Freescale MQX Example Guide
core_mutex example

This document describes the core_mutex example application. The example allocates and creates a core mutex that is used for inter-core mutual exclusion to a shared data structure placed in the shared memory. This memory area is shared between individual cores of the multicore system like Freescale Vybrid MCU. The example shows how to work with the core_mutex driver and how to lock/unlock the core mutex to get an access to shared data.
The same code is executed on both cores, i.e. both cores are performing the same instructions. First, the application attempts to create the core mutex. The core mutex is created after the first attempt, it means the core that attempts to create the core mutex as the second just receives the pointer to the core mutex structure already created by the core that attempts to create the core mutex as a first. When calling the _core_mutex_create() function the core number, the mutex number and the task queueing policy is specified. Returned pointer to the core mutex is used each time the mutex lock/unlock function is called later.

Afterwards, shared data structure is cleared. The shared data structure is declared as follows:

```
typedef struct shared_data_struct {
    uint_32 count;
    uint_32 core_count[2];
} SHARED_DATA_STRUCT, * SHARED_DATA_STRUCT_PTR;
```

In fact, structure members are software counters. Each time a core locks the mutex it increments the “count” counter and the respective core counter “core_count[x]”. Then the core mutex is released, i.e. _core_mutex_unlock() function is called.

The state of all counters can be monitored by the serial console. The core with the core number 1 prints the state of all counters every 1000 iterations.

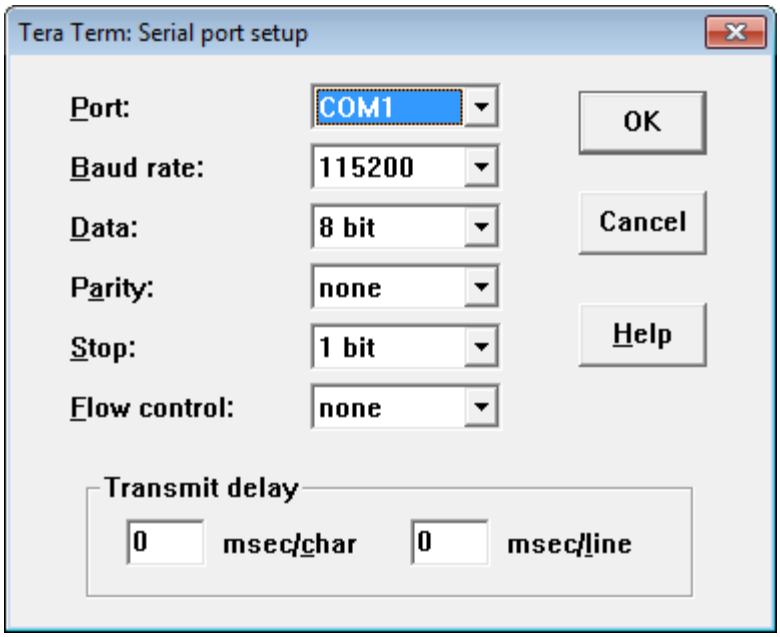
Running the example

Build the BSP and PSP projects for the target multicore platform. In case of Vybrid device build BSPs and PSPs for both A5 and M4 cores.

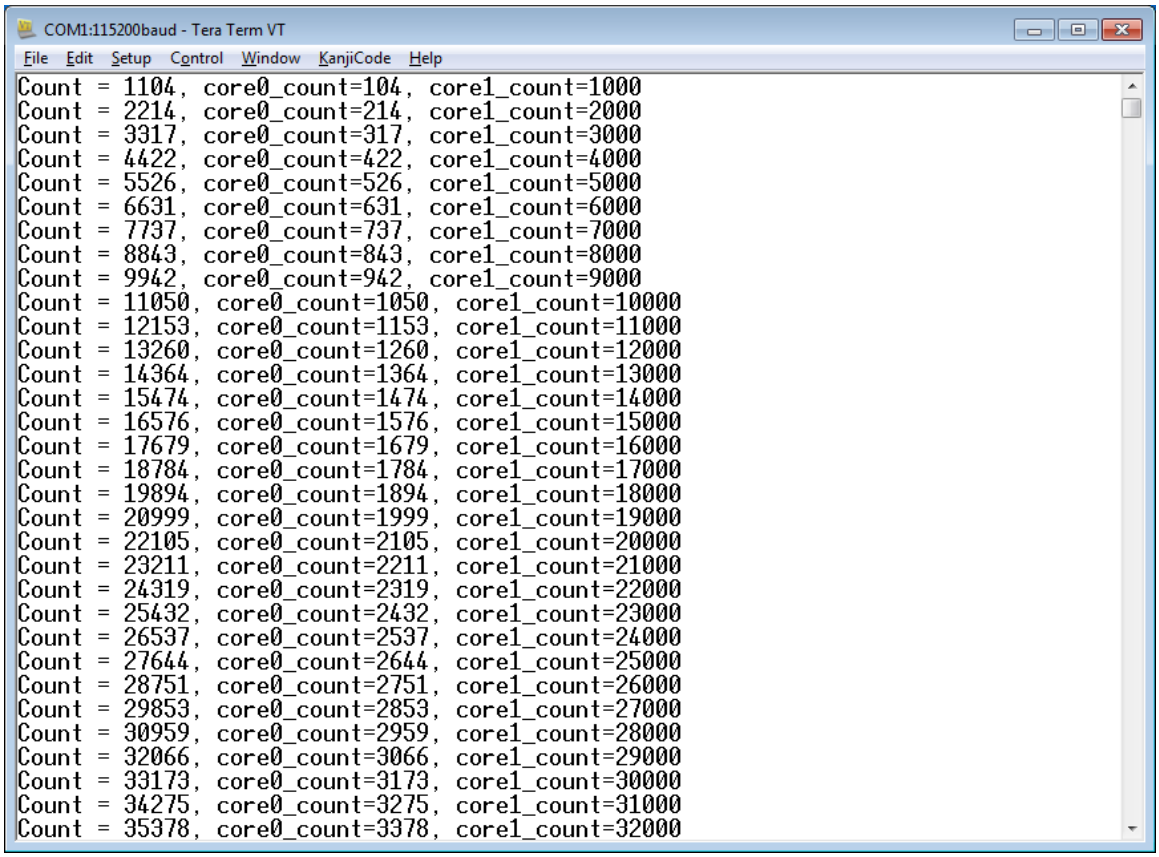
Refer to development tool – related MQX documentation for more information about multicore debugging (<MQX installation folder>\doc\tools\iar\MQX-IAR-Getting-Started.pdf).

