nxtOSEK Hack (Updated: June 2008)

If you see LEGO MINDSTORMS NXT as an embedded system, you may want to port your RTOS or may want to enhance existing features or may replace everything... it was exactly me. I would like to write all my knowledge about the NXT and nxtOSEK which I could learn through this project. I will update this section step by step, so stay tuned.

Source code organization

Name	Directory path	Description
TOPPERS ATK	\toppers_osek	TOPPERS ATK is the heart of nxtOSEK. Unfortunately, it is hard to read the source code for non Japanese native because of source code comments written in Japanese. However, NXT porting part might be useful for those who consider to port other RTOS to the NXT. Essential part of TOPPRS ATK porting is in \toppers_osek\config \at91sam7s-gnu\cpu_support.S and irq.s. These assembly code have comments in English and written by Prof. Masaaki Mizuno.
leJOS device driver	\lejos_nxj\src\nxtvm\platform\nxt	NXJ device driver code is written in C and Assembly and these are simpler and more comprehensive compared to LEGO standard fw source code. leJOS team has updated device driver release by release, so device source code also will be updated.
PC console programs	\lejos_nxj\src\libnxt	PC console programs (bisflash, appflash and ramboot) are based on LibUsb and David Anderson's libNXT. build.sh is the shell script to build executables to be stored in \bin directory and it requires host GCC (not GNU ARM).
Sample programs	\samples	nxtway, nxtway_gs and nxtremocon samples are not hand written code. These C programs are automatically generated from Simulink models.
C++ API	\c++\src	All C++ API is written by Prof. Rober Kramer. the most impressive API is sleep.cc. if sleep API was executed, a running Task is turned into sleep mode until the specified time duration is elasped. While that period, other Tasks in ready queue can run. This is a nice feature which uses OSEK event mechanism.
ECRobot API	\ecrobot	main C function for nxtOSEK application is in ecrobot.c. In main function, TOPPERS ATK system call (StartOS) is called. Other ecrobot_*.c files are TOPPERS ATK independent, so these source code might be reused for other RTOS porting.
NXT BIOS	\ecrobot\bios	NXT BIOS uses AT91SAM7S embedded flash controller feature to flash nxtOSEK application. NXT BIOS code is independent from TOPPERS ATK and ECRobot API, so it might be reused for other RTOS porting.

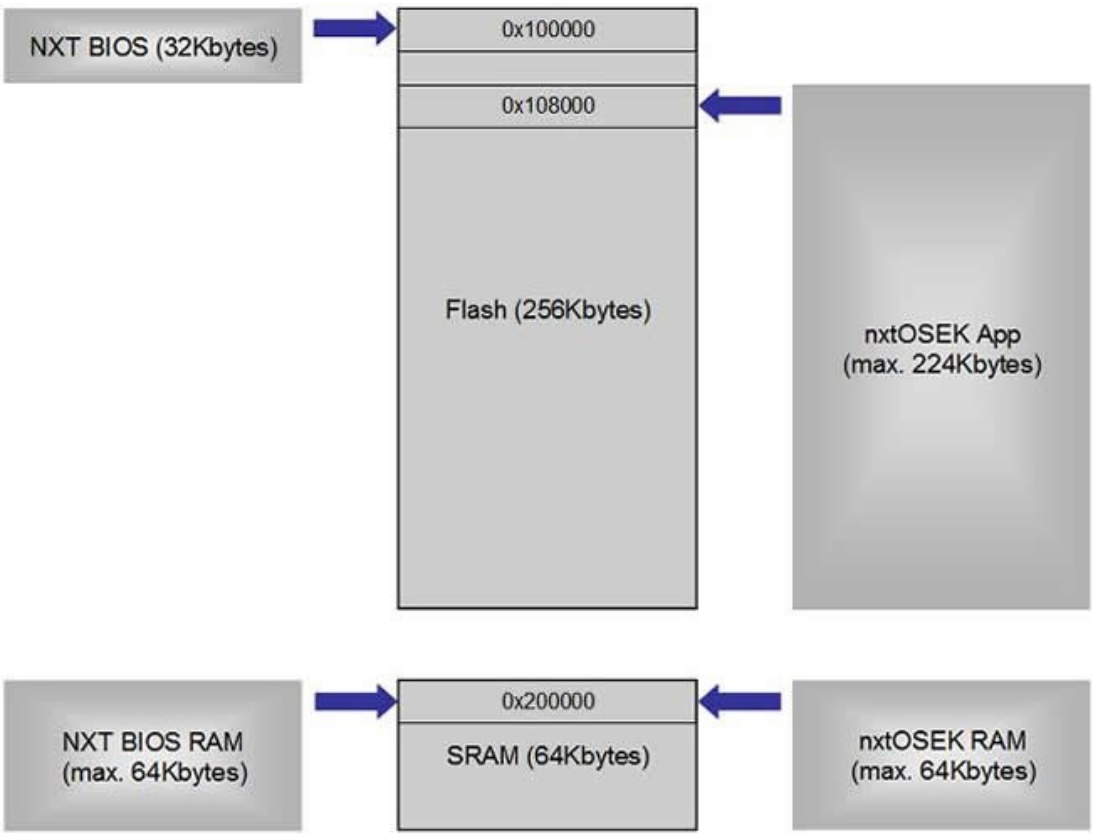
Memory layout

NXT BIOS (nxt_bios_rom.bin) is uploaded to the beginning of Flash memory (0x100000) by biosflash.exe. The size of nxt_bios_rom.bin file is 14Kbytes, however 32Kbytes memory space is preserved for flash record and for feature enhancement.
nxtOSEK Flash application (*_app_rom.bin) includes platform software (leJOS device driver, TOPPERS ATK RTOS, ECRobot and C++ API) is uploaded to the Flash memory with 32Kbytes address offset by appflash.exe. NXT BIOS and a nxtOSEK Flash application use same SRAM memory space. It means that NXT BIOS and a nxtOSEK Flash application can not run together.

