Tutorial:
First Steps with CODESYS Control

Document Version 3.0

CONTENT

1 Quickstart

1.1 Getting the Binaries

Please have a look to the file Contents.txt in your Runtime Delivery. There might be a hint about the form of the delivery. In general we provide the following delivery forms:

- Starter Package:
  You will receive the binaries separately via FTP.
- Binary Delivery:
  The binaries for your platform are located in the platform specific "Bin" folder.
  e.g.: Files/Platforms/Linux/Bin
- Source Delivery:
  You need to compile the runtime by your own. Please refer to the runtime migration and adaptation guide in the "Documentation" folder of your delivery. This will explain in detail how to configure and compile your runtime.

1.2 Getting the Configuration

There are a few sample configurations placed in the „Configuration" folder of your runtime delivery.

Please use the following for the different target platforms:

- Linux:
  A configuration will be included in the tar archive.
- VxWorks:
  Use the CODESYSControl_VxWorks.cfg from the „Configuration" folder.
- Windows CE:
  Use the CODESYSControl_WinCE.cfg from the „Configuration" folder.
- CODESYS Control RTE: The cfg-file is installed by the setup.

1.3 Loading and Starting the Binaries

The procedure of loading and starting the runtime hardly depends on your platform. Here are the basic procedures for our standard platforms:

- Linux:
  Extract the binaries in a folder on your target. Note, that you might have absolute paths within the tar file. So you should really extract it directly on your target if possible.
  Change the directory into the extracted tar archive and execute the „CODESYScontrol" binary.
- VxWorks:
  Download the Binary with the debugger to the target or load it with the command „ld" on the shell. Start the runtime with the command „PlcStart(char *pszFilename)":
  -> ld 1, 0, „/tffs0/CDS.out"
  -> PlcStart „/tffs0/CODESYSControl.cfg"
- Windows CE:
  Copy „CODESYScontrolwince.exe", your *.cfg file and the 3S.dat from the Configuration folder to your flash disk.
- CODESYS Control RTE:
  After setup the system can be started via the tray menu.

1.4 Change the VendorID

To identify the different kinds of targets in the network, we are using a combination of an ID for every customer (VendorID: 2 Bytes) and a target specific ID (also of 2 Bytes).

Except in the case of Linux (which is a custom build for every customer) the default TargetID and VendorID of our runtimes are in the range of 3S. To customize this to your own VendorID, you need to load an own runtime component.

You can use the template „SysTargetOEM“ as a starting point. This should already set the correct VendorID and TargetID and define a new name for the target.

- Linux:
  Your binary should already be compiled with the correct VendorID. However, if you want to overwrite the default node name or target name, you will also need to compile the SysTargetOEM component.
  There is a subfolder, called „Linux“ in the SysTargetOEM component, which contains a default makefile which can be used for this purpose. You just need to make one small adaptation to it: You need to replace the define „-DTRG_X86“ with the correct define for your platform (e.g. PPC, X86, ARM, MIPS, ...). If you are unsure which one to use, you should ask your first level support for this.
- VxWorks 6.x:
  - Open the WindRiver workbench
  - Choose „Downloadable Kernel Module“
  - Select „Create project at external location“
  - Choose the path to the SysTargetOEM directory
  - Remove the default recursive build target and select only the *.c file.
  - Go to the project properties and add the build macro -DMIXED_LINK
- Windows CE 4.2/5.0:
  - Open the EmbeddedVisualStudio
  - Create a new project for a WCE dynamic link library
  - Add the *.c file
  - Set the file path for additional header files for the runtime (components folder)
- Windows CE6.0:
  - Open The MS Visual Studio 2005/2008
  - Create a new project for Smart Devices
  - Select a platform, set the application type to “DLL” and create an empty project
  - Add the *.c file
  - Set the file path for additional header files for the runtime (components folder)
- CODESYS Control RTE:
  - Open the MS Visual Studio 6.0 (with higher versions an import of the workspace is required, some settings need to be adapted manually).
  - Open the workspace CmpDriver from the \Platforms\3SRTE3\CmpDriver_Toolkit folder.
  - Copy the file targetdefines.h from the \Platforms\3SRTE3\3SRTE3\Sys folder to \Templates\SysTargetOEM.
  - Compile the project SysTargetOEM.
  - Install the driver using the registry file from \Platforms\3SRTE3\CmpDriver_Toolkit\SysTarget_Overwrite\Registration on the target computer. Copy the .sys file to the \system32\drivers folder of the target computer.
  - Add the SysTargetOEM component to the cfg file of the RTE (not the SysTargetOEM). Make sure the OverloadableFunctions=1 option is set in the ComponentManager section.

