TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

ANALÝZA KOMUNIKACE NA I2C SBĚRNICI

Bakalářská práce

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Autor práce: **Petr Vošta**
Vedoucí práce: Ing. Petr Pfeifer

Liberec 2014

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr Vošta**
Osobní číslo:	**M11000185**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronické informační a řídicí systémy**
Název tématu:	**Analýza komunikace na I2C sběrnici**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Nastudujte standardy sběrnice I2C (Inter-Integrated Circuit).
2. Nastudujte možnosti a formát dat přístroje Agilent 9000 s ohledem na použití při analýze komunikace na této sběrnici.
3. Navrhněte a realizujte program, který provede analýzu navzorkovaných dat komunikace na sběrnici pomocí osciloskopu a vytvoří detailní výpis komunikace ve formátu HTML včetně časových značek.
4. Pokuste se rovněž vhodně zakomponovat analýzu a zobrazení upozornění na základní chybové stavy a kritické konflikty na logické vrstvě. U fyzické vrstvy pouze indikujte přítomnost signálů s napětím a dobou trvání vně nebo uvnitř zvoleného pásma.

Rozsah grafických prací: **Dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 30 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Dokumentace k osciloskopu Agilent série 9000 včetně formátů ukládaných dat**

[2] **PRATA, Stephen. C++ primer plus. 5th ed. Indianapolis: SAMS, 2005, 1202 s. ISBN 0-672-32697-3**

[3] **VIRIUS, Miroslav. Pasti a propasti jazyka C++. 2. aktualiz. a rozš. vyd. Brno: CP Books, 2005, 375 s. ISBN 80-251-0509-1**

Vedoucí bakalářské práce: **Ing. Petr Pfeifer**
Ústav informačních technologií a elektroniky

Datum zadání bakalářské práce: **12. září 2013**

Termín odevzdání bakalářské práce: **16. května 2014**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 -
L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
vedoucí ústavu

V Liberci dne 12. září 2013

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Tímto děkuji svému vedoucímu bakalářské práce panu Ing. Petru Pfeiferovi za vedení bakalářské práce, konzultace, cenné rady a odbornou pomoc. Dále bych chtěl poděkovat za podporu své rodině.

Abstrakt

Tato bakalářská práce se zabývá analýzou sběrnice I2C. Začíná seznámením se se sběrnicí I2C a s osciloskopem z řady Agilent 9000 a jeho využitím pro měření a analýzu sběrnice I2C. Součástí práce je samotný návrh a realizace analýzy s ohledem na co nejrychlejší a nejefektivnější zpracování vstupních dat. Součástí práce je popis jednotlivých důležitých funkcí, kterými například je rozbor důležitých hlaviček, načítání dat a jejich komprimace, dekódování dat nebo výpis výsledků. Zároveň se práce zabývá porovnáním rychlosti výsledné aplikace na různých počítačích a porovnáním s komerčně dostupnými řešeními.

Klíčová slova

I2C, Inter-Integrated Circuit, analýza sběrnice, Agilent 9000, detekce chyb

Abstract

This bachelor deals with analysis of the bus I2C. In this document, the I2C bus is described and the oscilloscope Agilent 9000 series introduced as well, especially with regards to its use for measurements and analysis of the I2C bus. The design and realisation of the analysis and software is the key part of my work. The analysis has been optimized in order to maximize the speed and effective data processing. Another part of this work is the description of the important functions, like the description of important headers, data loading and data compression and decode, as well as presentation of the results. This bachelor thesis compares the speed of the final application on different computers, applying the presented solution on selected commercially available platforms.

Keywords

I2C, Inter-Integrated Circuit, bus analysis, Agilent 9000, debug

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

I2C	Inter-Integrated Circuit	Sériová sběrnice
IIC	Inter-Integrated Circuit	Sériová sběrnice
SDA	Signal Data	Datový signál
SCL	Serial clock	Hodinový signál sběrnice I2C
CLK	Clock signal	Hodinový signál
GND	Ground	Zemní nebo společný potenciál
ACK	Acknowledge	Potvrzení přijetí
NACK	Not Acknowledge	Potvrzení nepřijetí
SPI	Serial Peripheral Interface	Sériová sběrnice
x86	32bit system	32bitový systém
x64	64bit system	64bitový systém
GSa/s	Giga Samples per second	Miliarda vzorků za sekundu
Gpts	Giga points	Miliarda bodů
Mpts	Mega points	Milion bodů
TSV	Tab-Separated Values	Hodnoty oddělené tabulátorem
CSV	Coma-Separated Values	Hodnoty oddělené čárkou
HDF5		Datový model a přípona souboru
BIN		Formát a přípona binárního souboru
Html	HyperText Markup Language	Značkovací jazyk
RAM	Random Access Memory	Paměť s přímým přístupem
SSD	Solid State Drive	Pevný disk založený na flash pamětech

1 Úvod

Ke zpracování této bakalářské práce na téma analýza komunikace na I2C sběrnici mne motivovala touha vytvořit práci, která bude prakticky využitelná nejen k jednomu specifickému využití, ale bude univerzálnější s možností jí dále rozšířit. Zároveň mne práce zaujala tím, že jsem si mohl rozšířit své znalosti v oblasti sběrnic a mohl pracovat se špičkovým vybavením.

Cílem této bakalářské práce bylo nastudovat standard sběrnice I2C a nabyté znalosti využít při návrhu programu, který bude zpracovávat naměřená data pomocí osciloskopu. Zároveň s tím bylo nutno nastudovat dokumentaci k osciloskopu Agilent DSO9254A, respektive dokumentaci k řadě osciloskopů Agilent 9000. Cílem navrženého programu je následné zpracování naměřených dat z tohoto osciloskopu. Zjištění která zařízení spolu komunikují, co si posílají, zjišťovat základní chyby a následně vytvořit výsledný HTML soubor se všemi dekódovanými informacemi, včetně podrobného rozepsání v jakém čase k čemu došlo.

Celá práce je rozdělena do dvou hlavních částí. První část práce se zabývá teoretickým rozborem včetně seznámení se s dostupnou dokumentací k I2C sběrnici a osciloskopu Agilent DSO9254A. Druhá část práce se následně zabývá návrhem a realizací samotného programu tak, aby byl co možná nejuniverzálnější pro další rozšíření.

2 Teoretický rozbor

2.1 Požadavky

Na výslednou aplikaci jsou kladeny požadavky na několik různých parametrů. Do první skupiny požadavků, které jsou nejdůležitější, patří zejména schopnost dekódovat soubory uložené osciloskopem Agilent řady 9000 a analýza takto naměřených průběhů včetně detekce základních chyb. Dalším požadavkem je detailní a srozumitelný výpis analyzovaných dat ve formátu HTML tak, aby byl zobrazitelný na každém počítači pomocí webového prohlížeče. Dále aplikace musí umět data zpracovat v krátkém časovém úseku s co možná nejmenšími systémovými požadavky tak, aby aplikace fungovala rychle i na starších počítačích. V neposlední řadě je samozřejmá podpora x64 systémů a tím zaručená podpora zpracování i velikých naměřených souborů.

Do druhé, již trochu méně podstatné, skupiny požadavků spadá jednoduchost celé aplikace z pohledu uživatele. S tímto je kladen důraz také na autonomii celého řešení tak, aby uživatel nebyl zbytečně obtěžován detaily, které se dají zpracovat přímo z naměřených dat, bez jakéhokoliv dalšího zásahu uživatele. Zároveň ale musí aplikace splňovat požadavky, kladené v souvislosti s jejím začleněním do nového systému vyvíjeného na TUL.

Pro pozvednutí výsledné aplikace na vyšší úroveň oproti stávajícím řešením je nutné, aby byla co nejvíce univerzální a tím byla dobře rozšiřitelná o podporu dalších osciloskopů, výstupních formátů dat, případně dalších sběrnic.

2.2 Osciloskop

Tato kapitola popisuje použitá zařízení a jejich možnosti s ohledem na zadání práce.

2.2.1 Agilent DSO9254A

Osciloskop Agilent řady 9000, konkrétně typ DSO9254A s maximální možnou velikostí paměti, byl zakoupen z projektu ESF CZ.1.07/2.2.00/28.0050 Modernizace didaktických metod a inovace výuky technických předmětů.

O osciloskopu

Agilent DSO9254A (Agilent Technologies, 2013) byl vyvinut firmou Agilent pro široké a náročné využití. Tento model má čtyři kanály se šířkou pásma 2,5 GHz při vstupní impedanci 50 Ω, nebo 500 MHz při vstupní impedanci 1 MΩ. Při obou vstupních impedancích dosahuje vzorkovací frekvence 20 GSa/s při použití dvou kanálů a 10 GSa/s při použití všech 4 kanálů. Vzorkovací paměť má nejvyšší možnou velikost 1 Gpts na kanál. Dále je možno osciloskop rozšířit o 16kanálový logický analyzátor nebo zvětšit šířku pásma na 4 GHz pouze softwarovým klíčem.

Největší výhodou toho osciloskopu je především jeho velká paměť, díky níž lze ve spojení s rychlým vzorkováním vzorkovat rychlé signály po dostatečně dlouhou dobu na to, aby bylo možno tato data potřebně analyzovat ať již pomocí nástrojů samotného osciloskopu, nebo dalších vlastních SW nástrojů. Nebo naopak lze pomalejší signály, jako například sběrnici I2C vzorkovat dlouhou dobu a z její analýzy a zjištění přenášených dat odladit vyvářenou elektroniku, nebo diagnostikovat opravovanou elektroniku. Druhou největší výhodou je přítomnost desktopového operačního systému Windows 7 Professional x64. Díky tomuto operačnímu systému, který je aktuálně velice rozšířen, je práce s osciloskopem snadná a navíc umožňuje připojení a využívání periferií a služeb jako na stolním počítači nebo notebooku bez dalších omezení. Přenos naměřených dat je urychlen rovněž díky přítomnosti 1 Gbit LAN.

Výstup z osciloskopu

Výstup z osciloskopu Agilent DSO9254A lze provést dvěma způsoby. První způsob je přenos dat pomocí vlastního nebo zakoupeného SW od výrobce přímo z osciloskopu po počítačové síti LAN (1 Gbit). Druhou možností je uložení naměřených dat do souboru na flash disk, pevný disk, FTP nebo další uložiště, které podporuje operační systém Windows 7 Professional. Při ukládání dat do souboru je několik možností, využití HDF5, což je datový model určený především pro práci s velkými objemy dat a práci s různorodými daty. Druhý podporovaný formát je CSV nebo TSV, kdy se jedná ve své podstatě o textový soubor, ve kterém jsou jednotlivé údaje oddělené oddělovači. Pro CSV je oddělovačem čárka a u TSV je to tabelátor. Poslední možností jak data uložit je binární soubor s příponou BIN, ve kterém jsou veškerá naměřená data uložena v binární podobě.

Pro HDF5 je specifické, že data uložená v tomto formátu jsou uvnitř souboru strukturovaná podobně jako soubory na pevném disku počítače a i tak se s nimi pracuje. Tím je zaručena univerzálnost tohoto datového modelu. V soboru jsou tak zvlášť uložená data obsahující hlavičku souboru a zvlášť samotná data.

