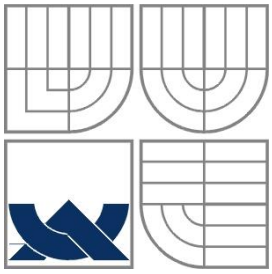
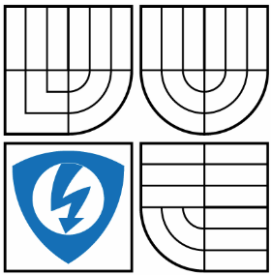


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

PŘIPOJENÍ EXTERNÍCH ZAŘÍZENÍ K MIKROKONTROLÉRU FREESCALE MC9S08LH64 PŘES SBĚRNICE IIC A SPI

CONNECTING EXTERNAL DEVICES WITH FREESCALE MC9S08LH64 MICROCONTROLLER BY IIC AND SPI BUS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

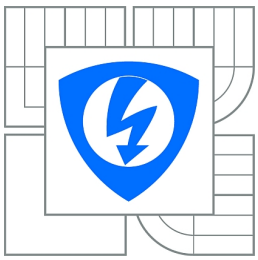
AUTOR PRÁCE — DAVID PAMÁNEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. TOMÁŠ MACHO, Ph.D.

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	David Pamánek	***ID:***	146921
Ročník:	3	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Připojení externích zařízení k mikrokontroléru Freescale MC9S08LH64 přes sběrnice IIC a SPI

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s interface IIC a SPI mikrokontroléru Freescale MC9S08LH64 a vývojovou deskou TWR-S08LH64.
2. Navrhněte připojení externích zařízení k vývojové desce TWR-S08LH64 prostřednictvím sběrnic IIC a SPI. Uvažujte připojení např. kalendářového obvodu, externí paměti, A/D, D/A převodníků a posuvných registrů. Nakreslete schémata zapojení.
3. Realizujte přípravky se 2 externími zařízeními komunikujícími přes sběrnici IIC a 2 externími zařízeními komunikujícími přes sběrnici IIC.
4. Vytvořte potřebné softwarové vybavení pro mikrokontrolér ve formě knihovních funkcí umožňující využívat jednotlivá externí zařízení.
5. Software odlaďte a jednotlivé funkce zdokumentujte.

DOPORUČENÁ LITERATURA:

[1] MC9S08LH64 Reference Manual. Rev. 5.1. Freescale Semiconductor. 2012.
[2] TWR-S08 User gude. Axiom manufacturing. 2010.

Termín zadání:	10.2.2014	***Termín odevzdání:***	26.5.2014

Vedoucí práce: Ing. Tomáš Macho, Ph.D.
Konzultanti bakalářské práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

Abstrakt

Obsahem této práce je seznámení se sériovými sběrnicemi SPI a IIC, jejich vlastnosti a popis principu jejich funkce. Dále je zde seznámení s mikrokontrolérem Freescale MC9S08LH, nejprve obecně s celým zařízením, a poté jsou detailně rozebrány moduly pro sériovou komunikaci SPI a IIC. Také jsou v práci popsány čtyři vybrané externí zařízení a jsou navrženy čtyři přípravky, každý s jedním z těchto obvodů. Ke každému přípravku je vytvořeno schéma zapojení, deska plošných spojů. Nakonec jsou zde popsány knihovní funkce pro komunikaci mezi vybranými obvody a mikrokontrolérem.

Klíčová slova

Sběrnice, sériová, SPI, Serial Peripheral Interface, IIC, I2C, Inter-Integrated Circuit, mikrokontrolér, Freescale, MC9S08LH64, procesor, HCS08.

Abstract

The content of this thesis is to introduce the SPI and IIC serial buses, their properties and description of their function. After this part there is the description of the microcontroller Freescale MC9S08LH. It gives the basic description of whole device and more detailed description of SPI and IIC modules. In the next part of the document there is described four external devices, which I have chosen. For each device there is shown connetion diagram and printed circuit board. In the end of this document is description of libraries created for communication between chosen devices and microcontroller.

Keywords

Bus, serial, SPI, Serial Peripheral Interface, IIC, I2C, Inter-Integrated Circuit, microcontroller, Freescale, MC9S08LH64, processor, HCS08.

Bibliografická citace

PAMÁNEK, D. *Připojení externích zařízení k mikrokontroléru Freescale MC9S08LH64 přes sběrnice IIC a SPI*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014. 51 s. Vedoucí bakalářské práce Ing. Tomáš Macho, Ph.D..

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma „Připojení externích zařízení k mikrokontroléru Freescale MC9S08LH64 přes sběrnice IIC a SPI“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení §11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně ……. ……...

(podpis autora)

Poděkování

Rád bych poděkoval vedoucímu práce panu Ing. Tomáši Machovi, Ph.D. za odborné vedení, konzultace a ponechanou volnost při vypracování.

Obsah

1 ÚVOD

S rozmachem digitální techniky začala vznikat potřeba přenosu dat mezi jednoduchými digitálními zařízeními. Kvůli tomu začaly vznikat standardy komunikačních sběrnic a protokoly předepisující průběh komunikace.

V dnešní době existuje celá řada takových sběrnic a protokolů, které se dělí podle různých parametrů. Například podle způsobu přenášení dat na sériové a paralelní, kdy u paralelních sběrnic dochází k přenosu celého bloku dat najednou, je tedy potřeba většího množství vodičů. U sériových sběrnic jsou data posílána za sebou po jednotlivých bitech buď pomocí jednoho vodiče poloduplexně tedy tak, že obě zařízení dokážu posílat i přijímat data, nikoli však současně. Druhou skupinou jsou sériové protokoly, které komunikují po dvou vodičích plně duplexně, tudíž mohou vysílat i přijímat data ve stejný časový okamžik. Mezi další parametry patří například topologie zapojení, počet jiných než datových vodičů, způsob adresování atd.

Vybral jsem si téma, které se zabývá komunikací přes sériové sběrnice SPI a IIC mezi mikrokontrolérem *Freescale MC9S08LH64* a několika periferními zařízeními. Možnost propojování zařízení je velmi užitečná, a proto jsem se při výběru semestrální práce rozhodl pro toto téma, abych mohl prohloubit své znalosti v oblasti této problematiky.

Řešení tohoto úkolu hodlám rozdělit do několika částí. Nejdříve je zapotřebí seznámit se s tím, jak jednotlivé sběrnice fungují, jaké jsou jejich vlastnosti a v čem spočívají výhody jejich využití. Dále si musím nastudovat vlastnosti zadaného mikrokontroléru, jeho jednotlivé moduly, zejména ty, které zajišťují komunikaci dle uvedených protokolů. Dalším úkolem je výběr čtyř zařízení, která budou použita pro připojení k mikrokontroléru a pro komunikaci s ním. Dvě zařízení s podporou sběrnice SPI, které jsem si vybral, jsou AD převodník *TLC549* od firmy *Texas Instruments* a digitálně řízený potenciometr *AD5160* od firmy *Analog Devices*. Pro ukázku komunikace po sběrnici IIC jsem si vybral obvod reálného času *DS1307* od firmy *Maxim* a DA převodník *TC1320* od firmy *Microchip*. U všech zmíněných obvodů je nutné nastudovat jejich vlastnosti, princip jejich funkce a způsob jakým komunikují po dané sběrnici. Dále je potřeba vytvořit čtyři přípravky s uvedenými obvody, navrhnout schémata jejich zapojení a desky plošných spojů jednotlivých přípravků. Posledním úkolem bude vytvoření knihovních funkcí, které budou zajišťovat komunikaci mezi mikrokontrolérem a vybranými externími obvody. Na závěr je ještě potřeba vytvořit ke každému přípravku program, který bude tyto knihovny využívat, odladit jej a otestovat funkčnost knihovních funkcí.

2 SBĚRNICE

Pod pojmem sběrnice se rozumí soubor vodičů určených k přenosu dat mezi zařízeními. Každý typ sběrnice definuje způsob, jakým je komunikace realizována a kvůli těmto vlastnostem se dělí do několika skupin. Jak již bylo zmíněno, hlavní rozdělení sběrnic je na paralelní a sériové.

Základní vlastností paralelních sběrnic je jejich šířka, která udává počet bitů přenesených najednou. Je tedy zřejmé, že k realizaci paralelní sběrnice je potřeba stejného počtu vodičů, jako je její šířka, což je hlavní nevýhoda sběrnic tohoto typu. Dalším problémem u paralelního přenosu dat je fakt, že není možné zajistit, aby se data přenesla po všech vodičích za stejnou dobu, díky čemuž je nutné čekat určitou dobu, než může dorazit ten nejpomalejší.

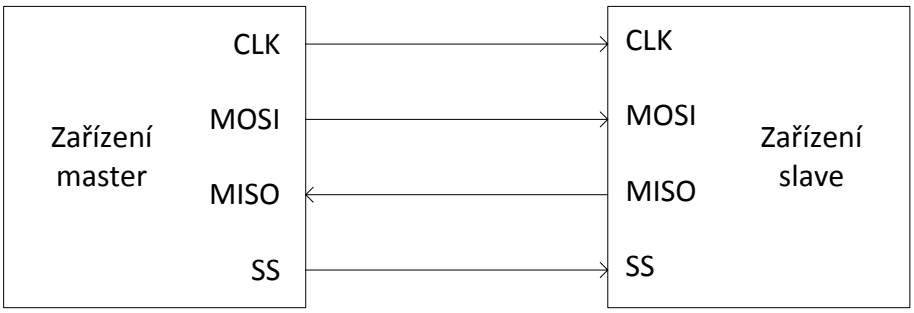
Sériové sběrnice pracují tak, že jsou jednotlivé bity posílány za sebou. Přináší tedy výhodu v tom, že je značně zredukován počet vodičů. Oproti paralelním je tedy zapotřebí složitějšího přenosového protokolu. Mezi sériové sběrnice patří například SPI a IIC, které jsou hlavní náplní této práce.

Další rozdělení sběrnic je podle časování datového toku na synchronní a asynchronní. Synchronní sběrnice se vyznačují přítomností vodiče pro hodinový signál, který generuje nadřazené zařízení. Asynchronní sběrnice tento vodič nemají, časování je řízeno jiným způsobem, například pomocí *acknowledge* bitů.

2.1 SPI (Serial Peripheral Interface)

SPI je sériová sběrnice určená ke komunikaci mezi dvěma a více zařízeními, kde je právě jedno zařízení pracuje jako řadič sběrnice (master) a ostatní jako slave zařízení. Sběrnice SPI se vyznačuje synchronním přenosem dat, který je zajištěn tím, že zařízení typu master generuje hodinový signál, který je rozveden do ostatních zařízení. Dalším základním parametrem je obousměrný přenos dat (full duplex). Jsou tedy potřeba dva vodiče označované jako MISO (master in slave out) a MOSI (master out slave in), díky čemuž není potřeba přepínání jednotlivých portů ze vstupních na výstupní a naopak. Hlavní výhodou je tedy značná jednoduchost implementace této sběrnice.

Jelikož v jeden okamžik je možné komunikovat pouze s jedním zařízením typu slave, je pořeba označit jedno zařízení pro aktuální přenos. Způsob výběru také přispívá k jednoduchosti této sběrnice, jelikož nepoužívá žádné adresování, ale využívá vodičů označovaných jako SS (slave select), kde ke každému slave zařízení je přiveden právě jeden takový vodič. Z toho plyne hlavní nevýhoda této sběrnice a to, že je potřeba větší množství vodičů a volných výstupních portů na řadiči sběrnice.

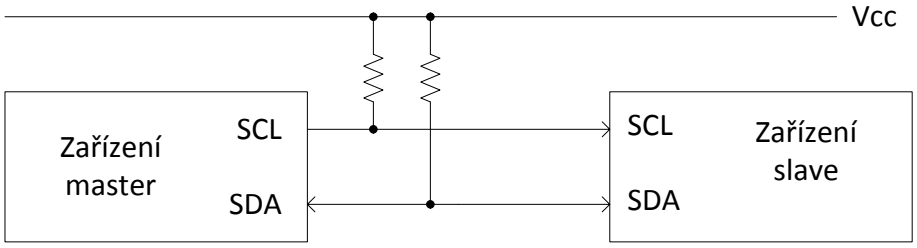


Obrázek 2-1: Propojení dvou zařízení sběrnicí SPI

Sběrnice SPI se využívá ke komunikaci s různými zařízeními jako například AD/DA převodníky, externími paměťmi (flash, EPROM), hodinami reálného času, LCD panely nebo obyčejnými posuvnými registry.

2.2 IIC (Inter-Integrated Circuit)

IIC je další sériovou sběrnicí, která se vyznačuje komunikací mezi několika zařízeními, z nichž může být na rozdíl od již zmíněné sběrnice SPI více zařízení typu master. Sběrnice IIC disponuje pouze jedním datovým vodičem značeným jako SDA, jedná je tedy o poloduplexní komunikaci. Z toho plyne, že pin SDA musí umožňovat přepínání mezi vstupním a výstupním režimem. Dalšími znaky IIC je označení slave zařízení adresou, která má 7 nebo 10 bitů a přítomnost acknowledge signálu. Vše je přesně definováno komunikačním protokolem. Přenosová rychlost IIC je omezena na 10 kbps, také je omezena délka vodičů jejich maximální kapacitou 400 pF. V klidovém stavu jsou jednotlivé vstupy a výstupy ve stavu logické 1, tudíž musí být oba vodiče sběrnice opatřeny pull-up rezistory, jak je vidět na obrázku 2-2. Velikost odporů je určena dobou trvání náběžné hrany signálů a celkovou kapacitou sběrnice.



Obrázek 2-2: Propojení dvou zařízení sběrnicí IIC

Sběrnice IIC se stejně jako SPI využívá ke komunikaci se zařízeními jako například AD/DA převodníky, externími paměťmi, hodinami reálného času, dále se využívá pro řízení displejů atd.

3 POPIS MIKROKONTROLÉRU MC9S08LH64

MC9S08LH64 je mikrokontrolér od firmy Freescale s osmibitovým procesorem HCS08. Hlavními přednostmi je velmi dobrý poměr ceny a výkonu, také vyniká velmi nízkou spotřebou energie, která je dána několika úrovněmi úsporných režimů a režimů spánku. V této kapitole se budu věnovat popisu procesoru a ostatních periferií, které mikrokontrolér obsahuje, zejména modulům SPI a IIC.

3.1 Základní parametry

- 64 KB FLASH paměť
- 4 KB RAM
- Komparátor
- Osmikanálový AD převodník
- Komunikační moduly IIC, SCI a SPI
- Osm pinů podporujících externí přerušení
- LCD displej
- Generátor PWM

3.2 Provozní režimy

Jak již bylo zmíněno, zařízení může, kromě standardního režimu (run), pracovat v několika režimech, které snižují spotřebu energie. Prvním z nich je režim Low-Power Run, ve kterém sice procesor normálně vykonává instrukce, ale vše probíhá s nižší frekvencí a hodinový signál je odpojen od nevyužívaných periferií. Přerušení nebo reset přepne zařízení zpět do běžného run režimu. Dalším je wait režim, ve kterém dojde k vypnutí hodinového signálu a zařízení čeká na přerušení, které jej opět přepne do run módu. Posledními režimy jsou dva stop módy (stop2 a stop3), ve kterých dojde k pozastavení hodinového signálu a k vypnutí většiny periferií. Po probuzení ze stop3 módu je zachován obsah registrů a paměti RAM. U stop2 režimu je jejich obsah vynulován.

