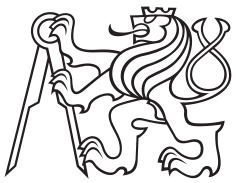
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra Řídící techniky

Integrace Ethernetového přepínače do embedded zařízení s OS Linux

Jan Kaisrlík

Květen 2015
Vedoucí práce: Ing. Michal Sojka Ph.D.

Poděkování / Prohlášení

Rád bych poděkoval vedoucímu práce Ing. Michalu Sojkovi, Ph.D. za vedení práce a odborné konzultace. Dále bych rád poděkoval svým rodičům, Ing. Iloně Kaisrlíkové a Janu Kaisrlíku, za neocenitelnou morální i materiální podporu v průběhu celého studia.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 11.5.2015

.....

Abstrakt / Abstract

Cílem této diplomové práce je portace operačního systému Linux na modul Colibri T20 a implementace ovladače do tohoto operačního systému pro teplotně odolný Ethernetový přepínač. Tento přepínač je postaven na obvodu 88E6065 vyvinutým firmou Marvell, ke kterému je připojen USB-Ethernet kontrolérem AX88772b od firmy Asix, který splňuje požadovaný průmyslový rozsah teplot. Tento kontrolér slouží k připojení přepínače za pomoci USB sběrnice.

Dané zadání bylo vyřešeno rozšířením stávajícího ovladače pro kontrolér AX88772b, úpravou přepínacího subsystému pro podporu zařízení připojených za běhu systému a vytvořením ovladače pro přepínací obvod 88E6065.

V práci byl vytvořen systém, který úspěšně zprovozní přepínač a vytvoří pro něj rozhraní viditelné v uživatelském prostoru operačního systému.

Klíčová slova: Linux, Ovladače, USB, DSA, Colibri T20, Marvell mv88e6065, Asix ax88772b

The aim of this diploma thesis is to integrate operating system Linux to embedded module Colibri T20 and implement drivers for thermally resilient Ethernet switch connected via USB. This Ethernet switch is based on switch circuit Marvell 88E6065. This switch circuit is connected to USB RMII interface provided by Asix AX88772b controller.

The practical part of this thesis includes design and implementation of the driver for operating system Linux. Mentioned driver extends current driver of controller Asix AX88772b. It was required to add possibility to Distributed Switch Architecture switch chip to connect switch without significant interrupting to the system. Subsequently, a driver for switch circuit 88E6065 was added.

Result of this thesis is successful connection of the switch to the module and creation of an interface which is visible from a user space.

Keywords: Linux, Drivers, USB, DSA, Colibri T20, Marvell mv88e6065, Asix ax88772b

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

Cílem mé diplomové práce je vytvoření ovladače do operačního systému Linux pro Ethernetový přepínač, který by splňoval průmyslové teplotní rozsahy.

Tyto přepínače se v dnešní době téměř nevyrábějí a moduly, které by splňovaly tyto teplotní rozsahy a měly integrované MII rozhraní, které složí ke komunikaci s přepínači a Ethernetovými fyzickými vrstvami, jsou finančně velmi nákladné. Proto se tato práce zabývá možností připojení Ethernetového přepínače pomocí USB sběrnice, což umožní dosáhnout požadovaných vlastností levnějším způsobem.

Vytvoření tohoto ovladače bylo zadáno firmou Retia a.s. se sídlem v Pardubicích. Tato firma se zabývá tvorbou elektronických vojenských systémů, záznamových zařízení a lokalizačních a bezpečnostních systémů. Účelem je nahrazení staršího modelu konfigurovatelného přepínače, který splňuje industriální teplotní rozsahy, za výkonnější. Firma poskytla pro tuto práci modul Colibri T20 [1]) a konfigurovatelný přepínač i s návrhem jeho zapojení. Přepínač je založen na přepínacím obvodu 88E6065, který je vyvinut firmou Marvell a který tvoří jádro samostatné desky plošných spojů. Deska konfigurovatelného přepínače je připojena Ethernetovým kontrolérem s USB rozhraním AX88772b, vyraběným firmou Asix, který podporuje připojení fyzické vrstvy za pomoci rozhraní RMII.

Pro vytvoření ovladače bylo potřeba provést portaci operačního systému Linux na modul Colibri T20, který má integrované hostitelské USB rozhraní a splňuje průmyslové teplotní rozsahy. Dálé bylo nutné implementovat ovladače do operačního systému Linux pro zařízení přepínacího obvodu s možností inicializace za běhu. Implementace byla provedena na současné verzi Linuxového jádra se snahou zařadit ovladač do hlavní větve Linuxového jádra.

Práce má následující strukturu. Ve druhé kapitole 2 je popsána použitá technologie, jako je sběrnice USB, MII rozhraní i rozšířené rozhraní RMII. Tato kapitola se také věnuje popisu relevantních Linuxových subsystémů – síťového, USB a subsystému přepínače.

V kapitole 3 Použitý Hardware je popsán zvolený procesorový modul Colibri T20 a použitá deska konfigurovatelného Ethernetového přepínače. Zvláštní důraz je kladen na popis čipů použitých na této desce – USB kontrolér Asix AX88772b a přepínací obvod Marvell 88E6065.

V následující kapitole 4 s názvem Implementace je popsán postup nahrání nového Linuxového jádra a navazující postup při oživování desky. Dále pak popisuje architekto-

[1]) `https://www.toradex.com/computer-on-modules/colibri-arm-family/nvidia-tegra-2`

nický návrh daného ovladače pro operační systém Linux a rozebírá implementaci tohoto návrhu.

V kapitole 5 s názvem Testování je popsáno, jak bylo testováno nastavení přepínacího obvodu. Dále se zde popisují proběhlé testy ovladače přepínacího obvodu. Na závěr kapitoly jsou výsledky těchto testů vyhodnoceny.

Kapitola 2
Použité technologie

V této kapitole je popsána použitá technologie. První část je věnovaná popisu sběrnice USB, která je popsána v 2.1. V další části je popsáno rozhraní Media Independent Interface (MII). Ke konci této kapitoly je pojednáváno o tom, jak vypadají ovladače v operačním systému Linux a jak vypadá rozhraní Linuxového subsystému Distributed Switch Architecture, které je použito pro daný ovladač a s ním související subsystémy.

2.1 Univerzální sériová sběrnice

Univerzální sériová sběrnice (USB) je technický standard vyvíjený od roku 1994 pro připojení periferií k počítači. Tato sběrnice měla nahradit pomalé sběrnice (jako jsou sériová linka, paralelní port, PS2 a mnohé další) jednou sběrnicí [1]. V dnešní době USB umožňuje přenášet velké množství dat s rychlostí až 640 MB/s u USB zařízení verze 3.0. Díky tomu se použití USB rozšířilo a nyní je podporováno skoro každým druhem zařízení, které umožňuje mimojiné přenašet video, audio či dokonce i Ethernetové rámce. Podporovaná zařízení jsou rozdělena do tříd z důvodu sjednocení funkcionality a chování zařízení. Nejznámějšími třídami jsou Human Interface Device, Printer a Mass Storage, které jsou více popsány ve vývojářské dokumentaci [2]. Toto sjednocené chování má za důsledek využívání společných ovladačů, které mohou uživatelé využívat pro práci s těmito zařízeními. Problém s některými druhy zařízení je takový, že je nejde do daných tříd zařadit, popřípadě mají rozšířené chování a potřebují speciální přístup.

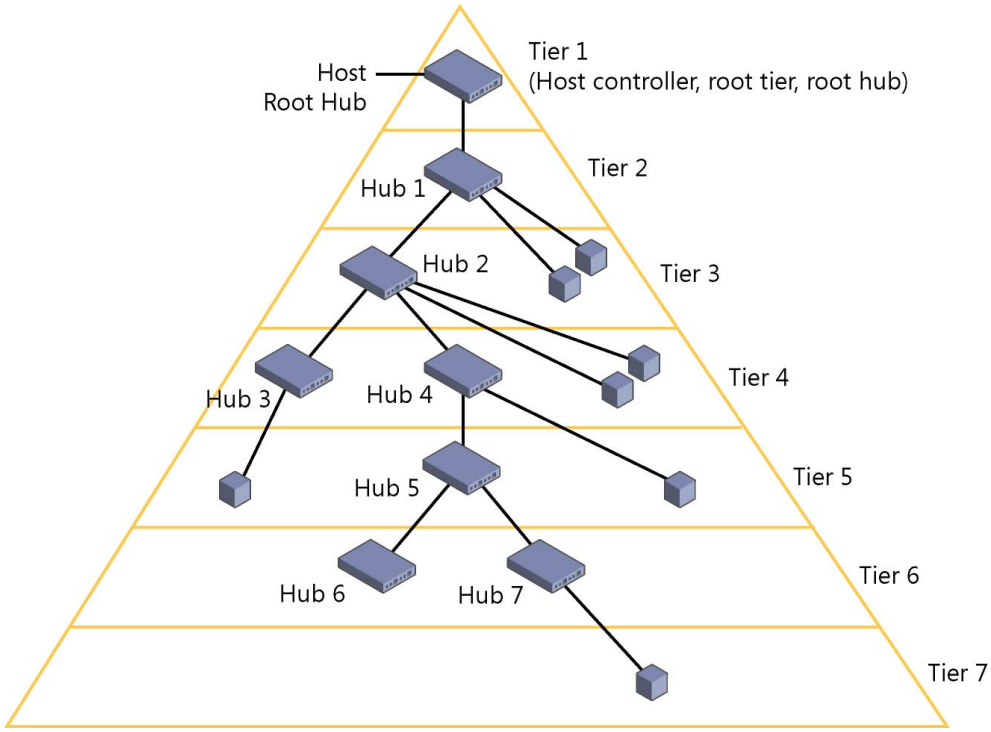
2.1.1 Rysy USB zařízení

USB sběrnice využívá metodu Master/Slave pro přístup k médiu a podporuje detekci připojeného zařízení za běhu (plug–and–play) a následnou automatickou konfiguraci (hotplug). USB síť se skládá ze tří druhů zařízení [2]:

- Hostitel – je zařízení typu Master a je jako jediný v systému, řídí komunikaci a obvykle integruje rozbočovač označený jako kořenový. Přiděluje zařízením unikátní adresy v síti.
- Rozbočovač – distribuuje datové toky a identifikuje připojení a odpojení dalších zařízení.
- Zařízení – pojmem zařízení je míněno koncové zařízení s požadovanou funkcionalitou, které odpovídá na řídící zprávy.

USB zařízení je asymetrická sběrnice se stromovou topilogií a s jedním zařízením typu Master. Strom je složený z linek bod-bod, přičemž USB rozbočovače vytvářejí

jednotlivé uzly tohoto stromu a zařízení tvoří jejich listy. Linky USB sběrnice jsou 4–vodičové skládající se z diferenciálního páru datových linek, napěťového a zemnícího vodiče, které jsou připojeny k zařízení a USB rozbočovačům. USB umožňuje připojit až 127 zařízení v rámci jedné USB sítě o hloubce maximálně pěti rozbočovačů tak, jak můžeme vidět na obrázku 2.1.



Obrázek 2.1. Příklad topologie sběrnice USB. Obrázek převzat z [3].

Díky tomu, že sběrnice USB je typu Master/Slave, umožňuje jednoduchý mechanismus detekce a enumerace, nebo-li průběhu konfigurace (více informací níže 2.1.2), za běhu systému. Průběh enumerace je řízen a konfigurován automaticky hostitelem.

2.1.2 Komunikace

USB komunikace je založena na logických kanálech, tzv. rourách. Každému výstupnímu kanálu by měl odpovídat právě jeden vstupní. Koncový bod obsahuje rouru s definovaným směrem.

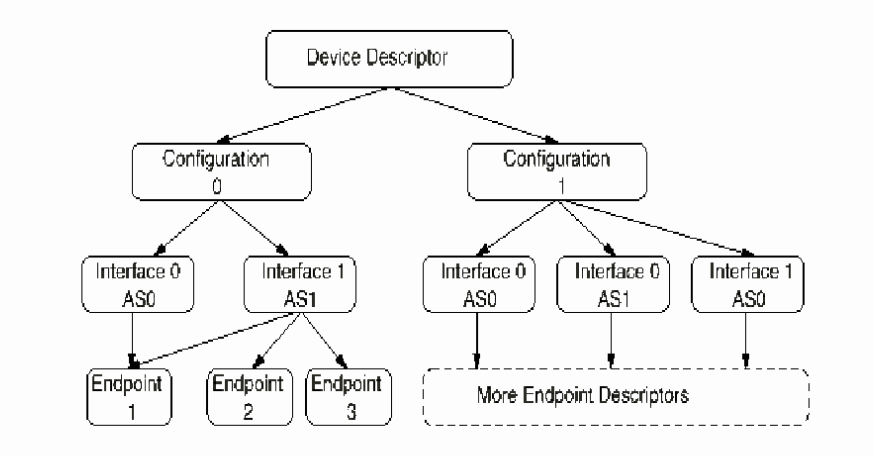
USB může mít maximalně 30 koncových bodů. Koncový bod je asociován s rourou, která má předem definovaný směr a typ přenosu:

- Řídící – obousměrná roura sloužící ke konfiguraci zařízení. Každé zařízení disponuje tímto druhem roury. Má rezervovanou určitou přenosovou kapacitu.
- Izochronní – jednosměrná roura sloužící ke stálému přenosu většího objemu dat. Má garantovanou latenci, přenos však není spolehlivý. Tento typ roury je vhodný pro audio a video.
- Přerušovací – jednosměrná roura sloužící pro časté přenosy malého množství dat. Má garantovanou šířku pásma. A přenos dat je spolehlivý. V případě chyby se přenos opakuje.

- Blokový – jednosměrná roura sloužící k přenosu velkého množství dat. Nemá rezervovanou žádnou přenosovou kapacitu ani dobu odezvy. Komunikace po této rouře je spolehlivá a v případě chyby se přenos opakuje.

Koncové body jsou inicializovány v průběhu enumarace USB zařízení, která probíhá po kontrolní rouře označené číslem 0, která je unikátní pro každé zařízení a jako jediná z rour je obousměrná.

Enumerace je posloupnost standardizovaných příkazů, kterou započal hostitel. V průběhu enumerace se předávají deskriptory, které obsahují důležité informace o zařízení. Hierarchii deskriptorů lze vidět na obrázku 2.2.



Obrázek 2.2. Hierarchie deskriptorů USB zařízení. Obrázek převzat z [4].

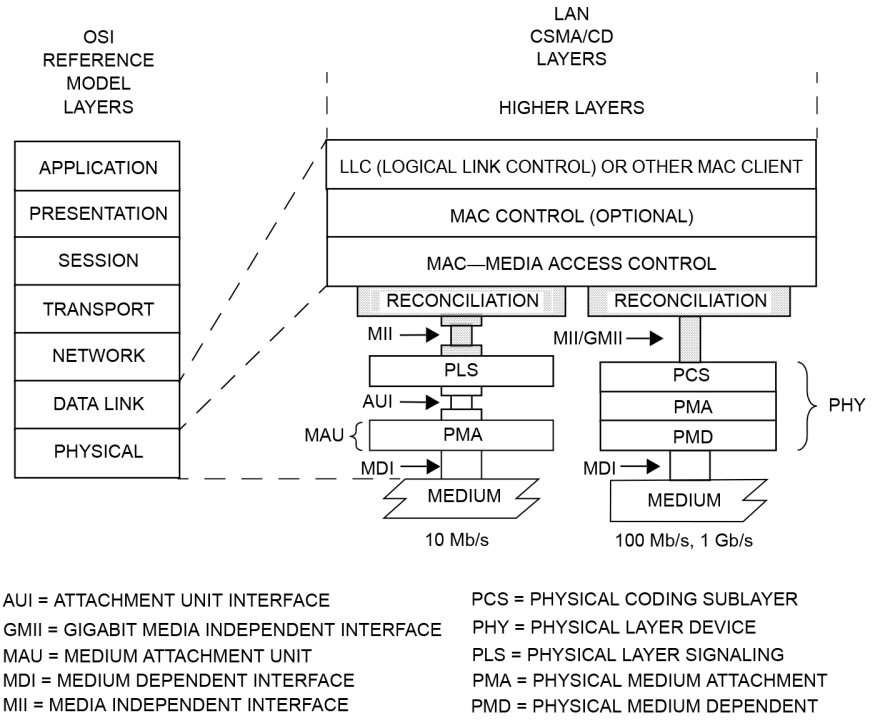
- Deskriptor zařízení – obsahuje informace jako je identifikátoru (ID) výrobce a produktu. Též obsahuje informace o třídě a maximální délce paketů, které může roura 0 přijmout.
- Konfigurační deskriptor – oznamuje, jak má být zařízení napájeno, jaký má zařízení maximální odběr proudu a počet rozhraní. Těchto dekriptorů může být více.
- Deskriptor rozhraní – tento deskriptor informuje o funkcionalitě celku, jako je třída koncových zařízení a počet koncových bodů.
- Deskriptor koncového bodu – používá se k přesnému popisu koncového bodu.
- Textový deskriptor – tento deskriptor přenáší pouze čitelné informace o zařízeních v kódování Unicode.

