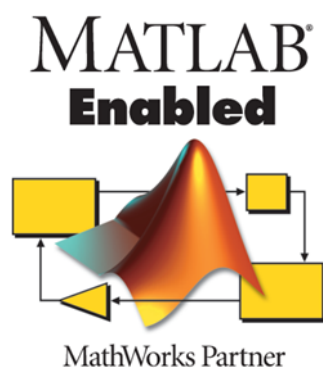


SM155E
PI MATLAB Driver GCS 2.0 (PI_MATLAB_Driver_GCS2)
Software Manual

Version: 1.1.0

Date: February 29, 2016



Physik Instrumente (PI) GmbH & Co. KG is the owner of the following trademarks:

PI®, PIC®, PICMA®, PILine®, PiezoWalk®, NEXACT®, NEXLINE®, NanoCube®, Picoactuator®, PInano®, PIMag®

The following designations are protected company names or registered trademarks of third parties:

Microsoft, Windows, LabVIEW, MathWorks, MATLAB

Software products that are provided by PI are subject to the General Software License Agreement of Physik Instrumente (PI) GmbH & Co. KG and may incorporate and/or make use of third-party software components.

For more information, please read the General Software License Agreement and the Third Party Software Note linked below.

[General Software License Agreement](#)

[Third Party Software Note](#)

PI owns the following patents or patent applications for the technology field Piezo Stepping Drive (PiezoWalk®, NEXACT®, NEXLINE®):

DE10148267B4, EP1267478B1, EP2209202B1, EP2209203B1, US6800984B2, DE4408618B4

PI owns the following patents or patent applications for the technology field Multilayer Piezo Actuators (PICMA®):

DE10021919C2, DE10234787C1, ZL03813218.4, EP1512183A2, JP4667863, US7449077B2

PI owns the following patents or patent applications for the technology field Ultrasonic Piezo Motors (PILine®):

Germany: DE102004024656A1, DE102004044184B4, DE102004059429B4, DE102005010073A1, DE102005039357B4, DE102005039358A1, DE102006041017B4, DE102008012992A1, DE102008023478A1, DE102008058484A1, DE102010022812A1, DE102010047280A1, DE102010055848, DE102011075985A1, DE102011082200A1, DE102011087542B3, DE102011087542B3, DE102011087801B4, DE102011108175, DE102012201863B3, DE19522072C1, DE19938954A1

Europe: EP0789937B1EP1210759B1, EP1267425B1, EP1581992B1, EP1656705B1, EP1747594B1, EP1812975B1, EP1861740B1, EP1915787B2, EP1938397B1, EP2095441B1, EP2130236B1, EP2153476B1, EP2164120B1, EP2258004B1, EP2608286A2

USA: US2010/0013353A1, US5872418A, US6765335B2, US6806620B1, US6806620B1, US7218031B2, US7598656B2, US7737605B2, US7795782B2, US7834518B2, US7973451B2, US8253304B2, US8344592B2, US8482185B2

Japan: JP2011514131, JP2011522506, JP3804973B2, JP4377956, JP4435695, JP4477069, JP4598128, JP4617359, JP4620115, JP4648391, JP4860862, JP4914895, JP2013539346

China: ZL200380108542.0, ZL200580015994.3, ZL200580029560.9, ZL200580036995.6, ZL200680007223.4, ZL200680030007.1, ZL200680042853.5

International patent applications: WO2009059939A2, WO2010121594A1, WO2012048691A2, WO2012113394A1, WO2012155903A1, WO2013034146A3, WO2013117189A2

PI owns the following patents or patent applications for the technology field Magnetic Direct Drives (PIMag®):
WO212146709A2, DE102012207082A1

PI owns the following patents or patent applications for the technology field Piezo Inertia Drives (PIShift, PiezoMike):

EP2590315A1, WO213064513A1, DE102011109590A1, WO2013020873A1

© 2015-2016 Physik Instrumente (PI) GmbH & Co. KG, Karlsruhe, Germany. The text, photographs, and drawings in this manual are protected by copyright. With regard thereto, Physik Instrumente (PI) GmbH & Co. KG retains all the rights. Use of said text, photographs, and drawings is permitted only in part and only upon citation of the source.

Original instructions

First printing: February 29, 2016

Document number: SM155E, SMoe, BRo

Version: 1.1.0

PI_MATLAB_Driver_GCS2_Manual_SM155E.docx

Subject to change without notice. This manual is superseded by any new release. The newest release is available for download at www.pi.ws.

Contents

1	Prerequisites	6
1.1	To Which MATLAB Driver Version Does This Documentation Apply?	6
1.2	Which Operating Systems Does the PI MATLAB Driver GCS2 Support?	6
1.3	Which PI Controllers Does the PI MATLAB Driver GCS2 Support?	6
2	Introduction to the PI MATLAB Driver GCS2	7
3	Installation	8
3.1	Standard Installation (Recommended)	8
3.2	User-Defined Installation	8
3.3	Installation Location	8
4	Basic Use of the MATLAB Driver	9
4.1	Start Sample: “BeginWithThisSampleWhenUsing_<ControllerName>.m”	9
4.2	Configuring and Referencing	13
4.3	Basic Controller Functions	13
4.4	Closing the Connection	14
4.5	Unloading the Driver	14
4.6	Using the Stage	14
5	Using the Controller with GCS Commands	15
5.1	Communicating with the Controller Using GCS2 Commands	15
5.2	Operating Principle behind the MATLAB Driver	16
6	Further Samples	21
6.1	Further Samples – Functions and Scripts	21
6.2	Operating Multiple Controllers	22
6.3	Daisy Chain	22
7	Tips & Tricks	24
7.1	Logging GCS2 Commands in PIMikroMove During Initialization	24
7.2	Creating Shortcuts in MATLAB	25
7.3	Using the Command Window in MATLAB	26

8	Troubleshooting	27
8.1	The MATLAB Script Generates Error Messages When Started on Another PC	27
8.2	Error when Loading the PI GCS2 DLL	27
8.3	MATLAB Crashes Unexpectedly	32
8.4	“Error using callib”	34
9	PI MATLAB Driver GCS2: Changing to Version 1.2.0	35
9.1	Changing over to Version 1.2.0	35
9.2	Notes on the Changeover	36
9.3	Comparison of Old and New Structure	38

1 Prerequisites

1.1 To Which MATLAB Driver Version Does This Documentation Apply?

This documentation applies to version 1.2.0 of PI MATLAB Driver GCS2. You can find the version number in the source code of the PI MATLAB Driver GCS2 or via the PI Update Finder. If the PI Update Finder is unable to find the PI MATLAB Driver GCS2, the driver is not installed or there is an older version on your system (for more information, see Chapter 9 – “PI MATLAB Driver GCS2: Changing to Version 1.2.0”).

When the PI MATLAB Driver GCS2 is successfully loaded in MATLAB, you can determine the version number by using the following command:

```
>> Controller.GetVersionNumber()  
  
ans =  
PI MATLAB Driver GCS2 Version Number: 1.2.0
```

This function is not implemented in older versions of the PI MATLAB Driver GCS2. MATLAB will therefore return an error instead of the version number.

1.2 Which Operating Systems Does the PI MATLAB Driver GCS2 Support?

Windows

All versions from WinXP and up. The PI MATLAB Driver GCS2 is installed and configured on your system by an installer.

1.3 Which PI Controllers Does the PI MATLAB Driver GCS2 Support?

The PI MATLAB Driver GCS2 supports all PI controllers that can be addressed via the GCS2 protocol. This protocol is implemented in almost all recent PI controllers.

2 Introduction to the PI MATLAB Driver GCS2

The “PI MATLAB Driver GCS2” – hereinafter always referred to as the “MATLAB driver” – is the MATLAB interface for operating most PI controllers (see Section 1.3 – “Which PI Controllers Does the PI MATLAB Driver GCS2 Support?”).