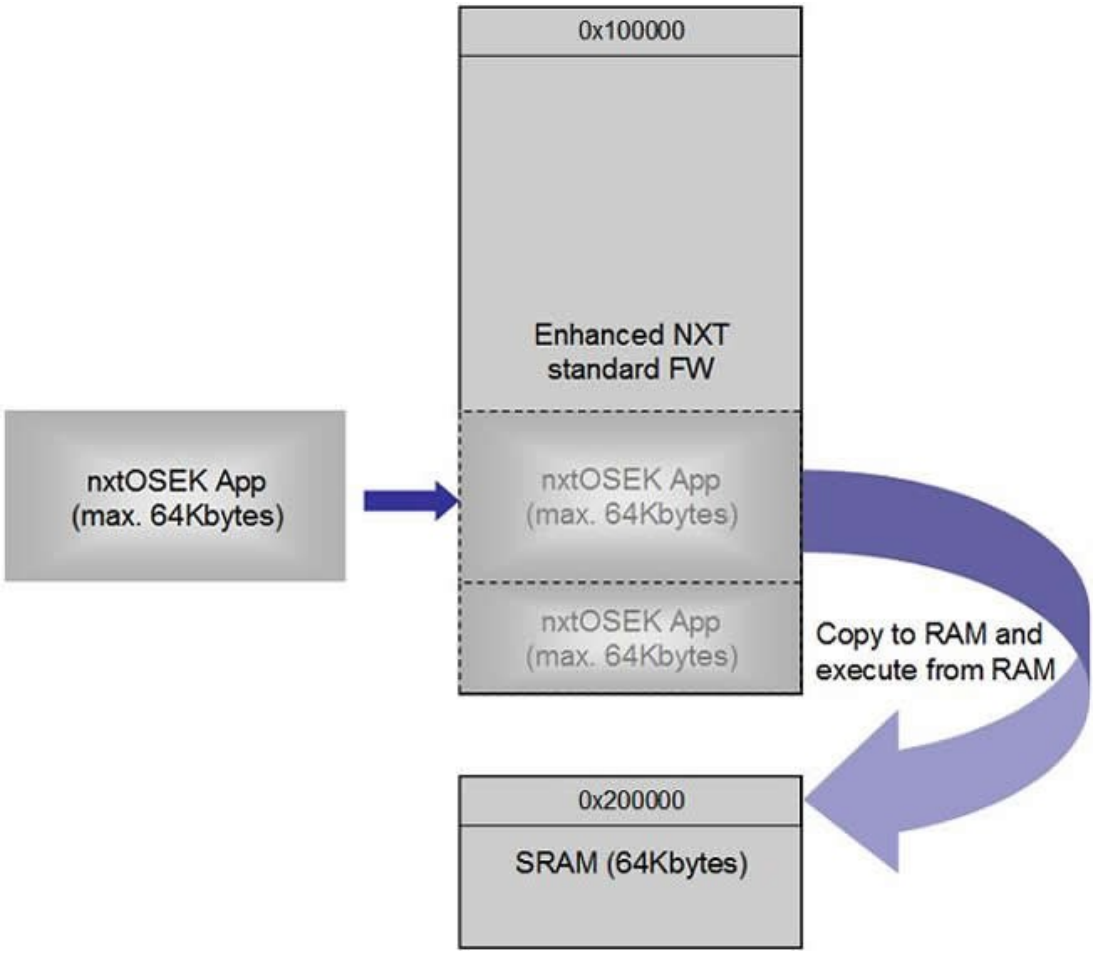
nxtOSEK SRAM application (*_samba_ram.bin) is uploaded to the beginning of SRAM memory (0x200000) by ramboot.exe. It means that all program (includes code, read-only data, data with initializer, bss data and stack) is allocated in SRAM. Hence, the maximum program size is limited to 64Kbytes.

nxtOSEK application is statically linked with platform software, hence only platform software which used in application program is linked and

uploaded to the NXT (e.g. if you did not use Ultrasonic Sensor API, Ultrasonic Sensor related objects would not be loaded into the NXT).
This is a significant difference compared to other VM based NXT firmware. In case of VM based firmare, all platform software includes VM and API have to be located in Flash memory regardless of in use or not.



Since nxtOSEK 2.02, nxtOSEK has supported John Hansen's enhanced NXT standard firmware. The memory layout for ARM7 native programs (nxtOSEK applications) in the enhanced NXT standard firmware is a little bit complicated compared to the above cases. nxtOSEK application is built for RAM execution, however, it is uploaded into Flash, thus before executing the application, whole application program is copied to RAM, then executed. This mechanism restricts the size of a nxtOSEK application up to 64Kbytes. The enhanced NXT standard firmware checks the binary header information of an application program to judge whether a NXT standard program or an ARM7 native program. Once a nxtOSEK application program is executed from RAM, it overwrites RAM data of NXT standard firmware, thus it is not possible to back to the NXT firmware when you terminated the nxtOSEK application program. This is a restriction of ARM7 native execution in the enhanced NXT standard firmware.


0x100000
Enhanced NXT
standard FW
nxtOSEK App
(max. 64Kbytes)
nxtOSEK App
(max. 64Kbytes)
nxtOSEK App
(max. 64Kbytes)
Copy to RAM and
execute from RAM
0x200000
SRAM (64Kbytes)

Home