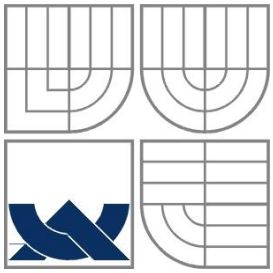
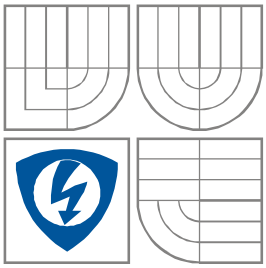


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

VÝŠKOMĚR PRO RC MODELY LETADEL
ALTIMETER FOR RC AIRCRAFT MODELS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Milan Kotulek
AUTHOR

VEDOUCÍ PRÁCE Ing. Jan Prokopec, Ph.D.
SUPERVISOR

BRNO, 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Bakalářská práce

bakalářský studijní obor
Elektronika a sdělovací technika

Student: Milan Kotulek ***ID:*** 125250
Ročník: 3 ***Akademický rok:*** 2011/2012

NÁZEV TÉMATU:

Výškoměr pro RC modely letadel

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte možnosti realizace výškoměru pro RC modely letadel. Navrhněte koncepci zařízení pro záznam letové polohy RC modelu. Při návrhu minimalizujte rozměry zařízení a proudový odběr. Připravte software pro zařízení a koncept software pro komunikaci s PC pro stahování údajů o letu. Realizujte navržené zařízení, vytvořte software pro komunikaci s PC. Ověřte funkci zařízení a sestavte podrobnou dokumentaci.

DOPORUČENÁ LITERATURA:

[1] BURKHARD, M. C pro mikrokontroléry. Praha: BEN - technická literatura, 2003.

[2] FRÝZA, T., FEDRA, Z., ŠEBESTA, J. Mikroprocesorová technika. Počítačová cvičení. Elektronické skriptum. Brno: FEKT VUT v Brně, 2009.

Termín zadání: 6.2.2012 ***Termín odevzdání:*** 25.5.2012

Vedoucí práce: Ing. Jan Prokopec, Ph.D.
Konzultanti bakalářské práce:

prof. Dr. Ing. Zbyněk Raida
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Cílem bakalářské práce bylo navrhnout koncepci zařízení pro záznam letové polohy RC modelu. K tomu bylo využito tlakové čidlo, paměť a to vše řízené mikrokontrolérem ATmega16. Data průběhu letu se budou stahovat pomocí rozhraní USB do počítače.

Klíčová slova

atmosférický tlak, ATmega16, I2C, EEPROM

Abstract

The aim of this thesis is to design a concept device for recording attitude RC model. For this is used a pressure sensor, memory, and all controlled ATmega16 microcontroller. Process of the flight data will be downloaded via USB to your computer.

Keywords

atmospheric pressure, ATmega16, I2C, EEPROM

Bibliografická citace

KOTULEK, M. *Výškoměr pro RC modely letadel*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav radioelektroniky, 2012. 33 s., 4 s. příloh. Bakalářské práce. Vedoucí práce: Ing. Jan Prokopec, Ph.D.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Výškoměr pro RC modely letadel 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 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 dne 25. května 2012 .....
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Jan Prokopec, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne 25. května 2012 .....
podpis autor

Obsah

Seznam obrázků

1. Úvod

Tématem práce bylo měření výšky modelu letadla. Toho se v tomto případě asi nejlépe dosáhne měřením tlaku snímačem. Ten má být umístěn v modelu a tudíž je kladen důraz na jeho co možná nejmenší váhu a velikost. Je tedy zřejmé, že nejlepší je použít integrovaný obvod který je nejen malý, ale má i minimální proudový odběr. Jak je uvedeno níže celé zapojení bude řízeno mikrokontrolérem a všechny obvody budou komunikovat po sběrnici I^2C.

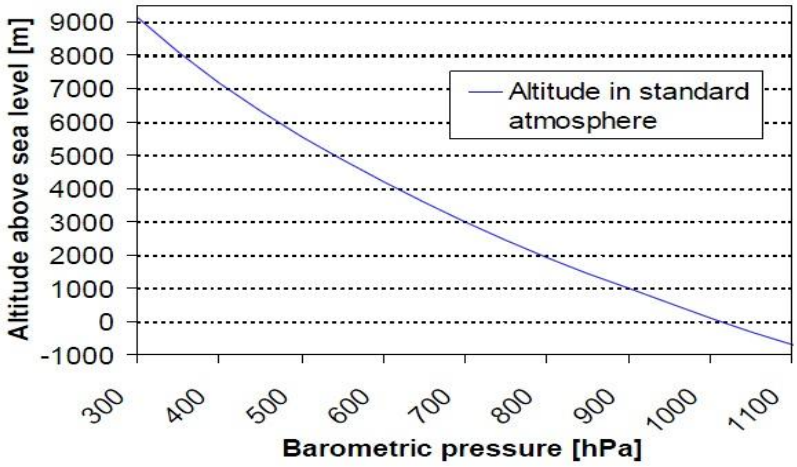
1.1. Atmosférický tlak

Na výpočet výšky v které se pohybuje letadlo lze použít hned tři způsoby měření tlaku. Vždy jde o nějaký přepočet změřeného tlaku na výšku, ale kvůli změnám v atmosféře se hodnoty tlaku na určitém místě s časem mění a to i dosti výrazně. Nelze proto říct, že ve sto metrech je určitý tlak. Vždy se musí počítat se zvolenou referenční hodnotou. Ta je tím co určuje jakou výšku ve výsledku měříme.

- prvním způsobem je, že výškoměr počítá s nadmořskou výškou a musíme mu tedy zadat jaká je její hodnota při startu a dále k ní přičítat a odečítat podle průběhu letu.
- dalším způsobem je použít jako referenční hodnotu tlak při startu a považovat tento bod za nulovou výšku.
- Posledním způsobem je měření letové hladiny, kdy se jako referenční hodnota nastaví 1013,25 hPa (29,92 in Hg) - pak po výpočtu bych dostanu tzv. letovou hladinu (flight level)

V datasheetu čidla je uvedena rovnice (1.1), která se dá použít k zobrazení výsledků ve tvaru nadmořské výšky. Je zde také ukázáno jak se dá změna tlaku převést na změnu výšky. Toho se dosáhne díky mezinárodnímu standardnímu modelu atmosféry (ISA) jenž je jejím zjednodušením a který říká jak se mění hustota,viskozita, tlak a teplota s rostoucí výškou. [3]

Výpočet nadmořské výšky [3] $$výška = 44330 * (1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5,255}}) \quad (1.1)$$



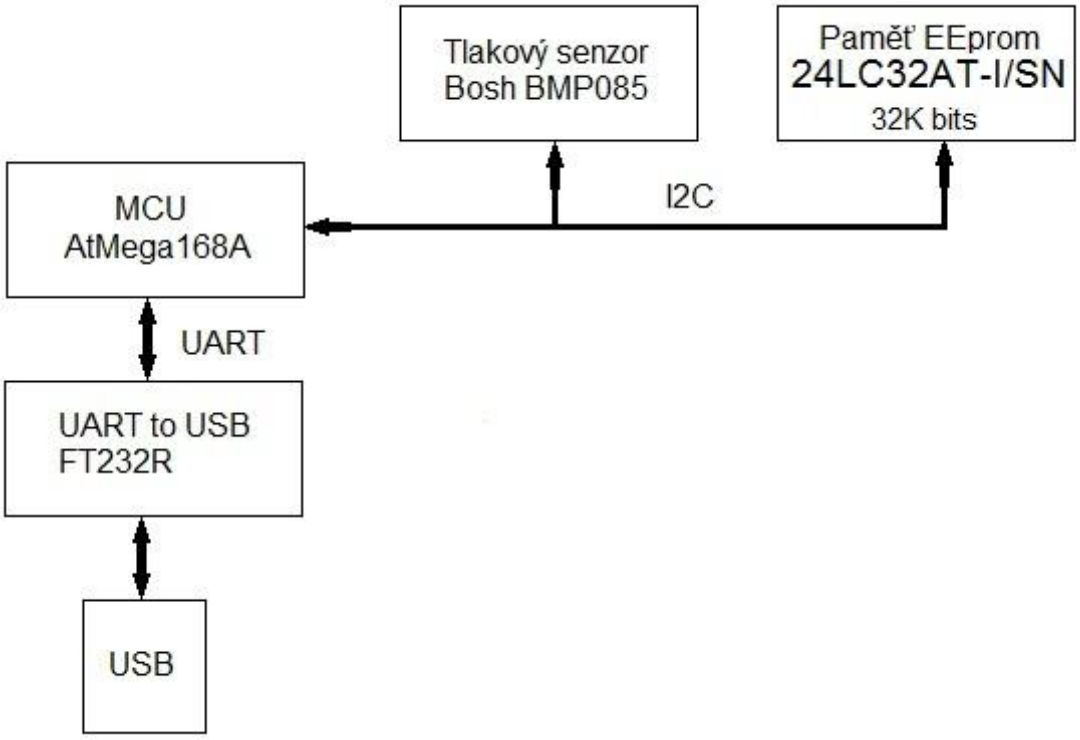
Obr 1.1 Závislost barometrického tlaku na nadmořské výšce [3]

Výpočet referenčního tlaku [3] $$p_0 = \frac{p}{(1-\frac{výška}{44330})^{5,255}} \quad (1.2)$$

V tomto zařízení se jeví jako nejlepší využití druhého z uvedených způsobů. Na začátku letu zařízení změří tlak v místě startu. Tuto hodnotu bude dále brát jako nulovou výšku. Při letu se budou cyklicky měřit úrovně tlaku. Tyto hodnoty tlaku se porovnají s první změřenou a dostane se tak rozdíl v pascalech. Ten se podělí deseti - změna výšky o 1 metr je vyjádřena změnou atmopsférického tlaku právě o 10Pa. Tento přepočet platí do stovek metrů nad mořem. Přičemž střední nadmořská výška České republiky je 450 m n. m. Tímto způsobem se dostane v okamžicích měření výška nad místem vzletu.

2. Praktické řešení

Pro komunikaci čidla s mikrokontrolérem byla zvolena sběrnice I^2C kdy odpadá nutnost výstupní data převádět a výsledek je reprezentován přímo sérií bitů které pak mohu rovnou zapisovat do paměti. Paměti se bohužel nedělají tak rychlé jako čidla. Jelikož je rychlost dána nejpomalejším zařízením na sběrnici, rychlost komunikace tak dosahuje maxima 400kHz.



