APPLICATION NOTE YASKAWA

Title: Conveyor Tracking Setup	
Product(s): MP3300iec-RBT, MP3200iec-RBT, MotionWorks IEC	**Doc. No.** AN.MP3300iec.03

Table of Contents

Application Overview 2
Products Used 2
Implementation 2
Coordinate System Definitions 3
MC_TrackConveyorBelt Operation 4
Aborting Conveyor Tracking 4
Example Conveyor Tracking Pick and Place Sequence 5
Setup 8
- ☐ Photo Eye 8
- ☐ Example – Photo eye inside robot workspace 8
- ☐ Example – Photo eye outside robot workspace 14
- ☐ Vision 20
- ☐ Example – Camera inside robot workspace 20
- ☐ Example – Camera outside robot workspace 23

FAQ 27
- ☐ Motion Details 27

Title: Conveyor Tracking Setup

Product(s): MP3300iec-RBT, MP3200iec-RBT, MotionWorks IEC

Doc. No. AN.MP3300iec.03

Application Overview

This document explains how the MC_TrackConveyorBelt can be used. It contains common example setups to find the inputs InitialObjectPosition, ConveyorBeltOrigin, StartDistance, and EndDistance with a photo eye or 2D camera are explained with a step by step guide.

Products Used

Component	Product and Model Number
Controller	MP3300iec-RBT, MP3200iec-RBT, firmware 3.4 or higher
Software	MotionWorksIEC Pro, 3.4 or higher

Implementation

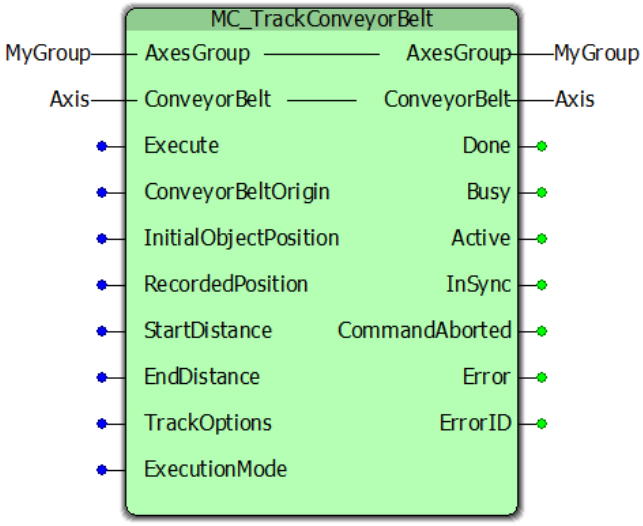

FIGURE 1 MC_TRACKCONVEYORBELT FUNCTION BLOCK

Title: Conveyor Tracking Setup

Product(s): MP3300iec-RBT, MP3200iec-RBT, MotionWorks IEC

Doc. No. AN.MP3300iec.03

Coordinate System Definitions

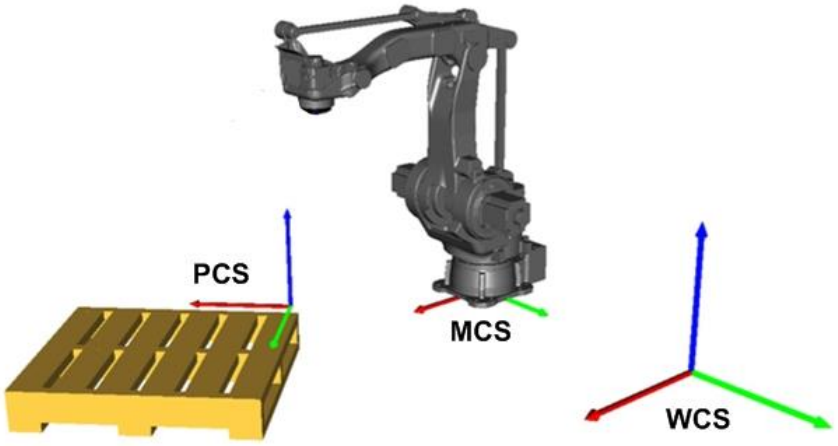

FIGURE 2 VISUALIZATION OF PCS MCS AND WCS

Coordinate System	ENUM	Long Description
Part Coordinate System	PCS	User defined frame, usually represents location and orientation of an object in the workcell. PCS are defined relative to WCS. PCS can be moving. Defined as [X Y Z Rx Ry Rz] offset from WCS and is stored in AXES_GROUP_REF.Part.Offset. Trajectory data relative to PCS is reported in AXES_GROUP_REF.Part. Data is reported as TCP trajectory relative to PCS origin.
Machine Coordinate System	MCS	Located at the base of the robot. Positions defined in MCS will be relative to the base of the robot instead of WCS. Same as Robot Frame for MLX. Trajectory data relative to MCS is reported in AXES_GROUP_REF.Machine. Data is reported as TCP trajectory relative to MCS origin
World Coordinate System	WCS	Base coordinate frame (defined in Cartesian space (X,Y,Z)) of the overall workcell. The position of any object in a workcell (robots, conveyors, pallets, parts, PCS definitions etc.) is defined relative to this frame. Same as World Frame for MLX. Defined by offset from MCS and is stored in AXES_GROUP_REF.World.Offset. Trajectory data relative to WCS is reported in AXES_GROUP_REF.World. Data is reported as TCP trajectory relative to WCS origin.

MC_TrackConveyorBelt Operation

MC_TrackConveyorBelt does the following:

- Tracks a PCS defined by InitialObjectPosition, ConveyorBeltOrigin, RecordedPosition, and current Conveyor position.
- On execute, the PCS is set at the position defined by InitialObjectPosition, ConveyorBeltOrigin, and StartDistance.
- Tracking becomes active after ExecutionMode is satisfied, and the origin of InitialObjectPosition passes StartDistance.
- The PCS definition will be moving, since it is tracking a moving PCS. PCS definition can be seen in AXES_GROUP_REF.Part.Offset
- PCS moves executed after MC_TrackConveyorBelt will be relative to the moving PCS.
- Tracking can be aborted by executing an aborting MCS move, when a buffered MCS move becomes active, or MC_GroupStop.
- MCS moves executed BEFORE MC_TrackConveyorBelt execute will be superimposed with tracking motion.
- MCS moves executed AFTER MC_TrackConveyorBelt execute will abort tracking when it becomes active.
- In normal operation, tracking is aborted by user before InitialObjectPosition reaches EndDistance.
- By default, tracking will finish and report done when InitialObjectPosition reaches EndDistance if it is not aborted by other means.

Aborting Conveyor Tracking

Conveyor Tracking will be aborted if any of the following occurs:

- MC_GroupStop
- MCS move that was executed after MC_TrackConveyorBelt becomes active
  - Aborting MCS move will abort tracking immediately since it becomes active on execute
  - Blended/Buffered MCS moves executed after MC_TrackConveyorBelt will abort tracking when the MCS move becomes active in the motion queue

Example Conveyor Tracking Pick and Place Sequence

A simple application would be to pick a part from a conveyor belt, then place on a static position. The setup of the MC_TrackConveyorBelt function block inputs will be discussed in the next section. This would be a good sequence to start with:

1. Start from a safe position

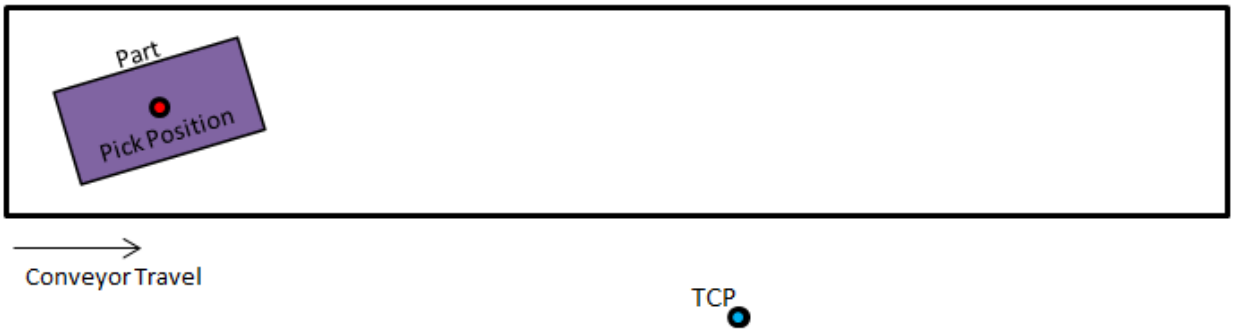

FIGURE 3 START OF PICK AND PLACE SEQUENCE, TCP STARTING FROM SAFE POSITION

2. Execute MC_TrackConveyorBelt
3. Execute blended MC_MoveLinearAbsolute with CoordSystem PCS to a hover position above part when MC_TrackConveyorBelt becomes active

FIGURE 4 TCP IS COMMANDED TO HOVER POSITION OVER PART WHILE SYCNING INTO CONVEYOR

4. Execute blended MC_MoveLinearAbsolute with CoordSystem PCS down to the part when tracking is InSync

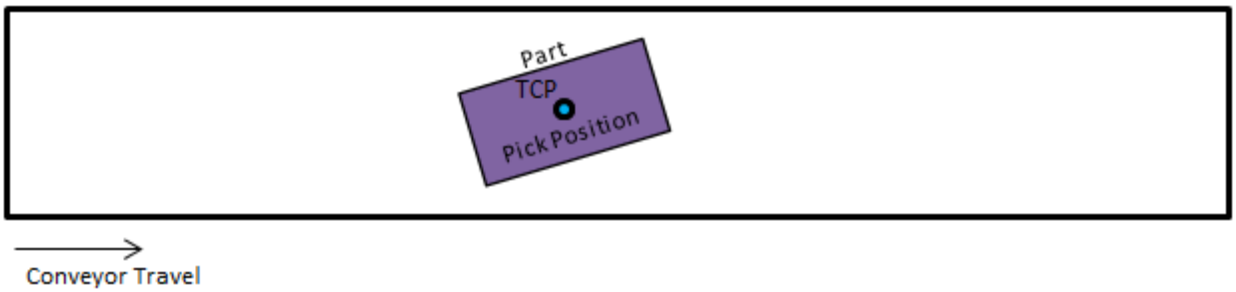

FIGURE 5 TCP IS SYNCED INTO PART, AND REACHED PICK POSITION AND HOVER POSITION

5. Execute blended MC_MoveLinearAbsolute with CoordSystem PCS up to hover above part
6. Execute blended MC_MoveLinearAbsolute with CoordSystem MCS to hover position above the place position. Conveyor tracking will be aborted when this move becomes active

