

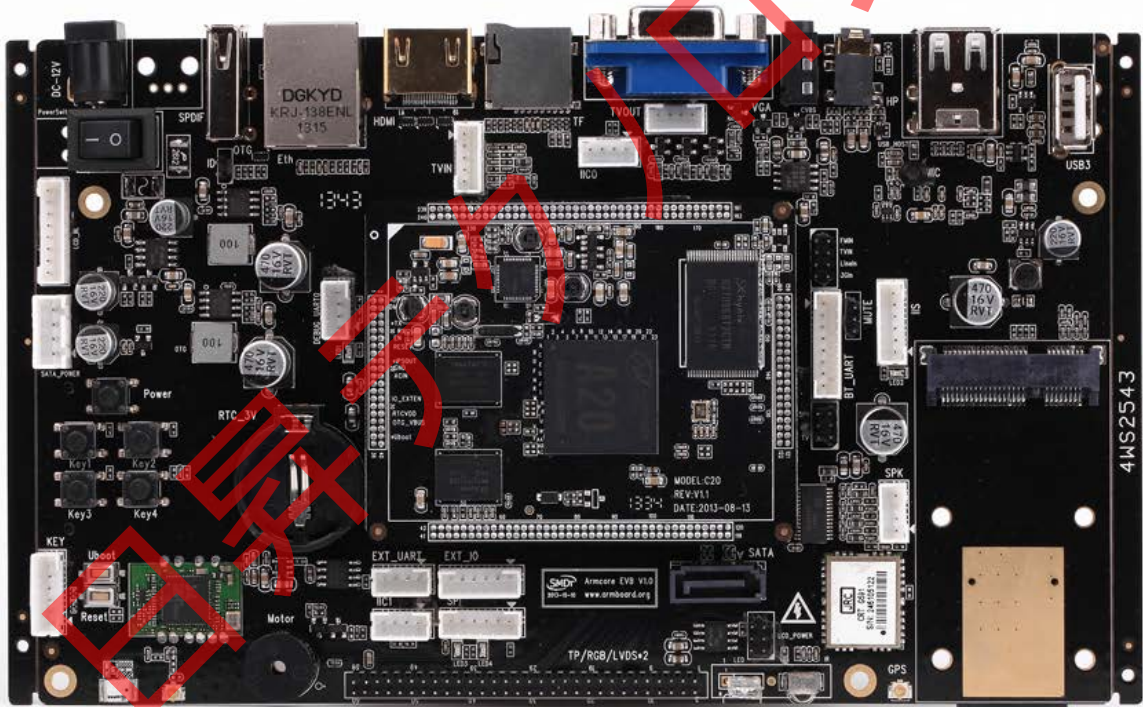
A20 Android4.2 マニュアル

株式会社日昇テクノロジー

<http://www.csun.co.jp>

info@csun.co.jp

作成日 2014/7/5



copyright@2014

・ 修正履歴

NO	バージョン	修正内容	修正日
1	Ver1.0	新規作成	2014/7/5

※ この文書の情報は、文書を改善するため、事前の通知なく変更されることがあります。最新版は弊社ホームページからご参照ください。「<http://www.csun.co.jp>」

※ (株)日昇テクノロジーの書面による許可のない複製は、いかなる形態においても厳重に禁じられています。

日昇テクノロジー

目次

第一章 Android 開発環境の構築.....	5
第二章 Android システムコンパイル.....	6
2.1 ソースコード構成説明.....	6
2.2 linux カーネルのコンパイル.....	6
2.3 Android4.2 ソースコードのコンパイル.....	9
2.4 イメージファイルをパッケージ化する.....	10
2.5 One キーでコンパイルとイメージファイルをパッケージ化.....	11
第三章 A20 システムイメージのダウンロード.....	15
第四章 A20 システムのカスタマイズ実例.....	16
4.1 sys_config.fex 設定ファイル.....	16
4.2 実例.....	16
4.2.1 HDMI 出力の表示フォーマットを設定.....	16
4.2.2 VGA 出力に設定.....	17
4.3 出力の表示方式を動的に変更する方法.....	17
4.3.1 display_param.cfg 設定ファイルを編集.....	17
4.3.2 ボードの TF スロットに挿し込み、パワーを入れる.....	18
4.4 設定情報.....	18
4.4.1 システム (system).....	18
4.4.2 TFRAM.....	21
4.4.3 GPU(mali).....	22
4.4.4 2D 加速 (G2D).....	22
4.4.5 ネットワーク (Ethernet MAC).....	22
4.4.6 I2C バス.....	24
4.4.7 シリアルポート (UART).....	24
4.4.8 SPI バス.....	28
4.4.9 抵抗スクリーン (rtp).....	30
4.4.10 静電スクリーン(capacitor tp).....	31
4.4.11 タッチキー (touch key).....	32
4.4.12 モーター (motor).....	33
4.4.13 フラッシュ (nand flash).....	33
4.4.14 表示を初期化 (disp init).....	34
4.4.15 LCD スクリーン0.....	36
4.4.16 LCD スクリーン1.....	40
4.4.17 カメラ (CSI).....	44
4.4.18 TV 出力 (TV OUT).....	50
4.4.19 SATA ディスク.....	50
4.4.20 TF/MMC.....	51
4.4.21 メモリスティック (memory stick).....	55
4.4.22 SIM カード.....	55
4.4.23 PS/2 マウス.....	56
4.4.24 CAN バス.....	56

4.4.25	マトリックスキーボード (key martrix)	57
4.4.26	USB コントロールフラグ	58
4.4.28	重力センサー (G SENSOR)	61
4.4.29	GPS	62
4.4.30	WIFI (TFIO)	62
4.4.31	WIFI (USB)	63
4.4.32	3G	63
4.4.33	gyroscope	64
4.4.33	光センサー	64
4.4.35	コンパス (compass)	65
4.4.36	bluetooth	65
4.4.37	デジタルオーディオバス (I2S)	65
4.4.38	デジタルオーディオバス (S/PDIF)	66
4.4.38	スピーカー制御	66
4.4.40	赤外線 (ir)	67
4.4.41	PMU パワー	67
第五章	Android アプリ環境の構築	72
5.1	Ubuntu で android アプリ環境を構築	72
5.1.1	JDK をインストール	72
5.1.2	eclipse をインストール	73
5.1.3	Android TFK をインストール	73
5.1.4	ADT (Android Development Tools) をインストール	74
5.1.5	ADT (Android Development Tools) を設定	74
5.1.6	Android OS システムとコントローラを追加	74
5.1.7	シミュレータ設定を行う	75
5.1.8	hello, android テストプログラム	75
5.2	windowsXP で android アプリ開発環境を構築	78

第一章 Android 開発環境の構築

手軽にAndroidシステム開発環境を構築できる様に、ubuntu-12.04-SmdtSDK-amd64.iso (付属DVDのimageフォルダ) を作成し提供している。このファイルはubuntu - 12.04 64bit OSシステムに基づき、事前にAndroid4.4システムの開発環境とツールをインストールした。

詳細は「A20_Android開発環境構築マニュアル.pdf」ファイルをご参考ください。

日昇テクノロジー

第二章 Android システムコンパイル

Androidソースコードコンパイルは二つの部分に分ける。一つはlinuxカーネルのコンパイル（ubootも含む）とAndroidソースコードコンパイルである。独自に機能としてコンパイルして、最後に書き込み用のイメージファイルにパッケージ化する。

2.1 ソースコード構成説明

ソースコードパッケージA20-420-V12_XXX.tar.bz2（XXXの部分はバージョン及び日付が付いている）を共用フォルダSMDT-A20のディレクトリにコピーして、解凍する。

```
smdt@smdt:~/SMDT-A20$ tar jxvf A20-420-V12.tar.bz2

smdt@smdt:~/SMDT-A20$ ls

A20-420-V12  A20-420-V12.tar.bz2

smdt@smdt:~/SMDT-A20$ cd A20-420-V12/

smdt@smdt:~/SMDT-A20/A20-420-V12$ ls

android4.2  build.sh  lichee  release_a20_v1.2.sh
```

ソースコードは二つの部分があり、linuxカーネルソースコードはlichee、もう一つはAndroid4.2のソースコードである。注意：この二つ部分のソースコードは同じディレクトリに保存しなければならない。

2.2 linux カーネルのコンパイル

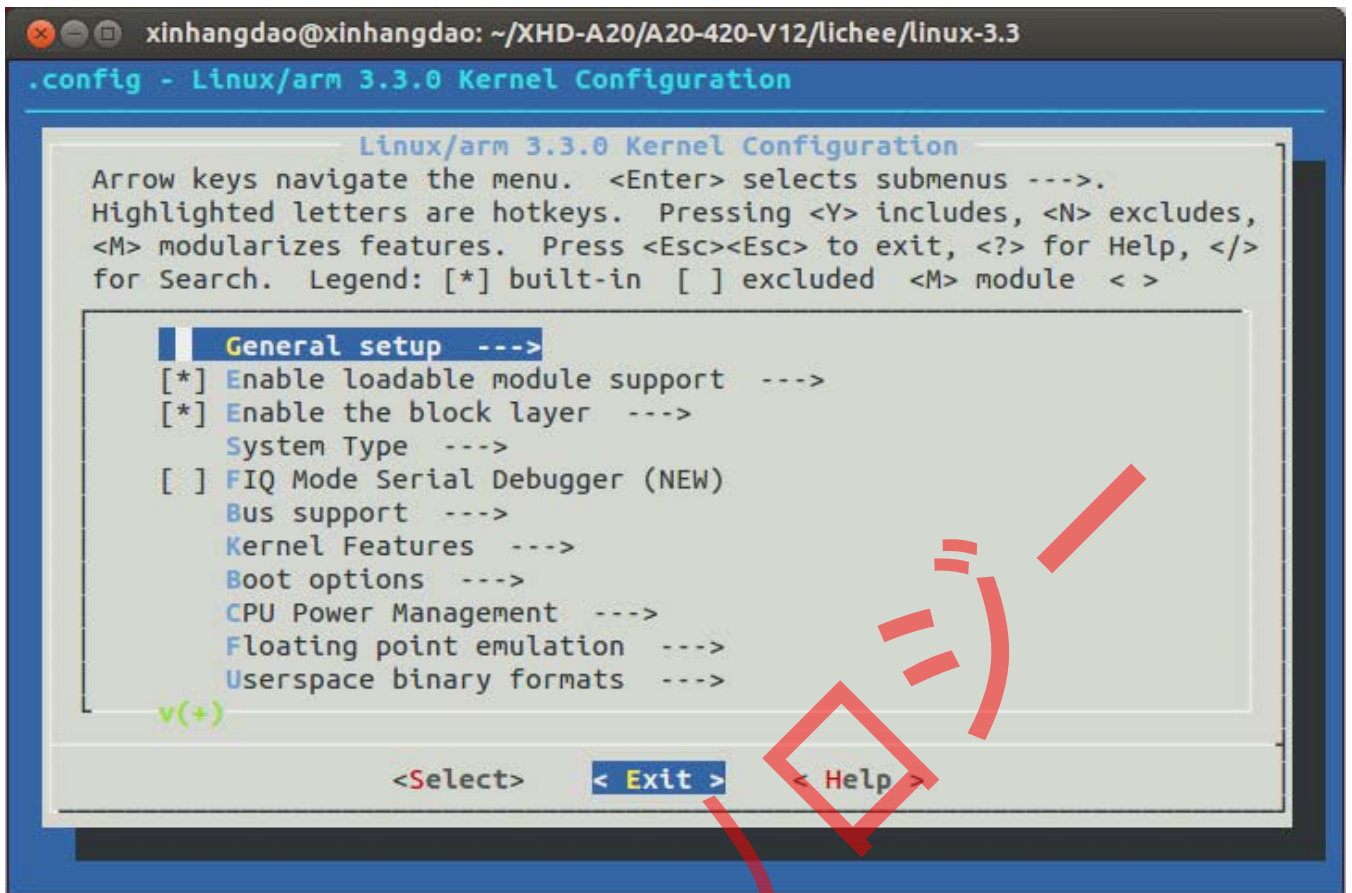
Step1 :

Lichee/linux-3.3ディレクトリに入って、make ARCH=arm menuconfig操作を行う。

```
smdt@smdt:~/SMDT-A20/A20-420-V12$ cd lichee/linux-3.3/

smdt@smdt:~/SMDT-A20/A20-420-V12/lichee/linux-3.3$ make ARCH=arm menuconfig
```

設定はデフォルトでarmパラメータ、コマンドを実行したあと、次のインタフェースが現れる。



exitを選択し、現在の設定を保存してから終了。

Step2：一回目のコンパイルの場合に、次の操作を行う。

```
smdt@smdt:~/SMDT-A20/A20-420-V12/lichee/linux-3.3$make clean
```

もし、リビルド又は一回目のコンパイルではない場合に、この操作は必要ない。

Step3：

```
smdt@smdt:~/SMDT-A20/A20-420-V12/lichee/linux-3.3$cd ..
smdt@smdt:~/SMDT-A20/A20-420-V12/lichee$rm linux-3.3/.config
smdt@smdt:~/SMDT-A20/A20-420-V12/lichee$./build.sh -p sun7i_android
```

コンパイルする。下図の通りに

```
xinhangdao@xinhangdao:~/XHD-A20/A20-420-V12/lichee$
xinhangdao@xinhangdao:~/XHD-A20/A20-420-V12/lichee$
xinhangdao@xinhangdao:~/XHD-A20/A20-420-V12/lichee$ ./build.sh -p sun7i_android

mkscript current setting:
  chip: sun7i
  Platform: android
  Board:
  output Dir: /home/xinhangdao/XHD-A20/A20-420-V12/lichee/out/android/common

INFO: build lichee ...
INFO: build buildroot ...
external toolchain has been installed
INFO: build buildroot OK.
INFO: build kernel ...
INFO: prepare toolchain ...
building kernel

using default config ...

build standby
Generating autoconf.h for standby
scripts/kconfig/conf --silentoldconfig kconfig
```

コンパイル成功後：

```
xinhangdao@xinhangdao: ~/a20/A20-420-V12/lichee
s/bios_emulator/libatlibiosenu.o drivers/block/libblock.o drivers/dma/libdma.o drivers/fpga/libfpga.o d
drivers/gpio/libgpio.o drivers/hwmon/libhwmon.o drivers/tzic/libtzic.o drivers/input/libinput.o drivers/n
isc/libnisc.o drivers/nmc/libnmc.o drivers/mtd/libmtd.o drivers/mtd/nand/libnand.o drivers/mtd/onenand
/libonenand.o drivers/mtd/spi/libspi_flash.o drivers/mtd/ubi/libubi.o drivers/net/libnet.o drivers/net
/phy/libphy.o drivers/pci/libpci.o drivers/pcnclia/libpcnclia.o drivers/power/libpower.o drivers/rtc/lib
rtc.o drivers/serial/libserial.o drivers/spi/libspi.o drivers/storage_type/libstorage_type.o drivers/t
wserial/libtw.o drivers/usb/eth/libusb_eth.o drivers/usb/gadget/libusb_gadget.o drivers/usb/host/libu
sb_host.o drivers/usb/musb/libusb_musb.o drivers/usb/phy/libusb_phy.o drivers/video/libvideo.o drivers
/watchdog/libwatchdog.o fs/cranfs/libcranfs.o fs/ext2/libext2fs.o fs/fat/libfat.o fs/fdos/libfdos.o fs
/jffs2/libjffs2.o fs/reiserfs/libreiserfs.o fs/ubifs/libubifs.o fs/yaffs2/libyaffs2.o lib/libfdt/libfd
t.o lib/libgeneric.o lib/lzma/liblzma.o lib/lzo/liblzo.o lib/zlib/libz.o nand_sunxi/libnand net/libnet
.o post/libpost.o board/allwinner/sun7i-evb/libsun7i-evb.o --end-group /home/xinhangdao/a20/A20-420-V1
2/lichee/u-boot/arch/arm/lib/eabi_compat.o -L /home/xinhangdao/a20/A20-420-V12/lichee/out/android/com
mon/buildroot/external-toolchain/bin/./lib/gcc/arm-linux-gnueabi/4.6.3 -lgcc -Map u-boot.map -o u-boo
t
arm-linux-gnueabi-objcopy -O srec u-boot u-boot.srec
arm-linux-gnueabi-objcopy --gap-fill=0xff -O binary u-boot u-boot.bin
make[1]: Leaving directory `/home/xinhangdao/a20/A20-420-V12/lichee/u-boot'
INFO: build u-boot OK.
INFO: build rootfs ...
INFO: skip make rootfs for android
INFO: build rootfs OK.
INFO: build lichee OK.
xinhangdao@xinhangdao:~/a20/A20-420-V12/lichee$
```

この時、lichee/out にカーネルイメージと boot イメージを生成する。

```
xinhangdao@xinhangdao: ~/a20/A20-420-V12/lichee
isc/libnisc.o drivers/nmc/libnmc.o drivers/mtd/libmtd.o drivers/mtd/nand/libnand.o drivers/mtd/onenand
/libonenand.o drivers/mtd/spi/libspi_flash.o drivers/mtd/ubi/libubi.o drivers/net/libnet.o drivers/net
/phy/libphy.o drivers/pci/libpci.o drivers/pcnclia/libpcnclia.o drivers/power/libpower.o drivers/rtc/lib
rtc.o drivers/serial/libserial.o drivers/spi/libspi.o drivers/storage_type/libstorage_type.o drivers/t
wserial/libtw.o drivers/usb/eth/libusb_eth.o drivers/usb/gadget/libusb_gadget.o drivers/usb/host/libu
sb_host.o drivers/usb/musb/libusb_musb.o drivers/usb/phy/libusb_phy.o drivers/video/libvideo.o drivers
/watchdog/libwatchdog.o fs/cranfs/libcranfs.o fs/ext2/libext2fs.o fs/fat/libfat.o fs/fdos/libfdos.o fs
/jffs2/libjffs2.o fs/reiserfs/libreiserfs.o fs/ubifs/libubifs.o fs/yaffs2/libyaffs2.o lib/libfdt/libfd
t.o lib/libgeneric.o lib/lzma/liblzma.o lib/lzo/liblzo.o lib/zlib/libz.o nand_sunxi/libnand net/libnet
.o post/libpost.o board/allwinner/sun7i-evb/libsun7i-evb.o --end-group /home/xinhangdao/a20/A20-420-V1
2/lichee/u-boot/arch/arm/lib/eabi_compat.o -L /home/xinhangdao/a20/A20-420-V12/lichee/out/android/com
mon/buildroot/external-toolchain/bin/./lib/gcc/arm-linux-gnueabi/4.6.3 -lgcc -Map u-boot.map -o u-boo
t
arm-linux-gnueabi-objcopy -O srec u-boot u-boot.srec
arm-linux-gnueabi-objcopy --gap-fill=0xff -O binary u-boot u-boot.bin
make[1]: Leaving directory `/home/xinhangdao/a20/A20-420-V12/lichee/u-boot'
INFO: build u-boot OK.
INFO: build rootfs ...
INFO: skip make rootfs for android
INFO: build rootfs OK.
INFO: build lichee OK.
xinhangdao@xinhangdao:~/a20/A20-420-V12/lichee$ ls
boot buildroot build.sh linux-3.3 out README tools u-boot
xinhangdao@xinhangdao:~/a20/A20-420-V12/lichee$
```


2.3 Android4.2 ソースコードのコンパイル

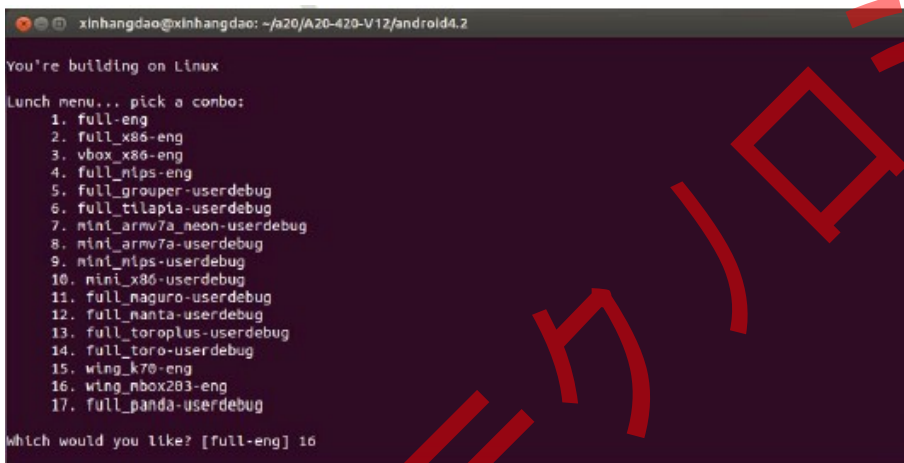
Android4.2ディレクトリに入って、スクリプトを実行する。

```
smdt@smdt:~/SMDT-A20/A20-420-V12$ cd android4.2/  
smdt@smdt:~/SMDT-A20/A20-420-V12/android4.2$ source build/envsetup.sh
```

そして lunch

```
smdt@smdt:~/SMDT-A20/A20-420-V12/android4.2$ lunch
```

次のように選択メニューを表示する。



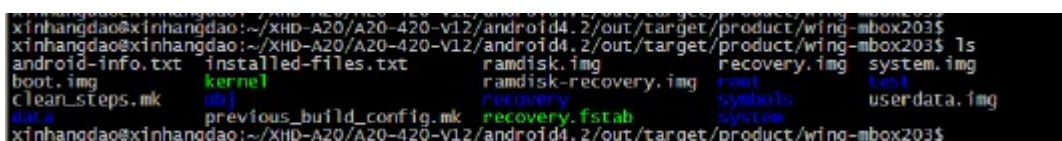
```
xinhangdao@xinhangdao: ~/A20/A20-420-V12/android4.2  
You're building on Linux  
Lunch menu... pick a combo:  
1. full-eng  
2. full_x86-eng  
3. vbox_x86-eng  
4. full_nips-eng  
5. full_grouper-userdebug  
6. full_tilapia-userdebug  
7. nini_armv7a_neon-userdebug  
8. nini_armv7a-userdebug  
9. nini_nips-userdebug  
10. nini_x86-userdebug  
11. full_naguro-userdebug  
12. full_manta-userdebug  
13. full_toroplus-userdebug  
14. full_toro-userdebug  
15. wing_k70-eng  
16. wing_mbox203-eng  
17. full_panda-userdebug  
Which would you like? [full-eng] 16
```

数字16を入力してEnterを押す。次のコマンドを実行する。

```
smdt@smdt:~/SMDT-A20/A20-420-V12/android4.2$ extract_bsp  
smdt@smdt:~/SMDT-A20/A20-420-V12/android4.2$ make
```

