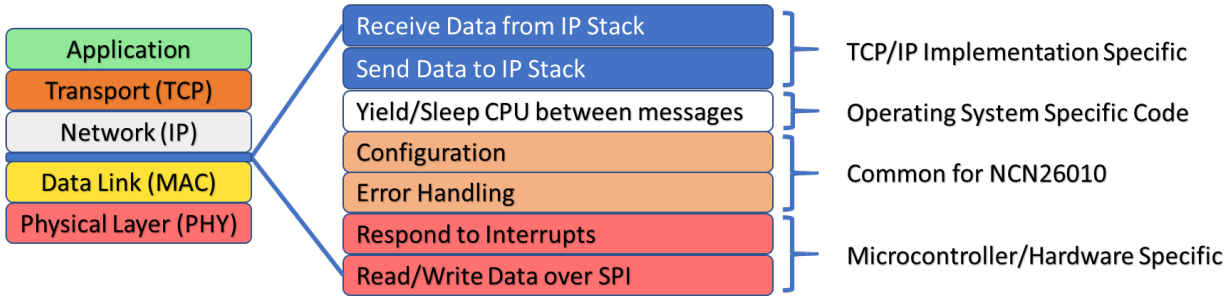
NCN26010 Example Software Users Guide

Version 1.0.0

The NCN26010 device is an Ethernet Transceiver designed for industrial multi-drop Ethernet. It provides all physical layer functions needed to transmit and receive data over a single unshielded twisted pair. The example software provided by onsemi is designed to provide the necessary firmware and application examples to help designers quickly integrate the NCN26010 into their systems.

Software Architecture

The provided software follows a modular design to allow the software to be quickly adapted to be used with any microcontroller or to be used with various implementations of the TCP and IP layers of the network stack. As illustrated below, the standard network stack consists of five layers: Physical (PHY), Data Link (MAC), Network (IP), Transport (TCP), and Application. The NCN26010 provides a hardware implementation of the MAC and PHY layers of the network stack. There are many existing open source or proprietary implantations of the TCP and IP layers that a designer may choose from. The software provided with the NCN26010 is designed to provide the connection between the IP and MAC layers of the stack.



The provided software includes three required modules and one optional module. The required modules include a module to handle interaction with the TCP/IP Implementation, a module to handle hardware specific details, and the core driver module that configures the NCN26010 and handles errors. The optional module handles interactions with the operating system to yield the CPU to other tasks when not processing messages. If no OS module is provided the system defaults to call functions in the hardware specific module to sleep the CPU until the NCN26010 signals data is available with an interrupt. The interfaces for these modules are described in T1S_TCP-IP.h, T1S_Hardware.h, NCN26010.h and T1S_OS.h, respectively.

The core driver module remains unchanged for any combination of hardware and TCP/IP implementations. The hardware specific module and the TCP/IP modules are intended to be implemented by the system designer to match their system. We provide the following example implementations to support systems with the following hardware and TCP/IP implementations:

		Microcontroller		
		Raspberry Pi (Zero, 3B)	STM32 (L4xx)	RSL10
TCP/IP	Linux TAP	✓		
	Lightweight IP (LwIP)		✓	✓
	FreeRTOS TCP-IP		✓	✓

If other microcontrollers or TCP/IP implementations are required for the system, only the functions defined in T1S_Hardware.h or T1S_TCP-IP.h need to be implemented.

Installing the software

Raspberry Pi

If you are using the NCN26010XMNEVK getting the system up and running is simple.

1. Install the latest version of Raspberry Pi OS onto a SD card as described at https://www.raspberrypi.com/software/
2. Download and extract the Raspberry Pi NCN26010 software package on the raspberry pi. This can be accomplished with the following command:
```
curl -o NCN26010Firmware.zip "REPLACE_URL_TO_ZIP_FILE_HERE" && \
unzip NCN26010Firmware.zip && \
rm RaspberryPiFirmware.zip
```
3. Install the required dependencies:
```
sudo apt install pigpio
```
4. Enter the new directory and execute the makefile:
```
cd NCN26010_RaspberryPi && make
```
5. Execute the compiled program (root access is required to access SPI peripherals)
```
sudo ./NCN26010_Driver
```
6. Setup the a TAP connection for the user space driver to handle IP traffic.
```
./netconfig
```
7. Test by pinging on the 192.1.1.x network. (By default our IP address is 192.1.1.2)
```
ping 192.1.1.1
```

