

MMnet1000 System Linux

Przewodnik
Użytkownika

Wersja 20090601

PROPOX®
Many ideas one solution

Spis treści

1. Wstęp	3
2. Pierwsze uruchomienie	4
3. Sekwencja startowa i komponenty systemu.....	7
4. Rozmieszczenie komponentów w pamięci NAND Flash.....	12
5. Aktualizacja oprogramowania	13
6. Obsługa peryferiów	15
7. Środowisko programistyczne i programy przykładowe	23
8. Kompilacja systemu.....	24
9. FAQ.....	26
10. Przydatne linki	27
11. Licencja, gwarancja i wsparcie techniczne	27
12. Załączniki	28

1. Wstęp

W dokumencie opisano system Linux i oprogramowanie dostarczane wraz z modułami MMnet1001 i MMnet1002.

Poniższy opis dotyczy wersji oprogramowania 20090601. Jeśli jesteś posiadaczem starszej wersji, prosimy wykonać aktualizację. Najnowszą wersję można pobrać pod adresem: <http://www.propox.com/download/software/MMnet1000-CD-20090601.zip>

System w wersji 20090601 bazuje na:

- U-Boot-2009.01
- linux-2.6.29.3
- OpenWrt KAMIKAZE r13340

Przed rozpoczęciem pracy prosimy zapoznać się również z dokumentacją do modułów dotyczącą sprzętu:

- MMnet1001: http://www.propox.com/download/docs/MMnet1001_pl.pdf
- MMnet1002: http://www.propox.com/download/docs/MMnet1002_pl.pdf

2. Pierwsze uruchomienie

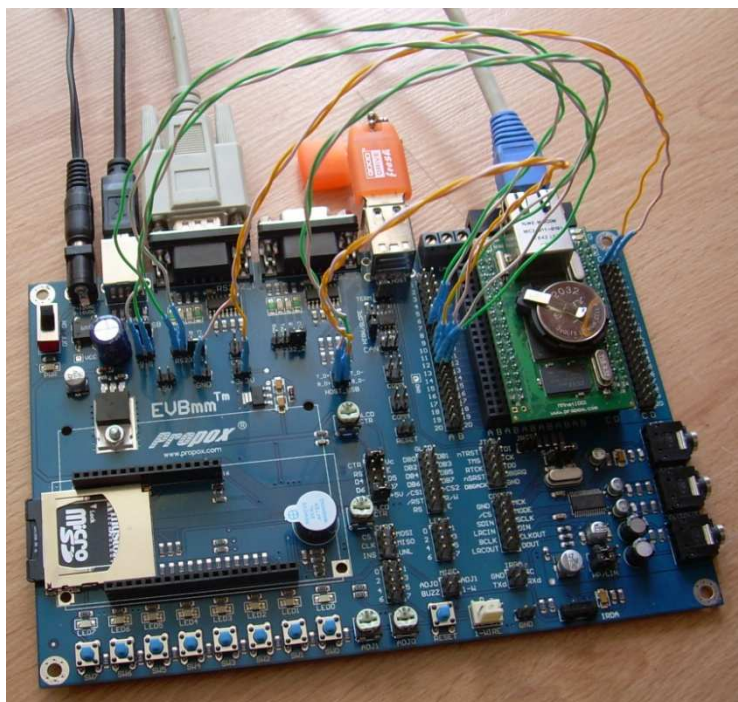
2.1. Podłączenie

A. Podłączenie modułu MMnet1001

1. Podłącz port RS232 DBGU modułu do komputera PC za pośrednictwem konwertera poziomów (można do tego użyć portu RS232 na płycie ewaluacyjnej EVBmmTm).
2. Podłącz do modułu zasilanie 3.3V DC o wydajności ok. 500mA. Zwróć szczególną uwagę na to połączenie, moduł nie jest zabezpieczony przed odwrotnym podłączeniem zasilania ani przed zbyt dużym napięciem.
3. Podłącz moduł do lokalnej sieci Ethernet za pomocą przewodu z przeplotem lub bez przeplotu (podłączenie jest opcjonalne).

Przykład połączeń na płycie EVBmmTm:

Interfejs	Płyta	Podstawka pod moduł	Połączenie
Zasilanie	+3.3V	C1	Konieczne
	GND	D1	
RS232 – DBGU	RS232_1 RxD	A7	
	RS232_1 TxD	B7	
Reset	RESET	A16	Opcjonalne
USB Device	USB DP	B12	
	USB DM	A12	
USB Host Top	HOST_USB T_D+	B10	
	HOST_USB T_D-	A10	
USB Host Bottom	HOST_USB D_D+	B11	
	HOST_USB D_D-	A11	
RS232 – UART0	RS232_2 RxD	B5	
	RS232_2 TxD	A5	
SD/MMC	CARD CS	TBD.	
	CARD MOSI	TBD.	
	CARD CLK	TBD.	
	CARD MISO	TBD.	
	CARD INS	TBD.	
	CARD UNL	TBD.	



Rysunek 1 Moduł MMnet1001 na płycie EVBmmTm.

B. Podłączenie modułu MMnet1002

1. Podłącz moduł do komputera PC za pomocą przewodu RS232 DB9F-DB9F typu null-terminal (z przeplotem). Zworki na module powinny być w pozycji „DBGU”.
2. Podłącz zasilacz 8 – 24V DC (minus na obudowie złącza) do modułu. Moduł jest zabezpieczony przed odwrotnym podłączeniem zasilania.
3. Podłącz moduł do lokalnej sieci Ethernet za pomocą przewodu z przeplotem lub bez przeplotu (podłączenie jest opcjonalne).

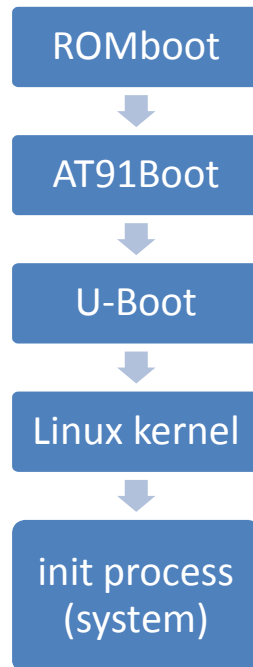


Rysunek 2 Moduł MMnet1002 z podłączoną kartą USB Wi-Fi

3. Sekwencja startowa i komponenty systemu

3.1. Sekwencja startowa

Proces uruchamiania wygląda następująco:



3.2.ROMboot

Program zawarty w pamięci ROM procesora. Dokładny opis znajduje się w dokumentacji procesora w rozdziale 13 „AT91SAM9260 Boot Program”.

W modułach MMnet1001/1002 ROMboot wczytuje do wewnętrznej pamięci RAM program AT91 Boot zawarty w pierwszych 4096 bajtach pamięci NAND Flash i uruchamia go

3.3. AT91Boot

AT91Boot jest bootloaderem pierwszego stopnia. Jego zadaniem jest wstępne zainicjowanie systemu oraz załadowanie bootloadera drugiego stopnia. Kod źródłowy bootloadera dostosowanego do modułów MMnet1001/1002 znajduje się za płytkie CD w archiwum /src/ Bootstrap-v1.10-MMnet1000.tar. Dokumentacja znajduje się wewnątrz archiwum w katalogu doc.

AT91Boot skonfigurowany dla modułów MMnet1001/1002 z pamięcią NAND Flash ładuje program zawarty w pamięci NAND Flash pod adresem 0x00040000 do pamięci SDRAM pod adres 0x23F00000 i uruchamia go.