3.3 Hodinový signál a jeho distribuce

Generování hodinových pulzů zajišťuje modul ICS (Internal Clock Source), na jehož výstupu může být signál s maximální frekvencí 40 MHz a je připojen přímo k procesoru. Dále je jeho kmitočet vydělen 1, 2, 4 nebo 8 a rozveden k periferiím. Pro některé periferie jsou zde i další výstupy s upravenou frekvencí právě pro ně. Také je možné připojení externího oscilátoru. Základní kmitočet se dá vydělit 2 až 1024 podle nastavení bitů RDIV a RANGE.

3.4 CPU

Srdcem mikrokontroléru je osmibitový procesor HCS08, který obsahuje 64 KB adresního prostoru, pět registrů a umožňuje sedm způsobů adresování. Prvním registrem je osmibitový akumulátor, který slouží pro uložení výsledků aritmeticko-logických operací. Dalším je indexový registr složený ze dvou osmibitových registrů H a X, které dohromady fungují jako ukazatel na určité místo v paměti. Registr X také může pracovat samostatně pro uchování dat a lze s ním provádět podobné operace jako s akumulátorem. Dalšími registry jsou dva šestnáctibitové adresové registry, kde jeden slouží jako ukazatel na zásobník (Stack Pointer) a druhý jako čítač instrukcí (Program Counter), který ukazuje na instrukci, která má být načtena. Poslední je registr příznaků, který indikuje různé výsledky aritmetických nebo logických operací, jako například nulový nebo záporný výsledek, carry bit po sčítání atd.

3.5 Vstupně/výstupní rozhraní

Mikrokontrolér má pět paralelních V/V portů, které obsahují 39 pinů. Většinu z nich využívají jednotlivé moduly, jako například moduly pro sériovou komunikaci SPI a IIC, které budou popsány později. Dalším zařízením je modul zajišťující externí přerušení od tlačítek. Celkem využívá osm pinů, z nichž může být aktivován každý zvlášť. Hlavní využití těchto pinů je možnost probuzení mikrokontroléru z wait nebo stop režimů. U jednotlivých pinů je možné nastavit, jestli mají reagovat pouze na hranu nebo na hladinu, dále se dá nastavit, jestli jsou využívány pull-up nebo pull-down rezistory. Mezi další moduly využívající V/V porty patří analogový komparátor, který má dva analogové vstupy. Pin ACMP+ je připojený k neinvertujícímu vstupu komparátoru, ACMP- k invertujícímu vstupu. Je možné připojit signál na oba vstupy, poté je komparátor porovnává mezi sebou, nebo je možné připojit pouze invertující vstup, poté je porovnáván s referenční úrovní. Komparátor pracuje i ve wait a stop3 módech, ve stop2 režimu je modul vypnutý. Mikrokontrolér také obsahuje modul s šestnáctibitovým AD převodníkem, LCD displej a generátor PWM pulzů.

3.6 Modul SPI

Součástí mikrokontroléru MC9S08LH64 je modul SPI, který zajišťuje komunikaci podle standardů této sběrnice. V této části dokumentu se budu věnovat podrobnému popisu této periferie a významu jednotlivých registrů.

3.6.1 Popis jednotlivých vstupů, výstupů

Pro práci s tímto modulem si můžeme vybrat jednu ze dvou kombinací pinů pro jednotlivé vstupy a výstupy, které budou využity ke komunikaci přes SPI. Vybíráme

pomocí bitu SPIPS v registru SOPT2. Rozložení jednotlivých pinů je vidět v následující tabulce.

Tabulka 3-1: Přehled vstupů a výstupů SPI modulu

SPIPS	MISO	MOSI	SPSCK	SS
0 (výchozí)	PTA2	PTA2	PTA1	PTA0
1	PTB4	PTB5	PTB6	PTB7

Hodinový signál je přenášen pomocí pinu SPSCK. Pokud je mikrokontrolér v režimu master je využit jako zdroj hodinového signálu pro ostatní zařízení, pokud pracuje jako slave, je použit jako vstup pro tento signál.

Pro datovou komunikaci využíváme dvojici pinů MOSI a MISO. MOSI je v režimu master využíván pro odesílání dat, v režimu slave spro jejich příjem. Pin MISO je naopak v režimu master pin pro vstupní data, v režimu slave pracuje jako výstup.

Poslední pin, který je přímo využíván modulem SPI je SS. Ten v režimu slave slozží jako vstup signál, kterým může být zařízení vybráno pro komunikaci. V klidovém režimu je v logické 1, vybrán je, pokud přijde logická 0. V režimu master se dá nastavit jako klasický V/V pin, který není ovládán SPI modulem, také může fungovat jako vstup, který indikuje konflikt při existenci více zařízení typu master nebo jako výstupní pin pro výběr slave zařízení. Za účelem výběru slave zařízení můžeme využít libovolného V/V pinu mikrokontroléru.

3.6.2 Registry modulu SPI

Modul SPI pracuje s dalšími pěti registry. Tři z nich jsou určeny pro různá nastavení, dalším registr indikuje různé stavy a poslední slouží k uložení dat pro odeslání nebo pro příjem dat.

Prvním registrem je SPIC1 (SPI Control Register 1), který umožňuje zapnutí celého modulu, povolení přerušení a výběr režimu master/slave. Dále je možné nastavit různé modifikace hodinového signálu (přenos při nástupné nebo sestupné hraně, změna polarity), nastavovat využití pinu SS v master režimu a měnit směr posuvného registru, tedy jestli se má posílat od nejvyššího nebo nejnižšího bitu. Další nastavení jsou možná pomocí registru SPIC2 (SPI Control Register 2), který obsahuje například nastavení režimu, kdy komunikace probíhá obousměrně pouze po jednom vodiči[1], nebo povolení hodinového signálu během wait módu. Poslední parametr, který můžeme nastavit, je přenosová rychlost. Jak již bylo zmíněno v kapitole 3.3, je do modulu přiveden

[1] Při komunikaci po jednom vodiči je v master režimu zvolen pro komunikaci pin MOSI, který tedy pracuje jako MOMI (master out master in). Ve slave režimu je zvolen pin MISO pracující jako SOSI (slave out slave in).

hodinový signál s poloviční frekvencí oproti základnímu kmitočtu, se kterým pracuje procesor. Tento signál je dále vydělen dvěma koeficienty, které se nastavují pomocí registru SPIBR (SPI Baud Rate Register). Nejvyšší tři bity udávají první dělící koeficient, zbylé čtyři druhý koeficient. Konkrétní význam jednotlivých bitů ukazuje tabulka 3-2.

Pro indikaci různých stavů, které při komunikaci můžou nastat slouží stavový registr SPIS (SPI Status Register). Hlásí například zaplnění bufferu, což znamená příchod nového bytu, také stav, kdy je buffer prázdný, a je tedy možné do něj zapsat data. Také detekuje chybový stav, kdy jsou nakonfigurovány dvě zařízení jako master.

Posledním registrem je datový registr SPID (SPI Data Register), který pracuje s odesílanými a přijímanými daty. Pokud je buffer indikován jako prázdný, můžeme do SPID zapsat data, která jsou do bufferu následně odeslány a tím začne přenos tohoto bytu. Po dokončení přenosu máme v bufferu právě přijatý byte a přečtením datového registru je obsah bufferu přesunut do SPID.

Tabulka 3-2: Nastavení frekvence hodinového signálu

SPPR2:SPPR1:SPPR0	Dělící koeficient
0:0:0	1
0:0:1	2
0:1:0	3
0:1:1	4
1:0:0	5
1:0:1	6
1:1:0	7
1:1:1	8

SPR3:SPR2:SPR1:SPR0	Dělící koeficient
0:0:0:0	2
0:0:0:1	4
0:0:1:0	8
0:0:1:1	16
0:1:0:0	32
0:1:0:1	64
0:1:1:0	128
0:1:1:1	256
1:0:0:0	512
Další kombinace	Rezervováno

3.7 Modul IIC

Pro komunikaci pomocí sběrnice IIC je mikrokontrolér vybaven modulem, který pro tuto komunikaci slouží. Jeho vlastnostem, nastavení a významu jednotlivých registrů se budu podrobně věnovat v následující části práce.

3.7.1 Chování v jednotlivých režimech

Modul je v provozu i během wait módu, kdy může vyvolat přerušení a mikrokontrolér probudit. Pokud je MCU ve stop3 módu, je modul IIC vypnut, ale obsah registrů je zachován. Ve stop2 módu je také vypnut, ale obsah registrů je vymazán.

3.7.2 Popis vstupů a výstupů

Pomocí bitu IICPS v registru SOPT2 je možné vybrat jednu ze dvou kombinací V/V portů pro SDA a SCL. Jejich rozložení popisuje následující tabulka.

Tabulka 3-3: Přehled vstupů a výstupů IIC modulu

IICPS	SDA	SCL
0 (výchozí)	PTB4	PTB5
1	PTA2	PTA3

SDA je označení pinu pro datový vodič, SCL značí pin pro hodinový signál. Oba umožňují obousměrný přenos. Porty jsou opatřeny pull-up rezistory, tudíž v klidovém stavu nabývají hodnoty logické 1.

3.7.3 Popis registrů modulu IIC

Registry, které modul využívá, jsou velmi podobné těm, které byly popsány v souvislosti s modulem SPI. Opět zde máme dva registry pro různá nastavení modulu, další pro nastavení frekvence přenosu, indikaci stavů a jeden datový registr. Novinkou je registr IICA (IIC Address Register), jehož horních 7 bitů slouží pro uložení adresy, která je přiřazena mikrokontroléru, pokud pracuje ve slave režimu. Pokud je využíváno 10 bitových adres, jedná se o spodních 7 bitů z této adresy.

Stejně jako u SPI je i do tohoto modulu přiveden hodinový signál s poloviční frekvencí, než základní, se kterým pracuje CPU. Kmitočet tohoto hodinového signáli je dále upraven koeficienty MULT a ICR, které jsou obsaženy v registru IICF (IIC Frequency Divider Register). Kombinací těchto dvou koeficientů můžeme získat jednu ze 64 možných frekvencí hodinového signálu SCL. Základní nastavení modulu se provádí nastavením bitů v registru IICC1 (IIC Control Register 1), kterými je možné zapnutí IIC modulu, povolení přerušení, výběr módu master/slave, nastavení režimu pro příjem nebo odesílání dat. Dále je možné zakázat odesílání acknowledge bitu nebo vygenerovat opakovaný start bit, který se využívá pro přepnutí na jiné slave zařízení, popřípadě ke změně směru komunikace se stávajícím zařízením. IICC2 (IIC Control Register 2) obsahuje další dva řídící signály, a to povolení všesměrového vysílání a přepínání mezi 7bitovoým a 10bitovoým adresovacím režimem. V případě schématu s 10bitovou adresou, jsou bity 0 – 2 využity jako horní tři bity slave adresy.

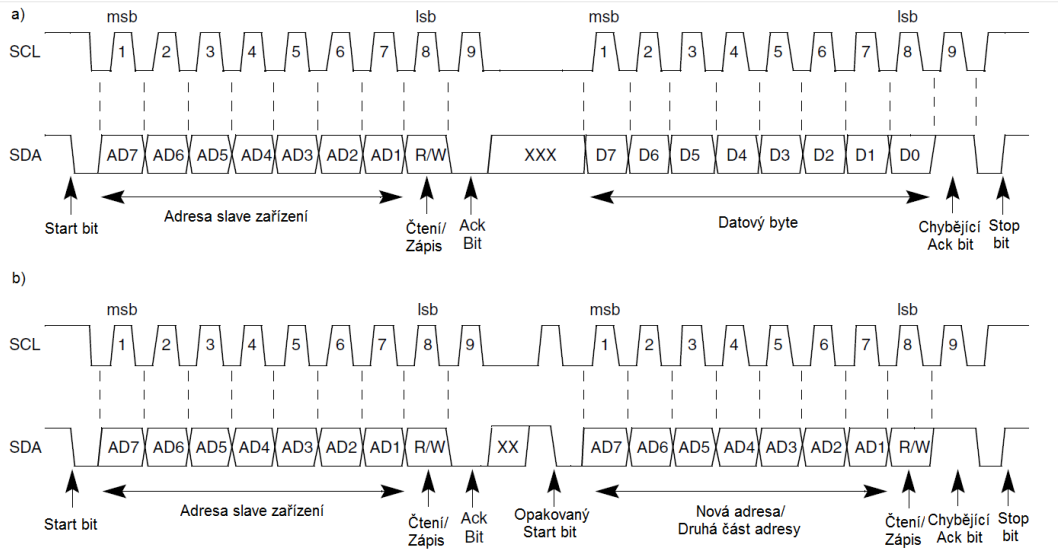
Ke sledování stavů, které mohou při komunikaci nastat, slouží registr IICS (IIC Status Register), který obsahuje bity detekující příchozí byte, stav kdy je mikrokontrolér pracující v režimu slave adresován pro aktuální komunikaci, dále bit indikující probíhající přenos dat (období mezi start a stop bitem), detekci konfliktu dvou master zařízení, které se pokoušejí řídit komunikaci ve stejný okamžik. V tomto případě dojde k přepnutí na slave režim. Další bit ukazuje, jestli mikrokontrolér pracující jako slave má data číst nebo odesílat. Předposledním bitem se indikuje probíhající přerušení a poslední bit detekuje příchozí acknowledge signál.

Posledním resistrem je datový registr IICD. Pokud zařízení pracuje v master režimu, tak zapsáním do IICD (IIC Data I/O Register) začneme odesílání daného bytu (jako první je odesílán nevyšší bit). Pokud data přijímáme, tak je po přenosu můžeme z tohoto registru vyčíst. V režimu slave jsou funkce tohoto registru stejné, musí ovšem být zařízení adresováno master zařízením.

3.7.4 Princip komunikace

Standardní komunikační cyklus má čtyři části, a to start bit, odeslání adresy požadovaného slave zařízení, vlastní přenos dat a stop bit.

Start bit je realizován přechodem signálu SDA z klidového stavu logické 1 na logickou 0, přičemž SCL je stále v logické 1. Poté je odeslán byte, kde horních 7 bitů obsahuje adresu *slave* zařízení a nejnižší bit značí, jestli má dané zařízení přijímat nebo odesílat data. Zařízení s odpovídající adresou odpoví *acknowledge* signálem, který je realizován jako logická 0 při deváté periodě hodinového signálu. Poté následuje datový byte, opět následovaný *acknowledge* signálem. Stop bit je reprezentován přechodem SDA z logické 0 na 1, pokud je SCL na hodnotě logické 1. Průběh jednotlivých signálů je znázorněn na obrázku 3-1 a).



Obrázek 3-1: Časové průběhy přenosu: a) sedmibitové adresování b) desetibitové adresování (Převzato z [3])

Pokud je aktivní 10bitový adresovací režim, je navázání spojení se *slave* zařízením následující. Po start bitu se odešle první část adresy s „předčíslím“ 11110 a je potřeba nastavit *slave* zařízení do režimu čtení, aby si přečetl druhou část adresy. Poté můžeme odesílat data. Pokud ovšem chceme od slave zařízení data přijímat, musíme jej přepnout do stavu vysílání dat a to tím způsobem, že odešleme opakovaný start bit a první část adresy, tentokrát s povelem, že se *slave* zařízení má nastavit do režimu odesílání dat. Postup je znázorněn na obrázku 3-1 b).

Jelikož má každé zařízení svůj vlastní zdroj hodinového signálu, musí dojít k jejich synchronizaci. Ta je realizována tak, že vodiče SCL jsou opatřeny hradly AND, tudíž je výsledný signál SCL logickým součinem všech hodinových signálů na sběrnici.

