



Hands-on Workshop: MQX™ Lite APF-ENT-T1286

Susan Su
IMM FAE

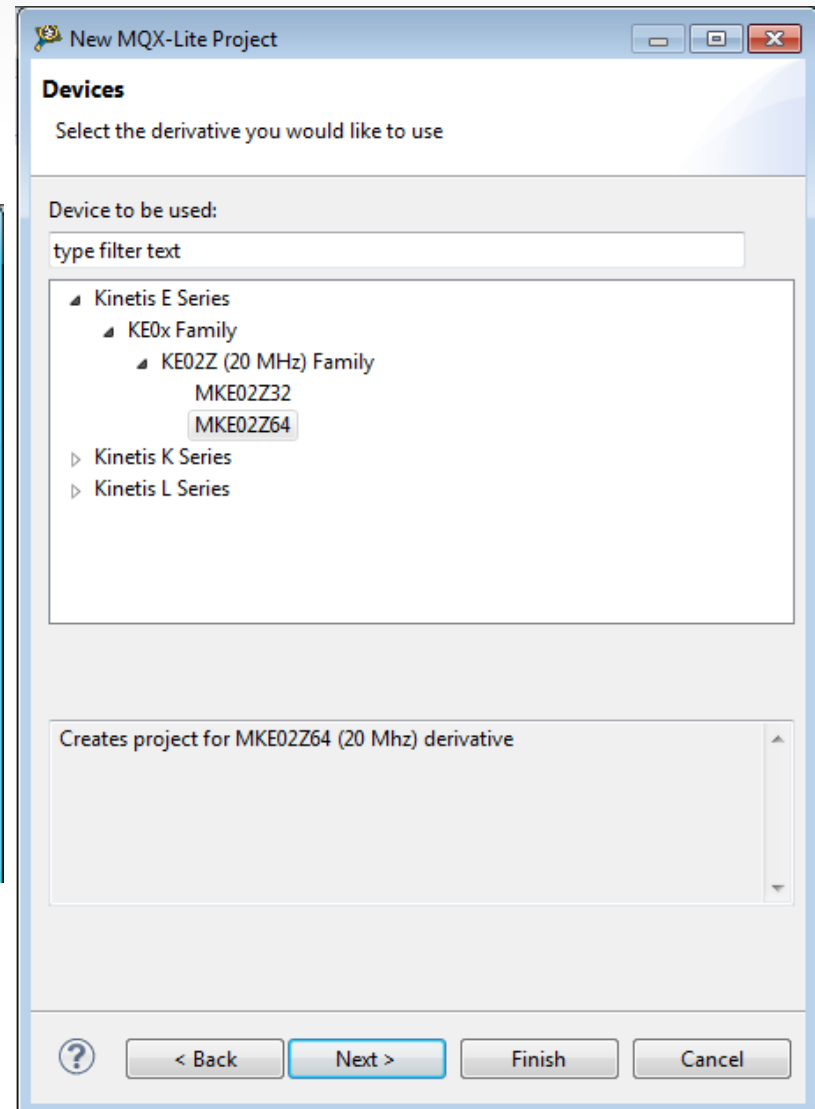
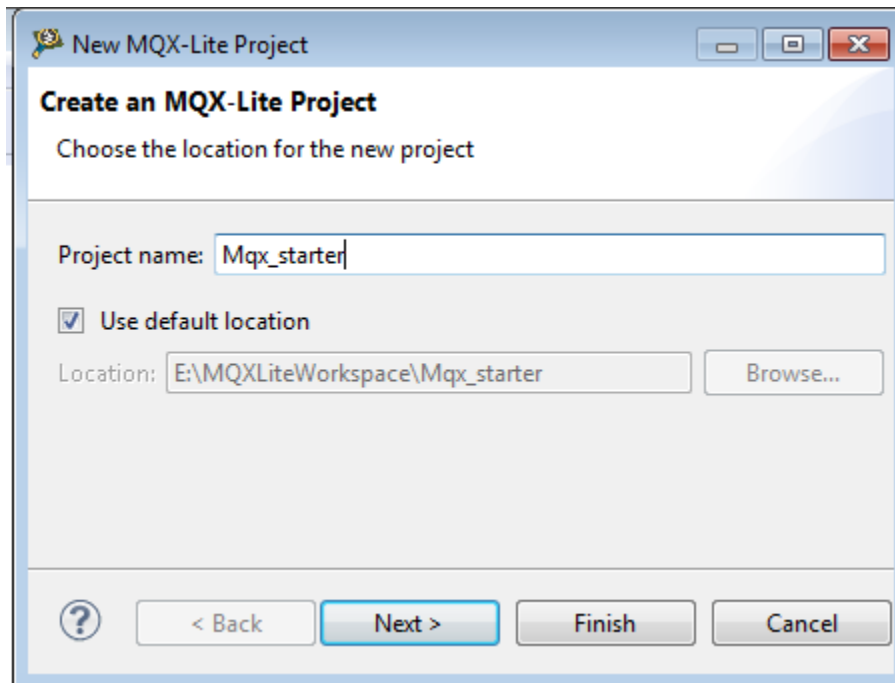


