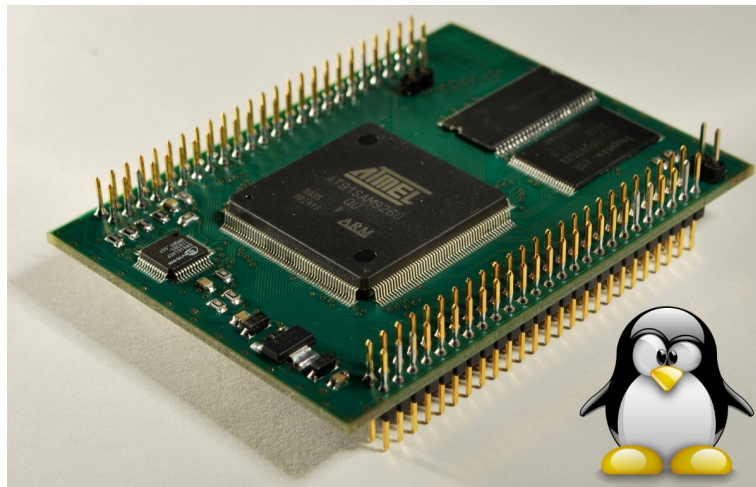


Procesorový modul SAM9260

Linuxový manuál



verze 00.15

Copyright (C) Elvoris s.r.o.

Obsah

1 Úvod	4
2 Začínáme	5
2.1 Vybavení modulu	5
2.1.1 Paměť	5
2.1.2 Sériové porty	6
2.1.3 USB porty	6
2.1.4 Ethernetové rozhraní	6
2.1.5 Programové vybavení	6
2.2 Přístup přes sériový port	7
2.2.1 Zapojení	7
2.2.2 Terminálový program	8
2.2.3 Přihlášení	9
2.2.4 Odhlášení a vypnutí systému	9
2.3 Přístup přes telnet	10
2.3.1 Zapojení	10
2.3.2 Telnetový klient	11
2.3.3 Změna MAC adresy a první PING	12
2.3.4 Přihlášení	13
2.3.5 Odhlášení a vypnutí systému	13
2.4 Změna konfigurace modulu	13
2.5 Kořenový souborový systém přes NFS	15
2.5.1 Počítač jako NFS server	16
2.5.2 Kořenový adresář modulu přes NFS	17
2.5.3 Modul jako NFS klient	18
3 Nahrávání firmware	20
3.1 Bootování modulu	20
3.2 Rozhraní USB	22
3.3 Program SAM-BA	23
3.3.1 Instalace	24
3.3.2 Připojení přes USB	24
3.3.3 Ovládání	25
3.3.4 Tipy a triky	26
3.4 Nahrání firmware	26
3.4.1 Předpoklady	27
3.4.2 Nahrání firmware do DataFlash	27
3.4.3 Nahrání firmware do NandFlash	29
4 Vlastní firmware	31
4.1 Nástroje	31
4.1.1 devkitARM release 32	31
4.1.2 UnixUtils	31
4.1.3 PC s Linuxem	31
4.1.4 Sourcery CodeBench Lite Edition	32
4.1.5 Knihovna ncurses	32
4.2 Bootstrap	32

4.3	Zavaděč U-Boot	33
4.4	Linuxové jádro (kernel)	34
4.5	Kořenový souborový systém	35
4.5.1	Příprava obsahu	36
4.5.2	Příprava obrazu ubifs/ubi	37
5	Příklady	40
5.1	USB host	40
5.1.1	Zapojení	40
5.1.2	USB mass storage	40
5.2	MMC/SD rozhraní	42
5.2.1	Zapojení	42
5.2.2	SD karta	43
5.3	USB-WiFi modul	44
5.3.1	Než začneme	44
5.3.2	Ovladač, firmware a wireless-tools	44
5.3.2.1	Ovladač	44
5.3.2.2	Firmware	45
5.3.2.3	Wireless-tools	45
5.3.3	WiFi pod Linuxem	46
5.3.3.1	Automatická připojení k AP	49

1 Úvod

Držíte v ruce modul SAM9260 a nejspíše i naši základní desku Baseboard. Doufáme, že vám oba výrobky budou sloužit dobře a že si s nimi užijete tolik legrace a zábavy, jako jsme si užili i my, když jsme je pro vás připravovali.

Společně se naučíme, jak začít náš modul používat, jak si do něj nahrát firmware (SW části jako je Linux, obsah systémové části atd.) a pro ty zvědavější je připravena sekce o tom, jak si sestavit svůj vlastní firmware úplně od základů.

A pro hračky máme připravenou kapitolu s několika příklady použití, kde je mimo jiné i ukázka zapojení několika dalších periférií modulu SAM9260.

2 Začínáme

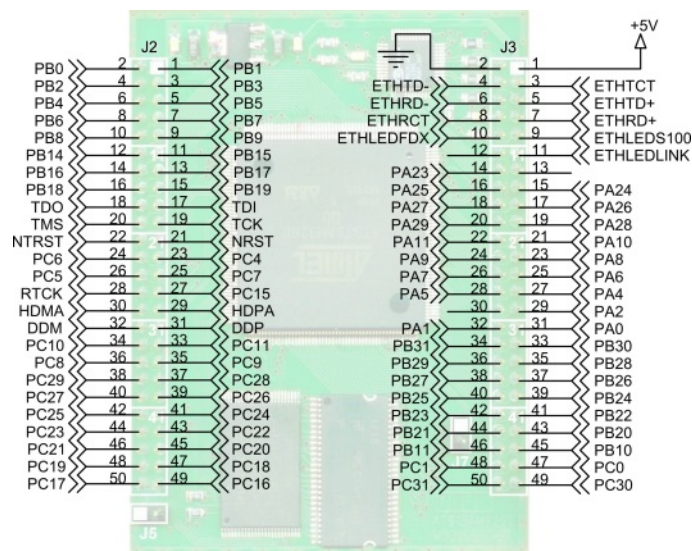
Začátky bývají těžké. Ne však s námi! Nejprve si ukážeme jak zprovoznit modul SAM9260 tak, aby nabootoval do předpřipraveného prostředí operačního systému Linux a aby bylo možné přihlásit se do Linuxu prostřednictvím sériové konzole nebo vzdáleně pomocí telnetového klienta.

Pak si povíme něco o tom, jak upravit konfiguraci modulu SAM9260, resp. Linuxu, který je do něj od výroby nahrán. Uvidíte, jak je snadné změnit IP adresu modulu, název modulu na síti (hostname) nebo třeba celý postup uživatelské části inicializace Linuxu.

A na závěr se dozvíme jak si zjednodušit život v případě, že často mění obsah souborového systému v NandFlash paměti (rozuměj harddisku modulu). Třeba kvůli tomu, že vyvíjíte nějakou aplikaci, kterou je třeba v modulu často aktualizovat.

2.1 Vybavení modulu

V dostupné technické dokumentaci modulu SAM9260 si můžete nastudovat celou řadu údajů, které popisují vybavení modulu. My se zde nebudeme pokoušet znovu popsat již jednou popsané, ale z technické dokumentace si vytáhneme ty nejdůležitější údaje a pokusíme se na ně podívat z hlediska možnosti využití modulu SAM9260.



Obrázek 2.1.1: Modul SAM9260 - popis pinů

2.1.1 Paměť

Modul SAM9260 je vybaven dvěma paměťmi typu *RAM*¹. První, integrovaná v pouzdře samotného mikrokontroléru AT91SAM9260, je paměť SRAM o velikosti 2x 4kB. První část této paměti SRAM0 je primárně určena k tomu, aby do ní byl nahrán tzv. *bootstrap*, což je krátký program, který se stará o základní inicializaci mikrokontroléru a hlavně o inicializaci řadiče paměti SDRAM (je-li jí modul vybaven). Druhá část paměti SRAM, paměť SRAM1 slouží pro uchování zásobníku a proměnných. Samozřejmě můžeme místo bootstrapu do paměti SRAM nahrát

¹Paměť typu RAM slouží jako operační, protože je velmi rychlá, ale svůj obsah si uchovává jen když je připojeno napájecí napětí.

přímo svoji aplikaci, ale sami uznáte, že dostupné 4kB jsou v dnešní době skutečně limitujícím faktorem.

Druhá paměť typu RAM je už zmíněná paměť SDRAM. Její maximální velikost je omezena odpovídajícím prostorem v paměťové mapě modulu a to na 256MB. Tato paměť je vzhledem ke své maximální velikosti určena pro běh uživatelských aplikací přímo na holém CPU nebo pro běh operačního systému a aplikací určených pro běh v prostředí operačního systému.

Vedle paměti typu RAM je modul vybaven i dvěma paměťmi typu *ROM*². Jedná se o paměť DataFlash o velikosti 4MB a o paměť NandFlash o kapacitě 256MB. Přítomnost dvou paměti typu ROM a schopnosti mikrokontroléru AT91SAM9260 nahrát do paměti SRAM bootstrap z obou pamětí nám nabízí dvě základní varianty jejich využití. Buď uložíme bootstrap, zavaděč operačního systému U-Boot a jádro operačního systému do paměti DataFlash, zatímco celou kapacitu paměti NandFlash vyhradíme pro souborový systém operačního systému a uživatelská data. Anebo paměť DataFlash nevyužijeme a kompletní firmware (bootstrap, U-Boot atd) uložíme do paměti NandFlash. První varianta má tu výhodu, že kritické části firmware jsou v paměti DataFlash, která je rychlá a hlavně netrpí neduhy pamětí typu NandFlash jako je chybovost, pomalejší přístup a opotřebením častými zápisy.

2.1.2 Sériové porty

Modul SAM9260 má na svých pinech dostupné čtyři z šesti sériových USART portů (USART0 - 3), které nabízí mikrokontrolér AT91SAM9260 a dále jeden sériový DBGU určený pro ladění programů. My budeme pro komunikaci s modulem SAM9260 využívat přednostně sériový port DBGU.

2.1.3 USB porty

Vedle sériových portů je mikrokontrolér AT91SAM9260 vybaven jedním portem USB device pro připojení k běžnému počítači, kdy se mikrokontrolér, resp. modul chová jako USB periférie, a pak dvěma porty USB host, přes které je k mikrokontroléru, resp. modulu, možné připojit libovolnou USB periférii (modul se chová jako počítač). Na pinech modulu SAM9260 je zpřístupněn port USB device a port A typu USB host.

2.1.4 Ethernetové rozhraní

Modul SAM9260 je vybaven síťovým rozhraním, díky kterému lze modul zapojit do počítačové sítě typu ethernet. Modul je schopen komunikovat rychlostmi 10 a 100Mbit/s. K fyzickému připojení modulu do počítačové sítě chybí jen konektor RJ45, signálové oddělovací trafo a indikační LED diody. Tyto součástky nejsou součástí modulu a musí být k modulu dodány a připojeny zvlášť. Případně je možné využít nabízenou základní desku Baseboard, která obsahuje vše potřebné.

2.1.5 Programové vybavení

Do modulu SAM9260 je už ve výrobě nadržáno kompletní programové vybavení tzv. firmware ve verzi, která je aktuální v době výroby modulu.

Firmware se skládá z primárního zavaděče zvaného **bootstrap** (stará se o základní inicializaci modulu a zavedení sekundárního zavaděče do SDRAM paměti), dále ze sekundárního zavaděče **U-boot** (zavádí operační systém) a **jádra** operačního systému Linux spolu se **souborovým systémem** uloženým v paměti NandFlash.

Primární a sekundární zavaděč jsou spolu s linuxovým jádrem nahrány do paměti DataFlash. Obsah souborového systému je nahrán do paměti NandFlash.

²Paměť typu ROM slouží pro uchování spustitelných souborů aplikací, pro uchování operačního systému, uživatelských dat apod.. Svůj obsah si uchovává i po přerušení/odpojení napájecího napětí. Ve vašem počítači slouží jako paměť typu ROM pevný disk, paměťová karta nebo USB flash disk.

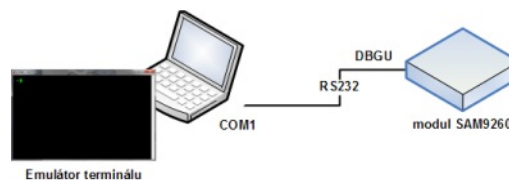
Modul SAM9260 je po zapojení napájecího napětí a po zapojení nezbytných rozhraní schopen naběhnout do promptu operačního systému Linux.

2.2 Přístup přes sériový port

První a základní možností jak přistupovat k modulu SAM9260, je přístup přes sériový port.

Po naboštění do systému Linux se tímto způsobem můžeme do Linuxu přihlásit a ovládat jej přes tzv. sériovou konzoli (rozuměj náš počítač s terminálovým programem) připojenou přes jeden ze sériových portů modulu SAM9260.

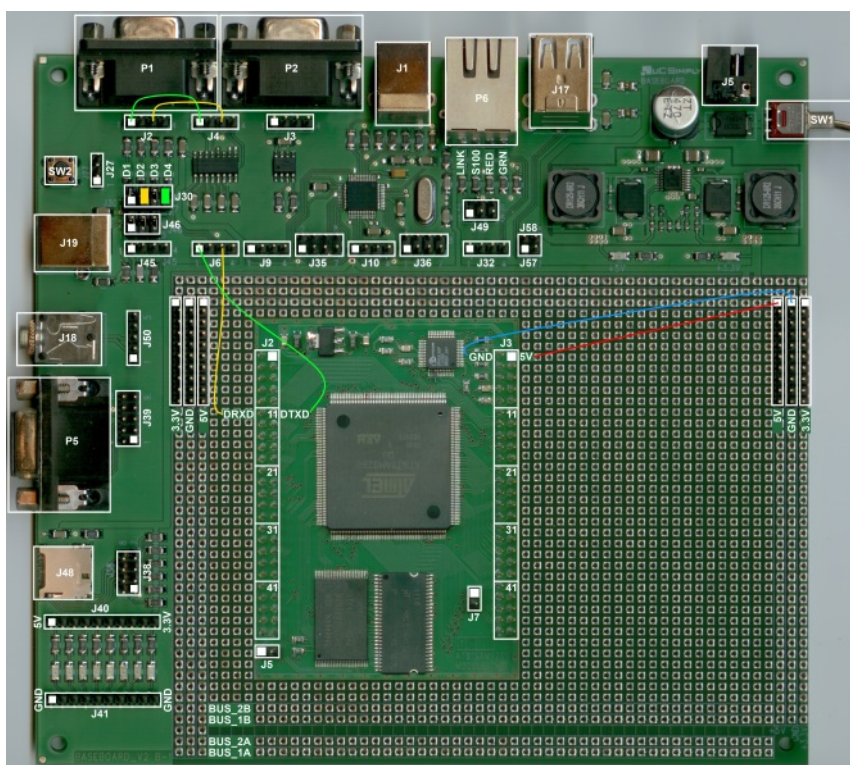
Ve výchozí konfiguraci systému Linux dodávaného s modulem se předpokládá, že sériová konzole je k modulu připojena přes sériový port DBGU (obrázek 2.2.1). Tento port se u mikrokontroléru AT91SAM9260 běžně využívá pro ladící účely. Tato konfigurace není samoučelná. Přes sériový port DBGU totiž komunikuje vestavěný monitor RomBoot, bootstrap a také zavaděč U-Boot. S ohledem na tuto skutečnost je i tzv. *systémová konzole*³ systému Linux nastavena na port DBGU. Díky tomu můžeme na portu DBGU sledovat všechna hlášení od startu vestavěného monitoru až po systémová hlášení a textový výstup Linuxu.



Obrázek 2.2.1: Připojení přes sériový port

2.2.1 Zapojení

Referenční zapojení modulu SAM9260, resp. jeho portu DBGU v případě použití základní desky Baseboard je na obrázku 2.2.2. Modul je zasunutý do pájecího pole základní desky a samotné propojení modulu a jednotlivých komponent základní desky je realizováno drátovými propojkami různých barev.



Obrázek 2.2.2: Modul SAM9260 - zapojení portu DBGU

³Konzole, na kterou systém vypisuje všechna hlášení ať už během bootování nebo při svém běhu

Nejprve něco málo k signálům portu DBGU. Signály *DRXD* (signál *Rx* podle RS232) a *DTXD* (signál *Tx* podle RS232) portu DBGU jsou na modulu SAM9260 přístupné na pinheadu J2 (*DRXD* - pin 12, *DTXD* - pin 11). Signály jsou popsány z pohledu modulu, který se chová jako koncové zařízení (DTE), např. jako modem. Pro připojení k PC, které vystupuje jako řídicí zařízení (DCE), proto postačí libovolný sériový kabel pro připojení modemu, tj. s dutinkami na straně PC a kolíky na straně modulu SAM9260, resp. základní desky Baseboard. Tzv. null-modem kabel (překřížené vodiče Tx a Rx) nelze použít.

Signály *DRXD* a *DTXD* portu DBGU mají úroveň TTL. O jejich převedení do/z úrovní dle specifikace rozhraní RS232 se postará integrovaný obvod U11 typu MAX3232 umístěný na základní desce Baseboard. Oba signály vyvedeme z pinheadu J2 (piny 12 a 11) na modulu a přivedeme je na pinhead J2 (piny 3 a 1) na základní desce. Žlutou propojku použijeme na signál *DRXD*, zelenou pak pro signál *DTXD*. Tím jsme signály v úrovních TTL přivedli na TTL stranu převodníku MAX3232.

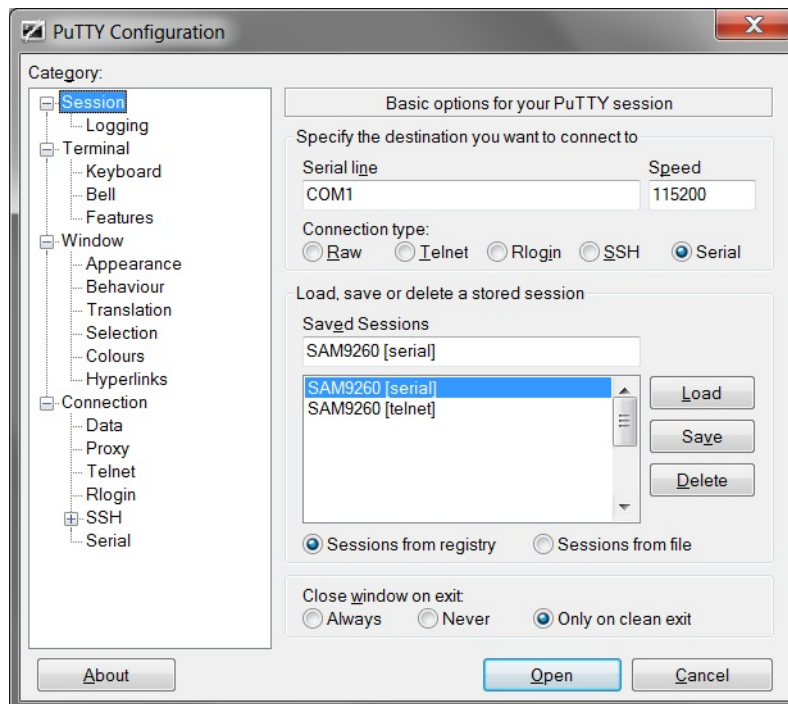
Dále potřebujeme oba signály již v úrovni RS232 přivést na konektor P1 (CAN9F) na základní desce. Spojíme proto piny 3 a 1 pinheadu J4 s piny 3 a 1 pinheadu J2 na základní desce. Tím je zapojení dokončeno.

Pro možnost opticky indikovat probíhající komunikaci je vhodné pomocí jumperů připojit k signálům *DRXD* a *DTXD* LED diody. V případě naší základní desky je to úkol veskrze snadný - pomocí dvou jumperů propojíme druhou a čtvrtou dvojici pinů na pinheadu J30 na základní desce počítáno zleva od USB device konektoru J19 (viz obrázek 2.2.2).