Modul IIC může vyvolat tři přerušení a to: přerušení po příchozím bytu, při shodě příchozí adresy pokud je mikrokontrolér ve *slave* režimu a v případě konfliktu dvou *master* zařízení, které se snaží komunikovat ve stejný okamžik.[2]

[2] Parafrázováno z [3]

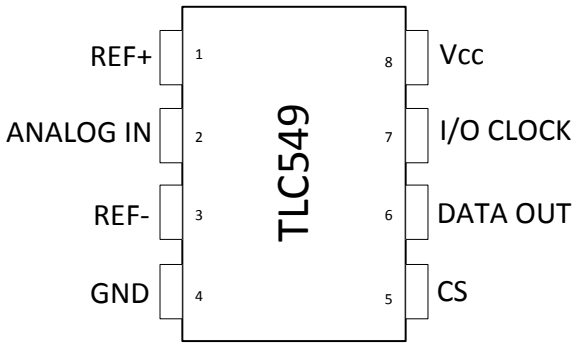
4 EXTERNÍ ZAŘÍZENÍ

Jak již bylo uvedeno v úvodu, vybral jsem si čtyři zařízení, které budou připojeny k mikrokontroléru, se kterým budou komunikovat. Dvě zařízení budou připojeny přes sběrnici SPI a dvě přes IIC. V této kapitole se budu věnovat jejich popisu, zejména toho, jak probíhá komunikace s těmito obvody.

4.1 AD převodník TLC549

TLC 549 je osmibitový AD převodník s postupnou aproximací, který podporuje datovou komunikaci přes sběrnici SPI. Základními parametry obvodu jsou maximální doba převodu 17 μs, maximální počet přístupů za vteřinu 40 000 a maximální frekvence pro komunikaci 1,1 MHz. Napájecí napětí se může pohybovat mezi 3 V a 6 V. Převodník má velmi malou spotřebu a to nejvýše 15 mW. Zařízení obsahuje interní oscilátor s frekvencí 4 MHz.

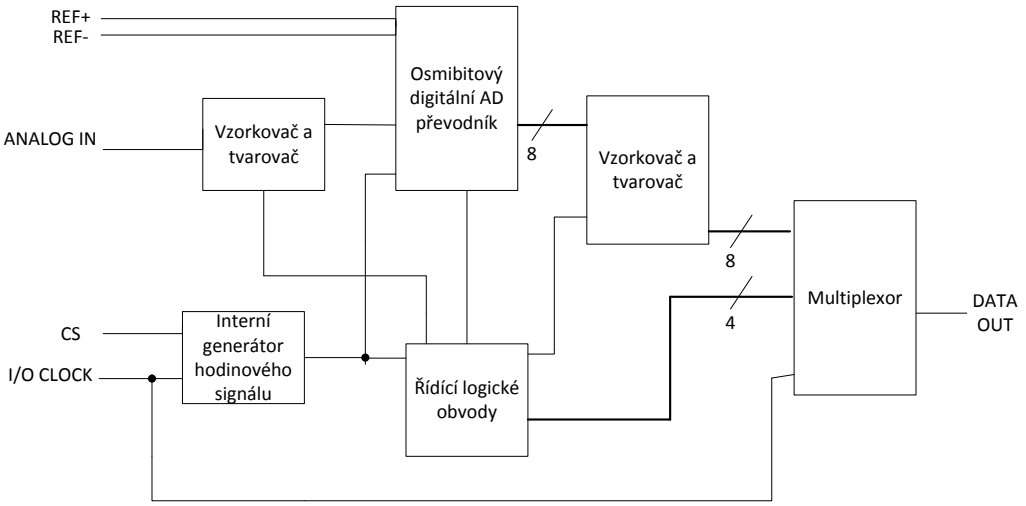
4.1.1 Vstupy a výstupy



Obrázek 4-1: Přehled pinů obvodu TLC549

Na zařízení se nachází pin pro napájení V_{CC}, další pro připojení země *GND*, analogový vstup *ANALOG IN*, dva referenční *REF+* a *REF-* a dva piny pro komunikaci *DATA OUT* jako sériový výstup a *CS* pro výběr zařízení.

4.1.2 Princip činnosti



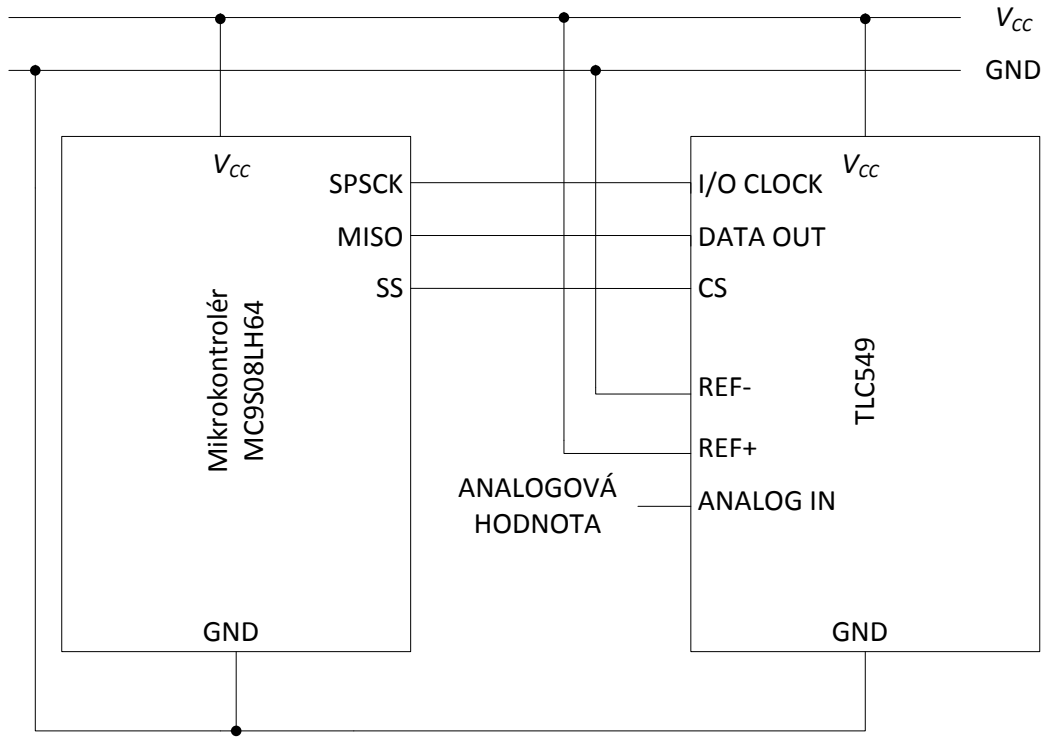
Obrázek 4-2: Blokové schéma obvodu TLC549

Činnost převodníku a interní logiky je řízena interním hodinovým signálem o frekvenci 4 MHz. Převod analogové hodnoty na digitální trvá maximálně 17 µs, kompletní cyklus, tedy převod i s odesláním, může být zopakován každých 25 µs. Interní hodinový signál je zcela nezávislý na hodinovém signálu pro komunikaci a nevyžaduje žádnou synchronizaci. Vzhledem k tomuto faktu je při čtení hodnoty odeslána hodnota z minulého převodu a v tuto chvíli započne další převod. Pokud zařízení není v danou dobu vybráno pro komunikaci, vstup *CS* je ve stavu logické 1, pin *DATA OUT* je v režimu vysoké impedance a hodinový signál pro komunikaci, pin *I/O CLOCK*, je odpojen.

Standardní cyklus čtení dat z převodníku vypadá následovně: Nejprve je potřeba vybrat zařízení a to tak, že vstup *CS* musí být uveden do stavu logické 0. Aby byla zajištěna odolnost vůči šumu, obvod ještě čeká dva vnitřní hodinové cykly, než se uvede do chodu. V tuto chvíli se na výstupu *DATA OUT*, dále jen výstupu, objeví nejvyšší bit. S každou z prvních čtyř sestupných hran je vysunut další bit, tedy druhý, třetí čtvrtý a pátý nejvyšší bit z minulého převodu. V tuto chvíli začne čtení dalšího vzorku, a s dalšími třemi hodinovými cykly dojde k vysunutí posledních tří bitů. Po vysunutí posledního bitu je na čtyři interní hodinové cykly přidržena hodnota ze vstupu. Poté je během dalších 32 interních cyklů převedena analogová hodnota na digitální. Po celou dobu, kdy je přidržena hodnota ze vstupu a následně převedena, tedy 36 interních cyklů, musí být uveden do stavu logické 1, aby mohl být převod správně dokončen. V případě, že chceme číst více hodnot za sebou, je možné ponechat vstup *CS* na hodnotě logické 0,

ale je potřeba dát pozor na vliv rušení. Kdyby došlo během konverze ke změně na tomto vstupu, dojde k resetu zařízení a tím se zruší aktuální převod.[3]

4.1.3 Blokové schéma připojení k mikrokontroléru



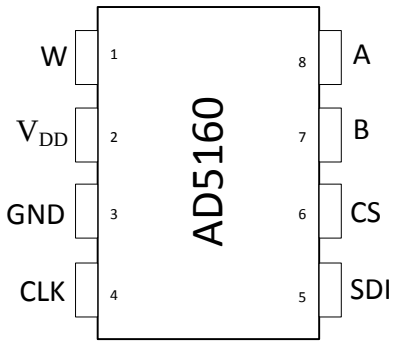
Obrázek 4-3: Propojení mikrokontroléru s obvodem TLC549

4.2 Digitální potenciometr AD5160

Druhým zařízením s podporou SPI, které jsem si pro tuto práci vybral, je digitálně nastavitelný potenciometr AD5160. Zařízení je k dostání ve čtyřech variantách s různými rozsahy a to 5 kΩ, 10 kΩ, 50 kΩ a 100 kΩ. Odpor je na každém rozsahu nastavitelný na jednu z 256 hodnot. Obvod může být napájen v rozsahu od 2,7 V do 5,5 V a má velmi malý odběr 8 μA.

[3] Parafrázováno z [4]

4.2.1 Vstupy a výstupy



Obrázek 4-4: Přehled pinů obvodu AD5160

Napájecí napětí se připojuje k pinu V_{DD}, zem k pinu GND, pro komunikaci slouží piny CLK, CS a SDI, kde CLK je vstup pro hodinový signál, CS vstup pro výběr zařízení a SDI je sériový vstup pro data. Piny A, B a W jsou výstupy, mezi kterými je nastavený odpor. Mezi A a B je maximální odpor (dle zvoleného rozsahu) a mezi W a B popřípadě W a A je nastavená hodnota odporu.

4.2.2 Princip činnosti

Výstupní odpor se nastavuje osmibitovou hodnotou, která se posílá přes rozhraní SPI. Výběr zařízení je opět proveden přivedením logické 0 na vstup CS, poté se s každou náběžnou hranou hodinového signálu nasune do datového registru jeden bit, počínaje tím nejvyšším. Po uvedení pinu CS zpět do klidového stavu je hodnota z datového registru přesunuta do registru RDAC, kde je dále zpracována a převedena na výstupní odpor. Výsledný odpor mezi piny W a B je dán následujícím vztahem.

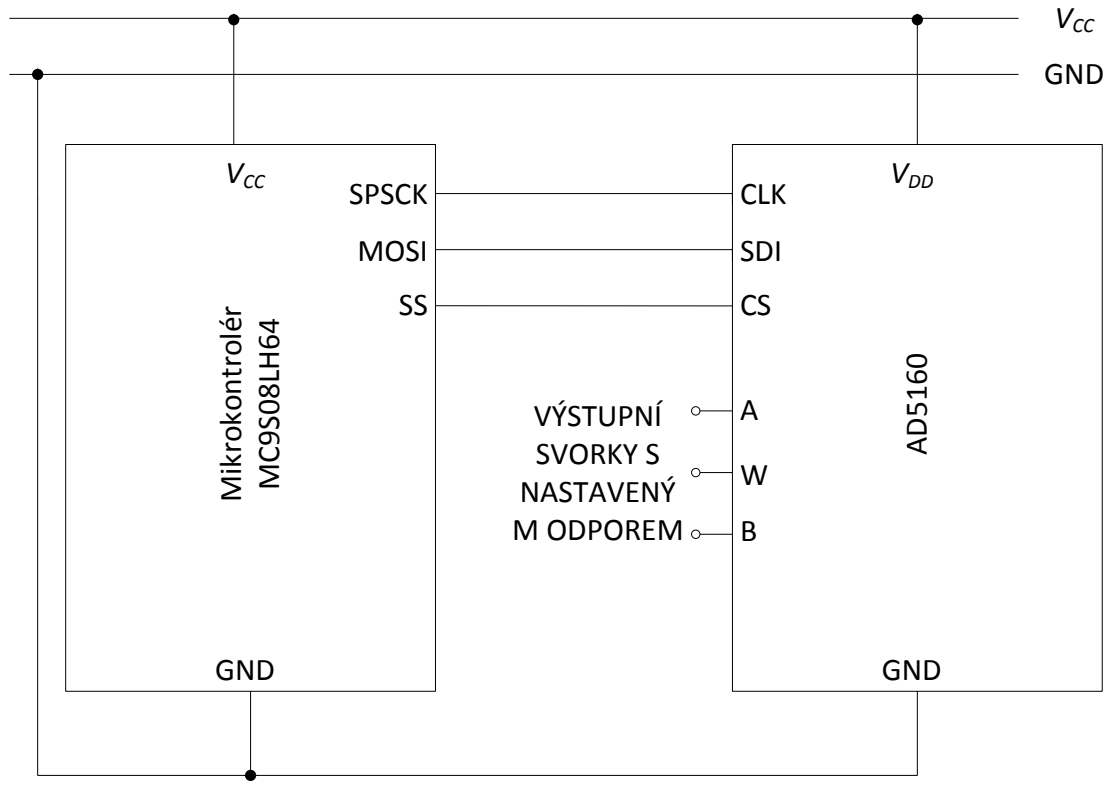
$$R_{WB} = R_{AB}\frac{D}{256} + R_W \qquad 4\text{-}1$$

Kde D je decimální hodnota nastavená v registru, R_{AB} je rozsah dané součástky a odpor R_W je roven 60 Ω. Také je možné využít odporu mezi piny A a W, který se spočítá podle vztahu 4-2.[4]

$$R_{AW} = R_{AB}\frac{256 - D}{256} + R_W \qquad 4\text{-}2$$

[4] Parafrázováno z [2]

4.1.3 Blokové schéma připojení k mikrokontroléru



Obrázek 4-5: Propojení mikrokontroléru s obvodem AD160

4.3 Obvod reálného času DS1307

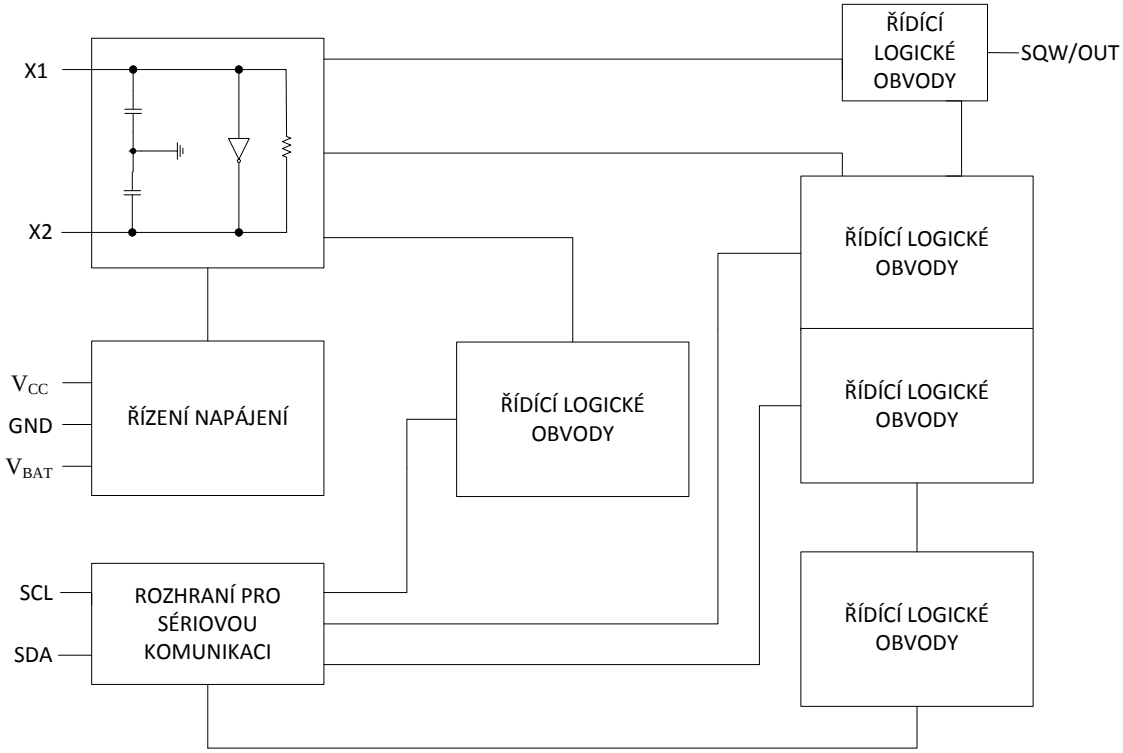
Nyní přejděme ke druhé sběrnici a to IIC. První zařízení, které jsem si pro tuto aplikaci vybral je sériový obvod reálného času DS1307. Obvod poskytuje vteřiny, minuty, hodiny, dny v týdnu, datum, měsíc, a rok. Obvod rozpoznává délku jednotlivých měsíců a detekuje přestupné roky. Údaj o hodině lze nastavit na 24 nebo 12 hodinový formát. Dále je vybaven detekcí poklesu napájení, při kterém přechází na záložní zdroj napájení.

