NDS Technical Documentation

Martin Roth and Filip Jerkovic

Contents

About

About this Documentation

GBATEK written 2001-2014 by Martin Korth, programming specs for the GBA and NDS hardware, I've been trying to keep the specs both as short as possible, and as complete as possible. The document is part of the no$gba debuggers built-in help text.

Updates

The standalone docs in TXT and HTM format are updated when having added any major changes to the document. The no$gba built-in version will be updated more regularly, including for minor changes, along with all no$gba updates.

Homepage

- http://problemkaputt.de/gba.htm - no$gba emulator homepage (freeware)
- http://problemkaputt.de/gba-dev.htm - no$gba debugger homepage
- http://problemkaputt.de/gbapics.htm - no$gba debugger screenshots
- http://problemkaputt.de/gbatek.htm - gbatek html version
- http://problemkaputt.de/gbatek.txt - gbatek text version

Feedback

If you find any information in this document to be misleading, incomplete, or incorrect, please say something! My spam-shielded email address is found at:
http://problemkaputt.de/email.htm - contact
Mail from programmers only, please. No gaming questions, thanks.

Credits

Thanks for GBATEK fixes, and for info about GBA and NDS hardware,

- Jasper Vijn
- Remi Veilleux (DS video details)
- Randy Linden

- Sebastian Rasmussen
- Stephen Stair (DS Wifi)
- Cue (DS Firmware bits and bytes)
- Tim Seidel (DS Wifi RF2958 datasheet)
- Damien Good (DS Bios Dumping, and lots of e-Reader info)
- Kenobi and Dualscreenman (lots of ARDS/CBDS cheat info)
- Flubba (GBA X/Y-Axis tilt sensor, and GBA Gameboy Player info)
- DarkFader (DS Key2)
- Dstek by neimod (DS Sound)
- Christian Auby
- Jeff Frohwein
- NDSTech Wiki, http://www.bottledlight.com/ds/ (lots of DS info)

Formatting
TXT is 80 columns, TXT is 80 columns, TXT is 80 columns.
Don't trust anything else. Never.

1 Overview

1.1 DS Technical Data

1.1.1 Processors

- 1x ARM946E-S 32bit RISC CPU, 66MHz (NDS9 video) (not used in GBA mode)
- 1x ARM7TDMI 32bit RISC CPU, 33MHz (NDS7 sound) (16MHz in GBA mode)

1.1.2 Internal Memory

- 4096KB **Main RAM (8192KB in debug version)**
- 96KB **WRAM (64K mapped to NDS7, plus 32K mappable to NDS7 or NDS9)**
- 60KB **TCM/Cache (TCM: 16K Data, 32K Code) (Cache: 4K Data, 8K Code)**
- 656KB **VRAM (allocateable as BG/OBJ/2D/3D/Palette/Texture/WRAM memory)**
- 4KB **OAM/PAL (2K OBJ Attribute Memory, 2K Standard Palette RAM)**
- 248KB **Internal 3D Memory (104K Polygon RAM, 144K Vertex RAM)**
- ?KB **Matrix Stack, 48 scanline cache**
- 8KB **Wifi RAM**
- 256KB **Firmware FLASH (512KB in iQue variant, with chinese charset)**
- 36KB **BIOS ROM (4K NDS9, 16K NDS7, 16K GBA)**

1.1.3 Video

2x LCD screens (each 256x192 pixel, 3 inch, 18bit color depth, backlight)
2x 2D video engines (extended variants of the GBA's video controller)
1x 3D video engine (can be assigned to upper or lower screen)
1x video capture (for effects, or for forwarding 3D to the 2nd 2D engine)

1.1.4 Sound

16 sound channels (16x PCM8/PCM16/IMA-ADPCM, 6x PSG-Wave, 2x PSG-Noise)
2 sound capture units (for echo effects, etc.)
Output: Two built-in stereo speakers, and headphones socket
Input: One built-in microphone, and microphone socket

1.1.5 Controls

Gamepad	4 Direction Keys, 8 Buttons
Touchscreen	(on lower LCD screen)

1.1.6 Communication Ports

- Wifi IEEE802.11b

1.1.7 Specials

- Built-in Real Time Clock
- Power Management Device
- Hardware divide and square root functions
- CP15 System Control Coprocessor (cache, tcm, pu, bist, etc.)

1.1.8 External Memory

- NDS Slot (for NDS games) (encrypted 8bit data bus, and serial 1bit bus)
- GBA Slot (for NDS expansions, or for GBA games) (but not for DMG/CGB games)

1.1.9 Manufactured Cartridges

- ROM: 16MB, 32MB, or 64MB
- EEPROM/FLASH/FRAM: 0.5KB, 8KB, 64KB, 256KB, or 512KB

1.1.10 Can be booted from

- NDS Cartridge (NDS mode)
- Firmware FLASH (NDS mode) (eg. by patching firmware via ds-xboo cable)
- Wifi (NDS mode)
- GBA Cartridge (GBA mode) (without DMG/CGB support) (without SIO support)

1.1.11 Power Supply

- Built-in rechargeable Lithium ion battery, 3.7V 1000mAh (DS-Lite)
- External Supply: 5.2V DC

1.1.12 NDS-Lite

Slightly smaller than the original NDS, coming in a more decently elegant case. The LCDs are much more colorful (and thus not backwards compatible with any older NDS or GBA games), and the LCDs support wider viewing angles. Slightly different power managment device (with selectable backlight brightness, new external power source flag, lost audio amplifier mute flag). Slightly different Wifi controller (different chip ID, different dirt effects when accessing invalid wifi ports and unused wifi memory regions, different behaviour on GAPDISP registers, RF/BB chips replaced by a single chip). Slightly different touch screen controller (with new unused input, and slightly different powerdown bits).

