Release Notes for the QNX Neutrino 6.4.0 BSP for Atmel AT91SAM9G45-EK Board#

System requirements#

Target system#

- QNX Neutrino RTOS 6.4.0
- Board: Atmel AT91SAM9G45-EK Evaluation Board
- Data-Flash: AT45DCB008D (4 MB) CARD
- DDR2 SDRAM: 16-bit Micron MT47H64M8CF (64 MB) SDRAM
- NAND Flash: Micron 29F2G08ABD (256 MB) NAND Flash

Host development system#

- QNX Momentics 6.4.0, one of the following host systems:
  - QNX Neutrino 6.4.0
  - Microsoft Windows Vista, XP SP2 or SP3, 2000 SP4
  - Linux Red Hat 8 or 9, Linux Red Hat Enterprise Workstation 3 or 4, Red Hat Fedora Core 3 or 4, or SUSE 10
- Terminal emulation program (Qtalk, Momentics IDE Terminal, tip, HyperTerminal, etc.)
- Windows Machine required SAM-BA appication to download the image on Data-Flash.
- RS-232 serial port
- NULL-modem serial cable
- USB cable to connect board with windows Machine

Getting Started#

Step 1: Connect your hardware#

Connect the DEBUG port of the AT91SAM9G45-EK board to the first serial port of your windows machine. Install the SAM-BA application provided by Atmel. Connect the board with Windows Machine using USB Cable.

Step 2: Build the BSP#

You can build an OS image from the source code or the binary components contained in a BSP package. For instructions about building an OS image, please refer to the chapter Working with a BSP in the Building Embedded Systems manual. After Building the BSP three bin files will be created in images directory.
*ifs-at91sam9g45.bin
*ipl-at91sam9g45.bin
*ipl-ifs-at91sam9g45.bin

```
mkflashimage script creates a combined IPL/IFS image as ipl-ifs-at91sam9g45.bin
The mkflashimage script:

#!/bin/sh
# script to build a binary IPL and boot image for ATMEL AT91SAM9G45 Evaluation Kit board.
# NOTE the image (ipl-ifs-at91sam9g45.bin) must be built as binary, i.e. [virtual=armle,binary] in the buildfile
set -v

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

```
#       Pad BINARY IPL
mkrec -s16k -ffull -r ipl-tmp-at91sam9g45.bin > ipl-at91sam9g45.bin

#       Combine the BINARY IPL with the BINARY OS Image
cat ./ipl-at91sam9g45.bin ./ifs-at91sam9g45.bin > ipl-ifs-at91sam9g45.bin

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

Step 3A: Download the Bootable IFS image.#

The Boot Program integrates different programs that manage download and/or upload into the different memories of the product. First, it initializes the Debug Unit serial port (DBGU) and the USB High Speed Device Port.

- NAND Flash Boot program is then executed. The NAND Flash Boot program searches for a valid application in the NAND Flash memory. If a valid application is found, this application is loaded into internal SRAM and executed by branching at address 0x0000_0000 after remap.
- If no valid ARM vector sequence is found, the Data-Flash Boot program is then executed. It looks for a sequence of seven valid ARM exception vectors in a Data-Flash connected to the SPI. All these vectors must be B-branch or LDR load register instructions except for the sixth vector. This vector is used to store the size of the image to download. If a valid sequence is found, code is downloaded into the internal SRAM. This is followed by a remap and a jump to the first address of the SRAM. If no valid ARM vector sequence is found, SAM-BA Boot is then executed. It waits for transactions either on the USB device, or on the DBGU serial port.

Install and setup SAM-BA#

1. Install "Install AT91-ISP v1.13.exe" .

2. Intall "ActiveTcl8.5.5.0.287690-win32-ix86-threaded.exe" i.e.TCL environment which is used by SAM-BA (any other TCL environment can also be used) .

