# [[si:lab:lab1]]

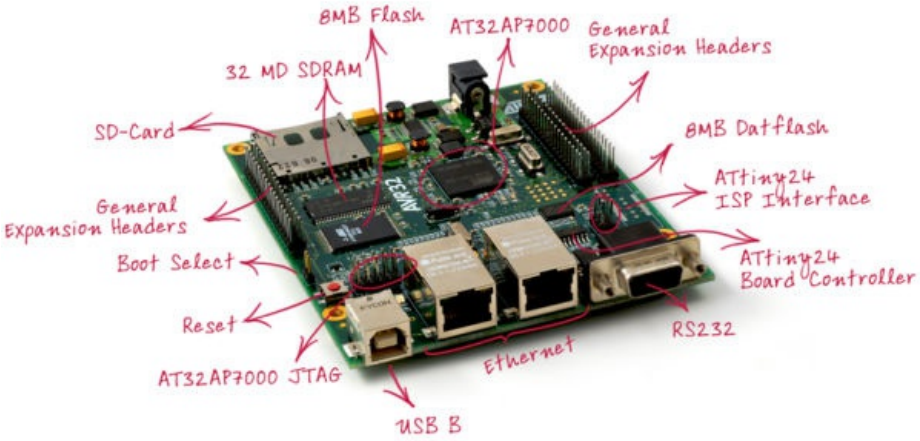


## Laboratorul 1 - Introducere

Table of Contents



### Network Gateway

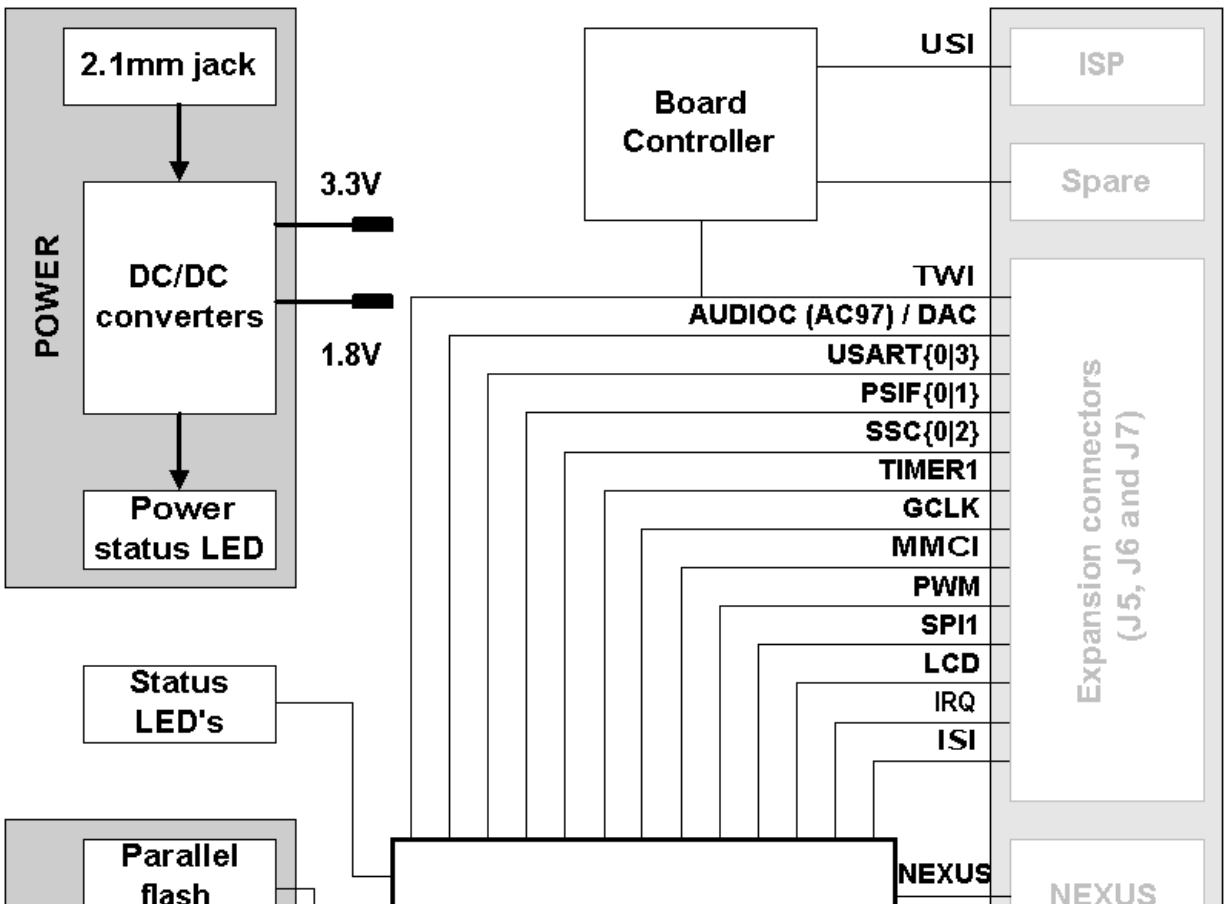


