

Agenda and Intro

- **A brief intro to basic concepts**
 - Why things work the way they do
 - Assume you are familiar with RTOSs and MQX
- **Real work: hands-on labs**
 - Create a project, add the MQX Lite component
 - Create tasks, watch the flashing lights

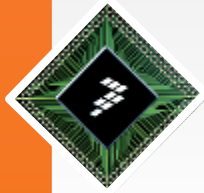




Abstract

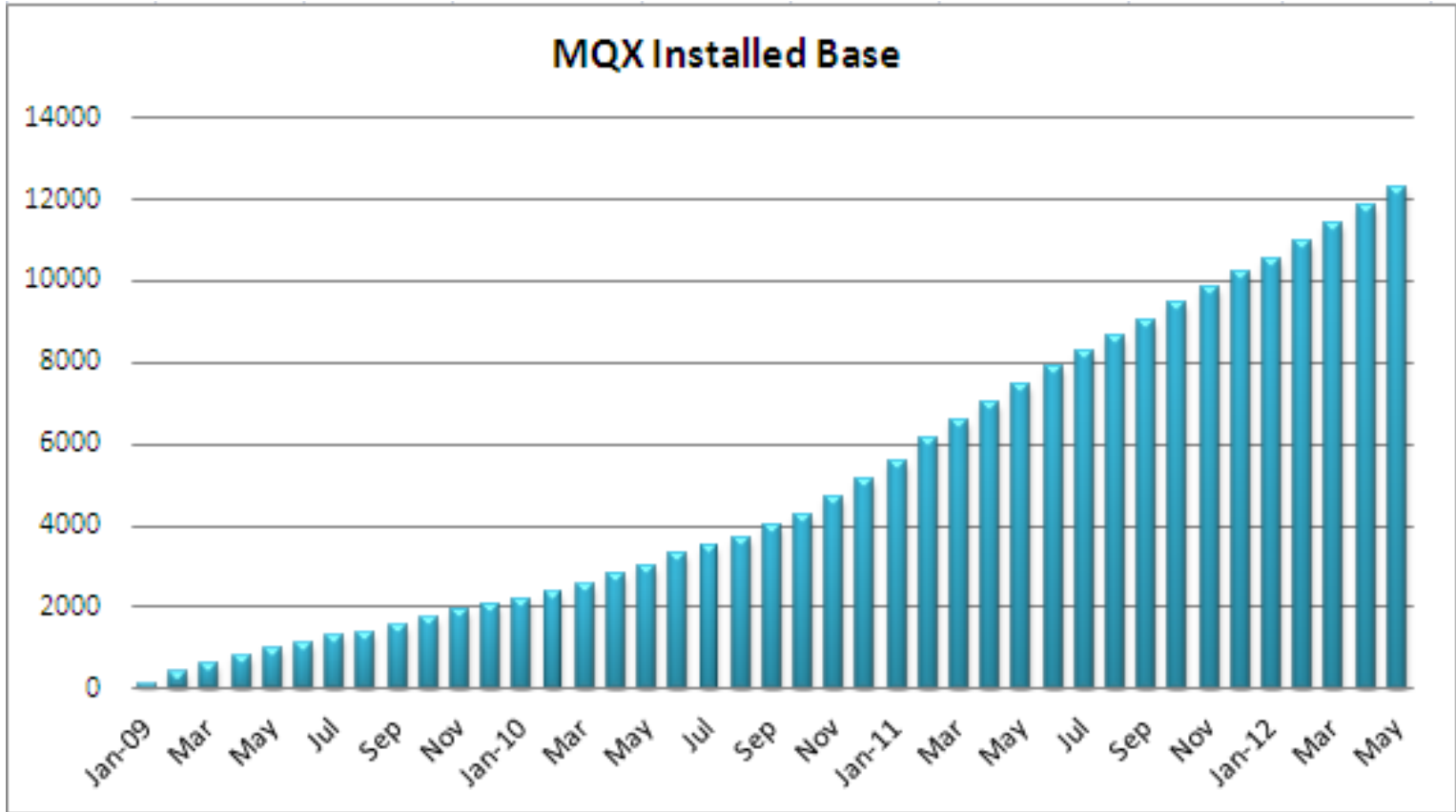
- You will use Processor Expert (PEX) to configure MQX Lite (a PEX component) and initialize multiple tasks on the L series processor. Light the LED and print message from UART.

- We got most of it...



General Impact of MQX RTOS

- Adding ~450 new users every month



MQX Lite – Overview

- **Very light MQX kernel for resource-limited MCUs**
 - Targeted at the Kinetis L family initially
 - Packaged as a Processor Expert component
- **I/O capability provided by Processor Expert**
 - USB via FSL bare-metal stack, also a Processor Expert component
 - No file access
- **Programming model allows upward code migration**
 - Code built with MQX Lite should move to full MQX RTOS easily

MQX Lite – Main Features

- **Scheduler**
 - Priority pre-emptive schedule
 - Support for lightweight semaphore, and mutex (with polling)
- **Task Management will not support dynamic task creation**
 - All task resources allocated at compile time
- **Lightweight events and messaging only**
- **Dynamic memory management not allowed**
- **Lightweight timer included (one shot, and periodic notification)**

How Small Is MQX Lite?

