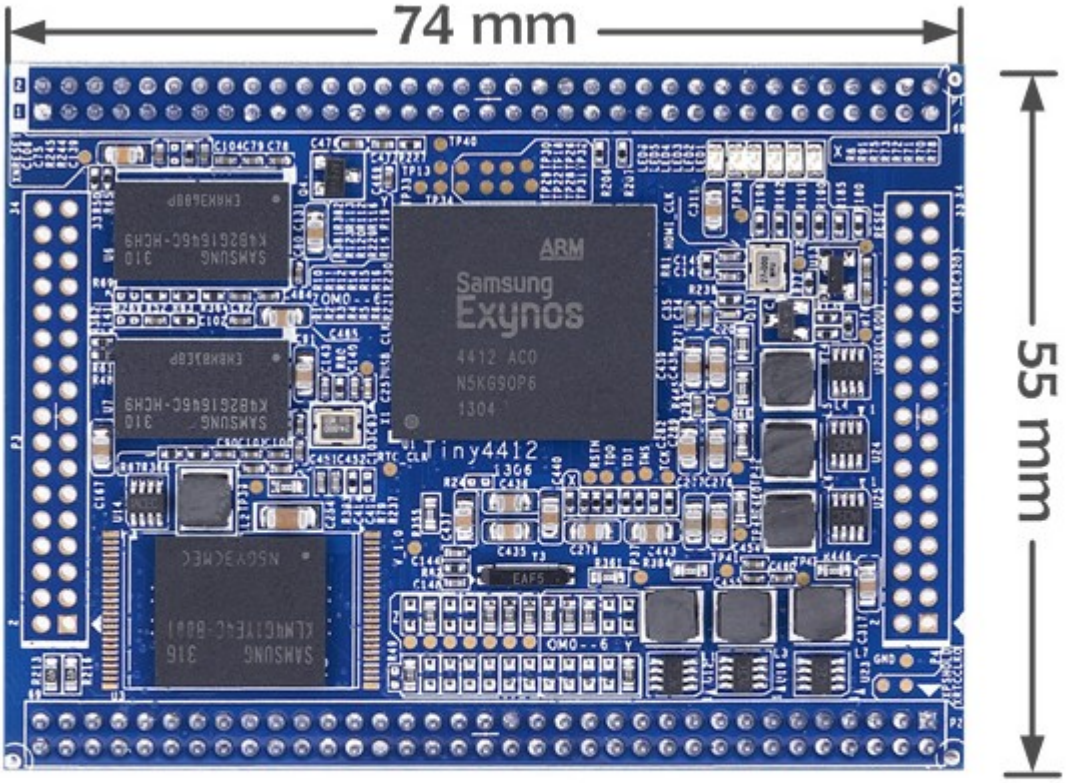
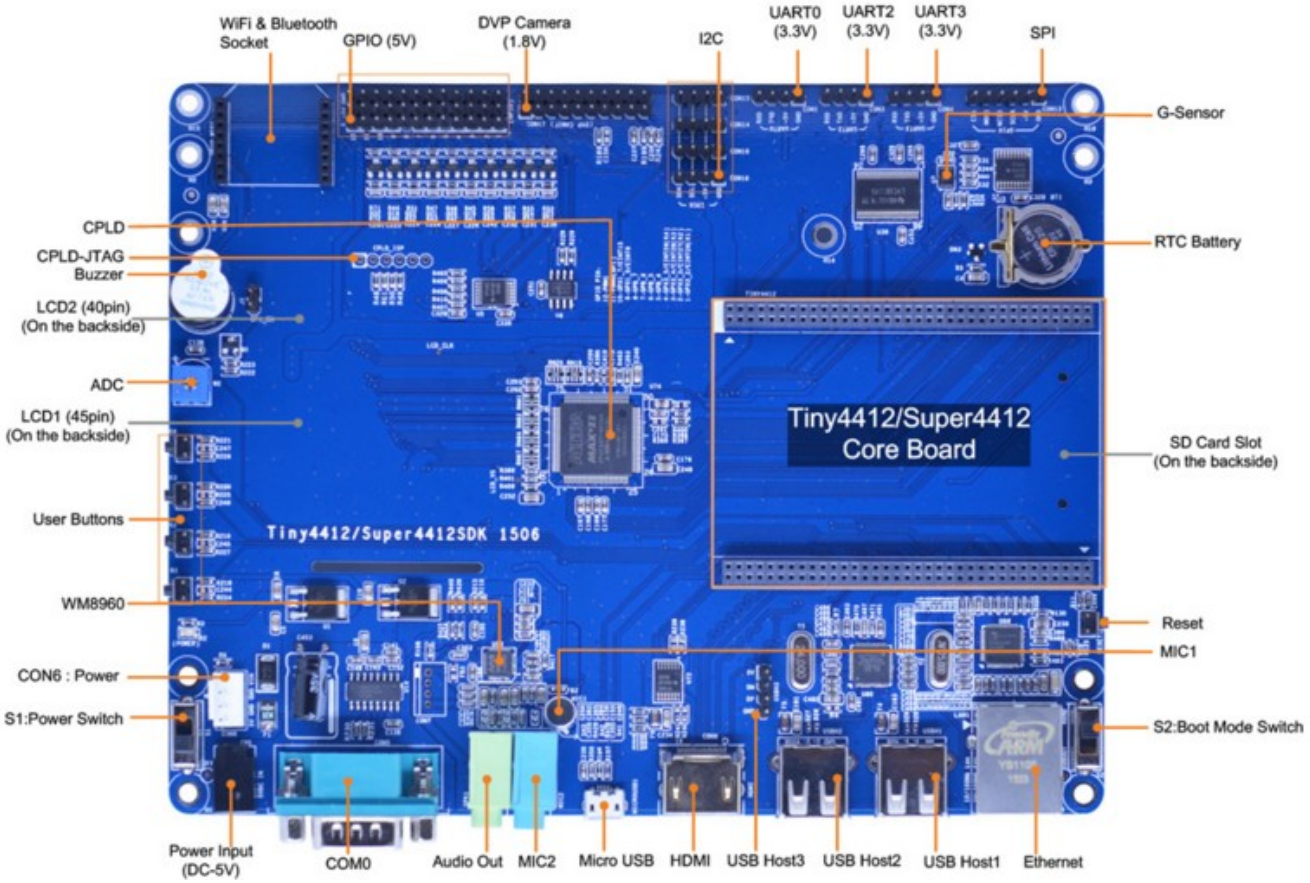
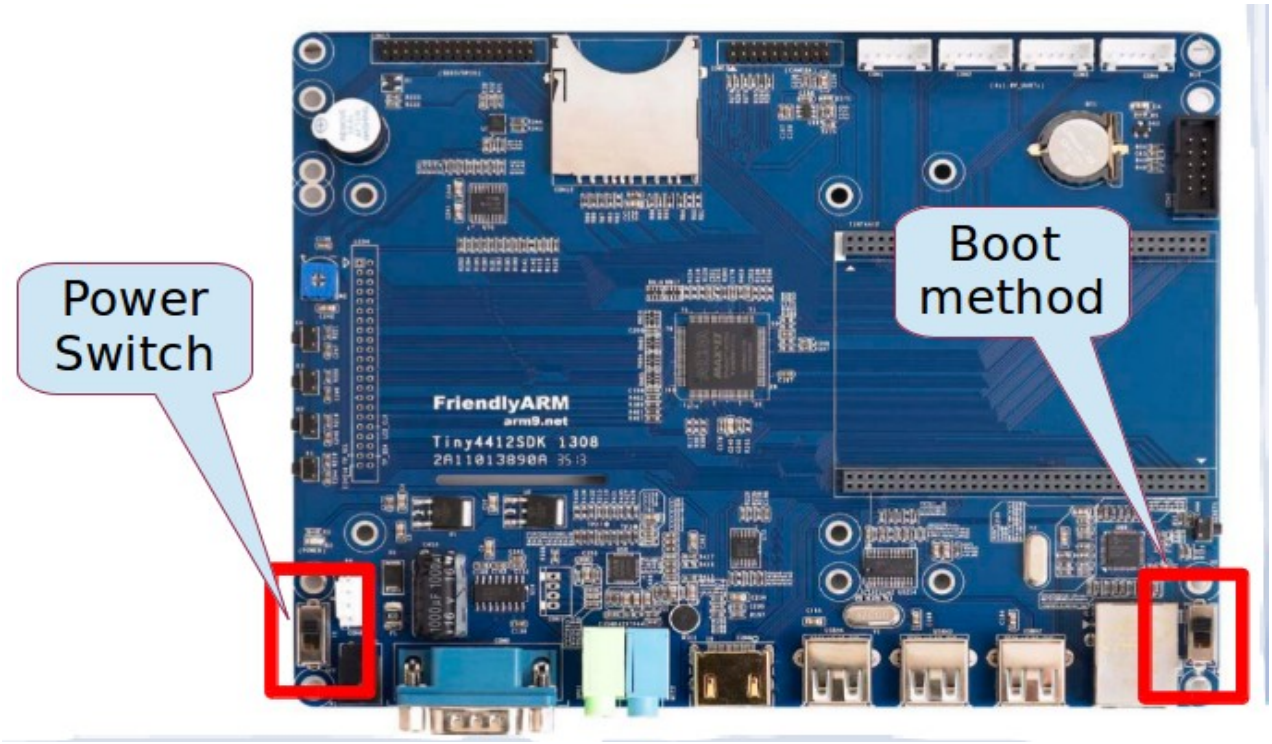
Day 1

Tiny4412 Hardware




WiFi & Bluetooth Socket
GPIO (5V)
DVP Camera (1.8V)
I2C
UART0 (3.3V)
UART2 (3.3V)
UART3 (3.3V)
SPI
G-Sensor
CPLD
CPLD-JTAG
Buzzer
LCD2 (40pin) (On the backside)
ADC
LCD1 (45pin) (On the backside)
User Buttons
WM8960
CON6 : Power
S1:Power Switch
RTC Battery
Tiny4412/Super4412 Core Board
SD Card Slot (On the backside)
Tiny4412/Super4412SDK 1506
Reset
MIC1
S2:Boot Mode Switch
Power Input (DC-5V)
COM0
Audio Out
MIC2
Micro USB
HDMI
USB Host3
USB Host2
USB Host1
Ethernet

