

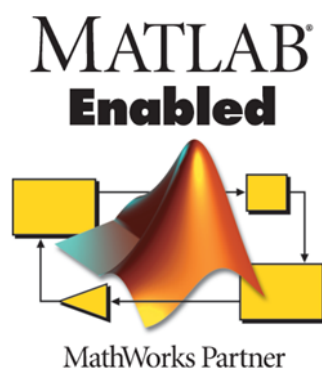
**SM155D**

**PI MATLAB Treiber GCS 2.0** (PI\_MATLAB\_Driver\_GCS2)

**Software-Handbuch**

Version: 1.1.0

Datum: 29.02.2016



Physik Instrumente (PI) GmbH & Co. KG ist Inhaberin der nachfolgend aufgeführten Marken:

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

Bei den nachfolgend aufgeführten Bezeichnungen handelt es sich um geschützte Firmennamen bzw. eingetragene Marken fremder Inhaber:

Microsoft, Windows, LabVIEW, MathWorks, MATLAB

Von PI zur Verfügung gestellte Softwareprodukte unterliegen den Allgemeinen Softwarelizenzbestimmungen der Physik Instrumente (PI) GmbH & Co. KG und können Drittanbieter-Softwarekomponenten beinhalten und/oder verwenden. Weitere Informationen finden Sie in den Allgemeinen Softwarelizenzbestimmungen und in den Drittanbieter-Softwarehinweisen, die nachfolgend verlinkt sind.

[General Software License Agreement](#)

[Third Party Software Note](#)

Im technologischen Umfeld der Piezoschreitantriebe (PiezoWalk®, NEXACT®, NEXLINE®) hält PI folgende Patente oder Patentanmeldungen:

DE10148267B4, EP1267478B1, EP2209202B1, EP2209203B1, US6800984B2, DE4408618B4

Im technologischen Umfeld der Multilayer-Piezoaktoren (PICMA®) hält PI folgende Patente oder Patentanmeldungen:

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

Im technologischen Umfeld der Ultraschall-Piezomotoren (PILine®) hält PI folgende Patente oder Patentanmeldungen:

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

Europe: EP0789937B1, EP1210759B1, 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

Im technologischen Umfeld der magnetischen Direktantriebe (PIMag®) hält PI folgende Patente oder Patentanmeldungen:

WO212146709A2, DE102012207082A1

Im technologischen Umfeld der Piezoträgheitsantriebe (PIShift, PiezoMike) hält PI folgende Patente oder Patentanmeldungen:

EP2590315A1, WO213064513A1, DE102011109590A1, WO2013020873A1

© 2015-2016 Physik Instrumente (PI) GmbH & Co. KG, Karlsruhe, Deutschland. Die Texte, Bilder und Zeichnungen dieses Handbuchs sind urheberrechtlich geschützt. W Physik Instrumente (PI) GmbH & Co. KG behält insoweit sämtliche Rechte vor. Die Verwendung dieser Texte, Bilder und Zeichnungen ist nur auszugsweise und nur unter Angabe der Quelle erlaubt.

Originalbetriebsanleitung

Erstdruck: 29.02.2016

Dokumentnummer: SM155D, SMoe, BRo

Version: 1.1.0

PI\_MATLAB\_Driver\_GCS2\_Manual\_SM155D.docx

Änderungen vorbehalten. Dieses Handbuch verliert seine Gültigkeit mit Erscheinen einer neuen Revision. Die jeweils aktuelle Revision ist auf unserer Website [www.pi.de](http://www.pi.de) zum Herunterladen verfügbar.

## Inhalt

1	Voraussetzungen	6
1.1	Für welche Version des MATLAB-Treibers gilt diese Dokumentation?	6
1.2	Welche Betriebssysteme unterstützt der PI MATLAB Driver GCS2?	6
1.3	Welche PI-Controller unterstützt der PI MATLAB Driver GCS2?	6
2	Einführung PI MATLAB Driver GCS2	7
3	Installation	8
3.1	Standardinstallation (empfohlen)	8
3.2	Benutzerdefinierte Installation	8
3.3	Installationsort	8
4	Grundlegende Verwendung des MATLAB-Treibers	9
4.1	Startsample: „BeginWithThisSampleWhenUsing_<ControllerName>.m“	9
4.2	Konfiguration und Referenzierung	13
4.3	Grundlegende Controller-Funktionen	13
4.4	Beenden der Verbindung	14
4.5	Unload des Treibers	14
4.6	Nutzung des Verstellers	14
5	Nutzung des Controllers mit GCS-Befehlen	15
5.1	Kommunikation mit dem Controller durch GCS2-Befehle	15
5.2	Funktionsweise des MATLAB-Treibers	16
6	Weiterführende Samples	21
6.1	Weiterführende Samples – Funktionen und Skripte	21
6.2	Ansteuerung mehrerer Controller	22
6.3	Daisy-Chain	22
7	Tipps & Tricks	24
7.1	Mitloggen von GCS2-Befehlen in PIMikroMove bei der Initialisierung	24
7.2	Shortcuts erstellen in MATLAB	25
7.3	Nutzung des „Command Window“ in MATLAB	26
8	Störungsbehebung	27

8.1	Das MATLAB-Skript erzeugt Fehlermeldungen, wenn man es auf einem anderen PC startet	27
8.2	Fehler beim Laden der PI GCS2 DLL	27
8.3	MATLAB stürzt unerwartet ab	32
8.4	„Error using callib“	34
9	PI MATLAB Driver GCS2: Umstieg auf Version 1.2.0	35
9.1	Umstieg auf Version 1.2.0	35
9.2	Hinweise zum Umstieg	36
9.3	Gegenüberstellung von alter und neuer Struktur	38

# 1 Voraussetzungen

## 1.1 Für welche Version des MATLAB-Treibers gilt diese Dokumentation?

Diese Dokumentation gilt für die Version 1.2.0 des PI MATLAB Driver GCS2. Die Versionsnummer finden Sie im Quellcode des PI MATLAB Driver GCS2 oder über den PI Update Finder. Kann der PI Update Finder den PI MATLAB Driver GCS2 nicht finden, ist der Treiber nicht installiert oder es befindet sich eine ältere Version auf Ihrem System (siehe hierzu Kapitel 9 – „PI MATLAB Driver GCS2: Umstieg auf Version 1.2.0“).

Nach erfolgreichem Laden des PI MATLAB Driver GCS2 in MATLAB kann die Versionsnummer auch mit dem folgenden Befehl ermittelt werden:

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

Bei älteren Versionen des PI MATLAB Driver GCS2 ist diese Funktion nicht implementiert. Statt der Versionsnummer wird MATLAB deshalb einen Fehler zurückliefern.

## 1.2 Welche Betriebssysteme unterstützt der PI MATLAB Driver GCS2?

### Windows

Alle Versionen ab einschließlich WinXP. Der PI MATLAB Driver GCS2 wird über einen Installer auf Ihrem System installiert und konfiguriert.

## 1.3 Welche PI-Controller unterstützt der PI MATLAB Driver GCS2?

Der PI MATLAB Driver GCS2 unterstützt alle PI-Controller, die über das GCS2-Protokoll angesprochen werden können. Dieses Protokoll ist bei fast allen neueren PI-Controllern implementiert.

## 2 Einführung PI MATLAB Driver GCS2

Der „PI MATLAB Driver GCS2“, im Folgenden immer „MATLAB-Treiber“ genannt, ist das MATLAB-Interface für die Ansteuerung der meisten PI-Controller (vgl. Abschnitt 1.3 – „Welche PI-Controller unterstützt der PI MATLAB Driver GCS2?“).

Der MATLAB-Treiber basiert auf dem GCS2-Protokoll von PI<sup>1</sup>, mit dem fast alle PI-Controller angesteuert werden können. Dieses Protokoll besteht aus mnemonischen Kürzeln, die i. d. R. eine Länge von 3 Buchstaben haben, z. B. MOV (für „move“), FRF (für „find reference“) oder POS? (für „query position“). Diese Kürzel werden mit Parametern kombiniert und an den Controller gesendet. Mit dem folgenden Befehl soll der Versteller an Achse 1 des Controllers auf die Position 31 µm bewegt werden:

```
MOV 1 31
```

Dieser Befehl wird mittels ASCII-Zeichen an den Controller gesendet. Mit Hilfe des MATLAB-Treibers wird derselbe Befehl für den (fiktiven) Controller E-042 z. B. mit folgendem MATLAB-Kommando abgesetzt:

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

Damit dieser Befehl in MATLAB funktioniert, müssen Sie den MATLAB-Treiber installieren, den MATLAB-Treiber in MATLAB laden und eine Verbindung zum Controller herstellen. Wie dies funktioniert, wird in den Kapiteln 3 – „Installation“ und Kapitel 4 – „Grundlegende Verwendung des MATLAB-Treibers“ beschrieben. In Kapitel 5 – „Nutzung des Controllers mit GCS-Befehlen“ wird die Verwendung der GCS-Befehle in MATLAB beschrieben, und Kapitel 8 – „Störungsbehebung“ behandelt häufige Problemfälle, z. B. den Umstieg von einem 32-Bit-System auf ein 64-Bit-System.