4.3.1 Vlastnosti

- 56 Bytová NV SRAM
- Podpora IIC
- Programovatelný výstup obdélníkového signálu
- Detekce výpadku napájení s automatickým přepnutím na záložní
- Spotřeba méně než 500nA v záložním módu
- Pracovní teplotní rozsah -40°C až 85°C

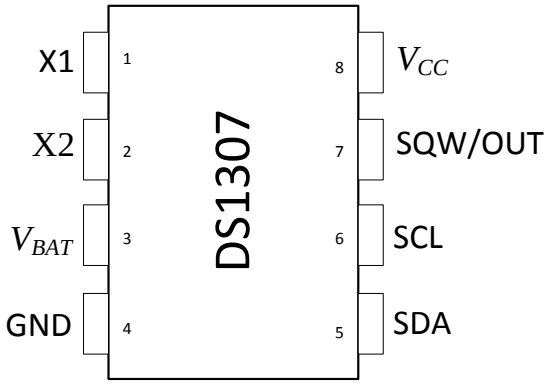
Tabulka 4-1: Doporučená provozní napětí

Parametr	Značka	Minimální hodnota	Typická hodnota	Maximální hodnota
Napájecí napětí [V]	V_{CC}	4.5	5.0	5.5
Logická 1 [V]	V_{IH}	2.2		V_{CC} + 0.3
Logická 0 [V]	V_{IL}	-0.3		0.8
Záložní napájení [V]	V_{BAT}	2.0	3.0	3.5



Obrázek 4-6: Blokové schéma obvodu DS1307

4.3.2 Vstupy a výstupy



Obrázek 4-7: Přehled pinů obvodu DS1307

Piny *X1* a *X2* slouží pro připojení externího oscilátoru s frekvencí 32.768kHz, kapacitou 12.5pF a maximálním sériovým odporem 45kΩ. Požadavky splňuje například krystal *Q 32.768KHZ*.

Pin V_{CC} slouží pro připojení napájecího napětí +5V a V_{BAT} pro připojení záložního napájení, kde bude použita 3V knoflíková baterie. *GND* je společná zem. *SDA* a *SCL* slouží pro připojení k IIC modulu *master* zařízení.

SQW/OUT může pracovat ve dvou režimech, Buď jako klasický digitální výstup, nebo jako zdroj obdélníkového signálu s jednou ze čtyř možných frekvencí (1Hz, 4kHz, 8kHz, 32kHz).

4.3.3 Záložní napájení

Obvod je chráněn proti výpadku hlavního napájení díky 3V baterii, díky které nedojde ke ztrátě dat. Pokud napájecí napětí klesne pod hodnotu 1.25 x V_{BAT} zařízení přestane komunikovat. Pokud klesne pod V_{BAT}, přejde do úsporného režimu. Opětovné probuzení nastane, pokud je napájecí napětí větší než V_{BAT} + 0.2V, ale komunikace je možná, až pokud vzroste nad 1.25 x V_{BAT}.

4.3.4 Struktura vnitřní paměti

Hodnoty reálného času jsou uloženy v registrech, které se nachází na adresách od 00_H až 07_H. Registry paměti RAM mají adresy od 08_H až $3F_H$. Rozmístění jednotlivých časových údajů je zobrazeno v následující tabulce.

Tabulka 4-2: Mapa vnitřní paměti obvodu

Adresa	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	Funkce	Rozsah
00_H	CH	Desítky sekund			Sekundy				Sekundy	00-59
01_H	0	Desítky minut			Minuty				Minuty	00-59
02_H	0	12hod. Formát	Desítky hodin	Desítky hodin	Hodiny				Hodiny	1-12 Nebo 00-23
		24hod. Formát	AM/PM							
03_H	0	0	0	0		Den			Den v týdnu	01-07
04_H	0	0	Desítka data		Datum				Datum	01-31
05_H	0	0	0	Desítka měsíce	Měsíc				Měsíc	01-12
06_H	Desetiletí				Rok				Rok	00-99
07_H	OUT	0	0	SQWE	0	0	RS1	RS0	Ovládání výstupu	-
08_H – $3F_H$									RAM	00_H - FF_H

Pokud je bit *CH* (clock halt) nastaven na logickou 1, je vypnut oscilátor a obvod není aktivní. Pro zapnutí je tedy nutné nastavit tento bit do logické 0.

Dále je možné nastavit výstup *SQW/OUT*. Pomocí bitu *SQWE* přepínáme mezi režimem digitálního výstupu, jehož hodnota je nastavována pinem *OUT*, a výstupem obdélníkového signálu, jehož frekvenci nastavujeme bity *RS1* a *RS2*.

4.3.5 Komunikace přes IIC

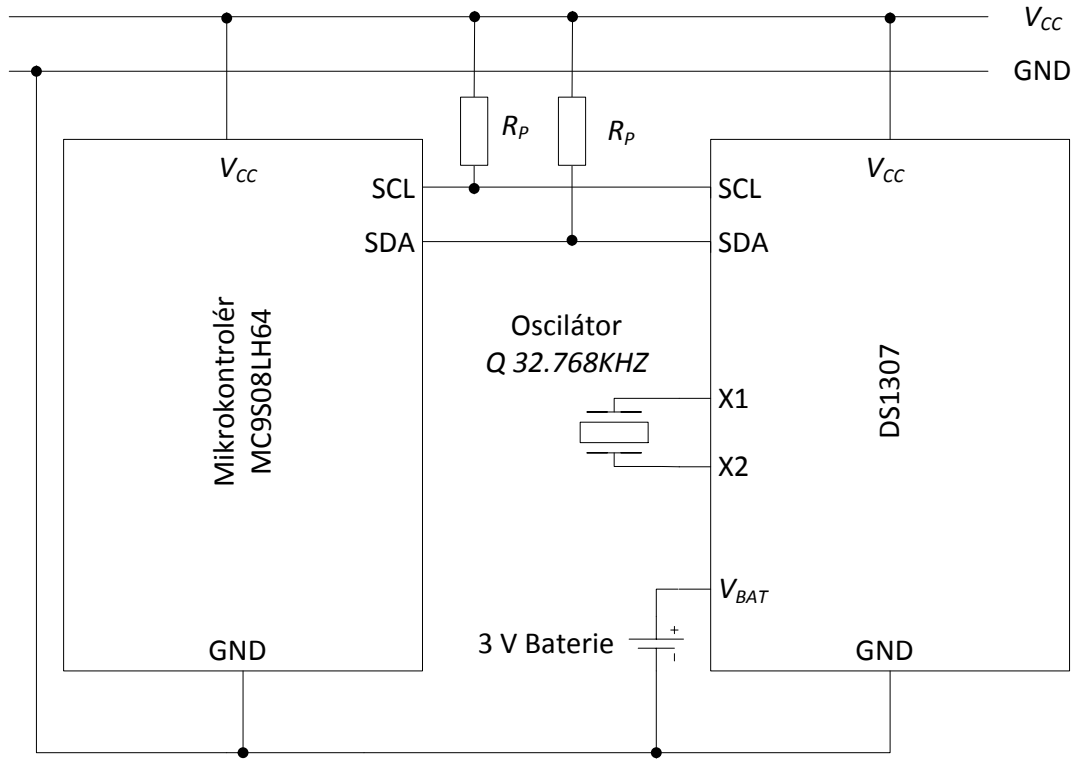
DS1307 plně podporuje komunikaci pomocí protokolu IIC. Zařízení vždy pracuje v režimu *slave* s adresou 1101000.

Po přijetí start bitu očekává první byte s adresou a pokynem jestli má zařízení číst nebo odesílat data. Pokud souhlasí adresa vyslána zařízením typu *master* s adresou zařízení, je odeslán *ach* bit a může začít přenos dat.

Pokud je zvolen režim čtení dat, tedy byl přijat *R/W* bit 0, zařízení očekává jako první byte adresu, která se zapíše do ukazatele *register pointer*, který udává, kam se budou přijatá data zapisovat. Po přijetí dalšího bytu se *register pointer* inkrementuje. Komunikace probíhá dokud *master* neodešle *stop* bit.

Pokud zvolíme režim pro odesílání dat (*R/W* = 1), je po přijetí *ach* bitu zahájeno odesílání dat od aktuálního místa, kam ukazuje *register* pointer. Ten se opět inkrementuje po odeslání jednoho bytu. Pokud si chceme zvolit výchozí hodnotu ukazatele, je potřeba nejprve nastavit zařízení jako příjemce, odeslat mu adresu, která se nastaví do ukazatele. Poté odešleme opakovaný start bit a opět odešleme byte s adresou, ale tentokrát nastavíme zařízení do režimu pro odesílání. Přenos je opět ukončen *stop* bitem vygenerovaným zařízením typu *master*.[5]

4.3.6 Blokové schéma připojení k mikrokontroléru



Obrázek 4-8: Připojení obvodu DS1307 k mikrokontroléru

Je potřeba vypočítat odpor R_P, a to podle následujícího vztahu.

$$R_P = \frac{t_r}{C_b} \qquad 4\text{-}3$$

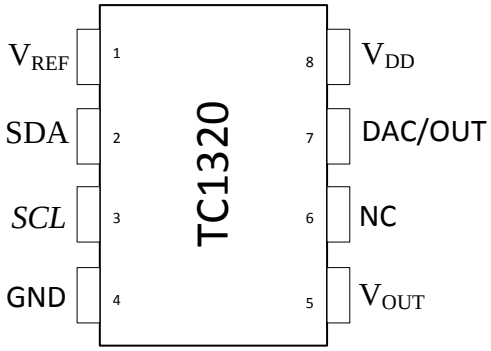
Kde t_r je doba náběhu signálů SCL a SDA a C_b je celková kapacita na jednom vodiči sběrnice. V tomto případě je maximální hodnota t_r rovna 1000 ns a C_b určíme jako součet vstupní kapacity pinů mikrokontroléru a připojeného obvodu. Vstupní kapacita pinů obvodu DS1307 je 10 pF a kapacita pinů mikrokontroléru je také 10 pF. Hodnota odporů je tedy 50 kΩ.

[5] Parafrázováno z [5]

4.4 DA převodník TC1320

Posledním zařízením, které jsem si vybral je osmibitový DA převodník TC1320 podporující komunikaci přes rozhraní IIC. Výstupem převodníku je napětí v rozsahu od hodnoty zemnící svorky až po napájecí napětí. Obvod může pracovat při napájecím napětí od 2,7 V do 5,5 V. Zařízení může být uvedeno do pohotovostního režimu s velmi malým odběrem 350 µA.

4.4.1 Vstupy a výstupy

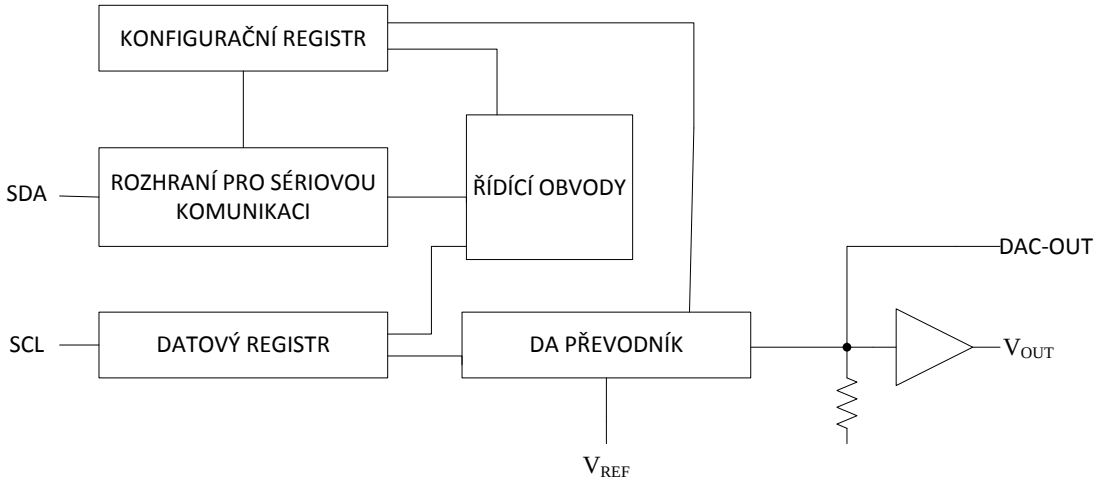


Obrázek 4-9: Přehled pinů obvodu TC1320

K připojení napájecího napětí slouží pin V_{DD}, k připojení země GND. Pro komunikaci slouží dva piny SDA a SCL, které klasicky označují obousměrný port pro data a vstup pro hodinový signál. Výstupní analogová hodnota je na pinu V_{OUT}. Také obsahuje pin pro připojení externí referenční hodnoty V_{REF}.

4.4.2 Princip činnosti

Na obrázku je vidět blokové schéma obvodu. Zařízení sestává kromě samotného převodníku z datového a konfiguračního registru, rozhraní pro sériovou komunikaci, řídící logiky a výstupního zesilovače.



Obrázek 4-10: Blokové schéma obvodu TC1320

Datový registr jak již název napovídá je místo, kam se ukládá přijatá digitální hodnota k převodu. Konfigurační registr má pouze jeden využitý bit a to ten nejnižší, který slouží k nastavení režimu, ve kterém má zařízení pracovat. Pokud bit nastavíme na 0, bude pracovat v normálním režimu, 1 znamená, že se zařízení uvede do pohotovostního režimu. Hodnota, která se objeví na výstupu, je dána vztahem 4-4.

