Renode Example Guide

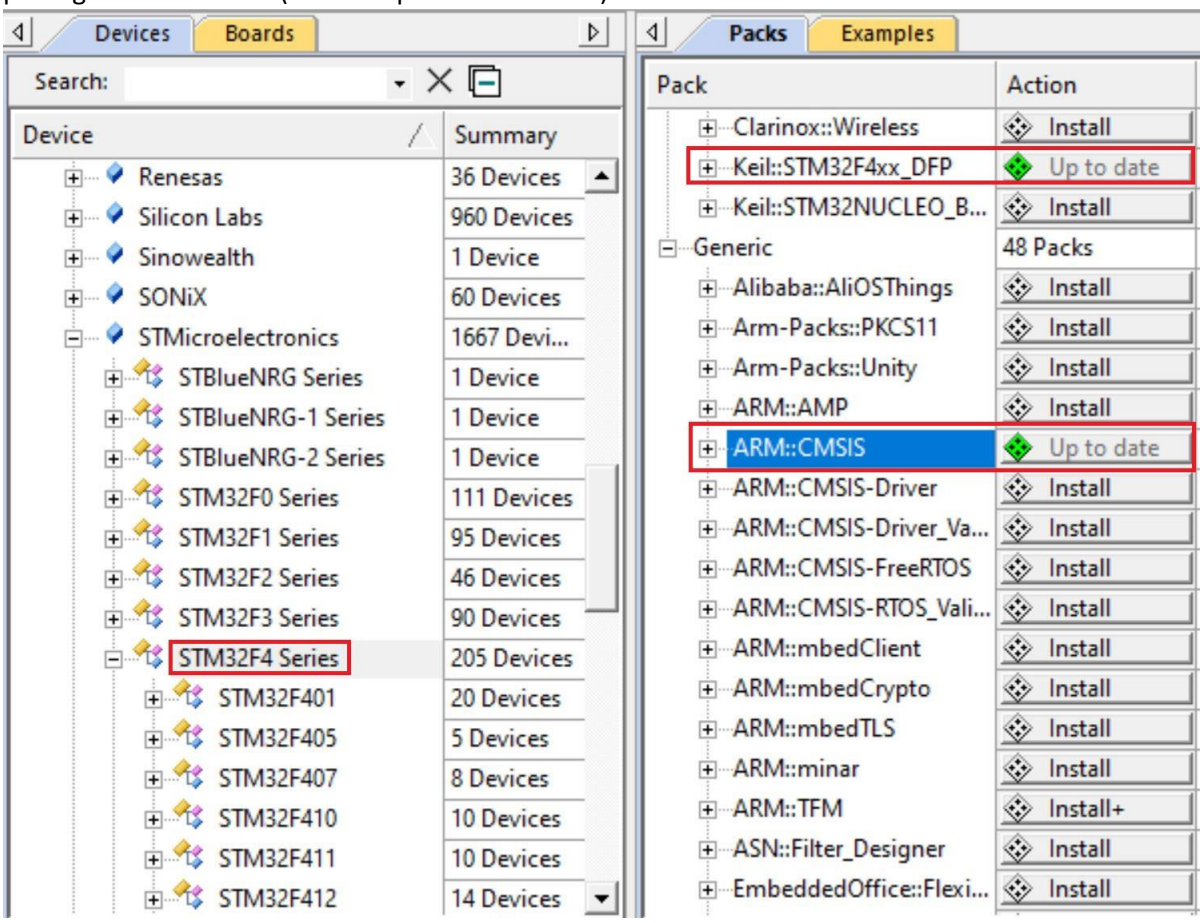
Introduction

This guide introduces how to emulate applications on Renode, the following platform is used
Renode platform: STM32F4_discovery kit
Microcontroller: STM32F407xx
CPU: Cortex® -M4 with FPU core

