Nintendo DS - the (un)protections

by Dr. Andrea Q. - www.retrofixer.it

Youtube channel: https://www.youtube.com/channel/UCEw0CQ8LKyA9jVvWXkEwp4Q

And here we are at a new installment of this apparently popular column.

After covering the now-no-longer-mysterious iQue Player, today we try to unveil what lies inside Nintendo's next handheld console.

Released at the end of 2004 and "upgraded" later with the "Lite" version in 2006, the Nintendo DS began to have several features that we would later find in future consoles from this major video game and consoles company.

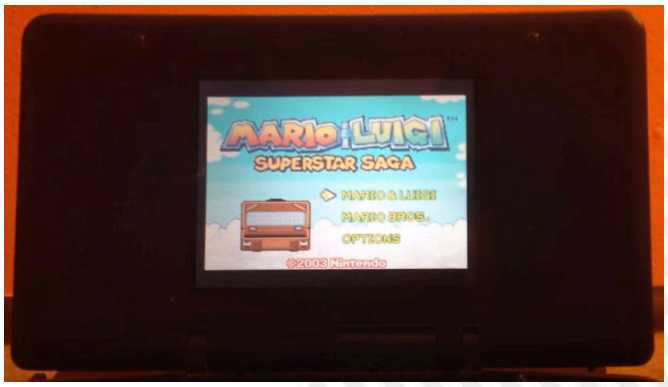
On the right of the photo you can see the "Fat" version while on the left is the "skinny" Lite version.

The main differences between the 2 versions are as follows. The DS Lite is:
- about 42% smaller and 21% lighter while retaining all the features of the "normal" DS
- its backlight is always on and has 4 adjustable brightness levels
- aesthetically, the case is brighter, the speakers take up less space, and the position of buttons and microphone has been changed
- the touch stylus got bigger
- because of the smaller size, the GBA cartridges protrude from the Lite system; therefore, in order to give a feeling of continuity of profiles, a cartridge is included to act as a "filler" when there are no GBA games inserted in the appropriate slot.

Its product code is NTR-001 (USG-001 for the Lite version) and is a contraction of its development code name "Nitro"; the meaning of the suffix DS stands for "Developers System" but also for "Dual Screen."

As Nintendo has accustomed us, this console is also backward compatible with games from its portable predecessor, the GBA, and booting can be done either from an NDS cartridge, a GBA cartridge, Firmware, or via WiFi. When a GBA cartridge (32pin) is inserted, the NDS automatically swtcha into GBA Mode and becomes "almost" like a real GBA.

The differences with a real GBA are as follows:
- Does not support GB/GBC cartridges
- The undocumented register Internal Memory Control (Port 800h) is not supported so NDS does not allow overclocked RAM
- there is no link port (thus no support for hardware link port) consequently GBA games can only be used in 1 player mode
- the audio frequency seems to go from 16.78MHz to 16.76MHz, so slightly slower than the original GBA (imperceptible difference but apparently documented)
- In the BIOS GBA changes a byte from the value 00 to 01 causing the checksum of the same to change
- The GBA logo can be made to appear in either the upper or lower screen (selectable option in the start menu)
- The GBA game screen has black bands on the side as in the picture below:

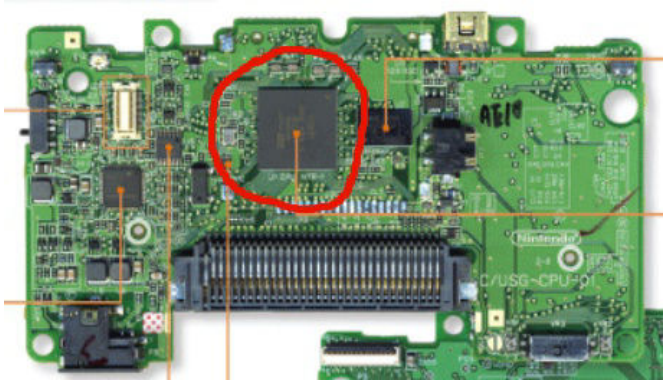


The engine of the DS is an ARM9 processor that cooperates with an ARM7 probably for reasons related to easier backward compatibility with the GBA, which is precisely equipped with an ARM7.

