

Release Notes for the QNX Neutrino 6.4.0 BSP for Atmel AT91SAM9RL64-EK Board#

System requirements#

Target system#

- QNX Neutrino RTOS 6.4.0
- Board: Atmel at91sam9rl64-EK Evaluation Board
- Data-Flash: AT45DB642D (8 MB)
- SDRAM: 16-bit Micron MT48LC16M16A2 (64 MB) SDRAM
- NAND Flash: Micron MT29F2G08AAC (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 application 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 AT91SAM9RL64 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-at91sam9rl64.bin

*ipl-at91sam9rl64.bin

*ipl-ifs-at91sam9rl64.bin

mkflashimage script creates a combined IPL/IFS image as ipl-ifs-at91sam9rl64.bin

The mkflashimage script:

```
#!/bin/sh
```

```
# script to build a binary IPL and boot image for ATMEL AT91SAM9RL64 Evaluation Kit board
```

```
# NOTE the image (ipl-ifs-at91sam9rl64.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-at91sam9rl64.bin > ipl-at91sam9rl64.bin
```

```
# Combine the BINARY IPL with the BINARY OS Image
```

```
cat ./ipl-at91sam9rl64.bin ./ifs-at91sam9rl64.bin > ipl-ifs-at91sam9rl64.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.

- Then NAND Flash Boot program is 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 NAND Flash Boot program 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 "AT91-ISP v1.12.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. Remove jumpers J13 .
4. 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.
5. Restart the board, hyperterminal will be showing following message .

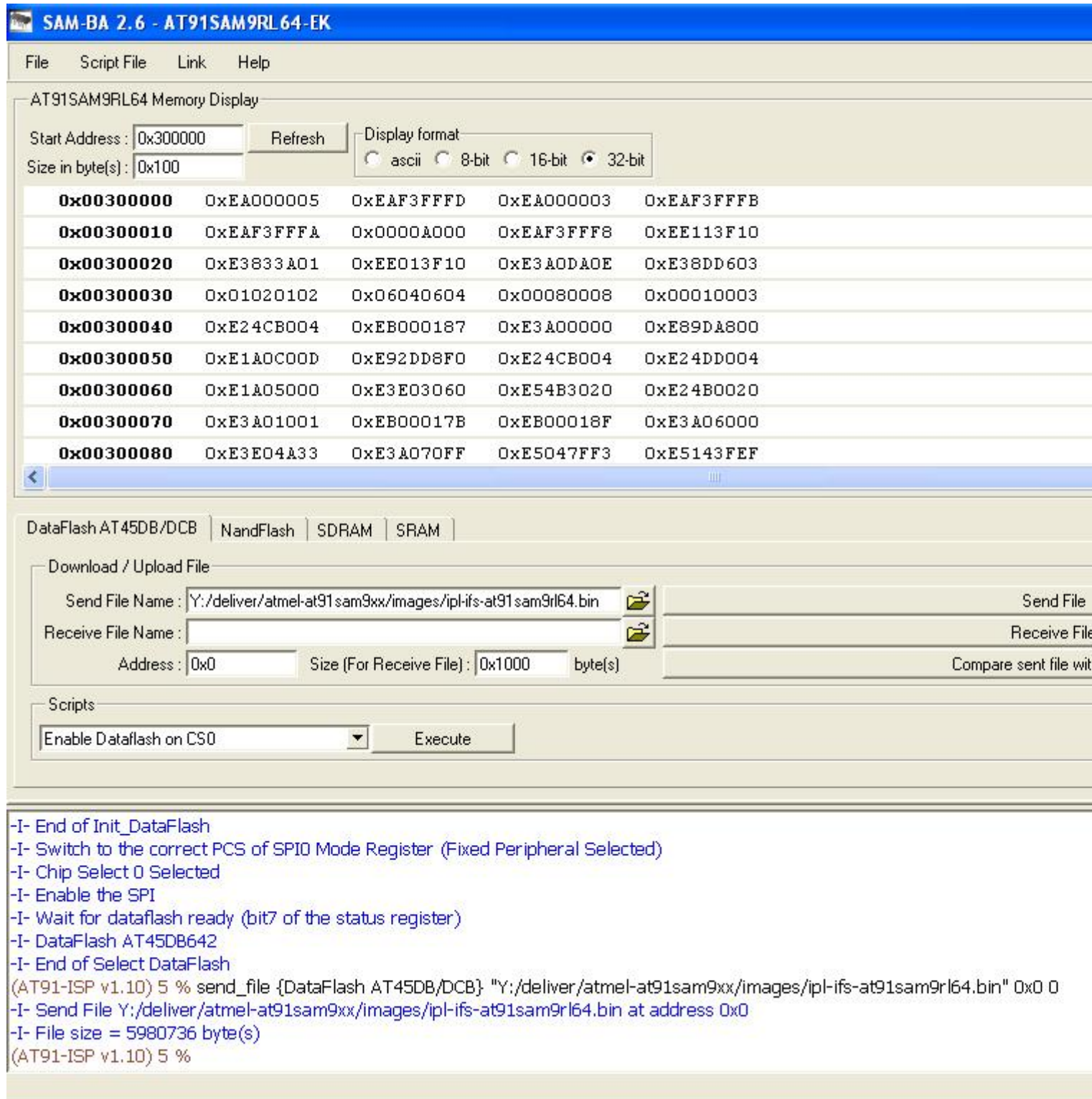
```
RomBOOT
```