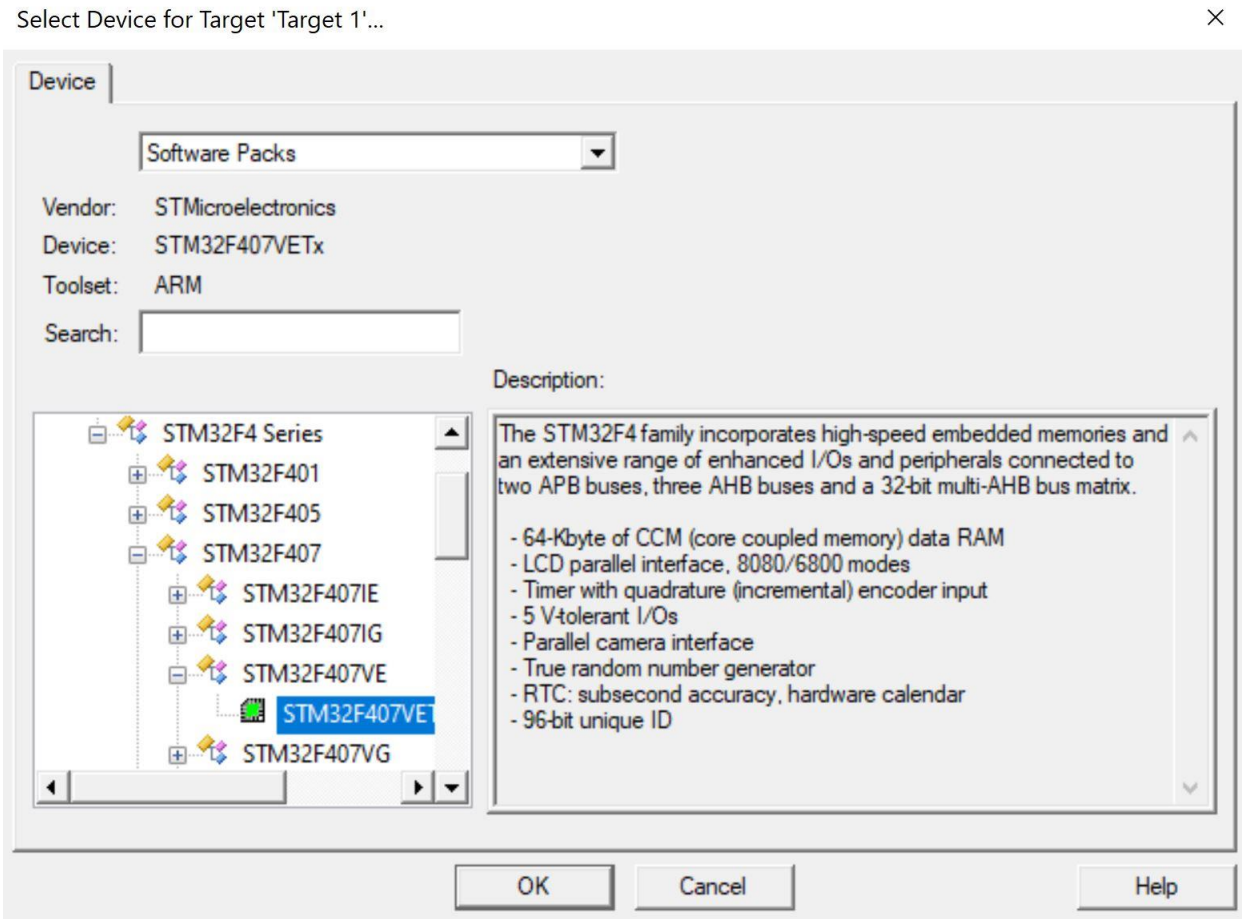
The guide starts with describing how to setup the development environment of the application on Keil. Then it describes how to run the application over Renode.

