



PathPlannerLib

LabVIEW

Reference

Table of Contents

.....	1
Introduction.....	3
Function Menus.....	3
Function Help.....	4
Function Examples.....	5
Function Groups.....	6
Cmd.....	7
CommandUtil.....	17
ConstraintsZone.....	19
Ctrl.....	21
EventMarker.....	26
FieldUtil.....	31
GeomUtil.....	34
GoalEndState.....	38
GridPosition.....	41
IdealStartingState.....	43
ObstacleGrid.....	46
Path.....	49
PathConstraints.....	68
PathFinder.....	74
PathFinding.....	79
PathPoint.....	81
PointTowardsZone.....	85
RotationTarget.....	88
Trajectory.....	91
TrajectoryState.....	101
Waypoint.....	106
WPITrajHolPose.....	110
Type Definitions.....	112
TypeDef.....	113
Enumerated Type Definitions.....	146
Enum.....	147

Introduction

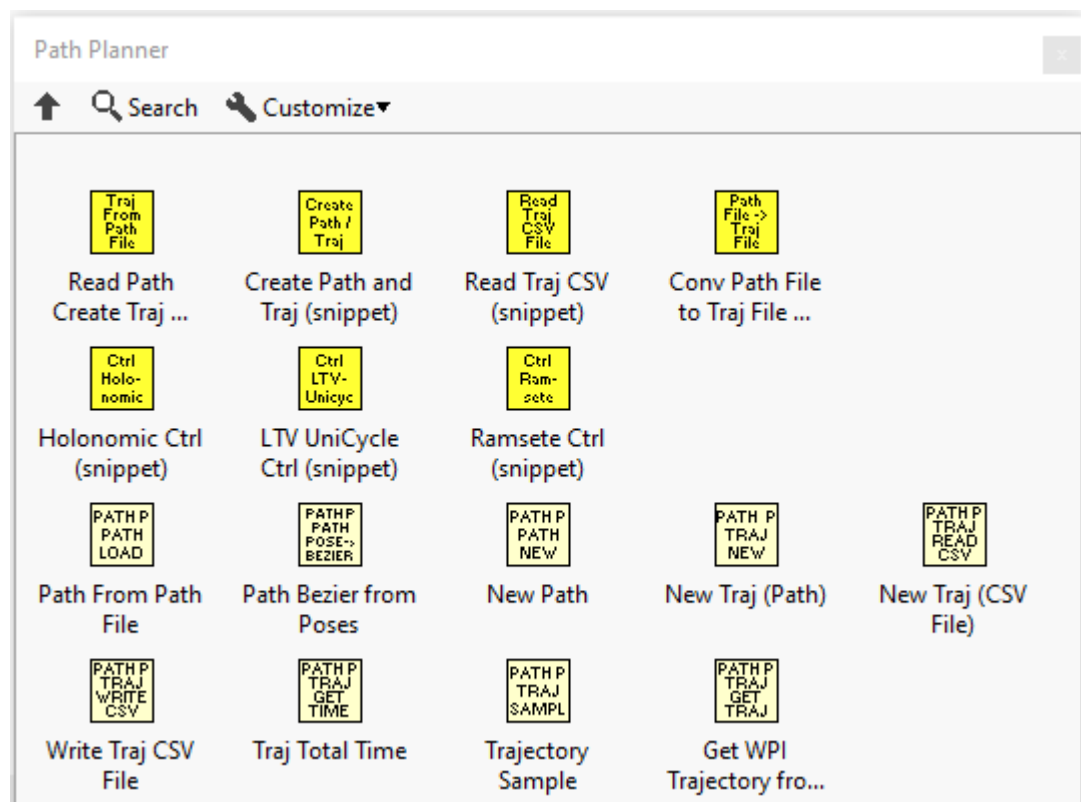
The PathPlanner LabVIEW library provides utility functions to read, create, and follow PathPlanner paths.

The library source code, package build specifications, and test package can be found here

<https://github.com/jsimpso81/PathPlannerLabVIEW>

Function Menus

A PathPlanner function palette contains the PathPlanner functions and type definitions. This palette can be accessed from the WPI Robotics Library Third Party palette.



Function Help

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

Context Help

PathPlanner24.lvlib:PathPlanner_Ctrl_HolonomicDrvExecute.vi

HOLONOMIC_CTRL_PACK_TUNING

Tolerance SI

Current Pose SI

PathPlannerState

Robot Parameters

Enabled

Reset

Period

calcChassisSpeed

At Reference

Position Error M

outHOLONOMIC_DRV_CTRL

Calculates the next output of the path following controller for holonomic drive robots. This wraps a Holonomic Drive Control function with the particulars needed to follow a PathPlanner trajectory.

Inputs:

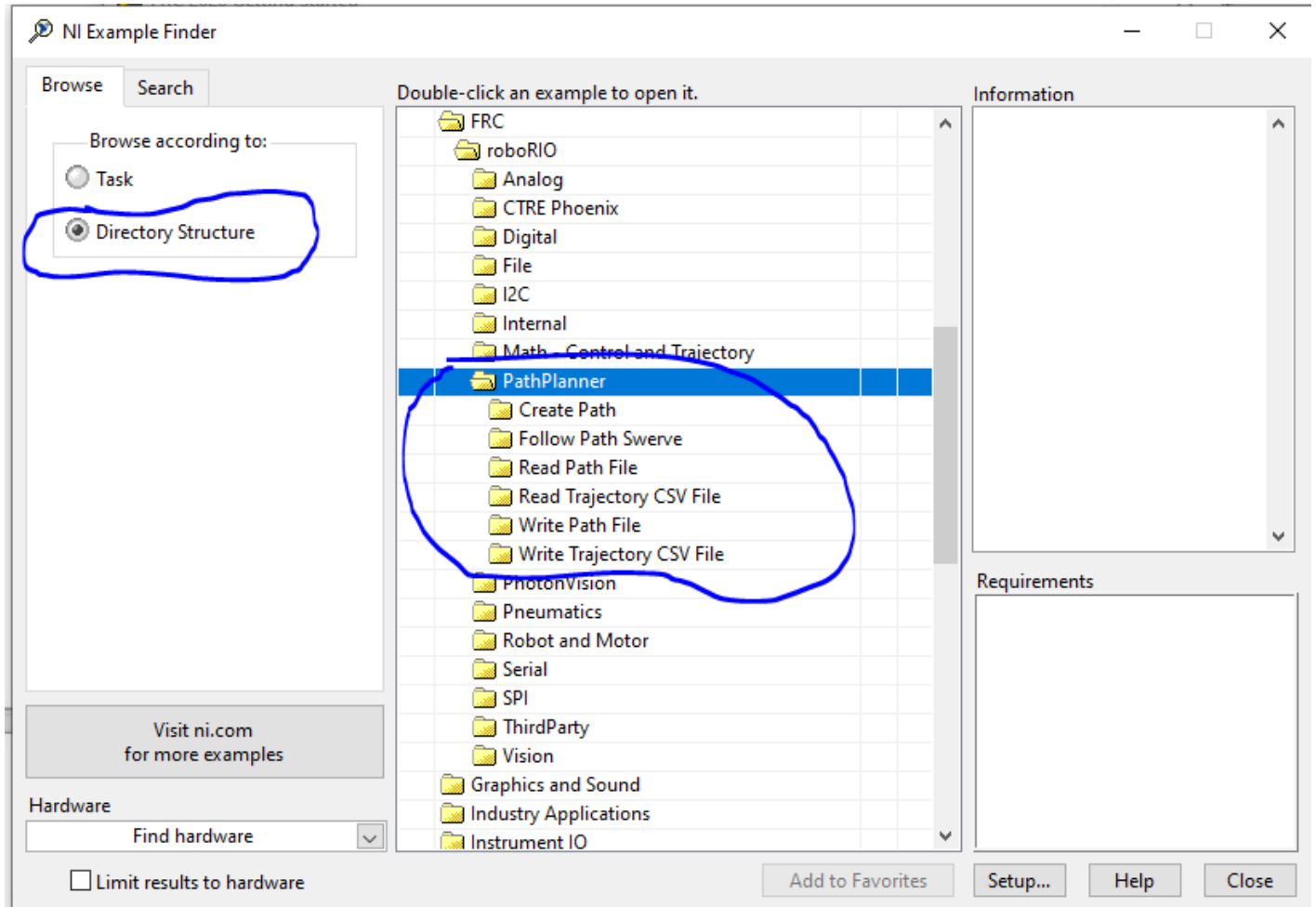
- Holonomic_Ctrl_Pack_Tuning - cluster - The tuning parameters for the holonomic drive controller. These are static. They should not be changed after the first call to this function.
- Tolerance SI - pose2d - The values used to calculate "At Reference". These are in SI units.
- CurrentPoseSi - pose2d - The current pose calculated by odometry or pose estimation. The pose can be absolute or relative depending on what type of trajectory is being followed.
- PathPlannerState - cluster - The path planner trajectory state containing the desired position and velocities of the robot.
- RobotParams - cluster - Contains details about the robot needed to control the chassis speed. These include:
 - The max speed of a drive module in meters/sec
 - The radius of the drive base in meters. For swerve drive, this is the distance from the center of the robot to the furthest module. For mecanum, this is the drive base width / 2
- Enabled - boolean - When TRUE, closed loop control is enabled. (Optional. Default: TRUE)
- Reset - boolean - When TRUE the holonomic drive controller is reset. This should be set to TRUE for the first execution of a trajectory.
- Period - double - The loop period in seconds for this controller. (Optional. Default: 0.020)

Outputs:

- CalcChassisSpeed - cluster - The desired chassis speed in SI units.
- AtReference - boolean - TRUE if the robots position is within the defined tolerance.
- PositionError - pose2d - Position error in SI units. How far the robot is away from the desired trajectory position.
- out_Holonomic_Drive_Ctrl - The current holonomic drive controller data cluster. This is mainly for diagnostic purposes.

Function Examples

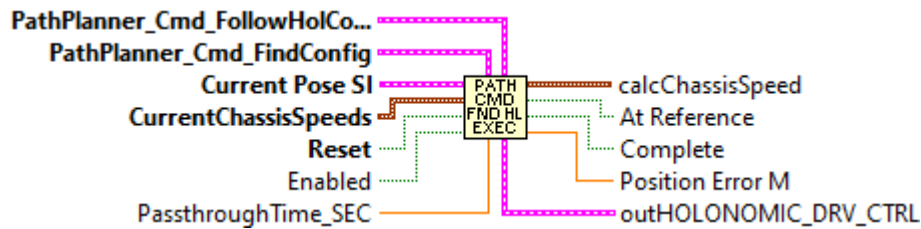
Many of the functions have examples that can be found under the LabVIEW "Find examples..." function. (Help -> Find Examples...). The function examples are easiest to find when "Directory Structure" is selected.



Function Groups

Cmd

PathPlanner_Cmd_FindFollowHolonomicPathSimpleExecute



DOCUMENTATION IS NOT COMPLETE.

Find and follow a path for holonomic (swerve and mecanum) drive robots using holonomic controller.

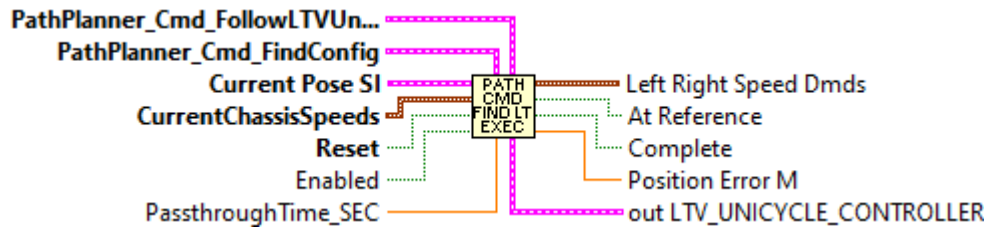
Inputs:

- CurrentPoseSi - pose2d - The current pose calculated by odometry or pose estimation. The pose can be absolute or relative depending on what type of trajectory is being followed.
- Reset - boolean - When TRUE the holonomic drive controller is reset. This should be set to TRUE for the first execution of a trajectory.
- Enabled - boolean - When TRUE, closed loop control is enabled. (Optional. Default: TRUE)
- Period - double - The loop period in seconds for this controller. (Optional. Default: 0.020)

Outputs:

- CalcChassisSpeed - cluster - The desired chassis speed in SI units.
- AtReference - boolean - TRUE if the robots position is within the defined tolerance.
- PositionError - pose2d - Position error in SI units. How far the robot is away from the desired trajectory position.
- out_Holonomic_Drive_Ctrl - THE current holonomic drive controller data cluster. This is mainly for diagnostic purposes.

PathPlanner_Cmd_FindFollowLTVUnicyclePathSimpleExecute



DOCUMENTATION IS NOT COMPLETE.

Find and follow a path for differential drive robots using LTV Unicycle controller.

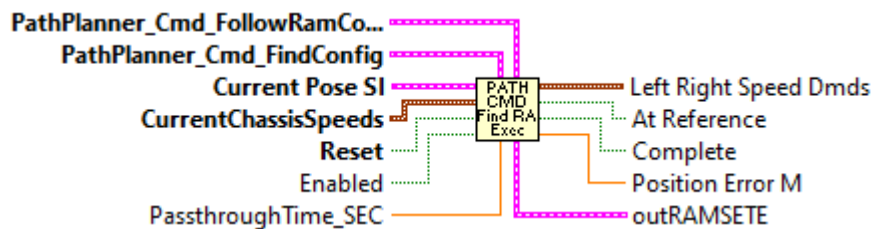
Inputs:

- CurrentPoseSi - pose2d - The current pose calculated by odometry or pose estimation. The pose can be absolute or relative depending on what type of trajectory is being followed.
- Reset - boolean - When TRUE the holonomic drive controller is reset. This should be set to TRUE for the first execution of a trajectory.
- Enabled - boolean - When TRUE, closed loop control is enabled. (Optional. Default: TRUE)

Outputs:

- AtReference - boolean - TRUE if the robots position is within the defined tolerance.
- PositionError - pose2d - Position error in SI units. How far the robot is away from the desired trajectory position.

PathPlanner_Cmd_FindFollowRamsetePathSimpleExecute



DOCUMENTATION IS NOT COMPLETE.