Make 時間は約2時間がかかる (i5 4コア 8G メモリの場合)。最後に、Android4.2ディレクトリに out ディレクトリを生成する。

最後に boot.img、recovery.img、system.img 三つのイメージファイルを生成する。



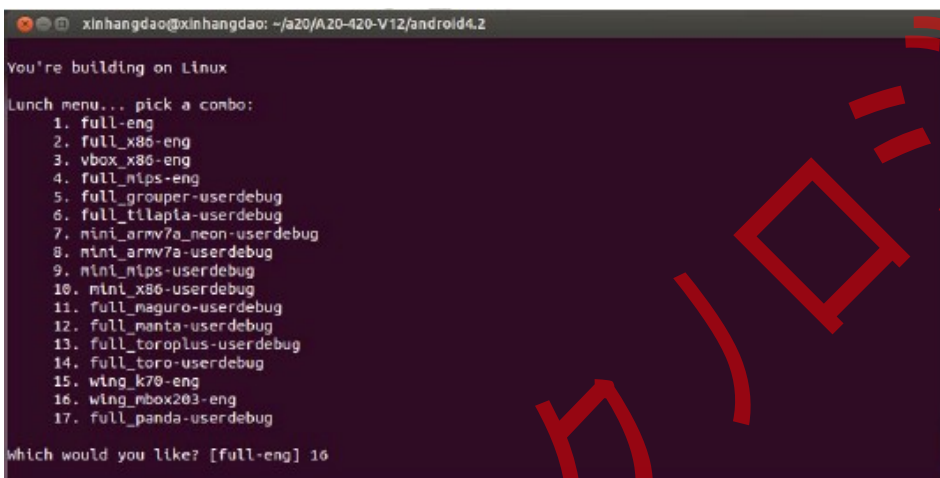
```
xinhangdao@xinhangdao: ~/XHD-A20/A20-420-V12/android4.2/out/target/product/wing-mbox203$  
xinhangdao@xinhangdao: ~/XHD-A20/A20-420-V12/android4.2/out/target/product/wing-mbox203$ ls  
android-info.txt  installed-files.txt  ramdisk.img  recovery.img  system.img  
boot.img         kernel                ramdisk-recovery.img  root         test  
clean_steps.mk  obj                  recovery      symbols      userdata.img  
data            previous_build_config.mk  recovery.fstab  system  
xinhangdao@xinhangdao: ~/XHD-A20/A20-420-V12/android4.2/out/target/product/wing-mbox203$
```

2.4 イメージファイルをパッケージ化する

最終に書き込みイメージファイルをパッケージ化する。Android4.2ディレクトリで直接 pack する。もし単独でパッケージする場合は、次の手順で行う。

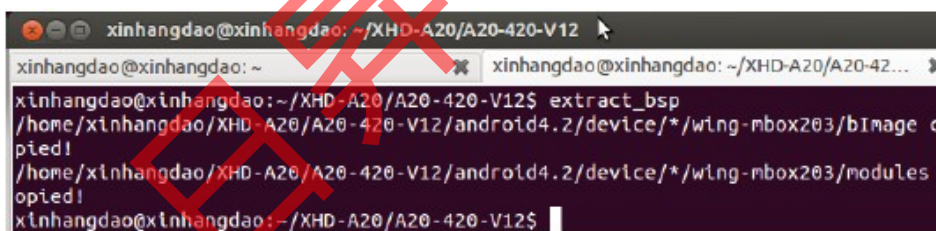
```
smdt@smdt:~/SMDT-A20/A20-420-V12/android4.2$ source build/envsetup.sh  
  
smdt@smdt:~/SMDT-A20/A20-420-V12/android4.2$ lunch
```

Lunch の表示：



```
xinhangdao@xinhangdao: ~/a20/A20-420-V12/android4.2  
You're building on Linux  
Lunch menu... pick a combo:  
1. full-eng  
2. full_x86-eng  
3. vbox_x86-eng  
4. full_nips-eng  
5. full_grouper-userdebug  
6. full_tllapta-userdebug  
7. nini_armv7a-neon-userdebug  
8. nini_armv7a-userdebug  
9. nini_nips-userdebug  
10. nini_x86-userdebug  
11. full_maguro-userdebug  
12. full_manta-userdebug  
13. full_toroplus-userdebug  
14. full_toro-userdebug  
15. wing_k70-eng  
16. wing_mbox203-eng  
17. full_panda-userdebug  
Which would you like? [full-eng] 16
```

```
smdt@smdt:~/SMDT-A20/A20-420-V12/android4.2$ extract_bsp
```



```
xinhangdao@xinhangdao: ~/XHD-A20/A20-420-V12  
xinhangdao@xinhangdao: ~/XHD-A20/A20-420-V12$ extract_bsp  
/home/xinhangdao/XHD-A20/A20-420-V12/android4.2/device/*/wing-mbox203/bImage copied!  
/home/xinhangdao/XHD-A20/A20-420-V12/android4.2/device/*/wing-mbox203/modules copied!  
xinhangdao@xinhangdao:~/XHD-A20/A20-420-V12$
```

パッケージして生成後の表示。Sun7i_android_wing - mbox203.img は最終に ROM に書き込むイメージである。

```
xinhangdao@xinhangdao: ~/XHD-A20/A20-420-V12/lichee/tools/pack
xinhangdao@xinhangdao: ~
update_boot1 boot1_nand.fex sys_config.bin NAND [Uncheck]
update_boot1 boot1_sdcard.fex sys_config.bin SDMMC_CARD [Uncheck]
fsbuild bootfs.ini split_xxxx.fex [OK]
dragon image.cfg sys_partition.fex [OK]
-----image is at-----
/home/xinhangdao/XHD-A20/A20-420-V12/lichee/tools/pack/sun7i_android_wing-mbox203.img
/home/xinhangdao/XHD-A20/A20-420-V12/android4.2
```

この時、書き込む用の **Sun7i_android_wing - mbox203.img** イメージファイルが生成した。

2.5 One キーでコンパイルとイメージファイルをパッケージ化

コンパイルとパッケージ化の手順をもっと簡易にできる様に、`build.sh` と `prj.sh` 二つのスクリプトを提供する。

まず、`build.sh` のソースコードを紹介する。

```
#!/bin/sh
#
# Description   : Android Build Script,
# Authors      : yxchen - cyx1203@qq.com
# Version      : 1.00
# Notes       : None
#
export ANDROID_JAVA_HOME=/usr/lib/jvm/java-6-sun/
BUILD_SCRIPT_TOP_DIR=$( cd $(dirname $0) ; pwd )
PROJECT=mbox203 #52
PROJECT=wing

CONFIG_KERNEL=sun71bsp_android_defconfig
CONFIG_FILESYSTEM_ANDROID=${PROJECT}_1-${PROJECT}-eng

ANDROID_SOURCE_TOPDIR=${BUILD_SCRIPT_TOP_DIR}/android4.2/
LICHEE_SOURCE_TOPDIR=${BUILD_SCRIPT_TOP_DIR}/lichee/

setup_environment()
{
    cd $(source_source_top_dir) || return 1
}

build_kernel()
{
    cd $(lichee_source_top_dir) || return 1
    rm lichee/linux-3.3/.config
    ./build.sh -p sun71_android
    return 0
}

build_system()
{
    cd $(android_source_top_dir) || return 1
    if [ -f $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/system/build.prop ];then
        echo "rm $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/system/build.prop"
        echo "rm $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/system/build.prop" success!!!!"
    else
        echo "File $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/system/build.prop not exist!!!!"
    fi
    source build/envsetup.sh
    lunch $(source_source_top_dir)/eng-mbox203
    extract_bsp
    make -j$(nproc) || return 1
    pack || return 1
}

pack_bootimg()
{
    cd $(source_source_top_dir) || return 1
    ./build.sh -p sun71_android
    source build/envsetup.sh
    lunch $(source_source_top_dir)/eng-mbox203
    extract_bsp
    make bootimage || return 1
    echo "pack boot.img ok!" >&2
}

build_ota_package()
{
    cd $(source_source_top_dir) || return 1
    rm $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/ota-*.zip
    source build/envsetup.sh
    lunch $(source_source_top_dir)/eng-mbox203
    get_aboot
    make otapackage -j$(nproc) || return 1
    cp -o $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/ota-*.zip $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/update.zip || exit 1;
    echo "" >&2
    echo "%s ota package have been build. the path is $(source_source_top_dir)/update.zip" >&2
    return 0
}

threads=4;
kernel=no;
system=no;

if [ -z $1 ]; then
    kernel=yes
    system=yes
fi

while [ " " ]; do
    case "$1" in
        -j=*)
            threads=${1#*=}
            ;;
        -p=*)
            project=${1#*=}
            echo "Build project: ${project} #eng > ./prj.sh"
            ;;
        -k|--kernel)
            kernel=yes
            ;;
        -s|--system)
            system=yes
            ;;
        -o|--ota)
            ota=yes
            ;;
        -b|--boot-image)
            bootimage=yes
            ;;
        -a|--all)
            kernel=yes
            system=yes
            ;;
        -h|--help)
            cat >&2 <<EOF
Usage: build.sh [options]
Build script for compile the source of telechips project.
-j=n          going n threads when building source project (example: -j=16)
-k, --kernel  build kernel from source file and using default config file
-s, --system  build file system from source file
-o, --ota     pack bootimg
no, --ota     build OTA packet
-p=          which project/plotform you build (example: -p=C20_C2)
-a, --all    build all, include anything
-h, --help   display this help and exit
EOF
            exit 0
            ;;
        *)
            echo "build.sh: unrecognized option '$1'" >&2
            exit 1
            ;;
    esac
    shift
done
```

```
#!/bin/sh
#
# Description   : Android Build Script,
# Authors      : yxchen - cyx1203@qq.com
# Version      : 1.00
# Notes       : None
#
export ANDROID_JAVA_HOME=/usr/lib/jvm/java-6-sun/
BUILD_SCRIPT_TOP_DIR=$( cd $(dirname $0) ; pwd )
PROJECT=mbox203 #52
PROJECT=wing

CONFIG_KERNEL=sun71bsp_android_defconfig
CONFIG_FILESYSTEM_ANDROID=${PROJECT}_1-${PROJECT}-eng

ANDROID_SOURCE_TOPDIR=${BUILD_SCRIPT_TOP_DIR}/android4.2/
LICHEE_SOURCE_TOPDIR=${BUILD_SCRIPT_TOP_DIR}/lichee/

setup_environment()
{
    cd $(source_source_top_dir) || return 1
}

build_kernel()
{
    cd $(lichee_source_top_dir) || return 1
    rm lichee/linux-3.3/.config
    ./build.sh -p sun71_android
    return 0
}

build_system()
{
    cd $(android_source_top_dir) || return 1
    if [ -f $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/system/build.prop ];then
        echo "rm $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/system/build.prop"
        echo "rm $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/system/build.prop" success!!!!"
    else
        echo "File $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/system/build.prop not exist!!!!"
    fi
    source build/envsetup.sh
    lunch $(source_source_top_dir)/eng-mbox203
    extract_bsp
    make -j$(nproc) || return 1
    pack || return 1
}

pack_bootimg()
{
    cd $(source_source_top_dir) || return 1
    ./build.sh -p sun71_android
    source build/envsetup.sh
    lunch $(source_source_top_dir)/eng-mbox203
    extract_bsp
    make bootimage || return 1
    echo "pack boot.img ok!" >&2
}

build_ota_package()
{
    cd $(source_source_top_dir) || return 1
    rm $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/ota-*.zip
    source build/envsetup.sh
    lunch $(source_source_top_dir)/eng-mbox203
    get_aboot
    make otapackage -j$(nproc) || return 1
    cp -o $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/ota-*.zip $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/update.zip || exit 1;
    echo "" >&2
    echo "%s ota package have been build. the path is $(source_source_top_dir)/update.zip" >&2
    return 0
}

threads=4;
kernel=no;
system=no;

if [ -z $1 ]; then
    kernel=yes
    system=yes
fi
```

```
#!/bin/sh
#
# Description   : Android Build Script,
# Authors      : yxchen - cyx1203@qq.com
# Version      : 1.00
# Notes       : None
#
export ANDROID_JAVA_HOME=/usr/lib/jvm/java-6-sun/
BUILD_SCRIPT_TOP_DIR=$( cd $(dirname $0) ; pwd )
PROJECT=mbox203 #52
PROJECT=wing

CONFIG_KERNEL=sun71bsp_android_defconfig
CONFIG_FILESYSTEM_ANDROID=${PROJECT}_1-${PROJECT}-eng

ANDROID_SOURCE_TOPDIR=${BUILD_SCRIPT_TOP_DIR}/android4.2/
LICHEE_SOURCE_TOPDIR=${BUILD_SCRIPT_TOP_DIR}/lichee/

setup_environment()
{
    cd $(source_source_top_dir) || return 1
}

build_kernel()
{
    cd $(lichee_source_top_dir) || return 1
    rm lichee/linux-3.3/.config
    ./build.sh -p sun71_android
    return 0
}

build_system()
{
    cd $(android_source_top_dir) || return 1
    if [ -f $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/system/build.prop ];then
        echo "rm $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/system/build.prop"
        echo "rm $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/system/build.prop" success!!!!"
    else
        echo "File $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/system/build.prop not exist!!!!"
    fi
    source build/envsetup.sh
    lunch $(source_source_top_dir)/eng-mbox203
    extract_bsp
    make -j$(nproc) || return 1
    pack || return 1
}

pack_bootimg()
{
    cd $(source_source_top_dir) || return 1
    ./build.sh -p sun71_android
    source build/envsetup.sh
    lunch $(source_source_top_dir)/eng-mbox203
    extract_bsp
    make bootimage || return 1
    echo "pack boot.img ok!" >&2
}

build_ota_package()
{
    cd $(source_source_top_dir) || return 1
    rm $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/ota-*.zip
    source build/envsetup.sh
    lunch $(source_source_top_dir)/eng-mbox203
    get_aboot
    make otapackage -j$(nproc) || return 1
    cp -o $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/ota-*.zip $(source_source_top_dir)/out/target/product/${PROJECT}-${PROJECT}/update.zip || exit 1;
    echo "" >&2
    echo "%s ota package have been build. the path is $(source_source_top_dir)/update.zip" >&2
    return 0
}

threads=4;
kernel=no;
system=no;

if [ -z $1 ]; then
    kernel=yes
    system=yes
fi

while [ " " ]; do
    case "$1" in
        -j=*)
            threads=${1#*=}
            ;;
        -p=*)
            project=${1#*=}
            echo "Build project: ${project} #eng > ./prj.sh"
            ;;
        -k|--kernel)
            kernel=yes
            ;;
        -s|--system)
            system=yes
            ;;
        -o|--ota)
            ota=yes
            ;;
        -b|--boot-image)
            bootimage=yes
            ;;
        -a|--all)
            kernel=yes
            system=yes
            ;;
        -h|--help)
            cat >&2 <<EOF
Usage: build.sh [options]
Build script for compile the source of telechips project.
-j=n          going n threads when building source project (example: -j=16)
-k, --kernel  build kernel from source file and using default config file
-s, --system  build file system from source file
-o, --ota     pack bootimg
no, --ota     build OTA packet
-p=          which project/plotform you build (example: -p=C20_C2)
-a, --all    build all, include anything
-h, --help   display this help and exit
EOF
            exit 0
            ;;
        *)
            echo "build.sh: unrecognized option '$1'" >&2
            exit 1
            ;;
    esac
    shift
done
```

```

if [ -f ./prj.sh ];then
if [ -x ./prj.sh ];then
echo ""
else
chmod 777 ./prj.sh
fi
source ./prj.sh
CONFIG_FILESYSTEM_ANDROID=wing_prj
else
if [ -z $1 ];then
echo "e.g : ./build.sh -a -pmbox203"
exit 1
fi
fi
if [ "${product_name_android}" = "wing_mbox203-eng" ];then
echo "e.g : ./build.sh -a -pmbox203"
elif [ "${product_name_android}" = "wing_kro-eng" ];then
echo "e.g : ./build.sh -a -pmbox203"
else
echo "product name err"
echo "e.g : ./build.sh -a -pmbox203"
exit 1
fi
setup_environment || exit 1
if [ "${kernel}" = yes ]; then
build_kernel || exit 1
fi
if [ "${system}" = yes ]; then
build_system || exit 1
fi
if [ "${booting}" = yes ]; then
pack_booting || exit 1
fi
if [ "${ota}" = yes ]; then
build_ota_package || exit 1
fi
exit 0

```

上述の操作はコンパイルのコマンドをこのスクリプトで一つずつ実現させる。

次に、prj.sh スクリプトを紹介する。次はソースコードである。

```

wing_prj=wing_mbox203-eng

```

一行だけの表現式である。目的は対応するプロジェクトを選択する。例えば、今 wing_mbox203 - eng を使っていて、build.sh は自動的にこのプロジェクトのソースコードをコンパイルする。

最後に、コンパイル方法を紹介する。

```

cyx@Server-102:~/a20_box/A20-420-V12$ ls
android4.2 build.sh 11chips prj.sh release_a20_v1.2.txt
cyx@Server-102:~/a20_box/A20-420-V12$ ./build.sh --help
Usage: build.sh [OPTION]
Build script for compile the source of telechips project.

-j=n          using n threads when building source project (example: -j=16)
-k, --kernel  build kernel from source file and using default config file
-s, --system  build file system from source file
-b, --booting pack booting
-o, --ota     build OTA packet
-p=n          which project/platform you build (example: -p=C20_CJ)
-a, --all     build all, include anything
-h, --help   display this help and exit
cyx@Server-102:~/a20_box/A20-420-V12$

```

直接に現在の sdk ソースコードパスで：

```

#. /build.sh -help

```

でコマンドの説明が見える。

もし単独にカーネルをコンパイルする場合に、

```

#. /build.sh -k

```

もし単独にファイルシステムをコンパイルする場合に、

```
#!/build.sh -s
```

もし全部コンパイルする場合に、

```
#!/build.sh -a
```

recovery モードアップグレードする場合に、

```
#!/build.sh -a (前に既に実行したら、省略できる)
```

```
#!/build.sh -o
```

一般的には、#!/build.sh -a を推奨する。

日昇テクノロジー

第三章 A20 システムイメージのダウンロード

付属 DVD の¥manual¥A20_install_manual.pdf をご参照ください。

日昇テクノロジー

第四章 A20 システムのカスタマイズ実例

4.1 sys_config.fex 設定ファイル

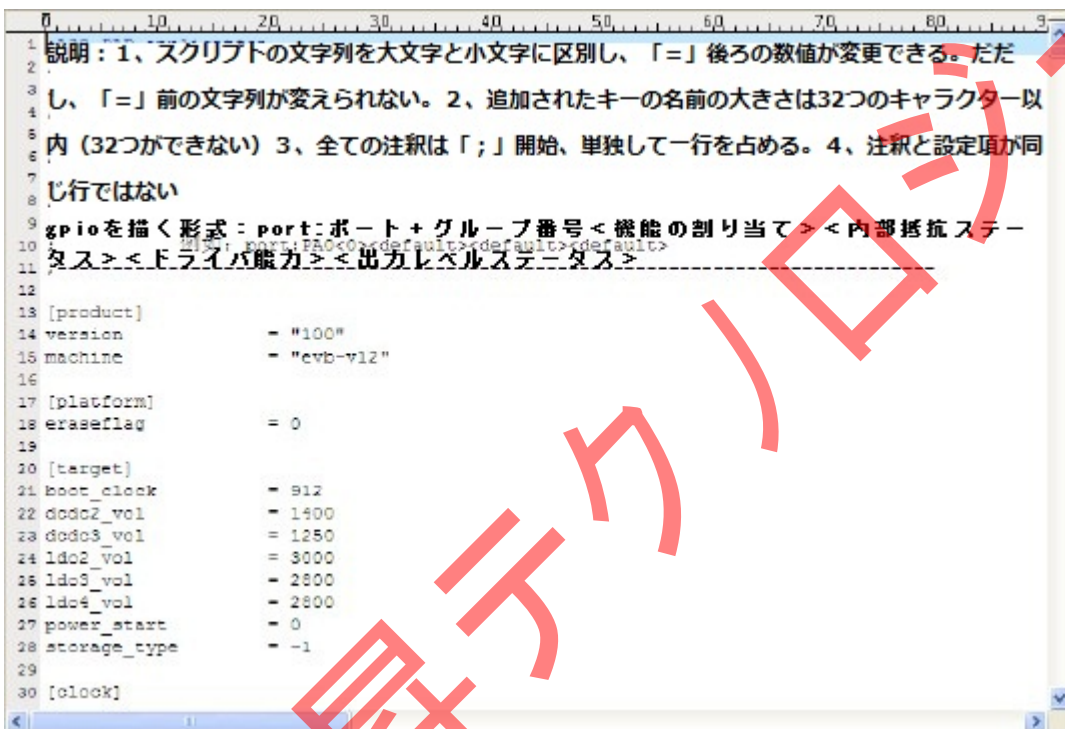
sys_config.fexはボードの関連機能の設定ファイルである。このファイルを修正することによって、各機能モジュールドライバのenable、close、及び他のパラメータを設定ができる。

ユーザーはハードウェアモジュールのニーズによって、設定と修正を行って、対応するドライバ機能を実現する。もしこれだけでまだニーズに合わない場合はICのdatasheet「A20 user manual V1.0 20130322.pdf」をご参照してカスタマイズが必要です。

sys_config.fexファイルのパス：

lichee/tools/pack/chips/sun7i/configs/android/wing-mbox203/

(実際のプロジェクト名によって違う。ここでwing-mbox203は本例のプロジェクト名である。)



```

1  説明：1、スクリプトの文字列を大文字と小文字に区別し、「=」後ろの数値が変更できる。ただ
2
3  し、「=」前の文字列が変えられない。2、追加されたキーの名前の大きさは32つのキャラクター以
4
5  内（32つができない）3、全ての注釈は「;」開始、単独して一行を占める。4、注釈と設定項が同
6
7  じ行ではない
8
9  gpioを描く形式：port:ポート+グループ番号<機能の割り当て><内部抵抗ステ
10
11  タス><ドライバ能力><出力レベルステータス>
12
13 [product]
14 version          = "100"
15 machine          = "cwb-v12"
16
17 [platform]
18 eraseflag       = 0
19
20 [target]
21 boot_clock      = 912
22 dcdc2_vol       = 1400
23 dcdc3_vol       = 1250
24 ldc2_vol        = 3000
25 ldc3_vol        = 2800
26 ldc4_vol        = 2800
27 power_start     = 0
28 storage_type    = -1
29
30 [clock]
  
```

4.2 実例

例を挙げて設定の修正方法を説明する。

4.2.1 HDMI 出力の表示フォーマットを設定

1、sys_config.fexファイルを編集

```
#cd lichee/tools/pack/chips/sun7i/configs/android/wing-mbox203
```

```
#vim sys_config.fex
```

2.修正

```
Screen0_output_type=xx
```

スクリーン0出力タイプ (0:none、1:lcd、2:tv、3:hDMI、4:vga)

Screen0_output_mode=xx スクリーン0出力モード (used for tv/hdmi output、0:480i 1:576i 2:480p
3:576p 4:720p50 5:720p60 6:1080i50 7:1080i60 8:1080p24
9:1080p50 10:1080p60 11:pal 14:ntsc)

