Securing IoT Devices using Arm TrustZone®

Class 3: Creating your First TrustZone Application

November 28, 2018

Jacob Beningo

DesignNews

CEC CONTINUING EDUCATION CENTER

Presented by: Digi-Key ELECTRONICS

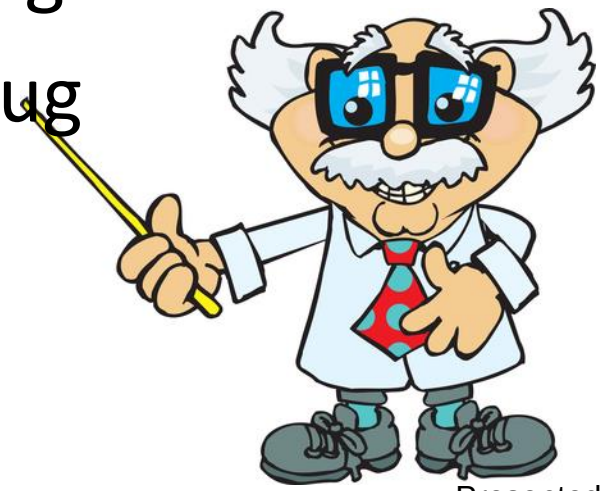
Course Overview

Topics:

- Understanding Embedded System Security
- Introduction to Arm TrustZone®
- **Creating your First TrustZone Application**
- Designing and Debugging a Secure Boot Solution
- Securing a RTOS Application with TrustZone

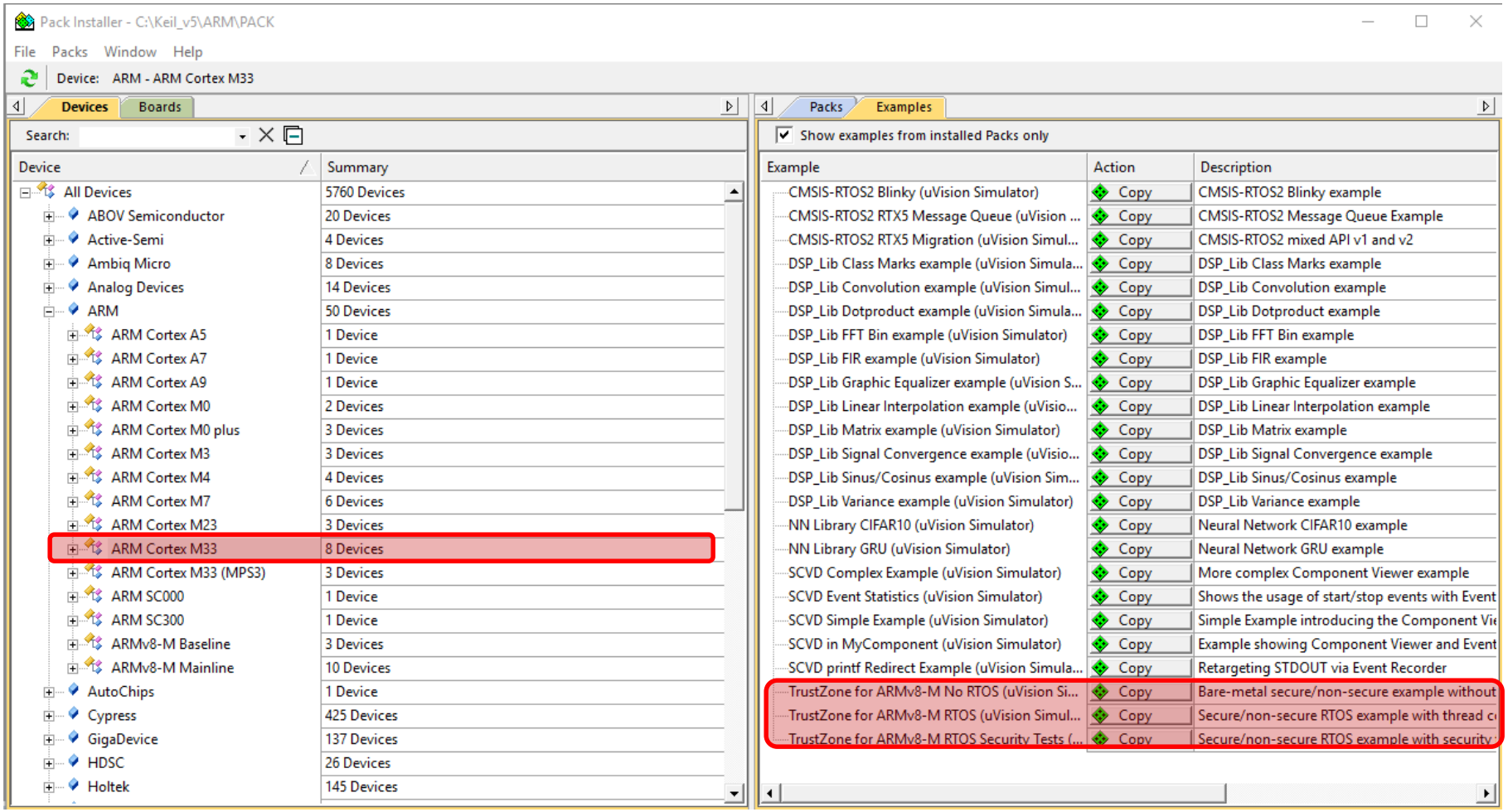
Session Overview

- Project Setup
- The TrustZone Workspace
- The SAU
- TrustZone Example Walk-through
- Setting up simulation and debug