1.1.13 Notice

NDS9 means the ARM9 processor and its memory and I/O ports in NDS mode
NDS7 means the ARM7 processor and its memory and I/O ports in NDS mode
GBA means the ARM7 processor and its memory and I/O ports in GBA mode

1.1.14 The two Processors

Most game code is usually executed on the ARM9 processor (in fact, Nintendo reportedly doesn't allow developers use the ARM7 processor, except by predefined API functions, anyways, even with the most likely inefficient API code, most of the ARM7's 33MHz horsepower is left unused).
The ARM9's 66MHz "horsepower" is a different tale - it seems Nintendo thought that a 33MHz processor would be too "slow" for 3D games, and so they (tried to) badge an additional CPU to the original GBA hardware.
However, the real 66MHz can be used only with cache and tcm, all other memory and I/O accesses are delayed to the 33MHz bus clock, that'd be still quite fast, but, there seems to be a hardware glitch that adds 3

waitcycles to all nonsequential accesses at the NDS9 side, which effectively drops its bus clock to about 8MHz, making it ways slower than the 33MHz NDS7 processor, it's even slower than the original 16MHz GBA processor.
Altogether, with the bugged 66MHz, and the unused 33MHz, Nintendo could have reached almost the same power when staying with the GBA's 16MHz processor :-)
Although, when properly using cache/tcm, then the 66MHz processor ¡can¿ be very fast, still, the NDS should have worked as well with a single processor, though using only an ARM9 might cause a lot of compatibility problems with GBA games, so there's at least one reason for keeping the ARM7 included.

1.2 DS I/O Maps

1.2.1 ARM9 Display Engine A

4000000h 4 2D Engine A - DISPCNT - LCD Control (Read/Write)
4000004h 2 2D Engine A+B - DISPSTAT - General LCD Status (Read/Write)
4000006h 2 2D Engine A+B - VCOUNT - Vertical Counter (Read only)
4000008h 50h 2D Engine A (same registers as GBA, some changed bits)
4000060h 2 DISP3DCNT - 3D Display Control Register (R/W)
4000064h 4 DISPCAPCNT - Display Capture Control Register (R/W)
4000068h 4 DISP_MMEM_FIFO - Main Memory Display FIFO (R?/W)
400006Ch 2 2D Engine A - MASTER_BRIGHT - Master Brightness Up/Down

1.2.2 ARM9 DMA, Timers, and Keypad

40000B0h 30h DMA Channel 0..3
40000E0h 10h DMA FILL Registers for Channel 0..3
4000100h 10h Timers 0..3
4000130h 2 KEYINPUT
4000132h 2 KEYCNT

1.2.3 ARM9 IPC/ROM

4000180h 2 IPCSYNC - IPC Synchronize Register (R/W)
4000184h 2 IPCFIFOCNT - IPC Fifo Control Register (R/W)
4000188h 4 IPCFIFOSEND - IPC Send Fifo (W)
40001A0h 2 AUXSPICNT - Gamecard ROM and SPI Control
40001A2h 2 AUXSPIDATA - Gamecard SPI Bus Data/Strobe
40001A4h 4 Gamecard bus timing/control
40001A8h 8 Gamecard bus 8-byte command out
40001B0h 4 Gamecard Encryption Seed 0 Lower 32bit
40001B4h 4 Gamecard Encryption Seed 1 Lower 32bit
40001B8h 2 Gamecard Encryption Seed 0 Upper 7bit (bit7-15 unused)
40001BAh 2 Gamecard Encryption Seed 1 Upper 7bit (bit7-15 unused)

1.2.4 ARM9 Memory and IRQ Control

4000204h 2 EXMEMCNT - External Memory Control (R/W)
4000208h 2 IME - Interrupt Master Enable (R/W)
4000210h 4 IE - Interrupt Enable (R/W)
4000214h 4 IF - Interrupt Request Flags (R/W)
4000240h 1 VRAMCNT_A - VRAM-A (128K) Bank Control (W)
4000241h 1 VRAMCNT_B - VRAM-B (128K) Bank Control (W)
4000242h 1 VRAMCNT_C - VRAM-C (128K) Bank Control (W)
4000243h 1 VRAMCNT_D - VRAM-D (128K) Bank Control (W)
4000244h 1 VRAMCNT_E - VRAM-E (64K) Bank Control (W)
4000245h 1 VRAMCNT_F - VRAM-F (16K) Bank Control (W)
4000246h 1 VRAMCNT_G - VRAM-G (16K) Bank Control (W)

4000247h 1 WRAMCNT - WRAM Bank Control (W)
4000248h 1 VRAMCNT_H - VRAM-H (32K) Bank Control (W)
4000249h 1 VRAMCNT_I - VRAM-I (16K) Bank Control (W)

1.2.5 ARM9 Maths

4000280h 2 DIVCNT - Division Control (R/W)
4000290h 8 DIV_NUMER - Division Numerator (R/W)
4000298h 8 DIV_DENOM - Division Denominator (R/W)
40002A0h 8 DIV_RESULT - Division Quotient (=Numer/Denom) (R)
40002A8h 8 DIVREM_RESULT - Division Remainder (=Numer MOD Denom) (R)
40002B0h 2 SQRTCNT - Square Root Control (R/W)
40002B4h 4 SQRT_RESULT - Square Root Result (R)
40002B8h 8 SQRT_PARAM - Square Root Parameter Input (R/W)
4000300h 4 POSTFLG - Undoc
4000304h 2 POWCNT1 - Graphics Power Control Register (R/W)

1.2.6 ARM9 3D Display Engine

4000320h..6A3h

1.2.7 ARM9 Display Engine B

4001000h 4 2D Engine B - DISPCNT - LCD Control (Read/Write)
4001008h 50h 2D Engine B (same registers as GBA, some changed bits)
400106Ch 2 2D Engine B - MASTER_BRIGHT - 16bit - Brightness Up/Down

1.2.8 ARM9 DSi Extra Registers

40021Axh .. DSi Registers
4004xxxh .. DSi Registers

1.2.9 ARM9 IPC/ROM

4100000h 4 IPCFIFORECV - IPC Receive Fifo (R)
4100010h 4 Gamecard bus 4-byte data in, for manual or dma read

1.2.10 ARM9 DS Debug Registers (Emulator/Devkits)

4FFF0xxh .. Ensata Emulator Debug Registers
4FFFAxxh .. No$gba Emulator Debug Registers

1.2.11 ARM9 Hardcoded RAM Addresses for Exception Handling

27FFD9Ch .. NDS9 Debug Stacktop / Debug Vector (0=None)
DTCM+3FF8h 4 NDS9 IRQ Check Bits (hardcoded RAM address)
DTCM+3FFCh 4 NDS9 IRQ Handler (hardcoded RAM address)

1.2.12 Main Memory Control

27FFFFEh 2 Main Memory Control

1.2.13 Further Memory Control Registers

ARM CP15 System Control Coprocessor

1.2.14 ARM7 I/O Map

4000004h 2 DISPSTAT
4000006h 2 VCOUNT
40000B0h 30h DMA Channels 0..3
4000100h 10h Timers 0..3
4000120h 4 Debug SIODATA32
4000128h 4 Debug SIOCNT
4000130h 2 keyinput
4000132h 2 keycnt
4000134h 2 Debug RCNT
4000136h 2 EXTKEYIN
4000138h 1 RTC Realtime Clock Bus
4000180h 2 IPCSYNC - IPC Synchronize Register (R/W)

