

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE  
FAKULTA ELEKTROTECHNICKÁ

# Bakalářská práce

---

Ovládání experimentu řídicím modulem s  
rozhraním Ethernet



## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Radek Řípa**  
Obor: **Kybernetika a měření**  
Název tématu česky: **Ovládání experimentu řídicím modulem s rozhraním Ethernet**  
Název tématu anglicky: **Controlling of Experiment by Ethernet Based Control Unit**


### Pokyny pro vypracování:

Pro ovládání experimentu řídicí jednotkou s rozhraním Ethernet vyberte vhodnou implementaci TCP/IP stacku a upravte ji pro procesor STM32F107 v modulu STM32-Comstick. Navrhněte metodu ovládání osvětlovacích LED a polohovacího mechanismu s krokovým motorkem v experimentu pomocí modulu s STM32. Navrhněte příslušné elektrické obvody a vytvořte potřebné programové vybavení

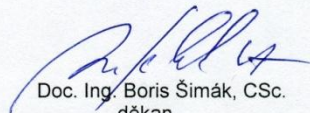
### Seznam odborné literatury:

- [1] Yiu J The definitive Guide to the ARM Cortex- M3. Elsevier, 2007, ISBN: 978-0-7506-8534-4
- [2] RM0008 - Reference manual STM32F101xx and STM32F103xx. 2008, STMicroelectronics
- [3] Cortex-M3™, Technical Reference Manual. Publ. ARM DDI 0337B, 2006, ARM Limited.

Vedoucí bakalářské práce: Ing. Jan Fischer, CSc.  
Datum zadání bakalářské práce: 23. listopadu 2009  
Platnost zadání do<sup>1</sup> 4. února 2011

  
Prof. Ing. Pavel Ripka, CSc.  
vedoucí katedry



  
Doc. Ing. Boris Šimák, CSc.  
děkan

V Praze dne 23. 11. 2009

<sup>1</sup> Platnost zadání je omezena na dobu dvou následujících semestrů.

# Obsah

Seznam obrázků .....	4
Seznam babulek.....	5
Anotace.....	6
Poděkování .....	7
Čestné prohlášení autora práce.....	8
1. Úvod .....	9
2. Komunikace.....	10
2.1 USART .....	11
2.2 USB .....	11
2.3 Ethernet .....	12
3. Počítačová síť .....	12
3.1 Vrstva síťového rozhraní.....	13
3.2 Síťová vrstva .....	14
3.2.1 IP protokol.....	14
3.2.2 ARP protokol.....	14
3.3 Transportní vrstva.....	15
3.4 Aplikační vrstva .....	16
4. Ethernet na STM32F107 .....	16
4.1 MII.....	18
4.2 RMII .....	19
4.3 Fyzické vrstvy .....	20
4.3.1 STE100P.....	20
4.3.2 ST802RT1 .....	21
5. Výběr TCP-IP stacku.....	22
5.1 Implementace lwIP stacku.....	23
6. Vývojové kity .....	24
6.1 comStick.....	24
6.1.1 Programování STM32comStick .....	25
6.1.2 Nahrání hex souboru do STM32comStick .....	26
6.2 STEEVAL-PCC010V1 .....	27
6.2.1 Programování STEEVAL-PCC010V1 .....	27
7. Blokové schéma zapojení celého experimentu.....	29
7.1 Ovládání LED diod .....	30

7.2 Ovládání krokového motoru.....	31
8. Program .....	33
9. Popis programu pro komunikaci s experimentem .....	38
10. Zhodnocení výsledků .....	39
Použitá literatura.....	40
Obsah příloženého CD .....	41

## Seznam obrázků

Obrázek 1: Vnitřní uspořádání uP STM32F107 .....	10
Obrázek 2: Schéma práce TCP/IP protokolu .....	13
Obrázek 3: Ukázka zapouzdření dat v síti TCP/IP .....	16
Obrázek 4: Schéma uspořádání ethernetu v uP STM32F107 .....	17
Obrázek 5: Znázornění zapisování dat do PHY .....	18
Obrázek 6: Znázornění průběhu čtení dat z PHY .....	18
Obrázek 7: Znázornění vodičů nutných při zapojení v režimu MII .....	19
Obrázek 8: Znázornění vodičů nutných při zapojení v režimu RMII .....	19
Obrázek 9: Vnitřní uspořádání STE100P.....	20
Obrázek 10: Vnitřní uspořádání ST802RT1 .....	21
Obrázek 11: Uspořádání lwIP stacku .....	23
Obrázek 12: STM32comStick .....	25
Obrázek 13: Vývojové prostředí HiTOP .....	25
Obrázek 14: Vývojový kit SEEVAL-PCC010V1 .....	27
Obrázek 15: Uspořádání STEEVAL-PCC010V1 .....	27
Obrázek 16: ST-Link.....	28
Obrázek 17: Software ST-Link Utility pro nahrávání programů do uP přes ST-Link.....	29
Obrázek 18: Blokové schéma zapojení experimentu .....	29
Obrázek 19: Znázornění konektoru u SEEVAL-PC010V1 který umožňuje připojení externích periférií .....	30
Obrázek 20: Posuvný registr 74HCT595 .....	30
Obrázek 21: Schéma zapojení STM32F107 a posuvného registru 74HCT595 .....	31
Obrázek 22: Schéma zapojení obvodu L6208 pro ovládání krokového motoru.....	32

Obrázek 23: Schéma zapojení obvodu L6208 k uP STM32F107 .....	33
Obrázek 24: Inicializace programu .....	34
Obrázek 25: Práce lwIP stacku.....	36
Obrázek 26: Práce programu pro ovládání LED diod a krokového motoru .....	37
Obrázek 27: Obsluha přerušení čítače TIM2 .....	38
Obrázek 28: Program pro ovládání experimentu přes PC.....	39

## **Seznam babulek**

Tabulka 1: Formát ethernetového rámce.....	14
Tabulka 2: Formát IP datagramu.....	14
Tabulka 3: Formát TCP/IP hlavičky .....	15
Tabulka 4: Formát zpráv pro nastavení PHY .....	17

## **Anotace**

Práce se zabývá implementací TCP-IP stacku do mikroprocesoru pro obsluhu obvodu ethernet, napsáním programu, který bude využívat TCP-IP stacku pro komunikaci s PC, přes které se bude ovládat krokový motor a LED diody. Dále se zabývá sestavením obvodů pro ovládání krokového motoru a LED diod a jejich připojením k mikroprocesoru STM32F107.

## **Annotation**

Work deals with implementation of TCP-IP stack to the microprocessor for operating the circuit ethernet, typing program that will use TCP-IP stack for communication with PC, through which it will be controlled the stepper motor and LEDs. It also deals with assembling of circuits for controlling the stepper motor, and LEDs and their connection to STM32F107 microprocessor.

## **Poděkování**

Na tomto místě bych rád poděkoval Ing. Jan Fischerovi za vedení bakalářské práce a za cenné rady a připomínky

## **Čestné prohlášení autora práce**

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 .....

.....

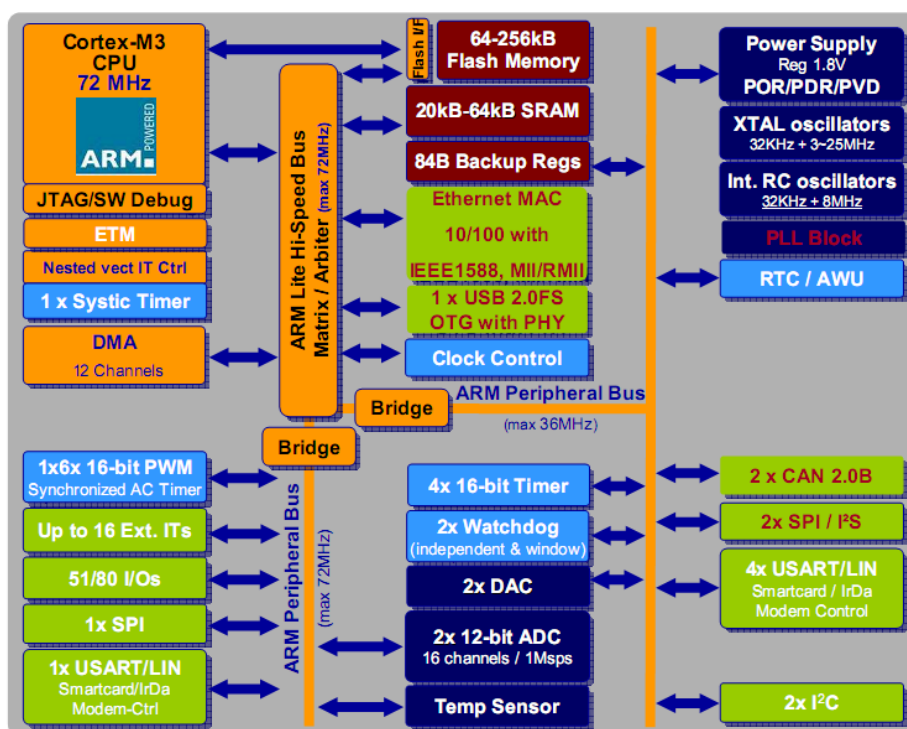
Podpis autora práce

## 1. Úvod

Naším úkolem je vytvořit ovládání krokového motoru a led diod. Pro ovládání těchto zařízení postačí jakýkoliv obyčejný uP. Jestliže však požadujeme, aby ovládání bylo vzdálené, a zároveň aby byla možnost řídit více zařízení současně, stává se z toho větší problém. Tato práce má za úkol vyzkoušet pro vzdálené ovládání ethernet, který je dnes používám k tvorbě sítí a je obsažen již v každém novém PC. Pro vzdálené řízení je tedy vhodné rozhraní ethernet, které umožní námi vzdálené řízení experimentu, a navíc je možné ovládat více jednotek současně. Další výhodou je že pro ovládání je možné využít již existující počítačové sítě, aniž by bylo nutné nějak je upravovat. Je tedy možné daný experiment zapojit kdekoliv v naší počítačové síti a bude možné aby k jeho ovládání byl využit počítač, který je na tuto síť napojen.

## 2. Komunikace

Moderní mikroprocesory(dále jen uP) získávají stále více periférií, které jim umožňují komunikovat s okolím pomocí standardizovaných protokolů. V této práci se budeme zabývat konkrétním uP od výrobce STMicroelectronics(dále jen STM). Jedná se o 32bitový uP STM32F107 (viz [5]).



Obrázek 1: Vnitřní uspořádání uP STM32F107

uP obsahuje několik periférií(obr.1), které dnes obsahuje každý osobní počítač(PC) a pomocí nichž může být uskutečněn přenos dat mezi PC a uP, což je i úkolem této práce. Pro komunikaci s PC je tedy možné využít jeden z obvodů, který obsahuje uP a to:

1. USART
2. USB
3. Ethernet

## 2.1 USART

Připojení převodníku na RS232 umožňuje připojení k PC. Je to již poměrně zastaralé rozhraní. Přenosová rychlost je dnes velice podprůměrná a jelikož se již toto rozhraní u moderních PC téměř nevyskytuje, je vytlačováno rozhraním USB. Výhoda může být délka kabelu, která je okolo 15metrů a může se měnit s rychlostí modulace. Další výhodou je že pomocí tohoto rozhraní může být měněn program uP, čímž odpadá nutnost použití dalšího hardware pro programování.

### Základní parametry rozhraní RS 232

- Komunikační rychlost až 20kb/s
- Komunikační vzdálenost 15m
- Pouze přenos mezi dvěma vysílači/přijímači

## 2.2 USB

Novější rozhraní oproti USARTu dosahuje větších přenosových rychlostí a možnost připojení více zařízení na jednu sběrnici. Nevýhodou je malá komunikační vzdálenost. Výhodou je že pro připojení USB k uP není nutný další obvod. Podle specifikací výrobce může být toto rozhraní použito pro nahrání programu do uP.

### Základní parametry USB

- Komunikační rychlost do 480Mbit/s
- Komunikační vzdálenost do 5m
- Možnost připojení více zařízení
- Rozhraní obsahuje 5V napájení
- Lze připojit až 127 zařízení pomocí jednoho typu konektoru.

## 2.3 Ethernet

Je to protokol pro komunikaci, který je dnes hojně využíván v lokálních sítích díky své jednoduchosti. V modelu TCP/IP realizuje fyzickou vrstvu síťového rozhraní. Pro přenos dat je zde možnost použití více způsobů. Nejstarší je použití koaxiálního kabelu, který umožnil komunikaci až na vzdálenost 500m. Rozšířenější je dnes použití kroucené dvojlinky pro přenos dat na vzdálenost 100m. Poslední přenosovou cestou definovanou pro ethernet je optické vlákno, umožňující přenos dat na vzdálenost kilometrů. Rychlost přenosu, kterou podporuje uP STM32F107 je 10MB/s a 100MB/s.

