



# TMcraft Toolbar Tutorial Basic Development

Original Instructions

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. Prerequisites.....	6
3. TMcraft API Program Structure .....	7
3.1 Brief Explanation of TMcraft API.....	7
3.2 Program structure of a TMcraft Toolbar.....	8
4. Start Programming a TMcraft Toolbar .....	10
4.1 System Design .....	10
4.2 Create a WPF Application Project.....	11
4.3 Source Code.....	13
4.4 Compile and Test UI.....	15
5. Build TMcraft Toolbar Package.....	18
6. Installation Code and Checksum .....	22
7. Use TMcraft Toolbar on TMflow .....	23
8. Dos & Don' ts.....	25

## Revision History

Revision	Date	Description
1.0	2024-02-01	Original release

# 1. Introduction

TMcraft Toolbar is a C#/WPF based plugin GUI, which can be used anytime and anywhere within TMflow, for example:

- (1) As a control interface, such as function test, parameter settings, calibration, etc., of the device.
- (2) As an assistant during application development, such as enhancing efficiency; for instance, quickly setting up Freebot for various requirements."

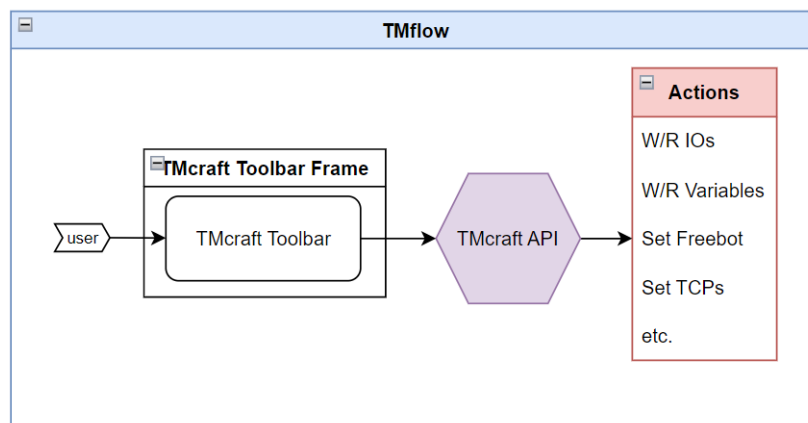


Figure 1: The Architecture of TMcraft Toolbar

To determine whether the TMcraft Toolbar is the right fit to realize concepts of users, consider these questions:

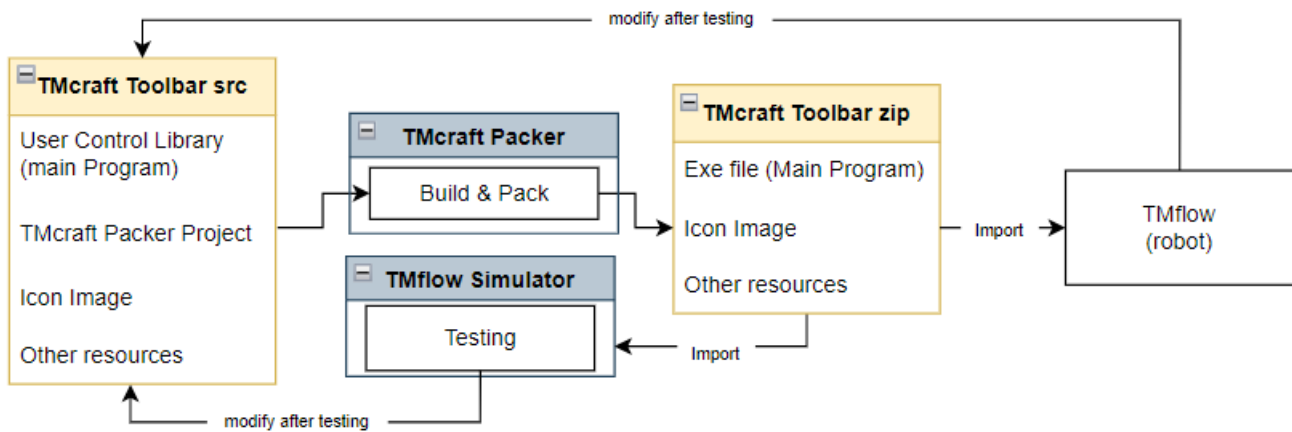
## Are users familiar with script programming?

As built with C#, developers need C# skills to develop the TMcraft Toolbar.

## Is a control interface needed for the device when the robot is not running a project?

For instance, if the robot encounters a CAT. 1 or CAT. 2 Stop error, users can release the gripper and save the workpiece using the TMcraft Toolbar for the device.

To understand how to develop a TMcraft Toolbar, please check the following diagram:



Developers create the TMcraft Toolbar UI as a User Control Library and use TMcraft Packer to build and package it into a zip file. They can then test it on TMflow Simulator or directly on a robot. Following multiple iterations of modifications and testing, they can release the TMcraft Toolbar.

This tutorial organizes the content based on TMcraft.dll version 1.16.1400 from TMflow 2.16.

- Section 2: prerequisites for developing a TMcraft Toolbar
- Section 3: the concept and features of the TMcraft API and TMcraft Toolbar Program
- Section 4: the process of how to develop a simple TMcraft Toolbar
- Section 5: how to complete building a TMcraft Toolbar Package
- Section 6: the concept of Unique Identifier
- Section 7: how to import and use the TMcraft Toolbar on TMflow
- Section 8: the dos & don'ts

## 2. Prerequisites

To develop TMcraft Toolbars, developers should be capable of:

- Having Basic knowledge of programming with C#
- Using WPF to build UI.

Software requirements,

- An integrated development environment for C# and WPF (such as Microsoft Visual Studio 2022)
- TMflow 2.16 or above
- .NET 6.0 and .NET SDK
- TMcraft Packer for building and packaging the TMcraft item

### 3. TMcraft API Program Structure

#### 3.1 Brief Explanation of TMcraft API

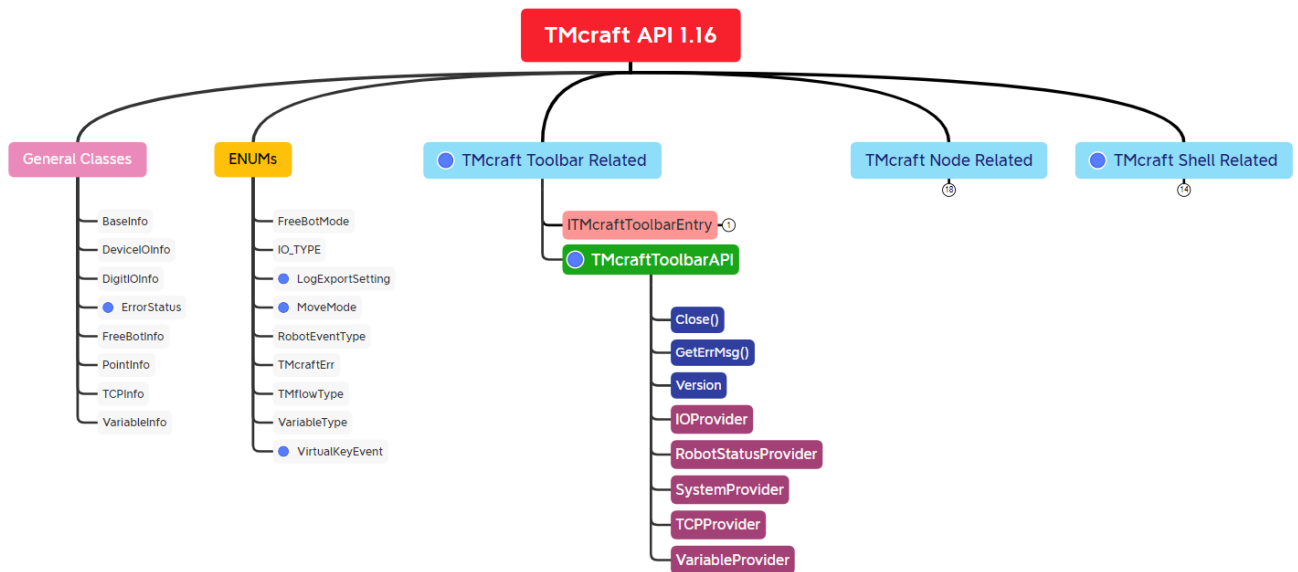


Figure 2: TMcraft API architecture

TMcraft API is an interface that connects TMcraft Toolbar with TMflow to get all sorts of robot information and request TMflow to take different actions. In addition, this API is a Dynamic Link Library file (DLL); developers should add it to program dependencies so that the program can use the functions within the API.