Find and follow a path for differential drive robots using Ramsete controller.

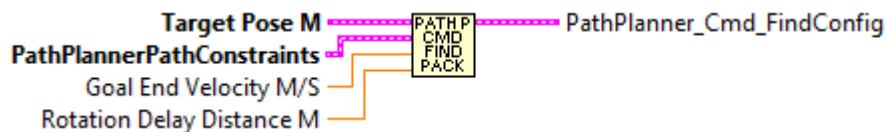
Inputs:

- CurrentPoseSi - pose2d - The current pose calculated by odometry or pose estimation. The pose can be absolute or relative depending on what type of trajectory is being followed.
- Reset - boolean - When TRUE the holonomic drive controller is reset. This should be set to TRUE for the first execution of a trajectory.
- Enabled - boolean - When TRUE, closed loop control is enabled. (Optional. Default: TRUE)

Outputs:

- AtReference - boolean - TRUE if the robots position is within the defined tolerance.
- PositionError - pose2d - Position error in SI units. How far the robot is away from the desired trajectory position.

PathPlanner_Cmd_FindPathPack



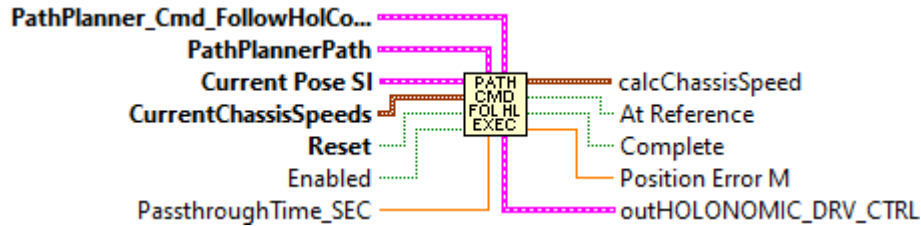
DOCUMENTATION IS NOT COMPLETE.

Pack path finding configuration data.

Inputs:

Outputs:

PathPlanner_Cmd_FollowHolonomicPathExecute



DOCUMENTATION IS NOT COMPLETE.

Calculates the next output of the path following controller for holonomic drive robots. This wraps a Holonomic Drive Control function with the particulars needed to follow a PathPlanner trajectory.

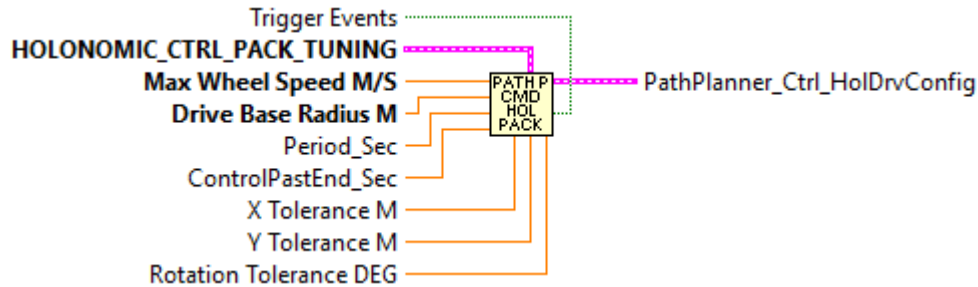
Inputs:

- PathPlanner_Holonomic_Ctrl_Pack_Tuning - cluster - The tuning parameters for the holonomic drive controller. These are static. They should not be changed after the first call to this function.
- CurrentPoseSi - pose2d - The current pose calculated by odometry or pose estimation. The pose can be absolute or relative depending on what type of trajectory is being followed.
- PathPlannerState - cluster - The path palnner trajectory state containing the desired position and velocities of the robot.
- Reset - boolean - When TRUE the holonomic drive controller is reset. This should be set to TRUE for the first execution of a trajectory.
- Enabled - boolean - When TRUE, closed loop control is enabled. (Optional. Default: TRUE)
- Period - double - The loop period in seconds for this controller. (Optional. Default: 0.020)

Outputs:

- CalcChassisSpeed - cluster - The desired chassis speed in SI units.
- AtReference - boolean - TRUE if the robots position is within the defined tolerance.
- PositionError - pose2d - Position error in SI units. How far the robot is away from the desired trajectory position.
- out_Holonomic_Drive_Ctrl - THE current holonomic drive controller data cluster. This is mainly for diagnostic purposes.

PathPlanner_Cmd_FollowHolonomicPathPack



DOCUMENTATION IS NOT COMPLETE.

Pack controller tuning configuration for Path Planner Holonomic Drive Controller.

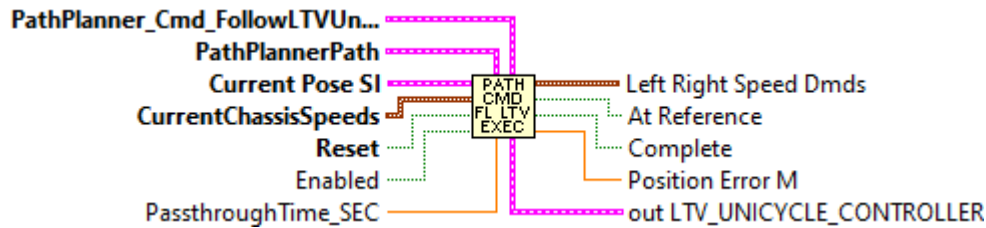
Inputs:

- X PID Tuning - cluster - Input from PACK PROF PID VI
- Y PID Tuning - cluster - Input from PACK PROF PID VI
- thetaController - cluster - A profiled PID controller to respond to error in angle.
- Max Wheel Speed M/.S - double - Maximum wheel speed (M/S)
- DriveBaseRadius - double - The radius of the drive base in meters. For swerve drive, this is the distance from the center of the robot to the furthest module. For mecanum, this is the drive base width / 2
- X Tolerance - double - X tolerance for calculating At Reference (Meters) (Optional. Default: 0.0762)
- Y Tolerance - double - Y tolerance for calculating At Reference (Meters) (Optional. Default: 0.0762)
- Rotation Tolerance - double - Rotation tolerance for calculating At Reference (Degrees) (Optional. Default: 0.0)873)

Outputs:

- PathPlannerHolCtrlConfig -- cluster - packed controller configuration.

PathPlanner_Cmd_FollowLTVUnicyclePathExecute



DOCUMENTATION IS NOT COMPLETE.

Calculates the next output of the path following controller for holonomic drive robots. This wraps a Holonomic Drive Control function with the particulars needed to follow a PathPlanner trajectory.

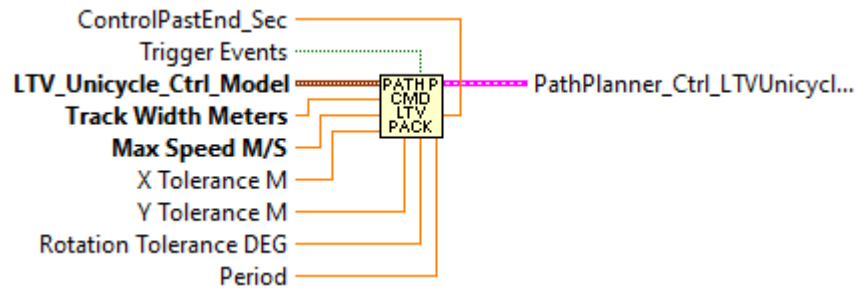
Inputs:

- PathPlanner_Holonomic_Ctrl_Pack_Tuning - cluster - The tuning parameters for the holonomic drive controller. These are static. They should not be changed after the first call to this function.
- CurrentPoseSi - pose2d - The current pose calculated by odometry or pose estimation. The pose can be absolute or relative depending on what type of trajectory is being followed.
- PathPlannerState - cluster - The path planner trajectory state containing the desired position and velocities of the robot.
- Reset - boolean - When TRUE the holonomic drive controller is reset. This should be set to TRUE for the first execution of a trajectory.
- Enabled - boolean - When TRUE, closed loop control is enabled. (Optional. Default: TRUE)
- Period - double - The loop period in seconds for this controller. (Optional. Default: 0.020)

Outputs:

- CalcChassisSpeed - cluster - The desired chassis speed in SI units.
- AtReference - boolean - TRUE if the robots position is within the defined tolerance.
- PositionError - pose2d - Position error in SI units. How far the robot is away from the desired trajectory position.
- out_Holonomic_Drive_Ctrl - The current holonomic drive controller data cluster. This is mainly for diagnostic purposes.

PathPlanner_Cmd_FollowLTVUnicyclePathPack



DOCUMENTATION IS NOT COMPLETE.

Pack controller tuning configuration for Path Planner Holonomic Drive Controller.

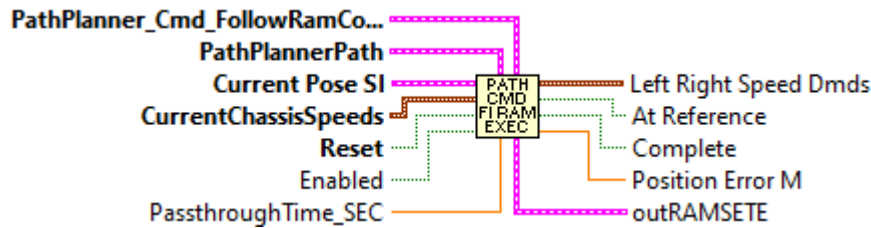
Inputs:

- X PID Tuning - cluster - Input from PACK PROF PID VI
- Y PID Tuning - cluster - Input from PACK PROF PID VI
- thetaController - cluster - A profiled PID controller to respond to error in angle.
- Max Wheel Speed M/.S - double - Maximum wheel speed (M/S)
- DriveBaseRadius - double - The radius of the drive base in meters. For swerve drive, this is the distance from the center of the robot to the furthest module. For mecanum, this is the drive base width / 2
- X Tolerance - double - X tolerance for calculating At Reference (Meters) (Optional. Default: 0.0762)
- Y Tolerance - double - Y tolerance for calculating At Reference (Meters) (Optional. Default: 0.0762)
- Rotation Tolerance - double - Rotation tolerance for calculating At Reference (Degrees) (Optional. Default: 0.0)873)

Outputs:

- PathPlannerHolCtrlConfig -- cluster - packed controller configuration.

PathPlanner_Cmd_FollowRamsetePathExecute



DOCUMENTATION IS NOT COMPLETE.

Calculates the next output of the path following controller for holonomic drive robots. This wraps a Holonomic Drive Control function with the particulars needed to follow a PathPlanner trajectory.

Inputs:

- PathPlanner_Holonomic_Ctrl_Pack_Tuning - cluster - The tuning parameters for the holonomic drive controller. These are static. They should not be changed after the first call to this function.
- CurrentPoseSi - pose2d - The current pose calculated by odometry or pose estimation. The pose can be absolute or relative depending on what type of trajectory is being followed.
- PathPlannerState - cluster - The path planner trajectory state containing the desired position and velocities of the robot.
- Reset - boolean - When TRUE the holonomic drive controller is reset. This should be set to TRUE for the first execution of a trajectory.
- Enabled - boolean - When TRUE, closed loop control is enabled. (Optional. Default: TRUE)
- Period - double - The loop period in seconds for this controller. (Optional. Default: 0.020)

Outputs:

- CalcChassisSpeed - cluster - The desired chassis speed in SI units.
- AtReference - boolean - TRUE if the robots position is within the defined tolerance.
- PositionError - pose2d - Position error in SI units. How far the robot is away from the desired trajectory position.
- out_Holonomic_Drive_Ctrl - The current holonomic drive controller data cluster. This is mainly for diagnostic purposes.

PathPlanner_Cmd_FollowRamsetePathPack



DOCUMENTATION IS NOT COMPLETE.

Pack controller tuning configuration for Path Planner Holonomic Drive Controller.

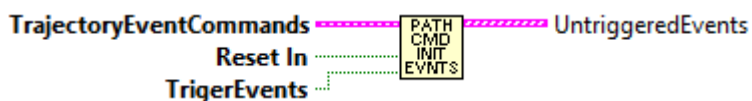
Inputs:

- X PID Tuning - cluster - Input from PACK PROF PID VI
- Y PID Tuning - cluster - Input from PACK PROF PID VI
- thetaController - cluster - A profiled PID controller to respond to error in angle.
- Max Wheel Speed M/.S - double - Maximum wheel speed (M/S)
- DriveBaseRadius - double - The radius of the drive base in meters. For swerve drive, this is the distance from the center of the robot to the furthest module. For mecanum, this is the drive base width / 2
- X Tolerance - double - X tolerance for calculating At Reference (Meters) (Optional. Default: 0.0762)
- Y Tolerance - double - Y tolerance for calculating At Reference (Meters) (Optional. Default: 0.0762)
- Rotation Tolerance - double - Rotation tolerance for calculating At Reference (Degrees) (Optional. Default: 0.0)873)

Outputs:

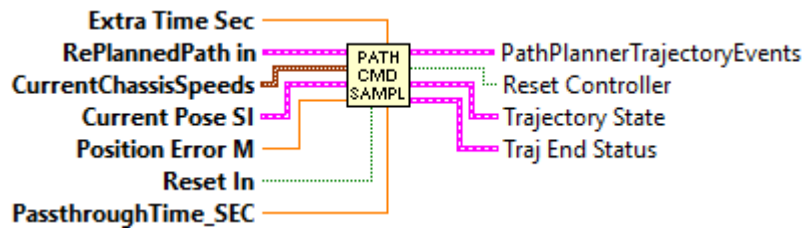
- PathPlannerHolCtrlConfig -- cluster - packed controller configuration.

PathPlanner_Cmd_Follow_Internal_InitEvents



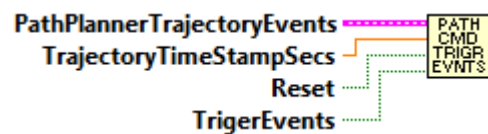
This is an internal function. It is not to be called by end users.

PathPlanner_Cmd_Follow_Internal_Sample



This is an internal function. It is not to be called by end users.

PathPlanner_Cmd_Follow_Internal_TriggerEvents



This is an internal function. It is not to be called by end users.