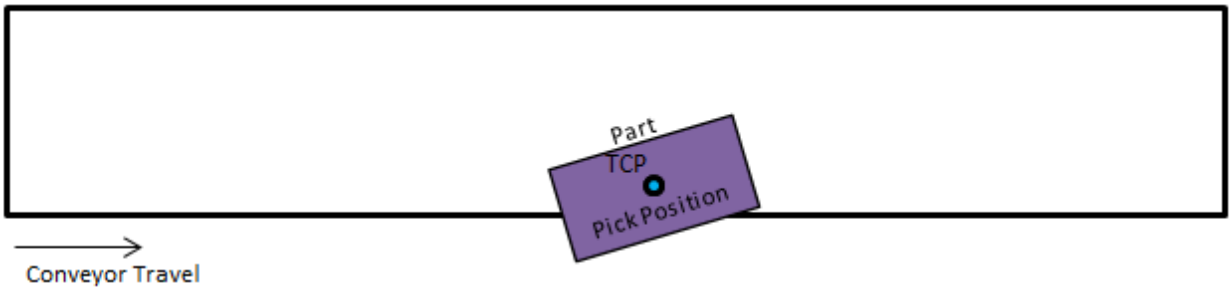

FIGURE 6 PART IS PICKED AND IS MOVING TOWARD PLACE POSITION

7. Execute blended MC_MoveLinearAbsolute with CoordSystem MCS down to place position

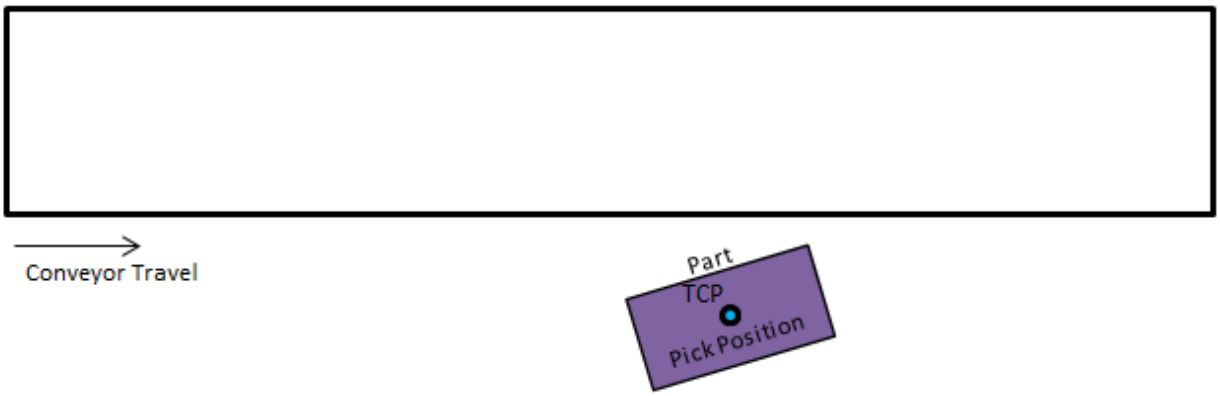

FIGURE 7 PART ARRIVES AT PLACE POSITION

8. Execute blended MC_MoveLinearAbsolute with CoordSystem MCS up to hover above place position
9. Loop to step 2 for new part

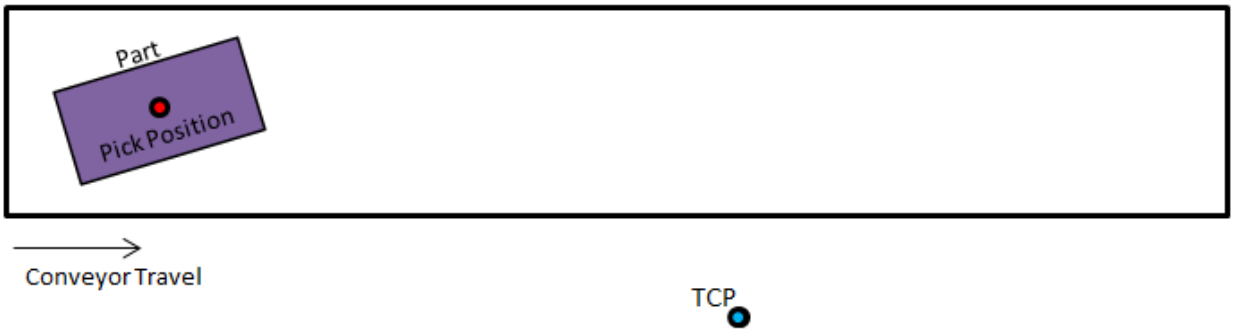

FIGURE 8 REPEAT FROM STEP 2 WITH A NEW PART

Setup

Getting values for **ConveyorBeltOrigin**, **InitialObjectPosition, StartDistance** and **EndDistance** may be slightly different procedurally depending on hardware set up. The following are a few common examples.

ConveyorBeltOrigin is generally a frame that needs to be taught using the mechanism/robot being used. The recommended way is to define a frame using 3 points relative to WCS (OO – Origin, XX – Point on the +X axis, XY – Point on the XY plane), then use function block **FrameTypeTransform** to convert it to a 6 value VECTOR definition for input into MC_TrackConveyorBelt. This method is used in all the following examples.

☐ Photo Eye

Photo eye is a sensor that reports back a BOOL value when it is triggered such as a proximity sensor. Since this is only gives information about whether a part is present, InitialObjectPosition should be a fixed value. A calibration part can be used to teach ConveyorBeltOrigin, and is typically just the part that needs to be picked.

☐ Example – Photo eye inside robot workspace

This example places ConveyorBeltOrigin INSIDE the robot workspace. In this case, the point to track is in the middle of the part. It is convenient to define ConveyorBeltOrigin to be coincident with the pick point when the part triggers the photo eye. This way, InitialObjectPosition (the frame to be tracked) is always [0 0 0 0 0 0] relative to ConveyorBeltOrigin when the sensor is triggered. The PCS position that correspond to the pick point is also [0 0 0 0 0 0].

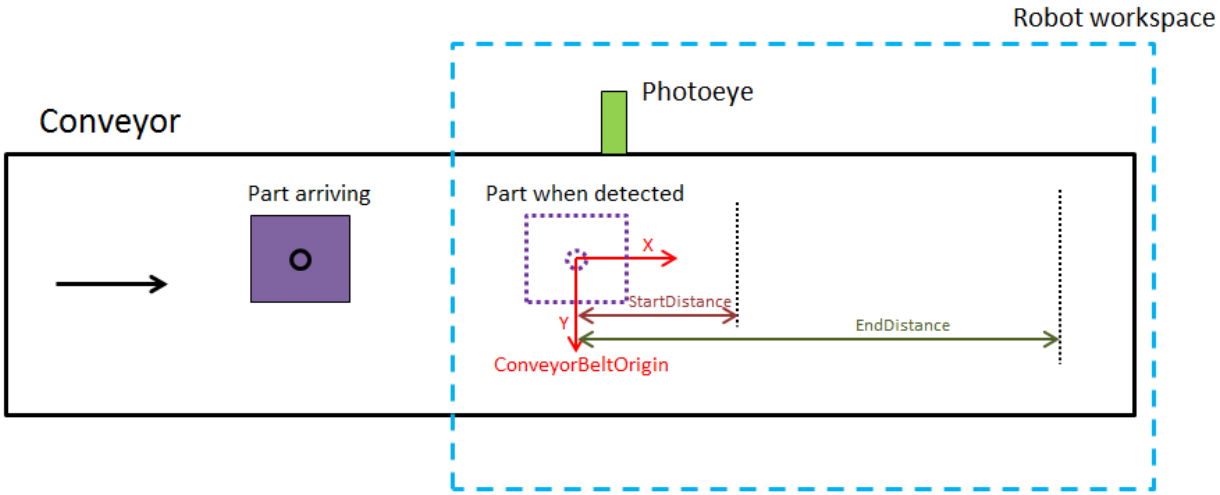

FIGURE 9 REPRESENTATION OF CONVEYOR TRACKING SET UP WHEN EXAMPLE IS FINISHED

1. Place calibration part upstream of photo eye in same location and orientation as expected operation. The pick point of the part is also noted. In this set up, the frame to be tracked is attached to the pick point.

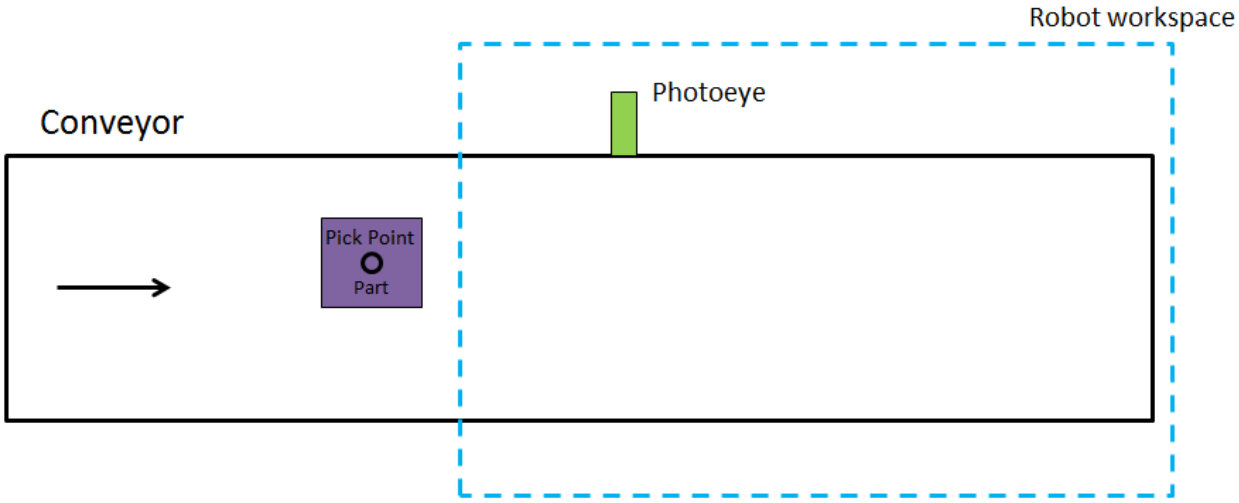

FIGURE 10 PART ARRIVING FROM OUTSIDE ROBOT WORKSPACE

2. Move Conveyor downstream until part triggers photo eye

3. Record ConveyorBeltOrigin, OO by moving the robot TCP to the pick point and recording the TCP position relative to WCS. This is the origin of ConveyorBeltOrigin

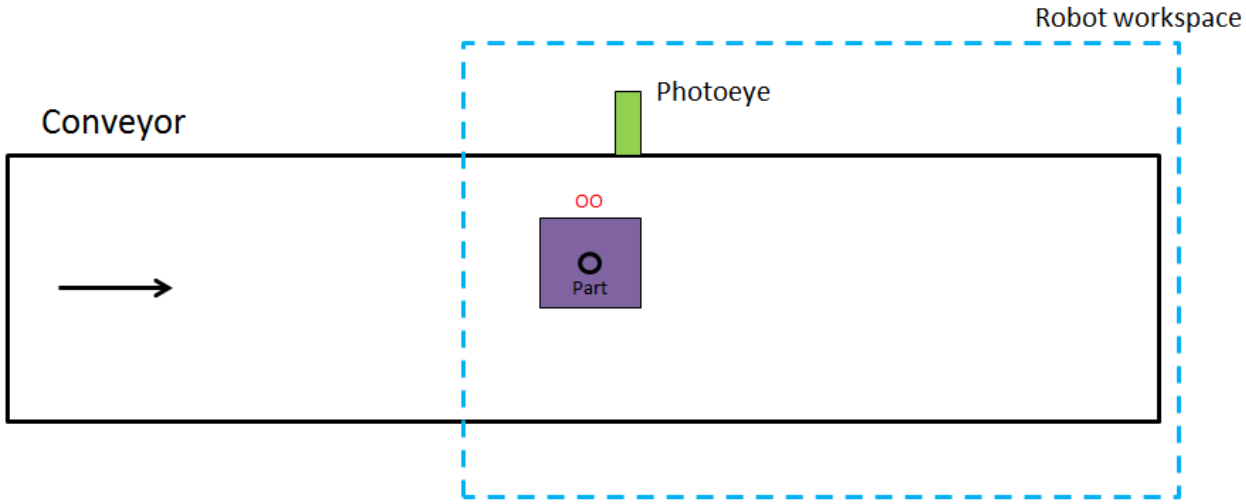

FIGURE 11 CONVEYOR MOVES PART DOWNSTREAM AND TRIGGERS PHOTOEYE AND REPRESENTS POINT OO

4. Find StartDistance and EndDistance by measuring from OO to where it is desired for conveyor tracking to start and end.