October 2013

Freescale, the Freescale logo, ARMv6, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Wire, the Energy Efficient Solutions logo, iMoteM, iMoteMGT, PGG, PowerQUICC, Processor Expert, QoIQ, QoIQv2, SafeAssure, the SafeAssure logo, StarCore, Symphony and VortiQo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. AirStar, BeeBit, BeeStack, Coresight, Flexis, LayerScope, Magick, M6C, Platform in a Package, QoIQ Converge, QUICC Engine, ReadyPlay, SMARTMOS, Tower, TurboLink, Vybrid and Vybrid are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2013 Freescale Semiconductor, Inc.

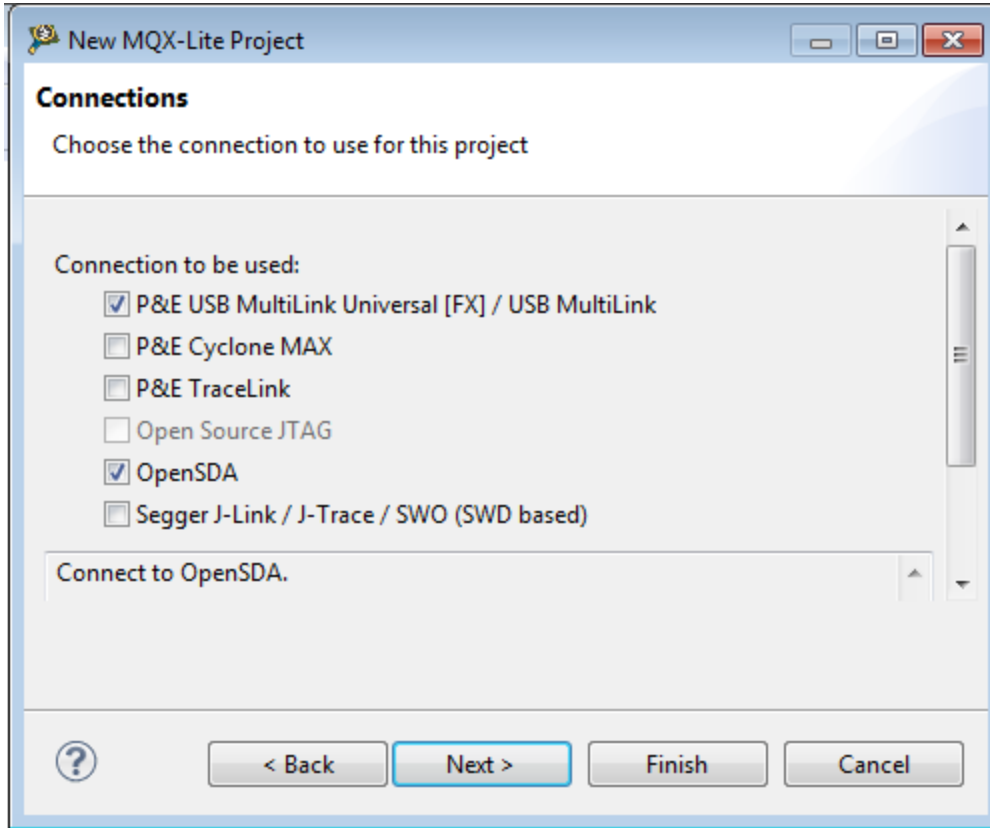
Create a new MQX Lite project

- Click Import Project in Commander view
 - File → new → MQX-Lite Project



Create a new MQX Lite project

- Choose openSDA as connection interface



Examine CPU Component

- Provided and configured in starter project
- Select CPU, go to component inspector, look at System Clock
- 16.7MHZ

