

# **WSG Series of Intelligent Servo-Electric Grippers Interfacing to Mitsubishi Robots**

Firmware Version 4.0.0

May 2015



[www.weiss-robotics.com](http://www.weiss-robotics.com)

## Contents

<b>1</b>	<b>Introduction .....</b>	<b>3</b>
<b>1.1</b>	<b>Preliminary remarks.....</b>	<b>3</b>
<b>1.2</b>	<b>System requirements .....</b>	<b>3</b>
<b>1.3</b>	<b>Restrictions.....</b>	<b>3</b>
<b>2</b>	<b>Preparation .....</b>	<b>4</b>
<b>2.1</b>	<b>Gripper configuration.....</b>	<b>4</b>
<b>2.2</b>	<b>Robot configuration .....</b>	<b>5</b>
2.2.1	Network settings for the robot interface .....	6
<b>3</b>	<b>Getting started .....</b>	<b>9</b>
<b>4</b>	<b>Robot program .....</b>	<b>12</b>
<b>4.1</b>	<b>Main program „GripCycle“ .....</b>	<b>12</b>
<b>4.2</b>	<b>Main program „Test“ .....</b>	<b>12</b>
<b>4.3</b>	<b>Sub programs providing the gripper functions .....</b>	<b>12</b>
4.3.1	WSGBye .....	13
4.3.2	WSGVerbose.....	13
4.3.3	WSGDevType.....	13
4.3.4	WSGHome .....	14
4.3.5	WSGMove.....	14
4.3.6	WSGGrip .....	14
4.3.7	WSGRelease.....	15
4.3.8	WSGSysFlag .....	15
4.3.9	WSGGripState.....	15
4.3.10	WSGPosition .....	16
4.3.11	WSGSpeed .....	16
4.3.12	WSGForce .....	16
4.3.13	WSGTemp.....	17
4.3.14	WSGFastStop .....	17
4.3.15	WSGFSAck.....	17
4.3.16	WSGFTYPE .....	17
4.3.17	WSGTare .....	18
<b>4.4</b>	<b>Helper Programs .....</b>	<b>18</b>
4.4.1	ParseErr .....	18

# 1 Introduction

## 1.1 Preliminary remarks

This document describes how to set up and use a WSG series gripper with Mitsubishi robot arms and controllers of type *CRnD-7xx/CR75x-D* using the MELFA-BASIC V programming language and the *RT ToolBox2* program environment via TCP/IP network communication.

It is assumed in this manual that the robot arm has been set up as described in the user's manual. A basic knowledge of the MELFA-BASIC V programming language is required. Also, the information given in the gripper's user's manual is mandatory.

The connection between the robot control and the gripping module is established via the TCP/IP network interface by using the text based command protocol "Gripper Control Language" (GCL). This protocol is available from firmware version 4.0.0. Additional information can be found in the "WSG GCL Reference Manual", which comes with the gripper.

## 1.2 System requirements

The example program described in this document has been implemented and tested on a robot of type *Mitsubishi RV-4FL-D* with a controller of type *CR-750D*. It has been written in the MELFA-BASIC V programming language using Mitsubishi's *RT ToolBox2* programming environment. It should be possible to easily port the program to other MELFA-BASIC V supporting controllers as well. However, Weiss Robotics cannot guarantee that the program runs on other controllers, too.

## 1.3 Restrictions

The described program is considered as an example, illustrating the communication between robot control and gripping module.

Although the programs have been checked and tested carefully, Weiss Robotics cannot guarantee that these programs are stable and free of errors.

Before using the program in productive environments, appropriate tests must be run to make sure that the system works properly under all conditions and circumstances. This is especially necessary if only parts of the program are used. The responsibility lies with the robot programmer.

## 2 Preparation

Connect the gripping module to the power supply. Then, connect the module's Ethernet cable to the robot controller using an appropriate switch. The switch may also provide a connection to other components (e. g. a laptop) or the local network.

Please refer to the information given in the gripping module's user's manual.

### 2.1 Gripper configuration

To run the gripper on a Mitsubishi robot arm, it must first be configured via the integrated web interface. Select "*Settings -> Command Interface*" from the menu (cf. Figure 1) and choose „TCP/IP“ as the command interface (default). Then activate the option "Use text-based interface" by enabling the corresponding checkbox (cf. Figure 2). The module will now accept incoming commands using the "Gripper Control Language" (GCL).