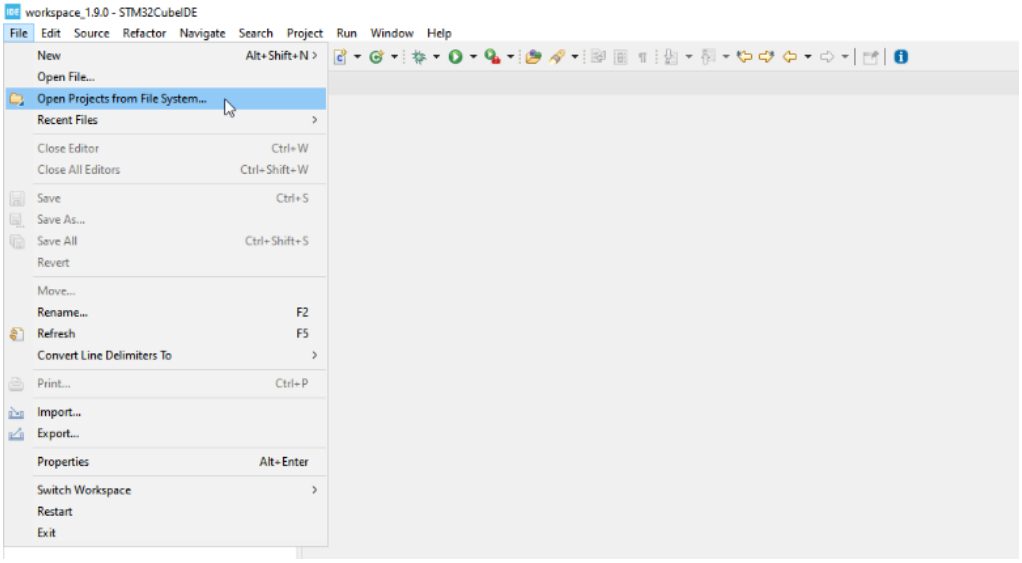
STM32

Install STM32CubeIDE

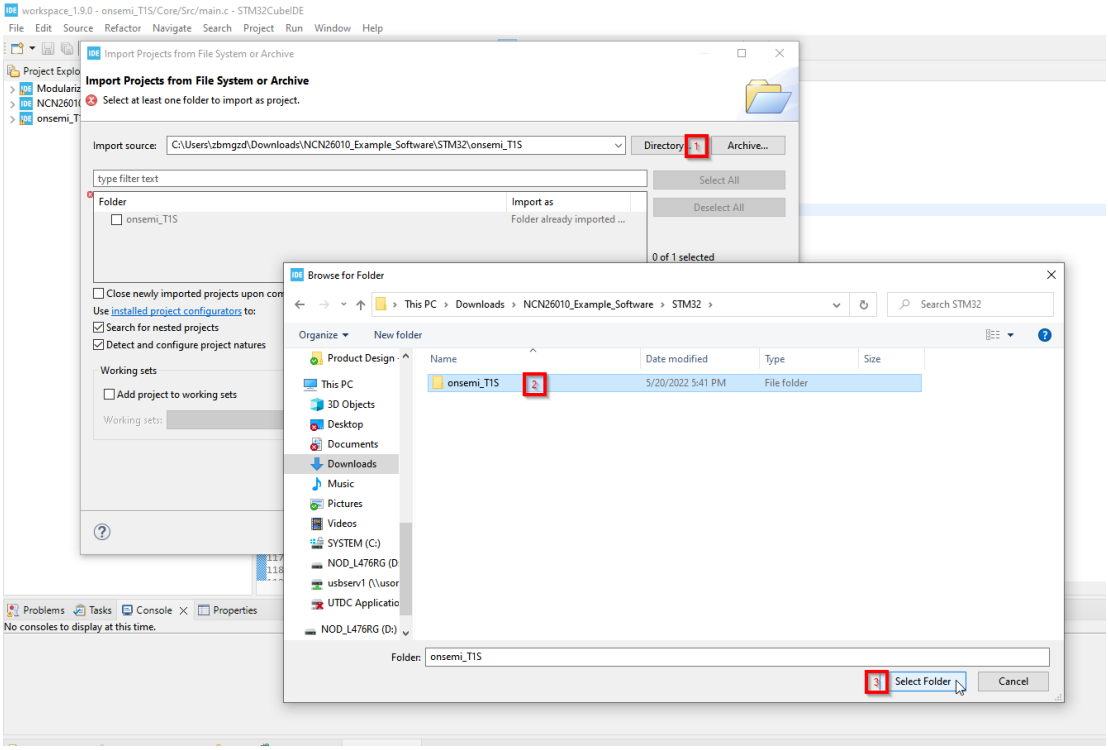
Follow the instructions provided by ST Micro here:
https://www.st.com/resource/en/user_manual/um2563-stm32cubeide-installation-guide-stmicroelectronics.pdf

Open the Configured Example

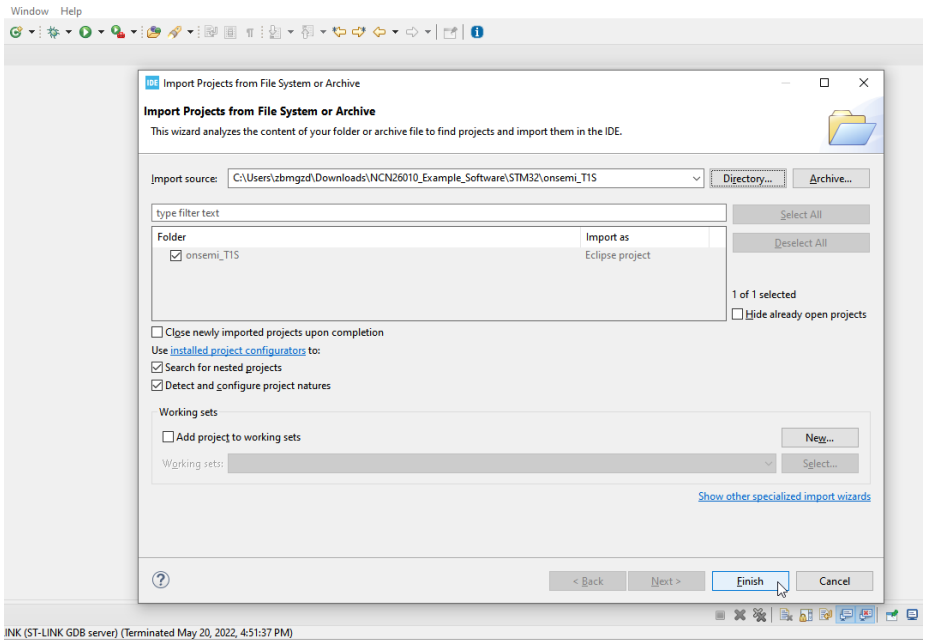
1. Under file, select “Open Projects from File System…”



