VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

BUZENÍ POČÍTAČE S ATMEGA

WAKE ON LAN WITH ATMEGA

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR
STANISLAV HOMOLA

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. TOMÁŠ MARADA, Ph.D.

BRNO 2013

Tady se musí vložit zadání

Abstrakt

Tato práce podrobně popisuje realizaci zapnutí počítačů v síti pomocí funkce Wake on Lan, implementování www serveru do mikroprocesoru a měření teploty. K tomuto je využito mikroprocesoru ATmega 32, teplotního čidla DS18B20 a ethernetového modulu ENC28J60. V úvodní části jsou popsány jednotlivé komponenty a vysvětlena jejich činnost. Dále jsou zde popsány využívané packety protokol TCP/IP a princip www serveru. V poslední části je popsána realizace www serveru s funkcí Wake on Lan a zobrazováním teploty na internetové stránce serveru.

Abstract

The bachelor´s thesis describes realization of switch on computers using Wake on Lan function, implementation of web server into the Microcontroller and temperature measurement. For this purpose, it is used microcontroller ATmega 32, temperature sensor DS18B20 and Ethernet module ENC28J60. The introductory chapter describes all components and explains their functions. Following part contains a description of packet protocols TCP/IP and principle of the web server. In the last section, it is described the realization of th web server with funcion Wake on Lan and depicting of the temperature on the web page of the server.

Klíčová slova

ATmega 32, ethernet, mikrokontroler, teplotní čidlo, Wake on Lan, DS18B20, ENC28J60.

Key words

ATmega 32, ethernet, microcontroller, thermometer, Wake on Lan, DS18B20, ENC28J60.

Bibliografická citace

HOMOLA, S. *Buzení počítače (WOL) s ATMega*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2013. 36 s. Vedoucí bakalářské práce Ing. Tomáš Marada, Ph.D

Čestné prohlášení

Prohlašuji, že jsem bakalářskou práci na téma „Buzení počítače (WOL) s ATmega“ vypracoval samostatně s použitím odborné literatury a pramenů uvedených v seznamu, který tvoří přílohu této práce.

24. května 2013

……
Stanislav Homola

Poděkování

Rád bych poděkoval Ing. Lubomíru Starému a svému vedoucímu práce Tomáši Maradovi za cenné rady, trpělivost a předané zkušenosti při vypracovávání bakalářské práce.

Obsah

1 Úvod 7
2 Cíl práce 8
3 Vývojový kit EvB 4.3 9
3.1 Základní údaje o vývojovém kitu EvB 4.3 v 4 9
4 ATmega 32 10
4.1 Základní údaje o mikroprocesoru ATmega 32 10
4.2 Obsluha portů 10
4.3 SPI rozhraní 11
5 ENC28J60 13
5.1 Základní údaje o ENC28J60 13
5.2 Organizace paměti 14
5.3 Kontrolní registry 15
5.3.1 ECON1 15
5.4 Odesílání/přijímání dat po síti 16
5.4.1 Inicializace 16
5.4.2 Odesílání dat 17
5.4.3 Přijímání dat 17
5.5 PHY registry 18
5.5.2 half/full duplex mód 19
6 Teplotní čidlo DS18B20 20
6.1 Princip měření DS18B20 20
6.1.1 Příprava čipu na měření teploty 21
6.1.1.1 Inicializace 21
6.1.1.2 Skip ROM 21
6.1.2 Měření teploty 21
6.1.3 Příprava čipu ke čtení naměřené teploty 21
6.1.4 Čtení teploty 21
7 Wake on Lan 23
7.1 Ethernetová komunikace 23
7.2 Princip Wake on Lan 24
7.3 UDP (User datagram packet) 24
7.4 Magic packet 25
8 www server 26

8.1 HTTP protokol 26
9 Praktická realizace www serveru s funkcí WOL 27
9.1 Uživatelská obsluha www serveru 27
9.2 Spouštění www serveru 27
9.3 Modifikace projektu 28
9.3.1 Implementace měření teploty do projektu www serveru 28
9.3.2 Implementace funkce Wake on Lan do projektu www serveru 29
9.3.2.1 Zadávací políčko 29
9.3.2.2 Sestavení Magic packetu 29
9.4 Rozbor funkcí obstarávající síťovou komunikaci 30
9.4.1 Funkce fill_tcp_data_p 30
9.4.2 Funkce fill_tcp_data 30
9.4.3 Funkce print_webpage 30
9.4.4 Funkce init_ip_arp_udp_tcp 31
9.4.5 Funkce enc28j60_packet_send 31
9.4.6 Funkce enc28j60_packet_receive 31
9.4.7 Funkce enc28j60_init 32
9.5 Struktura funkce main 32
9.5.1 Tělo funkce main bez nekonečného cyklu 33
9.5.2 Nekonečný cyklus funkce main 33
9.5.2.1 Detekce neúplného zadáni URL 34
9.5.2.2 Test požadavku na funkci Wake on Lan 34
9.6 Obsluha teplotního čidla DS18B20 34
9.6.1 Funkce ds18b20_reset 34
9.6.2 Funkce ds18b20_send_instruction 35
9.6.3 Funkce ds18b20_read_bite 35
9.7 Převod naměřené teploty na pole znaků 35
10 Závěr 36
Seznam použitých zdrojů 37
Seznam příloh: 39

1 Úvod

Pro současnou společnost je charakteristické, že každodenní činnost člověka je značně spjata s výpočetní technikou. Počítače vykonávají činnosti, kterými lidem usnadňují práci, či dokonce umožňují realizace projektů, jež by jinak nebyly proveditelné. Počítače jsou v současnosti již zcela běžným vybavením domácností, kanceláří, škol, atd. Je tedy logické, že došlo k propojení jednotlivých PC v rámci institucí do tzv. lokálních sítí, které umožňují komunikaci mezi počítači.

Systém BIOS každého počítače umožňuje aktivaci funkce Wake on Lan, která, je-li aktivována, udržuje síťovou kartu napájenou i při vypnutém počítači. Díky tomu může síťová karta sledovat dění na lokální síti. Takto aktivovaná karta může zapnout počítač, obdrží-li tzv. „Magic packet“.

Funkce Wake on Lan umožní zapnout ostatní počítače v lokální síti, aniž by musely být pro uživatele fyzicky dostupné. Z důvodu možné fyzické absence uživatele v místnosti (učebně) se zapínanými počítači je vhodné měřit teplotu, aby bylo možné sledovat, zda jsou zde optimální teplotní podmínky pro práci uživatele a činnost zařízení.

Jelikož je „Magic packet“ malý datový soubor vyslaný po síti, lze pro tuto aplikaci využít jednoduchého www serveru implementovaného do mikroprocesoru ATmega 32, který bude připojen k lokální síti pomocí ethernetového modulu ENC28J60 a dále obsluhovat teplotní čidlo DS18B20.

Toto řešení je vhodné z důvodů nízké energetické náročnosti, velkého teplotního rozpětí se zaručenou funkčností, malých pořizovacích nákladů a snadné obsluhy.

2 Cíl práce

Cílem této práce bylo seznámit se s problematikou programování mikroprocesorů, správně propojit jednotlivé komponenty použité k realizaci a napsat program, který by splňoval zadání.

Dílčí úkoly, jež bylo nutné splnit:

- Nastudovat a správně zapojit ethernetový konektor ENC28J60.
- Modifikovat použitý kód pro www server.
- Vložit do použitého kódu funkci WOL.
- Nastudovat a správně zapojit teplotní čidlo DS18B20.
- Naprogramovat obsluhu teplotního čidla DS18B20.
- Zobrazit naměřenou teplotu na html stránce www serveru.

3 Vývojový kit EvB 4.3

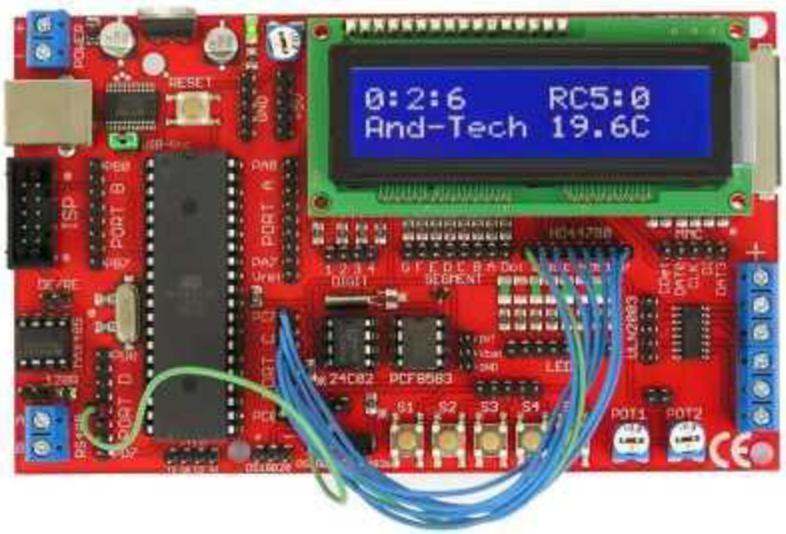
3.1 Základní údaje o vývojovém kitu EvB 4.3 v 4

Pro realizaci bakalářské práce byl využit vývojový kit EvB 4.3 v 4 s osazeným mikroprocesorem ATmega 32. Mikroprocesor je zasazen do patice, z níž jsou vyvedeny jednotlivé piny, což je zřetelné z obrázku 3.1.

Napájení kitu se uskutečňuje pomocí USB konektoru, který dodává napájecí napětí V_{cc} (5V) mikroprocesoru, a také toto napětí přivádí na piny umístěné nad paticí s mikroprocesorem s označením +5V. Tyto piny slouží jako napájení připojených periferií k mikroprocesoru.

Další možností napájení je připojení externího napájecího zdroje na napájecí svorkovnici. Tato svorkovnice se nachází vedle USB portu. Avšak při použití tohoto způsobu napájení je nutné odstranit zelený jumper zobrazený na obrázku 3.1, a také na obrázku 3.2 (pole G2).

Spolu s mikroprocesorem obsahuje kit i řadu periferií, např. modrý LCD display, teplotní čidlo DS18B20, 8 LED, čtečku SD/MMC karet aj. Piny jednotlivých periferií je nutno pomocí kabelů propojit s I/O piny mikroprocesoru. K propojení mikroprocesoru s periferiemi jsou k dispozici 4 porty, každý se řadou 8 I/O pinů.



Obrázek 3.1: Vývojový kit EvB 4.3 v 4. Zdroj: [1]

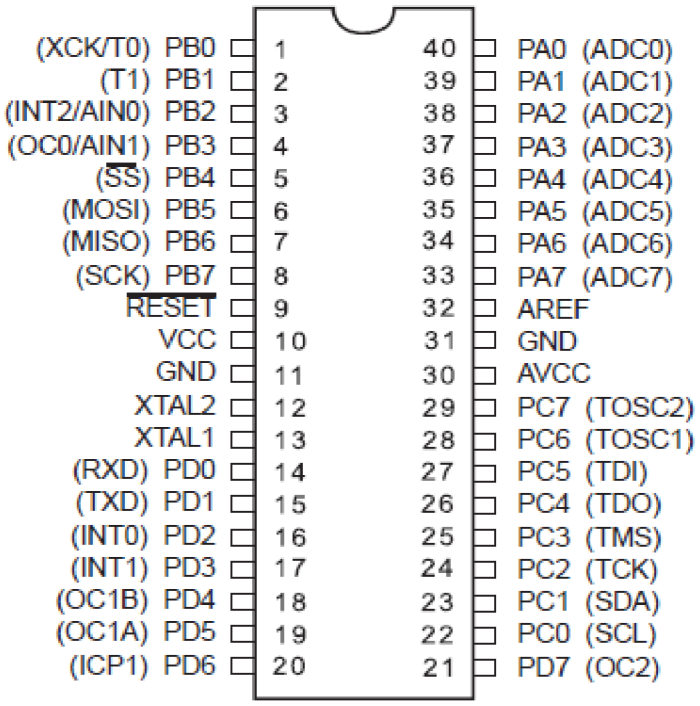
ISP port umístěný v levé části vývojového kitu (viz obrázek 3.1), je určen k programování mikroprocesoru pomoci SPI rozhraní. Schéma zapojení komunikace ISP portu s mikroprocesorem ATmega je zřejmé z přílohy 1 v poli C3.

4 ATmega 32

4.1 Základní údaje o mikroprocesoru ATmega 32

Mikroprocesor ATmega 32 je výrobek firmy Atmel. Je osazen ve vývojovém kitu EvB 4.3 v 4 a jeho pouzdro obsahuje 40 vývodů, jejichž pozice a popis jsou patrné z obrázku 4.1.

Velikost paměti RAM jsou 2 kB, vnitřní frekvence procesoru je až 16 MHz a programovací paměť FLASH má velikost 32 kB. Mikroprocesor má garantovanou funkčnost 20 let v prostředích s teplotou okolí 85°C a až 100let při provozu ve 20°C. Maximální počet přepsání paměti FLASH je 10000, poté dojde k jejímu znehodnocení. [3]



Obrázek 4.1: ATmega 32 popis pinů. Zdroj: [3]

4.2 Obsluha portů

Porty jsou 8 bitové, příslušné piny jsou číslovány od 0 do 7. Při práci s mikroprocesorem je nutné nejprve nastavit jednotlivé piny, či celé porty A až D mikroprocesoru jako vstupní či výstupní. Tuto operaci umožňuje registr **DDR** (Data Direction Register), který, jak je již z názvu patrné, ovládá směr toku dat na jednotlivých pinech portů. Součástí příkazu **DDR** musí být koncové písmeno portu, jež se má nastavit.

Např.:

```
DDRY = 0b11111111;      port Y je nastaven jako výstup
DDRY = 0b00000000;      port Y je nastaven jako vstup
```

Kde Y je zástupné označení portů A,B,C nebo D.