3. Connect serial cable with windows machine, and use any serial port application such as teraterm or hyperterminal, and attach it with COM device, with baud rate set as 115200.

4. Restart the board and connect usb cable with windows machine. It will prompt with a new usb hardware found message, and will try to install the corresponding driver.

5. In case, a new usb hardware is not found. The followings sequence of operations should be done to fix ROMBoot on AT91SAM9G45-EK board:

5.1. Unplug power supply and unplug usb device cable from the board.

5.2. Remove jumper JP10 (NandFlash Chip Select) & JP12 (DataFlash Chip Select).

5.3. Plug serial cable, launch hyper terminal (115200 bauds, 8 bits, parity none, 1 stop bit, no flow control and then, plug power supply.

5.4. Type on hyper terminal : "Alt-0128 Alt-0128 #" and AT91SAM9G45-EK returns ">" .

5.5. Close hyper terminal and close jumper JP12 (DataFlash Chip Select).

5.6. Launch SAM-BA (Choose right COM port and AT91SAM9G45-EK).

5.7. Choose DataFlash media tab in the SAM-BA GUI interface.

5.8. Initialize DataFlash choosing the Enable action in the Scripts rolling menu and press Execute.

5.9. Choose Send boot file, press Execute.

5.10. Select AT91SAM9G45_RomCode_Replacement.bin binary file and press Open ; the media is written down

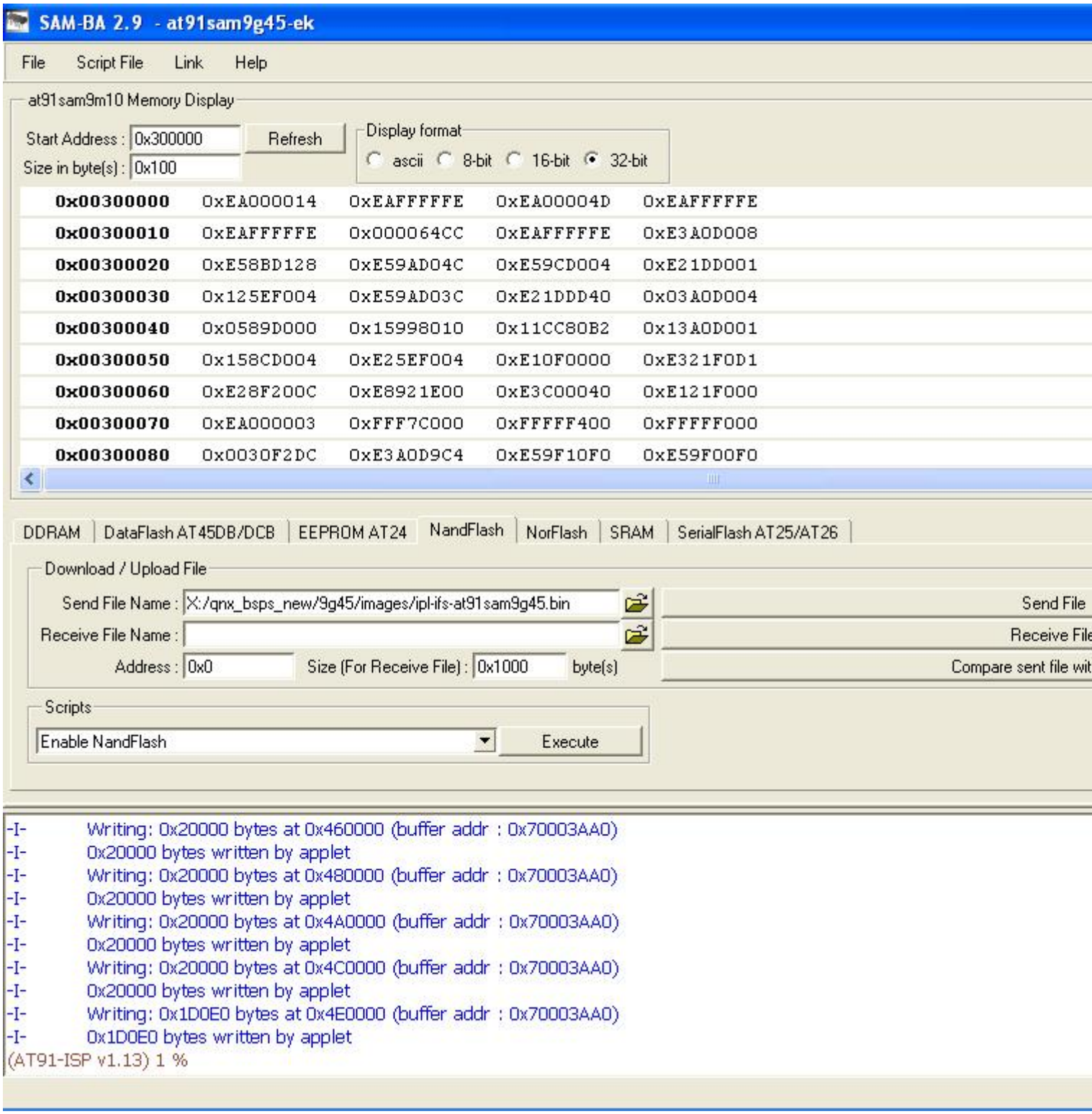
5.11. Close SAM-BA

6. Reset the board. Restart SMA-BA2.9.

7. Start SAM-BA 2.9 with \usb\ARM0 as connection.

Loading the IFS image using SAM-BA#

- Remove jumper JP10, restart the board, restart the SAM-BA application, select the proper board (at91sam9g45-ek).
- Select the Nandflash tab in SAM-BA application window. Put the jumper JP10 back on the board. Execute the Enable Nandflash script.
- Select the ipl-ifs-at91sam9g45.bin file to be sent to the target, and then press Send File. It will take a few seconds up to a minute.
- Sanpshot is as follows:



- Reset the board. Now on your terminal you will see output as follows:

```
QNX Neutrino Initial Program Loader for ATMEL AT91SAM9G45-EK
Commands:

Press 'D' for serial download, using the 'sendnto' utility
Press 'N' to Boot an OS image from NAND flash
```