V binárním souboru, který je vhodný pro další strojové zpracování dat, jsou opět data formátovaná následovně. První je hlavička celého souboru, ve které jsou uloženy informace ve dvou znacích, že se jedná o osciloskop Agilent, následuje verze souboru, která je pro každou řadu osciloskopů jiná, dále je velikost souboru v bytech a informace o počtu měřených kanálů. Za hlavičkou souboru jsou za sebou uložena naměřená data jednotlivých kanálů. Tato data jsou strukturovaná tak, že jim předchází hlavička měřeného signálu, která obsahuje informace o velikosti hlavičky, typu měřeného signálu, počtu vzorků, rychlosti vzorkování, jednotkách, popis kanálů a samozřejmě nechybí datum a čas. Za touto hlavičkou je poslední krátká hlavička dat signálu, která obsahuje informaci o tom, v jakém datovém typu jsou naměřená data uložená, kolik bytů dat je jeden vzorek a následně surová naměřená data daného kanálu v podobě aktuální amplitudy vzorku. Čas, který tomuto vzorku odpovídá, se musí dopočítat z počátečního data a času měření, známé rychlosti vzorkování a informace kolikátý vzorek je právě načten (Agilent Technologies, 2014). Ilustrační náhled jak vypadá binární soubor, viz Tabulka 1: Uspořádání dat v binárním souboru, kde jsou jednotlivé signály odděleny silnější čarou.

Soubory CSV a TSV tím, že jsou textové, tak nejsou nejvhodnější jako výstup pro další zpracování dat, protože výstupní data jsou větší se stejnou informační hodnotou než soubory BIN a HDF5. Na druhou stranu jsou ale vhodné pro uživatele, protože jsou pro něho čitelná a snadno se v nich vyzná. I v těchto formátech jsou nejprve uloženy hlavičky a následně jednotlivá naměřená data.

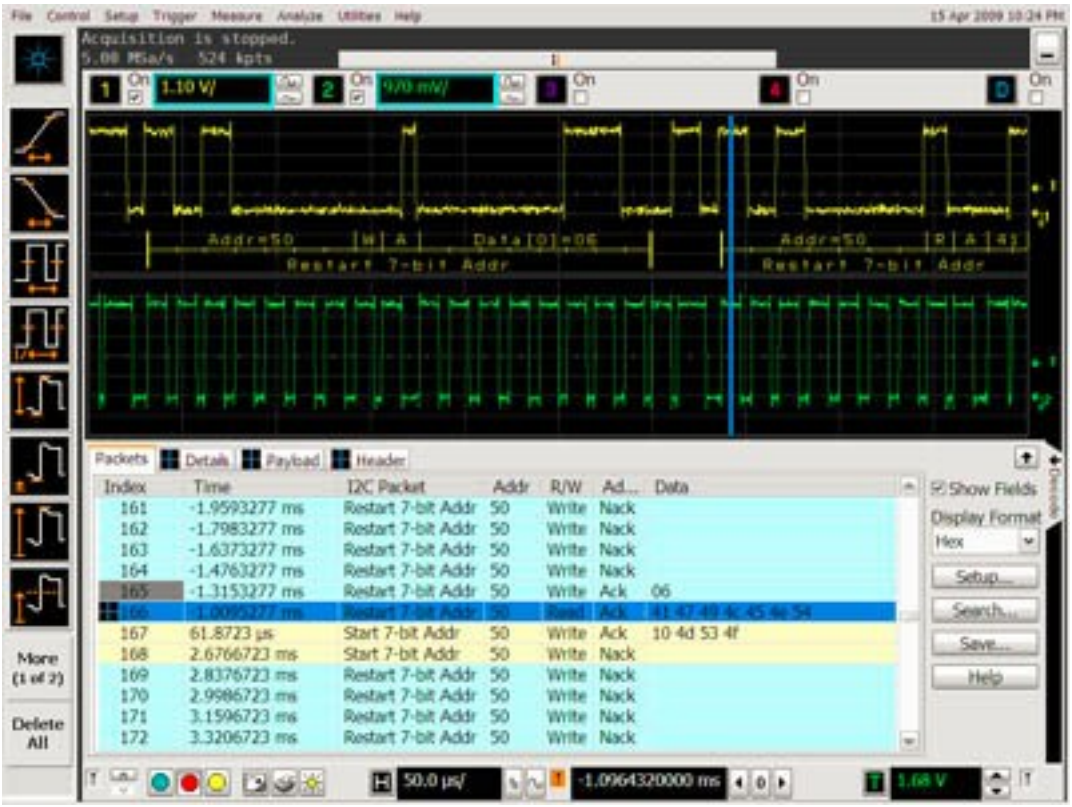
Tabulka 1: Uspořádání dat v binárním souboru

File Header (Hlavička souboru) 8 bytů
Waveform Header 1 (Hlavička 1. signálu) 136 bytů
Waveform Data Header 1 (Datová hlavička 1. signálu) 12 bytů
Data N_1 bytů
Waveform Header 2 (Hlavička 2. signálu) obsahuje informace o 136 bytů
Waveform Data Header 2 (Datová hlavička 2. signálu) 12 bytů
Data N_2 bytů
Waveform Header n (Hlavička n-tého. signálu) 136 bytů
Waveform Data Header n (Datová hlavička n-tého. signálu) 12 bytů
Data N_n bytů

Každá z hlaviček Waveform Data Header obsahuje informace o měřeném průběhu a nastavení kanálu osciloskopu. Spolu s obsahem hlavičky Waveform Data Header, která obsahuje informace o formátu uložených dat, poskytují informaci o počtu bytů naměřených dat. Konkrétně se jedná o součin parametru *Count* z hlavičky Waveform Header, který nese informaci o počtu vzorků a parametr *Bytes Per Point z hlavičky* Waveform Data Header.

2.2.2 Možnosti dekódování dat

Pro porovnání možností dekódování dat na sběrnici I2C jsem si vybral 3 rozšířené výrobce osciloskopů a logických analyzátorů, firmy Agilent, GW Instek a Rigol.

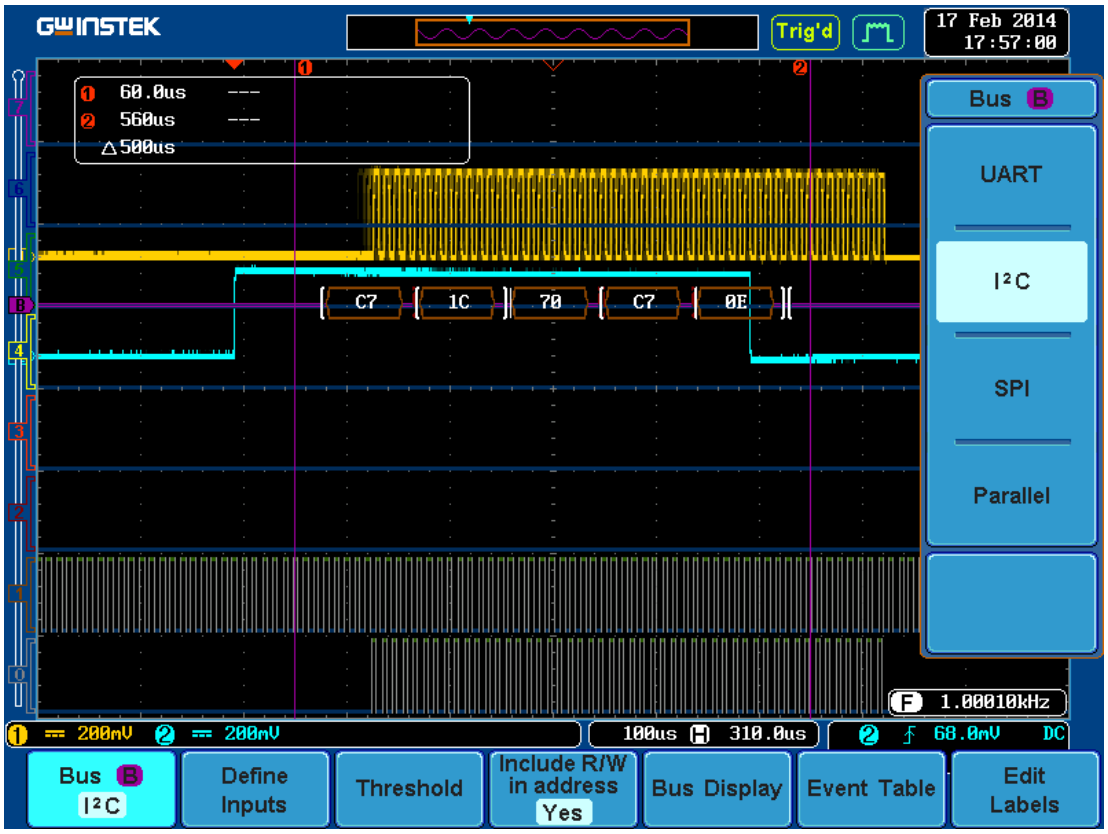


Obrázek 1: N5391B

U osciloskopů Agilent řady 9000A a 9000H je od výrobce dostupná utilita s označením N5391B (Agilent Technologies, 2013), která umožňuje dekódování I2C nebo SPI. Tato utilita se doinstaluje přímo do SW prostředí, ve kterém probíhá veškerá práce s osciloskopem, což je její největší výhoda. Tato utilita přehledně analyzuje a zobrazuje dekódovaná data v přehledné tabulce, ve které je číslo paketu, čas od počátku vzorkování, následuje typ počáteční podmínky komunikace a velikost adresy. Následně je zobrazena adresa, směr komunikace, její potvrzení a samotná data viz Obrázek 1: N5391B. Nastavení dekódování je široké a umožňuje nastavení podmínky triggeru, vyhledávání v dekódovaných datech a nastavení jednotlivých kanálů pro SDA a CLK signály, kde se mohou vybrat buď analogové kanály 1 až 4 nebo v případě vybavení osciloskopu logickým analyzátorem také jeden

z šestnácti kanálů logického analyzátoru. Poslední možností je uložení dekódovaných dat ve formátu csv nebo txt.

Hlavní nevýhodou tohoto řešení je vysoká pořizovací cena, protože utilita N5391B není standardní součástí všech osciloskopů řady 9000. V cizině je cena u prodejců, kteří uvádějí veřejně na webu cenu, v přepočtu mezi 37 000 Kč až 47 000 Kč.



Obrázek 2: Dekódování I2C GW Instek

Řešení od firmy GW Instek je součástí osciloskopu v případě, že osciloskop je vybaven logickým analyzátorem. V opačném případě při dokoupení modulu logického analyzátoru se jeho cena pohybuje v přepočtu přibližně okolo 11 000 Kč. Z toho jednoznačně vyplývá, že dekódování sběrnic lze použít pouze s logickým analyzátorem. Umožňuje dekódování sériové linky UART, I2C, SPI a paralelní sběrnici. Jediné jak zobrazit dekódovaná data je zobrazením přímo v měřeném průběhu a nelze je uložit, viz Obrázek 2: Dekódování I2C GW Instek. V tomto případě jsme ale limitováni poměrně malou velikostí paměti tohoto osciloskopu.

Firma Rigol do svých osciloskopů, stejně jako firma Agilent, dodává nástroj pro dekódování sběrnic jako volitelnou součást. Cena tohoto řešení dosahuje v přepočtu přibližně 10 000 Kč. V českých internetových obchodech stojí cca 14 500 Kč včetně DPH. Stejně jako řešení od firmy Agilent se dekódovaná data zobrazují v tabulce s tím rozdílem, že je vypsán pouze čas, adresa, směr přenosu dat, data a potvrzovaní ACK/NACK, viz Obrázek 3: Dekódování I2C Rigol. Opět i zde funguje nastavení triggeru podle posílaných dat na zkoumané sběrnici. Export dekódovaných dat je možný pouze ve formátu csv.

Obrázek 3: Dekódování I2C Rigol

Všechna tři předešlá řešení umožňují jenom základní dekódování dat s jedinou nejzákladnější detekcí chyb a to kontrolou ACK bitu. Tato kontrola ale nemusí být dostatečná pro náročné uživatele, nebo pro měření sběrnice v případě, že komunikace nefunguje správně. To lze u řešení od výrobců odstranit pouze ručním dekódováním chyb. Hlavní výhodou všech řešení je to, že jsou vyvinuty samotnými výrobci, jejich implementace přímo v osciloskopu a online dekódování již během měření.

