ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

TECHNICKÁ FAKULTA

Záznamové zařízení pro rozhodčí vodního slalomu

Diplomová práce

VEDOUCÍ PRÁCE: DOC. ING. PAPEŽOVÁ, CSC.

AUTOR: BC. DAVID KYSELA

PRAHA 2014

CESKA ZEMEDELSKA UNIVERZITA V PRAZE

Katedra elektrotechniky a automatizace

Technická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Kysela David

Informační a řídící technika v agropotravinářském komplexu

Název práce

Záznamové zařízení pro rozhodčí vodního slalomu

Anglický název

Recording equipment for referees of the canoe slalom

Cíle práce

Návrh a realizace záznamového zařízení pro rozhodčí vodního slalomu

Metodika

1. Seznamte se se základními vlastnostmi dostupných mikropočítačů a vyberte vhodný obvod
2. Dle požadavků záznamového zařízení zvolte vhodné ovládací a zobrazovací prvky a způsob komunikace pro přenos dat
3. Navrhněte obvodové schéma systému a systém realizujte
4. Systém vybavte potřebným řídicím programem

Osnova práce

1. Obecný popis mikropočítače pro aplikaci v systému
2. Volba mikropočítače a periferních obvodů mikropočítače s ohledem na požadavky systému pro rozhodčí vodního slalomu
3. Hardwarová realizace systému
4. Softwarové vybavení systému

Rozsah textové části
40 - 50 stran

Klíčová slova
sběr dat, vyhodnocování sportovních soutěží

Doporučené zdroje informací

Brtník Bohumil, Matoušek David: Programování mikrokontrolérů s jádrem 8051 v jazyce C, BEN – technická literatura, 2010
Alexandrescu, Andrej: Moderní programování v C++ návrhové vzory a generické programování v praxi, Brno, Computer Press, 2004
Pravidla ČSKDV, http://www.kanoe.cz/

Vedoucí práce
Papežová Stanislava, doc. Ing., CSc.

Termín zadání
listopad 2012

Termín odevzdání
duben 2014

prof. Ing. Jaromír Volf, DrSc.
Vedoucí katedry

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE
TECHNICKÁ FAKULTA
-3-

prof. Ing. Vladimír Jurča, CSc.
Děkan fakulty

Čestné prohlášení:

Já, David Kysela, student Technické fakulty ČZU v Praze prohlašuji, že jsem svou diplomovou práci vypracoval samostatně pod vedením doc. ing. Papežové, Csc. a veškeré materiály, z nichž jsem čerpal pro svoji práci, jsou uvedeny v seznamu literatury.

V Praze dne: …… podpis: ……

Poděkování:

Rád bych poděkoval vedoucí práce doc. ing. Papežové, Csc. za poskytnuté informace a odbornou konzultaci.

Abstrakt

Cílem této diplomové práce je návrh záznamového zařízení pro rozhodčí vodního slalomu, dále jen záznamové zařízeni, které by mělo umožňovat zápis hodnocení průjezdů slalomových bran na daném úseku pro jednotlivé závodníky a online přenos těchto záznamů do počítače.

V úvodu zmiňují motivaci na konstrukci tohoto zařízení a také požadavky, které budou na toto zařízení kladeny.

V teoretické části se zabývám architekturou mikropočítačů a obecným popisem mikropočítače řady 8051.

V praktické části již navrhuji a popisuji konkrétní schéma zapojení a provedení záznamového zařízení. A popisuji hlavní myšlenku programu pro mikropočítač a také program samotný.

Klíčová slova: sběr dat, vyhodnocování sportovních soutěží, mikropočítač

Summary

The goal of this diploma work is a design of a Recording Equipment for Referees of the Canoe Slalom, later in the text just Recording Equipment which should be able to record assessments of passing through slalom gates in a given section for individual competitors and online transfer of these records to the computer.

In the introduction I speak about the motivation for the construction of the equipment and then about the requirements which will be posed on this equipment.

In the theoretic part I deal with the architectue of microcontrolers and with the general description of a microcontroler serie 8051.

In the practical part I propose and describe a particular scheme of connection and how to make a record device. I describe the main idea of the programme for the microcontroler as well as the programme itself.

Key words: colekting data, evaluation sports competition, microcontroler

Úvod

Myšlenka na online přenos trestných bodů ve vodním slalomu mě provází již od doby, kdy jsme si do oddílu vodního slalomu TJ Kralupy nad Vltavou, dále jen oddílu, pořídili profesionální časomíru REI2 od firmy Microgate, tedy již asi 6 let. Tato časomíra totiž umožňuje přenos naměřených časů do počítače. Firma Microgate pro přenos dat od rozhodčích používá intercom. Po vyzkoušení intercomu jsme ale jeho používání zavrhli, neb nepatrné zrychlení přenosu výsledku, bylo vykoupeno neúměrným zatížením obsluhy intercomu, kterou si zajišťoval počtář. Počtářem je rozuměna osoba zpracovávající výsledky závodu. A tak jsme se vrátili zpět ke sběru papírových lístečků a jejich následnému ručnímu přepisováni počtářem do počítače. K myšlence na zautomatizování vyhodnocování výsledků jednotlivých závodníků jsem se vrátil asi před třemi lety, po té co jsem byl v oddíle jmenován zástupcem počtáře a ono nezábavné přepisování dat do počítače občas dolehlo na mne samotného. Po předchozích zkušenostech bylo ale jasné, že přenos trestných bodů musí probíhat digitálně stejně jako je tomu u přenosu naměřených času z časomíry. Proto jsem byl rád, když se paní doc. ing. Papežové, Csc. na přednáškách vestavěných mikroprocesorových systémů zalíbil můj nápad na záznamové zařízení pro rozhodčí vodního slalomu a vypsala jej jako zadání pro diplomovou práci, které jsem si tedy vybral i přes to, že má znalost mikropočítačů zahrnovala pouze obsah několika prvních přednášek tohoto předmětu.

Cílem závodu ve vodním slalomu na divoké vodě je zdolat rychlý úsek řeky, či umělé slalomové dráhy, vymezený brankami, bez trestných bodů a v co možná nejkratším čase. Dle pravidel kanoistiky na divokých vodách [22] je maximální počet branek vymezující závodní trať 25 a minimálně musí mít závodní trať alespoň 18 branek a „*délka trati nesmí být kratší než 200 metrů, měřeno od startovní do cílové linie (měřeno střednicí řeky), a doporučuje se, aby maximální délka nebyla více než 400 metrů.*“[22] Průjezdy brankami na břehu sledují brankoví rozhodčí. Místu, odkud rozhodčí hodnotí přidělený úsek závodní trati, se říká post. Postů bývá obvykle kolem 7 a tak na každý post v průměru připadají 3 branky, ale rozložení branek po trati nemusí být zrovna symetrické a na různých postech může být dosti odlišný, ale počtu 5 dosahují jen výjimečně a to jen na veřejných závodech, neb na závodech vyššího významu je počet 5 branek na jeden post, brán již jako velké zatížení rozhodčích, z kterého by mohli pramenit chybná rozhodnutí. Záměrně jsem napsal post a ne rozhodčí, protože na jednom postu bývají zpravidla dva rozhodčí. Pokud závodník projede závodní branku správně, nezískává od rozhodčích žádné trestné body, v případě kdy závodník projede závodní branku, ale jakoukoliv částí své výstroje se dotkne kterékoli brankové tyče, získává od rozhodčího 2 trestné body, které mu v koncovém výsledku k jeho cílovému času přičtou 2 sekundy. Pokud ale závodník závodní branku neprojede vůbec, nebo ve špatném pořadí či směru získá od rozhodčích za tuto branku 50 trestných bodů a tedy i 50 sekund ke svému výslednému času. Chyby pří průjezdu se nesčítají, tedy pokud závodník branku neprojede, ale dotkne se jí, získává jen 50 trestných bodů. Rozhodčí je dle pravidel povinen vést o jednotlivých průjezdech podrobný záznam.

Na závodech ve vodním slalomu se soupeří v kategoriích kajak jednotlivci ženy a muži (tedy K1ž a K1m), kanoe jednotlivci ženy a muži (tedy C1m a C1ž), kanoe dvojic ženy a muži (tedy C2ž a C2m) a navíc každá ze zmíněné kategorie se může jet i jako závod družstev, takzvané hlídky, kdy závodí tým skládající se ze tří lodí. Nejvyšší počet závodníků v jedné kategorii, který jsem za svou závodní kariéru zaznamenal, byl 211, ale na většině závodů tento počet spíše představuje počet všech startujících. Toto je hrubý, ale z mého

pohledu dostačující popis základní problematiky, provázející vznik záznamového zařízení. Podrobnější popis výše popsaného je obsahem Pravidel kanoistiky na divokých vodách [22].

Z výše popsaného vyplývají požadavky na záznamové zařízení. Zařízení bude používáno ve volné přírodě a mělo by proto být odolné proti povětrnostním podmínkám, zejména pak proti stříkající vodě. Zařízení musí nadále umožnit zadání sledovaného startovního čísla a k němu přiřadit hodnocení průjezdů odpovídajících závodních bran a tento záznam následně přenést do PC ke zpracování výsledků. Vzhledem k potřebě přenosu dat do počítače bude i záznamové zařízení nějakým typem počítače a s přihlédnutím na provoz, v ne zrovna optimálních klimatických podmínkách pro počítač, půjde s největší pravděpodobností o zařízení, obsahující mikropočítač, který má oproti klasickému PC nižší nároky na napájení a chlazení.

1. Obecný popis mikropočítače v systému

1.1. Vývoj mikropočítačů

Pro výrobu mikropočítače bylo nejdříve potřeba zvládnout výrobu mikroprocesoru. Impulzem k vývoji mikroprocesoru byla roku 1969 objednávka japonské firmy Busicom, která vyráběla kalkulačky a u právě vzniklé firmy Intel si objednala speciální řešení integrovaných obvodů pro programovatelné kalkulátory. Firma Intel tímto úkolem pověřila Marciana E. Holfa, který záhy zjistil, že pro řešení daného problému, v té době standardním řešením, by na čip musel umístit přes padesát tisíc tranzistorů, ale to v té době bylo ještě nerealizovatelné. Holf při hledání náhradního řešení se nechal inspirovat uspořádáním počítače DEC, na kterém právě zadanou úlohu řešil. Vsadil do jednoho integrovaného obvodu operační jednotku a řadič, do druhého integrovaného obvodu pak vsadil paměť s programem pro daný kalkulátor, který říkal mikroprocesoru, co má dělat. Oba integrované obvody následně propojil sběrnicí, která zajišťovala i vstupy a výstupy dat. Tímto vznikl první mikroprocesor 4004, který byl univerzální součástkou a tím odpadl zdlouhavý vývoj individuálních integrovaných obvodů, jejichž chytrost se dala využít jen pro aplikaci, pro kterou byly určeny. Naproti tomu chování mikroprocesoru určoval jeho program, který mu říkal, zda mikroprocesor bude řídit výše zmíněnou kalkulačku anebo třeba například pračku.

Dnes již víme, že vznik 4 bitového mikroprocesoru 4004 byl milníkem ve vývoji elektroniky, ale jeho nasazení v programovatelných kalkulátorech tehdejší veřejnost nijak zvlášť neoslovilo. U odborné části populace to bylo ale jiné a mikrokontroler 4004 pro svou univerzálnost se rozšířil do jiných oborů a dodnes řídí například vesmírnou sondu Pioneer 10. Od výroby prvního mikrokontroleru šel vývoj rychle kupředu. Firma Intel již v roce 1972 přišla s prvním 8 bitovým mikroprocesorem s označením 8008. Byl sice pomalejší, něž jeho 4 bitový předchůdce 4004, ale to, že 8008 zpracovával celých 8 bitů, mohl tím přistupovat k mnohem větší paměti RAM, což z něj ve výsledku dělalo 3-krát až 4-krát rychlejší procesor než jeho předchůdce. Na procesoru 8008 bylo založeno již několik osobních počítačů, ale opravdový bum v tomto odvětví nastal v roce 1974 s mikroprocesorem 8080, který zlidověl pod označením BOBO. 8080 se stal oblíbeným zejména proto, že v roce 1975 v časopise Popular Elektronics vyšla stavebnice osobního počítače Altai s tímto procesorem. Mikroprocesor 8080 nebyl ale stále monolitickým mikropočítačem. Prvním monolitickým mikropočítačem byl v roce 1976 8048. Tento mikropočítač měl integrovanou paměť jak pro program, tak pro data, pracovní frekvenci 2MHz a obsahoval 6000 tranzistorů a byl primárně určen pro vestavné systémy. Zde se cesty osobních počítačů tedy mikroprocesorů pro osobní počítače a mikropočítačů pro vestavné systémy rozešly. Mikroprocesory pro PC pokračovaly

směrem vzrůstajícího výpočetního výkonu a paměťového prostoru a mikropočítače pro vestavné systémy šly spíše cestou vysoké spolehlivosti, zachování kompaktních rozměrů a nízké spotřeby. Tím ale nechci říci, že by se výpočetní výkon a paměťový prostor mikropočítačů vůbec nezlepšoval. Již o čtyři roky později přišla firma Intel s vylepšením mikropočítačů a vznikl populární mikropočítač 8051, který se stal základem Široké rodiny MCS-51, která se pro své vlastnosti, kterými jsou jednoduchost a univerzálnost, používá v různých modifikacích dodnes. V současné době je vyráběn více jak dvaceti výrobci, mezi nejvýznamnější patří společnosti Atmel, Infineon (dříve Siemens), NXP (dříve Philips), Silicon Laboratories (dříve Cygnal) a Texas Instruments.

V této době vznikaly i další typy mikropočítačů. Vzhledem k podobné době svého vzniku, tedy 70 léta 20 století, mají i některé podobné rysy, kterými je například nízký počet akumulátorů. MCS-51 obsahují pouze jeden 8 bitový akumulátor stejně jako mikropočítače řady PIC od společnosti Microchip Technogy a mikrokontrolery řady 6800 od společnosti Motorola. Řada 6800 měla dva 8 bitové akumulátory, které se v případě potřeby dají spřáhnout do jednoho 16 bitového akumulátoru.

Dalším pokrokem ve vývoji mikropočítačů je v polovině 90 let minulého století vznik mikrokontrolérů s podstatně větším počtem akumulátorů, než mají mikropočítače ze sedmdesátých let. Typickým představitelem těchto mikropočítačů je řada AVR, kde se jako akumulátor může chovat všech 32 registrů i když u všech mikropočítačů rodiny AVR nejsou obsazeny všechny banky registrů. Mikropočítače úspornějších řad mají obsazenu pouze horní polovinu registrových bank, tedy registry 16 až 32. Větší počet univerzálních registrů lépe vyhovuje vyšším programovacím jazykům tím, že odpadají mnohdy zdlouhavé přesuny dat do akumulátoru.

1.2. Architektura mikropočítače

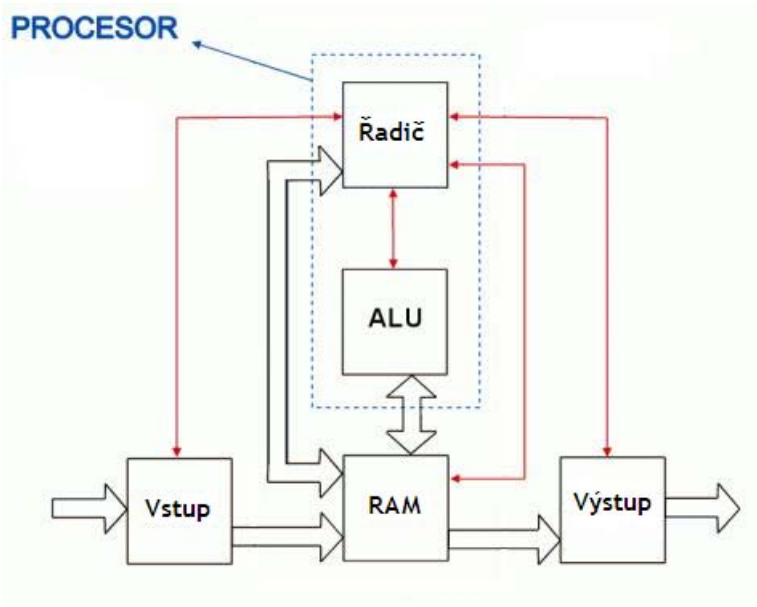
V současné době je na trhu téměř nepřeberné množství různých typů mikropočítačů, lišících se od sebe jak svou výbavou, tak hlavně svou architekturou. Architektura mikropočítače udává jeho výkonnost.

Nejpoužívanějším rozlišováním mikropočítačů, a to i mezi laickou veřejností, je jejich dělení podle šířky použité sběrnice. Podle šířky sběrnice dělíme na 8 bitové, 16 bitové, 32 bitové a 64 bitové. I přes trend stále se rozšiřující sběrnice je dle [9] poptávka po 8 a 16 bitových mikropočítačích stále představuje 80 % trhu. Přesto, že jde o starší údaj, s kterým pravděpodobně v poslední době hodně zahýbala poptávka po 32 bitových mikropočítačích, vzhledem k nárůstu obliby digitálních fotoaparátů, chytrých televizorů a telefonů. Domnívám se, že tento údaj má stále svou vypovídající hodnotu o využívání mikropočítačů v řídících aplikacích. Potvrzuje výše uvedené tvrzení, že při výběru mikropočítače pro danou aplikaci hraje velkou roli poměr ceny k potřebnému výkonu pro danou aplikaci.

Pro návrh mikropočítačů se používá architektura von Neumanova nebo Harwardská.

1.2.1 Von Neumanova architektura

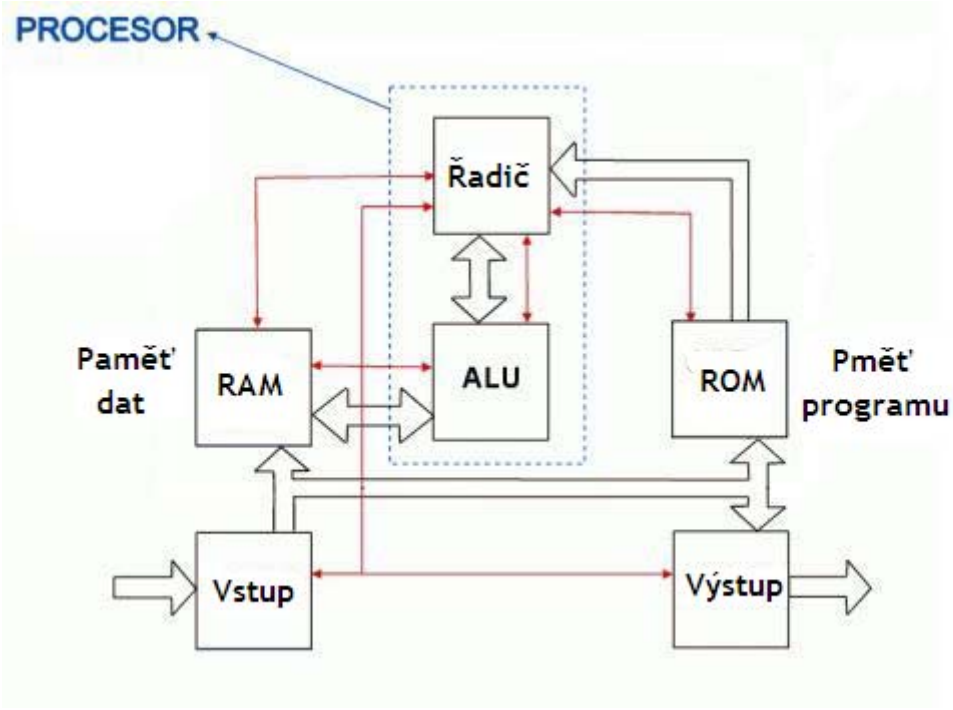
Základním rysem von Neumanovy architektury je, že data i program používají stejnou paměť a tím i stejnou sběrnici pro přenos dat i programových instrukcí. Výhodou je hospodárné využití sběrnicových systémů, ale na úkor rychlosti zpracování.



Obrázek 1 von Neumanova architektura zdroj [27][9]

1.2.2 Harwardská architektura

Harvardská architektura, která je z obou těchto architektur starší, se vyznačuje odděleným paměťovým prostorem pro data a program. Díky tomu je možné se současně obracet na data i příkazy, což je výchozím předpokladem pro vysokou rychlost zpracování.



Obrázek 2 Harwardská architektura mikropočítače zdroj [27][9]

Při oddělené paměti dat a programů mohou mít data a programové instrukce rozdílnou bitovou šířku. Tohoto využívají například již výše zmiňované mikropočítače PIC, které používají šířku instrukce 12, 14 či 16 bitů nebo od jiného výrobce procesor F21, který používal dvacetibitovou instrukci.

Největší zastoupení v dostupných mikropočítačích má modifikace Hardvardské architektury, kdy je mikropočítač interně navržen podle Harvardské architektury a externě pro přístup k vnější paměti podle von Neumanovy architektury. Toto provedení je hospodárnější na počet potřebných vývodů na pouzdře mikropočítače. To snižuje náklady na výrobu mikroprocesoru i jeho konečnou velikost.

1.3. Instrukční soubor

Další charakteristikou mikropočítače je rozsah zpracovávaných příkazů a počet hodinových cyklů na jeden příkaz, toto je rozhodující pro druh interpretu příkazu, jestli RISC nebo CISC, po případě CISP.

1.3.1. CISC (Complex Instruction Set Computer)

Jde o kompletní instrukční sadu příkazu, jejichž počet může být větší než 200. Je to tradiční interpret příkazů, zpracovávající mnoho výkonných příkazů, takže jediný příkaz umožňuje realizovat i složité zpracování dat. Využíváním velkého počtu instrukcí vzniká kratší programový kód, který pro uložení potřebuje menší prostor v paměti. Nevýhodou velkého počtu instrukcí je, že potřebují složitý interpret příkazů, který daný příkaz zpracovává během několika hodinových cyklů mikropočítače. I kompilátory C mají větší problém s překladem většího počtu příkazů a ve výsledku testů, zabývajících se tímto problémem, bylo zjištěno, že nakonec okolo 50% instrukcí nevyužíjí.

1.3.2. RISC (Reduced Instruction Set Computer)

Po zkušenostech s CISC se přešlo k architektuře RISC, tedy silně redukované sadě instrukcí, vzhledem k tomu, že složitou instrukci lze rozložit na posloupnost jednodušších kroků, realizovanými jednoduššími příkazy. Kde všechny příkazy mají stejnou bitovou šířku, jsou méně složité, a tedy nepotřebují tak složitý interpret příkazu jako architektura CISC. Tím je možno příkazy efektivněji dekódovat a zpracovávat, zpravidla v jednom hodinovém cyklu mikropočítače. Ale program zapsaný větším počtem jednodušších příkazů zas logicky zabírá větší paměťový prostor.