Při nastavování jednotlivých pinů jako vstup/výstup je nutné správně přiřadit bitové hodnoty jednotlivým pinům, kdy PORT A0 je bitově indexován jako 0b0000000X a PORT A7 má index 0bX0000000. Kde X = 0/1.

Pokud jsou piny nastaveny jako vstup/výstup, je možné posílat/přijímat data. Při práci s porty jsou důležité příkazy **PORT** a **PIN**:

PORTY – je příkaz, který pošle jemu přiřazenou hodnotu na výstupní port Y. Tímto příkazem se dají obsluhovat i jednotlivé piny, opět pomocí správného indexu přiřazené hodnoty tomuto portu.

Např.:

```
int x = 0b00101010;   (stejné jako 0x2A)
PORTA = x;            (nebo PORTA = 0x2A;)
```

PINY – je příkaz, který čte aktuální v stav (hodnotu) na portu Y, což při programování znamená, že se v programu proměnné přiřadí tento příkaz.

Např.:

```
int x = PINA;
```

Těmito třemi příkazy (registry) je tedy možné snadno obsluhovat piny, a tím i jednoduché periferie, jako např. DS18B20, u něhož je ovšem důležité správné časování, což je podrobněji popsáno v 5. kapitole, a také v kapitole 9.5.

4.3 SPI rozhraní

Z předchozí podkapitoly je tedy zřejmé, jakým způsobem se dají obsluhovat jednoduché periferie. Protože však ethernetový modul ENC28J60 je zařízení, které v sobě obsahuje vlastní mikroprocesor, je třeba ke komunikaci využít SPI rozhraní.

SPI je zkratka Serial Peripheral Interface, což je rozhraní používané ke komunikaci mezi mikroprocesory či jinými, složitějšími zařízeními.

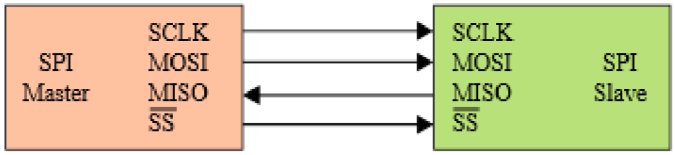
SPI rozhraní má vždy dva komunikační kanály MISO, MOSI a dále dva signály, které obsluhují komunikaci, což jsou piny $\overline{SS}$ a SCLK portu B. [4]

MISO: Master Input, Slave output, což je komunikační kanál, při kterém Master (mikroprocesor) přijímá informace (data) ze Slave zařízení (periferie).

MOSI: Master Output, Slave input, což je komunikační kanál, na kterém Master vysílá informace (data) do Slave zařízení.

$\overline{SS}$: (Slave select), tento signál řídí komunikaci. Je-li pin SS v logickém stavu 0, může probíhat příjem i vysílání dat. Je-li v logickém stavu 1, komunikace je zastavena.

SCLK: Zaručuje, že data budou přenesena správně, neboť při přijímání a vysílání dat je potřeba synchronizovat časovače Master i Slave zařízení, což vykonává tento signál. Data jsou posílána v časových sekvencích, které určuje perioda tohoto signálu.



Obrázek 4.2: Znázornění SPI rozhraní Master, Slave. Zdroj: [4]

Mikroprocesor ATmega 32 má SPI komunikaci zprostředkovanou pouze portem B, kde:

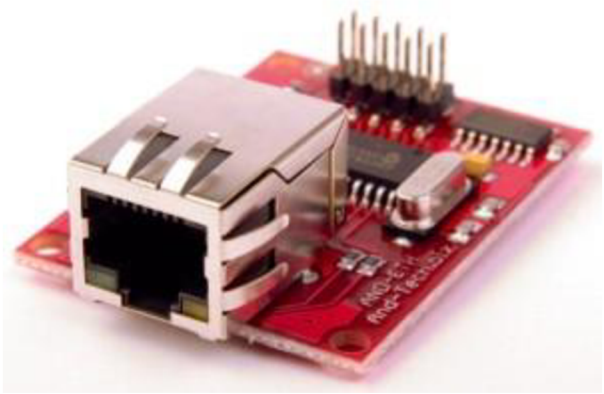
PIN7 = SCLK (v datasheetu ATmega 32 pojmenovaný jako SCK)
PIN6 = MISO
PIN5 = MOSI
PIN4 = $\overline{SS}$.

Na stejném komunikačním principu funguje i programování mikroprocesoru ATmega 32, ovšem při vkládání programu do mikroprocesoru se ATmega jeví jako Slave a programátor má funkci Master. [4]

5 ENC28J60

5.1 Základní údaje o ENC28J60

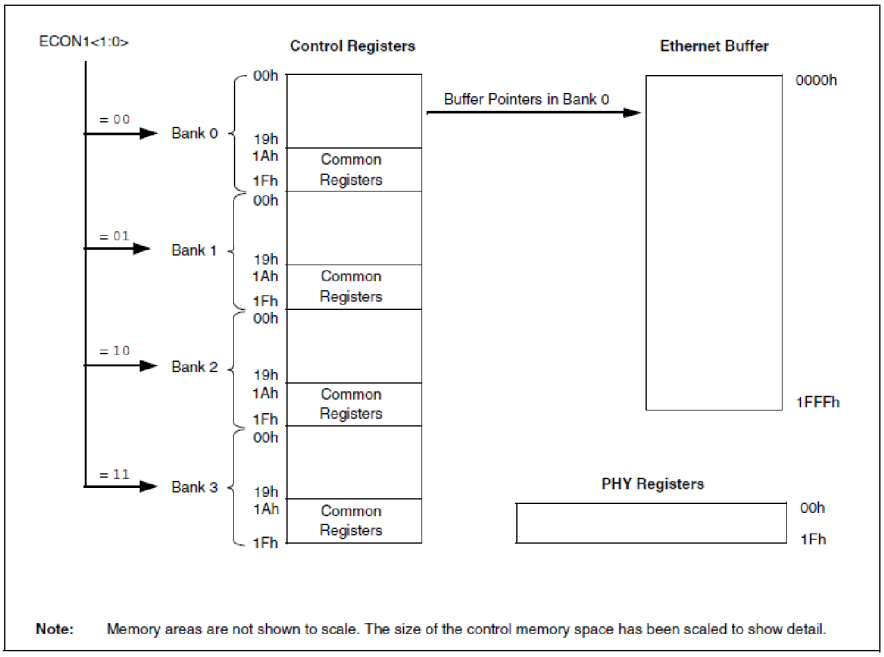
Tento ethernetový konektor je výrobek firmy Microchip Technology, komunikace s mikroprocesorem se uskutečňuje pomocí SPI (Serial Peripheral Interface) rozhraní.



Obrázek 5.1: Ethernetový modul ENC28J60. Zdroj: [5]

Úkolem ENC28J60 je zpracovat obdržené informace ze sítě LAN a poté je pomocí SPI rozhraní předat mikroprocesoru, který je naprogramovaný jako www server a jeho obsluha. Kromě samotných přijatých dat (packetů) si mikroprocesor a ENC28J60 vyměňují přes SPI rozhraní informace o stavech a registrech.

Pro komunikaci pomocí SPI rozhraní je tento modul vybaven externím oscilátorem s frekvencí 25 MHz. Konektor je také vybaven dvěma indikačními LED, které znázorňují tok dat a aktivitu síťě.

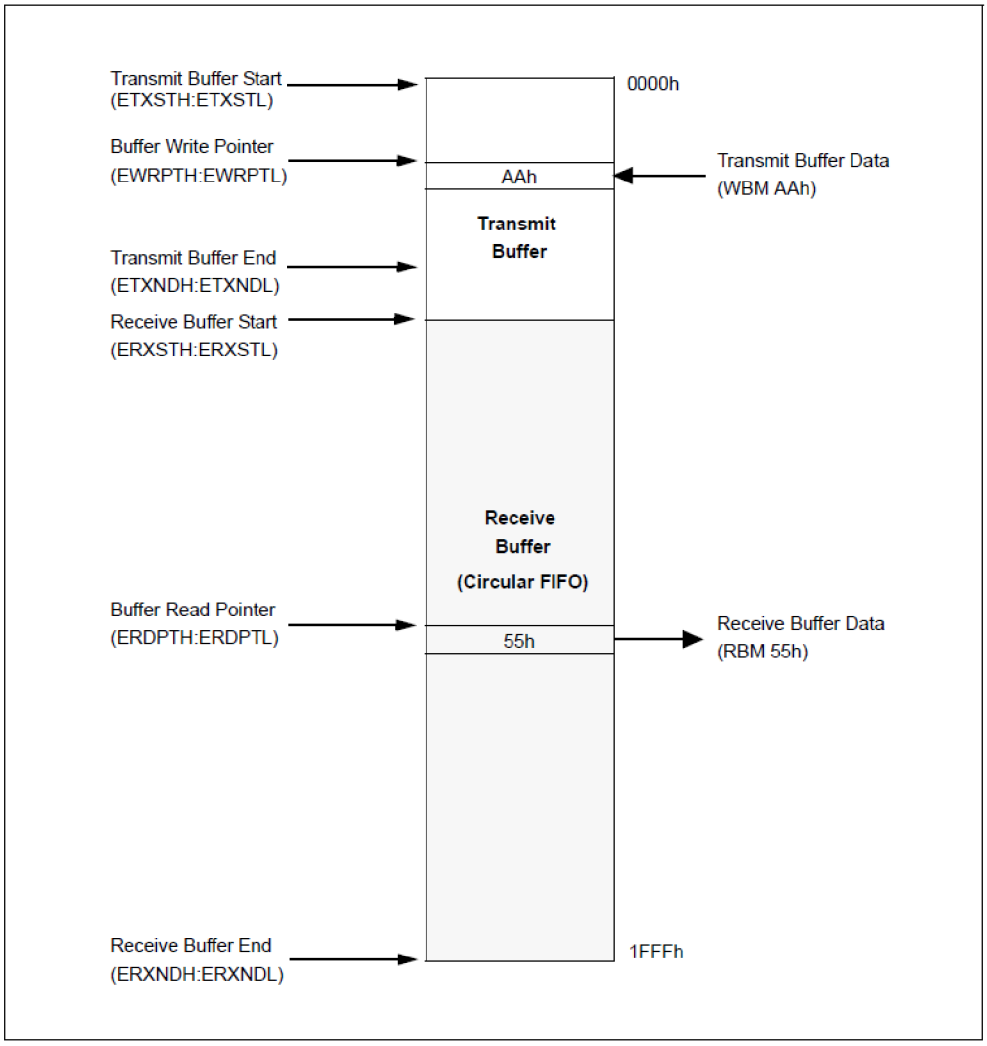


Obrázek 5.2: Organizace paměti ENC28J60. Zdroj: [6]

5.2 *Organizace paměti*

Při obdržení dat ze sítě, je nutné přijatá data uložit do paměti, a v případě splnění podmínek implementovaného filtru, jsou dále odeslána pomocí SPI rozhraní do mikroprocesoru. K tomuto účelu je ENC28J60 vybaven pamětí zvanou Ethernet buffer, která se dělí na dvě hlavní části: Transmit (dále jen odesílací) Buffer a Recieve (dále jen přijímací) Buffer. Celková velikost bufferu je 8Kb.

Jak je vidět na obrázku 5.2, čtení dat z bufferu zajišťují registry umístěné v bance 0 kontrolního registru.



Obrázek 5.3: Organizace bufferu. Zdroj: [6]

Další část paměti ENC28J60 tvoří kontrolní registry, kterými se ovládají veškerá nastavení a aktivity modulu ENC28J60. Tyto registry jsou rozděleny do čtyř bank, mezi kterými se přepíná registrem ECON1. Do těchto registrů se ukládá např. MAC adresa, dílčí data a nastavení potřebná ke správné funkci modulu.

Poslední částí paměti jsou PHY registry, kterými se ovládají 2 signalizační LED vestavěné v ethernetovém konektoru. Těmto diodám se dají pomocí PHY registrů nastavit různé režimy blikání.

5.3 *Kontrolní registry*

Jak bylo již zmíněno výše, obsahuje paměť ethernetového modulu několik částí, z nichž jednou částí jsou kontrolní registry (tyto jsou vidět na obrázku 5.4). K realizaci web serveru jsem využil již vyřešeného a naprogramovaného příkladu obsluhy ethernetového modulu, který jsem stáhl ze stránek tuxgraphics.com.