2. Select Directory and navigate to the folder called onsemi_T1S

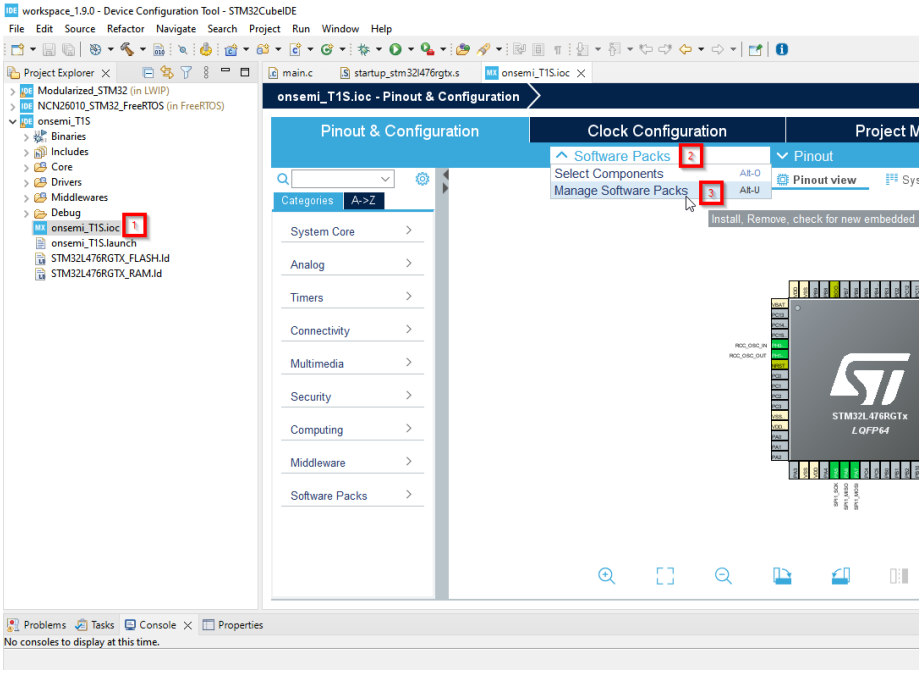


3. Leave the default values and select finish.

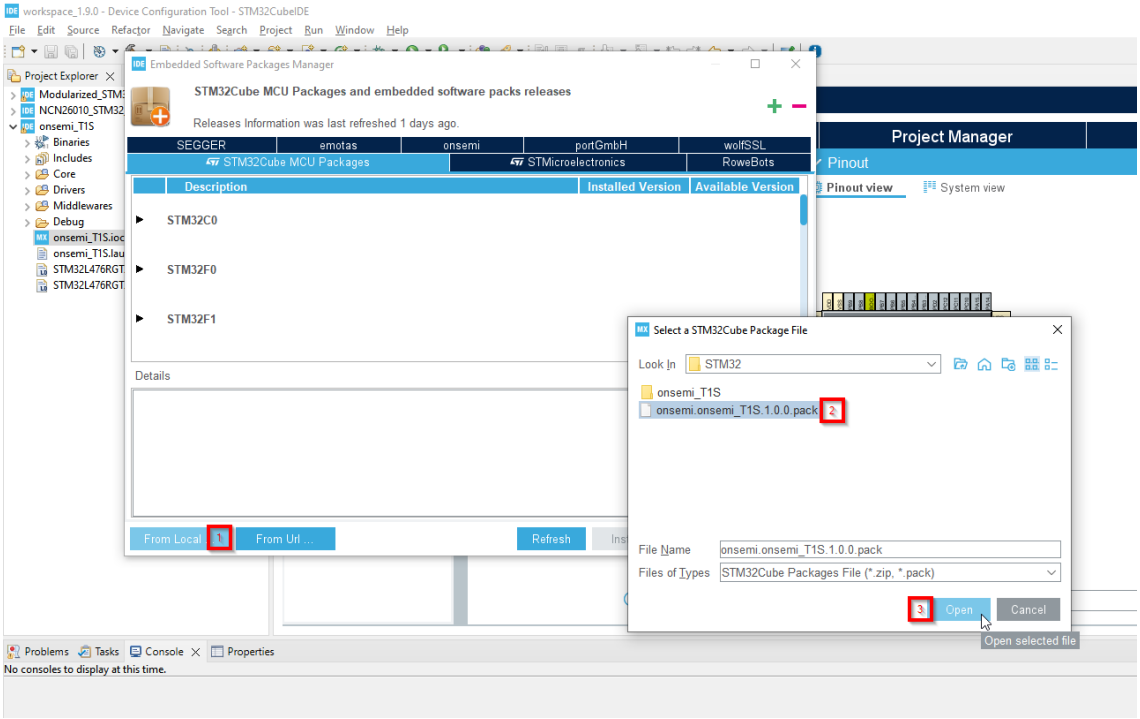


Install the onsemi_T1S Software Pack

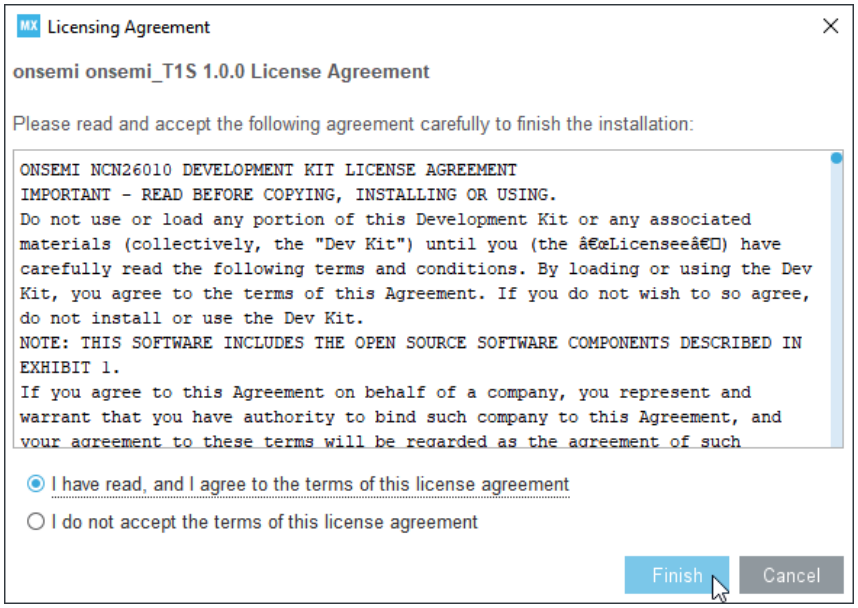
1. Double click onsemi_T1S.ioc, once the GUI loads select “Software Packs,” then Manage Software Packs



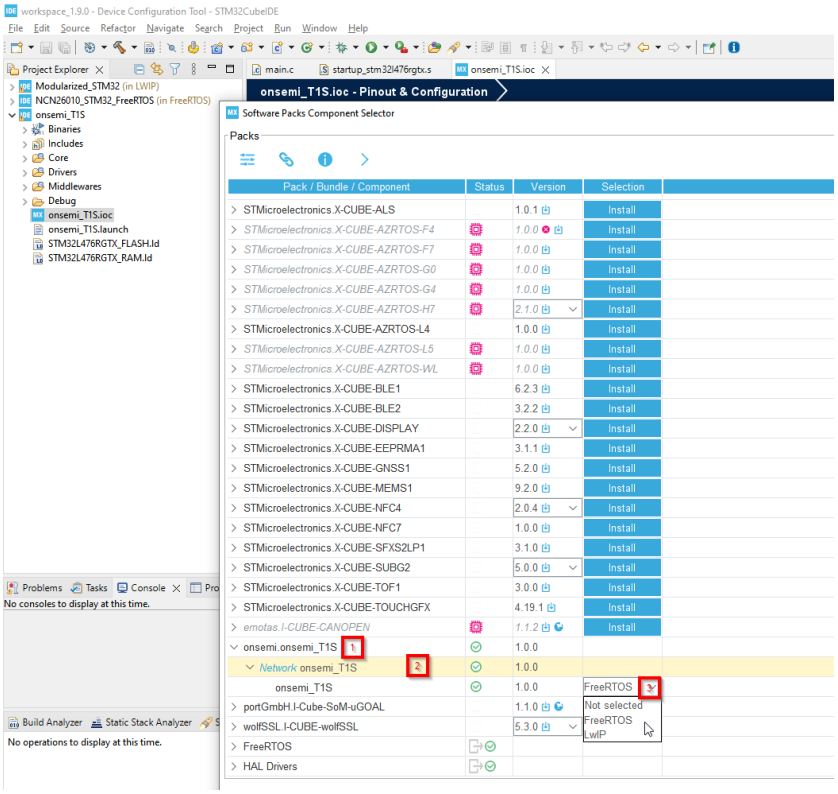
2. Select “From Local…” then navigate to onsemi.onsemi_T1S.1.0.0.pack, and open it.



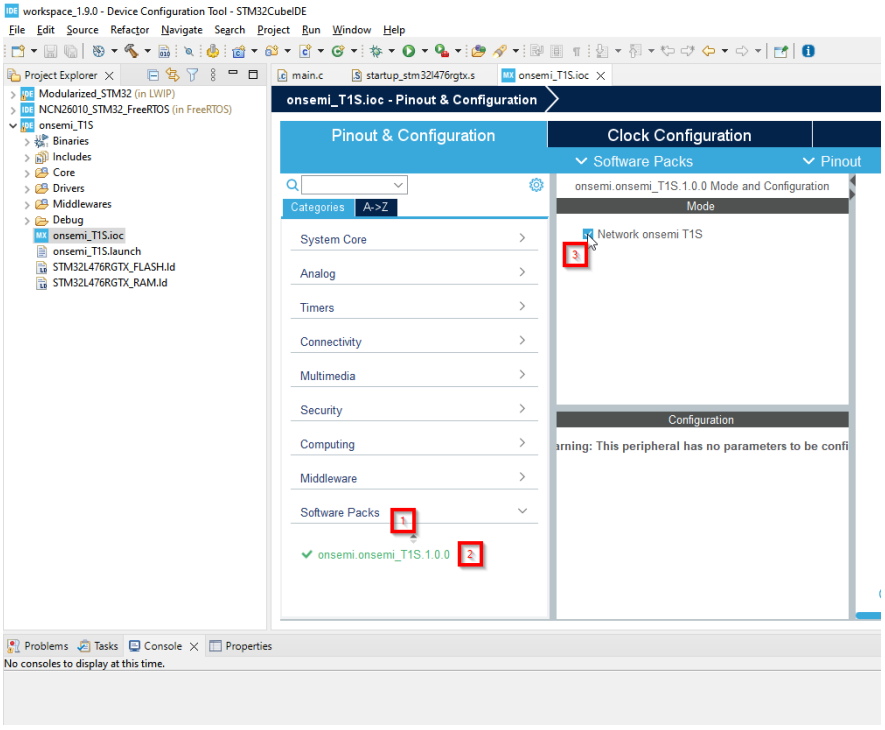
3. Read and Accept the License Agreement then close the manage dialog.