1.3.3. CISP (Cofigurable Instruction Set Processor)

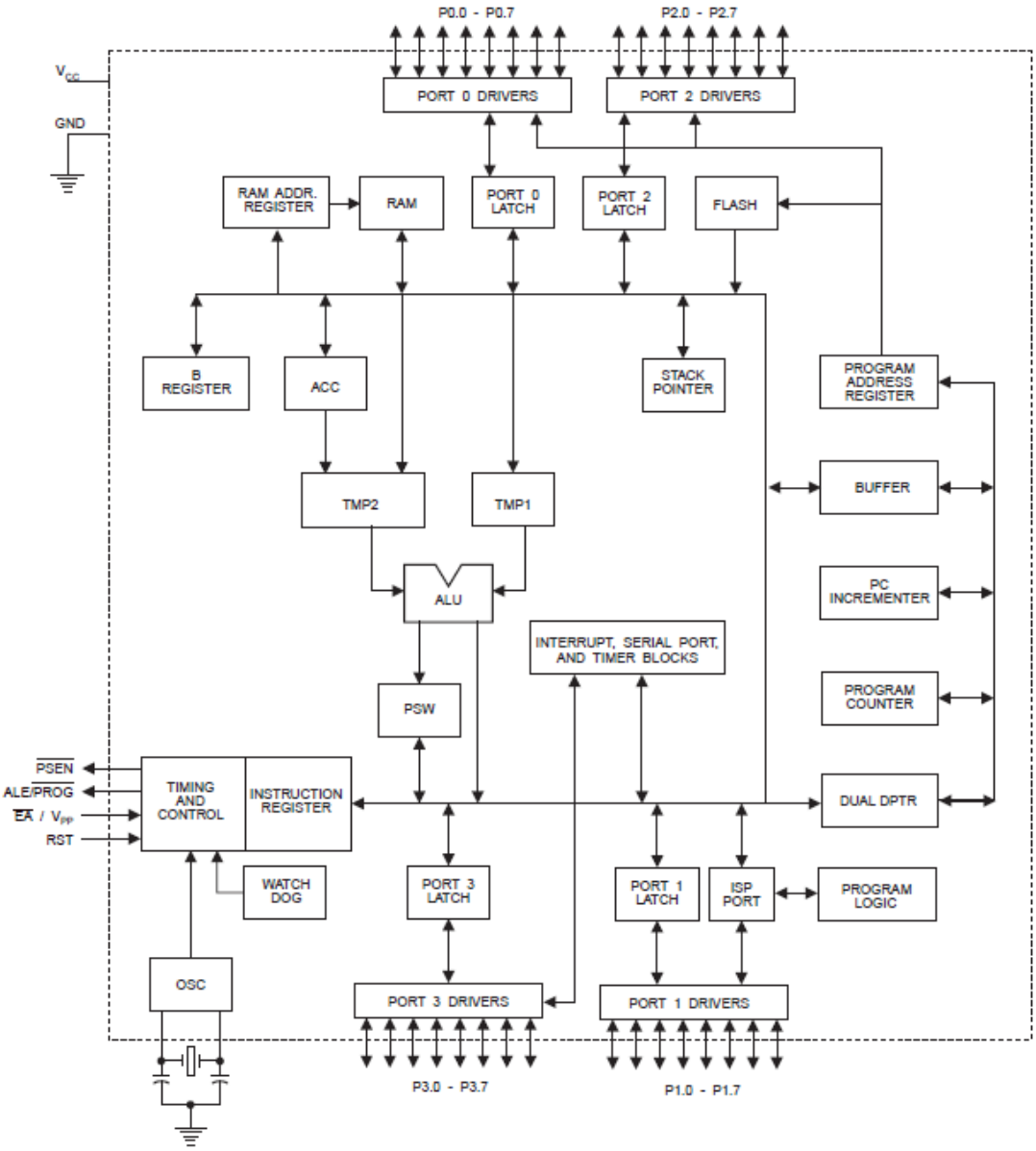
Tato architektura se využívá spíše od 32 bitových mikropočítačů výše, "*pro aplikace pracující v reálném čase a jiné náročné aplikace. Interpret příkazů, příkazová sada a architektura registrů jsou volně konfigurovatelné ze strany uživatele. V závislosti na požadavcích aplikace, na tom, jedná-li se o multimedia, DPS nebo jiné použití, může uživatel rozšiřovat základní architekturu aritmetickými jednotkami, datovými paměťmi X/Y, metodou circular buffer addressing atp.*„ [9] Tuto architekturu například používá procesorové jádro CORE od ARC Cores.

1.4.Obecný popis mikropočítačů MCS-51

1.4.1. Charakteristické vlastnostmi:

- 8 bitový procesor (CPU)
- oscilátor na čipu
- vnitřní paměť programu až 4 kB (u vylepšené řady 8052 až 8 kB)
- až 64 kB vnější paměti programu.
- vnitřní paměť´pro data RAM (většinou 128 B +128 B SFR u řady 8052 256 B + 128 B SFR – Speciální Funkční Registr)
- až 64 kB vnější paměti dat
- 32 vstupně výstupních linek
- dva 16 bitové čítače/časovače (u 8052 tří 16 bitové čítače/časovače)
- 5 zdrojů přerušení (u 8052 šest zdrojů přerušení)
- plně duplexní sériové rozhraní
- booleovský procesor
- výkonný soubor 111 instrukcí
    - 1bytových - 49
    - 2bytových - 45
    - 3bytových - 17
- jednotné napájecí napětí 5V
- plně kompatibilní s TTL obvody

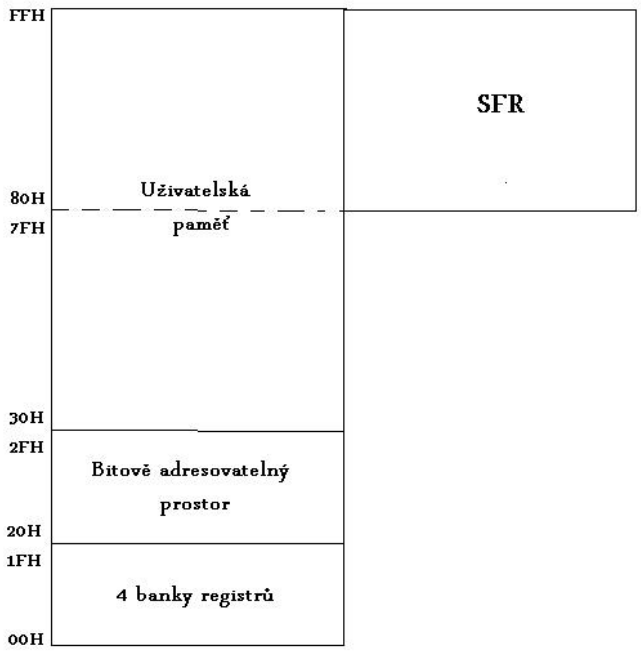
1.4.2. Blokové schéma mikropočítače z rodiny MCS-51



Obrázek 3 Blokové schéma mikropočítače zdroj [7]

1.4.2.1. Datová paměť RAM

Jak jsem již uvedl, mikropočítače řady 8051 jsou navrženy podle harvardské architektury, tedy má oddělený prostor pamětí pro data a program. V této části se budu věnovat rozdělení datové paměti, které je znázorněno na obr. 4.



Obrázek 4 Rozdělení vnitřní datové paměti [15]

Jak je z obr. 4. patrné uživatel má k dispozici 256 Bajtů vnitřní datové pamětí a systém se dá rozšířit o 64 kB vnější pamětí, která je adresovatelná přes šestnáctibitový registr DPTR, nebo přes registry R0 a R1 ve stránkovém režimu po 256 B v 256 stránkách pouze přes instrukci MOVX. Vzhledem k tomu, že tato část paměti není v mikropočítači fyzicky přítomna, budu se jí zabývat později.

Registry R0 až R7 umístěné v prvních 8 bajtech vnitřní datové paměti, z čehož vyplývá, že jde o osm osmibitových registrů, které tvoří Banku 0. Banky jsou celkem 4, tedy Banka 0 až Banka 3, a zabírají prvních 32B vnitřní datové paměti, dále jen RAM. Výběr aktuální funkční banky se provádí bity RS0 a RS1 v registru PSW (stavové slovo programu), registry vybrané banky jsou pak dostupné pod označením R0 až R7. Po restartu mikropočítače se RS0 a RS1 rovnají 0 a voláním například R0 volám registr R0 z Banky 0.

Od 32. bajtu, tedy od adresy 20H v hexadecimálním tvaru, následuje 16 B bitově adresovatelného prostoru, tedy 128 bitů. Pro přístup k těmto buňkám lze použít běžnou adresu (20H až 2FH tedy 32 až 47) pro práci s celým bajtem, nebo pokud potřebuji pracovat s konkrétním bitem, mohu použit bitovou adresu (00H až 7FH tedy 0 až 128).

Od 48. bajtu do 127. bajtu se nachází volně využitelný datový prostor, ke kterému ale lze již přistupovat pouze po celých bajtech.

K této doposud popsaná část paměti lze přistupovat přímo i nepřímo. Při přímém přístupu k volání daného paměťového místa mohu použit přímou adresu daného místa, kdežto u nepřímého přístupu, musím dané místo volat přes registry R0 nebo R1, v kterém je pak uložena adresa daného paměťového místa. S adresováním souvisí RAR (registr adresy RAM).

Od bajtu 128 je situace o něco složitější. U všech mikropočítačů z rodiny 8051 od tohoto bajtu začínají registry speciálních funkcí dále jen SFR. Tato oblast paměti je adresovatelná pouze přímo. Rozmístění SFR je znázorněno na obr. 5.

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000								0AFH
0A0H	P2 11111111		AUXR1 XXXXXXX0				WDTRST XXXXXXXX		0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	AUXR XXX00XX0		8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000		PCON 0XXX0000	87H

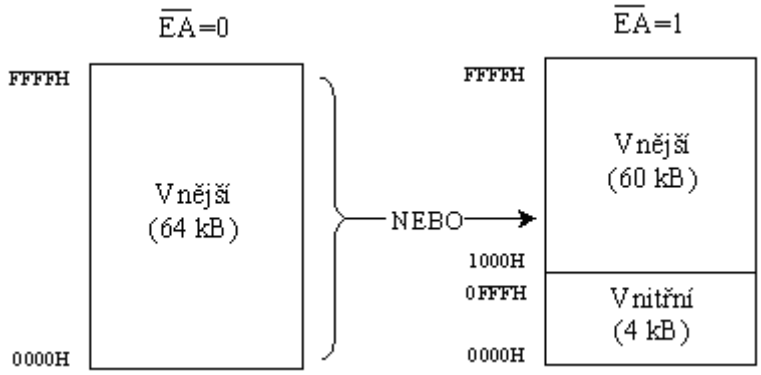
Obrázek 5 Přehled a rozmístění SFR zdroj [7]

Jak jsem se již zmínil, s adresami od 128. bajtu výše to není tak jednoduché, jak je patrno z obr. Uživatelsky volně použitelná paměť má stejný rozsah jako paměť SFR. To je dáno tím, že tyto dvě oblasti se překrývají. Z tohoto důvodu je tato paměťová oblast adresovatelná pouze nepřímo přes registry R0 a R1. Podle způsobu adresování pak mikropočítač pozná, jestli se jedná o uživatelsky volně použitelnou část pamětí anebo o oblast SFR. Jak označení tohoto adresového prostoru naznačuje, je využití tohoto prostoru plně na vůli uživatele. Tímto paměťovým prostorem nejsou vybaveny všechny typy mikropočítačů. O funkci jednotlivých SFR registrů se zmíním v poslední kapitole v závislosti na tom, jak je budu používat.

1.4.2.2. Paměť programu ROM

Paměť programu slouží k uložení programu a konstant. Mikropočítače se vyrábí v provedení s vnitřní pamětí programu ale i bez ní, v tom případě musí být mikropočítač vybaven vnější pamětí programu. Mikropočítač s vnitřní pamětí programu může využívat jak vnitřní, tak vnější paměť, nebo obě paměti současně. Jak ukazuje níže uvedený obr. 6, volba paměti je závislá na hodnotě signálu EA. Pokud je tento signál roven logické 0, čte se program jen z vnější pamětí, ale když je signál EA = 1, čte se program mikropočítače z vnitřní paměti a teprve až když hodnota čtené adresy překročí rozsah vnitřní paměti, čte mikropočítač instrukce z vnější paměti. Vnitřní paměť programu má kapacitu 4 kB a u vylepšené verze z řady 8052 pak obvykle 8 kB.

Mezi výrobci existuje nepsané pravidlo ve značení mikropočítačů podle typu vnitřní paměti programu. Verze bez jakékoliv vnitřní paměti programu začíná číslem 80, verze s pamětí v provedení OTP číslem 83, verze s pamětí EPROM číslem 87 a verze s pamětí FLASH/EEPROM začíná číslem 89. Jak jsem ale napsal, jde o nepsané pravidlo a tak ho ne všichni výrobci používají.



Obrázek 6 Rozložení vnitřní a vnější paměti programu zdroj [17]

Pro adresování paměti programu slouží šestnáctibitový čítač programu PC. Z toho vyplývá, že maximální délka programu je 64 kB.

Oblast paměti programu z rozsahu 00H až 23H, po případě až 2BH pro řadu 8052, má speciální význam a v tomto rozsahu adres musí být umístěny začátky programů pro obsluhu přerušení podle tabulky 1. Na paměťové místo 00H skočí mikropočítač vždy po restartu systému.

Zdroj přerušení		**Adresa**
Příznak	**Význam**	
IE0	Vnější přerušení 0	0003H
TF0	Čítač / časovač 0	000BH
IE1	Vnější přerušení 1	0013H
TF1	Čítač / časovač 1	001BH
RI + TI	Sériový kanál	0023H

Tabulka 1 Paměťová místa obsluhy přerušení zdroj [23]

Pokud by se někomu zdálo, že pro program přerušení je vyhrazeno málo místa, tak v této oblasti se nemusí nacházet celý program, ale jak již jsem napsal pouze jeho začátek. Tedy pokud mi toto místo pro program nestačí, na příslušnou adresu umístím instrukci skoku k místu s programem přerušení.

1.4.2.3. Vstupy a výstupy

1.4.2.3.1. Piny XTAL1 a XTAL2

Jde o vstup pro invertující oscilátor XTAL1 a výstup XTAL2. na tyto piny se připojí krystal oscilátoru, který udává frekvenci oscilátoru mikropočítače, nebo k připojení externího oscilátoru, který připojíme přes hradlo k vývodu XTAL2 a pin XTAL1 uzemníme.

1.4.2.3.2. Pin RST

Slouží pro restart mikropočítače. Restart mikropočítače zajistí nastavení čítače instrukci na nulu a nastavení registrů do výchozího stavu, tedy program se po restartu rozeběhne od adresy 0000H. Bez restartu po spuštění by mikropočítač pokračoval v práci od náhodné instrukce a jeho chování by bylo chaotické. Pro zajištění restartu mikropočítače je potřeba na tento pin po ustálení napájecího napětí přivést signál o délce nejméně dvou strojových taktů mikropočítače. Strojový takt mikropočítače se rovná jedné dvanáctině frekvence oscilátoru ať již vnitřního anebo vnějšího. Řada mikropočítačů 8051 pro restart (nulováni) vyžaduje signál o hodnotě log. 1 (HI).

1.4.2.3.3. Pin ALE/PROG

Tento signál má dvě funkce. Signál ALE (Adress Latch Enable) má konstantní kmitočet, který se rovná jedné šestině kmitočtu oscilátoru. Lze jej použít pro časování anebo jako pomocný zdroj hodin. Hlavní využití signálu ALE souvisí s využitím vnější paměti, tímto signálem se určuje, je-li na portu P0 přítomna spodní část adresy nebo data. Signál ALE se ovládá bitem DIALE v SFR registru AUX., je-li tento bit nastaven, ALE se generuje pouze v průběhu instrukce MOVX.

Při paralelním programování se tento signál značí PROG a slouží k potvrzení adresy a dat pro programování jedné buňky.

1.4.2.3.4. Pin PSEN

Z anglického Program Strobe ENable se používá jako strobovací signál při čtení z vnější pamětí programu.

1.4.2.3.5. Pin EA

Rozhoduje, jestli se program bude vykonávat z vnitřní a nebo vnější paměti programu. Blíže jsem tento signál popsal již v kapitole rozdělení paměti ROM.

1.4.2.3.6. Porty

Mikropočítače řady 8051 jsou vybaveny 4 porty (P0 až P3). Všechny porty jsou vybaveny vyrovnávacím registrem, jde o klopny obvod typu latch, který reaguje na změnu na svých vstupech po celou dobu trvání hodinového pulzu. Po skončení hodinového pulzu je na

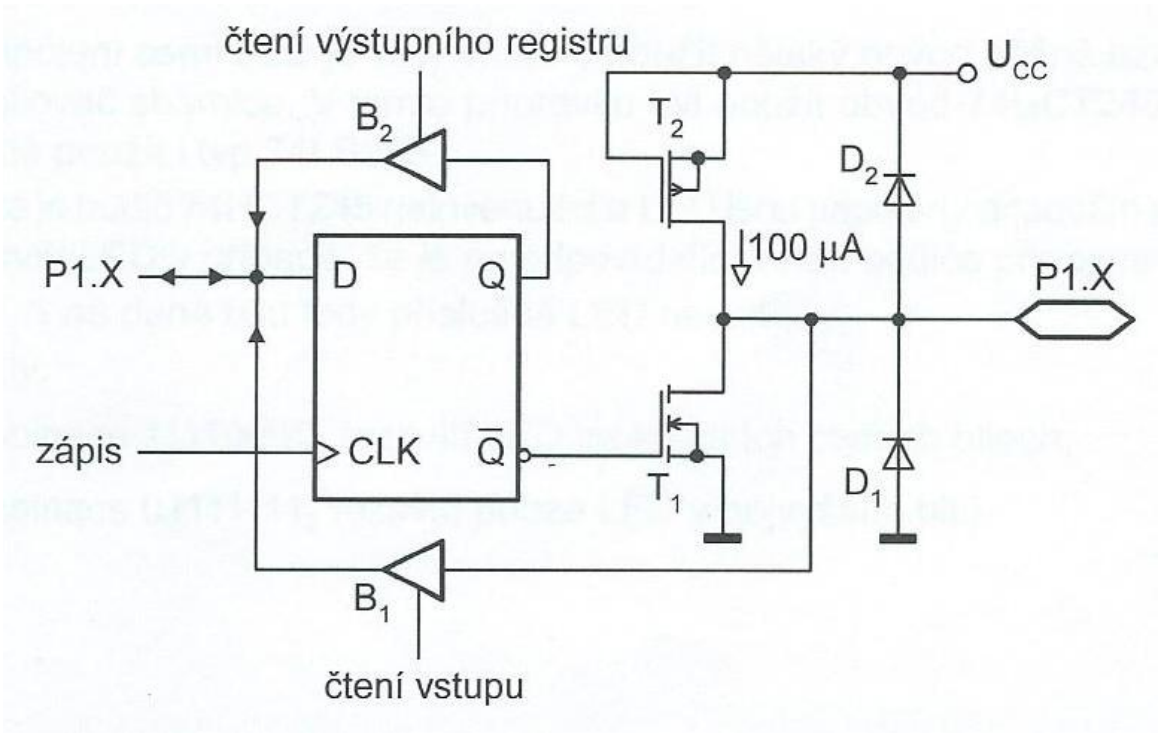
výstupu vyrovnávacího registru poslední platný stav. Úmyslně píši port a ne brána, což je starší a stale používaný termín a rozdíl mezi nimi spočívá v tom, že u brány byly všechny její linky vždy nastaveny jako vstup, nebo výstup u portu může být každá jeho linka nastavena nezávisle na ostatních. Na obr. 7 je znázorněno schéma zapojení jedné linky, zde konkrétně pro port P1, ale toto zapojeni je shodné pro porty P1 až P3. Port P0 se odlišuje tím, že nemá zabudovaný tranzistor T2, který u ostatních portů realizuje funkci proudového zdroje 100 μA, tím, že má zabudovaný vodivý kanál a vede již při nulovém napětí. Tím realizuje na nezatíženém vývodu portu úroveň logické jedničky, dále jen log. 1 a realizuje takto funkci pul-up (zvyšovacího) rezistoru. Z toho vyplývá, že port P0 je realizován s otevřeným kolektorem a díky tomu je třístavový a lze jej použít jako systémovou sběrnici, na které je multiplexován dolní bajt adresy a data při připojené vnější paměti dat, nebo se po něm přenáší kód instrukce z vnější paměti programu.

Na portu P2 je pak možno vysílat horní bajt adresy

Po resetu jsou všechny porty nastaveny do log. 1, tedy na 11111111 pro jeden port. Takto slabou log. 1 mohou vnější obvody snadno přetáhnout, což umožní čtení stavu daného vývodu. Vývod se tedy chová jako vstup.

Po zápisu log. 0 na daný bit, je na hradlo T_1 přivedená log. 1, vzhledem k tomu, že hradlo je připojeno na invertujicí výstup vyrovnávajícího registru a tranzistor T_1 je sepnut, tím je stažen výstupní proud tranzistoru T_2 vůči zemi a vývod se chová jako výstup s log. 0.

Ochranu vývodu před napětím vyšším než U_{cc} + 0,7 V nebo nižším než GND – 0,7 V zajišťují diody D_1 a D_2.



Obrázek 7 Vnitřní zapojení jednoho bitu portu zdroj [14]

Některé vývody portů alternativní funkce. O možnostech připojení vnější paměti programu a dat na porty P0 a P3 jsem se zmínil již výše v této kapitole.

Alternativní funkce linek portu P1 a P3 vis tabulka 2.

Vývod	Alternativní funkce	Popis
P1_0	T2	Externí vstup čítače/časovače 2 (jen u řad 8032 a 8052)
P1_1	T2EX	Signál pro zachycení hodnoty nebo nového naplnění čítače/časovače2 (jen u řad 8032 a 8052)
P3_0	RxD	Sériový vstupní port
P3_1	TxD	Sériový výstupní port
P3_2	INT0	Negovaný vstup vnějšího přerušení
P3_3	INT1	Negovaný vstup vnějšího přerušení
P3_4	T0	Vnější vstup čítače/časovače0
P3_5	T1	Vnější vstup čítače/časovače1
P3_6	WR	Negovaný zápis do vnější pamětí dat
P3_7	RD	Negované čtení z vnější pamětí dat

Tabulka 2 Alternativní funkce linek portů zdroj [14]

1.4.2.4. Aritmetickologická jednotka ALU

ALU je tvořena kombinačními logickými obvody a zpracovává osmibitové operace s jedním nebo se dvěma operandy. ALU může vykonávat tyto operace:

- čtyři aritmetické operace (sčítání, odčítání, násobení a dělení) s přenosem i bez přenosu. Základem všech aritmetických operací je logická operace EXCLUSIVE OR.
- logické operace AND, OR a EXCLUSIVE OR
- přičtení a odečtení jedničky (inkrement a dekrement)
- negaci bitu (komplement)
- rotace vpravo a vlevo s přenosem i bez přenosu
- výměnu vyšší a nižší poloviny bytu
- nastavení bajtu do BCD tvaru

Po dobu zpracovávání instrukce si ALU uchovává operandy v pomocných registrech THP1 a THP2, proto jsou někdy tyto registry nazývány dočasnými registry. Na tyto registry se uživatel nemůže obracet, jsou zcela pod správou ALU.

1.4.2.5. Řadič přerušení

Přerušení programu je reakce mikropočítače na vnější událost. Jak je vidět z blokového schématu na obr. 3, s řadiči přerušení souvisí vnější vstupy a z vnitřních přerušení to jsou přerušení od čítačů/časovačů a sériového kanálu.

Pokud řadič přerušení obdrží žádost o přerušení programu v podobě některého z výše popsaných signálu, dojde k zastavení probíhajícího programu a ke skoku na obsluhu přerušení. Paměťová místa obsluhy přerušení jsou v tabulce 1. Po vykonání obsluhy přerušení se mikropočítač vrátí zpět k původnímu programu od místa, ve kterém byl probíhající program přerušen. Adresu poslední návratové hodnoty si mikropočítač ukládá do ukazatele zásobníku SP, po odebrání poslední návratové hodnoty se ukazatel zásobníku dekrementuje, čímž opět ukazuje na poslední návratovou hodnotu. Zásobník může být uložen kdekoliv v RAM s výjimkou SFR.

1.4.2.6. Střadač A

Jindy nazývaný akumulátor je základním registrem procesoru. Jde o zdrojový a cílový registr prakticky aritmetických operací, většiny logických operací a operací s přesunem dat. Tento registr se v instrukcích většinou značí jako „A“ nebo „ACC“.

1.4.2.7. Registr B

Jde o pomocný registr, do kterého se ukládá druhý operand při operacích násobení a dělení. Pokud se neprovádí některá z těchto operací, je registr volně k dispozici.

