



FTF 2016
TECHNOLOGY FORUM

FreeRTOS SOLUTIONS FOR KINETIS MCU'S

FTF-DES-N1956

ERICH STYGER
FTF-DES-N1956
MAY 17, 2016

PUBLIC USE



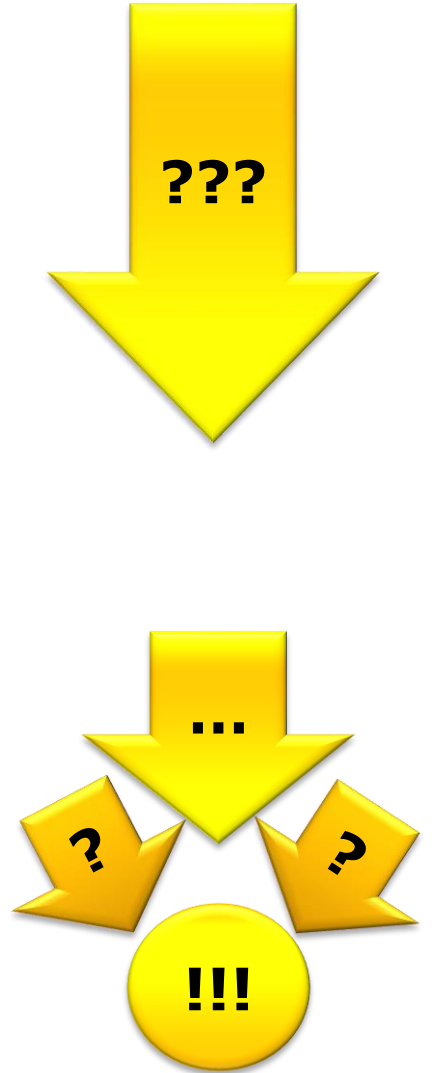
AGENDA

- Introduction
 - FRDM-K22F
 - FreeRTOS
 - Kinetis SDK
 - Kinetis Design Studio (KDS)
- Eclipse Kernel Awareness
- SEGGER SystemView
- Percepio FreeRTOS+Trace
- Wrap-up and Closure

Session Introduction and Objectives

- This session shows how you can use Kinetis SDK + FreeRTOS Solutions
- It shows how to debug a SDK application with FreeRTOS which includes an instrumented RTOS with Trace
- The application uses Kinetis Design Studio V3.2.0, Kinetis SDK V2.0 and the FRDM-K22F board, but the concepts are applicable to different setup as well

- After completing this session you will be able to:
 - Effectively use solutions for Kinetis SDK + FreeRTOS applications
 - Reduce debugging time
 - Improve application performance



Session Philosophy

- Preinstalled software on lab computers
 - Repeat things on your own machine: installation instructions for installation at a later time
- You have received a Kinetis Design Studio + Kinetis SDK + FreeRTOS Application with the FRDM-K22F board
- Theory input with Lab in parallel
 - Going through the slide material
 - Perform the things learned with your lab computer
 - Additional information/links provided



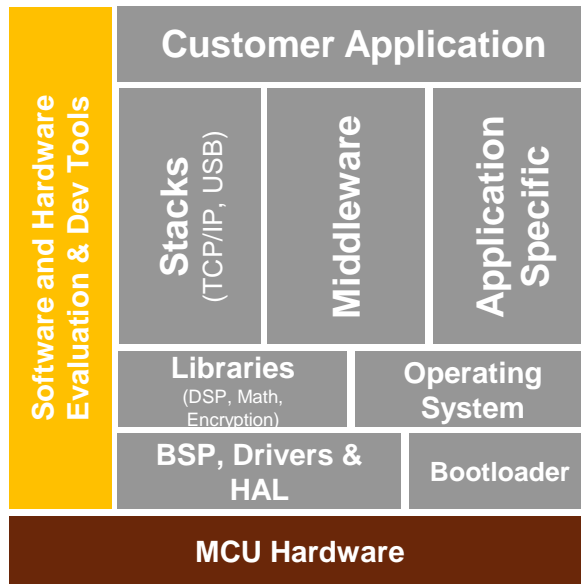
Kinetis Design Studio



No-cost integrated development environment (IDE) for Kinetis MCUs



Eclipse and GCC-based IDE for C/C++ editing, compiling and debugging



Product Features:

- A free of charge and unlimited IDE for Kinetis MCUs
- A basic IDE that offers robust editing, compiling and debugging
- Based on Eclipse, GCC, GDB and other open-source technologies
- Includes Processor Expert with Kinetis SDK integration
- Supports all existing Kinetis devices via Processor Expert and new project wizard
- All new Kinetis devices will also feature the Kinetis SDK with Processor Expert configurability
- Code Generation for GNU, IAR and Keil
- Host operating systems:
 - Microsoft Windows 7/8/10
 - Linux (Ubuntu, Redhat, Centos) (64bit)
 - Mac OS X and Segger J-Link
- Support for SEGGER, P&E and OpenSDA/CMSIS-DAP debugger targets
- Support for Eclipse plug-ins including RTOS-awareness (i.e. MQX, FreeRTOS)
- Commercial upgrade path: Somnium DRT

Learn more at: www.nxp.com/KDS

Community: <https://community.freescale.com/community/kinetis-design-studio>



Kinetis Software Development Kit v2.x (KSDK v2)

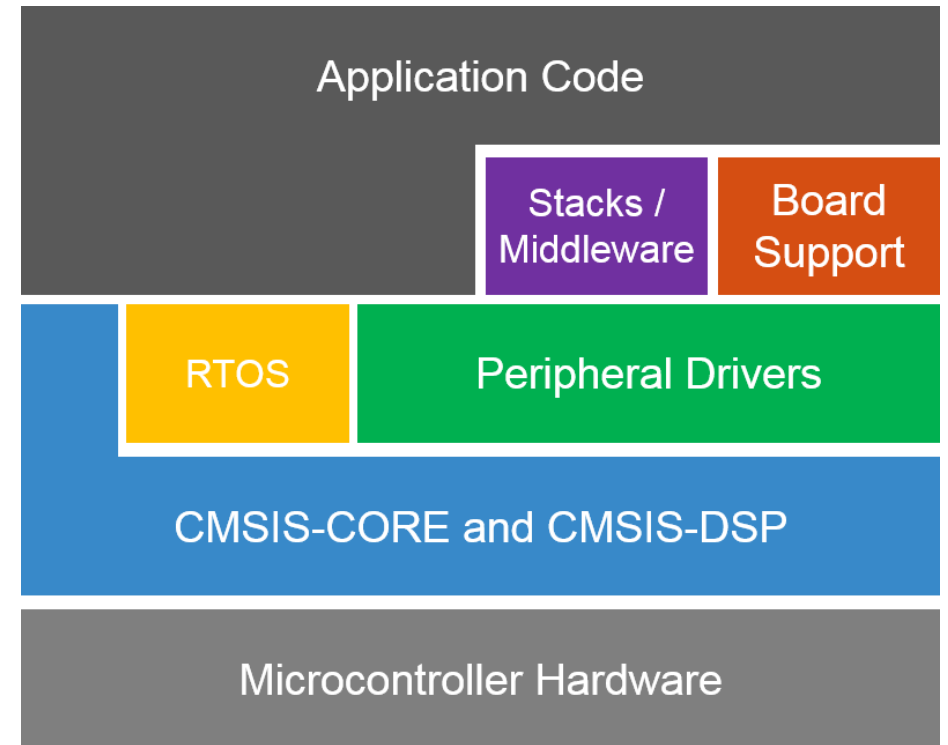
Learn more at: www.nxp.com/KSDK



- Architecture:
 - CMSIS-CORE compatible
 - Single driver for each peripheral
 - Transactional APIs w/ optional DMA support for communication peripherals
- Integrated RTOS:
 - FreeRTOS, Micrium uC/OS-II & -III
 - RTOS-native driver wrappers
- Integrated Stacks & Middleware
 - USB Host, Device and OTG
 - lwIP, FatFS
 - Crypto acceleration plus wolfSSL & mbedTLS
 - SD and eMMC card support
- Reference Software:
 - Peripheral Driver usage examples
 - Application Demos
 - FreeRTOS usage demos
- License:
 - BSD 3-clause for startup, drivers, USB stack
- Toolchains:
 - KDS, IAR, Keil, Atollic, GCC w/ CMake
- Quality
 - Production grade software
 - MISRA 2004 compliance
 - Checked with Coverity Static Analysis tools



The software framework and reference for Kinetis MCU application development



Freedom Development Platforms

Product Features

- Low-cost (starting at \$12.95 USD)
- Designed in an industry-standard compact form factor (Arduino R3)
- Easy access to the MCU I/O pins
- Integrated open-standard serial and debug interface (OpenSDA)
- Compatible with a rich-set of third-party expansion boards

Software and Support

- Rich ARM ecosystem includes Kinetis Design Studio, Keil, IAR, SEGGER, mbed and more
- mbed-enablement through the built-in USB flash programming interface (OpenSDA)

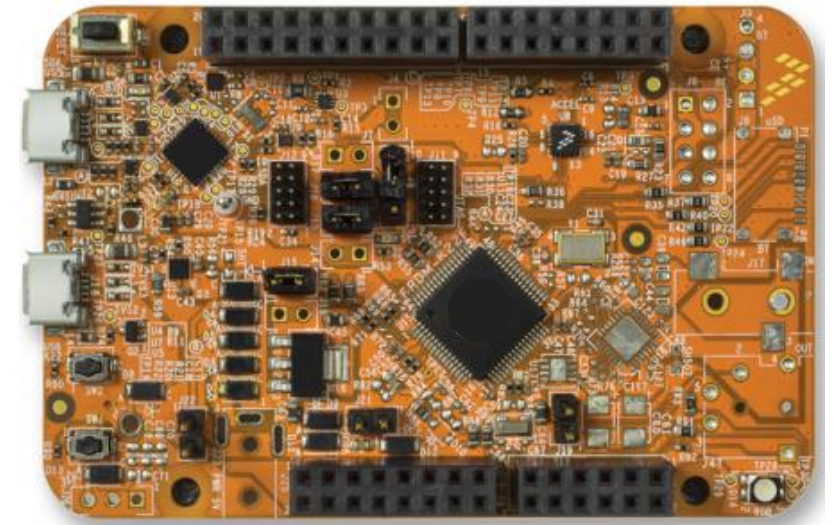
Learn more at: www.nxp.com/Freedom
www.nxp.com/mbed



Low-cost/low-power development hardware



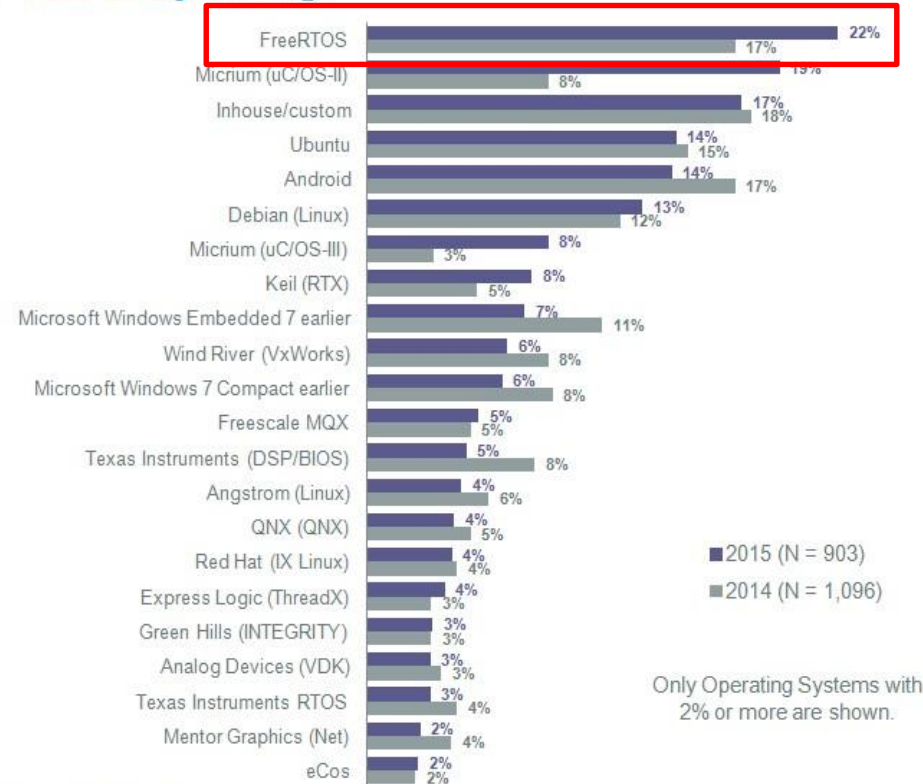
Enables quick application prototyping and demonstration of **Kinetis MCU families**



UBM Embedded Market Study

2015 UBM Electronics Embedded Markets Study

Please select ALL of the operating systems you are currently using.



Base: Currently using an operating system

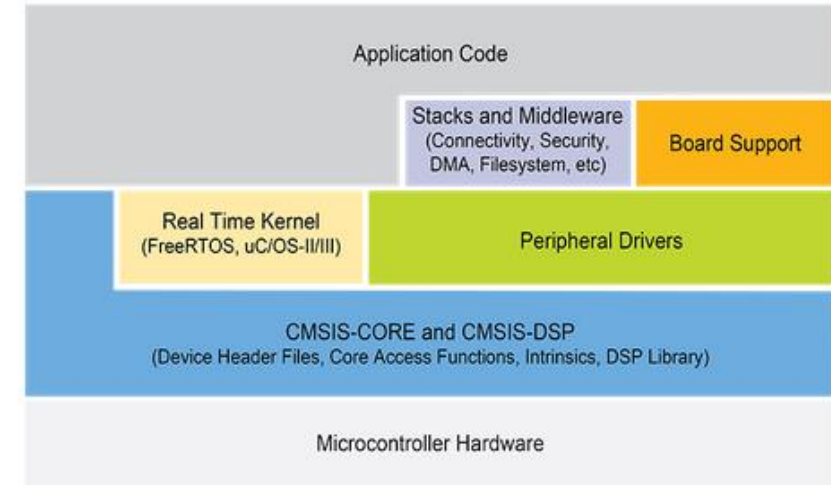
FreeRTOS and Tools

- <http://www.freertos.org>
- Maintained by Real Time Engineers Ltd., London (Richard Barry)
- Open Source, free-of-charge, royalty free
- >35 architectures, >113,000 downloads in 2015
- Portable, simple to learn and use
- Ecosystem and commercial supported ports available
 - OpenRTOS: commercial supported version
 - SafeRTOS: special version dedicated to safety critical systems
 - **SEGGER SystemView**: System Visibility with Segger RTT
 - **PERCEPIO Tracalizer**: Powerful Analysis Views



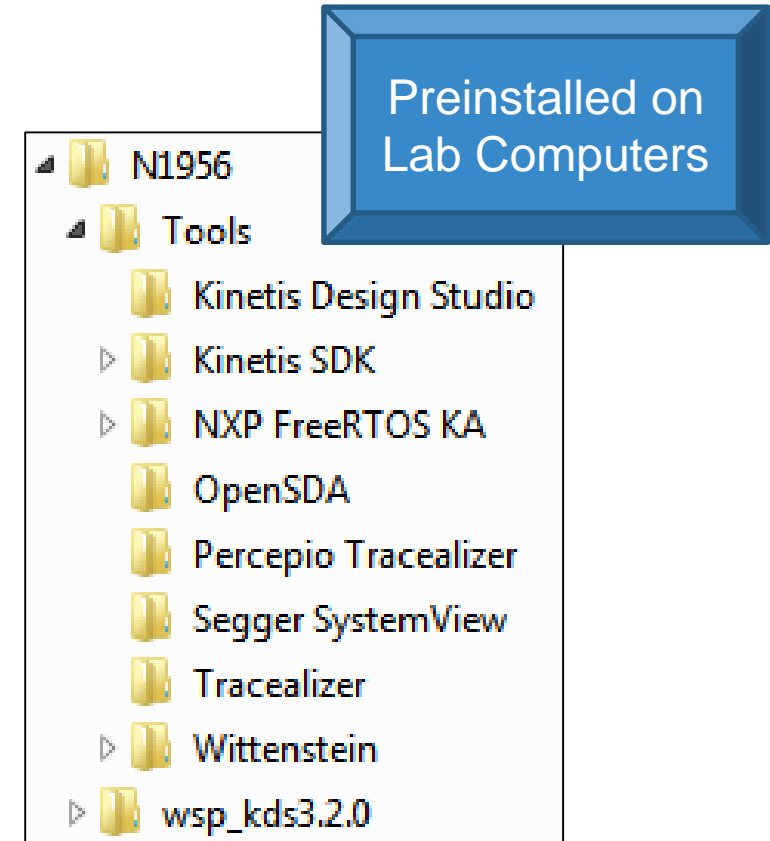
NXP Software and Tools for FreeRTOS

- **FreeRTOS** included in NXP Kinetis SDK:
 - <http://www.nxp.com/ksdk>
 - Open Source, permissible license
 - CMSIS-Core startup and drivers
 - FreeRTOS, Micrium μ C/OS and bare metal support
- **Kinetis Design Studio** based on GNU/Eclipse:
 - <http://www.nxp.com/kds>
 - Kinetis SDK device support added with Eclipse Updates
 - Processor Expert used for Rapid Application Development
- **FreeRTOS Kernel Awareness Plugin**



Lab Notes

- Lab Files in C:\FTF\N1956
- Lab_KSDK_FreeRTOS
 - KDS Project used in labs
- Tools
 - Tools used in labs
- wsp_kds3.2.0
 - Kinetis Design Studio workspace used in labs



Please leave the boards/cables/notebooks in the room!

Lab Setup

Setup and installation of the tools used in this session...



Preinstalled on
Lab Computers

Outline: Setup

- **Problem**

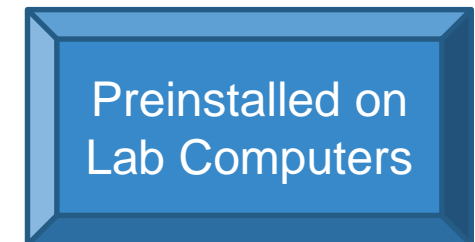
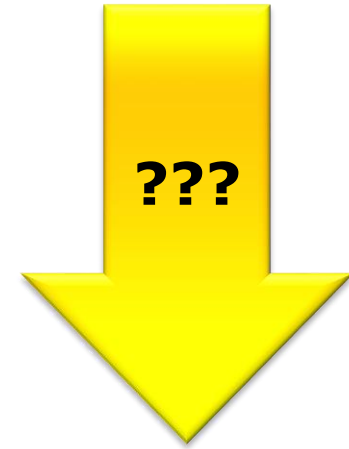
- Getting Tools and Project in place

- **Solution**

- Installation of Kinetis Design Studio IDE, Kinetis SDK and Eclipse Update

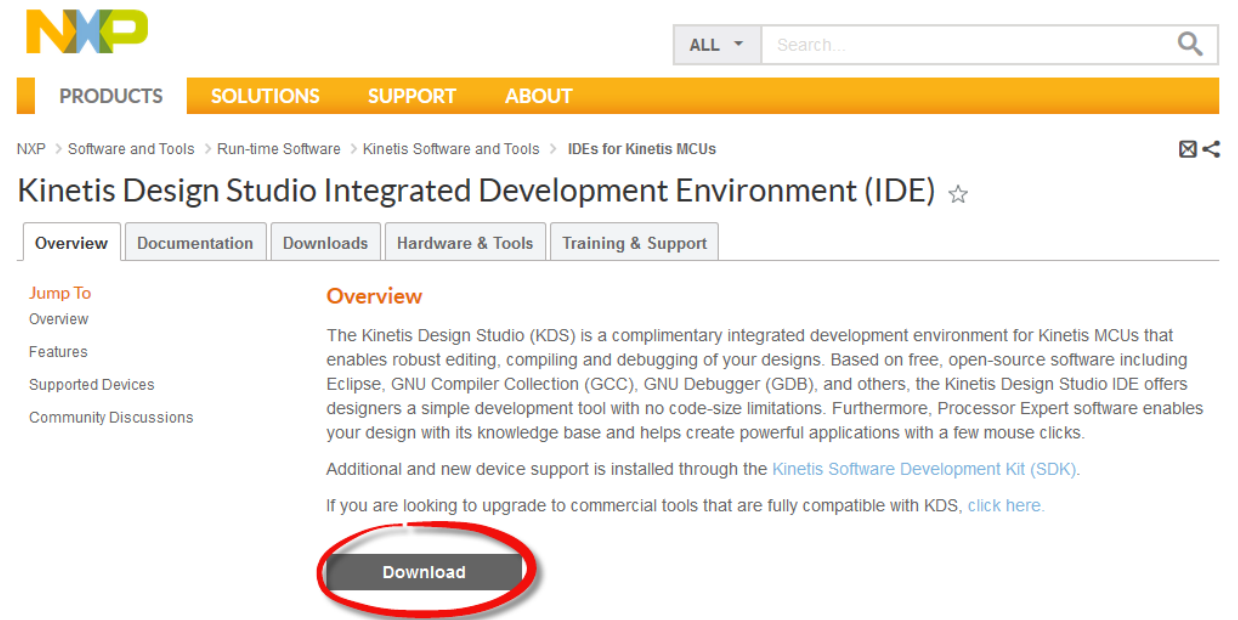
- **Steps**

- Installation of Kinetis Design Studio
- Installation of Kinetis SDK
- Adding Kinetis SDK support to Kinetis Design Studio
- Importing existing project into workspace



Installation of Kinetis Design Studio IDE

- Download Kinetis Design Studio from <http://www.nxp.com/kds>
- Version used in Lab: KDS v3.2.0
- Hosts: Windows, Linux, Mac OS X
- Run the setup with default settings

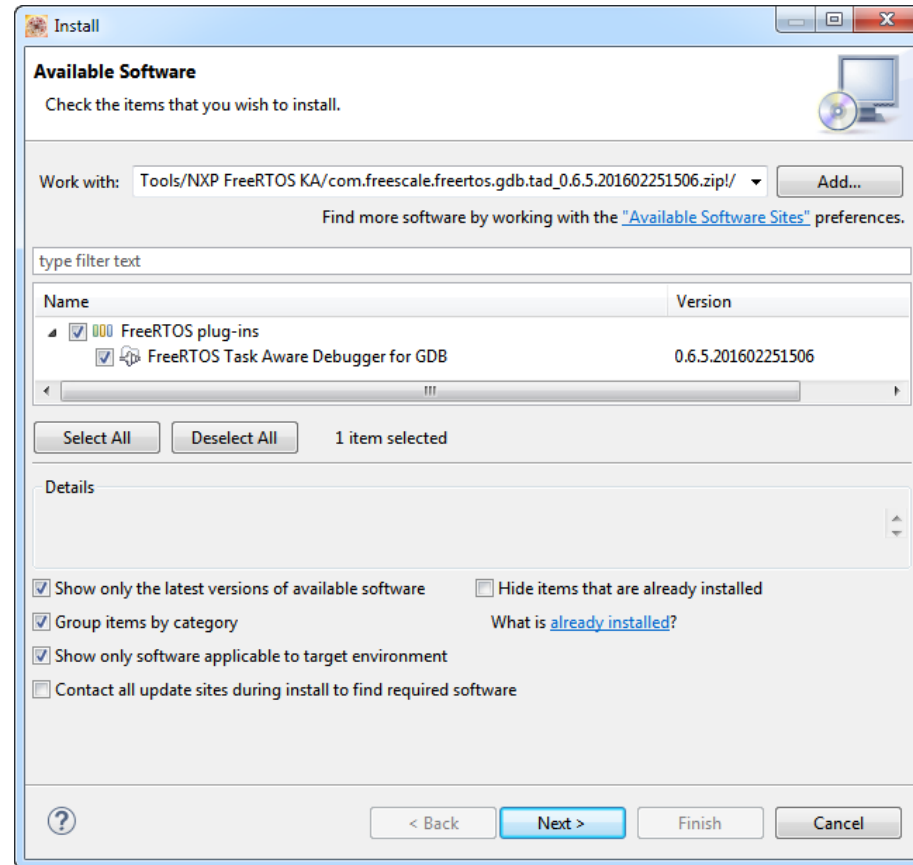


Preinstalled on
Lab Computers

Kernel Awareness Installation

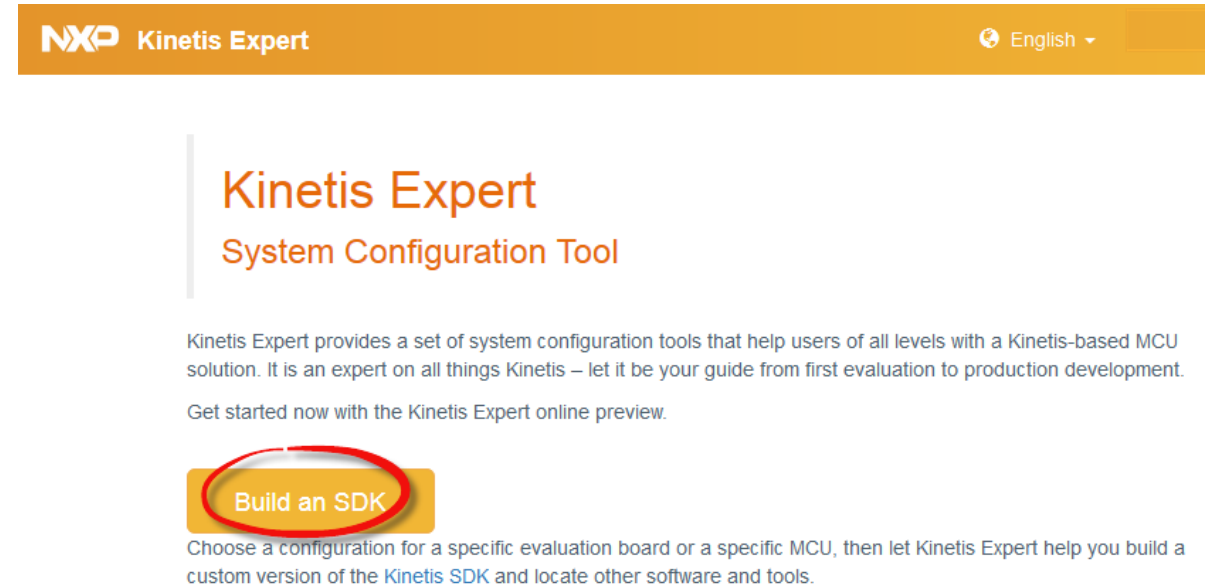
- Menu Help > Install New Software
- N1956/Tools/NXP FreeRTOS KA/* .zip

Pre-Installed on
Lab Computers



Get the Kinetis SDK

- Build the SDK on <http://kex.nxp.com>
- Download file from the Vault
- Extract into c:\nxp\KSDK
- → Lab project has SDK inside lab project ('standalone')

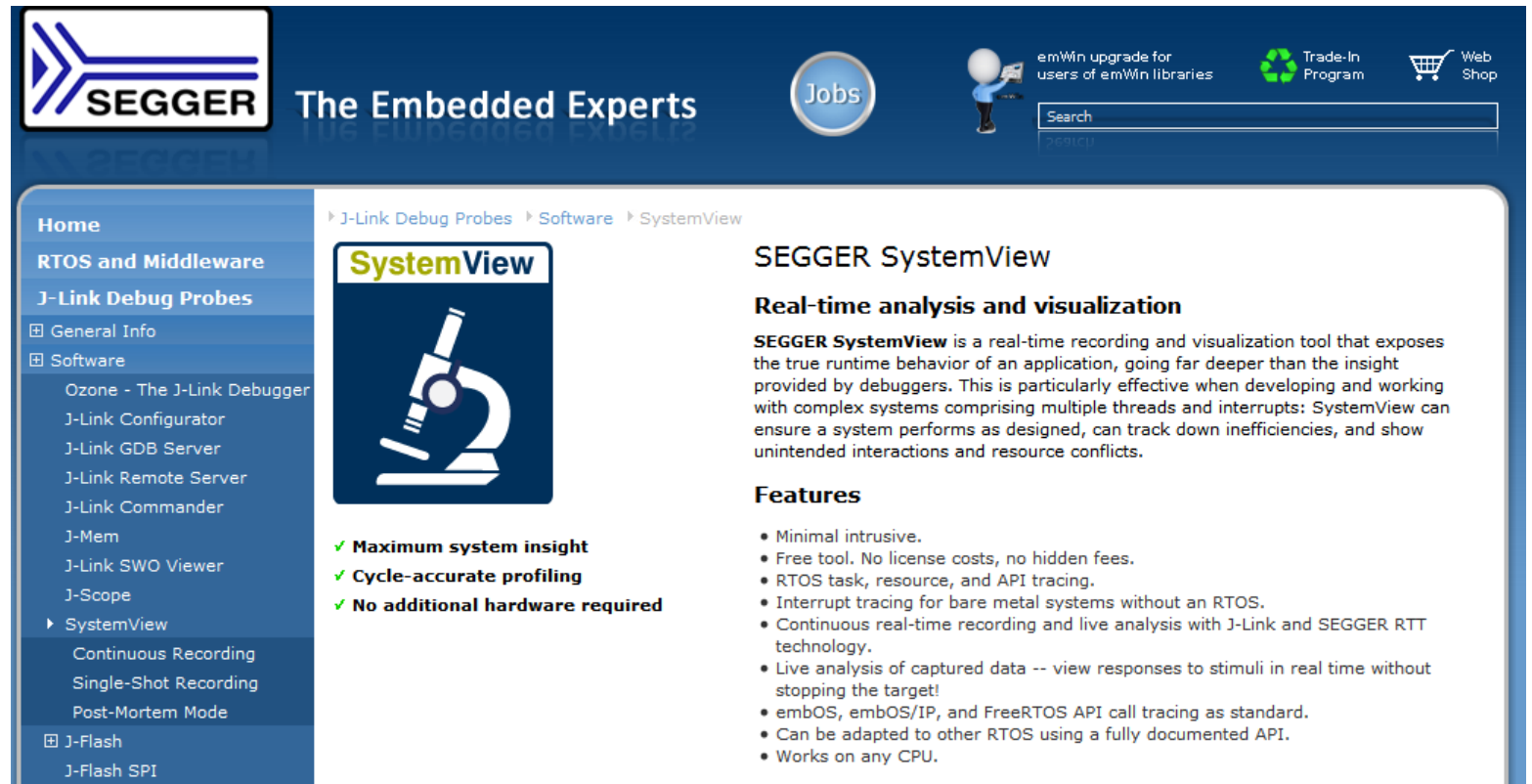


The screenshot shows the NXP Kinetis Expert website. At the top, there is a navigation bar with the NXP logo and 'Kinetis Expert' text on the left, and a language dropdown menu set to 'English' on the right. Below the navigation bar, the main heading reads 'Kinetis Expert System Configuration Tool'. A descriptive paragraph states: 'Kinetis Expert provides a set of system configuration tools that help users of all levels with a Kinetis-based MCU solution. It is an expert on all things Kinetis – let it be your guide from first evaluation to production development. Get started now with the Kinetis Expert online preview.' Below this text is a prominent yellow button with the text 'Build an SDK', which is circled in red. Underneath the button, a smaller line of text says: 'Choose a configuration for a specific evaluation board or a specific MCU, then let Kinetis Expert help you build a custom version of the Kinetis SDK and locate other software and tools.'