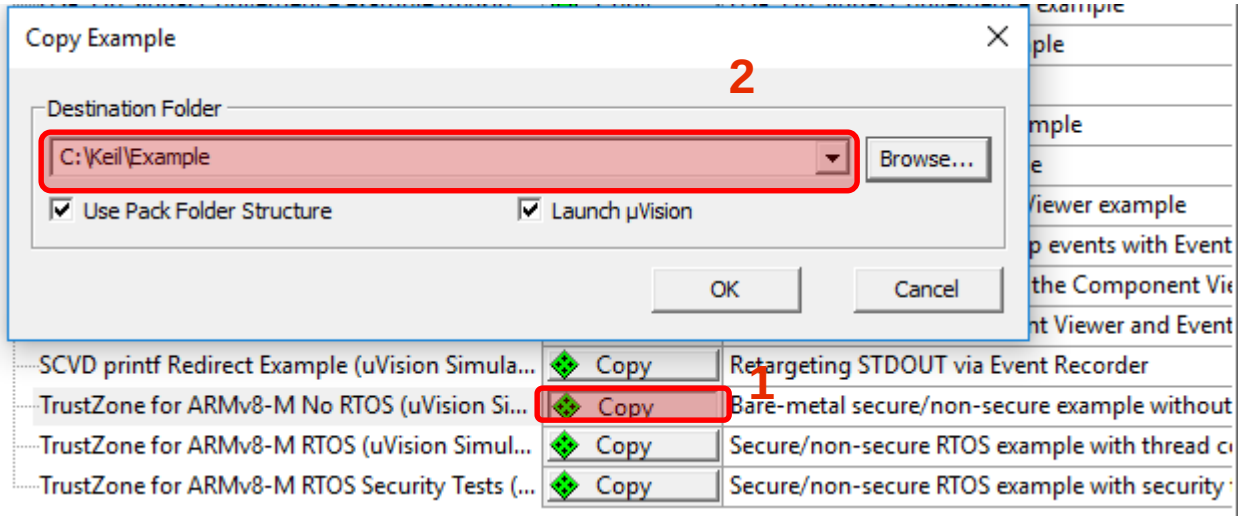


Presented by:

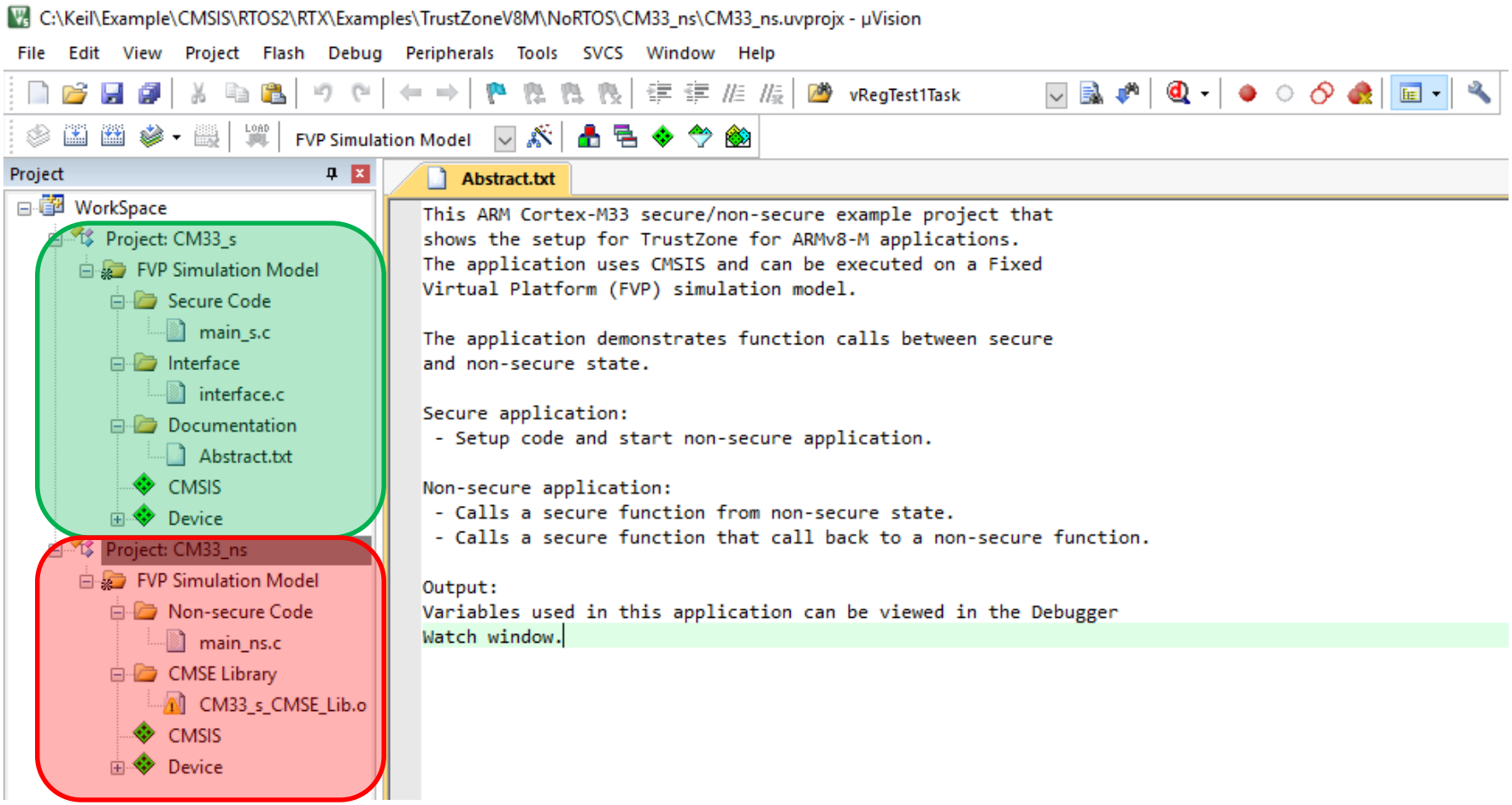
Project Setup



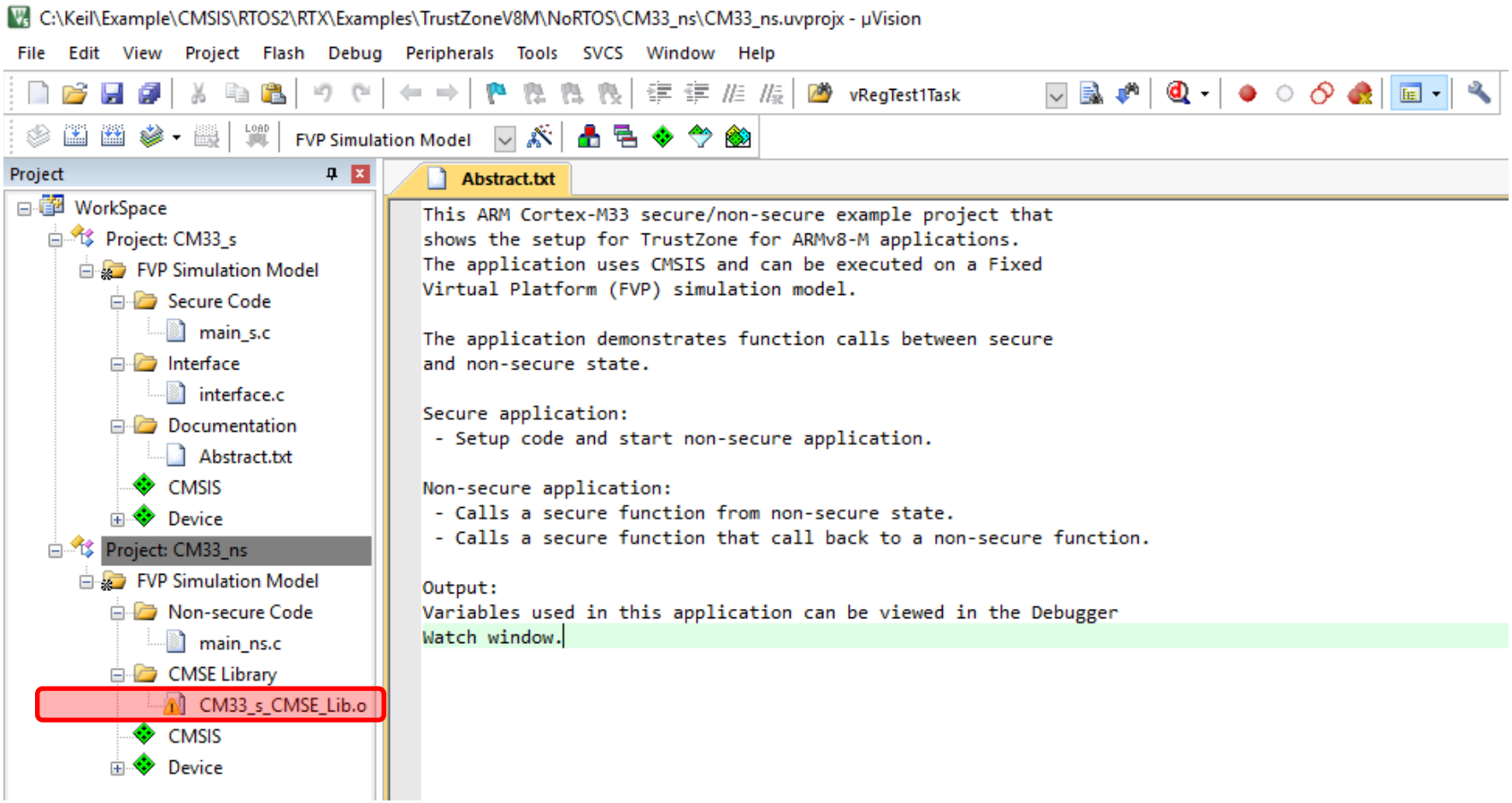
Project Setup



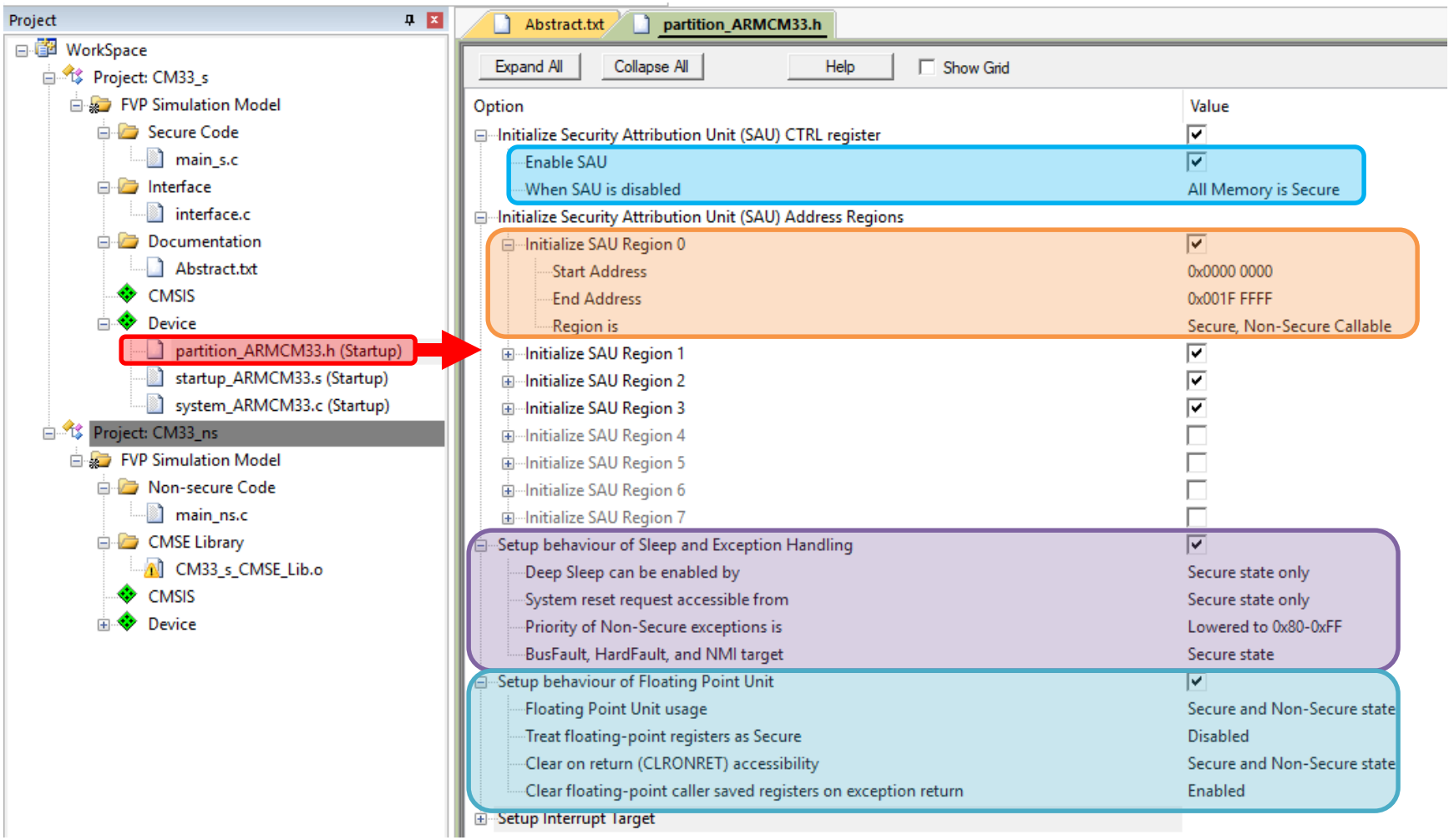
The TrustZone Workspace



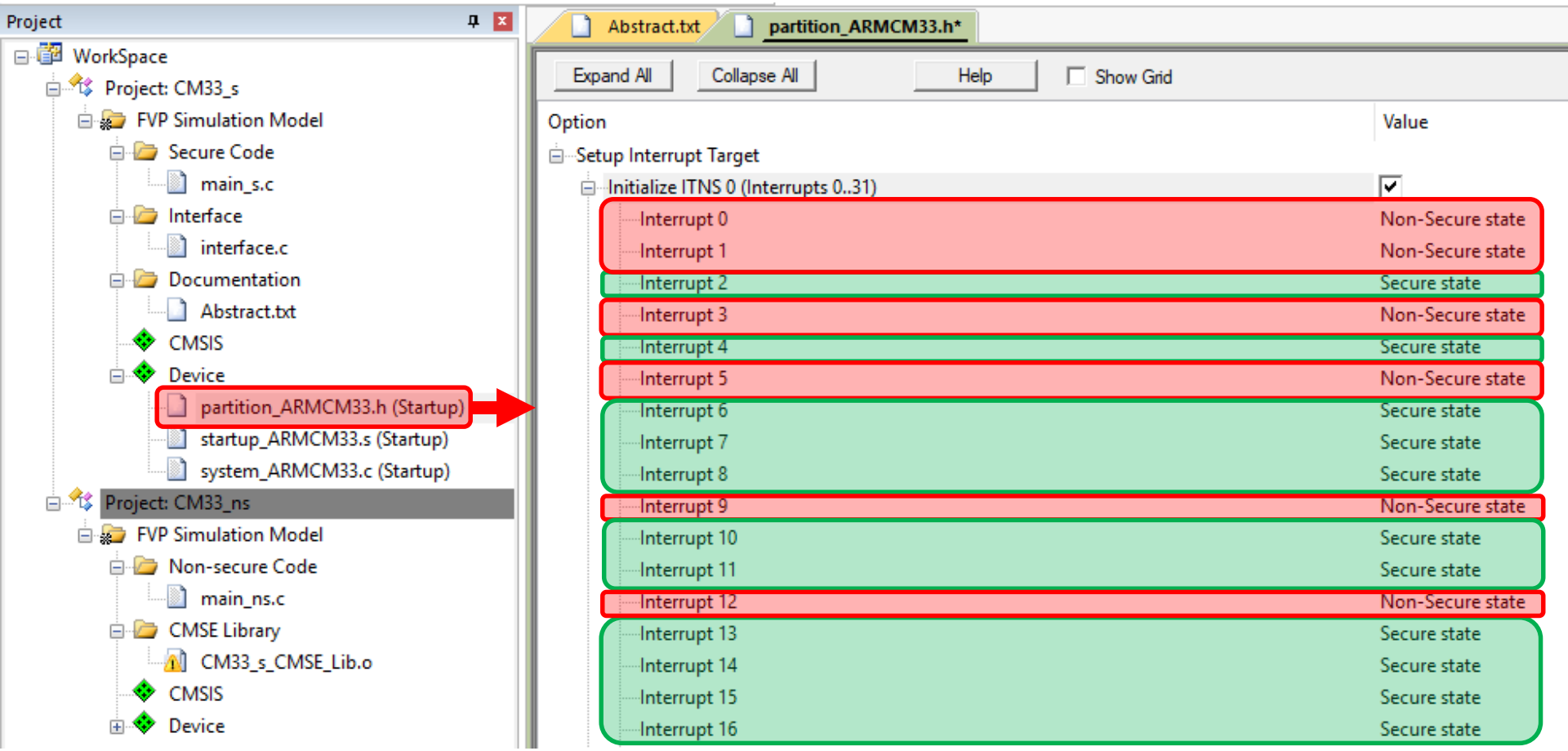
The TrustZone Workspace



Configuring the SAU



Configuring the SAU



Secure Code – main.c

```c
/* Use CMSE intrinsics */
#include <arm_cmse.h>

#include "RTE_Components.h"
#include CMSIS_device_header

/* TZ_START_NS: Start address of non-secure application */
#ifndef TZ_START_NS
#define TZ_START_NS (0x200000U)
#endif

/* typedef for non-secure callback functions */
typedef void (*funcptr_void) (void) __attribute__((cmse_nonsecure_call));

/* Secure main() */
int main(void) {
  funcptr_void NonSecure_ResetHandler;

  /* Add user setup code for secure part here*/

  /* Set non-secure main stack (MSP_NS) */