The MATLAB driver is based on the GCS2 protocol from PI¹, which can be used to operate almost all PI controllers. This protocol consists of mnemonics that are generally 3 letters long, e.g. MOV (for “move”), FRF (for “find reference”), or POS? (for “query position”). These mnemonics are combined with parameters and sent to the controller. The following command moves the stage on axis 1 of the controller to position 31 μm :

MOV 1 31

This command is sent to the controller using ASCII symbols. Using the MATLAB driver, the same command is issued for the (fictional) E-042 controller, for example, with the following MATLAB command:

```
E042.MOV ( '1' , 31 );
```

For this command to work in MATLAB, you must install the MATLAB driver, load the MATLAB driver in MATLAB, and establish a connection to the controller. How this works is described in Chapter 3 – “Installation” and Chapter 4 – “Basic Use of the MATLAB Driver”. Using GCS commands in MATLAB is described in Chapter 5 – “Using the Controller with GCS Commands”, and frequent problem cases, such as changing over from a 32-bit system to a 64-bit system, are covered in Chapter 8 – “Troubleshooting”.

¹ The protocol is based on ASCII symbols. You can find a detailed description of this protocol in the “GCS Commands” section of the user manual for your controller.

3 Installation

3.1 Standard Installation (Recommended)

We recommend performing a complete installation of the PI software. This installs the MATLAB driver and all the necessary utility software. The relevant installer is on your product CD and is called:

- PI_<ControllerName >.CD_Setup.exe

Then perform a software update using the PI Update Finder (particularly important for older product CDs).

The MATLAB driver is not included in the installation on product CDs that were released before 2015. How to download and install it is described in Chapter 9.1 – “Changing over to Version 1.2.0”.

When the MATLAB driver is successfully installed, we recommend first activating the controller and the stage with PIMikroMove to ensure that the entire system is connected correctly and operational.

3.2 User-Defined Installation

If you do not want to perform the complete installation, you must execute the three installers on the product CD listed below as a minimum. However, not performing the complete installation means that diagnostics, configuration, and update tools will not be installed.

- PI_MATLAB_Driver_GCS2_Setup.exe
- PI_GCS_Library_PI_GCS2_DLL_Setup.exe
- PI_Stage_Database_PIStages2_Setup.exe (only if on product CD)

If you no longer have access to the product CD, contact your local sales partner or support-software@pi.ws.

3.3 Installation Location

Performing the installation saves the MATLAB driver in the following directory:

- Win7/8, Vista: C:\Users\Public\PI\PI_MATLAB_Driver_GCS2
- WinXP: C:\Documents and Settings\All Users\PI\PI_MATLAB_Driver_GCS2

4 Basic Use of the MATLAB Driver

4.1 Start Sample: “BeginWithThisSampleWhenUsing_<ControllerName>.m”

A sample is used to demonstrate the use of the MATLAB driver. You can find this sample on your product CD. In the standard installation, the samples are saved in the following location on your PC:

- Win7/8 und Vista: C:\Users\Public\PI\<ControllerName>\Samples
- WinXP: C:\Documents and Settings\All Users\PI\<ControllerName>\Samples

4.1.1 Loading the MATLAB Driver

```
%% Load PI MATLAB Driver GCS2

if (strfind(evalc('ver'), 'Windows XP'))
    addpath('C:\Documents and Settings\All
Users\PI\PI_MATLAB_Driver_GCS2_Setup.exe');
elseif (strfind(evalc('ver'), 'Windows'))
    addpath('C:\Users\Public\PI\PI_MATLAB_Driver_GCS2_Setup.exe');
end

if(~exist('Controller','var'))
    Controller = PI_GCS_Controller();
end;
if(~isa(Controller,'PI_GCS_Controller'))
    Controller = PI_GCS_Controller();
end;
```

In this section, the MATLAB driver is loaded. Firstly, the driver path is added to the MATLAB search path. The MATLAB driver is then loaded using `Controller = PI_GCS_Controller();`. The MATLAB driver can then be accessed via the “Controller” driver object in the script.

4.1.2 Configuring the Stage and the Connection

```
%% List connected USB and TCP/IP controller

devicesUsb = Controller.EnumerateUSBAsArray();
devicesTcpIp = Controller.EnumerateTCPIPDevicesAsArray();
```

Using the Enumerate function of the MATLAB driver, you can display all powered-on PI devices that are connected to your PC via USB or a network (RS-232 and other connections do not have this functionality).

In the following section, you must configure two settings yourself to make the sample executable.

Setting 1 – Stage type (only applies to C controllers²)

Indicate the type (not the serial number) of the connected stage, e.g. “M-664.164”. You can find the name of the stage on the type plate of the stage. ATTENTION! Ensure that you enter the correct stage type. An incorrect stage type can result in an incorrect operation of the stage by the controller and can damage or destroy the stage.

Setting 2 – Connection parameters

As the second setting, you must activate and then configure a certain connection. For a connection to be made, the following parameters must be set correctly:

- **RS-232 connection:** COM port and baud rate
You can find out which COM port is used in the device manager or via PIMikroMove in “Start up Controller”. You can find the baud rate in the manual for the controller. The baud rate for some controllers can be manually set via DIP switches on the controller.
- **USB connection:** Serial number of the controller
You can find the serial number on the type plate of the controller or by using the function `Controller.EnumerateUSBAsArray()`.
- **TCP/IP connection:** IP number and port number
You can find out the IP and port numbers with PIMikroMove or by using the function `Controller.EnumerateTCPIPDevicesAsArray()`. The port number for PI controllers is usually “50000”. Should problems occur during the TCP/IP connection, contact your network administrator.

```
%% Parameters
% You MUST EDIT AND ACTIVATE the parameters to make your system run
properly.
% 1. Set the correct stage type. AN INCORRECT STAGE TYPE CAN DAMAGE YOUR
STAGE!
% 2. Activate the connection type you want to use.
% 3. Set the connection settings.
```

²A CST is also possible for some E controllers – such as E-712 – in combination with certain stage types. Refer to the controller manual.

```
% Set the correct stage type. AN INCORRECT STAGE TYPE CAN DAMAGE YOUR
STAGE!
stageType = 'M-664.164';

% Activate connection settings
use_RS232_Connection    = false;
use_USB_Connection      = true;
use_TCPIP_Connection    = false;

% Set connection settings
if (use_RS232_Connection)
    comPort = 1;
    baudRate = 115200;
end

if (use_USB_Connection)
    controllerSerialNumber = '123456789';
end

if (use_TCPIP_Connection)
    ip    = '217.243.255.19';
    port = 50000;
end
```

If you have configured the settings in this section correctly and selected the appropriate sample for your controller, your sample should now be executable without errors.

In the example above, during the MATLAB script, your PC would try to make a connection to the controller via USB with the serial number 123456789. When the connection is established successfully, the controller would be told that a stage of type “M-664.164” is connected to it.

4.1.3 Establishing a Connection

```
%% Start connection

boolE042connected = false;

if (exist('E042','var'))
    if (E042.IsConnected)
        boolE042connected = true;
    end
end

if (~boolE042connected)
    if (use_RS232_Connection)
        E042 = Controller.ConnectRS232(comPort, baudRate);
    end
end
```

```

    if (use_USB_Connection)
        E042 = Controller.ConnectUSB(controllerSerialNumber);
    end

    if (use_TCPIP_Connection)
        E042 = Controller.ConnectTCPIP(ip, port);
    end
end

% query controller identification
E042.qIDN()

% initialize controller
E042 = E042.InitializeController();

```

In this section, a connection to a specific controller is established. This is described using the fictional controller E-042.

To be able to address the E-042 controller, the connection must first be established using the MATLAB driver. The connection is saved in the new “E042” controller object:

```
E042 = Controller.ConnectXXX(...)
```

All further accesses to E-042 then occur via the E042 object.

`ConnectXXX` is just a placeholder in the example for the actual connection type (RS-232, USB, TCP/IP, ...). The connection functions are presented below.

Should a connection fail, first try to establish a connection with PIMikroMove. If the connection works in PIMikroMove, you have probably just chosen an incorrect configuration of the communication interface.

Also ensure that no other program (PITerminal, PIMikroMove, etc.) is connected to the controller at the same time if you establish a connection to the controller from MATLAB.

RS-232

```
E042 = Controller.ConnectRS232(comPort, baudRate)
```

USB

```
E042 = Controller.ConnectUSB(controllerSerialNumber);
```

TCP/IP

```
E042 = Controller.ConnectTCPIP(IP, port);
```

Additional connection options

Additional connection options are

- Simultaneous connection to multiple controllers
(see Section 6.2 – “Operating Multiple Controllers”)
- Daisy chain
(see Section 6.3 – “Daisy Chain”)
- NI GPIB

Function: `ConnectNIgpiB`

Checking whether connection is successful

If the connection was established successfully, the name of the connected controller is displayed in the MATLAB Command Window when you execute the follow command:

```
E042.qIDN( )
```

4.2 Configuring and Referencing

Generally, you then configure the controller and subsequently reference the stage. This procedure can vary considerably depending on the controller and stage. The corresponding commands are given in the specific sample for your controller:

“BeginWithThisSampleWhenUsing_<ControllerName>.m”

If you do not have a specific sample for your controller, you can look up the necessary initialization commands in the manual for your controller. In principle, understanding initialization and referencing can be advantageous because you can skip individual steps if necessary, which means that your MATLAB script can run more quickly.

