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

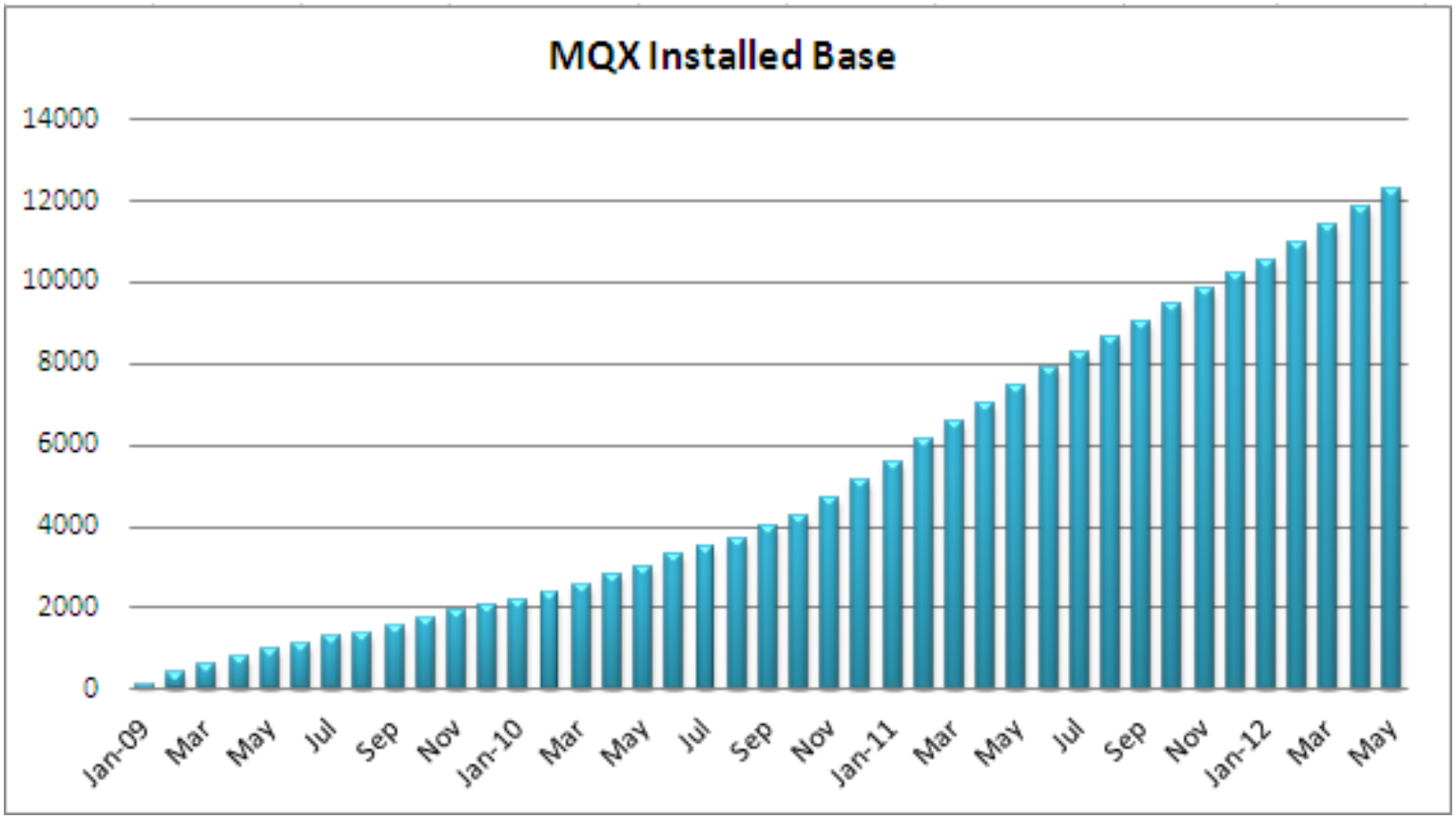
Abstract

- 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?

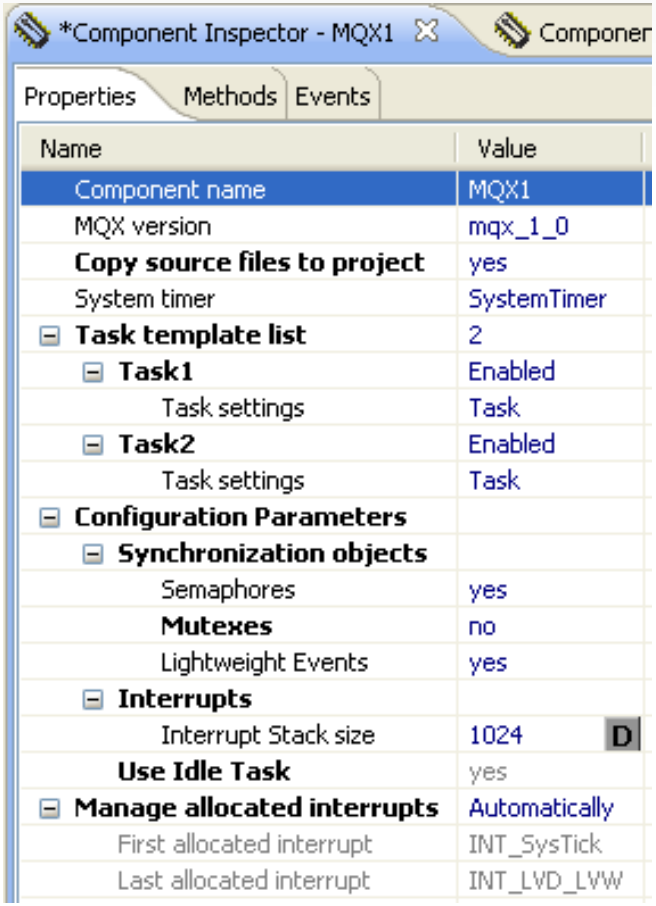
- **Minimal App – Hello Task, Idle task, interrupt stack**
  - Code = 10.4K
  - Data = 3.7K (including 1.5K for stacks)

