Create Low-Power Applications with MQX™ and MQX™ Lite RTOS

FTF-SDS-F0462

Shaozhong Liang | Field Application Engineer

M A Y . 2 0 1 4
Hands-on Workshop: Everything You Want to Know About Entering and Leaving Low-Power Modes using MQX™ RTOS

FTF-SDS-F0462
2 Hour Class

Learn how to take advantage of the power management capabilities of MQX and MQX Lite RTOS. See how to create feature-rich applications without killing battery life. Attendees will get hands-on experience with system power mode transitions, driver state transitions and slowing or stopping the system tick timer for power savings.
Session Objectives

• After completing this session you will be able to:
  - Understand how power management capabilities can be used in RTOS-based applications
  - Understand how to perform system power mode transitions, driver state transitions, and slow or stop the RTOS system tick timer for power savings
Agenda

• MQX RTOS Overview
• Tools for Analyzing MCU Power
• Kinetis MCU Low Power Features
• MQX RTOS Power Management Features
  – Hands On
• Optimizing Applications for Low Power
  – Architecting State Transitions
  – Optimizing/Disabling the System Tick Timer
  – Hands On
• MQX Lite RTOS Power Management Features
  – Hands On
Implementing the techniques discussed in this class, we were able to reduce power consumption by TBD%.
Freescale Bundled MQX RTOS

**Free** Scalable, fully-featured and proven RTOS with 32-bit MCUs

- **Full-featured and powerful**
  - BSPs incorporate tightly integrated RTOS, Middleware (USB, TCP/IP stacks), file system, and I/O drivers
  - Designed for speed and size efficiency
- **Market proven**
  - Available on Freescale processors for 20+ years
  - Used in millions of products including Medical and Heavy Industrial applications
- **Simple and scalable**
  - As small as ~10KB for smallest implementation, or scale up to support sophisticated networking and threading
  - Intuitive API & modular architecture enables straight-forward fine-tuning of features
  - Production source code provided
- **Similar to other “pay-for” software OS**

Software integration headache

$95K of free Software

Integrated MQX Solution

- Stable
- Upgradable
- Easy to maintain
Proven: 20+ years in the market place

Used in millions and millions of products in Medical, Industrial, and Defense Fortune 500 Companies.

Freescale holds a broad license to MQX across its portfolio.


Precise Founded
• MQX Began

ARC acquired Precise with MQX

Embedded Access Inc. (EAI)
• Licensed Provider of MQX
• Roots from Precise & ARC
• Strategic Freescale Partner

Introducing Freescale MQX along with MCF5225x
• Acquired MQX license across Freescale’s portfolio from ARC.

2009  2010  2011  2012  2013
Why use an RTOS?

• Create Deterministic and Connected Designs
  – Lower latency verses super loop software architecture
  – Multiple communication interfaces much easier to manage with an RTOS
  – Integrated middleware and upper layer protocols such as TCP/IP, USB, Flash Systems, Web Servers, Wi-Fi support, CAN protocols, SSL, SNMP

• Ease of Development
  – Board Support Packages (BSPs) available with peripheral drivers
  – Spend time developing application code, not maintaining a scheduling system

• Portability and Scalability
  – Standard APIs for high portability of application code to other MCUs
  – Configurable features to scale across products
  – Enable only features needed for low overhead

• Maintainability and Stability
  – Add new features without affecting system timing and higher priority functions
Freescale MQX Growing in Popularity!

Freescale MQX
3rd Most Popular
MCU RTOS in
2013 UBM Survey
MQX: What’s free and what’s add-on?

**Free Components**
- RTOS (priority-based, pre-emptive scheduler)
- Real-time TCP/IP Communication Suite (RTCS) - TCP/IP, FTP, Telnet, DHCP, SNMP, …
- USB Host – PHDC (medical), HID, MASS, HUB, CDC, …
- USB Device - HID, MASS, CDC, PHDC
- MS-DOS File System (MFS)
- BSP I/O Driver: CAN, UART etc…
- Basic HTTP Web server
- Flash File System  **NEW**
- Wi-Fi Support Patches  **NEW**
- Benchmarking Tool
- Freescale MQX Level 1 Support

**Add-on Components**
- Freescale Low Cost NanoSSL™ and NanoSSH™ Client Software
- Bluetooth
- RTA & IXXAT: Industrial Protocols - 1588, CANopen etc.
- Swell PEG Window Builder/ Graphic Libraries
- Segger: Graphic Libraries
- MQX Task Aware Debug plug-ins for CW, IAR, Keil, DS5.
- Freescale MQX Level 2 Support
Tool Chain and MQX Task-aware Debugging

- **Advanced kernel analysis tool**
  - Allows developers to gain greater visibility into their embedded systems
  - Obtain detailed data about system performance, enabling optimization work that can reduce potential performance bottlenecks in embedded applications
- TAD is included in CodeWarrior Professional Edition (30-day evaluation available) for ColdFire® V2 Core and above
- TAD is included with IAR’s C-SPY Debugger for additional debugging functionality
MQX Runtime Debug & Productivity Tools **NEW**

Runtime TAD - Plug-in for CW10.2 and later
- Access live TAD while the application is running.
- Stream live system performance data to TAD data views or FreeMaster data visualization tool

Performance Tool - Plug-in for CW10.2 and later
- Live graphical views of RTOS data
- Timeline of task switches, interrupts, synchronization objects, stack utilization,…

BSP Cloning Wizard - Desktop application
- Easy cloning of MQX projects. Enhances productivity when developing MQX-based embedded applications.
Hard Real Time Performance
Backed by built-in benchmarking tools

• Available in MQX examples/benchmark folder
• Generate code size and timing reports
• Pre-generated reports available for each platform
High-Level RTOS Landscape

Potential Size (Memory Footprint)

Features, Performance, Sophistication

Bare Metal (No OS)

Basic Featured Thread-Based RTOS

Full Featured Thread-Based RTOS

MQX
Nucleus
Integrity
ThreadX
CMX
eCOS

Linux
QNX
VxWorks
WinCE

MQX Lite
CMX-Tiny

FreeRTOS uCOS-II

Each “process” has own memory space. More Multi-media and sophisticated applications.