Preinstalled on
Lab Computers

Segger SystemView

- Download SystemView from <https://www.segger.com/systemview.html>
- Run the setup



SEGGER The Embedded Experts

Jobs

emWin upgrade for users of emWin libraries

Trade-In Program

Web Shop

Search

Home

RTOS and Middleware

J-Link Debug Probes

- General Info
- Software
 - Ozone - The J-Link Debugger
 - J-Link Configurator
 - J-Link GDB Server
 - J-Link Remote Server
 - J-Link Commander
 - J-Mem
 - J-Link SWO Viewer
 - J-Scope
 - SystemView
 - Continuous Recording
 - Single-Shot Recording
 - Post-Mortem Mode
- J-Flash
 - J-Flash SPI

SystemView

Maximum system insight

Cycle-accurate profiling

No additional hardware required

SEGGER SystemView

Real-time analysis and visualization

SEGGER SystemView is a real-time recording and visualization tool that exposes the true runtime behavior of an application, going far deeper than the insight provided by debuggers. This is particularly effective when developing and working with complex systems comprising multiple threads and interrupts: SystemView can ensure a system performs as designed, can track down inefficiencies, and show unintended interactions and resource conflicts.

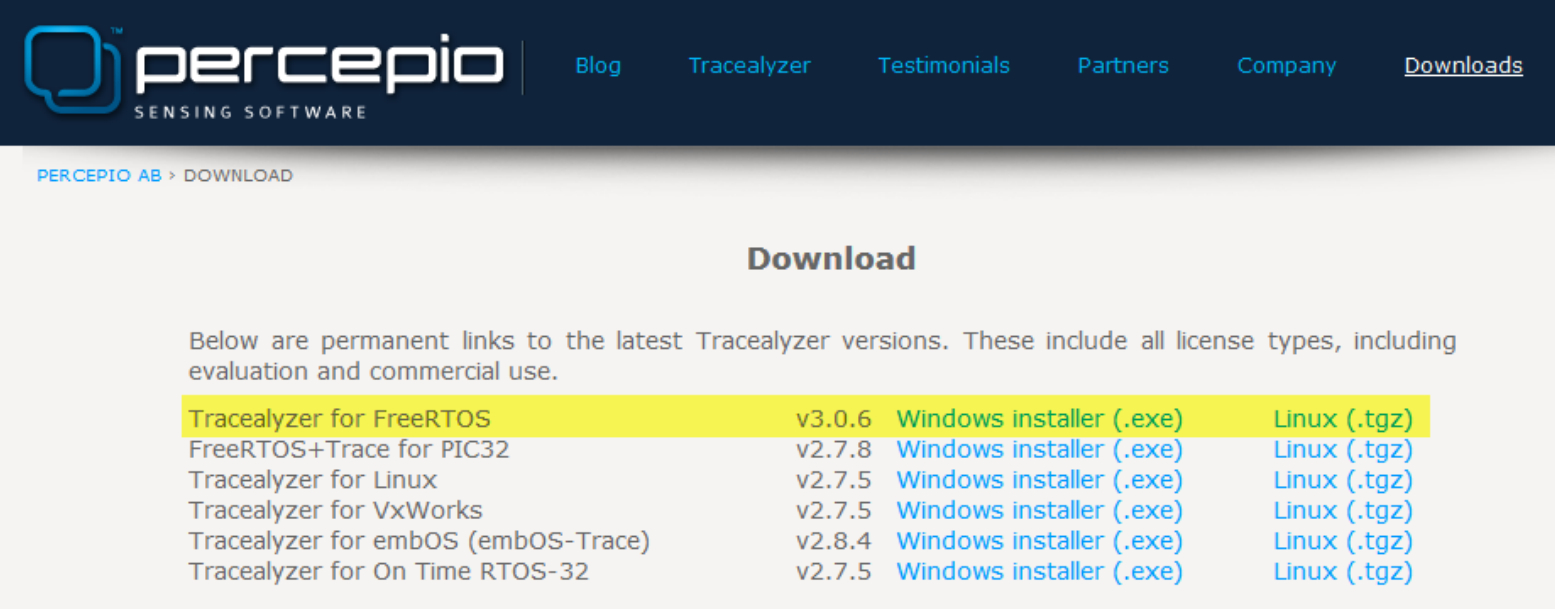
Features

- Minimal intrusive.
- Free tool. No license costs, no hidden fees.
- RTOS task, resource, and API tracing.
- Interrupt tracing for bare metal systems without an RTOS.
- Continuous real-time recording and live analysis with J-Link and SEGGER RTT technology.
- Live analysis of captured data -- view responses to stimuli in real time without stopping the target!
- embOS, embOS/IP, and FreeRTOS API call tracing as standard.
- Can be adapted to other RTOS using a fully documented API.
- Works on any CPU.

Preinstalled on
Lab Computers

FreeRTOS+Trace Tracealyzer from Percepio

- Download Tracealyzer for FreeRTOS from <http://percepio.com/download/>
- Free demo version (task scheduling only)
- 30 day evaluation (Professional Edition) installed on lab computers



percepio
SENSING SOFTWARE

Blog Tracealyzer Testimonials Partners Company Downloads

PERCEPIO AB > DOWNLOAD

Download

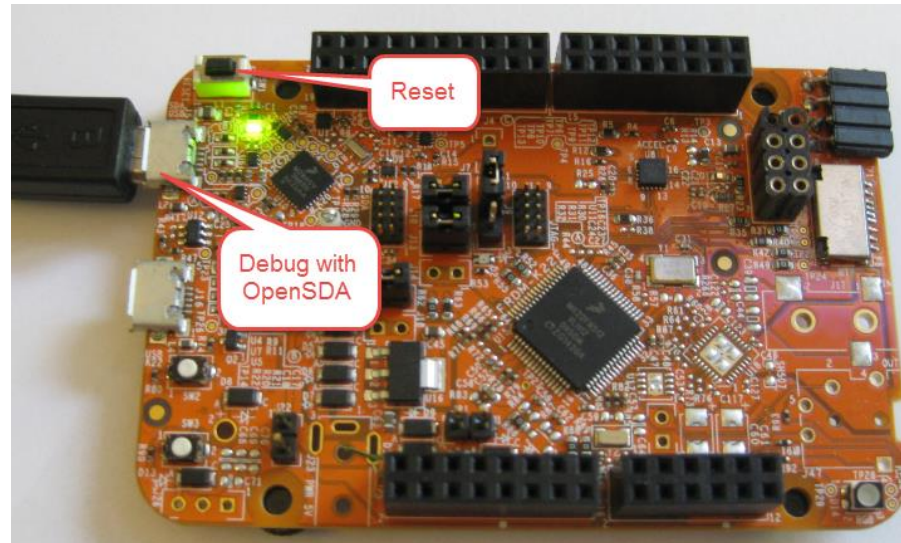
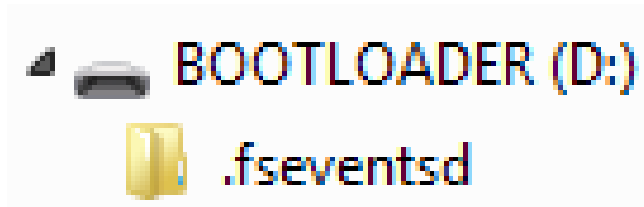
Below are permanent links to the latest Tracealyzer versions. These include all license types, including evaluation and commercial use.

Tracealyzer for FreeRTOS	v3.0.6	Windows installer (.exe)	Linux (.tgz)
FreeRTOS+Trace for PIC32	v2.7.8	Windows installer (.exe)	Linux (.tgz)
Tracealyzer for Linux	v2.7.5	Windows installer (.exe)	Linux (.tgz)
Tracealyzer for VxWorks	v2.7.5	Windows installer (.exe)	Linux (.tgz)
Tracealyzer for embOS (embOS-Trace)	v2.8.4	Windows installer (.exe)	Linux (.tgz)
Tracealyzer for On Time RTOS-32	v2.7.5	Windows installer (.exe)	Linux (.tgz)

Preinstalled on
Lab Computers

Lab Setup: Segger OpenSDA Firmware

- Power Board with 'SDA USB' while RESET button pressed
 - Board enumerates as BOOTLOADER
 - Copy firmware file (JLink_OpenSDA_V2_2015-10-13.bin) to BOOTLOADER drive
 - Green LED blinks fast, then mostly on
- Repower board with SDA USB port normally
 - USB drivers might enumerate
 - Green LED mostly on

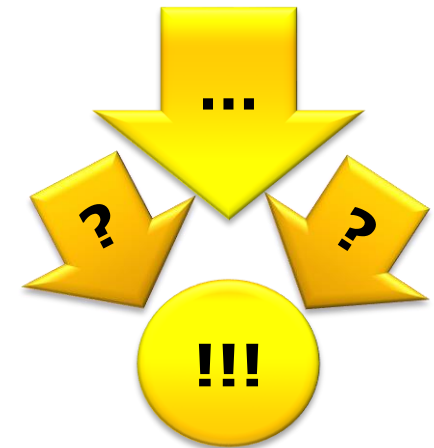


Preinstalled on
Lab Boards

Summary: Lab Setup

- Setup on Lab Machines
 - FRDM-K22F Board
 - OpenSDA Debug (Segger J-Link)
 - Kinetis Design Studio
 - FreeRTOS Kernel Awareness
 - Kinetis SDK
 - Segger SystemView
 - Percepio Tracealyzer

Preinstalled on
Lab Boards

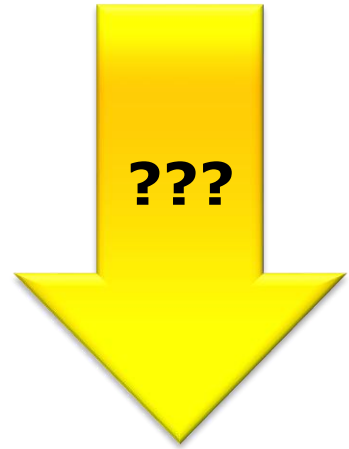


Lab Application

Go, figure it out and make it better!

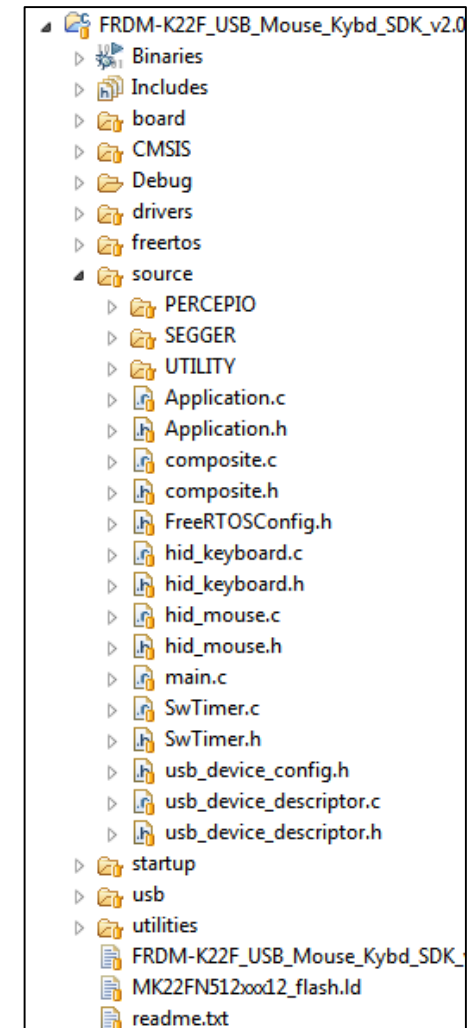
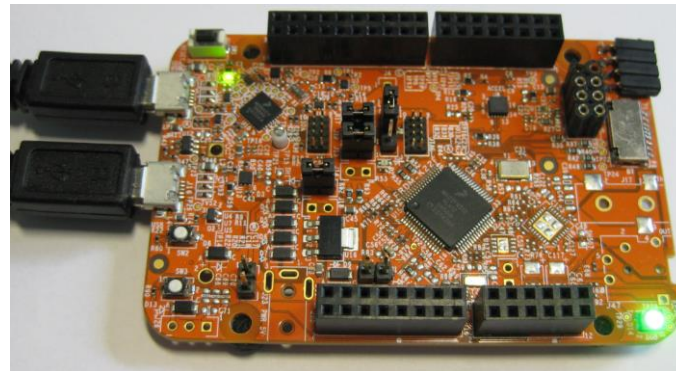
Outline: Lab Application

- High Level overview (Tools, board, Application)
- First steps
 - Starting Kinetis Design Studio IDE
 - New workspace
 - Importing Project
 - Building
 - Debugging



Lab Board and Application

- **FRDM-K22F** (ARM Cortex M4F)
- OpenSDA v2 with **Segger J-Link** OpenSDA firmware
- **Kinetis SDK v2.0** with **FreeRTOS v8.2.3**
- USB HID Composite (Mouse + Keyboard)
- **FreeRTOS+Trace** enabled
 - **Segger SystemView** with Segger RTT
 - **Percepio Tracealyzer** with Segger RTT



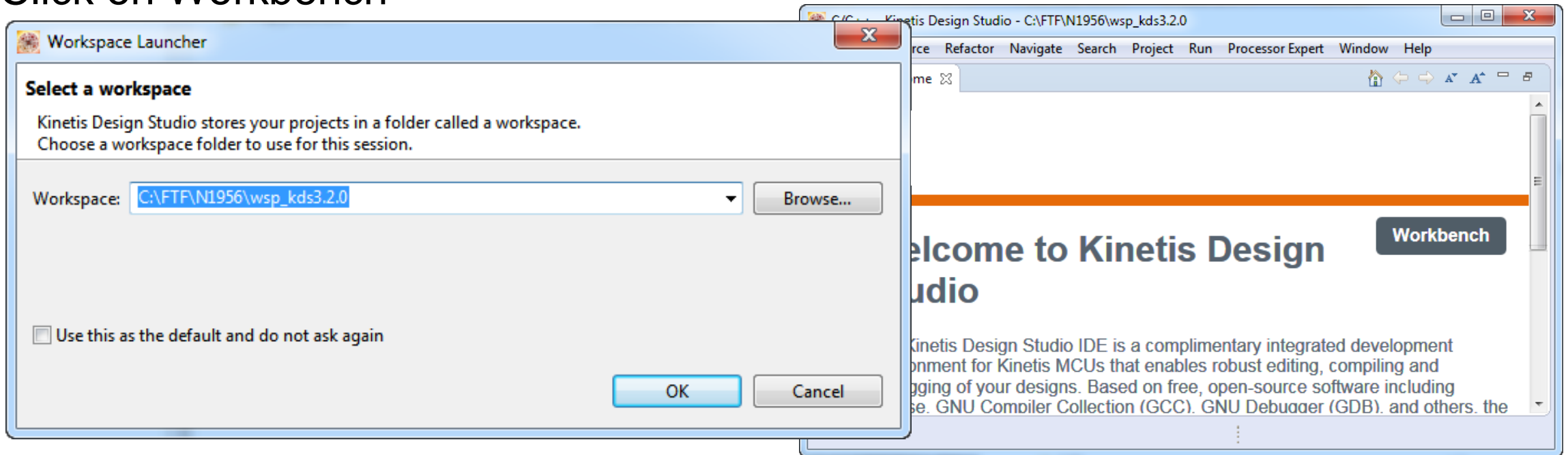
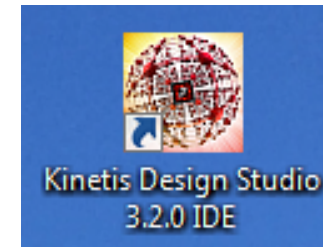
Board Functions

Connect both
USB cables to
host PC



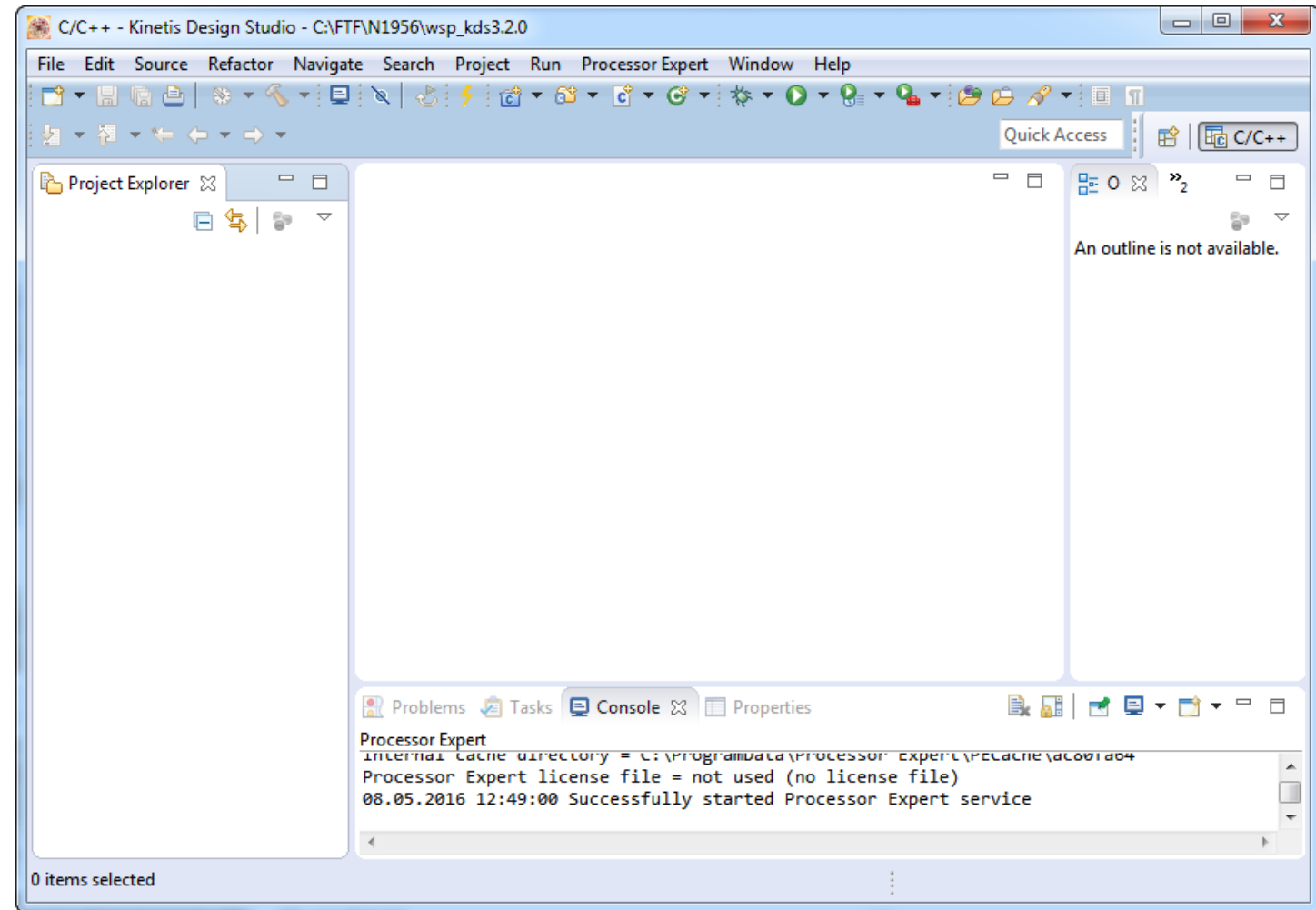
Launching Kinetis Design Studio IDE

- Launch KDS from Shortcut on Desktop
- Choose Workspace
 - C:\FTF\N1956\wsp_kds3.2.0
- Click on Workbench



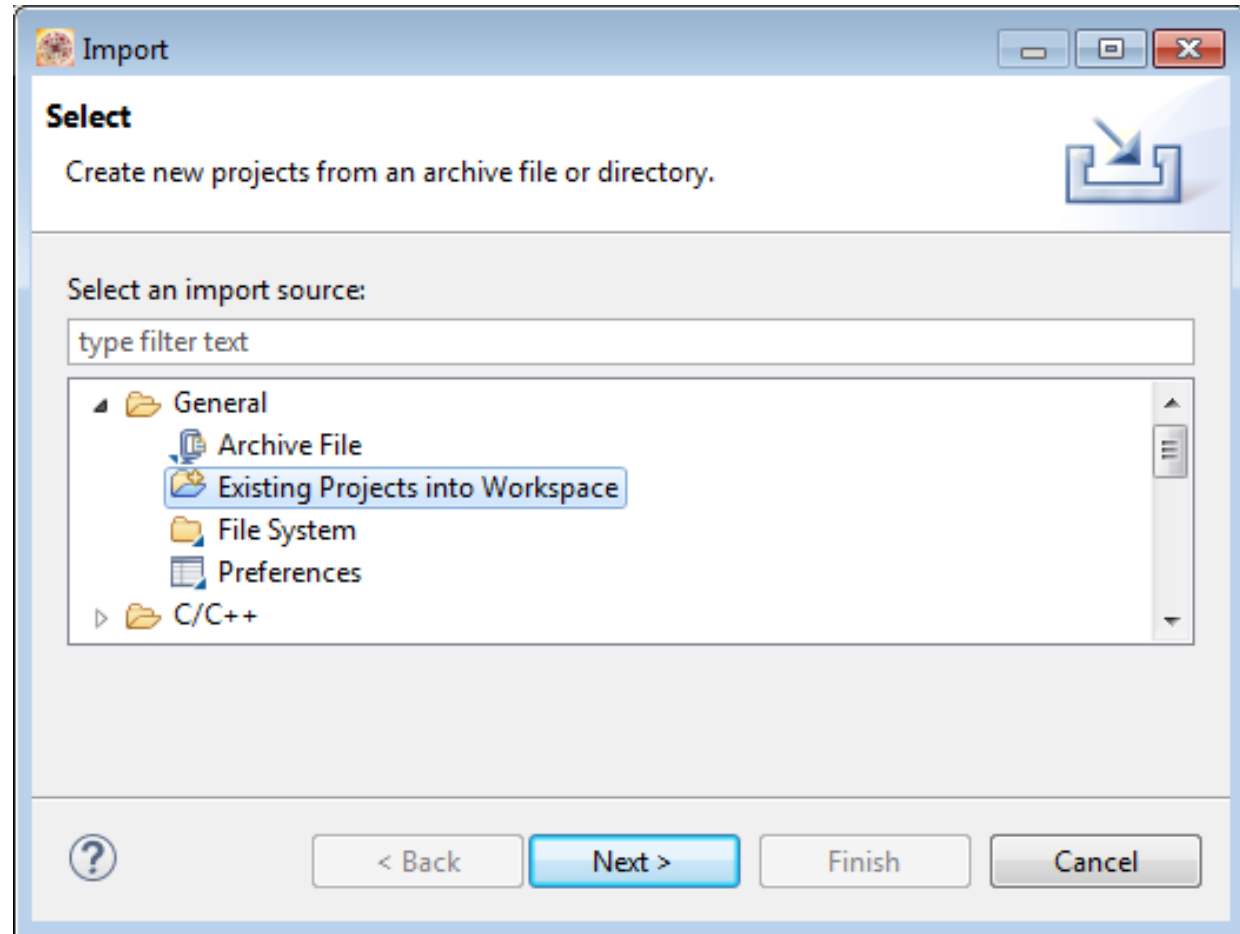
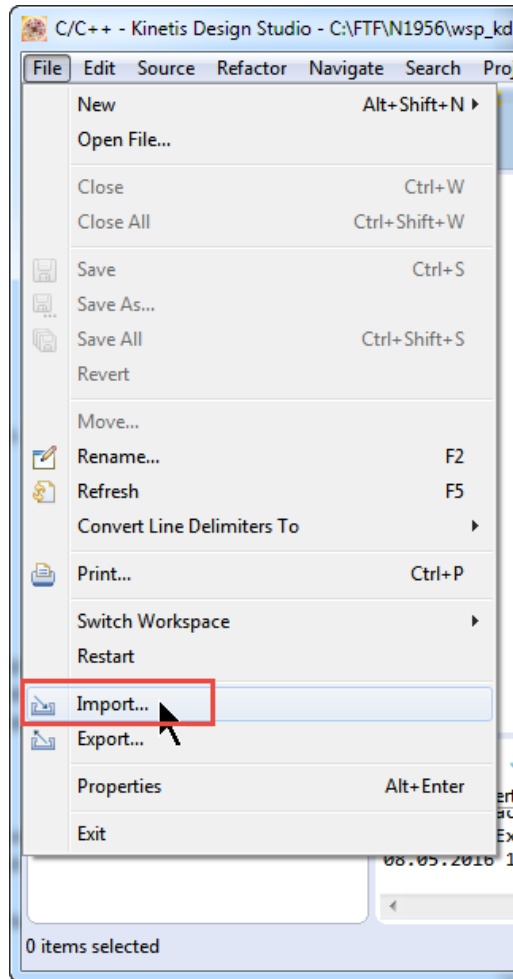
Kinetis Design Studio

