Overview

The USB MSC RAM disk application is a simple demonstration program based on the MCUXpresso SDK. It is enumerated as a U-disk and can be read and written to as a normal U-disk .

System Requirement

Hardware requirements

- Mini/micro USB cable
- USB A to micro AB cable
- Hardware (Tower module/base board, and so on) for a specific device
- Personal Computer (PC)

Software requirements

- The project files for lite version examples are located in:
  <MCUXpresso_SDK_Install>/boards/<board>/usb_examples/usb_device_msc_ramdisk_lite/<rtos>/<toolchain>.
  For non-lite version example, the path is:
  <MCUXpresso_SDK_Install>/boards/<board>/usb_examples/usb_device_msc_ramdisk/<rtos>/<toolchain>.
  Note
  The <rtos> is Bare Metal or FreeRTOS OS.

Getting Started

Hardware Settings

Note

Set the hardware jumpers (Tower system/base module) to default settings.

Prepare the example

1. Download the program to the target board.
2. Connect the target board to the external power source (the example is self-powered).
3. Power off the target board. And then power on again.
4. Connect a USB cable between the PC and the USB device port of the board.

Note

For detailed instructions, see the appropriate board User's Guide.

Run the example

1. Plug in the MSD disk device, which is running the usb_device_msc_ramdisk example into PC. A USB Mass Storage Device is enumerated in the Device Manager.
2. If you enable the RAM disk function, the windows prompts to format the u-disk.

Figure 1: Format the disk

When the format is completed, the computer will display the capacity of 4k removable disk.

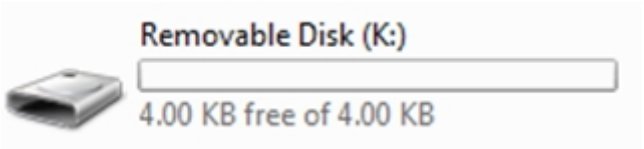

Figure 2: RAM u-disk

Note

- Mac® OS 10.9 default creates .fseventsd, .Trashes folder, and some other files the disk is formatted on Mac OS. The total file size is about 8 K. If the USB mass storage example is running on Mac, increase the RAM size to at least 32 K. Change the MACRO TOTAL_LOGICAL_ADDRESS_BLOCKS_NORMAL in disk.h from 48 to 64. If the Mac OS 10.11 EI Capitan is used to format the disk, the least ram size should be 2.1 MByte. Otherwise, Mac OS shows "not enough space for allocate" and can't format the disk.
- The throughput of the FreeRTOS example may be lower than the bare metal example when using the KHCI (full speed controller) on some types of PCs. The root cause of the issue is that OUT tokens are sent too quickly on some types of PCs so that the receiving buffer on the device side can't be prepared in time. The detail analysis is in the subsequent chapter. throughput of freertos example may be improved through adjusting the code optimizations of IAR. The setting method is shown in the below figure: In IAR, setting Option->C/C++ Compiler->Optimizations.