Bank 0		Bank 1		Bank 2		Bank 3	
Address	**Name**	**Address**	**Name**	**Address**	**Name**	**Address**	**Name**
00h	ERDPTL	00h	EHT0	00h	MACON1	00h	MAADR5
01h	ERDPTH	01h	EHT1	01h	Reserved	01h	MAADR6
02h	EWRPTL	02h	EHT2	02h	MACON3	02h	MAADR3
03h	EWRPTH	03h	EHT3	03h	MACON4	03h	MAADR4
04h	ETXSTL	04h	EHT4	04h	MABBIPG	04h	MAADR1
05h	ETXSTH	05h	EHT5	05h	—	05h	MAADR2
06h	ETXNDL	06h	EHT6	06h	MAIPGL	06h	EBSTSD
07h	ETXNDH	07h	EHT7	07h	MAIPGH	07h	EBSTCON
08h	ERXSTL	08h	EPMM0	08h	MACLCON1	08h	EBSTCSL
09h	ERXSTH	09h	EPMM1	09h	MACLCON2	09h	EBSTCSH
0Ah	ERXNDL	0Ah	EPMM2	0Ah	MAMXFLL	0Ah	MISTAT
0Bh	ERXNDH	0Bh	EPMM3	0Bh	MAMXFLH	0Bh	—
0Ch	ERXRDPTL	0Ch	EPMM4	0Ch	Reserved	0Ch	—
0Dh	ERXRDPTH	0Dh	EPMM5	0Dh	Reserved	0Dh	—
0Eh	ERXWRPTL	0Eh	EPMM6	0Eh	Reserved	0Eh	—
0Fh	ERXWRPTH	0Fh	EPMM7	0Fh	—	0Fh	—
10h	EDMASTL	10h	EPMCSL	10h	Reserved	10h	—
11h	EDMASTH	11h	EPMCSH	11h	Reserved	11h	—
12h	EDMANDL	12h	—	12h	MICMD	12h	EREVID
13h	EDMANDH	13h	—	13h	—	13h	—
14h	EDMADSTL	14h	EPMOL	14h	MIREGADR	14h	—
15h	EDMADSTH	15h	EPMOH	15h	Reserved	15h	ECOCON
16h	EDMACSL	16h	Reserved	16h	MIWRL	16h	Reserved
17h	EDMACSH	17h	Reserved	17h	MIWRH	17h	EFLOCON
18h	—	18h	ERXFCON	18h	MIRDL	18h	EPAUSL
19h	—	19h	EPKTCNT	19h	MIRDH	19h	EPAUSH
1Ah	Reserved	1Ah	Reserved	1Ah	Reserved	1Ah	Reserved
1Bh	EIE	1Bh	EIE	1Bh	EIE	1Bh	EIE
1Ch	EIR	1Ch	EIR	1Ch	EIR	1Ch	EIR
1Dh	ESTAT	1Dh	ESTAT	1Dh	ESTAT	1Dh	ESTAT
1Eh	ECON2	1Eh	ECON2	1Eh	ECON2	1Eh	ECON2
1Fh	ECON1	1Fh	ECON1	1Fh	ECON1	1Fh	ECON1

Obrázek 5.4: Znázornění kontrolních registrů. Zdroj: [6]

5.3.1 *ECON1*

Registr ECON1 (Ethernet control register), slouží k přepínání mezi jednotlivými bankami modulu. Proto je v každé bance umístěn na adrese 0x1F. ECON1 slouží jako rozhraní mezi bankami.

ECON1: ETHERNET CONTROL REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TXRST	RXRST	DMAST	CSUMEN	TXRTS	RXEN	BSEL1	BSEL0
bit 7							bit 0

Obrázek 5.5: Registr ECON1 a jeho bity. Zdroj: [6]

TXRST – odeslání/reset odesílacího registru
RXST – zapnutí/reset přijímacího bufferu
DMAST – Start a Busy status
CSUMEN – DMA checksum enable
TXRTS – předávání packetů
RXEN – bit který určuje zda data prošla či neprošla filtrem
BSEL0 a BSEL1 – bity přepínající mezi bankami

Na obrázku 5.5 je znázorněno 8 bitů kontrolního registru ECON1. Přepínání mezi bankami však zprostředkovávají pouze dva bity a to bit 0 a bit 1, jejichž kombinacím jsou přiřazeny všechny čtyři banky:

Bit [hodnota]	**BSEL0 [0]**	**BSEL0 [1]**
BSEL1 [0]	Bank 0	Bank 1
BSEL1 [1]	Bank 2	Bank 3

Tabulka 5.1: Kombinace nastavení bitů registru ECON1. Zdroj: [6]

5.4 Odesílání/přijímání dat po síti

5.4.1 Inicializace

Než se začne samotný proces odesílání a přijímání dat a jejich obsluha, je nejprve nutné ethernetový modul inicializovat, což znamená nastavit MAC adresu ENC28J60 a podle typu aplikace správně vyplnit potřebné kontrolní registry. Inicializace má několik fází:

- Inicializuje přijímací buffer nastavením hodnot ERXSTH, ERXSTL a ERXNDH, ERXNDL registrů. Pozice a význam těchto registrů je zobrazena na obrázku 5.3. ERXST registry tedy uvozují přijímací buffer a ERXND registry zakončují tuto část bufferu. Rozdíl hodnot ERXND a ERXST je velikost přijímacího bufferu. Přijatá data se zapisují na pozice začínající adresou ERXST a končící ERXND.
- Inicializace odesílacího bufferu je jednodušší, neboť místo, které zůstalo volné v bufferu po inicializaci přijímací části je přiřazeno jako odesílací buffer.
- Dále se v části inicializace dá nastavit přijímací filtr, např. packet, který neobsahuje MAC adresu ENC28J60 bude automaticky ignorován, atp.
- Ethernetovému modulu je také nutné přiřadit MAC adresu. MAC adresa se ukládá do registrů MAADR1 až MAADR6, přičemž je nutné zohlednit pořadí těchto registrů v BANK3. Z obrázku 5.4 je patrné pořadí těchto registrů.

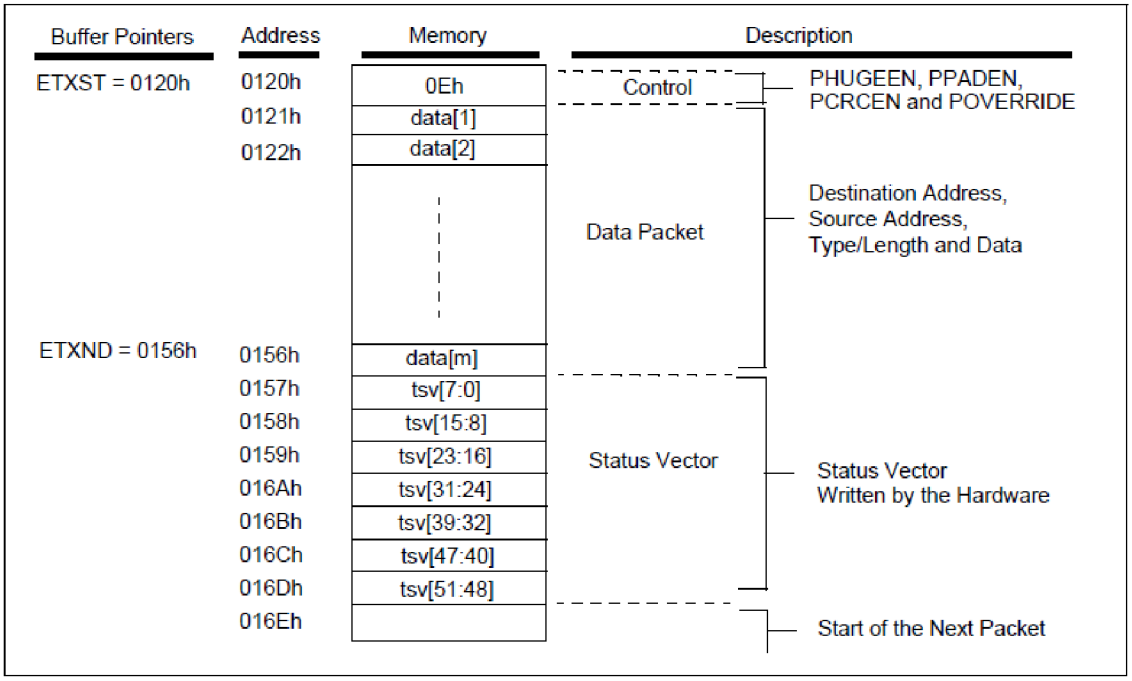
- Posledním úkolem inicializačního procesu je nastavení PHY registrů, které obsluhují signalizační LED. Tyto registry jsou popsány v kapitole 5.4.

5.4.2 Odesílání dat

ENC28J60 automaticky generuje úvodní rámec packetu (fyzickou vrstvu), a také kontrolní součet. Ostatní data packetu musí obstarat a vyplnit master (ATmega 32).

Jak již bylo popsáno v kapitole inicializace, odesílací část bufferu je ta část, jež nebyla obsazena přijímacím bufferem a je uvozena registrem ETXST a zakončena hodnotou registru ETXND. Mezi hodnotami těchto registrů se pohybuje ukazatel (pointer) EWRPT, který zaručuje zápis dat do tohoto registru.

Samotné vyslání obsahu odesílacího bufferu je spuštěno nastavením bitu TXRTS v registru ECON1 na logickou 0.



Obrázek 5.5: Odesílací buffer. Zdroj: [6]

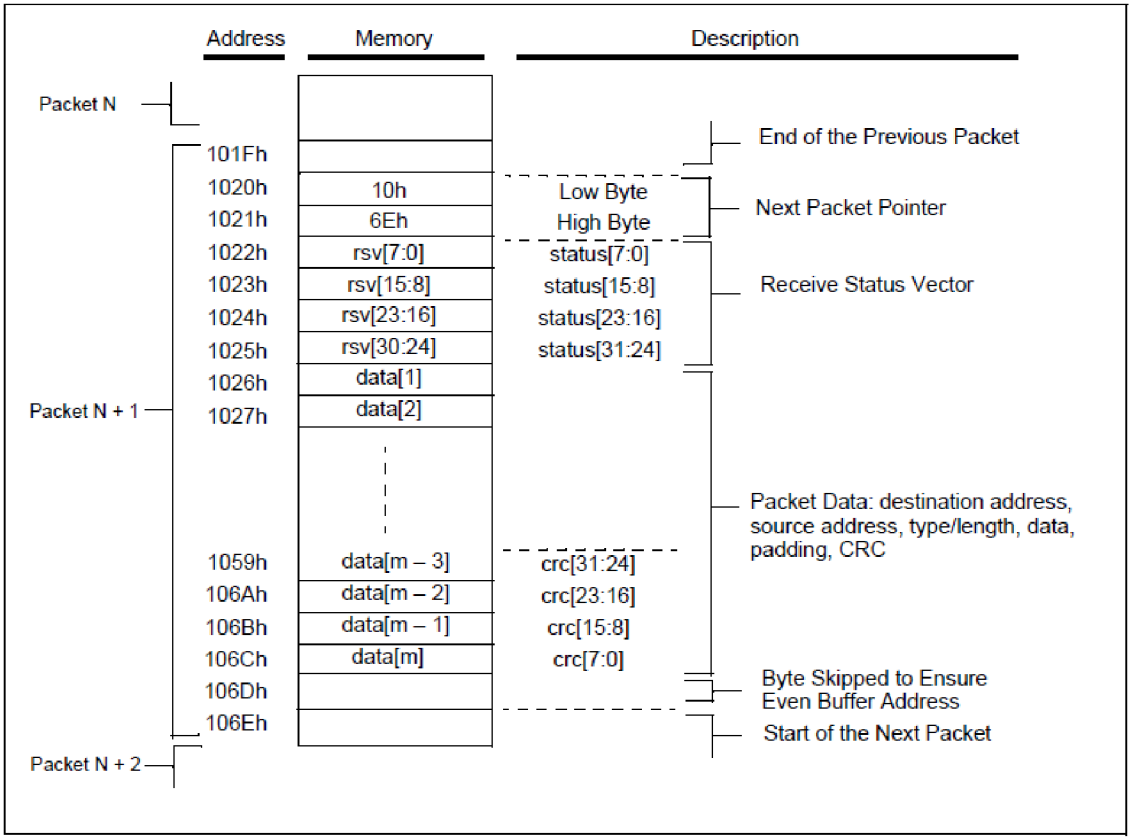
5.4.3 Přijímání dat

V kapitole 5.3.1 je popsána inicializace, ze které vyplývá, že paměť uvolněná pro přijímací buffer je uložena v registrech ERXST až ERXND, dále se nastavoval přijímací filter a MAC adresa, což jsou nutné předpoklady k úspěšnému fungování komunikace.

Jsou-li přijatá data validní, úspěšně projdou filtrem, čímž se v bitu RXEN registru ECON1 nastaví logická úroveň 1, nastaví se počáteční a koncový ukazatel packetu. Validní packet je po těchto procedurách zapsán do přijímacího bufferu a hodnota registru EPKTCNT se inkrementuje o 1.

EPKTCNT – registr do něhož se zapisuje počet přijatých packetů do bufferu.

ERXWRPT – je ukazatel na aktuální položky přijatého packetu.



Obrázek 5.6: Přijímací buffer. Zdroj: [6]

5.5 *PHY registry*

Běžně dostupné síťové karty jsou kromě mnoha jiných funkcí vybaveny také světelnou signalizací aktivity dat na lince. K tomuto účelu je ethernetový modul ENC28J60 vybaven dvěma indikátory LED nesoucí označení LED A (zelená) a LED B (oranžová). Tyto LED mají několik funkčních módů, které je rozsvěcují v závislosti na zobrazení požadované aktivity. Nejběžnější nastavení těchto LED je takové, že LED A zobrazuje, zda je připojený konektor RJ45 aktivní (zda je zapojen do lokální sítě) a LED B signalizuje aktivitu na lince (přijímání / odesílání packetů).

PHY registry jsou tedy registry, pomocí nichž se nastavuje či kontroluje komunikace, např. bit PDPXMD registru PHCON1 kontroluje konfiguraci komunikaci, zda je half/full duplex.

Dalším řídícím registrem PHY registrů je PHLCON. Tento registr určuje nastavení módu blikání LED A a B. Jeho znázornění je na obrázku 5.7.

R/W-0	R/W-0	R/W-1	R/W-1	R/W-0	R/W-1	R/W-0	R/W-0
r	r	r	r	LACFG3	LACFG2	LACFG1	LACFG0
bit 15							bit 8

R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-1	R/W-x
LBCFG3	LBCFG2	LBCFG1	LBCFG0	LFRQ1	LFRQ0	STRCH	r
bit 7							bit 0

Obrázek 5.7: PHLCON registr. Zdroj: [6]

Bity LACFG3 až LACFG0 nastavují konfiguraci LED A, bity LBCFG3 až LBCFG0 nastavují konfiguraci LED B.

