Release Notes for the QNX Neutrino 6.5.0 BSP for Texas Instruments AM3517 EVM Board#

System requirements#

Target system#

- QNX Neutrino RTOS 6.5.0
- Board version: TI AM3517 EVM (SOM Board: REV D, experimenter: REV B, Application board: REV C)
- ROM Monitor version: Texas Instruments X-Loader 1.44, UBoot 2009.08-dirty
- Micron 512 MB NAND flash: MT29F4G16ABCHC/MT29F4G16ABBDAH4
- Intel 8MB NOR flash: PC28F640P30B85
- 256 MB DDR2 SDRAM

Host development system#

- QNX Momentics 6.5.0
- Terminal emulation program (Qtalk, Momentics IDE Terminal, tip, HyperTerminal, etc.)
- RS-232 serial port
- NULL-modem serial cable
- Ethernet link

System Layout#

The tables below depict the memory layout for the image and for the flash.

Item	Address
OS image loaded at:	0x80100000
NOR flash base address:	0x10000000

The interrupt vector table can be found in the buildfile located at **`src/hardware/startup/boards/am3517evm/build`**

Getting Started#

This BSP includes a prebuilt Out-Of-Box IFS image designed to support development with the Momentics IDE. This is also the default image created by the BSP and includes the following features:

- IPv4 networking support using DHCP to automatically configure the ethernet network interface.
- an LCD splash screen displaying the configured network settings together with Neutrino kernel version and hardware information.
- qconn and telnetd network services to support Momentics IDE remote target access.
  - telnet login is root with no password (please disable this for production images!).
- pdebug program and rcheck, mudflap libraries to support debugging and memory analysis from the Momentics IDE.
- runs an instrumented Neutrino kernel to support Momentics kernel tracing (also includes tracelogger program for tracing from the shell on the TI AM3517 EVM board).

- nfs2 filesystem resource manager to allow mounting network filesystems on the AM3517 EVM to simplify development processes.
- USB mass storage and SDCard drivers together with QNX4 and DOS file system resource managers to support removable storage devices on the AM3517 EVM.
- ftp client to transfer files to/from the AM3517 EVM from the shell prompt.
- a basic set of target shell commands for file system, process, and network management.

Step 1: Connect your hardware#

- Connect the serial cable from the serial port P1(UART3) of the AM3517 EVM experimenter board to the first serial port of your host machine (e.g. ser1 on a Neutrino host).
  - If you have a Neutrino host with a serial mouse, you may have to move the mouse to the second serial port on your host, because some terminal programs require the first serial port.
- Connect a Cat-5 network cable from the RJ45 10/100 network port J31 on the AM3517 EVM experimenter board to a port on an ethernet network that has TCP/IP connectivity to your Momentics IDE workstation, ideally on the same IP subnet.

Step 2: Build the BSP (optional)#

This BSP includes a prebuilt IFS Out-Of-Box image named ifs-am3517evm.raw in the images directory. You can use this image or you can build a BSP OS image from the source code or the binary components contained in the BSP package. For instructions about building a BSP OS image, please refer to the chapter Working with a BSP in the Building Embedded Systems manual.

Step 3: Transfer the OS image to the target using the ROM monitor#

On your host machine, start your favorite terminal program with these settings:

- Baud: 115200
- Bits: 8
- Stop bits: 1
- Parity: none
- Flow control: none

Setting up the environment#

Apply power to the target board. You should see output on your terminal console, similar to the following:

```
Texas Instruments X-Loader 1.44 (Dec  8 2009 - 22:58:46)
Starting OS Bootloader...



U-Boot 2009.08-dirty (Jan 05 2010 - 10:21:46)

AM35xx-GP ES1.0, L3-165MHz
am3517evm board + LPDDR/NAND
I2C:   ready
DRAM:  256 MB
NAND:  512 MiB
In:    serial
Out:   serial
Err:   serial
HECC U20: port before = 00000040
HECC U20: programmed CAN_STB low
HECC U20: port after = 00000000
```

```
Die ID #2f280000000000000154417b0201300a
Net:   davinci_emac_initialize
Ethernet PHY: GENERIC @ 0x00
DaVinci EMAC
Hit any key to stop autoboot:  0
```

Use the **`printenv`** command to show the current settings for **`serverip, gatewayip, netmask, bootfile, ipaddr, ethaddr`** etc.