- **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

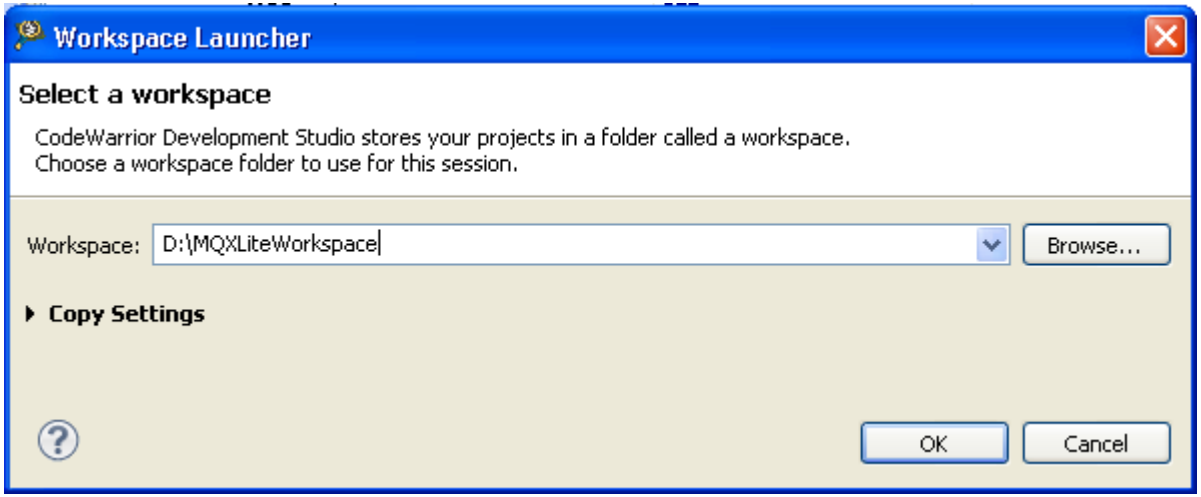


Enough Talk, Time to Work

- **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

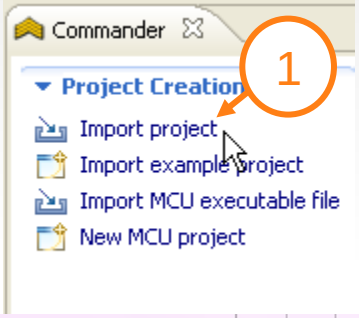
- This makes sure we don't have any confusion
- File→Switch Workspace→Other



- 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.

