

# Renode Example Guide

## Introduction

This guide introduces how to emulate applications on Renode, the following platform is used

Renode platform: STM32F4\_discovery kit

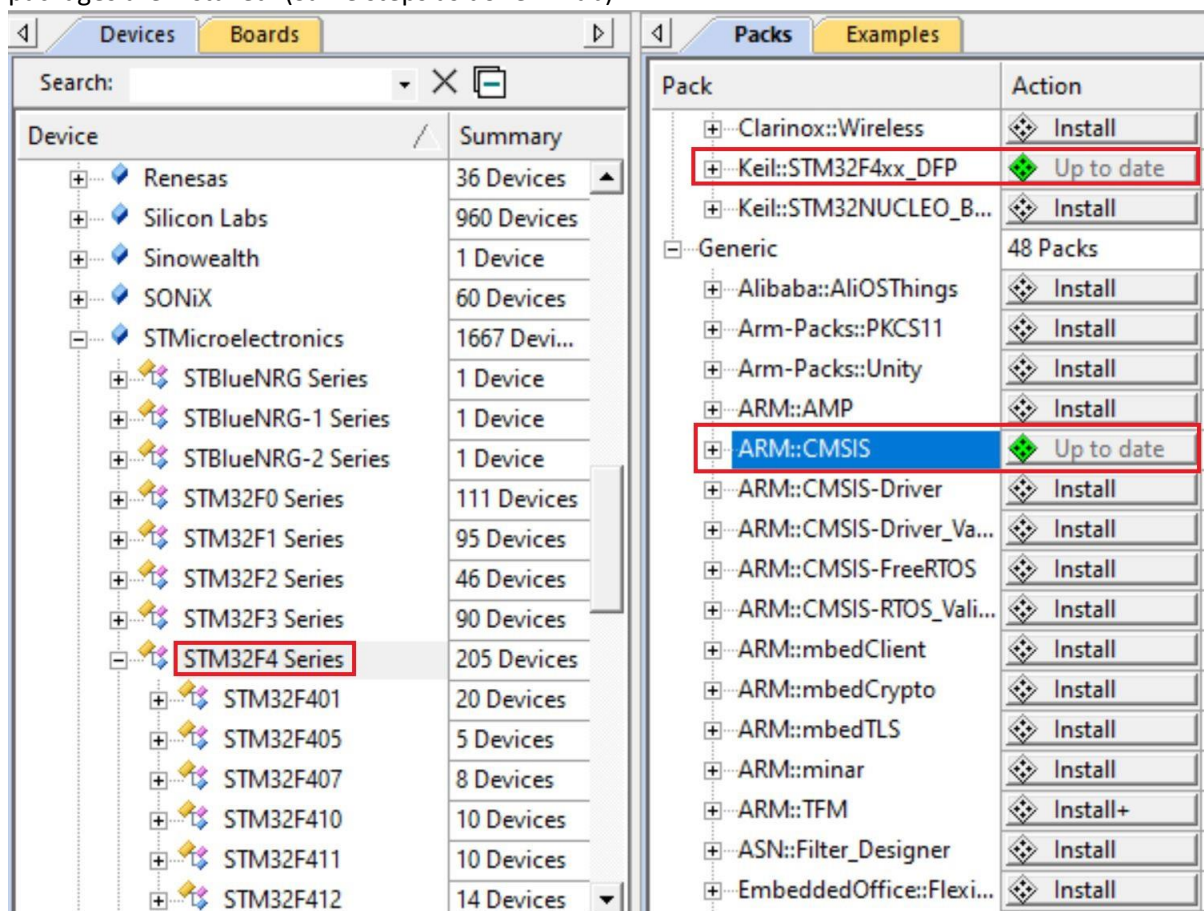
Microcontroller: STM32F407xx

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.