3.4. U-Boot

U-Boot jest bootloaderem drugiego stopnia. Jego zadaniem jest załadowanie i uruchomienie systemu operacyjnego lub innego programu mającego docelowo pracować na module. Ma on bardzo rozbudowane możliwości, umożliwia załadowanie programu z pamięci NAND Flash, DataFlash, przez Ethernet z serwera TFTP, z zewnętrznej pamięci USB, oraz programowanie pamięci Flash i inne. Dokumentację można znaleźć pod adresem: <http://www.denx.de/wiki/U-Boot/Documentation>. Poniżej przedstawiono listę obsługiwanych komend:

```
U-Boot> help
?      - alias for 'help'
base   - print or set address offset
boot   - boot default, i.e., run 'bootcmd'
bootd  - boot default, i.e., run 'bootcmd'
bootm  - boot application image from memory
bootp  - boot image via network using BOOTP/TFTP protocol
cmp    - memory compare
coninfo - print console devices and information
cp     - memory copy
crc32  - checksum calculation
dhcp   - boot image via network using DHCP/TFTP protocol
echo   - echo args to console
erase  - erase FLASH memory
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls  - list files in a directory (default /)
flinfo - print FLASH memory information
go     - start application at address 'addr'
help   - print online help
imxtract - extract a part of a multi-image
itest  - return true/false on integer compare
loadb  - load binary file over serial line (kermit mode)
loady  - load binary file over serial line (ymodem mode)
loop   - infinite loop on address range
md     - memory display
mm     - memory modify (auto-incrementing)
mtest  - simple RAM test
mw     - memory write (fill)
nand   - NAND sub-system
nboot  - boot from NAND device
nfs    - boot image via network using NFS protocol
nm     - memory modify (constant address)
ping   - send ICMP ECHO_REQUEST to network host
printenv - print environment variables
protect - enable or disable FLASH write protection
rarpboot - boot image via network using RARP/TFTP protocol
reset  - Perform RESET of the CPU
run    - run commands in an environment variable
saveenv - save environment variables to persistent storage
setenv - set environment variables
sleep  - delay execution for some time
tftpboot - boot image via network using TFTP protocol
```



```
usb      - USB sub-system
usbboot  - boot from USB device
version  - print monitor version
```

Aby uzyskać dostęp do konsoli U-Boot-a należy podczas jego uruchamiania wcisnąć dowolny klawisz w konsoli:

```
RomBOOT
>AT91Boot-20081201

U-Boot 2009.01 (Mar 20 2009 - 13:43:24)

DRAM: 64 MB
NAND: 1024 MiB
In:    serial
Out:   serial
Err:   serial
Net:   macb0
macb0: Starting autonegotiation...
macb0: Autonegotiation complete
macb0: link up, 100Mbps full-duplex (lpa: 0x45e1)
Hit any key to stop autoboot:  0
U-Boot>
```

Kod źródłowy bootloadera U-boot przystosowanego do modułów MMnet1001/1002 znajduje się na płycie CD w archiwum /sources/u-boot-2009.01-MMnet1000.tar.gz. Wykorzystana została konfiguracja at91sam9260ek, która została zmodyfikowana do potrzeb modułów.

3.5. Linux Kernel

Na modułach pracuje standardowa wersja 2.6.29.3 jądra Linuxa, która posiada pełne wsparcie dla procesorów AT91SAM9260/AT91SAM9G20. Źródła wersji z dodaną obsługą modułów MMnet1000 znajdują się w archiwum /sources/linux-2.6.29.3-MMnet1000.tar.gz na CD. Dokumentację jądra Linuxa można znaleźć pod adresem <http://www.kernel.org/doc/> oraz po rozpakowaniu źródeł w katalogu Documentation.

Peryferia procesorów AT91SAM9 obsługiwane przez jądro:

- Porty szeregowo
- Ethernet
- USB Host
- USB Device
- MMC / SD
- I2C
- SPI
- SSC
- RTT (as RTC)
- Watchdog
- NAND Flash
- Compact Flash

3.6.System

System oparty jest na dystrybucji OpenWrt (<http://openwrt.org/>). Dokumentację systemu oraz fora dyskusyjne (w języku angielskim) można znaleźć na podanej wyżej stronie.

Standardowo dostępne są następujące komendy i programy:

```
[, [[, addgroup, adduser, adjtimex, ar, arp, arping, ash,
awk, basename, bbconfig, brctl, bunzip2, bzip2, cat,
catv, chat, chgrp, chmod, chown, chpst, chroot, chrt, chvt,
cksum, clear, comm, cp, crond, crontab, cut, date, dd, delgroup,
deluser, df, dhcprelay, diff, dirname, dmesg, dnsd, dpkg,
dpkg-deb, du, dumpleases, echo, egrep, env, envdir, envuidgid,
ether-wake, expr, fakeidentd, false, fetchmail, fgrep, find,
fold, free, ftpget, ftpput, fuser, getty, grep, gunzip,
gzip, halt, head, hexdump, hostid, hostname, httpd, hwclock,
id, ifconfig, ifdown, ifenslave, ifup, inetd, init, inotifyd,
insmod, install, ip, ipaddr, ipcalc, iplink, iproute, iprule,
iptunnel, kill, killall, killall5, klogd, last, length,
less, ln, lock, logger, login, logname, logread, ls, lsmod,
lzmecat, makedevs, md5sum, mdev, mesg, microcom, mkdir,
mkfifo, mknod, mkswap, mktemp, modprobe, more, mount, mountpoint,
mv, nameif, nc, netmsg, netstat, nice, nmeter, nslookup,
openvt, passwd, pgrep, pidof, ping, ping6, pivot_root, pkill,
poweroff, printenv, printf, ps, pscan, pwd, rdate, readahead,
readlink, realpath, reboot, renice, reset, resize, rm, rmdir,
rmmmod, route, rpm, rtcwake, runlevel, runsv, runsvdir, rx,
sed, sendmail, seq, setconsole, setkeycodes, setsid, setuidgid,
sh, sleep, softlimit, sort, split, start-stop-daemon, stat,
strings, stty, su, sulogin, sv, svlogd, swapoff, swapon,
switch_root, sync, sysctl, syslogd, tail, tar, tcpsvd, tee,
telnet, telnetd, test, tftp, tftpd, time, top, touch, tr,
traceroute, true, tty, ttysize, udhcpc, udhcpd, udpsvd,
umount, uname, uniq, unlzma, unzip, uptime, usleep, vconfig,
vi, watch, watchdog, wc, wget, which, who, whoami, xargs,
yes, zcat, zcip
```

Dodatkowo zainstalowane są pakiety dropbear (klient/serwer SSH), wireless tools, oraz mtd-tools:

```
dropbear
iwconfig, iwgetid, iwlist, wpriv, iwspy
flash_erase, flash_eraseall, flash_info, flash_lock,
flash_unlock, flashcp, nanddump, nandtest, nandwrite, ubiattach,
ubidetach, ubiformat, ubimkvol, ubirmvol, ubiupdatevol
```

Dostępny jest również system zarządzania pakietami opkg. Na naszym serwerze przygotowaliśmy ok. 300 pakietów z oprogramowaniem (są one też dołączone na płycie CD). Listę pakietów wraz z opisami można zobaczyć tutaj: <http://www.propox.com/download/linux/at91/packages/Packages.gz>

Wyświetlenie listy pakietów w konsoli modułu:

```
opkg  
opkg list
```

update

Instalacja pakietu na module:

```
opkg  
opkg install nazwa_pakietu
```

update

(opkg update wystarczy wykonać raz po każdym restarcie systemu).

