REV-SA01

SMARC Evaluation Carrier Board

User's Manual

3rd Ed – 24 January 2016

Part No. E2047VS0102R

Content

1. Getting Started

1.1 Safety Precautions

Warning!

Always completely disconnect the power cord from your chassis whenever you work with the hardware. Do not make connections while the power is on. Sensitive electronic components can be damaged by sudden power surges. Only experienced electronics personnel should open the PC chassis.

Caution!

Always ground yourself to remove any static charge before touching the CPU card. Modern electronic devices are very sensitive to static electric charges. As a safety precaution, use a grounding wrist strap at all times. Place all electronic components in a static-dissipative surface or static-shielded bag when they are not in the chassis.

Always note that improper disassembling action could cause damage to the motherboard. We suggest not removing the heatsink without correct instructions in any circumstance. If you really have to do this, please contact us for further support.

1.2 Packing List

Before you begin installing your single board, please make sure that the following materials have been shipped:

- 1 x REV-SA01 Micro Module

If any of the above items is damaged or missing, contact your retailer.

1.3 Document Amendment History

Revision	Date	By	Comment
1st	December 2013		Initial Release
2nd	October 2014		Update Linux User Guide
3rd	January 2017		Update System Specifications

1.4 Manual Objectives

This manual describes in details of the REV-SA01 Single Board.

We have tried to include as much information as possible but we have not duplicated information that is provided in the standard IBM Technical References, unless it proved to be necessary to aid in the understanding of this board.

We strongly recommend that you study this manual carefully before attempting to set up REV-SA01 series or change the standard configurations. Whilst all the necessary information is available in this manual we would recommend that unless you are confident, you contact your supplier for guidance.

Please be aware that it is possible to create configurations within the CMOS RAM that make booting impossible. If this should happen, clear the CMOS settings, (see the description of the Jumper Settings for details).

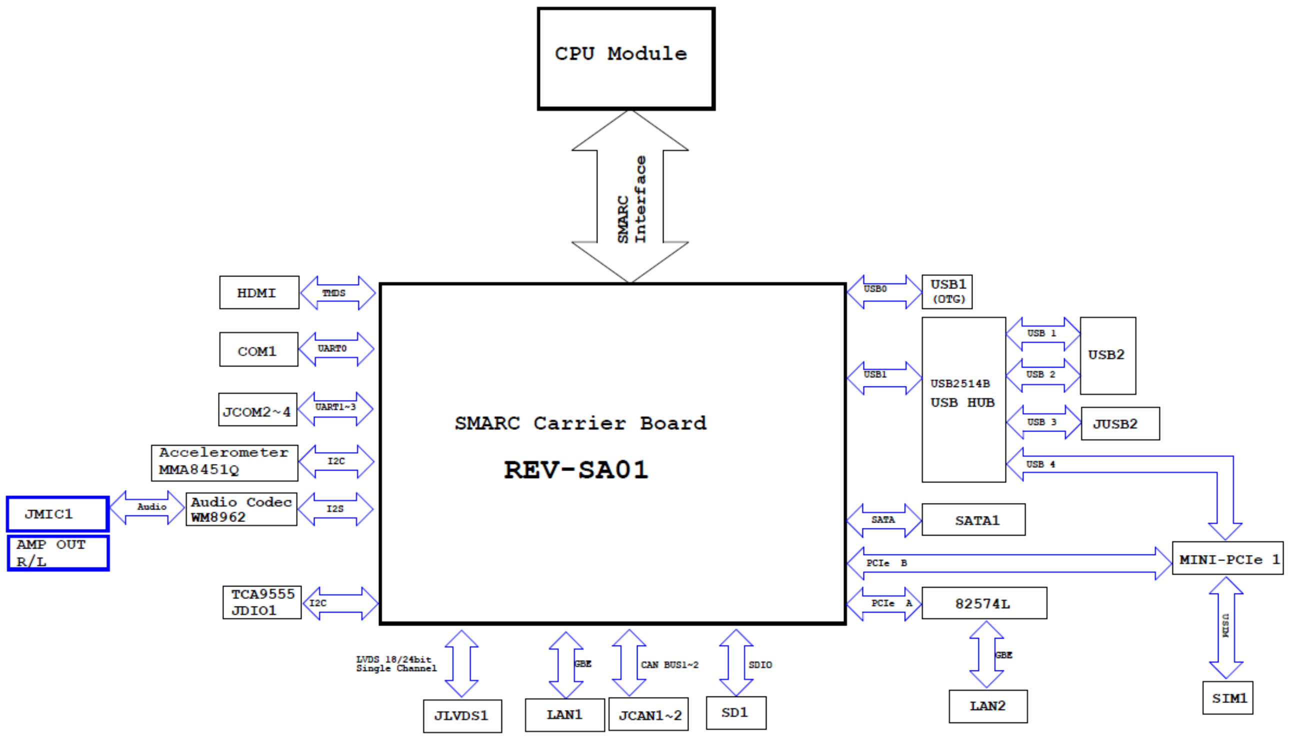
If you have any suggestions or find any errors regarding this manual and want to inform us of these, please contact our Customer Service department with the relevant details.

1.5 System Specifications

System	SMARC CPU Module socket: Accepts 82mm x 50mm SMARC Modules
Edge conn	DB9 x 1 DB15 x 1 HDMI x 1 Mini-USB x 1 USB Type A double deck x 1 RJ45 x 2 SD Socket x 1
Backlight conn	5V, GND, ENBKL, VR, PWM
LVDS connector	Hirose DF13-40DS-1.25, support 1 x 18/24bit LVDS
HDMI	HDMI connector
VGA	By Chrontel CH7055 (convert 24bit TTL signal to VGA)
CAN BUS	CAN BUS connector x 2 (JST PHR-4) (CAN PHY TJA1041)
GPIO	10bit GPIO(pin header)
Audio connector	Speaker out (L & R)
USB	USB Type A double deck x 1 USB signal for mPCIe (x1) USB 2.0 pin header (x1) Mini USB connector for USB OTG
SATA	SATA connector x 1, 2pin wafer with 5V, 1A for SATA power
CAN Bus	CAN bus pin header x 2
PCIe	mPCIe socket x 1 (with PCIe x 1 & USB2.0 & USIM signal)
Ethernet	RJ45 connector for GBE (with LED)
2nd Ethernet	INTEL 82574L
Audio Codec	WM8962, Line out, MIC in, Speaker out
RTC battery	CR2032, RTC chip is ISL1208
Boot select	8 pin 2.0mm jumper
SD Socket	SD Socket, support SDHC
LED	Power on LED

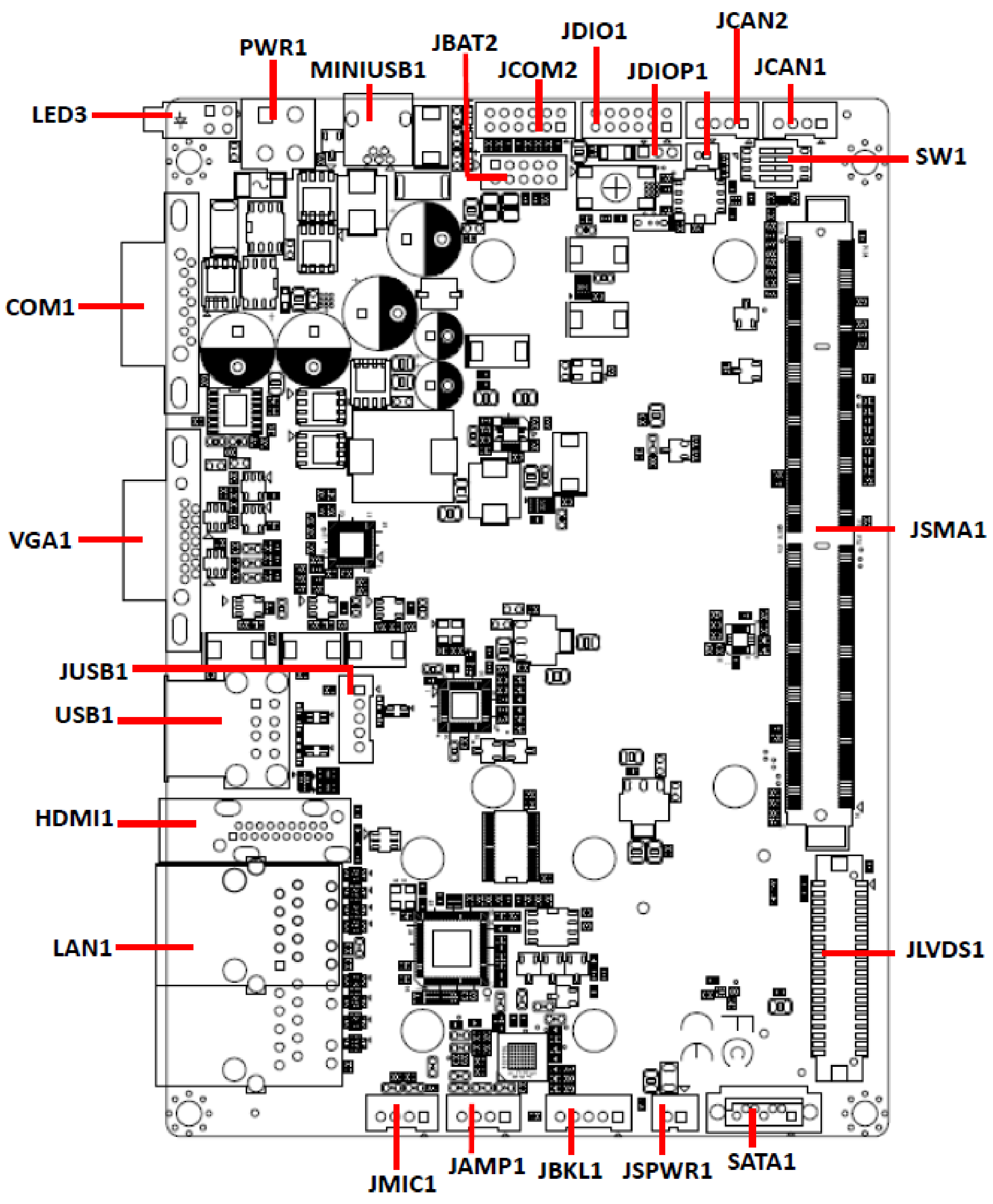
1.6 Architecture Overview—Block Diagram

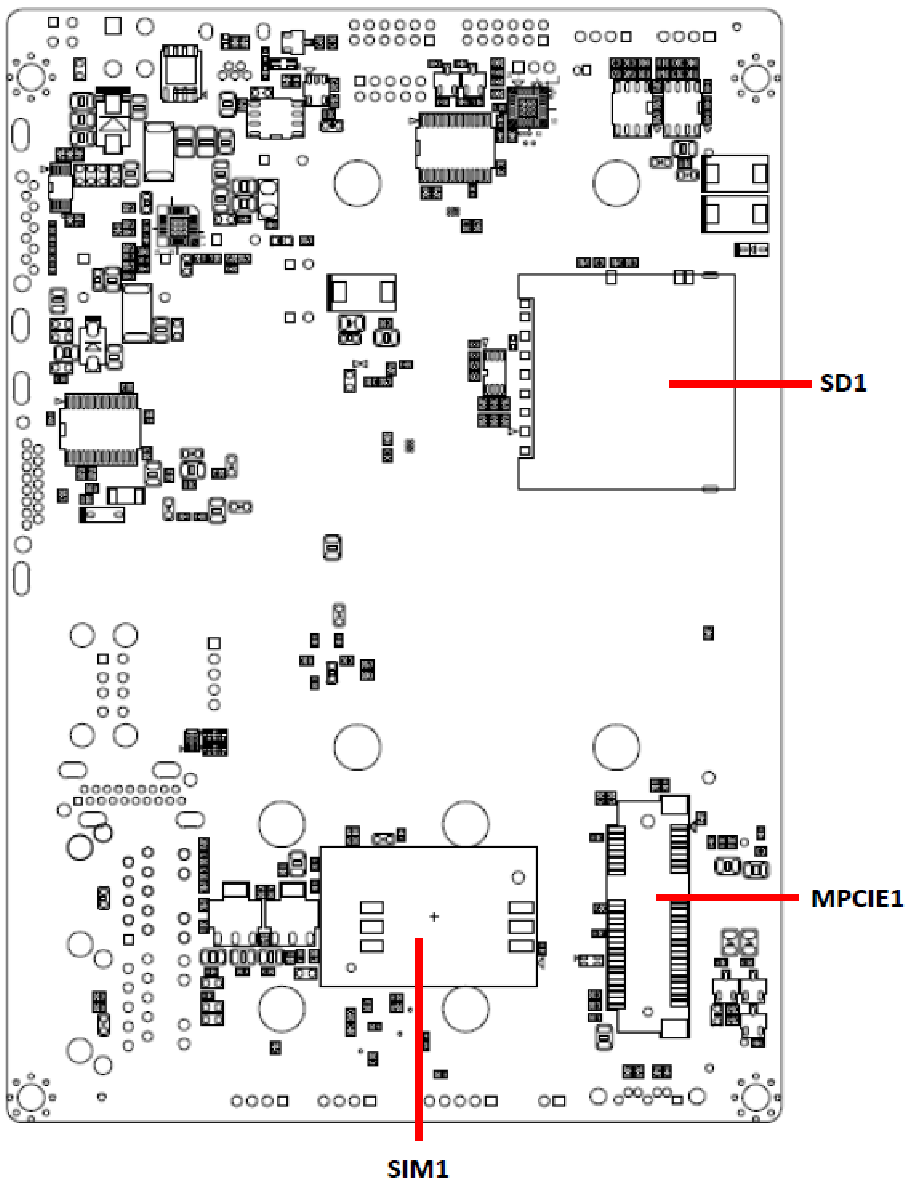
The following block diagram shows the architecture and main components of REV-SA01.



2. Hardware Configuration

2.1 Product Overview

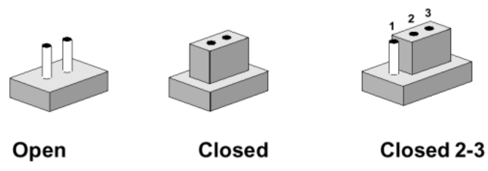



SD1
MPCIE1
SIM1

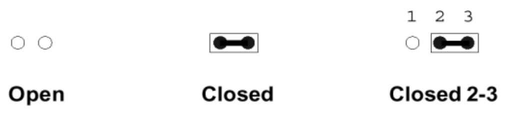
2.2 Jumper and Connector List

You can configure your board to match the needs of your application by setting jumpers. A jumper is the simplest kind of electric switch.

It consists of two metal pins and a small metal clip (often protected by a plastic cover) that slides over the pins to connect them. To "close" a jumper you connect the pins with the clip. To "open" a jumper you remove the clip. Sometimes a jumper will have three pins, labeled 1, 2, and 3. In this case, you would connect either two pins.



The jumper settings are schematically depicted in this manual as follows:



A pair of needle-nose pliers may be helpful when working with jumpers.

Connectors on the board are linked to external devices such as hard disk drives, a keyboard, or floppy drives. In addition, the board has a number of jumpers that allow you to configure your system to suit your application.

If you have any doubts about the best hardware configuration for your application, contact your local distributor or sales representative before you make any changes.

The following tables list the function of each of the board's jumpers and connectors.

