





# **TMcraft Tutorial End-button Click Event**

Original Instructions

Document version: 1.0

Release date: 2024-10-29

This Manual contains information of the Techman Robot product series (hereinafter referred to as the TM AI Cobot). The information contained herein is the property of Techman Robot Inc. (hereinafter referred to as the Corporation). No part of this publication may be reproduced or copied in any way, shape or form without prior authorization from the Corporation. No information contained herein shall be considered an offer or commitment. It may be subject to change without notice. This Manual will be reviewed periodically. The Corporation will not be liable for any error or omission.

 and  logo are registered trademark of TECHMAN ROBOT INC. in Taiwan and other countries and the company reserves the ownership of this manual and its copy and its copyrights.

 **TECHMAN ROBOT INC.**

## Table of Contents

Revision History.....	3
1. Introduction.....	4
2. Concept.....	5
3. Sample Code.....	6
3.1 MainWindow.xaml.....	6
3.2 MainWindow.xaml.cs .....	6
3.3 Result .....	9

## Revision History

Revision	Date	Description
1.0	2024-10-29	Original release

## 1. Introduction

This document explains how a TMcraft plugin can capture the end-button click event. In some scenarios, developers may want their TMcraft plugin to be the sole recipient of the end-button event signal to prevent other components, such as TMflow or other TMcraft plugins, from responding. This tutorial will also cover obtaining exclusive ownership of the end-button event, ensuring that only the user's plugin processes the signal.

Readers should have the following prerequisites:

- Basic knowledge of programming C# and WPF
- Having read *TMcraft Toolbar Tutorial: Basic Development*
- Having read *TMcraft Toolbar API Function Manual* (API version 1.20 or above)
- Having read *TMcraft Node Tutorial: Basic Development*
- Having read *TMcraft Node API Function Manual* (API version 1.20 or above)

## 2. Concept

Since version 1.20, TMcraft API comes with a new class, `EndButtonEventProvider`, which is a collection of functions to the end-button click Event:

- `bool HasEndButtonEventOwnership()`

The TMcraft plugin can use this function to verify whether it comes with the ownership of the end-button event. If yes, this TMcraft plugin is the only one that can receive the end-button event signal.

- `bool IsEndButtonBroadcastMode()`

The TMcraft plugin can call this function to check if the end-button event is currently in broadcast mode. If it is, all TMcraft plugins can receive the event signal. Otherwise, one of the TMcraft plugins has ownership, meaning other plugins will not receive the signal from the event.

- `uint ReleaseEndButtonEventOwnership()`

The TMcraft plugin can call this function to release the button event ownership.

- `uint SetEndButtonEventOwnership()`

The TMcraft plugin can call this function to get the end-button event ownership.

- `event EndButtonClickEvent`

An event type denotes the click event that occurred on the buttons of the End Module. A function can be linked to this event and activated once the event is triggered.



### IMPORTANT:

Note the `RobotStatusProvider.EndButtonClickEvent` has been deprecated since version 1.20 and replaced by `EndButtonEventProvider.EndButtonClickEvent`.

After initializing the TMcraft plugin, the program should declare and define an `EndButtonClickEvent` by assigning a function to this event. The assigned function will execute whenever the `EndButtonClickEvent` is triggered. To obtain exclusive access to the button event, the program can use `IsEndButtonBroadcastMode()` and `HasEndButtonEventOwnership()` to check the current event mode and whether the plugin holds ownership. If not, the program should call `SetEndButtonEventOwnership()` to gain ownership of the button event. Once the event handling is complete, the program should call `ReleaseEndButtonEventOwnership()` to return the button event to broadcast mode, allowing other components to respond.

### 3. Sample Code

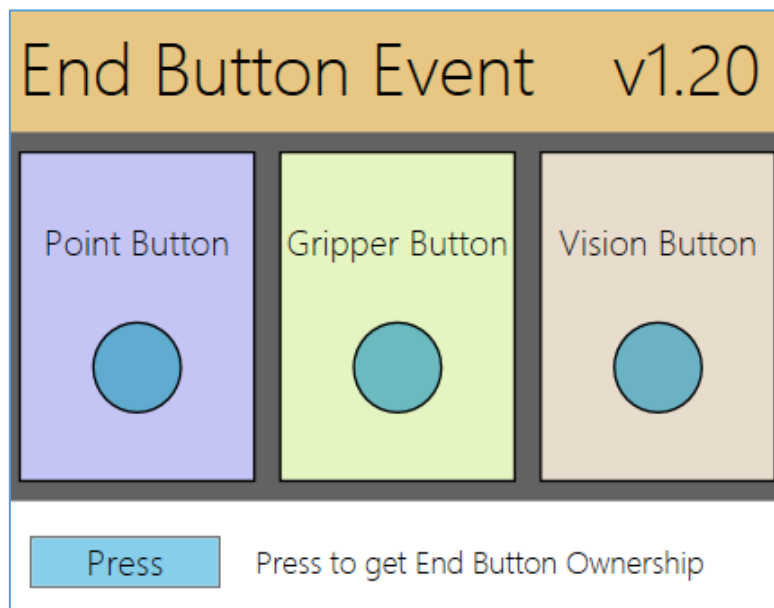
Consider an example of two TMcraft plugins, a Node and a Toolbar. Once the end-button event is in broadcast mode, both plugins receive the event signal. On the other hand, if one of the plugins has obtained ownership, the other plugin will not receive the signal.

