# CODESYS Control V3

Guidelines for runtime adaptation - platforms

Version: 4.1
Template: templ_tecdoc_en_V1.0.docx
Dateiname: CODESYSControlV3_Adaptation-Platforms.docx

# CONTENT

## 1 Overview

For OEM manufacturers, the CODESYS Control runtime system is not a ready-to-use product. It is a toolkit that has to be adapted by the OEM.

An OEM who wants to use the CODESYS Control runtime system has to provide some prerequisites before the adaption begins, and has to finish the adaptation after the workshop provided by 3S–Smart Software Solutions (in the following called 3S).

In the workshop, the OEM learns about how to adapt and extend the runtime, but the result of the workshop is not a finished PLC product.

3S assists the OEM by telephone and email support in case questions arise after the workshop.

On request, 3S can help with the OEM's work packages of adaptation and test; 3S will estimate the effort and send quotations for the customer specific service.

## 2 Runtime adaptation work packages

Depending on the different types of adaptation, different preparation and adaptation work has to be done.

Basically, there are 3 different starting points for a runtime adaptation:

- Using standard processors and operating systems like Linux, VxWorks, Windows
- Ready-to-start embedded system with use of supported evaluation board
- Embedded system with other processor/OS/toolchain combination

In the following chapters, runtime toolkit contents, prerequisites and step to build a system are listed.

As the OEM has to do maintenance for its product and support for its customers for a longer period of time, it is recommended that the OEM assigns an engineer as specialist for these tasks.

### 2.1 Standard System

Examples of such systems are Windows 7 on an Intel Pentium, or VxWorks on a PowerPC, or Linux on an ARM processor.

Supported operating systems are Linux, VxWorks, Windows 7, Windows CE/Windows Embedded Compact and QNX.

The CODESYS Control runtime system for Windows 7 is available in two different versions:

- CODESYS Control RTE: with hard real time extension made by 3S
- Some hardware requirements must be observed, see RTE manual.
- CODESYS Control WinV3: without real time capabilities:
- No special hardware requirements, so it can run on every Windows / x86 platform.

3S maintains and tests these systems with new releases of CODESYS and/or new releases of the operating system.

The runtime toolkit contains:

| | Contents of Runtime Toolkit |
|---|---|
| a) | Sample device description file for that platform |
| b) | CODESYS Control runtime system headers and m4 files |
| c) | Tools to create own runtime system header files |
| d) | 3s.dat license file for licensed fieldbus stacks and other libraries |
| e) | Sample runtime configuration file (*.cfg) |
| f) | Runtime adaptation (with multi tasking scheduler):<br>• Processor specifics<br>• Timer<br>• Memory<br>• Debug Outputs |

| | Contents of Runtime Toolkit |
|---|---|
| | • Flash/File system, directories<br>• Exception handling<br>• Interrupt handling<br>• Serial interface for communication<br>• Socket interface for communication and Modbus<br>• CANmini driver to support CANopen and other CAN-based protocols (see below)<br>• PCI and Shared memory to support Hilscher fieldbus cards<br>• Ethernet driver to support EtherCAT (see below)<br>• Event handling<br>• Semaphore handling<br>• Task handling |

Prerequisites:

| | Responsible | Task |
|---|---|---|
| i) | OEM | Test hardware (evaluation or final board) |
| ii) | OEM | C-compiler and C-debugger |
| iii) | OEM | Operating system installed, including drivers for file system, TCP stack, … |

Steps to build your system:

| | Responsible | Task |
|---|---|---|
| 1) | OEM | Inform 3S about processor, OS and version |
| 2) | 3S | Compile and deliver runtime toolkit to OEM |
| 3) | 3S | Workshop with OEM. OEM learns about<br>• how to work with CODESYS (basics)<br>• how to adapt device description file<br>• how to adapt runtime configuration file<br>• how to build additional components, libraries and IO drivers |
| 4) | OEM | Create additional components, libraries and IO drivers |
| 5) | OEM | Test |

## 2.2 Ready-to-start embedded system

3S supports a number of evaluation board/toolchain combinations with basic runtime adaptations.

Examples of such systems are Infineon TriCore with Infineon TriBoard TC1798, or ARM with TI TMS570 MCU Development Kit, or Renesas SH2A with Renesas Starterkit RSK-SH7216. For all these boards, specific toolchain versions are supported. A list of those combinations is available on request.

3S maintains and tests these systems with new releases of CODESYS.

