Häufig gestellte Fragen

netSCRIPT

Bewährtes und Praktisches

Hilscher Gesellschaft für Systemautomation mbH

www.hilscher.com

DOC100208FAQ02DE | Revision 2 | Deutsch | 2010-03 | Freigegeben | Öffentlich

Inhaltsverzeichnis

1 Einleitung

1.1 Über dieses Handbuch

In diesem Handbuch können Sie die am häufigsten gestellten Fragen aus allen Bereichen zum Thema netSCRIPT und die dazugehörigen Antworten nachlesen.

1.1.1 Änderungsübersicht

Index	Datum	Kapitel	Änderungen
1	2010-02-10	Alle	Erstellt
2	2010-03-12	2.2	Performanceaussagen hinzugefügt
			Funktionen mit mehrere Rückgabewerten
			Antwort des `return` Kommandos erweitert
		2.3	Erklärung Verhalten bei schweren Fehlern
		2.4	Einarbeitung der Defaultzykluszeit des Debuggers

Tabelle 1: Änderungsübersicht

1.1.2 Konventionen in diesem Handbuch

Handlungsanweisungen, ein Ergebnis eines Handlungsschrittes bzw. Hinweise sind wie folgt gekennzeichnet:

Handlungsanweisungen:

➢ <Anweisung>

oder

1. <Anweisung>
2. <Anweisung>

Ergebnisse:

⇘ <Ergebnis>

Hinweise:

Wichtig: <Wichtiger Hinweis>

Hinweis: <Hinweis>

<Hinweis, wo Sie weitere Informationen finden können>

1.2 Rechtliche Hinweise

1.2.1 Copyright

1.2.2 Wichtige Hinweise

Das Benutzerhandbuch, Begleittexte und die Dokumentation wurden mit größter Sorgfalt erarbeitet. Fehler können jedoch nicht ausgeschlossen werden. Eine Garantie, die juristische Verantwortung für fehlerhafte Angaben oder irgendeine Haftung kann daher nicht übernommen werden. Sie werden darauf hingewiesen, dass Beschreibungen in dem Benutzerhandbuch, den Begleittexte und der Dokumentation weder eine Garantie, noch eine Angabe über die nach dem Vertrag vorausgesetzte Verwendung oder eine zugesicherte Eigenschaft darstellen. Es kann nicht ausgeschlossen werden, dass das Benutzerhandbuch, die Begleittexte und die Dokumentation nicht vollständig mit den beschriebenen Eigenschaften, Normen oder sonstigen Daten der gelieferten Produkte übereinstimmen. Eine Gewähr oder Garantie bezüglich der Richtigkeit oder Genauigkeit der Informationen wird nicht übernommen.

Wir behalten uns das Recht vor, unsere Produkte und deren Spezifikation, sowie zugehörige Benutzerhandbücher, Begleittexte und Dokumentationen jederzeit und ohne Vorankündigung zu ändern, ohne zur Anzeige der Änderung verpflichtet zu sein. Änderungen werden in zukünftigen Manuals berücksichtigt und stellen keine Verpflichtung dar; insbesondere besteht kein Anspruch auf Überarbeitung gelieferter Dokumente. Es gilt jeweils das Manual, das mit dem Produkt ausgeliefert wird.

Die Hilscher Gesellschaft für Systemautomation mbH haftet unter keinen Umständen für direkte, indirekte, Neben- oder Folgeschäden oder Einkommensverluste, die aus der Verwendung der hier enthaltenen Informationen entstehen.

1.2.3 Haftungsausschluss

Die Software wurde von der Hilscher Gesellschaft für Systemautomation mbH sorgfältig erstellt und getestet und wird im reinen Ist-Zustand zur Verfügung gestellt. Es kann keine Gewährleistung für die Leistungsfähigkeit und Fehlerfreiheit der Software für alle Anwendungsbedingungen und -fälle und die erzielten Arbeitsergebnisse bei Verwendung der Software durch den Benutzer übernommen werden. Die Haftung für etwaige Schäden, die durch die Verwendung der Hard- und Software oder der zugehörigen Dokumente entstanden sein könnten, beschränkt sich auf den Fall des Vorsatzes oder der grob fahrlässigen Verletzung wesentlicher Vertragspflichten. Der Schadensersatzanspruch für die Verletzung wesentlicher Vertragspflichten ist jedoch auf den vertragstypischen vorhersehbaren Schaden begrenzt.

Es ist strikt untersagt, die Software in folgenden Bereichen zu verwenden:

- für militärische Zwecke oder in Waffensystemen;
- zum Entwurf, zur Konstruktion, Wartung oder zum Betrieb von Nuklearanlagen;
- in Flugsicherungssystemen, Flugverkehrs- oder Flugkommunikations-systemen;
- in Lebenserhaltungssystemen;
- in Systemen, in denen Fehlfunktionen der Software körperliche Schäden oder Verletzungen mit Todesfolge nach sich ziehen können.

Sie werden darauf hingewiesen, dass die Software nicht für die Verwendung in Gefahrumgebungen erstellt worden ist, die ausfallsichere Kontrollmechanismen erfordern. Die Benutzung der Software in einer solchen Umgebung geschieht auf eigene Gefahr; jede Haftung für Schäden oder Verluste aufgrund unerlaubter Benutzung ist ausgeschlossen.

1.2.4 Gewährleistung

Obwohl die Hard- und Software mit aller Sorgfalt entwickelt und intensiv getestet wurde, übernimmt die Hilscher Gesellschaft für Systemautomation mbH keine Garantie für die Eignung für irgendeinen Zweck, der nicht schriftlich bestätigt wurde. Es kann nicht gewährleistet werden, dass die Hard- und Software Ihren Anforderungen entspricht, die Verwendung der Software unterbrechungsfrei und die Software fehlerfrei ist. Eine Garantie auf Nichtübertretung, Nichtverletzung von Patenten, Eigentumsrecht oder Freiheit von Einwirkungen Dritter wird nicht gewährt. Weitere Garantien oder Zusicherungen hinsichtlich Marktgängigkeit, Rechtsmängelfreiheit, Integrierung oder Brauchbarkeit für bestimmte Zwecke werden nicht gewährt, es sei denn, diese sind nach geltendem Recht vorgeschrieben und können nicht eingeschränkt werden. Gewährleistungsansprüche beschränken sich auf das Recht, Nachbesserung zu verlangen.