This section will talk about the essential parts of the toolbar sample code (node sample code is similar). For the complete source code, please refer to General Examples: [Node][Toolbar] EndButton Event Demo in the TMcraft Development Kit.

#### 3.1 MainWindow.xaml

MainWindow.xaml defines the UI, which includes the following elements:

- Bulb\_PointButton: turns green when the Point Button is pressed.
- Bulb\_GripperButton: turns green when the Gripper Button is pressed.
- Bulb\_VisionButton: turns green when the Vision Button is pressed.
- Btn\_Ownership: click this button to get or release the ownership of the end-button event.



#### 3.2 MainWindow.xaml.cs

MainWindow.xaml.cs defines the program of the TMcraft Toolbar. Here are some of the essential parts of the program:

- Declaration of the necessary global objects.

```
public partial class MainPage : UserControl, ITMcraftToolbarEntry
{
    TMcraftToolbarAPI toolbarUI;
    bool getOwnership;
    SolidColorBrush Color_Disable = new SolidColorBrush();
    SolidColorBrush Color_Enable = new SolidColorBrush();
}
```

- Definition of the ITMcraftToolbarEntry member function InitializeToolbar. Most importantly, it declares an `EndButtonEventProvider.EndButtonClickEvent` and increments with the function `EndButtonClickEventReaction`.

```
public void InitializeToolbar(TMcraftToolbarAPI _toolbarUI)
{
    toolbarUI = _toolbarUI;
    if (toolbarUI == null || toolbarUI.EndButtonEventProvider == null)
    {
        MessageBox.Show("No Connection with TMflow...");
    }
    else
    {
        toolbarUI.EndButtonEventProvider.EndButtonClickEvent +=
            EndButtonClickEventReaction;
    }
}
```

- Definition of the event function `EndButtonClickEventReaction`. It uses a switch to check the event data and sets the respective bulb to the right color.



```

private void EndButtonClickEventReaction(RobotEventType type, object data)
{
    Dispatcher.Invoke(
        DispatcherPriority.Background,
        new Action(delegate ()
        {
            switch (type.ToString() + " " + data.ToString())
            {
                case "EndButtonPointChanged True":
                    Bulb_PointButton.Fill = Color_Enable;
                    break;
                case "EndButtonPointChanged False":
                    Bulb_PointButton.Fill = Color_Disable;
                    break;

                case "EndButtonGripperChanged True":
                    Bulb_GripperButton.Fill = Color_Enable;
                    break;
                case "EndButtonGripperChanged False":
                    Bulb_GripperButton.Fill = Color_Disable;
                    break;

                case "EndButtonVisionChanged True":
                    Bulb_VisionButton.Fill = Color_Enable;
                    break;
                case "EndButtonVisionChanged False":
                    Bulb_VisionButton.Fill = Color_Disable;
                    break;

                default:
                    MessageBox.Show("Invalid RobotEvent: " + type.ToString() + " " +
                    data.ToString());
                    break;
            }
        }));
}

```



#### IMPORTANT:

- Event functions need a dispatcher to ensure that invocations are on the thread that created the UI object. It is crucial because WPF UI objects are not thread-safe. Making changes from a different thread can lead to issues like exceptions and crashes.
- The dispatcher queues up event functions and executes them on the UI thread, guaranteeing the safe and orderly execution of all event functions.

- Definition of the click event function of the button Btn\_Ownership. To obtain ownership, the function first calls `IsEndButtonBroadcastMode()` and `HasEndButtonEventOwnership()` to check if the end-button event is in broadcast mode and no one secures the ownership; if so, calls `SetEndButtonEventOwnership()` to obtain the ownership.

```

if (!getOwnership)
{
    Btn_Ownership.IsEnabled = false;

    if (!toolbarUI.EndButtonEventProvider.IsEndButtonBroadcastMode() && !
        toolbarUI.EndButtonEventProvider.HasEndButtonEventOwnership())
    {
        MessageBox.Show("End Button Event Ownership has been obtained by someone else...");
        Btn_Ownership.IsEnabled = true;
        return;
    }

    uint result = toolbarUI.EndButtonEventProvider.SetEndButtonEventOwnership();
    if (result == 0)
    {
        getOwnership = true;
        Label_Ownership.Content = "Press To Release End Button Ownership";
        Btn_Ownership.Background = Brushes.GreenYellow;
        Btn_Ownership.IsEnabled = true;
        return;
    }
    else
    {
        string msg = string.Empty;
        toolbarUI.GetErrMsg(result, out msg);
        MessageBox.Show(msg);
        Btn_Ownership.IsEnabled = true;
        return;
    }
}