Na základě informací v deskriptorech se operační systém rozhoduje, který druh ovladače přiřadí danému druhu zařízení.

2.2 Media Independent Interface

Tato sekce předpokládá znalost modelu ISO/OSI.

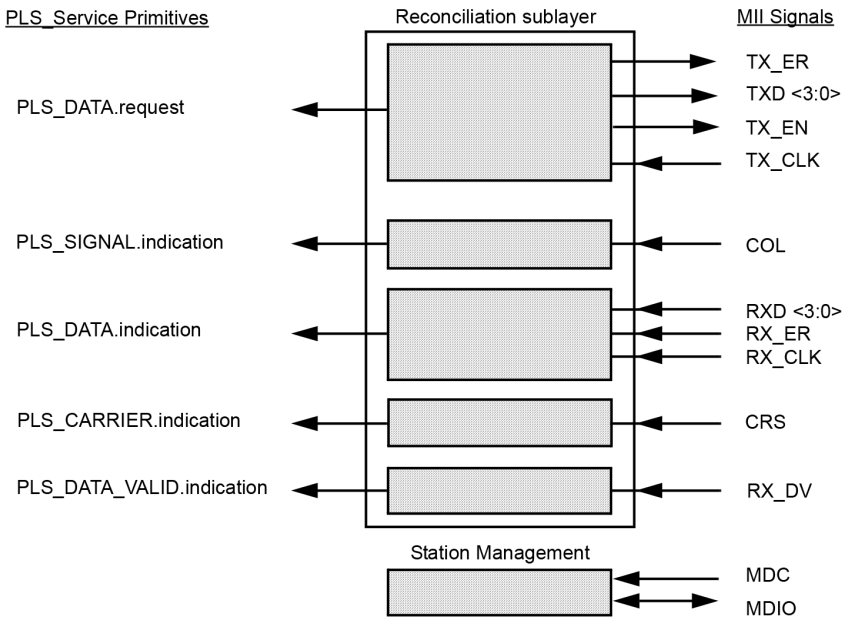
Media Independent Interface (MII) je typ rozhraní, které umožňuje připojení procesoru k fyzickému médiu a je na těchto prvcích nezávislé. Tento typ rozhraní je specifikován standardem IEEE 802.3, který je popsán v [[5], kapitola 22]. MII propojuje dvě vrstvy modelu ISO/OSI, a to vrstvu spojovou, konkrétně její část Media Access Constroll (MAC) [1]), s vrstvou fyzickou (PHY) [2]). MII rozhraní podporuje dva datové toky – 10 Mb/s a 100 Mb/s. Na obou datových tocích je funkcionalita MII identická, liší se pouze v nominální frekvenci hodin. Vyšší rychlosti přenosu je možno dosáhnout použitím rozhraní na Gigabite Media Independent Interface, které podporuje rychlost až 1 Gb/s.



Obrázek 2.3. Spojitost mezi rozhraním MII, modelem ISO/OSI a modelem IEEE 802.3 CSMA/CD LAN. Obrázek převzat z [5].

Na obrázku 2.3 je vidět připojení MII na Reconsiliation Sublayer (RS) a Physical Coding Sublayer (PCS) nebo Physical Layer Signaling (PLS). RS je podvrstva, která mapuje signály z MII na MAC/PLS obsluhu. PLS a PCS jsou vrstvy, které se starají o kódování a dekódování signálů. Konkrétní mapování MII signálů znázorňuje obrázek 2.4.

[1]) MAC je popsána standardem IEEE 802.2001. Stará se o přijímání a odesílání rámců (přístup k médiu).
[2]) PHY převádí bitový stream na signál a definuje jeho doručení.



Obrázek 2.4. Mapování signálů podvrstvy RS mezi rozhraním MII a vrstvou PLS. Také lze vidět připojení STA. Obrázek převzat z [5].

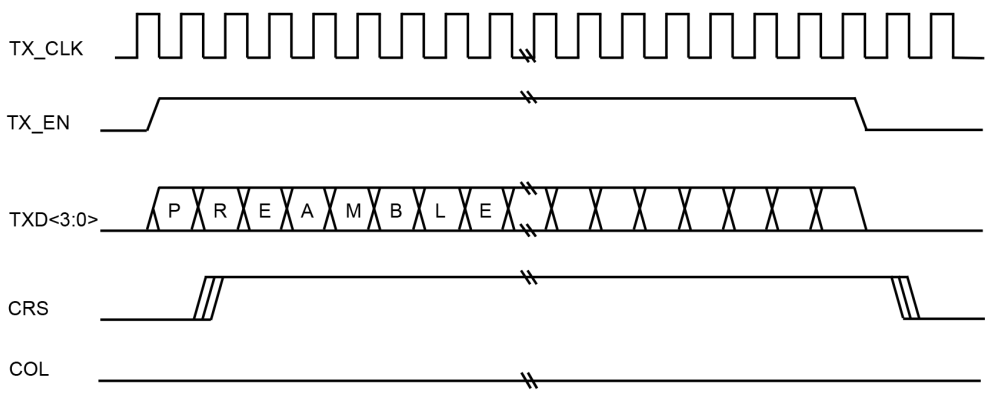
Jak můžeme vidět na obrázku 2.4, fyzické rozhraní MII se skládá ze dvou druhů sběrnic, a to datové 2.2.1 a ovládací (STA) 2.2.2. STA může být připojeno až k 32 fyzickým jednotkám, které obsluhuje.

2.2.1 Datová část MII rozhraní

Datová část MII disponuje několika druhy signálů, které lze vidět na pravé straně obrázku 2.4. Tyto signály jsou připojeny k přemapovací vrstvě RS. Vývody vysílající datové signály jsou:

- TX_CLK – vývod vysílající referenční hodinový signál pro synchronizaci signálů vysílaných po vývodech TX_EN, TXD a TX_ER. Zdrojem signálu TX_CLK je vrstva PHY. Frekvence tohoto signálu by měla být 20% z nominální hodnoty přenosu dat, $\pm 100 ppm$.
- RX_CLK – vývod vysílající referenční hodinový signál pro synchronizaci signálů vysílaných po vývodech RX_DV, RXD a RX_ER. Zdrojem signálu RX_CLK je podvrstva RS. RX_CLK může odvodit referenční hodnotu frekvence z přijímaných dat nebo z nominální hodnoty jako u signálu vývoduu TX_CLK.
- TX_EN – vývod vysílající signál, který indikuje, že je podvrstva RS připravena odesílat data.
- TDX – čtveřice datových vývodů vysílající signál (TDX[3:0]) ovládaná podvrstvou RS, která přenáší data synchronně na základě signálu z vývodu TX_CLK. TDX[0] přenáší nejméně významný bit. Pokud je TX_EN fyzicky odpojen, TDX nemá žádný efekt na činnost vrstvy PHY.

- TX_ER – vývod vysílající signál řízený podvrstvou RS, který indikuje. že vrstva PHY začala vysílat jeden či více symbolů, které nejsou součástí dat, nebo že předčasně došlo k přerušení rámce. Signál na tomto vývodu je synchronní s hodinovým signálem z vývodu TX_CLK.
- RX_DV – vývod vysílající signál, který značí, že přijatá data jsou validní. Tento signál je řízen vrstvou PHY a je synchronní se signálem z vývodu RX_CLK.
- RXD – čtveřice datových vývodů vysílající signál (RDX[3:0]) ovládaná vrstvou PHY. Tato čtveřice vývodů slouží k přenosu dat z vrstvy PHY do podvrstvy RS. RXD[0] přenáší nejméně významný bit.
- RX_ER – vývod vysílající signál řízený vrstvou PHY, který indikuje, že nastala chyba detekovaná v průběhu přijímání rámce. Pokud je vývod RX_DV odpojen, vývod RX_ER nemá žadný vliv na činnost podvrstvy RS.
- CRS – vývod vysílající signál indikující, zda je příjemce nebo odesílatel zaneprázdněn, nebo je ve stavu idle. Signál z vývodu není synchronizován se signálem z vývodů TX_CLK, RX_CLK. Pokud je bit z registru 0.8, popsaný v 2.8, aktivní, signál na vývodu má nedefinované chování.
- COL – vývod vysílající signál, který je vyvolán vrstvou PHY, indikující kolize, které nastaly na médiu. Signál z vývodu není synchronizován se signálem z vývodů TX_CLK, RX_CLK. Pokud je bit z registru 0.8, popsaný v 2.8, aktivní, signál na vývodu má nedefinované chování.



Obrázek 2.5. Bezchybná komunikace po MII rozhraní. Obrázek převzat z [5].

Na obrázku 2.5 lze vidět, jak vypadá komunikace po MII rozhraní. Pin TX_CLK přenáší referenční hodinový signál, který se pravidelně mění. Při odesílání dat se aktivuje signál z vývodu TX_EN. Následně se po vývodu začne odesílat preambule. Přenos těchto dat je popsán níže. S přijímaním signálu začne být příjemce zaneprázdněn, a proto aktivuje signál z vývou CRS. Celá komunikace probíhá pouze jedním směrem a neobsahuje žádnou chybu, jak je možno vidět ze signálu na vývodu COL.

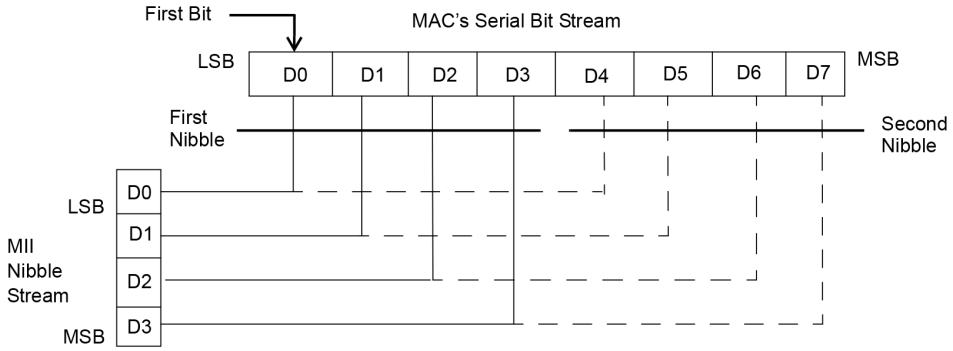
Rámec dat, který je přenášen po MII rozhraní, je doplněn o hlavičky a patičky popsáné formátem:

```
<vstupní rámec><preambule><sdf><data><edf>
```

kde jednotlivé položky mají následující význam:

- vstupní rámec – je doba mezi rámci, při které nesmí dojít k žádné činnosti na rozhraní MII. Délka této periody není přesně specifikována.
- preambule – je definovaná posloupnost dat, která se skládá ze 7 oktetů (bytů) a je přenášena ve formě pravidelného střídání jedniček a nul.
- sdf – oktet, který oznamuje začátek rámce.
- data – značí přenesená data.
- efd – konec rámce, který deaktivuje signál z vývodu TX_EN.

Rámce jsou přenášeny tak, že oktet je rozdělen na dvě poloviny tzv. nibbles. Tyto poloviny jsou odeslány po datových linkách TXD nebo RXD. Příklad odeslání oktetu po signálech TDX je znázorněn na obrázku 2.6 níže.



Obrázek 2.6. Rozdělení bytů, který je následně odeslán po vývodech TDX. Obrázek převzat z [5].

2.2.2 Ovládací rozhraní a příslušný registrový prostor

Ovládací část sběrnice, která se značí STA, se používá k nastavování a čtení hodnot mezi vrstvami MAC a PHY. Toto rozhraní je dvoulinkové sériové a obsahuje dva datové vodiče:

- MDC – vývod vysílající signál určující časovou základnu pro signál vysílaný vývodem MDIO. Signál z vývodu MDC je aperiodický, nemá žádnou minimální ani maximální frekvenci, měl by se však pohybovat v rozmezí od 160 do 400 ns.
- MDIO – vývod vysílající signál. Tento vývod je vstupně výstupní a jsou po něm posílány signály oběma směry. Je tvořen třístavovým obvodem mezi vrstvami PHY a STA. Pin se používá k přenosu kontrolních zpráv mezi zařízeními. Přenos je řízen z vrstvy PHY. Signál z vývodu MDIO je synchronizovaný se signálem z vývodu MDC.

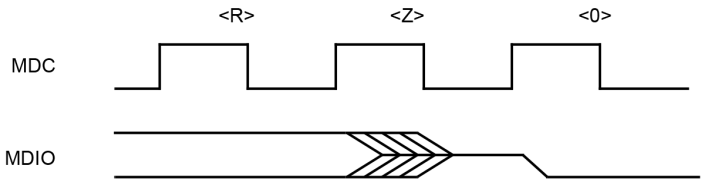
Formát ovládacích rámců je popsán v tabulce 2.1. Data z této tabulky jsou odesílána po vývodu MDIO ve směru z levé strany tabulky do pravé.

V tabulce 2.1 lze vidět hodnotu preambule (PRE), což je posloupnost 32 bitů skládajících se z hodnot logických jedniček. Preambule slouží k synchronizaci signálu z vývodu

	PRE	ST	OP	PHYAD	REGAD	TA	DATA	IDLE
čtení	1...1	01	10	AAAAA	RRRRR	Z0	D...D	Z
zápis	1...1	01	01	AAAAA	RRRRR	10	D...D	Z

Tabulka 2.1. Formát ovládacích zpráv. Tabulka převzata z [[5], Tabulka 22-12].

MDC. Začátek rámce je označený ST a má hodnotu 01. Poté jsou odesílána data popisující operační kód (OP), která určují druh operace: čtení nebo zápis. PHYAD je zkratka pro fyzickou adresu zařízení o délce 5 bitů. Tím dokážeme naadresovat až 32 zařízení. První bit adresy je nejvíce významný bit. Další položkou je adresa registru (REGAD) číslovaná podle níže uvedených registrů 2.8. Dalším blokem je TA. Při operaci čtení hodnota TA obsahuje stav vysoké impedance (Z) sloužící k prohození příjemce za odesílatele, jak lze vidět na obrázku 2.7. Prohození odesílatelů probíhá z důvodu požadavku odpovědi na operaci čtení. Odpovědí se myslí vyžádaná hodnota z registru. Při operaci zápis tato změna není potřeba a stav vysoké impedance je nahrazen logickou jedničkou. Odesílaná data mají 16 bitovou hodnotu, tedy stejnou jako je velikost registrů.



Obrázek 2.7. Čtení z registru při komunikaci po sériové ovládací lince STA. Obrázek převzat z [5].

Operacemi čtení a zápis může vrstva MAC přistupovat k registrovému prostoru na vrstvě PHY, který tak může měnit. Část tohoto registrového prostoru je definována standardem 802.3. Registrový prostor obsahuje dva základní skupiny registrů – kontrolní a stavový. Všechny vrstvy PHY, které podporují rozhraní MII, by měly používat právě tyto skupiny registrů 2.8.

Register address	Register name	Basic/Extended	
		MII	GMII
0	Control	B	B
1	Status	B	B
2,3	PHY Identifier	E	E
4	Auto-Negotiation Advertisement	E	E
5	Auto-Negotiation Link Partner Base Page Ability	E	E
6	Auto-Negotiation Expansion	E	E
7	Auto-Negotiation Next Page Transmit	E	E
8	Auto-Negotiation Link Partner Received Next Page	E	E
9	MASTER-SLAVE Control Register	E	E
10	MASTER-SLAVE Status Register	E	E
11	PSE Control register	E	E
12	PSE Status register	E	E
13	MMD Access Control Register	E	E
14	MMD Access Address Data Register	E	E
15	Extended Status	Reserved	B
16 through 31	Vendor Specific	E	E

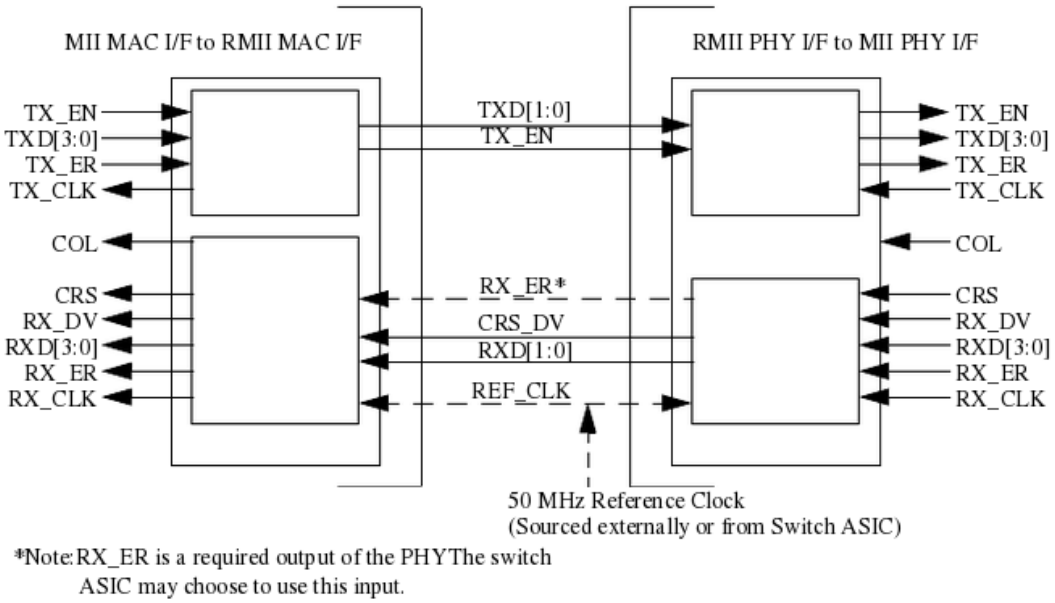
Obrázek 2.8. Registry STA rozhraní. Obrázek převzat z [5].

