

# **PathPlannerLib**

## **LabVIEW**

## **Reference**

## Table of Contents

Introduction.....	3
Function Help.....	3
Function Examples.....	3
Function Groups.....	4
CommandUtil.....	5
ConstraintsZone.....	6
Ctrl.....	10
EventMarker.....	11
GeomUtil.....	15
GoalEndState.....	19
Path.....	21
PathConstraints.....	33
PathPoint.....	36
PathSegment.....	38
RotationTarget.....	40
Trajectory.....	43
TrajectoryState.....	52
WPITrajHolPose.....	56
Type Definitions.....	57
TypeDef.....	58
Enumerated Type Definitions.....	71
Enum.....	72

# 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

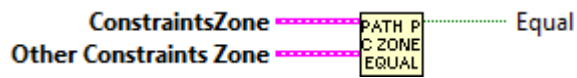
Outputs:

- type - enum - Evaluated command util type.

# ConstraintsZone

---

## PathPlanner\_ConstraintsZone\_Equals



Compares two Constraints Zone definitions

Inputs:

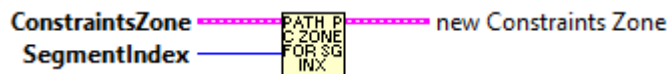
- ConstraintsZone - cluster - first definition to compare
- Other ConstraintsZone - cluster - other definition to compare

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]

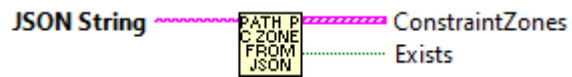
Inputs:

- 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.

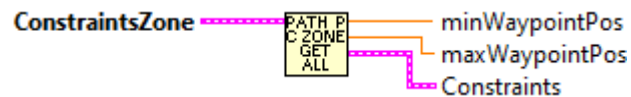
Outputs:

-- 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:

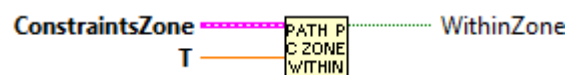
-- minWayPointPos - double - Waypoint relative starting position

-- maxWayPointPos - double - Waypoint relative end position

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

---

## PathPlanner\_ConstraintsZone\_IsWithinZone



Get if a given waypoint relative position is within this zone

Inputs:

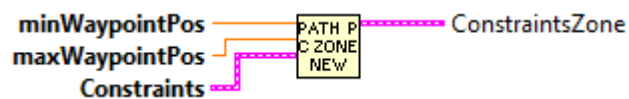
- ConstraintsZone -- cluster -- data structure containing zone definition.
- t - double - Waypoint relative position

Outputs:

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

---

## PathPlanner\_ConstraintsZone\_New



Create a new constraints zone

Inputs:

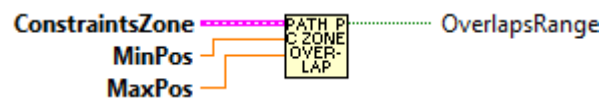
- minWaypointPos - double - Starting position of the zone
- maxWaypointPos - double - End position of the zone
- constraints - cluster - The constraints to apply within the zone

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

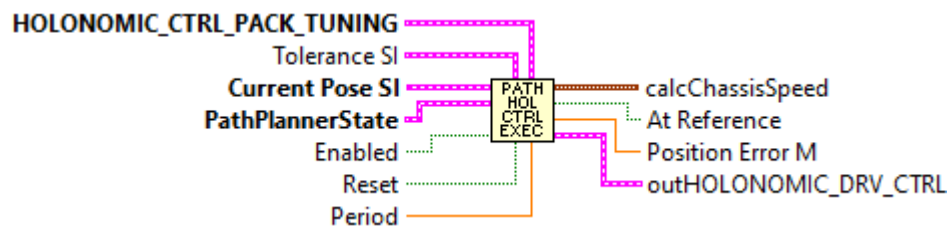
Outputs:

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

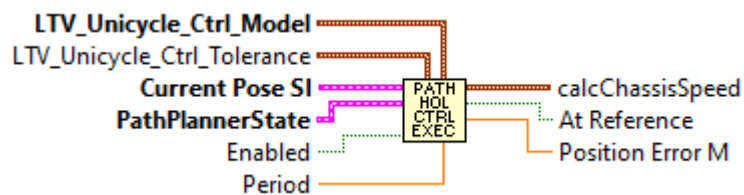
# Ctrl

---

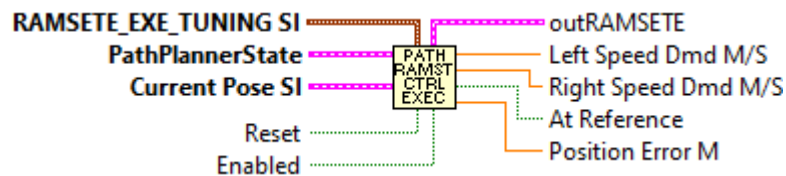
## PathPlanner\_Ctrl\_HolonomicDrvExecute