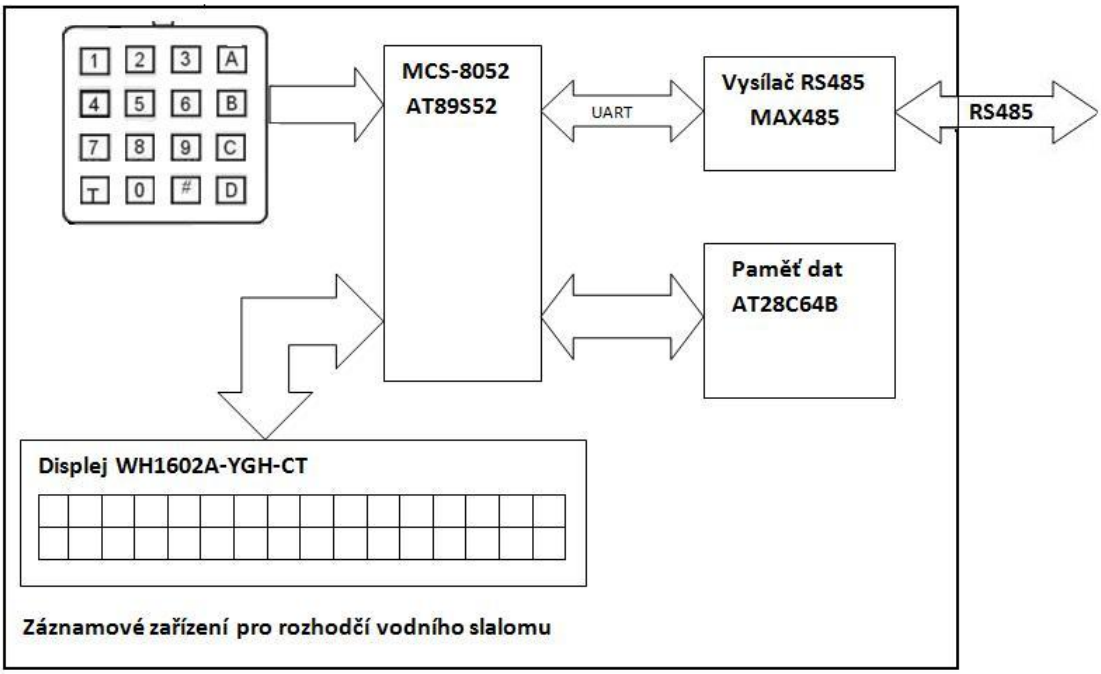
2. Volba mikropočítače a periferii

2.1. Základní požadavky na záznamové zařízení pro rozhodčí vodního slalomu

Jak jsem popsal již v úvodu, základní požadavky na záznamové zařízení jsou:

- Odolnost proti povětrnostním vlivům. Zařízení bude provozováno pod širým nebem.
- Možnost vkládání záznamů obsluhou, tedy brankovým rozhodčím o průjezdu závodníka. Pro zadávání klávesnice a pro kontrolu zadaných dat displej. Displej musí být schopen přehledně zobrazit označení závodní kategorie, sledované startovní číslo a trestné body k jednotlivým závodním branám, kterých by mělo být 4 až 5 na stanoviště. Klávesnice pak zákonitě musí umožnit zadávání všech těchto údajů + ovládaní zařízení.
- Online přenos záznamů do PC ke zpracování výsledků. Výsledky se zpracovávají v programu Eskymo od firmy Results.cz s.r.o. a jde o balík rozšiřujících funkcí pro program Calc z kancelářského balíku programů OpenOffice.
- Zálohování dat z důvodu, že pravidla kanoistiky na divokých vodách [22], ukládají povinnost pořadateli závodu uchovávat důležité písemnosti závodu nejméně po dobu 4 měsíců po závodě z důvodu případné reklamace výsledků, nebo kontroly výkonného výboru ČSKDV. Mezi tyto dokumenty patří i záznamy od brankových rozhodčích.
- Pro zavedení zařízení do provozu je důležitá i jeho finanční náročnost. Tedy, aby měl projekt šanci na realizaci, mělo by zařízení být co nejlevnější.

Na základě výše popsaných požadavků na záznamové zařízení si mohu udělat představu o záznamovém zařízení a vytvořit blokové schéma tohoto zařízení, vis blokové schéma na obr. 8.



Obrázek 8 Blokové schéma záznamového zařízení

2.2. Mikropočítač

Nad volbou mikropočítače pro záznamové zařízení jsem dlouho neváhal a zvolil jsem mikropočítač AT89S52 v pouzdře DIL40, vis obr. 9. Jde o mikropočítač od společnosti Atmel z řady 8052, což je vylepšená varianta řady 8051 a se svým předchůdcem je plně kompatibilní. Pro tento mikropočítač jsem se rozhodl z důvodu, že s touto řadou jsme pracovali ve škole a je to jediná řada, se kterou mám nějakou zkušenost. Pro AT89S52 jsem se pak rozhodl z důvodu:

- Existuje pro něj spousta dostupné kvalitní literatury se spoustou praktických příkladů a je tím vhodný i pro začátečníky [14][15]
- Pro tento mikropočítač se prodává cenově dostupný vývojový kit USB51KIT.[18] Základní vlastnosti pro daný vývojový kit:
    a. Je určen pro mikrokontroleryAT89S51 a AT89S52 v provedení DIP40, výhodou těchto mikropočítačů je, že podporují rychlé programování v dávkovém režimu.
    b. Rychlé programování zhruba 100 B/s, přes USB rozhraní, které je snadno dostupným počítačovým rozhraním.
    c. Poskytuje možnost napájet připojené přípravky přímo z počítače až do odběru 500 mA. Pro vyšší odběry je možnost připojení vnějšího zdroje.
    d. Programy lze do mikropočítače zavádět ve formátu BIN, nebo HEX.
- V neposlední řadě jde o mikropočítač s příznivou pořizovací cenou okolo 45,- Kč a pro danou aplikaci s dostatečným výkonem, vis vlastnosti AT89S52 níže.
- Pro variantu v pouzdře DIP40 jsem se rozhodl proto, že právě toto pouzdro je podporované vývojovým kitem USB51KIT.

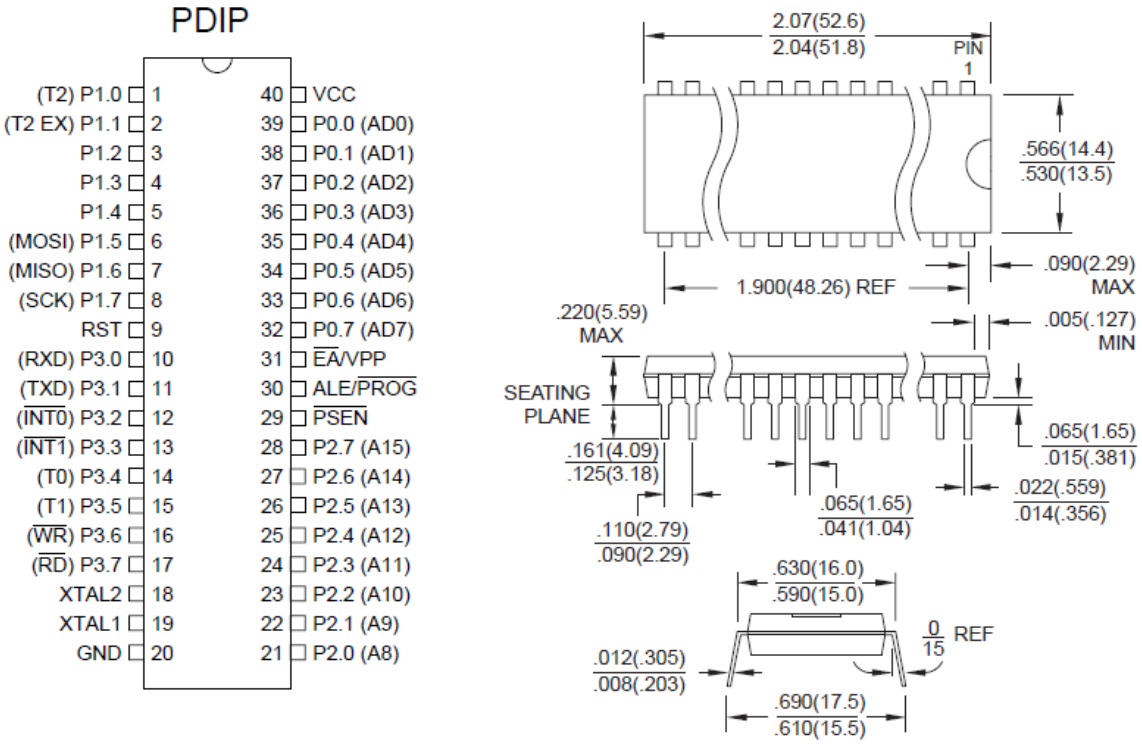
Možná by bylo efektivnější použít některý z mikropočítačů z řady AVR, ale tuto variantu jsem zamítl z prostého důvodu, bál jsem se vykročit úplně neznámým směrem.

2.2.1. Vlastnosti AT89S52

Tento mikropočítač patří do řady 8052, která je plně kompatibilní s řadou 8051, obě jsou popsány v kapitole obecný popis mikropočítače i s přihlédnutím k některým drobným odlišnostem.

Základní parametry AT89S52:

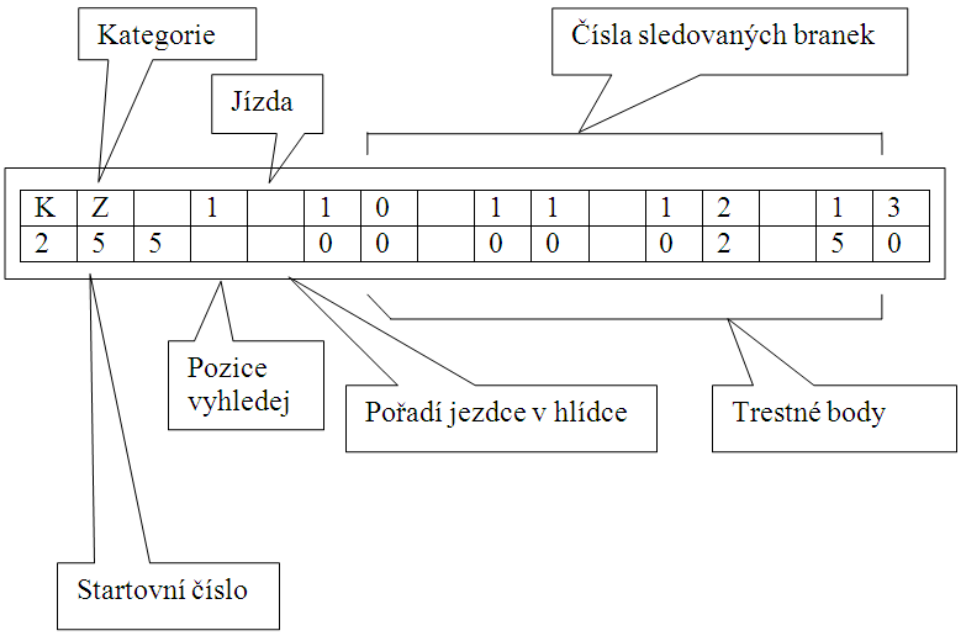
- mikropočítač CMOS
- absolutní rozsahy:
  - maximální provozní teploty -55 až +125° C
  - maximální skladovací teplota -65 až +150° C
  - maximální napájecí napětí 6,6 V
  - maximální napájecí proud při 12 MHz je 25 mA
  - maximálně odebíraný proud ze všech výstupů 15 mA pro P1, P2 a P3. Pro P0 je to 26 mA.
  - maximální odebíraný proud z pinu je 10 mA proti Ucc a proti a GND 100 µA.
- vnitřní paměť programu Flash 8 kB
- hodinový kmitočet 0 až 24 MHz
- vnitřní paměť dat RAM 256 B
- 32 vstupně/vstupních linek
- 3 šestnáctibitové čítače/časovače
- 9 zdrojů přerušení
- zabudovaný sériový kanál UART
- 2 režimy sníženého odběru



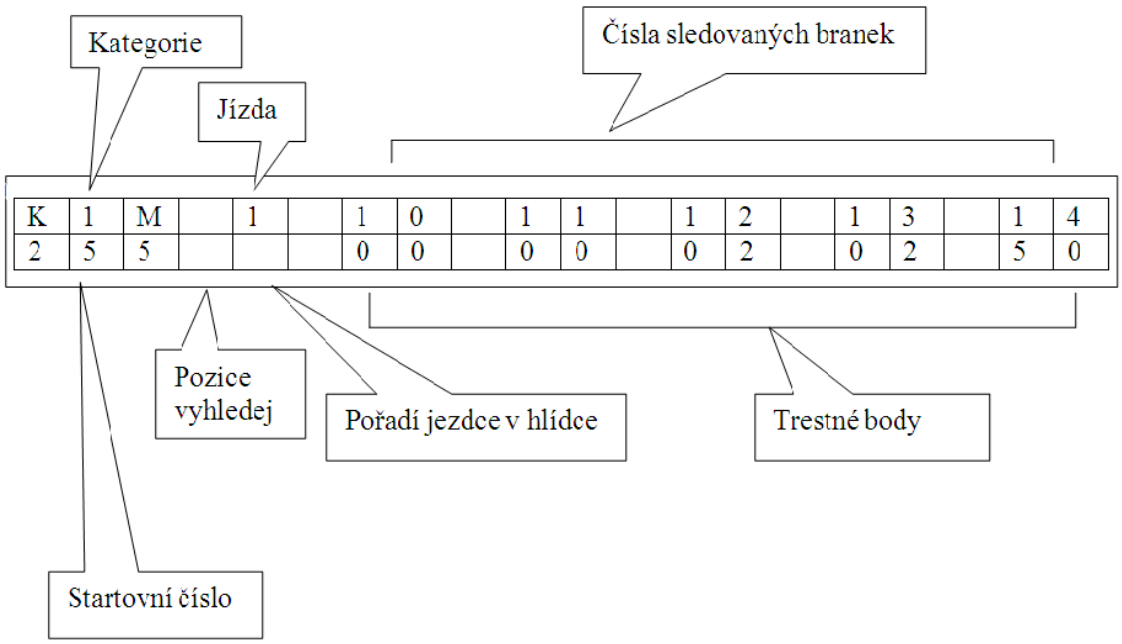
Obrázek 9 Pouzdro DIP40 mikropočítače AT89S52 zdroj [7]

2.3. Displej

K výběru displeje jsem si připravil dva návrhy pro displej 2 řádky po 16 znacích, pro který je uvedena řada příkladů v literatuře [16] a pro displej 2 krát 20 znaků, vis obr. 10 a obr. 11.



Obrázek 10 Návrh zobrazení pro displej 2 řádky po 16 znacích

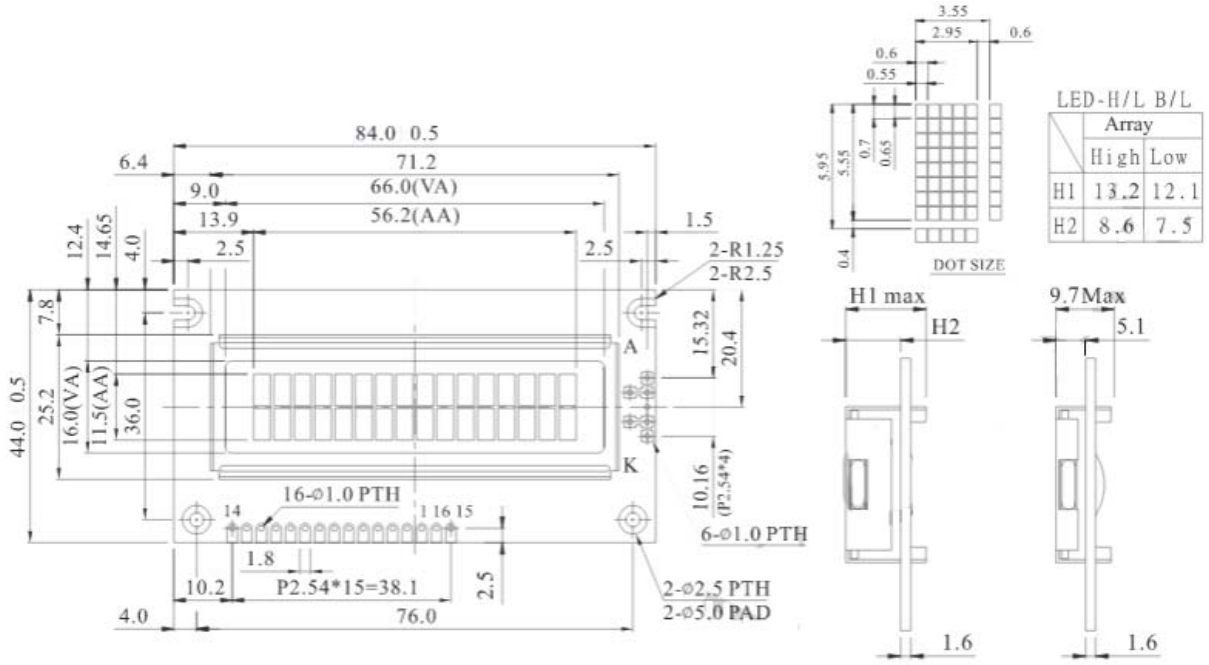


Obrázek 11 Návrh zobrazení pro displej 2 řádky po 20 znacích

Jak je z obr. 10 a z obr. 11 patrné displej 2x20 znaků by plně odpovídal požadavkům na záznamové zařízení, ale právě tuto řadu displejů výrobce vyrábí, z mého pohledu, s méně praktickým připojením než řadu 2x16, proto jsem se vrátil k variantě displeje 2x16. Z nabídky displejů 2x16 jsem si vybral displej WH1602A-YGH-CT od společnosti Winstar, zobrazení displeje, vis obr. 12. Ponechávám zde ale i návrh pro displej s 20 znaky v řádku, neboť bych se k němu v budoucnu někdy rád vrátil. Dle mého by v sadě asi 8 záznamových zařízení bylo dobré mít pro závod i jednu verzi s 20 znaky ve dvou řádcích.

2.3.1. Charakteristické vlastnosti displeje WH1602A-YGH-CT:

- napájecí napětí 2,7 až 5,5 V, typicky 5 V
- proudový odběr maximálně 1 mA
- provozní teplota -20 až 70° C
- displej podporuje jak 4 bitovou tak 8 bitovou komunikaci
- jde o transflektivní provedení displeje, tedy může pracovat s podsvětlením i bez podsvětlení displeje

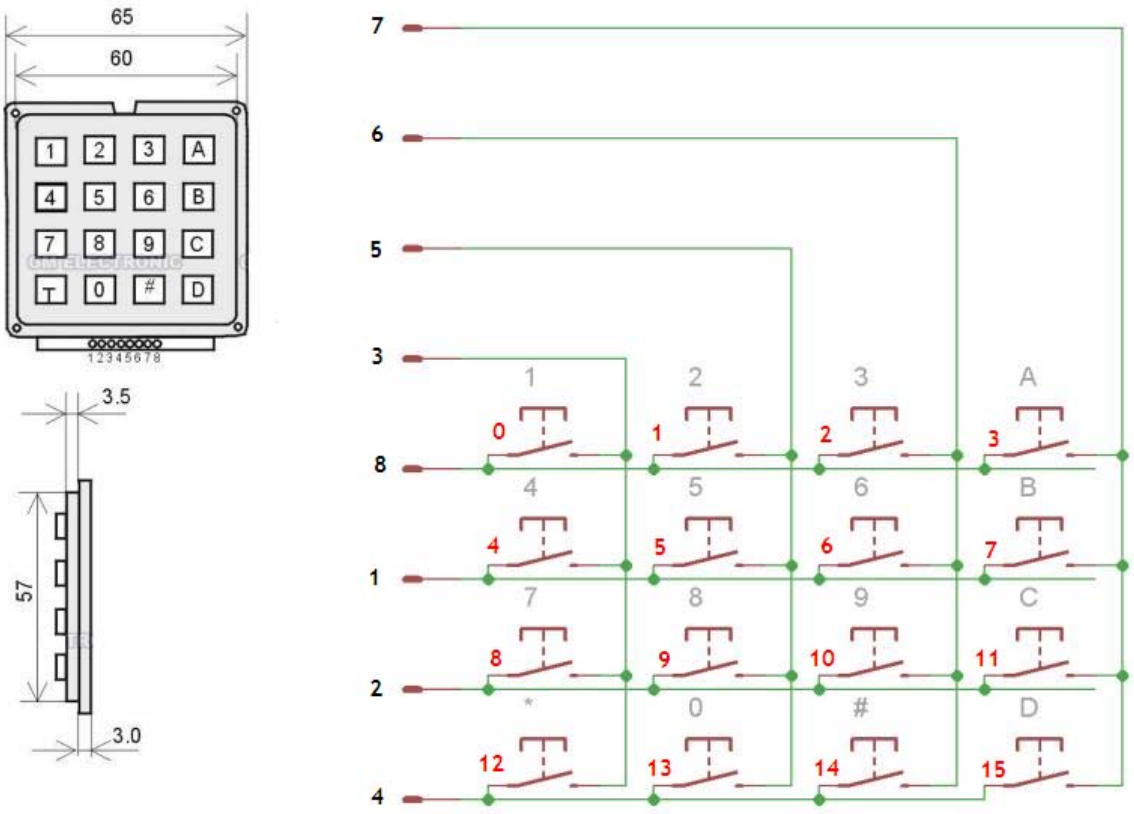


Obrázek 12 Zobrazení a rozměrový výkres displeje WH1602 zdroj [33]

2.4. Klávesnice

Klávesnicí se celé záznamové zařízení ovládá, proto nemůže být schovaná uvnitř zařízení jako většina ostatních komponentů záznamového zařízení, ale musí být snadno přístupná, tedy bude vystavená vlivům okolí. Z tohoto důvodu je zapotřebí, aby klávesnice byla voděodolná pro případ deště.

Obsluha záznamového zařízení bude potřebovat do přístroje zadávat číselné údaje o startovním čísle závodníka, trestné body pro průjezd závodní branou a pořadí jízdy, tedy znaky 0 až 9. Obsluha bude dále potřebovat ovládací klávesy pro uložení záznamu, pohyb po displeji (doleva, doprava, nahoru a dolu) a pro opravu případných chybných zápisů klávesu smaž. Tedy pro ovládání zařízení bude potřeba 16 kláves. Z tohoto důvodu jsem pro ovládání záznamového zařízení zvolil maticovou klávesnici 4x4 F-KV16KEY, která zapojením tlačítek do řádků a sloupců účinně snižuje počet potřebných vstupů pro připojení 16 kláves. Tato klávesnice má navíc svou mechanickou a elektrickou část oddělenou gumovým těsněním a tak je i ze strany kláves vodotěsná. Maximální elektrické parametry této klávesnice jsou 24 V/20 mA a maximální odpor je 200 Ω, což jsou parametry pro záznamové zařízení zcela dostačující. Tato klávesnice se svou cenou okolo 160,- Kč je pro zařízení vhodným řešením i z ekonomického hlediska. Vzhled, rozměry a zapojení jednotlivých tlačítek této klávesnice vis obr. 13. Ze schématu zapojení klávesnice je patrný i princip funkce maticové klávesnice. Na klávesnici je vysílán definovaný signál postupně na všechny řádky, když je signál na daném řádku přítomen, postupně otestuji všechny sloupce a pokud je některá klávesa v daném řádku stisknuta, objeví se daný signál i na daném sloupci. Z průsečíku sloupce a řádku získám pozici stisknuté klávesy.



