# Jump Starting Code Development to Minimize Defects

#### Class 4: Mastering Application Tracing

#### December 13, 2018 Jacob Beningo



Presented by:



# **Course Overview**

#### **Topics:**

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







#### **Session Overview**

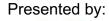
- Tracing
- Finding RTOS Application Bugs
- Analyzing Application Events
- Communication Flow
- Memory Leaks
- User Events
- State Machines





#### Analyze Trace Results

Event Log	Trace View - Vertical	× Context Switch Intensity	<b>*</b>
Q Q 100.000	<ul> <li>Sync</li> </ul>	View	
Tmr Svc	Tz <b>ana</b> Ma	Logging Ir Svo	0
	T z	ging	10.000.000
TzCtrl			20.000.000
MQTTEcho		Zoom in to show 298500 events	30.000.000
			40.000.000
			50.000.000
			1:00.000.000
Echoing		Zoom in to show 298500 events	1:10.000.000
			1:20.000.000 🛢
	·		1:30.000.000
		Zoom in to show 298500 events	1:40.000.000
			1:50.000.000
			2:00.000.000
	MQTTE: Echoing		2:10.000.000
	ritecho	Zoom in to show 298500 events	2:20.000.000
	R.		2:30.000.000
			2:40.000.000 🔻



CONTINUING EDUCATION

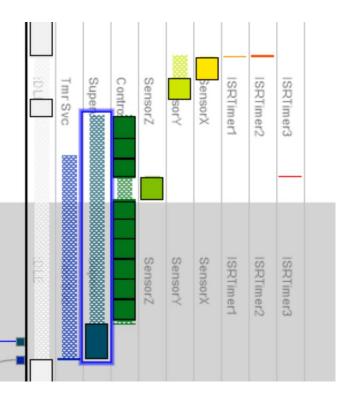


#### Analyze Trace Results

Actor	Count	CPU Usage	]	<b>Execution Time</b>		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

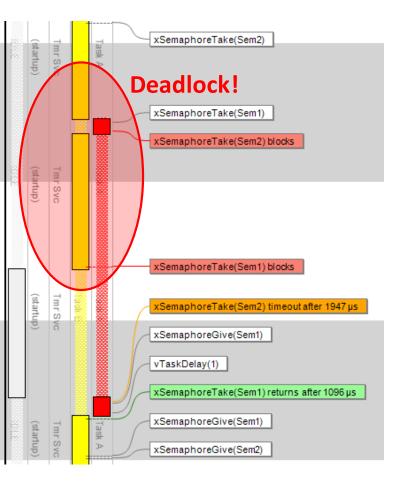
#### Solutions

- 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**

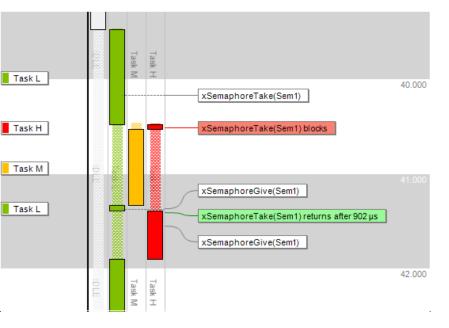
- Tasks that require two or more resources should acquire them in the same order
- Use timeouts to release the first resource if the second can't be acquired