Nesmíme ani zapomenout přivést na modul SAM9260 napájecí napětí +5V. Napájení +5V a zem *GND* přivedeme ze stejnojmenných pinheadů na boku základní desky na pinhead J3 (piny 1 a 3) na modulu SAM9260.

2.2.2 Terminálový program

Pro komunikaci s modulem SAM9260 přes sériový port budeme na straně našeho počítače potřebovat nějaký terminálový program, tzv. emulátor terminálu. Můžeme použít oblíbený *Putty*⁴ nebo klasický Hyperterminál známý z prostředí Windows XP (ve Windows 7 už není dostupný).



Obrázek 2.2.3: Putty - nastavení spojení přes sériový port

⁴<http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe>

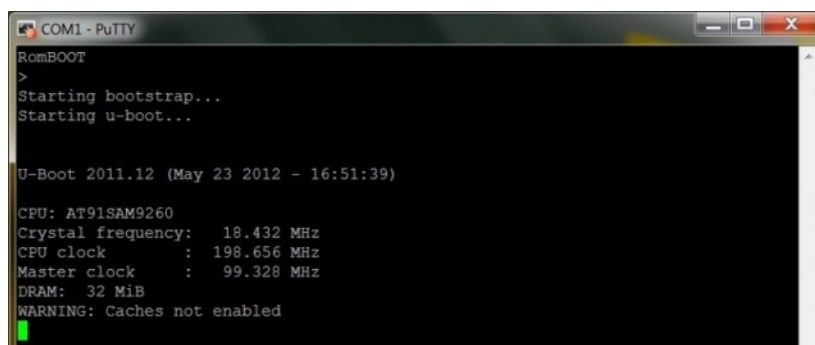
Doporučujeme program Putty, protože je velmi univerzální a umí celou řadu komunikačních protokolů, např. i telnet. Navíc nevyžaduje instalaci, stačí jej zkopírovat do vybraného adresáře a pak rovnou spustit. Níže uvádíme parametry spojení přes sériový port:

- Rychlost: 115200 Bd
- Rámeček: 1 start bit, 8 datových bitů a 1 stop bit
- Parita: žádná
- Řízení toku: žádné

Uvedené parametry a číslo sériového portu na straně počítače nastavíme v Putty v kategorii *Connection*, podkategorie *Serial* (viz strom kategorií *Category* vlevo v okně *Putty*). Pak se přepneme do kategorie *Session*, kde nastavíme jméno předvolby v poli *Saved Sessions*, typ spojení na *Serial* a předvolbu uložíme tlačítkem *Save* (viz obrázek 2.2.3). Samotné spojení s modulem SAM9260 otevřeme tlačítkem *Open*.

2.2.3 Přihlášení

DBGU port máme zapojen, Putty běží na našem počítači. V Putty stiskem tlačítka *Open* otevřeme spojení s modulem SAM9260, připojíme základní desku přes konektor J5 na zdroj napětí 12V / min. 2A a zapneme vypínač SW1 na základní desce. Rozsvítí se LED diody indikující přítomnost napájecího napětí 3.3V a 5V na základní desce a v okně Putty bychom měli vidět hlášení > **RomBOOT** a dále hlášení startujícího bootstrapu a zavaděče U-Boot (viz obrázek 2.2.4).



```

COM1 - PuTTY
RomBOOT
>
Starting bootstrap...
Starting u-boot...

U-Boot 2011.12 (May 23 2012 - 16:51:39)

CPU: AT91SAM9260
Crystal frequency: 18.432 MHz
CPU clock      : 198.656 MHz
Master clock   : 99.328 MHz
DRAM: 32 MiB
WARNING: Caches not enabled

```

Obrázek 2.2.4: Bootování monitoru, bootstrapu a U-Bootu

Pokud tomu tak není, bude chyba nejspíš v propojení počítače a modulu, případně v nastavení parametrů spojení. Opticky můžeme zkontrolovat spojení pomocí LED diod u konektoru P1 (CAN9F). Druhá (modul přijímá) a čtvrtá (modul vysílá) LED dioda zleva od USB device konektoru J19 (viz obrázek 2.2.2) by měly při komunikaci poblepávat.

Po naběhnutí zavaděče U-Boot se čeká jednu sekundu na stisk jakékoliv klávesy, která přeruší automatické zavedení Linuxu. Je tak možné upravit nastavení U-Boot, parametry pro zavádění Linuxu apod. Po uplynutí prodlevy je rozbaleno jádro a začne bootovat samotný systém Linux. Na konci bootování jádra Linuxu je proveden inicializační skript a pak se konečně zobrazí přihlašovací prompt uvozený názvem počítače (hostname) modulu (viz obrázek 2.2.5).

Zadáme výchozí přihlašovací údaje. Jelikož jediným povoleným uživatelem je ve výchozí konfiguraci superuživatel **root** s výchozím heslem **toor**, přihlásíme se pod jeho účtem. Objeví se informace o verzi Busyboxu a jsme tam!

2.2.4 Odhlášení a vypnutí systému

Z Linuxu se odhlásíme pomocí příkazu **exit**.

Bezpečné ukončení práce s Linuxem zařídíme příkazem **poweroff**, pokud chceme odpojit napájení, nebo příkazem **reboot**, jestliže potřebujeme systém pouze restartovat. Napájecí napětí

```

Loading modules... [done]
Enabling network interfaces... [done]
Starting telnetd... [done]

ucsimply-sam9260 login: root
Password:

BusyBox v1.19.4 (2012-05-16 14:36:08 CEST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

~ #

```

Obrázek 2.2.5: Přihlášení přes sériovou konzoli

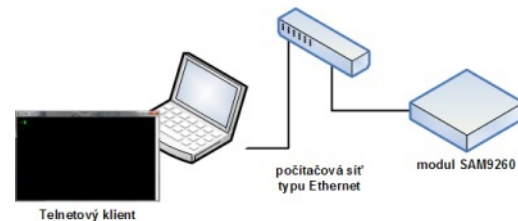
modulu můžeme bezpečně odpojit, až když Linux na příkaz `poweroff` zareaguje hlášením **Power down**. Teprve tehdy je činnost systému skutečně ukončena a nemůže dojít např. k poškození souborového systému v paměti NandFlash.

2.3 Přístup přes telnet

Ovládání modulu SAM9260 přes telnet je další běžně využívanou variantou. Na rozdíl od sériového portu se však k modulu přistupuje vzdáleně prostřednictvím sítě ethernet (obrázek 2.3.1).

Telnetové spojení s modulem lze ovšem navázat jen tehdy, jestliže jsou modul a telnetový klient (např. náš počítač s Putty) vzájemně propojeni přes počítačovou síť, jestliže je modul nabootován až do systému Linux a jestliže v systému Linux běží služba (daemon) protokolu telnet. Tato služba se chová jako server, který na TCP portu 23 čeká na žádost o otevření nového spojení s modulem.

Z povahy telnetového spojení je zřejmé, že tento typ přístupu nelze použít pro sledování hlášení během bootování modulu.



Obrázek 2.3.1: Připojení přes telnet

2.3.1 Zapojení

Modul SAM9260 má integrované kompletní ethernetové rozhraní, které podporuje komunikaci rychlostmi 10/100Mbit. MAC vrstva rozhraní je realizována periferií EMAC mikrokontroléru AT91SAM9260, zatímco externí obvod DM9161AEP tvoří PHY vrstvu. Zbývající chybějící součástky: signálové oddělovací trafo, konektor typu RJ45 a indikační LED diody jsou dostupné na základní desce Baseboard. Jediné, co vlastně potřebujeme, je propojit vývody obvodu PHY s konektorem RJ45, s oddělovacím trafem a s indikačními LED diodami.

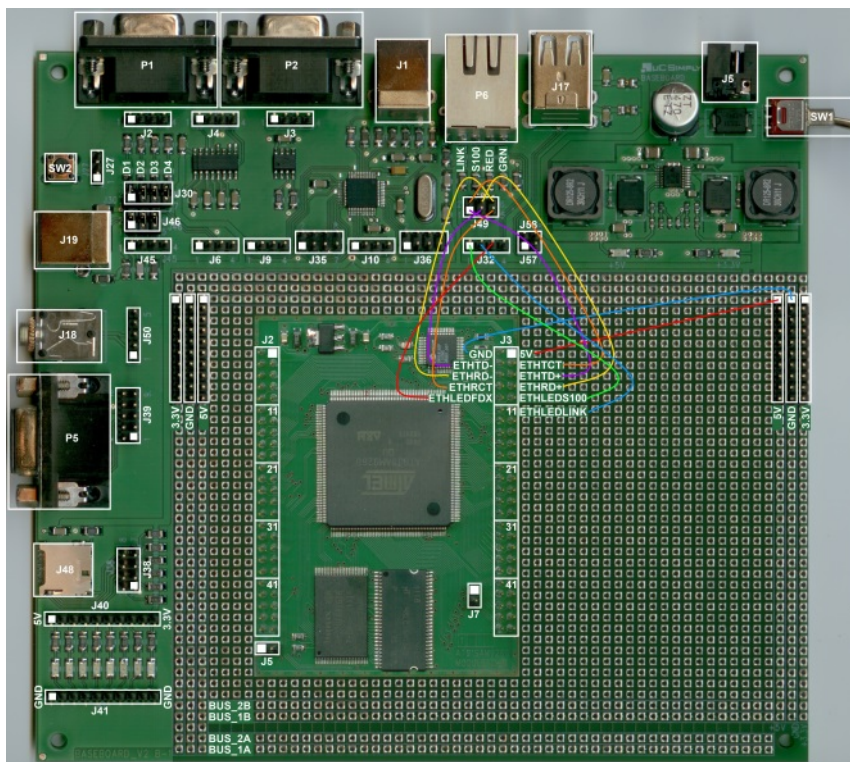
Referenční zapojení ethernetového rozhraní modulu SAM9260 je v případě použití základní desky Baseboard vidět na obrázku 2.3.2. Součástí konektoru typu RJ45, označeném na základní desce jako P6, je vedle indikačních LED diod i oddělovací trafo, takže zapojování bude ještě o něco jednodušší.

Modul zasuneme do pájecího pole základní desky a nachystáme si několik drátových propojek různých barev.

Jako první připojíme k modulu napájecí napětí $+5V$ a zem GND - stejnojmenné pinheady na boku základní desky propojíme s piny 1 a 3 pinheadu J3 na modulu SAM9260.

Dále přivedeme signály *ETHTD-*, *ETHRD-* a *ETHRCT* z PHY vrstvy na konektor P6 - propojíme piny 4, 6 a 8 pinheadu J3 na modulu s piny 3, 6 a 5 pinheadu J49 na základní desce. Obdobně přivedeme na konektor P6 i signály *ETHCT*, *ETHTD+* a *ETHRD+* z PHY vrstvy - propojíme piny 3, 5 a 7 pinheadu J3 na modulu s piny 2, 1 a 4 pinheadu J49 na základní desce.

Zapojování zakončíme připojením indikačních LED diod osazených u konektoru P6. Na obrázku 2.3.2 jsou tyto označeny jako *LINK* (D26), *S100* (D25), *RED* (D28) a *GRN* (D27). Čísla v



Obrázek 2.3.2: Modul SAM9260 - zapojení síťového rozhraní

závorce udávají pak označení ve schématu. Paralelně k LED diodám *LINK* a *S100* jsou zapojeny i LED diody integrované do konektoru P6.

Zpět k samotnému zapojení indikačních LED diod: výstupy obvodu PHY pro ovládání LED diod: *ETHLEDS100*, *ETHLEDLINK* a *ETHLEDFDX* přivedeme na LED diody *S100* (D25), *LINK* (D26) a *RED* (D28) na základní desce - propojíme piny 9, 11 a 10 pinheadu J3 na modulu s piny 1, 2 a 3 pinheadu J32 na základní desce. LED dioda *LINK* (žlutá) a žlutá LED dioda v konektoru P6 budou indikovat připojení linky/aktivitu (svítí/bliká); LED dioda *S100* (zelená) a zelená LED dioda v konektoru P6 budou indikovat aktuální komunikační rychlost 10/100Mbit (nesvítí/svítí). A konečně dioda *RED* (červená) bude indikovat full duplex/kolize (svítí/bliká). LED dioda *GRN* zůstane nevyužita. Tímto jsme dokončili zapojení ethernetového rozhraní.

2.3.2 Telnetový klient

Pro připojení k modulu přes protokol telnet potřebujeme program zvaný telnetový klient, který dokáže navázat spojení se službou běžící v Linuxu na straně modulu. Opět, stejně jako v případě spojení pře sériový port, nám poslouží program Putty.

Nejprve však uvedeme výchozí parametry síťového rozhraní modulu, tak jak jsou nastaveny v předinstalovaném Linuxu a v zavaděči U-Bootu:

- IP adresa: 192.168.1.12
- Masky podsítě: 255.255.255.0
- Brána: 192.168.1.254

IP adresu modulu nastavíme v Putty do pole *Host Name*, v poli *Connection type* zvolíme typ spojení *Telnet* a zkontrolujeme, že cílový port je nastaven na 23. V poli *Saved Sessions* napíšeme jméno předvolby a uložíme ji stiskem tlačítka *Save*. Tím jsme připraveni na otevření spojení s modulem.

```

Hit any key to stop autoboot: 0
u-boot> setenv ethaddr 02:40:50:60:70:90
u-boot> printenv ethaddr
ethaddr=02:40:50:60:70:90
u-boot> saveenv
Saving Environment to dataflash...
u-boot> ping 192.168.1.254
macb0: Starting autonegotiation...
macb0: Autonegotiation complete
macb0: link up, 100Mbps full-duplex (lpa: 0x45e1)
Using macb0 device
host 192.168.1.254 is alive
u-boot>

```

Obrázek 2.3.3: Změna MAC adresy a ping

2.3.3 Změna MAC adresy a první PING

Před použitím síťového rozhraní nás čeká ještě jeden úkon - musíme změnit výchozí MAC adresu síťového rozhraní.

Každé síťové rozhraní má přiřazenu jedinečnou adresu, pomocí které se jednotlivé síťové rozhraní na síti ethernet na vzájemně identifikují. Nazývá se MAC adresa. Je to něco jako IP adresa, ale na nižší úrovni. Seznam MAC adres je spravován globálně institucí IEEE, která přiděluje rozsahy MAC adres jednotlivým výrobcům. Tak je zajištěna globální jedinečnost MAC adresy. Pak je tu ještě možnost použít speciálního rozsahu MAC adres - jsou to adresy, které si můžeme nastavit sami. Za jejich jedinečnost v rámci sítě odpovídáme tudíž my. A právě tuto druhou možnost využijeme, když si budeme nastavovat naši vlastní MAC adresu.

Poznámka: Jestliže chcete mít jistotu, že MAC adresa vašeho modulu je jedinečná, a nechcete používat lokální rozsah, tak můžete použít MAC adresu nějaké staré vyřazené síťové karty. Případně si můžete najít na stránkách IEEE prefix MAC adresy takového výrobce síťových karet, které v síti určitě nemáte. Např. firma DEC, resp. DIGITAL.

Výchozí MAC adresa modulu SAM9260 je zakompilována v binárním souboru sekundárního zaváděče U-Bootu a její hodnota je **02:FF:FF:FF:FF:FF**. Hodnota **02** v nejvýznamnějším bajtu zajišťuje, že druhý bit tohoto bajtu je nastaven na 1. Tak se označují lokálně přidělené MAC adresy. Všechny MAC adresy přidělované globálně mají totiž tento bit nulový. Zbylých nižších pět bajtů může u lokálně přidělované MAC adresy nabývat libovolné hodnoty. Zvolíme si tedy pro ně nějaká hezká čísla, řekněme, že naše nová MAC adresa bude **02:40:50:60:70:90**.

Samotné nastavení MAC adresy provedeme v prostředí zaváděče U-Bootu. Linux totiž nastavení MAC adresy přebírá z U-Bootu a během svého zavádění ji nastavuje do síťového rozhraní.

Spustíme si program Putty, nahrajeme si předvolbu pro sériový port a otevřeme spojení stiskem tlačítka *Open*. Zapneme napájecí napětí na základní desce a jakmile uvidíme, že zaváděč U-Boot startuje (hláška **U-BOOT** s číslem verze a datem sestavení) stiskneme libovolnou klávesu, abychom přerušili automatické zavedení jádra Linuxu. Tím se dostaneme do promptu U-Bootu.

MAC adresa je uložena v proměnné prostředí U-Bootu (něco jako proměnná v linuxovém shellu) z názvem **ethaddr**. Změnu hodnoty proměnné na naši novou MAC adresu provedeme příkazem (celý postup je na obrázku 2.3.3:

```
u-boot> setenv ethaddr 02:40:50:60:70:90
```

Úspěšnost si můžeme zkontrolovat vyčtením aktuální hodnoty pomocí příkazu:

```
u-boot> printenv ethaddr
```

Aby hodnota proměnné **ethaddr** zůstala zachována i mezi restarty a po odpojení napájení, musíme ji uložit do paměti DataFlash (případně NandFlash) modulu. To zajistíme snadno příkazem:

```
u-boot> saveenv
```

U-Boot potvrdí uložení všech proměnných prostředí do DataFlash/NandFlash.

A můžeme jít do finále - zkusíme, jestli síťové rozhraní funguje. Připojíme modul do počítačové sítě. Měly by se rozsvítit obě LED diody na konektoru P6. Příkazem **ping** směrovaným třeba na naši síťovou bránu ověříme, že vše funguje jak má:

```
u-boot> ping 192.168.1.254
```

Pokud vše funguje, U-Boot odpoví takto:

```
macb0: Starting autonegotiation...
macb0: Autonegotiation complete
macb0: link up, 100Mbps full-duplex (lpa: 0x45e1)
Using macb0 device
host 192.168.1.254 is alive
```

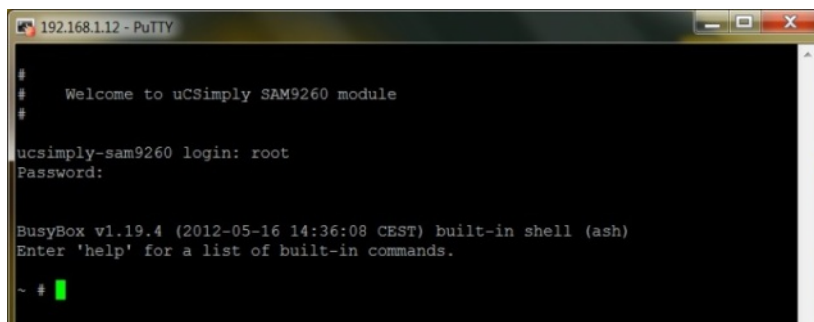
Síťové rozhraní máme nastaveno a oživeno. Teď už můžeme nabootovat do Linuxu. Zadáme příkaz:

```
u-boot> run bootcmd
```

Linux nabootuje a bude čekat na přihlášení. Služba *telnetd* je automaticky spuštěna jako poslední krok při inicializaci systému.

2.3.4 Přihlášení

Spustíme Putty a nahrajeme si předvolbu pro telnetové spojení. Otevřeme spojení tlačítkem *Open*. Objeví se uvítací obrazovka modulu a přihlašovací prompt (obrázek 2.3.4). Přihlásíme se jako superuživatel **root**, heslo **toor** (stejně jako přes sériový port). Zobrazí se úvodní hlášení Busyboxu a prompt. Pak už můžeme Linux ovládat stejně jako přes sériovou konzoli. Ovšem s tím rozdílem, že hlášení jádra zde zobrazeny nebudou, ty jsou posílány na systémovou konzoli (ve výchozím nastavení je to port DBGU). Pokud o ně stojíme, musíme si otevřít druhé okno Putty a připojit se tentokrát přes sériový port.