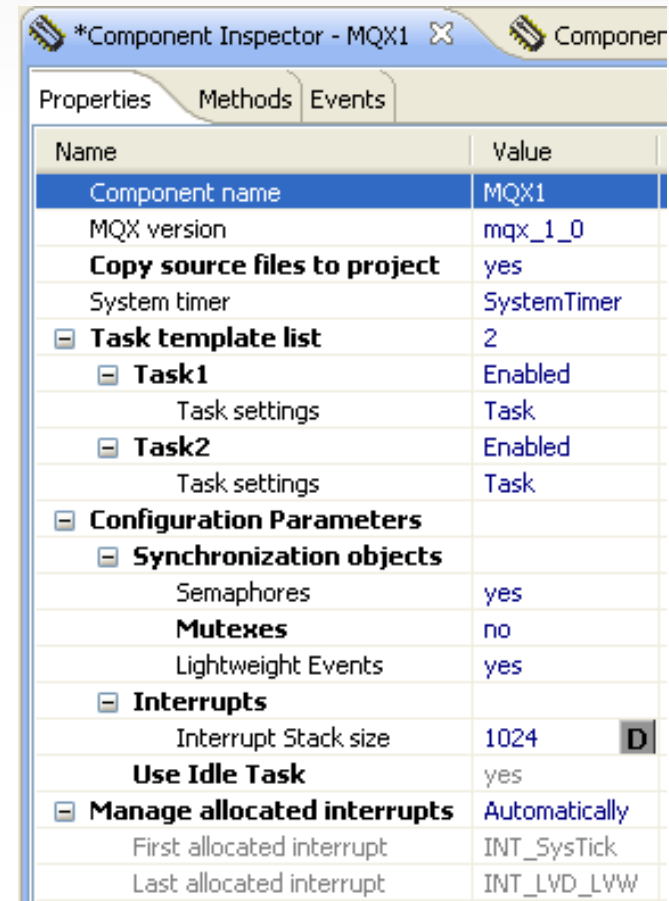
- **Minimal App – Hello Task, Idle task, interrupt stack**
 - Code = 10.4K
 - Data = 3.7K (including 1.5K for stacks)

- **Typical App – 7 tasks + idle, lightweight events, queues**
 - Code = 27K
 - Data = 10K (5K for stacks)

- **Your mileage will vary**

MQX Lite and Processor Expert Integration

- **MQX Lite delivered as an RTOS adapter**
 - Interrupt mechanism in MQX is unchanged
 - Processor Expert LDDs work with the RTOS
- **The entire I/O from standard MQX removed**
 - I/O provided by LDD components
- **Set up and configure tasks in Component Inspector**
- **Easy to add MQX Lite to existing app**
 - Just drop in the MQX Lite component



The screenshot shows the Component Inspector window for a component named MQX1. The window has tabs for Properties, Methods, and Events. The Properties tab is active, displaying a list of configuration parameters and their values.

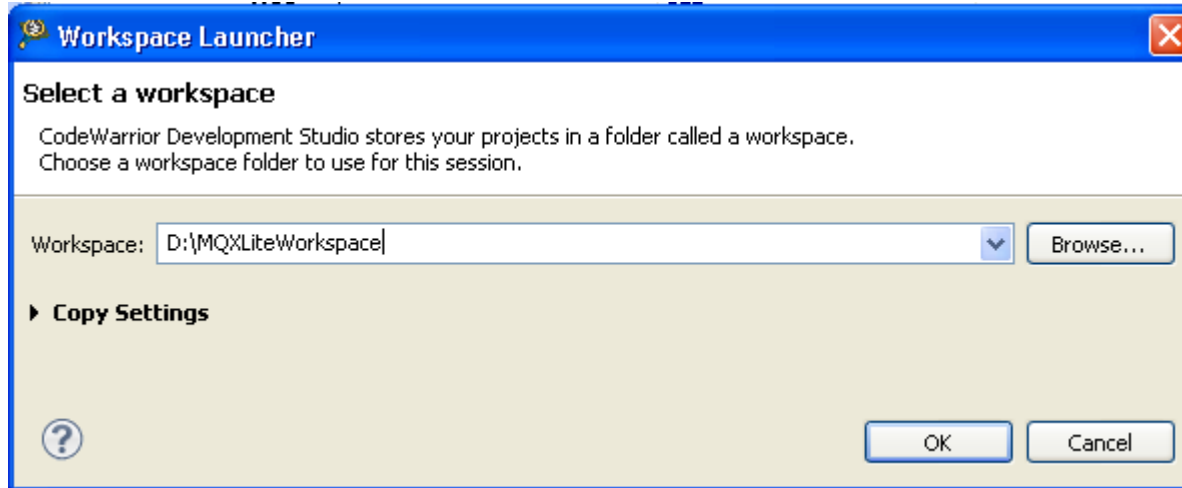
Name	Value
Component name	MQX1
MQX version	mqx_1_0
Copy source files to project	yes
System timer	SystemTimer
Task template list	2
Task1	Enabled
Task settings	Task
Task2	Enabled
Task settings	Task
Configuration Parameters	
Synchronization objects	
Semaphores	yes
Mutexes	no
Lightweight Events	yes
Interrupts	
Interrupt Stack size	1024 D
Use Idle Task	yes
Manage allocated interrupts	Automatically
First allocated interrupt	INT_SysTick
Last allocated interrupt	INT_LVD_LVW

