TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

BAKALÁŘSKÁ PRÁCE

Záznam jízdy

Liberec 2006 Roman Bulíř

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Katedra softwerového inženýrství

Studijní program: B2612 - Elektrotechnika a informatika

Roman Bulíř

Záznam jízdy

Registration drive

Vedoucí bakalářské práce: Ing. Martin Vlasák
Fakulta mechatroniky a mezioborových inženýrských studií
Technická univerzita v Liberci

Konzultant: Ing. Tomáš Martinec
Fakulta mechatroniky a mezioborových inženýrských studií
Technická univerzita v Liberci

Anotace

Cílem bakalářské práce je realizace softwaru i hardwaru pro měření záznamu průběhu jízdy pohybujícího se autíčka. Hlavními body práce jsou navržení desky plošných spojů (DPS), osazení DPS součástkami a oživení dílčích obvodů, následně změření hodnot napětí na stejnosměrném motorku s uložením do paměti a jako poslední částí je transport zaznamenaných dat (obousměrně) mezi mikrokontrolérem a počítačem.

Abstract

The aim of this bachelor thesis is the realization of as the software so the hardware for the measuring of the record the moving toy-car´s course of the ride. The main points of the thesis are the proposals of the surface connection board ("DPS"), recess the "DPS" with the components and galvanization of the partial perimeters. Consequently the next points of the thesis are dealing with taking the measurements of the merits of the tension on the direct-current motor, and also with the storing. The concluding part is the transport of the recorded data (bi-directionally) between the micro-controller and the computer.

Úvodem bych chtěl poděkovat Ing. Martinu Vlasákovi a Ing. Tomáši Martincovi za odborné vedení a cenné rady, které napomohly ke splnění uložené práce.

Obsah

1. Úvod

V současné době je využití mikrokontrolérů zcela běžné a lze je potkat téměř v kterémkoliv odvětví průmyslu (strojírenství, automobilky, lékařství, spotřební elektronika, energetika ....). Několik příkladů aplikací, které mikrokontroléry mohou plnit jsou : měřící technika, regulační technika, komunikační technika, řídící technika … .

V části nazvané hardware jsou popsány základní pojmy jak z oblasti stejnosměrných elektromotorků, tak i z oblasti mikrokontrolérů. Bude zde podrobněji popsáno připojení veškerých periferií, které s mikrokontrolérem spolupracují. Nebude chybět ani seznam použitých součástek.

Část nazvaná RS232 má za úkol přiblížit sériové rozhraní s jeho parametry, možnostmi, výhodami i nevýhodami.

Část s názvem software bude věnována vysvětlení postupu při vývoji celé aplikace. Popis bude pro zjednodušení rozdělen do jednotlivých módů. Nebudou chybět příslušné vývojové diagramy, počáteční nastavení registrů a proměnných.

Pátá část s nazvaná zhodnocení experimentu přiblíží výsledky práce a souvislosti, které úlohu provázely. Budou zde i grafy navzorkovaného průběhu napětí na elekromotorku.

Část šest s názvem závěr uvede zrekapitulování výsledků s následnými dalšími možnostmi.

V příloze se nalézá aplikace vytvořená v programu C++Builder 5, zdrojový kód mikrokontroléru, dokumentace k použitým součástkám.

2. Hardware

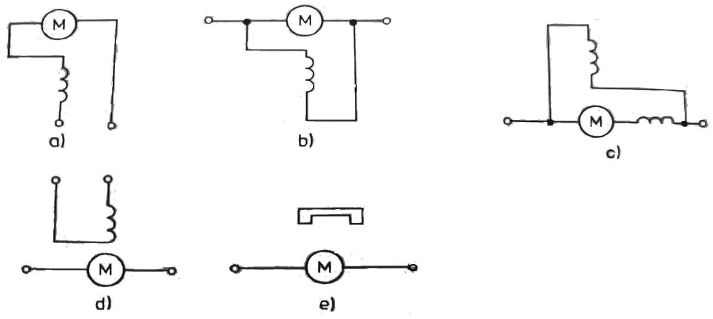
2.1 Stejnosměrné elektromotory

Elektromotory jsou stroje, které přeměnují dodanou elektrickou energii na energii rotační.

2.1.1 Základní rozdělení stejnosměrných elektromotorů

Rozdělení elektromotorů dle způsobu buzení viz obr.1:

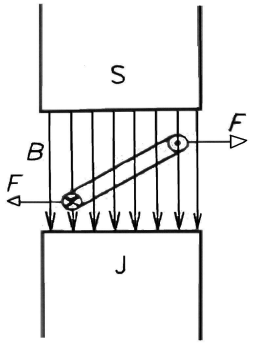
- sériové obr.1.a
- paralelní obr.1.b
- smíšené obr.1.c
- cizí obr.1.d
- s permanentními magnety obr.1.e



Obr. 1: Přehled elektromotorů dle způsobu buzení

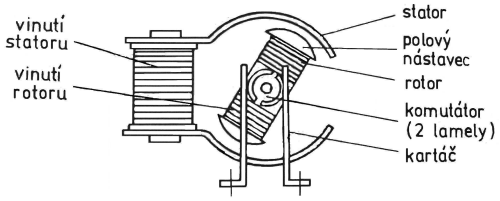
2.1.2 Princip elektromotoru

Točivý moment je důsledkem vzájemného působení magnetického pole rotoru a statoru. Tato pole jsou závislá na velikostech proudů, které procházejí vinutími. Točivý moment závisí na součinu proudu, procházejícího vinutím rotoru, a magnetického toku statoru. Ke vzniku točivého momentu dochází následovně viz obr.2. Mezi póly statoru a rotoru působí magnetické pole. Vodiči ve vinutí rotoru prochází proud a v magnetickém poli na tyto vodiče působí síla dle vztahu $F = B\,I\,l \sin\alpha [N, T, A, m]$, kde B je



Obr. 2: Vznik točivého magnetického pole elektromotoru

magnetická indukce, I je velikost protékajícího proudu, l je aktivní délka smyčky v magnetickém poli a α je úhel, pod kterým protíná vodič magnetické siločáry. Působením komutátoru prochází elektrický proud vinutím pod póly stále stejným směrem. Vlivem působící síly se rotor pootočí a komutátor přepne proud do dalších částí vinutí. Toto se neustále opakuje. Proud je tedy do jednotlivých vinutí kotvy přepínán komutátorem. Komutátor (kolektor) je složen z měděných lamel, které mají obvykle tvar povrchu válcových výsečí, vzájemně od sebe izolovaných a upevněných na hřídeli rotoru, s níž se otáčejí. Na jednotlivé lamely jsou vyvedena vinutí kotvy. Na komutátor dosedají grafitové kartáče, které po lamelách klouzají. Takto je přiváděn proud do rotoru viz obr.3.



Obr. 3: Funkce komutátoru

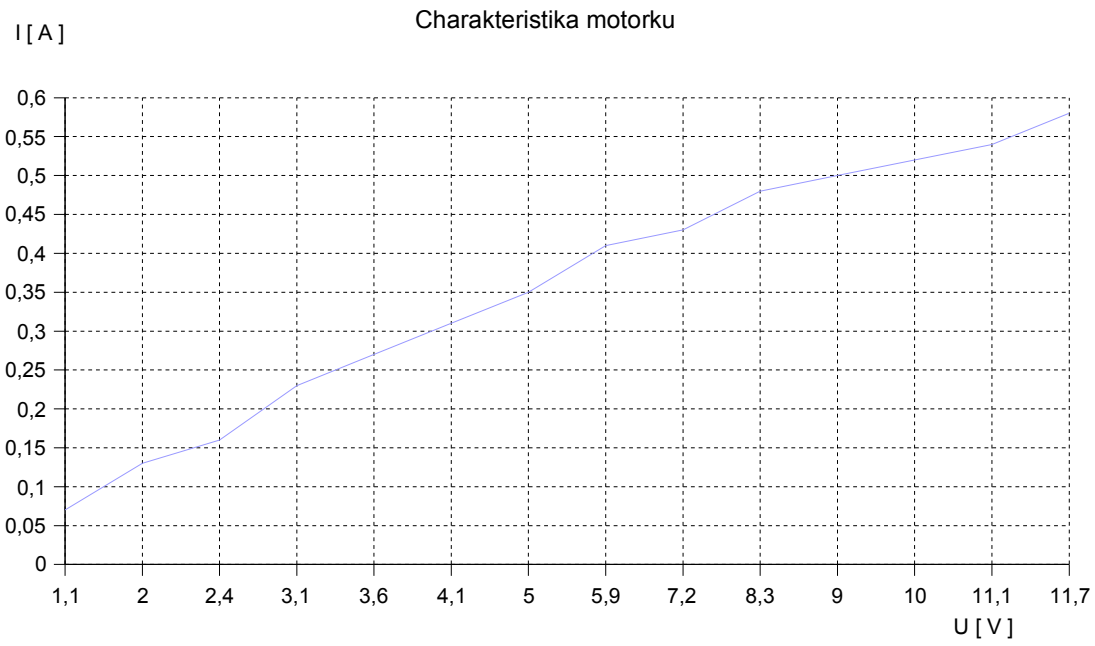
Ve skutečnosti nejsou lamely dvě, ale je jich více, a to v závislosti na počtu závitů. Je obecně známo, že otáčející se komutátor generuje elektromagnetické rušení a je také zdrojem poruch. Měl by proto být náležitě ošetřován (nevydřený, nezkratované lamely,..).

2.1.3 Stejnosměrný motor s permanentními magnety

Stator je tvořen permanentími magnety, které vytvářejí magnetické pole statoru a není tak třeba statorového vinutí. Po připojení motorku k napájení dosáhne velikost magnetického pole statoru okamžitě jmenovité hodnoty a bude po celou dobu konstantní v celém rozsahu otáček. V rotoru se rychle v závislosti na otáčkách zvyšuje indukované napětí, takže se rychle zmenšuje procházející proud, ale i točivý moment. Když dosáhne indukované napětí téměř velikosti napájecího napětí, potom se otáčky ustálí. Rozdíl bude dán jen úbytkem napětí na vinutí kotvy, způsobeným procházejícím proudem. Tento úbytek je poměrně malý, protože i odpor rotoru je malý. Protože je magnetické pole statoru konstantní, zmenšuje se také velmi málo i indukované napětí a otáčky motorku se mění se zatížením jen málo. Regulovat otáčky takovéhoto motorku lze změnou proudu procházejícího rotorem. Motorek je konstrukčně poměrně jednoduchý a proto je často používán pro pohon různých hraček (autodráha, vláčky,...).

2.1.4 VA-charakteristika motorku s permanentními magnety

Tato charakteristika (viz graf 1) byla odměřena na rovném úseku a byla zaznamenávána hodnota proudu, při které se autíčko rozjelo.



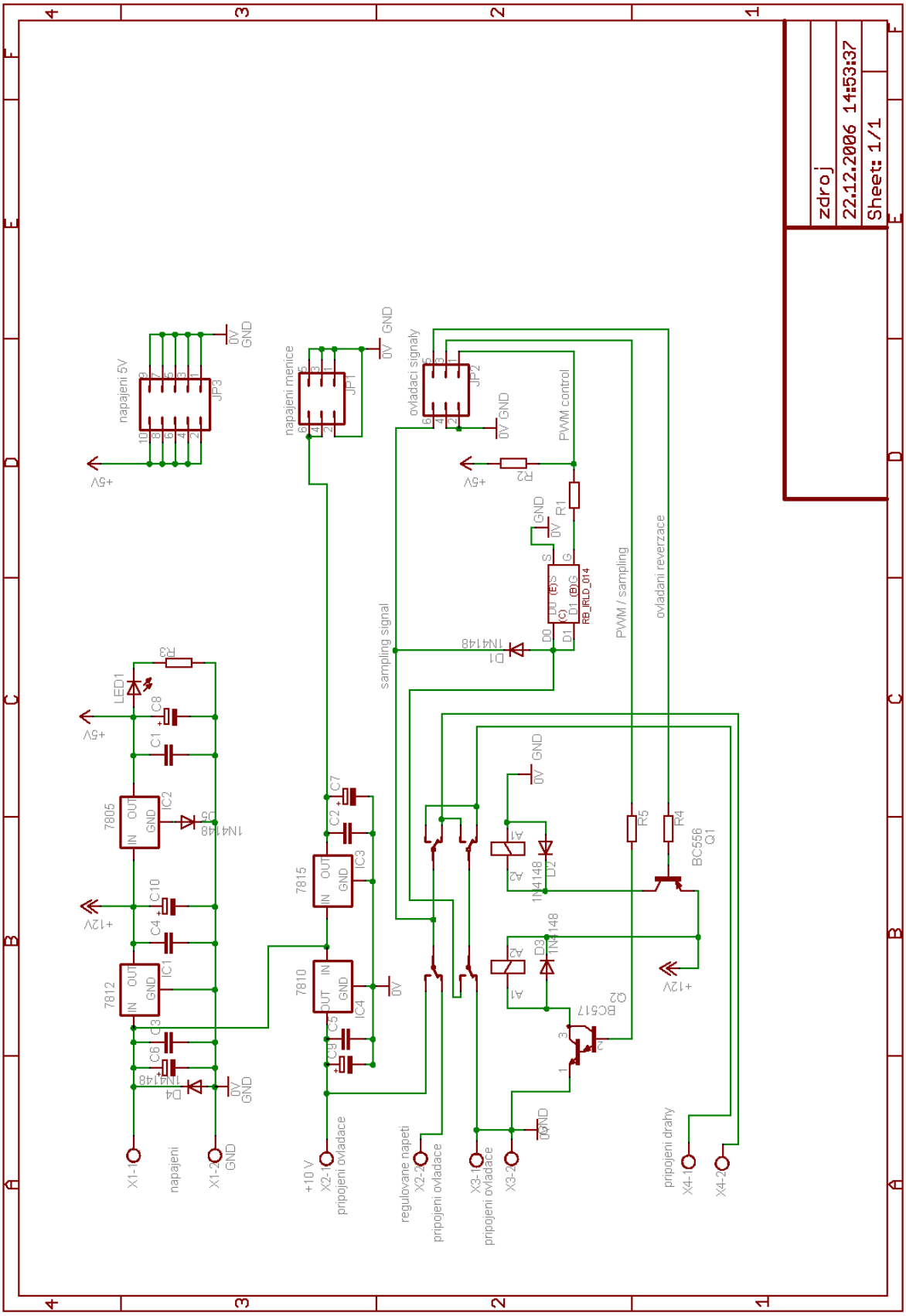
Graf 1: Charakteristika motorku s permanentními magnety ve statoru

2.2 Vývojové prostředí Eagle 4.16

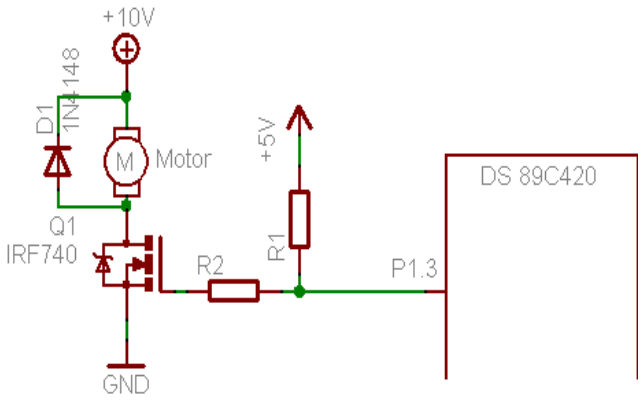
Jedná se o vývojové prostředí určené pro návrh, a to jak schemat, tak i desek plošných spojů. Je volně dostupné na webu a disponuje mnoha předdefinovanými knihovnami součástek s příslušnými pouzdry. Malou „vadou na kráse“ je, že rastr, kam lze pokládat pouzdra součástek, nesmí přesahnout rozměry 80x100 mm. Pro konstrukci desek s většími rozměry lze po zakoupení licence obdržet plnou verzi bez omezení na rozměrech.

