

Jump Starting Code Development to Minimize Defects

Class 4: Mastering Application Tracing

December 13, 2018
Jacob Beningo

Course Overview

Topics:

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

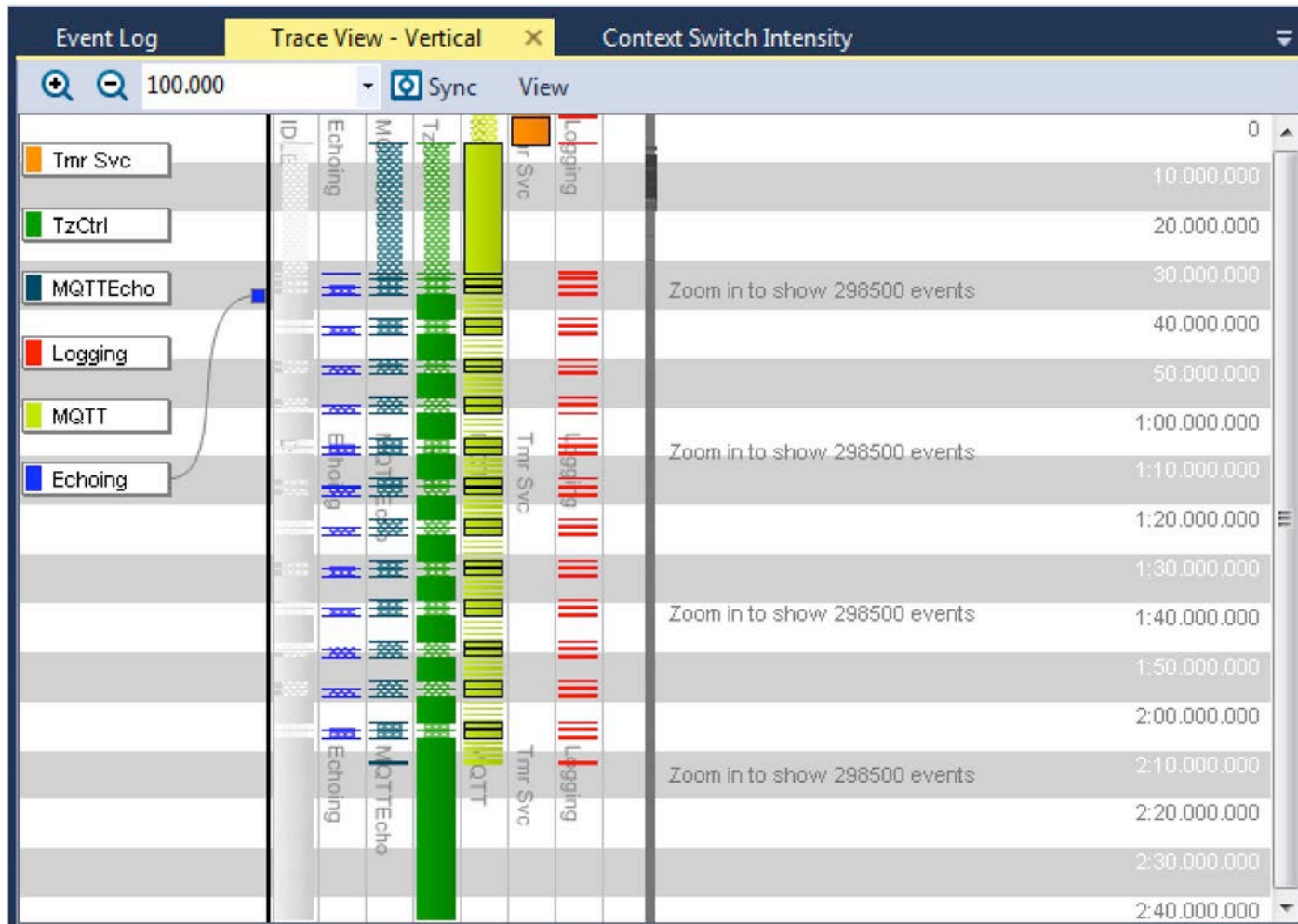
Session Overview

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



Presented by:

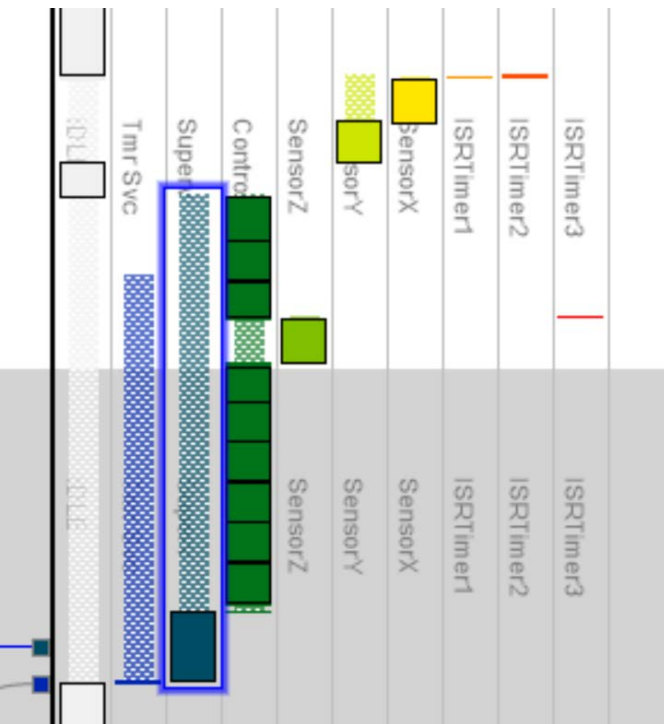
Analyze Trace Results



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

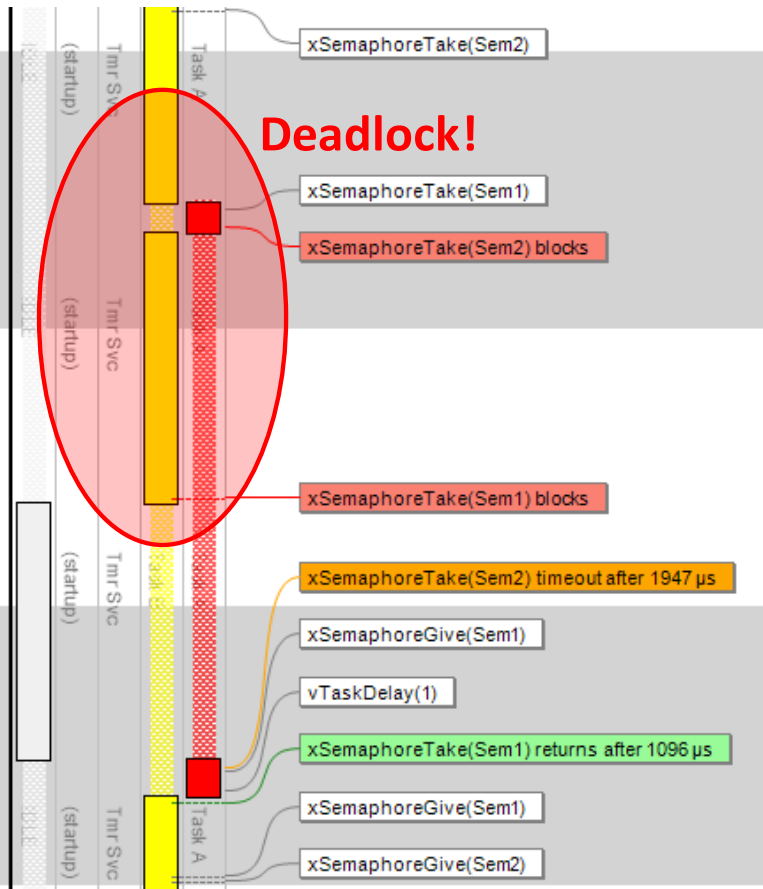
- Using RMA to properly schedule tasks
 - High priority tasks have shortest execution time
- Avoid polling and delay techniques
- Preemption Threshold
- Dynamic Thread Prioritization

