# Freescale MQX RTOS Example Guide
# Security Telnet

This document explains the Security Telnet demo, what to expect from the example and a brief introduction to the API.

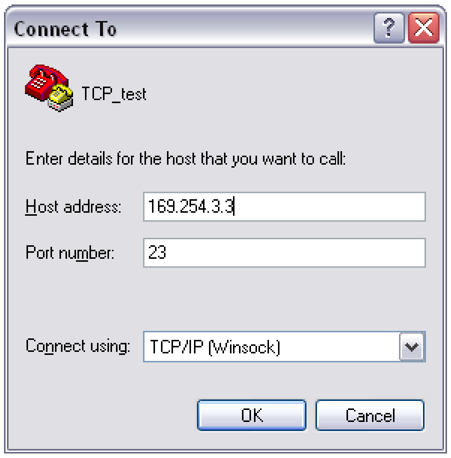
## The Example

A simple Security system has been implemented as the example application to demonstrate the features of the MQX RTOS. It detects button presses and movement of the tower system, keeps a log of the events and provides with a user interface over telnet protocol.

## Running the example

A telnet connection is required, therefore open a HyperTerminal on the PC, usually will be found at Start menu->Programs->Accessories->Communications.

Write TELNET_SESSION as the connection name, in "Connecting using" parameter select "TCP/IP (Winsock)" option. Enter IP address of the board into "Host address", in this case 169.254.3.3 and set "Port number" to 23.



TCP_test

Enter details for the host that you want to call:

Host address: 169.254.3.3

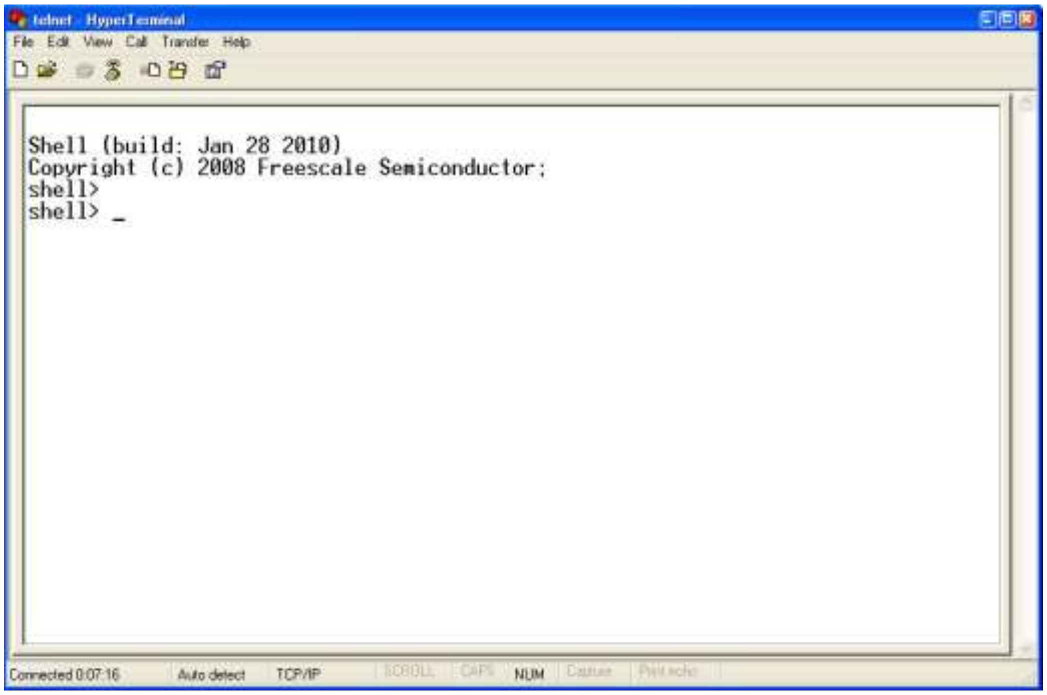
Port number: 23

Connect using: TCP/IP (Winsock)

Connect an Ethernet cable from the board to your PC. Your PC IP address should be on the same subnet. Typically, when you connect your computer directly to the board, the computer will default to an auto IP address on the same subnet as the board (169.254.x.x), therefore requiring no setup.

Note: It may take several minutes before PC gives up attempts to obtain an IP address and sets the default one. If you have trouble connecting, you may configure the IP address of the computer manually. Select Start > Settings > Network Connections > Local Area Connection. Note your original TCP/IP settings, and then set your IP address to 169.254.3.4 and your subnet mask to 255.255.0.0.