2.2.1 Schema a deska plošných spojů „Zdroj“

Na obr. 4 je schema „Zdroj“. Zapojení je poměrně jednoduché a proto je uveden jen rámcový popis. Jsou zde čtyři integrované stabilizátory napětí. Na vstupech i výstupech mají zapojené kondenzátory. Elektrolytický tvoří „zásobník“ energie pro případný krátkodobý pokles napájecího napětí, a keramický vyfiltruje případný rušivý signál šířící se po napájecínm vedení. Dále je zde dvojice relátek. RE1 svými kontakty přepíná mezi jízdou na ovladač nebo jízdou na PWM regulaci. RE2 slouží pro změnu polarity na autdráze. Obě relé jsou spínána tranzistory. K cívce relé je nutné (paralelně v opačném směru) připojit diodu, která chrání tranzistor před indukovaným napětím, vznikajícím při rozpojování induktivní zátěže. V bázi mají tranzistory zařazeny srážecí odpory vypočtené takto: $h_{21E} = I_C / I_B \quad [-, A, A]$, kde I_C je proud procházející cívkou relé. Tento byl odměřen digitálním multimetrem na nepájivém kontaktním poli a činil $I_C = 12\ 10^{-3}$A, proudový zesilovací činitel h_{21E} = 30 000 (Darlinghtonův tranzistor) => $I_B = I_C / h_{21E}$ po číselném dosazení činí $I_B = 1210^{-3} / 30000$. Byl získán $I_B = 4\ 10^{-7}$A. Skutečný I_B však musí být o 50 až 80% větší, aby byl tranzistor v úplné saturaci. Skutečný $I_B = 7\ 10^{-7}$ A. Odpor bude podílem úbytku napětí na rezistoru a procházejícího proudu $R5 = (U_{CC} - U_{BE}) / I_B$ => $R5 = (5 - 0{,}7) / 7\,10^{-7}$ => R5 = 6,2 MΩ. Obdobným způsobem je vypočten i rezistor R4. Jediný rozdíl je, že druhý tranzistor je typu PNP. Dále je na schematu MOSFET tranzistor s kanálem N, který je ovládán mikrokontrolérem při PWM regulaci. Podrobnější zapojení je na obr.5. Protože tranzistor opět spíná induktivní zátěž, byla k motorku paralelně (v závěrném směru) připojena dioda. Rezistor R2 urychluje spínání (rozpínání) MOSFETU. Rezistor R1 plní úlohu při eventuálním průrazu elektrody GATE, aby se nezničila výstupní brána mikrokontroléru nadměrným proudem. Dále je zde ještě led dioda s předřadným rezistorem R3. Signalizuje funkčnost zdroje +5V. Stabilizovaná napětí jsou vyvedena na lámací lišty odkud jsou plochým vodičem spojeny s druhou deskou. Na obr.6 je zobrazena deska plošných spojů s názvem „Zdroj“. Deska je

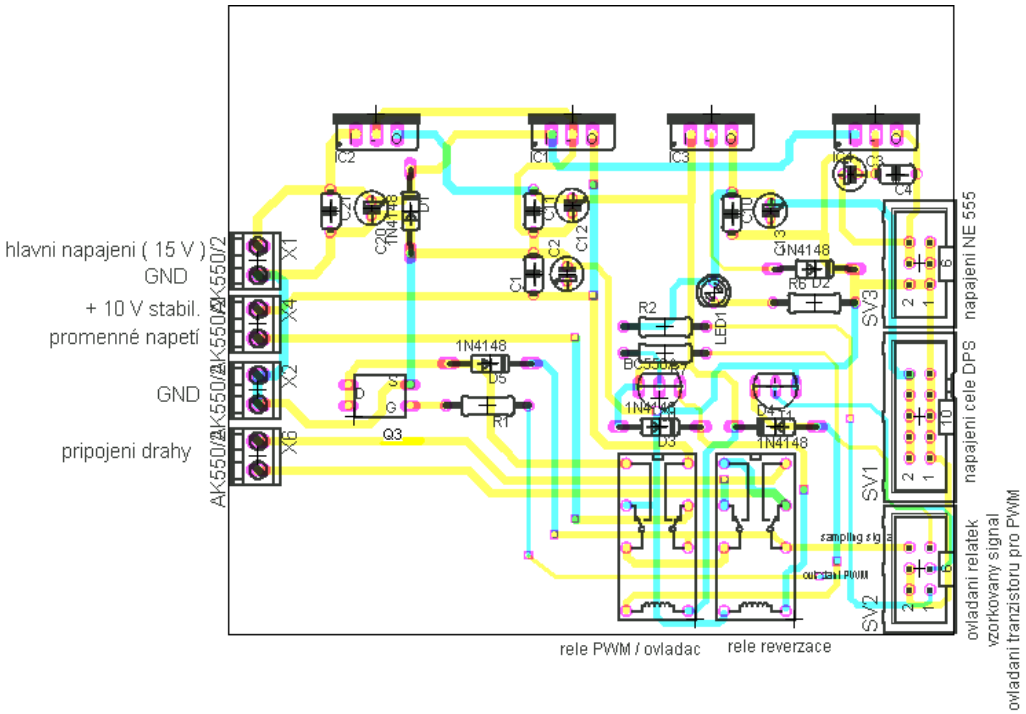


Obr. 4: Schema „Zdroj“



Obr. 5: Schema PWM regulace

oboustaná s rozměry 75x90mm. V levé části jsou umístěny svorky, ke kterým je přivedeno napájecí napětí, připojen ovladač a připojena dráha. V pravé části jsou již zmiňované lámací lišty. V horní části jsou integrované stabilizátory. Za nimi je ponecháno místo pro připevnění chladiče.



Obr. 6: DPS „Zdroj“

IC2 78L05	1 ks
IC1 78L12	1 ks
IC3 78L15	1 ks
IC4 78L10	1 ks

D1 až D6 1N4148	6 ks
C6 až C10 47µF/35V	5 ks
C1 až C5 100nF	5 ks
R1 8k2	1 ks
R2 10k	1 ks
R3 330Ω	1 ks
LED1 (o 3 mm, žlutá)	1 ks
BC 556B (PNP)	1 ks
BC 517 (NPN)	1 ks
IRLD 014 (kanál N)	1 ks
Relé RY12W-K	2 ks
AK 550/2	4 ks
SV1, SV2, SV3 (lámací lišta dvouřadá)	11 ks

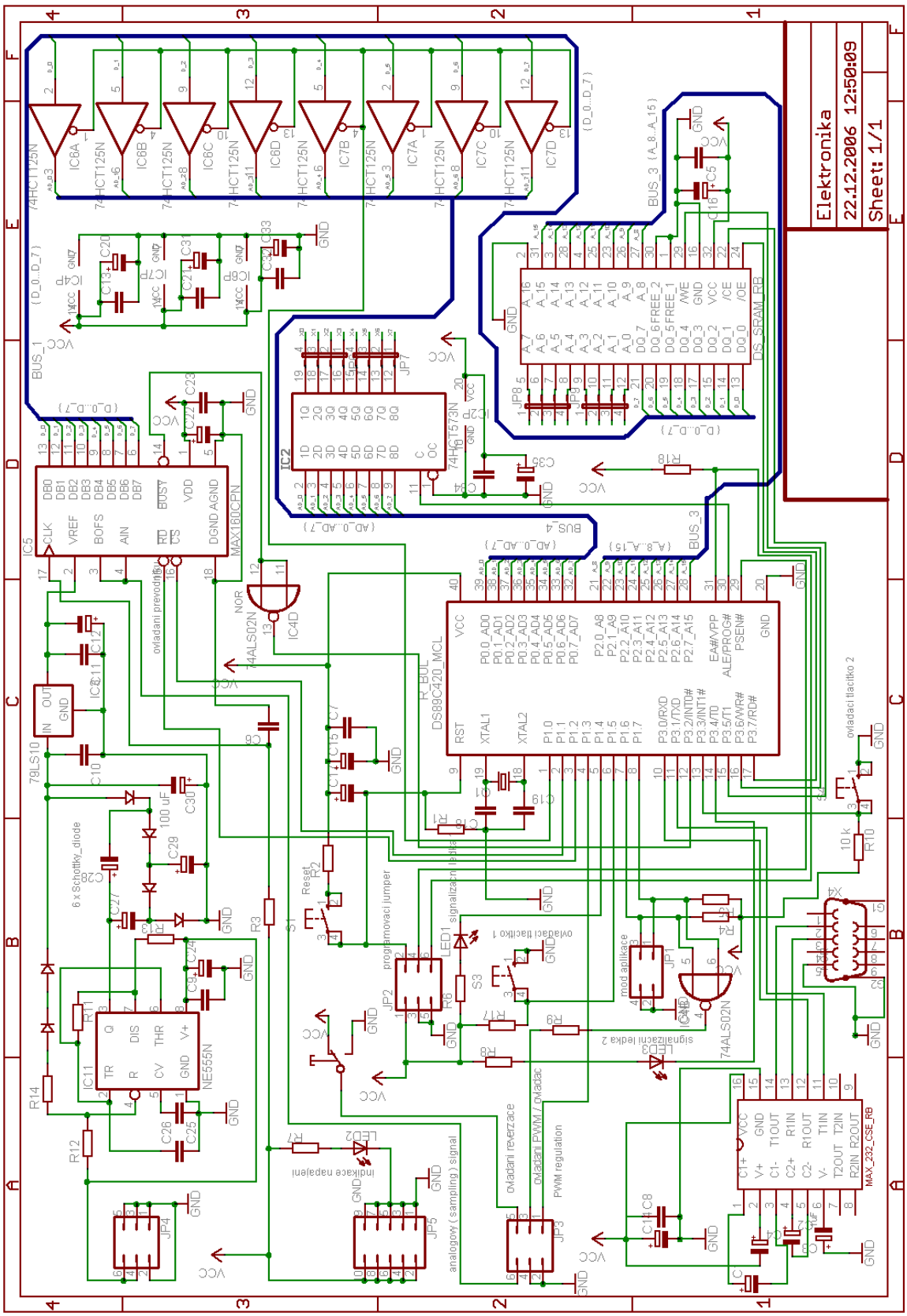
Tab 1: Seznam použitých součástek na DPS „Zdroj“

S oživením desky by pro celkovou jednoduchost neměly být žádné problémy. Spoje, které jsou proudově namáhány, byly navrženy s větší šířkou, aby nedocházelo k zahřívání a úbytkům na napětí. Také zemnící spoje (GND) jsou dostatečně dimenzovány, aby nevznikaly napěťové rozdíly mezi jednotlivými deskami. Ostré rohy jsou zalamovány pod úhlem 45°, aby jimi proud lépe protékal.

2.2.2 Schema „Elektronika“

Schema na obr.7 ukazuje kompletní zapojení. Z důvodu poměrné složitosti je celé schema rozděleno do několika částí. Tyto menší celky jsou poté podrobněji přiblíženy. U každé z těchto dílčích kapitol je uveden seznam použitých součástek.

- mikrokontrolér DS89C420
- A/D převodník MAX 160
- externí paměť K6X4008CIF
- sériová komunikace
- měnič polarity a násobič napětí



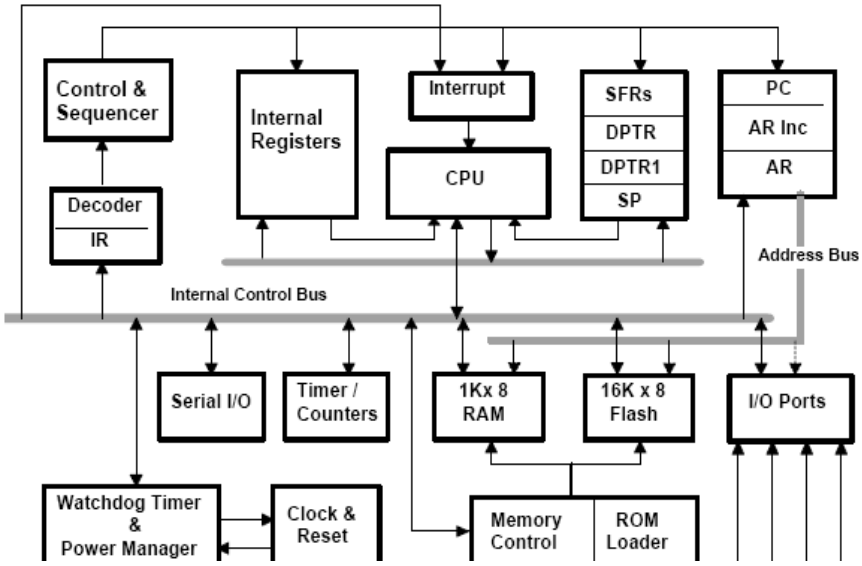
Obr. 7: Schema „Elektronika“

2.2.2.1 Mikrokontrolér DS89C420

Jde o mikrokontrolér vycházející svou podstatou z procesorů řady 8051. Má tedy harwardskou architekturu (oddělená paměť dat od programové paměti). Tento mikrokontrolér je vybrán pro relativní jednoduchost, dostupné vývojové prostředí a volně stažitelný překladač. Pro přiblížení je uvedeno několik nejdůležitějších informací.

- napájení + 5V, max. taktovací kmitočet 33 MHz
- 4 vstupně/ výstupní porty (P0 bez pull-up, P1, P2, P3)
- 16 kB programové paměti, 1 kB vnitřní datové SRAM paměti
- 2 plně duplexní sériové kanály
- programovatelný Watchdog Timer (hlídací obvod)
- 5 úrovní přerušení
- 13 zdrojů přerušení (napájení, 2x sérivá linka, watchdog, 3x Timer, 6x externí)
- 3 Timery

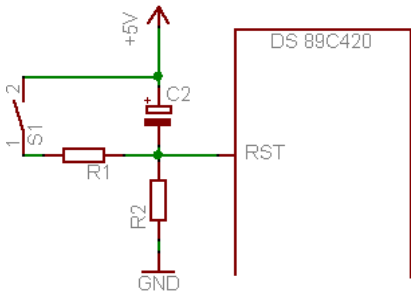
Na obr.8 je vnitřní uspořádání tohoto mikrokontroléru.



Obr. 8: Vniřní struktura DS89C420

Aby mohl mikrokontrolér pracovat, je nutné připojit jej nejen na napájecí napětí (+5V), ale zároveň připojit piezokeramický rezonátor („krystal“) k vývodům XTAL1 a XTAL2. Hodnota použitého krystalu činí 22118400 Hz. Někdy je vhodné připojit od každé z výše uvedených elektrod ještě kondenzátory (2x27pF) vůči nulovému potenciálu (GND), z důvodu případného zakmitávání krystalu. Další nezbytností pro činnost mikrokontroléru je obvod RESETU viz obr.9. Jako poslední uvedu informace o programování

mikroprocesoru, k čemuž slouží propojka JP2. Mikrokontrolér je uveden do programovacího stavu nasazením všech tří jumperů na propojku, která je na DPS umístěna nad mikrokontrolérem. Příslušné piny budou zapojeny takto: RESET=log.1, EA a PSEN=log.0. Při standartním vykonávání programu (jumpery odstraněny) budou jednotlivé piny zapojeny následovně: PSEN zůstává volný, pin RESET v log1 nejméně po 2 strojové cykly při plné činnosti oscilátoru a poté na potenciálu GND a EA přes rezistor 10kΩ na +5V.

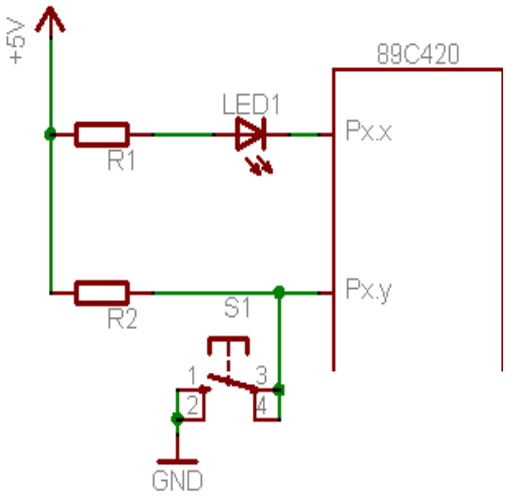


Obr. 9: Obvod RESETU

Hodnoty použitých součástek viz tab.2 jsou převzaty z použité literatury.

R1 470 Ω	1 ks
R2 1k	1 ks
C2 4,7µF/ 10V	1 ks

Tab.2: Hodnoty součástek v obvodu RESETU



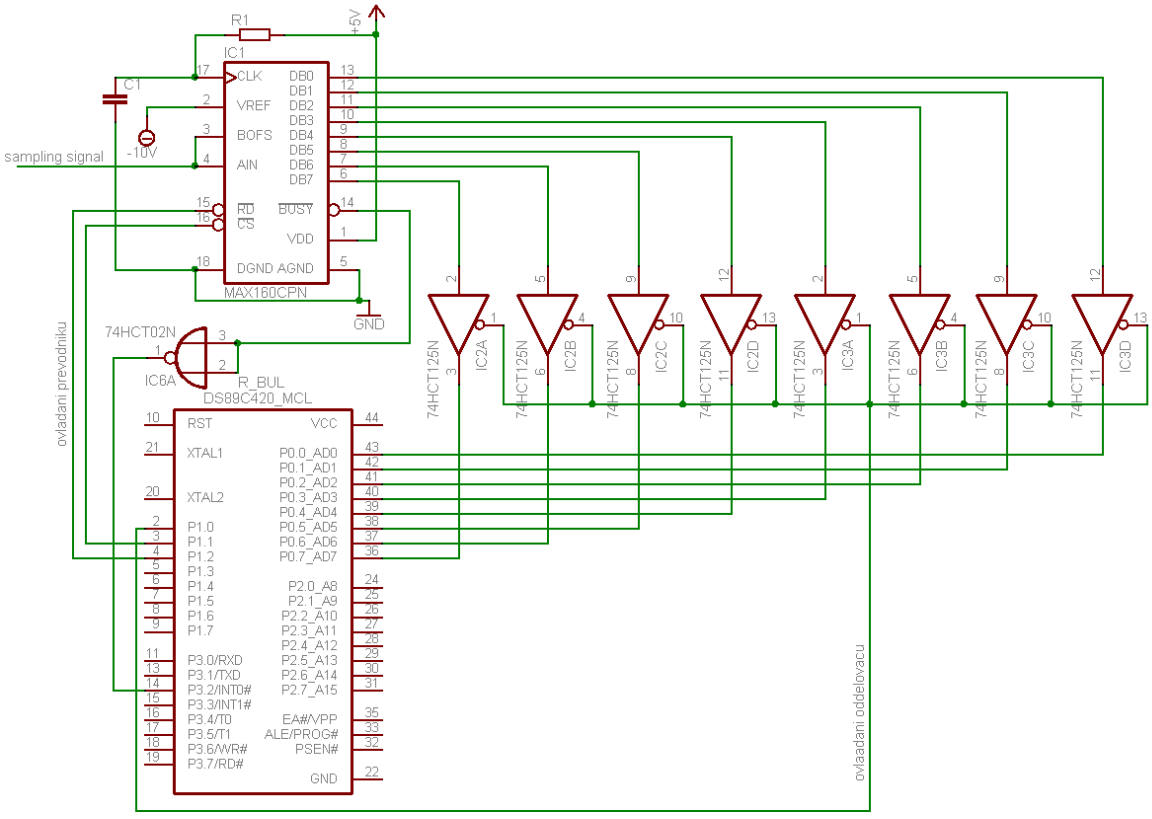
Obr. 10: Přpojení led diody a tlačítka

Jen pro úplnost je uvedeno připojení tlačítka a led diody k mikrokontroléru viz obr.10. Led dioda bude svítit, když na *Px.x* bude log.0. Při *Px.x* = log.1 led dioda svítit nebude.
K připojení tlačítka jen tolik. Při rozepnutém tlačítku bude na *Px.y* log.1 přiváděná přes rezistor R2. Při stisknutí tlačítka bude *Px.y* = log.0.. Hodnoty součástek viz.tab.3

R1 330 Ω	1 ks
R2 10k	1 ks
LED dioda (d=3 mm)	1 ks
tlačítko	1 ks

Tab.3: Hodnoty součástek pro připojení led diody a tlačítka

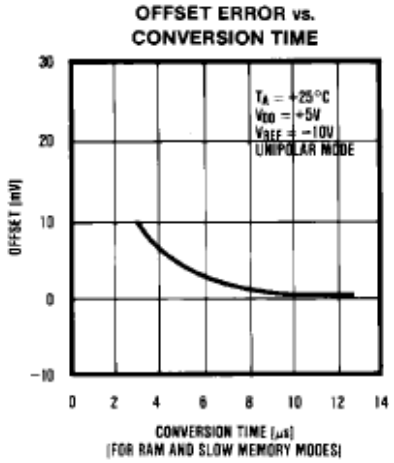
2.2.2.2 A/D převodník MAX 160



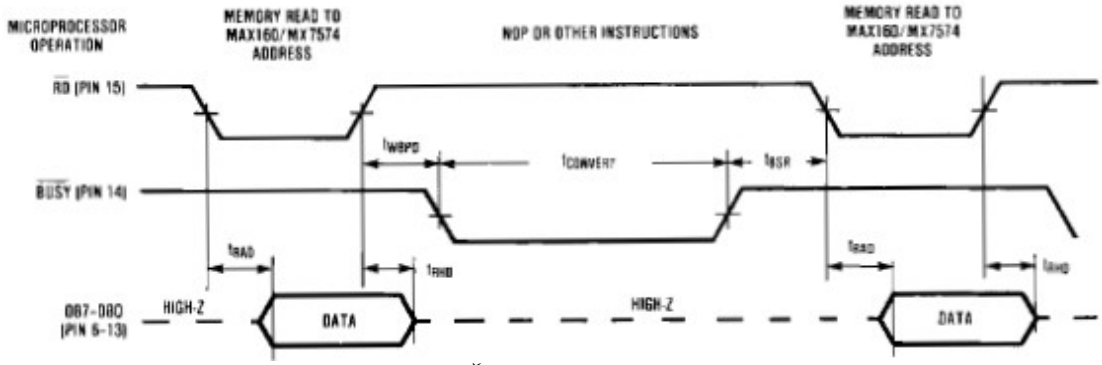
Obr. 11: Připojení A/D převodníku k mikrokontroléru

