Release Notes for the QNX Neutrino 6.4.0 BSP for Atmel At91sam9263 EK trunk#

System requirements#

Target system#

- QNX Neutrino RTOS 6.4.0
- Board version: Atmel At91sam9263 EK
- ROM Monitor version UBoot 1.1.5
- DataFlash:AT45DB642 (8MB)

Host development system#

- QNX Momentics 6.4.0
- 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:	0x20100000
Flash base address	0xC0000000

The interrupt vector table can be found in the buildfile located at `src/hardware/startup/boards/at91sam9263ek/build`

Getting Started#

Step 1: Connect your hardware#

1. Connect the serial cable to the terminal debug port of the At91sam9263 EK board and to the first serial port on the host machine (e.g. ser1 on a Neutrino host).

Note: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.

2. Connect an RJ-45 ethernet cable between the ethernet port on the At91sam9263 EK and your local network.

Step 2: Build the BSP#

You can build a BSP OS image from the source code or the binary components contained in a 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 U-Boot#

Note: If you don't want to boot your target with U-Boot and you prefer to replace your U-Boot on the board with a native QNX IPL, you can skip this step and go to Step 4.

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

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

2. Start your target. You should see output similar to the following:

```
RomBOOT
>
U-Boot 1.1.5 (Dec 22 2006 - 10:43:42)

DRAM:  64 MB
NAND:  NAND device: Manufacturer ID: 0xec, Chip ID: 0xda (NAND 256MiB 3,3V 8-bit)
256 MiB
DataFlash:AT45DB642
Nb pages:   8192
Page Size:   1056
Size= 8650752 bytes
Logical address: 0xC0000000
Area 0: C0000000 to C0003FFF (RO)
Area 1: C0004000 to C0007FFF
Area 2: C0008000 to C0037FFF (RO)
Area 3: C0038000 to C083FFFF
In:    serial
Out:   serial
Err:   serial
DM9161A PHY Detected
End of Autonegociation
Hit any key to stop autoboot:  0
U-Boot>
```

Note: For booting from U-Boot, the image ipl-ifs-at91sam9263ek.bin must be built as raw, i.e. **`virtual=armle,raw`** in the buildfile 3. Setting up the environment

This method requires a raw image, which the buildfile creates by default.

On your target, type the following, filling in the appropriate IP addresses and `ifs` file:

```
=> setenv ipaddr 192.168.200.2
=> setenv serverip 192.168.200.1
=> setenv bootfile ifs-at91sam9263ek.bin
=> setenv loadaddr 0x20100000
=> setenv bootcmd 'tftpboot $loadaddr $bootfile; go $loadaddr'
=> setenv bootdelay 2
=> saveenv
```

4. Restart your target, You should see output similar to the following:

```
Hit any key to stop autoboot:  0
TFTP from server 10.42.107.173; our IP address is 10.42.105.249
```

```
Filename 'at9263.bin'.
Load address: 0x20100000
Loading:  #################################################################
      #################################################################
      #################################################################
      #################################################################
      ##################
done
Bytes transferred = 1423692 (15b94c hex)
## Starting application at 0x20100000 ...
Dcache: 512x32 WB
Icache: 512x32
arm926 rev 5 200MHz
Header size=0x0000009c, Total Size=0x00000490, #Cpu=1, Type=4
Section:system_private offset:0x000001d8 size:0x00000068
Section:qtime offset:0x00000148 size:0x00000048
Section:callout offset:0x000000a0 size:0x00000048
Section:cpuinfo offset:0x00000190 size:0x00000020
Section:cacheattr offset:0x00000450 size:0x00000040
Section:meminfo offset:0x00000490 size:0x00000000
Section:asinfo offset:0x00000310 size:0x00000100
Section:hwinfo offset:0x000002c8 size:0x00000048
Section:typed_strings offset:0x00000240 size:0x00000030
Section:strings offset:0x00000270 size:0x00000058
Section:intrinfo offset:0x00000410 size:0x00000040
Section:smp offset:0x00000490 size:0x00000000
Section:pminfo offset:0x00000490 size:0x00000000
Section:mdriver offset:0x00000490 size:0x00000000
Section:boxinfo offset:0x000001b0 size:0x00000028
Section:cpu offset:0x00000128 size:0x00000020

System page at phys:20014000 user:fc404000 kern:fc404000
Starting next program at vfe0203a4
Welcome to QNX Neutrino 6.x on the Atmel AT91SAM9263-EK Board
Starting on-board ethernet with TCP/IP stack...
#
```

Step 4: Replace the U-Boot with a native QNX IPL and OS image in flash#

The Atmel AT91SAM9263 EK (Evaluation Kit) Development board supports booting from a native QNX bootable IPL/OS image in flash.

