



Slave Conformizer 1.0

Development & Test Environment

for SERCOS Slave Devices

User's Guide

Interest Group for SERCOS interface (IGS) e.V.
Im Muehlenfeld 28
53123 Bonn

Contents

<u>1</u>	<u>Introduction</u>	3
1.1	<u>SERCOS Slave Conformizer</u>	3
1.2	<u>Slave Conformizer Environment</u>	3
<u>2</u>	<u>Installing the Slave Conformizer</u>	5
2.1	<u>Recommended System Requirements</u>	5
2.2	<u>Installing the Hardware</u>	5
2.3	<u>Installing the Software</u>	5
<u>3</u>	<u>Using the Slave Conformizer</u>	6
3.1	<u>Entering the Slave's Configuration</u>	6
3.2	<u>Using the Parameter Browser</u>	6
3.3	<u>Using the Command Shell</u>	7
3.4	<u>The Protocoll Window</u>	7
3.5	<u>Using the Script Editor</u>	8
3.5.1	<u>Creating a New Script File</u>	8
3.5.2	<u>Open a Existing Script File</u>	8
3.5.3	<u>Executing a Script File</u>	8
3.5.4	<u>Debugging a Script File</u>	9
3.6	<u>Using the Batch Test Mode</u>	9
<u>4</u>	<u>SERCOS Scripting Language (SSL)</u>	10
4.1	<u>Expressions</u>	10
4.2	<u>Variables</u>	10
4.3	<u>Numbers</u>	10
4.4	<u>Operators</u>	10
4.5	<u>Functions</u>	10
4.6	<u>Flow Control</u>	11
4.7	<u>Special Functions</u>	11
<u>5</u>	<u>Learning More</u>	12

1 Introduction

1.1 SERCOS Slave Conformizer

The Slave Conformizer provides a powerful but easy-to-use development environment for testing SERCOS Drives as well as I/O stations according to the international standard IEC/EN 61491. The Slave Conformizer has been specifically designed to enable manufacturer's to test the SERCOS interface communication of their Slave implementations more extensively and in a fraction of the time than would be necessary with tests written using C or C++.

1.2 Slave Conformizer Environment

The Slave Conformizer environment consists of following components:

SERCOS Master Driver:

The real-time Master Driver fully supports the new SERCON816 ASIC with transmission rates up to 16 Mbits/s and cycle times down to 500 μ s. The driver can be easily configured either via a dialog or using script functions. Toolbars are provided for starting and stopping the driver, changing the communication phase, selecting the active device address and setting the telegram type. Application Telegrams can be easily configured using a powerful dialog.

SERCOS Scripting Language (SSL)

This is a high-level language with control flow statements, over 80 functions, variables and input/output programming features.

Parameter Browser

This Browser shows which SERCOS Slave Devices are currently in the ring and displays all parameters and their elements as a tree structure. Elements can be either read or written and the name of a parameter is automatically displayed on moving the mouse's cursor over a parameter's number.

Command Shell

The Command Shell allows Master Driver functions to be accessed directly. An intelligent recall buffer allows previously executed commands to be quickly recalled without having to retype them. Recognized functions are automatically highlighted.

Script Editor

The Script Editor allows complex tests (SME-files) to be created, managed, executed and debugged. Recognized functions and programming constructs are automatically highlighted. The first line of the script can be used to define a name for the script file which is displayed both in the Batch Test Overview and by the Script Browser.

Function Browser

This Browser displays all script functions as a tree structure. Script functions can be found either using the index or the contents tree. Each function lists both its input and return parameter and can be inserted either in the Command Shell or Script Editor by simply double-clicking the function name.

Script Browser

This Browser shows all predefined and user-defined directories and script files as a tree structure. It can be used to open script files or to create a batch test by selecting which script files should be added to the Batch Test Overview. The script test name is automatically displayed on moving over a script file with the mouse cursor.

Batch Test Overview

The Batch Test Overview together with the Script Browser allows test scenarios to be created, configured and executed.

Protocol

The protocol window shows which functions have been called and their return values. This formatted and color-highlighted protocol can be saved in Rich Text Format and viewed using Microsoft's Word or printed out directly from the Slave Conformizer. Four different information levels are available from low (Level 1) to high detail (Level 4). Level 4 includes echoing of a script's command lines and comments.

Error Code Browser

This Browser lists all error codes in hexadecimal returned either by the Slave Device or from the Master Driver with the corresponding error messages.

2 Installing the Slave Conformizer

2.1 Recommended System Requirements

- ✱ Intel Pentium II or Pentium III
- ✱ 128 MB RAM
- ✱ Microsoft's Windows NT (Service Pack 6)
- ✱ Microsoft's Internet Explorer 5.x

2.2 Installing the Hardware

- ✱ Switch off your PC and open the housing in accordance with the manufacturer's instructions.
- ✱ Insert the Beckhoff PCI card into a free PCI slot.
- ✱ Boot your PC

2.3 Installing the Software

- ✱ Before installing the Slave Conformizer software please make sure that RTX 5.0 RT WinNT and Microsoft's Internet Explorer 5.x are properly installed.
- ✱ Insert the Slave Conformizer CD into your CD-ROM drive.
- ✱ If the setup program does not start automatically, please click on the setup.exe located on the Slave Conformizer CD.

