RENESAS synergy™

Accelerate. Innovate. Differentiate.

CANopen®

for

S1 S5 S7

Quick Start Guide

port PROFESSIONAL INDUSTRIAL COMMUNICATION

Caution: Make sure that the EVAL board you are using is safely installed according to the instructions!

Caution: The follow described software is for use for evaluation and test only!

Table of Contents

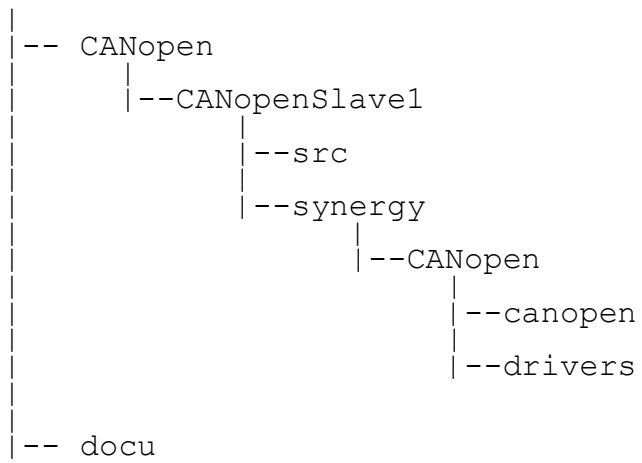
1 Project overview

The port CANopen implementation on the Renesas Synergy platform.
Based on the Synergy Software Package (SSP Version 1.1.1).

Environment:

- Evaluation-Board: Renesas SK-S7G2
- Compiler: GNU for ARM gcc v4.9
- IDE: e2-studio Version: 5.0.0.43

2 Software structure



From port generated folders (e.g. unfolded CANopenSlave1 project):

➔ CANopenSlave1 - Synergy Project structure, with included Synergy configuration file (configuration.xml), module descriptions and e2-Studio project files.

- synergy/CANopen/canopen - CANopen stack from port
- synergy/CANopen/drivers - CANopen drivers (CAN, CPU and CANopen timer)
- src - sources from port example code (for a simple CANopen device).

➔ Documentation - user manual, reference manual from port CANopen stack

Note: All other developed example projects have the same structure.

3 CANopen examples

In this framework implemented CANopen example projects:

Name	Typ	CANopen Services	Notes
CANopen - Slave s1	Slave	NMT-Slave, SDO-Server, Heartbeat, PDO	The node ID is 32 and all communication objects have the default COB-IDs.
CANopen - Slave s2	Slave	NMT-Slave, SDO-Server, NodeGuarding, Emergency Producer, PDO	The node ID is 32 and all communication objects have the default COB-IDs.
CANopen - Slave s3	Slave	NMT-Slave, SDO-Server, Sync-Consumer, NodeGuarding, PDO	The node ID is 32 and all communication objects have the default COB-IDs.
CANopen - Master m1	Master	NMT-Master, SDO-Server/Client, Node-Guarding, Emergency	Implementation of a CANopen master device with a minimum application functionality. The node ID is 5 and all communication objects have the default COB-IDs.
CANopen - Master m2	Master	NMT-Master, SDO-Server/Client, Heartbeat	Implementation of a CANopen master device with a minimum application functionality. The node ID is 5 and all communication objects have the default COB-IDs.

CAN-Bitrates is set in the main file from the examples default on 125Kb.

4 Notes

The example projects are supported by ThreadX RTOS and be based on the "BlinkyThreadX" template. The ThreadX components, the BSP and the peripheral driver interfaces (HAL) was created and generated with the Synergy configurator.

For further notes please read the CANopen driver Readme.
(../synergy/CANopen/drivers/synergy/Readme)

