PathPlannerLib

LabVIEW

Reference

Version 4.0.0

Table of Contents

Introduction

The PathPlanner LabVIEW library provides utility functions to read, create, and follow PathPlanner paths.

The library source code, package build specifications, and test package can be found here https://github.com/jsimpso81/PathPlannerLabVIEW

Function Help

Each VI includes help that can be accessed using the standard LabVIEW help toggle (Ctrl H).

TO DO YET

Function Examples

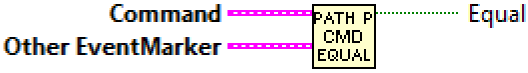
Many of the functions have examples that can be found under the LabVIEW "Find examples..." function. (Help -> Find Examples...). The function examples are easiest to find when "Directory Structure" is selected.

TO DO YET

Function Groups

CommandUtil

PathPlanner_CommandUtil_Equals



Determines if two Goal End State definitions are equal

Inputs:

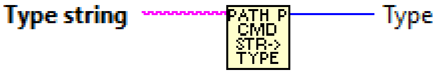
-- GoalEndState - cluster - goal end state definition

-- Other GoalEndState - cluster - goal end state definition

Outputs:

-- Equal - boolean - TRUE if both definitions are the same.

PathPlanner_CommandUtil_TypeFromString



Get the command utility type enum from a string

Inputs:

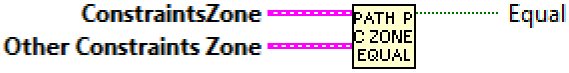
-- Type string - string - string to evaluate for command util type

Outputs:

-- type - enum - Evaluated command util type.

ConstraintsZone

PathPlanner_ConstraintsZone_Equals



Compares two Constraints Zone definitions

Inputs:

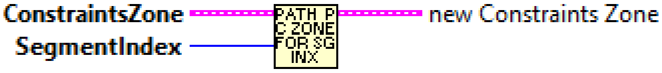
-- ConstraintsZone - cluster - first definition to compare

-- Other ConstraintsZone - cluster - other definition to compare

Outputs:

-- Equal - boolean - TRUE if equal.

PathPlanner_ConstraintsZone_ForSegmentIndex



Transform the positions of this zone for a given segment number.

For example, a zone from [1.5, 2.0] for the segment 1 will have the positions [0.5, 1.0]

Inputs:

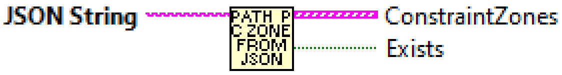
-- ConstraintsZone - cluster - definition of zone

-- segmentIndex - int - The segment index to transform positions for

Outputs:

-- NewConstraintsZone - cluster - The transformed zone

PathPlanner_ConstraintsZone_FromJSON



Create a constraints zone from json

Inputs:

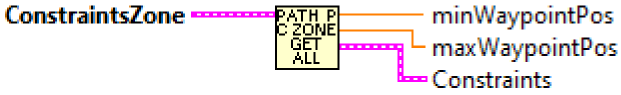
-- JsonString - string - String containing the JSON to parse.

Outputs:

-- ConstraintsZone - cluster - The constraints zone defined by the given json object

-- Exists -- boolean -- True if a constraints zone was found and parsed.

PathPlanner_ConstraintsZone_GetAll



Get the elements of the constraints zone cluster.

Inputs:

-- ConstraintsZone - cluster - Data structure containing constraints zone.

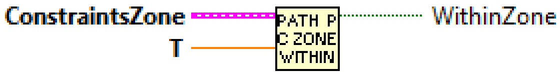
Outputs:

-- minWayPointPos - double - Waypoint relative starting position

-- maxWayPointPos - double - Waypoint relative end position

-- Constaints -- cluster -- Constraints to apply within this region.

PathPlanner_ConstraintsZone_IsWithinZone



Get if a given waypoint relative position is within this zone

Inputs:

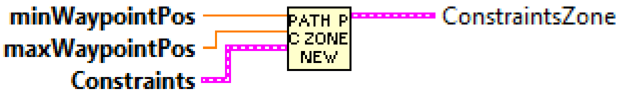
-- ConstraintsZone -- cluster -- data structure containing zone definition.

-- t - double - Waypoint relative position

Outputs:

-- WithinZone - boolean - True if given position is within this zone

PathPlanner_ConstraintsZone_New



Create a new constraints zone

Inputs:

-- minWaypointPos - double - Starting position of the zone

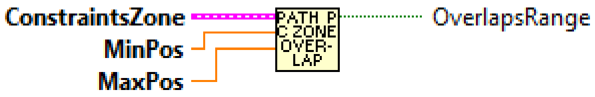
-- maxWaypointPos - double - End position of the zone

-- constraints - cluster - The constraints to apply within the zone

Outputs:

-- ConstraintsZone - cluster - data cluster with constraint

PathPlanner_ConstraintsZone_OverlapsRange



Get if this zone overlaps a given range

Inputs:

-- ConstraintsZone - cluster - zone definition.

-- minPos - double - The minimum waypoint relative position of the range

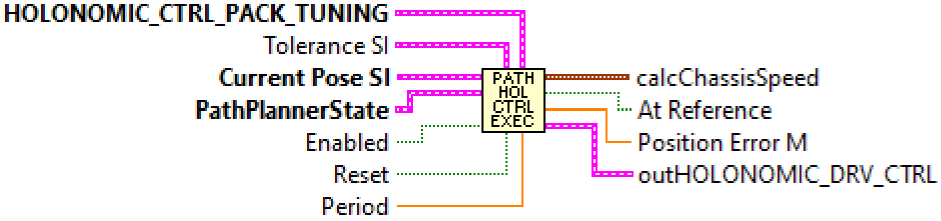
-- maxPos - double - The maximum waypoint relative position of the range

Outputs:

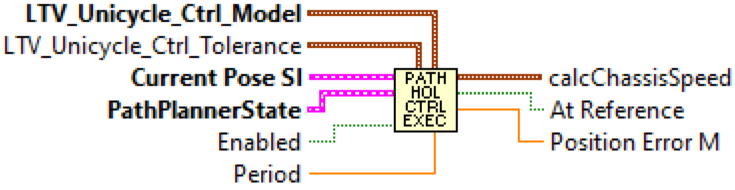
-- OverlapsRange - boolean - True if any part of this zone is within the given range

Ctrl

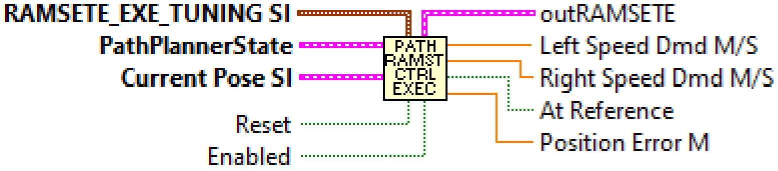
PathPlanner_Ctrl_HolonomicDrvExecute



PathPlanner_Ctrl_LTVExecute

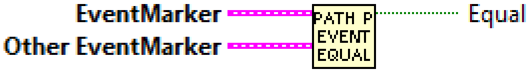


PathPlanner_Ctrl_RamseteExecute



EventMarker

PathPlanner_EventMarker_Equals



Determinesif two event markers are equal

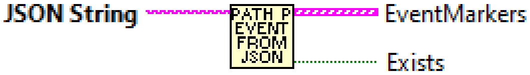
Inputs:

- EventMarker - cluster - Data cluster
- OtherEventMarker - cluster - Data cluster

Outputs:

- Equal - boolean - TRUE if both event markers are equal

PathPlanner_EventMarker_FromJSON



Create a list of event markers from json string

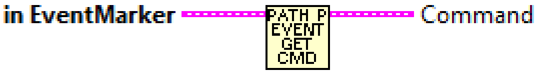
Inputs:

- JSONString - string - String potentially containing an event marker

Outputs:

- EventMarkers - array of cluster - The event markers defined by the given json object
- Exists - boolean- TRUE if any event markers were found in the JSON string.

PathPlanner_EventMarker_GetCommand



Get the command associated with this marker

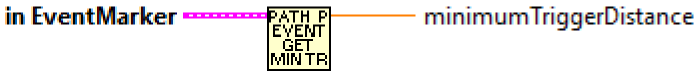
Inputs:

- EventMarker - cluster - Data cluster

Outputs:

- Command - cluster - command for this marker

PathPlanner_EventMarker_GetMinimumTriggerDistance



Get the minimum trigger distance for this marker

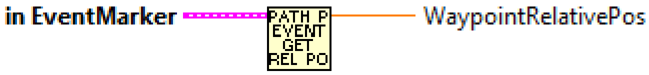
Inputs:

- EventMarker - cluster - Data cluster

Outputs:

- MinTriggerDistance - double - The minimum trigger distance in meters

PathPlanner_EventMarker_GetWaypointRelativePos



Get the waypoint relative position of this marker

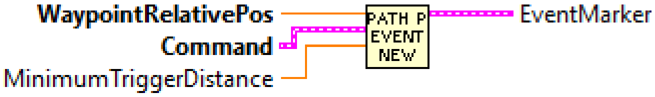
Inputs:

- EventMarker - cluster - Data cluster

Outputs:

- WaypointRelativePose - double - Waypoint relative position of this marker

PathPlanner_EventMarker_New



Create a new event marker.   This describes a position along the path that will trigger a command when reached

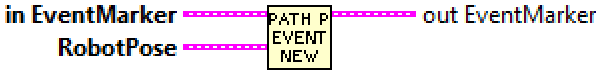
Inputs:

- waypointRelativePos - double - The waypoint relative position of the marker
- command - cluster - The command that should be triggered at this marker
- minimumTriggerDistance - double - The minimum distance the robot must be within for this marker to be triggered (Optional. Default: 0.5)

Outputs:

- EventMarker - cluster - Data cluster

PathPlanner_EventMarker_Reset



Reset the current robot position

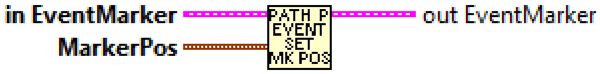
Inputs:

- EventMarker - cluster - Data cluster
- robotPose - pose2d - The current pose of the robot

Outputs:

- EventMarker - cluster - Data cluster

PathPlanner_EventMarker_SetMarkerPos



Get the marker position for this event

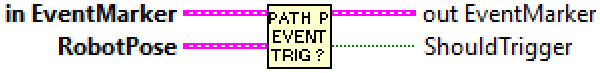
Inputs:

- EventMarker - cluster - Data cluster
- MarkerPos - double - Marker position

Outputs:

- EventMarker - cluster - Data cluster

PathPlanner_EventMarker_ShouldTrigger



Get if this event marker should be triggered

Inputs:

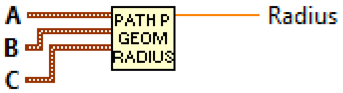
- EventMarker - cluster - Data cluster
- robotPose - pose2d - Current pose of the robot

Outputs:

- EventMarker - cluster - Data cluster
- Trigger - boolean - True if this marker should be triggered

GeomUtil

PathPlanner_GeomUtil_CalculateRadius



Calculate the curve radius given 3 points on the curve

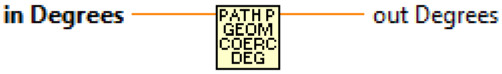
Inputs:

- a - translation2d - Point A
- b - translation2d - Point B
- c - translation2d - Point C

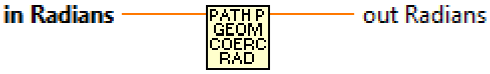
Outputs:

- Radius - double - Curve radius

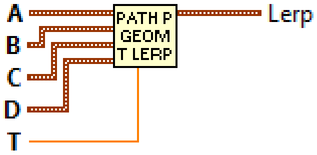
PathPlanner_GeomUtil_CoerceHeadingDegrees



PathPlanner_GeomUtil_CoerceHeadingRadians



PathPlanner_GeomUtil_CubicLerp



Cubic interpolation between Translation2ds

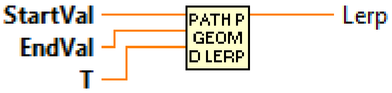
Inputs:

- a - translation2d - Position 1
- b - translation2d - Position 2
- c - translation2d - Position 3
- t - double - Interpolation factor (0.0-1.0)

Outputs:

- Lerp - translation2d - Interpolated value

PathPlanner_GeomUtil_DoubleLerp



Interpolate between two doubles

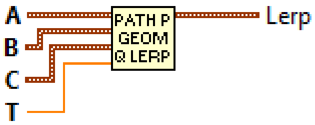
Inputs:

- startVal - double - Start value
- endVal - double - End value
- t - double - Interpolation factor (0.0-1.0)

Outputs:

- Lerp - double - Interpolated value

PathPlanner_GeomUtil_QuadraticLerp



Quadratic interpolation between Translation2ds

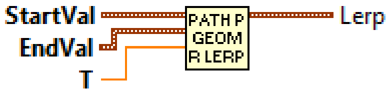
Inputs:

- a - translation2d - Position 1
- b - translation2d - Position 2
- c - translation2d - Position 3
- d - translation2d - Position 4
- t - double - Interpolation factor (0.0-1.0)

Outputs:

- Lerp - translation2d - Interpolated value

PathPlanner_GeomUtil_RotationLerp



Interpolate between two Rotation2ds

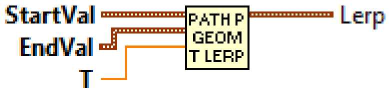
Inputs:

- startVal - rotation2d - Start value
- endVal - rotation2d - End value
- t - double - Interpolation factor (0.0-1.0)

Outputs:

- Lerp - rotation2d - Interpolated value

PathPlanner_GeomUtil_TranslationLerp



Inputs:

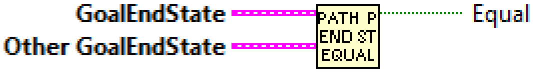
- a - translation2d - Position 1
- b - translation2d - Position 2
- t - double - Interpolation factor (0.0-1.0)

Outputs:

- lerp - translation2d - Interpolated value

GoalEndState

PathPlanner_GoalEndState_Equals



Determines if two Goal End State definitions are equal

Inputs:

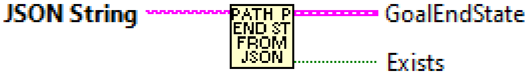
-- GoalEndState - cluster - goal end state definition

-- Other GoalEndState - cluster - goal end state definition

Outputs:

-- Equal - boolean - TRUE if both definitions are the same.

PathPlanner_GoalEndState_FromJSON



Create a goal end state from json

Inputs:

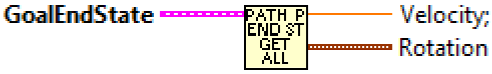
-- JSON String - string - string to parse for GoalEndState

Outputs:

-- GoalEndState - cluster - The goal end state defined by the given json. If not found, default is returned.

-- exists - boolean - TRUE if GoalEndState was found and parsed in the JSON string.

PathPlanner_GoalEndState_GetAll



Get the goal end velocity and end rotation

Inputs:

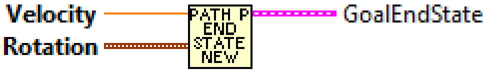
-- GoalEndState - cluster - definition data structure

Outputs:

-- Goal end velocity (M/S)

-- Goal rotation

PathPlanner_GoalEndState_New



Describes the goal end state of the robot when finishing a path */

Create a new goal end state

Inputs:

-- velocity - double - The goal end velocity (M/S)

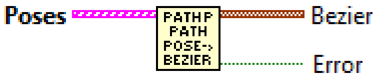
-- rotation - rotation2d - The goal rotation

Outputs:

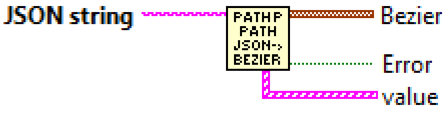
-- GoalEndState - cluster - data structure

Path

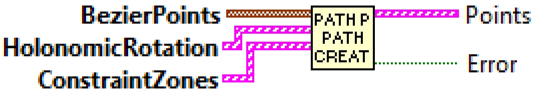
PathPlanner_Path_BezierFromPoses



PathPlanner_Path_BezierFromWaypointsJSON



PathPlanner_Path_CreatePath



Create a path from an array of PathPlanner Waypoints.   This routine calculates the trajectory states. It handles combining separate paths when reversal is flagged.

Inputs:

- PathPlannerWaypoints -- An array of PathPlannerWaypoints. This array must contain at least 2 entries.

- maxVel -- Max velocity of the path

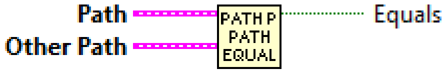
- maxAccel -- Max velocity of the path

- reversed -- Should the robot follow the path reversed

Outputs:

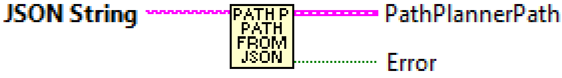
- PathPlanner Trajectory -- The generated path (path planner trajectory) can be converted to trajectory)

PathPlanner_Path_Equals

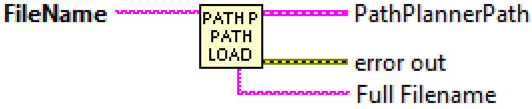


NOT DONE - DONT USE

PathPlanner_Path_FromJSON



PathPlanner_Path_FromPathFile



Load a path file from storage.   This loads the waypoints from a file and calculates the trajectory states. It handles combining separate paths when reversal is flagged.

Inputs:

- FileName -- The name of the path to load.   Absolute or relative.
- maxVel -- Max velocity of the path
- maxAccel -- Max velocity of the path
- reversed -- Should the robot follow the path reversed

Outputs:

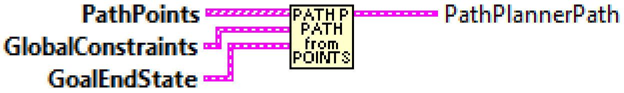
- PathPlanner Trajectory -- The generated path (path planner trajectory) can be converted to trajectory)
- Error -- error cluster

Note:

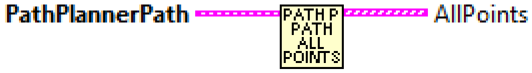
-- Default path for roboRIO is   /home/lvuser/natinst/LabVIEW Data

-- Default path for Windows is: %USERPROFILE%\Documents\LabVIEW Data

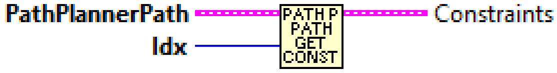
PathPlanner_Path_FromPathPonts



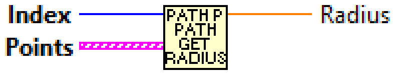
PathPlanner_Path_GetAllPathPoint



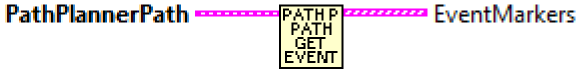
PathPlanner_Path_GetConstraintsForPoint



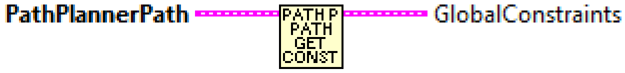
PathPlanner_Path_GetCurveRadiusAtPoint



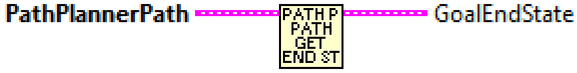
PathPlanner_Path_GetEventMarkers



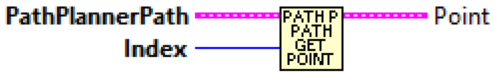
PathPlanner_Path_GetGlobalConstraints



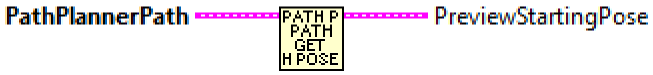
PathPlanner_Path_GetGoalEndState



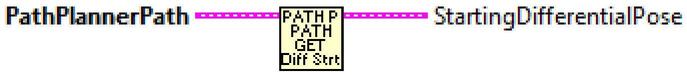
PathPlanner_Path_GetPoint



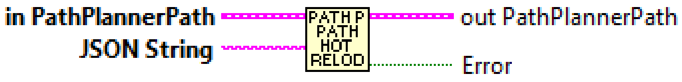
PathPlanner_Path_GetPreviewStartingHolonomicPose



