Freescale Semiconductor Document Number: USBSPNPUG
User's Guide Rev. 1, 04/2015

Freescale USB Unified Stack Porting New Platform User's Guide

1 Read Me First

This document provides the detailed steps to port the USB unified Stack to a new platform. There are two main steps involved:

- Porting USB core
- Porting examples

Note

USB unified stack supports multiple operating systems. When porting to a new platform, use the MQX™ RTOS-specific code. If you need to add new code, make sure that it is applicable to MQX RTOS.

Contents

2 Porting USB core

2.1 Porting HAL(Hardware Abstract Layer)

HAL provides the software interface to access hardware registers. There are separate files for ehci and khci located in the `<usb_rootdir>/usb_core/hal`.

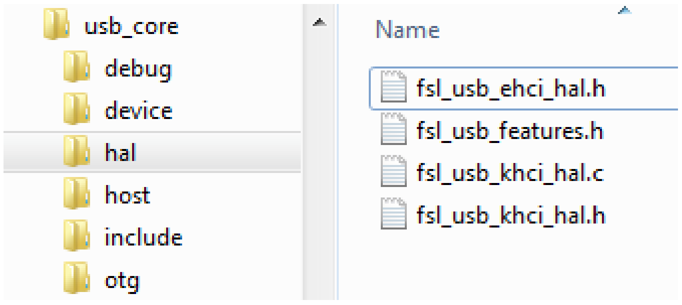


Figure 1 Porting HAL

- Port the fsl_usb_ehci_hal.h file or fsl_usb_khci_hal.h file or both files depending on the supported speed. Add the include path to the header file of new platform, as shown below:

  ```
  #elif (defined(CPU_MK70F12))
  #include "MK70F12.h"
  #define NEW_USB_HAL_ENABLE    0
  ```

  Define the CPU_MK70F12 macro in the project option to enable the newly ported HAL.
- Add a macro to define the MCU name of the new platform to the fsl_usb_features.h file, such as:
  defined (CPU_MK70FN1M0VMJ12)