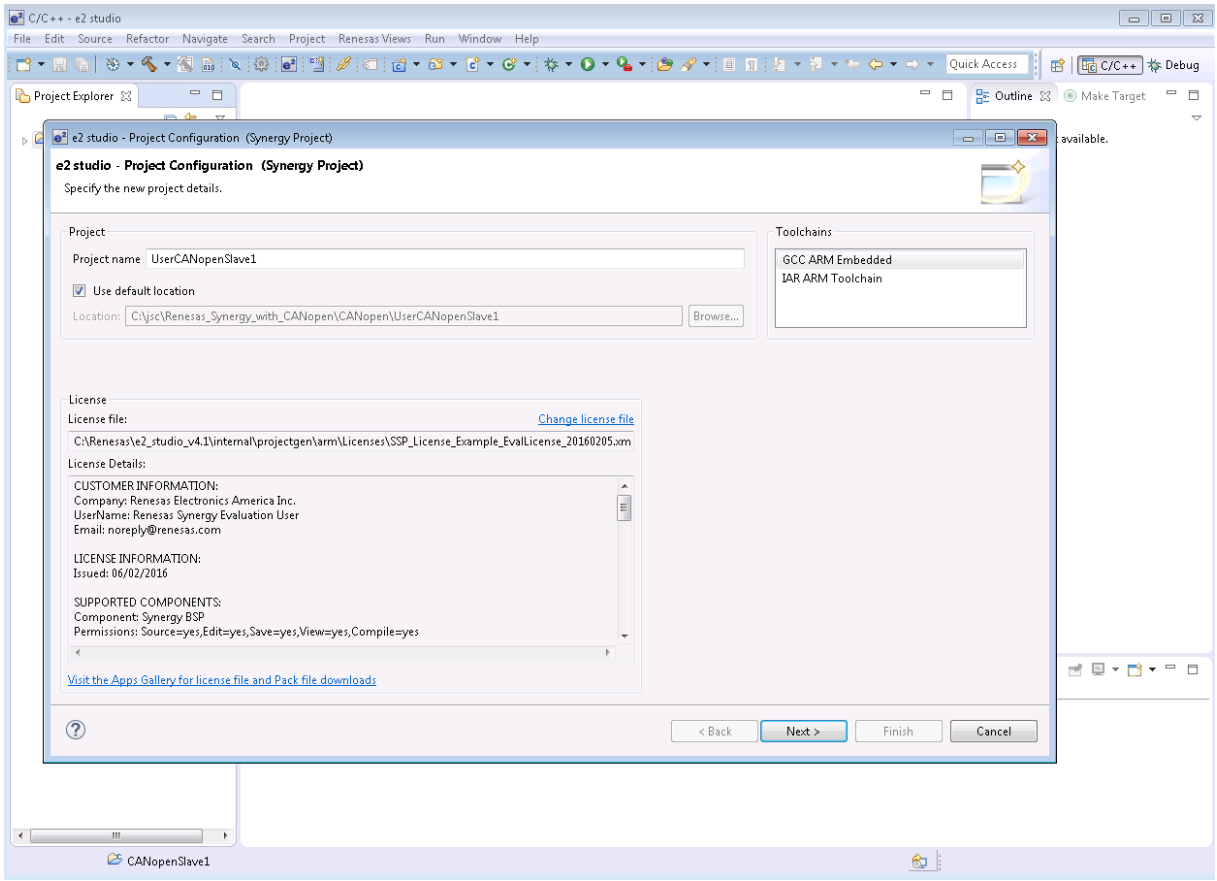
5 Initial quick starting

5.1 Create a new user project with CANopen in Synergy framework:

Prerequisite:
The example project CANopenSlave1 from port is already loaded in the e2studion IDE.

5.1.1 Basic steps

- ➔ Create a new project over the menu “File - New - SynergyProject ”
- ➔ assign a name for the new project (e.g. UserCANopenSlave1) and set a location for the project