Enough Talk, Time to Work

- **Create a workspace**
- **Import a project**
- **Add and configure the MQX Lite component**
 - **Define a task and write the code**
- **Add and walk through PEx components**
 - **Using templates (this is way cool if you aren't familiar)**
- **Build it and run it – watch the blinking lights**

Set CodeWarrior to a new workspace

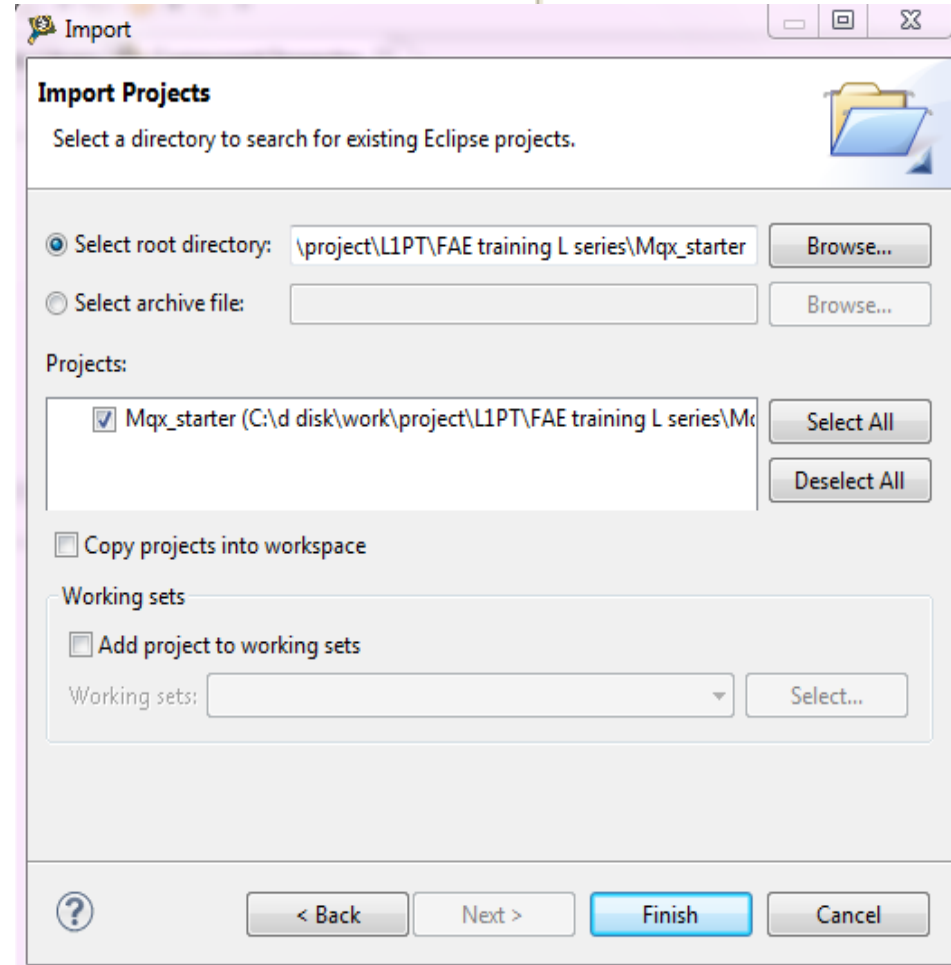
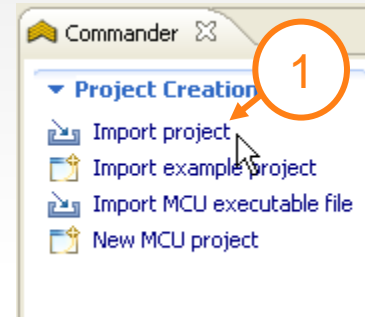
- This makes sure we don't have any confusion
- File → Switch Workspace → Other



- Type in d:\MQXLiteWorkspace
- For the sake of others, turn off “Use this as the default...”
- Then you can hide the Welcome Screen that appears.

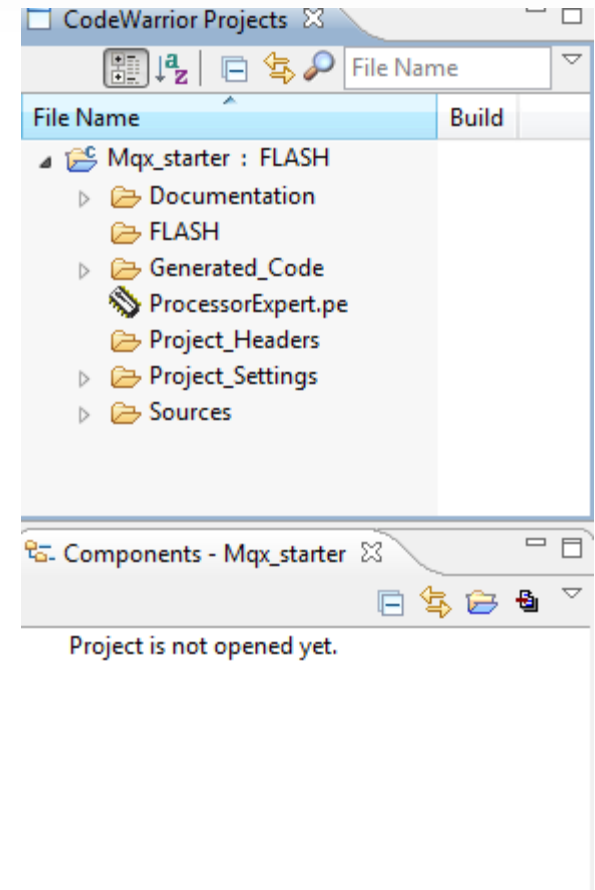
Import the starter project

- Click Import Project in Commander view
 - File→Import→General→Existing Projects Into Workspace
- Select root directory
- C:\d disk\work\project\L1PT
\FAE training L series\Mqx_starter