```
>
```

6. Connect usb cable with windows machine, It will prompt with a new usb hardware found message, and will try to install the corresponding driver.
7. Check device has enumerated properly by looking into device manager .
8. Enable jumpers J13.
9. Start SAM-BA 2.6 with \usb\ARM0 as connection.

Loading the IFS image using SAM-BA#

- Select the Data-Flash AT45DB/DCB tab in SAM-BA Application. Execute the Enable Data-flash on CS0 script.
- Select the ipl-ifs-at91sam9rl64.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.



The screenshot shows the SAM-BA 2.6 - AT91SAM9RL64-EK application interface. The top menu bar includes File, Script File, Link, and Help. Below the menu is the AT91SAM9RL64 Memory Display section, which includes a Start Address field (0x300000), a Refresh button, a Display format dropdown (set to 32-bit), and a Size in byte(s) field (0x100). The memory display shows a grid of hexadecimal values. Below this is the DataFlash AT45DB/DCB section, which includes tabs for NandFlash, SDRAM, and SRAM. The Download / Upload File section has fields for Send File Name (Y:/deliver/atmel-at91sam9xx/images/ipl-ifs-at91sam9rl64.bin), Receive File Name, Address (0x0), and Size (For Receive File) (0x1000). The Scripts section has a dropdown menu (Enable Dataflash on CS0) and an Execute button. The bottom section shows the output of the application, including messages like "-I- End of Init_DataFlash" and "-I- Send File Y:/deliver/atmel-at91sam9xx/images/ipl-ifs-at91sam9rl64.bin at address 0x0".

Press the reset button, now on your terminal you will see output as follows:

RomBOOT

>

QNX Neutrino Initial Program Loader for ATMEL AT91SAM9RL-EK

Commands:

Press 'D' to download an OS image from serial port, using sendnto utility

Press 'F' to Boot an OS image from SPI/SERIAL flash
ATMEL AT45DB642D SPI Flash detected.

QNX IFS image detected on page: 00000010 Offset: 00000220 Size: 005C0240
#####Done
found image, calling image setup...
image_setup OK, calling image start...

PIO init : DBGU, USART, Audio(AC97), NAND, LCD.

CPU0: Dcache: 128x32 WB

CPU0: Icache: 128x32

CPU0: 41069265: arm926 rev 5 100MHz

elf_map: 1M va=fe000000 pa=20100000 sz=00100000

elf_map: 1M va=fe000000 pa=20100000 sz=00100000

Header size=0x0000009c, Total Size=0x00000508, #Cpu=1, Type=4

Section:system_private offset:0x000001f0 size:0x00000068

syspage ptr user:fc404000 kernel:fc404000

cpupage ptr user:fc404808 kernel:fc404808 spacing:84

kdebug info:00000000 callback:00000000

boot pgms: idx=0

0) base paddr:20110000 start addr:fe03d838

ramsize:00000000 pagesize:00001000

Section:qtime offset:0x00000148 size:0x00000060

boot:00000000 CPS:0000000002faf08 rate/scale:320000000/-15 intr:1

Section:callout offset:0x000000a0 size:0x00000048

reboot:fc404680 power:fc4046a0

timer_load:fc4046b4 reload:fc4046dc value:fc404708

0) display:fc404728 poll:fc40474c break:fc404770

1) display:00000000 poll:00000000 break:00000000

Section:cpuinfo offset:0x000001a8 size:0x00000020

0) cpu:41069265 flags:40000000 speed:00000064 cache i/d:1/0 name:53

Section:cacheattr offset:0x000004c8 size:0x00000040

0) flags:32 size:0020 #lines:0080 control:fc404508 next:255

1) flags:11 size:0020 #lines:0080 control:fc40455c next:255

Section:meminfo offset:0x00000508 size:0x00000000

Section:asinfo offset:0x00000328 size:0x00000160

0000) 0000000000000000-00000000ffffff o:ffff a:0010 p:100 c:00000000 n:21

0020) 0000000020000000-0000000023ffffff o:0000 a:0017 p:100 c:00000000 n:28

0040) 0000000000000000-00000000ffffff o:ffff a:0010 p:100 c:00000000 n:21

0060) 0000000020000000-0000000023ffffff o:0040 a:0007 p:100 c:00000000 n:32

0080) 000000002010e108-00000000206c023f o:0000 a:0005 p:100 c:00000000 n:60

00a0) 0000000020100000-000000002010e107 o:0000 a:0007 p:100 c:00000000 n:68

00c0) 000000002010e108-00000000206c023f o:0000 a:0007 p:100 c:00000000 n:76

00e0) 0000000020000000-0000000020007fff o:0020 a:0007 p:100 c:00000000 n:84

0100) 000000002001085c-00000000200fffff o:0020 a:0007 p:100 c:00000000 n:84

0120) 00000000206c0240-0000000023dcffff o:0020 a:0007 p:100 c:00000000 n:84

0140) 0000000023fd0000-0000000023ffffff o:0020 a:0007 p:100 c:00000000 n:84

Section:hwinfo offset:0x000002e0 size:0x00000048

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:1

24) size:3 tag:3 isize:3, iname:37, owner:12, kids:1

36) size:4 tag:46 isize:4, iname:41, owner:24, kids:0

00 00 00 00

Section:typed_strings offset:0x00000258 size:0x00000028

off:0 type:5 string:'AT91SAM9RL64'

off:20 type:2 string:'localhost'

Section:strings offset:0x00000280 size:0x00000060

[0]'hw' [3]'Group' [9]'unknown' [17]'Bus' [21]'memory' [28]'ram' [32]'lto1'