Screen_output_type=3

Screen_output_mode=5

表示タイプはHDMI、750 P 60HZのフォーマットで出力する。

3. イメージファイルを再パッケージする。

Android4.2 ディレクトリに入って、以下の操作を実施し、新しいイメージファイルを生成する。

```
#source build/envsetup.sh
#lunch
#16 // (注意：この選択は自分のプロジェクトと関連する)
#extract_bsp
#pack
```

新生成したイメージファイルをダウンロードする。

4.2.2 VGA 出力に設定

1、HDMIと同じように設定する。

デフォルトの出力

```
screen0_output_type = 3
screen0_output_mode = 5
```

を screen0_output_type = 4
screen0_output_mode = 11

に変更する。

即 VGAタイプ、PALフォーマットで出力する。

他の手順はHDMIと同じ。

4.3 出力の表示方式を動的に変更する方法

実際には、表示デバイスを変更する時に、再びコンパイル、パッケージ、ダウンロードするのはとても複雑です。この問題を解決するために、出力の表示方式を動的に変更する方法を紹介する。

4.3.1 display_param.cfg 設定ファイルを編集

まず、要求通りに付属された資料のdisplay_param.cfgファイルを編集する。実際のニーズによって、screen0_output_typeとscreen0_output_modeを修正する。

詳細は次にご参考ください。もっと詳しい情報は付属DVDの¥manual¥表示設定方法及びCFGファイルをご参照ください。

Screen_output_type=xx スクリーン0出力種類 (0:none、1:lcd、2:tv、3:hdmi、4:vga)

Screen_output_mode=xx スクリーン0出力モード (used for tv/hdmi output、 0:480i 1:576i 2:480p
3:576p 4:720p50 5:720p60 6:1080i50 7:1080i60 8:1080p24
9:1080p50 10:1080p60 11:pal 14:ntsc)

4.3.2 ボードの TF スロットに挿し込み、パワーを入れる。

修正されたdisplay_param.cfgファイルをTFカードのルートディレクトリにコピーしてから、ボードにTFカードを挿し込み、パワーを入れる。この時、ボードはdisplay_param.cfg設定ファイルによって出力種類を選ぶ。同時に、今回の設定はnand flashに保存される。

4.4 設定情報

4.4.1 システム (system)

1. [Target]

備考：ブルーはモジュールチップのピン設定、ブラックはモジュール内部制御の設定である。

GPIO設定の形式：

PORT：ポート+グループ番号<機能の割り当て><内部抵抗のステータス><ドライバの能力><出力レベルのステータス>

設定アイテム	意味
boot_clock=xx	起動周波数、xxは〇〇MHz.
dcdc2_vol=1400	dcdc2の出力電圧、mV
dcdc3_vol=1250	dcdc3の出力電圧、mV
ldo2_vol=3000	ldo2の出力電圧、mV
ldo3_vol=2800	ldo3の出力電圧、mV
ldo4_vol=2800	ldo4の出力電圧、mV

設定の例：

```
[target]
boot_clock = 912
dcdc2_vol = 1400
dcdc3_vol = 1250
ldo2_vol = 3000
ldo3_vol = 2800
ldo4_vol = 2800
```

2. [card_boot]

設定アイテム	意味
Logical_start=xx	/
Sprite_gpio0=	/

設定の例：

```
[card_boot]
```

```
logical_start      = 40960
sprite_gpio0      =
```

3. [card0_boot_para]

設定アイテム	意味
card_ctrl=0	カードによる量産に関するコントローラ
card_high_speed=xx	スピードモード 0：低速、1：高速
Card_line=4	4線カード
TFc_cmd =xx	TFc コマンド信号の GPIO 設定
TFc_clk =xx	TFc カードクロック信号の GPIO 設定
TFc_d0 =xx	TFc カードデータ0線信号の GPIO 設定
TFc_d1 =xx	TFc カードデータ1線信号の GPIO 設定
TFc_d3 =xx	TFc カードデータ3線信号の GPIO 設定
TFc_d2 =xx	TFc カードデータ2線信号の GPIO 設定

設定の例：

```
card_ctrl = 2
card_high_speed = 1
card_line = 4
sdc_d1 = port:PF0<2><1><default><default>
sdc_d0 = port:PF1<2><1><default><default>
sdc_clk = port:PF2<2><1><default><default>
sdc_cmd = port:PF3<2><1><default><default>
sdc_d3 = port:PF4<2><1><default><default>
sdc_d2 = port:PF5<2><1><default><default>
```

4. [card2_boot_para]

設定アイテム	意味
card_ctrl=2	カード起動コントローラ
card_high_speed=xx	スピードモード；1：低速；2：高速
card_line=4	4線カード
TFc_cmd =xx	TFc コマンド信号の GPIO 設定
TFc_clk =xx	TFc カードクロック信号の GPIO 設定
TFc_d0 =xx	TFc カードデータ0線信号の GPIO 設定
TFc_d1 =xx	TFc カードデータ1線信号の GPIO 設定
TFc_d3=xx	TFc カードデータ3線信号の GPIO 設定
TFc_d2=xx	TFc カードデータ2線信号の GPIO 設定

設定の例：

```
card_ctrl= 2
card_high_speed= 1
card_line= 4
sdc_cmd= port:PC6<3><1>
sdc_clk= port:PC7<3><1>
sdc_d0= port:PC8<3><1>
```

```
sdc_d1= port:PC9<3><1>  
sdc_d2= port:PC10<3><1>  
sdc_d3= port:PC11<3><1>
```

5. [twi_para]

設定アイテム	意味
twi_port= xx	Boot の twi コントローラの番号
twi_scl=xx	Boot の twi のクロックの GPIO 設定
twi_TFa=xx	Boot の twi のデータの GPIO 設定

設定の例:

```
twi_port      = 0  
twi_scl      = port:PB0<2><default><default><default>  
twi_TFa     = port:PB1<2><default><default><default>
```

6. [uart_para]

設定アイテム	意味
uart_debug_port=xx	Boot シリアルポートのコントローラの番号
uart_debug_tx=xx	Boot シリアルポート送信の GPIO 設定
uart_debug_rx=xx	Boot シリアルポート受信の GPIO 設定

設定の例 :

```
uart_debug_port = 0  
uart_debug_tx  = port:PB22<2><1><default><default>  
uart_debug_rx  = port:PB23<2><1><default><default>
```

7. [jtag_para]

設定アイテム	意味
jtag_enable=xx	JTAG の enable
jtag_ms=xx	テストモードセレクトの入力 (TMS) の GPIO 設定
jtag_ck=xx	テストクロック入力 (TMS) の GPIO 設定
jtag_do=xx	テストデータ出力 (TD0) の GPIO 設定
jtag_di=xx	テストデータ入力 (TDI) の GPIO 設定

設定の例 :

```
[jtag_para]  
jtag_enable    = 1  
jtag_ms       = port:PB14<3><default><default><default>  
jtag_ck       = port:PB15<3><default><default><default>  
jtag_do       = port:PB16<3><default><default><default>  
jtag_di       = port:PB17<3><default><default><default>
```

4.4.2 TFRAM

1. [dram_para]

設定アイテム	意味
dram_baseaddr=xx	DRAM がアクセスする物理初期アドレス、0x40000000固定である
dram_clk =xx	DRAM のクロック周波数、単位は MHz、24 の整数倍で120から480までに設定できる
dram_type =xx	DRAM の種類：2 は DDR2、3 は DDR3
dram_rank_num =xx	DRAM のチップセレクト数：1：一つのチップセレクト、2：二つのチップセレクト
dram_chip_density =xx	一枚 DRAM の容量、単位は Mb、例えば2048、1024
dram_io_width=xx	一枚 DRAM のビット幅、一般的には16に設定する
dram_bus_width=xx	全ての DRAM のバス幅。例えば、二枚の16ビットの DRAM が32ビットのソリューションになったら、ここで32に設定する。
dram_cas=xx	DRAM CAS 値。6、7、8、9が設定できる。DRAM の仕様書とスピードによって確定する。
dram_zq=xx	DRAM コントローラ内部パラメータ、出荷時設定され、変更しないでください。 ODT は enable が必要かどうか。
dram_odt_en=xx	0：必要なし、1：必要。電力を節約するために、一般的には0に設定する。
dram_size=xx	全ての DRAM の総容量値、単位は MB。
dram_tpr0=xx	DRAM コントローラ内部パラメータ、工場に調整され、変更しないでください。
dram_tpr1=xx	DRAM コントローラ内部パラメータ、工場に調整され、変更しないでください。
dram_tpr2=xx	DRAM コントローラ内部パラメータ、工場に調整され、変更しないでください。
dram_tpr3=xx	DRAM コントローラ内部パラメータ、工場に調整され、変更しないでください。
dram_tpr4=xx	DRAM コントローラ内部パラメータ、工場に調整され、変更しないでください。
dram_emr1=xx	DRAM コントローラ内部パラメータ、工場に調整され、変更しないでください。
dram_emr2=xx	DRAM コントローラ内部パラメータ、工場に調整され、変更しないでください。
dram_emr3=xx	DRAM コントローラ内部パラメータ、工場に調整され、変更しないでください。

設定の例：

```
[dram_para]
dram_baseaddr = 0x40000000
dram_clk = 360
dram_type = 3
dram_rank_num = 0xffffffff
dram_chip_density = 0xffffffff
dram_io_width = 0xffffffff
dram_bus_width = 0xffffffff
dram_cas = 9
dram_zq = 0x7b
dram_odt_en = 0
dram_size = 0xffffffff
dram_tpr0 = 0x42d899b7
dram_tpr1 = 0xa090
dram_tpr2 = 0x22a00
dram_tpr3 = 0x0
dram_tpr4 = 0x1
dram_tpr5 = 0x0
dram_emr1 = 0x4
dram_emr2 = 0x10
dram_emr3 = 0x0
```

4.4.3 GPU(mali)

1. [mali_para]

設定アイテム

mali_usde=xx

mali_clkdiv=xx

意味

MALI モジュール enable

960MHz/mali_clkdiv は GPU の入力 CLOCK
とする

4.4.4 2D 加速 (G2D)

1. [g2d_para]

設定アイテム

g2d_usde=xx

g2d_size=xx

意味

g2d モジュールは使用かどうか

g2d メモリサイズ、デフォルトは0x1000000

4.4.5 ネットワーク (Ethernet MAC)

1. [emac_para]

設定アイテム

意味

```

emac_used=xx          emac モジュール enable フラグ
emac_rxd3 =xx         emac 受信データバスの GPIO 設定
emac_rxd2 =xx         /
emac_rxd1 =xx         /
emac_rxd0 =xx         /
emac_txd3 =xx         emac 送信データバスの GPIO 設定
emac_txd2 =xx         /
emac_txd1 =xx         /
emac_txd0 =xx         /
emac_rxclk =xx        emac 受信クロックの GPIO 設定
emac_rxerr =xx        emac 受信エラーの GPIO 設定
emac_rxdV =xx         emac 受信 enable の GPIO 設定
emac_mdc =xx          emac mii クロックの GPIO 設定
emac_mdio =xx         Emac mii データの GPIO 設定
emac_txen=xx          emac 送信 enable の GPIO 設定
emac_txclk=xx         emac 送信クロックの GPIO 設定
emac_crs = xx         emac キャリア状態の GPIO 設定
emac_col=xx           emac コンフリクト検出の GPIO 設定
emac_reset =xx        emac phy reset 信号の GPIO 設定
dram_emr3=xx          //
  
```

設定の例;

```

[emac_para]
emac_used      = 0
emac_rxd3     = port:PA00<2><default><default><default>
emac_rxd2     = port:PA01<2><default><default><default>
emac_rxd1     = port:PA02<2><default><default><default>
emac_rxd0     = port:PA03<2><default><default><default>
emac_txd3     = port:PA04<2><default><default><default>
emac_txd2     = port:PA05<2><default><default><default>
emac_txd1     = port:PA06<2><default><default><default>
emac_txd0     = port:PA07<2><default><default><default>
emac_rxclk    = port:PA08<2><default><default><default>
emac_rxerr    = port:PA09<2><default><default><default>
emac_rxdV     = port:PA10<2><default><default><default>
emac_mdc      = port:PA11<2><default><default><default>
emac_mdio     = port:PA12<2><default><default><default>
emac_txen     = port:PA13<2><default><default><default>
emac_txclk    = port:PA14<2><default><default><default>
emac_crs      = port:PA15<2><default><default><default>
emac_col      = port:PA16<2><default><default><default>
emac_reset    = port:PA17<1><default><default><default>
  
```

4.4.6 I2C バス

1、[twi0_para]

設定アイテム	意味
twi0_used=xx	TWI 使用コントロール：1：使用、0：未使用
twi0_scl=xx	TWI SCK の GPIO 設定
twi0_sda=xx	TWI SDA の GPIO 設定

設定の例：

```
twi0_used      =1
twi0_scl      = port:PB0<2><default><default><default>
twi0_sda      = port:PB1<2><default><default><default>
```

2、[twi1_para]

設定アイテム	意味
twi1_used=xx	TWI 使用コントロール：1. 使用；0:未使用
twi1_scl=xx	TWI SCK の GPIO 設定
twi1_sda=xx	TWI SDA の GPIO 設定

設定の例：

```
[twi1_para]
twi1_used = 1
twi1_scl = port:PB18<2><default><default><default>
twi1_sda = port:PB19<2><default><default><default>
```

3、[twi2_para]

設定アイテム	意味
Tw2_used=xx	TWI 使用コントロール：1. 使用；0:未使用
Tw2_scl=xx	TWI SCK の GPIO 設定
Tw2_sda=xx	TWI SDA の GPIO 設定

設定の例：

```
[tw2_para]
twi2_used = 1
twi2_scl = port:PB20<2><default><default><default>
twi2_sda = port:PB21<2><default><default><default>
```

4.4.7 シリアルポート (UART)

1、[uart_para0]

設定アイテム	意味
uart_used=xx	UART 使用コントロール：1. 使用；0:未

	使用
uart_port =xx	URAT ポート番号
uart_type = xx	URAT の種類 : 2 (2 wire), 4 (4 wire), 8 (8 wire)
uart0_tx =xx	URAT TX の GPIO 設定
uart0_rx=xx	URAT RX の GPIO 設定

設定の例 :

```
[uart_para0]
uart_used = 1
uart_port = 0
uart_type = 2
uart_tx = port:PB22<2><1><default><default>
uart_rx = port:PB23<2><1><default><default>
```

2、[uart_para 1]

設定アイテム	意味
uart_used=xx	URAT 使用コントロール : 1: 使用 ; 0:未使用
uart_port =xx	URAT ポート番号
uart_type = xx	URAT の種類 : 2 (2 wire), 4 (4 wire), 8 (8 wire)
Uart1_tx =xx	URAT TX の GPIO 設定
Uart1_rx=xx	URAT RX の GPIO 設定
Uart1_rts=xx	URAT RTS の GPIO 設定
Uart1_cts=xx	URAT CTS の GPIO 設定
Uart1_dtr=xx	URAT DTR の GPIO 設定
Uart1_dsr=xx	URAT DSR の GPIO 設定
Uart1_dcd=xx	URAT DCD の GPIO 設定
Uart1_ring=xx	URAT RING の GPIO 設定

設定の例 :

```
[uart_para1]
uart_used = 0
uart_port = 1
uart_type = 8
uart_tx = port:PA10<4><1><default><default>
uart_rx = port:PA11<4><1><default><default>
uart_rts = port:PA12<4><1><default><default>
uart_cts = port:PA13<4><1><default><default>
uart_dtr = port:PA14<4><1><default><default>
uart_dsr = port:PA15<4><1><default><default>
uart_dcd = port:PA16<4><1><default><default>
uart_ring = port:PA17<4><1><default><default>
```

3、 [uart_para2]

設定アイテム	意味
uart_used=xx	URAT 使用コントロール： 1. 使用 ; 0:未使用
uart_port =xx	URAT ポート番号
uart_type = xx	URAT の種類： 2 (2 wire), 4 (4 wire), 8 (8 wire)
Uart2_tx =xx	UART TX の GPIO 設定
Uart2_rx=xx	UART RX の GPIO 設定
Uart2_rts=xx	UART RTS の GPIO 設定
Uart2_cts=xx	UART CTS の GPIO 設定

設定の例;

```
[uart_para2]
uart_port = 2
uart_type = 4
uart_tx = port:PI18<3><1><default><default>
uart_rx = port:PI19<3><1><default><default>
uart_rts = port:PI16<3><1><default><default>
uart_cts = port:PI17<3><1><default><default>
```

4、 [uart_para3]

設定アイテム	意味
uart_used=xx	URAT 使用コントロール： 1. 使用 ; 0:未使用
uart_port =xx	URAT ポート番号
uart_type = xx	URAT の種類： 2 (2 wire), 4 (4 wire), 8 (8 wire)
Uart3_tx =xx	UART TX の GPIO 設定
Uart3_rx=xx	UART RX の GPIO 設定
Uart3_rts=xx	UART RTS の GPIO 設定
Uart3_cts=xx	UART CTS の GPIO 設定

設定の例

```
[uart_para3]
uart_used = 0
uart_port = 3
uart_type = 4
uart_tx = port:PH00<4><1><default><default>
uart_rx = port:PH01<4><1><default><default>
uart_rts = port:PH02<4><1><default><default>
uart_cts = port:PH03<4><1><default><default>
```

5、 [uart_para4]

設定アイテム	意味
--------	----

uart_used=xx URAT 使用コントロール：1. 使用；0:未
使用
uart_port =xx URAT ポート番号
uart_type = xx UART の種類：2 (2 wire), 4 (4 wire), 8
(8 wire)
Uart4_tx =xx UART TX の GPIO 設定
Uart4_rx=xx UART RX の GPIO 設定
設定の例
[uart_para4]
uart_used = 0
uart_port = 4
uart_type = 2
uart_tx = port:PH04<4><1><default><default>
uart_rx = port:PH05<4><1><default><default>

6、[uart_para5]

設定アイテム 意味
uart_used=xx URAT 使用コントロール：1. 使用；0:未
使用
uart_port =xx URAT ポート番号
uart_type = xx UART の種類：2 (2 wire), 4 (4 wire), 8
(8 wire)
Uart5_tx =xx UART TX の GPIO 設定
Uart5_rx=xx UART RX の GPIO 設定

設定の例

[uart_para5]
uart_used = 1
uart_port = 5
uart_type = 2
uart_tx = port:PI10<3><1><default><default>
uart_rx = port:PI11<3><1><default><default>

7、[uart_para6]

設定アイテム 意味
uart_used=xx URAT 使用コントロール：1. 使用；0:未
使用
uart_port =xx URAT ポート番号
uart_type = xx UART の種類：2 (2 wire), 4 (4 wire), 8
(8 wire)
Uart6_tx =xx UART TX の GPIO 設定
Uart6_rx=xx UART RX の GPIO 設定

設定の例：

[uart_para6]

```
uart_used = 1
uart_port = 6
uart_type = 2
uart_tx = port:PI12<3><1><default><default>
uart_rx = port:PI13<3><1><default><default>
```

8、 [uart_para7]

設定アイテム	意味
uart_used=xx	URAT 使用制御： 1. 使用 ; 0:不使用
uart_port =xx	URAT ポート番号
uart_type = xx	UART TX の種類 : 2 (2 wire), 4 (4 wire), 8 (8 wire)
Uart7_tx =xx	UART TX の GPIO 設定
Uart7_rx=xx	UART RX の GPIO 設定

設定の例

```
[uart_para7]
uart_used = 0
uart_port = 7
uart_type = 2
uart_tx = port:PA14<4><1><default><default>
uart_rx = port:PA15<4><1><default><default>
```

4.4.8 SPI バス

1、 [spio_para]

設定アイテム	意味
spi_used=xx	SPI 使用コントロール： 1. 使用 ; 0:未使用
spi_cs0=xx	SPI CS0の GPIO 設定
spi_cs1=xx	SPI CS1の GPIO 設定
spi_sclk=xx	SPI CLK の GPIO 設定
spi_mosi=xx	SPI MOSI の GPIO 設定
spi_miso=xx	SPI MISO の GPIO 設定

設定の例 :

```
[spio_para]
spi_used = 0
spi_cs_bitmap = 1
spi_cs0 = port:PI10<2><default><default><default>
spi_cs1 = port:PI14<2><default><default><default>
spi_sclk = port:PI11<2><default><default><default>
spi_mosi = port:PI12<2><default><default><default>
spi_miso = port:PI13<2><default><default><default>
```

2. [spi1_para]

設定アイテム	意味
spi_used=xx	SPI 使用コントロール：1. 使用；0:未使用
spi_cs0=xx	SPI CS0の GPIO 設定
spi_cs1=xx	SPI CS1の GPIO 設定
spi_sclk=xx	SPI CLK の GPIO 設定
spi_mosi=xx	SPI MOSI の GPIO 設定
spi_miso=xx	SPI MISO の GPIO 設定

設定の例；

```
[spi1_para]
spi_used = 0
spi_cs_bitmap = 1
spi_cs0 = port:PA00<3><default><default><default>
spi_cs1 = port:PA04<3><default><default><default>
spi_sclk = port:PA01<3><default><default><default>
spi_mosi = port:PA02<3><default><default><default>
spi_miso = port:PA03<3><default><default><default>
```

3. [spi2_para]

設定アイテム	意味
spi_used=xx	SPI 使用コントロール：1. 使用；0:未使用
spi_cs0=xx	SPI CS0の GPIO 設定
spi_cs1=xx	SPI CS1の GPIO 設定
spi_sclk=xx	SPI CLK の GPIO 設定
spi_mosi=xx	SPI MOSI の GPIO 設定
spi_miso=xx	SPI MISO の GPIO 設定

設定の例：

```
spi_used = 0
spi_cs_bitmap = 1
spi_cs0 = port:PB14<2><default><default><default>
spi_sclk = port:PB15<2><default><default><default>
spi_mosi = port:PB16<2><default><default><default>
spi_miso = port:PB17<2><default><default><default>
```

4 [spi3_para]

設定アイテム	意味
spi_used=xx	SPI 使用コントロール：1. 使用；0:未使用
spi_cs0=xx	SPI CS0の GPIO 設定
spi_cs1=xx	SPI CS1の GPIO 設定

spi_sclk=xx	SPI CLK の GPIO 設定
spi_mosi=xx	SPI MOSI の GPIO 設定
spi_miso=xx	SPI MISO の GPIO 設定

設定の例 :

```
[spi3_para]
spi_used = 0
spi_cs_bitmap = 1
spi_cs0 = port:PA05<3><default><default><default>
spi_cs1 = port:PA09<3><default><default><default>
spi_sclk = port:PA06<3><default><default><default>
spi_mosi = port:PA07<3><default><default><default>
spi_miso = port:PA08<3><default><default><default>
```

5 [spi_devices]

設定アイテム

意味

次の「spi_board0」に関係がある。マザーボードと spi デバイスに接続する数を指定する。もし N つの spi デバイスがあれば、「spi_devices」には N つの設定（「spi_board0」から「spi_boardN」まで）がある。

spi_dev_num=xx

6. [spi_board0]