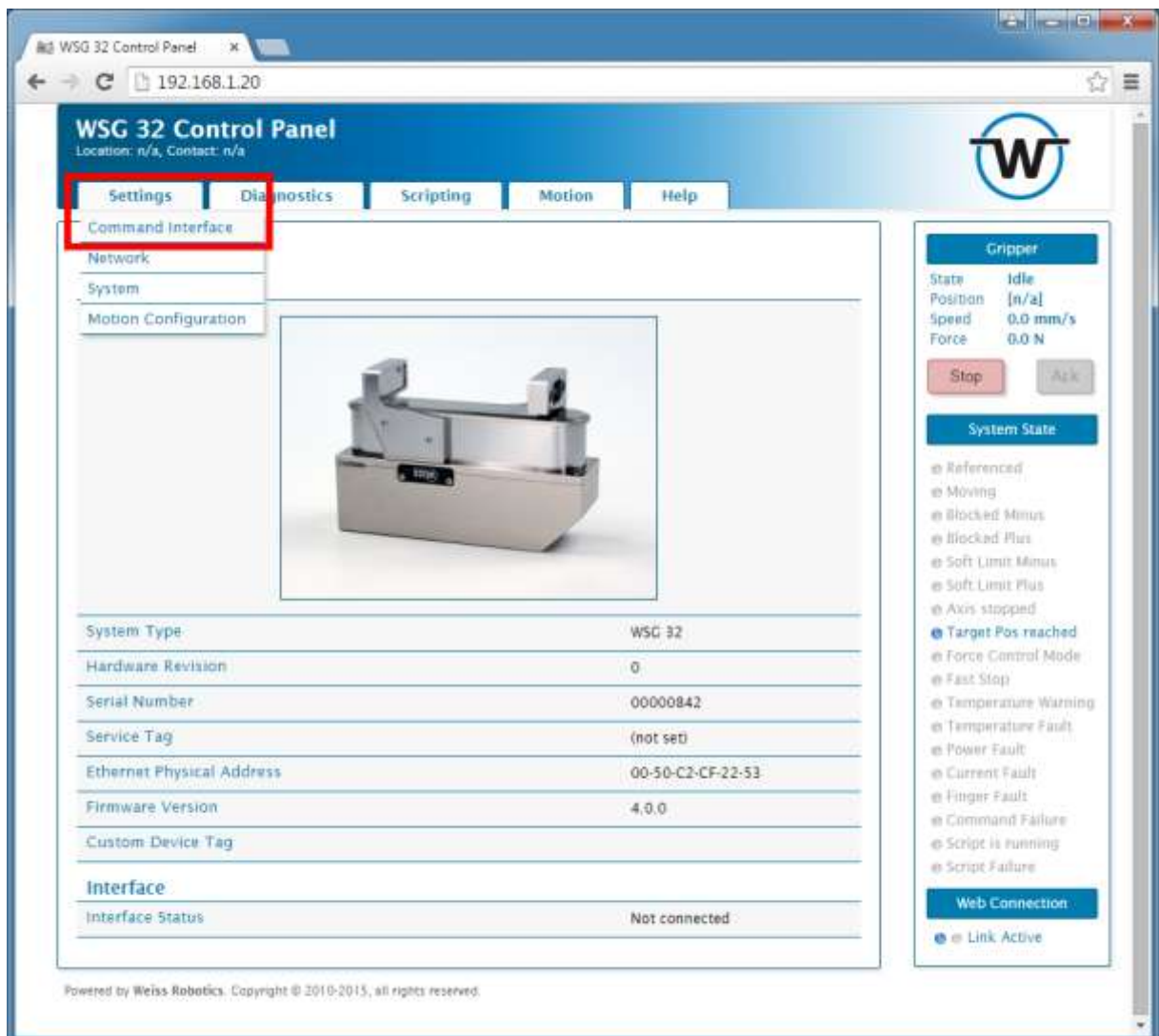


Figure 1: Main page and menu of the gripper's web interface

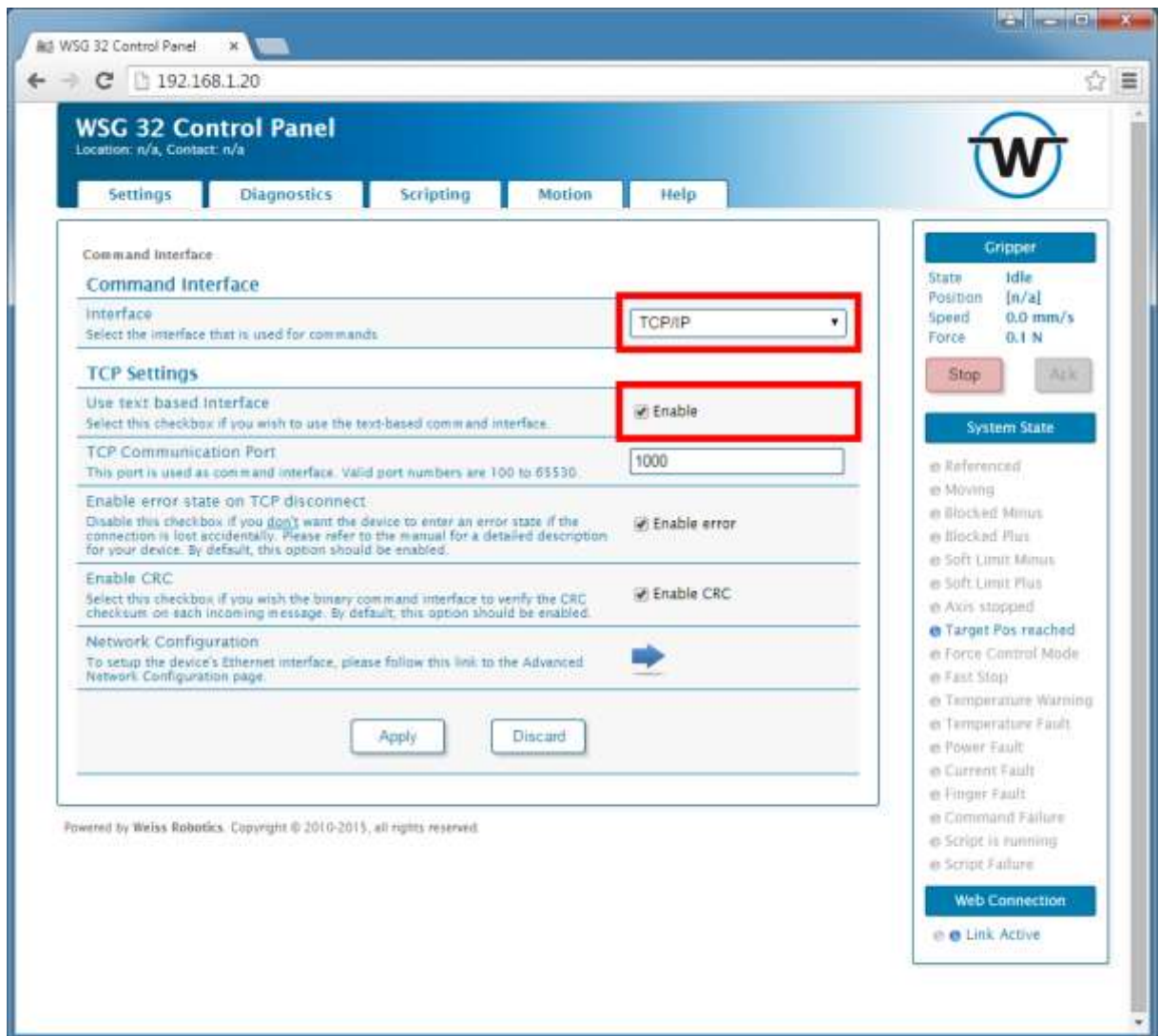


Figure 2: Interface configuration on the gripper's web interface

The web interface can also be used to adjust any settings concerning the network connections like IP address, subnet mask, gateway or DNS server. However, the example described in this manual assumes that the gripper is set to default settings. Please refer to the user's manual for any further information.