The screenshot shows the CodeWarrior IDE interface. On the left, the 'CodeWarrior Projects' tree shows the 'Mqx_starter : FLASH' project structure. Below it, the 'Components - Mqx_starter' tree shows the selected CPU component 'Cpu:MKE02Z64VQH2'. On the right, the 'Component Inspector - Cpu' window is open, displaying the 'Properties' tab with a table of settings.

Name	Value
CPU type	MKE02Z64VQH2
▲ Clock settings	
▲ Internal oscillator	
Slow internal reference clock [kHz]	32.768
▶ System oscillator	Disabled
▲ Clock source settings	1
▲ Clock source setting 0	
▶ ICS settings	
Initialization priority	interrupts enabled
Watchdog disable	yes
▶ CPU interrupts/resets	
▲ Clock configurations	1
▲ Clock configuration 0	
▲ Clock source setting	configuration 0
ICS mode	FEI
▲ System clocks	
Core clock	16.777216
Bus clock	16.777216

Add and configure ConsoleIO component

- Open Components Library
- CPU External Devices -> Display -> ConsoleIO
- Double click the component to add it into project

The screenshot displays the CodeWarrior IDE interface. On the left, the 'CodeWarrior Projects' window shows a tree view for 'Mqx_starter : FLASH', with 'ProcessorExpert.pe' selected. Below it, the 'Components - Mqx_starter' window shows a tree view with 'CsIO1:ConsoleIO' selected under the 'Components' folder. On the right, the 'Components Library' window is open, showing a tree view of components. The 'CPU External Devices' folder is expanded, and the 'Display' folder is also expanded, with 'ConsoleIO' selected. The 'Component Inspector - CsIO1' window is also visible at the top right.



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);
```


Task3 Code

- Loops endlessly flashing RED

```

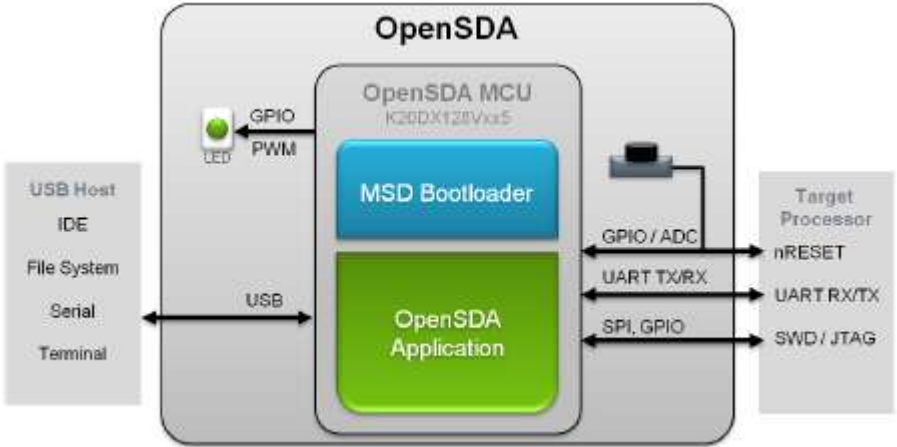
.98 **    Description :
.99 **        MQX task routine. The routine is gen
.00 **        file.
.01 **    Parameters :
.02 **        NAME          - DESCRIPTION
.03 **        task_init_data -
.04 **    Returns       : Nothing
.05 ** =====
.06 */
.07 void Task3_task(uint32_t task_init_data)
.08 {
.09     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

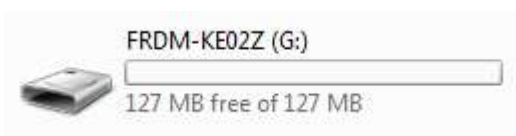
Introduction to openSDA

- Open-standard serial and debug adapter.
- Bridges serial and debug communications between a USB host and an embedded target processor.
- OpenSDA software includes a flash-resident USB mass-storage device (MSD) bootloader and a collection of OpenSDA Applications. FRDM-KE02Z preinstalled with the MSD Flash Programmer OpenSDA Application.



