QUICK START GUIDE

VAB-600

Linux BSP v1.2.1

100-05202016-113800

Revision History

Version	Date	Remarks
1.00	5/19/2016	Initial release

Table of Contents

1. Introduction

The purpose of this document is to provide an overview of getting started with the VAB-600 board using Debian Linux.

1.1. Package Contents

There are three folders in the package as listed below.

BSP folder	Description
VAB-600_Linux_source_code.zip	Source Code
Document folder	**Description**
VAB-600_Linux_BSP_v1.2.1_Quick_Start_Guide_v1.00_20160519.pdf	Quick Start Guide
VAB-600_Linux_EVK_v1.2.1_Image_Installation_Guide_v1.00_20160519.pdf	Image Installation Guide
VAB-600_Linux_EVK_CL_v1.2.1_Image_Installation_Guide_v1.00_20160519.pdf	Command Line Only Image Installation Guide
EVK folder	**Description**
arm_201103_gcc4.5.2.tgz	Toolchain
VAB-600_Linux_EVK_v1.2.1.zip	Precompiled Image
VAB-600_Linux_EVK_CL_v1.2.1.zip	Precompiled Command Line Only Image
VAB600-Debian7.0-Graphics-v1.0.4.tgz	Graphics Driver
VAB600-Debian7.0-Multimedia-v1.0.3.tgz	Multimedia Driver

VAB-600 Linux BSP contents

1.1.1. BSP Folder Contents

VAB-600_Linux_source_code.zip: is a complete Debian BSP including kernel source code, U-Boot source code, and none built-in driver source code of the EMIO-1533 USB Wi-Fi module.

1.1.2. Document Folder Contents

VAB-600_Linux_BSP_v1.2.1_Quick_Start_Guide_v1.00_20160519.pdf: The Quick Start Guide is to provide an overview of getting started with the VAB-600 board using Debian Linux.

VAB-600_Linux_EVK_v1.2.1_Image_Installation_Guide_v1.00_20160519.pdf: The Image Installation Guide explains how to boot the Linux EVK system image on the VAB-600 board in order to begin evaluating the platform.

VAB-600_Linux_EVK_CL_v1.2.1_Image_Installation_Guide_v1.00_20160519.pdf: The Image Installation Guide explains how to boot the Linux EVK command line system image on the VAB-600 board in order to begin evaluating the platform.

1.1.3. EVK Folder Contents

arm_201103_gcc4.5.2.tgz: is a toolchain, which is a set of software development tools for building images for the VAB-600 board.

VAB-600_Linux_EVK_v1.2.1.zip: is a precompiled Debian image for evaluating the VAB-600 board.

VAB-600_Linux_EVK_CL_v1.2.1.zip: is a precompiled command line only Debian image for evaluating the VAB-600 board.

VAB600-Debian7.0-Graphics-v1.0.4.tgz: contains the graphics driver for the Display (Xorg), 2D (EXA), 3D (OpenGL ES), video decoder and Mali memory management.

VAB600-Debian7.0-Multimedia-v1.0.3.tgz: contains the multimedia package for the Gstreamer 0.10.35 environment and ffmpeg plugin.

1.2. Version Information and Supported Features

- U-Boot version: 1.1.4
- Kernel version: 3.0.8
- Evaluation image: Debian Wheezy 7.0
- Development based on VIA WM8950_Linux_BSP_1.00.02_121212
- Supports eMMC (default) or Micro SD boot
- Supports HDMI or LVDS single display (default HDMI output)
- Supports HDMI audio output
- Supports Acmepoint TFT-LCD resistive touch panel (through 4-wire interface)
  Acmepoint 7" TP070C01 (800×480)
- Supports AUO LVDS resistive touch panel (through USB interface)
  AUO 22" G220SVN01.0 (1680×1050)
- Supports COM1 debug port
- Supports CIR
- Supports Line-Out and Mic-In
- Supports USB: keyboard, mouse, flash disk, hard disk, UVC camera
- Supports10M/100M Ethernet
- Supports EMIO-1533 USB Wi-Fi module
- Supports EMIO-2550 miniPCIe Mobile Broadband module
- Supports Watchdog, GPIO, and RTC wake-up

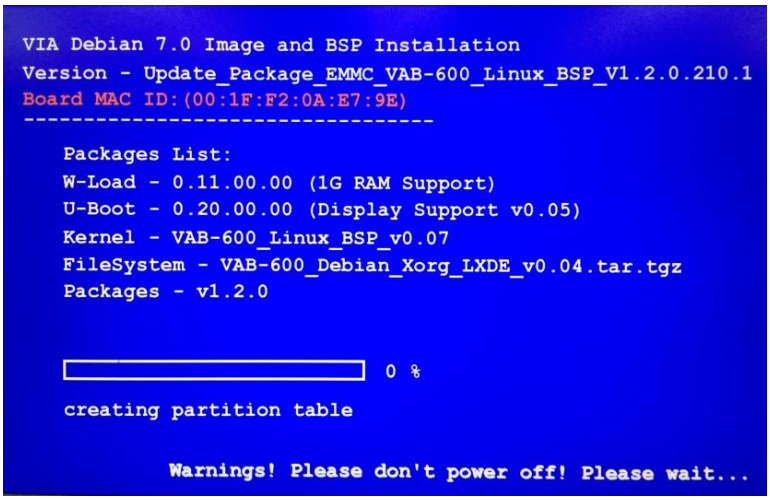
2. Image Development

This section explains the setup requirements for booting from the eMMC (default) or a Micro SD card.