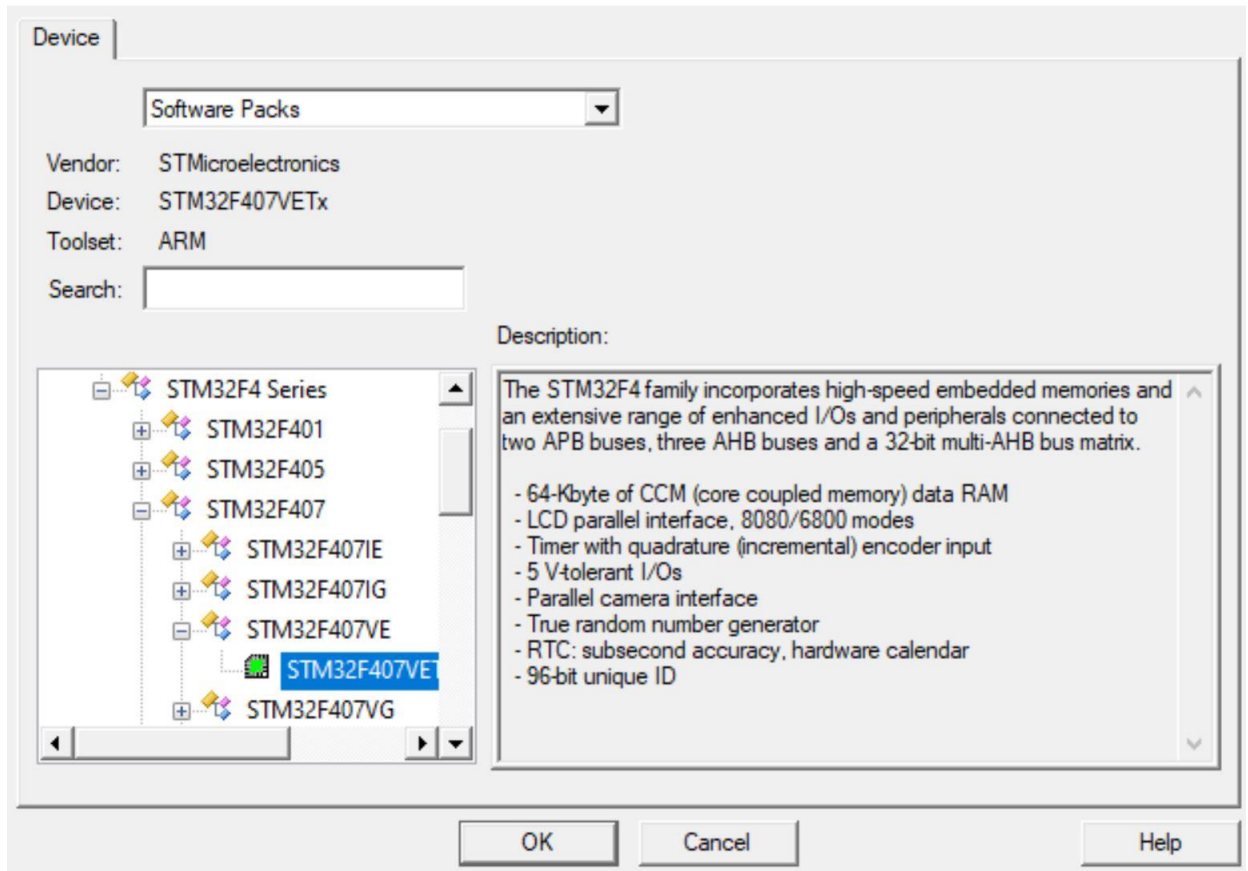
## Development Environment

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)

