In your home direction(/home/xxx):

mkdir arm

cd arm

1. Serial port

minicom:

(1) install

sudo apt-get install minicom

or:

cd ../ForStudent/apt/

sudo dpkg -i minicom\_2.5-2\_i386.deb

minicom -v

(2) configuire

sudo minicom -s

--------------------------------------------

Serial port setup

A - Serial Device : /dev/ttyUSB0

F - Hardware Flow Control : No

Save setup as dfl

Exit from Minicom

--------------------------------------------

(3) run

ls -l /dev/ttyUSB0

sudo chmod 666 /dev/ttyUSB0

minicom

switch to NOR, power up board.

minicom

Device /dev/ttyUSB0 is locked.

rm /var/lock/LCK..ttyUSB0

2. Update Board System

cd ForStudent/MiniTools-20130513/

chmod 777 MiniTools

chmod 777 start.sh

sudo ./start.sh

裸机程序:

Install to Nand Flash

superboot: images/Superboot2440.bin

2440test\_W35.bin

![A description...](data:None;base64,)

Linux:

![A description...](data:None;base64,)

mini2440用户手册.pdf:

read and test:

1.1

1.2

2.4

***2.6***

**Target**:

[root@FriendlyARM /]#

**Host**:

xfliao@xfliao:~$

3. Linux Development Enviorment

(1) Network

watch sudo ifconfig eth0 192.168.0.1

Target:

ifconfig eth0 192.168.0.230

ping 192.168.0.1

(2) TFTP

Host:

(1) install

sudo apt-get install tftpd-hpa

sudo apt-get install tftp-hpa

or:

cd ../ForStudent/apt/

sudo dpkg -r tftp

sudo dpkg -r tftpd

sudo dpkg -i tftp-hpa\_5.2-1ubuntu1\_i386.deb

sudo dpkg -i tftpd-hpa\_5.2-1ubuntu1\_i386.deb

(2) configuire

sudo vim /etc/default/tftpd-hpa

--------------------------------------------

TFTP\_DIRECTORY="**/tftpboot**"

TFTP\_ADDRESS="0.0.0.0:69"

--------------------------------------------

or:

sudo vim /etc/xinetd.d/tftp

--------------------------------------------

server\_args = -s **/tftpboot**

flags = IPv4

--------------------------------------------

cd /

sudo mkdir tftpboot

sudo chmod 777 /tftpboot/

cd /tftpboot/

vi tftptest

(3) run

sudo service tftpd-hpa restart

tftpd-hpa start/running, process 11633

Or:

sudo /usr/sbin/in.tftpd restart

netstat -anu|grep 69

udp 0 0 0.0.0.0:69 0.0.0.0:\*

**Target**:

Network is good.

ifconfig eth0 192.168.0.230

ping 192.168.0.1

tftp -r tftptest -g 192.168.0.1

cat tftptest

4. Embed Linux Development

(1) Edit code

mkdir 02

cd 02/

vi hello.c

(2) Compile

gcc hello.c -o hello1

(3) **download** & Run

cp hello1 /tftpboot/

Host:

watch sudo ifconfig eth0 192.168.0.1

sudo service tftpd-hpa restart

Target:

ifconfig eth0 192.168.0.230

ping 192.168.0.1

tftp -r hello1 -g 192.168.0.1

ls -l hello1

chmod u+x hello1

./hello1

error reason:

gcc: Compiler which runs a x86 platform, compiles a executed program runing on X86.

Cross-Compiler (arm-linux-gcc):

Compiler which runs a x86 platform, compiles a executed program runing on ARM.

cd ..

cp ../ForStudent/Src/arm-linux-gcc-4.4.3.tar.gz .

tar zxvf arm-linux-gcc-4.4.3.tar.gz

cd 4.4.3/bin/

pwd

/home/xfliao/arm/4.4.3/bin

cd ~/

pwd

/home/xfliao

vi .bashrc

-----------------------------------------

export PATH=$PATH:/home/xfliao/arm/4.4.3/bin

-----------------------------------------

exit

**New a terminal.**

-----------------------------------------

64 bit OS:

14.04:

sudo apt-get install lib32z1

sudo apt-get install lib32stdc++6

------------------------------------------

arm-linux-gcc -v

cd arm/02/

arm-linux-gcc hello.c -o hello2

file hello1

file hello2

cp hello2 /tftpboot/

**Target**:

ifconfig eth0 192.168.0.230

ping 192.168.0.1

tftp -r hello2 -g 192.168.0.1

ls -l hello2

chmod u+x hello2

./hello2

5. Embed Linux Software

(1) Boot Loader

(2) Linux Kernel

(3) Root File System

BootLoader:

Initate hardware, loader OS.