Jdná se o 8-bitový převodník s postupnou aproximací pracující s TTL úrovněmi a paralelním výstupem dat. Připojení převodníku k mikrokontroléru ukazuje obr.11.
Převodník má možnost pracovat ve dvou režimech. První režim je bipolární, kde vstupní napětí je v rozsahu -10V až +10V (B_{OFS} = +10V). Druhý režim je unipolární se vstupním napětím v rozsahu 0V až +10V (B_{OFS} = A_{IN}). Druhý z režimů je použit pro tuto aplikaci,

čímž je získáno rozlišení $Q = rozsah / 2^{počet.bitů} [V/bit, V, bit]$, což po číselném dosazení $Q = 10/256$ vychází $Q = 39\text{mV}/bit$. Rozlišení je dle mého soudu dostatečné s ohledem na citlivost elektromotorku poháněnjícího autíčko. Dalším nastavením lze určit, jaký signál bude použit pro převod. První z možností je použití externího taktovacího signálu, nebo mohu použít integrovaný časovač v A/D převodníku. Pro jednoduchost je zvolen interní časovač. Proto je k převodníku připojen kondenzátor C1 s rezistorem R1. Hodnoty obou součástek mají vliv na dobu převodu. Čím nižší bude časová konstanta tohoto obvodu, tím kratší bude i čas převodu. Dle dostupné dokumentace se zvolily hodnoty součástek takto C1=100pF a R1=50kΩ, čímž je získána doba převodu 7 až 9µs. Doba převodu zpětně ovlivňuje chybu offsetu viz. obr.12. Jen pro úplnost, nejkratší možná doba převodu, udávána dokumentací, činí 4µs. Poté je nutno zvolit ještě jeden ze tří možných ovládacích režimů převodníku, v tomto případě ROM mód, ve kterém je převodník ovládán pouze signálem rd_. Druhý ovládací signál (cs_) má být dle



Obr. 12: Vliv chyby offsetu



Obr. 13: Časování v ROM módu

dokumentace trvale uzeměn. Časový diagram je znázorněn na obr.13. Náběžná hrana signálu rd_ způsobí vyresetování a „nastartuje“ nový převod. Dokončení převodu je

signalizováno pinem BUSY (náběžnou hranou). Ten je připojen přes invertor na pin P3.2 (externího přerušení) mikrokontroléru. Pro čtení dat z převodníku musí být pin rd_ uzemněn. Následně jsou zprůchodněny 3-stavové oddělovače a hodnota se objeví na portu P0. Použití 3-stavových oddělovačů je nezbytné, protože za použití standartní instrukce pro čtení (zápis) nelze naadresovat externí paměť procesorem a data poslat jiným zařízením. Data se musí nejprve načíst do mikrokontroléru a až poté odeslat do externí paměti. Výše zmíněný převodník potřebuje ke své činnosti referenční napětí -10 V. Získání tohoto napětí bude pro jeho komplikovanost popsáno v samostatné kapitole nazvané: Měnič polarity. Seznam použitých součástek obsahuje tab.4.

IC1 A/D MAX 160	1 ks
R1 50 kΩ	1 ks
C1 100 pF	1 ks
IC2 SN74HCT125	2 ks

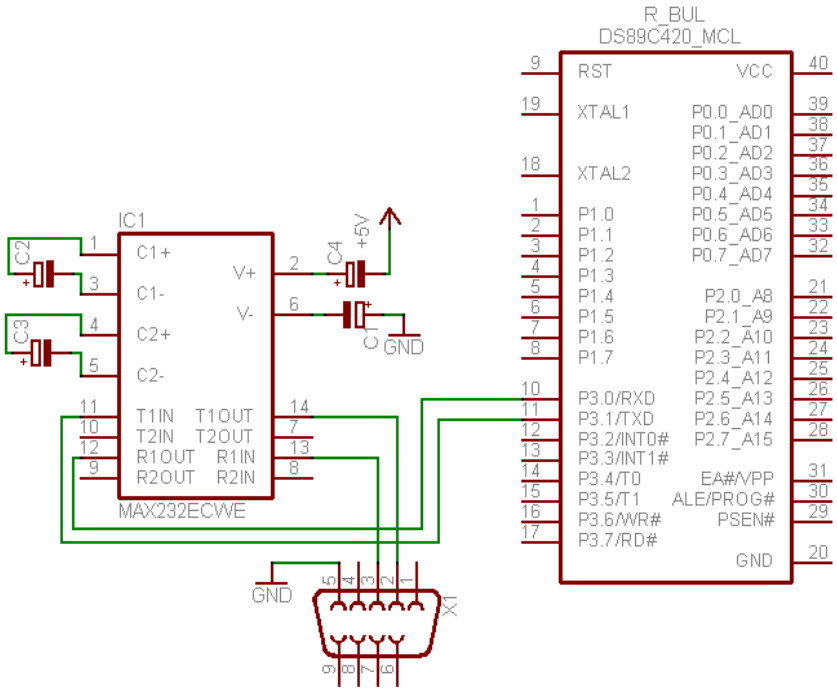
Tab.4: Seznam součástek pro připojení A/D převodníku

2.2.2.3 MAX 232

Na obr. 14 je nakresleno zapojení obvodu MAX 232 k mikrokontroléru. Jedná se o IO, který přizpůsobuje napěťové úrovně z RS 232 na TTL a opačně. Napětí z TTL úrovní na RS232 je získáváno pomocí „nábojové pumpy“, která je uvnitř obvodu. Převod opačným směrem je jednodušší, prorože je napětí snižováno, což obecně většinou nečiní větších problémů. Zmiňovaný obvod potřebuje ke své činnosti čtyři elektrolytické kondenzátory. Seznam použitých součástek je v tab.5

IC1 MAX 232	1 ks
C1 až C4 1μF/50V	4 ks
X1 CANON 9 Z 90	1 ks

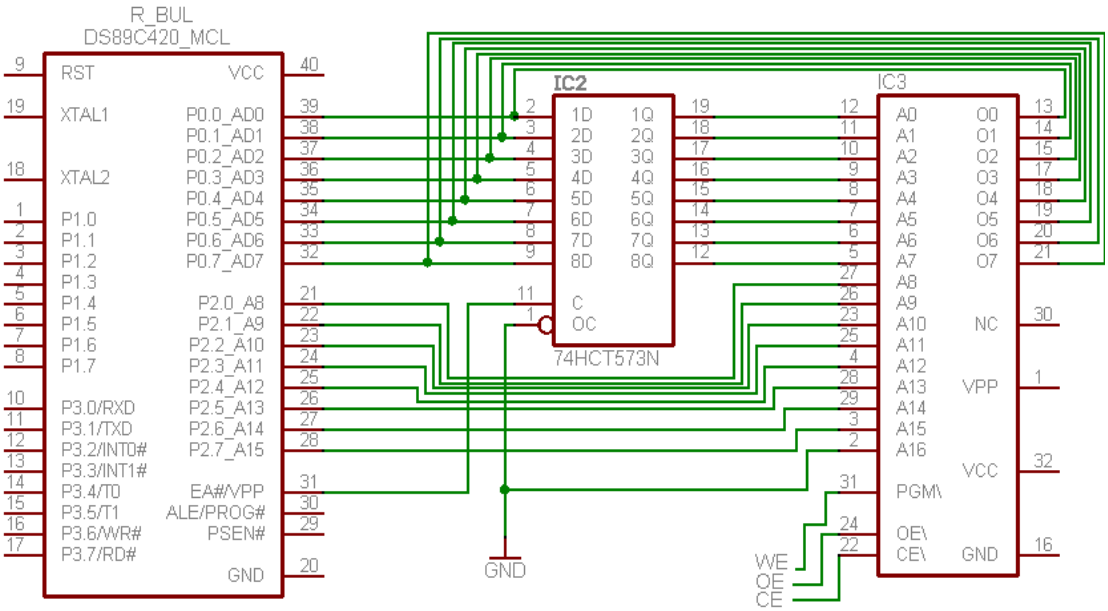
Tab.5: Seznam součástek pro IO MAX 232



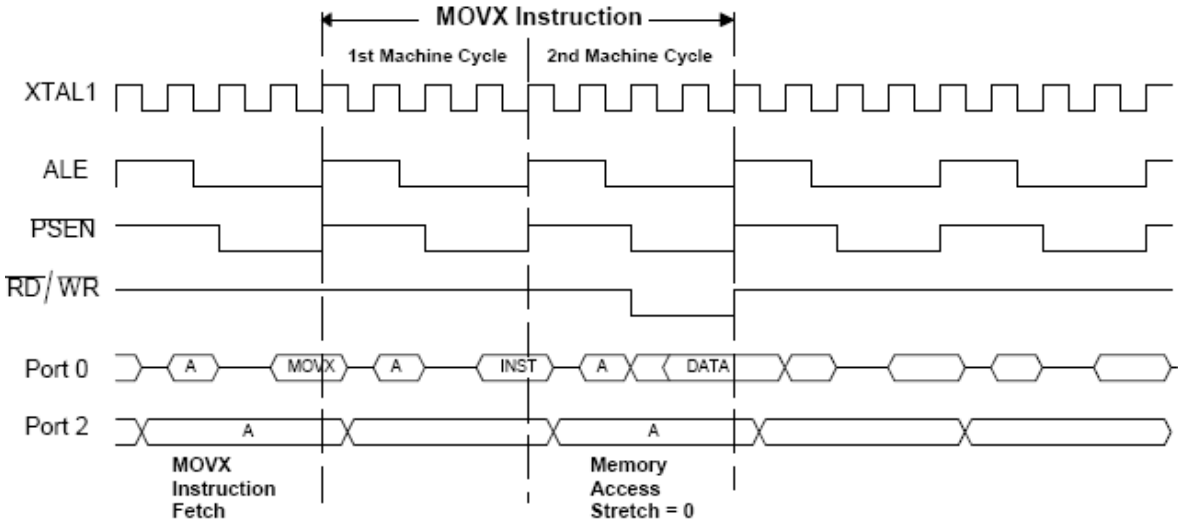
Obr. 14: Přpojení IO MAX 232 k mikrokontroléru

2.2.2.4 Externí datová paměť

Jedná se o paralelní paměť typu SRAM s ovelikosti 512kB pracující v nestránkovém módu. Připojení externí paměti je provedeno standartním způsobem, a to přes 8-bitový záchytný registr (8 klopných odvodů typu D) viz.obr.15. Protože má paměť 19-bitovou adresovou šířku sběrnice, je nutné 3 nejvýznamnější bity uzemnit. Velikost paměti se tím tedy zmenší na 64kB. Princip činnosti (při zápisu) je následující: mikrokontrolér má pro tento účel porty P0 a P2. Port P2 vysílá MSB byte (adresovací) a to přímo na příslušné adresovací piny paměti. Sběrnice P0 je multiplexovaná tzn. nejdříve se na sběrnici objeví (adresovací) LSB byte, který je po ustálení načten do záchytného registru a ihned odeslán na příslušné adresové piny paměti. Poté se na P0 objeví data, která však putují rovnou na datové piny paměti, která by již měla být naadresovaná a tedy schopna data zapsat. Čas, kdy má být stav sběrnice zachycen, je řízen signálem ALE. Pro úplnost je na obr.16 znázorněno časování při zápisu do externí paměti. Čtení z paměti probíhá obdobným způsobem. Pro nastavení komunikace s externí datovou pamětí slouží v mikrokontroléru registry PMR a CKCON. Nastavením registrů lze přizpůsobovat časování sběrnic P0 a P2 a tím komunikovat s různě rychlými externími paměťmi. Protože jde o softwerové nastavení bude podrobněji popsáno v kapitole nazvané Software.



Obr. 15: Připojení externí datové paměti



Obr. 16: Časování P0 a P2 při zápisu do externí datové peměti

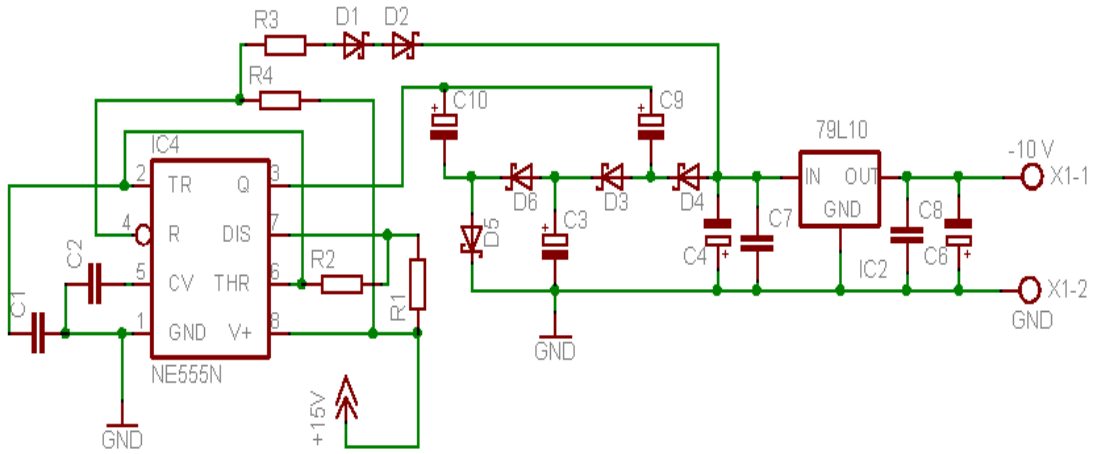
Seznam použitých součástek je v tab.6

IC2 SN74HCT573	1 ks
IC3 K6X4008CIF	1 ks

Tab.6: Seznam součástek pro komunikaci s ext. datovou pamětí

2.2.2.5 Měnič polarity

Výřez ze schematu je na obr.17. Zapojení je pro ulehčení popisu rozděleno na dvě části. První je IO NE555 ve funkci astabilního klopného obvodu, který je nakreslen v levé části. V pravé části je nakreslen obvod násobiče napětí. Celé zapojení pracuje následovně: časovač 555 generuje na svém výstupu (pin3) obdelníkové kmity s přibližnou frekvencí $f = 1{,}4/(R1 + 2R2)C1 \Rightarrow f = 1{,}4/(1000 + 2*100000)10*10^{-9}$, což číselně odpovídá hodnotě $f = 6666{,}7\,Hz$. Výstupní střídavý proud je usměrněn a poté téměř beze ztrát zvýšen.



Obr. 17: Schema měniče polarity

Teď něco o principu násobiče. S příchodem čela impulsu na pinu 3 se kondenzátor C10 v součinnosti s diodou D5 nabíjí kladným napětím. Podobně i C9 se nabíjí přes D3, D6, D5. Při týlové hraně je D5 pólována v závěrném směru, za to D6 se otevírá směrem k C10, čímž umožní přechod náboje z C10 do C3. Následující čelo impulsu nabíjí kondenzátor C9 již na dvojnásobek výstupního napětí, a to pomocí D3 a C3. V další půlperiodě se otevírá D4 a náboj s dvojnásobným napětím přechází do C4. Popsaný proces probíhá ve skutečnosti mnohem rychleji a spojitě. Stabilizátor IC2 má na vstupu a výstupu dvojice kondenzátorů. Elektrolytické tvoří „zásobníky energie“ pro případné kolísání výstupního napětí na pinu3 IO, keramické vyfiltrují případně vysokofrekvenční rušení, generované obvodem. Poměrem odporů R3 a R4 lze regulovat velikost výstupního napětí. Kondenzétory v obvodu násobiče jsou elektrolytické, aby dokázaly udržet náboj a nedocházelo ke kolísání výstupního napětí. Pro správnou činnost násobiče napětí je vhodné použít rychlé (Schottkyho) diody. Poměrem R1, R2a C1 lze regulovat kmitočet astabilního klopného obvodu. Celý tento obvod je vhodné umístit na DPS co nejblíže k zařízení, které

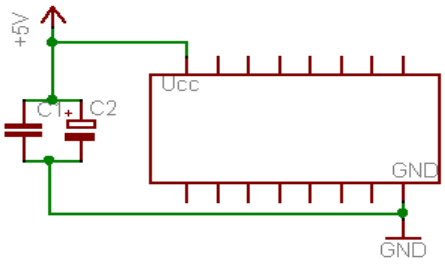
má napájet. Je to z důvodu, že obvod pracuje na poměrně vysokém kmitočtu a mohlo by docházet k šíření rušení a ovlivňování dalších obvodů. Uvedené zapojení dokáže dodat do napájeného zařízení až několik desítek mA, což je pro naše účely dostačující, protože vstupní impedance pinu V_{REF} (A/D převodník) je deklarována minimálně 15kΩ. Napětí je poté ještě stabilizováno integrovaným stabilizátorem napětí 79L10 na -10V. Seznam použitých součástek je uvedeny v tab.7.