Jumpers

Label	Function	Note
SW1	Boot Mode selector	DIP Switch 4P
JDIOP1	General purpose I/O Power selector	3 x 1 header, pitch 2.00mm

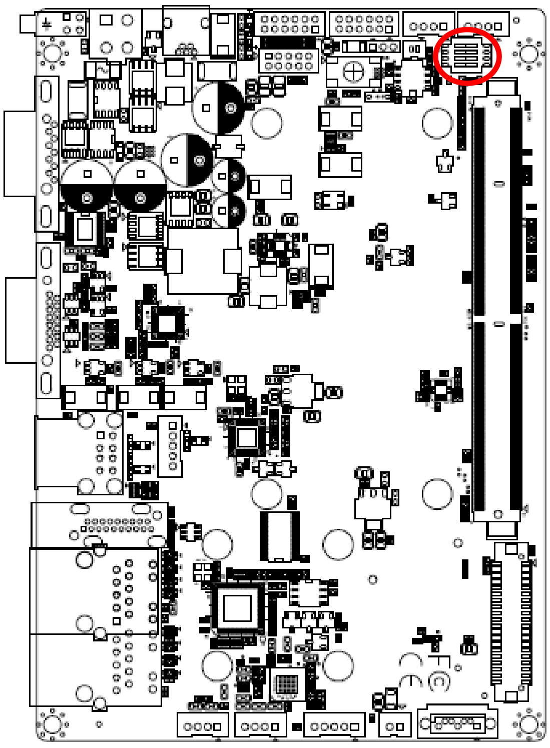
Connectors

Label	Function	Note
JBAT1	Battery connector 1	2 x 1 wafer, pitch 1.25mm
JBAT2	Battery connector 2	2 x 5 wafer, pitch 2.00mm
JDIO1	General purpose I/O connector	2 x 6 wafer, pitch 2.00mm
JCAN1/2	Can Bus connector 1/2	4 x 1 wafer, pitch 2.00mm
JSMA1	Smart Mobility ARChitecture slot	
SATA1	Serial ATA connector 1	
JBKL1	LCD inverter connector	5 x 1 wafer, pitch 2.00mm

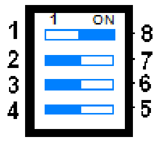
JAMP1	AMPLIFIER connector	4 x 1 wafer, pitch 2.00mm
JMIC1	Line In, MIC connector	4 x 1 wafer, pitch 2.00mm
LAN1	RJ-45 Ethernet connector	
VGA1	VGA connector	
LED3	LED connector	
MINIUSB1	Mini USB connector for Boot/Debug	MINI USB-MAB_5P
COM1	Serial Port 1 connector	
JCOM2	Serial Port 2 connector	2 x 6 wafer, pitch 2.00mm
HDMI1	HDMI connector	HDMI_19P
JLVDS1	LVDS Interface connector	20 x 2 wafer, pitch 1.25mm
PWR1	Power connector	2 x 2 wafer, pitch 4.20mm
USB1	USB connector	
JUSB1	USB connector	5 x 1 wafer, pitch 2.00mm
JSPWR1	SATA Power connector	2 x 1 wafer, pitch 2.00mm
MPCIE1	Mini-PCI connector	
SD1	SD Card Slot	
SIM1	SIM Card Slot	SDCARD_9H, Push/Push Type

2.3 Setting Jumpers & Connectors

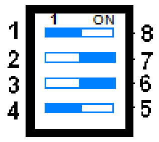
2.3.1 Boot Mode selector (SW1)



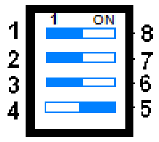
Booting from onboard eMMC



Boot from SD card

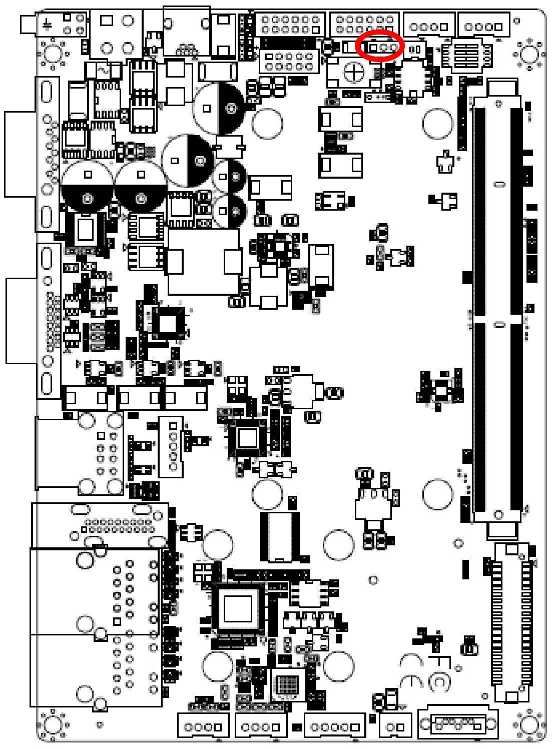


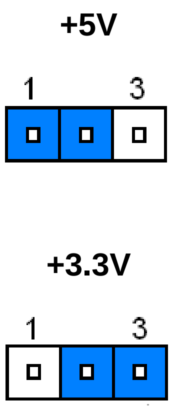
MFG tool mode mode(burning image file to onboard eMMC)



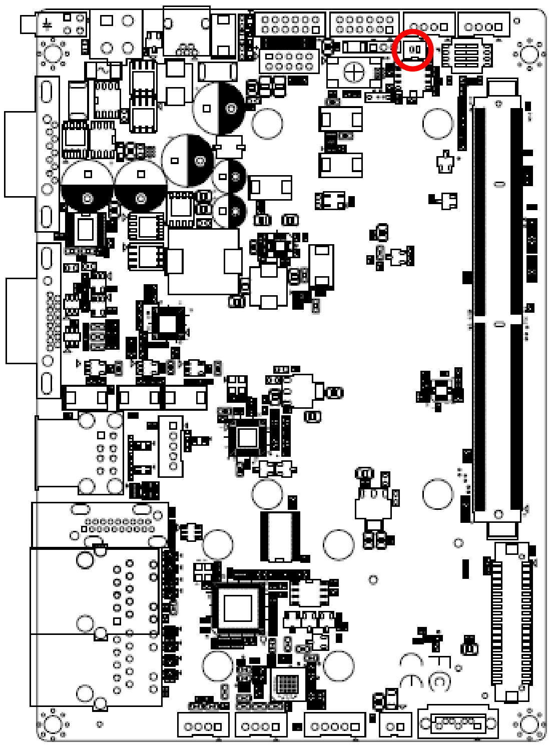
Signal	PIN	PIN	Signal
GND	1	8	BOOT_SEL0#
GND	2	7	BOOT_SEL1#
GND	3	6	BOOT_SEL2#
GND	4	5	FORCE_RECOV#

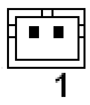
2.3.2 General purpose I/O Power selector (JDIOP1)





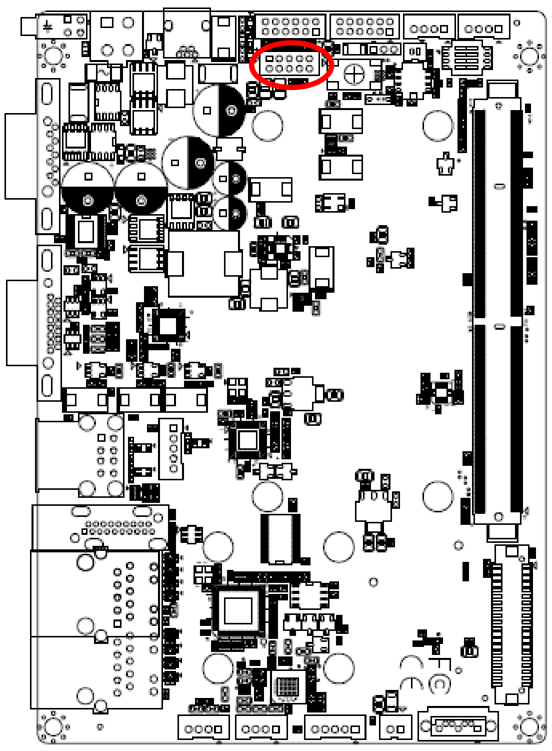
2.3.3 Battery connector 1 (JBAT1)

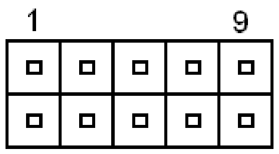




Signal	PIN
+V_BAT	1
GND	2

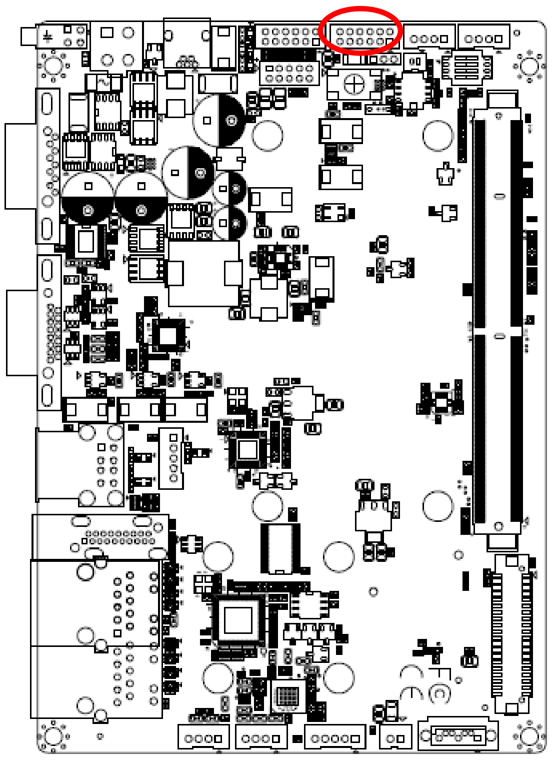
2.3.4 Battery connector 2 (JBAT2)

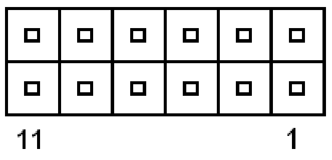




Signal	PIN	PIN	Signal
+V_BATTERY	1	2	+V_BATTERY
+V_BATTERY	3	4	GND
GND	5	6	BAT_DATA
BAT_TS	7	8	BAT_CLK
GND	9	10	GND

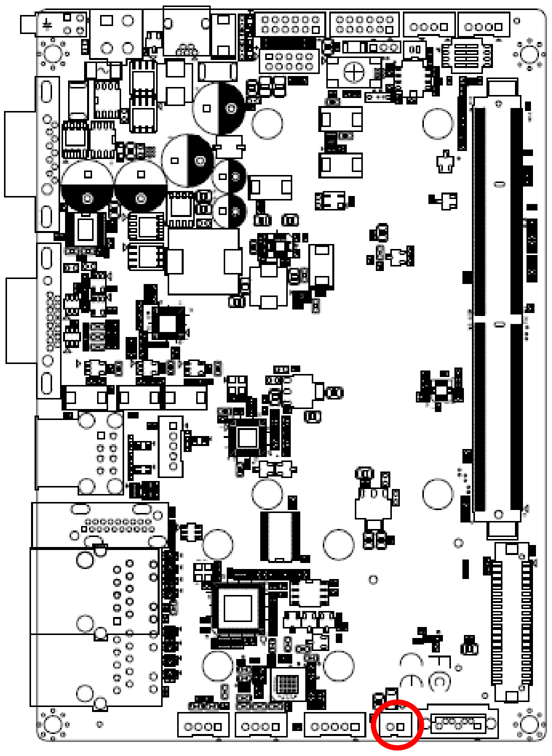
2.3.5 General purpose I/O connector (JDIO1)

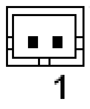




Signal	PIN	PIN	Signal
DIO_GP10	1	2	DIO_GP20
DIO_GP11	3	4	DIO_GP21
DIO_GP12	5	6	DIO_GP22
DIO_GP13	7	8	DIO_GP23
SMB_DATA_9555	9	10	SMB_CLK_9555
+VDIO	11	12	GND

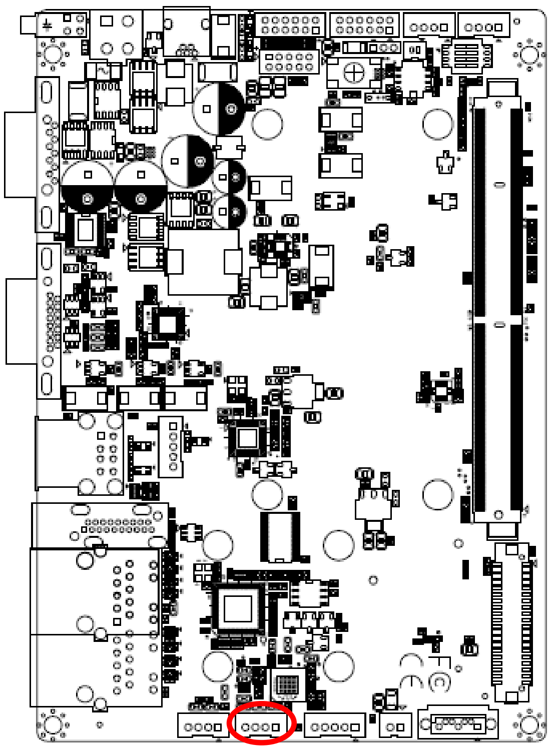
2.3.6 SATA Power connector (JSPWR1)

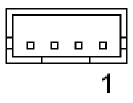




Signal	PIN
GND	1
+5V	2

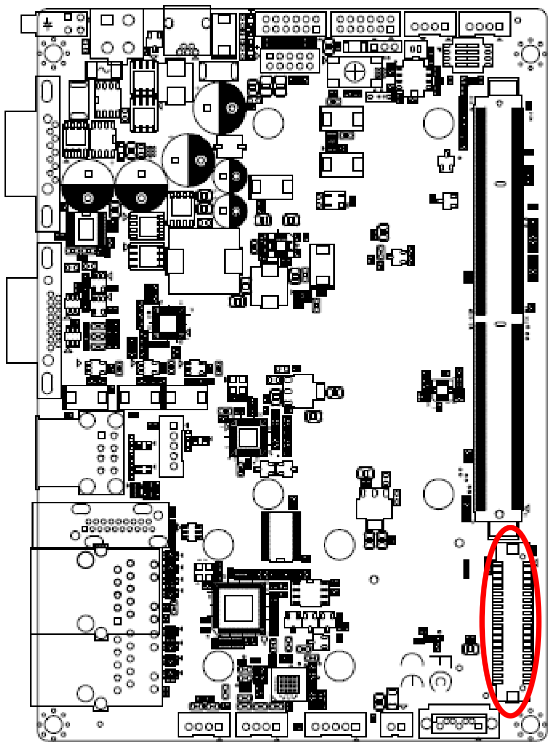
2.3.7 AMPLIFIER connector (JAMP1)