Migrate bootloader (u-boot) to board:

pwd

/home/xfliao/arm

cp ~/ForStudent/Src/bootloader.tar .

tar xvf bootloader.tar

cd bootloader/u-boot/

make distclean

make mini2440\_config

make

ls -l u-boot.bin

Fire into board using MiniTools. (Switch to NOR)

cd ForStudent/MiniTools-20130513/

sudo ./start.sh

![A description...](data:None;base64,)

Low format NAND flash

Linux BootLoader:

/home/xfliao/arm/bootloader/u-boot/u-boot.bin

Switch to S2, Restart board,Hit any key.

[u-boot@MINI2440]#

NO OS, Only U-boot.

6. U-boot Input and Output

Target:

printenv

ping 192.168.0.1

tftp 0x30000000 hello2

go 0x30000000

WRONG!!!!

vi include/common.h

void printf(const char \*fmt, ...)

vi u-boot.map

0x33f963a8 printf

in arm/02:

mkdir 2\_hello

cd 2\_hello/

vi hello.c

vi Makefile

Target:

tftp 0x30000000 hello.bin

go 0x30000000

---------------------------------------------------

sudo chmod 666 /dev/ttyUSB0

minicom

watch sudo ifconfig eth0 192.168.0.1

sudo service tftpd-hpa restart

---------------------------------------------------

cp -r 2\_hello/ 3\_hello

cd 3\_hello/

vi include/common.h

int getc(void);

vi u-boot.map

0x33f965f0 getc

Target:

ping 192.168.0.1

tftp 0x30000000 hello.bin

go 0x30000000

7. ARM underlayer hardware development(裸机程序)

(1) analysis hardware [电路图](http://cn.bing.com/images/search?q=电路图&qpvt=电路图&qpvt=电路图&FORM=IGRE), make clear how it works.

mini2440原理图.pdf

蜂鸣器

MCU GPB0 pin: output mode

Output 0: buzzer off

Output 1: buzzer on

(2)refer chipset manual, make clear how to control hardware.

S3C2440.pdf

GPIO:general purple input/output

9. I/O Ports

PORT CONFIGURATION REGISTER GPBCON

GPBCON 0x56000010

GPB0 [1:0] =01

00 = Input

10 = TOUT0

01 = Output

11 = reserved

PORT DATA REGISTER GPBDAT

GPBDAT 0x56000014

data can be written to the corresponding bit of GPBDAT

GPB0 [0]

(3)program

define two variable which point to 0x56000010 and 0x56000014

variable x, n bit set 0:

x &= ~(0x01<<n)

xxxx xxxx

~ 0010 0000

& 1101 1111

-------------------------

xx0x xxxx

variable x, n bit set 1:

x |= (0x01<<n)

xxxx xxxx

| 0010 0000

-------------------------

xx1x xxxx

(4)run & test

===LED============================================

(1)

pin: Output mode

Led 1 2 3 4

GPB 5 6 7 8

output '0', led ON

output '1', led OFF

i) how to configure GPB5-8 as output mode.

ii) how to output high/low electronic level

(2)

i)GPBCON 0x56000010

![A description...](data:None;base64,)

01:Output mode

ii)GPBDAT 0x56000014

![A description...](data:None;base64,)

GPB[5] GPB[6] GPB[7] GPB[8]

========key===================================

(1)pin: Input mode

Key 1 2 3 4 5 6

GPG 0 3 5 6 7 11

Press: low level,'0'

Not Press: High level,'1'

i) how to configure GPGx as input mode.

ii) MCU how to read high/low electronic level

(2)

![A description...](data:None;base64,)

00:Input mode

![A description...](data:None;base64,)

![A description...](data:None;base64,)

![A description...](data:None;base64,)

When the port is configured as an input port, the corresponding bit is the

pin state

![A description...](data:None;base64,)

val, judge n bit 0 or 1?

val & (0x01<<n)

result bit

not 0 1

0 0

=========================================

8. Interrupt

Key 1 2 3 4 5 6

EINT 8 11 13 14 15 19

GPG 0 3 5 6 7 11

EINT8\_23: External interrupt 8 – 23

(1) SRCPND: (auto set 1, you must clear it after interruption handler)

![A description...](data:None;base64,)

![A description...](data:None;base64,)

bit 5: 1, clear it: SRCPND |= 0x01<<5

(2) INTMOD: configure bit 5 to 0.

![A description...](data:None;base64,)

![A description...](data:None;base64,)

(3) INTMSK: configure bit 5 to 0.

![A description...](data:None;base64,)

![A description...](data:None;base64,)

(4) INTPND (auto set 1, you must clear it after interruption handler))

![A description...](data:None;base64,)

![A description...](data:None;base64,)

After interrupt handler, clear bit 5: INTPND |= 0x01<<5

(5)INTOFFSET(euqal to 5)

![A description...](data:None;base64,)

![A description...](data:None;base64,)

External Interuption 8-23 configuration.