IC2 79L10	1 ks
IC4 NE555	1 ks
D1až D6 BAT 43	6 ks
R3,R4 47 kΩ	2 ks
C1, C2 10nF	2 ks
C7, C8 100nF	2 ks
C4 100µF/25V	1 ks
C3, C6, C9, C10 47µF/25V	4 ks
R1 1kΩ	1 ks
R2 10kΩ	1 ks

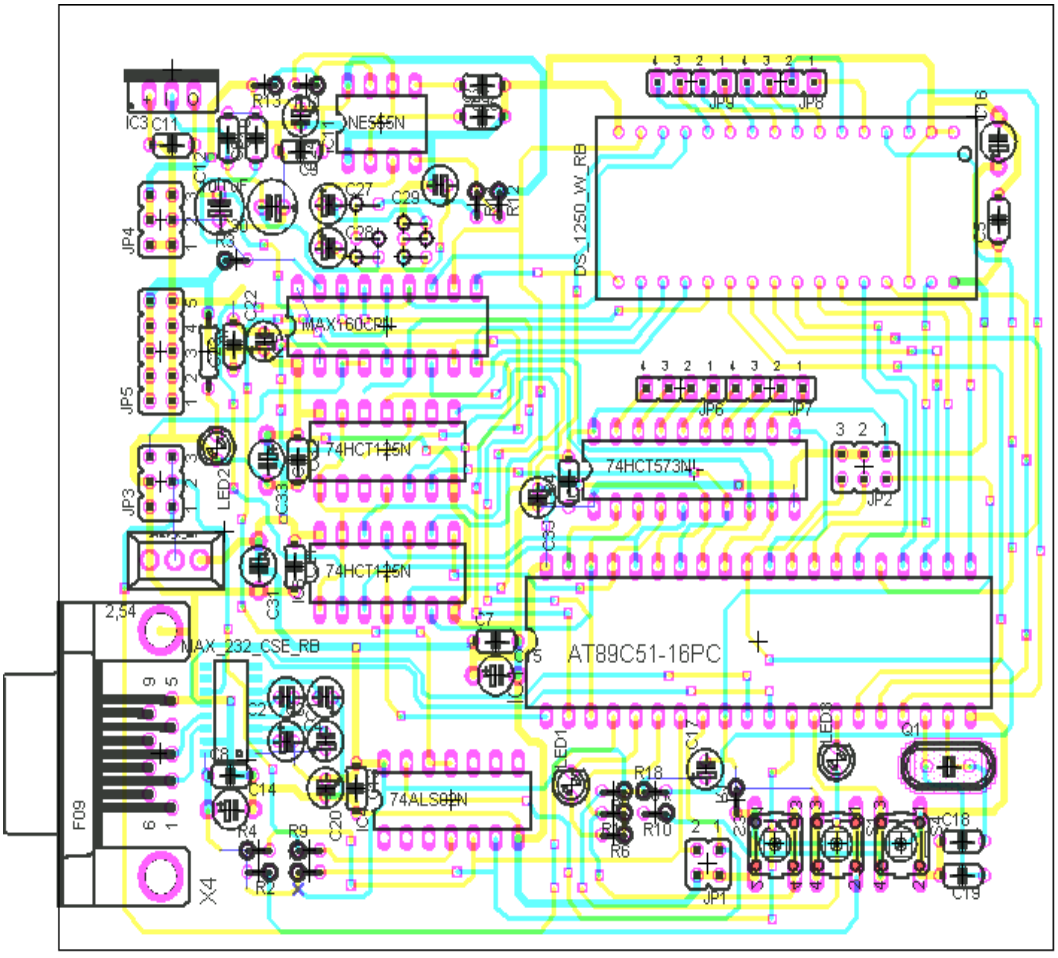
Tab.7: Hodnoty součástek v obvodu měniče polarity

2.2.3 Deska plošných spojů „Elektronika“

Všechny IO jsou zapojeny dle obr.18. Co nejblíže napájecím vývodům je umístěna dvojice kondenzátorů. Elektrolytický tvoří „zásobu energie“pro případné poklesy napájení, keramický vyfiltruje případné rušení, šířící se po napájecím vedení.Osazení a rozmístění součástek ukazuje obr.19. Deska o rozměrech 113x95mm byla navržena jako oboustranná, s galvanicky prokovenými otvory. Pro poměrně velkou hustotu součástek bylo obtížné najít vhodné rozmístění. Všechny IO jsou v pouzdrech DIL a vsazeny do patic pro eventuelní výměnu. To neplatí pro IO MAX232, který je jako jediný v pouzdru SMD (velikost 12,05). S oživením desky nebyly větší problémy. Nejprve byl oživen obvod měniče napětí, poté komunikace po sériovém kanálu. Jako další byla ověřena funkce A/D převodníku. Jedinou komplikací, která se vyskytla, byla komunikace s externí datovou pamětí. Původně navržená paměť s mikrokontrolérem vůbec nespolupracovala. Proto musela být použita ekvivalentní náhrada, která již pracovala bez problémů. V rámci možností byl signál GND veden širšími cestami oproti ostatním spojům, aby nevznikaly napěťové komplikace.



Obr. 18: Připojení IO k napájení



Obr. 19: DPS „Elektronika“

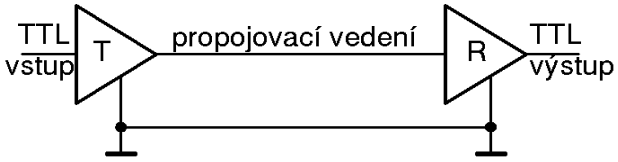
3. RS 232

3.1 Sériové komunikační rozhraní TIA/EIA 232F

RS 232 je standard sériové dvoubodové datové komunikace. První verze rozhraní byla definována americkou organizací EIA (Electronic Industiries Alliance) v roce 1962 pod označením RS 232. Později byla opakovaně revidována a významného rozšíření pak doznala především třetí revize RS 232 C, která pochází z roku 1969. V současnosti nejnovější verze je podle EIA a TIA (Telecommunications Industry Association) oficiálně uváděna pod názvem TIA/EIA 232 F. Vzhledem k tomu, že rozdíly mezi RS 232 C a pozdějšími variantami jsou poměrně malé, a „zažitost" staršího označení je značná, je oficiální označení používáno jen vyjímečně. V literatuře i mezi odbornou veřejností se proto o tomto rozhraní nejčastěji hovoří jako o RS 232 C nebo prostě jen RS 232.

3.2 Elektrické parametry RS 232

Struktura tohoto rozhraní je schematicky znázorněna na obr.20. Zem vysílače



Obr. 20: Schéma rozhraní RS 232

i přijímače je společná a logické úrovně na signálovém vodiči jsou definovány vůči této společné zemi. Vzhledem k tomu, že šumová imunita signálů v úrovních TTL je velmi malá (v nejnepříznivějším případě jen 0,4 V) a navíc budiče TTL nejsou určeny k buzení delších vedení s parazitními kapacitami, jsou pro přenos po tomto rozhraní stanoveny jiné logické úrovně. Log.0 na výstupu vysílače odpovídá napětí +5 az + 15 V a log.1 odpovídají napětí -15 až -5V. Na vstupu přijímače je jako log.0 interpretován signál kladné polarity s napětím od +3 V a jako log.1 signál záporné polarity s napětím do -3 V. Tak je zaručeno, že šumová imunita bude minimálně 2 V. Chování přijímače v zakázaném pásmu -3 V až +3 V není ve standardu TIA/EIA 232 F definováno. U jednotlivých typů převodníků mezi úrovněmi TIA/EIA 232 F a TTL se liší. Většinou bývají prahy logických úrovní stanoveny tak, aby přijímač mohl správně zpracovat i signál v úrovních TTL. Například u velmi rozšířeného převodníku MAX 232 je jako log.0 vyhodnoceno již vstupní napětí větší než 2,4 V a jako log.1 napětí menší než 0,8 V. Vedle logických úrovní standard EIA/TIA 232 F dále specifikuje vstupní odpor přijímače, který by se měl pohybovat mezi $3 až 7\ k\Omega$.

Maximální délka vedení byla původně podle doporučení RS 232 C stanovena na

15 m. Vzhledem k tomu, že limitujícím faktorem však není ani tak délka vedení, ale jeho parazitní kapacita, nespecifikuje již současná varianta již přímo maximální délku vedení, ale stanoví, že maximální zatěžovací kapacita vysílače smí být nejvýše 2500 pF. Tato kapacita je součtem vstupní kapacity vysílače a kapacity vedení. Vstupní kapacita vysílače se zpravidla pohybuje kolem 20 pF, takže rozhodujícím vliv má kapacita vedení. Vzdálenost na níž bude možné komunikovat, bude tedy značně závislá na typu použitých kabelů. Při stejném uspořádání bude ještě záležet na tom, zda použitý kabel bude stíněný nebo nestíněný. Bude-li použit například nízkokapacitní nestíněný kabel, jehož vzájemná kapacita mezi vodiči je 40 pF/m, bude možné jej použít na maximální vzdálenost cca 40 m. Stíněný kabel se stejnou měrnou kapacitou však bude použitelný jen do vzálenosti cca 20 m.

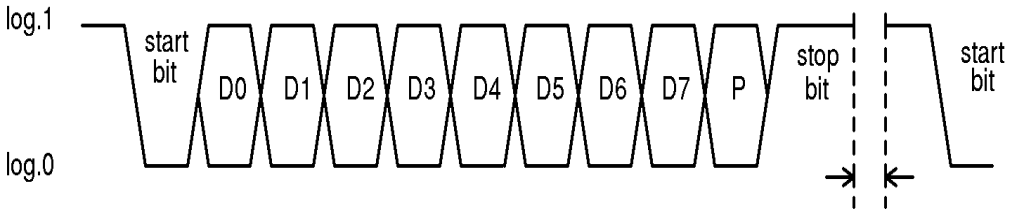
Na kapacitě vedení také závisí dosažitelná rychlost přenosu. Propojení přijímače a vysílače na sériové lince obvykle nezahrnuje pouze dva vodiče s případným stíněním, ale obsahuje ještě vodič pro přenos dat opačným směrem a většinou i další vodiče pro řídící signály. Proto bývá nejčastěji realizováno jako vícežilový kabel a zde se uplatní nejen kapacity a indukčnosti mezi jednotlivými signálovými vodiči. Vlivem těchto kapacitních a indukčnostních vazeb dochází k tzv. přeslechům. Při změně logické hodnoty signálu v jednom vodiči se v sousedních vodičích objevuje naindukované rušivé napětí. Přitom platí, že velikost tohoto rušivého napětí vzrůstá se strmostí náběžných a sestupných hran signálu. Z tohoto důvodu je maximální rychlost změny napětí na signálových vodičích normou TIA/EIA 232 F omezena na $30\,V/\mu s$.

Na druhou stranu však nelze strmost náběžných a sestupných hran příliš snižovat. Problémem je zakázané pásmo -3 V až +3 V, kde chování přijímače není normou specifikováno. Z tohoto důvodu je ve standardu TIA/EIA 232 F předepsáno, že průchod zakázaným pásmem nesmí být delší než 4% doby, která je vyhrazena pro přenos jednoho bitu při dané rychlosti.

Z předchozích požadavků tedy vychází teoretická maximální přenosová rychlost 200 kbit/s. Prakticky je však požitelná rychlost přenosu omezena tím, že mnohé komunikační programy stejně jako starší komunikační obvody nejsou schopny pracovat s vyšší přenosovou rychlostí než 115,2 kbit/s. Podstatnější je fakt, že vzhledem k nutnosti splnit požadavek na rychlost průchodu zakázaným pásmem do 4% časového úseku pro přenos jednoho bitu, docházíme při vyšších přenosových k tomu , že vedení, které se chová jako kondenzátor, je nutné nabít na jednu nebo druhou hodnotu napětí během krátké doby. Z toho je patrné, že má-li se změnit rychle, jsou potřebné nabíjecí proudy poměrně značné.

Jelikož maximální zkratové proudy, které jsou běžné budiče schopny poskytnout, nejsou nijak vysoké (10 mA pro MAX 232), proto nelze v podstatě na maximální vzdálenost (tzn. při maximální zatěžovací kapacitě 2500 pF) komunikovat rychlostí větší než 20 kbit/s. Z tohoto důvodu specifikuje ve své současné variantě TIA/EIA 232 F jako maximální přenosovou rychlost pouze 19,2 kbit/s. Pokud však použijeme kratší vedení, nebo vedení s nižší kapacitou, není problém komunikovat i rychlostmi vyššími. Např. při rychlosti 115,2 kbit/s je možno vcelku bez problémů komunikovat na vzdálenost do 3 m.

V rámci uvedených omezení je přenosová rychlost teoreticky libovolná, prakticky se však používají většinou ryhlosti ve standardní řadě tzn. 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600 a 19200 bit/s. Při rychlostech převyšující maximum doporučované normou se nejčastěji používají rychlosti 28800, 38400, 57600, 115200 bit/s.



Obr. 21: Struktura datového slova pro TIA/EIA 232 F

Toto rozhraní používá variantu asynchronního přenosu, při níž jsou nezávislé hodiny přijímače a vysílače vždy znovu synchronizovány při vysílání každého datového slova pomocí speciálních synchronizačních značek tzv. start a stop bitů. V klidovém stavu, kdy není vysíláno nic je na datové lince log.1. Stav datové linky je periodicky vzorkován s frekvencí, která je celočíselným násobkem přenosové rychlosti (zpravidla 16x nebo 64x). Pokud přijímač zjistí, že došlo ke změně stavu z log.1 na log.0, interpretuje to jako začátek start bitu, počká po dobu odpovídající polovině doby potřebné pro přenos 1 bitu a stav linky otestuje znovu. Pokud zjistí, že se vrátil do log.1 znamená to, že předchozí změna byla pouze náhodným šumem a nikoliv skutečným start bitem. Přijímač proto začne znovu pravidelně vzorkovat stav přenosové linky jako předtím a čeká na další přechod z log.1 do log.0. Jestliže však signál se po uplynutí poloviny doby vyhrazené pro přenos 1 bitu stále rovná log.0, jedná se pravděpodobně o skutečný start bit a přijímač začne testovat stav datové linky vždy po uplynutí jedné bitové periody. Tímto způsobem jsou postupně načteny hodnoty jednotlivých bitů a na konec je pak jednou nebo dvakrát testována hodnota stop bitu a podle výsledku tohoto testu se určí, zda je datové slovo bylo správně přeneseno, nebo došlo k tzv. chybě rámce (tzn. stop bity byly skutečně nějakým způsobem porušeny nebo jsou přijímač i vysílač nakonfigurovány na jiný počet stop bitů). Celé

vysílané slovo je tedy zahrnuto do rámce, který začíná jedním nulovým start bitem a zakončují jej volitelně jeden, jeden a půl nebo dva jedničkové stop bity viz obr.21. Varianta jeden a půl stop bitu znamená, že log.1 se na datové lince objeví po dobu odpovídající jeden a půl násobku času který je při zvolené přenosové rychlosti vymezen pro přenos jednoho bitu. Hlavním důvodem proč jsou stop bity vysílány je poskytnout přijímači čas, aby se mohlo připravit na přijetí dalšího slova. Používání většího počtu stop bitů má tedy význam jen u velmi pomalých zařízení jako jsou např. elektromechanické dálnopisy. Vysílané slovo může obsahovat 5 až 8 datových bitů a k tomu jeden paritní bit. Paritní bit přitom může být nastaven jedním z následujících způsobů:

a) **sudá parita –** bit je nastaven tak, aby celkový počet jedničkových bitů ve vysílaném slově včetně paritního bylo sudé číslo

b) **lichá parita -** bit je nastaven tak, aby celkový počet jedničkových bitů ve vysílaném slově včetně paritního bylo liché číslo.

c) nulová parita (space parity) **-** paritní bit je vždy log. 0.

d) **jedničková parita** (mark parity) – paritní bit je vždy log. 1.

e) **žádná parita** - paritní bit se nepoužívá

V praxi se používají, až na vyjímky, pouze varianty a) a b), které představují nejjednodušší formou zabezpečení přenosu dat. Dojde-li u jednoho bitu k chybě přenosu tzn. log.0 bude přijata jako log.1 nebo naopak, počet jedniček se změní podle nastaveného druhu parity ze sudého na lichý nebo opačně, pak tato situace bude vyhodnocena jako chyba parity. Toto je jednoduché zabezpečovací schéma však již není schopno odhalit 2 chyby v přenosu, neboť v tomto případě se parita nezmění. Stejně tak nedokáže rozeznat, zda došlo jen k jedné nebo k většímu, lichému počtu chyb a není také schopno chyby opravit. Má-li přenos proběhnout správně, je třeba nejen, aby přenosové rychlosti přijímače i vysílače byly nastaveny na stejnou hodnotu, ale také musí oba mít nastaven stejný počet datových bitů, stop bitů a stejnou paritu.

Při výše popsaném způsobu sériového přenosu dat je počátek vysílání datového slova asynchronní událostí, která může nastat kdykoliv bez vazby na jakýkoliv synchronizační signál. V rámci jednoho jednoho vysílaného rámce však již přenos probíhá synchronně. Celý přenos je proto možné chápat jako kombinaci asynchronního a synchronního přenosu. Někdy se tento způsob přenosu označuje jako **arytmický přenos**. K vysílání i příjmu datových slov po tomto rozhraní jsou zkonstruovány speciální obvody označované obvykle jako UART (Universal Asynchronous Receiver and Transmitter –

univerzální asynchronní přijímač a vysílač). Tyto obvody zajišťují obousměrný převod mezi sériovým a paralelním formátem dat, generováním a kontrolou ostatních prvků vysílaného rámce (start, stop bit, parita) a zpravidla části nebo i všech řídících signálů rozhraní.