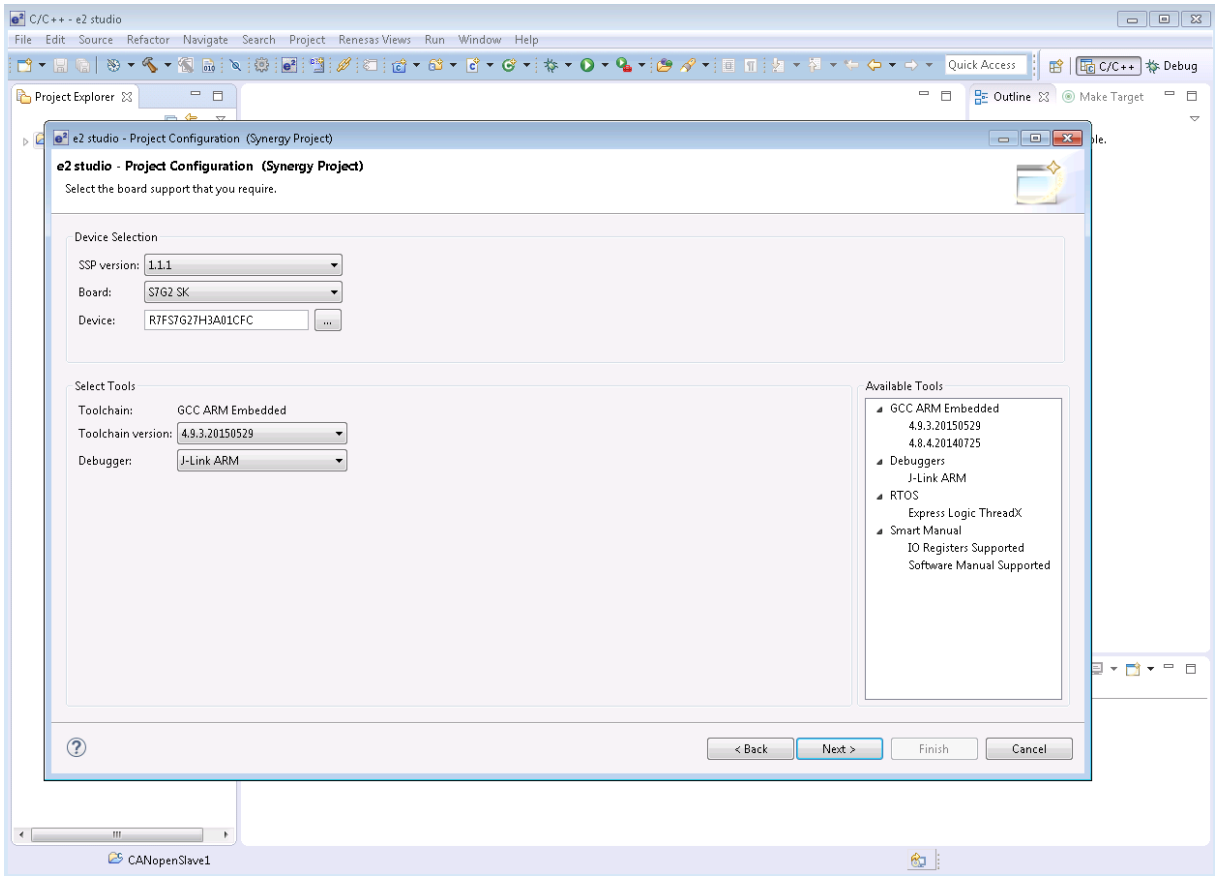


In the next step select the used toolchain and board support:

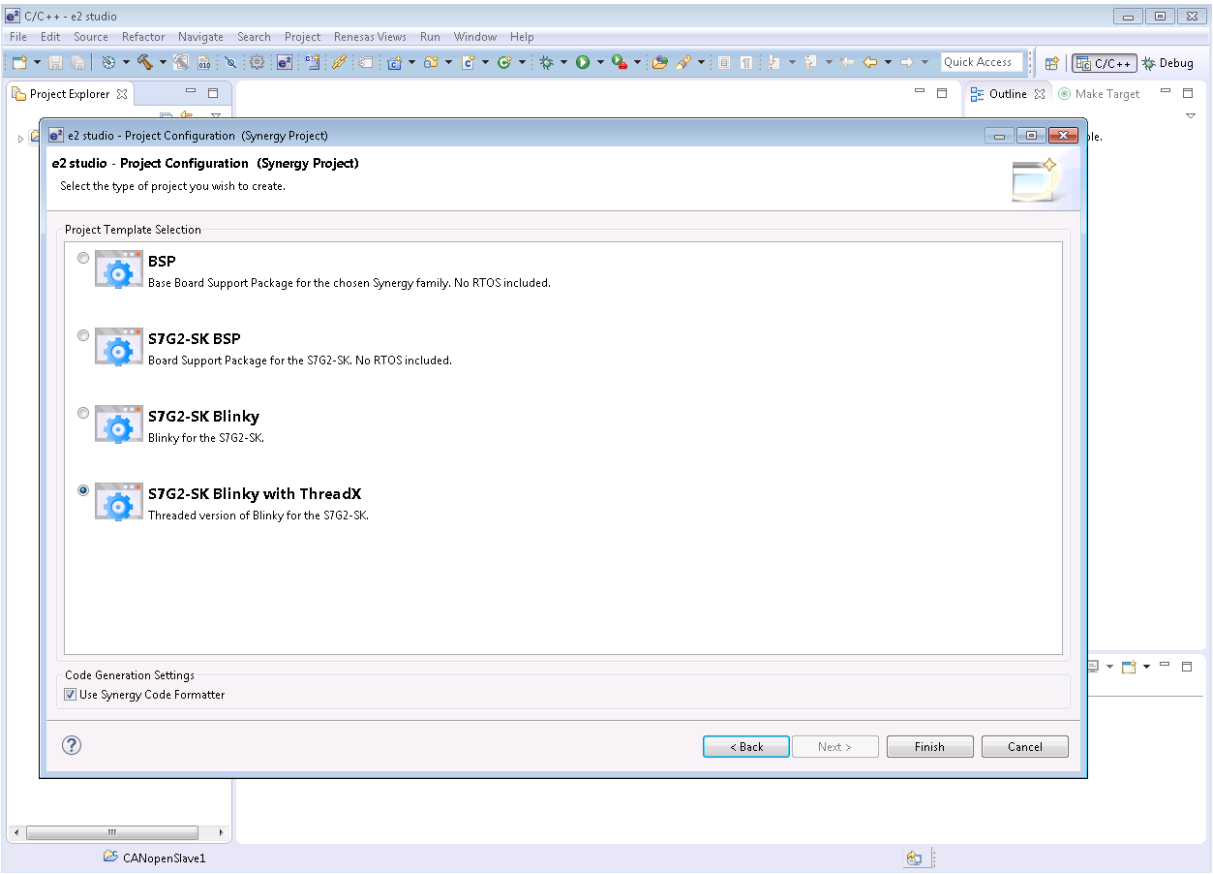
- ➢ Synergy software package (ssp version 1.1.1)
- ➢ Board S7G2-SK (S1, S5)
- ➢ Compiler ARM GCC v4.9