Więcej opcji:

```
opkg
```

4. Rozmieszczenie komponentów w pamięci NAND Flash

Pamięć NAND Flash podzielona jest na pięć partycji, które opisane są w tabeli poniżej:

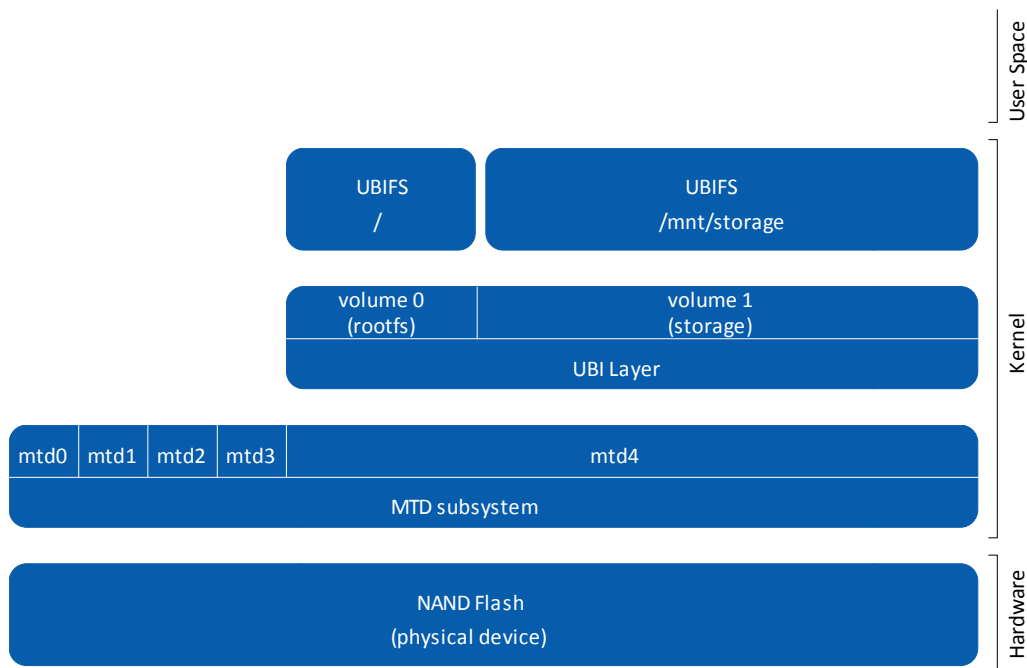
Urządzenie	Adr. Pocz.	Adr. Końc.	Rozmiar	Nazwa
mtd0	0x00000000	0x0003ffff	0x00040000	bootstrap
mtd1	0x00040000	0x0007ffff	0x00040000	u-boot
mtd2	0x00080000	0x001fffff	0x00180000	u-boot environment
mtd3	0x00200000	0x007fffff	0x00600000	kernel
mtd4	0x00800000	0x3fffffff	0x3f800000	filesystems

Partycja mtd0 zawiera bootloader AT91Boot, w partycji mtd2 umieszczone są zmienne środowiskowe bootloadera U-Boot (mtd1). Partycja mtd4 przeznaczona jest na systemy plików.

Na partycji mtd4 umieszczona jest warstwa UBI, która została podzielona na dwa woluminy:

- „rootfs” o rozmiarze 128MB
- „storage” o rozmiarze ok. 840MB (reszta pamięci)

Na obu tych woluminach zostały utworzone systemy plików UBIFS. Wolumin „rootfs” jest montowany przy uruchamianiu jądra w głównym katalogu / i zawiera cały system. Wolumin „storage” montowany jest podczas uruchamiania systemu OpenWrt w katalogu /mnt/storage.



Rysunek 4 Organizacja pamięci stałej.

Dokumentację podsystemu mtd, warstwy UBI, systemu plików UBIFS oraz związanych z nimi narzędzi mtd-utils można znaleźć na stronie: <http://www.linux-mtd.infradead.org/>

5. Aktualizacja oprogramowania

Do wykonania aktualizacji potrzebny jest komputer PC z serwerem TFTP z którego pobierane będą pliki do aktualizacji. W systemie Windows można do tego celu użyć prostego w konfiguracji programu Tftpd32, który można pobrać pod adresem: <http://tftpd32.jounin.net/> (jest on również załączony na płycie CD). W systemie Linux należy skonfigurować np. serwer tftpd (opis konfiguracji w Ubuntu można znaleźć w załączniku). Serwer TFTP powinien mieć adres IP 192.168.1.20.

5.1. Aktualizacja kernela

5.1.1. Z poziomu Linuxa

Pobierz obraz ulmage nowego kernela z serwera TFTP:

```
tftp -g -r uImage 192.168.1.20
```

Skasuj partycję mtd zawierającą kernel:

```
flash_eraseall /dev/mtd3
```

Zapisz nowy kernel w partycji mtd:

```
nandwrite -p -m /dev/mtd3 uImage
```

5.1.2. Z poziomu U-Boota

Pobierz obraz ulmage nowego kernela z serwera TFTP:

```
tftp 0x22000000 192.168.1.20:uImage
```

Skasuj obszar w pamięci NAND Flash zawierający kernel:

```
nand erase 0x200000 0x600000
```

Zapisz nowy kernel w pamięci NAND Flash:

```
nand write 0x22000000 0x200000 0x600000
```

Przy aktualizacji jądra do innej wersji niż domyślna należy pamiętać do skopiowaniu nowych modułów jądra do katalogu /lib/modules

5.2. Instalacja systemu od początku

Poniżej opisano procedurę instalacji całego systemu od początku.

UWAGA: w wyniku tej operacji wszystkie dane zapisane w systemach plików w pamięci NAND Flash zostaną stracone!

Na komputerze PC powinno być zainstalowane oprogramowanie SAM-BA (AT91-ISP) firmy Atmel (znajduje się na CD). Atmel oficjalnie dostarcza oprogramowanie jedynie dla systemu Windows, istnieje również odpowiednik SAM-BA dla Linuxa, jednak nie był on przez nas testowany:

http://www.linux4sam.org/twiki/bin/view/Linux4SAM/SoftwareTools#SAM_BA_Linux_initiative

W sieci lokalnej, do której podłączony jest moduł, powinien znajdować się serwer DHCP.

1. Najpierw trzeba na module uruchomić bootloader ROMBoot. Aby to zrobić trzeba skasować bootloader AT91Boot.

Aby skasować AT91Boot z poziomu Linuxa:

```
flash_eraseall /dev/mtd0
```

Aby skasować AT91Boot z poziomu U-Boot'a:

```
nand erase 0x0 0x40000
```

Jeśli nie ma dostępu do Linuxa ani U-Boot'a należy założyć zworę SAMBA (dioda USB będzie się świecić) i dwukrotnie wyłączyć/włączyć zasilanie. W terminalu DBGU powinno pokazać się RomBOOT (i nic poza tym). Zworę SAMBA należy zdjąć.

2. Port USB Device modułu podłącz do komputera PC. W menedżerze urządzeń powinno pojawić się urządzenie USB o nazwie „atm6124.Sys ATMEL AT91xxxx Test Board”.
3. Z katalogu flashing na płycie CD należy uruchomić skrypt "MMnet1000_prog.bat". Po kilku sekundach pojawi się log z programowania, można go zamknąć (prawidłowy log z programowania można znaleźć w załączniku). Mamy już zaprogramowane bootloadery AT91Boot i U-Boot.
4. W serwerze TFTP (adres 192.168.1.20) udostępnić katalog /flashing/tftp z płyty CD.
5. Po zresetowaniu modułu (konieczne jest wyłączenie i ponowne włączenie zasilania) uruchomi się U-Boot i załaduje z serwera TFTP specjalną wersję systemu Linux, przygotowaną do programowania pamięci Flash.
6. Po uruchomieniu się systemu Linux należy wydać polecenie

```
./program
```

Spowoduje to ściągnięcie z TFTP plików z systemem i zaprogramowanie ich. Po zaprogramowaniu uruchomi się gotowy system.

6. Obsługa peryferiów

6.1. Ethernet

System pracujący na module posiada pełne wsparcie dla interfejsu Ethernet. Obsługiwany jest on tak jak w każdym systemie Linux.

```
root@MMnet:/# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 3A:1F:34:08:54:54
          inet          addr:192.168.1.13          Bcast:192.168.1.255
Mask:255.255.255.0
          inet6 addr: fe80::381f:34ff:fe08:5454/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1941 errors:0 dropped:0 overruns:0 frame:0
          TX packets:55 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:218829 (213.7 KiB)  TX bytes:6655 (6.4 KiB)
          Interrupt:21 Base address:0x4000
```

6.2. RS232

W systemie pracującym na module skonfigurowane są trzy interfejsy RS232:

- DBGU jako ttyS0 (konsola systemowa)
- USART0 jako ttyS1 (używa sygnały modemowe: CTS, RTS, DTR, DSR, DCD, RI)
- USART1 jako ttyS2 (używa sygnały modemowe: CTS, RTS)

Aby dodać więcej interfejsów lub usunąć istniejące konieczna modyfikacja źródeł jądra Linuxa i jego rekompilacja. Interfejsy USART rejestrowane są w funkcji `ek_map_io` w pliku `arch/arm/mach-at91/board-mmnet1000.c`.