Tyto registry specifikují základní vlastnosti vrstvy PHY pro MII rozhraní o rychlostech 100 Mb/s a 1 Gb/s. Registry 2 – 14 jsou částí rozšířeného registrového prostoru [5] (22.2.4). Bližší popis stavového a kontrolního registru je uveden v přílohách B.3 a B.4.

2.2.3 Reduced Media Independent Interface

Reduced Media Independent Interface (RMII) je obdoba rozhraní MII, která má ale snížený počet vývodů z 16 na 8 [6]. Toto snížení je zajištěno přidáním přemapující podvrstvy Reconsiliation Sublayer (RS). Ta může nahradit stávající podvrstvu RS nebo může být připojena za ni [7].

Připojenou podvrstvu RS za stávající rozhraní lze vidět na obrázku 2.9.



Obrázek 2.9. Rozšíření rozhraní MII o další podvrstvu RS. Obrázek převzat z [6].

RMII používá stejné ovládací rozhraní jako MII definované standardem IEEE 802.3u [5], který je popsán v sekci 2.2.2.

Hlavními rozdíly RMII rozhraní oproti MII jsou datové signály TXD a RXD. které jsou v RMII rozhraní pouze 2 bity široké. Dále signály COL a CRS jsou zde sjednoceny do jednoho signálu na vývodu CRS_DV. Dalším rozdílem je detekce kolizí, která je na vrstvě MAC provedena jako logický součin signálů TX_EN a CRS_DV. Zdrojem hodinového signálu je vrstva MAC nebo je generován externími hodinami.

2.3 Subsystémy operačního systému Linux

Tato kapitola obsahuje informace o tom, co to je ovladač, a dále popisuje základní stavební prvky poskytováné operačním systému Linux: device, net_device, usb subsystém, usbnet, mii_bus, DSA a sysfs.

2.3.1 Ovladač

Ovladač je část programu (algoritmu), která spravuje nebo kontroluje určité zařízení připojené k počítači. Ovladače vytvářejí softwarové rozhraní pro hardwarové zařízení, které umožňují operačnímu systému ovládat tento hardware, aniž by uživatel věděl, jak přesně funguje [1].

Linuxové jádro je monolitický operační systém, a proto ovladače zařízení jsou nahrány ve stejném paměťovém prostoru jako samotné jádro. Operační systém Linux umožňuje zavádět tyto ovladače dynamicky za běhu systému nebo staticky při jeho kompilaci.

2.3.2 Struktura device

Každé zařízení je v operačním systému Linux reprezentováno instancí struktury *device*. Struktura device obsahuje informace o tom, co zařízení potřebuje, aby správně fungovalo v systému. Mnoho subsystémů uchovává informace o struktuře *device* a tyto informace využívá k dalším úkonům. Většina zařízení není reprezentována pouze strukturou *device*. Místo toho některé instance ve struktuře *device*, jako je kobj, obecně implementují zařízení, která jsou nadřazená struktuře *device* [8] [1]).

2.3.3 Zařízení typu net device

Struktura *net_device* popisuje síťové rozhraní jak fyzické tak virtuální (jako jsou VLAN)[9]. Síťové rozhraní může přijímat a vysílat velké množství dat, a proto je implementace podobná diskovým zařízením [1]. Síťová zařízení přijímají data asynchronně a ukládají je do vyrovnávací paměti.

V Linuxovém jádře je síťový subsystém implementován tak, že je na protokolu nezávislý.

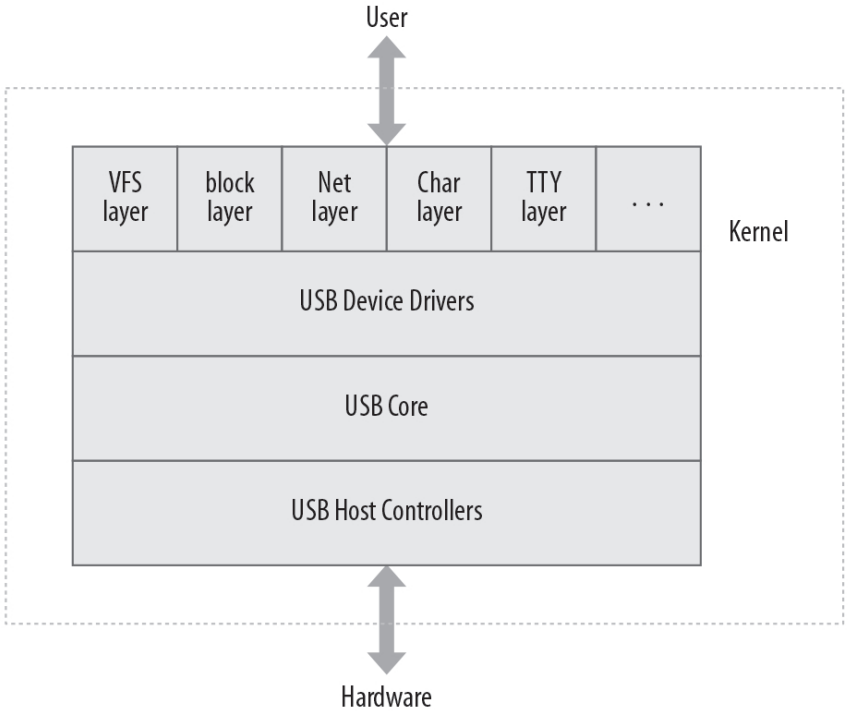
Jednotlivé položky struktury *net_device* mohou být klasifikovány do několika kategorií [9]:

- Konfigurační položky – část konfigurace je automaticky naistalována Linuxovým jádrem a je možné je měnit programy z uživatelského rozhraní, jako je ifconfig a ip.
- Statistiky – například statistiky jako počet odeslaných paketů a paketů přijatých.
- Stav zařízení
- Správa přenosu – nastavení kvality služeb (QoS).
- Tabulka virtuálních funkcí – je tabulka ukazatelů implementovaných ovladačů zařízení. Obsahuje funkce jako je aktivace a deaktivace portů, nastavení adres a mnohé další popsané v dokumentaci Linuxového jádra nebo v [9].

2.3.4 USB subsystém

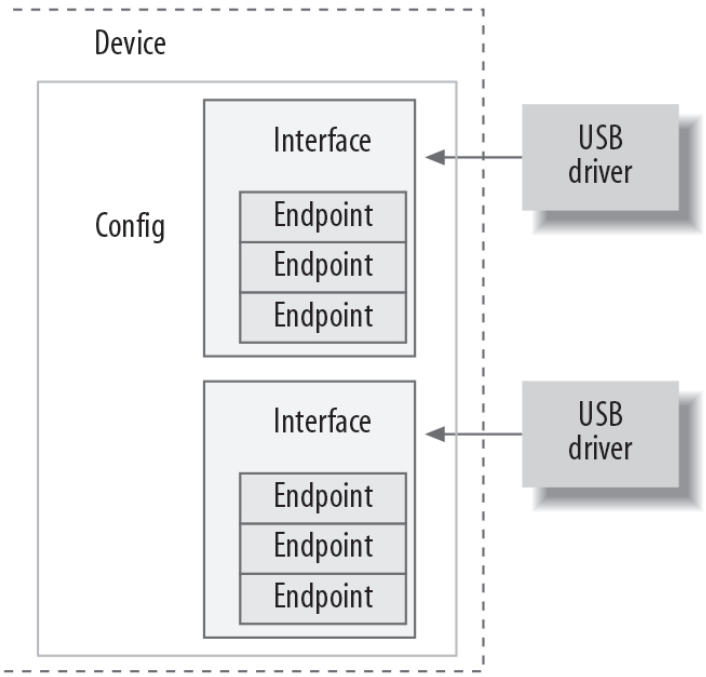
Linuxové jádro podporuje dva druhy USB ovladačů – ovladače na hostovském systému a ovladače na periferním zařízení [1]. USB ovladač na hostovském systému pracuje na systému, ke kterému je USB zařizení připojeno. Ovladač v periferních zařízeních, neboli „USB gadget drivers“ kontroluje připojená zařízení přes USB. Na obrázku níže můžeme vidět USB hierarchii datových struktur USB subsystému, kde USB zařízení může existovat v několika různých subsystémech (net, block, char ...). Jádro USB je část USB subsystému, která implementuje rozhraní pro USB ovladače, které chtějí ovládat hardware a přistupovat k němu tak, jak můžeme vidět na obrázky 2.10.

[1]) Popis datové struktury *device* lze nalézt na `http://lxr.free-electrons.com/source/include/linux/device.h?v=3.19#L730`



Obrázek 2.10. Hierarchie USB subsystému. Obrázek převzat z [1].

Koncové body USB jsou popsány v jádře Linuxu strukturou *usb_host_endpoint*. Tato struktura obsahuje informace o reálném koncovém bodu připojeného zařízení ve struktuře *usb_endpoint_descriptor*. Struktura *usb_endpoint_descriptor* popisuje data pomocí přijatých deskriptorů.



Obrázek 2.11. Hierarchie USB popisuje, jak jsou zaobaleny jednotlivé části USB subsystému. Obrázek převzat z [1].

USB rozhraní tak, jak můžeme vidět na obrázku 2.11, je složeno z několika koncových bodů, které tvoří jedno logické připojení, jako je například myš, klávesnice, video atd. Některá USB zařízení však mohou mít více rozhraní. Například USB repoduktory mohou mít dvě rozhraní, jako je USB klávesnice pro tlačítka reproduktoru a USB audio pro jeho zvuk [1]. Linuxové jádro používá pro každé rozhraní právě jeden ovladač. Každé rozhraní je popsáno strukturou *usb_interface*. Tato struktura obsahuje informace, které jádro USB předává USB ovladačům.

Každé zařízení může mít různé nastavení, které je předáváno v deskriptorech. Linuxové jádro popisuje toto nastavení ve struktuře *usb_host_config* a celé USB zařízení strukturou *usb_device*. Jedno USB zařízení však může mít konfigurací více. Konfigurace se mohou v průběhu práce zařízení měnit.

Ovladače USB zařízení obyčejně odkazují data ze struktury *usb_interaface* do struktury *usb_device*.

Subsystém sysfs (viz v sekci 2.3.8) umožňuje přístup ke strukturám *usb_device* a *usb_interface*, které jsou zobrazeny jako složky a soubory. Příklad cesty v sysfs reprezentující strukturu *usb_device* může vypadat následovně:

```
/sys/devices/pci0000:00/0000:00:09.0/usb2/2-1
```

K výše uvedené cestě k USB zařízení je na zvýrázněném řádku USB rozhraní reprezentováno strukturou *usb_interface*. Tato struktura je pojmenována podle formátu root_hub-hub_port:config.interface (pro hlubší stromy je schéma root_hub-hub_port-hub_port:config.interface):

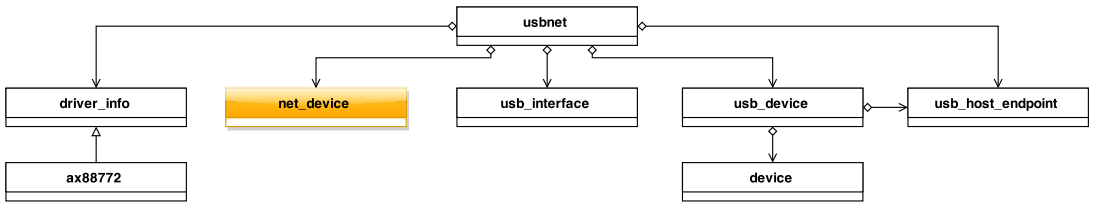
```
/sys/devices/pci0000:00/0000:00:09.0/usb2/2-1/2-1:1.0
```

Pro komunikaci se všemi USB zařízeními používá Linuxové jádro strukturu USB request block (URB). URB definuje asynchronní typ komunikace, která slouží k posílaní a příjmu dat pro specifický koncový bod USB a specifické USB zařízení.

2.3.5 Subsystém usbnet

Usbnet je subsystém Linuxového jádra, který umožňuje ovládat síťová zařízení jako je Ethernet, DSL, IDSN atd.

Ovladač pro zařízení Asix AX88772b patří do třídy usbnet. Na obrázku 2.12 níže lze vidět UML diagram důležitých struktur pro tento ovladač.



Obrázek 2.12. Diagram tříd vybraných datových struktur v subsystému usbnet.

Na obrázku 2.12 je znázorněn subsystém usbnet obsahující vztahy mezi jednotlivými strukturami, které mají následující význam:

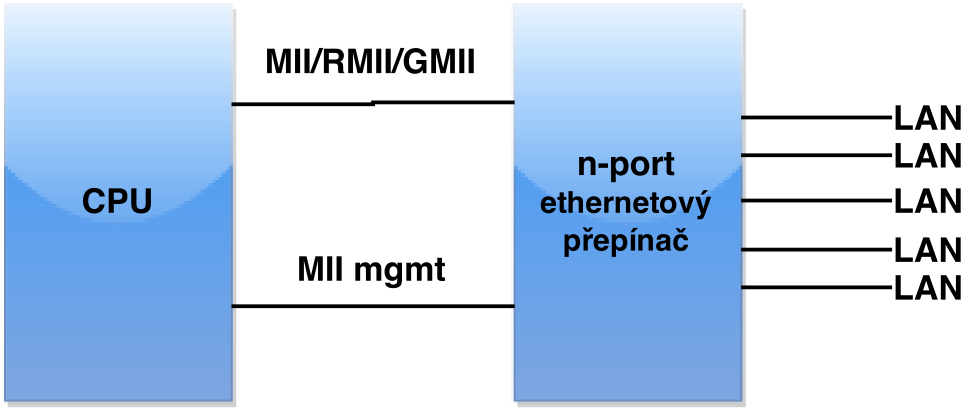
- net_device – struktura reprezentující síťové rozhraní
- driver_info – struktura pro ovladač usbnet zařízení
- usb_device – struktura reprezentující zařízení USB

2.3.6 Struktura mii bus

Struktura *mii_bus* reprezentuje MII rozhraní. Obsahuje tabulku virtuálních funkcí a umožňuje přístup k registrům popsaných v sekci 2.8.

2.3.7 Subsystém Distributed Switch Architecture

Distributed Switch Architecture (DSA) je subsystém pro správu hardwarových Ethernetových přepínačů [10]. Tento subsystém podporuje přepínače, které jsou připojeny způsobem zobrazeným na obrázku 2.13.

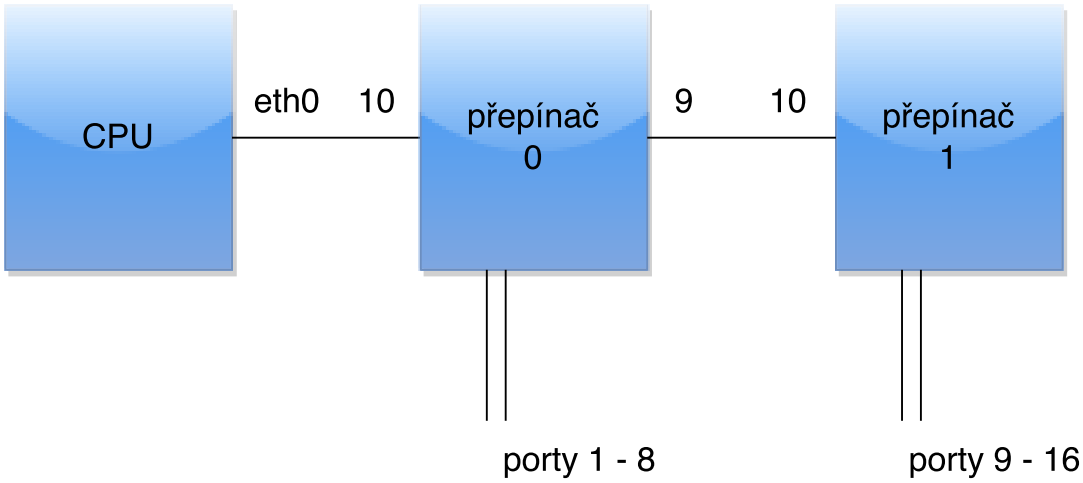


Obrázek 2.13. Možné připojení Ethenetového přepínače k procesoru, které využívá subsystém DSA. Obrázek je převzat z [10].