1.2.5 Exportbestimmungen

Das gelieferte Produkt (einschließlich der technischen Daten) unterliegt den gesetzlichen Export- bzw. Importgesetzen sowie damit verbundenen Vorschriften verschiedener Länder, insbesondere denen von Deutschland und den USA. Die Software darf nicht in Länder exportiert werden, in denen dies durch das US-amerikanische Exportkontrollgesetz und dessen ergänzender Bestimmungen verboten ist. Sie verpflichten sich, die Vorschriften strikt zu befolgen und in eigener Verantwortung einzuhalten. Sie werden darauf hingewiesen, dass Sie zum Export, zur Wiederausfuhr oder zum Import des Produktes unter Umständen staatlicher Genehmigungen bedürfen.

1.2.6 Eingetragene Warenzeichen

Windows® 2000 und Windows® XP sind eingetragene Warenzeichen der Microsoft Corporation.

Alle anderen erwähnten Marken sind Eigentum Ihrer jeweiligen rechtmäßigen Inhaber.

2 Häufig gestellte Fragen

2.1 Vorwort - netSCRIPT Erfolgserlebnis in wenigen Minuten

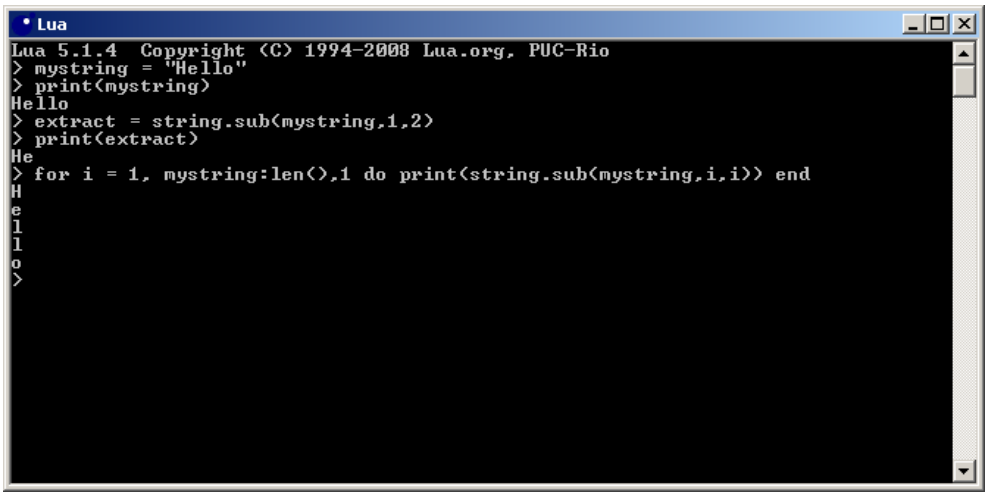
Die folgenden Schritte führen in wenigen Minuten zum ersten netSCRIPT Erfolgserlebnis. Der gesamte Ablauf befindet sich als Video-Tutorial auf der mitgelieferten Geräte-DVD.

1. Ein Gateway NT 100-XX-RS mit serieller Schnittstelle erwerben
2. Das Gateway und den PC über die USB-Konfigurationsschnittstelle verbinden
3. Das Gateway mit der seriellen Schnittstelle eines freien COM Port des PC verbinden
4. Das Windows Hyperterminal-Programm auf diesem PC aufstarten und mit 115200,8,N,1 den COM-Port konfigurieren
5. Das Konfigurationswerkzeug SYCON.net installieren und es starten
6. Das richtige Gateway im Gerätekatalog auswählen und die entsprechende netSCRIPT-Firmware einspielen
7. Den netSCRIPT Debugger installieren und starten
8. Das „Hello World“ Beispiel laden, ins Gateway einladen und anstarten
9. Den Empfang des Textes im Hyperterminal-Programm prüfen

2.2 Fragen zu den Sprachbesonderheiten

Kann man netSCRIPT bzw. Lua auch unter Windows testen?

Ja, unter http://luaforwindows.luaforge.net/ kann man einen Lua Interpreter herunterladen um die Sprache Lua zu unter Windows ganz ohne weiterere Hardware zu testen.



Welche Unterschiede von netSCRIPT zur Sprache „C“ sind am auffälligsten?

Bei der Variablendeklaration muss kein Datentyp angegeben werden. Es gibt keine Header-Files und kein Befehl `#define`. Definitionen werden immer als Variablen angelegt.

Wie erzeugt man eine Variable?

Alle Zeichenfolgen im Quellcode die von den Schlüsselwörtern der Sprache selbst abweichen, werden als Namen behandelt. So ist `whereIam` oder `I_AM_HERE` zunächst nur ein Name. Ein Name wird zur Variablen, wenn eine Zuweisung erfolgt. `whereIam = “I am here“` erzeugt die Variable `whereIam` des Datentyps `string`.

Wie werden Variable intern verwaltet?

Variable werden intern in einer Tabelle geführt. Eine Variable hat in ihrem Tabelleneintrag einen Namen und einen Wert. Der Wert wiederum führt den Datentyp. Existiert eine Variable nicht in der Tabelle, so hat sie den Wert `nil`.

Benötigen Variablen einen Datentyp?

Erzeugt eine Zuweisung eine Variable erhält sie den Datentyp implizit über ihren Wert. Eine Variable kann sogar ihren Typ bei der nächsten Zuweisung ändern.

Beispielcode:

```
mystring = “Hello“  -- is of type string
mynumber = 5 -- is of type number
dynvar = 5  -- is of type number now
dynvar = “Hello“ -- is of type string now
```

Gibt es Schlüsselwörter die nicht als Namen verwendet werden dürfen?

Ja. Insgesamt reserviert die Sprache 21 Schlüsselwörter, wie `if`, `then`, `else`, `and` usw. die man als Namen nicht verwenden darf.

Arbeitet netSCRIPT case-sensitiv?

Ja. Ein `hello` oder `Hello` sind zwei unterschiedliche Namen.

Müssen Funktionen immer alle Parameter mitgegeben werden?

Nein. Erwartete Parameter, die nicht angegeben werden, werden beim Aufruf auf `nil` gesetzt.

Wie kann von einer Funktion vorzeitig zurückgekehrt werden?

Mit dem `return` Befehl. Befindet man sich im Hauptzweig des Skriptes bewirkt der Befehl das Verlassen des Skriptes. Erst nach Ablauf der Zykluszeit wird das Skript dann wieder erneut aufgerufen.

Kann eine Funktion auch mehrere Rückgabewerte haben?