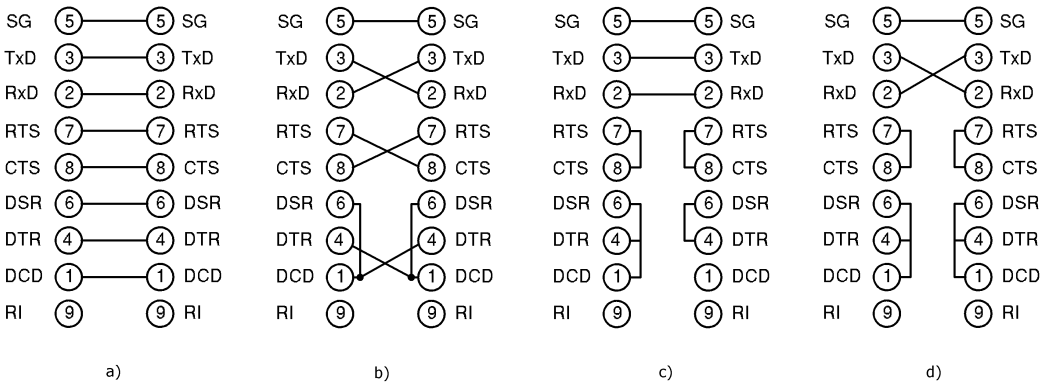
3.3 Řídící signály rozhraní TIA/EIA 232 F

TIA/EIA 232 F specifikuje kromě vlastních datových signálů také signály pro řízení přenosu dat. Jejich označení a funkce vycházejí z toho, že i když se toto rozhraní běžně používá v řadě nejrůznějších aplikací, jeho původním určení je přenos dat mezi počítačem a modemem nebo obecněji mezi koncovým zařízením pro přenos dat (DTE – Data Terminal Equipment) a zařízením ukončujícím datový okruh (DCE – Data Circuit-terminating Equipment resp. Data Communication Equipment). Jako spojovací prvek norma specifikuje 25ti kolíkový konektor CANNON DB-25. V praxi se však nejčastěji používá 9-ti kolíkový konektor CANNON, který obsahuje pouze nejdůležitější z řídících signálů. Signály, které se jsou dostupné pouze na 25-ti kolíkovém konektoru se používají jen velmi zřídka a běžné obvody typu UART s nimi nepracují. Existuje několik variant zapojení signálů. Tzv. malá varianta obsahuje pouze signály nezbytné pro obousměrnou komunikaci: signálovou zem a vysílaná a přijímaná data. Střední varianta obsahuje navíc další signály pro řízení přenosu viz tab.8. Jako velkou variantu by pak bylo možné označit kompletní soubor signálů dostupných na 25-ti kolíkovém konektoru. S ohledem na výše uvedenou definici logických úrovní TIA/EIA 232 F, pak aktivní úroveň odpovídá log.0 neboť jsou-li aktivní, je na příslušných vodičích kladné napětí +5 až +15 V. V případě, že využíváme všech signálů pro řízení přenosu dostupných ve střední variantě rozhraní (kromě RI, který se využívá jen velmi zřídka) a rozhraní je použito ke svému základnímu účelu, tedy propojení mezi DTE a DCE, je zapojení propojovacího kabelu velice jednoduché. Jak je vidět na obr. 22 a), stačí jednoduše spojit spolu vývody náležející stejným signálům na obou konektorech. Poněkud komplikovanější situace nastává, pokud chceme propojit tímto rozhraním dvě zařízení stejného typu (zpravidla DTE). V tomto případě je především nutné překřížit datové vodiče, protože jinak by obě zařízení vysílala data na stejný vodič. Další postup závisí na tom, jak je v dané aplikaci využíváno hardwarové řízení toku dat. K tomu, aby přenos mohl proběhnout musí obě zařízení DTE obdržet aktivní signály na vstupech DCD, DSR a CTS, přitom však mohou vysílat pouze dva signály RTS a DTR. Ve variantě na obr. 22 b) jsou proto vstupy DCD a DSR propojeny a aktivní hodnota signálu na nich je v případě, pokud druhé zařízení hlásí připravenost

signálem DTR. V tomto zapojení mají obě strany k dispozici 2 signály s jejichž pomocí mohou řídit průběh komunikace. Jeden z nich může být například použit k hlášení o připravenosti zařízení, druhý pak k vlastnímu řízení průběhu přenosu.

9 pin	25 pin	označení vývodu		význam signálu
vývod č.	vývod č.	TIA/EIA 232	z pohledu DTE	
5	7	SG		Signal Ground – signálová zem
3	2	TxD	Výstup	Transmitted Data – data, která vysílá DTE tzn.DCE naopak očekává na tomto vývodu přijímaná data. Aby DCE mohlo začít vysílat data musí být aktivní signály DSR, DTR, RTS a CTS.
2	3	RxD	Vstup	Received Data – data, která přijímá DTE tzn. DCE naopak na tomto vývodu data vysílá. Není-li aktivní signál DCD je RxD vždy v log.1
7	4	RTS	Výstup	Request To Send – požadavek na vysílání, DTE tímto signálem oznamuje, že je připraveno vysílat data do DCE
8	5	CTS	Vstup	Clear To Send – pohotovost k příjmu, DCE tímto signálem v odezvě na aktivní hodnotu signálu RTS oznamuje, že je schopno přijímat data
6	6	DSR	Vstup	Data Set Ready – signalizuje, že DCE je připojen a v pohotovosti
4	20	DTR	Výstup	Data Terminal Ready – signalizuje pohotovost DTE k příjmu nebo vysílání, pouze je-li tento signál aktivní může DCE hlásit pohotovost signálem DSR

Tab. 8: Význam a rozložení signálů malé a střední varianty RS 232 na konektoru CANNON 9 a 25



Obr. 22: Možnosti zapojení propojovacích kabelů pro střední a malou variantu rozhraní

V řadě případů jsou hardwarové řídící signály zbytečné a propojení pak lze zjednodušit. Vůbec nejjednodušší možné propojení je na obr. 22 d). Zde jsou propojeny pouze překřížené datové vodiče a signálová zem, u tohoto zapojení je ale nutné zabezpečit, že řídící signály budou mít správné logické hodnoty. Je tedy dátovými propojkami zajištěno, že vyšle-li jedno zařízení hlášení o připravenosti signálem DTR, obdrží správnou odpověď jako by tam byl rovněž připravený modem tzn. signály DSR a DCD budou aktivní a podobně se bude chovat i dvojice signálů CTS a RTS. V takovém případě je však

nutné, aby průběh přenosu byl řízen výhradně pomocí softwarových prostředků (např. pomocí dohodnutých znaků oznamující počátek a konec přenosu, případně potvrzující přijetí zprávy apod.). Varianty na obr. 22 b) a d) lze i zkombinovat a propojit signály RTS a CTS podle varianty b) a ostatní signály podle varianty d). V Tomto případě bude mít každá strana k dispozici jeden signál pro hardwarové řízení přenosu. Ve všech třech případech však spolu budou komunikovat dvě zařízení typu DTE pomocí vhodně zapojeného propojovacího kabelu. Toto zapojení propojovacího kabelu se někdy označuje termínem **null modem**.

3.4 Použitelnost TIA/EIA 232 F

Významnou výhodou tohoto rozhraní je jeho všeobecná rozšířenost a dostupnost, neboť je součástí téměř každého osobního počítače. Dále již následují spíše nevýhody. Rozhraní umožňuje komunikaci pouze relativně nízkými rychlostmi a na krátké vzdálenosti. Podstatný nedostatek je dán samotným elektrickým uspořádáním rozhraní, v němž je země vysílače propojena se zemí přijímače a logické úrovně jsou vyhodnocovány vůči této společné zemi, Vzhledem k tomu, že zemní potenciály zařízení připojených k různým větvím síťového rozvodu se můžou i dost výrazně lišit, mohou zemním vodičem jednak protékat vyrovnávací proudy předem neodhadnutelné velikosti a jednak tento posuv zemního potenciálu mezi vysílačem a přijímačem může vést k chybnému vyhodnocení logické hodnoty signálu a ve zvláště nepříznivém případě může vést až ke zničení obvodů rozhraní. Toto uspořádání také činí přenos dat poměrně málo odolným vůči rušení, neboť rušivé napětí, které se indukuje v signálovém vodiči se sčítá s užitečným napětím a je vyhodnocováno vzhledem k relativně stálému potenciálu společné země může tak být chybně vyhodnocena logická úroveň na vodiči. Proti tomuto jevu je samozřejmě možné se částečně chránit použitím stíněného kabelu, ale za cenu zkrácení maximální vzdálenosti, na kterou lze komunikovat nebo snížením maximálně dosažitelné přenosové rychlosti.

Vzhledem k výše uvedeným vlastnostem rozhraní se jeho použití více méně omezuje na laboratorní použití pro připojení externích měřících přístrojů, vstupně výstupních modulů k osobnímu počítači nebo na propojení které je realizováno pouze na krátkou vzdálenost a má spíše nastavovací či diagnostický charakter. Jako propojení, které by mělo obstát v trvalém nasazení v podmínkách průmyslového provozu je však zpravidla nepoužitelné a je tedy nutné použít vhodnější rozhraní (např. IEEE 422/485, CAN bus).

4. Software

4.1 Úvod

K popisu softweru jen tolik. Program pro snímání dat (měřící rozhraní) byl vyvinut v programu C++Builder 5. Pro komunikaci byla použita komponenta VaComm. Pro přenos dat není stanoven žádný protokol. Je však možné se dotázat mikrokontroléru, kolik je v externí paměti načtených dat. Tuto informaci je schopen odeslat do měřícího rozhraní, které jak při odesílání, tak při příjmu počítá odeslané byty nezávisle na mikrokontroléru. Lze tak porovnat, kolik dat bylo odesláno, a kolik jich mikrokontrolér obdržel. Komunikační rychlost je poměrně nízká, dochází tak k malé chybovosti. Veškeré požadavky a povolení vysílá měřící rozhraní. Mikrokontrolér pouze vyčká příkazu (povolení) a poté požadovanou činnost vykoná. Příslušný mód aplikace je určen dvojcí jumperů umístěných pod mikrokontrolérem. Po zvolení příslušného módu je nutné stisknout levé tlačítko, aby se aktivovala příslušná část programu v mikrokontroléru.

Celkový popis bude rozdělen do tří samostatnějších celků. Tyto celky spolu souvisí, ale lze je popsat i odděleně. U každého celku bude uveden vývojový diagram, počáteční nastavení regisrů (i proměnných) a vysvětlena činnost. Zmiňované celky jsou následující:

- Mód 0 : vzorkování průběhu napětí (bez jumperů)
- Mód 1 : komunikace mezi mikrokontrolérem a měř. rozhraním (nasazen jeden jumper)
- Mód 2 : jízda na data ze souboru pomocí PWM regulace (nasazeny oba jumpery)

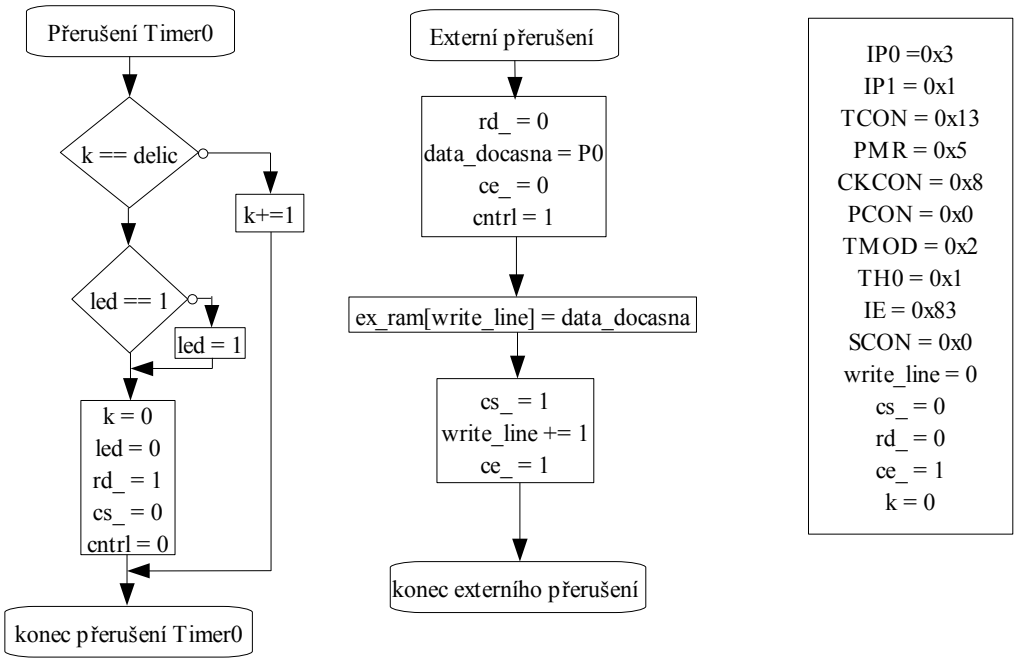
.

Parametry přenosu na RS 232:

- rychlost přenosu: 9600
- počet datových bitů: 8
- parita: žádná
- start bit: 1
- stop bit: 1

4.2 Mód 0 vzorkování průběhu napětí

Režim je určen dvojcí jumperů. Mód 0 je nastaven, když není nasazena ani jedna propojka. Popis režimu je následující. Timer0 generuje vzorkovací kmitočet tak, že neustále dochází k jeho „přetékání“ a vyvolávání přerušení. Při přechodu do obsluhy přerušení je testována proměnná (*k == delic)*. Je-li výraz vyhodnocen jako nepravdivý, provede se inkrementace proměnné *k* a návrat z přerušení do hlavního programu. Je-li však výraz vyhodnocen jako pravdivý, dojde k rozsvícení (zhasnutí) led diody v závislosti na předchozím stavu. Následně se odešle taktovací signál pro A/D převodník. Tím končí přerušení od Timeru0. A/D po obdržení taktovacího signálu sejme vzorek vstupního napětí a začne převádět hodnotu do číslicové podoby. Je-li převod dokončen, vyšle náběžnou hranu na pin BUSY, který je přes invertor připojen na pin P3.2 (externího přerušení). Přerušení je nastaveno na sestupnou hranu. Signál BUSY tedy vyvolá externí přerušení, zprůchodní se cesta směrem z A/D převodníku k portu P0, odkud si mikrokontrolér vyzvedne data. Uloží je na okamžik do proměnné *data_docasna.* Poté se aktivuje externí paměť pro zapsání dat, zprůchodní se cesta do externí paměti a data jsou do ní zapsána. Tím je ukončeno externí přerušení. Konstanta *delic* byla změřena experimentálně pomocí osciloskopu a jejím nastavením je možné volit vzorkovací kmitočty (15.15Hz,30.1Hz,45.4Hz), které se nastavují ve zdrojovém kódu. Přerušení od Timeru0 má nižší prioritu, než externí přerušení. Je to dáno nastavením registrů IP0 a IP1. Registr PMR povoluje používání externí datové paměti. Registr CKCON obsahuje nastavení časování sběrnice P0 a P2. Timer0 pracuje jako časovač s obvodovým přednastavením na hodnotu uloženou v TH0. Řízení Timeru0 je ryze softwarové, a to pomocí bitu TR0. Na obr.23 je znázorněn vývojový diagram módu 0 a počáteční nastavení registrů a proměnných.



Obr. 23: Vývojový diagram módu 0 a nastavení registrů a proměnných

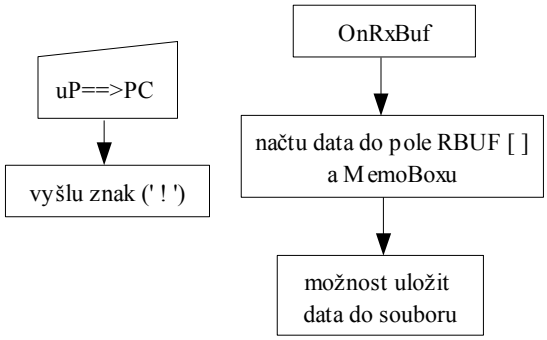
4.3 Mód 1 komunikace mezi mikrokontrolérem a měřícím rozhraním

Tento režim je nastaven, je-li nasazen jeden jumper. Z důvodu lepšího popisu je tato kapitola rozdělena na dvě části. V první se budeme zabývat vysíláním dat z mikrokontroléru do měřícího rozhraní. V druhé části pak popisem, kdy mikrokontrolér přijímá data z měřícího rozhraní. Nastavení registrů je jednotné.

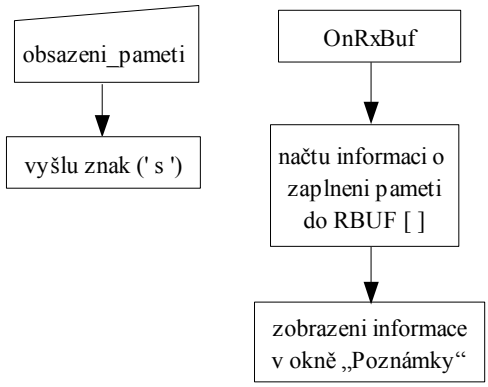
4.3.1 Vysílání dat mikrokontrolérem

Jako první si vysvětlíme činnost, kterou vykonává měřící rozhraní viz. obr.24. Je-li tedy v externí paměti uložen navzorkovaný průběh, potom při vybrání záložky *Prenos_dat* -> a volby *uP -> PC* vyšle rozhraní znak vykřičník ('!'). Poté se vyvolá událost OnRxBuf signalizující přijatá data. Všechna data jsou načtena do bufferu (alokované paměti). Následně se data zkopíruji do pole a zároveň do Memoboxu. Data je možno uložit do souboru, a také z nich vykreslit graf. Pro jednoduchou kontrolu počtu odeslaných dat lze v měřícím rozhraní zadat *Prenos_dat -> obsazení paměti* viz obr.25. Mikrokontrolér odešle 2 byty, které nesou informaci o velikosti zaplněné paměti..