TMcraft.dll consists of classes, enums, and functions for every TMcraft items. Here are those most related to TMcraft Toolbar:

- **ITMcraftToolbarEntry** is an interface that defines the program is a TMcraft Toolbar. Users must define the member function, **InitializeToolbar()**, in order to make it work.
- **TMcraftToolbarAPI** has all functions and provider classes to build TMcraft Toolbar. Note that each TMcraft item comes with its own set of provider classes. Even if some share the same name, their member functions vary.
- **Providers** are a set of classes with functions associated with different interactions with TMflow frequently used among the Programs.
- **General classes** are a collection of classes, such as DeviceIOInfo, TCPInfo, and ErrorStatus, used by different types of Provider functions.



- **Enum**, or enumerators, are value types defined by a set of named constants of the underlying integral numeric type used by provider functions.

Refer the table below to overview the API capabilities across various TMcraft plugins:

capabilities \ Plugins	Node	Shell	Toolbar
Base (Add\Edit\Delete)	0		
Point (Add\Edit\Delete)	0		
Tool (Add\Edit\Delete)	0	0	0
Digital IO (Read/Write)	0	0	0
Analog IO (Read/Write)	0	0	0
Variables (New/Edit)	0	0	0
Vision Job (Add\Open\Delete)	0		
Jog the robot	0	0	
Freebot (Set/Get)	0	0	0
End Button Event	0	0	0
Get Current Language	0	0	0
Get TMflow Type	0	0	0
TMscript on flow project (Read/Write)	0		
Login/Logout/Get Control		0	
script Project (Add\Edit\Delete)		0	
Robot status (Error, Run, etc.)		0	0
Error Event		0	
Virtual Robot Stick		0	
Export/Import		0	
Variables Runtime Value (Read/Write)		0	Read only

### 3.2 Program structure of a TMcraft Toolbar

Refer to the sample code below for the program structure.

```

using TMcraft;

namespace TMcraftSample
{
    public partial class MainPage : UserControl, ITMcraftToolbarEntry
    {
        TMcraftToolbarAPI ToolbarUI;

        public MainPage()
        {
            InitializeComponent();
        }

        public void InitializeToolbar(TMcraftToolbarAPI _toolbarUI)
        {
            ToolbarUI = _toolbarUI; //connect TMflow
        }

        private void UserControl_Loaded(object sender, RoutedEventArgs e) {}
    }
}

```

To create the foundation of a TMcraft Toolbar Program, please remind the key points below:

1. Include TMcraft.dll as a reference. Remember to add the namespace, `using TMcraft`, onto the program.
2. Declare a global object based on the class `TMcraftToolbarAPI`.
3. Implement the interface `ITMcraftToolbarEntry` and define its member function `InitializeToolbar` to connect the TMcraft Toolbar with TMflow.

The rest of the program should be all sorts of event functions that can interact with TMflow through TMcraft functions.

## 4. Start Programming a TMcraft Toolbar

Developers are favored to follow the steps to develop a TMcraft Toolbar.

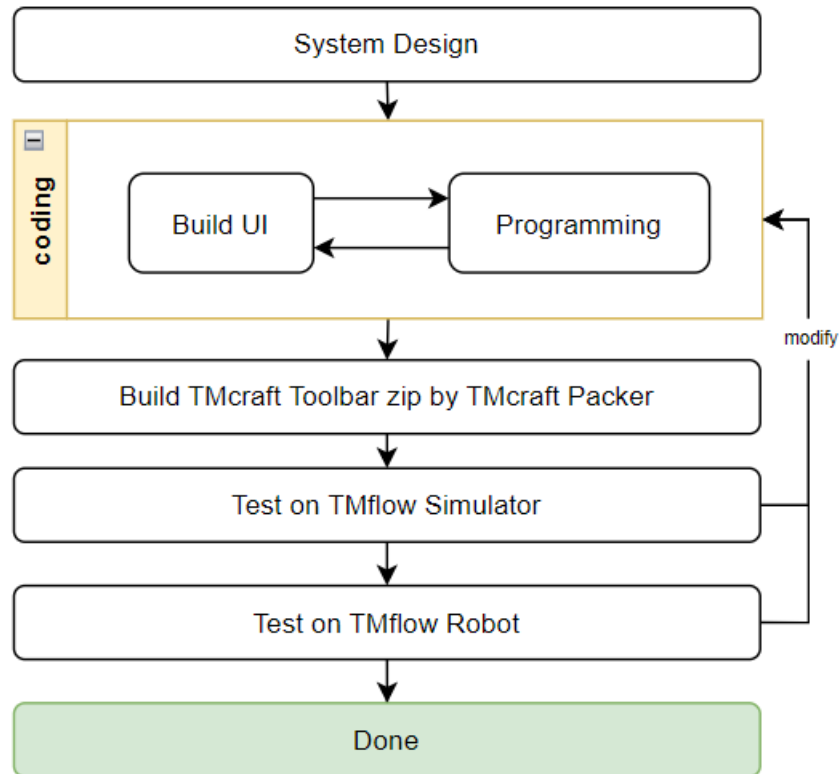


Figure 3: The Recommended Procedure of Developing a TMcraft Toolbar

From section 4 to section 6, users will learn how to approach these steps with a simple TMcraft Toolbar, *HelloWorldToolbar*.

### 4.1 System Design

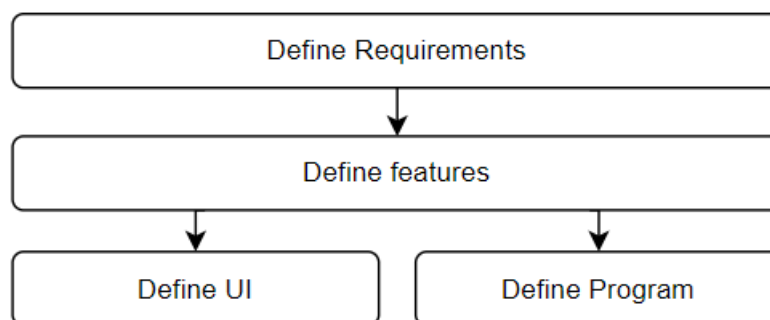


Figure 4: Recommended procedure of System Design

The outline of the application:

- At startup, the Toolbar GUI displays Digital Output channel 0's status from the Control Box and the

End Module on the associated button, with green indicating ON and white indicating OFF.

- Clicking the button allows users to turn the corresponding Digital Output channel 0 on or off.

Refer figures below for the UI and the program architecture.

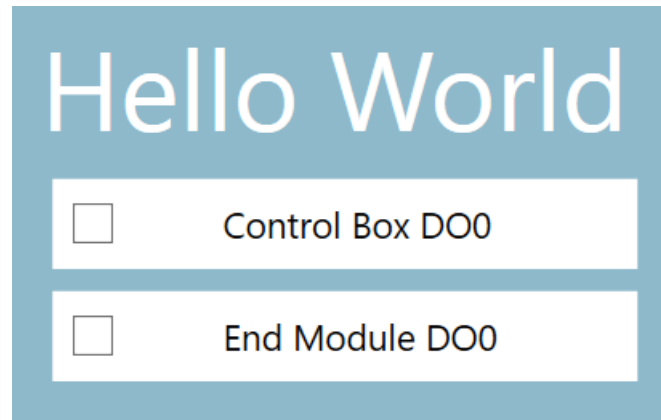


Figure 5: HelloWorldToolbar UI

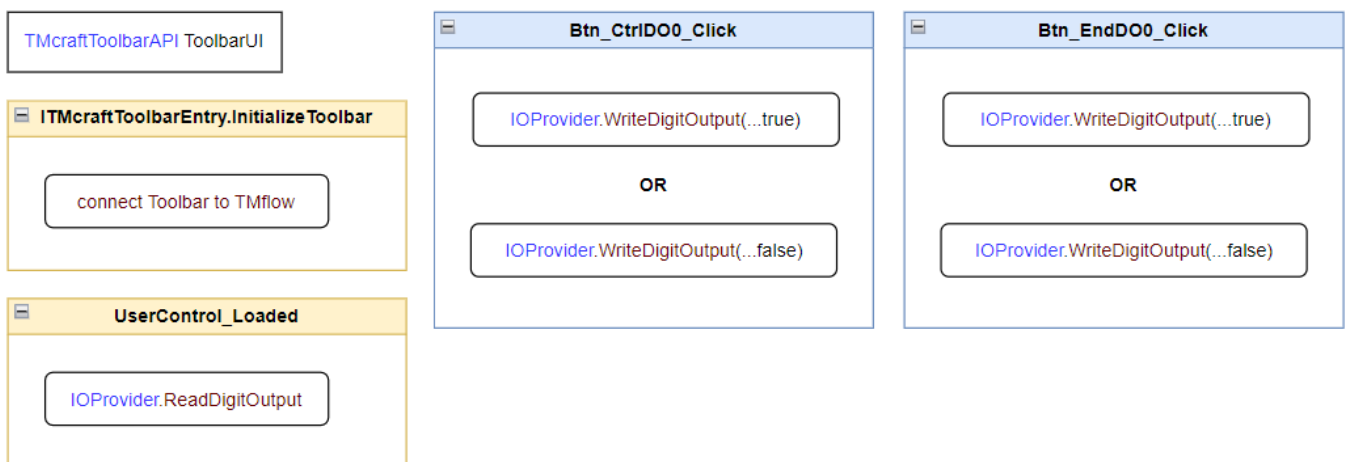


Figure 6: Program architecture of HelloWorldToolbar

## 4.2 Create a WPF Application Project

This section is written based on the usage of Microsoft Visual Studio 2022. First, users can create a new project with the *WPF User Control Library* template.