3 Using the Slave Conformizer

3.1 Entering the Slave's Configuration

After starting the Slave Conformizer, a configuration dialog for the Slave Device to be tested appears. The baud rate and cycle time supported by the Slave Device must be correctly configured before starting the Master Driver. The Master Driver uses the maximum baud rate supported by the Slave Device and minimum cycle time supported down to 500 μ s. For cycle times less than 1 ms, a cycle time of 1 ms is used for CP0-2 according to the IEC/EC 61491. Otherwise the cycle time for CP0-2 is the same as that for CP3-4.

The device address of the Slave to be tested should also be correctly set, but can also be changed after the Master Driver has been started by using the Slave Device Toolbar. It should be noted that unlike normal SERCOS Masters, the SERCOS Conformizer Master Driver always communicates with only one Slave Device at a time. If several Slave Devices are in the SERCOS ring, it is still possible to communicate with each Slave Device by changing the active Slave Device using the Dialog Toolbar provided. It is however not possible to change the communication phase of all Slave Devices simultaneously. This was the only way to prevent other Slave Devices in the SERCOS ring from interfering with the Slave Device being tested. However when performing a Conformance Test it is still recommended that a Slave Device is tested with no other Devices in the ring, since incorrectly implemented Slave Device's in the same ring can still cause incorrect test results.

After entering the Slave's configuration you have the option to start the Master Driver. The Master Driver can also be started or stopped using the corresponding buttons on the Main Toolbar. The communication phase of the active Slave Device can be changed using the Communication Phase Toolbar. After starting the Slave Conformizer, the Parameter Browser, Command Shell and Protocol Window are displayed.

3.2 Using the Parameter Browser

On expanding the Parameter Browser, the Master Driver searches the SERCOS ring for Slave Devices and displays them. After expanding a found Slave Device, a list of S- und P-Parameters can be viewed. Further expanding of a particular Parameter displays its individual elements which if allowed can also be changed. If several Slave Devices are found it is also possible to read and write parameters for each Slave Device. The parameter browser automatically changes the active Slave Device and communication phase for you. To change the value of a parameter's element, either click to edit or use the context menu. On pressing return the value is changed if the write was successful. To check that the value was written correctly simply re-expand the element that was changed.

3.3 Using the Command Shell

In the Command Shell script functions listed in the Function Browser can be executed. As already mentioned the Slave Conformizer only communicates with the active Slave Device selected by the Slave Device Toolbar. For example if Slave Device 1 is active before entering the following, pressing return causes the Slave Conformizer to try and change this device's Communication Phase to CP 2:

```
change_phase(2);
```

The communication phase of the Slave Device should change to 2. This should also be displayed by the Communication Phase Toolbar, but can also be queried using:

```
read_phase();
```

Any other devices in the SERCOS ring should show CP0. Using the Communication Phase Toolbar has the same affect as calling `change_phase()`.

Furthermore entering the following and pressing return, commands the Slave Conformizer to the read data value of IDN S-0-0002 from the Slave Device which is currently active:

```
read_data(2);
```

In the Command Shell and Script Editor, as soon as a entered function name is recognized it is automatically color-highlighted. Try out further functions by simply double-click functions listed in the Function Browser and editing if necessary the input arguments. You can use the up and down cursor keys to recall past commands. For an intelligent search of past commands, enter one or some of the first letters of the command desired and press the up cursor key until the command desired is shown.

It is also possible to use variables and **if** expressions in the command shell. For more information please see Section 4 of this document.

3.4 The Protocol Window

When a script function is called, either from the Command Shell or from a script file, a message is displayed in the Protocol Window showing:

- ✱ the time when the function called (timestamp),
- ✱ the script file line number (for script files only),
- ✱ the communication phase as the function was called,
- ✱ the active Slave Device as the function was called,
- ✱ the parameter or info read or written,
- ✱ the element type read or written,
- ✱ the value of element read or written or an error code and message, if the value could not be changed or read.

The Protocol Window displays formatted and color-highlighted messages for each script function. This Protocol can either be printed directly from the SERCOS Conformizer or saved in Rich Text Format (RTF). The RTF-Format preserves the formatting and color-highlighting and allows the Protocol to be viewed and printed with an external viewer such as Microsoft's Word or with WordPad. For the purposes for File Comparing protocols it is possible to deactivate the timestamp (see Edit->Preferences->Timestamp).

Using the protocol's context menu the information level for the protocol can be selected.

Info Level 1: lowest level, no diagnosis info included (S-0-0095)

Info Level 2: default level, includes diagnosis info (S-0-0095) when changing phase

Info Level 3: debugging level, includes Level 2 info and echoes Script file commands

Info Level 4: highest level, includes Level 3 info and also echoes Script file comments

3.5 Using the Script Editor

Files that contain code in the SSL language for the Slave Conformizer are called SME (SERCOS Master Emulator) script files. You can create same script files using the Script Editor or an external text editor of your choice. When you run or step a SME script file from the Script Editor, the Slave Conformizer automatically executes the commands found in this script file.