Another easy way of finding the corresponding commands for configuration and referencing is by logging the GCS commands in PIMikroMove via the Log Window. This is described in Section 7.1 – “Logging GCS2 Commands in PIMikroMove During Initialization”.

4.3 Basic Controller Functions

```
%% Basic controller functions
dMin = E042.qTMN('1');
dMax = E042.qTMX('1');

position = rand(1)*(dMax-dMin)+dMin;
E042.MOV('1', position);
% wait for motion to stop
while(0 ~= E042.IsMoving('1'))
    pause(0.1);
end
```

```
E042.qPOS(axisname)
```

`E042.MOV('1', position);` is used to command a position. If the MATLAB script is only supposed to continue when the target position is reached, check for `E042.IsMoving('1')`.

4.4 Closing the Connection

The following command is used to close the connection to the E-042 controller:

```
E042.CloseConnection;
```

Closing the connection means that the controller can be addressed from other programs (e.g. PIMikroMove) again. If necessary, the connection can be re-established using `E042 = Controller.ConnectUSB(controllerSerialNumber);`.

4.5 Unloading the Driver

The following command is used to unload the driver:

```
Controller.Destroy;
clear Controller;
```

This automatically closes all connections with the controllers. All controller objects lose the ability to communicate with the controllers. Therefore, only unload the driver when your script has ended completely.

Since unloading the MATLAB driver causes the controller object to lose its function, it also makes sense to clear it as well: `clear E042;`

Closing MATLAB also unloads the driver.

4.6 Using the Stage

If all previous steps have been completed successfully, you can use almost all available GCS commands via the MATLAB driver³. How you find the corresponding GCS command and how you need to activate it is described in the following chapter.

³ Should a GCS command not be implemented in PI_MATLAB_Driver_GCS2, you may not have the most up-to-date version of MATLAB GCS. Launch the PIUpdateFinder to obtain the most recent version of the MATLAB driver. Note that not all controllers implement all GCS commands.

5 Using the Controller with GCS Commands

5.1 Communicating with the Controller Using GCS2 Commands

As already described in Chapter 2 – “Introduction to the PI MATLAB Driver GCS2”, the MATLAB driver is based on the PI GCS2 protocol, which can be used to address almost all PI controllers.

The MATLAB driver implements most GCS2 commands. Most PI controllers, however, can only implement some of the GCS2 commands. The controller manual contains information about which commands your controller knows and how they function. You can also query a list of the GCS2 commands that the controller knows directly via the controller object using the following command⁴:

```
E042.qHLP( )
```

Integrated help function

There is currently only a help function integrated in MATLAB for a few functions (e.g. for the function `EnumerateUSBAsArray`). To open the help for the MATLAB driver function, highlight the function name and press F1 or use the context menu (see Figure 1). For this to work, the path of the MATLAB driver must be set using `addpath(...)`, which occurs in the section `%% Load PI MATLAB Driver GCS2`. We intend to gradually add help for all functions in future versions.

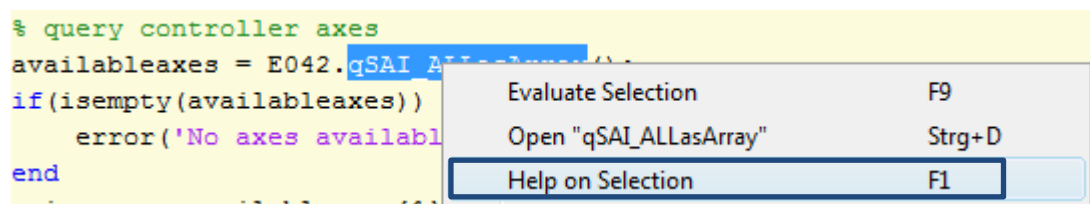


Figure 1 Opening the help for a controller object function, e.g. “qSAI_ALLasArray” (currently only available for a few functions)

⁴ This query delivers the GCS2 commands, not the function names of the controller object. Refer to Table 1 – “Differences and similarities between the C interface in PI GCS2 DLL and MATLAB driver”.

At the moment, most functions are still undocumented. Therefore, a workaround is currently required to apply all GCS commands correctly and to transmit the input and output parameters to the function correctly. This is described in the following sections.

5.2 Operating Principle behind the MATLAB Driver

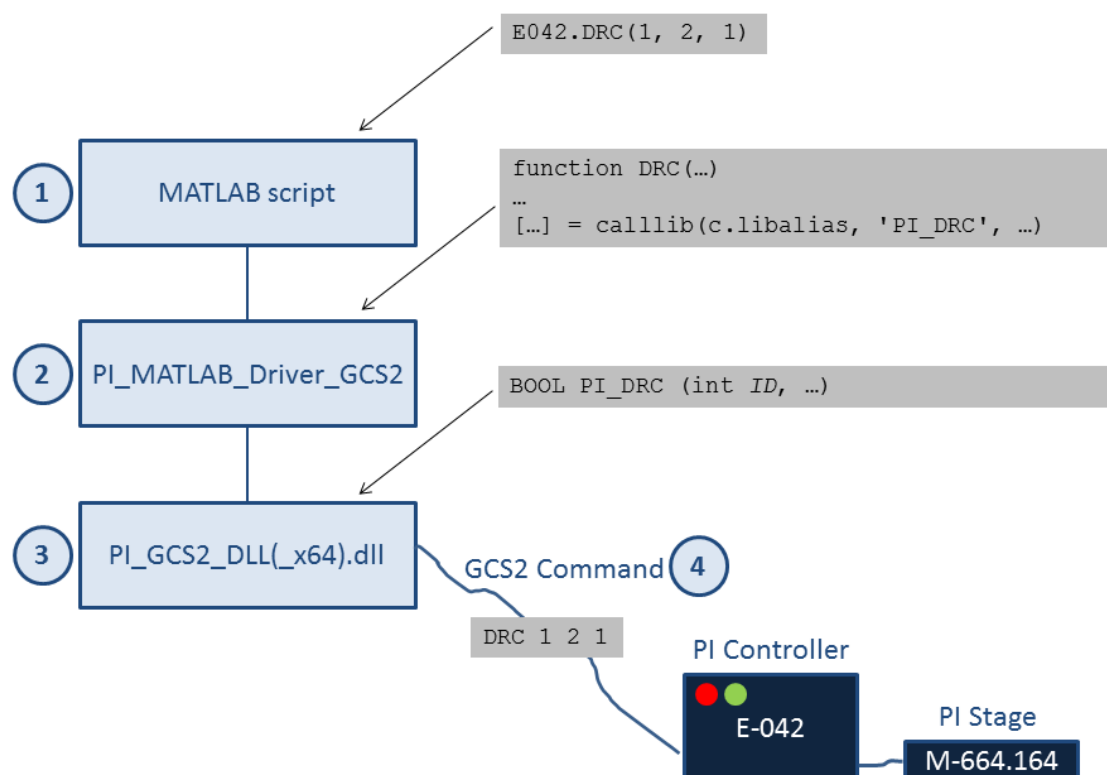


Figure 2 A command is issued in the MATLAB script (1).
The MATLAB driver activates the PI GCS2 DLL using the callib function (2 and 3).
The PI GCS2 DLL sends a GCS command from the PC to the PI controller (4).

The MATLAB driver uses the file “PI_GCS2_DLL.dll”⁵. This is a dynamic program library (hereinafter abbreviated as “DLL”), which, among other things, implements communication via the GCS2 protocol. This DLL has an interface for the C programming language and this C interface of the “PI_GCS2_DLL.dll” is used by the MATLAB driver.

⁵ This applies to 32-bit systems. For 64-bit systems, PI_GCS2_DLL_x64.dll is used.

All of the communication commands for this C interface are documented in the document “PIGCS_2_0_DLL_pdf”. Using this document, you can understand the operating principles behind all MATLAB driver functions.