**i** By default, the gripper is configured to use IP address 192.168.1.20 and subnet mask 255.255.255.0.

## 2.2 Robot configuration

It is assumed that the robot arm has been set up as described in the user's manual.

To connect the gripper to the robot control, the network interface settings have to be adjusted and a new network connection must be configured to be able to communicate with the gripper.

### 2.2.1 Network settings for the robot interface

The robot control's network settings can either be set using the teaching box or the programming environment *RT ToolBox2*. This manual focuses on the teaching box.

To change the network settings, choose "Parameters" from the main menu, then click the button "Parameter menu" on the lower left side. From the popup menu, choose "Communication Parameter" -> "Ethernet" (cf. Figure 3, Figure 4 and Figure 5).

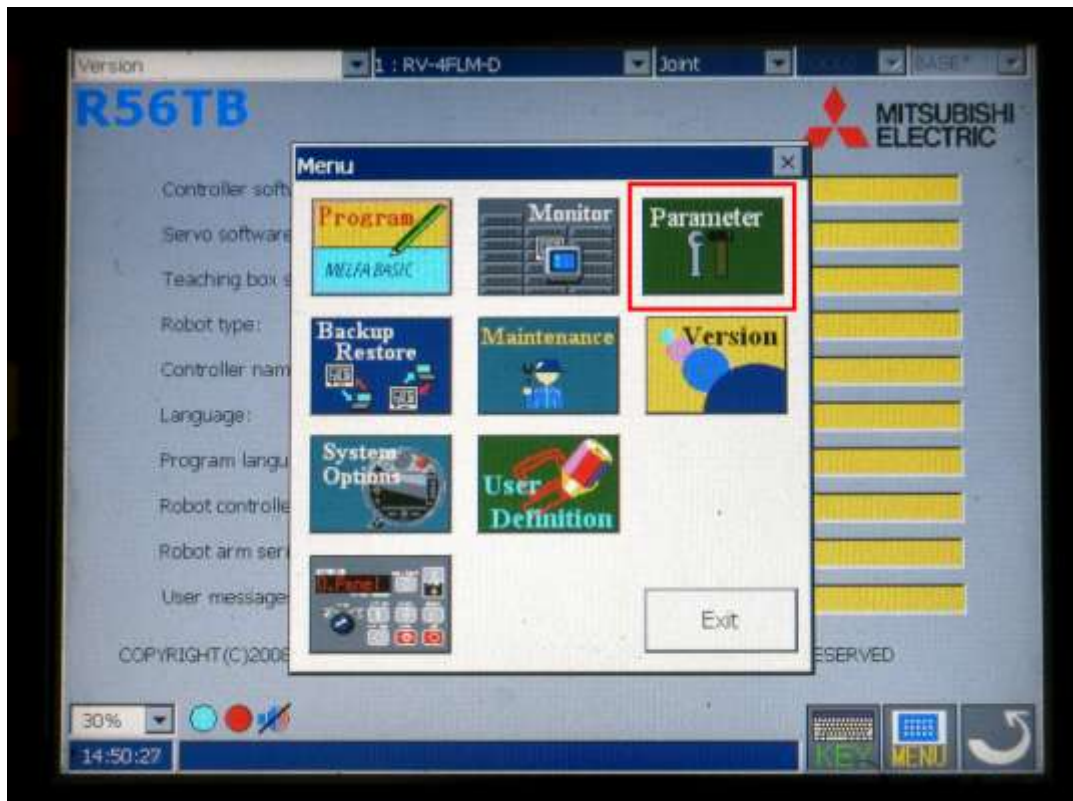


Figure 3: Main menu on the teaching panel

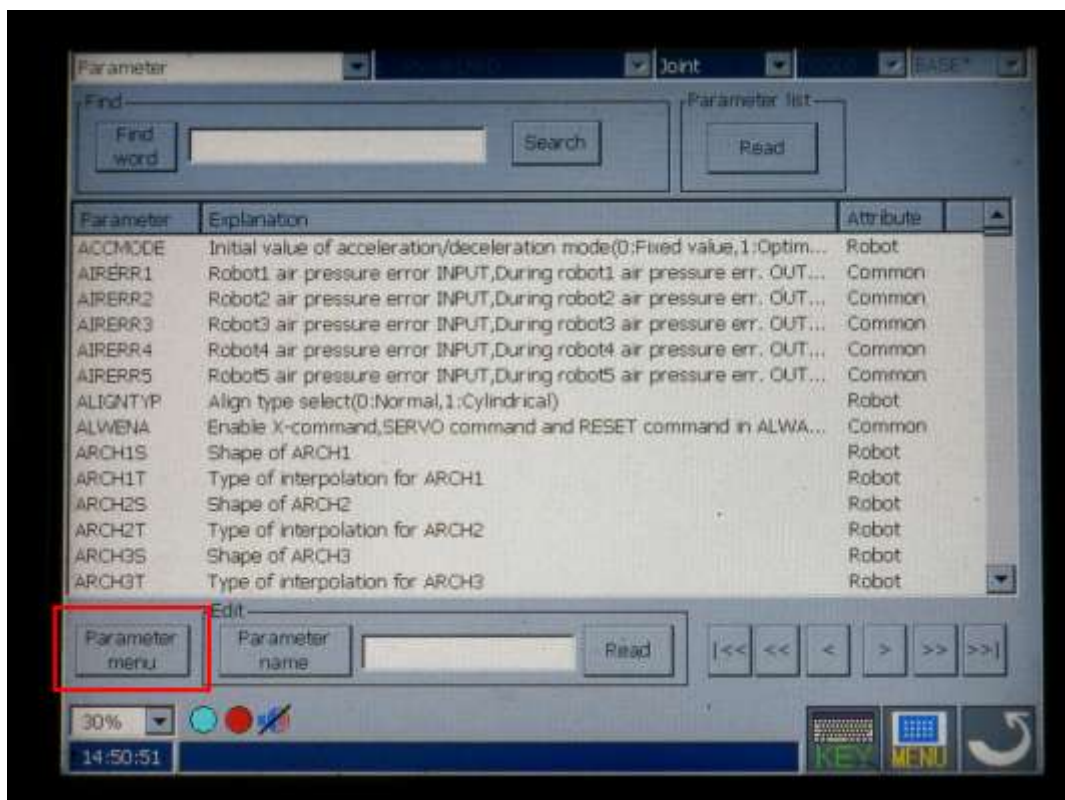


Figure 4: Overview of the configuration parameters

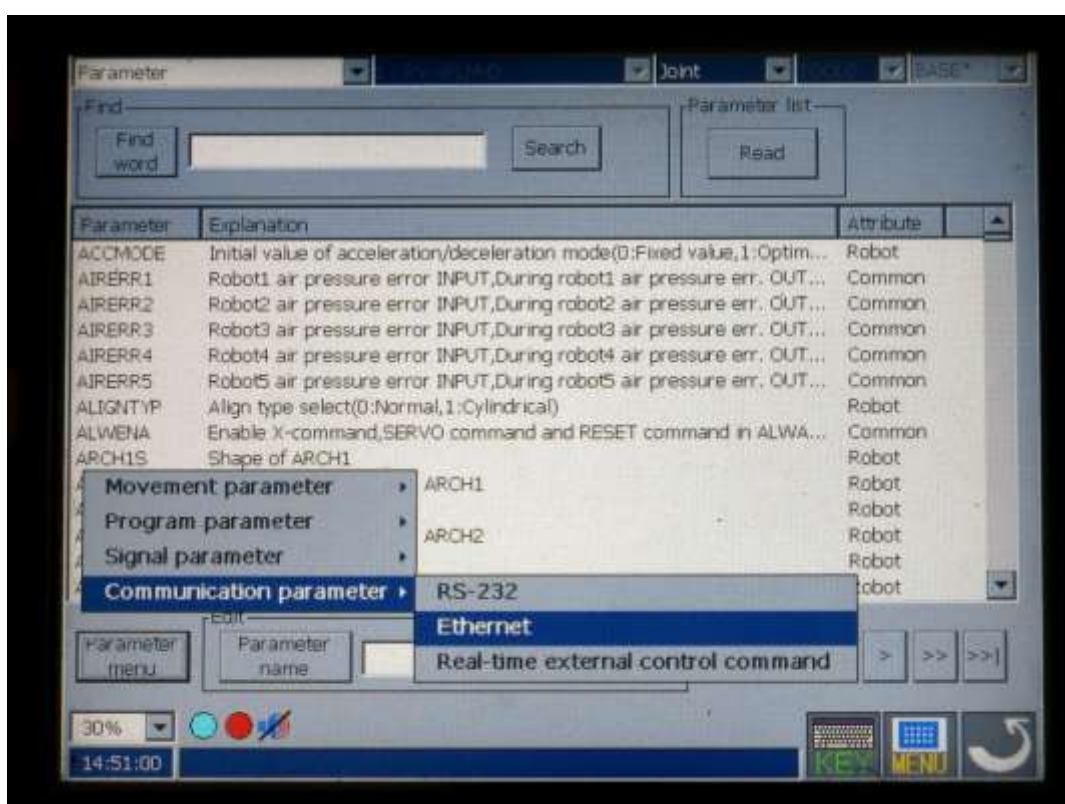


Figure 5: Communication parameters of the Ethernet interface

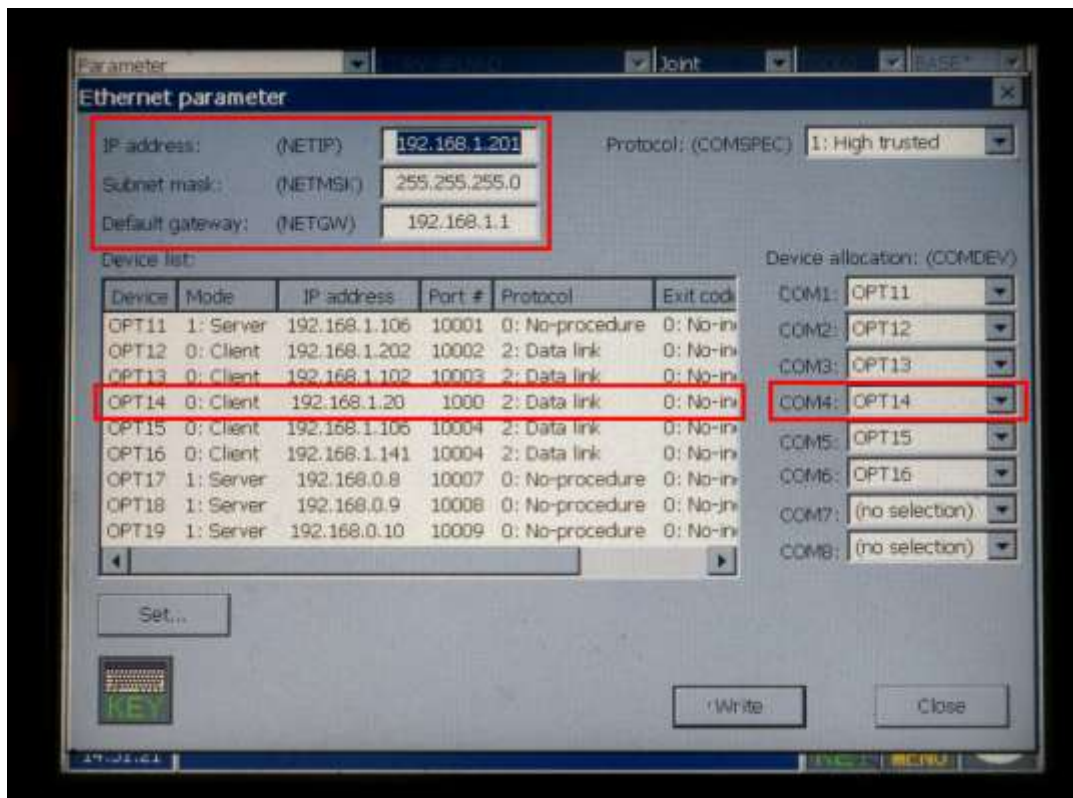


Figure 6: Overview of the network settings

Change the IP address (parameter NETIP) to 192.168.1.201. The subnet mask (parameter NETMSK) must be set to 255.255.255.0 (cf. Figure 6). The address of the default gateway is not relevant here unless you need the robot controller to connect to a remote network using a router. If this is the case, change the default gateway address to the address of that router. If in doubt, please contact your network administrator.