- Press 'n', you will see the output as follows:

```
Atmel K9F2G08U0M detected.

QNX IFS image detected on page: 00000009 Offset: 00000000 Size: 005517EC
```

```
############################Done
found image, calling image setup...
image_setup OK, calling image start...


PIO init : DBGU, USART1, SPI0, Audio(AC97), NAND, USB, LCD, TWI (I2C), EMAC, SD/
MMC0,1
CPU0: Dcache: 1024x32 WB
CPU0: Icache: 1024x32
CPU0: 41069265: arm926 rev 5 400MHz
elf_map: 1M va=fe000000 pa=71000000 sz=00100000
elf_map: 1M va=fe000000 pa=71000000 sz=00100000
Header size=0x0000009c, Total Size=0x000005a8, #Cpu=1, Type=4
Section:system_private offset:0x000001c8 size:0x00000068
 syspage ptr user:fc404000 kernel:fc404000
 cpupage ptr user:fc4048a0 kernel:fc4048a0 spacing:84
 kdebug info:00000000 callback:00000000
 boot pgms: idx=0
   0) base paddr:71011000 start addr:fe0417f8
 ramsize:00000000 pagesize:00001000
Section:qtime offset:0x00000148 size:0x00000060
 boot:49e7a84c CPS:00000000007f2815 rate/scale:1200000048/-16 intr:1
Section:callout offset:0x000000a0 size:0x00000048
 reboot:fc404718 power:fc40473c
 timer_load:fc404750 reload:fc404778 value:fc4047a0
 0) display:fc4047bc poll:fc4047e0 break:fc404804
 1) display:00000000 poll:00000000 break:00000000
Section:cpuinfo offset:0x000001a8 size:0x00000020
 0) cpu:41069265 flags:40000000 speed:00000190 cache i/d:1/0 name:56
Section:cacheattr offset:0x00000568 size:0x00000040
 0) flags:32 size:0020 #lines:0400 control:fc4045a8 next:255
 1) flags:11 size:0020 #lines:0400 control:fc4045fc next:255
Section:meminfo offset:0x000005a8 size:0x00000000
Section:asinfo offset:0x000003c8 size:0x00000160
 0000) 0000000000000000-00000000ffffffff o:ffff a:0010 p:100 c:00000000 n:21
 0020) 0000000070000000-0000000077ffffff o:0000 a:0017 p:100 c:00000000 n:28
 0040) 0000000000000000-00000000ffffffff o:ffff a:0010 p:100 c:00000000 n:21
 0060) 0000000070000000-0000000077ffffff o:0040 a:0007 p:100 c:00000000 n:32
 0080) 000000007100e108-0000000071b9801f o:0000 a:0005 p:100 c:00000000 n:110
 00a0) 0000000071000000-000000007100e107 o:0000 a:0007 p:100 c:00000000 n:118
 00c0) 000000007100e108-0000000071b9801f o:0000 a:0007 p:100 c:00000000 n:126
 00e0) 0000000070000000-0000000070007fff o:0020 a:0007 p:100 c:00000000 n:134
 0100) 00000000700108f4-0000000070ffffff o:0020 a:0007 p:100 c:00000000 n:134
 0120) 0000000071b98020-0000000073dcffff o:0020 a:0007 p:100 c:00000000 n:134
 0140) 0000000073fd0000-0000000077ffffff o:0020 a:0007 p:100 c:00000000 n:134
Section:hwinfo offset:0x000002e8 size:0x000000e0
 0) size:3 tag:3 isize:3, iname:0, owner:65535, kids:1
 12) size:3 tag:17 isize:3, iname:9, owner:0, kids:3
 24) size:3 tag:3 isize:3, iname:37, owner:12, kids:1
 36) size:4 tag:49 isize:4, iname:41, owner:24, kids:0
   00 00 00 00
 52) size:3 tag:3 isize:3, iname:63, owner:12, kids:1
 64) size:4 tag:49 isize:16, iname:71, owner:52, kids:0
   00 00 00 00
 80) size:6 tag:76
   08 00 00 00 00 c0 fb ff 00 00 00 00 00 00 ff ff 00 00 00 00
 104) size:2 tag:85
   19 00 00 00
 112) size:4 tag:89
```

