# N8x0 support in linux-omap

Current kernel: 2.6.31-rc1-omap1

## Startup plan

- find optimal debugging techniques and real HW/qemu split;
- make minimal working/debuggable configuration;
- compile domain glossary.

## Debug interfaces

## Subsystems' status

## Debugging in qemu

Need CONFIG_DEBUG_INFO=y in kernel configuration for symbolic debugging to work.

### Kernel command line parameters

- initcall_debug=1 – make kernel print all _init function calls during kernel_init;

### Magic numbers

- 0x80000000 – here the kernel is loaded
- 0x80008000 – here we jump after decompression (.text.head that has VA of 0xc0008000 has PA 0x80008000 until MMU is active)
- 0xc0026000 – 'arm-linux-gnu-objdump -x vmlinux' says that .text starts here

### Script for gdb session

```
target remote 127.0.0.1:1234
break *0x80008000
```

To debug compression-related stuff from the very beginning (start, arch/arm/boot/compressed/head.S) till start_kernel:

```
add-symbol-file ~/ws/osll/omap/20090610/linux-
omap-2.6/arch/arm/boot/compressed/vmlinux 0x80000000
```

To debug kernel from stext (arch/arm/kernel/head.S) until MMU is active:

```
add-symbol-file ~/ws/osll/omap/20090610/linux-omap-2.6/vmlinux 0x80026000 -s
.text.head 0x80008000
```

To debug kernel from start_kernel (init/main.c):

```
add-symbol-file ~/ws/osll/omap/20090610/linux-omap-2.6/vmlinux 0xc0026000
```

### "Blank screen" debugging

Whatever happens, ^C breaks into the running kernel. If the screen is blank, dmesg-like log may be viewed through

```
x/10000s log_buf
```

Or even through

```
dump memory kmsg.log *log_buf *log_buf+10000
```

## Debugging on real HW