Obrázek 13 Vzhled a zapojení klávesnice F-KV16KEY zdroj [13][14]

2.5. Komunikace s osobním počítačem

Pro výběr způsobu přenosu dat do osobního počítače, jsem se nechal inspirovat již používanou časomírou REI2, která pro komunikaci s ostatními rozšiřujícími moduly, jako například výsledková tabule, používá sběrnici RS485 a pro komunikaci s osobním počítačem má i sériové rozhraní RS232. O bezdrátový přenos se daná časomíra dá rozšířit pouze na vstupu startovní fotobuňky, nebo se dá použít spojení metalickou dvojlinkou. Vzhledem k tomu, že náš předseda a vrchní časoměřič v jedné osobě se používání bezdrátového přenosu brání se slovy, že nevěří ničemu, na co si nemůže sáhnout a elektřině obzvlášť. Nevím sice, co ho k tomuto rozhodnutí vede, neb sáhnout si na elektřinu jsem ho již několikrát viděl, ale vzhledem k tomu, že jde o hlavního uživatele a vzhledem k rozloze sportovního areálu umělé slalomové dráhy ve Veltrusech, vis obr. 14, kde je červeně mezi praporky vyznačena délka závodní tratě, podél které se rozmisťují posty brankových rozhodčích, rozhodl jsem se pro metalický přenos dat po sběrnici RS485. Tato sběrnice má vyhovující parametry pro přenos dat v dané aplikaci.

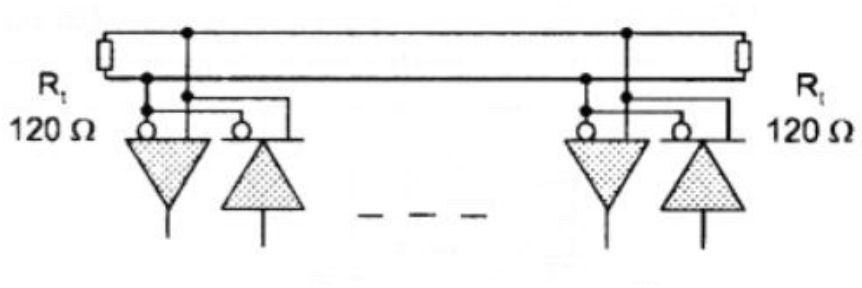
Obrázek 14 Areál umělé slalomové dráhy ve Veltrusích zdroj [31]

2.5.1. RS-485

Standard RS-485 je navržen tak, aby umožňoval vytvoření dvouvodičového poloduplexního vícebodového sériového spoje, který umožňuje propojení až 32 zařízení až na vzdálenost 1200 m bez použití opakovače. RS485 má stejný základ jako RS232, od kterého se odlišuje definicí napěťových úrovní tak, že detekce logického stavu je založená na rozdílovém napětí mezi oběma vodiči. Toto je výhodné zejména kvůli snížení vlivu indukovaného rušivého signálu, který se obvykle přičítá k oběma vodičům stejně. Standard RS485 byl původně navržen pro průmyslové využití, z toho vyplývá, že nejde o datové přenosy zaměřené na velké objemy dat. Přenosová rychlost může být u krátkých vedení až 10 Mb/s, ale s rostoucí vzdáleností spojení klesá maximální přenosová rychlost v závislosti na kapacitě vedení, vis tab. 3. Vedení musí být na obou stranách zakončeno zakončovacími odpory, takzvanými terminátory. Úkolem "terminátorů" je zabránit odrazům signálu od konců vedení, také pomáhají zvýšit odolnost linky proti rušivým signálům. Terminátor by měl mít hodnotu 120 Ω. Zapojení sběrnice je na obr. 15. Jednotlivé vodiče se u RS485 značí „A“ a „B“ nebo také „+“ a „-“, což je odvozeno od klidového stavu na lince, kdy A je kladnější než B. Správně musí být tento rozdíl minimálně 0.2 V, jinak jde o nedefinovaný, tedy zakázaný stav na lince. Log. 1 je na lince pokud A-B>= +0.2 V a obráceně log. 0 je na lince, když A-B<= -0.2 V. Pro zajištění bezchybného přenosu dat je zapotřebí zajistit, aby v daný okamžik vysílalo na sběrnici pouze jedno zařízení, nejčastěji se používá řešení s jedním řídícím zařízením (Mastr), které ostatním zařízením (Slave) povoluje přístup na sběrnici. RS485 je čistě sběrnicové řešení, a proto by odbočky z takzvaných „T“ uzlů neměly být delší než 5 m.

Přenosová rychlost	1200bd	2400bd	4800bd	9600bd	19200bd	38400bd	57600bd	115200bd
Max. kapacita vedení	250nF	120nF	60nF	30nF	15nF	750pF	500pF	250pF

Tabulka 3 Vliv kapacity vedení na přenosovou rychlost zdroj [24]

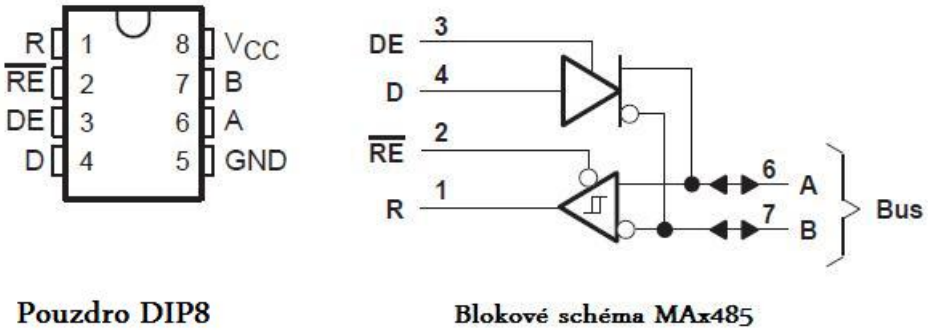


Obrázek 15 Princip zapojení sběrnice RS-485 zdroj [24]

2.5.2. Vysílač RS-485

Při výběru vysílače RS485 jsem se řídil recenzí na tyto integrované obvody [34] a vybral jsem si základní obvod pro tuto sběrnici od firmy Maxim MAX485. Pouzdro a blokové schéma MAX485 vis obr. 16. Charakteristické vlastnosti integrovaného obvodu MAX485:

- napájecí napětí 5 V±5 %, maximální napětí 12 V
- maximální odebíraný proud 0.9 mA
- provozní teplota 0° C až 70° C
- maximální přenosová rychlost 2.5 Mbps
- half duplex přenos pouzdro plastové 8DIP
- přijatelná cena okolo 55,- Kč.



Obrázek 16 Vzhled a blokové schéma vysílače MAX485 zdroj [19]

DE – povolení vysílání
D – Data pro vysílání
/RE– negovaný vstup povolení příjmu (příjem povolen když /RE=log.0)
R – Přijatá data
A a B – Vodiče sběrnice RS485 (význam popsán již výše)
Vcc – Napájení
GND – Zem ke které je vztaženo Vcc

2.5.3. Přenosové medium

Jak jsem již uvedl výše, sběrnice RS485 pro přenos dat využívá kroucený pár vodičů. Maximální přenosová rychlost je pak závislá na kapacitě použitého kabelu, vis tab. 3.

Vzhledem k plánovanému nízkému objemu přenášených dat a s přihlédnutím k cenám dostupných kabelů jsem se rozhodl nepřekračovat přenosovou rychlost 2400 b/s, tedy použití kabelu, který bude se svou kapacitou na 1000 m mezi 120 nF až 250 nF. Vybral jsem kabel J-Y(ST)Y 1x2x0.8 ROT od firmy Prakab s.r.o. Tento kabel má výborné vlastnosti v porovnání ke své ceně, která je okolo 4,- Kč/m. [12]

2.5.3.1. Vlastnosti kabelu J-Y(ST)Y 1x2x0.8 ROT:

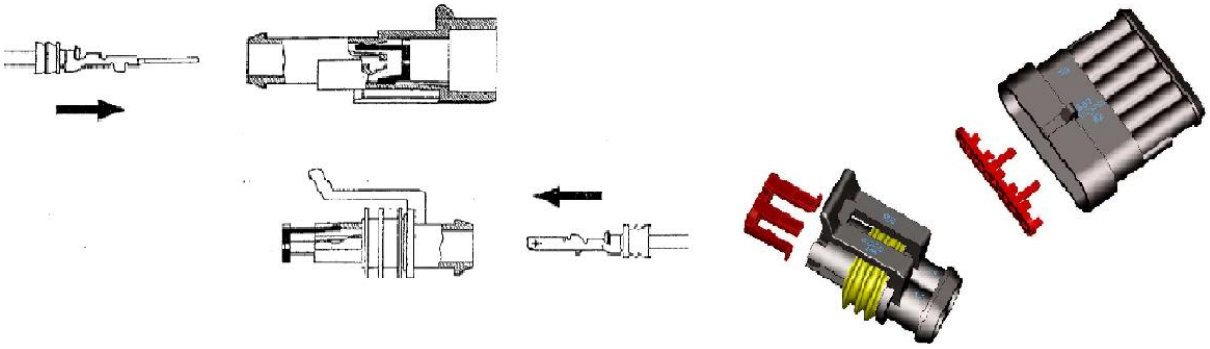
- Průřez jednoho vodiče 0.8 mm
- Stínění Al folie a přiložený Cu drátem
- Maximální odpor ve smyčce 73.2 Ω/km
- Maximální kapacita 120 nF/km
- Pracovní teplota -30 až 70° C
- Manipulační teplota -5 až 70° C

2.5.4. Konektory

Sběrnice RS485 nemá stanovený žádný standard, co se konektoru týká. Vzhledem k požadavku na provoz zařízení pod širým nebem jsem se rozhodl pro konektory s vyšším krytím aspoň IP61, tedy s krytím proti vnikání prachu a padajících kapalin s ekvivalentem deště 3 až 5 mm/min. Po průzkumu trhu s konektory odpovídajících vlastnosti jsem zvolil konektory Superseal serie 1.5 s krytím IP67. Tyto konektory jsem vybral hlavně pro jejich snadnou dostupnost na trhu a přijatelnou cenu. Nevýhodou těchto konektorů je pouze to, že se nevyrábí v provedení do panelu, ale jen v provedení kabel – kabel. Záznamové zařízení by pak bylo připojeno tímto způsobem:

- napájení – dvoupinový konektor vidlice TE CONNECTIVITY 282104-1
- vstup RS485 – třípinový konektor zásuvka TE CONNECTIVITY 202887-1
- výstup RS485 a zároveň připojení terminátoru na posledním záznamovém zařízení na sběrnici – pětipinový konektor zástrčka TE CONNECTIVITY 282889-1

Tento způsob různých konektorů zabrání i případnému nesprávnému připojení zařízení. Typ konektorů a způsob jejich zapojování je vyobrazen na obr. 17.



Obrázek 17 Konektor superseal serie 1.5 zdroj [4][5]

2.5.5. Rozšíření systému

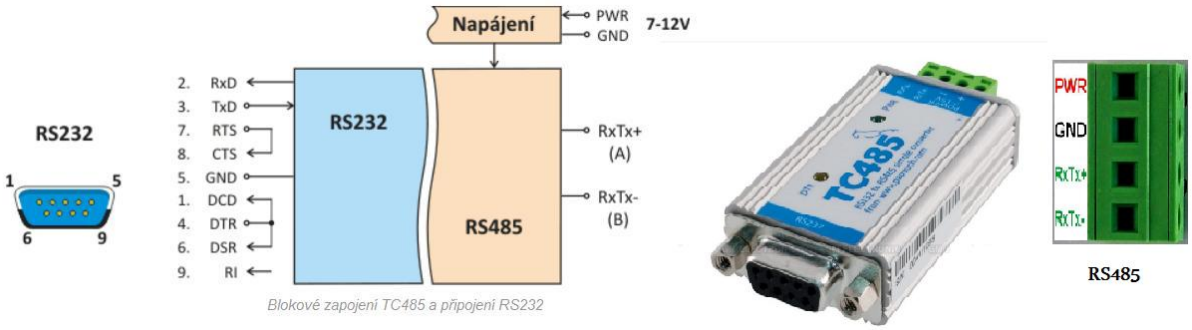
Vzhledem k tomu, že sběrnice RS485 podporuje až 32 zařízení na sběrnici a postů bývá do osmi plus osobní počítač, má tato sběrnice dostatečný potenciál k rozšíření celého systému. Bylo by tedy možné na tuto sběrnici připojit i informační tabule, které by sledovaly komunikaci na sběrnici a odchytávaly si startovní čísla a k nim hodnocení průjezdu daného závodníka, tato informace by pak mohla být zobrazena v podstatě hned poté, co jí brankový rozhodčí potvrdí. Takto rychlé zobrazení trestných bodů divákům by jistě vodnímu slalomu ještě přidalo na jeho atraktivitě.

2.5.6. Převodník RS232 na RS485

Zaznamenaná data potřebuji dostat do osobního počítače. Osobní počítače, ale nejsou standardně vybaveny rozhraním RS485, proto jsem se rozhodl pro převodník z RS232 na RS485. S komunikací přes RS232 již nějaké zkušenosti mam a doplnění PC o toto rozhraní není žádným problémem, případný přechod na rozhraní USB, by představoval jen výměnu převodníku. Konkrétně jsem zvolil převodník F-TC485. Jde o jednoduchý obousměrný a cenově dostupný převodník s cenou okolo 1.100,- Kč. [13] Převodník má automatické přepínání směru komunikace, to je výhodná vlastnost z hlediska, že obsluha nemusí nic nastavovat, ale za cenu, že převodník pro přepnutí z vysílání na příjem potřebuje časovou prodlevu 0.9 ms. Blokové schéma a celkový vzhled převodníku je zobrazeno na obr. 18.

2.5.6.1. Vlastnosti převodníku F-TC485:

- Galvanicky oddělená linka RS485
- Automatické přepínání směru komunikace
- Indikace zapnutí a komunikace kontrolkami.
- Napájení 7 až 12 V galvanicky spojeno s RS485.
- Část RS232 je napájena ze signálů DTR a RTS (postačuje jeden z nich v log. 1)
- Maximální rychlost komunikace až 115.2 kb/s
- Proudový odběr 5 mA.
- Odpory definující klidový stav 10 kΩ.
- Pracovní teplota 0 až 50 °C



Obrázek 18 blokové schéma a vyobrazení převodníku F-TC485 zdroj [26]

2.5.7. Zálohování záznamů

Dle pravidel kanoistiky [22], jak jsem uvedl již výše, musí brankový rozhodčí vést pečlivý záznam o probíhajícím závodu a tyto záznamy se mají uschovávat nejméně 4 měsíce po daném závodu, a to nejlépe v tištěné podobě. Z tohoto důvodu jsem plánoval vybavit záznamové zařízení nějakým typem tiskárny, jejímž prostřednictvím by daný rozhodčí měl přehled nad záznamy jednotlivých závodníků a také v pravidlech se počítá spíše s písemnou formou záznamů. Pro takovéto zařízení by byla vhodná malá termotiskárna do panelu, kterou je například termotiskárna GPT - 4454 od společnosti GeBE Elektronice [20], ale finanční náročnost tohoto typu tiskáren by pohřbila, dle mého, naděje na uvedení záznamového zařízení do provozu.

Z tohoto důvodu jsem usoudil, že bude lepší, když zařízení bude vybaveno paměťovým modulem a obsluha zařízení bude mít v této paměti možnost listovat. Mikropočítač řady 8052 ze své podstaty dovoluje obsluhu vnější pamětí do velikosti 64 kB. Z tohoto důvodu bude zapotřebí ukládat záznamy hospodárným způsobem. Rozhodl jsem se celý záznam pro jednoho závodníka uložit do 3 B, následujícím způsobem:

- 1. bajt

7. bit	**6. bit**	**5. bit**	**4. bit**	**3. bit**	**2. bit**	**1. bit**	**0. bit**
Jízda	1. bit Jezdec	0. bit Jezdec	1. bit TB5	0. bit TB5	2. bit Kategorie	1. bit Kategorie	0. bit Kategorie

Tabulka 4 Rozložení dat v 1. ukládaném bajtu

Jízda – možnost uložení dvou hodnot, reprezentuje 1. a 2. jízdu závodu. Log. 0 = 1. jízda, log. 1 = 2. jízda

Jezdec – zabírá v paměti dva bity a umožňuje uložení 4hodnot v rozsahu 0 až 3. Hodnoty 1 až 3 reprezentují pořadí jezdce v družstvu, takzvaný závod hlídek. Nula zůstává nastavena, pokud se nejedná o závod hlídek a s pořadím jezdce se nepracuje.

TB5 – zabírá v paměti rovněž dva bity a představuje získané trestné body závodníka na páté bráně daného postu s významem:

1. bit	**0. bit**	**Hodnota**	**Význam**
0	0	0	Čistý průjezd (bez trestných bodů)
0	1	1	Rezerva (pravděpodobné využití DNF)
1	0	2	S dotykem (reprezentuje 2 trestné sekundy)
1	1	3	Neprojeto (Reprezentuje 50 trestných sekund)

Tabulka 5 Kódování trestných bodů

DNF – Dit Not Finish (nedokončení závodu)

Kategorie – rozsah 3 bitů představující závodní kategorii ve vodním slalomu s významem:

2.bit	1.bit	0.bit	Hodnota	Kategorie	Zobrazení displej
0	0	0	0	Rezerva (nezařaditelné)	Os
0	0	1	1	K1M	KM
0	1	0	2	K1Ž	KZ
0	1	1	3	C1M	CM
1	0	0	4	C1Ž	CZ
1	0	1	5	C2M	C2
1	1	0	6	C2Ž	CW
1	1	1	7	Hlídky	Hl

Tabulka 6 Způsob kódování kategorie

- 2. bajt

Druhý ukládaný bajt obsahuje na všech svých 8 bitech startovní číslo závodníka. Tedy hodnotu v rozsahu 0 až 255.

- 3. bajt

7. bit	6. bit	5. bit	4. bit	3. bit	2. bit	1. bit	0. bit
1. bit TB4	0. bit TB4	1. bit TB3	0. bit TB3	1. bit TB2	0. bit TB2	1. bit TB1	0. bit TB1

Tabulka 7 Způsob uložení informací do 3. ukládaného bajtu

TB1 až TB4 – mají stejný význam, jako TB5 v ukládaném prvním bajtu, vis tab. 5, číslo 1 až 4 představuje pořadí závodní brány na daném postu.

Budu-li tedy počítat s plným obsazením kategorii v rozsahu, které záznamové zařízení dovoluje, bude velikost minimální potřebné paměti rovna:

sedmi kategoriím po 255 závodnících, tedy 1785 závodníků.

jedna kategorie po 3 x 255 = 765 závodnících (u každého startovního čísla v závodě hlídek jsou tři záznamy)

Tedy záznamové zařízení dovoluje uložit 2550 startujících a pro každého závodníka zařízení potřebuje 3 bajty, tedy celkem 7650B a každý závodník startuje ve dvou jízdách, tedy krát 2, což je 15300B paměti bez uvažování potřebné rezervy pro opravované záznamy. Jak již jsem ale napsal výše v úvodu, na většině závodů počet uskutečněných startů se pohybuje

okolo 250 a to se již jedná o dobře obsazený závod. Navíc výše vypočítaná teoretická hodnota 2550 startujících by nebyla reálná z několika důvodů:

Ne vždy se mi přihlásí plný počet závodníků v každé kategorii a ne vždy se jedou všechny kategorie z důvodu neobsazenosti, nebo hlídky se například na domácí scéně pevně jezdí jen na mistrovství republiky.

A důvod mnohem vážnější, že při počtu 2550 startujících by se závod při minimálním startovním intervalu nestihl odjet v jeden den, natož pak za denního světla, od kterého je dokonce odvozena denní doba, po kterou může závod ve vodním slalomu probíhat.

Z těchto důvodů jsem se rozhodl teoreticky spočítanou velikost potřebné paměti pouze zaokrouhlit směrem nahoru na nejbližší vyráběnou velikost paměti, tedy na 32 kB.

Proto jsem se rozhodl pro paměť EEPROM 28C256B. Starší typ paměti EEPROM před modernějším řešením s FLASH pamětí jsem upřednostnil z důvodu, že u obchodního zástupce GM electronic, odkud jsem jednotlivé el. součástky objednával, neměli paměť FLASH v provedení v pouzdře DIP, které je vhodné pro testování v nepájivém kontaktním poli. Navíc paměť EEPROM pro své smazání vyžaduje signál o rozdílném napětí 12 V, než při zápisu či čtení. Z tohoto důvodu je smazání této paměti komplikovanější a data by měly být více v bezpečí. Schéma zapojení paměti 28C256B je na obr. 24.

2.5.7.1. Charakteristické parametry 28C256B:

- napájení 5 V+-10 %
- proudový odběr v aktivním stavu 50 mA
- přístupový čas 150 ns
- pouzdro DIP28
- kapacita 32000 x 8 b, tedy celkem 32 kB

Uložené záznamy se poté budou moci uchovávat v digitální podobě ať již v původním zakódovaném tvaru, tedy tak jak budou uloženy v záznamovém zařízení, nebo v dekódovaném tvaru, nejspíše v textovém formátu, popřípadě ve formátu některého tabulkového editoru, odkud se pak případně budou moci vytisknout k založení v papírové podobě. Počítám s tím, že při zavádění nového systému do provozu, bude nový systém ze začátku běžet paralelně se stávajícím systémem, pro lepší nalezení a doladění případných chyb.

2.5.8. Napájení

Vzhledem k požadovanému napájecímu napětí 5 V pro všechny výše zmíněné komponenty záznamového zařízení a předpokládanému odběru do 160mA včetně podsvětlení displeje a kontrolky napájení, jsem se rozhodl napájecí zdroj záznamového zařízení postavit na integrovaném obvodu 7805, což je stabilizátor pevného napětí 5 V. Jde navíc o stejný typ stabilizačního obvodu, který používá pro přídavný zdroj i vývojový kit USBKIT a tak jde o osvědčený způsob napájení.

2.5.8.1. Parametry stabilizačního obvodu 7805:

- výstupní napětí 5 V
- výstupní proud maximálně 1 A.
- vstupní napětí 7 až 30 V (tedy stabilizátor má úbytek napětí 2 V)

Na základě parametrů stabilizátoru by pak zařízení mohlo být napájeno z jakéhokoliv zdroje stejnosměrného napětí od 7 do 30 V s dostatečným výstupním proudem aspoň 160 mA. V případě použití diody s funkcí ochrany proti přepólování od 8 V. Závod ve své nejdelší podobě dle pravidel může trvat od 8:00 do 19:00 hodin, tedy 11 hodin. Kapacita baterie by měla být nejméně 11*160 = 1760 mAh. Těchto a lepších kapacit dosahují akumulátory typu AA, při zachování přijatelné ceny. Z tohoto důvodu do záznamového zařízení umístil držák baterie A305, který je pro 8 baterií typu AA s výstupním napětím 9.6 V. Kapacita tohoto napájecího zdroje je dána typem použitých baterií, já bych doporučil baterii Sanyo AA NiMH 2500 mAh s cenou okolo 70,- Kč/kus, kde výrobce udává životnost až 19 let a udržení 90 % kapacity po 12 měsících skladování a 75 % po 60 měsících skladování. Baterie bych nabíjel mimo tělo záznamového zařízení, kde bude moci být nabíjena každá samostatně a tím zajištěna lepší kvalita nabíjení. Ze života vím, že baterie, hlavně vlivem svého stárnutí, dokáží vypovědět svou funkci v ten nejméně vhodný okamžik, z tohoto důvodu jsem vybavil zařízení i konektorem pro připojení externího napájecího zdroje.

2.5.9. Ostatní příslušenství pro záznamové zařízení

Všechny výše zmíněné komponenty záznamového zařízení bude potřeba umístit do odpovídajícího obalu. Vzhledem k rozměru displeje 84x44x10mm, rozměru klávesnice 65x65x7mm a rozměru plošného spoje 82x88x45mm, potřebuje zařízení prostor o rozměrech minimálně 128x84x55mm. Z tohoto důvodu jsem vybral plastovou krabičku od společnosti Valleman F-KVWCAH285, která má rozměry 190x95x55mm a je tvořena dvěma kusy, které se k sobě spojí 4 vruty. Jde tedy o rozebíratelný spoj.

Pro připojení výše zmíněných konektorů superseal bude zapotřebí plastovou krabičku dovybavit průchodkou pro kabel. Vybral jsem průchodku KSS F0604AG-16zr pro kabel od 5 do 10mm s krytím IP68.