  - o After starting the system, the RTE should be found in a network scan using the new vendor/device IDs.

To get active the component SysTargetOEM, you will need to add it to your *.cfg file. Just add it as a new component. Additionally to this, you need to specify the setting „OverloadableFunctions“ in the *.cfg file. So it will look somehow like this:

```
[ComponentManager]
OverloadableFunctions=1
...
Component.64=SysTargetOEM
```

1.5 Create a Device Description

To configure a CODESYS project for your special device, you need to use a file called device description, which has the file ending *.devdesc.xml. In this file, you will define target information, such as:

Target Name, TargetID, VendorID, Supported I/O Devices, translations of Logger Texts, memory layout, ...

In the directory „Templates/Devices“, you will find a few templates for several CPU platforms, which can be used as a starting point for your own device description. For every example we are offering some different Versions:

- Without any prefix: The default device description without any special features.
- Compact_*: Necessary for targets that are using CmpAppEmbedded, which implements a simpler download format. This download format performs better with small projects.
- SoftMotion_*: These device descriptions define some default connectors, which are necessary for SoftMotion devices.

Just take the one which fits your target best as a starting point. In most cases the templates should already work for the first tests on your target. So it will be worth a try.

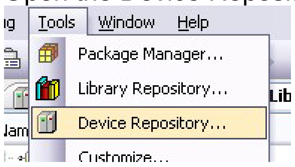
Note, that the templates are containing already your personal Vendor- and TargetID.

Note that the device description oft he RTE is shipped with CODESYS and is installed under the application data of all users in the Window’s standard folder, under \CODESYS\Devices\4096\0000 0002\<version>. This file can be used as a template for your own device description.

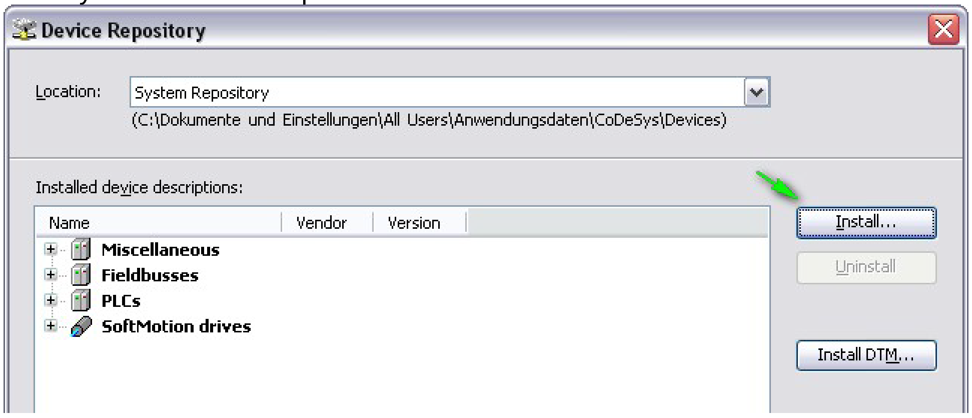
1.6 Install the Device Description

To use your device description within your CODESYS project, you will need to install it first.

- Open the Device Repository:

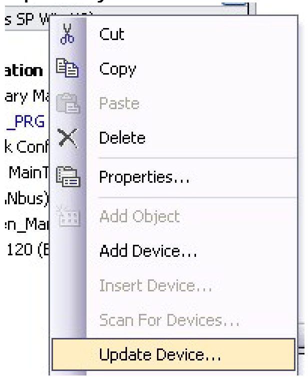


- Install your Device Description:



Note: The device will be „inserted“ (or „copied“) into the project in which you are using it. So, if you make changes in the device description, you need one additional step after installing it before the changes will take effect:

- Update your device in the project:



Note: If you don’t enable the check box „Display all Versions“, devices with the same ID are always grouped together, and you will only see the one with the highest priority.

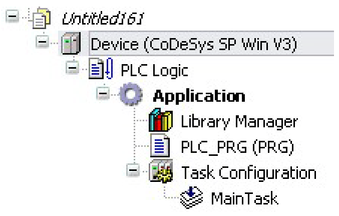
1.7 Create a Project

To test your new device, you should create a new, simple standard project:

- Click on this symbol, to get a wizard:
- Select the type „Standard Project“:

- Select your freshly installed Device Description from the drop down list and press OK.