Opis i przykłady programowania portów szeregowych można znaleźć w instrukcji „Serial Programming Guide for POSIX Operating Systems”: <http://www.easysw.com/~mike/serial/serial.html>

6.3. USB Host

Host USB na module jest w pełni wspierany przez system Linux. System posiada wiele skompilowanych modułów jądra obsługujących różne urządzenia USB, np. dyski twarde, karty dźwiękowe, karty Wi-Fi, kamery. Ponieważ moduł pracuje pod aktualnym jądrem 2.6.29.3, wszystkie urządzenia USB obsługiwane przez Linuxa powinny pracować również na module.

- Przykład zamontowania dysku twardego USB:

```
modprobe sd_mod
modprobe vfat
mkdir /mnt/usbstorage/
mount /dev/sda1 /mnt/usbstorage/
```

(po podłączeniu dysku do USB w katalogu `/dev/` powinno automatycznie pojawić się urządzenie `sda1`).

- Przykład uruchomienia karty Wi-Fi USB z chipsetem RT73:

Skopiuj plik firmware rt73.bin dostarczony wraz kartą do katalogu /lib/firmware na module:

```
cd /lib/firmware/  
tftp -g -r rt73.bin 192.168.1.20
```

Po podłączeniu karty załaduj moduł rt73usb:

```
modprobe rt73usb
```

Uruchom interfejs sieciowy wlan0:

```
ifconfig wlan0 up
```

Skonfiguruj sieć Wi-Fi:

```
iwconfig wlan0 essid MY_NETWORK_NAME
```

Wymuś pobranie adresu IP z serwera DHCP:

```
udhcpc -i wlan0
```

Dostępne sieci można wyszukać poleceniem iwlist:

```
root@MMnet:/# iwlist wlan0 scan  
wlan0 Scan completed :  
Cell 01 - Address: 00:12:17:70:E0:A1  
Channel:11  
Frequency:2.462 GHz (Channel 11)  
Quality=50/70 Signal level=-60 dBm  
Encryption key:off  
ESSID:"MY_NETWORK_NAME"  
Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s  
24 Mb/s; 36 Mb/s; 54 Mb/s  
Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 48 Mb/s  
Mode:Master  
Extra:tsf=000000c8ef1c818f  
Extra: Last beacon: 20ms ago  
IE: Unknown: 0006507230703058  
IE: Unknown: 010882848B962430486C  
IE: Unknown: 03010B  
IE: Unknown: 2A0107  
IE: Unknown: 2F0107  
IE: Unknown: 32040C121860  
IE: Unknown: DD050010180104
```

6.4.USB Device

Port USB Device jest w pełni wspierany przez jądro. System Linux posiada framework „USB Gadget” który potrafi pracować jako różne urządzenia USB Device. Dokumentację można znaleźć na stronie: <http://www.linux-usb.org/gadget/>

6.5. MicroSD

Aby zamontować kartę pamięci microSD włożoną do złącza w module MMnet1002:


```
modprobe at91_mci
modprobe mmc_block
modprobe vfat
mkdir /mnt/mmc
mount /dev/mmcblk0p1 /mnt/mmc/
```

6.6. LED „USR”

Jądro Linuxa posiada podsystem LED, którego dokumentację można znaleźć w źródłach kernela w pliku Documentation/LED-class.txt.

W systemie pracującym na module zarejestrowane jest jedno urządzenie led o nazwie usr, które sterują diodą „USR” podłączoną do portu PC15:

```
root@MMnet:/# ls /sys/class/leds/usr/
brightness device power subsystem trigger uevent
```

Urządzenia LED sterowane są za pomocą zdarzeń “trigger”. Zdarzenia dostępne na module to:

```
root@MMnet:/# cat /sys/class/leds/usr/trigger
none nand-disk mmc0 timer [heartbeat] default-on
```

Domyślnie ustawione jest zdarzenie “heartbeat”, które mruga diodą z szybkością uzależnioną od obciążenia systemu.

Aby ustawić inny trigger, np. świecący diodą w czasie korzystania z pamięci NAND Flash:

```
root@MMnet:/# echo nand-disk > /sys/class/leds/usr/trigger
```

Aby umożliwić ręczne sterowanie diodą USR należy ustawić trigger “none”:

```
root@MMnet:/# echo none > /sys/class/leds/usr/trigger
```

Sterowanie diodą odbywa się wtedy poprzez plik brightness:

```
root@MMnet:/# echo 0 > /sys/class/leds/usr/brightness
root@MMnet:/# echo 1 > /sys/class/leds/usr/brightness
```

Na płycie CD znajduje się program demonstracyjny napisany w C, który mruga diodą USR zadaną częstotliwością.

Więcej informacji na temat podsystemu LED należy szukać w dokumentacji jądra Linux.

6.7.GPIO

Jądro Linuxa uruchomione na module posiada wsparcie dla obsługi portów gpio z poziomu userspace. Dokumentacja podsystemu gpio znajduje się w źródłach jądra w katalogu Documentation/gpio.txt.

Dostęp do gpio z poziomu userspace jest możliwy poprzez wirtualny system plików /sys/. Dostępne są w nim następujące pliki i katalogi:

```
root@MMnet:/# ls /sys/class/gpio/
export gpiochip32 gpiochip64 gpiochip96 unexport
```

Aby uzyskać dostęp do danego wyprowadzenia gpio należy je wyeksportować do userspace. W poniższych przykładach zostanie użyty pin PC7 (gpio103):

```
root@MMnet:/# echo 103 > /sys/class/gpio/export
```

Po tej operacji w /sys/class/gpio pojawia się nowy katalog odpowiadający wyeksportowanemu pinowi:

```
root@MMnet:/# ls /sys/class/gpio/gpio103/
direction power subsystem uevent value
```

Numeracja pinów dla procesora AT91SAM9260 wygląda następująco:

Tabela 1 Numeracja gpio.

port	gpio	port	gpio	port	gpio	port	gpio
PA0	gpio32	PB0	gpio64	PB16	gpio80	PC0	gpio96
PA1	gpio33	PB1	gpio65	PB17	gpio81	PC1	gpio97
PA2	gpio34	PB2	gpio66	PB18	gpio82	PC2	gpio98
PA3	gpio35	PB3	gpio67	PB19	gpio83	PC3	gpio99
PA4	gpio36	PB4	gpio68	PB20	gpio84	PC4	gpio100
PA5	gpio37	PB5	gpio69	PB21	gpio85	PC5	gpio101
PA6	gpio38	PB6	gpio70	PB22	gpio86	PC6	gpio102
PA8	gpio40	PB7	gpio71	PB23	gpio87	PC7	gpio103
PA9	gpio41	PB8	gpio72	PB24	gpio88	PC8	gpio104
PA23	gpio55	PB9	gpio73	PB25	gpio89	PC9	gpio105
PA24	gpio56	PB10	gpio74	PB26	gpio90	PC10	gpio106
PA30	gpio62	PB11	gpio75	PB27	gpio91	PC11	gpio107
PA31	gpio63	PB12	gpio76	PB28	gpio92	PC12	gpio108
		PB13	gpio77	PB29	gpio93	PC15	gpio111
		PB14	gpio78	PB30	gpio94		
		PB15	gpio79	PB31	gpio95		

Aby skonfigurować wybrany pin jako wejście lub wyjście należy do pliku direction wpisać jeden z atrybutów: „in”, „out”, „high”, „low”. Atrybuty „high” i „low” konfiguruja pin jako wyjście, jednocześnie ustawiając na nim poziom odpowiednio wysoki i niski. Np.:

```
root@MMnet:/# echo "low" > /sys/class/gpio/gpio103/direction
```

Plik direction można również odczytać:

```
root@MMnet:/# cat /sys/class/gpio/gpio103/direction
out
```

Ustawianie i odczytywanie wartości wyjściowej (oraz wejściowej w przypadku skonfigurowania pinu jako wejście) odbywa się poprzez plik value:

```
root@MMnet:/# echo 1 > /sys/class/gpio/gpio103/value
root@MMnet:/# cat /sys/class/gpio/gpio103/value
```

```
1
root@MMnet:/# echo 0 > /sys/class/gpio/gpio103/value
root@MMnet:/# cat /sys/class/gpio/gpio103/value
0
```

Na płycie CD znajduje się program demonstracyjny napisany w C, który mruga diodą LED za pośrednictwem wybranego portu gpio.