4. Then select “Software Packs” again but this time click “Select Components.” Navigate to onsemi.onsemi_T1S, Network onsemi_T1S, then select either FreeRTOS or LwIP depending on which TCP/IP implementation you would like to use.



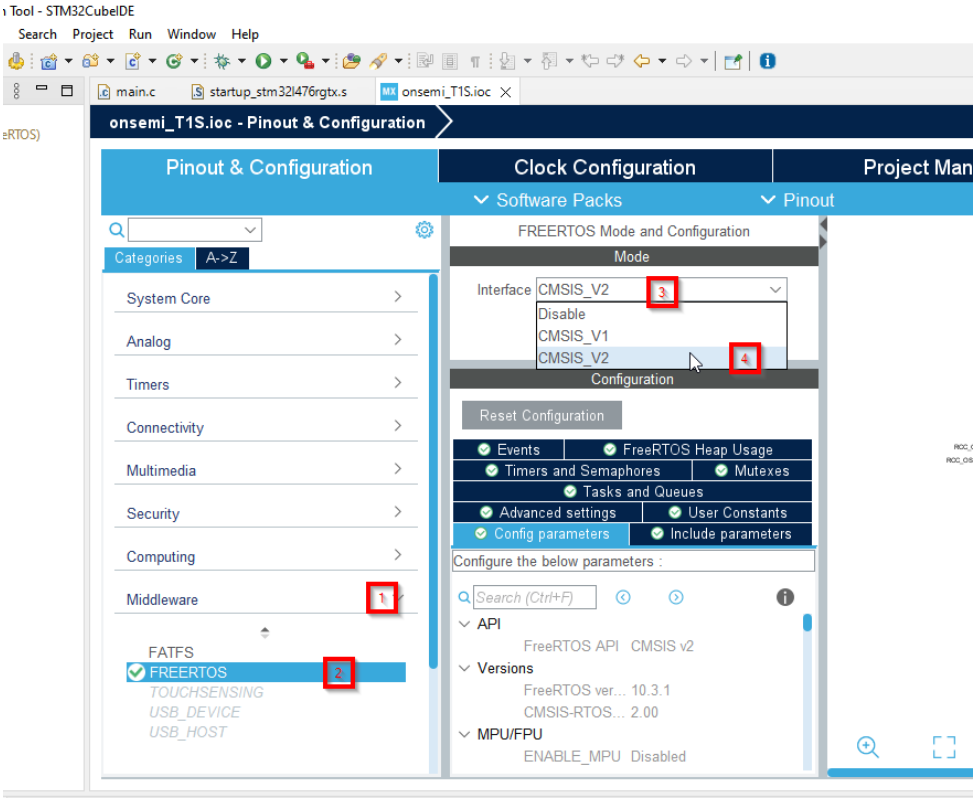
5. Enable the onsemi_T1S software pack by selecting “Software Packs,” “onsemi_T1S,” then check the box next to “Network onsemi T1S”



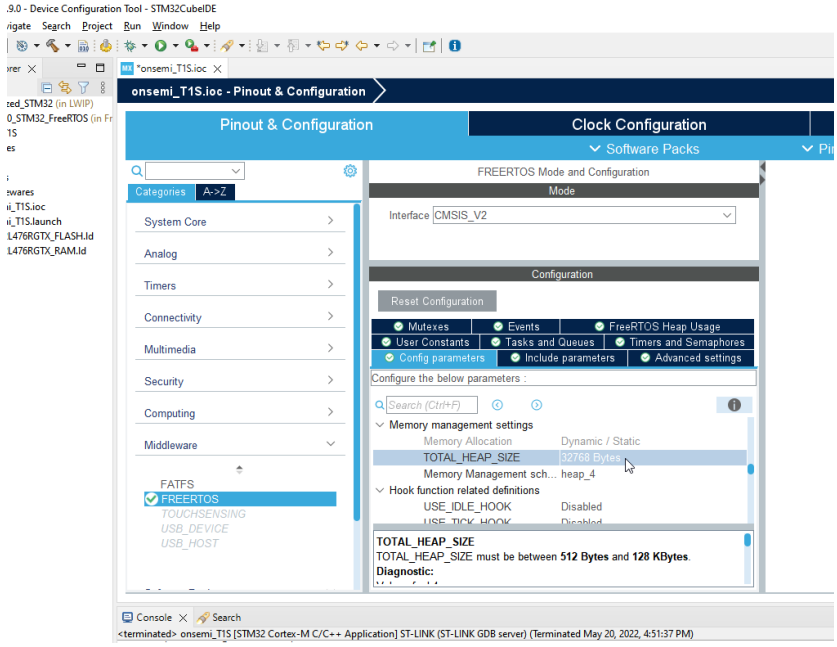
Setup FreeRTOS

If you selected the LwIP TCP/IP stack, you can skip this section.

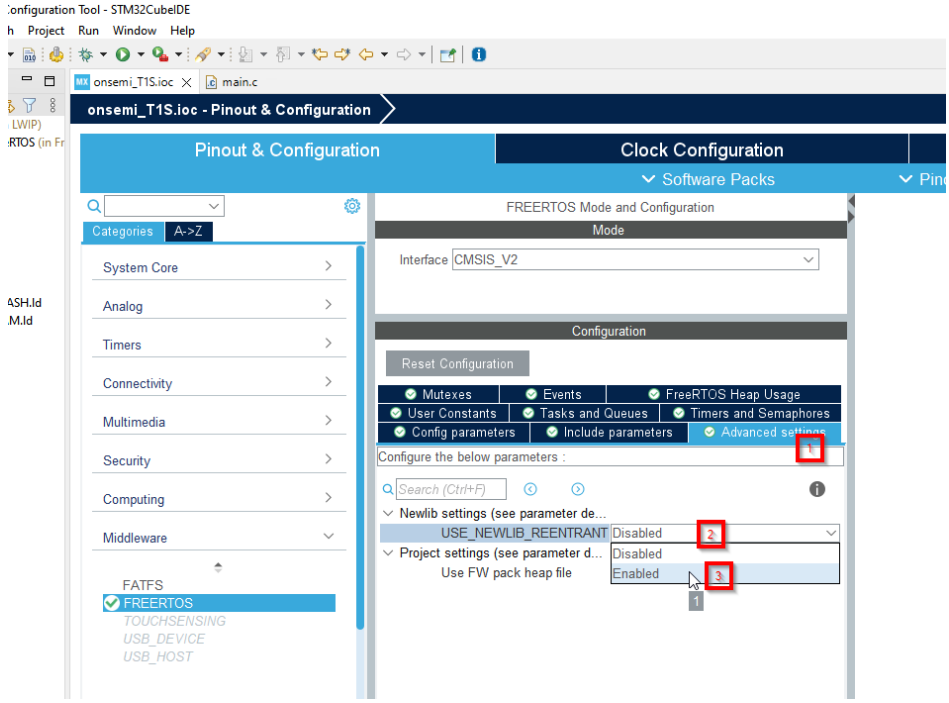
1. Enable FreeRTOS under “Middleware” “FreeRTOS.” Then select “CMSIS_V2.”