4000184h 2 IPCFIFOCNT - IPC Fifo Control Register (R/W)

4000188h 4 IPCFIFOSEND - IPC Send Fifo (W)

40001A0h 2 AUXSPICNT - Gamecard ROM and SPI Control

40001A2h 2 AUXSPIDATA - Gamecard SPI Bus Data/Strobe

40001A4h 4 Gamecard bus timing/control

40001A8h 8 Gamecard bus 8-byte command out

40001B0h 4 Gamecard Encryption Seed 0 Lower 32bit

40001B4h 4 Gamecard Encryption Seed 1 Lower 32bit

40001B8h 2 Gamecard Encryption Seed 0 Upper 7bit (bit7-15 unused)

40001BAh 2 Gamecard Encryption Seed 1 Upper 7bit (bit7-15 unused)

40001C0h 2 SPI bus Control (Firmware, Touchscreen, Powerman)

40001C2h 2 SPI bus Data

1.2.15 ARM7 Memory and IRQ Control

4000204h 2 EXMEMSTAT - External Memory Status

4000206h 2 WIFIWAITCNT

4000208h 4 IME - Interrupt Master Enable (R/W)

4000210h 4 IE - Interrupt Enable (R/W)

4000214h 4 IF - Interrupt Request Flags (R/W)

4000218h - IE2 ;DSi only (additional ARM7 interrupt sources)

400021Ch - IF2 ;/

4000240h 1 VRAMSTAT - VRAM-C,D Bank Status (R)

4000241h 1 WRAMSTAT - WRAM Bank Status (R)

4000300h 1 POSTFLG

4000301h 1 HALTCNT (different bits than on GBA) (plus NOP delay)

4000304h 2 POWCNT2 Sound/Wifi Power Control Register (R/W)

4000308h 4 BIOSPROT - Bios-data-read-protection address

1.2.16 ARM7 Sound Registers

4000400h 100h Sound Channel 0..15 (10h bytes each)

40004x0h 4 SOUNDxCNT - Sound Channel X Control Register (R/W)

40004x4h 4 SOUNDxSAD - Sound Channel X Data Source Register (W)

40004x8h 2 SOUNDxTMR - Sound Channel X Timer Register (W)

40004xAh 2 SOUNDxPNT - Sound Channel X Loopstart Register (W)

40004xCh 4 SOUNDxLEN - Sound Channel X Length Register (W)

4000500h 2 SOUNDCNT - Sound Control Register (R/W)

4000504h 2 SOUNDBIAS - Sound Bias Register (R/W)

4000508h 1 SNDCAP0CNT - Sound Capture 0 Control Register (R/W)

4000509h 1 SNDCAP1CNT - Sound Capture 1 Control Register (R/W)

4000510h 4 SNDCAP0DAD - Sound Capture 0 Destination Address (R/W)

4000514h 2 SNDCAP0LEN - Sound Capture 0 Length (W)

4000518h 4 SNDCAP1DAD - Sound Capture 1 Destination Address (R/W)

400051Ch 2 SNDCAP1LEN - Sound Capture 1 Length (W)

1.2.17 ARM7 DSi Extra Registers

40021Axh .. DSi Registers

4004xxxh .. DSi Registers

4004700h 2 DSi SNDEXCNT Register ;mapped even in DS mode

4004C0xh .. DSi GPIO Registers ;

1.2.18 ARM7 IPC/ROM

4100000h 4 IPCFIFORECV - IPC Receive Fifo (R)

4100010h 4 Gamecard bus 4-byte data in, for manual or dma read

1.2.19 ARM7 WLAN Registers

4800000h .. Wifi WS0 Region (32K) (Wifi Ports, and 8K Wifi RAM)
4808000h .. Wifi WS1 Region (32K) (mirror of above, other waitstates)

1.2.20 ARM7 Hardcoded RAM Addresses for Exception Handling

380FFC0h 4 DSi7 IRQ IF2 Check Bits (hardcoded RAM address) (DSi only)
380FFDCh .. NDS7 Debug Stacktop / Debug Vector (0=None)
380FFF8h 4 NDS7 IRQ IF Check Bits (hardcoded RAM address)
380FFFCh 4 NDS7 IRQ Handler (hardcoded RAM address)

1.3 DS Memory Maps

1.3.1 NDS9 Memory Map

00000000h Instruction TCM (32KB) (not moveable) (mirror-able to 1000000h)
0xxxx000h Data TCM (16KB) (moveable)
02000000h Main Memory (4MB)
03000000h Shared WRAM (0KB, 16KB, or 32KB can be allocated to ARM9)
04000000h ARM9-I/O Ports
05000000h Standard Palettes (2KB) (Engine A BG/OBJ, Engine B BG/OBJ)
06000000h VRAM - Engine A, BG VRAM (max 512KB)
06200000h VRAM - Engine B, BG VRAM (max 128KB)
06400000h VRAM - Engine A, OBJ VRAM (max 256KB)
06600000h VRAM - Engine B, OBJ VRAM (max 128KB)
06800000h VRAM - "LCDC"-allocated (max 656KB)
07000000h OAM (2KB) (Engine A, Engine B)
08000000h GBA Slot ROM (max 32MB)
0A000000h GBA Slot RAM (max 64KB)
FFFF0000h ARM9-BIOS (32KB) (only 3K used)
The ARM9 Exception Vectors are located at FFFF0000h. The IRQ handler redirects to [DTCM+3FFCh].

1.3.2 NDS7 Memory Map

00000000h ARM7-BIOS (16KB)
02000000h Main Memory (4MB)
03000000h Shared WRAM (0KB, 16KB, or 32KB can be allocated to ARM7)
03800000h ARM7-WRAM (64KB)
04000000h ARM7-I/O Ports
04800000h Wireless Communications Wait State 0 (8KB RAM at 4804000h)
04808000h Wireless Communications Wait State 1 (I/O Ports at 4808000h)
06000000h VRAM allocated as Work RAM to ARM7 (max 256K)
08000000h GBA Slot ROM (max 32MB)
0A000000h GBA Slot RAM (max 64KB)
The ARM7 Exception Vectors are located at 00000000h. The IRQ handler redirects to [3FFFFFCh aka 380FFFCh].