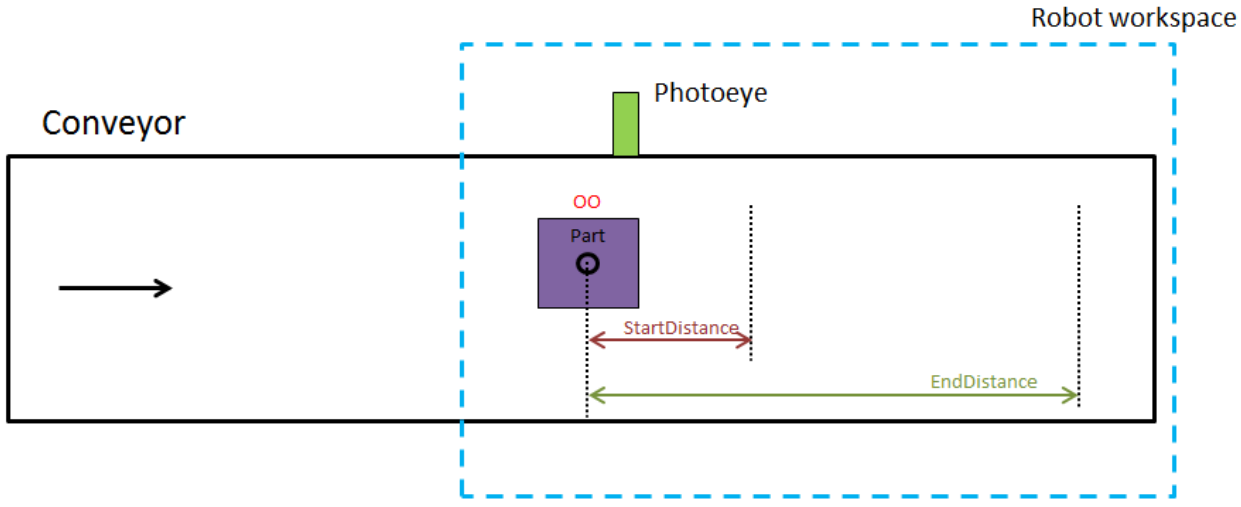

FIGURE 12 MEASUREMENTS OF STARTDISTANCE AND ENDDISTANCE

5. Move Conveyor downstream and stop with part still inside robot workspace

6. Record ConveyorBeltOrigin, XX by moving the robot TCP to the pick point and recording the TCP position relative to WCS. This is a point on the +X axis of ConveyorBeltOrigin

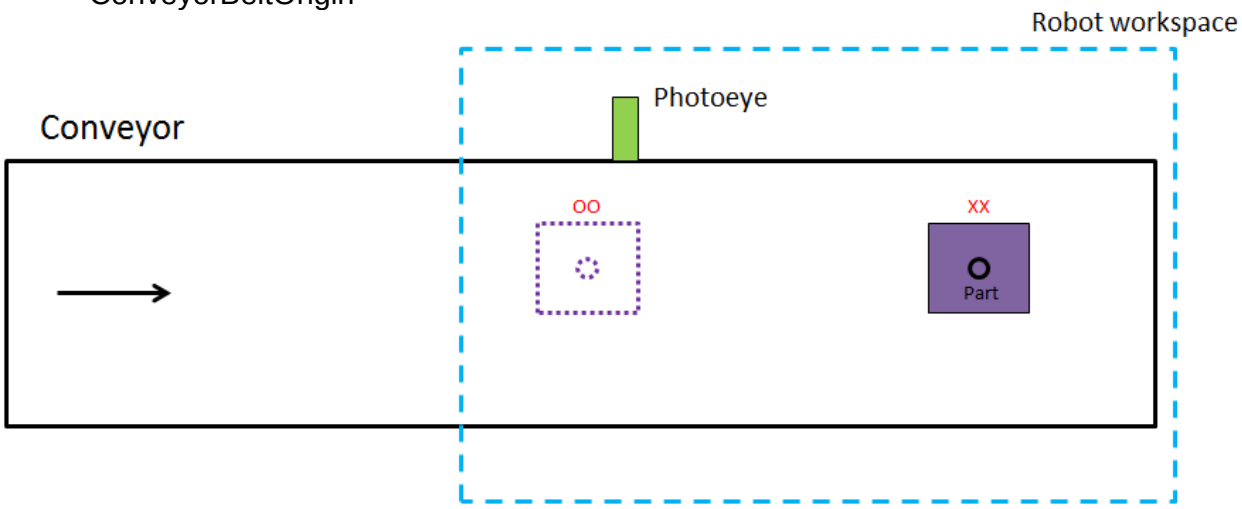

FIGURE 13 PART IS MOVED DOWNSTREAM USING CONVEYOR TO REPRESENT POINT XX

7. Record ConveyorBeltOrigin, XY by moving the block by hand to somewhere else on the conveyor belt then moving the robot TCP to the pick point and recording the TCP position relative to WCS. This is a point on the XY plane of ConveyorBeltOrigin.

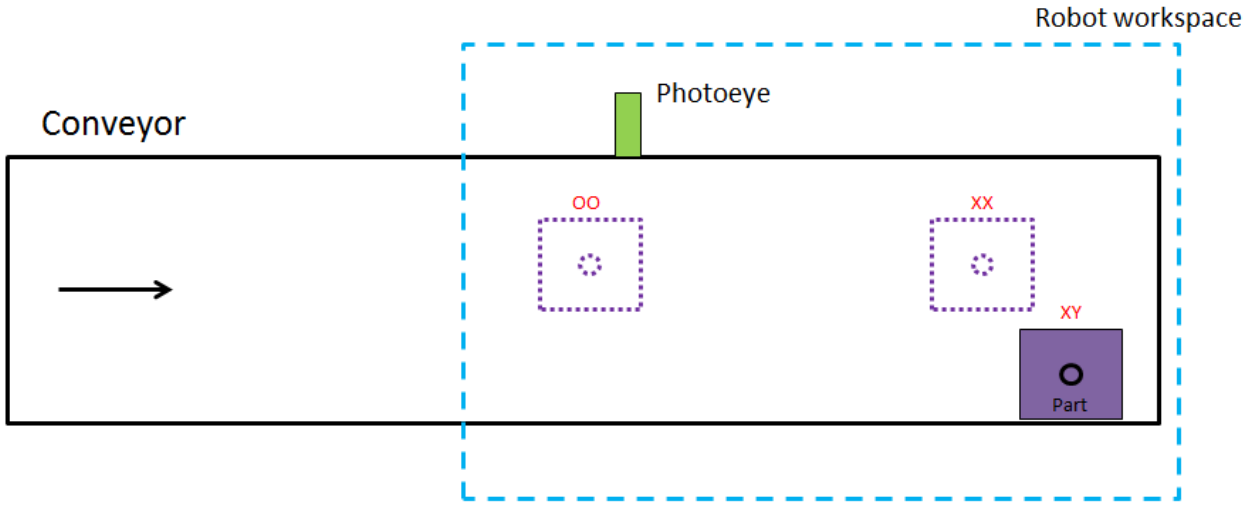

FIGURE 14 PART IS MOVED BY HAND TO DIFFERENT POSITION ON CONVEYOR TO REPRESENT POINT XY

8. Calculate ConveyorBeltOrigin using FrameTypeTransform, found in the Math_Toolbox

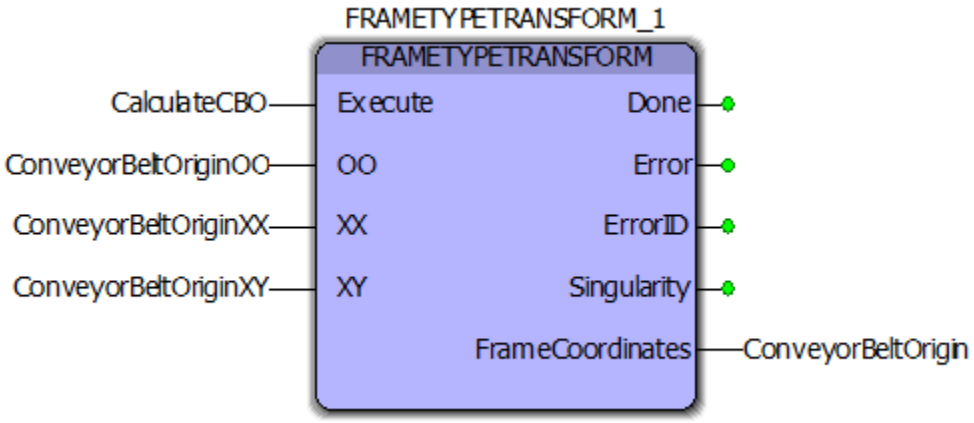

FIGURE 15 FUNCTION BLOCK IN MATH_TOOLBOX TO CALCULATE A FRAME USING OO, XX, AND XY

9. Find InitialObjectPosition. In this setup, InitialObjectPosition will always be [0 0 0 0 0 0] because the point to track is always coincident with ConveyorBeltOrigin when the sensor gets triggered.

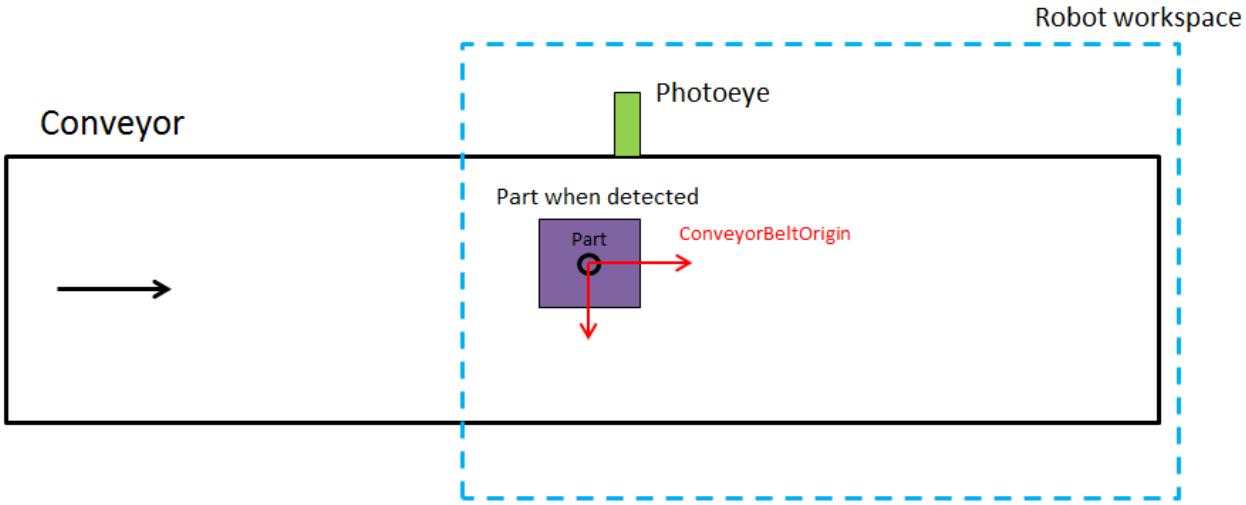

FIGURE 16 REPRESENTATION OF PART WHEN IT TRIGGERS PHOTOEYE. INITIALOBJECTPOSITION IN THIS SET UP IS [0 0 0 0 0 0]

Comments about the frame being tracked in this example set up:

The part triggers the sensor with its leading edge, and point that was recorded as OO is the middle of the part, which is also the X Y Z position of ConveyorBeltOrigin. So, ConveyorBeltOrigin is actually placed slightly upstream of the sensor when the leading edge triggers it. The InitialObjectPosition is set to [0 0 0 0 0 0] when the sensor is detected, that means the frame to track is at the same position as ConveyorBeltOrigin when detected, which is slightly upstream of the sensor and in the middle of the part. So, PCS position [0 0 0 0 0 0] should get to the pick position.