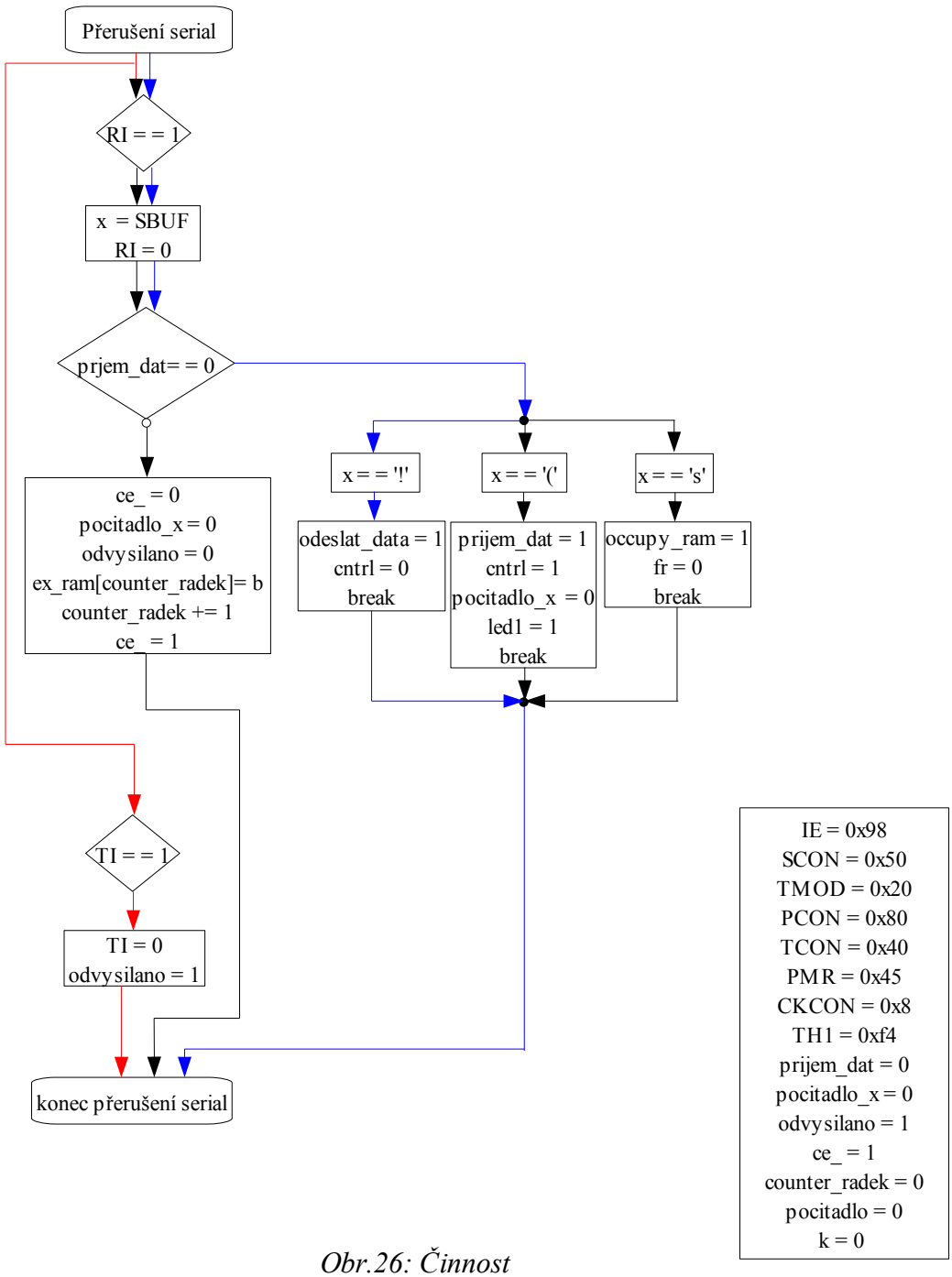
Činnost mikrokontroléru a počáteční nastavení registrů a proměnných je na obr.26 a obr.27. Timer1 generuje přenosovou rychlost pro sériový kanál dle vztahu $TH1 = 256 - ((2^{SMOD} f_{OSC})/(384\ přen.rychl))$ Po přijmutí znaku vykřičník ('!') se nastaví příznak *odeslat_data.* a provede se činnost dle modrých čar. Červené čáry ukazují činnost mikrokontroléru při vysílání s využitím přerušení od sériové linky, vyvolané příznakem TI, který říká, že vysílací buffer je prázdný. V obsluze přerušení se příznak TI vynuluje, a smí tak být odeslán další znak. Při každém načtení hodnoty z externí paměti se inkrementuje proměnná *pocitadlo,* která je porovnávána s proměnnou *write_line,* z čehož lze určit, kolik platných dat zbývá v paměti. Proměnná *write_line* je z módu 0 a udává, kolik bylo uloženo vzorků (obsazené řádky v paměti). Po skončení vysílání se musí vynulovat příznak *odeslat_data.* Zeleně je vyznačena činnost programu při požadavku na zaplnění paměti platnými daty. Na obr.28 jsou vývojové diagramy. Ty se liší pouze v tom, která z proměnných uchovává nenulovou hodnotu zaplněné paměti.



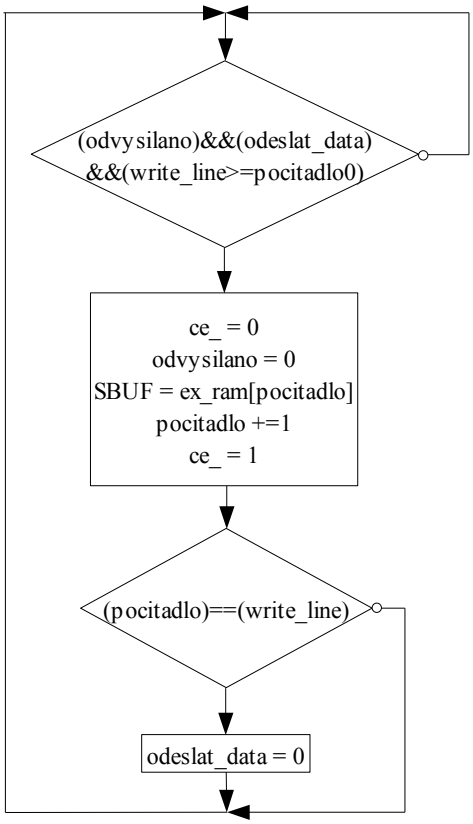
Obr.24: Činnost měřícího rozhraní při příjmu dat



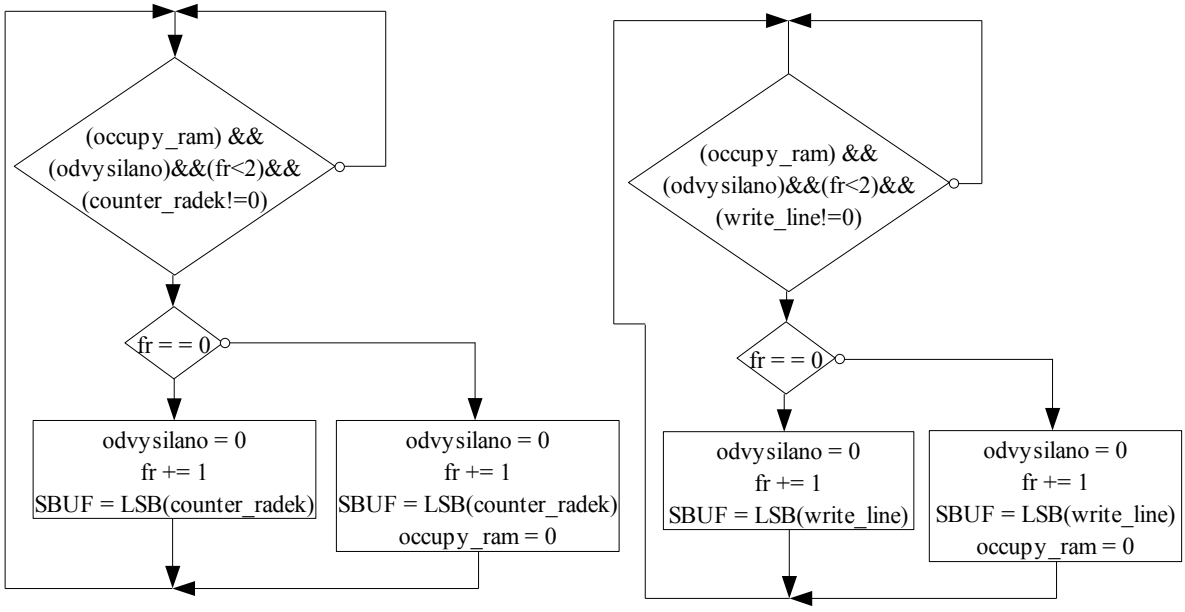
Obr.25: Činnost měřícího rozhraní při příjmu dotazu na zaplnění paměti



Obr.26: Činnost mikrokontroléru při vysílání dat a počáteční nastavení registrů a proměnných



Obr.27: Činnost hlavního programu při vysílání dat

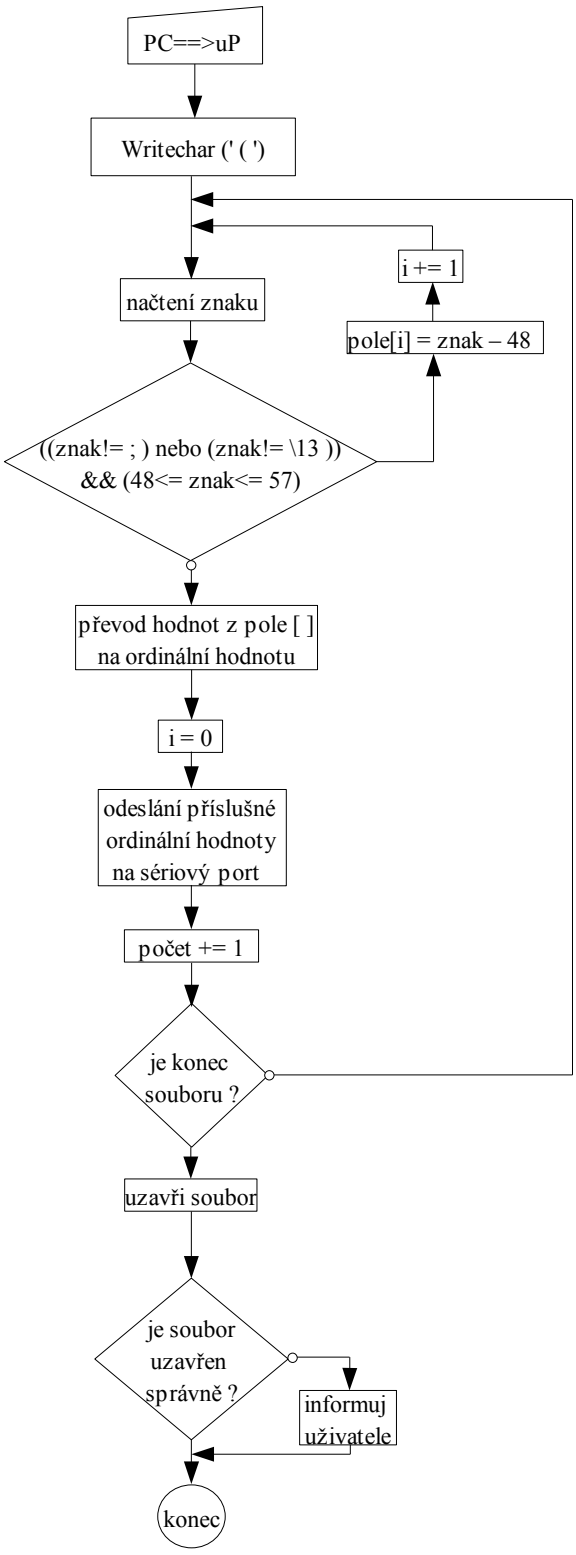


Obr.28: Činnost hlavního programu při vysílání stavu zaplnění paměti

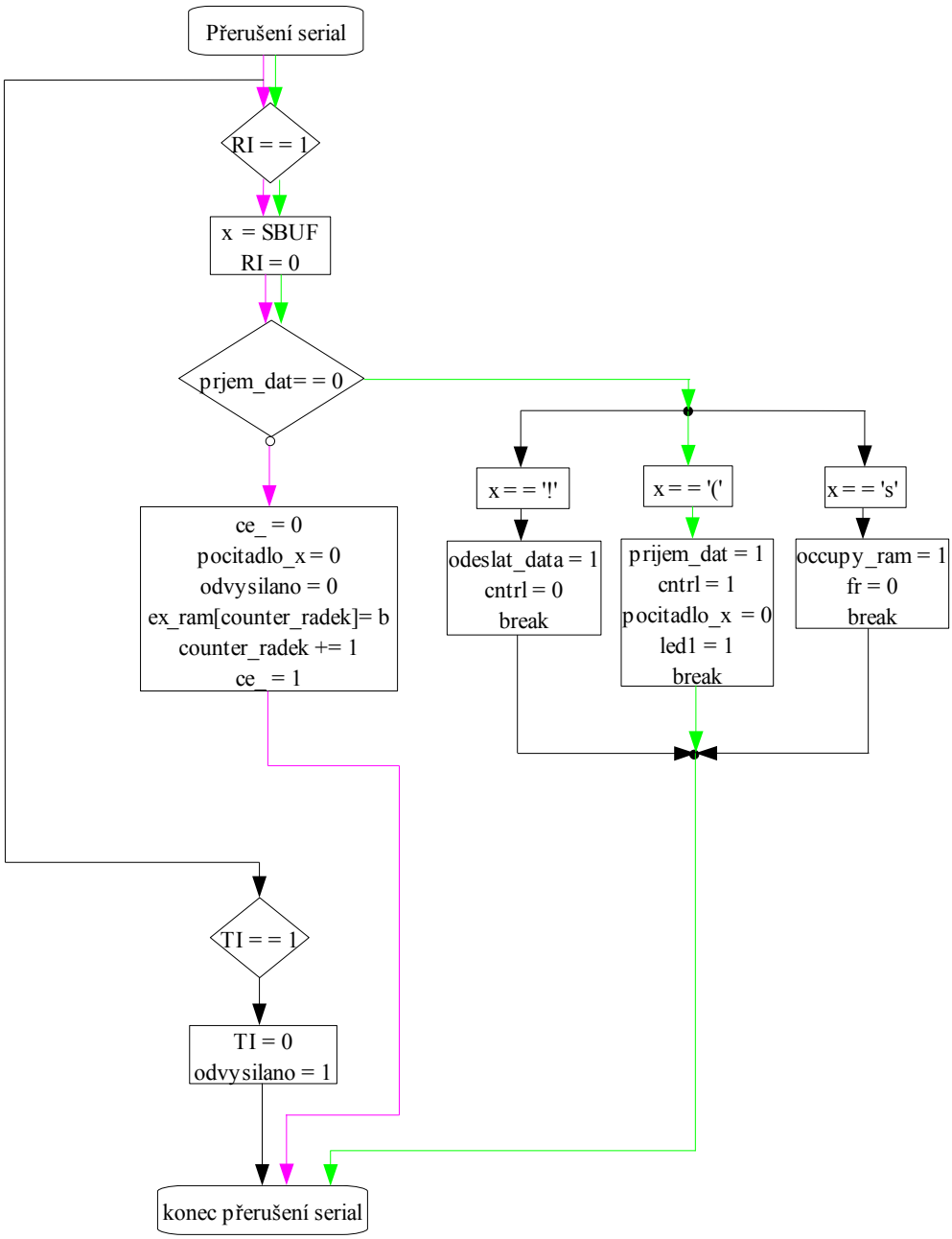
4.3.2 Přijímání dat mikrokontrolérem

Nejprve si opět osvětlíme činnost, kterou vyvíjí měřící rozhraní. V rozhraní se standartním „open dialogem“ vybere a načte příslušný soubor do MemoBoxu. Po zadání volby *Prenos_dat ->* a volby *PC-> uP* vyšle rozhraní znak levá kulatá závorka ('('), což je pro mikrokontrolér zpráva, že bude přijímat data a ukládat je do externí datové paměti. Příslušný vývojový digram je na obr.29. Aktuální soubor je otevřen pro čtení. Protože je každá hodnota zakončena znakem středník, načítá se řádek do doby, než se objeví středník. Mimo to se ještě kontroluje, zda-li je načítána pouze číslice. Je-li načtený znak vyhovující, odečte se od něho konstanta 48 a uloží se do proměnné *pole*[*i*] a inkrementuje *i.* V okamžiku načtení středníku se číslice načtené do proměnné *pole*[] převedou na příslušnou ordinální hodnotu a odešlou na sériové rozhraní. Přitom se ještě inkrementuje proměnná *pocet.* Toto se děje řádek po řádku, vždy ke znaku středník. V případě, že se na dalším řádku objeví znak konce souboru, se tento uzavře.

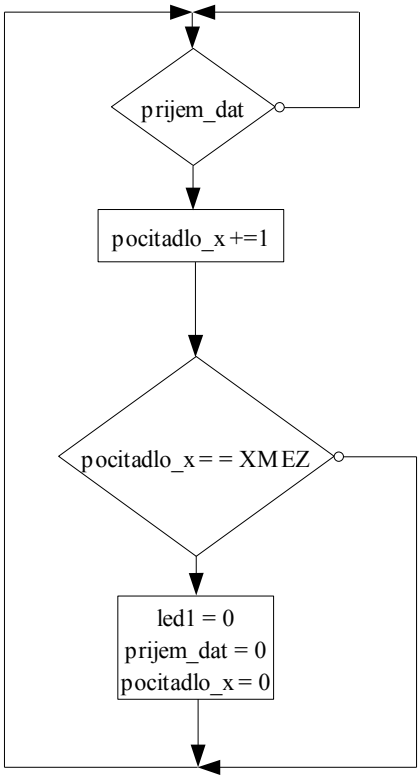
Popis činnosti na staně mikrokontroléru je následující. Timer1 opět generuje přenosovou rychlost. Při příjmu znaku ('(') je činnost vyznačena zeleně. Je vyvoláno přerušení díky nastavení příznaku RI (přijímací buffer je naplněn). V příslušné obsluze přerušení je nutno vynulovat příznak RI a nastavit příznak *přijem_ dat.* Takto mikrokontrolér pozná, že následná data budou ukládána do externí datové paměti. Růžově je vyznačena cesta, kudy bude digram procházen při příjmu obecného (ne řídícího) bytu. Při zápisu do externí paměti je inkrementována proměnná *counter_radek* a lze přijmout další znak. Proměnná *counter_radek* je použita v módu 2, aby byla uchována informace, kolik bylo zapsáno do paměti bytů a včas se zastavilo autíčko. Vývojový diagram činnosti mikrokontroléru je na obr.30. Na obr.31 je zakreslena činnost hlavního programu. V něm inkrementuje proměnnou *pocitadlo_x.* Neustále je testován výraz (*XMEZ==pocitadlo_x*). Bude-li pravdivý, potom nastal konec přenosu. Proměnná *pocitadlo_x* je totiž při příjmu každého znaku vynulována. Pouze když „dlouho“ není voláno přerušení, tak se „stihne“ nainkrementovat. Opět jako v předchozí kapitole se lze dotázat na stav zaplnění paměti. Popis na tuto činnost je uveden v předchozí kapitole a na obr.28.