# **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.

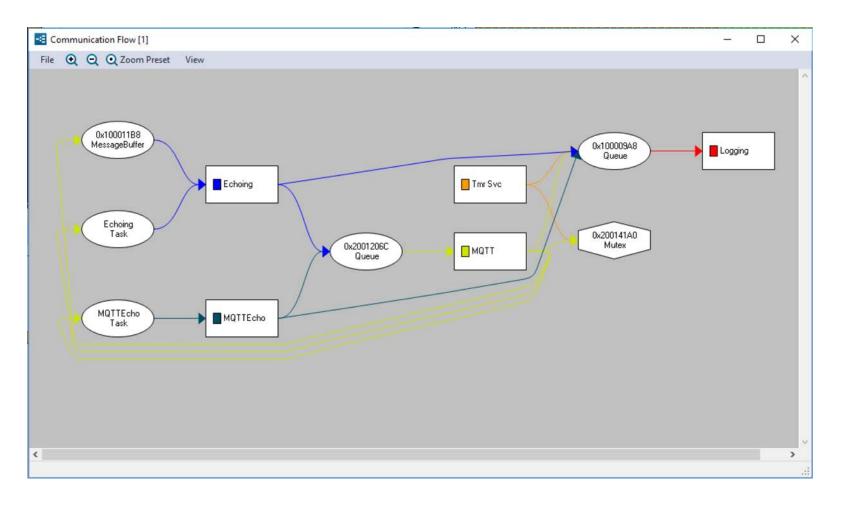
9

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

Event	Load Graphs ts Sync Export tamp 6041 6341	View Find Formattin malloc Event Text malloc(152) returned (	Apply	ervice Info Ever	nts Advanced
Times [ [ [	604)	malloc(152) returned (			
1 1 1 1			and the second second second second second second		
1 1 1	634]		0x100009A8		-
1 1		malloc(1808) returned	0x10000A40		
1	650]	malloc(104) returned (	0x10001150		
	1.229]	malloc(112) returned (	0x100011B8		
I	5.251]	malloc(112) returned (	0x100011B8		
[	8.998]	malloc(112) returned (	0x100011B8		
[	12.925]	malloc(2056) returned	0x100011B8		
[	59.236]	malloc(2056) returned	0x100011B8		
[	105.360]	malloc(112) returned (	0x100011B8		
1	109.372]	malloc(2056) returned	0x100011B8		
t	155.622]	malloc(2056) returned			
t	201.751]	malloc(112) returned (			
t	787.288]	malloc(112) returned (			
	206.349]	malloc(112) returned (			
	210.742]	malloc(112) returned (			
S. 3.	215.242]	malloc(112) returned (			
	219.335]	malloc(72) returned 0:			
	219.366]	malloc(1088) returned			
[ 6. 1 6	219.382]	malloc(104) returned ( melloc(112) returned (			