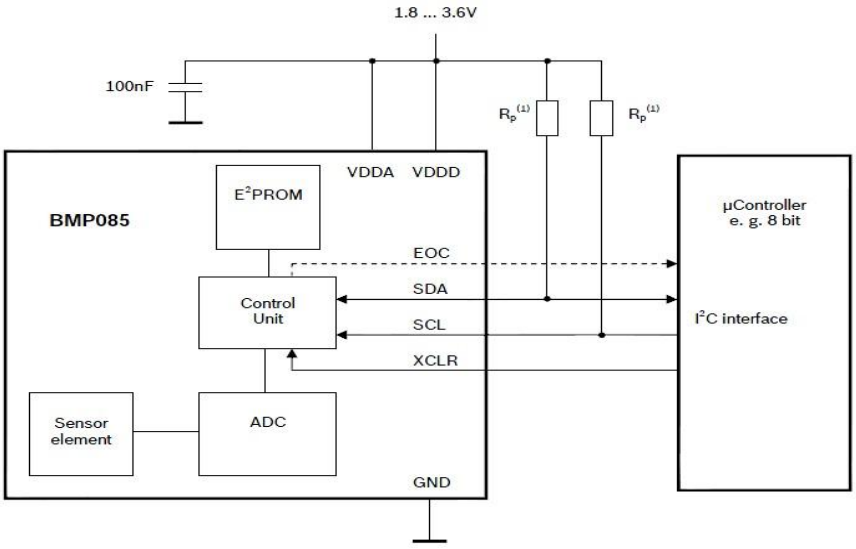
Obr 2.1 Blokové schéma zapojení

2.1. Tlakové čidlo

Pro tento úkol bylo vybráno čidlo tak aby bylo schopno změřit tlak v požadovaném rozmezí. Použiji-li hodnoty které se mohou vyskytnou na území ČR pak je to od 970hPa do 1055hPa. Druhým kritériem při výběru byla jeho rozlišovací schopnost. Tady je myslím rozumné požadovat rozlišení nejméně jednoho metru což odpovídá změně tlaku o 10Pa.

Zvolil jsem si čidlo BOSCH BMP085. Je založeno na piezo-rezistivní technologii. Jeho rozsah měření je od 300...1100hPa což naprosto dostačuje. Napájecí napětí je typicky 2,5V a max. 3,6V. Odebíraný proud závisí na zvoleném režimu. Čidlo je schopno pracovat celkem ve čtyřech přičemž každý má jiné proudové odběry, rychlosti a přesnost. V absolutní hodnotě je přesnost měřeného tlaku udávána do 1hPa.

Je schopno kromě měření tlaku i měření teploty a komunikuje po sběrnici I2C. S tím souvisí rozlišovací schopnost výstupních dat, která je udávána jako 1Pa a 0,1°C.



Obr 2.2 Typické zapojení čidla [3]

2.1.1. Čtení výsledků měření

Celé měření je nejprve nutno zahájit posláním startovací sekvence. Poté se musí počkat než proběhne konverze. Její čas se odvíjí od zvoleného módu, který ovlivňuje taktéž i rychlost opakování měření, proudový odběr a odchylky v měření.

Mode	Parameter *oversampling_setting*	Internal number of samples	Conversion time pressure max. [ms]	Avg. current @ 1 sample/s typ. [µA]	RMS noise typ. [hPa]	RMS noise typ. [m]
ultra low power	0	1	4.5	3	0.06	0.5
standard	1	2	7.5	5	0.05	0.4
high resolution	2	4	13.5	7	0.04	0.3
ultra high resolution	3	8	25.5	12	0.03	0.25

Tab 2.1 Použitelné módy měření [3]

Zvolil bych jako dostatečný "ultra low power" mimo jiné i kvůli minimálnímu proudu a dostačující rychlosti. Rušení v hodnotě 0,06hPa odpovídá asi půl metru. Jednotlivé módy se nastavují v proměnné oversampling_setting (osrs).
Změřené hodnoty se ještě kalibrují. V paměti E^2PROM je uloženo 11 kalibračních koeficientů individuálních pro každý senzor. Každý je reprezentován 16bity, uloženými od MSB. Před prvním měřením si tak master musí zjistit jejich hodnoty přečtením paměti senzoru.

Read calibration data from the E²PROM of the BMP085		bmp085_get_cal_param		
read out E²PROM registers, 16 bit, MSB first				
AC1 (0xAA, 0xAB)	(16 bit)	AC1 =	408	short
AC2 (0xAC, 0xAD)	(16 bit)	AC2 =	-72	short
AC3 (0xAE, 0xAF)	(16 bit)	AC3 =	-14383	short
AC4 (0xB0, 0xB1)	(16 bit)	AC4 =	32741	unsigned short
AC5 (0xB2, 0xB3)	(16 bit)	AC5 =	32757	unsigned short
AC6 (0xB4, 0xB5)	(16 bit)	AC6 =	23153	unsigned short
B1 (0xB6, 0xB7)	(16 bit)	B1 =	6190	short
B2 (0xB8, 0xB9)	(16 bit)	B2 =	4	short
MB (0xBa, 0xBB)	(16 bit)	MB =	-32767	short
MC (0xBC, 0xBD)	(16 bit)	MC =	-8711	short
MD (0xBE, 0xBF)	(16 bit)	MD =	2868	short

Obr 2.3 Kalibrační koeficienty [3]

Následuje samotné čtení výsledků měření. Výsledky jenž čidlo posílá jsou nekompenzované a je třeba je přepočítat a zjistit skutečné hodnoty. Každá čtená hodnota, ať se jedná o teplotu nebo tlak má šestnáct bitů.

calculate true temperature		bmp085_get_temperature		
$X1 = (UT - AC6) * AC5 / 2^{15}$	X1 =	4743		long
$X2 = MC * 2^{11} / (X1 + MD)$	X2 =	-2344		long
B5 = X1 + X2	B5 =	2399		long
$T = (B5 + 8) / 2^4$	**T =**	**150**	*temp in 0.1℃*	long

calculate true pressure		BMP085_calpressure		
B6 = B5 - 4000	B6 =	-1601		long
$X1 = (B2 * (B6 * B6 / 2^{12})) / 2^{11}$	X1 =	1		long
$X2 = AC2 * B6 / 2^{11}$	X2 =	56		long
X3 = X1 + X2	X3 =	57		long
B3 = ((AC1*4+X3) << oss + 2) / 4	B3 =	422		long
$X1 = AC3 * B6 / 2^{13}$	X1 =	2810		long
$X2 = (B1 * (B6 * B6 / 2^{12})) / 2^{16}$	X2 =	59		long
$X3 = ((X1 + X2) + 2) / 2^2$	X3 =	717		long
$B4 = AC4 * (unsigend long)(X3 + 32768) / 2^{15}$	B4 =	33457		unsigned long
B7 = ((unsigned long)UP - B3) * (50000 >> oss)	B7 =	1171050000		unsigned long
if (B7 < 0x80000000) { p = (B7 * 2) / B4 }	p =	70003		long
else { p = (B7 / B4) * 2 }				long
$X1 = (p / 2^8) * (p / 2^8)$	X1 =	74529		long
$X1 = (X1 * 3038) / 2^{16}$	X1 =	3454		long
$X2 = (-7357 * p) / 2^{16}$	X2 =	-7859		long
$p = p + (X1 + X2 + 3791) / 2^4$	**p =**	**69964**	*press. in Pa*	long

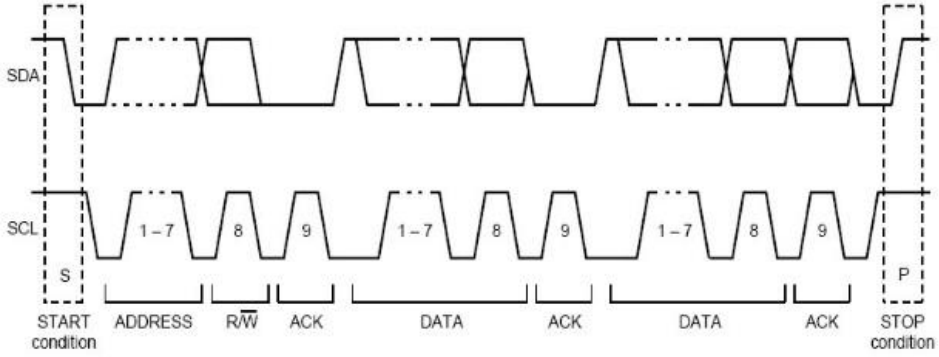
Obr 2.4 Algoritmus přepočtu na skutečnou teplotu a tlak [3]

Kalibrační koeficienty se čtou z paměti čidla tak, že se pošle adresa registru který chceme přečíst a po opakované startovací podmínce nám čidlo pošle dva bajty reprezentující daný koeficient.

2.1.2. Komunikace čidla po I2C

Samotné čidlo dokáže po této sběrnici komunikovat rychlostí až 3,4Mbit/sec při fast a high-speed modu. Vodiče SCA a SCL samozřejmě potřebují pull-up rezistory, typicky 4.7kOhm. Po této sběrnici se provádí veškerá komunikace s čidlem. Vodič SDA se používá pro přenos sériových dat. SCL slouží pro hodinový signál. Čidlo obsahuje i vstup XCLR kterým lze provést reset obvodu a inicializaci interních registrů a čítače pulsem trvajícím minimálně 1µs. Tento vstup se při nevyužívání nemusí připojovat a má vlastní pull-up rezistor. Před první komunikací po připojení napájení se musí vyčkat alespoň 10ms.
Komunikační protokol I^2C má přesně specifikované podmínky jak mají vypadat signály pro start, stop i pro přenos binárních dat. Startovací podmínka vypadá tak, že SCL je ve vysoké

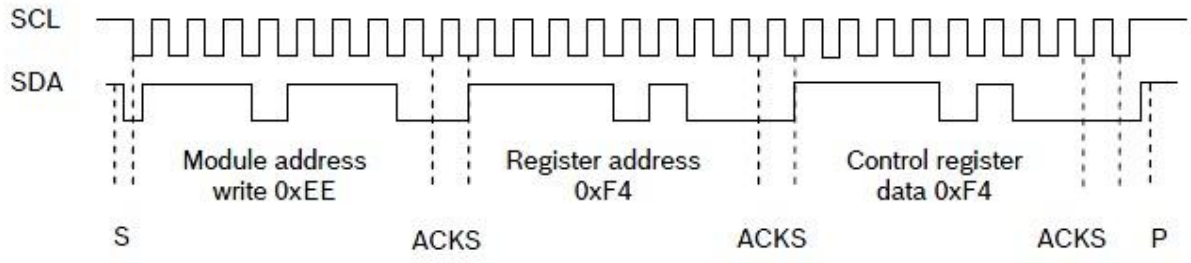
úrovni a signál SDA přejde z vysoké na nízkou úroveň neboli jde o sestupnou hranu. Následuje poslání adresy slave zařízení (7 bit), a bit R/W pro výběr zda-li se má číst a nebo zapisovat. Jakmile zařízení rozpozná, že se jedná o jeho adresu potvrdí tuto skutečnost stažením signálu SDA na nízkou úroveň.
Koncová podmínka se pozná nástupnou hranou signálu SDA při SCL ve vysoké hodnotě. Tyto podmínky jsou jediné, které se mění při SCL ve vysoké úrovni. Datové signály musí být v této době stabilní.



Obr 2.5 Komunikační protokol I^2C [3]

2.1.2.1. Začátek měření

Časový průběh komunikace s čidlem po staru měření teploty UT a tlaku UP je ukázán na obrázku níže. Po startovací podmínce pošle mikrokontrolér adresu zařízení, adresu registru a data pro kontrolní registr. Čidlo BMP085 posílá potvrzovací bit ACKS po každých 8 bitech, přijme-li je. Mikrokontrolér ukončí komunikaci po posledním potvrzovacím bitu od čidla.