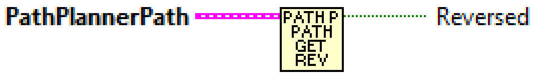
PathPlanner_Path_GetStartingDifferentialPose



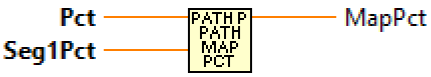
PathPlanner_Path_HotReload



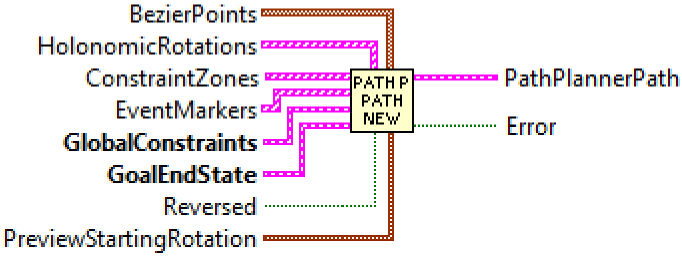
PathPlanner_Path_IsReversed



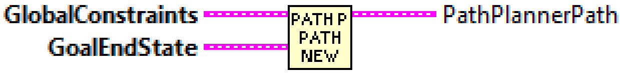
PathPlanner_Path_MapPct



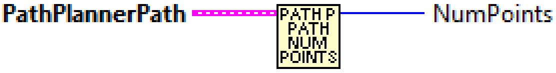
PathPlanner_Path_New



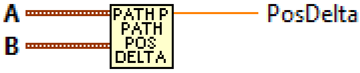
PathPlanner_Path_New_Empty



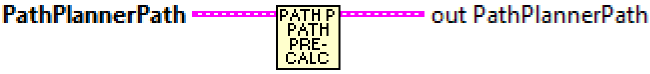
PathPlanner_Path_NumPoints



PathPlanner_Path_PositionDelta

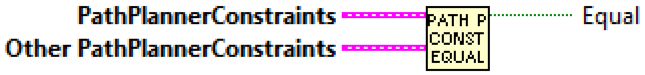


PathPlanner_Path_PreCalcValues



PathConstraints

PathPlanner_PathConstraints_Equals



Determines if two Path Constraints definitions are nearly identical. The values have to be within 0.001 of eah other.

Inputs:

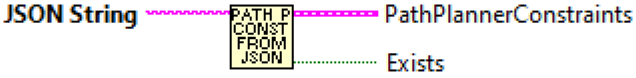
- PathPlannerConstraints - cluster - definition of path constraints

- OtherPathPlannerConstraints - cluster - definition of seond path constraints for comparision

Outputs:

- Equal - boolean - TRUE indicates the provided definitions are nearly identical.

PathPlanner_PathConstraints_FromJSON



Create a path constraints object from json string

Inputs:

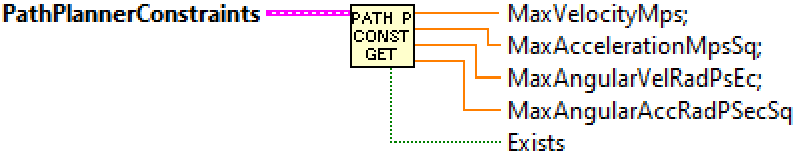
- JSON String - string - string potentially containing a path constraints definition

Outputs:

- PathConstraint - cluster - The path constraints defined by the given json

- Exists - boolean - TRUE if the string contained a path constraints definition

PathPlanner_PathConstraints_GetAll



Get all elements of Path Constraints cluster

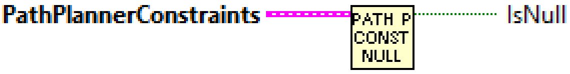
Inputs:

- PathConstraint - cluster - The path constraints to query

Outputs:

- maxVelocityMps - double - Max linear velocity (M/S)
- maxAccelerationMpsSq - double - Max linear acceleration (M/S^2)
- maxAngularVelocityRps - double - Max angular velocity (Rad/S)
- maxAngularAccelerationRpsSq - double - Max angular acceleration (Rad/S^2)
- exists - boolean - TRUE if this data cluster is not null.

PathPlanner_PathConstraints_IsNull



Return indication that the PathConstraints data definition isn't null (not defined)

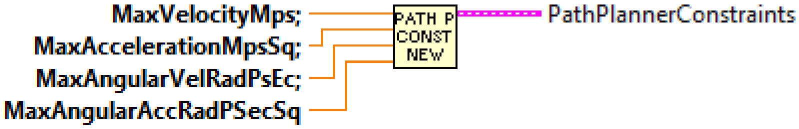
Inputs:

- PathPlannerConstraints - cluster - Path Constraints definition to evaluate.

Outputs:

- IsNull - boolean - TRUE if definition is NULL.

PathPlanner_PathConstraints_New



Create a new path constraints object

Inputs:

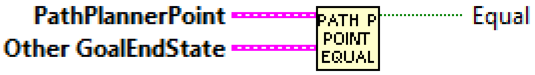
- maxVelocityMps - double - Max linear velocity (M/S)
- maxAccelerationMpsSq - double - Max linear acceleration (M/S^2)
- maxAngularVelocityRps - double - Max angular velocity (Rad/S)
- maxAngularAccelerationRpsSq - double - Max angular acceleration (Rad/S^2)

Outputs:

- PathConstraint - cluster - path constraint data

PathPoint

PathPlanner_PathPoint_Equals



Determines if two Path Point definitions are equal

Inputs:

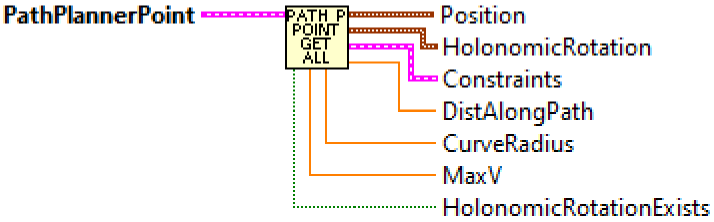
-- PathPoint - cluster - point definition

-- Other PathPoint - cluster - point definition

Outputs:

-- Equal - boolean - TRUE if both definitions are the same.

PathPlanner_PathPoint_GetAll



Gets elements of PathPoint

Inputs:

-- PathPoint - cluster - point definition

Outputs:

-- Position - Translation2d - position of point

-- HolonomicRotation - Rotation2d - rotational orientation of point

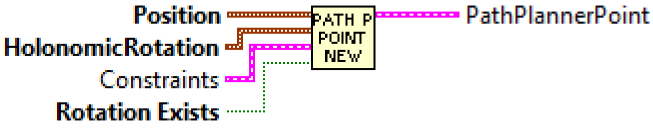
-- Constraints - cluster - contraints at this oint

-- DistAlongPath - double -

-- CurveRadius - double -

-- MaxV - double

PathPlanner_PathPoint_New



Create a path point

Inputs:

-- position Position of the point

-- holonomicRotation Rotation target at this point

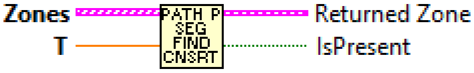
-- constraints The constraints at this point

Outputs:

-- PathPlannerPoint - cluster - point definition

PathSegment

PathPlanner_PathSegment_FindConstraintZone



Find a constraints zone within this path segment.

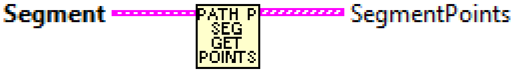
Inputs:

- PathSegment - cluster - Data defining path segment
- Zones - array - List of constraint zones to search.

Outputs:

- ReturnedZone - cluster - Found constraints zone definition.
- IsPresent - boolean - TRUE if a constraints zone was found.

PathPlanner_PathSegment_GetSegmentPoints



Get the path points for this segment

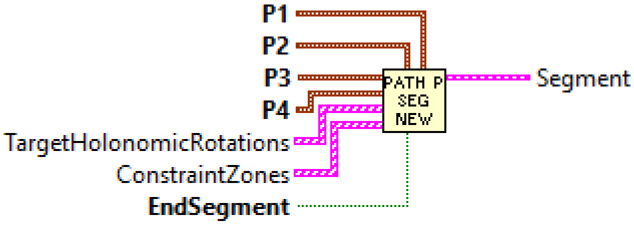
Inputs:

- PathSegment - cluster - Data defining path segment

Outputs:

- SetmentPoint - array - points for this segment

PathPlanner_PathSegment_New



Generate a new path segment

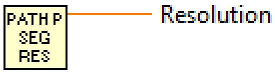
Inputs:

- p1 - translation2d - Start anchor point
- p2 - translation2d - Start next control
- p3 - translation2d - End prev control
- p4 - translation2d - End anchor point
- targetHolonomicRotations - array Rotation targets for within this segment (Optional. Default: empty)
- constraintZones - array - Constraint zones for within this segment (Optional. Default: empty)
- endSegment - boolean - Is this the last segment in the path

Outputs:

- Segment - cluster - Data defining path segment

PathPlanner_PathSegment_Resolution



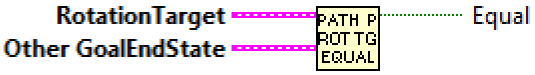
The resolution used during path generation

Outputs:

- Resolution - double

RotationTarget

PathPlanner_RotationTarget_Equals



Determine if two rotation targets are equal

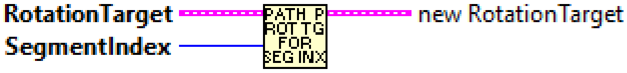
Inputs:

- RotationTarget - cluster - defined rotation target data structure
- OtherRotationTarget - cluster - defined rotation target data structure

Outputs:

- Equal - boolean - TRUE if both rotation targets are the same

PathPlanner_RotationTarget_ForSegmentIndex



Transform the position of this target for a given segment number.

For example, a target with position 1.5 for the segment 1 will have the position 0.5

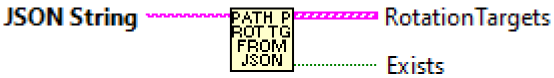
Inputs:

- RotationTarget - cluster - defined rotation target data structure
- segmentIndex- integer - The segment index to transform position for

Outputs:

- NewRotationTarget - cluster - The transformed target

PathPlanner_RotationTarget_FromJSON



Create a rotation target from json

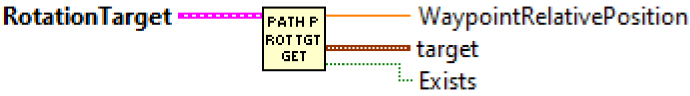
Inputs:

- JSON String - string - string potentially containing one or more of rotation target

Outputs:

- RotationTarget - array - Set of rotation targets defined by the given json string
- Exists - boolean - TRUE if any rotation targets were found in the JSON string.

