



TMcraft Node Case Study Welding Node

Original Instructions

Document version: 1.0

Release date: 2023-06-28

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) and shall not be reproduced in whole or in part without prior authorization from the Corporation. No information contained herein shall be considered an offer or commitment. The information herein is subject to change without notice. The document is periodically reviewed and revised. The Corporation assumes no responsibility for any errors or omissions in the documentation.

TM logo is 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.

Table of Contents

Revision History.....	3
1. Overview	4
2. Prerequisites	4
3. Workflow.....	5
4. Data Communication	6
5. Start to program the welding node	7
6. User Interface	8
7. User Friendly Features (Optional).....	11
8. Keywords.....	13
9. Appendix.....	13

Revision History Table

Revision	Date	Description
1.0	2023-06-28	Original release

1. Overview

The example in this case study demonstrates how to use TMcraft to create a node for Welding Applications.

A welding application has many line/arc movements, data communication, and user inputs. In TMflow 1 series, users must build applications through multiple nodes and project function settings. For better data sharing between weld points, TMflow 2 series uses TMcraft (Node) API to make the entire welding job within a single node instead.

Developing welding nodes requires a basic knowledge of TMcraft Node. Please read *TMcraft Node Development Tutorial* beforehand.

2. Prerequisites

To develop TMcraft Nodes, developers should be capable of having:

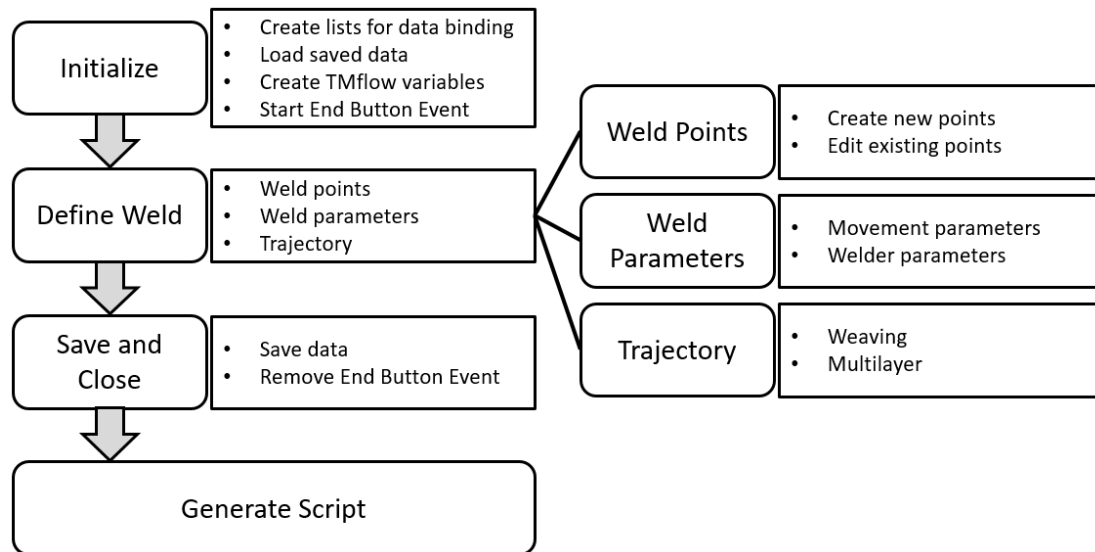
- Basic knowledge of C# programming
- Using WPF to build UI
- Usage of Programming Language TMsript

Software requirements,

- An integrated development environment for C# and WPF (such as Microsoft Visual Studio 2022)
- TMstudio Pro
- TMflow 2.14 or above

3. Workflow

The operation process of the node goes as below.



The node initializes when users open the node edit page, calling the UserControl Load event. In the case of the Welding Node, it initializes objects for data binding, loads previously saved data, and sets up communication between the node and TMflow.

Users then define the welding job in the Node Edit Interface, creating weld points and setting welding parameters. In addition, users can call API functions in this section for better user interaction and data communication.