Obr 2.7 Prvotní komunikace s čidlem [3]

Measurement	Control register value (register address 0xF4)	Max. conversion time [ms]
Temperature	0x2E	4.5
Pressure (osrs = 0)	0x34	4.5
Pressure (osrs = 1)	0x74	7.5
Pressure (osrs = 2)	0xB4	13.5
Pressure (osrs = 3)	0xF4	25.5

Adresy a hodnoty pro poslání kontrolním registrům sloužící k nastaveni osrs.

Tab 2.2 Nastavení osrs

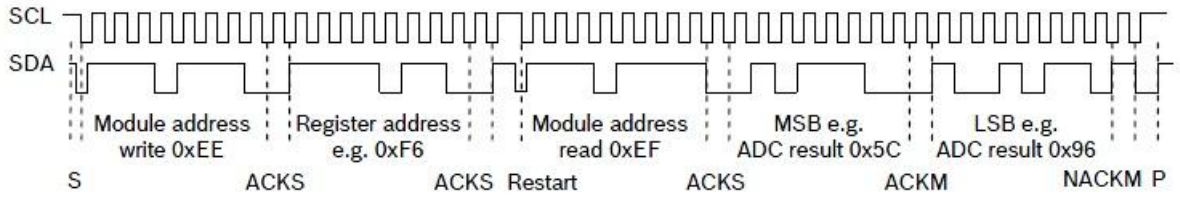
Časy uvedené v tabulce jsou pouze maximálními hodnotami. Nemusí se však čekat tak dlouho. Čidlo má pin EOC (end of conversion) pomocí něhož informuje o ukončení převodu. Informace se projeví logickou jedničkou v případě ukončení a logickou nulou je-li proces stále v běhu.

2.1.2.2. Čtení výsledků

Výsledná data pro teplotu jsou v bitovém slově o délce 16bitů, tlaková informace je taktéž šestnácti bitová a při použití přesnějšího módu může mít 19bitů.

Po zahájení komunikace master posílá adresu zařízení a adresu registru. Touto adresou se vybere registr z kterého se bude číst. Následuje restart následovaná opět adresou čidla tentokrát však je vybráno čtení. Čidlo poté pošle prvních 8 bitů MSB, což tentokrát potvrzuje master a následuje 8 bitů LSB a ukončení komunikace.

Při vybraném ultra high resolution se ještě čte XLSB registr s adresou 0xF8, který výsledek měření tlaku rozšiřuje z 16 na 19bitů.

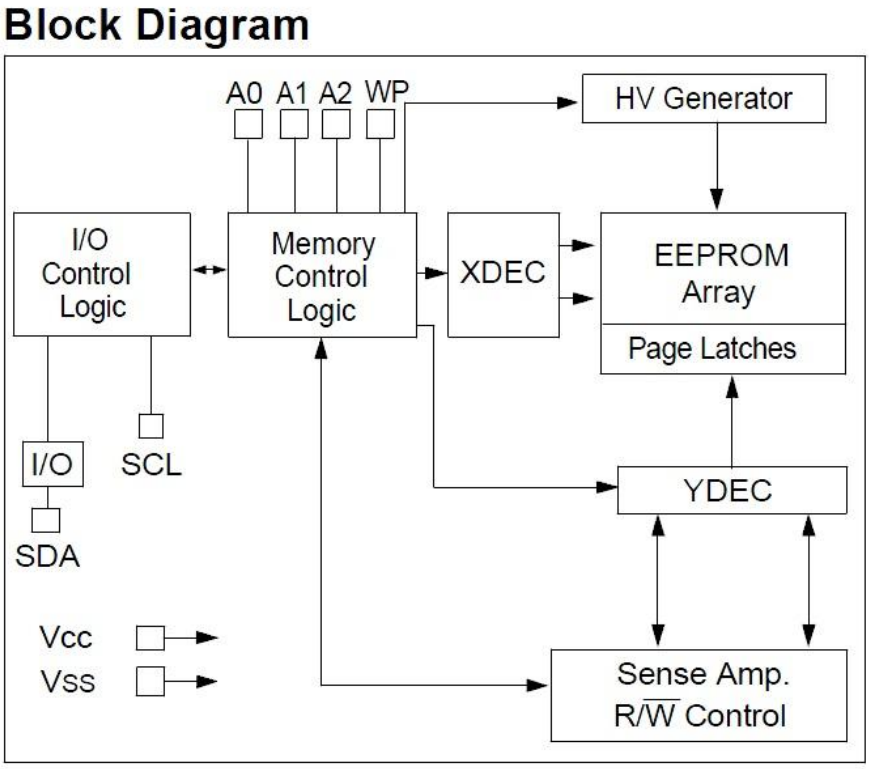


Obr 2.8. Čtení výsledků měření [3]

2.2. EEPROM

Vybral jsem paměť o velikosti 32kbit do které lze psát i formou stránek (32x8bitů). Výhodou je, že jí lze ovládat pomocí I^2C a je tak možno po jedné sběrnici komunikovat jak s pamětí tak s čidlem. Krom pinů pro I2C má ještě ochranu proti zápisu (WP), která se dá nastavit vysokou úrovní. Piny označené jako A0 až A2 se využijí při více těchto pamětech zapojených na jedné sběrnici. Jejich hodnota je zahrnuta v adrese paměti. Dalšími parametry paměti jsou:

- Velikost paměti: 32Kbit = 4kbyt
- Vnitřní konfigurace: 32Byt stránka
- Možnosti komunikace: I2C, serial
- Maximální frekvence: 400kHz
- Napájecí napětí: 2.5 to 5.5V
- Typ pouzdra: SOIC8
- Počet pinů: 8
- Provozní rozsah teplot: -40°C to +85°C

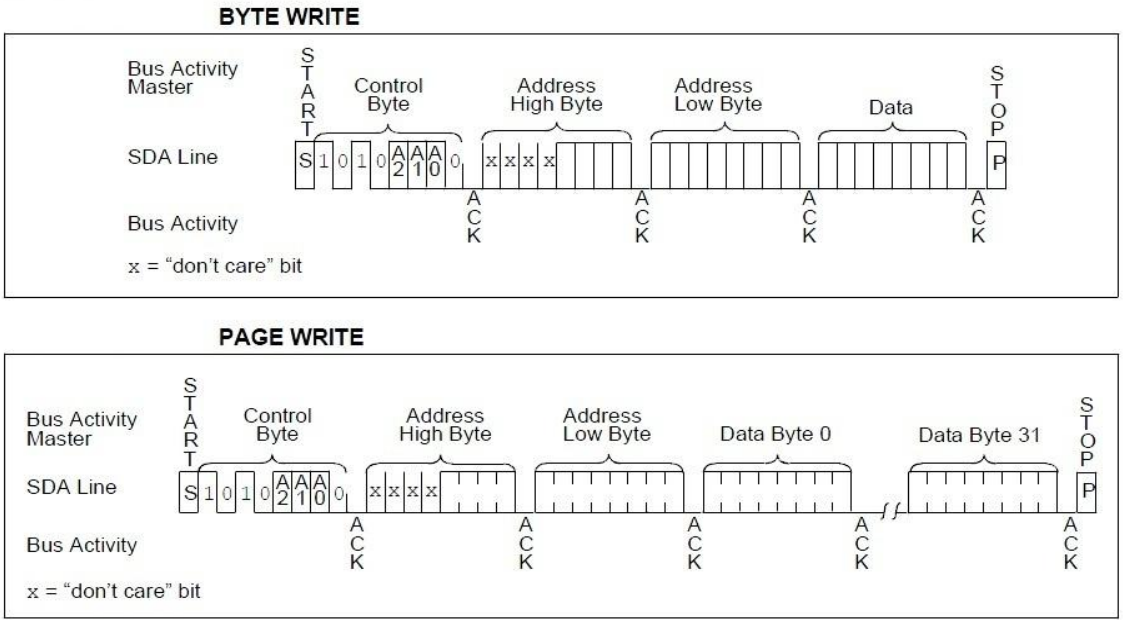


Obr 2.9 Blokové schéma paměti [5]

2.2.1. Komunikace po I2C

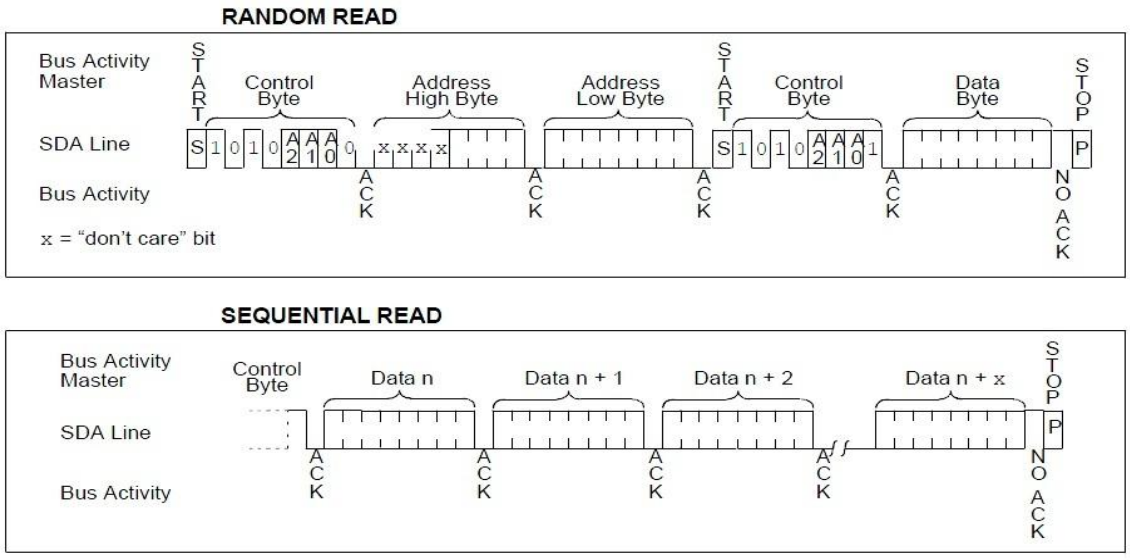
Zápis se tedy zahájí start. podmínkou. Adresa paměti se skládá z pevně daných prvních 4 bitů, 3 bitů nastavených piny a bitu R/W. Adresa má tedy tvar 0xA0 pro zápis a 0xA1 pro čtení. Poté co slave potvrdí adresu posílá se vyšší a nižší část adresy kam se budou ukládat data. Tu si paměť uloží do čitače a přijímá data která má na tuto adresu ukládat. Po skončení zápisu opět potvrdí načež můžeme komunikaci ukončit.
Vnitřní čítač do nějž byla uložena adresa buňky pro uložení dat se nemaže, ale naopak se automaticky inkrementuje a ukazuje na další buňku čehož lze využít hlavně při čtení kdy se nemusí posílat adresa buňky.