1.3.3 Further Memory (not mapped to ARM9/ARM7 bus)

3D Engine Polygon RAM (52KBx2)
3D Engine Vertex RAM (72KBx2)
Firmware (256KB) (built-in serial flash memory)
GBA-BIOS (16KB) (not used in NDS mode)
NDS Slot ROM (serial 8bit-bus, max 4GB with default protocol)
NDS Slot FLASH/EEPROM/FRAM (serial 1bit-bus)

1.3.4 Shared-RAM

Even though Shared WRAM begins at 3000000h, programs are commonly using mirrors at 37F8000h (both ARM9 and ARM7). At the ARM7-side, this allows to use 32K Shared WRAM and 64K ARM7-WRAM as a continous 96K RAM block.

1.3.5 Undefined I/O Ports

On the NDS (at the ARM9-side at least) undefined I/O ports are always zero.

1.3.6 Undefined Memory Regions

16MB blocks that do not contain any defined memory regions (or that contain only mapped TCM regions) are typically completely undefined.
16MB blocks that do contain valid memory regions are typically containing mirrors of that memory in the unused upper part of the 16MB area (only exceptions are TCM and BIOS which are not mirrored).

2 Hardware Programming

2.1 DS Memory Control

2.1.1 Cache and TCM

TCM and Cache are controlled by the System Control Coprocessor,
ARM CP15 System Control Coprocessor
The specifications for the NDS9 are:

2.1.1.1 Tightly Coupled Memory (TCM) ITCM 32K, base=00000000h (fixed, not move-able)
DTCM 16K, base=moveable (default base=27C0000h)
Note: Although ITCM is NOT moveable, the NDS Firmware configures the ITCM size to 32MB, and so, produces ITCM mirrors at 0..1FFFFFFh. Furthermore, the PU can be used to lock/unlock memory in that region. That trick allows to move ITCM anywhere within the lower 32MB of memory.

2.1.1.2 Cache Data Cache 4KB, Instruction Cache 8KB
4-way set associative method
Cache line 8 words (32 bytes)
Read-allocate method (ie. writes are not allocating cache lines)
Round-robin and Pseudo-random replacement algorithms selectable
Cache Lockdown, Instruction Prefetch, Data Preload
Data write-through and write-back modes selectable

2.1.1.3 Protection Unit (PU)
Recommended/default settings are:

Region	Name	Address	Size	Cache	WBuf	Code	Data
-	Background	00000000h	4GB	-	-	-	-
0	I/O and VRAM	04000000h	64MB	-	-	R/W	R/W
1	Main Memory	02000000h	4MB	On	On	R/W	R/W
2	ARM7-dedicated	027C0000h	256KB	-	-	-	-
3	GBA Slot	08000000h	128MB	-	-	-	R/W
4	DTCM	027C0000h	16KB	-	-	-	R/W
5	ITCM	01000000h	32KB	-	-	R/W	R/W
6	BIOS	FFFF0000h	32KB	On	-	R	R
7	Shared Work	027FF000h	4KB	-	-	-	R/W

Notes: In Nintendo's hardware-debugger, Main Memory is expanded to 8MB (for that reason, some addresses are at 27NN000h instead 23NN000h) (some of the extra memory is reserved for the debugger, some can be used for game development). Region 2 and 7 are not understood? GBA Slot should be max 32MB+64KB, rounded up to 64MB, no idea why it is 128MB? DTCM and ITCM do not use Cache and Write-Buffer because TCM is fast. Above settings do not allow to access Shared Memory at 37F8000h? Do not use cache/wbuf for I/O, doing so might suppress writes, and/or might read outdated values.
The main purpose of the Protection Unit is debugging, a major problem with GBA programs have been faulty accesses to memory address 00000000h and up (due to [base+offset] addressing with uninitialized

(zero) base values). This problem has been fixed in the NDS, for the ARM9 processor at least, still there are various leaks: For example, the 64MB I/O and VRAM area contains only ca. 660KB valid addresses, and the ARM7 probably doesn't have a Protection Unit at all. Alltogether, the protection is better than in GBA, but it's still pretty crude compared with software debugging tools.
Region address/size are unified (same for code and data), however, cachabilty and access rights are non-unified (and may be separately defined for code and data).
Note: The NDS7 doesn't have any TCM, Cache, or CP15.

2.1.2 Cartridges and Main RAM

2.1.2.1 4000204h - NDS9 - EXMEMCNT - 16bit - External Memory Control (R/W)

2.1.2.2 4000204h - NDS7 - EXMEMSTAT - 16bit - External Memory Status (R/W..R)

0-1 32-pin GBA Slot SRAM Access Time (0-3 = 10, 8, 6, 18 cycles)
2-3 32-pin GBA Slot ROM 1st Access Time (0-3 = 10, 8, 6, 18 cycles)
4 32-pin GBA Slot ROM 2nd Access Time (0-1 = 6, 4 cycles)
5-6 32-pin GBA Slot PHI-pin out (0-3 = Low, 4.19MHz, 8.38MHz, 16.76MHz)
7 32-pin GBA Slot Access Rights (0=ARM9, 1=ARM7)
8-10 Not used (always zero)
11 17-pin NDS Slot Access Rights (0=ARM9, 1=ARM7)
12 Not used (always zero)
13 NDS:Always set? ;set/tested by DSi bootcode: Main RAM enable, CE2 pin?
14 Main Memory Interface Mode Switch (0=Async/GBA/Reserved, 1=Synchronous)
15 Main Memory Access Priority (0=ARM9 Priority, 1=ARM7 Priority)
Bit0-6 can be changed by both NDS9 and NDS7, changing these bits affects the local EXMEM register only, not that of the other CPU.
Bit7-15 can be changed by NDS9 only, changing these bits affects both EXMEM registers, ie. both NDS9 and NDS7 can read the current NDS9 setting.
Bit14=0 is intended for GBA mode, however, writes to this bit appear to be ignored?
DS Main Memory Control

2.1.2.3 GBA Slot (8000000h-AFFFFFFh)

The GBA Slot can be mapped to ARM9 or ARM7 via EXMEMCNT.7.
For the selected CPU, memory at 8000000h-9FFFFFFh contains the "GBA ROM" region, and memory at A000000h-AFFFFFFh contains the "GBA SRAM" region (repeated every 64Kbytes). If there is no cartridge in GBA Slot, then the ROM/SRAM regions will contain open-bus values: SRAM region is FFh-filled (High-Z). And ROM region is filled by increasing 16bit values (Addr/2), possibly ORed with garbage depending on the selected ROM Access Time:
6 clks – returns "Addr/2"
8 clks – returns "Addr/2"
10 clks – returns "Addr/2 OR FE08h" (or similar garbage)
18 clks – returns "FFFFh" (High-Z)
For the deselected CPU, all memory at 8000000h-AFFFFFFh becomes 00h-filled, this is required for bugged games like Digimon Story: Super Xros Wars (which is accidently reading deselected GBA SRAM at [main_ram_base+main_ram_addr*4], whereas it does presumably want to read Main RAM at [main_ram_base+index*4]).