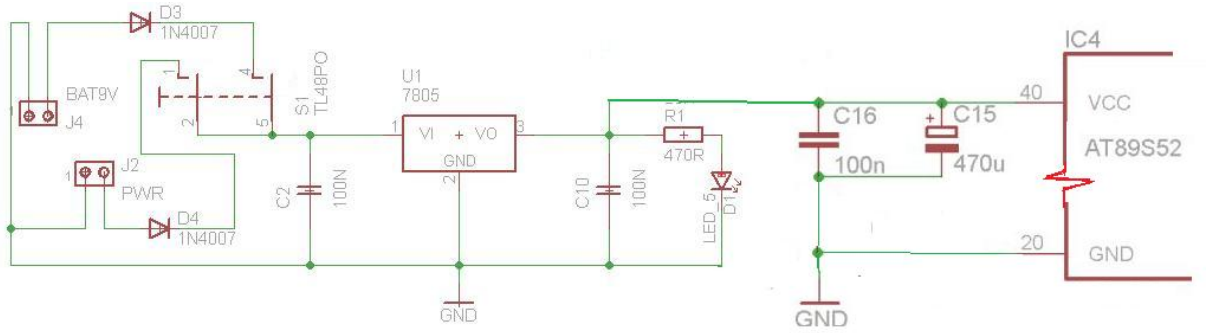
Pro ovládání záznamového zařízení budu potřebovat i vypínač pro zapnutí a vypnutí zařízení, tlačítko pro restart zařízení a vypínač pro podsvětlení displeje. Pro zajištění krytí jsem se rozhodl pro hmatník pro tlačítko i vypínač SSM27, který zajistí krytí zařízení na úroveň IP65. Pro restart bych použil tlačítko červené a pro funkci zapni - vypni vypínač zelený, pro osvětlení displeje pak tlačítko žluté.

3. Hardwarová realizace systému

V této kapitole popíši potřeby jednotlivých komponentů a z toho vyplývající způsob jejich zapojení. Ze zapojení jednotlivých komponentů dále odvodím zapojení celého záznamového zařízení a navrhnu plošný spoj pro záznamové zařízení. Pro návrh plošného spoje použiji program Eagle verze 6.5.0. Eagle je produkt firmy CadSoft a je dostupný s určitými omezeními i ve freeware verzi. Freeware verze dovoluje návrh maximálně dvouvrstvého plošného spoje do rozměrů 100x80 mm ze schématu, které musí být zakresleno na jednom listě. Tato omezení programu Eagle by neměla být limitujícími faktory pro návrh záznamového zařízení. Velkou výhodou tohoto programu je, že je pro něj volně dostupné velké množství knihoven s elektrosoučástkami. To velmi usnadňuje jak tvorbu schémat zapojení, tak i samotný návrh plošného spoje pro dané zařízení. Podrobnější informace o programu Eagle jsou dostupné. [11][21]

3.1.Napájení mikropočítače

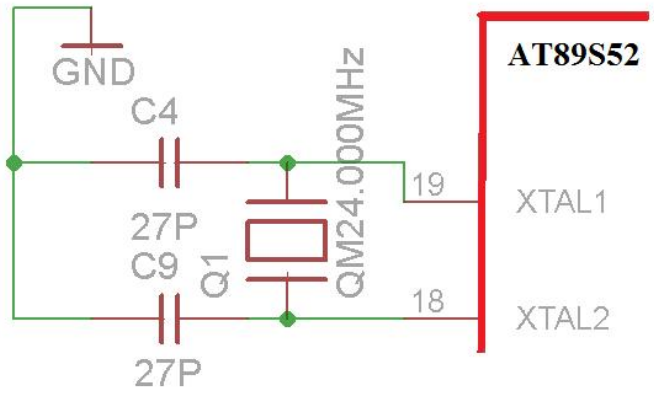
Lidstvo doposud nevymyslelo zařízení, které by bylo schopné pracovat bez zdroje energie a tak se ani záznamové zařízení neobejde bez zdroje napájecí energie. Systém napájení je zobrazen na obr. 19, kde J2 je konektor pro připojení stejnosměrného vnějšího zdroje 8 až 30 V minimální proud 200 mA a konektor J4 umožňuje připojení vnitřní baterie, vis kapitola 2.5.8.1. Vypínač S1 slouží pro připojení či odpojení obou zdrojů. Samotné zapojení integrovaného obvodu 7805, to se hlavně týká kondenzátorů C2 a C10, je převzato z USB51KIT, kde se jeho funkce osvědčila a vychází z katalogového listu obvodu 7805. Odpor R1 spolu s LED diodou D1 slouží pouze pro signalizaci připojeného napájecího napětí. Kondenzátor C16 je filtrační a slouží k odfiltrování vysokých frekvencí na napájení, naopak kondenzátor C15 zálohuje napájení proti krátkým výpadkům napájení. Obvod 7805 je spojen s chladičem V4330N.



Obrázek 19 Schéma napájení záznamového zařízení

3.2. Zapojení vnitřního oscilátoru

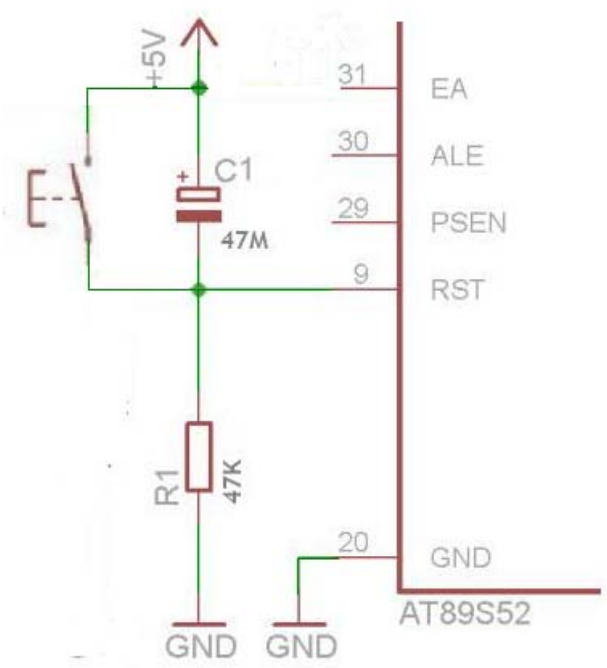
Hodinový kmitočet mikropočítače je dán typem připojeného krystalu, nebo keramického rezonátoru. Rozhodl jsem se pro krystal definující kmitočet 24 MHz. Z důvodu, že s tímto kmitočtem pracuje i USB51KIT a všechny přípravky pro záznamové zařízení tedy byly odzkoušeny s touto frekvencí. Zapojení krystalu je zobrazeno na obr. 20, hodnota kondenzátorů C4 a C9 je předepsána katalogovým listem mikropočítače [7] na hodnotu 30 pF ± 10 pF, tento požadavek hodnota 27 pF zcela splňuje.



Obrázek 20 Schéma připojení krystalu k vnitřnímu oscilátoru

3.3. Resetovací obvod

Funkci vstupu RTS jsem popsal již v teoretické části v kapitole 1.4.2.3.2. Výše zmíněný požadavek mikropočítače je řešitelný jednoduchým RC obvodem, vis obr. 21. Tlačítko ve schématu slouží pro vyvolání restartu záznamového zařízení obsluhou.

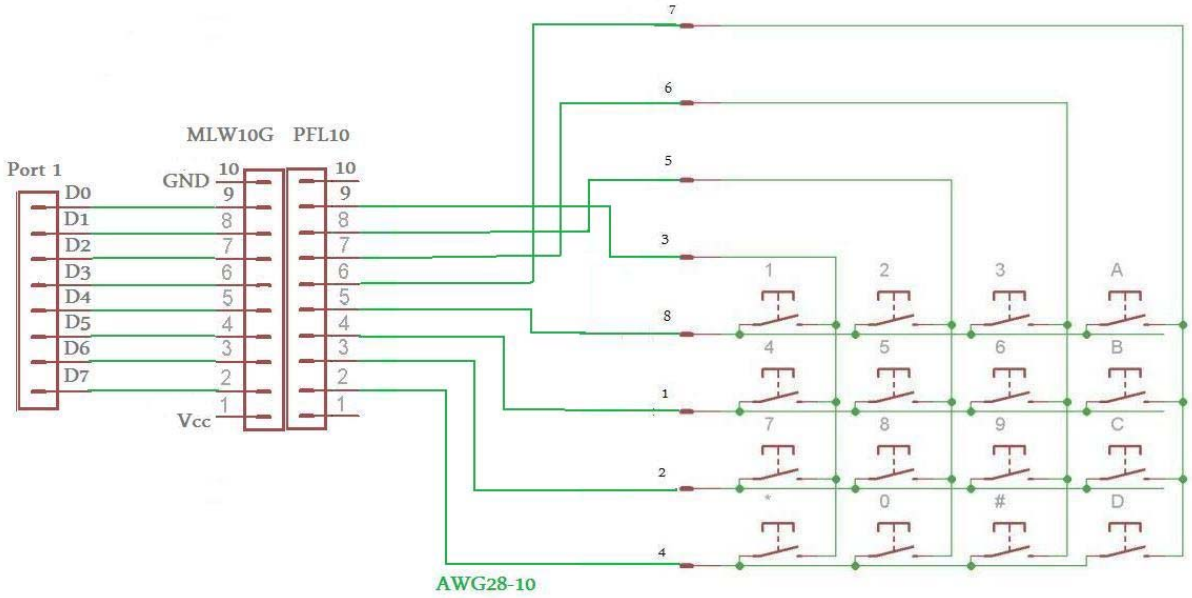


Obrázek 21 Schéma resetovacího obvodu

3.4. Zapojení klávesnice

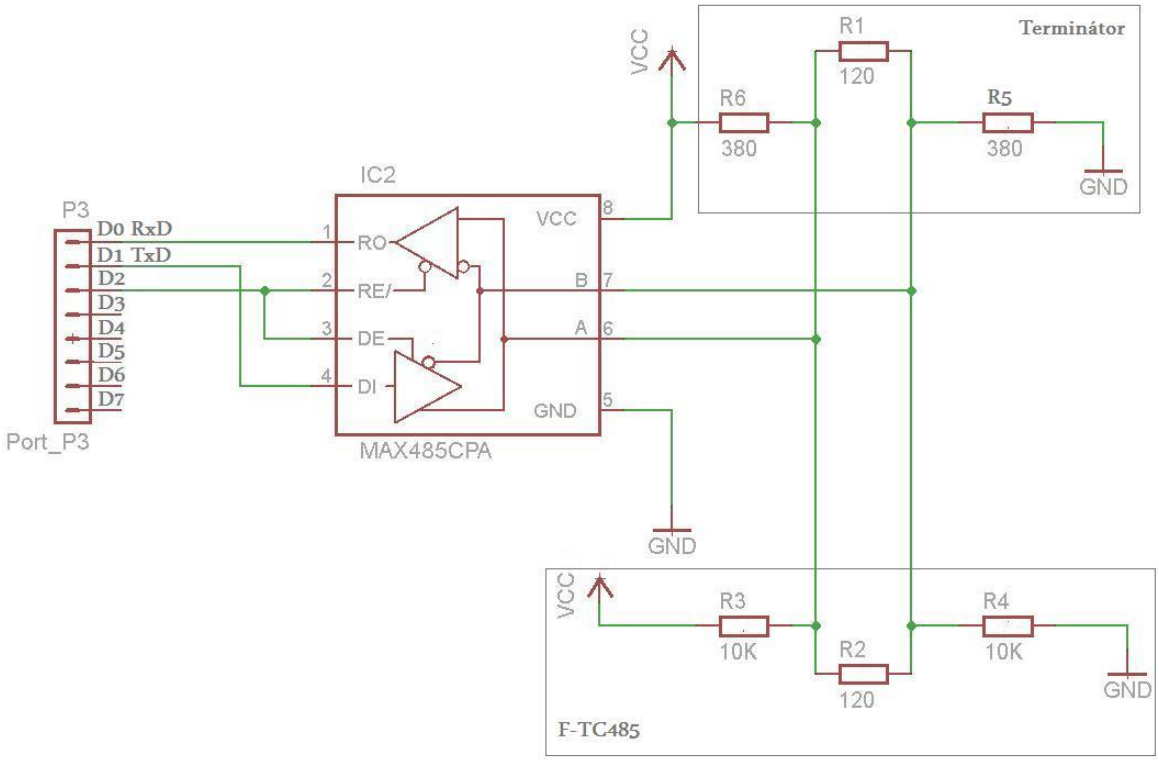
Na obrázku 22 je znázorněno připojení klávesnice k mikropočítači k portu 1 přes konektory MLW10G a PFL10, ze kterého spoj pokračuje 10 žilovým plochým kabelem AWG28-10. Konektor MLW10G je použit z důvodu, že těmito konektory je osazen i USB51KIT, čímž je zaručena kompatibilita připojení mezi záznamovým zařízením a

vývojovým kitem, na kterém byla klávesnice testována. Zapojení klávesnice je převzato z literatury [14], aby pro její programové ovládání bylo možno použít obslužnou funkci z téže knihy.



Obrázek 22 Připojení klávesnice

3.5. Zapojení vysílače RS-485



Obrázek 23 Zapojení vysílače RS-485

RO – výstup přijatých dat
/RE - negovaný vstup povolení příjmu, když /RE= log. 0, je příjem povolen.
DE – vstup povolení vysílání, když DE = log. 1, je povoleno vysílání.
DI – vstup pro odesílaná data
Vcc – napájecí kladné napětí 5 V±5 %
B – B vodič, nebo také „ – „ vodič sběrnice RS-485
A – A vodič, nebo také „ + „ vodič sběrnice RS-485
GND – zem napájení
Terminátor – zakončovací obvod sběrnice RS-485
F-TC485 – vstupní zapojení převodníku F-TC485

Jak je patrné z obr. 23 vysílač MAX485 převádí napěťový rozdílový vstupní signál na logiku TTL a přiveden na vstup sériové linky mikropočítače a obráceně vysílaná data z mikropočítače po sériové lince převádí MAX485 na rozdílový napěťový signál, který vyšle na sběrnici. Výhodou ovládání MAX485 je, že povolení příjmu je na rozdíl od povolení vysílání negovaný signál, tedy tyto signály pracují s přesně obrácenou logikou, a proto mohou být ovládány pouze jedním výstupem mikropočítače v případě, kdy nepožadujeme nastavení stavu vysoké impedance. Logika funkce MAX485 je dobře patrná z tab. 8.

Terminátor je v tomto případě jednoduchý zakončovací obvod, tvořený, jak je vidět z obr. 23, pouze třemi rezistory R1, R5 a R6. R1 je zakončovací rezistor sběrnice pro nastavení odpovídající impedance sběrnice. Rezistory R5 a R6 slouží k definování klidového stavu sběrnice. Tyto rezistory jsou rovněž součástí převodníku F-TC485, který je druhým

koncem sběrnice a odpory R3 a R4 mají výrobcem danou hodnotu 10 kΩ. Rezistor R2 má stejnou funkci jako rezistor R1.

Vysílač

INPUT D	ENABLE DE	OUTPUTS A	OUTPUTS B
H	H	H	L
L	H	L	H
X	L	Z	Z

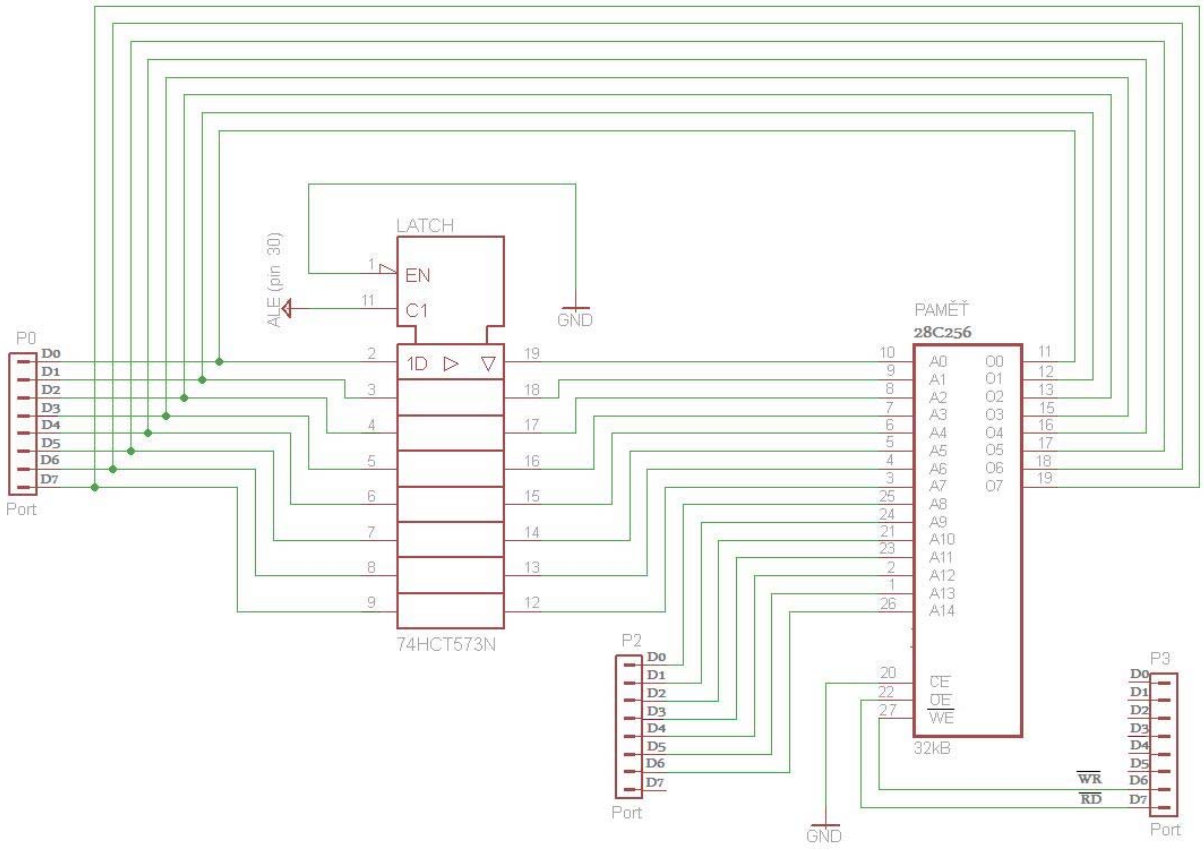
Přijímač

DIFFERENTIAL INPUTS A–B	ENABLE $\overline{RE}$	OUTPUT R
$V_{ID} \geq 0.2$ V	L	H
-0.2 V $< V_{ID} < 0.2$ V	L	?
$V_{ID} \leq -0.2$ V	L	L
X	H	Z
Open	L	?

Tabulka 8 Logika funkce obvodu MAX485 zdroj [19]

H – log. 1
L – log. 0
? – nedefinovaný stav
X – hodnota vstupu v dané situaci nezáleží
Z – vysoká impedance

3.6. Zapojení externí paměti dat



Obrázek 24 Schéma připojení externí paměti dat

Jak je vidět na obr. 24, pro zapojení paměti potřebuji i pomocný osmibitový Latch obvod 74HC573N, který je řízen signálem ALE z mikropočítače a slouží k multiplexování spodní části adresy a dat čtených či zapisovaných do paměti na portu P0. Signál ALE má frekvenci rovnu 1/6 hodinového kmitočtu mikropočítače. Pokud má signál ALE úroveň log. 1 jeho výstup se rovná jeho vstupu, při hodnotě signálu log. 0 na výstupu zůstává zachována poslední platná hodnota až do okamžiku změny signálu ALE. Na port P2 je zapisován horní bajt adresy, který musí být přítomen po celou dobu práce s pamětí. O tom jestli se z paměti čte, nebo zapisuje, rozhodují signály /WR pro zápis a /RD pro čtení. Jde o negativní logiku, tedy ke čtení nebo zápisu dochází, když je daný signál v úrovni log. 0. Mimo napájení potřebuje paměť a obvod LATCH i signál ENABLE u LATCH na pinu EN a u paměti pin CE, tyto signály mají také negativní logiku a proto jsou oba trvale připojeny na GND. Tedy je na nich úroveň log. 0, funkce těchto obvodů je trvale povolena.

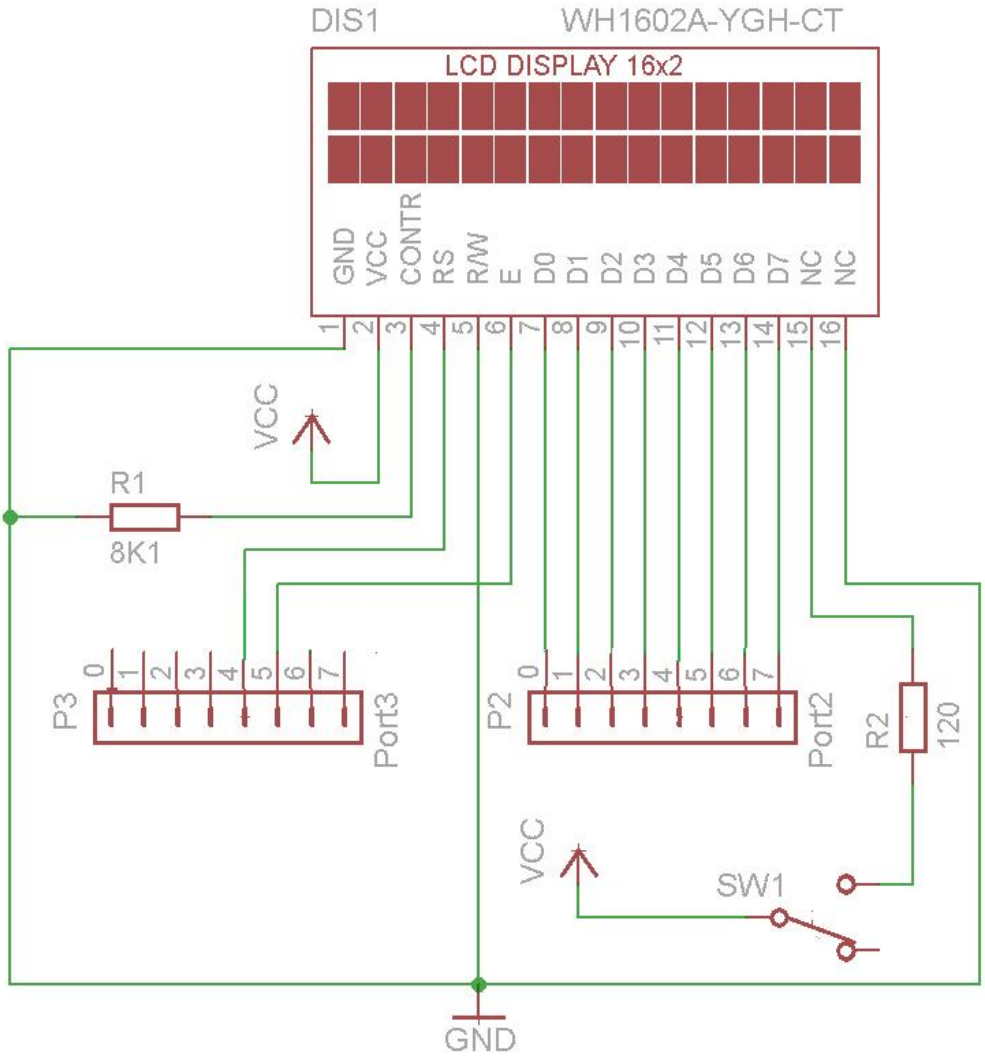
Pro udržení vyšší přehlednosti schématu jsem na portu P0 vynechal připojení pul-up odporů, které je uvedeno následně v celkovém zapojení a je realizováno odporovou sítí 8x10 kΩ.

3.7. Zapojení displeje

Se zapojením displeje teď nastává problém, na P0 a P2 je připojena vnější paměť dat, na P1 klávesnice a na P3 zbývají poslední 3 volné piny. Zde jsem se nechal inspirovat zapojením

pro dynamické řízení sedmisegmentového displeje, kde je jedna datová sběrnice pro větší počet displejů a ke každému displeji jde jeden řídící signál, který mu říká, že data přítomna na datové sběrnici jsou právě pro něj. Stejně tak i paměť pro operaci s daty potřebuje přítomnost jednoho ze signálu /RW nebo /RD i vybraný displej má vstup pro signál ENABLE, který říká displeji, že na datové sběrnici jsou přítomna platná data.