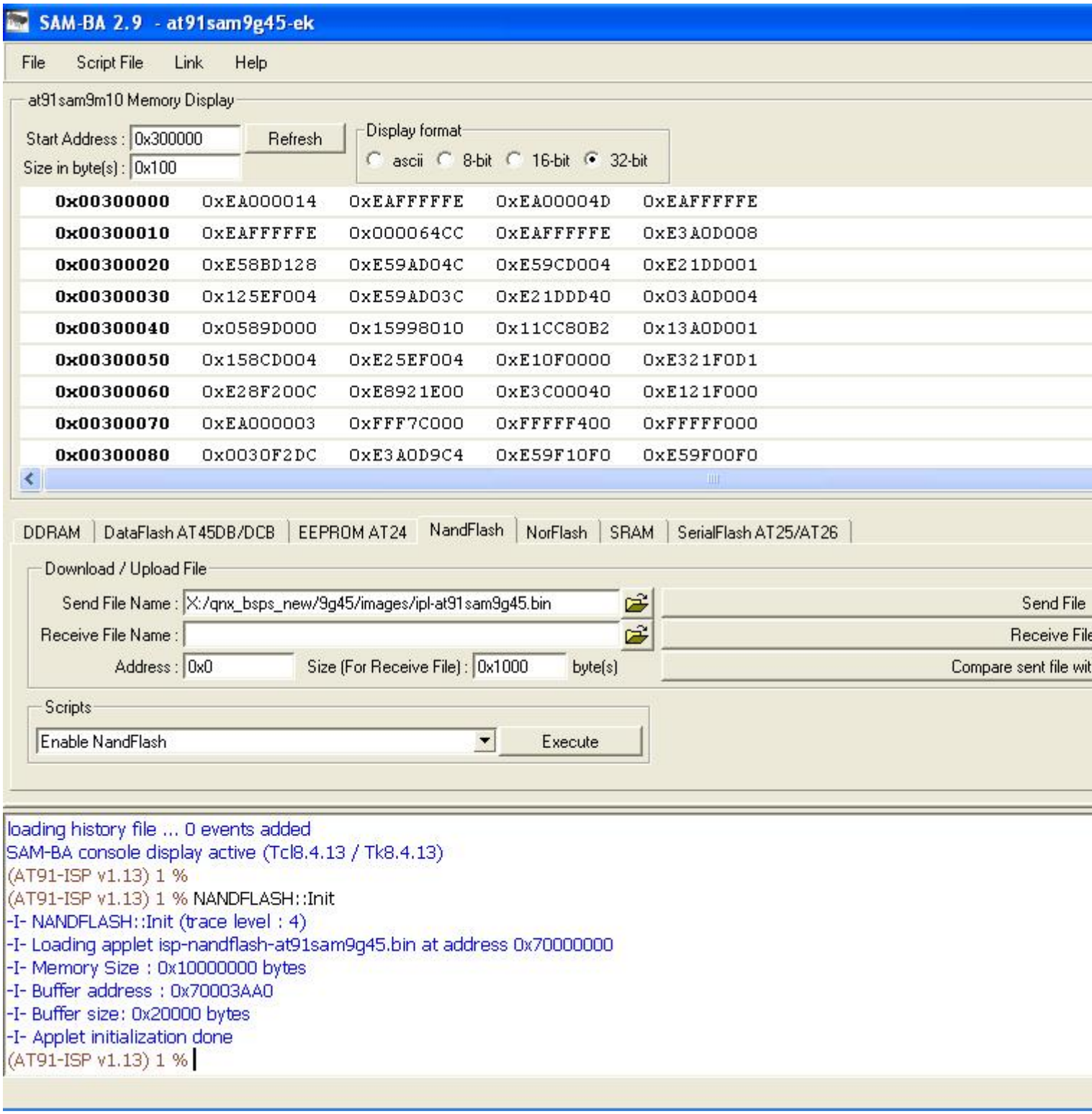
```
    06 00 00 00 66 22 00 02 16 15 00 00
  128) size:3 tag:3 isize:3, iname:97, owner:12, kids:1
  140) size:4 tag:49 isize:13, iname:101, owner:128, kids:0
    00 00 00 00
  156) size:1 tag:106
  160) size:6 tag:76
    08 00 00 00 00 ec ff ff 00 00 00 00 00 00 ff ff 00 00 00 00
  184) size:2 tag:85
    15 00 00 00
Section:typed_strings offset:0x00000230 size:0x00000028
  off:0 type:5 string:'AT91SAM9G45'
  off:16 type:2 string:'localhost'
Section:strings offset:0x00000258 size:0x00000090
  [0]'hw' [3]'Group' [9]'unknown' [17]'Bus' [21]'memory' [28]'ram' [32]'1to1'
  [37]'rtc' [41]'at91rtc' [49]'Device' [56]'arm926' [63]'network' [71]'emac'
  [76]'location' [85]'irq' [89]'nicaddr' [97]'dma' [101]'dmac' [106]'pad'
  [110]'imagefs' [118]'startup' [126]'bootram' [134]'sysram'
Section:intrinfo offset:0x00000528 size:0x00000040
  0) vector_base:00000000, #vectors:32, cascade_vector:7fffffff
     cpu_intr_base:00000000, cpu_intr_stride:0, flags:0000
      id => flags:8000, size:0028, rtn:fc404668
     eoi => flags:9000, size:0028, rtn:fc404690
     mask:fc4046b8, unmask:fc4046dc, config:00000000
Section:smp offset:0x000005a8 size:0x00000000
