FTF2016
TECHNOLOGY FORUM

FreeRTOS SOLUTIONS FOR KINETIS MCU'S

FTF-DES-N1956 & FTF-DES-N2048

ERICH STYGER
FTF-DES-N1956 & FTF-DES-N2048
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

?+?

=!

List of Labs

- Lab 1: Installation
  - Overview software and tools
  - Setting up the machines and boards
- Lab 2: Getting Started
  - Build, debug and run the application on the board
- Lab 3: Kernel Awareness
  - Inspect RTOS using Eclipse plugin and GDB
- Lab 4: Segger SystemView
  - Real-time trace of application with RTT
- Lab 5: Percepio Tracealizer
  - Advanced real-time trace with RTT

Outline: Lab 1 - Installation

- **Problem**
  - Application with FreeRTOS does not run as expected

- **Goal**
  - Learn about the tools and lab setup to solve the problems
  - Install the tools (preinstalled on lab computers)

- **Lab**
  - Prepare and install the software and tools

Lab Setup

- Lab Files in C:\FTF\N1956

- FRDM-K22F_USB_Mouse…
  - Kinetis Design Studio SDK Project used in labs
- Tools
  - Setup of Tools used in labs (preinstalled)
- wsp_kds3.2.0
  - Eclipse workspace used for labs

Preinstalled on Lab Computers

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

NXP Software and Tools for FreeRTOS

- NXP Kinetis SDK
  - http://www.nxp.com/ksdk
  - Open Source, permissible license
  - CMSIS-Core startup and drivers
  - FreeRTOS, Micrium µC/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
  - Stop-Mode Kernel information (tasks, queues, timers, …)
- FRDM-K22F Board
  - ARM Cortex-M4F

TCB#	Task Name	Task Handle	Task State	Prio...	Stack Usage	Runtime
1	Buttons	0x1fff13a8	Blocked	2 (2)	232 B / 676 B	0x0 (0.0%)
2	App	0x1fff1738	Blocked	1 (1)	196 B / 668 B	0x0 (0.0%)
3	TzCtrl	0x1fff1ac8	Blocked	1 (1)	156 B / 660 B	0x0 (0.0%)
4	USB task	0x1fff2ff8	Blocked	4 (4)	180 B / 4.75 kB	0x0 (0.0%)
5	IDLE	0x1fff3388	Running	0 (0)	0 B / 708 B	0x12900 (97.6%)
6	Tmr Svc	0x1fff3718	Blocked	5 (5)	212 B / 668 B	0x0 (0.0%)
7	usb device	0x1fff4bd0	Suspended	5 (5)	376 B / 4.68 kB	0x740 (2.4%)

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

KSDK

The software framework and reference for Kinetis MCU application development

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')

Preinstalled on Lab Computers

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)

Learn more at: www.nxp.com/KDS
Community: https://community.freescale.com/community/kinetis-design-studio

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

NXP FreeRTOS Eclipse Plugins

- NXP FreeRTOS Kernel Awareness plugins for Eclipse/KDS
- Views
  - Task List
  - Queue List
  - Timer List
  - Heap Usage
- ‘Stop-Mode’
  - Suspend debugger

Kernel Awareness Installation

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

Pre-Installed on Lab Computers

FreeRTOS Tread Aware Debugging

COMING SOON

- With Segger GDB Server, updated Segger GDB Server

Freedom Development Platforms

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

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)

Low-cost/low-power development hardware

Enables quick application prototyping and demonstration of Kinetis MCU families

Segger J-Link OpenSDA Debug 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

FreeRTOS and 3rd Party Tools

- FreeRTOS is integrated in Kinetis SDK
- 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

Segger SystemViewer

SEGGER SystemViewer V2.34 - FRDM-K22F [FreeRTOS] on Cortex-M4F

File View Go Target Tool Window Help

Events

#	Timestamp	Context	Event	Detail
2168	00:00:03.440.092.800	Scheduler	System Idle	Idle for 19.9106 ms (149 330 cycles)
2169	00:00:03.459.992.800	Idle	Task Ready	App, runs after 10.6 us (80 cycles)
2170	00:00:03.460.003.467	App	Task Run	Runs for 26.8 us (201 cycles)
2171	00:00:03.460.015.200	App	xQueueGenericRece…	xQueue=0x000005d2 pvBuffer=0x00000822 xTicksToWait=0 xJustPeek=0
2172	00:00:03.460.023.200	App	vTaskDelay	xTicksToDelay=5
2173	00:00:03.460.030.267	App	Task Block	App, Wait for timeout
2174	00:00:03.460.042.400	Scheduler	System Idle	Idle for 29.9610 ms (224 708 cycles)
2175	00:00:03.489.992.800	Idle	Task Ready	Buttons, runs after 10.6 us (80 cycles)
2176	00:00:03.490.003.467	Buttons	Task Run	Runs for 120.114 ms (900 855 cycles)
2177	00:00:03.509.992.800	Buttons	Task Ready	App, runs after 533.6556 ms (4 002 417 cycles)
2178	00:00:03.609.993.467	Buttons	Task Ready	Tmr Svc, runs after 17.6 us (132 cycles)
2179	00:00:03.610.001.200	Buttons	Task Ready	USB task, runs after 89.6 us (672 cycles)
2180	00:00:03.610.011.067	Tmr Svc	Task Run	Runs for 64.6 us (485 cycles)
2181	00:00:03.610.022.000	Tmr Svc	xQueueGenericRece…	xQueue=0x00000858 pvBuffer=0x00000f30 xTicksToWait=0 xJustPeek=0