Low to high footprint.

More features. More integrated protocol stacks and middleware. Low to medium footprint.
Freescale *MQX Lite RTOS* Launched in 2012

<table>
<thead>
<tr>
<th></th>
<th>MQX RTOS</th>
<th>MQX Lite RTOS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Delivery Mechanism</strong></td>
<td>Traditional installer with full source for Kernel, services and BSPs</td>
<td>Processor Expert (PEx) Kernel and services component, configurable software generated by PEx</td>
</tr>
<tr>
<td><strong>I/O Drivers</strong></td>
<td>MQX POSIX compatible drivers with option for using PEx drivers</td>
<td>PEx drivers only</td>
</tr>
<tr>
<td><strong>Configurability</strong></td>
<td>User selects needed services from full or lightweight versions</td>
<td>Reduced services available; lightweight options only</td>
</tr>
<tr>
<td><strong>Example Footprint</strong></td>
<td>&lt;12 Kbytes FLASH</td>
<td>&lt;8 Kbytes FLASH</td>
</tr>
<tr>
<td>(3 tasks, sem, event)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Processor Support</strong></td>
<td>Kinetis K Series, ColdFire, Vybrid, PPC, DSC</td>
<td>Kinetis L and K Series</td>
</tr>
</tbody>
</table>
Tools for Analyzing MCU Power

Measuring dynamic power and debugging through low power modes
- Seamless integration with IAR Embedded Workbench
- Supports ARM7/9/11, ARM Cortex-M/R/A
- Plug-and-play compatible
- Download speed of up to 1MB/sec
- JTAG and SWD clocks up to 32MHz
- SWV with UART and Manchester encoding
- Support for SWO speeds of up to 60MHz
- Support for MTB for ARM Cortex-M0/M0+
- Enables high-resolution measurements of target power consumption, ~200μA resolution at 200kHz
- Can supply the target board with power, entirely powered by USB

What is Power Debugging?
- Samples power consumption and correlates it with the MCU’s program counter
- Source code debugger visualizes power consumption data both statically and dynamically in different views
- Provides a view of the power profile of an application
- Measures current and voltages and sends it to I-jet, which synchronizes the data with the program counter of the running application.
- The data can be graphed and profiled in real time and analyzed using the C-SPY Debugger in IAR Embedded Workbench.
- The current sensing is done by connecting two differential current measurement leads across a shunt resistor on the target board.

- I+ and I- differential voltage, 110mV full scale across shunt resistor
- One differential current channel, 0-6V common mode
- Three voltage channels, 0-6V
- Sampling rate up to 200 kHz with 12 bit resolution
IAR I-scope

- 12-bit A/D converter for current and voltage measurements.
- Current Measurements
  - Implemented using a sensitive differential amplifier
  - Measures differential voltage across a shunt resistor
  - max voltage = 100mV
- For ~41mA max we are using 2.4 ohm shunt (this gives us plenty of headroom)

What power modes does it work in? Talk to IAR about this
Debugging through Low Power States

- Some low power modes shut off the debug engine

<table>
<thead>
<tr>
<th>Module</th>
<th>STOP</th>
<th>VLPR</th>
<th>VLPW</th>
<th>VLPS</th>
<th>LLS</th>
<th>VLLSx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug Port</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>OFF</td>
<td>static</td>
<td>OFF</td>
</tr>
<tr>
<td>AHB-AP</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>OFF</td>
<td>static</td>
<td>OFF</td>
</tr>
<tr>
<td>ITM</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>OFF</td>
<td>static</td>
<td>OFF</td>
</tr>
<tr>
<td>TPIU</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>OFF</td>
<td>static</td>
<td>OFF</td>
</tr>
<tr>
<td>DWT</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>OFF</td>
<td>static</td>
<td>OFF</td>
</tr>
</tbody>
</table>

- The IDE may or may not support graceful reconnect of the debug connection if the low power state was exited
  - Most of the IDE debugger tools crash when low power mode is entered, unless it's stop mode. When the debugger is attached and STOP is entered, it really enters a pseudo stop mode with clocks still on to the debugger.

- When measuring power during the very low power states (~uA); please, detach the debugger.
CodeWarrior
Support for debugging through Low Power Modes

- In LLS and VLLSx debugger loses connection with target and an reconnect action is needed to continue with debug session once the target has recovered from these states

- Control for reconnect behavior can be defined either on the first pop-up windows when a connection is lost either in:
  - Remote System View → Properties of the connection used → Advanced

- NOTE: behavior is saved per connection, but you can re-use connections in multiple projects
Kinetis MCU Low Power Features

Overview of the power-sipping technology making Kinetis microcontrollers some of the most power efficient devices in the industry.
Kinetis MCU Low Power Features

- Overview of the power-sipping technology making Kinetis microcontrollers some of the most power efficient devices in the industry.
Why worry about power?

• There are many reasons why a designer of an embedded system is increasingly striving to stay within tight power consumption budgets.

• One reason is money, having a portable device with a poor battery life can be the cause of losing in the market against more power efficient competitors.