Additionally, a device and a connection must be configured for the robot control to connect to the gripper. Select OPT14 and click the button “Set...”. Choose the mode “Client”, IP address 192.168.1.20 (or whatever address you have assigned to it in chapter 2.1), port 1000, protocol “Data Link” and packet type “CR+LF”. Then, assign OPT14 to communication device COM4.

- i** The address settings of the gripper and the robot control can of course be chosen freely according to your requirements. However, the program example will always try to connect to the gripper using COM4 (unless you change it in the source code), so it’s strongly recommended to always use COM4 as communication device.
- i** If the IP address has been changed on the gripper, the IP address used for the device connection on the robot control must be set accordingly.

### 3 Getting started

Open the project *WSGExample* in the *RT ToolBox2* programming environment. First, unpack the ZIP archive into a folder of your choice. Then, start the *RT ToolBox2* and choose “Workspace” -> “Open” from the menu. Now select the folder you have previously unpacked the ZIP archive to and select the sub folder “WSGExample”.

Connect to the robot control by clicking the “Online” button in the tool bar. In the tree on the left, unfold the “Online” branch (cf. Figure 8) and open the “Program Manager” by right-clicking onto “Program”.

Now load the project’s example programs onto the robot.

Load and start the program “GripCycle” either by using the teach panel or via the Operation panel available in *RT ToolBox2* (cf. Figure 10).