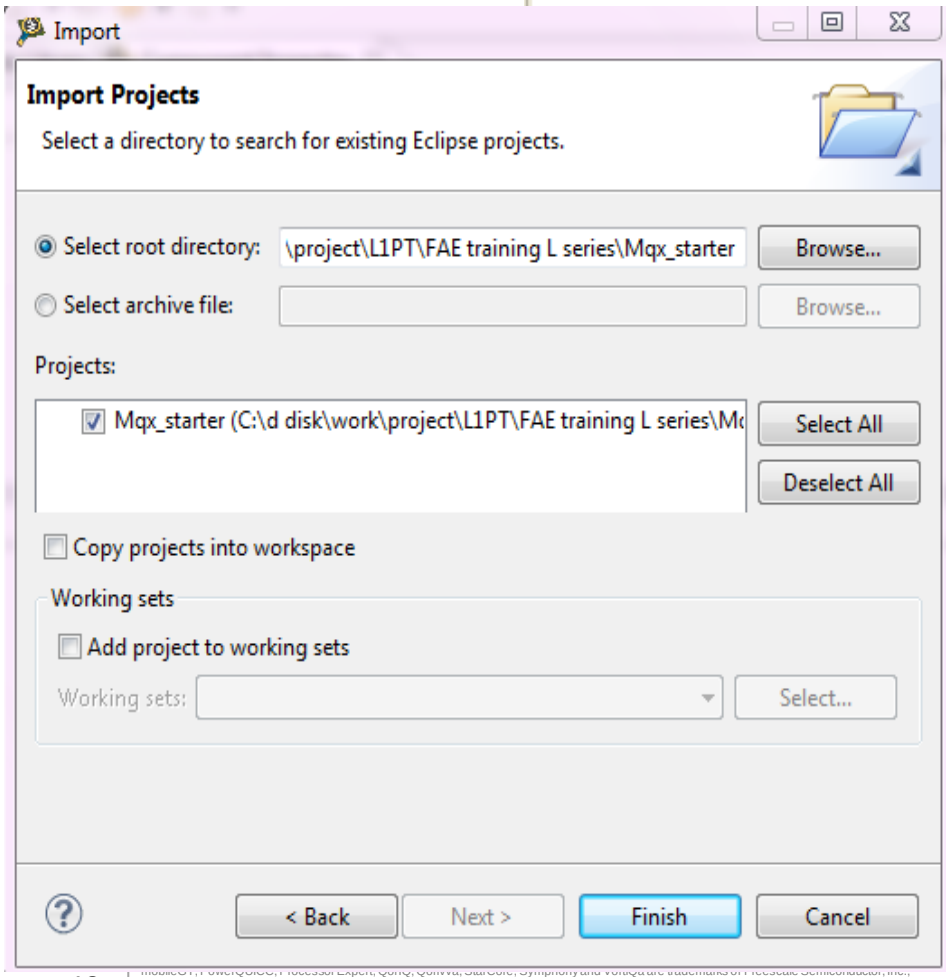
Import the starter project

- Click Import Project in Commander view
  - File→Import→General→Existing Projects Into Workspace