## PathPlanner\_Ctrl\_LTVExecute



## PathPlanner\_Ctrl\_RamseteExecute



# EventMarker

---

## PathPlanner\_EventMarker\_Equals



Determines if 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

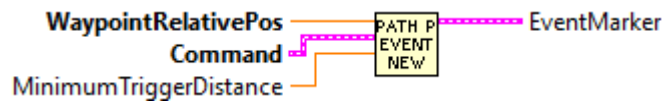
Inputs:

- 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

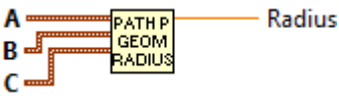
Outputs:

- 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



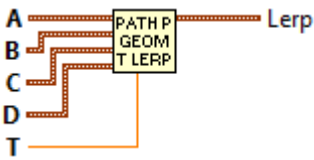
---

## PathPlanner\_GeomUtil\_CoerceHeadingRadians



---

## PathPlanner\_GeomUtil\_CubicLerp



Cubic interpolation between Translation2ds

Inputs:

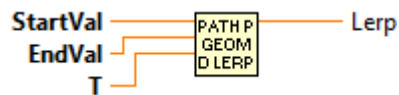
- a - translation2d - Position 1
- b - translation2d - Position 2
- c - translation2d - Position 3
- t - double - Interpolation factor (0.0-1.0)

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)

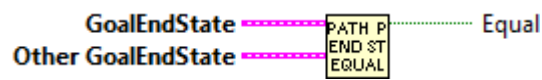
Outputs:

- 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



Create the bezier points necessary to create a path using a list of poses

Inputs:

- poses - pose2d array - List of poses. Each pose represents one waypoint.

Outputs:

- Bezier - translation2d array - List of bezier points
- Error - boolean - TRUE if an error occurred. (Too few poses)

---

## PathPlanner\_Path\_BezierFromWaypointsJSON



Parse bezier points from a JSON string formatted as waypoint.

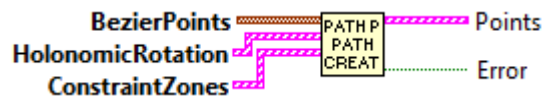
Inputs:

- JSON String - string - JSON containing waypoint to parse and convert to bezier point array

Outputs:

- Bezier - array of translation2s - List of bezier points
  - error - boolean - TRUE if an error occurred.
  - value - array of cluster - bezier points -- for debugging
-

## PathPlanner\_Path\_CreatePath



Create the path points for this path. This is an internal function.

Inputs:

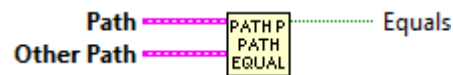
- `BezierPoints` - array of bezier oints
- `HolonomicRotations` - array of holonomic rotatios.
- `ConstraintZones` - array of constraint zones.

Outputs:

- `PathPoints` - `PathPoint` array - Array of points along the path
- `Error` - boolean - TRUE if an error occured.

---

## PathPlanner\_Path\_Equals



Determines if two paths are identical.

Note: `Reversed` and `PreviewEndState` are not part of the comparison

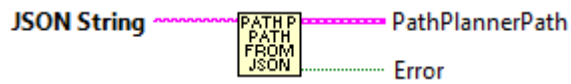
Inputs:

- `Path` `Path` - Data structure containing path definition
- `OtherPath` - `Path` - Data structure containing path definition

Outputs:

- `Equal` - boolean - TRUE if paths are identical.
-

## PathPlanner\_Path\_FromJSON



Load a path from a JSON string.

Inputs:

- JSON String - string- The string containing the path definition.

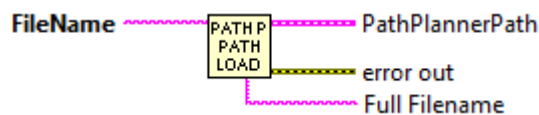
Outputs:

- Path - cluster - PathPlannerPath created from the given JSON string
- Error out - cluster - Error cluster

---

---

## PathPlanner\_Path\_FromPathFile



Load a path from a path file in storage. The path normally has a .PATH extension. Internally this file is formatted as JSON.

Inputs:

- filename - string - The name of the path to load

Outputs:

- Path - cluster - PathPlannerPath created from the given file name
- Error out - cluster - Error cluster
- Full Filename - string - fully qualified file name.

Notes on file naming:

- The file name must include the extension. ".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: **8**%HOMEDRIVE%%HOMEPATH%\Documents\LabView Data".
- Filenames on the RoboRIO, which runs Linux, are case sensitive.