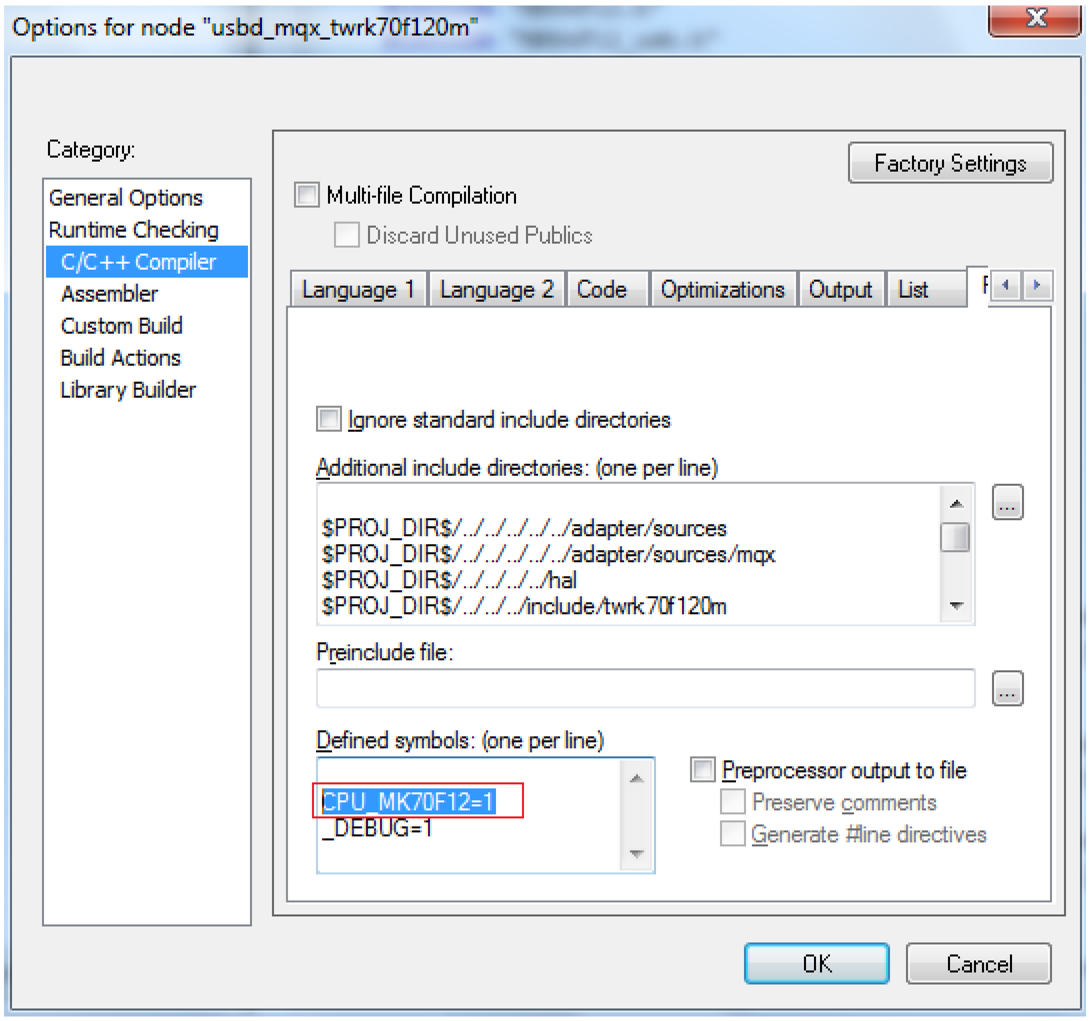
2.2 Porting USB libraries

2.2.1 Creating a library project

Create a new library project for the new platform. For example, the library project file for TWR-K70F120M IAR is located in:

- `<usb_rootdir>/usb_core/device/build/iar/usbd_mqx_twrk70f120m`
- `<usb_rootdir>/usb_core/host/build/iar/usbh_mqx_twrk70f120m`
- `<usb_rootdir>/usb_core/otg/build/iar/usbotg_mqx_twrk70f120m`

After creating a library project, add the CPU_MK70F12 macro to the project option to enable the newly ported HAL.



Additional include directories: (one per line)

```
$PROJ_DIR$/../../../../../adapter/sources
$PROJ_DIR$/../../../../../adapter/sources/mqx
$PROJ_DIR$/../../../../../hal
$PROJ_DIR$/../../../../include/twrk70f120m
```

Defined symbols: (one per line)

```
CPU_MK70F12=1
_DEBUG=1
```

Figure 2 Options for node

For the OTG library project, add more USBCFG_OTG macros to the project option.

2.2.2 OS adapter

The OS adapter includes the adapter files which allow the USB stack to run on different RTOS with the same USB core code. Search the `<usb_rootdir>/adapter/sources/mqx/adapter_mqx.c` and change this function if needed:

- `soc_get_usb_base_address`: return USB base address of USB controller
- `soc_get_usb_vector_number`: return interrupt vector of USB controller

2.2.3 USB Device library

- BSP code: Located in `<usb_rootdir>/usb_core/device/sources/bsp/{BOARD_NAME}/usb_dev_bsp.c`. This file contains these functions:

    - `bsp_usb_dev_soc_init()`: USB module initializes and generates 48 MHz clock for the USB module. See function `_bsp_usb_dev_io_init()` in the `<mqx_rootdir>/mqx/source/bsp/{BOARD_NAME}/init_gpio.c` to write to this function.
    - bsp_usb_dev_board_init(): It is a weak function and can be over-written by user examples. It performs board-specific initialization related to the USB module.
    - `bsp_usb_dev_init()`: Initializes the register in the USB module. See function `_bsp_usb_dev_init()` in the `<mqx_rootdir>/mqx/source/io/usb/usb_mk70f_device`.c to write to this function.

Note

For the platform which has an MPU module, add code to disable the MPU and allow access from all bus masters.

- Configuration file: Defines the configurations for the new platform located in the `<usb_rootdir>/usb_core/device/include/{BOARD_NAME}/usb_device_config.h`.

  See Chapter 6 in the *USB Stack Device Reference Manual* (document USBSDRM) for more details about macros.

- .bat file: Creates a bat file for the new platform `<usb_rootdir>/usb_core/device/build/bat/usbd_{BOARD_NAME}.bat` which is used to copy all USB public header files to the `<usb_rootdir>/output` folder.