CommandUtil

PathPlanner_CommandUtil_Equals



Determines if two Command definitions are equal

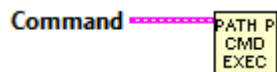
Inputs:

- Command - cluster - command definition
- Other Command - cluster - command definition

Outputs:

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

PathPlanner_CommandUtil_Execute



Issues the given command.

Currently the only type of commands that can be issued are Boolean commands. The value is forced to TRUE when the command is issued.

The following table lists the actions performed for different types of commands:

- Unknown - nothing - not supported.
- Wait - nothing - not supported
- Named - Issue boolean command with TRUE value using the "name" as the command name.
- Path - nothing - not supported
- Sequential, Parallel, Race, Deadline - Issue boolean command with TRUE value for each of the commands contained in the "commands" array. The value for "name" is used as the command name.

Inputs:

-- Command - cluster - Command definition.

Outputs:

--

PathPlanner_CommandUtil_TypeFromString



Get the command utility type enum from a string

If the string is not a valid command type, then "Unknown" is used.

Inputs:

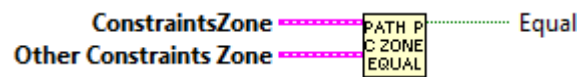
-- Type string - string - string to evaluate for command util type

Outputs:

-- type - enum - Evaluated command util type.

ConstraintsZone

PathPlanner_ConstraintsZone_Equals



Compares two Constraints Zone definitions

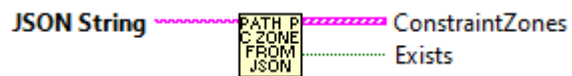
Inputs:

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

Outputs:

- Equal - boolean - TRUE if equal.

PathPlanner_ConstraintsZone_FromJSON



Create a constraints zone from json

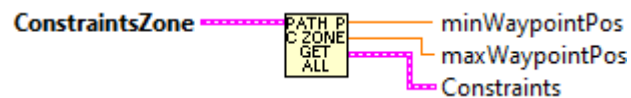
Inputs:

- 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_New



Create a new constraints zone

Inputs:

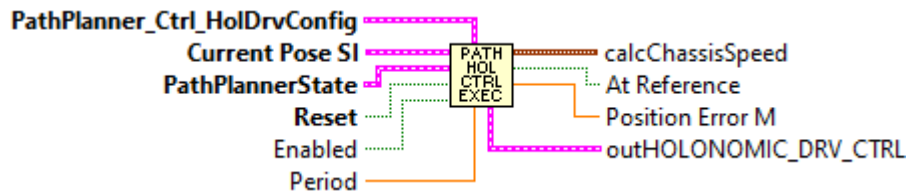
- MinPosition - double - Starting waypoint relative position of the zone
- MaxPosition - double - End waypoint relative position of the zone
- constraints - cluster - The constraints to apply within the zone

Outputs:

- ConstraintsZone - cluster - data cluster with constraint

Ctrl

PathPlanner_Ctrl_HolonomicDrvExecute



Calculates the next output of the path following controller for holonomic drive robots. This wraps a Holonomic Drive Control function with the particulars needed to follow a PathPlanner trajectory.

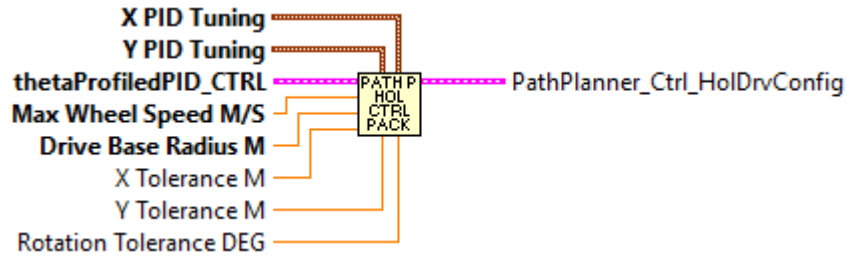
Inputs:

- PathPlanner_Holonomic_Ctrl_Pack_Tuning - cluster - The tuning parameters for the holonomic drive controller. These are static. They should not be changed after the first call to this function.
- CurrentPoseSi - pose2d - The current pose calculated by odometry or pose estimation. The pose can be absolute or relative depending on what type of trajectory is being followed.
- PathPlannerState - cluster - The path planner trajectory state containing the desired position and velocities of the robot.
- Reset - boolean - When TRUE the holonomic drive controller is reset. This should be set to TRUE for the first execution of a trajectory.
- Enabled - boolean - When TRUE, closed loop control is enabled. (Optional. Default: TRUE)
- Period - double - The loop period in seconds for this controller. (Optional. Default: 0.020)

Outputs:

- CalcChassisSpeed - cluster - The desired chassis speed in SI units.
 - AtReference - boolean - TRUE if the robots position is within the defined tolerance.
 - PositionError - pose2d - Position error in SI units. How far the robot is away from the desired trajectory position.
 - out_Holonomic_Drive_Ctrl - The current holonomic drive controller data cluster. This is mainly for diagnostic purposes.
-

PathPlanner_Ctrl_HolonomicDrvPack



Pack controller tuning configuration for Path Planner Holonomic Drive Controller.

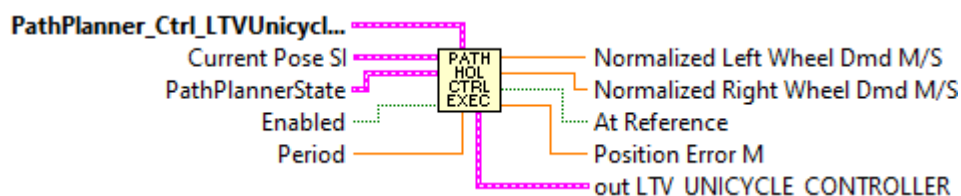
Inputs:

- X PID Tuning - cluster - Input from PACK PROF PID VI
- Y PID Tuning - cluster - Input from PACK PROF PID VI
- thetaController - cluster - A profiled PID controller to respond to error in angle.
- Max Wheel Speed M/.S - double - Maximum wheel speed (M/S)
- DriveBaseRadius - double - The radius of the drive base in meters. For swerve drive, this is the distance from the center of the robot to the furthest module. For mecanum, this is the drive base width / 2
- X Tolerance - double - X tolerance for calculating At Reference (Meters) (Optional. Default: 0.0762)
- Y Tolerance - double - Y tolerance for calculating At Reference (Meters) (Optional. Default: 0.0762)
- Rotation Tolerance - double - Rotation tolerance for calculating At Reference (Degrees) (Optional. Default: 0.0)873)

Outputs:

- PathPlannerHolCtrlConfig -- cluster - packed controller configuration.

PathPlanner_Ctrl_LTVUnicycleExecute



Calculates the next output of the path following controller for differential drive robots using a LTV (linear time varying) Unicycle controller. This wraps a LTV Control function with the particulars needed to follow a PathPlanner trajectory.

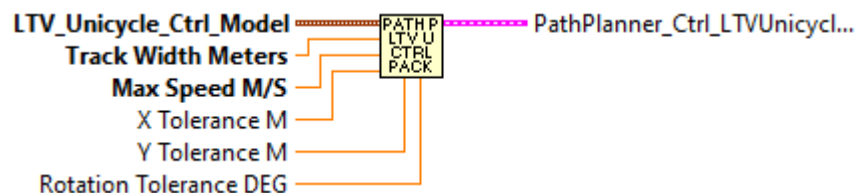
Inputs:

- PathPlanner_Ctrl_LTV_Unicycle_Ctrl_Config - cluster - Configuration from the PathPlanner_Ctrl_LTV_Unicycle_Pack_Config VI.
- CurrentPoseSi - pose2d - The current pose calculated by odometry or pose estimation. The pose can be absolute or relative depending on what type of trajectory is being followed.
- PathPlannerState - cluster - The path palnner trajectory state containing the desired position and velocities of the robot.
- Enabled - boolean - When TRUE, closed loop control is enabled. (Optional. Default: TRUE)
- Period - double - The loop period in seconds for this controller. (Optional. Default: 0.020)

Outputs:

- NormalizedLeftWheelSpeedDmd - The desired speed for the left drive wheel in SI units. The left and right speeds have been normalized so not to exceed the maximum allowed speed.
- NormalizedRightWheelSpeedDmd - The desired speed for the right drive wheel in SI units. The left and right speeds have been normalized so not to exceed the maximum allowed speed.
- AtReference - boolean - TRUE if the robots position is within the defined tolerance.
- PositionError - double - Position error in SI units. How far the robot is away from the desired trajectory position.
- LTVUnicycle_Ctrl - cluster - controller data cluster. Primarily usefull for diagnostics.

PathPlanner_Ctrl_LTVUnicyclePack



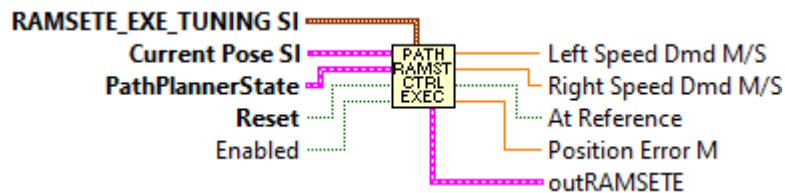
Pack controller tuning configuration for Differential Drive LTV Unicycle Controller.

Inputs:

- LTV_Unicycle_Ctrl_Model - cluster - The modeling parameters for the LTV drive controller. These are static. They should not be changed after the first call to this function.
- Track With - double - Track width (Meters)
- Max Wheel Speed M/S - double - Maximum wheel speed (M/S)
- X Tolerance -- X tolerance for calculating At Reference (Meters) (Optional. Default: 0.0762)
- Y Tolerance -- Y tolerance for calculating At Reference (Meters) (Optional. Default: 0.0762)
- Rotation Tolerance -- Rotation tolerance for calculating At Reference (Degrees) (Optional. Default: 0.0873)

Outputs:

- PathPlannerLTVUnicycleCtrlConfig -- cluster - packed controller configuration.

PathPlanner_Ctrl_RamseteExecute

Calculates the next output of the path following controller for differential drive robots using a Ramsete controller. This wraps a Ramsete function with the particulars needed to follow a PathPlanner trajectory.

Inputs:

- Ramsete_Exe_Tuning_SI - cluster - The tuning parameters for the Ramsete drive controller. These are static. They should not be changed after the first call to this function.
- CurrentPoseSi - pose2d - The current pose calculated by odometry or pose estimation. The pose can be absolute or relative depending on what type of trajectory is being followed.
- PathPlannerState - cluster - The path palnner trajectory state containing the desired position and velocities of the robot.
- Reset - boolean - When TRUE, resets the Ramsete controller. This should be set to TRUE for the first sample of following any trajectory.
- Enabled - boolean - When TRUE, closed loop control is enabled. (Optional. Default: TRUE)

Outputs:

- NormalizedLeftWheelSpeedDmd - The desired speed for the left drive wheel in SI units. The left and right speeds have been normalized so not to exceed the maximum allowed speed.
- NormalizedRightWheelSpeedDmd - The desired speed for the right drive wheel in SI units. The left and right speeds have been normalized so not to exceed the maximum allowed speed.
- AtReference - boolean - TRUE if the robots position is within the defined tolerance.
- PositionError - double - Position error in SI units. How far the robot is away from the desired trajectory position.
- OutRamsete - cluster - Current Ramsete data cluster. Primarily for diagnostic purposes.

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.

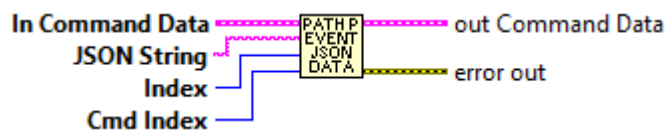
Notes:

Currently the only type of commands that can be issued are Boolean commands. The value is forced to TRUE when the command is issued.

The following table lists the actions performed for different types of commands:

- Unknown - nothing - not supported.
- Wait - nothing - not supported
- Named - Issue boolean command with TRUE value using the "name" as the command name.
- Path - nothing - not supported
- Sequential, Parallel, Race, Deadline - Issue boolean command with TRUE value for each of the commands contained in the "commands" array. The value for "name" is used as the command name.

PathPlanner_EventMarker_FromJSON_Data



Internal function to parse JSON data for Event Markers. This is data that is different for each different type of Event Marker command.

Notes:

Currently the only type of commands that can be issued are Boolean commands. The value is forced to TRUE when the command is issued.

The following table lists the actions performed for different types of commands:

- Unknown - nothing - not supported.
- Wait - nothing - not supported
- Named - Issue boolean command with TRUE value using the "name" as the command name.
- Path - nothing - not supported
- Sequential, Parallel, Race, Deadline - Issue boolean command with TRUE value for each of the commands contained in the "commands" array. The value for "name" is used as the command name.

PathPlanner_EventMarker_GetCommand



Get the command associated with this marker

Inputs:

- EventMarker - cluster - Data cluster

Outputs:

- Command - cluster - command for this marker

PathPlanner_EventMarker_GetEndPosition



Get the waypoint relative position of this marker

Inputs:

- EventMarker - cluster - Data cluster

Outputs:

- EndPosition - double - The end waypoint relative position of the event's zone. A value of -1.0 indicates that this event is not zoned.

PathPlanner_EventMarker_GetTriggerName



Get the trigger name of this marker

Inputs:

- EventMarker - cluster - Data cluster

Outputs:

- triggerName - string - The name of the trigger this event marker will control

PathPlanner_EventMarker_GetWaypointRelativePos



Get the waypoint relative position of this marker

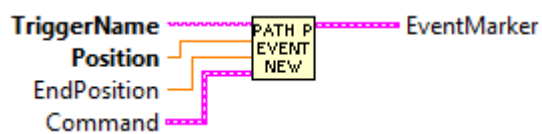
Inputs:

- EventMarker - cluster - Data cluster

Outputs:

- WaypointRelativePose - double - The waypoint relative position of the marker

PathPlanner_EventMarker_New



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

Inputs:

- triggerName - string - The name of the trigger this event marker will control
- position - double - The waypoint relative position of the marker

- endPosition - double - The end waypoint relative position of the event's zone. A value of -1.0 indicates that this event is not zoned.
- Command - cluster -- Contains the command(s) to execute. Can be null to not run a command.

