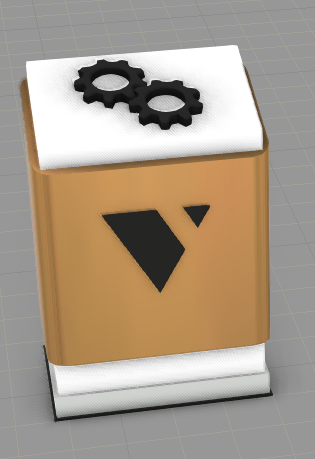
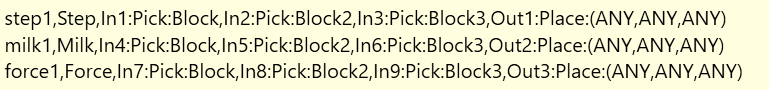
Route Control Reference Guide

A guide for using Routes in Works Library

Version: November 04, 2019

|  |
| --- |
| **Disclaimer:**  This document was made using Visual Components 4.1.2 and version 5.3 of the Works library.  The Revision metadata property of a Works Task Control was 13. |



Contents

[1 Description 1](#_Toc23756670)

[2 Route Definition and syntax 2](#_Toc23756671)

[2.1 Types of actions 3](#_Toc23756672)

[Pick 3](#_Toc23756673)

[Place 3](#_Toc23756674)

[Move 4](#_Toc23756675)

[Wait 4](#_Toc23756676)

[2.2 Multi-picking and placing 5](#_Toc23756677)

[3 Route types and logic 6](#_Toc23756678)

[3.1 Milk 6](#_Toc23756679)

[3.2 Step 6](#_Toc23756680)

[3.3 StepOnDemand and MilkOnDemand 7](#_Toc23756681)

[3.4 Force 7](#_Toc23756682)

[3.5 Priorities 7](#_Toc23756683)

# Description

Route Control is used to predetermine certain Pick, Place, Wait, and Move patterns for Works Resources such as AGVs (found under “Mobile Robotics” in eCatalog), Human and Forklift (under “Works Resources”) except “Works Robot Controller” which does not support Route Control.

Products are fed to the route control by using "route" as the *TaskName* in a Feed task instead of directly referring to a *Tasklist* item of a resource.

Each defined route needs a unique name. The name will be used in a resource's *Tasklist* for the Task Control to be able to dispatch resources to the correct route.

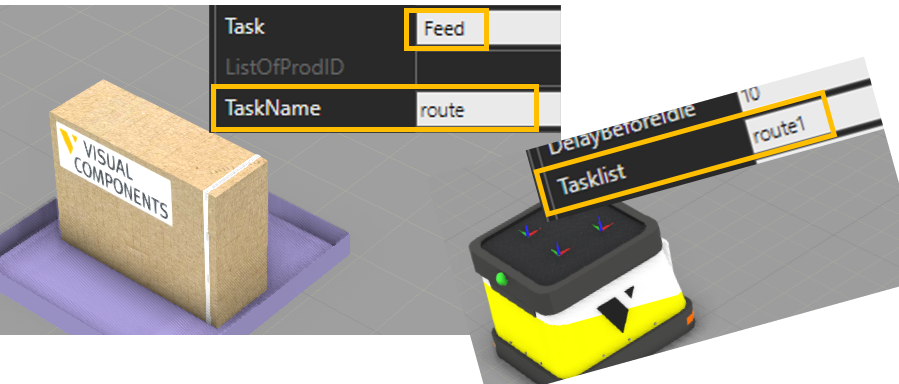
There are five types of routes: **Milk**, **Step**, **Force, MilkOnDemand** and **StepOndemand**.

**Milk** and **Step** route types are run infinitely, meaning that it’s not possible to carry out any other tasks with resources than running the assigned route. **MilkOnDemand** and **StepOndemand** are dispatched/activated on a specific condition and are ran once, meaning that it’s possible to carry out other tasks while the route is not active.

**Force** route will be activated/dispatched only when all the actions (steps) in the route can be completed (the Works Task Controller received all Feeds and Needs). It's possible to carry out other tasks while the route can't be executed.