Obrázek 2.3.4: Přihlášení přes telnet

2.3.5 Odhlášení a vypnutí systému

Ukončení spojení provedeme prostým uzavřením okna Putty nebo zadáním příkazu **exit**. Ukončené spojení ovšem neznamená, že by se modul vypnul. Ne, ten běží spokojeně dál až do zadání příkazu **poweroff** nebo **reboot**. Důkazem mého tvrzení je fakt, že si můžeme znovu otevřít telnetové spojení.

Příkaz **poweroff** provede bezpečné vypnutí systému, ovšem nesmíme se nechat zmást tím, že bezprostředně po zadání příkazu dojde k ukončení spojení ze strany modulu. Modul ještě není připraven na vypnutí napájení. Ukončení spojení způsobilo vypnutí síťového rozhraní, což je jeden z prvních kroků při ukončování běhu systému. Že už můžeme vypnout napájení přes telnet prostě nepoznáme. Hlášku **Power down** uvidíme pouze a jen na systémové konzoli, tj. přes sériový port DBGU.

2.4 Změna konfigurace modulu

Nejprve se budeme zabývat nastavením systému Linux, který je na modulu SAM9260 předinstalován. Pak si ukážeme jak změnit také nastavení zavaděče U-Boot.

V této souvislosti bych chtěl upozornit, že zde není žádné kouzelné tlačítko, jak špatně nakonfigurovaný systém uvést do původního výchozího stavu. Buď dokážete špatnou konfiguraci opravit sami (kromě neznámého hesla superuživatel **root** to vcelku není problém) nebo musíte přehrát kořenový souborový systém v paměti NandFlash (umístění firmware a jeho částí je probráno v jiné kapitole). Je proto potřeba postupovat s rozmyslem.

Konfigurace systému Linux je dána obsahem textových konfiguračních souborů, průběh inicializace a ukončování systému pak ve spouštěcích a ukončovacích skriptech. Tyto soubory a skripty nalezneme v adresáři */etc*. Pro editaci konfiguračních souborů a skriptů je na modulu k dispozici editor *vi*. Jeho ovládání je popsáno v příslušné kapitole. Jen si dovolím upozornit na skutečnost, že místo standardní sady příkazů a utilit je předinstalovaný Linux postaven na bázi Busyboxu, což je ve výsledku binární soubor, který v sobě sdružuje zjednodušenou sadu příkazů a utilit. Některé konfigurační soubory proto mají mírně odlišné způsoby zápisu.

Níže uvádíme seznam základních parametrů systému Linux, které lze změnit:

- **jméno počítače** - anebo také **hostname**. Jako **hostname** se použije obsah souboru */etc/hostname*. Výchozí hodnota je *ucsimplify-sam9260*, která se zobrazuje jako součást přihlašovacího promptu. Změna je platná po restartu.
- **úvodní obrazovka** - úvodní obrazovka pro systémovou konzoli je dána obsahem souboru */etc/issue.res*, pro telnet pak obsahem souboru */etc/issue.net*. Změna je platná po restartu, resp. po otevření nového spojení přes telnet.
- **prostředí po přihlášení** - globální nastavení prostředí, resp. proměnných prostředí, společně pro všechny uživatele je v souboru */etc/profile*. Zde se nastavují různé systémové proměnné jako proměnná **PATH**, délka historie příkazů (proměnná **HISTSIZE**), časová zóna atd. Změna nastavení je platná ihned po opětovném přihlášení.
- **síťové rozhraní** - IP adresa, maska podsítě a brána se nastavují pro síťové rozhraní v souboru */etc/network/interfaces*. DNS servery se nastavují v souboru */etc/resolv.conf*. Změnu nastavení můžeme ihned uvést v platnost vypnutím síťového rozhraní příkazem **ifdown eth0** a opětovným zapnutím příkazem **ifup eth0**.
- **uživatelé, skupiny** - seznam uživatelů je v souboru */etc/passwd*, seznam uživatelských skupin pak v souboru */etc/group*. Tyto soubory lze samozřejmě editovat přímo, nicméně je doporučeno pro manipulaci s uživatelskými účty využívat příkazů **adduser** a **deluser**, stejně jako pro skupiny příkazů **addgroup** a **delgroup**. Změny jsou platné ihned, pro přihlášené uživatele ovšem až po jejich opětovném přihlášení.
- **úroveň běhu, virtuální a sériové konzole** - nastavení, který skript se má pouštět při inicializaci, který při ukončení systému, jak reagovat na stisk kláves Ctrl+Alt+Del, počet a vlastnosti virtuálních a sériových konzolí - to vše se nastavuje v souboru */etc/inittab*. Změny jsou platné po restartu.
- **seznam připojitelných zařízení** - seznam blokových zařízení, na která lze ukládat data (obsahují totiž podporovaný souborový systém) a která může připojit i jiný uživatel než jen **root**, jsou v souboru */etc/fstab*. Sazmořejmě lze připojit i bloková zařízení, která v uvedeném souboru nejsou, ale to může udělat jen a pouze superuživatel **root**. Standardně jsou zde uvedené různé interní zařízení pro potřeby operačního systému, dále zařízení, které obsahuje kořenový souborový systém (rozuměj systémový "disk"), dále různá přenosná úložiště (USB flash disk) apod. Změny pro ještě nepřipojená zařízení jsou platné ihned jinak až po opětovném připojení zařízení.
- **inicializační a ukončovací skripty** - tyto skripty definují kroky, které se vykonávají při startu systému, resp. při jeho ukončování. Uživatel si tak může snadno přizpůsobit poslední část inicializace systému, resp. první část ukončování běhu systému. Typicky se zde připojují interní souborové systémy, startuje se služba pro záznam událostí, aktivují

se síťová rozhraní a další služby. Ukončovací skript bývá obvykle opakem inicializačního skriptu.

Oba skripty včetně dalších podpůrných skriptů jsou umístěny v adresáři `/etc/init.d`. Inicializačním skriptem je skript `rcS`, ukončovacím skriptem pak skript `rcK`. Změny v inicializačním skriptu jsou platné až po restartu, změny v ukončovacím skriptu pak ihned.

Vedle předinstalovaného systému Linux je možné měnit i konfiguraci zavaděče U-Boot, který má na starosti právě zavedení systému Linux. Z nastavení U-Bootu nás bude nejvíc zajímat:

- **nastavení parametrů síťového rozhraní** - parametry síťového rozhraní v U-Bootu se stejně jako ostatní parametry nastavují v proměnných prostředí U-Bootu (viz např. změna MAC adresy). Níže uvádíme příkazy pro změnu jednotlivých parametrů síťového rozhraní:

```
u-boot> setenv ipaddr 192.168.1.20
u-boot> setenv netmask 255.255.255.0
u-boot> setenv gatewayip 192.168.1.254
```

Samozřejmě pro trvalou změnu je nutné uložit proměnné prostředí do DataFlash / Nand-Flash příkazem `saveenv`.

- **nastavení IP adresy TFTP serveru** - TFTP server slouží úložiště dat, např. aktualizací firmware, odkud může U-Bootu stáhnout do RAM paměti nový firmware, který je pak možné z U-Bootu nahrát do modulu. IP adresa TFTP serveru se změní takto:

```
u-boot> setenv serverip 192.168.1.2
```

- **parametry příkazové řádky jádra** - jak už je uvedeno výše, zavaděč U-Boot se stará o zavedení systému Linux, resp. jeho jádra. Mimo to umí i jádru předat parametry, kterými jádru řekneme, kde má hledat kořenový souborový systém atd. O těchto parametrech mluvíme jako o parametrech příkazového řádku jádra a slouží k tomu, abychom mohli měnit parametry bootování jádra, aniž bychom museli jádro znovu překonfigurovat a zkompileovat.

Parametry příkazové řádky jádra jsou uloženy v proměnné `bootargs` prostředí U-Bootu. Níže uvádíme hodnotu této proměnné, získanou příkazem `printenv bootargs`:

```
bootargs=console=ttyS0,115200 ubi.mtd=0 root=ubi0:rootfs rw rootfstype=ubifs
```

Vše za textem `bootargs=` jsou už parametry příkazové řádky jádra. Pokud je chceme změnit, musíme použít odpovídající syntaxi (tedy nikoliv syntaxi U-Bootu, ale jádra). Seznam dostupných parametrů jádra naleznete např. *zde*⁵

2.5 Kořenový souborový systém přes NFS

V rané fázi vývoje může být neustálé nahrávání nového firmware, resp. obrazu kořenového souborového systému do paměti NandFlash časově náročnější a hlavně obtěžující. Každá změna v obsahu kořenového souborového systému (dále jako `rootfs`), např. nový binární soubor vyvíjené aplikace, znamená, že vývojář musí vytvořit nový obraz `rootfs`, smazat paměť NandFlash a vypálit do ní nový obraz `rootfs`.

Alternativou může být použití USB flash disku, z kterého se nová binárka vyvíjené aplikace překopíruje nebo rovnou spustí, případně lze pro stejné účely použít SD/MMC kartu. Toto řešení je ovšem dosti limitující v případě, kdy teprve ladíme samotný obsah `rootfs`. Tehdy a nejen tehdy je výhodné přistupovat na `rootfs` přes NFS.

NFS neboli **Network File System** je speciální souborový systém, který umožňuje sdílet soubory a adresáře mezi počítači prostřednictvím počítačové sítě ethernet. Dokonce je možné na tento tento sdílený adresář přistupovat i ve fázi zavádění linuxového jádra na jiném počítači.

⁵<http://www.kernel.org/doc/Documentation/kernel-parameters.txt>

Prakticky to znamená, že na svém pracovním počítači máte sdílený adresář, jehož obsah představuje kořenový souborový systém modulu SAM9260. Jak modul bootuje a probíhá zavádění linuxového jádra, tak v určitém okamžiku připojí jádro tento sdílený adresář jako kořenový souborový systém. Úplně stejně jako kdyby kořenový souborový systém byl v paměti NandFlash. Z pohledu jádra, ale i z pohledu uživatele v tom není rozdíl. Samozřejmě přístup k rootfs přes NFS je pomalejší, jelikož se obsah rootfs přenáší přes síť ethernet. Je nasnadě, že při použití NFS je přenesení jakákoliv změny v obsahu rootfs okamžité, vždyť modul jen přistupuje přes síť na sdílený adresář v našem pracovním počítači. Takto lze snadno a hlavně rychle odladit obsah rootfs, otestovat funkčnost nové verze aplikace apod. Obraz rootfs pro nahrání do paměti NandFlash vytváříme až v momentě, kdy obsah rootfs považujeme za stabilizovaný.

2.5.1 Počítač jako NFS server

Aby bylo možné nějaký adresář v našem počítači sdílet, musíme ho přidat do seznamu sdílených adresářů služby *NFS server* a určit, kdo k němu bude mít přístup. Náš počítač bude v síti vystupovat jako server, který nabízí ke sdílení nějaké prostředky, v našem případě adresář s obsahem kořenového souborového systému modulu SAM9260.

Poznámka: Níže uvedený postup byl vytvořena odzkoušen pro linuxovou distribuci Debian 6.0 Squeeze. Pro jiné distribuce může být postup mírně odlišný.

Poznámka: Níže uvedené příkazy musíme provádět pod účtem superuživatele root.

Postup:

1. Nejprve ověříme, že je služba *NFS server* nainstalována na našem počítači:

```
# dpkg --get-selections | grep nfs-kernel-server
```

Výstup příkazu:

```
nfs-kernel-server      install
```

Vidíme, že na mém počítači je služba *NFS server* už nainstalována. Pokud tomu u vás tak není, tak ji nainstalujete snadno pomocí příkazů:

```
# apt-get update
# apt-get install nfs-kernel-server
```

Služba *NFS server* bude automaticky spuštěna a nakonfigurována.

2. Dále ověříme, že je služba *NFS server* v našem počítači spuštěna:

```
# service nfs-kernel-server status
```

Výstup příkazu:

```
nfsd running
```

Jak vidíme v mém případě služba *NFS server* běží. Pokud systém odpoví **nfsd not running**, tak je to snadné - stačí službu *NFS server* pustit příkazem:

```
# service nfs-kernel-server start
```

3. V libovolném textovém editoru (např. v editoru souborového manažeru *Midnight Commander*) otevřeme soubor */etc/exports* a přidáme do něj řádek pro sdílení naše adresáře. V příkladu budeme předpokládat, že takovým adresářem je adresář */home/emlin/nfs-share*. Soubor pak uložíme.

Řádek do souboru */etc/exports*:

```
/home/emlin/nfs-share 192.168.1.12(rw, sync, no_subtree_check, no_root_squash)
```


Tento zápis říká, že adresář `/home/emlin/nfs-share` je sdílen přes NFS a přistupovat na něj může počítač s IP adresou 192.168.1.12 v režimu čtení/zápis. Volba `no_subtree_check` vypíná kontrolu, že požadovaný soubor/adresář se nachází uvnitř sdíleného adresáře (urychluje přístup). A volba `no_root_squash` zajišťuje, že superuživatel `root` na modulu bude mít stejná práva jako kdyby byl superuživatelem `root` na našem počítači (normálně je jakýkoliv uživatel z modulu namapován na uživatele `nfsnobody` na pracovním počítači/NFS serveru). A jelikož obsah `rootfs` je vlastněn superuživatelem `root`, tak toto zabezpečení musíme vypnout.

4. Restartujeme službu *NFS server*, aby byl upravený soubor `/etc/exports` znovu načten:

```
# service nfs-kernel-server restart
```

Výstup příkazu:

```
Stopping NFS kernel daemon: mountd nfsd.
Unexporting directories for NFS kernel daemon....
Exporting directories for NFS kernel daemon....
Starting NFS kernel daemon: nfsd mountd.
```

Během restartu služby proběhne i kontrola obsahu souboru `/etc/exports`. V případě nějaké chyby je tato vypsána a služba *NFS server* zůstane pozastavena.

2.5.2 Kořenový adresář modulu přes NFS

Modul SAM9260 se v případě přístupu ke kořenovému souborovému systému (`rootfs`) přes NFS chová jako NFS klient. Tedy jako počítač, který přistupuje ke sdíleným prostředkům NFS serveru. My ale potřebujeme, aby linuxové jádro modulu už při svém zavádění dokázalo přistoupit ke sdílenému adresáři na našem pracovním počítači a připojit tento adresář jako kořenový souborový systém modulu. Stejně jako kdyby byl kořenový souborový systém uložen v oddílu paměti NandFlash. Jinými slovy - musíme jádru přikázat, aby kořenový souborový systém hledalo na NFS serveru (= našem počítači) a ne jako normálně v NandFlash paměti modulu.

Mimo to musíme jádru předat i parametry síťového rozhraní (IP adresu atd.), aby jádro vůbec dokázalo kontaktovat NFS server. Normálně se parametry síťového rozhraní nastavují až po zavedení jádra v inicializačním skriptu, který si tyto parametry vyčte z konfiguračního souboru `/etc/network/interfaces`. To ale v případě kořenového souborového systému na NFS serveru není možné, protože jaksi kořenový souborový systém není připojen a tudíž dostupný. Vzniká zde tedy problém: "Co bylo dřív? Slepice či vejce?". Tento problém se řeší právě tím, že jádro inicializuje síťové rozhraní už před pokusem přistoupit na NFS server.

Když to shrneme: za a) musíme jádru přikázat, že kořenový souborový systém má hledat v adresáři na NFS serveru a za b) musíme mu předat parametry síťového rozhraní. Jak na to? Máme v zásadě jen jednu možnost - použít příkazový řádek jádra. A předat parametry na příkazový řádek jádra umí zavaděč U-Boot.

Postup:

1. Spustíme modul (případně modul restartujeme, pokud už běží) a automatické zavedení jádra zavaděčem U-Boot přerušíme stiskem libovolné klávesy při startu zavaděče.
2. V příkazové řádce zavaděče U-Boot zadáme příkaz:

```
u-boot> setenv bootargs console ttyS0,115200 ip
=192.168.1.12:192.168.1.123:192.168.1.254:255.255.255.0:ucsimply-sam9260::
off root=/dev/nfs rw nfsroot=192.168.1.123:/home/emlin/nfs-share
```

Tím linuxovému jádru předáme informace uložené v proměnné `bootargs` zavaděče U-Boot. Tj. informace o systémové konzole (parametr `console`), o nastavení síťového rozhraní (parametr `ip` ve tvaru *IP adresa modulu : IP adresa NFS serveru : IP adresa brány : síťová maska : hostname modulu :: autoconf*), určení zařízení s `rootfs` (parametr `root`) a určení cesty ke sdílenému adresáři s `rootfs` (parametr `nfsroot` ve tvaru *IP adresa NFS serveru : adresář na NFS serveru*).

- Uložíme změny v proměnné prostředí U-Bootu do paměti DataFlash / NandFlash a resetujeme modul:

```
u-boot> saveenv
u-boot> reset
```

- Modul se resetuje. Hned v úvodu svého zavádění jádro vypíše obsah příkazové řádky jádra, tak jak mu byl předán zavaděčem U-Boot. Ve výpisech jádra bychom tedy měli vidět něco takového:

```
Linux version 2.6.38.8 (root@vpc-debian) (gcc version 4.5.1 (Sourcery G++ Lite
 2 010.09-50) ) #1 PREEMPT Wed May 30 13:38:42 CEST 2012
CPU: ARM926EJ-S [41069265] revision 5 (ARMv5TEJ), cr=00053177
CPU: VIVT data cache, VIVT instruction cache
Machine: uCSimply AT91SAM9260 module
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: 8128
Kernel command line: console ttyS0,115200 ip
=192.168.1.12:192.168.1.123:192.168.1.254:255.255.255.0:ucsimply-sam9260::
off root=/dev/nfs rw nfsroot=192.168.1.123 :/home/emlin/nfs-share
...
```

Tak si lze snadno zkontrolovat, že jsme obsah proměnné U-Bootu **bootargs** zadali správně. Ke konci zavádění jádra, těsně před spuštěním inicializačního skriptu, bychom měli ve výpisech jádra vidět něco takového:

```
IP-Config: Complete:
device=eth0, addr=192.168.1.12, mask=255.255.255.0, gw=192.168.1.254,
  host=ucsimply-sam9260, domain=, nis-domain=(none), bootserver
  =192.168.1.123, rootserver=192.168.1.123, rootpath=
eth0: link up (100/Full)
VFS: Mounted root (nfs filesystem) on device 0:12.
Freeing init memory: 116K
```

Příčemž výpis **VFS: Mounted root (nfs filesystem) on device 0:12.** nás informuje o tom, že jádro úspěšně připojilo sdílený adresář na NFS serveru jako kořenový souborový systém.

Na tomto místě je určitě vhodné zmínit jak vrátit nastavení proměnné U-Boot **bootargs** do původního stavu tak, aby jádro hledalo kořenový souborový systém v oddílu paměti NandFlash. Je to prosté. Nabootujeme do U-Bootu a obsah proměnné přepíšeme výchozím obsahem.

Pokud je obraz linuxového jádra uložen v paměti DataFlash, tak zadáme příkaz:

```
u-boot> setenv bootargs console ttyS0,115200 ubi.mtd=0 root=ubi0:rootfs rw
  rootfstype=ubifs
```

A jestliže je jádro uloženo v paměti NandFlash (stejně jako rootfs), tak zadáme příkaz:

```
u-boot> setenv bootargs console ttyS0,115200 ubi.mtd=4 root=ubi0:rootfs rw
  rootfstype=ubifs
```

Změny trvale uložíme příkazem **saveenv** a modul resetujeme příkazem **reset**. Modul by měl pak už nabootovat normálně, tj. jádro by mělo připojit kořenový souborový systém uložený v paměti NandFlash.

