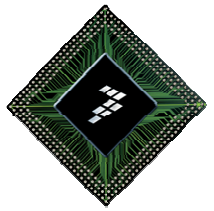

i.MX233 BSP Howto

freescale™
semiconductor

- The necessary packages for ltib
- How to boot and mount rootfs from NAND
- How to mount host's nfs as rootfs over usb0
- How to make USB Mass Storage

freescale™ semiconductor

The necessary packages for ltib

- ltib can work under Ubuntu 8.1 with following packages installed.
  - bison
  - g++
  - gettext
  - libbeecrypt6
  - libglib2.0-dev
  - libncurses5-dev
  - librpm4.4
  - m4
  - patch
  - rpm
  - tcl
  - tcl8.4
  - zlib1g-dev

How to boot and mount rootfs from NAND

- Prepare rootfs and linux kernel with gpmi built-in support
  - ./ltib -m config
    - You will see this menu at the very first time after ltib installed
      - Selection (imx233/stmp3780)
      - Selection (Test and Development packages)
      - Save and exit.
    - Add [*] in "Configure the kernel“
    - Save and exit
    - Edit your_ltib_dir/config/platform/imx/.config
    - CONFIG_PKG_BOOT_STREAM_CMDLINE1="console=ttyAM0,115200 ubi.mtd=1 root=ubi0:rootfs0 rootfstype=ubifs lcd_panel=lms350 ssp2=gpmi"
    - CONFIG_PKG_BOOT_STREAM_CMDLINE3="console=ttyAM0,115200 root=/dev/mmcblk0p2 rw rootwait lcd_panel=lms350 ssp1=mmc”
  - ./ltib
    - Enter "Device Drivers"
    - Enter “Memory Technology Device (MTD) support"
    - Enter "NAND Device Support“
    - Remove <*/M> in "GPMI LBA NAND driver"
    - Add <*> in "GPMI NAND driver"
  - Save and exit. Wait the rebuild complete.
  - ./ltib -p boot_stream -f

How to boot and mount rootfs from NAND

- Create two partitions on SD/MMC
  - 16MB for Linux kernel (type id 53, OnTrack DM6 Aux3)
  - The rest for rootfs (type id 83, Linux)
- Prepare Linux kernel on SD/MMC
  - Assume /dev/sdX is your SD/MMC device
  - dd if=/dev/zero of=mmc_boot_partition.raw bs=512 count=4
  - dd if=your_ltib_dir/rootfs/boot/stmp378x_linux.sb of=mmc_boot_partition.raw ibs=512 seek=4 conv=sync,notrunc
  - dd if=mmc_boot_partition.raw of=/dev/sdX1
- Prepare rootfs on SD/MMC
  - mkfs.ext2 /dev/sdX2
  - mkdir /mnt/mmc
  - mount /dev/sdX2 /mnt/mmc -t ext2
  - cp -a your_ltib_dir/rootfs/* /mnt/mmc
- Compress rootfs and store it on SD/MMC
  - cd your_ltib_dir/rootfs
  - tar vjcf /mnt/mmc/root/rootfs.tar.bz2 *
  - umount /mnt/mmc

How to boot and mount rootfs from NAND

- Flash Linux kernel to NAND
  - Boot from SD/MMC and **hold SW5** during bootup
    - Hold SW5 to select CONFIG_PKG_BOOT_STREAM_CMDLINE3 which will mount MMC as rootfs
  - Login as root
  - flash_eraseall /dev/mtd0
  - kobs-ng init /boot/stmp378x_linux.sb /dev/mtd0
- Flash rootfs to NAND
  - flash_eraseall /dev/mtd1
  - ubiattach /dev/ubi_ctrl -m 1
  - ubimkvol /dev/ubi0 -N rootfs0 –m
  - mkdir /mnt/ubifs
  - mount -t ubifs ubi0:rootfs0 /mnt/ubifs
  - tar vjxf rootfs.tar.bz2 -C /mnt/ubifs
  - umount /mnt/ubifs
- Set boot mode to 0100 to boot from NAND