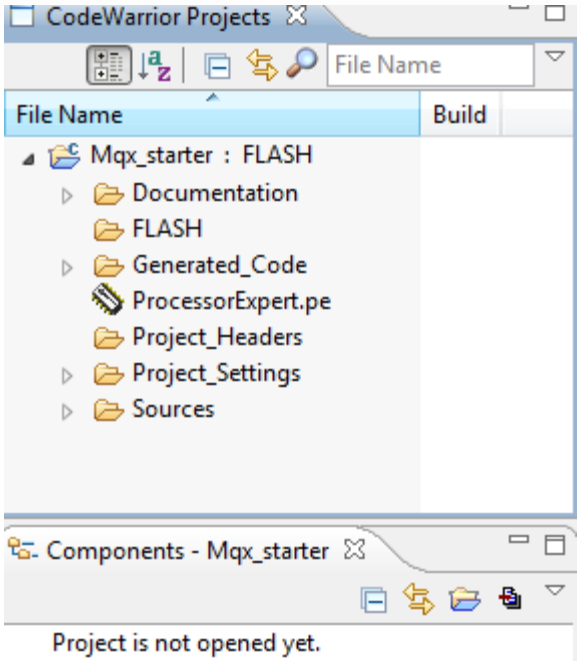
- Select root directory

- C:\d disk\work\project\L1PT \FAE training L series\Mqx_starter



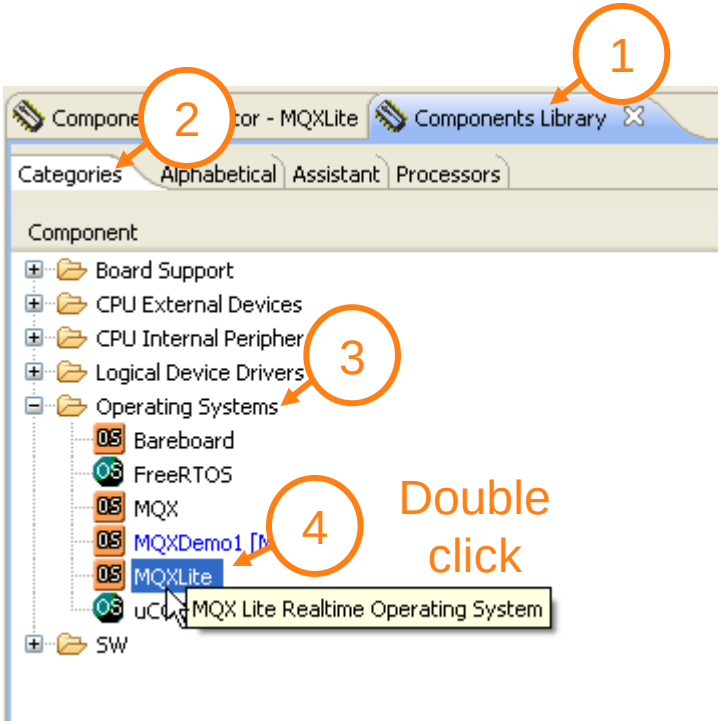
Open Processor Expert project

- 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



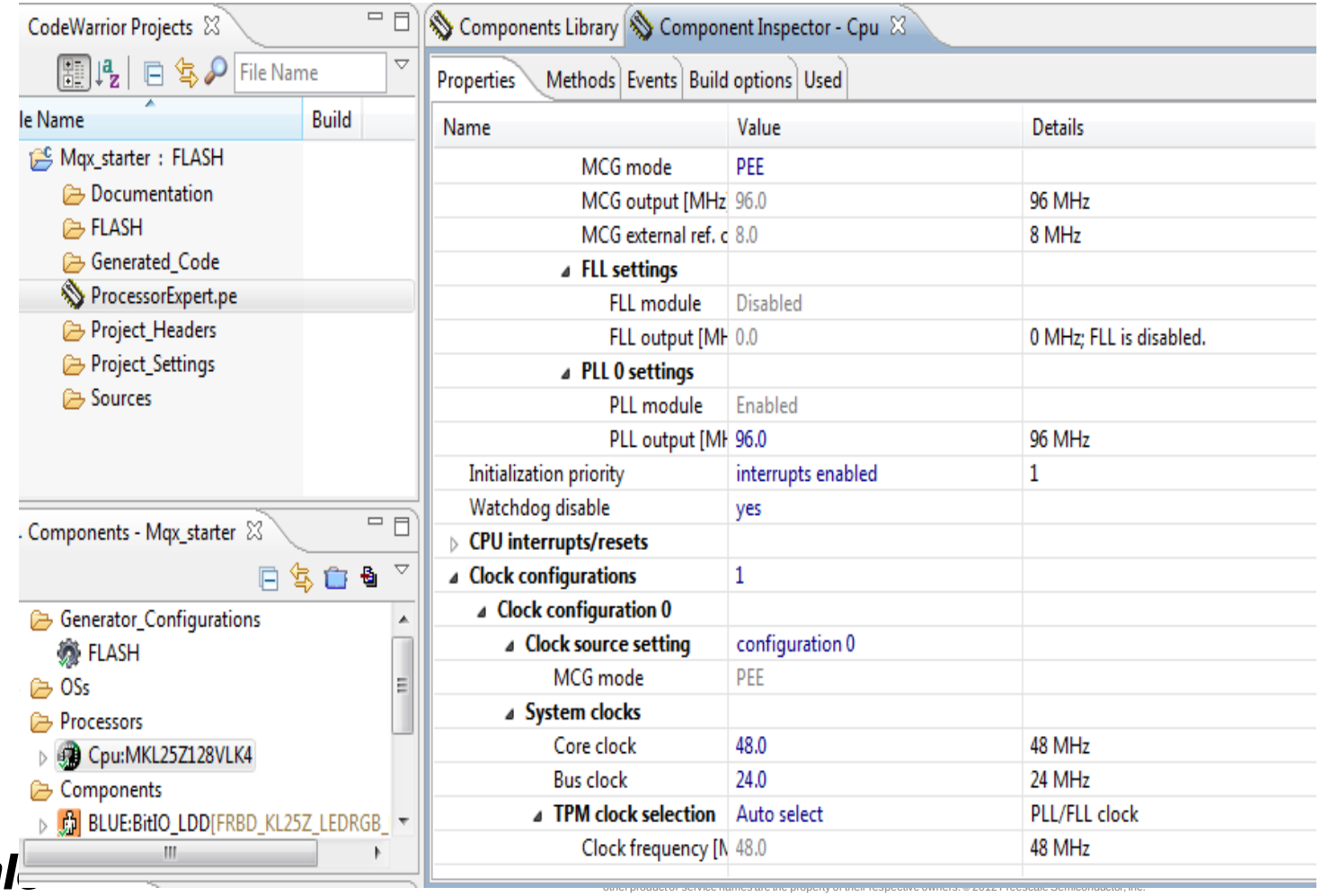