At this point you can load the image to the AM3517 EVM either over the network with TFTP or via an SD/MMC memory card. The U-Boot bootloader on the AM3517 EVM can then be configured to automatically boot an IFS image using the chosen technique.

SD card download#

Copy the raw OS image **`ifs-am3517evm.raw`** onto a mmcsd card, then insert the SD card in J14 MMCSD1 port and apply power to the target board. After U-Boot is started, download the **`ifs-am3517evm.raw`** image in the mmcsd card as follows from the AM3517EVM # prompt:

```
mmc init or mmcinit (depends on which version of uboot you are using)
fatload mmc 0 0x80100000 ifs-am3517evm.raw
go 80100000
```

At this point you should see the U-Boot download the boot image, indicated by a series of number signs. You'll also see output similar to this when it completes downloading:

```
AM3517_EVM # mmc init
mmc1 is available
AM3517_EVM # fatload mmc 0 0x80100000 ifs-am3517evm.raw
reading ifs-am3517evm.raw

3421956 bytes read
AM3517_EVM # go 80100000
## Starting application at 0x80100000 ...
CPU0: L1 Icache: 256x64
CPU0: L1 Dcache: 256x64 WB
CPU0: L2 Dcache: 4096x64 WB
CPU0: VFP 410330c1
CPU0: 411fc087: Cortex A8 rev 7 500MHz
Loading IFS...decompressing...done

System page at phys:80011000 user:fc404000 kern:fc404000
Starting next program at vfe04d604
cpu_startnext: cpu0 -> fe04d604
VFPv3: fpsid=410330c1
coproc_attach(10): replacing fe07d01c with fe07c8bc
coproc_attach(11): replacing fe07d01c with fe07c8bc
Welcome to QNX Neutrino 6.5.0 on the Texas Instruments AM3517 EVM (ARMv7 Cortex-A8 core)
```

Now you can test the QNX OS, simply by executing any shell builtin command or any command residing within the OS image (e.g. ls).

TFTP download#

This method requires a raw image, which is simply a binary image, with a header on the beginning, that allows the bootloader to jump to the very beginning of the image (the raw header), where it executes another jump to the first instruction of the image.

Use the **`setenv`** command to configure the following parameters: **`serverip gatewayip netmask bootfile ipaddr ethaddr`**

Once these parameters are configured, use the **`saveenv`** command to store your changes and then use the **`reset`** command to reset the B-Boot. Refer to the U-Boot documentation for more information.

After U-Boot is configured, download the **`ifs-am3517evm.raw`** image as follows from the AM3517EVM # prompt (we'll assume it's in a directory called /qnx/ exported by the TFTP server running at IP address $serverip):

```
tftpboot 0x80100000 /qnx/ifs-am3517evm.raw
```

At this point you should see the U-Boot download the OS image, indicated by a series of number signs. You'll also see output similar to this when it completes downloading:

```
AM3517_EVM # tftpboot 0x80100000 /qnx/ifs-am3517evm.raw
Using DaVinci EMAC device
TFTP from server 10.42.104.16; our IP address is 10.42.104.253
Filename '/qnx/ifs-am3517evm.raw'.
Load address: 0x80100000
Loading: ##################################################################
	#################################################################
	#################################################################
	#################################################################
	#################################################################
	#################################################################
	#################################################################
	#################################################################
	#################################################################
	#################################################################
	###################
done
Bytes transferred = 3421956 (343704 hex)
AM3517_EVM #
```

Now, to run the image, enter:

```
go 0x80100000
```

You should see output similar to the following, with the QNX Neutrino welcome message on your terminal screen:

```
AM3517_EVM # go 0x80100000
## Starting application at 0x80100000 ...
CPU0: L1 Icache: 256x64
CPU0: L1 Dcache: 256x64 WB
CPU0: L2 Dcache: 4096x64 WB
CPU0: VFP 410330c1
CPU0: 411fc087: Cortex A8 rev 7 500MHz
Loading IFS...decompressing...done

System page at phys:80011000 user:fc404000 kern:fc404000
Starting next program at vfe04d604
cpu_startnext: cpu0 -> fe04d604
VFPv3: fpsid=410330c1
coproc_attach(10): replacing fe07d01c with fe07c8bc
coproc_attach(11): replacing fe07d01c with fe07c8bc
Welcome to QNX Neutrino 6.5.0 on the Texas Instruments AM3517 EVM (ARMv7 Cortex-A8 core)
```

You can now test the QNX OS simply by executing any shell builtin command or any command residing within the OS image (e.g. ls).