---

---

## PathPlanner\_Path\_FromPathPonts



Create a path with pre-generated points. This should already be a smooth path.

Inputs:

- pathPoints - Path points along the smooth curve of the path
- constraints - The global constraints of the path
- goalEndState - The goal end state of the path

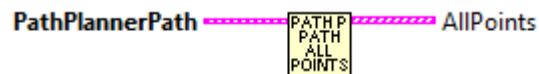
Outputs:

- Path - cluster - A PathPlannerPath following the given pathpoints

---

---

## PathPlanner\_Path\_GetAllPathPoint



Get all the path points in this path

Inputs:

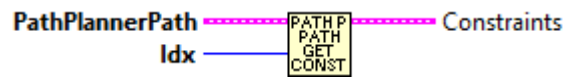
- Path - cluster - path definition data structure

Outputs:

- AllPoints - PathPoint array - Path points in the path
- 
-



## PathPlanner\_Path\_GetConstraintsForPoint



Get the constraints for a point along the path

Inputs:

- Path - cluster - path definition data structure
- idx - integer - Index of the point to get constraints for

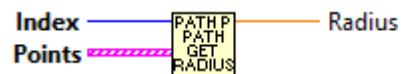
Outputs:

- Constraints - cluster - The constraints that should apply to the point

---

---

## PathPlanner\_Path\_GetCurveRadiusAtPoint



This is an internal function

Inputs:

- index
- Points

Outputs:

- Radius

---

---

## PathPlanner\_Path\_GetEventMarkers



Get all the event markers for this path

Inputs:

- Path - cluster - path definition data structure

Outputs:

- EventMarkers - cluster - The event markers for this path

---

---

## PathPlanner\_Path\_GetGlobalConstraints



Get the global constraints for this path

Inputs:

- Path - cluster - path definition data structure

Outputs:

- GlobalConstraints - cluster - Global constraints that apply to this path

---

---

## PathPlanner\_Path\_GetGoalEndState



Get the goal end state of this path

Inputs:

- Path - cluster - path definition data structure

Outputs:

- GoalEndState - cluster - The goal end state

---

---

## PathPlanner\_Path\_GetPoint



Get a specific point along this path

Inputs:

- Path - cluster - path definition data structure
- index - integer - Index of the point to get

Outputs:

- Point - PathPoint - The point at the given index

---

---

## PathPlanner\_Path\_GetPreviewStartingHolonomicPose



Get the starting pose for the holonomic path based on the preview settings.

NOTE: This should only be used for the first path you are running, and only if you are not using an auto mode file. Using this pose to reset the robots pose between sequential paths will cause a loss of accuracy.

Inputs:

- Path - cluster - path definition data structure

Outputs:

- PreviewStartingPose - pose2d - Pose at the path's starting point

---

---

## PathPlanner\_Path\_GetStartingDifferentialPose



Get the differential pose for the start point of this path

Inputs:

- Path - cluster - path definition data structure

Outputs:

- StartingDifferentialPose - pose2d - Pose at the path's starting point

---

---

## PathPlanner\_Path\_HotReload



Hot reload the path. This is used internally.

Inputs:

- Path - cluster - path definition data structure
- JSON String - string - JSON string containing the new path to load

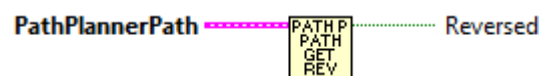
Outputs:

- Path - cluster - path definition data structure
- Error - boolean - TRUE if an error occurred.

---

---

## PathPlanner\_Path\_IsReversed



Should the path be followed reversed (differential drive only)

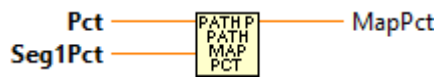
Inputs:

- Path - cluster - path definition data structure

Outputs:

- Reversed - boolean - True if reversed
- 
-

## PathPlanner\_Path\_MapPct



Map a given percentage/waypoint relative position over 2 segments This is an internal routine.

Inputs:

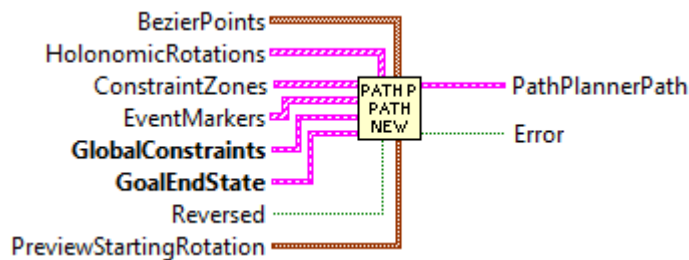
- pct - The percent to map
- seg1Pct - The percentage of the 2 segments made up by the first segment

Outputs:

- MapPct - The waypoint relative position over the 2 segments

---

## PathPlanner\_Path\_New



Create a new path planner path

You likely want to use bezierFromPoses to create the bezier points.

