Board Bring up with CMSIS-Drivers

By Trevor Martin

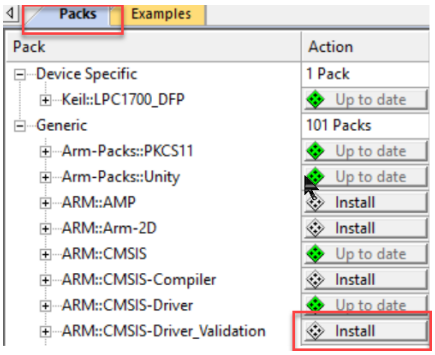
This Tech Tip assumes a moderate level of experience.

Introduction

In most embedded projects we are designing custom firmware and hardware. At the beginning of a project there is a phase of board bring up where the new hardware is tested, and each peripheral is proven to work correctly. This tech tip looks at developing a universal board bring up test suite based on the CMSIS-Drivers and the CMSIS-Driver validation pack. The key advantage of this approach is that most of the code already exists and if you make any extension to the existing tests then this work can be used on many future projects.

The CMSIS driver specification provides a standard API for many common communication peripherals. Alongside the driver specification, Arm also provides a test suite that can be used to validate each driver before it is released for general use. Since the validation test suite is available as a software pack for general use, we can use it to both validate the CMSIS drivers and test any new hardware. It's really important to realise that this test framework can be used on any microcontroller that has a range of CMSIS Drivers. Therefore, time spent getting this working can be reused on multiple future projects. It is also possible to extend the test framework with your own custom tests and reuse these with little or no extra work. Here, we will look at configuring a CMSIS Driver, adding the validation suite and how to configure and run the tests.

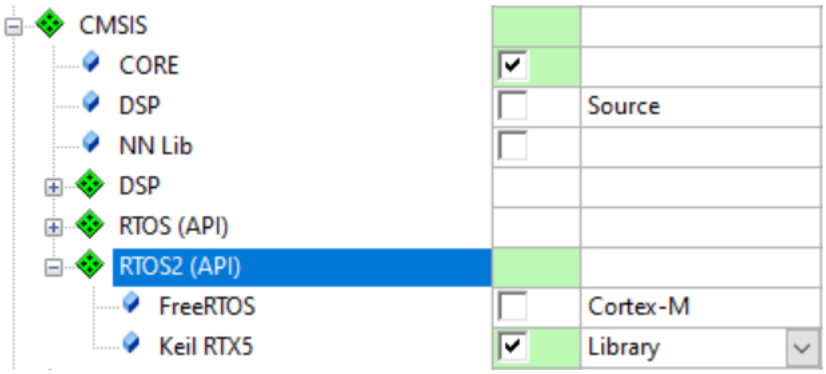
Start the pack installer and install the CMSIS Driver Validation pack.



Create a minimal project for your target microcontroller.

This should just consist of the startup code and have the Instrumented Trace Macrocell (ITM), or event viewer configured. This provides a STDIO channel back to the debugger that the validation suite can use to print the test report.

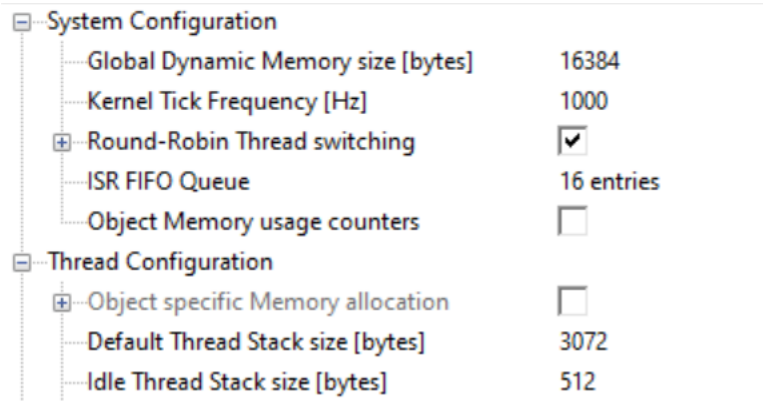
Add the CMSIS::RTOS::RTX RTOS from within the Run Time Environment (RTE).



Configure the RTX memory requirements for the validation tests.