設定アイテム

意味

modalias=xx

spi モジュールの名前

max_speed_hz=xx

最大伝送スピード

bus_num=xx

spi デバイスコントローラの番号

chip_select=xx

理論的には0.1.2.3が選択でき、現在は1.2のみ選択できる（チップがインタフェースを引き出さない）

mode=xx

SPI MOSI の GPIO 設定

full_duplex=xx

動作モード(1: Duplex、0: Half-duplex)

manual_cs=xx

C S レベルを制御する。現在はこの機能をサポートしない。

4.4.9 抵抗スクリーン (rtp)

1. [rtp_para]

設定アイテム

意味

rtp_used=xx

使用かどうか

rtp_screen_size=xx

スクリーンのサイズ、単位はインチ。対角線の方向の長さを基準にする

rtp_regidity_level=xx	表面スクリーンの硬さ。指で押し、上げてからカウントする。ハードウェアがデータを採集できないまでの時間（単位は10ms）をカウンタする。一般的には、5インチのスクリーンは5、7インチのスクリーンは7に設定する。
rtp_press_threshold_enbale=xx	圧力限界のプルプを開くかどうか。0：開かないを推奨する
rtp_press_threshold=xx	rtp_press_threshold_enbale は1になると、このアイテムが有効になる。0から0xFFFFFF までの任意数値が取られる。数値は小さい方が敏感する。0xF を推奨する。
rtp_sensitive_level=xx	敏感レベル。0から0xF までの任意数値。数値が大きい方が敏感する。0xF を推奨する。
rtp_exchange_x_y_flag=xx	スクリーンの x/y 軸を変換する場合に、1に設定する。一般的には0に設定する。

4.4.10 静電スクリーン (capacitor tp)

1. [ctp_para]

設定アイテム

ctp_used=xx

意味

静電タッチスクリーンを起動かどうか。

1：起動；0：起動しない。

ctp_name =xx

タッチプラン。現在は「ft5x_ts」又は「Goodix-TS」選択できる

ctp_twi_id=xx

i2c adapter を選択する。現在は0、2が選択できる

tp_twi_addr =xx

i2c デバイスのアドレスを指定する。ハードウェアに関する。

ctp_screen_max_x=xx

タッチボード x 軸の最大値

ctp_screen_max_y=xx

タッチボード y 軸の最大値

ctp_revert_x_flag=xx

x 軸をフリップするかどうか。1：フリップ、0：フリップなし

ctp_revert_y_flag=xx

y 軸をフリップするかどうか。1：フリップ、0：フリップなし

cpt_int_port=xx

静電スクリーンの割り込み信号の GPIO 設定

cpt_wakeup=xx

静電スクリーンウエイクアップ信号の GPIO 設定

cpt_io_port=xx

静電スクリーンの io 信号、今は割り込み信号とピンを共用する。

設定の例：

```

ctp_used = 1
ctp_name = "gs11680"
ctp_twi_id = 2
ctp_twi_addr = 0x40
ctp_screen_max_x = 1024
ctp_screen_max_y = 600
ctp_revert_x_flag = 0
ctp_revert_y_flag = 0
ctp_exchange_x_y_flag = 1
ctp_int_port = port:PH21<6><default><default><default>
ctp_wakeup = port:PB13<1><default><default><1>
  
```

注意事項：新静電タッチicをサポートするために、A10 bspの設定状況によって元のタッチicコードを修正しなければならない。

sys_configに、cyp_twi_idはハードウェアと同じように接続する。

ドライバのコードに、TWIのデバイス名+アドレスはsys_configのcyp_name、ctp_twi_addrと同じである。同時に、sys_configにあるほかのキーワードも設定必要。

4.4.11 タッチキー (touch key)

1. [tkey_para]

設定アイテム

意味

tkey_used =xx

1:タッチスクリーンをサポートする。0 : サポートしない

tkey_name =xx

タッチプラン。現在は hv_keypad のみ選択できる

tkey_twi_id =xx

i2c adapter を選択する。現在は0.2が選択できる

tkey_twi_addr=xx

i2c デバイスのアドレスを表明する。ハードウェアに関する。

tkey_int=xx

タッチキー割り込み信号の gpio 設定

設定の例：

```

tkey_used = 0
tkey_name = "hv_keypad"
tkey_twi_id = 2
tkey_twi_addr = 0x62
tkey_int = port:PI13<6><default><default><default>
  
```

注意事項：

サポートすれば、tkey_usedを1に設定すると同時に、対応するキーも設定する。でなければ、0に設定する。

4.4.12 モーター (motor)

1. [tmotor_para]

設定アイテム	意味
motor_used = xx	モーターを起動するかどうか、1 : 起動 ; 0 : 起動しない
motor_shake =xx	モーターの gpio 設定

設定の例 :

```
motor_used = 0
motor_shake = port:PB03<1><default><default><1>
```

注意事項 :

```
motor_shake = port:PB03<1><default><default><0>
```

ioインタフェースの出力はデフォルト0にする。そうすれば、初期化してから振動しない。

4.4.13 フラッシュ (nand flash)

1. [nand_para]

設定アイテム	意味
nand_used=xx	nand モジュール enable フラグ
nand_we=xx	nand クロック信号を書く gpio 設定
nand_ale=xx	nand アドレス enable 信号の gpio 設定
nand_cle=xx	nand コマンド enable 信号の gpio 設定
nand_ce1=xx	nand チップセレクト1信号の gpio 設定
nand_ce0=xx	nand チップセレクト0信号の gpio 設定
nand_nre=xx	nand クロック信号を読む gpio 設定
nand_rb0=xx	nand read/busy 0信号の gpio 設定
nand_rb1=xx	nand read/busy 1信号の gpio 設定
nand_d0=xx	nand データバス信号の gpio 設定
nand_d1=xx	/
nand_d2=xx	/
nand_d3=xx	/
nand_d4=xx	/
nand_d5=xx	/
nand_d6=xx	/
nand_d7=xx	/
nand_wp=xx	nand 保護信号を書く gpio 設定
nand_ce2=xx	nand チップセレクト2信号の gpio 設定
nand_ce3=xx	nand チップセレクト3信号の gpio 設定
nand_ce4=xx	nand チップセレクト4信号の gpio 設定
nand_ce5=xx	nand チップセレクト5信号の gpio 設定
nand_ce6=xx	nand チップセレクト6信号の gpio 設定
nand_ce7=xx	nand チップセレクト7信号の gpio 設定

```
nand_spi=xx //
nand_ndqs=xx nand ddr クロック信号の gpio 設定
```

設定の例：

```
[nand_para]
nand_used = 1
nand_we = port:PC00<2><default><default><default>
nand_ale = port:PC01<2><default><default><default>
nand_cle = port:PC02<2><default><default><default>
nand_ce1 = port:PC03<2><default><default><default>
nand_ce0 = port:PC04<2><default><default><default>
nand_nre = port:PC05<2><default><default><default>
nand_rb0 = port:PC06<2><default><default><default>
nand_rb1 = port:PC07<2><default><default><default>
nand_d0 = port:PC08<2><default><default><default>
nand_d1 = port:PC09<2><default><default><default>
nand_d2 = port:PC10<2><default><default><default>
nand_d3 = port:PC11<2><default><default><default>
nand_d4 = port:PC12<2><default><default><default>
nand_d5 = port:PC13<2><default><default><default>
nand_d6 = port:PC14<2><default><default><default>
nand_d7 = port:PC15<2><default><default><default>
nand_wp = port:PC16<2><default><default><default>
nand_ce2 = port:PC17<2><default><default><default>
nand_ce3 = port:PC18<2><default><default><default>
nand_ce4 =
nand_ce5 =
nand_ce6 =
nand_ce7 =
nand_spi = port:PC23<3><default><default><default>
nand_ndqs = port:PC24<2><default><default><default>
```

4.4.14 表示を初期化 (disp init)

1, [disp_init]

設定アイテム	意味
disp_init_enable=xx	表示を初期化するかどうか
	表示モード：
	0:screen0<screen0, fb0>
	1:screen1<screen1, fb0>
disp_mode=xxsys_config.fex	2:two_diff_screen_diff_contents <screen0、screen1、fb0、fb1>;
	3:two_same_screen_diff_contets

<screen0、screen1、fb0>

4:two_diff_screen_same_contents<screen0、screen1、fb0>

screen0_output_type=xx スクリーン0出力種類：(0:none、1:lcd、2:tv、3:hDMI、4:vga)

screen0_output_mode =xx スクリーン0出力モード：(used for tv/hDMI output、0:480i 1:576i 2:480p 3:576p 4:720p50 5:720p60 6:1080i50 7:1080i60 8:1080p24 9:1080p50 10:1080p60 11:pal 14:ntsc)

screen1_output_type=xx スクリーン1出力種類：(0:none、1:lcd、2:tv、3:hDMI、4:vga)

screen1_output_mode =xx スクリーン1出力モード：(used for tv/hDMI output、0:480i 1:576i 2:480p 3:576p 4:720p50 5:720p60 6:1080i50 7:1080i60 8:1080p24 9:1080p50 10:1080p60 11:pal 14:ntsc)

fb0_framebuffer_num=xx fb0のbuffer number (ドライバはfb0にいくつかのbufferを割り当てる。例えば、ダブルbufferを使う時、2を書く。)

fb0_format=xx fb0のフォーマット(4:RGB655 5:RGB565 6:RGB556 7:ARGB1555 8:RGBA5551 9:RGB888 10:ARGB8888 12:ARGB4444)

fb0_pixel_sequence=xx fb0のpixel sequence(0:ARGB 1:BGRA 2:ABGR 3:RGBA)

fb0_scaler_mode_enable=xx fb0はscaler modeを使用かどうか、FEに設定する

fb1_framebuffer_num=xx fb1のbuffer number (ドライバはfb1にいくつかのbufferを割り当てる。例えば、ダブルbufferを使う時、2を書く。)

fb1_format=xx fb1のフォーマット(4:RGB655 5:RGB565 6:RGB556 7:ARGB1555 8:RGBA5551 9:RGB888 10:ARGB8888 12:ARGB4444)

fb1_pixel_sequence=xx fb1のpixel sequence(0:ARGB 1:BGRA 2:ABGR 3:RGBA)

fb1_scaler_mode_enable=xx fb1はscaler modeを使用かどうか

設定の例;

```
[disp_init]
disp_init_enable = 1
disp_mode = 0
screen0_output_type = 3
```

```

screen0_output_mode = 5
screen1_output_type = 1
screen1_output_mode = 4
fb0_framebuffer_num = 2
fb0_format = 10
fb0_pixel_sequence = 0
fb0_scaler_mode_enable = 1
fb1_framebuffer_num = 2
fb1_format = 10
fb1_pixel_sequence = 0
fb1_scaler_mode_enable = 0
  
```

4.4.15 LCD スクリーン 0

1[lcd0_para]

設定アイテム	意味
lcd_used=xx	lcd0 interface を使用かどうか。以下の設定は使用するとき、有効となる
lcd_x=xx	lcd active width
lcd_y=xx	lcd active height
lcd_dclk_freq=xx	pixel clock、in MHZ unit
lcd_pwm_not_used=xx	PWM を使うかどうか。0:使用、1:未使用 (PWM ioを直接に高める/低める)
lcd_pwm_ch=xx	PWM 通路。0:PWM0、1:PWM1。一般的には LCD0 はPWM0を使う、LCD1 はPWM1を使う。
lcd_pwm_freq=xx	pwm freq、in HZ unit
lcd_pwm_pol =xx	pwm polarity
lcd_if =xx	lcd interface(0:hv(sync+de); 1:8080; 2:ttl; 3:lvds)
lcd_hbp=xx	hsync back porch
lcd_ht=xx	hsync total cycle
lcd_vbp=xx	vsync back porch
lcd_vt=xx	vsync total cycle *2
lcd_hv_if =xx	hv interface(0:hv parallel 1:hv serial)
lcd_hv_smode=xx	serial i/f mode(0:RGB888 1:CCIR656)
lcd_hv_s888_if=xx	serial RGB forma
lcd_hv_syuv_if=xx	serial YUV format
lcd_hv_vspw=xx	vysnc plus width
lcd_hv_hspw=xx	hsync plus width
lcd_lvds_ch=xx	0:single channel; 1:dual channel
lcd_lvds_mode=xx	0:NS mode; 1:JEIDA mode
lcd_lvds_bitwidth=xx	0:24bit; 1:18bit
lcd_lvds_io_cross=xx	0:normal; 1:pn cross

```

cpu i/f mode(0:18bit; 1:16bit mode0;
2:16bit mode1; 3:16bit mode2;
4:16bit mode3; 5:9bit; 6:8bit 256K;
7:8bit 65K)

lcd_cpu_if=xx

lcd_frm=xx
0:disable; 1:enable rgb666 dither;
2:enable rgb656 dither

lcd_io_cfg0=xx
lcd io inv

lcd_gamma_correction_en=xx
gamma 補正を行うかどうか。行われれば、後
ろの256つの gamama パラメータを書く必要

lcd_gamma_tbl_0=xx
gamama パラメータ第0項目、
(red<<16) || (gree<<8) || blue.

lcd_gamma_tbl_1=xx
gamama パラメータ第1項目、
(red<<16) || (gree<<8) || blue.

lcd_gamma_tbl_255=xx
gamama パラメータ第255項目、
(red<<16) || (gree<<8) || blue.

lcd_bl_en_used=xx
LCD_BL_EN ピンを使用かどうか
lcd_bl_en=xx
LCD_BL_EN の gpio 設定

lcd_power_used=xx
LCD_VCC_control のピンを使用かどうか
lcd_power=xx
LCD_VCC_control の gpio 設定

lcd_pwm_used=xx
lcd pwm のピンを使用かどうか (使用)
lcd_pwm=xx
lcd pwm の gpio 設定 (PWM0は固定で PB02
を利用し、PWM1は PI03を固定で利用する)

lcd_gpio_0=xx
2/3 - wire I/F の SCL の GPIO 設定
lcd_gpio_1=xx
2/3 - wire I/F の TFA の GPIO 設定
lcd_gpio_2=xx
2/3 - wire I/F の SCEN の GPIO 設定
lcd_gpio_3=xx
lcd モジュールの reset の GPIO 設定

lcdd0=xx
lcd データの GPIO 設定
lcd1=xx
lcd データの GPIO 設定
lcdd2=xx
lcd データの GPIO 設定
lcdd3=xx
lcd データの GPIO 設定
lcdd4=xx
lcd データの GPIO 設定
lcdd5=xx
lcd データの GPIO 設定
lcdd6=xx
lcd データの GPIO 設定
lcdd7=xx
lcd データの GPIO 設定
lcdd8=xx
lcd データの GPIO 設定
lcdd9=xx
lcd データの GPIO 設定
lcdd10=xx
lcd データの GPIO 設定
lcdd11=xx
lcd データの GPIO 設定
lcdd12=xx
lcd データの GPIO 設定
lcdd13=xx
lcd データの GPIO 設定
lcdd14=xx
lcd データの GPIO 設定
lcdd15=xx
lcd データの GPIO 設定
lcdd16=xx
lcd データの GPIO 設定
lcdd17=xx
lcd データの GPIO 設定
lcdd18=xx
lcd データの GPIO 設定

```

lcd19=xx	lcd データの GPIO 設定
lcd20=xx	lcd データの GPIO 設定
lcd21=xx	lcd データの GPIO 設定
lcd22=xx	lcd データの GPIO 設定
lcd23=xx	lcd データの GPIO 設定
lcdclk=xx	lcd データの GPIO 設定 (具体的な信号は実際の回路と関係する)
lcdde=xx	lcd データの GPIO 設定 (具体的な信号は実際の回路と関係する)
lcdhsync=xx	lcd データの GPIO 設定 (具体的な信号は実際の回路と関係する)
lcdvsync=xx	lcd データの GPIO 設定 (具体的な信号は実際の回路と関係する)

設定の例 :

```
[lcd0_para]
Lcd_used=0
lcd_x = 800
lcd_y = 480
lcd_dclk_freq = 33
lcd_pwm_freq = 1000
lcd_pwm_pol = 0
lcd_srgb = 0x00202020
lcd_swap = 0
lcd_if = 0
lcd_hbp = 215
lcd_ht = 1055
lcd_vbp = 34
lcd_vt = 1050
lcd_hv_if = 0
lcd_hv_smode = 0
lcd_hv_s888_if = 0
lcd_hv_syuv_if = 0
lcd_hv_vspw = 0
lcd_hv_hspw = 0
lcd_hv_lde_used = 0
lcd_hv_lde_iovalue = 0
lcd_ttl_stvh = 0
lcd_ttl_stvdl = 0
lcd_ttl_stvdp = 0
lcd_ttl_ckvt = 0
lcd_ttl_ckvh = 0
lcd_ttl_ckvd = 0
lcd_ttl_oevt = 0
lcd_ttl_oevh = 0
```

```
lcd_ttl_oevd = 0
lcd_ttl_sthh = 0
lcd_ttl_sthd = 0
lcd_ttl_oehh = 0
lcd_ttl_oehd = 0
lcd_ttl_revd = 0
lcd_ttl_datarate = 0
lcd_ttl_revsel = 0
lcd_ttl_datainv_en = 0
lcd_ttl_datainv_sel = 0
lcd_lvds_ch = 0
lcd_lvds_mode = 0
lcd_lvds_bitwidth = 0
lcd_lvds_io_cross = 0
lcd_cpu_if = 0
lcd_cpu_da = 0
lcd_frm = 0
lcd_io_cfg0 = 0x10000000
lcd_io_cfg1 = 0
lcd_io_strength = 0
lcd_bl_en_used = 1
lcd_bl_en = port:PH07<1><0><default><1>
lcd_power_used = 1
lcd_power = port:PH08<1><0><default><1>
lcd_pwm_used = 1
lcd_pwm = port:PB02<2><default><default><default>
lcd_gpio_0 =
lcd_gpio_1 =
lcd_gpio_2 =
lcd_gpio_3 =
lcdd0 = port:PD00<2><default><default><default>
lcdd1 = port:PD01<2><default><default><default>
lcdd2 = port:PD02<2><default><default><default>
lcdd3 = port:PD03<2><default><default><default>
lcdd4 = port:PD04<2><default><default><default>
lcdd5 = port:PD05<2><default><default><default>
lcdd6 = port:PD06<2><default><default><default>
lcdd7 = port:PD07<2><default><default><default>
lcdd8 = port:PD08<2><default><default><default>
lcdd9 = port:PD09<2><default><default><default>
lcdd10 = port:PD10<2><default><default><default>
lcdd11 = port:PD11<2><default><default><default>
lcdd12 = port:PD12<2><default><default><default>
lcdd13 = port:PD13<2><default><default><default>
lcdd14 = port:PD14<2><default><default><default>
```

```

lcdd15 = port:PD15<2><default><default><default>
lcdd16 = port:PD16<2><default><default><default>
lcdd17 = port:PD17<2><default><default><default>
lcdd18 = port:PD18<2><default><default><default>
lcdd19 = port:PD19<2><default><default><default>
lcdd20 = port:PD20<2><default><default><default>
lcdd21 = port:PD21<2><default><default><default>
lcdd22 = port:PD22<2><default><default><default>
lcdd23 = port:PD23<2><default><default><default>
lcdclk = port:PD24<2><default><default><default>
lcdde = port:PD25<2><default><default><default>
lcdhsync = port:PD26<2><default><default><default>
lcdvsync = port:PD27<2><default><default><default>

```

4.4.16 LCD スクリーン 1

1[lcd1_para]

設定アイテム	意味
lcd_used=xx	lcd0 interface を使用かどうか。以下は使用するとき、有効となる
lcd_x=xx	lcd active width
lcd_y=xx	lcd active height
lcd_dclk_freq=xx	pixel clock, in MHZ unit
lcd_pwm_not-used=xx	PWM を使うかどうか。0 : 使用、1 : 未使用 (PWM io を直接に高める/低める)
lcd_pwm_ch=xx	PWM 通路。0:PWM0、1:PWM1。一般的には LCD0 は PWM0 を利用し、LCD1 は PWM1 を利用する。
lcd_pwm_freq=xx	pwm freq, in HZ unit
lcd_pwm_pol =xx	pwm polarity
lcd_if =xx	lcd interface (0:hv(sync+de)、1:8080、2:ttl、3:lvds)
lcd_hbp=xx	hsync back porch
lcd_ht=xx	hsync total cycle
lcd_vbp=xx	vsync back porch
lcd_vt=xx	vsync total cycle *2
lcd_hv_if =xx	hv interface (0:hv parallel 1:hv serial)
lcd_hv_smode=xx	serial i/f mode (0:RGB888 1:CCIR656)
lcd_hv_s888_if=xx	serial RGB forma
lcd_hv_syuv_if=xx	serial YUV format
lcd_hv_vspw=xx	vsync plus width
lcd_hv_hspw=xx	hsync plus width
lcd_lvds_ch=xx	0:single channel; 1:dual channel

lcd_lvds_mode=xx	0:NS mode、 1:JEIDA mode
lcd_lvds_bitwidth=xx	0:24bit、 1:18bit
lcd_lvds_io_cross=xx	0:normal、 1:pn cross
lcd_cpu_if=xx	cpu i/f mode(0:18bit、 1:16bit mode0、 2:16bit model、 3:16bit mode2、 4:16bit mode3、 5:9bit、 6:8bit 256K、 7:8bit 65K)
lcd_frm=xx	0:disable; 1:enable rgb666 dither; 2:enable rgb656 dither
lcd_io_cfg0=xx	lcd io inv
lcd_gamma_correction_en=xx	gamma 補正を行うかどうか。行われれば、後 ろの256つの gamama パラメータを書く必要 がある
lcd_gamma_tbl_0=xx	gamama パラメータ第0項目、 (red<<16) (gree<<8) blue.
lcd_gamma_tbl_1=xx	gamama パラメータ第1項目、 (red<<16) (gree<<8) blue.
lcd_gamma_tbl_255=xx	gamama パラメータ第255項目、 (red<<16) (gree<<8) blue.
lcd_bl_en_used=xx	lcd_bl_en ピンを使用かどうか
lcd_bl_en=xx	LCD_BL_EN の gpio 設定
lcd_power_used=xx	LCD_VCC_control のピンを使用かどうか
lcd_power=xx	LCD_VCC_control の gpio 設定
lcd_pwm_used=xx	lcd pwm のピンを使用かどうか (使用)
lcd_pwm=xx	lcd pwm の gpio 設定 (PWM0が PB02, PWM1 が PI03を使用する)
lcd_gpio_0=xx	2/3 - wire I/F における SCL の GPIO 設定
lcd_gpio_1=xx	2/3 - wire I/F における TFA の GPIO 設定
lcd_gpio_2=xx	2/3 - wire I/F における SCEN の GPIO 設 定
lcd_gpio_3=xx	lcd モジュールの reset の GPIO 設定
lcdd0=xx	lcd データの GPIO 設定
lcdd1=xx	lcd データの GPIO 設定
lcdd2=xx	lcd データの GPIO 設定
lcdd3=xx	lcd データの GPIO 設定
lcdd4=xx	lcd データの GPIO 設定
lcdd5=xx	lcd データの GPIO 設定
lcdd6=xx	lcd データの GPIO 設定
lcdd7=xx	lcd データの GPIO 設定
lcdd8=xx	lcd データの GPIO 設定
lcdd9=xx	lcd データの GPIO 設定
lcdd10=xx	lcd データの GPIO 設定
lcdd11=xx	lcd データの GPIO 設定
lcdd12=xx	lcd データの GPIO 設定
lcdd13=xx	lcd データの GPIO 設定

lcdd14=xx	lcd データの GPIO 設定
lcdd15=xx	lcd データの GPIO 設定
lcdd16=xx	lcd データの GPIO 設定
lcdd17=xx	lcd データの GPIO 設定
lcdd18=xx	lcd データの GPIO 設定
lcdd19=xx	lcd データの GPIO 設定
lcdd20=xx	lcd データの GPIO 設定
lcdd21=xx	lcd データの GPIO 設定
lcdd22=xx	lcd データの GPIO 設定
lcdd23=xx	lcd データの GPIO 設定
lcdclk=xx	lcd データの GPIO 設定 (具体的な信号は実際の回路と関係する)
lcdde=xx	lcd データの GPIO 設定 (具体的な信号は実際の回路と関係する)
lcdhsync=xx	lcd データの GPIO 設定 (具体的な信号は実際の回路と関係する)
lcdvsync=xx	lcd データの GPIO 設定 (具体的な信号は実際の回路と関係する)