2.1. Booting from the eMMC

The first step is to insert a Micro SD card into the host machine and create a FAT formatted partition. Next, extract the **VAB-600_Linux_EVK_v1.2.1.zip** or **VAB-600_Linux_EVK_CL_v1.2.1.zip** (command line only) and copy the **bspinst** folder and **scriptcmd** file onto the Micro SD card.

Insert the prepared Micro SD card into the VAB-600, connect an HDMI display, and power on the device to initiate the update process.



Update process screen

When the update process is completed, remove the Micro SD card. The system will automatically shut down in 3 seconds.

Power on the VAB-600 again, and wait for the login prompt to appear.

The default user name is *debian* and the password is *temppwd*.

2.2. Booting from a Micro SD Card

The first step is to insert a Micro SD card (4GB or larger) into your host machine and make sure it is not mounted. Delete all partitions on the Micro SD card then create a FAT partition with 200MB, and name it **kernel**. Next, create an EXT3 partition with the remaining space and name it **fs**, then mount the **kernel** and **fs** partitions.

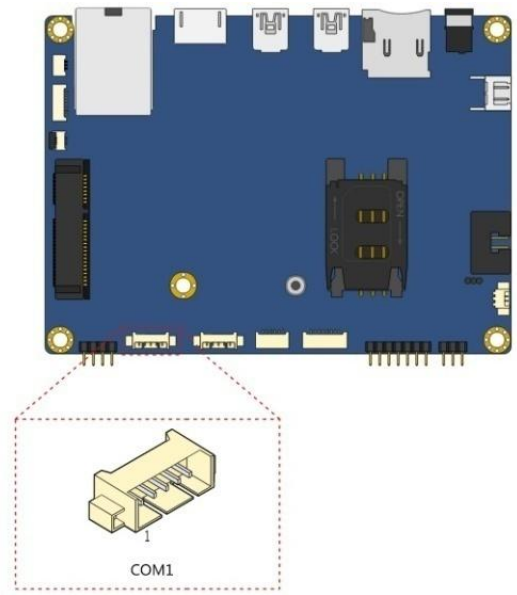
Extract the contents from the **VAB-600_Linux_EVK_CL_v1.2.1.zip** (command line only) to the host machine. Copy the kernel image **uzImage.bin** into the **bspinst** folder on the **kernel** partition.

Extract the root file system from **armel-rootfs-20151203_v0.02.tgz** to the **fs** partition.

```
$ tar -xzvf armel-rootfs-20151203_v0.02.tgz -C /media/fs
```

Once the extraction procedure has completed, insert the precompiled Micro SD card into the VAB-600 board.

To begin setting up the U-Boot parameters, connect the host machine and the VAB-600 board through the debug COM1 port. Use a serial port communication program such as Minicom or GtkTerm to connect to the debug console. There you will be able to see the U-Boot boot log and adjust settings in the U-Boot console.



Debug COM1 port diagram

```
+-----------------------------------------------------------------+
|  A    -        Serial Device               : /dev/ttyS0       |
|  B    -      Lockfile Location             : /var/lock        |
|  C    -        Callin Program              :                  |
|  D    -       Callout Program              :                  |
|  E    -         Bps/Par/Bits               : 115200 8N1       |
|  F    -      Hardware Flow Control         : No               |
|  G    -      Software Flow Control         : No               |
+-----------------------------------------------------------------+
```

Serial port setting of host machine

Next, power on the VAB-600 board to initiate the boot process. When prompted, press any key to stop the boot process and use the following command to set the Micro SD card as the boot device.

```
WMT # setenv boot-sd "mmcinit0; fatload mmc 0 ${cfg.load-addr-kernel} uzimage.bin; if iminfo
${cfg.load-addr-kernel}; then run set-rfs-sd; bootm ${cfg.load-addr-kernel}; fi"
WMT # setenv set-rfs-sd "setenv bootargs mem=${memtotal} root=/dev/mmcblk0p2 rw
rootfstype=ext3 noinitrd mbtotal=${mbsize} ${bootargs-common} ${bootargs-extra}"
WMT # setenv boot-method boot-sd
WMT # saveenv
WMT # reset
```

To boot from the eMMC, restore the U-boot parameters by using the following command:

```
WMT # setenv boot-method boot-emmc
WMT # saveenv
WMT # reset
```

When the reboot process is completed, you will be prompted to login.

The default user name is *debian* and the password is *temppwd*.

3. Graphical Environment Installation

This section guides you through adding a graphical environment to the command line version of Debian.

3.1. Installing the Window Manager

Plug-in the ethernet cable into the VAB-600 board, and then enter the following command to get the DHCP IP in order to activate the internet connection.

```
$ sudo dhclient eth0
```

Next, update the source list.

```
$ sudo apt-get update
```

If **apt-get** gives a warning message, type the following command to update the debian-archive-keyring package.

```
$ sudo apt-get install debian-archive-keyring
```

When the update has completed, type the following command to install **Xorg** and **LXDE**.

```
$ sudo apt-get install xorg
$ sudo apt-get install lxde
```

3.2. Installing the Graphics Driver Package

The graphics driver package contains the Display (Xorg), 2D (EXA), 3D (OpenGL ES), video decoder and Mali memory management drivers.

