CMX-RTX Freescale Kinetis Challenge Version for the K60N Processor

Getting Started Guide

The versions of the Freescale tools used for CMX-RTX were:
Codewarrior for MCU v10.1.
Freescale TWR-K60N512 board

Installation

Extract the .zip file into a temporary directory and create an empty directory where the workspace will be.

Run Codewarrior and select the directory just created as the workspace directory.

Right-click in the project window and select Import, General, Existing Projects into Workspace, and click Next. In the next window, for the Root directory select the temporary directory used for the .zip file, select the project, check “Copy projects into workspace” and click finish.

If “Remote System Missing” messages appear click yes to add the remote system to the workspace.

Please check http://www.cmx.com/Freescale_Kinetis_Challenge for updates to the CMX-RTX evaluation software.

Caution: We recommend installing the software in a root-level directory. The reason for this is that you may experience problems linking if the software is installed in too “deep” a directory. Some of the tools have a finite limit on the length of a directory/path specification, and if this limit is exceeded, then you will experience problems. This kind of issue has become much more common in recent years, with the advent of long file names.

What is installed:
Folders:

- Manual – We highly recommend reading the manual, ver53dmo.pdf.
- Example – contains example project for the K60N512.

Files:

- Cmxdemo.c – an example program using CMX-RTX evaluation version.
- Cmxsampa.c – SysTick handler for the sample project.
- Cxfuncs.h – header file for CMX-RTX evaluation version.
- license.txt – the software license

Please email CMX at the following email address to report bugs or problems with the CMX-RTX Evaluation Version: cmx@cmx.com

CMX-RTX Freescale Kinetis Challenge Version limitations

The CMX-RTX Evaluation Version has some limitations that the full CMX-RTX does not have.

- There is a 30 minute time limit before the application will lock up. The time limit is based on a 10 millisecond system timer interrupt interval. After the 30 minutes has expired the board may be reset for another 30 minutes of running time.
- The Freescale Kinetis Challenge version of the RTOS will count the number of times tasks are started or resumed. After 500,000 starts and resumes the application will lock up.
- The CMX-RTX Freescale Kinetis Challenge version is not intended to be used in conjunction with the CMX-MicroNet Freescale Kinetis Challenge version.
- No source code for the library or scheduler.
- Low power function and time slicing are disabled in the scheduler.
- No CMXBug, CMXTracker or CMX-UART support.
- Only function K_Task_Create_Stack may be used to create tasks. Function K_Task_Create must not be used.
- The only CMX functions that may be called from interrupts are K_OS_Tick_Update, K_Intrp_Event_Signal and K_Intrp_Semaphore_Post.
- The RTOS configuration is fixed with the following values:

Max tasks	5
Max resources	4
Max cyclic timers	2
Max messages	32
Max queues	1
Max mailboxes	4
Max semaphores	4
interrupt stack size	384
RTC scale	1
Interrupt Pipe size	20
CMX_RAM_INIT	1

Getting Started

➔ The projects and example files are set up for the TWR-K60N512 board with the K60N processor in little-endian mode. If you are using a different board then modifications may need to be made to the projects and startup code.

➔ Unless the task does not end, function K_Task_End must be called at the end of a task. If this is not done, serious undesirable consequences may and will happen.

➔ Do not change any of the header files in the main directory. The cmxlib evaluation library has been built using those files so changing any of them would cause a conflict between the library code and the application code.

To debug the project, select project, build all. When building is completed open the Debug Perspective, click the debug icon, select Debug Configurations…, Codewarrior Download, example_MK60N512VMD100_Internal_RAM_PnE U-Multilink and click Debug.

The example application

The example application shows how to set up CMX-RTX, call various CMX-RTX functions from tasks and how to call certain CMX-RTX functions from an interrupt.

Function K_OS_Init must be called before calling any other CMX function. This initializes the CMX variables and prepares the OS for use.

Tasks may be created and started from other tasks but at least one task must be created and triggered before the OS is started. Function K_Task_Create_Stack must be used to create tasks. On processors such as the Kinetis, which have a stack that grows down, the task pointer parameter to K_Task_Create_Stack must point to the top of the stack buffer used. Task stacks must be 8-byte aligned. Setting task stacks improperly or using too small of a task stack will cause serious and potentially hard to find problems. Function K_Task_Start is used to start, or trigger, a task. See the CMX-RTX manual for details on these functions.

Before starting the OS, semaphores, queues, etc. may be created and cyclic timers started. This may also be done from tasks if desired. An interrupt that occurs on a regular interval and calls function K_OS_Tick_Update must be set up before calling K_OS_Start. The K60N port uses the SysTick interrupt for the timer tick interrupt. The sample ISR code also calls function K_Intrp_Semaphore_Post, that code may be removed for your own programs.

Function K_OS_Start starts the OS and does not return. The task with the highest priority that is able to start will be the first task that is run. In the example application that task is task1.

Interrupt Service Routines

To implement your own ISRs first add the handler function name to the appropriate spot in the vector table. See kinetis_sysinit.c and cmxsampa.c for an example ISR handler. Again, the only CMX functions that may be called from interrupts are K_OS_Tick_Update, K_Intrp_Event_Signal and K_Intrp_Semaphore_Post.