DSA subsystém interpretuje každý port přepínače jako oddělené síťové rozhraní popsané v sekci 2.3.3. Konfigurace přepínače, jako je nastavení portů a správa MII rozhraní, se děje na základě struktury ukazatelů na funkce *ethtool_ops*, kterou využívá například nástroj Ethtool [1]).

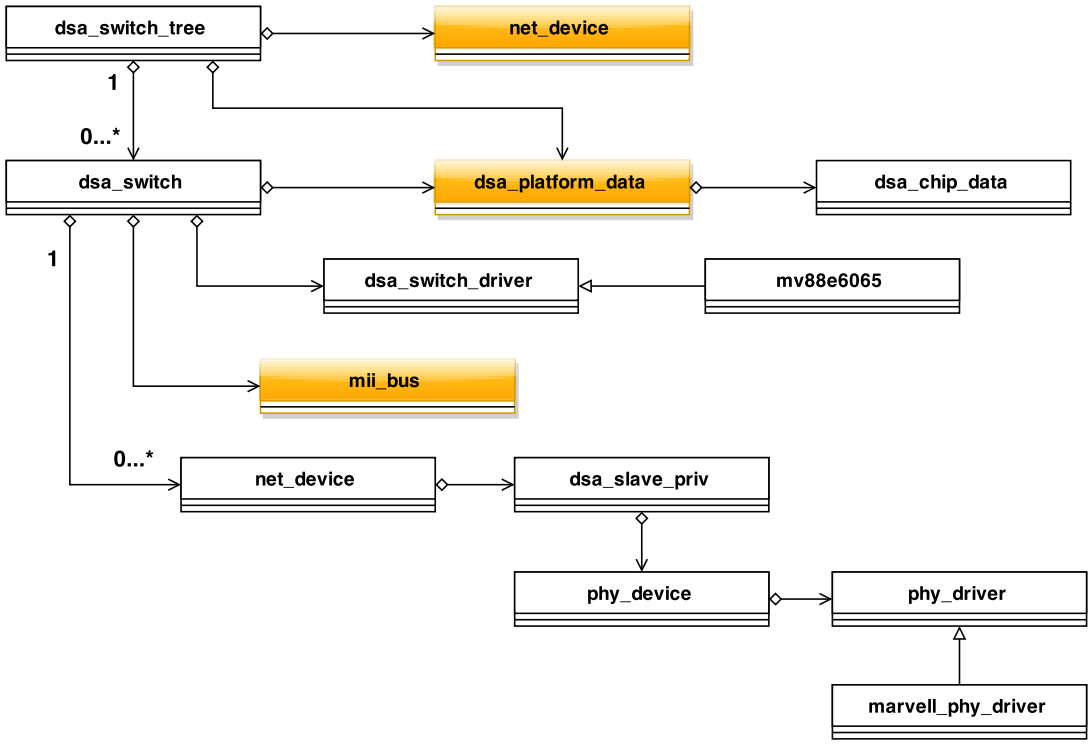
Subsystém DSA podporuje i propojení mezi přepínači tak, jak je možno vidět na obrázku 2.14. Takto propojené přepínače lze ovládat skrz MII rozhraní.

[1]) Ethtool je standardní nástroj Linuxu pro ovládání a podporu ovladačů a zařízení.



Obrázek 2.14. Možné připojení DSA přepínačů k procesoru. Obrázek je převzat z [10].

Na obrázku 2.15 je diagram tříd důležitých struktur v subsystému DSA, které jsou nezbytné pro funkčnost subsystému.



Obrázek 2.15. Diagram tříd vybraných datových struktur DSA subsystému.

Na tomto obrázku 2.15 jsou zvýrazněny struktury *mii_bus*, *net_device* a struktura *dsa_platform_data*. Tyto struktury DSA subsystém nevytváří, a proto je potřeba je předat subsystému způsobem, který je popsán v kapitole 4.4 Implementace.

Instance subsystému DSA je reprezentována strukturou *dsa_switch_tree*, která obsahuje odkazy na struktury:

- dsa_platform_data – tato struktura obsahuje informace o kořenovém zařízení. Také obsahuje strukturu *dsa_chip_data* popsanou níže.
- net_device – tato struktura odkazuje na hlavní síťové zařízení (viz sekce 2.3.3), která obsahuje informace o použitém protokolu.
- dsa_switch – obsahuje data pro jednotlivé přepínače:
  - dsa_switch_tree – je odkaz na rodičovskou strukturu.
  - dsa_chip_data – struktura obsahující informace o fyzickém přepínači – kolik má portů a jaké jsou jejich názvy. Navíc obsahuje informace o popisu hardwaru ve zkompilovaném device tree [1]).
  - device – je odkaz zařízení přepínače.
  - dsa_switch_driver – struktura popisující ovladač přepínačů, obsahuje funkce na správu připojeného přepínače. V jádře se struktura obsahující odkazy na funkce označuje jako tabulka virtuálních funcí [9].
  - mii_bus – je ovladač MII sběrnice.
  - net_device – je odkaz na síťové zařízení reprezentující port zařízení.

2.3.8 Souborový systém sysfs

Souborový systém sysfs je obsažen v Linuxovém jádře od verze 2.6. Umožňuje jádru operačního systému exportovat informace do uživatelského prostoru za použití paměti [12]. Hlavním účelem je reprezentovat objekty jádra, jejich atributy a jejich vzájemné vztahy. Většina atributů je reprezentována textovými soubory, které jsou v kódování ASCII a obsahují pouze jednu hodnotu. Sysfs poskytuje dvě rozhraní k reprezentování dat:

- kernel programming interface – slouží k exportování a importování položek (viz níže) skrz sysfs do jádra.
- user interface – slouží k zobrazení a manipulaci s těmito položkami (viz níže), které mapuje zpět na objekty v jádře.

Mapování objektů a atributů na objekty v sysfs můžeme vidět v tabulce 2.2.

Interní	Externí
Jaderné objekty	Složky
Atributy objektů	Soubory
Vazby mezi objekty	Symbolické odkazy

Tabulka 2.2. Tabulka popisující mapování objektů a atributů z jádra. Tabulka převzata z [12].

[1]) Device tree je struktura, předávána zavaděčem operačního systému, která popisuje připojení hardwaru. A tak odlehčuje kódu, který by popisoval každý detail. [11].

Sysfs je připojen do složky */sys/*. Složka /sys/ může vypadat následovně:

```
/sys/
|-- block
|-- bus
|-- class
|-- devices
|-- firmware
|-- module
‘-- power
```

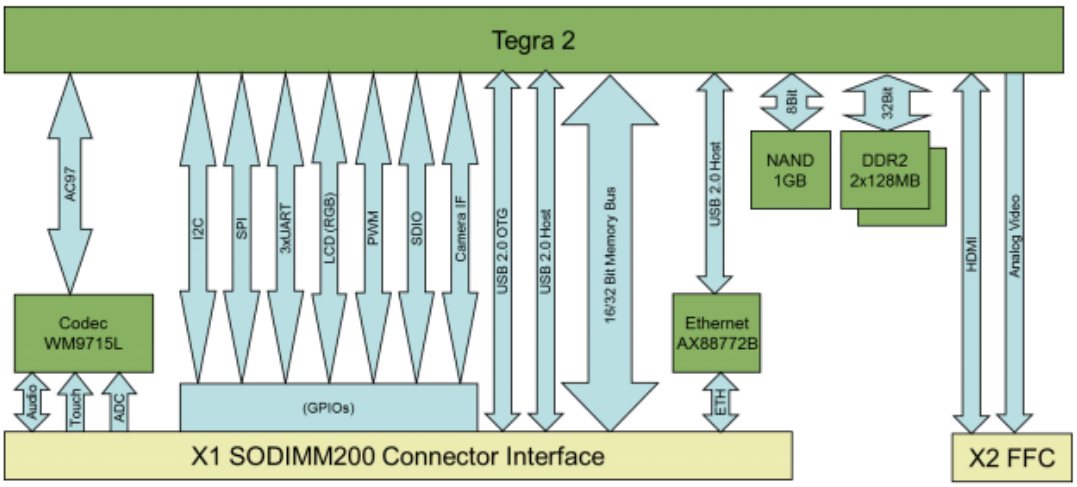
Sysfs je reprezentováno v Linuxovém jádře jako struktura *kerfs_root*, která se skládá z uzlů *kernfs_node*. Tyto struktury využívají instance struktur kobj k reprezentaci dat, a také obsahují tři druhy obslužného volání – na zápis, čtení a na uvolnění.

Kapitola 3
Použitý hardware

Tato kapitola popisuje použitý procesorový modul, na kterém byl provaděn vývoj ovladačů, desku konfigurovatelného přepínače a čipy, které jsou na této desce použity.

3.1 Procesorový modul Colibri T20

Modul Colibri T20, vyráběný firmou Toradex, je počítačový modul založen na NVIDIA Tegra 2 vestavěném systému na čipu [13]. Tento modul dále obsahuje 256 MB DDR2 paměti a 512 MB NAND flash paměti. Má velké množství komunikačních rozhraní, jako je I2C, SPI, UART, MMC, USB a Ethernet a výstup pro PWM. Také obsahuje audio a video výstup. Čip obsahuje periferie popsané v blokovém diagramu na obrázku 3.1.



Obrázek 3.1. Procesorový modulu Colibri T20. Obrázek převzat z [13].

Firma Toradex dodává k tomuto modulu podporu pro Linuxové jádro verze 3.1.10.

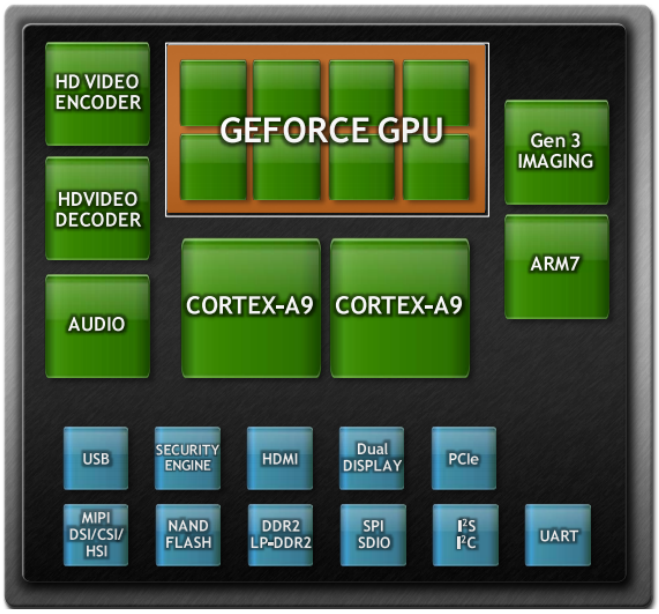
3.2 Systém na čipu NVIDIA Tegra 2

Systém na čipu NVIDIA Tegra 2 od firmy NVIDIA je založen na dvoujádrovém procesoru Cortex A9 se symetrickým procesorovým jádrem s frekvencí okolo 1 GHz [14]. Jelikož se jedná o systém na čipu, obsahuje tento čip mnoho bloků, jako je například Audio/Video rekordér, grafický procesor s podporou 2D rendrovaní a 3D pixelové

a vektorové shadery. Tento systém na čipu dosahuje vysokého grafického i výpočetního výkonu, viz [15].

Procesor Cortex A9 má 32-bitovou RISC [1]) architekturu vyvinutou společností ARM Holding věnující se vývoji procesorových jader. Tento procesor podporuje out–of–order a spekulativní provádění instrukcí. Má plnou podporu koherence pamětí pro symetrické procesory. Procesor též obsahuje jednotku pro výpočty s plovoucí řádovou čárkou.

Bloky systému na čipu NVIDIA Tegra 2 lze vidět na obrázku níže 3.2.



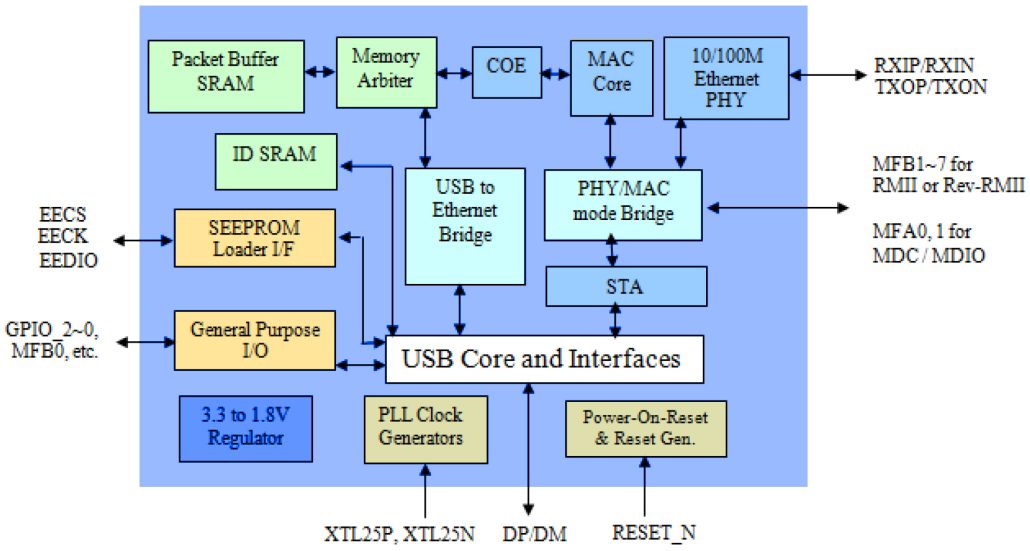
Obrázek 3.2. Bloky procesoru NVIDIA Tegra 2. Obrázek převzat z [14].

3.3 USB–Ethernet kontrolér Asix AX88772b

AX88772b je zákaznický integrovaný obvod vyráběný firmou Asix, který umožňuje připojení Fast Ethernet rozhraní za běhu systému. To je určeno pro zařízení disponující hostitelským rozhraním USB verze 1.1 nebo verze 2.0.

Jako jediný z vyráběných modelů dosahuje v revizi AX88772b teplotních rozsahů od -40 do $+85°C$. Což odpovídá průmyslovým teplotním rozsahům. AX88772b má modifikovatelnou vícefunkční sběrnici, která umožňuje připojit RMII rozhraní, popsané v sekci 2.2.3, nebo Reverse-RMII způsobem MAC–to–MAC k mikrokontroléru s Ethernet MAC RMII rozhraním. MAC rozhraní je plně kompatibilní se standardy IEEE 802.3, IEEE 802.3u.

[1]) Architektura RISC označuje procesory s redukovanou instrukční sadou.



Obrázek 3.3. Blokový diagram čipu AX88772b od firmy Asix. Obrázek je převzat z [16].

Jak je vidět na obrázku 3.3, čip AX88772b obsahuje Ethernetovou vrstvu PHY, která však nemůže být aktivní současně s RMII rozhraním. Dále obsahuje periferii pro připojení EEPROM paměti a nastavitelné vývody, nejčastěji používáné pro připojení LED diod. Tento čip též obsahuje regulátor napětí z 3,3 V na 1,8 V.