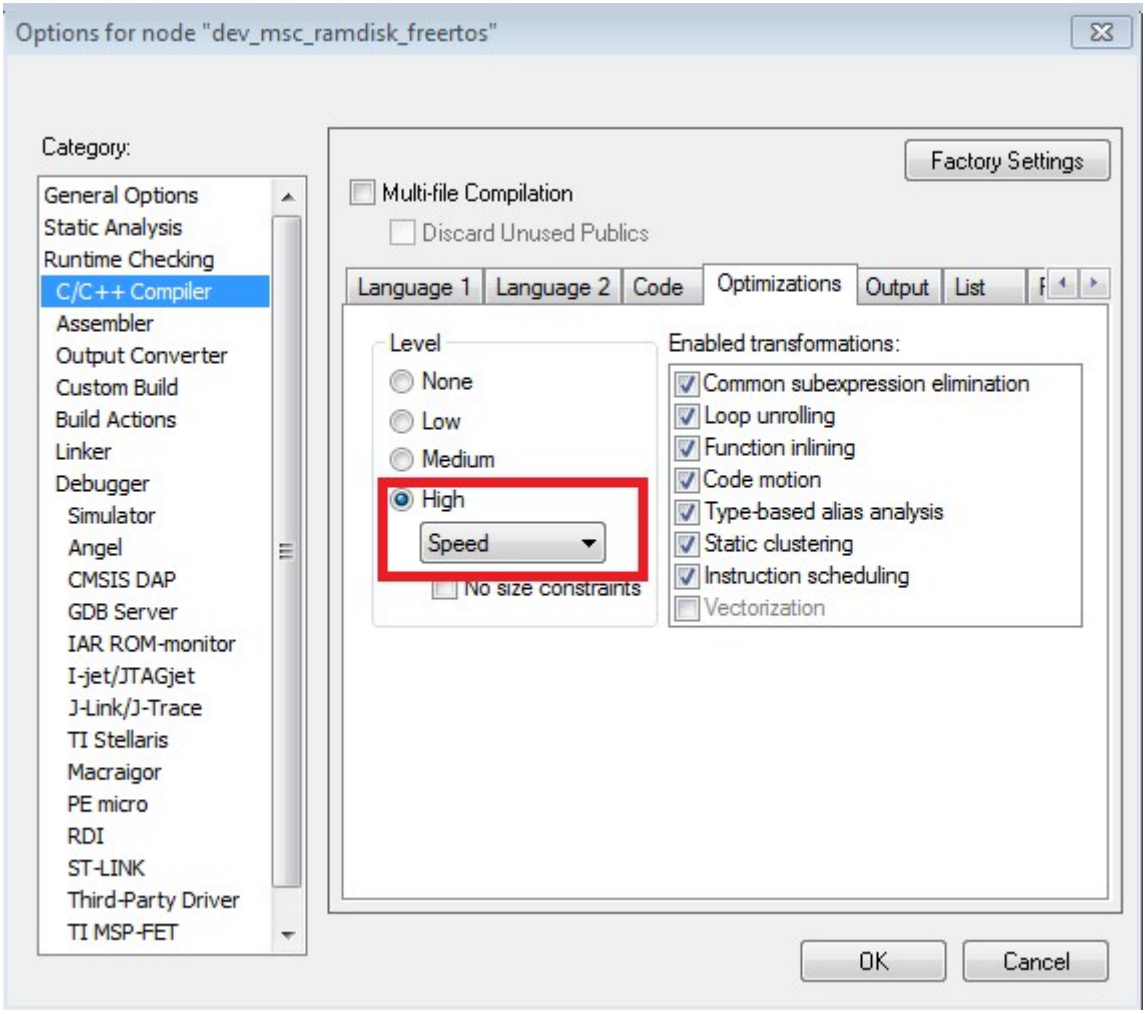

Figure 3: IAR setting

The reason that the device does not prepare the receiving buffer in time is explained as the following (the following measurement is done on the TWR-K65 module):

GPIO Pulling up means: the previous token done interrupt received

GPIO Pulling down means: the next data receiving buffer is primed.