Development Environment

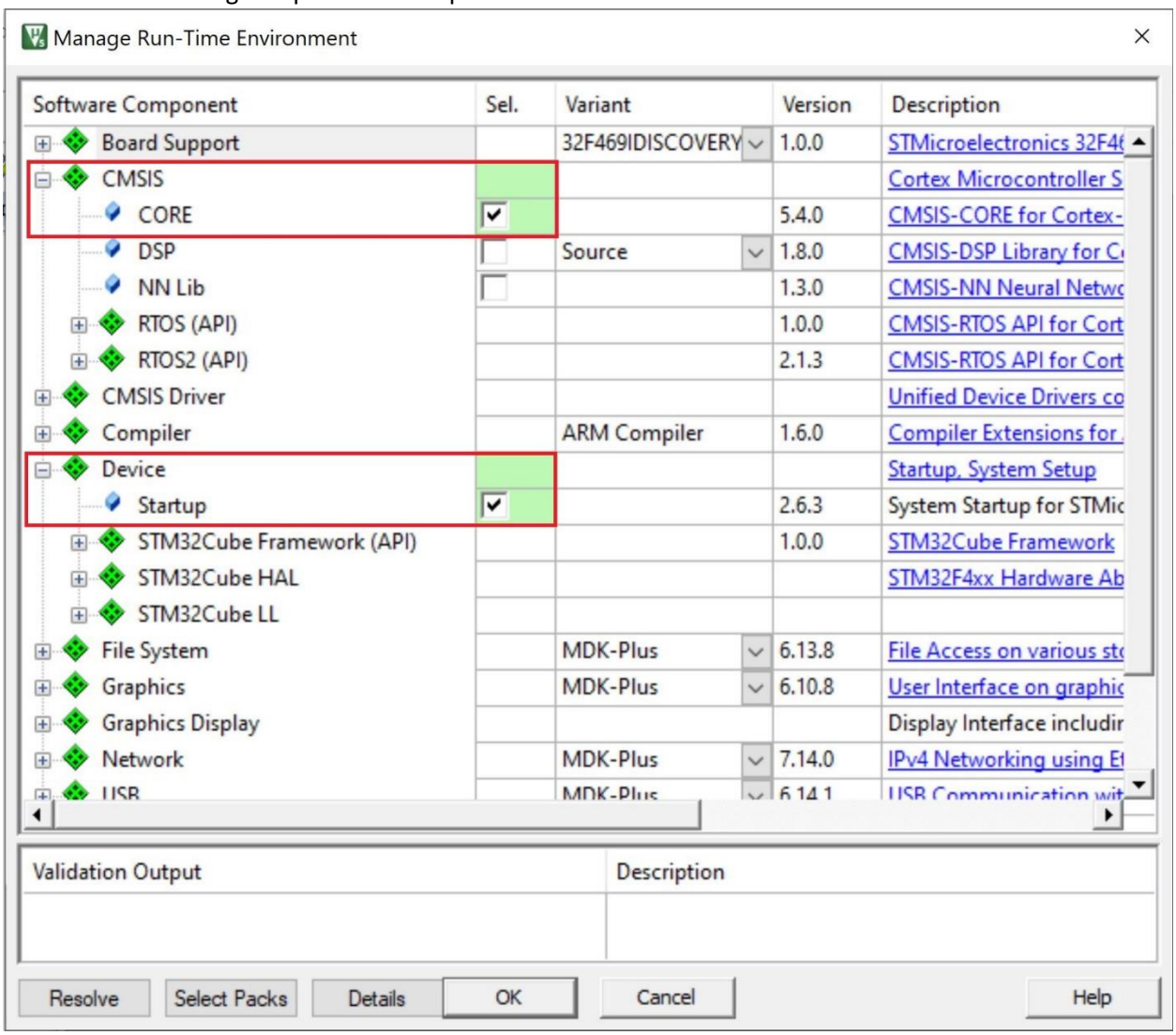
1. Open Keil, then open package manager and make sure that STM32F4 series and CMSIS core packages are installed. (Same steps as done in Lab)