Nemilou vlastností může být, že pokud je aktivní WP pak paměť nedokáže uložit data, přesto však posílá ACK.
Další vlastností je, že během zapisování dat paměť nereaguje a neodpoví tedy na volání potvrzením ACK. Toho se dá využít pokud by chtěl člověk co nejrychleji pokračovat dalším zápisem. Jak je ukázáno v datasheetu stačí poté co paměť začne zapisovat posílat jí dotazy tak dlouho než pošle potvrzení.



Obr 2.10 Komunikace s pamětí při zápisu [5]

Čtení se taktéž dá provést více způsoby. Nejvhodnější je "Random Read" kdy postup vypadá jako při zápisu. Nejdříve se pošle adresa zařízení s tím, že chceme zapisovat a pak pošleme adresu kterou chceme z paměti získat. Poté se pošle adresa zařízení znovu, tentokrát už ale s nastaveným čtením a přečte se slovo z paměti.

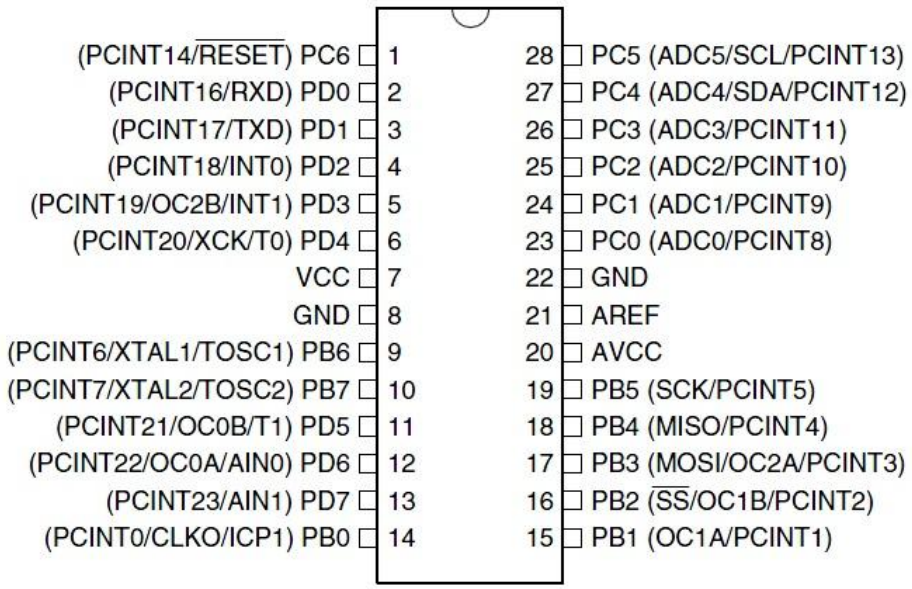


Obr 2.11 Komunikace s pamětí při čtení [5]

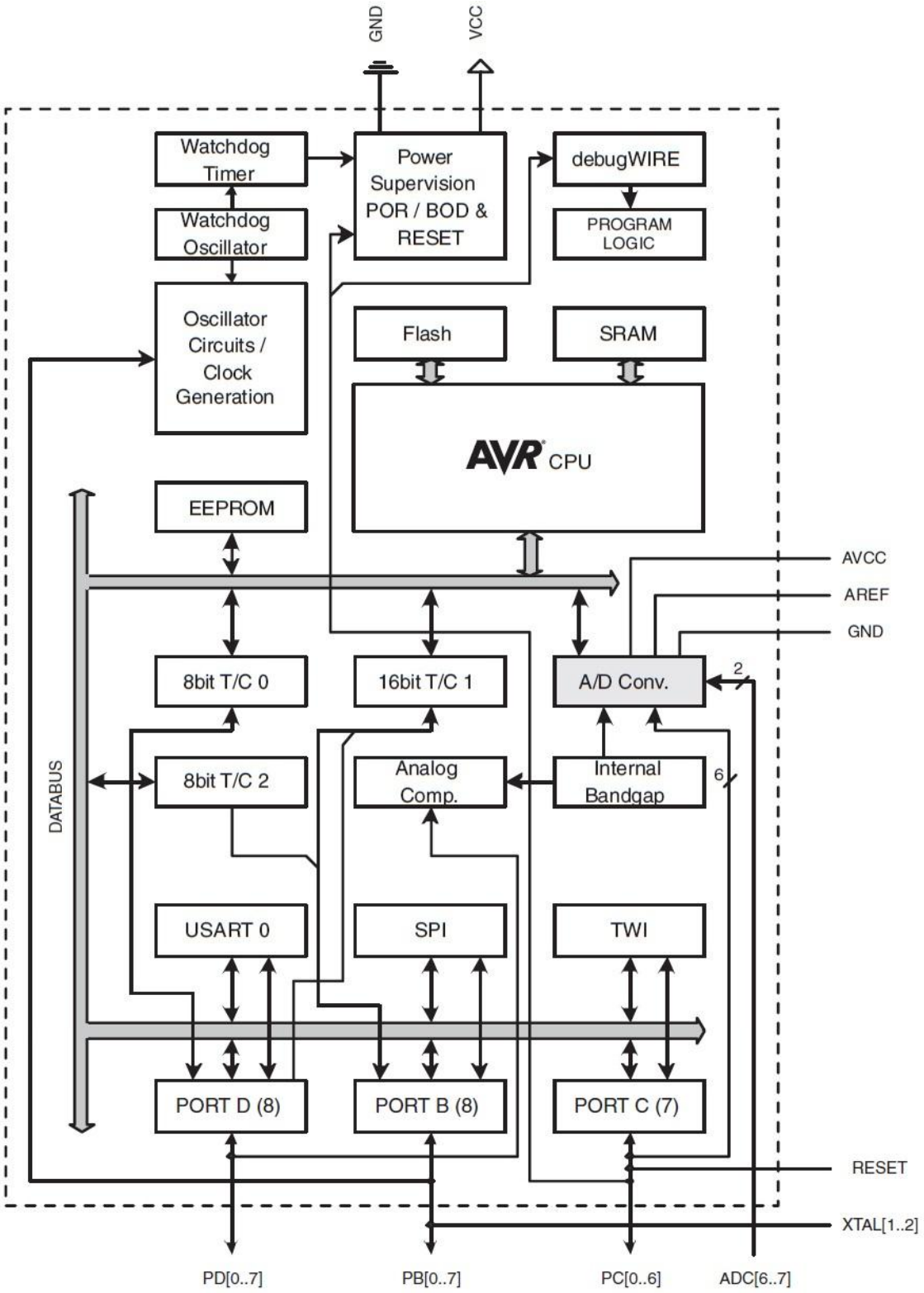
2.3. Mikrokontrolér

K řízení celého zapojení jsem zvolil mikrokontrolér (MCU) ATmega16. Hlavním důvodem který mě vedl k výběru právě tohoto typu byla skutečnost, že jsem se s ním seznámil v rámci výuky předmětu Bmpt v zimním semestru. Jedná se o 8bitový mikrokontrolér a některé z jeho vlastností jsou vypsány níže:

- počet I/O pinů: 23
- Velikost programové paměti: 16 KB
- Velikost paměti EEprom: 512Byte
- Velikost paměti RAM: 1KB
- CPU rychost: 20MHz
- Typ oscilátoru: External, Internal
- Počet časovaču: 3
- Peripherals: ADC, Comparator, PWM, Timer
- Počet PWM kanálů: 6
- Napájecí napětí: 2.7V to 5.5V
- Provozní rozsah teplot: -40°C to +85°C
- počet pinů: 32
- Možnosti komunikace: I2C, Serial, SPI, USART, 2-Wire
- Velikost paměti programu: 16KB
- Pouzdro DIL28



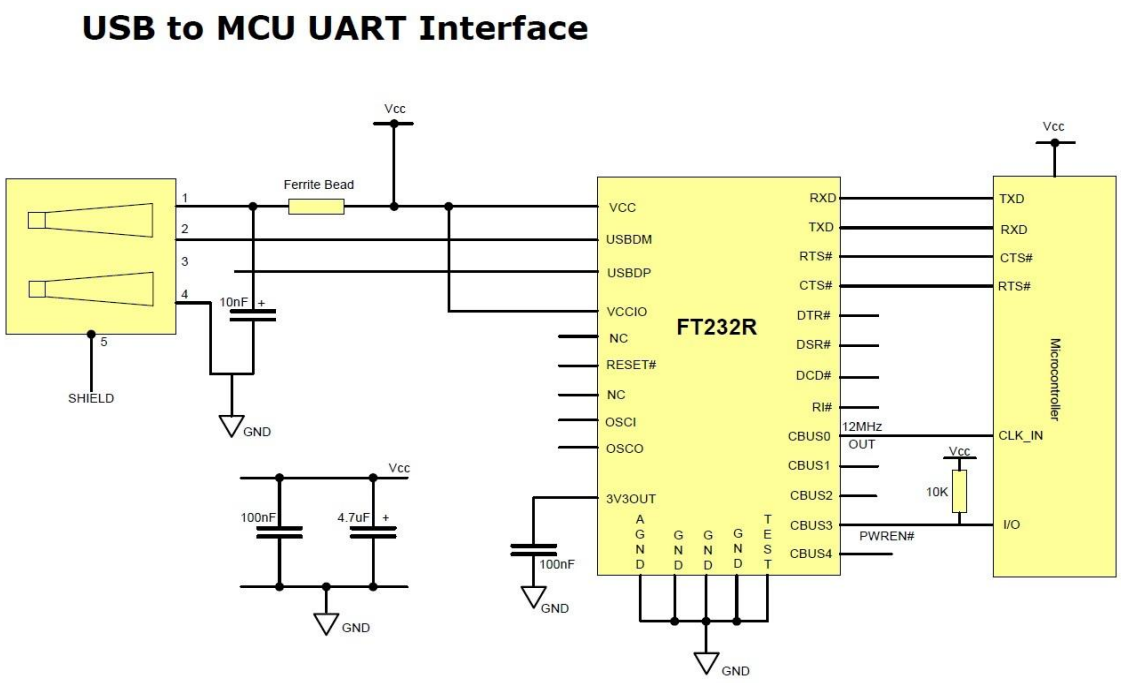
Obr 2.11 Pouzdro DIL28mikrokontroléru ATmega16 s vývody [4]



Obr 2.12 Vnitřní blokové schéma mikrokontroléru ATmega16 [4]

2.4. Převodník

Poté co bude zařízení připojeno k PC generuje převodník na svém pinu nízkou úroveň čímž dá signál mikrokontroléru aby začal posílat naměřená dat. Obvod má za úkol data která mu pošle MCU převést z komunikace po rozhraní UART na rozhraní usb. Díky tomu se zařízení snadno spojí s počítačem kterému změřená data předá. Převodník jsem zvolil FT232R.