Ja das ist möglich. Der Beispielcode zeigt drei Rückgabewerte der Funktion `PortIsExchangeDone()`.

Beispielcode:

```
status, RXDATA, rxerror = MyPort:PortIsExchangeDone()

if status == port.STA_CHAR_TIMEOUT and RXDATA and not rxerror then
   -- reception successful
end
```

Gibt es Aussagen zur Performance einzelner Befehle?

Ja, die folgende Tabelle gibt eine Übersicht:

Befehl	Zeit
Number assigned to variable `'a = 5'` :	1.56 usec/exec
Floating assigned to variable`'a = 3.1416'`	1.64 usec/exec
String assigned to variable `'a = ”Hello World”'`	1.68 usec/exec
Adding `'a = 2 + 3'`	1.62 usec/exec
Multiplying `'a = 2 * 3'`	1.56 usec/exec
Dividing `'a = 2 / 3'`	1.56 usec/exec
Simple if `'if a == nil then end'`	1.72 usec/exec
Call a simple function `util.SetLed('run', true)`	12.4 usec/exec
Call simple function via object `b:BusIOSetRun(false)`	9.6 usec/exec
Format a string `'string.format('%02d:%02d:%02d',11,22,33)'`	41.7 usec/exec
Parse string `'string.match('10:30:00','^([0-2][0-9]):?([0-5][0-9]):?([0-5][0-9])?$')'`	23.88 usec/exec

Tabelle 2: Performance

2.3 Fragen zum generellen Aufbau des Programms

Läuft das Skript immer zyklisch?

Ja das Skript wird beginnend mit der ersten Codezeile und beendend mit der letzten Zeile zyklisch aufgerufen.

Ist es erlaubt lange Schleifen mit Durchlaufzeit größer der Zykluszeit zu programmieren?

Ja. Die Anzahl der verpassten Zyklen werden intern gezählt und nachgeholt wenn das Skript wieder dem gewöhnlichen Ablauf folgt. Dabei wird das Skript so schnell wie möglich und ohne Einhaltung der Zykluszeit aufgerufen. Sind die Zyklen nachgeholt, folgt das Skript wieder dem Zyklusraster.

Wie vermeidet man, dass immer das gesamte Skript durchlaufen wird?

Das Skript muss in Phasen und Codesegmente unterteilt werden.

Welche Phasen sollte ein Skript unterscheiden?

Zwei Phasen.

A.) Die Initialisierungsphase enthält den Codeanteil der nur einmal durchlaufen werden soll

B.) Die Laufzeitphase mit dem Codeanteil der zyklisch abgearbeitet werden soll

Wie realisiert man die Initialisierungsphase?

Die Phasen sollten mit einer Zustandsvariablen unterschieden werden. Die Prüfung einer Variablen auf `nil` erkennt den Neuanlauf, da der Variablen noch kein Wert zugewiesen wurde

Beispielcode:

```
if mystate == nil then
  -- script is in init state
end
```

Wie realisiert man weitere Phasen oder ganze Schrittketten im Skript?

Durch Wertezuweisung der Zustandsvariablen mit anschließender Bedingungsprüfung.

Beispielcode:

```
if mystate == nil then
  -- script is in init state
  MY_STATE_INIT_DONE, MY_STATE_1, MY_STATE_2 = 1,2,3
  …
  mystate = MY_STATE_INIT_DONE

-- script is in running state
elseif mystate == MY_STATE_INIT_DONE then
    …
  mystate = MY_STATE_1
elseif mystate == MY_STATE_1 then
    …
  mystate = MY_STATE_1
elseif mystate == MY_STATE_2 then
    …
  mystate = MY_STATE_INIT_DONE
end
```

Gibt es Konstanten?

netSCRIPT im eigentlichen Sinne kennt keine Konstanten, nur Zuweisungen. Eine Zuweisung muss im Code einmal durchlaufen werden, um einen Wert anzunehmen.

Wo definiert man Konstanten?

Befehle wie `MY_STATE_INIT_DONE = 1` zur Festlegung konstanter Werte sollten in der Initialisierungsphase „definiert“ werden, da sie in der Abarbeitung nur einmal benötigt werden.

Wo definiert man Funktionen?

Eine Funktion kann an jeder Stelle im Skript definiert werden. Die Definition allerdings muss aber einmal durchlaufen worden sein, bevor man die Funktion aufrufen kann.

Welchen Wert haben nicht definierte Variablen?

Da Variable intern über Tabellen verwaltet werden, hat eine Variable ohne Wertzuweisung keine Tabelleneintrag und den Wert `nil`. Mit einer Zuweisung `myvariable = nil` wird eine Variable wieder aus der intern verwalteten Tabelle.

Was sollte alles in der Initialisierungsphase ausprogrammiert werden?

A.) Zuweisungen der konstanten Werte wie z.B. `MY_STATE_INIT_DONE = 1` die nur einmal im Ablauf benötigt werden. Auch Funktionsdefinitionen gehören dazu.

B.) Aufruf der Initialisierungsfunktionen der speziellen netSCRIPT Kommunikationsdienste wie `PortOpen()` die nur einmal im Ablauf benötigt werden.

Beispielcode:

```
if mystate == nil then
  MY_STATE_INIT_DONE, MY_STATE_1, MY_STATE_2 = 1,2,3 -- constants
  myfunction print() –- functions declarations
   ..
  end

  port = PortOpen()  -- call of initializing functions
  myio = BusIoOpen()
  …
  mystate = MY_STATE_INIT_DONE
else
```

Einige Funktionen liefern in der Variablen `lasterror` eine Fehlermeldung, ist diese zu prüfen?

Für eine sichere Kommunikation ist die Auswertung der Variablen `lasterror` zu empfehlen. Zu empfehlen ist auch die Fehlernummer an das überlagerte Netzwerk über eines der beiden Fehlerregister zu übergeben. So ist die Steuerung in der Lage ein Problem innerhalb des Skriptes zu erkennen.

Gib es Zeitgeber die man zusätzlich als Timer anprogrammieren kann?

Nein. Der einzige Takt ist der Zyklus in dem das Skript aufgerufen wird. Von einer globalen Variablen, die die Zyklen beispielsweise zählt, sind relative Zeiten durch Differenzbildung der Werte abzuleiten.