Routes have the highest priority in the Works Task Control and are handled before anything else. This means that the resources are dispatched to routes first whenever possible.

When prioritizing resources to a route the selected *ResourcePriority* strategy of Works Task Control is respected (e.g. Nearest/Least Utilized/Cyclic).



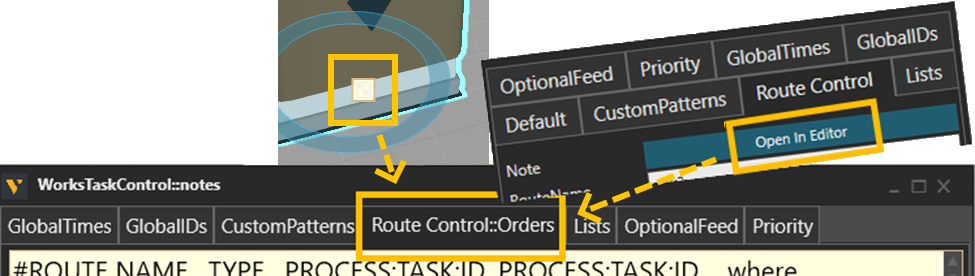
A video tutorial can be found from Visual Component Academy

<http://academy.visualcomponents.com/lessons/define-routes-for-mobile-robots/>

**Note:** The version of Works Library is **older in the video** and has differences in its logic and syntax to the current one.

# Route Definition and syntax

Routes are defined in Works Task Control’s **Route Control** Note tab.



Route is defined as a comma “,” separated list of values.

**<Name>,<Type>,<Action>,<Action>…**

where

**<Name>:** Unique name for the route that is referenced by the resources

**<Type>:** Type of the route one of these Milk, Step or Force

<**Action>:** a list of three values separated by a colon “:”.

The list of actions defines the route pattern. One action consists of the following possible values:

**<Process><Task><Id>**

where

**<Process>:** Name of a component. Typically, a Works Process or “ANY” for a wildcard

**<Task>:** Type of action: Pick, Place, Wait or Move

**<Id>:** Identifier of a product or a parameter used in Wait.

Example:

route1,Step,A:Pick:111,B:Pick:222,C:Place:(111,222)

## Types of actions

### Pick

Pick one item/product from a process. For picking multiple items from the same process, refer to “Multi picking and placing”.

In pick action the syntax is as follows:

**<Process>:Pick:<ProdID>**

e.g. “WP1:Pick:111”

where

**<Process>:** is the name of the Works Process component or “ANY”. If ANY is used as the value, the process that is feeding the given ProdID will be chosen. If multiple processes are feeding the same ProdID the process with higher priority (Task Control:Priority) will be chosen.

**<ProdID>:** is the id of the product or “ANY”. If ANY is used as the value, the first product created or arrived in the process is typically picked (FIFO).

Note: Both the ProdID and the Process can’t be ANY

### Place

Place one item/product to a process. For placing multiple items to the same process, refer to “Multi picking and placing”.

In place action the syntax is as follows:

**<Process>:Place:<ProdID>**

e.g. “WP1:Place:111”

where

**<Process>:** is the name of the Works Process component or “ANY”. If ANY is used as the value, the process that is needing the given ProdID will be chosen. If multiple processes are needing the same ProdID the process with higher priority (Task Control:Priority) will be chosen.

**<ProdID>:** is the id of the product or “ANY”. If ANY is used as the value, the last picked matching product will be placed in the process (LIFO).

Note: Both the ProdID and the Process can’t be ANY

### Move

Sends a resource to given location e.g. add via points to the route.

**<Location>:Move:**

e.g. “WP1:Move:”

where

**<Location>:** name of a component (destination). If the given component is a Works Process the dispatched resource will move to and align with ResourceLocation of the process, otherwise to the origin of a component. This means that any component can be used as the destination.

***Note***: Move has no third parameter, but the colon is required. Any value in the third parameter will be neglected.

### Wait

Hold a resource still for given period.

**<Process>:Wait:<Time>**

e.g. “:Wait:10”or“WP1:Wait:MyProcessTime”

where

**<Process> (Optional):** Name of a Works Process where the time value is read from. It can be left blank if time is given as numeric value instead of the name of a UserVariable.