☐ Example – Photo eye outside robot workspace

This example places ConveyorBeltOrigin OUTSIDE the robot workspace. To define the ConveyorBeltOrigin outside the workspace, a temporary frame can be defined inside the workspace that is the same as the ConveyorBeltOrigin except offset downstream (+X direction) by a known amount. Then CalcFrameOffset can be used to calculate ConveyorBeltOrigin using the temporary frame and the known offset.

In this case, the point to track is in the middle of the part. It is convenient to define ConveyorBeltOrigin to be coincident with the pick point when the part triggers the photo eye. This way, InitialObjectPosition (the frame to be tracked) is always [0 0 0 0 0 0] relative to ConveyorBeltOrigin when the sensor is triggered. The PCS position that correspond to the pick point is also [0 0 0 0 0 0].

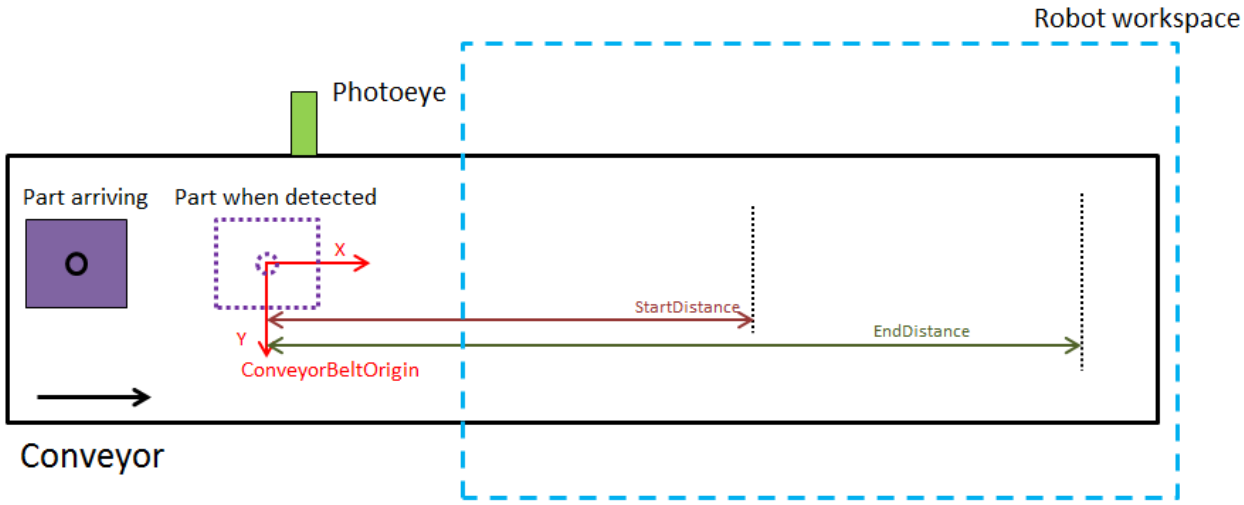

FIGURE 17 REPRESENTATION OF CONVEYOR TRACKING SETUP WHEN EXAMPLE IS FINISHED

1. Place calibration part upstream of photo eye in same location and orientation as expected operation. In this set up, the frame to be tracked is attached to the pick point.

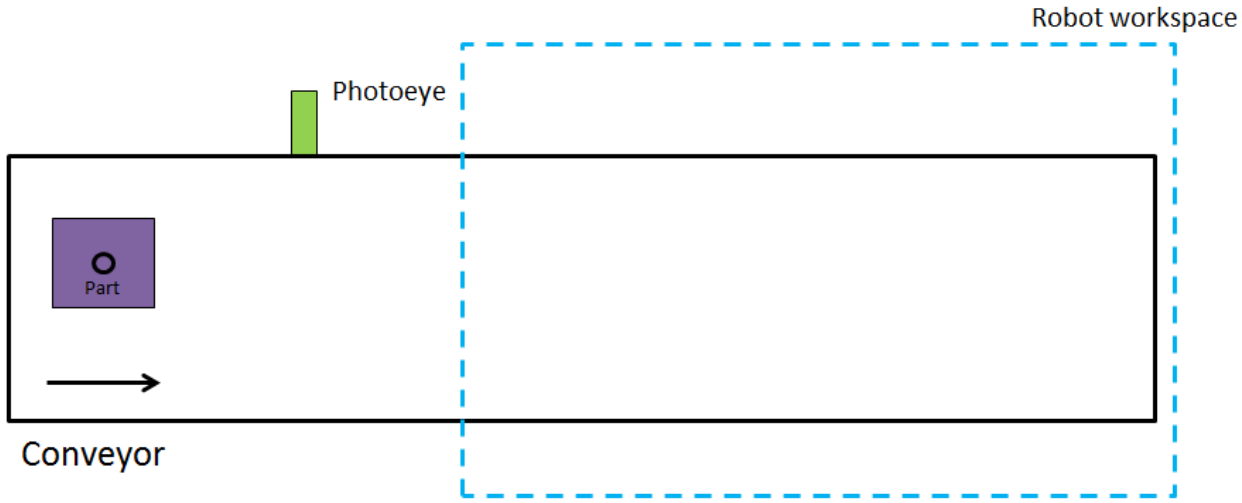

FIGURE 18 PART IS MOVING TOWARDS PHOTOEYE OUTSIDE OF ROBOT WORKSPACE

2. Move Conveyor downstream until part triggers photo eye
3. Record Conveyor Position 1

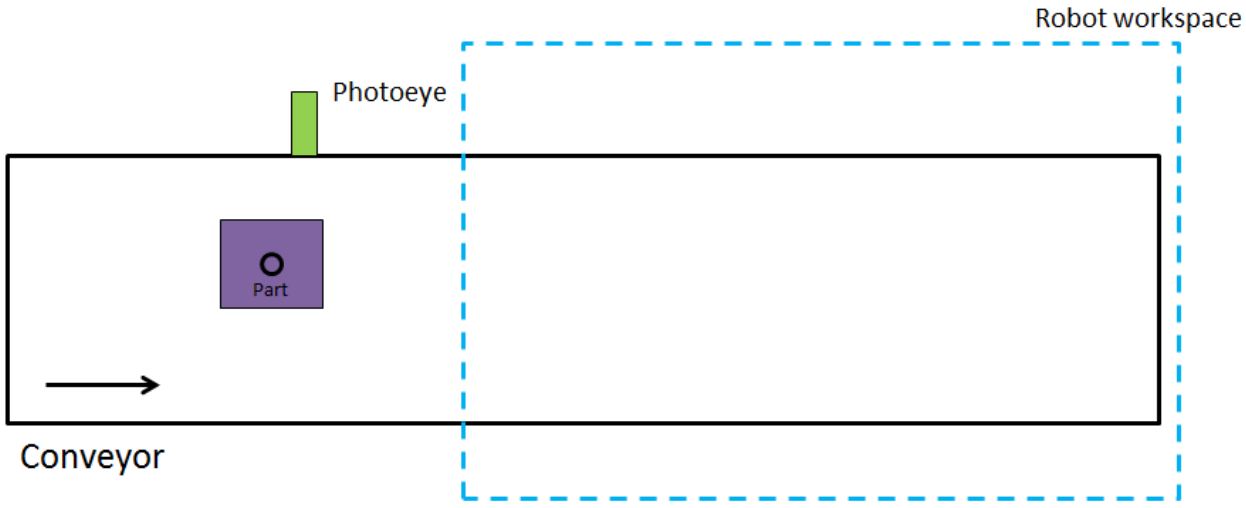

FIGURE 19 CONVEYOR MOVES PART DOWNSTREAM AND TRIGGERS PHOTOEYE

4. Find StartDistance and EndDistance by measuring from the pick point to where it is desired for conveyor tracking to start and end.

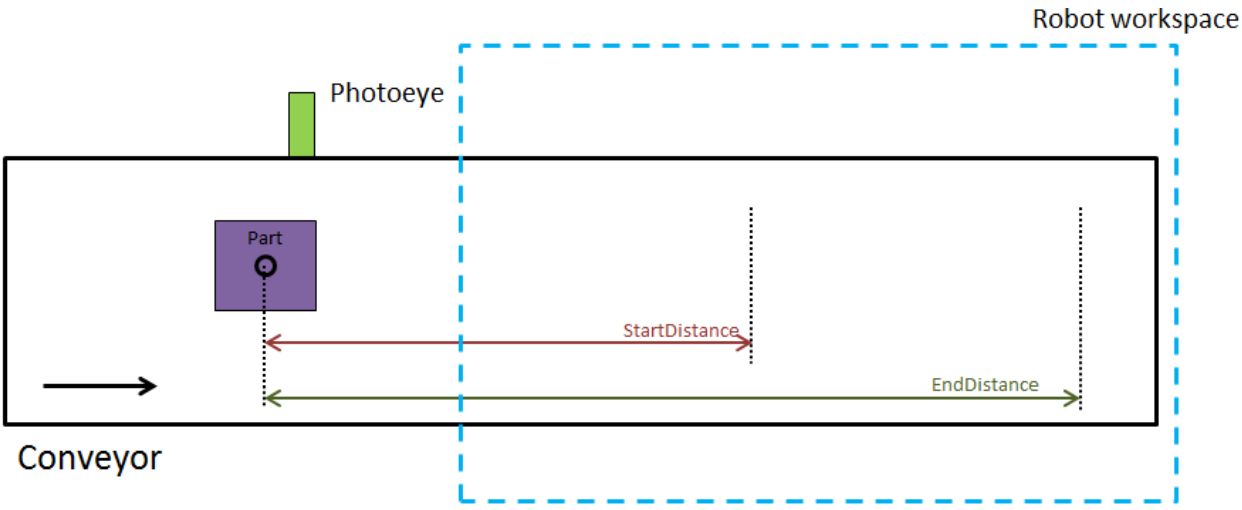

FIGURE 20 MEASUREMENTS OF STARTDISTANCE AND ENDDISTANCE

5. Move Conveyor downstream until part is inside robot workspace
6. Record Conveyor Position 2
7. Record Temp Frame, OO by moving the robot TCP to the pick point and recording the TCP position relative to WCS. This is the origin of a temporary frame that will be used to calculate ConveyorBeltOrigin.