Outputs:

- EventMarker - cluster - Data cluster

Notes:

Currently the only type of commands that can be issued are Boolean commands. The value is forced to TRUE when the command is issued.

The following table lists the actions performed for different types of commands:

- Unknown - nothing - not supported.
- Wait - nothing - not supported
- Named - Issue boolean command with TRUE value using the "name" as the command name.
- Path - nothing - not supported
- Sequential, Parallel, Race, Deadline - Issue boolean command with TRUE value for each of the commands contained in the "commands" array. The value for "name" is used as the command name.

FieldUtil

PathPlanner_FieldUtil_FieldSizeX



The X size or length of the current field in meters

Input:

--

Outputs:

-- FieldSizeX -- X length of field (meters)

PathPlanner_FieldUtil_FieldSizeY



The Y size or length of the current field in meters

Input:

--

Outputs:

-- FieldSizeY -- Y length of field (meters)

PathPlanner_FieldUtil_FlipFieldPosition



Flip a field position

Input:

-- FieldPosition

Outputs:

-- FlippedFieldPosition

PathPlanner_FieldUtil_FlipFieldRotation



Flip a field rotation

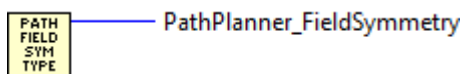
Input:

-- FieldRotation

Outputs:

-- FlippedFieldRotation

PathPlanner_FieldUtil_SymmetryType



The type of symmetry for the current field

Inputs:

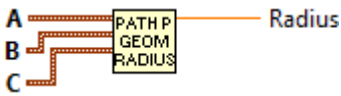
--

Outputs:

-- FieldSymmetryType - The type of symmetry for the current field

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:

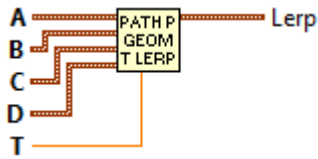
- Radius - double - Curve radius

PathPlanner_GeomUtil_CoerceHeadingDegrees



PathPlanner_GeomUtil_CoerceHeadingRadians



PathPlanner_GeomUtil_CubicLerp

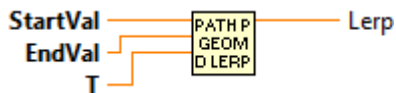
Cubic interpolation between Translation2ds

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

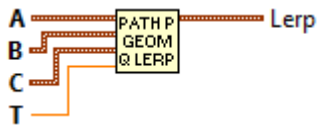
Inputs:

- 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:

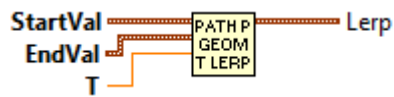
- startVal - rotation2d - Start value
- endVal - rotation2d - End value

- 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

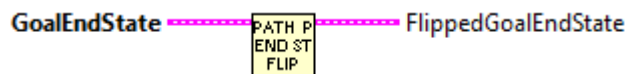
Inputs:

- 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_Flip



Flip the goal end state for the other side of the field, maintaining a blue alliance origin

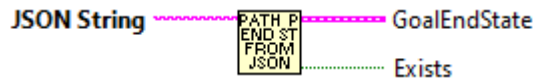
Inputs:

- GoalEndState - cluster - definition data structure

Outputs:

- FlippedGoalEndState - cluster - The flipped end state

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:

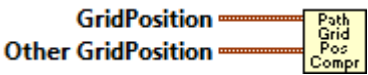
- velocity - double - The goal end velocity (M/S)
- rotation - rotation2d - The goal rotation

Outputs:

- GoalEndState - cluster - data structure

GridPosition

PathPlanner_GridPosition_CompareTo



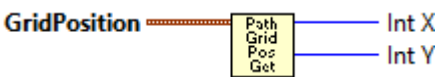
PathPlanner_GridPosition_Equals



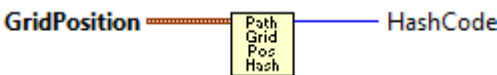
PathPlanner_GridPosition_FromHash



PathPlanner_GridPosition_Get



PathPlanner_GridPosition_HashCode



PathPlanner_GridPosition_New

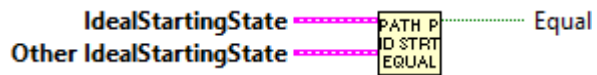


PathPlanner_GridPosition_Print



IdealStartingState

PathPlanner_IdealStartingState_Equals



Determines if two Ideal Starting State definitions are equal

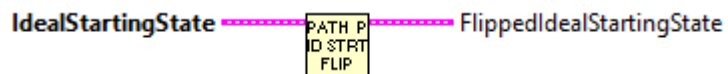
Inputs:

- IdealStartingState - cluster - ideal start state definition
- Other IdealStartingState - cluster - ideal start state definition

Outputs:

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

PathPlanner_IdealStartingState_Flip



Flip the ideal starting state for the other side of the field, maintaining a blue alliance origin

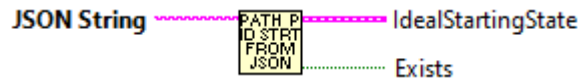
Inputs:

- IdealStartingState - cluster - definition data structure

Outputs:

- Flipped IdealStartingState - cluster - The flipped start state
-

PathPlanner_IdealStartingState_FromJSON



Create an ideal starting state from json

Inputs:

-- JSON String - string - string to parse for IdealStartingState

Outputs:

-- IdealStartingState - cluster - The ideal starting state defined by the given json. If not found, default is returned.

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

PathPlanner_IdealStartingState_GetAll



Get the ideal starting state velocity and end rotation

Inputs:

-- IdealStartingState - cluster - definition data structure

Outputs:

-- Ideal starting velocity (M/S)

-- Ideal starting rotation

PathPlanner_IdealStartingState_New



Describes the ideal starting state of the robot when starting a path

Inputs:

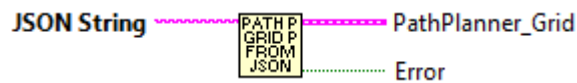
- velocityMPS - double - The ideal starting velocity (M/S)
- rotation - Rotation2d - The ideal starting rotation

Outputs:

- IdealStartingState - cluster - data structure

ObstacleGrid

PathPlanner_ObstacleGrid_FromJSON



Parse an obstacle grid from a JSON formatted string.

Inputs:

- JSON String - string- The string containing the obstacle grid definition.

Outputs:

- PathPlanner_ObstacleGrid - cluster - Obstacle grid read from JSON file.
- Error - boolean - TRUE if an error occurred.

PathPlanner_ObstacleGrid_ProcessNavFile



Load an obstacle grid from a JSON formatted file.

Inputs:

- filename - string - The name of the obstacle grid file. See notes below on file naming.

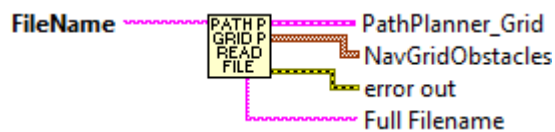
Outputs:

- PathPlanner_ObstacleGrid - cluster - Obstacle grid read from JSON file.
- Error out - cluster - Error cluster
- Full Filename - string - fully qualified file name.

Notes on file naming:

- The file name must include the extension. ".json" 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.

PathPlanner_ObstacleGrid_ReadNavFile



Load an obstacle grid from a JSON formatted file.

Inputs:

- filename - string - The name of the obstacle grid file. See notes below on file naming.

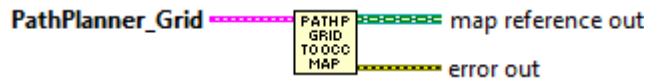
Outputs:

- PathPlanner_ObstacleGrid - cluster - Obstacle grid read from JSON file.
- Error out - cluster - Error cluster
- Full Filename - string - fully qualified file name.

Notes on file naming:

- The file name must include the extension. ".json" 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.

PathPlanner_ObstacleGrid_ToOccupancyGrid



Converts a PathPlanner Obstacle Grid to a LabVIEW occupancy map reference.

Inputs:

- PathPlanner_ObstacleGrid - cluster -- data to convert

Outputs:

- Map Reference Out -- reference -- Reference to Occupancy map for use by LabVIEW AD* pathfinding.
- Error out - cluster - Error cluster

Path

PathPlanner_Path_BezierFromWaypointsJSON



**** OBSOLETE FUNCTION ****

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_BezierToWaypoints



Convert bezier points to waypoints allowing them to be written to a JSON string or used to create Paths.

Inputs:

- Bezier - array of translation2s - List of bezier points

Outputs:

- waypoints - array of cluster - waypoints compatible with pathplanner path JSON strings.

PathPlanner_Path_ConstTargetIncrement



CONSTANT - TargetIncrement

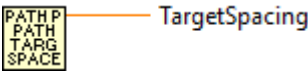
Inputs:

-

Outputs:

- TargetIncrement - double -

PathPlanner_Path_ConstTargetSpacing



Constant - TargetSpacing

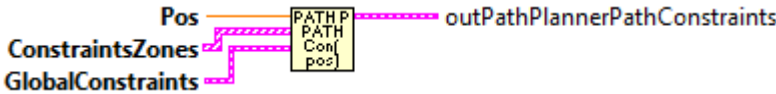
Inputs:

-

Outputs:

- TargetSpacing - double - constant

PathPlanner_Path_ConstraintsForWaypointPos



Internal function - DO NOT CALL

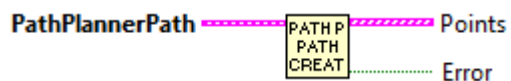
Inputs:

- Pos
- ConstraintZones
- GlobalConstraints

Outputs:

- PathConstraints

PathPlanner_Path_CreatePath



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

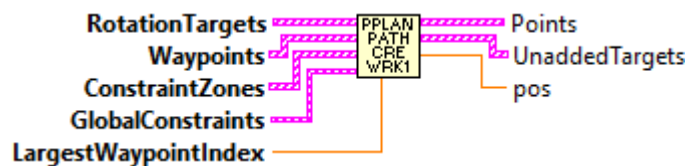
Inputs:

- PathPlannerPath - cluster - Path definition data structure

Outputs:

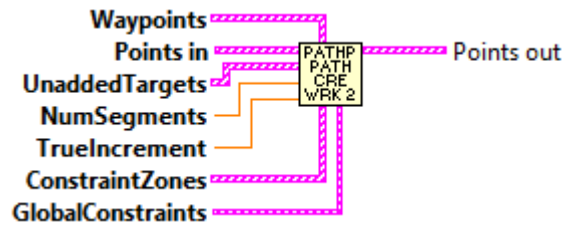
- PathPoints - PathPoint array - Array of points along the path
- Error - boolean - TRUE if an error occurred.

PathPlanner_Path_CreatePath_Worker1



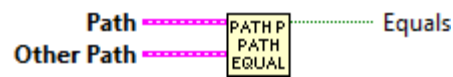
Internal function for CreatePath -- DO NOT CALL.

PathPlanner_Path_CreatePath_Worker2



Internal function for CreatePath -- DO NOT CALL.

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:

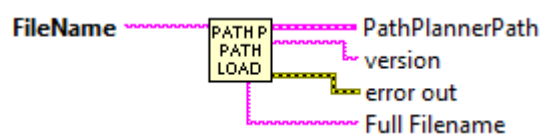
- Path - cluster - PathPlannerPath created from the given JSON string
- version - string - version string from JSON
- Error out - cluster - Error cluster

Notes:

Currently the only type of commands that can be issued are Boolean commands. The value is forced to TRUE when the command is issued.

The following table lists the actions performed for different types of commands:

- Unknown - nothing - not supported.
- Wait - nothing - not supported
- Named - Issue boolean command with TRUE value using the "name" as the command name.
- Path - nothing - not supported
- Sequential, Parallel, Race, Deadline - Issue boolean command with TRUE value for each of the commands contained in the "commands" array. The value for "name" is used as the command name.

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
- version - string - Version string from file contents.
- 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: %HOMEDRIVE%%HOMEPATH%\Documents\LabView Data".
- Filenames on the RoboRIO, which runs Linux, are case sensitive.

Notes:

Currently the only type of commands that can be issued are Boolean commands. The value is forced to TRUE when the command is issued.

The following table lists the actions performed for different types of commands:

- Unknown - nothing - not supported.
- Wait - nothing - not supported
- Named - Issue boolean command with TRUE value using the "name" as the command name.
- Path - nothing - not supported
- Sequential, Parallel, Race, Deadline - Issue boolean command with TRUE value for each of the commands contained in the "commands" array. The value for "name" is used as the command name.

PathPlanner_Path_FromPathPonts



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

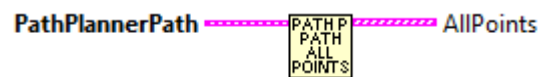
Inputs:

- 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_GetConstraintZones



Get the constraint zones for the path

Inputs:

- Path - cluster - path definition data structure

Outputs:

- ConstraintZones - array of cluster - The constraints zones for this path

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_GetIdealStartingState



Get the ideal starting state for this path.

Inputs:

- Path - cluster - path definition data structure

Outputs:

- IdealStartingState - cluster - The ideal starting state.

PathPlanner_Path_GetInitialHeading



Get the initial heading, or direction of travel, at the start of the path.

Inputs:

- Path - cluster - path definition data structure

Outputs:

- InitialHeading - rotation2d - Initial direction of travel.

PathPlanner_Path_GetPathPoses



Get a list of poses representing every point in this path. This can be used to display a path on a field 2d widget, for example.

Inputs:

- Path - cluster - path definition data structure

Outputs:

- PathPoses - array of Pose2d - List of poses for each point in this path

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_GetPointTowardsZones



Get the PointTowardsZones for this path

Inputs:

- Path - cluster - path definition data structure

Outputs:

- PointTowardsZones - array cluster - List of PointTowardsZones along the path.

PathPlanner_Path_GetPoints



Get all points along this path

Inputs:

- Path - cluster - path definition data structure

Outputs:

- Points - array of PathPoint - All path points
-
-

PathPlanner_Path_GetRotationTargets



Get the RotationTargets for this path

Inputs:

- Path - cluster - path definition data structure

Outputs:

- RotationTargets - array of cluster - The rotation targets for this path
-
-

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_GetStartingHolonomicPose



Get the holonomic pose for the start point of this path. If the path does not have an ideal starting state, this will return an empty optional.