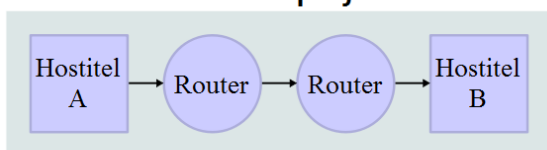
### Základní parametry Ethernet

- Přenosová rychlost 10MB/s,100MB/s
- Komunikační vzdálenost do 100m (kroucená dvojlinka)
- Možnost připojení až 127 zařízení
- Galvanické oddělení

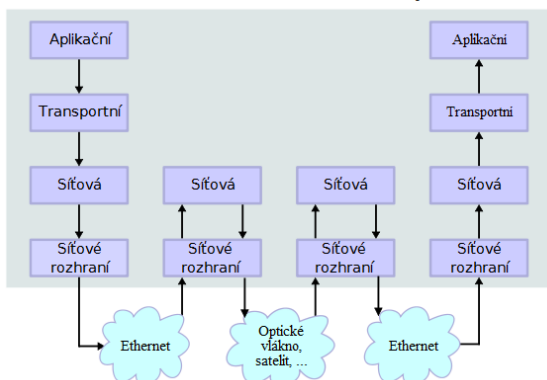
## 3. Počítačová síť

Počítačová síť podléhá standardu OSI, který definuje, jakým způsobem jsou přenášena data. OSI model obsahuje 7 vrstev, z nichž ethernet zajišťuje poslední dvě vrstvy. My se však budeme zabývat komunikačním protokolem TCP/IP(obr. 2), který na rozdíl od OSI modelu obsahuje pouze 4vrstvy. Protokol rozděluje síťovou komunikaci do několika vrstev, kdy nižší vrstvy poskytují služby vyšším vrstvám. Architektura TCP/IP je tvořena ze čtyř vrstev: aplikační vrstvy, transportní vrstvy,síťové vrstvy a vrstvy síťového rozhraní.

## Síťová spojení



## Architektura TCP/IP



Obrázek 2: Schéma práce TCP/IP protokolu

### 3.1 Vrstva síťového rozhraní

Vrstva zajišťuje základní přenos dat mezi uzly sítě, v naší aplikaci je použit protokol ethernet, který je definován standardem IEEE802.3. Ethernet používá hvězdicovou topologii a pro přístup k přenosovému médiu je použita metoda CSMA/CD viz. [14]. Jako přenosové médium může být použit koaxiální kabel, kroucená dvojlinka nebo optické vlákno. V našem případě bude využito kroucené dvojlinky. Jednotlivé stanice, které jsou připojeny, mají jedinečnou rozpoznávací adresu (MAC adresu). Jestliže stanice přijme paket jehož adresa neodpovídá adrese zařízení, paket zahodí. Stanice vždy posílají rámec definovaných parametrů viz. tabulka 1. V naší úloze vrstvu síťového rozhraní zajistí uP STM32F107 a fyzická vrstva. Fyzická vrstva zajišťuje vlastní přenos dat po kroucené dvojlince. Fyzická vrstva komunikuje s uP STM32F107 pomocí standardizovaného rozhraní MII/RMII.

**Tabulka 1: Formát ethernetového rámce**

Ethernetový rámec

Preamble	SFD	MAC cíle	MAC zdroje	Typ/dálka	Data a výplň	CRC32	Mezera mezi rámci
7bajtů 10101010	1bajt 10101011	6bajtů	6bajtů	2bajty	46-1500bajtů	4bajty	12bajty

### 3.2 Síťová vrstva

Síťová vrstva zajišťuje adresování pomocí IP adres, směrování a předávání datagramů. V naší úloze tyto operace jsou prováděny pomocí protokolů IP a ARP. Tyto protokoly implementuje do našeho uP TCP/IP stack, který budeme využívat.

#### 3.2.1 IP protokol

**Tabulka 2: Formát IP datagramu**

Formát IP datagramu

Bajty	0		1	2	3
Bajt 0 až 3	Verze	Délka hlavičky	Typ služby	Celková délka	
Bajt 4 až 7	Identifikace			Příznaky	Offset fragmentu
Bajt 8 až 11	TTL		Protokol	Kontrolní součet hlavičky	
Bajt 12 až 15	Adresa odesílatele				
Bajt 16 až 19	Adresa cíle				
Bajt 20 až 23	Volby			Výplň	
...	Data				

Protokol IP využívá své čtvrté verze pro komunikaci. Je to protokol přenášející data bez záruky. Může dojít ke ztrátě paketu, či vícenásobnému přenosu. Vyloučení těchto nežádoucích vlivů je na vyšší vrstvě. Formát diagramu ukazuje tabulka 2.

#### 3.2.2 ARP protokol

Address Resolution Protocol (ARP). Tento protokol se využívá k získání MAC adresy stanice v síti z její IP adresy. Tento protokol se použije, jestliže je třeba poslat data jiné stanici, která leží ve stejné síti jako odesílatel, ale ten zná pouze IP adresu a pro poslání ethernetového rámce je tedy zapotřebí zjistit MAC adresu. Stanice tedy vyšle ARP dotaz(ARP request),

který obsahuje hledanou IP adresu a IP adresu a MAC adresu odesílatele, všem okolním stanicím. Ostatní stanice tuto zprávu přijmou a údaje o odesílateli si uloží do ARP cache, aby je mohli v budoucnu použít a nemuseli se zdržovat hledáním majitele IP adresy. Jakmile hledaná stanice obdrží zprávu, pak odešle žadateli odpověď (ARP reply), která obsahuje hledanou IP adresu a MAC adresu.

### 3.3 Transportní vrstva

Tato vrstva zajišťuje správný přenos dat. Pro naši aplikaci bude použit protokol TCP. Tento protokol ručí za správnost přenesených dat a příjem či posílání dat ve správném pořadí. Protokol také rozlišuje data patřící jiným aplikacím, které souběžně pracují v jednom zařízení.

**Tabulka 3: Formát TCP/IP hlavičky**

TCP hlavička

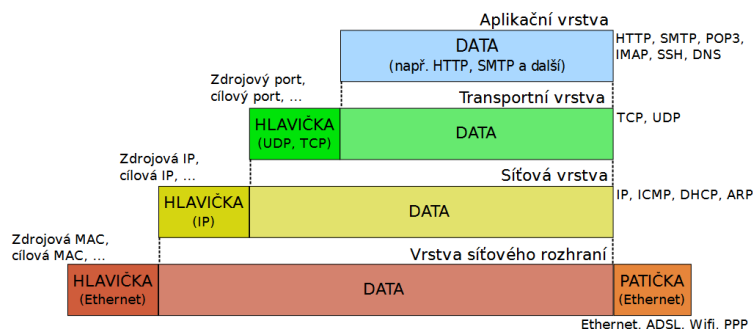
Bity	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0	zdrojový port																cílový port																
32	číslo sekvence																																
64	potvrzený bajt																																
96	offset dat				rezervováno				příznaky				okénko																				
128	kontrolní součet																Urgent Pointer																
160	volby(volitelné)																																
192	volby(pokračování)																								výplň(do 32)								
224	data																																

TCP přiřadí každému odchozímu paketu pořadové číslo, aby mohl TCP protokol na straně příjemce sestavit pakety ve správném pořadí. Pokud mu paket nepřijde do určené doby, ohlásí jeho ztrátu. Kdyby kontrolní součet paketu byl chybný, ohlásí to odesílateli a ten mu paket pošle znovu. Formát TCP zprávy v tabulce 3. Tento protokol stejně jako protokol IP a ARP budou implementovány pomocí TCP/IP stacku.

### 3.4 Aplikační vrstva

Bude realizována vlastním programem uloženým v uP, který bude využívat služeb TCP/IP stacku. Přímou bude využívat protokol TCP pro přenos dat.

#### ZAPOUZDŘENÍ DAT V SÍTI TCP/IP

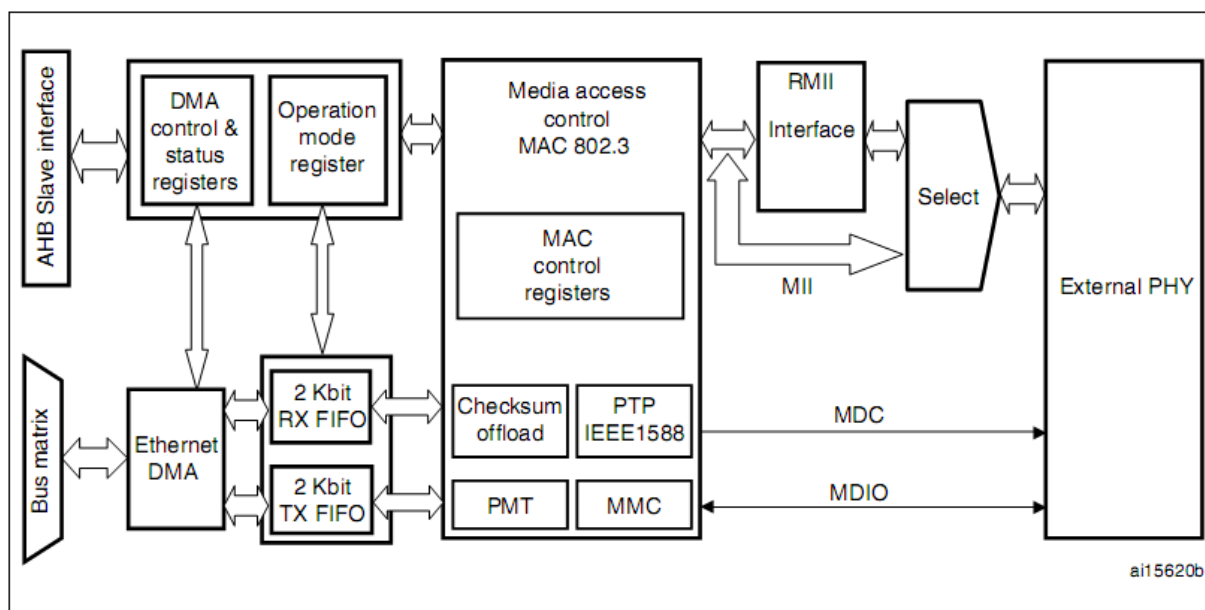


Obrázek 3: Ukázka zapouzdření dat v síti TCP/IP

## 4. Ethernet na STM32F107

Rozhraní ethernet umístěné na uP([1] str. 845) vyhovuje normě IEEE802.3, avšak pro možnost připojení na síť je potřeba ještě fyzická vrstva, která obstará příjem a vysílání dat na sběrnici. Pro přenos dat mezi PHY a uP definuje norma rozhraní MII, a jeho redukovanou verzi. RMII. Uspořádání ethernetu je na obr. 4.

Ethernet lze tedy rozdělit na několik částí. Nejdůležitější součástí je MAC obvod, který vyhovuje standardu IEEE802.3 a stará se o zajištění komunikace se sítí. Obvod přijímá a vysílá data tak, že odpovídají formátu ethernetového rámce. Data jsou pomocí rozhraní MII nebo RMII předávána PHY, která se postrá o jejich vyslání. Data, jež byla přijata nebo budou vyslána jsou ukládána do dvou FIFO registrů sloužících jako vyrovnávací paměť. K obvodu ethernet náleží vlastní DMA registr, který se stará o přenášení dat z FIFO paměti do určeného paměťového prostoru uP, nebo naopak o zápis dat z paměti do FIFO. Popis kontrolních registrů obvodu ethernet lze nalézt v [1] na stránce 845. Pro ovládání registrů ethernetu je od firmy ST dispozici knihovna STM32\_eth. S touto knihovnou poté pracuje implementovaný TCP/IP stack.



Obrázek 4: Schéma uspořádání ethernetu v uP STM32F107

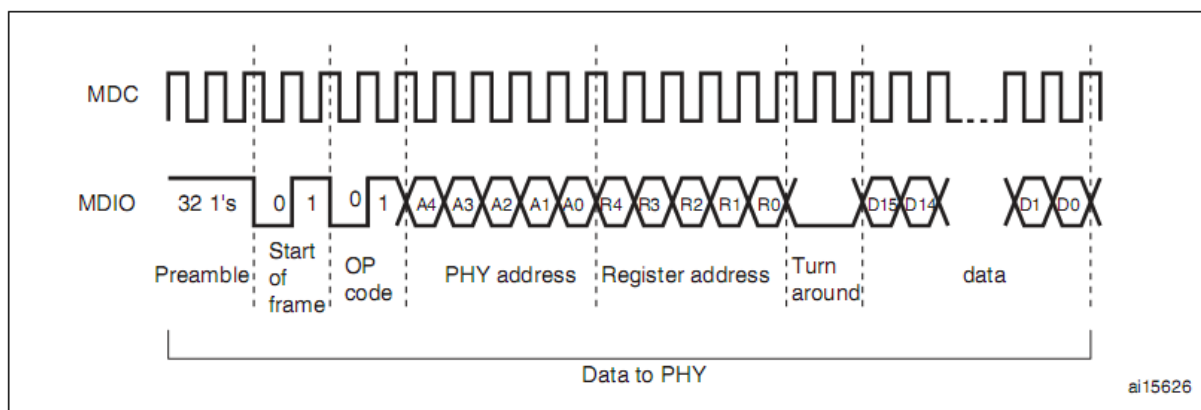
Jelikož PHY má vlastnosti které je třeba nastavit obsahuje MAC uP sériové rozhraní pro komunikaci s PHY, tzv. station management interface (SMI). Využívá dvou vodičů z nichž jeden slouží jako hodinový signál MDC a druhý pro vstup a výstup dat. Komunikace pomocí SMI probíhá po přesně definovaných rámcích(tabulka 4). Složení rámce je v tab.