Property	Detail
Target System	
Name	FRDM-K22F
OS	FreeRTOS
Device	Cortex-M4F
Cycle Freque…	7 500 000 Hz
Cycle Period	133 ns
Uptime	00:03:34.529.952.933
Recording	
Host Time	10 Mai 2016 21:23:08
Duration	00:00:07.610.104
Title	
Author	
Description	
Tasks	

Timeline Width: 2.0 s

00:00:03.479.679.862 +500 ms +1000 ms +1500 ms

CPU Load Number of bins: 50, Bin width: 40.0 ms

Scheduler
Tmr Svc
usb device
USB task
Buttons
App
Idle

Contexts

Name	Type	Stack Information	Run Count	Frequency	Last Run Time	Min Run Time	Max Run Time	Total Run Time	Run Time/s	
Scheduler			800	124 Hz	0.0125 ms	0.0118 ms (#963)	0.0152 ms (#2245)	10.0397 ms	1.5558 ms	0.16%
Tmr Svc	@5	108 @ 0x1fff39e8	16	2 Hz	0.0649 ms	0.0646 ms (#1960)	0.0666 ms (#2244)	0.9761 ms	0.1298 ms	0.01%
usb device	@5	1185 @ 0x1fff3e38	0	0 Hz	0.0 ms			0.0 ms	0.0 ms	0.00%
USB task	@4	1173 @ 0x1fff2260	16	2 Hz	0.0136 ms	0.0134 ms (#715)	0.0136 ms (#404)	0.2030 ms	0.0272 ms	0.00%
Buttons	@2	145 @ 0x1fff1a08	640	100 Hz	0.0148 ms	0.0144 ms (#1874)	437.7204 ms (#2741)	1200.8077 ms	1.4846 ms	0.15%
App	@1	122 @ 0x1fff1d98	130	20 Hz	0.0250 ms	0.024 ms (#962)	0.0268 ms (#2173)	3.2429 ms	0.5014 ms	0.05%

Terminal

Context	Message	Timestamp
Buttons	Push butt…	00:03:30.329.984
Buttons	Push butt…	00:03:30.933.515
Buttons	Push butt…	00:03:31.749.984

4 054 Events 00:00:07.610.104

Segger SystemViewer

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

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 connection

Segger SystemView

- Download and install latest J-Link Software from https://www.segger.com/jlink-software.html
- Download SystemView from https://www.segger.com/systemview.html

Preinstalled on Lab Computers

Percepio Tracealizer

- Percepio (http://www.percepio.com)
- Free/Professional (Paid) 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

FreeRTOS+Trace Tracealizer from Percepio

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

Preinstalled on Lab Computers

Summary: Lab 1 - Installation

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

Outline: Lab 2 - Getting Started

- **Problem**
  - Where to get started?
  - What is the application on the board doing?

- **Goal**
  - Learn the software tools and about the embedded application
  - Be able to build and debug the application

- **Lab**
  - Starting Kinetis Design Studio IDE
  - Using new workspace and importing Project
  - Build, Debug, Run the application

Start 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

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 (patched)
- USB HID Composite (Mouse + Keyboard)
- FreeRTOS+Trace enabled
  - Segger SystemView with Segger RTT
  - Percepio Tracealizer with Segger RTT

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

Board Functions

- Connect both USB cables to host PC
- SW2: enables USB mouse
- SW3: changes RGB LED color

ToDo List

- Task View (“ToDo” List)
- ToDo’s in source files
- Double-Click to jump to file/location

Problems | Console | Tasks | Search | Call Hierarchy | (x)= Variables | Breakpoints | Progress

8 items

✓	!	Description	Resource	Path	Location
		TODO 1 Increase stack size by 50	Application.c	/FRDM-K22F_USB_Mouse_Kybd_SDK_v2.0/source	line 155
		TODO 2 Add delay of 10 ms	Application.c	/FRDM-K22F_USB_Mouse_Kybd_SDK_v2.0/source	line 99
		TODO 3 Enable Segger SystemView...	FreeRTOSCon...	/FRDM-K22F_USB_Mouse_Kybd_SDK_v2.0/source	line 167
		TODO 4 Add delay of 50 ms	Application.c	/FRDM-K22F_USB_Mouse_Kybd_SDK_v2.0/source	line 81
		TODO 5 Add delay of 50 ms	Application.c	/FRDM-K22F_USB_Mouse_Kybd_SDK_v2.0/source	line 93
		TODO 5 Fix Memory leak	Application.c	/FRDM-K22F_USB_Mouse_Kybd_SDK_v2.0/source	line 113

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

First Debug: Open Launch Configuration

- Go to Debug Configurations
- Select Configuration and press Debug

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

Test 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 2 - Getting Started

- 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. Includes FreeRTOS
- Application: high level functions and overview

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

Outline: Lab 3 - Kernel Awareness

- **Problem**
  - What is the problem?
  - What are the tasks doing?