Inputs:

- bezierPoints - List of points representing the cubic Bezier curve of the path (Optional. Default: empty. Bezier points are necessary for creation of a valid path.)
- holonomicRotations - List of rotation targets along the path. (Optional. Default: empty)
- constraintZones - List of constraint zones along the path (Optional: Default: empty)
- eventMarkers - List of event markers along the path (Optional. Default: empty)
- globalConstraints - The global constraints of the path
- goalEndState - The goal end state of the path
- reversed - Should the robot follow the path reversed (differential drive only) (Optional. Default: false)
- previewStartingRotation - The settings used for previews in the UI (Optional. Default: 0 )

Outputs:

- Path - cluster - path definition data structure

---

---

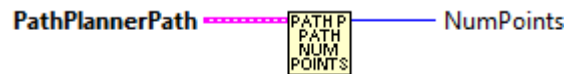
## PathPlanner\_Path\_New\_Empty



---

---

## PathPlanner\_Path\_NumPoints



Get the number of points in this path

Inputs:

- Path - cluster - path definition data structure

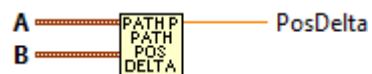
Outputs:

- NumPoints - integer - Number of points in the path

---

---

## PathPlanner\_Path\_PositionDelta



This is an internal routine

Inputs:

- A
- B

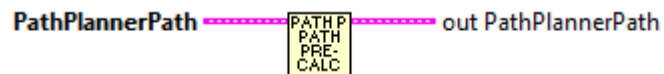
Outputs

- PosDelta

---

---

## PathPlanner\_Path\_PreCalcValues



This is an internal routine.

Inputs:

- Path - cluster - path definition data structure

Outputs:

- Path - cluster - path definition data structure
- 
- 

## PathPlanner\_Path\_RePlan



Hot reload the path. This is used internally.

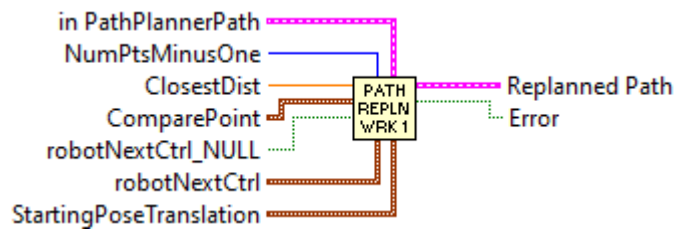
Inputs:

- Path - cluster - path definition data structure
- JSON String - string - JSON string containing the new path to load

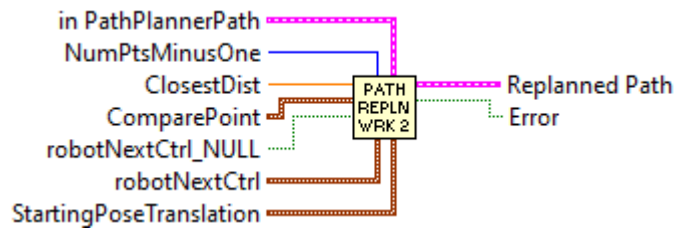
Outputs:

- Path - cluster - path definition data structure
  - Error - boolean - TRUE if an error occurred.
- 
-

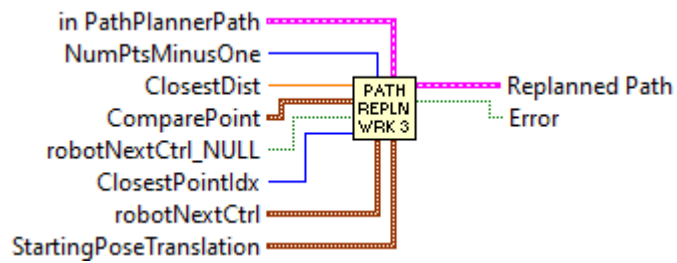
## PathPlanner\_Path\_RePlan\_Worker1



## PathPlanner\_Path\_RePlan\_Worker2



## PathPlanner\_Path\_RePlan\_Worker3

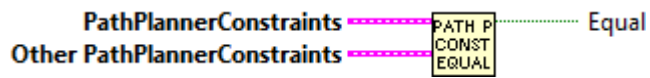




# PathConstraints

---

## PathPlanner\_PathConstraints\_Equals



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

Inputs:

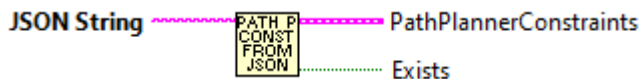
- `PathPlannerConstraints` - cluster - definition of path constraints
- `OtherPathPlannerConstraints` - cluster - definition of second path constraints for comparison

Outputs:

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

---

## PathPlanner\_PathConstraints\_FromJSON



Create a path constraints object from json string

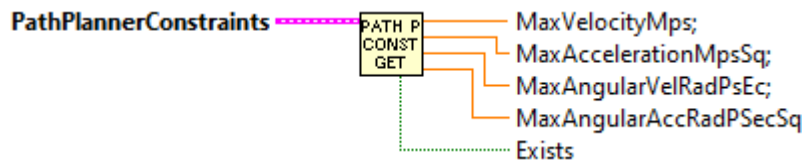
Inputs:

- `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

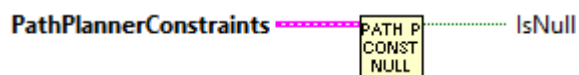
Outputs:

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

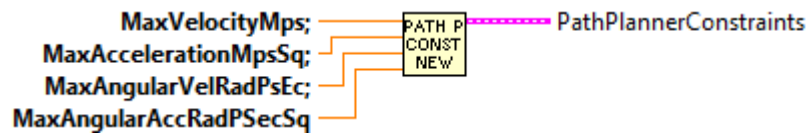
Inputs:

- 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<sup>2</sup>)
- maxAngularVelocityRps - double - Max angular velocity (Rad/S)
- maxAngularAccelerationRpsSq - double - Max angular acceleration (Rad/S<sup>2</sup>)

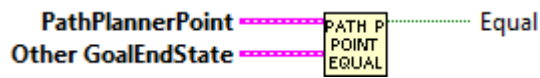
Outputs:

- PathConstraint - cluster - path constraint data

# PathPoint

---

## PathPlanner\_PathPoint\_Equals



Determines if two Path Point definitions are equal

Inputs:

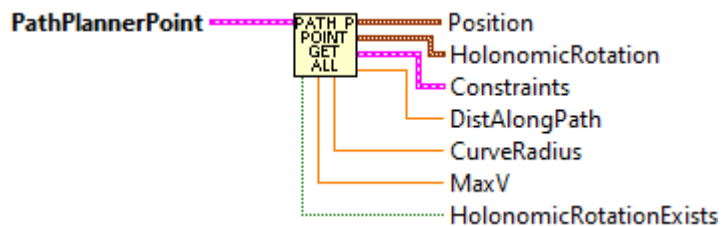
- PathPoint - cluster - point definition
- Other PathPoint - cluster - point definition

Outputs:

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

---

## PathPlanner\_PathPoint\_GetAll



Gets elements of PathPoint

Inputs:

- PathPoint - cluster - point definition

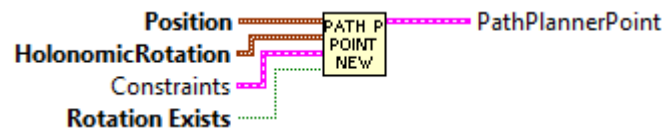
Outputs:

- Position - Translation2d - position of point
- HolonomicRotation - Rotation2d - rotational orientation of point
- Constraints - cluster - constraints at this point
- 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

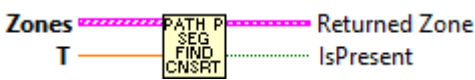
Outputs:

- 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

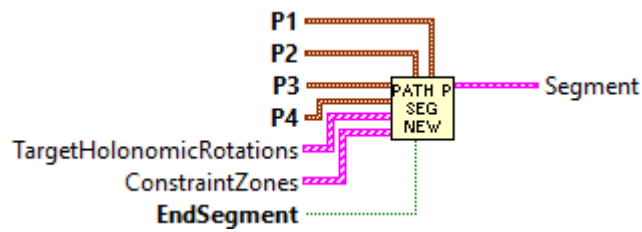
Inputs:

- 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

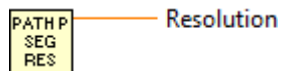
Outputs:

- 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:

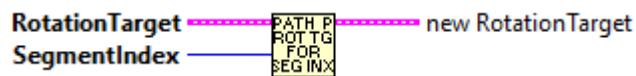
- RotationTarget - cluster - defined rotation target data structure
- OtherRotationTarget - cluster - defined rotation target data structure

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:

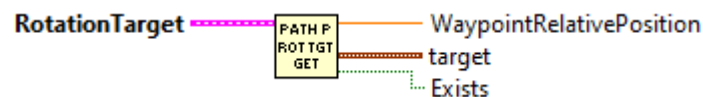
- JSON String - string - string potentially containing one or more of rotation target

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

---

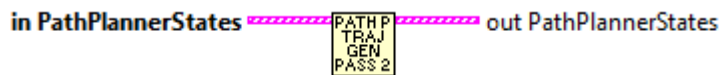
## PathPlanner\_Trajectory\_GenerateStates



## 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

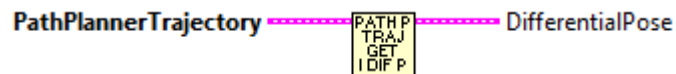
Outputs:

- 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

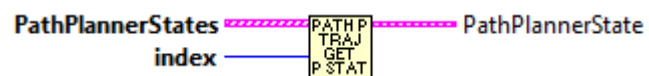
Outputs:

- NextRotationTargetIndex - integer -

---

---

## PathPlanner\_Trajectory\_GetState



Get the goal state at the given index

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

Inputs:

- Trajectory -- 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

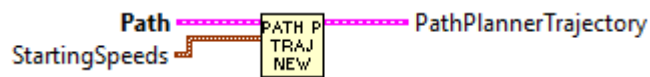
## Outputs

- Trajectory -- LabVIEW trajectory 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

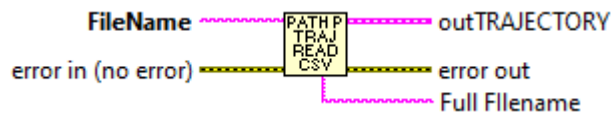
### Inputs:

- 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 messages to the console and to the driver station log.

### Notes on file naming:

- The file name must include the extension. ".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: **%HOMEDRIVE%%HOMEPATH%\Documents\LabView Data**.
- Filenames on the RoboRIO, which runs Linux, are case sensitive.

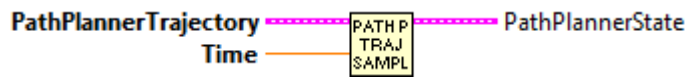
### Notes on file contents:

- Blank lines are ignored.
- Lines that begin with either #, !, or ' in the first character are considered comments and are ignored.
- Other lines are interpreted as comma separated data





## PathPlanner\_Trajectory\_Sample



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

Inputs:

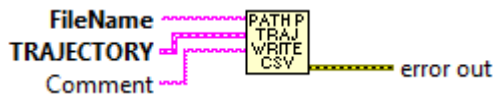
- PathPlannerTrajectory -- trajectory - PathPlanner Trajectory data cluster
- time -- double - The time to sample

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 extension. ".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: **8**%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 interpreted 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:

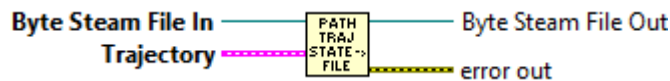
- Byte stream in - file stream
- comment - comment for this line
- TrajectoryState - The state to write

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:

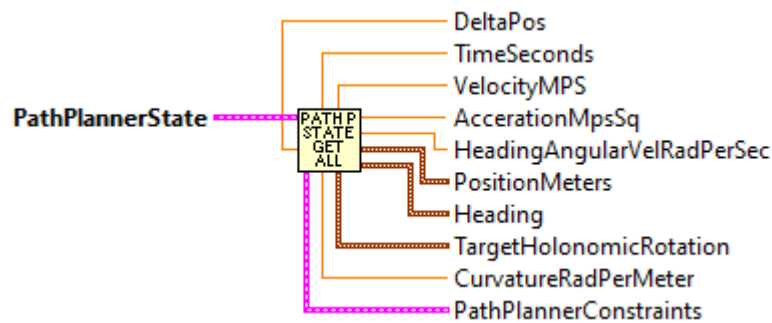
- ByteStreamOut - File stream

- 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<sup>2</sup> ( 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 - the target pose

---

---

## PathPlanner\_TrajectoryState\_GetWPITrajectoryState



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

Inputs:

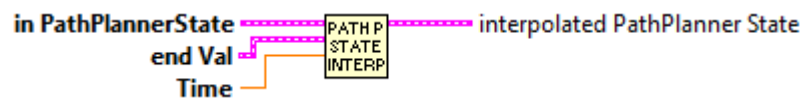
- PathPlannerState -- Path Planner trajectory state

Outputs:

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

---

## PathPlanner\_TrajectoryState\_Interpolate



Interpolate between this state and the given state

Inputs:

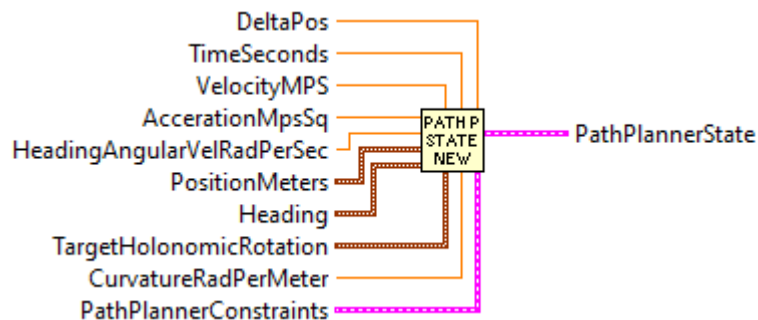
- trajectoryState - cluster - this trajectory state
- endVal - cluster - State to interpolate with
- t - double - Interpolation factor (0.0-1.0)

