Jump Starting Code Development to Minimize Defects

Class 4: Mastering Application Tracing

December 13, 2018

Jacob Beningo

DesignNews

CEC CONTINUING EDUCATION CENTER

Presented by: Digi-Key ELECTRONICS

Course Overview

Topics:

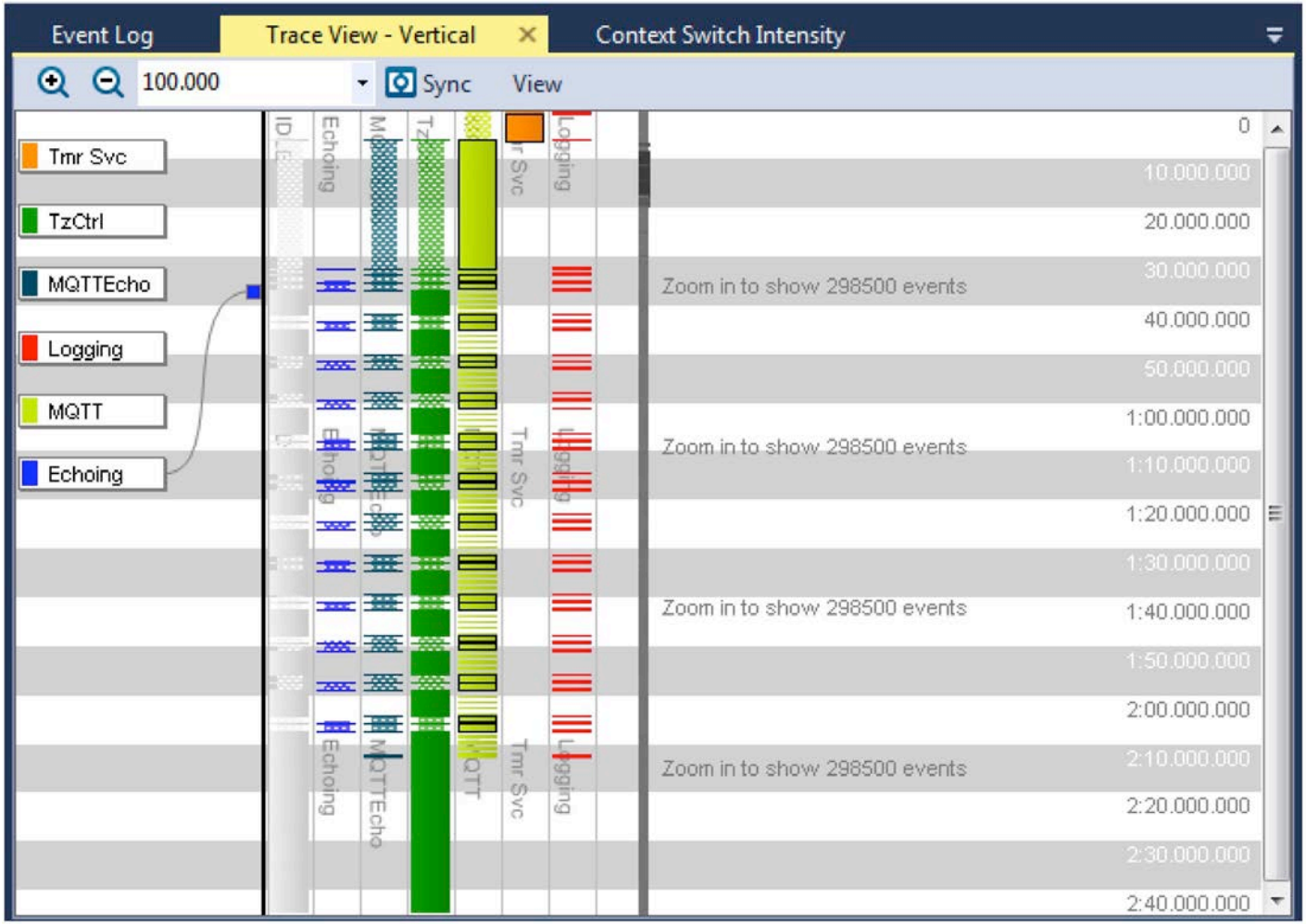
- Errors, Defects and Bugs
- Managing Design Processes
- The Jump Start Development Process
- **Mastering Application Tracing**
- Advanced Techniques

Session Overview

- Tracing
- Finding RTOS Application Bugs
- Analyzing Application Events
- Communication Flow
- Memory Leaks
- User Events
- State Machines

Presented by:

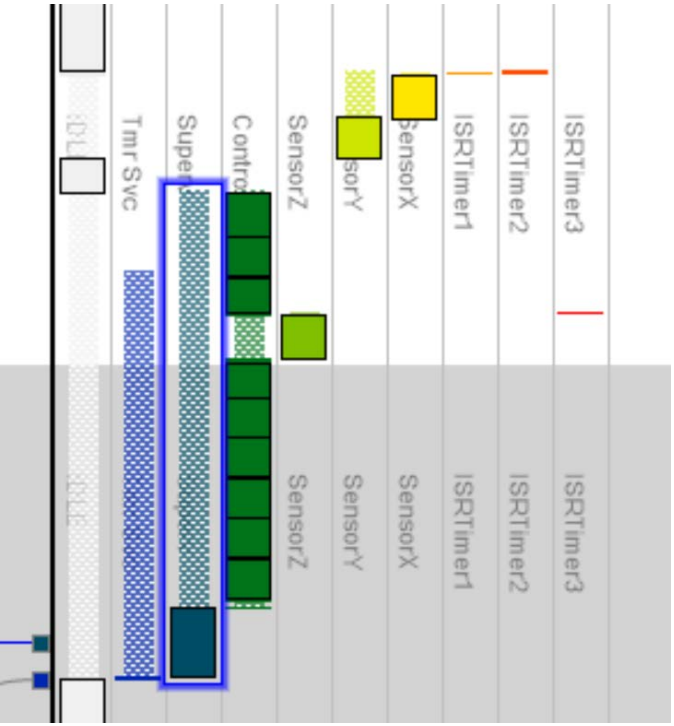
Analyze Trace Results



Analyze Trace Results

Actor	Count	CPU Usage	Execution Time			Response Time		
			Min	Avg	Max	Min	Avg	Max
IDLE	1	45.521	1:10.124.584	1:10.124.584	1:10.124.584	2:27.845.514	2:27.845.514	2:27.845.514
Echoing	13	0.004	43	479	540	5.313	1.719.220	1.865.440
MQTTEcho	13	0.005	465	568	1.735	409.187	5.181.579	30.919.700
TzCtrl	7088	0.142	31	31	31	31	10.905	26.568.101
MQTT	76	49.806	66	1.009.647	26.563.786	209.069	1.093.060	26.577.236
Tmr Svc	1	4.013	6.183.176	6.183.176	6.183.176	6.218.355	6.218.355	6.218.355
Logging	174	0.509	2.941	4.505	8.585	2.955	4.518	8.599

RTOS Problems – Thread Starvation

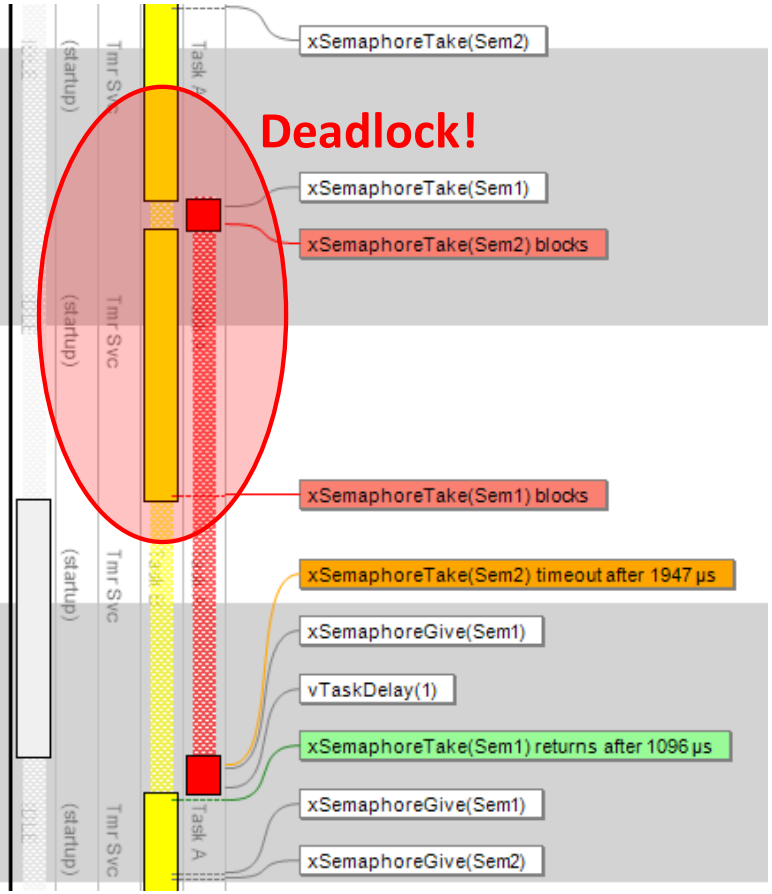


Thread Starvation occurs when a low priority thread is rarely executed due to higher priority tasks always using the CPU

Solutions

- Using RMA to properly schedule tasks
  - High priority tasks have shortest execution time
- Avoid polling and delay techniques
- Preemption Threshold
- Dynamic Thread Prioritization