Beispielcode:

```
mytime = mytime + 1
…
if mystate == START then
  timesave = mytime
  mystate = WAIT_TIMEOUT
elseif mystate == WAIT_TIMEOUT
  timediff = mytime - timesave
  if timediff > 1000 then

  end
end
```

Welche Priorität hat netSCRIPT im Gesamtkontext des Gateways?

netSCRIPT hat im System eine niedrige Priorität. Vorrang hat immer die Abarbeitung des EA-Protokolls. Aufgrund der niedrigen Priorität ist netSCRIPT nicht in der Lage den EA-Protokollablauf zu stören oder gar zu unterbinden.

Wie soll sich das Skript verhalten bei schweren Laufzeitfehlern?

Es sollte die Funktion `assert()` verwendet werden. Ist der Prüfwert `nil` oder `false` wird das Skript gestoppt und ein zum Fehler übergebener Text hinterlegt, der im Debugger oder im SYCON.net auslesbar ist.

Beispielcode.

```
myPort = PortOpen()
assert(myPort, ”Opening the port failed”)
```

Ausgabe im netSCRIPT Debugger:

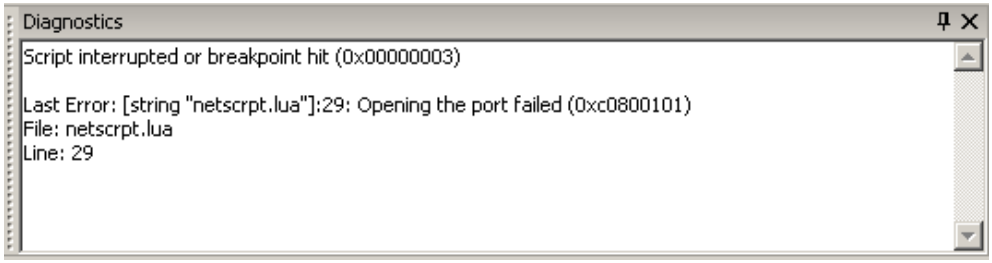


Abbildung 1: Ausgabe bei schweren Laufzeitfehlern im netSCRIPT Debugger

Ausgabe in SYCON.net

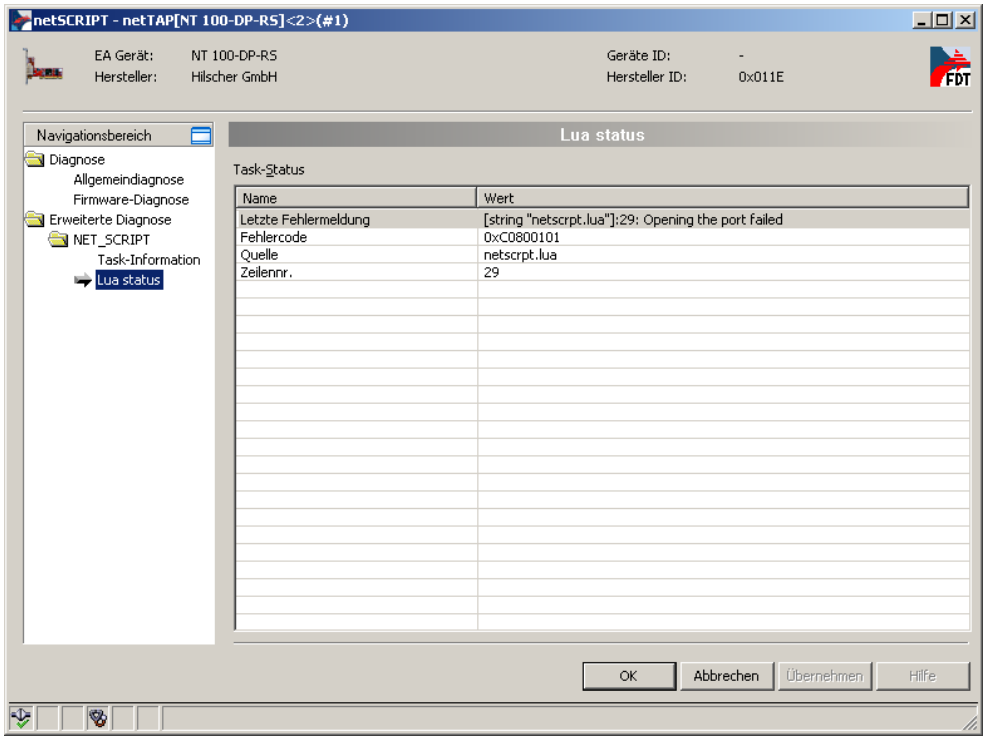


Abbildung 2: Ausgabe bei schweren Laufzeitfehlern in SYCON.net

2.4 Fragen zu SYCON.net und netSCRIPT-Debugger

Wieso gibt es zwei Werkzeuge um ein Skript zu laden?

SYCON.net und netSCRIPT-Debugger sind die beiden Werkzeuge im Umgang mit netSCRIPT. SYCON.net konfiguriert das gesamte Gateway. Er lädt das Skriptfile und die Parameter des übergeordneten Netzwerkes. Der netSCRIPT-Debugger wird temporär eingesetzt dient zur Fehlersuche bis das Skript-Programm fehlerfrei arbeitet. Das resultierende Skript ist am Ende mit SYCON.net zu laden.

In welcher Reihenfolge sollten die beiden Werkzeuge verwendet werden?

Es ist mit SYCON.net zu beginnen und die gesamte Gateway-Konfiguration zu erstellen. Damit wird das Gateway von der übergeordneten EA-Steuerung über den EA-Bus ansprechbar. Während dieser Konfiguration muss keine Skriptdatei angegeben werden. Zur Erstellung des Skriptes ist anschließend der netSCRIPT-Debugger zu verwenden. Ist die Skriptdatei fehlerfrei ist sie am Ende mit SYCON.net in das Gateway einzuspielen.

Wird das Skript mit SYCON.net oder netSCRIPT-Debugger erstellt?

SYCON.net hat einen Skript-Fileeditor, einen Syntaxchecker und kann ein Skriptfile zusammen mit der Netzwerkkonfiguration laden. SYCON.net besitzt keine Möglichkeit zum Debuggen eines Skriptes. Mit SYCON sollte daher immer nur ein fertiges und fehlerfreies Skript geladen werden.

Können SYCON.net und netSCRIPT-Debugger gleichzeitig über die USB Diagnoseschnittstelle mit dem Gateway kommunizieren?

Ja. Es ist möglich sich mit beiden Applikationen mit dem Gerät zu verbinden.

Muss ich als Sprachgeübter netSCRIPT-Debugger verwenden?