Boot Select

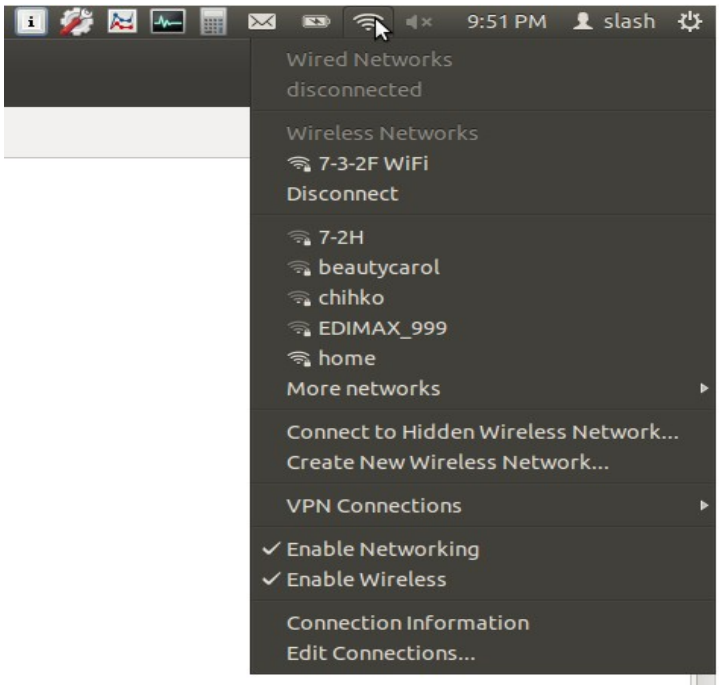


Boot Medthod :

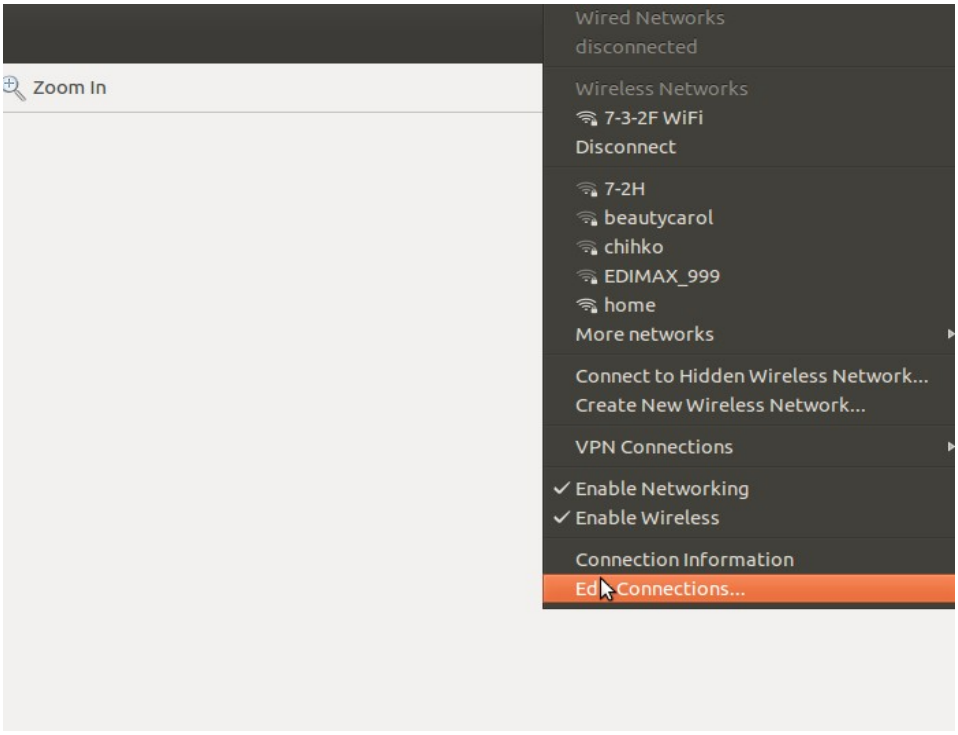
1. **SD Boot**
2. **eMMC boot (板子上面寫 NAND)**

Network setting on your [Host PC]

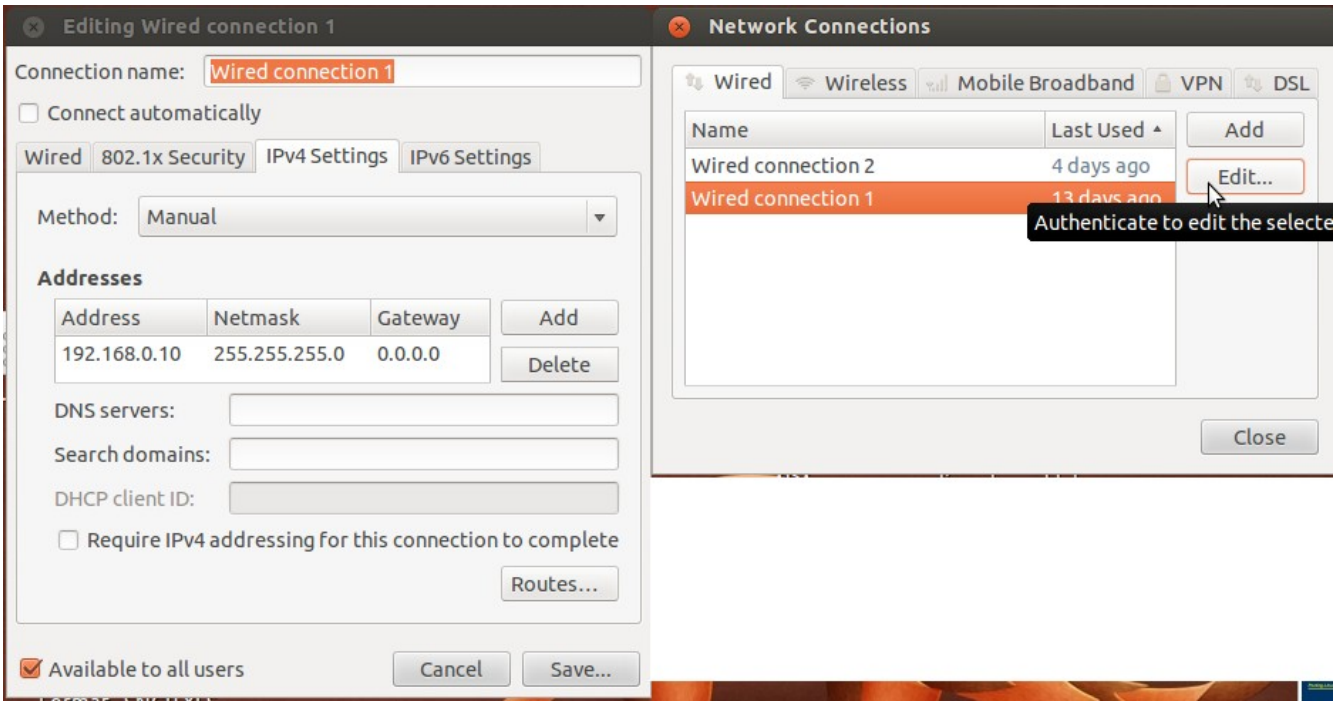
1.



2.



3.


Editing Wired connection 1
Connection name:
Wired connection 1
Connect automatically
Wired
802.1x Security
IPv4 Settings
IPv6 Settings
Method:
Manual
Addresses
Address
Netmask
Gateway
Add
192.168.0.10
255.255.255.0
0.0.0.0
Delete
DNS servers:
Search domains:
DHCP client ID:
Require IPv4 addressing for this connection to complete
Routes...
Available to all users
Cancel
Save...
Network Connections
Wired
Wireless
Mobile Broadband
VPN
DSL
Name
Last Used
Add
Wired connection 2
4 days ago
Edit...
Wired connection 1
Authenticate to edit the selecte
Close

==========

Develop Environment Setting

==========

1. install gtkterm

   #sudo apt-get install gtkterm

   install geany

   #sudo add-apt-repository ppa:geany-dev/ppa
   #sudo apt-get update
   #sudo apt-get install geany

2. install toolchan

   # cd ~/tiny4412/experiment/
   # tar -xvf toolchain/arm-linux-gcc-4.5.1-v6-vfp-20120301.tgz

3. setting toolchain

   #source set_arm_4412_toolchain_path.sh

4. install library

   #sudo apt-get install autoconf automake libtool libexpat1-dev \
   libncurses5-dev bison flex patch curl cvs texinfo git bc \
   build-essential subversion gawk python-dev gperf unzip \
   pkg-config wget

==========

set_arm_4412_toolchain_path.sh

==========

1. #geany set_arm_4412_toolchain_path.sh

2. We can found below Linux shell code ...

   ########################
   #!/bin/sh

   PATH=$PATH:/home/cadtc/tiny4412/experiment/opt/FriendlyARM/toolschain/4.5.1/bin

   export CROSS_COMPILE=arm-none-linux-gnueabi-
   export ARCH=arm
   ########################

3. #arm-none-linux-gnueabi-gcc -v

==========

Setting and Test Tool chain

==========

1. # source ~/tiny4412/experiment/set_arm_4412_toolchain_path.sh.sh

2. check toolchain : #arm-none-linux-gnueabi-gcc -v

3. test tool chain : build example application (hello_word.c)
   # cd ~/tiny4412/experiment/day1
   # arm-none-linux-gnueabi-gcc hello.c -o hello

==========

Build DNW Transmit Application (Host)

==========

0. Install library
   sudo apt-get install libusb-dev

1. Build DNW module
   a) cd ~/tiny4412/experiment/tool/dnw
   b) gcc dnw_new.c -o dnw -lusb
   c) sudo cp -a ./dnw /usr/bin

2. Test DNW – Download file to EVB
   a) u-boot side
      # dnw [address] (default is 0xc00000000)
   b) Host side
      # cd ~/tiny4412/experiment/init_boot
      # sudo ./dnw zImage

==========

SD boot - Make SD boot card

==========

1. find pre-build binary file (~/tiny4412/experiment/init_boot)

2. Check pre-build binary file
   # ls ~/tiny4412/experiment/init_boot_v2/sd_fuse/tiny4412/E4412_N.bl1.bin
   # ls ~/tiny4412/experiment/init_boot_v2 /sd_fuse/tiny4412/mkbl2.bin
   # ls ~/tiny4412/experiment/init_boot_v2 /sd_fuse/tiny4412/E4412_tzsw.bin
   # ls ~/tiny4412/experiment/init_boot_v2 /u-boot-v2.bin

3. Format SD card
   #**~/tiny4412/experiment/init_boot_v2/sd_fuse/sd_fdisk /dev/sdb**

4. Run image install script
   #~/tiny4412/experiment/init_boot_v2/sd_fuse/tiny4412/sd_fusing.sh

==========

eMMC boot

==========

[Target] u-boot 環境

0.

a) use sd boot (SD 卡 開機) → # setenv mmc_device 1

b) use emmc boot (eMMC 開機) → # setenv mmc_device 0

[Host]

8. sudo dnw ~/tiny4412/experiment/init_boot_v2/E4412_N.bl1.bin
11. sudo dnw ~/tiny4412/experiment/init_boot_v2/bl2.bin
14. sudo dnw ~/tiny4412/experiment/init_boot_v2/u-boot.bin
17. sudo dnw ~/tiny4412/experiment/init_boot_v2/E4412_tzsw.bin

[if you have done below command process before, you can bypass below]

