinder with cone base volume

---

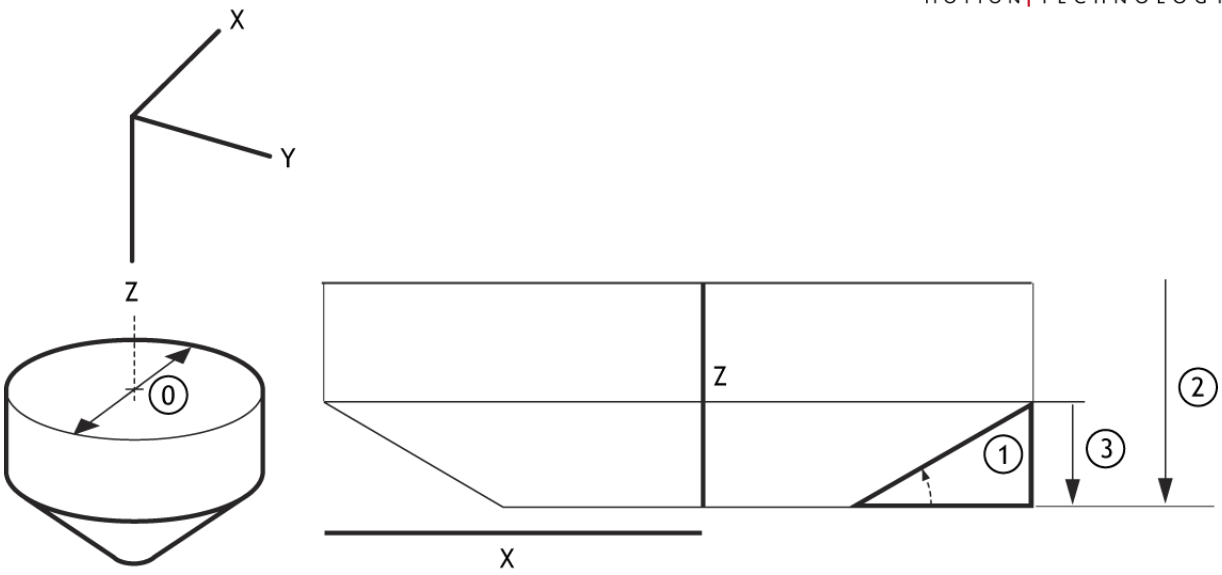
19.5. Syntax:

VOLUME_LIMIT(1, [,table_offset])

19.6. Description:

Mode 1 enables a cylinder with a cone base, this is a typical working volume for a delta robot.

The origin for the shape is the top middle. It is possible to align this with your coordinate system using the X,Y and Z offsets



Tip:

If you wish to check which USER_FRAME, TOOL_OFFSET and VOLUME_LIMIT are active you can print the details using FRAME_GROUP(group).

19.7. Parameters:

mode: 0 = VOLUME_LIMIT is disabled
1 = Cylinder with cone base volume

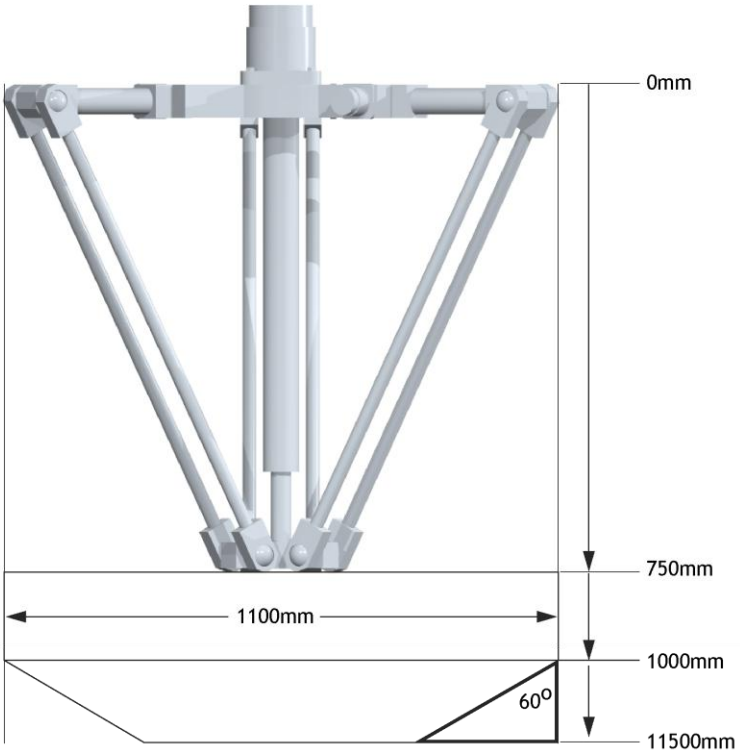
table_offset: The start position in the table to store the VOLUME_LIMIT configuration

Mode 0 table values, all liner values use UNITS from the first axis in the FRAME_GROUP.

0 = Cylinder Diameter
1 = Cone angle in radians
2 = Total height
3 = Cone height
4 = X offset
5 = Y offset
6 = Z offset

19.8. Example:

The cylinder with a flat base is typically used with delta robots (FRAME=14), the following example configures the VOLUME_LIMIT with this configuration.




```
TABLE(100,1100)' Cylinder diameter
TABLE(101,(60/360)* 2* PI)' Cone angle
TABLE(102,400)' Total height
TABLE(103,150)' Cone height
TABLE(104,0)' X offset
TABLE(105,0)' Y offset
TABLE(106,750)' Z offset

VOLUME_LIMIT(1,100)
```