---

<sup>1</sup> Das Protokoll basiert auf ASCII-Zeichen. Eine ausführliche Beschreibung dieses Protokolls finden Sie im Benutzerhandbuch Ihres Controllers im Abschnitt „GCS Commands“.

## 3 Installation

### 3.1 Standardinstallation (empfohlen)

Es wird empfohlen, die Komplettinstallation der PI-Software durchzuführen. Dadurch wird der MATLAB-Treiber und alle benötigte Hilfssoftware installiert. Der Installer hierfür befindet sich auf Ihrer Produkt-CD und trägt den Namen:

PI\_<ControllerName >.CD\_Setup.exe

Führen Sie anschließend mit Hilfe des PI Update Finders ein Software-Update durch (besonders wichtig bei älteren Produkt-CDs).

Bei Produkt-CDs mit einem Release-Datum vor 2015 wird der MATLAB-Treiber nicht mitinstalliert. Wie Sie ihn herunterladen und installieren, ist in Kapitel 9.1 – „Umstieg auf Version 1.2.0“ beschrieben.

Nach erfolgreicher Installation wird empfohlen, den Controller und den Verstärker zunächst mit PIMikroMove in Betrieb zu nehmen, um sicherzustellen, dass das Gesamtsystem richtig angeschlossen und einsatzbereit ist.

### 3.2 Benutzerdefinierte Installation

Möchten Sie auf die Gesamtinstallation verzichten, müssen mindestens die drei nachfolgend aufgeführten Installer auf der Produkt-CD ausgeführt werden. Dadurch verzichten Sie allerdings auf wichtige Diagnose-, Konfigurations- und Update-Tools.

PI\_MATLAB\_Driver\_GCS2\_Setup.exe

PI\_GCS\_Library\_PI\_GCS2\_DLL\_Setup.exe

PI\_Stage\_Database\_PIStages2\_Setup.exe (nur falls auf der Produkt-CD vorhanden)

Sollte Ihnen die Produkt-CD nicht mehr zur Verfügung stehen, wenden Sie sich an Ihren lokalen Vertriebspartner oder [support-software@pi.ws](mailto:support-software@pi.ws).

### 3.3 Installationsort

Durch die Installation wird der MATLAB-Treiber in folgendes Verzeichnis gespeichert:

Win7/8, Vista: C:\Users\Public\PI\PI\_MATLAB\_Driver\_GCS2

WinXP: C:\Documents and Settings\All Users\PI\PI\_MATLAB\_Driver\_GCS2



## 4 Grundlegende Verwendung des MATLAB-Treibers

### 4.1 Startsample: „BeginWithThisSampleWhenUsing\_<ControllerName>.m“

Die Verwendung des MATLAB-Treiber soll anhand eines Samples dargestellt werden. Dieses Sample finden Sie auf Ihrer Produkt-CD. Bei der Standardinstallation werden die Samples an folgende Stelle auf Ihrem PC gespeichert:

Win7/8 und Vista: C:\Users\Public\PI\<ControllerName>\Samples

WinXP: C:\Documents and Settings\All Users\PI\<ControllerName>\Samples

#### 4.1.1 Laden des MATLAB-Treiber

```
%% 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 diesem Abschnitt wird der MATLAB-Treiber geladen. Zunächst wird der Pfad des Treibers zum Suchpfad von MATLAB hinzugefügt. Dann wird mit `Controller = PI_GCS_Controller();` der MATLAB-Treiber geladen. Anschließend kann über das Treiber-Objekt „Controller“ im Skript auf den MATLAB-Treiber zugegriffen werden.

#### 4.1.2 Konfiguration des Verstellers und der Verbindung

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

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

Mit Hilfe der Enumerate-Funktionen des MATLAB-Treibers können Sie sich alle angeschalteten PI-Geräte anzeigen lassen, die über USB oder Netzwerk mit Ihrem PC verbunden sind (RS-232 und andere Verbindungen besitzen diese Funktionalität nicht).

Im nun folgenden Abschnitt müssen Sie selbst zwei Einstellungen vornehmen, um das Sample lauffähig zu machen.

## Einstellung 1 – Verstellertyp (gilt nur für C-Controller<sup>2</sup>)

Geben Sie den Typ (nicht die Seriennummer) des angeschlossenen Verstellers an, z. B. „M-664.164“. Den Namen des Verstellers finden Sie auf dem Typenschild des Verstellers. ACHTUNG! Stellen Sie sicher, dass Sie den richtigen Verstellernamen eingeben. Ein falscher Verstellernamen kann zu einer Fehlansteuerung des Verstellers durch den Controller führen und den Versteller beschädigen oder zerstören.

## Einstellung 2 – Verbindungsparameter

Als zweites wird eine bestimmte Verbindung aktiviert und anschließend konfiguriert. Damit eine Verbindung aufgebaut werden kann, müssen folgende Parameter korrekt gesetzt werden:

### RS-232-Verbindung: COM-Port und Baudrate

Welcher COM-Port genutzt wird, können Sie im Geräte-Manager nachschauen oder über PIMikroMove in „Start up Controller“ herausfinden. Die Baudrate kann dem Manual des Controllers entnommen werden. Bei einigen Controllern kann die Baudrate über DIP-Schalter am Controller manuell eingestellt werden.

### USB-Verbindung: Seriennummer des Controllers

Die Seriennummer können Sie auf dem Typenschild des Controllers nachlesen oder mit Hilfe der Funktion `Controller.EnumerateUSBAsArray()` ermitteln.

### TCP/IP-Verbindung: IP-Nummer und Port-Nummer

IP und Port-Nummer können Sie mit PIMikroMove oder mit Hilfe der Funktion `Controller.EnumerateTCPIPDevicesAsArray()` ermitteln. Für gewöhnlich ist die Port-Nummer bei PI-Controllern „50000“. Sollten sich bei der TCP/IP-Verbindung Probleme ergeben, wenden Sie sich an Ihren Netzwerkadministrator.

```
%% Parameters
% You MUST EDIT AND ACITVATE 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.
```

<sup>2</sup>Auch für einige E-Controller, z. B. den E-712, ist in Kombination mit bestimmten Verstellertypen ein CST möglich. Beachten Sie hierzu das Controller-Handbuch.

```
% 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
```

Haben Sie die Einstellungen in diesem Abschnitt korrekt angegeben und das passende Sample für Ihren Controller ausgewählt, sollte Ihr Sample von nun ab fehlerfrei ausführbar sein.

Im oben dargestellten Beispiel würde Ihr PC im Laufe des MATLAB-Skripts versuchen, über USB zum Controller mit der Seriennummer 123456789 eine Verbindung herzustellen. Nach erfolgreichem Verbindungsaufbau würde dem Controller übermittelt, dass ein Verstärker vom Typ „M-664.164“ an ihn angeschlossen ist.

### 4.1.3 Aufbau einer Verbindung

```
%% 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();

```

Im diesem Abschnitt wird eine Verbindung mit einem spezifischen Controller hergestellt. Dies soll anhand des fiktiven Controllers E-042 beschrieben werden.

Um den Controller E-042 ansprechen zu können, muss zunächst mit Hilfe des MATLAB-Treibers die Verbindung aufgebaut werden. Die Verbindung wird in dem neuen Controller-Objekt „E042“ gespeichert:

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

Alle weiteren Zugriffe auf den E-042 erfolgen anschließend über das Objekt E042.

`ConnectXXX` ist dabei nur ein Platzhalter für den eigentlichen Verbindungstyp (RS-232, USB, TCP/IP, ...). Die Verbindungs-Funktionen sind im Folgenden dargestellt.

Sollte eine Verbindung fehlschlagen, versuchen Sie sich zunächst mit PIMikroMove zu verbinden. Funktioniert die Verbindung in PIMikroMove, haben Sie in MATLAB wahrscheinlich nur eine falsche Konfiguration der Kommunikationsschnittstelle gewählt.

Stellen Sie außerdem sicher, dass kein anderes Programm (PITerminal, PIMikroMove, ...) zeitgleich mit dem Controller verbunden ist, wenn Sie aus MATLAB heraus eine Verbindung zum Controller aufbauen.

## RS-232

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

## USB

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

## TCP/IP

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

## Weitere Verbindungsvarianten

Weitere Verbindungsmöglichkeiten sind

- Gleichzeitiges Verbinden mit mehreren Controllern  
(siehe Abschnitt 6.2 – „Ansteuerung mehrerer Controller“)
- Daisy-Chain  
(siehe Abschnitt 6.3 – „Daisy-Chain“)
- NI GPIB  
Funktion: `ConnectNIgplib`

## Überprüfung ob Verbindung erfolgreich

War der Verbindungsaufbau erfolgreich, wird durch den folgenden Befehl der Name des angeschlossenen Controllers im MATLAB „Command Window“ angezeigt:

```
E042.qIDN()
```

## 4.2 Konfiguration und Referenzierung

Anschließend folgt i. d. R. die Konfiguration des Controllers und danach die Referenzierung des Verstellers. Dieser Ablauf kann je nach Controller und Versteller stark variieren. Die passenden Befehle sind in dem spezifischen Sample für Ihren Controller angegeben:

„BeginWithThisSampleWhenUsing\_<ControllerName>.m“

Haben Sie kein spezifisches Sample für Ihren Controller, können Sie die notwendigen Initialisierungsbefehle im Handbuch Ihres Controllers nachlesen. Ein Verständnis für die Initialisierung und Referenzierung kann prinzipiell von Vorteil sein, weil Sie ggf. einzelne Schritte überspringen können und Ihr MATLAB-Skript dadurch schneller durchläuft.

Eine andere einfache Möglichkeit, die passenden Befehle für Konfiguration und Referenzierung zu finden, besteht im Mitloggen der GCS-Befehle in PIMikroMove über das Log Window. Dies ist im Abschnitt 7.1 – „Mitloggen von GCS2-Befehlen in PIMikroMove bei der Initialisierung“ beschrieben.

## 4.3 Grundlegende Controller-Funktionen

```
% 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)
```

Mit `E042.MOV('1', position);` wird eine Position kommandiert. Soll das MATLAB-Skript erst weiterlaufen, wenn die Ziel-Position erreicht ist, muss auf `E042.IsMoving('1')` überprüft werden.

## 4.4 Beenden der Verbindung

Durch den folgenden Befehl wird die Verbindung mit dem Controller E-042 beendet:

```
E042.CloseConnection;
```

Durch das Schließen der Verbindung kann der Controller wieder aus anderen Programmen (z. B. PIMikroMove) heraus angesprochen werden. Bei Bedarf kann die Verbindung mit `E042 = Controller.ConnectUSB(controllerSerialNumber);` neu aufgebaut werden.

## 4.5 Unload des Treibers

Mit dem folgenden Befehl wird ein Unload des Treibers durchgeführt:

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

Dadurch werden automatisch alle Verbindungen zu den Controllern geschlossen. Alle Controller-Objekte verlieren die Möglichkeit, mit den Controller zu kommunizieren. Führen Sie das Unload des Treibers deshalb möglichst erst ganz am Ende Ihres Skripts durch.

Da das Controller-Objekt mit dem Unload des MATLAB-Treiber seine Funktion verliert, ist es sinnvoll, dieses ebenfalls zu löschen: `clear E042;`

Das Schließen von MATLAB führt ebenfalls zu einem Unload des Treibers.

## 4.6 Nutzung des Verstellers

Sind alle vorherigen Schritte erfolgreich abgearbeitet, können Sie über den MATLAB-Treiber fast alle zur Verfügung stehenden GCS-Befehle nutzen<sup>3</sup>. Wie Sie den passenden GCS-Befehl finden und wie Sie ihn aufrufen müssen, ist im nachfolgenden Kapitel beschrieben.

---

<sup>3</sup> Sollte ein GCS-Befehl nicht in PI\_MATLAB\_Driver\_GCS2 implementiert sein, haben Sie möglicherweise nicht die aktuellste Version von MATLAB-GCS. Starten Sie den PI Update Finder, um die aktuellste Version des MATLAB-Treibers zu erhalten. Beachten Sie auch, dass nicht alle Controller alle GCS-Befehle implementieren.

## 5 Nutzung des Controllers mit GCS-Befehlen

### 5.1 Kommunikation mit dem Controller durch GCS2-Befehle

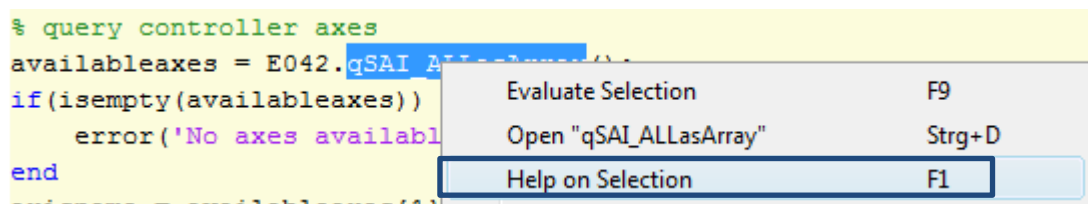
Wie bereits in Kapitel 2 – „Einführung PI MATLAB Driver GCS2“ beschrieben, basiert der MATLAB-Treiber auf dem GCS2-Protokoll von PI, mit dem fast alle PI-Controller angesprochen werden können.

Der MATLAB-Treiber implementiert einen Großteil der GCS2-Befehle. Die meisten PI-Controller beherrschen jedoch nur eine Teilmenge der GCS2-Befehle. Welche Befehle Ihr Controller kennt und wie sie funktionieren, ist im Controllerhandbuch dokumentiert. Eine Liste der GCS2-Befehle, die dem Controller bekannt sind, können Sie mit dem folgenden Befehl auch direkt über das Controller-Objekt abfragen<sup>4</sup>:

```
E042.qHLP( )
```

#### Integrierte Hilfefunktion

Momentan gibt es nur für vereinzelte Funktionen eine in MATLAB integrierte Hilfefunktion (z. B. für die Funktion `EnumerateUSBAsArray`). Um die Hilfe für eine MATLAB-Treiber-Funktion zu öffnen, markieren Sie den Funktionsnamen und drücken F1 oder gehen über das Kontext-Menü (vgl. Bild 1). Damit das funktioniert, muss mit `addpath(...)` der Pfad des MATLAB-Treibers gesetzt sein, was im Abschnitt `%% Load PI MATLAB Driver GCS2` erfolgt. In zukünftigen Versionen soll diese Hilfe schrittweise für alle Funktionen ergänzt werden.

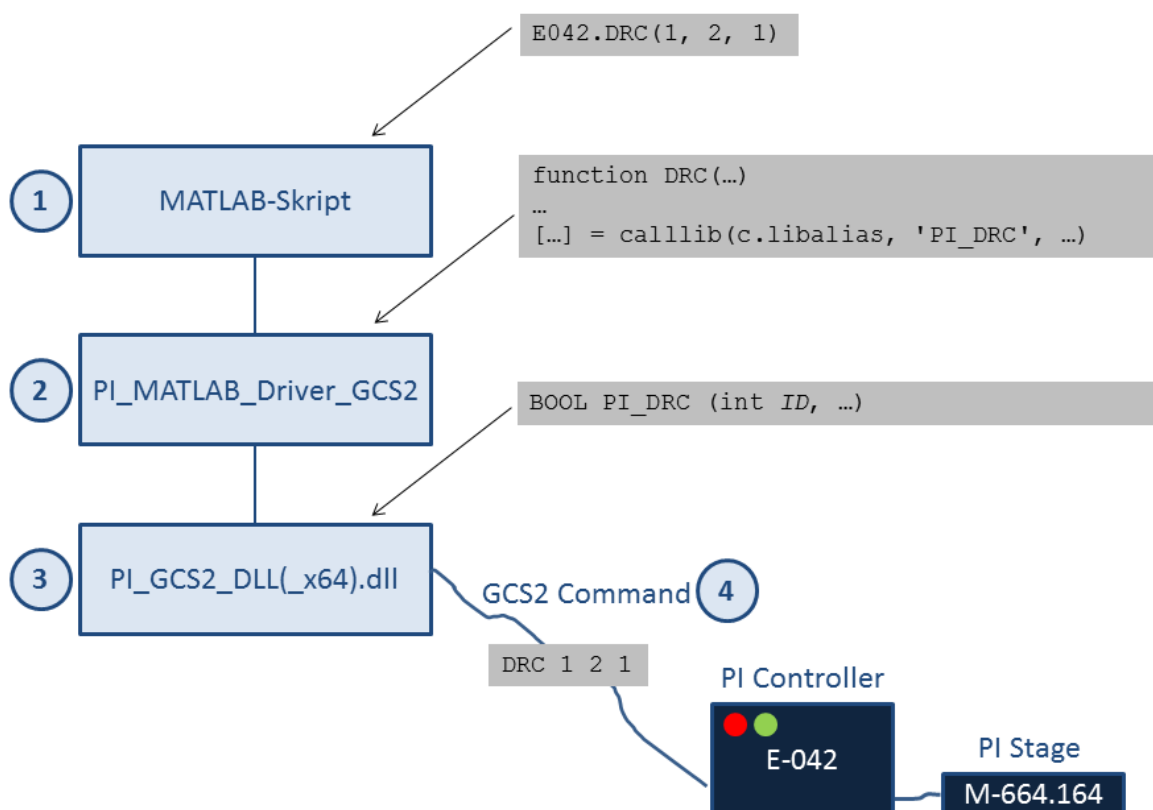


**Bild 1** Öffnen der Hilfe einer Controller-Objekt-Funktion, z. B. „qSAI\_ALLasArray“ (momentan nur für wenige Funktionen verfügbar)

<sup>4</sup> Diese Abfrage liefert die GCS2-Befehle, nicht die Funktionsnamen des Controller-Objekts. Beachten Sie hierzu Tabelle 1 – „Unterschiede und Gemeinsamkeiten von C-Schnittstelle der PI GCS2 DLL und MATLAB-Treiber“.

Momentan sind die meisten Funktionen noch undokumentiert. Um alle GCS-Befehle richtig anzuwenden und die Ein- und Ausgabeparameter korrekt an die Funktion zu übergeben, ist deshalb momentan ein Workaround notwendig. Dieser wird in den folgenden Abschnitten beschrieben.

## 5.2 Funktionsweise des MATLAB-Treibers



**Bild 2** Im MATLAB-Skript wird ein Befehl abgesetzt (1).  
Der MATLAB-Treiber ruft durch die Funktion calllib die PI GCS2 DLL auf (2 und 3).  
Die PI GCS2 DLL sendet einen GCS-Befehl vom PC an den PI-Controller (4).

Der MATLAB-Treiber benutzt die Datei „PI\_GCS2\_DLL.dll“<sup>5</sup>. Dies ist eine dynamische Programmibibliothek (im Weiteren abgekürzt als „DLL“), die u. a. die Kommunikation über das GCS2-Protokoll implementiert. Diese DLL besitzt eine Schnittstelle für die Programmiersprache C und diese C-Schnittstelle der „PI\_GCS2\_DLL.dll“ wird vom MATLAB-Treiber genutzt.

<sup>5</sup> Dies gilt für 32-Bit-Systeme. Bei 64-Bit-Systemen wird die PI\_GCS2\_DLL\_x64.dll verwendet.



Im Dokument „PIGCS\_2\_0\_DLL\_ ... .pdf“ sind alle Kommunikations-Befehle dieser C-Schnittstelle dokumentiert. Mit Hilfe dieses Dokuments können Sie die Funktionsweise aller MATLAB-Treiber-Funktionen nachvollziehen.

Wenn Sie z. B. verstehen möchten, wie der Befehl `E042.DRC(1, 2, 1)` genau funktioniert und welche Parameter er erwartet und zurückgibt, suchen Sie im PDF nach „DRC“. Dort werden Sie die folgende Schnittstellenbeschreibung finden:

```
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)

**Bild 3** Beschreibung der C-Schnittstelle der DLL in „PIGCS\_2\_0\_DLL\_ ... .pdf“

Lassen Sie sich nicht davon abschrecken, dass es sich um eine Beschreibung für die Programmiersprache C handelt. Es ist relativ einfach, von der Schnittstellenbeschreibung der PI GCS2 DLL auf die Schnittstellenbeschreibung in MATLAB zu schließen. Wie dies im Detail funktioniert, ist in der nachfolgenden Tabelle beschrieben.

**Tabelle 1** Unterschiede und Gemeinsamkeiten von C-Schnittstelle der PI GCS2 DLL und MATLAB-Treiber

Beschreibung	C-Schnittstelle in der PI GCS2 DLL	MATLAB
<b>Funktionsname</b> Das Präfix „PI_“ wird in MATLAB weggelassen.	<code>PI_XXX(...)</code> Beispiel: <code>PI_DRC(...)</code>	<code>XXX</code> Beispiel: <code>DRC(...)</code>
<b>Controller-ID</b> In MATLAB werden verschiedene Controller nicht über eine ID, sondern über den Namen des Controller-Objekts unterschieden.	Variable ID <code>PI_XXX(ID, ...)</code> Beispiel: <code>PI_DRC(1, ...)</code> <code>PI_DRC(2, ...)</code>	Variable ID <code>ControllerName.XXX(...)</code> Beispiel: <code>E042.DRC(...)</code> <code>C007.DRC(...)</code>
<b>Datentypen für Zahlen</b> In MATLAB gibt es nur einen Datentyp für Zahlen.	<code>int, double, const double, const int*, ...</code>	<code>double</code>
<b>Datentyp für Strings</b> In MATLAB gibt es nur einen Datentyp für Strings.	<code>char*, const char*, ...</code>	<code>char</code>
<b>Rückgabewerte</b> In MATLAB werden neue Werte direkt zurückgegeben. Die DLL ändert Werte der übergebenen Variablen, wenn sie einen Stern, aber nicht das Schlüsselwort „const“ vorangestellt haben.  Keine Änderung durch DLL: <code>const char* param1</code>  Änderung durch DLL: <code>int* param2</code>	<code>PI_XXX(int ID, const char* param1, int* param2, double* param3)</code> Beispiel: <code>BOOL PI_qPOS(int ID, const char* szAxes, double* pdValueArray)</code>	<code>[param2, param3] = E042.XXX(param1)</code> Beispiel: <code>pdValueArray = E042.qPOS(szAxes)</code>

Eine direkte Gegenüberstellung von C-Schnittstelle und MATLAB zeigt das folgende Beispiel:

MATLAB:

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

C-Schnittstelle der PI GCS2 DLL:

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

## 5.2.1 Interne Funktionsweise des MATLAB-Treibers

Wenn Sie eine Funktion mit sehr vielen Ein- und Ausgabeparametern benutzen, reicht das in Tabelle 1 dargestellte Mapping zwischen C-Schnittstelle der DLL und dem MATLAB-Treiber ggf. nicht aus. Für diesen Fall wird in diesem Abschnitt auf die interne Funktionsweise des MATLAB-Treibers eingegangen.

Jede Funktion des MATLAB-Treibers ist ähnlich aufgebaut. Dabei gibt es drei Abschnitte, die auch für den Anwender interessant sind.

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

In Zeile 1 werden die Übergabeparameter an die Funktion definiert, im Beispiel der Parameter `szAxes`<sup>6</sup>. Der Rückgabeparameter der Funktion ist `dValues`.

Ggf. sind Eingabeparameter optional. Im Beispiel ist `szAxes` optional. Dies wird im zweiten Abschnitt des Beispiels dargestellt. Die Logik hier besagt: wenn „Number of ARGuments IN“ (`nargin`) gleich eins, dann wird ein Default-Parameter `szAxes = ''` zugewiesen.

Im dritten Abschnitt wird die GCS2-DLL aufgerufen. Hierbei sind `c.libalias` und `FunctionName` zur Verwaltung der PI GCS2 DLL notwendig, die der MATLAB-Treiber benutzt. Diese zwei Parameter sind immer vorhanden und können vom Anwender ignoriert werden. Durch `c.ID` wird der spezifische Controller verwaltet. Die Parameter `szAxes` und `pdValues` sind Übergabeparameter an die PI GCS2 DLL. Die Parameter `c.ID`, `szAxes` und

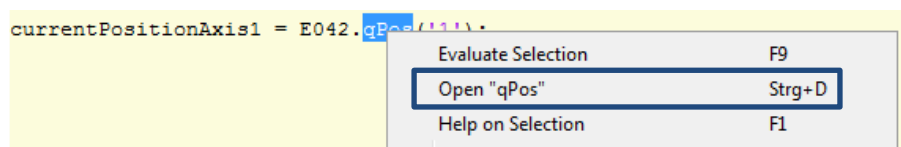
<sup>6</sup> Der Parameter `c` ist zur internen Verwaltung der PI\_MATLAB\_Driver\_GCS2 notwendig und muss vom Anwender nicht übergeben werden.

`pdValues` entsprechen exakt dem Funktionsaufruf der C-Schnittstelle der „PI\_GCS2\_DLL.dll“, die im Dokument „PIGCS\_2\_0\_DLL\_ ... .pdf“ ausführlich dokumentiert ist.

Der Rückgabeparameter der PI GCS2 DLL ist `ret`. Die Parameter `szAxes` und `dValues` sind Parameter, die ggf. durch die DLL verändert wurden. Der zweite Parameter `dValues` wird anschließend als Rückgabeparameter der Funktion des MATLAB-Treibers an den Anwender nach außen weitergereicht.

## 5.2.2 Aufruf des Quellcodes einer MATLAB-Treiber-Funktion

Um den Programmcode einer MATLAB-Treiber-Funktion zu öffnen, markieren Sie den Funktionsnamen und drücken Sie Strg+D oder gehen über das Kontext-Menü (vgl. Bild 4). Damit das funktioniert, muss im mit `addpath(...)` der Pfad des MATLAB-Treiber gesetzt sein, was im Abschnitt `%% Load PI MATLAB Driver GCS2` erfolgt.



**Bild 4** Öffnen des Programmcodes einer Objekt-Funktion.

## 6 Weiterführende Samples

### 6.1 Weiterführende Samples – Funktionen und Skripte

In Ihrem Samples-Ordner finden Sie weiterführende Samples. Diese Samples lassen sich in zwei Kategorien einteilen:

- Sample-Funktionen:  
sind durch das Präfix „**PI\_**“ gekennzeichnet, z. B. „PI\_DataRecorder.m“
- Stand-alone Skripte:  
sind durch das Präfix „**Sample\_**“ gekennzeichnet, z. B. „Sample\_DaisyChain.m“

#### Sample-Funktionen

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

Sample-Funktionen erwarten ein Controller-Objekt, das bereits mit dem Controller verbunden ist. Der Controller muss korrekt konfiguriert und referenziert sein. Nicht alle Sample-Funktionen passen zu jedem Versteller. Überprüfen Sie ggf. mit `E042.qHLP()` ob Ihr Controller alle GCS2-Befehle unterstützt, die in der Sample-Funktion verwendet werden.

