Freescale MQX RTOS Example Guide

MFS_USB example

This document explains the mfs_usb example, what to expect from the example and a brief introduction to the API used.

The example

The example illustrates the usage of USB host, MFS and Shell API to deal with the USB memory stick. The application example allows user to perform a set of commands with USB memory stick through the terminal output. This including creating and deleting files, writing to and read reading from files as well as manipulation of directory.

Running the example

User needs to build usbh library and mfs library in addition to building basic MQX libraries – bsp and psp.
To run the example the corresponding IDE, compiler, debugger and a terminal program are needed.

Explaining the example

The example creates two tasks, namely Shell_Task and USB_task. Below is the description of these two tasks.

- Shell_Task:
  - This task allow user to interact with USB memory stick via the terminal output for example “cd” to change directory, “format” to format the whole device, “read” to read a file in USB memory stick and so on.
  - This task invokes function Shell() from Shell API with one parameter is an array of commands, namely Shell_commands. The Shell_commands array is an array of structures with each element including the string to be the command and the corresponding function to be invoked when the command string is typed into the terminal window from user.
  - Those functions implementing commands are from the Shell API.
- USB_task:
  - This task firstly creates a lightweight semaphore named *USB_Stick* which is used to signalize other tasks about the status of USB memory stick including insertion, removal.
  - USB_task then initializes a lightweight message queue called *usb_taskq* which is also used for synchronizing tasks and triggering specific action associated with events when USB memory stick is inserted or removed.
  - Install the *_int_unexpected_isr()* to be the interrupt handler for all interrupts that do not have the application-installed ISR.
  - Install the USB host driver into the IO manager of kernel using function *_usb_host_driver_install()* with the default host controller interface define in the twr-xxx.h file.

- Initialize the USB module (full speed or high speed module depends on the setting of macro BSP_USB_TWR_SER2 in the file twr-xxx.h) of the MCU. This includes the IO configuration relating to clock configuration of USB module, installing the interrupt handler *_usb_khci_isr()* (in the USB API), creating one more task specific for ehci or khci interface of USB module, and enabling the USB module to run in host mode.
- Call functions *_usb_host_driver_info_register(),_usb_host_register_service()* to assign the information related to the host driver including the protocol type, USB class type, the event handler called *usb_host_mass_device_event()* into a structure of device's information. The event handler is used in the process of signalizing task about status of USB memory stick in the system when it is removed or inserted.
- This task then enters an endless loop where it is blocked and waits for any event related to USB inserting (insertion and change of USB interface) and removing.
    - In case of insertion of the USB memory stick the new USB interface is installed via function *_usb_hostdev_select_interface()*
    - If new USB interface is detected, function *usb_msd_install()* is invoked to displayed the characteristic of the USB device and to install partition manager and MFS file system handler over the USB memory.
    - When USB memory is removed, the partition handler and MFS handler are uninstalled before memory allocated for storing USB data is released.