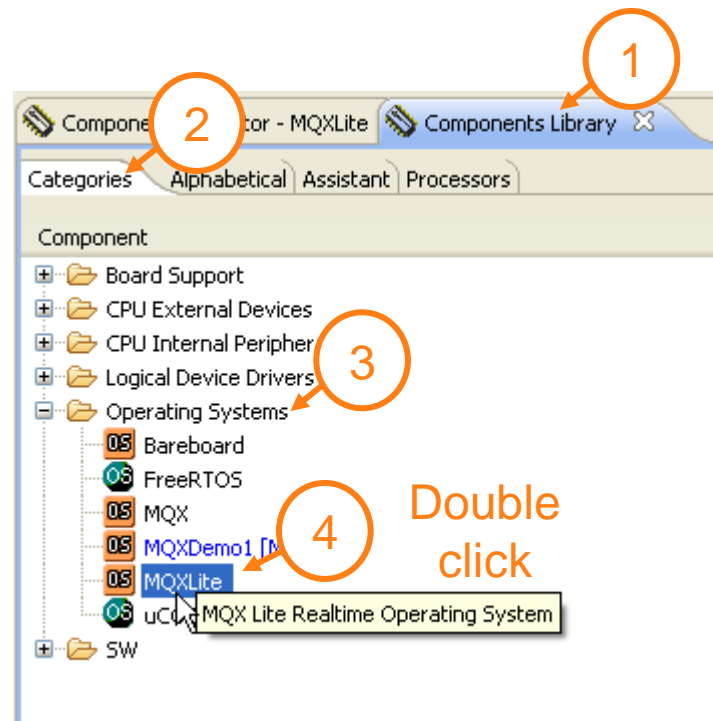
Open Processor Expert project

- This takes a few second on first launch
- Double click the .pe file
- The Components view opens
 - New UI in 10.3
 - No longer called “Project Panel”
 - No longer shares the same space as the CodeWarrior Projects view