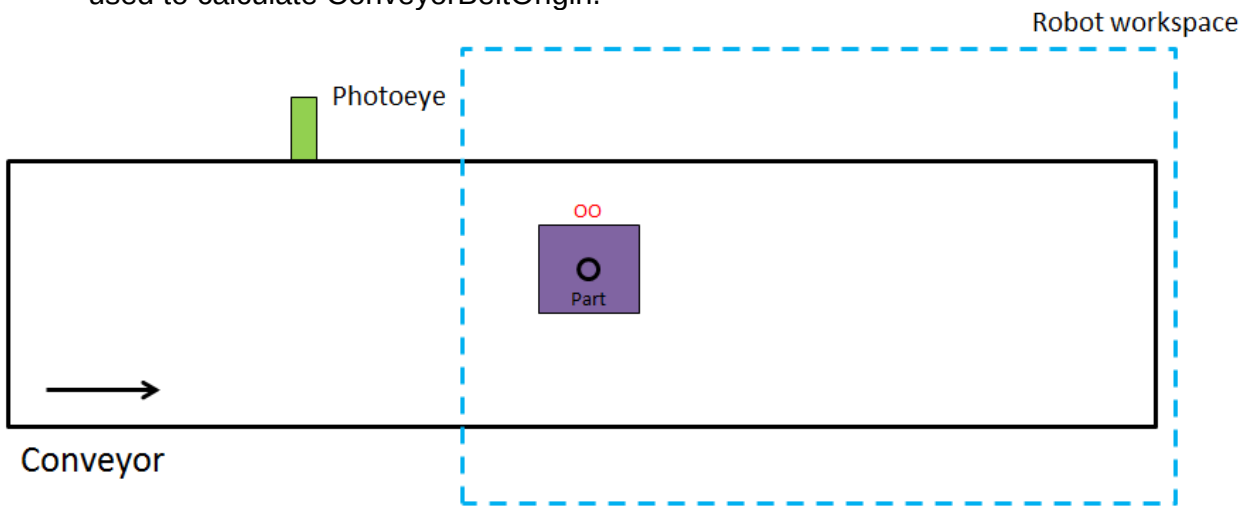

FIGURE 21 CONVEYOR MOVES PART DOWNSTREAM TO WITHIN ROBOT WORKSPACE TO REPRESENT POINT OO

8. Move Conveyor downstream and stop with part still inside robot workspace
9. Record Temp Frame, XX by moving the robot TCP to the pick point and recording the TCP position relative to WCS. This is a point on the +X axis of the temporary frame.

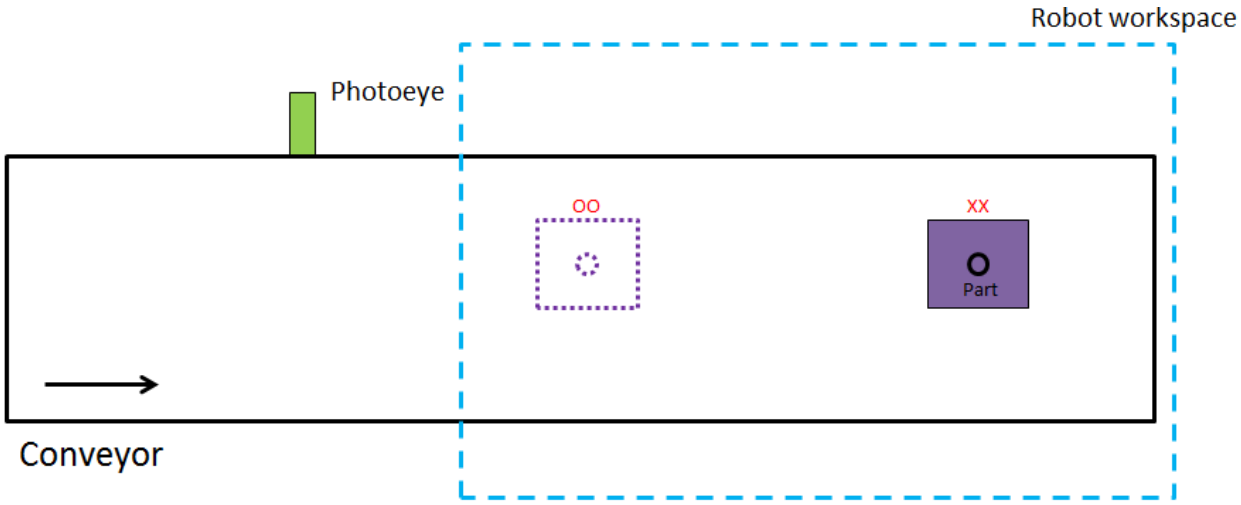

FIGURE 22 CONVEYOR MOVES PART DOWNSTREAM TO REPRESENT POINT XX

10. Record Temp Frame, XY by moving the block by hand to somewhere else on the conveyor belt then moving the robot TCP to the pick point and recording the TCP position relative to WCS. This is a point on the XY plane of the temporary frame.

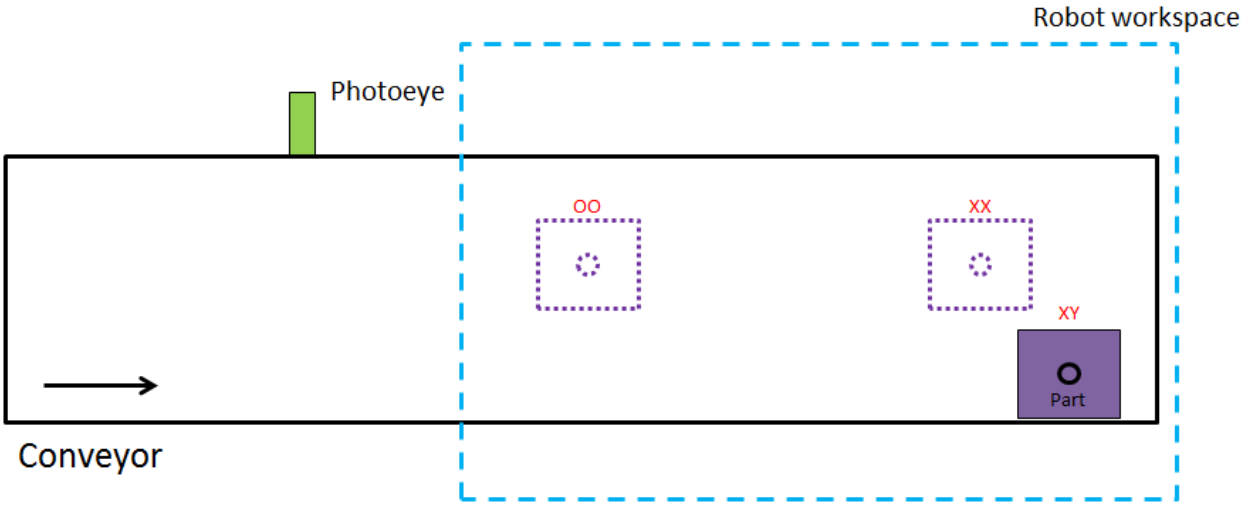

FIGURE 23 PART IS MOVED BY HAND TO NEW POSITION ON CONVEYOR TO REPRESENT POINT XY

11. Calculate ConveyorBeltOrigin using FrameTypeTransform and CalcFrameOffset, found in the Math_Toolbox. Here, Offset[1] = ConveyorPosition1 – ConveyorPosition2 (should be negative), Offset[2..6] = 0.

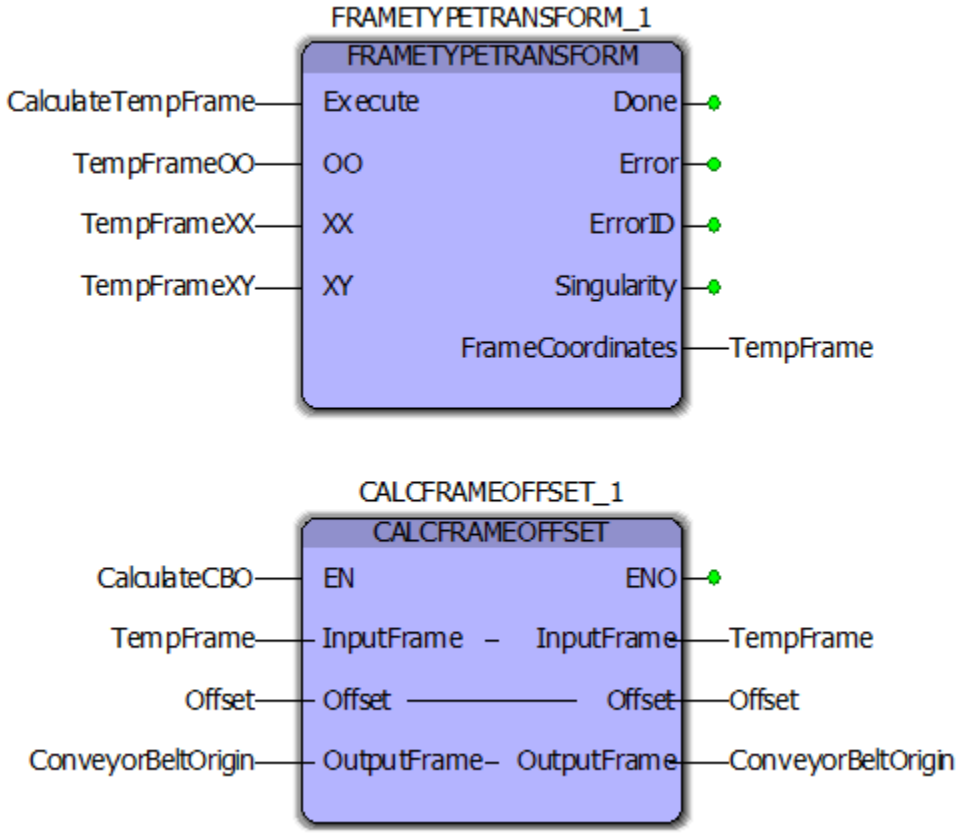

FIGURE 24 FRAMETYPETRANSFORM AND CALCFRAMEOFFSET FROM MATH_TOOLBOX IS USED TO CALCULATE CONVEYORBELTORIGIN

12. Find InitialObjectPosition. In this setup, InitialObjectPosition will always be [0 0 0 0 0 0] because the point to track is always coincident with ConveyorBeltOrigin when the sensor gets triggered.

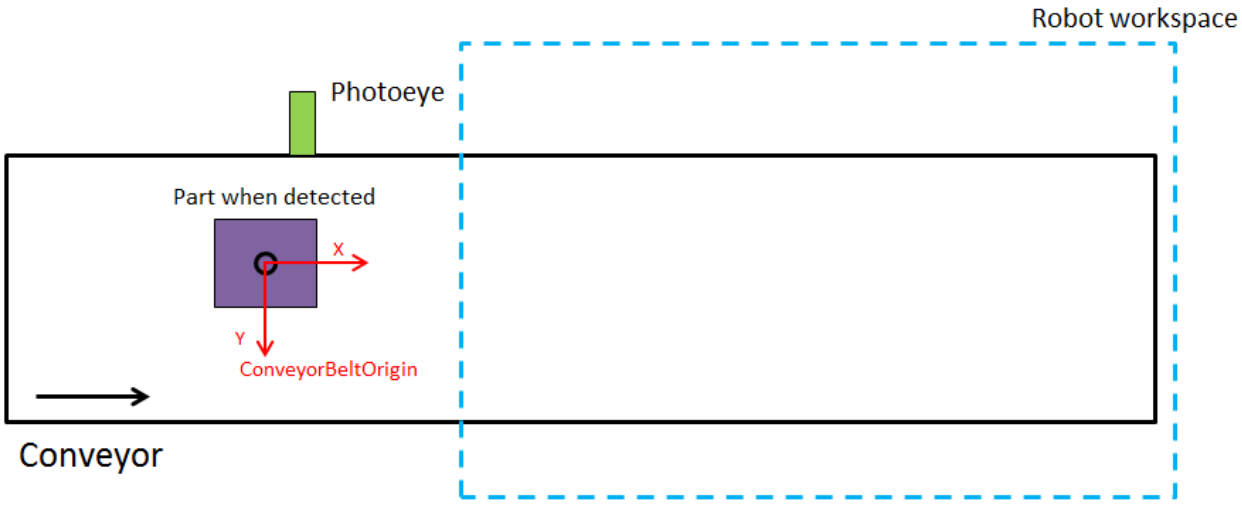

FIGURE 25 REPRESENTATION OF WHEN A PART TRIGGERS PHOTOEYE IN THIS SET UP

Comments about the frame being tracked in this example set up:

The part triggers the sensor with its leading edge, and point that was recorded as OO is the middle of the part, which is also the X Y Z position of ConveyorBeltOrigin. So, ConveyorBeltOrigin is actually placed slightly upstream of the sensor when the leading edge triggers it. The InitialObjectPosition is set to [0 0 0 0 0 0] when the sensor is detected, that means the frame to track is at the same position as ConveyorBeltOrigin when detected, which is slightly upstream of the sensor and in the middle of the part. So, PCS position [0 0 0 0 0 0] should get to the pick position.