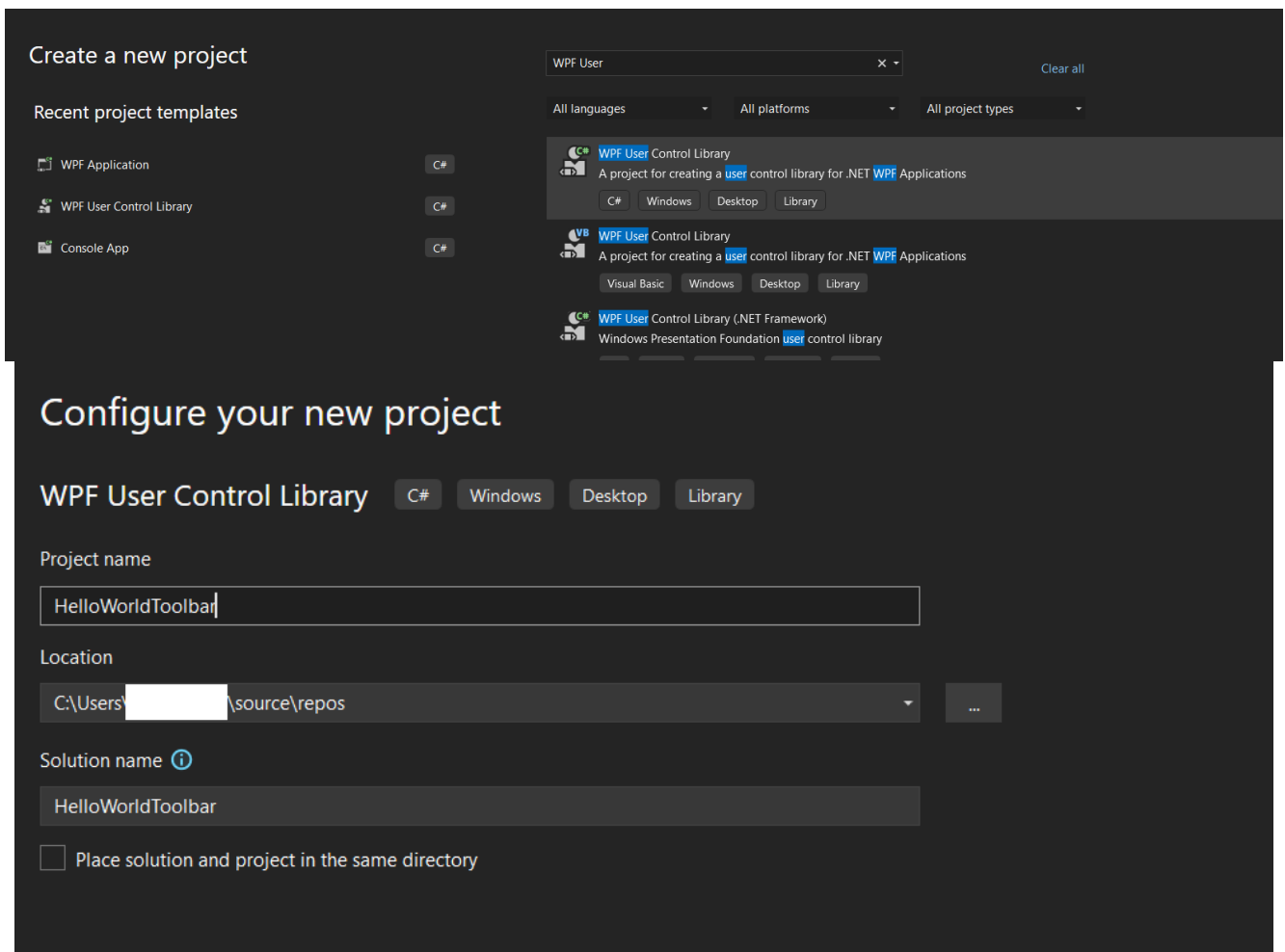


Figure 7: Create a New Project with the WPF User Control Library Template

Remember to set up with .NET 6.0 as the framework of the project.

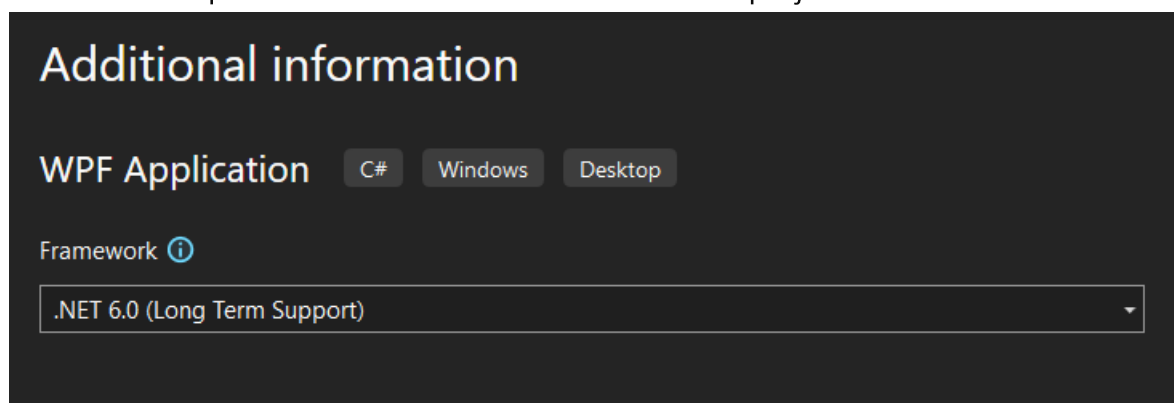


Figure 8: .NET 6.0 Setup as the Framework

Once opened the project in the editor, add the TMcraft.dll as the reference.

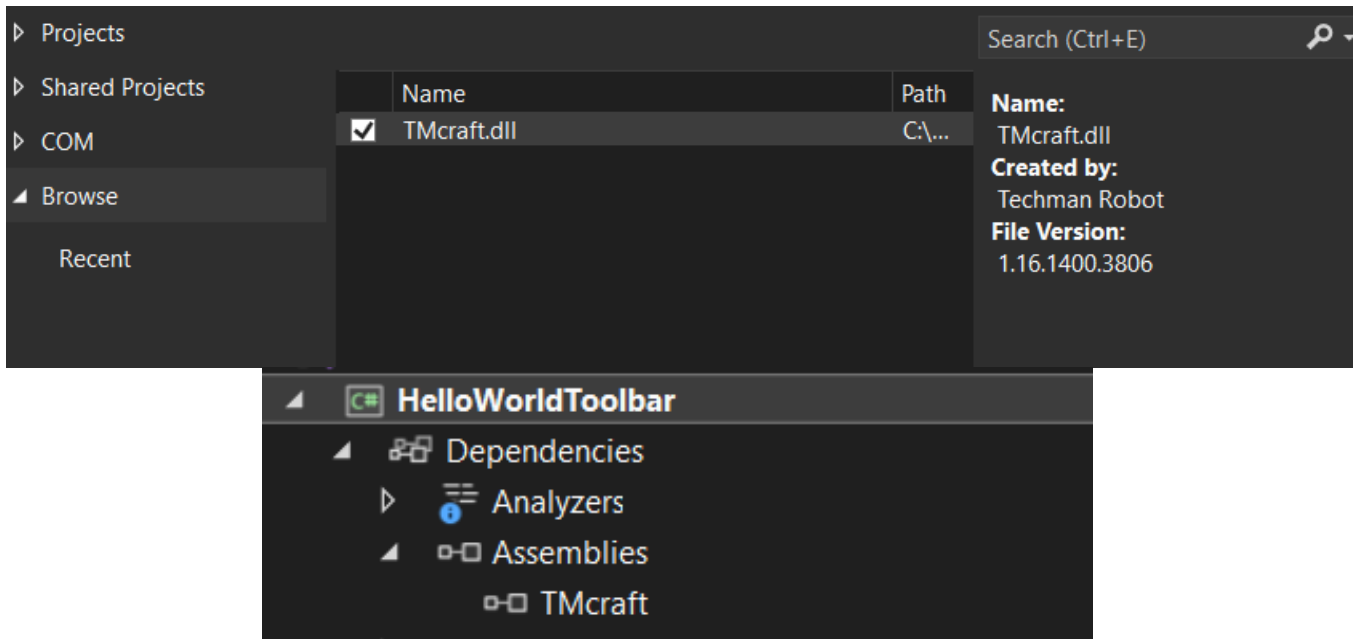


Figure 9: Add TMcraft.dll as a Reference

### 4.3 Source Code

This section focuses on the key aspects of the HelloWorldToolbar source code. For the complete source code, please refer to TMcraft Development Kit\Toolbar\Samples\ HelloWorldToolbar\