THE PROTECTIONS

BIOS

The BIOS of the ARM7 (called NDS7), about 45KB in size, is divided into 2 regions: the first one that is excluded from being read by writing the 1024h register just before reading the game cartridge as for the GBA and another one that is always active.



The protected part contains the cartridge KEY1 encryption seed used to encrypt/decrypt data stored on cartridge. In addition, the BIOS own functions have software protection that prevents them from reading the protected area.

The system to read the protected data is similar to that used for GBA and that is to use hardware tricks to read the memory before it is protected; alternatively, 2 opcodes (THUMB opcodes 05ECh and 05EEh) capable of bypassing this constraint were discovered.

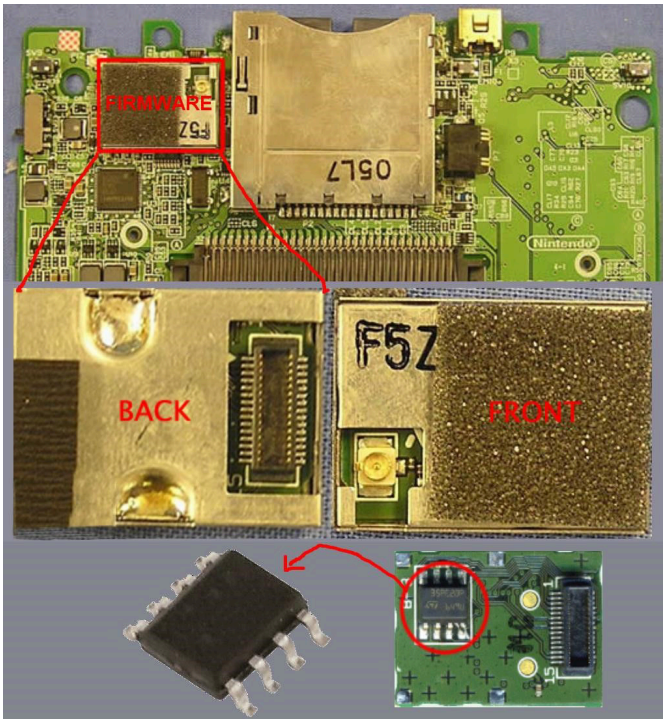
The BIOS of the ARM9 (NDS9), about 65KB in size, is not protected from being read by either hardware or software.

The BIOS checks the cartridge header (header ranging from offset 0x0000 to 0x0200), in particular it checks the Nintendo logo checksum stored at offset 0x15C (checksum calculated with CRC-16 algortim from offset 0x00C0 to offset 0x015B) and corresponds to a fixed value = CF56
The firmware later will also check the bytes of the logo itself (the Nintendo logo goes from offset 0x00C0 to offset 0x015B).

THE FIRMWARE

The firmware is stored inside an NVRAM chip that has different sizes depending on the type of console and bundle:
ST M45PE20 - 256 KBytes (Nintendo DS)
ST M35PE20 - 256 KBytes (Nintendo DS-Lite)
ST M25PE40 - 512 KBytes (iQue DS, Chinese conc aracteries)
ST 45PE40V6 - 512 KBytes (DS Zelda, NTR-AZEP-0)
ST 45PE80V6 - 1024 Kbytes (e.g., Spirit Tracks, NTR-BKIP)
Sanyo LE25FW203T - 256 KBytes (Mariokart backup)

In each case the "core" is 256KB and is divided into 5 encrypted and compressed sections:
part1 - ARM9 BOOT CODE
part2 - ARM7 BOOT CODE
part3 - ARM9 GUI CODE
part4 - ARM7 GUI CODE
part5 - DATA/GFX
which are verified with checksums. The coder Chishm was able to reverse engineer the compression/encryption and developed a tool to extract the various parts of the firmware; the tool is called Fwunpacker. There is also a tool called CFW Suite to process a firmware and modify its GUI.



The ability to act at such a low level in the system also allowed the creation of a virus that could brick the console, apparently functioning only if you had done the softmod, showing a brick wall in the dual screen at startup:

It is, however, possible to do a "hot swap" of the chip containing the firmware, thus allowing you to recover brikked consoles by flashing a valid firmware with FlashMe.