設定の例 :

```
[lcd1_para]
Lcd_used=0
lcd_x = 800
lcd_y = 480
lcd_dclk_freq = 33
lcd_pwm_freq = 1000
lcd_pwm_pol = 0
lcd_srgb = 0x00202020
lcd_swap = 0
lcd_if = 0
lcd_hbp = 215
lcd_ht = 1055
lcd_vbp = 34
lcd_vt = 1050
lcd_hv_if = 0
lcd_hv_smode = 0
lcd_hv_s888_if = 0
lcd_hv_syuv_if = 0
lcd_hv_vspw = 0
lcd_hv_hspw = 0
lcd_hv_lde_used = 0
lcd_hv_lde_iovalue = 0
lcd_ttl_stvh = 0
lcd_ttl_stvdl = 0
lcd_ttl_stvdp = 0
```

```
lcd_ttl_ckvt = 0
lcd_ttl_ckvh = 0
lcd_ttl_ckvd = 0
lcd_ttl_oevt = 0
lcd_ttl_oevh = 0
lcd_ttl_oevd = 0
lcd_ttl_sthh = 0
lcd_ttl_sthd = 0
lcd_ttl_oehh = 0
lcd_ttl_oehd = 0
lcd_ttl_revd = 0
lcd_ttl_datarate = 0
lcd_ttl_revsel = 0
lcd_ttl_datainv_en = 0
lcd_ttl_datainv_sel = 0
lcd_lvds_ch = 0
lcd_lvds_mode = 0
lcd_lvds_bitwidth = 0
lcd_lvds_io_cross = 0
lcd_cpu_if = 0
lcd_cpu_da = 0
lcd_frm = 0
lcd_io_cfg0 = 0x10000000
lcd_io_cfg1 = 0
lcd_io_strength = 0
lcd_bl_en_used = 1
lcd_bl_en = port:PH07<1><0><default><1>
lcd_power_used = 1
lcd_power = port:PH08<1><0><default><1>
lcd_pwm_used = 1
lcd_pwm = port:PB02<2><default><default><default>
lcd_gpio_0 =
lcd_gpio_1 =
lcd_gpio_2 =
lcd_gpio_3 =
lcdd0 = port:PD00<2><default><default><default>
lcdd1 = port:PD01<2><default><default><default>
lcdd2 = port:PD02<2><default><default><default>
lcdd3 = port:PD03<2><default><default><default>
lcdd4 = port:PD04<2><default><default><default>
lcdd5 = port:PD05<2><default><default><default>
lcdd6 = port:PD06<2><default><default><default>
lcdd7 = port:PD07<2><default><default><default>
lcdd8 = port:PD08<2><default><default><default>
lcdd9 = port:PD09<2><default><default><default>
```

```

lccd10 = port:PD10<2><default><default><default>
lccd11 = port:PD11<2><default><default><default>
lccd12 = port:PD12<2><default><default><default>
lccd13 = port:PD13<2><default><default><default>
lccd14 = port:PD14<2><default><default><default>
lccd15 = port:PD15<2><default><default><default>
lccd16 = port:PD16<2><default><default><default>
lccd17 = port:PD17<2><default><default><default>
lccd18 = port:PD18<2><default><default><default>
lccd19 = port:PD19<2><default><default><default>
lccd20 = port:PD20<2><default><default><default>
lccd21 = port:PD21<2><default><default><default>
lccd22 = port:PD22<2><default><default><default>
lccd23 = port:PD23<2><default><default><default>
lcdclk = port:PD24<2><default><default><default>
lcdde = port:PD25<2><default><default><default>
lcdhsync = port:PD26<2><default><default><default>
lcdvsync = port:PD27<2><default><default><default>
  
```

4.4.17 カメラ (CSI)

1. [CSI0_para]

設定アイテム

意味

csi_used=xx

csi0を使用かどうか

csi_twi_id=xx

csi0に使用された iic

csi_mname=xx

csi0に使用されたモジュール名、ドライバに合わせなければならない。ドライバディレクトリの readme を参考ください。現在は ov7670、gc0308、gt2005、hi704、sp0338、mt9m112が選択できる。

csi_twi_addr=xx

csi0に使用されたモジュールの IIC アドレス。ドライバディレクトリの readme を参考ください。

使用しているモジュールのインタフェースのシーケンスを設定する

csi_if

0:8bit データ回線、Hsync, Vsync 付き

1:16bit データ回線、Hsync, Vsync 付き

2:24bit データ回線、Hsync, Vsync 付き

3:8bit データ回線 BT656組み込む同期、単一チャンネル

4:8bit データ回線、BT656組み込み同期、2重チャンネル

5:8bit データ回線、BT656組み込み同期、4チャンネル

	csi 受信 buffer モードを設定する :
csi_mode	0 : 一つの csi 受信が一つの buffer を対応する、1 : 二つの csi 受信内容を一つの buffer になる
csi_dev_qty	csi が接続するデバイス数を設定する。現在は1又は2に設定できる
csi_vflip	csi が写真を受信する時、上下逆転の状況。 0 : 正常、1 : 上下逆転
csi_hflip	csi が写真を受信する時、左右逆転の状況。 0 : 正常、1 : 左右逆転
csi_stby_mode	csi が stanby に入る時の処理を設定する。 0 : パワー on、stanbyio を引く ; 1 : パワー off、stanbyio を引く
csi_iovdd	csi io のパワーソースを設定する。 " axp20_p11 " : パワーが pmu の ldo3 から生成。 " axp20_hdmi " : パワーが pmu の ldo4 から生成。
csi_avdd	" : 非 pmu の ldo から生成。 一般的には、本ボードでは " " に設定する。 他のソリューションでは " axp20_p11 " に設定する。 csi avdd パワーソースを設定する。 " axp20_p11 " : パワーが pmu の ldo3 から生成。 " axp20_hdmi " : パワーが pmu の ldo4 から生成。 " : 非 pmu の ldo から生成。 一般的には、本ボードでは " " に設定する。 他のソリューションでは " axp20_p11 " に設定する。
csi_dvdd	csi dvdd パワーソースを設定する。 " axp20_p11 " : パワーが pmu の ldo3 から生成。 " axp20_hdmi " : パワーが pmu の ldo4 から生成。 " : 非 pmu の ldo から生成。 一般的には、本ボードでは " " に設定する。 他のソリューションでは " axp20_p11 " に設定する。
csi_pck=xx	モジュールから CSI0 に送信した clock の GPIO 設定
csi_ck=xx	CSI0 からモジュールに送信した clock の GPIO 設定
csi_hsync=xx	csi0 からモジュールに送信した行同期信号の GPIO 設定
csi_vsync=xx	モジュールから csi0 に送信したフレーム同期信号の GPIO 設定
csi_d0=xx	モジュールから csi0 に送信した 8bit/16bit

...

データの GPIO 設定

`csi_d15=xx`

モジュールの reset 制御する GPIO 設定、デフォルトでは reset が有効。(モジュールによって、High レベルで有効又は Low レベルで有効となる)

`csi_reset=xx`

モジュールのパワーの gpio 設定。

`csi_power_en=xx`

`csi_stby_mode=0`の場合に、`csi_power_en=1`をデフォルトにする。`csi_stby_mode=1`の場合に、`csi_power_en=0`をデフォルトにする。

`csi_stby=xx`

モジュールの standby の gpio 設定。デフォルトでは standby 有効。(モジュールによって、High レベルで有効、又は Low レベルで有効。)

`csi_reset_b=xx`

二つのモジュールは同時に一つの CSI に接続する時に、別の io 制御が必要ある。モジュールの reset の GPIO 設定、デフォルトでは reset 有効。(モジュールによって、High レベルで有効、又は Low レベルで有効)

`csi_power_en_b=xx`

二つのモジュールは同時に一つの CSI に接続する時に、別の io 制御が必要ある。モジュールのパワーの gpio 設定。`csi_stby_mode=0`の場合に、`csi_power_en=1`をデフォルトする。`csi_stby_mode=1`の場合に、`csi_power_en=0`をデフォルトする。

`csi_stby_b=xx`

二つのモジュールは同時に一つの CSI に接続する時に、別の io 制御が必要ある。モジュールの standby の gpio 設定。standby が有効をデフォルトする。(モジュールによって、High レベルで有効、又は Low レベルで有効)

設定の例 :

```
[csi0_para]
csi_used = 1
csi_dev_qty = 1
csi_stby_mode = 0
csi_mname = "gc0308"
csi_if = 0
csi_iovdd = ""
csi_avdd = ""
csi_dvdd = ""
csi_vol_iovdd =
```

```
csi_vol_dvdd =
csi_vol_avdd =
csi_vflip = 0
csi_hflip = 0
csi_flash_pol = 0
csi_facing = 0
csi_twi_id = 1
csi_twi_addr = 0x42
csi_pck = port:PE00<3><default><default><default>
csi_ck = port:PE01<3><default><default><default>
csi_hsync = port:PE02<3><default><default><default>
csi_vsync = port:PE03<3><default><default><default>
csi_d0 = port:PE04<3><default><default><default>
csi_d1 = port:PE05<3><default><default><default>
csi_d2 = port:PE06<3><default><default><default>
csi_d3 = port:PE07<3><default><default><default>
csi_d4 = port:PE08<3><default><default><default>
csi_d5 = port:PE09<3><default><default><default>
csi_d6 = port:PE10<3><default><default><default>
csi_d7 = port:PE11<3><default><default><default>
csi_reset = port:PH13<1><default><default><0>
csi_stby = port:PH19<1><default><default><0>
```

2 [CSI1_para]

設定アイテム

意味

csi_used=xx

csi0を使用かどうか

csi_twi_id=xx

csi0に使用された iic

csi_mname=xx

csi0に使用されたモジュール名、ドライバに
合わせなければなりません。ドライバディレクト
リの readme をご参照ください。現在は
ov7670, gc0308, gt2005, hi704, sp0338, mt9m112
が選択できる。

csi_twi_addr=xx

csi0に使用されたモジュールの IIC アドレ
ス。ドライバディレクトリの readme をご参照
ください。

使用しているモジュールのインタフェース
のシーケンスを設定する：

csi_if

0:8bit データ回線、Hsync, Vsync 付き
1:16bit データ回線、Hsync, Vsync 付き
2:24bit データ回線、Hsync, Vsync 付き
3:8bit データ回線 BT656組み込む同期、単一チ
ャンネル
4:8bit データ回線、BT656組み込み同期、2重
チャンネル
5:8bit データ回線、BT656組み込み同期、4チ

チャンネル

csi_mode	csi 受信 buffer を設定する：0：一つの csi 受信が一つの buffer を対応する、1：二つの csi 受信内容を一つの buffer になる
csi_dev_qty	csi が接続するデバイス数を設定する。現在は1又は2に設定できる
csi_vflip	csi が写真を受信する時、上下逆転の状況。 0：正常；1：上下逆転
csi_hflip	csi が写真を受信する時、左右逆転の状況。 0：正常；1：左右逆転
csi_stby_mode	csi が stanby に入る時の処理を設定する。 0：パワーon、stanbyio を引く；1：パワーoff、stanbyio を引く
csi_iovdd	csi io のパワーソースを設定する。 “axp20_p11”：パワーは pmu の ldo3から生成。 ” axp20_hdmi”：パワーは pmu の ldo4から生成。 ” ”：非 pmu の ldo から生成。 一般的には、本ボードは” ” に設定される。 多くのソリューションでは「axp20_p11」に設定される。
csi_avdd	csi avdd パワーソースを設定する。 “axp20_p11”：パワーは pmu の ldo3から生成。 ” axp20_hdmi”：パワーは pmu の ldo4から生成。 ” ”：非 pmu の ldo から生成。 一般的には、本ボードは” ” に設定される。 多くのソリューションでは「axp20_p11」に設定される。
csi_dvdd	csi dvdd パワーソースを設定する。 “axp20_p11”：パワーは pmu の ldo3から生成。 ” axp20_hdmi”：パワーは pmu の ldo4から生成。 ” ”：非 pmu の ldo から生成。 一般的には、本ボードは” ” に設定される。 多くのソリューションでは「axp20_p11」に設定される。
csi_pck=xx	モジュールから CSI0に送信した clock の GPIO 設定
csi_ck=xx	CSI0からモジュールに送信した clock の GPIO 設定
csi_hsync=xx	csi0からモジュールに送信した行同期信号の GPIO 設定
csi_vsync=xx	モジュールから csi0に送信したフレーム同

期信号の GPIO 設定

`csi_d0=xx …csi_d15=xx` モジュールから csi0 に送信した 8bit/16bit データの GPIO 設定

`csi_reset=xx` モジュールの reset 制御する GPIO 設定、デフォルトでは reset が有効。(モジュールによって、High レベルで有効又は Low レベルで有効となる)

`csi_power_en=xx` モジュールのパワーの gpio 設定。
`csi_stby_mode=0` の場合に、`csi_power_en=1` をデフォルトにする。`csi_stby_mode=1` の場合に、`csi_power_en=0` をデフォルトにする。

`csi_stby=xx` モジュールの standby の gpio 設定。デフォルトでは standby 有効。(モジュールによって、High レベルで有効、又は Low レベルで有効。)

`csi_reset_b=xx` 二つのモジュールは同時に一つの CSI に接続する時に、別の io 制御が必要ある。モジュールの reset の GPIO 設定、デフォルトでは reset 有効。(モジュールによって、High レベルで有効、又は Low レベルで有効)

`csi_power_en_b=xx` 二つのモジュールは同時に一つの CSI に接続する時に、別の io 制御が必要ある。モジュールのパワーの gpio 設定。`csi_stby_mode=0` の場合に、`csi_power_en=1` をデフォルトする。`csi_stby_mode=1` の場合に、`csi_power_en=0` をデフォルトする。

`csi_stby_b=xx` 二つのモジュールは同時に一つの CSI に接続する時に、別の io 制御が必要ある。モジュールの standby の gpio 設定。standby が有効をデフォルトする。(モジュールによって、High レベルで有効、又は Low レベルで有効)

設定の例 :

```
[csi1_para]
csi_used = 0
csi_dev_qty = 1
csi_stby_mode = 0
csi_mname = "gc0308"
csi_if = 0
csi_iovdd = ""
csi_avdd = ""
```

```
csi_dvdd = ""
csi_vol_iovdd =
csi_vol_dvdd =
csi_vol_avdd =
csi_vflip = 0
csi_hflip = 0
csi_flash_pol = 0
csi_facing = 1
csi_twi_id = 1
csi_twi_addr = 0x42
csi_pck = port:PG00<3><default><default><default>
csi_ck = port:PG01<3><default><default><default>
csi_hsync = port:PG02<3><default><default><default>
csi_vsync = port:PG03<3><default><default><default>
csi_d0 = port:PG04<3><default><default><default>
csi_d1 = port:PG05<3><default><default><default>
csi_d2 = port:PG06<3><default><default><default>
csi_d3 = port:PG07<3><default><default><default>
csi_d4 = port:PG08<3><default><default><default>
csi_d5 = port:PG09<3><default><default><default>
csi_d6 = port:PG10<3><default><default><default>
csi_d7 = port:PG11<3><default><default><default>
csi_reset = port:PH14<1><default><default><0>
csi_stby = port:PH19<1><default><default><0>
```

4.4.18 TV 出力 (TV OUT)

1. [TVOUT_para]

設定アイテム 意味

```
tvout_used=xx
tvout_channel_num=xx
tv_en=xx
```

2. [tv_in_para]

設定アイテム 意味

```
tvout_used=xx
tvout_channel_num=xx
```

4.4.19 SATA ディスク

1. [sata_para]

設定アイテム 意味

```
sata_used=xx
```

sata_power_en=xx

設定の例：

[sata_para]

Sata_used=1

Sata_power_en=port:PH17<1><default><default>

4.4.20 TF/MMC

1、[mmc0_para]

設定アイテム

意味

sdc_used=xx

SDC を使用かどうか。

1：使用

0：未使用

検査モード：

1、GPIO 検査

2、DATA 検査

3、検査なし、カードは常に付き

4、manual mode(from proc file system node)

sdc_detmode=xx

ビット幅：

bus_width=xx

1：1bit

4：4bit

sdc_d1=xx

sdc_data1の gpio 設定

sdc_do=xx

sdc_data0の gpio 設定

sdc_clk=xx

sdc_clkの gpio 設定

sdc_cmd=xx

sdc_cmdの gpio 設定

sdc_d3=xx

sdc_data3の gpio 設定

sdc_d2=xx

sdc_data2の gpio 設定

sdc_det=xx

sdc_detの gpio 設定

sdc_use_wp=xx

sdc の書き保護設定：

1：使用、0：未使用

sdc_wp=xx

sdc WP の gpio 設定

設定の例：

[mmc0_para]

sdc_used = 1

sdc_detmode = 1

sdc_buswidth = 4

sdc_clk = port:PF02<2><1><2><default>

sdc_cmd = port:PF03<2><1><2><default>

sdc_d0 = port:PF01<2><1><2><default>

sdc_d1 = port:PF00<2><1><2><default>

sdc_d2 = port:PF05<2><1><2><default>

```
sdc_d3 = port:PF04<2><1><2><default>
sdc_det = port:PH1<0><1><default><default>
sdc_use_wp = 0
sdc_wp =
sdc_isio = 0
sdc_regulator = "none"
```

2. [mmc1_para]

設定アイテム

意味

sdc_used=xx	SDC を使用かどうか。 1 : 使用 0 : 未使用
sdc_detmode=xx	検査モード： 1、GPIO 検査 2、DATA 検査 3、検査なし、カードは常に付き 4、manual mode(from proc file system node)
bus_width=xx	ビット幅： 1 : 1bit 4 : 4bit
sdc_d1=xx	sdc data1の gpio 設定
sdc_do=xx	sdc data0の gpio 設定
sdc_clk=xx	sdc clk の gpio 設定
sdc_cmd=xx	sdc cmd の gpio 設定
sdc_d3=xx	sdc data3の gpio 設定
sdc_d2=xx	sdc data2の gpio 設定
sdc_det=xx	sdc det の gpio 設定
sdc_use_wp=xx	sdc の書き保護設定： 1 : 使用、0 : 未使用
sdc_wp=xx	sdc WP の gpio 設定

設定の例:

```
[mmc1_para]
sdc_used = 0
sdc_detmode = 4
sdc_buswidth = 4
sdc_clk = port:PG00<2><1><2><default>
sdc_cmd = port:PG01<2><1><2><default>
sdc_d0 = port:PG02<2><1><2><default>
sdc_d1 = port:PG03<2><1><2><default>
sdc_d2 = port:PG04<2><1><2><default>
sdc_d3 = port:PG05<2><1><2><default>
sdc_det =
sdc_use_wp = 0
```

```
sdc_wp =
sdc_isio = 0
sdc_regulator = "none"
```

3. [mmc2_para]

設定アイテム

意味

sdc_used=xx

SDC を使用かどうか。

1 : 使用

0 : 未使用

検査モード :

1、GPIO 検査

2、DATA 検査

3、検査なし、カードは常に付き

4、manual mode(from proc file system node)

ビット幅 :

bus_width=xx

1 : 1bit

4 : 4bit

sdc_d1=xx

sdc data1の gpio 設定

sdc_do=xx

sdc data0の gpio 設定

sdc_clk=xx

sdc clk の gpio 設定

sdc_cmd=xx

sdc cmd の gpio 設定

sdc_d3=xx

sdc data3の gpio 設定

sdc_d2=xx

sdc data2の gpio 設定

sdc_det=xx

sdc det の gpio 設定

sdc_use_wp=xx

sdc の書き保護設定 :

1 : 使用、0 : 未使用

sdc_wp=xx

sdc WP の gpio 設定

設定の例 :

```
[mmc2_para]
sdc_used = 0
sdc_detmode = 3
sdc_buswidth = 4
sdc_cmd = port:PC06<3><1><2><default>
sdc_clk = port:PC07<3><1><2><default>
sdc_d0 = port:PC08<3><1><2><default>
sdc_d1 = port:PC09<3><1><2><default>
sdc_d2 = port:PC10<3><1><2><default>
sdc_d3 = port:PC11<3><1><2><default>
sdc_det =
sdc_use_wp = 0
sdc_wp =
sdc_isio = 0
sdc_regulator = "none"
```

4. [mmc3_para]

設定アイテム

意味

sdc_used=xx	SDC を使用かどうか。 1 : 使用 0 : 未使用
sdc_detmode=xx	検査モード : 1、GPIO 検査 2、DATA 検査 3、検査なし、カードは常に付き 4、manual mode(from proc file system node)
bus_width=xx	ビット幅 : 1 : 1bit 4 : 4bit
sdc_d1=xx	sdc data1の gpio 設定
sdc_do=xx	sdc data0の gpio 設定
sdc_clk=xx	sdc clk の gpio 設定
sdc_cmd=xx	sdc cmd の gpio 設定
sdc_d3=xx	sdc data3の gpio 設定
sdc_d2=xx	sdc data2の gpio 設定
sdc_det=xx	sdc det の gpio 設定
sdc_use_wp=xx	sdc の書き保護設定 : 1 : 使用、0 : 未使用
sdc_wp=xx	sdc WP の gpio 設定

設定の例:

```
[mmc3_para]
sdc_used = 1
sdc_detmode = 1
sdc_buswidth = 4
sdc_cmd = port:PI04<2><1><2><default>
sdc_clk = port:PI05<2><1><2><default>
sdc_d0 = port:PI06<2><1><2><default>
sdc_d1 = port:PI07<2><1><2><default>
sdc_d2 = port:PI08<2><1><2><default>
sdc_d3 = port:PI09<2><1><2><default>
sdc_det = port:PH16<0><1><default><default>
sdc_use_wp = 0
sdc_wp =
sdc_isio = 1
sdc_regulator = "none"
```