Extract the graphics driver package from **VAB600-Debian7.0-Graphics-v1.0.4.tgz**.

```
$ tar -xzvf VAB600-Debian7.0-Graphics-v1.0.4.tgz
```

Install the graphics driver with the provided script, and then reboot the system.

```
$ cd VAB600-Debian7.0-Graphics-v1.0.4
$ sh install.sh
$ sudo reboot
```

3.3. Installing the Multimedia Package

The multimedia package contains the Gstreamer 0.10.35 environment and ffmpeg plugin.

Enter the following command to install and set locales by selecting "en_US.UTF-8" from the list of options presented.

```
$ sudo apt-get update
$ sudo apt-get install locales
$ sudo dpkg-reconfigure locales
```

Extract the multimedia driver package from **VAB600-Debian7.0-Multimedia-v1.0.3.tgz**.

```
$ tar -xzvf VAB600-Debian7.0-Multimedia-v1.0.3.tgz
```

Install the multimedia package with the provided script, and then reboot the system.

```
$ cd VAB600-Debian7.0-Multimedia-v1.0.3
$ ./install_quick.sh
$ sudo reboot
```

4. Build Environment Setup

This section guides you through setting up the build environment for development. All instructions are based on Ubuntu 12.04 x64 or higher versions.

4.1. Setting Up the Cross-Compiling Environment

Make sure that the host machine is connected to the network and run the packages update.

```
$ sudo apt-get update
```

Install the required software packages for cross-compilation.

```
$ sudo apt-get install git-core gnupg flex bison gperf build-essential \
zip curl zlib1g-dev libc6-dev lib32ncurses5-dev ia32-libs \
x11proto-core-dev libx11-dev lib32readline-gplv2-devlib32z1-dev \
libgl1-mesa-dev g++-multilib mingw32 tofrodos python-markdown \
libxml2-utils xsltprocsharutils
```

The tool to create images for use with the U-Boot boot loader,"mkimage", is provided by different packages in Ubuntu 12.04 and in newer Ubuntu releases.

On Ubuntu 12.04 install it as below.

```
$ sudo apt-get install uboot-mkimage
```

On Ubuntu 14.04 and newer versions install it as below.

```
$ sudo apt-get install u-boot-tools
```

Confirm that the GNU C library version is 2.11 or higher.

```
$ ldd-version
```

Extract the toolchain to /usr/local/arm/ (If this folder does not exist in the system, please create it manually).

```
$ tar -xzvfarm_201103_gcc4.5.2.tgz -C /usr/local/arm/
```

The cross compiler will then be found in the **/usr/local/arm/arm_201103_gcc4.5.2/** directory.

Export the toolchain to system PATH.

```
$ export PATH=/usr/local/arm/arm_201103_gcc4.5.2/mybin/:$PATH
```

To achieve the best floating point operations performance, create an alias for the toolchain.

```
# alias arm 1103 le-as='arm 1103 le-as -mcpu=cortex-a9 -mfpu=neon -mfloat-abi=softfp'
# alias arm_1103_le-c++='arm_1103_le-c++ -mcpu=cortex-a9 -mfpu=neon -mfloat-abi=softfp'
# alias arm 1103 le-cpp='arm 1103 le-cpp -mcpu=cortex-a9 -mfpu=neon -mfloat-abi=softfp'
# alias arm 1103 le-g++='arm 1103 le-g++ -mcpu=cortex-a9 -mfpu=neon -mfloat-abi=softfp'
# alias arm 1103 le-gcc='arm 1103 le-gcc -mcpu=cortex-a9 -mfpu=neon -mfloat-abi=softfp'
#  alias  arm 1103 le-gcc-4.5.2='arm 1103 le-gcc-4.5.2  -mcpu=cortex-a9  -mfpu=neon  -mfloat-
abi=softfp'
```

5. Image Build

5.1. Building the U-Boot Binary

This section describes how to build the U-Boot image from the source code.

Extract the U-Boot source.

```
$ tar -xzvf uboot.130522_Display_Support_v0.05.tgz
```

Type the following command in order to use the default configurations.

```
$ cd uboot.130522 Display Support v0.05
$ make wmt_config
```

Build the U-boot image with the ARM cross compiler.

```
$ make CROSS_COMPILE=/usr/local/arm/arm_201103_gcc4.5.2/mybin/arm_1103_le-
```

When the process is completed, the **u-boot.bin** file will be stored in the **uboot.130522_Display_Support_v0.05** directory. Copy this file to the **bspinst** folder on the SD card for U-Boot update.

5.2. Building the Linux Kernel

This section describes how to build the kernel image from the source code.

To begin, extract the kernel source.

```
$ tar -xzvf VAB-600_Linux_BSP_Kernel_Source_v0.08.tgz
```

Type the following command to use the default configurations.

```
$ cd VAB-600_Linux_BSP_Kernel_Source_v0.08/Kernel_3.0.8
$ make VAB-600_Linux_defconfig
```

Use the kernel's menuconfig to make modifications to the default settings.

```
$ make menuconfig
```

Finally, compile the kernel.

```
$ make ubin CROSS_COMPILE=arm_1103_le- -jX
```

"X" above represents the number of parallel jobs to be run by make. A reasonable default value is the number of CPU cores of your host system. For example, use "-j4" on a quad-core system.

When the compilation process is completed, the resulting **uzImage.bin** file will be stored in the root folder of the kernel source folder. Copy this file to the **bspinst** folder on the SD card for kernel update.