- Debugger J-Link ARM

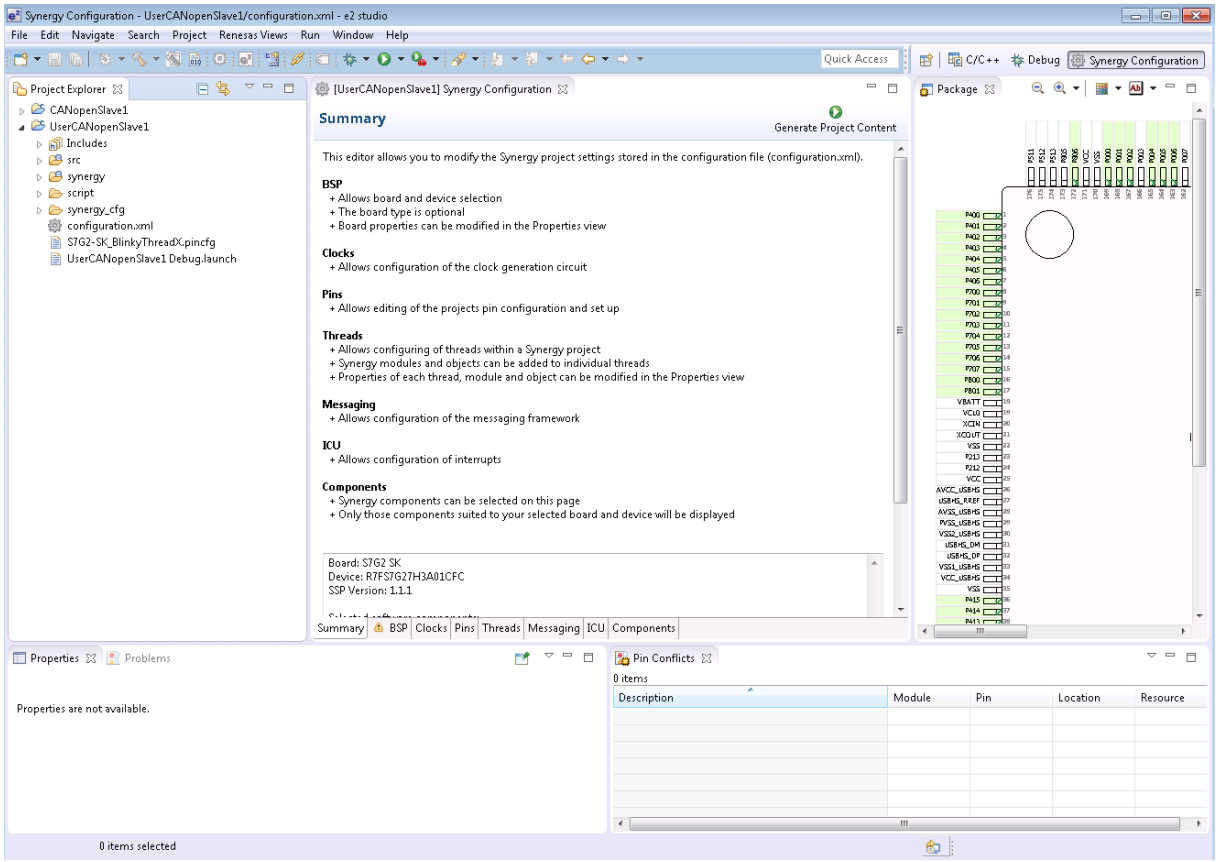
This environment is current supported and tested from port.



➔ next step you select a project template. Please select the “S7G2-SK Blinky with ThreadX” The CANopen implementation from port based on this template.