#### 4.3.1.1 MainPage.xaml

MainPage.xaml defines the UI, here is a few points necessary:

- `x:Class="HelloWorldToolbar.MainPage"`, used as one of the required parameter to build the executable file of the TMcraft Toolbar. Refer to section 5 for more details.
- `Loaded="UserControl_Loaded"`: defines the actions taken during UI initialization.
- Add click events on each buttons including `Btn_CtrlDO0_Click`, `Btn_EndDO0_Click`.

#### 4.3.1.2 MainPage.xaml.cs

MainPage.xaml.cs defines the program of the TMcraft Toolbar; here are a few points necessary:

- First, add TMcarft.dll onto the program reference.

```
using System.Windows.Shapes;
using TMcraft;

namespace HelloWorldToolbar
{
    /// <summary>
```

- Declare a global `TMcraftToolbarAPI` object, `ToolbarUI`, and global Boolean variables to represent the status of the target digital output.

```
public partial class MainPage : UserControl, ITMcraftToolbarEntry
{
    TMcraftToolbarAPI ToolbarUI;
    bool status_CD00 = false;
    bool status_ED00 = false;
}
```

- Add the interface `ITMcraftToolbarEntry`, and implement the relevant member function, `InitializeToolbar()`

```
0 references
public void InitializeToolbar(TMcraftToolbarAPI _toolbarUI)
{
    ToolbarUI = _toolbarUI;
}
```

- Define the `UserControl_Loaded` function to get the status of Digital Output channel 0 from the Control Box and End Module by calling `IOPProvider.ReadDigitOutput` and then update the associated buttons.

```
ToolbarUI.IOPProvider.ReadDigitOutput(IO_TYPE.CONTROL_BOX, 0, 0, out status_CD00);
if (status_CD00)
{
    Btn_CtrlD00.Background = Brushes.GreenYellow;
}
else
{
    Btn_CtrlD00.Background = Brushes.White;
}
```

- Define the click events, `Btn_CtrlD00_Click` and `Btn_EndD00_Click`. These functions change the Digital Output value by calling `IOPProvider.WriteDigitOutput`.

```
if (status_CD00)
{
    ToolbarUI.IOPProvider.WriteDigitOutput(IO_TYPE.CONTROL_BOX, 0, 0, false);
    status_CD00 = false;
    Btn_CtrlD00.Background = Brushes.White;
}
else
{
    ToolbarUI.IOPProvider.WriteDigitOutput(IO_TYPE.CONTROL_BOX, 0, 0, true);
    status_CD00 = true;
    Btn_CtrlD00.Background = Brushes.GreenYellow;
}
```

#### Note

#### NOTE:

There are some instructions recommended when using the TMcraft API functions:

1. Check if the `TMcraftToolbarAPI` object is null or not
2. Use a `uint` object to get the result of the API function (replied by TMflow)

3. Check if the result is 0 or not; if not, call `GetErrMsg()` to get the corresponding error message
4. Be aware that errors might not only originate from TMflow but also from the API itself. These errors are represented as `TMcraftErr` objects and can be retrieved by calling `GetErrMsg()`.

#### 4.4 Compile and Test UI

Once source code is ready, compile the code and generate the DLL file. Be reminded to double-check items below beforehand.

1. Right-click on the DLL Project on the Solution Explorer to open the Properties page.
2. On the Properties\Application page, check the parameters. The most important is to make sure the **Output type** is **Class Library** and **Target framework** is **.NET 6.0**.

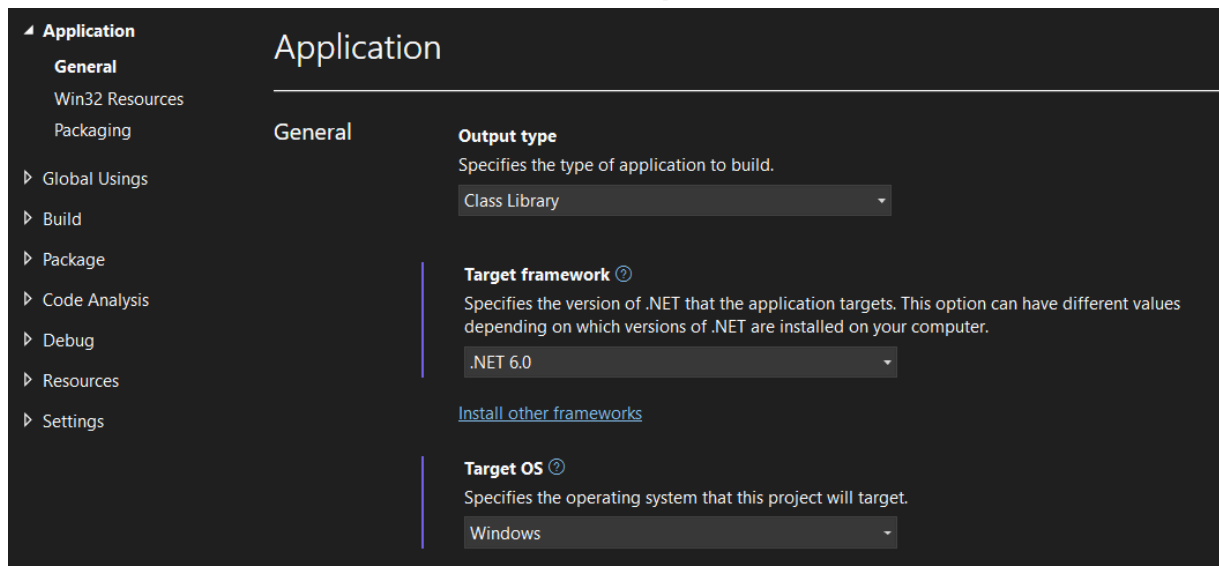


Figure 10: Set the Application Property before Building the UserControl Library

Before further constructing the TMcraft Toolbar Package with the UserControl DLL, it has much to recommend that users test the UI functioning first.

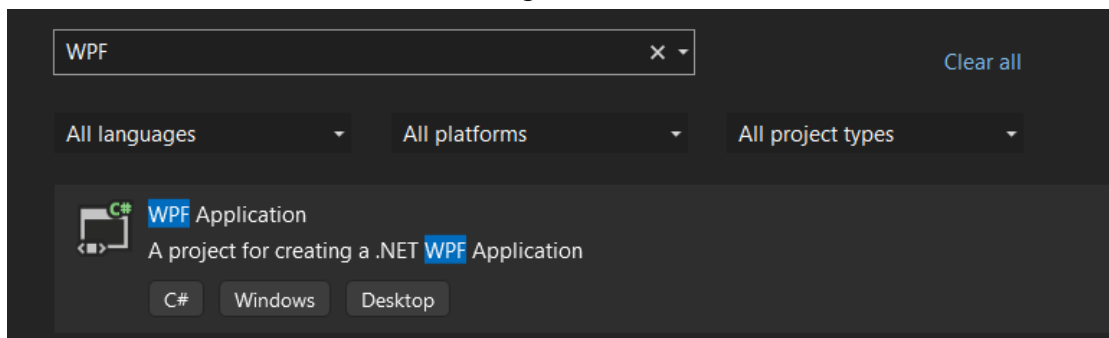


Figure 11: Create a WPF Application for UI Testing



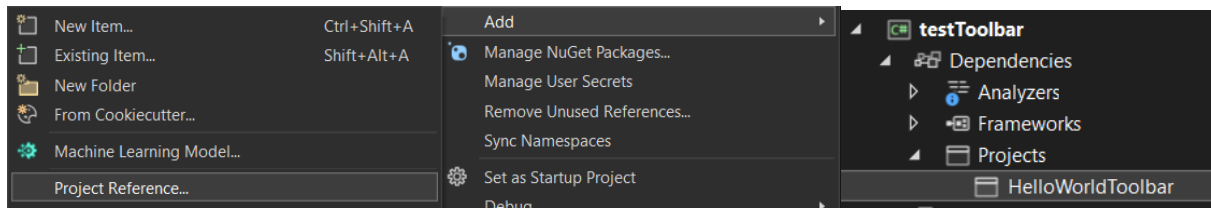


Figure 12: Add the HelloWorldToolbar User Control as Reference

The application source code is simple, i.e., to define a UserControl object with HelloWorldToolbar.MainPage (namespace of the Toolbar UI) and add it to the Grid. (Remember to name it first on the xaml file or the Properties page of the Grid)

```
using HelloWorldToolbar;

namespace testToolbar
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    2 references
    public partial class MainWindow : Window
    {
        0 references
        public MainWindow()
        {
            InitializeComponent();
            UserControl toolbarUI = new MainPage();
            gd_main.Children.Clear();
            gd_main.Children.Add(toolbarUI);
        }
    }
}
```