  __TZ_set_MSP_NS(*((uint32_t *)(TZ_START_NS)));

  /* Get non-secure reset handler */
  NonSecure_ResetHandler = (funcptr_void)(*((uint32_t *)((TZ_START_NS) + 4U)));

  /* Start non-secure state software application */
  NonSecure_ResetHandler();

  /* Non-secure software does not return, this code is not executed */
  while (1) {
    __NOP();
  }
}
```

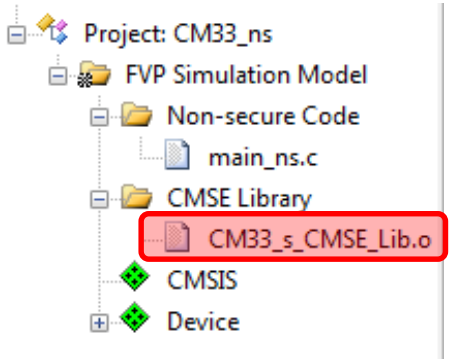
Secure Code – interface.c

```
27  #include <arm_cmse.h>      // CMSE definitions
28  #include "interface.h"     // Header file with secure interface API
29
30  /* typedef for non-secure callback functions */
31  typedef funcptr funcptr_NS __attribute__((cmse_nonsecure_call));
32
33  /* Non-secure callable (entry) function */
34  int func1(int x) __attribute__((cmse_nonsecure_entry)) {
35    return x+3;
36  }
37
38  /* Non-secure callable (entry) function, calling a non-secure callback function */
39  int func2(funcptr callback, int x) __attribute__((cmse_nonsecure_entry)) {
40    funcptr_NS callback_NS;                // non-secure callback function pointer
41    int y;
42
43    /* return function pointer with cleared LSB */
44    callback_NS = (funcptr_NS)cmse_nsfptr_create(callback);
45
46    y = callback_NS (x+1);
47
48    return (y+2);
49  }
```

Compile Secure Code

- Right click on the secure project
- Make sure it is set as active project
- Project -> Build CM33_s

```
Build Output
compiling interface.c...
assembling startup_ARMCM33.s...
compiling system_ARMCM33.c...
linking...
Program Size: Code=924 RO-data=2016 RW-data=4 ZI-data=1028
".\Objects\CM33_s.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:01
```