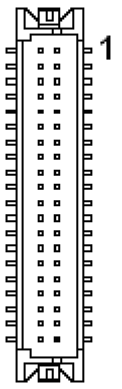




Signal	PIN
SPKL_P	1
SPKL_N	2
SPKR_P	3
SPKR_N	4

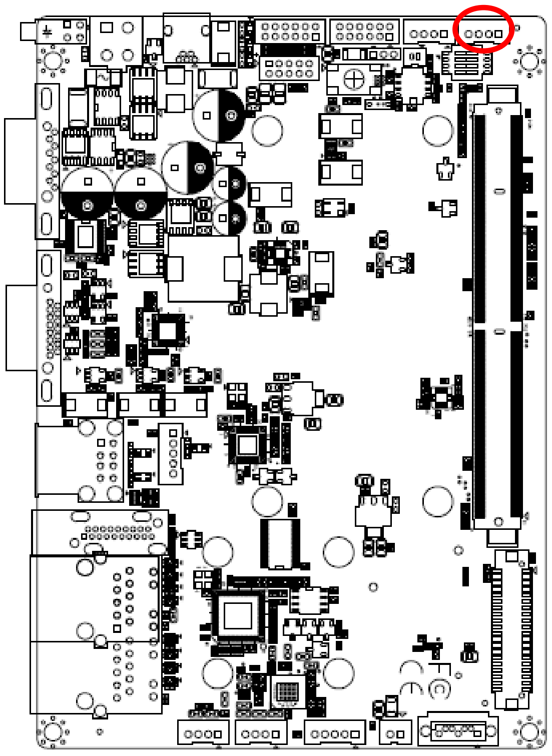
2.3.8 LVDS Interface connector (JLVDS1)

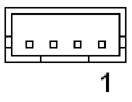




Signal	PIN	PIN	Signal
+5V	2	1	+3.3V
+5V	4	3	+3.3V
I2C_LCD_DAT	6	5	I2C_LCD_CK
GND	8	7	GND
LVDS0+	10	9	LVDS1+
LVDS0-	12	11	LVDS1-
GND	14	13	GND
LVDS2+	16	15	LVDS3+
LVDS2-	18	17	LVDS3-
GND	20	19	GND
NC	22	21	NC
NC	24	23	NC
GND	26	25	GND
NC	28	27	NC
NC	30	29	NC
GND	32	31	GND
LVDS_CK+	34	33	NC
LVDS_CK-	36	35	NC
GND	38	37	GND
NC	40	39	NC

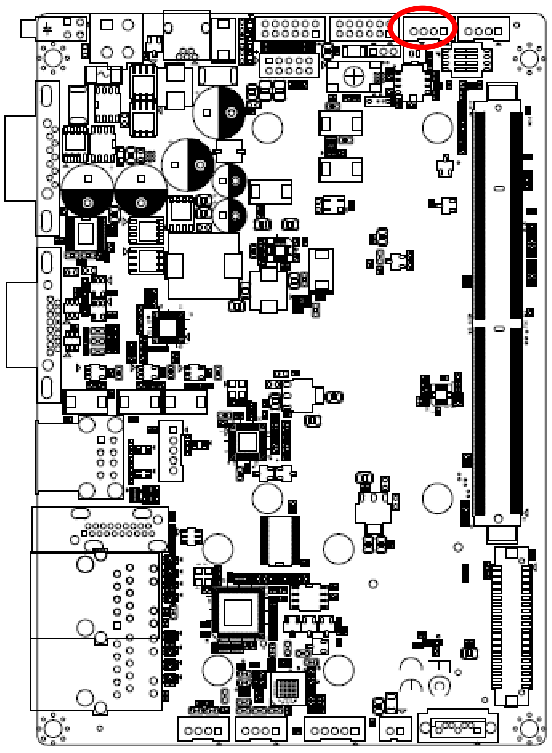
2.3.9 Can Bus connector 1 (JCAN1)

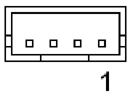




Signal	PIN
CAN0_H	1
GND	2
CAN0_L	3
GND	4

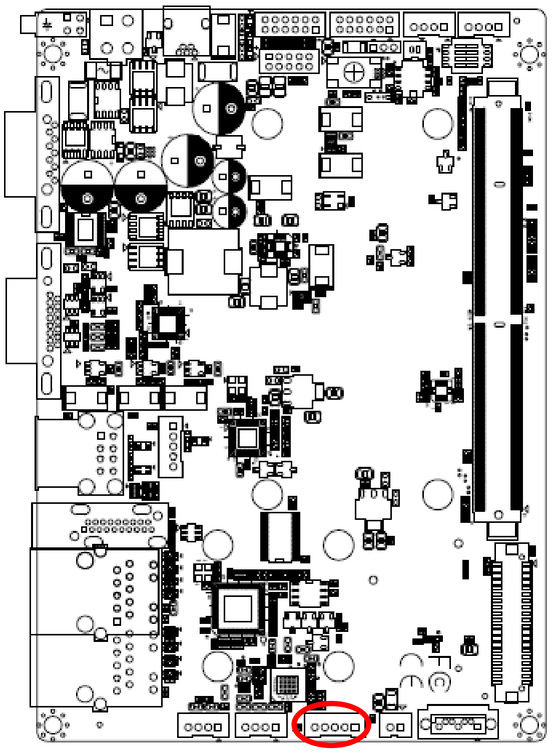
2.3.10 Can Bus connector 2 (JCAN2)

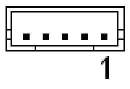




Signal	PIN
CAN1_H	1
GND	2
CAN1_L	3
GND	4

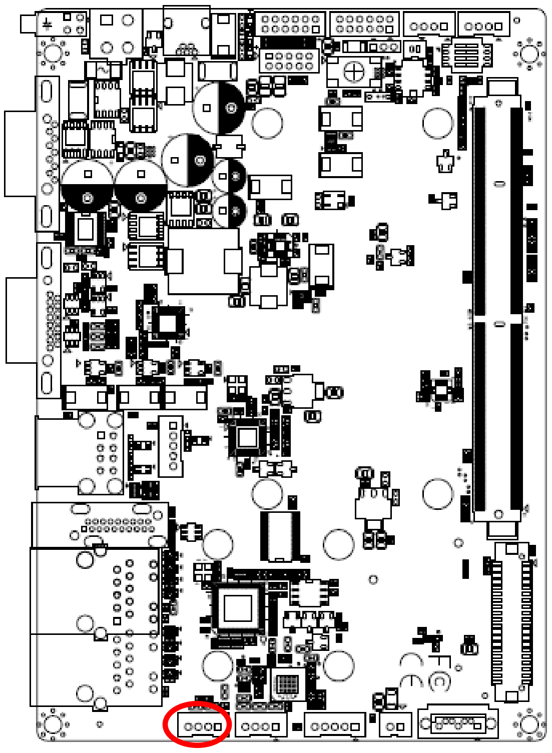
2.3.11 LCD inverter connector (JBKL1)

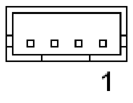




Signal	PIN
NC	1
GND	2
LCD_BKLT_EN	3
LCD_BKLT_PWM	4
+5V	5

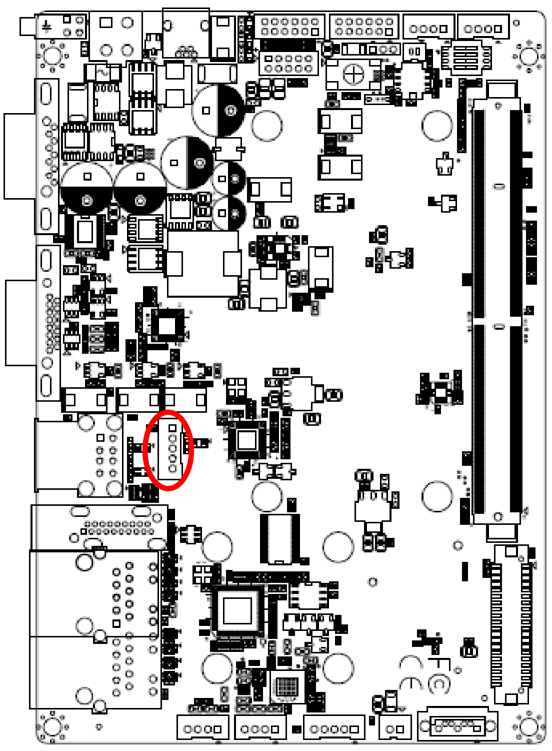
2.3.12 Line In, MIC connector (JMIC1)

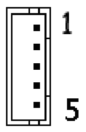




Signal	PIN
MICIN_DET	1
MIC_RAW	2
MICBIAS2_RAW	3
GND	4

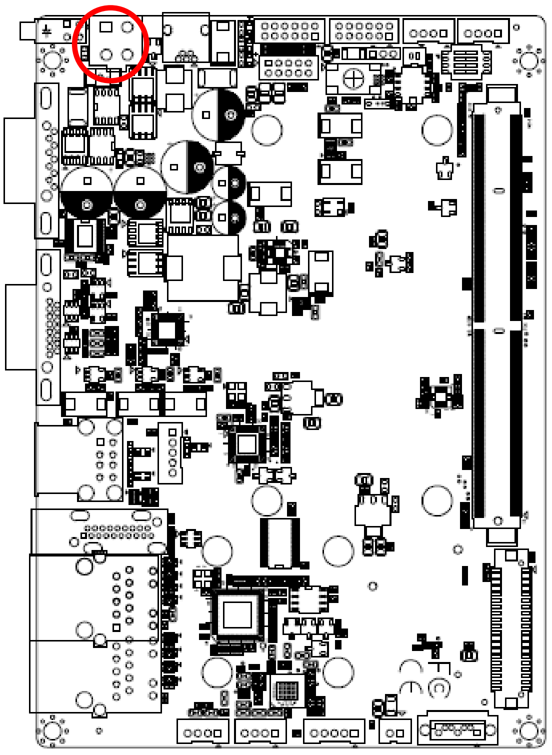
2.3.13 USB connector (JUSB1)

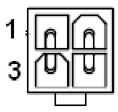




Signal	PIN
+4V	1
USB3_D-	2
USB3_D+	3
GND	4
GND	5

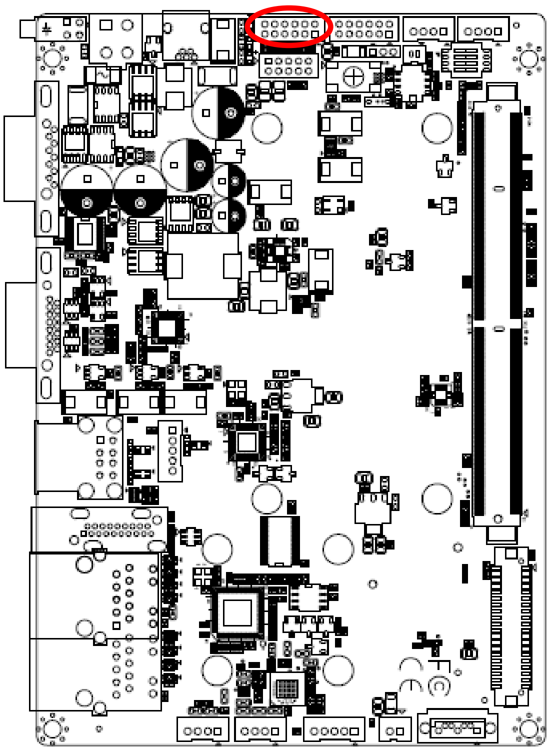
2.3.14 Power connector (PWR1)

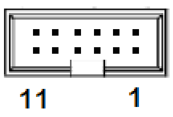




Signal	PIN	PIN	Signal
RVSP_G	1	2	RVSP_G
+VIN	3	4	+VIN

2.3.15 Serial Port 2 connector (JCOM2)





Signal	PIN	PIN	Signal
RXDD3	1	2	RXDD2
TXDD3	3	4	TXDD2
GND	5	6	GND
CTS3	7	8	RXDD4
RTS3	9	10	TXDD4
GND	11	12	GND

3. Linux User Guide

3.1 Download Source code for building Ubuntu image file

Please make a folder for storing the source code first then typing the command below to get started for the source code download.

$ Sudo apt-get install git
$ git clone guest@202.55.227.57:freescale/core.git -b SMARC
About password, please check with our support team.

3.2 Set up a Linux host for building U-boot & Kernel Image

We support to compile u-boot & Kernel on Ubuntu 12.04 (64bit version), other version of Ubuntu is not currently supported and may have built issues.

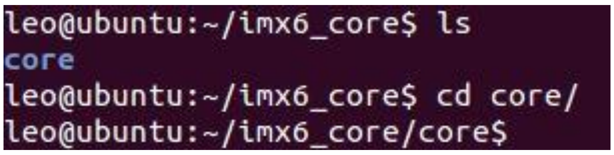
Install host packages needed by building code. This document assumes you are using Ubuntu. Not a requirement, but the packages may be named differently and the method of installing them may be different.

```
sudo apt-get install ia32-libs
sudo apt-get install uboot-mkimage
```

3.3 Building up U-boot & Kernel image

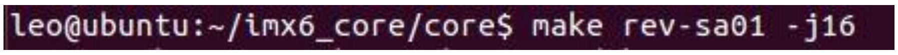
You can follow up the steps below to compile the u-boot & Kernel after downloading the source code.
1.Please move to folder "core" then start to compile both the u-boot & Kernel.

▪▪~/$ cd core/



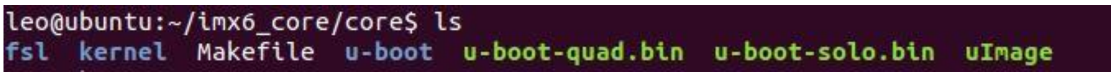
```
leo@ubuntu:~/imx6_core$ ls
core
leo@ubuntu:~/imx6_core$ cd core/
leo@ubuntu:~/imx6_core/core$
```

2.Type the command to compile both u-boot & Kernel.
$ make rev-sa01 –j number
(**-j number** means multi jobs for more efficiant building, you can add it according to your CPU performance of PC, e.g. mine is "–j16" as below).



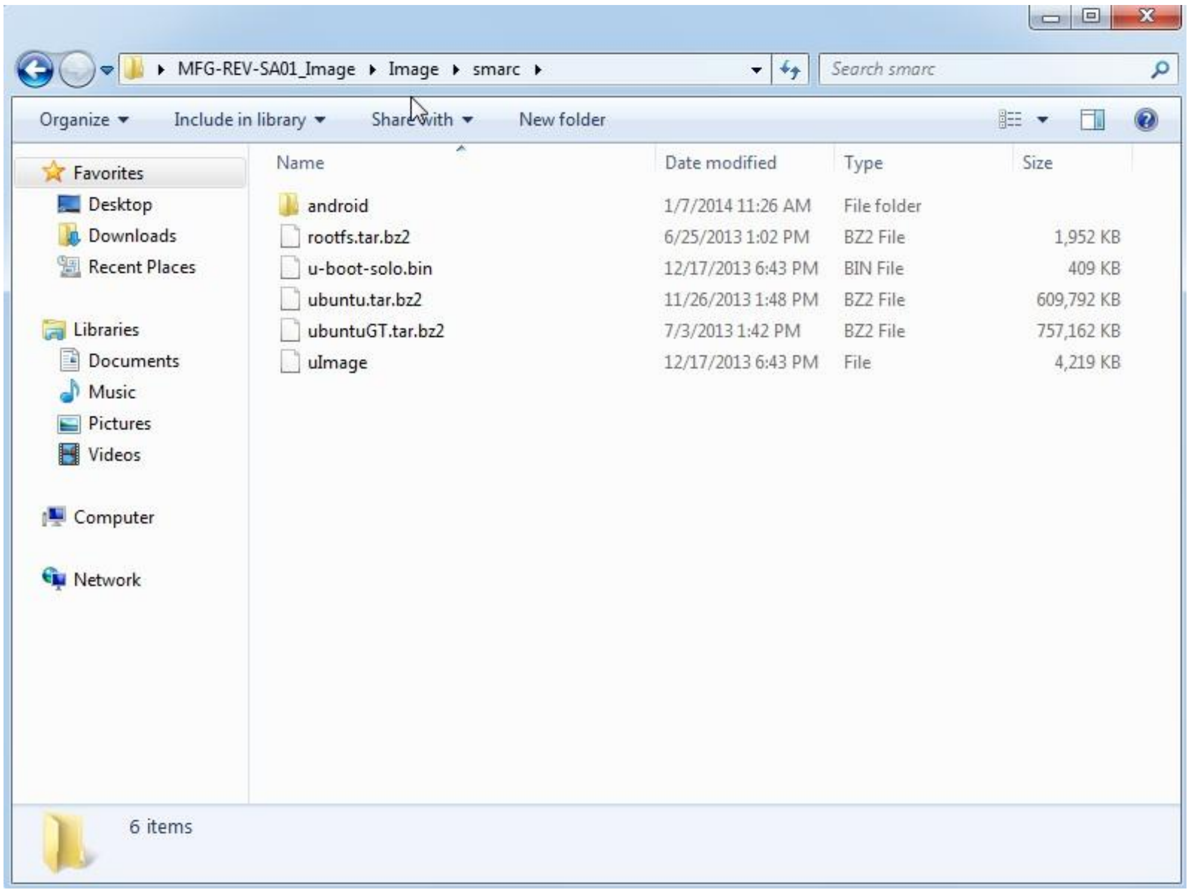
```
leo@ubuntu:~/imx6_core/core$ make rev-sa01 -j16
```

3. You can find the u-boot(u-boot.bin) & Kernel(uImage) under folder "core" as below after the compiling is finish.



```
leo@ubuntu:~/imx6_core/core$ ls
fsl  kernel  Makefile  u-boot  u-boot-quad.bin  u-boot-solo.bin  uImage
```

PS: If you would like to use **Mfgtool** for flashing image file, you must put the file **u-boot.bin** and **uImage** under "~\Image\smarc" for right detected path.