Proto jsem se rozhodl, že port P2 bude společnou datovou sběrnicí pro horní bajt adresy paměti, tak i pro data displeje. Vzhledem k tomuto není zapotřebí šetřit šířkou přenášených dat pro displej a použiji pro něj osmibitovou komunikaci, která má nižší nároky na řízení přenášených dat pro displej. Principiální schéma zapojení displeje WH1602-YGH-CT je znázorněno na obr. 25 a význam jednotlivých vstupů displeje popisuje tabulka 9.



Obrázek 25 Zapojení displeje

Číslo vývodu	Signál	Funkce
1	GND	Zem
2	VCC	Napájecí napětí
3	CONTR	Nastavení kontrastu displeje
4	RS	Příkaz (0), data (1)
5	R/W	Čtení (1), zápis (0) dat nebo příkazu
6	E	Vstup povolen
7	DB0	Data/příkaz (dolní bit)
8	DB1	Data/příkaz
9	DB2	Data/příkaz
10	DB3	Data/příkaz
11	DB4	Data/příkaz
12	DB5	Data/příkaz
13	DB6	Data/příkaz
14	DB7	Data/příkaz (horní bit)
15	A	Anoda podsvětlovací LED (dle typu)
16	K	Katoda podsvětlovací LED (dle typu)

Tabulka 9 Zapojení displeje WH1602-YGH-CT zdroj [33]

Jak je vidět na obr. 25 pin 5, určující zda jde o čtení či zápis dat, ten je připojen trvale na GND z důvodu, že na displej budu pouze zapisovat. Odpor R1 nastavuje kontrast zobrazení, jeho hodnotu jsem určil experimentálně. Nejdříve jsem měl na daném pinu připojen trimr, jímž jsem požadovaný kontrast nastavil a ten následně vyměnil za pevný odpor požadované hodnoty. Stejně tak i odpor R2, který určuje jas podsvětlení displeje. Funkce SW1 je zřejmá a slouží pro zapnutí podsvětlení displeje.

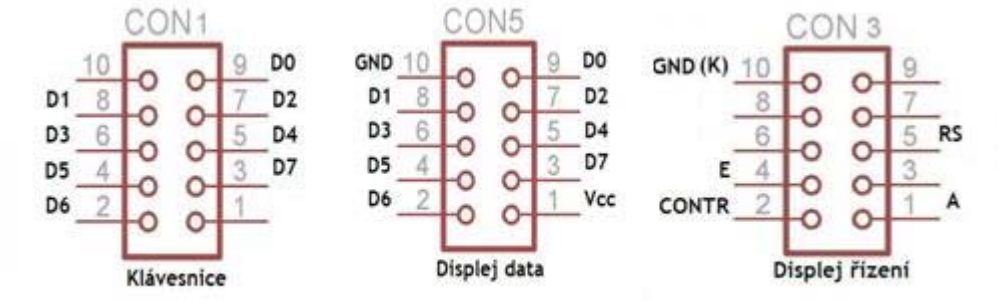
Displej bude připojen dvěma plochými kabely AWG28-10 přes konektory MLW10G a PFL10 stejně jako u připojení klávesnice vis celkové zapojení záznamového zařízení na obr. 26.

3.8.Celkové zapojení záznamového zařízení

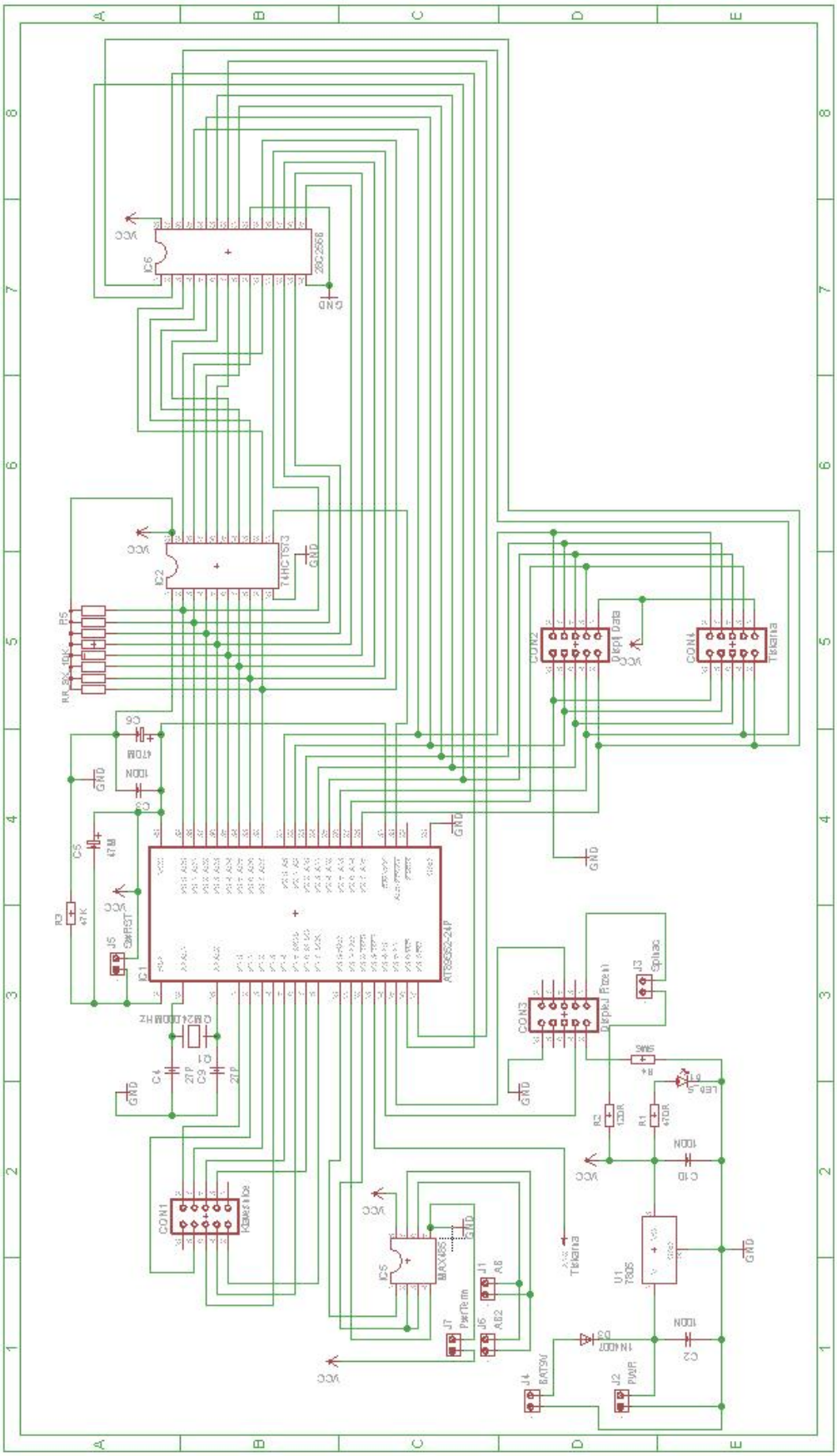
3.8.1. Schéma

Celkové schéma záznamového zařízení je zobrazeno na obr. 27. Toto schéma je již přizpůsobeno pro návrh plošného spoje. Tedy místo displeje a klávesnice jsou ve schématu pouze konektory MLW10G pro připojení těchto zařízení, zapojení těchto konektorů je zobrazeno na obr. 26. Popis zapojení displeje je popsán v tab. 9, u zapojení klávesnice je zachována stejná logika popisu. Na desce plošného spoje je navíc i konektor MLW10G s názvem Tiskárna, jde o paralelní připojení ke konektoru displeje data. Tento konektor jsem si na desce ponechal pro případné připojení tiskárny. Pro připojení vypínačů záznamového zařízení a podsvětlení displeje jako i pro tlačítko restartu jsou použity konektory

PSH02-02PG. Stejné konektory jsou použity i pro připojení výstupního kabelu záznamového zařízení a pro možnost připojení záložního zdroje napájení je řešena stejným typem konektoru.



obrázek 26 Detail zapojení konektorů MLW10G

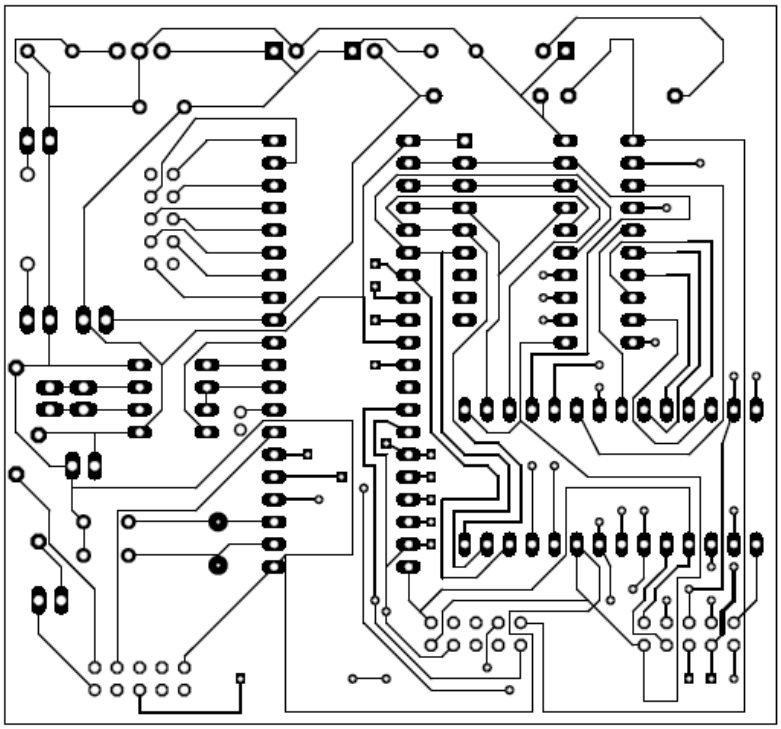


obrázek 27 Schéma zapojení plošného spoje záznamového zařízení

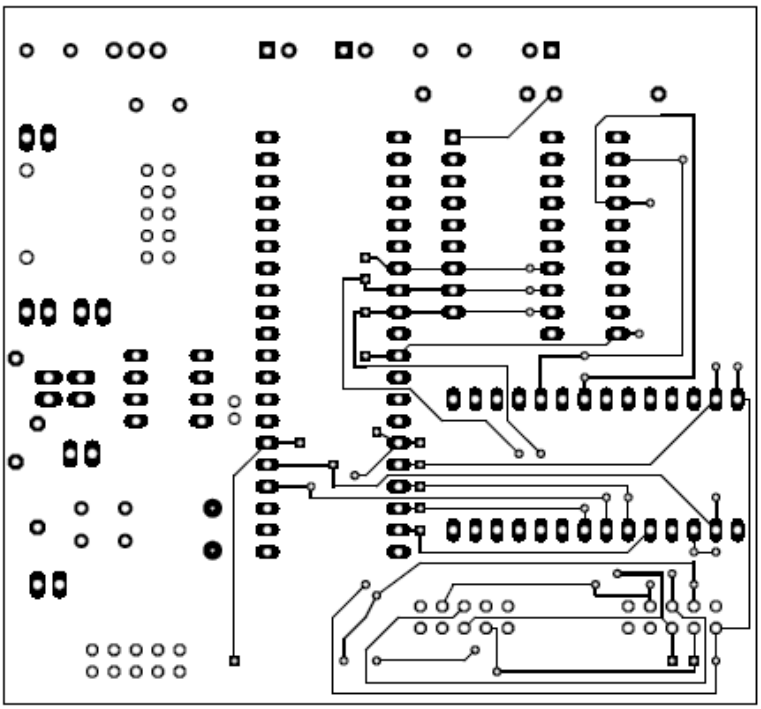
3.8.2. Plošný spoj

Při realizaci plošného spoje jsem se rozhodl pro dvouvrstvý plošný spoj. Návrh plošného spoje je zobrazen na obr. 28, spodní vrstva plošného spoje na obr. 29, horní vrstva plošného spoje a na obr. 30 je znázorněn osazovací plán plošného spoje. Všechna vyobrazení jsou z pohledu ze strany součástek. Podkladové výkresy jsou součástí doprovodného CD.

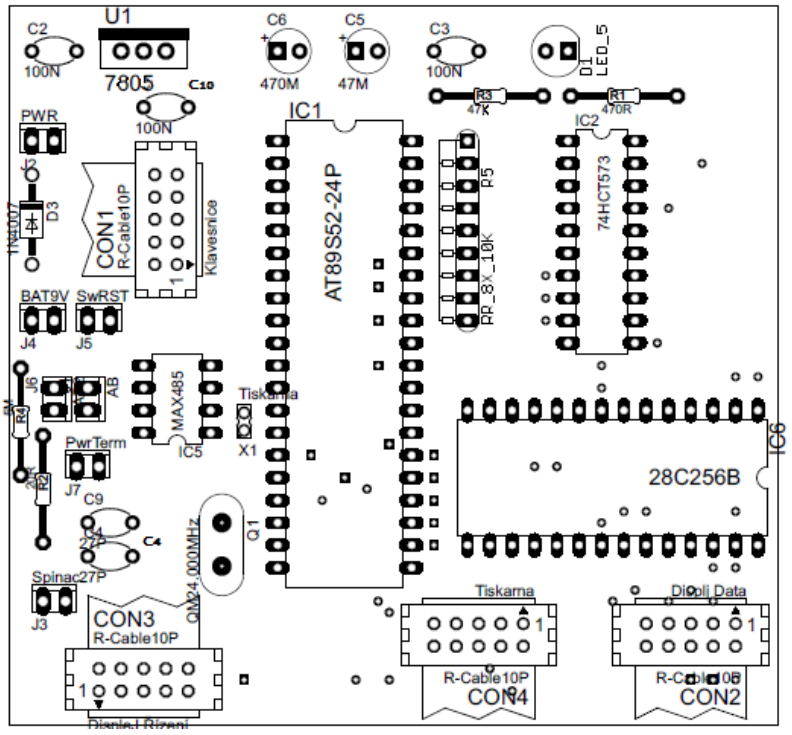
Pro osazení integrovaných obvodů jsem použil patice pro dané obvody, tedy DIL8, DIL20, DIL28 a DIL40, což mi umožnilo některé průchody mezi vrstvami plošného spoje umístit právě pod těmito obvody, z tohoto důvodu je zapotřebí před napájením patic propojit tyto průchody mezi vrstvami. Vzhledem k tomu, že záznamové zařízení neobsahuje interface pro změnu softwarového vybavení mikropočítače, je jeho umístění v patici DIL40 nutností pro případný přesun mikropočítače do programátoru. Rozpis použitých elektronických součástek vis tab. 10.



obrázek 28 Spodní vrstva plošného spoje, pohled ze strany součástek



obrázek 29 Horní vrstva plošného spoje, pohled ze strany součástek



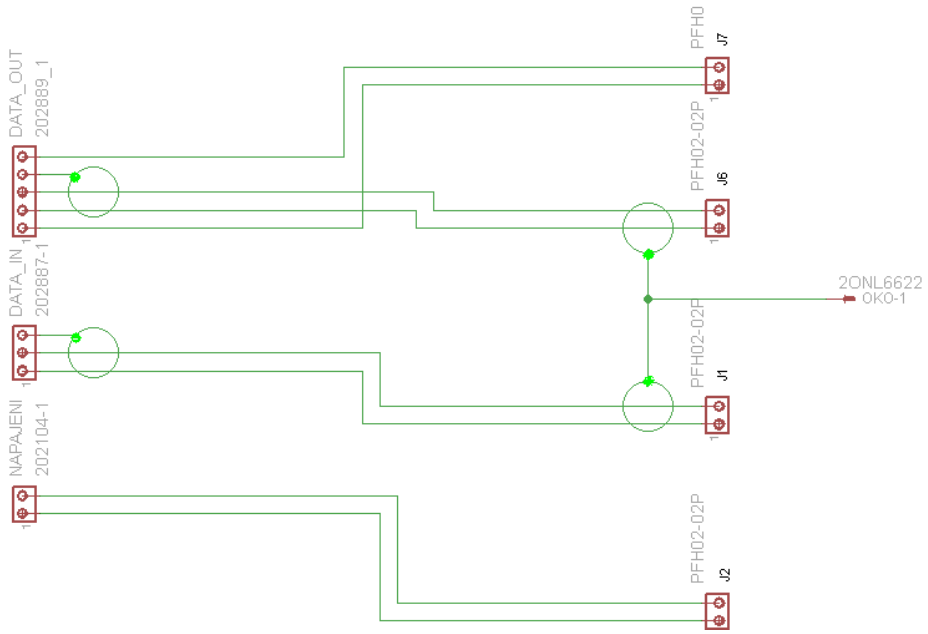
obrázek 30 Osazovací plán plošného spoje, pohled ze strany součástek

Značení	Součástka	Kusů
C2, C3, C10	CK 100N X7R	3
C6	E470M/25 V	1
C5	E47M/16 V	1
C4, C9	CK27P/500V	2
R5	RR 8X10K 2 %	1
R4	5M6	1
R2	120R	1
R3	47K	1
R1	470R	1
Q1	QM24,000MHz	1
D1	LED 5 mm	1
D3	1N4007	1
CON1, CON2, CON3, CON4	MLW10G	4
J1, J2, J3, J4, J5, J6, J7	PSH02-02PG	7
U1	7805 + chladič V4330N	1
IC1	AT89S52-24P + DIL40	1
IC2	74HCT573 + DIL20	1
IC5	MAX485CPA + DIL8	1
IC6	28C256B + DIL28	1

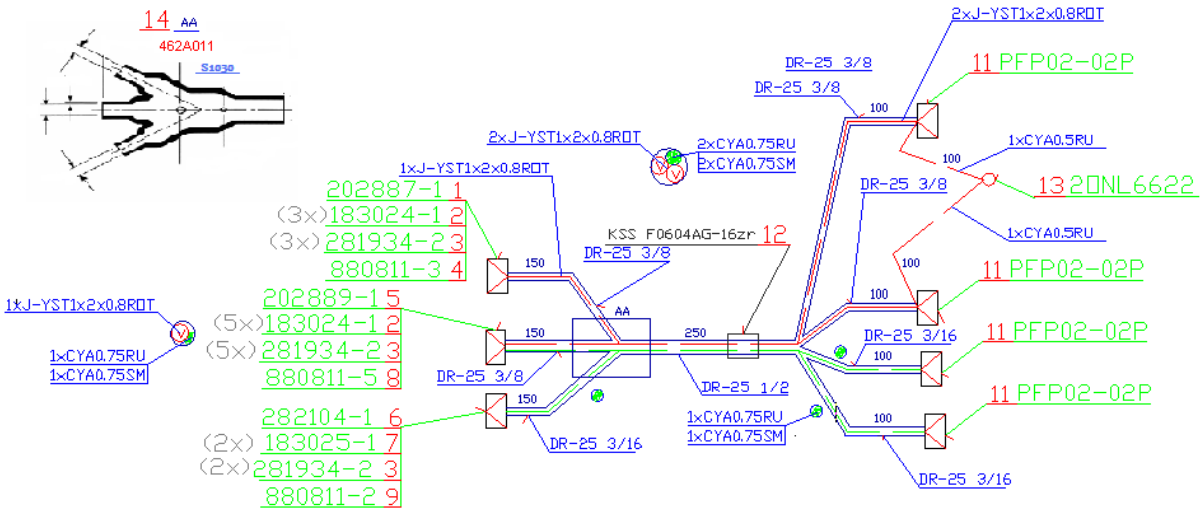
Tabulka 10 Soupis el. součástek pro osazení plošného spoje záznamového zařízení

3.8.3. Připojovací kabeláž záznamového zařízení

3.8.3.1. Připojovací kabel záznamového zařízení



Obrázek 31 Schéma zapojení připojovacího kabelu záznamového zařízení

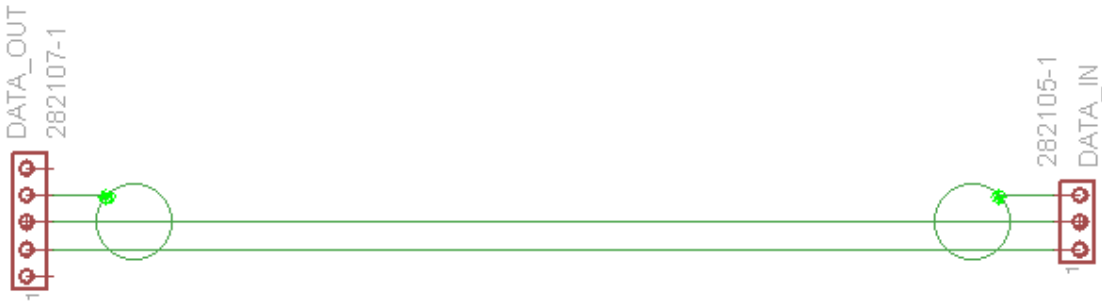


obrázek 32 Výkres připojovacího kabelu záznamového zařízení

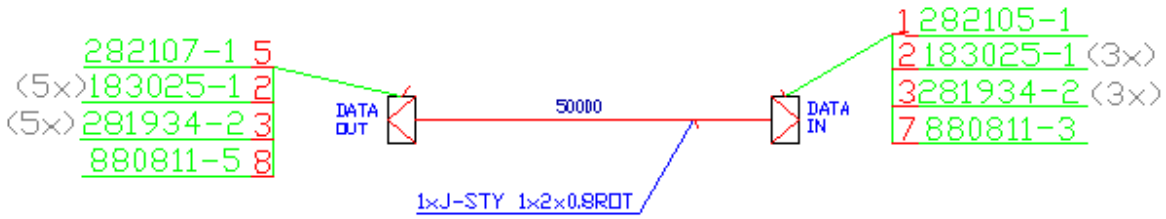
Při montáži kabelu, vis obr. 32, je zapotřebí osadit průchodku pozice 12 před osazením konektorů pozice 11 a 13. Při stavbě kabelu je zapotřebí počítat se zkrácením kabelu vlivem zkroucení vodičů.

Pozice	P/N	Název	Množství
1	202887-1	Konektor (3piny, zásuvka)	1 kus
2	183024-1	Kontakt	8 kusů
3	281934-2	Těsnění	10 kusů
4	880811-3	Koncovka (gumová)	1 kus
5	202889-1	Konektor (5piny, zásuvka)	1 kus
6	282104-1	Konektor (2piny. vidlice)	1 kus
7	183025-1	Kontakt	2 kusy
8	880811-5	Koncovka (gumová)	1 kus
9	880811-2	Koncovka (gumová)	1 kus
11	PFP02-02P	Konektor	4 kusy
12	KSS F0604AG-16zr	Průchodka	1 kus
13	2 ONL 6622	Oko kabelové	2 kusy
14	465A011	Tvarovka	1 kus
	DR-25 3/16	Hadice smrštitelná	350 mm
	DR-25 3/8	Hadice smrštitelná	500 mm
	DR-25 1/2	Hadice smrštitelná	250 mm
	J-Y(ST)Y 1x2x0.8ROT	Kabel stíněný kroucený pár	1000 mm
	CYA0,75RU	Vodič červený	1000 mm
	CYA0,75SM	Vodič modrý	1200mm
	S1030	Lepidlo RAYCHEM	Dle potř.
		Isolační PVC páska 15 mm	Dle potř.