Figure 13: Source code of the toolbar testing application

Before compiling the code, remember to set the testing application as the Startup Project. After that, run the program and test Toolbar UI.

Set the testing application as the Startup Project before compiling the code. Then, run the program to test the Toolbar UI.

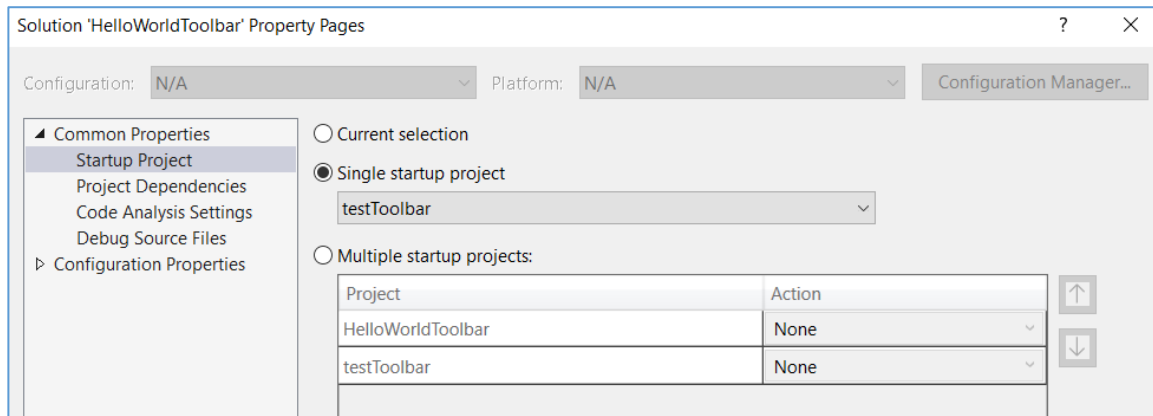


Figure 14: Set up the Startup Project

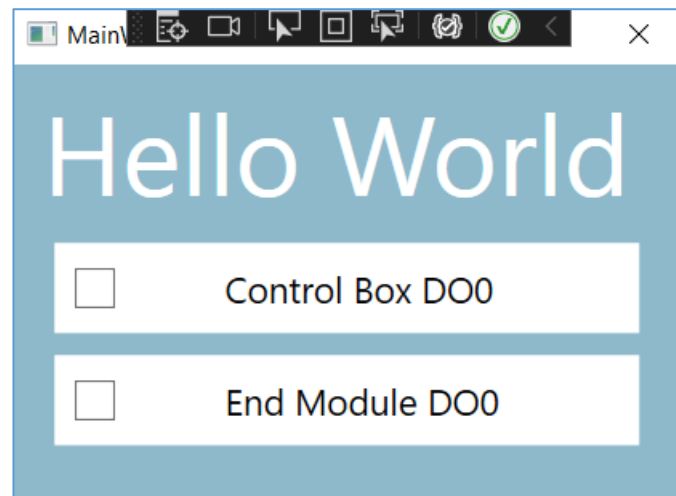


Figure 15: Test the Toolbar UI

## 5. Build TMcraft Toolbar Package

The previous section described how to build the TMcraft Toolbar Program. Before using it on TMflow, it must be packaged using the TMcraft Packer.

First, prepare a source folder with the following items.

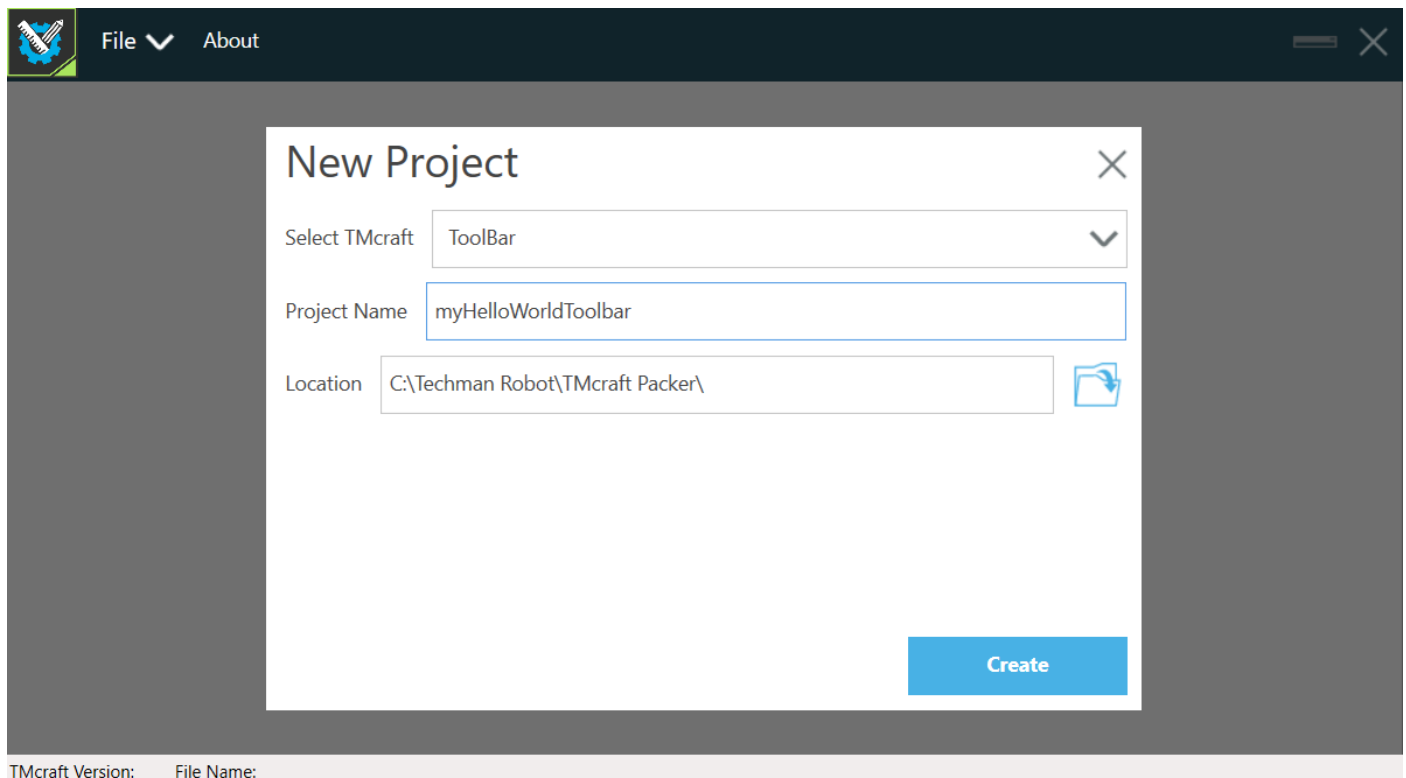
- UserControl Library file (dll) of the Toolbar UI
- All files from the TMcraft API (1.16 or above) folder from the development kit
- The Icon Image
- Other resource files, such as:
  - Other reference files used by the Program
  - Files used by the Toolbar, such as media, documents, etc.

### Note

#### NOTE:

Suggest putting all files from the ..\bin\Debug folder into the source folder.

Next, open TMcraft Packer (1.12 or above) and create a project. Select **Toolbar** as the target TMcraft item and complete the Project Settings. The system saves the project as a .tmcraft file at the specified file path in the **Location** field.