The runtime toolkit contains:

| | Contents of Runtime Toolkit |
|---|---|
| a) | Sample device description file for that platform |
| b) | CODESYS Control runtime system headers and m4 files |
| c) | Tools to create own runtime system header files |
| d) | 3s.dat license file for licensed fieldbus stacks and other libraries |

| | Contents of Runtime Toolkit |
|---|---|
| e) | Sample runtime configuration file (*.cfg) |
| f) | CODESYS control runtime C source files, including adaptations for embedded runtime (with single tasking scheduler):<br>• Processor specifics<br>• Timer<br>• Memory<br>• Debug Outputs<br>• Flash (most of the platforms)<br>• CANmini driver to support CANopen and other CAN-based protocols (most of the platforms)<br>• Serial interface for communication |
| g) | Makefile, workspace or project for C compiler, to generate binaries for the evaluation board. |

Prerequisites:

| | Responsible | Task |
|---|---|---|
| i) | OEM | Test hardware (evaluation board) |
| ii) | OEM | C-compiler and C-debugger |

Steps to build your system:

| | Responsible | Task |
|---|---|---|
| 1) | OEM | Inform 3S about evaluation board |
| 2) | OEM | Purchase evaluation board |
| 3) | 3S | Deliver runtime toolkit to OEM |
| 4) | 3S | Workshop with OEM. OEM learns about<br>• how to work with CODESYS (basics)<br>• how to adapt device description file<br>• how to adapt runtime configuration file<br>• how to build and download runtime system<br>• how to build additional components, libraries and IO drivers |
| 5) | OEM | Create additional components, libraries and IO drivers, adapt system specific components not included in runtime toolkit. |
| 6) | OEM | Port to final HW |
| 7) | OEM | Test |

### 2.3 Embedded system with other processor/OS/toolchain combination

Examples of such systems are Renesas SH with MQX or iTron operating system, or other systems where 3S has no ready-to-use adaptation.

3S does no test and does no maintenance of these systems. OEMs can get updates of the runtime toolkit and use it to update their system to newer version of CODESYS, the C compiler or the board.

The runtime toolkit contains:

| | Contents of Runtime Toolkit |
|---|---|
| a) | Sample device description file for that platform |
| b) | CODESYS Control runtime system headers and m4 files |
| c) | Tools to create own runtime system header files |

| | |
|---|---|
| d) | 3s.dat license file for licensed fieldbus stacks and other libraries |
| e) | Sample runtime configuration file (*.cfg) |
| f) | CODESYS control runtime C source files |
| g) | Empty function frames for runtime adaptation. In some cases, adaptation samples are available. |

Prerequisites:

| | Responsible | Task |
|---|---|---|
| i) | OEM | Test hardware (evaluation board or final board) |
| ii) | OEM | C-compiler and C-debugger |
| iii) | OEM | Software to initialize the processor and run until C main() function |
| iv) | OEM | Drivers to access serial interface, flash, timer, external and internal memory |

Steps to build your system:

| | Responsible | Task |
|---|---|---|
| 1) | OEM | Inform 3S about processor, toolchain and memory configuration |
| 2) | OEM | Purchase evaluation board |
| 3) | 3S | Deliver runtime toolkit to OEM |
| 4) | 3S | Workshop with OEM. OEM learns about<br>• how to work with CODESYS (basics)<br>• how to adapt device description file<br>• how to adapt runtime configuration file<br>• how to build and download runtime system<br>• how to build additional components, libraries and IO drivers |
| 5) | OEM | Finish adaptation after the workshop, some of these features are optional:<br>• Processor specifics<br>• Timer<br>• Memory<br>• Debug Outputs<br>• Flash/File system, directories<br>• Exception handling<br>• Interrupt handling<br>• Serial interface for communication<br>• Socket interface for communication and Modbus<br>• Additional interfaces for communication (USB, …)<br>• CANmini driver to support CANopen and other CAN-based protocols<br>• Ethernet driver to support EtherCAT<br>• PCI and Shared memory to support Hilscher fieldbus cards<br>In case a multitasking operating system shall be supported, the following interfaces need to supported in addition:<br>• Event handling<br>• Semaphore handling<br>• Task handling |
| 6) | OEM | Create additional components, libraries and IO drivers |
| 7) | OEM | Port to final HW |
| 8) | OEM | Test |

# 3 Platform specifics

This chapter contains information about specifics of standard platforms.