5.5.2 *half/full duplex mód*

Half-duplex (poloviční duplex)

Obrázek 5.8: Half-duplex komunikace. Zdroj: [7]

Obě strany mohou přijímat i vysílat, avšak nikoli současně – v každý jednotlivý okamžik probíhá přenos pouze jedním směrem (podobné simplexu). Při této komunikaci jsou však na rozdíl od simplexu využívány dvě frekvence. Na jedné frekvenci se vysílá, na druhé přijímá. Příkladem takové komunikace je vysílání radiostanic (vysílaček) přes opakovač; typické pro half-duplexní spojení je používání signalizace „přepínám“.[7]

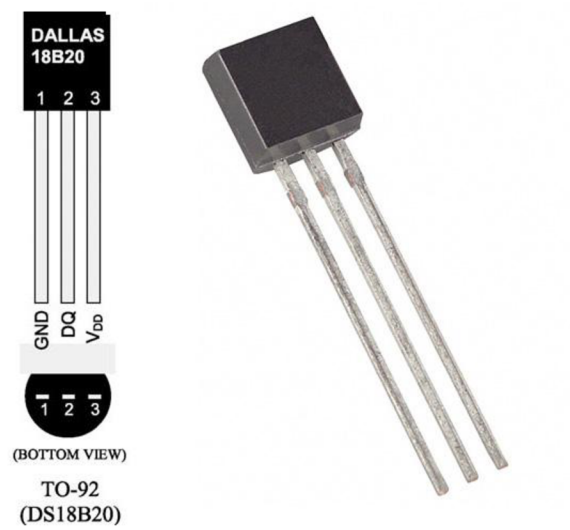
Full-duplex (plný duplex)

Obrázek 5.9: Full-duplex komunikace. Zdroj: [7]

U plného duplexu může obousměrná komunikace probíhat současně. Příkladem takové komunikace může být běžný telefonický hovor, kdy obě zúčastněné strany mohou hovořit zároveň.

Plný duplex v Ethernetu funguje tak, že dva páry vodičů v ethernetovém kabelu jsou využívány pro odesílání rámců a dva páry jsou využívány pro příjem. [7]

6 Teplotní čidlo DS18B20



Obrázek 6.1: Teplotní čidlo DS18B20. Zdroj: [6]

Teplotní čidlo DS18B20 je výrobek firmy Maxim integrated, který umožňuje časově nespojité měření teploty v rozsahu od -55 °C až do +125 °C. Čidlo umožňuje měřit teplotu s rozlišením až 2^{-4} °C, což je 0,0625 °C. Pro realizaci této práce je však dostatečné rozlišení 2^{-2}, což je 0,25°C. Každému vyrobenému čipu DS18B20 je přiděleno unikátní 64 bitové sériové číslo. Toto číslo umožňuje identifikaci jednotlivých čidel při sériovém propojení více čidel. [9]

Zařízení DS18B20 umožňuje dva módy napájení.

- Parazitní: napájení čidla je zprostředkováno pouze pomocí signálu (1-Wire sběrnice) [9].
- Externí napájení: čidlo je napájeno přídavným zdrojem napětí (piny 1 a 3 dle obrázku 6.1) [9].

6.1 Princip měření DS18B20

Princip snímání teploty je ve zkratce takový, že uvnitř čidla jsou dva oscilátory. Jeden je s nízkým teplotním koeficientem a druhý naopak s vysokým. Jakmile se dá po sběrnici čidlu příkaz k zahájení převodu (měření) teploty, spustí se čítání obou těchto oscilátorů. Tato akce trvá (podle rozlišení) až 700ms a jejím výsledkem je 12b hodnota odpovídající aktuální teplotě. Ta je poté uložena ve vnitřních registrech obvodu a čeká na její vyčtení z nadřazeného obvodu. [8]

Měření teploty se dá rozdělit do čtyř kroků:

1. Příprava čipu na měření teploty
2. Měření teploty
3. Příprava čipu ke čtení změřené teploty
4. Čtení teploty

Vývojový diagram měření a vyčtení teploty viz příloha 7.

6.1.1 Příprava čipu na měření teploty

6.1.1.1 Inicializace

Veškeré operace na „1-Wire“ sběrnici jsou nutné zahájit tímto krokem. Inicializační sekvence se skládá z tzv. reset pulsu vyslaného masterem (ATmega 32) do slave zařízení (DS18B20). Po tomto pulsu následuje tzv. presence puls, který vyšle slave zařízení zpět. Presence puls umožňuje masteru rozeznat, zda je zařízení připojené a připravené k provozu.

Inicializační funkce je volána i v kroku č. 3, kdy je nutné po změření teploty znovu inicializovat DS18B20. [9]

6.1.1.2 Skip ROM

Po přijetí Presence pulsu je dalším krokem příkaz Skip ROM, který připraví (nastaví adresu) DS18B20 na následný příkaz k měření teploty. Tento příkaz reprezentuje hexadecimální číslo 0xCC, které je odesláno na vstup teplotního čidla, čímž provede Skip ROM. Příkaz skip ROM, při zapojení více čidel za sebou, připraví k měření všechna čidla.

Příkaz Skip ROM je volán i v kroku č. 3 ihned po inicializaci. [9]

6.1.2 Měření teploty

Měření teploty se provede odesláním hexadecimálního čísla 0x44 na vstup teplotního čidla. Pro 12ti bitové rozlišení trvá toto měření cca 750ms. Po tuto dobu se měří teplota a zapisuje se do 12ti bitového teplotního registru. Při režimu napájení z externího zdroje lze během teplotní konverze vyčítat z příslušného pinu log. úroveň 0, po dokončení teplotní konverze je stav log. 1. Při parazitním režimu napájení nelze tuto vlastnost využít.

Předchází – li příkazu měření teploty příkaz Skip ROM, spustí se měření pro všechna čidla připojená na „1-Wire“ sběrnici. [9]

6.1.3 Příprava čipu ke čtení naměřené teploty

V tomto kroku se provede inicializace čipu a příkaz Skip ROM [0xCC] [9].

6.1.4 Čtení teploty

Vyčtení naměření teploty je prováděno pomocí příkazu Read scratchpad, který spustí vysílaní 0. až 8. bajtu (viz obrázek 6.2), z nichž první dva bajty obsahují hodnotu naměřené teploty.

Naměřená teplota je uložená v doplňkovém kódu. Prvních 5 bitů horního bajtu je znaménkových. Pro naměřenou kladnou hodnotu teploty je na pozicích znaménkových bitů zapsána 0, při naměřené záporné hodnotě jsou bity 15 až 11 naplněny hodnotou 1.

Příkaz Read scratchpad je reprezentován jako hexadecimální číslo 0xBE. [9]

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	**BIT 14**	**BIT 13**	**BIT 12**	**BIT 11**	**BIT 10**	**BIT 9**	**BIT 8**
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

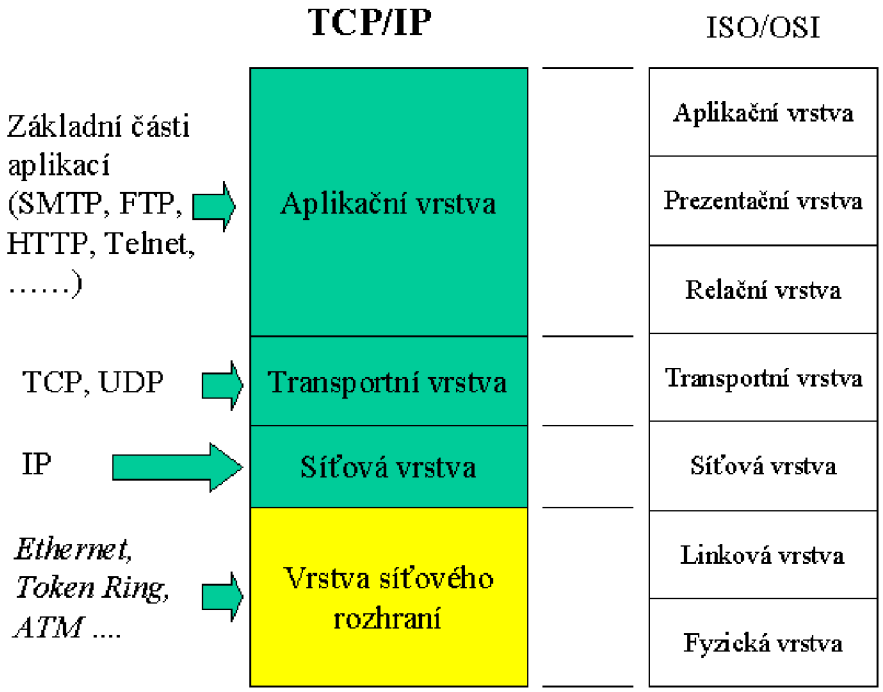
Obrázek 6.2: Znázornění Teplotních registrů. Zdroj: [9]

7 Wake on Lan

7.1 Ethernetová komunikace

Po lokální ethernetové síti se přenášejí data, tzv. packety, které se skládají z hlavičky a těla. V hlavičce jsou uložena různá kontrolní data, z nichž nejdůležitější jsou např. velikost packetu, zdrojová adresa, cílová adresa a kontrolní součet (CRC). V těle packetu jsou pak už přímo data, která jsou určena k přenesení ze zdrojové na cílovou stanici. Těchto packetů může být v jednom přijímacím bufferu několik, jelikož model ISO/OSI obsahuje 7 vrstev znázorněných v příloze 6. Každá vrstva obsahuje své packety, např. Transportní vrstva obsahuje TCP (Transmission Control Protocol), UDP (User Datagram Protocol) apod..

Nejběžněji používaným protokolem je TCP/IP protokol, který obsahuje pouze 4 ze 7 vrstev. TCP/IP protokol slouží majoritně jako zprostředkování dat uživatelům internetu.



Obrázek 7.1: Srovnání TCP/IP modelu a ISO ISI. Zdroj: [10]

7.2 Princip Wake on Lan

Wake on Lan je schopnost systému BIOS zapnout počítač, pokud síťová karta obdrží tzv. „Magic packet“ který obsahje MAC adresu buzeného počítače. V systému BIOS je však nutné tuto funkci nejprve povolit, neboť základní nastavení počítače tuto funkci zakazuje.

Povolením funkce Wake on Lan se udržuje síťová karta při vypnutém počítači stále pod napětím a přijímá ethernetová data, s nimiž však dále nepracuje. Tento stav se však změní, obdrží-li do přijímacího bufferu „Magic packet“. Po jeho obdržení vyšle síťová karta příkaz k zapnutí počítače.

7.3 UDP (User datagram packet)

Nejpoužívanějším packetem zprostředkujícím „Magic packet“ je UDP, který se nachází v Transportni vrstvě TCP/IP protokolu. Na rozdíl od TCP packetu však neposkytuje jistotu, že vyslaný packet bude správně doručen. Díky této vlastnosti je přesnější označení datagram, neboť packet poskytuje jistotu, že bude doručen s nezměněným obsahem.

UDP packet je součástí rodiny IP protokolů, z čehož vyplývá, že adresace probíhá pomocí IP adres. Každý UDP packet obsahuje v hlavičce zdrojový port, cílový port, svoji délku a kontrolní součet, tělo pak přenáší data. Jakožto transportní port, může UDP přenášet mnoho různých služeb (http, ftp, imap, pop3,…) aplikační vrstvy. Aby však bylo možné tyto služby od sebe odlišit v rámci jedne IP adresy, je nutné nastavit zdrojový a cílový port, neboť by jinak docházelo ke kolizím. Pro zjednodušení byl vytvořen Seznam čísel portů TCP a UDP, který je dostupný na internetu. Z tohoto seznamu je snadno dohledatelné, k čemu jsou jednotlivé porty využívány. [11]

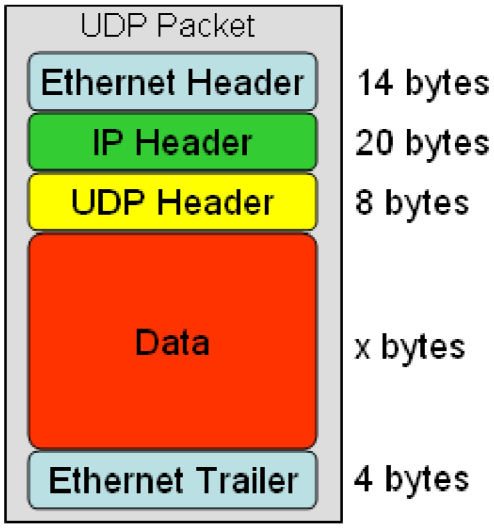
Formát IP datagramu

Bajty	0		1	2	3
Bajt 0 až 3	Verze	Délka hl.	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				

Obrázek 7.3: IP hlavička. Zdroj: [12]

+	bity 0 - 15	16 - 31
0	zdrojový port	cílový port
32	délka	kontrolní součet
64	data	

Obrázek 7.3: UDP packet. Zdroj: [11]



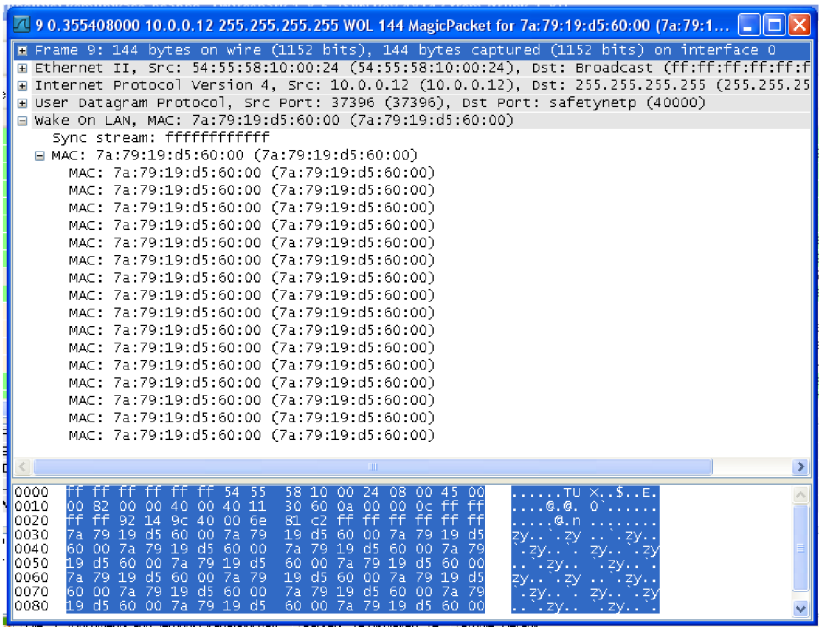
Obrázek 7.4: UDP packet včetně IP hlavičky. Zdroj: [13]

7.4 *Magic packet*

Účel Magic packetu je zapnout vybraný počítač s danou MAC adresou, což vyžaduje vyslání toho packetu po celé lokální síti a jeho přijmutí všemi síťovými kartami. Tento děj je zaručen vysláním rezervované MAC adresy 0xff 0xff 0xff 0xff 0xff 0xff (zápisem z commandline FF:FF:FF:FF:FF:FF), což je tzv. Broadcast který zaručí, že UDP protokol přijme každá aktivní síťová karta v lokální síti.