Using the MSD Flash Programmer

- MSD Flash Programmer - a composite USB application that provides a virtual serial port and an easy and convenient way to program applications into the KE02Z MCU. It emulates a FAT16 file system, appearing as a removable drive in the host file system with a volume label of FRDM-KE02Z.

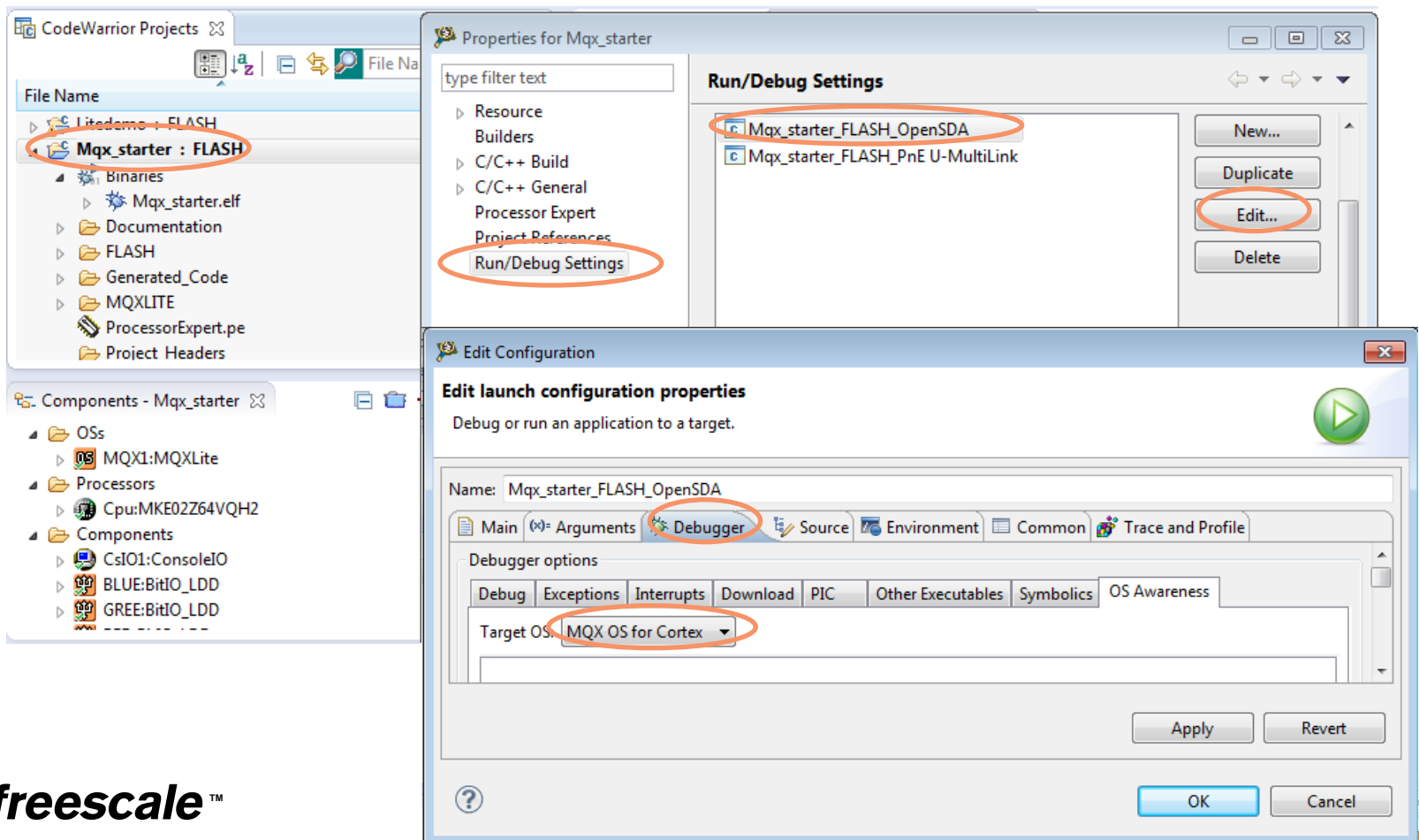


- 1 Locate the Precompiled Examples folder in the FRDM-KE02Z Quick Start Package.
- 2 Copy & paste or drag & drop one of the .srec files to the FRDM-KE02Z drive.


The new application should now be running on the FRDM-KE02Z. Starting with the MSD Flash Programmer, you can program repeatedly without the need to unplug and reattach the USB cable before reprogramming.

Enable OS Awareness to debug MQX Lite tasks

- Choose the project, right click and choose Properties -> Run/Debug Settings -> OpenSDA ->Edit
- Edit Configuration -> Debugger -> OS Awareness



Debug MQX Lite tasks

- Click resume  to run the application, the application will stop at the break point.

Debug - MQXLitedemo/Sources/mqx_tasks.c - CodeWarrior Development Studio

File Edit Source Refactor Search Project MQX Tools RTCS MQX PEMicro Run Window Help

Debug Console:

- MQXLitedemo_FLASH_OpenSDA [CodeWarrior]
- ARM Processors, MQXLitedemo.elf (Suspended)
- Thread [ID: 0x10001] (Suspended: Signal 'Process Suspended' received. Description: Process suspended)
- 2 _mqx_idle_task() idletask.c:56 0x00002139
- 1 _task_exit_function_internal() task.c:2980 0x0000144c
- Thread [ID: 0x10003] (Suspended: Signal 'Process Suspended' received. Description: Process suspended)
- 2 Task2_task() mqx_tasks.c:85 0x00002605