2.5.3 Modul jako NFS klient

Jednou z dalších možností, kdy je výhodné využít NFS, je situace, kdy z pracovního počítače potřebujeme do modulu přenést nějaká data. Není nic jednoduššího než tato data přenést do sdíleného adresáře v pracovním počítači a z modulu si je stáhnout v běžícím Linuxu přes NFS.

V tomto případě pracovní počítač není úložištěm pro kořenový souborový systém modulu, ale prostě jen poskytuje data.

Sdílený adresář pracovního počítače zpřístupníme na modulu stejně jako obsah jakéhokoliv jiného datového média. Prostě jej připojíme příkazem `mount` podobně jako USB flash disk, DVD-ROM atd. Ukážeme si to na jednoduchém příkladu. Předpokládejme, že IP adresa NFS serveru je `192.168.1.123` a cesta ke sdílené složce na NFS serveru je `/home/emlin/nfs-share`. Dále chceme, aby obsah sdíleného adresáře byl v modulu přístupný v adresáři `/mnt/nfs`. Do příkazové řádky Linuxu v modulu SAM9260 zadáme:

```
# mount -t nfs -o nolock 192.168.1.123:/home/emlin/nfs-share /mnt/nfs
```

Pokud jsme byli úspěšní (systém nezareagoval chybovým hlášením), tak je nyní obsah sdíleného adresáře přístupný v adresáři `/mnt/nfs`. Odborně řečeno - obsah sdíleného adresáře jsme připojili k přípojnému bodu `/mnt/nfs`.

K odpojení modulu od sdíleného adresáře pak provedeme příkazem `umount /mnt/nfs`. Tím NFS serveru řekneme, že už nehodláme nadále přistupovat na sdílené adresáře a zároveň zapíšeme všechny změny ve sdílených datech provedené ze strany modulu.

3 Nahrávání firmware

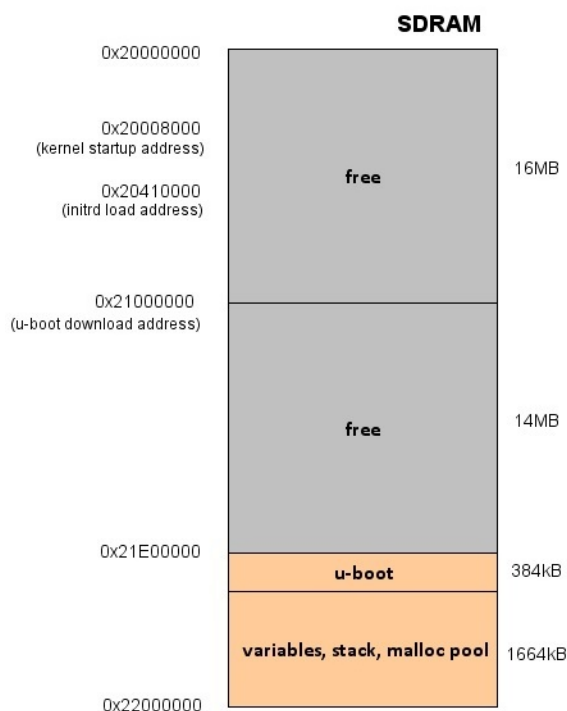
V této části manuálu k modulu SAM9260 vysvětlíme bootovací sekvenci modulu, zprovozníme si na modulu rozhraní USB device, abychom mohli nahrávat firmware přes USB port našeho počítače, představíme si aplikaci SAM-BA pro nahrávání firmware pro mikrokontroléry třídy SAM9 od společnosti Atmel a nakonec si ukážeme, jak s pomocí aplikace SAM-BA nahrát kompletní firmware do paměti DataFlash a/nebo NandFlash.

Úvodem by se asi slušelo vysvětlit, co že to vlastně je ten "firmware". *Firmware* je balíček v podobě jednoho nebo více binárních souborů, který představuje kompletní programové vybavení počítače / zařízení. V případě modulu SAM9260 se připravený firmware skládá z binárního souboru bootstrapu, zavaděče U-Bootu, zkomprimovaného obrazu linuxového jádra a obrazu kořenového souborového systému (něco jako obsah systémového oddílu vašeho počítače).

3.1 Bootování modulu

Bootovací sekvenci, tj. sled úkonů následujících po zapnutí napájení, má na starosti monitor RomBoot, speciální program uložený do vnitřní paměti ROM mikrokontroléru AT91SAM9260. Je to tedy úplně první program, který začne CPU modulu vykonávat po připojení napájení.

Úkolem monitoru RomBoot je inicializovat CPU modulu a zavést z vnitřní nebo vnější ROM paměti uživatelský program. V případě modulu SAM9260 budeme uvažovat pouze tzv. bootstrap, což je primární zavaděč. RomBoot prohledává nejprve paměť DataFlash a pak paměť NandFlash. Jakmile je v některé z nich nalezen platný kód primárního zavaděče (bootstrap), je tento přkopírován do paměti SRAM a spuštěn.



Obrázek 3.1.1: Mapa paměti SDRAM

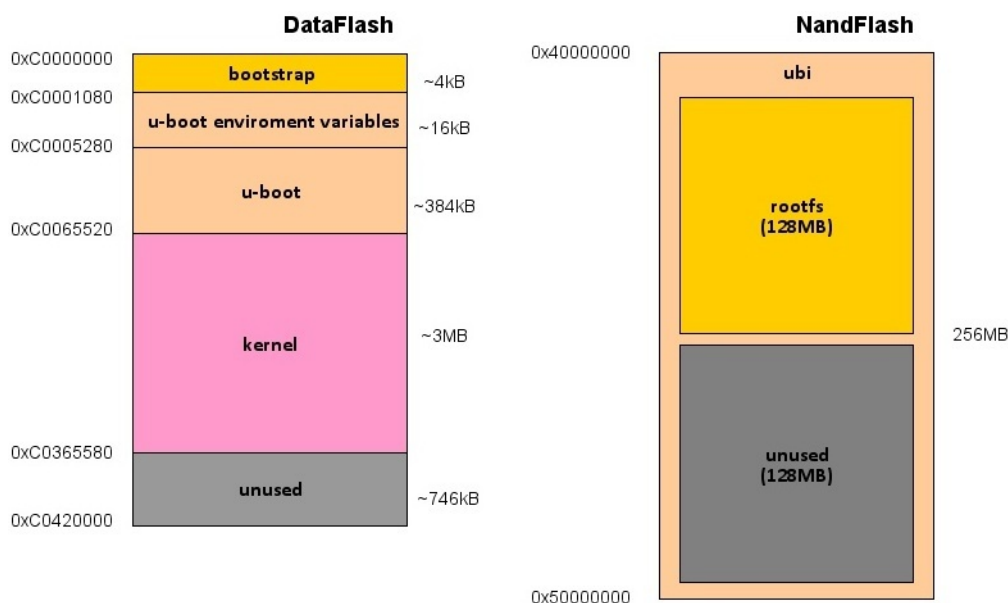
V momentě svého spuštění, přebírá bootstrap plnou kontrolu nad modulem, resp. mikrokontrolérem, a vedle základní inicializace potřebných částí mikrokontroléru, provede svůj hlavní úkol

- inicializuje řadič paměti SDRAM. Bez tohoto kroku by nebylo možné s ohledem na velikost interní paměti SRAM spustit program větší jak 4kB. Jakmile je řadič SDRAM úspěšně inicializován, bootstrap zkopíruje sekundární zavaděč (obvykle odkazovaný jen jako "zavaděč") U-Boot z paměti DataFlash příp. NandFlash na určené místo v paměti SDRAM a předá mu řízení (= spustí jej).

Zavaděč U-Boot v této fázi už žádnou další inicializaci mikrokontroléru neprovádí, pouze si ověří dostupnost a stav jednotlivých paměti a stav CPU. Ve výchozí konfiguraci U-Boot pak po uplynutí časové prodlevy překopíruje z paměti DataFlash příp. NandFlash zkomprimovaný obraz linuxového jádra na vybrané místo v paměti SDRAM a dále případně překopíruje i obraz úvodního RAM disku jádra (initrd), je-li přítomen. U-Boot jádro rozbalí na jeho startovací adresu (kernel startup address) a předá mu řízení.

Linuxové jádro zavede ovladače jednotlivých zařízení (USB, síťové rozhraní apod.) a ke konci svého bootování připojí i zařízení, na kterém je uložen kořenový souborový systém se systémovými soubory a daty. V posledním kroku předá řízení uživatelskému inicializačnímu skriptu. Tento skript uživatelsky definovaným způsobem dokončí zavádění operačního systému a následně spustí příkazový řádek s přihlašovacím dialogem. Tím bootování modulu končí.

Rozdělení paměti SDRAM, tak jak jej předpokládají jednotlivé komponenty firmware modulu je na obrázku 3.1.1



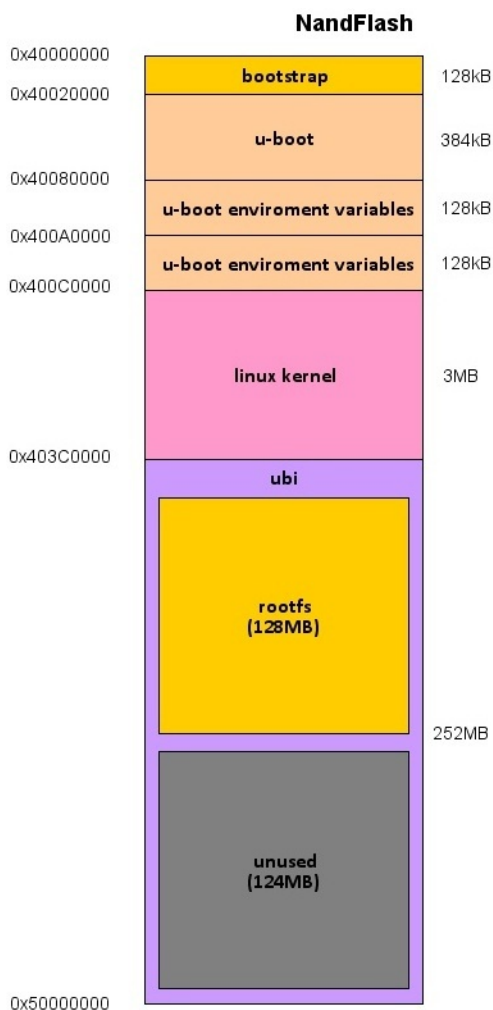
Obrázek 3.1.2: Mapa paměti DataFlash + NandFlash

Již dříve jsme zmínili, že vzhledem k tomu, že modul je standardně vybaven dvěma paměťmi typu ROM, přicházejí v úvahu dvě varianty bootování:

1. Bootování z DataFlash, kdy bootstrap, U-Boot a linuxové jádro jsou uloženy v DataFlash a kořenový souborový systém spolu s uživatelskými data v paměti NandFlash (viz obrázek 3.1.2).
2. Bootování čistě jen z paměti NandFlash, kde je pak uložen kompletní firmware a uživatelská data (viz obrázek 3.1.3).

První varianta má jednoznačně výhodu co se týče rychlosti i větší kapacity paměti NandFlash využitelné pro uložení uživatelských dat. Nehledě na problém při bootování z paměti typu NandFlash kvůli existenci špatných sektorů a opotřebením paměti.

Pozorný čtenář si asi říká, co se děje když RomBoot nenajde platný kód bootstrapu ani v paměti DataFlash ani v paměti NandFlash. Ano, tenhle scénář jsme zatím nezmnili. Jestliže RomBoot nenajde bootstrap ani v jedné z podporovaných externích paměti, tak zinicizuje



Obrázek 3.1.3: Mapa paměti NandFlash

rozhraní USB device a sériový port DBGU a v smyčce čeká na zahájení komunikace ze strany počítače. V případě úspěšného navázání spojení umožní přenést přes jedno z uvedených rozhraní krátký program (do 4kB), který následně spustí v interní paměti SRAM mikrokontroléru. Tento mechanismus je určen především pro účely nahrání firmware do jedné z pamětí typu ROM (Data/NandFlash), kdy zkopírování dat (firmware) přenesených do paměti SDRAM přes USB či sériový port do paměti DataFlash / NandFlash provede onen krátký program, tzv. applet, přenesený po zahájení spojení.

3.2 Rozhraní USB

Zapojení rozhraní USB device spočívá vlastně jen v napojení signálů *DDP* a *DDM* a pinu *PC5* mikrokontroléru AT91SAM9260, resp. modulu SAM9260, na konektor USB device dostupný například na základní desce Baseboard (konektor J19). Pin *PC5* musí být napojen na konektor USB device přes odporový dělič (také dostupný na základní desce).

Signály *DDM* a *DDP* modulu SAM9260 odpovídají datovým signálům *D+* a *D-* USB rozhraní. Sériové odpory požadované USB specifikací pro datové signály jsou už součástí modulu SAM9260, takže se o ně nemusíme starat.

Pin *PC5* spolu s děličem je připojen na +5V přítomných na USB device konektoru (napětí dodávané USB hostem - počítačem) a slouží mikrokontroléru pro detekci připojení k USB hostu (např. náš počítač). Jakmile totiž USB host povolí napájení nového USB device, tak toto USB zařízení musí následně svoji přítomnost na USB sběrnici potvrdit připojením datové linky *D+* přes pull-up odpor na +5V. Mikrokontrolér AT91SAM9260 toto zajišťuje automaticky, pouze

3.3.1 Instalace

Instalátor aplikace v nejnovější verzi je ke stažení se stránek firmy *Atmel*¹. Je nutná registrace. Po stažení program nainstalujeme - stačí spustit instalátor *sam-ba_2.11.exe* a následovat pokynů průvodce instalací.

Po instalaci programu SAM-BA musíme ještě přidat podporu pro náš modul uCSimply SAM9260. Stáhneme si *patch*² a rozbálíme jej. Vznikne tak adresář *samba-add-ucsimply_sam9260-support.patch*. Vstoupíme do něj a celý obsah adresáře *samba-add-ucsimply_sam9260-support.patch* (ne adresář samotný!) nakopírujeme do adresáře, kam jsme si program SAM-BA nainstalovali (výchozí je *C:\Program Files (x86)\ATMEL Corporation\sam-ba_2.11*). Tímto je program SAM-BA připraven na spolupráci s naším modulem SAM9260.

3.3.2 Připojení přes USB

Nyní si zprovozníme spojení mezi programem SAM-BA běžícím na našem počítači a modulem SAM9260 přes USB.

Jako první krok musíme v případě, že máme v modulu bootovatelný firmware, oddělat jumper z pinheadu J7 (DataFlash) nebo z pinheadu J5 (NandFlash) na modulu, abychom zabránili spuštění bootstrapu z DataFlash/NandFlash. Tím dosáhneme toho, že se spustí monitor RomBoot a modul bude připravený na komunikaci s programem SAM-BA. Pak připojíme napájecí napětí a propojíme náš počítač s modulem SAM9260 pomocí USB kabelu A-B.

Operační systém (předpokládáme Windows 7) zareaguje tak, že oznámí připojení nového zařízení. To znamená, že detekce našeho modulu na USB sběrnici proběhla v pořádku. V případě nějaké chyby v zapojení, systém Windows nebude schopn naše zařízení detekovat.

Po úspěšné detekci našeho modulu začne systém Windows 7 hledat odpovídající ovladač na serveru Windows Update. Po delší době systém oznámí, že našel odpovídající ovladač *GPS Camera Detect (COM7)*. Nevadí, nikdo není dokonalý. Ovladač sice s modulem funguje, ale ovladač od Atmelu je ovladač od Atmelu. Ovladač na ten správný změním takto:

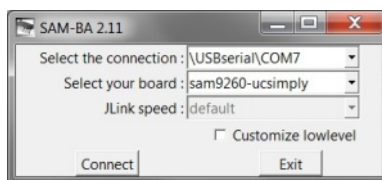
1. Spustíme si správce zařízení: *Start->Ovládací panely->Systém->Správce zařízení*. Náš modul jako zařízení *GPS Camera Detect (COM7)* je v kategorii *Porty (COM a LPT)*:
2. Klikneme na něj pravým tlačítkem a zvolíme *Aktualizovat software ovladače...*
3. V dalším dialogu vybereme *Vyhledat ovladač v počítači* a v dalším dialogu zvolíme *Vybrat ovladač ze seznamu*.
4. V následujícím okně stiskneme tlačítko *Z disku* a zadáme cestu k ovladači *atm6124_cdc.inf* od firmy Atmel, který je v adresáři *<instalační adresář\drv>*, nejčastěji tedy v adresáři *C:\Program Files (x86)\ATMEL Corporation\sam-ba_2.11\drv*.
5. V zadané cestě zvolíme soubor *atm6124_cdc.inf* a dáme *Otevřít*.
6. V původním okně se objeví v textovém okně *Model ovladač AT91 USB to Serial Converter*. Stiskneme tlačítko *Další*.
7. Objeví se okno s varováním, že ovladač není digitálně podepsán - to nám nevadí, proto zvolíme *Přesto nainstalovat tento software ovladače*.
8. Systém nám po chvíli činnosti oznámí, že ovladač zařízení *AT91 USB to Serial Converter* úspěšně nainstalovat, dáme *Zavřít*.

A jsme ve finále. Spustíme program SAM-BA. Může trvat dlouho (řádově i minuty) než se objeví okno SAM-BY, protože SAM-BA se skenuje všechny USB, JTAG a sériové porty a pokouší se najít nějaké AT91 zařízení.

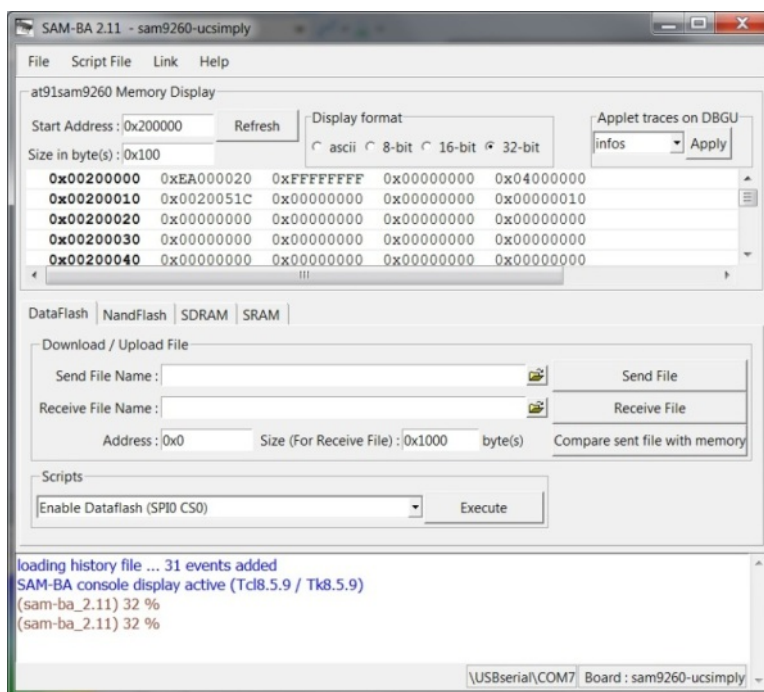
¹<http://www.atmel.com/System/BaseForm.aspx?target=tcm:26-17930>

²<http://www.ucsimply.cz/products/modsam9260/>

3 Nahrávání firmware



Obrázek 3.3.1: SAM-BA - úvodní okno



Obrázek 3.3.2: SAM-BA - hlavní okno

Pokud SAM-BA nějaké zařízení na jednom z komunikačních portů našla, objeví se úvodní okno (viz obrázek 3.3.1). V něm si vybereme komunikační kanál a desku s kterou chceme komunikovat. Jako komunikační rozhraní zvolíme USB, tj. `\\USBserial\\COM7` (číslo se může lišit podle počtu sériových portů ve vašem počítači), a jako desku vybereme náš modul, tj. `sam9260-ucsimply`. Stiskneme tlačítko *Connect*. Objeví se hlavní okno aplikace SAM-BA (viz obrázek 3.3.2).