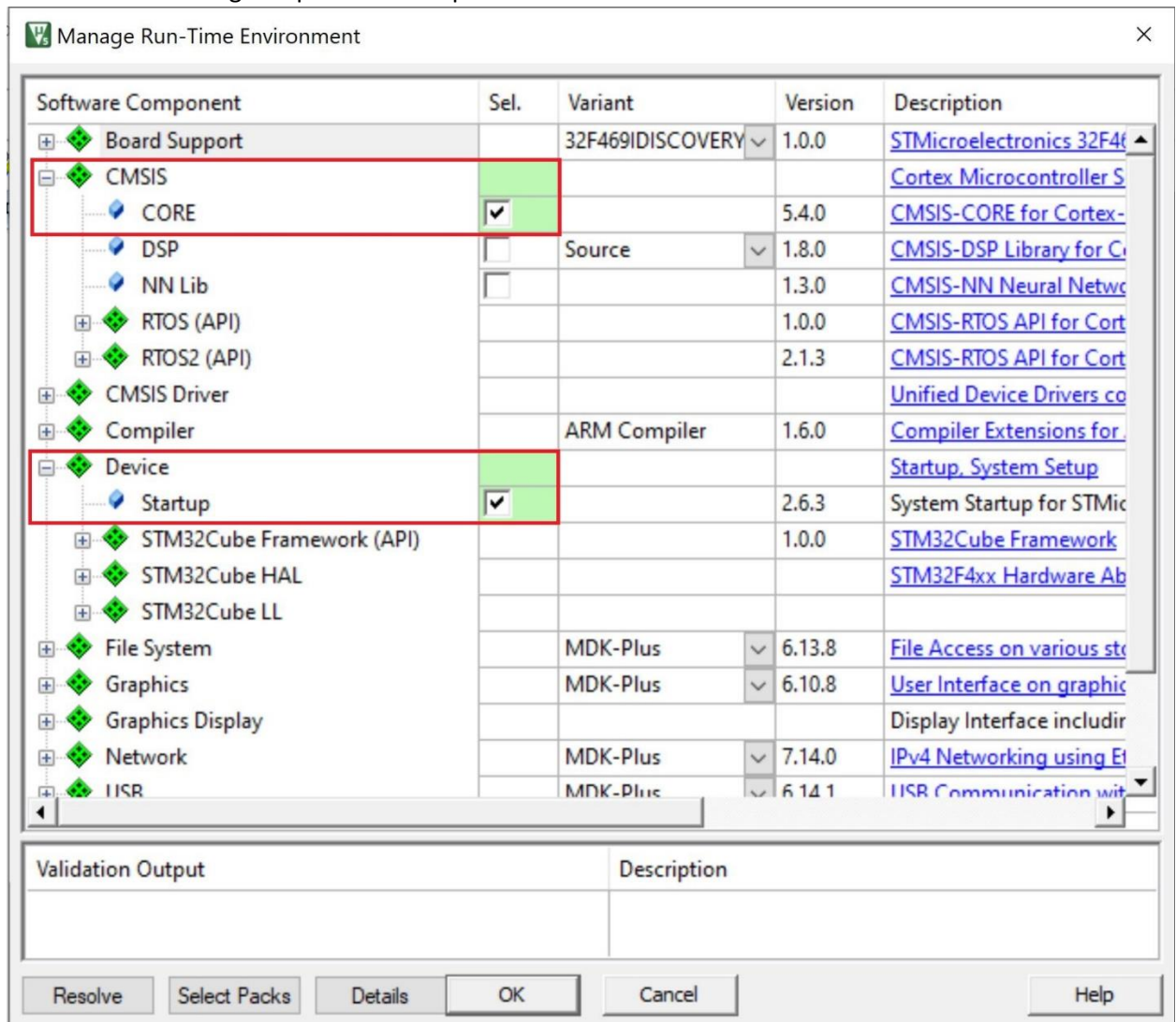


2. Create new project and select device as shown then press OK.

Select Device for Target 'Target 1'...



3. Choose the following components then press OK

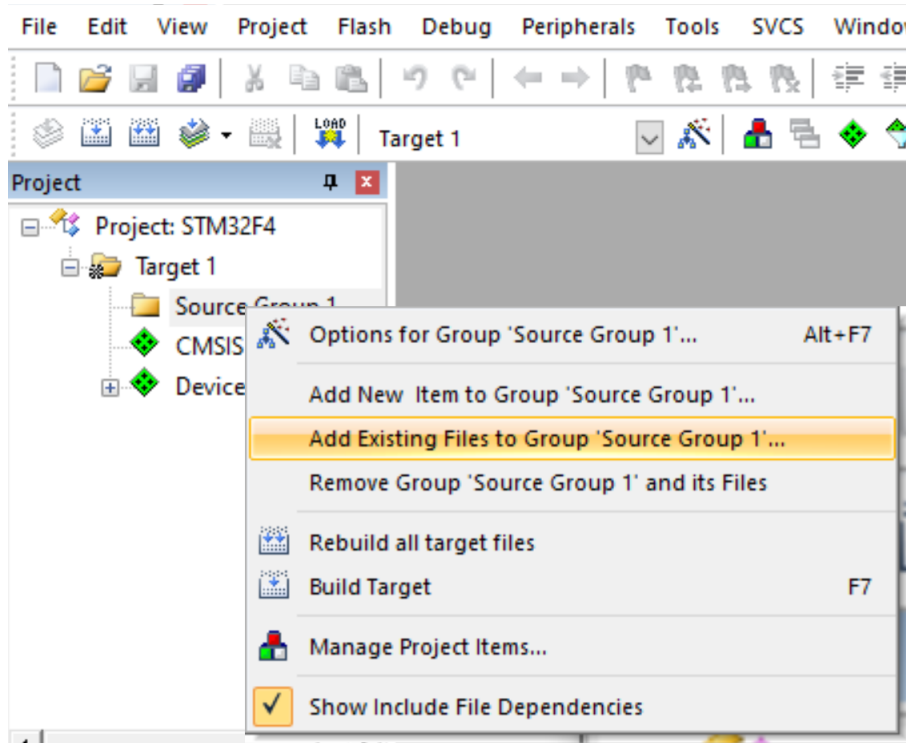
The image shows a 'Manage Run-Time Environment' dialog box. It features a tree view on the left with expandable categories like Board Support, CMSIS, RTOS, CMSIS Driver, Compiler, Device, File System, Graphics, Graphics Display, Network, and IISIR. The right side is a table with columns: Software Component, Sel., Variant, Version, and Description. Two rows are highlighted with red boxes: 'CMSIS' and 'Device'. In the 'Sel.' column, the 'CORE' sub-component under CMSIS and the 'Startup' sub-component under Device are checked. The bottom of the dialog has a 'Validation Output' and 'Description' section, and a row of buttons: Resolve, Select Packs, Details, OK, Cancel, and Help.

