



# TMcraft



# Shell Tutorial

# Jogging the Robot

Original Instructions

Document version: 1.0  
Release date: 2024-02-01

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.....	4
3. Sample Code .....	5
3.1 MainWindow.xaml .....	5
3.2 MainWindow.xaml.cs.....	6

## Revision History

Revision	Date	Description
1.0	2024-02-01	Original release

## 1. Introduction

This document explains how to perform jogging on the TMcraft Shell using TMcraft API functions. Readers should have the following prerequisites:

- Basic knowledge on programming C# and WPF
- Having read *TMcraft Shell Tutorial: Basic Development*
- Having read *TMcraft Shell API Function Manual*

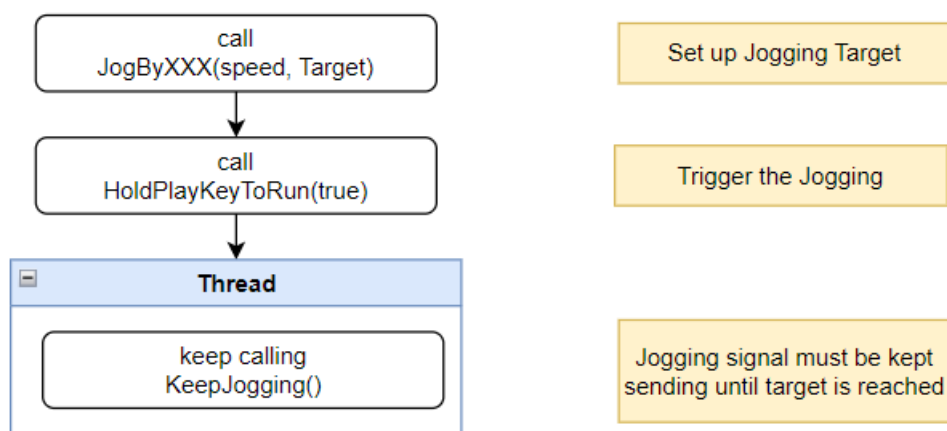
Based on TMcraft.dll version 1.16, this tutorial.



### IMPORTANT:

If the TMcraft Shell utilizes any RobotJogProvider functions for motion control, the developer is responsible for maintaining a single point of control as per ISO 10218-1.

## 2. Concept



TMcraft Shell API comes with the RobotJogProvider, a class consists of functions associated to robot jogging including:

- JogByXXX (speed, Target)  
These functions configure the required jogging.
- HoldPlayKeyToRun(bool)  
This function mimics manipulating the robot stick play key to start or stop the jogging.

- **KeepJogging()**  
This function must be continuously called to ensure jogging and serves as a safeguard. In case of an unexpected program crash, the KeepJogging signal will be terminated, halting the jogging.
- **StopJog()**  
This function terminates all Jogging actions.

To initiate a jogging motion, the program should use JogByXXX to configure the jogging settings. Next, it should call HoldPlayToRun(true) while simultaneously starting a thread to continuously call KeepJogging. When all three conditions are met, the robot will begin jogging. To stop jogging, terminate the KeepJogging thread and call either HoldPlayToRun(false) or StopJog().