Section:pminfo offset:0x000005a8 size:0x00000000
Section:mdriver offset:0x000005a8 size:0x00000000
Section:boxinfo offset:0x000005a8 size:0x00000000
Section:cpu offset:0x00000128 size:0x00000020
  page_flush:fc40462c page_flush_deferred:fc404664
  upte_ro:00000aae upte_rw:00000ffe
  kpte_ro:0000000e kpte_rw:0000055e
  mask_nc:0000000c
  mmu_cr1:00051078 set:0000317f clr:00000000 -> 0005317f

System page at phys:70010000 user:fc404000 kern:fc404000
Starting next program at vfe0417f8
cpu_startnext: cpu0 -> fe0417f8
Welcome to QNX Neutrino 6.4 on the Atmel AT91SAM9G45 Board
Starting DBGU driver...
Starting Serial USART 1 driver ...
Starting Ethernet driver ...
Starting SPI0 driver...
Starting Audio driver...
Starting Graphics driver...
Starting Touchscreen driver...
Starting USB driver...
starting Input Drivers...
Starting NAND driver...
fs-etfs-at91sam9xx: devio_init: Flash ID: Manufacturer ID=0x2c, Device ID=0xaa
Starting SD/MMC driver...

Process 20 (resource_seed) exited status=0.
Starting i2c1 driver...
Starting i2c2 driver...
# Path=0 - Atmel MCI/HSMCI
 target=0 lun=0     Direct-Access(0) - SD:39 SD1GB Rev: 2.0
Path=0 - Atmel MCI/HSMCI
 target=0 lun=0     Direct-Access(0) - SD:3 SU04G Rev: 8.0
```