1. Generate a bootable flash image Run the mkflashimage script, inside the /images directory of BSP. The output file from this script is a combined IPL/OS image called ipl-ifs-at91sam9263ek.bin. You need to write this image into a supported Atmel DataFlash Card using the SAM-BA programming tool provided by Atmel. The IPL is padded to 16 KB and will scan for an image at an offset of 16 KB (independently of the DataFlash page mode).

The mkflashimage script:

```
#!/bin/sh
#Script to build a binary IPL and boot image for ATMEL AT91SAM9263 Evaluation Kit board
set -v

#Convert IPL into BINARY format
${QNX_HOST}/usr/bin/ntoarm-objcopy --input-format=elf32-littlearm --output-format=binary -R.data ../install/armle

#Pad BINARY IPL
mkrec -s16k -ffull -r ipl-tmp-at91sam9263ek.bin > ipl-at91sam9263ek.bin
```

```
#Combine the BINARY IPL with the BINARY OS Image
cat ./ipl-at91sam9263ek.bin ./ifs-at91sam9263ek.bin > ipl-ifs-at91sam9263ek.bin

#Cleaning up temporary files
rm -f *tmp*
```

Note: For booting from IPL, the image ipl-ifs-at91sam9263ek.bin must be built as binary, i.e. **`virtual=armle,binary`** in the buildfile

2. Download the bootable image in flash

1. In a Windows system, install the SAM-BA application provided by Atmel to allow you to program the IPL and IFS on a DataFlash card. 2. Connect the DBGU UART of the board with a COM port of the Windows host computer with a serial cable. Remove the DataFlash card and reset the board. Once you have the RomBOOT prompt, you can insert the DataFlash card. Disconnect any terminal application so SAM-BA can take control of serial port if needed. 3. From the Atmel SAM-BA application, select the proper communication mode (serial or usb) and set the board to AT91SAM9263-EK. Press OK to move to the next screen. 4. From the main application window, select the DataFlash AT45DB/DCB tab located in the middle section of the screen. Execute the Enable ~Dataflash on CS0 script. Select the ipl-ifs-at91sam9263ek.bin file to be sent to the target, and then press Send File. Depending on the size of your file, it will take a few seconds up to a minute. 5. Close the SAM-BA application, reconnect your terminal and reset the board. Now it should boot from the native QNX IPL. You should see output as follows:

```
Dataflah info:
Name       : AT45DB642D
ID         : 0x00000028
Density(MB) : 0x00840000
Num Pages   : 0x00002000
Page Size   : 0x00000420
Page Shift  : 0x0000000B
Status      : 0x00000028

QNX Neutrino Initial Program Loader for ATMEL AT91SAM9263-EK board

Commands:
Press D for serial download, using the sendnto utility
Press F to boot an OS image in flash
```

6. Enter f or F, and the board will boot from the OS image in flash. You'll see output similar to the following:

```
Scanning Dataflash...
found image, calling image setup...
image_setup OK, calling image start...

Welcome to QNX Neutrino trunk on the Atmel AT91SAM9263-EK Board ...
```

7. Enter d or D. You can download the new OS image by using the command:

```
sendnto -d/dev/ser1 -b115200  ifs-at91sam9263ek.bin
```

Step 5: Start working with Neutrino OS#

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

Once the initial image is running, you can update the OS image using the network and flash drivers. For sample command lines

Driver Command Summary#

The driver command lines below are specific to the Atmel At91sam9263 EKboard. See the online docs for each driver for additional command-line options and other details.

NOTE: Some of these 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.

Component	Buildfile Command	Required Binaries	Required Libraries	Source Location
Startup	`startup-at91sam9263ek`	.	.	`src/hardware/startup/boards/at91sam9263ek`
Serial	`devc-seratdbgu -e -F -S -b115200 -c100000000 0xffffee00^2,1`	devc-seratdbgu	.	`src/hardware/devc/seratdbgu`
SPI master	`spi-master -u0 -d at91sam base=0xfffa4000,irq=14 & spi-master -u1 -d at91sam base=0xfffa8000,irq=15 &`	.	spi-at91sam.so	`src/hardware/spi/at91sam`
SPI slave	`devc-at91samspis devc-at91samspis 0xfffa4000,14`	devc-at91samspis	.	`src/hardware/spi/at91samspis`
Network	`io-pkt-v4 -dat91sam ioport=0xfffbc000,irq=21,rmii,mac=0x200021615,verbose -ptcpip -v &`	io-pkt-v4 ifconfig devc-sam2x.so	devn-at91sam.so libsocket.so	src/hardware/devn/at91sam}
I2C	`i2c-at91sam`	i2c-at91sam	.	`src/hardware/i2c/ at91sam`
USB	`io-usb -d ohci ioport=0x00a00000,irq=29`	io-usb usb	devu-ohci.so libusbdi.so	QNX SPD 6.4.x (Binary Only)
SD/MDI	`Start the driver on the first socket: devb-sd-at91sam9263 Start the driver on the second socket: devb-sd-at91sam9263 eide ioport=0xfff84000,irq=11`	devb-sd-at91sam9263	libcam.so io-blk.so cam-disk.so fs-qnx4.so	`src/hardware/devb/sd/arm/at91sam9263.le`