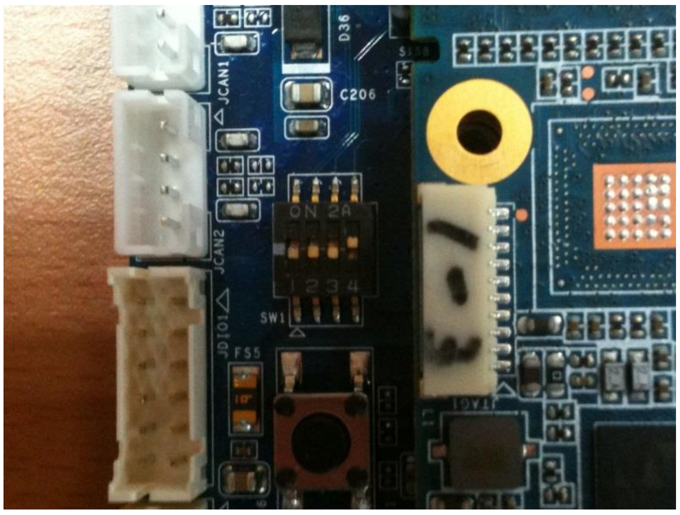


```
Name              Date modified       Type         Size
android           1/7/2014 11:26 AM   File folder
rootfs.tar.bz2    6/25/2013 1:02 PM   BZ2 File       1,952 KB
u-boot-solo.bin   12/17/2013 6:43 PM  BIN File         409 KB
ubuntu.tar.bz2    11/26/2013 1:48 PM  BZ2 File     609,792 KB
ubuntuGT.tar.bz2  7/3/2013 1:42 PM    BZ2 File     757,162 KB
uImage            12/17/2013 6:43 PM  File           4,219 KB

6 items
```

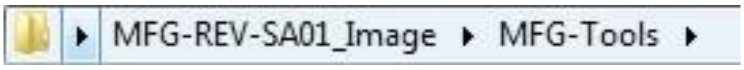
3.4 Use MfgTool to flash Ubuntu into onboard eMMC

Manufacturing tool, a successor of ATK, provides a series of new features to power your mass production work. The features like windows style GUI, multiple devices support, explicit status monitoring, versatile functionalities and highly flexible architecture make it a best choice to meet your critical timing, cost and customization requirements.
For using Mfgtool to flash image file into onboard eMMC, please follow up the steps below

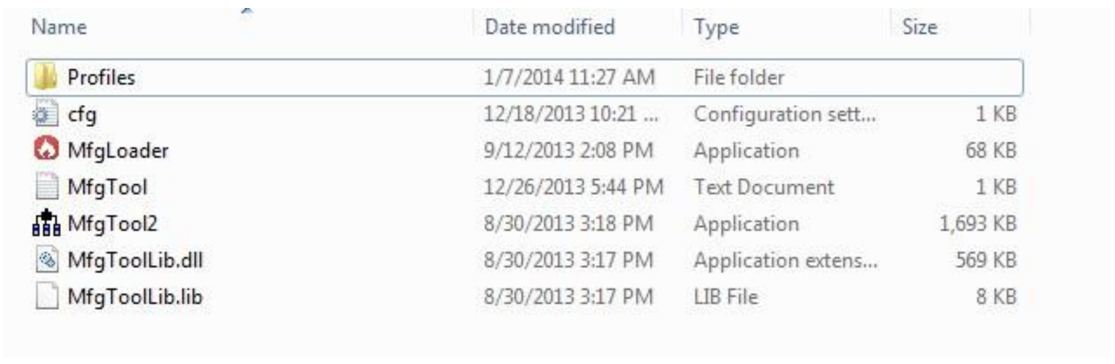
1. Please turn on the **Pin4** of the DIP switch as below into burning mode of Mfgtool.



2. Power on the mainboard then plug the cable from OTG socket to PC.
3. Click the folder "**~\MFG-Tools**", e.g. mine is D:\ MFG-REV-SA01_Image\MFG-Tools"

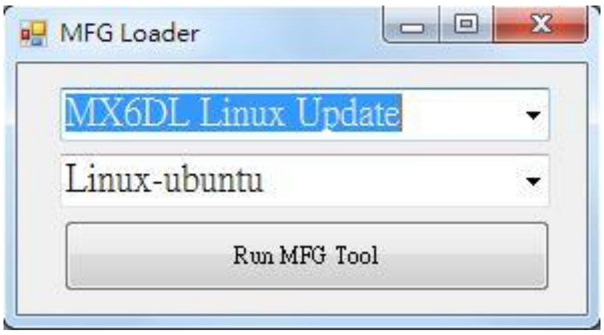


4. Click the "**MfgLoader.exe**".

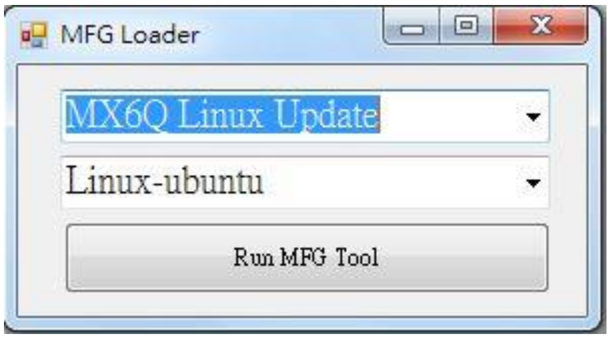


Name	Date modified	Type	Size
Profiles	1/7/2014 11:27 AM	File folder	
cfg	12/18/2013 10:21 ...	Configuration sett...	1 KB
MfgLoader	9/12/2013 2:08 PM	Application	68 KB
MfgTool	12/26/2013 5:44 PM	Text Document	1 KB
MfgTool2	8/30/2013 3:18 PM	Application	1,693 KB
MfgToolLib.dll	8/30/2013 3:17 PM	Application extens...	569 KB
MfgToolLib.lib	8/30/2013 3:17 PM	LIB File	8 KB

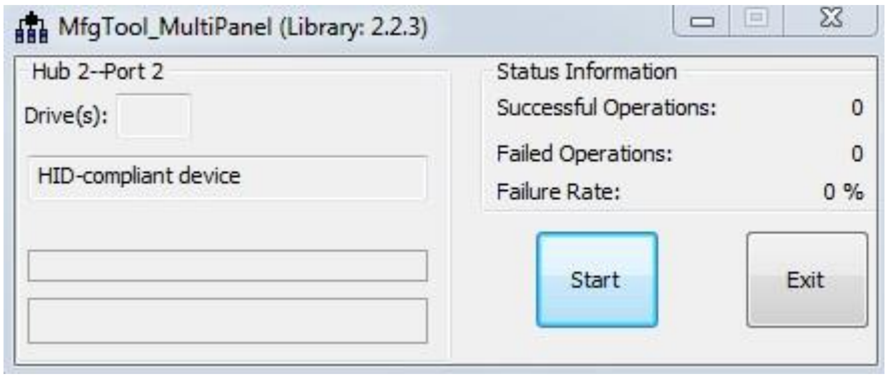
5. Select the MCU option by name, if the MCU of module board is "i.MX6 Solo", please click "MX6DL Linux Update", and click "Linux-ubuntu" (Ubuntu GUI version) for the OS of flashing, then click "Run MFG Tool".



Or the if the MCU of module board is "i.MX6 Quad core", please click "**MX6Q Linux Update**", and click "**Linux-ubuntu**" (Ubuntu GUI version) for the OS of flashing, then click "**Run MFG Tool**".



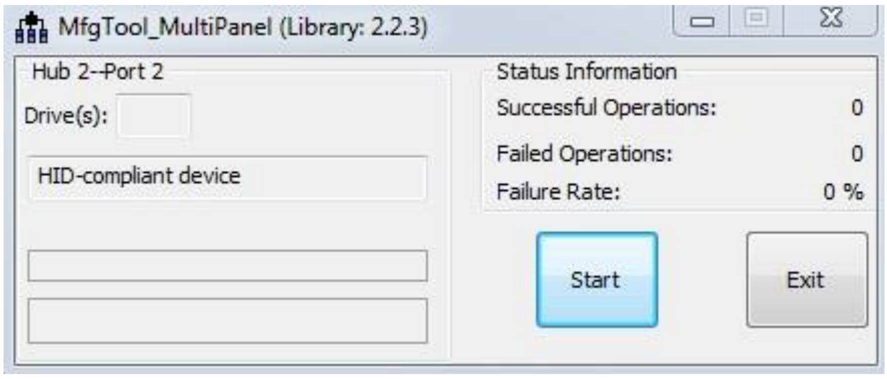
6. The second screen will show up after clicking "Run MFG Tool", and please check whether it shows "HID-compliant device" as below, if not, please re-check the cable connection and DIP switch setting between mainboard and PC.


MfgTool_MultiPanel (Library: 2.2.3)

Hub 2--Port 2
Drive(s):
HID-compliant device

Status Information
Successful Operations: 0
Failed Operations: 0
Failure Rate: 0 %

7. Click "**Start**" to flash image file.



Hub 2--Port 2

Drive(s):

HID-compliant device

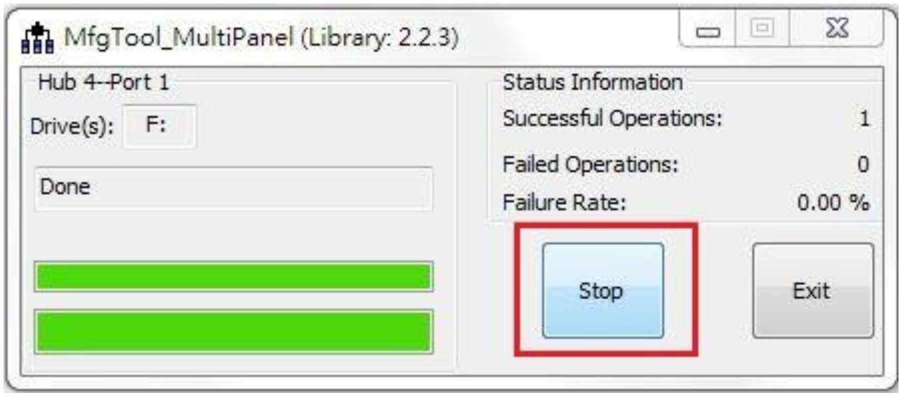
Status Information

Successful Operations: 0

Failed Operations: 0

Failure Rate: 0 %

8. It will show "Done" after flashing is finish, then click "Stop" and "Exit" to close the screen.



Hub 4--Port 1

Drive(s): F:

Done

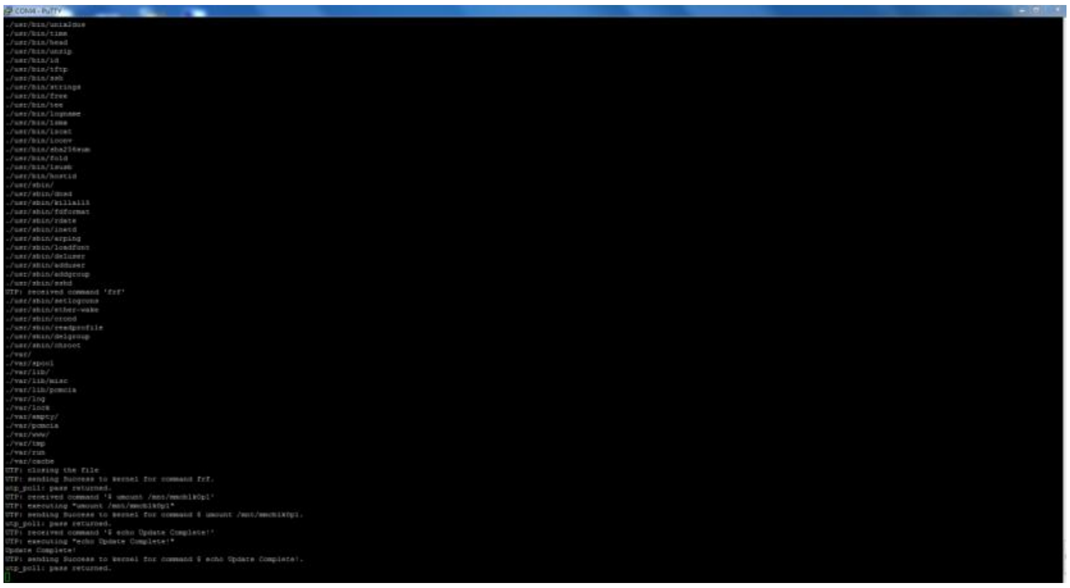
Status Information

Successful Operations: 1

Failed Operations: 0

Failure Rate: 0.00 %

9. You can also get the information from Terminal (debug port→COM1) after flashing is finish.

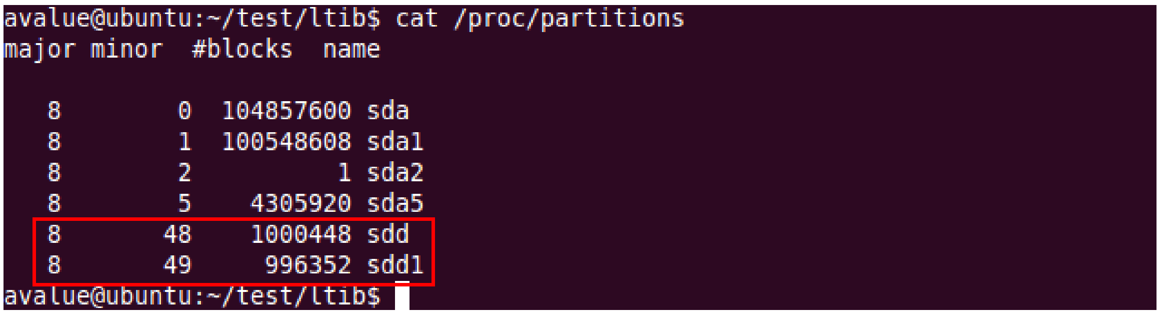


```
./usr/bin/unix2dos
./usr/bin/time
./usr/bin/head
./usr/bin/unzip
./usr/bin/id
./usr/bin/tftp
./usr/bin/ssh
./usr/bin/strings
./usr/bin/free
./usr/bin/tee
./usr/bin/logname
./usr/bin/lzma
./usr/bin/lzcat
./usr/bin/iconv
./usr/bin/sha256sum
./usr/bin/fold
./usr/bin/lsusb
./usr/bin/hostid
./usr/sbin/
./usr/sbin/dnsd
./usr/sbin/killall5
./usr/sbin/fdformat
./usr/sbin/rdate
./usr/sbin/inetd
./usr/sbin/arping
./usr/sbin/loadfont
./usr/sbin/deluser
./usr/sbin/adduser
./usr/sbin/addgroup
./usr/sbin/sshd
UTP: received command 'frf'
./usr/sbin/setlogcons
./usr/sbin/ether-wake
./usr/sbin/crond
./usr/sbin/readprofile
./usr/sbin/delgroup
./usr/sbin/chroot
./var/
./var/spool
./var/lib/
./var/lib/misc
./var/lib/pcmcia
./var/log
./var/lock
./var/empty/
./var/pcmcia
./var/www/
./var/tmp
./var/run
./var/cache
UTP: closing the file
UTP: sending Success to kernel for command frf.
utp_poll: pass returned.
UTP: received command '$ umount /mnt/mmcblk0p1'
UTP: executing "umount /mnt/mmcblk0p1"
UTP: sending Success to kernel for command $ umount /mnt/mmcblk0p1.
utp_poll: pass returned.
UTP: received command '$ echo Update Complete!'
UTP: executing "echo Update Complete!"
Update Complete!
UTP: sending Success to kernel for command $ echo Update Complete!.
utp_poll: pass returned.
```