Poznámka: Abychom zabránili zavedení bootstrapu (primární zavaděč) z paměti Data/NandFlash, tak jsme museli oddělat jumper z pinheadu J7, resp. J5 na modulu. Díky tomu zabudovaný monitor RomBoot přešel do režimu čekání na spojení s aplikací SAM-BA. Jestliže však chceme v aplikaci SAM-BA pracovat s pamětí Data/NandFlash, musíme po spuštění SAM-BY jumper nasadit zpět. Jinak nebude možné paměť Data/NandFlash inicializovat a číst/nahrávat do ní obsah (např. firmware).

3.3.3 Ovládání

Nemáme na mysli stejnojmenný tanec, to opravdu ne, ale program SAM-BA od firmy Atmel pro nahrávání firmware.

Hlavní okno SAM-BY (obrázek 3.3.2) je rozděleno do tří oblastí. První oblast nám ukazuje obsah vnitřních pamětí - ROM, SRAM0 a SRAM1. V druhé části je seznam dostupných pamětí (vnitřních i vnějších) v podobě horizontálních záložek. A konečně třetí část je textové okno, které slouží jako příkazový řádek a jako konzole pro výstup provedených příkazů a skriptů. Úplně dole je pak stavový řádek, který informuje o použitém komunikačním rozhraní a desce, s kterou právě pracujeme.

Nás bude z pohledu nahrávání firmware zajímat především část druhá. Výchozí záložkou je *DataFlash*. Obsah záložek se mění podle typu paměti. V zásadě však je vždy možné do paměti

přenést obsah nějakého souboru (tlačítko *Send File*) nebo naopak obsah paměti uložit do nějakého souboru (tlačítko *Receive File*). Pak ještě můžeme ověřit, že obsah aktuální paměti odpovídá obsahu souboru přeneseného do paměti (tlačítko *Compare sent file with memory*). Při zápisu i při čtení souboru musíme zadat výchozí adresu v poli *Address* v hexadecimálním formátu. Při čtení paměti lze omezit maximální velikost souboru, kam se výpis paměti bude ukládat, na velikost danou hodnotou v poli *Size*. Naši pozornosti by neměl uniknout ani rozbalovací seznam *Scripts*, kde jsou uvedeny všechny možné skripty (naprogramované sekvence příkazů), které jsou pro danou paměť předdefinovány. Například pro paměť DataFlash je to skript *Enable Dataflash (SPI0 CS0)*, který inicializuje paměť DataFlash, takže pak z ní můžeme číst či do ní ukládat firmware. Nebo skript *Erase All*, který vymaže obsah celé DataFlash. Poslední skript určený pro paměť DataFlash je skript *Send Boot File*, který nám umožní zapsat do paměti DataFlash na adresu 0x0 bootstrap (primární zavaděč). Při provádění skriptů jsou do třetí části okna aplikace vypisovány zprávy informující o průběhu vykonávání skriptu nebo o vzniklé chybě.

Podobné skripty jako pro paměť DataFlash najdeme i pro paměť NandFlash. Pro paměť SDRAM je vytvořen pouze jeden skript - *Enable SDRAM*, který provede opětovnou inicializaci řadiče paměti SDRAM. Pro paměť SRAM není připraven žádný skript.

3.3.4 Typy a triky

Níže uvádíme poznatky z našeho používání aplikace SAM-BA:

- Před novým spuštěním SAM-BY vždy resetujeme modul!
- Ujistíme se, že máme oddělán jumper z pinheadu J7, resp. J5 na modulu, takže se po zapnutí napájení nezavede automaticky bootstrap z Data/NandFlash paměti, ale zabudovaný monitor RomBoot přejde do režimu čekání na spojení s aplikací SAM-BA.
- Běžící bootstrap, resp. obecně probíhající bootování modulu z Data/NandFlash, nebo kvazistav modulu způsobený neprovedením resetu před novým spuštěním SAM-BY často vyvolá při spuštění SAM-BY chybové hlášení *Error in startup script* s výpisem kódu TCL skriptu.

3.4 Nahrání firmware

Z výroby je v modulu nahraný kompletní firmware v poslední verzi platné v době vyrobení modulu. Modul je proto možné ihned používat. Nicméně kvůli možnosti aktualizace nebo možnosti si nahrát vlastní upravený firmware si ukážeme jak nahrát kompletní firmware.

Teď je zřejmě nejvhodnější doba si říct jak je strukturován balíček firmware pro modul SAM9260. Balíček firmware je distribuován jako soubor archivu typu zip se jménem např. *ucsimply-sam9260-fw-00.10.zip*, kde dvě číselné dvojice označují verzi firmware. Čím je toto číslo větší, tím je firmware novější. Po rozbalení archivu vznikne stejnojmenný adresář, např. *ucsimply-sam9260-fw-00.10*, s tímto obsahem:

- podadresář *boot-dataflash* obsahuje binární soubory bootstrapu (*dataflash_ucsimply_sam9260.bin*) a zavaděče U-Bootu (*uboot-dataflash.bin*) určené pro bootování z paměti DataFlash. Podobně podadresář *boot-nandflash* obsahuje bootstrap a U-Boot pro bootování z paměti NandFlash.
- podadresář *kernel* obsahuje binární obraz linuxového jádra, např. soubor s názvem *2.6.38.8-ucsimply_sam9260.bin*.
- podadresář *rootfs* obsahuje soubor *ubi-nandflash.bin* s obrazem kořenového souborového systému, který je určen pro nahrání do paměti NandFlash. Tento obraz obsahuje vlastně dva oddíly souborového systému UBI/UBIFS. Jeden oddíl o velikosti 128MB obsahuje kompletní kořenový souborový systém pro Linux. Druhý je prázdný a doplňuje zbývající volnou kapacitu paměti NandFlash. Je určen pro uložení uživatelských dat v prostředí systému Linux.

Nejnovější firmware pro modul SAM9260 je dostupný *zde*³.

3.4.1 Předpoklady

Abýcho modli do modulu SAM9260 úspěšně nahrát firmware, je potřeba, aby byly splněny následující předpoklady:

- Modul je připojen na zdroj napětí 5V / min. 1A.
- Je zapojeno rozhraní USB device modulu (viz návod výše).
- Je zapojeno DBGU (sériové) rozhraní modulu (viz návod v sekci Začínáme).
- Je zapojeno ethernetové rozhraní modulu (viz návod v sekci Začínáme).

3.4.2 Nahrání firmware do DataFlash

V tomto odstavci si ukážeme jak se nahrává/aktualizuje firmware, když bootstrap, U-Boot a linuxové jádro jsou uloženy v paměti DataFlash a kořenový souborový systém v paměti NandFlash. Tato varianta je použita i ve výrobním nastavení.

Předpokládáme, že modul je odpojen od napájecího napětí a že v adresáři *D:\Temp* je rozbalený balíček firmware.

Postup:

1. Sundáme jumper z pinheadu J7 na modulu a zapneme napájecí napětí.
2. Spustíme aplikaci SAM-BA
3. V úvodní obrazovce vybereme spojení přes *\USBserial\COM7* a desku *sam9260-ucsimply*. Stiskneme tlačítko *Connect*.
4. Nasadíme zpět jumper na pinhead J7 na modulu.
5. Inicializujeme paměť DataFlash - v hlavním okně SAM-BY na záložce *DataFlash* v druhé části obrazovky vybereme skript *Enable Dataflash (SPI0 CS0)* a stiskneme tlačítko *Execute*.

SAM-BA zareaguje výpisy o provádění skriptu:

```
-I- DATAFLASH::Init 0 (trace level : 4)
-I- Loading applet applet-dataflash-sam9260.bin at address 0x20000000
-I- Memory Size : 0x420000 bytes
-I- Buffer address : 0x20002C30
-I- Buffer size: 0x18C0 bytes
-I- Applet initialization done
```

Tím nám SAM-BA oznamuje, že DataFlash byla úspěšně zinicializována. Pokud jsme ovšem zapomněli nasadit zpět jumper nebo je špatně ještě něco jiného, obdržíme chybové hlášení:

```
-E- Script error: Error Initializing DataFlash Applet (Can't detect known device).
```

6. Do DataFlash nahrajeme bootstrap - vybereme skript *Send Boot File* a stiskneme tlačítko *Execute*. V následujícím dialogu vybereme soubor *dataflash_ucsimply_sam9260.bin* a dáme *Otevřít*. SAM-BA následně zapíše bootstrap do DataFlash, o čemž nás informuje výpisy v textovém poli:

```
GENERIC::SendFile D:/Temp/ucsimply-sam9260-fw-00.10/boot-dataflash/
dataflash_ucsimply_sam9260.bin at address 0x0
-I- File size : 0xDD0 byte(s)
-I- Writing: 0xDD0 bytes at 0x0 (buffer addr : 0x20002C30)
-I- 0xDD0 bytes written by applet
```

³<http://www.ucsimply.cz/products/modsam9260/>

7. Do DataFlash nahrajeme U-Boot - do pole *Send File Name* napíšeme umístění souboru *uboot-dataflash.bin*, příp. použijeme tlačítko s obrázkem adresáře pro určení cesty k souboru. Do pole *Address* zadáme adresu *0x5280* (viz obrázek 3.1.2) a stiskneme tlačítko *Send File*. SAM-BA začne zapisovat binární soubor U-Bootu do DataFlash. Ukončení zápisu poznáme podle toho, že hlášení uvozené **-I-** jsou zakončeny promptem (**sam-ba_2.11**) **32 %** (procenta se mohou lišit) a že zmizí malé okýnko *Please wait*.
8. Do DataFlash nahrajeme linuxové jádro - postup je stejný jako v předchozím bodu, jen do pole *Address* zadáme tentokrát hodnotu *0x65520* a samozřejmě vybereme binární soubor obrazu jádra, např. *2.6.38.8-ucsimply_sam9260.bin*.
9. Přejdeme na záložku NandFlash, kde vybereme skript Enable NandFlash a dáme jej provést. SAM-BA zareaguje hlášením:


```
-I- NANDFLASH::Init (trace level : 4)
-I- Loading applet applet-nandflash-sam9260.bin at address 0x20000000
-I- Memory Size : 0x10000000 bytes
-I- Buffer address : 0x20003DA4
-I- Buffer size: 0x20000 bytes
-I- Applet initialization done
```
10. Do NandFlash nahrajeme obraz kořenového souborového systému - v poli *Send File Name* vybereme cestu k souboru *ubi-nandflash.bin*, pole *Address* necháme na hodnotě *0x0* a dáme *Send File*.

Poznámka: Pokud při vytváření obrazu kořenového souborového systému měníme počet a/nebo velikost oddílů souborového systému UBI/UBIFS, tak musíme nejprve celou paměť NandFlash smazat pomocí skriptu Erase All a teprve pak nahrát obraz kořenového souborového systému.
11. Zavřeme aplikaci SAM-BA a vypneme napájecí napětí.
12. Spustíme si terminálový program (např. Putty), abychom přes sériový port mohli sledovat bootovací zprávy jednotlivých komponent firmware (předpokládáme, že modul má zapojen sériový port DBGU na převodník TTL/RS232 a konektor CAN9F).
13. Zapneme napájecí napětí a sledujeme bootování modulu. Pokud U-Boot zobrazí za výpisem oddílů DataFlash zprávu: ***** Warning - bad CRC, using default environment**, je nutné přerušit bootování stiskem libovolné klávesy, změnit výchozí MAC adresu na svoji vlastní a uložit proměnné prostředí U-Bootu do DataFlash:
 - a) Do promptu zavaděče U-Boot napíšeme příkaz pro změnu MAC adresy (nová MAC adresa bude např. 02:40:50:60:70:90):


```
u-boot> setenv ethaddr 02:40:50:60:70:90
```
 - b) Změnu uložíme trvale do DataFlash příkazem:


```
u-boot> saveenv
```
 - c) Resetujeme module příkazem:


```
u-boot> reset
```
 - d) Pokud varovná zpráva zavaděče U-Boot zmizela, byli jsme úspěšní.
14. Počkáme až modul kompletně nabootuje a přihlásíme se do Linuxu, uživatel **root**, heslo **toor**.

3.4.3 Nahrání firmware do NandFlash

Tento odstavec je určen pro všechny, kteří chtějí používat pro uložení kompletního firmware modulu pouze NandFlash. Tj. když bootstrap, U-Boot, linuxové jádro a obraz kořenového souborového systému jsou pouze a jen v paměti NandFlash.

Předpokládáme, že modul je odpojen od napájecího napětí a že v adresáři *D:\Temp* je rozbalený balíček firmware.

Postup:

1. Sundáme jumperu z pinheadu J5 a pinheadu J7 na modulu a zapneme napájecí napětí.
2. Spustíme aplikaci SAM-BA
3. V úvodní obrazovce vybereme spojení přes *\USBserial\COM7* a desku *sam9260-ucsimply*. Stiskneme tlačítko *Connect*.
4. Nasadíme zpět jumper na pinhead J5 na modulu (pinhead J7 pro DataFlash necháme otevřený).
5. Inicializujeme paměť NandFlash - v hlavní okně SAM-BY na záložce *NandFlash* v druhé části obrazovky vybereme skript *Enable Nandflash* a stiskneme tlačítko *Execute*.

SAM-BA zareaguje výpisy o provádění skriptu:

```
-I- NANDFLASH::Init (trace level : 4)
-I- Loading applet applet-nandflash-sam9260.bin at address 0x20000000
-I- Memory Size : 0x10000000 bytes
-I- Buffer address : 0x20003DA4
-I- Buffer size: 0x20000 bytes
-I- Applet initialization done
```

Tím nám SAM-BA oznamuje, že NandFlash byla úspěšně zinicizována. Pokud jsme ovšem zapoměli nasadit zpět jumper, tak se skript sice jakoby provede, ale aplikace SAM-BA celá zamrzne a pomůže ji jen ukončit přes správce úloh.

6. Do NandFlash nahrajeme bootstrap - vybereme skript *Send Boot File* a stiskneme tlačítko *Execute*. V následujícím dialogu vybereme soubor *nandflash_ucsimply_sam9260.bin* a dáme *Otevřít*. SAM-BA následně zapíše bootstrap do NandFlash, o čemž nás informují výpisy v textovém poli:

```
GENERIC::SendFile D:/Temp/ucsimply-sam9260-fw-00.10/boot-nandflash/
nandflash_ucsimply_sam9260.bin at address 0x0
-I- File size : 0xF54 byte(s)
-I- Writing: 0xF54 bytes at 0x0 (buffer addr : 0x20003DA4)
-I- 0xF54 bytes written by applet
```

7. Do NandFlash nahrajeme U-Boot - do pole *Send File Name* napíšeme umístění souboru *uboot-nandflash.bin*, příp. použijeme tlačítko s obrázkem adresáře pro určení cesty k souboru. Do pole *Address* zadáme adresu *0x20000* (viz obrázek 3.1.3) a stiskneme tlačítko *Send File*. SAM-BA začne zapisovat do DataFlash binární soubor zavaděče U-Boot. Ukončení zápisu poznáme podle toho, že hlášení uvozené *-I-* jsou zakončeny promptem (*sam-ba_2.11*) *32 %* (procenta se mohou lišit) a že zmizí malé okýnko *Please wait*.
8. Do DataFlash nahrajeme linuxové jádro - postup je stejný jako v předchozím bodu, jen do pole *Address* zadáme tentokrát hodnotu *0xC0000* a samozřejmě vybereme binární soubor obrazu jádra, např. *2.6.38.8-ucsimply_sam9260-nand.bin*.
9. Do NandFlash nahrajeme obraz kořenového souborového systému - v poli *Send File Name* vybereme cestu k souboru *ubi-nandflash.bin*, pole *Address* necháme na hodnotě *0x3C0000* a dáme *Send File*.

Poznámka: Pokud při vytváření obrazu kořenového souborového systému měníme počet a/ nebo velikost oddílů souborového systému UBI/UBIFS, tak musíme nejprve celou paměť NandFlash smazat pomocí skriptu Erase All a teprve pak nahrát obraz kořenového souborového systému, resp. znovu kompletní firmware.

10. Zavřeme aplikaci SAM-BA a vypneme napájecí napětí.
11. Spustíme si terminálový program (např. Putty), abychom přes sériový port mohli sledovat bootovací hlášky jednotlivých komponent firmware (předpokládáme, že modul má zapojen sériový port DBGU na převodník TTL/RS232 a na konektor CAN9F).
12. Zapneme napájecí napětí a sledujeme bootování modulu. Pokud U-Boot zobrazí hlášku: ***** Warning - bad CRC, using default environment**, je nutné přerušit bootování stiskem libovolné klávesy, změnit výchozí MAC adresu na svoji vlastní a uložit proměnné prostředí U-Bootu do NandFlash:
 - a) Do promptu U-Bootu napíšeme příkaz pro změnu MAC adresy (nová MAC adresa bude např. 02:40:50:60:70:90):

```
u-boot> setenv ethaddr 02:40:50:60:70:90
```
 - b) Změnu uložíme trvale do NandFlash příkazem:

```
u-boot> saveenv
```
 - c) Resetujeme module příkazem:

```
u-boot> reset
```
 - d) Jestliže jsme byli úspěšní, varovná hláška U-Bootu zmizela.
13. Počkáme až modul kompletně nabootuje a přihlásíme se do Linuxu, uživatel **root**, heslo **toor**.

4 Vlastní firmware

Tato sekce je určena pro ty, kteří jdou rádi hlouběji pod povrch problému a mají chuť dané problematice více porozumět. Zde se ti zvědavější dozví, jak si sestavit vlastní firmware, resp. jeho komponenty.

4.1 Nástroje

Níže uvádíme seznam potřebných nástrojů, které si zájemce o sestavení vlastního firmware musí stáhnout a nainstalovat.

4.1.1 devkitARM release 32

Balík vývojových nástrojů pro příkazovou řádku pro procesory ARM. Běží na operačním systému Windows. Je potřebný pro kompilaci bootstrapu (primární zavaděč), kde dává nejmenší velikost výsledného binárního souboru. S novějšími verzemi bohužel nelze zkompileovat bootstrap pro bootování z NandFlash paměti. Ke stažení [zde](#)¹.

Instalace není potřeba. Pouze se spustí exe soubor, dojde k rozbalení archivu a pak stačí do systémové proměnné **PATH** přidat cestu k podadresáři bin v adresáři devkitARMu.

4.1.2 UnixUtils

Balík utilit známých ze světa Unixu. Nejdůležitější utilitou potřebnou z tohoto balíku je utilita make. Tato utilita umožňuje sestavení programu podle předpisu zvaného Makefile. Bez této utility není možné sestavit bootstrap.

Instalace balíku linuxových utilit je jednoduchá – stačí si stáhnout zkomprimovaný archiv *odsud*² a rozbalit jej do cílového adresáře, např.: `C:\Devel`. Vznikne tak adresář `C:\Devel\UnixUtils`.

Pak musíme nainstalovat utilitu fugující make (ta v UnixUtils zlobí). Stáhneme si instalátor `make-3.81.exe` *odsud*³ a spustíme jej. Vše potvrdíme, jen u výběru cílového adresáře nastavíme adresář `C:\Devel\GnuWin32`. Výsledkem našeho snažení je `C:\Devel\GnuWin32`, kde v podadresáři `\bin` je samotná utilita `make.exe` spolu s běhovými knihovnami.