1. fdisk -c $mmc_device 600 2048 220
2. fatformat mmc **$(mmc_device)**:1
3. ext3format mmc **$(mmc_device)**:2
4. ext3format mmc **$(mmc_device)**:3
5. ext3format mmc **$(mmc_device)**:4

[Target]
6. emmc open **$(mmc_device)**

(download and write E4412_N.bl1.bin)
7. dnw
9. mmc write **$(mmc_device)** 0xc0000000 0 0x10
(0x10 = 十進位 16, 16 個 block, 一個 block 是 512B, 16*512=8*1024=8KB)

"(download and write bl2.bin)"
10. dnw
12. mmc write **$(mmc_device)** 0xc0000000 0x10 0x1C

"(download and write u-boot.bin)"
13. dnw
15. mmc write **$(mmc_device)** 0xc0000000 0x30 0x21D
(u-boot.bin size 270K，space is 328K. So 48th block start，write 541 blocks,
541*512B=270.5*1024 :=270K)

(download and write E4412_tzsw.bin)
16. dnw
18. mmc write **$(mmc_device)** 0xc0000000 0x2c0 0xB8
(tzsw.bin size about 92K，space is 160K. So, 704 block start，write 184 blocks,
184*512=92*1024=92K)

19. emmc close **$(mmc_device)**

==========

Extract rootfs

==========

[Host]
1. # cd ~/tiny4412/experiment
2. # sudo tar -xvjf root_mkfs.tar.bz2

==========

Download and Booting kernel Image

==========

Please insert microUSB first

[Host]
2. sudo dnw ~/tiny4412/experiment/init_boot_v2/Image
6. ifconfig ethx (x = 0, 1, 2, 3 ….)192.168.0.10

[Target]
1. dnw 0xc0008000
3. setenv serverip 192.168.0.10
4. setenv ipaddr 192.168.0.20
5. set bootargs noinitrd init=/linuxrc root=/dev/nfs ip=192.168.0.20:192.168.0.10:192.168.0.1:255.255.255.0::eth0:on nfsroot=192.168.0.10:/home/cadtc/tiny4412/experiment/root_mkfs, console=ttySAC0 lcd=S70

6. savenv
7. bootm 0xc00080000

==========

write kernel Image to eMMC

==========

[Hots]
2. sudo dnw ~/tiny4412/experiment/init_boot/Image

[Target]
1. dnw
3. movi write kernel $mmc_device 0xc0000000

[**boot kernel]**
4.
movi read kernel $mmc_device C0008000
bootm c0008000

==========

Build NFS environment

==========

1. sudo apt-get install nfs-kernel-server nfs-common portmap

2. configure portmap
   a. sudo vim /etc/default/portmap
   b. del 127.0.0.1

3. comfigure exports
   a. sudo geany _/etc/exports
   b. add your share file folder

   example :
   /home/cadtc/tiny4412/experiment/root_mkfs *(rw,sync,no_root_squash)

   *: any ip can entry this

4. restart
   a. sudo /etc/init.d/portmap restart
   b. sudo /etc/init.d/nfs-kernel-server restart

5. Host :
   ifconfig ethx 192.168.0.10

6. sudo tar -xvjf root_mkfs.tar.bz2

7. local test
   a.
   # sudo mount -t nfs 192.168.0.10:/home/cadtc/tiny4412/experiment/root_mkfs   /mnt/

   b. # ls -l /mnt
   c. # sudo umount /mnt

Day 2

GNU C Compiler

==========

exercise 1 :

==========

-S :

Compile only; do not assemble or link

#gcc -S main.c

-c :

Compile and assemble, but do not link

#gcc -c main.c

==========

exercise 2 :

==========

-o <file> : Place the output into <file>

a)

gcc -o main main.o

./main

b)

ls

rm main.o main

ls

gcc -o main main.c

./main

==========

exercise 3 :

==========

-D : defines a macro to be used by the preprocessor

```
# rm main
# gcc -DD_FLAG -o main main.c
# ./main
```

==========

exercise 4 :

==========

-I : adds include directory of header files

```
#rm main.o
#gcc -DI_FLAG -Itest_inc -o main main.c func_1.c
#./main
```

==========

exercise 5 :

==========

-l : links with a library file

```
#rm main
gcc -lm -DL_FLAG -Itest_inc -o main main.c
```

Exercise Makefile

==========

hello_world_ex1 : Signal source file

==========

1. # cd hello_world_ex1
2. # make

==========

hello_world_ex2 : Multi source file

==========

1. # make
2. # ./main

==========

hello_world_ex4 : Double colon rule

==========

1. # make
2. # ls -l main

3. # touch debug
4. # make
5. # ls -l main

==========

hello_world_ex5 : Automatic Variables

==========

1. # make
2. # ./hello_world

Binutils Tool

==========

exercise ar:

==========

a) static library

1. cd create_static_lib
2. make

b) share library

1. cd create_dymamic_lib
2. make

==========

exercise nm:

==========

x86-PC

1. #make
2. #nm -help
3. #nm main.o
4. #nm -a -l main.o
5. #nm -a -A main.o

ARM

1. #make
2. #arm-none-linux-gnueabi-nm -help
3. #arm-none-linux-gnueabi-nm main.o
4. #arm-none-linux-gnueabi-nm -a -l main.o
5. #arm-none-linux-gnueabi-nm -a -A main.o

==========

objdump:

==========

x86-PC

1. #make
2. #objdump -help
3. #objdump -d main.o
4. #objdump -t main.o

ARM

1. #make
2. #arm-none-linux-gnueabi-objdump -help
3. #arm-none-linux-gnueabi-objdump -d main.o
4. #arm-none-linux-gnueabi-objdump -t main.o

==========

strip:

==========

x86-PC

#ls -l

#arm-none-linux-gnueabi-strip exerise

#ls -l

ARM

#ls -l

#arm-none-linux-gnueabi-strip exerise

#ls -l

Patch file Exercise

1) create patch file
#cd /home/cadtc/tiny4412/experiment/day2/6-patch_exerise/exercise_1
#diff -Naur hello_1.c hello_2.c > hello.patch

2) add patch file
#../cd exercise_2/
#cp ../exercise_1/hello.patch ./
#patch -p0 < hello.patch

3) remove patch file
patch -R -p0 < hello.patch

Library Exercise

ar:
A) static library

a) x86-PC
1. #cd ~/tiny4412/experiment/create_static_lib
2. #make

b) ARM
1. #source ~/tiny4412/experiment/tool/set_arm_4412_toolchain_path.sh
2. #cd create_static_lib
3. #make
4. #./test

b) share library

a) x86-PC
1 #cd create_dymamic_lib
2 #export CC=gcc
3 #export DIR=$('pwd')
4 #make
5 #export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/cadtc/tiny4412/experiment/day2/3-build_library/create_dymamic_lib

b) ARM
1 #cd create_dymamic_lib
2 #make
3 export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/cadtc/tiny4412/experiment/day2/3-build_library/create_dymamic_lib

Day3 U-Boot

BASIC Command

http://www.denx.de/wiki/view/DULG/UBoot

1 help

2 mtest - simple RAM test

example :
#mtest 40000000 400000010 0xffff 0x20

3 printenv - printenv - print environment variables

example :
#printenv

4 setenv - setenv - set environment variables
savenv - saveenv - save environment variables to persistent storage
-
example :
#printenv
#setenv bootdealy 10
#printenv
#saveenv

5 mm - memory modify (auto-incrementing address)

example
TINY4412 # mm
40000000: ffffffff ? ffffffff
40000004: 12345678 ? 12345678
40000008: 00000002 ? 76543210
4000000c: 00000003 ? 23456754

6 md - memory display

example
TINY4412 # md.l 40000000
40000000: ffffffff 12345678 76543210 23456754 ....xV4..2TvTgE#
40000010: 00000004 00000005 00000006 00000007 ....
40000020: 00000008 00000009 0000000a 0000000b ....

7 macro

example
#setenv test 'md.l 40000000;md.l 50000000'
#run test

8 **saveenv** - save environment variables to persistent storage

example
#setenv test 'md.l 40000000;md.l 50000000'
#saveenv

Build u-boot

1. source ~/tiny4412/experiment/set_arm_4412_toolchain_path.sh

2 cd ~/tiny4412/experiment/day3

3 tar -xvjf uboot_tiny4412_v2.tar.bz2 -C ~/tiny4412/experiment/

4 cd ~/tiny4412/experiment/uboot_tiny4412_v2

5 make distclean

6 make **tiny4412_config**

7 make -j4

8 Then we can check “u-boot.bin” with “ls” command
  # ls ./u-boot.bin

Update u-boot/kernel for SD card

for u-boot

SD boot - Make SD boot card

1. find pre-build binary file

2. Below is pre-build binary file
    ls ~/tiny4412/experiment/init_boot_v2/sd_fuse/tiny4412/E4412_N.bl1.bin
    ls ~/tiny4412/experiment/init_boot_v2/sd_fuse/tiny4412/mkbl2.bin
    ls ~/tiny4412/experiment/init_boot_v2/sd_fuse/tiny4412/E4412_tzsw.bin
    ls ~/tiny4412/experimentinit_boot_v2/u-boot-v2.bin

3. Format SD card

    PC-Linux# sudo ~/tiny4412/experiment/uboot_tiny4412_v2/sd_fuse/sd_fdisk /dev/sdb

4. Run image install script
    PC-Linux# sudo ~/tiny4412/experiment/uboot_tiny4412_v2/sd_fuse/tiny4412/sd_fusing.sh

Update u-boot for eMMC

binary compare tool : vbindiff (sudo apt-get install vbindiff)

***Please copy "/home/cadtc/tiny4412/experiment/day3/combine_bl2_uboot.sh" to U-BOOT source directory first**

[Host]

0. update bl2.bin
mkbl2 u-boot.bin bl2.bin 14336

1. combin bl2.bin and u-boot.bin
cat ./bl2.bin ../u-boot.bin > bl2_uboot.bin

4. sudo dnw ./bl2_uboot.bin

[Target]

==========

if you use sd 開機 boot "mmc_device" = 1

if you use emmc 開機 boot "mmc_device" = 0

==========

sd boot -> #setenv mmc_device 1
eMMC -> #setenv mmc_device 0

2. #emmc open $mmc_device

(download and write bl2_uboot.bin)
3. #dnw

(write bl2)
5. #mmc write $mmc_device **0xc0000000** 0x10 0x1C

(write uboot)
6. #mmc write $mmc_device **0xc0003800** 0x30 0x21D

(u-boot.bin size 270K，space is 328K. So 48th block start，write 541 blocks, 541*512B=270.5*1024 :=270K)

7. #emmc close $mmc_device

Customer command

0. cd ~/tiny4412/experiment/day3/3-create_cmd_exerise/cmd_helloworld

1. geany ~/tiny4412/experiment/uboot_tiny4412_v2/common/Makefile

   add below

========== start ==========
@@ -170,6 +170,7 @@ COBJS-$(CONFIG_LYNXKDI) += lynxkdi.o
 COBJS-$(CONFIG_MODEM_SUPPORT) += modem.o
 COBJS-$(CONFIG_UPDATE_TFTP) += update.o
 COBJS-$(CONFIG_USB_KEYBOARD) += usb_kbd.o
+COBJS-$(CONFIG_HELLOWORLD) += cmd_helloworld.o