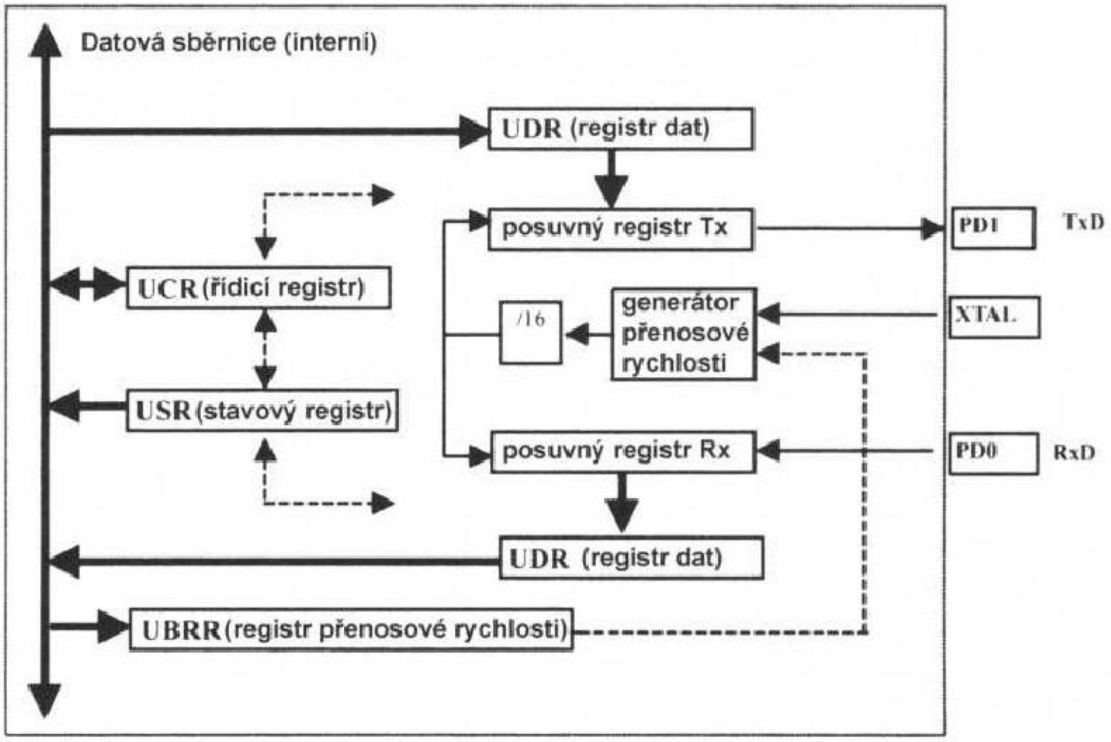


Obr 2.13 Příklad zapojení převodníku (z datasheetu [6])

Funkce PWREN# je přednastavena od výrobce a vyvedena na výstup pinu CBUS3. Toto nastavení se dá programově změnit. Její funkcí je: na pinu CBUS3 je nízká úroveň po připojení USB konektoru. Je-li kabel odpojen pak je na tomto výstupu vysoká úroveň. Toho se dá využít k probuzení zařízení při využívání sleep módu.

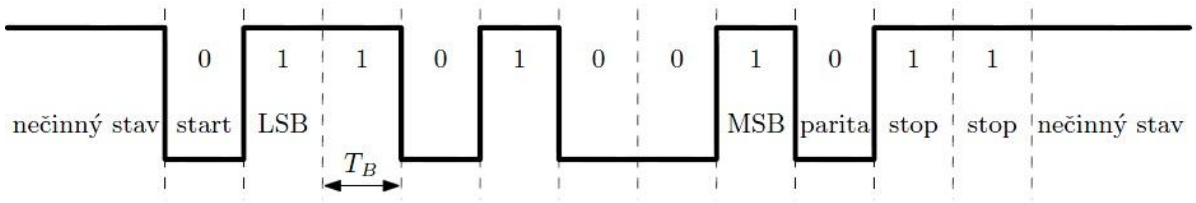
2.4.1. UART

Neboli univerzální asynchronní přijímač a vysílač. Základní úlohou jednotky UART je vyslat datové slovo (bajt) převzaté od CPU. To se přivádí do registru UDR. Ten informuje o tom zda je naplněn příznakem ve stavovém registru. Taktéž můžeme v tomto registru zjistit, že jsme naopak přijmuli bajt, který se opět uloží do registru UDR. Následně je jednotka UART samostatně sériově (bit po bitu) vyšle na linku TxD(Transmit), popřípadě načítá bajt z linky RxD (Receive). Protože na přijímač se přivádí jen datová linka a nikoliv bitový hodinový signál, hovoří se o asynchronním přenosu. Kromě již zmíněných přerušení informuje jednotka i o tom, že vysílaný bajt je už v registru Tx a můžeme tak opět posílat data do UDR. Je možno se i dozvědět zda došel vysílaný bajt vpořádku a přijímač identifikoval Stop bit nebo zda nastala chyba rámce.



Obr 2.14 Zjednodušené znázornění jednotky UART [6]

Sériový přenos dat začíná bitem s nejnižší váhou (LSB - Least Significant Bit). Každý bit je na lince udržován přesně určenou dobu, až se nakonec vydá poslední bit. K tomu dostává UART interní hodinový signál pro posuv bitů z mikrokontroléru. Každý bajt zarámován signálem nazývaným startbit na začátku, který má vždy stav LOW a jedním nebo dvěma stopbity na konci bajtu, které mají vždy stav HIGH. Nevysílá-li se žádný bit, je linka v klidovém stavu na úrovni HIGH.

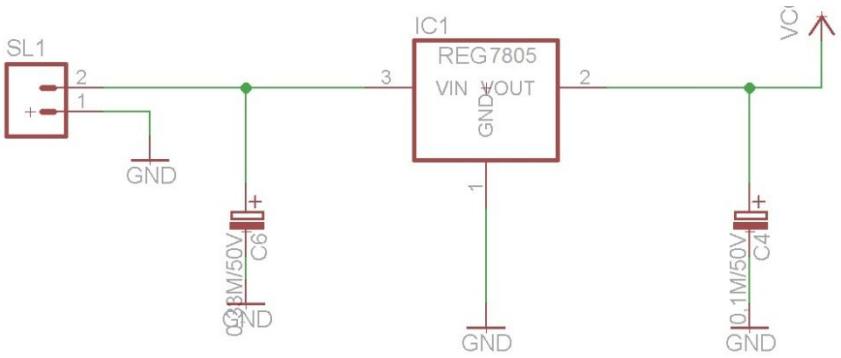


Obr 2.15 Tvar bajtu pro asynchronní přenos dat [1]

U jednotky UART rodiny AVR vzorkuje přijímač každý bit 16x, tedy 16násobkem kmitočtu nastavené přenosové rychlosti (baudrate). Pokud je přijímací linka na úrovni HIGH (klidový stav), rozpozná se první vzorkovací hodnota LOW jako okamžik hrany startbitu. V nejhorším případě s přesností 1/16 doby bitu.

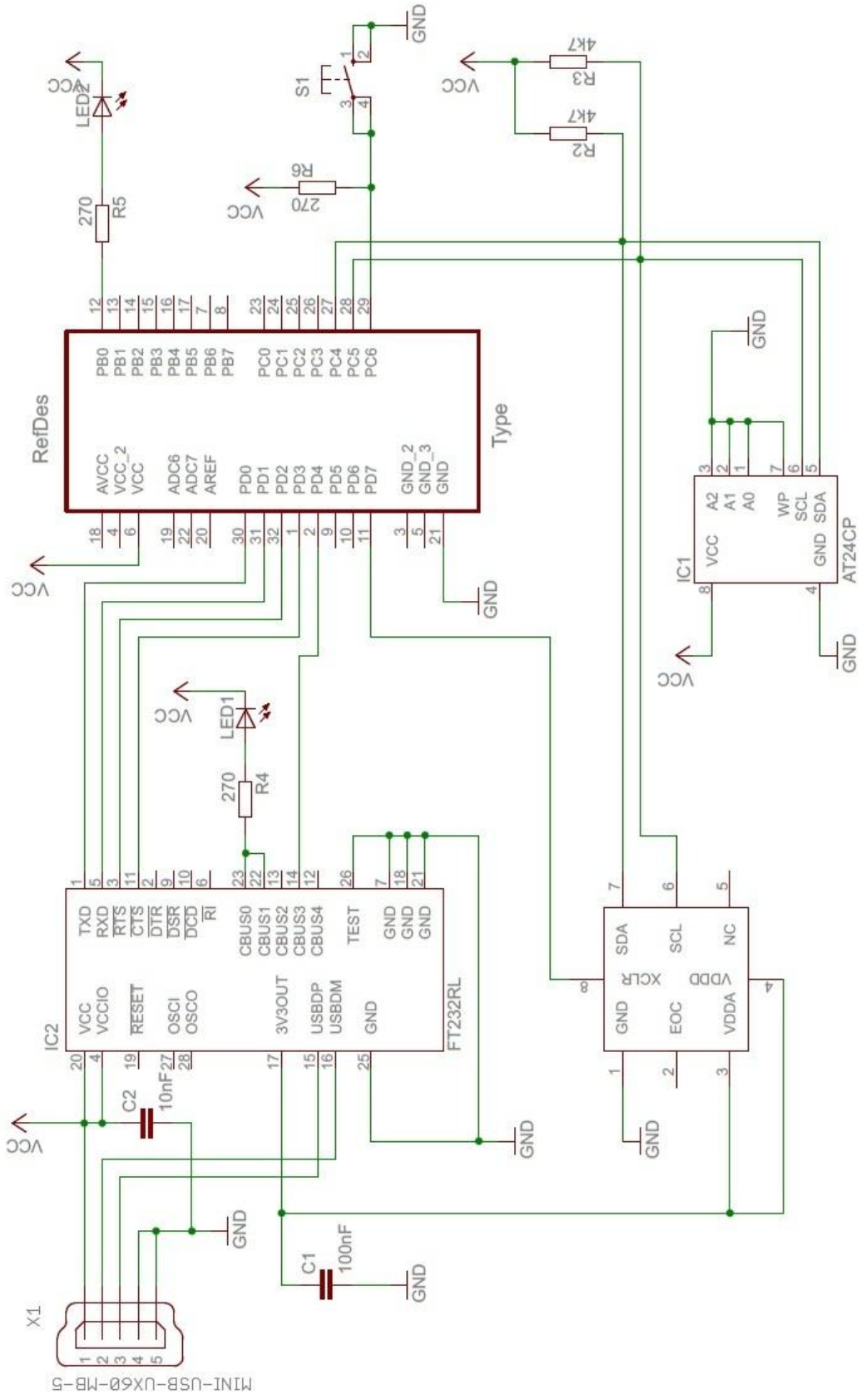
Baud rate: Udává počet změn stavu přenášeného signálu za sekundu a nazývá se také kroková rychlost nebo bitový takt. Baudrate se uvádí v jednotkách Baud.
Bitrate: Udává počet přenesených bitů za sekundu a uvádí se v jednotkách bit/s nebo bps.