Nein, sprachgeübte verwenden nur SYCON.net und verzichten auf den netSCRIPT-Debugger.

Wo wird die Skript-Zykluszeit eingestellt?

Ausschließlich im SYCON.net Konfigurationswerkzeug. Der netSCRIPT-Debugger hat keine Möglichkeit die Zykluszeit einzustellen und dient nur der Fehlersuche. Er verwendet immer 10ms als Wert. Möchte man das Skript in seiner endgültigen Version die richtige Zykluszeit zukommen lassen, muss es in letzter Instanz über SYCON.net in das Gateway geladen werden.

Wird das per netSCRIPT Debugger zuletzt geladene Skript permanent im Gateway gespeichert?

Ja. Der netSCRIPT-Debugger bedient sich dem Dateisystem des Gateway's und speichert das Skriptfile nicht flüchtig.

Funktioniert der Zugriff von netSCRIPT-Debugger auch ohne geladene SYCON.net Konfiguration?

Ja. Mit dem netSCRIPT-Debugger kann jederzeit die Fehlersuche erfolgen. Ist allerdings keine gültige Konfiguration für das übergeordnete Netzwerk geladen, ist davon auszugehen das das EA-Netzwerk von

Was ist im netSCRIPT Debugger der Unterschied zwischen dem „Sync“ bzw. „Neu Laden“ Befehl?

Mit dem Sync-Befehl wird der aktuelle Quellcode neu in das Gateway eingespielt und der existierende überschrieben. Mit dem Neu Laden-Befehl wird nur der bereits im Gateway befindliche Quellcode neu gestartet und in die erste Quellcodezeile gesprungen. Gibt es dabei Unterschiede zwischen dem im Editor befindlichen Quellcode und dem Quellcode im Gateway, erzeugt der Debugger eine „remote copy“-Version. Nur diese kann dann nach Fehler untersucht werden.

Was bedeutet der Hinweis „remote copy“ im Editor-Fenster des Debuggers?

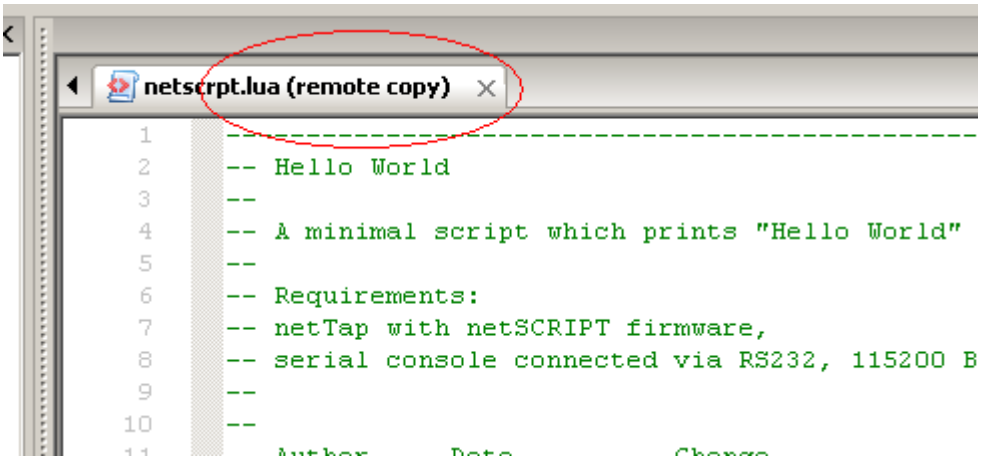


Abbildung 3: Angabe ‚remote copy’ im netSCRIPT Debugger

Die Bemerkung weist darauf hin, dass es sich bei dem hier angezeigten Quellcode um den derzeitig im Gateway geladenen Quellcode handelt. Der Debugger erzeugt den Hinweis dann, wenn der Editor Unterschiede zwischen dem aktuellen Quellcode im Editor und dem Quellcode im Gateway feststellt. Die „remote copy“-Version kann nicht editiert werden.

Wird auch die zyklische EA-Kommunikation des übergeordneten Netzwerkes gestoppt, wenn das Skript gestoppt wird?

Nein. Beide Prozesse laufen völlig unabhängig voneinander. netSCRIPT und sein Programmablauf ist niemals in der Lage das EA-Kommunikation zu stoppen oder zu unterbrechen. Die einzigen möglichen Zugriffe erlauben die `BusIOxx()` Funktionen über interne Puffer.

2.5 Fragen zu Optimierungsmaßnahmen

Kann man auch Zustandsautomaten über Funktionen programmieren?

Wie Variablen werden auch Funktionen intern über Tabellen verwaltet. Eine Funktion hat demnach einen Namen und einen Wert. Der Wert lässt sich zur Laufzeit verändern und so bestimmen, welche Funktion beim nächsten Aufruf aufgerufen wird.

Beispielcode:

```
if myfunc then
  myfunc()
else
-- defining all subfunctions
function my_state_init_done()
  ..
  myfunc = my_state_1
end

function my_state_1()
  ..
  myfunc = my_state_2
end

function my_state_2()
  ..
  myfunc = my_state_init_done
end

myfunc = my_state_init_done
end
```

Durchläuft eine `if-elseif`-Abfragenkette immer alle Prüfungen?

Nein, der netSCRIPT Interpreter optimiert. Eine Sequenz an `if-elseif` Anfragen wird mit einer erfüllten Bedingung verlassen und die restlichen übersprungen. Die Skriptabarbeitung wird also effektiver, wenn Bedingungen mit höherer Wahrscheinlichkeit ihres Eintretens zunächst prüft.

Wie unterscheidet sich eine lokale Variable von einer globalen?

Variable mit vorangestelltem `local` sind nur bis zum Ende des abgearbeiteten Blockes gültig in dem sie definiert wurden. Die Verwendung von lokalen Variablen hat einen leichten Geschwindigkeitsvorteil, weil der Interpreter diese Variablen über einen Index bzw. eine Adresse anspricht statt über ihren Namen.

Was passiert mit lokalen Variablen wenn das Skript am Ende angelangt ist?

Lokale Variable auf der Wurzelebene des Skriptes sind im nächsten Skriptdurchlauf nicht mehr existent.

2.6 Fragen zu Stringoperationen

Wie sind Variablen vom Typ `string` zu einem String zusammenzufügen?

Mit dem Operator `..`

Beispielcode:

```
string_a = “Hello“
string_b = “World“
result = string_a .. string b .. “2010“
-- will result in string “Hello World 2010“
```

Wie erhält man die Länge eines Strings?