3.5 Create a bootable SD card with Ubuntu 12.04 file system

Please insert a SD card in the card reader on your Linux host PC

1) Check device node of your SD card by command below.
   $cat /proc/partitions (for example, mine is /dev/sdd as below)



```
avalue@ubuntu:~/test/ltib$ cat /proc/partitions
major minor  #blocks  name

   8        0  104857600 sda
   8        1  100548608 sda1
   8        2          1 sda2
   8        5    4305920 sda5
   8       48    1000448 sdd
   8       49     996352 sdd1
avalue@ubuntu:~/test/ltib$
```

Create EXT3 partition for SD card
$ sudo fdisk /dev/sdd
Type the following parameters (each followed by <ENTER>):

d	[delete the previous partition]
n	[create a new partition]
p	[create a primary partition]
1	[the first partition]
20480	[20480x512bytes=10MB, which leaves enough space for the kernel, the boot loader and its configuration data]
<enter>	[using the default value will create a partition that spans to the last sector of the medium]
w	[this writes the partition table to the medium and fdisk exits]

2) Format new partition in EXT3 format
   $sudo umount /dev/sdd1
   $sudo mkfs.ext3 /dev/sdd1

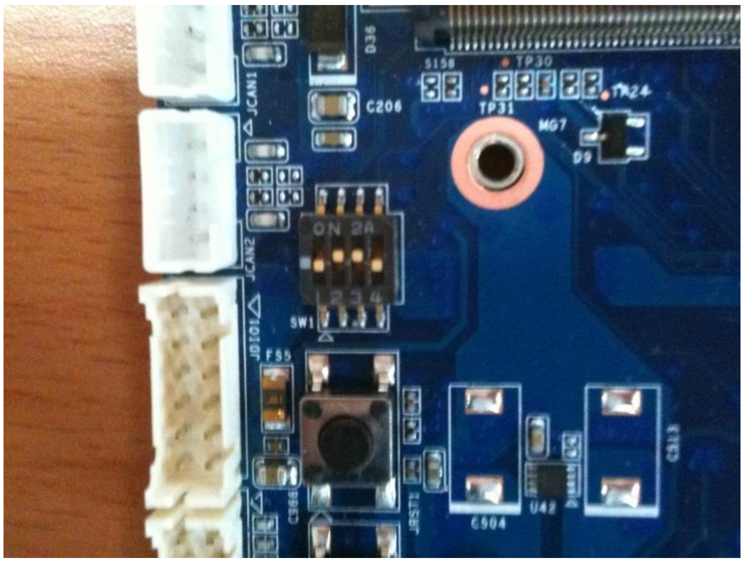
3) Install bootloader on SD card by command below.
   $ sudo dd if=u-boot-solo.bin(u-boot-quad.bin) of=/dev/sdd bs=**1k** seek=1 skip=1 conv=fsync

4) Install Linux kernel image on SD card by command below.
   $ sudo dd if=uImage of=/dev/sdd bs=**1M** seek=1 conv=fsync

5) Please find the Ubuntu file system from the path "CDROM\REV Image\MFG-REV-SA01_Image\Image\smarc\ ubuntu.tar.bz2" on User`s CD-ROM and copy it to the partition then follow up the command below.

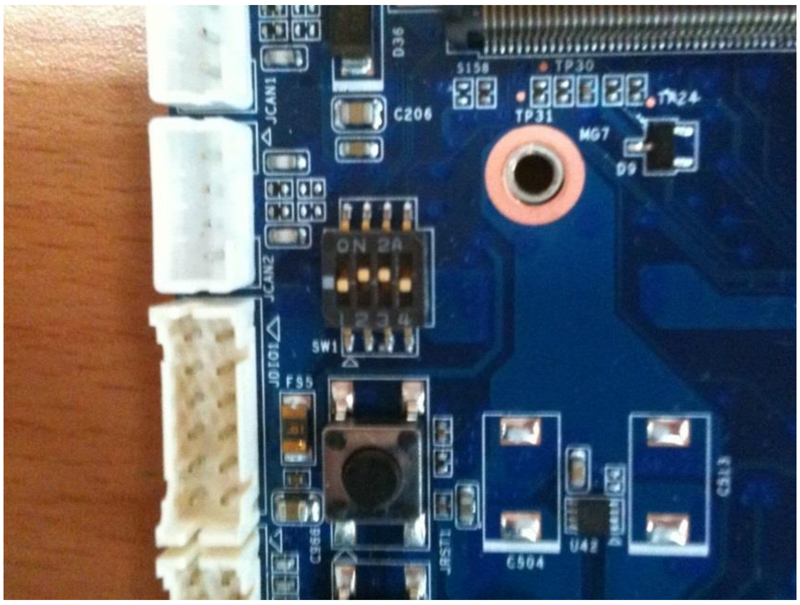
```
$ sudo umount /dev/sdd1
$ sudo mount /dev/sdd1 /mnt
$ cd /mnt
$ sudo tar jxvpf    ~/ubuntu.tar.bz2
$ cd
$ sudo umount /dev/sdd1
```

The Ubuntu file system content is now on the SD card. You can insert it to mainboard then turn on the DIP switch pin2&3 as below for booting.



3.6 Bootloader settings for booting from SD card

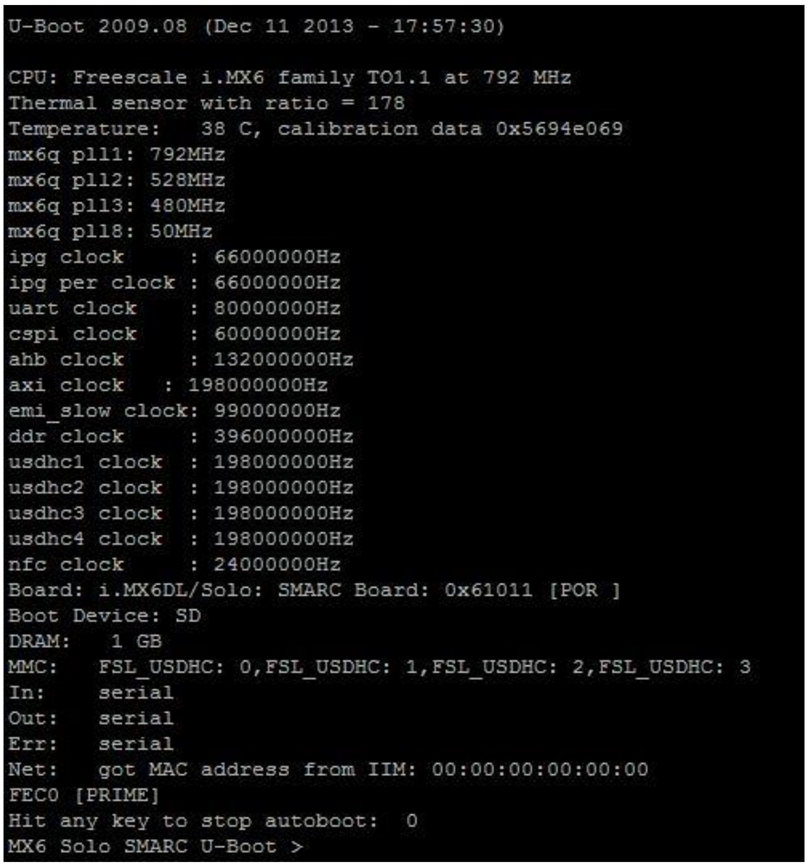
1) Please turn on the Pin 2&3 of the DIP switch as below for booting from SD card .



2) Insert SD card on SD socket. Connect RS232 cross over cable from **COM1** of mianboard to COM port of Host PC.

3) Run hyper terminal program on Host PC (teraterm on Windows or minicom on Linux)

4) Power on mainboard and press ”space” key to get into bootloader menu.



```
U-Boot 2009.08 (Dec 11 2013 - 17:57:30)

CPU: Freescale i.MX6 family TO1.1 at 792 MHz
Thermal sensor with ratio = 178
Temperature:   38 C, calibration data 0x5694e069
mx6q pll1: 792MHz
mx6q pll2: 528MHz
mx6q pll3: 480MHz
mx6q pll8: 50MHz
ipg clock     : 66000000Hz
ipg per clock : 66000000Hz
uart clock    : 80000000Hz
cspi clock    : 60000000Hz
ahb clock     : 132000000Hz
axi clock   : 198000000Hz
emi_slow clock: 99000000Hz
ddr clock     : 396000000Hz
usdhc1 clock  : 198000000Hz
usdhc2 clock  : 198000000Hz
usdhc3 clock  : 198000000Hz
usdhc4 clock  : 198000000Hz
nfc clock     : 24000000Hz
Board: i.MX6DL/Solo: SMARC Board: 0x61011 [POR ]
Boot Device: SD
DRAM:   1 GB
MMC:   FSL_USDHC: 0,FSL_USDHC: 1,FSL_USDHC: 2,FSL_USDHC: 3
In:    serial
Out:   serial
Err:   serial
Net:   got MAC address from IIM: 00:00:00:00:00:00
FEC0 [PRIME]
Hit any key to stop autoboot:  0
MX6 Solo SMARC U-Boot >
```

5) Setup boot device
SMARC U-Boot > print

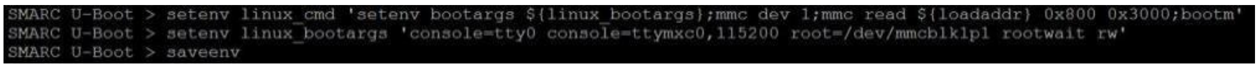
6) Set boot device as below

→ SMARC U-Boot >setenv linux_cmd ‘setenv bootargs ${linux_bootargs};mmc dev 1;mmc read ${loadaddr} 0x800 0x3000;bootm’

→ SMARC U-Boot > setenv linux_bootargs 'console=tty0 console=ttymxc0,115200 root=/dev/mmcblk1p1 rootwait rw'

→SMARC U-Boot> saveenv

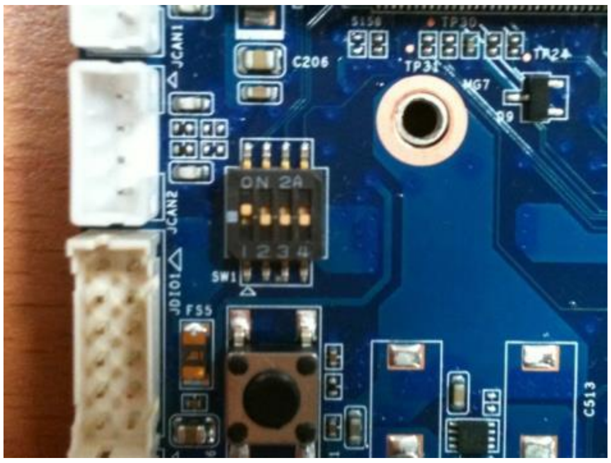
→SMARC U-Boot> boot



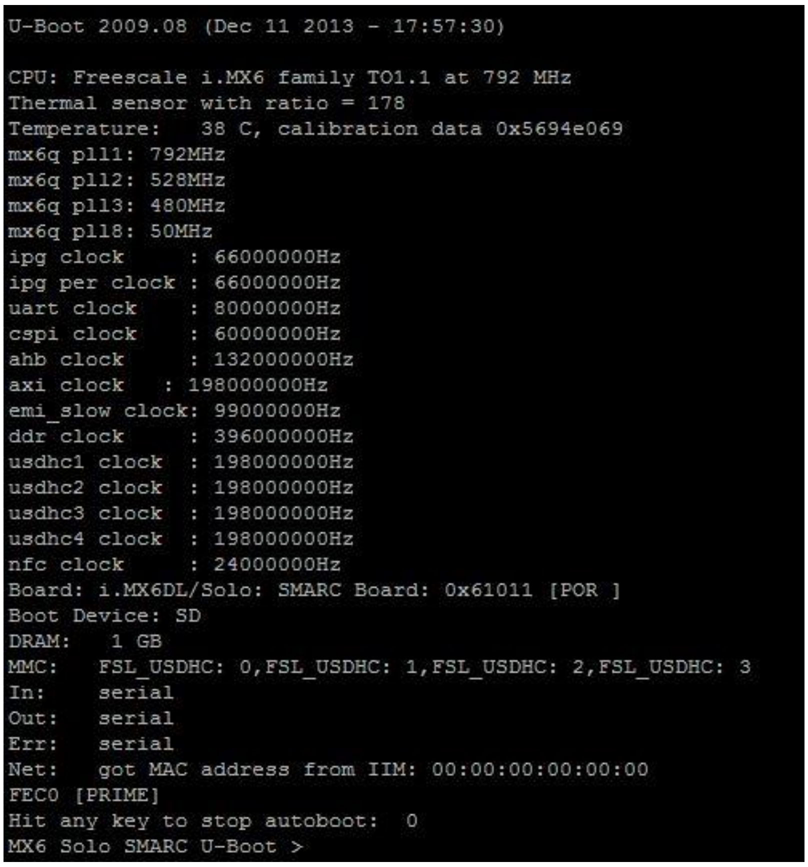
```
SMARC U-Boot > setenv linux_cmd 'setenv bootargs ${linux_bootargs};mmc dev 1;mmc read ${loadaddr} 0x800 0x3000;bootm'
SMARC U-Boot > setenv linux_bootargs 'console=tty0 console=ttymxc0,115200 root=/dev/mmcblk1p1 rootwait rw'
SMARC U-Boot > saveenv
```

3.7 Bootloader settings for booting from onboard eMMC

1) Please turn on the Pin 1 of the DIP switch as below for booting from onboard eMMC.