☐ Vision

Vision is a sensor that reports up to 6DOF value that represents location and orientation of a single or multiple parts relative to a defined frame such as a 2D camera. This sensor gives location data of the part, so InitialObjectPosition should be a variable that depends on the sensor output. For example, a 2D camera might give X, Y and Rz value of a part. It is recommended that the vision sensor reports data relative to ConveyorBeltOrigin, which means vision data can be filled into InitialObjectPosition directly without any additional transformations. The vision system typically triggers at a set conveyor distance, for example every time the conveyor travels 300mm, the vision gets triggered to detect new parts.

When the camera is calibrated to output user units, typically a calibration grid such as a dot grid or square grid is used. The same grid can be used to teach ConveyorBeltOrigin. A common procedure for calibrating vision is to use the vision system to locate each dot on the grid, and then the user can specify the location of each dot in user units.

☐ Example – Camera inside robot workspace

This example places ConveyorBeltOrigin INSIDE the robot workspace. In this case, a dot grid is used. 3 dots on the grid (OO, XX, XY) are marked as points to be used to calculate ConveyorBeltOrigin. When calibrating the camera, OO is the origin, XX is a point along the +X axis, and XY is a point on the XY plane. Do not move the calibration grid after calibrating camera.

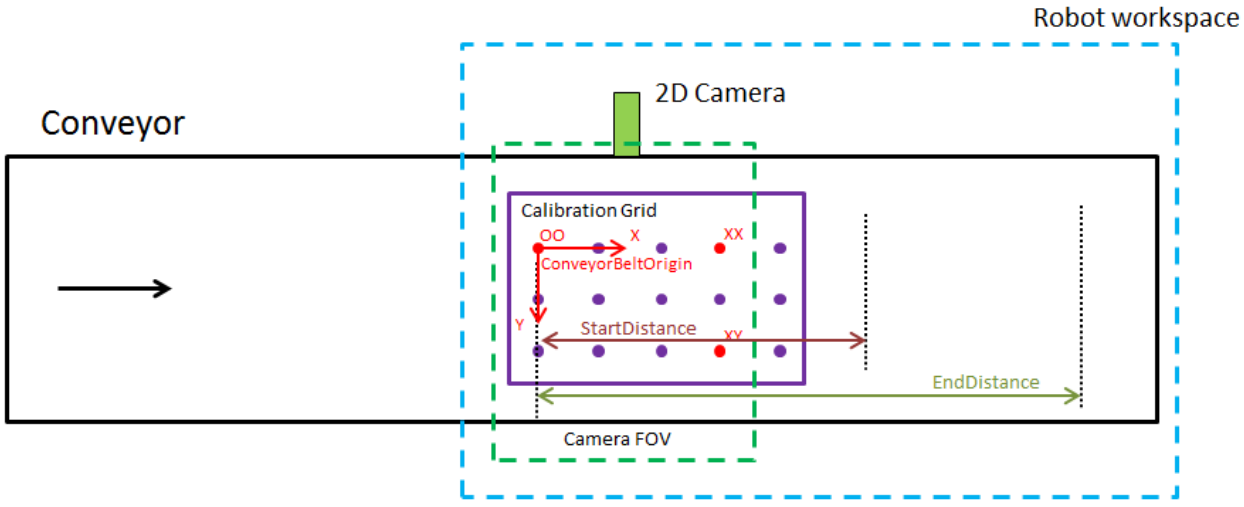

FIGURE 26 REPRESENTATION OF CONVEYOR TRACKING SETUP WHEN EXAMPLE IS FINISHED

1. Calibrate camera using vision calibration tool such that +X direction of the camera is in positive travel direction of conveyor.
2. Record position OO on the vision calibration tool by moving the robot TCP to the marked OO dot and recording the TCP position relative to WCS. This is the origin of ConveyorBeltOrigin.

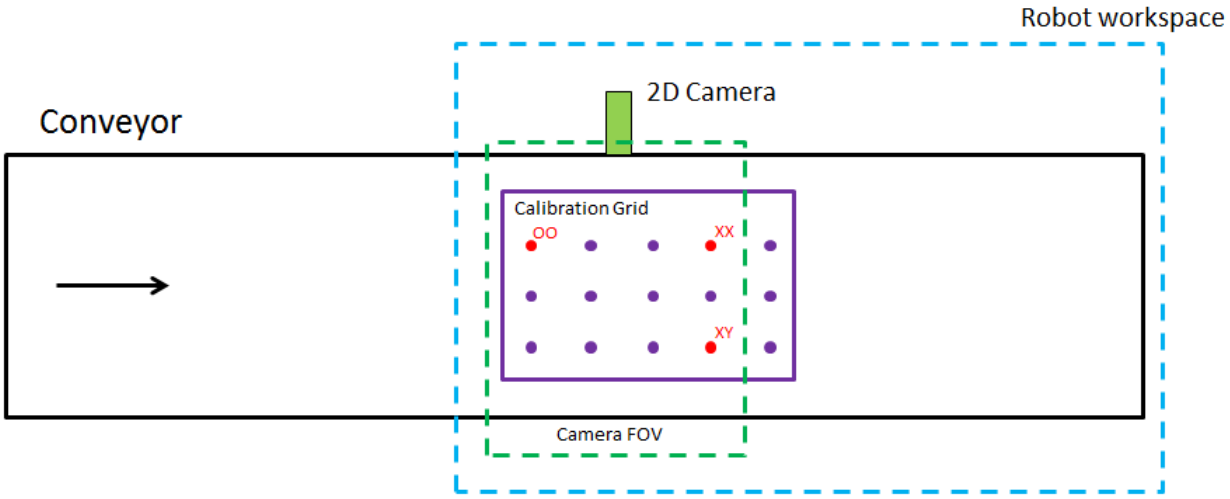

FIGURE 27 CALIBRATION GRID INSIDE ROBOT WORKSPACE AND REPRESENTATION OF POINT OO

3. Find StartDistance and EndDistance by measuring from OO to where it is desired for conveyor tracking to start and end.

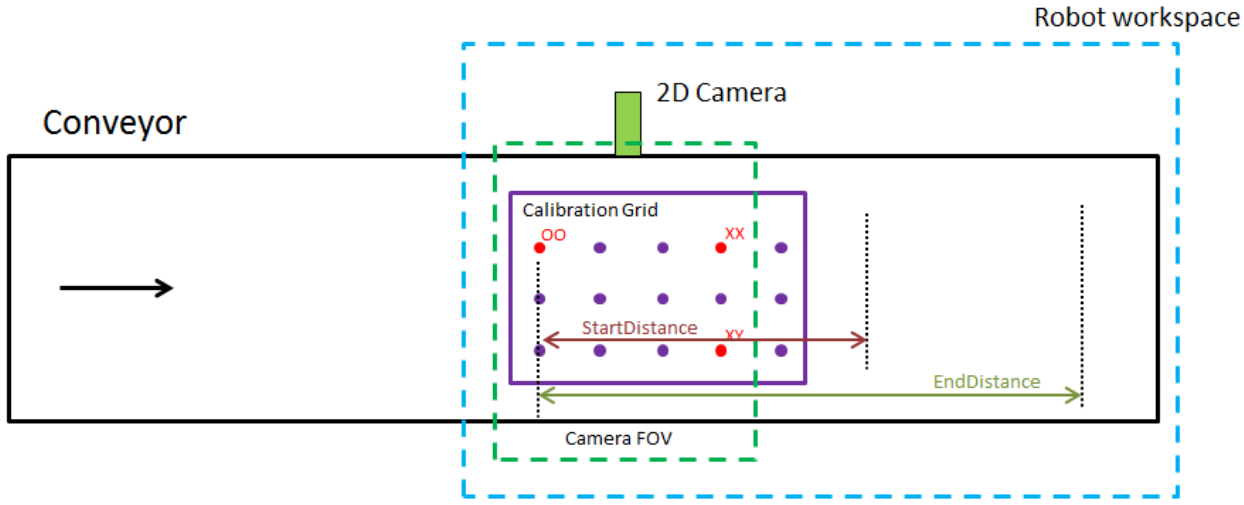

FIGURE 28 MEASUREMENTS OF STARTDISTANCE AND ENDDISTANCE

4. Record ConveyorBeltOrigin, XX using vision calibration tool by moving the robot TCP to the marked XX dot and recording the TCP position relative to WCS. This is a point on the +X axis of ConveyorBeltOrigin
5. Record ConveyorBeltOrigin, XY using vision calibration tool moving the robot TCP to the marked XY dot and recording the TCP position relative to WCS. This is a point on the XY plane of ConveyorBeltOrigin.