```
[37]'rtc' [41]'NONE' [46]'Device' [53]'arm926' [60]'imagefs' [68]'startup'
[76]'bootram' [84]'sysram'
Section:intrinfo offset:0x00000488 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:002c, rtn:fc4045c8
  eoi => flags:9000, size:0028, rtn:fc4045f4
  mask:fc40461c, unmask:fc404640, config:00000000
Section:smp offset:0x00000508 size:0x00000000
Section:pminfo offset:0x00000508 size:0x00000000
Section:mdriver offset:0x00000508 size:0x00000000
Section:boxinfo offset:0x000001c8 size:0x00000028
  hw_flags:00000000
Section:cpu offset:0x00000128 size:0x00000020
  page_flush:fc40458c page_flush_deferred:fc4045c4
  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:20010000 user:fc404000 kern:fc404000
Starting next program at vfe03d838
cpu_startnext: cpu0 -> fe03d838
Welcome to QNX Neutrino 6.4 on the Atmel AT91SAM9RL64 Board
Starting DBGU driver...
Starting Serial USART 1 driver ...
Starting SPI driver...
Starting Audio driver...
Starting Graphics driver...
Starting NAND driver...
#
```

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

Step 3B: Reprogram the IPL.#

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

- Select the Data-Flash AT45DB/DCB tab in SAM-BA application window, . Execute the Enable Data-flash on CS0 script.
- Select the ipl-at91sam9rl64.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:

File Script File Link Help

AT91SAM9RL64 Memory Display

Start Address: Refresh Display format: ascii 8-bit 16-bit 32-bit

Size in byte(s):

0x00300000	0xEA000005	0xEAF3FFFD	0xEA000003	0xEAF3FFFB
0x00300010	0xEAF3FFFA	0x0000A000	0xEAF3FFF8	0xEE113F10
0x00300020	0xE3833A01	0xEE013F10	0xE3A0DA0E	0xE38DD603
0x00300030	0x01020102	0x06040604	0x00080008	0x00010003
0x00300040	0xE24CB004	0xEB000187	0xE3A00000	0xE89DA800
0x00300050	0xE1A0C00D	0xE92DD8F0	0xE24CB004	0xE24DD004
0x00300060	0xE1A05000	0xE3E03060	0xE54B3020	0xE24B0020
0x00300070	0xE3A01001	0xEB00017B	0xEB00018F	0xE3A06000
0x00300080	0xE3E04A33	0xE3A070FF	0xE5047FF3	0xE5143FEF

DataFlash AT45DB/DCB | NandFlash | SDRAM | SRAM

Download / Upload File

Send File Name:

Receive File Name:

Address: Size (For Receive File): byte(s)

Scripts