#### Stand-Alone Sample-Skripte

Stand-Alone Skripte beinhalten den Verbindungsaufbau. ACHTUNG! Der Verbindungsaufbau ist i. d. R. nur beispielhaft dargestellt und muss von Ihnen noch an Ihr System aus einem oder mehreren Verstellern und Controllern angepasst werden (orientieren Sie sich für die Konfiguration an Ihrem Start-Sample „BeginWithThisSampleWhenUsing\_<Controller-Name>.m“ und an der Beschreibung im Handbuch Ihres Controllers).

## 6.2 Ansteuerung mehrerer Controller

Die Ansteuerung mehrerer PI-Controller von einem MATLAB-Skript aus ist sehr einfach. Sie instanziierten hierfür zwei Controller-Objekte, die Sie jeweils mit Hilfe der Funktion `ConnectXXX` initialisieren. Im unteren Beispiel wird eine Verbindung mit den Controllern E-871 und C-867 aufgebaut. Der E-871 wird über RS-232 verbunden, der C-867 über USB.

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

controllerSerialNumber = '109036090';
C867 = Controller.ConnectUSB(controllerSerialNumber);
```

Die Verbindungen können einzeln getrennt werden:

```
%% Close the connection
C867.CloseConnection;
E871.CloseConnection;

%% Unload the MATLAB-Treiber
Controller.Destroy;
clear Controller;
clear C867;
clear E871;
```

## 6.3 Daisy-Chain

Der MATLAB-Treiber ermöglicht auch die Ansteuerung mehrerer PI-Controller, die über Daisy-Chain miteinander verbunden sind<sup>7</sup>. Hierfür wird zunächst mit `Controller = Controller.OpenUSBdaisyChain('0123456789');` eine Verbindung zum Controller aufgebaut, der direkt mit dem PC verbunden ist. Anschließend wird dieser Controller als Daisy-Chain Controller mit seiner Daisy-Chain-Adresse angesprochen:

```
Controller1=Controller.ConnectDaisyChainDevice(controller1_HardwareAdress);
```

Ab diesem Zeitpunkt kann auf den Controller über das Controller-Objekt `Controller1` zugegriffen werden. Die Verbindung mit dem zweiten Controller (der über RS-232 mit dem ersten Controller verbunden ist) wird durch den gleichen Befehl aufgebaut, nur die Daisy-Chain-Adresse wird geändert.

<sup>7</sup> Momentan wird nur der folgende Verbindungstyp unterstützt: PC –Controller1 über USB.

Nicht unterstützt ist momentan: PC → Controller1 über RS-232. Sollten Sie diese Implementierung benötigen, wenden Sie sich an [support-software@pi.ws](mailto:support-software@pi.ws).

```
%% Set up the USB communication
Controller = Controller.OpenUSBChain('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()
```

Die Verbindungen können einzeln geschlossen werden. Ein Controller mit geschlossener Verbindung ist allerdings weiterhin in der Lage, Befehle an andere Controller weiterzureichen. Das Unloaden des Treibers funktioniert wie bei den vorherigen Beispielen.

```
%% Disconnet the controlles
Controller1.CloseConnection();
Controller2.CloseConnection();

%% Unload the MATLAB-Treiber
Controller.Destroy;
clear Controller;
clear Controller1
clear Controller2
```

## 7 Tipps & Tricks

### 7.1 Mitloggen von GCS2-Befehlen in PIMikroMove bei der Initialisierung

In vielen Fällen kann es hilfreich sein, die GCS2-Befehle, die PIMikroMove während der Initialisierung an den Controller sendet, mitzuloggen.

#### Schritt 1

Führen Sie im Fenster „Start up controller“ die **Schritte 1 bis 3** durch (vgl. Bild 5) und klicken Sie anschließend auf die Schaltfläche „Close“, ohne in **Schritt 3** „Start up axes“ eine Referenzierung durchzuführen.

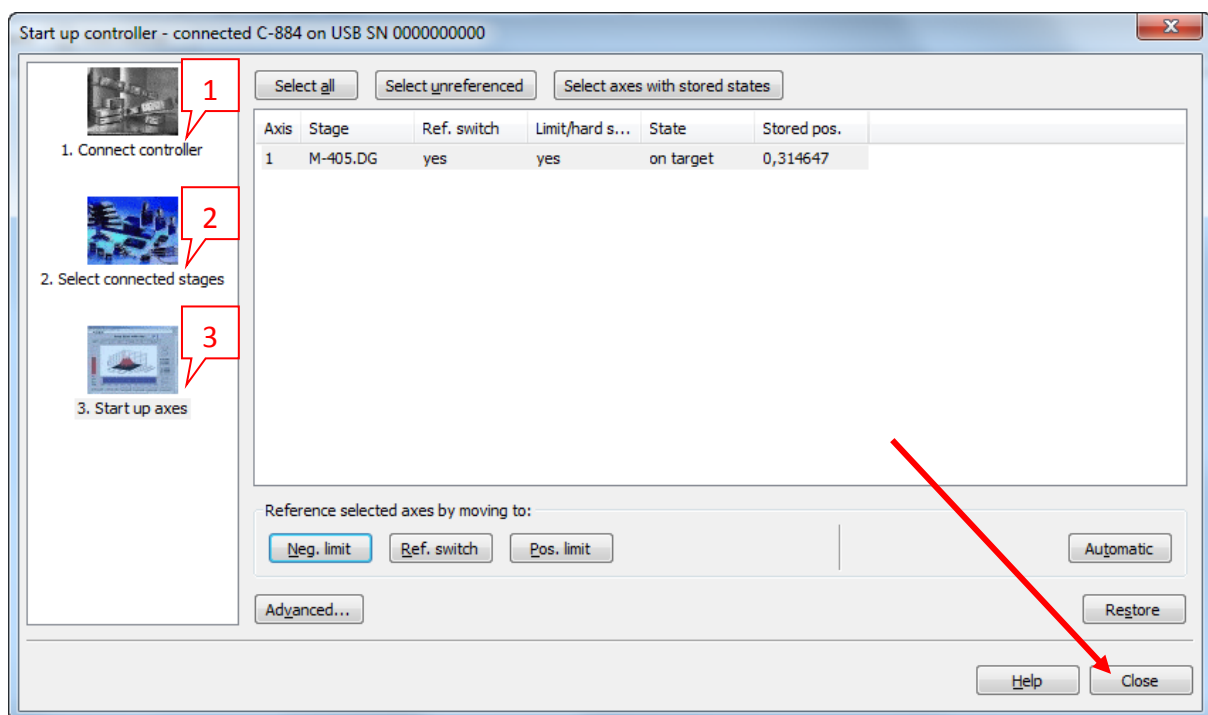
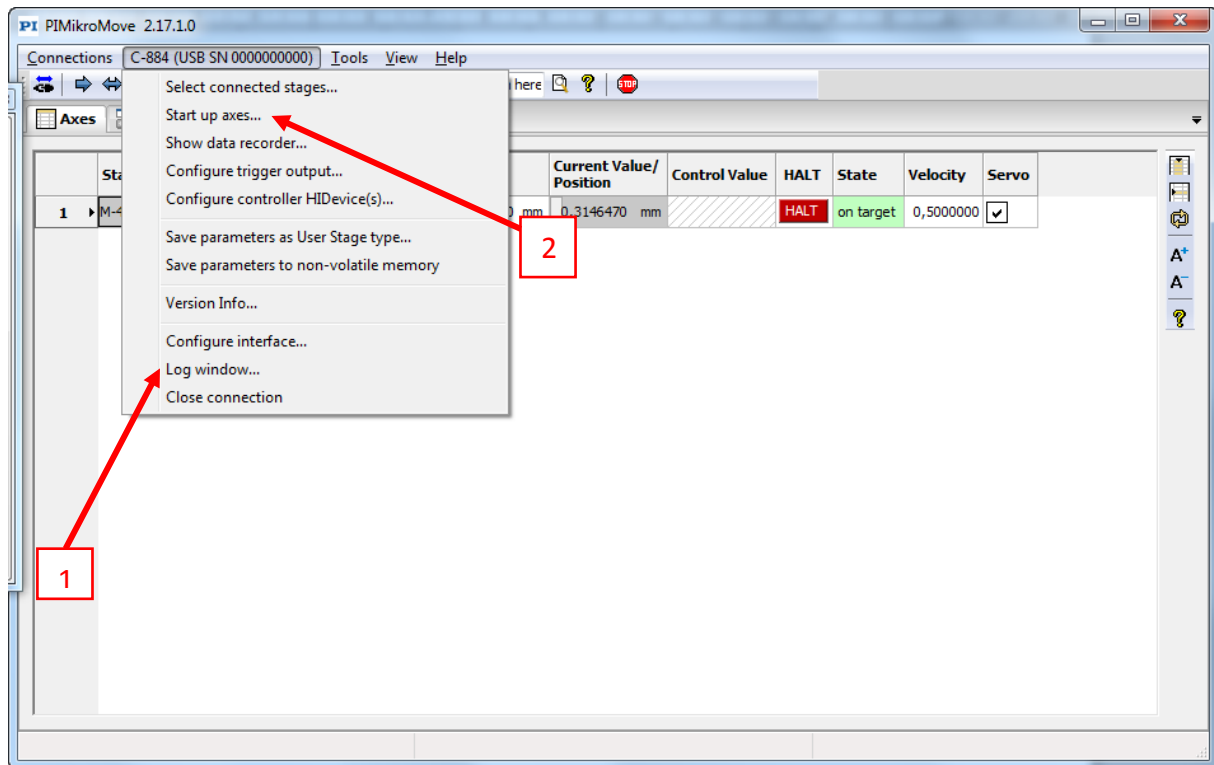


Bild 5 Start up controller



## Schritt 2

Nutzen Sie das „Log Window“, um die vom PC an den Controller gesendeten GCS2-Befehle sowie die Antworten des Controllers mitzuloggen (vgl. **Schritt 1** in Bild 6). Öffnen Sie anschließend „Start up axes...“ (vgl. **Schritt 2** in Bild 6). Für zusätzliche Informationen über das Loggen nutzen Sie die PIMikroMove Hilfe für das Suchstichwort „Log Window“.



**Bild 6** Start up axes

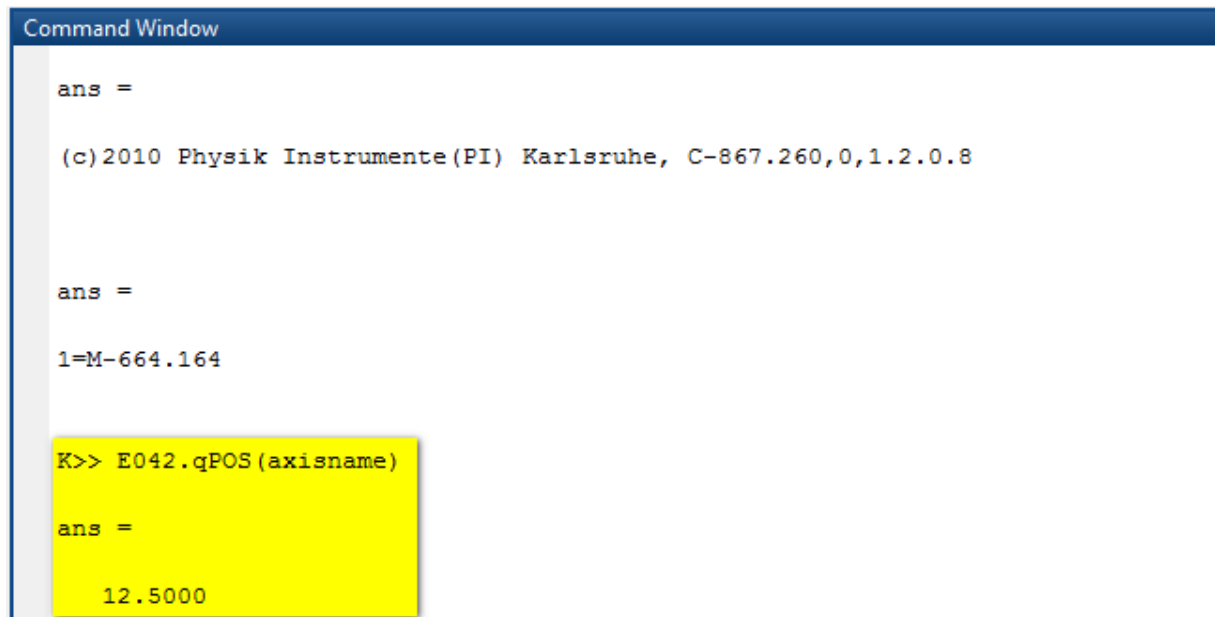
## 7.2 Shortcuts erstellen in MATLAB

Für häufig wiederkehrende Aufgaben, z. B. das Laden des Treibers oder den Verbindungsaufbau und -abbau, kann es sinnvoll sein, einen Shortcut in MATLAB zu erstellen. Wie Sie einen Shortcut erstellen, wird von MathWorks auf der folgenden Internetseite beschrieben:

[http://de.mathworks.com/help/matlab/matlab\\_env/create-matlab-shortcuts-to-rerun-commands.html?refresh=true](http://de.mathworks.com/help/matlab/matlab_env/create-matlab-shortcuts-to-rerun-commands.html?refresh=true)

## 7.3 Nutzung des „Command Window“ in MATLAB

Das Controller-Objekt kann nach erfolgreicher Initialisierung und Referenzierung des Systems natürlich auch direkt über das „Command Window“ von MATLAB benutzt werden.