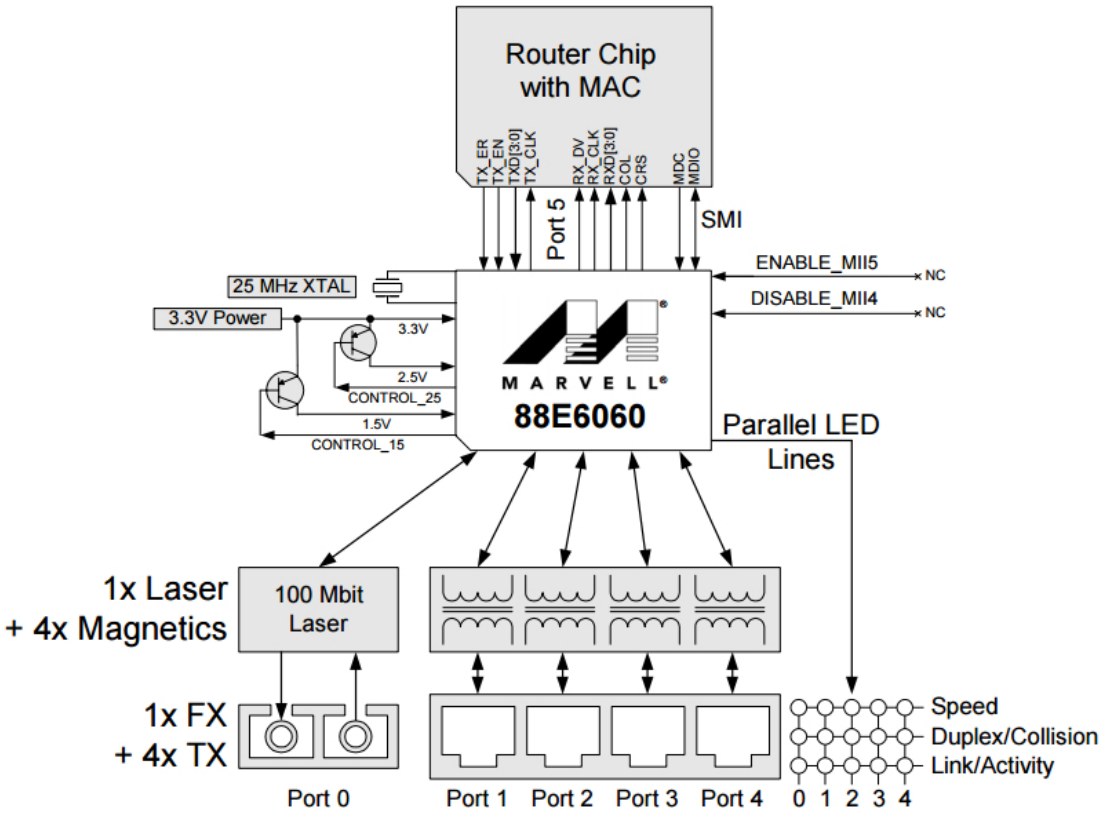
3.4 Ethernetový přepínač Marvell 88E6065

Marvell 88E6065 je 5 + 1 portový Ethernetový přepínač integrovaný na čipu. Obsahuje pět portů s fyzickou vrstvou 10 BASE-T/100 BASE-TX, z nichž dva porty mohou být využity pro optickou linku 100 BASE-FX. Navíc obsahuje port, pro připojení MAC vrstvy pomocí RMII nebo MII rozhraní. Též jde připojit pouze přes ovládací rozhraní z MII. Zařízení má vysokorychlostní neblokující čtyřúrovňový QoS [1]).

Vrstva PHY tohoto zařízení podporuje plug–and–play s možností automatického určení směru, automatického určení polarity a automatického určení rychlosti. Zařízení dále podporuje 64 z 4096 802.1Q WLAN s 3 úrovňovou ochranou. Obsahuje dvě RMII/MII/SNI rozhraní, která mohou být připojena k ovládacímu rozhraní nebo ke kontroléru s integrovanou vrstvou MAC. MAC a PHY vrstvy jsou plně kompatibilní se standardy IEEE 802.3, IEEE 802.3u a IEEE 802.3x.

Zařízení může být konfigurováno přes ovládací rozhraní nebo může načítat nastavení z EEPROM paměti.

[1]) QoS umožňuje síti, garantovat lepší kvalitu služeb. A umožňuje aplikacím požadovat a přijímat předvídatelné úrovně těchto služeb, aby zvýšili průtok sítě a její odezvu. [17].



Obrázek 3.4. Zapojení systému s čipem 88E6060 od firmy Marvell, které je velmi podobné zapojení čipu 88E6065. Obrázek je převzat z [18].

Jak lze vidět na obrázku 3.4, který popisuje obvod Marvell 88E6060, který je velmi podobný obvodu Marvell 88E6065. Oba tyto modely potřebují externí hodinový krystal a zdroj napětí.

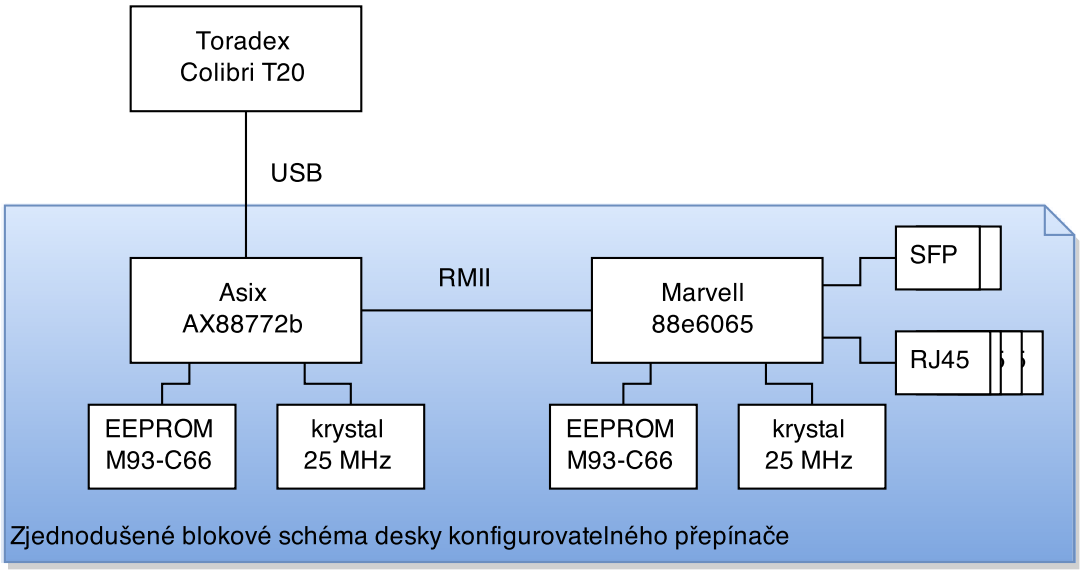
3.5 Deska konfigurovatelného přepínače

Deska konfigurovatelného přepínače byla vyrobena firmou Retia a.s. [1]) na základě jejich návrhu a zapojení, ktará jsou popsaná v přílohách B.1, B.2.

Tato deska byla vyrobena za účelem nahrazení staršího modelu konfigurovatelného přepínače, který splňuje průmyslové teplotní rozsahy, za výkonnější.

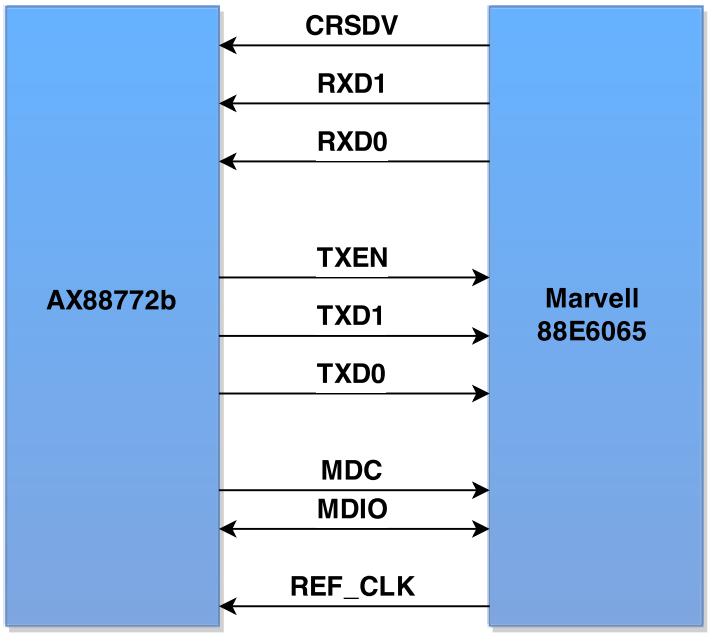
Zjednodušené schéma této desky je znázorněno na obrázku 3.5.

[1]) Firma zabývající se elektronickými vojenskými systémy, záznamovými zařízeními a lokalizačními a bezpečnostními systémy. `www.retia.cz`



Obrázek 3.5. Blokové schéma zapojení desky konfigurovatelného přepínače.

Konfigurovatelný přepínač umožňuje nastavení Ethernetového přepínače za pomoci USB Ethernet kontroléru AX88772b od firmy Asix 3.3, ke kterému je připojena externí pamět EEPROM M93-C66 [1]). Tento kontrolér je připojen přes RMII rozhraní k čipu Marvell 88E6065 3.4, viz na obrázku níže 3.6.



Obrázek 3.6. Připojení RMII rozhraní mezi zařízeními Marvell 88E6065 a Asix AX88772b. Obrázek upraven podle [18].

I k tomuto čipu je připojena externí pamět EEPROM.

[1]) Popsané na `http://www.st.com/web/catalog/mmc/FM76/CL1276/SC112/PF63995?referrer=70071840`

Oba čipy Asix AX88772b a Marvell 88E6065 mají vlastní krystal s frekvencí 25 MHz. Deska je navíc osazena regulátorem napětí a přepínači pro různá nastavení MII rozhraní, jak je možno vidět v příloze B.1.

Kapitola 4
Implementace

Tato kapitola popisuje portaci Linuxového jádra verze 3.19 pro modul Colibri T20 3.1, která byla zapotřebí pro další implementaci ovladačů. Dále se zabývá způsobem oživení desky konfigurovatelného přepínače 3.5 a následným popisem tvorby ovladače pro tento přepínač 4.4 a jeho implementace 4.5.

Vývoj byl prováděn na vývojovém kitu s modulem Colibri T20. Pro vývoj ovladače bylo zvoleno jádro z upstreamové větve Linuxového jádra [1]). Práce byla započata s verzí Linuxového jádra 3.17, a později aktualizována na verzi 3.19.

4.1 Portace Linuxového systému

Pro portaci hlavní větve Linuxového jádra bylo vyžadováno několik nástrojů a programů pro překlad kódu, jako je kompilátor pro architekturu ARM, v našem případě použit kompilátor Linaro[2]), který též obsahuje linker, jež je využíván pro slinkování přeložených kódů. Dále je potřeba nástroj pro verzování git a nástroj pro kontrolu generování spustitelných souborů – make.

Byla zvolena novější verze jádra, než kterou podporuje Toradex, a tou je verze 3.1.10. Důvodem byla komunikace s vývojáři hlavní větve Linuxového jádra a následné poslání patche.

4.1.1 Zavaděč U-Boot

Jako první krok bylo potřeba nainstalovat zavaděč, který by se postaral o zavedení Linuxového jádra na modul Colibri T20. Pro tento úkol byl zvolen zavaděč U-Boot [3]), který je využíván i výrobcem modulu Colibri T20, avšak byla zvolena jeho novejší verze.

V tomto kroku bylo nutné stáhnout a zkompilovat nejnovější zavaděč. Jako konfigurační soubor byl zvolen *colibri_t20_iris_config*, který popisuje modul Colibri T20 s vývojovou deskou Iris.

```
> export CROSS_COMPILE=~/PATH/gcc-linaro/bin/arm-linux-gnueabihf-
> export ARCH=arm
> git clone git://git.denx.de/u-boot.git
> make colibri_t20_iris_config
> make -j8
```

[1]) Linuxové jádro je možné stáhnout na `http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git`
[2]) To je možné stáhnout z `https://launchpad.net/gcc-linaro`.
[3]) U-Boot je možný stáhnout z repozitáře `git://git.denx.de/u-boot.git`

Dále bylo potřeba stáhnout nástroj, který by sloužil k vytvoření cílového binárního souboru zavaděče U-Boot. Ten využívá configurační soubory od firmy Toradex, které se spojí s přeloženým zavaděčem, které zkopírujeme do kořenového adresáře nástroje cbootimage.

```
> git clone https://github.com/NVIDIA/cbootimage-configs.git
> cd cbootimage-configs
> cbootimage -gbct -t20
tegra20/toradex/colibri-t20/colibri-t20_512_v12_nand.bct.cfg
  colibri-t20_512_v12_nand.bct
> cp ~/PATH/u-boot/u-boot-dtb-tegra.bin u-boot.bin
> cbootimage -t20
  tegra20/toradex/colibri-t20/colibri-t20_512_v12_nand.img.cfg
  colibri-t20_512_v12_nand.img
```

Poté byl zavaděč zaveden z diskového souboru následujícím příkazem:

```
> tegrarcm --bct colibri-t20_512_v12_nand.bct --loadaddr=0x00108000
  --bootloader=../u-boot/u-boot-dtb-tegra.bin
```

Po zavedení tohoto zavaděče bylo možné sledovat, jak se na konzoli modulu Colibri T20 zobrazil výpis tohoto zavaděče. A za pomoci USB-Ethernetového kontroléru, který byl integrovaný na modulu, byl nahrán obraz zavaděče do paměti RAM modulu.

```
Tegra20 (Colibri) # usb start
(Re)start USB...
USB1: USB EHCI 1.00
scanning bus 1 for devices... 1 USB Device(s) found
USB2: USB EHCI 1.00
scanning bus 2 for devices... 2 USB Device(s) found
 scanning usb for storage devices... 0 Storage Device(s) found
 scanning usb for ethernet devices... 1 Ethernet Device(s) found
Tegra20 (Colibri) # setenv ipaddr 192.168.80.90
Tegra20 (Colibri) # setenv serverip 192.168.80.3
Tegra20 (Colibri) # tftpboot 0x02100000 colibri-t20_512_v12_nand.img
Waiting for Ethernet connection... done.
TFTP from server 192.168.80.3; our IP address is 192.168.80.90
Filename 'colibri-t20_512_v12_nand.img'.
Load address: 0x02100000
Loading: #################################
         3.6 MiB/s
done
Bytes transferred = 729088 (b2000 hex)
```

Bylo nutné nahrát obraz zavaděče do paměti NAND a přepsat původní zavaděč současným. Délka přeneseného obrazu byla 0xb2000, viz výpis výše. Tato hodnota byla potřeba k určení délky přepisovaného úseku NAND paměti.

```
Tegra20 (Colibri) # nand erase.chip

NAND erase.chip: device 0 whole chip
Erasing at 0x3ffc0000 -- 100% complete.
OK
Tegra20 (Colibri) # nand write 0x02100000 0 0xb2000
```

```
NAND write: device 0 offset 0x0, size 0x95000
 577536 bytes written: OK
```

Po resetu zařízení byl na sériové konzoli vidět výpis aktualizovaného zavaděče U-Boot, který ale v tento moment nebyl schopen nalézt žádné jádro operačního systému. Jádro bylo přidáno v dalším kroku popsaném v sekci 4.1.2.

4.1.2 Linuxové jádro

Následujícím krokem byla instalace aktuální verze jádra, která byla nutná k vývoji ovladačů a komunikaci s Linuxovou komunitou. Jelikož upstreamové jádro nepodporuje řadič NAND paměti, který je součástí čipu Colibri T20, načítání tohoto jádra bylo realizováno přes Ethernetovou síť [19].

Pro instalaci Linuxového jádra bylo potřeba stáhnout a přeložit verzi 3.19 [19]. Pro nastavení jádra byla zvolena konfigurace *tegra_defconfig*, která obsahuje moduly a ovladače potřebné pro jeho běh.

```
> git clone git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
> cd linux
> git checkout v3.19
> export LOADADDR=0x408000
> export ARCH=arm
> export CROSS_COMPILE=/PATH/gcc-linaro/bin/arm-linux-gnueabihf-
> make tegra_defconfig
> make -j8 uImage
```

Po přeložení verze 3.19 bylo potřeba nahrát obraz jádra a přeloženého device tree[1]) do RAM paměti modulu za pomoci příkazů pro zavaděč U-Boot po sériové lince následujícími příkazy:

```
Tegra20 (Colibri) # usb start
Tegra20 (Colibri) # setenv ipaddr 192.168.80.90
Tegra20 (Colibri) # setenv serverip 192.168.80.3
Tegra20 (Colibri) # tftpboot ${kernel_addr_r} uImage
Tegra20 (Colibri) # tftpboot ${fdt_addr_r} tegra20-iris-512.dtb
```

Po nahrání obrazů byly nastaveny parametry pro Linuxové jádro [2]), tedy nastavení konzole, jejího portu a rychlosti, dále pak nastavení oddílu s distribucí a jejího typu souborového systému. Pokud v nastaveném oddílu bude obraz souborového systému s nainstalovanou Linuxovou distribucí, jádro tuto distribuci zavede.

```
Tegra20 (Colibri) # set bootargs console=ttyS0,115200n8 earlyprintk
root=/dev/mmcblk0p1 rootfstype=ext2 rw rootdelay=1
Tegra20 (Colibri) # bootm ${kernel_addr_r} - ${fdt_addr_r}
```

[1]) Device tree je struktura, předávána zavaděčem operačního systému, která popisuje připojení hardwaru. A tak odlehčuje kódu, který by popisoval každý detail. [11].
[2]) `https://www.kernel.org/doc/Documentation/kernel-parameters.txt`

4.2 Zavedení Linuxové distribuce OpenWrt

Pro zavedení Linuxové distribuce OpenWrt, která je postavena na nástroji Buildroot[1]), je potřeba stáhnout zdrojové kódy.

```
> git clone git://git.openwrt.org/openwrt.git
```

Poté musíme nastavit nástroj pro křížovou kompilaci nebo výběrem cílové architekturu.

```
cd openwrt
make menuconfig
```

Po nastavení cílové architektury a vzhledu systému, můžeme přeloži náši distribuci.

```
make -j8
```

Poté můžeme zkopírovat vzniklý souborový systém na SD kartu, která je zformátovaná na podporovaný souborový systém jádra operačního systému tak, jak je vidět v sekci 4.1.2.