4.4.21 メモリスティック (memory stick)

1. [ms_para]

設定アイテム	意味
ms_used=xx	ms を使用かどうか。 1 : 使用 0 : 未使用
Ms_bs=xx	MS BS の GPIO 設定
ms_clk=xx	ms clk の gpio 設定
ms_d0=xx	ms data0の gpio 設定
ms_d1=xx	ms data1の gpio 設定
ms_d2=xx	ms data2の gpio 設定
ms_d3=xx	ms data3の gpio 設定
sdc_det=xx	sdc det の gpio 設定

設定の例 :

```
[ms_para]
ms_used = 0
ms_bs = port:PH06<5><default><default><default>
ms_clk = port:PH07<5><default><default><default>
ms_d0 = port:PH08<5><default><default><default>
ms_d1 = port:PH09<5><default><default><default>
ms_d2 = port:PH10<5><default><default><default>
ms_d3 = port:PH11<5><default><default><default>
ms_det=
```

4.4.22 SIM カード

1. [smc_para]

設定アイテム	意味
smc_used =xx	
smc_rst=xx	
smc_vppen=xx	
smc_vppp=xx	
smc_det =xx	
smc_vccen=xx	
smc_sck=xx	
smc_sda=xx	

設定の例;

```
[smc_para]
smc_used = 0
smc_rst = port:PH13<5><default><default><default>
```

```
smc_vppen = port:PH14<5><default><default><default>
smc_vppp = port:PH15<5><default><default><default>
smc_det = port:PH16<5><default><default><default>
smc_vccen = port:PH17<5><default><default><default>
smc_sck = port:PH18<5><default><default><default>
smc_sda = port:PH19<5><default><default><default>
```

4.4.23 PS/2 マウス

1. [ps2_0_para]

設定アイテム

ps2_used=xx

ps2_scl =xx

ps2_sda=xx

意味

PS/2を使用かどうか：

1：使用

0：未使用

PS/2 SCK の GPIO 設定

PS/2 SDA の GPIO 設定

設定の例：

```
[ps2_0_para]
```

```
ps2_used = 0
```

```
ps2_scl = port:PI20<2><1><default><default>
```

```
ps2_sda = port:PI21<2><1><default><default>
```

2. [ps2_1_para]

設定アイテム

ps2_used=xx

ps2_scl =xx

ps2_sda=xx

意味

PS/2を使用かどうか：

1：使用

0：未使用

PS/2 SCK の GPIO 設定

PS/2 SDA の GPIO 設定

設定の例：

```
[ps2_1_para]
```

```
ps2_used = 0
```

```
ps2_scl = port:PI14<3><1><default><default>
```

```
ps2_sda = port:PI15<3><1><default><default>
```

4.4.24 CAN バス

1. [can_para]

設定アイテム

can_used=xx

意味

CANを使用かどうか：

1：使用、0：未使用


```
can_tx =xx          CAN TX の GPIO 設定
can_rx=xx          CAN RX の GPIO 設定
```

設定の例

```
[can_para]
can_used = 0
can_tx = port:PA16<3><default><default><default>
can_rx = port:PA17<3><default><default><default>
```

4.4.25 マトリックスキーボード (key martrix)

1. [keypad_para]

設定アイテム

意味

kp_used=xx	KEYPAD 使用かどうか； 1：使用 0：未使用
kp_in_size=xx	KEYPAD 列幅
kp_out_size=xx	KEYPAD 行幅
kp_in0=xx	KEYPAD IN0の GPIO 設定
kp_in1=xx	KEYPAD IN1の GPIO 設定
kp_in2=xx	KEYPAD IN2の GPIO 設定
kp_in3=xx	KEYPAD IN3の GPIO 設定
kp_in4=xx	KEYPAD IN4の GPIO 設定
kp_in5=xx	KEYPAD IN5の GPIO 設定
kp_in6=xx	KEYPAD IN6の GPIO 設定
kp_in7=xx	KEYPAD IN7の GPIO 設定
kp_out0=xx	KEYPAD OUT0の GPIO 設定
kp_out1=xx	KEYPAD OUT1の GPIO 設定
kp_out2=xx	KEYPAD OUT2の GPIO 設定
kp_out3=xx	KEYPAD OUT3の GPIO 設定
kp_out4=xx	KEYPAD OUT4の GPIO 設定
kp_ou5=xx	KEYPAD OUT5の GPIO 設定
kp_out6=xx	KEYPAD OUT6の GPIO 設定
kp_ou7=xx	KEYPAD OUT7の GPIO 設定

設定の例:

```
[keypad_para]
kp_used = 0
kp_in_size = 8
kp_out_size = 8
kp_in0 = port:PH08<4><1><default><default>
kp_in1 = port:PH09<4><1><default><default>
kp_in2 = port:PH10<4><1><default><default>
kp_in3 = port:PH11<4><1><default><default>
```

```
kp_in4 = port:PH14<4><1><default><default>
kp_in5 = port:PH15<4><1><default><default>
kp_in6 = port:PH16<4><1><default><default>
kp_in7 = port:PH17<4><1><default><default>
kp_out0 = port:PH18<4><1><default><default>
kp_out1 = port:PH19<4><1><default><default>
kp_out2 = port:PH22<4><1><default><default>
kp_out3 = port:PH23<4><1><default><default>
kp_out4 = port:PH24<4><1><default><default>
kp_out5 = port:PH25<4><1><default><default>
kp_out6 = port:PH26<4><1><default><default>
kp_out7 = port:PH27<4><1><default><default>
```

4. 4. 26 USB コントロールフラグ

1. [usbc0]

設定アイテム

意味

usb_used=xx

usb enable フラグ (xx=1 or 0)。
1: システムの usb モジュールが使用できる、0: 使用できない。このフラグは具体的な usb コントローラモジュールに有効となる。

usb_port_type =xx

usb ポートの使用状況。(xx=0/1/2)
0: device only 1: host only 2: OTG

usb_detect_type=xx

usb ポートの検査方法。

usb_id_gpio=xx

0: 検査なし、1: vbus/id 検査

USB ID pin の設定。詳細は

usb_det_vbus_gpio=xx

「A20_SettingAndGpioControl.pdf」をご参照ください。

USB DET_VBUS pin ピンの設定。詳細は

usb_drv_vbus_gpio=xx

「A20_SettingAndGpioControl.pdf」をご参照ください。

USB DRY_VBUS pin ピンの設定。詳細は

「A20_SettingAndGpioControl.pdf」をご参照ください。

usb_host_init_state=xx

host only モードで Host ポートの初期化状態。

0: 初期化後 usb 動作しない

1: 初期化後 usb 動作する

設定の例:

```
[usbc0]
```

```
usb_used = 1
```

```
usb_port_type = 2
```

```
usb_detect_type = 1
usb_id_gpio = port:PH04<0><1><default><default>
usb_det_vbus_gpio = port:PH05<1><0><default><0>
usb_drv_vbus_gpio = port:PB09<1><0><default><0>
usb_restrict_gpio = port:PH00<1><0><default><0>
usb_host_init_state = 0
usb_restric_flag = 0
usb_restric_voltage = 3550000
usb_restric_capacity= 5
```

2. [usbc1]

設定アイテム

意味

usb_used=xx	usb enable フラグ (xx=1 or 0)。 1:システムのusb モジュールが使用できる、0:使用できない。このフラグは具体的なusb コントローラモジュールに有効となる。
usb_port_type =xx	usb ポートの使用状況。(xx=0/1/2) 0: device only 1: host only 2: OTG
usb_detect_type=xx	usb ポートの検査方法。 0:検査なし、1: vbus/id 検査
usb_id_gpio=xx	USB ID pin の設定。詳細は「A20_SettingAndGpioControl.pdf」をご参照ください。
usb_det_vbus_gpio=xx	USB DET_VBUS pin ピンの設定。詳細は「A20_SettingAndGpioControl.pdf」をご参照ください。
usb_drv_vbus_gpio=xx	USB DRV_VBUS pin ピンの設定。詳細は「A20_SettingAndGpioControl.pdf」をご参照ください。
usb_host_init_state=xx	host only モードで Host ポートの初期化状態。 0:初期化後 usb 動作しない 1:初期化後 usb 動作する

設定の例:

```
[usbc1]
usb_used = 1
usb_port_type = 1
usb_detect_type = 0
usb_drv_vbus_gpio = port:PH06<1><0><default><0>
usb_restrict_gpio =
usb_host_init_state = 1
usb_restric_flag = 0
```

3. [usbc2]

設定アイテム	意味
usb_used=xx	usb enable フラグ (xx=1 or 0)。 1:システムの usb モジュールが使用できる、0:使用できない。このフラグは具体的な usb コントローラモジュールに有効となる。
usb_port_type =xx	usb ポートの使用状況。(xx=0/1/2) 0: device only 1: host only 2: OTG
usb_detect_type=xx	usb ポートの検査方法。 0:検査なし、1: vbus/id 検査
usb_id_gpio=xx	USB ID pin の設定。詳細は「A20_SettingAndGpioControl.pdf」をご参照ください。
usb_det_vbus_gpio=xx	USB DET_VBUS pin ピンの設定。詳細は「A20_SettingAndGpioControl.pdf」をご参照ください。
usb_drv_vbus_gpio=xx	USB DRY_VBUS pin ピンの設定。詳細は「A20_SettingAndGpioControl.pdf」をご参照ください。
usb_host_init_state=xx	host only モードで Host ポートの初期化状態。 0: 初期化後 usb 動作しない 1: 初期化後 usb 動作する

設定の例;

```
[usbc2]
usb_used = 1
usb_port_type = 1
usb_detect_type = 0
usb_drv_vbus_gpio = port:PH03<1><0><default><0>
usb_restrict_gpio =
usb_host_init_state = 0
usb_restric_flag = 0
```

4.4.27 USB Device

1. [usb_feature]

設定アイテム	意味
vendor_id=xx	usb メーカーの ID
mass_storage_id=xx	U ディスクの ID
adb_id=xx	usb デバッグブリッジ ID
manufacturer_name=xx	usb メーカーの名前
product_name=xx	USB 製品名
serial_number=xx	USB シリアル番号

設定の例：

```
[usb_feature]
vendor_id = 0x18D1
mass_storage_id = 0x0001
adb_id = 0x0002
manufacturer_name = "USB Developer"
product_name = "Android"
serial_number = "20080411"
```

2. [msc_feature]

設定アイテム	意味
vendor_name=xx	Uディスクのメーカー名
product_name=xx	Uディスクの製品名
release=xx	リリースバージョン
luns=xx	Uディスクの論理的ユニット数 (PCからUディスクのドライブ数が見られる)

設定の例：

```
[msc_feature]
vendor_name = "USB 2.0"
product_name = "USB Flash Driver"
release = 100
luns = 2
```

4.4.28 重力センサー (G SENSOR)

1. [gsensor_para]

設定アイテム	意味
gsensor_used=xx	gsensor をサポートするかどうか 12C の BUS コントローラを選択。
gsensor_twi_id=xx	0 : TWI0、1 : TWI1、2 : TWI2
gsensor_int1=xx	割り込み1の GPIO 設定
gsensor_int2=xx	割り込み2の GPIO 設定

設定の例：

```
[gsensor_para]
gsensor_used = 0
gsensor_twi_id = 1
gsensor_int1 =
gsensor_int2 =
```

4. 4. 29 GPS

1. [gps_para]

設定アイテム	意味
gps_used=xx	
gps_spi_id =xx	
gps_spi_cs_num =xx	
gps_lradc=xx	
gps_clk=xx	
gps_sign=xx	
gps_mag	
gps_vcc_en	
gps_osc_en	
gps_rv_en	

設定の例 :

```
[gps_para]
gps_used = 0
gps_spi_id = 2
gps_spi_cs_num = 0
gps_lradc = 1
gps_clk = port:PI00<2><default><default><default>
gps_sign = port:PI01<2><default><default><default>
gps_mag = port:PI02<2><default><default><default>
gps_vcc_en = port:PC22<1><default><default><0>
gps_osc_en = port:PI14<1><default><default><0>
gps_rx_en = port:PI15<1><default><default><0>
```

4. 4. 30 WIFI (TFIO)

1. [TFio_wifi_para]

設定アイテム	意味
TFio_wifi_used=xx	TFIO WIFI を使用かどうか。 1 : 使用、0 : 未使用
TFio_wifi_TFc_id =xx	TFIO WIFI に使用された TFC 番号
TFio_wifi_mod_sel =xx	TFio_wifi モジュールの選択 : 0 - none、1 - swl - n20(wifi)、2 - usibm - 01(wifi+bt+fm)、3 - ar6302qfn、4 - apm6xxx、 5 - swb b23(wifi+bt+fm)
swl_n20_shdn=xx	swl_n20_shdn の GPIO 設定
swl_n20_host_wakeup=xx	swl_n20_host_wakeup の GPIO 設定
swl_n20_vdd_en=xx	swl_n20_vdd_en の GPIO 設定
swl_n20_vcc_en=xx	swl_n20_vcc_en の GPIO 設定

設定の例:

```
[TFio_wifi_para]
TFio_wifi_used = 1
TFio_wifi_TFc_id = 3
TFio_wifi_mod_sel = 1
TFio_wifi_shdn = port:PH09<1><default><default><0>
TFio_wifi_host_wakeup = port:PH10<1><default><default><1>
TFio_wifi_vdd_en = port:PH11<1><default><default><0>
TFio_wifi_vcc_en = port:PH12<1><default><default><0>
```

4.4.31 WIFI (USB)

1. [usb_wifi_para]

設定アイテム	意味
usb_wifi_used=xx	usb enable フラグ (xx=1 or 0)。 1 : usb wifi モジュールが使用できる、 0 : 使用できない。
usb_wifi_usbc_num=xx	usb wifi に使用される usb コントローラ 番号。xx は0、1、2から取られる。 usb_host_init_state と一緒に使わなければ ならない。例えば、xx=2、[usbc2]の usb_host_init_state は0になる。

設定の例:

```
[usb_wifi_para]
usb_wifi_used = 0
usb_wifi_usbc_num = 2
```

4.4.32 3G

1. [3g_para]

設定アイテム	意味
3g_used=xx	
3g_usbc_num=xx	
3g_uart_num=xx	
3g_pwr=xx	
3g_wakeup=xx	
3g_int =xx	

設定の例:

```
[3g_para]
3g_used = 0
3g_usbc_num = 2
```

```
3g_uart_num = 0
3g_pwr =
3g_wakeup =
3g_int =
```

4.4.33 gyroscope

1. [gy_para]

設定アイテム 意味

```
gy_used=xx
gy_twi_id=xx
gy_twi_addr=xx
gy_int1=xx
gy_int2=xx
```

設定の例：

```
[gy_para]
gy_used = 0
gy_twi_id = 1
gy_twi_addr = 0x00
gy_int1 = port:PH18<6><1><default><default>
gy_int2 = port:PH19<6><1><default><default>
```

4.4.33 光センサー

1. [ls_para]

設定アイテム 意味

```
ls_used=xx
ls_twi_id=xx
ls_twi_addr=xx
ls_int=xx
```

設定の例：

```
[ls_para]
ls_used = 0
ls_twi_id = 1
ls_twi_addr = 0x00
ls_int = port:PH20<6><1><default><default>
```


4.4.35 コンパス (compass)

1. [lcompass_para]

設定アイテム	意味
compass_used=xx	
compass_twi_id=xx	
compass_twi_addr=xx	
compass_int=xx	

設定の例:

```
[compass_para]
compass_used = 0
compass_twi_id = 1
compass_twi_addr = 0x00
compass_int = port:PI13<6><1><default><default>
```

4.4.36 bluetooth

1. [bt_para]

設定アイテム	意味
bt_used=xx	BLUETOOTH を使用かどうか。 1:使用、0:使用しない
bt_uart_id=xx	BLUETOOTH1 に使用された UART 番号
bt_wakeup =xx	BT WAKEUP の GPIO 設定
bt_gpio=xx	BT の選択できる GPIO 設定
bt_rst=xx	BT RESET の GPIO 設定

設定の例;

```
[bt_para]
bt_used = 0
bt_uart_id = 2
bt_wakeup = port:PI20<1><default><default><default>
bt_gpio = port:PI21<1><default><default><default>
bt_rst = port:PB05<1><default><default><default>
```

4.4.37 デジタルオーディオバス (I2S)

1. [i2s_para]

設定アイテム	意味
i2s_used=xx	1:このモジュールをロードする 0:ロードしない。
i2s_channel=xx	チャンネル制御
i2s_mclk =xx	I2SMCLK 信号の GPIO 設定

i2s_bclk=xx	I2sBCLK 信号の GPIO 設定
i2s_lrclk =xx	I2sLRCK 信号の GPIO 設定
i2s_dout0	I2S out0の GPIO 設定
i2s_dout1	未使用
i2s_dout2	未使用
i2s_dout3	未使用
i2s_din	I2sIN 信号の GPIO 設定

設定の例：

```
i2s_used = 0
i2s_channel = 2
i2s_mclk = port:PB5<2><1><default><default>
i2s_bclk = port:PB6<2><1><default><default>
i2s_lrclk = port:PB7<2><1><default><default>
i2s_dout0 = port:PB8<2><1><default><default>
i2s_dout1 =
i2s_dout2 =
i2s_dout3 =
i2s_din = port:PB12<2><1><default><default>
```

4. 4. 38 デジタルオーディオバス (S/PDIF)

1. [spdif_para]

設定アイテム

意味

```
spdif_used=xx
spdif_mclk=xx
spdif_dout=xx
spdif_din=xx
```

4. 4. 38 スピーカー制御

1. [audio_para]

設定アイテム

意味

audio_used=xx	audio のスピーカーを設定する。 1 : オープン (デフォルト) 0 : クローズ
audio_pa_ctrl=xx	スピーカーの gpio インタフェース制御
audio_lr_change=xx	1 : 一部分のチャンネルを逆転する 0 : 普通

設定の例：

```
[audio_para]
```

```
audio_used = 1
audio_pa_ctrl = port:PH15<1><default><default><0>
audio_lr_change=xx = 0
```

4.4.40 赤外線(ir)

1. [ir_para]

設定アイテム	意味
ir_used=xx	
ir0_rx=xx	

設定の例：

```
[ir_para]
ir_used = 1
ir_rx = port:PB04<2><default><default><default>
```

4.4.41 PMU パワー

1. [pmu_para]

設定アイテム	意味
pmu_used=xx	pmu enable フラグ (xx=1 or 0) 1 : 使用、0 : 未使用
pmu_twi_addr=xx	pmu デバイスのアドレス
pmu_twi_id=xx	pmu にロードされた i2c コントローラの番号。0 : twi0、1 : twi1、2 : twi2
pmu_irq_id=xx	pmu の割り込み番号。 0:NMI、1 : 1 番目割り込み、2 : 2 番目割り込み...
pmu_battery_rdc=xx	バッテリー内部抵抗、mΩ、実際テスト結果によって書く
pmu_battery_cap=xx	バッテリー容量、mAh、実際テスト結果によって書く
pmu_init_chgcur=xx	起動充電電流を設定、mA、 300/400/500/600/700/800/900 /1000/1100/1200/1300 /1400/1500/1600/1700/1800
pmu_earlysuspend_chgcur=xx	画面オフの充電電流を設定、mA、 300/400/500/600/700/800/900 /1000/1100/1200/1300 /1400/1500/1600/1700/1800
pmu_suspend_chgcur=xx	休止状態の充電電流、mA、 300/400/500/600/700/800/900 /1000/1100/1200/1300

	/1400/1500/1600/1700/1800
pmu_resume_chgcur=xx	ウェークアップの充電電流を設定、mA、 300/400/500/600/700/800/900 /1000/1100/1200/1300 /1400/1500/1600/1700/1800
pmu_shutdown_chgcur=xx	シャットダウンの充電電流を設定。mA、 300/400/500/600/700/800/900 /1000/1100/1200/1300 /1400/1500/1600/1700/1800
pmu_init_chgvol=xx	充電対象の電圧を設定。mV、 4100/4150/4200/4360
pmu_init_chgend_rate=xx	充電終了の電流比率を設定。%, 10, 15
pmu_init_chg_enabled=xx	充電機能を設定。 0 : クローズ、1 : オープン
pmu_init_adc_freq=xx	adc のサンプリング周波数を設定。Hz, 25/50/100/200
pmu_init_adc_freqc=xx	クーロンメータのサンプリング周波数を 設定。Hz, 25/50/100/200
pmu_init_chg_pretime=xx	プリチャージのタイムアウト時間を設定。 min, 40/50/60/70
pmu_init_chg_csttime=xx	定電流充電のタイムアウト時間を設定。 min, 360/480/600/720
pmu_bat_para1=xx	無負荷バッテリー電圧に対応する比率を 設定、%。
pmu_bat_para2=xx	無負荷バッテリー電圧に対応する比率を 設定、%。
pmu_bat_para3=xx	無負荷バッテリー電圧に対応する比率を 設定、%。
pmu_bat_para4=xx	無負荷バッテリー電圧に対応する比率を 設定、%。
...	
pmu_bat_para16=xx	無負荷バッテリー電圧に対応する比率を 設定、%。
pmu_usbvol_limit=xx	usb の電圧制限機能を設定。 0 : クローズ、1 : オープン
pmu_usbvol=xx	usb の制限電圧を設定。mV、 4000/4100/4200/4300/4400/4500/4600/4700
pmu_usbcurlimit=xx	usb の電流制限機能を設定。 0 : クローズ、1 : オープン
pmu_usbcurlimit=xx	usb の制限電流を設定。mA, 100/500/900
pmu_pwroff_vol=xx	起動時のハードウェアの保護電圧を設定。 mV, 2600/2700/2800/2900/3000/3100/3200/3300

pmu_pwron_vol=xx	パワーオン状態のハードウェアの保護電圧を設定。mV, 2600/2700/2800/2900/3000/3100/3200/3300
pmu_pekoff_time=xx	ハードウェアのシャットダウン時間を設定。ms、4000/6000/8000/10000
pmu_pekoff_en=xx	ハードウェアのシャットダウン機能を設定。0：クローズ、1：オープン
pmu_peklong_time=xx	長押しキーの割り込み時間を設定。ms, 1000/1500/2000/2500
pmu_pekon_time=xx	起動時間を設定。ms, 128/1000/2000/3000
pmu_pwrok_time=xx	パワー起動後の pwrok 信号遅延時間を設定。ms, 8/64
pmu_pwrnoe_time=xx	n_oe が低いから高くなった後、シャットダウン遅延時間を設定。ms, 128/1000/2000/3000
pmu_intotop_en=xx	過熱シャットダウン機能を設定。 0：クローズ、1：オープン 他のケースのために設定したパラメータ。
pmu_used2=xx	pmu_adpdet [~] pmu_shutdown_chgcur2と合わせて実行する。0：クローズ；1：オープン
pmu_adpdet=xx	他のケースのために設定したパラメータ。 アダプタ検出ポートの設定。詳細は上述の gpio 設定にご参考ください。
pmu_init_chgcur2=xx	他のケースのために設定したパラメータ。 アダプタに挿し込む時、起動充電電流の設定。pmu_init_chgcur と同じ。
pmu_earlysuspend_chgcur2=xx	他のケースのために設定したパラメータ。 アダプタに挿し込む時、画面オフ充電電流の設定。pmu_earlysuspend_chgcur と同じ。
pmu_suspend_chgcur2=xx	他のケースのために設定したパラメータ。 アダプタに挿し込む時、休止状態の充電電流の設定。pmu_suspend_chgcur と同じ。
pmu_resume_chgcur=xx	他のケースのために設定したパラメータ。 アダプタに挿し込む時、ウェイクの充電電流の設定。pmu_resume_chgcur と同じ。
pmu_shutdown_chgcur2=xx	他のケースのために設定したパラメータ。 アダプタに挿し込む時、シャットダウンの充電電流の設定。pmu_shutdown_chgcur と同じ。
pmu_suspendpwroff_vol=xx	休止シャットダウンの電圧の設定。mV、 2867-4200、6mV で1段階。

```
pmu_batdeten=xx
```