RTOS Problems - Deadlock

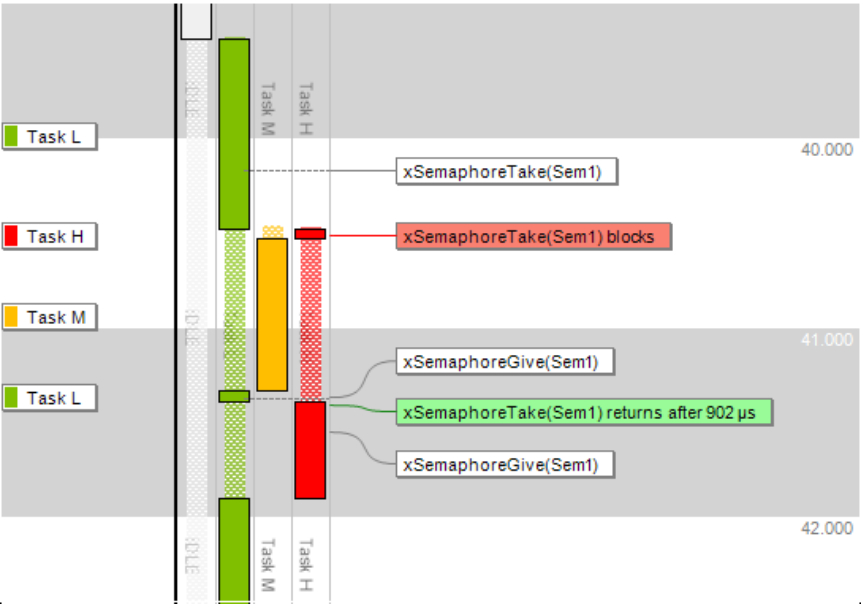


Deadlock occurs when two tasks need access to two or more resources to proceed but each task has only one resource.

Solutions

- Tasks that require two or more resources should acquire them in the same order
- Use timeouts to release the first resource if the second can't be acquired

RTOS Problems –Priority Inversion



Priority Inversion occurs when a high priority task is blocked by a lower priority task

Solutions:

- Use mutex with priority inheritance
  - Lower priority task is temporarily promoted to blocked task
- Properly select task priorities using formal methods such as RMA

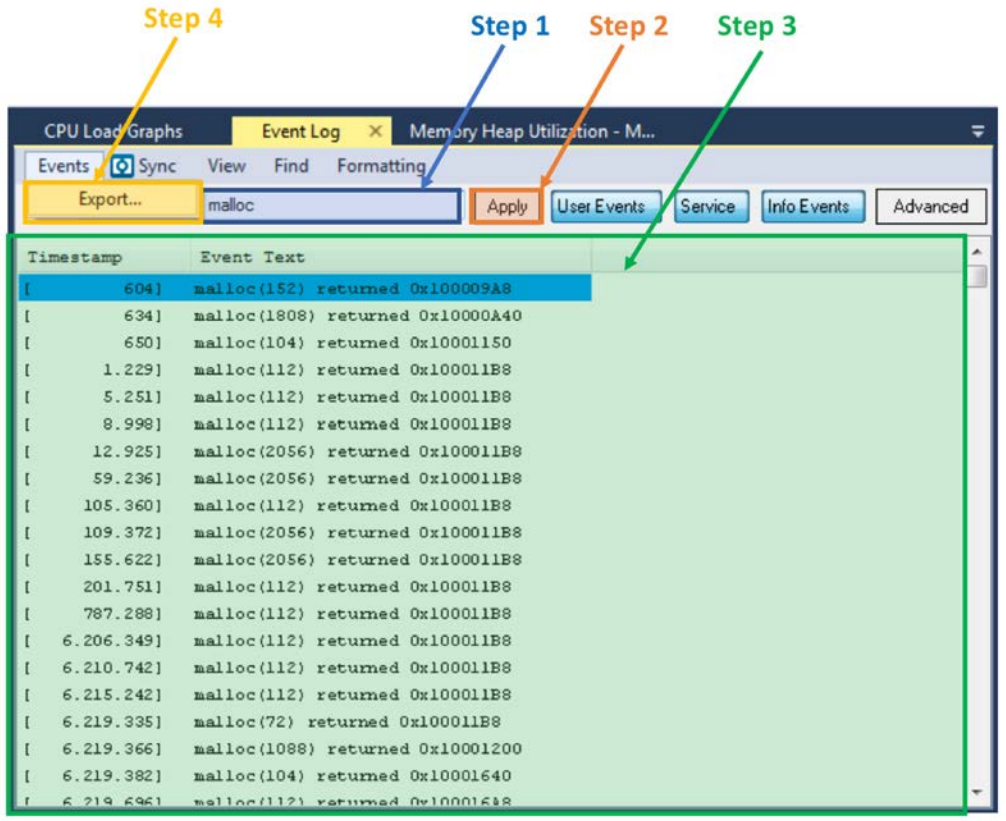
Analyzing Application Events

Step #1 - In the Event Log, enter the text phrase that you want to filter and export. This could be something like malloc, free or some other event text that is being generated in the data stream.