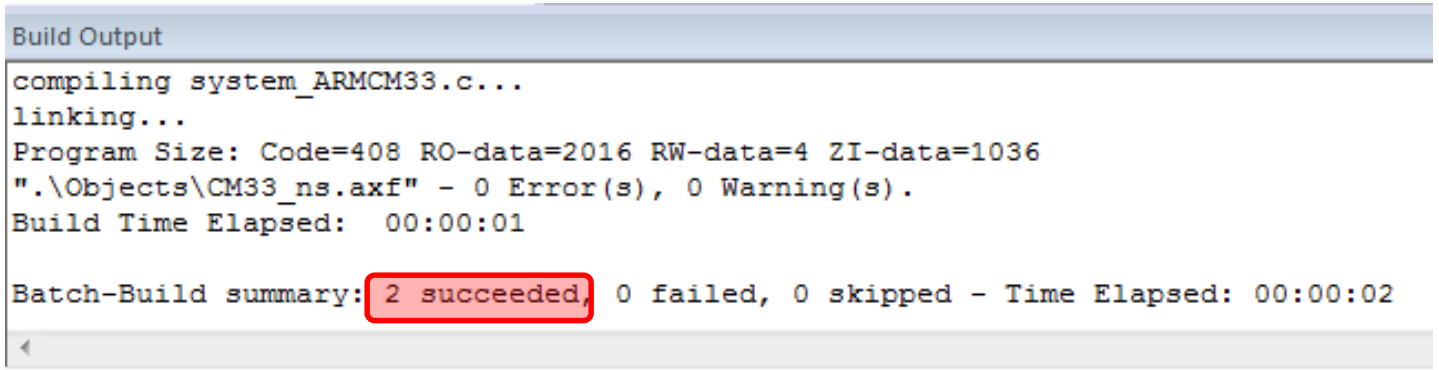


Non-Secure Code

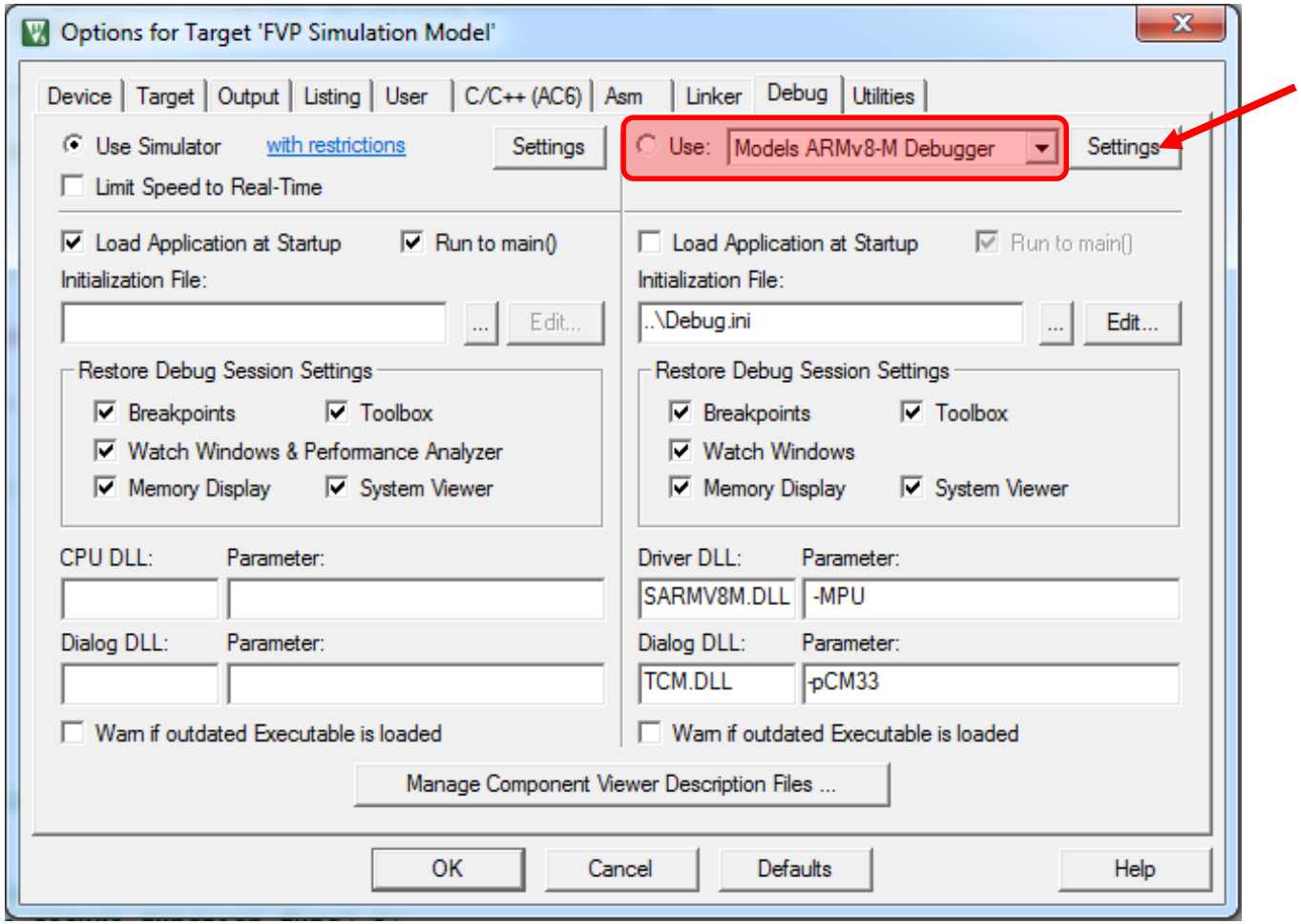
```
26  #include "..\CM33_s\interface.h"        // Interface API
27
28  extern volatile int val1, val2;
29  volatile int val1, val2;
30
31  /* Non-secure function */
32  int func3 (int x);
33
34  int func3 (int x)  {
35    return (x+4);
36  }
37
38  /* Non-secure main() */
39  int main(void) {
40
41    /* Call non-secure callable function func1 */
42    val1 = func1 (1);
43
44    /* Call non-secure callable function func2
45       with callback to non-secure function func3 */
46    val2 = func2 (func3, 2);
47
48    while (1);
49  }
```