The result should be an empty project, with one task and a POU called „PLC_PRG“, which will be your task entry:



1.8 Scan for the Target

Double click on „Device“ within your device tree. This should open a dialog, called „Communication Settings“. This dialog can be used to connect the device in your project with your physical device. If your device is connected to the network and you press „Scan network“ in this dialog, your should see your device after a short time.

If not, check the following:

- Change the „Filter“ from „Target ID“ to „None“.
  If your device appears then, you still have a wrong TargetID. Maybe your SysTargetOEM component was not correctly loaded!?
  Click on the device in the tree to see it’s configuration on the right side of the dialog.

- Is your device connected to your network?
  The default way to communicate with your device is UDP. If are using this connection method, your device needs to be in the same subnet as your host. Both devices need to have the exactly same network masked configured. Otherwise it will not work.
  You can check the network settings in the output of the runtime at startup. You should see s.th. like this:

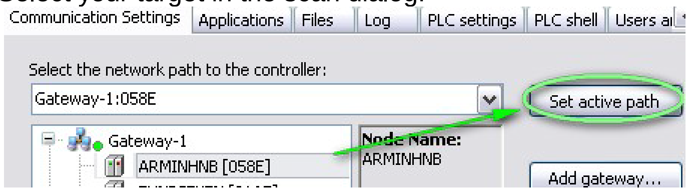
```
1287759127: Cmp=CmpBlkDrvUdp, Class=1, Error=0, Info=6, pszInfo=
Network interface: <ipaddress>192.168.101.41</ipaddress>
<subnetmask>255.255.252.0</subnetmask>
```

1.9 Login to the Target

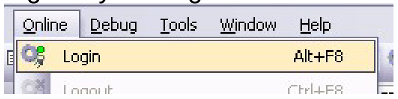
If you found your device in the scan dialog above, you can login and try your first download. Every PLC in your device tree has a scan dialog, which was described in section 1.8. That’s necessary, because you will use this dialog to create a connection between the PLC in your device tree and the physical device.

So, select your target and login:

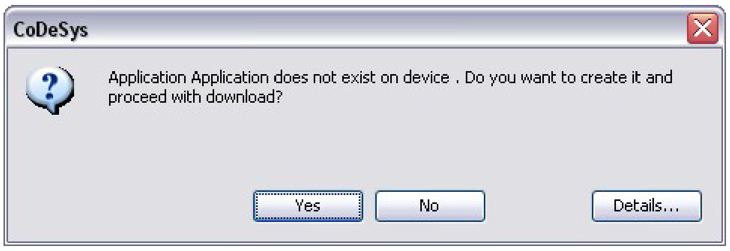
- Select your target in the scan dialog:



- Login to your target:

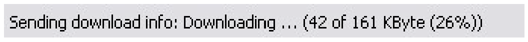


CODESYS tries to authenticate itself on that target and tries to get a „communication channel“. After a few seconds, you should get a response to acknowledge the download of the application. Most likely you will see this window, because you have no bootproject on your PLC, yet:

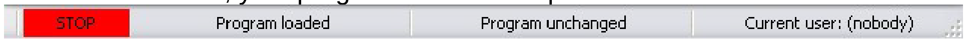

CoDeSys

Application Application does not exist on device . Do you want to create it and proceed with download?

- You will see the status of the download in the status bar of CODESYS:


Sending download info: Downloading ... (42 of 161 KByte (26%))

- After the download, your program will be in stop mode:


STOP | Program loaded | Program unchanged | Current user: (nobody)

2 Content of the Delivery

Depending on your exact order and contract, your CPU and your Platform, you might get a different delivery. Here we list the common parts as well as the general specialties of the different kinds of deliveries.

2.1 Common

There are a few folders and files, which you will find in every delivery, because they are necessary to do brand labeling on your PLC or to extend the CODESYS Control runtime with I/O drivers or own components.