 COBJS := $(sort $(COBJS-y))
 XCOBJS := $(sort $(XCOBJS-y))
========== end ==========

2. cp ./cmdhello.c ~/tiny4412/experiment/uboot_tiny4412_v2/common

3. make -j4

4. update u-boot to SD card

5. Target : #help hello

6. Target : #hello

Modify Prompt

1. gedit include/configs/tiny4412.h

2. search "CONFIG_SYS_PROMPT"

3. change "TINY4412 # " to "QEDWWCW #"

4. make (please reference above <Build u-boot>)

5. update u-boot (please reference above <update u-boot for SD card or update u-boot for eMMC>)

Standalone application

1 [Host] :

geany ~/tiny4412/experiment/uboot_tiny4412_v2/arch/arm/config.mk

ifeq ($(BOARD),omap2420h4)
STANDALONE_LOAD_ADDR = 0x80300000
else
ifeq ($(SOC),omap3)
STANDALONE_LOAD_ADDR = 0x80300000
else
STANDALONE_LOAD_ADDR = 0xc0000000
endif
endif

2 [Host] :

cd ~/tiny4412/experiment/uboot_tiny4412_v2&& make

3 [Target]:

dnw c0000000

4 [Host]:

sudo sudo dnw ~/tiny4412/experiment/uboot_tiny4412_v2/examples/standalone/hello_world.bin

5 [Target]:

go c0000000

Boot Linux Kernel

A)

1 [Target] :
dnw 40008000 (c0008000)

2 [Host]:
sudo dnw $(KERNEL_PATH)arch/arm/boot/zImage

3 [Target] :
bootm 40008000 (c0008000)

B)

1 [Target]:
movi read kernel 0 40008000 (c0008000)

2 [Target]:
bootm 40008000 (c0008000)

Day 4 Linux Kernel

Building Linux kerne

0. # source /home/cadtc/tiny4412/experiment/set_arm_4412_toolchain_path.sh

1.
```
# cd /home/cadtc/tiny4412/experiment/day4
# tar -xvjf /home/cadtc/tiny4412/experiment/day4 -C ../
# cd ../linux_3.5.0_tiny4412/
```

2. # make distclean

3. # make tiny4412_linux_defconfig

4. # make -j4

5. # check output kernel image : arch/arm/boot/zImage

MAKE HELP

1. # make help

```
Cleaning targets:
  clean           - Remove most generated files but keep the config and
                    enough build support to build external modules
  mrproper        - Remove all generated files + config + various backup files
  distclean       - mrproper + remove editor backup and patch files

Configuration targets:
  config          - Update current config utilising a line-oriented program
  nconfig         - Update current config utilising a ncurses menu based program
  menuconfig      - Update current config utilising a menu based program
  xconfig         - Update current config utilising a QT based front-end
  gconfig         - Update current config utilising a GTK based front-end
  oldconfig       - Update current config utilising a provided .config as base
  localmodconfig  - Update current config disabling modules not loaded
```

```
  localyesconfig  - Update current config converting local mods to core
  silentoldconfig - Same as oldconfig, but quietly, additionally update deps
  defconfig       - New config with default from ARCH supplied defconfig
  savedefconfig   - Save current config as ./defconfig (minimal config)
  allnoconfig     - New config where all options are answered with no
  allyesconfig    - New config where all options are accepted with yes
  allmodconfig    - New config selecting modules when possible
  alldefconfig    - New config with all symbols set to default
  randconfig      - New config with random answer to all options
  listnewconfig   - List new options
  oldnoconfig     - Same as silentoldconfig but set new symbols to n (unset)

Other generic targets:
  all               - Build all targets marked with [*]
* vmlinux          - Build the bare kernel
* modules          - Build all modules
  modules_install - Install all modules to INSTALL_MOD_PATH (default: /)
  firmware_install- Install all firmware to INSTALL_FW_PATH
                    (default: $(INSTALL_MOD_PATH)/lib/firmware)
  dir/            - Build all files in dir and below
  dir/file.[oisS] - Build specified target only
  dir/file.lst    - Build specified mixed source/assembly target only
                    (requires a recent binutils and recent build (System.map))
  dir/file.ko     - Build module including final link
  modules_prepare - Set up for building external modules
  tags/TAGS       - Generate tags file for editors
  cscope          - Generate cscope index
  gtags           - Generate GNU GLOBAL index
  kernelrelease   - Output the release version string
  kernelversion             - Output the version stored in Makefile
  headers_install - Install sanitised kernel headers to INSTALL_HDR_PATH
                    (default: /home/slash/work/exynos4412/build/linux_3.5.0_tiny4412/usr)

Static analysers
  checkstack      - Generate a list of stack hogs
  namespacecheck  - Name space analysis on compiled kernel
  versioncheck    - Sanity check on version.h usage
  includecheck    - Check for duplicate included header files
  export_report   - List the usages of all exported symbols
  headers_check   - Sanity check on exported headers
  headerdep       - Detect inclusion cycles in headers
  coccicheck      - Check with Coccinelle.

Kernel packaging:
  rpm-pkg           - Build both source and binary RPM kernel packages
  binrpm-pkg          - Build only the binary kernel package
```

```
  deb-pkg            - Build the kernel as a deb package
  tar-pkg            - Build the kernel as an uncompressed tarball
  targz-pkg          - Build the kernel as a gzip compressed tarball
  tarbz2-pkg         - Build the kernel as a bzip2 compressed tarball
  tarxz-pkg          - Build the kernel as a xz compressed tarball
  perf-tar-src-pkg   - Build perf-3.5.0.tar source tarball
  perf-targz-src-pkg - Build perf-3.5.0.tar.gz source tarball
  perf-tarbz2-src-pkg - Build perf-3.5.0.tar.bz2 source tarball
  perf-tarxz-src-pkg - Build perf-3.5.0.tar.xz source tarball

Documentation targets:
 Linux kernel internal documentation in different formats:
  htmldocs       - HTML
  pdfdocs        - PDF
  psdocs         - Postscript
  xmldocs        - XML DocBook
  mandocs        - man pages
  installmandocs - install man pages generated by mandocs
  cleandocs      - clean all generated DocBook files

Architecture specific targets (arm):
* zImage        - Compressed kernel image (arch/arm/boot/zImage)
  Image         - Uncompressed kernel image (arch/arm/boot/Image)
* xipImage      - XIP kernel image, if configured (arch/arm/boot/xipImage)
  uImage        - U-Boot wrapped zImage
  bootpImage    - Combined zImage and initial RAM disk
                  (supply initrd image via make variable INITRD=<path>)
  dtbs          - Build device tree blobs for enabled boards
  install       - Install uncompressed kernel
  zinstall      - Install compressed kernel
  uinstall      - Install U-Boot wrapped compressed kernel
                  Install using (your) ~/bin/installkernel or
                  (distribution) /sbin/installkernel or
                  install to $(INSTALL_PATH) and run lilo

  acs5k_defconfig          - Build for acs5k
  acs5k_tiny_defconfig     - Build for acs5k_tiny
  afeb9260_defconfig       - Build for afeb9260
  ag5evm_defconfig         - Build for ag5evm
  am200epdkit_defconfig    - Build for am200epdkit
  lart_defconfig           - Build for lart
  lpc32xx_defconfig        - Build for lpc32xx
  lpd270_defconfig         - Build for lpd270
  lubbock_defconfig        - Build for lubbock
  sam9_l9260_defconfig     - Build for sam9_l9260
  shannon_defconfig        - Build for shannon
```

```
  shark_defconfig          - Build for shark
  simpad_defconfig         - Build for simpad
  spear13xx_defconfig      - Build for spear13xx
  spear3xx_defconfig       - Build for spear3xx
  spear6xx_defconfig       - Build for spear6xx
  spitz_defconfig          - Build for spitz
  stamp9g20_defconfig      - Build for stamp9g20
  tct_hammer_defconfig     - Build for tct_hammer
  tegra_defconfig          - Build for tegra
  tiny4412_linux_defconfig - Build for tiny4412_linux
  tiny4412_ubuntu_defconfig - Build for tiny4412_ubuntu
  trizeps4_defconfig       - Build for trizeps4
  u300_defconfig           - Build for u300
  u8500_defconfig          - Build for u8500
  usb-a9260_defconfig      - Build for usb-a9260
  versatile_defconfig      - Build for versatile
  vexpress_defconfig       - Build for vexpress
  viper_defconfig          - Build for viper
  xcep_defconfig           - Build for xcep
  zeus_defconfig           - Build for zeus

  make V=0|1 [targets] 0 => quiet build (default), 1 => verbose build
  make V=2   [targets] 2 => give reason for rebuild of target
  make O=dir [targets] Locate all output files in "dir", including .config
  make C=1   [targets] Check all c source with $CHECK (sparse by default)
  make C=2   [targets] Force check of all c source with $CHECK
  make RECORDMCOUNT_WARN=1 [targets] Warn about ignored mcount sections
  make W=n   [targets] Enable extra gcc checks, n=1,2,3 where
                  1: warnings which may be relevant and do not occur too often
                  2: warnings which occur quite often but may still be relevant
                  3: more obscure warnings, can most likely be ignored
                  Multiple levels can be combined with W=12 or W=123

Execute "make" or "make all" to build all targets marked with [*]
For further info see the ./README file
```