If, for example, you want to understand how the command `E042.DRC(1, 2, 1)` works exactly and which parameters are expected and returned, search the PDF for “DRC”. There you will find the following interface description:

```
BOOL PI_DRC (int ID, const int* piRecordTableIdsArray, const char*
szRecordSourceIds, const int* piRecordOptionsArray)
```

Corresponding command: DRC

Set data recorder configuration: determines the data source (*szRecordSourceIdsArray*) and the kind of data (*piRecordOptionsArray*) used for the given data recorder table.

Arguments:

ID ID of controller

piRecordTableIdsArray ID of the record table

szRecordSourceIds ID of the record source, for example axis number or channel number. The value of this argument depends on the corresponding record option.

piRecordOptionsArray record option, i.e. the kind of data to be recorded

Returns:

TRUE if no error, **FALSE** otherwise (see p. 7)

Figure 3 Description of the C interface of the DLL in “PIGCS_2_0_DLL_pdf”

Do not be put off that it is a description for C programming language. It is relatively simple to extrapolate the interface description in MATLAB from the interface description of the PI GCS2 DLL. The following table provides a detailed description of how this works.

Table 1 Differences and similarities between the C interface in PI GCS2 DLL and MATLAB driver

Description	C Interface in the PI GCS2 DLL	MATLAB
Function name The prefix "PI_" is omitted in MATLAB.	<code>PI_XXX(...)</code> Example: <code>PI_DRC(...)</code>	<code>XXX</code> Example: <code>DRC(...)</code>
Controller ID In MATLAB, the name of the controller object rather than an ID is used to differentiate between different controllers.	Variable ID <code>PI_XXX(ID, ...)</code> Example: <code>PI_DRC (1, ...)</code> <code>PI_DRC (2, ...)</code>	Variable ID <code>ControllerName.XXX(...)</code> Example: <code>E042.DRC (...)</code> <code>C007.DRC (...)</code>
Data types for numbers There is only one data type for numbers in MATLAB.	<code>int, double, const double, const int*, ...</code>	<code>double</code>
Data type for strings There is only one data type for strings in MATLAB.	<code>char*, const char*, ...</code>	<code>char</code>
Return values In MATLAB, new values are returned directly. The DLL changes the values of the transmitted variables if they are prefixed by an asterisk but not by the keyword "const". Not changed by DLL: <code>const char* param1</code> Changed by DLL: <code>int* param2</code>	<code>PI_XXX(int ID, const char* param1, int* param2, double* param3)</code> Example: <code>BOOL PI_qPOS (int ID, const char* szAxes, double* pdValueArray)</code>	<code>[param2, param3] = E042.XXX(param1)</code> Example: <code>pdValueArray = E042.qPOS(szAxes)</code>

The following example shows a direct comparison of C interface and MATLAB:

MATLAB:

```
recordTableId = 1;
axisname = '1';
recordOption = 1;
Controller.DRC(recordTableId, axisname, recordOption);
```

C interface of the PI GCS2 DLL:

```
BOOL PI_DRC (int ID, const int* piRecordTableIdsArray, const char*
szRecordSourceIds, const int* piRecordOptionsArray)
```

5.2.1 Internal Operating Principle behind the MATLAB Driver

If you use a function with a large number of input and output parameters, the mapping between the C interface of the DLL and the MATLAB driver shown in Table 1 is insufficient. For such cases, the internal operating principle behind the MATLAB driver is addressed in this section.

All functions of the MATLAB driver have a similar structure. There are three sections that are also of interest to the user.

```
function dValues = qTMN(c,szAxes)
...
if(nargin==1), szAxes = '';
...
[ret,szAxes,dValues]=calllib(c.libalias,FunctionName,c.ID,szAxes,pdValues);
...
```

In line 1, the transfer parameters for the function are defined – the `szAxes` parameter in the example⁶. The return parameter of the function is `dValues`.

Where relevant, input parameters are optional. In the example, `szAxes` is optional. This is shown in the second section of the example. The logic here says: If “Number of ARGuments IN” (`nargin`) equals one, then a default parameter `szAxes = ''` is assigned.

In the third section, the GCS2 DLL is called up. Here `c.libalias` and `FunctionName` are required for the management of the PI GCS2 DLL used by the MATLAB driver. These two parameters are always present and can be ignored by the user. The specific controller is managed by `c.ID`. The `szAxes` and `pdValues` parameters are transfer parameters for the PI GCS2 DLL. The `c.ID`, `szAxes` and `pdValues` parameters exactly correspond to the function

⁶ The parameter `c` is required for the internal management of the PI_MATLAB_Driver_GCS2 and does not need to be transferred by the user.

call of the C interface of “PI_GCS2_DLL.dll”, which is documented in detail in the document “PIGCS_2_0_DLL_pdf.

The return parameter of the PI GCS2 DLL is `ret`. The `szAxes` and `dValues` parameters are parameters that have been changed by the DLL where necessary. The second parameter `dValues` is then passed on externally to the user as the return parameter for the function of the MATLAB driver.

5.2.2 Calling up the Source Code of a MATLAB Driver Function

To open the program code of a MATLAB driver function, highlight the function name and press Ctrl+D or use the context menu (see Figure 4). For this to work, the path of the MATLAB driver must be set using `addpath(...)`, which occurs in the section `%% Load PI MATLAB Driver GCS2.`

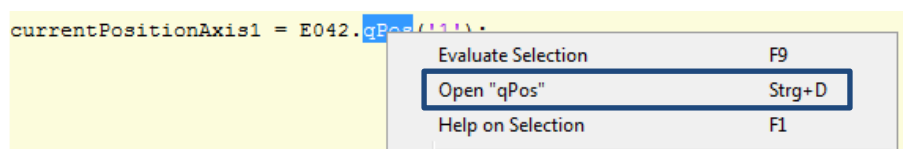


Figure 4 Opening the program code of an object function.

6 Further Samples

6.1 Further Samples – Functions and Scripts

You will find further samples in your Samples folder. These samples can be divided into two categories:

- Sample functions:
Identified by the prefix “**PI_**”, e.g. “PI_DataRecorder.m”
- Stand-alone scripts:
Identified by the prefix “**Sample_**”, e.g. “Sample_DaisyChain.m”

Sample functions

```
%% Call Function
relativeMove = -1; %in mm
PI_DataRecorder(E042, axisname, relativeMove);
```

Sample functions expect a controller object that is already connected to the controller. The controller must be correctly configured and referenced. Not all sample functions are suited to every stage. If necessary, use `E042.qHLP()` to check whether your controller supports all GCS2 functions used in the sample function.

Stand-alone sample scripts

Stand-alone scripts include the establishment of the connection. ATTENTION! The connection establishment is generally only shown as an example and you must adapt it to your system of one or several stages and controllers (for the configuration, see your start sample “BeginWithThisSampleWhenUsing_<ControllerName>.m” and the description in the manual for your controller).

6.2 Operating Multiple Controllers

It is very easy to operate multiple PI controllers using one MATLAB script. Instantiate two controller objects for this, which you initialize using the `ConnectXXX` function. In the example below, a connection to controllers E-871 and C-867 is established. E-871 is connected via RS-232 and C-867 via USB.

```
%% Parameters and Connection
...
comPort = 1;
baudRate = 115200;
E871 = Controller.ConnectRS232(comPort, baudRate);

controllerSerialNumber = '123456789';
C867 = Controller.ConnectUSB(controllerSerialNumber);
```

The connections can be disconnected one at a time:

```
%% Disconnect the controllers
C867.CloseConnection;
E871.CloseConnection;

%% Unload the PI MATLAB Driver GCS2
Controller.Destroy;
clear Controller;
clear C867;
clear E871;
```

6.3 Daisy Chain

With the MATLAB driver, you can also operate multiple PI controllers that are connected to one another by daisy chain⁷. For this purpose, first

`Controller = Controller.OpenUSBdaisyChain('0123456789');` is used to establish a connection with the controller which is connected directly to the PC. This controller is then addressed as the daisy chain controller with its daisy chain address:

```
Controller1=Controller.ConnectDaisyChainDevice(controller1_HardwareAddress);
```

From this point on, the controller can be accessed via the controller object `Controller1`. The connection to the second controller (which is connected to the first controller via RS-232) is established using the same command, only the daisy chain address is modified.

⁷ Currently, only the following connection type is supported: PC – Controller1 via USB.