• There are other applications in which money goes to second term, like medical devices that are implanted in a patient’s body. This device needs to run for years on battery life, replacing the battery means taking the patient to surgery, so in this scenario an efficient power solution is critical.
Low power overview

- Freescale offers many low-power microcontrollers (MCUs), including the 32-bit Kinetis family. Kinetis offers several different low-power modes of operation, providing very low standby and run current consumption.
Kinetis: Low Power Capabilities

• Flexible power modes
  – 10 Run, Wait & Stop modes – customize power usage to application requirements

• Industry leading 90nm process technology
  – 1/3 dynamic power reduction vs. existing technologies
  – Intelligent power management controller reduces dynamic and leakage currents

• Low power design techniques
  – Clock gating: only leakage currents are incurred
  – Power gating: shuts down un-used modules and memory reducing leakage

• Ultra fast wake up times
  – 4μs wake up from low leakage stop mode
  – New Low Leakage Wake-up Unit for leakage current reduction and increased low power wake-up functionality
Low power overview

• The available modes of operation for the Kinetis family devices are the following:
  - Run
  - Wait
  - Stop
  - VLPR (Very Low Power Run)
  - VLPW (Very Low Power Wait)
  - VLPS (Very Low Power Stop)
  - LLS (Low Leakage Stop)
  - VLLS3 (Very Low Leakage Stop3)
  - VLLS2 (Very Low Leakage Stop2)
  - VLLS1 (Very Low Leakage Stop1)
  - VLLS0 (Very Low Leakage Stop0)
  - BAT (Backup battery only)
## Kinetis Power Modes

<table>
<thead>
<tr>
<th>Typical Power Modes in an embedded system</th>
<th>Cortex M4 Power Modes</th>
<th>Kinetis Extended Power Modes</th>
<th>Recovery Time</th>
<th>“Typical” Idd Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>Run</td>
<td>Run</td>
<td></td>
<td>278uA/MHz</td>
</tr>
<tr>
<td>Wait</td>
<td>Sleep</td>
<td>VLPR</td>
<td></td>
<td>867uA @ 4MHz</td>
</tr>
<tr>
<td>Stop</td>
<td>DeepSleep</td>
<td>Wait</td>
<td>5.2us</td>
<td>7.5mA @ 50MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VLPW</td>
<td></td>
<td>509uA @ 4MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stop</td>
<td>5.2us</td>
<td>310uA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VLPS</td>
<td></td>
<td>3.5uA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LLS</td>
<td>6us</td>
<td>2.1uA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VLLS3</td>
<td>7035us</td>
<td>1.5uA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VLLS2</td>
<td>70us</td>
<td>1.4uA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VLLS1</td>
<td>130us+EE restore</td>
<td>678nA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VLLS0</td>
<td></td>
<td>367nA/176nA</td>
</tr>
</tbody>
</table>

**Freescale Adds Low Leakage Wake-up Unit**

- Enables complete shut-down of core logic, including WIC, further reducing leakage currents in all low power modes
- Supports 16 external input pins and 8 internal modules as wakeup sources
- Wakeup inputs are activated in LLS or VLLS modes
# Kinetis Power Modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>MCU can be run at full speed.</td>
</tr>
<tr>
<td>Wait</td>
<td>Allows peripherals to function, while CPU goes to sleep reducing power consumption.</td>
</tr>
<tr>
<td>VLP Run</td>
<td>CPU and peripheral clock maximum frequency is restricted. CPU/Platform clock is restricted to 2 MHz. Flash access is restricted to 1 MHz. LVD is off.</td>
</tr>
<tr>
<td>VLP Wait</td>
<td>Similar to VLP Run, with CPU in sleep to further reduce power.</td>
</tr>
<tr>
<td>Stop</td>
<td>MCU is in static state. Lowest power mode that retains all registers while maintaining LVD protection.</td>
</tr>
<tr>
<td>VLP Stop</td>
<td>MCU is in static state with LVD operation off. Lowest power mode with ADC, LPT, RTC, LCD, HSCMP, DAC, and pin interrupts functional.</td>
</tr>
<tr>
<td>LL Stop</td>
<td>MCU is in low leakage state retention power mode. LLWU controls wakeup sources including LPT, RTC, LCD, HSCMP, DAC and select pin interrupts.</td>
</tr>
<tr>
<td>VLL Stop 3</td>
<td>MCU is placed in a low leakage mode powering down most internal logic. All system RAM contents are retained and I/O states held. LLWU controls wakeup sources including LPT, RTC, LCD, HSCMP, DAC and select pin interrupts.</td>
</tr>
<tr>
<td>VLL Stop 2</td>
<td>Similar to VLL Stop 3, with only partial system RAM retention. FlexRAM contents can optionally be retained.</td>
</tr>
<tr>
<td>VLL Stop 1</td>
<td>Similar to VLL Stop 3, with only 32 byte register file retention.</td>
</tr>
<tr>
<td>VLL Stop 0</td>
<td>Lowest power mode. Similar to VLL Stop 1, 32 byte register file retention only. LLWU, RTC can be operational. All SRAM powered off. Additionally, the 1kHz LPO clock is disabled and the power on reset (POR) circuit can be optionally enabled.</td>
</tr>
</tbody>
</table>
Low power overview

- **STOP - Normal Stop Mode**

Expect IDDs around 200 µa. Higher performance devices (100+ MHz) may be higher. MCU current will be higher when using a debugger since the ARM Cortex-M4 core will need to have clocks alive in Stop with the debugger enabled.

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Trigger Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>Stop</td>
<td>Execute STOP( ); This means that sleep-now or sleep-on-exit modes entered with SLEEPDEEP set</td>
</tr>
<tr>
<td>Stop</td>
<td>Run</td>
<td>Interrupt or Reset – Interrupt goes to ISR (no LLWU)</td>
</tr>
</tbody>
</table>

**Power mode transition operating behaviors for K20_50**

<table>
<thead>
<tr>
<th>Description</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>STOP → RUN</td>
<td>5.2 µs</td>
</tr>
</tbody>
</table>
Low power overview

- **VLPR – Very Low Power Run Mode**

Expect IDDs in the range of 1 to 1.5 mA for Kinetis K, 200 µa to 1 mA for Kinetis L.