tiny4412_linux_defconfig - Build for tiny4412_linux

==========

Building and install modules

0. # cd ~/tint4412/experiment/linux_3.5.0_tiny4412

1. # make modules

2. # make kernelrelease

3. # INSTALL_MOD_PATH=${PRJROOT}/modules-KERNEL-VERSION

4. # make modules_install

Modify ourself kernel version name

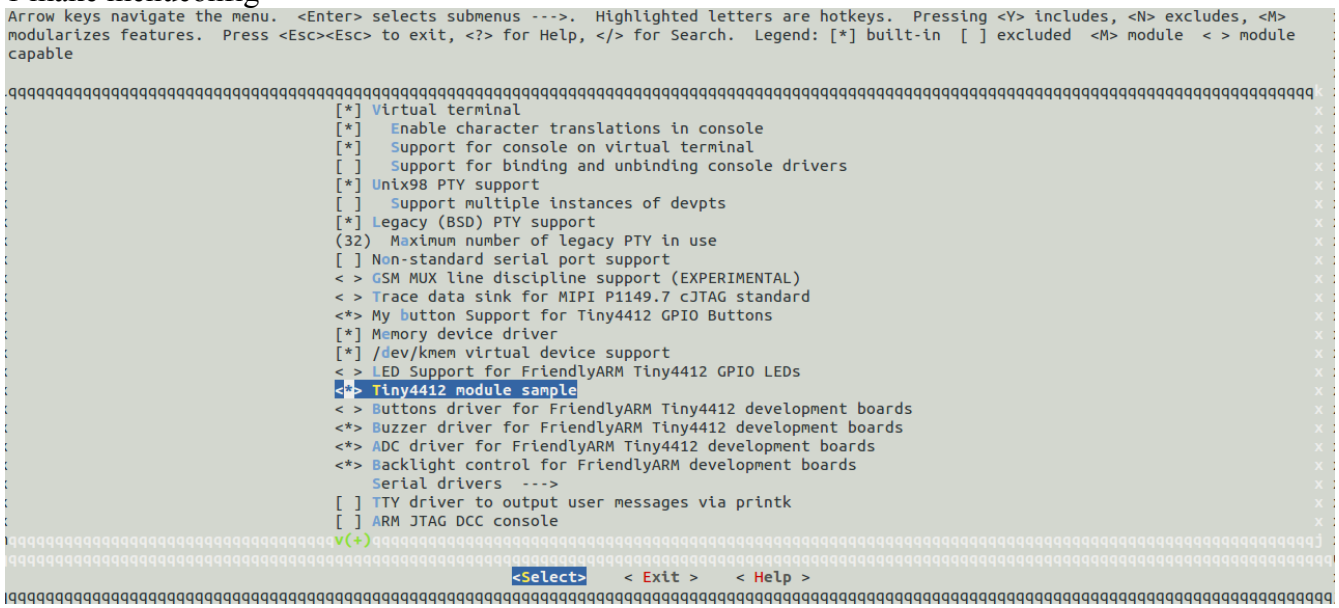
1. # make kernelrelease

2 # make menuconfig

3 # General setup → () Local version

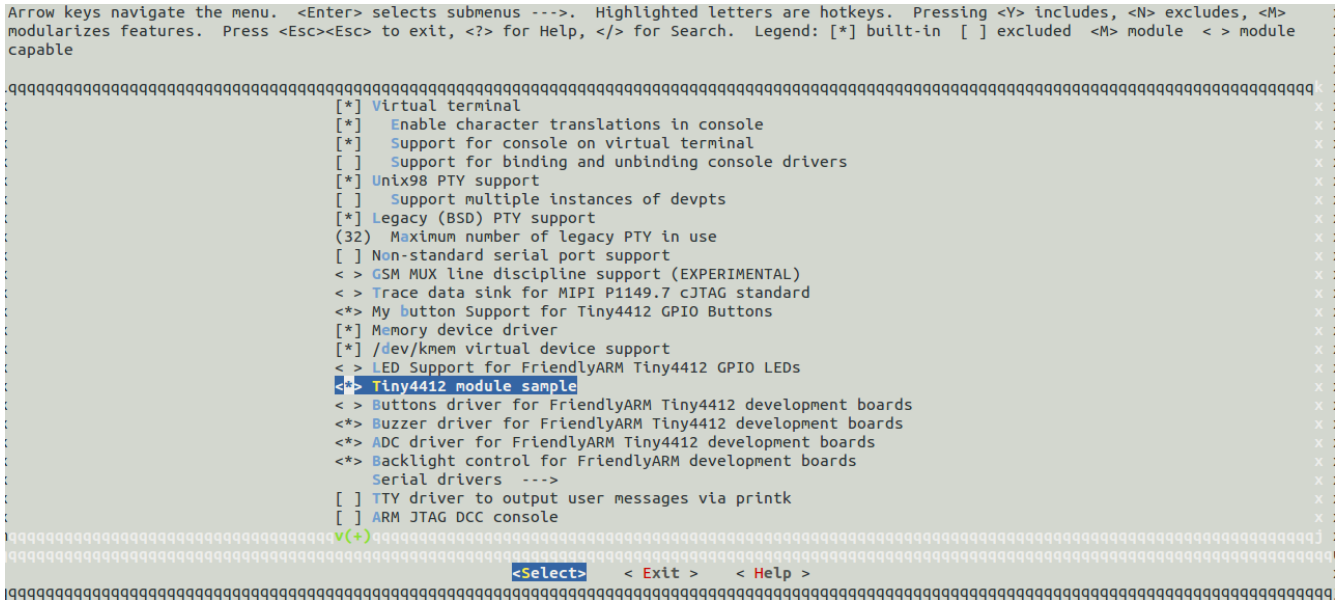
Configure kernel (use menuconfig for configuration)

1 make menuconfig

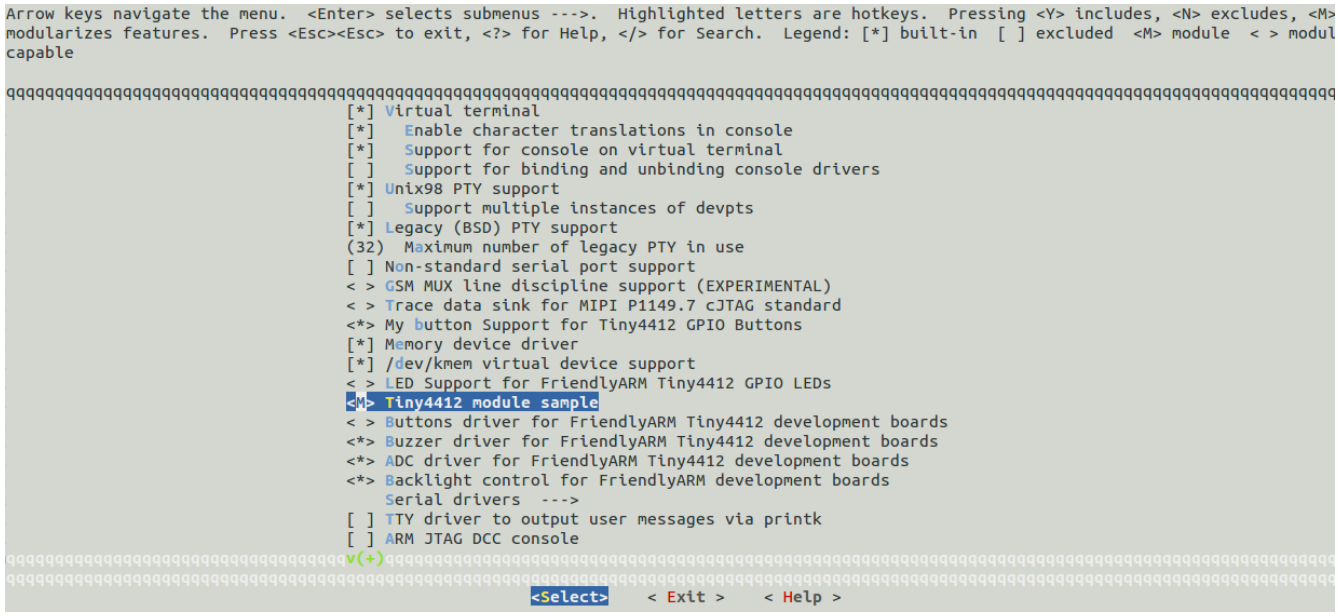


2 add/delete driver

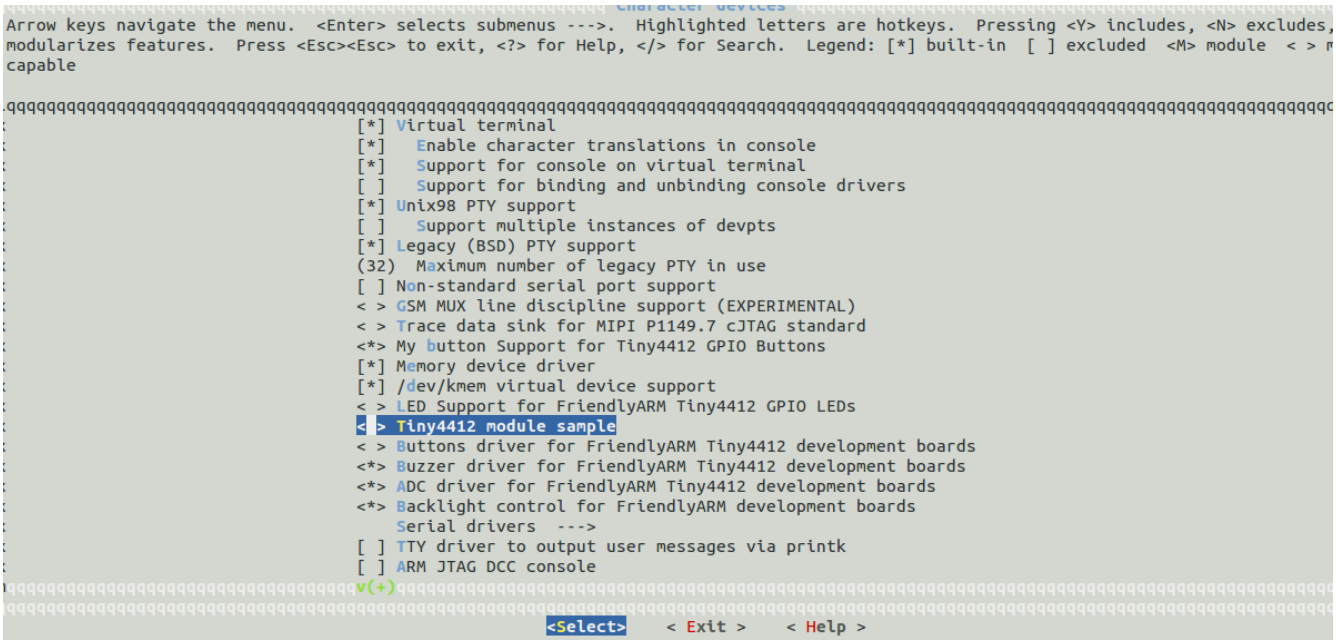
add – build-in



add – build module



delete



==========

Create kernel configure file

==========

1. We will get defconfig in local currently directory
   # make savedefconfig

2. rename defconfig to hello_defconfig
   # mv defconfig hello_defconfig

3. put hello_deconfig to arch/arm/configs
   # mv hello_deconfig arch/arm/configs

4. confirm our configure
   # make help | grep hello

==========

Change Linux boot screen

==========

1. [HOST PC] # sudo apt-get install netpbm

2. [HOST PC] # pngtopnm ~/tiny4412/experiment/day4/2-Change_Linux_boot_screen-exerise/larry.png | ppmquant -fs 223 | pnmtoplainpnm > logo_larry_clut224.ppm

3. copy the created .ppm file into ~/tiny4412//experiment/day4/linux_3.5.0_tiny4412/drivers/video/logo folder

   [HOST PC]
   # cp -a logo_larry_clut224.ppm ~/tiny4412/experiment/day4/linux_3.5.0_tiny4412/drivers/video/logo

4. Add the entry in the Kconfig (drivers/video/logo/Kconfig)
   [HOST PC]
   # geany ~/tiny4412/experiment/day4/linux_3.5.0_tiny4412/drivers/video/logo/Kconfig

   add below to "drivers/video/logo/Kconfig"