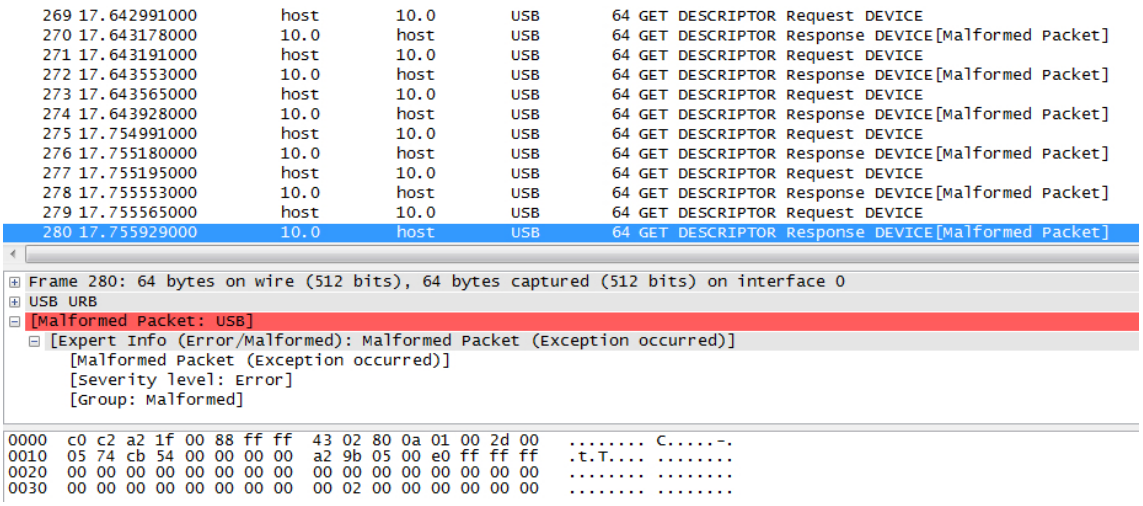
4.3 Oživení desky přepínače

Pro oživení desky byl nejprve testován USB kontrolér Asix AX88772b, viz sekce 4.3.1. Po oživení komunikačního rozhraní tohoto kontroléru bylo dalším krokem ladění Ethernetového přepínače Marvell 88E6065, viz sekce 4.3.2. Nakonec bylo provedeno testování Ethernetových portů přepínače. V průběhu tohoto testování bylo objeveno několik chyb v návrhu desky, které byly na základě mých připomínek opraveny.

4.3.1 Oživení USB–Ethernet kontroléru Asix AX88772b

V průběhu oživování USB kontroléru byla objevena závada v jeho připojení na desce konfigurovatelného přepínače. Tato závada se projevovala chybnou komunikací na USB sběrnici, která byla objevena při použití modulu Colibri T20. Tato chybná komunikace je znázorněna na obrázku 4.1. Je zde znatelné, že pakety, které dorazily k hostovskému zařízení, modul příjímal poškozené.

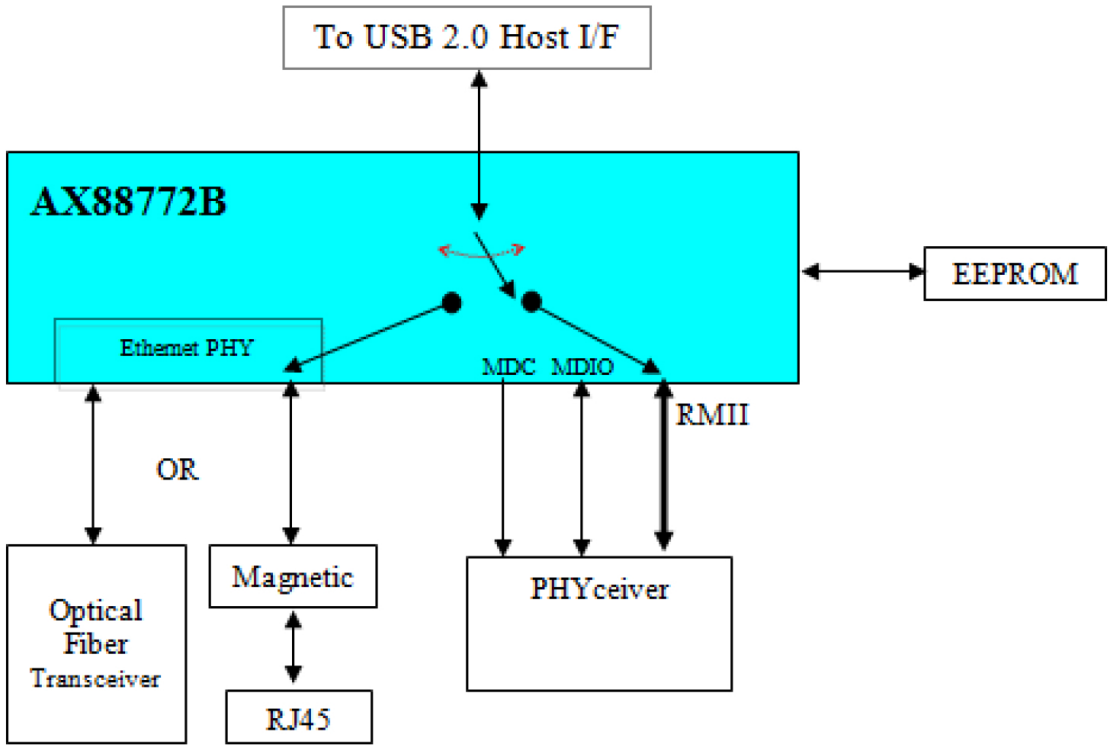
[1]) Buildroot je založen souborech typu Make určených pro křížovou kompilaci nástrojů, souborového systému, jádra a zavaděče.



Obrázek 4.1. Záznam chybné komunikace na USB sběrnici mezi PC a USB kontrolérem Asix AX88772b, která byla zavině na nepřipojením blokovacího kondenzátoru.

Po úpravě zapojení začal USB kontrolér odpovídat na dotazy aktuálního ovladače v operačním systému Linux. To bylo možné ověřit z uživatelského prostoru příkazem:

```
lsusb
```



Obrázek 4.2. připojení vrstvy PHY, popřípadě rozhraní RMII. Obrázek převzat z [16].

Z obrázku 4.2 lze vyčíst, že kontrolér AX88772b není schopen komunikovat po rozhraní PHY a RMII současně. Zápisem do registru číslo 0 [16] je aktivováno právě jedno z daných dvou rozhraní a též musí být správně nastanena spojka K2, viz B.1, která určuje funkčnost víceúčelových vývodů.

4.3.2 Oživení Ethernetového přepínače Marvell 88E6065

K oživení Ethernetového přepínače Marvell 88E6065 bylo zapotřebí aktivovat RMII rozhraní a provést několik zásahů do zapojení čipu na desce. Taktéž bylo zapotřebí aktivovat funkční komunikaci přes kontrolér Asix AX88772b, jak je popsané v sekci 4.3.1, který na základě USB příkazů je schopen posílat ovládací zprávy přes RMII rozhraní. Navíc bylo potřeba vyřešit, v jakém módu bude tento Ethernetový přepínáč připojen. Pro připojení přepínače bylo zvoleno připojení MAC-to-MAC, ve kterém hodinový signál pro signál MDIO vysílá zařízení AX88772b a hodiny o frekvenci 50 MHz pro RMII rozhraní Ethernetový přepínač Marvell 88E6065. Zapojení je znázorněno na obrázku 3.6.

Protože i v zapojení čipu byla nalezena chyba v komunikaci, byly doplněny chybějící blokovací kondenzátory. Navíc byl objeven problém, že tento čip se po každém zapnutí nastavil do odlišných nastavení. Pro vyřešení tohoto problému byl k čipu doplněn kondenzátor, který zařízení opakovaně resetoval po dobu časové konstanty RC.

Deska tak měla během této doby možnost, aby se vstupní hodnoty na všech vývodech ustálily ve správném stavu.

Pro zajištění správného a deterministického chování přepínače byly přidány pull-up a pull-down rezistory viz B.2.

- pull up rezistor – na vývodech SW_MODE, ADDR4, P5_MODE0, P5_MODE1 a P5_MODE3.
- pull down rezistor – na vývodu P5_MODE2.

4.4 Návrh subsystému

Pro dané zadání bylo potřeba vytvořit subsystém, který by propojoval ovladač Ethernetového přepínače Marvell 88E6065 se subsystémem DSA a s ovladačem pro kontrolér Asix AX88772b.

Cílem návrhu bylo propojení těchto částí tak, aby dodaná deska konfigurovatelného přepínače byla podporována operačním systémem Linux a aby tento přepínač bylo možno nastavit z uživatelského prostoru operačního systému.

Za tímto účelem bylo nejdříve potřeba rozšířit ovladač Asix AX88772b o podporu RMII rozhraní, které nebylo naimplementováno v operačním systému Linux, a o možnost připojení k subsystému DSA. To bylo vyřešeno vytvořením souboru s obslužnou rutinou v souborovém systému sysfs. Tato obslužná rutina aktivuje RMII rozhraní, přidělí potřebnou paměť a nastaví ji pro subsystém DSA a následně inicializuje tento subsytém. Navíc byla rozšířena uvolňovací funkce ovladače. Kromě uvolnění samotného ovladače také volá uvolnění subsytému DSA. Toto rozšíření bylo provedeno z důvodu navrácení přidělené paměti operačnímu systému.

Dalším krokem byla úprava subsystému DSA. Tento subsystém vytváří každému portu Ethernetového přepínače rozhraní viditelné z uživatelského prostoru operačního

systému a připojuje k nim příslušný ovladač pro daný Ethernetový přepínač. Tento sybsystém bylo potřeba rozšířit o schopnost uvolňování a alokace za běhu.

Následovalo vytvoření ovladače Marvell 88E6065. Tomuto ovladači byla přidána schopnost nastavit požadované chování Ethernetového přepínače a komunikovat skrz USB kontrolér.

Nakonec operační systém Linux zajistí, aby ovladače spolu komunikovaly a aby byl po připojení USB kontroléru a inicializaci subsystému DSA přiřazen správný ovladač Ethernetového přepínače.

4.5 Zpracování

Sekce zpracování popisuje provedení jednotlivých částí ovladače podle návrhu popsaném v předchozí sekci 4.4.

Jelikož nebyl ovladač pro zařízení Asix AX88772b připraven na připojení RMII rozhraní, bylo zapotřebí tento ovladač rozšířit o aktivaci rozhraní. To však nemůže být aktivní současně s rozhraním Ethernet PHY, které je taktéž integrováno na čipu, jak je zobrazeno na obrázku 4.2. To bylo provedeno zapsáním dat do registru označeném SW_PHY_SELECT, který aktivoval druhé rozhraní. Dále bylo potřeba zaregistrovat strukturu mii_bus, která bude sloužit k ovládání RMII rozhraní tohoto kontroléru.

Poté bylo potřeba upravit subsystém DSA tak, aby podporoval inicializace zařízení za běhu a jeho odhlašování a uvolňování. Posledním krokem bylo vytvoření samotného ovladače pro zařízení Ethernetového přepínače Marvell 88E6065, který řeší jeho funkčnost.

4.5.1 Změna ovladače zařízení Asix AX88772b

Ovladač zařízení Asix AX88772b je již přítomen v Linuxovém jádře. Ale jeho verze neměla implementovanou podporu rozhraní RMII, proto bylo potřeba doplnit ovladač o aktivaci tohoto rozhraní.

Byla zvolena možnost, že aktivace RMII rozhraní bude provedena přes zápis do souboru *usb_dsa_bind*, který je součástí souborového systému sysfs a je umístěn ve složce DSA_BIND, která je podsložkou USB zařízení. Tímto zápisem do souboru se nejen aktivuje RMII rozhraní, ale také se provede navázání se subsystémem DSA.

Vytvoření složky, která obsahuje zmíněný soubor, lze vidět na kódu níže:

```
1   priv->kobj.kset = kset_create_and_add("DSA_BIND", NULL,
                          kobject_get(&dev->udev->dev.kobj));
2   if (!priv->kobj.kset){
3      ret = -ENOMEM;
4      goto free;
5   }
6
7   ret = kobject_init_and_add(&priv->kobj, &dsa_bind_ktype,
                          NULL, "dsa_bind");
8   if (ret)
9      goto free_kobj;
```

Na řádku 7 tohoto kódu se v instanci *dsa_bind_ktype* struktury *kobj_type* předávají atributy a operace obslužné rutiny pro zápis a čtení ze souboru.

Popis struktur popisující atributy a operace souboru *usb_dsa_bind*, který obsahuje obslužnou rutinu, lze vidět níže.

```
1  static struct asix_attribute asix_attribute =  __ATTR(usb_dsa_bind,
0664, usb_dsa_show, usb_dsa_store);
2  static struct attribute *asix_default_attrs[] = {
3     &asix_attribute.attr,
4     NULL,
5  };
6  static const struct sysfs_ops dsa_bind_sysfs_ops = {
7     .show   = usb_dsa_attr_show,
8     .store  = usb_dsa_attr_store,
9  };
10 static struct kobj_type dsa_bind_ktype = {
11    .sysfs_ops      = &dsa_bind_sysfs_ops,
12    .release        = driver_release,
13    .default_attrs  = asix_default_attrs,
14 };
```

Po zápisu do tohoto souboru byla vyvolána obslužná rutina *usb_dsa_store()*, jejíž ukazatel byl nainicializován v průběhu inicializace ovladače Asix AX88772b opsané výše.

Tato rutina následně vyvolala funkci *ax88772_set_bind_dsa()*, která vypadá následovně:

```
1  static int ax88772_set_bind_dsa(struct asix_common_private *priv)
2  {
3     struct usbnet *dev = priv->dev;
4     int i, ret, embd_phy;
5     u32 phyid;
6     int temp = AX88772_RMII;
7
8     /* Enable RMII interface for external PHY */
9     asix_write_cmd(dev, AX_CMD_SW_PHY_SELECT, 0, 0, 1, &temp);
10
11    for (i = 0; i < AX88772_MAX_PORTS; i++) {
12       phyid = asix_mdio_read(dev->net, i, 0x3);
13       if (phyid == MV88e6065_ID)
14          break;
15    }
16
17    if (phyid == MV88e6065_ID) {
18       ret = ax88772_init_mdio(dev);
19       if (ret)
20          return ret;
21
22      priv->use_embphy = 1;
23
24      priv->pdev = dsa_create_pdev(priv->mdio, dev->net);
25      if (priv->pdev == NULL)
26         return -ENOMEM;
```

```
27
28          ret = dsa_probe_net_device(priv->pdev, dev->net);
29          if (ret)
30             return ret;
31      }
...
```

Nejprve tato funkce aktivuje RMII rozhraní zapsáním do registru, jak lze vidět na řádku 9. Na řádcích 11 až 15 je znázorněna její další funkce, a to testování, zda bylo k USB kontroléru připojeno zařízení Marvell 88E6065 dle standardizovaného registrového prostoru, viz obrázek 2.8, konkrétně vyčtením z registru číslo 3. Ten obsahuje identifikátor a revizi zařízení. Pokud bylo objeveno zařízení Marvell 88E6065, tato funkce navíc vyplní strukturu *asix_common_private* a následně zavolá funkci *dsa_probe_net_device*.

Struktura *asix_common_private* ve verzi jádra 3.19 obsahuje následující prvky:

- kobject – struktura popisující objekt v souborovém systému sysfs.
- mii_bus – struktura odkazující na komunikační rozhraní USB.
- use_embphy – příznak informující o typu použitého rozhraní (RMII nebo PHY).
- dsa_up – příznak informující o použití subsystému DSA.
- usbnet – struktura odkazující na zařízení Asix AX88772b.
- platform_device – struktura popisující Ethernetový přepínač, jako je počet portů a jejich názvy. Je naplněná ve funkci *dsa_crete_pdev*.

Po naplnění základních struktur ovladač Asix AX88772b nainicializuje subsystém DSA zavoláním funkce *dsa_probe_netdevice()*. Následná změna DSA subsystému je popsaná v 4.5.2.

4.5.2 Úprava subsystému DSA

Subsystém DSA byl rozšířen o veřejnou funkci *dsa_probe_netdevice()*, která vytváří instanci *dsa_switch_tree* reprezentující Ethernetový přepínač. Každý port přepínače je reprezentován vlastním virtuálním Ethernetovým rozhraním net_device popsaným v sekci 2.3.3. Tato Ethernetová rozhraní jsou závislá na rodičovském síťovém rozhraní ovladače Asix AX88772b, které je jim nadřazené.

V části ovladače, která má za úkol uvolňovat paměť, bylo potřeba provést několik změn, jelikož v původním provedení se nepředpokládalo, že by se tento ovladač někdy uvolňoval. To plyne z debaty na [20]. Funkce na uvolňování ovladače nebyla v původní verzi dokončena, a proto byla upravena část její uvolňovací rutiny *dsa_destroy()*, která ve verzi 3.19 Linuxového jádra podporuje uvolnění všech portů interpretovaných strukturou net_device a interních struktur DSA ovladače, jako je struktura *dsa_switch* a *dsa_switch_tree*.