(1) EXTINT1 (set 000: low level mode)

![A description...](data:None;base64,)

![A description...](data:None;base64,)

![A description...](data:None;base64,)

![A description...](data:None;base64,)

EXTINT2

![A description...](data:None;base64,)

(2) EINTMASK (set to 0)

![A description...](data:None;base64,)

![A description...](data:None;base64,)

(3) EINTPEND (Auto set to 1, you must clear by |=0x01<<n)

![A description...](data:None;base64,)

![A description...](data:None;base64,)

(4)GPGCON (set to 10 as EINT mode)

![A description...](data:None;base64,)

![A description...](data:None;base64,)

CPSR 7 bit, set 0 to enable IRQ.

1. Migrate Linux Kernel to board.

Kernel: Operation system core part.

(1) Memory management

(2) process schedule

(3) process inter commucation

(4) Network

migration step:

(1) download and modify source code

(2) configure

produce .config file.

(3) compile

(4) download and fire into board

pwd

/home/xfliao/arm

cp ~/ForStudent/Src/linux-2.6.32.2.tar.gz .

ls -l

tar zxvf linux-2.6.32.2.tar.gz

cd linux-2.6.32.2/

du -sm

ls .config

make menuconfig

if error:

cd ~/ForStudent/apt/

sudo apt-get install libncurses5-dev

or:

sudo dpkg -i libncurses5-dev\_5.7+20101128-1\_i386.deb

if error happened in install:

sudo apt-get -f install

sudo apt-get update

after installation:

cd ~/arm/linux-2.6.32.2/

cp config\_mini2440\_w35 .config

make menuconfig

cp config\_mini2440\_w35 .config

or:

cp ~/ForStudent/images/config\_mini2440\_p35 .config

make zImage

Kernel: arch/arm/boot/zImage is ready

cp arch/arm/boot/zImage /tftpboot/

Board:

tftp 0x30008000 zImage

bootm 0x30008000

restart board,watch output:

(1)Failed to execute /linuxrc. Attempting defaults...

Kernel panic - not syncing: No init found. Try passing init.

printenv

bootcmd=nand read.i 0x30008000 60000 500000;bootm 0x30008000

tftp 0x30008000 zImage

nand erase 0x60000 0x500000

nand write 0x30008000 0x60000 0x500000

(2)Starting kernel ...

Uncompressing Linux........................................

Host:

mv /tftpboot/zImage /tftpboot/zImage.img

Connection USB to board, switch to NOR, start Minitools.

sudo ./start.sh

![A description...](data:None;base64,)

Switch to S2,restrt board.

2. Root File System.

File system: How file stores in harddisk/flash.

Windows:NTFS, FAT32

Linux: Ext4

Embed system: cramfs, jiffs/2:NOR Flash, yaffs/2: NAND Flash

Root File System: Basic, core programs in user space.

(1) system command

busybox

(2) configuration file.

/etc/passwd, /etc/shadow …

Orignal linux system /etc

(3)run library

Cross-Compiler

(4)GUI

(5)application program

cp ~/ForStudent/Src/mkyaffs2image.tgz .

tar xvf mkyaffs2image.tgz

sudo cp usr/sbin/mkyaffs2image-128M /usr/sbin/

sudo chmod 777 /usr/sbin/mkyaffs2image-128M

cp ~/ForStudent/Src/rootfs\_qtopia\_qt4-20120626.tar.gz .

sudo tar zxvf rootfs\_qtopia\_qt4-20120626.tar.gz

cd rootfs\_qtopia\_qt4/

du -sm

cd ..

sudo mkyaffs2image-128M rootfs\_qtopia\_qt4 rootfs.img

rm rootfs.img

cd rootfs\_qtopia\_qt4/

cd usr/local/

sudo rm -rf Trolltech/

sudo rm -f qtopia-titles-\*

cd ../../root/

sudo rm -rf Documents/

sudo vi testfile

sudo mkdir testdir

sudo cp ~/arm/02/1\_hello/hello2 .

sudo chmod 755 hello2

cd ../..

sudo mkyaffs2image-128M rootfs\_qtopia\_qt4 rootfs.img

ls -l

sudo chmod 666 rootfs.img

cp rootfs.img /tftpboot/

bootargs=noinitrd root=/dev/mtdblock3 init=/linuxrc console=ttySAC0

vi arch/arm/mach-s3c2440/mach-mini2440.c

friendly\_arm\_default\_nand\_part

[3] = {

.name = "root",

.offset = 0x00560000,

.size = 1024 \* 1024 \* 1024, //

},

length: EE00000

tftp 0x30008000 rootfs.img

--------------------------------------------------------

done

Bytes transferred = 56449536 (35d5a00 hex)

-----------------------------------------------------------

nand erase 0x56**0000** 0xee00000

nand write.yaffs 0x30008000 0x56**0000** 0x35d5a00

reset board, go to embed linux:

cd root/

3. Glare at liunx kernel

(1) Makefile (tree)

What files will be compiled.