## **Analyze Communication Flow**





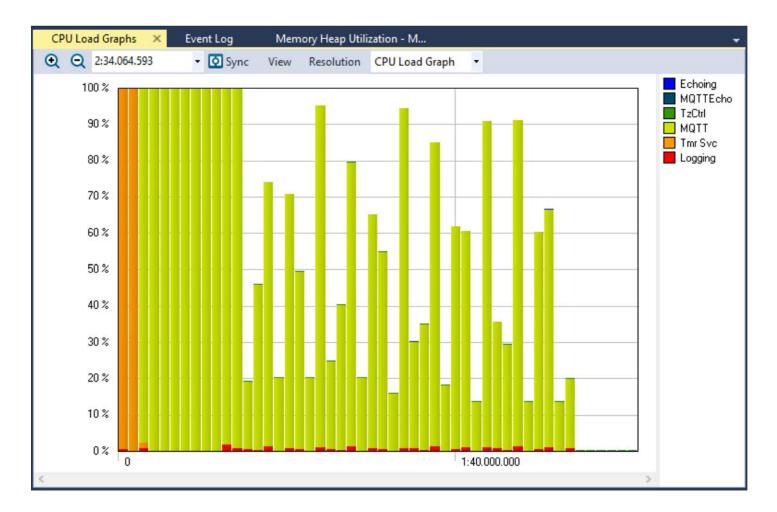


#### **Analyze Communication Flow**

ctor Overview -	Echoing				
Echoing	•				
Start Time	End Time	Execution Time	Response Time	Fragmentation	Instance Details
32.819.357	32.824.670	43	5.313	1	·····································
35.289.271	37.150.276	539	1.861.005	7	Triggered by: MQTTEcho
43.560.272	45.421.277	540	1.861.005	7	Triggers: None
					ier Execution Time: 43 (μs) ier Response Time: 5.313 (ms.μs)
51.837.272	53.701.277	540	1.864.005	7	
1:00.111.271	1:01.972.277	539	1.861.005	7	CPU Usage: 0.00404%
1:08.382.272	1:10.243.277	539	1.861.005	7	(A)
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	E Performed Events
1:33.200.271	1:35.061.475	491	1.861.204	5	Filter events by text
1:41.479.272	1:43.341.711	492	1.862.439	5	Timestamp Event
1:49.756.272	1:51.621.711	492	1.865.440	5	32.824.646 (s.ms.µs) xMessageBufferReceive(0x100011B8) blocks
1:58.036.272	1:59.898.885	493	1.862.613	5	32.824.658 (s.ms.µs) xTaskNotifyWait(Echoing, 4294967295) blocks
2:06.313.272	2:08.175.885	493	1.862.613	5	
					_ Showing 2 event(s)



#### **CPU Utilization**







# Hunting for Memory Leaks



Presented by:



## **Creating User Channels**

traceString MyChannel = xTraceRegisterString("DataChannel");

vTracePrint(MyChannel, "Button Pressed!");

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

ind ds listed Apply User Events Service Info Events Advanced Event Text Soms (Timing) PB_Tx_1 Soms (Timing) PB_Tx_2 coller (Timing) PB_Rx	Q         2.500.000         - ☑ Sync         View           100         -         -         -         -         DataChar           90         - <td< th=""></td<>
Event Text  Event Text  SOms (Timing) PE_Tx_1  Soms (Timing) PE_Tx_2	100 90 DataChar
Image         Start         Image         Image <th< th=""><th></th></th<>	
50ms [[Timing] PB_Tx_1 50ms [[Timing] PB_Tx_2	
50ms [[Timing] PB_Tx_2	80
roller 🗧 [Timing] PB_Rx 🗧	
	70
roller 🗌 (Timing) Tx	
Keeper 🗌 [Timing] Rx	60
roller 🗌 [Timing] Tx	
Keeper 🗌 [Timing] Rx	50
roller 🗌 (Timing) Tx	
Keeper 🗌 [Timing] Rx	40
roller 🗌 [Timing] Tx	
Keeper 🗌 [Timing] Rx	30
IDLE [Timing] PB_Tx_1	
IDLE [Timing] PB_Tx_2	20
roller 🗌 [Timing] PB_Rx	
roller 🗌 [Timing] Tx	10
Keeper 🗌 [Timing] Rx	
roller 🗌 (Timing) Tx	0 <sup>27</sup>
Keeper 🗖 [Timing] Rx 👘	2:30:10.000.000 2:30:11.000.000 2:30:12.000.000
	coller       [Timing] Tx         Geeper       [Timing] PE_Tx_1         IDLE       [Timing] PE_Tx_2         coller       [Timing] PE_Rx         coller       [Timing] Tx         Geeper       [Timing] Rx         coller       [Timing] Tx