4.5.3 Vytvoření ovladače zařízení Marvell 88E6065

Pro ovládání zařízení Marvell 88E6065 bylo zapotřebí vytvořit ovladač, protože není naimplementován ve verzi 3.19 v Linuxovém jádře.

Tento ovladač je instancí struktury dsa_switch_driver, jak je vidět na obrázku 2.15. Struktura dsa_switch_driver obsahuje metody, které zajišťují inicializování, uvolnění a nastavení přepínače jako celku i jeho jednotlivých portů, též obsahuje metody pro zápis na médium.

4.6 Spojení jednotlivých ovladačů

Všechny ovladače jsou implementovány tak, aby vykonávaly svou část.

Při připojení USB zařízení a vyčtení identifikátoru ze zařízení se zavede ovladač AX88772b, který aktivuje rozhraní PHY. Po zápisu do souboru *usb_dsa_bind* v adresáři příslušného USB zařízení se na čipu Asix AX88772b aktivuje rozhraní RMII a nainicializuje se Ethernetový přepínač. DSA subsystém tak vytvoří pro každý port přepínače rozhraní a připojí příslušný ovladač pro přepínací obvod. V případě dané desky se jedná o ovladač pro Ethernetový přepínač Marvell 88E6065.

Po úspěšném spojení jednotlivých ovladačů je možné konfigurovat ovladač pomocí uživatelského nástroje ethtool.

4.7 Odezva komunity

Změny v kódu Linuxového jádra, popsané v předchozích sekcích, byly posalány formou „patchů“ k okomentování komunitou vývojářů Linuxového jádra. Odezva komunity byla velmi rychlá [1]), odpovědi se mi dostalo během pouhých pár hodin. Vývojář Andrew Lunn se vyjádřil k mému nápadu a navrhl několik optimalizací: vyhnout se změnám v DSA subsystému a vytvořit obalovací ovladač pro ovladač kontroléru AX88772b.

Využil jsem radu vyhnout se změnám v subsystému DSA a needitoval jsem do velké míry alokující část subsystému DSA. Avšak druhou navrhnutou optimalizaci, kterou bylo vytvoření obalujícího ovladače pro ovladač Asix AX88772b, jsem nepoužil, protože má změna v tomto ovladači nijak neovlivňuje jeho chování.

Můj příspěvek strhl debatu o možnostech subsystému DSA v Linuxovém jádře – subsystém DSA prý není připraven pro podporu připojování a odpojování zařízení za běhu systému. Pravděpodobně však tato změna subsystému DSA nenastane ani v příštích verzích Linuxového jádra. Z diskuze však vyplynulo, že se pravděpodobně celý subsystém DSA v budoucnu přepíše.

4.8 Inicializace zařízení

Tato část testuje inicializaci zařízení a následné zapnutí Ethernetových portů přepínače. Nastavení připojeného konfigurovatelného přepínače bylo provedeno nástroji ifconfig a ip.

[1]) `http://lists.openwall.net/netdev/2015/04/21/20`

Dalším krokem po připojení zařízení bylo nastavení rodičovského síťového rozhraní jako aktivní. V našem případě se jednalo o zařízení *eth1*, které reprezentovalo zařízení Asix AX88772b.

```
> ifconfig eth1 up
```

Po aktivování rodičovského zařízení bylo nutné připojit tento uzel k DSA subsystému. To bylo provedeno zapsaním číslice 1 do souboru *usb_dsa_bind*, který se nachází ve složce patřící k příslušnému USB zařízení, v našem případě je tato cesta */sys/devices/soc0/c5008000.usb/usb3/3-1/3-1.1/DSA_BIND/dsa_bind/usb_dsa_bind*. Pak tento příkaz vypadá následovně.

```
> echo 1 > /sys/devices/soc0/c5008000.usb/\
    usb3/3-1/3-1.1/DSA\_BIND/dsa\_bind/usb\_dsa\_bind
```

Po tomto příkazu byl ovladač přepínače inicializován a vytvořil virtuální rozhraní pro každý ze svých portů. Zapnutí tohoto portu by mohlo probíhat například následovně, kde název rozhraní portu je *port1*.

```
> ifconfig port1 up
```

Tento příkaz aktivuje *port1*, který má nastavenou automatickou konfiguraci rychlosti a tento port není umístěn v žádné z VLAN.

Po provedení těchto příkazů se povedlo úspěšně nastavit Ethernetový port přepínače.

Kapitola 5
Testování

Tato kapitola se zabývá testováním implementovaného řešení zejména na propustnost a stabilitu sítě. Jelikož se jedná o systém s podporou inicializace za běhu, je také důležité, aby se v řešení neobjevovaly žádné neuvolněné bloky paměti.

5.1 Testování uvolňování paměti

Testování uvolňování paměti proběhlo několika způsoby.

Prvním způsobem bylo testování pomocí nástroje v Linuxovém jádře, kterou je Kernel Memory Leak Detektor [1]). Tento nástroj sleduje alokované bloky a ukladá si o nich informace. Pokud nalezne neuvolněný blok paměti, na který není žádná reference, vypíše chybovou hlášku, kterou můžeme vidět níže.

```
unreferenced object 0xdaafbfc0 (size 64):
  comm "kworker/0:1", pid 29, jiffies 4294949812 (age 272.380s)
  hex dump (first 32 bytes):
    64 73 61 5f 62 69 6e 64 00 5a 5a 5a 5a 5a 5a 5a  dsa_bind.ZZZZZZZ
    5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a  ZZZZZZZZZZZZZZZZ
  backtrace:
    [<c00b6ee0>] kstrdup+0x2c/0x54
    [<c01359f4>] __kernfs_new_node+0x28/0xc4
    [<c013691c>] kernfs_new_node+0x1c/0x38
    [<c0136c40>] kernfs_create_dir_ns+0x18/0x60
    [<c0138a60>] sysfs_create_dir_ns+0x48/0x94
    [<c0215cb0>] kobject_add_internal+0x9c/0x2d0
    [<c0216034>] kobject_init_and_add+0x48/0x70
    [<c0369184>] ax88772_bind+0x174/0x220
    [<c03741b0>] usbnet_probe+0x208/0x6b4
    [<c038a9ac>] usb_probe_interface+0x164/0x1f4
    [<c02f54c8>] driver_probe_device+0x10c/0x22c
    [<c02f3c64>] bus_for_each_drv+0x44/0x8c
    [<c02f5384>] device_attach+0x70/0x88
    [<c02f4adc>] bus_probe_device+0x84/0xa8
    [<c02f3034>] device_add+0x2ec/0x4e0
    [<c0388818>] usb_set_configuration+0x530/0x75c
```

Jak lze vidět z tohoto výpisu, nástroj Kernel Memory Leak Detektor drží informace ke každé alokované instanci objektu a ukladá o ní informace, jako je její velikost, stáří a místo v kódu a stav zásobníku, kde byla paměť alokována. Z těchto informací můžeme vyhledat, jaká funkce alokovala pamět, a na základě toho opravit kód.

[1]) Kernel Memory Leak Detektor je nástroj v Linuxovém jadře, který umožňuje nalezení špatného uvolnění paměti. Funguje podobně jako trasující garbage collector.

Chyba zmíněná výše nastala v mém kódu, kdy jsem zapomněl odregistrovat instanci typu kobj v subsystému sysfs.

Po opravení této chyby, další neočekavané neuvolnění paměti tento nástroj neobjevil.

Dalším způsobem bylo testování skriptem, který opakovaně registruje a odregistrovává ovladač. Ten objevil chybu v přístupu ke sdílenému objektu net_device, konkrétně v DSA subsystému, který obsahuje funkci *poll*, která je vyvolána časovačem. Tato funkce periodicky vyčítá stav portu ze zařízení.

Několikrát nastalo, že během zpracovávání funkce *poll* byla data, s nimiž funkce *poll* pracovala, uvolněna. To zapříčinilo zpanikaření jádra a následný pád systému.

Po uzamčení sdílených dat tento problém již nenastal.

5.2 Testování zařízení a výkonu

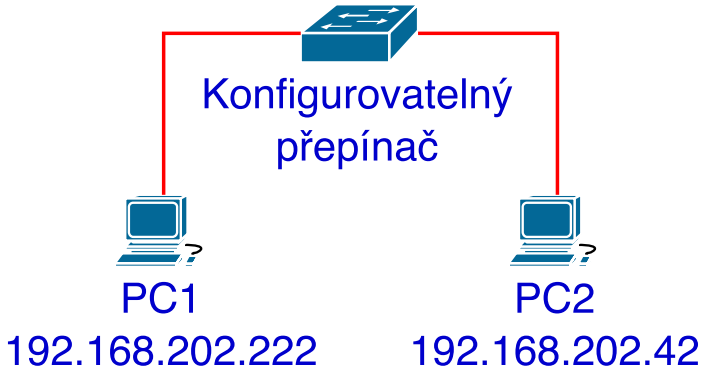
Cílem tohoto testu je otestovat kvalitu automatického nastavení fyzických linek přepínače a maximální rychlost přenosu dat přes přepínač.

V první řadě bylo potřeba se ujistit, že přepínač je dobře nastaven. To probíhalo kontrolou od nejnižších vrstev ISO/OSI, linkovou, až po nějvyšší, kterou je aplikační.

Po připojení UTP kabelu mezi přepínač a počítač bylo nutné zapnout port přepínače 4.8. Po nastavnení portu přepínače by zařízení mělo začít komunikovat s počítačem, tj. měla by proběhnout automatická konfigurace rychlosti. Toto je možné vidět níže z výpisu nástroje ethtool.

```
> ethtool eth0
Settings for eth0:
Supported ports: [ TP MII ]
Supported pause frame use: No
Supports auto-negotiation: Yes
Advertised pause frame use: Symmetric Receive-only
Advertised auto-negotiation: Yes
Link partner advertised link modes:  10baseT/Half 10baseT/Full
                                     100baseT/Half 100baseT/Full
Link partner advertised pause frame use: No
Link partner advertised auto-negotiation: Yes
Speed: 100Mb/s
Duplex: Full
Port: MII
PHYAD: 0
Transceiver: internal
Auto-negotiation: on
Link detected: yes
```

Zde je vidět, že je připojena fyzická linka a přepínač podporuje automatickou konfiguraci rychlosti, konktrétně 10 Mb/s a 100 Mb/s s obousměrným či jednosměrným přenosem dat.



Obrázek 5.1. Připojení konfigurovatelného přepínače pro testování komunikace.

Po připojení dalšího zařízení, aktivování dalšího z portů a nastavení IP adres vypadá zapojení tak, jak je uvedeno na obrázku 5.1. Po tomto připojení mohlo být testováno, zdali přepínač správně přenáší data. Mezi nejzákladnější testy přenosu dat patří ping, který je jeden z kontrolních zpráv ICMP protokolu.

```
> ping 192.168.202.222
PING 192.168.202.222 (192.168.202.222) 56(84) bytes of data.
64 bytes from 192.168.202.222: icmp_seq=1 ttl=64 time=0.368 ms
...
--- 192.168.202.222 ping statistics ---
11 packets transmitted, 9 received, 18% packet loss, time 9997ms
rtt min/avg/max/mdev = 0.368/0.424/0.466/0.042 ms
```

Jak je vidět na výpisu, zařízení PC2 úspěšně poslalo ICMP paket ping skrz přepínač, a také na tento dotaz dostalo odpověď formou ICMP paketu PONG.

Nyní bylo možno otestovat přenos dat na reálných datech, která byla pro tento účel načítána ze zařízení */dev/zero*. Pro spuštění testu rychlosti přenosu dat bylo nutno na jednom z těchto počítačů zapnout server, který bude naslouchat na volném portu, v našem případě na portu 4242. Pro tento úkol byl vybrán počítač PC1, který za použití nástroje netcat spustil server, který poslouchá na zvoleném portu.

```
> nc -lk -z 4242 > /dev/null
```

A na druhém počítači PC2 bylo nutno zapnout přenos dat na první počítač PC1 s danou IP adresou a daným portem tak, jak můžeme vidět na výpisu níže:

```
> dd if=/dev/zero bs=16000 count=625 | nc -v 192.168.202.222 4242
192.168.202.222 2112 (idonix-metanet) open
625+0 records in
625+0 records out
10000000 bytes (10 MB) copied, 0.877273 s, 11.4 MB/s
```

Na tomto příkladě bylo odesláno 10 MB dat v 625 paketech, které byly přenesena rychlostí 11.4 MB/s.

```
> dd if=/dev/zero bs=160000 count=625 | nc -v 192.168.202.222 2112
```

```
192.168.202.222 2112 (idonix-metanet) open
625+0 records in
625+0 records out
100000000 bytes (100 MB) copied, 8.49099 s, 11.8 MB/s
> dd if=/dev/zero bs=1600000 count=625 | nc -v 192.168.202.222 2112
192.168.202.222 2112 (idonix-metanet) open
625+0 records in
625+0 records out
1000000000 bytes (1.0 GB) copied, 85.9629 s, 11.6 MB/s
```

Tyto testy také byly prováděny na různých velikostech a rychlost přenosu se pohybovala v rozmezí od 11.3 MB/s do 11.8 MB/s. Toto rozmezí způsobuje délka a počet hlaviček paketů.

Pro kontrolu rychlosti přenosu byl proveden test nástrojem iperf:

```
> iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 4]loc 192.168.202.11 port 5001 connected with 192.168.202.22 port 3591
[ID]Interval       Transfer     Bandwidth
[ 4] 0.0-10.0 sec   110 MBytes  91.8 Mbits/sec
```

Ten přijal potvrzená data přenesená protokolem TCP s rychlostí okolo 11.47 MB/s.

```
> iperf -s -u -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[ 3]loc 192.168.202.11 port 5001 connected with 192.168.202.22 port 3468
[ID]Interval      Transfer    Bandwidth   Jitter   Lost/Total Datagrams
[ 3] 0.0- 1.0 sec 11.4 MBytes 95.4 Mb/sec 0.125 ms   7/ 8118 (0.086%)
[ 3] 1.0- 2.0 sec 11.4 MBytes 95.2 Mb/sec 0.122 ms  20/ 8117 (0.25%)
[ 3] 2.0- 3.0 sec 11.4 MBytes 95.4 Mb/sec 0.119 ms  10/ 8119 (0.12%)
[ 3] 3.0- 4.0 sec 11.4 MBytes 95.3 Mb/sec 0.121 ms  17/ 8118 (0.21%)
```

Ten přijal nepotvrzená data přenesená protokolem UDP s rychlostí se okolo 11.9MB /s.

Z testů lze tvrdit, že automatické nastavení rychlosti portů Ethernetového přepínače se nastaví správně na nejvýšší možnou rychlost a odesílaná data odpovídají rychlosti 100 Mb/s Ethernetu.

Kapitola 6
Závěr

V této práci byl vytvořen ovladač do operačního systému Linux pro konfigurovatelný Ethernetový přepínač dodaný firmou Retia a.s., který splňuje průmyslové teplotní rozsahy.

Na modul Colibri T20, který má integrované hostitelské USB zařízení a splňuje průmyslové teplotní rozsahy, byla naportována verze 3.19 jádra operačního systému Linux. Dálé byl implementován ovladač pro konfigurovatelný přepínač do této verze operačního systému, který umožňuje inicializaci zařízení za běhu a umí reprezentovat každý port přepínače jako virtuální rozhraní viditelné v uživatelském prostoru operačního systému Linux.

Tento ovladač je složen z úpravy stávajícího ovladače pro kontrolér Asix AX88772b, který byl rozšířen o podporu ovládání rozhraní RMII a podporu připojení k subsystému DSA. Další úpravou prošel subsystém DSA, který byl doplněn o možnost připojení a uvolnění zařízení za běhu. Nakonec byl naimplementován ovladač pro čip Ethernetového přepínače Marvell 88E6065, který ovládá funkčnost obvodu Ethernetového přepínače.

Implementace vytvořeného ovladače byla provedena na verzi 3.19 Linuxového jádra a změny v Linuxovém jádře byly zaslány k okomentování Linuxovou komunitou, kde bylo navrženo několik úprav a vylepšení. Avšak moje změna nebyla začlenena do jádra.

Finální verze ovladače byla testována na správnost uvolňování paměti, korektnost konfigurace portů přepínače a rychlost přenosu dat. Ve všech těchto parametrech dosáhl ovladač žádoucích výsledků.

Literatura

[1] Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman. *Linux Device Drivers, 3rd Edi tion. O'Reilly Media, Inc.*, 2005 [cit. 2013-5-5]. Dostupné z: http://lwn.net/Kernel/LDD3/.

[2] *USB documentation*, 2005[cit. 2013-5-5]. Dostupné z: http://www.usb.org/developers/docs/.

[3] Microsoft Corporation. http://technet.microsoft.com.

[4] Free Electrons. *Linux usb*, 2009[cit. 2013-5-5]. Dostupné z: http://free-electrons.com/doc/linux-usb.pdf.