Za Broadcastem se nachází 16 opakování MAC adresy počítače, který chceme zapnout.

Z obrázku 7.2 je patrné, že UDP datagram je velmi jednoduchý, což je pro Magic packet vhodný prostředek. [14]



Obrázek 7.3: Magic packet zobrazený pomocí programu WireShark

8 www server

Velmi rozšířeným použitím dnešních počítačů je www server, což je služba, která umožňuje, pomocí grafického rozhraní (html stránky), odesílat či přijímat požadavky po internetové síti. Ke komunikaci mezi klientem a serverem slouží http protokol.

8.1 HTTP protokol

HTTP protokol slouží ke komunikaci mezi klientem (webovým prohlížečem) a serverem. Protokol používá obvykle port TCP/80, verze 1.1 protokolu je definována v RFC 2616. Tento protokol je spolu s elektronickou poštou tím nejvíce používaným a zasloužil se o obrovský rozmach internetu v posledních letech.

HTTP používá jako některé další aplikace tzv. jednotný lokátor prostředků (URL - Uniform Resource Locator), který specifikuje jednoznačné umístění nějakého zdroje v Internetu.

Protokol funguje způsobem dotaz-odpověď. Uživatel (pomocí programu, obvykle internetového prohlížeče) pošle serveru dotaz ve formě čistého textu, obsahujícího označení požadovaného dokumentu, informace o schopnostech prohlížeče apod. Server poté odpoví pomocí několika řádků textu popisujících výsledek dotazu (zda se dokument podařilo najít, jakého typu dokument je atd.), za kterými následují data samotného požadovaného dokumentu. [15]

Ukázka dotazu vyslaného klientem:

```
GET /tajne/ HTTP/1.1
Host: 10.0.0.12
```

9 Praktická realizace www serveru s funkcí WOL

9.1 Uživatelská obsluha www serveru

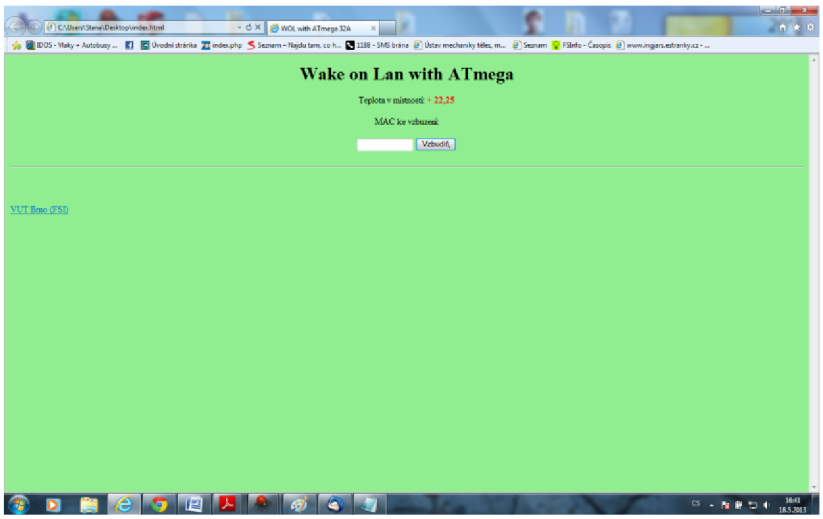
K zobrazení www stránky webového serveru je nutné zadat do URL adresy IP adresu webového serveru (ethernetového modulu) následovanou znakem "/ " za který se napíše heslo. Heslo je realizováno pouze jako specifická URL adresa. URL adresa má tedy tvar např.:

http://10.0.0.12/tajne

Při vstupu na webovou stránku se zobrazí poslední hodnota změřené teploty. Aktualizace zobrazované teploty se provede příkazem refresh (F5) webového prohlížeče.

Zadávací políčko zobrazené na webové stránce slouží k zadání MAC adresy počítače, který se má vzbudit. Do tohoto políčka tedy uživatel zadává MAC adresu. Po stisknutí tlačítka „Vzbudit" se provede odeslání „Magic packetu" a počítač se zapne. Stránka se automaticky obnoví i s naměřenou teplotou.

Pokud si uživatel není jistý tím, jak se správně zadává URL adresa, stačí napsat do adresového řádku pouze IP adresu a zobrazí se nápověda, jak se adresa serveru zadává.



Obrázek 9.1 Zobrazení www stránky

9.2 Spouštění www serveru

K obsluze ethernetového modulu ENC28J60 a zprostředkování komunikaci mezi tímto modulem a mikroprocesorem ATmega 32 jsem použil projekt nacházející se na stránkách http://www.tuxgraphics.org/common/src2/article06061/.

Prvním krokem k úspěšnému rozkrytí kódu a následné modifikaci bylo zjištění, zda je tento kód použitelný pro aplikaci této bakalářské práce. Výchozím bodem se stala podmínka pro příkaz PING, kdy se dalo lehce ověřit, zda je program funkční či nikoliv.

Před samotným zkomprimováním a nahráním programu do mikroprocesoru bylo nutné provést některé syntaktické změny, a také bylo nezbytné nastavit správnou frekvenci mikroprocesoru.

Po provedení několika úspěšných testů („PINGŮ“) bylo dalším krokem nahrání celého projektu webserveru do mikroprocesoru. Tento program původně sloužil jako webový server, který disponoval grafickým rozhraním (html), a dále obsahoval funkci switch on/switch off, která zapínala či vypínala tranzistor pomocí UDP rozhraní. Při zprovoznění tohoto web serveru bylo ověřeno, že projekt je funkční a pro aplikaci na tuto práci bude vhodný.

9.3 Modifikace projektu

Prvním problémem byla správná modifikace podmínek, které sloužily zapínání a vypínání připojeného tranzistoru pomocí html stránky. Program se musel upravit tak, aby bylo možné tyto podmínky odstranit, neboť na stránce www serveru s funkcí Wake on Lan bude pouze vykreslovaná teplota se zadávacím políčkem.

Druhým modifikačním krokem byla změna odesílané html kódu klientovi do podoby zobrazené na obrázku 9.1. A také bylo nutné sjednotit názvy funkcí, což se ukázalo jako velmi problematický krok, který vyžadoval velmi pečlivý a systematický postup.[19]

9.3.1 Implementace měření teploty do projektu www serveru

Tvorba programu obsluhující teplotní čidlo DS18B20 probíhala odděleně od modifikace www serveru, neboť zprovoznění teplotního čidla bylo vyřešeno dříve.

Při řešení obsluhy teplotního čidla byl k ověření správné funkčnosti využit LCD display WC160-2A, který je součástí vývojového kitu. Na druhý řádek LCD byla vypisována naměřená teplota v binárním formátu s rozlišením 0,0625 °C. První řádek zobrazoval teplotu převedenou do desítkové soustavy s rozlišením 0,25°C.

Jakmile byla fáze změny odesílaného html kódu webovým serverem hotova, bylo možné vložit do výsledného projektu soubor a hlavičkový soubor obsluhy teplotního čidla DS18B20. Po úspěšné implementaci zdrojového kódu teplotního čidla byl projekt schopen zobrazovat, po přístupu na webovou stránku serveru, naměřenou teplotu v místnosti s čidlem DS18B20.

Důležitým krokem při packetu s vložením naměřené teploty byl fakt, že mikroprocesor obdrží data naměřená teplotním čidlem do paměti RAM. Což znamená, že packet obsahující html kód webové stránky se skládá z obsahu paměti FLASH a RAM mikroprocesoru.

Principem zobrazení teploty na webovou stránku je převod 2 bajtové reprezentace teploty na pole znaků. Toto pole je poté odesláno jako součást html kódu klientovi. [19],[18]

9.3.2 Implementace funkce Wake on Lan do projektu www serveru

Z kapitoly 9.1 je patrné, že výsledná stránka bude obsahovat zadávací políčko pro MAC adresy buzených počítačů. Tato schopnost je zcela jasná, neboť bez tohoto políčka by byl program neadaptabilní a změna MAC adresy buzeného počítače by vyžadovala přeprogramování mikroprocesoru.

9.3.2.1 Zadávací políčko

Zadávací políčko je realizováno pomocí následujícího html kódu:

```
<form method="post">
        <input type="text" size = "12" name="MACADDR">
        <input type ="submit" value="VZBUĎ">
</form>
```

Standardní html kód neumožňuje práci s hodnotami zadanými do zadávacího políčka, pouze umožňuje způsob předávání dat. Tento formát nastavuje přiřazení hodnoty "get"/"post" proměnné method.

"get": toto nastavení zapíše předaná data jakou součást URL adresy.
"post": toto nastavení nemění URL adresu stránky, pouze odešle packet, jehož datová část je uvozena klíčovým slovem POST, a na konci packetu je za proměnné a znakem "=" uložena posloupnost zadaných znaků.

V této bakalářské práci je využit způsob předávání dat metodou "post", kdy po stisknutí potvrzovacího tlačítka odešle klient na server již zmiňovaný packet. Tento packet se poté analyzuje a po nalezení a zpracování přijaté posloupnosti reprezentující MAC adresu buzeného počítače je poté sestaven „Magic packet“. [19]

9.3.2.2 Sestavení Magic packetu

Jak již bylo zmíněno v kapitole 7.4, „Magic packet“ je posílán pomocí IP protokolu obsahující UDP packet.

Sestavení „Magic packetu“ je realizován dvěma cykly for:

První cyklus má za úkol sestavit úvodní šestici dvouciferných, hexadecimálních čísel reprezentující broadcast MAC adresu, což znamená, že packet bude přijat všemi aktivními síťovými kartami připojenými na lokální síť. Broadcast MAC adresa se nachází v datové oblasti UDP packetu.

Druhý cyklus má za úkol vložit za broadcast MAC adresu 16 po sobě jdoucích MAC adres buzeného počítače.

9.4 Rozbor funkcí obstarávající síťovou komunikaci

Nejdůležitějšími funkcemi tohoto projektu jsou:

- fill_tcp_data_p
- fill_tcp_data
- print_webpage
- init_ip_arp_udp_tcp
- enc28j60_packet_send
- enc28j60_packet_receive
- enc28j60_init

9.4.1 Funkce fill_tcp_data_p

K tomu, aby server fungoval správně a odesílal reakce na klientovy požadavky, slouží tato funkce. Jejím úkolem je zapsat data určená k odeslání do pole, které reprezentuje odesílací buffer. Data jsou předávána v hlavičce funkce a její návratovou hodnotou je hodnota ukazatele, ukazujícího na poslední zapsaný bajt v poli.

Funkce pracuje s daty nacházející se v paměti FLASH, což znamená, že tuto funkce nelze použít k odeslání naměřené teploty, neboť ta se nachází v paměti RAM.

Jak již bylo zmiňováno v kapitole 8, dotazy a odpovědi mezi klientem a serverem probíhají ve formě textu a tudíž je k reprezentaci odesílacího bufferu použito pole znaků. [19]

9.4.2 Funkce fill_tcp_data

Tato funkce je založena na stejném principu, jako funkce předchozí, avšak s tím rozdílem, že data, která jsou jí předávaná se nachází v paměti RAM mikroprocesoru, což znamená, že touto funkcí lze posílat data přijatá mikroprocesorem z periferií. Tohoto se využívá při zobrazování teploty naměřené teplotním čidlem. [19]

9.4.3 Funkce print_webpage

Jak již název napovídá, slouží tato funkce k odeslání zdrojového kódu zobrazujícího webovou stránku v internetovém prohlížeči. Nachází se v hlavním programu a je volána pouze tehdy, obdrží-li ethernetový modul packet určený pro IP adresu webového serveru a v datové části obdrženého packetu se nachází klíčové slovo "GET ". Význam tohoto slova v obdrženém packetu byl popsán v kapitole 8.

Jako všechny funkce, které mění obsah bufferu, musí i tato funkce mít návratovou hodnotu ukazatele na poslední záznam v poli znaků reprezentující odesílací buffer.

V těle této funkce se také vykonává převod naměřené teploty na pole znaků a následný zápis do odesílacího pole. [19]

9.4.4 Funkce init_ip_arp_udp_tcp

Účelem této funkce je vytvoření lokálních proměnných reprezentující IP a MAC adresu v souboru ip_arp_udp_tcp.c. IP a MAC adresa jsou deklarovány v hlavním programu a tudíž je nutné je převést také do zmiňovaného souboru, jelikož některé další funkce, které se v tomto souboru nachází, operují s IP a MAC adresou.

Funkce nemá návratovou hodnotu. Do funkce vstupují dva ukazatele na pole, které reprezentují IP a MAC adresu. Třetím argumentem je 8 bitová proměnná udávající číslo portu, přes který komunikace probíhá (pro internetovou komunikaci slouží port 80). [19]

9.4.5 Funkce enc28j60_packet_send

Při konečném naplnění pole reprezentující odesílací buffer daty je nutné zapsat packet do odesílacího bufferu a následně odeslat jeho obsah na síť. To je úkolem funkce popisované v této podkapitole.