1. In the bare metal example, the time between GPIO pulling up and down is about 3.2 us.

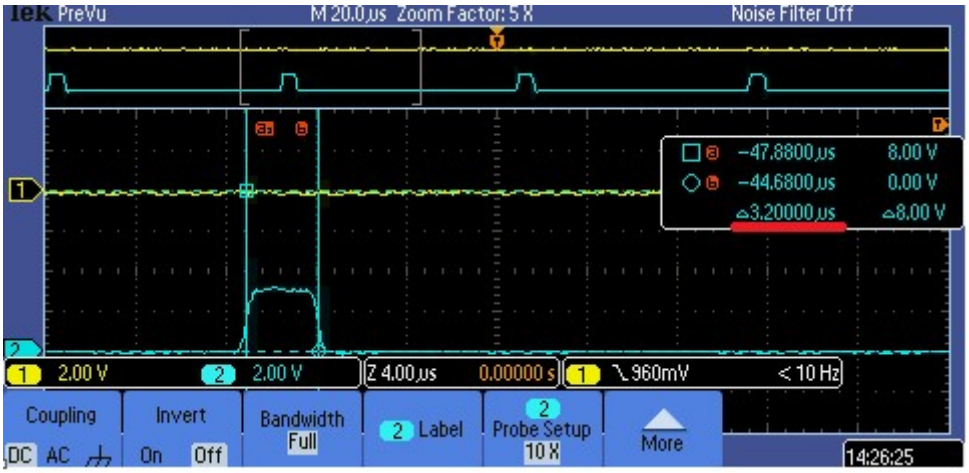

Figure 4: bm example timing

2. In the FreeRTOS example, it is about 4 us.

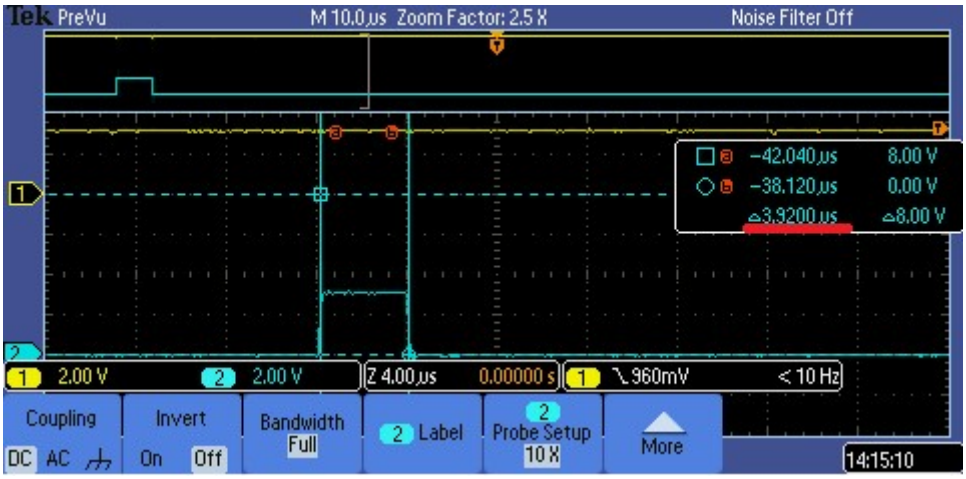

Figure 5: freertos example timing

It takes longer to prime the buffer in the FreeRTOS example, when the Host sends OUT a token too fast and the receiving data buffer might not be ready yet, which causes NAKs during the data transfer. After a NAK occurs, becauses there is no ping control in the USB 1.1, the host has to send out the data (64 bytes) packet before the NAK packet is received. On the full speed bus, this 64 bytes data packet consumes ∼45 us. The 45 us delay is the root cause that the throughput on the FreeRTOS drops.

For some hosts, the interval between the previous transaction and next is a little longer, for example 7 us, which is enough time for the device to prepare the next receiving buffer and no NAK occurs. As a result, the throughput is higher.

3. After adjusting the optimization, the time is about 3.6 us.

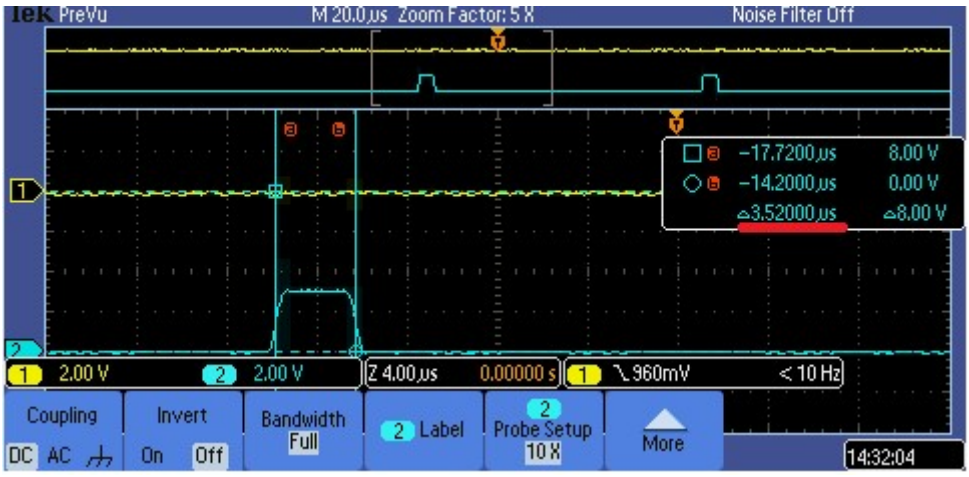

Figure 6: freertos example timing after optimization

Therefore, NAK number can be reduced during and the throughput can be increased during the transfer.