2. In the FreeRTOS settings under “Config parameters”, adjust the heap size. We suggest 32K bytes.

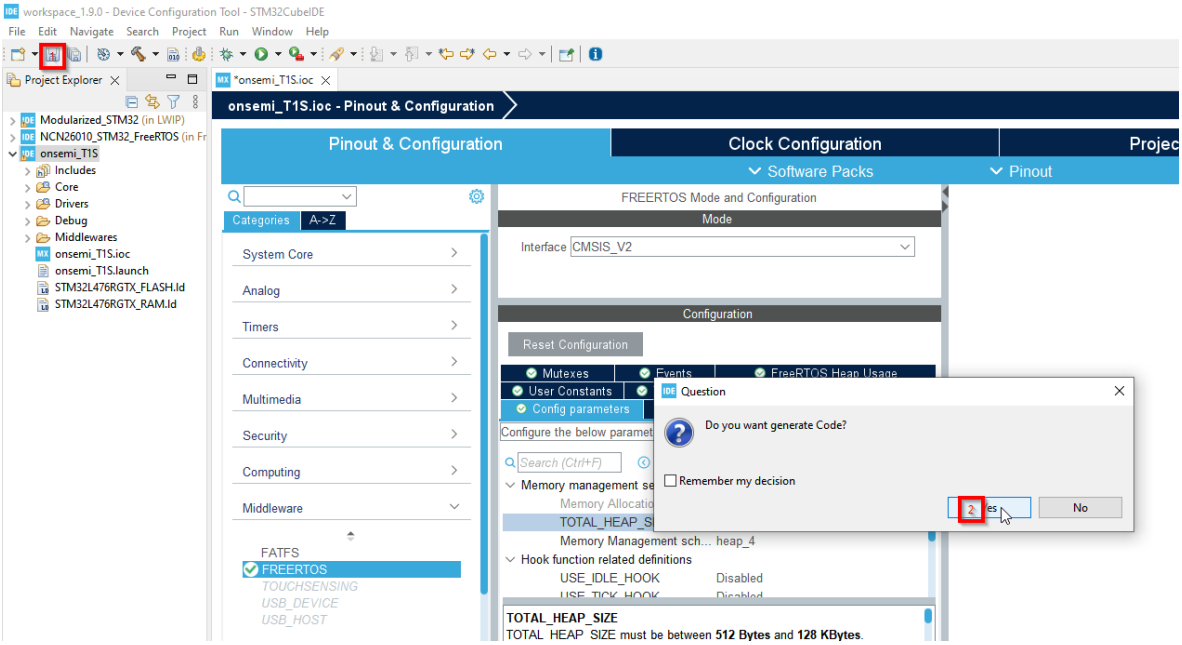


3. Under advanced settings Enable the FreeRTOS Reentrant mode.

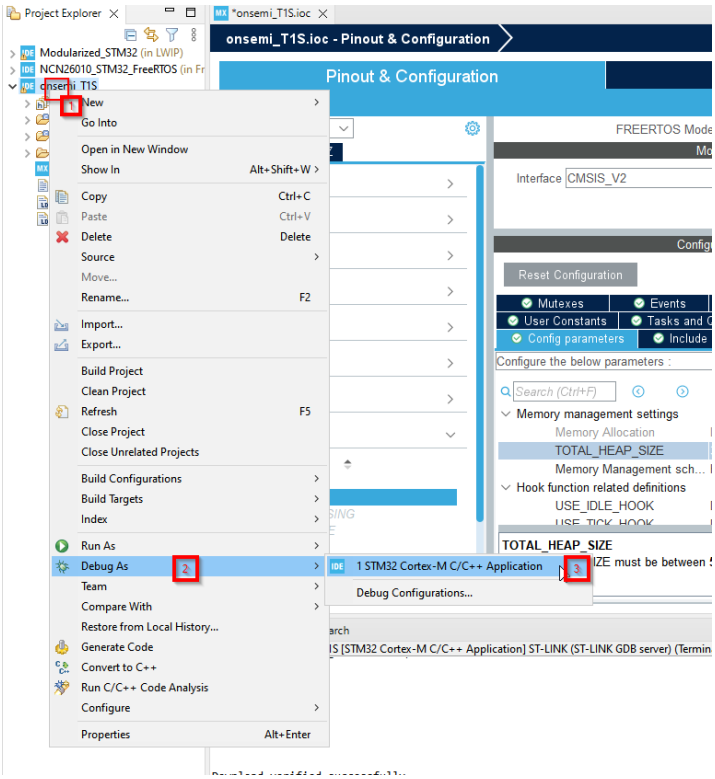


Flash and/or Debug the Example

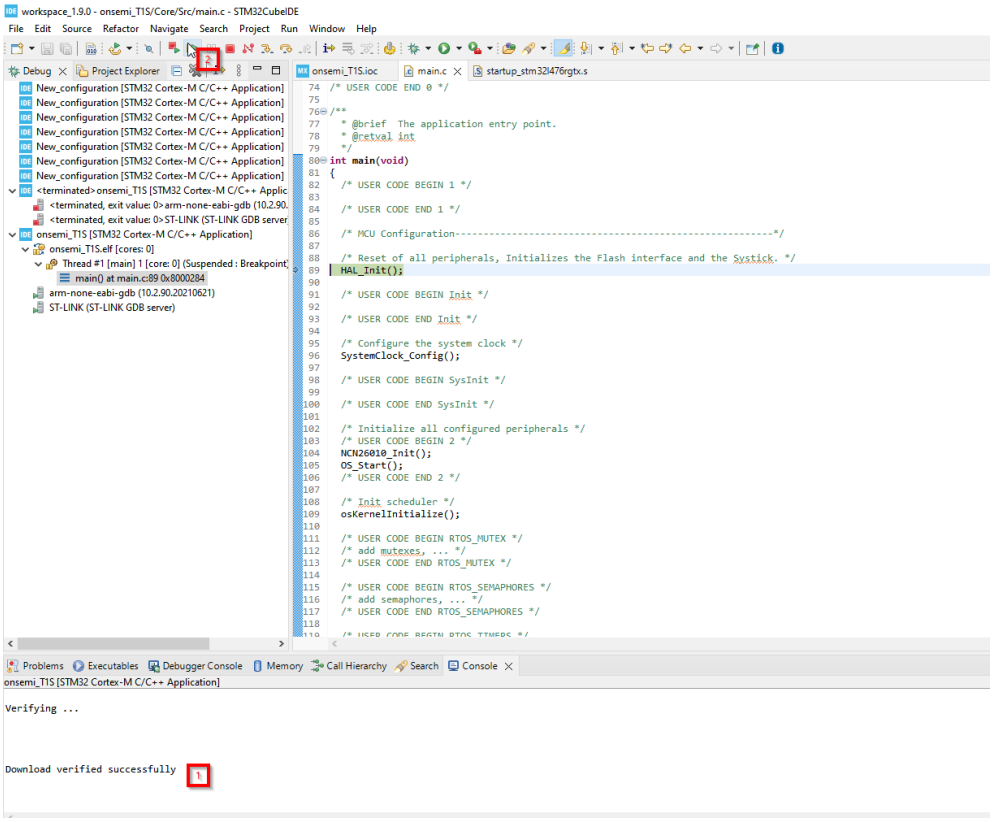
1. Save the Configuration and have the STM32CubeIDE generate code.