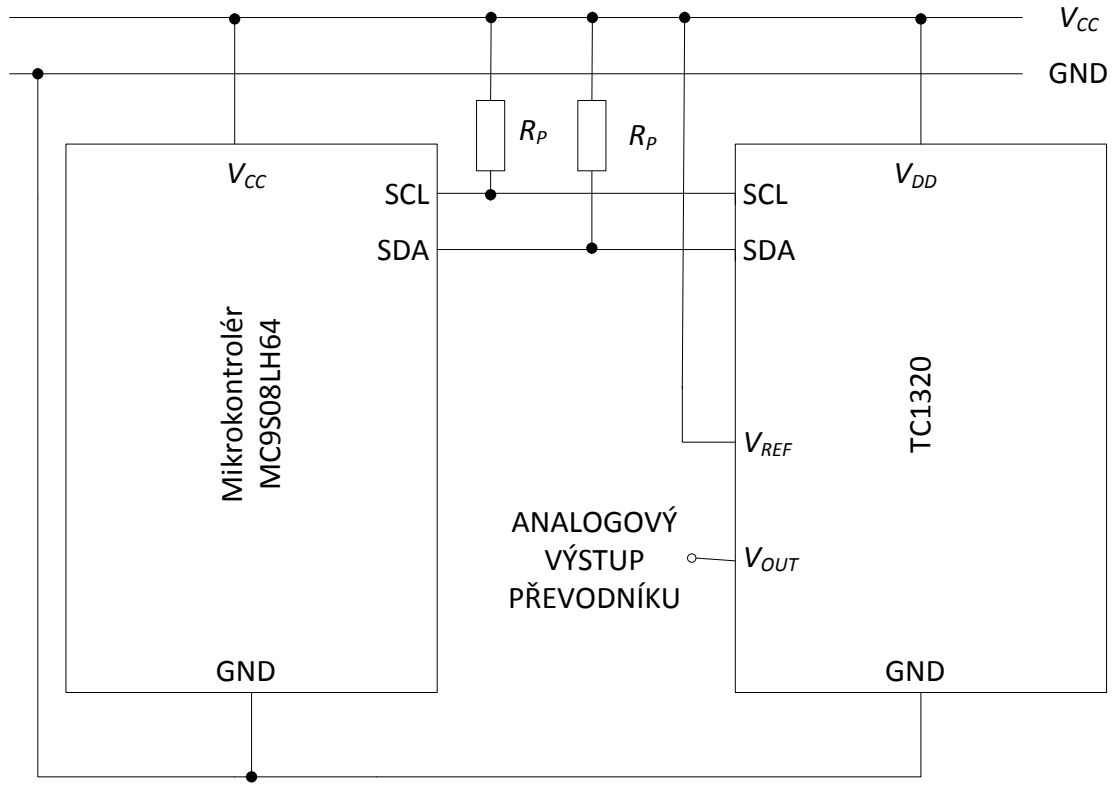
$$V_{OUT} = V_{REF} \frac{DATA_D}{256} \qquad 4\text{-}4$$

4.4.3 Komunikace přes IIC

Zařízení komunikuje podle standardního protokolu IIC, komunikace tedy začíná start bitem, který je realizován sestupnou hranou SDA během držení logické 1 na pinu SCL. Dále následuje adresa zařízení, která je v tomto případě 1001000 doplněná o bit, který určuje, zdali chceme posílat nebo přijímat data, poté jeden nebo více datových bytů. Přenos je ukončen stop bitem, realizovaným nástupnou hranou SDA během držení SCL pinu na hodnotě logické 1. Postup při zapisování hodnoty je následující. Odešleme start bit, poté následuje adresa doplněná o pokyn k zapisování hodnot, tedy 0, dále následuje adresa registru, do kterého chceme zapisovat. Datový registr má adresu 00000000 a konfigurační registr 00000001. Nakonec zapíšeme požadovanou hodnotu. Pro čtení dat je potřeba nejprve odeslat adresu zařízení s pokynem pro zápis, odeslat adresu registru, poté znovu odeslat adresu zařízení, tentokrát s pokynem pro čtení a zařízení odešle hodnotu, která se nachází ve zvoleném registru. Pokud se znovu spojíme se zařízením a odešleme mu jeho adresu ihned s pokynem pro čtení, nenastavíme tedy nejprve registr, ze kterého se má číst, bude odeslána hodnota registru, který byl zvolen při minulém čtení.[6]

[6] Parafrázováno z [6]

4.4.4 Blokové schéma připojení k mikrokontroléru



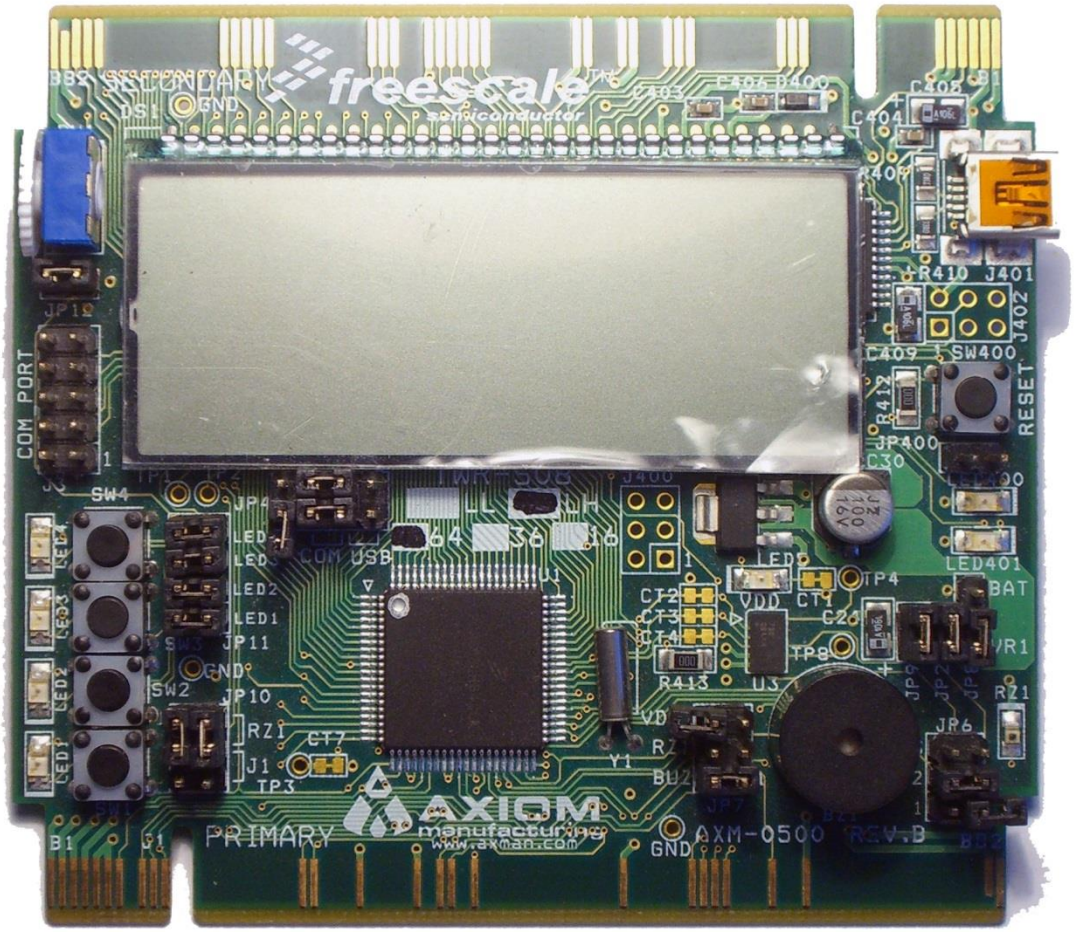
Obrázek 4-11: Připojení obvodu TC1320 k mikrokontroléru

Hodnoty pull-up rezistorů spočítáme opět podle vztahu **4-3**. Hodnota t_r je rovna 1000 ns a vstupní kapacita C_b opět získáme součtem kapacity pinů mikrokontroléru, což je 10 pF a kapacity pinů připojeného obvodu, tedy 5 pF. Rezistory by tedy měly mít hodnotu přibližně 66 kΩ.

5 HARDWAROVÁ REALIZACE

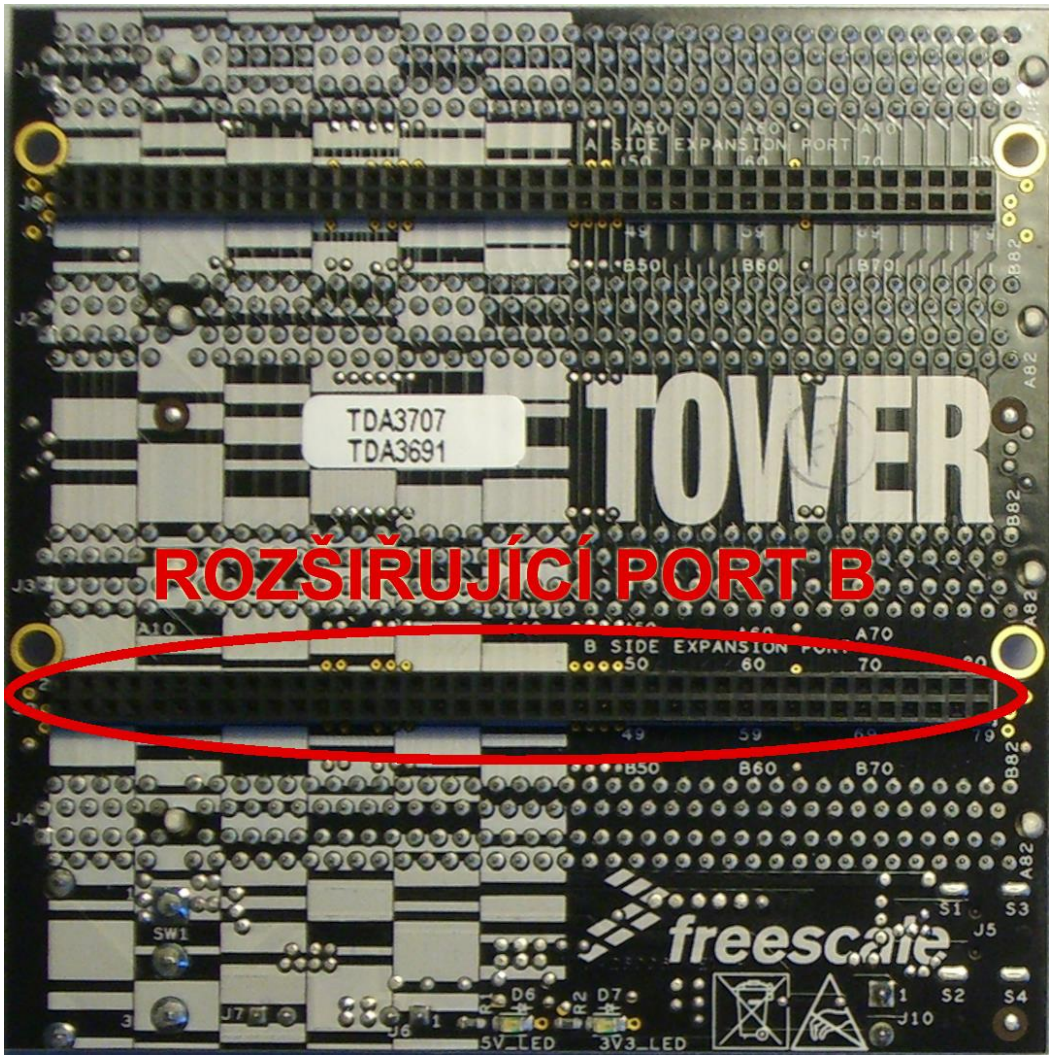
V následující kapitole se budu zabývat hardwarovými prostředky, které jsem při této práci využíval, a přípravky, které jsem navrhnul. Jedná se tedy o vývojovou desku TWR-S08LH64 a čtyři přípravky s výše popsanými externími obvody.

5.1 Vývojová deska TWR-S08LH64



Obrázek 5-1: Vývojová deska TWR-S08LH64

V této práci byla použita vývojová deska TWR-S08LH64, která obsahuje mikrokontrolér MC9S08LH64 a různé periferie jako například segmentový display, LED diody, tlačítka, konektor pro sériovou komunikaci, USB konektor atd. Vývojovou desku je možné rozšířit o postranní moduly (primární a sekundární), které obsahují porty s veškerými vstupy a výstupy mikrokontroléru. Vstupy a výstupy modulů SPI a IIC, které jsem využíval, jsou umístěny na rozšiřujícím portu B, zvýrazněném na obrázku 5-2.



Obrázek 5-2: Primární rozšiřující modul mikrokontroléru

Konkrétní rozmístění jednotlivých vstupů a výstupů ukazuje následující tabulka:

Tabulka 5-1: Rozmístění vybraných V/V mikrokontroléru na rozšiřujícím modulu

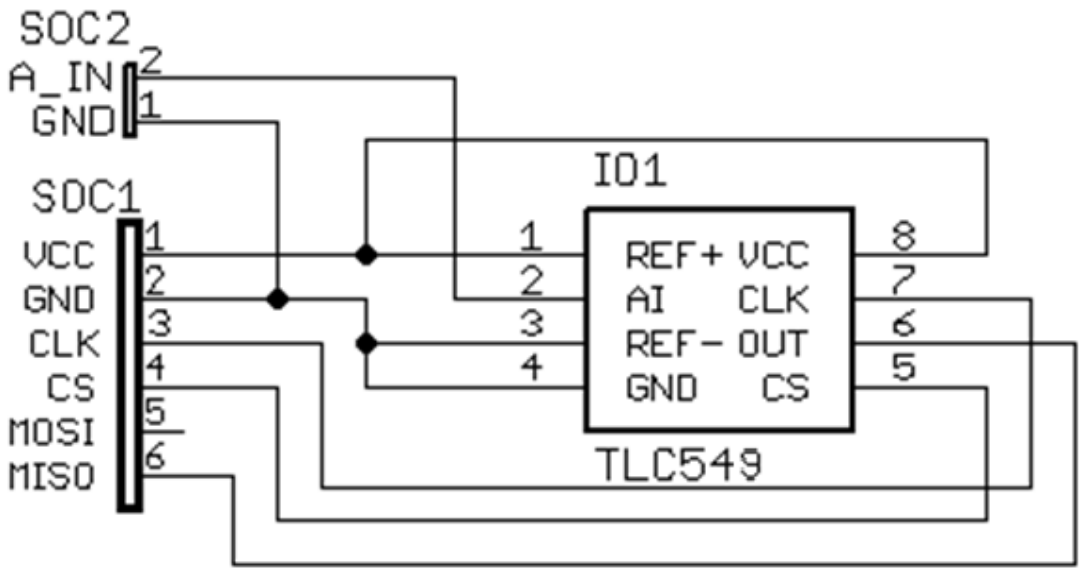
Umístění na rozšiřujícím modulu	Vstup/výstup mikrokontroléru
Pri_B01	Napájení 5V
Pri_B02	Zem GND
Pri_B07	PTB6/SPSCK
Pri_B09	PTB7/SS
Pri_B010	PTB5/MOSI/SCL
Pri_B011	PTB4/MISO/SDA

5.2 Přípravky s externími obvody

Vytvořil jsem čtyři přípravky s výše uvedenými externími obvody. Ke každému z nich jsem navrhl schéma zapojení a vlastní desku plošných spojů pomocí návrhového softwaru Eagle. Na následujících obrázcích a v tabulkách je vidět schéma každého přípravku, odpovídající náhled na DPS a tabulka použitých součástek. Kompletní výkresová dokumentace, tedy osvitové masky plošných spojů a osazovací plány se nacházejí v příloze A.