Getting U-Boot to automatically boot your image#

Read the U-Boot documentation for full details on loading and starting images but the basic technique is to put the above commands into the bootcmd environment variable. It's generally a good idea to save the previous bootcmd value should you need to use it again. The following commands do this:

```
setenv defbootcmd $bootcmd
setenv qnxsdboot 'mmc init; fatload mmc 0 0x80100000 ifs-am3517evm.raw; go 80100000'
setenv qnxtftpboot 'tftpboot 0x80100000 /qnx/ifs-am3517evm.raw;  go 80100000'
setenv bootcmd 'run qnxsdboot'
saveenv
```

Change the **`mmc init`** to **`mmcinit`** if required by your U-Boot version. This will setup three environment variables for booting:

- **`defbootcmd`** is the original AM3517 EVM boot command intended to boot Linux.
- **`qnxsdboot`** will boot an ifs-am3517evm.raw image from an SD card.
- **`qnxtftpboot`** will boot an /qnx/ifs-am3517evm.raw image over the network using TFTP.

and sets the default autoboot command to execute the QNX SD card boot sequence. By changing the **`run qnxsdboot`** bootcmd setting to use one of the other variables and using **`saveenv`** to save it you can select any of the three options to autoboot. For TFTP, the **`serverip, gatewayip, netmask, bootfile, ipaddr, ethaddr`** variables must also be set in the environment for static network configuration. Finally, make sure the **`bootdelay`** variable is set to a reasonable value. This sets the time U-Boot will wait for keyboard input before starting autoboot. Generally 2-4 seconds works well.

Flashing the IPL on to the target#

Step A: Create the IPL image#

Run 'make ipl' inside the /images directory of the BSP to run the mkflashimage script. The output file from this script is a binary IPL image called **`ipl-am3517evm.bin`** and **`nand-ipl-am3517evm.bin`**.

The **`ipl-am3517evm.bin`** file is used by serial download method through UART3 to place the IPL into the board's memory and will not be used to program on NAND flash.

The NAND IPL image **`nand-ipl-am3517evm.bin`** will be installed into the board's NAND flash by using u-boot or IPL. This IPL is padded to 24K.

The following steps describe how to generate the ipl-am3517evm.bin and nand-ipl-am3517evm.bin files:

- generate a bin format OS image called ifs-am3517evm.raw using the am3517evm.build file
- Convert IPL NAND boot header into Binary format
- Convert IPL into Binary format
- Cat NAND boot header and ipl together
- Pad Binary IPL to 24K image, this is the image used by boot from UART
- Pad Binary IPL with Header to 24K image, this is the image to put on NAND and boot from NAND

Here is the mkflashimage script:

#!/bin/sh
 # script to build a binary IPL and boot image for the TI AM3517 EVM board
set -v
 # Convert IPL header into Binary format

```
${QNX_HOST}/usr/bin/ntoarmv7-objcopy --input-format=elf32-littlearm --output-format=binary ../src/hardware/ipl/boa

# Convert IPL into Binary format
${QNX_HOST}/usr/bin/ntoarmv7-objcopy --input-format=elf32-littlearm --output-format=binary ../src/hardware/ipl/boa

# Cat boot header and ipl together
cat ./tmp-boot-header.bin ./tmp-ipl-am3517evm.bin > ./tmp-header-ipl-am3517evm.bin

# Pad Binary IPL to 24K image, this is the image used by boot from UART
mkrec -s24k -ffull -r ./tmp-ipl-am3517evm.bin > ./ipl-am3517evm.bin

# Pad Binary IPL with Header to 24K image, this is the image to put on NAND and boot from NAND
mkrec -s24k -ffull -r ./tmp-header-ipl-am3517evm.bin > ./nand-ipl-am3517evm.bin

# clean up temporary files
rm -f tmp*.bin

echo "done!!!!!!!"
```

Step B: Install NAND IPL image and IFS#

Refer to the AM/OMAP Boot Resource Pages for options for installing the IPL to flash.