How to compile them.

[]Not comile

[\*] built-in kernel

[M] module

Compile sequence.

ARCH ?= arm

CROSS\_COMPILE ?= arm-linux-

drivers-y := drivers/

obj-$(CONFIG\_BT) += bluetooth/

obj-y += net/

obj-$(CONFIG\_PHYLIB) += phy/

obj-$(CONFIG\_REALTEK\_PHY) += realtek.o

(2) .config

# CONFIG\_BT is not set

CONFIG\_WIRELESS=y

CONFIG\_CFG80211=m

menuconfig result: produce .config file.

(3)Kconfig (tree)

root: arch/arm/Kconfig

source "drivers/Kconfig"

source "drivers/net/Kconfig"

config GPIO\_SYSFS

bool "/sys/class/gpio/... (sysfs interface)"

config GPIO\_MAX732X

tristate "MAX7319, MAX7320-7327 I2C Port Expanders"

4. Add code into kernel.

1.

cd drivers/

mkdir mydriver

cd mydriver/

vi test.c

2. vi Kconfig

----------------

menu "my first driver"

config MY\_DRIVER

bool "Enable my first dirver bulit-in kerenl."

help

my first driver,which will be bulit-in kernel.

only print hello kernel string.

no other function.

endmenu

-----------------

3. vi Makefile

obj-$(CONFIG\_MY\_DRIVER) += test.o

4. cd ..

vi Makefile

obj-$(CONFIG\_MY\_DRIVER) += mydriver/

5. vi Kconfig

source "drivers/mydriver/Kconfig"

6.cd ..

cat .config |grep CONFIG\_MY\_DRIVER

make menuconfig

cat .config |grep CONFIG\_MY\_DRIVER

CONFIG\_MY\_DRIVER=y

make zImage

ls drivers/mydriver/

cp arch/arm/boot/zImage /tftpboot/

Board:

tftp 0x30008000 zImage

nand erase 0x60000 0x500000

nand write 0x30008000 0x60000 0x500000

5. Add system call into kernel.

cd drivers/

mkdir mysyscall

cd mysyscall/

vi syscalltest.c

vi Kconfig

-----------------------

menu "My first system call"

config MY\_SYSCALL

bool "Enable my syscall built-in kernel"

endmenu

-----------------------

vi Makefile

obj-$(CONFIG\_MY\_SYSCALL) += syscalltest.o

cd ..

vi Kconfig

source "drivers/mysyscall/Kconfig"

vi Makefile

obj-$(CONFIG\_MY\_SYSCALL) += mysyscall/

cd ..

vi arch/arm/include/asm/unistd.h

#define \_\_NR\_addup (\_\_NR\_SYSCALL\_BASE+365)

vi arch/arm/kernel/calls.S

/\* 365 \*/ CALL(sys\_addup)

cat .config|grep CONFIG\_MY\_SYSCALL

make menuconfig

cat .config|grep CONFIG\_MY\_SYSCALL

make zImage

ls drivers/mysyscall/

cp arch/arm/boot/zImage /tftpboot/

Board:

tftp 0x30008000 zImage

nand erase 0x60000 0x500000

nand write 0x30008000 0x60000 0x500000

restart board into linux.

arm-linux-gcc -o syscall\_test syscall\_test.c

cp syscall\_test /tftpboot/

Board:

ifconfig eth0 192.168.0.230

tftp -r syscall\_test -g 192.168.0.1

chmod u+x syscall\_test

./syscall\_test

6. Linux kernel module programming

Board:

ifconfig eth0 192.168.0.230

ping 192.168.0.1

tftp -r hello.ko -g 192.168.0.1

insmod hello.ko

lsmod

rmmod hello

7. Interrupt

irq:

vi arch/arm/mach-s3c2410/include/mach/irqs.h

irq flag:

vi include/linux/interrupt.h

irq return:

vi include/linux/irqreturn.h

8. Kernel thread:

1.创建内核线程:

struct task\_struct \*kthread\_create(int (\*threadfn)(void \*data),void \*data, const char namefmt[]);

2.唤醒内核线程(可以唤醒所有进程(线程)):

wake\_up\_process(struct task\_struct \*k);

3.创建并运行内核线程(3=1+2):

struct task\_struct \***kthread\_run**(int (\*threadfn)(void \*data),void \*data, const char namefmt[]);

通知内核线程停止:

int kthread\_stop(struct task\_struct \*k);

返回threadfn函数的返回值, 如果k没有被wake\_up\_process(k)过将返回-EINTR

检查是否收到停止信号:

int **kthread\_should\_stop**(void);