Software Component	Sel.	Variant	Version	Description
Board Support		32F469IDISCOVERY	1.0.0	STMicroelectronics 32F469
CMSIS	<input checked="" type="checkbox"/>			Cortex Microcontroller S
CORE	<input checked="" type="checkbox"/>		5.4.0	CMSIS-CORE for Cortex-
DSP	<input type="checkbox"/>	Source	1.8.0	CMSIS-DSP Library for C
NN Lib	<input type="checkbox"/>		1.3.0	CMSIS-NN Neural Netwo
RTOS (API)			1.0.0	CMSIS-RTOS API for Cort
RTOS2 (API)			2.1.3	CMSIS-RTOS API for Cort
CMSIS Driver				Unified Device Drivers co
Compiler		ARM Compiler	1.6.0	Compiler Extensions for
Device	<input checked="" type="checkbox"/>			Startup, System Setup
Startup	<input checked="" type="checkbox"/>		2.6.3	System Startup for STMic
STM32Cube Framework (API)			1.0.0	STM32Cube Framework
STM32Cube HAL				STM32F4xx Hardware Ab
STM32Cube LL				
File System		MDK-Plus	6.13.8	File Access on various st
Graphics		MDK-Plus	6.10.8	User Interface on graphic
Graphics Display				Display Interface includir
Network		MDK-Plus	7.14.0	IPv4 Networking using Et
IISIR		MDK-Plus	6.14.1	IISIR Communication wit

Validation Output	Description

Resolve   Select Packs   Details   **OK**   Cancel   Help

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()"

$!d1=`python "print rand.getrandbits(32)"`
$!d2=`python "print rand.getrandbits(32)"`
$!d3=`python "print rand.getrandbits(32)"`
macro reset
"""
... sysbus LoadELF $bin

... sysbus WriteDoubleWord 0x1FFF7A10 $!d1
... sysbus WriteDoubleWord 0x1FFF7A14 $!d2
... sysbus WriteDoubleWord 0x1FFF7A18 $!d3
"""