バッテリー検出機能の設定。
0 : クローズ、1 : オープン

注意 : pmu_used2~ pmu_shutdown_chgcur2のパラメータは他のケースのために設定したインタフェースである。削除しても構わない。

設定の例 :

```
[pmu_para]
pmu_used = 1
pmu_twi_addr = 0x34
pmu_twi_id = 0
pmu_irq_id = 32
pmu_battery_rdc = 100
pmu_battery_cap = 3200
pmu_init_chgcur = 300
pmu_earlysuspend_chgcur = 600
pmu_suspend_chgcur = 1000
pmu_resume_chgcur = 300
pmu_shutdown_chgcur = 1000
pmu_init_chgvol = 4200
pmu_init_chgend_rate = 15
pmu_init_chg_enabled = 1
pmu_init_adc_freq = 100
pmu_init_adc_freqc = 100
pmu_init_chg_pretime = 50
pmu_init_chg_csttime = 720

pmu_bat_para1 = 0
pmu_bat_para2 = 0
pmu_bat_para3 = 0
pmu_bat_para4 = 0
pmu_bat_para5 = 5
pmu_bat_para6 = 8
pmu_bat_para7 = 11
pmu_bat_para8 = 22
pmu_bat_para9 = 33
pmu_bat_para10 = 43
pmu_bat_para11 = 50
pmu_bat_para12 = 59
pmu_bat_para13 = 71
pmu_bat_para14 = 83
pmu_bat_para15 = 92
pmu_bat_para16 = 100

pmu_usbvol_limit = 1
```

```
pmu_usbcur_limit = 0
pmu_usbvol = 4000
pmu_usbcur = 0

pmu_usbvol_pc = 4400
pmu_usbcur_pc = 500

pmu_pwroff_vol = 3300
pmu_pwron_vol = 2900

pmu_pekoff_time = 6000
pmu_pekoff_en = 1
pmu_peklong_time = 1500
pmu_pekon_time = 1000
pmu_pwrok_time = 64
pmu_pwrnoe_time = 2000
pmu_intotp_en = 1
pmu_used2 = 0
pmu_adpdet = port:PH02<0><default><default><default>
pmu_init_chgcur2 = 400
pmu_earlysuspend_chgcur2 = 600
pmu_suspend_chgcur2 = 1200
pmu_resume_chgcur2 = 400
pmu_shutdown_chgcur2 = 1200

pmu_suspendpwroff_vol = 3500
pmu_batdeten = 1
```

第五章 Android アプリ環境の構築

5.1 Ubuntu で android アプリ環境を構築

5.1.1 JDK をインストール

まず、oracleのウェブサイトからlinux バージョンのJDKをダウンロードする。
(<http://www.oracle.com/technetwork/java/javase/downloads/jdk-6u26-download-400750.html>)、ここでLinux x86 - Self Extracting Installer を選ぶ。このファイルをディレクトリにコピーし、直接にインストールする。

```
./jdk-6u26-linux-i586.bin
```

失敗の場合に、実行権限を添加する。

```
chmod +x ./jdk-6u26-linux-i586.bin
```

プログラムが自動的に jdk1.6.0_26 ディレクトリにインストールする。インストール終了、次に環境変数を設定する。実行：

```
sudo gedit /etc/profile
```

実行中に、(gedit:4849):Gtk-WARNING **: Attempting to store changes into /toot/.local というエラーが提示される。Sudo mkdir -p /root/.local/share/ つまりスーパーユーザーとして (sudo コマンド) ディレクトリを作成し、問題を解決できる。

設定ファイルの最後に次の部分を追加する。

```
export JAVA_HOME=/home/jackwong/jdk1.6.0_26
export JRE_HOME=/home/jackwong/jdk1.6.0_26/jre
export PATH=/home/jackwong/jdk1.6.0_26/bin:$PATH
export CLASSPATH=../home/jackwong/jdk1.6.0_26/lib:/home/jackwong/jdk1.6.0_26/jre/lib
```

保存してから、現在のユーザーをログオフし、再度ログインする。この時、環境変数は有効となる。

インストールが成功となるかどうかを確認する場合に、下記コマンドを実行：

```
java -version
```



```
java version "1.6.0_26"  
  
Java(TM) SE Runtime Environment (build 1.6.0_26-b06)  
  
Java HotSpot(TM) Server VM (build 20.0-b11, mixed mode)
```

このような情報が現れたら、成功。

注意：Ubuntu は openjdk がインストールされたことをデフォルトとする。jdk をインストールしていない場合に、java-version を実行してもエラーが出ない。しかし、両方の報告情報が違う。

5.1.2 eclipse をインストール

eclipse+ADTは公式に推奨された開発環境である。ダウンロードサイト：
<http://www.eclipse.org/downloads/>。ここで、32bit linux をダウンロードした。
Eclipse IDE for Java Developers。
ダウンロードしたファイルは eclipse - java - juno - SR1 - linux - gtk. tar. gz
解凍し、コマンドを実行する。

```
tar zxvf eclipse-java-juno-SR1-linux-gtk.tar.gz
```

解凍が成功後、eclipse をインストール済みのディレクトリがある。このディレクトリに入って、eclipse を実行し、eclipse を起動させる。

デスクトップにショートカットを作成する。

5.1.3 Android TFK をインストール

Android TFK Starter Package はTFKの基本的なツールである。これを利用して、他の必要なツールがダウンロードできる。androidセルフファイルを実行し、Android TFK and AVDManagerを起動させる。
ダウンロードサイト：<http://developer.android.com/TFk/index.html>
ここで最新バージョンの android - TFk_r21.1 - linux. tgz をダウンロードした。
次に解凍する。

```
tar zxvf android-TFk_r21.1-linux.tgz
```

解凍終了、現在のディレクトリには android - TFk - linux_x86 がある。このディレクトリは TFK の基本的なツールである。

5.1.4 ADT (Android Development Tools) をインストール

ADT は、google が android アプリ開発のために開発された Eclipse のプラグインである。Eclipse を起動し、Help > Install New Software... を選ぶ。右上隅の Add ボタンをクリックし、ポップアップした Add Repository ダイアログボックスの name 欄に「ADT Plugin」を書き込む。Location 欄には二つのオプションがあり、一つはダウンロードのサイトを記入し、https://dl-ssl.google.com/android/eclipse/。もう一つのオプションは、もし ADT Plugin を既にダウンロードした場合に、直接に Archive ボタンをクリックし、プラグインがあるパッケージを見つける。ここで一番目の方法を使った。二番目の方法は、まず、Developer TFK Tools 前の checkbox を選択する。次に、next をクリックし、インストールし始める。その中に、ソフトウェアのインストールリストが現れて、next をクリックする。最後に、現れた license に agree を選び、finish をクリックし、完成。eclipse を再起動する。

5.1.5 ADT (Android Development Tools) を設定

eclipse における ADT プラグインのパスを修正する。(ADT に Android TFK ディレクトリをインストールした)。

eclipse を起動させ、window>preferences を選択する。

ポップアップした preferences ダイアログボックスの左側に android タグを選択し、右側の TFK location に元の Android TFK ディレクトリを選択する。例えば、ここで/home/AAA/android-TFK-linux_x86 を使用した (AAA はユーザー名)。それから、apply, ok を順次をクリックする。

この過程に、google の満足度アンケートに参加かどうかというダイアログボックスがポップアップする。process ボタンをクリックする。

インストール終了、help>check for updates を利用して、ADT を更新する。

5.1.6 Android OS システムとコントローラを追加

今まで、大部分の設定が完成した。しかし、android アプリを開発するために、Android OS システムを追加する必要がある (アプリケーションをシミュレートすることに使用される)。その上に、TFK ツールパッケージにおける Android TFK Manager を利用して他のコントローラをインストールする必要。直接に eclipse に Window > Android TFK Manager 又は、TFk > tools > android を運行してインストールできる。

次のコントローラを推奨する。

TFK Tools、TFk starter package をインストールする事で既にインストール完了。

TFK Platform - tools、インストール必要。

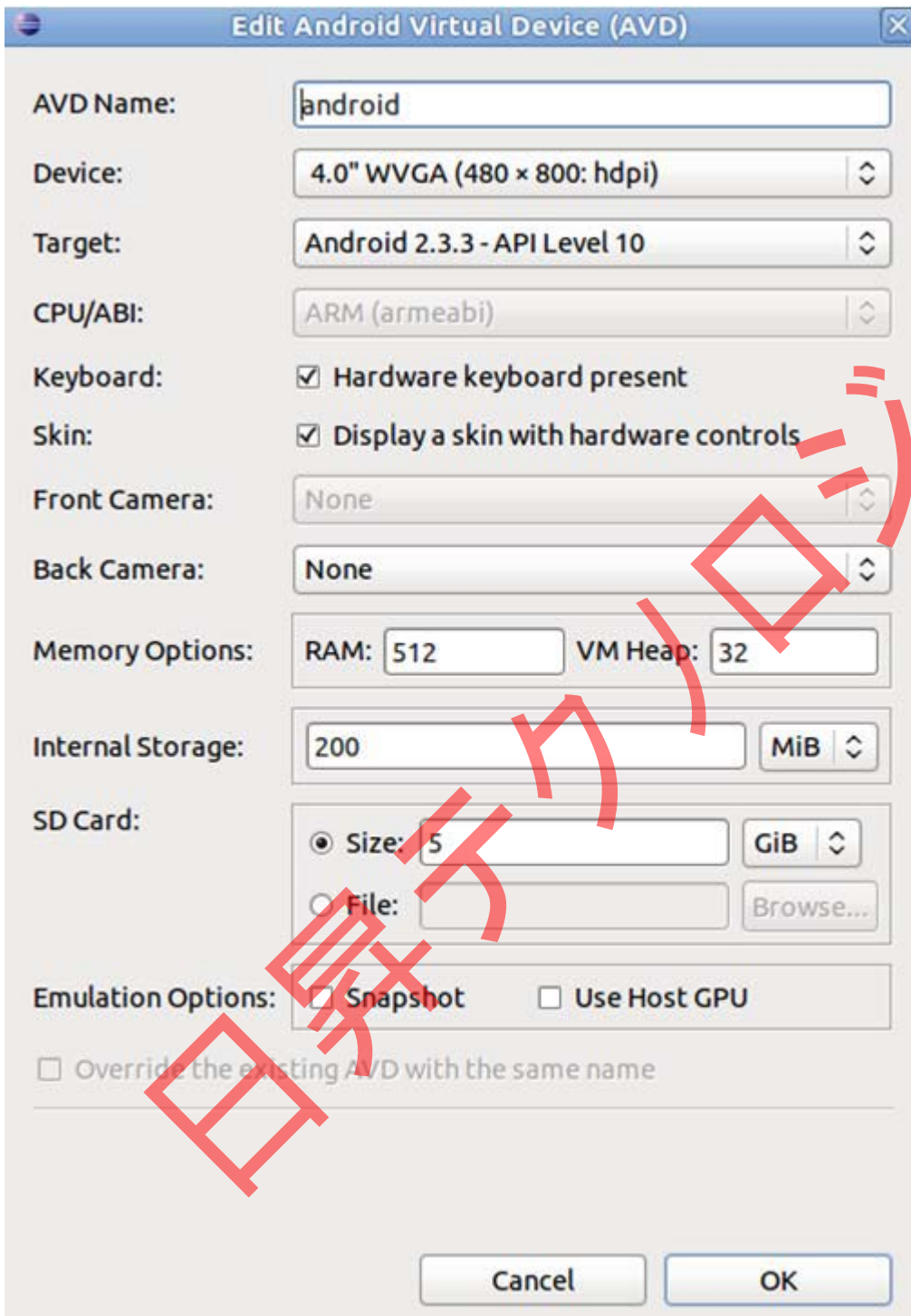
TFK platform。少なくとも一つ必要。OS システム。同時に 2.2、3.1 をダウンロードしても構わない。もし各種のプラットフォームをシミュレートしたら、各種バージョンの android をダウンロードする必要がある。

TFK Platform - tools、Android 2.3.3、Android 4.1、Extras をダウンロードする。

ダウンロード終了、android の開発環境の構築が完成した。

5.1.7 シミュレータ設定を行う

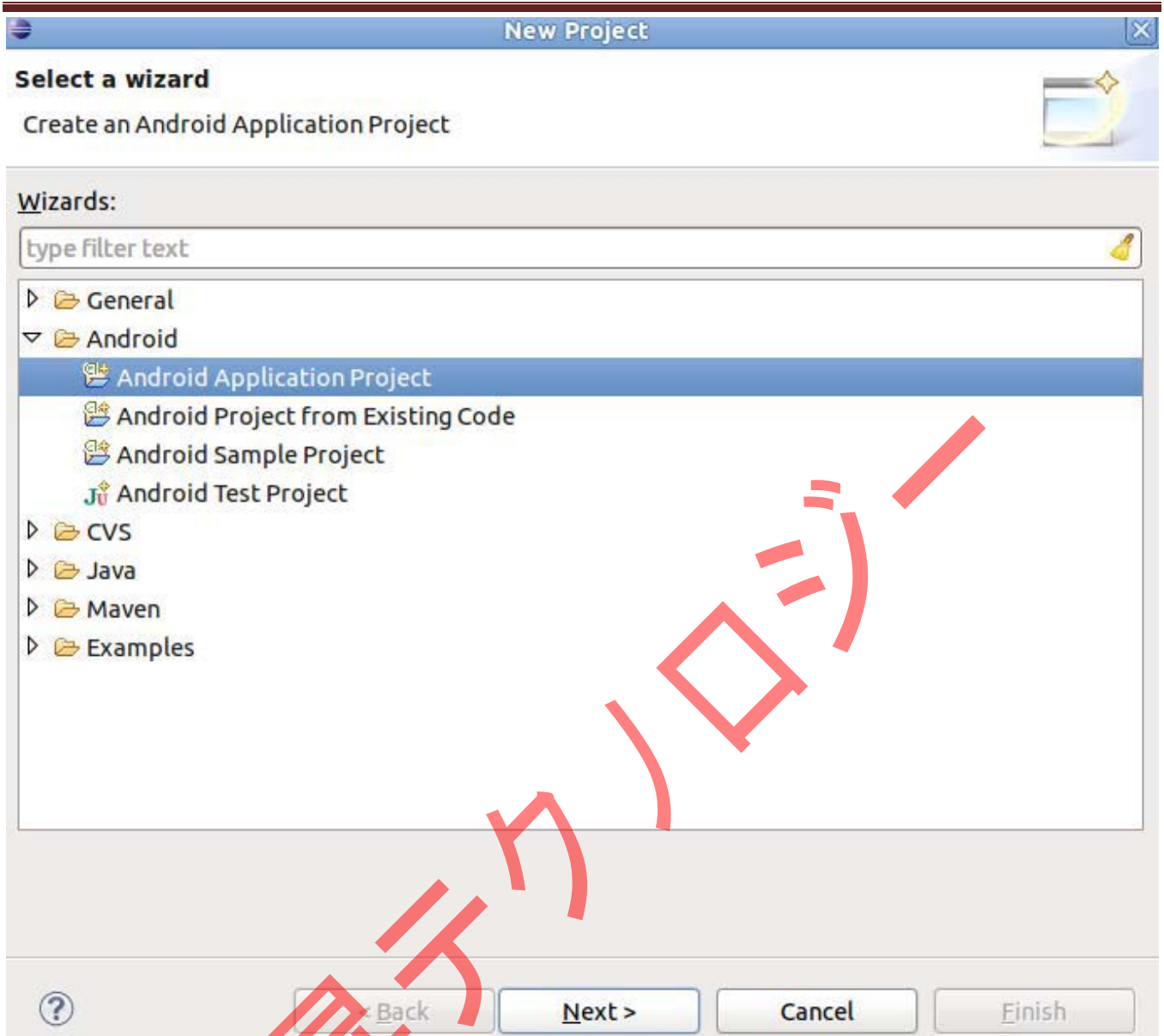
windows ->Android Virtual Device Manager を選択し、new をクリックする。



完成後、start をクリックする。パーチャルマシンを運行し、言語を選択する。

5.1.8 hello, android テストプログラム

eclipse を起動し、File->New->Project... を選択し、android の Android Application Project を選択する。



Application name: Hello, Android、アプリプログラムの名前、同時にプログラムが実行する時にスクリーンに表示した文字である。最初の文字は大文字となる必要である。

Project name: HelloAndroid、プロジェクト名、ファイルを含んでいるディレクトリの名前。

Package name: my.android。

他の所にはデフォルトのまま。

全て next を選択する。Activity name を Helloandroidactivity に設定する。(注意：最初の文字は大文字となる必要である)

完成後、my.android に Helloandroidactivity.java を見つける。オープンして、次のように onCreate 方法のコードを更新する。

```
import android.widget.TextView;

public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

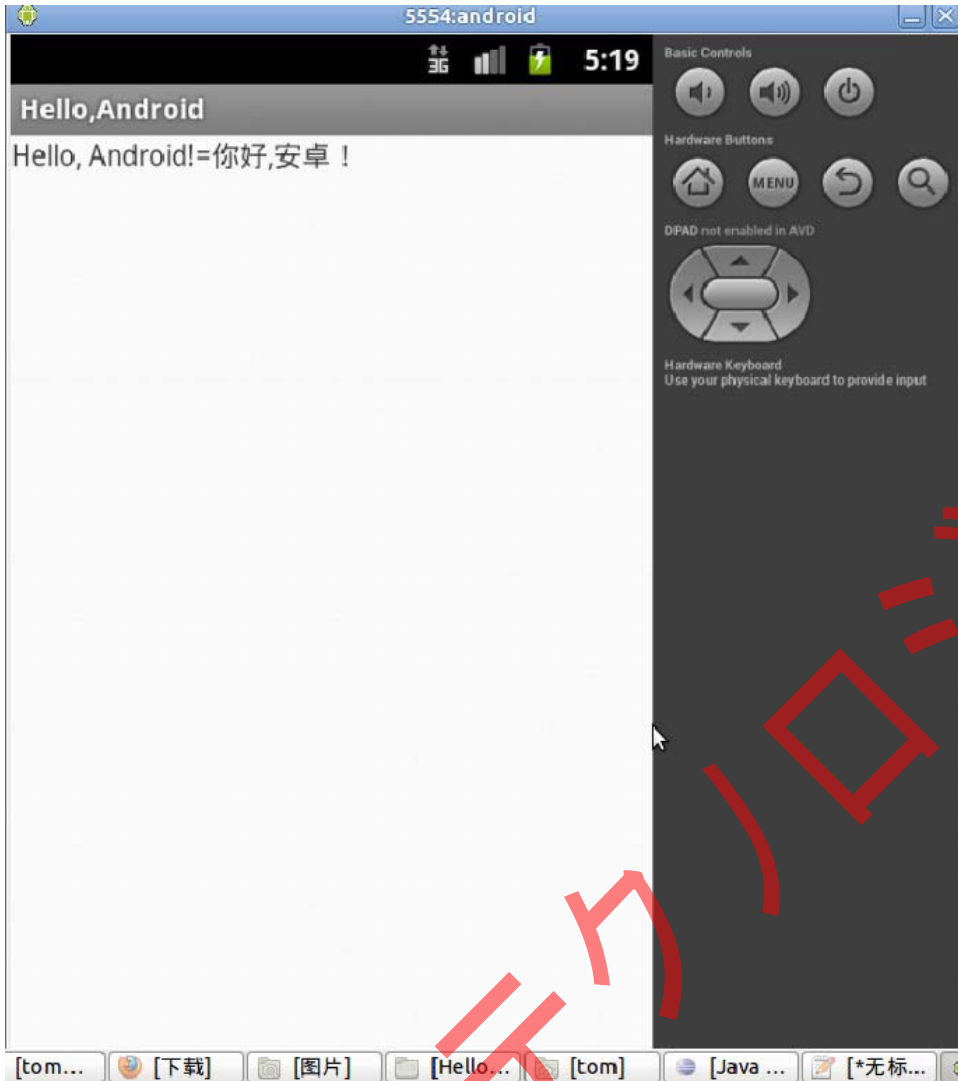
    TextView tv = new TextView(this);

    tv.setText("Hello, Android!=你好,安卓!");

    setContentView(tv);

}
```

実行する。暫くすると、android パーチャルマシンが表示する。そして、このプログラムが実行する。



5.2 windowsXP で android アプリ開発環境を構築

一、ツールをダウンロードする。

- 1、JDK : <http://java.sun.com/javase/downloads/index.jsp>
- 2、Eclipse : <http://www.eclipse.org/downloads/>
- 3、Android TFK : <http://developer.android.com>

次のバージョンをダウンロードした。

jdk - 7u2 - windows - i586

eclipse - TFK - 3.7.1 - win32

installer_r16 - windows.exe (Android TFK) (インストールするに約 6~7 時間がかかる)

二、インストール

JDK、TFK をインストールする。Eclipse を解凍する。

JDK がインストール終了後、次のように環境変数を設定する。

- 1、インストールディレクトリは C:\Program Files\Java\jdk1.7.0_02 だとする。
- 2、インストール終了、「My Computer」「プロパティ」を右クリックする。

3、「高級」を選択し、「環境変数」をクリックする。

4、「システム変数」に、JAVA_HOME, PATH, CLASSPATH 三つのプロパティを設定する（大文字でも小文字でも使える）。もう既にある場合に、「編集」をクリックする。ない場合に、「新規」をクリックする。

5、JAVA_HOME は JDK のインストールしたパスを指定する。即、C:\Program Files\Java\jdk1.7.0_02

PATH はシステムが任意のパスに JAVA コマンドを識別させる。次のように設定する：

```
%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin
```