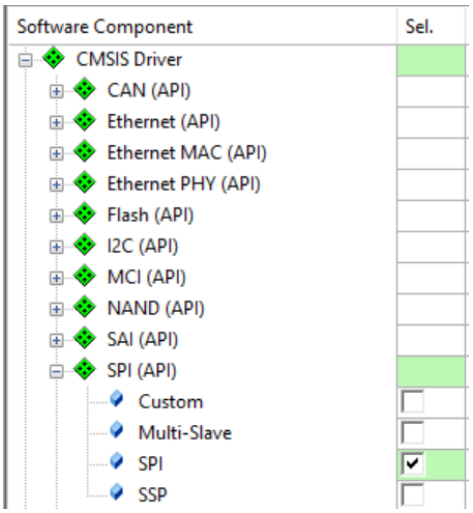
In startup.s configure the heap to have 16384 bytes of memory

In the rtx_config.h set the global memory pool to have 16384 bytes and the thread stack size to 3072 bytes.



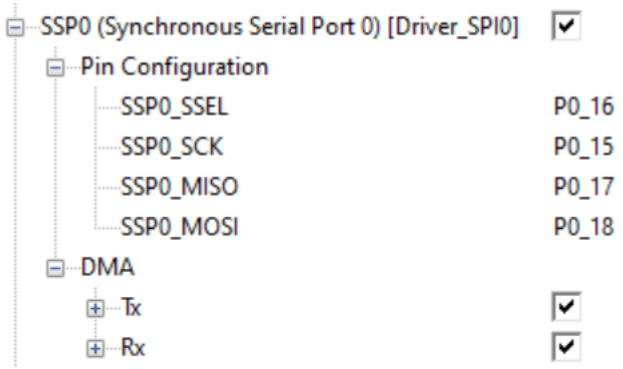
Add the required CMSIS drivers from within the RTE.

In this example we will add a CMSIS-SPI driver as shown below

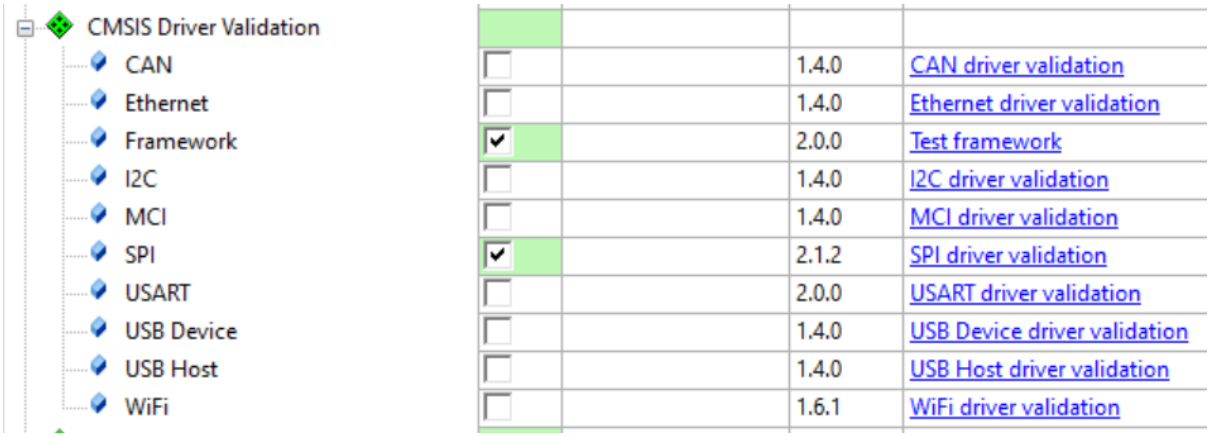


Configure the SPI interface pins in the RTE_conf.h file.