```
-I- End of Init_DataFlash
-I- Switch to the correct PCS of SPI0 Mode Register (Fixed Peripheral Selected)
-I- Chip Select 0 Selected
-I- Enable the SPI
-I- Wait for dataflash ready (bit7 of the status register)
-I- DataFlash AT45DB642
-I- End of Select DataFlash
(AT91-ISP v1.10) 5 % send_file {DataFlash AT45DB/DCB} "Y:/deliver/atmel-at91sam9xx/images/ipl-at91sam9rl64.bin" 0x0 0
-I- Send File Y:/deliver/atmel-at91sam9xx/images/ipl-at91sam9rl64.bin at address 0x0
-I- File size = 16384 byte(s)
(AT91-ISP v1.10) 5 %
```

Summary of driver commands#

The driver command lines below are specific to the Atmel AT91SAM9RL64 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-at91sam9rl64 -r 0x23dd0000,0x200000,1 -vvvvvvv
```

Serial:#

Command:

```
devc-serusart -F -S -u2 -b115200 -c50000000 0xfffb0000^2,6
```

Required binaries:

- devc-serusart

Command:

```
devc-serdebug -e -F -S -b115200 -c50000000 0xffff200,1
```

Required binaries:

- devc-serdebug

SPI:#

Command:

```
spi-master -d at91sam9xx base=0xFFFCC000,irq=13,clock=50000000
```

Required binaries:

- spi-master
- spi-at91sam9xx.so

Audio:#

Command:

```
io-audio -d at91sam9xx_ac97 ioport=0xfffd8000,irq=24
```

Required binaries:

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

ETFS NAND flash#

Command:

```
fs-etfs-at91sam9xx -D addr=0x40000000,board_id=at91sam92r164-ek -m /fs/etfs
```

Required binaries:

- fs-etfs-at91sam9xx
- etfsctl

Touchscreen:#

Command:

devi-at91sam9xx touch

Required binaries:

- devi-at91sam9xx

MMC SD:#

Command:

```
devb-mmcsd-at91sam9xx mmcsd ioport=0xFFFFA4000,irq=10
```

Required binaries and libraries:

- devb-mmcsd-at91sam9xx
- libcam.so
- cam-disk.so
- io-blk.so
- fs-dos.so
- fs-qnx4.so

DMA:#

Required library:

- libdma-at91sam9xx.so

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
- ttFFcore.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/at91sam9xx/at91sam9xx.conf
- /etc/system/config/display.conf=\${PWD}/../src/hardware/devg/at91sam9xx/display.conf

Note: For more information about these commands, see the Neutrino Utilities Reference.

About graphics#

This driver currently supports the AT91SAM9RL64 integrated LCD controller . It was developed on the Atmel AT91SAM9RL64 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 AT91SAM9RL64 reference board.

You can use the driver configuration file at91sam9xx.conf to other displays.

HW Format	QNX Format	Notes
8-bit	8-bit	8 bits-per-pixel palette-based
16-bit	15-bit	QNX framework format is ARGB1555, but MSB is not actually used because the hardware uses only 15 bits
24-bit	32-bit	actually unpacked 24-bit, meaning the upper 8 bits are not used

Reserving Memory / Memory Restrictions#

The Atmel AT91SMA9RL64 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-at91sam9rl64 -r 0x23dd0000,0x200000,1 -vvvvvvv
```

where 0x23dd0000 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.