Tabulka 4: Formát zpráv pro nastavení PHY

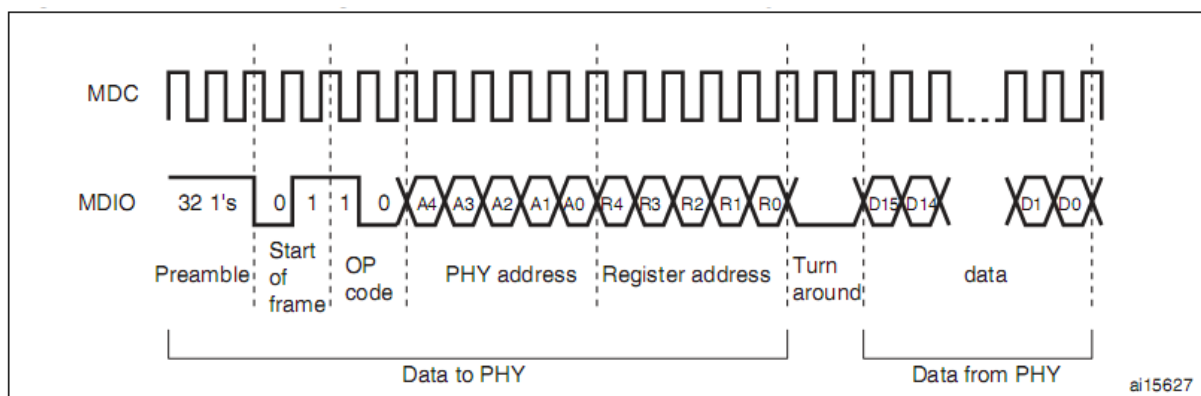
Formát zpráv pro komunikaci s PHY								
	Preamble(32bits)	Start	Operation	PADDR	RADDR	TA	Data(16bits)	Idle
Read	1....1	1	1	ppppp	rrrrr	Z0	dddddddddddddddd	Z
Write	1....1	1	1	ppppp	rrrrr	10	dddddddddddddddd	Z

Zápis do PHY proběhne nastavením datového registru MII a adresy kam se mají daná data zapsat(obr. 5). Poté stačí nastavit BSY bit a čekat na provedení operace. Po dokončení přenosu uP resetuje BSY bit.

Při čtení z PHY pouze určíme požadovanou adresu, nastavíme BSY bit a čekáme až bude BSY bit resetován. Poté můžeme přečíst požadovaná data v datovém registru MII.obr zápis a čtení z PHY(obr. 6).



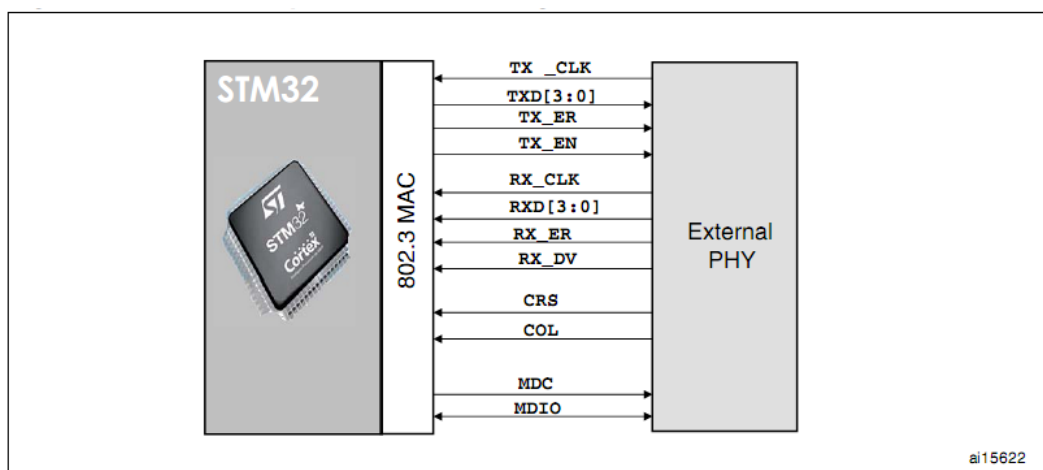
Obrázek 5: Znáznornění zapisování dat do PHY



Obrázek 6: Znáznornění průběhu čtení dat z PHY

## 4.1 MII

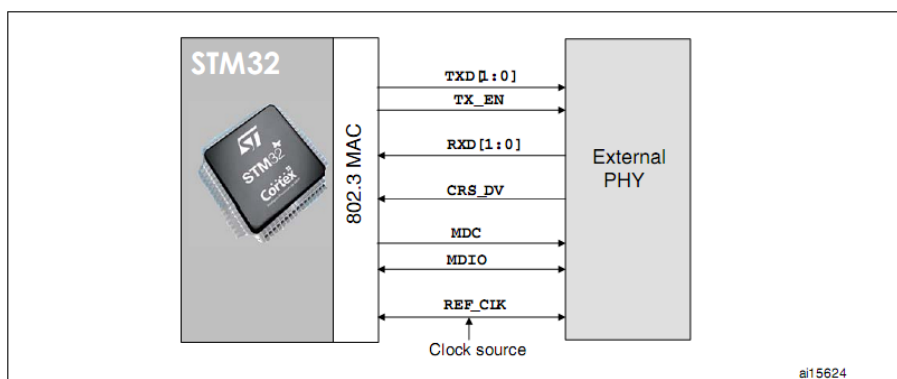
Definuje způsob přenosu dat mezi MAC a PHY (obr. 7). K přenášení dat je využito 16 vodičů. Čtyři vodiče pro posílaná data (MII\_TXD), hodinový signál pro posílaná data (MII\_TX\_CLK), signál žádající PHY o povolení přenosu dat (MII\_TX\_EN), signál oznamující chybu při posílání dat (MII\_TX\_ER), čtyři vodiče pro příjem dat (MII\_RXD), hodinový signál pro přijímaná data (MII\_RX\_CLK), signál upozorňující na přijímaná data (MII\_RX\_DV), upozornění na chybu v přijímaném rámci (MII\_RX\_ER), signál upozorňující, že zařízení nejsou připravena pro příjem nebo posílání dat (MII\_CRD), signál oznamující detekci kolize (MII\_COL). Obr. 7.



Obrázek 7: Znázornění vodičů nutných při zapojení v režimu MII

## 4.2 RMII

Je redukováná MII, kdy je počet pinů potřebný k přenosu u MII redukován z 16 na 7 za pomoci 2x většího hodinového signálu (obr. 8). Další rozdíl mezi MII a RMII je ve způsobu zavedení hodinového signálu, kdy u RMII musí být uP i PHY synchronizovány stejným hodinovým signálem. To neumožňuje jednoduché přepnutí režimu z MII na RMII ale musí dojít ke změně nastavení hodinových signálů přímo na plošném spoji.



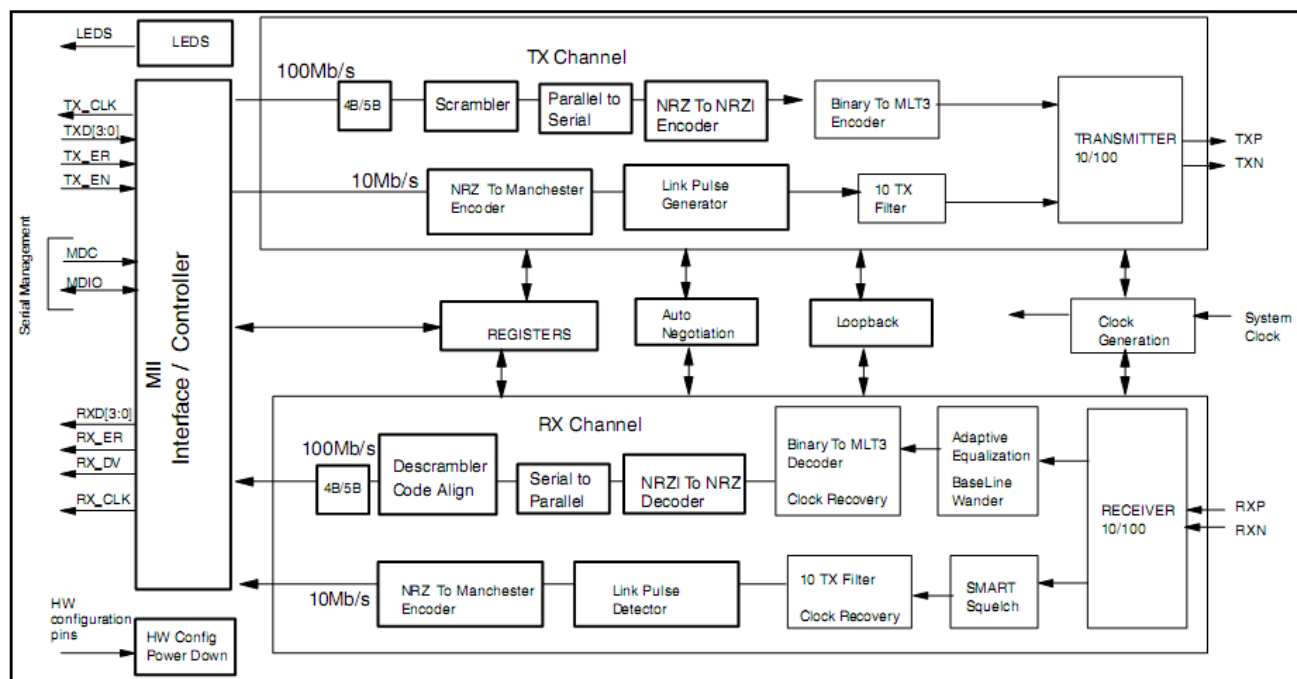
Obrázek 8: Znázornění vodičů nutných při zapojení v režimu RMII

Výběr MII nebo RMII nastavíme v programu bitem `MII_RMII_SEL` v `AFIO_MAPR` nebo při nastavování lwIP stacku.

## 4.3 Fyzické vrstvy

### 4.3.1 STE100P

Fyzická vrstva (dále jen PHY) od firmy STM (viz. [9]) je určena pro připojení ethernetu splňujícího standardy IEEE802.3 a IEEE802.3u. Pro přenos dat mezi PHY a uP je využito rozhraní MII, které se skládá z 16 vodičů. Režim RMII není u této fyzické vrstvy podporován. PHY však podporuje funkci Auto-negotiation, která slouží k nastavení přenosu na maximální možnou rychlost, již podporují dvě propojená zařízení. PHY má možnost změnit některá její nastavení, popřípadě zjistit informace o svém stavu. Nastavení některých funkcí je možno provést pomocí pinů, například zařazením pull-up či pull-down rezistorů k pinům určeným pro LED diody je možno nastavit počáteční adresu PHY. Takto se dají nastavit i tzv. multifunkční piny, které jsou označeny mf0-mf4. Pomocí nich lze například zapnout funkci Auto-negotiation. Tato nastavení jsou však hardwarového charakteru a nelze je u hotových