- **Goal**
  - Using NXP FreeRTOS Kernel Awareness with GDB

- **Lab**
  - Using Views
  - Inspecting task stack usage
  - Fixing stack problem
  - Fixing performance problem

Debug Application

- Select Project Folder
- Press Debug Icon

Open FreeRTOS Views

- Debug Perspective
- Menu FreeRTOS, open all views
- Or: Menu Window > Show View > Other..

Tips & Tricks: FreeRTOS View Arrangement

- Debug your application, then suspend it (Pause)
- Move FreeRTOS views to the side (Drag Tab to the side and drop)
- Own stack of views
- Does not interfere with console updates

Tasks

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

Task List (FreeRTOS) | Timer List (FreeRTOS) | Heap Usage (FreeRTOS) | Queue List (FreeRTOS)

TCB#	Task Name	Task Handle	Task State	Prior...	Stack Usage	Event Object	Runtime
1	Buttons	0x1fff1578	Running	2 (2)	4 B / 1.07 kB		0x9c0e (100.0%)
	Task Number:	0x0					
	Stack Base:	0x1fff10c0					
	Stack Top:	0x1fff150c					
	Stack High Water Mark:	4 B					
	Stack Size:	1.07 kB					
2	App	0x1fff1908	Ready	1 (1)	0 B / 724 B		0x0 (0.0%)
3	TzCtrl	0x1fff1c98	Ready	1 (1)	0 B / 724 B		0x0 (0.0%)
4	USB task	0x1fff31c8	Blocked	4 (4)	180 B / 4.75 kB		0x0 (0.0%)
5	IDLE	0x1fff3558	Ready	0 (0)	0 B / 724 B		0x0 (0.0%)
6	Tmr Svc	0x1fff38e8	Blocked	5 (5)	92 B / 668 B	TmrQ (Rx)	0x0 (0.0%)
7	usb device	0x1fff4da0	Suspended	5 (5)	0 B / 4.68 kB	USBNotify (Rx)	0x0 (0.0%)

Queue List

- Name, Address, Length, Item Size
- Tx and Rx waiting
- Read and write pointers

Task List (FreeRTOS) | Queue List (FreeRTOS) | Timer List (FreeRTOS) | Heap Usage (FreeRTOS)

#	Queue Name	Address	Length	Item Size	# Tx W...	# Rx W...	Queue Type
1	ShellQueue	0x1fff0e00	0/32	0x4 (4 B)	0	0	Queue
	Head:	0x1fff0e54					
	Tail:	0x1fff0ed4					
	Read from:	0x1fff0ed0					
	Write to:	0x1fff0e54					
2	Sem_SW2	0x1fff0ee0	0/1	Empty	0	0	Binary Semaphore
3	Sem_SW3	0x1fff0f40	0/1	Empty	0	0	Binary Semaphore
4	Sem_LED	0x1fff0fa0	0/1	Empty	0	0	Binary Semaphore
5	Sem_Mouse	0x1fff1000	0/1	Empty	0	0	Binary Semaphore
6	Sem_Kbd	0x1fff1060	0/1	Empty	0	0	Binary Semaphore

#	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

Task List (FreeRTOS) | Queue List (FreeRTOS) | Timer List (FreeRTOS) | Heap Usage (FreeRTOS)

ID	Timer Name	Period [ticks]	Auto reload?	Timer Number	Callback function
0x0	swtimer	50	Yes	0x0	vTimerCallback (0x000076e9)

Heap Usage

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

Task List (FreeRTOS) | Queue List (FreeRTOS) | Timer List (FreeRTOS) | Heap Usage (FreeRTOS)

Type	Heap Base	Heap End	Heap Usage	Free Space	Heap Usage Graph
4	0x1fff0df4	0x1fff6de8	16.01 kB / 23.99 kB	33.3% (7.98 kB)	66.7% Used

#	Details	Block Start	Block End	Size
1	Allocated	0x1fff0df4	0x1fff0dff	0xc (12 B)
2	ShellQueue (Queue)	0x1fff0e00	0x1fff0e53	0x54 (84 B)
3	Allocated	0x1fff0e54	0x1fff0edf	0x8c (140 B)
4	Sem_SW2 (Queue)	0x1fff0ee0	0x1fff0f33	0x54 (84 B)
5	Allocated	0x1fff0f34	0x1fff0f3f	0xc (12 B)
6	Sem_SW3 (Queue)	0x1fff0f40	0x1fff0f93	0x54 (84 B)
7	Allocated	0x1fff0f94	0x1fff0f9f	0xc (12 B)

Stacks and Heap 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' (Kinetis: 4 bytes!)
- Stacks, queues, Semaphores allocated in FreeRTOS Heap

FreeRTOS Task Stack Usage

- High Water Mark for Stack
- Mark: set at context switch
- Check the Stack Usage column
  - Everything looking good, right?