RTOS Problems - Deadlock

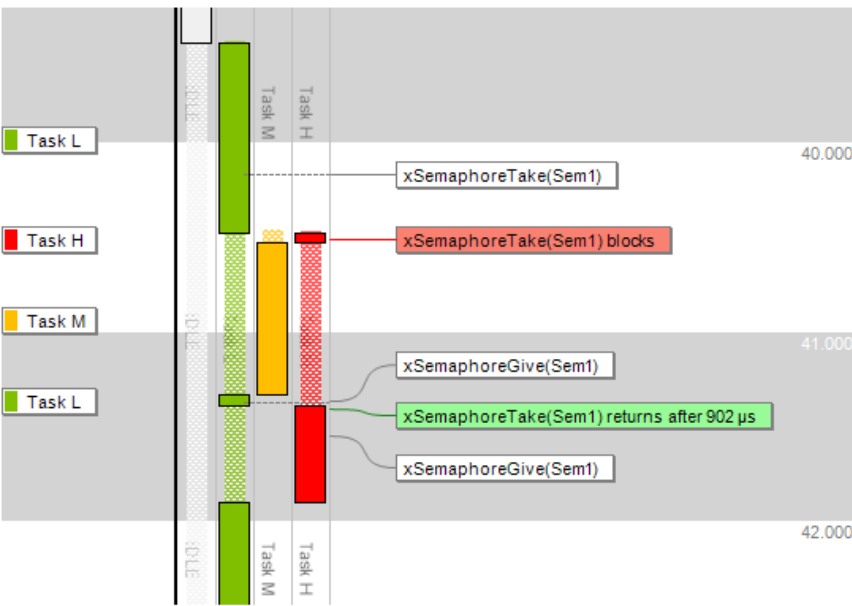
Deadlock occurs when two tasks need access to two or more resources to proceed but each task has only one resource.

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

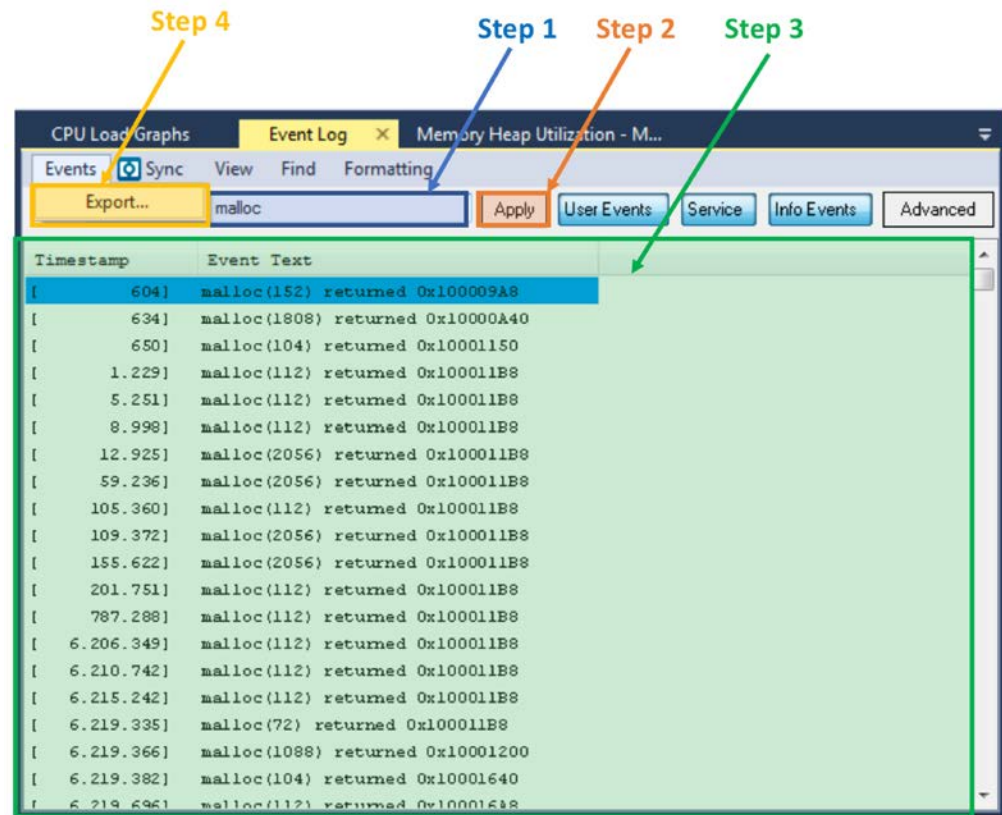
Analyzing Application Events

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.