Nyní je třeba utilitu *make.exe* v balíčku UnixUtils nahradit stejnojmennou utilítkou z balíčku GnuWin32. Nakopírujeme tedy soubory *make.exe*, *libiconv2.dll* a *libintl3.dll* z adresáře `C:\Devel\GnuWin32\bin` do adresáře `C:\Devel\UnixUtils\usr\local\wbin`.

Abychom nemuseli při volání těchto utilit vždy zadávat plnou cestu, tak upravíme obsah proměnné **PATH** systému Windows. Zvolíme tlačítko *Start-> Ovládací panely-> Systém a zabezpečení-> Systém*, z nabídek vlevo pak vybereme nabídku *Upřesnit nastavení systému* a v otevřeném okně klikneme na tlačítko *Proměnné prostředí*. Budeme měnit hodnotu proměnné **PATH** pro daného uživatele. Označíme tedy proměnou **PATH** a klikneme na tlačítko *Upravit*. Do pole *Hodnota proměnné* vložíme na první místo před původní položky hodnotu `C:\Devel\UnixUtils\bin; C:\Devel\UnixUtils\usr\local\wbin`; a potvrdíme.

4.1.3 PC s Linuxem

Pro kompilaci zavaděče U-Boot a linuxového jádra potřebujeme počítač, kde poběží Linux, protože kompilace jádra je v prostředí MS Windows úkol skutečně náročný až nemožný. Stejně tak

¹<http://www.ucsimplify.cz/products/modsam9260/>

²<http://www.ucsimplify.cz/resources/downloads/toolchains/unixutils.zip>

³<http://www.ucsimplify.cz/resources/downloads/toolchains/make-3.81.exe>

nástroje pro sestavení obrazu kořenového souborového systému jsou k dispozici pouze a jen pro Linux.

Pro ty z vás, kteří se Windows jako pracovního desktopového prostředí nehodláte vzdát, nabízím alternativu - virtuální počítač s nainstalovaným Linuxem (distribuce Debian 6.0 Squeeze). Návod na vytvoření virtuálního PC je *zde*⁴.

4.1.4 Sourcery CodeBench Lite Edition

Balík volně dostupných vývojových nástrojů od firmy Mentor Graphics (dříve CodeSourcery) pro příkazovou řádku ve verzi jak pro Windows, tak pro Linux. My budeme využívat variantu pro Linux, kterou nainstalujeme do našeho linuxového desktopu, případně do virtuálního počítače s Linuxem. Pro procesory ARM se dodávají dvě varianty vývojových nástrojů:

- ARM EABI - varianta pro holé CPU, se kterou budeme kompilovat zavaděč U-Boot.
- GNU/Linux - varianta pro Linux, se kterou budeme kompilovat linuxové jádro. Tato verze se také používá pro kompilaci všech aplikací, které mají běžet na modulu SAM9260 v prostředí Linuxu (např. i Busybox).

Budeme potřebovat obě varianty. Postupy pro přípravu vlastního firmware, které uvádím níže, jsou ověřeny pro verze ARM EABI: 2010.09-51 (ke stažení *zde*⁵) a GNU/Linux: 2010.09-50 (ke stažení *zde*⁶). Novější verze jsou případně ke stažení *zde*⁷ (ARM EABI) a *zde*⁸ (GNU/Linux).

Instalace v Linuxu je jednoduchá - stačí rozbalit tar archiv do vašeho domovského adresáře `/home` nebo do adresáře `/usr/local` a aktualizaci proměnné `PATH`. Podrobnější návod najdete *zde*⁹.

4.1.5 Knihovna ncurses

Vývojářská verze knihovny ncurses je potřeba pro konfiguraci linuxového jádra, protože vůči ní se sestavuje grafický konfigurátor jádra. Instalaci provedeme nejnázem instalací příslušného balíčku ve verzi pro vývojáře. V případě použití distribuce Debian stačí do příkazové řádky zadat jako superuživatel (root) tento příkaz:

```
# apt-get install libncurses5-dev
```

V případě jiných distribucí bude příkaz záviset na použitém balíčkovacím systému a správci balíčků. Samozřejmě lze využít i grafického instalátoru.

4.2 Bootstrap

Zdrojové kódy bootstrapu verze 1.16 včetně podpory pro modul uCSimply SAM9260 si stáhnete *zde*¹⁰. Originál si můžete stáhnout přímo ze stránek firmy *Atmel*¹¹.

Kompilaci budeme provádět v operačním systému Windows s pomocí balíku nástrojů devkitARM release 32 (novější verze vrátí při kompilaci bootstrapu pro NandFlash chybu).

Postup:

1. Rozbalíme archivní soubor *Bootstrap1.16-ucsimply.zip*. Vznikne adresář *Bootstrap-v1.16-ucsimply*.

⁴<http://www.ucsimply.cz/elnx/pocitac-programatora/jak-na-virtualni-pocitac/>

⁵<http://www.ucsimply.cz/products/modsam9260/>

⁶<http://www.ucsimply.cz/products/modsam9260/>

⁷<http://www.mentor.com/embedded-software/sourcery-tools/sourcery-codebench/platforms/arm-eabi>

⁸<http://www.mentor.com/embedded-software/sourcery-tools/sourcery-codebench/platforms/arm-gnulinux>

⁹<http://www.ucsimply.cz/elnx/kompilator-gcc/krizovy-kompilator/>

¹⁰<http://www.ucsimply.cz/products/modsam9260/>

¹¹<http://www.atmel.com/Images/AT91Bootstrap1.16.zip>

2. Spustíme si okno s příkazovým řádkem DOSu a vstoupíme do adresáře *Bootstrap-v1.16-ucsimply\board\ucsimply_sam9260*:


```
> cd <cesta k adresari>\Bootstrap-v1.16-ucsimply\board\ucsimply_sam9260
```
3. Podle typu paměti, pro kterou chceme bootstrap kompilovat, vstoupíme buď do podadresáře *dataflash* nebo *nandflash*.
4. Smažeme výsledky případné předchozí kompilace:


```
> make clean
```
5. Spustíme kompilaci zvolené varianty bootstrapu:


```
> make
```
6. Výsledek kompilace soubor *dataflash_ucsimply_sam9260.bin*, resp. *nandflash_ucsimply_sam9260.bin*, najdeme v aktuálním adresáři, tj. podadresáři *dataflash*, resp. *nandflash*.

Jen krátce ke změnám ve zdrojovém kódu bootstrapu s ohledem na přidání podpory pro modul uCSimply SAM9260. Do podadresáře *board* přibyl adresář *ucsimply_sam9260*. V tomto adresáři je umístěn kód pro náš modul. Jde o soubor *ucsimply_sam9260.c*, který je společný pro obě varianty bootstrapu (*dataflash* i *nandflash*). *Makefile* a konfigurační soubor bootstrapu *ucsimply_sam9260.h* je umístěn pro danou variantu bootstrapu v podadresáři *dataflash* a v podadresáři *nandflash*.

4.3 Zavaděč U-Boot

U-Boot budeme kompilovat v prostředí systému Linux s použitím kompilátoru ARM EABI z balíku CodeBench Lite Edition.

Zdrojové kódy zavaděče U-Boot verze 2011.12 s přidanou podporou pro modul uCSimply SAM9260 si můžete stáhnout *zde*¹². Originál najdete *zde*¹³.

Postup:

1. Otevřeme si emulátor terminálu nebo se přepneme do konzole Linuxu.
2. Rozbalíme archivní soubor *u-boot-2011.12-ucsimply.tar.bz2*, čímž vznikne adresář *u-boot-2011.12-ucsimply*:


```
$ tar xjf u-boot-2011.12-ucsimply.tar.bz2
```
3. Vstoupíme do vzniklého adresáře:


```
$ cd u-boot-2011.12-ucsimply
```
4. Nastavíme proměnné pro křížovou kompilaci U-Bootu:


```
$ export ARCH=arm
$ export CROSS_COMPILE=arm-none-eabi-
```
5. Smažeme výsledky případné předchozí kompilace:


```
$ make distclean
```
6. Nakonfigurujeme U-Boot pro kompilaci pro paměť DataFlash, resp. NandFlash:


```
$ make ucsimply_sam9260_dataflash_config
```

¹²<http://www.ucsimply.cz/products/modsam9260/>

¹³<ftp://ftp.denx.de/pub/u-boot/u-boot-2011.12.tar.bz2>

resp.

```
$ make ucsimply_sam9260_nandflash_config
```

7. Spustíme samotnou kompilaci:

```
$ make
```

8. Výsledek kompilace, soubor *u-boot.bin*, je dostupný v kořenovém adresáři U-Bootu - *u-boot-2011.12-ucsimply*.

4.4 Linuxové jádro (kernel)

Standardní zdrojový kód linuxového jádra neobsahuje ani podporu pro modul uCSimpy SAM9260 ani pro mikrokontroléry řady SAM9 od firmy Atmel. Tato podpora je řešena pomocí patchů (úprav), které se aplikují na zdrojový kód jádra před konfigurací a kompilací jádra.

Linuxové jádro budeme kompilovat nástroji GNU/Linux z balíku CodeBench Lite Edition. Následující návod je platný pro jádro verze 2.6.38. U novějších jader je pravděpodobný problém s aplikací patchů.

Než začneme:

- Nainstalujeme si balíček *uboot-mkimage* (pokud již není nainstalovaný), který je potřeba pro překonvertování výsledného binárního souboru jádra do podoby srozumitelné pro zavaděč U-Boot. Instalaci provedeme pod účtem superuživatele **root**:

```
# apt-get update
# apt-get install uboot-mkimage
```

Postup:

1. Otevřeme si emulátor terminálu nebo se přepneme do konzole Linuxu.
2. Stáhneme si *patch at91*¹⁴, který přidá podporu pro mikrokontroléry řady SAM9, a *patch ucsimply-sam9260*¹⁵, který přidá podporu pro modul uCSimpy SAM9260. Oba patche jsou kompatibilní pouze s verzí jádra 2.6.38. Patch *ucsimply-sam9260* lze aplikovat pouze na jádro 2.6.38 s patchem *at91*.
3. Do adresáře, kde jsou stažené patche, si stáhneme zdrojové kódy jádra 2.6.38 přímo od vývojářů:

```
$ wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.38.8.tar.bz2
```

Poznámka: archivní soubor s jádrem se stáhne do aktuálního adresáře.

4. Rozbalíme stažený archivní soubor jádra. Vznikne adresář *linux-2.6.38.8*.


```
$ tar xjf linux-2.6.38.8.tar.bz2
```
5. Na zdrojový kód jádra aplikujeme AT91 patch (pro podporu mikrokontrolérů SAM9):


```
$ cd linux-2.6.38.8/
$ zcat ../2.6.38-at91.patch.gz | patch -p1
```

6. Aplikujeme patch pro modul uCSimpy SAM9260:

```
$ zcat ../2.6.38-add-ucsimply_sam9260-support.patch.gz | patch -p1
```

¹⁴<http://maxim.org.za/AT91RM9200/2.6/2.6.38-at91.patch.gz>

¹⁵<http://www.ucsimply.cz/products/modsam9260/>

7. Nastavíme si proměnné prostředí pro křížovou kompilaci jádra:

```
$ export ARCH=arm
$ export CROSS_COMPILE=arm-none-linux-gnueabi-
```

8. Provedeme výchozí konfiguraci jádra pro modul SAM9260:

```
$ make ucsimply_sam9260_defconfig
```

9. Případně provedeme úpravu konfigurace jádra oproti výchozí konfiguraci pro modul SAM9260:

```
$ make menuconfig
```

Poznámka: Pokud chceme jádro pro scénář NAND_ONLY, kdy je celý firmware kompletně pouze v NandFlash, tak musíme zatrhnout volbu:

System Type->Atmel AT91 System-on-Chip->uCSimply AT91SAM9260 module:

[] NandFlash only mtd partition layout*

10. Zkompilujeme jádro do podoby komprimovaného souboru, tzv. *uImage*, který podporuje U-Boot:

```
$ make uImage -j2
```

Při úspěšném dokončení kompilace jádra obdržíme hlášení:

```
SYMAP System.map
SYMAP .tmp_System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
AS arch/arm/boot/compressed/head.o
GZIP arch/arm/boot/compressed/piggy.gzip
CC arch/arm/boot/compressed/misc.o
CC arch/arm/boot/compressed/decompress.o
SHIPPED arch/arm/boot/compressed/lib1funcs.S
AS arch/arm/boot/compressed/lib1funcs.o
AS arch/arm/boot/compressed/piggy.gzip.o
LD arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
UIMAGE arch/arm/boot/uImage
Image Name: Linux-2.6.38.8
Created: Thu Jun 14 16:56:24 2012
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 1872748 Bytes = 1828.86 kB = 1.79 MB
Load Address: 0x20008000
Entry Point: 0x20008000
Image arch/arm/boot/uImage is ready
```

11. Výsledkem našeho snažení je soubor *arch/arm/boot/uImage*. Tento soubor je obraz jádra, který je U-Boot schopen zavést z paměti Data/NandFlash. Stačí jej nahrát do Data/NandFlash a můžeme bootovat do Linuxu.

*Poznámka: Produkty kompilace jádra můžeme smazat příkazem **make distclean**. Úklid zatím odložíme a provedeme jej až po sestavení kořenového souborového systému.*

4.5 Kořenový souborový systém

Na závěr si připravíme obsah a následně i obraz kořenového souborového systému. Jinými slovy vlastně obsah systémového pevného disku, kde jsou uloženy příkazy, utility, konfigurační soubory, jádro, ovladače a další soubory. A z toho všeho pak vytvoříme binární obraz vhodný pro nahrání přímo do paměti NandFlash, která v případě modulu slouží jako systémový pevný disk.

4.5.1 Příprava obsahu

Obsah kořenového souborového systému (dále rootfs) je vlastně to, co tvoří distribuci Linuxu. Můžeme tedy hrdě prohlásit, že si teď sestavíme vlastní linuxovou distribuci pro modul uCSimply SAM9260. Naše distribuce stojí na čtyřech pilířích:

- Příkazy a utility vyřešíme pomocí Busyboxu.
- Ovladače, kterou nejsou zakompilované do jádra, získáme kompilací ze stromu jádra.
- Systémové knihovny jsou součástí balíku křížových nástrojů CodeBench Lite Edition, stačí je jen zkopírovat.
- Konfigurační soubory a skripty si vytvoříme sami podle návodu.

Poznámka: Pro kompilaci Busyboxu použijeme nástroje GNU/Linux z balíku Sourcery CodeBench Lite Edition. Návod předpokládá Busybox verze 1.19.4, pro který je připraven patch s výchozí konfigurací pro modul uCSimply SAM9260.

Postup:

1. Nejdříve si připravíme kostru rootfs (strukturu adresářů), zkopírujeme si do rootfs systémové knihovny a připravíme si konfigurační soubory a skripty. Návod jak na to naleznete na našich stránkách *zde*¹⁶.

Poznámka: Systémové knihovny vhodné pro modul SAM9260 najdeme v instalačním adresáři balíku Sourcery CodeBench Lite Edition v podadresáři arm-none-linux-gnueabi/arm-none-linux-gnueabi/libc/lib.

2. Dále si připravíme Busybox:

- a) Stáhneme si archiv se zdrojovými kódy Busyboxu v 1.19.4 *odsud*¹⁷.
- b) Stáhneme si *patch*¹⁸, který přidává do Busyboxu výchozí konfiguraci pro modul SAM9260.
- c) Rozbalíme archiv z Busyboxem. Vznikne adresář busybox-1.19.4.:

```
$ tar xjf busybox-1.19.4.tar.bz2
```

- d) Aplikujeme patch pro modul SAM9260:

```
$ cd busybox-1.19.4
$ zcat ../busybox-1.19.4-add-ucsimply_sam9260-support.patch.gz | patch -p1
```

- e) Provedeme výchozí konfiguraci Busyboxu pro modul SAM9260:

```
$ make ucsimply_sam9260_defconfig
```

- f) Výchozí konfiguraci případně upravíme:

```
$ make menuconfig
```

- g) Zkompilujeme Busybox:

```
$ make
```

- h) Nainstalujeme binární soubor Busyboxu a jeho applety do struktury rootfs (musíme to provést jako superuživatel **root** nebo přes **sudo**). Adresář, kde se nalézá rootfs v rámci adresářové struktury našeho počítače, určíme proměnnou **CONFIG_PREFIX**:

```
$ sudo make CONFIG_PREFIX=/home/emlin/workspace-sam9260/projects/ucsimply/
dist/rootfs install
```

¹⁶<http://www.ucsimply.cz/elnx/sestaveni-linuxove-distribuce/zaklad-distribuce/>

¹⁷<http://busybox.net/downloads/busybox-1.19.4.tar.bz2>

¹⁸<http://www.ucsimply.cz/products/modsam9260/>

Poznámka: Váš adresář s rootfs může být umístěn jinde.

- i) Nastavíme setuid bit na nainstalovaném binárním souboru Busyboxu (jsme v adresáři našeho kořenového souborového systému):

```
$ sudo chmod +s bin/busybox
```

3. Zkompilujeme si moduly ovladačů, které nejsou součástí jádra (jádro musí být nakonfigurované):

- a) Vstoupíme do adresáře se zdrojovými kódy jádra:

```
$ cd linux-2.6.38.8/
```

- b) Zkompilujeme ovladače, které nejsou zakompilované do obrazu jádra:

```
$ make modules
```

- c) Nainstalujeme zkompilované ovladače do struktury našeho rootfs. Umístění rootfs nastavíme do proměnné `INSTALL_MOD_PATH`.

```
$ export INSTALL_MOD_PATH=/home/emlin/workspace-sam9260/projects/ucsimply/
  dist/rootfs
$ sudo -E make modules_install
```

Poznámka: Váš adresář s rootfs může být umístěn jinde.

4.5.2 Příprava obrazu ubifs/ubi

Máme nachystaný obsah rootfs. Ovšem v téhle formě jej není snadné nahrát do modulu. Už jen kvůli tomu důvodu, že paměť NANDFlash by musela obsahovat vhodný oddíl zformátovaný souborovým systémem ubi/ubifs. Zdaleka nejsnazší je vytvořit obraz oddílu ubifs s kořenovým souborovým systémem a ten pak transformovat do nízkoúrovňového obrazu *ubi vrstvy*¹⁹. Obraz ubi vrstvy pak už můžeme nahrát přímo do paměti NANDFlash jako jakoukoliv jinou část firmware. Níže uvedený postup (vykonávaný pod účtem `root`) nám ukáže jak na to.

Než začneme:

- Nainstalujeme si (jako `root`) nástroje pro práci se souborovým systémem ubi/ubifs (platí pro linuxovou distribuci Debian 6.0 a vyšší):

```
# apt-get update
# apt-get install mtd-utils
```

Postup:

1. Vytvoříme obraz kořenového souborového systému pro vrstvu ubifs:

```
# mkfs.ubifs -v -r rootfs/ -m 2048 -e 129024 -c 1024 -o rootfs.ubifs
```

Uvedené parametry jsou `-m 2048` (velikost stránky 2048B), `-e 129024` (velikost logického bloku pro mazání - LEB - je 129.024B), `-c 1024` (max. počet LEB - vlastně max. velikost oddílu) a `-o rootfs.ubifs` (obraz ubifs vrstvy bude uložen do souboru rootfs.ubifs).