Task List (FreeRTOS) | Queue List (FreeRTOS) | Timer List (FreeRTOS) | Heap Usage (FreeRTOS)

TCB#	Task Name	Task Handle	Task State	Priority	Stack Usage	Event Object	Runtime
1	Buttons	0x1fff09a8	Running	2 (2)	0 B / 148 B		0x64 (100.0%)
2	App	0x1fff0ba8	Ready	1 (1)	0 B / 324 B		0x0 (0.0%)
3	USB task	0x1fff20d8	Blocked	4 (4)	136 B / 4.75 kB		0x0 (0.0%)
4	IDLE	0x1fff22d8	Ready	0 (0)	0 B / 324 B		0x0 (0.0%)
5	Tmr Svc	0x1fff24d8	Blocked	5 (5)	32 B / 268 B	TmrQ (Rx)	0x0 (0.0%)
6	usb device	0x1fff3990	Suspended	5 (5)	0 B / 4.68 kB	USBNotify (Rx)	0x0 (0.0%)

Enough Stack?

- Press SW2 or SW3 while program is running
- Problem: Stack usage increase and gets critical!

Task List (FreeRTOS) | Queue List (FreeRTOS) | Timer List (FreeRTOS) | Heap Usage (FreeRTOS)

TCB#	Task Name	Task Handle	Task State	Priority	Stack Usage	Event Object	Runtime
1	Buttons	0x1fff09a8	Running	2 (2)	144 B / 148 B		0x532 (98.5%)
2	App	0x1fff0ba8	Ready	1 (1)	0 B / 276 B		0x0 (0.0%)
3	USB task	0x1fff20d8	Blocked	4 (4)	136 B / 4.75 kB		0x0 (0.0%)
4	IDLE	0x1fff22d8	Ready	0 (0)	0 B / 300 B		0x14 (1.5%)
5	Tmr Svc	0x1fff24d8	Blocked	5 (5)	36 B / 268 B	TmrQ (Rx)	0x0 (0.0%)
6	usb device	0x1fff3990	Suspended	5 (5)	0 B / 4.68 kB	USBNotify (Rx)	0x0 (0.0%)

Increase Task Stack (TODO 1)

- Increase Stack size in Application.c

100!

```
  vQueueAddToRegistry(semKbd, "Sem_Kbd");
#if 1  /*! \todo 1 Increase stack size by 100 */
  if (xTaskCreate(ButtonTask, "Buttons", configMINIMAL_STACK_SIZE+100, NULL, tskIDLE_PRIORITY+2, NULL) != pdPASS) {
#else
    if (xTaskCreate(ButtonTask, "Buttons", configMINIMAL_STACK_SIZE, NULL, tskIDLE_PRIORITY+2, NULL) != pdPASS) {
#endif
```

Increase Timer Stack

- Increase configMINIMAL_STACK_SIZE

FreeRTOSConfig.h | Application.c | hid_mouse.c | main.c

```c
#define configTICK_RATE_HZ                      ((TickType_t)100) /* frequency of tick interrupt */
#define configSYSTICK_USE_LOW_POWER_TIMER       0 /* If using Kinetis Low Power Timer (LPTMR) instead of SysTick tim
#define configSYSTICK_LOW_POWER_TIMER_CLOCK_HZ  1 /* 1 kHz LPO timer. Set to 1 if not used */
#if configPEX_KINETIS_SDK
/* The SDK variable SystemCoreClock contains the current clock speed */
  #include <stdint.h>
  extern uint32_t SystemCoreClock;
  #define configCPU_CLOCK_HZ                    SystemCoreClock /* CPU clock frequency */
  #define configBUS_CLOCK_HZ                    SystemCoreClock /* Bus clock frequency */
#else
  #define configCPU_CLOCK_HZ                    CPU_CORE_CLK_HZ /* CPU clock frequency */
  #define configBUS_CLOCK_HZ                    CPU_BUS_CLK_HZ /* Bus clock frequency */
#endif /* configPEX_KINETIS_SDK */
#define configSYSTICK_USE_CORE_CLOCK            1 /* System Tick is using core clock  */
#define configSYSTICK_CLOCK_DIVIDER             1 /* no divider */
#define configSYSTICK_CLOCK_HZ                  ((configCPU_CLOCK_HZ)/configSYSTICK_CLOCK_DIVIDER) /* frequency of s
#define configMINIMAL_STACK_SIZE                (100+50) /* stack size in addressable stack units */ /* \todo 1b Inc
/*-----------------------------------------------------------*/
/* Heap Memory */
```

Runtime Problem

Problem: Buttons task is using 100% of the CPU?