==========

```
        #if LOGO

+       config LOGO_LARRY_CLUT224
+               bool "Test 224-colour logo"
+               depends on FB=y
```

==========

5. Add the code in
   a) logo.c
   # geany ~/tiny4412/Day4/linux_3.5.0_tiny4412/drivers/video/logo/logo.c

==========

```
+               #ifdef CONFIG_LOGO_LARRY_CLUT224;
+                       logo = &logo_larry_clut224;
+               #endif

                }
                return logo;
        }
```

```
                EXPORT_SYMBOL_GPL(fb_find_logo);
```

==========

b) include/linux/linux_logo.h
geany ~/tiny4412/Day4/linux_3.5.0_tiny4412/include/linux/linux_logo.h

==========

```
                extern const struct linux_logo logo_m32r_clut224;
                extern const struct linux_logo logo_spe_clut224;
+               extern const struct linux_logo logo_larry_clut224;
```

==========

6. Add an entry in the Makefile(drivers/video/logo/Makefile)

[HOST PC]
geany ~/tiny4412/experiment/day4/linux_3.5.0_tiny4412/drivers/video/logo/Makefile

==========

```
add below to "drivers/video/logo/Makefile"
obj-$(CONFIG_LOGO_SUPERH_CLUT224)       += logo_superh_clut224.o
obj-$(CONFIG_LOGO_M32R_CLUT224)         += logo_m32r_clut224.o
+       obj-$(CONFIG_LOGO_LARRY_CLUT224) += logo_larry_clut224.o
```

==========

7. [HOST PC]

```
# make menuconfig
Device Drivers  --->
        Graphics support  --->
                [*] Bootup logo  --->
                        --- Bootup logo
                        [*]   Test 224-colour logo
                        [ ]   Standard black and white Linux logo
                        [ ]   Standard 16-color Linux logo
                        [ ]   Standard 224-color Linux logo

# make -j4
```

8. Download Kernel Image and boot it
Please check “Download and Booting kernel Image” in ~/tiny4412/note/boot/day1_Exerise.txt

==========

Download kernel Image

==========

[Host]

2. sudo dnw ~/tiny4412/experiment/day4/linux_3.5.0_tiny4412/arch/arm/boot/zImage

6. ifconfig ethx 192.168.0.10

[Target] (u-boot)

1. dnw 0xc0008000
3. setenv serverip 192.168.0.10
4. setenv ipaddr 192.168.0.20
5. set bootargs noinitrd init=/linuxrc root=/dev/nfs ip=192.168.0.20:192.168.0.10:192.168.0.1:255.255.255.0::eth0:on nfsroot=192.168.0.10:/home/cadtc/tiny4412/experiment/root_mkfs, console=ttySAC0 lcd=S70

6. savenv
7. bootm 0xc00080000

==========
Please make sure ARM toolchain setting finish first !!!!!
==========

A. adc-test

[Host]

0. # cd ~/tiny4412/day4/linux_3.5.0_tiny4412/
1. # make menuconfig
   Device Drivers --->
   Character devices --->
   <*> ADC driver for FriendlyARM Tiny4412 development boards
2. # cd adc-test/
3. # make
4. # cp -a ./adc-test ~/tiny4412/experiment/root_mkfs

[Target]

5. # ./adc-test

B. buttons

[Host]

0. # cd ~/tiny4412/day4/linux_3.5.0_tiny4412/
1. # make menuconfig
   Device Drivers --->
   Character devices --->
   <*> Buttons driver for FriendlyARM Tiny4412 development boards
2. # cd buttons/
3. # make
4. # cp -a ./buttons ~/tiny4412/experiment/root_mkfs

[Target]

5. # ./buttons

C. I2C

[Host]

0. # cd ~/tiny4412/day4/linux_3.5.0_tiny4412/
1. # make menuconfig
   Device Drivers --->

<*> I2C support --->
I2C Hardware Bus support --->
<*> S3C2410 I2C Driver

2. # cd i2c/
3. # make
4. # cp -a ./i2c ~/tiny4412/experiment/root_mkfs

[Target]

5. # ./i2c

D. led-payer

[Host]

0. # cd ~/tiny4412/day4/linux_3.5.0_tiny4412/
1. # make menuconfig
Device Drivers --->
Character devices --->
<*> LED Support for FriendlyARM Tiny4412 GPIO LEDs
2. # cd led-payer/
3. # make
4. # cp -a ./led-payer ~/tiny4412/experiment/root_mkfs

[Target]

5. # ./led-payer

E. led

[Host]

0. # cd ~/tiny4412/day4/linux_3.5.0_tiny4412/
1. # make menuconfig
Device Drivers --->
Character devices --->
<*> LED Support for FriendlyARM Tiny4412 GPIO LEDs
2. # cd led
3. # make
4. # cp -a ./led ~/tiny4412/experiment/root_mkfs

[Target]

5. # ./led

F. pwm

[Host]

0. # cd ~/tiny4412/day4/linux_3.5.0_tiny4412/
1. # make menuconfig
   Device Drivers --->
   Character devices --->
   <*> Buzzer driver for FriendlyARM Tiny4412 development boards
2. # cd pwm/
3. # make
4. # cp -a ./pwm ~/tiny4412/experiment/root_mkfs

[Target]

5. # ./pwm

Application Exercise

FFMPEG

1 source ~/tiny4412/experiment/set_arm_4412_toolchain_path.sh
2 cd ~/tiny4412/experiment/day4/3-application-exerise/ffmpeg
3 wget http://ffmpeg.org/releases/ffmpeg-2.8.3.tar.bz2
4 tar -xvjf ffmpeg-2.8.3.tar.bz2
5 cd ffmpeg-2.8.3

==========

Configure ffmpeg

==========

6.

./configure --cross-prefix=arm-none-linux-gnueabi- --enable-cross-compile --target-os=linux --cc=arm-linux-gcc --arch=arm --cpu=armv5te --prefix=/home/cadtc/tiny4412/experiment/root_mkfs/usr/local/ffmpeg --enable-shared --disable-static --enable-gpl --enable-ffmpeg --disable-ffplay --disable-ffserver --enable-swscale --enable-pthreads --disable-armv5te --disable-yasm --disable-stripping --enable-gpl --disable-network --enable-avfilter

==========

Build source

==========

7.

a) make -j4

* **[Target]**

b) mkdir usr/local/bin/ffmpeg

==========

install and test

==========

8. **[Host]**

make install

9.

[Target]

a)

export PATH=$PATH:/usr/local/bin/ffmpeg/bin

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/ffmpeg/lib

b) **TEST**
record video
ffmpeg -f video4linux2 -s 320x240 -r 50 -i /dev/video15 /slash.mpeg

libjpeg

[Host]

****
* **please check arm tool chain if setting or not** *
****

==========
libtool
==========

- Get Source code -
0. # wget ftp://ftp.gnu.org/gnu/libtool/libtool-2.4.tar.gz
1. # tar -xvzf libtool-2.4.tar.gz && cd libtool-2.4

- Configure -
2. # ./configure --host=arm-linux CC=arm-linux-gcc –prefix=/home/cadtc/tiny4412/experiment/day4/3-application-exerise/libjpeg/libtool-2.4/install

- Build -
3. # make -j4
4. # ls -l ./libtool

==========
libjpeg_build
==========

- Get Source code -
0. wget http://www.ijg.org/files/jpegsrc.v6b.tar.gz && tar -xvzf jpegsrc.v6b.tar.gz

- Configure -
1. # cd jpeg-6b

2. # ./configure --help

3. # cp ../libtool-2.4/libtool ./

4. # ./configure --host=arm-none-linux-gnueabi CC=arm-none-linux-gnueabi-gcc –prefix=/home/cadtc/tiny4412/experiment/day4/3-application-exerise/libjpeg/jpeg-6b/install --enable-shared

5. # geany ./Makefile

please search "LIBTOOL"

****
 # If using GNU libtool, LIBTOOL references it; if not, LIBTOOL is empty.
LIBTOOL=./libtool --tag=arm-none-linux-gnueabi-gcc
 # $(O) expands to "lo" if using libtool, plain "o" if not.
 # Similarly, $(A) expands to "la" or "a".
****

- Build Source -

6. make -j4

- Install -

5.
   a. make -n install
   b. mkdir -p ./install/bin && mkdir -p ./install/man/man1 && mkdir -p ./install/lib && mkdir -p ./install/include
   c. make install

7.
   sudo cp -a ./install/bin/* ~/tiny4412/experiment/root_mkfs/usr/bin
   sudo cp -a ./install/lib/* ~/tiny4412/experiment/root_mkfs/usr/lib

8. sudo cp ./testimg.jpg ~/tiny4412/experiment/root_mkfs

- Test in Target -

[Target]

0.
   export PATH:$PATH:/usr/lib
   export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib

1. djpeg -grayscale -bmp ./testimg.jpg > testimg.bmp

2. check slash.bmp from host pc tool

madpaly

==========

Get Source code

==========

1. wget http://www.mirrorservice.org/sites/downloads.sourceforge.net/m/ma/mad/madplay/0.15.2b/madplay-0.15.2b.tar.gz

==========

Get Source code

==========

2. tar -xvzf madplay-0.15.2b.tar.gz

==========

Enter source code

==========

3. cd madplay-0.15.2b/

==========

Configure Source code

==========

4. # ./configure --host=arm-linux CC=arm-none-linux-gnueabi-gcc \
--prefix=/home/cadtc/tiny4412/experiment/day4/3-application-exerise/madplay/madplay-0.15.2b/install

==========

Build Source code

==========

5. # make -j4

==========

Install

==========

6. # make -n install
   # make install
   # cp -a install/bin/maplay ~/tiny4412/experiment/root_mkfs/usr/bin

==========

TEST in target

==========

7. [Target]
   # madplay sample.mp3

mplayer

==========

1. Introduction

==========

http://www.mplayerhq.hu/design7/news.html
http://en.wikipedia.org/wiki/MPlayer

==========

2. decompress MPlayer-1.0rc4.tar.bz2

==========

tar -jxvf MPlayer-1.0rc4.tar.bz2 && cd MPlayer-1.0rc4/

==========

3. ./configure ,make and copy to Tiny4412

==========

a)

```
# ./configure --host-cc=gcc --cc=arm-none-linux-gnueabi-gcc \
--target=arm-linux --enable-fbdev --disable-dvdread --disable-live \
--disable-mp3lib --enable-mad --disable-win32dll --disable-mencoder \
--disable-ivtv --disable-dvdnav --disable-dvdread-internal \
--disable-libdvdcss-internal --enable-libavcodec_a --prefix=./install
```

b) # make -j8

c)