2.1.3 WRAM

2.1.3.1 4000247h - NDS9 - WRAMCNT - 8bit - WRAM Bank Control (R/W)

2.1.3.2 4000241h - NDS7 - WRAMSTAT - 8bit - WRAM Bank

Status (R) Should not be changed when using Nintendo's API.
0-1 ARM9/ARM7 (0-3 = 32K/0K, 2nd 16K/1st 16K, 1st 16K/2nd 16K, 0K/32K)
2-7 Not used
The ARM9 WRAM area is 3000000h-3FFFFFFh (16MB range).
The ARM7 WRAM area is 3000000h-37FFFFFh (8MB range).
The allocated 16K or 32K are mirrored everywhere in the above areas.
De-allocation (0K) is a special case: At the ARM9-side, the WRAM area is then empty (containing undefined data). At the ARM7-side, the WRAM area is then containing mirrors of the 64KB ARM7-WRAM (the memory at 3800000h and up).

2.1.4 VRAM

2.1.4.1 4000240h - NDS7 - VRAMSTAT - 8bit - VRAM Bank Status (R)
0 VRAM C enabled and allocated to NDS7 (0=No, 1=Yes)
1 VRAM D enabled and allocated to NDS7 (0=No, 1=Yes)
2-7 Not used (always zero)
The register indicates if VRAM C/D are allocated to NDS7 (as Work RAM), ie. if VRAMCNT_C/D are enabled (Bit7=1), with MST=2 (Bit0-2). However, it does not reflect the OFS value.

2.1.4.2 4000240h - NDS9 - VRAMCNT_A - 8bit - VRAM-A (128K) Bank Control (W)

2.1.4.3 4000241h - NDS9 - VRAMCNT_B - 8bit - VRAM-B (128K) Bank Control (W)

2.1.4.4 4000242h - NDS9 - VRAMCNT_C - 8bit - VRAM-C (128K) Bank Control (W)

2.1.4.5 4000243h - NDS9 - VRAMCNT_D - 8bit - VRAM-D (128K) Bank Control (W)

2.1.4.6 4000244h - NDS9 - VRAMCNT_E - 8bit - VRAM-E (64K) Bank Control (W)

2.1.4.7 4000245h - NDS9 - VRAMCNT_F - 8bit - VRAM-F (16K) Bank Control (W)

2.1.4.8 4000246h - NDS9 - VRAMCNT_G - 8bit - VRAM-G (16K) Bank Control (W)

2.1.4.9 4000248h - NDS9 - VRAMCNT_H - 8bit - VRAM-H (32K) Bank Control (W)

2.1.4.10 4000249h - NDS9 - VRAMCNT_I - 8bit - VRAM-I (16K) Bank Control (W)
0-2 VRAM MST ;Bit2 not used by VRAM-A,B,H,I
3-4 VRAM Offset (0-3) ;Offset not used by VRAM-E,H,I
5-6 Not used
7 VRAM Enable (0=Disable, 1=Enable)
There is a total of 656KB of VRAM in Blocks A-I.
Table below shows the possible configurations.

VRAM	SIZE	MST	OFS	ARM9, Plain ARM9-CPU Access (so-called LCDC mode)
A	128K	0	-	6800000h-681FFFFh
B	128K	0	-	6820000h-683FFFFh
C	128K	0	-	6840000h-685FFFFh
D	128K	0	-	6860000h-687FFFFh
E	64K	0	-	6880000h-688FFFFh
F	16K	0	-	6890000h-6893FFFh
G	16K	0	-	6894000h-6897FFFh
H	32K	0	-	6898000h-689FFFFh
I	16K	0	-	68A0000h-68A3FFFh
VRAM	**SIZE**	**MST**	**OFS**	**ARM9, 2D Graphics Engine A, BG-VRAM (max 512K)**
A,B,C,D	128K	1	0..3	6000000h+(20000h*OFS)
E	64K	1	-	6000000h

F,G 16K 1 0..3 6000000h+(4000h*OFS.0)+(10000h*OFS.1)
VRAM SIZE MST OFS ARM9, 2D Graphics Engine A, OBJ-VRAM (max 256K)
A,B 128K 2 0..1 6400000h+(20000h*OFS.0) ;(OFS.1 must be zero)
E 64K 2 - 6400000h
F,G 16K 2 0..3 6400000h+(4000h*OFS.0)+(10000h*OFS.1)
VRAM SIZE MST OFS 2D Graphics Engine A, BG Extended Palette
E 64K 4 - Slot 0-3 ;only lower 32K used
F,G 16K 4 0..1 Slot 0-1 (OFS=0), Slot 2-3 (OFS=1)
VRAM SIZE MST OFS 2D Graphics Engine A, OBJ Extended Palette
F,G 16K 5 - Slot 0 ;16K each (only lower 8K used)
VRAM SIZE MST OFS Texture/Rear-plane Image
A,B,C,D 128K 3 0..3 Slot OFS(0-3) ;(Slot2-3: Texture, or Rear-plane)
VRAM SIZE MST OFS Texture Palette
E 64K 3 - Slots 0-3 ;OFS=don't care
F,G 16K 3 0..3 Slot (OFS.0*1)+(OFS.1*4) ;ie. Slot 0, 1, 4, or 5
VRAM SIZE MST OFS ARM9, 2D Graphics Engine B, BG-VRAM (max 128K)
C 128K 4 - 6200000h
H 32K 1 - 6200000h
I 16K 1 - 6208000h
VRAM SIZE MST OFS ARM9, 2D Graphics Engine B, OBJ-VRAM (max 128K)
D 128K 4 - 6600000h
I 16K 2 - 6600000h
VRAM SIZE MST OFS 2D Graphics Engine B, BG Extended Palette
H 32K 2 - Slot 0-3
VRAM SIZE MST OFS 2D Graphics Engine B, OBJ Extended Palette
I 16K 3 - Slot 0 ;(only lower 8K used)
VRAM SIZE MST OFS ¡ARM7¿, Plain ¡ARM7¿-CPU Access
C,D 128K 2 0..1 6000000h+(20000h*OFS.0) ;OFS.1 must be zero

2.1.4.11 Notes

In Plain-CPU modes, VRAM can be accessed only by the CPU (and by the Capture Unit, and by VRAM Display mode). In "Plain ¡ARM7¿-CPU Access" mode, the VRAM blocks are allocated as Work RAM to the NDS7 CPU.