3. Zapojeni



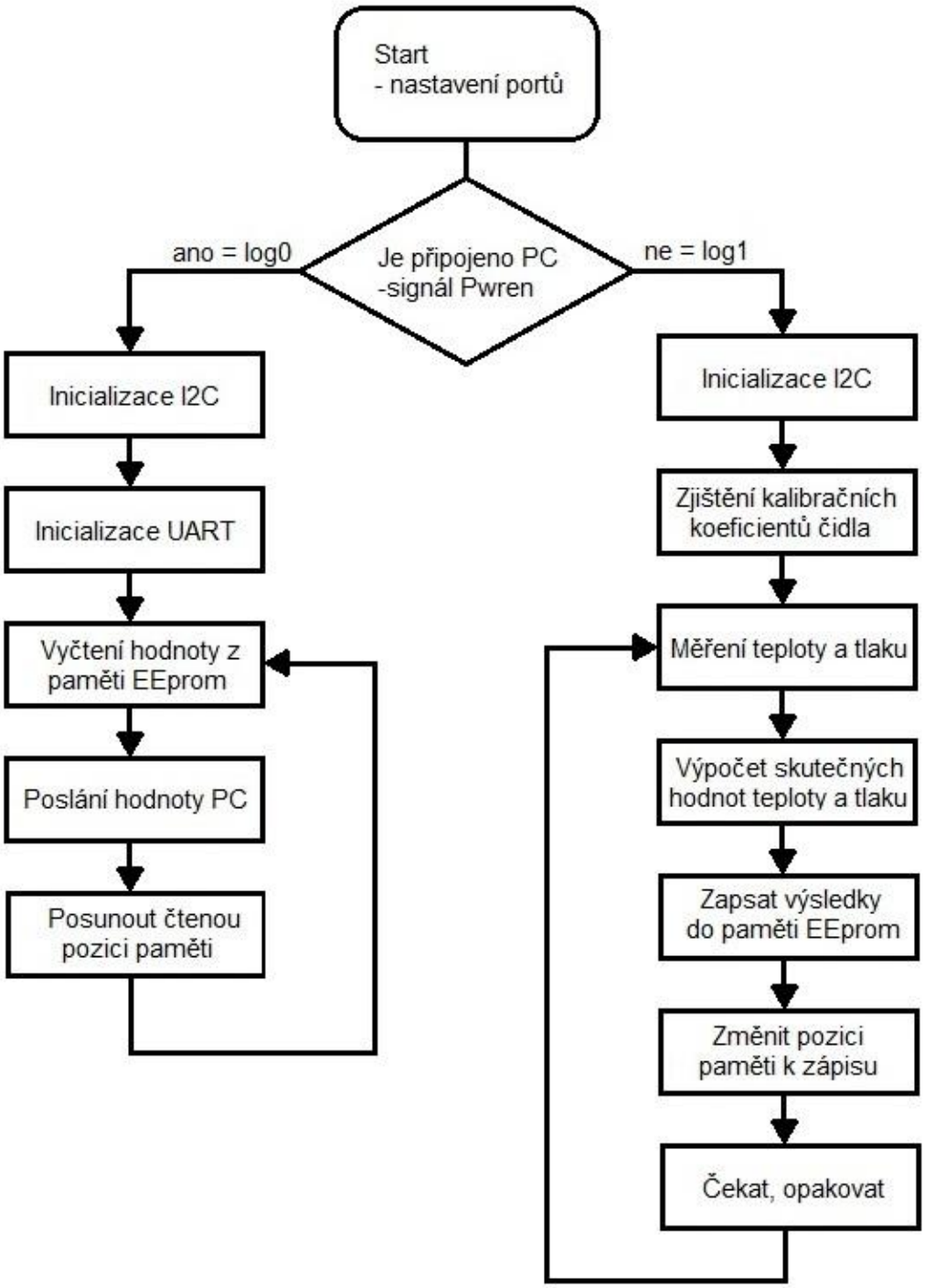
Obr 3.1 Zapojení stabilizátoru napětí

Celé zapojení bylo navrženo v programu Eagle. Zařízení má pracovat na palubě modelu letadla. Z toho důvodu je třeba ho napájet baterií a její napětí převádět na použitelných 5V. Pro napájení čidla jež pracuje s maximálním napětím pouze 3,6V bylo využito výstupu převodníku FT232R, který má na 17. pinu výstup 3,3V.
Mikrokontolér komunikuje jak s čidlem tak s pamětí po sběrnici I2C, kterou představují vodiče SDA a SCL. Ty potřebují ke správné činnosti pull-up rezistory, typicky s hodnotou 4,7kΩ. Čidlo Bmp085 je dále s MCU spojeno vodičem xclr jenž je možno využít k resetu čidla. MCU má na prvním pinu (PC6 - RESET) připojeno resetovací tlačítko a první bit portu B je spojen s LED diodou indikující činnost zařízení. S převodníkem FT232 komunikuje přes datové vodiče Rx (přijímá),Tx (vysílá) a RTS pin převodníku svou logickou nulou zděluje, že je schopen přijímat další data. FT232 má taktéž diodu informující o činnosti zařízení. K usb je připojen vodiči USBDM, USBDP pro posílání dat a vodiči pro napájení 5V.



Obr 3.2 Zapojení celého zařízení

4. Program



Obr 4.1 Vývojový diagram zařízení

4.1 Popis programu

Program pro MCU začíná funkcí main.

```
int main ( void )                    // kod hlavni funkce
{
    DDRB = (1<<PB0);              /* nastavi PB0 jako vystup */
    PORTB = (1<<PB0);             /* LED on */
        _delay_ms(100);
    PORTB = (0<<PB0);             /* LED off */

    DDRC = (1<<PC3) ;             //xclr - pin3;
    PORTC = (0<<PC3);             // Na vystupu log0 - xclr off

    DDRD = (0<<PC4) ;             //pwren signal - pin4 +cts-2i; rts-3o;
    PORTD = (0<<PC4);


    if ( bit_is_clear( PIND , 4)) {     //budu rozhodovat jestli se ma merit nebo posilat data
        usb ()      ;                 // Execute these statements if TRUE
    }
    else  {
        mereni ()   ;               // Execute these statements if FALSE

    }


    sei () ;                            // globalni povoleni vsech preruseni
    while ( 1 ) ;                        // nekonecna smycka
    return ( 0 ) ;

}
```

Obr 4.2 Funkce main

Jako první se nastaví PortB ke kterému je připojena Led dioda. Na portu C je pinem 3 připojeno čidlo s výstupem xclr, který lze využít pro jeho reset. K portu D je přiveden signál PWREN z převodníku FT232. Je-li k převodníku připojeno usb má tento výstup nízkou úroveň. To se změní s jeho odpojením, kdy má naopak vysokou úroveň. Toho využívá na dalších řádcích podmínka if, která rozhoduje zda se budou posílat data z paměti(pokud je usb připojeno) nebo zda se měří tlak čidlem.

4.1.1 Část měření

V této části se komunikuje s čidle a pamětí po sběrnici I2C. Nejdříve se musí kounikace inicializovat. Toho se dosáhne nastavením několika registrů. Jelikož jde o synchronní sběrnici je zde hodinový signál. Ten vysílá MCU jako master. U tohoto zapojenínení použit externí, ale interní oscilátor mikrokontroléru. Jeho frekvenci lze nastavit v rozsahu od 7,3 - 8,1MHz s přesností na 2%. Hodnotu na kterou se má nastavit uchovává registr OSCAL, kde při nulové hodnotě vybírá nejnižší možnou z rozsahu. Od výrobce je oscilátor už přednastaven na hodnotě 8MHz.

```
/* I2C clock in Hz */
#define SCL_CLOCK  100000L
#define F_CPU 8000000UL


void i2c_init(void)
{
  /* initialize TWI clock: 100 kHz clock, TWPS = 0 => prescaler = 1 */

  TWSR = 0;                         /* no prescaler */
  TWBR = ((F_CPU/SCL_CLOCK)-16)/2;  /* must be > 10 for stable operation (=32) */

}
```

Obr 4.3 Inicializace I2C

Při inicializaci I2C je třeba nastavit rychlost hodinového signálu. K tomu se využije právě i frekvence oscilátoru mikrokontroléru. Rychlost hodinového signálu přenášeného vodičem SCL je, jak je vidět na obrázku výše, nastavena na frekvenci 100kHz. Pro výpočet konečné hodnoty se využije i velikost registru TWSR, který je ve výpočtu jako "PrescalerValue". Výsledná hodnota se nastavuje v registru TWBR. Frekvence hodinového signálu na sběrnici I2C se vypočítá podle následujícího vztahu [4] :

$$SCLfrequency = \frac{CPU\ Clock\ frequency}{16+2(TWBR)\times(Prescaler\ \ Value\ )} \qquad (4.1)$$

Nyní můžeme zahájit komunikaci s senzorem. Bmp085, které jsem si vybral, obsahuje celkem jedenáct kalibračních koeficientů individuálních pro každý senzor. Ty se využijí při výpočtu naměřené skutečné hodnoty teploty a tlaku. Proto je potřeba je před samotným měřením přečíst z paměti senzoru.

```
int mereniAC1 (void)
{

    twi_start();                // TWI start init
    twi_address_w(0xEE);        // vyslani adresy SLA(+W)
        data = 0xAA;            //reg adresa 0xAA
    twi_write(data);            //reg adresa 0xAA

    twi_start();

    twi_address_w(0xEF);        // vyslani adresy SLA(+R)
    Msbx = twi_readA();         //cteni dat o AC1 - 2 byte
    Lsbx = twi_readNA();
    twi_stop();

    return AC1 = (Msbx<<8) + Lsbx;          // posklada lsb a msb cisla AC1
}
```

Obr 4.4 Příklad vyčítání koeficientu AC1

Na obrázku obrázku 4.4 je vidět, jak probíhá komunikace s senzorem. Nejdříve se posílá jeho adresa s příznakem zápisu a poté adresa místa paměti kde se daný koeficient nachází. Následuje opětovná startovací podmínka a adresa s příznakem čtení. Poté Senzor posílá data od MSB po LSB. Celkem 16 bitů. To je důvod proč je třeba je v posledním řádku posunout. Senzor totiž pošle první bajt a MCU mu odpoví posláním ACK. Následuje druhý bajt, který však už MCU nepotvrzuje, ale ukončuje komunikaci. V tomto okamžiku je jedne šestnácti bitový koeficient rozdělen do dvou proměnných po osmi bitech. Takovýto postup je třeba opakovat pro všech jedenáct koeficientů. Dalším krokem programu je samotné měření. Pro správný výpočet skutečných hodnot tlaku se používá aktuální hodnota teploty. Proto je třeba ji taktéž zjistit. Na obrázku 4.5 je popsána komunikace při měření tlaku. U teploty se popsaný postup liší jen v detailu, do registru 0xF4 se neposílá hodnota 0x34, ale 0x2E. Změnou této hodnoty lze i přenastavit přesnost měření (osrs) čidla jak je ukázáno v Tab 2.2.

```
int mereniUP (void)
{
    i2c_init();
    twi_start();                    // TWI start init
    twi_address_w(0xEE);            // vyslani adresy SLA(+W)
        data = 0xF4;
    twi_write(data);                // zapisujeme do reg 0xF4
        data = 0x34;
    twi_write(data);                // merime tlak s osrs=0 (34)
    twi_stop();                     // ukoncovaci podminka


         _delay_ms(5);

    twi_start();                    // TWI start init
    twi_address_w(0xEE);            // vyslani adresy SLA(+W)
        data = 0xF6;                //reg adresa 0xF6
    twi_write(data);                //reg adresa 0xF6

    twi_start();

    twi_address_w(0xEF);            // vyslani adresy SLA(+R)
    MsbUP = twi_readA();                //cteni dat o tlaku - 2 byte
    LsbUP = twi_readNA();
    twi_stop();

    return UP = (MsbUP<<8) + LsbUP;          // posklada lsb a msb cisla UT
}
```