Więcej informacji na temat gpio należy szukać w dokumentacji jądra Linux.

Uwaga: niektóre wyprowadzenia gpio są zarezerwowane przez inne peryferia (np. RS232, MMC itp.) i nie mogą być używane. Dostępność wyprowadzeń można sprawdzić w poniższej tabeli opisującej wyprowadzenia modułów MMnet1001/1002.

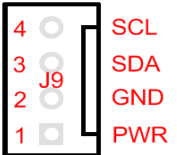
Tabela 2 Wyprowadzenia modułów MMnet1001/1002

Złącze na EVBmmTm	Domyślne skonfigurowana funkcja	Numer gpio	Funkcja alternatywna 1 [Funkcja alternatywna 2]	Nazwa/Port	J1		Nazwa/Port	Funkcja alternatywna 1 [Funkcja alternatywna 2]	Numer gpio	Domyślne skonfigurowana funkcja	Złącze na EVBmmTm
					1	2					
A1			GND	GND	1	2	PC15	NWAIT [IRQ1]	gpio111	LED	B1
A1	GPIO	gpio64	SPI1_MISO [TIOA3]	PB0	3	4	PB1	SPI1_MOSI [TIOB3]	gpio65	GPIO	B2
A3	GPIO	gpio66	SPI1_SPCK [TIOA4]	PB2	5	6	PB3	SPI1_NPCS0 [TIOA5]	gpio67	GPIO	B3
A4	USART0	gpio68	TXD0	PB4	7	8	PB5	RXD0	gpio69	USART0	B4
A5	USART0	gpio70	TXD1 [TCLK1]	PB6	9	10	PB7	RXD1 [TCLK2]	gpio71	USART1	B5
A6	GPIO	gpio72	TXD2	PB8	11	12	PB9	RXD2	gpio73	GPIO	B6
A7	DBGU	gpio78	DRXD	PB14	13	14	PB15	DTXD	gpio79	DBGU	B7
A8	GPIO	gpio80	TK0 [TCLK3]	PB16	15	16	PB17	TF0 [TCLK4]	gpio81	GPIO	B8
A9	GPIO	gpio82	TD0 [TIOB4]	PB18	17	18	PB19	RD0 [TIOB5]	gpio83	GPIO	B9
A10			HDMA	HDMA	19	20	HDPB	HDPB			B10
A11			HDMA	HDMA	21	22	HDPB	HDPB			B11
A12			DDM	DDM	23	24	DDP	DDP			B12
A13			TDI	TDI	25	26	TDO	TDO			B13
A14			TMS	TMS	27	28	TCK	TCK			B14
A15			NTRST	NTRST	29	30	RTCK	RTCK			B15
A16			NRST	NRST	31	32	JTAGSEL	JTAGSEL			B16
A17			SHDN	SHDN	33	34	WKUP	WKUP			B17
A18	GPIO	gpio104	NCS4/CFCS0[RTS3]	PC8	35	36	PC9	NCS5/CFCS1[TIOB0]	gpio105	GPIO	B18
A19	-	gpio106	A25/CFRNW [CTS3]	PC10	37	38	PC11	NCS2 [SPI0_NPCS1]	gpio107	GPIO	B19
A20			GND	GND	39	40	PC12	IRQ0 [NCS7]	gpio108	GPIO	B20
					J2						
C1				+3.3V	1	2	GND				D1
C2	MMC	gpio32	SPI0_MISO[MCDDB0]	PA0	3	4	PA1	SPI0_MOSI [MCCDB]	gpio33	MMC	D2
C3	MMC	gpio34	SPI0_SPCK	PA2	5	6	PA3	SPI0_NPCS0[MCDDB3]	gpio35	MMC	D3
C4	MMC	gpio36	RTS2 [MCDDB2]	PA4	7	8	PA5	CTS2 [MCDDB1]	gpio37	MMC	D4
C5	GPIO	gpio38	MCDA0	PA6	9	10	PA8	MCKK	gpio40	MMC	D5
C6	GPIO	gpio41	MCDA1	PA9	11	12	PA23	TWD [ETX2]	gpio55	GPIO/I2C	D6
C7	GPIO/I2C	gpio56	TWCK [ETX3]	PA24	13	14	PA30	SCK2 [RXD4]	gpio62	GPIO	D7
C8	GPIO	gpio63	SCK0 [TXD4]	PA31	15	16	PB10	TXD3 [ISI_D8]	gpio74	GPIO	D8
C9	GPIO	gpio75	RXD3 [ISI_D9]	PB11	17	18	PB12	TXD5 [ISI_D10]	gpio76	GPIO	D9
C10	GPIO	gpio77	RXD5 [ISI_D11]	PB13	19	20	PB20	RK0 [ISI_D0]	gpio84	GPIO	D10
C11	GPIO	gpio85	RF0 [ISI_D1]	PB21	21	22	PB22	DSR0 [ISI_D2]	gpio86	USART0	D11
C12	USART0	gpio87	DCD0 [ISI_D3]	PB23	23	24	PB24	DTR0 [ISI_D4]	gpio88	USART0	D12
C13	USART0	gpio89	RI0 [ISI_D5]	PB25	25	26	PB26	RTS0 [ISI_D6]	gpio90	USART0	D13
C14	USART0	gpio91	CTS0 [ISI_D7]	PB27	27	28	PB28	RTS1 [ISI_PCK]	gpio92	USART1	D14
C15	USART1	gpio93	CTS1 [ISI_VSYNC]	PB29	29	30	PB30	PCK0 [ISI_HSYNC]	gpio94	GPIO	D15
C16	GPIO	gpio95	PCK1 [ISI_MCK]	PB31	31	32	PC0	AD0 [SCK3]	gpio96	GPIO	D16
C17	GPIO	gpio97	AD1 [PCK0]	PC1	33	34	PC2	AD2 [PCK1]	gpio98	GPIO	D17
C18	GPIO	gpio99	AD3 [SPI1_NPCS3]	PC3	35	36	PC4	A23 [SPI1_NPCS2]	gpio100	-	D18
C19	-	gpio101	A24 [SPI1_NPCS1]	PC5	37	38	PC6	TIOB2 [CFCE1]	gpio102	GPIO	D19
C20	GPIO	gpio103	TIOB1 [CFCE2]	PC7	39	40	GND				D20

6.8.I2C

Jądro posiada podsystem I2C obsługujący wiele kontrolerów i urządzeń zgodnych ze standardem I2C. Kernel pracujący na module został skonfigurowany do korzystania z następujących wyprowadzeń:

Tabela 3 Linie I2C.

Linia I2C	Port procesora	Wyprowadzenie modułu	Wyprowadzenie na płycie EVBmmTm	Złącze na MMnet1002
SDA	PA23	J2-12	D6	
SCL	PA24	J2-13	C7	

UWAGA: magistrala I2C na module pracuje z poziomami napięć 3.3V. Wyprowadzenia I/O procesora NIE SĄ kompatybilne z poziomami napięć 5V !

Do korzystania z magistrali I2C konieczne jest załadowanie modułów jądra: i2c-gpio, i2c-core, i2c-algo-bit (dwa ostatnie załadują się automatycznie). Aby z I2C można było korzystać z poziomu userspace należy również załadować moduł i2c-dev.

Narzędzia do obsługi magistrali znajdują się w pakiecie i2c-tools:

```
opkg update
opkg install i2c-tools
```

Wchodzące w skład pakietu programy:

```
i2cdetect i2cdump i2cget i2cset
```

Przykład użycia:

Uruchomienie modułów:

```
root@MMnet:/# modprobe i2c-gpio
i2c-gpio i2c-gpio: using pins 55 (SDA) and 56 (SCL)
```

```
root@MMnet:/# modprobe i2c_dev
i2c /dev entries driver
```

W katalogu /dev/ automatycznie zostaje utworzone urządzenie i2c:

```
root@MMnet:/# ls /dev/ | grep i2c
i2c-0
```

Skanowanie za pomocą i2c-tools pokazuje urządzenia podłączone do magistrali:

```
root@MMnet:/# i2cdetect -y 0
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                --- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- 49 -- -- -- -- -- --
50: 50 51 52 53 -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- --
```

Szczagółową dokumentację podsystemu I2C można znaleźć w źródłach jądra w katalogu Documentation/i2c.