TMcraft Version: File Name:

Figure 16: Create a TMcraft Packaging Project

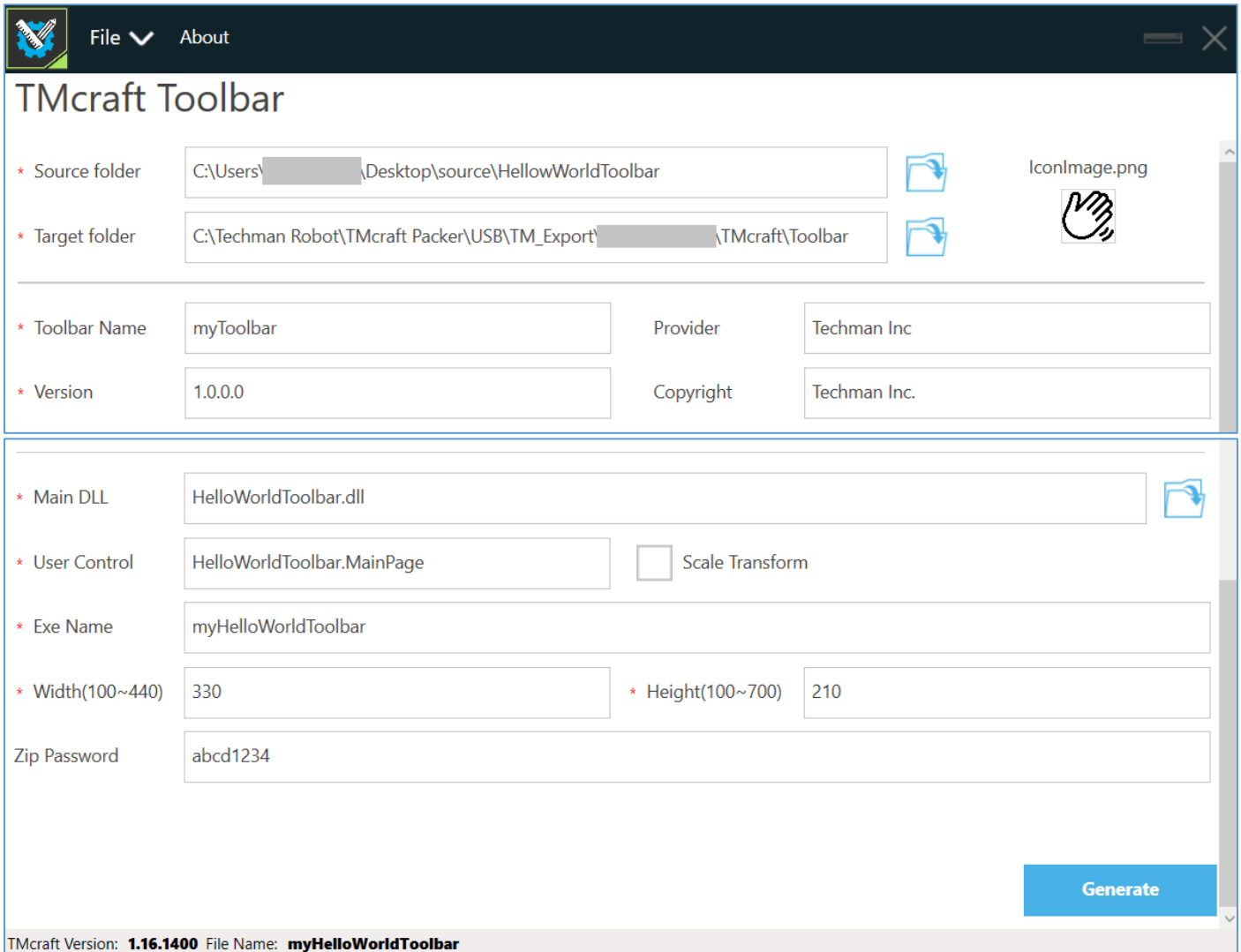
**IMPORTANT:**

Ensure that TMcraft Packer is running in the environment with .NET 6.0 installed.

After opening the project, users can see a form of parameters required to build and package the TMcraft Toolbar. Refer to the following for explanations.

Parameters	Description	Requisite
<b>Source folder</b>	Path of the source folder	<input type="radio"/>
<b>Target folder</b>	Path of the target folder. For any new projects, the default target folder is <code>..\TMcraft Packer\USB\TM_Export\[PC Name]\TMcraft\Toolbar</code>	<input type="radio"/>
<b>Toolbar Name</b>	Name of the Toolbar, which is shown on Toolbar top bar	<input type="radio"/>
<b>Provider</b>	Name of the developer or the company providing this Toolbar, which is shown on the TMcraft Management Page	
<b>Version</b>	The version of the TMcraft Toolbar, which is registered onto the exe file and also shown on TMcraft Management Page	<input type="radio"/>
<b>Copyright</b>	Copyright of the TMcraft Toolbar, which is registered onto the exe file	
<b>Main DLL</b>	File name of the User Control Library of the TMcraft Toolbar (as in Section 4.4)	<input type="radio"/>
<b>User Control</b>	The startup user control from the Main DLL. Since there might be several user controls within the User Control Library, it is necessary to define which one is used as the Main User Control. The format of this parameter should be <code>[Namespace].[UserControlName]</code>	<input type="radio"/>
<b>Exe Name</b>	Defines the name of the executable file generated. Be reminded that it should NOT be identical to the User Control Library file name.	<input type="radio"/>
<b>Scale Transform</b>	Denotes if the TMcraft Toolbar would change the scale automatically by the resolution of Control Box (1366 x 768 pixels). It is recommend to disable this transform if developer is confident with the width and height defined	<input type="radio"/>
<b>Width</b>	The width of toolbar, ranged between 100 ~ 440 pixels	<input type="radio"/>
<b>Height</b>	The height of toolbar, ranged between 100 ~ 700 pixels	<input type="radio"/>
<b>Zip Password</b>	Developers can define the password of the zip file. The password length should be between 6 and 256 characters of the non-case-sensitive Latin alphabet and numbers.	

※ Complete all parameters labeled with a red star in the TMcraft Packer GUI before proceeding.



The screenshot shows the 'TMcraft Toolbar' application window. The title bar includes a logo, 'File' menu, and 'About' button. The main area is titled 'TMcraft Toolbar' and contains several input fields for configuring the toolbar packaging. The 'Source folder' is set to 'C:\Users\...\Desktop\source\HellowWorldToolbar' and the 'Target folder' is 'C:\Techman Robot\TMcraft Packer\USB\TM\_Export\...\TMcraft\Toolbar'. The 'Toolbar Name' is 'myToolbar', the 'Provider' is 'Techman Inc', and the 'Version' is '1.0.0.0'. The 'Main DLL' is 'HellowWorldToolbar.dll' and the 'User Control' is 'HellowWorldToolbar.MainPage'. The 'Exe Name' is 'myHellowWorldToolbar'. The 'Width' is '330' and the 'Height' is '210'. The 'Zip Password' is 'abcd1234'. A 'Generate' button is located at the bottom right. The status bar at the bottom shows 'TMcraft Version: 1.16.1400' and 'File Name: myHellowWorldToolbar'.

TMcraft Toolbar

\* Source folder: C:\Users\...\Desktop\source\HellowWorldToolbar

\* Target folder: C:\Techman Robot\TMcraft Packer\USB\TM\_Export\...\TMcraft\Toolbar

\* Toolbar Name: myToolbar

Provider: Techman Inc

\* Version: 1.0.0.0

Copyright: Techman Inc.

\* Main DLL: HellowWorldToolbar.dll

\* User Control: HellowWorldToolbar.MainPage

☐ Scale Transform

\* Exe Name: myHellowWorldToolbar

\* Width(100~440): 330

\* Height(100~700): 210

Zip Password: abcd1234

Generate

TMcraft Version: 1.16.1400 File Name: myHellowWorldToolbar

Figure 17: Set up the Packaging of TMcraft Toolbar

Once all requirements are ready, click **Generate**. It might take several minutes to package, and users can check the current progress on the Page (Users will see an error message if anything goes wrong). After packaging the TMcraft Toolbar successfully, users should see the successful status label.