Při sestavování packetu je nutné vždy znát jeho délku. Tento údaj je velmi důležitý pro odesílání, neboť buffer se dělí na část pro odesílání a část pro přijímání. Pokud by nebyla známá délka odesílaného packetu, bylo by velmi obtížné odeslat požadovaný packet, neboť packet je funkcí enc28j60_packet_send zapsán do odesílacího bufferu a mohlo by se stát, že při neznámé délce odesílaného packetu by se přepsala i část přijímacího bufferu.

Funkce nejprve nastavuje horní a dolní bajt zapisovacího ukazatele na začátek odesílacího bufferu. Dále se nastaví horní i dolní bajt registru, který obsahuje konec odesílacího bufferu. Do toho „koncového“ registru se zapíše délka packetu.

Po nastavení koncového a počátečního ukazatele se provádí zápis dat zapsaných v poli, které v průběhu programu reprezentovalo odesílací buffer.

Jakmile jsou předchozí operace úspěšně vykonány, obsah odesílacího bufferu se pomocí řídícího registru ECON1 odešle na lokální síť. [19]

9.4.6 Funkce enc28j60_packet_receive

Předchozí kapitola popisovala, jak je realizováno odesílání dat. V této podkapitole se popisuje, jak je realizováno přijímání dat.

V prvním kroku je prováděna detekce obdržených dat v přijímacím bufferu. Tuto informaci zajišťuje registr EPKTCNT (ethernte packet counter). Zde je uložena hodnota počtu přijatých packetů. Pokud je tedy počet přijatých packetů 0, je jasné že se žádná data nepřijala, a také návratová hodnota funkce je 0.

Stejně jako v případě odesílacího bufferu je i při přijímání dat nutné nastavit dvoubajtový ukazatel, který ukazuje na počatek přijímacího bufferu (koncový ukazatel přijímacího bufferu je nastaven na poslední pozici celkové velikosti bufferu).

Dále je nastaven ukazatel na aktuální čtený packet. Přijatých packetů je zpravidla více než jeden, a při čtení přijímacího bufferu jsou data čteny po packetech.

V další části se kontroluje délka přijatého packetu. Nastane-li situace, že přijatý packet je delší než je definovaná maximální délka, je nutné jeho velikost „oříznout“ na hodnotu maximální možné délky definované v programu. Tato zdánlivě chybná operace je velmi důležitá, neboť v kapitole 5.2 je uvedeno, že velikost bufferu je 8kB. Větší část

tohoto bufferu tvoří přijímací buffer a data obdržená v tomto bufferu jsou posílána do RAM paměti mikroprocesoru, která je ovšem pouze 2kB. Z předcházejících dvou vět tedy plyne, že pokud by nebyla vložena operace ošetřující délku přijatých packetů, mohlo by se stát, že obdržený packet by byl delší než pamět RAM. V takovém případě by došlo k přetečení RAM paměti mikroprocesoru a program by přestal fungovat. Jako ideální maximální velikost byla zvolena hodnota 850 bajtů, lze ji však snadno změnit (viz program).

Pokud jsou data validní a kontrolní součet souhlasí, spustí se funkce která načte přijatý packet do paměti RAM mikroprocesoru ATmega 32.

Ukazatel na packet se po tomto procesu inkrementuje na následující packet. Po přečtení bufferu se dekrementuje packet counter, což znamená, že packet byl úspěšně přečten a již není potřeba ho nadále uchovávat v bufferu. [19]

9.4.7 Funkce enc28j60_init

Jak již bylo popsáno v kapitole 5.3.1, prvním krokem ke správné funkčnosti ethernetového modulu enc28j60 je inicializace. Tudíž funkce popisovaná v této kapitole má za úkol vyplnit všechny potřebné kontrolní registry. Součástí inicializace je také nastavení komunikačního SPI rozhraní, které je nutné správně konfigurovat.

Nejprve je však nutné správně nastavit I/O bity portu B a v případně výstupních pinů je nastavena logická (výchozí) úroveň:

PORT B4 (SCK) je nastaven jako výstup
PORT B5 (MOSI) je nastaven jako výstup
PORT B6 (MISO) je nastaven jako vstup
PORT B7 (SS) je nastaven jako výstup

PORT B5 (MOSI) má nastavenou log. úroveň 0
PORT B7 (SS) má nastavenou log. úroveň 0

Samotná inicializace SPI rozhraní se provádí následujícími dvěma příkazy (SPCR – SPI control register, SPST – SPI status register):

Po těchto korcích následuje již samotná inicializace popsaná v kapitole 5.4.1 (viz datasheet enc28j60 (str. 35 - 40)). [19]

9.5 Struktura funkce main

Jak již z názvu vyplývá, je funkce main hlavní funkcí, která sdružuje a ovládá všechny další funkce, které projekt obsahuje. Na rozdíl od programování programů spustitelných ve windows, programy ovládající mikroprocesor musí obsahovat v hlavní funkci nekonečný cyklus, ve kterém jsou ošetřeny možné hodnoty vstupů a reakce na ně. [19]

9.5.1 Tělo funkce main bez nekonečného cyklu

Před nekonečným while cyklem je část main funkce, která se provede pouze jednou. V této části se provádí inicializace ethernetového modulu, PHY registrů (LED A a LED B), časového čítače a TCP/IP protokolu. Také se zde provádí deklarace proměnných. [19]

9.5.2 Nekonečný cyklus funkce main

Tato klíčová část funkce main obsluhuje komunikaci a předávání dat mezi ethernetovým modulem a mikroprocesorem, vyhodnocuje přijatá data a odesílá odpovědi na dotazy klientů.

Na začátku cyklu se vždy provede nejdříve měření teploty. Měření je prováděno vždy, když hodnota čítače času překročí stanovenou mez (1 vteřina). Po provedení měření teploty následuje část, která obsluhuje www server.

Nejprve se provádí příkaz, který načte data čekající v bufferu. Následuje kontrola, která zjišťuje, zda byla obdržena nějaká data. Pokud je délka obdržených dat 0 provádí se příkaz continue, který přeskočí zbytek nekonečného cyklu.

Následují 3 podmínky, které analyzují přijatá data:

- Kontroluje se, zda byl přijat packet ARP, což by znamenalo dotaz na broadcast. Reakcí modulu na tento dotaz klienta je odeslání své IP a MAC adresy dotazovanému počítači.
- Dalším možnou událostí, kterou ošetřuje druhá podmínka, je stav, kdy přijatá data mají ve své hlavičce jinou cílovou IP adresu, než je IP adresa modulu. Pokud k tomuto dojde, znamená to, že data přijatá bufferem jsou určena pro jinou IP adresu a tudíž se jimi server již dále nezabývá.
- Poslední podmínkou je kontrola, zda byl přijat příkaz PING. Pokud je podmínka splněná, odešle se PONG, čímž modul ohlásí svou přítomnost na síti.

Předchozí část popisovaného kódu ošetřuje možné varianty příchozích packetů, jejichž obsluha vyžaduje pouze odeslání jednoho packetu (odpovědi) a nebo jsou data určena pro jiné zařízení na síti.

Pokud přijatý packet nesplnil žádnou s předchozích možností svého obsahu, je nutné otestovat, zda komunikační port packetu je shodný s www portem, kterým www server komunikuje. Jestliže i tímto testem packet prošel, znamená to, že klient vysílá požadavek na server. Aby mohla být komunikace uskutečněna, je nejprve nutné provést tzv. trojcestný handshaking. Tímto způsobem se zahájí komunikace mezi serverem a klientem, což znamená, že si mezi sebou již mohou vyměňovat požadovaná data.

Pokud jsou data validní pro kroky popisované v předchozím odstavci, je jisté, že klient odeslal serveru packet obsahující http protokol, což znamená, že do zadávacího pole pro URL adresu byla zadána IP adresa www serveru.

Před samotnou analýzou přijatého packetu je nejprve zkontrolována URL adresa, jelikož může nastat stav, kdy uživatel sice zadal správně IP adresu, avšak část za znakem "/" byla zadána chybně. V takovém případě odešle server klientovy odpověď, která zobrazí v klientském webovém prohlížeči chybovou hlášku.

Další ošetřenou možností chybného zadání URL adresy je pro případ, kdy uživatel zadal správně IP adresu a také posloupnost znaků reprezentující heslo, avšak za heslem byla zadána další posloupnost neočekávaných znaků. V takovém případě server klienta informuje, že požadovaná stránka byla permanentně přesunuta.

Z kapitol 9.1 a 9.3.2 lze vyvodit tři možné stavy, které mohou nastat po odeslání klientského požadavku na server: [19]

9.5.2.1 Detekce neúplného zadáni URL

Zadávání URL adresy je popsáno v kapitole 9.1, ze které vyplývá, že po zadání IP adresy webového serveru do kolonky adresa webového prohlížeče, musí následovat znak „/“ a za ním „heslo“ což je slovo, které je ke vstupu na server nutné znát.

Pokud si však uživatel není jist tím, v jakém formátu se URL adresa zadává, nastává po zadání pouze IP adresy stav 2 a dojde k zobrazení html stránky, jako nápovědy jak adresu správně zadat. Nutnou podmínkou ke správnému zadání adresy je však znalost hesla. Heslo se dá editovat pouze v režimu programování. Výchozím heslem je slovo "tajne". [19]

9.5.2.2 Test požadavku na funkci Wake on Lan

Z obrázku 9.1 je vidět, že webová stránka obsahuje zadávací i potvrzovací prvky, které umožní sestavit „Magic packet“ s MAC adresou zadanou uživatelem. Vlastností zadávacího políčka je, že zadaná hodnota se nezobrazí v URL adrese, což je podrobněji popsáno v kapitole 9.3.2.1. Z této kapitoly je patrné, že server obdrží od klienta packet, který v sobě obsahuje MAC adresu buzeného počítače. Server tuto adresu nalezne, převede ji z posloupnosti znaků na sekvenci šesti čísel. Těmito čísli je poté naplněn „Magic packet“ a poté odeslán. [19]

9.6 Obsluha teplotního čidla DS18B20

9.6.1 Funkce ds18b20_reset

Pro úspěšnou realizaci algoritmu měření teploty je nejprve nutné provést inicializaci. Ta se provádí funkcí ds18b20_reset.

Prvním krokem je nastavení příslušného pinu do logického stavu 0 (nastavení PINB0 jako výstup) na stanovenou dobu alespoň 480ms. Výsledkem popisované operace je tzv. reset puls.

Po této akci se díky pull-up rezistorům nastaví PINB0 do logického stavu 1 (nastavení PINB0 jako vstupu) a je potřeba, aby byla dodržena časová prodleva, kdy se nesmí měnit nastavení DDRB registru. Zpravidla to bývá až 15ms, po jejichž uplynutí DS18B20 odpoví, tzv. Presence pulsem. Presence puls nastaví PINB0 do hodnoty logická 0 (výstup), což znamená, že DS18B20 je funkční, inicializace proběhla úspěšně a čidlo je připraveno k měření. [9],[18], [20]

Zobrazení průběhu signálů viz Příloha 1.

9.6.2 Funkce ds18b20_send_instruction

Jak již bylo popsáno v kapitole 6, je každý příkaz reprezentovaný specifickým hexadecimálním číslem. Toto číslo je nutné odeslat z mikroprocesoru do teplotního čidla.

Rozlišení zda se odesílá log. 0 či log. 1 je (podobně jako u inicializace) řešeno pomocí časování a nastavování logických úrovní výstupů bitů portu B. Funkce tedy nastaví pin portu jako výstupní na dobu 7µs. Pokud se odesílá log. 1 nastaví se příslušný pin portu jako vstup ihned po uplynutí prodlevy. Pokud je odesílána log. 0, následuje prodleva 70 µs, po jejímž uplynutí je daný pin nastaven taktéž jako vstup. Grafické znázornění průběhu signálu pro zápis je uvedeno v příloze 4.

Jelikož posílaná instrukce je 1 bajtové číslo, provede se předchozí postup 8x. [9],[18], [20]

9.6.3 Funkce ds18b20_read_bite

Tato funkce je volána po provedení instrukce CONVERT_T. Po této instrukci následuje inicializace a provedení příkazu Skip ROM. Z postupu měření uvedeného v kapitole 6 je patrné, že naměřená teplota se uložila do scratchpad registru (viz příloha 2).

Pro vyčtení teploty z registru scratchpad je nutné zavolat příkaz Read scratchpad (0xBE). Po odeslání tohoto příkazu začne DS18B20 vysílat naměřenou teplotu jako 2 bajtové číslo v doplňkovém kódu.

K vyčtení jednoho bitu je nutné provést nastavení obslužného pinu jako výstup na dobu 2 µs, poté je nastaven opět jako vstup. Po cca 15 µs prodlevě je přečtena logická úroveň tohoto pinu. Je-li log. úroveň 1, provede se zapsání 1 do proměnné reprezentující teplotu. V opačném případě je do této proměnné zapsána 0. Grafické znázornění průběhu pulsů pro čtení jednotlivých bitů je uvedeno v příloze 5.

Následuje bitové rotace hodnoty proměnné reprezentující naměřenou teplotu o 1 bit směrem doprava. Tímto postupem se načtou celé 2 bajty. [9],[18], [20]

9.7 Převod naměřené teploty na pole znaků

Jak již bylo popsáno v kapitole 6, výchozí rozlišení měřené teploty je 12 bitové. Pro měření teploty v místnosti je však naprosto dostačující rozlišení 10 bitové (0,25°C). Vyčtená hodnota změřené teploty se tudíž rotuje o 2 bity směrem vpravo.