In BG/OBJ VRAM modes, VRAM can be accessed by the CPU at specified addresses, and by the display controller.

In Extended Palette and Texture Image/Palette modes, VRAM is not mapped to CPU address space, and can be accessed only by the display controller (so, to initialize or change the memory, it should be temporarily switched to Plain-CPU mode).

All VRAM (and Palette, and OAM) can be written to only in 16bit and 32bit units (STRH, STR opcodes), 8bit writes are ignored (by STRB opcode). The only exception is "Plain ¡ARM7¿-CPU Access" mode: The ARM7 CPU can use STRB to write to VRAM (the reason for this special feature is that, in GBA mode, two 128K VRAM blocks are used to emulate the GBA's 256K Work RAM).

2.1.4.12 Other Video RAM

Aside from the map-able VRAM blocks, there are also some video-related memory regions at fixed addresses:

5000000h Engine A Standard BG Palette (512 bytes)

5000200h Engine A Standard OBJ Palette (512 bytes)

5000400h Engine B Standard BG Palette (512 bytes)

5000600h Engine B Standard OBJ Palette (512 bytes)

7000000h Engine A OAM (1024 bytes)

7000400h Engine B OAM (1024 bytes)

2.1.5 BIOS

2.1.5.1 4000308h - NDS7 - BIOSPROT - Bios-data-read-protection address

Used to double-protect the first some KBytes of the NDS7 BIOS. The BIOS is split into two protection regions, one always active, one controlled by the BIOSPROT register. The overall idea is that only the BIOS can read from itself, any other attempts to read from that regions return FFh-bytes.

Opcodes at...	Can read from	Expl.
0..[BIOSPROT]-1	0..3FFFh	Double-protected (when BIOSPROT is set)
[BIOSPROT]..3FFFh	[BIOSPROT]..3FFFh	Normal-protected (always active)

The initial BIOSPROT setting on power-up is zero (disabled). Before starting the cartridge, the BIOS boot code sets the register to 1204h (actually 1205h, but the mis-aligned low-bit is ignored). Once when initialized, further writes to the register are ignored.

The double-protected region contains the exception vectors, some bytes of code, and the cartridge KEY1 encryption seed (about 4KBytes). As far as I know, it is impossible to unlock the memory once when it is locked, however, with some trickery, it is possible execute code before it gets locked. Also, the two THUMB opcodes at 05ECh can be used to read all memory at 0..3FFFh,

05ECh ldrb r3,[r3,12h] ;requires incoming r3=src-12h

05EEh pop r2,r4,r6,r7,r15 ;requires dummy values & THUMB retadr on stack

Additionally most BIOS functions (eg. CpuSet), include a software-based protection which rejects source addresses in the BIOS area (the only exception is GetCRC16, though it still cannot bypass the BIOSPROT setting).

2.1.5.2 Note

The NDS9 BIOS doesn't include any software or hardware based read protection.

2.1.6 DS Memory Timings

2.1.6.1 System Clock

Bus clock = 33MHz (33.513982 MHz) (1FF61FEh Hertz)
NDS7 clock = 33MHz (same as bus clock)
NDS9 clock = 66MHz (internally twice bus clock; for cache/tcm)
Most timings in this document are specified for 33MHz clock (not for the 66MHz clock). Respectively, NDS9 timings are counted in "half" cycles.

2.1.6.2 Memory Access Times

Tables below show the different access times for code/data fetches on arm7/arm9 cpus, measured for sequential/nonsequential 32bit/16bit accesses.

NDS7/CODE					NDS9/CODE					
N32	S32	N16	S16	Bus	N32	S32	N16	S16	Bus	
9	2	8	1	16	9	9	4.5	4.5	16	Main RAM (read) (cache off)
1	1	1	1	32	4	4	2	2	32	WRAM,BIOS,I/O,OAM
2	2	1	1	16	5	5	2.5	2.5	16	VRAM,Palette RAM
16	12	10	6	16	19	19	9.5	9.5	16	GBA ROM (example 10,6 access)
-	-	-	-	-	0.5	0.5	0.5	0.5	32	TCM, Cache_Hit
-	-	-	-	-	(–Load 8 words–)					Cache_Miss

NDS7/DATA					NDS9/DATA					
N32	S32	N16	S16	Bus	N32	S32	N16	S16	Bus	
10	2	9	1	16	10	2	9	1	16	Main RAM (read) (cache off)
1	1	1	1	32	4	1	4	1	32	WRAM,BIOS,I/O,OAM
1?	2	1	1	16	5	2	4	1	16	VRAM,Palette RAM
15	12	9	6	16	19	12	13	6	16	GBA ROM (example 10,6 access)
9	10	9	10	8	13	10	13	10	8	GBA RAM (example 10 access)
-	-	-	-	-	0.5	0.5	0.5	-	32	TCM, Cache_Hit
-	-	-	-	-	(–Load 8 words–)					Cache_Miss
-	-	-	-	-	11	11	11	-	32	Cache_Miss (BIOS)

- - - - - 23 23 23 - 16 Cache_Miss (Main RAM)
All timings are counted in 33MHz units (so "half" cycles can occur on NDS9).
Note: 8bit data accesses have same timings than 16bit data.
*** DS Memory Timing Notes ***
The NDS timings are altogether pretty messed up, with different timings for CODE and DATA fetches, and different timings for NDS7 and NDS9...

2.1.6.3 NDS7/CODE

Timings for this region can be considered as "should be" timings.

2.1.6.4 NDS7/DATA

Quite the same as NDS7/CODE. Except that, nonsequential Main RAM accesses are 1 cycle slower, and more strange, nonsequential GBA Slot accesses are 1 cycle faster.

2.1.6.5 NDS9/CODE

This is the most messiest timing. An infamous PENALTY of 3 cycles is added to all nonsequential accesses (except cache, tcm, and main ram). And, all opcode fetches are forcefully made nonsequential 32bit (the NDS9 simply doesn't support fast sequential opcode fetches). That applies also for THUMB code (two 16bit opcodes are fetched by a single nonsequential 32bit access) (so the time per 16bit opcode is one half of the 32bit fetch) (unless a branch causes only one of the two 16bit opcodes to be executed, then that opcode will have the full 32bit access time).