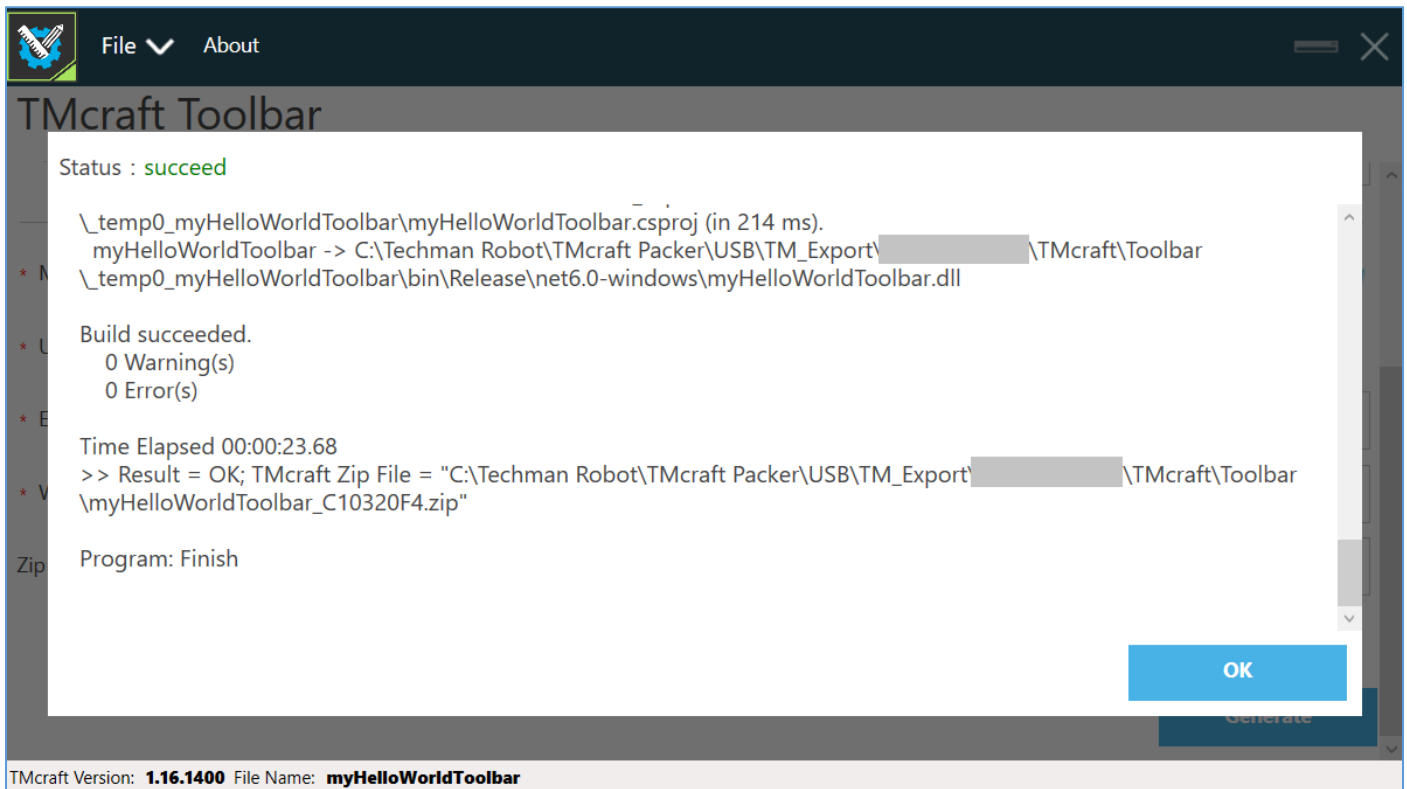


Figure 18: The Message Shown When Packaged Successfully

Finally, users can find the TMcraft Toolbar zip file named after *[Exe Name]\_[Checksum]* in the target folder.

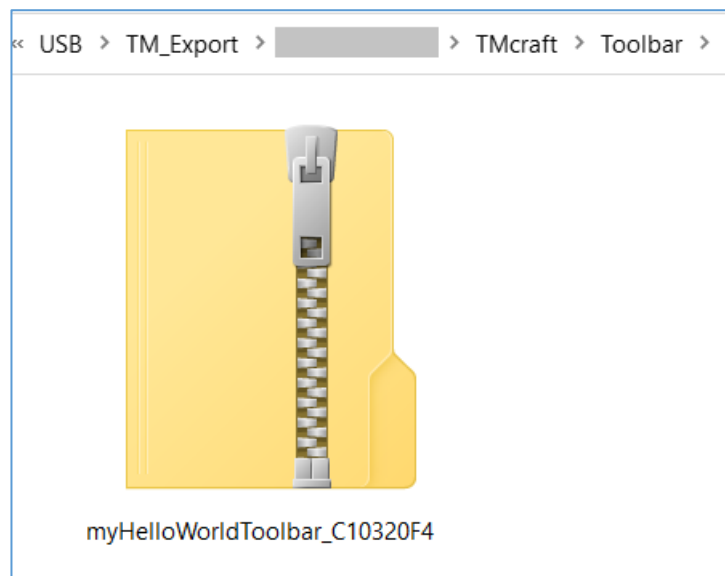


Figure 19: TMcraft Toolbar zip at Target folder

## 6. Installation Code and Checksum

To protect the rights and interests of developers, each TMcraft Packer Project has its own encrypted Installation Code (GUID-based). Any TMcraft items generated by this project will share the same Installation Code and be able to replace one another on the same robot. Additionally, items generated from two different TMcraft Packer Projects (without identical Installation Codes) cannot replace each other, even if they share the same name and configuration. Therefore, developers should keep their packer project safe.

On the other hand, TMcraft Packer will also generate a checksum for each TMcraft item based on the binary footprint of the files (exe and dll) within the source folder and the Installation Code. The TMcraft item saves the checksum onto the configuration file. When importing the TMcraft item, TMflow will calculate a checksum by the same method; if these two checksums are not identical, TMflow will block the importing.

Developers can announce the checksum, allowing end users to verify it on the TMcraft Management Page to ensure the product's authenticity.

## 7. Use TMcraft Toolbar on TMflow

To use a TMcraft Toolbar on both TMflow (robot) and TMflow Simulator, import the TMcraft Toolbar Package to a robot by placing the zip file in the following path:

*[Storage Drive]\TM\_Export\[Folder]\TMcraft\Toolbar*

Plug the storage device into the robot and navigate to **System > Import/Export > Import**. Note that there is a new category: **TMcraft**. Select **TMcraft > Toolbar**, choose the required TMcraft Toolbar zip file, and click **Import**.

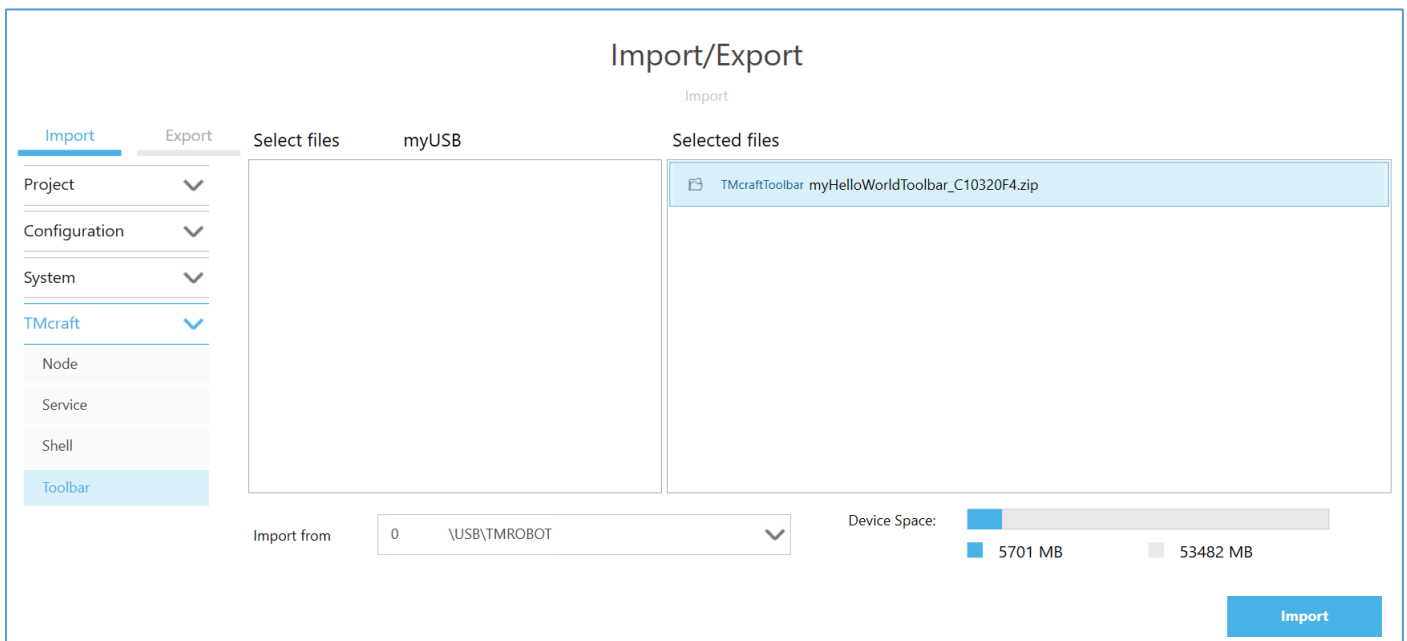


Figure 20: Import TMcraft Toolbar zip

Next, navigate to **Configuration > TMcraft Management > Toolbar**. Users can see the imported TMcraft Toolbar shown in the table with the following information:

- **Name:** the executable file name of the TMcraft Toolbar
- **Provider:** as defined in Chapter 5
- **Version:** as defined in Chapter 5
- **Checksum:** a number calculated based the binary footprint of the files (exe and dll) in the TMcraft Toolbar folder. Developers may publish this for identification purpose to their end-users.