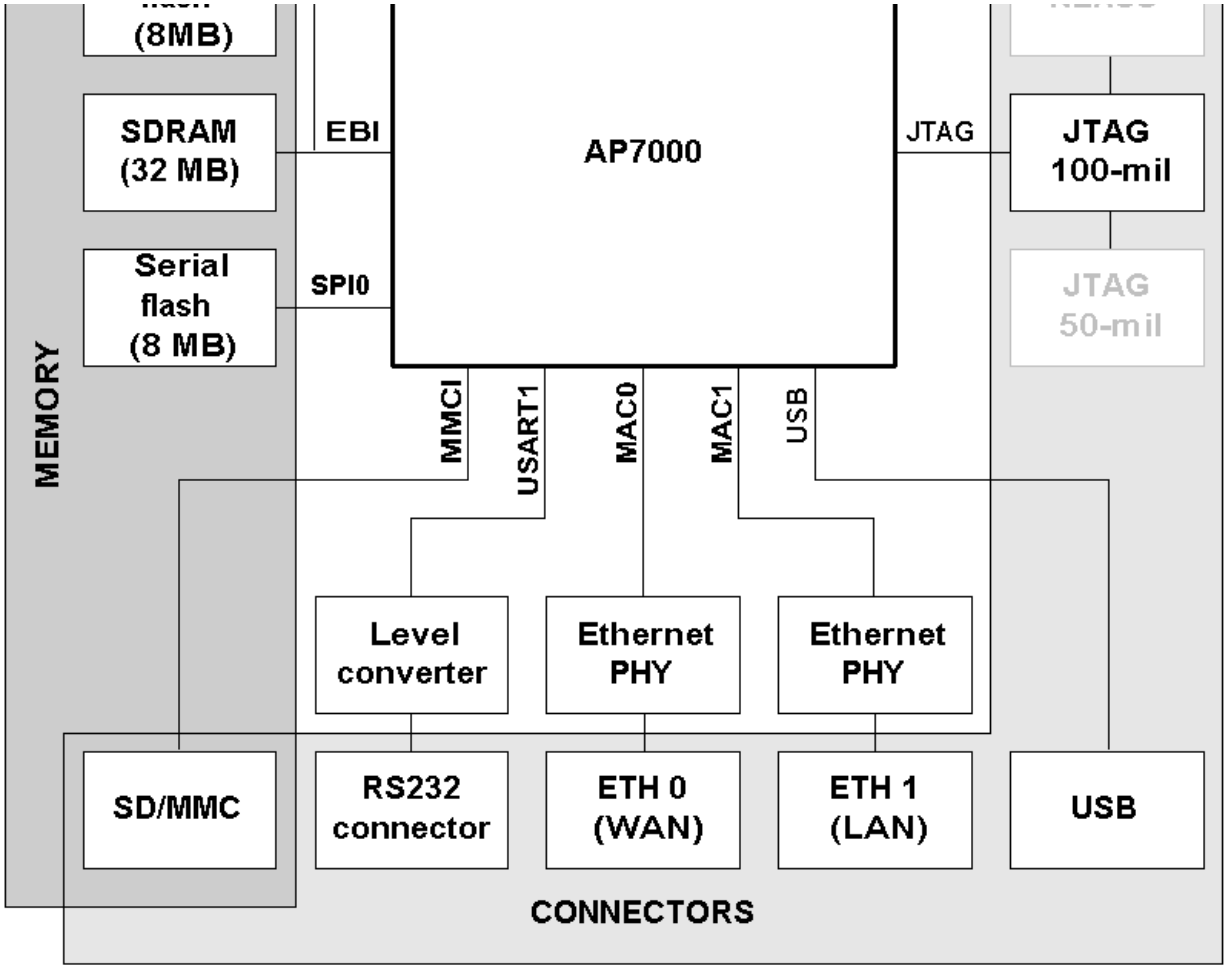

Network Gateway (NGW) este un sistem de calcul embedded complet functional care demonstreaza capabilitatile procesorului pe 32 de biti AP7000 din familia AVR32 de la Atmel.