5.2.1 AD převodník TLC549

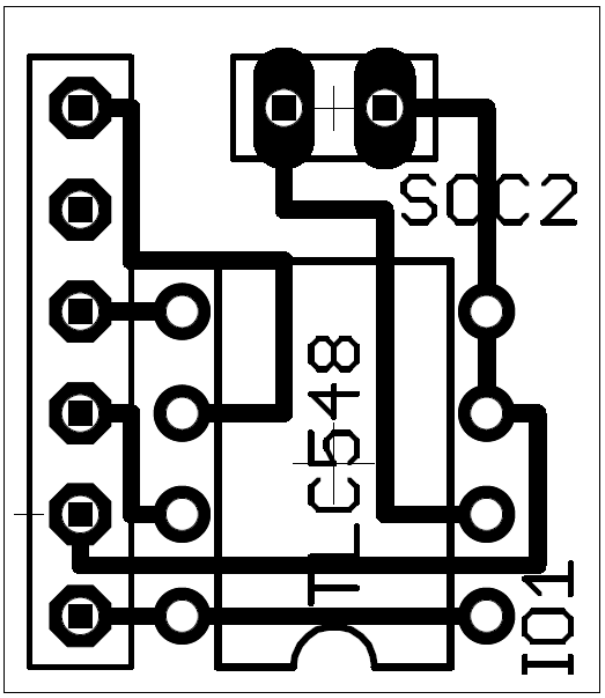
Schéma zapojení je velmi jednoduché, stačilo pouze vyvést vstupy a výstupy AD převodníku k paticím. K první patici bude přivedena propojka k rozšiřujícímu portu B vývojové desky. Nachází se zde napájecí vodiče a vodiče patřící ke sběrnici SPI, které jsou připojeny k odpovídajícím vstupům a výstupům převodníku. Dále je potřeba připojit referenční vstupy na napájení a zem, čímž se nastaví maximální rozsah převodníku na 0 – 5V. Analogový vstup AD převodníku je přiveden ke druhé patici.



Obrázek 5-3: Schéma zapojení přípravku s TLC549

Tabulka 5-2: : Seznam součástek přípravku s TLC549

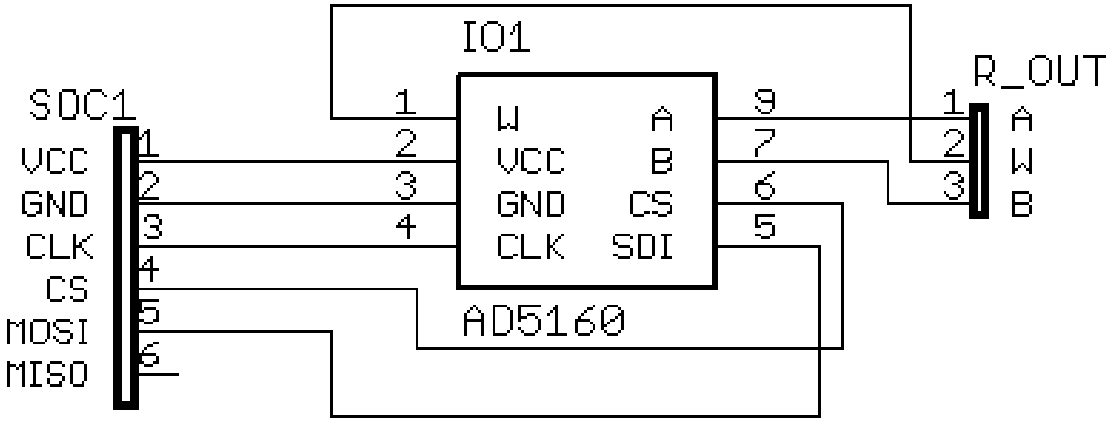
Součástka	Značka	Pouzdro
TLC549	IO1	DIP8
Patice 6 pinů	IN	
Patice 2 piny	ANALOG_IN	



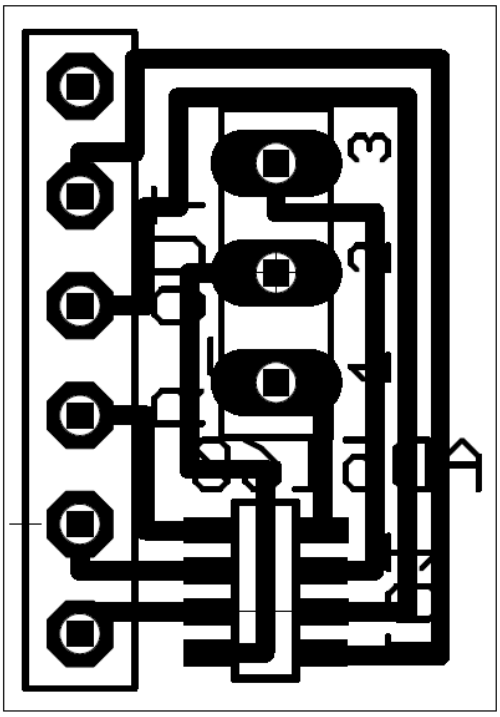
Obrázek 5-4: DPS přípravku s TLC549

5.2.2 Digitální potenciometr AD5160

Zapojení druhého zařízení pracujícího se sběrnicí SPI je opět velmi jednoduché. Zde stačilo pouze vyvést vstupy a výstupy k paticím. Opět vidíme jednu patici pro propojení s mikrokontrolérem a druhou s přivedeným výstupem obvodu.



Obrázek 5-5: Schéma zapojení přípravku s AD5160



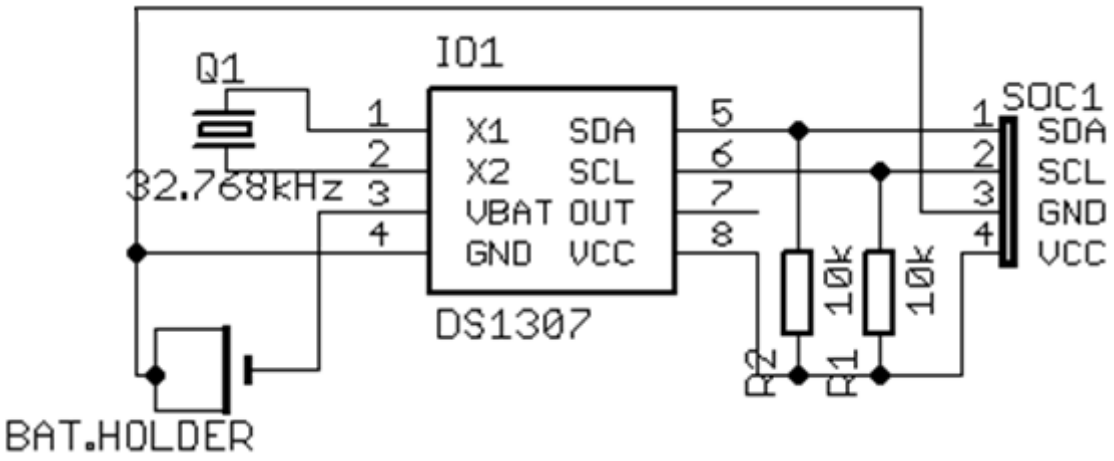
Obrázek 5-6: DPS přípravku s AD5160

Tabulka 5-3: Seznam součástek přípravku s AD5160

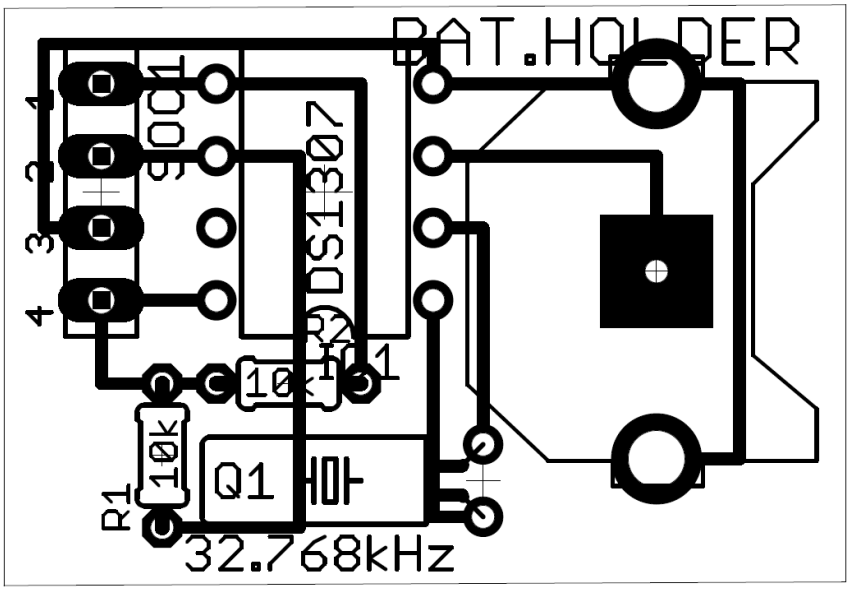
Součástka	Značka	Pouzdro
AD5160	IO1	SOT23-8
Patice 6 pinů	SOC1	
Patice 3 piny	R_OUT	

5.2.3 Obvod reálného času DS1307

U přípravků se zařízeními, která komunikují po sběrnici IIC, potřebujeme přidat pull-up rezistory k vodičům SDA a SCL. Ty jsou společně s napájením a zemnícím vodičem přivedeny k patici, přes kterou budou propojeny s mikrokontrolérem. Pro funkčnost obvodu reálného času je potřeba připojit oscilátor 32,768kHz a dále obvod umožňuje připojení záložního zdroje napájení. Přidal jsem tedy kostřičku pro baterii CR1220.



Obrázek 5-7: Schéma zapojení přípravku s DS1307



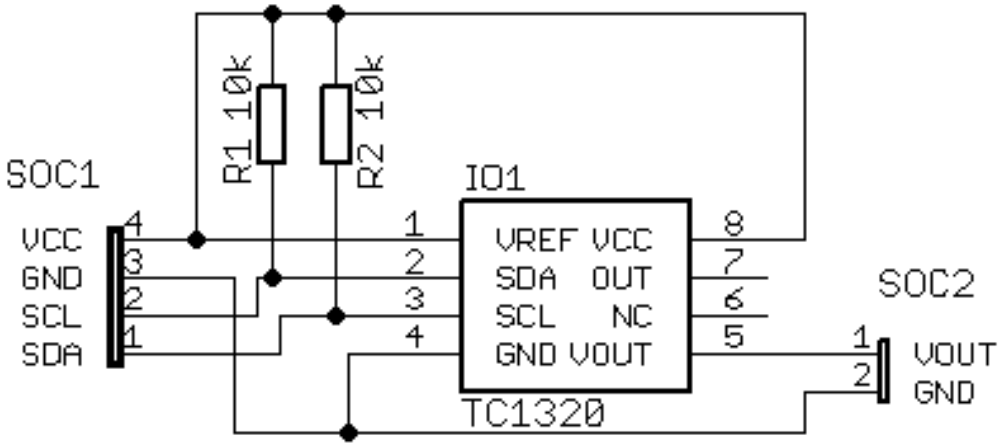
Obrázek 5-8: DPS přípravku s DS1307

Tabulka 5-4: Seznam součástek přípravku s DS1307

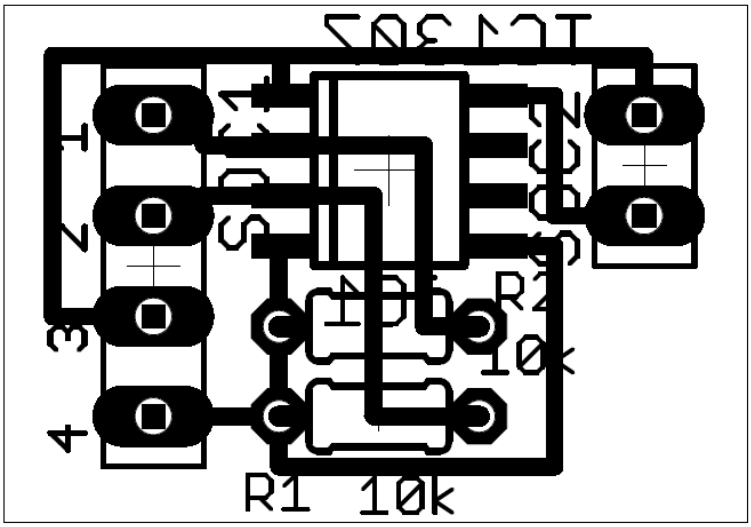
Součástka	Značka	Pouzdro
DS1307	IO1	DIP8
Oscilátor 32,768kHz	Q1	TC38H
Odpor 10k	R1,R2	0204/5
Kostřička na baterku	BAT.HOLDER	CR1220
Patice 4 piny	SOC1	

5.2.4 DA převodník TC1320

Poslední vytvořený přípravek obsahuje DA převodník TC1320, k jehož pinům SDA a SCL jsou opět připojeny pull-up rezistory. Dále již stačilo přivést vstupy a výstupy obvodu k odpovídajícím pinům dvou patic. Opět je zde jedna patice pro propojení s mikrokontrolérem a druhá, na kterou je přiveden analogový výstup převodníku.



Obrázek 5-9: Schéma zapojení přípravku s TC1320



Obrázek 5-10: DPS přípravku s TC1320

Tabulka 5-5: Seznam součástek přípravku s TC1320

Součástka	Značka	Pouzdro
TC1320	IO1	SOIC8
Odpor 10k	R1,R2	0204/5
Patice 4 piny	SOC1	
Patice 2 piny	SOC2	

6 SOFTWAROVÁ REALIZACE

V této kapitole se budu zabývat popisem vytvořeného softwaru. K programování jsem používal vývojové prostředí CodeWarrior a programy jsem vytvářel v jazyce C. Pro každou sběrnici jsem vytvořil knihovnu, která zajišťuje základní komunikaci, tedy odesílání a přijímání dat. Jedná se o knihobny *spilib* a *iiclib*, jejichž zdrojové kódy se necházejí v příloze B. Pro oba obvody s rozhraním SPI stačí pouze tato knihovna, jelikož jediné co tyto obvody umí, je buď odeslání bytu, nebo přijetí bytu. Co se týče obvodů s rozhraním IIC, tak u těch je funkcí, kterými disponují, více, tudíž jsem pro každé zařízení ještě vytvořil knihovnu, která daný obvod ovládá. Knihovny mají název *ds1307lib* a *tc1320lib*, a jejich zdrojové kódy jsou taktéž v příloze B. Ke každému přípravku jsem vytvořil a odladil program, který tyto knihovny využívá a tím i ověřuje jejich funkčnost. Všechny čtyři kompletní projekty vytvořené v CodeWarrioru se nacházejí v příloze C.

6.1 Knihovna pro komunikaci po sběrnici SPI

Pro komunikaci pomocí sběrnice SPI jsem vytvořil jednu společnou knihovnu, kterou využívají obě vybraná zařízení s tímto rozhraním. Knihovna obsahuje tři funkce: jednu pro inicializace modulu SPI, druhou pro odeslání jednoho bytu a poslední pro přečtení jednoho bytu. Oba vybrané obvody s tímto rozhraním jsou velmi jednoduchá zařízení a žádné další funkce už nepotřebují.

6.1.1 Popis jednotlivých funkcí

Dále se budu věnovat popisu jednotlivých funkcí. První funkcí knihovny pro sběrnici SPI je inicializace modulu.