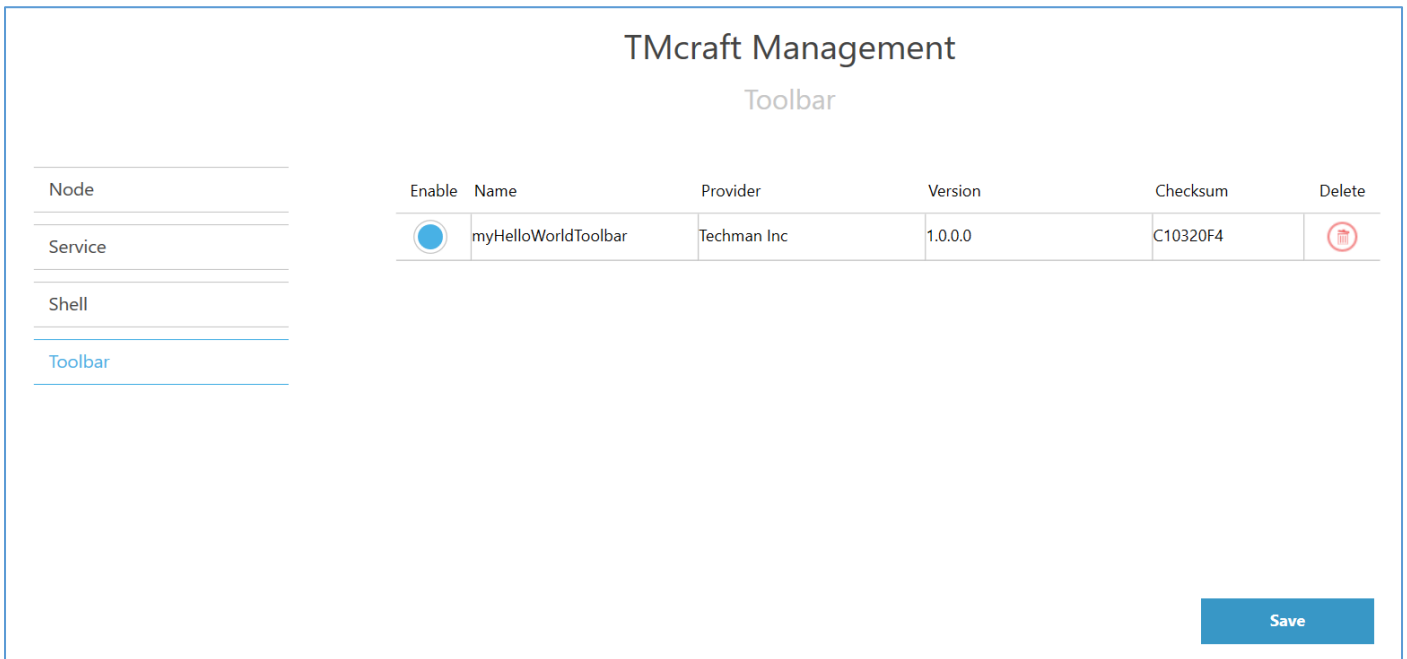


Figure 21: Enable the TMcraft Toolbar on TMcraft Management

Once the item is enabled, a TMcraft Toolbar button will show at the bottom bar of TMflow. Click it to pop up the relevant TMcraft Toolbar UI at the upper right. Users can drag the toolbar by holding the top bar of the TMcraft Toolbar. There are two buttons on the TMcraft Toolbar:

- The Hide button: click to hide the TMcraft Toolbar without closing the program
- The Close button: click to close the TMcraft Toolbar program

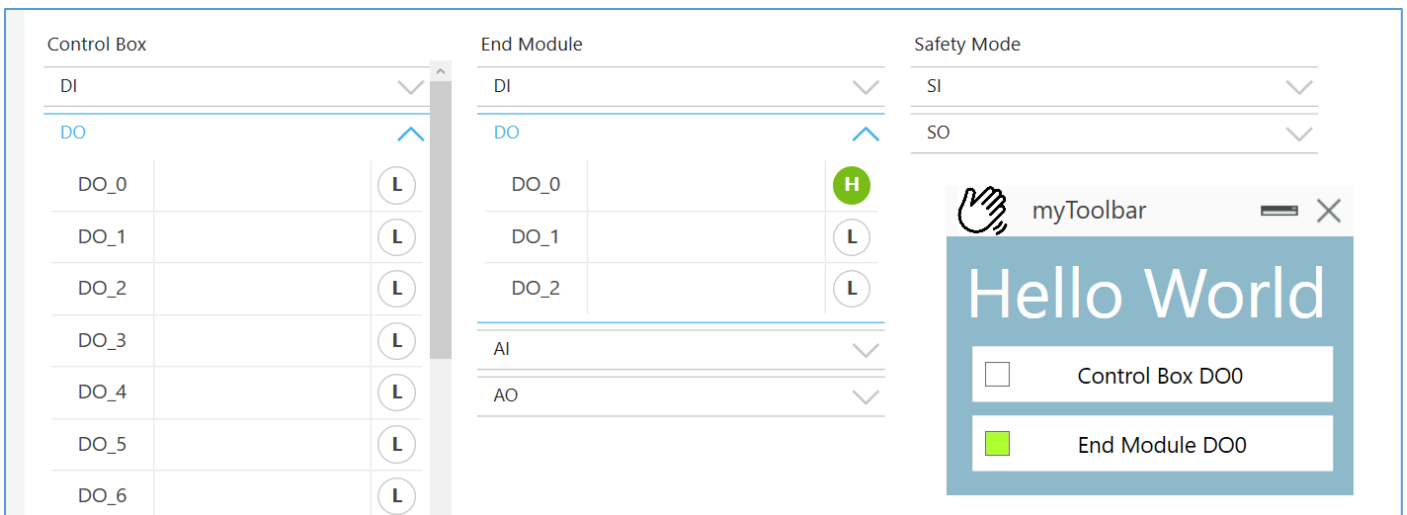


Figure 22: Back to TMcraft Toolbar by Clicking the Button at Upper Left Upper

## 8. Dos & Don' ts

- About program reference/libraries:

If using NuGet to install and manage libraries within the TMcraft program, ensure the library files are accessible for placement in the TMcraft packaging source folder. Accomplish this by adding or modifying the property <CopyLocalLockFileAssemblies> to true within the csproj file. After compiling the project, users should find all reference files within the bin folder.

- About resource path definition:

When using resources in the TMcraft Program, it is necessary to use an absolute path (`pack://application:...,/{Name of the User Control Library Project}; component/Resources/{String Resource file}`) instead of the relative path to access resources; otherwise, it will not work.

- Suggestions about how to use TMcraft API functions:

1. Check if the `TMcraftToolbarAPI` object is null or not
2. Use a `uint` object to get the result of the API function (replied by TMflow)
3. Check if the result is 0 or not; if not, call `GetErrMsg()` to get the corresponding error message
4. Note that, instead of TMflow error, there might also have error from the API itself; this error is represented as a `TMcraftErr` object and returned by calling `GetErrMsg()`.

- About writing files onto Control Box through TMcraft items

Robot System will deny the following access from TMcraft executable, including: (1) system shutdown and (2) accessing to drives C and D, except for the following folders:

- ✓ D:\...TMflow\TMcraft\
- ✓ D:\...TMflow\XmlFiles\
- ✓ D:\...TMflow\TextFiles\

- Having Log

The TMcraft Program should save log files within its folder, which allows end-users to export the TMcraft zip, including the log files, for developer analysis.

- Before packaging with TMcraft Packer:

- ✓ Place all API relevant files from the TMcraft Development Kit into the source folder.

- ✓ Place all files from the bin folder of the TMcraft Project into the source folder
- ✓ Ensure TMcraft Packer is running in the environment with .NET 6.0 installed.