Add the MQX Lite Component

- **Go to Library, Categories, Operating Systems**
- **Double-click MQXLite**



Examine CPU Component

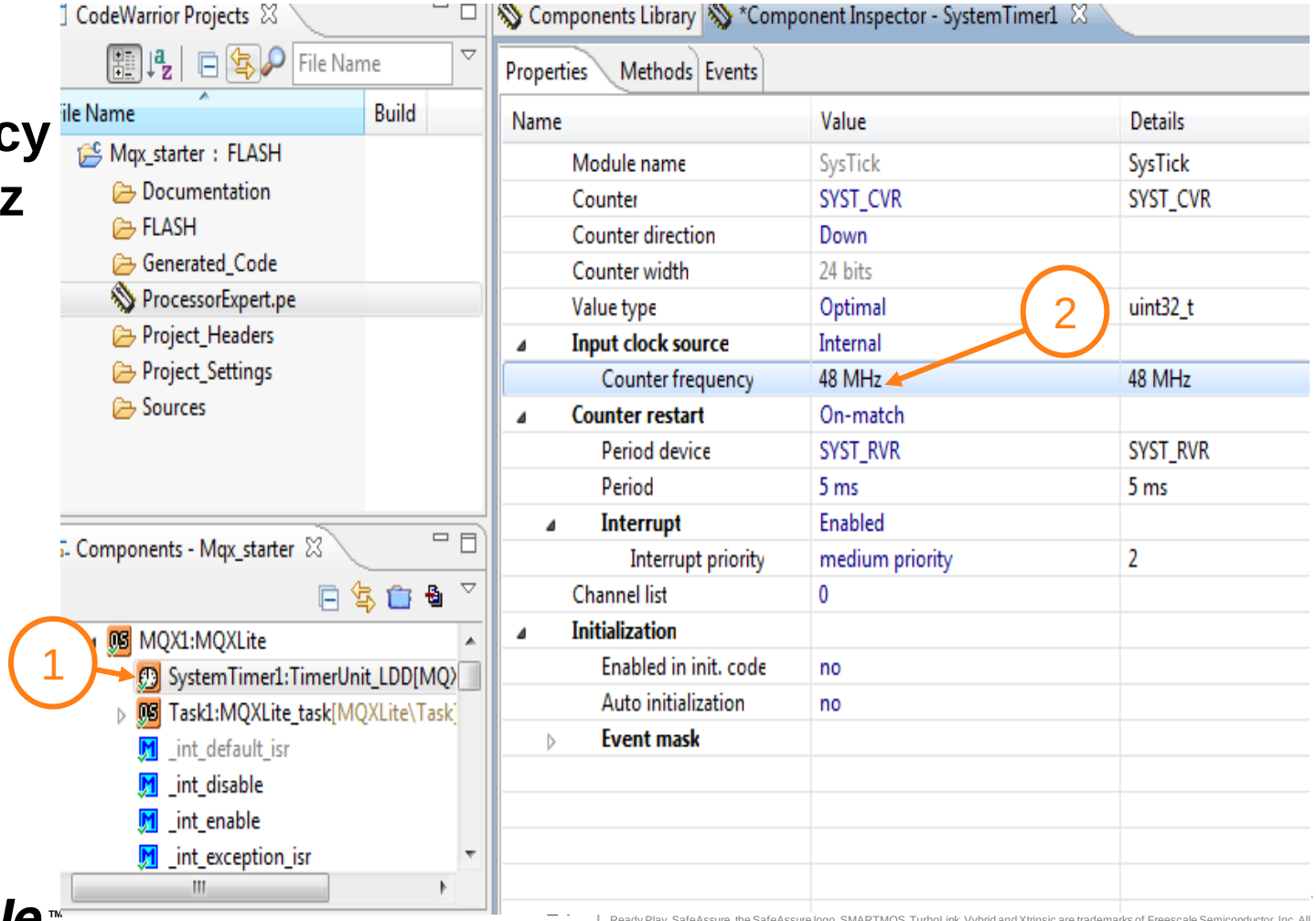
- Provided and configured in starter project
- Select CPU, go to component inspector, look at System Clock
- 48 MHz