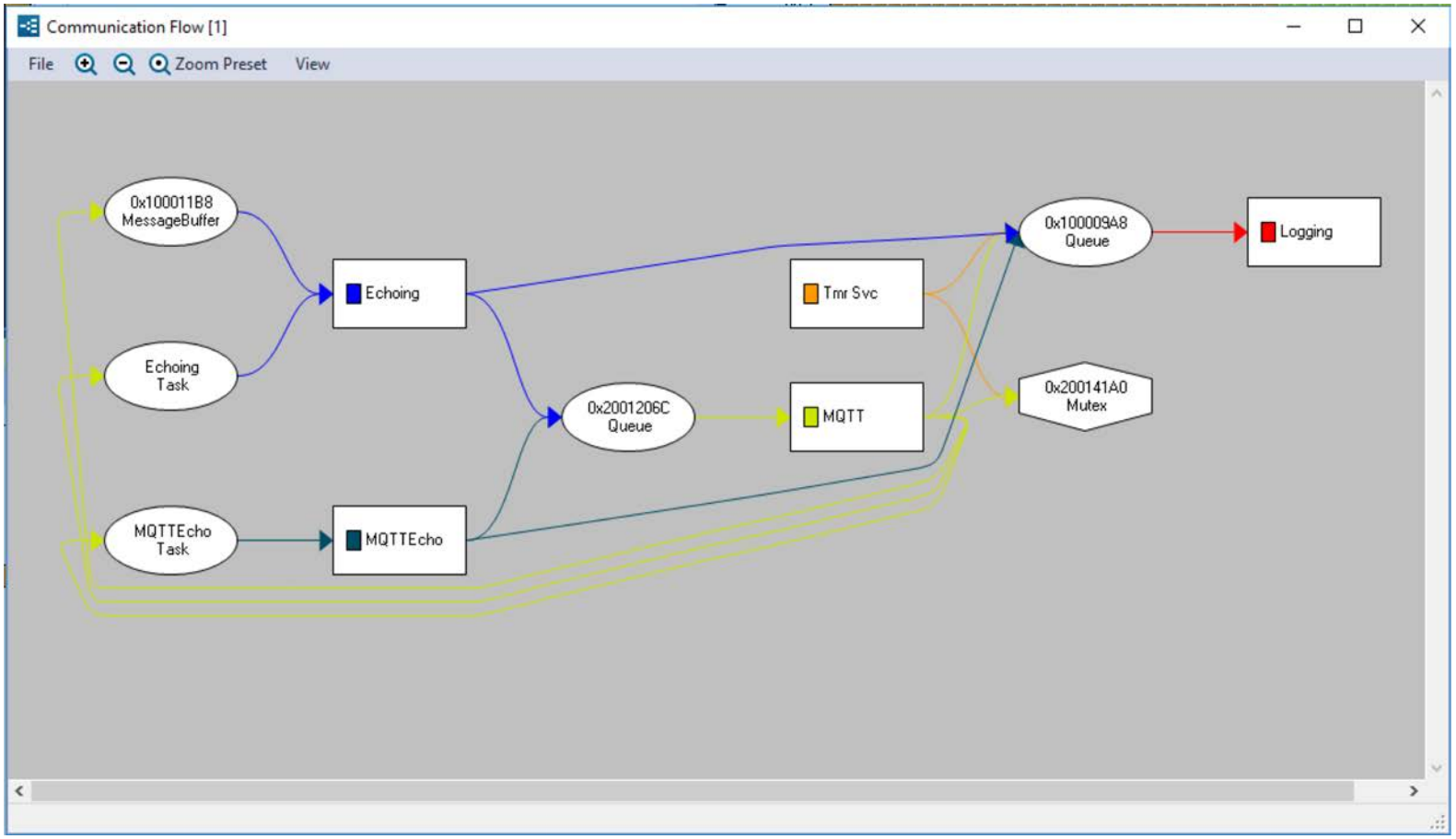
Step #2 – Click apply to filter the event log.

Step #3 – Verify that the Event Log has the information that you are looking for.

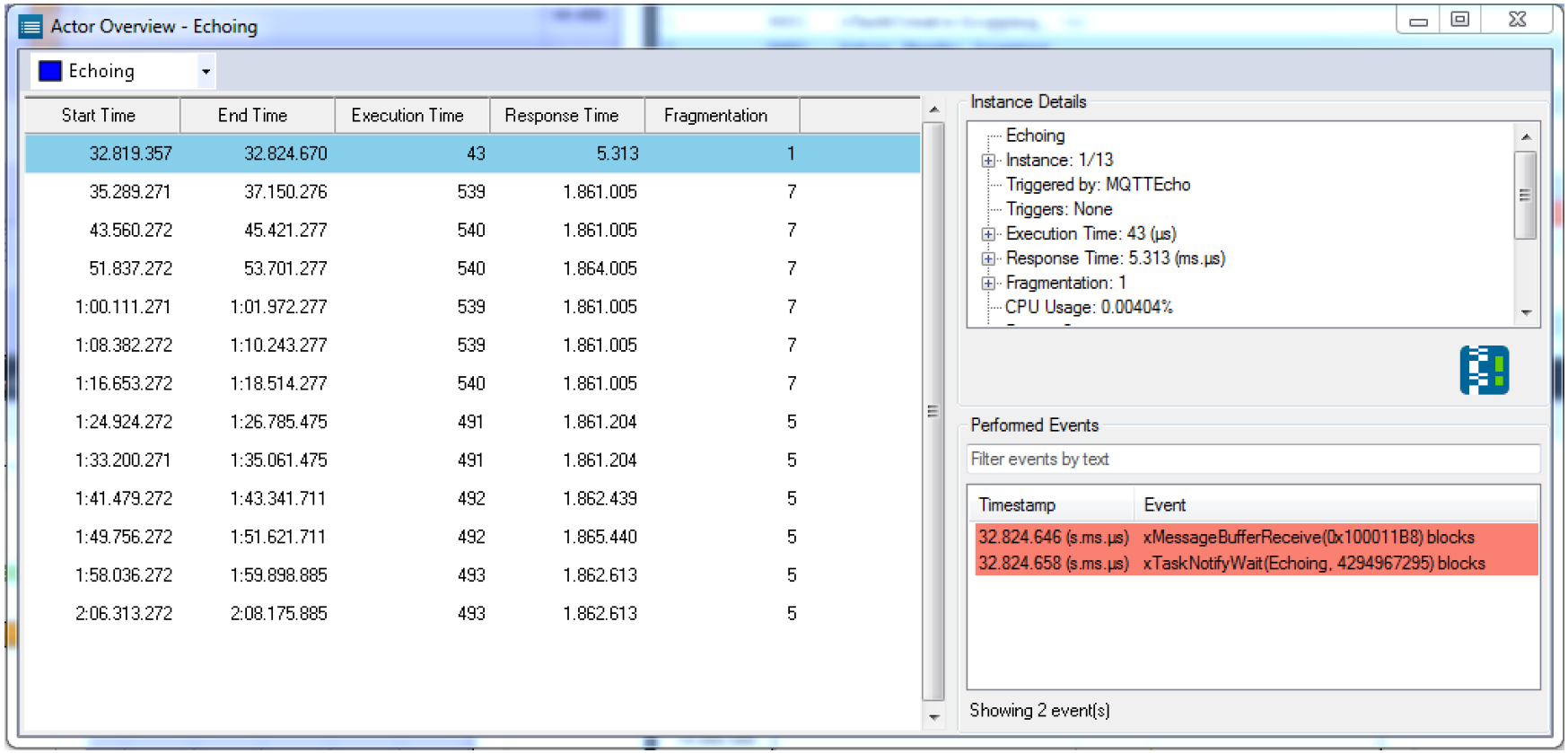
Step #4 – Click the Events->Export button and save the file.