Task List (FreeRTOS) | Queue List (FreeRTOS) | Timer List (FreeRTOS) | Heap Usage (FreeRTOS)

TCB#	Task Name	Task Handle	Task State	Priority	Stack Usage	Event Object	Runtime
1	Buttons	0x1fff12e0	Running	2 (2)	4 B / 364 B		0x12c (100.0%)
2	App	0x1fff14e0	Ready	1 (1)	0 B / 324 B		0x0 (0.0%)
3	TzCtrl	0x1fff16e0	Ready	1 (1)	0 B / 324 B		0x0 (0.0%)
4	USB task	0x1fff2c10	Blocked	4 (4)	180 B / 4.75 kB		0x0 (0.0%)
5	IDLE	0x1fff2e10	Ready	0 (0)	0 B / 324 B		0x0 (0.0%)
6	Tmr Svc	0x1fff3010	Blocked	5 (5)	92 B / 268 B	TmrQ (Rx)	0x0 (0.0%)
7	usb device	0x1fff44c8	Suspended	5 (5)	0 B / 4.68 kB	USBNotify (Rx)	0x0 (0.0%)

Give Back CPU Time (TODO 2)

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

Application.c | FreeRTOSConfig.h

```c
static void ButtonTask(void *pvParameters) {
  (void)pvParameters;

  for(;;) {
    if (SW2IsPressed()) { /* SW2 pressed */
      vTaskDelay(pdMS_TO_TICKS(50)); /* debounce */
      while(SW2IsPressed()) {
        /* wait until released */
      }
      (void)xSemaphoreGive(semSW2); /* send message */
    }
    if (SW3IsPressed()) { /* SW3 pressed */
      vTaskDelay(pdMS_TO_TICKS(50)); /* debounce */
      while(SW3IsPressed()) {
        /* wait until released */
      }
      (void)xSemaphoreGive(semSW3); /* send message */
    }
    vTaskDelay(pdMS_TO_TICKS(10));
  }
}
```

Improved Runtime

System is now mostly IDLE which is expected

Task List (FreeRTOS) | Queue List (FreeRTOS) | Timer List (FreeRTOS) | Heap Usage (FreeRTOS)

TCB#	Task Name	Task Handle	Task State	Priority	Stack Usage	Event Object	Runtime
1	Buttons	0x1fff12e0	Blocked	2 (2)	0 B / 340 B		0x0 (0.0%)
2	App	0x1fff14e0	Blocked	1 (1)	0 B / 276 B		0x0 (0.0%)
3	TzCtrl	0x1fff16e0	Blocked	1 (1)	0 B / 260 B		0x0 (0.0%)
4	USB task	0x1fff2c10	Blocked	4 (4)	180 B / 4.75 kB		0x0 (0.0%)
5	IDLE	0x1fff2e10	Running	0 (0)	0 B / 308 B		0xe6 (100.0%)
6	Tmr Svc	0x1fff3010	Blocked	5 (5)	92 B / 268 B	TmrQ (Rx)	0x0 (0.0%)
7	usb device	0x1fff44c8	Suspended	5 (5)	0 B / 4.68 kB	USBNotify (Rx)	0x0 (0.0%)

Summary: Lab 3 - Kernel Awareness

- NXP FreeRTOS plugin for Eclipse
  - Tasks List
  - Queue List
  - Timer List
  - Heap Usage
- Uses Debugger/GDB
- Static, stop mode, need to halt/suspend application
- Inspect system status

Outline: Lab 4 - Segger SystemView

- **Problem**
  - Need to see what the application does while running

- **Goal**
  - FreeRTOS Trace with Segger SystemView

- **Lab**
  - Enable FreeRTOS Trace hooks
  - Collect trace information at runtime
  - Use view to identify problem during push button press
  - Fix problem

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 Tracealizer and with Segger SystemView

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

Enabling Segger SystemView Hooks (TODO 3)

- Enable in FreeRTOSConfig.h

```
#define configUSE_SEGGER_SYSTEM_VIEWER_HOOKS 1
```

- Build and debug/run application again
- Launch Segger SystemView Application

Start Recording Trace

- While application is running
- Press Start Recording Button
- Device: MK22FN512XXX12
- Press OK

Recording Data

- Watch while data is recorded
- Use stop button to stop recording
- Use Mouse scroll-wheel to zoom
- Can restart recording anytime while application is running

Segger SystemView - Events

- Up to 1 Mio events
- Events are created by the hooks and sent with RTT to SystemView application

Events

#	Timestamp	Context	Event	Detail
27556	00:00:44.209.993.200	Idle	Task Ready	App, runs after 45.4 us (341 cycles)
27557	00:00:44.210.001.067	Idle	Task Ready	Buttons, runs after 10.0 us (75 cycle…
27558	00:00:44.210.011.067	Buttons	Task Run	Runs for 14.9 us (112 cycles)
27559	00:00:44.210.018.933	Buttons	vTaskDelay	xTicksToDelay=1
27560	00:00:44.210.026.000	Buttons	Task Block	Buttons, Wait for timeout
27561	00:00:44.210.038.667	App	Task Run	Runs for 25.0 us (188 cycles)
27562	00:00:44.210.049.467	App	xQueueGenericRece…	xQueue=0x000005d2 pvBuffer=0x00000822…
27563	00:00:44.210.056.800	App	vTaskDelay	xTicksToDelay=5
27564	00:00:44.210.063.733	App	Task Block	App, Wait for timeout
27565	00:00:44.210.076.000	Scheduler	System Idle	Idle for 9.9278 ms (74 459 cycles)
27566	00:00:44.219.993.200	Idle	Task Ready	Buttons, runs after 10.6 us (80 cycle…
27567	00:00:44.220.003.867	Buttons	Task Run	Runs for 14.8 us (111 cycles)
27568	00:00:44.220.011.733	Buttons	vTaskDelay	xTicksToDelay=1
27569	00:00:44.220.018.667	Buttons	Task Block	Buttons, Wait for timeout
27570	00:00:44.220.031.200	Scheduler	System Idle	