freescale

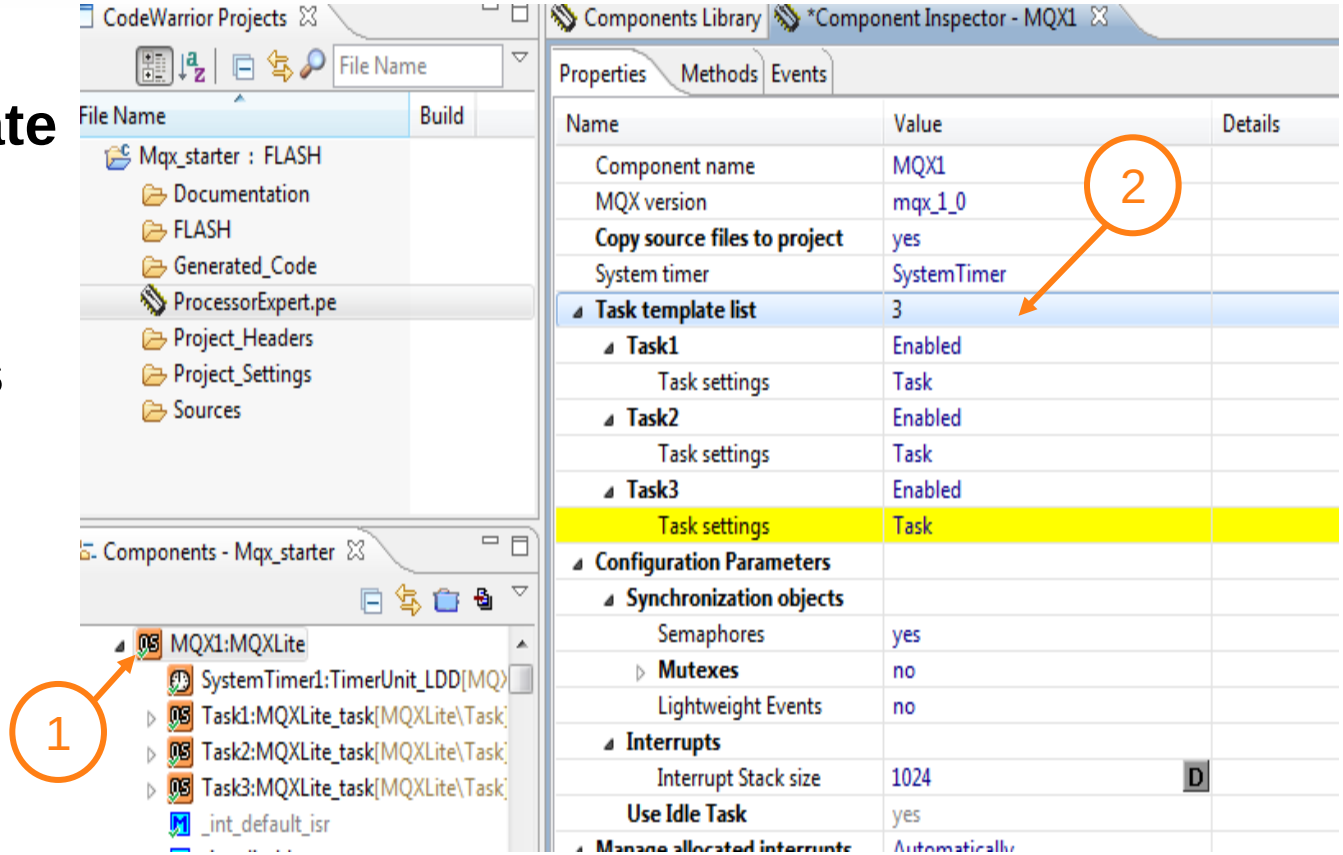
Configure System Timer

- **The MQX Lite Component depends on a timer – which must synch with the system**
- **frequency = 48 MHz**



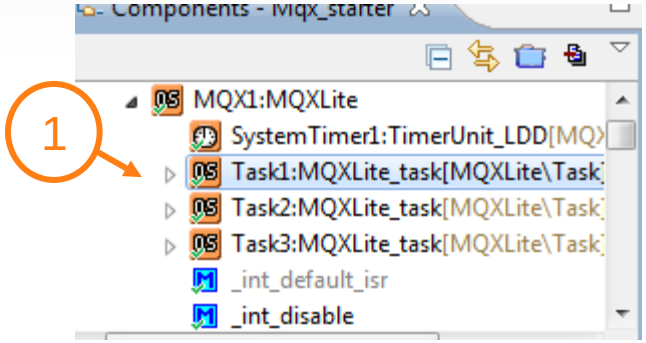
Configure the Component

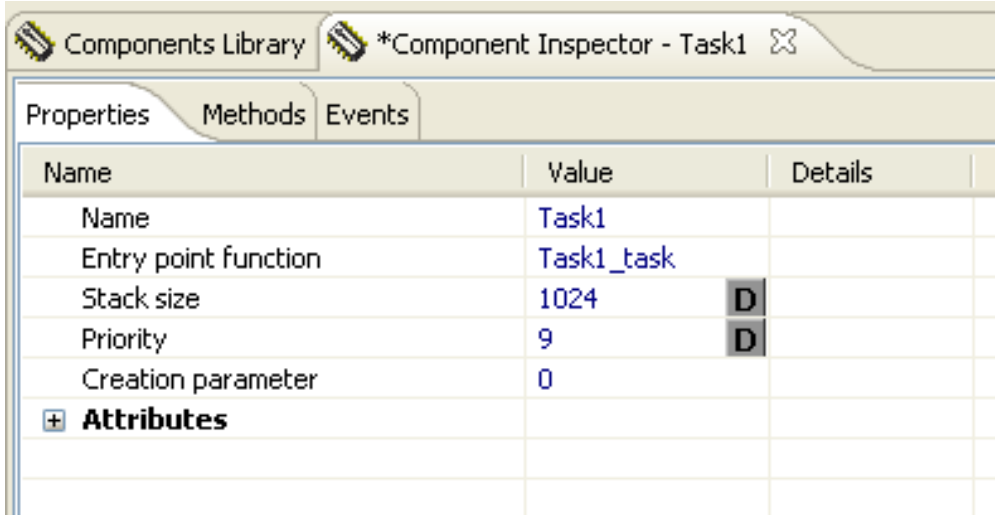
- **We will have two tasks, so increase by one**
- **Task Template List = 3**
- **Other values are default**



Configure the First Task

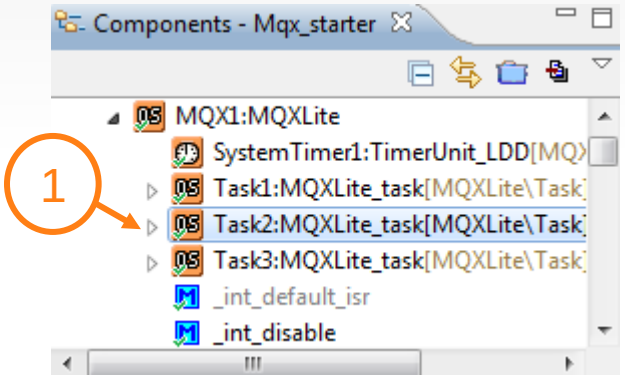
- **Select the task**
- **Examine the properties**
- **Actually, default values for all properties**
- **Task 1**
  - **Name = Task1 (default)**
    - **Case matters!**
      Code depends on this
  - **Entry point function**
  - **Stack size = 1024**