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Trigger Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>VLPR</td>
<td>Reduce system bus and core frequency to 2 MHz or less</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Flash access frequency limited to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 MHz, AVLP = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set RUNM = 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: Poll VLPRS bit before executing VLPR specific code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(You can wait ~5 us instead of waiting for VLPRS)</td>
</tr>
<tr>
<td>VLPR</td>
<td>Run</td>
<td>Set RUNM = 00 or Interrupt with LPWUI = 1 or Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: Poll REGONS bit before increasing frequency.</td>
</tr>
</tbody>
</table>
Low power overview

• VLPW – Very Low Power Wait Mode

Wait mode is fully operational. You should expect IDDs to be about VLPR IDD minus 0.5 to 1 mA

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Trigger Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLPR</td>
<td>VLPW</td>
<td>Execute WAIT( );</td>
</tr>
<tr>
<td>VLPW</td>
<td>VLPR</td>
<td>Interrupt with LPWUI = 0</td>
</tr>
<tr>
<td>VLPW</td>
<td>Run</td>
<td>Interrupt with LPWUI = 1 or Reset</td>
</tr>
</tbody>
</table>
Low power overview

- VLPS – Very Low Power Stop

Expect IDD from 0.2 to 1 mA.

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Trigger Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLPR</td>
<td>VLPS</td>
<td>STOPM = 000 or 010, execute STOP( );</td>
</tr>
<tr>
<td>VLPS</td>
<td>VLPR</td>
<td>Interrupt with LPWUI = 0</td>
</tr>
<tr>
<td>Run</td>
<td>VLPS</td>
<td>AVLP=1, STOPM =010, execute STOP( );</td>
</tr>
<tr>
<td>VLPS</td>
<td>Run</td>
<td>Interrupt with LPWUI= 1 or Reset</td>
</tr>
</tbody>
</table>

Power mode transition operating behaviors for K20_50

<table>
<thead>
<tr>
<th>Description</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLPS → RUN</td>
<td>5.2 μs</td>
</tr>
</tbody>
</table>
Low power overview

• LLS – Low Leakage Stop

Expect IDD from 2 up to 200 µA

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Trigger Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>LLS</td>
<td>Set ALLS in PMPROT, PMCTRL_STOPM = 011, Execute STOP()</td>
</tr>
<tr>
<td>LLS</td>
<td>Run</td>
<td>Wakeup from enabled LLWU pin or module source or Reset pin</td>
</tr>
<tr>
<td>VLPR</td>
<td>LLS</td>
<td>Set ALLS in PMPROT, PMCTRL_STOPM = 011, Execute STOP()</td>
</tr>
</tbody>
</table>

**Power mode transition operating behaviors for K20_50**

<table>
<thead>
<tr>
<th>Description</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLS → RUN</td>
<td>6 µs</td>
</tr>
</tbody>
</table>
Low power overview

- VLLS0/1/2/3 – Very Low Leakage Stop

Expect IDD from 1.5 up to 10 µA

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Trigger Conditions</th>
</tr>
</thead>
</table>
| Run             | VLLS (3,2,1,0) | Set AVLLSx in PMPROT, PMCTRL_STOPM = 100  
VLLSM = 011 for VLLS3, 010 for VLLS2, 001 for VLLS1, 000 for VLLS0  
Execute STOP(); |
| VLLS (3,2,1,0)  | Run         | Wakeup from enabled LLWU input source or Reset. All wakeup goes through Reset sequence. Check SRS for source of wakeup. Check VLLSM for mode |
| VLPR            | VLLS (3,2,1,0) | Set AVLLSx in PMPROT, PMCTRL_STOPM = 100  
VLLSM = 011 for VLLS3, 010 =for VLLS2, 001 for VLLS1, 000 for VLLS0  
Execute STOP(); |
Low power overview

- VLLS0/1/2/3 – Very Low Leakage Stop (cont…)

<table>
<thead>
<tr>
<th>Description</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLLS0 → RUN</td>
<td>130 μs</td>
</tr>
<tr>
<td>VLLS1 → RUN</td>
<td>130 μs</td>
</tr>
<tr>
<td>VLLS2 → RUN</td>
<td>70  μs</td>
</tr>
<tr>
<td>VLLS3 → RUN</td>
<td>70  μs</td>
</tr>
</tbody>
</table>
Low power overview

It is important to keep in mind that there are restrictions for switch among all power modes.
MQX Low Power Management Features

Need Brief overview of topic to be discussed
Freescale also offers MQX™, the full-featured and complimentary Real-Time Operating System (RTOS). Starting with version 3.8, MQX integrates a Low-Power Management (LPM) driver to take advantage of the low-power operating modes in MQX applications.
MQX Low Power Manager

• The LPM is an MQX driver included with specific BSPs. It enables an application to easily change operation modes to take advantage of the MCU's low-power modes.

• The LPM is configured with operation modes of that BSP, which are mapped to the CPU power modes of the MCU.
MQX Low Power Manager

LPM operation modes of K20 BSP mapped to CPU power modes

![Diagram showing the mapping between LPM operation modes and Kinetis CPU power modes.]