Inputs:

- Path - cluster - path definition data structure

Outputs:

- StartingHolonomicPose - pose2d - The ideal starting pose if an ideal starting state is present, empty optional otherwise

PathPlanner_Path_GetWaypoints



Get the Waypoints for this path

Inputs:

- Path - cluster - path definition data structure

Outputs:

- Waypoints - array of cluster - The waypoints for this path.

PathPlanner_Path_HotReload



Hot reload the path. This is used internally.

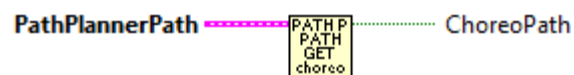
Inputs:

- 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_IsChoreoPath



Is this a Choreo path

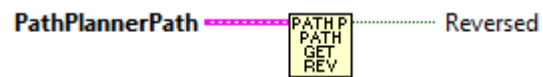
Inputs:

- Path - cluster - path definition data structure

Outputs:

- ChoreoPath - boolean - True if this is a ChoreoPath
-
-

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_MirrorTranslation



This is an internal function.

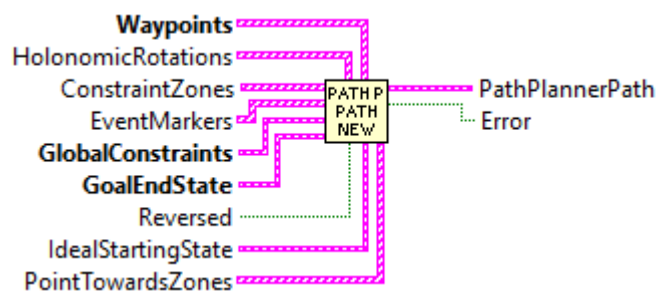
Inputs:

- InTranslation - cluster - The original translation

Outputs:

- MirrorTranslation - cluster - The mirrored translation

PathPlanner_Path_New



Create a new path planner path

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

Inputs:

- Waypoints - List of waypoints representing the path. For on-the-fly paths, you likely want to use waypointsFromPoses to create these.
- holonomicRotations - List of rotation targets along the path. (Optional. Default: empty)
- pointTowardsZones - List of point towards zones 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)
- idealStartingState - The ideal starting state of the path. Can be null if unknown (Optional. Default: 0)

Outputs:

- Path - cluster - path definition data structure

PathPlanner_Path_New_Empty



Creates an empty path data cluster. This is an INTERNAL function. Users should not call this.

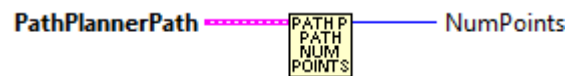
Inputs:

- Global Constraints - cluster
- GoalEndState - cluster

Outputs:

- PathPlannerPath - cluster - empty path planner path data cluster.

PathPlanner_Path_NumPoints



Get the number of points in this path

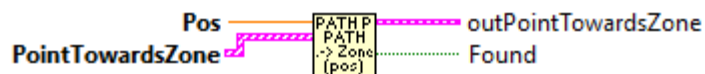
Inputs:

- Path - cluster - path definition data structure

Outputs:

- NumPoints - integer - Number of points in the path
-
-

PathPlanner_Path_PointZoneForWaypointPos



This is an internal function. It should not be called by the user.

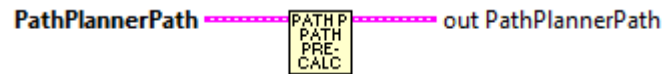
Inputs:

- pos -
- PointTowardsZones -

Outputs:

- outPointTowardsZone
 - found - boolean - TRUE if found.
-
-

PathPlanner_Path_PreCalcValues



This is an internal routine.

Inputs:

- Path - cluster - path definition data structure

Outputs:

- Path - cluster - path definition data structure

PathPlanner_Path_SamplePath



This is an internal function. It should not be called by the user.

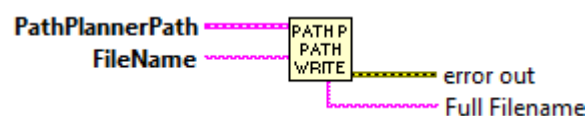
Inputs:

- WaypointRelativePose
- Waypoints

Outputs:

- Sample - translation2d

PathPlanner_Path_ToPathFile



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

Inputs:

- Path - cluster - PathPlannerPath created from the given file name
- filename - string - The name of the path to write. Existing files will be overwritten.

Outputs:

- 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: %HOMEDRIVE%%HOMEPATH%\Documents\LabView Data".
- Filenames on the RoboRIO, which runs Linux, are case sensitive.

PathPlanner_Path_WaypointsFromPoses

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

Inputs:

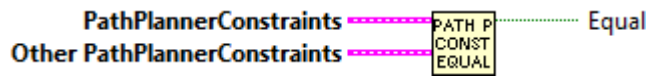
- poses - pose2d array - List of poses. Each pose represents one waypoint.

Outputs:

- Waypoints - cluster array - Bezier curve waypoints
- Error - boolean - TRUE if an error occurred. (Too few poses)

PathConstraints

PathPlanner_PathConstraints_Equals



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

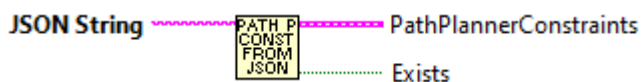
Inputs:

- 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

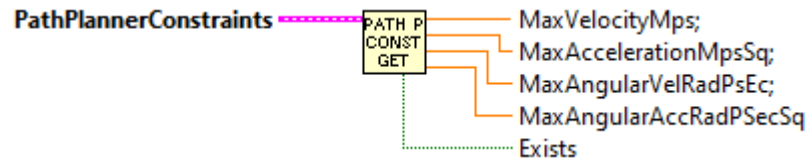
Inputs:

- 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/S²)
- maxAngularVelocityRps - double - Max angular velocity (Rad/S)
- maxAngularAccelerationRpsSq - double - Max angular acceleration (Rad/S²)
- exists - boolean - TRUE if this data cluster is not null.

PathPlanner_PathConstraints_GetMaxAcceleration



Get max acceleration of Path Constraints cluster

Inputs:

- PathConstraint - cluster - The path constraints to query

Outputs:

- maxAccelerationMpsSq - double - Max linear acceleration (M/S²)

PathPlanner_PathConstraints_GetMaxAngularAcceleration



Get max angular acceleration of Path Constraints cluster

Inputs:

- PathConstraint - cluster - The path constraints to query

Outputs:

- maxAngularAccelerationRpsSq - double - Max angular acceleration (Rad/S²)

PathPlanner_PathConstraints_GetMaxAngularVelocity



Get max angular velocity of Path Constraints cluster

Inputs:

- PathConstraint - cluster - The path constraints to query

Outputs:

- maxAngularVelocityRps - double - Max angular velocity (Rad/S)

PathPlanner_PathConstraints_GetMaxVelocity



Get Max Velocity of Path Constraints cluster


Inputs:

- PathConstraint - cluster - The path constraints to query

Outputs:

- maxVelocityMps - double - Max linear velocity (M/S)

PathPlanner_PathConstraints_GetNominalVoltage

PathPlannerConstraints  NominalVoltage_Volts;

Get nominal voltage elements of Path Constraints cluster

Inputs:

- PathConstraint - cluster - The path constraints to query

Outputs:

- NominalVoltage - double - Nominal voltage (volts)

PathPlanner_PathConstraints_GetUnlimited

PathPlannerConstraints  Unlimited

Get unlimited element of Path Constraints cluster

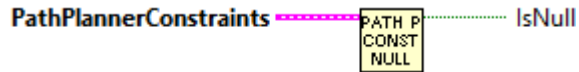
Inputs:

- PathConstraint - cluster - The path constraints to query

Outputs:

- unlimited - boolean - constraint is unlimited.

PathPlanner_PathConstraints_IsNull



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

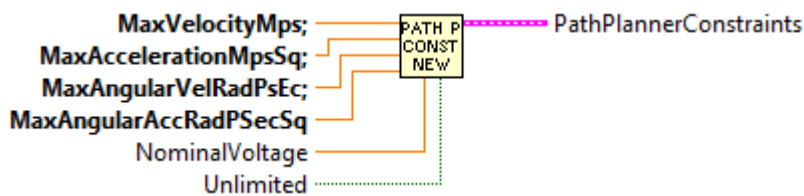
Inputs:

- 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/S²)
- maxAngularVelocityRps - double - Max angular velocity (Rad/S)
- maxAngularAccelerationRpsSq - double - Max angular acceleration (Rad/S²)
- NominalVoltageVolts - double - The nominal battery voltage (Volts), default: 12.0
- Unlimited - boolean - Should the constraints be unlimited, default: false

Outputs:

- PathConstraint - cluster - path constraint data

PathPlanner_PathConstraints_UnlimitedConstraints



Get unlimited PathConstraints

Inputs:

- NominalVoltageVolts - double - The nominal battery voltage (Volts), default: 12.0

Outputs:

- PathConstraint - cluster - unlimited path constraint data

PathFinder

PathPlanner_PathFinder_ApplyAnchorSmooth



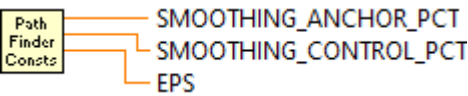
Internal function. Not to be used by end users.

PathPlanner_PathFinder_ApplyControlSmooth



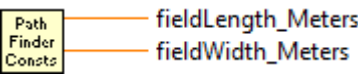
Internal function. Not to be used by end users.

PathPlanner_PathFinder_Constants



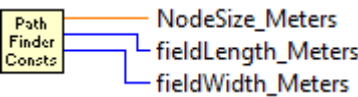
Internal function. Not to be used by end users.

PathPlanner_PathFinder_DefaultFieldSize



Internal function. Not to be used by end users.

PathPlanner_PathFinder_DefaultNodeInfo



Internal function. Not to be used by end users.

PathPlanner_PathFinder_DoMinorALT



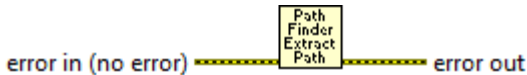
Internal function. Not to be used by end users.

PathPlanner_PathFinder_DoResetALT



Internal function. Not to be used by end users.

PathPlanner_PathFinder_ExtractPath



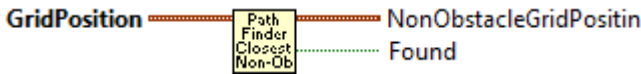
Internal function. Not to be used by end users.

PathPlanner_PathFinder_ExtractPathSub1ALT



Internal function. Not to be used by end users.

PathPlanner_PathFinder_FindClosestNonObstacle



Internal function. Not to be used by end users.

PathPlanner_PathFinder_GetAllNeighbors



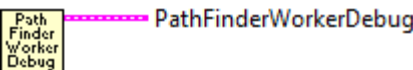
Internal function. Not to be used by end users.

PathPlanner_PathFinder_GetGridPos



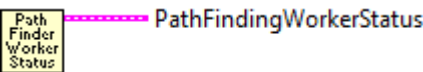
Internal function. Not to be used by end users.

PathPlanner_PathFinder_GetWorkerDebug



Get debug information for Path Finder worker

PathPlanner_PathFinder_GetWorkerStatus



Get status of Path Finder worker

PathPlanner_PathFinder_Globals



Globals for Path Finder

PathPlanner_PathFinder_GridPosToTranslation2d



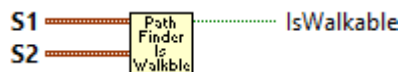
Internal function. Not to be used by end users.

PathPlanner_PathFinder_InitializeWorkerThread



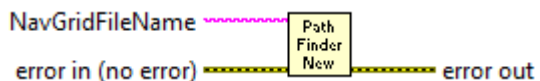
Internal function. Not to be used by end users.

PathPlanner_PathFinder_IsWalkable



Internal function. Not to be used by end users.

PathPlanner_PathFinder_New



Create and initialize all data structures and back end VI (threads) needed by the Path Finder / Path Finding functions. Only one instance of this can be used at a time.

PathPlanner_PathFinder_SendCommand



Internal function. Not to be used by end users.

PathPlanner_PathFinder_SetNewPathAvail



Internal function. Not to be used by end users.

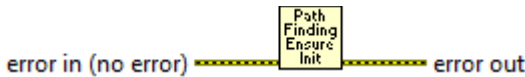
PathPlanner_PathFinder_WorkerThreadALT



Background worker VI (thread) for the Path Finder / Path Finding functions. This will be started by the New function. This should NOT be called by an end user.

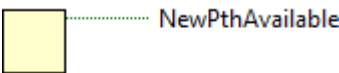
PathFinding

PathPlanner_PathFinding_EnsureInitialized

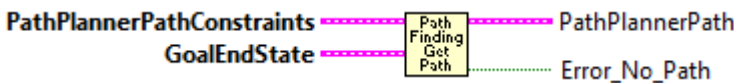


DOCUMENTATION NOT COMPLETE YET.

PathPlanner_PathFinding_GetCurrentPath



PathPlanner_PathFinding_GetNewPath



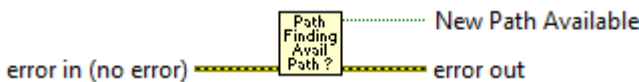
DOCUMENTATION NOT COMPLETE YET.

PathPlanner_PathFinding_Initialize



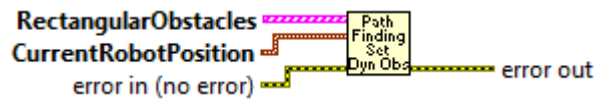
DOCUMENTATION NOT COMPLETE YET.

PathPlanner_PathFinding_IsNewPathAvailable



DOCUMENTATION NOT COMPLETE YET.

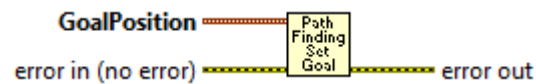
PathPlanner_PathFinding_SetDynamicObstacles



This has not been implemented yet. It can be called but it doesn't affect the created path.

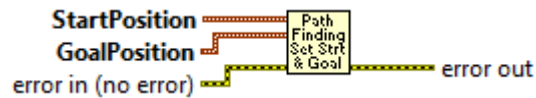
DOCUMENTATION NOT COMPLETE YET.