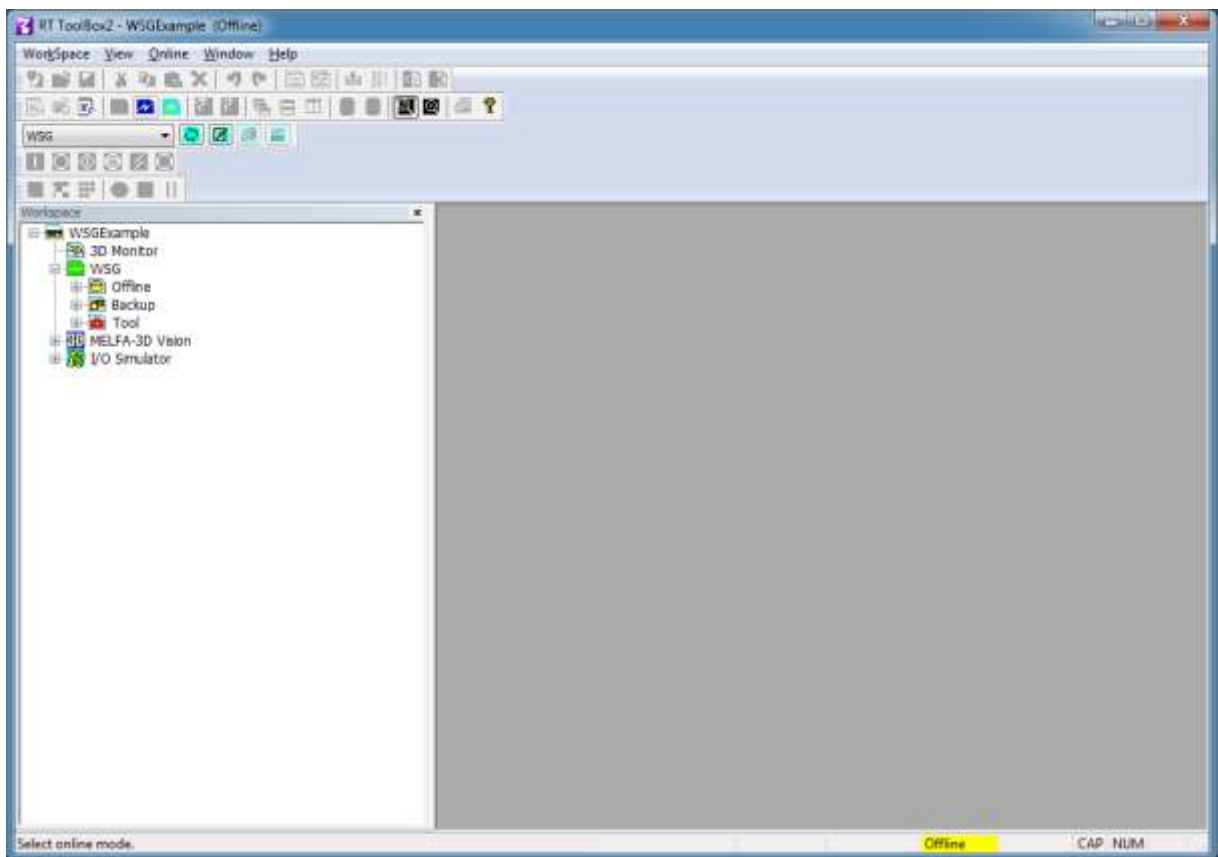


Figure 7: Workspace „WSGExample“ in the *RT ToolBox2* programming environment

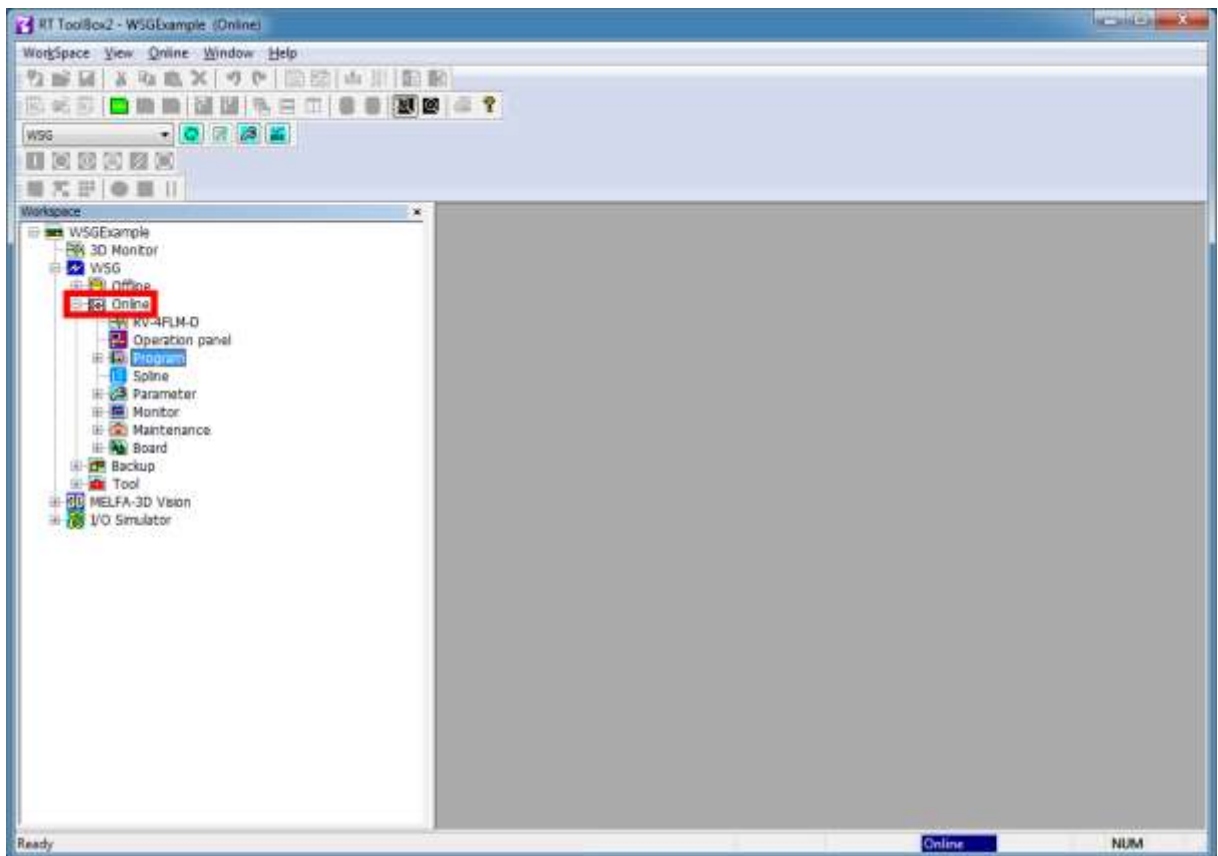


Figure 8: Workspace „WSGExample“ connected to the robot control

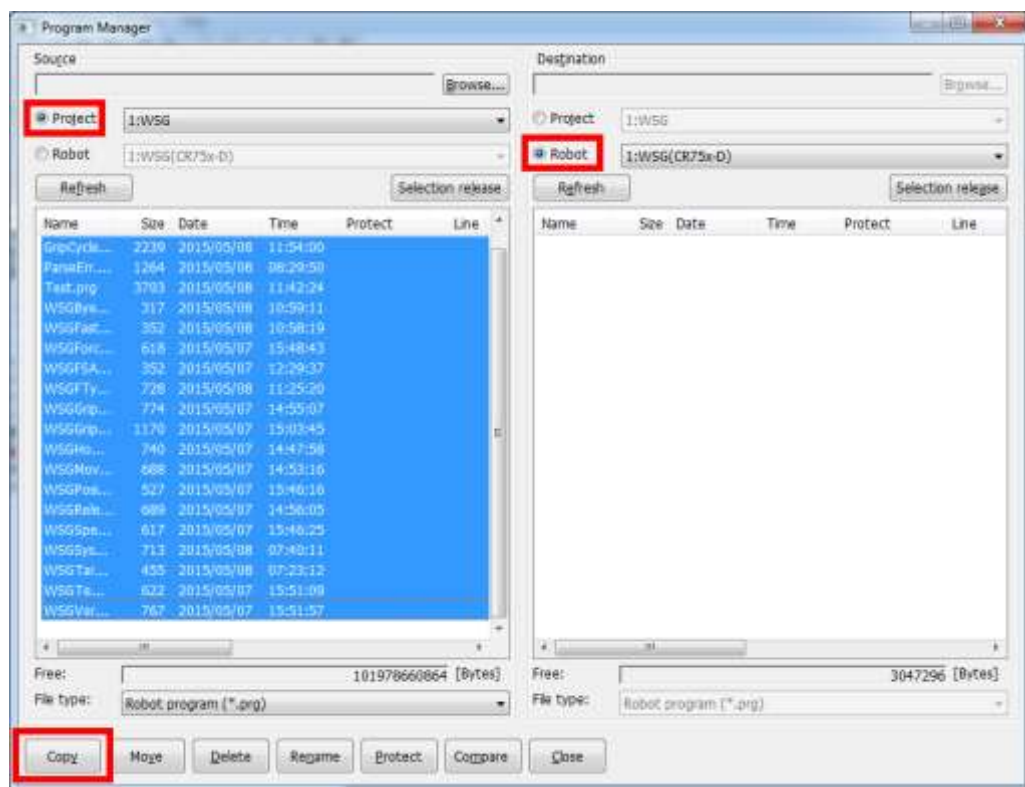


Figure 9: RT ToolBox2 Program Manager



Figure 10: „Operation panel“

## 4 Robot program

### 4.1 Main program „GripCycle“

The program “GripCycle” implements a simple gripping cycle that consists of pre-positioning the gripper fingers, gripping and releasing a part. It connects to the gripper using the connection parameters set in chapter 2 and first of all checks the type of the connected gripper.

Depending on the gripper type, the program then checks whether there are any sensor fingers of type “WSG-FMF” connected to measure the force<sup>1</sup>. If so, the program tares these sensor fingers.

After checking the gripper state (which must be set to “Idle”), a homing sequence is executed before the program starts the gripping cycle: Pre-position the fingers to an opening width of 40 mm, grip a part at 20 mm with a force of 20 N, release the part by opening the fingers relative to the current position by 20 mm. For all motion related commands, a speed of 100 mm/s is set.

After the gripping cycle has finished, the program sends a *BYE* message to the gripper and closes the connection. If the robot control is set to “automatic” mode, the program will start again from the beginning.


### 4.2 Main program „Test“

This program executes all the sub programs used to communicate with the gripper (starting with prefix “WSG...”) and checks the returned parameters. It is intended to test these sub programs. Please check the source code and integrated comments for any details.

### 4.3 Sub programs providing the gripper functions

The sub programs described in this chapter map the most important commands of the “Gripper Control Language” (GCL) to the robot control. Altogether, they implement a simple and easy to use interface that provides the gripper’s functions to the robot programmer.