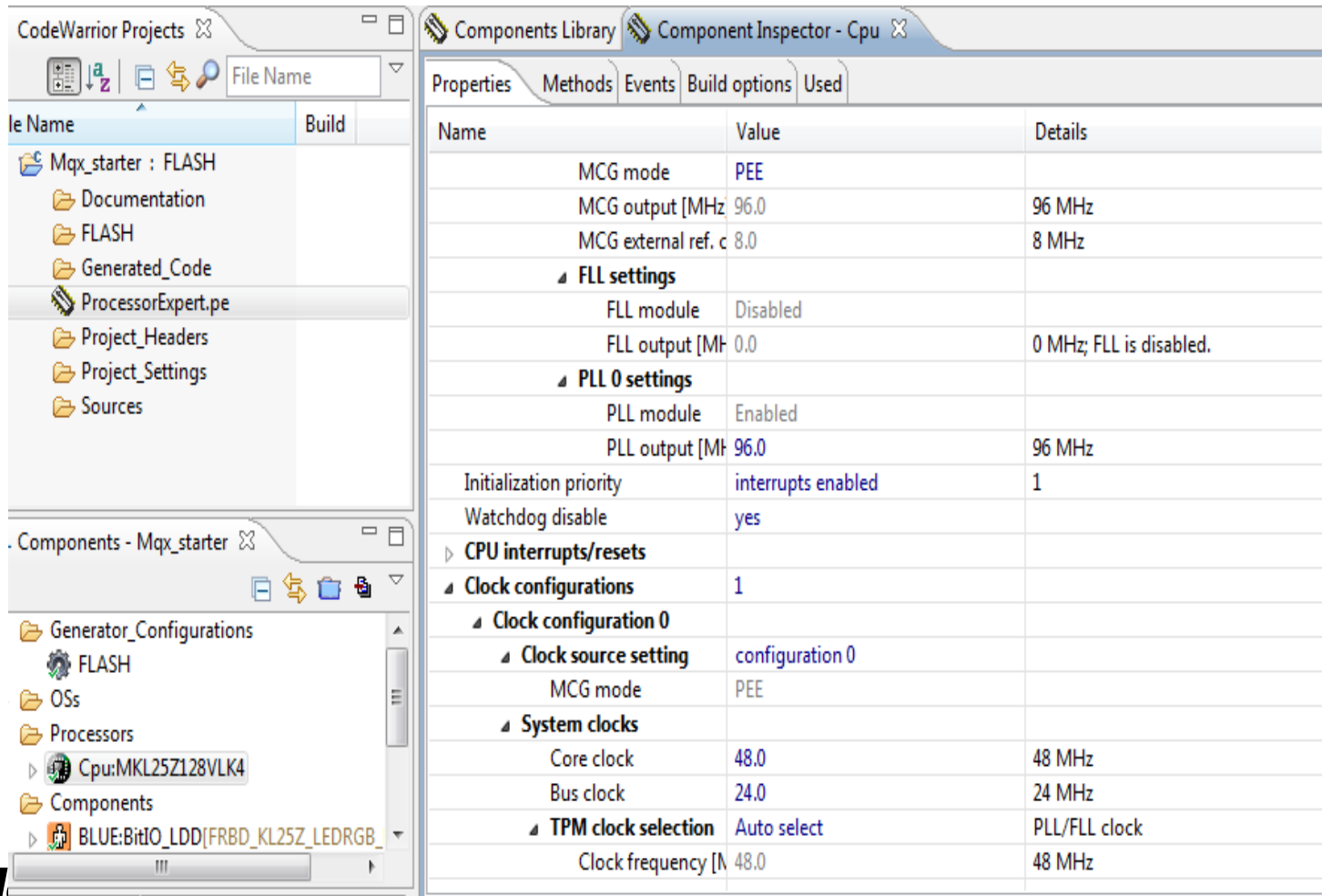
Add the MQX Lite Component

- Go to Library, Categories, Operating Systems
- Double-click MQX Lite



Examine CPU Component

- Provided and configured in starter project
- Select CPU, go to component inspector, look at System Clock
- 48 MHz



The screenshot shows the CodeWarrior IDE interface. The 'Component Inspector - Cpu' window is open, displaying the 'Properties' tab. The table below lists the properties of the CPU component.

Name	Value	Details
MCG mode	PEE	
MCG output [MHz]	96.0	96 MHz
MCG external ref. c	8.0	8 MHz
▲ FLL settings		
FLL module	Disabled	
FLL output [MHz]	0.0	0 MHz; FLL is disabled.
▲ PLL 0 settings		
PLL module	Enabled	
PLL output [MHz]	96.0	96 MHz
Initialization priority	interrupts enabled	1
Watchdog disable	yes	
▶ CPU interrupts/resets		
▲ Clock configurations		
1		
▲ Clock configuration 0		
▲ Clock source setting		
configuration 0		
MCG mode		
PEE		
▲ System clocks		
Core clock	48.0	48 MHz
Bus clock	24.0	24 MHz
▲ TPM clock selection		
Auto select		
PLL/FLL clock		
Clock frequency [MHz]	48.0	48 MHz

Configure System Timer

- The MQX Lite Component depends on a timer – which must synch with the system

- frequency = 48 MHz

The screenshot displays the CodeWarrior IDE interface. On the left, the 'Components Library' pane shows a tree view with 'MQX1:MQXLite' selected, indicated by a circled '1'. Below it, the 'Components - Mqx_starter' pane shows a list of components including 'SystemTimer1:TimerUnit_LDD[MQX1:MQXLite]'. On the right, the 'Component Inspector - SystemTimer1' pane shows the 'Properties' tab. The 'Counter frequency' property is highlighted in blue and circled with a '2', with an arrow pointing to its value '48 MHz'. Other properties include 'Module name' (SysTick), 'Counter' (SYST_CVR), 'Counter direction' (Down), 'Counter width' (24 bits), 'Value type' (Optimal, uint32_t), 'Input clock source' (Internal), 'Counter restart' (On-match), 'Period device' (SYST_RVR), 'Period' (5 ms), 'Interrupt' (Enabled, medium priority, 2), 'Channel list' (0), 'Initialization' (Enabled in init. code: no, Auto initialization: no), and 'Event mask'.

Name	Value	Details
Module name	SysTick	SysTick
Counter	SYST_CVR	SYST_CVR
Counter direction	Down	
Counter width	24 bits	
Value type	Optimal	uint32_t
Input clock source	Internal	
Counter frequency	48 MHz	48 MHz
Counter restart	On-match	
Period device	SYST_RVR	SYST_RVR
Period	5 ms	5 ms
Interrupt	Enabled	
Interrupt priority	medium priority	2
Channel list	0	
Initialization		
Enabled in init. code	no	
Auto initialization	no	
Event mask		

Configure the Component

- We will have two tasks, so increase by one

- Task Template List = 3

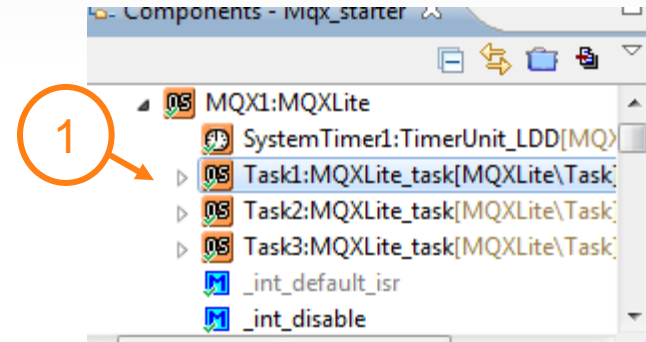
- Other values are default

The screenshot shows the CodeWarrior IDE interface. On the left, the 'Components Library' window displays a tree view of components. The component 'MQX1:MQXLite' is selected and highlighted with an orange circle containing the number '1'. Below it, three task components are listed: 'Task1:MQXLite_task', 'Task2:MQXLite_task', and 'Task3:MQXLite_task'. On the right, the 'Component Inspector' window is open, showing the configuration for the selected component. The 'Task template list' property is set to '3' and is circled with an orange circle containing the number '2'. Below this, the 'Task settings' row is highlighted in yellow. The 'Configuration Parameters' section shows various settings like 'Synchronization objects', 'Mutexes', 'Interrupts', and 'Use Idle Task'.