- New Workspace, no projects listed
- Importing lab project in next step



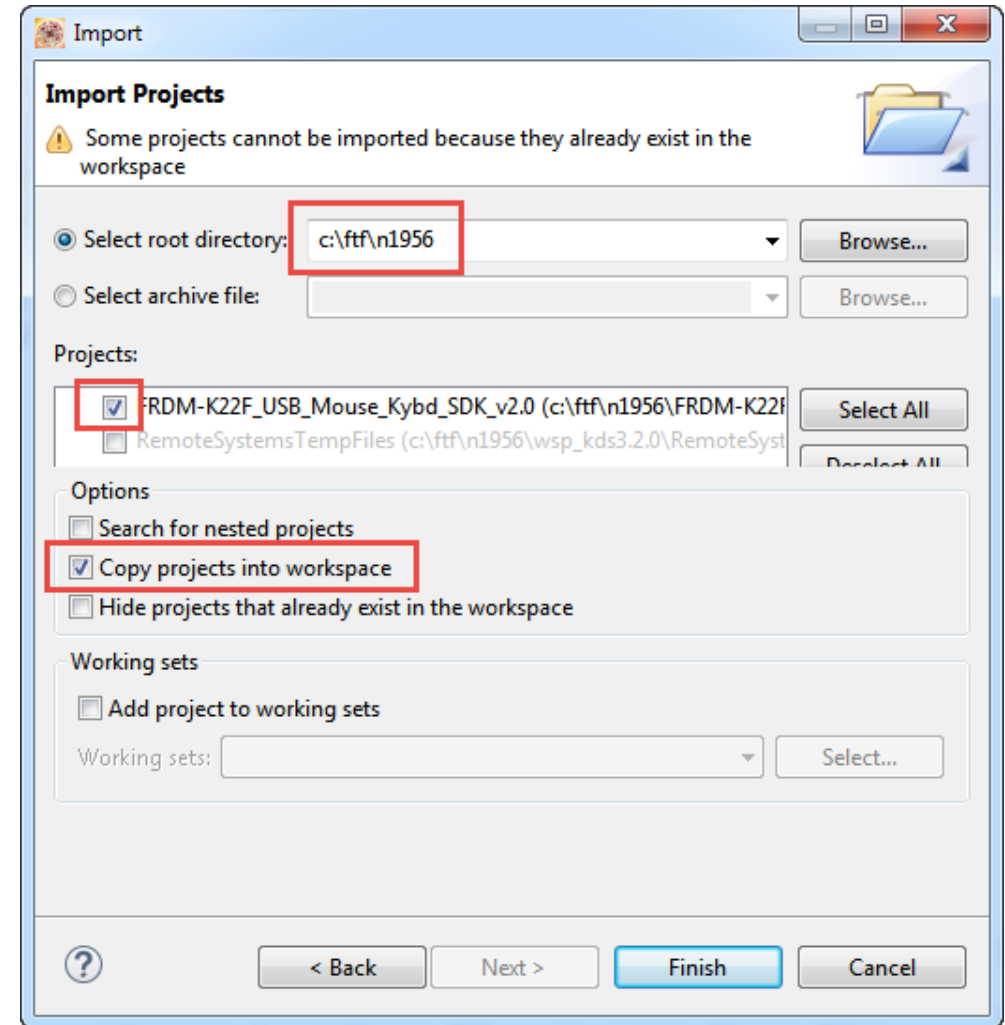
Importing Existing Project

Menu File > Import > General > Existing Projects into Workspace



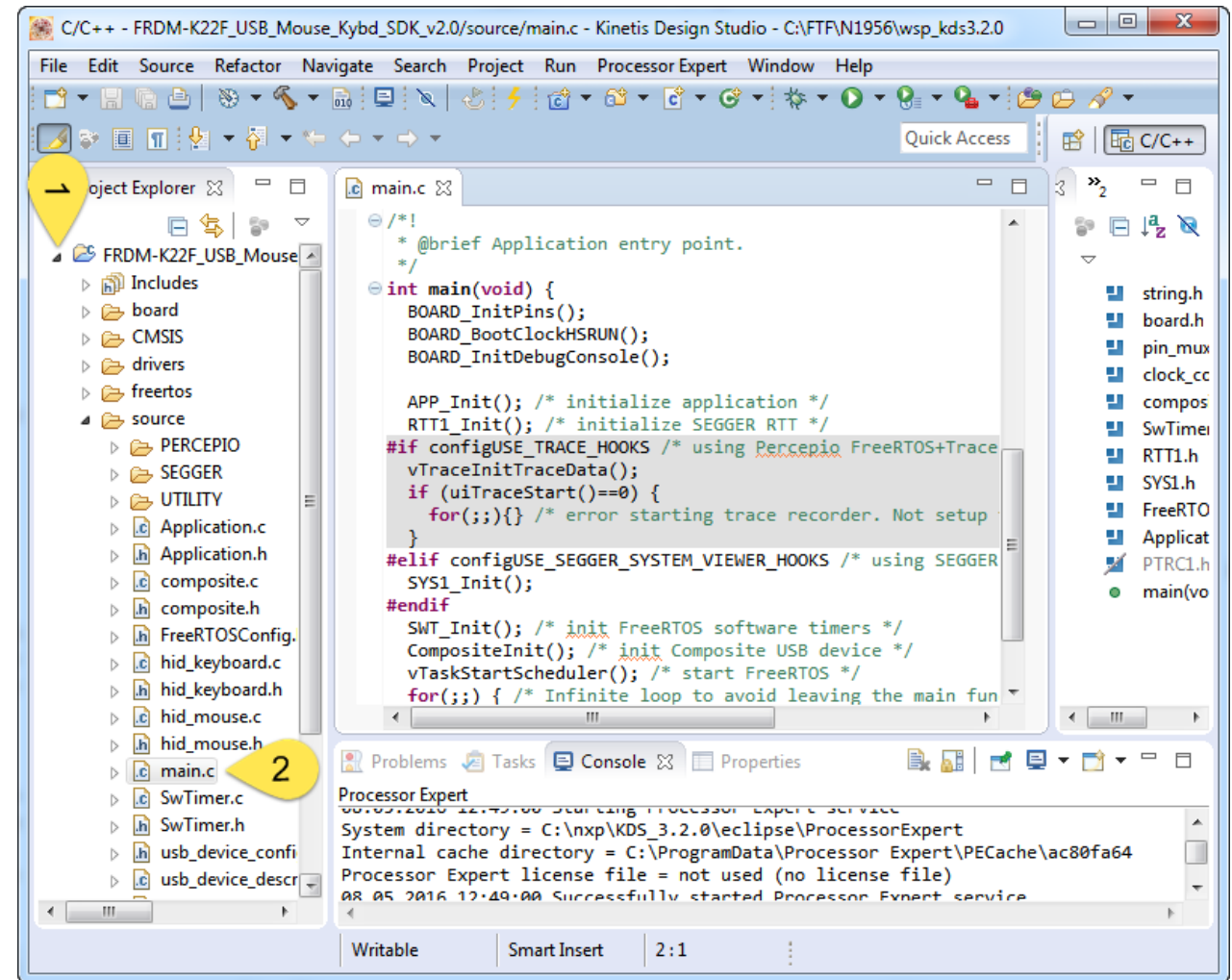
Importing Existing Project

- Enter C:\FTF\N1956
- Select project (FRDM-K22F_USB_...)
- Select Copy projects into workspace
- Press Finish to copy it

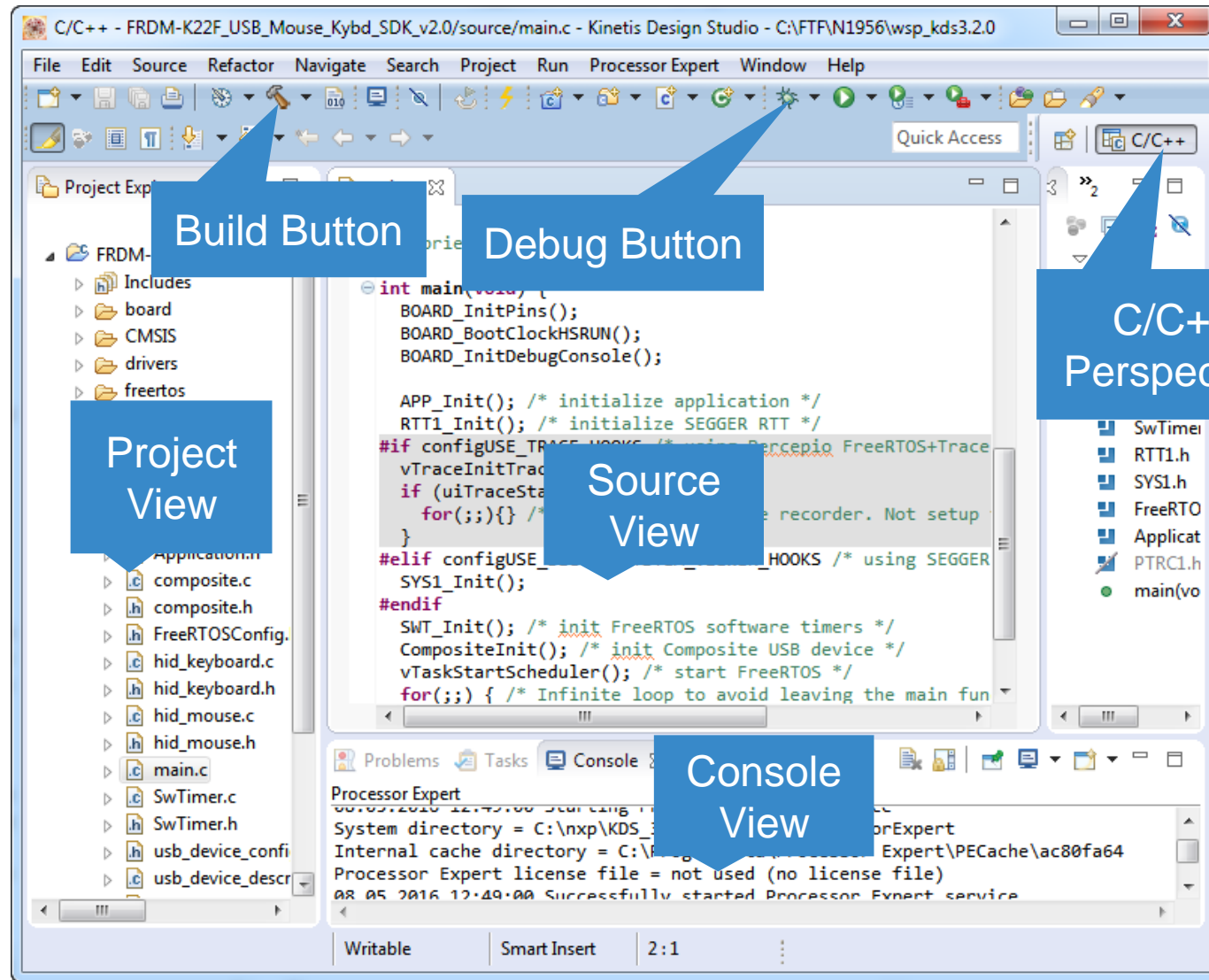


Editor View

- Expand project folder
- Double click on main to open it in editor view

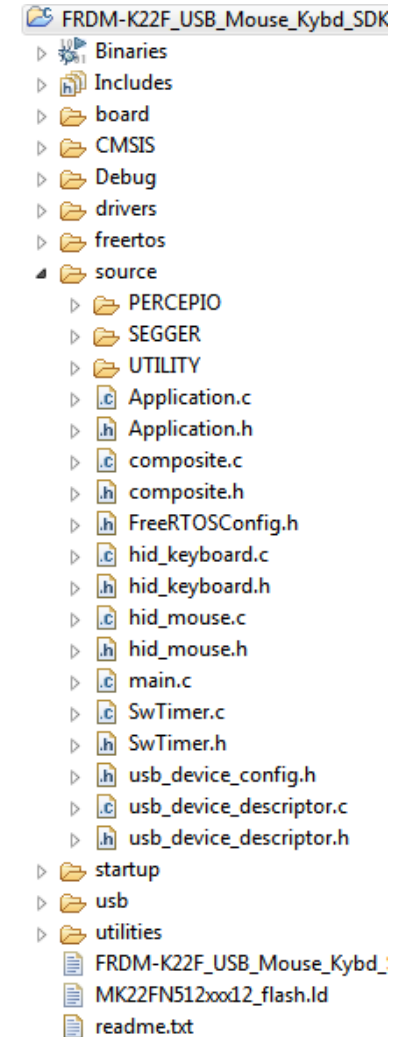


Kinetis Design Studio IDE Views



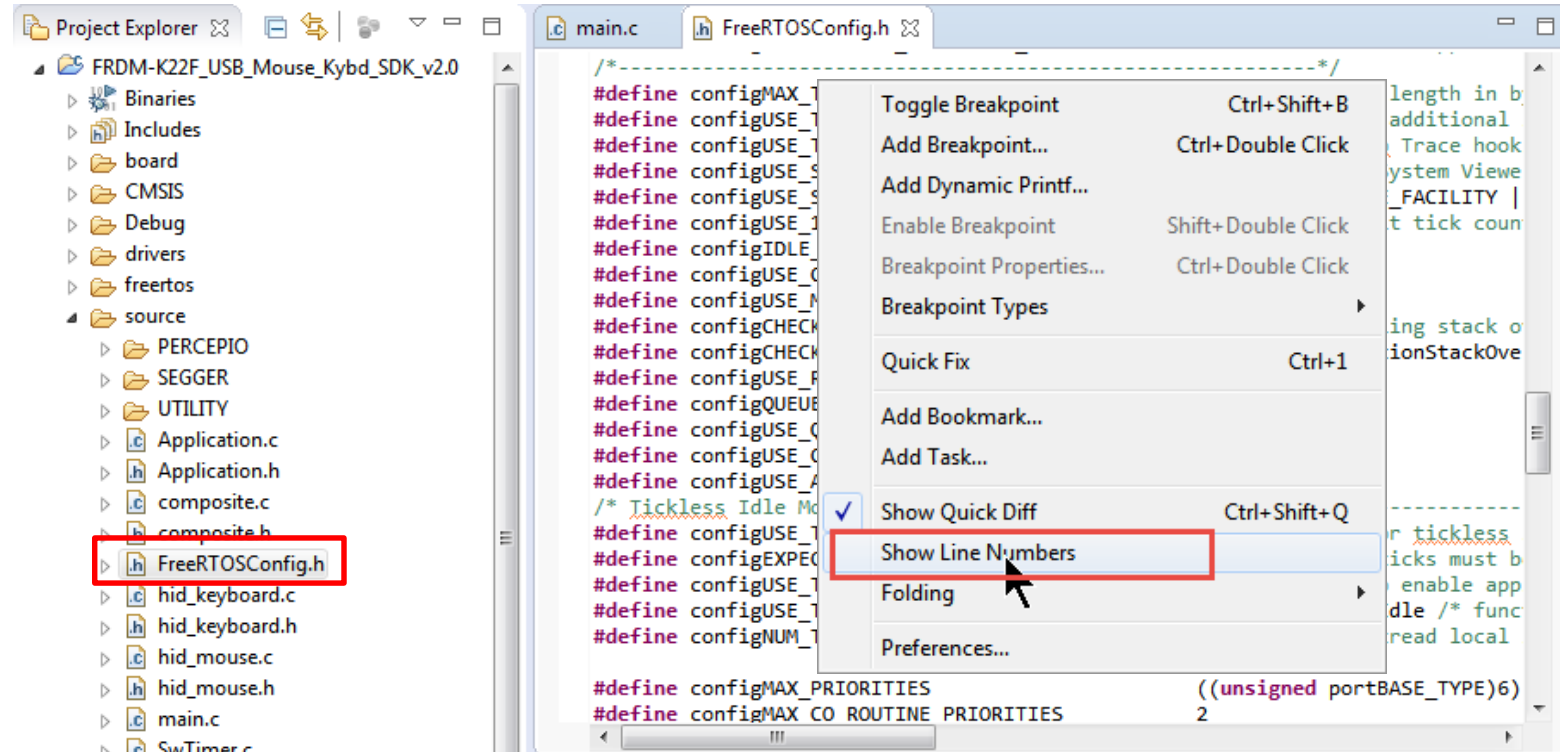
Application Project Structure

- **board, CMSIS & drivers:** Kinetis SDK
- **freertos:** RTOS
- **source:** Application sources
 - **PERCPIO:** Percepio FreeRTOS+Trace sources
 - **SEGGER:** Segger SystemView and RTT sources
 - **UTILITY:** helper routines
- **startup:** SDK startup code
- **USB:** USB stack
- **utilities:** SDK utilities



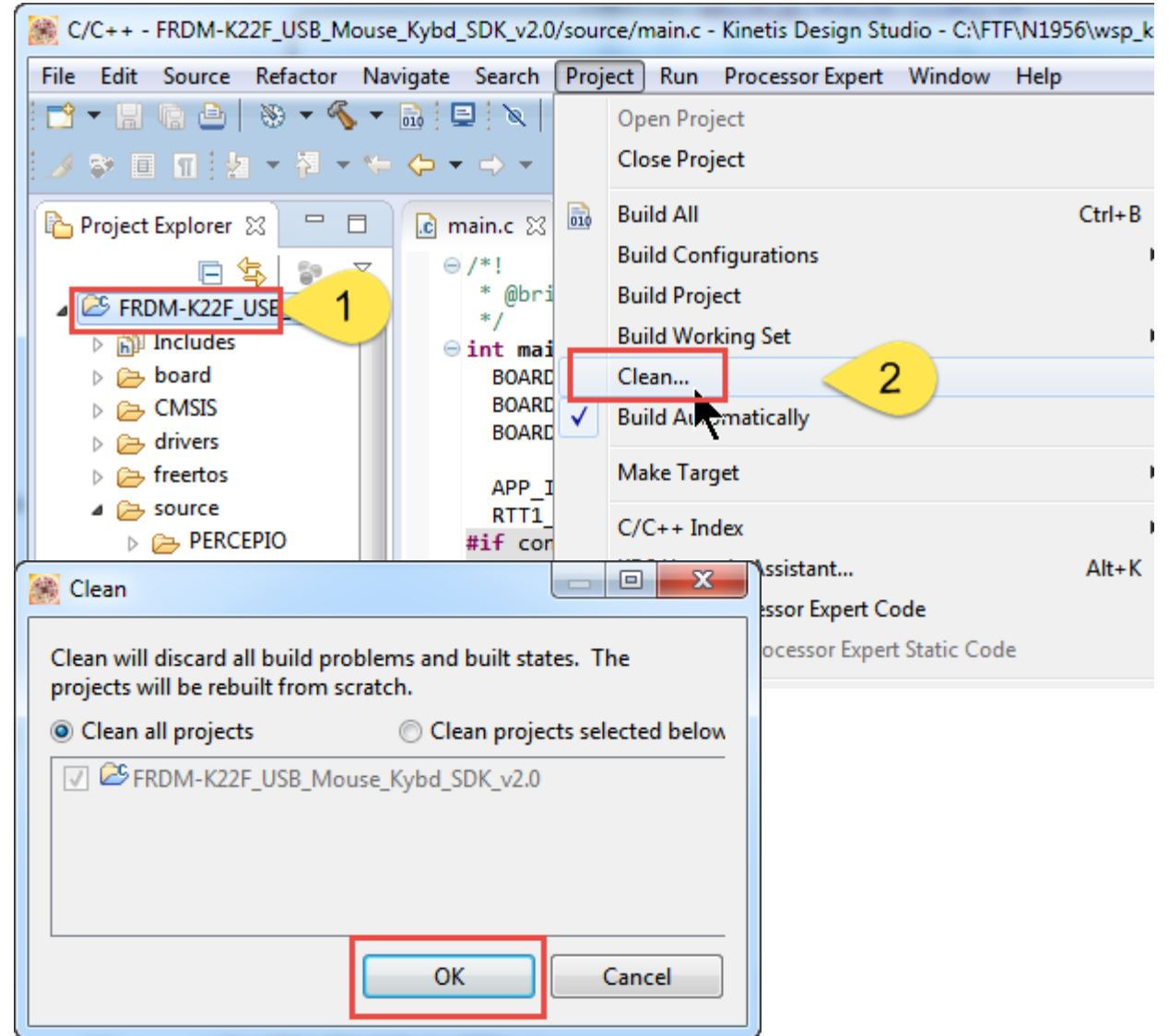
FreeRTOSConfig.h and Show Line Numbers

- RTOS Configuration File (FreeRTOSConfig.h)
- Will use it to turn on either Segger and Percepio Trace
- ➔ Show Line Numbers



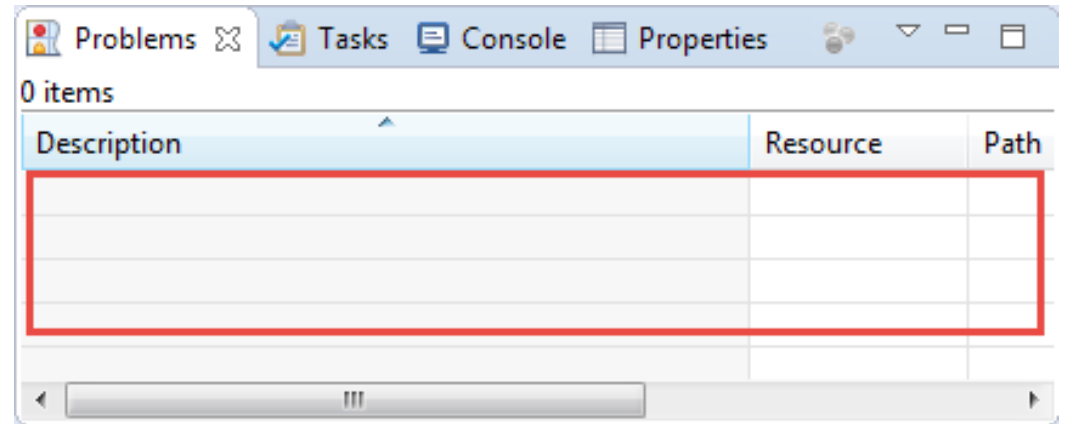
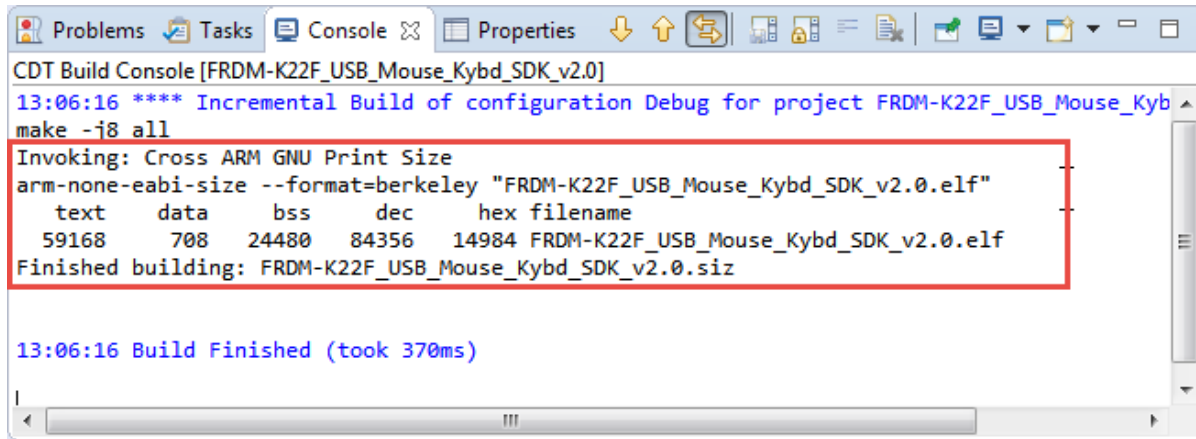
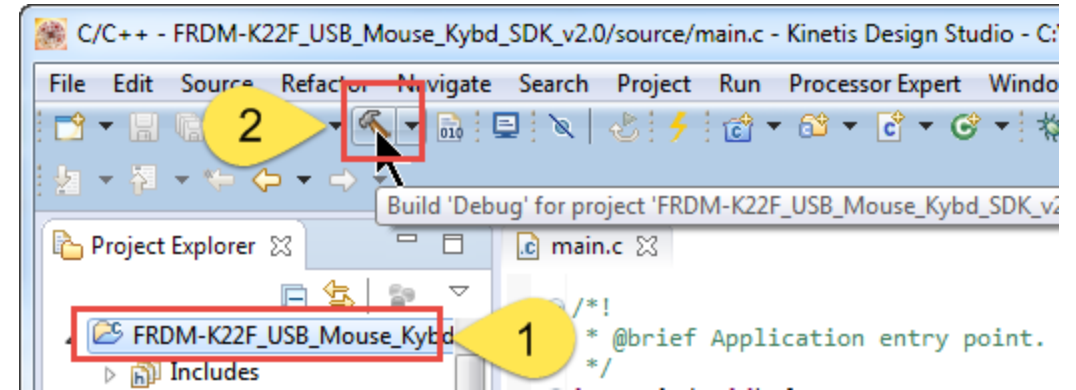
Clean Project

- After importing a project, a clean shall be done
 - Removes object files
 - Removes make files
- Perform a 'Clean'
 - Select project
 - Menu **Project > Clean...**
 - Press **OK** to clean



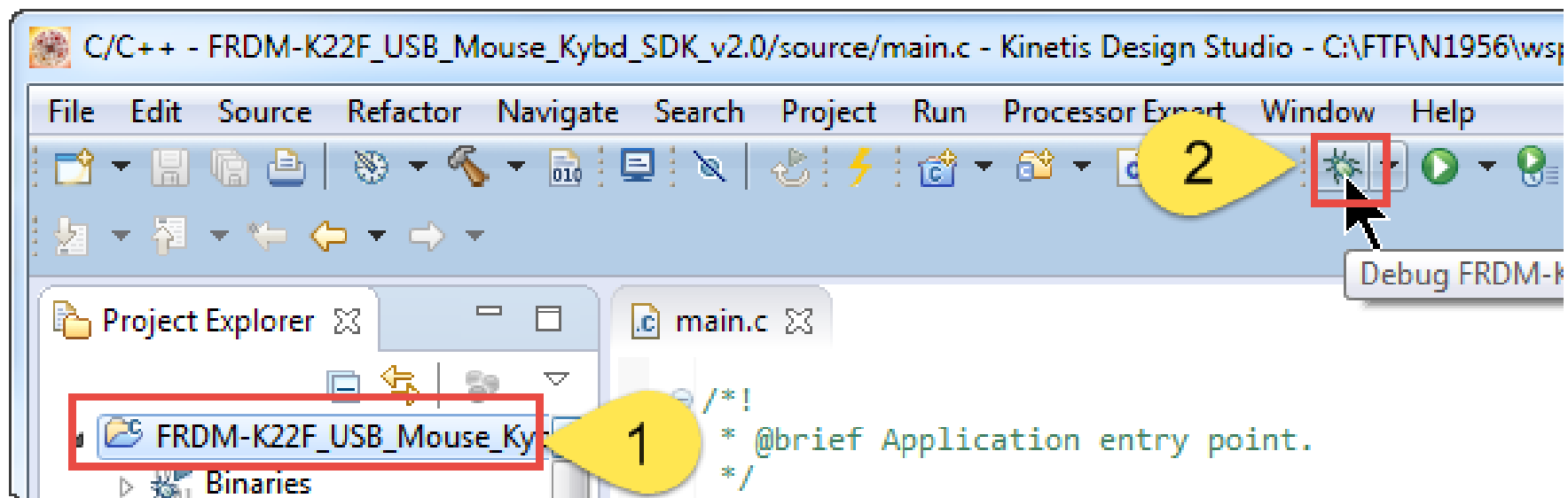
Build Project

- Select Project and Build
- Results
 - No errors in Problems view
 - Code/Data size reported in Console