After verifying the PC and the board are on the same subnet, click on "Ok" button and the telnet session will be established. The HyperTerminal should print the message shown below:



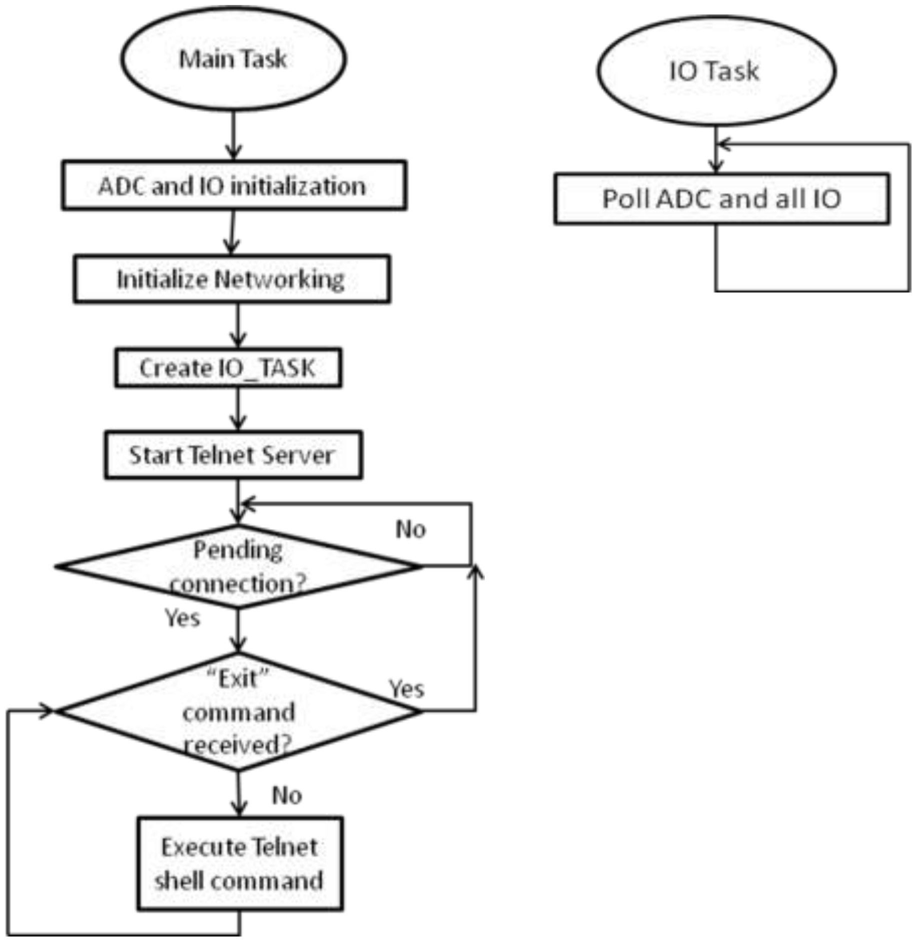
```
Shell (build: Jan 28 2010)
Copyright (c) 2008 Freescale Semiconductor;
shell>
shell> _
```

After the session is running, you can type "help" for the list of available commands. LED4 represents the door (SW2) or window (SW3) status as OPEN or CLOSED. Use the "status" command to see the current status of the board. It will display the current state of the switches, potentiometer and accelerometers, as well as the elapsed time since bootup.

Hold SW2 down, representing the door, with one hand, while typing "status" with the other and hit the Enter key. You will see that the door is Open. Type "displaylog" to see the last 10 events that the tower system has detected. "Clearlog" will clear the history.

## Explanation of the example

The demo consists of 2 tasks: “main task” and “IO task”:



The “Main task” first initializes the ADC and IO that will be used for the door, window and indication LED.

For setting the ADC function first the ADC device is open:

```
fd_adc = fopen(MY_ADC, (const char*)&adc_init);
```

Then the ADC channels to be used are open with proper parameters stored in the adc_ch_param[] array:

```
fd_ch[i] = fopen(dev_name, (const char*)&adc_ch_param[i]);
```

Inputs to simulate Door and Window are initialized through lwgpio driver this way (example for the door switch):

```
input_port = lwgpio_init(&button1, DOOR_STATE, LWGPIO_DIR_INPUT,
LWGPIO_VALUE_NOCHANGE);
if(!input_port)
{
    printf("Initializing LW GPIO for button1 as input failed.\n");
    _task_block();
}
lwgpio_set_functionality(&button1 ,BSP_BUTTON1_MUX_GPIO);
lwgpio_set_attribute(&button1, LWGPIO_ATTR_PULL_UP,
LWGPIO_AVAL_ENABLE);
```

The LED outputs are initialized in a similar fashion (example for LED1):

```
output_port = lwgpio_init(&led1, LED_1, LWGPIO_DIR_OUTPUT,
LWGPIO_VALUE_NOCHANGE);
if(!output_port){
    printf("Initializing LWGPIO for LED1 failed.\n");
}
lwgpio_set_functionality(&led1, BSP_LED1_MUX_GPIO);
/*Turn off Led */
lwgpio_set_value(&led1, LWGPIO_VALUE_LOW);
```

After the I/Os are initialized the network initialization takes place and “IO_TASK” for all the I/O management is created.

Then telnet server listens for a connection on port 23:

```
error = listen(listensock, 0);
```

This function is blocking, this means that the “Main task” will be on blocking state until a connection is detected. Once a connection is detected, the server will accept it and open a new socket with the following lines:

```
sock= accept(listensock, NULL, NULL);

sockfd = fopen("socket:", (char_ptr)sock);
```

Now with the connection established, the application will execute the Shell commands over Telnet until an “exit” command is received or the connection on the socket is lost. The shell is called by the function “Telnetd_shell_fn”.

The “IO_Task” will poll all the ADC channels to detect any movement (X,Y,Z) and the water level(just Y), the I/Os for any change of the door or window status and it will keep the log of the changes for being displayed.