2.2.4 USB Host library

- BSP code: Located in the `<usb_rootdir>/usb_core/host/sources/bsp/{BOARD_NAME}/usb_host_bsp.c`. This file contains these functions:
    - `bsp_usb_host_soc_init ()`: Initializes the USB module and generates the 48 MHz clock for the USB module. See function `_bsp_usb_host_io_init()` in the `<mqx_rootdir>/mqx/source/bsp/{BOARD_NAME}/init_gpio.c` to write this function.
    - bsp_usb_host_board_init(): It is a weak function and can be over-written by user examples. It performs board-specific initialization related to the USB Host module.
    - `bsp_usb_host_init()`: Initializes registers in the USB module. See function `_bsp_usb_host_init()` in the `<mqx_rootdir>/mqx/source/io/usb/usb_mk70f.c` to write to this function.

- Configuration file: Defines the configurations for new platform located in the `<usb_rootdir>/usb_core/host/include/{BOARD_NAME}/usb_host_config.h.`

  See Chapter 5 in the *USB Stack Host Reference Manual* (document USBSHRM for more information about macros.

- .bat file: Creates a bat file for the new platform - `<usb_rootdir>/usb_core/host/build/bat/usbh_{BOARD_NAME}.bat` which is used to copy all the USB public header files to the `<usb_rootdir>/output` folder.

2.2.5 USB OTG library

- BSP code: Port the `<usb_rootdir>/usb_core/otg/sources/bsp/{BOARD_NAME}/usb_otg_bsp.c`. This file contains:
  - Specific macros for MAX3353: The current OTG solution is using the MAX3353 to measure and control V_{Bus}, D+, D-. It can generate an interrupt when the status of ID pin changes. These are the macros that need to be defined to work with the MAX3353 via I2C:
    - MAX3353_INT_PORT: Interrupt port for MAX3353
    - MAX3353_INT_VECTOR: Interrupt vector for MAX3353
    - MAX3353_INT_PIN: Interrupt pin for MAX3353
    - MQX3353_I2C_CHANNEL: specific I2C channel to communicate with MAX3353
    - MAX3353_INT_VECTOR: I2C Interrupt vector
  - `bsp_usb_otg_soc_init ()`: Initializes the USB module and generates the 48 MHz clock for the USB module.
  - bsp_usb_otg_board_init(): It is a weak function and can be over-written by user examples. It performs board-specific initialization related to the USB OTG module.
  - `bsp_usb_otg_init()`: Initializes the register in the USB module.
  - `bsp_usb_otg_set_peripheral_init_param()`: Sets the initialization param of the peripheral. Call this function before calling the usb_otg_init when the pins are changed in the user's board. The function can only support MAX3353 and use the above macro as an initial value. If the new peripheral is used in the board, the function should be changed to handle a new case.
- .bat file: Create a bat file for the new platform - `<usb_rootdir>/usb_core/otg/build/bat/usbotg_{BOARD_NAME}.bat` which is used to copy all the USB public header files to the `<usb_rootdir>/output` folder.

3 Porting Examples

This section lists all available USB examples for MQX RTOS in the current USB Unified Stack and how to port them.

3.1 Creating an example project

List all the examples for each IDE that you want to port for new platform. The project file is kept in the following folder structure:

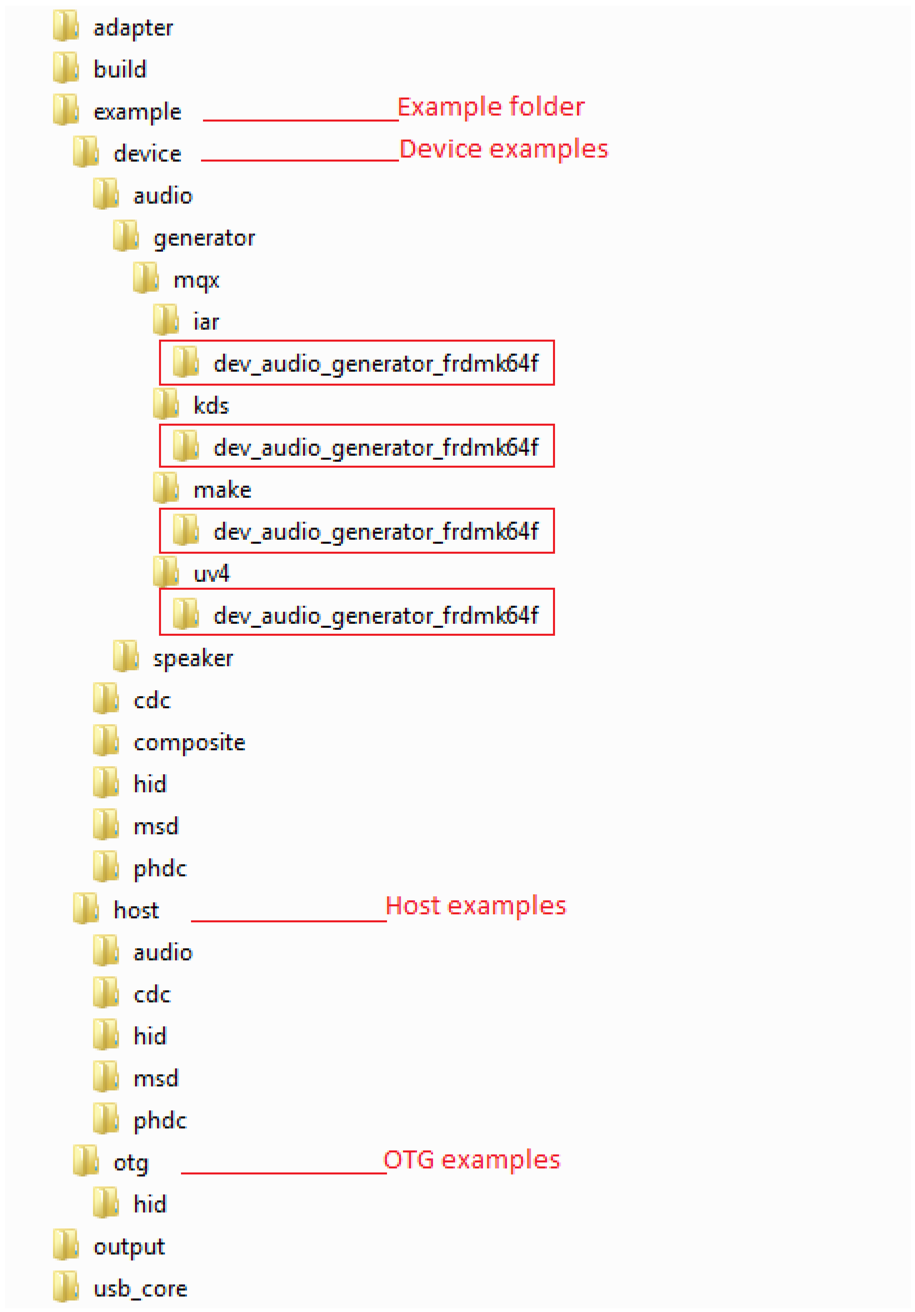
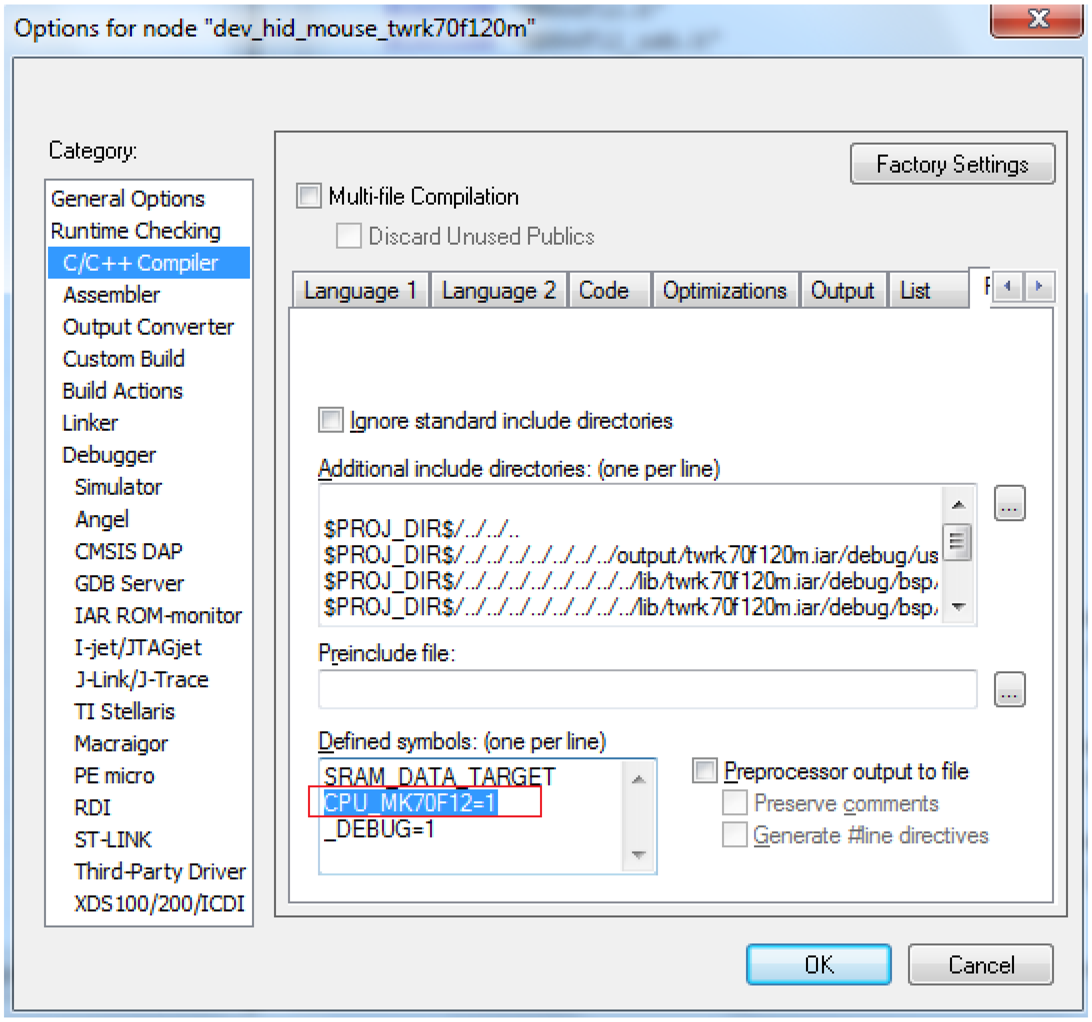