The node page should have a save and close button to prevent it from closing before script generation is complete. The click event would include saving data, removing the event, and generating a script in string List format before calling to close the node. Once closed the node interface, it will call `InscribeScript()` to receive the script for later execution.

4. Data Communication

- Data Storage

Node data has several serializable classes: WeldPoints, WeldDevice, WeldProgram, MultiLayer, MLLists, Variables, and IO Settings. For data saving, lists of each class instance are serialized into JSON strings and then saved via *DataStorageProvder* as strings. In addition, separate *.json* files preserve the JSON strings as backups.

- Simultaneous Control

Use TMNodeEditor Providers for simultaneous controls, including retrieving current pose, IO control, get and set variables, and jog control.

Providers and functions used in this example:

- RobotStatusProvider.GetCurrentPoseByRobotBase
- VariableProvider.GetProjectVariableList
- VariableProvider.CreateProjectVariable
- VariableProvider.ChangeProjectVariableValue
- VariableProvider.GetGlobalVariableList
- VariableProvider.CreateGlobalVariable
- VariableProvider.ChangeGlobalVariableValue
- IOProvider.WriteDigitOutput
- RobotJogProvider.OpenControllerPanel
- RobotJogProvider.JogByBase
- RobotJogProvider.StopJog
- RobotStatusProvider.EndButtonClickEvent
- RobotEventType

- Project Run Control

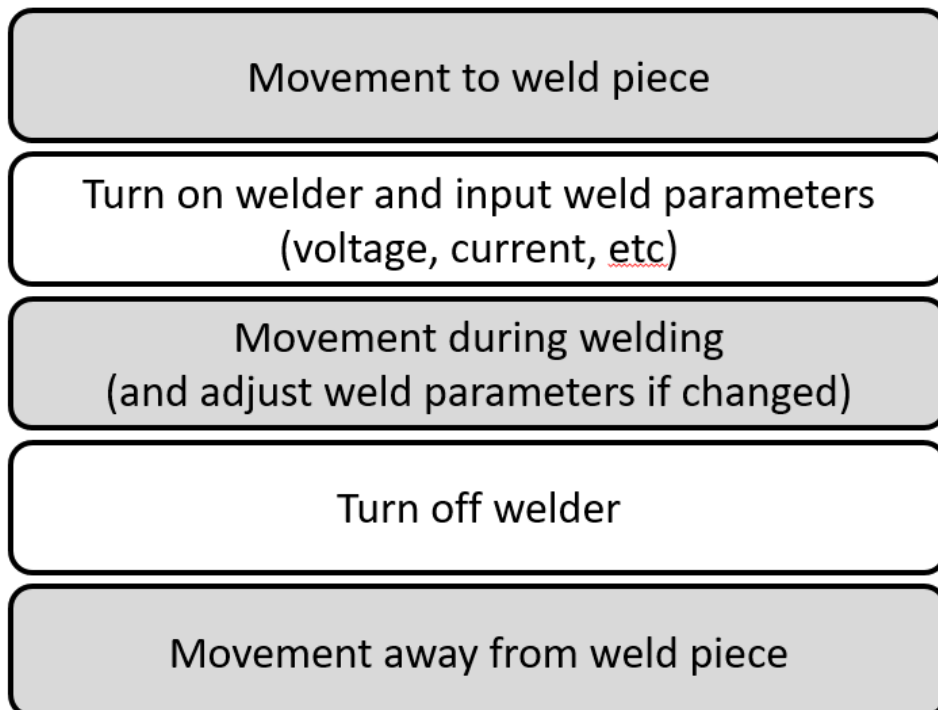
When pressing the Save and Close Button, the system calls the ScriptCommand method. It goes through all weld points and creates a List with strings of TMScript commands based on their functions and other parameters.

The InscribeScript method is called by TMflow when the node closes, saving the script commands into ScriptWriteProvider for control when the project executes.