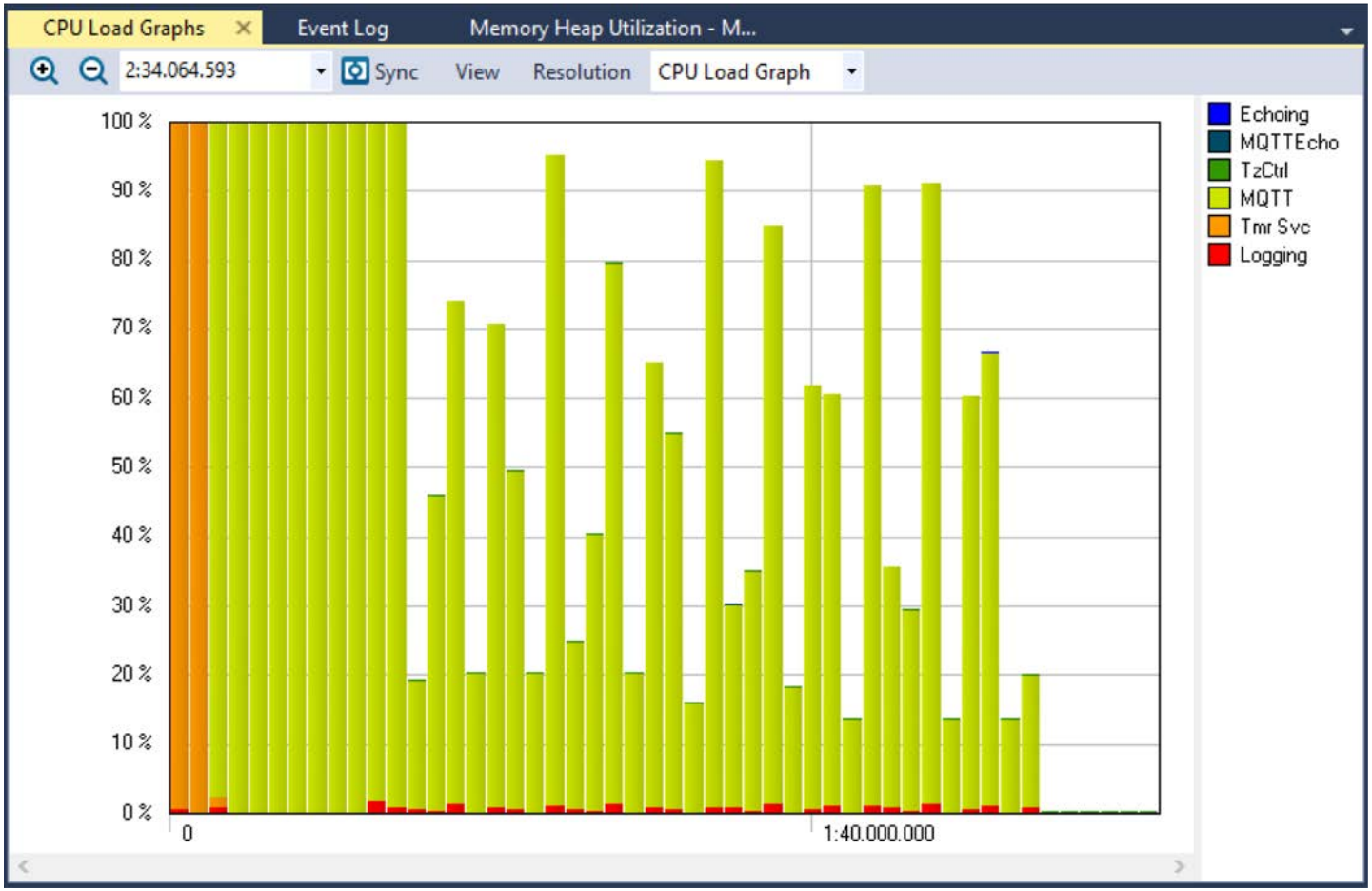
Analyze Communication Flow



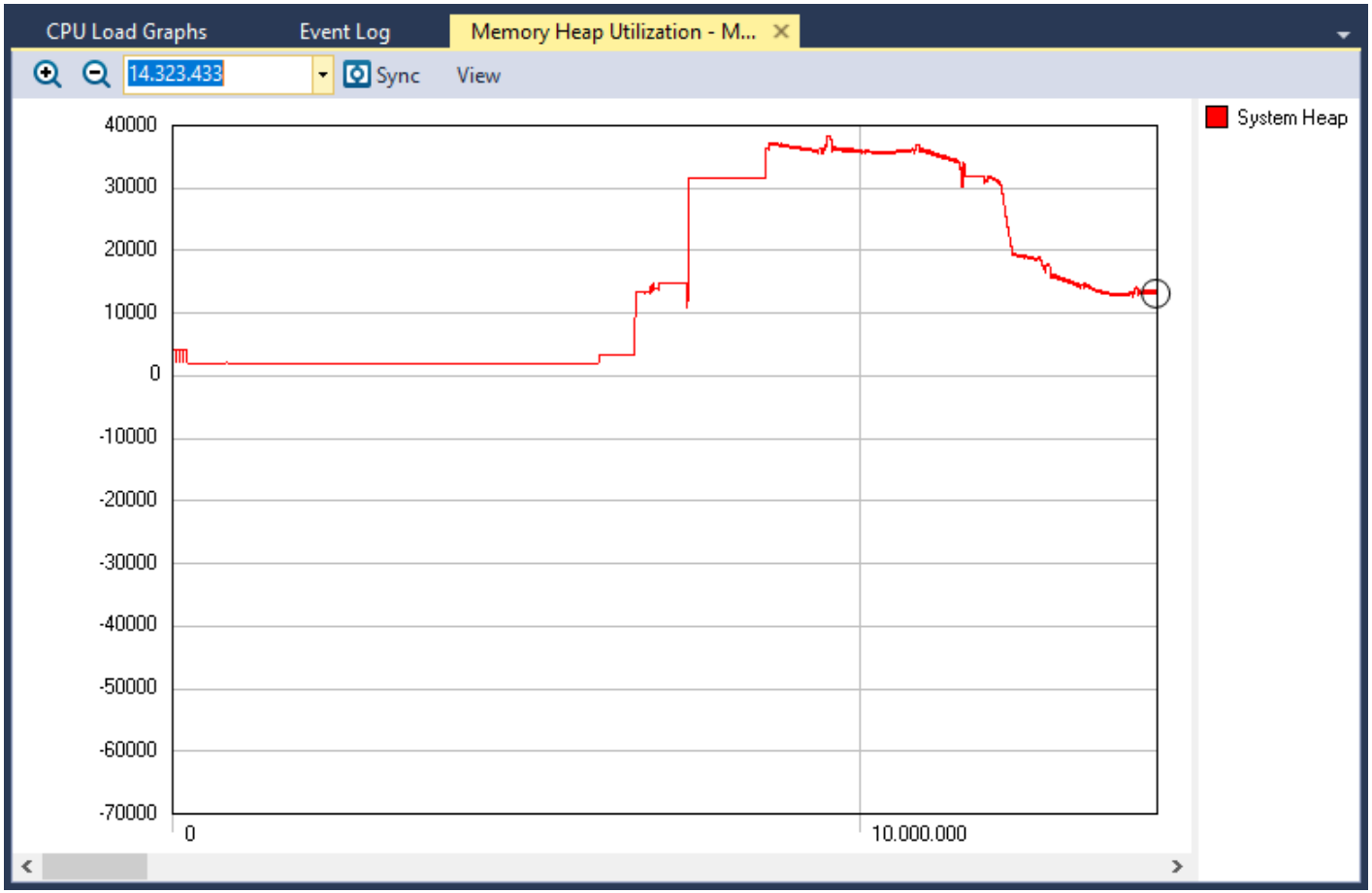
Analyze Communication Flow