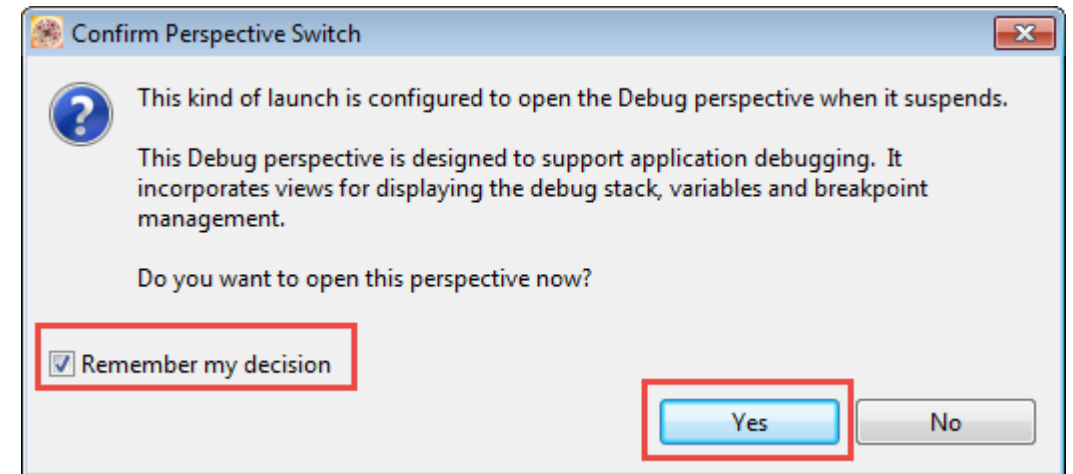
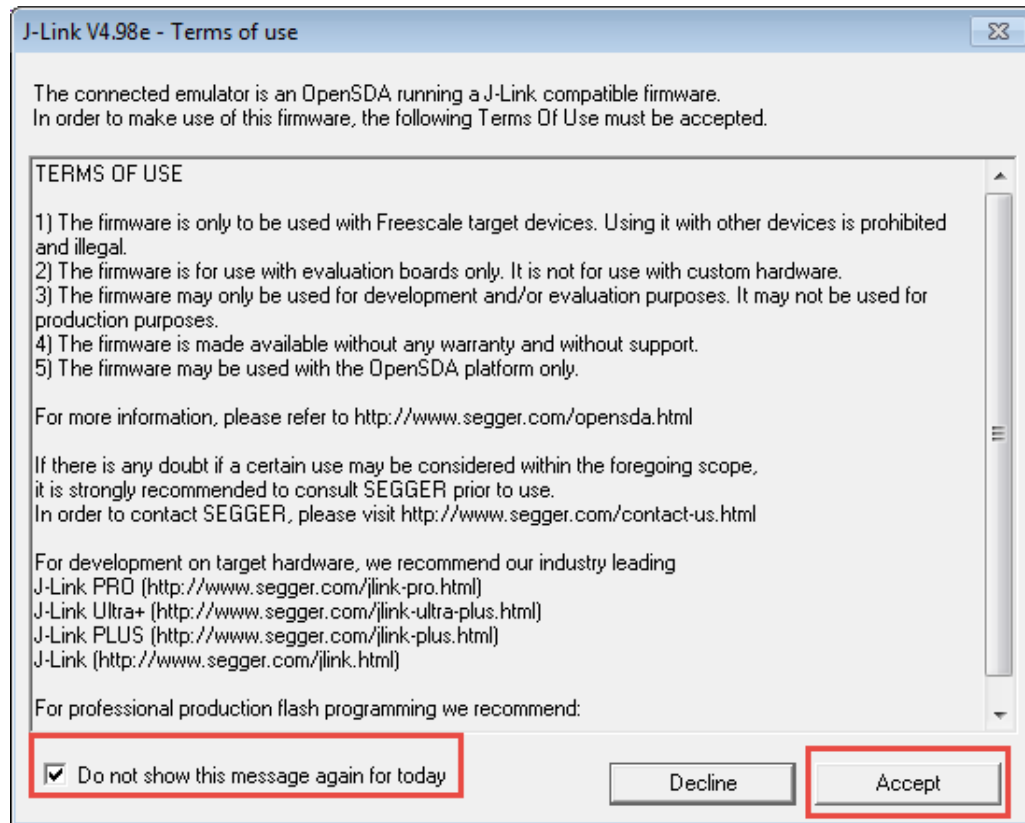
Debug Application

- Select **Project**
- Press **Debug Icon**



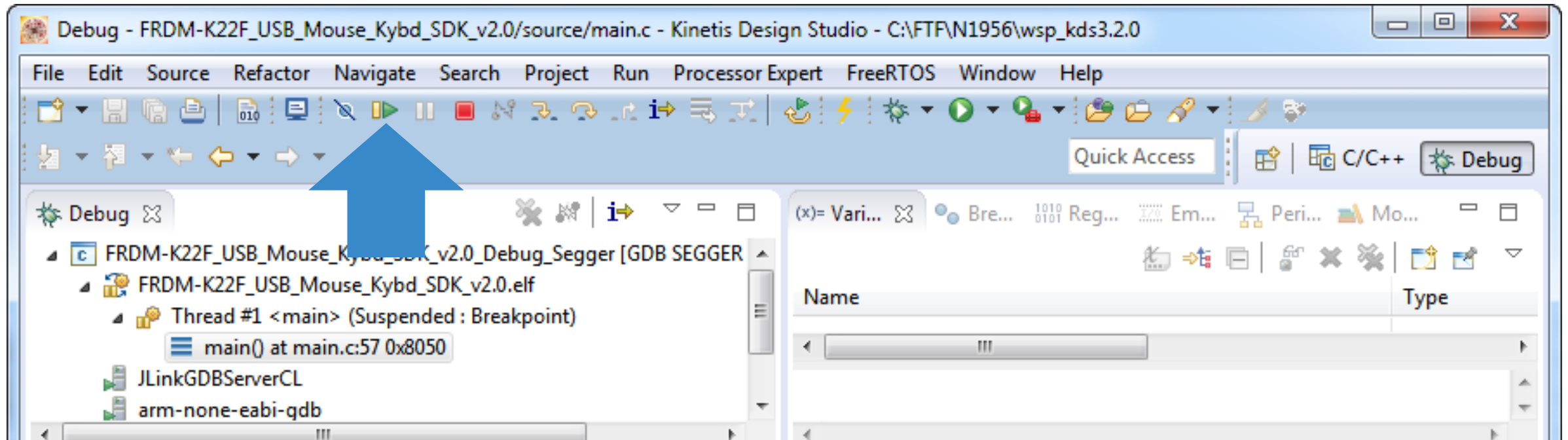
Segger Unlimited Flash Breakpoints

- J-Link Dialog to accept (enable checkbox and press Accept)
- Eclipse asks to switch perspective (enable checkbox and press Yes)



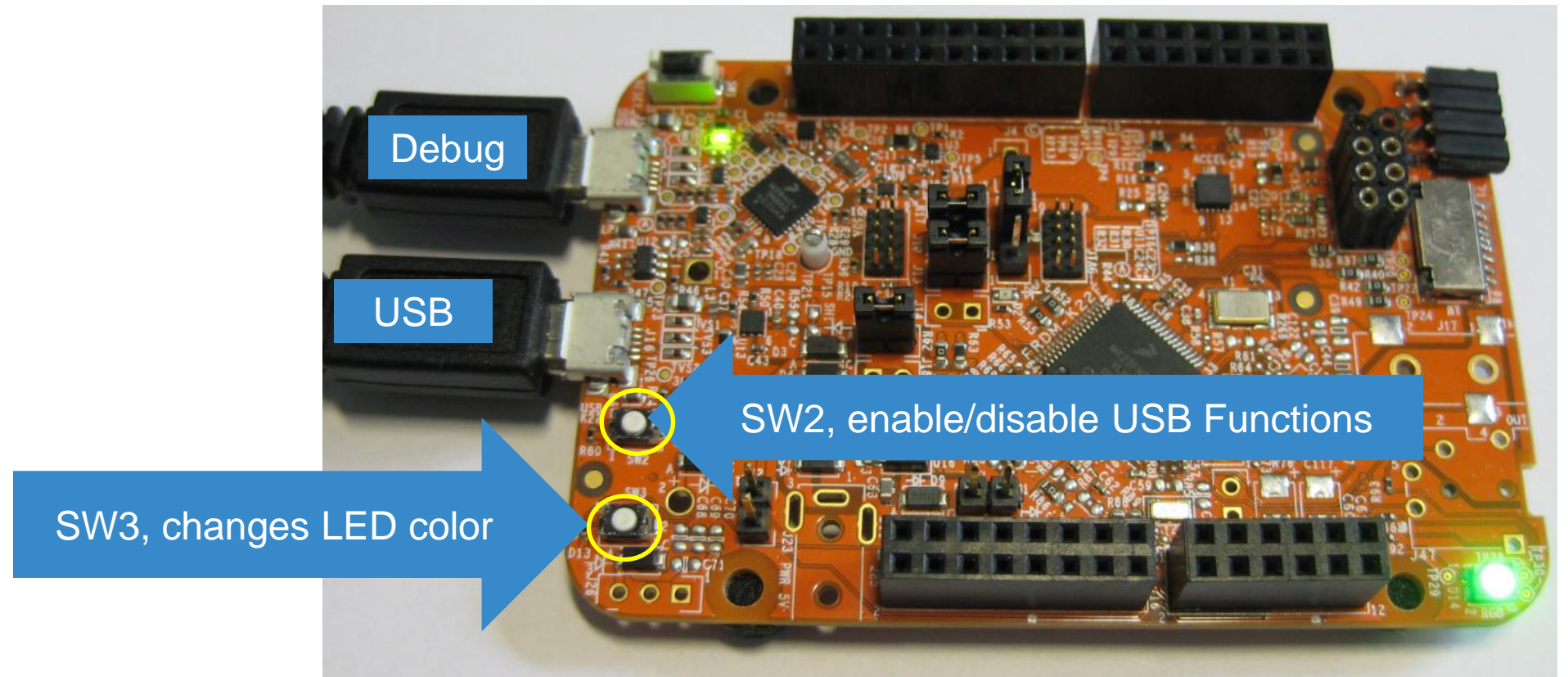
Debugging/Running Application

Run the application 



Running Lab Application

- Press SW3 to change RGB LED color
- Press SW2 to toggle USB functionality

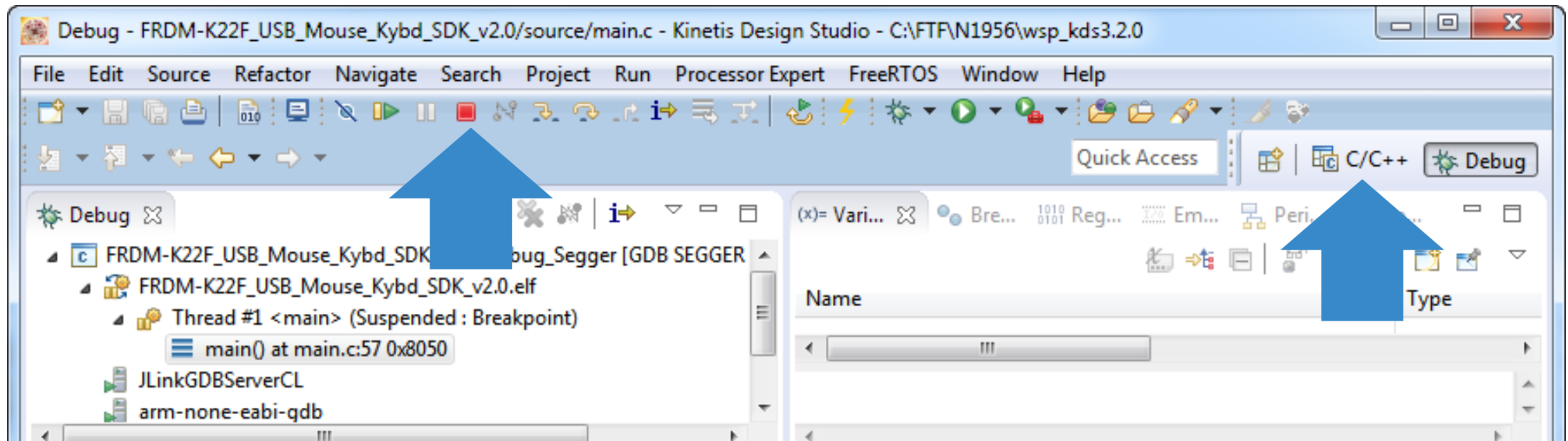


Debugging/Running Application

- Terminate Debug Session



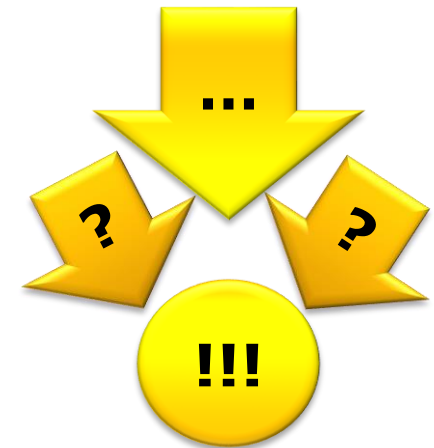
- Switch to C/C++ Perspective



Summary: Lab Setup

- Initial version of lab application running on board
- Kinetis Design Studio IDE: Eclipse based IDE for NXP Kinetis devices
- Kinetis SDK: Suite of peripheral drivers, stacks and middleware
- Application overview: high level functions and overview

- Further Information:
 - <http://www.nxp.com/kds>
 - <http://www.nxp.com/ksdk>
 - <http://www.nxp.com/freedom>



Lab: NXP FreeRTOS Kernel Awareness

Inspecting stack and runtime behavior of RTOS and Application

Outline: FreeRTOS Kernel Awareness

- **Problem**

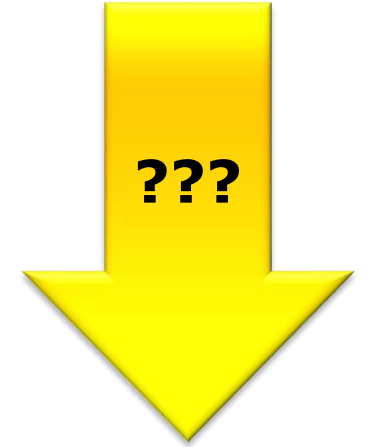
- Knowing tasks, queues, stack usage, system load

- **Solution**

- NXP FreeRTOS Kernel Awareness with GDB

- **Steps**

- Installation of Kernel Awareness plugins
- Inspect RTOS state



NXP FreeRTOS Eclipse Plugins

- NXP FreeRTOS Kernel Awareness plugins for Eclipse
- Views
 - Tasks
 - Stacks
 - Queues
 - Timers
 - Heap
- ‘Stop-Mode’ Views
 - Suspend debugger

COMING SOON



The screenshot shows the Eclipse IDE interface with the FreeRTOS plugin. The 'FreeRTOS' menu is open, displaying options: Task List, Stack Usage, Queue List, Timer List, Heap Usage, NXP Community, and About FreeRTOS TAD. Below the menu, three views are visible:

- Stack Usage (FreeRTOS):** A table showing task stack usage.

TCB#	Task Name	Task State	Stack Base	Stack Top	High Water Mark	High Water Mark Graph
3	Tmr Svc	Suspended	0x1fff19f8	0x1fff1c4c	20 B / 596 B	3.4%
2	IDLE	Ready	0x1fff1728	0x1fff1844	0 B / 284 B	0.0%
1	app task	Running	0x1fff0338	0x1fff1674	220 B / 4.81 kB	4.5%
- Task List (FreeRTOS):** A table showing task details.

TCB#	Task Name	Task State	Stack Usage	Priority	Event Object	Runtime
1	app task	Running	220 B / 4.81 kB (4.5%)	4		⚠
2	IDLE	Ready	0 B / 284 B (0.0%)	0		⚠
3	Tmr Svc	Suspended	20 B / 596 B (3.4%)	17	TmrQ (Rx)	⚠
- Heap Usage (FreeRTOS):** A table showing heap usage details.

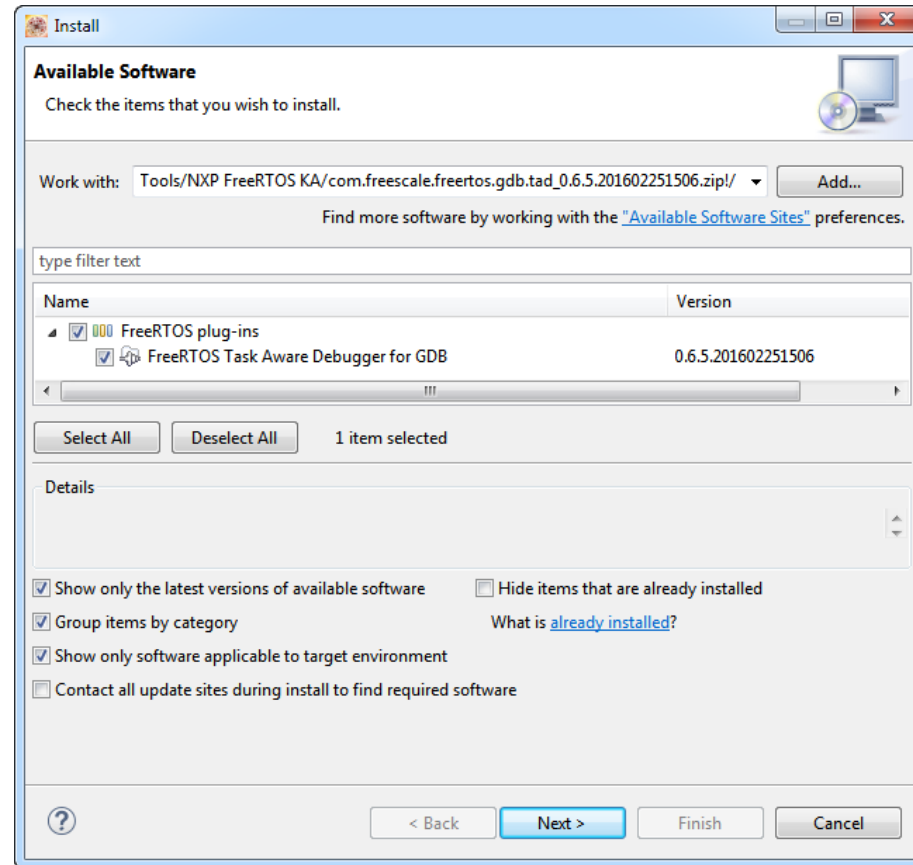
Type	Heap Base	Heap End	Heap Usage	Free Space	Heap Usage Graph
4	0x1fff032c	0x1fff8320	6.5 kB / 31.99 kB	79.7% (25.49 kB)	20.3% Used



Kernel Awareness Installation

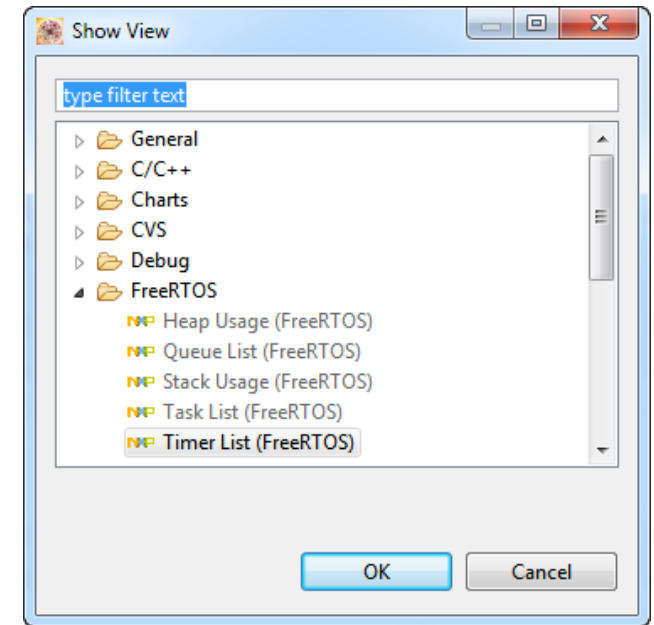
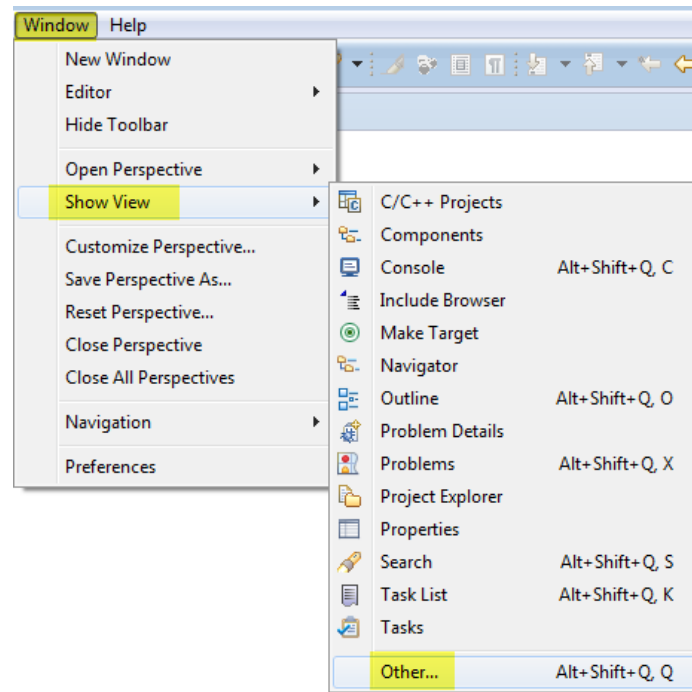
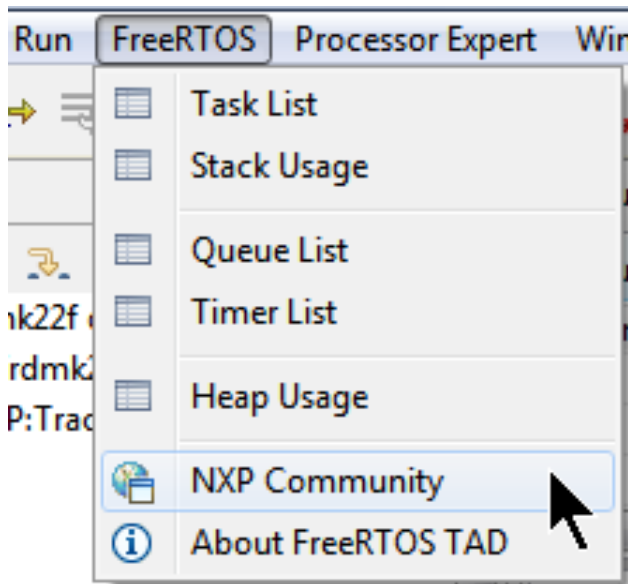
- Menu Help > Install New Software
- N1956/Tools/NXP FreeRTOS KA/* .zip

Pre-Installed on
Lab Computers



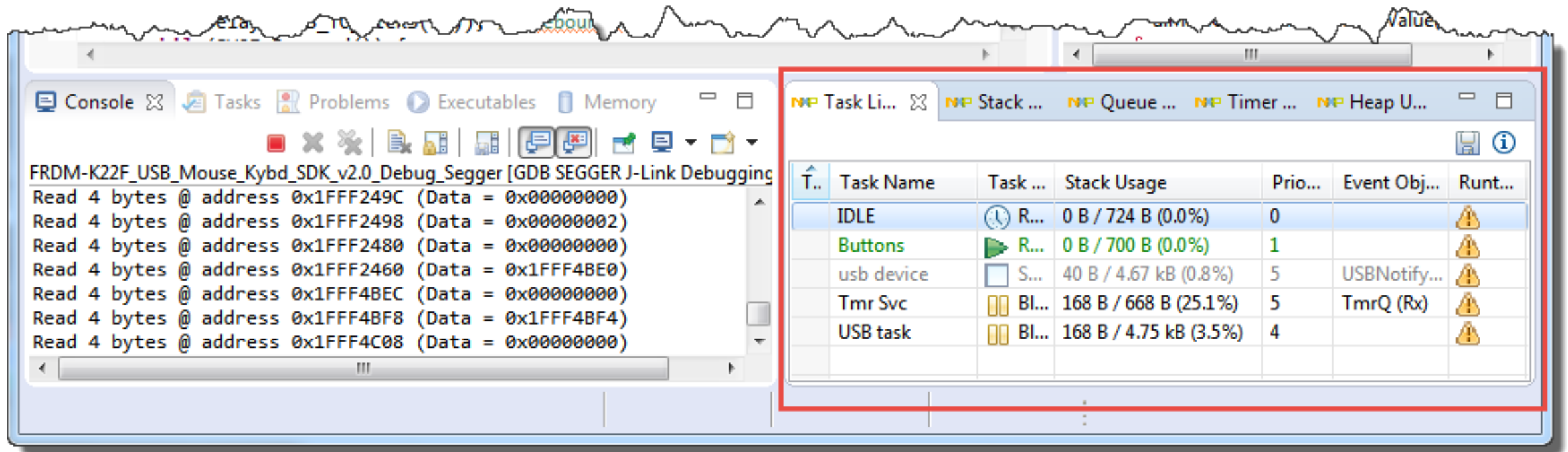
Open Views

- Press **Debug** → Debug Perspective
- Menu **FreeRTOS**
- Menu **Window > Show View > Other..**



Tips & Tricks: FreeRTOS View Arrangement

- Move FreeRTOS views to the side (Drag Tab to the side and drop)
- Own stack of views
- Does not interfere with console updates

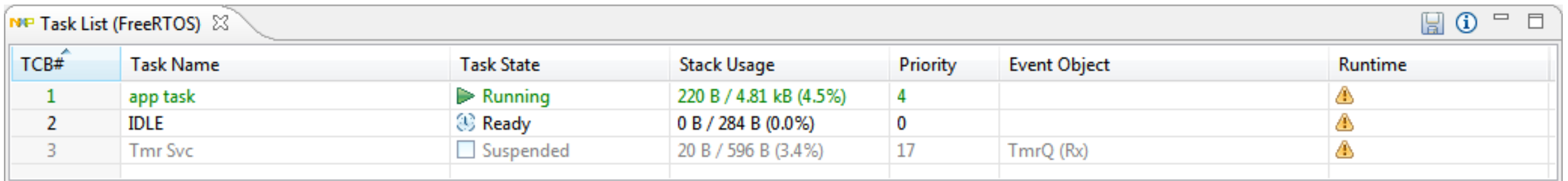


The screenshot shows the FreeRTOS view docked on the right side of the debugger interface. The view displays a table of task information, including task names, stack usage, priorities, and event objects. The table is highlighted with a red border.