Creating a new flash partition#

- Enter the following command to start the flash filesystem driver: `devf-generic -s0x10000000,8m`
- unlock the raw partition: `flashctl -p/dev/fs0 -o1M -l7M -U`
- Erase a section of the flash: `flashctl -p/dev/fs0 -o1M -l7M -ve`
- Format the new patition filesystem: `flashctl -p/dev/fs0p0 -o1M -l7M -vf`
- Slay the devf-generic driver: `slay devf-generic`
- Restart the driver to mount the new partition `devf-generic -s0x10000000,8M`

You should now have a /fs0p1 directory which you can copy files to.

Note: The 8MB Intel PC28F640P30B85 NOR flash is on the Application board. Please make sure the Application board has been connected properly before starting the NOR flash driver,

Summary of driver commands#

The following tables summarize the commands to launch the various drivers.

Component	Buildfile Command	Required Binaries	Required Libraries	Source Location
Startup	`startup-am3517evm`	.	.	`src/hardware/startup/boards/am3517evm`
Serial	`devc-seromap -e -F -b115200`	devc-seromap	.	`src/hardware/devc/seromap`

	-c48000000/16 0x49020000^2,74			
Ethernet	io-pkt-v4 -dam35xx	io-pkt-v4 ifconfig nicinfo ping	libsocket.so devnp-am35xx.so	src/hardware/devnp/am35xx
I2C	i2c-omap35xx For I2C1: i2c-omap35xx -p0x48070000 -i56 --u0 For I2C2: i2c-omap35xx -p0x48072000 -i57 --u1 For I2C3: i2c-omap35xx -p0x48060000 -i61 --u2	i2c-omap35xx	.	src/hardware/i2c/omap35xx
SPI	spi-master -d omap3530 base=0x48098000,bitrate=125000,clock=48000000,irq=65,force=1,num	spi-master	spi-omap3530.so	src/hardware/spi/omap3530
USB OTG Host	io-usb -dam3517-mg ioport=0x5c040400,irq=71	io-usb usb*	libusbdi.so devu-am3517-mg.so	*prebuilt only*
USB EHCI Host	io-usb -dehci-omap3 ioport=0x48064800,irq=77	io-usb usb*	devu-ehci-omap3.so libusbdi.so	*prebuilt only*
CAN	dev-can-am3517 can1	dev-can-am3517 canctl	.	src/hardware/cani/am3517
SD card	For MMCSD1: devb-mmcsd-am3517 cam quiet blk cache=2m mmcsd ioport=0x4809C000,ioport=0x48056000,irq=83,dma=30,dma=61,dma=62 For MMCSD2: devb-mmcsd-am3517 cam quiet blk cache=2m mmcsd ioport=0x480b4000,ioport=0x48056000,irq=86,dma=23,dma=47,dma=48	devb-mmcsd	libcam.so fs-dos.so cam-disk.so	src/hardware/devb/mmcsd
Graphics	io-display -dvid=0,did=0	am3517-evm.conf io-display egl-gears-lite vsync	libGLES_CL.so libffb.so libgf.so devg-omap35xx.so devg-soft3d-fixed.so libdisputil.so	src/hardware/devg/omap35xx
SGX Graphics Accesserator driver	pvrsrvd	pvrsrvd graphics.conf gles1-egl-gears	libsrv_um.so libglslcompiler.so libIMGegl.so libImgGLESv1_CM.so libImgGLESv2.so libImgOpenVG.so	*prebuilt only*

			libpvr2d.so libsrv_um.so libWFDdevg.so pvrsrvinit.so wsegl-gf.so libEGLdevg.so libWFDdevg.so libiow.so.1 libGLESv1_CM.so.1 libEGL.so.1	
NOR flash	`devf-generic -s0x10000000,8m`	devf-generic flashctl	.	`src/hardware/flash/boards/generic`
NAND flash	`fs-etfs-omap3530_micron -r65536 -m /fs/etfs`	fs-etfs-omap3530_micron etfsctl	.	`src/hardware/etfs/nand2048/omap3530_micron`
RTC	`rtc hw /dev/i2c0`	rtc date	.	`src/utils/r/rtc`
DMA Manager	`resource_seed dma=0,31`	resource_seed	.	`src/utils/r/resource_seed`

Some of the drivers are commented out in the default buildfile. To use the drivers in the target hardware, you'll need to uncomment them in your buildfile, rebuild the image, and load the image into the board.