For example, the Weld Line Functions send a list of PLine() script commands to move to calculated path points. The system performs Modbus communications with modbus_write() script commands.

5. Start to program the welding node

To create the Welding Node, users should first visualize the final TMscript output for a simple welding job:

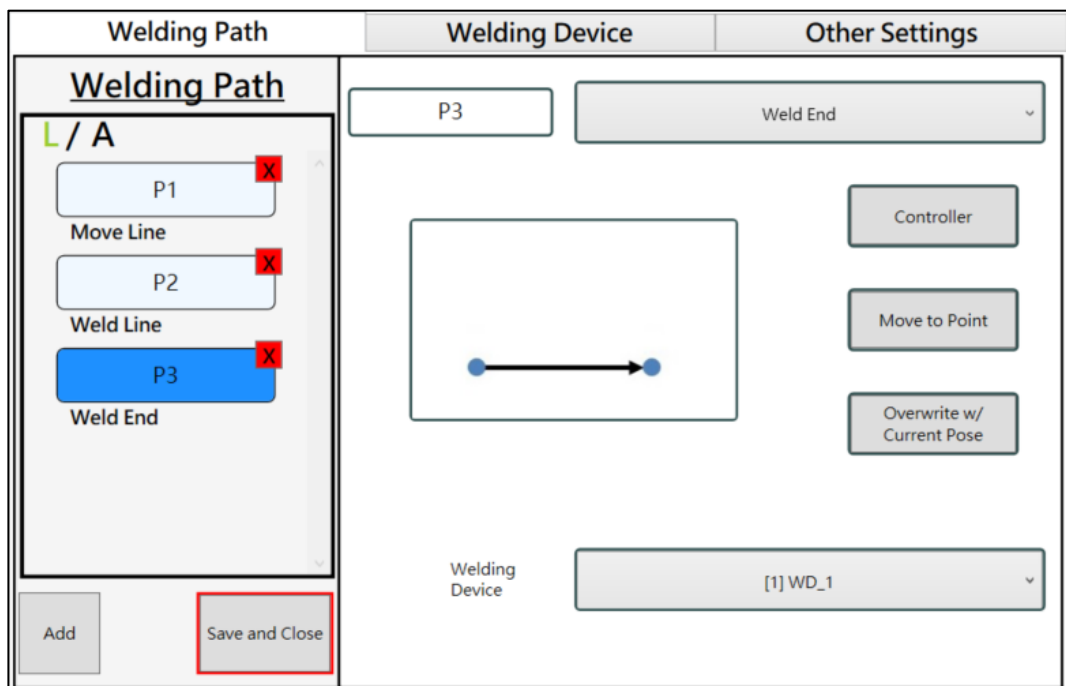


Then, users should define the parameters needed to create this script. The most important information would be the coordinates for all points. In addition, the definition requires the movement speed, the type of each movement, and data to send to the welder and the IO channel for the welder. After defining the required information to gather, users can move to plan out the user interface.

6. User Interface

The User Interface is where all the parameters are collected. In this example, the interface consists of three tabs: tab **Welding Path**: to define point and trajectory, tab **Welding Device**: to set parameters of movement and welding, and tab **Other Settings**: to set the welder behaviors.

1. Welding Path



Weld points are recorded as public class `WeldPoints` and added into an `ObservableCollection` `weldpointCollection`. The fields of `WeldPoints` consist of a weld point's function, coordinates, the position in the path, the previous and next point positions, and the program variable settings. The TCP coordinates are retrieved and recorded using `RobotStatusProvider.GetCurrentPoseByRobotBase` while adding a new point.

For clear visualization of the welding path, each weld point shows as a grid in an `ItemsControl` with data bound to the `ObservableCollection`, making it easier to display point addition, change, and removal.

The panel at the right displays the selected point information. The `Controller` Button calls `RobotJogProvider.OpenControllerPanel` and opens the `Controller` Panel at the side of the interface. The `MovetoPoint` button calls `RobotJogProvider.JogLineByCoordinates` to send jog commands into the queue.