PathPlanner_RotationTarget_GetAll



Get data elements of a rotation target.

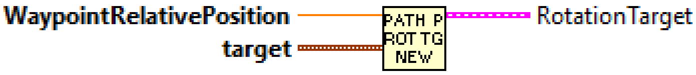
Inputs:

- RotationTarget - cluster - defined rotation target data structure

Outputs:

- WaypointRelativePos - double - waypoint relative position of this target
- TargetRotation - rotation2d - Rotation value
- Exists - boolean - TRUE if rotation target is not null

PathPlanner_RotationTarget_New



Create a new rotation target

Inputs:

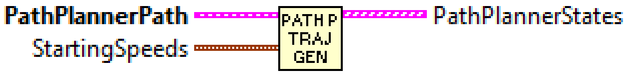
- waypointRelativePosition - double - Waypoint relative position of this target
- target - rotation2d - Target rotation

Outputs:

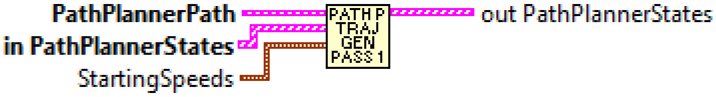
- RotationTarget - cluster - defined rotation target data structure

Trajectory

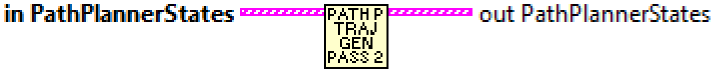
PathPlanner_Trajectory_GenerateStates



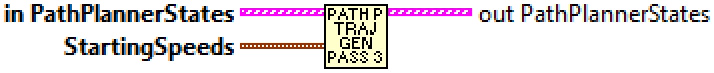
PathPlanner_Trajectory_GenerateStates_Pass1



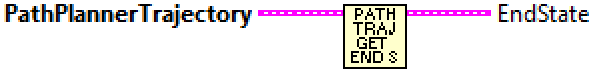
PathPlanner_Trajectory_GenerateStates_Pass2



PathPlanner_Trajectory_GenerateStates_Pass3



PathPlanner_Trajectory_GetEndState



Get the end state of the trajectory

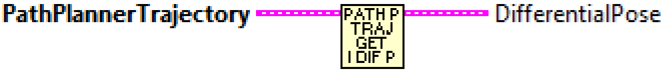
Inputs:

- Trajectory - cluster - trajectory definition

Outputs:

- EndState - trajectoryState - The end state

PathPlanner_Trajectory_GetInitialDifferentialPose



Get this initial pose for a differential drivetrain

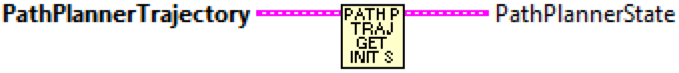
Inputs:

- Trajectory - cluster - trajectory definition

Outputs:

- DifferentialPose - pose2d - The initial pose

PathPlanner_Trajectory_GetInitialState



Get the initial state of the trajectory

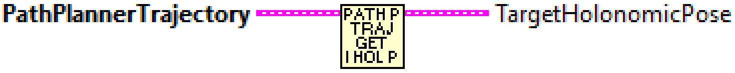
Inputs:

- Trajectory - cluster - trajectory definition

Outputs:

- InitialState - trajectoryState - The initial state

PathPlanner_Trajectory_GetInitialTargetHolonomicPose



Get the initial target pose for a holonomic drivetrain NOTE: This is a "target" pose, meaning the rotation will be the value of the next rotation target along the path, not what the rotation should be at the start of the path

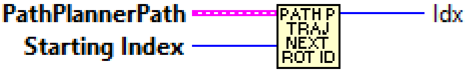
Inputs:

- Trajectory - cluster - trajectory definition

Outputs:

- TargetHolonomicPose - pose2d - The initial target pose

PathPlanner_Trajectory_GetNextRotationTargetIdx



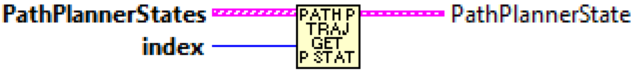
Inputs:

- trajectory - cluster - trajectory definition

Outputs:

- NextRotationTargetIndex - integer -

PathPlanner_Trajectory_GetState



Get the goal state at the given index

In most (all) cases, using sample() is a better method.

Inputs:

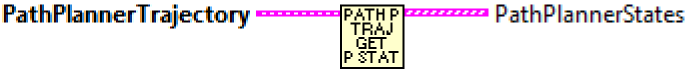
-- Traectory -- PathPlanner Trajectory data cluster

-- index -- The index of the state to retrieve

Outputs:

- TrajectoryState -- The state at the given index

PathPlanner_Trajectory_GetStates



Get all of the pre-generated states in the trajectory

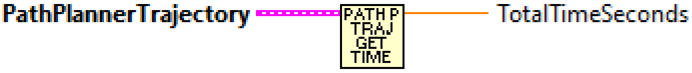
Inputs:

- Trajectory - cluster - trajectory definition

Outputs:

- TrajectoryStates - array - List of all states

PathPlanner_Trajectory_GetTotalTime



Get the total run time of the trajectory

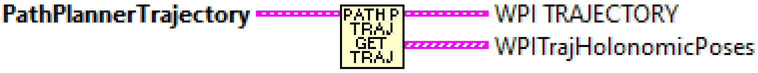
Inputs:

- Trajectory - cluster - trajectory definition

Outputs:

- TotalTime - seconds - Total run time in seconds

PathPlanner_Trajectory_GetWPITrajectory



Convert a PathPlanner trajectory into a LabVIEW / WPILib Trajectory.

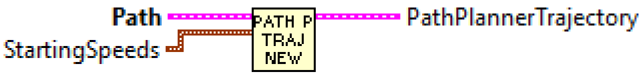
Inputs:

-- PathPlannerTrajectory -- PathPlanner Trajectory data cluster

Outputs

-- Traectory -- LabVIEW traectory library (WPlib style) trajectory data cluster.

PathPlanner_Trajectory_New



Generate a PathPlannerTrajectory

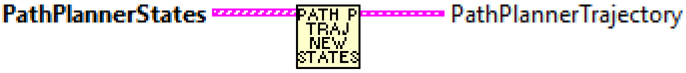
Inputs:

- path - cluster - path to generate the trajectory for
- startingSpeeds - chassis speeds - Starting speeds of the robot when starting the trajectory

Outputs:

- trajectory - cluster - created trajectory data

PathPlanner_Trajectory_New_States



Generate a PathPlannerTrajectory

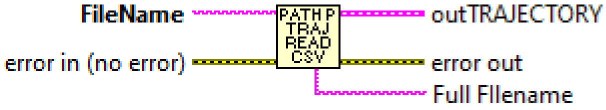
Inputs:

- PathPlannerStates - array of TrajectoryStates - States to use to create this trajectory/

Outputs:

- trajectory - cluster - created trajectory data

PathPlanner_Trajectory_ReadCSVFile



Create a trajectory from a CSV file. This can be used on a PC or the RoboRIO. Normally the CSV file is created as output from one of the trajectory utility programs. The file could also be created manually or by a custom written program.

Parameters:

- FileName -- Name of the CSV file to read. See file name notes for additional information.

- Error In -- Input error cluster (optional)

Returns:

- outTrajectory - Trajectory data structure cluster

- Error out - returned error cluster

Notes on use:

-- This routine writes informational messges to the console and to the driver station log.

Notes on file naming:

-- The file name must include the extention. ".csv" is not automatically appended to the name.

-- The file name can be a simple file or an absolute path. If a simple file name is used the default path on the RoboRIO is: "home:\lvuser\natinst\LabVIEW Data". On a Windows PC the default path is the LabVIEW default directory. Normally this is: ❽%HOMEDRIVE%%HOMEPATH%\Documents\LabView Data".

-- Filenames on the RoboRIO, which runs Linux, are case sensitive.

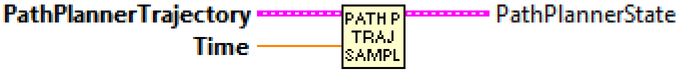
Notes on file contents:

-- Blank lines are ignored.

-- Lines that begin with either #, !, or ' in the first character are considered comments and are ignored.

-- Other lines are interpretted as comma separated data

PathPlanner_Trajectory_Sample



Get the target state at the given point in time along the trajectory

Inputs:

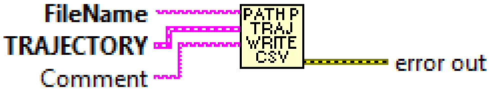
-- PathPlannerTraectory -- trajectory - PathPlanner Trajectory data cluster

-- time -- double - The time to sample

Outputs:

- PathPlannerState - trajectorystate - The state at the given point in time

PathPlanner_Trajectory_WriteCSVFile



Create a CSV file from a trajectory. This can be used on a PC or the RoboRIO.

Parameters:

- FileName -- Name of the CSV file to read. See file name notes for additional information.

- Trajectory - Trajectory data structure cluster

- Comment - string - Optional comment to place in CSV file.

Returns:

- Error out - returned error cluster

Notes on file naming:

-- The file name must include the extention. ".csv" is not automatically appended to the name.

-- The file name can be a simple file or an absolute path. If a simple file name is used the default path on the RoboRIO is: "home:\lvuser\natinst\LabVIEW Data". On a Windows PC the default path is the LabVIEW default directory. Normally this is: ❽%HOMEDRIVE%%HOMEPATH%\Documents\LabView Data".

-- Filenames on the RoboRIO, which runs Linux, are case sensitive.

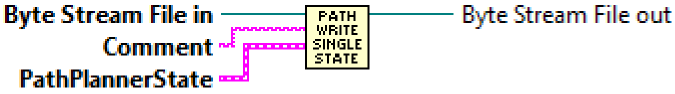
Notes on file contents:

-- Blank lines are ignored.