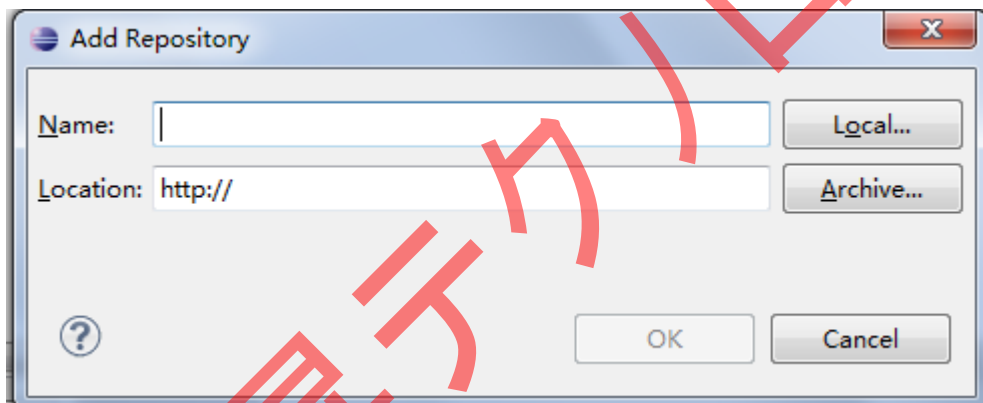
CLASSPATH は java のためにクラスパス(class or lib)。classpathにある場合、java コマンドが識別できる。次のように設定する：

```
.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar (.を追加必要、現在のパスを表明する)  
%JAVA_HOME%は前述したJAVA_HOMEを引用。
```

TFK インストール終了、環境変数を設定する。Path にC:\Program Files\Android\android-sdk\tools;を追加する。

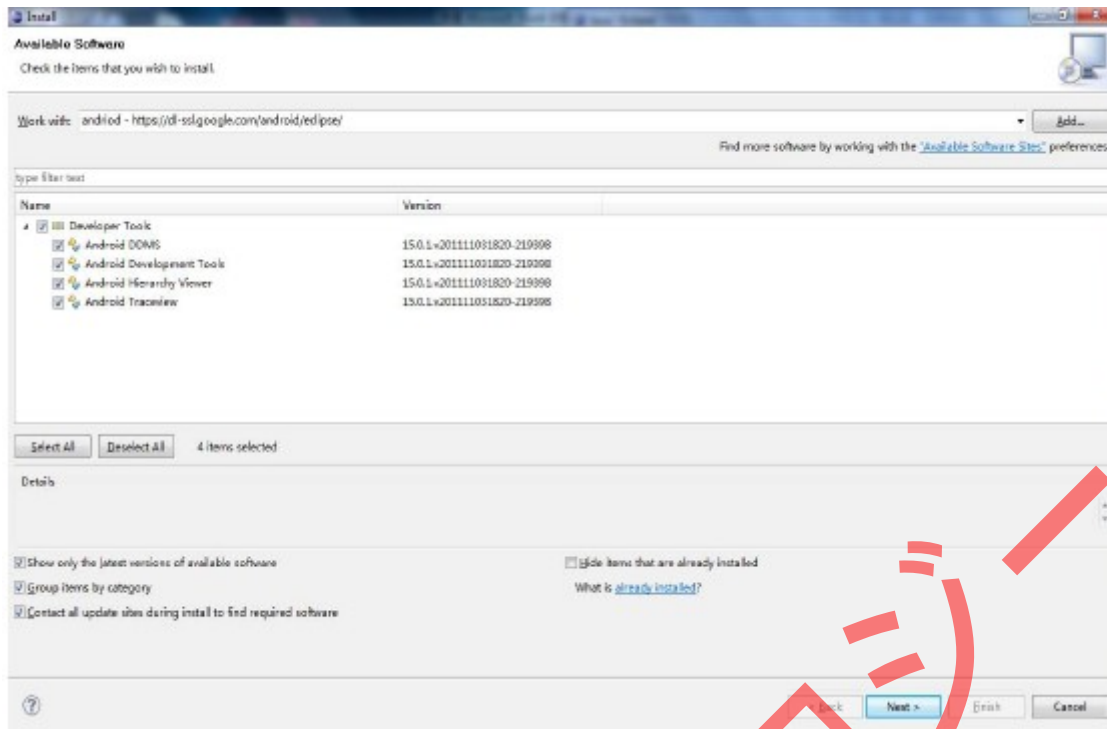
三、Eclipseを設定。

1) Eclipse をクリックし、 help ->Install New Software を選択する。 Add ボタンをクリックし、次の画面が出る：



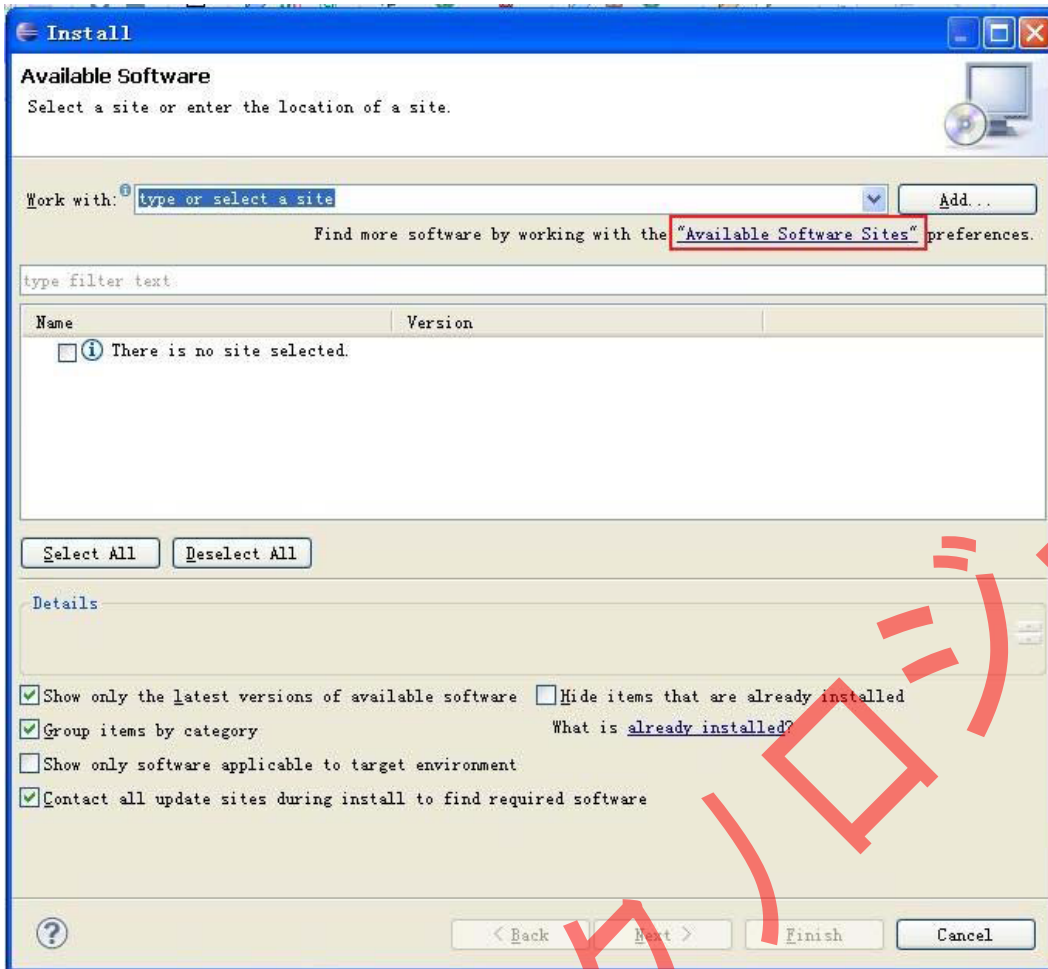
2) 上図の通りに、URL を入力する：https://dl-ssl.google.com/android/eclipse
名前：Android(カスタマイズ可能)

3) 全部選択する

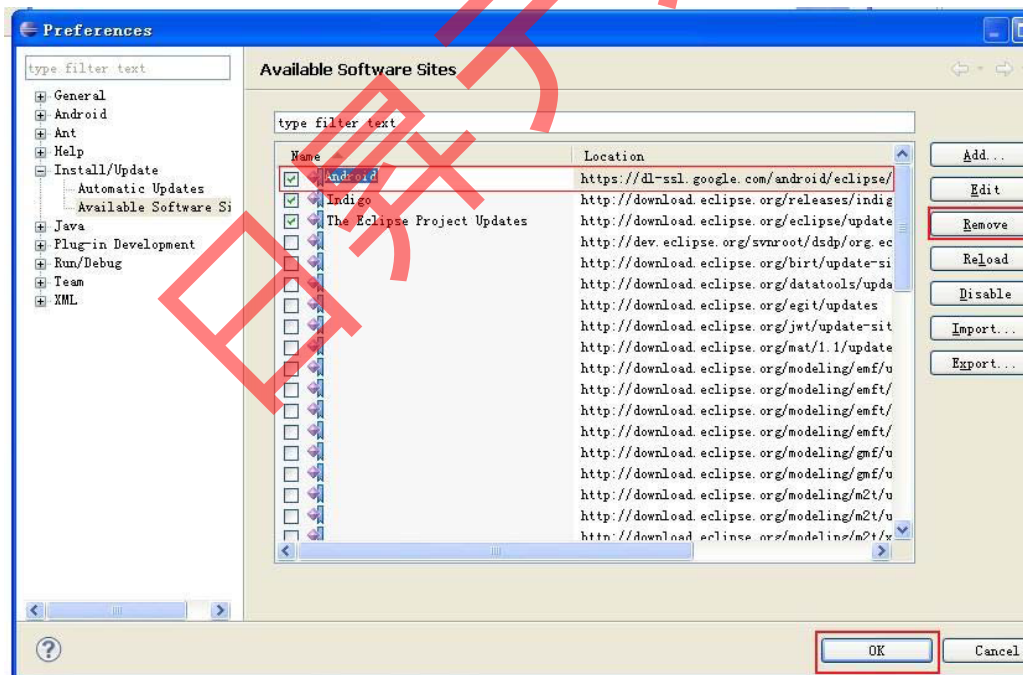


4) Eclipse を再起動するまでに、提示の通りに操作する。

インストールする時、「org.eclipse.core.runtime3.6.0」がなし又は見つけれない問題が起こった起こったケースもある。その時、最新バージョンのEclipseをダウンロードし、問題を解決したが、新しい問題を起こった。1)、2) 手順を再度に実行し、「Duplicate location」エラーが出た。解決方法は : help -> Install New Software をクリックし、



Available software sites をクリックし、次のインターフェースに入る。

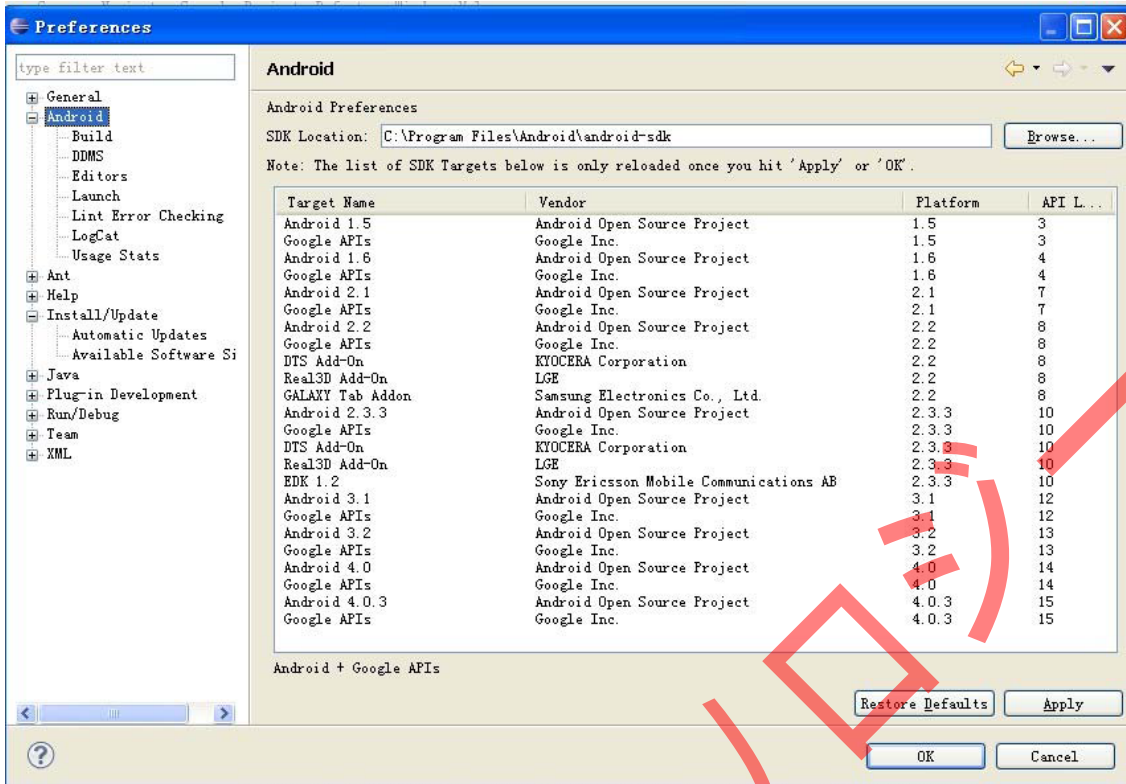


android を選択し、remove ボタンをクリックする。Ok.

これまで、Eclipse に android の開発プラグインをインストール終了。

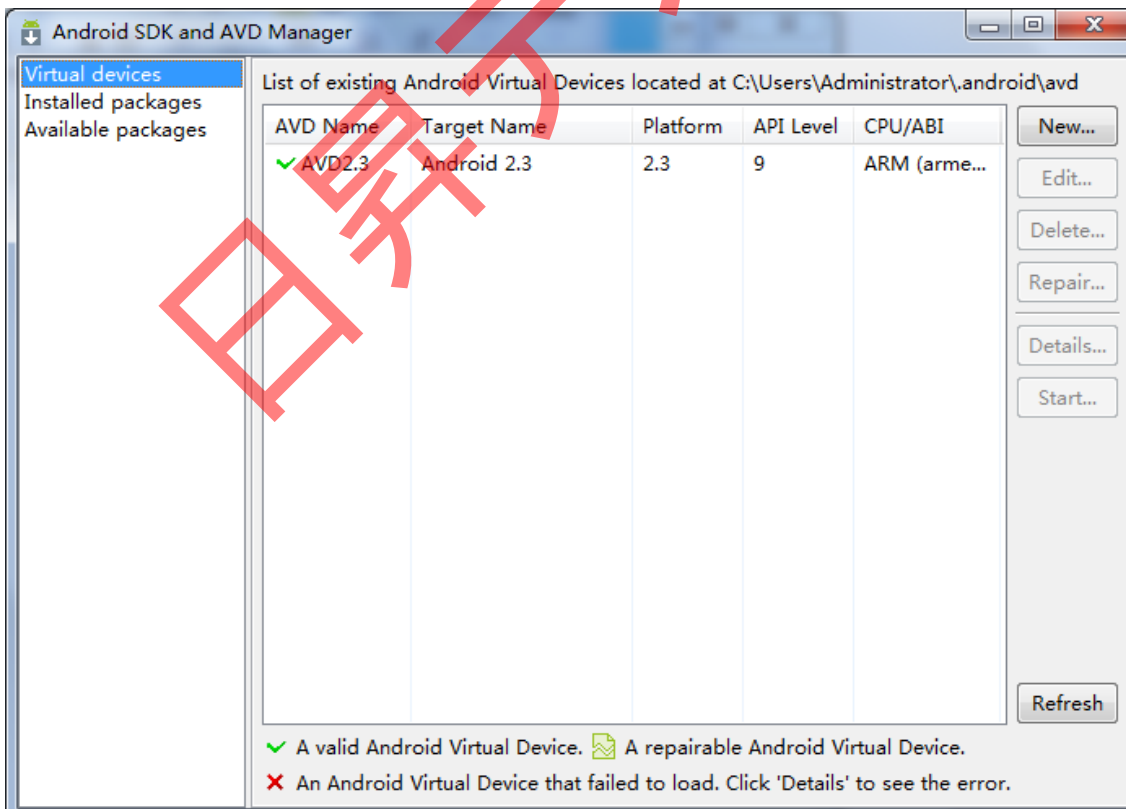
次に、Eclipse に android TFK を設定する。

1) メニューwindow->preferences をクリックし、次のインタフェースに入る。

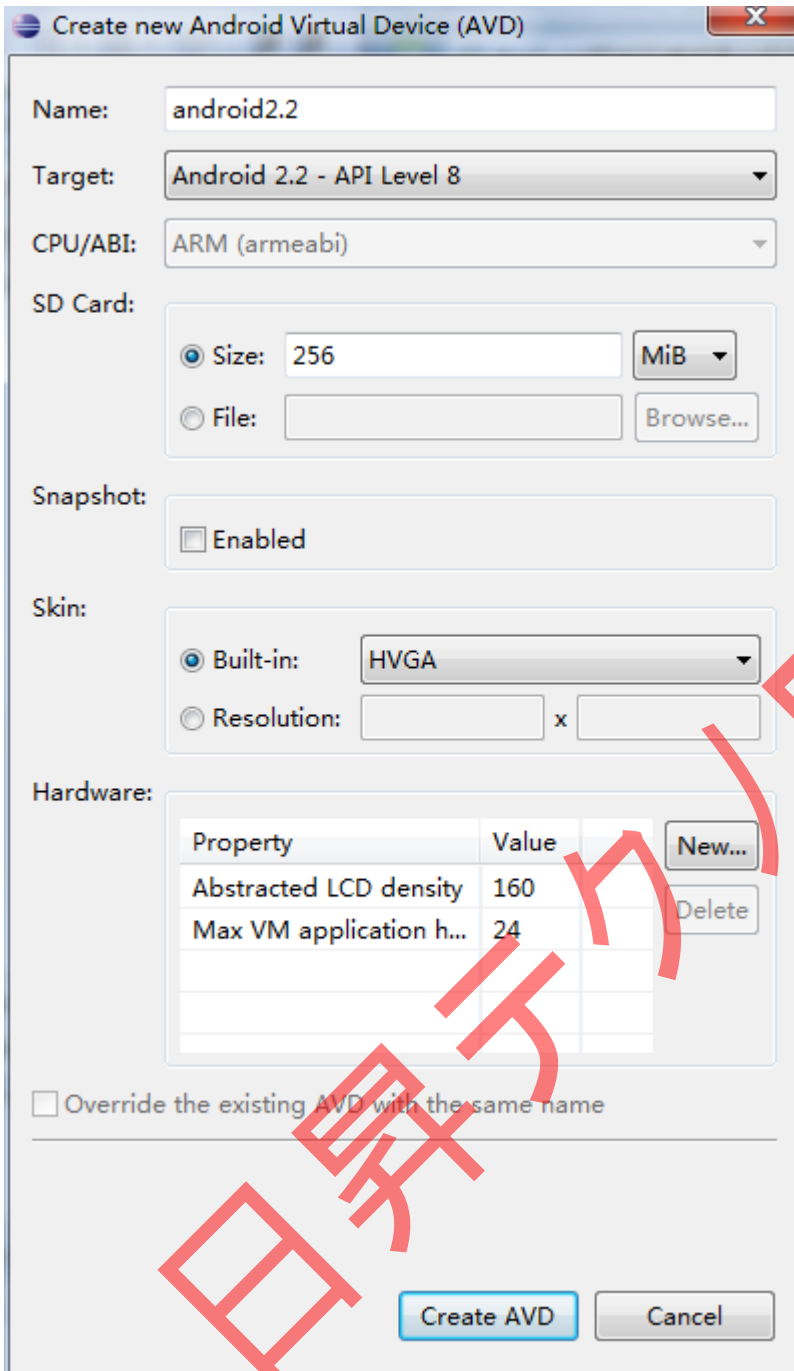


Location に android TFK のパスを書き込む。

上述が終わったら、AVD(android virtual device)を作成する。下図の通りに、virtual device を選択すると、



New をクリックと、



Property	Value
Abstracted LCD density	160
Max VM application h...	24

Name (パーチャルマシン、任意)

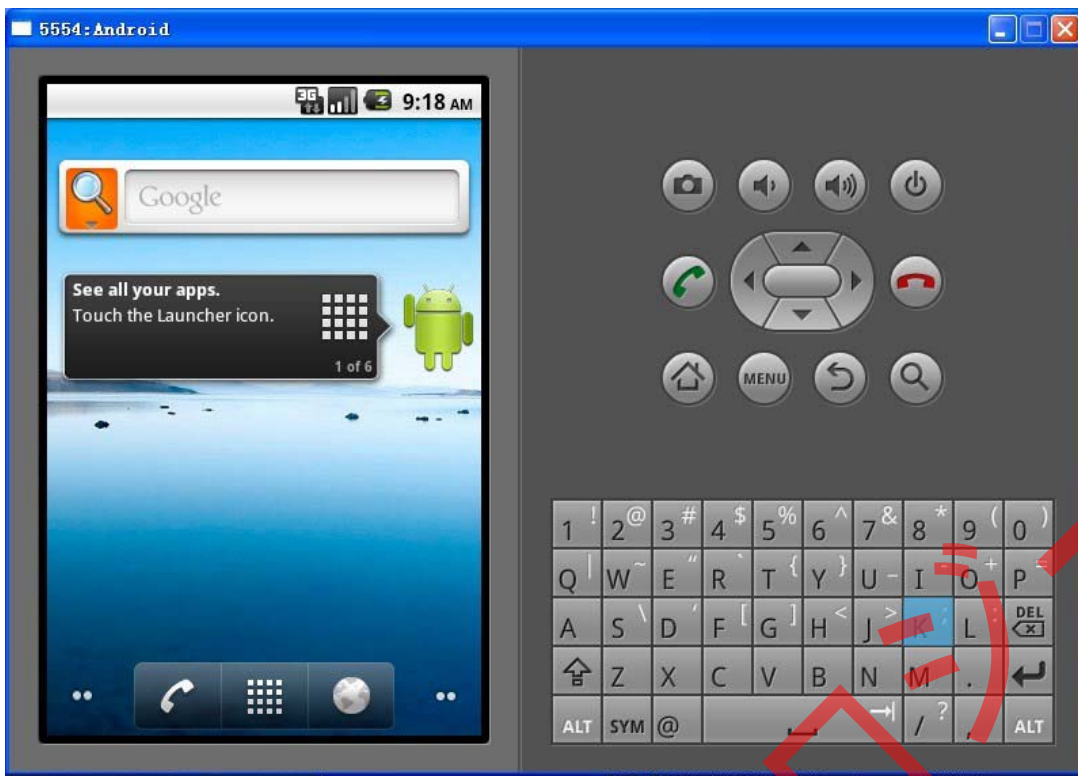
Target (開発に使用されたバージョン)

TF Card(アナログメモリのサイズ、任意)

Skin (スクリーンサイズのパラメータ)

上述のパラメータを書き込んで、Create AVD をクリックし、設定終了。

4) 2.2バージョンのシミュレータを作成した。このシミュレータを選択し、右側の start ボタンをクリックする。正常に設定すれば、シミュレータのインターフェースが現れる。(少なくとも2-5分がかかる。)



以上。