Figure 3 Example projects

The steps for creating a new project are shown below:

1. Create a new empty project.
2. Choose a device.
3. Add all the source code into project based on the existing platform.
4. Change the include path.

5. Change the debug configuration.
6. Add the above macro to enable the new HAL. For example, add the macro for the TWR-K70F120M Tower System module as shown in the figure below:


Options for node "dev_hid_mouse_twrk70f120m"

Category: General Options, Runtime Checking, C/C++ Compiler, Assembler, Output Converter, Custom Build, Build Actions, Linker, Debugger, Simulator, Angel, CMSIS DAP, GDB Server, IAR ROM-monitor, I-jet/JTAGjet, J-Link/J-Trace, TI Stellaris, Macraigor, PE micro, RDI, ST-LINK, Third-Party Driver, XDS100/200/ICDI

Factory Settings

☐ Multi-file Compilation
☐ Discard Unused Publics

Tabs: Language 1, Language 2, Code, Optimizations, Output, List

☐ Ignore standard include directories

Additional include directories: (one per line)

```
$PROJ_DIR$/../../..
$PROJ_DIR$/../../../../../../../output/twrk70f120m.iar/debug/us
$PROJ_DIR$/../../../../../../../../lib/twrk70f120m.iar/debug/bsp/
$PROJ_DIR$/../../../../../../../../lib/twrk70f120m.iar/debug/bsp/
```

Preinclude file:

Defined symbols: (one per line)

```
SRAM_DATA_TARGET
CPU_MK70F12=1
_DEBUG=1
```

☐ Preprocessor output to file
☐ Preserve comments
☐ Generate #line directives

OK Cancel

Figure 4 Options for node

3.2 Device examples

3.2.1 USB audio generator device

This application doesn’t contain any specifics for the platform.

3.2.2 USB audio speaker device

This application uses the SGTL5000 as codec for the audio output. The microcontroller controls it via I2C and sends audio data to it via I2S. This application is only available for platform that have I2C and I2S/SAI modules.