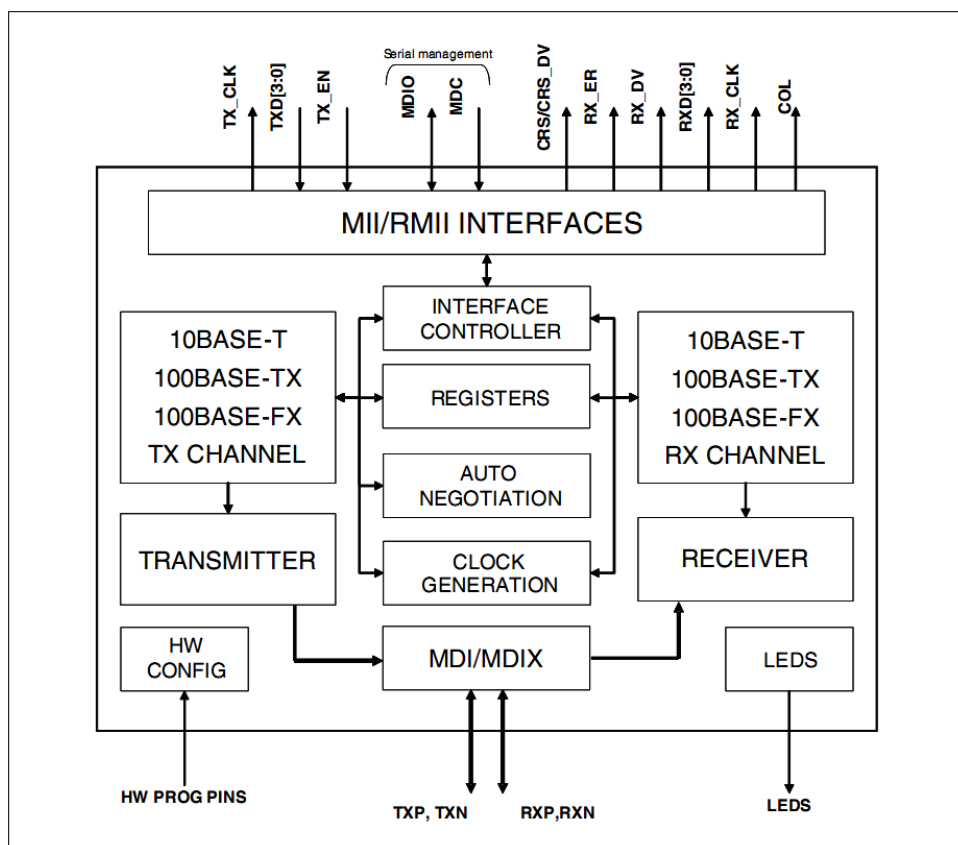
Obrázek 9: Vnitřní uspořádání STE100P

výrobků již měnit. Proto také PHY obsahuje sériové rozhraní pomocí kterého je možné změnit nastavení čipu. PHY obsahuje 11 registrů po 16 bitech (viz. [9] tabulka 4). Z toho je sedm registrů tzv. základních, které jsou definovány v MII. Díky standardizovanému rozhraní

je proto možné vyměnit jednu PHY za druhou aniž by se musel měnit program, za předpokladu, že se v PHY neprovádí nastavení v jiných než v 7 základních registrech.

#### 4.3.2 ST802RT1

Je nová fyzická vrstva od firmy STM(viz. [10]) která navazuje na předešlou PHY a rozšiřuje její možnosti. Mezi největší vylepšení této PHY patří možnost RMI rozhraní, které umožní redukovat počet vodičů nutných pro komunikaci s uP. Další funkcí která zjednodušuje práci s připojením do sítě ethernet, je funkce auto MDI-X. Dále se tato PHY liší od STE100P jiným uspořádáním registrů(viz. [10]-tabulka 8). Jelikož má tato vrstva více funkcí má logicky i více registrů pro nastavení. Avšak základní registry jsou stejné jako



Obrázek 10: Vnitřní uspořádání ST802RT1

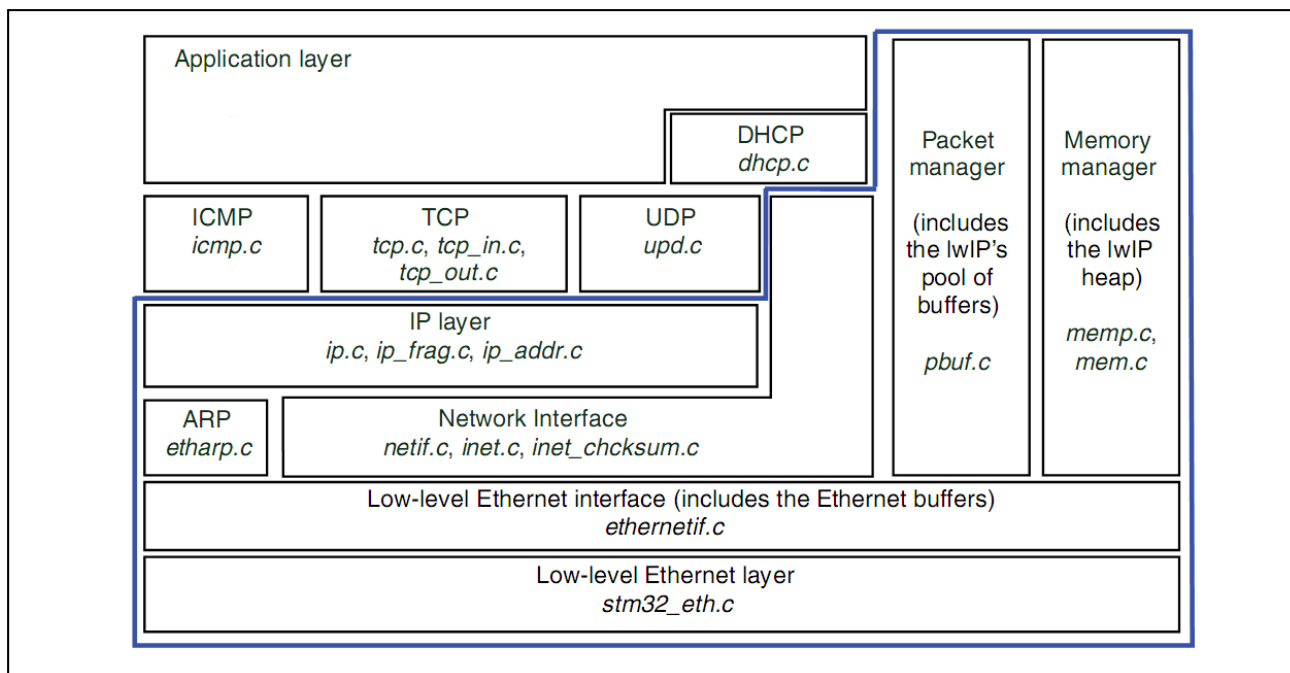
u předchozí vrstvy. První registr je kontrolní registr, který umožňuje základní nastavení PHY jako rychlost přenosu, nastavení funkce Auto-negotiation či přepnutí PHY do tzv. izolovaného stavu, kdy přestane komunikovat s uP i s vnější sítí a komunikuje pouze přes sériové rozhraní.

## 5. Výběr TCP-IP stacku

TCP-IP stack[2] zajišťuje rozhraní mezi registry uP a uživatelským programem. Je to v podstatě soubor knihoven zjednodušující práci s rozhraním ethernet. Nejdříve bylo třeba rozhodnout, který z TCP-IP bude použit pro program. STM dává možnost výběru ze tří TCP-IP stacků a to uIP, který je nejjednodušší, zabírá minimum paměti, zabere něco málo přes 32kB. Další k dispozici je TCP-IP stack NicheLite TCP/IP stack a lip stack. Oba tyto stacky jsou rozsáhlejší než uIP stack a umožňují vyšší přenosovou rychlost oproti uIP, což plyne z omezení velikosti uIP.

Jako první jsem se pokusil implementovat uIP stack, bohužel jelikož jsem s programováním uP teprve začínal a nedokázal jsem ho úspěšně implementovat. Zkusil jsem stack lwIP, který může pro budoucí potřeby nabídnout vyšší rychlost komunikace. Největší problém při implementaci tohoto stacku byl s hledáním dokumentace, jelikož se mi podařilo najít pouze minimum dokumentace stacku, největší část informací jsem zjistil z příložených ukázkových programů ukazující práci se stackem. Jako web server, klient/server aplikace, telnet, a FTP. Pro mou potřebu nejvíce vyhovoval program s ukázkou pro telnet.

Na obr. 11 je vidět jak je uspořádán lwIP stack a jeho knihovny. Pro komunikaci s MAC vrstvou STM32F107 se stará stm32\_eth.c. Nad ním je ethernetif.c pro přenos a zpracování dat s FIFO front. Network interface vytváří síťovou vrstvu, kde používá soubory z IP layer, aby mohlo být pracováno s IP protokolem a ještě ARP, jež je popsáno výše. Packet manager a Memory manager slouží pro správu paměti pro vysílaná a přijímaná data. Jelikož STM32F107 má paměť oproti PC malou, není zde možnost dynamické alokace. Proto pomocí těch to souborů si lwIP stack nejprve alokuje místo v paměti které je možno definovat. Poté sám sobě z tohoto místa přiděluje paměť, která se použije k ukládání přijatých a vysílaných dat. Nad těmito vrstvami jsou soubory zajišťující síťovou vrstvu, my použijeme protokol TCP, jelikož potřebujeme mít jistotu, že přenášená data budou určitě doručena. Nad touto vrstvou je aplikační, kterou bude tvořit náš program.



Obrázek 11: Uspořádání lwIP stacku

## 5.1 Implementace lwIP stacku

Nejprve nutné si vytvořit nový projekt v Rise 7. Dále je do něj nutné vložit všechny soubory, které slouží k fungování TCP/IP stacku. Nejprve je nutné do projektu vložit soubory, které se nacházejí v adresáři lwIP, jsou to soubory které jsou nutné pro funkci lwIP stacku: `ethernetif.c`, `inet.c`, `inet_chsum.c`, `ip_addr.c`, `ip_frag.c`, `init.c`, `mem.c`, `memp.c`, `netif.c`, `pbuf.c`, `raw.c`, `tcp.c`, `tcp_in.c`, `tcp_out.c`, `udp.c`, `etharp.c`, `err.c`. Dále soubory ze složky CMSIS, která obsahuje základní knihovny pro STM32: `core_cm3.c`, `system_stm32f10x.c`, `startup_stm32f10x_cl.s`. Je nutné vložit do projektu soubory pro obsluhu ethernetu na STM32: `misc.c`, `stm32f10x_dma.c`, `stm32f10x_flash.c`, `stm32f10x_gpio.c`, `stm32f10x_rcc.c`. Dále vložíme soubor `stm32_eth.c`, který slouží pro obsluhu ethernetu na STM32. Nakonec je nutné vložit do projektu soubory, které budou ovládat celý projekt a to soubor `netconf.c` obsahující nastavení ethernetu, soubor `stm32f107.c` sloužící k nastavení periférií uP, soubor `stm32f10x_it.c`, který ošetřuje přerušování uP a zbývá pouze soubor `main.c`, který bude volán při spuštění uP. Dále je možné vložit další soubor na který se bude odkazovat `main.c` a bude obsahovat vlastní program (v našem případě je to `komunikace.c`). Toto je seznam souborů nutných pro práci lwIP stacku, které musejí být vloženy v projektu vytvořeném v Rise 7. Adresář s těmito soubory a ukázkovým příkladem lwIP stacku je možné stáhnout na stránkách

STM. Dále je nutné v projectu říct kde jsou hlavičkové „.h“ soubory k vloženým „.c“ souborům. To provedeme, když pravým tlačítkem klikneme na projekt, zvolíme vlastnosti a v záložce Directories vyplníme pole Include Directories. Dále ještě v záložce *GCC compiler/Defines* nadefinujeme `USE_STDPERIPH_DRIVER`; `STM32F10X_CL` aby byli správně includovány soubory. Poslední věcí, kterou je třeba upravit je hlavičkový soubor `stm32_eth.h`, jelikož tento soubor je nutný pro práci s ethernetem a PHY. Bohužel však neobsahoval nastavení pro STE100P a ST802RT1. Proto se do místa označeného `@defgroup PHY_status_register` vloží:

```
/** @brief For STE100P */
#define PHY_SR                17    /*!< Tranceiver Status Register */
/** @brief For STE100P */
#define PHY_Speed_Status      ((u16)0x0200) /*!< Configured information of Speed: 10Mbps */
#define PHY_Duplex_Status     ((u16)0x0100) /*!< Configured information of Duplex: Full-duplex */
```