## 3.1 Build

### 3.1.1 Linux

A toolchain must be provided to 3S, to be able to build the runtime.

### 3.1.2 VxWorks

VxWorks runtime is delivered as a binary in several configurations, e.g. floating point support, PCI support, and interrupt support. The toolchain supported is GNU.

## 3.2 CANopen

To support CANopen fieldbus stack and communication, a CANMini driver is needed. On some platform, this driver is already available. In case a standard driver is not available for a platform, it has to be developed.

### 3.2.1 Linux

A driver based on SocketCAN is included in the runtime toolkit.

The SocketCAN driver must be installed and tested by the OEM.

### 3.2.2 Windows CE

A driver for SJA1000 controllers is available, the driver works with several PCI cards (Peak, Automata, … ) or can be configured through the settings file for systems without PCI.

### 3.2.3 VxWorks

No standard driver available.

### 3.2.4 QNX

No standard driver available.

### 3.2.5 RTE

A driver for SJA1000 controllers is available, the driver works with several PCI cards (Peak, Automata, …) or can be configured through the settings file for systems without PCI.

For cards currently not supported, please contact 3S.

## 3.3 EtherCAT

To support the EtherCAT fieldbus stack, an Ethernet driver is needed. On some platforms, this driver is already available:

### 3.3.1 Linux, QNX

No specific requirements.

### 3.3.2 Windows CE

An NDIS based component for handling Ethernet frames is included with the runtime system.

It is possible to replace this solution by a customer-specific driver with direct access to the Ethernet controller, to achieve better realtime behavior.

For SoftMotion applications, the OEM has to extend the CE kernel by a microsecond timer (without this timer, EtherCAT distributed clocks and the entire SoftMotion cannot work).

### 3.3.3 VxWorks

No specific requirements.

### 3.3.4 RTE

Drivers for some cards (RTL8139, 8169, and compatible, and all Intel PRO100, PRO1000 based cards) are available.

## 3.4 Target Visualization

### 3.4.1 Linux, QNX

Qt Version 4 is required. These directories of the configured Qt installation have to be supplied to 3S:

- bin
- include
- lib

### 3.4.2 Windows CE

The CODESYS targetvisu uses the native GDI of Windows CE.

Some features are not supported by the standard GDI, so they are only supported if the specific CE system provides additional functionality. This concerns ActiveX Controls, pictures in PNG format and gradient coloring.

### 3.4.3 VxWorks

Frame buffer solution available. Adaptation by OEM required.

### 3.4.4 RTE

No specific requirements.

## 3.5 Web visualization

The runtime contains a web server so that a Web client on a remote system like a PC or any other device with a browser supporting HTML5 can display the web visualization.

### 3.5.1 Linux, QNX

No specific requirements.

### 3.5.2 Windows CE

A different use case concerns Windows CE systems which should display the web visualization themselves!! In this case, the CE system cannot use the standard browser because of missing HTML5 support. The OEM has to include a HMTL5 browser in his system, e.g. web kit.

### 3.5.3 VxWorks

No specific requirements.

### 3.5.4 RTE

No specific requirements.

# 4 Glossary

| | |
|---|---|
| CANMini driver | 3S specific software driver adaptation layer, between the standard runtime IO Driver interface and the hardware CAN chip. Specific implementation needed for every supported CAN controller. |

## Change History

| Version | Description | Author | Date |
|---|---|---|---|
| 0.1 | Created | TZ | 24.06.2013 |
| 0.2 | Added WinCE specifics | PB | 27.06.2013 |
| 0.3 | Added other OS specifics | TZ | 22.07.2013 |
| 0.3 | Review | RW | 25.07.2013 |
| 0.4 | Rework according to review | TZ | 25.07.2013 |
| 1.0 | Release after formal review | MN | 25.07.2013 |
| 1.1 | Added glossary | TZ | 26.07.2013 |
| 1.2 | Added some clarification about Windows runtimes | TZ | 19.12.2013 |
| 2.0 | Applying new template templ_tecdoc_en_V1.0.docx<br>Release after formal review | MaH | 03.02.2014 |
| 2.1 | Added legal note | TZ | 23.08.2016 |
| 3.0 | Release after formal review | MaH | 23.08.2016 |
| 3.1 | Legal note removed | GeH | 24.10.2016 |
| 4.0 | Release after formal review | MN | 28.11.2016 |
| 4.1 | Document renamed | AH | 13.11.2018 |