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<sup>2</sup> ( 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

**PathPlannerState** .....  ..... **Reversed State**

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

## Inputs:

- trajectoryState - cluster - this trajectory state

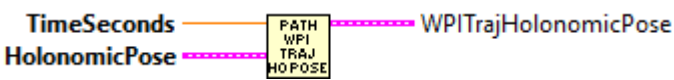
## Outputs:

- ReversedState- trajectorystate - The reversed state

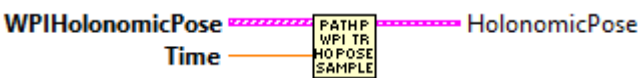
# WPITrajHolPose

---

## 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

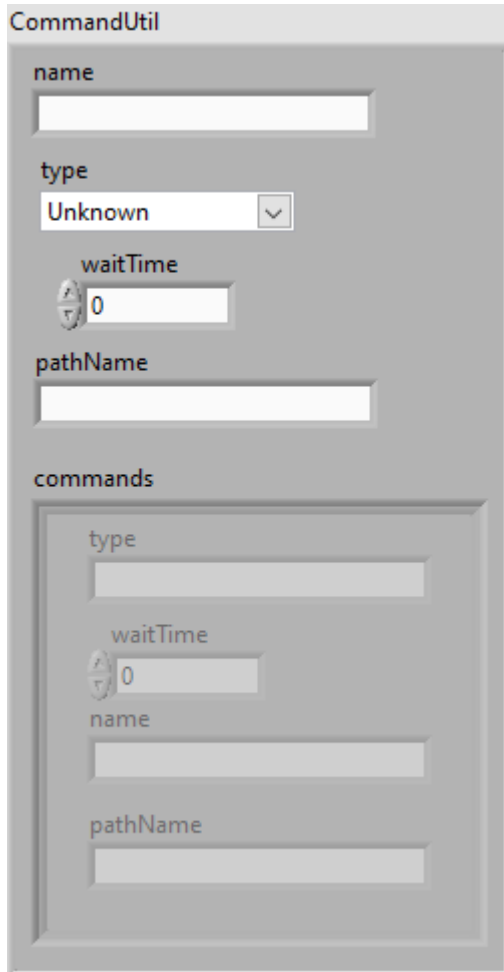
---

# TypeDef

---

## TypeDef-PathPlannerCommandUtil

PATH P  
CMD



The screenshot shows a GUI window titled "CommandUtil". It contains several input fields and a list of commands. The main fields are: "name" (text input), "type" (dropdown menu showing "Unknown"), "waitTime" (spin box showing "0"), and "pathName" (text input). Below these is a section titled "commands" which contains a list of command objects. Each command object has its own set of fields: "type" (text input), "waitTime" (spin box showing "0"), "name" (text input), and "pathName" (text input).

---

## 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

maxWaypointPos Present  
 ☒

Constraints

MaxVelocityMps;

MaxAccelerationMpsSq;

MaxAngularVelRadPsEc;

MaxAngularAccRadPSecSq

Exists  
☒

---

## TypeDef-PathPlannerEventMarker



Position along the path that will trigger a command when reached

Contains

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

EventMarker

WaypointRelativePos

minimumTriggerDistance

markerPos  
 X  
  
 Y

lastRobotPos  
 X  
  
 Y

markerPos Exists  
☒

LastRobotPoseExists  
☒

Command

name

type  
 Unknown

waitTime

pathName

commands

type

waitTime

name

pathName

---

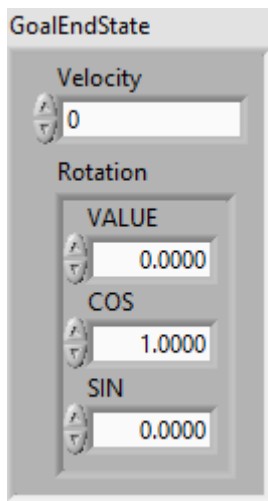
## TypeDef-PathPlannerGoalEndState



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

Detailed description: A vertical control panel titled 'GoalEndState'. It contains three sections. The first section is 'Velocity' with a numeric input field set to '0'. The second section is 'Rotation' and contains three sub-sections: 'VALUE' with a numeric input field set to '0.0000', 'COS' with a numeric input field set to '1.0000', and 'SIN' with a numeric input field set to '0.0000'. Each input field has small 'A' and 'V' buttons to its left.

---

## TypeDef-PathPlannerPath



A PathPlanner path. NOTE: This is not a trajectory and isn't directly followed.