CPU Utilization



Hunting for Memory Leaks

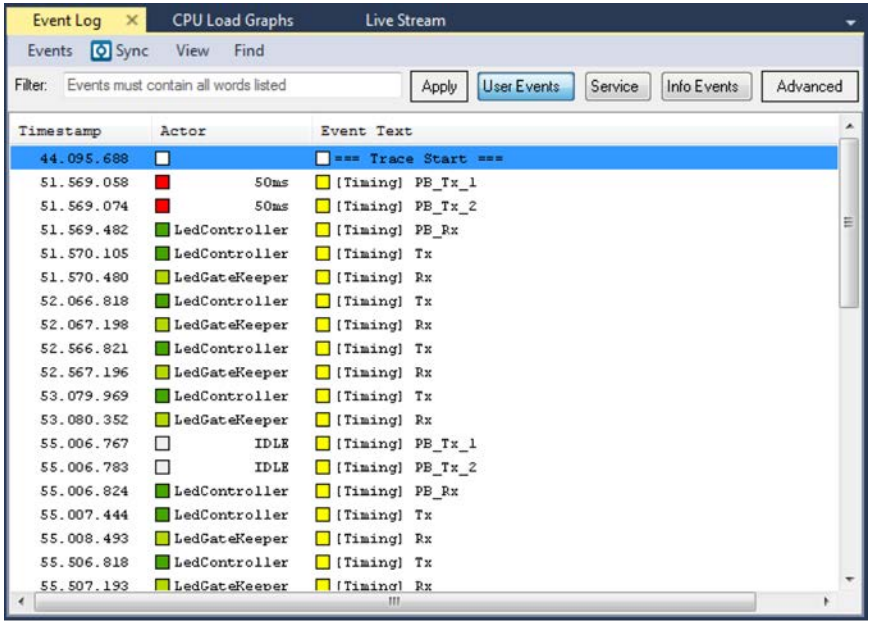


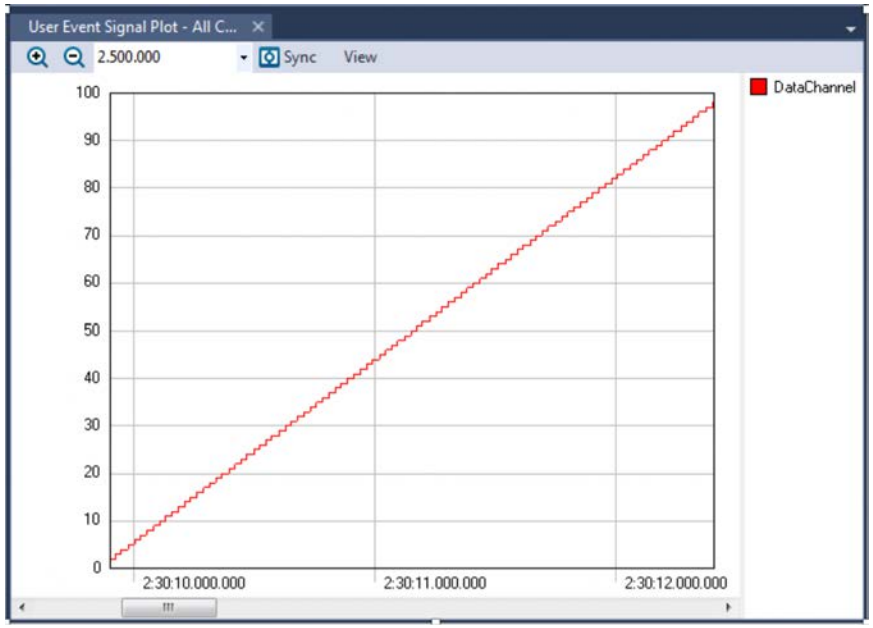
Creating User Channels