Výstup příkazu:

```
mkfs.ubifs
root:      rootfs/
min_io_size: 2048
leb_size:  129024
max_leb_cnt: 1024
```

¹⁹Souborový systém ubi/ubifs je dvouvrstvý. Vrstva ubi je první a pracuje přímo s paměťovým médiem. Vrstva ubifs je nad ní a využívá výhod abstrakce od paměťového média. Více zde: <http://www.linux-mtd.infradead.org/doc/ubifs.html>

```

output:      rootfs.ubifs
jrn_size:    8388608
reserved:    0
compr:       lzo
keyhash:     r5
fanout:      8
orph_lebs:   1
super_lebs:  1
master_lebs: 2
log_lebs:    5
lpt_lebs:    2
orph_lebs:   1
main_lebs:   30
gc_lebs:     1
index_lebs:  1
leb_cnt:     41
UUID:        EA9C1AA9-EEE4-446E-8E7A-DD97677FF379
Success!

```

*Poznámka: Pokud nástroj `mkfs.ubifs` spadne nebo zahlásí nějakou chybu, je třeba si ověřit, že velikost zdrojového adresáře není větší než součin velikost `LEB` * počet `LEB`.*

2. Vytvoříme konfigurační soubor `ubi.cfg`, který nástroji `ubinize` řekne, kolik oddílů a jakých vlastností bude v obrazu `ubi` vrstvy.

Příklad souboru `ubi.cfg`:

```

[rootfs]
mode=ubi
image=rootfs.ubifs
vol_id=0
vol_size=128MiB
vol_type=dynamic
vol_name=rootfs

[unused]
mode=ubi
vol_id=1
vol_size=96MiB
vol_type=dynamic
vol_name=unused
vol_flags=autoresize

```

Obsah souboru `ubi.cfg` popisuje dva oddíly `rootfs` a `unused` s ID 0 a 1. Oddíl `rootfs` bude obsahovat, to co je v obrazu `rootfs.ubifs`, zatímco oddíl `unused` bude prázdný. Velikost oddílu `rootfs` je 128MB, zatímco oddíl `unused` bude mít min. 96MB volného místa. Ovšem díky příznaku `autoresize` bude oddíl `unused` ovladačem `ubi/ubifs` zvětšen na maximální velikost až do vyčerpání volné kapacity média (paměti NandFlash). Zvětšení proběhne během prvního bootování po nahrání obrazu `ubi` vrstvy do NandFlash paměti.

3. Pomocí nástroje `ubinize` vytvoříme obraz `ubi` vrstvy:

```
# ubinize -v -o ubi-nandflash.bin -m 2048 -p 128KiB -s 512 ubi.cfg
```

Parametr `-m 2048` udává velikost stránky v Bytech, `-p 128kB` pak velikost PEB (fyzický blok pro mazání) v kB a `-s 512` udává velikost podstránky (subpage) v Bytech. Výstup - obraz `ubi` vrstvy - bude uložen do souboru `ubi-nandflash.bin`.

Výstup příkazu:

```

ubinize: LEB size:      129024
ubinize: PEB size:     131072

```

```
ubinize: min. I/O size: 2048
ubinize: sub-page size: 512
ubinize: VID offset: 512
ubinize: data offset: 2048
ubinize: loaded the ini-file "ubi.cfg"
ubinize: count of sections: 2
ubinize: parsing section "rootfs"
ubinize: mode=ubi, keep parsing
ubinize: volume type: dynamic
ubinize: volume ID: 0
ubinize: volume size: 134217728 bytes
ubinize: volume name: rootfs
ubinize: volume alignment: 1
ubinize: adding volume 0
ubinize: writing volume 0
ubinize: image file: rootfs.ubifs
ubinize: parsing section "unused"
ubinize: mode=ubi, keep parsing
ubinize: volume type: dynamic
ubinize: volume ID: 1
ubinize: volume size: 100663296 bytes
ubinize: volume name: unused
ubinize: volume alignment: 1
ubinize: autoresize flags found
ubinize: adding volume 1
ubinize: writing layout volume
ubinize: done
```

Obraz ubi vrstvy, soubor *ubi-nandflash.bin*, můžeme nahrát pomocí nástroje SAM-BA do paměti NandFlash na určenou adresu. Po nabofování uvidíme v adresáři */dev* dvě zařízení *ubi0_0* a *ubi0_1*, které představují naše dva oddíly - *rootfs* s *unused*.

5 Příklady

Kdo si hraje, nezlobí. Alespoň tak se to říkává. My si budeme hrát s linuxovým modulem uCSimply SAM9260. Zprovozníme si další periférie modulu a pomocí nich připojíme k modulu několik zajímavých zařízení - USB flash disk (úložného prostoru není nikdy dost), USB-WiFi modul (kdo dnes není bezdrátový, jako by nebyl), GSM/GPRS modul (vlastní modem? proč ne?), dotykový LCD displej a další.

U každého příkladu vám ukážeme, jak zapojit potřebnou periférie modulu s využitím základní desky Baseboard, jak ji zprovoznit v systému Linux na modulu SAM9260 a co udělat proto, aby připojené externí zařízení fungovalo. Přejali bychom si, abyste si hráli a aby vás to bavilo.

5.1 USB host

Zprovoznění rozhraní USB host na modulu SAM9260 nám otevře cestu k řadě zařízení, s kterými se dá komunikovat přes USB rozhraní. Ať už jde starou dobrou USB klávesnicí nebo USB monitor. V podstatě jsme limitováni pouze podporou daného zařízení v Linuxu, resp. existencí ovladače pro dané zařízení pro systém Linux. Základní příklad, který si vyzkoušíme, je připojení USB flash disku, tedy USB mass storage zařízení.

5.1.1 Zapojení

Zapojit rozhraní USB host modulu SAM9260 je jednoduché. Stačí připojit USB konektor typu A a vratnou pojistku 500mA typu polyswitch. Vratná pojistka má za úkol zabránit proudovému přetížení zdroje napětí +5V na základní desce Baseboard v případě, že se zařízení připojené k modulu přes USB nechová dle specifikace (max. povolený odběr s USB host je právě 500mA).

Referenční zapojení rozhraní USB host modulu SAM9260 je v případě použití základní desky Baseboard vidět na obrázku 5.1.1.

Modul zasuneme do pájecího pole základní desky a nachystáme si několik drátových propojek různých barev.

Jako vždy nejprve připojíme k modulu napájecí napětí +5V a zem GND - stejnojmenné pinheady na boku základní desky propojíme s piny 1 a 3 pinheadu J3 na modulu SAM9260.

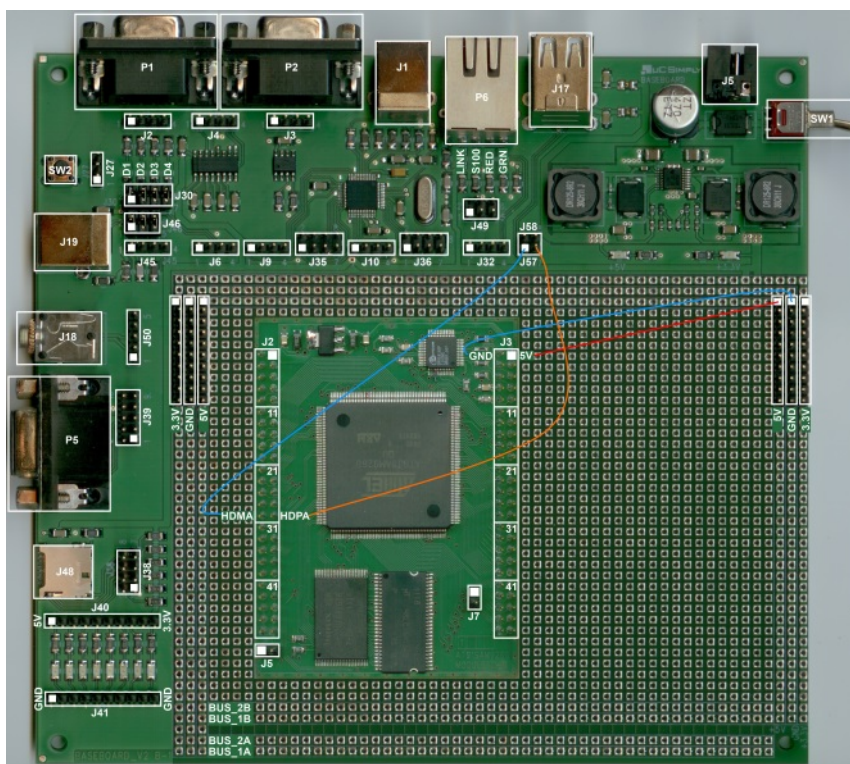
Pojistka polyswitch je již přítomná na základní desce. Konektor USB také a to dokonce dvojitý - konektor J17. Signály dolního konektoru jsou vyvedeny na pinhead J57 na základní desce, zatímco signály horního konektoru jsou přivedeny na pinhead J58. Můžete si tedy vybrat.

Signály HDMA a HDPA rozhraní USB host z modulu přivedeme na dolní konektor dvojkonektoru J17 typu 2x USB A. To znamená, že propojíme piny 30 a 29 pinheadu J2 na modulu s piny 1 a 2 pinheadu J57 na základní desce. Toť vše. Pokud byste trvali na použití horního konektoru dvojkonektoru J17, tak byste museli signály HDMA a HDPA z modulu SAM9260 přivést na piny 1 a 2 pinheadu J58.

5.1.2 USB mass storage

Rozhraní USB host máme zapojeno, takže nám nic nebrání jej otestovat. Nejjednodušším testem bude připojit přes něj k modulu nějaké nekomplikované USB zařízení. Takovým zařízením, který má dnes každý, je USB flash disk. Navíc je to zařízení, které je podporováno obecným ovladačem USB mass storage. Takže není nutné se nějak trápit s ovladačem - ten je už zakompilován do jádra.

Zapneme napájecí napětí, počkáme, až nabojuje modul do přihlašovacího promptu a přihlásíme se do Linuxu. Pak zasuneme USB flash disk (nebo třeba chytrý telefon s flash pamětí



Obrázek 5.1.1: Modul SAM9260 - zapojení USB host rozhraní

dostupnou jako USB disk) do spodního konektoru konektoru J17 na základní desce. Na systémové konzoli by se mělo objevit hlášení:

```
scsi0 : usb-storage 1-1:1.0
scsi 0:0:0:0: Direct-Access      A-DATA   USB Flash Drive  0.00 PQ: 0 ANSI: 2
sd 0:0:0:0: [sda] 7892087 512-byte logical blocks: (4.04 GB/3.76 GiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] Assuming drive cache: write through
          sda: sda1
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] Attached SCSI removable disk
```

Toto hlášení říká, že bylo připojeno USB storage zařízení s názvem *A-DATA USB Flash Drive* o velikosti 4.04GB z toho 3.76GB je volných. Toto zařízení bylo do systému namapováno jako zařízení `/dev/sda`, přičemž na zařízení je jeden jediný oddíl přístupný jako `/dev/sda1`. Skvělá zpráva - rozhraní USB host funguje!

Připojený USB flash disk sice funguje, ale ještě na něj nemůžeme přistupovat, protože souborový systém na USB disku není připojený do kořenového adresáře Linuxu. To změním snadno - připojíme si oddíl USB disku `/dev/sda1` do adresáře `/media/usb`:

```
# mount /dev/sda1 /media/usb
```

Pokud systém nenahlásil žádnou chybu, tak je vše v pořádku a obsah USB disku je dostupný v adresáři `/media/usb`. Pozornější čtenáři se možná zarazí a namítnou, že jsme příkazu `mount` nepředali typ souborového systému na USB flash disku. U běžných souborových systémů to nevádí - příkaz `mount` se souborový systém připojovaného zařízení pokusí určit sám.

Poznámka: Adresář /media/usb musí existovat. V připravené linuxové distribuci, nahrené na modulu SAM9260 z výroby, tomu tak je.

Když budeme chtít USB flash disk vytáhnout, je třeba jeho souborový systém odpojit. Jinak hrozí poškození dat na USB disku. To zařídíme příkazem:

```
# umount /dev/sda1
```

Další hrátky s rozhraním USB host budou následovat.

5.2 MMC/SD rozhraní

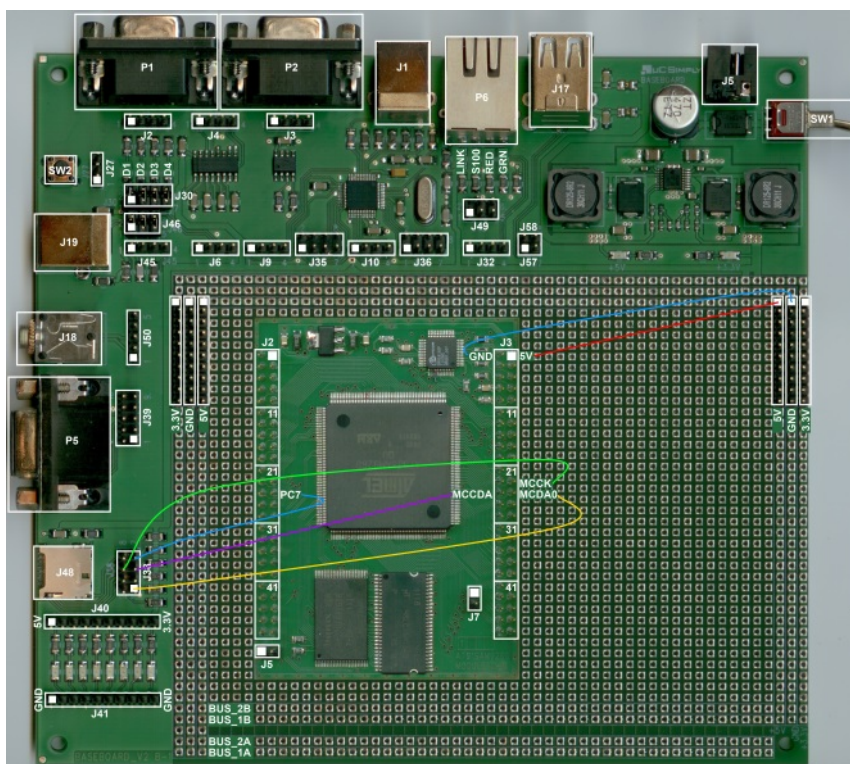
Jak asi z názvu nadpisu tušíte, nyní si ukážeme jak zprovoznit rozhraní MMC/SD pro připojení paměťových karet. Paměťové karty MMC, resp. SD, lze využívat jako další vysokokapacitní médium. Dokonce je možné paměťové karty využít i jako média pro uložení kořenového souborového systému.

5.2.1 Zapojení

Komunikace s kartami typu SD probíhá přes rozhraní SD card nebo přes rozhraní SPI. My jsme zvolili rozhraní SD, protože je pro paměťové karty nativní. Pro připojení Sd karty je na základní desce dostupný konektor J48 typu microSD.

Rozhraní SD můžeme zapojit dvěma způsoby - buď využijeme celou šíři datové sběrnice, tj. 4 bity, nebo využijeme jen jeden datový vodič. My jsme po řadě experimentů s několika SD kartami zvolili z důvodů stability variantu s 1bitovou datovou sběrnicí. Rozhraní SD je totiž taktováno až na 50MHz, což vyžaduje přizpůsobenou sběrnicí s definovanou impedancí. A to pomocí drátových propojek dosáhnout nelze (tedy ne snadno). Přenosová rychlost bude sice menší, ale stabilita je důležitější.

Potřebné pull-up odpory a blokuující kondenzátory dle definice SD rozhraní jsou už součástí zapojení microSD konektoru na základní desce Baseboard. Zapojení rozhraní SD card modulu SAM9260 je tak vskutku triviální.



Jako vždy nejprve připojíme k modulu napájecí napětí $+5V$ a zem GND - stejnojmenné pinheady na boku základní desky propojíme s piny 1 a 3 pinheadu J3 na modulu SAM9260.

Pak přivedeme signály $MCCK$, $MCCDA$ a $MCCDA0$ rozhraní SD card modulu SAM9260 na konektor J38 na základní desce - propojíme piny 23, 26 a 25 pinheadu J3 na modulu SAM9260 s piny 6, 5 a 1 pinheadu J38 na základní desce.

Potom přivedeme signál CD (Card Detect) z konektoru J38, který slouží jako indikace za/vysunutí paměťové karty, na pin $PC7$ mikrokontroléru AT91SAM9260. Tím je rozhraní SD card zapojeno.

Poznámka: To, že právě pin PC7 slouží jako vstup pro signál CD, je dáno konfigurací modulu SAM9260, která se pomocí patche přidává do zdrojového kódu linuxového jádra.

5.2.2 SD karta

Zkouška funkčnosti rozhraní SD je podobně jako u rozhraní USB host velmi jednoduchá - prostě necháme nabootovat modul, přihlásíme se do Linuxu a vložíme paměťovou kartu. Libovolná SD karta by měla fungovat. Po jejím vložení by se mělo objevit hlášení:

```
mmc0: host does not support reading read-only switch. assuming write-enable.
mmc0: new SD card at address 0002
mmcblk0: mmc0:0002 00000 1.86 GiB
mmcblk0: p1
```

Systém nám tímto hlášením oznámil, že na rozhraní $mmc0$ nepodporuje detekci zákazu zápisu na SD kartu a že tudíž předpokládá, že na kartu je povoleno zapisovat. Ano, má pravdu, signál WP (Write Protect) jsme k modulu nepřipojili. Dále nám systém oznámil, že našel novou SD kartu a přidělil jí v rámci sběrnice adresu 002. Následně se dozvídáme, že systém zaregistroval v adresáři $/dev$ nové blokové zařízení $mmcblk0$, což je zařízení na rozhraní $mmc0$ s adresou 002. Zařízení je paměťové médium o velikosti 1.86GB a obsahuje jeden oddíl $p1$. Výpisem adresáře $/dev$ snadno zjistíme, že tento oddíl je dostupný jako zařízení $/dev/mmcblk0p1$. Výborně, rozhraní MMC/SD funguje.

Pokud vám systém místo uvedeného hlášení vypsal I/O chyby či chyby zápisu, zkuste vyměnit drátové propojky, ověřte správnost zapojení nebo použijte jinou SD kartu. I/O chyby totiž značí nestabilitu na sběrnici, přes kterou je SD karta připojena. S ohledem na kvalitu sběrnice zhotovené pomocí drátových propojek se může stát, že modul není schopen s kartou komunikovat spolehlivě.

Dalším průběžným kamenem, který ukáže, jestli jsme vše správně zapojili a jestli rozhraní MMC/SD funguje spolehlivě, je připojení souborového systému na kartě (nejčastěji FAT). Ručně můžeme SD kartu, resp souborový systém na ní přítomný, připojit /odpojit pomocí příkazů:

```
# mount /dev/mmcblkp1 /media/mmc
# umount /dev/mmcblk0p1
```

Jako bonus jsme si pro vás připravili skript, který připojení a odpojení souborového systému karty řeší automaticky při za/vysunutí karty. Tento skript ($/sbin/automount-mmc$) je součástí připraveného firmwaru. Jeho volání zajišťuje správce souborů zařízení $mdev$, jak mu ostatně určuje jeho konfigurační soubor $/etc/mdev.conf$. Pokud jste tedy při zasunutí karty nezaznamenali žádné chybové hlášení, tak je obsah vaší karty automaticky připojen a je dostupný v adresáři $/media/mmc$.

Poznámka: Zkušenosti ukazují, že dráha, kterou vykoná microSD karta při vysouvání je příliš krátká na to, aby byl pomocí skriptu $/sbin/automount-mmc$ bezpečně odpojen souborový systém SD karty ještě před úplným vysunutím karty. Proto je bezpečnější odpojovat souborový systém SD karty ručně a teprve poté vysunout kartu z konektoru.

Poznámka: Skript $/sbin/automount-mmc$ připojuje jakékoliv blokové zařízení typu MMC/SD karta stále do jednoho adresáře bez ohledu na počet oddílů na paměťové kartě. Běžně paměťová karta více oddílů nemá, ale pokud ta vaše ano, tak musíte uvedený skript upravit, aby např. pro každý oddíl zvlášť vytvářel a pak zas opětovně rušil extra adresář.