PROPRIETARY FORMAT CARTRIDGES

The cartridges used by the DS have a proprietary format with 17 PINs and have product code NTR-005; there is a variant with infrared hardware support with product code NTR-031:

As with the GBA, clones (real on the right, fake on the left) also exist for these cartridges:

For more details on how to recognize them go here.

CRIPTED DATA

Data on cartridge can be encrypted.

Specifically, in the 0x4000-0x7FFF range of a cartridge there is a "SECURE AREA" accessible only to the ARM9 and decryptable by it through encrypted commands using the KEY1 seed; not all cartridges have the secure area; once decrypted, only the first 8 bytes (which contain the Secure Area ID) of this range are checked; if the check is correct the cartridge loading continues.

Games loaded via WLAN do not have the secure area and games that do have it cannot be started via WLAN.

The encryption algorithm is based on the "Blowfish Encryption Algorithm" devised by Bruce Schneier in 1993. However, cartridges are dumpable through specific SPI protocol commands; the dumpable chips of a cartridge are as follows:

Type / Total Size / Page Size / Chip-Example / Game-Example.

EEPROM 0.5K bytes 16 bytes ST M95040-W (eg. Metroid Demo)

EEPROM 8K bytes 32 bytes ST M95640-W (eg. Super Mario DS)

EEPROM 64K bytes 128 bytes ST M95512-W (eg. Downhill Jam)

FLASH 256K bytes 256 bytes ST M45PE20 (eg. Skateland)

FLASH 256K bytes Sanyo LE25FW203T (eg. Mariokart)

FLASH 512K bytes 256 bytes ST M25PE40? (eg. which/any games?).

FLASH 512K bytes ST 45PE40V6 (eg. DS Zelda, NTR-AZEP-0)

FLASH 1024K bytes ST 45PE80V6 (eg. Spirit Tracks, NTR-BKIP)

FLASH 8192K bytes MX25L6445EZNI-10G (Art Academy only, TWL-VAAV)

FRAM 8K bytes No limit ? (eg. which/any games?)

FRAM 32K bytes No limit Ramtr

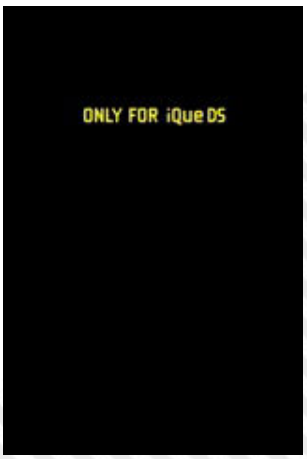
Note: FRAMs are RAM memories that hold data permanently without needing a battery (as opposed to SRAMs) and are faster than EEPROMs, with which they are still backward compatible.

Once recognition has taken place via KEY1-encrypted commands, data is exchanged through another algorithm called KEY2 based on 2 registers of 39 bits, of XORs and shifts. Note that ALL dumps of NDS cartridges found online have the header decrypted.

REGION CHECK

The console has no region check.

Instead, iQue DS game cartridges contain a feature that checks the console's region and if they do not find it, they will not start the game (such protection is therefore in the game, not the console).



To remedy the problem, in DS consoles, the offsed 0x1D of the ROM must be edited and the value 80 changed to 00:

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	4E	49	4E	54	45	4E	44	4F	47	53	00	00	41	33	36	43	NINTENDOGS..A36C
00000010	30	31	00	00	08	00	00	00	00	00	00	00	00	00	00	00	01..............
00000020	00	40	00	00	00	08	00	02	00	00	00	02	D8	C0	0B	00	.@..........ØÀ..

SYSTEMS FOR EXECUTING ARBITRARY CODE

FIRMWARE V1 to V3

Before the encryption algorithm was cracked in 2006, various systems existed to boot unencrypted code that took advantage of several missed checks by the console firmware:

PassMe

It is a hardware that interposes itself between the console and the cartridge in the GBA slot (SLOT-2), reads the ID from the latter tricking the system into thinking that the inserted flash cartridge is original. It was the basis for all software modifications to the console.

WiFiMe

The DS has the ability to connect via WiFi and even send games via WiFi!