```

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
  
```

**Bild 7** Direkte Benutzung des Controller-Objekts über das „Command Window“

## 8 Störungsbehebung

### 8.1 Das MATLAB-Skript erzeugt Fehlermeldungen, wenn man es auf einem anderen PC startet

#### Stelle im Skript, an der der Fehler auftritt

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

#### Erklärung

Der Zugriff von MATLAB auf den MATLAB-Treiber oder die PI GCS2 DLL funktioniert nicht richtig oder kann nicht gefunden werden.

#### Lösung

Installieren Sie alle notwendige Software, indem Sie der Installationsanleitung in Abschnitt 3.1 – „Standardinstallation (empfohlen)“ folgen. Starten Sie ggf. anschließend den , um Ihre Software auf den aktuellsten Stand zu bringen. Wenn Sie ein Installationsprogramm ausführen, das vor dem Jahr 2015 erstellt wurde, müssen Sie den MATLAB-Treiber manuell nachinstallieren (siehe Abschnitt 9.1 – „Umstieg auf Version 1.2.0“).

### 8.2 Fehler beim Laden der PI GCS2 DLL

#### Fehlermeldung

Neue Version (MATLAB-Treiber Version 1.2.0 oder höher):

```
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:
```

```
...
...
```

Alte Version (MATLAB-Treiber ohne Installer):

```
Loading dll ...
```

```
There was an error ...
```

```
...
...
```

## Erklärung

Bei diesem Fehler gibt es verschiedene mögliche Ursachen. Genauere Informationen zur Fehlerursache werden nach folgender Textpassage angezeigt: „Additional Information: MATLAB itself raised the following error:“.

### 8.2.1 PI GCS2 DLL wird nicht gefunden

#### Fehlermeldung

```
...
...
```

```
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.
```

## Erklärung

Die Datei „PI\_GCS2\_DLL.dll“ (bzw. „PI\_GCS2\_DLL\_x64.dll“) wurde nicht gefunden.

## Lösung

Installieren Sie alle notwendige Software, indem Sie der Installationsanleitungen in Abschnitt 3.1 – „Standardinstallation (empfohlen)“ folgen. Starten Sie ggf. anschließend den PI Update Finder, um Ihre Software auf den aktuellsten Stand zu bringen.

### 8.2.2 MATLAB findet keinen passenden C-Compiler

#### Fehlermeldung

```
...
...
```

```
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(ccin
clude,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();

```

## Erklärung

MATLAB macht eine Art Recompilierung der PI GCS2 DLL. Dafür benötigt MATLAB einen C-Compiler, der nicht standardmäßig in MATLAB konfiguriert bzw. integriert ist.

## Lösung

Im MATLAB Command Window wird das Konfigurations-Setup mit dem folgenden Befehl aufgerufen:

```
>> mex -setup
```

Die Konfiguration des C-Compilers unterscheidet sich zwischen der 32-Bit und der 64-Bit Variante von MATLAB.

### MATLAB 32-Bit

In der 32-Bit Variante von MATLAB bekommen Sie immer eine Liste von C-Compilern, da MATLAB einen 32-Bit C-Compiler mitliefert. Beispielsweise können Sie den Compiler „Lcc-win32 v2.4.1“ verwenden. Nach erfolgreicher Konfiguration sollten Sie folgende Antwort erhalten:

```

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

```

## MATLAB 64-Bit

In der 64-Bit Variante von MATLAB gibt es zwei Möglichkeiten.

1. Sie bekommen eine Liste mit mindestens einem C-Compiler. Konfigurieren Sie MATLAB so, dass einer dieser Compiler verwendet wird. Anschließend sollten Sie die folgende Antwort erhalten:

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

2. Sie erhalten die folgende Fehlermeldung:

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

Dieser Fehler tritt auf, weil MATLAB keinen 64-Bit-C-Compiler mitliefert. Einen solchen Compiler müssen Sie manuell nachinstallieren. Es gibt zahlreiche Möglichkeiten, einen solchen Compiler kostenlos herunterzuladen. MATLAB liefert auf der folgenden Internetseite eine umfassende Liste von möglichen Compilern:

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

Wir haben das Microsoft Windows SDK erfolgreich getestet, das Sie hier herunterladen können:

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

Installieren Sie das SDK und konfigurieren Sie in MATLAB anschließend den Microsoft Visual C Compiler:

```
>> mex -setup
```

Anschließend sollten Sie die folgende Antwort erhalten:

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

Eine alternative Möglichkeit ist die Installation einer IDE mit integriertem C-Compiler und anschließender Konfiguration dieses Compilers.

## 8.2.3 MATLAB meldet "PI\_GCS2\_DLL is not a valid Win32 application"

### Fehlermeldung

```
...  
...  
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.
```

### Erklärung

Dieser Fehler tritt nur bei älteren Versionen des MATLAB-Treibers auf (Versionsnummer kleiner als V1.2.0 – ohne Installer). Hier kann der Fehler auftreten, dass der MATLAB-Treiber unter MATLAB in der 64-Bit-Version die PI GCS2 DLL in der 32-Bit-Version lädt.

### Lösung

Steigen Sie auf den neuen MATLAB-Treiber mit der Versionsnummer 1.2.0 oder höher um. (siehe Abschnitt 9.1 – „Umstieg auf Version 1.2.0“).

Aktualisieren Sie Ihr Skript, das den MATLAB-Treiber benutzt, wie in Abschnitt 9.2 – „Hinweise zum Umstieg“ beschrieben.

## 8.3 MATLAB stürzt unerwartet ab

### 8.3.1 Fehler

Beim Laden oder Anwenden des Treibers stürzt MATLAB unerwartet ab. Es erscheint die nachfolgend dargestellte Fehlermeldung:

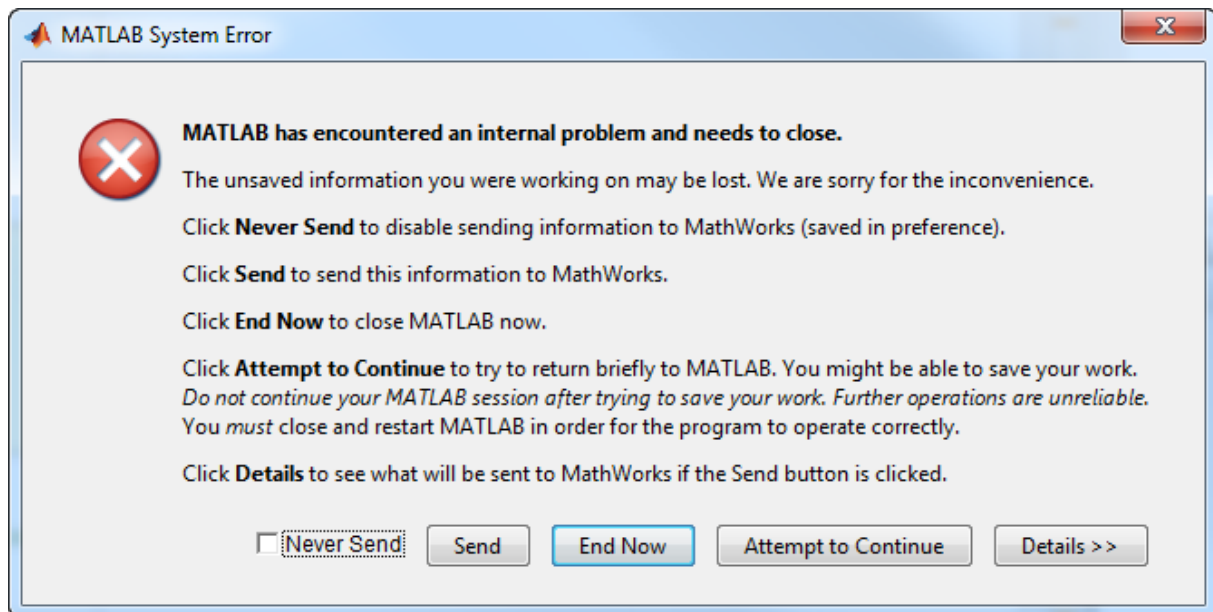


Bild 8 MATLAB-Error message

### 8.3.2 Ursache

Dieser Fehler tritt nur bei älteren Versionen des MATLAB-Treibers auf (Versionsnummer kleiner als V1.2.0). Wahrscheinlich fehlt Ihnen in Ihrer Header-Datei ein Include von `<windows.h>`.

### 8.3.3 Lösung

#### Variante 1 (empfohlen)

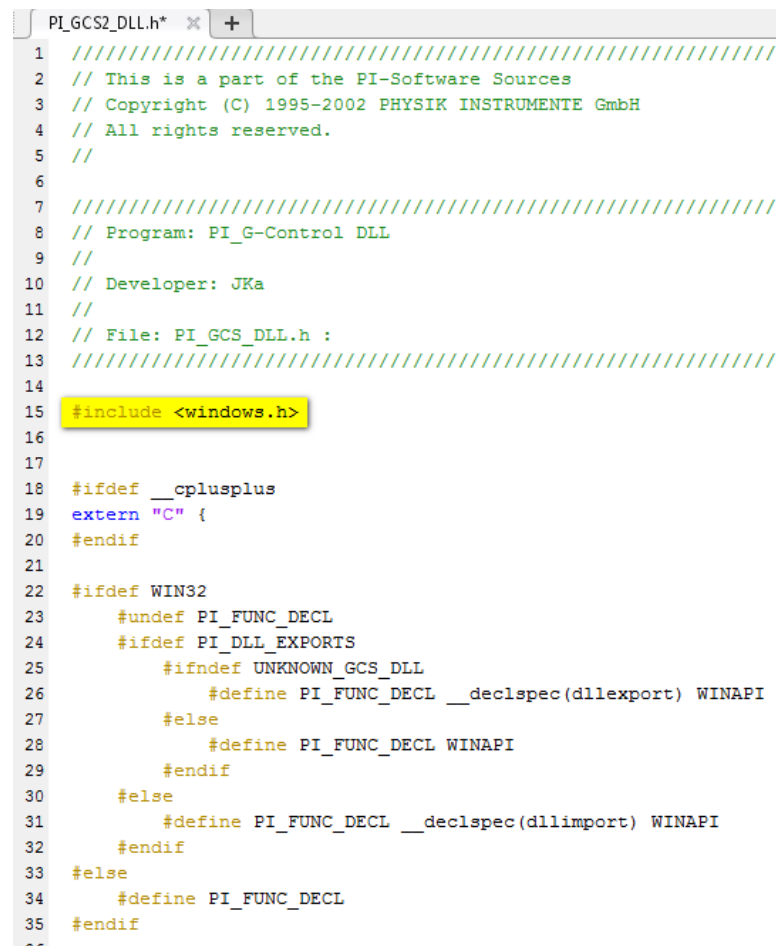
Steigen Sie auf den neuen MATLAB-Treiber mit der Versionsnummer 1.2.0 oder höher um. (siehe Abschnitt 9.1 – „Umstieg auf Version 1.2.0“).

Aktualisieren Sie Ihr Skript, das den MATLAB-Treiber benutzt, wie in Abschnitt 9.2 – „Hinweise zum Umstieg“ beschrieben.



## Variante 2

Wenn Sie die verwendete Header-Datei kennen, können Sie die Include-Anweisung selbst hinzufügen, wie in der nachfolgenden Abbildung dargestellt.



```

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

```

**Bild 9** Hinzugefügte Anweisung in der Header-Datei: `#include <windows.h>` Header.

## 8.4 „Error using calllib“

### 8.4.1 Fehler

```
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 Ursache

Der MATLAB-Treiber ist nicht mehr in MATLAB geladen.

### 8.4.3 Lösung

Wahrscheinlich haben Sie `Controller.Destroy` ausgeführt und anschließend ein Controller-Objekt verwendet, z. B. E042, oder auch das Treiber-Objekt „Controller“. Führen Sie `Controller.Destroy` erst dann aus, wenn Sie die MATLAB-Treiber nicht mehr verwenden wollen.

Sie können den MATLAB-Treiber erneut laden, indem Sie die den folgenden MATLAB-Befehl ausführen:

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

Anschließend muss die Verbindung zum Controller erneut aufgebaut werden.

## 9 PI MATLAB Driver GCS2: Umstieg auf Version 1.2.0

Die Vorteile der neuen Version sind:

- **Alte Samples und Skripte sind mit der Version 1.2.0 kompatibel**, wenn Sie die Schritte in Abschnitt 9.1 – „Umstieg auf Version 1.2.0“ beachten.
- Erweiterter Funktionsumfang, bessere Dokumentation.
- Automatische Unterstützung von 32-Bit- und 64-Bit-MATLAB-Versionen.
- Vereinfachtes Herunterladen von Updates durch den PI Update Finder.
- Besserer und schnellerer Support durch einheitliche Installation.
- Dadurch einfacher Umstieg auf andere Windows-PCs durch einen Installer.