Compile Unsecure Code

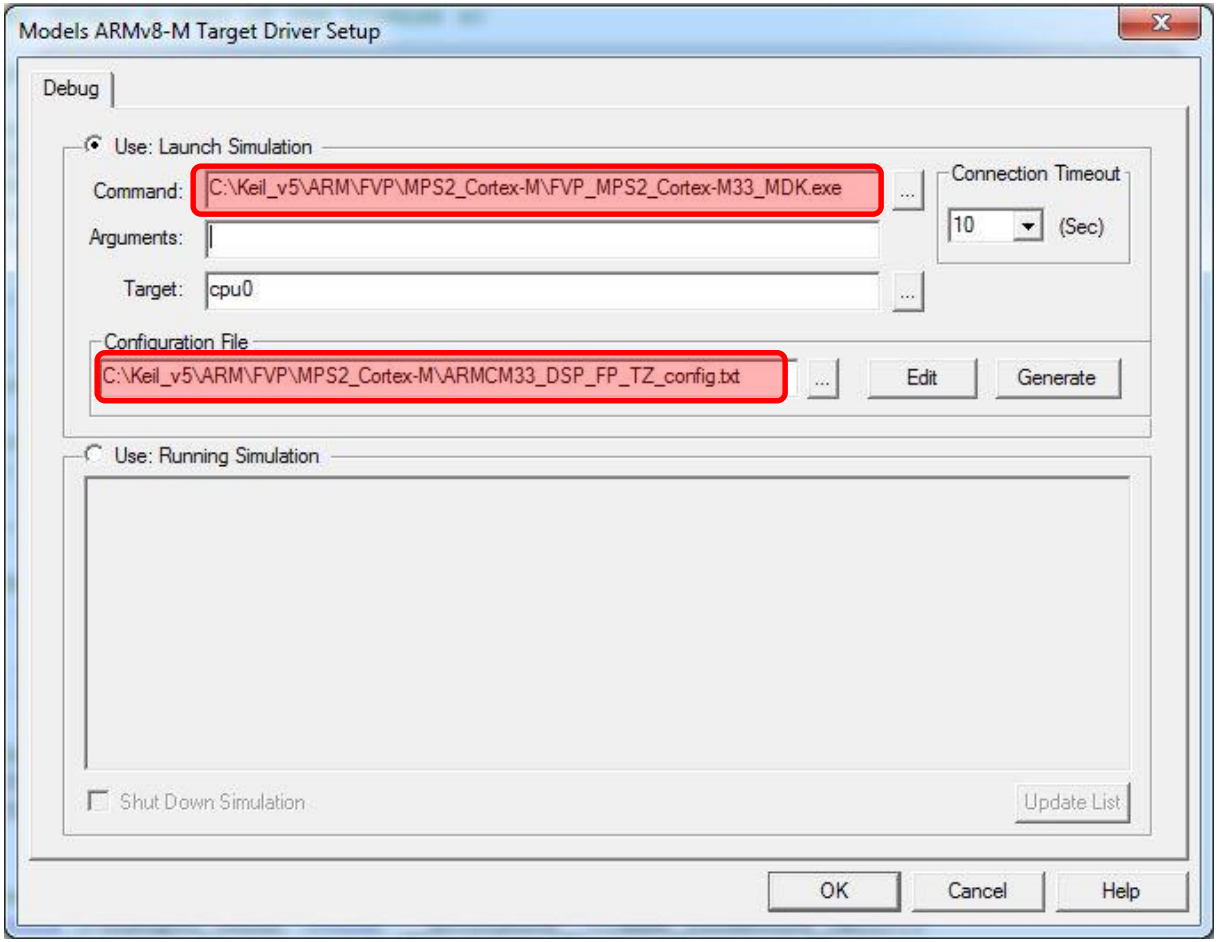
- Right click on the non-secure project
- Make sure it is the active project
- Project -> Build CM33_ns
- (alternatively, Project -> Batch Build)