- RUN mode
- WAIT mode
- SLEEP mode
- STOP mode
- VLPR mode
- VLPW mode
- VLPS mode
- LLS mode
LPM architecture

- The LPM also provides a mechanism for other drivers in the BSP to register with the LPM.
LPM driver

- The LPM driver is included in the MQX BSP to allow applications to easily change clock settings and low power modes.
- The application just calls `_lpm_set_operation_mode()` or `_lpm_set_clock_configuration()` to change the LPM state.
- The LPM also allows other drivers to register with the LPM and get notifications when the LPM changes the clock settings or low-power mode.
- Most of the source code for the LPM can be found in `<MQX Installation Directory>/mqx/source/io/lpm`. 
LPM driver

• The LPM requires a list of the available low-power modes of the CPU. The BSP maps the LPM operation modes to the CPU low-power modes, which is discussed further in the section “LPM Usage in BSP”.

• For the Kinetis BSP, the list of CPU power modes is defined in `lpm_mc.c`. Each power mode in this array is of type `LPM_CPU_POWER_MODE`, defined below in `lpm_kinetis.h`.

• The value in PMCTRL is written to the Kinetis register `MC_PMCTRL` which controls the power mode of the MCU. The FLAGS field is a group of bits used by the LPM driver when changing CPU power modes.
VLLSx – Wakeup Causes Reset?

- New MQX Feature in 4.1 – VLLSx support
- How does MQX handle VLLSx?
- What happens on wakeup?
- Is there a wakeup handler?
- Does MQX handle LLWU ISR?
- Does everything get re-initialized on reset? Why?
- When do you want to go into VLLSx? – when you go to sleep for a sufficient enough time that the power savings makes up for the longer wakeup time. Also, if you don’t have to wake up extremely fast.
Hands – On

- Running Low Power demo
- Enabling the Low Power demo for UART wake up
- Enabling the Low Power demo for SW2 wake up
Optimizing Applications for Low Power
To be Updated
Architecting State Transitions
# Example Application States

<table>
<thead>
<tr>
<th></th>
<th>POLL</th>
<th>CALCULATE</th>
<th>STANDBY</th>
<th>SHELF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>APPLICATION ACTIONS</strong></td>
<td>Collect serial data input, add to message queue</td>
<td>Get data from message queue, calculate result</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td><strong>FUNCTIONS NEEDED</strong></td>
<td>UART, CPU</td>
<td>CPU</td>
<td>UART INTERRUPT</td>
<td>RTC CPU/DSP</td>
</tr>
<tr>
<td><strong>FREESCALEMCU MODE</strong></td>
<td>Very Low Power Run (VLPR)</td>
<td>Run or Compute Mode</td>
<td>Low Leakage Stop (LLS)</td>
<td>Very Low Leakage Stop (VLLS0)</td>
</tr>
<tr>
<td><strong>FSL SYSTEM POWER</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>EXIT CONDITION</strong></td>
<td>QUEUE FULL, READY TO CALCULATE</td>
<td>CALCULATION COMPLETE</td>
<td>NEW DATA (FAST WAKEUP)</td>
<td>BUTTON PUSH (EXITS THROUGH RESET)</td>
</tr>
</tbody>
</table>
Example Application States

- CALC
- POLL
- STBY
- SHELF
Application States

• If no acceleration for 15 sec then go to first level of low power mode.
• After 30 sec of no movement, then go to deep low power mode.
• When you move it, it wakes up
• Use accelerometer interrupt as wakeup source!!!
Optimizing/Disabling the System Tick Timer

Need Brief overview of topic to be discussed
What is the RTOS tick?

- Periodic timer interrupt used to keep track of time
- Typically it is 5ms, 10ms, 100ms periods
- The priority of the interrupt is higher than tasks. Preempt tasks
- Operating system time unit
- Reflects the minimal time resolution
What is the RTOS tick?

- To allow for delays, timeouts, and measurements (MQX measures time in ticks, instead of in seconds and milliseconds)
- MQX keeps the time internally as a 64-bit count of the number of tick interrupts
- Features depending on the Tick
  - Time delay
  - Timeout while waiting for events, semaphores, and so on
  - Software timers
  - Time-slice scheduling
MQX System Tick

• What impact does the tick have on power consumption?
  – Run Modes: Minimal
  – Low Power Modes: Could wake up the system which has benefits and drawbacks – including the “tick wakeup” issue.

• What impact does the tick have on system efficiency?
  – Minimal, but it does preempt tasks and uses CPU cycles to update the time
SysTick Timer – Default MQX Tick

- ARMv7-M SysTick Timer – default MQX RTOS tick
- SysTick provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter
- Used for RTOS tick timer which fires at a programmable rate and invokes a SysTick routine.
- On Kinetis, it is clocked by the Cortex M4 FCLK (also clocks NVIC). Same frequency as system clock.
- Limited adjustability – cannot slow it down very much

<table>
<thead>
<tr>
<th>Modes</th>
<th>SysTick Running?</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN, VLPR</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>WAIT, VLPW</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>STOP, VLPS</td>
<td>No *</td>
<td></td>
</tr>
<tr>
<td>LLS</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>VLLSx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

* Unless debugger attached. If debugger attached it will continue to run.
SysTick Timer – Default MQX Tick

- **What affect does slowing down the tick have?**
  - Less granular time delay, timeouts, and software timers.

- **Can the tick be shut off completely?**
  - `SYST_CSR &= ~SysTick_CSR_TICKINT_MASK; /*disable systick interrupt*/`
  - `#define MQX_USE_TICK 0 /*this will not disable the tick, it will remove the time API functions.*/`
  - If there is no tick, then you can’t use the time delay, timeouts, and software timers. You have to use hardware timer interrupts or other interrupts to synchronize tasks.

- **Where is the tick service routine?**
  - `Init_bsp.c`
  - `void _bsp_systimer_callback(pointer dummy)`
SysTick Timer – Default MQX Tick

- Can the SysTick timer wake up the system?
  - Yes.