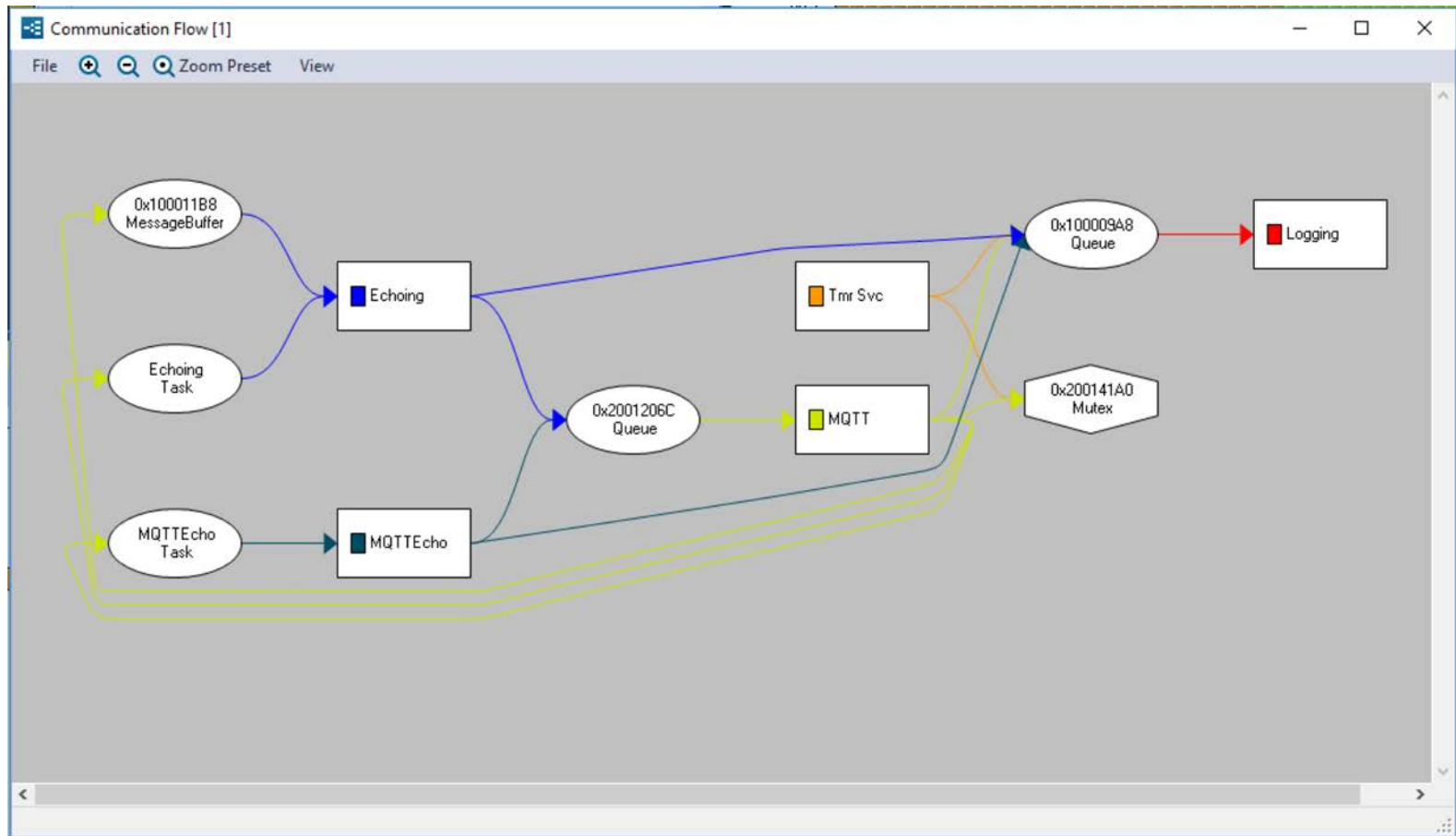
Step #2 – Click apply to filter the event log.

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

The screenshot displays the 'Actor Overview - Echoing' window in Visual Studio Performance Analyzer. The main table lists 13 instances of the 'Echoing' actor, with the first instance selected. The right-hand pane shows 'Instance Details' for the selected instance, including its execution time, response time, fragmentation, and CPU usage. Below this, the 'Performed Events' pane shows two events related to the selected instance.

| Start Time | End Time | Execution Time | Response Time | Fragmentation |
|--------------|--------------|----------------|---------------|---------------|
| 32.819.357 | 32.824.670 | 43 | 5.313 | 1 |
| 35.289.271 | 37.150.276 | 539 | 1.861.005 | 7 |
| 43.560.272 | 45.421.277 | 540 | 1.861.005 | 7 |
| 51.837.272 | 53.701.277 | 540 | 1.864.005 | 7 |
| 1:00.111.271 | 1:01.972.277 | 539 | 1.861.005 | 7 |
| 1:08.382.272 | 1:10.243.277 | 539 | 1.861.005 | 7 |
| 1:16.653.272 | 1:18.514.277 | 540 | 1.861.005 | 7 |
| 1:24.924.272 | 1:26.785.475 | 491 | 1.861.204 | 5 |
| 1:33.200.271 | 1:35.061.475 | 491 | 1.861.204 | 5 |
| 1:41.479.272 | 1:43.341.711 | 492 | 1.862.439 | 5 |
| 1:49.756.272 | 1:51.621.711 | 492 | 1.865.440 | 5 |
| 1:58.036.272 | 1:59.898.885 | 493 | 1.862.613 | 5 |
| 2:06.313.272 | 2:08.175.885 | 493 | 1.862.613 | 5 |