2.3 Sběrnice I2C

Tato kapitola obsahuje základní informace o sběrnici I²C, doplněné především o informace a údaje, které souvisejí s vyvíjeným analyzátorem a jeho požadovanými vlastnostmi.

2.3.1 Obecné informace

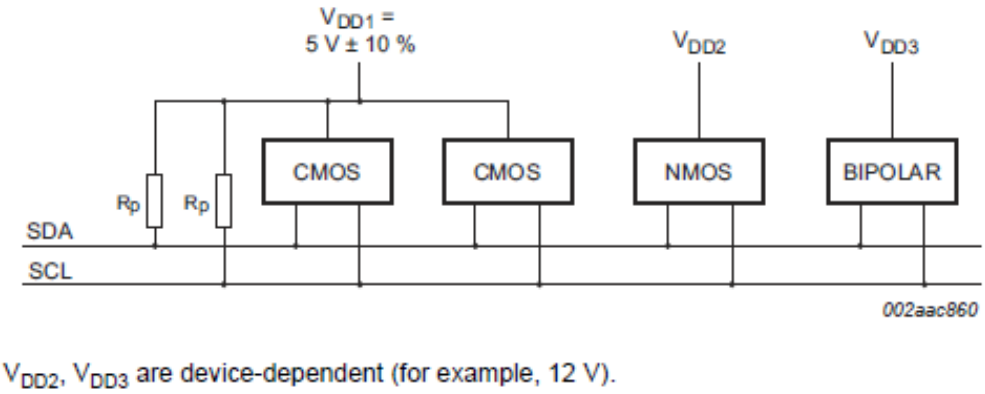
Jedná se o sériovou sběrnici, která využívá tři vodiče: SDA, SCL a GND. Komunikace probíhá pouze po dvou vodičích, vodiči SDA, na kterém se přenášejí data a dále pomocí vodiče SCL, někdy uváděného jako CLK, který generuje hodinový signál. Jelikož se využívá jen jeden datový vodič, je tato sběrnice polo-duplexní a je proto nutné celou komunikaci řídit. O řízení sběrnice se stará zařízení typu master, viz 2.3.3. Sběrnice se tedy dá provozovat v režimu single-master nebo režimu multi-master. (Tišnovský, 2009)

Komunikace probíhá v tzv. paketech, kdy každý paket obsahuje adresu zařízení, řídící bit, který je ve stejném bytu s adresou a data. Veškerá odesílaná data včetně adresy mají standardní šířku 8 bitů. Rychlost je pevně stanovená specifikací viz (NXP_Semiconductors, 2014), kde je celkový přehled všech možných rychlostí. Standardní rychlostí, kterou musí podporovat všechna zařízení, je tzv. Standard mode o rychlosti 100 kbit/s, většina zařízení ale umožňuje komunikaci i v rychlejším režimu FAST rychlostí 400 kbit/s až 1 Mbit/s, nebo dokonce rychlostí 3,4 Mbit/s v High-speed režimu. (NXP_Semiconductors, 2014)

Délka sběrnice a vodičů SDA a SCL je specifikací omezena kapacitou vodičů, kdy nesmí přesáhnout 400 pF pro rychlosti do 1 Mbit/s a pro rychlosti nad 1 Mbit/s platí maximální kapacita vodičů 550 pF. Kapacity vodičů způsobují, že změny úrovně signálu nejsou ostré, ale oblé. Při vysokých kapacitách je signál natolik deformován, že může docházet k překročení kritických hodnot časování náběžných a sestupných hran signálu. (NXP_Semiconductors, 2014)

Na obou vodičích musí být přítomen pull-up rezistor, který na sběrnici v klidovém stavu udrží logickou úroveň 1. Pull-up rezistor se obvykle připojuje na napájecí napětí 5 V, nebo je připojen na nejnižší napájecí napětí, které napájí zařízení připojené ke sběrnici.

Díky tomu ke sběrnici mohou být připojená zařízení s různými napájecími napětími. Jednotlivá zařízení tak pouze připojují příslušný signál SDA nebo CLK ke společnému vodiči GND při vysílání logické nuly. Pull-up rezistory zajistí, že na sběrnici bude v ostatních případech a v klidovém stavu potřebné napětí a tudíž logická úroveň jedna. Zapojení viz Obrázek 4: Připojení zařízení ke sběrnici I2C převzatý ze specifikace sběrnice I2C (NXP_Semiconductors, 2014).



Obrázek 4: Připojení zařízení ke sběrnici I2C (převzato z (NXP_Semiconductors, 2014))

Velikost pull-up rezistoru je závislá na napájecím napětí V_{DD}, maximálním výstupním napětí logické 0 $V_{OL(max)}$, kapacitě vedení C_b, proudu tekoucím rezistorem při logické 0 I_{OL} a na maximální době náběhu z logické 0 do logické 1 na SDA i SCL t_r, která je různá pro každou rychlost komunikace na sběrnici. Výpočet minimální a maximální velikosti pull-up rezistoru se provede podle následujících vzorců (NXP_Semiconductors, 2014):

$$R_{p(max)} = \frac{t_r}{0{,}8473 \times C_b}$$

$$R_{p(min)} = \frac{V_{DD} - V_{OL(max)}}{I_{OL}}$$

2.3.2 Adresy a adresace zařízení

Adresy pro zařízení mohou mít dvě podoby, 7bitovou adresu, nebo 10bitovou adresu. Z toho vyplývá, že 7 bity se může adresovat až 128 zařízení, ale ve skutečnosti je to jen 111 zařízení, protože jsou vyhrazeny tzv. speciální adresy, které daná zařízení nemohou užívat. U 10bitové adresy se může adresovat plných 1024 zařízení. Jednou ze speciálních adres je adresa 0b1111 0xxR/W, tedy čtyři jedničky, nula, nejvyšší dva bity 10bitové adresy a bit určující směr komunikace se slave zařízením. Tato speciální adresa slouží k odeslání 10bitové adresy, kdy se druhý byte adresy odešle v následujícím datovém bytu.

Další důležitou adresou je adresa 0b0000000X, díky které lze odeslat tzv. broadcast a data poslat všem zařízením. Všechny speciální adresy jsou uvedeny níže, viz Tabulka 2: Speciální adresy (Embedded Systems Academy).

Tabulka 2: Speciální adresy

Adresa	R/W	Význam
0b0000 000	0	Broadcast
0b0000 000	1	START byte
0b0000 001	X	Rezervování pro C-Bus formát
0b0000 010	X	Rezervováno pro jiný formát sběrnice
0b0000 011	X	Vyhrazeno pro budoucí účely
0b0000 1XX	X	Vyhrazeno pro budoucí účely
0b1111 1XX	X	Vyhrazeno pro budoucí účely
0b1111 0XX	X	Pro 10bitové adresování

2.3.3 Typy zařízení

Na sběrnici se připojují dva typy zařízení. Zařízení typu master, které řídí komunikaci na sběrnici a vždy generuje hodinový signál. Počet zařízení typu master je podle toho zda se sběrnice používá v režimu single-master nebo multi-master. U single-master je jen jedno zařízení typu master a u multi-master je jich několik. Zároveň ale musí být alespoň jedno zařízení typu master přítomno. Zařízením typu master je obvykle mikroprocesor, nebo PC pomocí převodníku. Oproti tomu zařízení typu slave může být na sběrnici přítomno od jednoho až po 111 při 7bitovém adresování nebo až 1024 při 10bitovém adresování. Zařízení typu slave nemůže požádat o komunikaci a ani ji začít, může jenom komunikaci mezi sebou a zařízením master ukončit, nebo pozdržet. Komunikaci pozdrží tím, že hodinový signál podrží v logické úrovni 0 a tím získá čas na zpracování přijímaných dat. (Devantech Ltd (Robot Electronics)) (Embedded Systems Academy)

Jelikož je možné sběrnici provozovat v multi-master módu, tak je nutné zavést základní arbitraci. Arbitrace na I2C funguje vcelku jednoduše, kdy zařízení, které vysílá, zároveň musí číst aktuální data na datovém vodiči a když zjistí rozdíl oproti svým odeslaným datům, musí okamžitě přerušit své vysílání. Rozdíl se většinou pozná během prvních několika bitů, kdy se odesílá adresa slave zařízení a kolizi zjistí ten master, který se snaží vyslat logickou úroveň 1 na SDA, přičemž na SDA je logická 0 od druhého zároveň vysílajícího zařízení typu master.

2.3.4 Přenosové rychlosti

Komunikace dle specifikace probíhá v určených rychlostech daných frekvencí generování hodinového signálu a má danou přesnou podobu paketů, které se posílají.

Rychlostí komunikace je několik, z níž se nejčastěji používají dvě Standard-mode u kterého je hodinový signál generován rychlostí 100 kHz a tudíž se komunikuje 100 kbit/s. Druhou nejčastější rychlostí je Fast-mode se 400 kHz hodinového signálu. Všechny rychlosti jsou uvedeny v tabulce Tabulka 3: Rychlost sběrnice I2C.

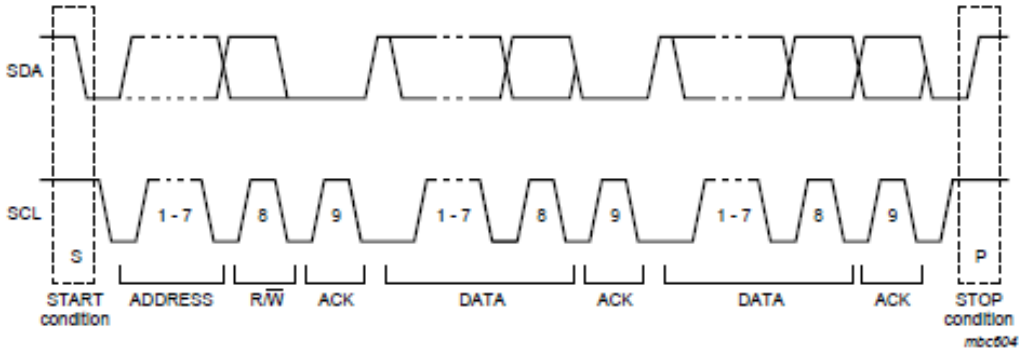
Tabulka 3: Rychlost sběrnice I2C

Rychlost	Název
100 kbit/s	Standard-mode
400 kbit/s	Fast-mode
1 Mbit/s	Fast-mode plus
3,4 Mbit/s	High-speed mode

Pro jednosměrný provoz sběrnice je ještě specifikovaná jedna rychlost 5 Mbit/s nazývaná také Ultra Fast-mode. (NXP_Semiconductors, 2014)

2.3.5 Formát dat

Jak již bylo naznačeno, komunikace probíhá v tzv. paketech, kdy každý paket začíná START podmínkou, obsahuje adresu slave zařízení, se kterou je odeslán řídící bit určující směr přenosu dat. Za adresním bytem je potvrzovací bit Acknowlege a následují jednotlivé byty dat, které jsou také potvrzovány. Jak může komunikace vypadat, viz Obrázek 5: Komunikace po sběrnici I2C převzatý ze specifikace I2C sběrnice (NXP_Semiconductors, 2014)