2. Welding Device

In the tab **Welding Device**, stored as an object defined by the custom class `WD`, each row of data contains parameters. The rows are included in a list and connected to the textboxes of a `Listview Table`, displaying

values and allowing change.

Welding Path			Welding Device					Other Settings			
#	Title	Start Voltage (V)	Start Current (A)	Start Time (s)	End Voltage (V)	End Current (A)	End Time (s)	Lin. Weld Speed (mm/s)	Rot. Weld Speed (°/s)	Move Speed (%)	Blending (%)
1	WD_1	0	0	0	0	0	0	0	0	0	0
2	WD_2	0	0	0	0	0	0	0	0	0	0
3	WD_3	0	0	0	0	0	0	0	0	0	0
4	WD_4	0	0	0	0	0	0	0	0	0	0
5	WD_5	0	0	0	0	0	0	0	0	0	0
6	WD_6	0	0	0	0	0	0	0	0	0	0
7	WD_7	0	0	0	0	0	0	0	0	0	0
8	WD_8	0	0	0	0	0	0	0	0	0	0
9	WD_9	0	0	0	0	0	0	0	0	0	0
10	WD_10	0	0	0	0	0	0	0	0	0	0
11	WD_11	0	0	0	0	0	0	0	0	0	0
12	WD_12	0	0	0	0	0	0	0	0	0	0
13	WD_13	0	0	0	0	0	0	0	0	0	0
14	WD_14	0	0	0	0	0	0	0	0	0	0
15	WD_15	0	0	0	0	0	0	0	0	0	0
16	WD_16	0	0	0	0	0	0	0	0	0	0
17	WD_17	0	0	0	0	0	0	0	0	0	0

Export Import Clear

3. Other Settings

In the tab **Other Settings**, users define the communication between the robot and the welder. Users can test with **Wire Forward**, **Wire Backward**, and **Gas** sections.

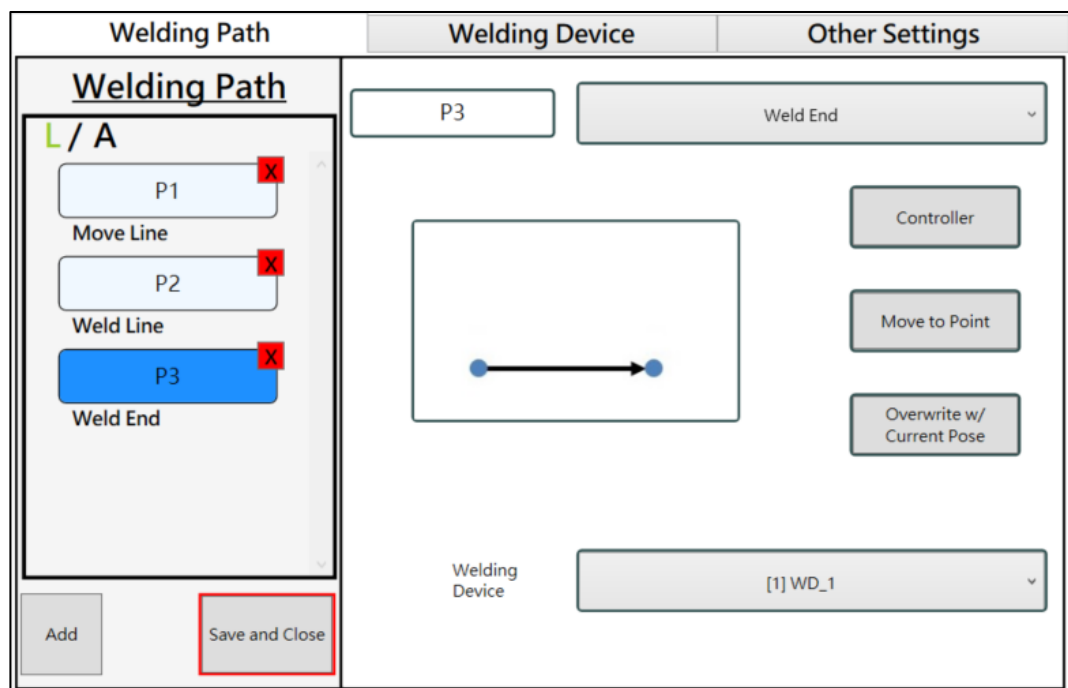
Welding Path	Welding Device	Other Settings
Modbus Device: <input type="text" value="localhost"/> <input type="checkbox"/> Enable Numpad		
▼ Wire Forward	<input type="button" value="Test Wire Forward"/>	
▼ Wire Backward	<input type="button" value="Test Wire Backward"/>	
▼ Gas	<input type="button" value="Test Gas"/>	
▼ Arc A	<input type="button" value="Test Arc A"/>	