Instance Details

- Echoing
 - Instance: 1/13
 - Triggered by: MQTTEcho
 - Triggers: None
 - Execution Time: 43 (µs)
 - Response Time: 5.313 (ms,µs)
 - Fragmentation: 1
 - CPU Usage: 0.00404%

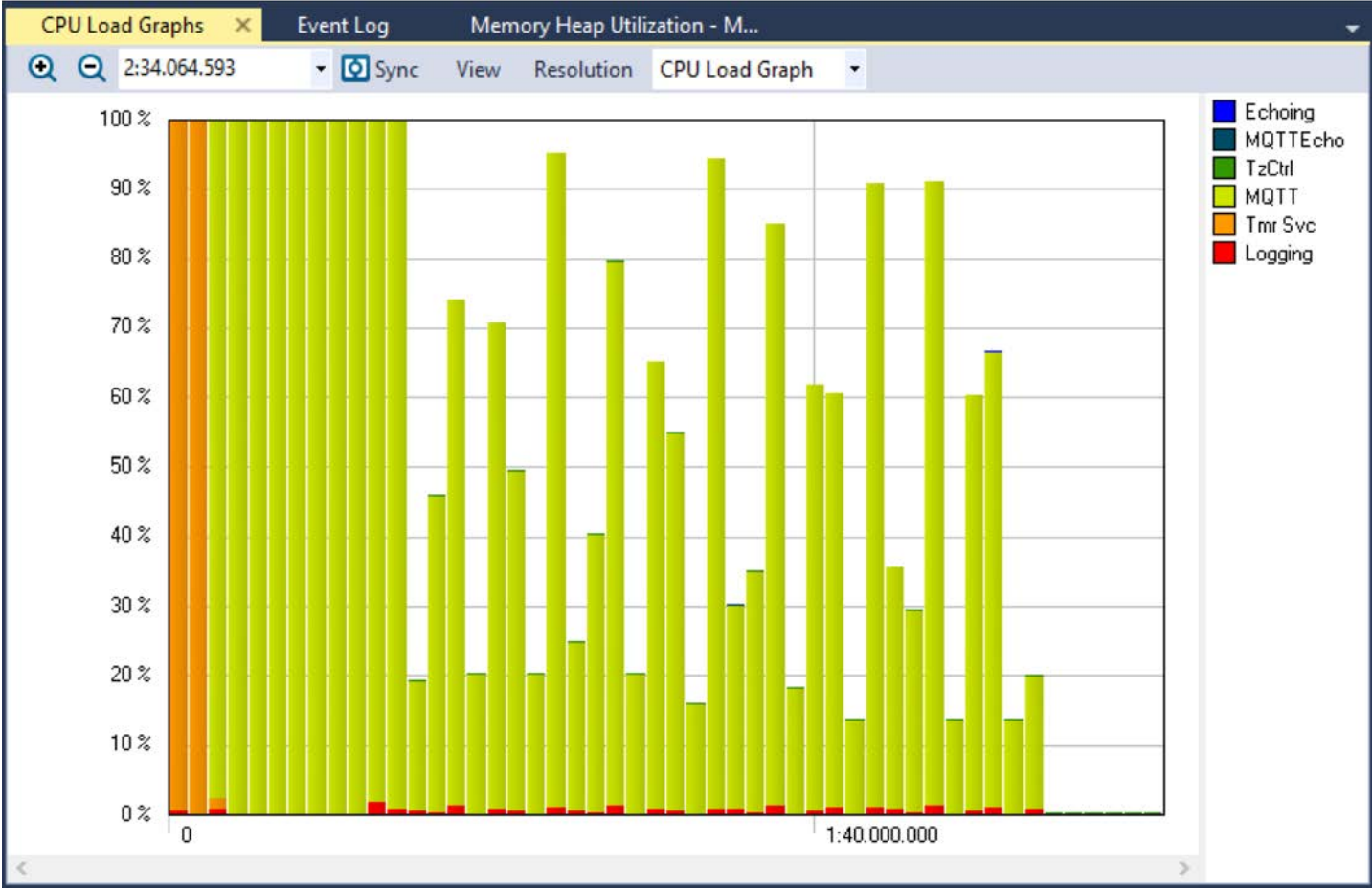
Performed Events