Currently not supported: PC → Controller1 via RS-232. Should this need to be implemented, contact support-software@pi.ws.

```
%% Set up the USB communication
Controller = Controller.OpenUSBdaisyChain('0123456789');

%% Connect via daisy chain
controller1_HardwareAddress = 1;
controller2_HardwareAddress = 2;

Controller1=Controller.ConnectDaisyChainDevice(controller1_HardwareAddress);
Controller2=Controller.ConnectDaisyChainDevice(controller2_HardwareAddress);

Controller1.qIDN()
Controller2.qIDN()
```

The connections can be closed one at a time. However, a controller with closed connection is still able to pass commands on to other controllers. The unloading of the driver works in the same way as described in the previous examples.

```
%% Close the connection
Controller1.CloseConnection();
Controller2.CloseConnection();

%% Unload the PI MATLAB Driver GCS2
Controller.Destroy;
clear Controller;
clear Controller1
clear Controller2
```

7 Tips & Tricks

7.1 Logging GCS2 Commands in PIMikroMove During Initialization

In many cases, it can be helpful to log the GCS2 commands that PIMikroMove sends to the controller during initialization.

Step 1

Perform **steps 1 to 3** in the “Start up controller” window (see Figure 5) and then click “Close” without referencing the axes in **step 3** “Start up axes”.

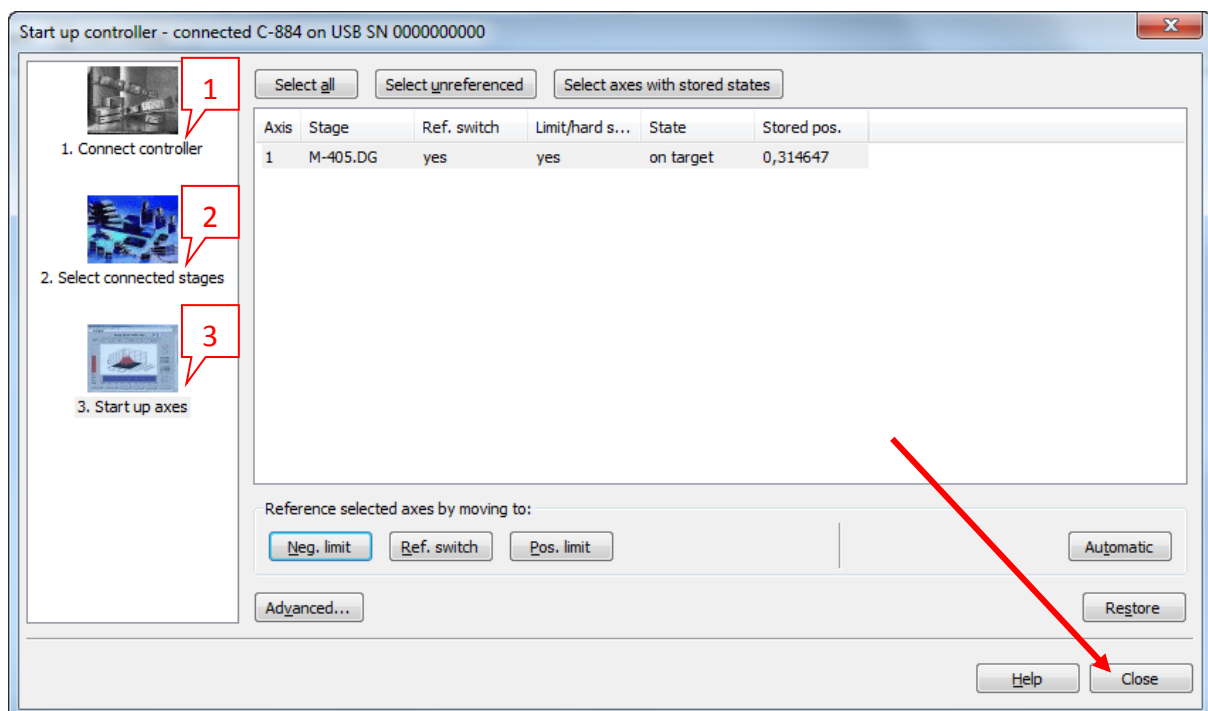


Figure 5 Start up controller

Step 2

Use the Log Window to log the GCS2 commands sent from the PC to the controller as well as the responses of the controller (see **step 1** in Figure 6). Then open “Start up axes” (see **step 2** in Figure 6). For additional information about logging, use the PIMikroMove help function, searching for the keyword “Log Window”.