-- Lines that begin with either #, !, or ' in the first character are considered comments and are ignored.

-- Other lines are interpretted as comma separated data

PathPlanner_Trajectory_WriteCSVFileIndividualState



Internal subVI used by Util_Trajectory_WriteFile (and others). This writes one trajectory state to a file.

Parameters:

- Byte stream in - file stream

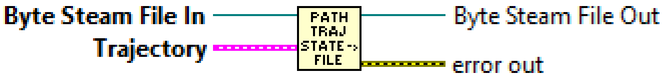
- comment - comment for this line

- TrajectoryState - The state to write

Returns:

- Byte Stream Out - file stream

PathPlanner_Trajectory_WriteCSVFileStates



Write trajectory states to a file. This is an internal routine

Parameters:

- ByteStreamIn - File stream

- Trajectory - Data structure containing trajectory

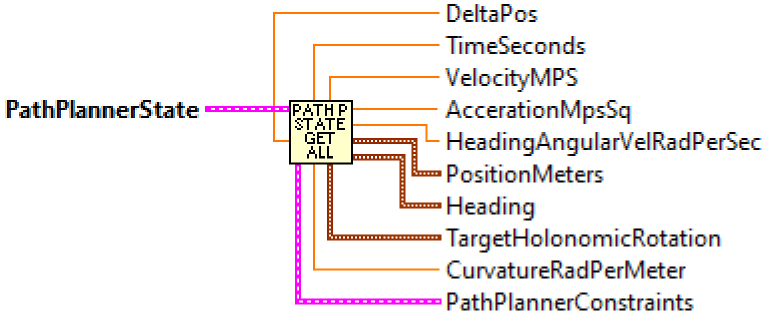
Returns:

- ByteStreamOut - File stream

- Error out - returned error cluster

TrajectoryState

PathPlanner_TrajectoryState_GetAll



Gets elements of trajectory state

Inputs

- PathPlannerTrajectoryState -- cluster -- State data structure

Outputs:

-- timeSeconds - double - The time at this state in seconds (default = 0;)

-- velocityMps - double - The velocity at this state in m/s (default = 0)

-- accelerationMpsSq - double - The acceleration at this state in m/s^2 (default = 0)

-- headingAngularVelocityRps - double - The time at this state in seconds (default = 0)

-- positionMeters - translation2d - The position at this state in meters (default = 0,0)

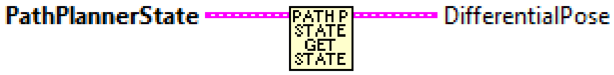
-- heading - rotation2d - The heading (direction of travel) at this state (default = 0)

-- targetHolonomicRotation - rotation2d - The target holonomic rotation at this state (default = 0)

-- curvatureRadPerMeter - double - The curvature at this state in rad/m (default = 0)

-- constraints -- cluster -- constraints to apply at this state (default - none)

PathPlanner_TrajectoryState_GetDifferentialPose



Get this pose for a differential drivetrain

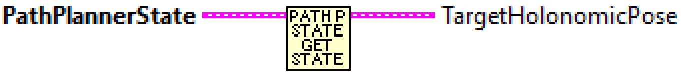
Inputs:

- trajectoryState - cluster - this trajectory state

Outputs:

- DifferentialPose - pose2d - The pose

PathPlanner_TrajectoryState_GetTargetHolonomicPose



Get the target pose for a holonomic drivetrain NOTE: This is a "target" pose, meaning the rotation will be the value of the next rotation target along the path, not what the rotation should be at the start of the path

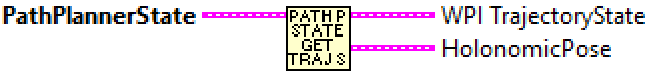
Inputs:

- trajectoryState - cluster - this trajectory state

Outputs:

- TargetHolonomicPose - pose2d - he target pose

PathPlanner_TrajectoryState_GetWPITrajectoryState



Get Trajectory Library / WPILIB trajectory state from a PathPlanner Trajectory State

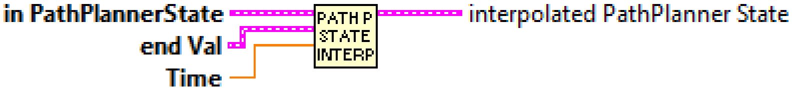
Inputs:

-- PathPlannerState -- Path Planner trajectory state

Outputs:

-- TrajectoryState -- LabVIEW trajectory library / WPILib trajectory state.

PathPlanner_TrajectoryState_Interpolate



Interpolate between this state and the given state

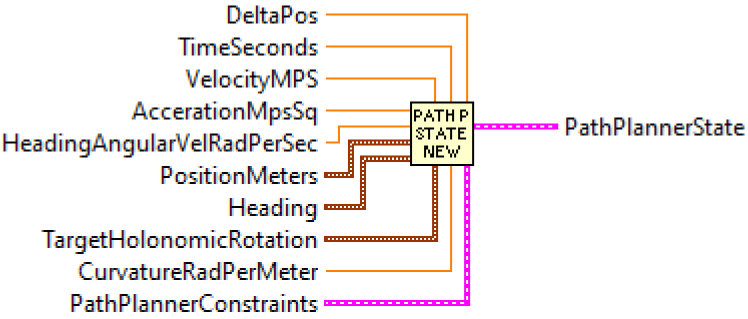
Inputs:

- trajectoryState - cluster - this trajectory state
- endVal - cluster - State to interpolate with
- t - double - Interpolation factor (0.0-1.0)

Outputs:

- Interpolated state - trajectory state - interpolated state

PathPlanner_TrajectoryState_New



Create a trajectory state

Inputs:

-- timeSeconds - double - The time at this state in seconds (default = 0;)

-- velocityMps - double - The velocity at this state in m/s (default = 0)

-- accelerationMpsSq - double - The acceleration at this state in m/s^2 (default = 0)

-- headingAngularVelocityRps - double - The time at this state in seconds (default = 0)

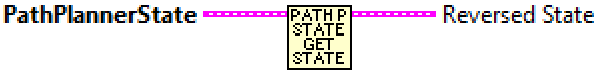
-- positionMeters - translation2d - The position at this state in meters (default = 0,0)

- -- heading - rotation2d - The heading (direction of travel) at this state (default = 0)
- -- targetHolonomicRotation - rotation2d - The target holonomic rotation at this state (default = 0)
- -- curvatureRadPerMeter - double - The curvature at this state in rad/m (default = 0)
- -- constraints -- cluster -- constraints to apply at this state (default - none)

Outputs

- PathPlannerTrajectoryState -- cluster -- Newly created state

PathPlanner_TrajectoryState_Reverse



Get the state reversed, used for following a trajectory reversed with a differential drivetrain

Inputs:

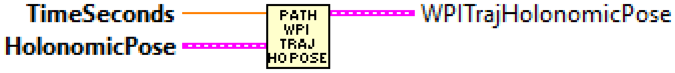
- trajectoryState - cluster - this trajectory state

Outputs:

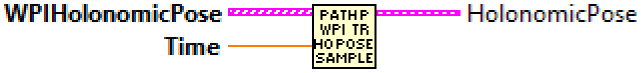
- ReversedState- trajectorystate - The reversed state

WPITrajHolPose

PathPlanner_WPITrajHolPose_New



PathPlanner_WPITrajHolPose_Sample



Sample the path at a point in time

Inputs:

-- PathPlannerTraectory -- PathPlanner Trajectory data cluster

-- time -- The time to sample

Outputs:

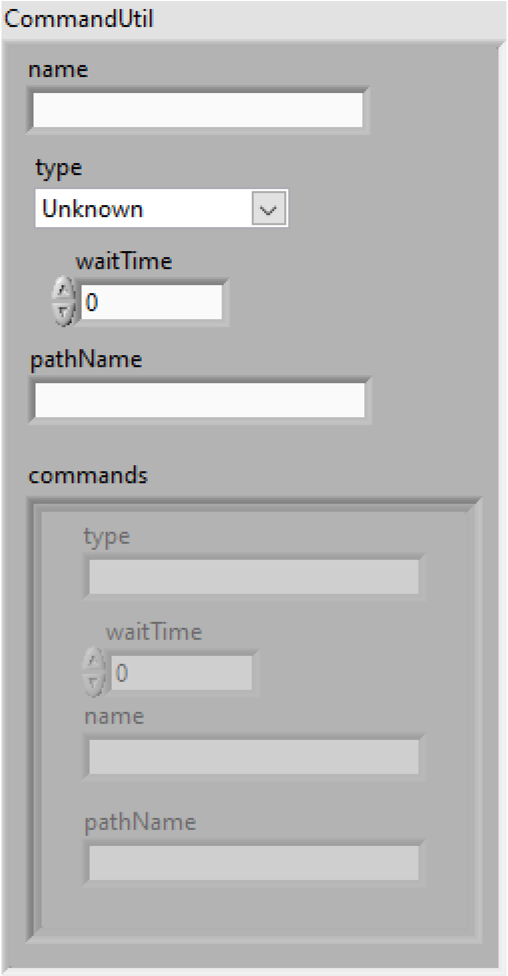
- PathPlannerState -- The state at the given point in time

Type Definitions

TypeDef

TypeDef-PathPlannerCommandUtil

PATH P
CMD



CommandUtil

name

type: Unknown

waitTime: 0

pathName

commands

type

waitTime: 0

name

pathName

TypeDef-PathPlannerConstraintsZone

PATH P
CNSTR
ZONE

A zone on a path with different kinematic constraints

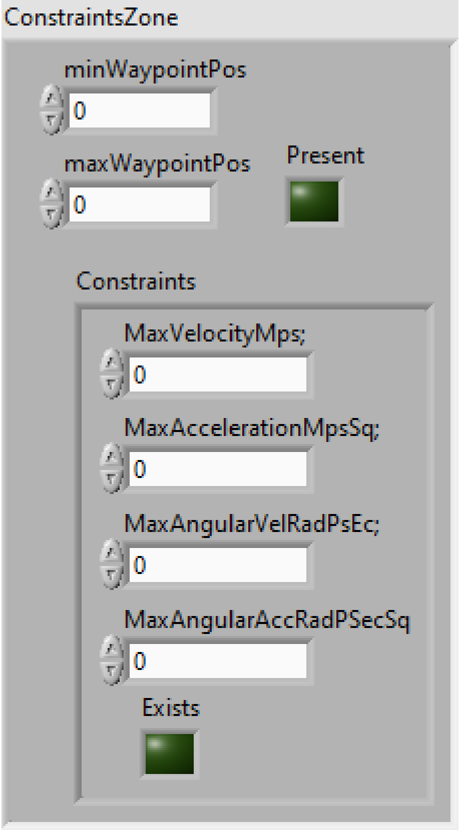
Contains:

-- MinWayPointPos - double - Starting distance on path to apply constraint

-- MaxWayPointPos - double - Ending distance on path to apply constraint

-- Constraint - cluster - Constraint to apply

-- Present - boolean - flag indicting this cluster is not null



ConstraintsZone

minWaypointPos 0

maxWaypointPos 0

Present

Constraints

MaxVelocityMps; 0

MaxAccelerationMpsSq; 0

MaxAngularVelRadPsEc; 0

MaxAngularAccRadPSecSq 0

Exists

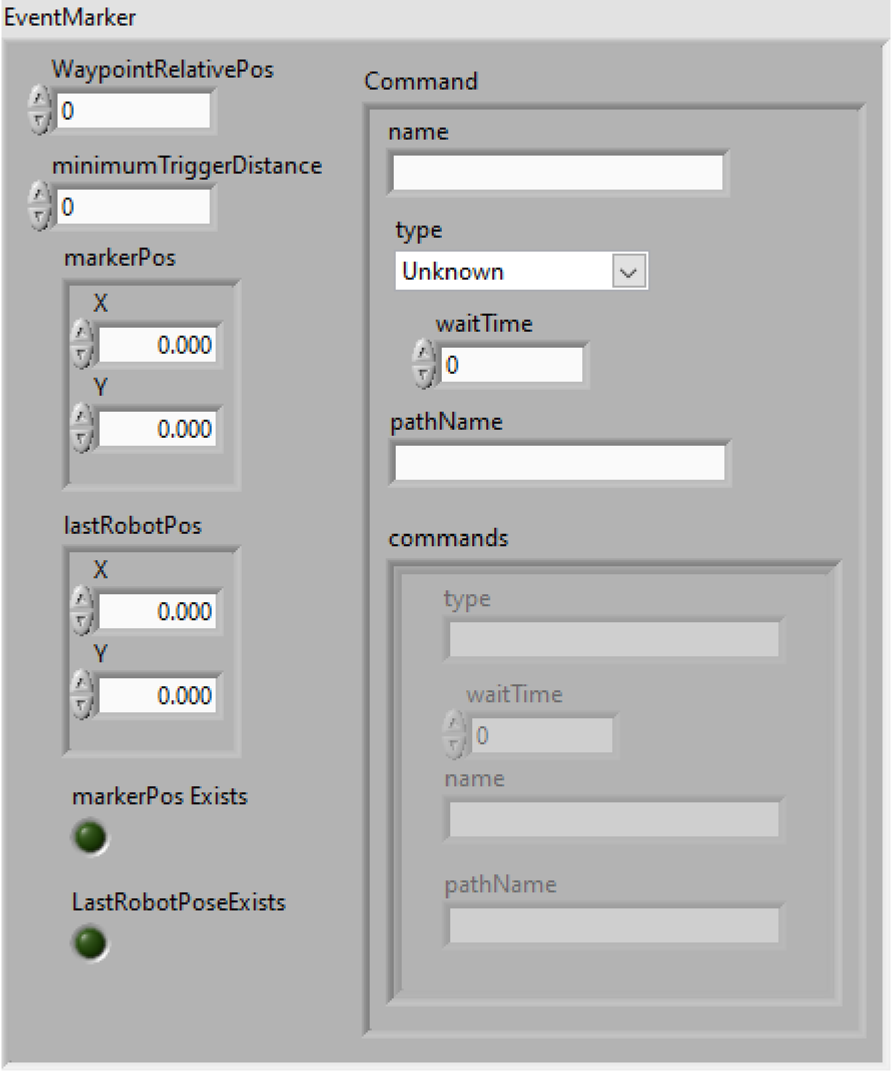
TypeDef-PathPlannerEventMarker

PATH P
EVENT
MARK

Position along the path that will trigger a command when reached

Contains

- WayPointRelativePose - double
- Command - cluster
- MinimumTriggerDistance - double
- MarkerPos - translation2d
- LastRobotPos - translation2d



EventMarker

WaypointRelativePos: 0

minimumTriggerDistance: 0

markerPos
X: 0.000
Y: 0.000

lastRobotPos
X: 0.000
Y: 0.000

markerPos Exists ●

LastRobotPoseExists ●

Command
name:
type: Unknown
waitTime: 0
pathName:

commands
type:
waitTime: 0
name:
pathName:

TypeDef-PathPlannerGoalEndState

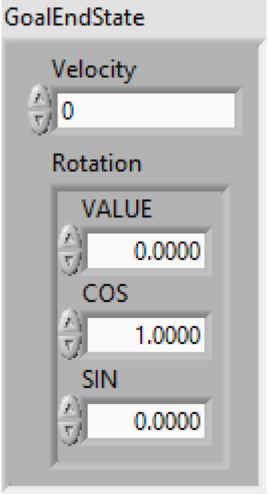
PATH P
END
STATE

Describes the goal end state of the robot when finishing a path

contains:

-- velocity - double

-- rotation - Rotation2d



GoalEndState

Velocity: 0

Rotation

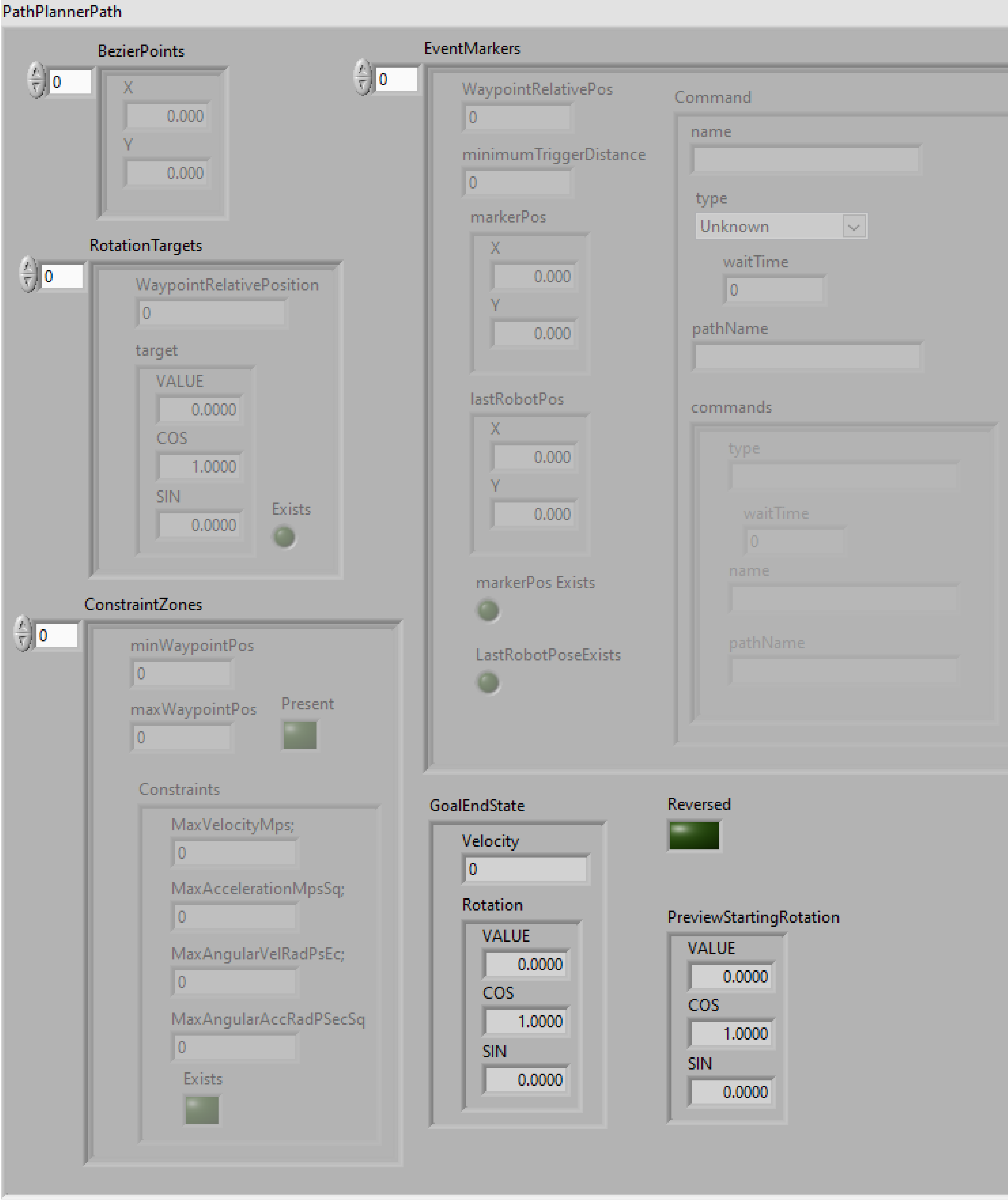
VALUE: 0.0000

COS: 1.0000

SIN: 0.0000

TypeDef-PathPlannerPath

PATH P
PATH



PathPlannerPath

BezierPoints
0
X
0.000
Y
0.000

RotationTargets
0
WaypointRelativePosition
0
target
VALUE
0.0000
COS
1.0000
SIN
0.0000
Exists

ConstraintZones
0
minWaypointPos
0
maxWaypointPos
0
Present
Constraints
MaxVelocityMps;
0
MaxAccelerationMpsSq;
0
MaxAngularVelRadPsEc;
0
MaxAngularAccRadPSecSq
0
Exists

EventMarkers
0
WaypointRelativePos
0
minimumTriggerDistance
0
markerPos
X
0.000
Y
0.000
lastRobotPos
X
0.000
Y
0.000
markerPos Exists
LastRobotPoseExists

Command
name
type
Unknown
waitTime
0
pathName
commands
type
waitTime
0
name
pathName