Configure the Second Task

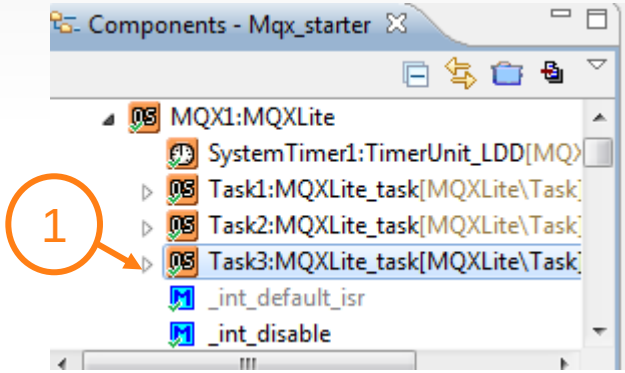
- **Task2**
  - **Name = Task2 (Default)**
    - **Case matters! Code depends on this**
  - **Priority = 9**
  - **AUTO_START_TASK = Enable**
    - **we will instantiate and start the task in our code**

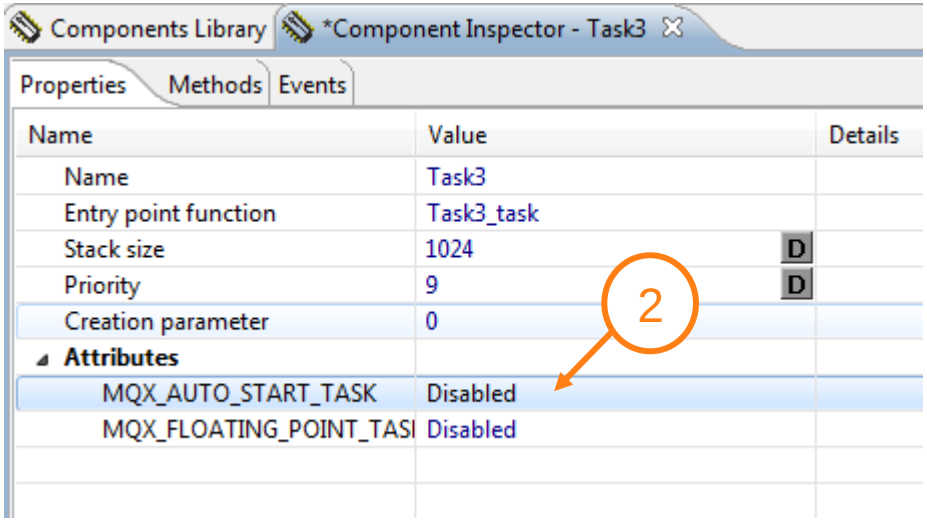


Name	Value	Details
Name	Task2	
Entry point function	Task2_task	
Stack size	1024	
Priority	9	
Creation parameter	0	
Attributes		
MQX_AUTO_START_TASK	Enabled	
MQX_FLOATING_POINT_TASI	Disabled	

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
- 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

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

This code creates the tasks – mqxlite.c

MQX1.h | mqxlite.c

```c
_mqx_uint _mqxlite(void)
{ /* Body */
    KERNEL_DATA_STRUCT_PTR              kernel_data;
    MQXLITE_TASK_TEMPLATE_STRUCT_PTR    template_ptr;
    TD_STRUCT_PTR                       td_ptr;

    _GET_KERNEL_DATA(kernel_data);

    /* Start Tick timer */
    MQXLITE_RTOS_ADAPTER_SYSTEM_TIMER_START(NULL);

    /* Create the idle task */
#if MQX_USE_IDLE_TASK
    td_ptr = _task_init_internal((MQXLITE_TASK_TEMPLATE_STRUCT_PTR)&kernel_data->IDLE_TASK_TEMPLATE,
                                 kernel_data->ACTIVE_PTR->TASK_ID,
                                 (uint_32)0,
                                 FALSE,
                                 (pointer)kernel_data->IDLE_TASK_TEMPLATE.TASK_STACKADDR,
                                 (_mem_size)kernel_data->IDLE_TASK_TEMPLATE.TASK_STACKSIZE);
#if MQX_CHECK_ERRORS
    if (td_ptr == NULL) {
        _mqx_exit(MQX_OUT_OF_MEMORY);
    } /* Endif */
#endif
    _task_ready_internal(td_ptr);
#endif

    /* Check here for auto-create tasks, and create them here */
    template_ptr = kernel_data->INIT.TASK_TEMPLATE_LIST;
    while (template_ptr->TASK_TEMPLATE_INDEX) {
        if (template_ptr->TASK_ATTRIBUTES & MQX_AUTO_START_TASK) {
            td_ptr = _task_init_internal(template_ptr,
                                         kernel_data->ACTIVE_PTR->TASK_ID,
                                         template_ptr->CREATION_PARAMETER,
                                         FALSE,
                                         (pointer)template_ptr->TASK_STACKADDR,
                                         (_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