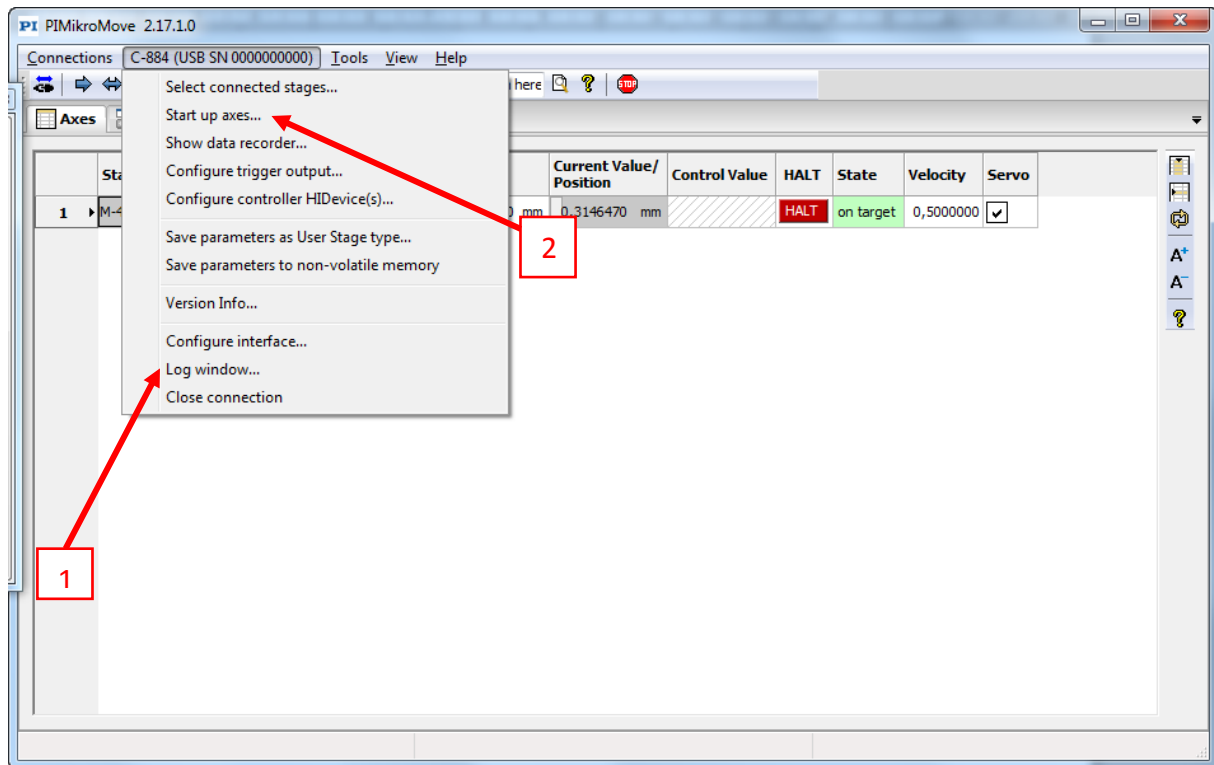


Figure 6 Start up axes

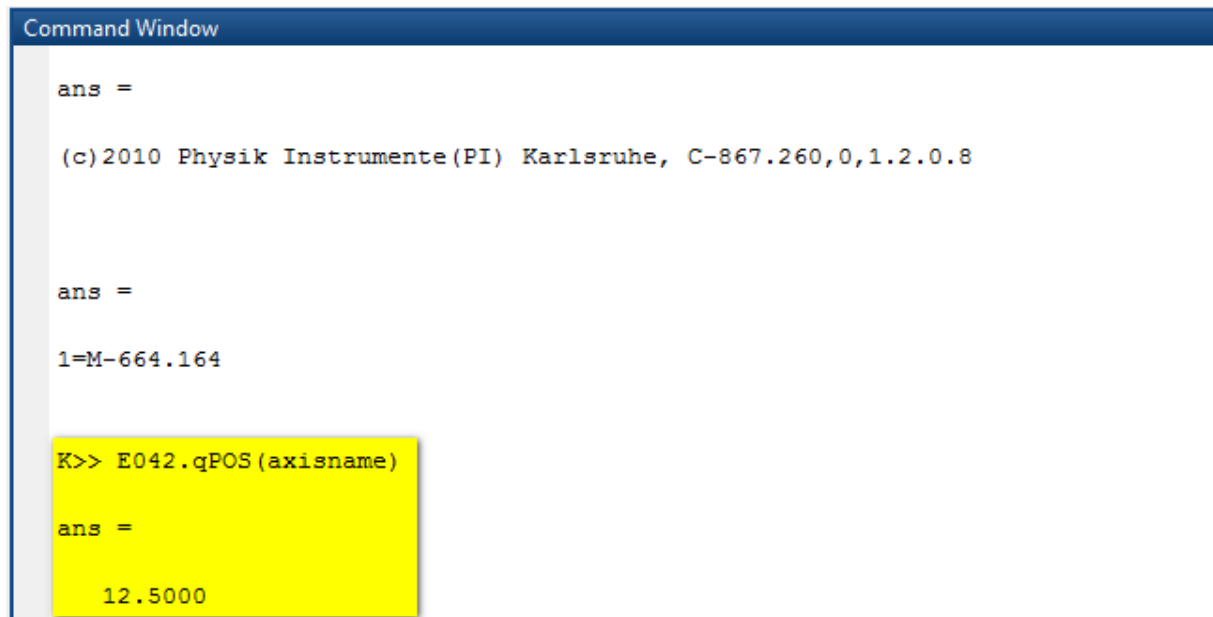
7.2 Creating Shortcuts in MATLAB

For frequently recurring tasks – like loading the driver or establishing/closing a connection – it can be helpful to create a shortcut in MATLAB. How to create a shortcut is described on the following website from MathWorks:

http://de.mathworks.com/help/matlab/matlab_env/create-matlab-shortcuts-to-rerun-commands.html?refresh=true

7.3 Using the Command Window in MATLAB

After successful initialization and referencing of the system, the controller object can of course also be used via the Command Window in MATLAB.



```

Command Window

ans =

(c)2010 Physik Instrumente(PI) Karlsruhe, C-867.260,0,1.2.0.8

ans =

1=M-664.164

K>> E042.qPOS(axisname)

ans =

12.5000
  
```

Figure 7 Directly using the controller object via the Command Window

8 Troubleshooting

8.1 The MATLAB Script Generates Error Messages When Started on Another PC

Position in script where error occurs

```
%% Load PI MATLAB Driver GCS2
...
...
```

Explanation

Access by MATLAB to the MATLAB driver or the PI GCS2 DLL is not functioning correctly or cannot be found.

Solution

Install all the required software by following the installation instructions in Section 3.1 – “Standard Installation (Recommended)”. Then, if necessary, launch the PI Update Finder to update your software to the most recent version. If you execute an installer that was created before 2015, you must manually reinstall the MATLAB driver (see Section 9.1 – “Changing over to Version 1.2.0”).

8.2 Error when Loading the PI GCS2 DLL

Error message

New version (MATLAB driver version 1.2.0 or higher):

```
Loading PI MATLAB Driver GCS2 ...
Error using LoadGCSDLL (line 67)
Error while loading PI GCS2 DLL.

The PI GCS2 DLL was assumed to have the following path:
C:\ProgramData\PI\GCSTranslator\PI_GCS2_DLL.dll

The PI MATLAB DRIVER GCS2 either could not find or could not use the
"PI_GCS2_DLL.dll" or "PI_GCS2_DLL_x64.dll". Make sure you have installed
the newest driver for your PI controller and look at the PI MATLAB DRIVER
GCS2 Manual in chapter "Troubleshooting" headword "PI GCS2 DLL" for known
problems and fixes for problems while loading the PI GCS2 DLL.
```

Additional Information: MATLAB itself raised the following error:

...
...

Old version (MATLAB driver without installer):

Loading dll ...
There was an error ...
...
...

Explanation

There are a number of possible causes for this error. More detailed information on the cause of the error is displayed after the following text section: “Additional Information: MATLAB itself raised the following error:”.

8.2.1 PI GCS2 DLL Cannot be Found

Error message

...
...
Additional Information: MATLAB itself raised the following error:
There was an error loading the library
"C:\ProgramData\PI\GCSTranslator\PI_GCS2_DLL.dll"
The specified module could not be found.

Explanation

The file “PI_GCS2_DLL.dll” (or “PI_GCS2_DLL_x64.dll”) could not be found.

Solution

Install all the required software by following the installation instructions in Section 3.1 – “Standard Installation (Recommended)”. Then, if necessary, launch the PI Update Finder to update your software to the most recent version.

8.2.2 MATLAB Cannot Find a Suitable C Compiler

Error message

...
...
Additional Information: MATLAB itself raised the following error:
Index exceeds matrix dimensions.

Error in loadlibrary>getLoadlibraryCompilerConfiguration (line 497)

```

        compilerConfiguration=compilerConfiguration(1); %unix machines
return c and cpp compilers here

Error in loadlibrary (line 253)

[thunk_build_fn,preprocess_command]=getLoadlibraryCompilerConfiguration(cci
nclude,header,headername,compilerConfiguration);

Error in LoadGCSDLL (line 138)
    [notfound,warnings] = loadlibrary
(c.DLLName,c.hfile,'alias',c.libalias);

Error in PI_GCS_Controller (line 19)
    c = LoadGCSDLL(c);

Error in SampleC867 (line 23)
    Controller = PI_GCS_Controller();

```

Explanation

MATLAB performs a sort of recompilation of the PI GCS2 DLL. To do so, MATLAB requires a C compiler, which is not configured or integrated in MATLAB by default.

Solution

In the MATLAB Command Window, you can start the configuration setup using the following command:

```
>> mex -setup
```

The configuration of the C compiler is different in the 32-bit and 64-bit MATLAB variants.

MATLAB 32-bit

In the 32-bit variant, you always receive a list of C compilers because MATLAB includes a 32-bit C compiler. For example, you can use the “Lcc-win 32 v2.4.1” compiler. After successful configuration, you should receive the following response:

```

>> mex -setup
MEX configured to use 'lcc-win32' for C language compilation.
...
...

```

MATLAB 64-bit

There are two options in the 64-bit variant of MATLAB.

1. You receive a list with at least one C compiler. Configure MATLAB so that one of these compilers is used. You should then receive the following response:

```
>> mex -setup
MEX configured to use '<TheCompilerYouHaveChosen>' for C language
compilation.
...
...
```

2. You receive the following error message:

```
>> mex -setup
Error using mex
No supported compiler or SDK was found.
```

This error occurs because MATLAB does not include a 64-bit C compiler. You must additionally download and install a compiler of this type. Many compilers of this type are available for free download. MATLAB provides a comprehensive list of possible compilers on the following website:

<http://mathworks.com/support/compilers/R2014b/index.html>

We have successfully tested Microsoft Windows SDK, which you can download here:

<http://www.microsoft.com/en-us/download/details.aspx?id=8279>

Install the SDK and then configure the Microsoft Visual C compiler in MATLAB:

```
>> mex -setup
```

You should then receive the following response:

```
mex -setup
MEX configured to use 'Microsoft Visual C' for C language compilation.
...
...
```

Alternatively, you can install an IDE with an integrated C compiler and configure MATLAB to use this compiler.

8.2.3 MATLAB reports “PI_GCS2_DLL is not a valid Win32 application”

Error message

```
...
...
Additional Information: MATLAB itself raised the following error:
There was an error loading the library
"C:\ProgramData\PI\GCSTranslator\PI_GCS2_DLL.dll"
PI_GCS2_DLL is not a valid Win32 application.
```

Explanation

This error only occurs in older versions of the MATLAB driver (version numbers lower than V1.2.0 – versions without installer). Here the error that the MATLAB driver loads the 32-bit version of the PI GCS2 DLL in the 64-bit version of MATLAB can occur.

Solution

Upgrade to the new MATLAB driver of version number 1.2.0 or higher. (see Section 9.1 “– Changing over to Version 1.2.0”).

Update your script that uses your MATLAB driver, as described in Section 9.2 – “Notes on the Changeover”.