## 6. Vývojové kity

### 6.1 comStick

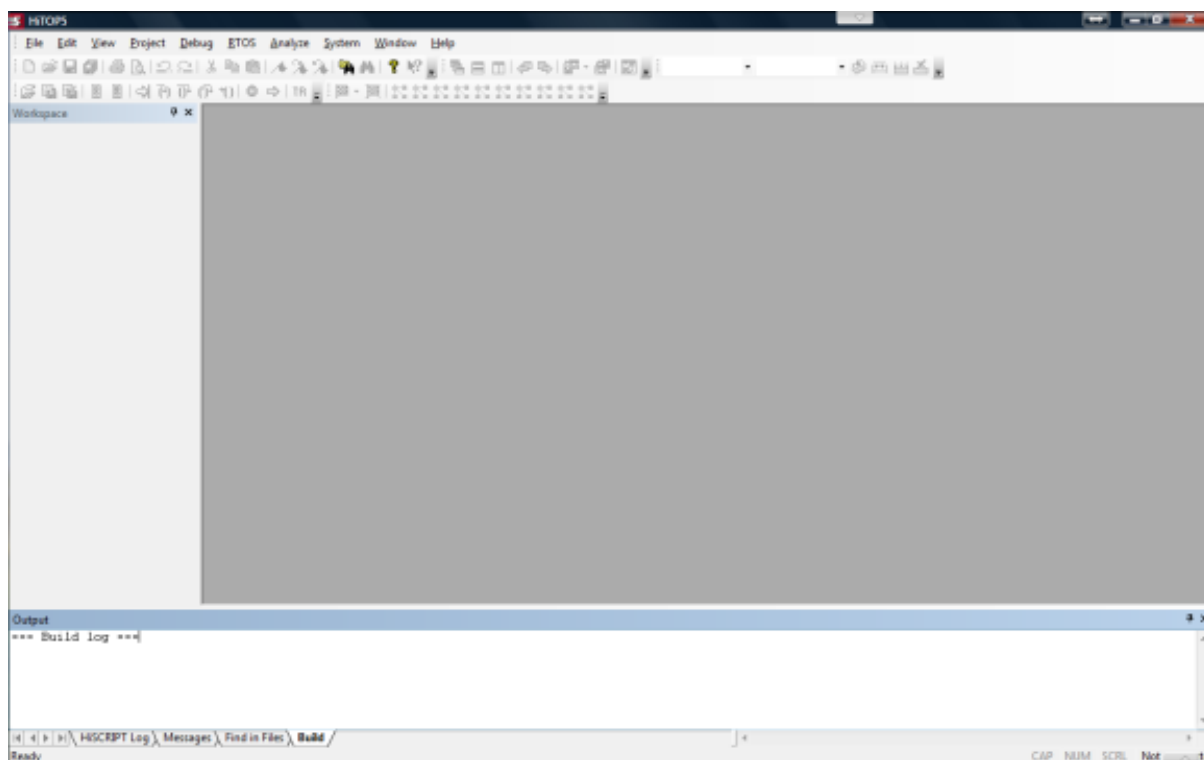
Je vývojová deska s uP STM32F107(obr. 12) s fyzickou vrstvou STE100P od firmy HITECH[13]. Deska navíc obsahuje převodník JTAG/USB pro možnost ladění pomocí vývojového prostředí HiTOP. Vývojový kit obsahuje konektor pro ethernet RJ45, konektor mikroUSB a 80pinový konektor pro připojení rozšiřující desky. Rozšiřující deska však nebyla k dispozici, proto comStick byl použit pouze pro oživení ethernetu.



Obrázek 12: STM32comStick

### 6.1.1 Programování STM32comStick

K programování a k vývoji aplikací pro comStick je primárně určeno vývojové prostředí HiTOP. Výrobce, bohužel, omezil i použití vývojového prostředí HiTOP.



Obrázek 13: Vývojové prostředí HiTOP

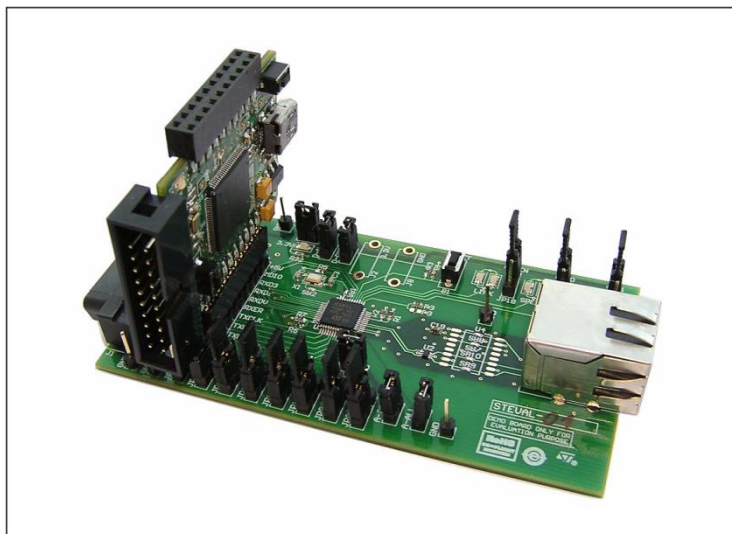
S verzí bez zakoupené licence bylo proto možné ladit pouze programy, které byly menší než 32kB. Stejné omezení, bohužel, platilo i pro nahrávání projektů do comSticku. Jelikož téměř každý TCP/IP stack převyšuje hodnotu 32kB, nebylo prostředí HiTOP absolutně použitelné pro práci s ethernetem. Jako vývojové prostředí bylo tedy zvoleno prostředí Ride7 od firmy

Raisonance. Prostředí umožňuje vytvářet projekty libovolné velikosti a ty poté i ladit v simulátoru. Ladění přímo pomocí ladícího rozhraní JTAG není, bohužel, možné. Tuto možnost nabízí pouze originální vývojové prostředí pro comStick. Z tohoto plyne nevýhoda, že ladění je možné pouze pomocí metody pokus–omyl. Další nevýhodou použití Ride7 je nemožnost nahrání aplikace přímo ve vývojovém prostředí. Proto se nejprve ve vlastnostech projektu musí nastavit, po přeložení projektu vytvoření hex souboru. Tento soubor poté může být nahrán do comSticku pomocí prostředí HiTOP, které naštěstí nemá omezenou možnost nahrávání externích hex souborů.

### 6.1.2 Nahrání hex souboru do STM32comStick

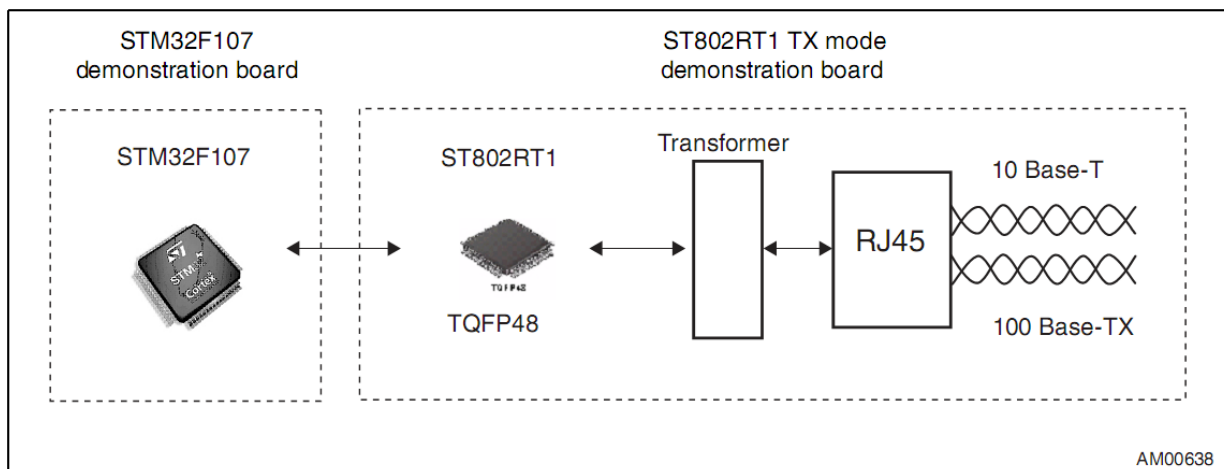
Nejprve spustíme vývojové prostředí HiTOP. Klikneme na *Project->New*, jde nám pouze o zpřístupnění voleb, které, bohužel, bez vytvoření projektu nejsou k dispozici. Je možné použít i volbu *open*, pokud máme již nějaký projekt k dispozici z dřívějšího. Po zvolení volby *new*, vybereme Empty Project a klikneme na next. Poté budeme vyzváni k vložení jména projektu a jeho umístění, zde není nutné nic měnit, můžeme použít next. Dále budeme vyzváni ke zvolení nástroje pro debutování. Zvolíme STM32comStick. Dále musíme zvolit typ uP. Standardně by měl již být nastaven STM32F107VC. Po stisku next budeme vyzváni ke zvolení rozhraní pro komunikaci s přípravkem comStick. Zvolíme proto USB a zadáme jeho pořadové číslo a dáme next. Další nabídku, zabývající se zvolením start up scriptu můžeme přeskóčit. Poslední nabídka se týká licencí. Jelikož žádnou nemáme, zvolíme *I want to continue evaluation*. Nyní je vytvořen prázdný projekt, který ovšem používat nebudeme. Hlavní je, že je k dispozici volba nahrání kódu do comSticku. Klikneme proto na *Debug->Download* a v další záložce zvolíme *Hex...*, poté již stačí vybrat umístění vytvořeného hex souboru. Po nahrání souboru zvolíme *System->Reset test system* a potvrdíme. Poté již stačí spustit program v comSticku pomocí volby *Debug->Go*. Toto spuštění je nutné, jsme-li v prostředí HiTOP, které po nahrání automaticky program nespustí. Vyjmeme-li však comStick z USB a znovu připojíme program v uP ten se již automaticky spustí.

## 6.2 STEEVAL-PCC010V1



Obrázek 14: Vývojový kit STEEVAL-PCC010V1

Je set skládající se ze dvou vývojových modulů[11](obr. 14). První modul obsahuje uP STM32F107VC, a má vyvedený konektor miniUSB, pomocí něhož je celý kit také napájen. Dále obsahuje 20pinový konektor rozhraní JTAG, sloužící k nahrání softwaru a ladění.

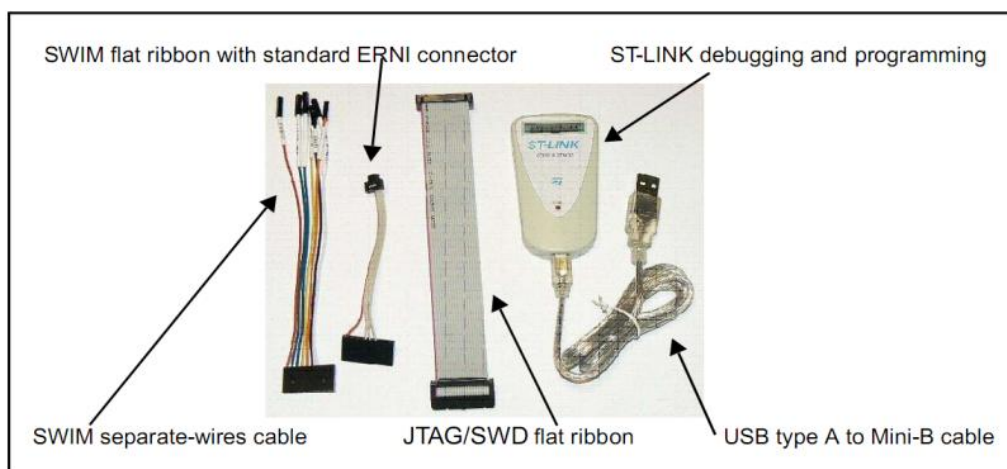


Obrázek 15: Uspořádání STEEVAL-PCC010V1

Dalším rozhraním je 20pinový konektor, na který jsou vyvedeny některé piny uP pro vytváření zkušebních aplikací(UART,SPI,TIM,GPIO). Poslední 20pinový konektor slouží pro připojení druhé části setu obsahující fyzickou vrstvu ST802RT1.

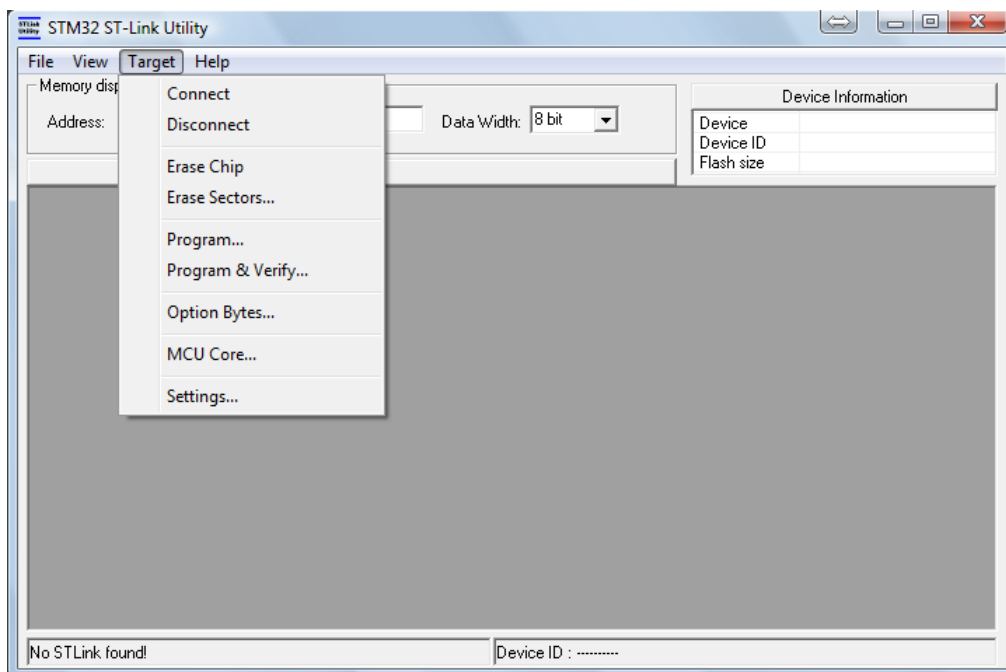
### 6.2.1 Programování STEEVAL-PCC010V1

Jako vývojové prostředí je použito Ride7 od Raisonance, které dokáže vytvořit hex soubor pro nahrání do uP. Nahrávání zde bude prováděno pomocí převodníku USB/JTAG od firmy STM ST-Link[7](obr. 16).



