



Master Conformizer V2.01

Development & Test Environment

for SERCOS Master Devices

User's Guide

Contents

1	Introduction.....	3
1.1	SERCOS Master Conformizer	3
1.2	Master Conformizer Environment	3
2	Installing the Slave Conformizer.....	5
2.1	Recommended System Requirements	5
2.2	Installing the Hardware	5
2.3	Installing the Software.....	5
3	Using the Master Conformizer	6
3.1	Connecting the Fiber Optic.....	6
3.2	Entering the Slave’s Configuration	6
3.2.1	Structure of the Init File.....	6
3.2.2	Structure of the Configuration File for the single device	7
3.3	Using the Parameter Browser	8
3.4	Using the Command Shell.....	8
3.5	The Protocol Window.....	9
3.6	Using the Script Editor	10
3.6.1	Creating a New Script File	10
3.6.2	Open a Existing Script File.....	10
3.6.3	Executing a Script File	10
3.6.4	Debugging a Script File	10
3.7	Using the Batch Test Mode.....	10
3.8	Using the Application Telegram Configuration.....	11
4	SERCOS Scripting Language (SSL)	12
4.1	Expressions	12
4.2	Functions	12
4.3	Special Functions.....	12
5	Learning More.....	14

1 Introduction

1.1 SERCOS Master Conformizer

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

1.2 Master Conformizer Environment

The Master Conformizer environment consists of following components:

✿ SERCOS Slave Driver:

The real-time Slave Driver fully supports the new SERCON816 ASIC with transmission rates up to 16 Mbits/s and cycle times down to 500 μ s. It is possible to simulate up to 8 configurable drives. The driver can be easily configured either via a XML-file or using script functions. Toolbars are provided for starting and stopping the driver and selecting the active device address.

✿ 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 configured and displays all parameters and their elements as a tree structure. Elements can be either read or written (only the data element) 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 Slave 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 (mcs-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 it’s 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 used-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 tests 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. It also protocols the current phase and the operations done by the Master using the service channel. 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 Master Conformizer. Four different protocol levels for the service channel protocol and two for the phase protocol are available.

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

3 Using the Master Conformizer

3.1 Connecting the Fiber Optic

For the Master Conformizer the “Channel B” receiver and transmitter must be used.

3.2 Entering the Slave’s Configuration

After starting the Slave Conformizer, a dialog to the path with the init file appears. In this init file all information of the simulation is included, also the path to the XML-files with the ident numbers for the single drives.

The Slave Driver can also be started or stopped using the corresponding buttons on the Main Toolbar. After starting the Master Conformizer, the Parameter Browser, Command Shell and Protocol Window are displayed.

3.2.1 Structure of the Init File

The Init file is used to describe the general structure of the Master Conformizer like the baudrate and transpower. The single elements of the Init file are:

✱ `<Slave>...</Slave>`

General information of the slave.

✱ `<Baudrate>...</Baudrate>`

Selected baudrate in Mbit/s. Allowed values are 2,4,8 and 16.

✱ `<Transpower>...</Transpower>`

Transmission power of the optical output. Allowed values are from 1 up to 4.

✱ `<BufferSizeMB>...</BufferSizeMB>`

Size of the communication buffer between GUI and real time driver.

✱ `<Device>...</Device>`

Information of the single device.

✱ `<Device_Index>...</Device_Index>`

Index of the single device from 0 up to 8. This value is not to be changed.

✱ `<Device_Address>...</Device_Address>`

Address of the single device. If the device should not be configured the address 0 should be entered here.

✱ `<Path>...</Path>`

Path of the configuration file with the ident numbers for this device. The path must be set absolutely. On some PCs \ must used a path separator, on some PCs \\.

3.2.2 Structure of the Configuration File for the single device

The Init file is used to describe the ident data base of the single drive. The single elements of the Configuration file are:

✱ <IDN>...</IDN>

Start and stop tag for one ident number.

✱ <Ident>...</Ident>

Ident number as a string (e.g. S-0-0002, P-2-0009).

✱ <Name>...</Name>

Name of the ident number as a string. The length of the name is calculated based on the string. The maximum allowed length is 60 bytes.

✱ <Attribute> ...</Attribute>

Start and stop tag for the attribute.

✱ <WriteProtectedCP4>...</WriteProtectedCP4>

Specifies if the ident is write protected in CP4. Allowed values are “true” and “false”.

✱ <WriteProtectedCP3>...</WriteProtectedCP3>

Specifies if the ident is write protected in CP3. Allowed values are “true” and “false”.