5.3. Building the EMIO-1533 USB Wi-Fi Module Driver

This section describes how to build the EMIO-1533 USB Wi-Fi module driver from the source code.

Extract the EMIO-1533 driver source.

```
$ tar -xzvf vt9271-202_20151209.tgz
```

Open the **vt9271-202/build/scripts/vt9271/Makefile.vt9271** file and edit "KERNELPATH" to match your own kernel source path. The following example uses **/home/work/VAB-600_Linux_BSP_Kernel_Source_v0.08/Kernel_3.0.8** as the kernel source path.

```
$ export KERNELPATH=/home/work/VAB-600_Linux_BSP_Kernel_Source_v0.08/Kernel_3.0.8
```

Type the following command in order to use the default configurations.

```
$ cd vt9271-202/build
$ make
```

When the process is completed, the **vt9271.ko** kernel module will be stored in the **vt9271-202/output/** directory. Copy this file to the **/lib/modules** folder on the VAB-600 system for EMIO-1533 USB Wi-Fi module support.

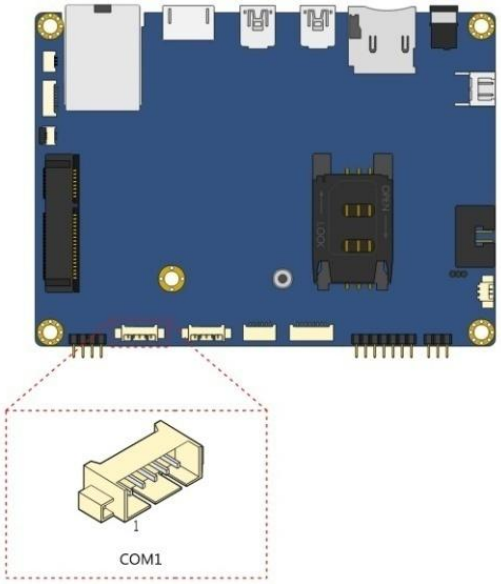
6. Hardware Functions

6.1. Configuring U-Boot Display Parameters

6.1.1. HDMI (default setting)

Connection:
Connect your display to the VAB-600 board through the Mini HDMI connector.

U-Boot:
Connect the VAB-600 board and the host machine through the debug COM1 port.



Debug COM1 port diagram

Please note that the system will automatically detect the EDID information and set the monitor default resolution. Configuring U-Boot only affects the resolution of the U-Boot logo.

Use the following command to update the U-Boot parameters.

```
WMT # setenv wmt.display.param 4:6:1:1920:1080:60
WMT # saveenv
WMT # reset
```

1920:1080 is the resolution of the HDMI monitor used in this example.

Xorg:

Edit the **/etc/X11/xorg.conf** file, commenting out all the rows containing "Modeline" by adding "#" at the beginning of those lines.

```
# Modeline "640x480_75" 30.75 640 664 728 816 480 483 487 504 -hsync +vsync
# Modeline "800x480_75" 38.50 800 832 912 1024 480 483 493 504 -hsync +vsync
# Modeline "800x600_75" 49.00 800 840 920 1040 600 603 607 629 -hsync +vsync
# Modeline "1024x768_75" 82.00 1024 1088 1192 1360 768 771 775 805 -hsync +vsync
# Modeline "1280x720_75" 95.75 1280 1360 1488 1696 720 723 728 755 -hsync +vsync
# Modeline "1280x768_75" 102.25 1280 1360 1488 1696 768 771 781 805 -hsync +vsync
# Modeline "1280x1024_75" 138.75 1280 1368 1504 1728 1024 1027 1034 1072 -hsync +vsync
# Modeline "1360x768_75" 109.00 1360 1448 1584 1808 768 771 781 805 -hsync +vsync
# Modeline "1440x900_75" 136.75 1440 1536 1688 1936 900 903 909 942 -hsync +vsync
# Modeline "1400x1050_75" 156.00 1400 1504 1648 1896 1050 1053 1057 1099 -hsync +vsync
# Modeline "1440x900_75" 136.75 1440 1536 1688 1936 900 903 909 942 -hsync +vsync
# Modeline "1920x1080_75" 220.75 1920 2064 2264 2608 1080 1083 1088 1130 -hsync +vsync
# Modeline "1680x1050" 146.25 1680 1784 1960 2240 1050 1053 1059 1089 -hsync +vsync
# Modeline "640x480" 23.75 640 664 720 800 480 483 487 500 -hsync +vsync
# Modeline "800x480" 29.50 800 824 896 992 480 483 493 500 -hsync +vsync
# Modeline "800x600" 38.25 800 832 912 1024 600 603 607 624 -hsync +vsync
# Modeline "1024x768" 63.50 1024 1072 1176 1328 768 771 775 798 -hsync +vsync
# Modeline "1280x720" 74.50 1280 1344 1472 1664 720 723 728 748 -hsync +vsync
# Modeline "1280x1024" 109.00 1280 1368 1496 1712 1024 1027 1034 1063 -hsync +vsync
# Modeline "1360x768" 84.75 1360 1432 1568 1776 768 771 781 798 -hsync +vsync
# Modeline "1400x1050" 121.75 1400 1488 1632 1864 1050 1053 1057 1089 -hsync +vsync
# Modeline "1440x900" 106.50 1440 1528 1672 1904 900 903 909 934 -hsync +vsync
# Modeline "1920x1080" 173.00 1920 2048 2248 2576 1080 1083 1088 1120 -hsync +vsync
# Modeline "800x400" 33.5 800 964 974 1063 480 500 523
```

In the section "Screen", make sure to comment out all the rows containing "modes" by adding "#" at the beginning of those lines.

```
Section "Screen"
        Identifier      "Mali Screen"
        Device          "Mali FBDEV"
        Monitor         "Mali Monitor"
        DefaultDepth    16
        SubSection "Display"
                Depth   16
#               Modes   "800x480"
#               Modes   "1680x1050"
        EndSubSection
EndSection
```

6.1.2. LVDS (AcmepointTP070C01)

Both the U-Boot and Xorg need to be modified in order for the LVDS panel to display correctly.