PathPlanner_PathFinding_SetGoalPosition



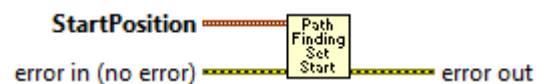
DOCUMENTATION NOT COMPLETE YET.

PathPlanner_PathFinding_SetStartAndGoalPosition



DOCUMENTATION NOT COMPLETE YET.

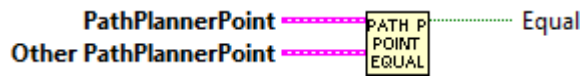
PathPlanner_PathFinding_SetStartingPosition



DOCUMENTATION NOT COMPLETE YET.

PathPoint

PathPlanner_PathPoint_Equals



Determines if two Path Point definitions are equal

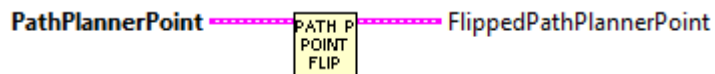
Inputs:

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

Outputs:

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

PathPlanner_PathPoint_Flip



Flip this path point to the other side of the field, maintaining a blue alliance origin

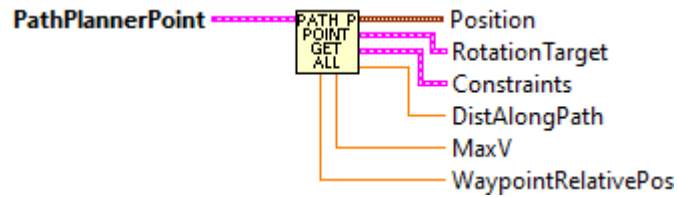
Inputs:

- PathPoint - cluster - point definition

Outputs:

- FlippedPathPoint - cluster - flipped point definition
-

PathPlanner_PathPoint_GetAll



Gets elements of PathPoint

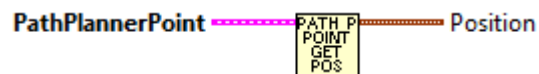
Inputs:

- PathPoint - cluster - point definition

Outputs:

- Position - translation2d - The position of this point
- DistAlongPath - double - The distance of this point along the path, in meters
- MaxV - double - The max velocity at this point
- WaypointRelativePos - double - The waypoint relative position of this point. Used to determine proper event marker timing
- RotationTarget - RotationTarget - The target rotation at this point
- Constraints - PathConstraints - The constraints applied to this point

PathPlanner_PathPoint_GetPosition



Gets position of PathPoint

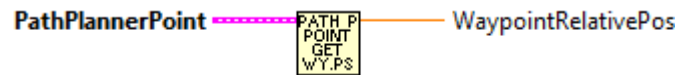
Inputs:

- PathPoint - cluster - point definition

Outputs:

- Position - translation2d - The position of this point

PathPlanner_PathPoint_GetWaypointRelPos



Gets waypoint relative position of PathPoint

Inputs:

- PathPoint - cluster - point definition

Outputs:

- WaypointRelativePos - double - The waypoint relative position of this point. Used to determine proper event marker timing

PathPlanner_PathPoint_New



Create a path point

Inputs:

- position - double - Position of the point
- holonomicRotation - rotation2d - Rotation target at this point (Default: none)
- constraints - cluster - The constraints at this point (Default: none)

Outputs:

- PathPlannerPoint - cluster - point definition

PathPlanner_PathPoint_SetWaypointRelPos

Sets waypoint relative position of this path point.

Inputs:

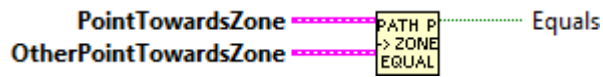
- PathPoint - cluster - point definition
- WaypointRelativePos - double - The waypoint relative position of this point. Used to determine proper event marker timing

Outputs:

- outPathPlannerPoint - cluster - updated point

PointTowardsZone

PathPlanner_PointTowardsZone_Equals



Determine if two PointTowardZone clusters are the same.

Inputs:

- PointTowardsZone - cluster - cluster to compare.
- OtherPointTowardsZone - cluster - other cluster to compare.

Outputs:

- Equal - boolean - TRUE if clusters are equal.
-

PathPlanner_PointTowardsZone_Flip



Flip this point towards zone to the other side of the field, maintaining a blue alliance origin

Inputs:

- PointTowardsZone - cluster -- The cluster to flip.

Outputs:

- FlippedPointTowardsZone - cluster -- The flipped cluster.
-

PathPlanner_PointTowardsZone_FromJSON



Create new array of clusters containing zone on a path that will force the robot to point towards a position on the field

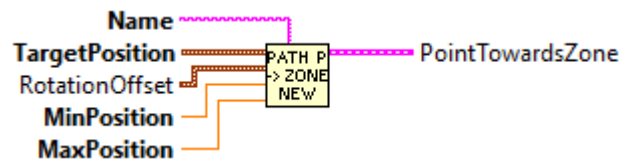
Inputs:

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

Outputs:

- PointTowardsZones- array of cluster - defined point towards zone data structures
- Exists - boolean - True if a point towards zone was found and parsed.

PathPlanner_PointTowardsZone_New



Create a new cluster containing zone on a path that will force the robot to point towards a position on the field

Inputs:

- name - string - The name of this zone. Used for point towards zone triggers
- targetPosition - translation2d - The target field position in meters
- rotationOffset - rotation2d - A rotation offset to add on top of the angle to the target position. For example, if you want the robot to point away from the target position, use a rotation offset of 180 degrees. (Default: 0.0)
- minPosition - double - Starting waypoint relative position of the zone
- maxPosition - double - End waypoint relative position of the zone

Outputs:

- PointTowardsZone - cluster - defined point towards zone data structure

RotationTarget

PathPlanner_RotationTarget_Equals



Determine if two rotation targets are equal

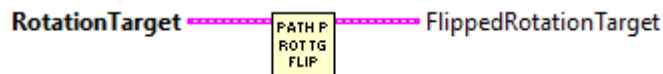
Inputs:

- 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_Flip



Flip a rotation target for the other side of the field, maintaining a blue alliance origin

Inputs:

- RotationTarget - cluster - defined rotation target data structure

Outputs:

- FlippedRotationTarget - cluster - The flipped rotation target

PathPlanner_RotationTarget_FromJSON



Create a rotation target from json

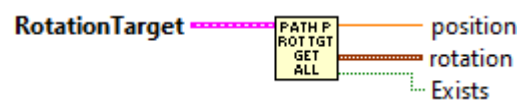
Inputs:

- 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.

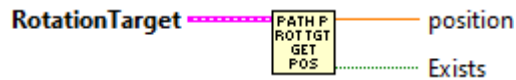
Inputs:

- RotationTarget - cluster - defined rotation target data structure

Outputs:

- position - double - Waypoint relative position of this target
- rotation - rotation2d - Target rotation
- exists - boolean - TRUE if not null - Exists - boolean - TRUE if rotation target is not null

PathPlanner_RotationTarget_GetPosition



Get position of a rotation target.

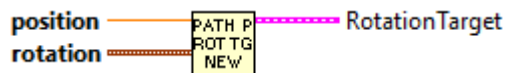
Inputs:

- RotationTarget - cluster - defined rotation target data structure

Outputs:

- position - double - Waypoint relative position of this target
- exists - boolean - TRUE if not null - Exists - boolean - TRUE if rotation target is not null

PathPlanner_RotationTarget_New



Create a new rotation target

Inputs:

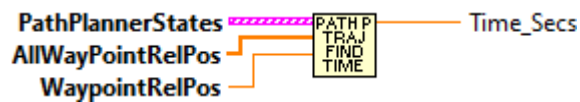
- position - double - Waypoint relative position of this target
- rotation - rotation2d - Target rotation

Outputs:

- RotationTarget - cluster - defined rotation target data structure

Trajectory

PathPlanner_Trajectory_FindTimeForWaypointRelPos



Find the time (seconds) for a particular waypoint relative position

Inputs:

- PathPlannerStates -- PathPlanner Trajectory States data cluster
- AllWaypointRelPos -- array of double - List of all waypoint relative positions.
- WaypointRelPos - double - The waypoint relative position whose time is desired.

Outputs:

- Time_Secs - double - The trajectory time for this waypointRelPos.

PathPlanner_Trajectory_GenerateStates



Internal routine to help create a trajectory from a path.

PathPlanner_Trajectory_GenerateStates_Pass1



Internal routine to help create a trajectory from a path.

PathPlanner_Trajectory_GenerateStates_Pass2

Internal routine to help create a trajectory from a path.

PathPlanner_Trajectory_GenerateStates_Pass3

Internal routine to help create a trajectory from a path.

PathPlanner_Trajectory_GetAllWaypointRelPos

Get a list of all waypoint relative position for the states.

Inputs:

-- Trajectory -- PathPlanner Trajectory data cluster

Outputs:

- AllWaypointRelPos - double array - All the waypoint relative positions

PathPlanner_Trajectory_GetEndState

Get the end state of the trajectory

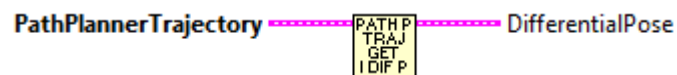
Inputs:

- 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

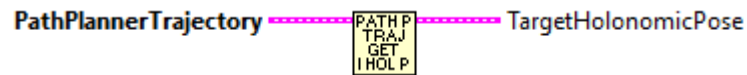
Inputs:

- 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:

- path - cluster - path definition

Outputs:

- NextRotationTargetIndex - integer -
- Found - boolean - rotation was found

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
- TrajectoryEvents - cluster - created trajectory events data cluster

PathPlanner_Trajectory_New_States



Generate a PathPlannerTrajectory

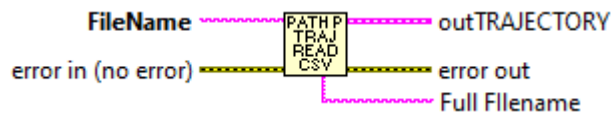
Inputs:

- 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.

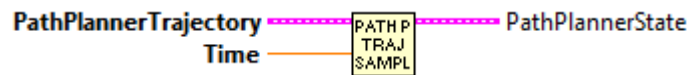
Notes on file naming:

- 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

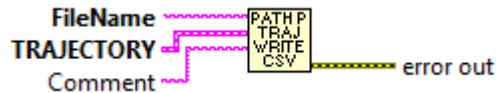
Inputs:

- 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

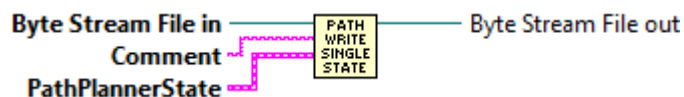
Notes on file naming:

- 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: %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.

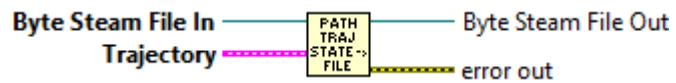
Parameters:

- 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:

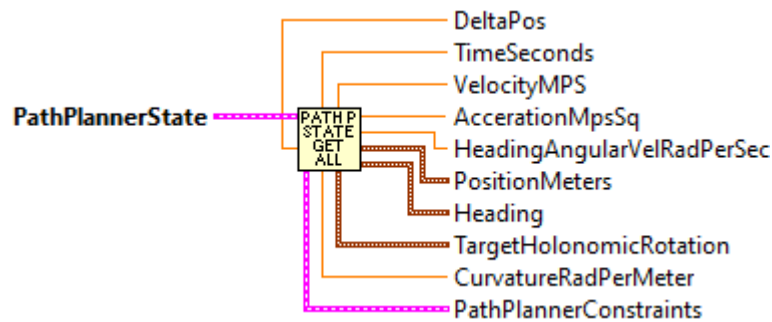
- ByteStreamIn - File stream
- Trajectory - Data structure containing trajectory

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/s² (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_GetWaypointRelPos



Gets WaypointRelPos of trajectory state

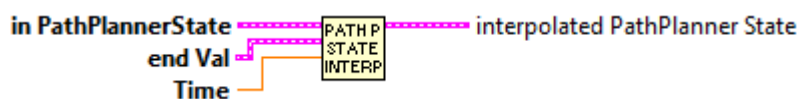
Inputs

- PathPlannerTrajectoryState -- cluster -- State data structure

Outputs:

- WaypointRelPos - double - The WaypointRelPos at this state

PathPlanner_TrajectoryState_Interpolate



Interpolate between this state and the given state

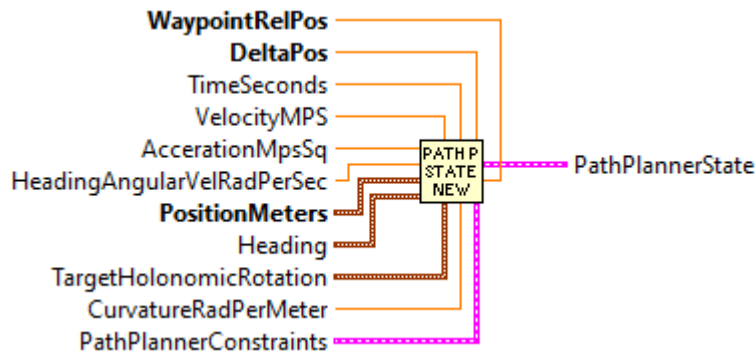
Inputs:

- 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/s² (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)
- WaypointRelPos - double - The waypoint relative pos for this state.

Outputs

- PathPlannerTrajectoryState -- cluster -- Newly created state

PathPlanner_TrajectoryState_Reverse



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

Inputs:

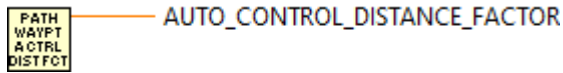
- trajectoryState - cluster - this trajectory state

Outputs:

- ReversedState- trajectorystate - The reversed state

Waypoint

PathPlanner_Waypoint_AutoCtrlDistFactor



Constant - AUTO_CONTROL_DISTANCE_FACTOR

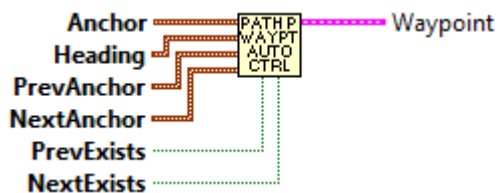
Input:

--

Outputs:

-- AUTO_CONTROL_DISTANCE_FACTOR -- value of the constant

PathPlanner_Waypoint_AutoCtrlPoints



Create a waypoint with auto calculated control points based on the positions of adjacent waypoints. This is used internally, and you probably shouldn't use this.