#

Step 3B: Reprogram the IPL.#

In case if Board is new or IPL gets corrupted, We can reprogram IPL using SAM-BA application.

- Remove jumper JP10, restart the board, restart the SAM-BA application, select the proper board (at91sam9g45-ek).
- Select the Nandflash tab in SAM-BA application window. Put the jumper JP10 back on the board. Execute the Enable Nandflash script.
- Select the ipl-at91sam9g45.bin file to be sent to the target, and then press Send File. It will take a few seconds up to a minute.
- Sanpshot is as follows:



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

Summary of driver commands#

The driver command lines below are specific to the Atmel AT91SAM9G45-EK board. See the online docs for each driver for additional command-line options and other details.

Note: Some of the following 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.

Startup:#

Command:

```
startup-at91sam9g45 -r 0x73dd0000,0x200000,1 -vvvvvvv
```

Serial:#

Command:

```
devc-serusart -F -S -u2 -b115200 -c133000000 0xfff90000^2,8
```

Required binaries:

- devc-serusart

Command:

```
devc-serdebug -e -F -S -b115200 -c133000000 0xffffee00,1
```

Required binaries:

- devc-serdebug

SPI:#

Command:

```
spi-master -d at91sam9xx base=0xFFFA4000,irq=14,clock=133000000
```

Required binaries:

- spi-master
- spi-at91sam9xx.so

Network:#

Command:

```
io-pkt-v4 -dat91sam9xx syspage
```

Required binaries:

- devnp-at91sam9xx.so
- io-pkt-v4

Audio:#

Command:

```
io-audio -d at91sam9xx_ac97 ioport=0xfffac000,irq=24
```

Required binaries:

- io-audio
- libasound.so
- deva-mixer-ac97.so
- deva-ctrl-at91sam9xx_ac97.so

USB:#

Command:

```
io-usb -dehci ioport=0x00800000,irq=22 -dohci ioport=0x00700000,irq=22
```