8.3 MATLAB Crashes Unexpectedly

8.3.1 Error

MATLAB crashes unexpectedly when loading or using the driver. The following error message is displayed:

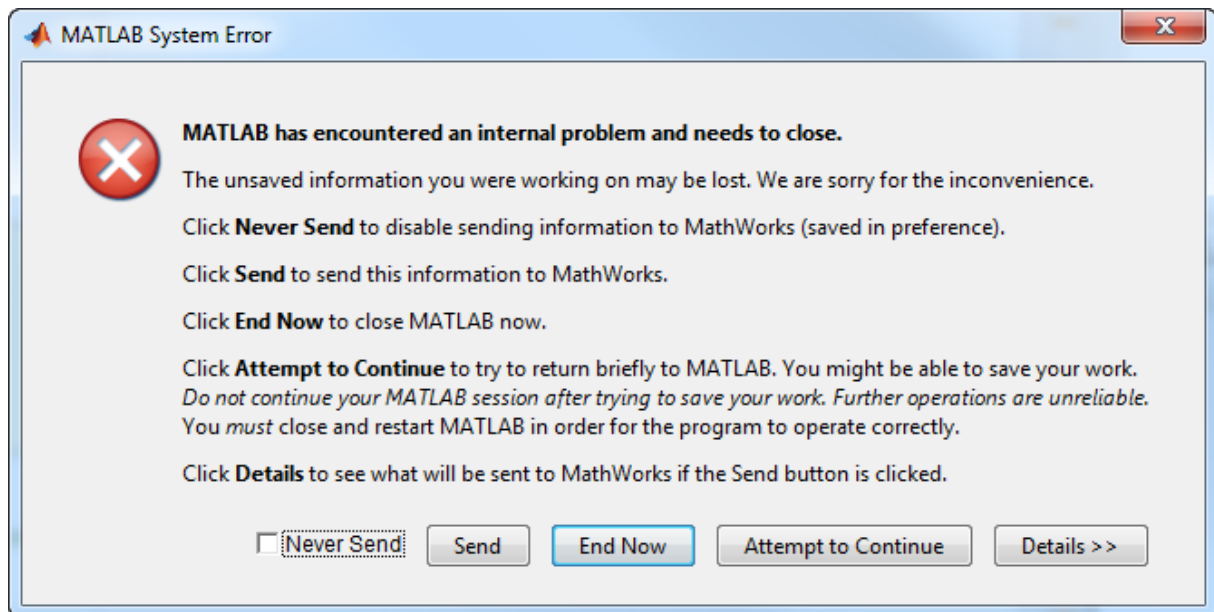


Figure 8 MATLAB error message

8.3.2 Cause

This error only occurs in older versions of the MATLAB driver (version numbers lower than V1.2.0). You are probably missing the include <windows.h> in your header file.

8.3.3 Solution

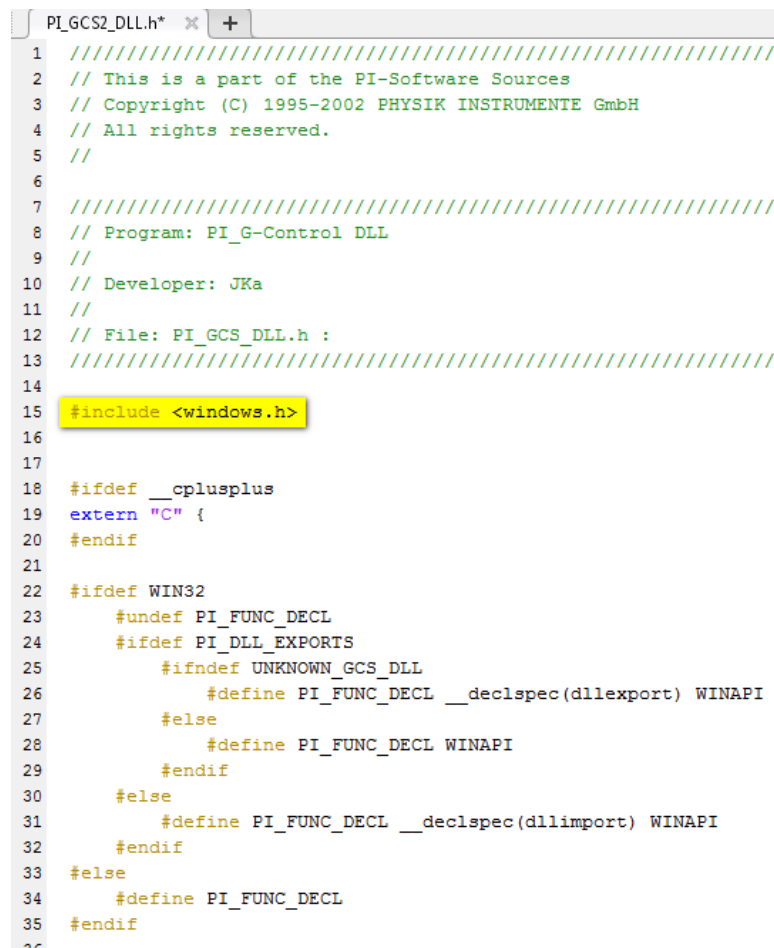
Variant 1 (recommended)

Upgrade to the new MATLAB driver of version number 1.2.0 or higher. (see Section 9.1 “– Changing over to Version 1.2.0”).

Update your script that uses your MATLAB driver, as described in Section 9.2 – “Notes on the Changeover”.

Variant 2

If you know the header file used, you can add the include directive yourself as shown in the figure below.



```

1  //////////////////////////////////////
2  // This is a part of the PI-Software Sources
3  // Copyright (C) 1995-2002 PHYSIK INSTRUMENTE GmbH
4  // All rights reserved.
5  //
6
7  //////////////////////////////////////
8  // Program: PI_G-Control DLL
9  //
10 // Developer: JKa
11 //
12 // File: PI_GCS_DLL.h :
13 //////////////////////////////////////
14
15 #include <windows.h>
16
17
18 #ifdef __cplusplus
19 extern "C" {
20 #endif
21
22 #ifdef WIN32
23     #undef PI_FUNC_DECL
24     #ifdef PI_DLL_EXPORTS
25         #ifndef UNKNOWN_GCS_DLL
26             #define PI_FUNC_DECL __declspec(dllexport) WINAPI
27         #else
28             #define PI_FUNC_DECL WINAPI
29         #endif
30     #else
31         #define PI_FUNC_DECL __declspec(dllimport) WINAPI
32     #endif
33 #else
34     #define PI_FUNC_DECL
35 #endif
36

```

Figure 9 Added directive in the header file: `#include <windows.h>` header.

8.4 “Error using callib”

8.4.1 Error

```
Error using calllib
Library was not found

Error in PI_GCS_Controller/qIDN (line 14)
    [bRet,szAnswer] =
        calllib(c.libalias,FunctionName,c.ID,szAnswer,8000);

Error in PI_GCS_Controller/subsref (line 47)
    varargout{1} = feval(fun,c);
```

8.4.2 Cause

The MATLAB driver is no longer loaded in MATLAB.

8.4.3 Solution

You have probably executed `Controller.Destroy` and then used a controller object, e.g. E042 or even the driver object “Controller”. Only execute `Controller.Destroy` if you no longer want to use the MATLAB driver.

You can reload the MATLAB driver by executing the following MATLAB command:

```
Controller = PI_GCS_Controller();
```

You must then re-establish the connection to the controller.

9 PI MATLAB Driver GCS2: Changing to Version 1.2.0

The advantages of the new version are:

- **Old samples and scripts are compatible with version 1.2.0** if you follow the steps in Section 9.1 – “Changing over to Version 1.2.0”.
- Extended range of functions, enhanced documentation.
- Automatically supports 32-bit and 64-bit MATLAB versions.
- Easier download of updates using the PI Update Finder.
- Uniform installation process for better and faster support.
- As a result, an installer makes it easy to change over to other Windows PCs.

9.1 Changing over to Version 1.2.0

The changeover can be performed very easily. Perform the following four steps:

- **Step 1**
Download the driver setup “PI_MATLAB_Driver_GCS2_Setup.exe” and the MATLAB samples “PI Matlab Driver Samples<Date>.zip”.
Contact support-software@pi.ws to get the login credentials for the download server.
- **Step 2**
Open “PI_MATLAB_Driver_GCS2_Setup.exe” to start the driver installation.
- **Step 3**
Start the sample “BeginWithThisSampleForTestingMatlabDriver.m”. The sample **must not** be in a folder with the “@PI_GCS_Controller” folder (see 9.3 – “Comparison of Old and New Structure”). The sample should run without errors⁸ and return the version name of the MATLAB driver installed.