Placa este dotata cu o memorie flash de 16MB si SDRAM de 32MB. Memoria poate fi extinsa prin conectarea unui card SD/MMC intr-un slot alocat. Conectarea la retea este posibila prin oricare dintre cele doua porturi Ethernet sau prin portul USB B (device). O consola este disponibila pe portul RS232 de pe placa. Prin cele 3 header-e de expansiune ale placii se pot interfata mai multe periferice ale procesorului, cum ar fi: UART, PS/2, ABDAC (convertor digital/analogic), ISI (Image Sensor Interface - pentru camera video), LCDC (periferic pentru LCD), EBI (magistrala externa), si multe altele.

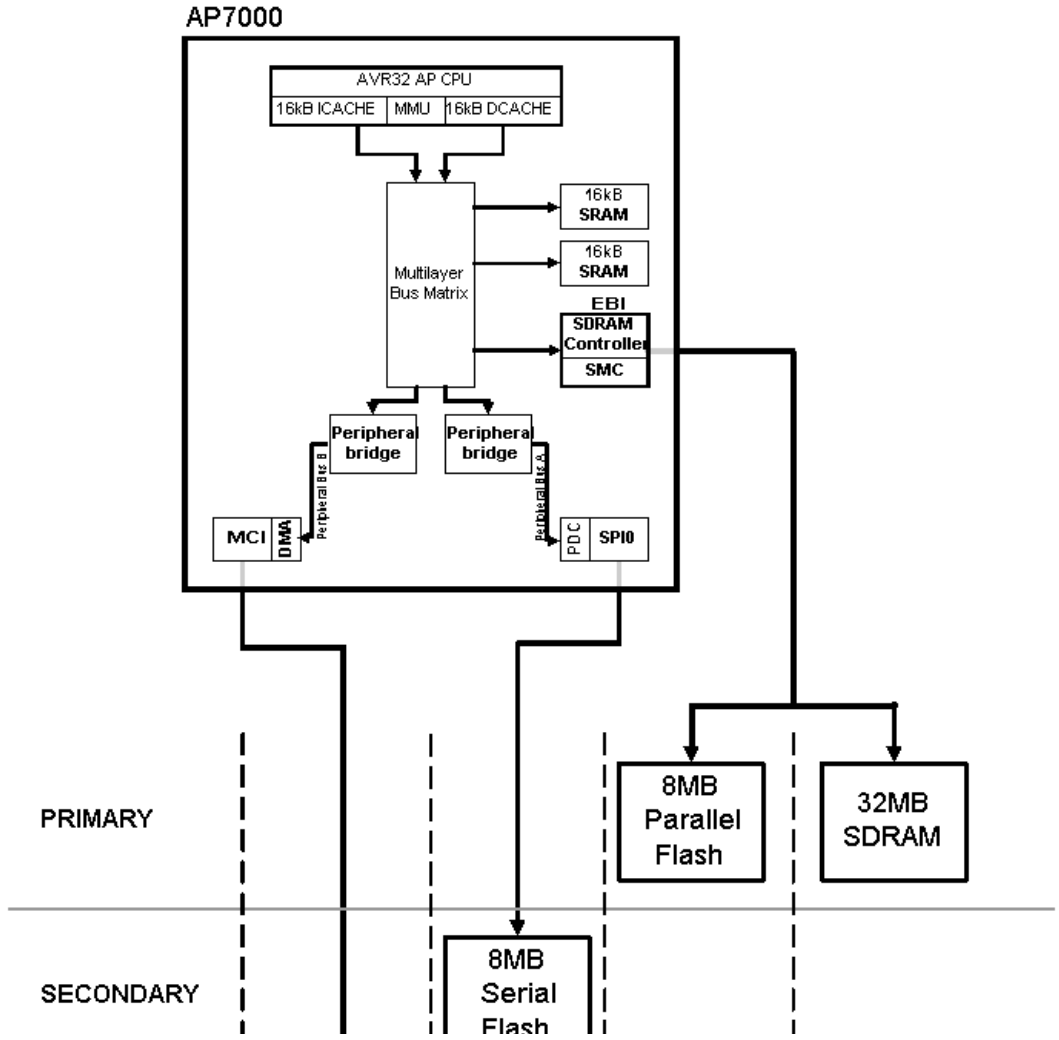
Bootloader-ul folosit in acest sistem (U-Boot) ofera posibilitatea incarcarii kernel/sistemului de pe retea, prin linia seriala, de pe flash sau de pe cardul MMC. Configurari ale bootloader-ului se pot face atat la programarea initiala cat si printr-o consola (pe RS232). Odata incarcat, sistemul Linux de pe NGW demonstreaza capabilitatile complete ale procesorului in materie de comunicatii la viteze inalte.