Obr.29: Činnost měřícího rozhraní při vysílání dat na RS232



Obr.30: Činnost měřícího rozhraní při příjmu dat z RS 232



Obr.31: Činnost hlavního programu při příjmu dat z RS232

4.4 Mód 2 jízda na data z ext. pameti pomocí PWM regulace

Mód 3 je nastaven, když jsou nasazeny oba jumpery. Autíčko projíždí dráhou na data, která jsou uložena v externí paměti. Okamžitá rychlost je upravována PWM regulací, která je napájena ze zdroje +10V. Oba Timery pracují jako časovače s obvodovým přednastavením na hodnotu uloženou v TH0 (resp. TH1). Timer0 generuje frekvenci, se kterou budou data načítána z externí paměti. Rychlost by měla být stejná jako v módu 0, aby tak rychle, jak se data do paměti načítají, byla i čtena. Timer1 generuje frekvenci PWM regulace, která je oproti Timeru0 výšší. Timer1 má menší prioritu přerušení. Příslušné vývojové diagramy jsou na obr. 32. Podrobnější popis k činnosti Timeru0. Opět je pravidelně generováno přerušení. V následné obsluze se opět vyhodnotí výraz (*k*== *delic*). Nepravdivé vyhodnocení vede k inkrementaci proměnné *k* a opuštění přerušení. Pravdivé vyhodnocení vede k vynulování proměnné *k* a k rozsvícení (zhasnutí) led diody v závislosti na předchozím stavu a vyhodnotí se výraz

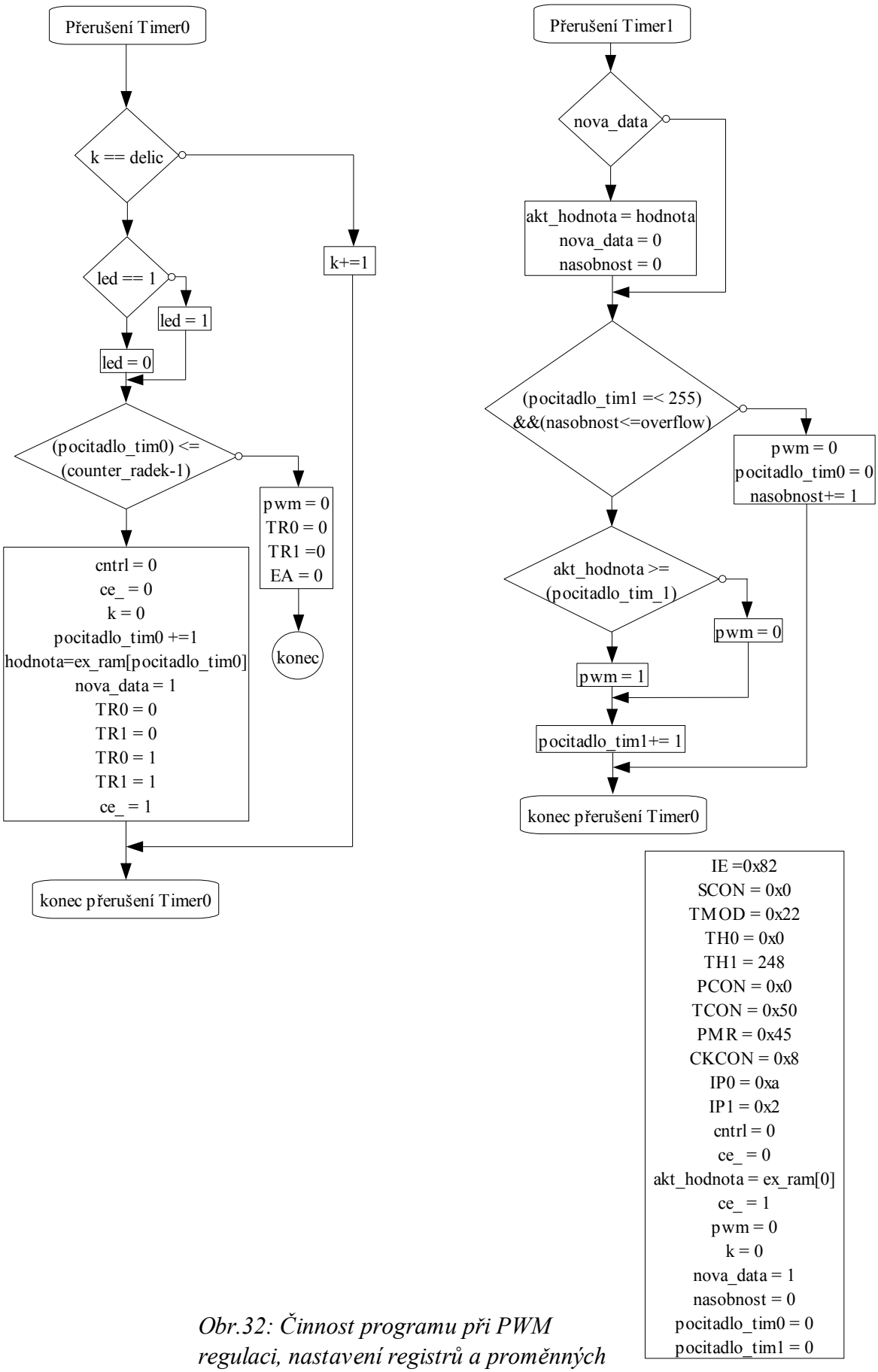
(*pocitadlo_tim0 =< counter_radek-1*). Testuje se zde, jestli jsou ještě v paměti platná data. Je-li výraz nepravdivý (v paměti již nejsou platná data), vypnou se oba Timery a motorek autíčka se odpojí od napájení (*pwm=0*). Pravdivé vyhodnocení vede na aktivování paměti a načtení nové hodnoty do proměnné *hodnota* se současným nastavením příznaku *nova_data.* Poté se vypnou a ihned zapnou oba Timery, aby se současně s nově načtenou hodnotou zesynchronizovaly. Timer1 také neustále „přetéká“ (avšak častěji) a vytváří tak frekvenci PWM regulace. Činnost se opět odehrává v přerušení a to nejprve vyhodnocením výrazu *nova_data.* Pravdivé vyhodnocení vede k načtení nové hodnoty, vynulování příznaků *nova_data* a *nasobnost.* Při nepravdivém vyhodnocení se výše uvedená sekvence nevykoná. Následně se otestuje výraz (*pocitadlo_tim1<=255*) && (*nasobnost<=overflow*). První z obou výrazů znamená, že se sekvence nachází uvnitř intervalu 0..255 (tomu odpovídá určitá střída a tím i úroveň napětí). Druhý výraz testuje, kolikrát se uvedená střída vykoná mezi dvěma vzorky. Hodnoty jsou ručně spočítány a uvedeny v tab.9. Nepravdivé vyhodnocení znamená, že je sekvence buď tesně před načtením nové hodnoty z paměti, nebo se právě vykonal interval 1. střídy a může se předchozí střída opakovat, avšak nejprve se musí inkrementovat proměnná *nasobnost*. Pravdivé vyhodnocení vede k dalšímu rozhodnutí. V něm se určí, v jaké části zmiňovaného 0..255 článkového intervalu se nacházím vůči proměnné *akt_hodnota*. Je-li (*akt_hodnota>=pocitadlo_tim1)* pravdivý, je motorek autíčka připojen na napájení (*pwm=1*). Nepravdivost výrazu způsobí odpojení motorku od napájení (*pwm=0*). Bez ohledu na vyhodnocení minulého výrazu se inkrementuje proměnná *pocitadlo_tim1*. Pro upřesnění je třeba uvést, že proměnná *nasobnost* se inkrementuje po 256ti přetečeních (vykonání jedné střídy) Timeru1.

F_{SAMPL} [Hz]	overflow = 6	overflow = 10	overflow =13	overflow = 20
15,15	TH1 = 176	TH1 = 208	TH1 = 219	TH1 =232

F_{SAMPL} [Hz]	overflow = 4	overflow = 8	overflow =16	overflow = 20
30,1	TH1 = 196	TH1 = 226	TH1 = 241	TH1 =244

F_{SAMPL} [Hz]	overflow = 2	overflow = 3	overflwow = 4
45,4	TH1 = 79	TH1 = 203	TH1 = 216

Tab. 9: Hodnoty proměnné overflov a registru TH1

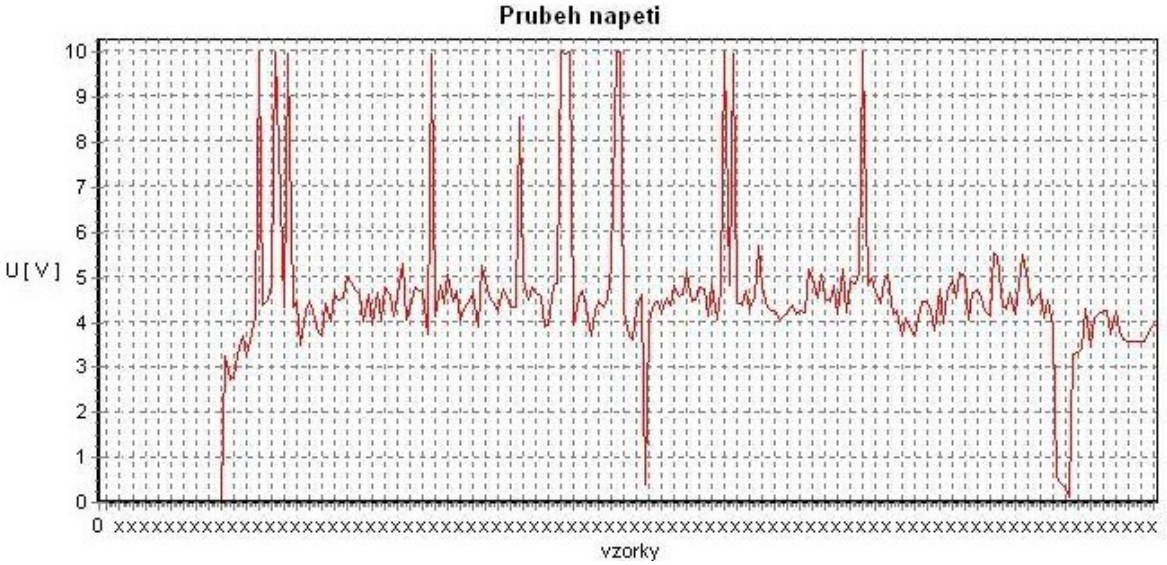


Obr.32: Činnost programu při PWM regulaci, nastavení registrů a proměnných

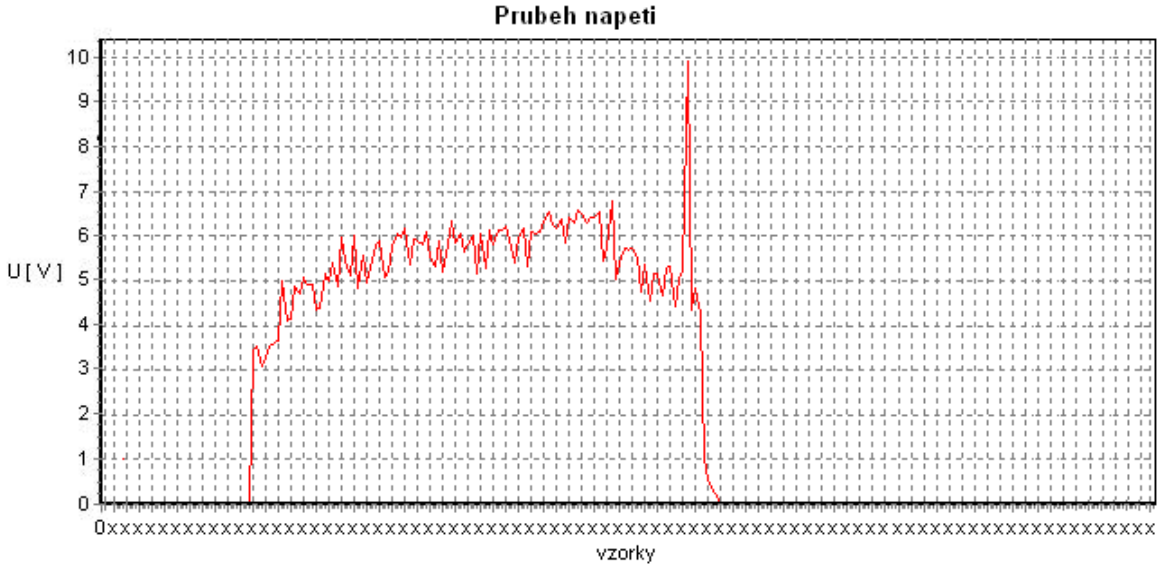
5. Zhodnocení experimentu

Testování celého zapojení bylo prováděno na autodráze s délkou jednoho okruhu vnějšího oválu 635cm. První se testoval mód 0 (snímání napětí na elektromotorku). Po projetí jednoho okruhu se data přenesla do měřícího rozhraní a uložila do souboru. Bylo vyzkoušeno použití různé vzorkovací frekvence. Tato část spolu s komunikací (a to obousměrně) pracovala bezchybně. Jiná situace ovšem nastala v módu 3, kdy mělo autíčko zopakovat dráhu, která byla předtím navzorkována. Zde již byly výsledky různorodé.
V zásadě je lze rozdělit do dvou kategorií. První z nich je případ, kdy autíčko projelo tu samou dráhu a zastavilo s malou tolerancí tam, odkud vyrazilo. Malou tolerancí rozumíme ± 10cm od místa startu. Druhým případem, ke kterému docházelo, byl přejezd navzorkované dráhy o několik desítek cm. V reakci na toto byl několikrát poupraven software v mikrokontroléru. Na zlepšení přesnosti dojezdu autíčka se to však nijak zřetelněji neprojevilo. Poté byl připojen přímo na analogový vstup A/D převodníku osciloskop, aby bylo možno vizuálně zjistit průběh signálu, který je snímán. Na obrazovce osciloskopu byl zobrazen signál, který byl značně zarušen. Toto rušení je způsobeno dvěma hlavními vlivy. Prvním je rušení, které vzniká na komutátoru. Toto rušení není svou amplitudou nikterak velké, protože jej pohlcuje filtrační kondenzátor, připojený přímo na kartáčky motorku. Frekvence rušení je závislá na rychlosti otáčení komutátoru. Druhý rušivý vliv vzniká při přejezdech spojů na dráze, které nejsou ideální. Vznikají poměrně vysoké napěťové špičky s nedefinovanou dobou trvání. O tvaru impulzu nelze říci nic bližšího, protože na téměř každém spoji je průběh jiný a nelze jej nějakým rozumným způsobem omezit, natož odstranit. Jedinou skutečností je, že čím horší bude provedení spoje, tím bude mít špička větší amplitudu a naopak. Na doporučení pana Ing. Martina Vlasáka bylo odzkoušeno nasnímaný průběh fitrovat několika jednoduchými způsoby, a to v měřícím rozhraní. První ze způsobů byl průměrovací filtr, a to čtvrtého, osmého a šestnáctého řádu. Ten měl za úkol vzniklé špičky zmenšit na úkor mírného zvednutí okolních hodnot. Řád filtru vypovídá o tom, kolik hodnot (budoucích) v okolí vzniklé špičky lze uvažovat. Výsledky této operace nebyly opět nikterak znatelné. Autíčko stále velmi často přejíždělo vzdálenost, která byla nasnímána, řádově i o několik desítek cm. Uvážíme-li, že délka testovacího okruhu byla 635cm, není výsledek příliš uspokojující. Poté se ještě zkoušelo provést u nasnímaného průběhu výběrový průměr. Ani tato filtrace příliš nepomohla. Na grafu 2 a grafu 3 jsou pro přehled uvedeny dva průběhy nasnímaného

napětí tak, jak byly vykresleny v měřícím rozhraním. Graf 2 zobrazuje průběh napětí na autíčku při velmi pomalé jízdě na jednom okruhu při vzorkovací frekvenci 30Hz. Graf 3 zobrazuje průběh běžné jízdy při vzorkování frekvencí 15Hz opět na jednom okruhu. Oba grafy ukazují závislost navzorkovaného napětí na čase.



Graf 2: Průběh napětí při velmi pomalé jízdě



Graf 3: Průběh napětí při průjezdu dráhy střední rychlostí

6. Závěr

V rámci bakalářské práce byl navržen způsob, jak zaznamenávat průběh jízdy autíčka na autodráze. Principem bylo snímání průběhu napětí na ovladači, který reguluje napětí pro elektromotorek. Činnost jednotlivých obvodů byla vcelku bezproblémová. Veškeré periférie spolu komunikovaly dle předpokladu, avšak celkový výsledek práce má (dle mého názoru) ještě určité rezervy.

Hlavním problémem bylo v původní teorii neuvažované poměrně silné rušení, způsobené mechanickými vlastnostmi celého systému. Z toho pramení vznik chyby, která celou práci, ať chceme, či nechceme znehodnocuje, protože převodník bohužel snímá kromě správných dat i tyto chyby, které jsou posléze ukládány do paměti, a skutečný signál je tak degradován.

Jedním z možných východisek, které by mohlo pomoci, je takto nasnímaný signál vyfiltrovat účinným číslicovým filtrem. To však již z časových důvodů nebylo možno provést.

Použitá literatura

[1] Arendáš, M. Ručka,M. Amatérské elektroniclé konstrukce.České Budějovice : KOOP, 1995. 101 s. ISBN 80-901342-7-0

[2] Hrbáček, Jiří. Komunikace mikrokontroléru s okolím. 1 vyd. 1.díl.Praha : BEN, 1999. 159 s. ISBN 80-86056-42-2

[3] Hrbáček, Jiří. Komunikace mikrokontroléru s okolím. 1 vyd. 2.díl.Praha : BEN, 2000. 152 s. ISBN 80-86056-73-2

[4] www.maxim-ic.com

[5] Ďaďo, S. Kreidl, M. Senzory a měřící obvody. Praha ČVUT, 1999. 315s. ISBN 80-01020-57-6

[6] GM Electronic Součástky pro elektroniku

[7] Hlava J. Prostředky automatického řízení II , 2000, skripta ČVUT

Přílohy

Aplikace vytvořená v C++ Builderu 5

Zdrojový kód mikrokontroléru

Aplikace vytvořená v C++ Builderu 5

Vytvořené měřící rozhraní je, co se ovládání týče, poměrně jednoduché. Hlavní menu tvoří tři základní nabídky. Jedná se o *Soubor, Zobrazit* a *Přenos_dat* viz obr.1. Nyní budou jednotlivé nabídky představeny.

Soubor:

Otevřít: otevře standartní okno známé z prostředí Windows pro otevírání souborů. V něm je možné vybrat soubor, který bude následně zobrazen v okně „Transmision data". O formátu tohoto souboru pojedná níže uvedená kapitola.

Uložit graf: výběrem této podnabídky se opět otevře dialogové okno tentokráte pro uložení obrázku. Zmiňovaňovaná podnabídka uloží právě zobrazený graf, a to jako rastrový obrázek.

Uložit: opět se otevře standartní okno pro ukládání textových souborů známé z prostředí Windows. Po zadání příslušného jména dojde k uložení do souboru obsahu okna „Recieve data". Automaticky je nastavena přípona **.txt,* která by neměla být měněna.

Konec: Ukončí běh měřícího rozhraní.

Zobrazit:

Stavový řádek: tato podnabídka poskytuje ještě další (vnořenou) nabídku. V ní je možno povolit (zakázat) zobrazování podrobnějších informací ve stavovém řádku o jednotlivých částech měřícího rozhraní.

Graf: výběrem položky dojde k vykreslení křivky napětí. Aby došlo k vykreslení křivky musí být v předchozím kroku buď data přijata od mikrokontroléru, nebo předchozí krok byl odeslání dat do mikrokontroléru. Zobrazit graf tedy lze jednak z přijatých dat (křivka červené barvy), nebo z dat odeslaných (graf má křivku modré barvy) .

Přenos_dat:

µP->PC: volba způsobí, že mikrokontrolér začne vysílat data do měřícího rozhraní resp. do okna „Recieve data".