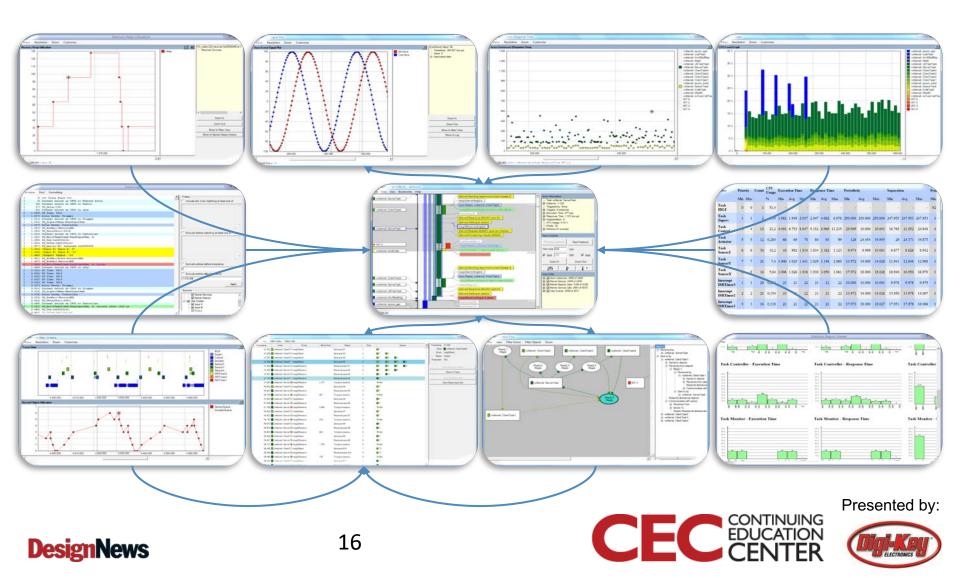


#### **User State Machines**

Q 12.658.669	- 💽 Sync	Vi	PW						Event Log Events 💽 S		
				for Stal	teate =	State :	ale:	🗘 Event Field 0 🔺	ALCONTRACTOR AND	ust contain all words	
C     Tmr Svc     Tmr description       Cikleper     LedGateKeeper                   Indian     LedController					4	ů.		1.000.000	Timestamp	Actor	Event Text
							11	2.000.000	242		=== Trace Start ===
			-					Zoom in to show 22269 events	28.814	📒 led_blink	[LedStackMonitor] StackSize = 96
	<b>T</b>	M	M	M	Mo	s g	3	3.000.000	36.118	📕 50ms (2)	[DataChannel] Sensor Data = 0
dCol II SV	÷.	otor	otor	otor	otor	MotorState			62.164	<b>50ms</b> (2)	[MotorState] MotorState = 0
r Svc dGateKeepe dController	호	torState	State		orState	State	2	4.000.000	62.170	<b>50ms</b> (2)	[BrakeState] BrakeState = 0
		11	11 11		TE 2	11 11			62.176	50ms (2)	[DataChannel] Sensor Data = 1
		0	-> K	1	4	u c	20	Zoom in to show 22269 events 000,000	88.170	<b>50ms</b> (2)	[MotorState] MotorState = 0
									88.176	<b>50ms</b> (2)	[BrakeState] BrakeState = 0
								6.000.000	88.181	<b>50ms</b> (2)	[DataChannel] Sensor Data = 2
									114.164	<b>50ms</b> (2)	[MotorState] MotorState = 0
Fed fed		Moto	MotorState :	MotorState	Moto			7.000.000	114.170	<b>50ms</b> (2)	[BrakeState] BrakeState = 0
\$att		or St	or St	2 52	ų.	22 G	2	Zoom in to show 22269 events 8.000.000	114.176	<b>50ms</b> (2)	[DataChannel] Sensor Data = 3
Svc þat¢Ke		rState	ate	ate		ate		20.000.000	126.920	📒 led_blink	[LedStackMonitor] StackSize = 52
		1.11.11.11.11				ແກ C		8 000 000	144.103	50ms (2)	[MotorState] MotorState = 0
									144.109	<b>50ms</b> (2)	[BrakeState] BrakeState = 0
		-					110	Zoom in to show 22269 events	144.115	<b>50ms</b> (2)	[DataChannel] Sensor Data = 4
								200m in to show 22269 events	169.505	<b>50ms</b> (2)	[MotorState] MotorState = 0
Sve Timr Sv Satekcepper     Leadsa bantroller     Leadsa		Mo	M	Mo	M	3		11.000.000	169.511	<b>50ms</b> (2)	[BrakeState] BrakeState = 0
		otor	to	otor	otor	otor	ake		169.517	<b>50ms</b> (2)	— [DataChannel] Sensor Data = 5



#### **Examine Different Views**



### **Additional Resources**

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



#### From <u>www.beningo.com</u> under

Blog > CEC – Jump Starting Code Development to Minimize
 Defects

**DesignNews** 