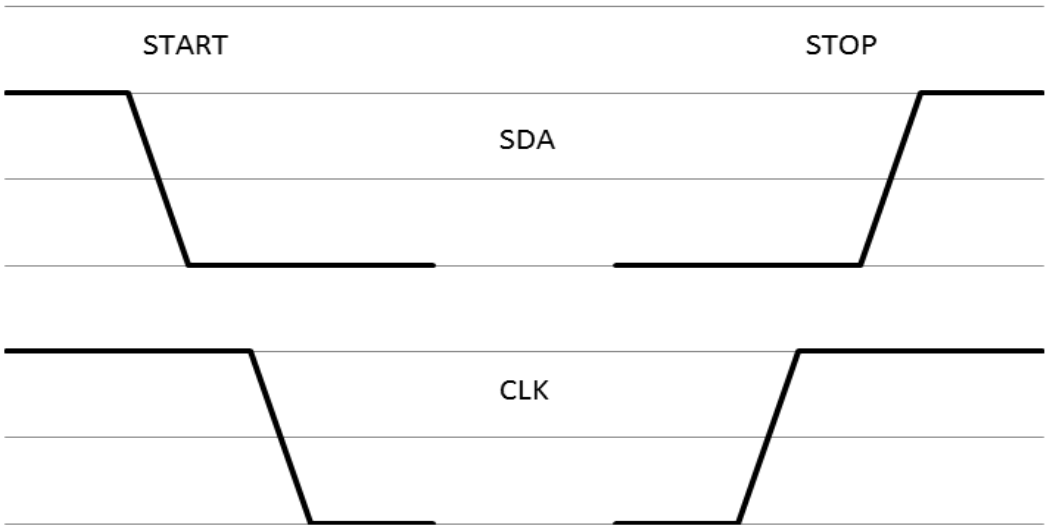
Obrázek 5: Komunikace po sběrnici I2C (převzato z (NXP_Semiconductors, 2014))

2.3.6 Pravidla komunikace

Nejdůležitější dvě pravidla jsou následující. První pravidlo zakazuje jakémukoliv zařízení začít vysílat, když jiné zařízení vysílá a slave zařízení nesmí vysílat, pokud k tomu není oprávněno zařízením typu master. Druhé pravidlo definuje, kdy jsou na datovém vodiči platná data a kdy se mohou data měnit. Úroveň na vodiči SDA se smí měnit, jen když je vodič SCL v nízké úrovni. Když je vodič SCL ve vysoké úrovni, znamená to, že data na datovém vodiči jsou platná a zařízení je mohou číst. Z tohoto pravidla jsou dvě výjimky, kdy se může úroveň na vodiči SDA změnit, při vysoké úrovni na vodiči SCL. Těmi dvěma výjimkami jsou START a STOP podmínka, které jsou takto definované. To lze vidět na obrázku Obrázek 6: START a STOP podmínka.

2.3.7 Přenos dat

Během celé komunikace platí pravidla popsaná v předchozí kapitole 2.3.6, tj. že se může měnit logická hodnota na datovém vodiči pouze v případě, že je logická nula na hodinovém signálu. Jediné dvě výjimky jsou START a STOP podmínky, která otevírají a ukončují komunikaci mezi zařízeními. START podmínku generuje master tak, že při klidovém stavu sběrnice tj. při SDA a CLK v logické úrovni 1 vygeneruje na datovém vodiči sestupnou hranu a na hodinovém ponechá logickou 1. Následně již začne generovat hodinový signál a odesílat adresu a data. STOP podmínka okamžitě ukončí přenos, všechna slave zařízení okamžitě začnou poslouchat a očekávat další START podmínku. STOP podmínka je generována vzestupnou hranou na datovém vodiči při logické úrovni 1 na hodinovém vodiči SCL. Po STOP podmínce zůstanou oba vodiče SDA CLK v logické 1 a sběrnice je v klidovém stavu. Obě podmínky jsou graficky znázorněny, viz Obrázek 6: START a STOP podmínka.



Obrázek 6: START a STOP podmínka

Po START podmínce master odešle 7 bitů adresy zařízení, se kterým chce komunikovat a osmým bitem určí, zda chce ze zařízení číst, nebo do něho zapisovat. Pokud použije 10bitovou adresaci, odešle vyhrazenou adresu ve formátu 0b11110XXR/W kde XX jsou nejvyšší dva bity 10bitové adresy a R/W je směr komunikace. V dalším bytu následně odešle zbytek adresy. S další logickou jedničkou na hodinovém signálu přečte potvrzovací bit, který zařízení odesílá po každém přijatém bytu. Když je tento bit 1 tak to znamená NACK, tudíž nepotvrzení dat, zde adresy a zařízení s touto adresou není připojeno ke sběrnici. Pokud tento bit je 0, tak to znamená ACK, tudíž že je adresa potvrzena a zařízení připraveno odesílat nebo přijímat v závislosti na předchozím bitu R/W a stavu hodinového signálu. Logická 1 v bitu R/W znamená čtení a logická 0 zápis do zařízení typu slave.

Následně se odesílají jednotlivé byty dat, kdy se každý byte musí po přijetí opět potvrdit. Když přijímající vygeneruje NACK, tudíž data nepotvrdí, znamená to konec komunikace a vysílající stanice by měla komunikaci ukončit STOP podmínkou. V praxi se ale může stát, že k ukončení dojde i bez STOP podmínky.

2.3.8 Elektrické parametry

Neméně důležité pro správnou funkci sběrnice jsou další elektrické parametry kromě kapacity vedení a správné hodnoty pull-up rezistorů. Dle závazných specifikací jsou definovány logické úrovně 0 a 1 jako určité rozsahy napájecího napětí. Konkrétně se jedná pro logickou nulu o napětí mezi $-0{,}5 \times V_{DD}$ až $0{,}3 \times V_{DD}$. Pro logickou jedničku se jedná o napětí větší než $0{,}7 \times V_{DD}$.

3 Aplikace

3.1 Výběr programovacího jazyka a vývojového prostředí

Nejprve jsem se musel rozhodnout, zda si vyberu programovací jazyk, ve kterém jsem již někdy programoval, nebo si zvolím pro mě nový programovací jazyk. Vybral jsem si možnost oživit a rozšířit si starší znalosti, protože učit se nový programovací jazyk by při vytváření bakalářské práce mohlo v konečném důsledku přinést spoustu problémů navíc. Proto mi zbylo na výběr z programovacích jazyků C, C++ a C#.

3.1.1 Programovací jazyk C

Programovací jazyk C je ze všech tří jazyků, ze kterých jsem vybíral, nejstarší a ve své podstatě nejjednodušší. Zároveň s tím je ale předchůdcem zbývajících dvou uvažovaných jazyků. Jazyk C se řadí mezi nízko úrovňové programovací jazyky a je tak velice rychlý v porovnání s dalšími vyššími programovacími jazyky. C nabízí programátorovy relativně volnou roku v tom, jak bude pracovat, především co se týče správy paměti a jejím přímým přístupem. Tím se dá dosáhnout velice dobrých optimalizací a nižších nároků na systémové prostředky. Tím jak je starý a rozšířený, není problém jeden program v C re-kompilovat na jiné platformě. Jeho rozšířenost je tak masivní, že doposud je to nejpoužívanější programovací jazyk podle webu *http://www.tiobe.com*. Hlavní nevýhodou jazyka C je velice malá typová kontrola a také to, že oproti jiným jazykům nenabízí některé tak pohodlné funkce.

3.1.2 Programovací jazyk C++

Jazyk C++ přímo vychází z programovacího jazyka C, a tak si s sebou nese veškeré jeho výhody a je s ním zpětně kompatibilní, tudíž téměř veškerý zdrojový kód napsaný v jazyce C funguje stejně v jazyce C++. Také rozšiřuje jazyk C o objekty a tak se s ním pracuje především objektově, ačkoliv v něm lze programovat strukturovaně. C++ bohužel nevylepšilo standardní knihovny a tak stále pro některé specifické aplikace nenabízí standardní knihovní funkce. Toto docela hodně omezil Microsoft svou platformou .NET i pro C++. Podle webu *http://www.tiobe.com* patří C++ mezi 5 nejpoužívanějších programovacích jazyků.

3.1.3 Programovací jazyk C#

Nejmladší z trojice programovacích jazyků. Vychází z jazyka C++ a tak si stále s sebou nese jeho výhody, které jsou ale hodně potlačovány tím, jak byl tento jazyk vyvinut a tím, že se jedná čistě o objektově orientovaný jazyk. Microsoft vyvinul C# spolu s platformou .NET, tudíž se v C# pracuje vcelku pohodlně a právě díky .NET má spoustu tříd a metod, jejíž funkcionalita chyběla ve standardních knihovnách C neb C++. Také si sám spravuje paměť, tudíž se programátor téměř o nic nemusí starat, což může být velkou nevýhodou právě tehdy, když je potřeba pracovat specificky například s velkými daty. Má tedy o dost vyšší systémové nároky na paměť i procesor a proto je pomalejší než například C.

3.1.4 Vybraný jazyk

Po posouzení kladů a záporů jednotlivých jazyků jsem pro svou bakalářskou práci nejprve zavrhnul jazyk C#, protože pro daný účel byly jazyky C a C++ podstatně lepší. Nakonec jsem zvolil jazyk C z několika důvodů. Prvním důvodem byl fakt, že jsem v C++ velice dlouho nic neprogramoval oproti C, které využívám podstatně častěji. Zároveň práce napsaná v jazyce C půjde v budoucnu v případě potřeby velice snadno použít v C++ a také, že jazyk C je ze všech tří jazyků nejrychlejší.

3.1.5 Vývojové prostředí

Nad vývojovým prostředím jsem příliš dlouho neváhal, ačkoliv výběr je veliký. Zvolil jsem si Visual Studio 2010 Professional, protože jsem ho již měl nainstalované v rámci MSDN AA a programoval jsem v něm aplikace v různých programovacích jazycích.

3.2 GUI

Uživatelské rozhraní bylo zvoleno v podobě klasického příkazového řádku (konzole), protože výsledná aplikace nepotřebuje žádná složitá nastavení. Jediné co od uživatele požaduje je zadání vstupního souboru s naměřenými daty, názvu výstupního html souboru, parametrů napěťového pásma, popřípadě výběr které naměřené kanály se mají použít v případě, že jich je naměřených více než dva.

Rovněž je možno před zkompilováním programu pomocí definice vybrat, v jakém jazyce bude výsledné uživatelské rozhraní. Na výběr je z jazyka českého a anglického. Čeština byla vybraná proto, aby se aplikace pohodlně používala českým uživatelům, kteří nemají anglický jazyk v oblibě. Anglický jazyk byl vybrán z důvodu použitelnosti aplikace kýmkoliv, kdo ovládá alespoň základy anglického jazyka, a tím se aplikace stala univerzálně použitelnou ve kterékoliv zemi.

3.3 Program

Z požadavků na výslednou aplikaci, ať již ze zadání nebo úvodu, jsem vymyslel základní koncept, jak bude program fungovat. Celý problém s analýzou dat jsem rozdělil do několika dílčích částí, které vykonávají jednotlivé funkce. Celým návrhem jsem se snažil postupovat systematicky tak, abych co nejvíce oddělil konkrétní data spjatá s osciloskopem, v mém případě osciloskopem Agilent DSO9254A, respektive osciloskopy řady Agilent 9000. Po oddělení se od těchto dat a uložení si potřebných informací pro další zpracování a reprezentaci výsledků. V praxi to znamená, že pro přidání dalšího osciloskopu bude stačit upravit funkci obstarávající uživatelské rozhraní a doprogramovat jednu funkci, která zpracuje data z přidávaného osciloskopu, a jejím výstupem budou data v podobě zpracovatelné dalšími funkcemi, které zůstanou stejné.

O trochu větší zásah do programu by znamenalo přidání dalšího typu sběrnice. V tomto případě by se přidala další funkce pro její dekódování, respektive sada funkcí. Ve funkcích zajišťující načítání dat by stačilo provést pouze drobné úpravy.