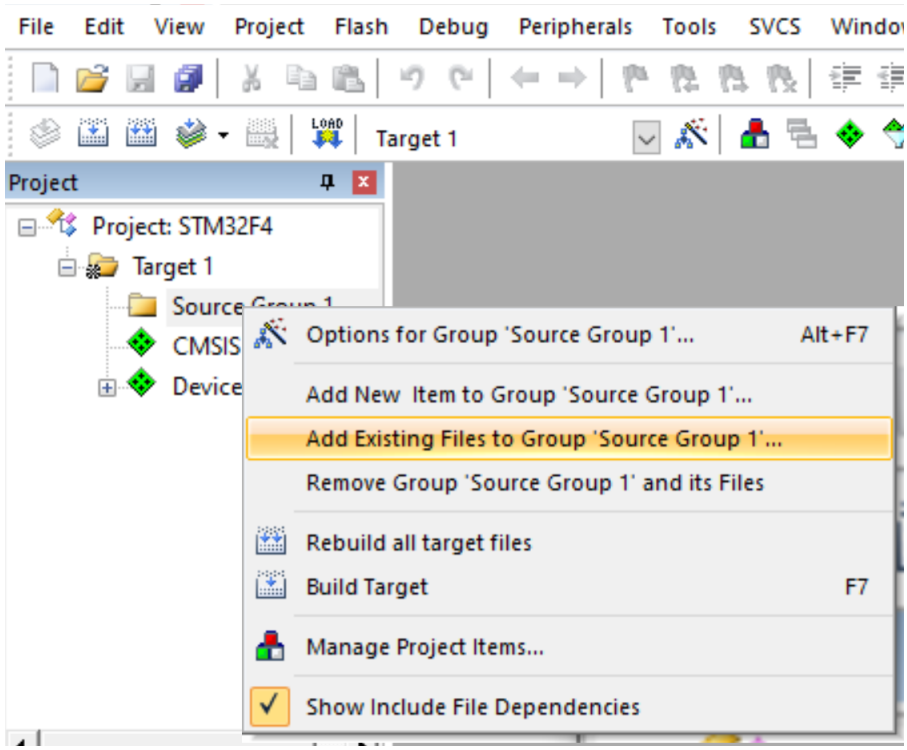
2. Create new project and select device as shown then press OK.



3. Choose the following components then press OK



4. Choose to add existing item to project as below and add main.c file



5. Build target. No errors should be shown and application binary < Keil_Project_Name >.axf should be generated in < Keil_Project_Path >\Objects folder

Now the development environment is ready, next step is to run the application using Renode.