Filter events by text

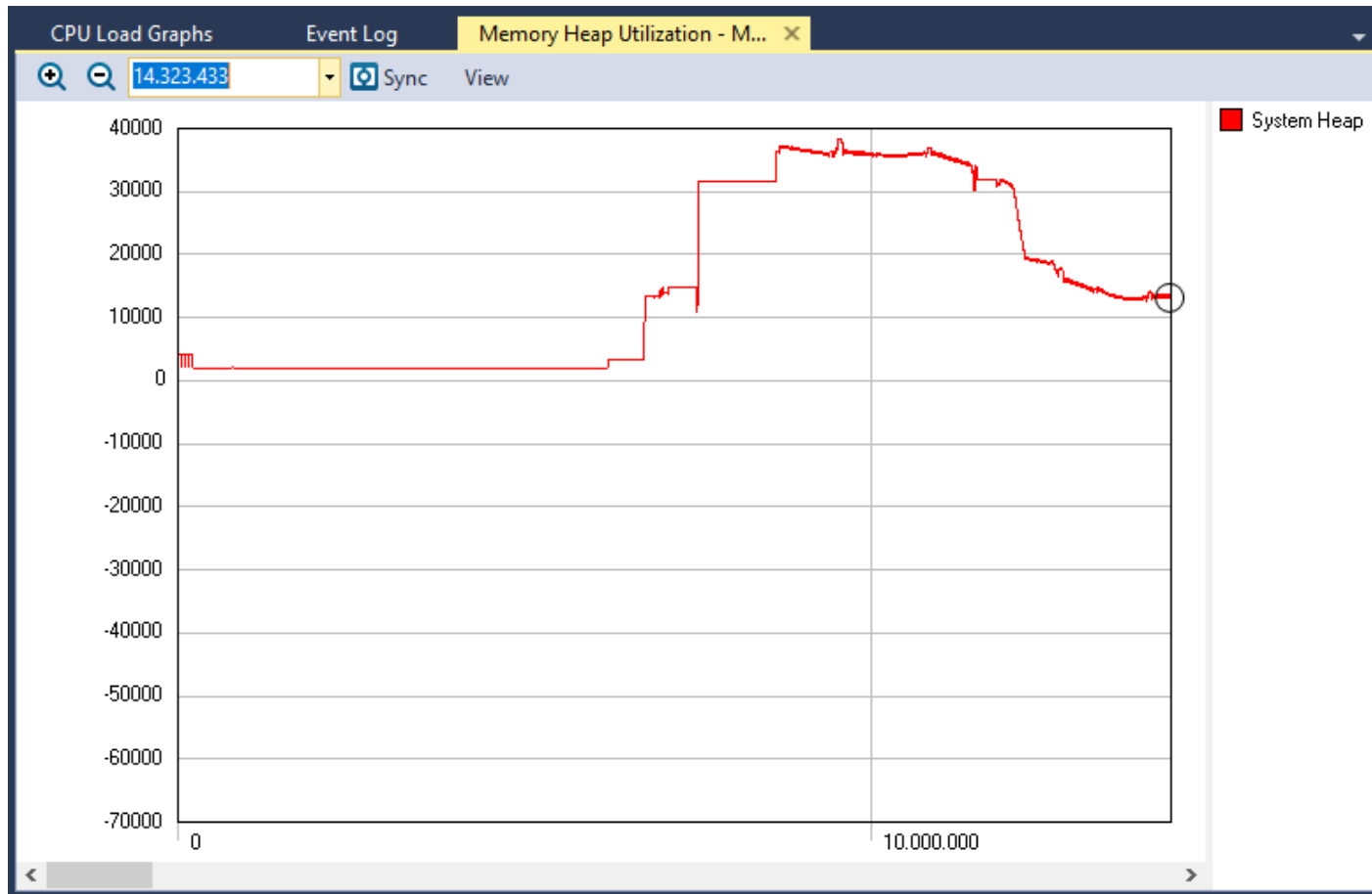
| Timestamp | Event |
|----------------------|---|
| 32.824.646 (s.ms.µs) | xMessageBufferReceive(0x100011B8) blocks |
| 32.824.658 (s.ms.µs) | xTaskNotifyWait(Echoing, 4294967295) blocks |

Showing 2 event(s)