5.3 USB-WiFi modul

USB-WiFi modul je druhým USB zařízením, které si k modulu připojíme přes rozhraní USB host. Modulu SAM9260 tak rázem přibude WiFi rozhraní, takže nebude problém se s ním připojit do počítačové sítě bezdrátově. A to všechno z příkazové řádky bez pomoci nějaké okenní aplikace!

S WiFi to bývalo v Linuxu někdy složité. Naštěstí dnes je už množina WiFi zařízení, které v Linuxu pracují spolehlivě, dostatečně široká. My jsme si jako USB-WiFi zařízení vybrali model TL-WN721N (obrázek 5.3.1) od firmy TP-LINK. Nabízí dobrou podporu na straně Linuxu, funguje stabilně a stojí jen pár stokorun. V následujících odstavcích si ukážeme, jak USB-WiFi modul rozběhat na modulu SAM9260 pod Linuxem, jak si zkompileovat nástroje pro konfiguraci WiFi připojení z příkazové řádky a jak se připojit k určenému access pointu.



Obrázek 5.3.1: USB-WiFi modul TL-WN721N

5.3.1 Než začneme

K tomu, abychom úspěšně zprovoznili USB-WiFi modul, bude potřeba do kořenového souborového systému přidat celou řadu souborů. Proto je nutné, abyste si do svého počítače s Linuxem stáhli *archiv*¹ s výchozím obsahem kořenového souborového systému (dále také jako rootfs) a rozbalili jej². Až do kořenového souborového systému přidáme všechny potřebné soubory, tak jej překonvertujeme do podoby binárního souboru, který pak nahrajeme do paměti NandFlash.

5.3.2 Ovladač, firmware a wireless-tools

K tomu, aby USB-WiFi modul mohl fungovat, potřebujeme dvě věci: za a) ovladač zakompilovaný v jádře (méně běžné) nebo ovladač v podobě modulu jádra, který není součástí monolitu jádra, a který se zavede, až když je potřeba, a za b) firmware, který ovladač nahraje do samotného USB-WiFi modulu. Ovladač zajišťuje, že se modul tváří jako WiFi síťová karta a firmware pak samotnou WiFi funkcionalitu (řízení rádiové části USB-WiFi modulu).

5.3.2.1 Ovladač

Ve výchozí konfiguraci jádra pro modul SAM9260 je zahrnuta podpora pro většinu dostupných USB-WiFi modulů v podobě modulů jádra. Tyto moduly jádra se kompilují samostatně po kompilaci jádra. Kompilací určitého modulu jádra získáme ovladač pro konkrétní USB-WiFi modul.

Zkompilované moduly jádra (ovladače) jsou už součástí archivu s obsahem kořenového souborového systému, který jsme si stáhli v předešlém kroku. Ovladač pro USB-WiFi modul TL-WN721N je také součástí zmíněného archivu, takže kompilaci příslušného modulu jádra řešit nemusíme.

¹<http://www.ucsimply.cz/products/modsam9260/>

²Archiv s obsahem rootfs rozbalíme příkazem `tar xjf nazev_archivu_rootfs`

5.3.2.2 Firmware

Soubor *ar9271.fw* s firmware pro modul TL-WN721N je ke stažení *zde*³. Stáhneme jej a uložíme do adresáře */lib/firmware* v kořenovém souborovém systému, který chystáme kvůli USB-WiFi modulu.

5.3.2.3 Wireless-tools

Wireless-tools je sada nástrojů pro příkazovou řádku, které potřebujeme pro konfiguraci WiFi části USB-WiFi modulu. Dále slouží pro zjištění seznamu dostupných sítí atd. Wireless-tools jsou již postupně nahrazovány balíčkem *compact-wireless*⁴, nicméně stále koexistují spolu a pro naše pokusy budou bohatě dostačovat.

Wireless-tools se musí kompilovat a pak nainstalovat do rootfs. Zdrojové kódy wireless-tools stáhneme *odsud*⁵ a to v minimálně ve verzi 27. Níže uvedený návod byl otestován s verzí 29. Wireless-tools budeme kompilovat kompilátorem *GNU/Linux*⁶ z balíku Sourcery CodeBench Lite Edition. My jsme použili starou verzi, konkrétně 2010.09-50 (ke stažení *zde*⁷).

Postup:

1. Rozbalíme stažený archiv se zdrojovými kódy wireless-tools. Vznikne adresář *wireless_tools.29*:

```
$ tar xf wireless_tools.29.tar.gz
```

2. Zkompilujeme a nainstalujeme wireless-tools pro modul SAM9260:

- a) Vstoupíme do adresáře *wireless_tools.29*:

```
$ cd wireless_tools.29
```

- b) Provedeme nastavení křížového kompilátoru a kompilaci v jednom kroku:

```
$ make CC='arm-none-linux-gnueabi-gcc -march=armv5te -mtune=arm926ej-s -mabi=aapcs-linux' BUILD_STRPPING='y' LDFLAGS='-Wl,-rpath=/usr/lib'
```

Poznámka: Nastavili jsme, že kompilaci bude provádět křížový kompilátor arm-none-linux-gnueabi-gcc, dále jsme upřesnili cílovou instrukční sadu (armv5te), cílový procesor (arm926ej-s) a typ ABI rozhraní (aapcs-linux). Povolili jsme ořezání spustitelných souborů a nastavili cestu k dynamickým knihovnám pro příkazy z balíku wireless-tool.

- c) Nainstalujeme výsledek kompilace do adresáře */usr/local* v rootfs pro modul SAM9260. Cestu k adresáři */usr* v našem rootfs nastavíme do proměnné **PREFIX**. Provést to musíme jako superuživatel **root** nebo přes příkaz **sudo**.

```
# make PREFIX='/home/emlin/workspace-sam9260/projects/ucsimply/dist/rootfs /usr/' install
```

- d) Protože *wireless-tools* obsahují dynamické knihovny, musíme aktualizovat cache zavaďče dynamických knihoven. Opět to provedeme jako **root**:

```
# ldconfig -v -r /home/emlin/workspace-sam9260/projects/ucsimply/dist/rootfs
```

3. Vytvoříme si nový obraz rootfs - viz návod v kapitole 4.5.⁸.

4. Nový obraz rootfs, kde je přidána podpora USB-WiFi, nahrajeme do modulu SAM9260 do paměti NandFlash pomocí programu SAM-BA.

5. Rebootujeme modul a přihlásíme se do Linuxu. Tím jsme připraveni na další část.

³<http://git.kernel.org/?p=linux/kernel/git/firmware/linux-firmware.git;a=tree;h=4c79292308ead42fc786c8e189921196ccc298ff;hb>

⁴http://linuxwireless.org/en/users/Download#Compat-wireless_release_types

⁵http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/wireless_tools.29.tar.gz

⁶<http://www.mentor.com/embedded-software/sourcery-tools/sourcery-codebench/platforms/arm-gnulinux>

⁷<http://www.ucsimply.cz/products/modsam9260/>

⁸<http://www.ucsimply.cz/elinx/sam9260-linux-manual/vlastni-firmware/korenovy-souborovy-system/>

5.3.3 WiFi pod Linuxem

Nyní, když máme v kořenovém souborovém systému ovladač, firmware a nástroje wireless-tools, si můžeme ukázat, jak zprovoznit USB-WiFi modul, konkrétně modul TL-WN721N, v prostředí Linuxu. Dost zbytečných řečí. Jdeme na to!

Postup:

1. Připojíme modul TL-WN721N do rozhraní USB host (např. prostřednictvím konektoru J17 na základní desce). Systém by měl zareagovat hlášením (adresa se může lišit):

```
usb 1-1: new full speed USB device using at91_ohci and address 3
```

2. Že je modul TL-WN721N připojený, ověříme ještě výpisem připojených USB zařízení:

```
# lsusb
```

Výstup příkazu:

```
Bus 001 Device 001: ID 1d6b:0001
Bus 001 Device 003: ID 0cf3:9271
```

kde zařízení s ID 0cf3:9271 je náš USB-WiFi modul.

3. Ručně zavedeme do jádra potřebný ovladač pro modul TL-WN721N:

```
# modprobe ath9k_htc
```

Reakce systému na konzoli:

```
usb 1-1: ath9k_htc: Transferred FW: ar9271.fw, size: 51312
usb 1-1: ath9k_htc: HTC initialized with 33 credits
usb 1-1: ath9k_htc: USB layer initialized
usbcore: registered new interface driver ath9k_hif_usb
```

V logu jádra bychom měli vidět podobné hlášení, byť trochu obsáhlejší:

```
# dmesg | grep ath
```

Výstup příkazu:

```
usb 1-1: ath9k_htc: Transferred FW: ar9271.fw, size: 51312
usb 1-1: ath9k_htc: HTC initialized with 33 credits
ath: EEPROM regdomain: 0x809c
ath: EEPROM indicates we should expect a country code
ath: doing EEPROM country->regdmn map search
ath: country maps to regdmn code: 0x52
ath: Country alpha2 being used: CN
ath: Regpair used: 0x52
Registered led device: ath9k-phy1::radio
Registered led device: ath9k-phy1::assoc
Registered led device: ath9k-phy1::tx
Registered led device: ath9k-phy1::rx
usb 1-1: ath9k_htc: USB layer initialized
usbcore: registered new interface driver ath9k_hif_usb
```

4. Ověříme, že je v systému zaregistrováno nové bezdrátové síťové rozhraní *wlan0*

```
# ifconfig -a
```

Výstup příkazu:

```
eth0      Link encap:Ethernet  HWaddr 02:40:50:60:70:90
          inet addr:192.168.1.12  Bcast:0.0.0.0  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0frame:0
```

5 Příklady

```
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
Interrupt:21 Base address:0x4000
```

```
lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        UP LOOPBACK RUNNING MTU:16436 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

wlan0   Link encap:Ethernet HWaddr F8:D1:11:60:37:07
        BROADCAST MULTICAST MTU:1500 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

Je to v pořádku, rozhraní *wlan0* je registrováno, stále ještě ovšem není aktivní (není ve stavu UP) a nemá přidělenou IP adresu.

5. Spustíme a nakonfigurujeme libovolný bezdrátový access point (AP). Pro jednoduchost budeme používat WEP šifrování, které je sice dnes už prolomitelné, ale zase nás nenutí instalovat a konfigurovat další nástroje pro rozběhání WPA zabezpečení na straně modulu. WiFi část AP nastavíme takto:

- SSID: ucsimply-net
- Šifrování na WEP, 128bit, klíč: d8e48ab6f352ef02268126bf0c (například)

Poznámka: USB-WiFi modul TL-WN721N akceptuje jako WEP klíč pouze 10-ti nebo 26-ti místné hexadecimální číslo. Bitově je pak délka klíče 40, resp. 104 bitů. Náš access point (AP) ovšem podporuje klíče pouze s 64, resp. 128 bitovou délkou. Experimentálně jsme ověřili, že když je klíč na straně modulu TL-WN721N dlouhý 104 a na straně AP 128 bitů, tak je spojení navázáno spolehlivě a je stabilní.

Poznámka: Podporované délky WEP klíčů USB-WiFi modulu zjistíme pomocí příkazu:

```
# iwlist wlan0 list
```

Výstup příkazu:

```
wlan0   2 key sizes : 40, 104bits
        4 keys available :
            [1]: D8E4-8AB6-F352-EF02-2681-26BF-0C (104 bits)
            [2]: off
            [3]: off
            [4]: off
        Current Transmit Key: [1]
```

6. Aktivujeme bezdrátové síťové rozhraní *wlan0* (i když nemá nastavené žádné parametry), abychom skenováním bezdrátových sítí ověřili, že USB-WiFi modul "vidí" náš AP. Pak zase rozhraní *wlan0* deaktivujeme:

```
# ifconfig wlan0 up
# iwlist wlan0 scan
# ifconfig wlan0 down
```

7. Příklad výsledku skenování bezdrátových sítí v dosahu (ostatní sítě byly pro stručnost vynechány):

```

8. Cell 01 - Address: 00:11:09:0C:00:4D
   Channel:7
   Frequency:2.442 GHz (Channel 7)
   Quality=66/70  Signal level=-44 dBm
   Encryption key:on
   ESSID:"ucsimply-net"
   Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s
   Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s
               36 Mb/s; 48 Mb/s; 54 Mb/s

   Mode:Master
   Extra:tsf=00000001344d0179
   Extra: Last beacon: 970ms ago
   IE: Unknown: 000C756373696D706C792D6E6574
   IE: Unknown: 010482848B96
   IE: Unknown: 030107
   IE: Unknown: 2A0100
   IE: Unknown: 32080C1218243048606C

```

Výborně, je evidentní, že USB-WiFi modul na naše AP "vidí". Navázání spojení by tedy mělo být úspěšné.

9. Nastavíme WiFi parametry USB-WiFi modulu (používat budeme příkaz `iwconfig` z `wireless-tools`):

a) nastavíme mód na *managed* = WiFi klient:

```
# iwconfig wlan0 mode managed
```

b) číslo kanálu nastavíme na *auto* (ať se modul domluví s AP):

```
# iwconfig wlan0 channel auto
```

c) max. velikost paketu nastavíme na *auto*:

```
# iwconfig wlan0 frag auto
```

d) nastavíme identifikátor sítě, kam se chceme připojit na *ucsimply-net*:

```
# iwconfig wlan0 essid ucsimply-net
```

e) nastavíme WEP klíč:

```
# iwconfig wlan0 key d8e48ab6f352ef02268126bf0c
```

f) ověříme, že nastavení bylo přijato:

```
# iwconfig wlan0
```

Výstup příkazu:

```

wlan0 IEEE 802.11bgn ESSID:"ucsimply-net"
      Mode:Managed Access Point: Not-Associated Tx-Power=0 dBm
      Retry long limit:7 RTS thr:off Fragment thr:off
      Encryption key:D8E4-8AB6-F352-EF02-2681-26BF-0C
      Power Management:off

```

Poznámka: Podle výpisu Access Point: Not-Associated poznáme, že USB-WiFi modul dosud není připojen (asociován) k žádnému AP.

10. Připojíme se prostřednictvím USB-WiFi modulu k naše mu AP:

a) Pro jistotu deaktivujeme drátové síťové rozhraní *eth0*, aby nekolidovalo s rozhraním *wlan0*:

```
# ifdown eth0
```


Poznámka: Kolizí myslíme použití stejné IP adresy příp. stejné směrovací pravidlo (viz příkaz `route`).

- b) nastavíme IP adresu pro USB-WiFi modul (nastavením fungující spolupráce s DHCP serverem se nebudeme zdržovat):

```
# ifconfig wlan0 add 192.168.1.12
```

- c) aktivujeme rozhraní wlan0 - tím USB-WiFi modulu přikážeme, aby se připojit k AP:

```
# ifconfig wlan0 up
```

Poznámka: Modul TL-WN721N by měl úspěšné připojení k AP indikovat rozsvícením zelené kontrolky na vrchní bílé straně. Pokud tato kontrolka po chvíli zhasne, znamená to, že autentizace u AP neproběhla dobře. Chyba bývá v hodnotě klíče nebo v jeho délce.

- d) Úspěšnost připojení na AP si ověříme podle výpisů z jádra:

```
# dmesg | tail
```

Výstup příkazu:

```
wlan0: authenticate with 00:11:09:0c:00:4d (try 1)
wlan0: authenticated
wlan0: associate with 00:11:09:0c:00:4d (try 1)
wlan0: RX AssocResp from 00:11:09:0c:00:4d (capab=0x431 status=0 aid=1)
wlan0: associated
```

- e) Úspěšnost spojení si ověříme samozřejmě i pomocí příkazu `iwconfig`:

```
# iwconfig wlan0
```

Výstup příkazu:

```
wlan0 IEEE 802.11bgn ESSID:"ucsimply-net"
Mode:Managed Frequency:2.442 GHz Access Point: 00:11:09:0C:00:4D
Bit Rate=1 Mb/s Tx-Power=20 dBm
Retry long limit:7 RTS thr:off Fragment thr:off
Encryption key:D8E4-8AB6-F352-EF02-2681-26BF-0C
Power Management:off
Link Quality=65/70 Signal level=-45 dBm
Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
Tx excessive retries:0 Invalid misc:2 Missed beacon:0
```

Poznámka: Výpis Access Point: 00:11:09:0C:00:4D indikuje, že jsme úspěšně připojeni k AP s danou MAC adresou.

5.3.3.1 Automatická připojení k AP

Ruční zavádění ovladače a ruční aktivace bezdrátového síťového rozhraní `wlan0` není zrovna pohodlná, zvláště, když bychom chtěli, aby se modul SAM9260 připojil do bezdrátové sítě automaticky po nabořování. Náprava je jednoduchá - upravit konfiguraci síťových rozhraní v souboru `/etc/network/interfaces` a zajistit automatické zavedení ovladače během bootování Linuxu.

Postup:

1. Nejprve si zajistíme, že ovladač bude automaticky zaveden během bootování Linux - to nám zajistí uživatelský inicializační skript `/etc/init.d/rcS`, který prochází soubor `/etc/modules` a který se všechny v něm uvedené moduly pokusí zavést. V příkazovém řádku modulu zadáme:

```
# echo "ath9k_htc" >> /etc/modules
```

2. Pak do souboru `/etc/network/interfaces` doplníme níže uvedené řádky. Využít můžeme editor *vi* přímo v modulu SAM9260 nebo můžeme soubor upravit v našem počítači v rootfs pro modul, pak si z něj vytvořit obraz rootfs a nahrát tento obraz do modulu SAM9260.

Doplňek do souboru `/etc/network/interfaces` (IP adresu, bránu atd. si upravte dle potřeby):

```
# Wireless network interface
iface wlan0 inet static
    address 192.168.1.12
    netmask 255.255.255.0
    gateway 192.168.1.254

# disable wired interface to not collidate with wlan0
pre-up ifdown eth0 &>/dev/null

# wireless settings
pre-up iwconfig wlan0 mode managed
pre-up iwconfig wlan0 channel auto
pre-up iwconfig wlan0 frag auto
pre-up iwconfig wlan0 essid ucsimply-net
pre-up iwconfig wlan0 key d8e48ab6f352ef02268126bf0c

# enable wired interface wireless is disabled afterwards
post-down ifup eth0 $>/dev/null
```

Výše uvedené nastavení nám zajistí, že bude možné bezdrátové rozhraní wlan0 de/aktivovat, tj. připojit/odpojit se pomocí příkazu `ifup wlan0`, resp. `ifdown wlan0`. Takže už nebudeme muset zadávat žádnou konfiguraci WiFi části, IP adresy apod. Zároveň bude zajištěno, že se automaticky deaktivuje a zpětně aktivuje drátové rozhraní *eth0* (kvůli případné kolizi - stejná IP adresa či stejné směrovací pravidlo). Stále však modul nenabootuje a automaticky neaktivuje rozhraní *wlan0*, ale dle výchozího nastavení drátové rozhraní *eth0*.

Náprava je jednoduchá - v souboru `/etc/network/interfaces` stačí zrušit řádek `auto eth0` a před řádek `iface wlan0 inet static` vložit řádek `auto wlan0`. To můžeme provést přímo z modulu SAM9260 pomocí editoru *vi*. Další informace o konfiguraci síťových rozhraní naleznete v češtině *zde*⁹ nebo v angličtině *zde*¹⁰.

⁹<http://www.abclinuxu.cz/clanky/system/debian-etc-network-interfaces-konfigurace-sitovych-rozhrani>

¹⁰http://www.debian.org/doc/manuals/debian-reference/ch05.en.html#_the_modern_network_configuration_for_desktop