2.1.6.6 NDS9/DATA

Allows both sequential and nonsequential access, and both 16bit and 32bit access, so it's faster than NDS9/CODE. Nethertheless, it's still having the 3 cycle PENALTY on nonsequential accesses. And, similar as NDS7/DATA, it's also adding 1 cycle to nonsequential Main RAM accesses.
*** More Timing Notes / Lots of unsorted Info ***

2.1.6.7 Actual CPU Performance

The 33MHz NDS7 is running more or less nicely at 33MHz. However, the so-called "66MHz" NDS9 is having ¡much¿ higher waitstates, and it's effective bus speed is barely about 8..16MHz, the only exception is code/data in cache/tcm, which is eventually reaching real 66MHz (that, assuming cache HITS, otherwise, in case of cache MISSES, the cached memory timing might even drop to 1.4MHz or so?).
****
ARM9 opcode fetches are always N32 + 3 waits.
S16 and N16 do not exist (because thumb-double-fetching) (see there).
S32 becomes N32 (ie. the ARM9 does NOT support fast sequential timing).
That N32 is having same timing as normal N32 access on NDS7, plus 3 waits.
Eg. an ARM9 N32 or S32 to 16bit bus will take: N16 + S16 + 3 waits.
Eg. an ARM9 N32 or S32 to 32bit bus will take: N32 + 3 waits.
Main Memory is ALWAYS having the nonsequential 3 wait PENALTY (even on ARM7).
****
ARM9 Data fetches however are allowed to use sequential timing, as well as raw 16bit accesses (which aren't forcefully expanded to slow 32bit accesses).
Nethertheless, the 3 wait PENALTY is added to any NONSEQUENTIAL accesses.
Only exceptions are cache and tcm which do not have that penalty.
Eg. LDRH on 16bit-data-bus is N16+3waits.
Eg. LDR on 16bit-data-bus is N16+S16+3waits.
Eg. LDM on 16bit-data-bus is N16+(n*2-1)*S16+3waits.
Eventually, data fetches can take place parallel with opcode fetches.
That is NOT true for LDM (works only for LDR/LDRB/LDRH).
That is NOT true for DATA in SAME memory region than CODE.

That is NOT true for DATA in ITCM (no matter if CODE is in ITCM).
****

2.1.6.8 NDS9 Busses

Unlike ARM7, the ARM9 has separate code and data busses, allowing it to perform code and data fetches simultaneously (provided that both are in different memory regions).
Normally, opcode execution times are calculated as "(codetime+datatime)", with the two busses, it can (ideally) be "MAX(codetime,datatime)", so the data access time may virtually take "NULL" clock cycles.
In practice, DTCM and Data Cache access can take NULL cycles (however, data access to ITCM can't).
When executing code in cache/itcm, data access to non-cache/tcm won't be any faster than with only one bus (as it's best, it could subtract 0.5 cycles from datatime, but, the access must be "aligned" to the bus-clock, so the "datatime-0.5" will be rounded back to the original "datatime").
When executing code in uncached main ram, and accessing data (elsewhere than in main memory, cache/tcm), then execution time is typically "codetime+datatime-2".

2.1.6.9 NDS9 Internal Cycles

Additionally to codetime+datatime, some opcodes include one or more internal cycles. Compared with ARM7, the behaviour of that internal cycles is slightly different on ARM9. First of, on the NDS9, the internal cycles are of course "half" cycles (ie. counted in 66MHz units, not in 33MHz units) (although they may get rounded to "full" cycles upon next memory access outside tcm/cache). And, the ARM9 is in some cases "skipping" the internal cycles, that often depending on whether or not the next opcode is using the result of the current opcode.
Another big difference is that the ARM9 has lost the fast-multiply feature for small numbers; in some cases that may result in faster execution, but may also result in slower execution (one workaround would be to manually replace MUL opcodes by the new ARM9 halfword multiply opcodes); the slowest case are MUL opcodes that do update flags (eg. MULS, MLAS, SMULLS, etc. in ARM mode, and all ALL multiply opcodes in THUMB mode).

2.1.6.10 NDS9 Thumb Code

In thumb mode, the NDS9 is fetching two 16bit opcodes by a single 32bit read. In case of 32bit bus, this reduces the amount of memory traffic and may result in faster execution time, of course that works only if the two opcodes are within a word-aligned region (eg. loops at word-aligned addresses will be faster than non-aligned loops). However, the double-opcode-fetching is also done on 16bit bus memory, including for unnecessary fetches, such like opcodes after branch commands, so the feature may cause heavy slowdowns.

2.1.6.11 Main Memory

Reportedly, the main memory access times would be 5 cycles (nonsequential read), 4 cycles (nonsequential write), and 1 cycle (sequential read or write). Plus whatever termination cycles. Plus 3 cycles on nonsequential access to the last 2-bytes of a 32-byte block.
That's of course all wrong. Reads are much slower than 5 cycles. Not yet tested if writes are faster. And, I haven't been able to reproduce the 3 cycles on last 2-bytes effect, actually, it looks more as if that 3 cycles are accidently added to ALL nonsequential accesses, at ALL main memory addresses, and even to most OTHER memory regions... which might be the source of the PENALTY which occurs on VRAM/WRAM/OAM/Palette and I/O accesses.

2.1.6.12 DMA

In some cases DMA main memory read cycles are reportedly performed simultaneously with DMA write cycles to other memory.

2.1.6.13 NDS9

On the NDS9, all external memory access (and I/O) is delayed to bus clock (or actually MUCH slower due to the massive waitstates), so the full 66MHz can be used only internally in the NDS9 CPU core, ie. with cache and TCM.

2.1.6.14 Bus Clock

The exact bus clock is specified as 33.513982 MHz (1FF61FEh Hertz). However, on my own NDS, measured in relation to the RTC seconds IRQ, it appears more like 1FF6231h, that inaccuary of 1 cycle per 657138 cycles (about one second per week) on either oscillator, isn't too significant though.

2.1.6.15 GBA Slot

The access time for GBA slot can be configured via EXMEMCNT register.

2.1.6.16 VRAM Waitstates

Additionally, on NDS9, a one cycle wait can be added to VRAM accesses (when the video controller simultaneously accesses it) (that can be disabled by Forced Blank, see DISPCNT.Bit7). Moreover, additional VRAM waitstates occur when using the video capture function.