Further information about syntax and semantics of the GCL commands can be found in the “WSG GCL Reference Manual” which is supplied with the gripper.

 **The sub programs described in this chapter only implement a transparent communication mechanism to and from the gripper. This means especially that any error handling is left to the programmer. The functions only provide error states in some global variables which have to be checked and processed regularly to make sure any errors during the gripping process are caught and the system runs stable. It is up to the programmer to implement proper error handling.**

---

<sup>1</sup> Not all types of WSG grippers offer a sensor port which is why this part is skipped for some types.

#### 4.3.1 WSGBye

This sub routine maps the GCL command *BYE* into the robot control and is used to signal a disconnect to the gripper. It must be called before any termination of the connection to make sure the gripper doesn't fall into a Fast Stop state which then must be committed before executing any further motion related commands.

##### **Parameters**

*None*

##### **Return value**

*None*

#### 4.3.2 WSGVerbose

This sub routine maps the GCL command *VERBOSE* into the robot control and is used to enable extended error messages on the GCL command interface. If active, all error messages are submitted with an additional text string describing the error. Please refer to the "WSG GCL Reference Manual" for any further information.

##### **Parameters**

*MEnable%* - indicates whether verbose mode should be enabled (1) or disabled (0)

##### **Return value**

In case of an error, the returned status code is written to the global variable *M\_StatusCode*.