****

install: strip process terminated abnormally

将 config.mak 中 INSTALLSTRIP=-s，-s 去掉即可

****

make -n install

make install

d) sudo cp -a ./install/bin ~/tiny4412/experiment/root_mkfs/usr/bin

sudo cp -a ./install/lib ~/tiny4412/experiment/root_mkfs/usr/lib

==========

4. plug webcam

(sonix chip idVendor=0c45, idProduct=6340) into Tiny4412 and run "mplayer"

==========

[root@Tiny4412 /]#mplayer -vo fbdev -fps 15 tv:// -tv driver=v4l2:device=/dev/video15 -vf scale=800:480

*There should be frames on the screen

==========

5. play mpg file

==========

[root@Tiny4412 /]# ./mplayer -vo fbdev MELT.MPG -fs

[root@Tiny4412 /]# ./mplayer -vo fbdev bugs.avi

==========

6. play raw file

(you can catch RAW file with luvcview pleaser refer to NOTE_luvcview)

==========

[root@Tiny4412 /]# ./mplayer -vo fbdev -demuxer rawvideo -rawvideo w=320:h=240:format=yuy2 stream.raw -loop 0

*for furthre user guide refer to http://www.mplayerhq.hu/design7/documentation.html
*Here are the webcam qualified that be able to capture Image/Video with this mplayer
->logitech UVC Camera (046d:080f),UVC
->sonix USB 2.0 Camera (0c45:6340),UVC

FBV

==========

0. Introduction

==========

A graphic file viewer for the Linux framebuffer device

==========

1. decompress "fbv-0.99.tar.gz"

==========

tar xvf fbv-1.0b.tar.gz

==========

3. Configure and build source code

==========

cd fbv-1.0b
./configure --without-libungif --without-libpng
cp -a ../../libjpeg/jpeg-6b/install/lib/* ./

==========

4. Modify Makefile

==========

geany ./Makefile

a) add below in Makefile
CFLAGS = -O2 -Wall -D_GNU_SOURCE -I/home/cadtc/tiny4412/experiment/day4/3-application-exerise/panel_screen/fbv-1.0b/

b) add below in Makefile

LIBS = -ljpeg

c) build code

make

d) Copy execute file to target

cp -a ./fbv ~/tiny4412/experiment/root_mkfs

??

5. download, compile, make

libpng-1.4.1.tar.bz2 and libungif-4.1.4.tar.gz

??

$ tar xvf libpng-1.4.1.tar.bz2
$ cd ~/libpng-1.4.1
$ ./configure --host=arm-linux CC=arm-linux-gcc --prefix=./install
$ make -j4
$ make install

$ tar xvf libungif-4.1.4.tar.gz
$ cd ~/libungif-4.1.4
$ ./configure --host=arm-linux CC=arm-linux-gcc --prefix=./install
$ make -j4
$ make install

Include png/gif library by modifying this in " fbv Makefile "
LIBS = -ljpeg -lpng -lungif

$ cd ~/fbv-0.99
$ make

then "fbv" is generated in current directory and copy it/JPEG picture to ~/tiny4412/Day1/root_mkfs

==========

6 run ./fbv demo.jpg and bmp on Target

==========

./fbv testimg.jpg
./fbv testimg.bmp

Step 1 – Creation of the actual EXT4.img

dd if=/dev/zero of=rootfs_ext4.img bs=512k count=60

Translation of the terms,
bs =blocksize,
count=60, the number of block`s, in our case will result an image of 30 Mb.
To get the exact size of the image that you create use simple maths.
60 * 512K = 31457280 byte = 30M bytes

Step 2 Formating the rootfs_ext4.img with EXT4

mkfs.ext4 rootfs_ext4.img

It will be a question where you will select yes (Y)

Step 3 mount the directories that we previous created.

mkdir rootfs_ext4 && mount -o loop rootfs_ext4.img rootfs_ext4/

Step 4 copy the content from the old system.img in the system_new.img

cp -v -r -p /home/cadtc/tiny4412/experiment/root_mkfs/* ./rootfs_ext4

Step 5 sync the files

sync

Step 6 Unmounting the partitons.

umount rootfs_ext4/

Step 7 reboot EVB and into Linux console with NFS

Step 8 dd rootfs_ext4.img to /dev/mmcblk0p2

dd if=/tmp/rootfs_ext4.img of=/dev/mmcblk0p2

Step 9 Modify u-boot args then reboot

setenv bootargs 'noinitrd init=/linuxrc root=/dev/mmcblk0p2 rw noinitrd rootfstype=ext4 console=ttySAC0 lcd=S70'

Day 5 Build Basic rootfs

1. create necessary directgory

a) mkdir my_rootfs

b) mkdir -p bin dev etc home lib proc root sbin sys tmp usr var etc/init.d

c) copy rootfs_sample/* my_rootfs/etc
(fstab hostname inittab mdev.conf modules.conf mtab passwd profile init.d/rcS)

2. build busybox

a) wget http://www.busybox.net/downloads/busybox-1.23.0.tar.bz2

b) tar -xvjf busybox-1.23.0.tar.bz2

c) source set_arm_4412_toolchain_path.sh

d) make menuconfig

e) make -j4

f) make install (intsall ./_install)

3. copy C library to rootfs

a) cp -a FriendlyARM/toolschain/4.5.1/arm-none-linux-gnueabi/sys-root/lib my_rootfs

4. change u-boot parameters about rootfs

a) setenv bootargs noinitrd init=/linuxrc root=/dev/nfs ip=192.168.0.20:192.168.0.10:192.168.0.1:255.255.255.0::eth0:on nfsroot=192.168.0.10:**/home/xlloss/work/tiny-4412/build/my_rootfs**, ip=192.168.0.20 console=ttySAC0 lcd=S70

5. create device note

a) sudo mknod console c 5 1

b) sudo mknod null c 1 3

Use Buildroot to create rootfs

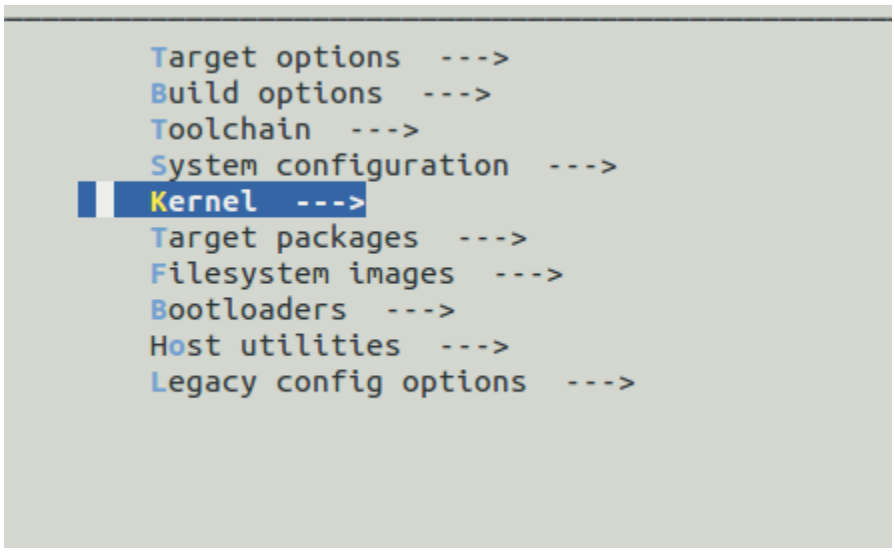
a) wget http://buildroot.uclibc.org/downloads/buildroot-2015.08.tar.gz

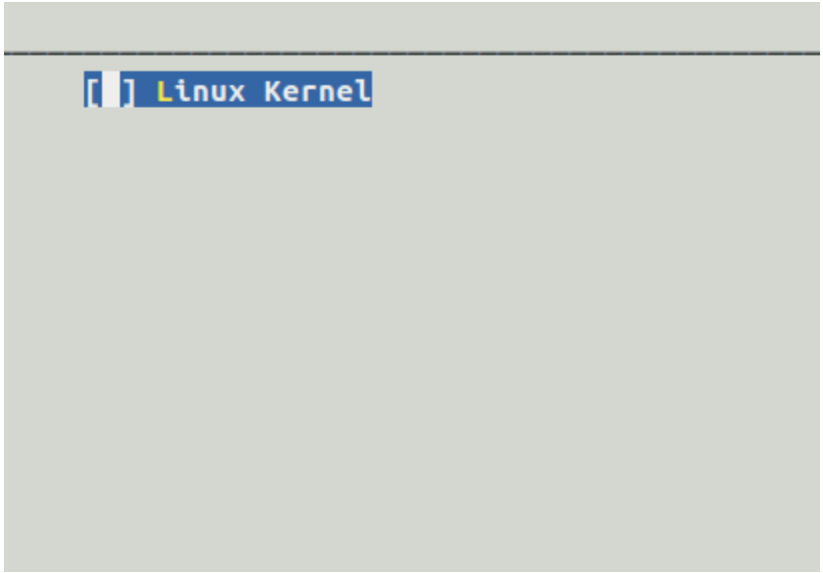
b) tar -xvzf buildroot-2015.08.tar.gz && cd buildroot-2015.08