2. Ensure that your STM32L4xx evaluation board is connected to your computer, then Right click the project, then select debug as, then select STM32 Cortex-M C/C++ Application


Project Explorer
*onsemi_T1S.ioc
onsemi_T1S.ioc - Pinout & Configuration
Pinout & Configuration
Modularized_STM32 (in LWIP)
NCN26010_STM32_FreeRTOS (in Fr
onsemi_T1S
New
Go Into
Open in New Window
Show In Alt+Shift+W
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Source
Move...
Rename... F2
Import...
Export...
Build Project
Clean Project
Refresh F5
Close Project
Close Unrelated Projects
Build Configurations
Build Targets
Index
Run As
Debug As
Team
Compare With
Restore from Local History...
Generate Code
Convert to C++
Run C/C++ Code Analysis
Configure
Properties Alt+Enter
1 STM32 Cortex-M C/C++ Application
Debug Configurations...
FREERTOS Mode
Interface CMSIS_V2
Reset Configuration
Mutexes
Events
User Constants
Config parameters
Include
Configure the below parameters :
Search (Ctrl+F)
Memory management settings
Memory Allocation
TOTAL_HEAP_SIZE
Memory Management sch...
Hook function related definitions
USE_IDLE_HOOK
TOTAL_HEAP_SIZE

3. Wait for the build to complete and the hex to download to the board, then click play.



RSL10

Install Toolchain for Firmware Development

1. Register and Log in to **MyON** account here.
2. Download and install "**ON Semiconductor IDE**".
3. Download and install base "**RSL10 CMSIS Pack**".
    a. To install the CMSIS pack open CMSIS Pack Manager. Click on the icon "Open Perspective" as shown below and select "CMSIS Pack Manger".

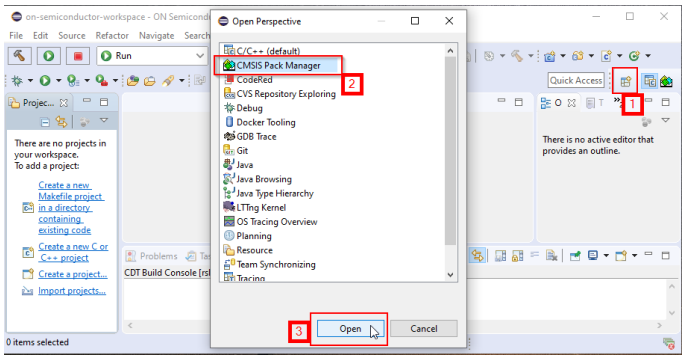


Figure 4. Open CMSIS Pack Manager

    b. Click on the icon "Import Existing Packs…" to import the CMSIS pack downloaded earlier. Select "Open" to install the pack.

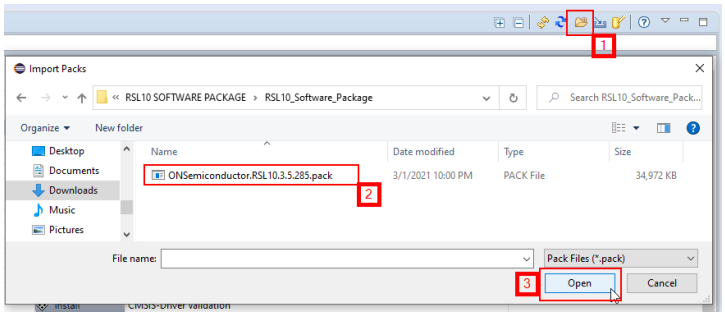


Figure 5. Import Existing CMSIS Pack

4. Download and install latest version of "**ARM CMSIS Pack**" as per step 3.
5. Download and install latest version of "**ARM CMSIS-FreeRTOS Pack**" as per step 3.
6. Download and install latest version of "**onsemi.RSL10T1S Pack**"as per step 3.
7. Once all the packs are installed, this is what CMSIS pack perspective should look like.

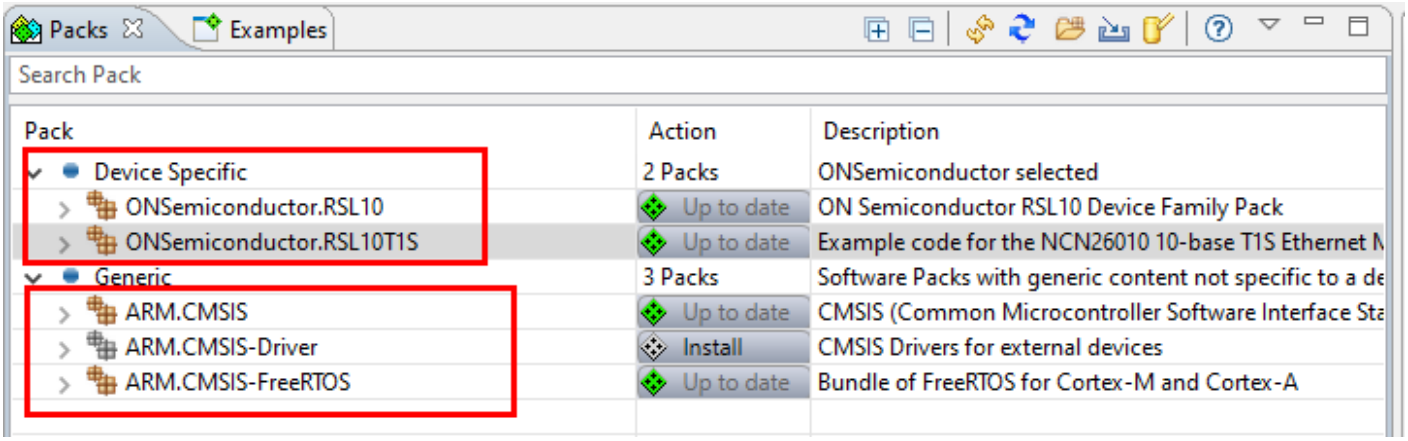


Figure 6. Installed CMSIS Packs

Import and Compile an Example Project from CMSIS Pack

1. Select RSL10T1S pack from CMSIS pack manager. Open the drop-down for "Examples". Right-click on the preferred example project. Select "Copy". This will import a copy of that example project into the workspace.