3.3.1 Struktura programu

Návrh struktury programu jsem tedy rozdělil do několika funkcí, které se obsluhují z funkce *main*. Oddělil jsem obsluhu uživatelského prostředí, respektive pouze uživatelský vstup do zvlášť funkce, protože vždy uživatelský vstup je potencionálním nebezpečím pro aplikaci z důvodu vstupu neočekávaných chyb. Následná část zajišťuje správné otevření, načtení hlaviček souboru a jejich kontrolu s případným dalším dotazem na uživatele. Další funkcí bude samotné načtení dat do paměti včetně nutného zpracování pro snížení systémových nároků. Z logického hlediska by další částí mělo být samotné dekódování dat, ale

ještě před ním je zařazena funkce na automatickou detekci rychlosti dekódované sběrnice. Poslední částí je interpretování dekódovaných dat a chyb do dvou oddělených souborů.

Dva oddělené výstupní soubory byly zvoleny z důvodu přehlednosti celého řešení. První výstupní soubor obsahuje veškeré dekódované informace včetně všech zjištěných chyb. V druhém souboru je uložena jen část informací a to konkrétně jen chybové stavy v celé komunikaci.

3.3.2 Struktury

Pro všechny hlavičky v binárním souboru, který generují osciloskopy Agilent řady 9000 jsem si udělal struktury podle dokumentace (Agilent Technologies, 2014). Jenže pro správnou funkcionalitu při načítání bylo nutné strukturu hlavičky naměřeného signálu a strukturu hlavičky dat signálu sloučit do jedné struktury, protože když se načítaly zvlášť, docházelo mezi načtením jedné a druhé hlavičky k posunu o dva bajty způsobeném zarovnáváním dat do bloků.

Další důležitou strukturou je výsledná struktura, do které se ukládá informace vždy o jednom paketu, tudíž čas, kdy paket začíná, čas mezi dvěma následujícími pakety, adresa slave zařízení. Přítomna je také proměnná, do které se zapisuje příznak chyby, které byla v daném paketu dekódovaná, směr komunikace mezi master a slave zařízeními a v neposlední řadě pointer na pole posílaných dekódovaných dat.

3.3.3 Definice a proměnné

Nejdůležitější proměnné v celém programu je šest pointerů. Jeden pointer ukazuje na pole bajtů reprezentující naměřený datový signál, druhý ukazuje na pole bajtů obsahující hodinový signál, třetí ukazuje na pole struktur, do kterých se ukládají dekódovaná data. Další tři pointery ukazují v paměti na místa, kde jsou uloženy řetězce s cestou ke vstupnímu a dvěma výstupním souborům.

Pomocí definic jsou definovány důležité parametry programu, které slouží k jeho správné funkci a jednoduché a rychlé modifikaci, například k výběru jazykové verze, definování rychlosti sběrnice podle normy nebo podrobného výpisu dat do konzole.

3.3.4 Ovládání aplikace

Aplikace lze spustit a ovládat dvěma způsoby. Prvním způsobem je spouštění s parametry a druhým způsobem spuštění bez parametrů.

Při spuštění bez parametrů, nebo se špatnými parametry aplikace požaduje uživatelský vstup v konzoli. Každý požadovaný vstup je název souboru včetně přípony, případně s celou cestou k souboru. Prvním požadovaným názvem je vstupní soubor, druhým je výstupní soubor, třetím žádaným názvem je výstupní soubor, do kterého se uloží pouze chybové pakety. Dalšími vstupy jsou minimální a maximální napětí, které definují napěťové pásmo, vně kterého se bude detekovat chybový stav. Tento chybový stav se detekuje až po době definovaného časem, který se zadává v milisekundách jako poslední parametr.

Při spuštění s parametry je na výběr ze dvou možných kombinací parametrů. První kombinace obsahuje identifikační číslo 1, název vstupního souboru, název výstupního souboru, název výstupního souboru, který bude obsahovat pouze chybové pakety. Dalšími třemi vstupy jsou minimální a maximální napětí a čas v milisekundách, které definují pravidlo pro detekci chyby signálu vně napěťového pásma po minimální dobu definovanou vstupním časem. Všechny tři názvy souborů je nutné vyplnit včetně přípony. Je také možné použít celou adresní cestu k souborům. Druhá možnost parametrů obsahuje pouze identifikační číslo 2, název vstupního souboru. Dalšími třemi vstupy jsou minimální a maximální napětí a čas v milisekundách, které definují pravidlo pro detekci chyby signálu vně napěťového pásma po minimální dobu definovanou vstupním časem. Opět platí pro vstupní soubor to samé jako v předešlých případech, že je nutné vyplnit včetně přípony, ale je také možno použít celou cestu k souboru. Názvy obou výstupních souborů se vygenerují na základě vstupního souboru a budou uloženy ve stejné složce.

3.3.5 Funkce

Main

Funkce *main* obstarává veškerý chod aplikace pomocí volání jednotlivých funkcí. Zároveň s tím inicializuje veškeré globální proměnné a zavádí další lokální proměnné. Zároveň zajišťuje správné předávání dat mezi jednotlivými funkcemi tak, aby se nejen šetřily systémové prostředky a zároveň s tím se co nejvíce omezil vstup uživatele do programu a tím se celý proces co nejvíce automatizoval a byl uživatelsky co nejjednodušší.

Zároveň se ve funkci *main* měří čas jednotlivých kroků programu a ten se spolu s tím, co aplikace právě vykonává, vypisuje do konzole. Uživatel tak má vždy přehled o tom, která část programu právě probíhá a jaký je k tomu potřebný čas.

OpenAgilent9000

```
void OpenAgilent9000(FILE *input, char **data_ptr, char
**clk_ptr, float *noiseLevel, float low_max, float high_max);
```

Z definované univerzálnosti a snadného rozšíření se tato funkce nejmenuje OpenFile, ale OpenAgilent9000. Kdyby se přidal další osciloskop, respektive řada osciloskopů, stačí přidat pouze jeho funkci. Zároveň s tím by pro jednodušší správu volání funkcí pro jednotlivé osciloskopy bylo vhodné vytvořit funkci OpenFile.

Vstupem této funkce je otevřený datový proud ke vstupnímu souboru v binárním režimu, pointery na pointery, které budou ukazovat na pole vzorkovaných dat a hodin. Globální pointery, na které ukazují vstupní pointery, zatím nikam neukazují, a až po zjištění velikosti souboru, respektive množství vzorkovaných dat je alokováno v paměti místo a lokální pointery začnou na tato místa ukazovat. Protože je nutné, aby data byla přístupná globálně, musí se hodnota lokálních pointerů ukazujících na pole hodnot uložit do globálních pointerů. Jelikož ale vstupem jsou pointery na tyto globální pointery, můžeme na adresy globálních pointerů uložit adresy na pole hodinového a datového signálu. Následuje pointer na proměnnou typu float, do které se uloží velikost šumu. Posledními dvěma vstupy jsou hodnoty minimálního a maximálního napětí, definující napěťové pásmo, které by signály SDA a SCL neměly překročit.

Nejprve se tedy přečte hlavička celého souboru, viz 2.2.1 a z ní se určí, zda se jedná o data ze správného osciloskopu pomocí tzv. cookie parametrů nebo identifikátorů formátů souborů. Zároveň s tím se provede kontrola, zda jsou přítomny alespoň dva naměřené signály z důvodu, že sběrnice I2C využívá pro svou komunikaci dvou vodičů viz 2.3.1 Pokud test neproběhne v pořádku, vypíše se do konzole příslušné chybové hlášení a funkce skončí. V opačném případě se přečtou další dvě hlavičky pro první signál a v případě, že soubor obsahuje více než dva naměřené signály, zjistí se od uživatele, které dva z nich se mají pro dekódování použít. Následně se zkontroluje, zda formát měřených dat odpovídá podporovaným formátům, v tomto případě normální nebo neznámý časový průběh signálu a postupně se přečtou vybrané dva kanály pomocí funkce *ReadBinaryFloatData*. Z výstupů funkce *ReadBinaryFloatData*, které určují počet logických nul v přečteném signálu, se jednoduchým algoritmem určí, který ze dvou signálů je hodinový a který datový.

Algoritmus spočívá v tom, že signál s větším počtem logických nul je signál hodinový, protože hodinový signál se s každým odeslaným bitem po sběrnici mění do logické úrovně 0 a logické úrovně 1, kdežto datový signál může zůstat stále v jedné úrovni nebo se změnit. Jelikož je nepravděpodobné, že by se v datech vždy v sousedních dvou bitech změnila logická hodnota po celou dobu komunikace, lze považovat detekování typu signálu podle počtu vzorků v logické úrovni 0 za spolehlivé. Zároveň toto řešení není nikterak náročné na výpočetní výkon a ani na velikost potřebné operační paměti.

ReadBinaryFloatData

```
int ReadBinaryFloatData(char *data_array, FILE *input, unsig-
ned int load_points, unsigned int scale, float *noiseLevel,
float low_max, float high_max);
```

Oproti předchozí funkci *OpenAgilent9000* se zde již neřeší žádné hlavičky, ale samotné načtení a zpracování dat do formy, pro kterou jsou připraveny následující funkce.

Vstupem je pointer na již alokované prázdné pole osmibitových hodnot char, otevřený proud na vstupní soubor s aktuální pozicí od které začínají načítaná data. Dalším vstupem je počet vzorků, které se mají načíst, zpracovat a přidat do pole. Dalším vstupem je *scale*, který určuje, kolikátý prvek se má vždy načíst. Zvedení měřítka bylo nutností z důvodu možnosti různé vzorkovací frekvence jednotlivých kanálů, ačkoliv se to pro tento případ nedoporučuje. Pointer **noiseLevel* na proměnnou typu float, slouží pro uložení velikosti šumu. Posledními dvěma vstupy jsou hodnoty minimálního a maximálního napětí, definující napěťové pásmo, které by signály SDA a SCL neměly překročit.

Jak již bylo několikrát zmíněno v předcházejících kapitolách, je kladen důraz na systémové nároky, mezi které samozřejmě kromě náročnosti na procesorový čas patří nároky na operační paměť RAM. Pro navržení optimálního algoritmu bylo nutné si stanovit svoje požadavky a z nich následně vycházet. Pro rychlost zpracování dat by bylo vhodné, kdyby se povedlo celý naměřený soubor z osciloskopu nahrát do operační paměti, která je řádově rychlejší než čtení z magnetického pevného disku. Jenže v případě velikých souborů řádově GB dat by nebylo toto řešení nejlepší. Dalším faktorem ovlivňujícím rychlost zpracování je samotné dekódování dat, které ale při nahrání celého souboru v operační paměti je velice rychlé, protože dnešní procesory jsou výkonné a rychlost práce s daty v operační paměti je rychlá. Proto tedy bylo nutné vymyslet způsob jak nahrát celý soubor do paměti, aniž by se ztratila jakákoliv důležitá informace a zároveň aby bylo potřeba méně paměti, než má originální naměřený soubor.