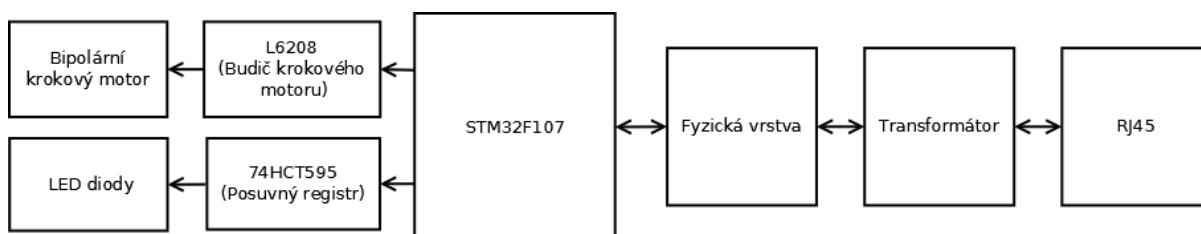
**Obrázek 16: ST-Link**

K tomuto převodníku je možno stáhnout software ST-Link Utility[8], s jehož pomocí bude program nahrán do uP. Nevýhodou programu ST-Link utility je, že nedokáže nahrát hex soubor, ale pouze bin soubory. Proto musíme převést hex soubor vytvořený v prostředí Ride na bin, například pomocí programu HEXCNOV. Vše provedeme tak, že umístíme hex soubor do stejného adresáře, kde je umístěn program HEXCONV.EXE, a v příkazové řádce zadáme příkaz: `hexconv nazev.hex nazev.bin`. Po proběhnutí programu získáme náš požadovaný binární soubor, který můžeme nahrát do uP. Po připojení napájení k STEEVAL-PCC010V1 a připojení rozhraní JTAG, spustíme program ST-Link Utility(obr. 17), klikneme na *Target->Connect*. Po úspěšném připojení již stačí kliknout na *File->Open file...* a vybrat námi vytvořený bin soubor, který bude nahrán do uP. Po vybrání souboru se nám zobrazí výzva, zda budeme chtít nahrát otevřený soubor do uP. Potvrdíme volbu stiskem OK. Po potvrzení se otevře okno s tlačítkem program, pomocí něhož vybraný soubor bude uložen do paměti uP. Po stisknutí tlačítka program budeme informováni nejprve o vymazání paměti uP a poté o průběhu zápisu. Po skončení této operace bude program, uložený do paměti uP, automaticky spuštěn.



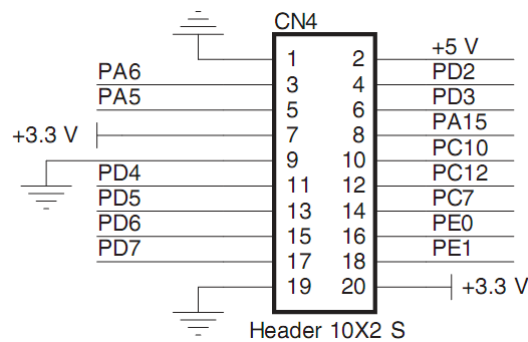
Obrázek 17: Software ST-Link Utility pro nahrávání programů do uP přes ST-Link

## 7. Blokové schéma zapojení celého experimentu



Obrázek 18: Blokové schéma zapojení experimentu

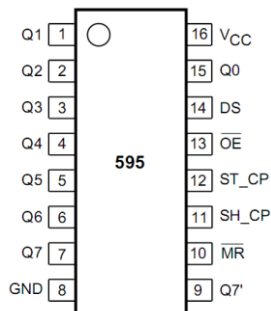
Experiment(obr. 18) se bude skládat z bloku kde je obsažen uP STM32F107. K němu bude připojena PHY a transformátor. Všechna tyto zařízení obsahuje vývojový kit STEEVAL-PCC010V1. K tomuto kitu bude připojen posuvný registr 74HCT595, který bude ovládat LED diody. Dále bude ke kitu připojen budič krokového motoru L6208, který bude zajišťovat ovládání krokového motoru. Budič L6208 i posuvný registr 74HCT595 budou připojeny k uP pomocí 20pinového konektoru(obr. 19), který obsahuje všechny potřebné výstupy uP pro ovládání.



Obrázek 19: Znáznornění konektoru u SEEVAL-PC010V1 který umožňuje připojení externích periférií

## 7.1 Ovládání LED diod

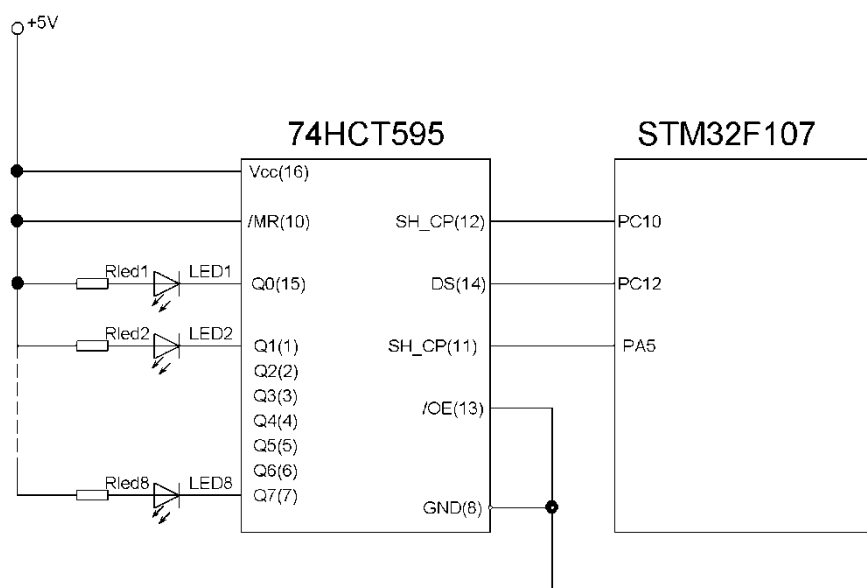
K ovládání led diod použijeme posuvný registr 74595[16](obr 20), s jehož pomocí může být ovládáno až 8 LED diod a po zapojení dvou obvodů za sebou až 16. Pro ovládání obvodu 74595 jsou nutné tři řídicí signály, a to SH\_CP udávající hodinový signál posuvnému registru, DS, který slouží pro zadávání dat do registru a ST\_CP sloužící pro přenos dat z posuvného registru na výstup obvodu. Další vstupy 74595 není třeba ovládat (jako třeba CE a MR, který potřebujeme mít stále aktivní, aby mohlo být s obvodem kdykoliv manipulováno).



Obrázek 20: Posuvný registr 74HCT595

K řízení budeme tedy potřebovat 3 řídicí signály(obr. 21), jeden pro zápis dat, další hodinový signál pro posuvný registr, který reaguje na náběžnou hranu signálu a hodinový signál pro zapsání dat na výstup obvodu, reagující také na náběžnou hranu. Tyto signály mohou být generovány uP jednak softwarově, což by zatěžovalo zbytečně uP a nebo můžeme výhodně využít rozhraní SPI, které je určeno k sériovému přenosu dat. SPI u STM32 má 4 signály a to MOSI (master output slave input), MISO(master input slave output), CLK a NSS. Rozhraní SPI umožňuje komunikaci mezi jedním zařízením MASTER a jedním aktivním zařízením

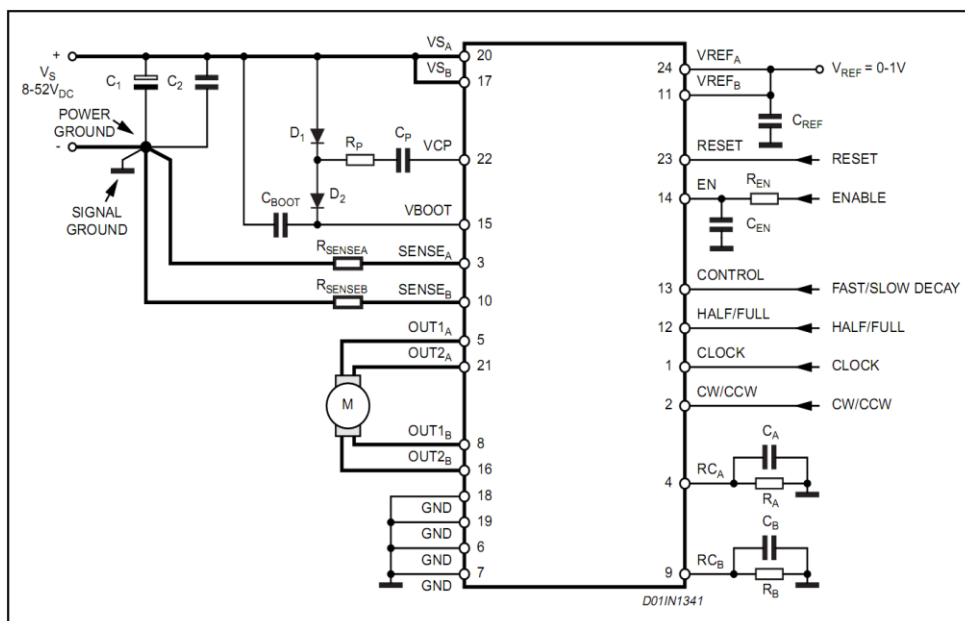
SLAVE. Pro naši aplikaci bude uP nastaven jako zařízení MASTER a bude mu umožněno pouze posílání dat. Jelikož obousměrná komunikace není třeba, pro posílání dat bude použit výstup MOSI. SPI obsahuje také hodinový signál CLK pro synchronizaci přenosu. Signál CLK použijeme jako hodinový signál pro zápis do posuvného registru. Jako hodinový signál pro přenos dat na výstup obvodu 74595 nám poslouží výstupní pin, který před zápisem pomocí SPI nastavíme do nuly a po skončení přenosu dat pomocí SPI nastavíme pin do jedničky a tím vytvoříme potřebnou náběžnou hranu pro obvod 74595. Ten zapíše přijatá data na výstup a tím změní nastavení LED diod sloužících pro identifikaci vysílacího obvodu s režimem master.



**Obrázek 21:** Schéma zapojení STM32F107 a posuvného registru 74HCT595

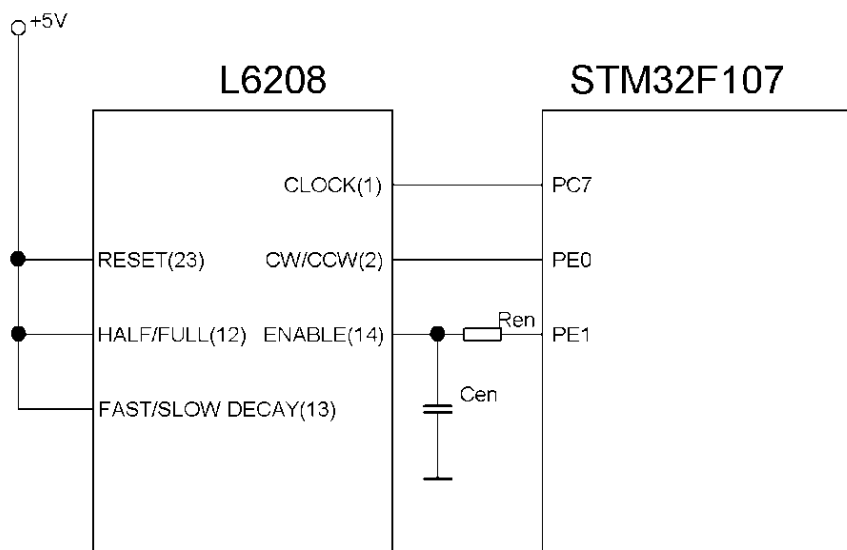
## 7.2 Ovládání krokového motoru

Krokový motor je ovládán pomocí budiče L6208. Je to budič bipolárního krokového motoru od společnosti STM. Zapojení budiče (obr. 22) provedeme podle doporučeného zapojení v dokumentaci [15]. Budič může být napájen napětím v rozsahu 8-52V. Pro řízení je však využito pětice TTL vstupů. Díky tomu je možné budič ovládat pomocí uP. K řízení budiče budou