SD card #

Command:

```
## MMCSD 1
devb-mmcsd-am3517 cam quiet blk cache=2m mmcsd ioport=0x4809C000,ioport=0x48056000,irq=83,dma=30,dma=61,dma=6

## MMCSD 2
devb-mmcsd-am3517 cam quiet blk cache=2m mmcsd ioport=0x480b4000,ioport=0x48056000,irq=86,dma=23,dma=47,dma=48
```

Graphics#

Command:

```
io-display  -dvid=0,did=0
```

Ethernet#

Command:

```
io-pkt-v4 -dam35xx
waitfor /dev/socket
ifconfig am0 xx.xx.xx.xx
# or
dhcp.client
```

USB OTG Host Controller driver#

Command:

```
io-usb -dam3517-mg ioport=0x5c040400,irq=71 -dehci-omap3 ioport=0x48064800,irq=77,verbose=5
```

Note:

1. Min A to min B usb cable is required.

2. Connect to the micro usb adapter.

3. Start io-usb with the stock options from the bsp build file.

SGX Graphics Accelerator driver#

Command:

```
GRAPHICS_ROOT=/usr/lib/graphics/am3517
pvrsrvd
```

Note:

1. **`io-display`** must be run first

2. the GRAPHICS_ROOT enviornment has to been set properly before running **`pvrsrvd`**

3. This BSP only include the necessary prebuild binaries and libraries for the SGX demo. The Composition_Manager patch(patch-650-2258-CompMgr.tar) need be installed first for the full SGX support.

- You can get the composition manager patch: patch-650-2258-CompMgr.tar here.
- To install the patch-650-2258-CompMgr.tar in the AM3517 board BSP's prebuilt directory:

```
# cd am3517_workdir
# tar -xvf patch-650-2258-CompMgr.tar
# cp -vr patches/650-2258/target/qnx6/armle-v7 prebuilt/
# make clean all
```

4. For the detailed documents about the composition_manager and SGX Graphics Accelerator, you can refer to

- composition_manager release notes
- SGX release notes

General Notes#

1. The ethernet driver generates two warnings in the kernel logs. These are normal warnings and can be ignored.

- Unable to attach to pci server: No such file or directory
- Using pseudo random generator. See "random" option

Known issues for this BSP#

1. Only the mainboard USB host port is supported, and can only be controlled by the EHCI controller. Due to a SoC limitation, the OHCI controller cannot be used as a companion controller on this port. As a result, a high speed hub must be used to connect full/low speed devices.
2. The host port on on the Application Board is not yet supported.
3. The MMCSD2 interface has a hardware conflict with the LCD device. You can not run graphic driver when use the MMCSD2.

4. RTC: There is no external battery for RTC chip by default. The RTC data would lose once the system power off.
5. Some SD cards are not detected properly in the MMC2 interface of the application board (ref #75227).
6. A log error is issued from the mentor graphics USB driver saying "Unknown option irq=71". This is a bogus error and can be ignored (the IRQ is working properly). (ref #75292)
7. USB-OTG: Some SanDisk Cruzer devices do not operate reliably on the OTG interface (ref #76047).
8. USB-OTG: A D-Link DUB E-100 USB to Ethernet adapter did not operate reliably on the OTG interface (ref #76107). Use the EHCI interface when possible.
9. Ethernet: Very rarely the ethernet driver fails to discover the PHY. sloginfo will show "PHY not found" (ref #76310). The only workaround at this time is to slay and re-start the driver.
10. CAN: The HECC bus may not work reliably when the transaction bitrate has been set to less that 20K/S. (ref #77624)
11. SGX: pvrsrvd driver would have a SIGSEGV error when you slay it. (ref #76821)