Note: VRAM being mapped to NDS7 is always free of additional waits.

2.2 DS Video

The NDS has two 2D Video Engines, each basically the same as in GBA, see GBA LCD Video Controller

2.2.0.1 NDS Specific 2D Video Fatures

2.2.1 DS Video Stuff

2.2.2 DS Video BG Modes / Control

2.2.3 DS Video OBJs

2.2.4 DS Video Extended Palettes

2.2.5 DS Video Capture and Main Memory Display Mode

2.2.6 DS Video Display System Block Diagram

2.3 DS 3D Video

2.4 DS Sound

2.5 DS System and Built-in Peripherals

2.6 DS Cartridges, Encryption and Firmware

2.7 DS Xboo

2.8 DS Wireless Communications

3 Other

3.1 BIOS Functions

The BIOS includes several System Call Functions which can be accessed by SWI instructions. Incoming parameters are usually passed through registers R0,R1,R2,R3. Outgoing registers R0,R1,R3 are typically containing either garbage, or return value(s). All other registers (R2,R4-R14) are kept unchanged.

3.1.0.1 Caution

When invoking SWIs from inside of ARM state specify SWI NN*10000h, instead of SWI NN as in THUMB state.

3.1.0.2 How BIOS Processes SWIs

SWIs can be called from both within THUMB and ARM mode. In ARM mode, only the upper 8bit of the 24bit comment field are interpreted.
Each time when calling a BIOS function 4 words (SPSR, R11, R12, R14) are saved on

Supervisor stack (_svc). Once it has saved that data, the SWI handler switches into System mode, so that all further stack operations are using user stack.

In some cases the BIOS may allow interrupts to be executed from inside of the SWI procedure. If so, and if the interrupt handler calls further SWIs, then care should be taken that the Supervisor Stack does not overflow.

3.1.1 BIOS Function Summary

3.1.2 BIOS Differences between GBA and NDS functions

3.1.3 BIOS Arithmetic Functions

3.1.4 BIOS Rotation/Scaling Functions

3.1.5 BIOS Decompression Functions

3.1.6 BIOS Memory Copy

3.1.7 BIOS Halt Functions

3.1.8 BIOS Reset Functions

3.1.9 BIOS Misc Functions

3.1.10 BIOS Multi Boot (Single Game Pak)

3.1.11 BIOS Sound Functions

3.1.12 BIOS SHA1 Functions (DSi Only)

3.1.13 BIOS RSA Functions (DSi Only)

3.1.14 BIOS Ram Usage

3.1.15 BIOS Dumping

3.2 ARM CPU Reference

3.2.0.1 General ARM7TDMI Information

3.2.1 ARM CPU Overview

3.2.2 ARM CPU Register Set

3.2.3 ARM CPU Flags & Condition Field (cond)

3.2.4 ARM CPU 26bit Memory Interface

3.2.5 ARM CPU Exceptions

3.2.6 ARM CPU Memory Alignments

3.2.6.1 ARM 32bit Instruction Set (ARM Code)

3.2.7 ARM Instruction Summary

3.2.8 ARM Opcodes: Branch and Branch with Link (B, BL, BX, BLX, SWI, BKPT)

3.2.9 ARM Opcodes: Data Processing (ALU)

3.2.10 ARM Opcodes: Multiply and Multiply-Accumulate (MUL, MLA)

3.2.11 ARM Opcodes: Special ARM9 Instructions (CLZ, QADD/QSUB)

3.2.12 ARM Opcodes: PSR Transfer (MRS, MSR)

3.2.13 ARM Opcodes: Memory: Single Data Transfer (LDR, STR, PLD)

3.2.14 ARM Opcodes: Memory: Halfword, Doubleword, and Signed Data Transfer

3.2.15 ARM Opcodes: Memory: Block Data Transfer (LDM, STM)

3.2.16 ARM Opcodes: Memory: Single Data Swap (SWP)

3.2.17 ARM Opcodes: Coprocessor Instructions (MRC/MCR, LDC/STC, CDP, MCRR/MRRC)

3.2.17.1 ARM 16bit Instruction Set (THUMB Code) When operating in THUMB state, cut-down 16bit opcodes are used.

THUMB is supported on T-variants of ARMv4 and up, ie. ARMv4T, ARMv5T, etc.

3.2.18 THUMB Instruction Summary

3.2.19 THUMB Opcodes: Register Operations (ALU, BX)

3.2.20 THUMB Opcodes: Memory Load/Store (LDR/STR)

3.2.21 THUMB Opcodes: Memory Addressing (ADD PC/SP)

3.2.22 THUMB Opcodes: Memory Multiple Load/Store (PUSH/POP and LDM/STM)

3.2.23 THUMB Opcodes: Jumps and Calls

3.2.23.1 Note Switching between ARM and THUMB state can be done by using the Branch and Exchange (BX) instruction.

3.2.24 Further Information

3.2.25 ARM Pseudo Instructions and Directives

3.2.26 ARM CP15 System Control Coprocessor

3.2.27 ARM CPU Instruction Cycle Times

3.2.28 ARM CPU Versions

3.2.29 ARM CPU Data Sheet

3.3 External Connectors

3.3.0.1 External Connectors

3.3.1 AUX GBA Game Pak Bus

3.3.2 AUX DS Game Card Slot

3.3.3 AUX Link Port

3.3.4 AUX Sound/Headphone Socket and Battery/Power Supply

3.3.5 AUX DSi SD/MMC Pin-Outs

3.3.5.1 Getting access to Internal Pins

3.3.6 AUX Opening the GBA

3.3.7 AUX Mainboard

3.3.8 AUX DSi Component Lists

3.3.9 AUX DSi Internal Connectors

3.3.10 AUX DSi Chipset Pinouts

3.3.10.1 More Internal Stuff

3.3.11 Pinouts - CPU - Signal Summary

3.3.12 Pinouts - CPU - Pinouts

3.3.13 Pinouts - Audio Amplifiers

3.3.14 Pinouts - LCD Cables

3.3.15 Pinouts - Power Switches, DC/DC Converters, Reset Generators

3.3.16 Pinouts - Wifi

3.3.17 Pinouts - Various

3.3.17.1 Xboo Multiboot Cable

3.3.18 AUX Xboo PC-to-GBA Multiboot Cable

3.3.19 AUX Xboo Flashcard Upload

3.3.20 AUX Xboo Burst Boot Backdoor