✱ <WriteProtectedCP2>...</WriteProtectedCP2>

Specifies if the ident is write protected in CP2. Allowed values are “true” and “false”.

✱ <DecimalPoint>...</DecimalPoint>

Specifies the places after the decimal point. Allowed values are from 0 up to 15.

✱ <DataType>...</DataType>

Specifies the data type and the display format. Allowed values are “Binary”, “UnsignedDecimal”, “SignedDecimal”, “Hexadezimal”, “Text” and “IDN”.

✱ <Command>...</Command>

Specifies if the ident is a command. Allowed values are “true” and “false”.

✱ <Variable>>false</Variable>

Specifies if the ident is of variable length (list). Allowed values are “true” and “false”.

✱ <Unit>...</Unit>

Unit of the ident number as a string. The length of the unit is calculated based on the string. The maximum allowed length is 12 bytes.

✱ <Min>...</Min>

Minimum data of the ident number.

✱ <Max>...</Max>

Maximum data of the ident number.

✱ <Data>...</Data>

Value of the ident number.

✱ <ListData>...</ListData>

If the ident number is a list this tags must be used instead of the <Data>...</Data> Tags.

✱ <actualLength>...</actualLength>

Actual length of the list in configured elements, not in bytes.

✱ `<maxLength>..</maxLength>`

Maximum allowed list length in configured elements, not in bytes.

✱ `<List>...</List>`

Every element of the list must be specified between this start and stop tags.

Exception: an ASCII list (text) can be completely written between one start and one stop tag (e.g. `<List>Master Conformizer</List>`).

The elements for the attribute and the data or list data are absolutely necessary for the specification of an ident number, the other elements are optional.

3.3 Using the Parameter Browser

On expanding the Parameter Browser, the Master Conformizer displays the configured Slave Devices and displays them. After expanding a configured Slave Device, a list of S- and 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 configured it is also possible to read and write parameters for each Slave Device. The parameter browser automatically changes the active Slave Device. To change the value of a parameter’s value, 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.

3.4 Using the Command Shell

In the Command Shell script functions listed in the Function Browser can be executed. As already mentioned the Master Conformizer does all the functions calls for the configured active Slave device.

Furthermore entering the following and pressing return, commands the Master 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.

3.5 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 for function calls:

- ✱ the time,
- ✱ 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.

All function call messages are displayed in brown color.

For the service channel protocol the following information are displayed:

- ✱ the time,
- ✱ the communication phase,
- ✱ the Slave Device that have transmitted or received the data,
- ✱ the parameter or info read or written,
- ✱ the element type read or written,
- ✱ the data read or written or the sent error message

The Protocol Window displays formatted and color-highlighted messages for each script function or service channel operation. This Protocol can either be printed directly from the SERCOS Conformizer (File->Print Protocol) or saved in Rich Text Format (RTF) (File->Save Protocol). 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 or clicking with the right mouse button on the protocol window the level for the service channel protocol can be selected.

Info Level 0: lowest level, no protocol

Info Level 1: service channel protocol for read and write access to the data element and errors

Info Level 2: service channel protocol for read and write access to all elements and errors

Info Level 3: service channel protocol for all operations

Using the protocol’s context menu or clicking with the right mouse button on the protocol window also the level for the phase protocol can be selected.

Info Level 0: lowest level, no protocol

Info Level 1: protocol of the phase

Using the right mouse button it is also possible to copy the content of the protocol window or to clear the protocol window.

3.6 Using the Script Editor

Files that contain code in the SSL language for the Master Conformizer are called MCS (Master Conformizer Scripts) 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 MCS script file from the Script Editor, the Slave Conformizer automatically executes the commands found in this script file.

3.6.1 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. 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.

3.6.2 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.



3.6.3 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.

3.6.4 Debugging a Script File

To debug a script file step into the script file using the  Button of the control toolbar. By pressing this button you can step through the script file.

3.7 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.

3.8 Using the Application Telegram Configuration

By selecting the Configuration->Application Telegram it is easily possible to view the application telegram . The lists S-0-0187, S-0-0188, S-0-0016 and S-0-0024 are read and their contents are displayed. Also the maximum allowed lengths (S-0-0185 and S-0-0186) are displayed.

4 SERCOS Scripting Language (SSL)

4.1 Expressions

The building blocks of expressions for SSL are

- Variables
- Numbers
- Operators
- Functions

4.2 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 Master Conformizer’s Reference Guide.

4.3 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 Master Conformizer. For the very latest information about the Master Conformizer including product updates, the newest Conformance Test script files, product documentation etc. point your Web browser to:

<http://www.sercos.de>

The documents can be found in the Download Area, Documents for Certification.