#### 4.3.3 WSGDevType

This sub routine maps the GCL command *DEVTYPE* into the robot control and is used to query the device type of the connected gripper. It can be used to distinguish between the different types of WSG grippers during operation.

##### **Parameters**

*None*

##### **Return value**

The returned device type string will be saved to the global variable *C\_DevType*. In case of an error, the returned status code is written to the global variable *M\_StatusCode* and the value of *C\_DevType* is undefined.

#### 4.3.4 WSGHome

This sub routine maps the GCL command *HOME* into the robot control and is used to reference the position of the gripper's fingers. Homing should always be performed in the direction in which a higher accuracy of the finger position is needed, i. e. into the same direction as the gripping is performed later. This guarantees a most accurate finger position accuracy.

##### **Parameters**

*MDirection%*    *Indicates in which direction the homing sequence should be performed.*  
*0: inner homing, 1: outer homing*

##### **Return value**

In case of an error, the returned status code is written to the global variable *M\_StatusCode*.

#### 4.3.5 WSGMove

This sub routine maps the GCL command *MOVE* into the robot control and is used to pre-position the gripper fingers prior to any gripping command. This command is intended only for positioning the fingers. Blocking the fingers during motion will lead to an error. To grip and release parts, the appropriate GCL commands *GRIP* and *RELEASE* must be used (cf. sub programs "WSGGrip", cf. chapter 4.3.6 and "WSGRelease", cf. chapter 4.3.7).

##### **Parameters**

*MWidth*            *Target position in mm*  
*MSpeed*           *Speed in mm/s*

##### **Return value**

In case of an error, the returned status code is written to the global variable *M\_StatusCode*.

#### 4.3.6 WSGGrip

This sub routine maps the GCL command *GRIP* into the robot control and is used to grip a part with the given width, speed and force.