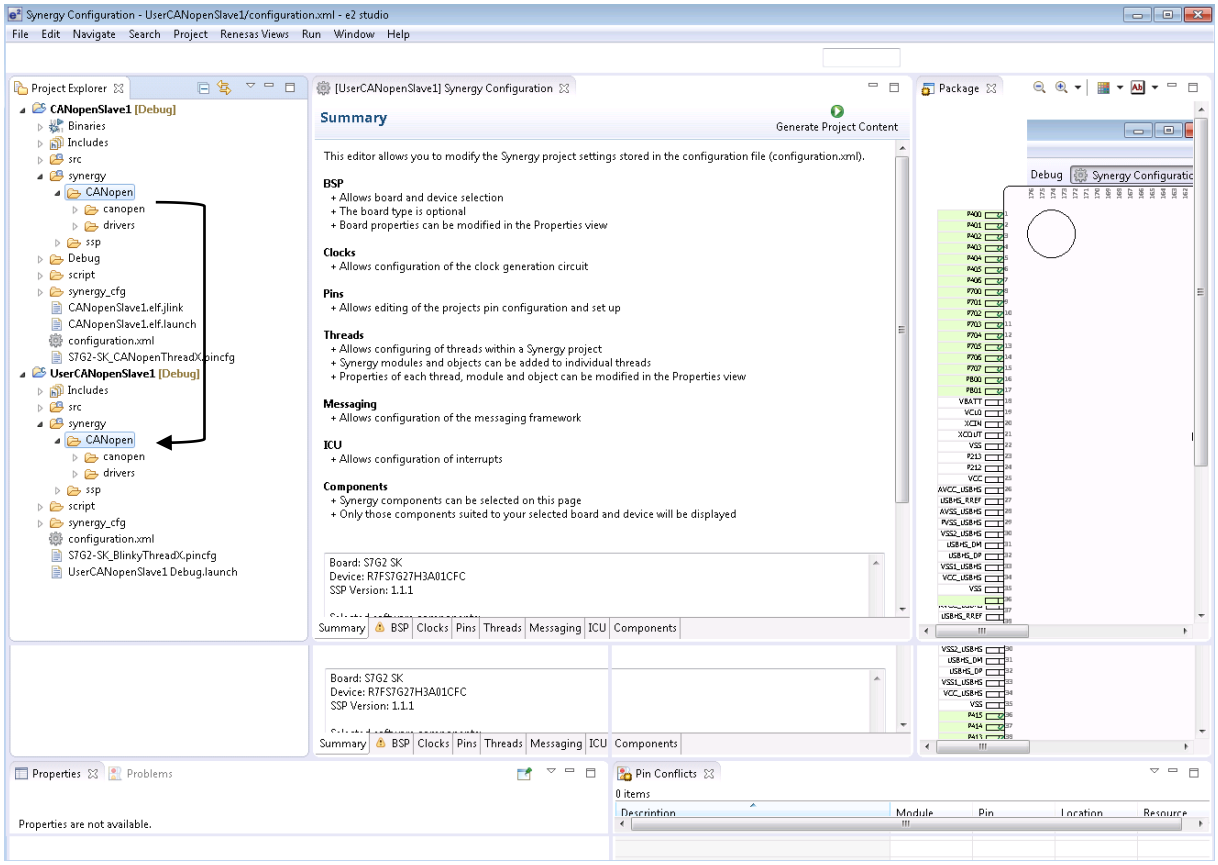


➔ after you pressed the Finish-Button you can see your new created project in the “Project Explorer” from e2studio with the typical folder structure for a Synergy project, beside the CANopenSlave1 example project from port.



5.1.2 CANopen integration

➔ next step you should copy and paste the CANopen Stack and drivers from port example project CANopenSlave1 to your user project. This software packages you can find under the synergy folder. You can copy and paste inside the e2studio IDE.



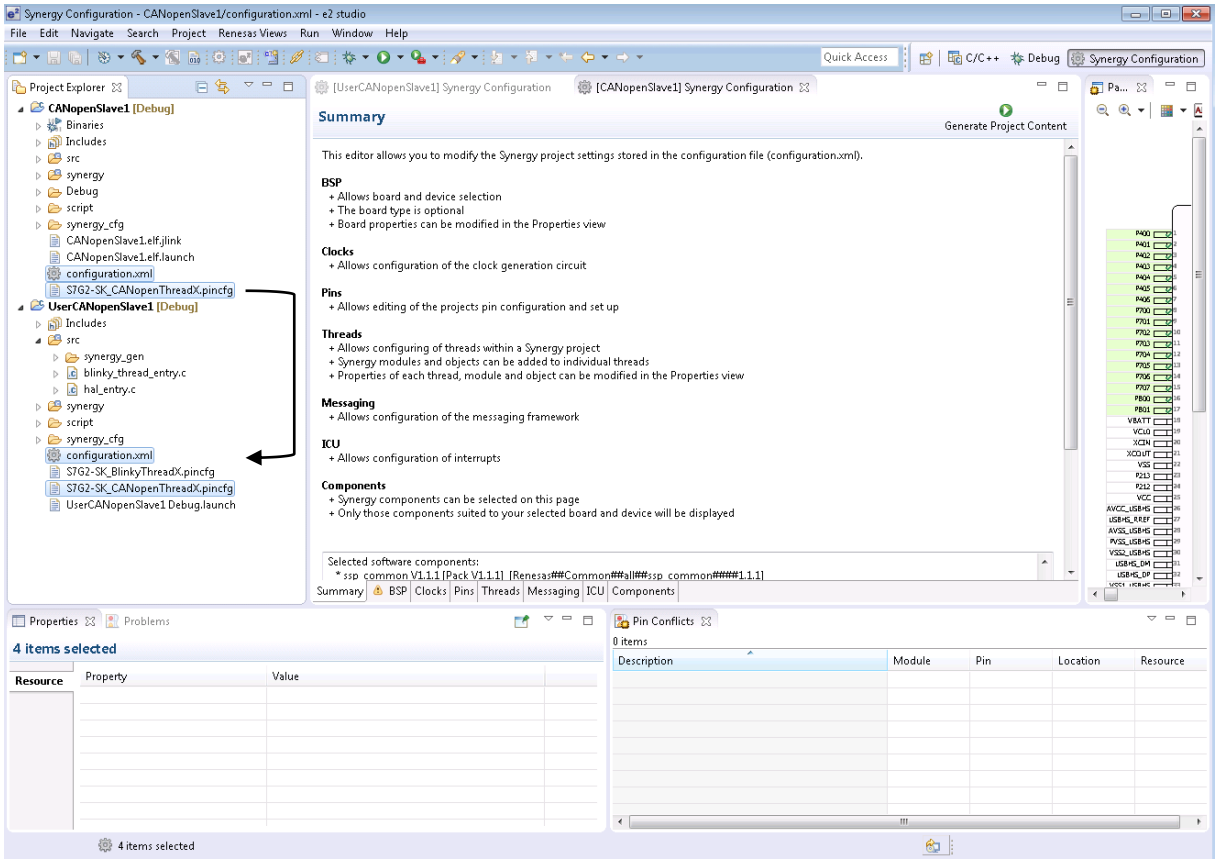
5.1.3 Prepare the Synergy configuration for CANopen