runMacro $reset
```

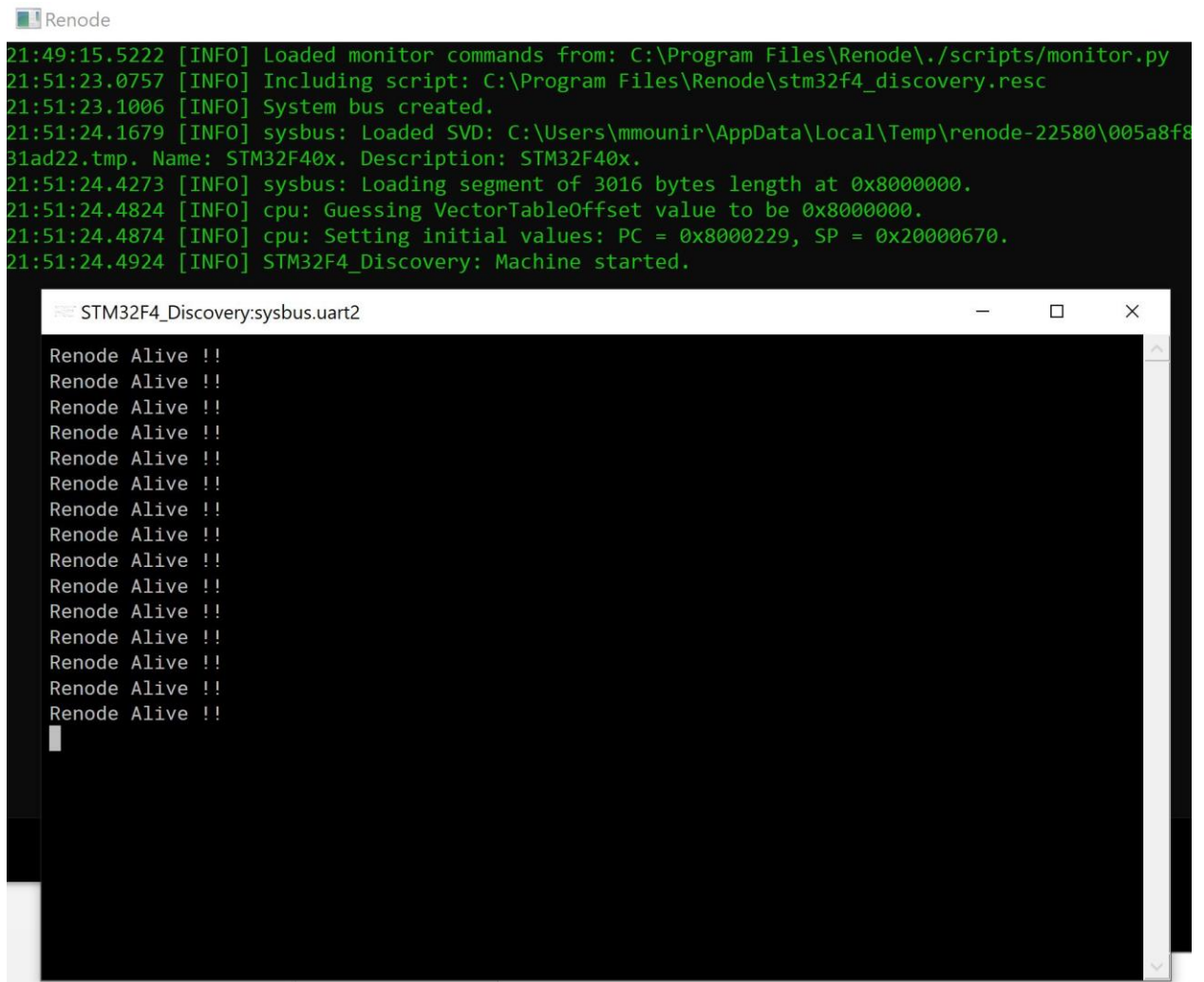
5. Open Renode.exe

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



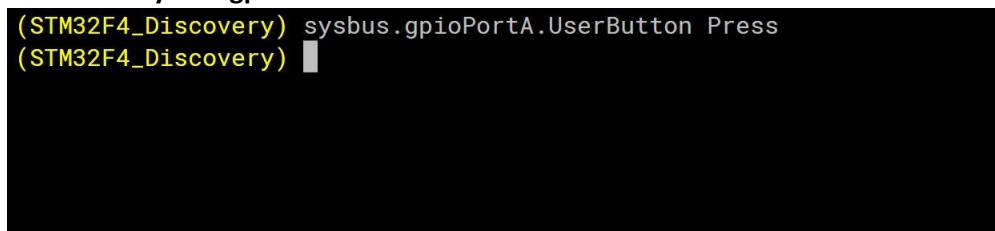
The screenshot shows the Renode application interface. The top pane displays a log of system initialization messages in green text on a black background. The bottom pane shows a UART analyzer window titled 'STM32F4\_Discovery:sysbus.uart2' with a black background and white text. The log messages include timestamps and information about loading monitor commands, including scripts, creating the system bus, loading SVD files, and setting initial values for the CPU. The UART analyzer window shows a series of 'Renode Alive !!' messages, indicating that the machine is alive and responding to the UART analyzer.

```
21:49:15.5222 [INFO] Loaded monitor commands from: C:\Program Files\Renode\./scripts/monitor.py
21:51:23.0757 [INFO] Including script: C:\Program Files\Renode\stm32f4_discovery.resc
21:51:23.1006 [INFO] System bus created.
21:51:24.1679 [INFO] sysbus: Loaded SVD: C:\Users\mmounir\AppData\Local\Temp\renode-22580\005a8f831ad22.tmp. Name: STM32F40x. Description: STM32F40x.
21:51:24.4273 [INFO] sysbus: Loading segment of 3016 bytes length at 0x8000000.
21:51:24.4824 [INFO] cpu: Guessing VectorTableOffset value to be 0x8000000.
21:51:24.4874 [INFO] cpu: Setting initial values: PC = 0x8000229, SP = 0x2000670.
21:51:24.4924 [INFO] STM32F4_Discovery: Machine started.
```

```
STM32F4_Discovery:sysbus.uart2
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
Renode Alive !!
```

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



The screenshot shows the machine shell in Renode. The prompt is '(STM32F4\_Discovery)'. The command 'sysbus.gpioPortA.UserButton Press' has been entered, and the cursor is at the end of the line, ready for the next command.

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



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.i
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 !!
█
```