To run this application, enable I2C, I2S/SAI in the file `user_config.h`.

Specific I2C channel to communicate with SGTL5000 for new platform is defined in the `<usb_rootdir>/example/device/audio/speaker/mqx/sgtl5000.c`.

3.2.3 USB virtual com device

This application doesn't contain any specifics for platforms.

3.2.4 USB virtual nic device

This application is only available on the platforms which support the Ethernet module.

3.2.5 USB HID keyboard device

This application doesn't contain any specifics for platforms.

3.2.6 USB HID mouse device

This application doesn't contain any specifics for platforms.

3.2.7 USB mass storage device

- For the SD card mode (SD_CARD_APP macro in disk.h file is set to 1):

  Either the board needs to support the SD card feature and the BSP_SDCARD_ESDHC_CHANNEL or BSP_SDCARD_SDHC_CHANNEL or BSP_SDCARD_SPI_CHANNEL needs to be defined in file

  `<mqx_rootdir>/mqx/source/bsp/{BOARD_NAME}/{BOARD_NAME}.h` like: `<mqx_rootdir>/mqx/source/bsp/twrk70f120m/twrk70f120m.h`.

  Otherwise, the SD_CARD_APP is not supported for this board.

- For the RAM disk mode (SD_CARD_AP, the macro in the disk.h file is set to 0):

  The example requires the RAM size of more than 24K.

3.2.8 USB PHDC device

This application doesn't contain any specifics for platforms.

3.2.9 USB HID + Audio generator composite device

This application doesn't contain any specifics for platforms.

3.2.10 USB CDC + MSD composite device

- For the SD card mode (SD_CARD_APP macro in disk.h file is set to 1):

Either the board needs to support the SD card feature and BSP_SDCARD_ESDHC_CHANNEL or BSP_SDCARD_SDHC_CHANNEL or BSP_SDCARD_SPI_CHANNEL needs to be defined in the file

`<mqx_rootdir>/mqx/source/bsp/{BOARD_NAME}/{BOARD_NAME}.h` like:
`<mqx_rootdir>/mqx/source/bsp/twrk70f120m/twrk70f120m.h`.

Otherwise, the SD_CARD_APP is not supported for this board.

- For the RAM disk mode (SD_CARD_APP macro in the disk.h file is set to 0):

The example requires the RAM size of more than 24K.

3.3 USB Host examples

3.3.1 USB audio microphone host

This application uses the SD card to store recorded data from the USB generator device. It is only available for platforms that have SPI or ESDHC module to communicate with the SD card.

Check the file `<usb_rootdir>/example/host/audio/microphone/audio_microphone.c` and modify code to raise an error if the SPI or ESDHC channel is not enabled in the `user_config.h`.

3.3.2 USB audio speaker host

This application reads audio data from the SD card and sends it to the USB speaker device. It is only available for platforms that have SPI module to communicate with the SD card.

3.3.3 USB CDC host

This application doesn’t contain any specifics for platforms.

3.3.4 USB HID keyboard host

This application doesn’t contain any specifics for platforms.

3.3.5 USB HID mouse host

This application doesn’t contain any specifics for platforms.

3.3.6 USB HID keyboard + mouse host

This application doesn’t contain any specifics for platforms.

3.3.7 USB MSD command host

This application doesn’t contain any specifics for platforms.

3.3.8 USB MSD FATFS host

This application doesn’t contain any specifics for platforms.

3.3.9 USB PHDC host

This application doesn’t contain any specifics for platforms.

Note

To run this application correctly, the MQX_INCLUDE_FLOATING_POINT_IO must be set to one in the file `user_config.h`.

3.4 USB OTG examples

3.4.1 USB HID OTG

This application needs to communicate with the MAX3353 via I2C.

Check the file `<usb_rootdir>/example/otg/hid/mouse/otg_mouse.c` and modify the code to raise an error if the I2C channel used to communicate with the MAX3353 is not enabled in the `user_config.h`.

If the new I2C channel or the new GPIO is used in the board, the function `bsp_usb_otg_set_peripheral_init_param()` must be called before the `usb_otg_init` is called.

Document Number: USBSPNPUG
Rev. 1
04/2015