Z požadavků na rychlost a paměťovou náročnost jsem vymyslel, že by bylo vhodné originální naměřená data, která jsou v datovém typu float a určují napěťovou úroveň, převést na menší datový typ. Zároveň s tím by se mohla veškerá originální data načítat postupně, aby se před převodem na menší datový typ nemusel načítat celý měřený průběh a tím se

ušetřila používaná operační paměť během načítání, která se po načtení průběhu uvolní. Proto funkce *ReadBinaryFloatData* nejprve načte část souboru definovanou v úvodu programu, u které zjistí, mezi jakými napětími se měřený signál nachází a z těchto maxim vypočítá napěťové úrovně pro logickou 0 a logickou 1. Tyto napěťové úrovně se vypočítají jako 0,3 násobek Ucc pro logickou 0 a 0,7 násobek Ucc pro logickou 1. Následně načtenou část převede na pseudo-logické úrovně, kdy logické úrovni 0 odpovídá uložená hodnota 0, logické úrovni 1 odpovídá hodnota 1 a zakázanému stavu odpovídá hodnota 2. Při překročení napětí vně definovaného napěťového pásma se použije hodnota 3. Jelikož programovací jazyk C standardně nenabízí datový typ BOOL, které by byli potřeba pro detekci zakázaného stavu dva, použil jsem datový typ char, který je 8bitový. Tímto jsem docílil zmenšení paměťových nároků na 25 %, přičemž by šlo využít 4 naměřených vzorků do jednoho prvku pole o datovém typu char a tím snížení paměťových nároků na pouhých 6,25 %. Kdyby se ale neuvažovalo v programu detekování chyb, mezi které patří i dlouhé setrvání signálu v zakázaném stavu, nebo překročení definovaných napětí, dalo by se docílit hodnoty 3,125 % původní velikosti. Snížení paměťových nároků pod 25 % původní velikosti by s sebou ale neslo zpomalení aplikace, protože pro práci s jednotlivými vzorky by se nejprve muselo provést maskování a potřebná rotace dat než by s jednotlivým vzorkem šlo dále pracovat. Proto jsem zvolil kompromisní řešení s teoretickým snížením paměťových nároků na 25 % původní velikosti.

Kromě snížení paměťových nároků z původní velikosti souboru na 25 % se tato funkce stará také o měření šumu signálu, které se provádí tak, že se spočítá průměrná hodnota napětí v logické 1 a v logické 0 z načtených dat a detekuje se maximální odchylka načítaných dat od tohoto průměru pro obě napěťové úrovně. Vyšší z absolutních hodnot šumů se uloží a na konci programu zapíše do výsledných souborů.

Návratovou hodnotou této funkce je počet logických nul, které se spočítají při samotném převodu jednotlivých načtených vzorků z napětí do hodnot 0, 1, 2 a 3. Této důležité informace se využije ve funkci *OpenAgilent9000*, kde se na základě této informace určí, který signál je hodinový a který datový.

DecodeSpeed

```
int DecodeSpeed (char *data_array, unsigned int size,
 float sample_time);
```

Funkce *DecodeSpeed* je jednou z jednodušších funkcí, která jakou svůj vstup má pointer na pole načtených a připravených dat pro dekódování, neznaménkové 32bitové číslo určující počet vzorků v poli a proměnná *sample_time* obsahující informaci o čase mezi dvěma sousedními vzorky.

Výstupem je hodnota rychlosti sběrnice, respektive frekvence hodinového signálu, nebo jedna ze záporných hodnot. Návratová hodnota -1 značí, že detekovaná rychlost sběrnice není v souladu se specifikací sběrnice I2C. Návratová hodnota -2 naopak signalizuje, že se nepodařilo správně detekovat rychlost sběrnice, respektive nebylo vůbec možno určit její rychlost, protože naměřený a načtený vzorek dat hodinového signálu nebyl dostatečně dlouhý, nebo během měření nedošlo k žádné komunikaci.

Algoritmus, který je v této funkci použit využívá změn logické úrovně hodinového signálu a informace o času mezi dvěma sousedními vzorky. Nejprve je nutné otestovat, zda je hodinový signál v logické jedničce. Pokud není, postupně se projde část pole signálu až k logické jedničce. Následně se hledá sestupná hrana signálu, od které začne samotné měření, respektive detekování rychlosti sběrnice. Hodnota indexu pole, kdy je detekována sestupná hrana signálu se uloží do proměnné a dále se prochází pole hodnot, až se nalezne vzestupná a následující sestupná hrana. Tím se zaručí, že se nalezla právě jedna perioda hodinového signálu. Rozdílem koncového a počátečního indexu se spočte počet vzorků na jednu periodu hodinového signálu. Z počtu vzorků jedné periody hodinového signálu a časové informace, jaký je rozdíl mezi jednotlivými vzorky se spočte frekvence hodinového signálu podle následujícího vzorečku:

$$f = \frac{1}{(i_{STOP} - i_{START}) \times XIncrement}$$

Spočtená frekvence nevyjde zcela přesně se specifikací sběrnice, proto je nutné ještě vypočtenou frekvenci otestovat, zda se nachází v toleranci jednotlivých frekvencí sběrnice ze specifikace. Pokud odpovídá toleranci je návratovou hodnotou přesná frekvence v Hz dle specifikace. Pokud ale neodpovídá toleranci, je výstupem již zmiňovaná -1.

DecodeData

```
unsigned int DecodeData (char *data_array, char *clk_array,
int speed, ChannelHeaderAgilent9000 header, Result
**result_array, Result last_packet, char *status, unsigned int
max_samples);
```

Další důležitou funkcí je *DecodeData*, která má mnoho vstupních proměnných. První dvě proměnné jsou pointery na pole načtených a připravených naměřených signálů. Následně další je hlavička jednoho kanálu pro možnost dalších úprav a zrychlení dekódovacího algoritmu. Dalším vstupem je pointer na pointer pole struktur výsledných dekódovaných paketů dat. Zde je nutné, aby se jednalo o pointer na pointer, obdobně jako v případě funkce *OpenAgilent9000* protože během načítání je nutné pole výsledných struktur re-alokovat, protože předem není známo, kolik dat mezi kolika zařízeními bude na měřené a dekódované sběrnici posíláno. Jelikož při realokaci pole je možné, že nové re alokované pole nebude na stejném místě, je nutné změnit globální ukazatel z původního na nové pole. Toho lze docílit právě ukazatelem na globální ukazatel. Dalším vstupem je minulý paket a proměnná *status*, které slouží pro budoucí rozšíření, kdy by bylo nutné načítat a dekódovat data po menších částech, nebo by celistvá data byla uložena ve více souborech. Tato funkcionalita je již připravena, ale ve výsledném programu zatím není implementována. Posledním vstupem je proměnná *max_samples*, určující maximální počet vzorků, které mohou být mimo definované napěťové pásmo, než se detekuje a uloží chybový stav.

Samotné dekódování dat je rozděleno do několika dílčích částí a využívá se k nim dalších menších funkcí.

Nejprve před samotným dekódováním se inicializují důležité proměnné pro správnou funkčnost dekódování. Mezi hlavní proměnné tak patří řídící proměnná *index*, proměnné počítající souvislý počet vzorků v zakázaném stavu hodinového nebo datového signálu. Proměnná *decoded_count* zaznamenávající počet dekódovaných paketů a funguje také jako index pole struktur s výsledky. Dále se načte z pointeru na pointer *result_array_ptr* adresa, na kterou ukazuje globální pointer, respektive adresa první struktury z pole struktur. Jelikož pointer na začátku ještě nikam neukazuje, alokuje se pole o výchozí velikosti 100 struktur. Zároveň s těmito nejdůležitějšími pointery a proměnnými se alokují ještě ostatní proměnné.

Samotné dekódování dat včetně detekce základních chyb probíhá podle následujícího algoritmu.

Nejprve se v obou signálech hledá start podmínka, kdy datový signál změní svou úroveň z logické jedničky na logickou nulu a zároveň s tím je hodinový signál stále v logické úrovni jedna. Po nalezení start podmínky se nalezne také sestupná hrana hodinového signálu, z indexu a znalosti vzorkovacího kroku dopočítá čas, který uběhl od počátku měření a ten se zapíše do aktuálního prvku pole struktur s výsledky.

Následně přijde optimalizace, která zvýší rychlost dekódování dat. Místo aby se dále postupovalo po jednotlivých vzorcích v polích, tak se nejprve index zvětší o počet vzorků, odpovídající 0,75 doby jednoho hodinového signálu a následně se teprve po jednom vzorku hledá další relevantní hodnota v poli odpovídajícím datovému signálu. Výpočet o kolik se má zvětšit index je následující:

$$step = \frac{1}{f * XIncrement} \times 0{,}75$$

Kde f je frekvence, respektive rychlost sběrnice, kterou vrátila funkce *DecodeSpeed*, XIncrement je časový rozdíl mezi jednotlivými vzorky uložený v hlavičce měřeného kanálu. Hodnota 0,75 je empiricky zjištěná jako optimální pro daný účel, jelikož se pomocí takto velké konstanty přeskočí relativně velké množství vzorků a zároveň se nemůže stát, že by se přeskočila následující náběžná hrana na hodinovém signálu.

Relevantní hodnota je právě tehdy, když hodinový signál je v logické jedničce, k čemuž se využívá detekování náběžné hrany na hodinovém signálu. To již obstarávají funkce *DecodeAddress* a *DeccodeByte.*

V obou těchto funkcích *DecodeAddress* a *DeccodeByte* je samozřejmě implementována detekce základních chyb. Funkce *DecodeAddress* samozřejmě automaticky dekóduje sedmi nebo desetibitovou adresu.

Detekce probíhá mezi jednotlivými bity přenosu, kdy se testuje, zda není mnoho vzorků v zakázaném stavu, nebo mimo definované napěťové pásmo. Nějaké vzorky se mohou objevit v zakázaném pásmu, například když osciloskop vzorkuje několikanásobně vyšší vzorkovací frekvencí než je frekvence sběrnice. Dále se samozřejmě detekuje přítomnost, zařízení s dekódovanou adresou. Když zařízení chybí, je uložen do struktury daného paketu kromě adresy také chybový příznak a již se ani další data v tomto paketu nedekódují. To je zajištěno výstupem funkce *DecodeAddress*, která vrací hodnotu 0, když se nepovedlo dekódovat adresu, -1 když zařízení není připojeno a 1 když vše proběhlo v pořádku. U funkce *DecodeByte* jsou pouze dvě výstupní hodnoty 0 pro neúspěšné dekódování dat a 1 pro úspěšné. Ještě se detekuje správnost odesílaných dat, v případě že data nebyla přijata správně, nebo nebyla odeslána celá, opět se nastaví chybový příznak.

RepresentResult

```
void RepresentResult (FILE *output, FILE *output_err, Result
*result, unsigned int count, char end_file, float
*noiseLevel);
```

Poslední důležitou funkcí je funkce *RepresentResult*, která ukládá dekódovaná data do výstupních HTML souborů. Jsou zvoleny dva výstupní soubory pro větší přehlednost, a proto také funkce má jako vstupní parametry dva otevřené streamy dat. Dalším vstupem je pointer na pole struktur obsahující dekódovaná data, které následuje počet struktur v poli. Předposledním vstupem je pouze osmi bitová proměnná *end_file*, která když je v ní uložena binárně hodnota 1 značí, že po uložení dat se má uložit i patička souboru. To je pro případ dalšího rozšíření, kdy by mohla být naměřená data ve víc souborech, nebo by se mohla data dekódovat a ukládat po částech pro další optimalizaci nároků na operační paměť. Poslední vstup obsahuje informaci o velikosti šumu signálu.

Samotné ukládání dat probíhá tak, že se nejprve uloží hlavička souboru obsahující informace pro uživatele o daném měření. Následně se postupně procházejí všechny struktury v poli a podle uložených informací se vypisují jednotlivé řádky s informacemi o datech včetně podbarvení jednotlivých políček, respektive řádků na základě detekované chyby. Zároveň s tím se do druhého souboru vypisují pouze pakety, které obsahují jakoukoliv chybu. Tudíž výstupem je kompletní výpis a chybový výpis.