GoalEndState
Velocity
0
Rotation
VALUE
0.0000
COS
1.0000
SIN
0.0000

Reversed

PreviewStartingRotation
VALUE
0.0000
COS
1.0000
SIN
0.0000

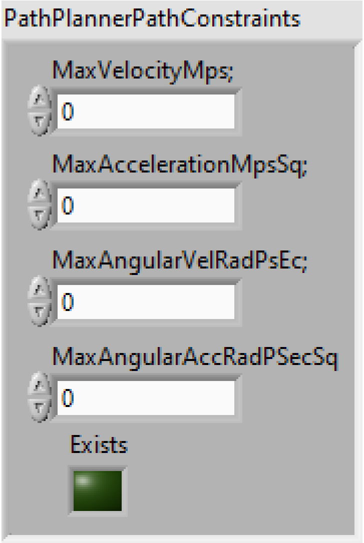
TypeDef-PathPlannerPathConstraints

PATH P
CONST

Kinematic path following constraints

Contains:

- Max Velocity (Meters/Second)
- Max Acceleration (Meters/Second^2)
- Max Angular Velocity (Radians/Second)
- Max Angular Acceleration (Radians/Second^2)
- Exists - boolean - flag indicating this data is not NULL



PathPlannerPathConstraints

MaxVelocityMps; 0

MaxAccelerationMpsSq; 0

MaxAngularVelRadPsEc; 0

MaxAngularAccRadPSecSq 0

Exists

TypeDef-PathPlannerPathPoint

PATH P
POINT

A point along a pathplanner path

Contains:

-- position - translation2d - The position of this point

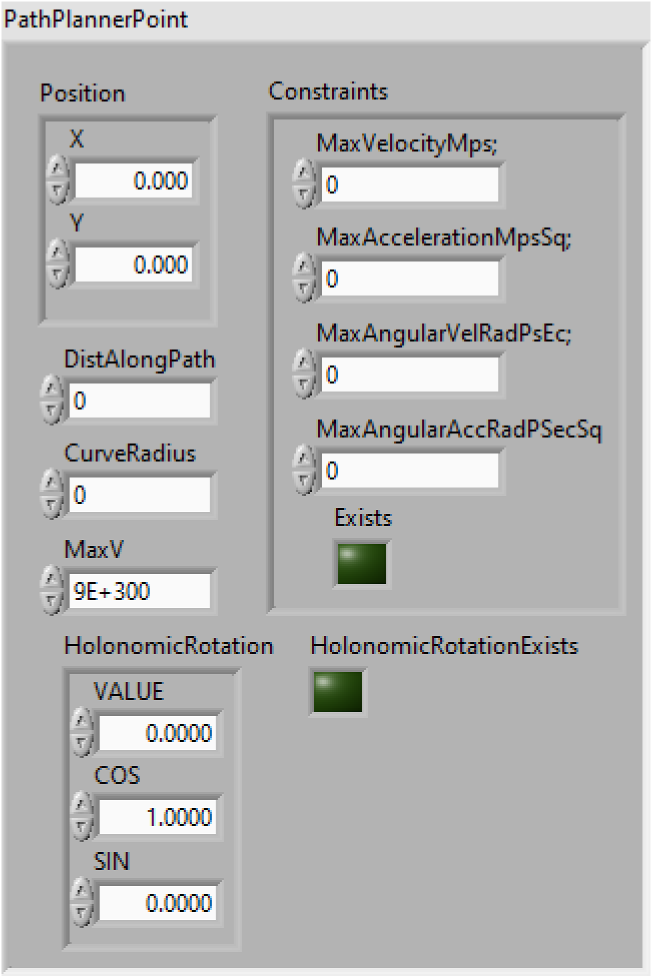
-- distanceAlongPath - double - The distance of this point along the path, in meters

-- CurveRadius - double - The curve radius at this point

-- MaxV - double - The max velocity at this point

-- holonomicRotation - Rotation2d - The target rotation at this point

-- constraints - cluster - The constraints applied to this point



PathPlannerPoint

Position
X 0.000
Y 0.000

DistAlongPath 0

CurveRadius 0

MaxV 9E+300

Constraints
MaxVelocityMps; 0
MaxAccelerationMpsSq; 0
MaxAngularVelRadPsEc; 0
MaxAngularAccRadPSecSq 0
Exists ●

HolonomicRotation
VALUE 0.0000
COS 1.0000
SIN 0.0000

HolonomicRotationExists ●

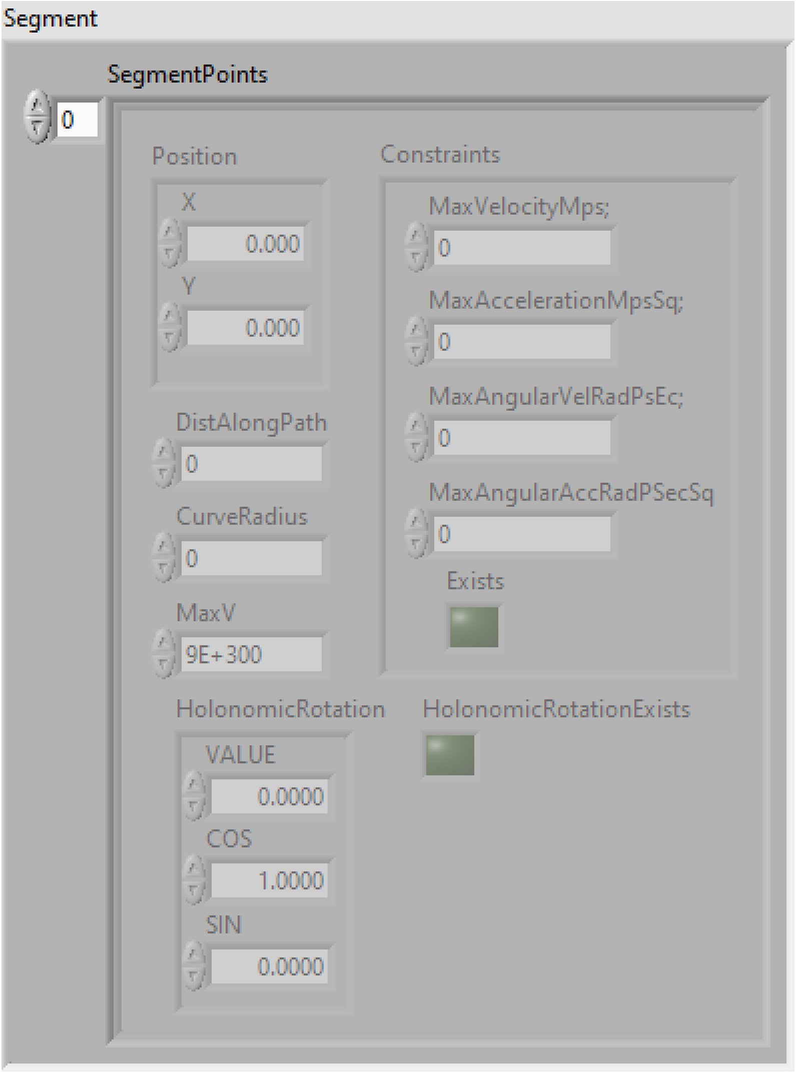
TypeDef-PathPlannerPathSegment

PATH P SEG-MENT

A bezier curve segment

Contains:

- SegmentPoints - array - Array of PathPoints



Segment

SegmentPoints

0

Position

X 0.000

Y 0.000

DistAlongPath 0

CurveRadius 0

MaxV 9E+300

Constraints

MaxVelocityMps; 0

MaxAccelerationMpsSq; 0

MaxAngularVelRadPsEc; 0

MaxAngularAccRadPSecSq 0

Exists

HolonomicRotation

VALUE 0.0000

COS 1.0000

SIN 0.0000

HolonomicRotationExists

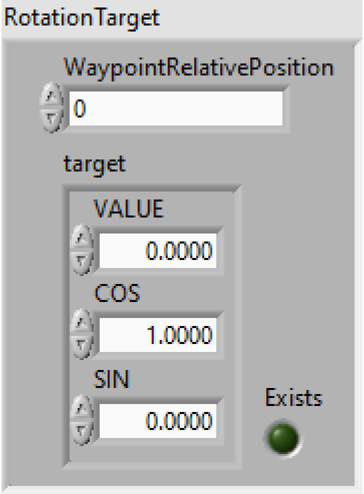
TypeDef-PathPlannerRotationTarget

PATH P
ROT
TARG

A target holonomic rotation at a position along a path

Contains:

- waypointRelativePosition - double
- target - rotation2d
- exists - boolean - TRUE if not null