Obr 4.5 Jak měřím tlak UP

Při měření tlaku se nejdříve zahájí komunikace a posílá adresa s příznakem zápisu. Poté se posílá adresa registru 0xF4 do kterého se zapíše další poslaná hodnota 0x34. Ta už senzoru říká, že má měřit tlak přičemž se tímto nastaví hodnota oversampling_setting(osrs) jako nulová. V takovém případě podle datasheetu je maximální doba měření 4.5ms. Kdyby byl připojen pin End of conversion nemusel bych čekat pevně danou dobu, ale vysoká úroveň tohoto pinu by informovala, že už lze přečíst výsledek. Takto je však třeba počkat 5ms. Posílá se znovu startovací podmínka a adresa s příznakem zápisu. Následuje adresa registru 0xF6 který obsahuje MSB výsledku. Teď už jen znovu poslat startovací podmínku a adresu s příznakem čtení a přečíst výsledek měření od MSB po LSB. Komunikaci ukončíme a stejně jako při čtení koeficintů je třeba spojit získané hodnoty do jediné veličiny UP.
Takto naměřené hodnoty se musí přepočítat na skutečnou hodnotu tlaku (teploty). Tento výpočet jsem převzal přímo z datasheetu čidla a je ukázán na obrázku 2.4.
Nyní se přenáší naměřený údaj do paměti EEprom opět přes I2C. Jelikož jsme z měření dostali 16bit hodnotu s kterou jsme pak dále přepočítali, tak ve výsledku je tlak vyjádřen 32bitovou proměnnou. Do paměti však lze zapisovat jen po osmi bitech a proto každou naměřenou hodnotu je potřeba rozdělit na 4 části - po bajtech.

```
a = P;
b = (char) a;
    a = a>>8;
c = (char) a;
    a = a>>8;
d = (char) a;
    a = a>>8;
e = (char) a;

doPameti(i,b);i++;
doPameti(i,c);i++;
doPameti(i,d);i++;
doPameti(i,e);
```

Obr 4.6 Dělení naměřené hodnoty a její poslání do paměti

Každý tento bajt postupně zapíši do paměti a k tomu jsem si vytvořil funkci *doPaměti*. Její vstupní parametr adres obsahuje adresu buňky v paměti kam se naměřená hodnota uloží.

```
void doPameti (int adres, char vysledek)
{
    char aa;
    char bb;
    aa =(char) adres;
        adres = adres>>8;
    bb =(char) adres;

    twi_start();                // TWI start init
    twi_address_w(0xA0);        // vyslani adresy pameti SLA(+W)
        data = bb;              //adresa bunky pameti
    twi_write(data);            //zapiseme adresu bunky
        data = aa;              //adresa bunky pameti
    twi_write(data);            //zapiseme adresu bunky

        data = vysledek;        //namereny vysledek
    twi_write(data);            //zapiseme vysledek
    twi_stop();

}
```

Obr 4.7 Posílání bajtů do paměti EEprom po I2C

Jako první krok ve funkci doPameti se rozdělí adresa na dvě 8bitové části. Jak již bylo uvedeno adresa má celkem 12bitů, čtyři nejvýznamější z 16 se neuvažují. Poté se posílá adresa paměti s příznakem zápisu a následuje část adresy paměťové buňky, která je v proměnné "bb" a pak nižší část adresy z proměnné "aa". Nakonec se posílá 8bitů dat. Takto pošlu všechny čtyři části jednoho měření a poskládám je v paměti za sebe. Celé toto měření se opakuje v cyklu for který začíná s parametrem i = 0. Ten je zároveň využit jako adresa buňky paměti (vstupní parametr adres). Každým měřením se cyklus for posune o 4 hodnoty "i" (neboli o 4 buňky paměti). Celý cyklus for končí na hodnotě i = 0b0000111111111111 která představuje poslední možnou pozici v paměti EEprom.

4.1.1 Část posílání dat na usb

Jako první je potřeba nastavit komunikaci po rozhraní UART. K tomu slouží několik registrů. Prvním z nich je UBRR který nastavuje baud rate, neboli přenosovou rychlost na lince. Ta se stejně jako u I2C počítá podle rychlosti krystalu.

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRRn Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$

Obr 4.8 Nastavení rychlosti Baud rate [bps] [4]

Registr je fyzicky rozdělen na dvě části, přičemž 4 nejvyšší bity jsou připraveny pro kompatibilitu s budoucími vyššími rychlostmi a musí být nastaveny jako nulové. Datasheet

říká, že při frekvenci oscilátoru 8MHz je třeba nastabit UBRR na hodnotu 51 pro baud rate = 9600bps.

USART Control and Status Register n B – UCSRnB

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB

Obr 4.9 Registr UCSRnB

- bit6 - TXCIE - pokud nastavíme tento bit na jedna, bude se po dokončení vysílání generovat přerušení v podobě nastaveného bitu TXC v UCSRnA
- bit3 - TXEN - nastavením tohoto bitu na jedna se povolí vysílání přes UART
- bit2 - UCSZ - spolu s UCSZn1:0 v UCSRnC nastavuje jak velký rámec dat se posílá

USART Control and Status Register n C – UCSRnC

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC

obr 4.10 Registr UCSRnC

- bit7..6 - UMSEL - vybírá mód komunikace; v programu použit asynchronní UART (00)
- bit5..4 - UPM - tyto bity nastavují jaká se používá parita; při nevyužití se nastavují jako nulové
- bit3 - USBS - tento bit specifikuje kolik se použije stop-bitů; program používá jeden (0)
- bit2..1 - UCSZ - spolu s UCSZn2 v UCSRnB nastavují jak velký rámec dat se posílá; v tomto programu nastaveno na 8 bitů (011)

```
ISR(INT1_vect)
{                                   // obsluha externiho preruseni INT1
    _delay_ms(10);
    return;                         // kod obsluhy
}

void usb (void)                     //program pro posilani dat po usb
{
    int k=0;
    char b;          //8bit=1bajt

        /* poslat do FT232 - baud9600, 8bitu, 1 stop, zadna parita*/

    UBRR0H = 0;
    UBRR0L = 51;
    UCSR0B = (1<<TXEN0)|(1<<TXCIE0)|(0<<UCSZ02);
    UCSR0C = (0<<UPM01)|(0<<UPM00)|(0<<USBS0)|(1<<UCSZ01)|(1<<UCSZ00)|(0<<UMSEL01)|(0<<UMSEL00);

    while(k==0)
    {
        b = zPameti(j);j++;
        UDR0 = b;       // - nastane přerušení INT1 značící že byl bajt doručen


        if(j>0b0000111111111111)
            j=0;

    }
}
```

obr 4.11 Posílání změřených údajů na usb

Obrázek obrázek 4.11 ukazuje, že posílání dat spočívá v přenesení našich dat do datového registru UDR a jednotka UART už se postará aby se vše poslalo. takto se bude po každém

poslaném bajtu zvyšovat parametr j, který reprezentuje adresu paměti. Je omezen podmínkou if, která zkoumá zda už nepřekračujeme kapacitu paměti, případně začneme od začátku. Přerušení INT1 je připojeno k výstupu převodníku FT232 a jeho pinu RTS. Ten mění svou úroveň podle toho, zda je schopen přijnout další data. Pokud není změní svůj stav z log0 na log1. V rámci obsluhy přerušení program počká a opět se vrátí k posílání dalších dat.

X. Závěr

Cílem této Bakalářské práce bylo vytvořit zařízení zaznamenávající průběh letu RC modelu. Ke zjištění v jaké výšce se zařízení nachází se využívá tlakové čidlo přičemž samotná výška se určí z rozdílu tlaků. Zařízení jsem navrhl a sestavil. Při případném použití by se jistě dalo ještě zmenšit. Napsal jsem i program pro měření i posílání výsledků do PC. Nepodařilo se mi však již nahrát tento program a ověřit tak jeho funkčnost.

5. Použitá literatura

[1] Frýza, Tomáš. Šebesta, Jiří. Fedra, Zbyněk. Mikroprocesorová technika a embedded systémy. Počítačová cvičení. Elektronické skriptum. VUT v Brně, Ústav radioelektroniky.

[2] Burkhard, Mann. C pro mikrokontroléry. Praha: BEN, 2003.

[3] BMP085 Datasheet [online]. Dostupné na www: <http://www.sparkfun.com/datasheets/Components/General/BST-BMP085-DS000-05.pdf>

[4] ATmega16 Datasheet [online]. Dostupné na www: <http://www.btinet.com/Faq.html>

[5] 24LC32AT-I/SN Datasheet [online]. Dostupné na www: <http://ww1.microchip.com/downloads/en/DeviceDoc/21713M.pdf>

[6] FT232R Datasheet [online]. Dostupné na www: <http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf>

[7] 7805 Datasheet [online]. Dostupné na www: <http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00000444.pdf>

6. Seznam zkratek

MCU	.....	Mikrokontroler
MSB	.....	Nejvýznamnější bit (Most Significant Bit)
LSB	.....	Nejméně významný bit (Least Significant Bit)
HIGH	.....	Představuje logickou úroveň signálu - log1
LOW	.....	Představuje logickou úroveň signálu - log0
I2C	.....	Dvouvodičové datové propojení (TWI)
SDA	.....	Datová linka sběrnice I2C
SCL	.....	Hodinová signál na sběrnici I2C
TWBR	.....	Registr jednotky I2C - nastavujeme přenosovou rychlost
UART	.....	Synchronní / asynchronní sériové rozhraní
Tx	.....	Datová linka rozhraní Uart - slouží k vysílání
Rx	.....	Datová linka rozhraní Uart - slouží k příjmu
UDR	.....	Datový registr jednotky Uart
UBRR	.....	Registr jednotky Uart - nastavujeme přenosovou rychlost
USB	.....	Univerzální sériová sběrnice
ISA	.....	International Standard Atmosphere
V_{DD}	.....	Napájecí napětí
GND	.....	Zemní svorka