Additional to the blinky template with ThreadX we need some Synergy components and objects for a CANopen project.

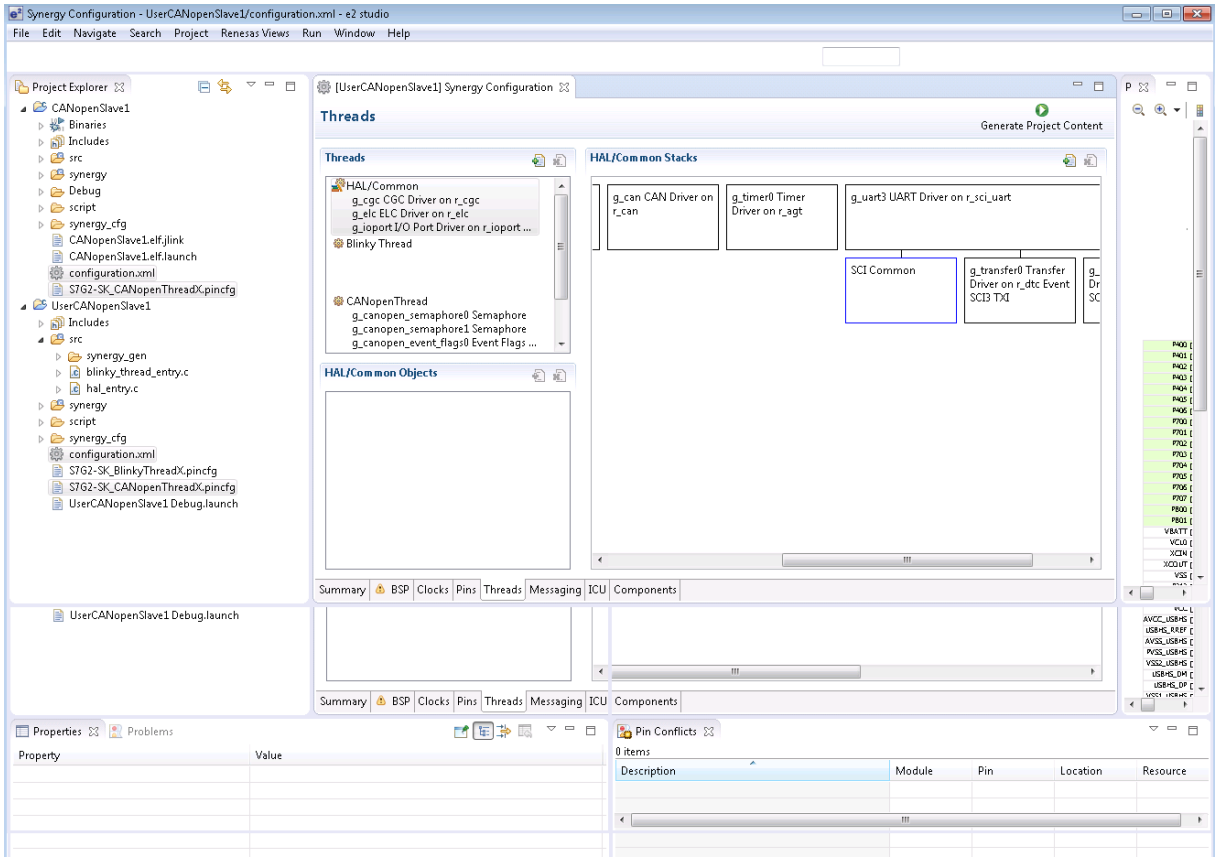
These are:

- ➔ HAL-driver CAN driver (r_can), Timer driver (g_agt), optionally Uart driver (r_sci_uart)
- ➔ CANopen Thread with 2 semaphores, 1 event flag, 1 mailbox queue

For a quick start it's recommended you copy and paste the files configuration.xml and S7G2-SK_CANopenThreadX.pincfg from the port example. This overwrites the template setting in the new project. This configuration files have made all necessary settings for use the CANopen functionality with Synergy.
Alternatively, you must add all components and objects in the new project like the example project.