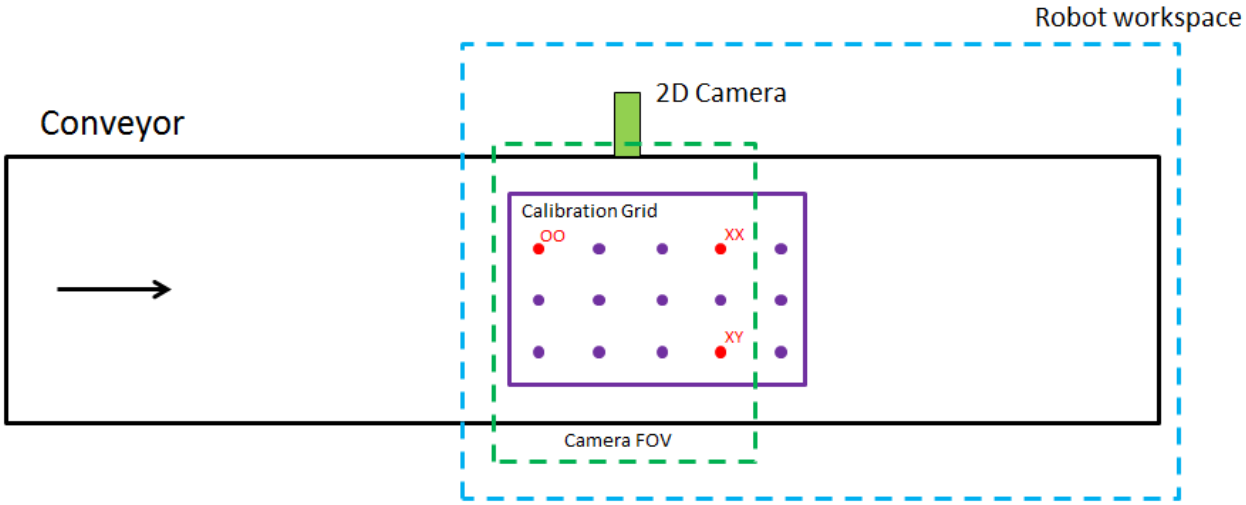

FIGURE 29 CALIBRATION GRID AND REPRESENTATION OF POINT XX AND XY

6. Calculate ConveyorBeltOrigin using FrameTypeTransform, found in the Math_Toolbox.

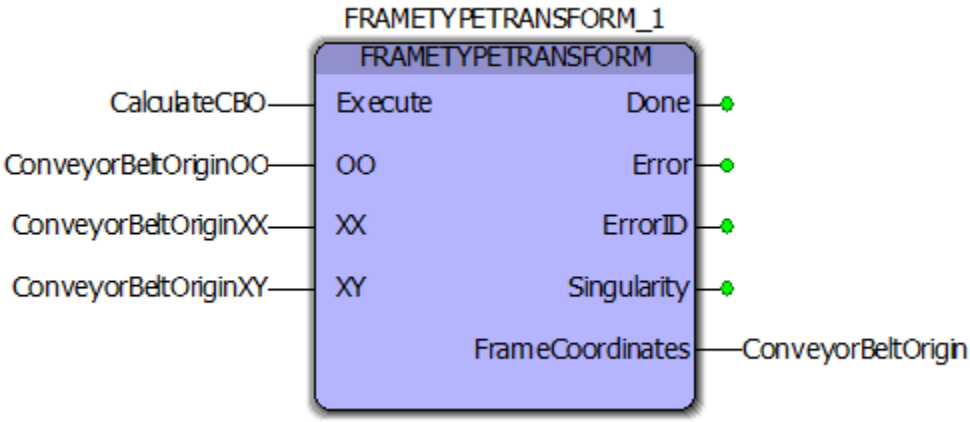

FIGURE 30 CALCULATE CONVEYORBELTORIGIN WITH FRAMETYPETRANSFORM IN MATH_TOOLBOX

☐ Example – Camera outside robot workspace

This example places ConveyorBeltOrigin OUTSIDE the robot workspace. To define the ConveyorBeltOrigin outside the workspace, a temporary frame can be defined inside the workspace that is the same as the ConveyorBeltOrigin except offset downstream (+X direction) by a known amount. Then CalcFrameOffset can be used to calculate ConveyorBeltOrigin using the temporary frame and the known offset.

In this case, a dot grid is used. 3 dots on the grid (OO, XX, XY) are marked as points to be used to calculate the temporary frame. When calibrating the camera, OO should be the origin, XX should be along the +X axis, and XY should be a point on the XY plane. Do not move the calibration grid after calibrating camera.

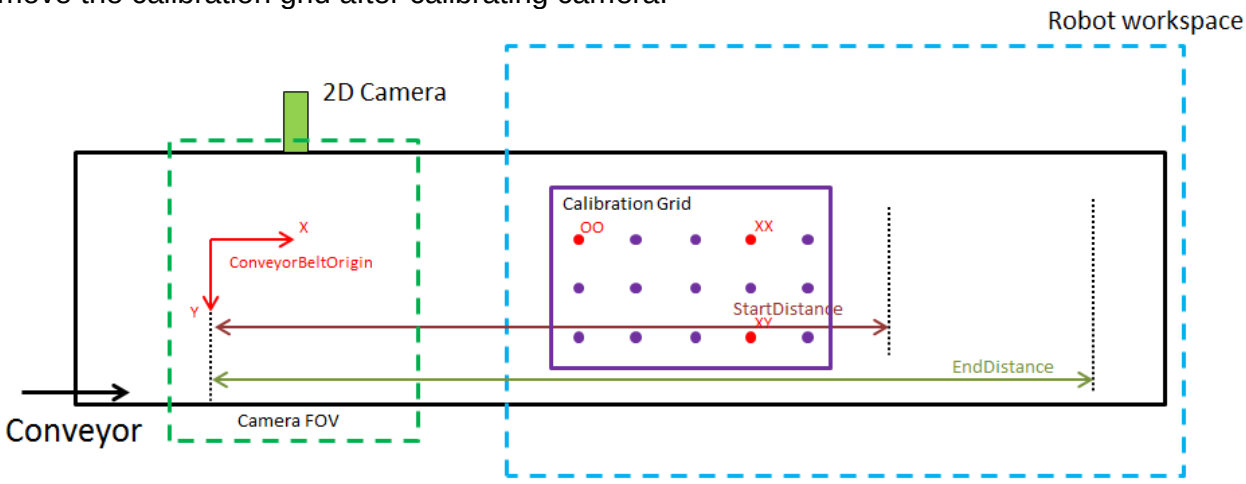

FIGURE 31 REPRESENTATION OF CONVEYOR TRACKING SETUP WHEN EXAMPLE IS FINISHED

1. Calibrate camera using vision calibration tool such that +X direction of the camera is in positive travel direction of conveyor

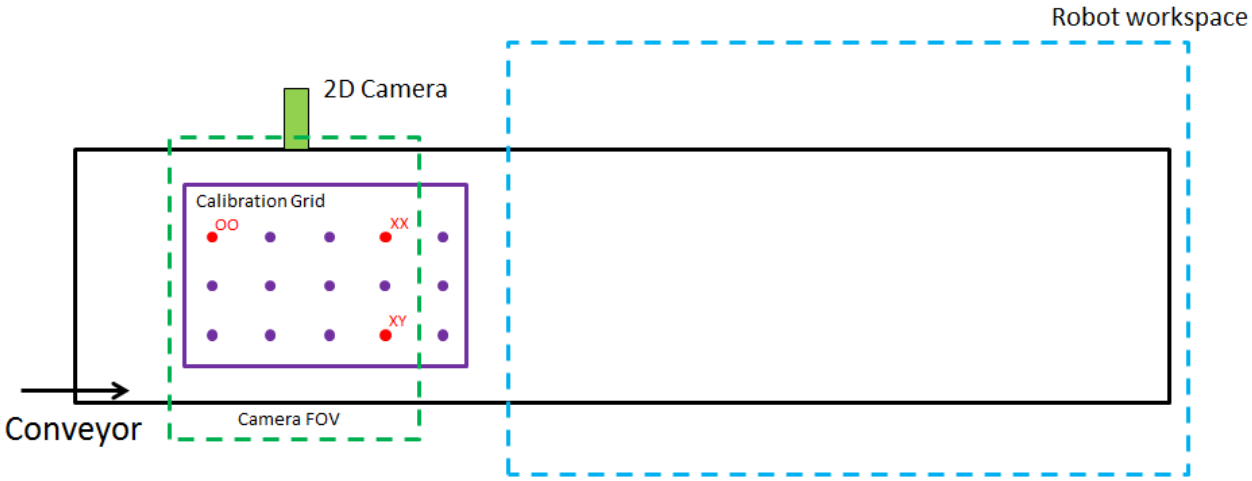

FIGURE 32 CALIBRATION GRID MARKED WITH OO XX XY PLACED OUTSIDE ROBOT WORKSPACE

2. Record Conveyor Position 1
3. Find StartDistance and EndDistance by measuring from OO to where it is desired for conveyor tracking to start and end.

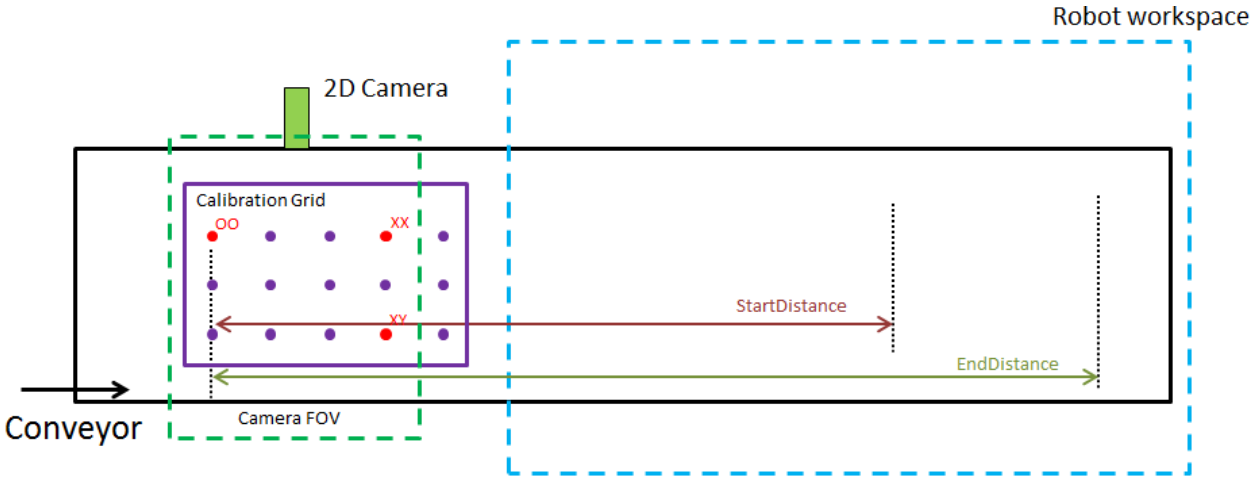

FIGURE 33 MEASURE STARTDISTANCE AND ENDDISTANCE

4. Jog conveyor until the marked points OO, XX, XY are inside robot workspace
5. Record Conveyor Position 2

6. Record Temp Frame, OO using vision calibration tool by moving the robot TCP to the marked OO dot and recording the TCP position relative to WCS. This is the origin of the temporary frame that will be used to calculate ConveyorBeltOrigin
7. Record Temp Frame, XX using vision calibration tool by moving the robot TCP to the marked XX dot and recording the TCP position relative to WCS. This is a point on the +X axis of the temporary frame
8. Record Temp Frame, XY using vision calibration tool moving the robot TCP to the marked XY dot and recording the TCP position relative to WCS. This is a point on the XY plane of the temporary frame

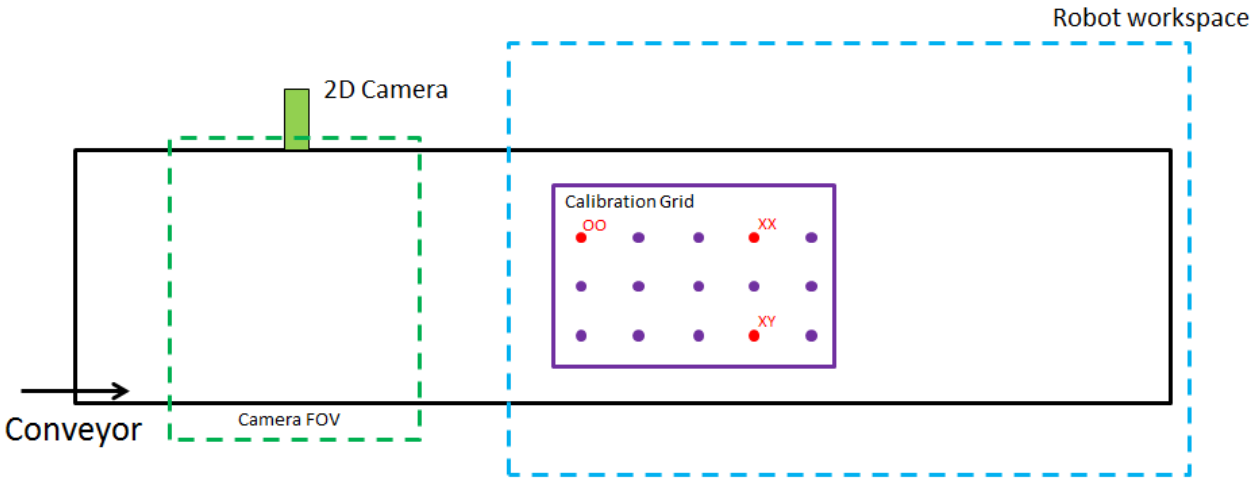

FIGURE 34 CONVEYOR IS MOVED DOWNSTREAM UNTIL OO XX XY ARE WITHIN ROBOT WORKSPACE

9. Calculate ConveyorBeltOrigin using FrameTypeTransform and CalcFrameOffset, found in the Math_Toolbox. Here, Offset[1] = ConveyorPosition1 – ConveyorPosition2 (should be negative), Offset[2..6] = 0.