6.9. SPI

Dokumentację podsystemu SPI oraz przykładowe programy można znaleźć w źródłach jądra w katalogu Documentation/spi. W domyślnej konfiguracji nie została skonfigurowana żadna z dwóch dostępnych w procesorze magistral SPI. Konfiguracji można dokonać dodając odpowiednie wpisy w pliku board-mmnet1000.c w znajdującym się w źródłach jądra w katalogu arch/arm/mach-at91. Należy pamiętać że magistrala SPI0 używa tych samych wyprowadzeń co MMC.

6.10. RTC

Sprzętowy zegar RTC kontrolowany jest za pomocą polecenia hwclock. Aby uruchomić zegar i ustawić go zgodnie z czasem systemowym należy wydać polecenie:

```
root@MMnet:/# hwclock -w
```

Czas systemowy ustawia się poleceniem date, np.:

```
root@MMnet:/# date 2009.04.23-12:18:00
Thu Apr 23 12:18:00 UTC 2009
```

7. Środowisko programistyczne i programy przykładowe

Oprogramowanie przeznaczone dla modułu musi być kompilowane na komputerze PC z systemem Linux. Na płycie CD dostarczany jest kros-kompilator, znajduje się on w archiwum OpenWrt-SDK-at91-for-Linux-i686.tar.bz2

Aby kompilator był widoczny w całym systemie należy dodać go do ścieżki:

```
export PATH=$PATH:/path/to/OpenWrt-SDK-at91-for-Linux-i686/staging_dir/toolchain-arm_gcc4.1.2/bin
```

Gdzie “/path/to/” należy zastąpić ścieżką do rozpakowanego katalogu z OpenWrt-SDK.

Środowisko programistyczne jest oparte na:

- gcc 4.1.2
- uClibc 0.9.29
- gdb 6.3

Na płycie dostarczane są również programy demonstracyjne. W tej chwili dostępne są:

- led-blink – program mrugający diodą LED na wybranym porcie z zadaną częstotliwością
- user-led-blink – program mrugający diodą LED USB z zadaną częstotliwością
- spidev_test – program demonstrujący obsługę magistrali SPI
- za programy przykładowe demonstrujące korzystanie z magistrali I2C mogą służyć źródła narzędzi i2c-tools oraz lm-sensors: <http://www.lm-sensors.org/>
- opis i przykłady programowania portów szeregowych można znaleźć w instrukcji „Serial Programming Guide for POSIX Operating Systems”: <http://www.easysw.com/~mike/serial/serial.html>

Aby skompilować program led-blink:

```
arm-linux-uclibc-gcc led-blink.c -o led-blink
```

Uzyskany plik wykonywalny można kopiować na moduł (np. poprzez sieć) i uruchomić. W zależności od sposobu kopiowania, może być konieczne ustawienie bitu wykonywalności dla pliku:

```
root@MMnet:/# tftp -g -r led-blink 192.168.1.20
root@MMnet:/# chmod +x led-blink
root@MMnet:/# ./led-blink 103 500
Exporting gpio 103...OK
Setting direction of gpio 103 to output...OK

Press Ctrl+C to exit.

Setting output value to 0
Setting output value to 1
Setting output value to 0
Setting output value to 1
```

8. Kompilacja systemu

8.1. Kompilacja AT91Boot

Źródła bootloadera AT91Boot przystosowanego do modułów MMnet1001/1002 znajdują się na płycie CD w archiwum /sources/Bootstrap-v1.10-MMnet1000.tar.gz. Aby skompilować bootloader należy z poziomu katalogu board/MMnet1000/nandflash wydać polecenie:

```
make CROSS_COMPILE=arm-linux-uclibc-
```

8.2. Kompilacja U-Boot-a

Źródła bootloadera U-Boot przystosowanego do modułów MMnet1001/1002 znajdują się na płycie CD w archiwum /sources/u-boot-2009.01-MMnet1000.tar.gz. Są one przygotowane do kompilacji, wystarczy je rozpakować. Wykorzystana została konfiguracja at91sam9260ek, która została zmodyfikowana do potrzeb modułów. Plik konfiguracyjny znajduje się w include/configs/at91sam9260ek.h.

Kompilacja:

```
make CROSS_COMPILE=arm-linux-uclibc-
```

Powstały plik u-boot.bin można użyć do zaprogramowania modułu.

8.3. Kompilacja jądra Linuxa

Źródła jądra Linuxa przystosowanego do modułów MMnet1001/1002 znajdują się na płycie CD w archiwum /sources/linux-2.6.29.3-MMnet1000.tar.gz. Po rozpakowaniu źródeł należy je skonfigurować dla płyt MMnet1000:

```
make ARCH=arm CROSS_COMPILE=arm-linux-uclibc- mmnet1000_defconfig
```

Konfiguracja jądra:

```
make ARCH=arm CROSS_COMPILE=arm-linux-uclibc- menuconfig
```

Kompilacja jądra:

```
make ARCH=arm CROSS_COMPILE=arm-linux-uclibc- uImage
```

W wyniku kompilacji w katalogu arch/arm/boot otrzymujemy plik ulmage który należy zaprogramować do pamięci NAND Flash.

Kompilacja modułów:

```
make ARCH=arm CROSS_COMPILE=arm-linux-uclibc- modules
```

Instalacja modułów:

```
make ARCH=arm CROSS_COMPILE=arm-linux-uclibc-  
INSTALL_MOD_PATH=/path/to/my/rootfs modules_install
```

Gdzie /path/to/my/rootfs to ścieżka do katalogu do którego zostaną skopiowane moduły jądra. Należy je następnie skopiować na moduł (np. po spakowaniu) lub utworzyć nowy obraz systemu plików do zaprogramowania w pamięci NAND Flash.

Kod inicjujący peryferia procesora dla modułów MMnet1000 znajduje się w katalogu arch/arm/mach-at91/board-mmnet1000.c

8.4. Kompilacja systemu OpenWrt

System pracujący na module został skompilowany ze źródeł pochodzących z repozytorium SVN (wersja r13340). Na płycie CD załączono źródła w postaci jaka została wykorzystana do przygotowania systemu dostarczanego wraz z modułami.

Ponieważ system OpenWrt nie generuje obrazów UBI, w konfiguratorze jako obraz wyjściowy należy wybrać opcję archiwum .tar.gz a następnie ręcznie przygotować obraz UBI do zaprogramowania pamięci.

Więcej informacji należy szukać w dokumentacji systemu OpenWrt: <http://kamikaze.openwrt.org/docs/openwrt.html#x1-390002>

8.5. Przygotowanie obrazów ubifs i ubi z nowym systemem plików

Potrzebne są dwa katalogi : ./storage (pusty katalog potrzebny do utworzenia systemu plików „storage”) oraz ./rootfs zawierający pliki dla systemu plików root. Na komputerze PC powinny być zainstalowane narzędzia mtd-utils.

Najpierw należy przygotować obrazy UBIFS:

```
mkfs.ubifs -r ./storage -m 2048 -e 258048 -c 4096 -o MMnet1000-storage.ubifs
```

```
mkfs.ubifs -r ./rootfs -m 2048 -e 258048 -c 4096 -o MMnet1000-OpenWrt-rootfs.ubifs
```

A następnie obraz UBI zawierający woluminy z utworzonymi systemami plików UBIFS:

```
ubinize -o MMnet1000-OpenWrt.ubi -m 2048 -p 256KiB ubinize.cfg
```

(gdzie ubinize.cfg to plik konfiguracyjny, który można znaleźć na płycie CD i w załączniku do tego dokumentu). Utworzony plik MMnet1000-OpenWrt.ubi zawiera obraz UBI do zaprogramowania w pamięci Flash.