Name	Value	Details
Component name	MQX1	
MQX version	mqx_1_0	
Copy source files to project	yes	
System timer	SystemTimer	
Task template list	3	
Task1	Enabled	
Task settings	Task	
Task2	Enabled	
Task settings	Task	
Task3	Enabled	
Task settings	Task	
Configuration Parameters		
Synchronization objects		
Semaphores	yes	
Mutexes	no	
Lightweight Events	no	
Interrupts		
Interrupt Stack size	1024	D
Use Idle Task	yes	
Manage allocated interrupts	Automatically	

Configure the First Task

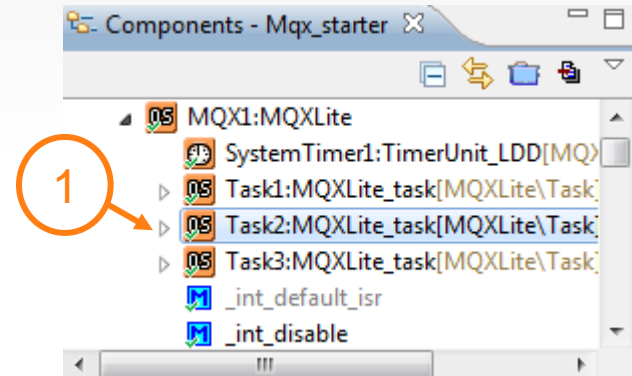
- **Select the task**
- **Examine the properties**
- **Actually, default values for all properties**
- **Task 1**
 - **Name = Task1 (default)**
 - **Case matters!**
Code depends on this
 - **Entry point function**
 - **Stack size = 1024**



Name	Value	Details
Name	Task1	
Entry point function	Task1_task	
Stack size	1024	D
Priority	9	D
Creation parameter	0	
+ Attributes		

Configure the Second Task

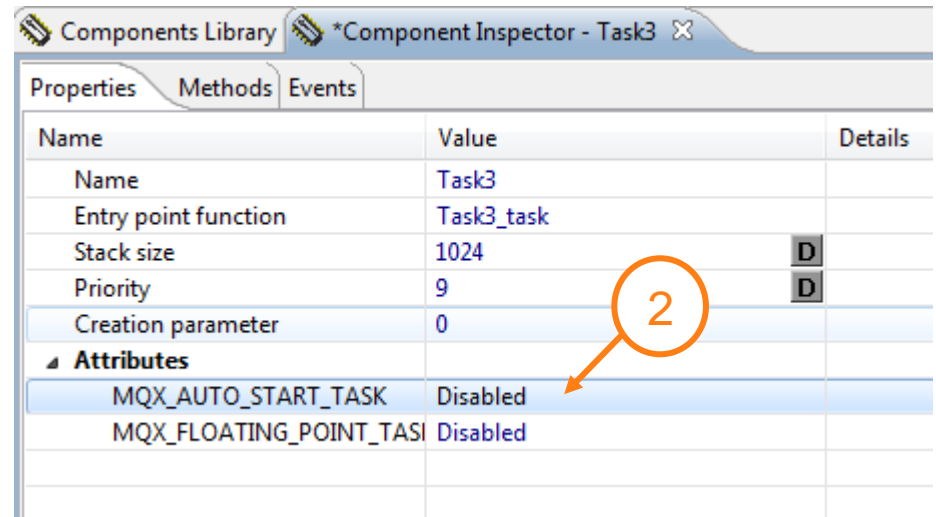
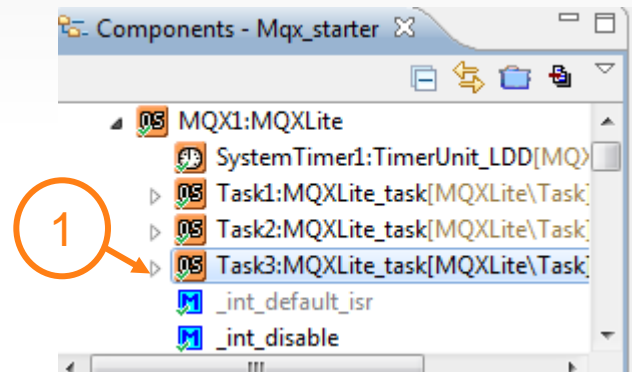
- **Task2**
 - Name = Task2 (Default)
 - Case matters! Code depends on this
 - Priority = 9
 - AUTO_START_TASK = Enable
 - we will instantiate and start the task in our code



Name	Value	Details
Name	Task2	
Entry point function	Task2_task	
Stack size	1024	D
Priority	9	D
Creation parameter	0	
▲ Attributes		
MQX_AUTO_START_TASK	Enabled	
MQX_FLOATING_POINT_TASI	Disabled	

Configure the Second Task

- **Task3**
 - **Name = Task3 (Default)**
 - **Case matters! Code depends on this**
 - **Priority = 9**
 - **AUTO_START_TASK = Disable**
 - **we will instantiate and start the task in our code**