2) Insert SD card on SD socket. Connect RS232 cross over cable from **COM1** of mianboard to COM port of Host PC.
3) Run hyper terminal program on Host PC (teraterm on Windows or minicom on Linux)
4) Power on mainboard and press "space" key to get into bootloader menu.



```
U-Boot 2009.08 (Dec 11 2013 - 17:57:30)

CPU: Freescale i.MX6 family TO1.1 at 792 MHz
Thermal sensor with ratio = 178
Temperature:   38 C, calibration data 0x5694e069
mx6q pll1: 792MHz
mx6q pll2: 528MHz
mx6q pll3: 480MHz
mx6q pll8: 50MHz
ipg clock     : 66000000Hz
ipg per clock : 66000000Hz
uart clock    : 80000000Hz
cspi clock    : 60000000Hz
ahb clock     : 132000000Hz
axi clock   : 198000000Hz
emi_slow clock: 99000000Hz
ddr clock     : 396000000Hz
usdhc1 clock  : 198000000Hz
usdhc2 clock  : 198000000Hz
usdhc3 clock  : 198000000Hz
usdhc4 clock  : 198000000Hz
nfc clock     : 24000000Hz
Board: i.MX6DL/Solo: SMARC Board: 0x61011 [POR ]
Boot Device: SD
DRAM:   1 GB
MMC:   FSL_USDHC: 0,FSL_USDHC: 1,FSL_USDHC: 2,FSL_USDHC: 3
In:    serial
Out:   serial
Err:   serial
Net:   got MAC address from IIM: 00:00:00:00:00:00
FEC0 [PRIME]
Hit any key to stop autoboot:  0
MX6 Solo SMARC U-Boot >
```

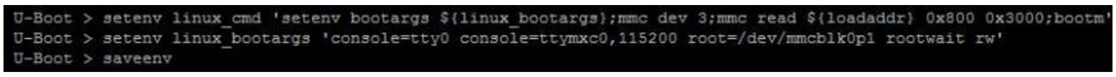
5) Setup boot device

   SMARC U-Boot > print

6) Set boot device as below

   SMARC U-Boot >setenv linux_cmd 'setenv bootargs ${linux_bootargs};mmc dev 3;mmc read ${loadaddr} 0x800 0x3000;bootm'

```
→ SMARC U-Boot > setenv linux_bootargs 'console=tty0 console=ttymxc0,115200
                        root=/dev/mmcblk0p1 rootwait rw'
→SMARC U-Boot> saveenv
→SMARC U-Boot> boot
```

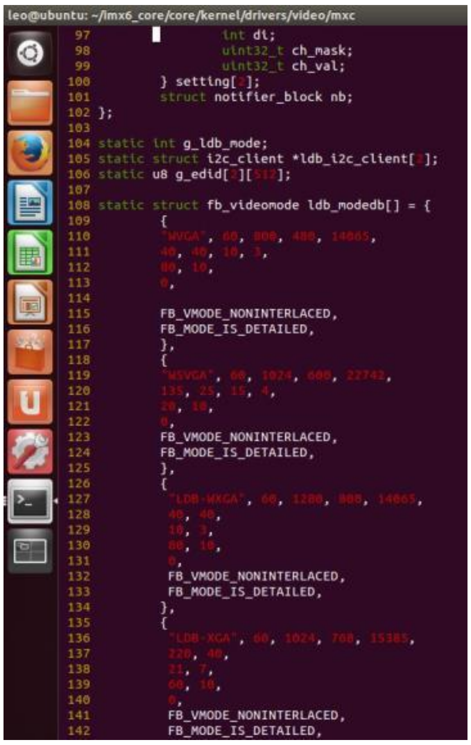


```
U-Boot > setenv linux_cmd 'setenv bootargs ${linux_bootargs};mmc dev 3;mmc read ${loadaddr} 0x800 0x3000;bootm'
U-Boot > setenv linux_bootargs 'console=tty0 console=ttymxc0,115200 root=/dev/mmcblk0p1 rootwait rw'
U-Boot > saveenv
```

3.8 Display output application of IMX6

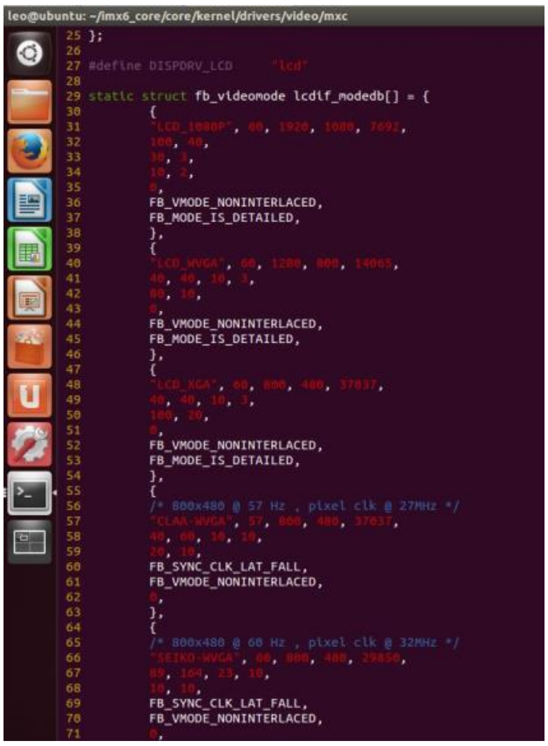
This section describes how to setup the display output for LVDS, HDMI, VGA of IMX6 module.

1. You can find the file of resolution setup of LVDS&VGA under the directory ”..~/core/kernel/drivers/video/mxc”, for LVDS is ”ldb.c”, and for VGA is ” mxc_lcdif.c”.



```
leo@ubuntu: ~/imx6_core/core/kernel/drivers/video/mxc
                        int di;
                        uint32_t ch_mask;
                        uint32_t ch_val;
                } setting[2];
                struct notifier_block nb;
};

static int g_ldb_mode;
static struct i2c_client *ldb_i2c_client[2];
static u8 g_edid[2][512];

static struct fb_videomode ldb_modedb[] = {
        {
        "WVGA", 60, 800, 480, 14065,
        40, 40, 10, 3,
        80, 10,
        0,

        FB_VMODE_NONINTERLACED,
        FB_MODE_IS_DETAILED,
        },
        {
        "WSVGA", 60, 1024, 600, 22742,
        135, 25, 15, 4,
        20, 10,
        0,
        FB_VMODE_NONINTERLACED,
        FB_MODE_IS_DETAILED,
        },
        {
         "LDB-WXGA", 60, 1280, 800, 14065,
         40, 40,
         10, 3,
         80, 10,
         0,
         FB_VMODE_NONINTERLACED,
         FB_MODE_IS_DETAILED,
        },
        {
         "LDB-XGA", 60, 1024, 768, 15385,
         220, 40,
         21, 7,
         60, 10,
         0,
         FB_VMODE_NONINTERLACED,
         FB_MODE_IS_DETAILED,
```

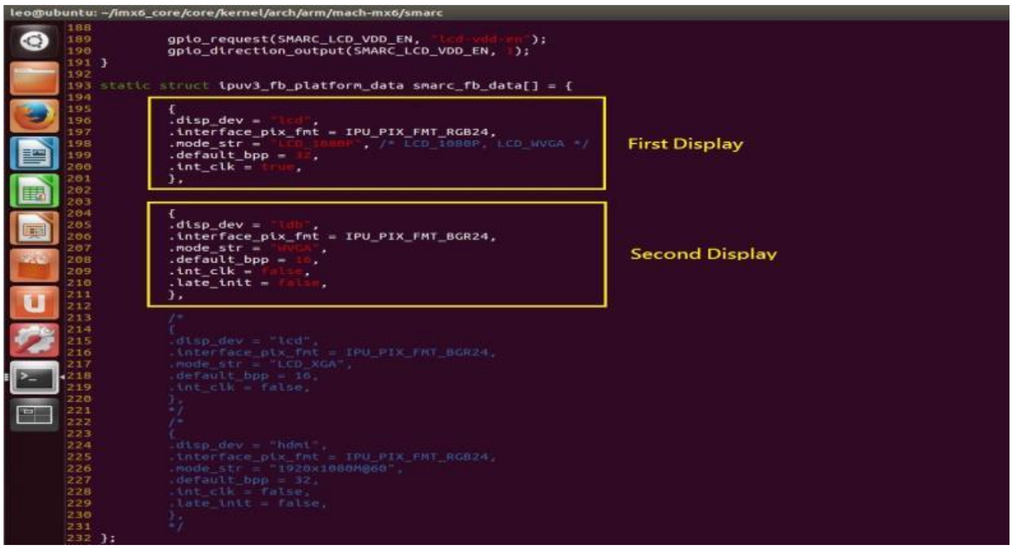
”ldb.c”



```
leo@ubuntu: ~/imx6_core/core/kernel/drivers/video/mxc
};

#define DISPDRV_LCD        "lcd"

static struct fb_videomode lcdif_modedb[] = {
        {
        "LCD_1080P", 60, 1920, 1080, 7692,
        100, 40,
        30, 3,
        10, 2,
        0,
        FB_VMODE_NONINTERLACED,
        FB_MODE_IS_DETAILED,
        },
        {
        "LCD_WVGA", 60, 1280, 800, 14065,
        40, 40, 10, 3,
        80, 10,
        0,
        FB_VMODE_NONINTERLACED,
        FB_MODE_IS_DETAILED,
        },
        {
        "LCD_XGA", 60, 800, 480, 37037,
        40, 40, 10, 3,
        100, 20,
        0,
        FB_VMODE_NONINTERLACED,
        FB_MODE_IS_DETAILED,
        },
        {
        /* 800x480 @ 57 Hz , pixel clk @ 27MHz */
        "CLAA-WVGA", 57, 800, 480, 37037,
        40, 60, 10, 10,
        20, 10,
        FB_SYNC_CLK_LAT_FALL,
        FB_VMODE_NONINTERLACED,
        0,
        },
        {
        /* 800x480 @ 60 Hz , pixel clk @ 32MHz */
        "SEIKO-WVGA", 60, 800, 480, 29850,
        89, 164, 23, 10,
        10, 10,
        FB_SYNC_CLK_LAT_FALL,
        FB_VMODE_NONINTERLACED,
        0,
```

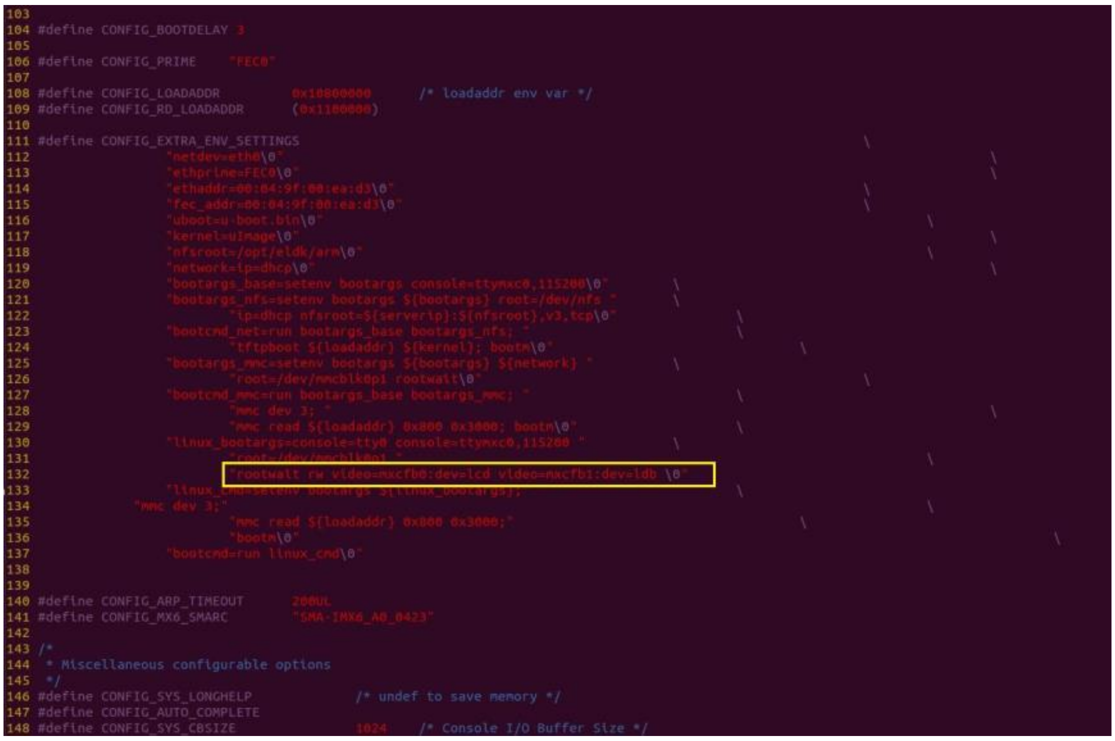
”mxc_lcdif.c”

2. You need to fill the resolution parameter(.mode_str) on the file ”rev_sa01.c” under ”~/core/kernel/arch/arm/mach-mx6/smarc/”, and the code of First Display is alway in the upper block.