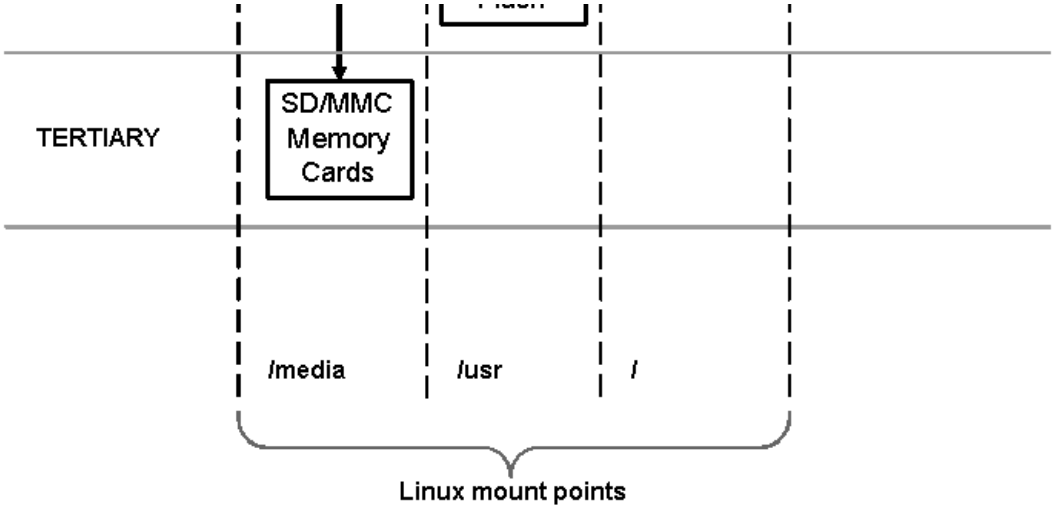
#### Schema-bloc a sistemului






**Layout-ul memoriei**






**SDRAM**

Memoria SDRAM de pe Network Gateway este de 256Mbit (4M x 16biti x 4bancuri) sau 32MB si este legata de AP7000 printr-un bus adresa/date de 16 biti.

**Flash Paralel**

Memoria de boot are o capacitate de 8MB si este adresata pe 16biti de catre AP7000. Ea vine pre-incarcata cu U-Boot (softul bootloader) si un sistem de operare Linux Embedded.

**Flash Serial**

Memoria flash seriala (conectata la interfata SPI) este pre-incarcata cu sistemul de fisiere Linux.

**Disk Space Layout**

Comanda df arata intreg spatiul de memorie disponibil si gradul lui de ocupare.

```
/ $ df
Filesystem         1k-blocks Used     Available Use%        Mounted on
/dev/root          8000      6688     1312      84%        /                  8M Parallel Flash
dev                15468     0        15468     0%         /dev               16M RAM Drive
tmp                15468     0        15468     0%          /tmp              16M RAM Drive
run                15468     24       15444     0%          /var/run          16M RAM Drive
samba              15468     236      15232     2%         /var/lib/samba     16M RAM Drive
log                15468     68       15400     0%         /var/log           16M RAM Drive
/dev/mmcblk0p1     31075     1        31074     0%         /media/mmcblk0p1 SD/MMC Card
/dev/mtdblock3     8448      8180     268       97%         /usr              8M Serial Flash
```

/etc si /home sunt montate pe Flash-ul Paralel, iar fisierele log sunt puse pe RAM Drive.