```
/* MQX task template list */
const TASK_TEMPLATE_STRUCT MQX_template_list[] =
{
  /* Task: Task1 */
  {
  /* Task number                    */  TASK1_TASK,
  /* Entry point                    */  (TASK_FPTR)Task1_task,
  /* Stack size                     */  TASK1_TASK_STACK_SIZE,
  /* Task priority                  */  9U,
  /* Task name                      */  "task1",
  /* Task attributes                */  (MQX_AUTO_START_TASK),
  /* Task parameter                 */  (uint32_t)(0)
  },
  /* Task: Task2 */
  {
  /* Task number                    */  TASK2_TASK,
  /* Entry point                    */  (TASK_FPTR)Task2_task,
  /* Stack size                     */  TASK2_TASK_STACK_SIZE,
  /* Task priority                  */  9U,
  /* Task name                      */  "task2",
  /* Task attributes                */  (MQX_AUTO_START_TASK),
  /* Task parameter                 */  (uint32_t)(0)
  },
  /* Task: Task3 */
  {
  /* Task number                    */  TASK3_TASK,
  /* Entry point                    */  (TASK_FPTR)Task3_task,
  /* Stack size                     */  TASK3_TASK_STACK_SIZE,
  /* Task priority                  */  9U,
  /* Task name                      */  "task3",
  /* Task attributes                */  (0),
  /* Task parameter                 */  (uint32_t)(0)
  }
```

Task1 Code

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

```
**      Returns      : Nothing
** ========================================
*/
void Task1_task(uint32_t task_init_data)
{
  int counter = 0;
  printf("task 1 start running!\n");
  while(1) {
    counter++;

    /* Write your code here ... */
    BLUE_ClrVal(0);
        _time_delay_ticks(50);
        BLUE_SetVal(0);
        _time_delay_ticks(50);
  }
}
```

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

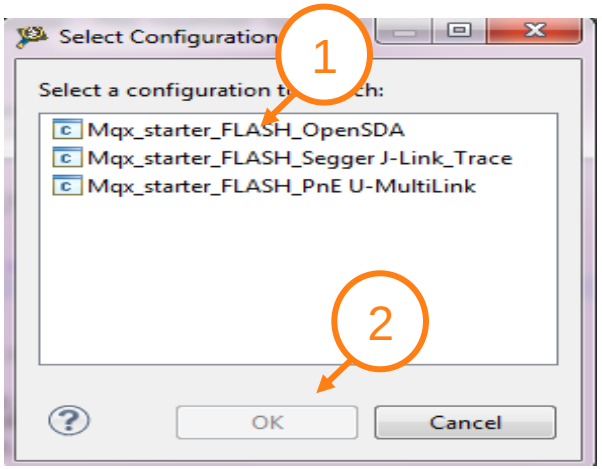
- Loops endlessly flashing RED

```
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 */
.07 void 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

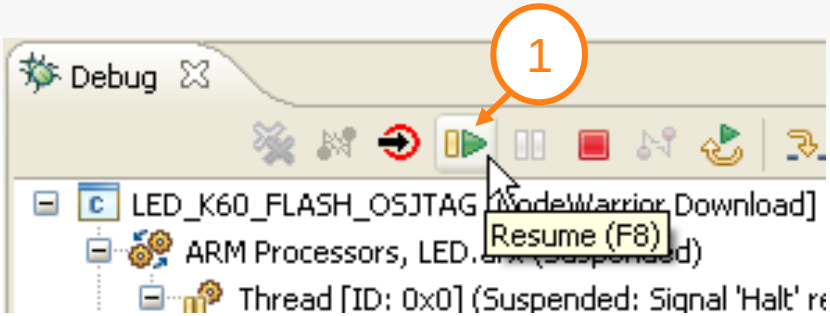


- Debug perspective appears
- Code stops at first line of main()
- Set a breakpoint at line 132 of mqx_tasks.c

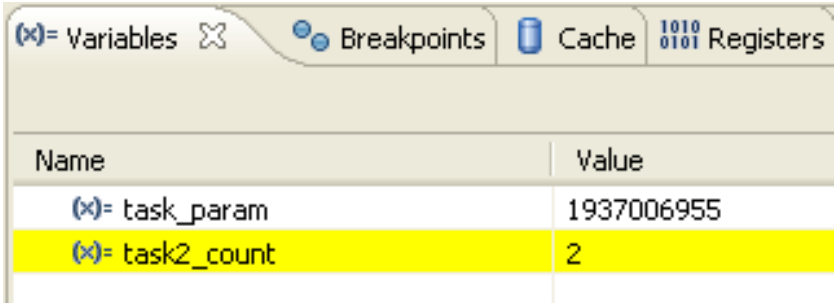
```
45  while(1) {
46      counter++;
47
48    /* Write your code here ... */
49    BLUE_ClrVal(0);
50        _time_delay_ticks(50);
51        BLUE_SetVal(0);
52        _time_delay_ticks(50);
```

Run

- Click the Resume button



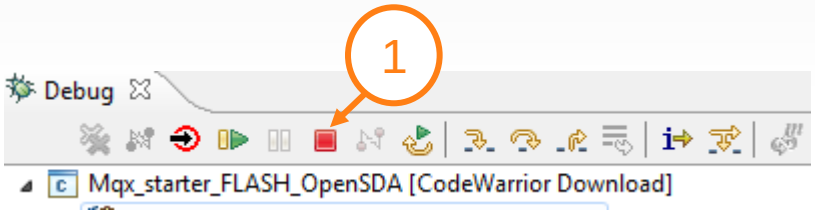
- Resume each time you stop
  - You'll see the count variable increase, and the lights on the board flash



- Remove the breakpoint to run the app without interference
  - Right click, Toggle breakpoint

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 MQXLite

- Now – if there's time, for more fun