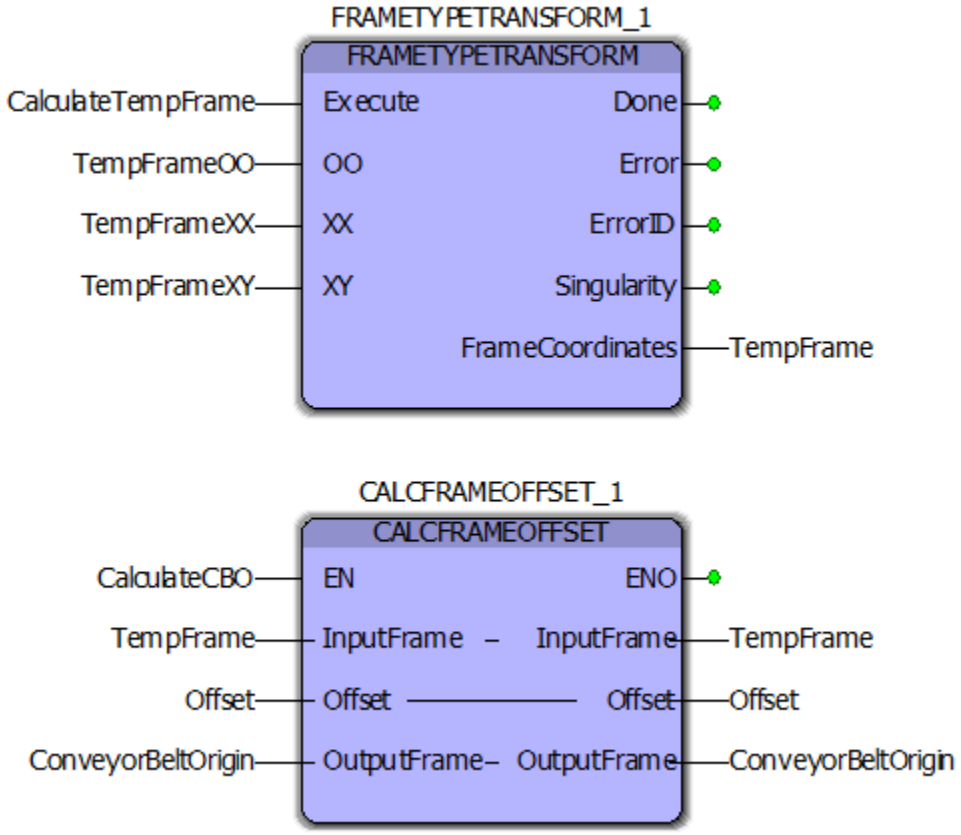

FIGURE 35 CONVEYORBELTORIGIN IS CALCULATED USING OO XX XY AND THE TRAVELED CONVEYOR DISTANCE

FAQ

☐ Motion Details

Is it possible to transition from tracking one conveyor to another?

As of 3.4, this is not possible. For MC_TrackConveyorBelt to work, the group must be in TrackState 0 (not tracking at all) AND have no PCS moves in queue. It is then not possible to blend PCS moves from one conveyor to another directly. Easiest way is to execute a MCS move to abort tracking and wait for TrackState to return to 0 before executing a new MC_TrackConveyorBelt.

Why does TrackOptions.SyncOut.ConveyorDistance seem to do nothing?

The tracking profile has a default end that is defined by SyncOut.ConveyorDistance if tracking is allowed to run all the way to EndDistance. However, in typical MC_ConveyorTracking use, it will be aborted before EndDistance is reached by the object. In this case, SyncOut.ConveyorDistance does nothing.

When can a PCS move be executed and have it be relative to the tracked frame?

Any PCS move executed after MC_TrackConveyorBelt will be relative to the frame being tracked.

What is the initial value of the PCS definition when conveyor tracking is valid? When does it get set?

PCS definition is initially defined at StartDistance + TrackOptions.SyncIn.ConveyorDistance – TrackOptions.SyncIn.TCPDistance. It is set when ExecutionMode is satisfied.

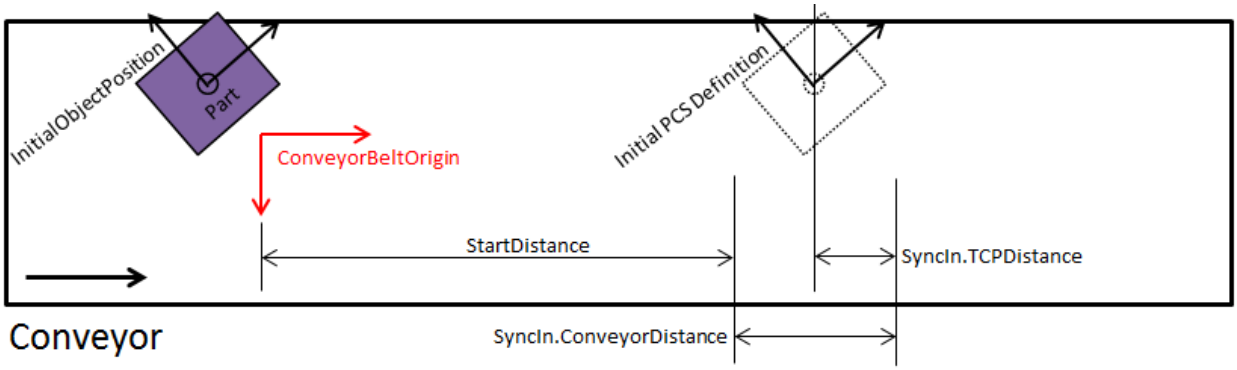

FIGURE 36 INITIAL DEFINITION OF PCS WHEN EXECUTIONMODE IS SATISFIED

How does the conveyor motion factor into the overall motion of the group while doing conveyor tracking? Can speed of conveyor be varied?

The MC_TrackConveyorBelt provides its conveyor own motion, and is superimposed in top of group point to point motion. The conveyor motion is always in the direction specified by ConveyorBeltOrigin's X direction, and is a position based tracking. The speed of conveyor belt can be varied, but direction should always be forward.

Why does the group over speed during the SyncIn (TrackState 2)?

This typically happens if ExecutionMode is satisfied after object (frame to track) has passed StartDistance. With ExecutionMode = Immediate, this will happen when MC_TrackConveyorBelt is executed with the object past StartDistance. With ExecutionMode=Delayed or Queued, this will happen if the object is past StartDistance when the relevant segment to satisfy ExecutionMode is done.

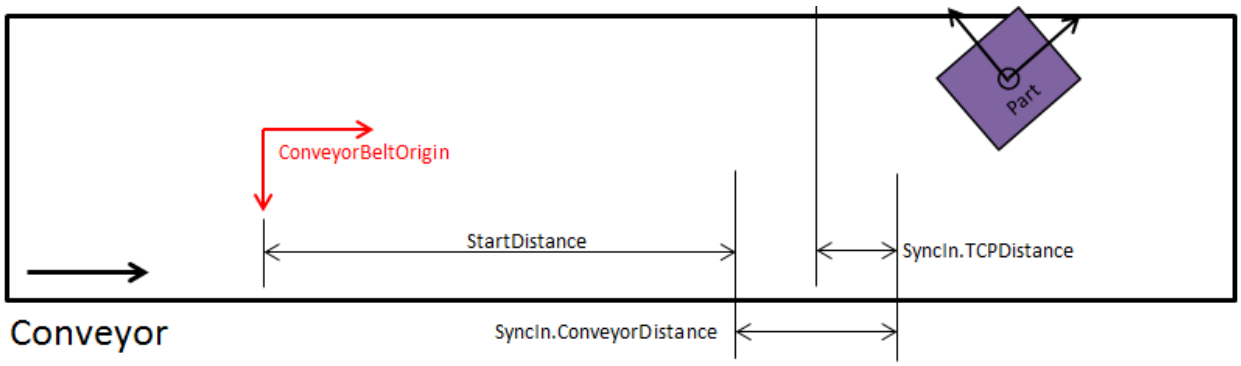

FIGURE 37 EXECUTIONMODE IS SATISFIED AFTER OBJECT PASSES STARTDISTANCE

Regardless of where the object is when ExecutionMode is satisfied, the initial PCS definition is always at StartDistance + TrackOptions.SyncIn.ConveyorDistance – TrackOptions.SyncIn.TCPDistance. Since the object is potentially far past this point, in order for tracking to be “InSync”, the PCS definition has to catch up to where the object currently is by the time the conveyor travels some distance defined by TrackOptions.SyncIn.ConveyorDistance. So, the conveyor motion part of the move has to over speed to catch up compared to if the object is upstream of StartDistance when ExecutionMode is satisfied.

In general, the further downstream the object is when ExecutionMode is satisfied, the more over speed it will have because the conveyor motion part of the tracking has a longer distance to cover in the same conveyor travel. A workaround for this is to have the StartDistance input of MC_TrackConveyorBelt be a variable that is computed to be where the object is when MC_TrackConveyorBelt is executed, and compare that to the desired StartDistance, and use the larger one. This will ensure that the initial PCS definition is placed at the object.

StartDistanceInput = (ConveyorPosition - RecordedPosition + InitialObjectPosition[1])

The downside is that the user has to only execute MC_TrackConveyorBelt after the object passes the real StartDistance.

Why is TrackState 4 being skipped?

TrackState 4 will be skipped if sync out is done with MC_GroupStop or an aborting MCS move. TrackState 4 refers to either the end of the default tracking profile when sync out is done over SyncOut.ConveyorDistance, or it refers to when a blended/buffered MCS move is used to sync out. In the latter case, TrackState 4 refers to when the unwind motion is active, which takes place at the same time as the MCS move.

What is the unwind motion when sync out with MCS move? Why does the SyncOut with blended/buffered MCS not move in the direction of the MCS end point initially (TrackState 4)?

In order to smoothly transition from doing PCS moves while conveyor tracking to the specified MCS position, there is an unwind motion that occurs at the same time as the blended/buffered MCS move. This is due to the fact that when the blended/buffered MCS move is execute, a trajectory is calculated but it is not yet active. In some later time, the MCS move becomes active to sync out of conveyor tracking. In the time between the MCS

move is executed and it becomes active, the conveyor has traveled some distance. So to use the previously calculated trajectory, an unwind motion has to be done in the –X direction of ConveyorBeltOrigin to account for the conveyor distance traveled between the execute and active of the MCS move used to sync out. This unwind motion is superimposed with the MCS move, and the resulting motion may not be in the same direction of the MCS position end point. Generally, it is more noticeable if the conveyor travels a longer distance during the time between the execute and active of the MCS move.

What if I do not want the unwind motion during sync out?

To not have the conveyor unwind motion during sync out, the user can allow all PCS moves to finish before executing a MCS move to sync out. In this case, the motion queue only contains the MCS move, and so there is no delay between execute and active, and will go directly to the end point of the MCS move. The disadvantage is the MCS move will not be blended with the previous PCS move.