Dokumentację podsystemu mtd, warstwy UBI, systemu plików UBIFS oraz związanych z nimi narzędzi mtd-utils można znaleźć na stronie: <http://www.linux-mtd.infradead.org/>

9. FAQ

9.1. Podczas uruchamiania się systemu pojawiają się komunikaty typu „Bad eraseblock 1843 at 0x1ccc0000”. Czy to jest normalne?

To jest normalne. Pamięci typu NAND Flash z natury posiadają Bad Blocki, w czasie pracy ich liczba może wzrastać, jednak producent pamięci gwarantuje że nie przekroczy 100. Użyta w module warstwa UBI i system plików UBIFS gwarantują ochronę przed utratą danych.

9.2. Jaka jest żywotność pamięci NAND Flash użytej w module?

Według danych katalogowych maksymalna ilość zapisów do jednego bloku pamięci to 10000. Jednak zastosowana w module warstwa UBI używa algorytmu „wear leveling” rozpraszającego zapisy po całej pamięci, dzięki czemu można oszacować, że pamięć zużyje się dopiero po zapisaniu do systemów plików ok. 10 TB danych. Deklarowany przez producenta pamięci czas przechowywania danych to 10 lat.

9.3. Karta microSD nie działa z modułem MMnet1002

Sprawdź, czy w module zamontowany jest rezystor R44, jeśli tak to wylutuj go. Niektóre z pierwszych wyprodukowanych modułów zostały zmontowane z tym rezystorem.

10. Przydatne linki

<http://kernel.org/>
<http://www.denx.de/wiki/U-Boot/WebHome>
<http://openwrt.org/>
<http://www.linux4sam.org>
<http://www.at91.com>
<http://www.network-theory.co.uk/gcc/intro/>
<http://www.linux-mtd.infradead.org/>
<http://www.linux-usb.org/>
<http://www.linux-usb.org/gadget/>
<http://elinux.org>

11. Licencja, gwarancja i wsparcie techniczne

Większość oprogramowania dostarczanego wraz z modułem jest na licencji GNU GPL lub podobnej. Szczegółów odnośnie licencji należy szukać w źródłach danego programu i w jego dokumentacji.

Oprogramowanie dostarczane wraz z modułem jest darmowym, wolnym oprogramowaniem. Jest ono udostępniane w postaci „takiej jaka jest”, z nadzieją że będzie użyteczne, jednak BEZ JAKIEJKOLWIEK GWARANCJI. Firma Propox nie ponosi żadnej odpowiedzialności za działanie oprogramowania.

Jesteśmy w stanie dostarczyć jedynie podstawowe wsparcie techniczne w sprawach ściśle związanych ze sprzedawanymi przez nas modułami MMnet1001/1002. Nie udzielamy wsparcia na tematy ogólno-Linuxowe ani w sprawie dostarczanego z modułem oprogramowania, które nie jest naszego autorstwa.

Pomoc techniczna udzielana jest wyłącznie przez e-mail. Pytania prosimy kierować na adres: support@propox.com

12. Załączniki

12.1. Log z uruchamiania się systemu

```
RomBOOT
>AT91Boot-20081201

U-Boot 2009.01 (Mar 20 2009 - 13:43:24)

DRAM: 64 MB
NAND: 1024 MiB
In: serial
Out: serial
Err: serial
Net: macb0
macb0: Starting autonegotiation...
macb0: Autonegotiation complete
macb0: link up, 100Mbps full-duplex (lpa: 0x45e1)
Hit any key to stop autoboot: 0

NAND read: device 0 offset 0x200000, size 0x200000
2097152 bytes read: OK
## Booting kernel from Legacy Image at 22000000 ...
Image Name: Linux-2.6.28.8
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 1469276 Bytes = 1.4 MB
Load Address: 20008000
Entry Point: 20008000
Verifying Checksum ... OK
Loading Kernel Image ... OK
OK

Starting kernel ...

Uncompressing
Linux.....
..... done, booting the kernel.
Linux version 2.6.28.8 (user@MMnet1000DevEnv) (gcc version 4.1.2) #1
PREEMPT Fri Mar 20 11:07:45 CET 2009
CPU: ARM926EJ-S [41069265] revision 5 (ARMv5TEJ), cr=00053177
CPU: VIVT data cache, VIVT instruction cache
Machine: Propox MMnet1000
Memory policy: ECC disabled, Data cache writeback
Clocks: CPU 198 MHz, master 99 MHz, main 18.432 MHz
Built 1 zonelists in Zone order, mobility grouping on. Total pages:
16256
Kernel command line: mem=64M console=ttyS0,115200 ubi.mtd=4
root=ubi0:rootfs rootfstype=ubifs init=/etc/preinit
AT91: 96 gpio irqs in 3 banks
PID hash table entries: 256 (order: 8, 1024 bytes)
Console: colour dummy device 80x30
console [ttyS0] enabled
Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
```

```

Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
Memory: 64MB = 64MB total
Memory: 61852KB available (2712K code, 197K data, 100K init)
Calibrating delay loop... 98.91 BogoMIPS (lpj=494592)
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
net_namespace: 636 bytes
NET: Registered protocol family 16
AT91: Power Management
AT91: Starting after user reset
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
NET: Registered protocol family 23
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 2048 (order: 2, 16384 bytes)
TCP bind hash table entries: 2048 (order: 1, 8192 bytes)
TCP: Hash tables configured (established 2048 bind 2048)
TCP reno registered
NET: Registered protocol family 1
NetWinder Floating Point Emulator V0.97 (double precision)
JFFS2 version 2.2. (NAND) © 2001-2006 Red Hat, Inc.
msgmni has been set to 120
alg: No test for stdrng (krng)
io scheduler noop registered
io scheduler deadline registered (default)
atmel_uart.0: ttyS0 at MMIO 0xfefff200 (irq = 1) is a ATMEL_SERIAL
atmel_uart.1: ttyS1 at MMIO 0xfffb0000 (irq = 6) is a ATMEL_SERIAL
atmel_uart.2: ttyS2 at MMIO 0xfffb4000 (irq = 7) is a ATMEL_SERIAL
loop: module loaded
MACB_mii_bus: probed
eth0: Atmel MACB at 0xfffc4000 irq 21 (3a:1f:34:08:54:54)
eth0: attached PHY driver [Davicom DM9161A]
(mii_bus:phy_addr=ffffffff:00, irq=-1)
NAND device: Manufacturer ID: 0x2c, Chip ID: 0xd3 (Micron NAND 1GiB 3,3V
8-bit)
Scanning device for bad blocks
Bad eraseblock 1843 at 0x1ccc0000
Bad eraseblock 3063 at 0x2fdc0000
Bad eraseblock 3129 at 0x30e40000
Bad eraseblock 3202 at 0x32080000
Creating 5 MTD partitions on "atmel_nand":
0x00000000-0x00040000 : "bootstrap"
0x00040000-0x00080000 : "u-boot"
0x00080000-0x00200000 : "u-boot environment"
0x00200000-0x00800000 : "kernel"
0x00800000-0x40000000 : "filesystems"
UBI: attaching mtd4 to ubi0
UBI: physical eraseblock size: 262144 bytes (256 KiB)
UBI: logical eraseblock size: 258048 bytes
UBI: smallest flash I/O unit: 2048
UBI: VID header offset: 2048 (aligned 2048)