Task Name	Task ...	Stack Usage	Prio...	Event Obj...	Runt...
IDLE	R...	0 B / 724 B (0.0%)	0		⚠
Buttons	R...	0 B / 700 B (0.0%)	1		⚠
usb device	S...	40 B / 4.67 kB (0.8%)	5	USBNotify...	⚠
Tmr Svc	Bl...	168 B / 668 B (25.1%)	5	TmrQ (Rx)	⚠
USB task	Bl...	168 B / 4.75 kB (3.5%)	4		⚠

Tasks

- Tasks with TCB (Task Control Block) number
- Task Name
- Task State
- Stack Usage
- Priority
- Event Object
- Runtime

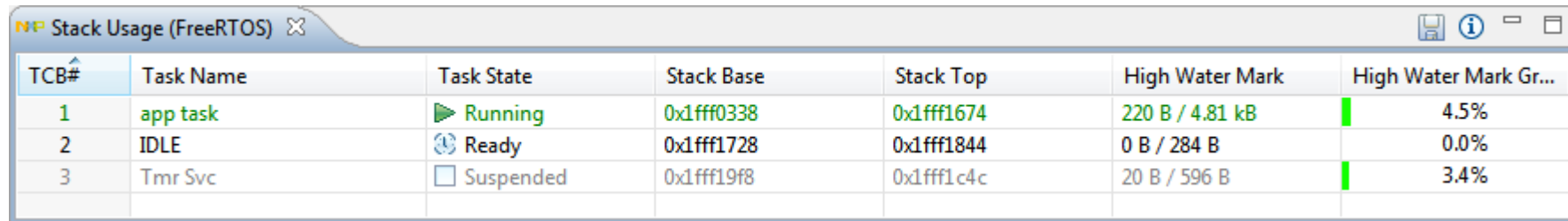


The screenshot shows a window titled "NXP Task List (FreeRTOS)" with a table containing the following data:

TCB#	Task Name	Task State	Stack Usage	Priority	Event Object	Runtime
1	app task	▶ Running	220 B / 4.81 kB (4.5%)	4		⚠
2	IDLE	🔄 Ready	0 B / 284 B (0.0%)	0		⚠
3	Tmr Svc	☐ Suspended	20 B / 596 B (3.4%)	17	TmrQ (Rx)	⚠

Stack Usage

- List of tasks
 - TCB
 - Task state
 - Stack memory location
 - High Water Marks
- Can save data as .csv

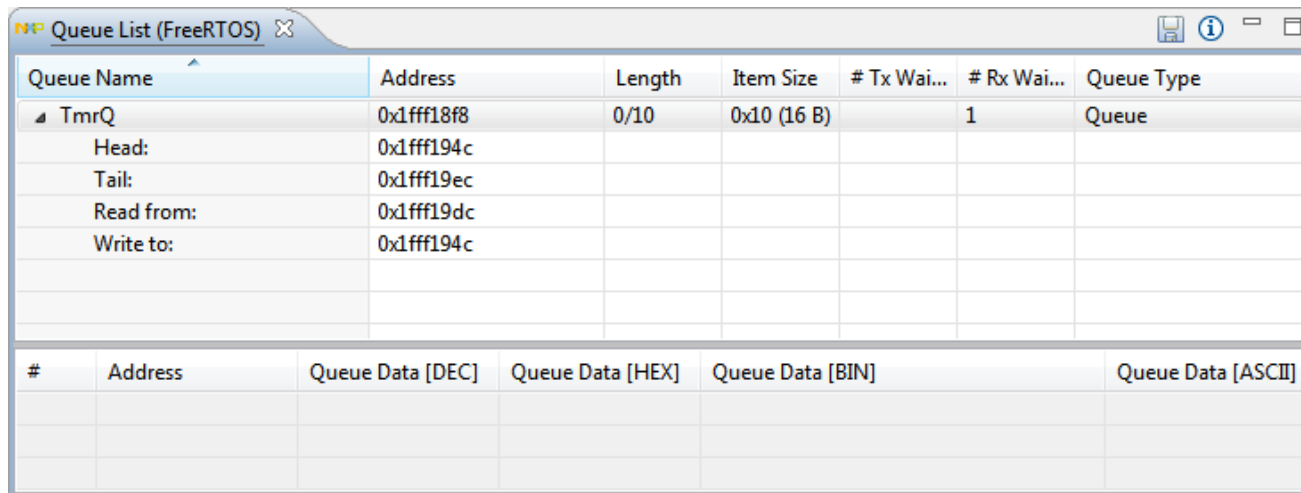


The screenshot shows a window titled "NXP Stack Usage (FreeRTOS)" with a table containing the following data:

TCB#	Task Name	Task State	Stack Base	Stack Top	High Water Mark	High Water Mark Gr...
1	app task	Running	0x1fff0338	0x1fff1674	220 B / 4.81 kB	4.5%
2	IDLE	Ready	0x1fff1728	0x1fff1844	0 B / 284 B	0.0%
3	Tmr Svc	Suspended	0x1fff19f8	0x1fff1c4c	20 B / 596 B	3.4%

Queue List

- configQUEUE_REGISTRY_SIZE
- Queue Information
 - Address, Length, Item Size
 - Tx and Rx waiting
 - Head/Tail
 - Read (out) and Write (in)

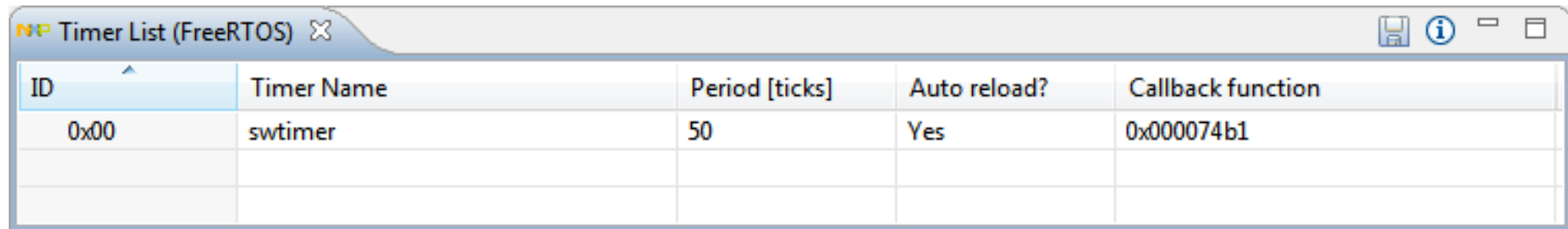


Queue Name	Address	Length	Item Size	# Tx Wai...	# Rx Wai...	Queue Type
▲ TmrQ	0x1fff18f8	0/10	0x10 (16 B)		1	Queue
Head:	0x1fff194c					
Tail:	0x1fff19ec					
Read from:	0x1fff19dc					
Write to:	0x1fff194c					

#	Address	Queue Data [DEC]	Queue Data [HEX]	Queue Data [BIN]	Queue Data [ASCII]

Timers

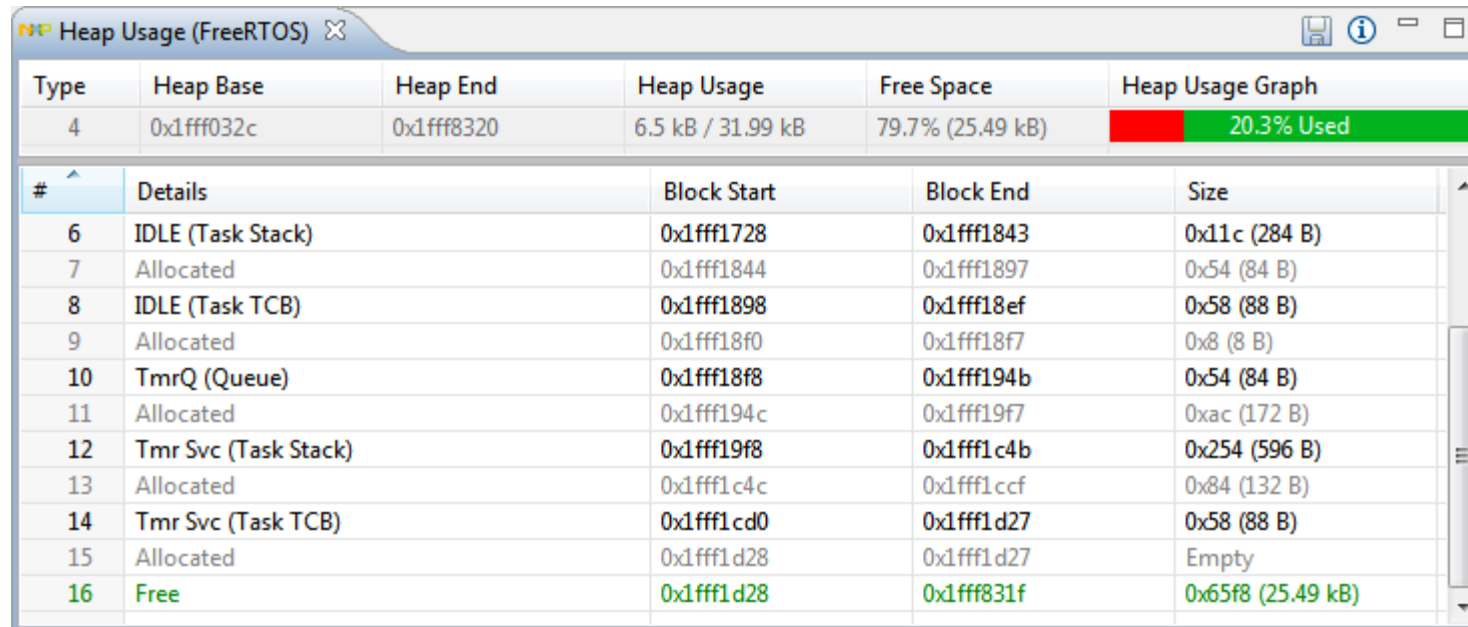
- View for FreeRTOS Software Timers
- Timer ID
- Timer name and period (ticks)
- Auto reload: timer gets restarted
- Address of timer callback function



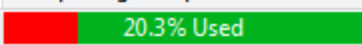
ID	Timer Name	Period [ticks]	Auto reload?	Callback function
0x00	swtimer	50	Yes	0x000074b1

Heap Usage

- Heap Type with address/size/usage information
- TCB (Task Control Block) and Task Stacks



The screenshot shows a window titled "NXP Heap Usage (FreeRTOS)". It contains a summary table and a detailed list of heap blocks. The summary table shows a total heap size of 31.99 kB, with 6.5 kB used (20.3%) and 25.49 kB free (79.7%). The detailed table lists 16 blocks, including task stacks, TCBs, and a free block.

Type	Heap Base	Heap End	Heap Usage	Free Space	Heap Usage Graph
4	0x1fff032c	0x1fff8320	6.5 kB / 31.99 kB	79.7% (25.49 kB)	

#	Details	Block Start	Block End	Size
6	IDLE (Task Stack)	0x1fff1728	0x1fff1843	0x11c (284 B)
7	Allocated	0x1fff1844	0x1fff1897	0x54 (84 B)
8	IDLE (Task TCB)	0x1fff1898	0x1fff18ef	0x58 (88 B)
9	Allocated	0x1fff18f0	0x1fff18f7	0x8 (8 B)
10	TmrQ (Queue)	0x1fff18f8	0x1fff194b	0x54 (84 B)
11	Allocated	0x1fff194c	0x1fff19f7	0xac (172 B)
12	Tmr Svc (Task Stack)	0x1fff19f8	0x1fff1c4b	0x254 (596 B)
13	Allocated	0x1fff1c4c	0x1fff1ccf	0x84 (132 B)
14	Tmr Svc (Task TCB)	0x1fff1cd0	0x1fff1d27	0x58 (88 B)
15	Allocated	0x1fff1d28	0x1fff1d27	Empty
16	Free	0x1fff1d28	0x1fff831f	0x65f8 (25.49 kB)

Lab: FreeRTOS Kernel Awareness

Knowing state of RTOS (Tasks, Timers, Queues, Stacks) with Eclipse



Outline: Stack Size

- **Problem**

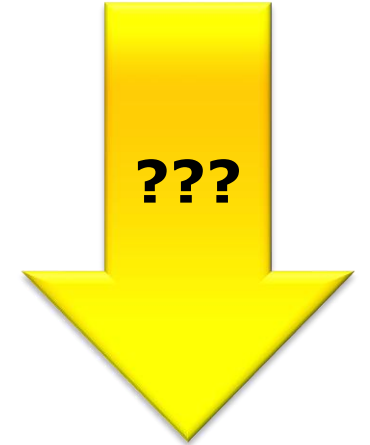
- Too much RAM/Heap used
- Check CPU load

- **Solution**

- Reduce amount of stack needed by stack
- Unblock CPU

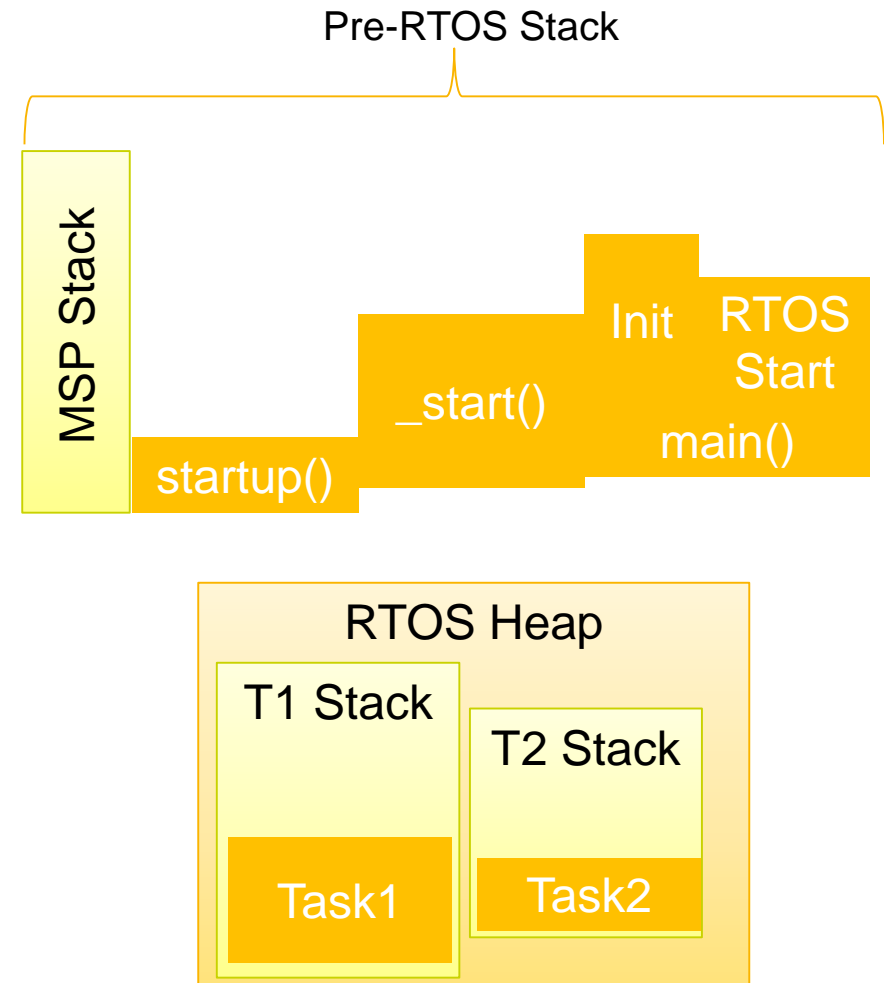
- **Steps**

- Inspect stack usage
- Fix Stack Size
- Free up CPU time



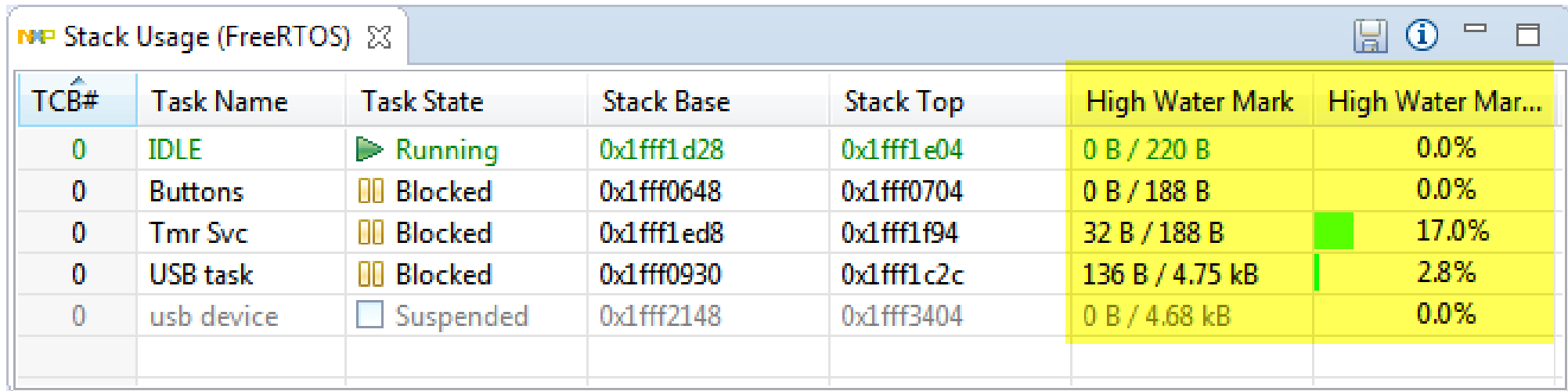
Heap/Stacks in FreeRTOS Application

- MSP (Mains Stack Pointer)
 - Startup code and scheduler start
 - Interrupts
- PSP (Process Stack Pointer)
 - configMINIMAL_STACK_SIZE for IDLE task
 - Number of 'addressable/aligned stack units' (Kineticis: 4 bytes!)



FreeRTOS Task Stack Usage

- High Water Mark for Stack
- Mark: set at context switch
- Everything looking good for you?

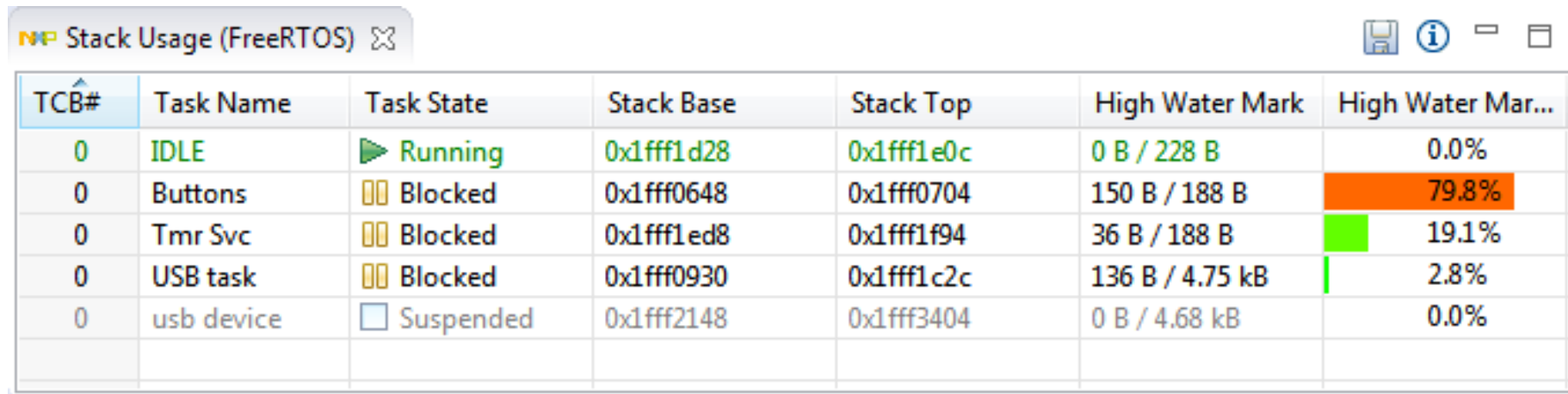


The screenshot shows a window titled "NXP Stack Usage (FreeRTOS)" with a table of task stack usage. The table has seven columns: TCB#, Task Name, Task State, Stack Base, Stack Top, High Water Mark, and High Water Mar... (truncated). The rows represent different tasks: IDLE (Running), Buttons (Blocked), Tmr Svc (Blocked), USB task (Blocked), and usb device (Suspended). The High Water Mark column shows the current usage and the total available stack space for each task. For example, the USB task has used 136 B out of 4.75 kB.

TCB#	Task Name	Task State	Stack Base	Stack Top	High Water Mark	High Water Mar...
0	IDLE	▶ Running	0x1fff1d28	0x1fff1e04	0 B / 220 B	0.0%
0	Buttons	⏸ Blocked	0x1fff0648	0x1fff0704	0 B / 188 B	0.0%
0	Tmr Svc	⏸ Blocked	0x1fff1ed8	0x1fff1f94	32 B / 188 B	17.0%
0	USB task	⏸ Blocked	0x1fff0930	0x1fff1c2c	136 B / 4.75 kB	2.8%
0	usb device	⏸ Suspended	0x1fff2148	0x1fff3404	0 B / 4.68 kB	0.0%

Higher Stack Usage

- Press SW2 or SW3
- Stack usage increase and gets critical
- Increase Stack size in Application.c



TCB#	Task Name	Task State	Stack Base	Stack Top	High Water Mark	High Water Mar...
0	IDLE	▶ Running	0x1fff1d28	0x1fff1e0c	0 B / 228 B	0.0%
0	Buttons	⏸ Blocked	0x1fff0648	0x1fff0704	150 B / 188 B	79.8%
0	Tmr Svc	⏸ Blocked	0x1fff1ed8	0x1fff1f94	36 B / 188 B	19.1%
0	USB task	⏸ Blocked	0x1fff0930	0x1fff1c2c	136 B / 4.75 kB	2.8%
0	usb device	⏸ Suspended	0x1fff2148	0x1fff3404	0 B / 4.68 kB	0.0%

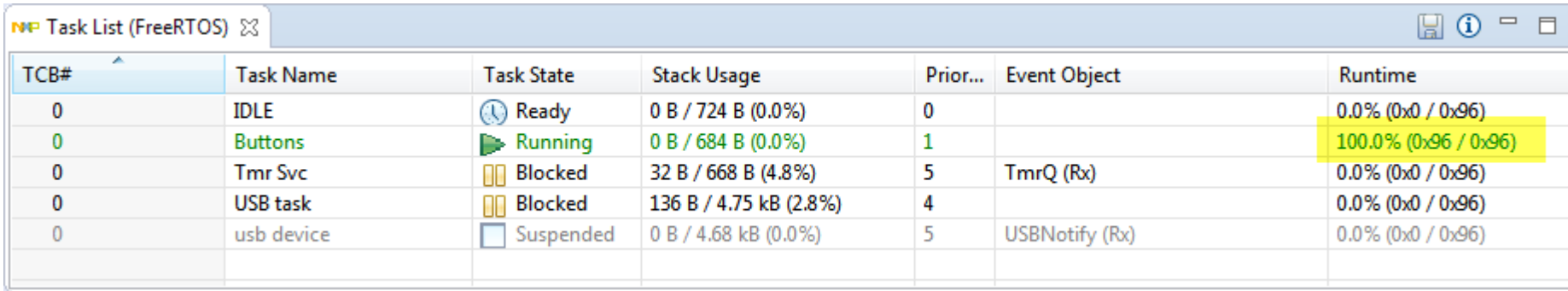
```
81 vQueueAddToRegistry(semSW3, "SW3_Sem");
82 if (xTaskCreate(ButtonTask, "Buttons", configMINIMAL_STACK_SIZE+50, NULL, tskIDLE_PRIORITY+1, NULL) != pdPASS) {
83     for(;;){} /* error */
84 }
85 }
```


Enabling Runtime Statistics

- Set to 1 in FreeRTOS_Config.h:
 - `configGENERATE_RUN_TIME_STATS` (line 125, uses SysTick)
 - `configUSE_TRACE_FACILITY` (line 164, additional structure members for execution visualization and tracing)
- Build
 - Adds about 100 bytes 'text' and 8 bytes 'bss'
- Debug
 - Inspect runtime

Runtime Problem

Problem: Buttons task is using 100% of the CPU



TCB#	Task Name	Task State	Stack Usage	Prior...	Event Object	Runtime
0	IDLE	Ready	0 B / 724 B (0.0%)	0		0.0% (0x0 / 0x96)
0	Buttons	Running	0 B / 684 B (0.0%)	1		100.0% (0x96 / 0x96)
0	Tmr Svc	Blocked	32 B / 668 B (4.8%)	5	TmrQ (Rx)	0.0% (0x0 / 0x96)
0	USB task	Blocked	136 B / 4.75 kB (2.8%)	4		0.0% (0x0 / 0x96)
0	usb device	Suspended	0 B / 4.68 kB (0.0%)	5	USBNotify (Rx)	0.0% (0x0 / 0x96)

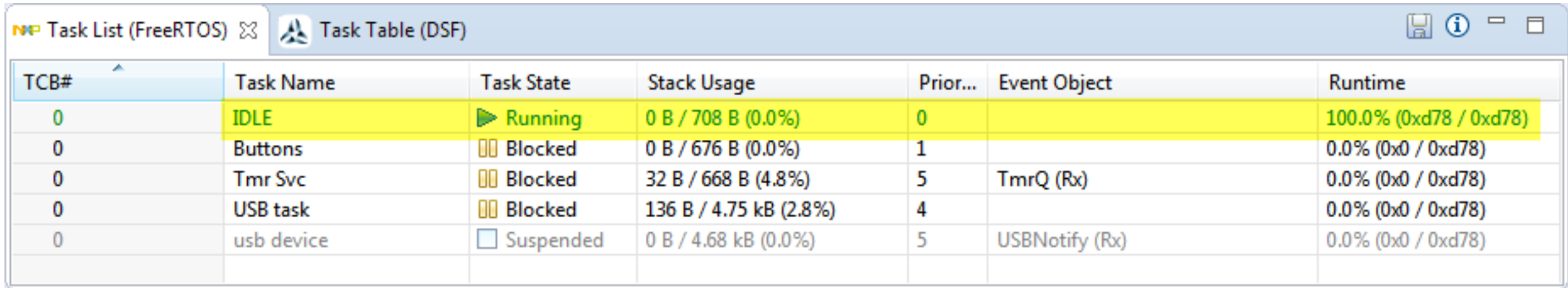
Solution

- Add **delay** to ButtonTask() in Application.c
- `vTaskDelay(pdMS_TO_TICKS(10));`
- Save, Build and re-Run






```
Application.c  FreeRTOSConfig.h
33 static void ButtonTask(void *pvParameters) {
34     (void)pvParameters;
35
36     for(;;) {
37         if (SW2IsPressed()) { /* SW2 pressed */
38             vTaskDelay(pdMS_TO_TICKS(50)); /* debounce */
39             while(SW2IsPressed()) {
40                 /* wait until released */
41             }
42             (void)xSemaphoreGive(semSW2); /* send message */
43         }
44         if (SW3IsPressed()) { /* SW3 pressed */
45             vTaskDelay(pdMS_TO_TICKS(50)); /* debounce */
46             while(SW3IsPressed()) {
47                 /* wait until released */
48             }
49             (void)xSemaphoreGive(semSW3); /* send message */
50         }
51         vTaskDelay(pdMS_TO_TICKS(10));
52     }
53 }
```

Improved Runtime

System is now mostly IDLE which is expected

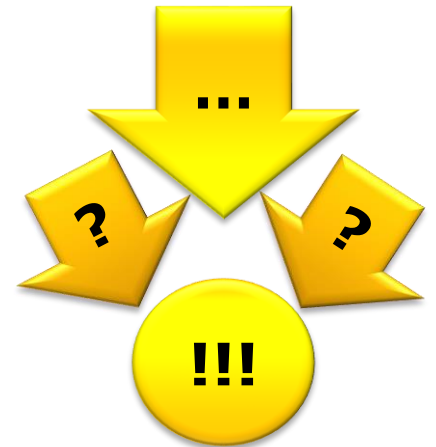


The screenshot shows a window titled 'Task Table (DSF)' with a table of task information. The table has seven columns: TCB#, Task Name, Task State, Stack Usage, Prior..., Event Object, and Runtime. The 'IDLE' task is highlighted in yellow and is in a 'Running' state, occupying 100.0% of the runtime. Other tasks include 'Buttons', 'Tmr Svc', 'USB task', and 'usb device', all in various states like 'Blocked' or 'Suspended'.