PC ->µP: výběrem položky dojde k odeslání dat z okna „Transmision data" na sériové rozhraní.

Obsazení_pameti: Je-li aplikace v módu1, tak po zadání této volby mikrokontrolér

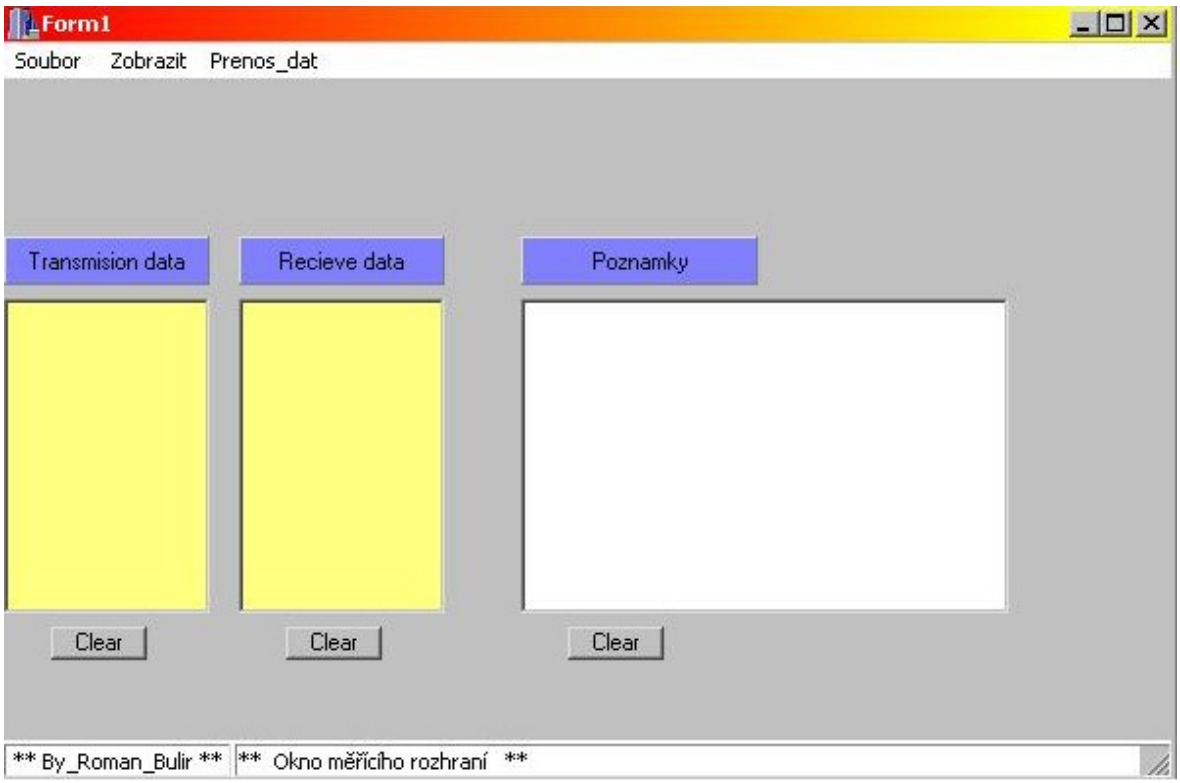
odešle informaci o počtu aktuálně obaszených řádcích externí paměti. Ta se zobrazí v okně „Poznámky“.

Nyní bude vysvětlen postup, a to jak pro načtení, tak i pro odeslání dat. Předpokládá se, že bude připojeno napájecí napětí (min.+15V) a zvolen příslušný mód. Pro aktivaci příslušného módu je nutné nejprve nasadit (sejmout) jeden nebo oba jumpery a poté stlačit levé tlačítko pod mikrokontrolérem.

1.Příjem dat do měřícího rozhraní: předpokládáme, že předchozí mód byl mód 0, kde se vzorkoval průběh napětí na elektromotorku autíčka a paměť má zapsaná platná data. Poté byl nasazen jeden (jedno který) jumper a stlačeno levé tlačítko. V měřícím rozhraní zadáme *Přenos_dat* a zvolíme *µP->PC.* V okně „Recieve data“ by se měla objevit data (čísla od 0 do 255) odeslaná mikrokontrolérem. Zadáním volby *Přenos_dat* a podnabídky *Obsazení_pameti* mikrokontrolér odešle počet obsazených řádků v paměti. Informace se objeví v okně „Poznámky“. Proběhla-li tato operace úspěšně je možno zadat v nabídce *Zobrazit* volbu *Graf*, načež by mělo dojít k vykreslení průběhu napětí (červenou barvou), tak, jak bylo nasnímáno na elektromotorku. Graf je možno uložit pomocí *Soubor* a nabídky *Uložit garf*. Obsah okna „Recieve data“ lze uložit následovně *Soubor* a nabídky *Uložit.*

2.Odeslání dat do externí paměti: měl by být aktivní mód 1 (nasazen jeden ze dvou jumperů). Jestliže se v předchozím kroku odesílala data do měřícího rozhraní pak je mód 1 stále aktivní a není tak nutné jej aktivovat tlačítkem. V měřícím rozhraní je postup následující. Z nabídky *Soubor* vybereme *Otevřít* a vybereme zvolený soubor. Mělo by dojít k načtení do okna „Transmision data“. Jestliže se tak stalo, můžeme zadat *Přenos_dat* a vybrat *PC->µP.* Tím dojde k odeslání dat z příslušného souboru do externí paměti. Okno „Poznámky“ by mělo automaticky zobrazit informaci o počtu odeslaných bytů. Opět je možno dotázat se mikrokontroléru na stav zaplnění paměti pomocí *Přenos_dat* a zvolením položky *Obsazení_pameti.* Příslušná informace je zobrazena v okně „Poznámky“. Další možností je zobrazení grafu , a toopět z nabídky *Zobrazit* položku *Graf.* Vykreslený napěťový průběh je zobrazen modrou barvou.

Formát souboru: soubor musí obsahovat příponu **.txt.* Data jsou v souboru uložena tak, že na každém řádku je jedna hodnota zakončená středníkem a znakem '\n' (Enter) . Hodnotou v tomto případě rozumíme celé číslo od 0 do 255.



obr. 1: Vzhled měřícího rozhraní

Zdrojový kód mikrokontroléru

```
#include <8051.h>
#include<Reg420.h>
#define  P1  0x90
#define  P2  0xa0
#define  P3  0xb0
#define XMEZ 0xffffU
#define DELIC 0x597U
//////////////////////////////////////////////////
// (0x1a9 = 19.6 ms = timer0 = 51.02 Hz) presne //
// (0x2d2 = 33.2 ms = timer0 = 30.1 Hz) presne  //
// (0x168U  = 16.6ms = timer0 = 60.20Hz) presne //
//(0x1ddU = 22ms = timer0 = 45.4 Hz) presne     //
//(0x597U = 66ms = timer0 = 15.15 Hz) presne    //
//////////////////////////////////////////////////
#define UNS_CHAR_MAX 0xff
#define UNS_INT_MAX  0xffff
// komunikacni rychlost 115200 kBaudu  TH1 =  0xff
// komunikacni rychlost 9600 kBaudu  TH1 =  0xf4

xdata unsigned char  ex_ram [UNS_INT_MAX] ;  // deklarace externi datove pameti
sbit at          P3+4       led ;   // signalizacni ledka
sbit at          P1+0       cntrl ;  // ovladani 74125
sbit at          P3+5       ce_; // signal pro ext. pamet
sbit at          P1+6       ax; // konfigurace aplikace
sbit at          P1+7       ay; // konfigurace aplikace
sbit at          P1+5       tl1; // leve tlacitko na desce
sbit at          P3+3       tl2; // prave tlacitko na desce
sbit at          P1+1       cs_;   // ovladani prevodniku
sbit at          P1+2       rd_;    // ovladani prevodniku
sbit at          P1+3       pwm; // ovladani tranzistoru pwm = 1 sepnuto
sbit at          P1+4       led1; // signalizacni ledka
unsigned char prijem,mod_aplikace,odvysilano;
unsigned char  nova_data,nasobnost,overflow;
unsigned int counter_radek,pocitadlo,pocitadlo_x=0,write_line ;
unsigned char hodnota,akt_hodnota,fr=0,occupy_ram,odeslat_data,prijem_dat,x;
unsigned int k , pocitadlo_tim0,pocitadlo_tim1;
unsigned char low_byte,high_byte,data_docasna;



void timer1 (void) interrupt 3    //timer 1 preruseni tady je docela ficak
{
```

```
 if(nova_data) //byla nabrana nova hodnota z EXTERNI RAMKY
   {
   akt_hodnota = hodnota;
   nova_data = 0;
   nasobnost = 0;
   }

 if(( pocitadlo_tim1 < UNS_CHAR_MAX )&&( nasobnost <= overflow))
    {
    if(akt_hodnota > pocitadlo_tim1 )
      pwm = 1;
     else
      pwm = 0;
    pocitadlo_tim1 += 1;// pruchody timerem 1
    }
   else
    {
    pwm = 0; // 255. pruchod vzdz sundat do log 0
    pocitadlo_tim1 = 0;
    nasobnost +=1 ;
    }
} // konec preruseni timeru 1

void timer0 (void) interrupt 1
{
  if( k == DELIC)
    {
    k = 0;
    if(led)
      led = 0; // blika ledkou
     else
      led = 1;
    if(mod_aplikace == 3) // sber dat do pameti
      {
       rd_ = 1; // takt pro A/D prevodnik
       cntrl = 0; // zpruchodnit cestu do uP
      }
     else//(mod_aplikace == 0)  // jizda na PWM regulaci
      {
       if( pocitadlo_tim0 <= (counter_radek-1) ) // nabere novou hodnotu z EX_RAMKY
         {
          cntrl = 0;
          pocitadlo_tim0 +=1;
```

```
          hodnota = ex_ram[pocitadlo_tim0];
          nova_data = 1; // signalizuje nabrani nove hodnoty
          TR0 = 0;
          TR1 = 0;
          TR0 = 1; // synchronizace timeru
          TR1 = 1;
          goto preskoc;
         }
        else
         {
          pwm = 0;
          led = 0;
          TR0 = 0;// vypnu oba Timery
          TR1 = 0;
          EA = 0;
          pwm = 0;  // vypnu veskera preruseni
         }
      }  // konec else ////
    }  //KONEC   if(k == DELIC)
  else
    {
     k+=1;
    }
  preskoc:
}   //konec PREUSENI timer 0

void prerus0 (void) interrupt 0   //externiho preruseni /INT0
{
 rd_ = 0;
 data_docasna = P0;  // A/D ma pripravena data, ktera jsou ulozena do docasne promenne
 ce_ = 0;  //aktivace pameti
 cntrl = 1; // zpruchodni cestu do pameti
 ex_ram[write_line] = data_docasna;
 write_line+=1;
 ce_ = 1;
}   // konec externiho preruseni /INT0

void main(void)
{
 zpet:
 if((ax == 0)&&(ay == 0))   //pwm regulace
  mod_aplikace = 0;         // oba jumpery nasazeny
 if(((ax == 0)&&(ay == 1)))  //uP --> PC nebo PC --> uP
```

```
 mod_aplikace = 1;         // jeden z jumperu nasazen
if(((ax == 1)&&(ay == 0)))
 mod_aplikace = 2;
if((ax == 1)&&(ay == 1))   //sber dat
 mod_aplikace = 3;        // jumpery nenasazeny
////////////////////////////////////////////////////////////////////////////////////////
 if(mod_aplikace == 3)   //sber dat a ukladani do pameti
  {
   IP0 = 0xb; // Res LPS1 LPT2 LPS0    LPT1 LPX1 LPT0 LPX0
   IP1 = 0x1; // Res MPS1 MPT2 MPS0    MPT1 MPX1 MPT0 MPX0
   TCON =  0x13;// TF1 TR1 TF0 TR0   IE1 IT1 IE0 IT0
   PMR = 0x5 ; // puvodne0x45 CD1 CD0 SWB CTM 4x/2x ALEON DME1 DME0
   CKCON = 0x08  ;// WD1 WD0 T2M T1M T0M MD2 MD1 MD0
   PCON = 0x0 ; //SMOD SMOD0 OFDF OFDE GF1 GF0 STOP IDLE
   TMOD = 0x2 ; //  1 GATE C//T M1 MO   0 GATE C//T M1 MO
   TH0 = 0x1; // pedvolba od ktere se zacina citat
   IE = 0x83;     //   EA ES1 ET2 ES0   ET1 EX1 ET0 EX0
   SCON =0x0;     // SM0 SM1 SM2 REN TB8 RB8 TI RI
   write_line = 0;  // aktualni pozice RAMKY na kterou se bude zapisovat
   counter_radek=0;
   cs_ = 0;
   rd_ = 0;
   ce_ = 1;
   k = 0;
  }
 if(( mod_aplikace == 1)||(mod_aplikace == 2)) //Nastaveni parametru komunikace
  {
  prijem_dat=0;
  pocitadlo_x = 0;
  odvysilano = 1;
  ce_ = 1;             // funkcni aplikace
  counter_radek = 0;  // pozor na posazeni jumperu
  pocitadlo= 0;
  led = 1;
  TH1 = 0xf4; // predvolba pro generovani prenosove rychlosti
  IE = 0x98;  // 1001 1000  EA ES1 ET2 ES0 ET1 EX1 ET0 EX0
  SCON = 0x50 ; // 0000 0000 SM0 SM1 SM2 REN TB8 RB8 TI RI
  TMOD = 0x20 ; //  1 GATE C//T M1 MO   0 GATE C//T M1 MO
  PCON = 0x80 ; //SMOD SMOD0 OFDF OFDE GF1 GF0 STOP IDLE
  TCON =  0x40;// TF1 TR1 TF0 TR0 IE1 IT1 IE0 IT0
  PMR = 0x45 ; // puvodne CD1 CD0 SWB CTM 4x/2x ALEON DME1 DME0
  CKCON = 0x08  ;// WD1 WD0 T2M T1M T0M MD2 MD1 MD0
  }
```

```
 if( mod_aplikace == 0)  // jizda na PWM regulaci
  {
  write_line=0;
  nova_data = 1;
  nasobnost = 0;
  pocitadlo_tim0 = 0; // pocitadlo prochodu timerem 0 (nacitani z pameti )
  pocitadlo_tim1 = 0 ; // pocitadlo prochodu timerem 1
  IE = 0x8a;  // 1000 1000  EA ES1 ET2 ES0   ET1 EX1 ET0 EX0
  SCON = 0x00 ; // 0000 0000 SM0 SM1 SM2 REN TB8 RB8 TI RI
  TMOD = 0x22 ; //  1 GATE C//T M1 MO   0 GATE C//T M1 MO
  TH0 = 0x0; // pedvolba od ktere se zacina citat ajko u mod_aplikace = 3 sber dat
  TH1 = 248; // 4x pretece timer1 a timer1 pretece 1
  PCON = 0x00 ; //SMOD SMOD0 OFDF OFDE GF1 GF0 STOP IDLE
  TCON =  0x50;// TF1 TR1 TF0 TR0   IE1 IT1 IE0 IT0
  PMR = 0x45 ; // puvodne CD1 CD0 SWB CTM 4x/2x ALEON DME1 DME0
  CKCON = 0x08 ;// WD1 WD0 T2M T1M T0M MD2 MD1 MD0
  IP0 = 0xa ; //  Res LPS1 LPT2 LPS0    LPT1 LPX1 LPT0 LPX0  timer 0 vyssi priorita
  IP1 = 0x2; //  Res MPS1 MPT2 MPS0    MPT1 MPX1 MPT0 MPX0  timer 1 nizsi priorita
  cntrl = 0; //zpruchodni cestu z pameti k procesoru
  ce_ = 0;
  k = 0;
  akt_hodnota = ex_ram[0];// prirazeni prvniho vzorku
  ce_ = 1;
  pwm = 0;
  led1=1;
  }
////////////////////////////////////////////////////////////////////////////////////////////////
while (1)
  {
//////////////////////////////////////////////////////////////
//vysilani dat do pocitace
    if((odvysilano)&&(odeslat_data)&&(pocitadlo<=write_line))
      {
       ce_ = 0;
       odvysilano =0;
       SBUF = ex_ram[pocitadlo];
       pocitadlo+=1;
       ce_ = 1;
       if(pocitadlo==write_line)
         odeslat_data=1;
      }
//////////////////////////////////////////////////////////////
// odeslani informace o poctu navzorkovanych hodnot ///////////
```

```
    if((occupy_ram)&&(odvysilano)&&(fr<2)&&(write_line!=0))
     {  // uP->PC
      if(fr==0)     // fungujici
        {     //lowbyte
         odvysilano = 0;
         fr+=1;
         SBUF =(unsigned char)(write_line);
        }
       else
        {       //highbyte
         odvysilano = 0;
         fr+=1;
         occupy_ram = 0;
         SBUF = (unsigned char)(write_line>>8);
        }
     }
    if((occupy_ram)&&(odvysilano)&&(fr<2)&&(counter_radek!=0))
     {
      if(fr==0)     // fungujici
        {
         odvysilano = 0;
         fr+=1;  //lowbyte
         SBUF =(unsigned char)(counter_radek);
        }
       else
        {
         odvysilano = 0;
         fr+=1;
         occupy_ram = 0;   //highbyte
         SBUF = (unsigned char)(counter_radek>>8);
        }
     }
////////////////////////////////////////////////////////////////
    if(!tl1)
     goto zpet;

    if(prijem_dat)
     pocitadlo_x+=1;

    if(pocitadlo_x==XMEZ)
     {
      led1=0;
      prijem_dat = 0;
```

```
        pocitadlo_x=0;
       }

   } // konec WHILE (1)
}   // konec procedury MAIN
//////////////////////////////////////////////////////////////////////////
void serial(void) interrupt 4
  {
    if( RI )
     { // ano byl prijat cely bajt
       x = SBUF;
       RI = 0;

     if(prijem_dat==0)
       {
        switch(x){
         case '!':{ odeslat_data = 1; cntrl = 0; pocitadlo=0;   break; }
         case '(' :{ prijem_dat = 1 ; write_line=0; cntrl = 1; pocitadlo_x=0; break; }
         case 's' :{ occupy_ram = 1; fr=0; break; }
                   }       //konec switch
       }
      else //prijem dat ze seriove linky
       {
        ce_ = 0;
        pocitadlo_x=0;
        ex_ram[counter_radek] = x;
        ce_ =1;
        counter_radek +=1;
       }
    }// konec if(RI)

    if(TI)
     {
      TI = 0;
      odvysilano = 1;
     }
 }
///////////////// konec serioveho preruseni//////////////////////////////////
```