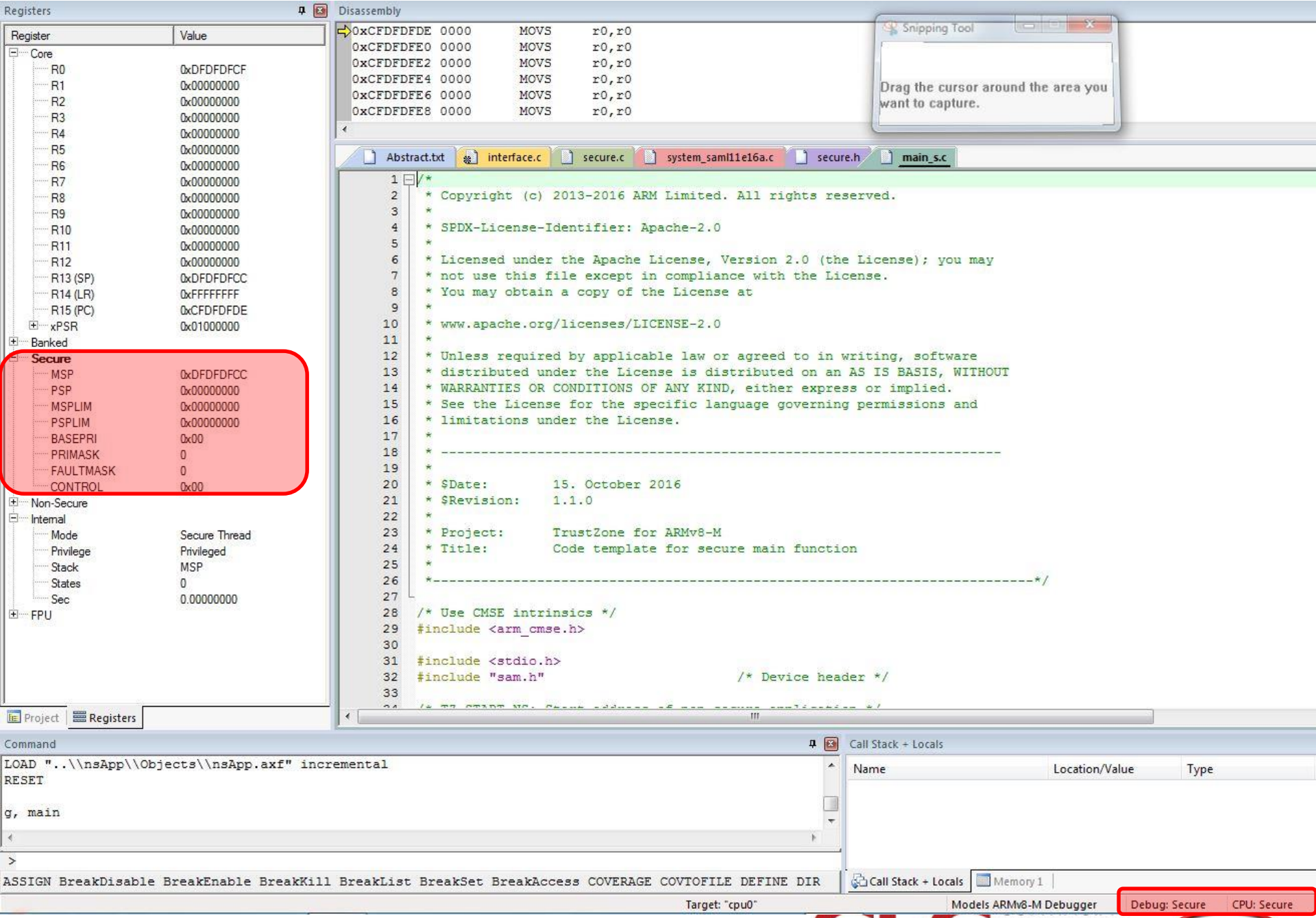


Debugging the Application



Debugging the Application




Registers
Register
Value
Core
R0 0xDFDFDFCF
R1 0x00000000
R2 0x00000000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0xDFDFDFCC
R14 (LR) 0xFFFFFFFF
R15 (PC) 0xCFDFDFDE
xPSR 0x01000000
Banked
Secure
MSP 0xDFDFDFCC
PSP 0x00000000
MSPLIM 0x00000000
PSPLIM 0x00000000
BASEPRI 0x00
PRIMASK 0
FAULTMASK 0
CONTROL 0x00
Non-Secure
Internal
Mode Secure Thread
Privilege Privileged
Stack MSP
States 0
Sec 0.00000000
FPU
Project
Registers
Disassembly
0xCFDFDFDE 0000 MOVS r0,r0
0xCFDFDFE0 0000 MOVS r0,r0
0xCFDFDFE2 0000 MOVS r0,r0
0xCFDFDFE4 0000 MOVS r0,r0
0xCFDFDFE6 0000 MOVS r0,r0
0xCFDFDFE8 0000 MOVS r0,r0
Snipping Tool
Drag the cursor around the area you want to capture.
Abstract.txt
interface.c
secure.c
system_saml11e16a.c
secure.h
main_s.c
1 /*
2 * Copyright (c) 2013-2016 ARM Limited. All rights reserved.
3 *
4 * SPDX-License-Identifier: Apache-2.0
5 *
6 * Licensed under the Apache License, Version 2.0 (the License); you may
7 * not use this file except in compliance with the License.
8 * You may obtain a copy of the License at
9 *
10 * www.apache.org/licenses/LICENSE-2.0
11 *
12 * Unless required by applicable law or agreed to in writing, software
13 * distributed under the License is distributed on an AS IS BASIS, WITHOUT
14 * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15 * See the License for the specific language governing permissions and
16 * limitations under the License.
17 *
18 * --
19 *
20 * $Date: 15. October 2016
21 * $Revision: 1.1.0
22 *
23 * Project: TrustZone for ARMv8-M
24 * Title: Code template for secure main function
25 *
26 *---*/
27
28 /* Use CMSE intrinsics */
29 #include <arm_cmse.h>
30
31 #include <stdio.h>
32 #include "sam.h" /* Device header */
33
Command
LOAD "..\\nsApp\\Objects\\nsApp.axf" incremental
RESET
g, main
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR
Call Stack + Locals
Name
Location/Value
Type
Call Stack + Locals
Memory 1
Target: "cpu0"
Models ARMv8-M Debugger
Debug: Secure
CPU: Secure

Additional Resources

- Download Course Material for
  - C/C++ Doxygen Templates
  - Example source code
  - Blog
  - YouTube Videos
- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm

From www.beningo.com under

- Blog > CEC – Securing IoT Devices using Arm TrustZone