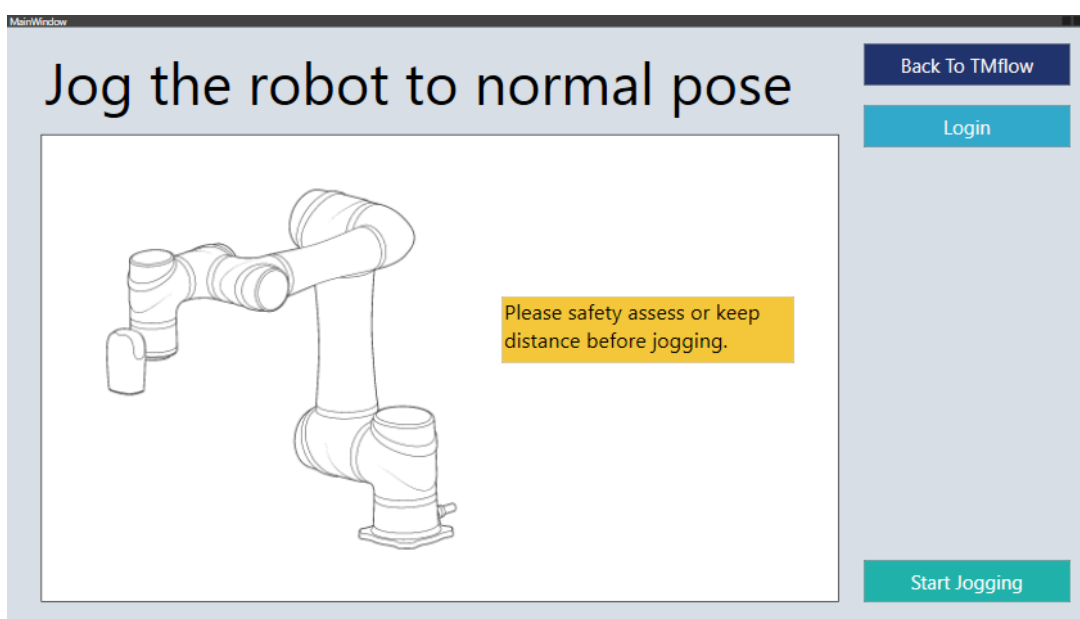
### 3. Sample Code

This section will talk about the essential parts of the JogByShell source code. For the complete source code, please refer to TMcraft Development Kit\Shell\Samples\JogByShell\.

#### 3.1 MainWindow.xaml

MainWindow.xaml defines the UI which includes:

- **Btn\_Back**: click to go back to TMflow
- **Btn\_Login**: click to login and get control
- **Btn\_Jog**: click to start or stop jogging



### 3.2 MainWindow.xaml.cs

The *TMcraft Shell Tutorial* has provided explanations for `Btn_Back_Click` and `Btn_Login_Click`. Now, the below discusses `Btn_Jog_Click`.

- Declaration of the necessary global variables and objects

```
public partial class MainWindow : Window
{
    TMcraftShellAPI ShellUI;
    bool JogStatus = false;
    Thread th_KeepJog;
```

- Definition of the thread function `_KeepJogging`. If `JogStatus` is true, the thread will keep calling `RobotJogProvider.KeepJogging()`.

```
private void _KeepJogging()
{
    while (JogStatus)
    {
        if (ShellUI == null || ShellUI.RobotStatusProvider == null)
        {
            MessageBox.Show("TMflow not connected...");
            return;
        }

        ShellUI.RobotJogProvider.KeepJogging();
        Thread.Sleep(300); //100 - 500 ms
    }
}
```

- Definition of `Btn_Jog_Click`.

1. Check if the `ShellUI` connects to `TMflow` or not.

```
if (ShellUI == null || ShellUI.RobotStatusProvider == null)
{
    MessageBox.Show("TMflow not connected...");
    return;
}
```

2. Verify whether the button is currently jogging or not. If it returns false, it indicates that the robot is not jogging. In this case, call `RobotJogProvider.JogByJoint(speed, target)` and `RobotJogProvider.HoldPlayKeyToRun(true)`. Then, set `JogStatus` to true and initiate the `KeepJogging` thread. Finally, change the button's color to red and update the content

of Btn\_Jog to “Stop Jogging” .

```
if (!JogStatus)
{
    float[] TargetAngle = { 0, 0, 90, 0, 90, 0 };
    ShellUI.RobotJogProvider.JogByJoint(3f, TargetAngle);
    ShellUI.RobotJogProvider.HoldPlayKeyToRun(true);
    JogStatus = true;

    th_KeepJog = new Thread(_KeepJogging);
    th_KeepJog.Start();

    Btn_Jog.Content = "Stop Jogging";
    Btn_Jog.Background = Brushes.PaleVioletRed;
}
```

3. If it returns true, it indicates that the robot is jogging. In this case, call `RobotJogProvider.HoldPlayKeyToRun(false)` to halt the jogging. Then, set JogStatus to false and end the KeepJogging thread. Finally, change the button's color to green and update the content of Btn\_Jog to “Start Jogging” .

```
if (!JogStatus)
else
{
    ShellUI.RobotJogProvider.HoldPlayKeyToRun(false);
    JogStatus = false;

    th_KeepJog.Join();
    ShellUI.RobotJogProvider.StopJog();

    Btn_Jog.Content = "Start Jogging";
    Btn_Jog.Background = Brushes.LightSeaGreen;
}
```