traceString MyChannel = xTraceRegisterString("DataChannel");

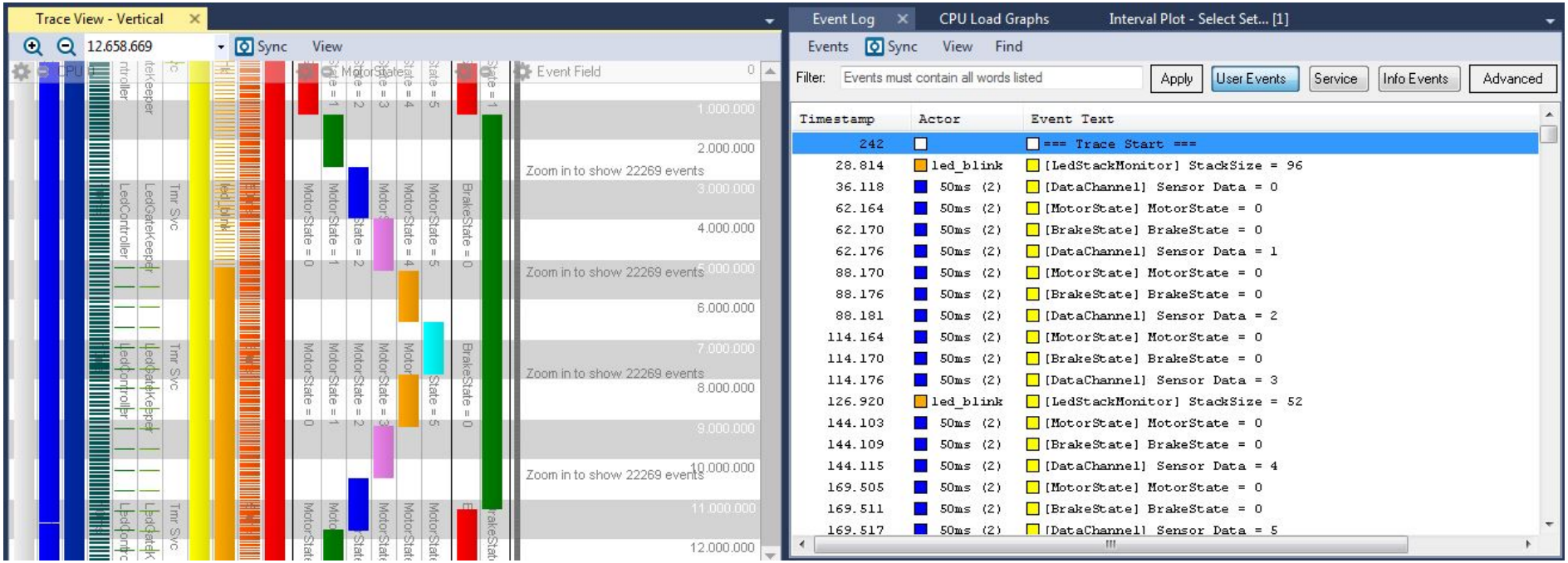
vTracePrint(MyChannel, "Button Pressed!");