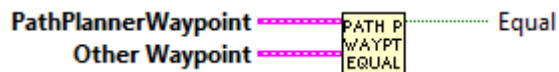
Inputs:

- anchor - translation2d - The anchor point of the waypoint to create
- heading - rotation2d - The heading of this waypoint
- prevAnchor - translation2d - The position of the previous anchor point. This can be null for the start point
- nextAnchor - translation2d - The position of the next anchor point. This can be null for the end point
- PrevExists - boolean - True if prevAnchor is not null
- NextExists - boolean - True if nextAnchor is not null

Outputs:

- Waypoint - cluster - Waypoint with auto calculated control points

PathPlanner_Waypoint_Equals



Determine if two waypoints are equal

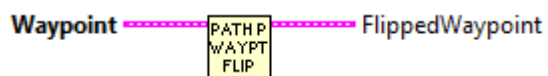
Inputs:

- Waypoint - cluster - defined waypoint cluster
- OtherWaypoint - cluster - defined waypoint cluster

Outputs:

- Equal - boolean - TRUE if both waypoints are the same

PathPlanner_Waypoint_Flip



Flip this waypoint to the other side of the field, maintaining a blue alliance origin

Inputs:

- Waypoint - cluster - The waypoint to flip

Outputs:

- FlippedWaypoint - cluster - The flipped waypoint

PathPlanner_Waypoint_New



Cluster used to describe a waypoint for a Bezier curve based path

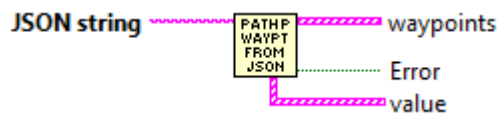
Inputs:

- prevControl - translation2d - Previous control point. Values $< -9.9E+28$ are used to determine null. (Default: null)
- anchor - translation2d - Anchor point
- nextControl - translation2d - Next control point. Values $< -9.9E+28$ are used to determine null. (Default: null)

Outputs:

- waypoint - cluster - waypoint cluster

PathPlanner_Waypoint_fromJSON



Create a waypoint from JSON text.

Inputs:

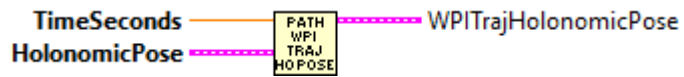
- JSON String - string - JSON containing waypoint to parse and convert to waypoint

Outputs:

- Waypoints - array of waypoint - The waypoints created from JSON
- error - boolean - TRUE if an error occurred.
- value - array of cluster - raw parse of JSON for debugging

WPITrajHolPose

PathPlanner_WPITrajHolPose_New



Create a WPITrajHolPose data cluster.

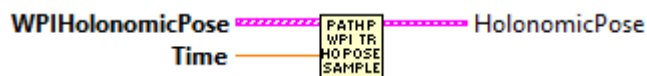
Inputs:

- TimeSeconds - double - Trajectory sample time
- HolonomicPose - pose2d - The holonomic Pose at this time.

Outputs:

- WPITrajHolonomicPose - cluster - created data structure
-

PathPlanner_WPITrajHolPose_Sample



Sample the WPI Trajectory Holonomic Pose array at a point in time

Inputs:

- WPIHolonomicPose - array - WPI Holonomic Pose data cluster
- time -- The time to sample

Outputs:

- HolonomicPose - pose2d - Desired holonomic pose at the given time.

Type Definitions

TypeDef

TypeDef-PathPlannerCmdFindConfig



Configuration parameters for Path Planner Holonomic Controller.

Contains:

- Holonomic_Ctrl_Pack_Tuning - cluster
- Drive Base Radius M - double
- Max Wheel Speed M/S - double
- Tolerance SI - pose2d

PathPlanner_Cmd_FindConfig

Goal Position	PathPlannerPathConstraints
X 0.000	MaxVelocityMps; 0
Y 0.000	MaxAccelerationMpsSq; 0
GoalEndState	MaxAngularVelRadPsEc; 0
VelocityMPS 0	MaxAngularAccRadPSecSq 0
Rotation	NominalVoltageVolts 12
VALUE 0.0000	Unlimited Exists <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
COS 1.0000	Rotation Delay Distance M 0
SIN 0.0000	

TypeDef-PathPlannerCmdFollowHolConfig



Configuration parameters for Path Planner Holonomic Controller.

Contains:

- Holonomic_Ctrl_Pack_Tuning - cluster
- Drive Base Radius M - double
- Max Wheel Speed M/S - double
- Tolerance SI - pose2d

PathPlanner_Cmd_FollowHolConfig

HOLONOMIC_CTRL_PACK_TUNING

X PID Tuning

Kp

0

Ki

0

Kd

0

MaximumIntegral

9.9E+30

MinimumIntegral

-9.9E+30

IntegralZone

9.9E+30

Y PID Tuning

Kp

0

Ki

0

Kd

0

MaximumIntegral

9.9E+30

MinimumIntegral

-9.9E+30

IntegralZone

9.9E+30

thetaProfiledPID_CTRL

Controller

Kp

0.00000E+0

Ki

0.00000E+0

Kd

0.00000E+0

Period

0.00000E+0

Max_Integra

1.00000E+0

Min_Integral

-1.00000E+0

Max_Input

0.00000E+0

Min_Input

0.00000E+0

Measurement

0.00000E+0

Kf

0.00000E+0

DerivativeFilter



Min_Output

-9.99000E+35

Prev_F_P

0.00000E+0

AntiWindupActive



IntegralZone

9.90000E+30

Continuous



PositionError

0.00000E+0

VelocityError

0.00000E+0

PrevError

0.00000E+0

Total_Integral

0.00000E+0

PositionTolerance

5.00000E-2

VelocityTolerance

9.99000E+35

SetPoint

0.00000E+0

Prev_Deriv

0.00000E+0

Prev_Deriv 2

0.00000E+0

Max_Output

9.90000E+35

Prev_F_P_2

0.00000E+0

Goal

Position

0.00000E+0

Velocity

0.00000E+0

Setpoint

Position

0.00000E+0

Velocity

0.00000E+0

Constraint

MaxVelocity

0.00000E+0

MaxAcceleration

0.00000E+0

TypeDef-PathPlannerCmdFollowLTVUnicycleConfig

Configuration parameters for Path Planner Holonomic Controller.

Contains:

- Holonomic_Ctrl_Pack_Tuning - cluster
- Drive Base Radius M - double
- Max Wheel Speed M/S - double
- Tolerance SI - pose2d

PathPlanner_Cmd_FollowLTVUnicycleConfig

LTV_Unicycle_Ctrl_Model

Max X Pos Meas Tol M
0.0625

Max Y Pos Meas Tol M
0.125

Max Heading Meas Tol Rad
0.2

Max linear ctrl effort M/S
1

Max angular ctrl effort Rad/S
2

Max Linear Vel M/S
9


Track Width Meters
1

Max Speed M/S
4

X Tolerance M
0.0762

Y Tolerance M
0.0762

Rotation Tolerance DEG
0.0872665

Trigger Events


ControlPastEnd_Sec
3

Period
0.02

TypeDef-PathPlannerCmdFollowRamConfig



Configuration parameters for Path Planner Holonomic Controller.

Contains:

- Holonomic_Ctrl_Pack_Tuning - cluster

- Drive Base Radius M - double
- Max Wheel Speed M/S - double
- Tolerance SI - pose2d

PathPlanner_Cmd_FollowRamConfig

RAMSETE_EXE_TUNING SI

Track Width M
2.0000

Max Speed M/S
2.0000

B
2.0000

Zeta
0.7000

X Tolerance M
2.0000

Y Tolerance M
0.7000

Heading Tolerance RAD
0.7000

ControlPastEnd_Sec
3

Trigger Events
☒

TypeDef-PathPlannerCommandUtil



The CommandUtil cluster stores the data for commands issued by Event Markers.

The cluster contains:

- name - string

- type - CommandUtilType_Enum
- WaitTime - double
- PathName - string
- Commands - array of cluster containing
 - Type - CommandUtilType_Enum
 - WaitTime - double
 - Name - string
 - PathName - string

The image shows a LabVIEW front panel for a cluster named "CommandUtil". The cluster contains the following fields:

- name**: A text input field.
- type**: A dropdown menu with "Unknown" selected.
- waitTime**: A numeric control with a value of 0.
- pathName**: A text input field.
- commands**: A numeric control with a value of 0, which is a sub-cluster. The sub-cluster contains:
 - type**: A dropdown menu with "Unknown" selected.
 - waitTime**: A numeric control with a value of 0.
 - name**: A text input field.
 - pathName**: A text input field.

TypeDef-PathPlannerConstraintsZone

A zone on a path with different kinematic constraints

Contains:

- MinPosition - double - Starting waypoint relative position of the zone
- MaxPosition - double - End waypoint relative position of the zone
- Constraint - cluster - The Constraints to apply within the zone
- Present - boolean - flag indicting this cluster is not null

ConstraintsZone

minPosition
0

maxPosition
0

Present
☒

Constraints

MaxVelocityMps;
0

MaxAccelerationMpsSq;
0

MaxAngularVelRadPsEc;
0

MaxAngularAccRadPSecSq
0

NominalVoltageVolts
12

Unlimited ☒ Exists ☒

TypeDef-PathPlannerCtrlHolConfig

Configuration parameters for Path Planner Holonomic Controller.

Contains:

- Holonomic_Ctrl_Pack_Tuning - cluster
- Drive Base Radius M - double
- Max Wheel Speed M/S - double
- Tolerance SI - pose2d

PathPlanner_Ctrl_HolDrvConfig

HOLONOMIC_CTRL_PACK_TUNING

X PID Tuning

Kp

0

Ki

0

Kd

0

MaximumIntegral

9.9E+30

MinimumIntegral

-9.9E+30

IntegralZone

9.9E+30

Y PID Tuning

Kp

0

Ki

0

Kd

0

MaximumIntegral

9.9E+30

MinimumIntegral

-9.9E+30

IntegralZone

9.9E+30

thetaProfiledPID_CTRL

Controller

Kp

0.00000E+0

Ki

0.00000E+0

Kd

0.00000E+0

Period

0.00000E+0

Max_Integra

1.00000E+0

Min_Integral

-1.00000E+0

Max_Input

0.00000E+0

Min_Input

0.00000E+0

Measurement

0.00000E+0

Kf

0.00000E+0

DerivativeFilter

☒

Min_Output

-9.99000E+35

Prev_F_P

0.00000E+0

AntiWindupActive

☒

IntegralZone

9.90000E+30

Continuous

☒

PositionError

0.00000E+0

VelocityError

0.00000E+0

PrevError

0.00000E+0

Total_Integral

0.00000E+0

PositionTolerance

5.00000E-2

VelocityTolerance

9.99000E+35

SetPoint

0.00000E+0

Prev_Deriv

0.00000E+0

Prev_Deriv 2

0.00000E+0

Max_Output

9.90000E+35

Prev_F_P_2

0.00000E+0

Goal

Position

0.00000E+0

Velocity

0.00000E+0

Setpoint

Position

0.00000E+0

Velocity

0.00000E+0

Constraint

MaxVelocity

0.00000E+0

MaxAcceleration

0.00000E+0

TypeDef-PathPlannerCtrlLTVUnicycleConfig

Configuration parameters for Path Planner Differential Drive LTV Unicycle Controller.

Contains:

- LTV_Unicycle_Ctrl_Model - cluster
- LTV_Unicycle_Ctrl_Tolerance - cluster
- Track Width M - double
- Max Wheel Speed M/S - double

PathPlanner_Ctrl_LTVUnicycle_Config

LTV_Unicycle_Ctrl_Model	Track Width M
Max X Pos Meas Tol M 0.0625	0
Max Y Pos Meas Tol M 0.125	Max Wheel Speed M/S 0
Max Heading Meas Tol Rad 0.2	
Max linear ctrl effort M/S 1	
Max angular ctrl effort Rad/S 2	
Max Linear Vel M/S 9	

LTV_Unicycle_Ctrl_Tolerance
Max X Tolerance M 2
Max Y Tolerance M 0.2
Max Heading Tolerance Rad 1

TypeDef-PathPlannerEventMarker



Position along the path that will trigger a command when reached

Contains

- triggerName - string - The name of the trigger this event marker will control
- position - double - The waypoint relative position of the marker
- endPosition - double - The end waypoint relative position of the event's zone. A value of -1.0 indicates that this event is not zoned.
- Command - cluster -- Contains the command(s) to execute. Can be null to not run a command.

EventMarker

Position

0

EndPosition

-1

TriggerName

Command

name

type

Unknown

waitTime

0

pathName

commands

0

type

Unknown

waitTime

0

name

pathName

TypeDef-PathPlannerFinderDoublePair



PathPlannerFinderDoublePair

A

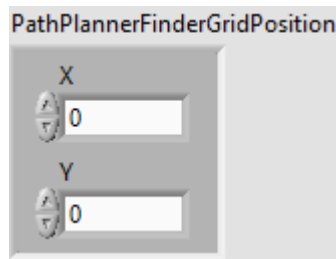
0

B

0

TypeDef-PathPlannerFinderGridPosition

PATH P
GRID
POS



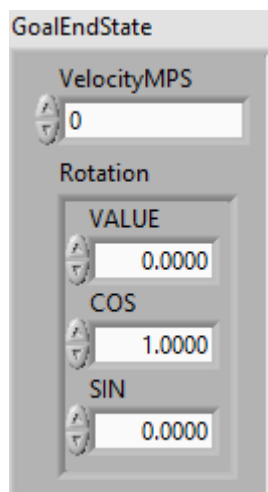
TypeDef-PathPlannerGoalEndState

PATH P
END
STATE

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

contains:

- velocityMPS - double - The goal end velocity (M/S)
- rotation - Rotation2d - The goal rotation



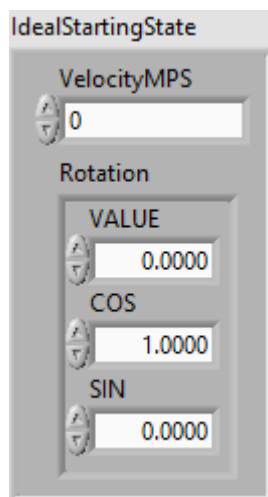
TypeDef-PathPlannerIdealStartingState



Describes the ideal starting state of the robot when starting a path

contains:

- velocityMPS - double - The ideal starting velocity (M/S)
- rotation - Rotation2d - The ideal starting rotation



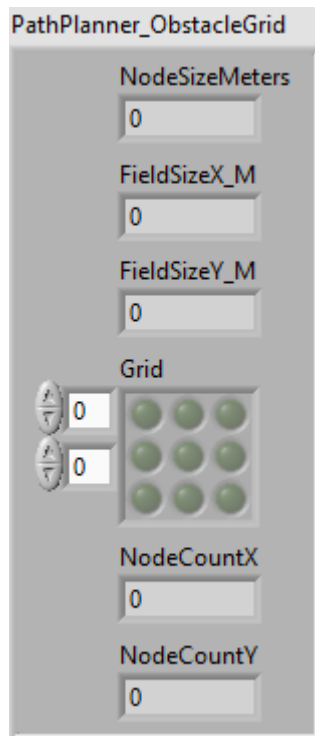
TypeDef-PathPlannerObstacleGrid



Contains definition of Obstacle grid. This grid is used to define areas pathfinding should avoid.