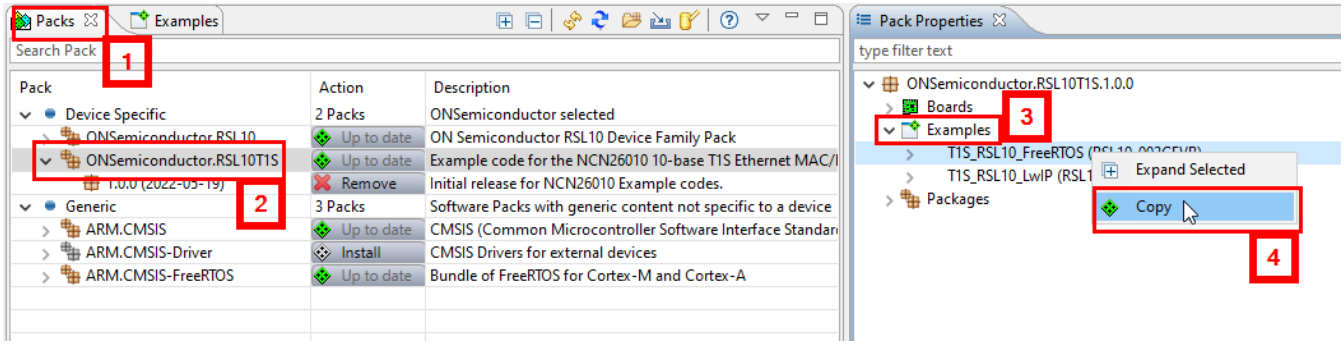


Figure 7. Copy Example Project

2. Right-click on the project name in Project Explorer and select "Build Configurations" to "Release" or "Debug". Next, select "Build Project" to compile the project.

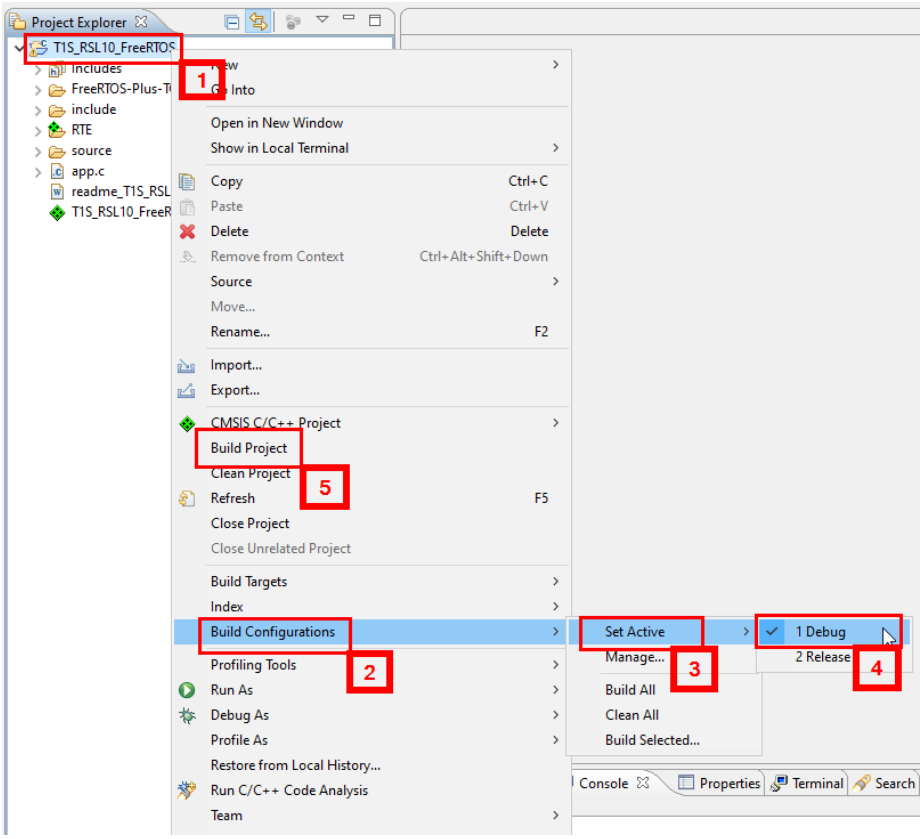


Figure 8. Set Build Configuration for a Project

Flash and/or Debug the Example

1. Create Run/Debug configuration for the project. Select “Run” -> “Run Configurations…”.

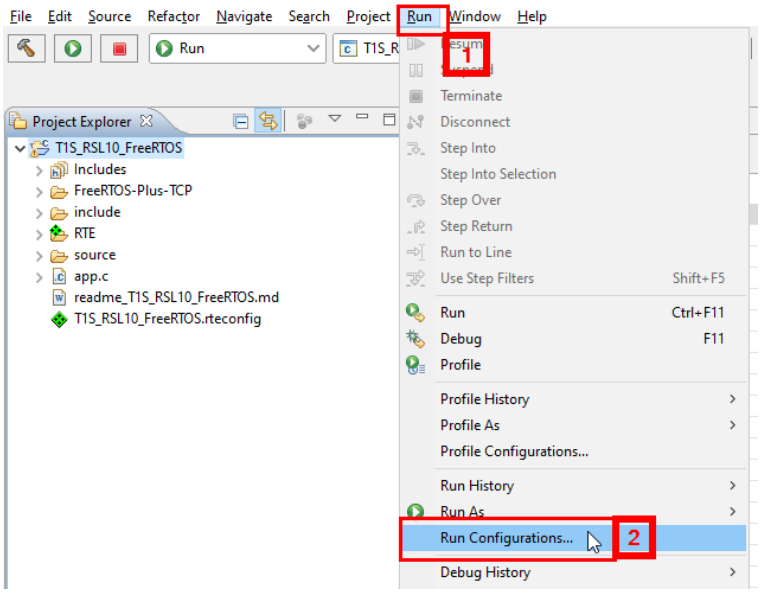


Figure 9. Open Run Configurations

2. Double-click on “GDB SEGGER J-Link Debugging”. This will generate a default configuration. Under “Main” tab, select the Project. Provide the .elf file path at “C/C++ Application”. Depending on the build

configuration it can be “Release/<project_name.elf>” or “Debug/<project_name.elf>”. Or add a variable to select automatically, “${config_name:${project_name}}/<project_name.elf>”. Select “Apply”.