3.8.3.2. Kabely k propojení postů



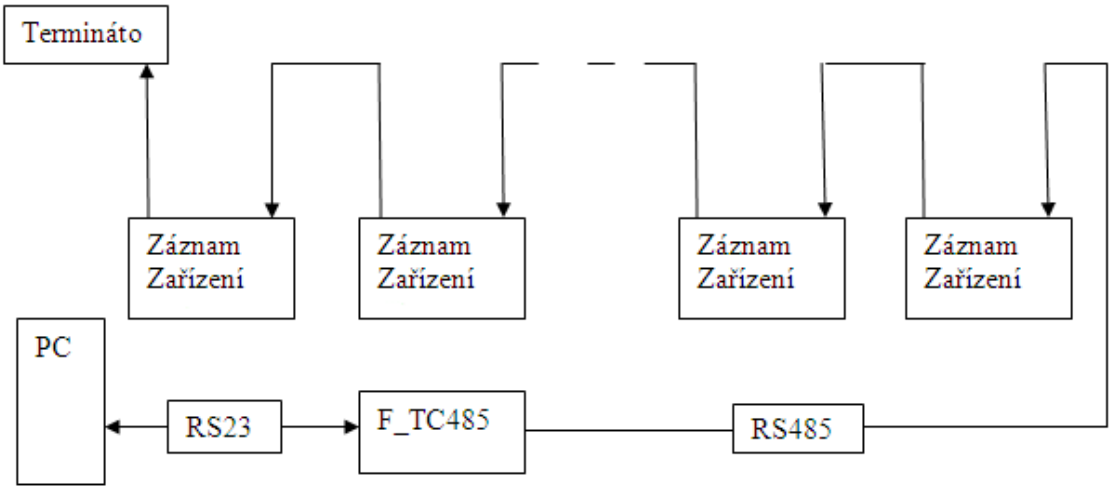
Obrázek 33 Schéma zapojení propojovacích kabelů



obrázek 34 Výkres propojovacího kabelu

Sada kabelů k propojení jednotlivých záznamových zařízení by se měla skládat z 8 kabelů a z toho by 4 kabely měly mít délku 75 m, 3 kabely 50 m a jeden délku 250 m. Zapojení těchto propojovacích kabelů je znázorněno na obr. 33 a jejich technické provedení na obr. 34, na kterém je ilustrativně zobrazen pouze kabel s nejkratší délkou, provedení delších kabelů je totožné s kabelem na obr. 34, jen se liší délka.

Způsob propojení jednotlivých záznamových zařízení je znázorněn na obr. 35, kde na blokovém schématu šipka značí konektor DATA OUT na sběrnici RS485.



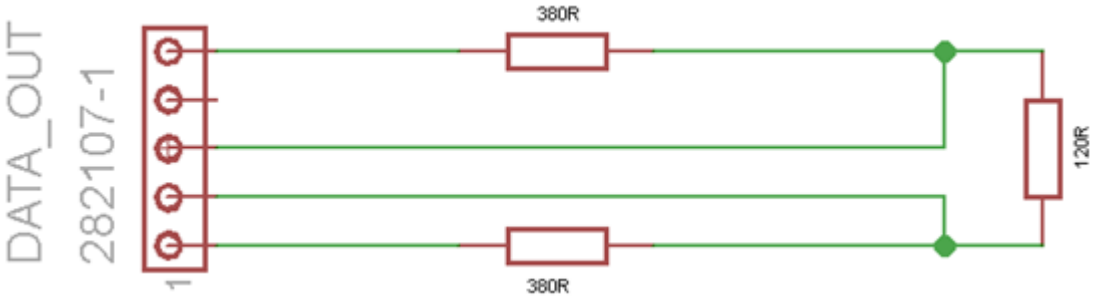
Obrázek 35 Blokové schéma propojení záznamových zařízení

Pozice	P/N	Název	Množství
1	282105-1	Konektor (3piny, vidlice)	1kus
2	183025-1	Kontakt	8 kusů
3	281934-2	Těsnění	8 kusů
5	282107-1	Konektor (5piny, vidlice)	1 kus
7	880811-3	Koncovka (gumová)	2 kusy
8	880811-5	Koncovka (gumová)	1 kus
	J-Y(ST)Y 1x2x0.8ROT	Kabel stíněný kroucený pár	50 m, (75 m, 250 m)

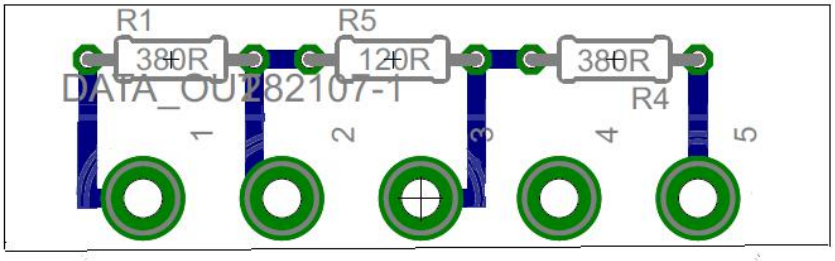
Tabulka 11 Soupis materiálu pro propojovací kabely

3.8.3.3. Zapojení terminátoru

Pro realizaci terminátoru jsem se rozhodl, tento jednoduchý obvod, vis obr. 36 a obr. 37, přichytit přímo ke konektoru smrštitelnou hadicí DR-25 1 ½, vis tab. 12.



obrázek 36 Schéma terminátoru (zakončovací obvod)



obrázek 37 Plošný spoj terminátoru, pohled ze strany součástek

Značení	Součástka	Množství
R1, R4	380R	2 kusy
R5	120R	1 kus
Data_OUT	Konektor 282107-1	1 kus
CYA0,75RU	Vodič červený	1 m
CYA0,75SM	Vodič modrý	1 m
DR-25 1 1/2	Hadice smrštitelná	1 m

Tabulka 12 Soupis materiálu pro terminátor

3.8.3.4. Kompletace

Tímto jsem popsal všechny jednotlivé část záznamového zařízení a konečná montáž se provede dle finálního soupisu záznamového zařízení, vis tab. 13.

P/N	Název	Množství	Poznámka
F-KVWCAH285	Krabička 190x95x55	1 kus	
SSM27	Hmatník	3 kusy	Pro spínače podsvětlení displeje, ON/OFF a Reset.
F-KV16KEY	Klávesnice	1 kus	Zapojení na obr. 22
WH1602-YGH-CT	LCD displej	1 kus	Zapojení na obr. 25, 26 a 27
	Osazený plošný spoj	1 kus	Kapitola 3.8.2
	Kabel připojovací	1 kus	Kapitola 3.8.3.1
PFL10	Konektor	3 kusy	Připojení klávesnice a displeje
AWG28-10	Kabel plochý 10 žil	0.5 m	Připojení klávesnice a displeje
P-B068BS	Spínač jednoduchý	1 kus	Podsvětlení displeje, zapojení na obr. 25, 26 a 27
P-B068ES	Spínač dvojitý	1 kus	ON/OFF zapojení na obr. 21 a 26
P-T250ROTA	Tlačítko	1 kus	Reset zapojení na obr. 21 a 26

P/N	Název	Množství	Poznámka
PFP02-02P	Konektor	5 kusy	K připojení spínačů podsvětlení displeje, ON/OFF a Reset a napájení z baterie BAT9V
A305	Držák baterie	1 kus	Pro akumulátory 8xAA
CYA0,75RU	Vodič červený	1m	K připojení spínačů podsvětlení displeje, ON/OFF a Reset a baterie
DR-25 3/16	Hadice smrštitelná	1 m	K připojení spínačů podsvětlení displeje, ON/OFF a Reset a baterie

Tabulka 13 Finální soupiska záznamového zařízení

4. Software

Pro tvorbu samotného zdrojového kódu zvolil programovací jazyk C, tedy konkrétně C51, který je dnes rozšířeným vyšším programovacím jazykem pro mikropočítače. Vzhledem ke skutečnosti, že jde o vyšší programovací jazyk, je mi práce v něm o něco bližší než práce v Assembleru.

Pro usnadnění tvorby programového vybavení záznamového zařízení jsem se rozhodl využít vývojové prostředí Keil µVision verze 4.02, které je přehledným a dobře vybaveným prostředím s intuitivním ovládáním a s rozsáhlou databází podporovaného hardwaru. Vývojové prostředí Keil µVision v sobě integruje řízení projektu v podobě různých výstupních profilů v rámci projektu, jako je například pro simulaci, emulaci a finální verzi EPROM. Výkonný editor se zvýrazněním syntaxe a interaktivní korekci chyb.

Program záznamového zařízení se skládá z hlavního programu a 20 funkcí, které jsou hlavním modulem využívány. Výjimku tvoří modul obsluhy přerušení, jenž naopak přeruší hlavní program a sám využívá jiné funkce.

Funkce umožňuje rozdělit program do kratších celků, které řeší dílčí problémy, tím se zjednoduší návrh programu a zajistí vyšší srozumitelnost zdrojového kódu.

Vzhledem k rozsáhlosti zdrojového kódu se v této části budu zabývat jen funkcemi programu, které jsou stěžejní pro komunikaci mikropočítače s ostatními periferiemi a zbylé funkce se nakonec pro využití daného hardwaru obrátí na jednu z funkcí pro komunikaci s okolím. Kompletní výpis zdrojového kódu s jeho vysvětlením je obsahem přílohy 1 a na doprovodném CD.

4.1.Funkce

4.1.1. Klávesnice

Tato funkce zajišťuje čtení stavu klávesnice a ze stavu sloupců v závislosti na přivedeném signálu do řádků klávesnice vypočítá pozici stisknuté klávesy. Poziční číslo stisknuté klávesy, vis obr. 13, kde je poziční číslo vyznačeno červeně. Funkce tedy vrací hodnotu 0 až 15 pro stisknutou klávesu a -1 pokud není stisknutá žádná klávesa. Základ funkce klávesnice je

převzat z literatury [14], kde je podrobně popsán. Zde popíši jen úpravu, kterou jsem v této funkci provedl a důvod, který mě k tomuto vedl.

Při testování klávesnice se mi pravidelně stávalo, že se mi při stisku klávesy vypsal větší počet znaků odpovídajících stisknuté klávese, někdy i více než 10, navzdory tomu, že mezi řádky 58 až 62, vis příloha 1, je umístěno softwarové ošetření zákmitů kontaktů tlačítek klávesnice v podobě smyčky, která čte stav klávesnice s časovou prodlevou dvakrát za sebou a nepustí program dále, dokud se oba vzorky neshodují. Problém se mi částečně podařilo vyřešit zvednutím času prodlevy mezi čtením stavu klávesnice, ale toto fungovalo jen částečně. Proto jsem na konec funkce klávesnice umístil podmínku, vis uvedený zdrojový kód pod textem, která nedovolí výpis dvou stejných znaků po sobě. Pokud potřebuji vypsat dva stejné znaky za sebou, vždy mezi nimi musí být uvolnění klávesy. Tato podmínka, ale potřebuje znát, která klávesa byla stisknutá před právě vyhodnocovanou klávesou, z tohoto důvodu jsem musel zavést globální proměnnou „predKod“, kde uchovávám kód předcházející stisknuté klávesy.

```
44)     if(kod==predKod)
45)         kod=0;
46)     else
47)         predKod=kod;
48)     return kod-1;
```

4.1.2. Funkce Povel

Tato funkce slouží k řízení displeje, způsob komunikace mikropočítače s displejem je uveden v tab. 14 a tab. 15, sloupec čas udává dobu, kterou displej potřebuje ke zpracování požadavku a pro správné provedení příkazu nesmí v tomto čase dostat jiný příkaz.

Instrukce	**Signál**										**Popis**	**čas**
	RS	**R/W**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**		
Smazat displej	0	0	0	0	0	0	0	0	0	1	Smaže displej a nastaví kurzor na pozici 0.	1.64 ms
Návrat na pozici 0	0	0	0	0	0	0	0	0	1	*	Nastaví adresu kurzor na pozici 0. Obsah DDRAM zůstane zachován.	1.64 ms
Nastavení módu	0	0	0	0	0	0	0	1	I/D	S	Nastaví směr pohybu kurzoru, a specifikuje posun displeje. Operace jsou prováděny během zápisu a čtení dat.	40 μs

Kontrola displeje zap/vyp	0	0	0	0	0	0	1	D	C	B	Zapne/vypne displej (D), zapne/vypne kurzor (C) a blikání kurzoru (B).	40 µs
Posun kurzoru, displeje	0	0	0	0	0	1	S/C	R/L	*	*	Pohyb kurzoru a posun displeje bez změny obsahu DDRAM.	40 µs
Nastavení funkce	0	0	0	0	1	DL	N	F	*	*	Délka rozhraní (DL), počet řádek displeje (N) a velikost fontu (F).	40 µs
Nastavení adresy CGRAM	0	0	0	1	CGRAM adresa						Nastaví adresu CGRAM. Data jsou přenesena po tomto nastavení.	40 µs
Nastavení adresy DDRAM	0	0	1	DDRAM adresa							Nastaví adresu DDRAM. Data jsou přenesena po tomto nastavení.	40 µs
Čtení příznaku Busy Flag a adresy	0	1	BF	CGRAM / DDRAM adresa							Čte příznak (BF), který indikuje provádění vnitřních operací a čte adresu CGRAM nebo DDRAM (v závislosti na předchozí instrukci).	0 µs
Zápis dat do CGRAM nebo DDRAM	1	0	zápis dat								Zapíše data do CGRAM nebo DDRAM.	40 µs
Čtení dat z CGRAM nebo DDRAM	1	1	čtení dat								Přečte data z CGRAM nebo DDRAM.	40 µs

Tabulka 14 Tabulka jednotlivých příkazů k ovládání displeje zdroj [16]

Název bitu	Popis	
I/D	0 - Snížení	1 - Zvýšení
S	0 - Není posun displeje	1 - Posun displeje při zápisu dat
D	0 - Vypnout displej	1 - Zapnout displej
C	0 - Vypnout kurzor	1 - Zapnout kurzor
B	0 - Vypnout blikání kurzoru	1 - Zapnout blikání kurzoru
S/C	0 - Posun kurzoru	1 - Posun displeje
R/L	0 - Posun doleva	1 - Posun doprava
DL	0 - 4bitová komunikace	1 - 8bitová komunikace
N	0 - 1 řádek	1 - 2 řádky
F	0 - 5x8 bodů	1 - 5x10 bodů
BF	0 - Operace je ukončena	1 - Operace probíhá

Tabulka 15 Význam nastavení jednotlivých bitu při ovládání displeje zdroj [16]

```
50)     void povel (unsigned char p)
51)     {
52)     RS=0;
53)     P2=p;
54)     E=1;
55)     cekej(1);
56)     E=0;
57)     }
```

Výše uvedenou funkci povel asi nemusím moc popisovat její chování je zřejmé z tabulek 14 a 15. Jen krátce hodnota povelu se funkci předává jako parametr „p“ a displej bera data na vstupu za platná v okamžiku sestupné hrany signálu E (Enable), tedy povoleni funkce displeje.

4.1.3. Funkce Vypiš znak

Jak název napovídá, funkce slouží k vypsání jednoho znaku na displej a její základ stejně jako u funkce povel vychází z tabulek 14 a 15. Jak je vidět, z níže uvedeného zdrojového kódu, základ této funkce se od funkce povel liší pouze v nastavení hodnoty signálu RS=1, který právě displeji říká, že nejde o povel, ale o zápis do paměti displeje.

```
58)   void vypisZnak (unsigned char kod)
59)   {
60)   unsigned char a;
61)   a=kurzor;
62)   RS=1;
63)   P2 = kod;
64)   E=1;
65)   cekej(1);
66)   E=0;
67)   kurzor=kurzor++;
68)   if (kod<48)
69)     kod=48;
70)   if (a<5)
71)     obsahD[a]=kod-48;
72)   if (kurzor>63)
73)     obsahD[a-59]=kod-48;
74)   cekej(40);
75)   return;
76)   }
```

Po samotném odeslání znaku na displej, který se funkci předává jako parametr „kod“, následuje inkrementace globální proměnné „kurzor“, v které je uložena hodnota reprezentující pozici kurzoru na displeji, z tohoto důvodu při vypsání znaku na displej se musí o jedničku navýšit.

Při zápisu znaku na displej počítám pouze s výpisem znaků reprezentující číslice 0 až 9, které v ASCII kódu mají hodnotu 48 až 57, vis tabulka s umístěním symbolů v paměti displeje, které právě vychází z rozložení znaků v ASCII tabulce v příloze 2. Proto podmínka na řádku 68 po vypsání znaku převede všechny znaky o nižší hodnotě na hodnotu právě 48, což zabrání při zápisu vypsaného znaku do globální proměnné „obsahD“ (obsah Displeje), kde se při ukládání zapsaný znak převádí právě odečtením hodnoty 48 na číselnou hodnotu, odpovídající hodnotě, kterou reprezentuje znak zapisované číslice, zabrání překročení rozsahu dané proměnné. Aby datové pole, reprezentující obsah displeje, nemuselo mít rozsah 79 pozic celého displeje, tedy 79 B, ukládám do něj jen oblasti, které později potřebuji ke zpracování záznamu, čímž snížím rozsah daného datového pole na hodnotu 21 B. O vypuštění nepotřebných údajů z obsahu displeje se starají podmínky na řádcích 70 a 72.

Kdybych měl v době psaní funkcí Povel a Vypiš znak zkušenosti, které jsem získal právě psaním tohoto programu, realizoval bych tyto dvě funkce pro svou podobnost jako funkci jednu nazvanou například Displej, která by mi pak zastřešovala celou komunikaci s displejem, ale v době kdy jsem se jazyk „C“ začínal učit, mi více vyhovovalo toto rozdělení do dvou funkcí.

4.1.4. Funkce Odešli

Jak název funkce napovídá jejím úkolem je odeslání jednoho bajtu přes rozhraní sériové linky a tím i přes vysílač RS485 na sběrnici RS485.

```
174)    void odesli (unsigned char p)
175)    {
176)    SBUF = p;
177)    while(TI!=1);
178)    TI=0;
179)    cekej(10);
180)    }
```

Tato funkce pracuje s plně duplexním rozhraním sériové linky, ale to by nebylo dostačující pro sběrnici RS485, které je poloduplexní a potřebuje přepínání směru komunikace, o to se v záznamovém zařízení stará signál DE, který je vyveden na vstupy vysílače RS485 DE a /RE, vis tab. 8. Pro povolení vysílání je potřeba nastavit signál DE na hodnotu „1“ a po odeslání opět shodit, aby zas zařízení mohlo přijímat data. Nastavení a shození signálu DE jsem do těla funkce odešli, neumístil z důvodu, že tato funkce je často volána v cyklu, kde by pak neustále docházelo k opětovnému nahazování a shazování tohoto signálu.

Jak je vidět z výše uvedeného zdrojového kódu funkce odešli, nejde o nic složitého. Pro odeslání bajtu přes sériovou linku stačí tento bajt vložit do registru pro příjem a vysílání znaku SBUF, zde se bajt předává jako parametr „p“. Pro správné odeslání znaku je zapotřebí s dalším odesíláním počkat až bude ukončeno odesílání předchozího bajtu, což je indikováno bitem indikace vyprázdnění vysílacího registru TI v registru SCON na jehož nastavení čeká cyklus WHILE. Po ukončení čekání je zapotřebí bit TI programově znulovat, aby mohlo dojít ke správné indikaci vyprázdnění zásobníku při dalším vysílání. Funkce čekej, na konci funkce je jen pro jistotu správného přenosu, zvyšuje rozestup mezi jednotlivými vysílanými bajty.

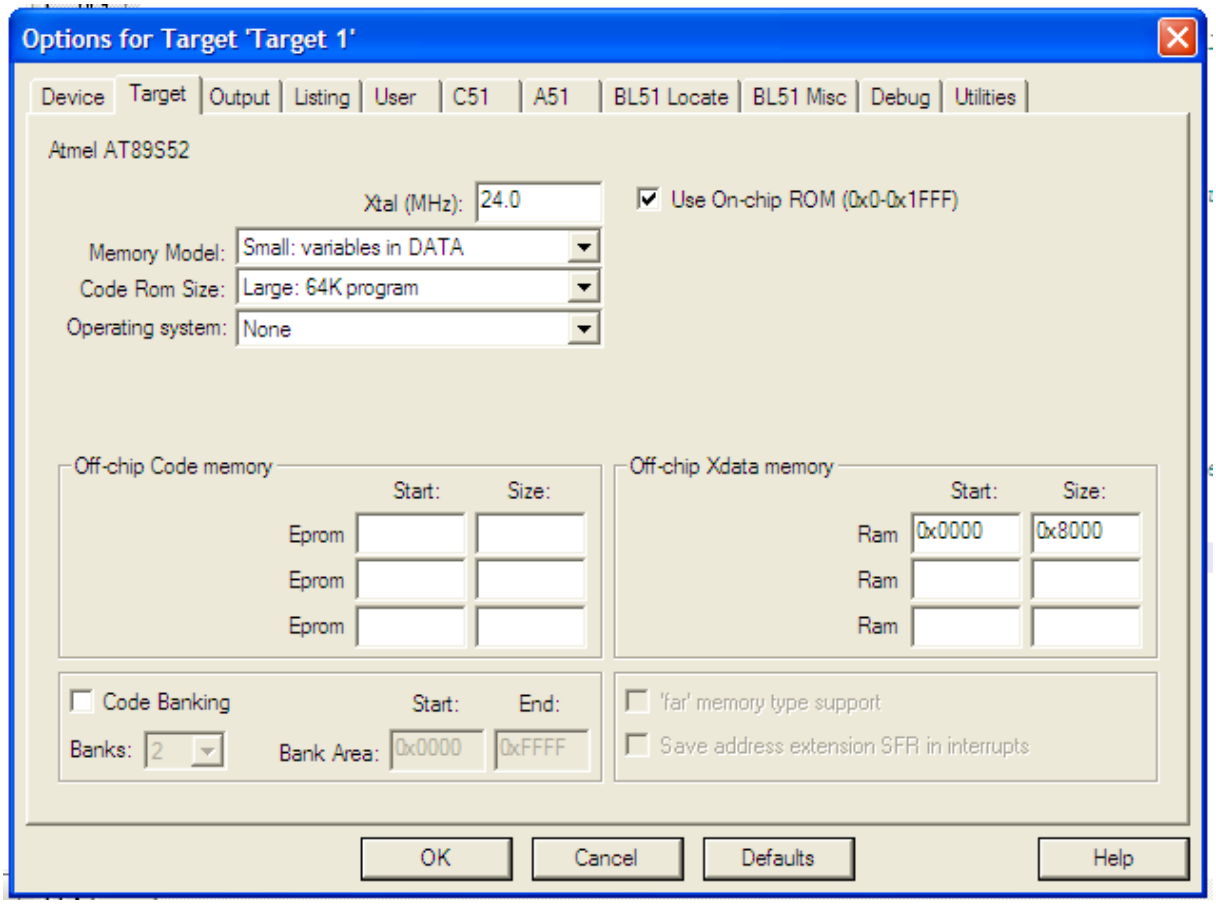
Pro správnou funkci sériové linky je důležité její nastavení. To probíhá v začátku hlavního programu a zmíním se o něm při popisu hlavního programu záznamového zařízení.

4.1.5. Funkce Ulož

Funkce ulož, jak již její název napovídá, zajišťuje uložení záznamu do napěťově nezávislé paměti EEPROM, ale trochu nelogicky i o vyhledávání konkrétních záznamů v již zmíněné paměti. Zdrojový kód funkce ulož je poněkud rozsáhlejší a z toho důvodu a vzhledem k tomu, že je součástí přílohy, ho zde uvádět nebudu. Zde zmíním důležitá nastavení vývojového prostředí Keil, bez kterého by práce s vnější datovou pamětí nebyla možná a logiku funkce Ulož.

Vývojové prostředí Keil pro práci s vnější pamětí potřebuje o této paměti vědět a znát její velikost. Tyto údaje se vývojovému prostředí sdělí na záložce Target, vis obr. 16, kde se v záložce Target kromě frekvence krystalu daného mikropočítače vyplní oblast „Off-chip Data memory“ způsobem jaký ukazuje obr. 38. V tomto případě start na nulté adrese a velikost paměti zde pro 32 kB paměti v hexadecimálním tvaru hodnota 8000.