Build the Code

- Just click Build in the Commander view OR...
- Select the project
- Project → Build Project
- Executable file
- Project
- Should be no errors



Code Walk Through: Initialize the OS

- Open ProcessorExpert.c
- RTOS initialization – auto generated code

```
118 | /** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS c
119 | #ifdef PEX_RTOS_START
120 |     PEX_RTOS_START();           /* Startup of the selected RTOS.
121 | #endif
122 | /** End of RTOS startup code. */
```

- That Macro = a call to `_mqxlite()` – which sets up the OS

```
.h MQX1.h X
76 /* MQX Lite start function */
77 #define PEX_RTOS_START() ( _mqxlite() )
78
```

This code creates the tasks – mqxlite.c

```
MQX1.h  mqxlite.c
266 _mqx_uint _mqxlite(void)
267 { /* Body */
268     KERNEL_DATA_STRUCT_PTR      kernel_data;
269     MQXLITE_TASK_TEMPLATE_STRUCT_PTR  template_ptr;
270     TD_STRUCT_PTR                td_ptr;
271
272     _GET_KERNEL_DATA(kernel_data);
273
274     /* Start Tick timer */
275     MQXLITE_RTOS_ADAPTER_SYSTEM_TIMER_START(NULL);
276
277     /* Create the idle task */
278 #if MQX_USE_IDLE_TASK
279     td_ptr = _task_init_internal((MQXLITE_TASK_TEMPLATE_STRUCT_PTR) &kernel_data->IDLE_TASK_TEMPLATE,
280                                kernel_data->ACTIVE_PTR->TASK_ID,
281                                (uint_32)0,
282                                FALSE,
283                                (pointer)kernel_data->IDLE_TASK_TEMPLATE.TASK_STACKADDR,
284                                (_mem_size)kernel_data->IDLE_TASK_TEMPLATE.TASK_STACKSIZE);
285 #if MQX_CHECK_ERRORS
286     if (td_ptr == NULL) {
287         _mqx_exit(MQX_OUT_OF_MEMORY);
288     } /* Endif */
289 #endif
290     _task_ready_internal(td_ptr);
291 #endif
292
293     /* Check here for auto-create tasks, and create them here */
294     template_ptr = kernel_data->INIT.TASK_TEMPLATE_LIST;
295     while (template_ptr->TASK_TEMPLATE_INDEX) {
296         if (template_ptr->TASK_ATTRIBUTES & MQX_AUTO_START_TASK) {
297             td_ptr = _task_init_internal(template_ptr,
298                                         kernel_data->ACTIVE_PTR->TASK_ID,
299                                         template_ptr->CREATION_PARAMETER,
300                                         FALSE,
301                                         (pointer)template_ptr->TASK_STACKADDR,
302                                         (_mem_size)template_ptr->TASK_STACKSIZE);
```

Task templates generated from component

- Open task_template_list.c
- – in Generated_Code folder
- Here are our tasks
 - Based on the properties set in the component
 - One is an auto-start task as we specified

```
12 /* MQX task template list */
13 const TASK_TEMPLATE_STRUCT MQX_template_list[] =
14 {
15     /* Task: Task1 */
16     {
17         /* Task number           */ TASK1_TASK,
18         /* Entry point           */ (TASK_FPTR)Task1_task,
19         /* Stack size            */ TASK1_TASK_STACK_SIZE,
20         /* Task priority         */ 9U,
21         /* Task name             */ "task1",
22         /* Task attributes       */ (MQX_AUTO_START_TASK),
23         /* Task parameter       */ (uint32_t){0}
24     },
25     /* Task: Task2 */
26     {
27         /* Task number           */ TASK2_TASK,
28         /* Entry point           */ (TASK_FPTR)Task2_task,
29         /* Stack size            */ TASK2_TASK_STACK_SIZE,
30         /* Task priority         */ 9U,
31         /* Task name             */ "task2",
32         /* Task attributes       */ (MQX_AUTO_START_TASK),
33         /* Task parameter       */ (uint32_t){0}
34     },
35     /* Task: Task3 */
36     {
37         /* Task number           */ TASK3_TASK,
38         /* Entry point           */ (TASK_FPTR)Task3_task,
39         /* Stack size            */ TASK3_TASK_STACK_SIZE,
40         /* Task priority         */ 9U,
41         /* Task name             */ "task3",
42         /* Task attributes       */ (0),
43         /* Task parameter       */ (uint32_t){0}
44     }
45 }
```

Task1 Code

- In `mqx_tasks.c`
- Function header automatic
 - You will still need to create body of function obviously
- Loops endlessly flashing BLUE LED

```
38 ** Returns : Nothing
39 ** =====
40 */
41 void Task1_task(uint32_t task_init_data)
42 {
43     int counter = 0;
44     printf("task 1 start running!\n");
45     while(1) {
46         counter++;
47
48         /* Write your code here ... */
49         BLUE_ClrVal(0);
50         _time_delay_ticks(50);
51         BLUE_SetVal(0);
52         _time_delay_ticks(50);
53     }
54 }
55
```