Property	Detail
Target System	
Name	FRDM-K22F
OS	FreeRTOS
Device	Cortex-M4F
Cycle Freque…	7 500 000 Hz
Cycle Period	133 ns
Uptime	00:00:53.359.983.067
Recording	
Host Time	10 Mai 2016 21:19:50
Duration	00:00:44.199.983
Title	
Author	
Description	
Tasks	
Switch Count	5 484

Segger SystemView - Timeline

- Timeline: what is executed
- Scheduler and Idle handled separately
- CPU load with ’bins’: how many items to stack/show in a bar
- Use mouse scroll wheel to zoom in/out

Segger SystemViewer - Context

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

Contexts

Name	Type	Stack Information	Run Count	Frequency	Last Run Time	Min Run Time	Max Run Time	Total Run Time	Run Time/s	
Scheduler			800	124 Hz	0.0125 ms	0.0118 ms (#963)	0.0152 ms (#2245)	10.0397 ms	1.5558 ms	0.16%
Tmr Svc	@5	108 @ 0x1fff39e8	16	2 Hz	0.0649 ms	0.0646 ms (#1960)	0.0666 ms (#2244)	0.9761 ms	0.1298 ms	0.01%
usb device	@5	1185 @ 0x1fff3e38	0	0 Hz	0.0 ms			0.0 ms	0.0 ms	0.00%
USB task	@4	1173 @ 0x1fff2260	16	2 Hz	0.0136 ms	0.0134 ms (#715)	0.0136 ms (#404)	0.2030 ms	0.0272 ms	0.00%
Buttons	@2	145 @ 0x1fff1a08	640	100 Hz	0.0148 ms	0.0144 ms (#1874)	437.7204 ms (#2741)	1200.8077 ms	1.4846 ms	0.15%
App	@1	122 @ 0x1fff1d98	130	20 Hz	0.0250 ms	0.024 ms (#962)	0.0268 ms (#2173)	3.2429 ms	0.5014 ms	0.05%
Idle			640	100 Hz	9.9726 ms	2.1138 ms (#2748)	29.9610 ms (#2176)	6413.1002 ms	996.3009 ms	99.63%

Segger SystemView - Terminal

- Press SW2 / SW3: what happens?
- Application can print messages to terminal
- Messages shown in TimeLine and Terminal View
- Problem: Task is blocked while button is pressed?

Button Task: Adding Delay (TODO 4 & 5)

- **`Application.c`**
  - Add Delay of 50 ms during wait for release
- Rebuild, Debug and verify proper operation

```c
static void ButtonTask(void *pvParameters) {
  (void)pvParameters;
  for(;;) {
    if (SW2IsPressed()) { /* SW2 pressed */
      PrintButtonPressed(2, TRUE);
      vTaskDelay(pdMS_TO_TICKS(50)); /* debounce */
      while(SW2IsPressed()) { /* wait until released */
        vTaskDelay(pdMS_TO_TICKS(50)); /*! \todo 4 Add delay of 50 ms */
      }
      PrintButtonPressed(2, FALSE);
      (void)xSemaphoreGive(semSW2); /* send message to toggle USB */
      (void)xSemaphoreGive(semMouse); /* send message to toggle USB */
      (void)xSemaphoreGive(semKbd); /* send message to toggle USB */
    }
    if (SW3IsPressed()) { /* SW3 pressed */
      PrintButtonPressed(3, TRUE);
      vTaskDelay(pdMS_TO_TICKS(50)); /* debounce */
      while(SW3IsPressed()) { /* wait until released */
        vTaskDelay(pdMS_TO_TICKS(50)); /*! \todo 5 Add delay of 50 ms */
      }
      PrintButtonPressed(3, FALSE);
      (void)xSemaphoreGive(semSW3); /* send message */
      (void)xSemaphoreGive(semLED); /* send message to change LED */
    }
    vTaskDelay(pdMS_TO_TICKS(10)); /*! \todo 2 Add delay of 10 ms */
  }
}
```

Summary: Lab 4 – Segger SystemView

- Instrumented RTOS to generate trace
- Trace is streamed in real-time with Segger RTT
- Segger SystemView to collect and inspect data
- Ability to see what is going on
- Application message logging

- **Further Information:**
  - http://segger.com/
  - https://mcuoneclipse.com/2015/11/16/segger-systemview-realtime-analysis-and-visualization-for-freertos/
  - https://mcuoneclipse.com/2015/07/07/using-segger-real-time-terminal-rtt-with-eclipse/