Connection:

Connect the LVDS panel to the VAB-600 through the LVDS connector on the VAB-600-D card.

U-Boot:

Connect the VAB-600 and the host machine through the debug COM1 port. Use the following command to update the U-Boot parameters.

```
WMT # setenv wmt.display.param 2:0:24:800:480:60
WMT # saveenv
WMT # reset
```

800:480 is the resolution of the LVDS panel.

Xorg:

Edit the **/etc/X11/xorg.conf** file, commenting out all the rows containing "Modeline" by adding "#" at the beginning of those lines, except the following "Modeline" below.

```
Modeline "800x480" 33.5 800 964 974 1063 480 490 500 523
```

In the section "Screen", delete "#" before "Modes "800x480"".

```
Section "Screen"
        Identifier      "Mali Screen"
        Device          "Mali FBDEV"
        Monitor         "Mali Monitor"
        DefaultDepth    16
        SubSection "Display"
                Depth   16
                Modes   "800x480"
        EndSubSection
EndSection
```

6.1.3. Dual Channel LVDS (AUO G220SVN01.0)

Both the U-Boot and Xorg need to be modified in order for the LVDS panel to display correctly.

Connection:

Connect the dual channel LVDS panel to the VAB-600 board through the LVDS connector on the VAB-600-D card.

U-Boot:

Connect the VAB-600 board and the host machine through the debug COM1 port. Use the following command to update U-boot parameters.

```
WMT # setenv wmt.display.param 2:0:24:1680:1050:60
WMT # saveenv
WMT # reset
```

1680:1050 is the resolution of the LVDS panel.

Xorg:

Edit the **/etc/X11/xorg.conf** file ,commenting out all the rows containing "Modeline" by adding "#" at the beginning of the lines, except the following "Modeline" below.

```
Modeline "1680x1050"  146.25  1680 1784 1960 2240  1050 1053 1059 1089 -hsync +vsync
```

In the section "Screen", delete "#" before "Modes "800x480" and change "800x480" to "1680x1050".

```
Section "Screen"
        Identifier      "Mali Screen"
        Device          "Mali FBDEV"
        Monitor         "Mali Monitor"
        DefaultDepth    16
        SubSection "Display"
                Depth   16
                Modes   "1680x1050"
        EndSubSection
EndSection
```

6.2. Configuring the EMIO-2550 miniPCIe Mobile Broadband Module

The EMIO-2550 miniPCIe Mobile Broadband module supports 3G and GPS functions.

The first step to enable these functions is to insert the EMIO-2550 module and an active SIM card into the VAB-600. Then connect the VAB-600 to the internet through LAN port, and power it on.

6.2.1. Connecting to the Internet

Type the following command to install the modemmanager package then reboot.

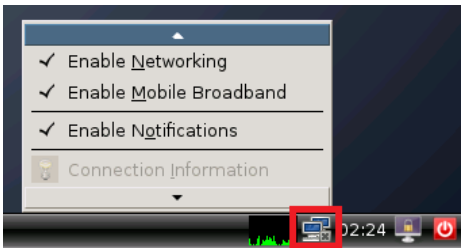
```
$ sudo apt-get install modemmanager
$ sudo reboot
```

After the system has rebooted, use the following command to confirm that the system has correctly detected the EMIO-2550 module.

```
$ lsusb
```

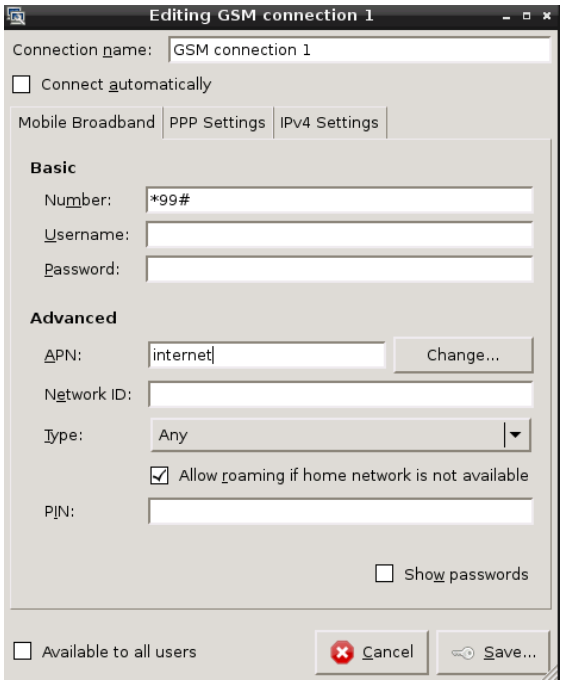
Look for "ID 1545:1102 U-Blox AG" in the ensuing list to ensure the module has been recognized.