Mit dem `:len()` Operator, mit vorangestellter Stringvariablen.

Beispielcode:

```
string_a = “Hello“
len = string_a:len()
-- will result in value 5 for variable len
```

Ein Teil eines Strings soll einer anderen Variablen zugewiesen werden, wie geht das?

Mit dem `string.sub()` Operator. Angegeben wird von welchem bis zu welchem Zeichen einschließlich, mit 1 beginnend zu zählen, die Zeichen extrahiert werden sollen. Negative Zahlen zählen von hinten, beginnend mit -1.

Beispielcode:

```
string_a = “Hello“
extract_a = string.sub(string_a,2,4)
-- will result in value “ell“ for variable extract
extract_b = string.sub(string_a,2,-2)
-- same result
```

Die seriellen Steuerzeichen liegen außerhalb des Alphabets, wie deklariert man diese?

Mit dem `string.char()` Operanden und Angabe von einem oder mehreren Werten in Dezimal- oder Hexadezimalangabe.

Beispielcode:

```
STX = string.char(0x02)
CR_LF = string.char(13, 10)
```

Wie wandelt man einen String in eine Echtzahl?

Mit dem `string.byte()` Operanden.

Beispielcode:

```
string.byte(”a”) -- will result a numeric value 97
string.byte(”abcde”,2) –- will result a numeric value 98 (”b”)
string.byte(”abcde”,-1) –- will result a numeric value 101 (”e”)
```

2.7 Fragen zur seriellen Kommunikation

Sind beim Öffnen der seriellen Ports alle Konfigurationsparameter zu übergeben?

Nein. Es werden nur diejenigen verändert, die angegeben sind. Ein simples `PortOpen()` ohne Parameter konfiguriert den Port zu 115200Baud, 8,N,1.

Im Konfigurationstool sind serielle Parameter einstellbar, wie erfolgt der Zugriff im Skript?

Über die Funktion `PortReadConfigDb()` und Zuweisung einer Variablen.

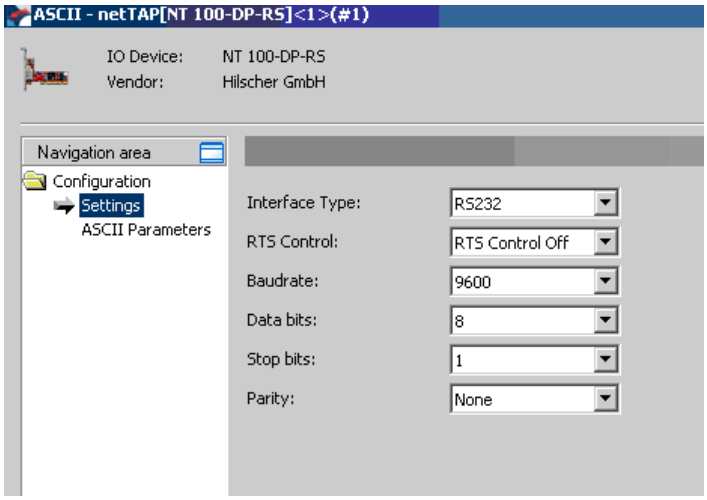


Abbildung 4: Serielle Schnittstellenparameter

Beispielcode:

```
conf = PortReadConfigDb()
```

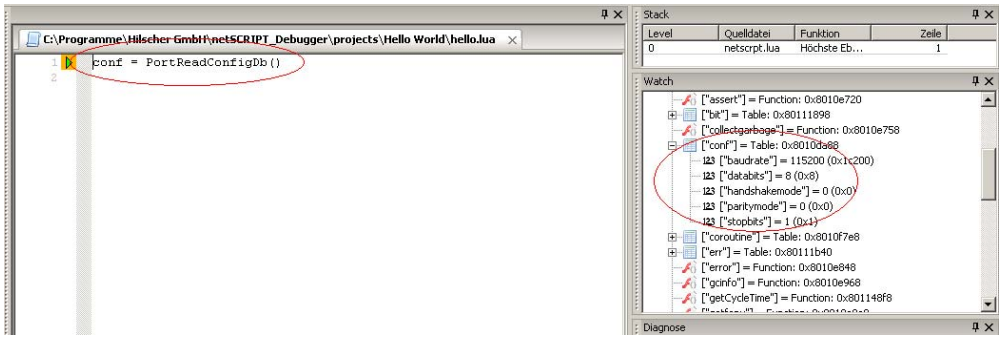


Abbildung 5: Beispiel PortReadConfigDb()

Sind die vom Konfigurationswerkzeug SYCON.net eingestellten seriellen Parameter im Skript abänderbar?

Ja. Die Werte einfach einer Variablen zuweisen und abändern.

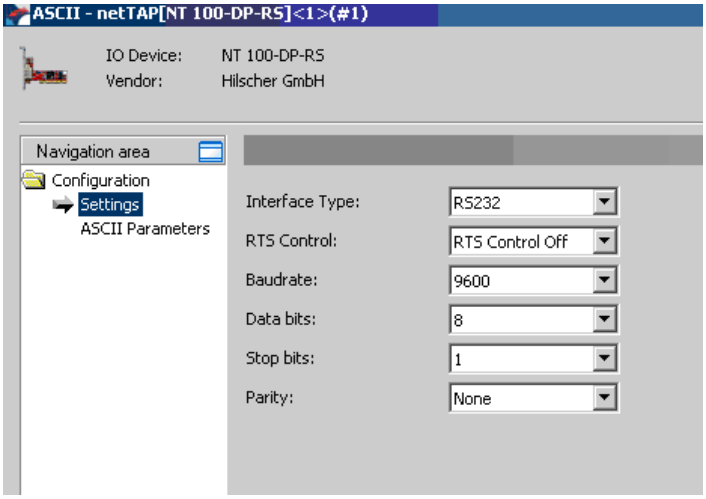


Abbildung 6: Serielle Schnittstellenparameter

Beispielcode:

```
conf = PortReadConfigDb()
conf.baudrate = 9600 -- set baudrate to 9600baud in any case
myport = PortOpen(conf)
```

Nach dem Senden eines Telegramms soll der UART sofort empfangsbereit sein, welche Funktion muss verwendet werden?