- 1 _task_exit_function_internal() task.c:2980 0x0000144c
- Thread [ID: 0x10002] (Suspended: Breakpoint hit.)
- 2 Task1_task() mqx_tasks.c:60 0x000025e0
- 1 _task_exit_function_internal() task.c:2980 0x0000144c

Source Code (mqx_tasks.c):

```
void Task1_task(uint32_t task_init_data)
{
    int counter = 0;
    printf("task 1 start running!\n");
    while(1) {
        counter++;
    }
    /* Write your code here ... */
}
```

Variable Window:

Name	Value
(x)- task_init_data	0
(x)- counter	0

Disassembly:

```
000025da: mov r0,r3
000025dc: bl puts (0x26a8) ; 0x000026a8
60 counter++;
000025e0: ldr r3,[r7,#12]
000025e2: adds r3,#1
000025e4: str r3,[r7,#12]
63 BLUE_ClrVal(0);
```


Debug MQX Lite tasks

- Click  to continue running application –four tasks.

The screenshot shows the CodeWarrior Development Studio interface during a debug session. The main window displays the source code for `mqx_tasks.c`, with the `Task1_task` function highlighted. The `counter` variable is shown to have incremented to 1. The `MQX Task Summary` table provides an overview of the tasks and their states.

Task Name	Task ID	TD	Priority	State
▷ <code>_mqx_idle_task</code>	0x10001	0x20000388	9	Ready
▷ <code>task1</code>	0x10002	0x200006b4	8	Active
▷ <code>task2</code>	0x10003	0x200004a8	8	Ready
▷ <code>task3</code>	0x10004	0x2000007c	8	Time delay blocked

Task	Stack Base	Stack Limit	Stack Used	% Used	O
▷ <code>_mqx_idle_task</code>	0x200004a0	0x20000400	0x2000042c	72 %	Ni
▷ <code>task1</code>	0x200008b0	0x20000720	0x200007f4	47 %	Ni
▷ <code>task2</code>	0x200006b0	0x20000520	0x20000560	84 %	Ni
▷ <code>task3</code>	0x20000280	0x200000f0	0x200001c4	47 %	Ni
Interrupt	0x20000070	0x1fffff70	0x1fffffdc	57 %	Ni