How to mount host's nfs as rootfs over usb0

- Prepare Linux kernel & rootfs
  - ./ltib -m config
    - Add [*] in "Configure the kernel"
    - Enter "Package list" and edit CONFIG_PKG_BOOT_STREAM_CMDLINE1
      - console=ttyAM0,115200 lcd_panel=lms350 ssp1=spi1 ip=192.168.1.9:192.168.1.1::::usb0:off root=/dev/nfs nfsroot=/targetrootfs,rsize=1024,wsize=1024
      - Where 192.168.1.9 is the target device's ip and 192.168.1.1 is the host's ip
    - Save your new configuration
  - ./ltib
    - Enter "Device Drivers" and then "USB support"
    - Add <*> in "USB Gadget Support"
    - Add <*> in "USB Gadget Drivers (Ethernet Gadget (with CDC Ethernet support…"
    - Save your new configuration
    - Exit from kernel config menu and ltib will start rebuild.
  - ./ltib -p boot_stream –f
  - ln -s your_ltib_dir/rootfs /targetrootfs
  - Add one line to /etc/exports
    - /targetrootfs192.168.1.*(rw,sync,no_root_squash,no_all_squash,no_subtree_check)
  - /etc/init.d/nfs-kernel-server restart
  - "dd" stmp378x_linux.sb to SD/MMC
    - Refer to "Prepare Linux kernel on SD/MMC" in previous Howto

How to mount host's nfs as rootfs over usb0

- On Target Side
  - Set 3780 boot mode to 1001 (MMC boot)
  - Connect serial port to host
  - Connect power adapter to device
  - Target will boot from SD/MMC upon connect USB port to host
  - Target will look for rpc service from 192.168.1.1

- On Host Side
  - modprobe cdc_ether
  - modprobe usbnet
  - minicom
    - Monitor target's activity
  - ifconfig usb0 192.168.1.1 up
    - May need to repeat multiple times because of the conflicts from NetworkManager
    - The next slide can eliminate NetworkManager problem

How to mount host's nfs as rootfs over usb0

- On Host Side
  - Stop & Disable NetworkManager
    - sudo /etc/init.d/NetworkManager stop
    - sudo update-rd.d -f NetworkManager remove
  - Edit /etc/udev/rules.d/85-ifupdown.rules
    - Add following line before the line LABEL="net_end"
    - KERNEL=="usb0", RUN+="/sbin/ifconfig usb0 192.168.1.1 netmask 255.255.255.0 up"
  - Edit /etc/network/interfaces
    - auto lo eth0 usb0
      - We need to add eth0 because NetworkManager has been disabled
    - iface lo inet loopback
      - This line should present already
    - iface eth0 inet dhcp
      - We need this line also because NetworkManager has been disabled
    - iface usb0 inet static
    - address 192.168.1.1
    - netmask 255.255.255.0

freescale™ semiconductor

How to make USB Mass Storage

- On Target Device, create a disk image (128MByte in this example)
  - dd if=/dev/zero of=/root/storage.img bs=1M count=128
  - fdisk -b 512 -C 128 -H 64 -S 32 /root/storage.img
    - Create New Primary partition with Type b (Win95 FAT32)
  - losetup -o 16384 /dev/loop0 /root/storage.img
    - The purpose of losetup is to setup loop0 device with one track (reserved for partition table) offset, so that mkfs/mount can work from the start location of FAT32 partition.
  - mkfs.vfat -v -F 32 -n MSC-3780BSP /dev/loop0
  - losetup –d /dev/loop0
    - De-associate /dev/loop0 device
- Load g_file_storage kernel module
  - modprobe g_file_storage file=/root/storage.img
- Connect Target Device to Windows PC or any USB host
- After disconnected from host, target can access the content in storage.img by:
  - losetup -o 16384 /dev/loop0 /root/storage.img
  - mount -o loop /dev/loop0 /mnt/msc
- Before connect to host again or shutdown
  - umount /mnt/msc
  - losetup –d /dev/loop0
- An Important Warning!
  - USB host will use the backing storage as a private disk drive. It will not expect to see any changes in the backing storage other than the ones it makes.
  - Only one system (normally, the USB host) may write to the backing storage, and if one system is writing that data, no other should be reading it.

Thank You!

freescale ™
semiconductor