Audio	`io-audio -d at91sam9263_ac97`	io-audio mix_ctl wave waverec	libasound.so deva-mixer-ac97.so deva-util-restore.so deva-ctrl-at91sam9263_ac97.so	`src/hardware/deva/ctrl/at91sam9263_ac97`
Graphics	`io-display -dvid=0,did=0`	io-display	devg-at91sam926x.so libgf.so.1 libGLES_CL.so.1 libffb.so.2\libm.so.2	`src/hardware/devg/at91sam926x`

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.

Note: Network driver will not work in half-duplex mode when forcing speed and duplex mode. When driver is configured in Auto-negotiation mode, half-duplex will work properly.

Additional Notes for Graphics:#

1. LCD Displays

The driver defaults to setup the Hitachi TX09D71VM1CCA TFT with AT91SAM9263 reference board. The driver configuration file (at91sam926x.conf) can be used to setup other displays.

2. Supported Color formats

HW FORMAT	QNX FORMAT	NOTES	
16-bit	15-bit	QNX framework format is ARGB1555: but MSB is not actually used due to HW only using 15bits	
24-bit	32-bit	actually un-packed 24bit meaning upper 8 bits are not used.	
24-bit	24-bit	packed 24-bit is not supported by the 2DGC	

3. Color Swapping

The LCD controller for the AT91SAM9263 expects colors to be in the format of BGR. The QNX graphics framework and the 2DGC specify the colorformat as RGB. The graphics driver currently performs the color swapping before programming the 2DGC and provides the entry points for the SW rendering to be done in the correct format.

4. Reserving Memory / Memory Restrictions

The Atmel AT91SMA9263 is a UMA system (Unified Memory Architecture).This means there is no dedicated video memory in the system. Surfacesdisplayed by the LCD controller, and rendered to either by the 2DGC or CPU reside in system memory. The Atmel 2DGC has a restriction that the base address of a memory surface be aligned on a 1 MB boundary. From that 1 MB boundary 2048 lines can be addressed. There is also a restriction that sizes that the stride of memory can be. (256, 512, 1024, 2048) With the reference platform, the display is 240 pixels wide. This means the following amounts of memory can be accessed by the 2DGC based on pixel format:

16-bit (stride = 512) 512 * 2048 = 1048576 bytes 32-bit (stride = 1024) 1024 * 2048 = 2097152 bytes

To ensure there is enough memory available for graphics, it is recommened that memory be reserved at startup by using the -r option to startup. For example to reserve 2MB of memory:

startup-at91sam9263 -r0x23E00000,0x200000 -vv

where: 0x23E00000 is the physical base address of memory 0x200000 is the size of memory reserved in bytes

When the graphics driver is started this physical address and size of memory is passed to the devg-at91sam926x.so through the following options in the driver configuration file, at91sam926x.conf:

vidbase=0x23E00000,vidsize=0x200000

For a complete list of options available to the driver please see the at91sam926x.conf file.

5. 2DGC Issues

1. The QNX graphics architecture dictates that the graphics driver perform clipping when rendering lines. We've found through experimentation that when enabling HW clipping of lines that system lockups can occur under high loads. To recover a hard reset of the system is required. To work around this issue in at91sam926x.conf specify 'noline=1' to not use the at91sam926x 2DGC to render lines.

2. Polygon Fill is not implememnted due to HW errrata filled polygons are not accelerated using the 2DGC

3. Command queue is not used by this driver because not all 2D registers are accessible using the queue.

Known Issues for this BSP#

- Network driver will not work in half-duplex mode when forcing speed and duplex mode. When driver is configured in Auto-negotiation mode, half-duplex will work properly.
- Network driver drops packet due to transmitter under run. A known HW limitation with the Ethernet transmitter and usage of slow SDRAM memory will cause the transmitter to generate under run and flush all packets/fragments in the transmitter queue. For more details about this problem refer to the Atmel AT91SAM9263 reference manual (doc6249.pdf) section 49.2.6 “EMACB Errata” for more details.