Contains:

- bezierPoints - Translation2d array
- rotationTargets - RotationTarget array
- constraintZones - ConstraintsZone array
- eventMarkers - EventMarker array
- globalConstraints - PathConstraints
- goalEndState - GoalEndState
- allPoints - PathPoint array
- reversed - boolean
- previewStartingRotation - Rotation2d

## 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

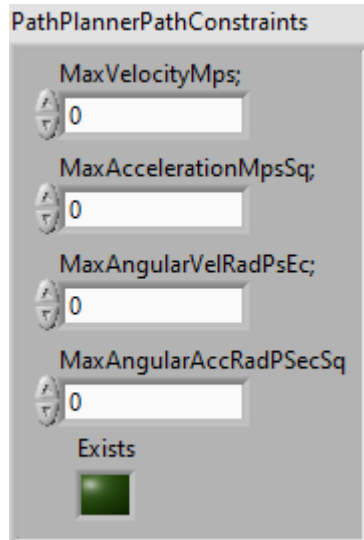
## TypeDef-PathPlannerPathConstraints



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



---

## TypeDef-PathPlannerPathPoint



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	Constraints
X 0.000	MaxVelocityMps; 0
Y 0.000	MaxAccelerationMpsSq; 0
DistAlongPath 0	MaxAngularVelRadPsEc; 0
CurveRadius 0	MaxAngularAccRadPSecSq 0
MaxV 9E+300	Exists <input checked="" type="checkbox"/>
HolonomicRotation	HolonomicRotationExists
VALUE 0.0000	<input checked="" type="checkbox"/>
COS 1.0000	
SIN 0.0000	

---

## TypeDef-PathPlannerPathSegment



A bezier curve segment

Contains:

- SegmentPoints - array - Array of PathPoints



Segment

SegmentPoints

0

Position	Constraints
X 0.000	MaxVelocityMps; 0
Y 0.000	MaxAccelerationMpsSq; 0
DistAlongPath 0	MaxAngularVelRadPsEc; 0
CurveRadius 0	MaxAngularAccRadPSecSq; 0
MaxV 9E+300	Exists <input type="checkbox"/>
HolonomicRotation VALUE 0.0000 COS 1.0000 SIN 0.0000	HolonomicRotationExists <input type="checkbox"/>

---

## 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

0

TimeSeconds	0	TargetHolonomicRotation	VALUE
VelocityMPS	0	COS	0.0000
AccerationMpsSq	0	SIN	1.0000
HeadingAngularVelRadPerSec	0	CurvatureRadPerMeter	0
PositionMeters		DeltaPos	0
X	0.000	PathPlannerConstraints	
Y	1.000	MaxVelocityMps;	0
Heading		MaxAccelerationMpsSq;	0
VALUE	0.0000	MaxAngularVelRadPsEc;	0
COS	1.0000	MaxAngularAccRadPSecSq	0
SIN	0.0000	Exists	<input checked="" type="checkbox"/>

## TypeDef-PathPlannerTrajectoryState



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
- headingAngularVelocityRPS - 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

<p>TimeSeconds</p> <input type="text" value="0"/>	<p>TargetHolonomicRotation</p> <p>VALUE</p> <input type="text" value="0.0000"/> <p>COS</p> <input type="text" value="1.0000"/> <p>SIN</p> <input type="text" value="0.0000"/>
<p>VelocityMPS</p> <input type="text" value="0"/>	<p>CurvatureRadPerMeter</p> <input type="text" value="0"/>
<p>AccerationMpsSq</p> <input type="text" value="0"/>	<p>DeltaPos</p> <input type="text" value="0"/>
<p>HeadingAngularVelRadPerSec</p> <input type="text" value="0"/>	<p>PathPlannerConstraints</p> <p>MaxVelocityMps;</p> <input type="text" value="0"/>
<p>PositionMeters</p> <p>X</p> <input type="text" value="0.000"/> <p>Y</p> <input type="text" value="0.000"/>	<p>MaxAccelerationMpsSq;</p> <input type="text" value="0"/>
<p>Heading</p> <p>VALUE</p> <input type="text" value="0.0000"/> <p>COS</p> <input type="text" value="1.0000"/> <p>SIN</p> <input type="text" value="0.0000"/>	<p>MaxAngularVelRadPsEc;</p> <input type="text" value="0"/>
	<p>MaxAngularAccRadPSecSq</p> <input type="text" value="0"/>
	<p>Exists</p> <input checked="" type="checkbox"/>

## 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- \_Obsolete\_PathPlannerWaypoint

PATH P  
WAY-  
POINT

# PathPlannerWaypoint

AnchorPoint

X

Y

NextControl

X

Y

PrevControl

X

Y

HolonomicRotation

VALUE

COS

SIN

VelOverride

isReversal



# Enumerated Type Definitions

---

# Enum

---

## Enum-PathPlanner\_CommandUtilType\_ENUM



CommandUtil\_Type

Unknown 