# SimHub Dual Clutch Paddle Integration – Full Setup & Plugin Guide

Document generated on 2025-06-05 17:30 UTC

## 1. Project Overview

This guide describes how to implement a fully dynamic dual clutch system using SimHub, SimMagic hardware, and the LaunchPlugin architecture. The aim is to replicate dual-clutch launch behavior—where one paddle holds at a bite point and the other releases—while allowing the bite point to be adjusted in real time from SimHub’s UI or wheel-mounted controls.

## 2. Hardware Setup

- Ensure your SimMagic clutch paddles are configured in Sync OFF mode.

- Confirm both paddles operate independently:

• ClutchLeft: Rx axis

• ClutchRight: Ry axis

- Rename them in SimHub as `ClutchLeft` and `ClutchRight` for clarity.

- If Sync is re-enabled, SimHub reverts to reading only ClutchLeft (Rx) — safe fallback.

## 3. Software Signal Flow

1. Driver pulls paddles independently (ClutchLeft holds, ClutchRight releases).

2. SimHub plugin reads both paddles as analog inputs.

3. Plugin uses a user-defined bite point value.

4. Plugin calculates a `virtualClutch` output:

virtualClutch = ClutchRight + (ClutchLeft \* BitePoint)

5. This output is sent to SimHub Bridge and mapped to the clutch control in-game.

## 4. Plugin Implementation Steps

Open your `Plugin.cs` file from the LaunchPlugin project.

### 4.1 Add New Fields

At the top of your class, near other state variables, add:

private double \_clutchLeft = 0.0;  
private double \_clutchRight = 0.0;  
private double \_virtualClutch = 0.0;

### 4.2 Read Input Data

Inside the `DataUpdate` method, near the top where other input values are read, insert:

\_clutchLeft = Convert.ToDouble(pluginManager.GetPropertyValue("DataCorePlugin.GameRawData.Input.ClutchLeft") ?? 0.0);  
\_clutchRight = Convert.ToDouble(pluginManager.GetPropertyValue("DataCorePlugin.GameRawData.Input.ClutchRight") ?? 0.0);

### 4.3 Compute Virtual Clutch

Immediately after reading clutch values, calculate the virtual clutch:

double bitePointFraction = Settings.TargetBitePoint / 100.0; // 0 to 1 range  
\_virtualClutch = Math.Min(1.0, \_clutchRight + (\_clutchLeft \* bitePointFraction));

### 4.4 Expose Property to SimHub

In your `Init` method, add a delegate after existing property bindings:

this.AttachDelegate("LaunchPlugin\_VirtualClutch", () => \_virtualClutch);

## 5. Control Mapper Configuration

- Open SimHub > Control Mapper.

- In Bridge Output section, add new mapping:

• Source: `[Plugin.LaunchPlugin\_VirtualClutch]`

• Target: Game clutch axis

- Ensure this replaces direct mapping from Rx or Ry.

## 6. Optional Dashboard Elements

- Add a slider for `[Plugin.LaunchPlugin\_OptimalBitePoint]` to change bite live.

- Add bar graphs or indicators for:

• ClutchLeft (Rx)

• ClutchRight (Ry)

• VirtualClutch

## 7. Summary

This setup enables full dual-clutch launch emulation using your SimMagic paddles, while introducing a dynamic bite point that can be tuned per session or car. The LaunchPlugin architecture allows extensibility for logging, UI feedback, and clutch optimization over time.