Outline: Lab 5 - Percepio Tracealizer

- **Problem**
  - Need advanced analysis tools to identify problem like memory leaks or performance problems

- **Goal**
  - Using Percepio Tracealizer to identify memory leaks and performance problems

- **Lab**
  - Turn on Percepio Trace hooks in FreeRTOS
  - Collect Trace information
  - Fixing memory leak and semaphore problem

How It Works…

- Trace Hooks in FreeRTOS, similar as for Segger SystemView
  - 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 Hooks (TODO 6)

- FreeRTOSConfig.h
  - Trace facility enabled
  - Enable (Percepio) Trace Hooks
  - Disable Segger Trace Hooks
- Only on tracing method can be enabled at the same time
- Build and Debug/Run

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

TzCtrl Task

- Percepio Trace Library adds ‘TzCtrl’ Task
- Command/Control task
- Do not starve TzCtrl task!

Task List (FreeRTOS) | Queue List (FreeRTOS) | Timer List (FreeRTOS) | Heap Usage (FreeRTOS)

TCB#	Task Name	Task Handle	Task State	Priority	Stack Usage	Event Object	Runtime
1	Buttons	0x1fff12e0	Blocked	2 (2)	0 B / 340 B		0x0 (0.0%)
2	App	0x1fff14e0	Blocked	1 (1)	0 B / 276 B		0x0 (0.0%)
3	TzCtrl	0x1fff16e0	Blocked	1 (1)	0 B / 260 B		0x0 (0.0%)
4	USB task	0x1fff2c10	Blocked	4 (4)	180 B / 4.75 kB		0x0 (0.0%)
5	IDLE	0x1fff2e10	Running	0 (0)	0 B / 308 B		0xe6 (100.0%)
6	Tmr Svc	0x1fff3010	Blocked	5 (5)	92 B / 268 B	TmrQ (Rx)	0x0 (0.0%)
7	usb device	0x1fff44c8	Suspended	5 (5)	0 B / 4.68 kB	USBNotify (Rx)	0x0 (0.0%)

Percepio Debugger Selection

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

J-Link Settings

- J-Link Settings Tab
  - Debugger (USB Port)
  - 4000 kHz
  - Default debugger interface
- If target device MK22FN512xxx12 is not selected:
  - Button Select Device…

Device Selection

- Select device
  - Freescale
  - 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

Stimulating Application

- Shows CPU load
- Red line: (USB) Interrupts
- Trace Session Information
  - Number of events
  - Number of events/second
  - Recording time

- Press SW2 to toggle USB
- Press SW3 to change LED Color
  - After ~8 times SWx, color does not change any more?
- Press Stop Recording

Saving and Viewing Trace

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

Inspecting Events

- Zoom into Data
  - Zoom In Button
  - Select and context menu
- Enable all items in View Filter

Communication Flow

- Menu View > Communication Flow
- Buttons Task uses ShellQueue
- Nobody is consuming from queue?

Blocked Shell Queue

- Button task tries to send to queue
- Blocked because of queue full
- Select entry, and Press Show in Trace

Timestamp	Actor	Event	Block time	Status	Size	Queue
9.170.074	Buttons	xQueueSend		Sent post #1	1	1
9.420.074	Buttons	xQueueSend		Sent post #2	2	1 2
11.620.074	Buttons	xQueueSend		Sent post #3	3	1 2 3
11.923.430	Buttons	xQueueSend		Sent post #4	4	1 2 3 4
15.710.074	Buttons	xQueueSend		Sent post #5	5	1 2 3 4 5
16.010.074	Buttons	xQueueSend		Sent post #6	6	1 2 3 4 5 + 1
19.480.074	Buttons	xQueueSend		Sent post #7	7	1 2 3 4 5 + 2
19.680.074	Buttons	xQueueSend		Sent post #8	8	1 2 3 4 5 + 3
20.090.074	Buttons	xQueueSend		Sent post #9	9	1 2 3 4 5 + 4
20.340.074	Buttons	xQueueSend		Sent post #10	10	1 2 3 4 5 + 5
21.040.074	Buttons	xQueueSend		Sent post #11	11	1 2 3 4 5 + 6
21.240.074	Buttons	xQueueSend		Sent post #12	12	1 2 3 4 5 + 7
21.940.074	Buttons	xQueueSend		Sent post #13	13	1 2 3 4 5 + 8
22.240.074	Buttons	xQueueSend		Sent post #14	14	1 2 3 4 5 + 9
22.860.074	Buttons	xQueueSend		Sent post #15	15	1 2 3 4 5 + 10
23.110.074	Buttons	xQueueSend		Sent post #16	16	1 2 3 4 5 + 11
24.060.079	Buttons	xQueueSend		Trying to send...	16	1 2 3 4 5 + 11

Blocked Button Task

- Sending to Queue Blocks Button Task
- No '*Next Start*' for Button Task

Enable Debug Print (TODO 7)