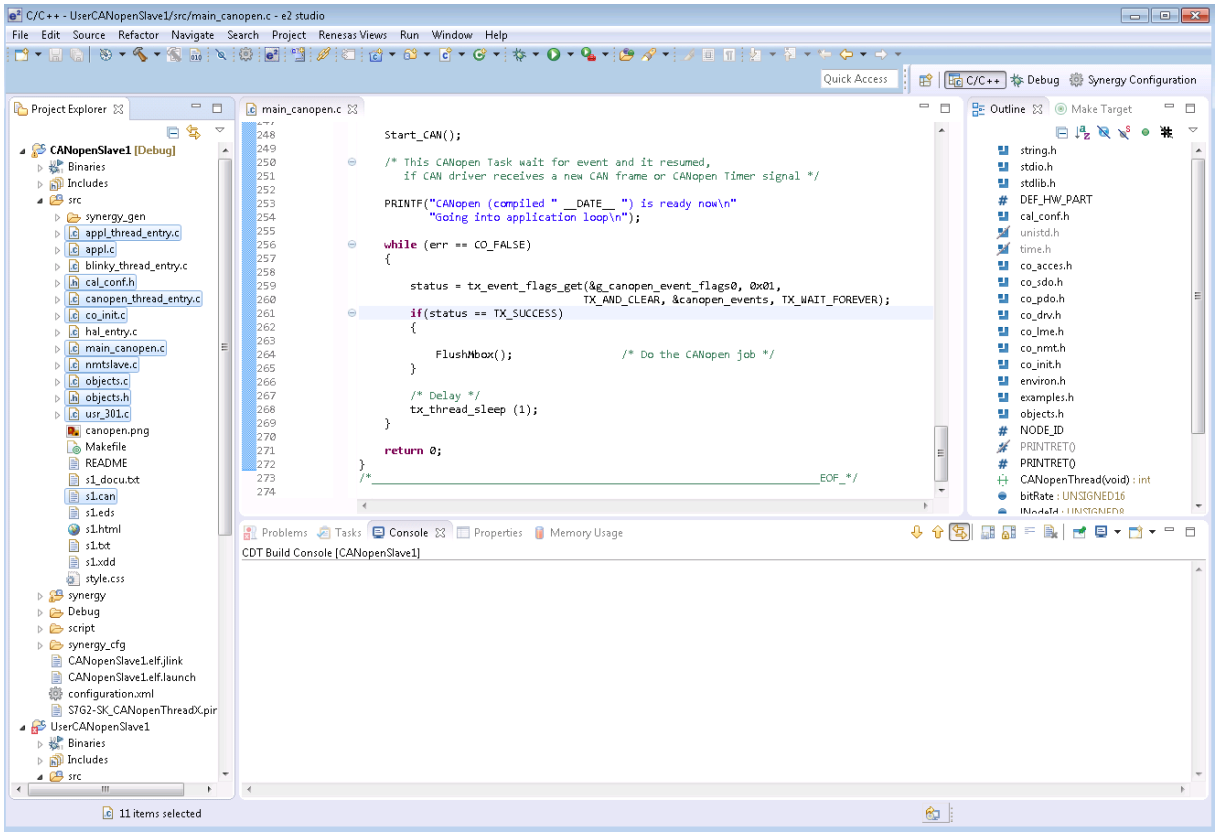
After this step you see the created Objects in the Threads view from project Synergy Configuration (configuration.xml).



Just press the button “Generate Project Content”. This step will generate all Synergy sources in order to the configuration setting.

5.1.4 First CANopen Application

- ➔ For a CANopen application you need additionally a basically set of source files.
- ➔ For a quick start it’s recommended you copy and paste the highlighted application files from the port example src folder into the src folder from new project. This files included all necessary sources for a simple CANopen slave functionality.



They are:
Main_canopen.c - CANopen Thread call initializations from stack and drivers further include the main loop with the cyclic functionality from CANopen

- ➔ appl.c - a simple application thread
- ➔ usr_301.c, nmt_slave.c - user indication functions called from CANopen stack
- ➔ s1.can – DesignTool project file
- ➔ application_thread_entry.c, canopen_thread_entry.c – entry point and joining layer between Synergy generated Thread objects and own sources with Thread functions

From port DesignTool generated sources:

- ➔ cal_conf.h - configuration file for the CANopen stack and drivers
- ➔ co_init.c - customized initialization functions for the stack
- ➔ object.c, object.h - object dictionary from example