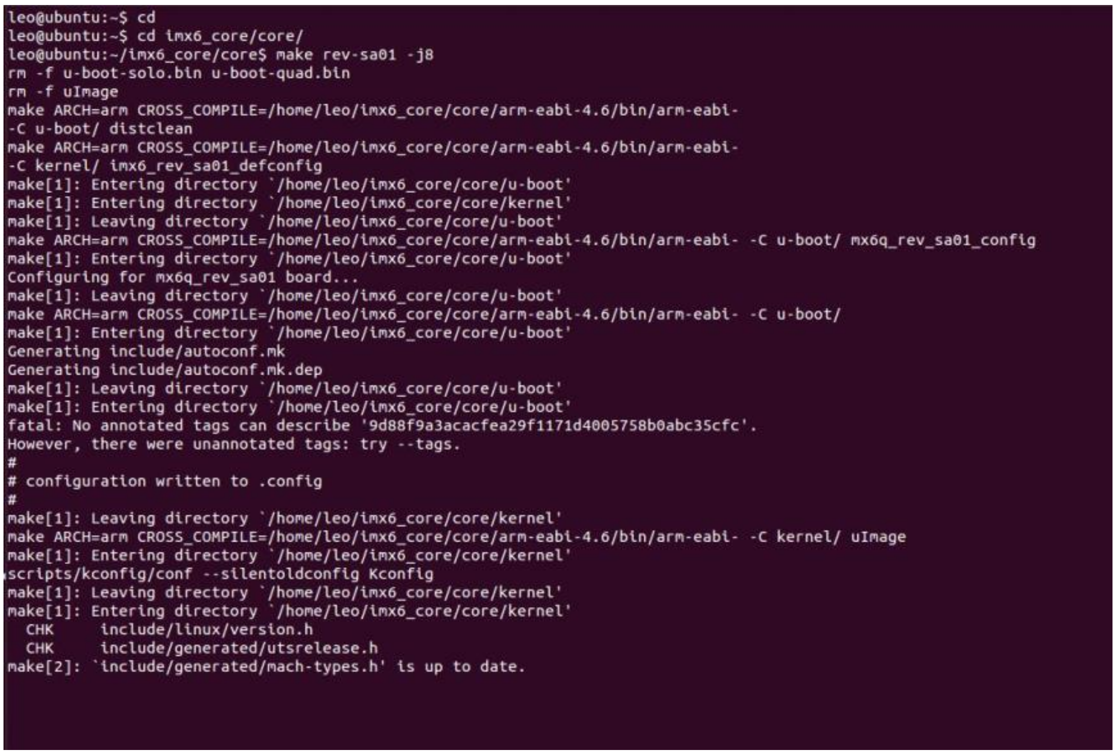
```
leo@ubuntu: ~/imx6_core/core/kernel/arch/arm/mach-mx6/smarc

                gpio_request(SMARC_LCD_VDD_EN, "lcd-vdd-en");
                gpio_direction_output(SMARC_LCD_VDD_EN, 1);
}

static struct ipuv3_fb_platform_data smarc_fb_data[] = {

        {
        .disp_dev = "lcd",
        .interface_pix_fmt = IPU_PIX_FMT_RGB24,
        .mode_str = "LCD_1080P", /* LCD_1080P, LCD_WVGA */
        .default_bpp = 32,
        .int_clk = true,
        },

        {
        .disp_dev = "ldb",
        .interface_pix_fmt = IPU_PIX_FMT_BGR24,
        .mode_str = "WVGA",
        .default_bpp = 16,
        .int_clk = false,
        .late_init = false,
        },
        /*
        {
        .disp_dev = "lcd",
        .interface_pix_fmt = IPU_PIX_FMT_BGR24,
        .mode_str = "LCD_XGA",
        .default_bpp = 16,
        .int_clk = false,
        },
        */
        /*
        {
        .disp_dev = "hdmi",
        .interface_pix_fmt = IPU_PIX_FMT_RGB24,
        .mode_str = "1920x1080M@60",
        .default_bpp = 32,
        .int_clk = false,
        .late_init = false,
        },
        */
};
```

3. Finally, you should fill the parameter for booting on the file ”**mx6_smarc.h**” under the directory ”~/core/u-boot/include/configs/”.
   Please add ”video=mxcfb0:dev=**display name**” on the column 132 "rootwait rw \0" to enable display output function when booting.



```
103
104 #define CONFIG_BOOTDELAY 3
105
106 #define CONFIG_PRIME    "FEC0"
107
108 #define CONFIG_LOADADDR         0x10800000      /* loadaddr env var */
109 #define CONFIG_RD_LOADADDR      (0x1100000)
110
111 #define CONFIG_EXTRA_ENV_SETTINGS                                       \
112                 "netdev=eth0\0"                                                 \
113                 "ethprime=FEC0\0"                                               \
114                 "ethaddr=00:04:9f:00:ea:d3\0"                           \
115                 "fec_addr=00:04:9f:00:ea:d3\0"                          \
116                 "uboot=u-boot.bin\0"                                    \
117                 "kernel=uImage\0"                                               \
118                 "nfsroot=/opt/eldk/arm\0"                                       \
119                 "network=ip=dhcp\0"                                             \
120                 "bootargs_base=setenv bootargs console=ttymxc0,115200\0"        \
121                 "bootargs_nfs=setenv bootargs ${bootargs} root=/dev/nfs "       \
122                         "ip=dhcp nfsroot=${serverip}:${nfsroot},v3,tcp\0"       \
123                 "bootcmd_net=run bootargs_base bootargs_nfs; "                  \
124                         "tftpboot ${loadaddr} ${kernel}; bootm\0"               \
125                 "bootargs_mmc=setenv bootargs ${bootargs} ${network} "          \
126                         "root=/dev/mmcblk0p1 rootwait\0"                        \
127                 "bootcmd_mmc=run bootargs_base bootargs_mmc; "                  \
128                         "mmc dev 3; "                                           \
129                         "mmc read ${loadaddr} 0x800 0x3000; bootm\0"            \
130                 "linux_bootargs=console=tty0 console=ttymxc0,115200 "           \
131                         "root=/dev/mmcblk0p1 "                                  \
132                         "rootwait rw video=mxcfb0:dev=lcd video=mxcfb1:dev=ldb \0"
133                 "linux_cmd=setenv bootargs ${linux_bootargs};"                  \
134         "mmc dev 3;"                                                            \
135                         "mmc read ${loadaddr} 0x800 0x3000;"                    \
136                         "bootm\0"                                                       \
137                 "bootcmd=run linux_cmd\0"
138
139
140 #define CONFIG_ARP_TIMEOUT      200UL
141 #define CONFIG_MX6_SMARC        "SMA-IMX6_A0_0423"
142
143 /*
144  * Miscellaneous configurable options
145  */
146 #define CONFIG_SYS_LONGHELP             /* undef to save memory */
147 #define CONFIG_AUTO_COMPLETE
148 #define CONFIG_SYS_CBSIZE               1024    /* Console I/O Buffer Size */
```

4. Please refer ch1.3 to re-build the u-boot &Kernel binary file for booting.



```
leo@ubuntu:~$ cd
leo@ubuntu:~$ cd imx6_core/core/
leo@ubuntu:~/imx6_core/core$ make rev-sa01 -j8
rm -f u-boot-solo.bin u-boot-quad.bin
rm -f uImage
make ARCH=arm CROSS_COMPILE=/home/leo/imx6_core/core/arm-eabi-4.6/bin/arm-eabi-
-C u-boot/ distclean
make ARCH=arm CROSS_COMPILE=/home/leo/imx6_core/core/arm-eabi-4.6/bin/arm-eabi-
-C kernel/ imx6_rev_sa01_defconfig
make[1]: Entering directory `/home/leo/imx6_core/core/u-boot'
make[1]: Entering directory `/home/leo/imx6_core/core/kernel'
make[1]: Leaving directory `/home/leo/imx6_core/core/u-boot'
make ARCH=arm CROSS_COMPILE=/home/leo/imx6_core/core/arm-eabi-4.6/bin/arm-eabi- -C u-boot/ mx6q_rev_sa01_config
make[1]: Entering directory `/home/leo/imx6_core/core/u-boot'
Configuring for mx6q_rev_sa01 board...
make[1]: Leaving directory `/home/leo/imx6_core/core/u-boot'
make ARCH=arm CROSS_COMPILE=/home/leo/imx6_core/core/arm-eabi-4.6/bin/arm-eabi- -C u-boot/
make[1]: Entering directory `/home/leo/imx6_core/core/u-boot'
Generating include/autoconf.mk
Generating include/autoconf.mk.dep
make[1]: Leaving directory `/home/leo/imx6_core/core/u-boot'
make[1]: Entering directory `/home/leo/imx6_core/core/u-boot'
fatal: No annotated tags can describe '9d88f9a3acacfea29f1171d4005758b0abc35cfc'.
However, there were unannotated tags: try --tags.
#
# configuration written to .config
#
make[1]: Leaving directory `/home/leo/imx6_core/core/kernel'
make ARCH=arm CROSS_COMPILE=/home/leo/imx6_core/core/arm-eabi-4.6/bin/arm-eabi- -C kernel/ uImage
make[1]: Entering directory `/home/leo/imx6_core/core/kernel'
scripts/kconfig/conf --silentoldconfig Kconfig
make[1]: Leaving directory `/home/leo/imx6_core/core/kernel'
make[1]: Entering directory `/home/leo/imx6_core/core/kernel'
  CHK     include/linux/version.h
  CHK     include/generated/utsrelease.h
make[2]: `include/generated/mach-types.h' is up to date.
```

Note: If you need to use double display output in Ubuntu, you should setup the ” rev_sa01.c” file first then add the content ”video=mxcfb0:dev=first display name video=mxcfb1:dev=second display neme” to mx6_smarc.h, but for this application, you also need to write a program for controling the second diplay first or the second display will not enable after you follow up all the setting above.

3.9 Download Android Source Code for building image file

Please make a folder for storing the source code first then typing the command below to get started for the source code download.

```
$ sudo apt-get install git
$ git clone guest@202.55.227.57:freescale/imx6/Android.git -b 4.4.2-SMARC   _
```

About password, please check with our support team.

3.10 Set up for building Android image file

We support to compile u-boot & Kernel on Ubuntu 12.04 (64bit version), other version of Ubuntu is not currently supported and may have built issues.

Install host packages needed by building code. This document assumes you are using Ubuntu. Not a requirement, but the packages may be named differently and the method of installing them may be different.

1) Please follow up the commands below to install "Oracle JDK6.0" first for building up Android image file.

```
$ sudo apt-get install python-software-properties
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java6-installer
```

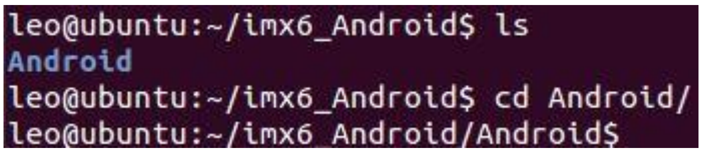
2) Please follow up the commands below to install the necessary package for build image file.

```
$ sudo apt-get install git-core gnupg flex bison gperf build-essential \
    zip curl libc6-dev libncurses5-dev x11proto-core-dev \
    libx11-dev:i386 libreadline6-dev:i386 \
    libgl1-mesa-dev g++-multilib mingw32 openjdk-6-jdk tofrodos \
    python-markdown libxml2-utils xsltproc zlib1g-dev:i386 \
    ia32-libs u-boot-tools minicom lib32ncurses5-dev \
```

3.11 Building up Andrioid image file

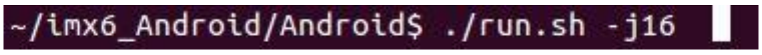
You can follow up the steps below to compile Android image file after download the source code.

1. Please move to the folder "Android" then start to compile image file.



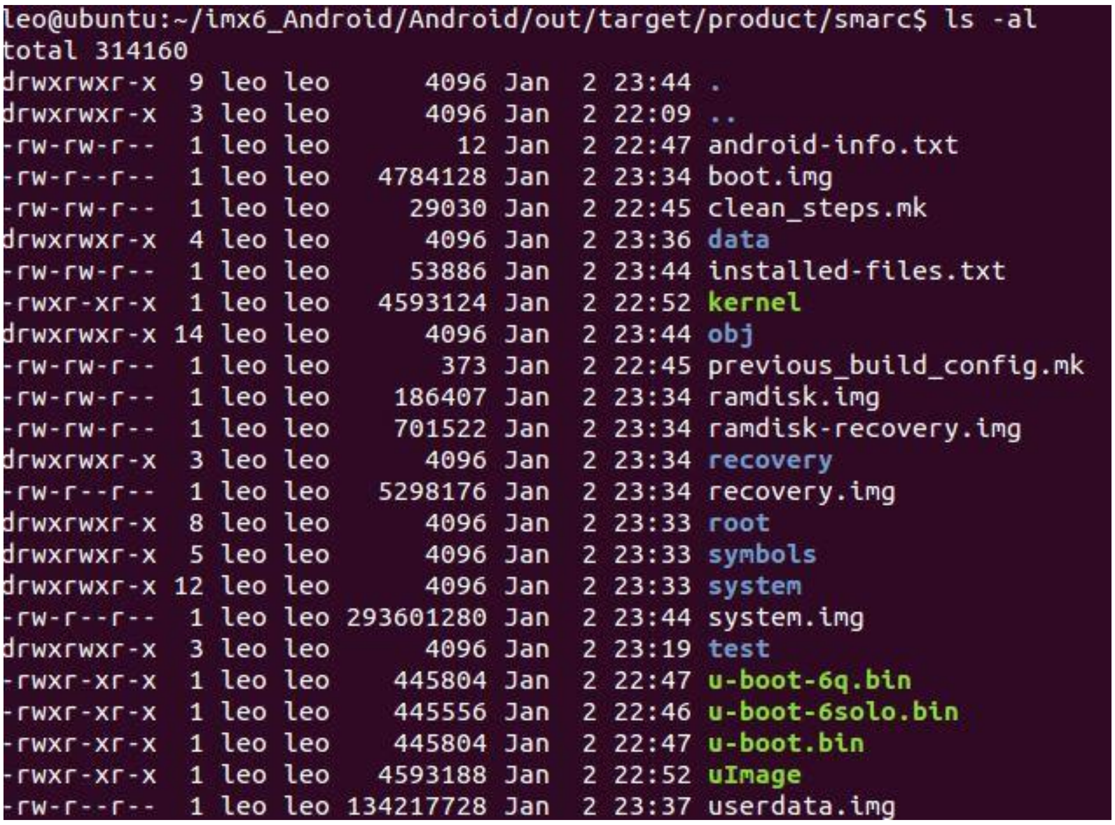
```
leo@ubuntu:~/imx6_Android$ ls
Android
leo@ubuntu:~/imx6_Android$ cd Android/
leo@ubuntu:~/imx6_Android/Android$
```

2. Type the command to compile image file.
   $ **./run.sh –j16**
   (**-j number** means multi jobs for more efficiant building, you can add it according to your CPU performance of PC, e.g. mine is "–j16" as below).



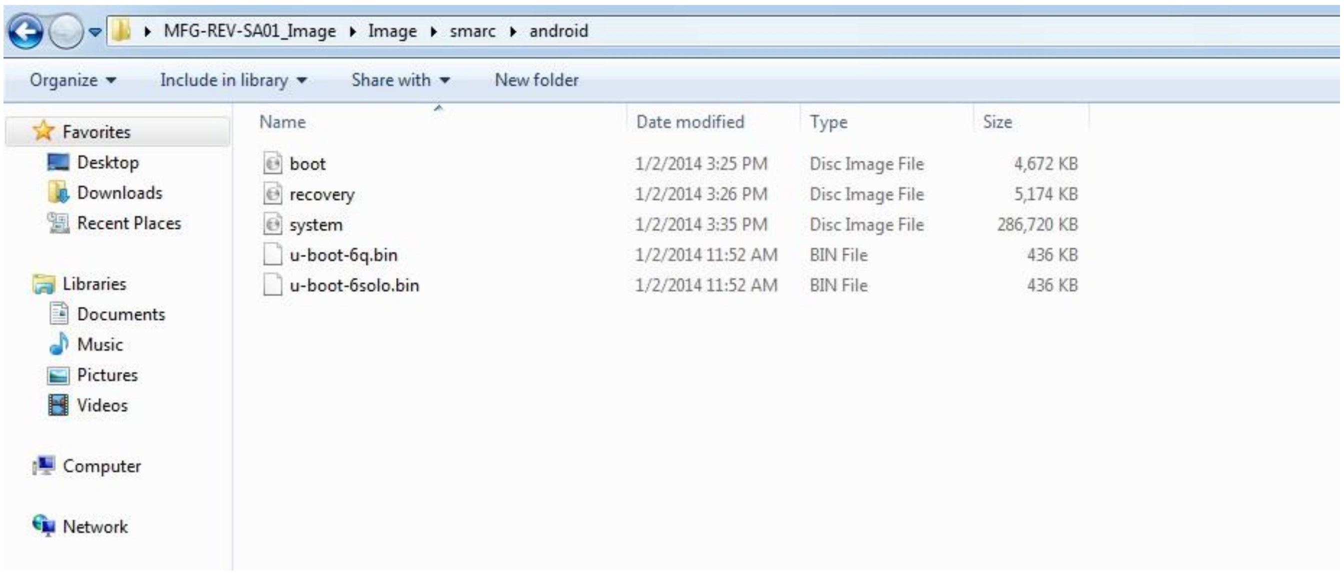
```
~/imx6_Android/Android$ ./run.sh -j16
```