Jelikož je rozlišení naměřené teploty 0,25°C a proměnná, ve které je tato hodnota uložena je celočíselný neznaménkový datový typ, musí být tato hodnota vynásobena číslem 25, čímž se naměřená hodnota teploty stane celým číslem. Pro zobrazení naměřené teploty na webové stránce je nutné převést číselnou hodnotu teploty na řetězec znaků.

Převod čísla na znak je proveden jako součet hodnoty znaku "0" (0x30) a zbytku po celočíselném dělení proměnné reprezentující teplotu deseti.

Tento součet se zapíše do pole znaků a proměnná reprezentující teplotu se vydělí beze zbytku deseti. [9]

10 Závěr

Prvním cílem bakalářské práce bylo seznámení se s danou problematikou na internetu. Tato část je rozebrána v kapitolách 3 až 7, kde jsou popsány základní principy, na kterých je tato práce postavena.

Po nastudování jednotlivých komponent bylo nutné správně propojit mikroprocesor s periferiemi. Jakmile byl tento bod praktické realizace splněn, začala fáze implementace programu. K implementaci programu pro WOL do mikrokontroleru ATmega jsem použil AVR Studio, což je vývojový software firmy Atmel, který je zdarma ke stažení na oficiálních stránkách této firmy.

Proces realizace však velmi často přecházel v proces analýzy, neboť byl použit již naprogramovaný projekt webového serveru, a tudíž bylo nejprve nutné pochopit původní projekt a následně ho správně modifikovat. K monitoringu odesílaných a přijímaných dat ze serveru ke klientovi jsem používal software Wireshark, což je velmi názorný a užitečný program při monitoringu sítí.

Paměť RAM mikroprocesoru ATmega 32 má dle datasheetu 2Kb. Ethernetový buffer modulu ENC28J60 má velikost 8Kb (viz datasheet). Jelikož jsou data vyslaná modulem do mikroprocesoru ukládána do paměti RAM, je zřejmé, že maximální velikost přijatého packetu by neměla přesáhnout 2Kb, avšak tato hodnota je pouze teoretická, neboť část paměti RAM je použita pro provoz mikroprocesoru a dále nelze vyloučit možnost, že mikroprocesor bude komunikovat s dalšími periferiemi (např. DS18B20). Z uvedených důvodů byla zvolena maximální velikost přijatého packetu 850 bajtů, což dává mikroprocesoru dostatek prostoru pro ostatní operace.

Komplikace nastaly při náhradě provizorních vodičů mezi mikroprocesorem a ethernetovým modulem za nové. Po této výměně ethernetový modul přestal komunikovat s mikroprocesorem, avšak dle indikačních LED stále přijímal data a bylo možné ho přeprogramovat. K analýze tohoto problému bylo nutné využít osciloskop, neboť vzniklý problém bylo velmi složité odhalit. Vylučovacím postupem se poté dospělo k závěru, že vzniklý problém byl důsledkem absence stabilizátoru napájecího napětí při napájení kitu pomocí USB portu. Zapojené USB napájení totiž namísto žádaných 5V napájelo vývojový kit napětím 6V (nevhodné napájení usb switche), což se ukázalo jako fatální. Odhalení tohoto problému bylo značně složité, jelikož I/O porty mikroprocesoru se jevily jako funkční. Avšak z obrázku 4.1 je zřejmé, že kromě funkce I/O plní každý pin další, specifickou funkci. Důvod této poruchy byl nefunkční 6. bit portu B, který při komunikaci pomocí SPI rozhraní plní funkci MISO, tedy přijímání dat z ethernetového modulu. Původní mikroprocesor se tedy už nedal k tomuto účelu využít, neboť nastalá změna byla nevratná a bylo nutné koupit nový mikroprocesor.

K programování mikroprocesoru jsem využíval programátor JTAG AVRPROG USB v2.

Seznam použitých zdrojů

[1] OnPa design & development embedded systems: kit EvB 4.3. ONDŘEJ PAVELKA. *OnPa design & development embedded systems* [online]. 2010 [cit. 2013-05-21]. Dostupné z: http://shop.onpa.cz/?kit-evb-4.3,27.

[2] OnPa design & development embedded systems: kit EvB 4.3. ONDŘEJ PAVELKA. *OnPa design & development embedded systems* [online]. 2010 [cit. 2013-05-21]. Dostupné z: http://shop.onpa.cz/?podpora,28.

[3] ATMEL. *ATmega32/6 Datasheet*. 2011, 346 s. Dostupné z: http://www.atmel.com/Images/doc2503.pdf.

[4] Serial Peripheral Interface. In: *Wikipedie* [online]. 2007. vyd. 2007, 11.3.2013 [cit. 2013-05-21]. Dostupné z: http://cs.wikipedia.org/wiki/Serial_Peripheral_Interface.

[5] AND-TECH: elektronika, hobby, pasja. *ENC28J60* [online]. 2012 [cit. 2013-05-21]. Dostupné z: http://and-tech.pl/modul-ethernetowy-and-eth-v2-2/.

[6] MICROCHIP. *ENC28J60 Data Sheet: Stand-Alone Ethernet Controller with SPI™ Interface*. 2004, 102 s. Dostupné z: http://ww1.microchip.com/downloads/en/devicedoc/39662a.pdf.

[7] Duplexní spojení. In: *Wikipedie* [online]. 2006, 9.3.2013 [cit. 2013-05-21]. Dostupné z: http://cs.wikipedia.org/wiki/Duplexn%C3%AD_spojen%C3%AD.

[8] Teplotní čidlo DS18B20 ve spojení s PIC12F629. In: *Pandatron* [online]. 2008, 3.12.2008 [cit. 2013-05-21]. Dostupné z: http://pandatron.cz/?566&teplotni_cidlo_ds18b20_ve_spojeni_s_pic12f629.

[9] MAXIM INTEGRATED. *DS18B20: Programmable Resolution 1- Wire Digital Thermometer*. 2008. Dostupné z: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

[10] Sága rodů LAN a WAN. In: *EArchiv* [online]. 1997 [cit. 2013-05-21]. Dostupné z: http://www.earchiv.cz/a708s600/a708s684.php3.

[11] User Datagram Protocol. In: *Wikipedie* [online]. 2005, 8.3.2013 [cit. 2013-05-21]. Dostupné z: http://cs.wikipedia.org/wiki/User_Datagram_Protocol.

[12] NAVRÁTIL, Petr. Síťový teploměr. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. *Ústav radioelektroniky* [online]. 2012 [cit. 2013-05-21]. Dostupné z: http://www.urel.feec.vutbr.cz/MIA/2012/Navratil/index.html.

[13] National Instruments. NATIONAL INSTRUMENTS. *Acquiring from GigE Vision Cameras with Vision Acquisition Software - Part I* [online]. 2007 [cit. 2013-05-21]. Dostupné z: http://www.ni.com/white-paper/5651/en.

[14] Wake on LAN. In: *Wikipedie* [online]. 2007, 10.3.2013 [cit. 2013-05-21]. Dostupné z: http://cs.wikipedia.org/wiki/Wake_on_LAN.

[15] Webový server. In: *Wikipedie* [online]. 2007, 8.3.2013 [cit. 2013-05-21]. Dostupné z: http://cs.wikipedia.org/wiki/Webov%C3%BD_server.

[16] KDYŽ ZLOBÍ TEPLOMĚRY. In: Czech Amateur Near-Space Object [online]. 2010 [cit. 2013-05-23]. Dostupné z: http://blog.czanso.com/tag/ds18b20/.

[17] EArchiv. In: Referenční model ISO/OSI - přenos dat [online]. 1992 [cit. 2013-05-24]. Dostupné z: http://www.earchiv.cz/a92/a215c110.php3.

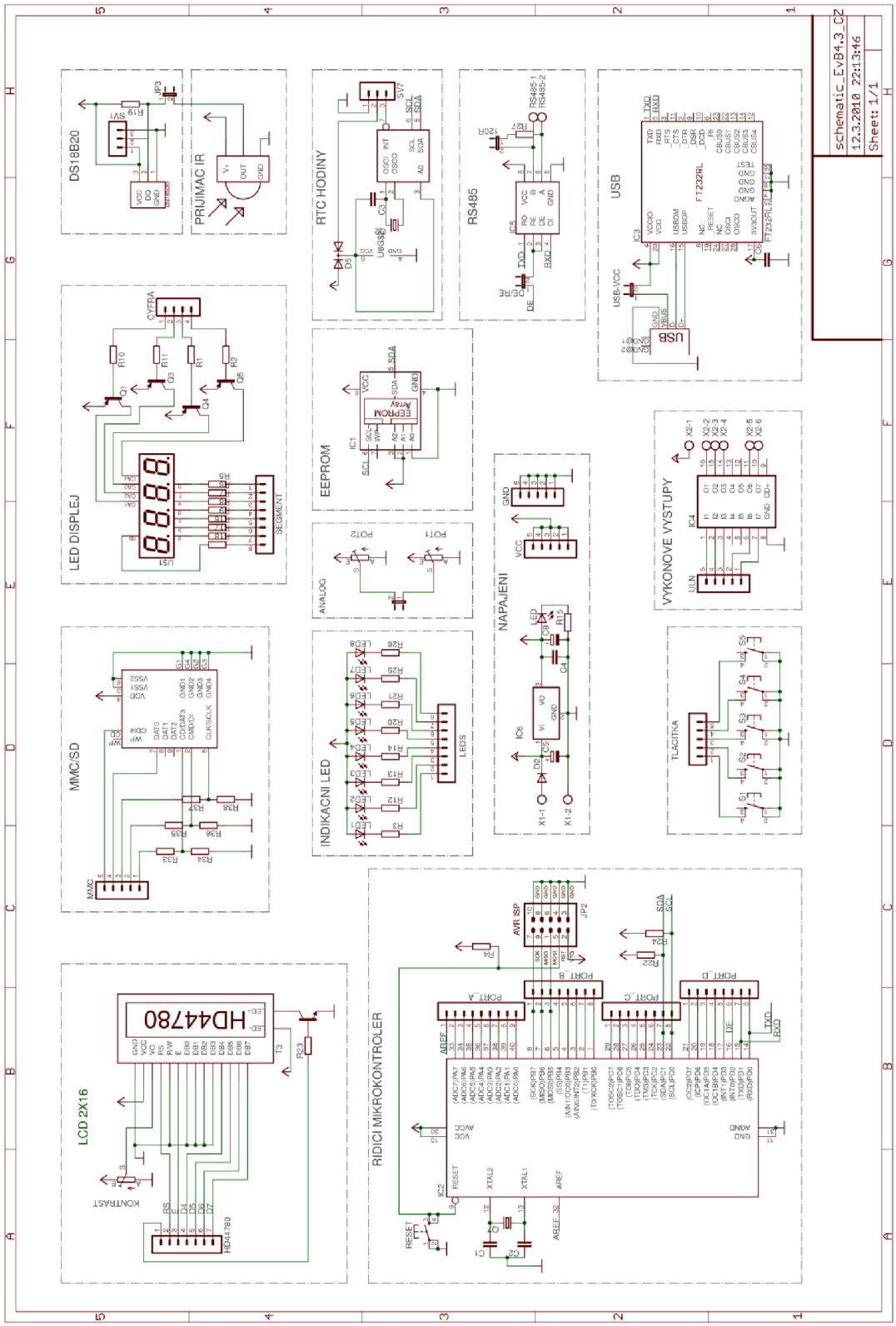
[18] Osobní stránky. 1-wire sběrnice a teploměr DS18B20 [online]. 2006 [cit. 2013-05-24]. Dostupné z: http://www.ok2jnj.wz.cz/ds18b20.htm.

[19] Tuxgraphics. In: An AVR microcontroller based Ethernet device [online]. 2008 [cit. 2013-05-24]. Dostupné z: http://www.tuxgraphics.org/electronics/200606/article06061.shtml.

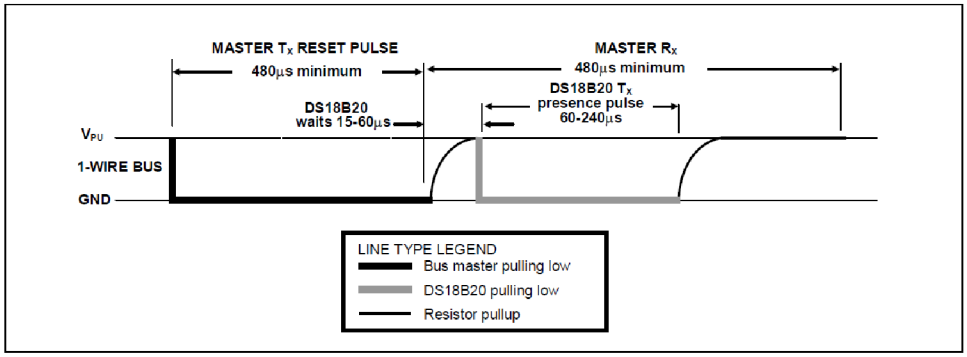
[20] Teslabs. Using DS18B20 digital temperature sensor on AVR microcontrollers [online]. 2007 [cit. 2013-05-24]. Dostupné z: http://teslabs.com/openplayer/docs/docs/other/ds18b20_pre1.pdf.

