This simple example shows MCC communication between two Endpoints (EP), each of them created on different core. Endpoints are receive buffer queues, implemented in shared SRAM, and are addressed by a triplet containing core, node, and port:

- Identifies the core within the processor. In case of Vybrid platform (twrvf65gs10), the A5 is core 0, and the M4 is core 1.
- In Linux any user process participating in MCC is a unique node. Node numbering is arbitrary. MQX has only one node and can also be an arbitrary number.
- All OSes can have an arbitrary number of ports per node (up to a configurable maximum), arbitrarily numbered with the exception of port 0 (MCC_RESERVED_PORT_NUMBER) being reserved.

In case of Vybrid platform (twrvf65gs10) the A5 core is the sender and the M4 core is the responder. The following EPs are defined on the application level:

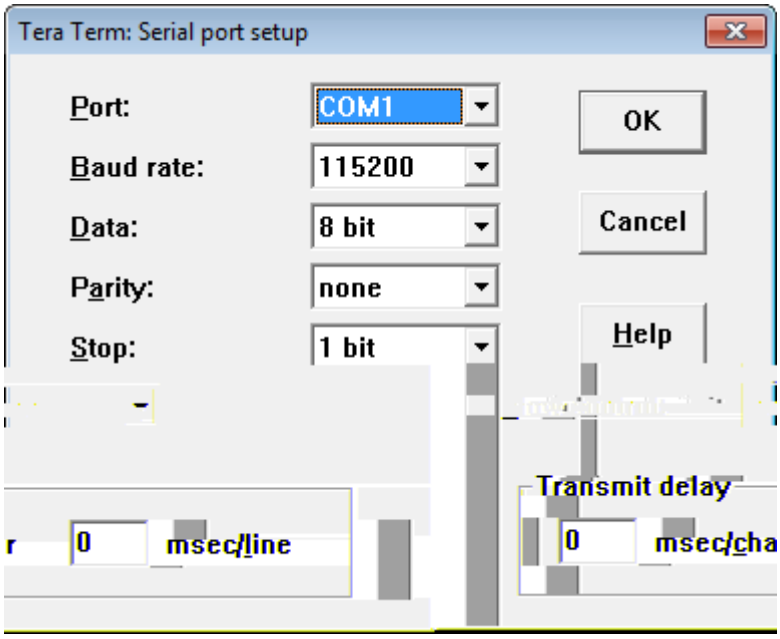
A5 core: mqx_endpoint_a5 = {0, MCC_MQX_NODE_A5, MCC_MQX_SENDER_PORT}
M4 core: mqx_endpoint_m4 = {1, MCC_MQX_NODE_M4, MCC_MQX_RESPONDER_PORT}

The application running on both core first initializes the MCC (if not already done by the other core), then it checks that both cores are using the same version of the MCC library and finally creates particular EPs on each core.

Once the receiver EP is created the sender initiates sending simple messages to the receiver EP. The message contains just a counter value, starting with the value of 1. Each time the receiver EP receives a new message, it increments the counter value and sends the message back to the sender. This sequence repeats on each EP/Core and the “message pingpong” continues forever, unless a receive error occurs.

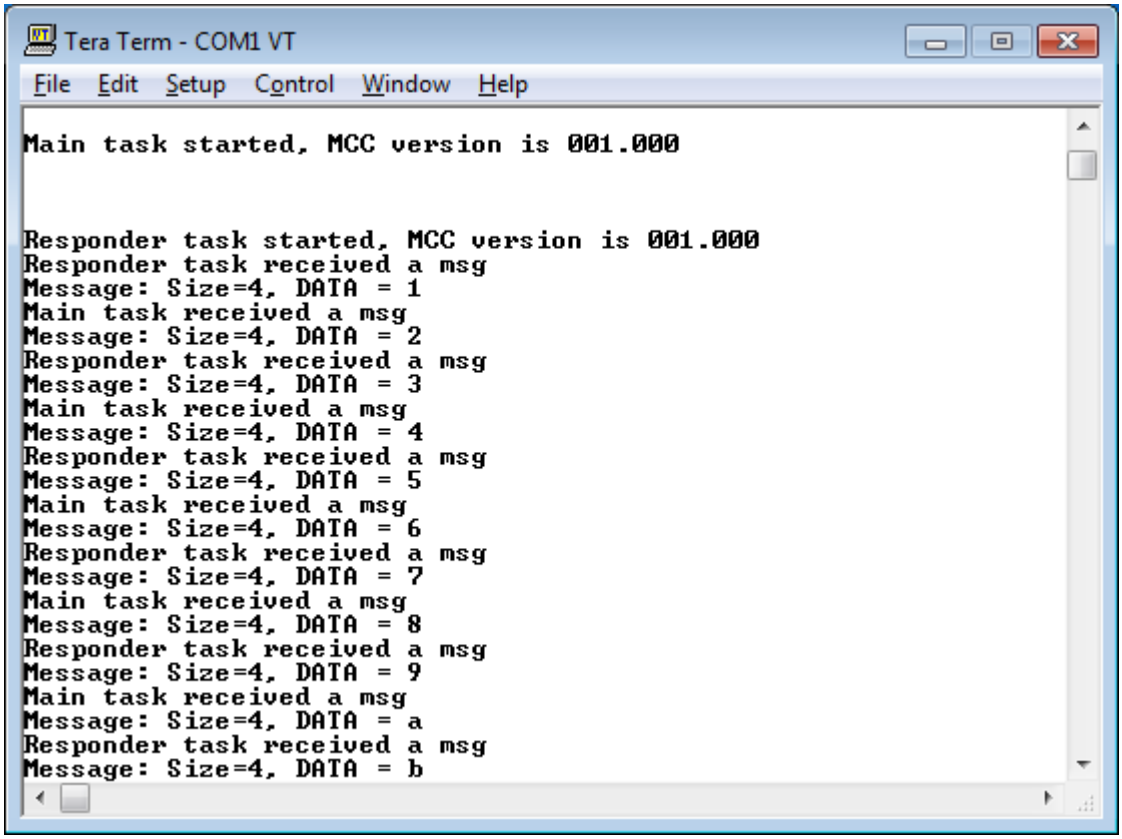
You should see outputs of both applications on the console (default console for each core).

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.



Start pingpong example on both cores (the order is not strict). For instructions how to do that in different IDEs, see the MQX documentation (<MQX installation folder>/doc/tools).

After starting both applications, you will see the printed message as the following picture.



The flow of the tasks is described in the next figure. There is the main task that runs on one core (A5) and the responder task running on the other core (M4). These tasks are exchanging messages (pingpong) between EPs created on both sides/cores.