Langt die Zeit zwischen den beiden Aufrufen `:PortSend()` und `:PortReceive()` innerhalb des Skriptes nicht, um sicherzustellen das insbesondere bei höheren Baudraten der UART rechtzeitig empfangsbereit ist, muss die Kombinationsfunktion `:PortExchange()` verwendet werden.

2.8 Fragen zur Kommunikation zum übergeordnetem EA-Netzwerkes

Wo sollte die Initialisierung des EA-Ports des übergeordneten Netzwerkes durch Aufruf von `BusIOOpen()` geschehen?

In der Initialisierungsphase – in einem Programmteil der nur einmal durchlaufen wird. Realisiert wird die Steuerung des Durchlaufes über eine Zustandsvariable.

Kann man einen weiteren EA Port öffnen?

Nein, es ist nur die Öffnung eines Ports vorgesehen.

Kann man den EA-Port schließen und wieder öffnen?

Ja, beliebig oft.

Sollte die abgeschlossene Initialisierung an die EA-Steuerung gemeldet werden?

Die Benutzung der Funktion `:BusIOSetRun()` ist optional. Die Meldung hat folgenden Vorteil: Nach dem Neustart sind alle Eingangsdaten von netSCRIPT zur Steuerung Null. Ein Setzen des Bereitmeldebits meldet daher der Steuerung , ob netSCRIPT bzw. das Gateway generell initialisiert ist.

Muss vor dem Absetzen eines Kommandos von der übergeordneten Steuerung zu netSCRIPT erst eine Freigabe erteilt werden?

Nein nicht unmittelbar. Ein Schreibkommando kann von der Steuerung auch ohne Freigabe angefordert werden, jedoch reagieren in netSCRIPT die Funktionen `:BusIORead()` oder `:BusIOWrite()` auf kein Kommando oder führen keine Quittung aus. Erst das Setzen der Freigabebits `APP_HS_TX_ENABLE_CMD` und `APP_HS_RX_ENABLE_CMD` im E/A Synchronisationsregister schaltet die Funktionen ein bzw. beim Löschen wieder aus.

Werden die Freigabebit `APP_HS_TX/RX_ENABLE_CMD` in dem Synchronisationsregister zur Steuerung automatisch quittiert?

Ja.. Das Setzen der Bits `APP_HS_TX_ENABLE_CMD` oder `APP_HS_RX_ENABLE_CMD` wird von netSCRIPT durch das Setzen des Bits `PROT_HS_TX_ENABLE_ACK` bzw. `PROT_HS_RX_ENABLE_ACK` automatisch quittiert. Vorraussetzung ist, dass im Skript der EA-Port mit der Funktion `BusIOOpen()` geöffnet wurde.

Ist die Verwendung der Fehlermeldefunktionen zum übergeordneten Netzwerk optional?

Ja. Die Funktion `:BusIOSetError()` kann beliebig verwendet werden.

Können die Fehlercodes der Fehlermeldefunktion beliebig verwendet werden?

Ja, es gibt keine Vorgaben. Der volle Wertebereich von 0- 2^31-1 steht zur Verfügung.

Die Steuerung kann eine Resetanforderung schicken, muss dabei wirklich ein Reset durchgeführt werden?

Nein. Die Funktion soll zwar im ursprünglichen Sinne diese Anforderung erfüllen, sie kann aber dennoch für jede beliebige zusätzliche Übergabemeldung verwendet werden.

Wann verwendet man die Lesefunktion `:BusIORead()` ohne Quittungsmeldung?

Daten von der Steuerung ohne Quittung zu lesen wird angewendet, wenn die weitere Datenverarbeitung in netSCRIPT zeitlich ausgedehnt ist. Man vermeiden so neue Daten von der Steuerung zu erhalten, noch bevor die alten bearbeitet wurden. Die Quittung erfolgt am Ende durch nochmaliges Aufrufen von `:BusIORead()` mit Quittungsanforderung,

Wie übergibt man Fehler die an ein Lesekommando gekoppelt sein sollen?

Für die Fehlerübergabe an die Steuerung gibt es keinen Auftrags- oder Quittungsmechanismus, der Fehler kann zu jeder Zeit gesetzt oder gelöscht werden. Ist eine Kopplung an die Lesefunktion dennoch gewünscht, muss der Eintrag eines Fehlers vor der Quittierung des Leseauftrages erfolgen. Die Steuerung kann so bei Erhalt der Quittung nach dem Fehler prüfen.

Beispielcode:

```
mystring = myport:BusIORead(false) -- read data, no confirm
… -- work with the data
myport:BusIOSetError( “receive”,true, myerrorcode)
myport:BusIORead(true) -- confirm data after setting error
```

2.9 Fragen zur EA-Steuerung

Über eine SIMATIC-S7 sollen die Sende und Empfangsfreigabe gesetzt werden, wie geht das?

Es müssen die Bits 7 und 6 des Synchronisationsregisters gesetzt werden. Das Beispiel nimmt an das das Ausgangs-Synchronisationsregister auf MB 0 liegt.

Beispielcode:

```
L     B#16#C0       // load hex 0xC0, bit 7 and 6 being set
T     MB 0          // set bitmask to address 0
```

Die übertragene EA-Datenlänge überschreitet die Länge von 4 Bytes bei einer SIMATIC-S7, was ist zu tun?

Es müssen die SFC14 und SFC15 zur Datenübergabe verwendet werden, um eine konsistente Datenübertragung zu wahren.

Beispielcode SFC15:

```
CALL  "DPWR_DAT"   // call output write function SFC15
 LADDR  :=W#16#0   // start at output address 0
 RECORD :=P#M 0.0 BYTE 20 // copy from merker 0
 RET_VAL:=MW200           // where to return function's error
```

Beispielcode SFC14:

```
CALL  "DPRD_DAT"  // call input read function SFC14
 LADDR  :=W#16#0  // start at input address 0
 RET_VAL:=MW200   // where to return function's error
 RECORD :=P#M 20.0 BYTE 20 // copy to meker 20
```

Wie realisiert man die Erkennung einer Schreibanforderung seitens netSCRIPT in einem SIMATIC-S7 Programm?

Das Beispiel nimmt an das das Aus- und Eingangs-Synchronisationsregister auf MB 0 und MB 20 liegen.

Beispielcode:

```
 L MB 0    // load output handshake byte
 L MB 20   // load input handshake byte
 XOW       // do XOR knotting
 L 2       // load bit mask for APP_HS_RX_CMD = new command?
 UW        // make AND knotting and test for bit
 SPZ label // jump if result is zero, no command
```

Wie quittiert man die Schreibanforderung von netSCRIPT in einem SIMATIC-S7 Programm?