To configure the Mobile broadband connection, click on the "Network Manager" icon located in the notification area.



Network Manger configuration

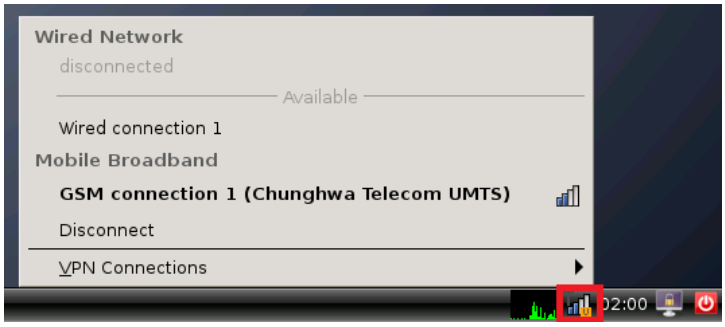
Fill in the required fields to setup your connection. If you are unsure of what the required fields and values are, check with your Mobile Broadband provider.



3G connection configuration

After the connection is created, click on the "Network Manager" icon again and select the connection you just created.

The successful connection looks like this:



Successful connection screen

6.2.2. Enabling GPS

Use a serial port communication program such as Minicom or GtkTerm to configure the GPS function using the serial port setup configuration.

For example, install the Minicom program with the following command, and then open a Minicom terminal window.

```
$ sudo apt-get install minicom
$ sudo minicom -s
```

Select the "Serial port setup" option to configure the serial port, and then set the following settings provided in the screenshot below.

```
+-----------------------------------------------------------------+
|  A    -         Serial Device        : /dev/ttyUSB3             |
|  B    -       Lockfile Location      : /var/lock                |
|  C    -         Callin Program       :                          |
|  D    -        Callout Program       :                          |
|  E    -          Bps/Par/Bits        : 115200 8N1               |
|  F    -       Hardware Flow Control  : No                       |
|  G    -       Software Flow Control  : No                       |
+-----------------------------------------------------------------+
```

Serial port setting of GPS

When the setting is completed, type the following GPS AT commands into Minicom: AT+UGPRF=1, AT+UGRMC=1, AT+UGGSV=1, AT+UGZDA=1, and AT+UGPS=1,0

Wait for the OK message to appear before sending the next AT command.

```
AT+UGPRF=1
OK
AT+UGRMC=1
OK
AT+UGGSV=1
OK
AT+UGZDA=1
OK
AT+UGPS=1,0
OK
```

To ensure the GPS is functioning correctly, open a second Minicom terminal window and set the serial device to **/dev/ttyUSB7**. It will then output the data received from the GPS as illustrated in the screenshot below.

```
$GPRMC,142015,00,A,2501.93961,N,12133.66111,E,0.068,,030314,,,A*73
$GPGSV,4,1,13,01,34,184,29,03,49,023,47,06,26,042,41,07,44,317,26*75
$GPGSV,4,2,13,08,13,323,,11,60,192,,13,29,242,38,16,33,071,16*79
$GPGSV,4,3,13,19,65,356,44,23,18,208,12,27,35,034,27,30,37,145,29*7B
$GPGSV,4,4,13,32,01,154,*4B
$GPZDA,142015.00,03,03,2014,00,00*62
```

GPS NMEA message

6.3. Setting the Watchdog Timer

This section describes how to use Watchdog timer function of the VAB-600.

To enable the Watchdog timer, enter the following command:

```
$ sudo/usr/bin/wdt_app-et
```

To disable the Watchdog timer, enter the following command:

```
$ sudo /usr/bin/wdt_app-dt
```

To set the Watchdog timer, enter the following command:

```
$ sudo /usr/bin/wdt_app-t<second>
```

<second>: Input the desired reboot delay time in seconds between 1 to 255.

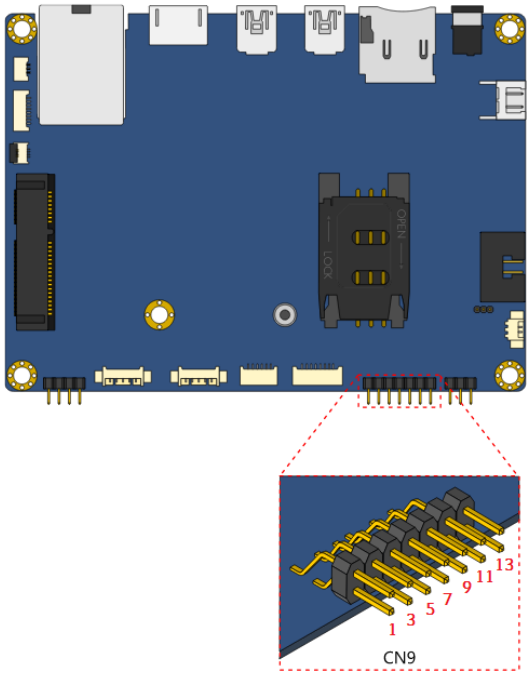
To read the Watchdog countdown timer, enter the following command:

```
$ sudo /usr/bin/wdt_app-tg; echo $?
```

It will show how many seconds are left before the system reboots.

6.4. Programming the GPIO

This section describes how to use the GPIO on the VAB-600. GPIO uses pin 5 to pin 12 of CN9 pin header. The onboard pin header and pinout are shown below.