In fact, the DS Download Play service in the V1 firmware allows applications to be sent, and a flaw in the management of the original RSA signature of that system allowed hackers to create WiFiMe, a program to boot homebrew. In detail, the first version of the Download Play title uses an execution address from a header that is not part of the code and data digital signature calculation. WiFiMe was extracted from Super Mario 64 DS with the modified execution address in the GBA ROM space, but it could have been any DS Download Play program.

Wireless Multiboot (WMB)

This Windows application created by coder FireFly requires d a PCI or PCMCIA network card with Ralink chipset; it can send small Nintendo-signed programs to the DS Download Play.

FlashMe

This system overwrites the NDS firmware allowing immediate execution of DS code even in the GBA ROM space, removing the digital signature check for DS Download Play and thus allowing homebrew to be sent. We are looking at one of the very first softmods for consoles.

FIRMWARE V4

Since firmware version V4, the bugs that allowed the aforementioned exploits have been patched but this has not deterred hackers who have gone ahead in the classic cat-and-mouse chase.

How to check the firmware version of the NDS:

We rely on the behavior of the PictoChart application: you start the DS with a game on, start PictoChart, and pull out the cartridge with the console still on; a color of the screens will appear that is different for each firmware version (this technique is also called "Pink Screen of Death" or "TakeMeOut"):

v1: Pictochat hangs/no color is shown

v2: two blue-gray screens

v3/iQue: two dark green screens

v4: two golden yellow screens

v5: two magenta screens (DS lite come out with this version, along with some pink DSs):

v6: two dark blue screens

v7: the system does not crash (only in the gold version of the Japanese DS Lite limited edition).

PassMe2

Like PassMe but exploiting a different DS BIOS bug by using specific games (a list of supported ones is available here)

FlashMe v5

In this version the program code is scaled down and written in protected areas of the firmware that cannot be rewritten by any original game. Previous versions of FlashMe were in fact written in areas that were supposed to be "unused" discovering later that this was not the case.

NoPass

From the moment in 2006 when the cartridge encryption algorithm was reversed by Martin Korth (author of the No$GBA emulator as well as the one who also first managed to dump the NDS bios) it was possible to execute code even from cartridges inserted in SLOT-1 (the one for NDS games); this method was named "NoPass."

NoPass Cartridges

Thus cartridges began to come out that did not require an original game to run arbitrary code because they were coded just like the original cartridges such as Datel's Max Media Launcher, which you can see in the photo below in comparison with a more dated PassMe:

Next came the dozens and dozens of R4s, M3s, AceKards, etc. each with their own boot firmwares some of them even reversed like the R4 one again thanks to Chishm and its r4crypt tool (there is also injektor2 1.08 - latest version 1.09 apparently unobtainable).

SOUNDHAX

In late 2014, the reverser/dev/coder Smealum showed a video of the audio signal-based hack for the game Bangai-o-Spirit; the game allows levels to be transferred via audio: by reversing the code, it was possible to buffer overrun and execute arbitrary code simply by sending a .WAV to play (you can also see it at work in the DeSmuME emulator).

CONCLUSIONS

As you can see things are starting to get very interesting in the world of console hacking especially because for the first time we are starting to have the possibility to arbitrarily write into the internal memory of a console so as to have a sort of very first and rudimentary custom firmware. We also saw how countermeasures were put in place for the first time to limit the damage caused by the bugs uncovered by the reversers: the nowadays dreaded FIRMWARE UPDATES!

Note that the console firmware could not update itself either via WiFi or via cartridge so when buying the console we went a little "by luck" when the "Pink Screen" method explained above was not yet known. Although very late there are also exploits that exploit flaws in games such as the SoundHax exploit.

I apologize if I may have seemed hasty on some topics but the DS was a rather unfamiliar console to me and I had to study it practically from scratch. If you have any questions or clarifications please write to me.

The next installment will discuss the evolution of this Nintendo in terms of both hardware and security.

WARNING: Disclaimer

The information contained in this article is for informational purposes only. This documentation is not guaranteed to be error-free. If this information is used to modify your hardware, it is your responsibility to take all necessary emergency, backup, redundancy, and other measures to ensure its safe use. RetroMagazine World disclaims all liability for any damages caused by the use of the information in this article.