```

To release the ownership, call **ReleaseEndButtonEventOwnership()**.

```

else
{
    Btn_Ownership.IsEnabled = false;

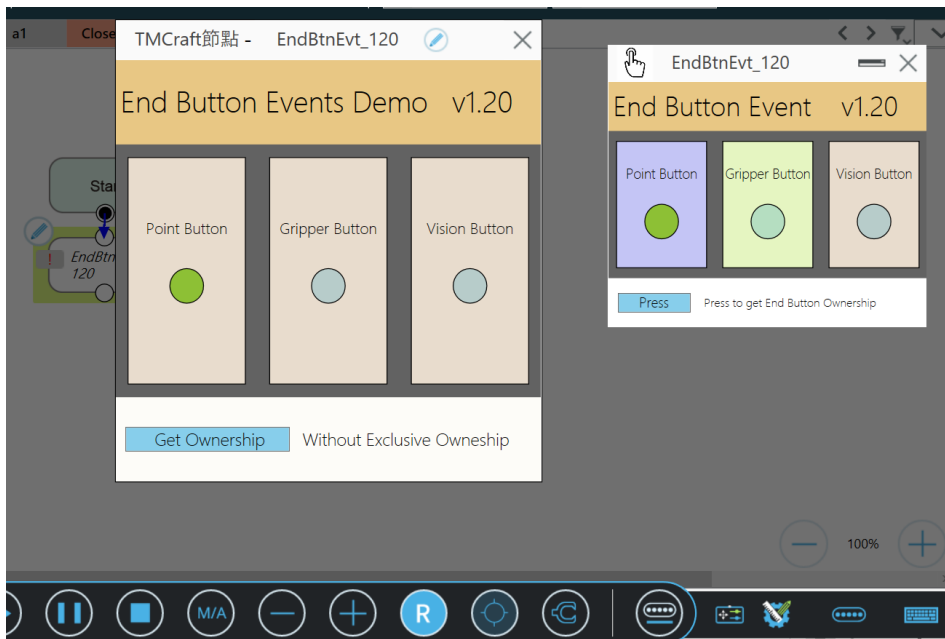
    uint result =
        toolbarUI.EndButtonEventProvider.ReleaseEndButtonEventOwnership();
    if (result == 0)
    {
        getOwnership = false;
        Label_Ownership.Content = "Press To Obtain End Button Ownership";
        Btn_Ownership.Background = Brushes.SkyBlue;
        Btn_Ownership.IsEnabled = true;
        return;
    }
    else
    {
        string msg = string.Empty;
        toolbarUI.GetErrMsg(result, out msg);
        MessageBox.Show(msg);
        Btn_Ownership.IsEnabled = true;
        return;
    }
}

```

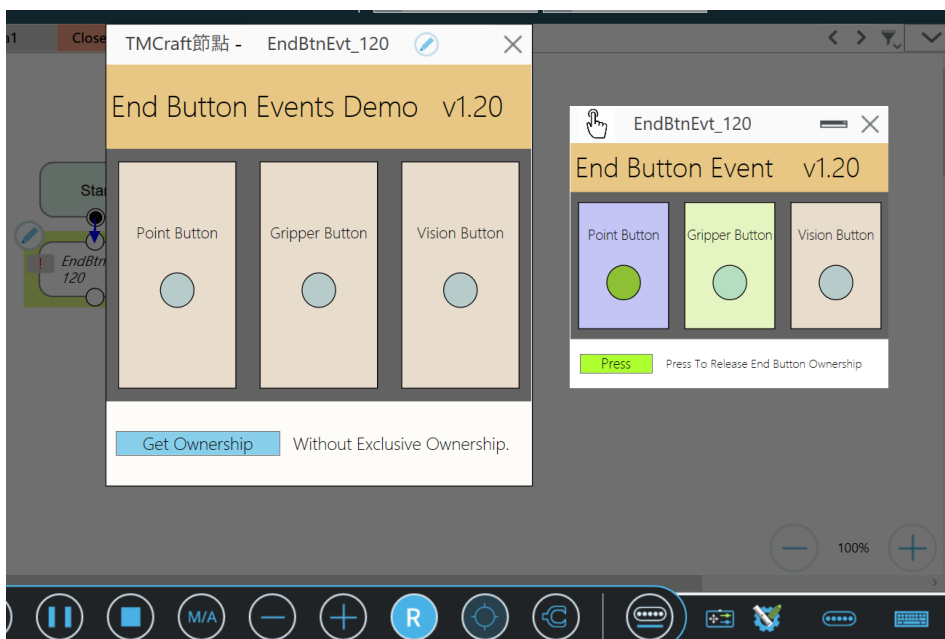
### 3.3 Result

Package the plugins and import them into TMflow. Initially, the end-button event was in broadcast

mode, allowing both plugins to receive the signal.



Press the ownership button to obtain the event ownership so that the other plugin cannot receive the signal.



Since the plugin checks the status through the API function `HasEndButtonEventOwnership()`, a message will appear if ownership has already been obtained.