Contains:

- NodeSizeMeters - double - Size of each grid element. Meters
- Field size - translation2d - X and Y size of grid. Meters
- Grid - boolean array - Array containing obstacles to avoid when finding a path. Index into array is X, Y, where X and Y are increments of the node size in meters. If node size is 0.2 then array element (1,2) = position 0.2, 0.4 meters.



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

PathPlannerPath

Name

Waypoints

0

prevControl

X

0.000

Y

0.000

anchor

X

0.000

Y

0.000

nextControl

X

0.000

Y

0.000

PrevExists

NextExists

RotationTargets

0

position

0

rotation

VALUE

0.0000

COS

1.0000

SIN

0.0000

Exists

PointTowardsZones

0

Name

TargetPosition

X

0.000

Y

0.000

RotationOffset

VALUE

0.0000

COS

1.0000

SIN

0.0000

MinPosition

0

MaxPosition

0

ConstraintZones

0

minPosition

0

maxPosition

0

Present

Constraints

MaxVelocityMps;

0

MaxAccelerationMpsSq;

0

MaxAngularVelRadPsEc;

0

MaxAngularAccRadPSecSq

0

NominalVoltageVolts

12

Unlimited

Exists

EventMarkers

Position

0

EndPosition

-1

TriggerName

GlobalConstraints

MaxVelocityMps;

0

MaxAccelerationMpsSq;

0

MaxAngularVelRadPsEc;

0

MaxAngularAccRadPSecSq

0

NominalVoltageVolts

12

Unlimited

Exists

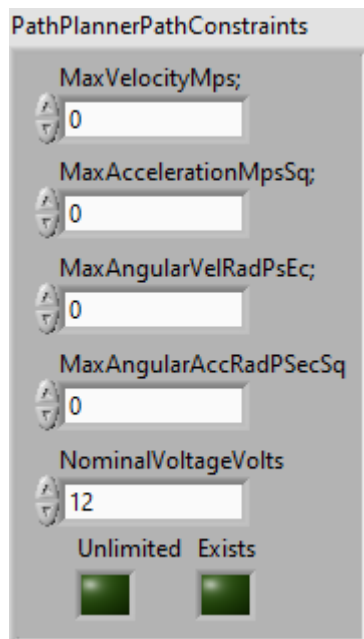
TypeDef-PathPlannerPathConstraints



Kinematic path following constraints

Contains:

- maxVelocityMPS - Max linear velocity (Meters/Second)
- maxAccelerationMPSSq - Max linear acceleration (Meters/Second²)
- maxAngularVelocityRadPerSec - Max Angular Velocity (Radians/Second)
- maxAngularAccelerationRadPerSecSq - Max Angular Acceleration (Radians/Second²)
- NominalVoltageVolts -- The nominal battery voltage (Volts)
- Unlimited -- Should the constraints be unlimited
- 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
- MaxV - double - The max velocity at this point
- rotationTarget - RotationTarget - The target rotation at this point
- pathConstraints - PathConstraints - The constraints applied to this point
- waypointRelativePos - double - The waypoint relative position of this point. Used to determine proper event marker timing

PathPlannerPoint

Position	Constraints
X 0.000	MaxVelocityMps; 0
Y 0.000	MaxAccelerationMpsSq; 0
DistAlongPath 0	MaxAngularVelRadPsEc; 0
MaxV 9E+300	MaxAngularAccRadPSecSq 0
RotationTarget	NominalVoltageVolts 12
position 0	Unlimited <input type="checkbox"/> Exists <input type="checkbox"/>
rotation	WaypointRelativePos 0
VALUE 0.0000	
COS 1.0000	
SIN 0.0000	
Exists <input type="checkbox"/>	

TypeDef-PathPlannerPointTowardsZone



Cluster containing zone on a path that will force the robot to point towards a position on the field

Contains:

- name - string - The name of this zone. Used for point towards zone triggers
- targetPosition - translation2d - The target field position in meters
- rotationOffset - rotation2d - A rotation offset to add on top of the angle to the target position. For example, if you want the robot to point away from the target position, use a rotation offset of 180 degrees
- minPosition - double - Starting waypoint relative position of the zone

- maxPosition - double - End waypoint relative position of the zone

PointTowardsZone

Name

TargetPosition

RotationOffset

X

0.000

Y

0.000

MinPosition

0

MaxPosition

0

VALUE

0.0000

COS

1.0000

SIN

0.0000



TypeDef-PathPlannerRectangularObstacle



RectangularObstacle

CornerPos1

CornerPos2

X

0.000

Y

0.000

X

0.000

Y

0.000



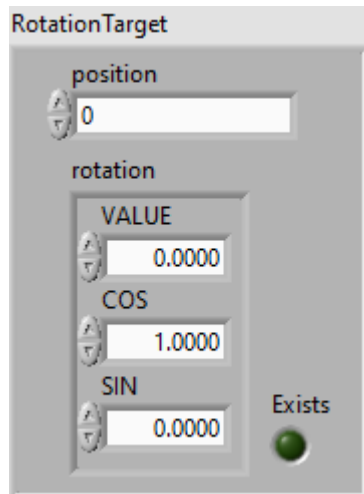
TypeDef-PathPlannerRotationTarget



A target holonomic rotation at a position along a path

Contains:

- position - double - Waypoint relative position of this target
- rotation - rotation2d - Target rotation
- exists - boolean - TRUE if not null



TypeDef-PathPlannerSwerveModTrajState



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/s²
- 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

PathPlannerSwerveModuleTrajectoryState

FieldPos	SWERVE_DRIVE_MODULE_STATE
X 0.000	Speed 0
Y 0.000	Angle VALUE 0.0000
FieldAngle VALUE 0.0000	COS 1.0000
COS 1.0000	SIN 0.0000
SIN 0.0000	
DeltaPos 0	

TypeDef-PathPlannerTrajectory



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

WaypointRelPos

0

TargetHolonomicRotation

VALUE

0.0000

COS

1.0000

SIN

0.0000

CurvatureRadPerMeter

0

DeltaPos

0

PathPlannerConstraints

MaxVelocityMps;

0

MaxAccelerationMpsSq;

0

MaxAngularVelRadPsEc;

0

MaxAngularAccRadPSecSq

0

NominalVoltageVolts

12

Unlimited Exists

TypeDef-PathPlannerTrajectoryEvent



TrajectoryEvent

TimeStampSec
0

CommandUtil

name
[]

type
Unknown

waitTime
0

pathName
[]

commands
0

type
Unknown

waitTime
0

name
[]

pathName
[]

TypeDef-PathPlannerTrajectoryEvents



Trajectory created from a pathplanner path

Contains:

- States - array - List of trajectory states

PathPlannerTrajectoryEvents

EventCommands

0

TimeStampSec

0

CommandUtil

name

type

Unknown

waitTime

0

pathName

commands

0

type

Unknown

waitTime

0

name

pathName

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/s²
- 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
- WaypointRelPos - double - Waypoint relative position

PathPlannerState24

<p>TimeSeconds</p> <p>0</p> <p>VelocityMPS</p> <p>0</p> <p>AccerationMpsSq</p> <p>0</p> <p>HeadingAngularVelRadPerSec</p> <p>0</p> <p>PositionMeters</p> <p>X</p> <p>0.000</p> <p>Y</p> <p>0.000</p> <p>Heading</p> <p>VALUE</p> <p>0.0000</p> <p>COS</p> <p>1.0000</p> <p>SIN</p> <p>0.0000</p> <p>WaypointRelPos</p> <p>0</p>	<p>TargetHolonomicRotation</p> <p>VALUE</p> <p>0.0000</p> <p>COS</p> <p>1.0000</p> <p>SIN</p> <p>0.0000</p> <p>CurvatureRadPerMeter</p> <p>0</p> <p>DeltaPos</p> <p>0</p> <p>PathPlannerConstraints</p> <p>MaxVelocityMps;</p> <p>0</p> <p>MaxAccelerationMpsSq;</p> <p>0</p> <p>MaxAngularVelRadPsEc;</p> <p>0</p> <p>MaxAngularAccRadPSecSq</p> <p>0</p> <p>NominalVoltageVolts</p> <p>12</p> <p>Unlimited Exists</p>
---	---

TypeDef-PathPlannerTrajectoryState25



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/s²

- 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

PathPlannerState25

<p>TimeSeconds</p> <p>0</p>	<p>LinearVelocity</p> <p>0</p>	<p>PathPlannerSwerveModuleTrajectoryState</p>		<p>Constraints</p>
<p>FieldSpeeds</p> <p>vX</p> <p>0.000</p> <p>vY</p> <p>0.000</p> <p>vOmega</p> <p>0.000</p>	<p>Heading</p> <p>VALUE</p> <p>0.0000</p> <p>COS</p> <p>1.0000</p> <p>SIN</p> <p>0.0000</p>	<p>FieldPos</p> <p>X</p> <p>0.000</p> <p>Y</p> <p>0.000</p>	<p>SWERVE_DRIVE_MODULE_STATE</p> <p>Speed</p> <p>0</p> <p>Angle</p> <p>VALUE</p> <p>0.0000</p> <p>COS</p> <p>1.0000</p> <p>SIN</p> <p>0.0000</p>	<p>MaxVelocityMps;</p> <p>0</p> <p>MaxAccelerationMpsSq;</p> <p>0</p> <p>MaxAngularVelRadPsEc;</p> <p>0</p> <p>MaxAngularAccRadPSecS;</p> <p>0</p> <p>NominalVoltageVolts</p> <p>12</p> <p>Unlimited <input type="checkbox"/> Exists <input type="checkbox"/></p>
<p>pose</p> <p>TRANSLATION</p> <p>X</p> <p>0.000</p> <p>Y</p> <p>0.000</p> <p>ROTATION</p> <p>VALUE</p> <p>0.0000</p> <p>COS</p> <p>1.0000</p> <p>SIN</p> <p>0.0000</p>	<p>DeltaPos</p> <p>0</p> <p>DeltaRot</p> <p>VALUE</p> <p>0.0000</p> <p>COS</p> <p>1.0000</p> <p>SIN</p> <p>0.0000</p> <p>WaypointRelPos</p> <p>0</p>	<p>FieldAngle</p> <p>VALUE</p> <p>0.0000</p> <p>COS</p> <p>1.0000</p> <p>SIN</p> <p>0.0000</p> <p>DeltaPos</p> <p>0</p>		

TypeDef-PathPlannerWPITrajHolonomicPose



The WPITrajHolonomicPose stores the Holonomic pose (position of robot and holonomic rotation) for a PathPlanner trajectory when iits states are converted to WPI Trajectory states. This data structure can be sampled similar to other trajectory states. It allows the user to replace the normal WPI trajectory pose with this to help control holonomic robots.

Contains

- TimeSeconds - double
- HolonomicPose - pose2d

The image shows a LabVIEW control panel for the WPI Traj Holonomic Pose data type. It features a title bar labeled 'WPITrajHolonomicPose'. Below the title bar, there is a 'TimeSeconds' section with a numeric display showing '0'. Below that is a 'HolonomicPose' section. This section is divided into two sub-sections: 'TRANSLATION' and 'ROTATION'. The 'TRANSLATION' section contains two numeric displays: 'X' with a value of '0.000' and 'Y' with a value of '0.000'. The 'ROTATION' section contains three numeric displays: 'VALUE' with a value of '0.0000', 'COS' with a value of '1.0000', and 'SIN' with a value of '0.0000'.

TypeDef-PathPlannerWaypoint



Class used to describe a waypoint for a Bezier curve based path

Contains:

- prevControl - translation2d - Previous control point
- anchor - translation2d - Anchor point
- nextControl - translation2d - Next control point
- PrevExists - boolean - TRUE if prevControl is not null
- NextExists - boolean - TRUE if nextControl is not null

PathPlannerWaypoint

prevControl

X

0.000

Y

0.000

anchor

X

0.000

Y

0.000

nextControl

X

0.000

Y

0.000

PrevExists

NextExists

Enumerated Type Definitions

Enum

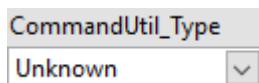
Enum-PathPlanner_CommandUtilType_ENUM



Enumerated variable type for the type of command contained in the Command Util cluster. This is converted from a string contained in the path JSON.

The types are:

- Unknown
- Wait
- Named
- Path
- Sequential
- Parallel
- Race
- Deadline



Enum-PathPlanner_FieldSymmetry_ENUM



Enum representing the different types of field symmetry

Values:

- kRotational -- Field is rotationally symmetric. i.e. the red alliance side is the blue alliance side rotated by 180 degrees

- kMirrored -- Field is mirrored vertically over the center of the field

PathPlanner_FieldSymmetry

kRotational

Enum-PathPlanner_PathFinding_Cmd_ENUM

Path
Find
Cmd
ENUM

PathFindingCommand

Do Nothing