vTracePrintF(MyChannel, "Sensor Data = %d", SensorData);

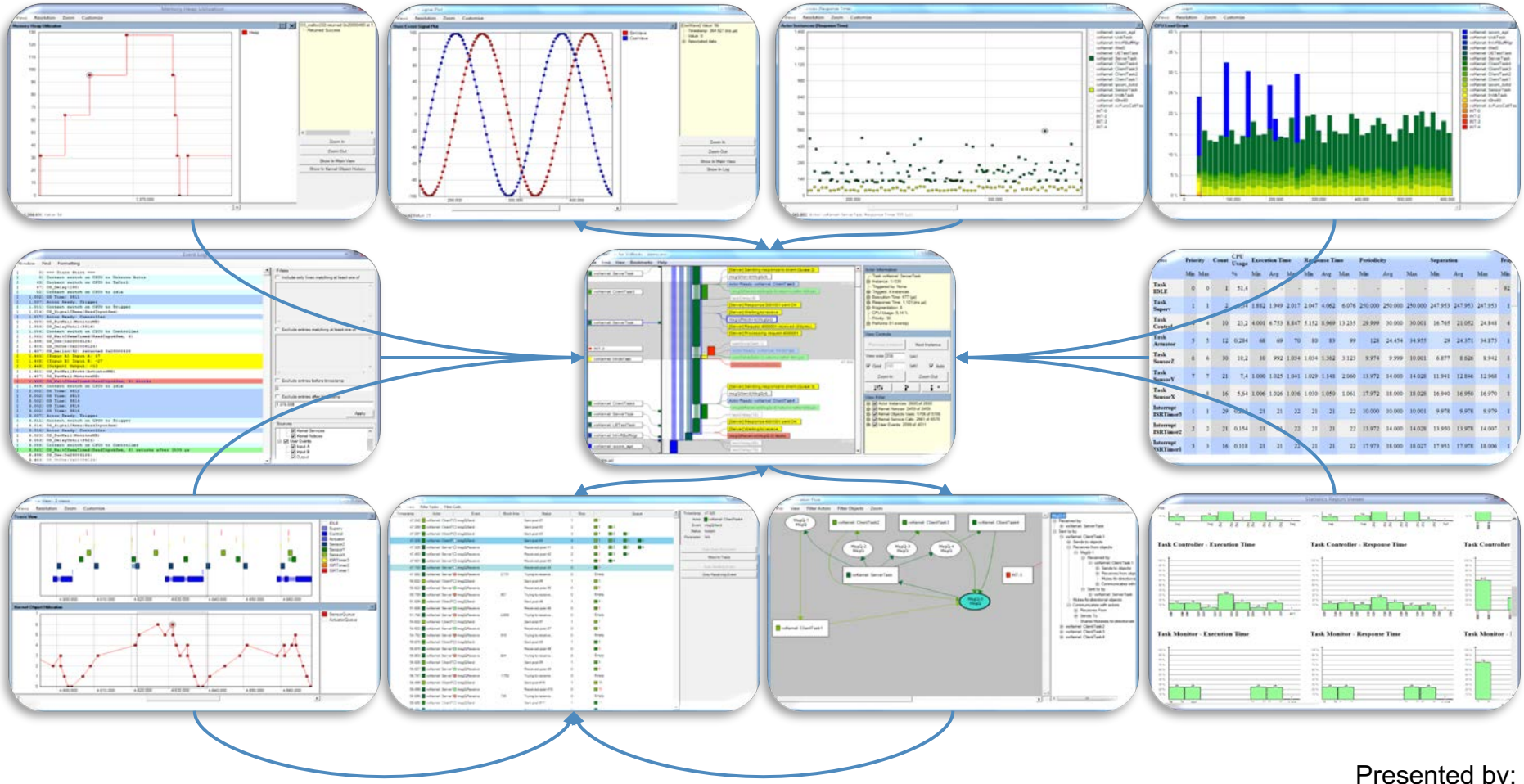




User State Machines



Examine Different Views



Additional Resources

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

From www.beningo.com under

- Blog > CEC – Jump Starting Code Development to Minimize Defects