CPU Utilization



Hunting for Memory Leaks

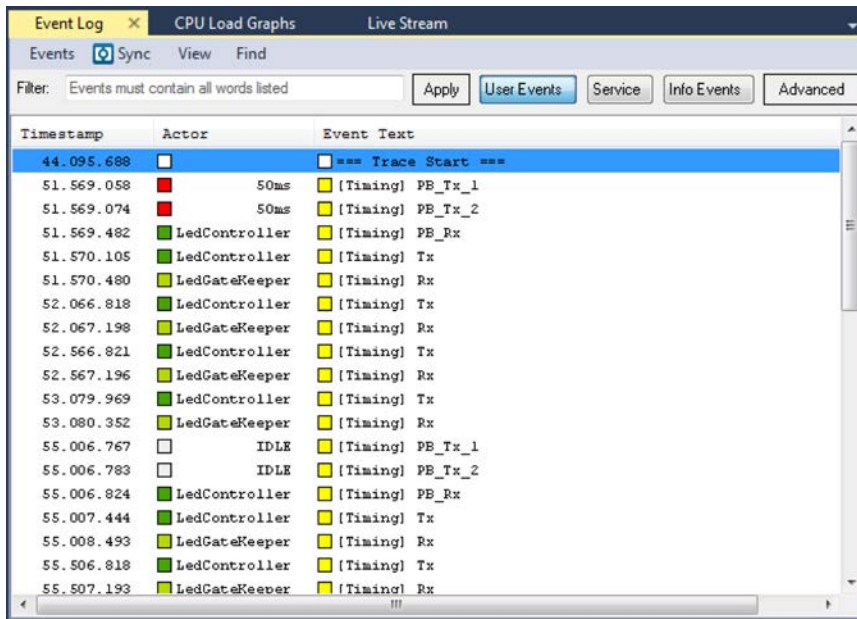


Creating User Channels

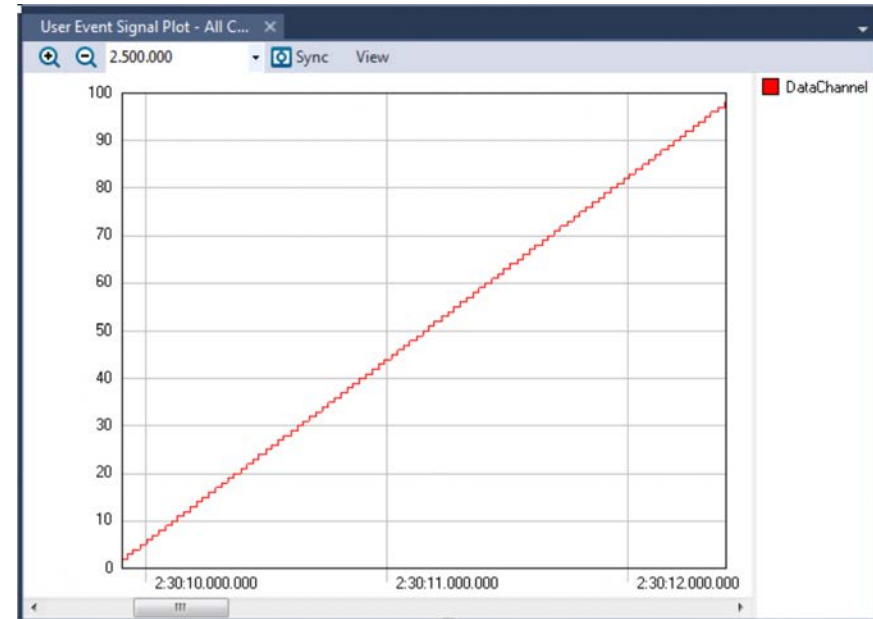
```
traceString MyChannel = xTraceRegisterString("DataChannel");
```

```
vTracePrint(MyChannel, "Button Pressed!");
```