- When will the SysTick wake up the system?
  - It WILL wake up the system from Wait modes (Wait & VLPW)
  - It WILL NOT wake up the system from Stop modes (Stop, VLPS, LLS, & VLLSx)

- What is required to wake up the system from Stop modes?
  - Stop & VLPS – Another Interrupt
  - LLS & VLLSx – Low Leakage Wakeup Unit (LLWU) source
Low Power Timer – Can also be used for Tick

- Low Power Timer, available in all low power modes
- Freescale Kinetis MCUs include a Low Power Timer.
- 16-bit time counter or pulse counter with compare
- Optional interrupt can generate asynchronous wakeup from any low-power mode
- Configurable clock source and 4-bit prescaler – Can clock slower than SysTick
- Can use as system tick or a wakeup timer

<table>
<thead>
<tr>
<th>Modes</th>
<th>LPT Running?</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN, VLPR</td>
<td>Yes</td>
</tr>
<tr>
<td>WAIT, VLPW</td>
<td>Yes</td>
</tr>
<tr>
<td>STOP, VLPS</td>
<td>Yes</td>
</tr>
<tr>
<td>LLS</td>
<td>Yes</td>
</tr>
<tr>
<td>VLLSx</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Low Power Timer (LPT) as the RTOS Tick

- Not currently built-in to MQX API, but coming.
- Lower power than SysTick
- LPT tick will wake you up from all low power modes
- You can set the period longer than SysTick
- No need for separate wakeup source if going into very low power modes.

Systick as the RTOS tick

- Default tick timer in MQX Kinetis BSPs
- Wait modes – you can use it to wake up
- Stop modes – You must wake from interrupt or wakeup source. It will not wake from time calculated by the system tick
Waking up for Tick Interrupt, going right back to Sleep

- Kinetis MCUs return into low power mode after servicing the tick interrupt. (If the *sleep on exit* bit is set)

Keeping the Tick Running

- Benefit: Keeps the time in sync.
- Drawback: Might cause unnecessary wakeups to update the time. The sleep on exit feature, helps but does not solve this problem.
**Tick Wakeup Problem**

Entering Wait modes with SysTick as tick timer or any modes with LPT as tick timer

---

Tried to go into low power mode for 100 ms, but had to wake up every 5 ms to service the tick
Tick Wakeup Problem Solutions

• Easiest solution
  - Slow the tick down (extend tick period)
    - Longer *tick-less* period when you are in low power mode
    - Can still use ticks for scheduling tasks while using low power modes
    - Time granularity reduced for tick-dependent features (Note: not interrupt servicing or context switch time)

• Other solution
  - Disable the tick before entering low power, re-enable on exit - or - use a mode/timer combo that does this automatically.
    - Unlimited *Tick-less* period
    - Requires waking up from an interrupt / wakeup source
    - Time gets out of sync / Can’t rely on ticks for task timing.
Affect of Slower Ticks – Reduced Granularity

- Example – Task A tries to wait for 10ms.

```c
_time_delay(10ms) /* block task for 10 ms */
```

**Case 1 – 5ms tick**

```
<table>
<thead>
<tr>
<th>Time</th>
<th>Task A</th>
<th>Task B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Blocked for ~10ms!</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Case 2 – 15ms tick**

```
<table>
<thead>
<tr>
<th>Time</th>
<th>Task A</th>
<th>Task B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Blocked for ~15ms...</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Note: Tick accuracy is 0-1 tick
When do context switches happen?

- Not just on system ticks.
- **MQX RTOS is Event Driven.** Example: event gets set – immediately triggers scheduler. Does not need to wait for system tick

- Scheduling occurs if any one of the following occurs:
  - the active task calls yields control
  - the active task calls a blocking MQX function
  - an interrupt occurs
  - a timeslice expires
  - a higher-priority task becomes ready
Simple Task Scheduling – using Ticks

For an application with 2 tasks (Task 1 & Task 2)

- Task 1 is higher priority, it runs.
- Task 1 calls \_time\_delay(100)
- This blocks Task 1 and lets Task 2 run
- The system tick interrupt fires every 5 ms to keep track of the time.
- When the 100 ms (20 ticks) is up, then Task 1 takes control again.
Alternative way to schedule tasks – Event Driven, not using Ticks

For an application with 2 tasks (Task 1 & Task 2)
• Task 1 is higher priority, it runs.
• Task 1 calls <hardware timer> to interrupt in 100 ms
• Task 1 waits for event A
• This blocks Task 1 and lets Task 2 run
• In 100 ms, the hardware timer interrupt sets event A
• Task 1 takes control again.

• Benefits: Doesn’t rely on tick. More precise.
Typical Kinetis MQX Stop Mode *Tick-less*

SysTick as tick timer

With systick timer on Kinetis, ticks are automatically shut off in stop mode.

5ms

Tried to go into low power mode for 25 ms

Stayed asleep until it was time to get up!

Note: Must wakeup from an interrupt or wakeup source.
Staying awake after an interrupt

- MQX low power manager function call used within an ISR to keep the system from returning to sleep (wait or stop modes) on exit from an interrupt.
- Useful if an interrupt is used to wakeup from low power mode.

/* Do not return to sleep after isr again */

_lpm_wakeup_core ();

- This clears the sleep on exit bit

SCB_SCR &= (~ (SCB_SCR_SLEEPONEXIT_MASK));
Re-syncing MQX time with RTC

- When the tick is disabled, the time gets out of sync.
- Easily re-synced using the real time clock (RTC) or another clock source (network time, etc)
- Note: Kinetis RTC precision is +/-1 second. RTC interrupt can be used to sync the time the most precisely.
Hands-On Demo

• Entering Sleep mode (Kinetis Wait mode) - ticks continue to run
• Disabling the Ticks before entering sleep mode.
• Re-syncing the time after tick-less period
• Entering Stop mode (Kinetis LLS mode) – ticks stopped
• Re-syncing the time after tick-less period
Modifying the Tick Timer

Need Brief overview of topic to be discussed
Changing the TICK Time in MQX

• There are two ways to change the TICK timer.
  1. Changing the SysTick within the BSP
  2. Use the LPTMR as the TICK
Changing the SysTick within the BSP

- BSP timer is clocked by system tick.
- It is possible to edit BSP_ALARM_FREQUENCY in order to have smaller ticks.