```

```
UBI: data offset:                4096
UBI: attached mtd4 to ubi0
UBI: MTD device name:            "filesystems"
UBI: MTD device size:            1016 MiB
UBI: number of good PEBs:        4060
UBI: number of bad PEBs:         4
UBI: max. allowed volumes:       128
UBI: wear-leveling threshold:    256
UBI: number of internal volumes: 1
UBI: number of user volumes:     2
UBI: available PEBs:             0
UBI: total number of reserved PEBs: 4060
UBI: number of PEBs reserved for bad PEB handling: 40
UBI: max/mean erase counter: 28/5
UBI: background thread "ubi_bgt0d" started, PID 266
usbmon: debugfs is not available
ohci_hcd: USB 1.1 'Open' Host Controller (OHCI) Driver
at91_ohci at91_ohci: AT91 OHCI
at91_ohci at91_ohci: new USB bus registered, assigned bus number 1
at91_ohci at91_ohci: irq 20, io mem 0x00500000
usb usb1: configuration #1 chosen from 1 choice
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 2 ports detected
usb usb1: New USB device found, idVendor=1d6b, idProduct=0001
usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
usb usb1: Product: AT91 OHCI
usb usb1: Manufacturer: Linux 2.6.28.8 ohci_hcd
usb usb1: SerialNumber: at91
usbcore: registered new interface driver libusual
mice: PS/2 mouse device common for all mice
rtc-at91sam9 at91_rtt.0: rtc core: registered at91_rtt as rtc0
cpuidle: using governor ladder
sdhci: Secure Digital Host Controller Interface driver
sdhci: Copyright(c) Pierre Ossman
Registered led device: usr
Netfilter messages via NETLINK v0.30.
TCP cubic registered
NET: Registered protocol family 17
rtc-at91sam9 at91_rtt.0: setting system clock to 1970-02-22 20:30:09 UTC
(4566609)
UBIFS: recovery needed
UBIFS: recovery completed
UBIFS: mounted UBI device 0, volume 0, name "rootfs"
UBIFS: file system size: 131862528 bytes (128772 KiB, 125 MiB, 511
LEBs)
UBIFS: journal size: 9420800 bytes (9200 KiB, 8 MiB, 37 LEBs)
UBIFS: media format: 4 (latest is 4)
UBIFS: default compressor: LZO
UBIFS: reserved for root: 0 bytes (0 KiB)
VFS: Mounted root (ubifs filesystem).
Freeing init memory: 100K
Warning: unable to open an initial console.
- preinit -
```

```

Press CTRL-C for failsafe
- init -

Please press Enter to activate this console. PPP generic driver version
2.4.2
UBIFS: recovery needed
UBIFS: recovery completed
UBIFS: mounted UBI device 0, volume 1, name "storage"
UBIFS: file system size:      899297280 bytes (878220 KiB, 857 MiB, 3485
LEBs)
UBIFS: journal size:         9420800 bytes (9200 KiB, 8 MiB, 37 LEBs)
UBIFS: media format:         4 (latest is 4)
UBIFS: default compressor: LZO
UBIFS: reserved for root:    0 bytes (0 KiB)
eth0: link up (100/Full)
NET: Registered protocol family 10

BusyBox v1.11.3 (2008-12-03 11:05:39 CET) built-in shell (ash)
Enter 'help' for a list of built-in commands.

|_| .----- .----- .----- | | | | .----- | |_| | | | | | | | | | | | | | | | | | | | | | |
| - | | _ | -__| | | | | | | | | | | | | | | | | | | | | |
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
|_| W I R E L E S S F R E E D O M
KAMIKAZE (bleeding edge, r13340) -----

Propox MMnet1000
www.propox.com
-----

root@MMnet:/#

```

12.2. Zmienne środowiskowe U-Boot-a służące do programowania systemu

```

ethact=macb0
ethaddr=3a:1f:34:08:54:54
serverip=192.168.1.20
ipaddr=192.168.1.55
baudrate=115200
bootdelay=1
bootargs=mem=64M console=ttyS0,115200 root=/dev/ram0 init=/etc/preinit
bootcmd=tftp 0x22000000 uImage-prog; bootm 0x22000000
stdin=serial
stdout=serial
stderr=serial

```

12.3. Zmienne środowiskowe U-Boot-a

```

ethact=macb0
ethaddr=3a:1f:34:08:54:54
serverip=192.168.1.20

```

```
ipaddr=192.168.1.55
baudrate=115200
bootdelay=1
bootargs=mem=64M console=ttyS0,115200 ubi.mtd=4 root=ubi0:rootfs
rootfstype=ubifs init=/etc/preinit
bootcmd=nand read 0x22000000 0x00200000 0x00200000; bootm 0x22000000
stdin=serial
stdout=serial
stderr=serial
```

12.4. Skrypt służący do programowania systemu

```
#!/bin/sh

udhcpc -f -q -A 3

echo "Getting files from TFTP server"

tftp -g -r uImage 192.168.1.20
tftp -g -r MMnet1000-OpenWrt.ubiimg 192.168.1.20
tftp -g -r uboot-env.bin 192.168.1.20

echo "Writing rootfilesystem"

ubiformat /dev/mtd4 -f MMnet1000-OpenWrt.ubiimg

echo "Writing Linux kernel"

flash_eraseall /dev/mtd3
nandwrite -p /dev/mtd3 uImage

echo "Writing new U-Boot env. file"

flash_eraseall /dev/mtd2
nandwrite -p /dev/mtd2 uboot-env.bin

echo "Rebooting to new system"

reboot
```

12.5. Zawartość pliku konfiguracyjnego programu ubinize

```
[rootfs-volume]
mode=ubi
image=MMnet1000-OpenWrt-rootfs.ubifsimg
vol_id=0
vol_size=128MiB
vol_type=dynamic
vol_name=rootfs

[storage-volume]
mode=ubi
image=MMnet1000-storage.ubifsimg
vol_id=1
vol_size=800MiB
```



```
vol_type=dynamic
vol_name=storage
vol_flags=autoresize
```

12.6. Prawidłowy log z programowania za pomocą SAM-BA

```
-I- Waiting ...
connection : \usb\ARM0
board : AT91SAM9260-EK
target(handle) : 17976840
read chip ID : 0x00000010 at addr: 0xFFFFEE40
read chip ID : 0x019803A2 at addr: 0xFFFFF240
-I- Found processor : AT91SAM9260 (0x019803A0)
-I- Loading applet isp-extram-at91sam9260.bin at address 0x200000
-I- Memory Size : 0x4000000 bytes
-I- Buffer address : 0x2007C4
-I- Buffer size: 0x0 bytes
-I- Applet initialization done
-I- External RAM initialized
script file : MMnet1000_prog.tcl
-I- === Initialize the NAND access ===
-I- NANDFLASH::Init (trace level : 3)
-I- Loading applet isp-nandflash-at91sam9260.bin at address 0x20000000
-I- Memory Size : 0x40000000 bytes
-I- Buffer address : 0x200047E4
-I- Buffer size: 0x40000 bytes
-I- Applet initialization done
-I- === Load the bootstrap: nandflash_at91sam9-ek in the first sector
===
GENERIC::SendFile AT91Boot_nandflash_MMnet1000.bin at address 0x0
-I- File size : 0xFF4 byte(s)
-I- Writing: 0xFF4 bytes at 0x0 (buffer addr : 0x200047E4)
-I- 0xFF4 bytes written by applet
-I- === Load the u-boot in the next sectors ===
-I- Send File u-boot_nandflash.bin at address 0x00040000
GENERIC::SendFile u-boot_nandflash.bin at address 0x40000
-I- File size : 0x29F30 byte(s)
-I- Writing: 0x29F30 bytes at 0x40000 (buffer addr : 0x200047E4)
-I- 0x29F30 bytes written by applet
-I- === Load the u-boot environment variables ===
-I- Send File uboot-env-prog.bin at address 0x00080000
GENERIC::SendFile uboot-env-prog.bin at address 0x80000
-I- File size : 0x40000 byte(s)
-I- Writing: 0x40000 bytes at 0x80000 (buffer addr : 0x200047E4)
-I- 0x40000 bytes written by applet
```

12.7. Opis konfiguracji serwera TFTP w systemie Ubuntu Linux

Zainstaluj tftpd i potrzebne pakiety:

```
sudo apt-get install xinetd tftpd tftp
```

Utwórz plik konfiguracyjny /etc/xinetd.d/tftp z zawartością:

```
service tftp
```

```
{
protocol      = udp
port          = 69
socket_type   = dgram
wait         = yes
user         = nobody
server       = /usr/sbin/in.tftpd
server_args   = /tftpboot
disable      = no
}
```

Utwórz katalog /tftpboot w którym będą udostępniane pliki do pobrania:

```
sudo mkdir /tftpboot
sudo chmod -R 777 /tftpboot
sudo chown -R nobody /tftpboot
```

Uruchom serwer tftpd:

```
sudo /etc/init.d/xinetd start
```