Please note that the command returns an error (status code 18 – E\_CMD\_FAILED), if no part is found that can be gripped.

##### **Parameter**

*MForce*            *Gripping force in N*  
*MWidth*           *Nominal part width of the part to be gripped in mm*  
*MSpeed*           *Gripping speed in mm/s*

##### **Return value**

In case of an error, the returned status code is written to the global variable *M\_StatusCode*.

#### 4.3.7 WSGRelease

This sub routine maps the GCL command *RELEASE* into the robot control and is used to release a part previously gripped.

##### **Parameter**

*MDist*            *Open width relative to the current position in mm*  
*MSpeed*        *Open speed in mm/s*

##### **Return value**

In case of an error, the returned status code is written to the global variable *M\_StatusCode*.

#### 4.3.8 WSGSysFlag

This sub routine maps a subset of the GCL command *SYSFLAGS* into the robot control and is used to query the system flag with the given index from the gripper. An overview of all available system flags and their meaning can be found in the appendix of the “WSG GCL Reference Manual” which is supplied with the gripper.

##### **Parameters**

*MIndex%*        *Index of the system flag to be checked*

##### **Return value**

The state of the queried system flag will be saved to the global variable *M\_SysFlag*. In case of an error, the returned status code is written to the global variable *M\_StatusCode* and the value of *M\_SysFlag* will be set to 0.

#### 4.3.9 WSGGripState

This sub routine maps the GCL command *GRIPSTATE* into the robot control and is used to query the gripper state. An overview of the available gripper states and their meaning can be found in the appendix of the “WSG GCL Reference Manual” which is supplied with the gripper.

##### **Parameters**

*None*

##### **Return value**

The queried gripper state will be saved to the global variable *M\_GripStateVal* as a numerical value as well as in the global variable *C\_GripStateStr* as a string value.

In case of an error, the returned status code is written to the global variable *M\_StatusCode*.

#### 4.3.10 WSGPosition

This sub routine maps the GCL command *POSITION* into the robot control and is used to query the current finger opening width in mm.

##### **Parameters**

*None*

##### **Return value**

The returned opening width will be saved to the global variable *M\_FingerPos*. In case of an error, the returned status code is written to the global variable *M\_StatusCode* and the value of *M\_FingerPos* will be set to 0.

#### 4.3.11 WSGSpeed

This sub routine maps the GCL command *SPEED* into the robot control and is used to query the current finger speed in mm/s.

##### **Parameters**

*None*

##### **Return value**

The returned speed value will be saved to the global variable *M\_FingerSpeed*. In case of an error, the returned status code is written to the global variable *M\_StatusCode* and the value of *M\_FingerSpeed* will be set to 0.

#### 4.3.12 WSGForce

This sub routine maps the GCL command *FORCE* into the robot control and is used to query the current gripping force in N.

##### **Parameters**

*None*

##### **Return value**

The returned force value will be saved to the global variable *M\_FingerForce*. In case of an error, the returned status code is written to the global variable *M\_StatusCode* and the value of *M\_FingerForce* will be set to 0.

#### 4.3.13 WSGTemp

This sub routine maps the GCL command *TEMP* into the robot control and is used to query the current temperature in °C measured by the temperature sensor on the controller board inside the gripper case..

##### **Parameters**

*None*

##### **Return value**

The returned temperature value will be saved to the global variable *M\_Temperature*. In case of an error, the returned status code is written to the global variable *M\_StatusCode* and the value of *M\_Temperature* will be set to 0.

#### 4.3.14 WSGFastStop

This sub routine maps the GCL command *FASTSTOP* into the robot control and is used to trigger a Fast Stop on the gripper module which must be acknowledged (cf. *WSGFSAck*, chapter 4.3.15) prior to executing any further motion related commands.

##### **Parameters**

*None*

##### **Return value**

In case of an error, the returned status code is written to the global variable *M\_StatusCode*.

#### 4.3.15 WSGFSAck

This sub routine maps the GLC command *FSACK* into the robot control and is used to acknowledge a Fast Stop previously triggered by the *FASTSTOP* command (cf. *WSGFastStop*, chapter 4.3.14) or raised by the gripper itself.

##### **Parameters**


*None*

##### **Return value**

In case of an error, the returned status code is written to the global variable *M\_StatusCode*.

#### 4.3.16 WSGFType

This sub routine is used to map the GCL command *FTYPE* into the robot control and is used to determine the type of the optional sensor finger mounted onto the gripper. Especially, it can be used to detect optional force measurement fingers of type WSG-FMF.

 This command is only available on WSG grippers that provide a sensor port on which active sensor fingers can be mounted. Please refer to the user's manual for any further information.

**Parameters**

*MIndex%*                      *Finger index (0 or 1).*

**Return value**

The returned value will be saved to the global variable *C\_FingerTypeStr*. In case of an error, the returned status code is written to the global variable *M\_StatusCode* and the value of *C\_FingerTypeStr* is undefined.

### 4.3.17 WSGTare

This sub routine is used to map the GCL command *TARE* into the robot control and is used to tare optional sensor fingers of type WSG-FMF that are mounted onto the gripper.

**Parameters**

*MIndex%*                      *Finger index (0 to tare finger 1, 1 to tare finger 2).*

**Return value**

In case of an error, the returned status code is written to the global variable *M\_StatusCode*.

## 4.4 Helper Programs

### 4.4.1 ParseErr

This sub program is used by the “WSG...” sub programs to parse error messages and to save the returned status code into a global variable which then in turn can be checked by the programmer to handle any errors.  
and describes their reason.



[www.weiss-robotics.com](http://www.weiss-robotics.com)

© Weiss Robotics GmbH & Co. KG. All rights reserved.

The technical data mentioned in this document can be changed to improve our products without prior notice. Used trademarks are the property of their respective trademark owners. Our products are not intended for use in life support systems or systems whose failure can lead to personal injury.