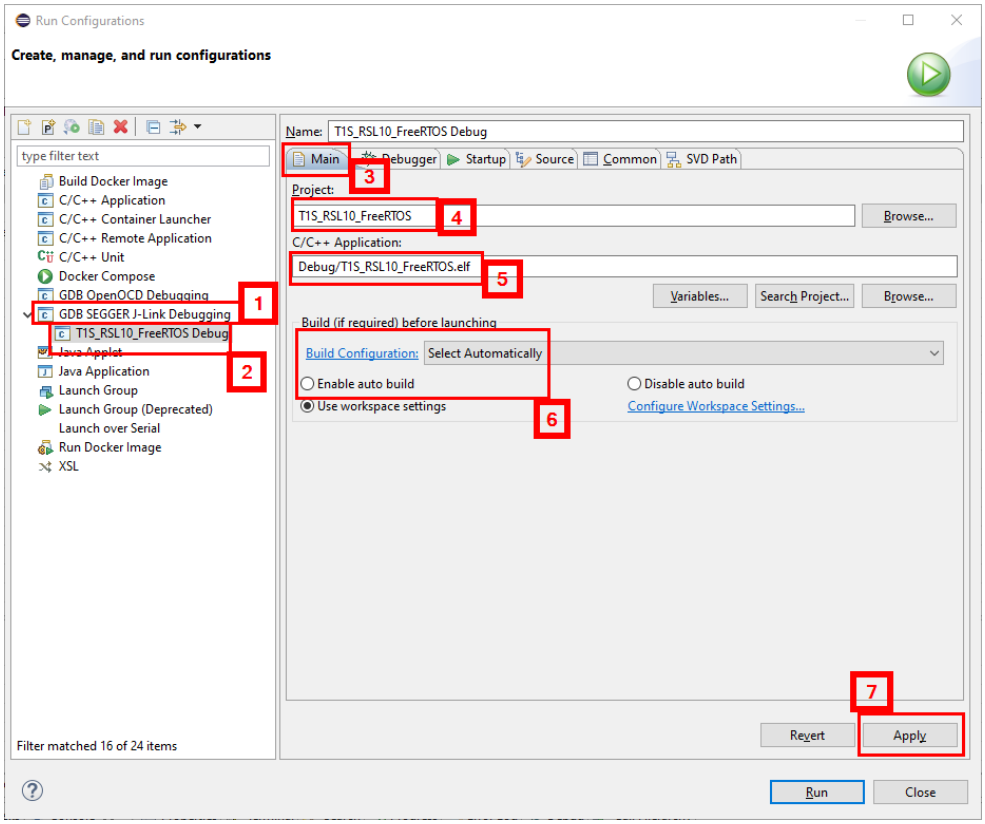


Figure 10. Create Run Configuration

3. Under “Debugger” tab, Update “Device Name” to RSL10. Select “Apply” and “Close”.

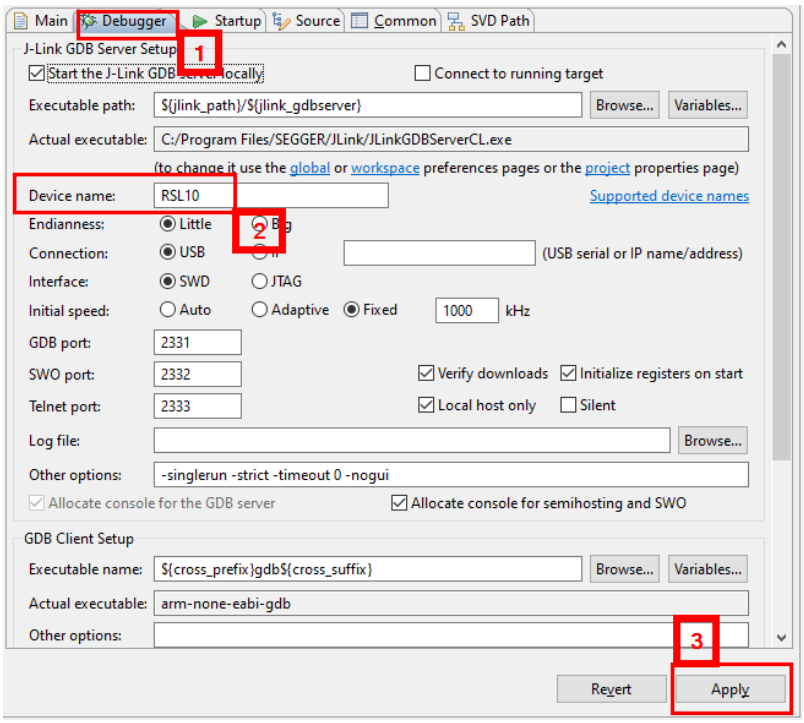


Figure 11. Add Device name

4. To flash the board, click green play button looking icon in “Run” mode.

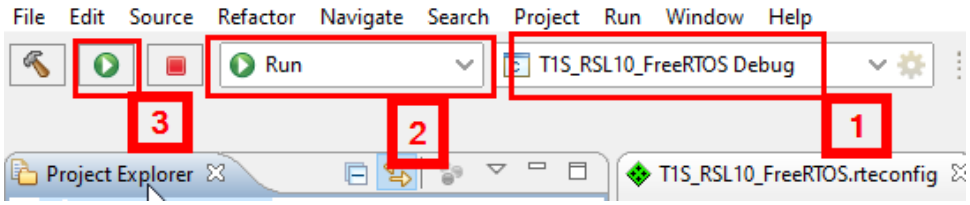


Figure 12. Run the configuration to Flash

5. To debug, click green bug looking icon in “Debug” mode.

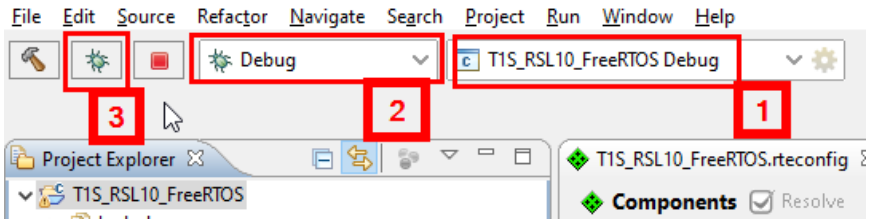


Figure 13. Start Debug configuration

6. Detailed instructions about using the IDE can be found in RSL10 Getting Started Guide.

Unimplemented Features

This example software is provided as a basic example to support designers to use the NCN26010 in their system. There are several optimizations and additional features that onsemi has identified as potential improvements to this software. onsemi may implement some of these features and will use feedback from customers to determine how to prioritize features.

Zero Copy Data Transfer

As currently implemented, the example implementations copy data to and from buffers when transferring from the NCN26010 over the SPI. With some pointer manipulation we could eliminate these copies which saves memory and eliminates the CPU time required to copy the data. Implementing this optimization may reduce system power requirements and may allow for a less expensive MCU to be used.

Runtime Configuration

As currently implemented, all the configuration options are selected at compile time by adjusting macro constants in the ncn26010.h header file. If the configuration needs to be changed after compile time this must be done using the remote configuration options described in ncn26010.h. An option to read a file or some piece of memory to decide how to configure the NCN26010 when initializing would simplify software deployment on multiple nodes.

Serial Communication for Command Line Interface

As currently implemented, there are empty macros defined for printing debug and other information to a console. Filling in these macros with functions that send this console information over a serial interface to a host computer would help simplify debugging. It would also allow for a command line interface to configure and communicate with the MCU.

Software Timestamping

The NCN26010 does not provide timestamps for incoming frames. Certain time sensitive messages require an accurate understanding of when the message arrived. The NCN26010 can be configured to signal on its DIO pins when it detects a start of frame delimiter. Triggering an interrupt on this signal and recording a timestamp would allow a software algorithm to pair together messages with their associated timestamp.

Documentation

The software package is documented in the source using Doxygen style comments. The generated documentation is distributed as part of the software package.