**Obrázek 22: Schéma zapojení obvodu L6208 pro ovládání krokového motoru**

využity 3 signály z uP (obr. 23). Prvním chip enable (EN), který aktivuje tranzistory v můstcích sloužících k ovládání vinutí krokového motoru. Ovládání výstupu je dobré kvůli snížení odběru a zatížení budícího čipu v okamžiku, kdy s motorem nepracujeme. Jelikož je obvod umístěn na kontaktním poli a proudový odběr může dosáhnout řádu ampér, je lepší po dobu, kdy nepotřebujeme pracovat s krokovým motorem nechat budič vypnutý, aby se nepřehříval. Dalším vstupem je směr otáčení motoru. Tento vstup bude ovládán normálním výstupem uP. Posledním řídícím signálem nutným pro ovládání budiče je hodinový signál, udávající krok motoru. Pro ovládání tohoto výstupu použijeme čítač, který nastavíme tak, aby při načítání jeho maximální hodnoty změnil hodnotu na výstupu na opačnou polaritu. Tím dosáhneme přesného hodinového signálu se střídou 1:1. Pro naši úlohu je však ještě nutno měřit počet kroků, které motor vykoná. STM32 však umožňuje za náš čítač generující hodinový impulz zapojit další, který bude čítat počet překlopení prvního čítače. Budeme-li chtít provést určitý počet kroků, nastavíme nejprve druhému čítači hodnotu do které má čítat a zapneme oba čítače. Po načítání požadovaného počtu kroků čítač provede přerušení, ve kterém oba čítače vypneme a vynulujeme. Dalšími dvěma vstupy, které obsahuje budič L6208, je vstup určující, zda bude posun krokového motoru probíhat pouze po polovičních nebo celých krocích. V našem případě jsme tento vstup nastavili k napětí +5V, na poloviční kroky. Poslední vstup je pro určení SLOW/FAST DECAY módu. Tento mód určuje způsob, jakým obvod reaguje na překročení proudu. V našem případě nastavíme mód na SLOW DECAY kdy se při překročení proudu tento proud pomalu vybíjí přes diody v můstku viz. Hodnotu tohoto pinu nastavíme tedy na +5V.



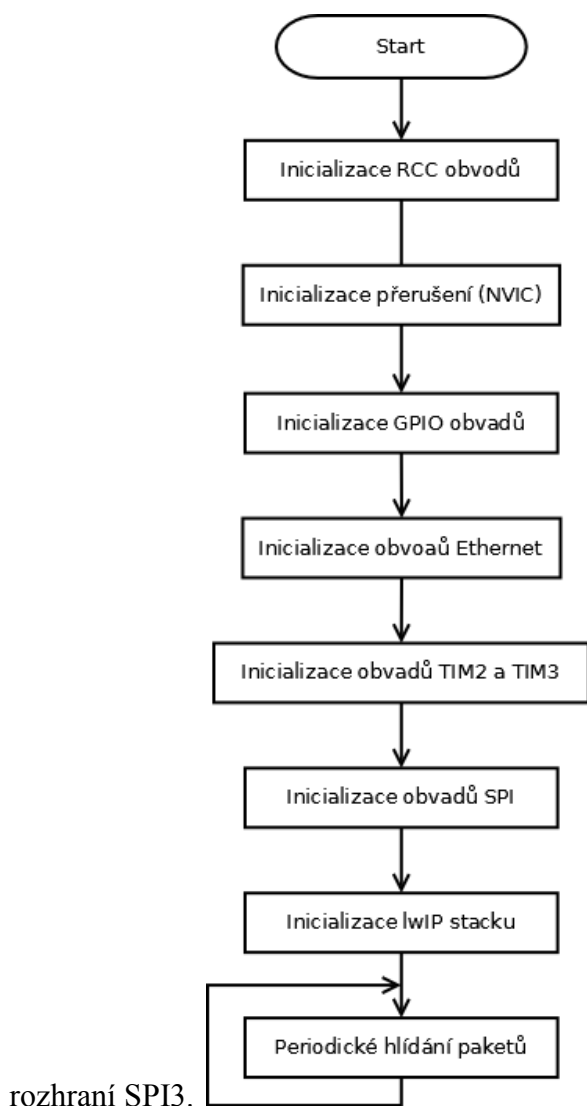
Obrázek 23: Schéma zapojení obvodu L6208 k uP STM32F107

## 8. Program

Jako vzor programu byl použit ukázkový program ukazující základní práci lwIP stacku na vývojovém kitu STM3210C-EVAL. Celý program byl proto upraven aby odpovídal nastavení pro kity comStick a STEEVAL-PCC010V1. Před umístěním programu d uP je zapotřebí provést vlastní nastavení TCP/IP stacku. V souboru netconf.c se nastaví IP adresa, pomocí které se bude možno připojit k experimentu a MAC adresa. Soubor stm32f107.c slouží k základnímu nastavení periférií, je zde také možnost odkomentováním volby #define RMII\_MODE přepnout uP do modu, kdy komunikuje s PHY pomocí RMII. V souboru lwipopts.h je možné nastavit, zda bude použita statická adresa, nebo použito DHCP pro přidělování IP adres. Dále je možné určit, kolik TCP připojení může být připojeno souběžně. Pro naši aplikaci postačí jedno. Dle je možné nastavit, jak velká část paměti uP bude určena pro příjem přijímaná data. Je zde také možné nastavit zda kontrolní součty odchozích a příchozích paketů budou prováděny hardwarově nebo softwarově. Po provedení nastavení lwIP stacku může být již program přeložen a uložen do paměti uP.

Pro spuštění programu po připojení napájení k uP program nejprve provede pomocí metody System\_Setup() inicializaci(obr. 24) periférií uP, které budeme používat. Provede se nastavení obvodu zajišťujícího hodinové signály pro celý uP. Bez povolení hodinového

signálu pro všechny periferie které budeme používat, nebudou dané periferie funkční. Povoleny budou všechny brány A-E a také jejich alternativní funkce, aby bylo možno namapovat vstupně výstupní brány, pro ovládání ostatními periferiemi uP. Dále povolíme rozhraní ethernet, čítač TIM3, který bude zajišťovat hodinový signál pro krokový motor a čítač TIM2 který bude počítat počet kroků motoru. Další povolenou periferií bude sériové



**Obrázek 24: Inicializace programu**

Dalším krokem inicializace je konfigurace vektorů přerušení. Přerušení uP budeme potřebovat pro obvody ethernetu a také pro TIM2, který bude oznamovat, že dosáhl své maximální hodnoty.

Následuje nastavení vstupních a výstupních bran. Nejprve jsou aktivovány a přemapovány piny bran, které budou využívat obvody ethernetu. Provedené namapování je pro MII. Kdybychom použili RMII, bylo by možné redukovat počet pinů. Podrobnějším

nastavením pinů pro ethernet není nutné se zabývat, jelikož zapojení na vývojové desce nedovoluje jejich jiné využití. Další piny budou zapotřebí, pro ovládání budiče krokového motoru a posuvného registru. Pro posuvný registr musíme přemapovat piny pro použití rozhraním SPI. Proto výstup SPI MOSI bude namapován na pin PC12 a výstup CLK na pin PC10. Poslední pin, který bude ovládat zápis dat na výstup posuvného registru, je PA5. Dále budeme potřebovat výstupy pro ovládání krokového motoru. Pro hodinový signál krokového motoru bude využit výstup TIM3, který namapujeme na pin PC7. Pro ovládání krokového motoru použijeme ještě piny PE0 jako povolení budiče krokového motoru a PE1, který bude udávat směr otáčení krokového motoru.

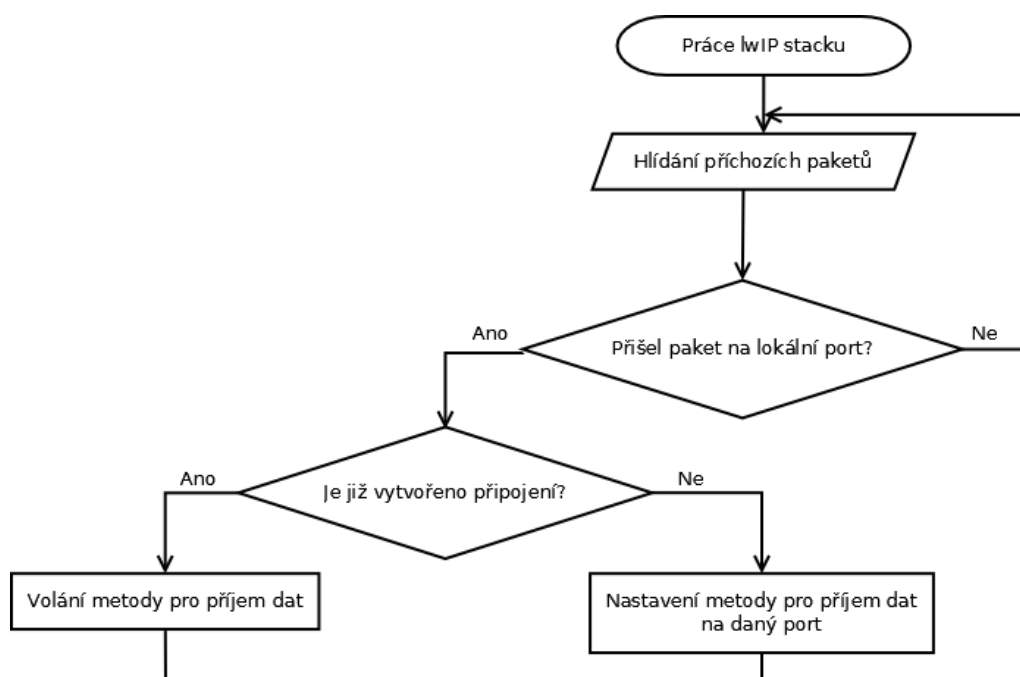
Dále se nastaví vlastní obvody Ethernetu s DMA a PHY. A přerušení. Jelikož obvod ethernet bude pod správou TCP-IP stacku, není dobré do nastavení zasahovat. Nakonec je povoleno přerušení, které bude rozhraní vytvářet při přijetí paketu.

Dále je nakonfigurováno rozhraní SPI, které bude posílat seriově 8b data posuvnému registru. Jelikož registr reaguje na náběžnou hranu signálu, nastavíme SPI tak, aby platná data byla na výstupu s náběžnou hranou hodinového signálu. Proto nastavíme hodnotu CPOL na 1 a hodnotu CPHA na 1. Nakonec nastavíme SPI jako master. Tím zaručíme, že MOSI(PC12) bude pracovat jako výstup.

Následuje nastavení dvou čítačů TIM2 a TIM3. TIM3 bude čítat do hodnoty 60000, s předděličkou 3. Výstup TIM3 pak nastavíme tak, aby při každém dosažení maximální možné hodnoty překlátil svůj stav. Jelikož hodinový signál pro čítač má hodnotu 72MHz, bude čítač vytvářet obdélníkový signál pro krokový motor s periodou 200Hz. Aby bylo možné spočítat počet kroků čítače, připojíme za TIM3 čítač TIM2 který bude časován přetečením TIM3. Hodnotu do které bude čítač čítat udává uživatel přes rozhraní ethernet.

Po provedení inicializace uP je provedena inicializace lwIP stacku metodou LwIP\_Init(); v této metodě je provedeno nastavení lwIP stacku, je nastavena IP adresa, kterou využijeme pro připojení k přípravku. Dále je nastavena MAC adresa pro jednoznačnou identifikaci zařízení. A provede se alokace paměti pro přijímaná data.

Následuje inicializace modulu pro komunikaci přes ethernet. Metoda Komunikace\_init() nejprve nastaví, že budeme očekávat připojení přes TCP na portu 23 a po žádosti o vytvoření spojení zavolá metodu Komunikace\_accept().