Seznam příloh

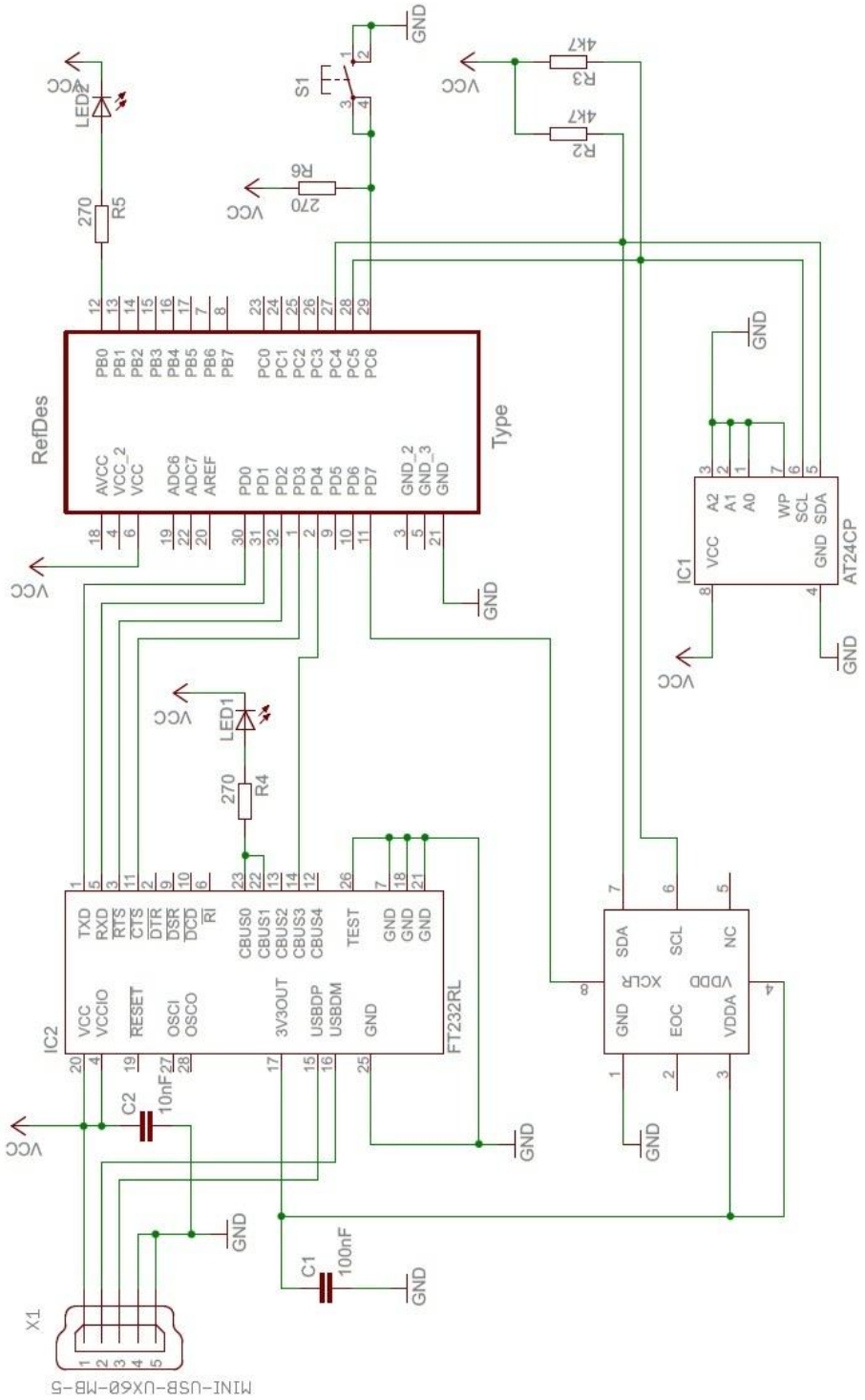
A Návrh zařízení

- A.1 Obvodové zapojení
- A.2 Obvodové zapojení - stabilizátor napětí
- A.3 Deska plošného spoje - Top
- A.4 Deska plošného spoje - bottom
- A.5 Osazení součástek desky plošného spoje

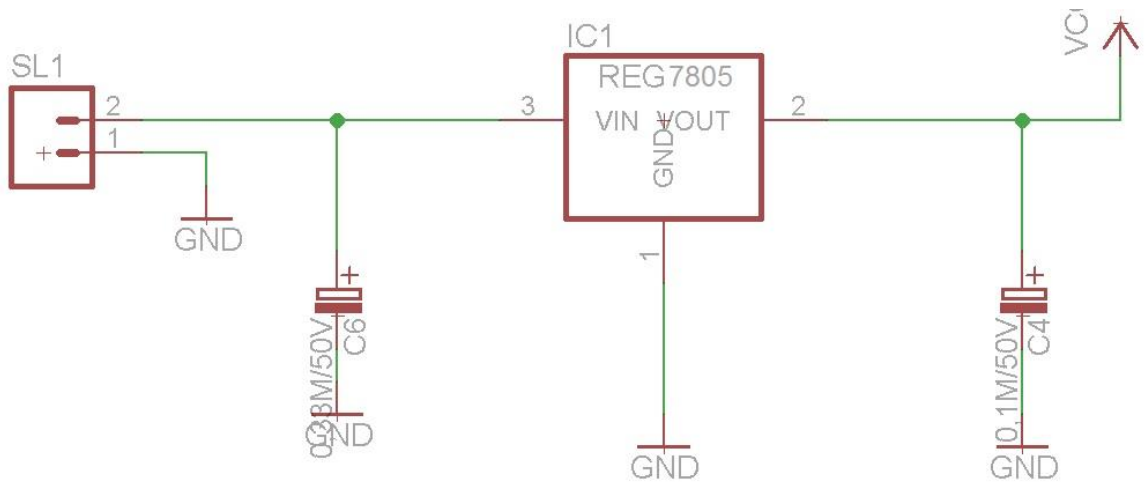
B Seznam součástek

A Návrh zařízení

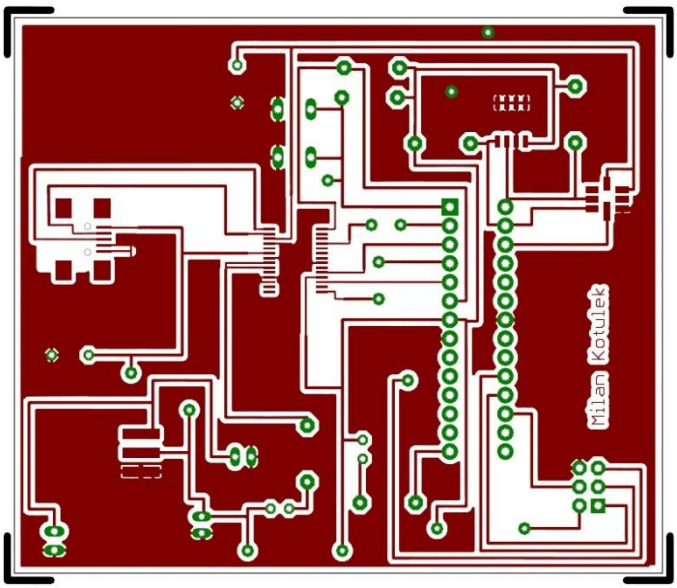
A.1 Obvodové zapojení



A.2 Obvodové zapojení - stabilizátor napětí

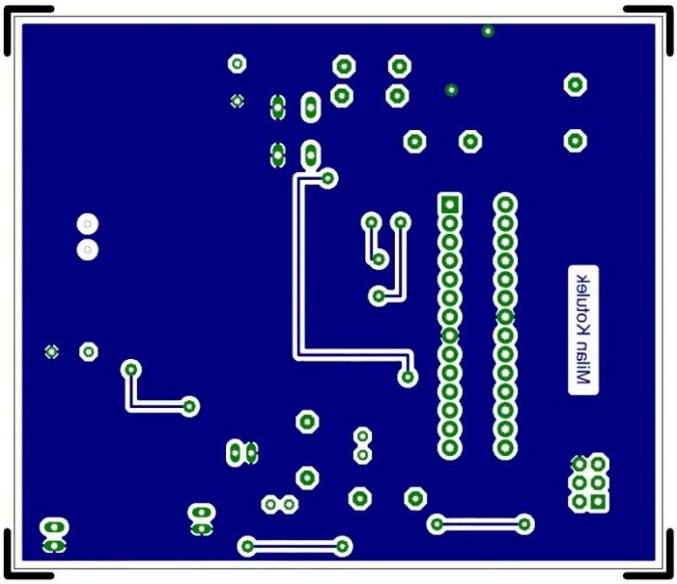


A.3 Deska plošného spoje - Top



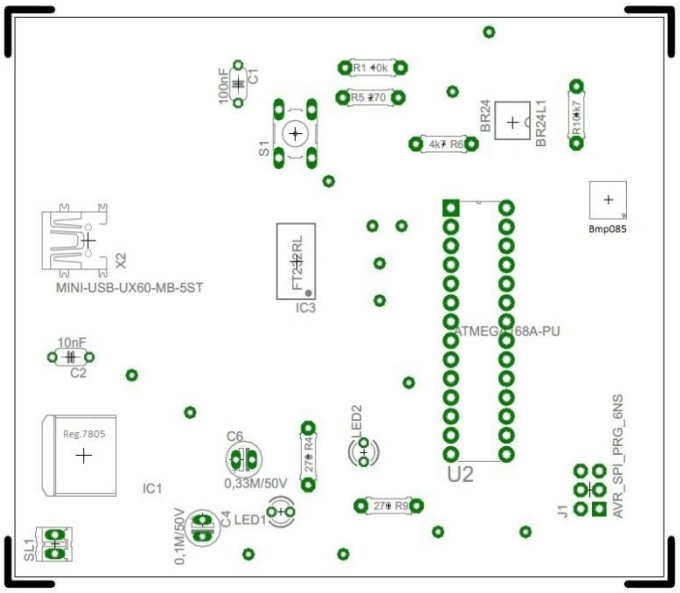
Rozměr desky 90 x 78 [mm], měřítko M1:1

A.4 Deska plošného spoje - bottom



Rozměr desky 90 x 78 [mm], měřítko M1:1

A.5 Osazení součástek desky plošného spoje



Rozměr desky 90 x 78 [mm], měřítko M1:1

B Seznam součástek

Označení	Hodnota	Pouzdro	Popis
C1	10nF /0,01uF	5 mm	Keramický kondenzátor
C2	100nF /0,1uF		Elektrolytický kondenzátor
C3	0,33uF	5 x 11 mm	Elektrolytický kondenzátor
C4	0,1uF	4 x 7 mm	Elektrolytický kondenzátor
LED	3mm 2x		Svítivá - zelená
R1	270 Ω 3x	R0206	Metalizovaný rezistor
R2	10k Ω	R0207	Metalizovaný rezistor
R3	4k7 Ω 2x	R0207	Metalizovaný rezistor
FT232RL	FT232RL	SSOP	Převodník USB - UART
Bmp085	Bmp085	LCC8	Tlakové čidlo
BR24	24LC32AT-I/SN	SOIC8	Paměť Eeprom
AtMega	ATMEGA168-20PU	DIL28	Mikrokontrolér
S1	TD-03XG SMD	6 x 6 x 3,5 mm	Mikrospínač SMD
Mini USB	USB Mini B SMD	USB Mini B	USB konektor
Reg.7805	7805 CD2T SMD	D2PAK	Stabilizátor napětí +5V 1A
Avr.SPI	MLW06G	2x3 kontakty	Konektor pro ploché kabely do DPS přímý
SL1	DCI 006-PI	klips baterie 9V	Napájecí konektor