TCB#	Task Name	Task State	Stack Usage	Prior...	Event Object	Runtime
0	IDLE	 Running	0 B / 708 B (0.0%)	0		100.0% (0xd78 / 0xd78)
0	Buttons	 Blocked	0 B / 676 B (0.0%)	1		0.0% (0x0 / 0xd78)
0	Tmr Svc	 Blocked	32 B / 668 B (4.8%)	5	TmrQ (Rx)	0.0% (0x0 / 0xd78)
0	USB task	 Blocked	136 B / 4.75 kB (2.8%)	4		0.0% (0x0 / 0xd78)
0	usb device	 Suspended	0 B / 4.68 kB (0.0%)	5	USBNotify (Rx)	0.0% (0x0 / 0xd78)

Summary: FreeRTOS Kernel Awareness

- Eclipse Plug-in to show FreeRTOS information
 - Tasks
 - Queues
 - Timers
- Stack used
 - Startup, Tasks + Interrupts
- Default stack size
 - IDLE Task
 - Task Stacks
- Further Information:
 - <http://www.freertos.org>
 - <http://mcuoneclipse.com/2013/08/04/diy-free-toolchain-for-kinetis-part-5-freertos-eclipse-kernel-awareness-with-gdb/>
 - <http://mcuoneclipse.com/2014/11/08/tutorial-freertos-with-the-kinetis-sdk-and-processor-expert/>



Lab: Segger SystemView

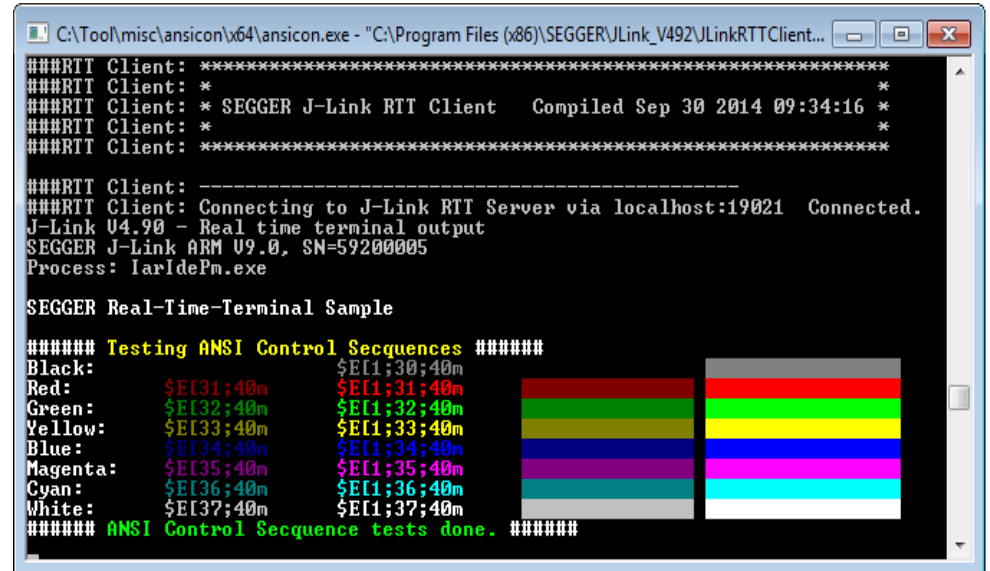
Inspecting runtime behavior of RTOS and Application



Segger Real Time Transfer

- Bi-directional data transfer through debug connection (Segger Real Time Transfer), multiple named up and down named channels
- Requires Segger J-Link debug probe or debug firmware (e.g. NXP OpenSDA)
- Small footprint: 500 Bytes Flash, 50 Bytes pro channel (plus buffer size, 64-1024 bytes).
- Special viewers by Segger, or standard Telnet

[\(http://mcuoneclipse.com/2015/07/07/\)](http://mcuoneclipse.com/2015/07/07/)



```
##### Client: *****
##### Client: *
##### Client: * SEGGER J-Link RTT Client   Compiled Sep 30 2014 09:34:16 *
##### Client: *
##### Client: *****

##### Client: -----
##### Client: Connecting to J-Link RTT Server via localhost:19021 Connected.
J-Link V4.90 - Real time terminal output
SEGGER J-Link ARM V9.0, SN=59200005
Process: IarIdePm.exe

SEGGER Real-Time-Terminal Sample

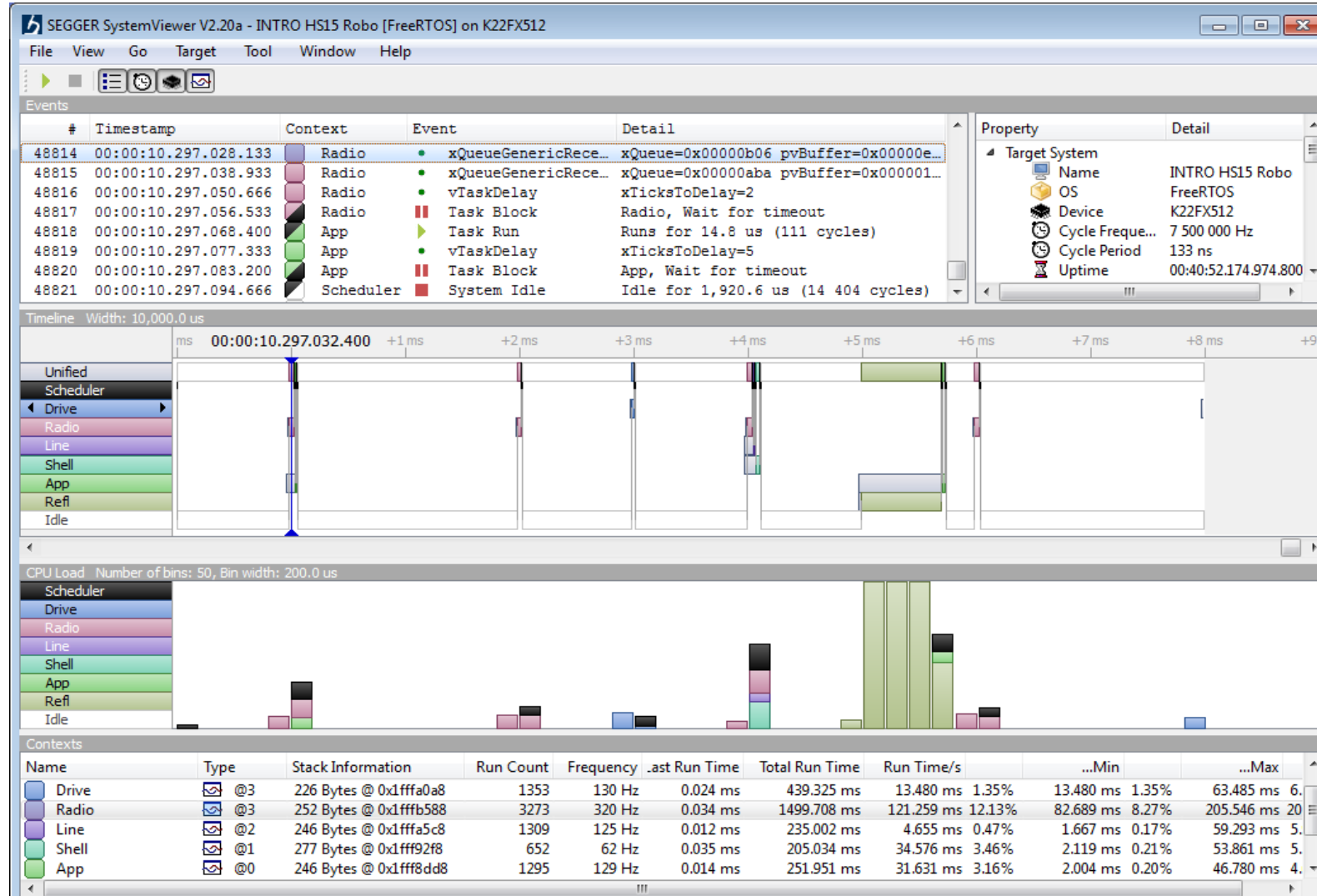
##### Testing ANSI Control Sequences #####
Black:      $E[1;30;40m      $E[1;30;40m
Red:        $E[1;31;40m      $E[1;31;40m
Green:     $E[1;32;40m      $E[1;32;40m
Yellow:    $E[1;33;40m      $E[1;33;40m
Blue:      $E[1;34;40m      $E[1;34;40m
Magenta:   $E[1;35;40m      $E[1;35;40m
Cyan:      $E[1;36;40m      $E[1;36;40m
White:     $E[1;37;40m      $E[1;37;40m
##### ANSI Control Sequence tests done. #####
```

FreeRTOS Trace Hooks

- Trace Hooks in the Kernel (task creation, context switch, ...)
- Implemented with macros (instrumented), empty by default
- Can be 'filled' with recording kernel activities
- configUSE_TRACE_HOOKS adds additional information to kernel objects
 - Custom recording and tracing
 - Percepio FreeRTOS+Trace
 - Segger SystemViewer

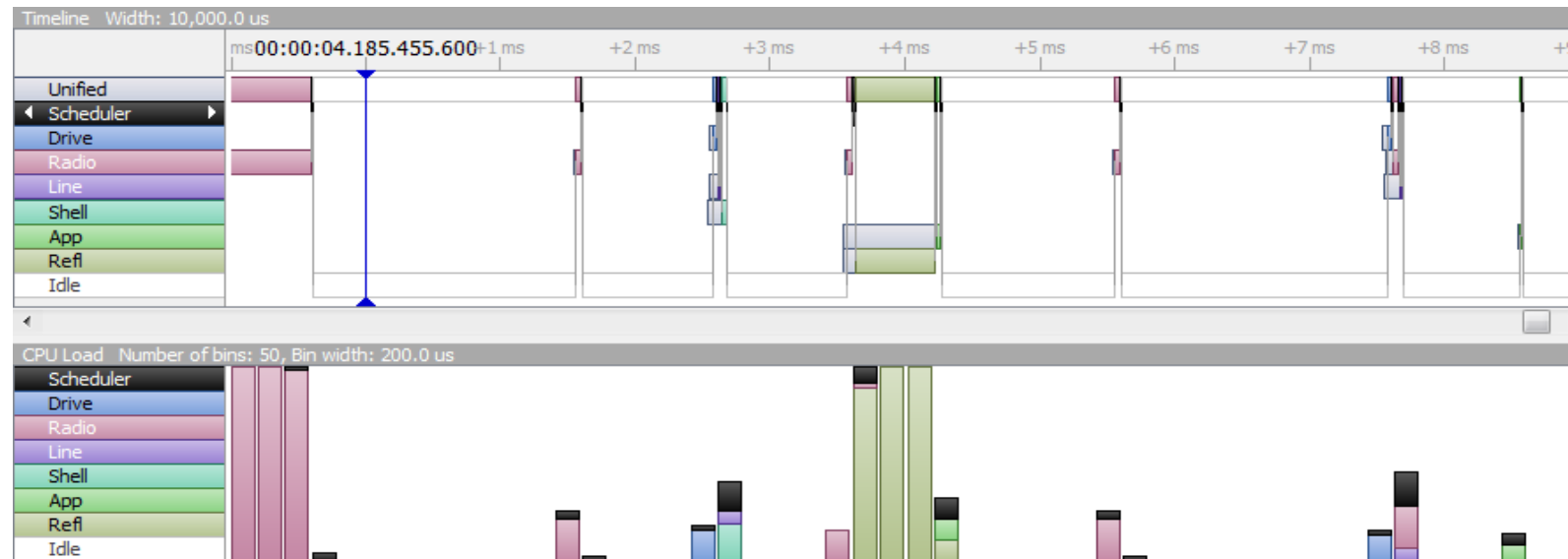
```
1 #ifndef traceTASK_SWITCHED_OUT
2     /* Called before a task has been selected to run.   pxCurrentTCB
3     holds a pointer
4     to the task control block of the task being switched out. */
5     #define traceTASK_SWITCHED_OUT()
6 #endif
```


Segger SystemViewer



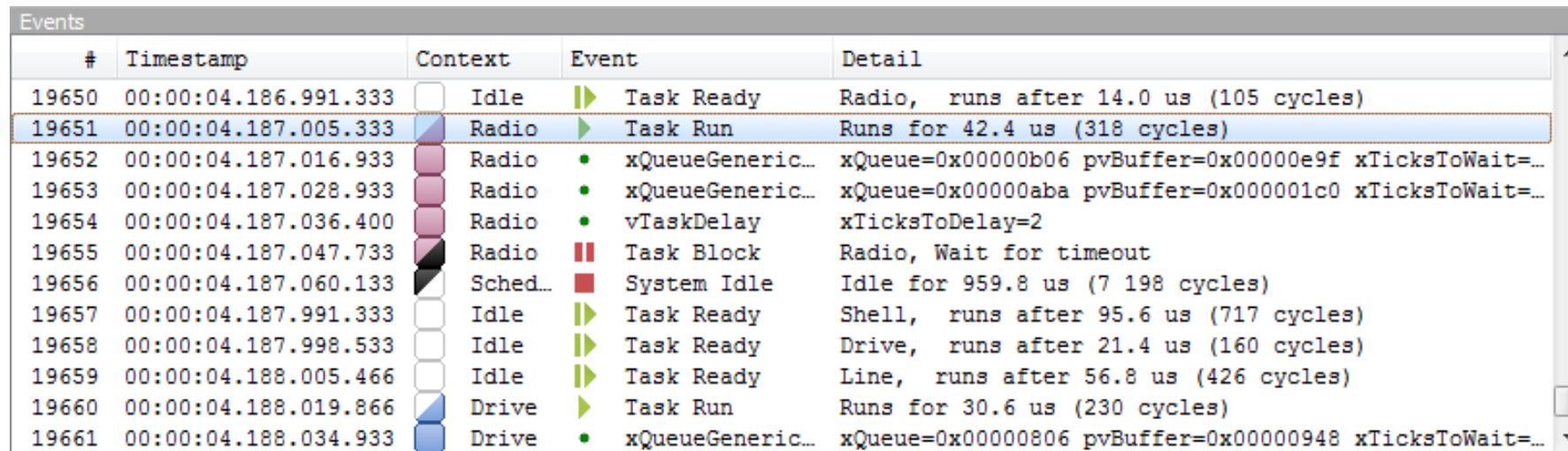
Segger SystemViewer

- Free-of-Charge, requires Segger debug interface
- Based on Segger RTT (Real Time Transfer)
- Realtime data recording and time measurement
- Uses Cortex M4 Cycle count register, SysTick on M0+
- <http://mcuoneclipse.com/2015/11/16/segger-systemview>



Segger SystemViewer - Timeline

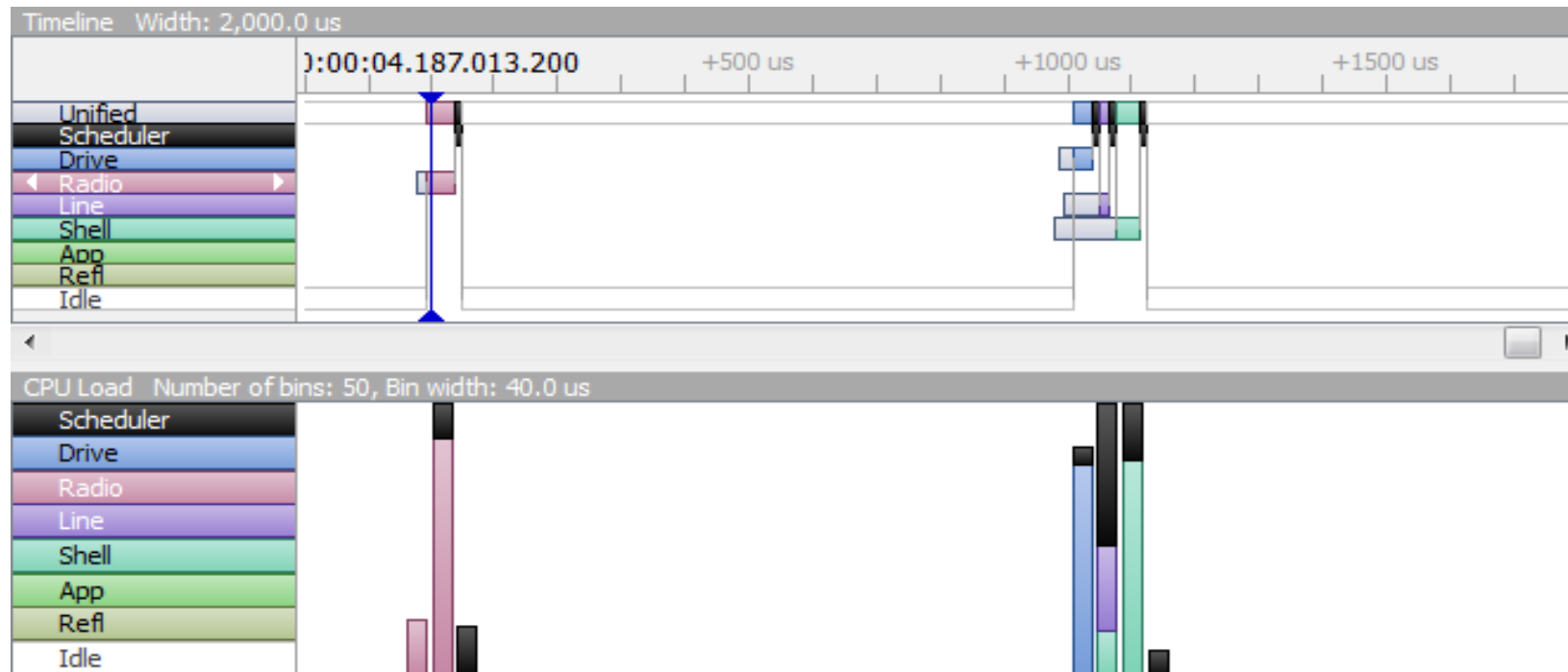
- Up to 1 Mio events
- Scheduler and Idle handled separately
- CPU load with 'bins': how many items to stack/show in a bar



#	Timestamp	Context	Event	Detail
19650	00:00:04.186.991.333	Idle	▶ Task Ready	Radio, runs after 14.0 us (105 cycles)
19651	00:00:04.187.005.333	Radio	▶ Task Run	Runs for 42.4 us (318 cycles)
19652	00:00:04.187.016.933	Radio	• xQueueGeneric...	xQueue=0x00000b06 pvBuffer=0x00000e9f xTicksToWait=...
19653	00:00:04.187.028.933	Radio	• xQueueGeneric...	xQueue=0x00000aba pvBuffer=0x000001c0 xTicksToWait=...
19654	00:00:04.187.036.400	Radio	• vTaskDelay	xTicksToDelay=2
19655	00:00:04.187.047.733	Radio	▬ Task Block	Radio, Wait for timeout
19656	00:00:04.187.060.133	Sched...	■ System Idle	Idle for 959.8 us (7 198 cycles)
19657	00:00:04.187.991.333	Idle	▶ Task Ready	Shell, runs after 95.6 us (717 cycles)
19658	00:00:04.187.998.533	Idle	▶ Task Ready	Drive, runs after 21.4 us (160 cycles)
19659	00:00:04.188.005.466	Idle	▶ Task Ready	Line, runs after 56.8 us (426 cycles)
19660	00:00:04.188.019.866	Drive	▶ Task Run	Runs for 30.6 us (230 cycles)
19661	00:00:04.188.034.933	Drive	• xQueueGeneric...	xQueue=0x00000806 pvBuffer=0x00000948 xTicksToWait=...

Segger SystemViewer - Timeline

- Timeline: what is executed
- Scheduler and Idle handled separately
- CPU load with 'bins': how many items to stack/show in a bar



Segger SystemViewer - Context

- List of tasks
- Priorities, stack information
- Runtime statistics, CPU load

Contexts												
Name	Type	Stack Information	Count	Frequency	Run Time	Total Run Time	Run Time/s		...Min		...Max	
Scheduler			3420	831 Hz	0.011 ms	285.398 ms	126.670 ms	12.67%	0.000 ms	0.00%	126.670 ms	12.67%
Drive		@3 226 Bytes @ 0x1fffa0a8	562	140 Hz	0.025 ms	100.098 ms	36.338 ms	3.63%	5.119 ms	0.51%	36.338 ms	3.63%
Radio		@3 252 Bytes @ 0x1fffb588	1345	328 Hz	0.042 ms	679.113 ms	96.286 ms	9.63%	96.286 ms	9.63%	204.636 ms	20.46%
Line		@2 246 Bytes @ 0x1fffa5c8	540	134 Hz	0.012 ms	105.481 ms	20.680 ms	2.07%	16.665 ms	1.67%	34.466 ms	3.45%
Shell		@1 274 Bytes @ 0x1fff92f8	268	64 Hz	0.037 ms	54.540 ms	13.941 ms	1.39%	3.635 ms	0.36%	27.940 ms	2.79%
App		@0 246 Bytes @ 0x1fff8dd8	513	122 Hz	0.015 ms	197.500 ms	66.941 ms	6.69%	22.612 ms	2.26%	66.941 ms	6.69%
Refl		@0 230 Bytes @ 0x1fff9a98	244	55 Hz	0.598 ms	194.751 ms	32.893 ms	3.29%	32.893 ms	3.29%	61.344 ms	6.13%
Idle			1828	447 Hz	1.945 ms	2578.060 ms	631.947 ms	63.19%	0.000 ms	0.00%	656.183 ms	65.62%

Lab: Percepio FreeRTOS+Trace

Inspecting runtime behavior of RTOS and Application with Percepio FreeRTOS+Trace

Outline: FreeRTOS+Trace

- **Problem**

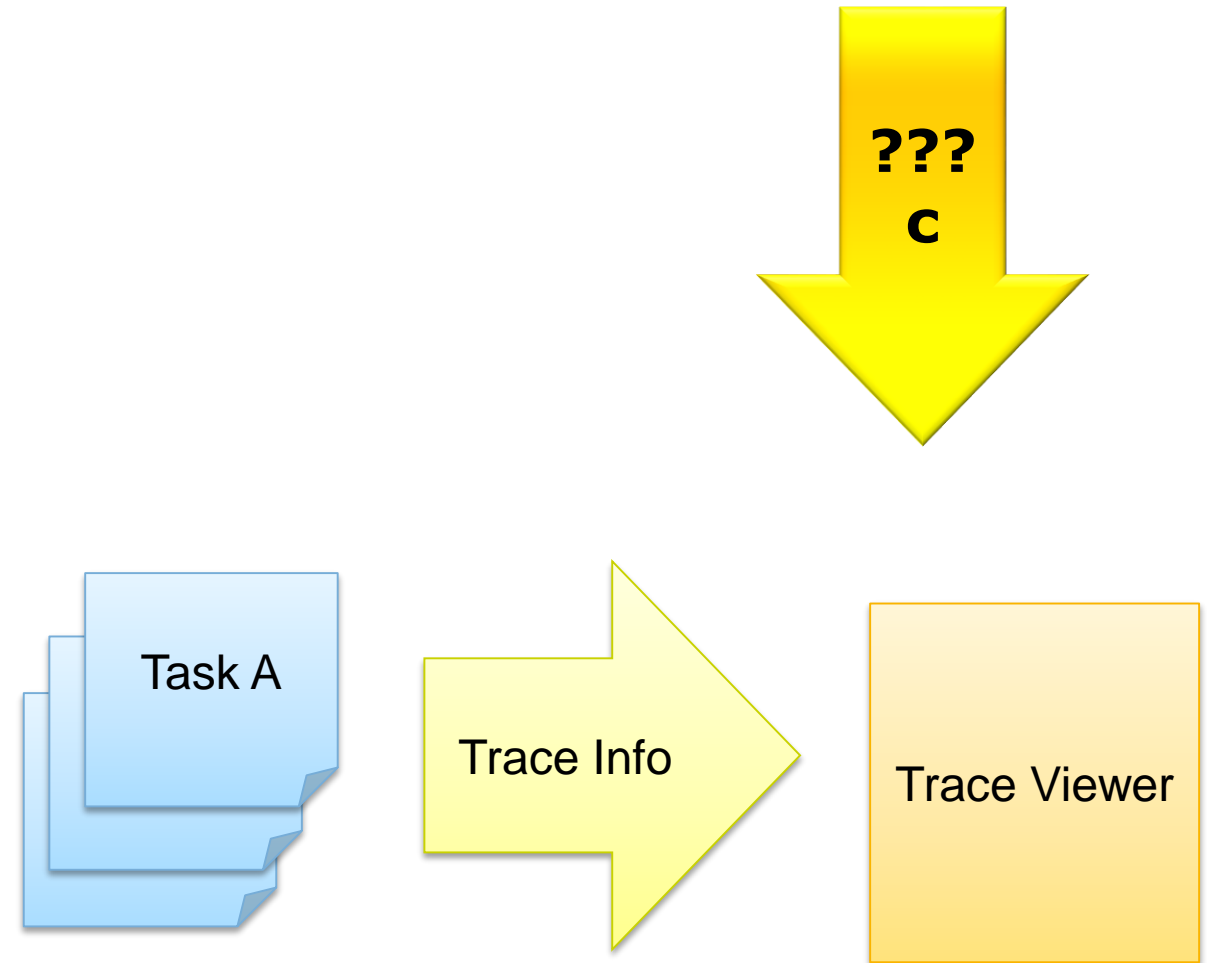
- Need insight into what is going on with application
- Heap allocation

- **Solution**

- Enable trace hooks in FreeRTOS

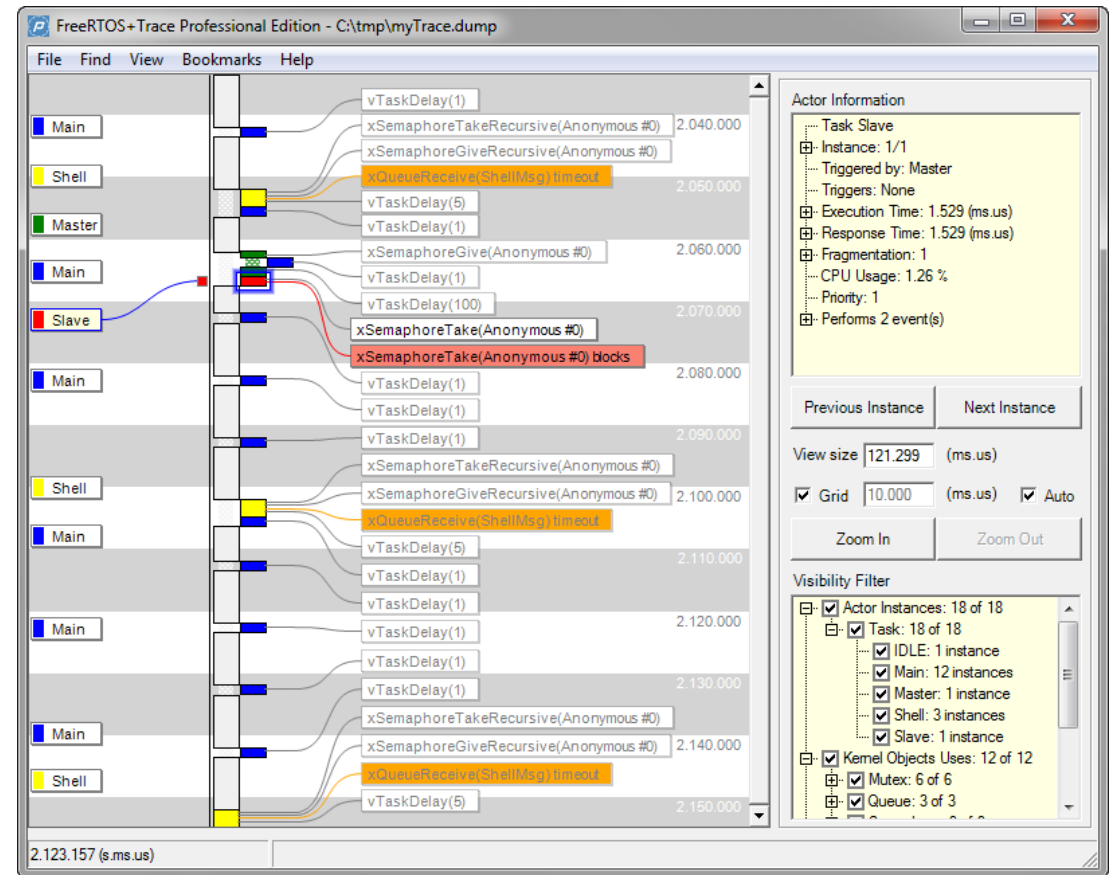
- **Steps**

- Enable trace hooks
- Dump trace data to host
- Inspect data



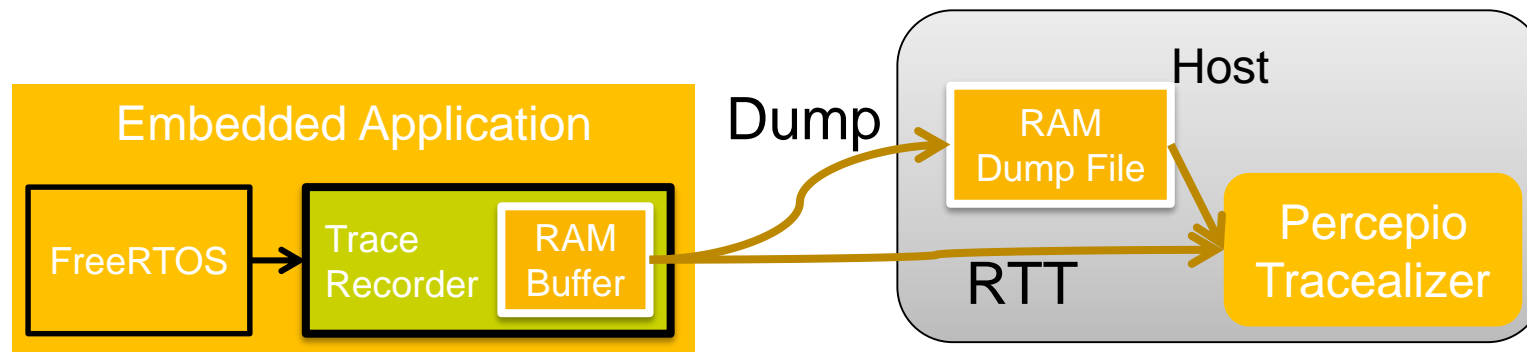
Goal: FreeRTOS+Trace

- From Percepio (<http://www.percepio.com>)
- Free and Professional edition
- Hosts: Windows, Linux
- Over 20 graphical views
- Tasks, System Calls and User Events
- CPU Load
- Timing Variations
- Communication Flow
- Kernel Object History
- User Events, Signal Plots
- ...



How It Works...