Task2 Code

- Loops endlessly flashing GREE
- Setup task3

```
62 **      MQX task routine. The routine is generated into mqx_tasks.c
63 **      file.
64 **      Parameters :
65 **          NAME          - DESCRIPTION
66 **          task_init_data -
67 **      Returns       : Nothing
68 **      =====
69 */
70 void Task2_task(uint32_t task_init_data)
71 {
72     int counter = 0;
73     _task_id task_id;
74     printf("task 2 start running!\n");
75     task_id = _task_create_at(0, TASK3_TASK, 10, Task3_task_stack, TASK3_TASK_STACK_SIZE);
76     if ( task_id == MQX_NULL_TASK_ID )
77     {
78         printf("task3 create fail!\n");
79     }
80
81     while(1) {
82         counter++;
83
84         /* Write your code here ... */
85         GREE_SetVal(0);
86         _time_delay_ticks(50);
87
88         GREE_ClrVal(0);
89         _time_delay_ticks(50);
90     }
91 }
```

- The OS handles task priority and switching

Task2 Code

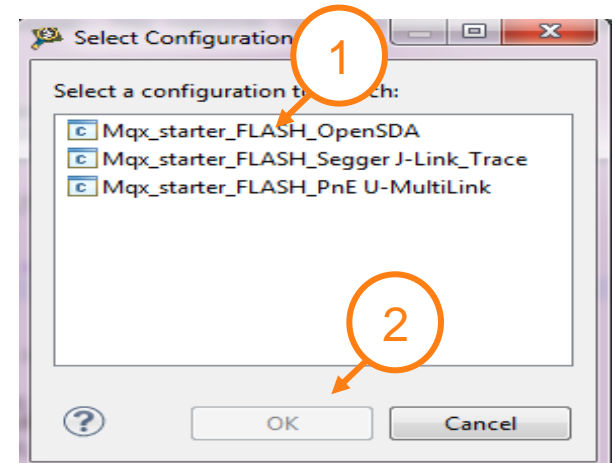
- Loops endlessly flashing RED

```
.98**      Description :
.99**          MQX task routine. The routine is gen
.00**          file.
.01**      Parameters :
.02**          NAME          - DESCRIPTION
.03**          task_init_data -
.04**      Returns       : Nothing
.05**      =====
.06*/
.07void Task3_task(uint32_t task_init_data)
.08{
.09    int counter = 0;
.10    printf("task 3 start running!\n");
.11    while(1) {
.12        counter++;
.13
.14        /* Write your code here ... */
.15        RED_ClrVal(0);
.16        _time_delay_ticks(50);
.17        RED_SetVal(0);
.18        _time_delay_ticks(50);
.19    }
.20}
.21
.22/* END mqx_tasks */
--
```

- The OS handles task priority and switching

Download

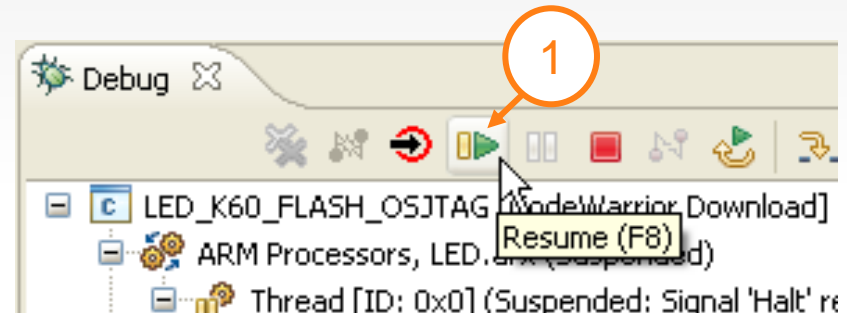
- Run→Debug As→CodeWarrior Download
 - Pick the P&E Tracelink configuration
- Debug perspective appears
- Code stops at first line of main()
- Set a breakpoint at line 132 of mxq_tasks.c



```
45 while(1) {  
46     counter++;  
47 }  
48 /* Write your code here ... */  
49 BLUE_ClrVal(0);  
50     _time_delay_ticks(50);  
51     BLUE_SetVal(0);  
52     _time_delay_ticks(50);
```

Run

- Click the Resume button
- Resume each time you stop
 - You'll see the count variable increase, and the lights on the board flash
- Remove the breakpoint to run the app without interference
 - Right click, Toggle breakpoint

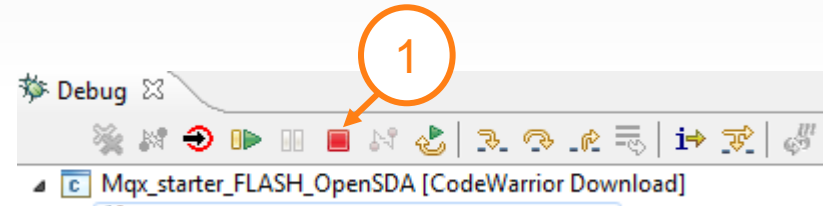


A screenshot of the IDE's Variables window. The window displays a table with two columns: 'Name' and 'Value'. The table contains two rows of data. The first row is '(x)= task_param' with a value of '1937006955'. The second row is '(x)= task2_count' with a value of '2'. The second row is highlighted in yellow.

Name	Value
(x)= task_param	1937006955
(x)= task2_count	2

All Done

- Click the Terminate button



- You have built an app using
 - the MQX Lite component and PEx
 - Adding a PEx component template
- You should understand some of the key differences between MQX RTOS and MQX Lite
- Now – if there's time, for more fun