Zdrojový kód 1: Funkce pro inicializaci SPI

```
void SPIInit(void){
  SOPT2_SPIPS=1;
  SPIC1 = 0b01010000;
  SPIBR = 0b00100010;
  PTBDD_PTBDD7 = 1;
  PTBPE_PTBPE7 = 1;
  SS = 1;
}
```

Nejprve je potřeba pomocí bitu SPIPS v registru SOPT2 vybrat sadu vstupů a výstupů SPI modulu tak, aby odpovídala tabulce 5-1. Dále je potřeba nastavit řídící registr SPI, konkrétně zapnout celý modul SPI a přepnout do režimu master. Dále je zde nastavena rychlost přenosu a jako poslední musíme nakonfigurovat pin pro výběr slave zařízení SS. Jelikož jsme už nic dalšího nenastavovali, tak pin SS funguje jako standardní V/V

pin, který není ovládán modulem SPI, musíme jej tedy ručně nastavit jako výstup a zapnou na něm interní pull-up rezistor.

Další funkce je pro odeslání jednoho bytu. Vzhledem k jednoduchosti zařízení, pro které je tato knihovna vytvořena, není potřeba řešit odesílání více bytů za sebou. Nejprve je potřeba uvést pin SS do stavu logické 0, čímž se aktivuje slave zařízení, dále je potřeba ověřit, zdali je prázdný buffer pro odesílání dat. Pokud není, program setrvá ve smyčce, dokud se tak nestane. Samotný přenos dat začíná zapsáním do datového registru. Pak už jen stačí počkat, než je přenos dat ukončen a uvést pin SS do stavu logické 1.

Zdrojový kód 2: Funkce pro odesílání dat přes SPI

```
void SPITransmit(unsigned char data){
	SS = 0;
	while(!(SPIS & 0x20))
	__RESET_WATCHDOG();
	SPID = data;
	while(!(SPIS & 0x80))
	__RESET_WATCHDOG();
	SS = 1;
}
```

Poslední funkcí této knihovny je funkce pro přijetí jednoho bytu. Opět nám stačí odesílat jen jeden byte, jelikož jich AD převodník, pro ktey je funkce vytvořena, více najednou neodesílá.

Zdrojový kód 3: Funkce pro příjem dat přes SPI

```
unsigned char SPIReceive(void){
	SS = 0;
	while(!(SPIS & 0x20))
	    __RESET_WATCHDOG();
	SPID = 0x00;
	while(!(SPIS & 0x80))
	      __RESET_WATCHDOG();
	SS = 1;
	return(SPID);
}
```

Většina kódu je stejná jako u odesílání dat, tedy nejprve aktivujeme slave zařízení pomocí pinu SS a čekáme, dokud není prázdný buffer. K zahájení přenosu je opět potřeba zapsat to datového registru. I když chceme pouze číst data, musíme nějaký byte odeslat, aby začal přenos. Odeslaný byte tedy může být náhodný, jelikož AD převodník žádná data nepřijímá. Během přenosu se tento náhodný byte odešle a na jeho místo se nasune přijatý byte, který po ukončení přenosu můžeme přečíst.

6.1.2 Ověření funkčnosti

Pro ověření funkčnosti slouží dva programy, které se nacházejí v příloze C. Jeden z nich je vytvořen pro AD převodník TLC549, který využívá funkci pro příjem dat. Na začátku programu jsou definovány makra pro tlačítka a LED na vývojové desce. Poté je zavolána funkce pro inicializaci a v následující nekonečné smyčce je volána funkce pro čtení dat přes SPI. Aby bylo možné alespoň něco vidět, rozdělil jsem hodnotu na intervaly a rozsvěcuji LED indikátory podle velikosti přiloženého napětí na vstupu AD převodníku.

Zdrojový kód 4: Hlavní část programu pro čtení z AD převodníku TLC549

```
SPIInit();

  for(;;) {
        voltage = SPIReceive();
        LED1=1;
        LED2=1;
        LED3=1;
        LED4=1;

        if (voltage>50) LED1=0;
        if (voltage>100) LED2=0;
        if (voltage>150) LED3=0;
        if (voltage>200) LED4=0;

    __RESET_WATCHDOG();
  }
```

Druhý program je vytvořen pro digitální potenciometr AD5160 a testuje funkci pro odesílání dat, pomocí které nastavuje hodnotu odporu na výstupu obvodu. Na začátku jsou opět definovány makra pro tlačítka a LED, dále je zavolána funkce pro inicializaci SPI a v nekonečné smyčce je pomocí tlačítek nastavována proměná, která je následně odeslána po sběrnici SPI. Hodnotu na výstupu můžeme ověřit například multimetrem.

Zdrojový kód 5: Hlavní část programu pro nastavení potenciometru AD5160

```
SPIInit();

for(;;) {
  if (!SW1) {
    if(resistance>0) resistance++;
  }
  if (!SW2) {
    if(resistance<255) resistance--;
  }
  SPITransmit(resistance);

  __RESET_WATCHDOG();
}
```

6.2 Knihovna pro komunikaci po sběrnici IIC

Stejně jako knihovna pro komunikaci přes SPI obsahuje i tato funkce pro inicializaci modulu, odesílání dat a jejich příjem. Tentokrát již obvody potřebují přenos více než jednoho bytu, tudíž jsou k tomu přizpůsobeny funkce. Koncepce knihovny vychází z [1]. Princip činnosti je takový, že funkce pro odeslání a přijímání dat odešlou pouze adresu slave zařízení a zapíší do globální proměnné počet bytů k odeslání nebo přijetí. O zbytek se už postará obsluha přerušení modulu IIC. Přerušení od IIC je vyvoláno odesláním nebo přijetím jednoho bytu.

6.2.1 Popis jednotlivých funkcí

Než začnu popisovat vytvořené funkce, uvedu nejprve globální proměnné, které v knihovně používám. Jsou to *dataTransfering*, která indikuje probíhající přenos dat, *bytesToSend*, ve které je uložen počet bytů k odeslání nebo přijetí, *bytesSent*, která funguje jako počítadlo odeslaných či přijatých bytů, a *direction*, která indikuje směr přenosu dat. Poslední proměnnou, kterou využívám, je ukazatel na pole bytů k odeslání nebo na pole bytů, do kterého jsou ukládána přijatá data. Dále se již budu věnovat popisu jednotlivých funkcí. První je inicializace modulu IIC.

Zdrojový kód 6: Funkce pro inicializaci IIC

```
void IICInit(void){
  IICC1 = 0xC8;
  IICF = 0x95;
  dataTransfering=0;
}
```

Funkce obsahuje nastavení řídícího registru 1, konkrétně zapnutí IIC, zapnutí přerušení od tohoto modulu a vypnutí odesílání acknowledge bitu, který bude zapnut až ve chvíli, kdy bude potřeba. Dále je zde nastavena rychlost přenosu dat a vynulována pomocná proměnná, která indikuje probíhající přenos.

Další funkce je pro odesílání dat, která odesílání pouze inicializuje a o zbytek se již stará obsluha přerušení.

Zdrojový kód 7: Funkce pro odesílání dat přes IIC

```
void IICTransmit(unsigned char addr, unsigned char bytes, char* ptr){
  direction = 1;
  bytesToSend = bytes;
  bytesSent =  0;
  dataTransfering = 1;
  databuffer = ptr;

  IICC1_IICEN = 0;
  IICC1_IICEN = 1;

  IICS_IICIF = 1;

  IICC1_TX = 1;
  IICC1_MST = 1;
  IICD=(addr<<1)&0xFE;
  while(dataTransfering) __RESET_WATCHDOG();
}
```

Nejprve jsou nastaveny globální proměnné, poté následuje vypnutí a zapnutí modulu IIC. Je potřeba tuto akci provést, jelikož se občas stává, že některé flagy zůstanou aktivní, i když by teoreticky neměly a může tedy dojít k neočekávanému chování. Doporučuje se tedy modul vypnout a zapnout. Poté se IIC nastaví do režimu odesílání dat a odešle se start bit, což je realizováno zapnutím master režimu. Následně je možné odeslat adresu slave zařízení doplněnou o R/W bit, který je v případě odesílání dat 0. Zbytek již řeší obsluha přerušení, která bude popsána dále v dokumentu. Ve funkci je ještě přidána vyčkávací smyčka, dokud se neukončí přenos.

Funkce pro příjem dat je velmi podobná, opět pouze inicializuji přenos.

Zdrojový kód 8: Funkce pro odesílání dat přes IIC

```
void IICReceive(unsigned char addr, unsigned char bytes, char* ptr){
  direction = 0;
  bytesToSend = bytes;
  bytesSent =  0;
  dataTransfering = 1;
  databuffer = ptr;

  IICS_IICIF = 1;

  IICC1_TXAK = 0;
  IICC1_TX = 1;
  IICC1_MST = 1;

  IICD=(addr<<1)|0x01;
  while(dataTransfering) __RESET_WATCHDOG();

}
```

Na začátku opět nastavíme globální proměnné, resetujeme modul IIC. Navíc je zde potřeba zapnout odesílání acknowledge bitu po přijetí nového bytu. Dále nastavíme

odesílání dat, jelikož nejprve musíme odeslat adresu slave zařízení, poté odešleme start bit a adresu slave zařízení doplněnou o R/W bit, tentokrát 1 pro příjem dat. Další postup je opět součástí obsluhy přerušení, kterou si nyní rozebereme. Pro začátek ještě zmíním, že přerušení je mimo jiné vyvoláno přenesením jednoho bytu, ať už jde o přijatý nebo odeslaný.

Zdrojový kód 9: Hlavička obsluhy přerušení modulu IIC

```
interrupt VectorNumber_Viic void IIC_Control_handler(void){
  IICS_IICIF=1;
```

Nejprve vynulujeme bit indikující přerušení, což se provede zapsáním logické 1 do tohoto bitu. Následující kód se provede, pokud jsme v režimu přenosu dat, což nastavuji ve funkcích pro příjem nebo odesílání dat.

Zdrojový kód 10: Část obsluhy přerušení pro odesílání dat

```
  if(dataTransfering==1){
    if(direction==1){
      if(bytesSent<bytesToSend){
        IICD = *(databuffer + bytesSent);
       bytesSent++;
        return;
      }
      else{
        dataTransfering = 0;
        IICC1_TX = 0;
        IICC1_MST = 0;
        return;
        }
    }
```

První část se zabývá odesíláním dat, tedy pokud *direction = 1*. Nejprve testujeme, jestli jsou nějaká data k odeslání. Pokud ano, je odeslán jeden byte, navýšeno počítadlo odeslaných bytů a přerušení se ukončí. Až je tento byte odeslán, přijde znovu přerušení. Pokud jsme již odeslali všechny byty, nastavíme proměnnou *dataTransfering* do stavu logické 0, vypneme režim odesílání dat, odešleme stop bit nastavením bitu *MST* na 0 a ukončíme přerušení.

Další část se týká režimu příjmu dat, která je o něco složitější.

Zdrojový kód 11: Část obsluhy přerušení pro příjem dat

```
    else{ /* Příjem dat*/
      if(IICC1_TX==1){
        IICC1_TX = 0;
        IICD;
        return;
      }
```

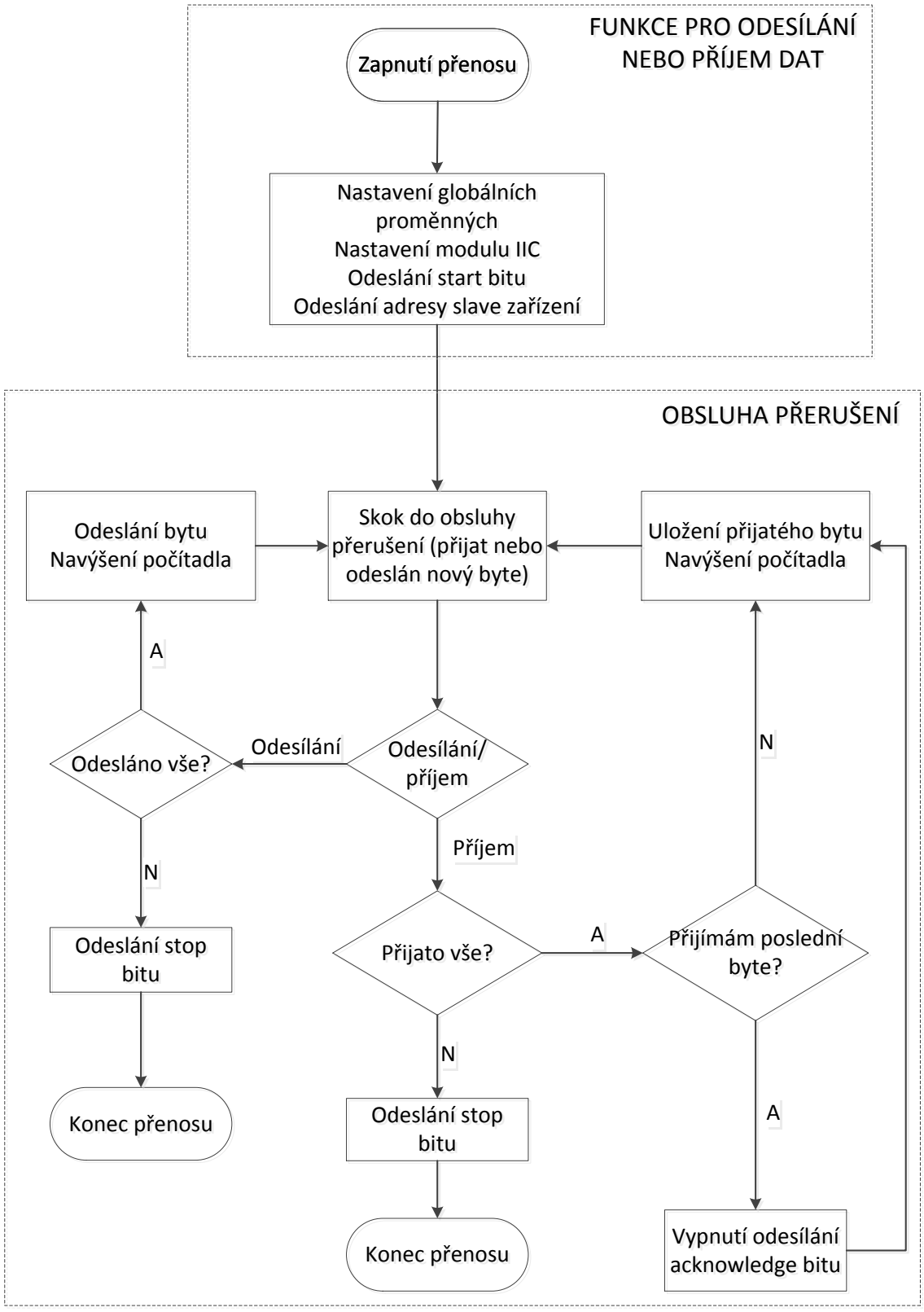
Nejprve musím zajistit, aby se po prvním skoku do přerušení po odeslání adresy slave zařízení přepnul modul IIC do režimu přijmu dat. Využívám tedy toho, že tento stav nastává ve chvíli, kdy mám globální proměnnou *direction* nastavenou do logické nuly, ale bit *TX* je ještě v logické 1. Kromě přepnutí do přijímacího režimu, musím ještě přečíst datový registr, abych vynuloval flag, který indikuje přijetí nebo odeslání nového bytu. Poté teprve začne přenos dalšího bytu a můžu ukončit přerušení.

Zdrojový kód 12: Část obsluhy přerušení pro příjem dat

```
      if(bytesSent<bytesToSend){
        if((bytesSent+1) == bytesToSend){
          IICC1_TXAK = 1;
        }
        *(databuffer + bytesSent) = IICD;
        bytesSent++;
        return;
        }
      else{
        dataTransfering=0;
        IICC1_MST = 0;
        return;
      }
    }
  }
}
```

Zbytek je velmi podobný odesílání dat. Opět při každém průchodu kontroluji jestli jsem již přijal všechna data. Pokud ne, uložím nově přijatý byte do pole, v opačném případě ukončím přenos stop bitem a vynulováním proměnné *dataTransfering*. Musím ovšem dbát ještě na to, abych po posledním přijatém bytu neodeslal acknowledge bit, tudíž před posledním bytem musím nastavit bit *TXAK* do stavu logické 1.