RotationTarget

WaypointRelativePosition: 0

target

VALUE: 0.0000

COS: 1.0000

SIN: 0.0000

Exists ●

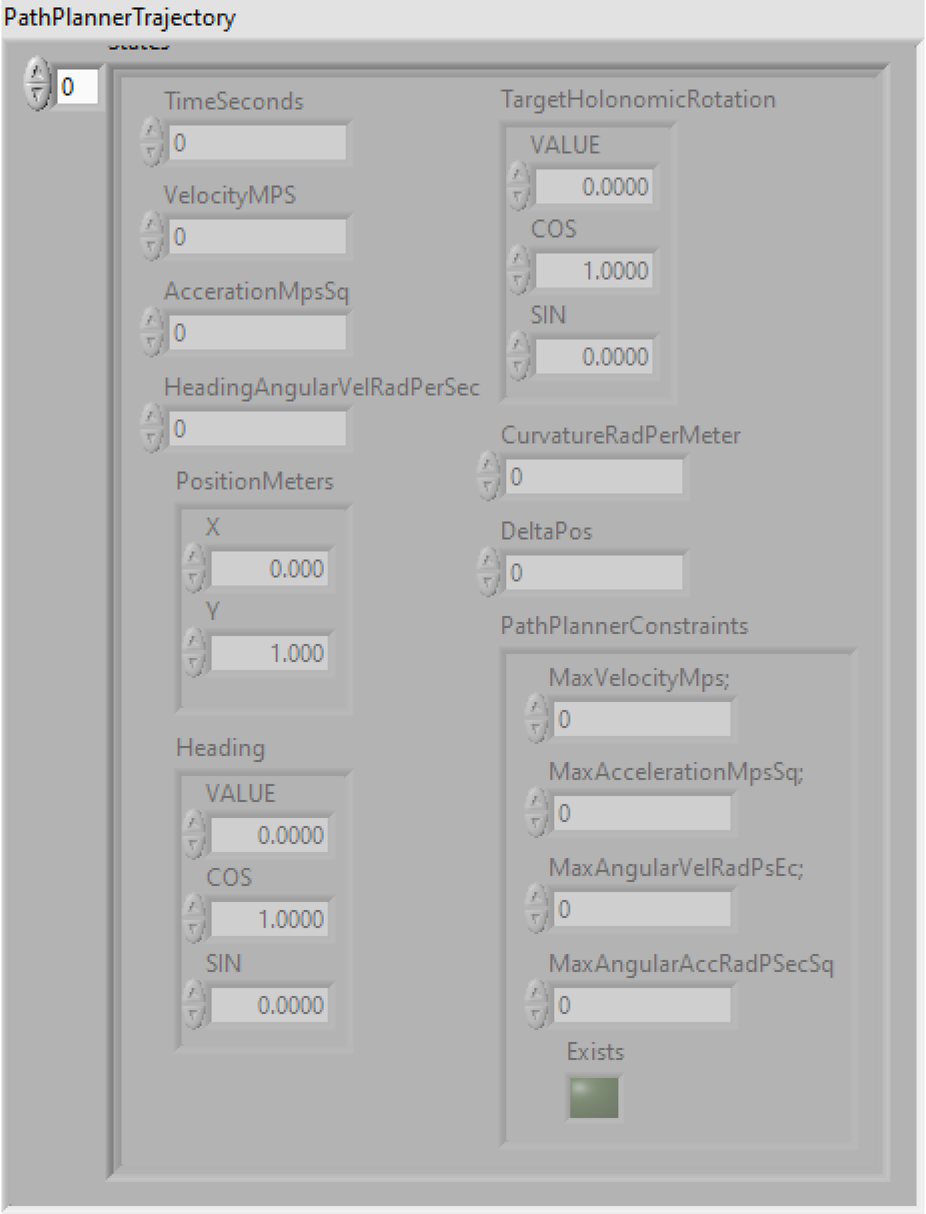
TypeDef-PathPlannerTrajectory

PATH P
TRAJ

Trajectory created from a pathplanner path

Contains:

- States - array - List of trajectory states



PathPlannerTrajectory

States

0

TimeSeconds: 0

VelocityMPS: 0

AccerationMpsSq: 0

HeadingAngularVelRadPerSec: 0

PositionMeters
X: 0.000
Y: 1.000

Heading
VALUE: 0.0000
COS: 1.0000
SIN: 0.0000

TargetHolonomicRotation
VALUE: 0.0000
COS: 1.0000
SIN: 0.0000

CurvatureRadPerMeter: 0

DeltaPos: 0

PathPlannerConstraints
MaxVelocityMps;: 0
MaxAccelerationMpsSq;: 0
MaxAngularVelRadPsEc;: 0
MaxAngularAccRadPSecSq: 0
Exists

TypeDef-PathPlannerTrajectoryState

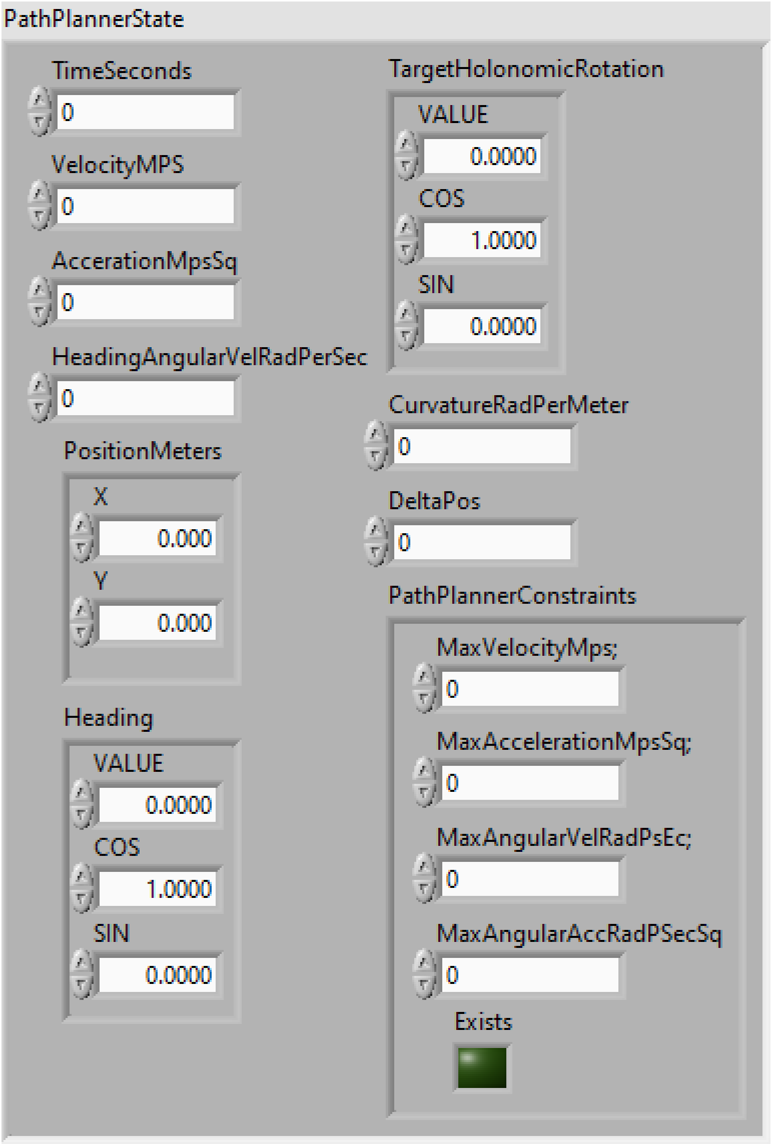
PATH P
TRAJ
STATE

A state along the trajectory

Contains:

- timeSeconds - double - The time at this state in seconds
- velocityMps - double - The velocity at this state in m/s
- accelerationMpsSq - double - The acceleration at this state in m/s^2
- headingAngularVelociyRPS - double - The time at this state in seconds

- positionMeters - translation2d - The position at this state in meters
- heading - rotation2d - The heading (direction of travel) at this state
- targetHolonomicRotation - rotation2d - The target holonomic rotation (orientation) at this state
- curvatureRadPerMeter - double - The curvature at this state in rad/m
- constraints - pathconstraints - The constraints to apply at this state
- deltaPos - double - Values only used during generation



PathPlannerState

TimeSeconds: 0

VelocityMPS: 0

AccerationMpsSq: 0

HeadingAngularVelRadPerSec: 0

PositionMeters
- X: 0.000
- Y: 0.000

Heading
- VALUE: 0.0000
- COS: 1.0000
- SIN: 0.0000

TargetHolonomicRotation
- VALUE: 0.0000
- COS: 1.0000
- SIN: 0.0000

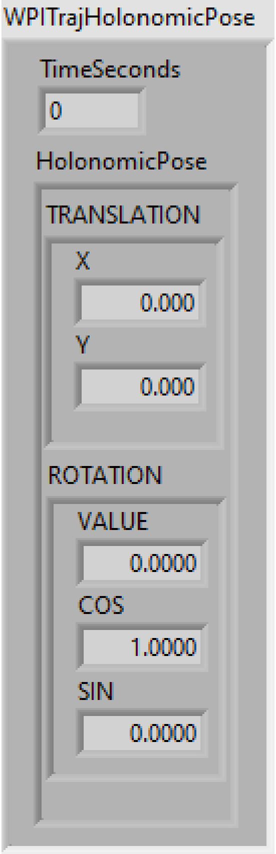
CurvatureRadPerMeter: 0

DeltaPos: 0

PathPlannerConstraints
- MaxVelocityMps;: 0
- MaxAccelerationMpsSq;: 0
- MaxAngularVelRadPsEc;: 0
- MaxAngularAccRadPSecSq: 0
- Exists: ●

TypeDef-PathPlannerWPITrajHolonomicPose

WPI
TRAJ
HOL
POSE



WPITrajHolonomicPose

TimeSeconds
0

HolonomicPose

TRANSLATION

X
0.000

Y
0.000

ROTATION

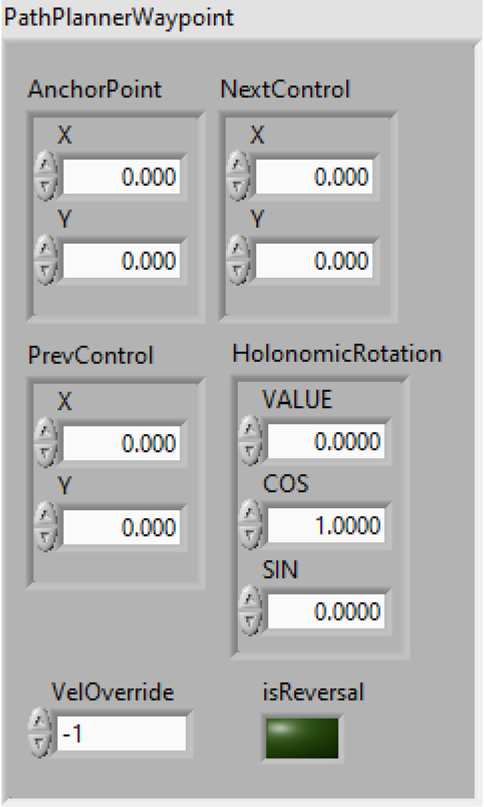
VALUE
0.0000

COS
1.0000

SIN
0.0000

TypeDef-PathPlannerWaypoint

PATH P
WAY-
POINT



PathPlannerWaypoint

AnchorPoint
X 0.000
Y 0.000

NextControl
X 0.000
Y 0.000

PrevControl
X 0.000
Y 0.000

HolonomicRotation
VALUE 0.0000
COS 1.0000
SIN 0.0000

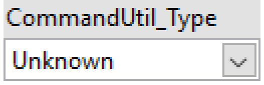
VelOverride -1

isReversal

Enumerated Type Definitions

Enum

Enum-PathPlanner_CommandUtilType_ENUM



CommandUtil_Type

Unknown