3.5.1 Creating a New Script File



To create a new test select File->New Script or use the Main Toolbar button. If you enter a comment in the first line, this comment is displayed in the script browser (when you move the cursor over a script file) and in the batch test overview as the test name. In the script file you can use any of the script functions listed in the Function Browser and the programming constructions described in Section 4.6. As in the Command Shell, Script Commands can be copied to the Script Editor Window by double-clicking a function listed in the Function Browser. Functions are also automatically color-highlighted as soon as they are recognized. Keywords such as **if** and **end** are also color-highlighted. To deactivate or reactivate the syntax-highlighting select Edit->Preferences->Syntax-Highlighting.

3.5.2 Open a Existing Script File


To open an existing script file you can either use:

- ✳ File->Open Script
- ✳ double-click a file using the Script Browser,
- ✳ or use the context menu of the Script Browser.



3.5.3 Executing a Script File

To execute a script select the  button from the Control Toolbar. The Protocol Window shows you which script was started and when it was been finished. To abort the execution of the script function use the  button.

3.5.4 Debugging a Script File

To debug a script file, first set the info level of the Protocol Window to Level 3 or 4 use the context menu option. Then step into the script file using the  Button of the control toolbar. By pressing this button you can step through the script file.

3.6 Using the Batch Test Mode

By selecting the Batch Test view, it is possible to create a batch test by checking the directories or files displayed in the script browser. By moving the cursor over a script file, the test's name is displayed automatically. The Batch Test Overview window shows which files have been selected and allows you to change the order in which they are executed using the context menu. To execute the batch test select the  button from the control toolbar. The Batch Test Overview shows which test is currently being executed and the result and number of fails (if applicable) for completed tests. A batch test can be aborted using the  button.

4 SERCOS Scripting Language (SSL)

4.1 Expressions

The building blocks of expressions for SSL are

- Variables
- Numbers
- Operators
- Functions

4.2 Variables

SSL does not require any type declarations. When SSL-interpreter encounters a new variable name, it automatically creates the variable. If the variable already exists, the SSL-interpreter changes its contents.

Variable names consist of a letter, followed by any number of letters, digits, or underscores. SSL is case sensitive; it distinguishes between uppercase and lowercase letters. A and a are *not* the same variable.

4.3 Numbers

SSL uses conventional decimal notation, with an optional decimal point and leading plus or minus sign, for numbers. *Scientific notation* uses the letter *e* to specify a power-of-ten scale factor.

All numbers are stored internally using the *double* format specified by the IEEE floating-point standard. Floating-point numbers have a finite *precision* of roughly 16 significant decimal digits and a finite *range* of roughly $-2.23e-308$ to $1.79e+308$.

4.4 Operators

Expressions use familiar arithmetic operators and precedence rules. For example:

- + Addition
- Subtraction
- * Multiplication
- / Division
- ^ Power
- () Specify evaluation order

4.5 Functions

SSL provides a over 80 predefined functions. For a complete list of available functions use the Function Browser. A detailed description of each function can be found in the Slave Conformizer's Reference Guide.

4.6 Flow Control

SSL currently only supports simple **if** statement flow control constructs. The **if** statement evaluates a logical expression and executes a single statement when the expression is *true*. Keywords such as **elseif** or **else** which are normally provided for the execution of alternate statements are not yet supported. An **end** keyword, which matches the **if**, terminates the last statement. The statements are delineated by these keywords – no braces or brackets are involved.

4.7 Special Functions

`get_last_error()`

If you look at the functions listed in the function browser you will see that some functions return error codes while other functions return a double value such as a element's data value. If the element's data value for example could not be read then the function returns `DBL_MAX` (1.79e+308). In order to find out the error code of such functions you can use the function `get_last_error()`

`fail_test()`

When this function is called the fail counter for this script file is incremented by one. The total number of fails is displayed during a batch test. By combining the `get_last_error()` function with an **if** construct, the `fail_test()` function can be used to determine whether or not the previously called function returned 0 for no errors or a known error code. Since it is generally not possible for the Slave Conformizer to know whether a function call should succeed or not, function calls returning an error are displayed in green. Only when the `fail_test()` is called or a serious communication error occurs is a message displayed in red.

`abort_test()`

The `abort_test()` function should be used instead of `fail_test()` if a error in the test is so severe that it prevents the test from being continued. After `abort_test()` has been called the test is then immediately aborted. In the Protocol Window and Batch Test Overview the test is displayed as having been aborted. This test is then considered to have failed.

`skip_test()`

In some causes, a test cannot be executed because the functionality being tested does not have to be supported. In such situations the `skip_test()` function can be used to stop a test prematurely. In the Protocol Window and Batch Test Overview such tests are displayed as having been skipped. These tests are not considered to have failed.

5 Learning More

For more SLL examples, look at the Conformance Test script files provided with the Slave Conformizer. For the very latest information about the Slave Conformizer including product updates, the newest Conformance Test script files, product documentation etc. point your Web browser to:

<http://www.isw.uni-stuttgart.de/sercos>