- **Components/*Itf.h, Components/*Itf.m4**
  These are the interfaces of the 3S CODESYS Control components. Those interfaces can be used or implemented from your OEM specific components and I/O drivers.
- **BuildUtils/msys/bin/*.bat**
  This directory contains a few batch scripts that can be used to compile the M4 files of our or your own components. Those M4 files are describing the interfaces and dependencies of the component and generate the header and C++ files based on that.
- **Configuration/*.cfg**
  You will find several examples of configuration files in this folder. You can use them as a starting point for your own configuration. If you want to have an overview about the available configuration entries, you can check Runtime System Reference document, which you find in the documentation folder.
- **Configuration/3S.dat**
  This file contains license information for your ordered field busses or advanced features, like SoftMotion or Visualization.
- **Platforms/*/**
  Depending on the platform that you ordered, you may find a few system specific files below this folder.
- **RtsConfigurator/Bin/RtsConfigurator.exe**
  This tool is included in every runtime delivery. However, it is only really useful for those, who ordered the complete source code of the runtime. It is used to resolve the dependencies between the components and create a configuration for a custom built runtime.

2.2 Standard Platforms

Standard Platforms are for example RTE, Linux, VxWorks, WinCE, ... If you have one of those platforms and didn't order a „Source Delivery“, you will get a binary package of the runtime, which is specifically built for your platform.

In most cases these binaries will be included directly in the delivery in the subfolder „Platforms/*/Bin/*“. However, if you use a newer or exotic platform, it is possible that we need to make a custom build, specifically for your target (maybe with your tool chain). In those cases you will get a Starter Package without any binaries. The binaries will be delivered from your first level support during the adaptation.

Note: Most Linux platforms need to be built with the tool chain of the OEM. So you will most likely get a Starter Package without any binaries.

2.3 Custom- / Embedded Platforms

If you don't have an operating system, or you have a system that we don't yet support, you will retrieve an embedded delivery. This kind of delivery contains all sources of the runtime, so that you can port it to your OS or CPU.

By default you get a folder with some empty implementations of the system layer, which you can find in the folder „Platforms/SysTemplates“. Furthermore you might get some help and maybe some platform templates for your CPU or OS from your first level support (just ask).

2.4 Source Deliveries

Source deliveries are very similar to embedded deliveries, because they are also containing all the runtime sources. But a source delivery is always specific for one of the standard platforms and contains all the source code of that platform adaptation.

There is no common way of compiling those platforms. This is something you should learn during your initial workshop.

3 Handle the TargetIDs and Signatures

Every derivate of your targets will need to have their own TargetID. The TargetID is used to have a clear linkage between your device Description and your Target Runtime. So every time you need to make a change in the device description for a new target, you will need to request a new TargetID from 3S.

This is necessary, because one part of the target signature is the TargetID. So, you will most likely need a new signature, too.

If you have a huge amount of targets, you might like to manage them with the same target signature. In this case, you can request a range of signatures from 3S. Then you will get a new targetdefines.h which contains a new Signature in combination with a Device Mask. This is then valid for a specific range of TargetIDs.

To include the new targetdefines.h in your system, you need to use the component SysTargetOEM, which is described in section 1.4.

4 Manage the License Files

Every field bus that you are using on your PLC, needs to have a valid license from 3S. Additionally there are a few features, like the Target Visualization or Soft Motion, which are secured by a license on the target.

All those licenses are contained in a file, called 3S.dat. This file needs to be accessible from IEC on your target. You can do this like this:

```
[SysFile]
FilePath.1=ide0:/root
```

Alternatively, you can place only files with the extension *.dat to a special folder:

```
[SysFile]
FilePath.1=ide0:/root
FilePath.2=ide0:/license, *.dat
```

If you have an embedded system, where you don’t have a file system, but you are using SysFileFlash to manage your files, you can use the tool „ConvertFileToArray“, which you will find in the CODESYS Control Delivery. This tool will output a C-Preprocessor define, that you can copy to your „sysdefines.h“. Just name the define like this:

```
#define LICENSEFILE_DAT {\
0xd0,0x4e,0x7f,0x23,0xe6,0xba,0x82,0xa3,0xf8,0x22,\
...
```

Change History

Version	Description	Editor	Date
0.1	Issued	IH	12.10.2010
0.2	Review, ok	AF	08.11.2010
1.0	Release	MN	11.11.2010
1.1	Changed Template name of CmpOEMTarget	IH	29.11.2010
1.2	Added description to start the VxWorks Runtime	IH	02.12.2010
2.0	Release after formal review	MN	02.12.2010
2.1	CDS-29303; Spelling corrections	MN	17.09.2012
3.0	Release	MN	03.12.2012