Remember: we cannot guarantee the correct behavior if you modify original settings. Only change this value if you know what you are doing.
Changing the SysTick within the BSP

• By default it is defined to get 200 ticks per second. In order to change this value it is necessary to go to the BSP project and change the value of the next macro:

```c
#ifndef BSP_ALARM_FREQUENCY
#define BSP_ALARM_FREQUENCY (200)
#endif
```

• This macro sets the number of ticks per second.
• This macro can be found in `<Board_Name>.h` file.

`<MQX Installation Directory>/mqx/source/bsp/<board_name>`
Changing the SysTick within the BSP

- Delays are calculated in the following way:

\[
N_{TICKS} = \frac{REQ_{ms} \times ticks_{per\_s}}{1000\ ms}
\]

\begin{align*}
N_{TICKS} & \quad = \text{Number of ticks} \\
REQ_{ms} & \quad = \text{Number of milliseconds requested} \\
ticks_{per\_s} & \quad = \text{Number of ticks per second}
\end{align*}
Changing the SysTick within the BSP

• Example:

If `BSP_ALARM_FREQUENCY` is set as default \((100)\) and you request a 55ms delay you will have:

\[
N_{\text{TICKS}} = \frac{55 \text{ ms} \times 100}{1000 \text{ ms}} \frac{\text{ticks}}{\text{ticks ms}} = 5.5 \text{ ticks}
\]

This value will be truncated to 5 ticks. If it is requested any delay from 0 to 9 ms then it is necessary to have 0 ticks. But you will have some latency delay.
Using the LPTMR as the TICK

• Three steps:
  1. At the beginning of the task used for initialization call this function:
     \_\_lpt\_install\_kernel();
     - With these parameters:
       • Timer to initialize
       • Period in microseconds
       • Input clock sources mask
       • Interrupt priority
       • Unmask the timer after installation
     - i.e.
     \_\_lpt\_install\_kernel(0,5000,LPT\_FLAG\_CLOCK\_SOURCE\_LPO,2,TRUE);
Using the LPTMR as the TICK

• Three steps:
  2. After disable the systick from the core:
     SYST_CSR &= ~SysTick_CSR_TICKINT_MASK;
Using the LPTMR as the TICK

• Three steps:
  3. Take a beer and a rest!!!
Hands – On

• Enabling the Low Power Timer as RTOS TICK
MQX Lite Power Management Features
MQX Lite Low Power Features

• MQX Lite V1.1.0 (CW 10.4)
  – Supports clock mode changes and sleeping while idle.
  – Processor Expert Low Power support

• MQX Lite V1.2.0 (CW 10.5)
Processor Expert Low Power Features

• Logical device drivers in conjunction with processor component implement low power features of a target microcontroller.

• Each LDD component defines two methods related to the low power capability
  – SetOperationMode()
  – GetDriverState().
Processor Expert

Cpu_SetOperationMode

Kinetis CPU power modes

RUN mode

WAIT mode

STOP mode

VLPR mode

VLPW mode

VLPS mode

LLS mode

LPM operation modes

RUN mode

WAIT mode

SLEEP mode

STOP mode
Usage of Low Power API in Logical Device Drivers

- In this example, DPM (Dynamic Power Manager) task may opt to care for a selected number of peripherals for graceful power mode change (for example, FEC, CAN) and rest of the peripheral drivers need not know the power mode change.
- When opted for informing a peripheral device driver, the DPM can build a semaphore object for low power acknowledgement from the device drivers. When all such acknowledgements arrive (i.e., Semaphore count equals zero) the processor can be placed into a wait/sleep power mode.
Changing the System Tick in MQX Lite

Go to ProcessorExpert components -> MQXLite -> SystemTimer1 -> Properties and change
1. Counter from SYST_CVR to LPTMR0_CNR
2. Counter direction from Down to Up
3. Regenerate code recompile
4. That’s all.
Hands-On Demo

- MQX Lite Example – Entering/Exiting Low Power
Key Take-Aways and References
References

- AN4503 Power Management for Kinetis and ColdFire+ MCU
- AN4447 Freescale MQX Low-Power Management
- AN4470 Using Low Power modes on Kinetis family
MQX Level 2 Support Packages NEW
Freescale’s Commercial Support Option for MQX RTOS

• Customer software complexity & support needs are growing
• MQX Level 2 offers higher priority support from MQX experts
• Targeted for customer’s MQX-based development project needs
  – Support for customer’s customizations for their own hardware
  – Support for older versions of MQX
• New Website & MQX Support User’s Guide
  – Clearly defined support including free and commercial levels
• New Easy Ordering Process - Orderable Today!
• Go to www.freescale.com/mqx/support for details
<table>
<thead>
<tr>
<th>Plan Overview</th>
<th>Level 1 Support</th>
<th>Level 2 Support</th>
<th>Professional Engineering Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Releases Available</td>
<td>Approx. once a Quarter</td>
<td>Approx. once a Quarter</td>
<td>Early¹</td>
</tr>
<tr>
<td>Access to MQX Online Community / Online Training</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Service Requests (SW running on FSL Eval HW only)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Report bugs for fix in quarterly releases</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>BSP customization support</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Private Support Portal</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Access to Early Beta Releases¹</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Hot Fixes ²</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Initial Response Time ³</td>
<td>-</td>
<td>2 business days</td>
<td>1 business day</td>
</tr>
<tr>
<td>Max Hours of Support Engineer’s Time ⁴</td>
<td>-</td>
<td>20 hours</td>
<td>100 hours</td>
</tr>
<tr>
<td>Web Conferencing to debug issue ⁵</td>
<td>-</td>
<td>4 hours</td>
<td>10 hours</td>
</tr>
<tr>
<td>Phone Support ⁵</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Hands-on Support of Customer-Provided Hardware (shipped to Freescale Support)</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Support Plan Term</td>
<td>-</td>
<td>6 Months</td>
<td>12 Months</td>
</tr>
<tr>
<td>Price</td>
<td>Free</td>
<td>$3,000</td>
<td>$12,000</td>
</tr>
<tr>
<td>Part Number</td>
<td>-</td>
<td>DL-MQXLVL2-S</td>
<td>DL-MQXLVL2-P</td>
</tr>
</tbody>
</table>