3. You can find the finished image file(**u-boot-6q.bin, u-boot-6solo.bin, system.img, recover.img, boot.img**) as below after compiling on the directory **~/Android/out/target/product/smarc.**



```
leo@ubuntu:~/imx6_Android/Android/out/target/product/smarc$ ls -al
total 314160
drwxrwxr-x  9 leo leo      4096 Jan  2 23:44 .
drwxrwxr-x  3 leo leo      4096 Jan  2 22:09 ..
-rw-rw-r--  1 leo leo        12 Jan  2 22:47 android-info.txt
-rw-r--r--  1 leo leo   4784128 Jan  2 23:34 boot.img
-rw-rw-r--  1 leo leo     29030 Jan  2 22:45 clean_steps.mk
drwxrwxr-x  4 leo leo      4096 Jan  2 23:36 data
-rw-rw-r--  1 leo leo     53886 Jan  2 23:44 installed-files.txt
-rwxr-xr-x  1 leo leo   4593124 Jan  2 22:52 kernel
drwxrwxr-x 14 leo leo      4096 Jan  2 23:44 obj
-rw-rw-r--  1 leo leo       373 Jan  2 22:45 previous_build_config.mk
-rw-rw-r--  1 leo leo    186407 Jan  2 23:34 ramdisk.img
-rw-rw-r--  1 leo leo    701522 Jan  2 23:34 ramdisk-recovery.img
drwxrwxr-x  3 leo leo      4096 Jan  2 23:34 recovery
-rw-r--r--  1 leo leo   5298176 Jan  2 23:34 recovery.img
drwxrwxr-x  8 leo leo      4096 Jan  2 23:33 root
drwxrwxr-x  5 leo leo      4096 Jan  2 23:33 symbols
drwxrwxr-x 12 leo leo      4096 Jan  2 23:33 system
-rw-r--r--  1 leo leo 293601280 Jan  2 23:44 system.img
drwxrwxr-x  3 leo leo      4096 Jan  2 23:19 test
-rwxr-xr-x  1 leo leo    445804 Jan  2 22:47 u-boot-6q.bin
-rwxr-xr-x  1 leo leo    445556 Jan  2 22:46 u-boot-6solo.bin
-rwxr-xr-x  1 leo leo    445804 Jan  2 22:47 u-boot.bin
-rwxr-xr-x  1 leo leo   4593188 Jan  2 22:52 uImage
-rw-r--r--  1 leo leo 134217728 Jan  2 23:37 userdata.img
```

PS: If you would like to use **Mfgtool** for flashing image file, you must put all the files **u-boot-6q.bin, u-boot-6solo.bin, system.img, recover.img, boot.img** under “~\Image\smarc\android” for right detected path.

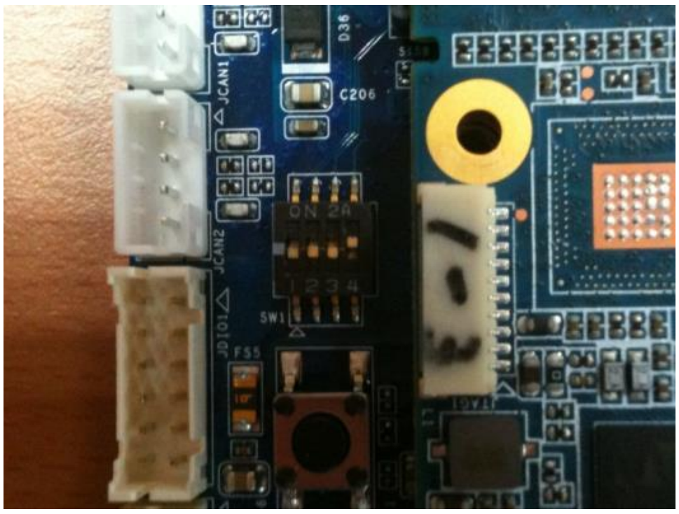


Name	Date modified	Type	Size
boot	1/2/2014 3:25 PM	Disc Image File	4,672 KB
recovery	1/2/2014 3:26 PM	Disc Image File	5,174 KB
system	1/2/2014 3:35 PM	Disc Image File	286,720 KB
u-boot-6q.bin	1/2/2014 11:52 AM	BIN File	436 KB
u-boot-6solo.bin	1/2/2014 11:52 AM	BIN File	436 KB

3.12 Use MfgTool to flash Android into onboard eMMC

Manufacturing tool, a successor of ATK, provides a series of new features to power your mass production work. The features like windows style GUI, multiple devices support, explicit status monitoring, versatile functionalities and highly flexible architecture make it a best choice to meet your critical timing, cost and customization requirements.
For using Mfgtool to flash image file into onboard eMMC, please follow up the steps below

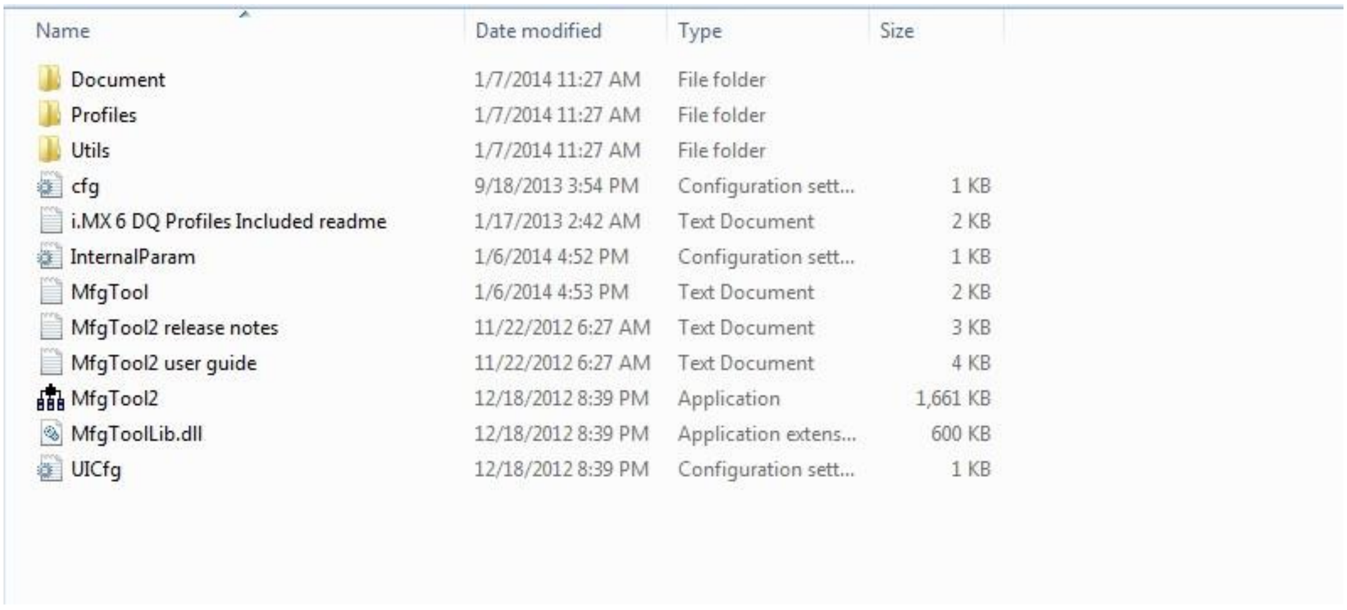
1) Please turn on the **Pin4** of the DIP switch as below into burning mode of Mfgtool.



2) Power on the mainboard then plug the cable from OTG socket to PC.
3) Select the right folder by MCU name, if the MCU of module board is "**i.MX6 Solo**", please click the folder "~\ **MX6DL-IMX6**" to flash image file, e.g. mine is D:\ MFG-REV-SA01_Image**MX6DL-IMX6**"

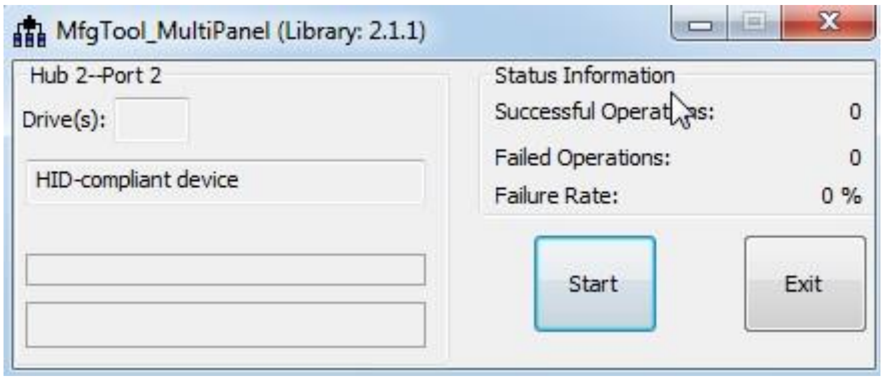
On the other hand, if the MCU of module board is "**i.MX6 Quad core**", please click the folder "~\ **MX6Q-IMX6**" to flash image file, e.g. mine is
D:\ MFG-REV-SA01_Image\ **MX6Q-IMX6**"

4) Click "MfgTool2.exe" to flash image file into smarc module.



Name	Date modified	Type	Size
Document	1/7/2014 11:27 AM	File folder	
Profiles	1/7/2014 11:27 AM	File folder	
Utils	1/7/2014 11:27 AM	File folder	
cfg	9/18/2013 3:54 PM	Configuration sett...	1 KB
i.MX 6 DQ Profiles Included readme	1/17/2013 2:42 AM	Text Document	2 KB
InternalParam	1/6/2014 4:52 PM	Configuration sett...	1 KB
MfgTool	1/6/2014 4:53 PM	Text Document	2 KB
MfgTool2 release notes	11/22/2012 6:27 AM	Text Document	3 KB
MfgTool2 user guide	11/22/2012 6:27 AM	Text Document	4 KB
MfgTool2	12/18/2012 8:39 PM	Application	1,661 KB
MfgToolLib.dll	12/18/2012 8:39 PM	Application extens...	600 KB
UICfg	12/18/2012 8:39 PM	Configuration sett...	1 KB

5) Click "**Start**" to flash image file.



MfgTool_MultiPanel (Library: 2.1.1)

Hub 2--Port 2

Drive(s):

HID-compliant device

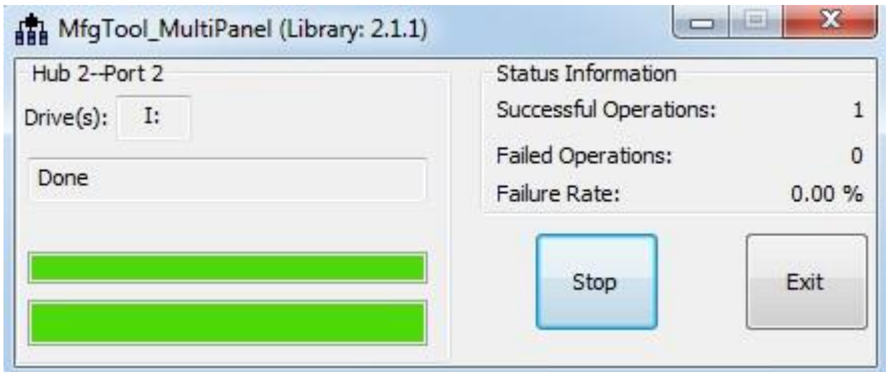
Status Information

Successful Operations: 0

Failed Operations: 0

Failure Rate: 0 %

6) It will show "Done" after flashing is finish, then click "Stop" and "Exit" to close the screen.



MfgTool_MultiPanel (Library: 2.1.1)

Hub 2--Port 2

Drive(s): I:

Done

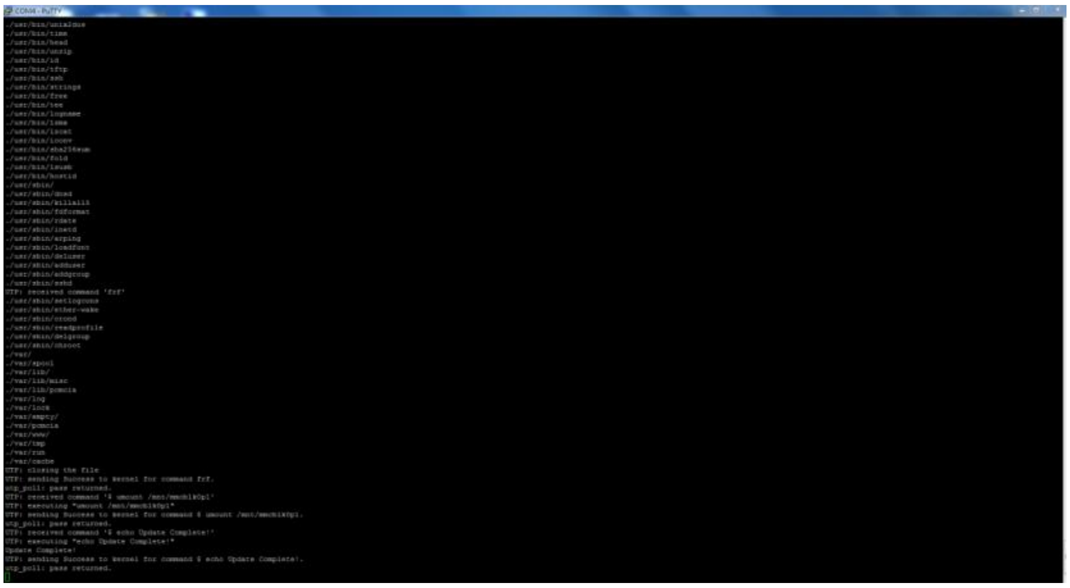
Status Information

Successful Operations: 1

Failed Operations: 0

Failure Rate: 0.00 %

7) You can also get the information from Terminal (debug port→COM1) after flashing is finish.



```
./usr/bin/unix2dos
./usr/bin/time
./usr/bin/head
./usr/bin/unzip
./usr/bin/id
./usr/bin/tftp
./usr/bin/ssh
./usr/bin/strings
./usr/bin/free
./usr/bin/tee
./usr/bin/logname
./usr/bin/lzma
./usr/bin/lzcat
./usr/bin/lzop
./usr/bin/sha256sum
./usr/bin/fold
./usr/bin/lsusb
./usr/bin/hostid
./usr/sbin/
./usr/sbin/dnsd
./usr/sbin/killall5
./usr/sbin/fdformat
./usr/sbin/rdate
./usr/sbin/inetd
./usr/sbin/arping
./usr/sbin/loadfont
./usr/sbin/deluser
./usr/sbin/adduser
./usr/sbin/addgroup
./usr/sbin/sshd
UTP: received command 'frf'
./usr/sbin/setlogcons
./usr/sbin/ether-wake
./usr/sbin/crond
./usr/sbin/readprofile
./usr/sbin/delgroup
./usr/sbin/chroot
./var/
./var/spool
./var/lib/
./var/lib/misc
./var/lib/pcmcia
./var/log
./var/lock
./var/empty/
./var/pcmcia
./var/www/
./var/tmp
./var/run
./var/cache
UTP: closing the file
UTP: sending Success to kernel for command frf.
utp_poll: pass returned.
UTP: received command '$ umount /mnt/mmcblk0p1'
UTP: executing "umount /mnt/mmcblk0p1"
UTP: sending Success to kernel for command $ umount /mnt/mmcblk0p1.
utp_poll: pass returned.
UTP: received command '$ echo Update Complete!'
UTP: executing "echo Update Complete!"
Update Complete!
UTP: sending Success to kernel for command $ echo Update Complete!.
utp_poll: pass returned.
```