Seznam příloh:

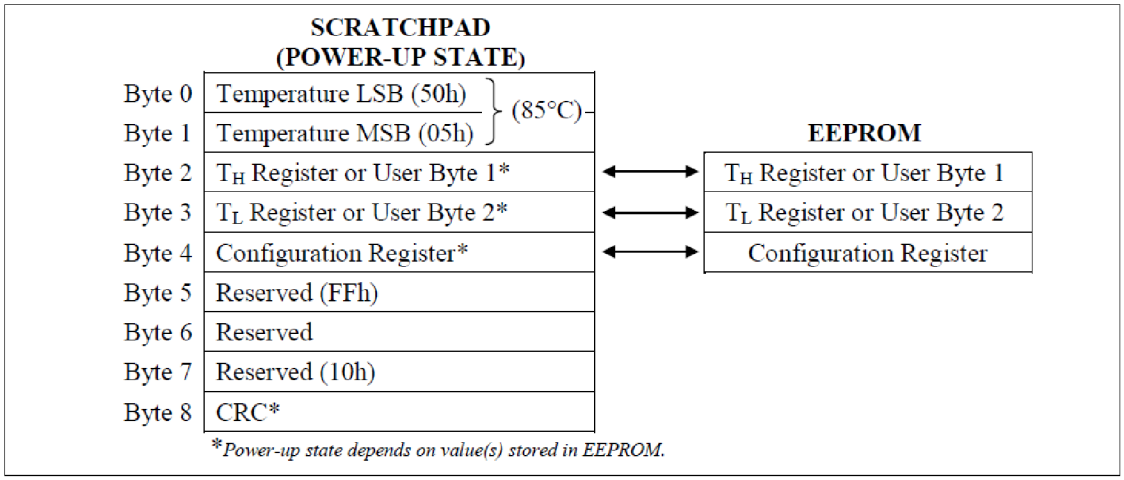
Příloha 1	Schéma vývojového kitu EvB 4.3 v 4
Příloha 2	Resetu puls DS18B20
Příloha 3	Paměť scratchpad DS18B20
Příloha 4	Zobrazení správného časování pro zápis log.0/log1 pro DS18B20
Příloha 5	Zobrazení správného časování pro vyčtení log. 0/log. 1 z teplotního čidla
Příloha 6	Model ISO/OSI
Příloha 7	Vývojový diagram měření a načtení teploty



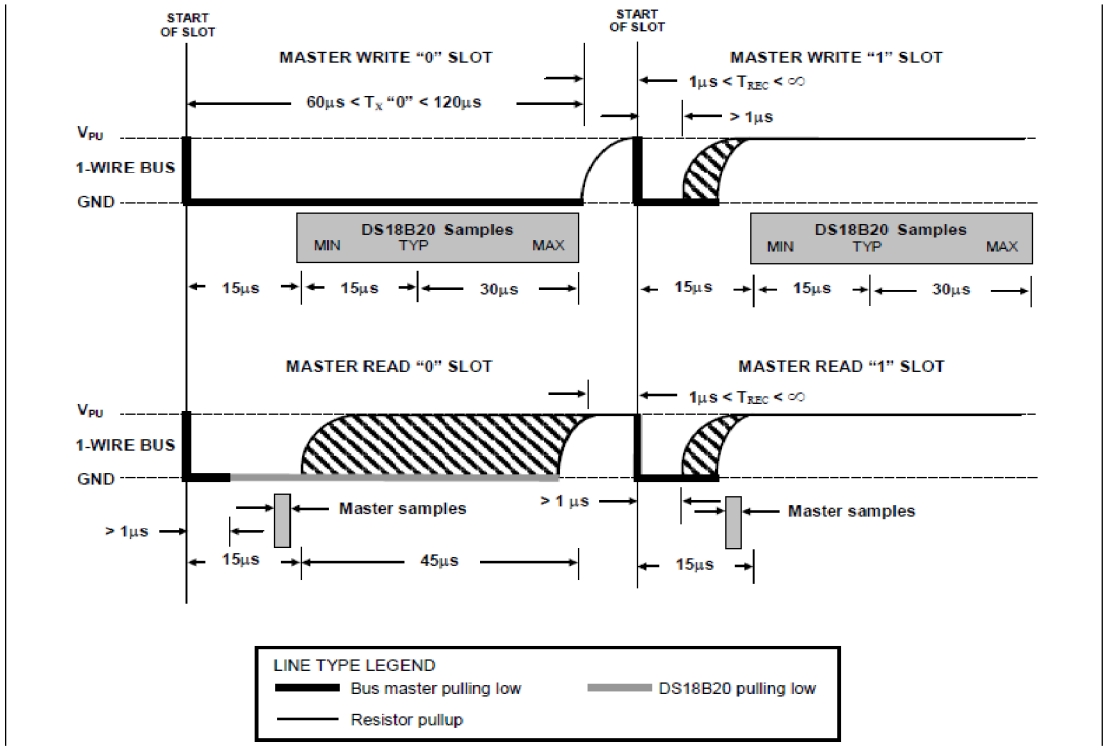
Příloha 1: Schéma vývojového kitu EvB 4.3. zdroj: [2]



Příloha 2: Reset puls DS18B20. Zdroj: [9]



Příloha 3: Paměť scratchpad



Příloha 4: Zobrazení správného časování pro zápis log.0/log1 pro DS18B20. Zdroj: [9]

Figure 15. Detailed Master Read 1 Timing

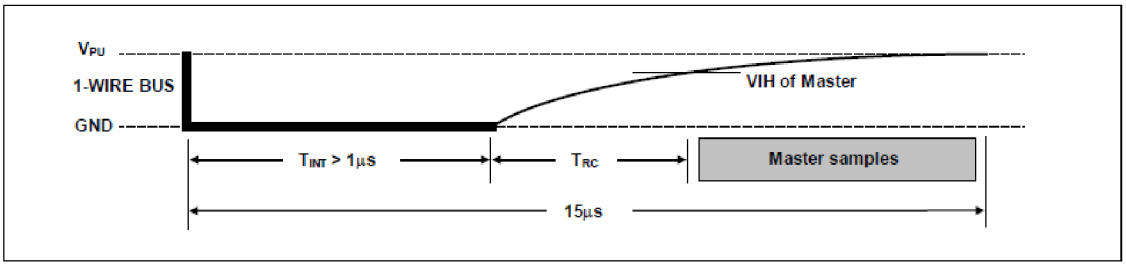
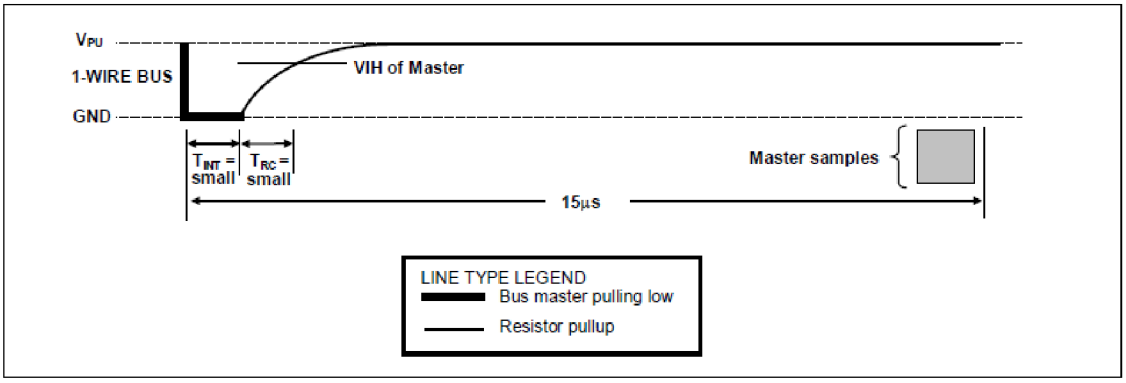
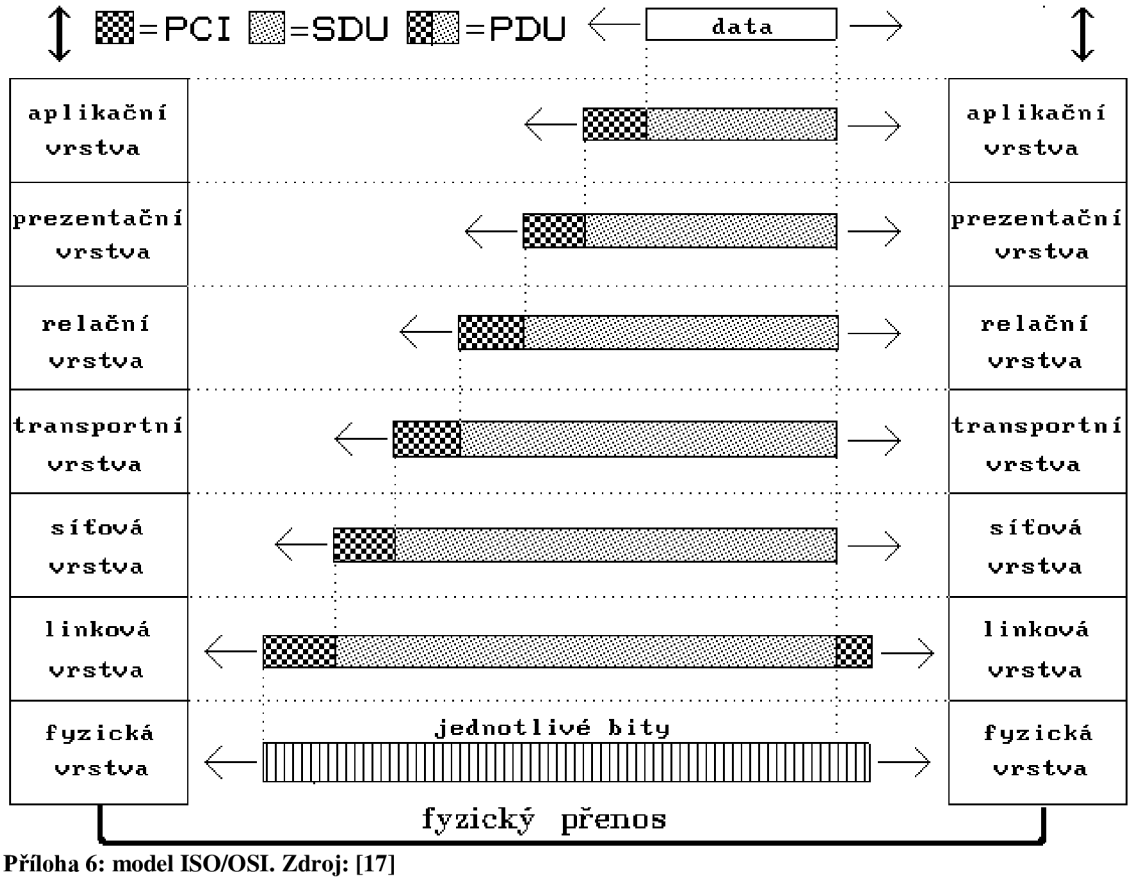


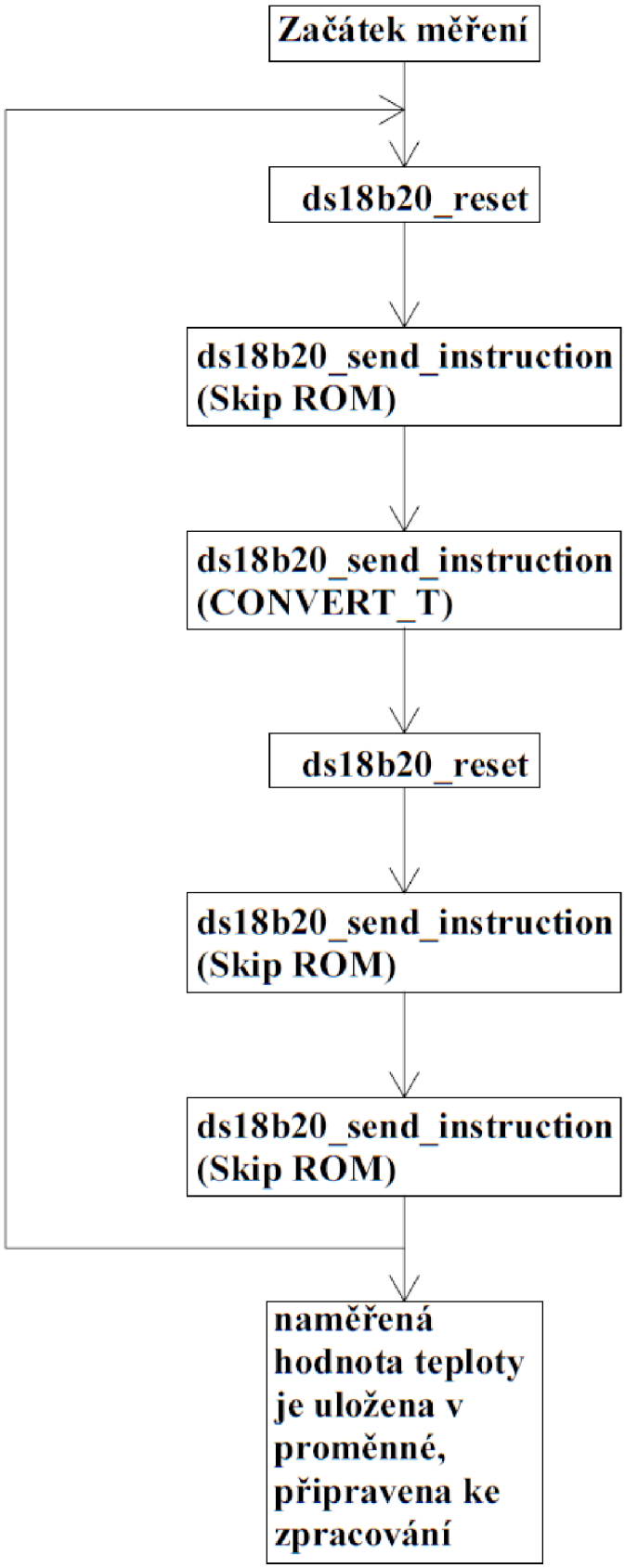
Figure 16. Recommended Master Read 1 Timing



Příloha 5: Zobrazení správného časování pro vyčtení log. 0/log. 1 z teplotního čidla. Zdroj: [9]



Příloha 6: model ISO/OSI. Zdroj: [17]



Příloha 7: Vývojový diagram měření a načtení teploty. Zdroj: [9]