- Trace Hooks in FreeRTOS
 - Macros which call recorder functions
- Recorder: task create, context switch, mutex ready, ...
 - Recorder uses RAM/RTT buffer (circular, linear), 4 bytes/event, ~1-2% CPU load
- Dump/Export data: SD card, UART, debugger, semihosting, J-Link, ...
- Direct streaming with SEGGER RTT



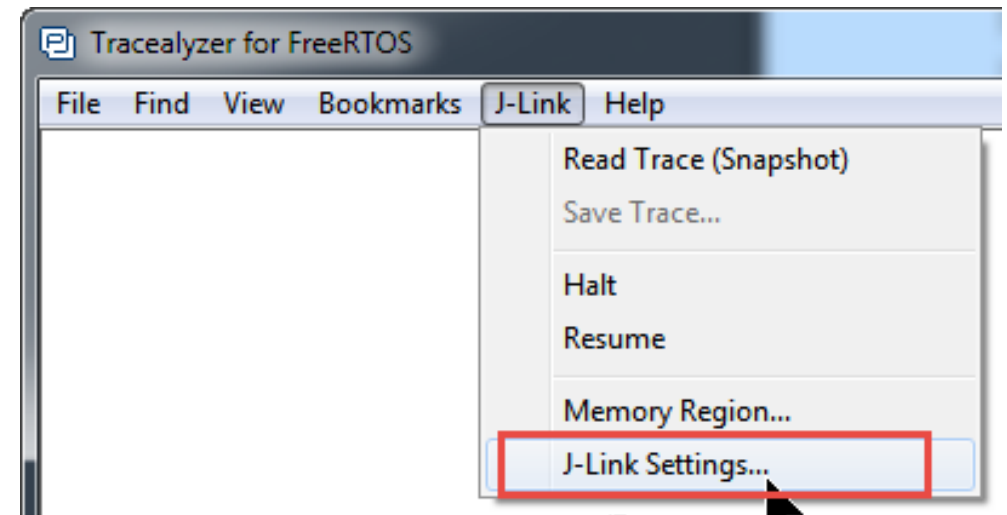
Enabling Percepio Trace

- Settings in source\FreeRTOS_Config.h
- Enable (around line #164)
 - configUSE_TRACE_FACILITY 1
 - configUSE_TRACE_HOOKS 1
- configUSE_SEGGER_SYSTEM_VIEWER_HOOKS has to be disabled!

```
#define configUSE_TRACE_FACILITY          1  
#define configUSE_TRACE_HOOKS            1  
#define configUSE_SEGGER_SYSTEM_VIEWER_HOOKS 0
```

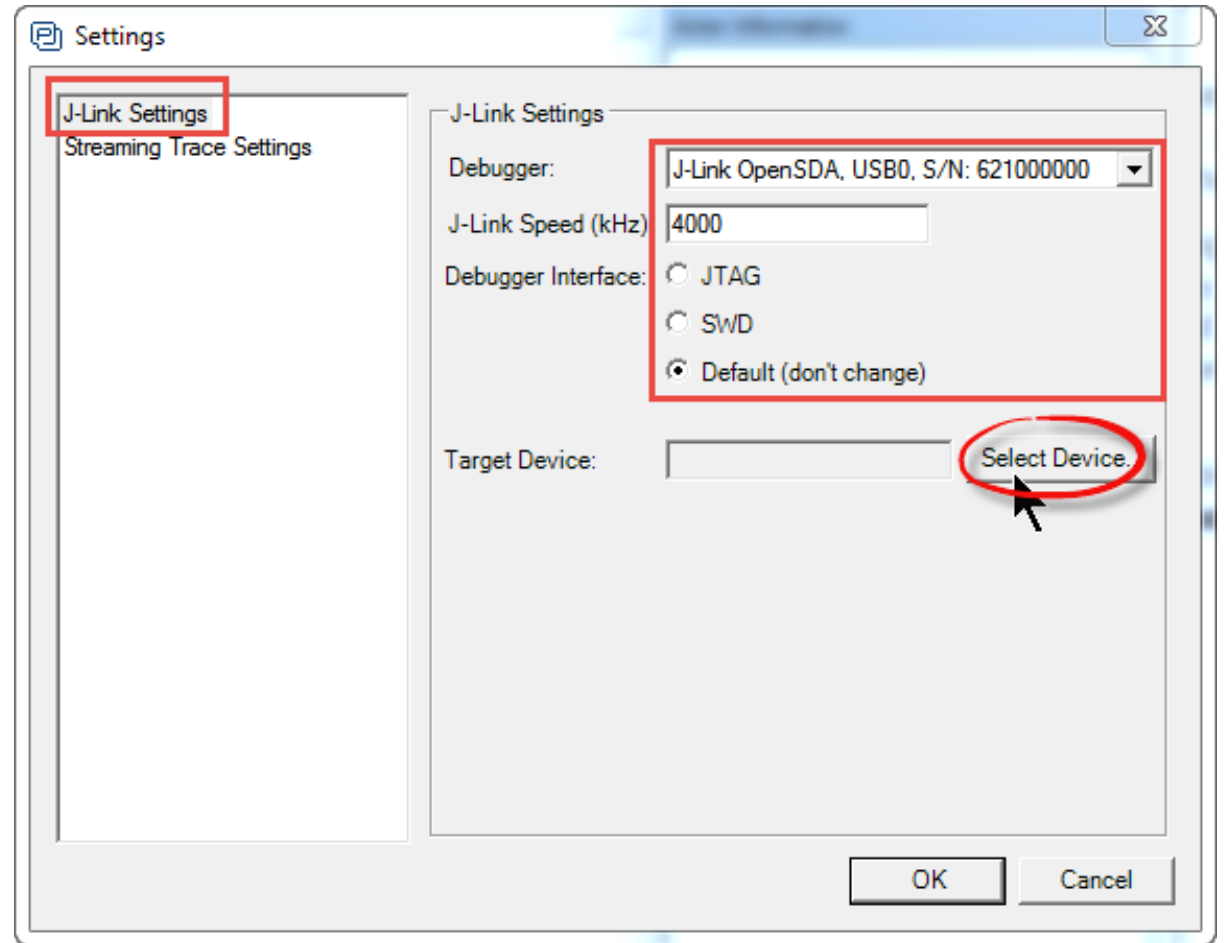
Percepio Debugger Selection

- **Launch** Percepio FreeRTOS+Trace
 - Shortcut on Desktop
- Menu ***J-Link > J-Link Settings...***



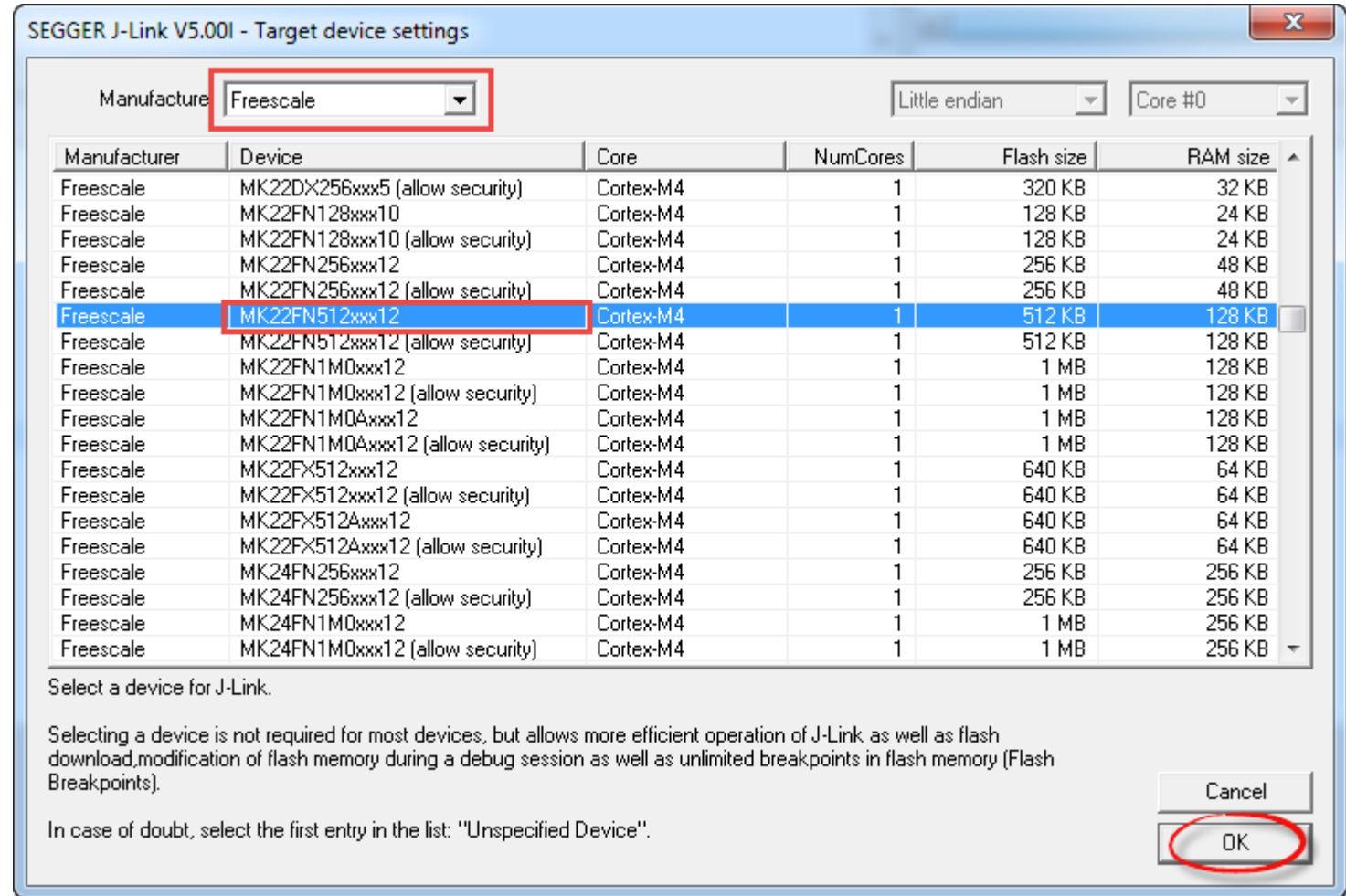
Connect to Target System

- J-Link Settings
 - Debugger (USB Port)
 - 4000 kHz
 - Default debugger interface
- Press Device Selection...



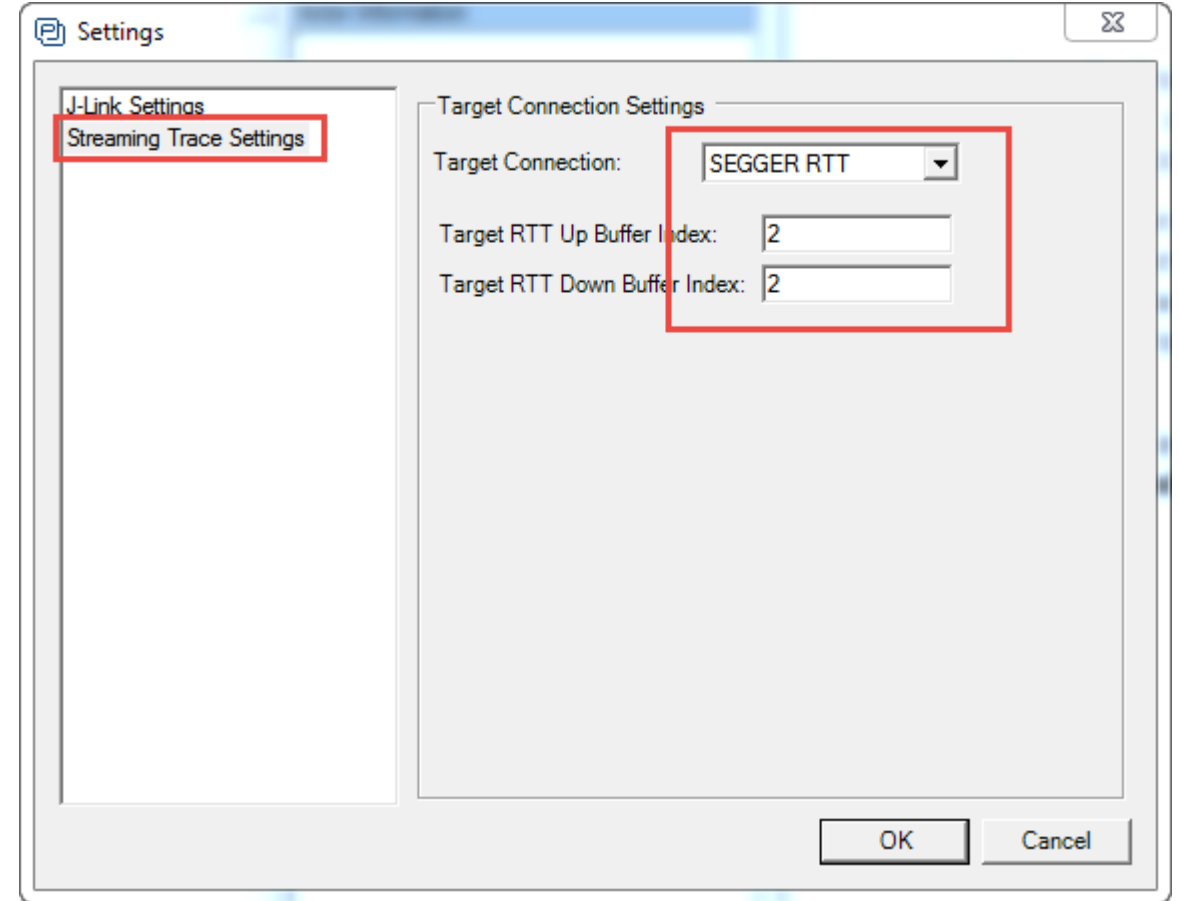
Device Selection

- Select device
 - **Freescal**
 - **MK22FN512xxx12**
- Press **OK**



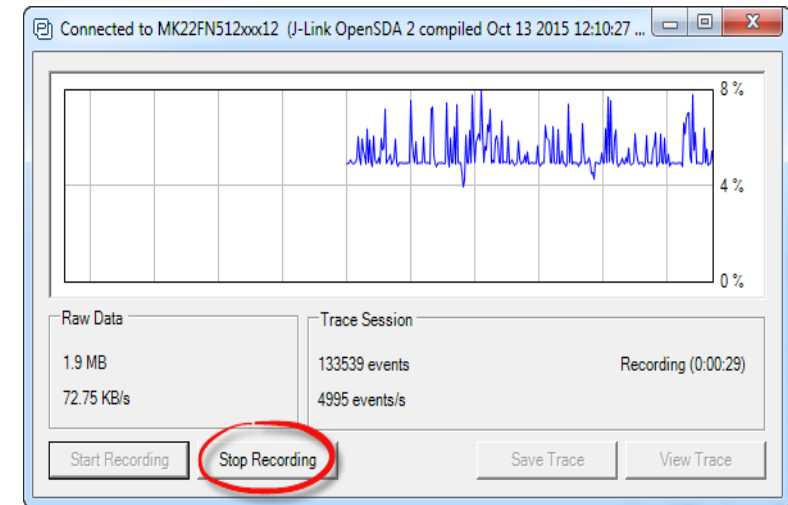
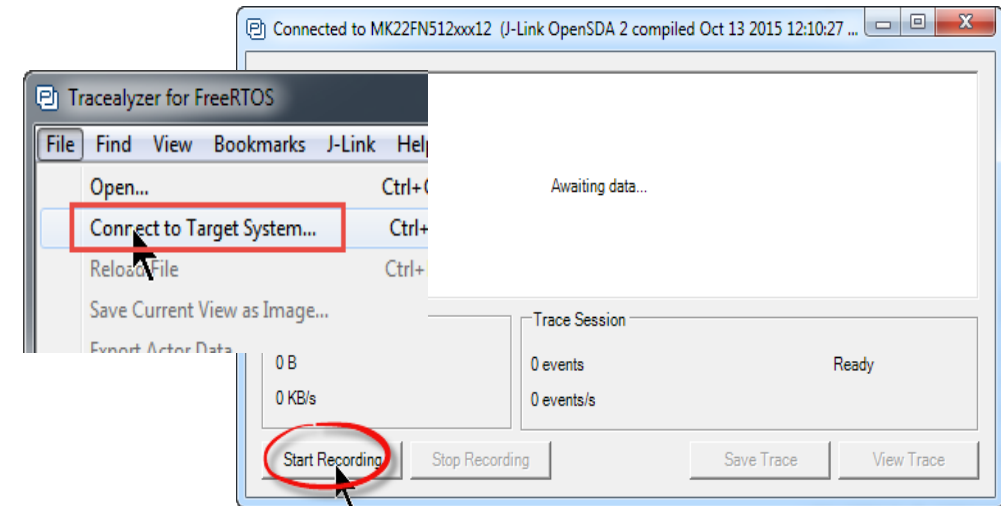
Streaming Trace Settings

- Select Streaming Trace Settings
 - SEGGER RTT
 - Up Buffer Index: 2
 - Down Buffer Index: 2
- Press OK



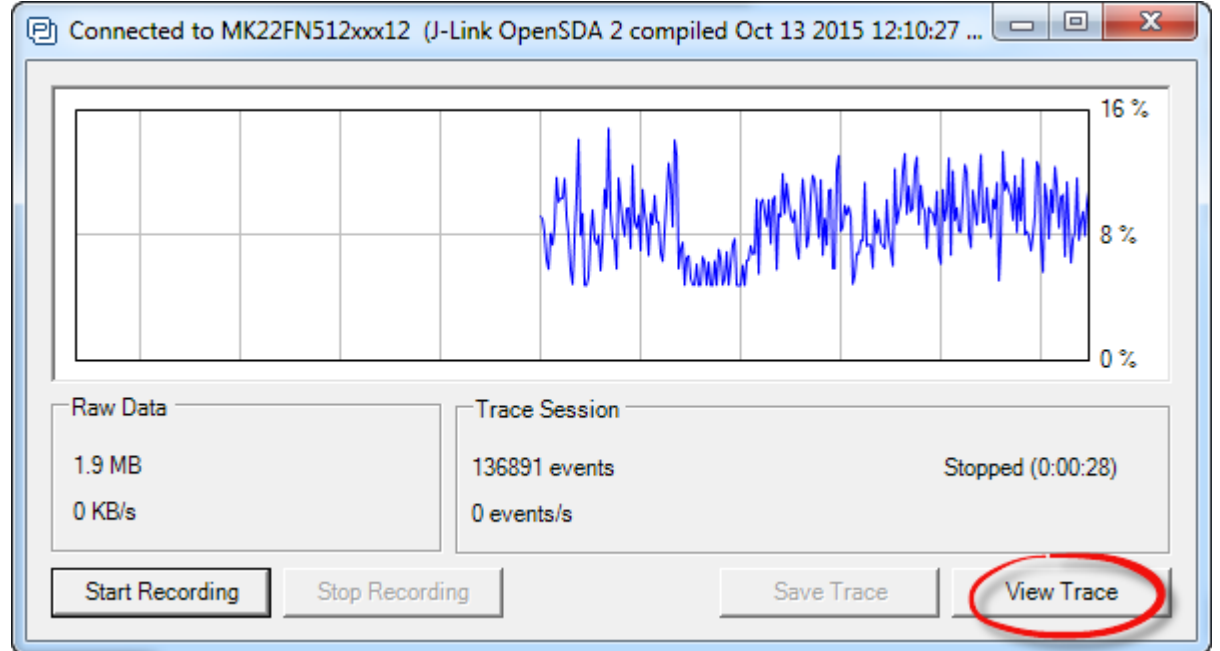
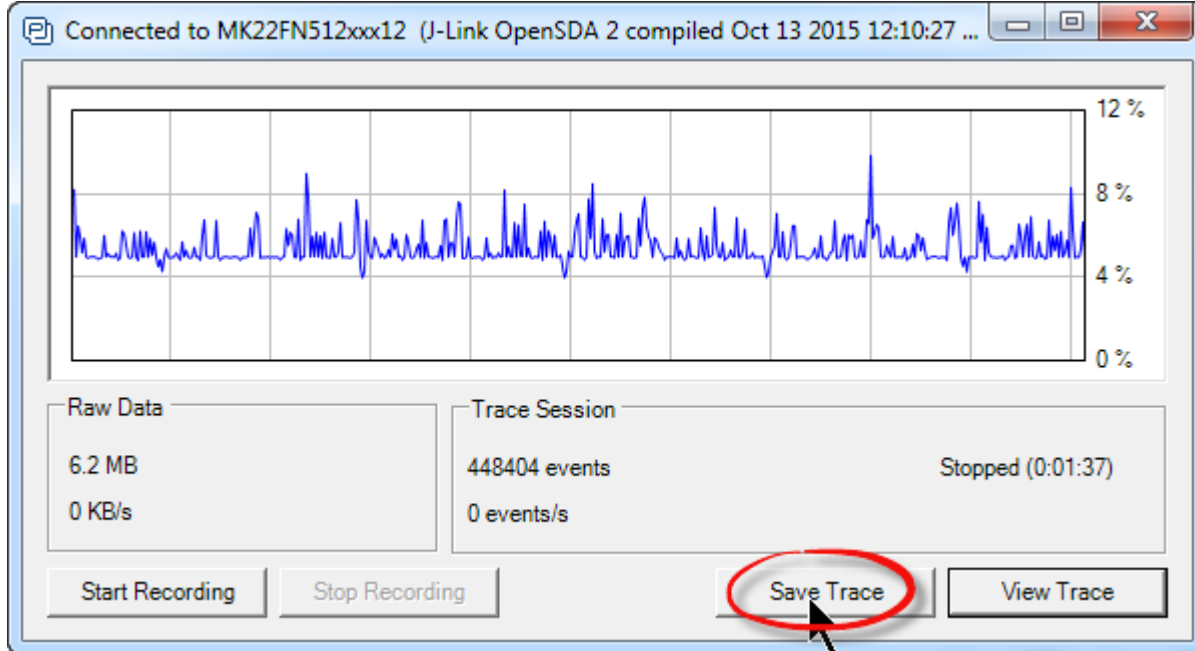
Collecting Trace

- Menu File > Connect to Target System...
- Press Start Recording
 - Percentage shows CPU load
 - Trace Session Information
 - Number of events
 - Number of events/second
 - Recording time
- Collect trace for a while
- Press Stop Recording



Saving and Viewing Trace

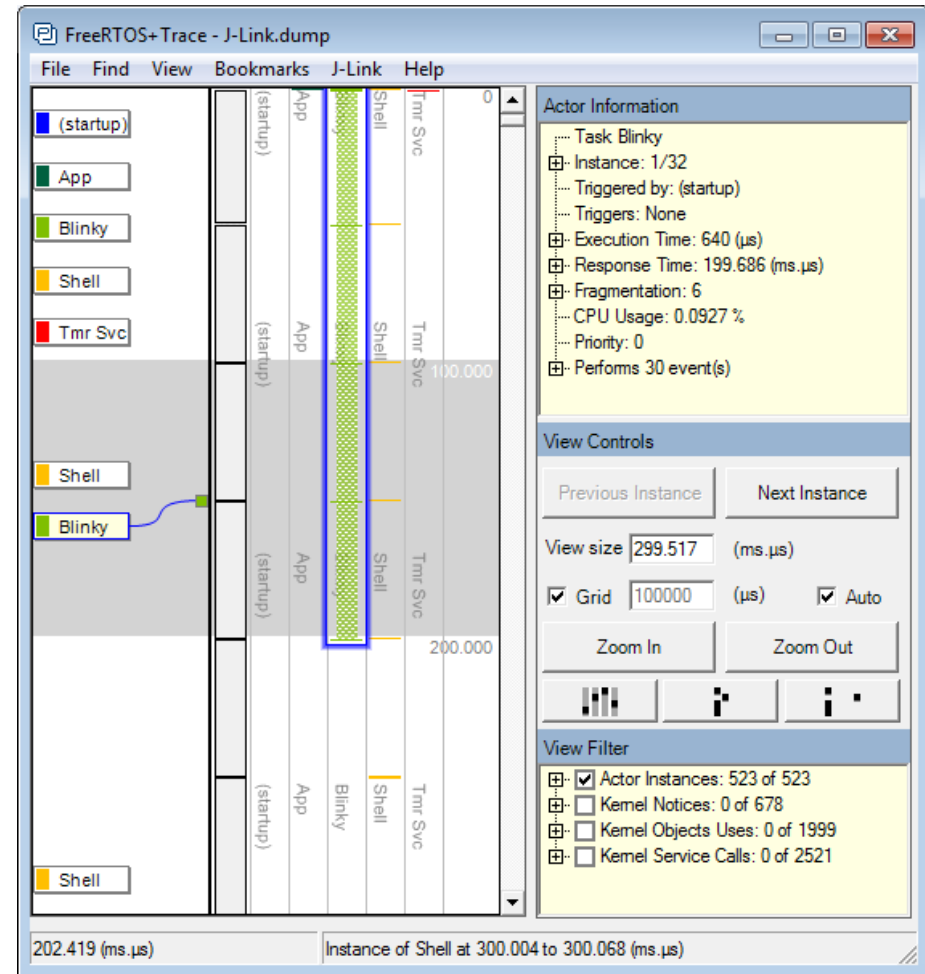
- Press Save Trace
 - Store it in the default location
- Press View Trace



Blinky, What Are You Doing?

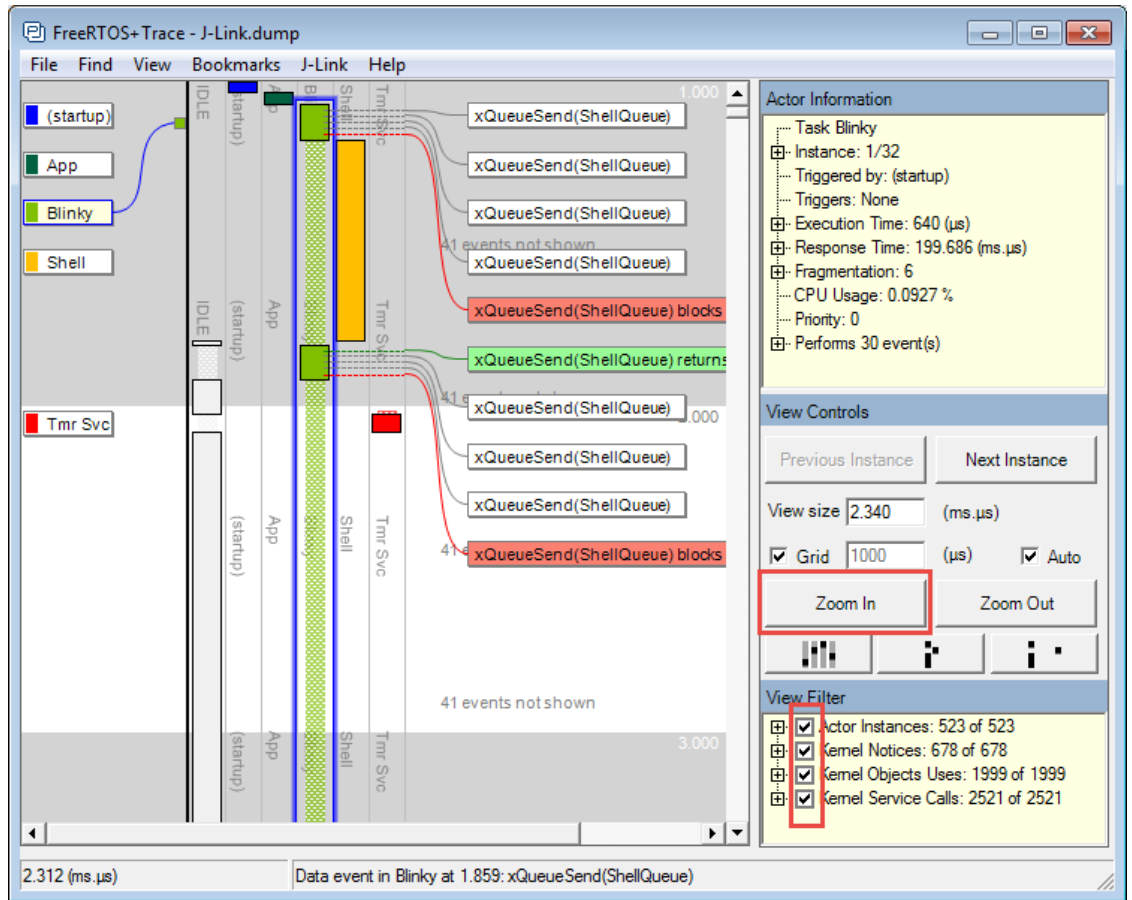
- Trace right after RTOS startup
- Blinky blinks the and writes to the UART
- Blinky Task execution is extended to 200 ms?

trc1.dump



Blinky?

- Zoom into the very first milliseconds
- Enable all Kernel items
- Blinky sends for items to queue, is then blocked
- Shell receives four items, is then blocked



Communication Flow

- Menu View > Communication Flow
- Double Click on Queue to show Queue details

The screenshot displays the 'Communication Flow' tool interface. The main window shows a flow diagram with two queues: 'ShellQueue Queue' and 'TmrQ Queue'. 'Blinky' is connected to 'ShellQueue Queue', and 'TmrQ Queue' is connected to 'Tmr Svc'.