Welding Path	Welding Device	Other Settings
<input type="checkbox"/> Enable Numpad		
▼ Wire Forward	Test Wire Forward	
▼ Wire Backward	Test Wire Backward	
▼ Gas	Test Gas	
▲ Arc A		
Arc On	▼ IO	
	▼ Variables	
	▼ Modbus	
Arc Off	▼ IO	
	▼ Variables	
	▼ Modbus	
Test Arc A		

Welding Path	Welding Device	Other Settings																																
	▲ IO																																	
	<table border="1"> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td> </tr> <tr> <td>H</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> </table>	0	1	2	3	4	5	6	7	H								8	9	10	11	12	13	14	15									
0	1	2	3	4	5	6	7																											
H																																		
8	9	10	11	12	13	14	15																											
	Light Module	On																																
Arc On	▲ Variables																																	
	Weld_VoltageOutput1	= (var_Weld_VoltageTarget1-7)*135.2																																
	Weld_CurrentOutput1	= (var_Weld_CurrentTarget1-32)*8.62																																
	▲ Modbus																																	
	preset_Voltage	= (int)var_Weld_VoltageOutput1																																
	preset_Current	= (int)var_Weld_CurrentOutput1																																

7. User Friendly Features (Optional)

Adding some features will make for the user experience enhancement. For example, if enabled, a number pad prompts when ticked parameter textboxes to help users input values well with the teach pendant. The number pad is created with a separate WPF Class Library Project and referenced by the main UserControl.




To prevent users from clicking the Overwrite button accidentally, the mechanism of a RepeatButton that triggers every second is applied. A countdown initiates once pressed the button, and users must hold the button for 3 seconds to confirm overwriting.

For safety reasons, TM robots require a MovingBeacon command to trigger the movement and a StopBeacon to discard the movement command. The robot movements cannot start with UI buttons only. MovingBeacon is bound to the PLAY button on the Robot Stick, so the robot does not move instantly after clicking the MovetoPoint button. Users have to hold the PLAY button for motion.

Created as a ToggleButton, if toggled, the MovetoPoint button sends the movement command and waits for the robot stick input. In addition, the MovetoPoint button disables all the controls except itself, the Weld Path ItemsControl, and the Save and Close button. When users select a different weld point, the robot will move toward the newly selected one instead. If not toggled, the button unchecks, enables all elements, and sends a StopBeacon, clearing pending movement commands.

In addition to the Add button, users can add points with different functions using the end buttons on the robot flange. Once pressed any end button, the system calls the RobotStatusProvider.EndButtonClick event and RobotEventType identifies the button. Please refer to *TMcraft Node Tutorial End Button event* for



details on using the End Button.

8. Keywords

C#	WPF
INotifyPropertyChanged	Binding
JsonConvert.SerializeObject	Define and Reference a Resource
JsonConvert.DeserializeObject	ItemsControl
Dispatcher.Invoke	Triggers, Setter
BindingExpression	Storyboard
ICollectionView	

9. Appendix

Users can download the sample solution of the welding node from the Techman Official Website.

TECHMAN ROBOT



www.tm-robot.com