3.4 Výstup

Výstupem z programu jsou dva HTML soubory, jak již bylo zmíněno. Jedná se o soubor s kompletním výpisem dekódovaných dat a soubor pouze s výpisem paketů dat, ve kterých byla detekována chyba. Toto rozdělení vzniklo z praktického důvodu, protože z velké části případů se měření provádí v případě, že nějaká část elektroniky nefunguje. Proto je tedy vhodné, aby byl přehledně jeden soubor s chybami a ty pak nemuseli hledat ve všech dekódovaných datech, což by zpomalovalo práci a ztížilo hledání případné souvislosti mezi chybami.

3.4.1 Formát

Oba soubory jsou ve formátu HTML. Tento formát byl zvolen z několika důvodů. Prvním důvodem byla jeho univerzálnost, jelikož ho lze otevřít v jakémkoliv webovém prohlížeči na kterékoliv platformě. Není tak nutné mít další specializovaný program pro prohlížení výsledků. Dalším důvodem byla jednoduchost a především dřívější zkušenost tvorby webu v HTML. Zároveň s rozšířeností HTML je velice snadno dostupná a přehledná literatura, jak již v klasickém knižním (Domes, 2005), nebo webovém vydání. Posledním zásadním důvodem volby právě HTML bylo to, že o grafickou reprezentaci výsledků se postará webový prohlížeč.

3.4.2 Hlavička souborů

V hlavičce obou výstupních souborů je na začátku tabulka obsahující důležité údaje vyčtené z hlaviček vstupního souboru. Jedná se o typ a sériové číslo osciloskopu, O typ a velikost vstupního souboru, data a času měření a v neposlední řadě také informace o rychlosti měřené sběrnice.

Následuje jednoduchá legenda, která určuje, jakou barvou bude zvýrazněn konkrétní chybový stav. Jedná se tedy o barvu červenou pro nejzásadnější chybu, že se zařízení typu master snaží komunikovat se zařízením typu slave, které není připojeno. Méně zásadní chybou je pomalý přechod z logické úrovně jedna na logickou úroveň nula a naopak, respektive napětí uvnitř zakázaného pásma po uživatelem definovanou dobu. Ta je zvýrazně-

na oranžovou barvou. Když se napětí dostane vně uživatelsky definované napěťové pásmo, je paket zvýrazněn růžovou barvou. Nejméně důležitou chybou je nepotvrzení přijetí dat, což značí barva sytě žlutá. Nepotvrzení dat je nejméně významnou chybou z důvodu, že se pomocí něho dá také komunikace mezi zařízeními ukončit.

3.4.3 Data

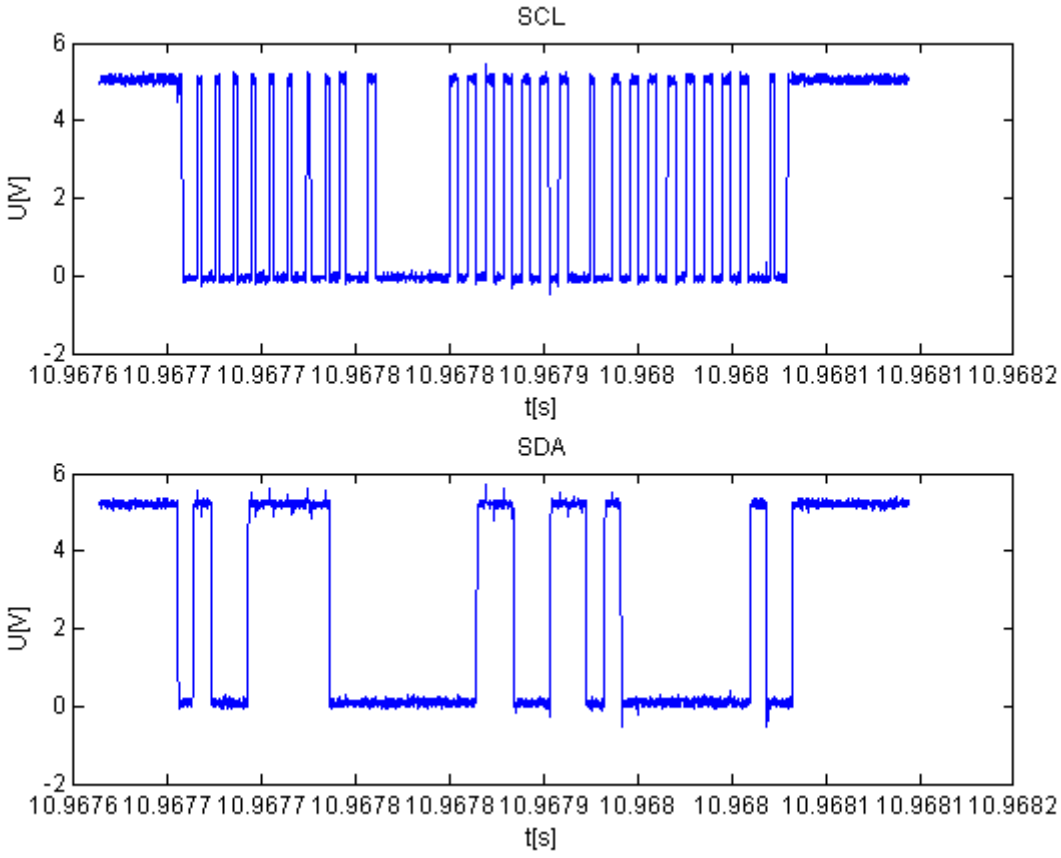
Za hlavičkami následuje informace o přítomnosti šumu v měřeném signálu včetně jako maximální absolutní hodnoty. Tento údaj je pro uživatele pouze informativní, protože aplikace umí zpracovat i zašuměný signál, ale některá zařízení na sběrnici s ním mohou mít problémy.

Jako poslední následuje tabulka s dekódovanými daty. V rozložení tabulky jsem se inspiroval řešením právě od firmy Agilent, které se mi zdálo velice jednoduché a přehledné. Proto tedy na každém řádku je informace o pořadovém číslu paketu, za kterým následuje informace o čase začátku paketu. Čas je počítaný velice přesně na jednu miliontinu sekundy od začátku měření. Ve třetím sloupci se nachází informace, zda se jedná o sedmi nebo desetibitovou adresu, která je hexadecimálně zapsaná v následujícím sloupci. V předposledním sloupci se tak nachází směr komunikace, zda se za zařízení typu slave čte, nebo se do něho zapisuje. Poslední sloupec, respektive skupina sloupců obsahuje samotná data odesílaná v každém paketu.

Pro snadnější orientaci v tabulce je každých 100 řádků znovu zapsaná hlavička tabulky, která je vždy podbarvená tmavě šedou, tak aby bylo na první pohled jasné, že se jedná o hlavičku tabulky a ne data. Liché a sudé řádky tabulky jsou každé podbarvené jinak světlou šedou barvou tak, aby oba tyto odstíny byli jednoznačně oddělené od tmavě šedé, kterou má hlavička. Poslední zvýraznění v tabulce je provedeno ztmavením odstínu šedé daného řádku ve sloupci s daty tak, aby i při nezobrazení hlavičky tabulky na obrazovce bylo jednoznačně jasné, že data jsou v tomto sloupci.

Ukázka jak vypadá vstupní průběh signálu, viz Obrázek 7: Příklad naměřeného vstupního signálu a výstupní soubor, viz Obrázek 8: Ukázka výstupního souboru. Další chybové stavy viz Obrázek 9: Ukázka chybového stavu – napětí uvnitř zakázaného pásma (pomalý

přechod mezi logickými úrovněmi), Obrázek 10: Ukázka chybového stavu – napětí vně definovaného pásma (překročené maximální hodnoty napětí)a Obrázek 11: Ukázka chybového stavu (Nepřipojené zařízení).



Obrázek 7: Příklad naměřeného vstupního signálu

Na průběhu signálů SDA a SLC je vidět, že oba signály obsahují drobné rušení. Zároveň je vidět v některých okamžicích, například u signálu SDA od času 10,9677 s je několik napěťových špiček, nebo okolo času 10,968 s je výraznější špička směrem do záporných hodnot. Stejné špičky se dají pozorovat i u signálu SCL v čase mezi 10,9678 s a 10,968 s. Maximální odchylku šumu, respektive napěťových špiček v celých naměřených datech aplikace změří a vypíše, viz Obrázek 8: Ukázka výstupního souboru.

Data přenášená na sběrnici I2C

Osciloskop	DSO9254A:MY53130116
Zdroj	Binární soubor (2GB)
Datum	21 OCT 2013
Čas	15:15:16
Rychlost I²C	100kHz
Počet paketů	1001

Legenda

Zařízení není připojeno	Napětí uvnitř zakázaného pásma (Pomalý přechod 0->1 nebo 1->0)	Napětí vně definovaného pásma (Překročené maximální hodnoty napětí)	Nepotvrzená data

Absolutní hodnota šumu je: 0.309926V

Dekódovaná data

Index	Čas	I2C Paket	Adresa	Směr	Data	
0	0.108034s	7-Bit Address	0x68	R	0xE3	
1	0.112632s	7-Bit Address	0x68	W	0xFF	0xBF
2	0.113057s	7-Bit Address	0x68	R	0xC2	
3	0.117781s	7-Bit Address	0x68	W	0x00	0x33
4	0.118204s	7-Bit Address	0x68	R	0xA3	

Obrázek 8: Ukázka výstupního souboru

236	6.342813s	7-Bit Address	0x68	R	0xA3	
237	6.347539s	7-Bit Address	0x68	W	0xFF	0xC1
238	6.347964s	7-Bit Address	0x68	R	0x80	
239	6.352688s	7-Bit Address	0x68	W	0x05	0xC3
240	6.547284s	7-Bit Address	0x68	R	0xE3	

Obrázek 9: Ukázka chybového stavu – napětí uvnitř zakázaného pásma (pomalý přechod mezi logickými úrovněmi)

776	20.928547s	7-Bit Address	0x68	R	0xE3	
777	20.933146s	7-Bit Address	0x68	W	0xFF	0xBF
778	20.933571s	7-Bit Address	0x68	R	0xC2	
779	20.938295s	7-Bit Address	0x68	W	0x00	0x33
780	20.938719s	7-Bit Address	0x68	R	0xA3	

Obrázek 10: Ukázka chybového stavu – napětí vně definovaného pásma (překročené maximální hodnoty napětí)

137	3.761541s	7-Bit Address	0x68	W	0xFF	0xBF
138	3.761968s	7-Bit Address	0x68	R	0xC2	
139	3.766690s	7-Bit Address	0x68	W		
140	3.767112s	7-Bit Address	0x68	R	0xA3	
141	3.771839s	7-Bit Address	0x68	W	0xFF	0xC1

Obrázek 11: Ukázka chybového stavu (Nepřipojené zařízení)

4 Dosažené výsledky

4.1 Test rychlosti aplikace

4.1.1 Konfigurace

Pro změření rychlosti zpracování naměřených dat jsem si vybral 2 GB naměřený soubor a čtyři počítače.