Das Beispiel nimmt an das das Ausgangs-Synchronisationsregister auf MB 0 liegt.

Beispielcode:

```
 L MB 0   // load current handshake byte
 L 2      // load bit mask APP_HS_RX_ACK, acknowledge
 XOW      // flip bit
 T MB 0   // transfer handshake byte for next cycle
```

2.10 Fragen zu den von „außen“ definierbaren Variablen

Im Konfigurationstool SYCON.net können Variablen definiert werden, wie erfolgt der Zugriff im Skript?

Die Variablen werden in der Tabelle VAR global angelegt und ihnen die konfigurierten Werte automatisch zugewiesen. Über den Zugriffspfad `VAR.Tabellenname.Variablenname` sind die Variablen im Skript direkt verwendbar.

Der Screenshot zeigt die Variablen `mynumber`, `mybool` und `mystring` mit den zugehörigen Werten die in der Tabelle `mytable` definiert sind.

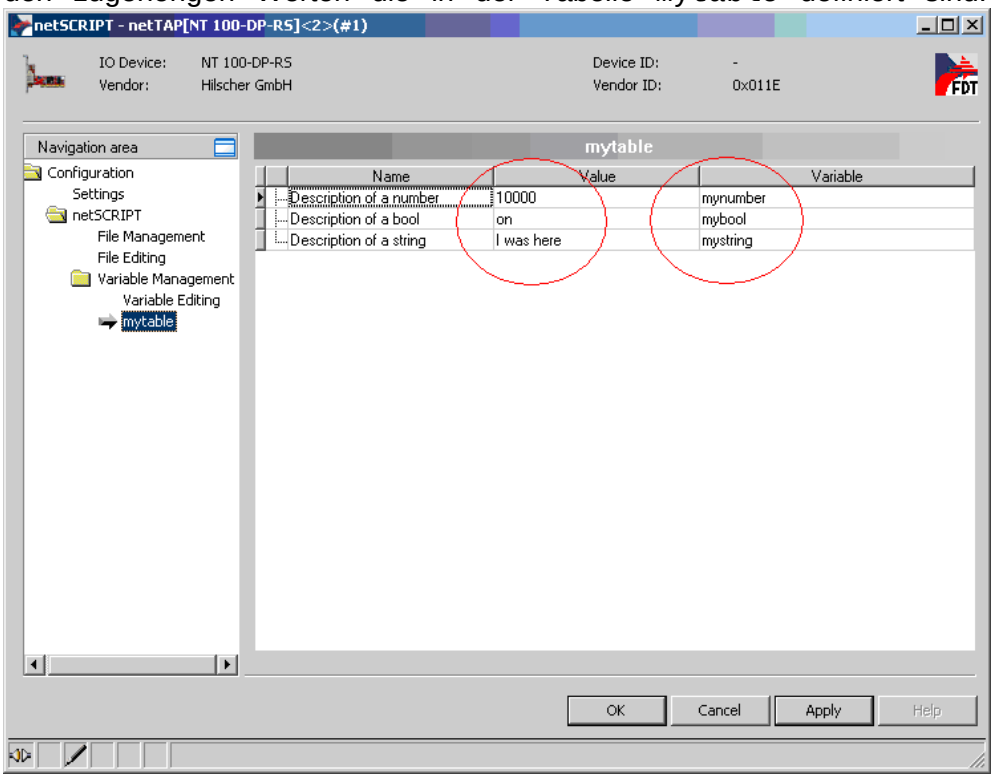


Abbildung 7: Von „außen“ definierbare Variable

Beispielcode:

```
strlocal = VAR.mytable.mystring      -- equals to “I was here“
```

Bleiben die einmal über SYCON.net definierten Variablen und deren Inhalte erhalten, wenn anschließend der netSCRIPT-Debugger verwendet wird?

Ja. Die Variablen werden in einer speziellen Konfigurationsdatei auf dem Gateway gespeichert. Auf diese Datei kann der netSCRIPT-Debugger und das Skript-File selbst nur lesend zugreifen.

3 Anhang

3.1 Abbildungsverzeichnis

3.2 Tabellenverzeichnis

3.3 Kontakte

Hauptsitz

Deutschland
Hilscher Gesellschaft für
Systemautomation mbH
Rheinstrasse 15
65795 Hattersheim
Telefon: +49 (0) 6190 9907-0
Fax: +49 (0) 6190 9907-50
E-Mail: info@hilscher.com

Support
Telefon: +49 (0) 6190 9907-99
E-Mail: de.support@hilscher.com

Niederlassungen

China
Hilscher Ges.f.Systemaut. mbH
Shanghai Representative Office
200010 Shanghai
Telefon: +86 (0) 21-6355-5161
E-Mail: info@hilscher.cn

Support
Telefon: +86 (0) 21-6355-5161
E-Mail: cn.support@hilscher.com

Frankreich
Hilscher France S.a.r.l.
69500 Bron
Telefon: +33 (0) 4 72 37 98 40
E-Mail: info@hilscher.fr

Support
Telefon: +33 (0) 4 72 37 98 40
E-Mail: fr.support@hilscher.com

Indien
Hilscher India Pvt. Ltd.
New Delhi - 110 025
Telefon: +91 11 40515640
E-Mail: info@hilscher.in

Italien
Hilscher Italia srl
20090 Vimodrone (MI)
Telefon: +39 02 25007068
E-Mail: info@hilscher.it

Support
Telefon: +39/02 25007068
E-Mail: it.support@hilscher.com

Japan
Hilscher Japan KK
Tokyo, 160-0022
Telefon: +81 (0) 3-5362-0521
E-Mail: info@hilscher.jp

Support
Telefon: +81 (0) 3-5362-0521
E-Mail: jp.support@hilscher.com

Korea
Hilscher Korea Inc.
Suwon-Si, 443-810
Telefon: +82-31-204-6190
E-Mail: info@hilscher.kr

Schweiz
Hilscher Swiss GmbH
4500 Solothurn
Telefon: +41 (0) 32 623 6633
E-Mail: info@hilscher.ch

Support
Telefon: +49 (0) 6190 9907-99
E-Mail: ch.support@hilscher.com

USA
Hilscher North America, Inc.
Lisle, IL 60532
Telefon: +1 630-505-5301
E-Mail: info@hilscher.us

Support
Telefon: +1 630-505-5301
E-Mail: us.support@hilscher.com