### 9.1 Umstieg auf Version 1.2.0

Der Umstieg ist sehr einfach möglich. Führen Sie die folgenden vier Schritte durch:

- **Schritt 1**  
Laden Sie sich die Installationsdatei „PI\_MATLAB\_Driver\_GCS2\_Setup.exe“ und die MATLAB-Samples „PI Matlab Driver Samples<Date>.zip“ herunter.  
Wenden Sie sich an [support-software@pi.ws](mailto:support-software@pi.ws), um die Zugangsdaten für den Download-Server zu erhalten.
- **Schritt 2**  
Öffnen Sie „PI\_MATLAB\_Driver\_GCS2\_Setup.exe“, um die Treiber-Installation zu starten.
- **Schritt 3**  
Starten Sie das Sample „BeginWithThisSampleForTestingMatlabDriver.m“. Das Sample darf **nicht** mit dem Ordner „@PI\_GCS\_Controller“ in einem Ordner liegen (siehe 9.3 – Gegenüberstellung von alter und neuer Struktur). Das Sample sollte fehlerfrei<sup>8</sup> durchlaufen und die Versionsnummer des installierten MATLAB-Treibers zurückgeben.

---

<sup>8</sup> Sollten Fehler auftreten, stellen Sie sicher, dass alle Komponenten korrekt installiert sind (siehe Abschnitt 3.1 – „Standardinstallation (empfohlen)“) und stellen Sie durch den PI Update Finder sicher, dass Ihre Software auf dem neuesten Stand ist. Suchen Sie in Kapitel 8 – „Störungsbehebung“ nach Lösungen für bekannte Fehler.

- **Schritt 4**

Fügen Sie Ihr bisher verwendetes MATLAB-Sample bzw. Ihr MATLAB-Skript in das Sample „BeginWithThisSampleForTestingMatlabDriver.m“ ein (wie im nächsten Abschnitt beschrieben).

## 9.2 Hinweise zum Umstieg

### Einfügen eines alten Samples in „BeginWithThisSampleForTestingMatlabDriver.m“

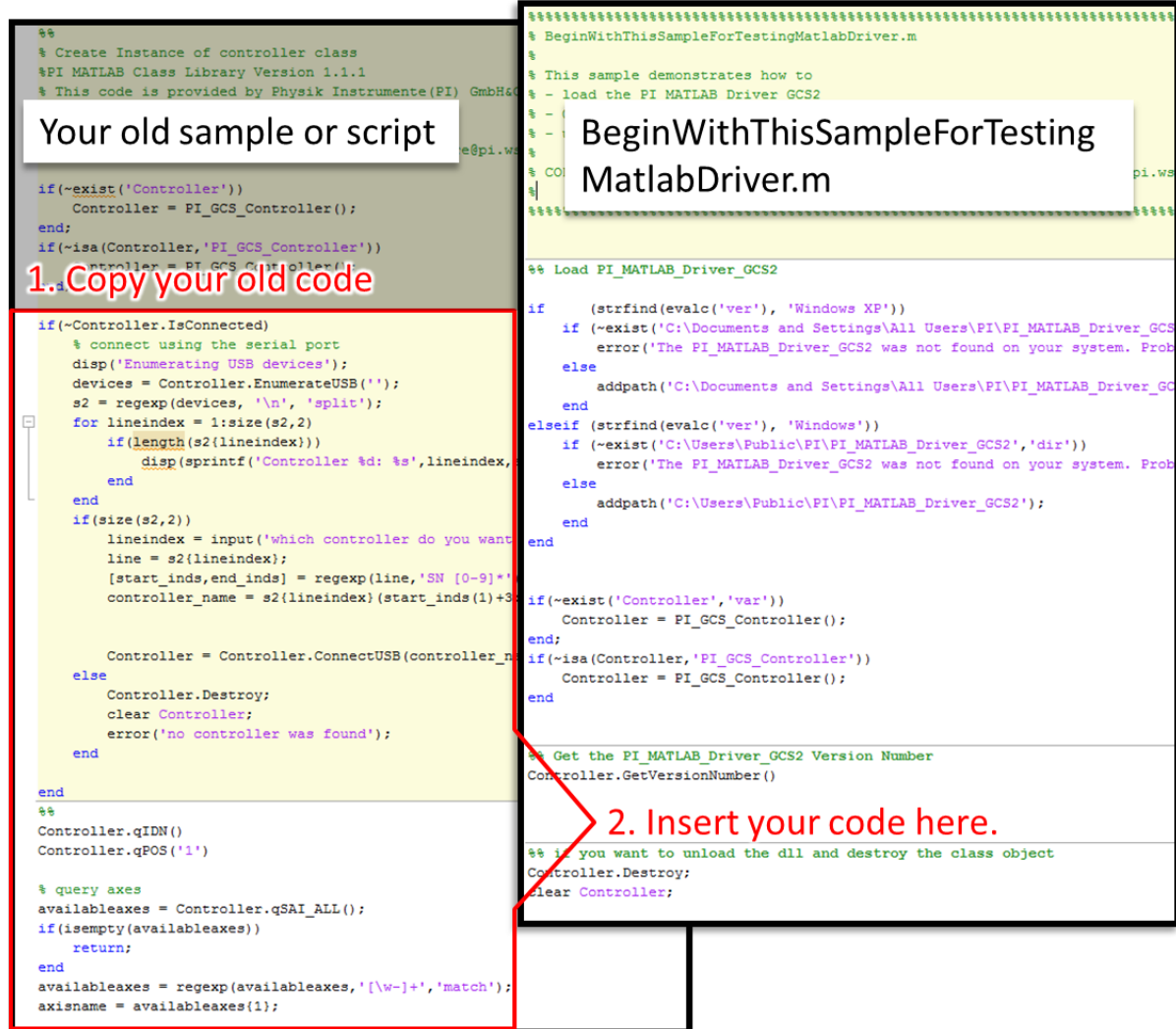
Das Einfügen ist in Bild 10 dargestellt:

Kopieren Sie den kompletten alten Programmcode, außer dem Teil zu Beginn des Skripts, in dem der MATLAB-Treiber geladen wird.

Fügen Sie den alten Programmcode in Ihr neues Skript „BeginWithThisSampleForTesting-MATLABDriver.m“ ein.

Achten Sie darauf, dass am Ende des neuen Skripts der Programmcode `Controller.Destroy;` und `clear Controller;` nicht doppelt vorhanden ist.

Stellen Sie sicher, dass Ihr neues Skript nicht mit dem Ordner „@PI\_GCS\_Controller“ in einem Ordner liegt.








**Bild 10** Beispielhaftes Einfügen von altem Quellcode in „BeginWithThisSampleForTestingMatlabDriver.m“

## 9.3 Gegenüberstellung von alter und neuer Struktur

### Struktur vor Version 1.2.0






Der Treiber, die verwendete PI GCS2 DLL mit zugehörigem h.-File (Header-Datei) und das eigentliche Sample mussten in einem gemeinsamen Ordner liegen (vgl. Bild 11). Das eigentliche Sample hatte meist den Namen „FirstSample.m“. Der Treiber bestand i. d. R. aus den MATLAB-Dateien 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

**Bild 11** Beispielhafte Struktur des alten MATLAB-Treibers vor der Version V1.2.0

### Struktur ab Version 1.2.0

Ab Version 1.2.0 sind die drei Komponenten – Samples bzw. Skripte, MATLAB-Treiber und PI GCS2 DLL – getrennt voneinander gespeichert. Im Samples Ordner befinden sich nur noch die Beispiele. Wenn der MATLAB-Treiber installiert ist, kann jedes Stand-Alone-Sample von jedem beliebigen Ort auf dem System gestartet werden. Dadurch wird die Handhabung der Skripte deutlich einfacher.

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

**Bild 12** Sample-Ordner

Der eigentliche Treiber wird durch den Installer in ein eigenes Verzeichnis installiert (vgl. Abschnitt 3.3 – „Installationsort“). Dadurch ist er nur einmal auf dem System vorhanden und kann gezielt durch ein Update aktualisiert werden.

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

**Bild 13** Dateipfad des MATLAB-Treibers

Die genutzte PI GCS2 DLL liegt im Verzeichnis „C:\ProgramData\PI\GCSTranslator“ und wird auch von anderen Anwendungen genutzt, z. B. von PIMikroMove. Dadurch muss für die PI GCS2 DLL nur ein Update durchgeführt werden, und es liegen nicht verschiedene, ggf. veraltete Versionen auf Ihrem System.

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

**Bild 14** Dateipfad „C:\ProgramData\PI\GCSTranslator“ der PI GCS2 DLL

## Weitere Unterschiede

Ein weiterer Unterschied zwischen den alten und den neuen Samples besteht darin, dass es meist keine Trennung zwischen Treiber-Objekt und Controller-Objekt gab:

### Alte Version:

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

### Neue Version:

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

Das alte Sample ist trotzdem voll funktionsfähig.

Bei Bedarf kann die Unterscheidung zwischen Treiber-Objekt und Controller-Objekt im alten Sample nachgezogen werden. Sie ist jedoch nicht zwingend erforderlich. Die neu eingeführte Trennung in Treiber-Objekt und Controller-Objekt hat den Vorteil, dass eine klare Abgrenzung zwischen Treiber-Funktionalität und Controller-Funktionalität vorgenommen werden kann (siehe auch Abschnitt 4.1.3 – „Aufbau einer Verbindung“). Wichtig wird dies vor allem bei der Ansteuerung von mehreren Controllern aus einem MATLAB-Skript heraus.