CN9 pin header diagram

onboard pin #	Signal	onboard pin #	Signal
1	VCC33	2	VCC_5V
3	GND	4	VSUS33
5	GPI20_CH	6	GPIO24_CH
7	GPIO21	8	GPIO25
9	GPIO_22	10	GPIO_26
11	GPIO_23	12	GPIO_27
13	I2C0SDA	14	I2C0SCL

CN9 pin definitions table

Use the devmem2 command to read/write the GPIO registers.

```
$ sudo devmem2 <address> <type> <data>
```

<address> : physical address of the register
<type> : data type to be read/written, [b]yte and [h]alfword(2 byte)
<data> : data to be written to the register

The corresponding addresses and BIT of GPIO pins are listed below:

Enable GPIO function: 1 for enable / 0 for disable				
Address	BIT	Attribute	Signal	Onboard pin
0xd8110070	4:7	RW	GPIO[20:23]	5,7,9,11
0xd8110071	0:3	RW	GPIO[24:27]	6,8,10,12
Set GPIO mode: 1 for output mode / 0 for input mode				
Address	BIT	Attribute	Signal	Onboard pin
0xd81100b0	4:7	RW	GPIO[20:23]	5,7,9,11
0xd81100b1	0:3	RW	GPIO[24:27]	6,8,10,12
GPIO output: 1 for high state / 0 for low state				
Address	BIT	Attribute	Signal	Onboard pin
0xd81100f0	4:7	RW	GPIO[20:23]	5,7,9,11
0xd81100f1	0:3	RW	GPIO[24:27]	6,8,10,12
Read GPIO input: 1 for high state / 0 for low state				
Address	BIT	Attribute	Signal	Onboard pin
0xd8110030	4:7	RW	GPIO[20:23]	5,7,9,11
0xd8110031	0:3	RW	GPIO[24:27]	6,8,10,12
Enable GPIO Pull-up Pull-down function: 1 for enable / 0 for disable				
Address	BIT	Attribute	Signal	Onboard pin
0xd81104b0	4:7	RW	GPIO[20:23]	5,7,9,11
0xd81104b1	0:3	RW	GPIO[24:27]	6,8,10,12
Set GPIO pull up/down: 1 for up / 0 for down				
Address	BIT	Attribute	Signal	Onboard pin
0xd81104f0	4:7	RW	GPIO[20:23]	5,7,9,11
0xd81104f1	0:3	RW	GPIO[24:27]	6,8,10,12

GPIO control register address table

The following example is to enable the GPIO 20 ~ GPIO 23 then set them to output mode and high state.

```
$ sudo devmem2 0xd8110070 b 0xf0
$ sudo devmem2 0xd81100b0 b 0xf0
$ sudo devmem2 0xd81100f0 b 0xf0
```

The following example is to enable and read the status of GPIO 24 ~ GPIO 27 then set them to input mode.

```
$ sudo devmem2 0xd8110071 b 0x0f
$ sudo devmem2 0xd81100b1 b 0x00
$ sudo devmem2 0xd8110031 b
```

6.5. Configuring RTC Wake-Up

RTC wake-up is enabled by the IDT1337G[1] chip to support auto power-on function. The IDT1337G chip is controlled by the I²C0 bus of VAB-600.

The first step is to make sure the RTC time is consistent with the system time. To read the RTC time, use the following command:

```
$ sudo i2cdump -f -y -r 0-6 0 0x68
```

The following example value 03410101050516 means Monday, May 5th 2016 01:41:03.

```
root@arm:~# i2cdump -f -y -r 0-6 0 0x68
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 03 41 01 01 05 05 16                               ?A?????
root@arm:~#
```

RTC time setting sample

The following example has incorrect time value. The minutes in the RTC time is not consistent with the system time.

```
root@arm:~# date
Tue May 10 14:54:02 CDT 2016
root@arm:~# sudo i2cdump -f -y -r 0-6 0 0x68
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 03 45 14 02 10 05 16                               ?E?????
root@arm:~#
```

RTC time setting with incorrect value

Use the following command to correct inconsistent values:

```
$ sudo i2cset -f -y 0 0x68 <address> <Value>
```

The corresponding addresses and values to make them consistent are listed below:

Address	Value	Description
0x0	0x$(date +"%S")	Seconds
0x1	0x$(date +"%M")	Minutes
0x2	0x$(date +"%-k")	Hours
0x3	0x$(date +"%u")	day of week
0x4	0x$(date +"%d")	date of month
0x5	0x$(date +"%m")	Month
0x6	0x$(date +"%g")	Year

RTC time setting table

From the example above, the minutes did not match. To correct this, enter the following command:

```
$ sudo i2cset -f -y 0 0x68 0x1 0x$(date +"%M")
```

For information of setting the RTC function and its activation time, please refer to "Timekeeper Registers" in the IDT1337G's datasheet.