Running Applications on Renode

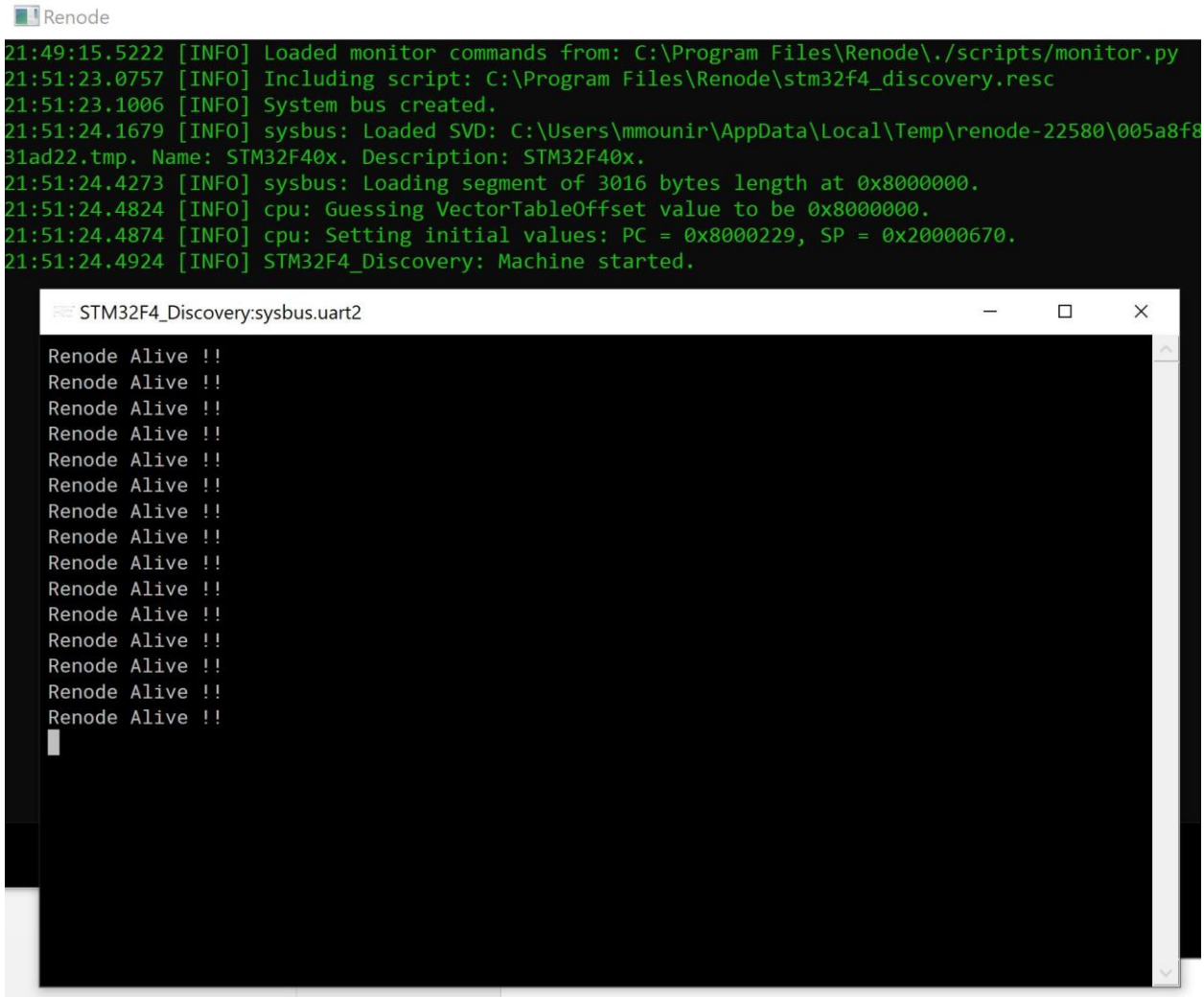
1. After installing Renode, open single node scripts folder which exists in <Renode_Installation_Path>\scripts\single-node
2. Open stm32f4_discovery.resc using any text editor
3. Change the value of $bin to the path of the application binary as shown below (the path of the application binary is <Keil_Project_Path>\Objects\<Keil_Project_Name>.axf)
4. Make sure showAnalyzer command uses sysbus.uart2

```
:name: STM32F4 Discovery
:description: This script runs Contiki on STM32F4 Discovery.

using sysbus
$name?="STM32F4_Discovery"
mach create $name
machine LoadPlatformDescription @platforms/boards/stm32f4_discovery-kit.repl

cpu PerformanceInMips 125

$bin?=@C:/STM32F4/Objects/STM32F4.axf

showAnalyzer sysbus.uart2

### Set random board UNIQUE ID ###

python "import _random"
python "rand = _random.Random()"

$id1 = `python "print rand.getrandbits(32)"`
$id2 = `python "print rand.getrandbits(32)"`
$id3 = `python "print rand.getrandbits(32)"`
macro reset
"""
    sysbus LoadELF $bin

    sysbus WriteDoubleWord 0x1FFF7A10 $id1
    sysbus WriteDoubleWord 0x1FFF7A14 $id2
    sysbus WriteDoubleWord 0x1FFF7A18 $id3
"""

runMacro $reset
```

5. Open Renode.exe

6. In Renode monitor shell execute stm32f4_discovery script by typing the command **s @scripts/single-node/stm32f4_discovery.resc**

7. The script will be executed, Renode will switch to machine shell and the UART analyzer window should be shown as below



Once the application run it will start sending “Renode Alive !!” messages on UART, if any keyboard key is pressed in the UART analyzer window it will pause/resume the application messages.

8. Stm32f4_discovery platform in Renode has some external Buttons attached, in this demo *UserButton* is used which is connected to GPIO Port A, to emulate pressing the button type the command: **sysbus.gpioPortA.UserButton Press** in machine shell

9. Pressing *UserButton* generates an interrupt in application and the following message is sent through UART to indicate that the button is pressed

```
STM32F4_Discovery:sysbus.
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Button is pressed !!
```

10. To release the button execute the command: **sysbus.gpioPortA.UserButton Release** in machine shell

```
(STM32F4_Discovery) sysbus.gpioPortA.UserButton Press
(STM32F4_Discovery) sysbus.gpioPortA.UserButton Release
(STM32F4_Discovery)
```

11. Releasing button also generates interrupt in application and the following message is sent through UART to indicate the button is released

```
STM32F4_Discovery:sysbus.uart2
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Button is pressed !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Button is released !!
```