## Cross-compilare

Pentru a dezvolta programe pe o astfel de platforma, avem doua posibilitati:

1. Prima posibilitate este sa avem un compilator (impreuna cu restul de utilitare necesare procesului de compilare si linkare - *toolchain*) direct pe sistemul pe care vrem sa folosim aplicatia. Aceasta este cale folosita de obicei in dezvoltarea aplicatiilor pe PC, unde programul va rula pe acelasi sistem pe care este dezvoltat. Intr-un sistem embedded insa ne lovim de constrangeri de spatiu si putere de calcul. In primul rand, toolchain-ul impreuna cu fisierele intermediare din timpul compilarii vor ocupa spatiu pe care poate nu il avem si in al doilea rand, pe un sistem cu capacitati de calcul limitate compilarea ar putea dura mult mai mult decat ne permitem.
2. A doua posibilitate este sa folosim un toolchain de cross-compilare. Acesta ruleaza pe un sistem de dezvoltare (diferit de sistemul embedded - numit host/gazda), sa zicem un PC (ex:arhitectura x86) , si creeaza executabile si biblioteci dinamice pentru sistemul embedded (in cazul nostru cu arhitectura AVR32, denumit sistem target/tinta).

In laborator se va folosi un toolchain x86→avr32-linux. Pentru a distinge acest toolchain de cel nativ sistemului, numele compilatorului incepe cu arhitectura, de ex: avr32-linux-gcc

## Hello World

Vom rula acum un exemplu de aplicatie pe NGW:

- Codul sursa, in arhiva laboratorului este un (aproape) tipic hello world.
- Makefile-ul contine modificarile necesare pentru cross-compilare

```
CC=/home/student/build_avr32/staging_dir/usr/bin/avr32-linux-gcc
CFLAGS=-I/home/student/build_avr32/staging_dir/usr/include -L/home/student/build_avr32/staging_dir/usr/lib
hello: hello.c
   $(CC) $(CFLAGS) -c -o hello.o hello.c
   $(CC) $(CFLAGS) -o hello hello.o
clean:
```

```
rm -rf *.o hello
```

sau varianta foarte scurta

```
CC=/home/student/build_avr32/staging_dir/usr/bin/avr32-linux-gcc
all:hello
```

Observati ca se modifica variabila de mediu CC pentru a folosi avr32-linux-gcc din path-ul dat. In cazul programelor cu dependente, este nevoie si de path-ul de include-uri si de biblioteci, care vor trebui urmarite cu atentie sa fie cele pentru arhitectura respectiva si nu cele native!

## Debugging hello world

Asemanator cazului cu compilatorul, vom avea un cross-debugger pe statia de dezvoltare care se va conecta sistemul target prin retea sau conexiune seriala pentru a face debugging.

### Setup pe target

Aplicatia care intermediaza acest mod de lucru pe sistemul de target este gdbserver:

```
gdbserver :4242 hello
```

asteapta conexiuni pe 4242 pentru a face debugging programului hello.

### Setup pe host

Cross-debugger ar nevoie de cel putin doua informatii:

1. locatia unde va rula programul (va rula pe sistemul embedded, nu pe cel de dezvoltare!)
2. informatii despre simbolii programului

*ATENTIE*: pentru 2) trebuie adaugat flag-ul de debugging in compilator (-g)

```
avr32-linux-gdb -q hello
(gdb) target remote 10.0.0.1:4242
...
```

```
In cazul unei conexiuni seriale, comanda in gdb va fi target remote /dev/tty..
```

## Quick Network copy How-To

### Copierea prin SSH

```
scp <fisier> <user>@<adresa>:<cale>
```

### Copierea prin SAMBA

NGW-ul ruleaza implicit un server de samba care se numeste netdisk si se gaseste in /media. Pentru a copia pe acest share trebuie intai montat local:

```
smbmount  //10.0.0.1/netdisk board -o user=root
```

sau

```
mount -t cifs -o user=root \\\\10.0.0.1/netdisk board
```

apoi

```
cp fisier board/
```

### Copierea pe un root montat pe retea prin NFS

In cazul in care sistemul embedded ruleaza un sistem de fisiere incarcat peste retea, copierea un fisier pentru a fi disponibil pe sistem se reduce la o copierea in share-ul de pe care este montat sistemul de fisiere.

De exemplu, daca root-ul sistemului embedded este de fapt /home/student/target pe sistemul de dezvoltare, atunci este de ajuns

```
 cp fisier /home/andrei/target/subdir
```

## Referinte

- Wiki-ul AVRFreaks dedicat AVR32
- Application Notes de la Atmel