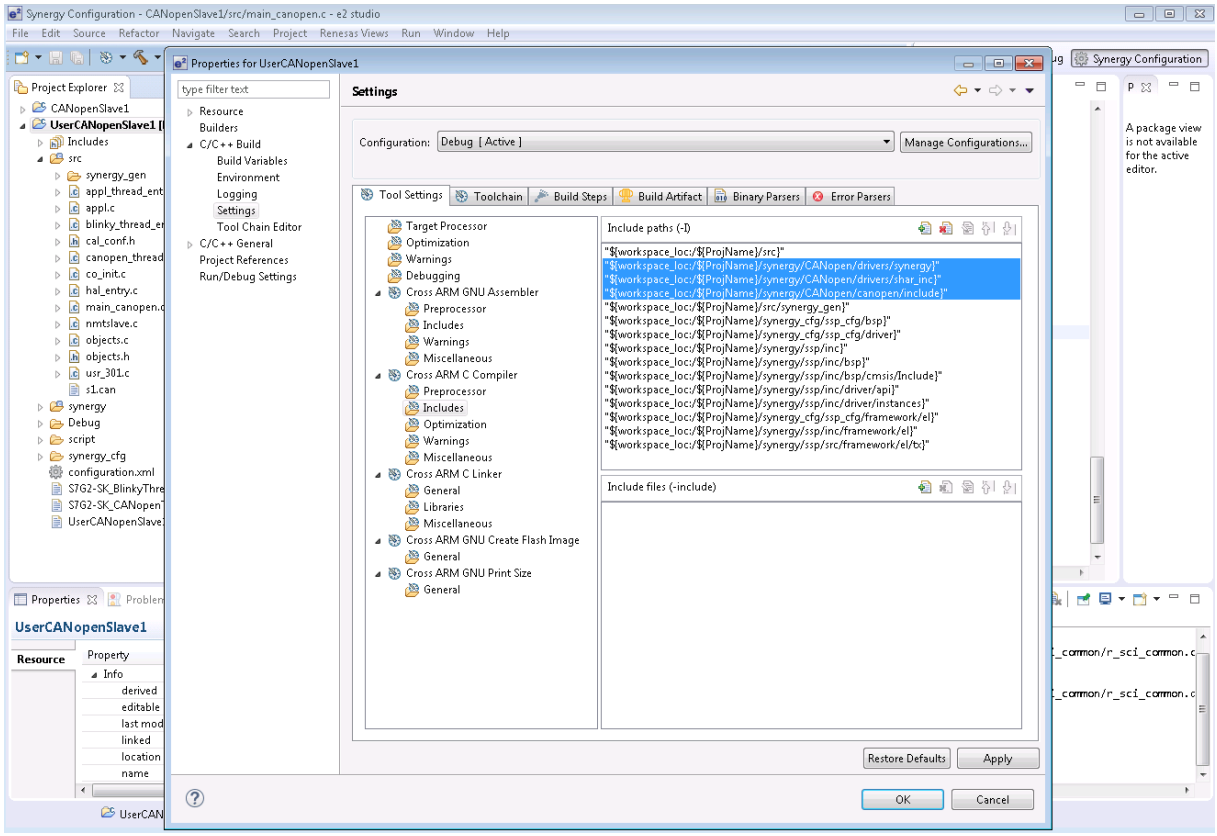
You can adapt this sources on your requirements or alternatively you have the possibility to included your own sources.

In the next step you must add the following include paths in the compiler settings from new project

- ➔ "${workspace_loc:/${ProjName}/synergy/CANopen/canopen/include}"
- ➔ "${workspace_loc:/${ProjName}/synergy/CANopen/drivers/shar_inc}"

➔ "${workspace_loc:/${ProjName}/synergy/CANopen/drivers/synergy}"

This paths are including for the CANopen stack and drivers.



Just should make the project successful build.

Import the project into your ***e2-studio workspace*** by ***opening "File -> Import"*** and ***selecting "Existing-Projects"***.

The project files can be found in the folder (../CANopen/CANopenSlave1/).
(e. g. project CANopenSlave1)

In order to generate the prepared Synergy components from the Synergy Configurator.
Open the file "configuration.xml" and press the button "Generate Project Content".
Once all necessary files were created, the project can be built and loaded into the Synergy S7G2 target.

6 Service and Support

If you have questions or problem´s please contact our service team:

Life Chat: http://synergyxplorer.renesas.com/support/feature/chat

7 helpful Tools

For a quick success and better results, we recommend the use of our CANopen Design Tool (CDT)

The free trial version (WIN and LINUX) you can download here.

http://www.port.de/en/products/canopen/tools/canopen-design-tool.html#tab-9

8 License Terms for the Evaluation Version of the Binary (demo)

License Terms for the Evaluation Version of the Binary (demo)

Licensor: port GmbH, www.port.de
Licensee: You - the user of the demo

This demo is intended to be used in a disconnected laboratory environment (stand-alone network) to verify fit and basis functions. Please back-up all data before employing the demo.
It comes AS-IS, no warranty of any kind - not expressed, not implied. Use at your own risk. The demo is functionally and timely limted and must not be used in any real-life network.

The demo has the status of a Proof-of-Concept; to show the capabilities of the software in a specific environment.

You are granted the non-exclusive License to run the demo for evaluation for testing as long as you are in legal possession of the corresponding original CD (carrying this demo) and only as long as you use the License for the intended use for non-commercial evaluation.
You are not permitted to reverse engineer, dis-assemble or modify the binary software. You have no rights to re-distribute or otherwise publish the demo. All rights reserved. All and any not expressively granted rights remain intellectual property of the Licensor.

www.port.de - June 2016

9 Space for your own notes