Princip výše popsaných funkcí je znázorněn na vývojovém diagramu (Obrázek 6-1), který ukazuje, jak je celý princip přenosu rozdělen mezi inicializační funkce a obsluhu přerušení.



Obrázek 6-1: Vývojový diagram obsluhy IIC modulu

6.3 Knihovna pro obsluhu obvodu reálného času DS1307

Tato knihovna využívá funkce výše popsané knihovny pro komunikaci přes sběrnici IIC. Obsahuje funkce pro jeho ovládání, tedy zápis všech položek data a času nejednou nebo zvlášť, čtení dat z obvodu, a také převod přijatých dat na jednotky času a data.

6.3.1 Popis jednotlivých funkcí

Nejprve jsem si nadefinoval strukturu *time*, která obsahuje všechny položky, které jdou obvodu nastavit. Jedná se tedy o čas, den v týdnu a datum. Dále jsou ještě vytvořena makra pro dny v týdnu, aby bylo možné psát místo čísel zkratky PO – NE.

Dále se budu věnovat jednotlivým funkcím této knihovny. První je pro nastavení všech položek v obvodu reálného času najednou.

Zdrojový kód 13: Funkce pro nastavení data a času

```
void setRTC(time now){
      if (now.seconds>59) now.seconds=59;
      if (now.minutes>59) now.minutes=59;
      if (now.hours>23) now.hours=23;
      if (now.dayOfWeek<PO) now.dayOfWeek=PO;
      if (now.dayOfWeek>NE) now.dayOfWeek=NE;
      if (now.day<1) now.day=1;
      if (now.day>31) now.day=31;
      if (now.month<1) now.month=1;
      if (now.month>12) now.month=12;
      if (now.year>2099) now.year=2099;
      if (now.year<2000) now.year=2000;
      year-=2000;

      data[0] = 0x00;
      data[1] = ((now.seconds/10)<<4)|(now.seconds%10);
      data[2] = ((now.minutes/10)<<4)|(now.minutes%10);
      data[3] = ((now.hours/10)<<4)|(now.hours%10);
      data[4] = now.dayOfWeek;
      data[5] = ((now.day/10)<<4)|(now.day%10);
      data[6] = ((now.month/10)<<4)|(now.month%10);
      data[7] = ((now.year/10)<<4)|(now.year%10);

      IICTransmit(0b1101000,8,&data[0]);
}
```

Nejprve musím zajistit, že všechny proměnné, které jsou funkci předány, jsou v rozsahu hodnot data a času. Dále je potřeba naplnit pole dat, které předáme funkci pro odesílání přes sběrnici IIC. Tento obvod reálného času jako první datový byte očekává adresu registru, do kterého má zapisovat. Jelikož nastavujeme všechny hodnoty, začínáme od nultého registru v obvodu. Další prvky pole nastavuji v pořadí dle tabulky 4-2. Je ovšem

potřeba převést čísla do BCD formátu. Dále už jen zavolám funkci pro odeslání dat s patřičnými parametry.

Také jsem vytvořil sedm funkcí, které nastavují každý z těchto parametrů zvlášť. Takto rozdělené funkce se můžou hodit například tehdy, pokud budeme chtít pomocí displeje a tlačítek nastavovat aktuální čas za běhu programu. Jak již bylo zmíněno, externí obvod nejprve očekává adresu registru, do kterého má zapsat, tudíž v každé funkci musíme nejprve odeslat odpovídající adresu umístění daného údaje.

Zdrojový kód 14: Nastavení pouze hodin

```
void setHours(unsigned char hours){
      if (hours>23) hours=23;
      data[0] = 0x02;
      data[1] = ((hours/10)<<4)|(hours%10);
      IICTransmit(0b1101000,2,&data[0]);
}
```

Jako příklad uvádím nastavení pouze hodin. Jelikož se tento údaj nachází v registru s adresou 02_H, tak jako první byte v poli dat bude právě tato hodnota. Další položka bude nastavovaná hodina převedená opět do BCD formátu. Následně už jen stačí zavolat funkci pro odeslání dat.

Další funkce je pro čtení všech položek obvodu reálného času.

Zdrojový kód 15: Funkce pro čtení data a času

```
time readRTC(){
      int i;
      time now;
      data[0] = 0x00;

      IICTransmit(0b1101000,1,&data[0]);
      for(i=0;i<10000;i++);

      IICReceive(0b1101000,7,&data[0]);
      for(i=0;i<10000;i++);

      now.seconds = (((data[0]>>4)&0x0F)*10)+(data[0]&0x0F);
      now.minutes = (((data[1]>>4)&0x0F)*10)+(data[1]&0x0F);
      now.hours = (((data[2]>>4)&0x0F)*10)+(data[2]&0x0F);
      now.dayOfWeek = data[3];
      now.day = (((data[4]>>4)&0x0F)*10)+(data[4]&0x0F);
      now.month = (((data[5]>>4)&0x0F)*10)+(data[5]&0x0F);
      now.year = (((data[6]>>4)&0x0F)*10)+(data[6]&0x0F)+2000;
      return now;
}
```

Nejprve musím odeslat jeden byte s adresou registru, abych nastavil výchozí pozici, od které se budou číst data. Po krátké prodlevě zavolám funkci pro čtení dat a nakonec jen převedu data zpět z BCD formátu na jednotky času.

6.3.2 Ověření funkčnosti

K ukázce funkčnosti této knihovny byl vytvořen program, který se nachází v příloze C. Demonstrační program nastavuje a čte hodnoty z obvodu reálného času. Jelikož tyto funkce využívají knihovnu pro komunikaci po sběrnici IIC, je tímto ověřena i funkčnost této knihovny samotné. Hodnoty, které nastavuji si můžeme ověřit pouze tak, že budeme sledovat obsah proměnných přes prostředí CodeWarrior.

Zdrojový kód 16: Hlavní část programu pro ovládání obvodu DS1307

```
now.seconds = 1;
now.minutes = 2;
now.hours = 3;
now.dayOfWeek = 4;
now.day = 5;
now.month = 6;
now.year = 2014;

IICStart();

for(;;) {
  if(!SW1) setRTC(now);
  if(!SW2) now = readRTC();
  if(!SW3) setHours(7);

  __RESET_WATCHDOG();
}
```

6.4 Knihovna pro obsluhu DA převodníku TC1320

Uvedený DA převodník obsahuje dva registry, přičemž se oba dají nastavit nebo přečíst. Ke každému úkonu jsem tedy vytvořil funkci, která jej provede.

6.4.1 Popis jednotlivých funkcí

Jak již bylo zmíněno, funkce jsou celkem čtyři. Jedna dvojice vyměňuje data s datovým registrem, ve kterém je uložena hodnota, kterou obvod převádí na analogový výstup. Druhá dvojice funkcí vyměňuje data s řídícím registrem DA převodníku, který má pouze jednu funkci, a to přepínání mezi normálním a pohotovostním režimem.

Zdrojový kód 17: Funkce pro nastavení datového registru

```
void setData(unsigned char value){
      data[0] = 0x00;
      data[1] = value;

      IICTransmit(0b1001000,2,&data[0]);
}
```

Stejně jako u předchozího obvodu je jako první potřeba odeslat adresu registru, kterého budeme zapisovat, a poté zařízení očekává data, které do zvoleného registru chceme

zapsat. Adresu, která je pro datový registr 00_H, a nastavovanou hodnotu tedy uložím do pole dat a zavolám funkci pro odesílání z knihovny pro komunikaci přes IIC.

Zdrojový kód 18: Funkce pro čtení datového registru

```
unsigned char readData(void){
	int i;
	data[0] = 0x00;

	IICTransmit(0b1001000,1,&data[0]);
	for(i=0;i<10000;i++);

	IICReceive(0b1001000,1,&data[0]);
	for(i=0;i<10000;i++);

	return data[0];
}
```

Princip čtení je také velmi podobný jako u předchozího zařízení. Opět je nejprve potřeba odeslat jeden byte s adresou registru, který budeme později číst. Po krátké prodlevě můžu zavolat funkci pro čtení z knihovny pro komunikaci přes IIC.

DA převodník dále obsahuje ještě již zmíněný řídící registr s adresou 01_H, ve kterém můžeme nastavit jeden bit. Funkce je strukturou stejná jako ta pro nastavení datového registru, akorát odesílám jinou adresu registru.

Zdrojový kód 19: Funkce pro nastavení řídícígo registru (CR)

```
void setCR(unsigned char value){
	if (value>1) value = 1;

	data[0] = 0x01;
	data[1] = value;

	IICTransmit(0b1001000,2,&data[0]);
}
```

Obdobně je vytvořena funkce pro čtení tohoto údaje, která je opět téměř totožná s funkcí pro čtení datového registru.

Zdrojový kód 20: Funkce pro čtení z řídícího registru (CR)

```
unsigned char readCR(void){
	int i;

	data[0] = 0x01;
	IICTransmit(0b1001000,1,&data[0]);
	for(i=0;i<10000;i++);

	IICReceive(0b1001000,1,&data[0]);
	for(i=0;i<10000;i++);

	return data[0];
}
```

Opět nejprve odešlu adresu registru, ze kterého chci číst a po krátké prodlevě můžu zavolat funkci pro čtení.

6.4.2 Ověření funkčnosti

Funkčnost této knihovny ověřuje demonstrační program, který zapisuje a čte hodnoty DA převodníku. Opět se tímto ověřuje i funkčnost knihovny pro komminikaci přes sběrnici IIC. V programu pomocí dvou tlačítek nastavuji hodnotu odesílanou do převodníku a pomocí druhé dvojice tlačítek přepínám mezi normálním a pohotovostním režimem. V každém cyklu dále čtu oba registry DA převodníku a ukládám do jiných proměnných, aby bylo možné ověřit funkčnost čtení dat. Toto ověření je opět možné pouze sledováním těchto proměnných v prostředí CodeWarrior.

Zdrojový kód 21: Hlavní část programu pro řízení DA převodníku TC1320

```
IICInit();

for(;;) {
  if(!SW1) {
    if(dataout<255)
      dataout++;
  }
  if(!SW2) {
    if(dataout>0)
      dataout--;
  }
  if(!SW3) setCR(0);
  if(!SW4) setCR(1);
  setData(dataout);
  datain = readData();
  CRin = readCR();
__RESET_WATCHDOG();
}
```

7 ZÁVĚR

V této práci jsem měl za úkol navrhnout připojení externích zařízení k mikrokontroléru Freescale MC9S08LH64 přes sériové sběrnice SPI a IIC. Nejprve jsem si nastudoval princip zadaných sběrnic a seznámil se s uvedeným mikrokontrolérem. Popsal jsem základní vlastnosti mikrokontroléru MC9S08LH64, jeho procesor a periferní moduly, zejména pak ty, které se týkají sériové komunikace. I když jsme v předchozí výuce s tímto mikrokontrolérem pracovali, nevěděl jsem příliš o jeho periferních modulech, tudíž jsem během práce získal spoustu cenných informací jak o sériové komunikaci, tak i o ostatních modulech.

Dalším krokem byl výběr čtyř zařízení, dvou, které podporují sběrnici SPI, a dvou, které umožňují komunikovat přes IIC. Pro demonstraci komunikace pomocí SPI jsem zvolil AD převodník a digitálně řízený potenciometr, a zařízení, která komunikují pomocí IIC, jsem zvolil obvod reálného času a DA převodník. U každého z těchto zařízení jsem si nastudoval jejich vlastnosti a princip, jak fungují. Poté jsem se zaměřil na tom jakým způsobem se s nimi komunikuje, jaké data a jakým způsobem posílají nebo přijímají údaje. Vše je stručně zpracováno v této práci.

Následně jsem vytvořil čtyři přípravky s vybranými externími obvody a ke každému z nich navrhl schéma zapojení a desku plošných spojů. Pro účely testování byly přípravky vytvořeny pouze na univerzálním pájivém poli.

Dále jsem napsal knihovní funkce pro komunikaci s vybranými externími obvody. Vytvořil jsem tedy knihovnu pro komunikaci přes sběrnici SPI, kterou využívá AD převodník TLC549 i digitální potenciometr AD5160. Jelikož jsou zařízení, která komunikují po sběrnici IIC složitější a podporují více funkcí, tak kromě knihovny, která zajišťuje pouze komunikaci přes sběrnici IIC, jsem vytvořil další dvě knihovny. Jednu pro ovládání obvodu reálného času DS1307 a druhou pro ovládání DA převodníku TC1320.

Na závěr jsem tyto knihovny využil ve čtyřech demonstračních programech. Každý program komunikuje s jedním z externích zařízení a dohromady testují funkčnost všech vytvořených knihovních funkcí.

8 SEZNAM ZDROJŮ

[1] ARENDARIK, Stanislav. *How to Use IIC Module on M68HC08, HCS08, and HCS12 MCUs.* [online]. [cit. 2014-05-21]. Dostupné z: http://cache.freescale.com/files/microcontrollers/doc/app_note/AN3291.pdf.

[2] ANALOG DEVICES. *256-Position SPI Compatible Digital Potentiometer AD5160* [online]. 2003 [cit. 2014-01-06]. Dostupné z: http://www.analog.com/static/imported-files/data_sheets/AD5160.pdf.

[3] FREESCALE SEMICONDUCTOR. *MC9S08LH64 Reference Manual* [online]. Revision 5.1. 2012 [cit. 2014-01-06]. Dostupné z: http://cache.freescale.com/files/microcontrollers/doc/ref_manual/MC9S08LH64RM.pdf.

[4] TEXAS INSTRUMENTS. *TLC548C, TLC548I, TLC549C, TLC549I 8-bit analog to digital converters with serial control* [online]. 1983 [cit. 2014-01-06]. Dostupné z: http://www.ti.com/lit/ds/symlink/tlc549.pdf.

[5] MAXIM INTEGRATED PRODUCTS. *DS1307 64 x 8, Serial, I2C Real-Time Clock* [online]. 2008 [cit. 2014-01-06]. Dostupné z: http://datasheets.maximintegrated.com/en/ds/DS1307.pdf.

[6] MICROCHIP TECHNOLOGY INC. *TC1320 8-Bit Digital-to-Analog Converter with Two-Wire Interface* [online]. 2002 [cit. 2014-01-06]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/21386b.pdf.

[7] *Serial Peripheral Interface Bus*. Wikipedia, the free encyclopedia [online]. 2014 [cit. 2014-01-06]. Dostupné z: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus.

[8] *I²C*. Wikipedia, the free encyclopedia [online]. 2014 [cit. 2014-01-06]. Dostupné z: http://en.wikipedia.org/wiki/I%C2%B2C.

9 SEZNAM PŘÍLOH NA CD

A Schémata, DPS a výrobní plány přípravků s externími obvody

- A.1 Výkresová dokumentace k AD5160
- A.2 Výkresová dokumentace k DS1307
- A.3 Výkresová dokumentace k TC1320
- A.4 Výkresová dokumentace k TLC549

B Knihovní funkce

- B.1 Knihovna pro komunikaci po sběrnici SPI
- B.2 Knihovna pro komunikaci po sběrnici IIC
- B.3 Knihovna k ovládání obvodu DS1307
- B.4 Knihovna k ovládání obvodu TC1320

C Programy k ověření funkce knihoven

- C.1 Demonstrační program k obvodu AD5160
- C.2 Demonstrační program k obvodu DS1307
- C.3 Demonstrační program k obvodu TC1320
- C.4 Demonstrační program k obvodu TLC549