⁸ Should errors occur, ensure that all components are installed correctly (see Section 3.1 – “Standard Installation (Recommended)”) and use the PIUpdateFinder to ensure that the software is up to date. Search Chapter 8 – “Troubleshooting” for solutions to known errors.

- **Step 4**

Insert your previously used MATLAB sample or your MATLAB script into the sample “BeginWithThisSampleForTestingMatlabDriver.m” (as described in the following section).

9.2 Notes on the Changeover

Inserting an old sample into “BeginWithThisSampleForTestingMatlabDriver.m”

Figure 10 shows an example of inserting an old sample:

- Copy the complete old program code except for the part at the beginning of the script in which the MATLAB driver is loaded.
- Paste the old program code into your new script “BeginWithThisSampleForTesting-MATLABDriver.m”.
- Ensure that the program codes `Controller.Destroy;` and `clear Controller;` do not appear twice at the end of the new script.
- Ensure that your new script is not in a folder with the “@PI_GCS_Controller” folder.

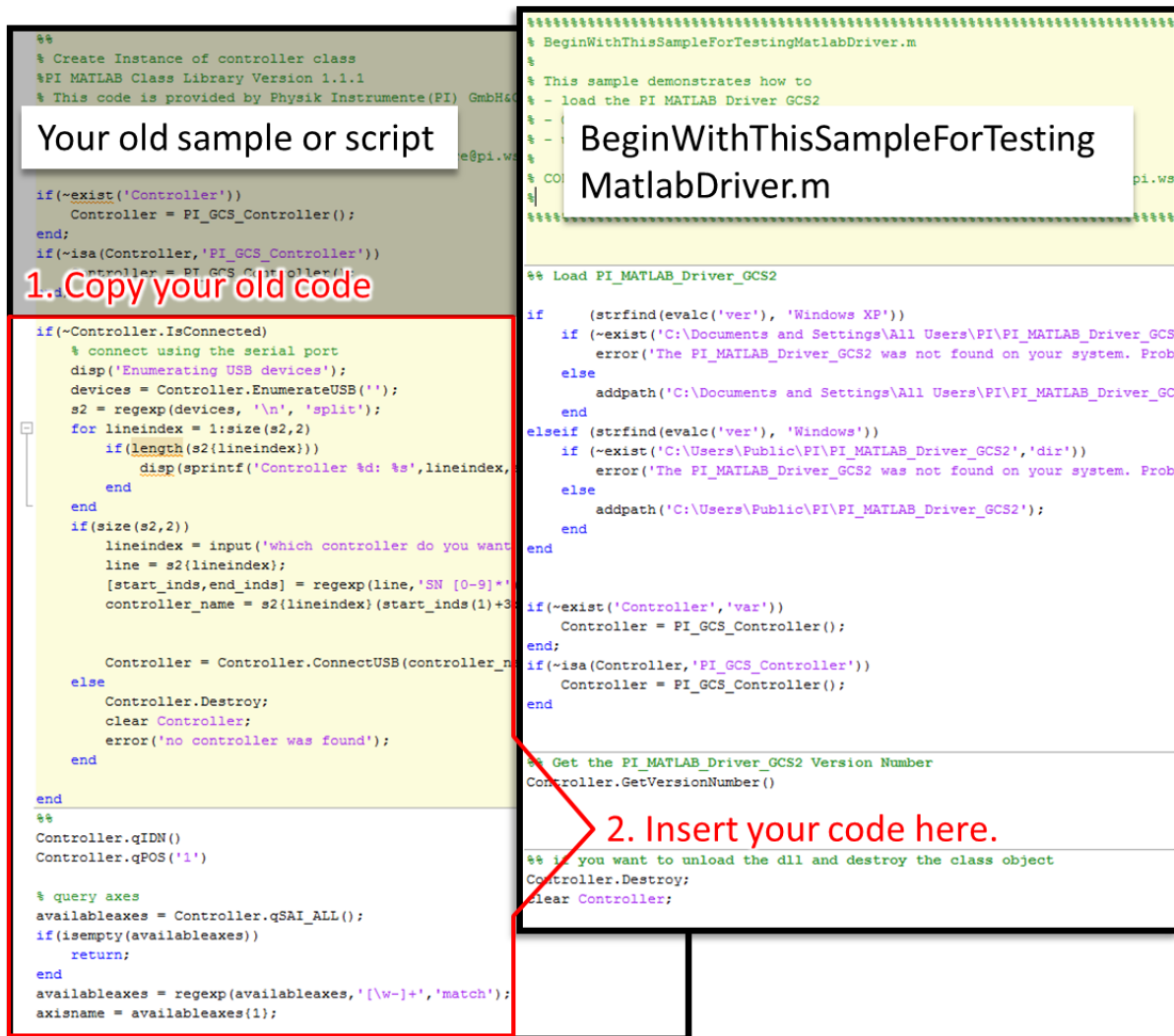


Figure 10 Example of inserting old source code into “BeginWithThisSampleForTestingMatlabDriver.m”

9.3 Comparison of Old and New Structure

Structure before version 1.2.0

The driver, the PI GCS2 DLL with corresponding h. file (header file) used, and the actual sample had to be in the same folder (see Figure 11). In most cases, the actual sample was called “FirstSample.m”. The driver generally consisted of the MATLAB files in “@PI_GCS_Controller”.

Name	Typ	Größe
@PI_GCS_Controller	Dateiordner	
PI_GCS2_DLL.dll	Anwendungserwe...	2.562 KB
PI_GCS2_DLL_x64.dll	Anwendungserwe...	3.405 KB
PI_GCS2_DLL.h	H-Datei	41 KB
FirstSample.m	M-Datei	10 KB

Figure 11 Example structure of the old MATLAB driver before version V1.2.0

Structure after Version 1.2.0

From version 1.2.0, the three components – samples or scripts, the MATLAB driver, and the PI GCS2 DLL – are saved separately from one another. The Samples folder only contains the examples. If the MATLAB driver is installed, each stand-alone sample can be started from any location on the system. This makes it significantly easier to handle the scripts.

Name	Typ	Größe
BeginWithThisSampleForTestingMatlabDriver.m	MATLAB Code	2 KB
BeginWithThisSampleWhenUsing_C867.m	MATLAB Code	4 KB
PI_DataRecorder.m	MATLAB Code	1 KB
Sample_DaisyChain.m	MATLAB Code	4 KB
Sample_TwoControllers_TwoStages.m	MATLAB Code	5 KB

Figure 12 Sample folder

The actual driver is installed by the installer in its own directory (see Section 3.3 – “Installation Location”). Therefore, it only exists once on the system and can be upgraded specifically with an update.

Name	Typ	Größe
@PI_GCS_Controller	Dateiordner	
PI_GCS2_DLL.h	H-Datei	41 KB
Version.txt	TXT-Datei	1 KB

Figure 13 File path of the MATLAB driver

The PI GCS2 DLL used is in the directory “C:\ProgramData\PI\GCSTranslator” and is also used by other applications, such as PIMikroMove. This means that only one update must be performed for the PI GCS2 DLL and that there are no different or outdated versions on your system.

Name	Typ	Größe
PI_GCS2_DLL.dll	Anwendungserwe...	7.609 KB
PI_GCS2_DLL_x64.dll	Anwendungserwe...	1.477 KB

Figure 14 File path “C:\ProgramData\PI\GCSTranslator” of the PI GCS2 DLL

Other Differences

Another difference between the old and new samples is that in most cases there was no separation between the driver object and the controller object:

Old version:

```
Controller = Controller.ConnectUSB(controllerSerialNumber);
```

New version:

```
E042 = Controller.ConnectUSB(controllerSerialNumber);
```

The old sample is still fully operational.

If necessary, the distinction between the driver object and the controller object in the old sample can be made later. However, this is not absolutely necessary. The newly introduced separation of driver objects from controller objects has the advantage that a clear distinction can be made between the driver functionality and the controller functionality (see also Section 4.1.3 – “Establishing a Connection”). This becomes especially important when operating multiple controllers from one MATLAB script.