¹ Early access to beta releases weeks/months before quarterly releases are typically available
² Access to bug fixes on issues you report immediately when available
³ Amount of time to receive acknowledge of support request
⁴ Any time support engineer spends including time preparing bug fixes.
⁵ Hours counted in Support Engineers Time
## Freescale MQX Support

### Summary: What is free and what is paid?

<table>
<thead>
<tr>
<th>Free (Level 1)</th>
<th>Paid (Level 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best effort response time</td>
<td>Specified response time</td>
</tr>
<tr>
<td>Quarterly releases</td>
<td>Hot fixes, early access releases</td>
</tr>
<tr>
<td>Standard issue resolution priority</td>
<td>Increased issue resolution priority</td>
</tr>
<tr>
<td>Support on FSL evaluation boards</td>
<td>Support on customer boards</td>
</tr>
<tr>
<td>Evaluation support</td>
<td>Customization support</td>
</tr>
<tr>
<td>Community &amp; service requests</td>
<td>Private portal, phone, web conferencing</td>
</tr>
<tr>
<td>Technical support professionals</td>
<td>Senior developers</td>
</tr>
</tbody>
</table>
MQX Level 2 Support Key Takeaways

• For higher priority support with direct access to MQX experts – Level 2
• For customization support for their own hardware – Level 2
• Recommend customers buy before project start – before problems
• Support is now easier to order
• Nominal fee - not a hard-sell to savvy customers.
MQX Support Details

- www.freescale.com/mqx/support

- MQX Support User’s Guide

- Contact Email: mqxL2mgr@freescale.com
Become an MQX RTOS Pro!

- Download the Latest Releases
- Check out the Learn to use MQX page
- Watch the Online Training Videos
- See the 80+ Examples & Demos
- Read the 12+ Application Notes
- View the Getting Started Guides
- Attend more in-person Training Events
- Collaborate in the MQX Community
- Create your own demos! Start with an example.
Freescale MQX™ Software Solutions

www.freescale.com/mqx

Freescale MQX™ Software Solutions

Accelerate your design success with complimentary RTOS, TCP/IP and USB stacks provided by Freescale MQX Software Solutions. Available on Freescale processors for more than 15 years, Freescale MQX Software Solutions offer a straightforward API with a modular architecture, making it simple to fine tune custom applications and scalable to fit most requirements. The combination of our market-proven Freescale MQX Software Solutions and silicon portfolio provides a streamlined and powerful platform by creating a comprehensive source for hardware, software, tools and services needs.

Certifiable to Medical and Aerospace Standards

Even if your application does not require formal certification, the robustness of the MQX™ RTOS helps provide a rock-solid platform proven in thousands of time-critical, sophisticated applications.

Products

Freescale MQX Software Solutions

Components Included in Freescale MQX Software Solutions

- Freescale MQX RTOS
- Freescale MQX RTCS (Communication Suite)
- Freescale MQX File System (MFS)
- Freescale MQX USB Host/Device Stack

Additional MQX Components

- Freescale MQX Design and Development Tools
- Freescale MQX Add-on Software

By Products

- ColdFire Processors
- Kinetics Microcontrollers
- Power Architecture Controllers (mobileGT)

Download

Download the latest releases and patches for MQX and licensing for free. Previous versions also available.

Featured Videos

- Freescale Tower - MCF5225X-KIT Overview (Video - 7:35) TWR-MCF5225X-KIT Overview Outline of the MCF5225X 32-bit Connectivity MCU
- TWR-MCF51CN-KIT Overview (Video - 6:34) Get an overview of the TWR-MCF51CN-KIT.
- MCF5225x DemoKit Video Guide (Video - 5:27) Get started with MCF5225x and Freescale MQX Software Lab Tutorial Demo.

Enhanced Security Software for Freescale’s MQX NanoSSL™ and NanoSSL™ for Freescale MQX RTOS is purpose-built for resource-constrained, high-performance device environments. Its super-small and super-fast, and with easy implementation and customization it is an ideal choice for your next great design.

Learn about extended cryptography solutions

Download the new MQX 3.6.2 release now

This Software Includes

- Customer Application
- Stacks
- Applications
- Drivers
- APIs
- Libraries
- OS
- BSP
- HAL
- Hypervisor
- Processor

Training & Events

Live Training
- Designing with Freescale Rapid Prototyping Solutions: Seminars, Throughout 2013

On-Demand Training
- View All MQX Webinars
- Introduction to the Freescale Tower System and MQX™ RTOS

Events
- Freescale Technology Forum
- Introduction to the RTOS: 2 Day Tutorial Workshop

Featured Tool

- Tower System
  Freescale’s modular development
Making the World a Smarter Place.
Designing with Freescale

Tailored live, hands-on training in a city near you

2014 seminar topics include

- QorIQ product family update
- Kinetis K, L, E, V series MCU product training

freescale.com/DwF