- Application.c: Debug printing is disabled
  - Does not consume from Shell Queue
- Set DEBUG_PRINT macro to 1
  - `#define DEBUG_PRINT  1`
- Terminate, Build and Debug again
- Start Recording Trace again
- Verify that color changes now properly

```
#define DEBUG_PRINT  0 /*! \todo 7 Set macro to 1 */

static void AppTask(void *pvParameters) {
  (void)pvParameters;
  for(;;) {
#if DEBUG_PRINT && PL_CONFIG_HAS_SHELL_QUEUE
    {
      const unsigned char *msg;

      msg = SQUEUE_ReceiveMessage();
      if (msg!=NULL) {
        PrintDebugMessage(msg);
      }
    }
#endif /* PL_CONFIG_HAS_SHELL_QUEUE */
    vTaskDelay(pdMS_TO_TICKS(50));
  }
}
```

Fixed Communication Flow

- Stop Recording and inspect trace
- Check Communication Flow again
  - App is now consuming from ShellQueue
- Any other possible problems?

Heap Memory

Tracealyzer for FreeRTOS - trace_2016-05-11_11.2
File
Find
View
Bookmarks
J-Link
Help
New Trace Window
Horizontal Trace View
CPU Load Graph
Kernel Object Utilization
Scheduling Intensity
Kernel Call Intensity
Kernel Blocking Times
Actor Instance Graphs
Memory Heap Utilization
usb devic
User Event Signal Plot
Communication Flow
Statistics Report
Event Log
User Event Log
Kernel Object List
Trace Details
Trace View Settings

Memory Heap Utilization
Views
Resolution
Zoom
Customize
Memory Heap Utilization
Heap
2200
2000
1800
1600
1400
1200
1000
800
600
400
200
0
10.000.000
20.000.000
30.000
malloc(40) returned 0x1FFF5430 at 29.370.057 (s.ms.µs)
Returned Success
Zoom In
Zoom Out
Show In Main View
Show In Kernel Object History
Time: 29.170.057, Value: 2040

What is consuming Memory?

- ‘Show in Main View’ or double-click on red box
- Buttons Task allocates memory for Shell Queue
- Where should memory be released?

```c
#define SQUEUE_LENGTH      16 /* items in queue */
#define SQUEUE_ITEM_SIZE   sizeof(char*) /* each item is a char pointer to a string, allocated in the heap */

void SQUEUE_SendString(const unsigned char *str) {
  unsigned char *ptr;
  size_t bufSize;

  bufSize = UTIL1_strlen((const char*)str)+1;
  ptr = pvPortMalloc(bufSize);
  UTIL1_strcpy(ptr, bufSize, str);
  if (xQueueSendToBack(SQUEUE_Queue, &ptr, portMAX_DELAY)!=pdPASS) {
    for(;;){} /* ups? */
  }
}
```

Fixing Memory Leak (TODO 8)

- Application.c
  - Function `PrintDebugMessage()`
  - Call `vPortFree((void*)msg)` for memory previously allocated
- Terminate, Build, Debug, Collect Trace again…

```
static void PrintDebugMessage(const unsigned char *msg) {
  DbgConsole_Printf((char*)msg);
  vPortFree((void*)msg); /*! \todo 8 Fix Memory leak */
}
```

Fixed Memory Leak

CPU Load

- CPU load high during USB activity?
- Do we have a performance problem?

CPU Load Graph

- Double click on task bar to inspect it

USB Task and Semaphore

- Problem: Semaphore usage

Fixing USB Semaphore usage (TODO 9)

- File hid_mouse.c
- Check Semaphore only once
- Build, Debug and collect trace again…

```
#if 1 /*! \todo 9 use correct implementation checking semaphore only once */
    if (xSemaphoreTake(semMouse, 0)==pdPASS) {
      mouseEnabled = !mouseEnabled; /* toggle */
    }
```

Improved Semaphore Usage

- Reduced CPU load
- Semaphore taken only once

Summary: Lab 5 – Percepio Tracealizer

- Trace collection at runtime
- Interrupt Logging
- Advanced views to deeply inspect system behavior
- Performance analysis
- Event and RTOS call tracking
- Memory/Heap allocation over time

- **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-freertostrace-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-freertostrace-v2-2-2-released/

Session Closing – FreeRTOS Solutions for Kinetis MCU's

- Hardware, with built in debug interface
  - Inexpensive and versatile FRDM boards
- Runtime software, open source and free of charge
  - Kinetis SDK with integrated FreeRTOS
  - Kinetis USB stack integrated in Kinetis SDK
- IDE, unlimited and free of charge
  - Kinetis Design Studio with Kernel Awareness
  - build, debug and stop-mode system debugging
- FreeRTOS tracing solution, collecting data at runtime
  - Segger SystemView: free of charge, cool features with application logging
  - Percepio Tracealizer: free and professional version, many advanced views

For Further Information

- Kinetis Design Studio IDE: www.nxp.com/kds
- Kinetis SDK: www.nxp.com/ksdk
- FreeRTOS: www.freertos.org
- Segger: www.segger.com
- Percepio: www.percepio.com
- Examples and Tutorials: 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 µVision 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.