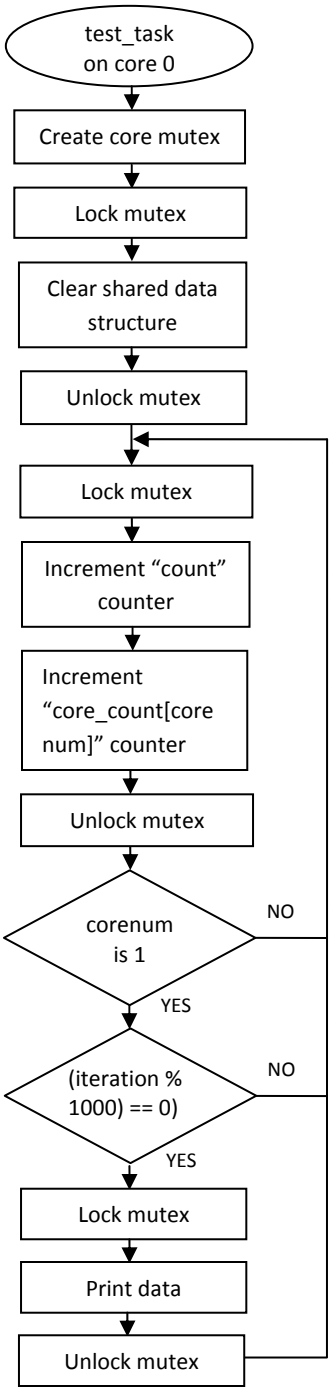
Refer to “Using Freescale MQX RTOS on Multi-core Devices” document for more information about the core_mutex driver (<MQX installation folder>\doc\mqx\MQX_using_multicore.pdf).

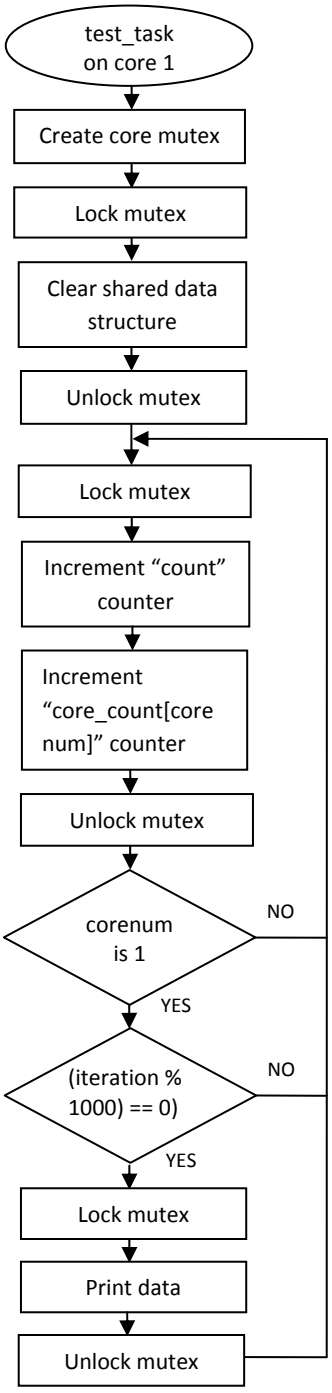
Start a terminal application on your PC and set the serial connection for 115200 baud, 8 data bits, 1 stop bit, no parity and no flow control. Connect your PC with the target platform via the serial cable.



Build and start core_mutex example applications on both cores of the target multicore platform. After starting both applications you will see the application log on the console.




test_task
on core 0
Create core mutex
Lock mutex
Clear shared data structure
Unlock mutex
Lock mutex
Increment “count” counter
Increment “core_count[core num]” counter
Unlock mutex
corenum is 1
NO
YES
(iteration % 1000) == 0)
NO
YES
Lock mutex
Print data
Unlock mutex


test_task
on core 1
Create core mutex
Lock mutex
Clear shared data structure
Unlock mutex
Lock mutex
Increment “count” counter
Increment “core_count[core num]” counter
Unlock mutex
corenum is 1
NO
YES
(iteration % 1000) == 0)
NO
YES
Lock mutex
Print data
Unlock mutex