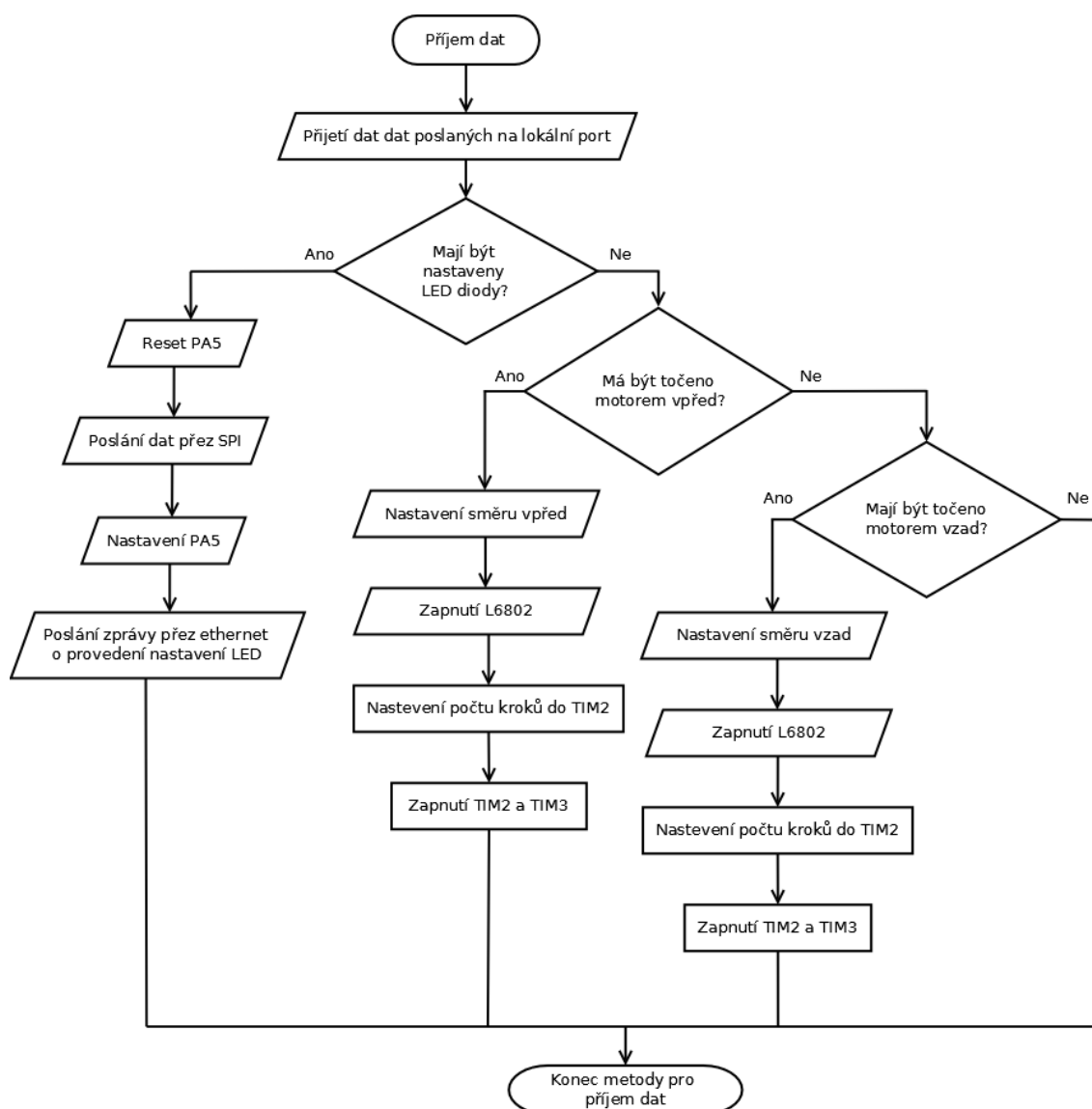


**Obrázek 25: Práce lwIP stacku**

Po skončení této metody program přejde do nekonečné smyčky(obr. 25), v níž pouze kontroluje pomocí metody `System_Periodic_Handle()`, zda na aktivních připojeních nebyla přijata data.

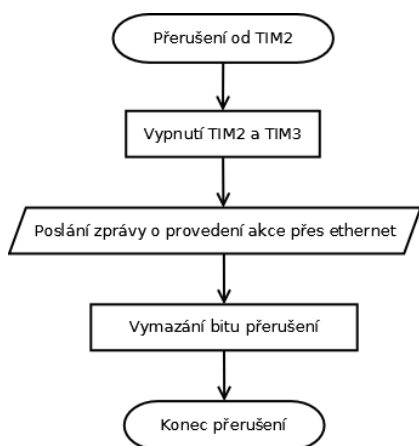
Jakmile TCP-IP stack zjistí, že je zde požadované nové připojení zavolá metodu `Komunikace_accept()`, která nastaví metodu `Komunikace_recv` jako metodu, která se bude volat, přijdou-li platná data. A `Komunikace_conn_err` jako metodu, která se zavolá při chybě. Poté odešle metoda zpět zprávu, že komunikace byla navázána.

Při příjmu zprávy je tedy zavolána metoda `Komunikace_recv`(obr. 26). V této metodě se přečte přijatá zpráva a pokud obsahuje text skládající se z jednoho písmene a čísla, pak se provede jedna z následujících možností:



**Obrázek 26: Práce programu pro ovládání LED diod a krokového motoru**

Je-li počáteční písmeno zprávy L, pak číslice za ním znamená hodnotu, která bude zapsána do posuvného registru SPI. Nejprve však pomocí metody `GPIO_WriteBit(GPIOA,GPIO_Pin_5,Bit_RESET)` resetujeme pin PA5, abychom jím po úspěšném zápisu přes SPI mohli vytvořit náběžnou hranu, která zapíše data v posuvném registru na jeho výstup. Poté pomocí metody `SPI_I2S_SendData(SPI3, ((u8)done))` pošleme data po SPI. Kontrolujeme, zda již přenos proběhl pomocí bitu TXE, který se nachází v registrech SPI. Metodou `GPIO_WriteBit(GPIOA,GPIO_Pin_5,Bit_SET)` vytvoříme již zmíněnou náběžnou hranu pro posuvný registr. Po skončení zápisu do posuvného registru je poslána po ethernetu zpráva o úspěšném provedení akce.

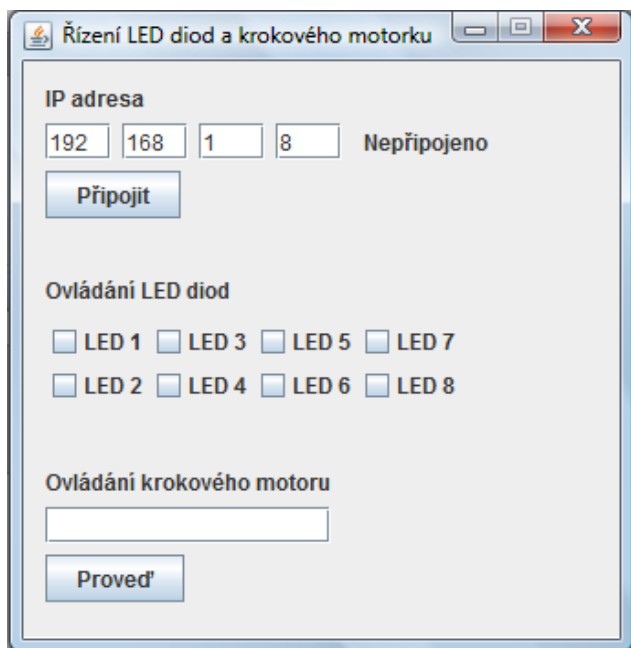


**Obrázek 27: Obsluha přerušení čítače TIM2**

Je-li počáteční písmeno zprávy R nebo D, nastaví se nejprve pin PE1, který udává směr, kterým se bude krokový motor otáčet. Poté se nastaví pin PE0 do logické jedničky, tím dojde k aktivaci budiče krokového motoru. Dále je na čítači TIM2 nastavena hodnota čítání na dvakrát větší hodnotu než je požadovaný počet kroků. Po tomto nastavení je již aktivován TIM3 a TIM2. Jakmile TIM2 napočítá požadovaný počet kroků a dojde k jeho přetečení, vyvolá se přerušení (obr. 27), ve kterém se vynulují hodnoty obou čítačů a dojde k jejich vypnutí. Vypne se i budič krokového motoru resetováním pinu PA0. Nakonec je po ethernetu poslána zpráva o skončení požadované činnosti.

## 9. Popis programu pro komunikaci s experimentem

Program je napsaný v programovacím jazyce Java ve vývojovém prostředí NetBeans. Program pro komunikaci pomocí ethernetu využívá knihovny Java.net. Program obsahuje jednoduché uživatelské prostředí sloužící k jednoduchému ovládání experimentu (obr. 28). Aplikace nejprve vyžaduje připojení k experimentu. Pro připojení je nutno zadat IP adresu na kterou se má připojit. Při zadání špatné IP adresy k připojení nedojde. Proběhne-li připojení v pořádku, aplikace dovolí ovládání experimentu. Ovládání je rozděleno do dvou částí, první umožňuje ovládat LED diody připojené na posuvný registr 74HCT595. Druhá část dovolí ovládat krokový motor připojený přes budič L6802. Pro ovládání krokového motoru je nutné zadat počet kroků, který má být proveden. Pro kroky vpřed se zadá kladné celé číslo, pro kroky vzad se zadá záporné celé číslo poté již stačí pouze stisknout tlačítko *Proved'* a aplikace pošle experimentu povel pro provedení námi zvoleného počtu kroků. Proté aplikace čeká až experiment provede danou akci a po obdržení zprávy o úspěšném odkrokování umožní opětovné řízení experimentu.



Obrázek 28: Program pro ovládání experimentu přes PC

## 10. Zhodnocení výsledků

Práce byla zaměřena na prozkoumání komunikačního rozhraní ethernet, které je využito pro komunikaci s programem umístěným na PC. Nutnou podmínkou pro vznik komunikace byla i nutnost výběru a implementace TCP/IP stacku. Nejprve jsem se pokusil implementovat uIP stack, bohužel neúspěšně vzhledem k tomu že jsem se s uP teprve učil pracovat. Poté jsem se rozhodl implementovat lwIP stack, který obsahoval ukázkový projekt a ten se mi již podařilo zprovoznit na STM32comStick, toto zařízení však neumožňovalo připojení posuvného registru a budiče krokového motoru. Proto se po konzultaci se zadávajícím bakalářské práce přešlo na zařízení STEEVAL-PCC010V1, které na rozdíl od STM32comStick obsahuje 20pinový konektor, ke kterému je možné připojit libovolná zařízení. Vlastní programování vývojových kitů nebylo však jednoduché. Jelikož vývojové prostředí Ride7 neumožňuje debugování na přípravcích, muselo být vše prováděno metodami pokus–omyl, což způsobovalo časovou náročnost programování. To neulehčuje ani vlastní pochopení práce uP, jehož referenční manuál obsahuje okolo 1000stran, jehož základy bylo nutné se naučit. Bohužel, jsem nemohl vyzkoušet možnost použít na komunikace mezi uP a PHY režim RMII, neboť se jednalo o zapůjčený vývojový kit a změna nastavení by

znamenalala nutnost přepájení některých spojovacích plošek na desce plošných spojů(v [11] na str. 18).

## Použitá literatura

- [1] RM0041-Referenční manuál k STM32F1XX [online]  
<http://www.st.com/stonline/products/literature/rm/16188.pdf>
- [2] AN3102-Popis aplikace lwIP [online]  
<http://www.st.com/stonline/products/literature/an/16620.pdf>
- [3] AN2820-Ovládání bipolárního krokového motoru [online]  
<http://www.st.com/stonline/products/literature/an/16620.pdf>
- [4] AN2592-Vytvoření 32bitového čítače, [online]  
<http://www.st.com/stonline/products/literature/an/13711.pdf>
- [5] STM32F105/107xx-Datasheet [online]  
<http://www.st.com/stonline/products/literature/ds/15274.pdf>
- [6] STM32F105/107xx revision Z errata sheet [online]  
<http://www.st.com/stonline/products/literature/es/15866.pdf>
- [7] ST-LINK-Příručka k produktu ST-LINK [online]  
<http://www.st.com/stonline/products/literature/um/15285.pdf>
- [8] STM32 ST-Link Utility-Popis programu ST-Link Utility [online]  
<http://www.st.com/stonline/products/literature/um/16987.pdf>
- [9] STE100P-Datasheet k fyzické vrstvě [online]  
<http://www.st.com/stonline/products/literature/ds/6806.pdf>
- [10] ST802RT1-Datasheet k fyzické vrstvě [online]  
<http://www.st.com/stonline/products/literature/ds/17049/st802rt1a.pdf>
- [11] UM0819-Manuál k vývojovému kitu s STM32F107 a ST802RT1 [online]  
<http://www.st.com/stonline/books/pdf/docs/16379.pdf>
- [12] Schéma přípravku STM32-comStick [online]  
[http://www.hitex.com/fileadmin/free/stm32-comstick/usr\\_scm\\_stm32com-a2-1.pdf](http://www.hitex.com/fileadmin/free/stm32-comstick/usr_scm_stm32com-a2-1.pdf)
- [13] Datasheet k přípravku STM32-comStick [online]  
<http://www.hitex.com/fileadmin/free/stm32-comstick/stm32-comstick-ds.pdf>
- [14] Popis CSMA/CD [online] <http://cs.wikipedia.org/wiki/CSMA/CD>

[15] L6208-Datasheet [online]

<http://www.st.com/stonline/products/literature/ds/7514.pdf>

[16] 74HCT595 – Datasheet [online]

[http://www.nxp.com/documents/data\\_sheet/74HC\\_HCT595.pdf](http://www.nxp.com/documents/data_sheet/74HC_HCT595.pdf)

### **Obsah přiloženého CD**

- Ethernet\_uP - Program pro uP.
- Ethernet\_PC - Program pro ovládání experimentu přes PC.
- Manuály – Manuály na něž se odkazuje literatura.