[1] See the IDT 1337G page at http://www.idt.com/products/general-parts/1337g-real-time-clock-i2c-serial-interface

7. SPI ROM Backup and Recovery

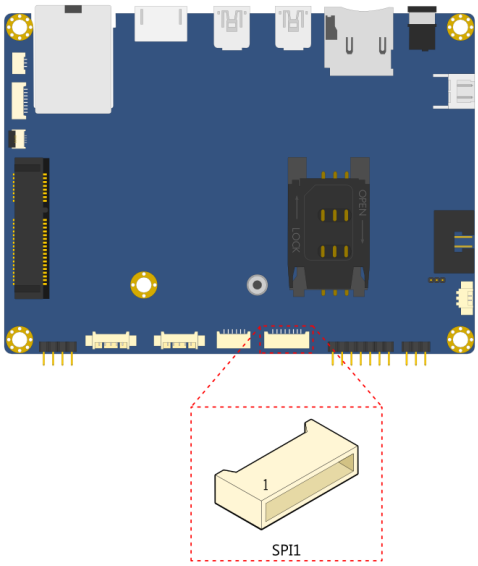
On the VAB-600 board, the SPI Flash ROM contains the U-Boot bootloader and its settings. If the ROM contents become corrupted due to a failure during a firmware update, or are flashed with a non-functional U-Boot binary during development, the board may become non-bootable. In that case, you will need to reflash the SPI ROM with a workable binary image in order to recover the board.

This section explains the SPI ROM backup and recovery procedures for the VAB-600 board.

7.1. Hardware Information

To backup or reflash the SPI ROM, a SPI Flash programmer is required. Dediprog SF100[2] is used as an example in this guide, however any other programmers can also be used as long as they are capable of handling the Mactronix MX25L4005 serial Flash chip on the VAB-600.

You will need to make your own SPI connector cable to connect the Flash programmer to the VAB-600. The SPI header (marked as SPI1) is an ACES[3] 87213-0800G connector, which is compatible with the ACES50233-008H0H0-001 housing and 87214-series cramp-on wafer-to-board terminals. The onboard connector and pinouts are shown below.



SPI1 connector diagram

The pinout matching between the SF100 and the SPI1 header is shown in the table below.

SF100 pin #	SF100 pin name	SPI1 pin #	SPI1 pin name
8	I/O3	1	(not connected)
7	Vpp/Acc	2	SFCS1
6	MOSI	3	SFDO
5	MISO	4	SFDI
4	CLK	5	SFCLK
3	CS	6	SFCS0-
2	GND	7	GND
1	Vcc	8	VPROG_SP1(3.3V)

SFI100 and SPI1 pin definition table

[2] See the Dediprog SF100 product page at http://www.dediprog.com/pd/spi-flash-solution/SF100
[3] See the ACES website at http://www.acesconn.com/

7.2. Software Information

For the backup and reflash procedures you should use the software provided by the vendor of your Flash programmer. When using the Dediprog SF100, download the "dediprog" programmer from their product page. Alternatively, flashrom[4] is a tool that supports a large number of different flash programmers.

7.2.1. Backing up the SPI ROM Data

First, make sure the VAB-600 board is powered off. Connect your Flash programmer and read out the contents of the SPI ROM into a backup file.

For the DediprogSF100, this task can be performed by using the graphical user interface tool in Windows or the following command in Windows or Linux.

```
$ dpcmd -r SF_BACKUP.bin
```

This saves the contents of the Flash chip to the file **SF_BACKUP.bin**.

For other Flash programmers, refer to their documentation to correctly read the chip contents.

7.2.2. Reflashing the SPI ROM

Use either the file you just backed up (refer to the previous section), or the default SPI ROM image that is included in the VAB-600 BSP/EVK. The file name of the default image is **SF_BOOTROM.bin**, which can be found in the **bspinst** folder inside the **VAB-600_Linux_EVK_v1.2.1.zip** file.

First, make sure the VAB-600 board is powered off. Connect your Flash programmer and perform an erase-and-write cycle.

For the Dediprog SF100, this task can be performed by using the graphical user interface tool in Windows, or the following command in Windows or Linux.

```
$ dpcmd -e -p SF_BOOTROM.bin
```

SF_BOOTROM.bin is the Flash ROM image that is used to reflash the ROM (in this case the default image).

For other Flash programmers, refer to their documentation to correctly write the chip contents.

7.2.3. Resetting the MAC Address

If the image you flashed is the default Flash ROM image from VIA, you will need to reset the MAC address in the U-Boot settings (also stored on the SPI ROM), as the default image will overwrite those settings.

Connect the VAB-600 and the host machine through the debug COM1 port. Update the U-Boot parameters as shown below, replacing xx:xx:xx:xx:xx:xx with the correct MAC address of your VAB-600 board.

```
WMT # setenv ethaddr xx:xx:xx:xx:xx:xx
WMT # saveenv
WMT # reset
```

[4] See the flashrom website at https://www.flashrom.org/Flashrom

The MAC address can be found on the sticker located on the top of the Ethernet connector.

Ethernet MAC address sticker

Taiwan Headquarters

1F, 531 Zhong-Zheng Road
Xindian, Taipei, 23148
Taiwan

TEL: 886.2.2218.5452
FAX: 886.2.2218.5453
Email: embedded@via.com.tw

USA

940 Mission Court
Fremont, CA 94539
USA

TEL: 1.510.683.3300
FAX: 1.510.687.4654
Email: embedded@viatech.com

Japan

3-15-7 Ebisu MT Bldg. 6F
Higashi, Shibuya-ku
Tokyo 150-0011
Japan

TEL: 81.3.5466.1637
FAX: 81.3.5466.1638
Email: embedded@viatech.co.jp

China

Tsinghua Science Park Bldg. 7
No. 1 Zongguancun East Road
Haiden District, Beijing, 100084
China

TEL: 86.10.59852288
FAX: 86.10.59852299
Email: embedded@viatech.com.cn

Europe

Email: embedded@via-tech.eu