Konfigurace testovacích počítačů viz Tabulka 4: Konfigurace testovacích PC

Tabulka 4: Konfigurace testovacích PC

	PC 1	PC 2	PC 3	PC 4
Procesor	Core i5-3230M	Core i7-3517U	C2D E6600	Core i5-3210M
Operační paměť	8 GB 1600 MHz	10 GB 1600 MHz	2 GB 800 MHz	4 GB 1600 MHz
Pevný disk	HGST HTS725050A7E630 (magnetický, 2,5“, 7200 RPM)	SSD 512 GB	ST3250620AS (magnetický, 3,5“, 7200 RPM)	TOSHIBA MQ01ABD075 (magnetický, 2,5“, 5400 RPM)
Operační systém	Windows 7 Professional x64	Windows 8.1 Professional x64	Windows 7 Professional x32	Windows 8.1 Professional x64

4.1.2 Podmínky testu

Na všech počítačích byla spuštěna stejná verze aplikace, pro 64bitový operační systém zkompilovaná 64bitová aplikace a pro 32bitový systém 32bitová aplikace. U všech počítačů se dekódovala stejná naměřená data osciloskopem Agilent DSO9254A uložená v jednom 2 GB binárním souboru. Aplikace během svého chodu měří čas jednotlivých kroků, které provádí. Dílčí a celkové výsledky viz Tabulka 5: Výsledky měření rychlosti aplikace

Tabulka 5: Výsledky měření rychlosti aplikace

	PC 1	PC 2	PC 3	PC 4
Otevření a načtení souboru	24,756 s	2,047 s	45,371 s	27,894 s
Detekování rychlosti	0,003 s	0,000 s	0,002 s	0,016 s
Dekódování dat	0,893 s	0,390 s	0,467 s	0,953 s
Uložení výsledků	0,009 s	0,000 s	0,040 s	0,062 s
Celkový čas	**25,661 s**	**2,437 s**	**45,880 s**	**28,925 s**

4.1.3 Vyhodnocení testu

Z testu je jasně patrné, že nejdéle na celém zpracování vstupních dat trvá načtení souboru z pevného disku. Samotné dekódování a uložení výsledků je oproti načítání téměř zanedbatelné. Nejlépe je tedy na tom notebook s SSD diskem a nejhůře počítač, který je cca 7 let starý.

4.2 Systémové prostředky

Během testování se pomocí správce úloh systému potvrdilo, že aplikace je nenáročná na systémové prostředky, nejenom že rychle zpracuje veškerá data, ale také nepotřebuje mnoho operační paměti ke svému chodu oproti původní velikosti naměřených dat.

Aplikace pro svůj chod, kdy má v operační paměti uložený celý vstupní 2 GB soubor, veškeré struktury s hlavičkami a struktury s dekódovanými daty nepotřebuje více jak 514 MB paměti.

Jediná doba, kdy je potřeba více operační paměti je během načítání dat ze souboru, kde špičková potřeba paměti dosahuje až 580 MB.

Typická spotřeba operační paměti tak dosahuje 25 % velikosti původního souboru. U maximální potřeby paměti je to asi 28 %.

4.3 Doporučená nastavení a maximální doba vzorkování osciloskopu

Pro správné vzorkování sběrnice I2C je nutné nastavit vzorkovací frekvenci osciloskopu tak, aby nebyl porušen Shannon-Kotělnikovův teorém. Tudíž aby vzorkovací frekvence byla alespoň dvojnásobná oproti vzorkované frekvenci.

Ideální pro zpracování měřeného signálu je, aby vzorkovací frekvence byla alespoň čtyřnásobnou oproti vzorkované, protože se tím předejde možným problémům se zpracováním, kdy by teoreticky osciloskop mohl zrovna vzorkovat v době přechodu mezi logickými úrovněmi a aplikace by to mohla špatně vyhodnotit. Vyšší vzorkovací frekvencí se tyto vzorky schovají mezi ostatní relevantní vzorky. Minimální a doporučená vzorkovací frekvence viz Tabulka 6: Nastavení vzorkovací frekvence.

Tabulka 6: Nastavení vzorkovací frekvence

Rychlost sběrnice	Minimální vzorkovací frekvence	Doporučená vzorkovací frekvence
100 kHz	200 kHz	400 kHz
400 kHz	800 kHz	2 MHz
1 MHz	2 MHz	4 MHz
3,4 MHz	6,8 MHz	20 MHz

Při vzorkování osciloskopem Agilent DSO9254A, který má mnohem větší paměť než běžné osciloskopy, umožňuje tak vzorkovat a uložit průběh naměřené sběrnice při nejnižší možné vzorkovací frekvenci 200 kHz pro 100 kHz rychlost sběrnice téměř až jeden a půl hodiny. Přesné hodnoty viz Tabulka 7: Doba vzorkování osciloskopu Agilent DSO9254A.

Tabulka 7: Doba vzorkování osciloskopu Agilent DSO9254A

Vzorkovací frekvence	Maximální doba vzorkování
200 kHz	1 h 23 min 20 s
400 kHz	41 min 40 s
1 MHz	16 min 40 s
2 MHz	8 min 20 s
4 MHz	4 min 10 s
20 MHz	50 s
50 MHz	20 s

5 Závěr

Cílem této bakalářské práce bylo seznámení se se standardy sběrnice I2C a parametry osciloskopu Agilent řady 9000 s ohledem na jeho využití při analýze komunikace na sběrnici I2C. Dalším cílem byl samotný návrh a realizace programu, který provede samotnou analýzu naměřených dat z osciloskopu a vytvoří výpis komunikace do HTML souboru. V neposlední řadě výsledný program měl obsahovat analýzu a upozornění na základní chybové stavy.

Nejprve tedy bylo provedeno samotné nastudování standardu sběrnice I2C s ohledem na realizaci samotné bakalářské práce. Dále před samotnou realizací proběhlo nastudování parametrů osciloskopu Agilent DSO9254A z uživatelské dokumentace k tomuto osciloskopu. Následovalo nastudování příslušných částí programátorské dokumentace.

Samotný návrh a realizace výsledné aplikace se povedlo dokončit do stavu, kdy aplikace splňuje zadání bakalářské práce včetně dalších požadavky definované v úvodu. Povedlo se tedy vytvořit aplikaci, která je univerzální pro další rozšíření, umožňuje zpracovat velké množství dat ve velice krátkém čase, kdy rychlost celého procesu zpracování je závislá především na rychlosti pevného disku. Díky tomu bohužel nejde zpracování již více významně zrychlit. Po načtení je již samotné zpracování dat řádově v desetinách sekundy maximálně v jednotkách sekund u pomalejšího počítače. Výstup celé aplikace je nakonec nejenom v jednom detailním výpisu ve formátu HTML, ale je přidán také druhý stejně detailní výpis v HTML souboru pouze s chybovými pakety, čímž se zvýšila přehlednost.

V porovnání s komerčně dostupnými řešeními od výrobců osciloskopů má tato aplikace větší funkcionalitu s ohledem na detekci základních chybových stavů. Oproti komerčním řešením nabízí detekci absolutní hodnoty velikosti šumu signálu a zároveň je schopna signál včetně toho šumu zpracovat. Dále detekuje pomalé přechody mezi logickými úrovněmi jedna a nula, čímž splňuje také zadání této práce detekce signálu uvnitř zvoleného napěťového pásma. Je možné detekovat významné překmity nebo podkmity napětí na sběrnici přesně identifikovat místa chyb. V neposlední řadě oproti komerčním řešením umožňuje detekci, zda zařízení s adresou, kterou volá master je připojeno ke sběrnici či nikoliv. Poslední detekcí chyb, kterou mají všechna komerční řešení včetně aplikace vytvořené v této

bakalářské práci je detekování potvrzování přijetí dat, kdy je jeden rozdíl že aplikace z této práce potvrzování nevypisuje jako text, ale signalizuje nepotvrzení podbarvením dané buňky tabulky.

Jediné čím výsledná aplikace, oproti jiným komerčním řešením, nedisponuje je analyzování aktuálně měřených dat osciloskopem, což může být jako jedno z rozšíření této práce v rámci magisterského projektu nebo diplomové práce. Dalším rozšířením by mohla být podpora dalších osciloskopů a dalších sběrnic, respektive začlenění do nějakého konkrétního programu nebo grafického uživatelského prostředí. Nejdůležitější úkol rychlé analýzy a dekódování dat na sběrnici I2C byl ale touto prací splněn.

Seznam použité literatury

Agilent Technologies, Inc. 2007. Binary Oscilloscope File to MATLAB Translator Program | Agilent. *Web Agilent Technologies.* [Online] 10. 5 2007. [Citace: 3. 11 2013.] http://www.home.agilent.com/agilent/editorial.jspx?cc=CZ&lc=eng&ckey=1185953&nid=-11143.0.00&id=1185953.

—. 2013. I2C and SPI Protocol Triggering and Decode - Data sheet. *Web Agilent Technologies.* [Online] 11. 1 2013. [Citace: 13. 4 2014.] http://cp.literature.agilent.com/litweb/pdf/5990-3925EN.pdf.

—. 2013. Infiniium 9000 Series Oscilloscopes - Data Sheet. *Web Agilent Technologies.* [Online] 6. 11 2013. [Citace: 13. 4 2014.] http://cp.literature.agilent.com/litweb/pdf/5990-3746EN.pdf.

—. 2014. Programmer's Reference for Infiniium 9000 Series Oscilloscopes. *Web Agilent Technologies.* [Online] 23. 1 2014. [Citace: 13. 4 2014.] http://www.home.agilent.com/upload/cmc_upload/All/9000_series_prog_ref.pdf?&cc=CZ&lc=eng.

Bílek, Petr. 2013. Programování v jazyku C/C++. *Sally.* [Online] 1. 8 2013. [Citace: 13. 4 2014.] http://www.sallyx.org/sally/c/.

2008. Céčko. *Céčko.* [Online] 2008. [Citace: 13. 4 2014.] http://www.jazykc.ic.cz/.

Devantech Ltd (Robot Electronics). I2C Tutorial. *Robot Electronics.* [Online] [Citace: 13. 11 2013.] http://www.robot-electronics.co.uk/acatalog/I2C_Tutorial.html.

Domes, Martin. 2005. *Tvorba internetových stránek pomocí HTML, CSS a JavaScriptu.* Kralice na Hané : Computer Media s.r.o., 2005. ISBN: 80-86686-39-6.

Embedded Systems Academy, Inc. I2C Bus Technical Overview and FAQ - Embedded Systems Academy. *Embedded Systems Academy.* [Online] [Citace: 13. 4 2014.] http://www.esacademy.com/en/library/technical-articles-and-documents/miscellaneous/i2c-bus/.

Ltd), OOO "Program Verification Systems" (Co. Lessons on development of 64-bit C/C++ applications. *Static Code Analyzer for C/C++/C++11 and 64-bit migration.* [Online] [Citace: 13. 4 2014.] http://www.viva64.com/en/l/.

NXP_Semiconductors. 2014. I2C-bus specificatioon and user manual. *NXP Semiconductors.* [Online] 4. 4 2014. [Citace: 13. 4 2014.] http://www.nxp.com/documents/user_manual/UM10204.pdf.

Plíva, Zdeněk, a další. 2014. Metodika zpracování bakalářských a diplomových prací. *Fakulta mechatroniky, informatiky a mezioborových studií, TUL.* [Online] 3 2014. [Citace: 13. 4 2014.] http://www.fm.tul.cz/cs/jak_psat_prace. ISBN 978-80-7494-049-1.

Prata, Stephen. 2005. *C++ primer plus 5th.* Indianapolis : Sams, 2005. ISBN 0-672-32697-3.

Tišnovský, Pavel. 2009. Komunikace po sériové sběrnici I2C - Root.cz. *ROOT.CZ.* [Online] 8. 1 2009. [Citace: 13. 4 2014.] http://www.root.cz/clanky/komunikace-po-seriove-sbernici-isup2supc/.

Virius, Miroslav. 2005. *Pasti a propasti jazyka C++. 2. aktualizované a rozšířené vydání.* Brno : CP Books, 2005. ISBN 80-251-0509-1.

Obsah přiloženého CD

Text bakalářské práce

- Bakalarska_prace_2014_Petr_Vosta.pdf
- Bakalarska_prace_2014_Petr_Vosta.doc
- Bakalarska_prace_2014_Petr_Vosta.docx

Zdrojové kódy

Spustitelný program pro platformu x86

Spustitelný program pro platformu x64