c) make mini2440_defconfig

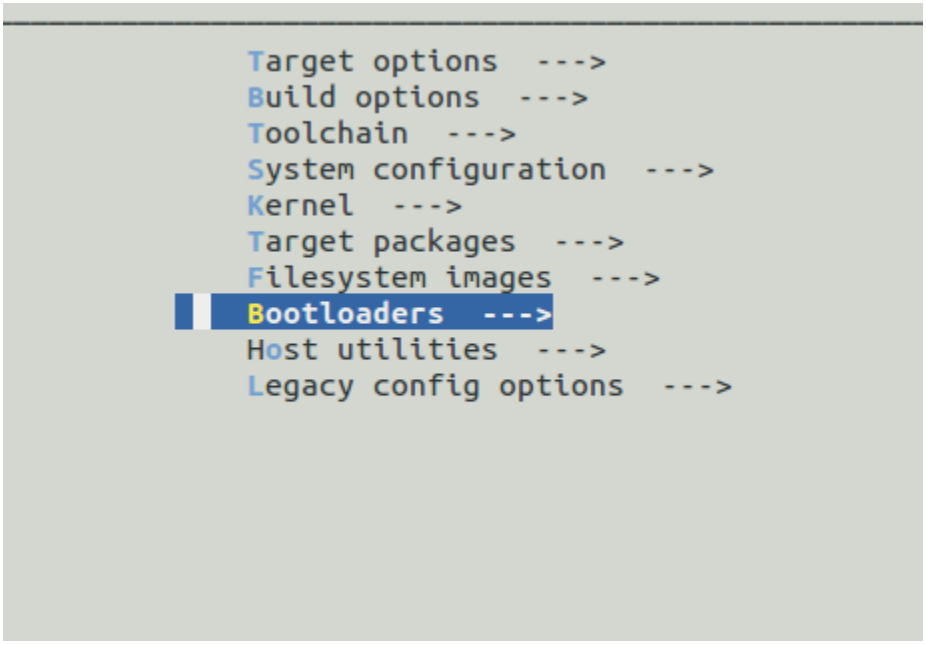
d) make menuconfig

e) cancel build kernel

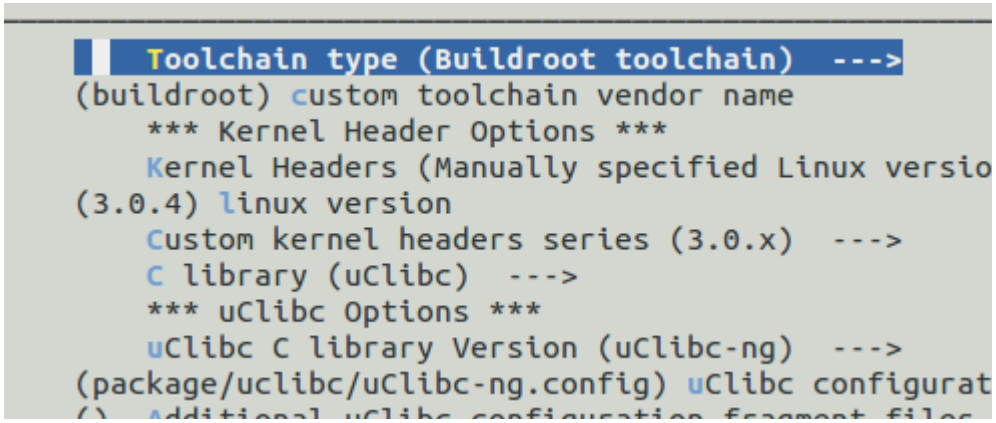


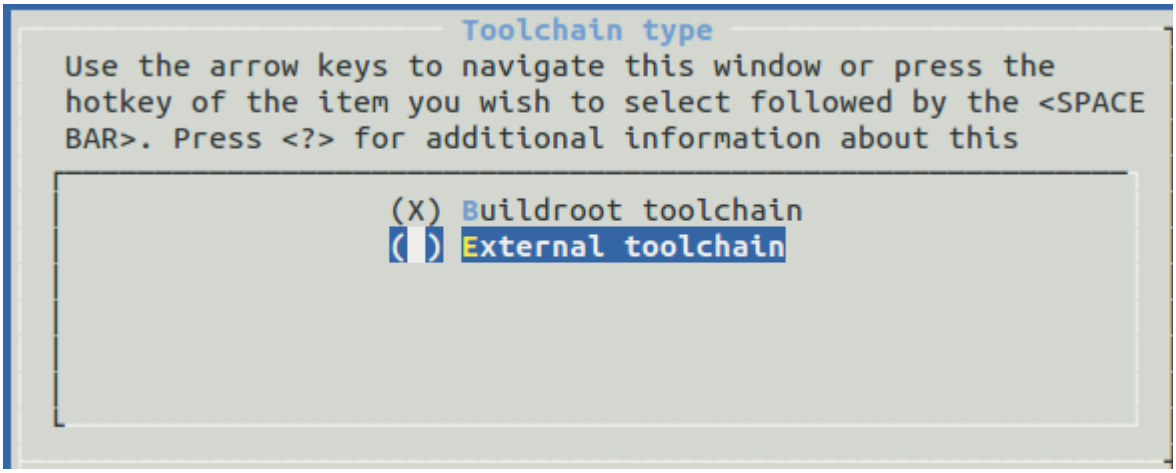


f) cancel buid bootloader

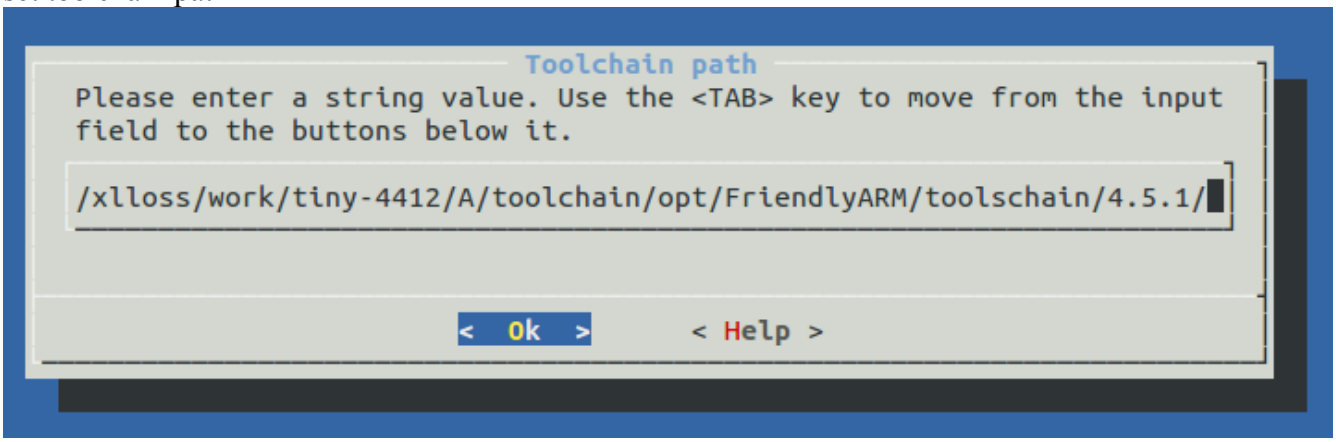


g) Toolchain setting

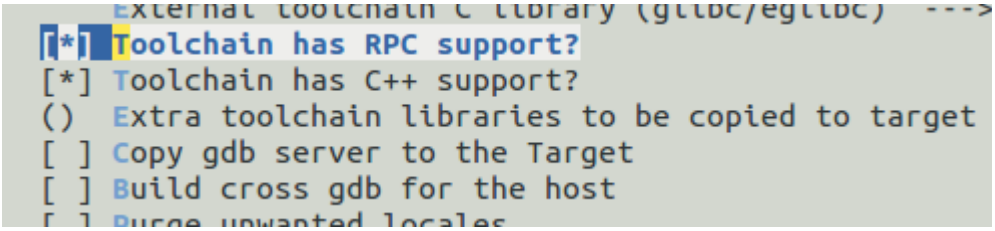




set toolchain path



h) set Toolchain has RPC support



i) build buildroot
#make

j) make finish, we enter output directory
cd output/images

k) extract rootfs.tar
mkdir my_rootfs && tar -xvf rootfs.tar -C my_rootfs/

l) modify etc/passwd file
sudo vim etc/passwd
-root:x:0:0:root:/root:/bin/sh
+root::0:0:root:/root:/bin/sh

m) add init process in etc/init.d
cd etc/init.d
sudo vim S00init, paste below :

###################### Start ##############################

```
#!/bin/sh
#
# Start init
#

start() {
        echo -n "Starting init: "
        /bin/hostname -F /etc/hostname

        /bin/mount -n -t proc none /proc
        /bin/mount -n -t sysfs none /sys
        /bin/mount -t ramfs none /dev

        echo /sbin/mdev > /proc/sys/kernel/hotplug
        /sbin/mdev -s
        /bin/hotplug
        # mounting file system specified in /etc/fstab
```

```
        mkdir -p /dev/pts
        mkdir -p /dev/shm
        /bin/mount -n -t devpts none /dev/pts -o mode=0622
        /bin/mount -n -t tmpfs tmpfs /dev/shm
        /bin/mount -n -t ramfs none /tmp
        /bin/mount -n -t ramfs none /var
        mkdir -p /var/empty
        mkdir -p /var/log
        mkdir -p /var/lock
        mkdir -p /var/run
        mkdir -p /var/tmp

        /bin/sh /etc/modules.conf

        /sbin/getty -L ttySAC0 115200 vt100

        echo "OK"
}

stop() {
        echo -n "Stopping logging: "
        start-stop-daemon -K -q -p /var/run/syslogd.pid
        start-stop-daemon -K -q -p /var/run/klogd.pid
        echo "OK"
}

case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  restart|reload)
        stop
        start
        ;;
  *)
        echo "Usage: $0 {start|stop|restart}"
        exit 1
esac
exit $?

###################### End ############################
```

n) # sudo 777 chmod S00init

['n and o ' option, please check NTFS_Setting]

o) update build_root rootfs to /etc/exports

p) restart nfs server

Day6 Linux driver module

[Host]

1. Setting build module
   # cd ~/tiny4412/experiment/linux_3.5.0_tiny4412
   # make menuconfig
   Device Drivers --->
   Character devices --->
   <M> Tiny4412 module sample

2. build module
   # cd ~/tiny4412/experiment/linux_3.5.0_tiny4412
   # make modules

3. set install path
   # make help | grep modules_install
   # export INSTALL_MOD_PATH=../
   # make modules_install
   # ls ../lib

4 Copy module to roofs
   # sudo cp -a ../lib/modules/ ~/tiny4412/experiment/root_mkfs/lib/

[Target]

5. # ls lib/modules/3.5.0-FriendlyARM/kernel/drivers/char/

6. # insmod lib/modules/3.5.0-FriendlyARM/kernel/drivers/char/tiny4412_hello_module.ko
   # lsmod

7. # rmmod tiny4412_hello_module
   # lsmod

8. # modprobe tiny4412_hello_module
   # lsmod

9. #modprobe -r tiny4412_hello_module
   # lsmod

example_char_dev

a.

[Host]

1. cd ~/tiny4412/experiment/day6/module/example_char_dev/led_driver
2. cp ./leds-tiny4412-char.c ~/tiny4412/experiment/linux_3.5.0_tiny4412/drivers/char/
3. cd ~/tiny4412/experiment/linux_3.5.0_tiny4412/drivers/char/
4. geany Kconfig

```
config SLASH_LED
        tristate "My LED Support for Tiny4412 LED"
```

5. geany Makefile

```
obj-$(CONFIG_SLASH_LED)    += leds-tiny4412-char.o
```

6.
   a. cd $(KERNEL_PATH)
   b. #make menuconfig
   c. #make modules

7. cp -a drivers/char/leds-tiny4412-char.ko ~/tiny4412/experiment/root_mkfs/lib/modules/3.5.0-FriendlyARM/kernel/drivers/char/

[Target]

8. depmod

9. modprobe leds-tiny4412-char

Application program

1. # cd ~/tiny4412/experiment/day6/module/example_char_dev/example_app/leds
2. # make
3. # sudo cp -a ./led ~/tiny4412/experiment/root_mkfs
4. # depmod -n
   # depmod

5. # modprobe leds-tiny4412-char

6. # led

example_build_out_of_kernel

[Host]

1. # cd ~/tiny4412/experiment/day6/module/example_build_out_of_kernel

2. # make

3. # cp ./leds-tiny4412_out_of_kernel.ko ~/tiny4412/experiment/root_mkfs/lib/modules/3.5.0-FriendlyARM/kernel/drivers/char/

[Target]

6. # depmod -n
   # depmod

7. # modprobe leds-tiny4412_out_of_kernel [install module]

8. # modprobe -r leds-tiny4412_out_of_kernel [remove module]