[5] Ethernet, Energy Efficient. *Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and Physical Layer specifications.*, 2008 [cit. 2013-5-5], DOI: 10.1109/IEEESTD.2008.4726975. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=4726975&queryTexti%3DPArt+3+Carrier.

[6] Mike Jones, Micrel Inc. *Interfacing Fast Ethernet to Processors* [cit. 2013-5-5]. Dostupné z: http://www.micrel.com/_PDF/Ethernet/White%20Paper/Ethernet_to_processors.pdf.

[7] RMII Consortium. RMII-Specification. AMD Inc., Broadcom Corp., National Semiconductor Corp., and Texas Instruments Inc. *RMII Specification*, 1997 [cit. 2013-5-5].

[8] *Struct device*[cit. 2013-5-5]. Dostupné z: https://www.kernel.org/doc/htmldocs/device-drivers/API-struct-device.html.

[9] Benvenuti, Christian. *Understanding Linux network internals. O'Reilly Media, Inc.*, 2006 [cit. 2013-5-5].

[10] Lennert Buytenhek. *net: Distributed Switch Architecture protocol support.*, 2008 [cit. 2013-5-5]. Dostupné z: https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=91da11f870f00a3322b81c73042291d7f0be5a17.

[11] Device tree , [cit. 2013-5-5]. Dostupné z: http://www.devicetree.org/Main_Page.

[12] Patrick Mochel *The Sysfs Filesystem* Linux Symposium, 2005 [cit. 2013-5-5]. Dostupné z:

https://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf.

[13] Toradex. *NVIDIA Tegra 2 Computer on Module - Colibri T20*, 2012. Dostupné z: www.toradex.com/computer-on-modules/colibri-arm-family/nvidia-tegra-2

[14] NVIDIA Corporation. *NVIDIA Tegra 2*, 2011. Dostupné z: www.nvidia.com/content/PDF/tegra_white_papers/Bringing_High-End_Graphics_to_Handheld_Devices.pdf.

[15] Notebookcheck.net. *NVIDIA Tegra 2 benchmark*, 2013 [cit. 2013-5-5]. Dostupné z: http://www.notebookcheck.net/NVIDIA-Tegra-250-SoC.54654.0.html.

[16] Asix Electronics Corporation. [cit. 2013-5-5]. Dostupné z: http://www.asix.com.tw/download.php?sub=guidedetail&PItemID=86.

[17] Cisco. *QoS Policing*, 2008. Dostupné z: www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-policing/22833-qos-faq.html.

[18] Marvell. *Link Street 88E6060*, 2008. Dostupné z: www.marvell.com/switching/assets/marvell_linkstreet_88E6060_datasheet.pdf.

[19] *Colibri T20 upstream Linux kernel*, 2014[cit. 2013-5-5]. Dostupné z: http://falstaff.agner.ch/2014/03/16/colibri-t20-upstream-linux-kernel/.

[20] *Malling list archive*, [cit. 2013-5-5]. Dostupné z: http://marc.info/?l=linux-netdev&m=142972103930390&w=2.

Příloha A
Zadání práce

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Jan Kaisrlík**

Studijní program: Otevřená informatika
Obor: Počítačové inženýrství

Název tématu: **Integrace Ethernetového přepínače do embedded zařízení s OS Linux**

Pokyny pro vypracování:

1. Nastudujte síťový subsystém OS Linux, zaměřte se na zařízení typu Distributed Switch Architecture. Dále se seznamte s procesorovým modulem Toradex Colibri T20 a s obvody ASIX AX88772B, Marvel 88E6083.
2. Naportujte linuxovou distribuci OpenWrt na modul Colibri T20.
3. Implementujte chybějící kód potřebný pro zprovoznění Ethernetového přepínače Marvel 88E6083 připojeného přes USB zařízení ASIX AX88772B. Pokuste se výsledný kód zahrnout do hlavní větve Linuxu.
4. Otestujte implementované řešení zejména s ohledem na propustnost sítě a stabilitu.
5. Vše důkladně zdokumentujte.

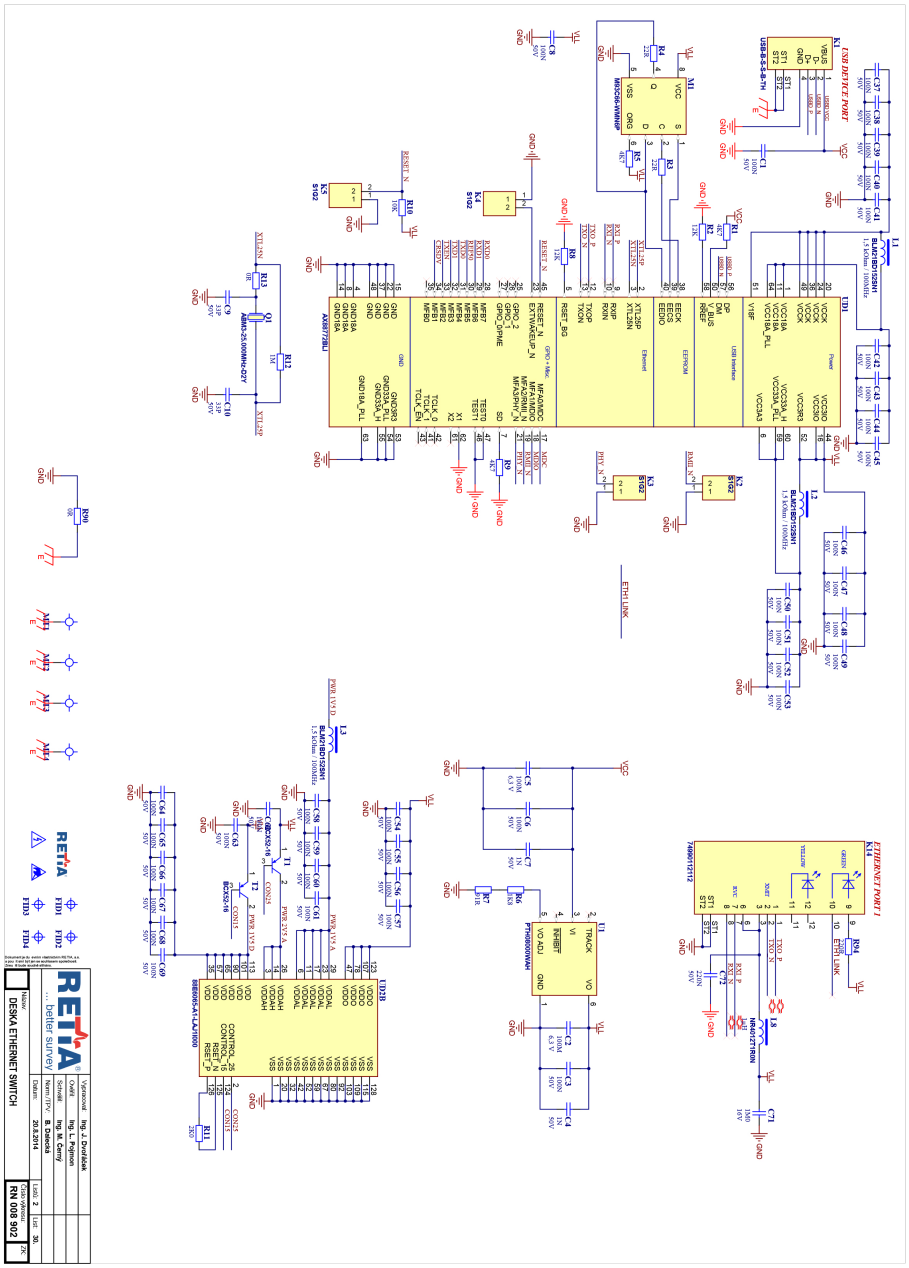
Seznam odborné literatury:

Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman. 2005. Linux Device Drivers, 3rd Edition. O'Reilly Media, Inc.
https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=91da11f870f00a3322b81c73042291d7f0be5a17

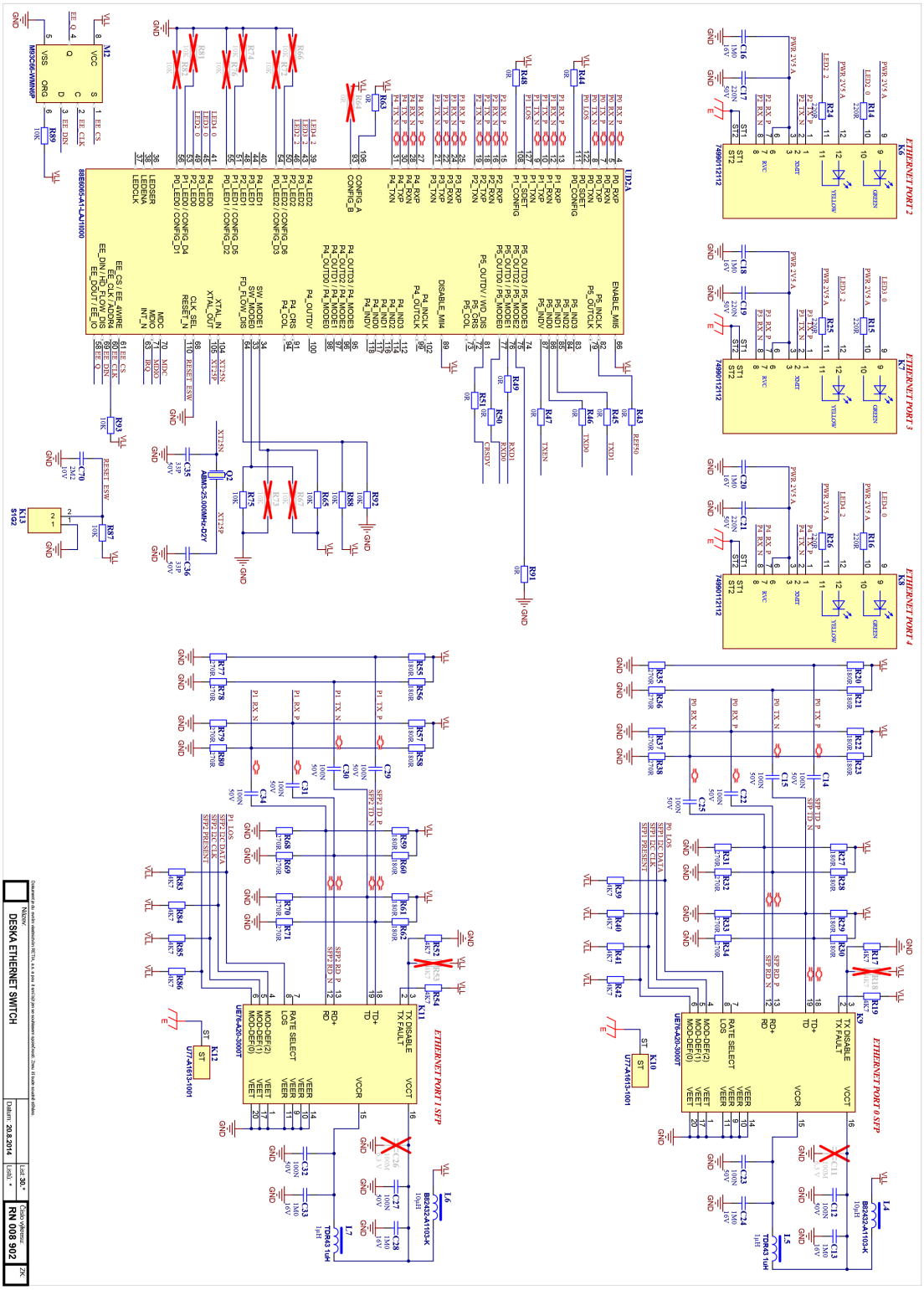
Vedoucí: Ing. Michal Sojka, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

Příloha B
Přílohy



Obrázek B.1. Schéma zapojení kontroléru Asix AX88772b. Zapojení poskitnuto firmou Retia a.s.



Obrázek B.2. Schéma zapojení přepínače Marvell 88e6065. Zapojení poskitnuto firmou Retia a.s.

Bit(s)	Name	Description	R/W[a]
0.15	Reset	1 = PHY reset 0 = normal operation	R/W SC
0.14	Loopback	1 = enable loopback mode 0 = disable loopback mode	R/W
0.13	Speed Selection (LSB)	0.6 0.13 1 1 = Reserved 1 0 = 1000 Mb/s 0 1 = 100 Mb/s 0 0 = 10 Mb/s	R/W
0.12	Auto-Negotiation Enable	1 = enable Auto-Negotiation process 0 = disable Auto-Negotiation process	R/W
0.11	Power Down	1 = power down 0 = normal operation[b]	R/W
0.10	Isolate	1 = electrically Isolate PHY from MII or GMII 0 = normal operation[b]	R/W
0.9	Restart Auto-Negotiation	1 = restart Auto-Negotiation process 0 = normal operation	R/W SC
0.8	Duplex Mode	1 = full duplex 0 = half duplex	R/W
0.7	Collision Test	1 = enable COL signal test 0 = disable COL signal test	R/W
0.6	Speed Selection (MSB)	0.6 0.13 1 1 = Reserved 1 0 = 1000 Mb/s 0 1 = 100 Mb/s 0 0 = 10 Mb/s	R/W
0.5	Unidirectional enable	When bit 0.12 is one or bit 0.8 is zero, this bit is ignored. When bit 0.12 is zero and bit 0.8 is one: 1 = Enable transmit from media independent interface regardless of whether the PHY has determined that a valid link has been established 0 = Enable transmit from media independent interface only when the PHY has determined that a valid link has been established	R/W
0.4:0	Reserved	Write as 0, ignore on read	R/W

[a]R/W = Read/Write, SC = Self-clearing.
[b]For normal operation, both 0.10 and 0.11 must be cleared to zero; see 22.2.4.1.5.

Obrázek B.3. Kontrolní registr STA rozhraní. Obrázek je převzat z [5].

Bit(s)	Name	Description	R/W[a]
1.15	100BASE-T4	1 = PHY able to perform 100BASE-T4 0 = PHY not able to perform 100BASE-T4	RO
1.14	100BASE-X Full Duplex	1 = PHY able to perform full duplex 100BASE-X 0 = PHY not able to perform full duplex 100BASE-X	RO
1.13	100BASE-X Half Duplex	1 = PHY able to perform half duplex 100BASE-X 0 = PHY not able to perform half duplex 100BASE-X	RO
1.12	10 Mb/s Full Duplex	1 = PHY able to operate at 10 Mb/s in full duplex mode 0 = PHY not able to operate at 10 Mb/s in full duplex mode	RO
1.11	10 Mb/s Half Duplex	1 = PHY able to operate at 10 Mb/s in half duplex mode 0 = PHY not able to operate at 10 Mb/s in half duplex mode	RO
1.10	100BASE-T2 Full Duplex	1 = PHY able to perform full duplex 100BASE-T2 0 = PHY not able to perform full duplex 100BASE-T2	RO
1.9	100BASE-T2 Half Duplex	1 = PHY able to perform half duplex 100BASE-T2 0 = PHY not able to perform half duplex 100BASE-T2	RO
1.8	Extended Status	1 = Extended status information in Register 15 0 = No extended status information in Register 15	RO
1.7	Unidirectional ability	1 = PHY able to transmit from media independent interface regardless of whether the PHY has determined that a valid link has been established 0 = PHY able to transmit from media independent interface only when the PHY has determined that a valid link has been established	RO
1.6	MF Preamble Suppression	1 = PHY will accept management frames with preamble suppressed. 0 = PHY will not accept management frames with preamble suppressed.	RO
1.5	Auto-Negotiation Complete	1 = Auto-Negotiation process completed 0 = Auto-Negotiation process not completed	RO
1.4	Remote Fault	1 = remote fault condition detected 0 = no remote fault condition detected	RO/ LH
1.3	Auto-Negotiation Ability	1 = PHY is able to perform Auto-Negotiation 0 = PHY is not able to perform Auto-Negotiation	RO
1.2	Link Status	1 = link is up 0 = link is down	RO/ LL
1.1	Jabber Detect	1 = jabber condition detected 0 = no jabber condition detected	RO/ LH
1.0	Extended Capability	1 = extended register capabilities 0 = basic register set capabilities only	RO

[a]RO = Read only, LL = Latching low, LH = Latching high

Obrázek B.4. Stavový registr STA rozhraní. Obrázek je převzat z [5].

Příloha C
Obsad přiloženého CD

switch_patch_3-19.patch	Patch ovladače Ethernetového přepínače pro verzi jádra 3.19.
.config_kernel	Konfigururace Linuxového jádra.
.config_openwrt	Konfigurace distribuce OpenWrt.
Adresář ml:	
000*	Patche poslané do hlavní větve Linuxového jádra.
Adresář docs:	
Jan_Kaisrlik_dp.pdf	Soubor s textem diplomové práce.
board.pdf	Schéma desky ethernetové přepínače.
board2.pdf	Schéma desky ethernetové přepínače.

Requests for correction