The 'ShellQueue (Queue)' window is open, showing a table of events and a detailed view of the selected event.

mestam	Actor	Event	Block time	Status	Size	Queue
1.000	(startup)	xQueueCreate			0	Empty
1.088	Blinky	xQueueSend		Sent post #1	1	1
1.105	Blinky	xQueueSend		Sent post #2	2	1 2
1.123	Blinky	xQueueSend		Sent post #3	3	1 2 3
1.140	Blinky	xQueueSend		Sent post #4	4	1 2 3 4
1.161	Blinky	xQueueSend	662	Trying to send...	4	1 2 3 4
1.196	Shell	xQueueReceive		Received post #1	3	1 2 3 4
1.225	Shell	xQueueReceive		Received post #2	2	2 3 4
1.244	Shell	xQueueReceive		Received post #3	1	3 4
1.501	Shell	xQueueReceive		Received post #4	0	4
1.761	Shell	xQueueReceive		Failed to receive	0	Empty

The detailed view for the selected event (1.161) shows:

- Timestamp: 1.161
- Actor: Blinky
- Event: xQueueSend
- Status: Blocked for 662 (µs)
- Parameter: N/A

Buttons for navigation: Goto Entry/Exit Event, Show in Trace, Goto Sending Event, Goto Receiving Event.

Blinky Task

- Queue Length is problem
- ShellQueue.c: Increase SQUEUE_LENGTH to 32

```
static void blinky_task(void *param) {
    (void)param;
    for(;;) {
        switch(whichLED) {
            case RGB_LED_GREEN:
                SQUEUE_SendString("Blinking GREEN RGB LED\r\n");
                GPIO_DRV_TogglePinOutput(kGpioLED1);
                GPIO_DRV_SetPinOutput(kGpioLED2);
                GPIO_DRV_SetPinOutput(kGpioLED3);
                break;
            case RGB_LED_RED:
                SQUEUE_SendString("Blinking GREEN RED LED\r\n");
                GPIO_DRV_SetPinOutput(kGpioLED1);
                GPIO_DRV_TogglePinOutput(kGpioLED2);
                GPIO_DRV_SetPinOutput(kGpioLED3);
                break;
            case RGB_LED_BLUE:
                SQUEUE_SendString("Blinking GREEN BLUE LED\r\n");
                GPIO_DRV_SetPinOutput(kGpioLED1);
                GPIO_DRV_SetPinOutput(kGpioLED2);
                GPIO_DRV_TogglePinOutput(kGpioLED3);
                break;
        }
        vTaskDelay(pdMS_TO_TICKS(500)); /* wait for 500 ms */
    } /* for */
}
```

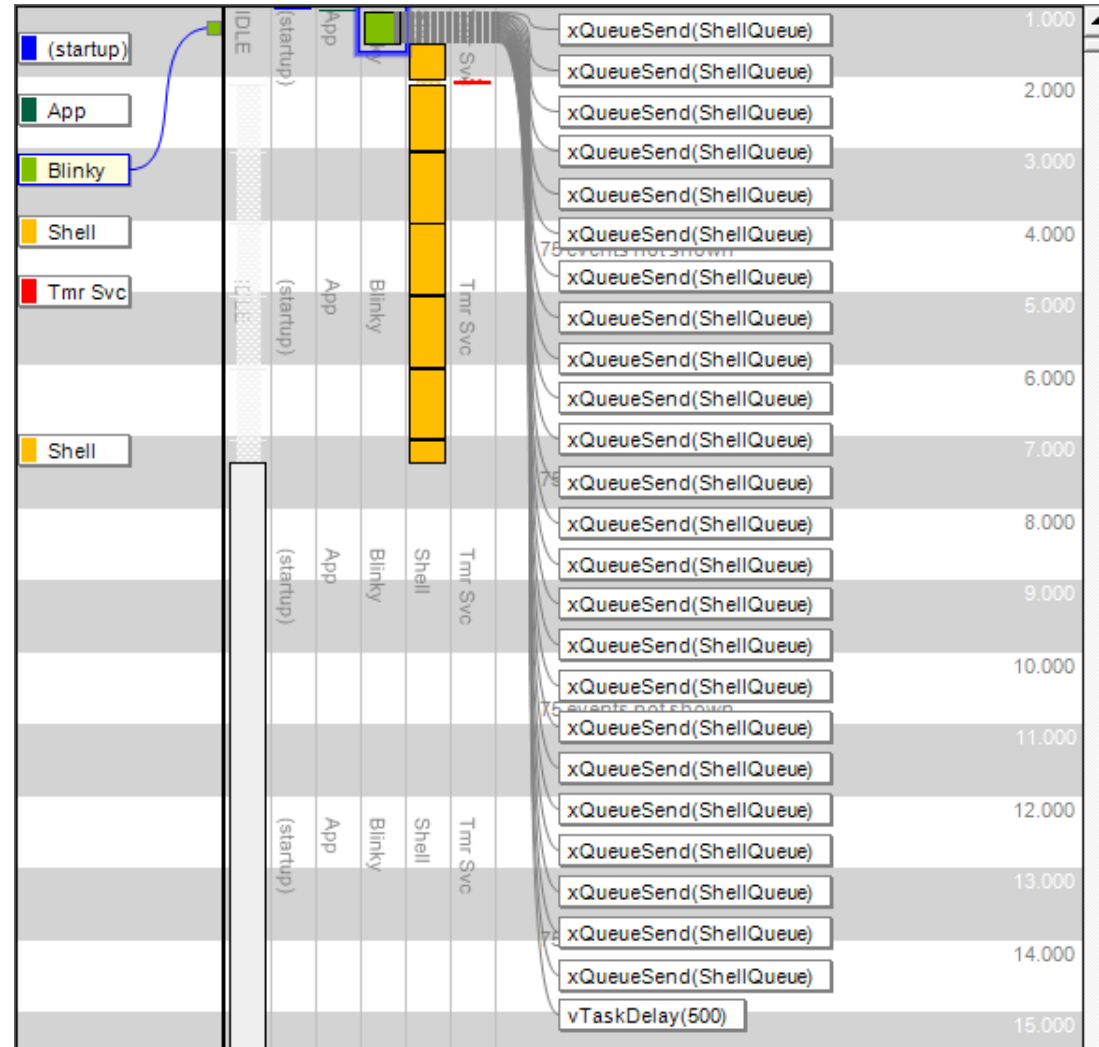
```
#define SQUEUE_LENGTH    4 /* items in queue, that's my buffer size */
#define SQUEUE_ITEM_SIZE 1 /* each item is a single character */

void SQUEUE_SendString(const unsigned char *str) {
    while(*str!='\0') {
        if (xQueueSendToBack(SQUEUE_Queue, str, portMAX_DELAY)!=pdPASS) {
            for(;;){} /* ups? */
        }
        str++;
    }
}
```

Increased Queue Length

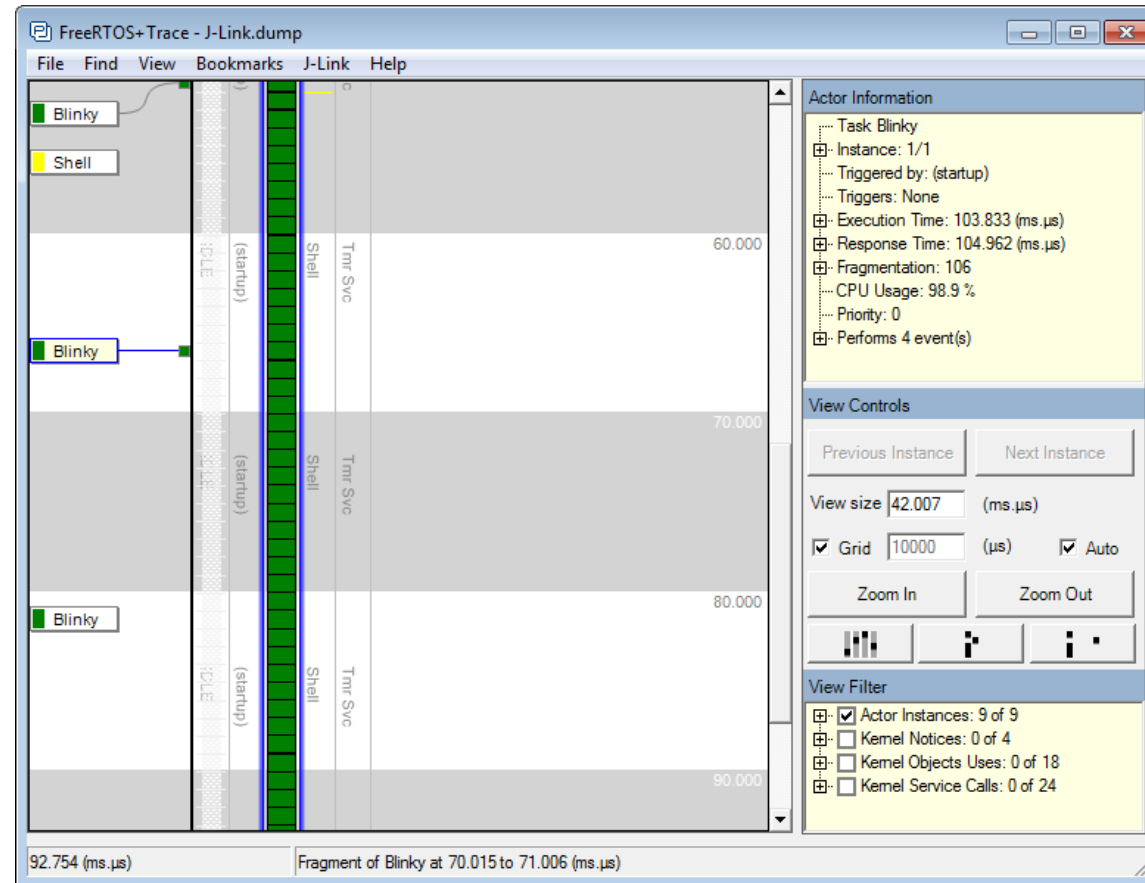
- Build, debug and trace
- Blinky task not blocked any more
- 200 ms → 7 ms 😊

trc2.dump



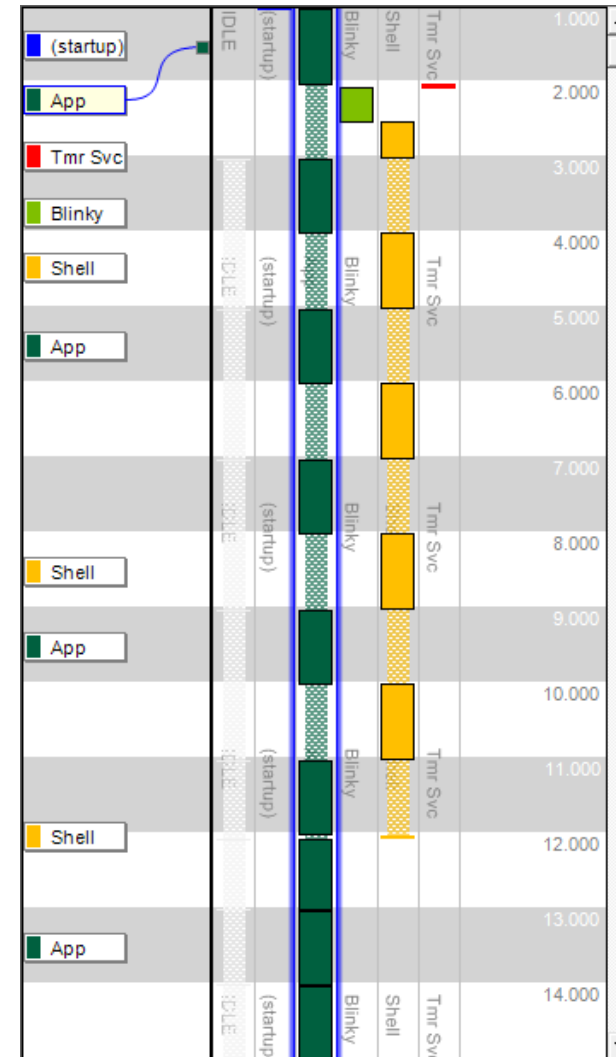
Blinky is Using a Lot of CPU...

Check what happens if you have the push buttons pressed



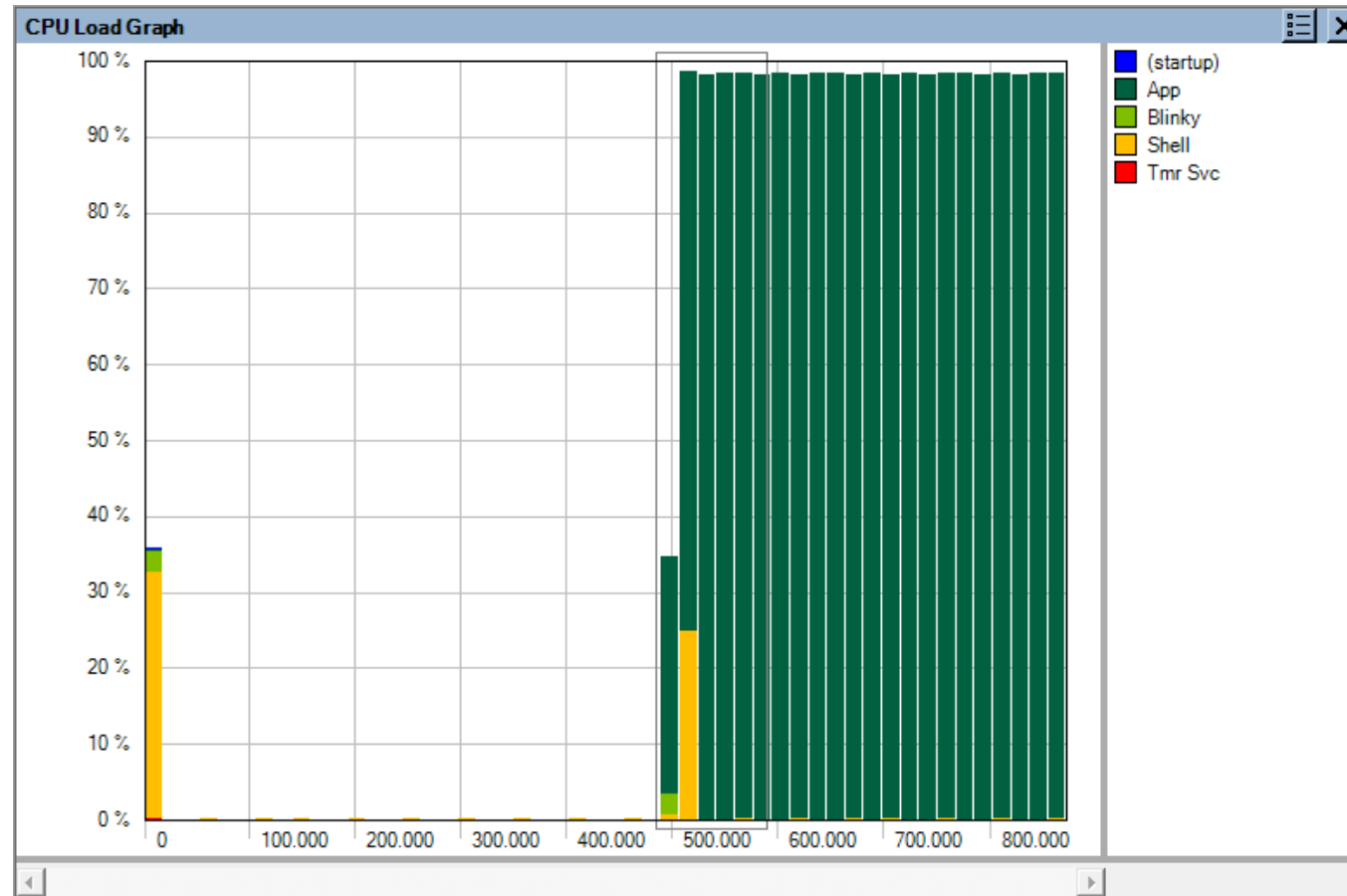
Problem with App Task

- Restart application with SW2
- Pressed
- App Task is blocking for a long time



CPU Load Graph

- Menu View > CPU Load Graph



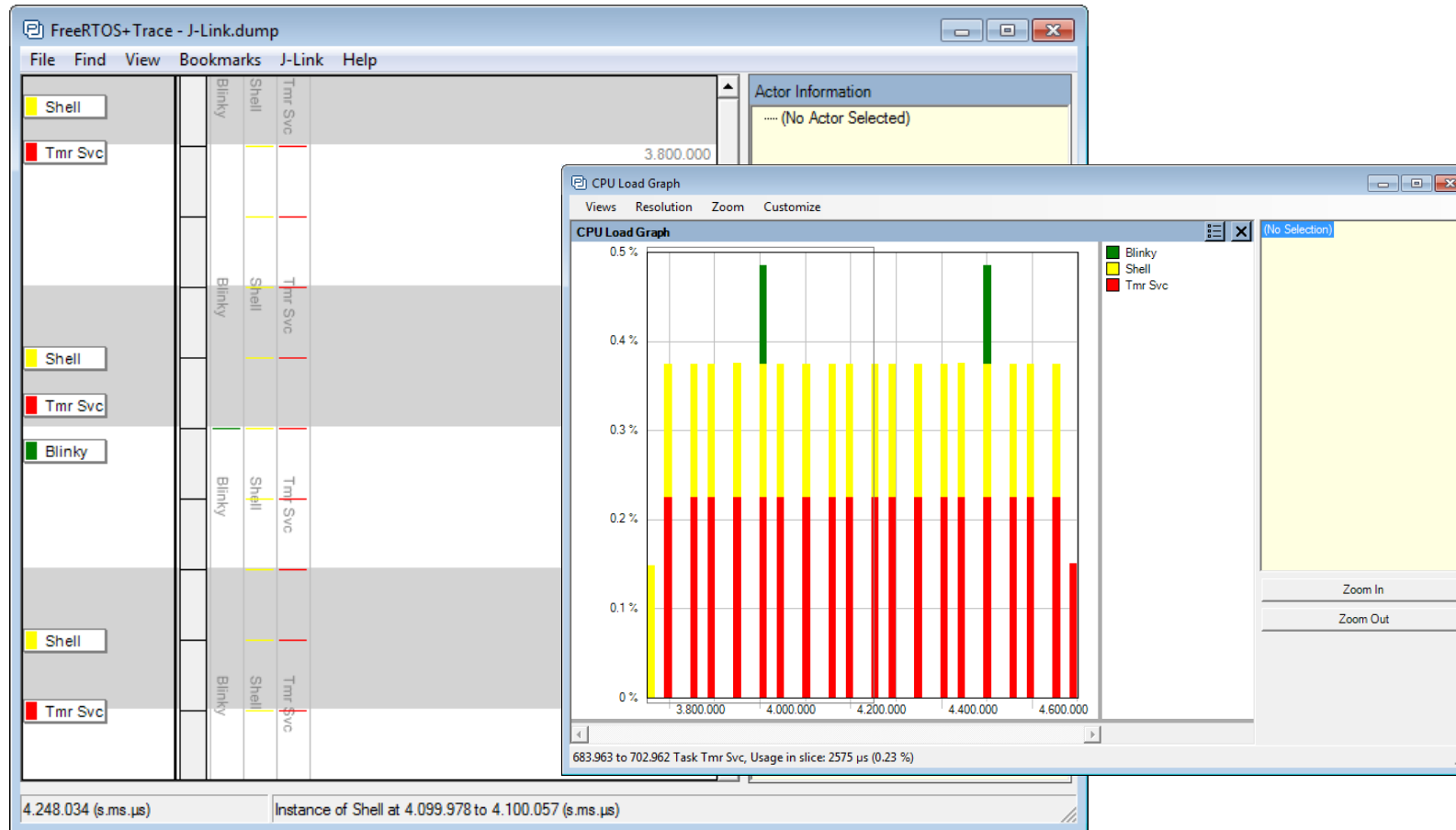
No Busy Delay

- vTaskDelay(pdMS_TO_TICKS(50));
- Build and debug/collect trace again

```
Application.c
/5
76 static void app_task(void *param) {
77     (void)param;
78     for(;;) {
79         vTaskDelay(pdMS_TO_TICKS(500)); /* wait for 500 ms */
80         if (GPIO_DRV_ReadPinInput(kGpioSW2)==0) { /* pressed */
81             //busydelay(10000); /* debounce */
82             vTaskDelay(pdMS_TO_TICKS(50));
83             while (GPIO_DRV_ReadPinInput(kGpioSW2)==0) {
84                 /* wait until released */
85             }
86             SQUEUE_SendString("SW2 has been pressed!\r\n");
87             whichLED <<= 1;
88             if (whichLED > RGB_LED_BLUE) {
89                 whichLED = RGB_LED_RED;
90             }
91         }
92         if (GPIO_DRV_ReadPinInput(kGpioSW3)==0) { /* pressed */
93             //busydelay(10000); /* debounce */
94             vTaskDelay(pdMS_TO_TICKS(50));
95             while (GPIO_DRV_ReadPinInput(kGpioSW3)==0) {
96                 /* wait until released */
97             }
98             SQUEUE_SendString("SW3 has been pressed!\r\n");
99             whichLED >>= 1;
100             if (whichLED < RGB_LED_RED) {
101                 whichLED = RGB_LED_BLUE;
102             }
103         }
104     } /* for */
105 }
106
```

Improved Application Timing

RTOS wait instead of busy wait



Summary: FreeRTOS+Trace

- FreeRTOS Hooks to intercept RTOS or application calls
- Dump/stream/store/send trace data to host
- Percepio FreeRTOS+Trace viewer to analyze data
- **Further Information:**
 - <http://percepio.com/>
 - http://www.freertos.org/FreeRTOS-Plus/FreeRTOS_Plus_Trace/FreeRTOS_Plus_Trace.shtml
 - <http://mcuoneclipse.com/2012/03/23/tracing-with-freertos-trace-from-percepio/>
 - <http://mcuoneclipse.com/2015/01/05/updated-percepio-tracealyzer-and-trace-library-to-version-v2-7-0/>
 - <http://mcuoneclipse.com/2012/06/03/percepio-freertos-trace-v2-2-2-released/>

FreeRTOS Heap

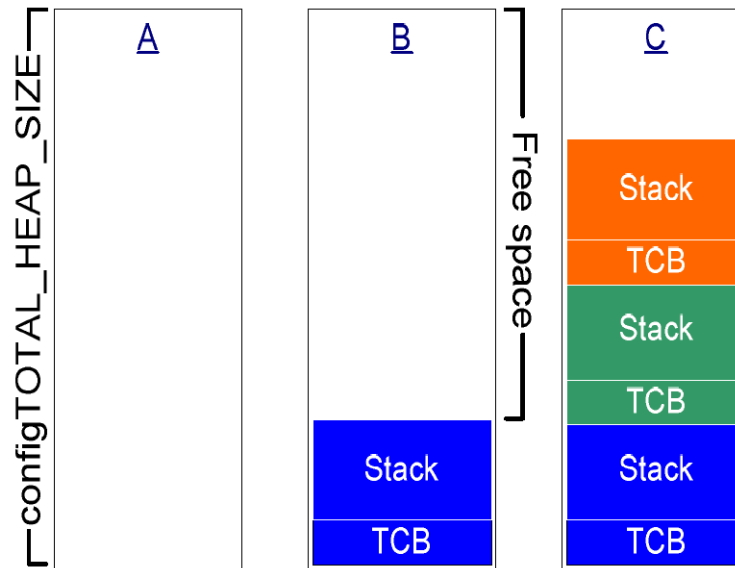
Reduce the amount of heap and RAM used...

Outline: FreeRTOS Heap

- **Problem**
 - Reducing RAM footprint of application
- **Solution**
 - Change allocation scheme
 - Reduce FreeRTOS Heap Size
- **Steps**
 - Inspect stack usage
 - Reduce default stack size
 - Reduce heap size

FreeRTOS Heap Size

- FreeRTOS uses dynamic heap memory for
 - Internal data structures (TCB, Task Control Block)
 - Queue, Semaphore, Mutex
 - Task Stack
- `#define configTOTAL_HEAP_SIZE ((size_t)(32*1024))`



Source: freeRTOS.org

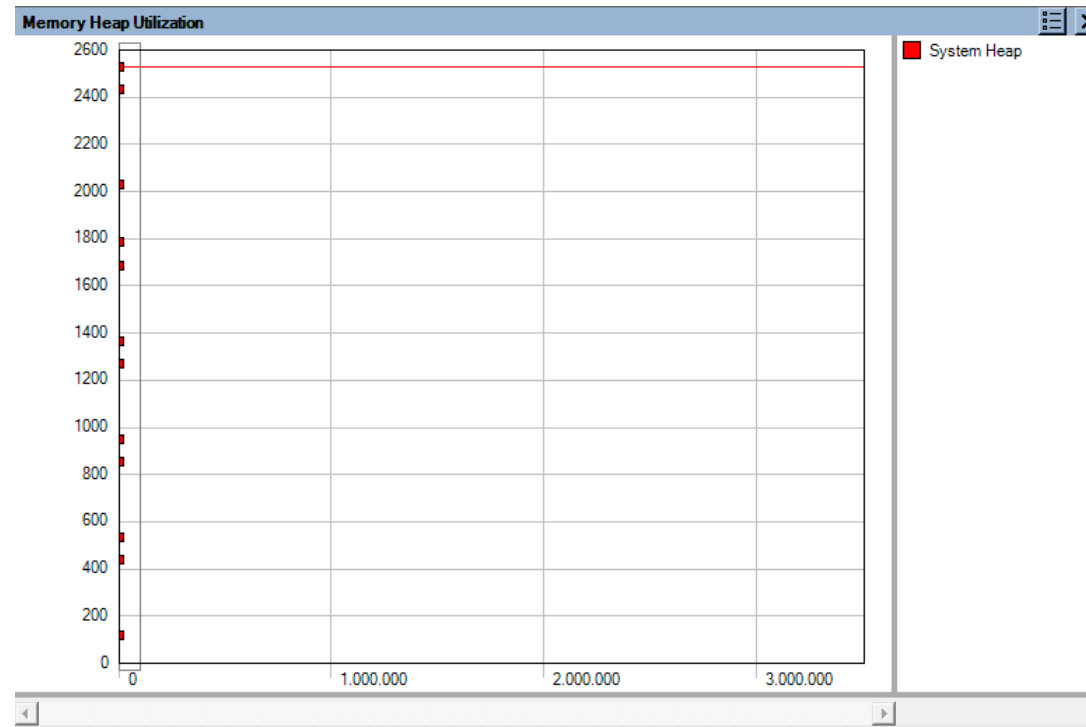
FreeRTOS Heap Schemes

- FreeRTOS Heap != ANSI/Runtime Heap
- Different schemes (allocate only, free with no merge, ...)
- `#define configFRTOS_MEMORY_SCHEME 4`
 - 1 (only alloc) <<= Change to this
 - 2 (alloc/free)
 - 3 (malloc)
 - 4 (coalesc blocks)
 - 5 (multiple blocks)

```
137 /*-----*/
138 /* Heap Memory */
139 #define configFRTOS_MEMORY_SCHEME 1 /* either 1 (only alloc), 2 (alloc/f
```

Heap Usage

- Menu View > Memory Heap Utilization
- Less than 3 Kbyte used now
- Line 140: `#define configTOTAL_HEAP_SIZE ((size_t)(3*1024))`



Lab: Heap

- Determine amount of Heap used
- Reduce heap size
- Determine that application is using allocation only
- Change heap scheme
- Rebuild application
 - Reduced RAM usage

Summary: Heap

- Heap used for FreeRTOS data structures
- Stack size allocated at task creation in heap
- Reduce heap size to the amount needed
- Use scheme1 (allocate only) if no task/object deletion

- Further Information:
 - <http://www.freertos.org>

Session Closing

What we have achieved...



Summary

- **Application**
 - Have enough headroom for diagnostics/trace
 - Architectural changes can have big impact
 - Use proper buffering
- **FreeRTOS and Kinetis SDK**
 - Configure what you need
 - Tune heap and stack size (but be careful)
- **Development Tools**
 - Use profiling/performance analysis
 - Use FreeRTOS+Trace with run control
 - Turn on optimizations in GNU Tools
 - Free tools can do a lot for you!

For Further Information

- Kinetis Design Studio IDE: www.nxp.com/kds
- Kinetis SDK: www.nxp.com/ksdk
- FreeRTOS: www.freertos.org
- Examples and Tutorials for Kinetis MCUs: www.mcuoneclipse.com

Contact us:

- Mark Ruthenbeck: mark.ruthenbeck@nxp.com
- Anthony Huereca: anthony.huereca@nxp.com
- Erich Styger: erich.styger@nxp.com



SECURE CONNECTIONS
FOR A SMARTER WORLD

ATTRIBUTION STATEMENT

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, CoolFlux, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE Classic, MIFARE DESFire, MIFARE Plus, MIFARE Flex, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TrenchMOS, UCODE, Freescale, the Freescale logo, AltiVec, C 5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and μ Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. © 2015–2016 NXP B.V.

