Freescale MQX Example Guide

The msg_lite example

This client/server model shows communication and task synchronization using message passing.

Server task blocks waiting for a request message from Client task. When Server receives the request, it executes the request and returns the message to Client. Two-way message exchange is used, in order to block Client while Server runs.

Server opens an input message queue that it will use to receive requests from Client tasks and creates a message pool, from which it allocates request messages. Server then creates a number of Client tasks. In a real application, the Client tasks most likely would not be created by Server.

When Server has opened its message queue and created its message pool, it enters a loop, receiving messages from the message queue, acting on them (in this case, printing the data), and returning the message to Client.

Client also opens a message queue. It allocates a message from the message pool, fills in the message field, sends the message to Server, and waits for a response from Server.

Running the example

The user only needs to do compilation of MQX libraries, ksdk library and the example without any further step.

In <MQX_folder>\rtos\mqx\config\mcu\<board>\mqx_sdk_config.h please set

```
#define MQX_USE_MESSAGES                1
#define MQXCFG_ALLOCATOR                MQX_ALLOCATOR_LWMEM
```

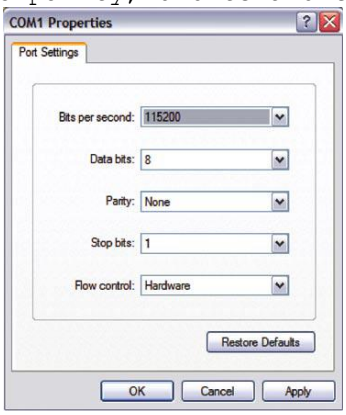
If the platform supports floating point, you have to disable floating point:

```
#define MQXCFG_ENABLE_FP                0
```

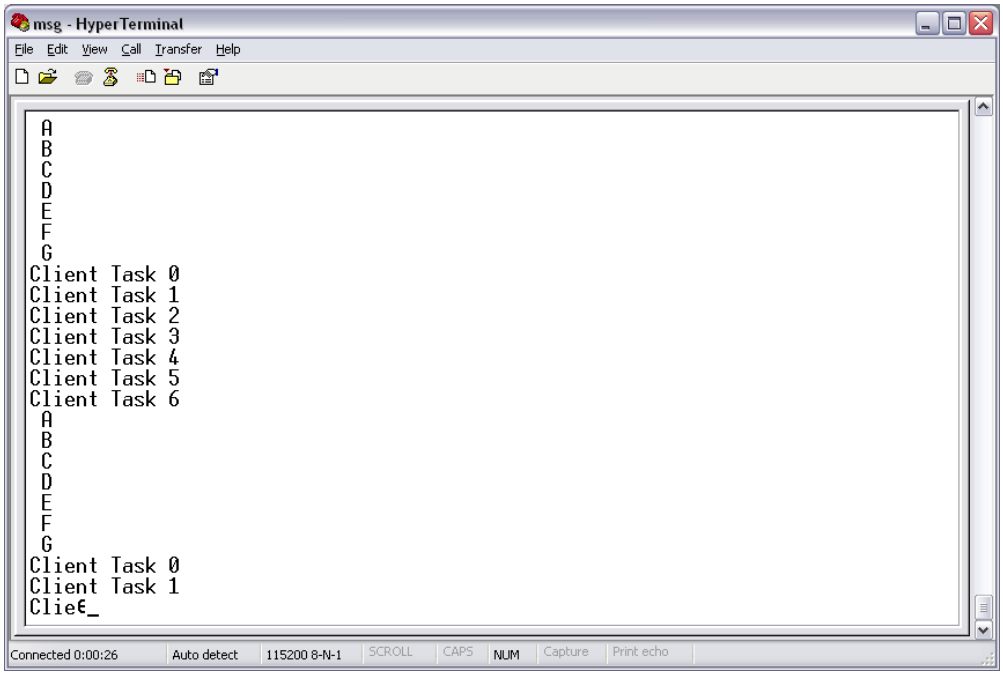
And rebuild MQX library.

Start HyperTerminal on the PC (Start menu->Programs->Accessories->Communications).
Make a connection to the serial port that is connected to the board (usually will be COM1).

Set it for 115200 baud, no parity, 8 bits and click OK.



After running the MCU, you will see the printed message as the following picture.



Explanation of the example

The flow of the tasks is described in the next figure. There is a server task that receives all the incoming data and responds to them.

There are seven client tasks. Each client sends a message to the server and receives the answer back from it.