**<Time>:** Duration in seconds or name of a UserVariable defined in the given Process.

## Multi-picking and placing

In Pick and Place actions it is possible to define multiple ProdIDs for picking and placing from a single process by typing the ProdID value within parentheses and separating ProdIDs by a comma e.g.

WP1:Pick:(111,111)

Will pick two products with id 111 from the WP1. It is equivalent to: “WP1:Pick:111,WP1:Pick:111”.

There is a difference if the given Process is “ANY” e.g.

ANY:Pick:(111,111)

It is **not** equivalent to: “ANY:Pick:111,ANY:Pick:111”. This is because multi-pick is bound to a single process i.e. the products are picked from the same process. Defining two separate Pick actions with ANY process may lead to picking each product from different process. It is depending on the product availability.

# Route types and logic

Three different route types are provided with different logic.

**Move** and **Wait** actions are always executed immediately in their turn in all route types.

Move action to the given process is prepended to every Pick and Place action automatically unless the Process is ANY.

## Milk

A resource dedicated to a Milk route is dispatched immediately at the beginning of the simulation to the route and will run the route infinitely (in a loop). Pick and Place actions in the Milk route are executed if possible; otherwise they are skipped. The resource travels to the process of each action and checks if the action can be carried out except for ANY as the Process.

In **Pick** action, the resource travels to the process (if not *ANY*), checks if the product is available for picking, picks it or skips and proceeds to the next action. In case the process is *ANY*, the check is done before traveling to any process and if a matching process with the correct active Feed is found the action is executed, otherwise resource proceeds to the next action.

In **Place** action, the resource travels to the process (if not *ANY*), checks if the product is available for placing (onboard) and the process has active Need for the product, places it or skips and proceeds to next action. In case the process is *ANY*, the check is done before traveling to any process and if a matching process with the correct active Need is found and the product is on board, the action is executed and the resource travels to the process and places the product, otherwise resource proceeds to the next action.

## Step

A resource dedicated to a Step route is dispatched immediately at the beginning of the simulation to the route and will run the route infinitely (in a loop). Pick and Place actions in Step route are executed when available. This means, that the resource travels to the Process and waits for the action to be completed at the process except for ANY as the Process.

In **Pick** action the resource travels to the process (if not *ANY*), checks if the product is available for picking, picks it or waits for the product to be available (active Feed). In case the process is *ANY*, the check is done before traveling to any process and if a matching process with the correct active Feed is found the action is executed, otherwise the resource stays in place and waits for the Feed.

In **Place** action, the resource travels to the process (if not *ANY*), checks if the process has active Need for the product, places it or waits for the Need. In case the process is *ANY*, the check is done before traveling to any process and if a matching process with the correct active Need is found, the action is executed and the resource travels to the process and places the product; otherwise the resource stays in place and waits for the Feed.

**Hint**: If using ANY as the process, define a waiting location for a resource by typing Move action before Pick or Place action.

## StepOnDemand and MilkOnDemand

A resource allocated to StepOndemand route is dispatched when a Feed for the **first Pick action** is available whereas MilkOndemand is dispatched when a Feed for **any Pick action** in the route is available i.e. received by the Task Controller. This enables other tasks to be carried out by the resource meanwhile the route is not active.

Otherwise, both route types are like Step or Milk routes.

## Force

A resource allocated to a Force route is dispatched when all actions in the route are available. The dispatch check is done continuously when the resource itself is available (not busy).

Note that in **multi-pick** and **multi-place** actions, all Feeds and/or Needs must be available during the check. This means that a “*Simultaneous*” option is a must in a Feed task and Need must be defined with multiple products separated by a comma.

Once dispatched to the route, the resource will execute the actions in order.

After the last action of the route, the dispatching check starts again.

## Priorities

Routes that defined higher in the list (Note) are processed first. This means that the order in which the routes are referred to in a resource Tasklist does not have any effect.

If multiple resources are available for a route, the closest resource to the next action (process) in a route (typically the first action) is dispatched first if the “*ResourcePriority”* property in Works Task Control is “*Nearest*” (default).

If multiple processes are available for a route action the process/task is defined by priority order defined in the Works Task Control “*Priority*” note.