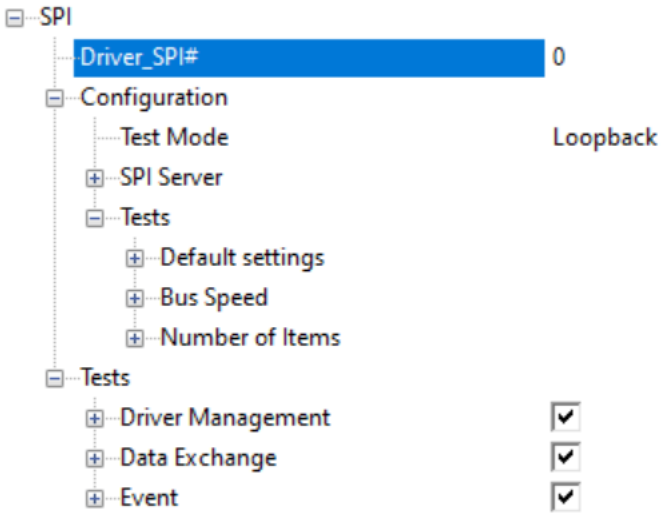
This RTE_Device.h file allows you to configure the low-level pin multiplex. Some manufacturers provide a separate tool which is used to configure the pins and autogenerate a basic project.



Add the CMSIS Validation framework and the test suite for the SPI driver.



Open DV_SPI_Conf.h this is a configuration wizard which allows you to define the SPI peripherals default test settings and the range of tests to be carried out.



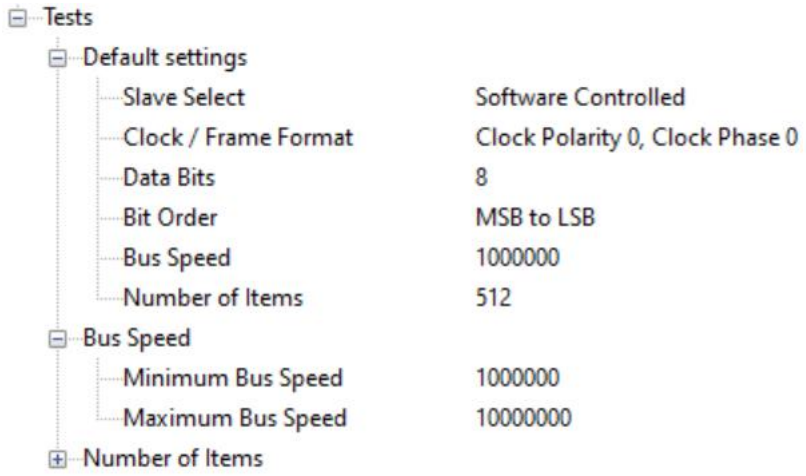
Select loopback mode.

Loopback mode is a good starting point as you can simply join the MISO and MOSI pins together. The driver validation pack also includes a project for a set of test servers. These can be found in the validation pack directory. When added the server will prove communication between two physical devices.

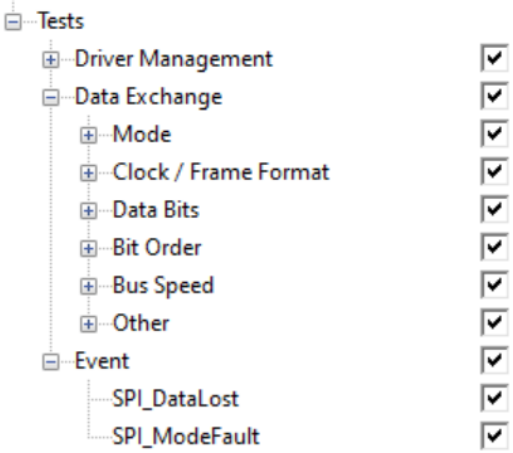
C:\Keil_v5\PACKS\ARM\CMSIS-Driver_Validation\<version>\Tools

Configure the default test driver settings in DV_SPI_Conf.h

These are found in the SPI\Configuration\Tests section and allow you to set the default communication parameters and the size of the test sets.



Select the data exchange tests in the SPI\test section.



These include basic test for the driver as well as data exchange and error event tests.

To enable the test suite, add the following code to main.

```
#include "cmsis_dv.h”
```

In main() start the driver validation test thread as follows

```
osThreadNew(cmsis_dv, NULL, NULL);
```

and then add the necessary code to start the RTOS as normal.

Build the project.

Start the debugger and run the project.

This will execute the validation tests and provide a report in the ITM Console window.

Example Project

An example project is available on request.

Extending the Validation tests

While the driver validation test suite supports all drivers not all are tested in loopback mode. It is possible to extend the driver validation test suite by adding your own custom tests. While this is outside the scope of this document do get in touch using the email below and we can send you further examples that show how to do this.