Required binaries and libraries:

- io-usb
- usb
- devu-ohci.so
- devu-ehci.so
- libusbdi.so
- class drivers

SD/MMC:#

Command:

```
devb-mmcsd-at91sam9xx mmcsd ioport=0xFFF80000,irq=11,bs=dmac:dbase=0xffffec00:dintf=0

devb-mmcsd-at91sam9xx mmcsd ioport=0xFFFD0000,irq=29,bs=dmac:dbase=0xffffec00:dintf=13
```

Required binaries:

- devb-mmcsd-at91sam9xx
- resource_seed

I2C#

Command:

```
i2c-at91sam9xx -p0xFFF84000 -i12 -c133000000 --u0

i2c-at91sam9xx -p0xFFF88000 -i13 -c133000000 --u1
```

Required binaries:

- i2c-at91sam9xx

Libraries:

- libi2c-master.a

ETFS NAND flash#

Command:

```
fs-etfs-at91sam9xx -D addr=0x40000000,board_id=at91sam9m10-ek -m /fs/etfs -r16384
```

Required binaries:

- fs-etfs-at91sam9xx
- etfsctl

Note: The first 16M bytes of NAND flash are reserved for IPL and QNX OS image.
Note: For more information about these commands, see the Neutrino Utilities Reference.

Touchscreen#

Command:

```
devi-at91sam9xx touch -a 0xfffb0000
```

Graphics#

Command:

```
Photon &
waitfor /dev/photon
io-display -dvid=0x0,did=0x0
io-graphics
pwm &
pterm -x10 -y10 -h250 -w200 -t"QNX 6.4.0" -K 03 &
devc-pty &
```

Required binaries:

- devg-at91sam9xx.so
- libph.so
- libAp.so
- libphexlib.so
- libphrender.so
- libffb.so
- libdisputil.so
- libimg.so.1
- ttfFFcore.so
- PHFcore.so
- FCcore.so
- libFF-T2K.so
- libblkcache.so
- libFF-T2K-fm.so
- libFF-T2K-cache.so
- phfont.so
- libfontutils.so
- libfont.so

Required configuration files:

- /usr/photon/config/at91sam9xx.conf=${PWD}/../src/hardware/devg/atl91sam9xx/at91sam9xx.conf
- /etc/system/config/display.conf=${PWD}/../src/hardware/devg/atl91sam9xx/display.conf

About graphics#

This driver currently supports the AT91SAM9G45 integrated LCD controller . It was developed on the Atmel AT91SAM9G45 Evaluation Board. This is the GF graphics driver is loaded by io-display.

LCD Displays#

- By default the driver sets up the Hitachi TX09D71VM1CCA TFT with the AT91SAM9G45 reference board.

You can use the driver configuration file at91sam9xx.conf to change display formats.

HW Format	QNX Format	Notes
16-bit	16-bit	QNX framework format is RGB565
24-bit	24-bit	QNX framework format is RGB888

Reserving Memory / Memory Restrictions#

The Atmel AT91SMA9G45 is a UMA system (Unified Memory Architecture). This means there is no dedicated video memory in the system. Surfaces displayed by the LCD controller, and rendered by the CPU, reside in system memory

- To ensure there is enough memory available for graphics, we recommend that memory be reserved at startup by using the -r option to startup.

For example to reserve 2 MB of memory:

startup-at91sam9g45 -r 0x73dd0000,0x200000,1 -vvvvvvv

where 0x73dd0000 is the physical base address of memory, and 0x200000 is the size of memory reserved in bytes.

For a list of options available to the driver please see the at91sam9xx.conf file.

Known Issues for This BSP#

- The ethernet driver may fail to come up sometime. The port seems to come up but pings fail. To recover the driver must be slayed and restarted (sometimes this needs to be done more than once).