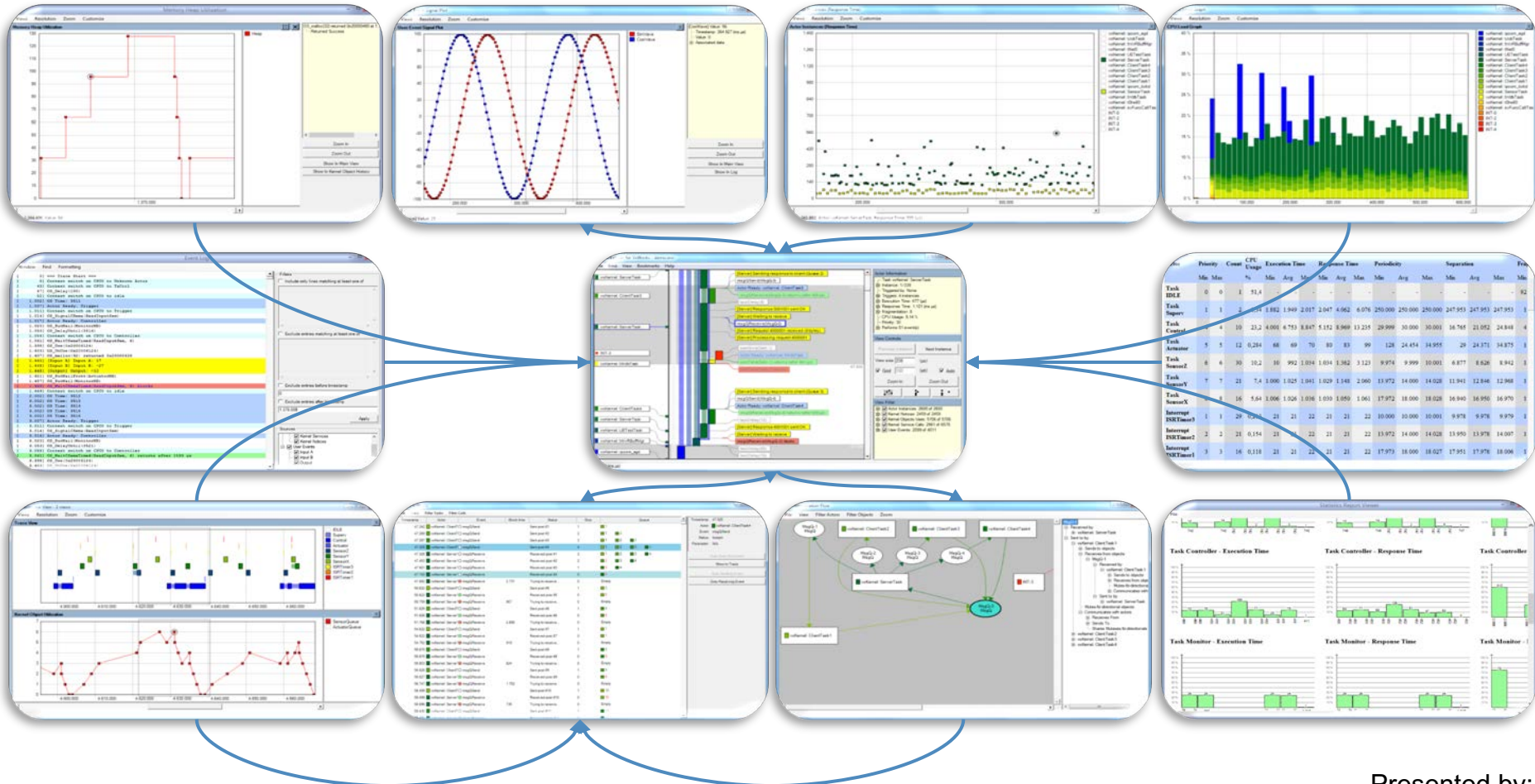
```
vTracePrintF(MyChannel, "Sensor Data = %d", SensorData);
```



| Timestamp | Actor | Event Text |
|------------|---------------|---------------------|
| 44.095.688 | | === Trace Start === |
| 51.569.058 | 50ms | [Timing] PB_Tx_1 |
| 51.569.074 | 50ms | [Timing] PB_Tx_2 |
| 51.569.482 | LedController | [Timing] PB_Rx |
| 51.570.105 | LedController | [Timing] Tx |
| 51.570.480 | LedGateKeeper | [Timing] Rx |
| 52.066.818 | LedController | [Timing] Tx |
| 52.067.198 | LedGateKeeper | [Timing] Rx |
| 52.566.821 | LedController | [Timing] Tx |
| 52.567.196 | LedGateKeeper | [Timing] Rx |
| 53.079.969 | LedController | [Timing] Tx |
| 53.080.352 | LedGateKeeper | [Timing] Rx |
| 55.006.767 | IDLE | [Timing] PB_Tx_1 |
| 55.006.783 | IDLE | [Timing] PB_Tx_2 |
| 55.006.824 | LedController | [Timing] PB_Rx |
| 55.007.444 | LedController | [Timing] Tx |
| 55.008.493 | LedGateKeeper | [Timing] Rx |
| 55.506.818 | LedController | [Timing] Tx |
| 55.507.193 | LedGateKeeper | [Timing] Rx |



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