Když zde uvádím obrázek záložky Target, v rychlosti se zmíním o záložce Device, kde se provádí výběr konkrétního mikropočítače a o záložce Output, kde pro potřebu naprogramování mikropočítače povolit výstup do souboru HEX.



obrázek 38 Záložka Target k nastavení programu Keil

Princip kódování a tím i ukládání záznamu do vnější paměti jsem popsal již v kapitole 2.5.7, takto připravená data ukládám do proměnné xdata „záznamy“. Oblast xdata představuje právě oblast vnější paměti dat. Základ zdrojového kódu pro ukládání dat do paměti uvádím níže.

```
205)    adr=adresy[0];
206)    adr++;
207)    zaznamy[adr]=kod;
208)     adr++;
```

Z datového pole adresy získám šestnáctibitovou adresu posledního uloženého záznamu. Inktementací této adresy získám první volnou adresu ve vnější paměti, na kterou uloží první bajt záznamu. Znovu inkrementuji adresu, tím získám adresu pro uložení druhého bajtu a to samé zopakuji i pro třetí bajt. Tedy v datovém poli záznamy se neustále opakují tyto 3 B představující jednotlivé záznamy v pořadí 1. kódovaný bajt, startovní číslo a 3. Bajt se zakódovanými trestnými body.

Z tohoto vychází i princip vyhledávání záznamu. Vyhledávání začíná vyhledáním startovního čísla. Adresu posledního uloženého čísla zjistím jako dekrementaci posledního uloženého záznamu a každé další uložené startovní číslo je na adrese prvního startovního čísla – 3. Tímto způsobem prohledávám startovní čísla, dokud nenajdu hledané startovní

číslo. Když mám nalezené startovní číslo, zkontroluji, jestli souhlasí kategorie, jízda a jezdec. Pokud vše souhlasí, mám hledaný záznam a mohu vypsat hledané trestné body.

4.1.6. Funkce Příjem

Je realizována obsluhou přerušení, která je vyvolána přijatým bajtem na sériové lince. Sériová linka je primárně nastavena na meziprocesorovou komunikaci, což znamená, že přerušení je vyvoláno jen tehdy, když uprostřed přijatého stop-bitu je i devátý přijatý bit roven log. 1. Tímto způsobem odlišuji bajty představující přenášená data od bajtů představujících příkazy, které tedy na rozdíl od dat mají v devátém bitu vždy hodnotu log. 1. Hodnota příkazu pro záznamové zařízení se skládá z pořadového čísla záznamového zařízení, které udává desítky daného příkazu, v níže uvedeném zdrojovém kódu je to číslo 7 a čísla operace, jak je vidět z níže uvedeného, jedna představuje nastavení čísel závodních bran, které se provádí přes sběrnici z PC z důvodu, aby si odpovídal záznam v záznamovém zařízení a v PC. Dvojka je požadavek na odeslání neodeslaných záznamů. Trojka požadavek na stažení všech záznamů a čtyřka pro smazání paměti. V kapitole 2.5.7 popisuji, že daná paměť se v záznamovém zařízení smazat nedá a teď tu mluvím a smazání paměti. Paměť smazat nejde, ale mohu jí přepsat. Funkce smaž paměti pouze znuluje adresy posledního zapsaného a odeslaného záznamu, čímž se záznamy začnou ukládat znovu od začátku pamětí.

```
443)    void prijem() interrupt 4
444)    {
445)    unsigned char a;
446)    a=SBUF;
447)    RI=0;
448)    RB8=0;
449)    if (brana!=0)
450)      nastaveni(a);
451)    else
452)      {
453)       if (a==71 )
454)          {
455)           SM2=0;
456)           brana=1;
457)          }
458)      if (a==72 )
459)          odesliZ();
460)      if (a==73 )
461)          stahniP();
462)       if (a==74 )
463)          smažP();
464)       }
465)    }
```

4.2. Hlavní program

V této části popíši klíčová místa hlavního programu, hlavně co se nastavení mikropočítače týká.

4.2.1. Nastavení časovače 1

```
477)    TH1=152;
478)    TMOD=32;
479)    TR1=1;
```

Výše uvedený zdrojový kód slouží k nastavení časovače 1, od kterého je odvozena přenosová rychlost sériové linky. Zadávané hodnoty do registru TMOD vysvětlují tabulky 16 a 17. Časovač zde běží v osmibitovém režimu 2 s reloadem.

7. bit	6. bit	5. bit	4. bit	3. bit	2. bit	1. bit	0. bit
G	C/T	M1	M0	G	C/T	M1	M0
čítač/časovač1				čítač/časovač0			

Tabulka 16 Registr TMOD zdroj [14]

G – hradlování čítač/časovač:
G=0 je řízen pouze bitem TRx v registru TCON (x = pořadí čítače/časovače)
G=1 je řízen bitem TRx a vstupem INTx.
C/T – volba zdroje hodinového signálu:
C/T=0 hodiny z vnitřního zdroje jde o časovač.
. C/T=1 hodiny z vnějšího zdroje jde o čítač.
M1 a M2 – režim čítač/časovač:

Mód	**M1**	**M0**	**Režim**
0	0	0	13 bitový režim
1	0	1	16 bitový režim
2	1	0	8 bitový režim s funkcí RELOAD
3	1	1	pracuje-li čítač/časovač0 v režimu 3 lze čítač/časovač1 pouze v aplikaci, která nepracuje s přerušením (například generátor přenosové rychlosti pro sériový kanál)

Tabulka 17 Režim čítače/časovače zdroj [14]

Časový interval odměřovaný časovačem 1 je dán hodnotou 152 v registru TH1. Tato hodnota se vypočítá podle požadované přenosové rychlosti dle vzorce 1. Požadovaná přenosová rychlost je v tomto případě 1200 b/s.

$$PR = \frac{2^{SMOD}}{32} * \frac{f}{12 * (256 - TH1)} \longrightarrow TH1 = 256 - \frac{2^{SMOD}}{32} * \frac{f}{12 * PR} \qquad (1)$$

PR – přenosová rychlost
f – hodinový kmitočet mikropočítače
TH1 – obsah registru TH1 časovače 1
SMOD – hodnota bitu SMOD, vis tab. 20

4.2.2. Nastavení sériového kanálu (UART)

477) SCON=240;
478) PCON=128;

Registry SCON a PCON slouží k nastavení vlastností sériové linky. V tomto případě je sériová linka nastavena na režim 3, tedy asynchronní devítibytový přenos v režimu meziprocesorové komunikace.

7. bit	6. bit	5. bit	4. bit	3. bit	2. bit	1. bit	0. bit
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Tabulka 18 Registr SCON zdroj [14]

SM1, SM0 - Kombinací těchto bitů se volí jeden ze čtyř módů sériového kanálu.

SM1	**SM0**	**Mód**	**Typ přenosu**	**Přenosová rychlost (b/s)**
0	0	0	Synchronní 8bitový	fosc/12
0	1	1	8-bitový UART	časovač 1
1	0	2	9-bitový UART	fosc/32, fosc/64
1	1	3	9-bitový UART	časovač 1

Tabulka 19 Mód sériového kanálu zdroj [14]

SM2 – povolení takzvané víceprocesorové komunikace
REN – povolení příjmu
TB8 – vysílaný 9. bit (v režimech 2 a 3)
RB8 – přijatý 9. bit (v režimech 2 a 3)
TI – indikace vyprázdnění vysílacího registru

V režimu 0 je aktivován po vyslání 8. bitu
V režimech 1, 2 a 3 je aktivován na začátku stop-bitu. Nastaví se hardwarově, nuluje se programově

RI – indikace naplnění přijímacího registru
V režimu 0 je aktivován po přijetí 8. bitu
V režimech 1, 2 a 3 je aktivován uprostřed stop-bitu. Nastaví se hardwarově, nuluje se programově

fosc – frekvence oscilátoru mikropočítače

V nastavení PCON nastavuji bit ovlivňující přenosovou rychlost sériové linky, vis tab. 20.

7. bit	6. bit	5. bit	4. bit	3. bit	2. bit	1. bit	0. bit
SMOD	-	-	POF	-	-	PD	IDL

Tabulka 20 Registr PCON zdroj [14]

4.2.3. Nastavení přerušovacího systému mikropočítače

```
477)    ES=1;
478)    EA=1;
```

Nastavení povolení přerušení od sériové linky. Pro povolení přerušení se musí povolit přerušení od konkrétního zdroje ES (od sériového kanálu) a zároveň i povolení systému přerušení EA tedy povolení všech přerušení v registru IE.

4.2.4. Činnost hlavního programu

Samotný program běží v nekonečném cyklu, kde neustále testuje klávesnici, jestli nedošlo ke stisku klávesy. Pokud ano, vyhodnotí požadavek plynoucí systému ze stisknuté klávesy a zavolá funkci pro vyřízení požadavku, vis příloha 1.

5. Závěr

Cílem této diplomové práce bylo navrhnout záznamové zařízení pro rozhodčí vodního slalomu s nízkou ekonomickou náročností na pořízení i provoz. Toto zařízení by mělo urychlit zpracovávání výsledků závodu jak pro potřeby pořadatelů závodu, ale i pro diváky a to díky své příznivé ceně i na závodech nižších tříd, což by mohlo být dalším krokem k větší popularitě vodního slalomu.

Při vidině výše popsaných cílů jsem se rozhodl brankovým rozhodčím sebrat papír a tužku a na místo toho je vybavit klávesnicí a displejem, čímž budou data o průjezdu jednotlivých závodníku pořízena rovnou v digitální podobě a odpadne zdlouhavé a pracné přepisování pořízených záznamů v papírové podobě do počítače, tím odpadne spousta práce pro počtáře závodu, kteří by tím pádem měli být pod nižším časovým tlakem a mít víc času ke kontrole výsledků. Záznamové zařízení by bylo přínosem i pro brankové rozhodčí, kteří by ke svému bodování měli přístup po celou dobu závodu. Okamžitá existence záznamů v digitální podobě otevírá i další možnosti pro využití těchto dat, například zobrazením na výsledkových tabulích.

Záznamové zařízení realizuji jako vestavný systém s mikropočítačem AT89S52 z řady 8052, která vychází z řady 8051 a je s ní zcela kompatibilní. Obecným popisem mikropočítačů řady 8051 se zabývám v teoretické části této práce. Pro zachování kompaktních rozměrů jsem použil šestnácti tlačítkovou maticovou klávesnici a dvouřádkový LCD alfanumerický displej s radičem, řádky jsou po šestnácti znacích.

Vzhledem k tomu, že plánovaný hlavní uživatel tohoto systému upřednostňuje pevné propojení systému před bezdrátovým, použil jsem k zajištění přenosu dat sběrnici RS-485, jejíž parametry dostačují pro sběr dat podél závodní tratě ve Veltrusech, pro kterou je zařízení primárně navrhováno. Rozloha umělé slalomové dráhy ve Veltrusech je srovnatelná s velkou částí areálů pro vodní slalom, které znám a záznamové zařízení by na nich tedy mohlo být bez potíží rovněž nasazeno.

Tato práce je zároveň i výrobními podklady pro tvorbu tohoto záznamového zařízení. Obsahuje jak samotná schémata se zapojením tohoto zařízení tak i návrh plošného spoje pro záznamové zařízení a ve své závěrečné části obsahuje i podrobný popis programového vybavení záznamového zařízení. Vzhledem k výše uvedenému, by tato práce mohla posloužit i jako výchozí krok pro návrh zařízení podobného typu.

Záznamové zařízení má v současné době za sebou úspěšné testy v laboratorních podmínkách, na základě kterých věřím, že uvedení zařízení do provozu se obejde bez vážných problémů.

I přes to, že při příchodu na CZU mě představa zvládnutí programování mikropočítačů značně děsila, přiznám se, že během studia jsem toto odvětví shledal zajímavým a tvorbu záznamového zařízení i zábavnou činností.

6. Seznam literatury

[1] 256K (32Kx8) Paged CMOS E2PROM [online]. [cit. 2014. 3. 6.]. Dostupné z: http://www.gme.cz/img/cache/doc/414/006/28c256-150-datasheet-1.pdf

[2] 3 POS.PLUG ASS'Y: AMP SUPERSEAL 1,5 SERIES CONN [online]. [cit. 2014 3. 6.] Dostupné z: http://www.gme.cz/img/cache/doc/899/152/superseal-konektor-te-connectivity-282087-1-datasheet-1.pdf

[3] 74HC/HCT573 Octal D-type transparent latch; 3-state[online]. [cit. 2014. 3. 6.]. Dostupné z: http://www.gme.cz/img/cache/doc/426/081/74hct573-datasheet-1.pdf

[4] AMP SUPERSEAL 1,5 SERIES CONNECTORS: Product specification 108-20090 [online]. [cit. 2014.3. 6.]. Dostupné z: http://www.gme.cz/img/cache/doc/899/152/superseal-konektor-te-connectivity-282087-1-datasheet-3.pdf

[5] AMP SUPERSEAL 1,5 SERIES: CUSTONER MANUAL Nr. 412-20000 [online]. [cit. 2014. 3. 6.]. Dostupné z: http://www.gme.cz/img/cache/doc/899/152/superseal-konektor-te-connectivity-282087-1-datasheet-4.pdf

[6] ATMEL 64K (8K x 8) Parallel EEPROM with Page Write and Software Data Protection AT28C64B [online]. [cit. 2014. 3. 6.]. Dostupné z: http://www.gme.cz/img/cache/doc/414/007/at28c64b-15pu-datasheet-1.pdf

[7] ATMEL 8-bit Mikrokontroller with 8K BytesIn-Systém Programmable Flash AT89S52 [online]. [cit.2014. 3. 6]. Dostupné z: http://www.gme.cz/img/cache/doc/432/198/at89s52-24pu-datasheet-1.pdf

[8] Burkhard, Kainka. Elektronika s podporou PC: Visual Basic v praxi. 1. vydání. 1. vydání. Ostrava: HEL. 2004. 184 s. ISBN 80-86167-22-4

[9] Burkhard, Mann. C pro mikrokontroléry. 1. vydání. Praha: BEN - technická literatura, 2003. 279 s. ISBN 80-7300-077-6

[10] GPT-4454 [online]. [cit.2014 3. 6.] Dostupné z: http://www.megatron.cz/download/operacni-manualy/gpt-4454.pdf

[11] Hrabovský, Miroslav. EAGLE pro začátečníky: návrhový systém plošných spojů: uživatelská a referenční příručka. 1. vydání. Praha: BEN - technická literatura, 2005. 191 s. ISBN 80-7300-213-2

[12] J-Y(ST)Y 1X2X0.8 ROT[online]. [cit. 2014. 3. 6.]. Dostupné z: http://www.argos.cz/produkty/j-y-st-y-1x2x0-8-rot

[13] Katalog elektronických součástek GM Electronic 2008 [CD-ROM]. [cit. 2014. 3. 9.].

[14] Matoušek, David. Brtnik, Bohumil. Programování mikrokontrolérů s jádrem 8051 v jazyce C. 1. vydání. Praha: BEN - technická literatura, 2010. 151 s. ISBN 978-80-7300-264-0

[15] Matoušek, David. C pro mikrokontroléry Atmel AT89S52. 1. vydání. Praha: BEN technická literatura, 2007. 240 s. ISBN 80-7300-215-9

[16] Matoušek, David. Práce s inteligentními displeji LCD: 1. díl. 1. vydání. Praha: BEN technická literatura, 2006. 224 s. ISBN 80-7300-121-7

[17] Matoušek, David. Práce s mikrokontroléry Atmel AT89S8252: 2. díl. 1. vydání. Praha: BEN - technicka literatura, 2002. 304 s. ISBN 80-7300-066-0

[18] Matoušek, David. Vývojový kit USB51KIT: Podrobný stavební návod s ovládacím programem pro Windows. 1. vydání. Praha: BEN - technická literatura, 2005. 23 s. ISBN 80-7300-162-4

[19] MAXIM Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers [online]. [cit. 2014. 3. 6]. Dostupné z: http://www.gme.cz/img/cache/doc/433/118/max485cpa-datasheet-1.pdf

[20] MTH-2500 & 3500: THERMAL PRINTERS WITH ELM-208 HEAD [online]. Vystaveno 29. 7. 2013 [cit.2014 3. 6.]. Dostupné z: http://www.megatron.cz/ke-stazeni/?dirpath=WPd1KT2OOHVdUH4wfM8CED97zgxJiRyx8rfq5u4C30W02eQfz6cCNsyevOTXl1BwX27iJJHIC%2F6ZbAVHyT0rvbM3NG%2Bmmv7SZgBSzz7wyGNqlgiM%2FqL9%2FBcQPBZYqxR4D2HrdWj1M%2B6PTP%2FP5FlwmyutWzrAsyvEEBfIMRPNkC0%3D&root=2
[21] Plíva, Zdeněk. EAGLE prakticky: řešení problémů při běžné práci. 1. vydání. Praha: BEN - technická literatura, 2007. 181 s. ISBN 80-7300-227-2
[22] Pravidla kanoistiky na divokých vodách [online]. [cit. 2014 3. 6.]. Dostupné z: http://www.kanoe.cz/files/CSKDV/rozhodci/2013%20CSKDV%20rozhodci/Pravidla.doc
[23] Skalický, Petr. Mikroprocesory řady 8051. 2. vydání. Praha: BEN - technická literatura, 1998. 160 s. ISBN 80-86056-39-2
[24] Stanek, Jan. RS 485 & RS 422 [online]. Vystaveno 23. 5. 2003 [cit. 2014. 3. 6.]. Dostupné z: http://hw-server.com/rs-485-rs-422
[25] Šalomoun, Petr. Programovací jazyk C++ pro zelenáče. 1. vydání. Praha: Neocortetex s.r.o., 2005. 252 s. ISBN 80-86330-18-4
[26] TC485: Převodník sběrnic RS232 na RS485 Jednoduché provedení pro základnípoužití [online]. [cit. 2014. 3. 6.]. Dostupné z: http://www.papouch.com/cz/shop/product/tc485-prevodnik-rs232-na-rs485/tc485.pdf/_downloadFile.php
[27] Tůma, Jan. Náš život s počítači. 1. vydání. Praha: Naše vojsko, 1990. 240 s. ISBN 80-206-0089-2
[28] Tvarové díly[online]. [cit. 2014. 3. 7.]. Dostupné z: http://www.rayservice-ade.cz/file/64/
[29] Vacek, Václav. Učebnice programovaní Atmel s jádrem 8051. 1. vydáni. Praha: BEN - technická literatura. 2001. 144 s. ISBN 80-7300-043-1
[30] Váňa, Vladimír. Mikrokontrolery Atmel AWR. 1. výdání. Praha: BEN - technická literatura. 2003. 336 s. ISBN 80-7300-083-0
[31] Veltrusy [online]. [cit. 2014. 3. 6.]. Dostupné z: http://www.mapy.cz/
[32] WH2002A Charakter 20x2 [online]. [cit. 2014. 3. 6.]. Dostupné z: http://www.gme.cz/img/cache/doc/513/224/lcd-alfanumericky-displej-winstar-wh2002a-nyg-et-datasheet-1.pdf
[33] Winstar WH1602A Charakter 16x2 [online]. [cit. 2014. 3. 6.]. Dostupné z: http://www.gme.cz/img/cache/doc/513/219/lcd-alfanumericky-displej-winstar-wh1602a-ygh-ct-datasheet-1.pdf
[34] Zajímavé IO Maxim pro RS485 [online]. Vystaveno 9. 1. 2005 [cit. 2014. 3. 6.]. Dostupné z: http://www.hw.cz/soucastky/zajimave-io-maxim-pro-rs-485.html

Seznam obrázků

Seznam tabulek

Přílohy:

Příloha 1:

Programové vybavení záznamového zařízení.

Příloha 2:

Uložení jednotlivých alfanumerických znaků v paměti LCD displeje.

Příloha 3:

Obsah CD.

Příloha 1:

Programové vybavení záznamového zařízení pro rozhodčí vodního slalomu

```
1) #include <regx52.h>
```

Tato direktiva připojí k právě zpracovávanému zdrojovému kódu obsah souboru, na který direktiva odkazuje. Soubor regx52.h obsahuje definice jednotlivých registrů mikropočítače řady 8052.

```
2) #define E  P3_5
3) #define RS  P3_4
4) #define DE  P3_2
```

Direktiva define přiřadí symbolické jméno prvku na který direktiva odkazuje. V tomto případě je to pojmenování výstupů portu P3 pinu 2, 4 a 5 jmény signálů pro ovládaný hardware.

```
5) static unsigned char kurzor = 0;
6) static signed char predKod = -1;
7) static unsigned char brana=0;
8) static idata unsigned char obsahD[21];
9) static xdata unsigned int adresy[2];
10) static xdata unsigned char zaznamy[32000];
11) static xdata unsigned char cislaB[6];
```

Globální proměnné, jestliže se proměnná deklaruje mimo těla funkce je viditelná všem častem programu. Nejlépe je deklarovat globální proměnné hned na začátku.

V proměnné kurzor je uložena aktuální pozice kurzoru na displeji.

Jestli je před proměnou příkaz idata, je tato proměnná ukládána v uživatelské časti paměti RAM od adresy 128 a já do této části paměti ukládám datové pole o délce 21B, v kterém je uchováván obsah vypsaný na displeji.

Obdobným způsobem příkaz xdata udává, že dané proměnné se uloží do vnější paměti dat, kterou v tomto případě je napěťově nezávislá paměť EEPROM. Do této paměti jsou ukládána datová pole s uloženými záznamy o závodě (záznamy), adresy posledního uloženého záznamu a posledního odeslaného záznamu (adresy) a čísla závodních bran daného postu „cislaB“.

O zbylých proměnných se zmíním, až se k jejich využití dostanu v samotném kódu.

Funkce

Zde postupně popíši jednotlivé funkce, s kterými hlavní program pracuje.

Čekej

```
10) void cekej (unsigned char T)
11) {
12) unsigned char i;
13) for(;T>0;T--)
14)    {
15)     for (i=1;i<=23;i++);
16)    }
17) return;
18) }
```

Čekej2

```
19) void cekej2 (unsigned char M)
20) {
21) unsigned char i;
22) M=M*4;
23) for (i=1;i<=M;i++)
24) cekej(250);
25) }
```

Jak název napovídá, tyto funkce se volají, když se nemá nic dělat, slouží pouze ke zpracování strojového času tím, že vykonávají daný počet prázdných instrukcí v cyklu. Funkce čekej2 prodlužuje čekací dobu tím, že opakovaně volá funkci čekej. Tyto funkce měly původně představovat přesný časový úsek v μs a ms, ale při multiplexovámí adresy pro paměť, dat a příkazů pro displej na portu P2 původní záměr nevyhovoval. Při ladění těchto operací jsem experimentálním způsobem nastavil časové prodlevy.

Klávesnice

```
49) signed char klavesnice()
50) {
51) unsigned char a,b,radek,sloupec;
52) signed char kod=0;
53) code unsigned char radky[]={239,223,191,127};
54) code unsigned char sloupce[]={0,1,2,0,3,0,0,0,4,0,0,0,0,0,0,0};
55) for(radek=0;radek<=3 && kod==0;radek++)
56) {
57)   P1=radky[radek];
58)   do{
59)      a=(~P1)&15;
60)      cekej(77);
61)       b=(~P1)&15;
62)   }while(a!=b);
63)   sloupec=sloupce[a];
64)   if(sloupec!=0)
65)      kod=4*radek+sloupec;
66) }
67) if(kod==predKod)
68)     kod=0;
69) else
70)     predKod=kod;
71) return kod-1;
72) }
```

V první části funkce jsou deklarovány 1B proměnné a následně dvě datová pole, v datovém poli řádky jsou uloženy konstanty pro výběr řádku a v datovém poli sloupce jsou konstanty pro dekódování sloupců.
Od řádku 32. se spustí cyklus, který prochází řádky, dokud nenarazí na stisklou klávesu, nebo neprojde všechny řádky. Mezi řádky 35. až 39. dochází ke čtení stavu řádku, tak že se přečte stav celého portu, který se hned zneguje a následně se vymaskují horní 4 bity tak, že zůstanou

jen spodní 4 bity. Toto se po časové prodlevě opakuje a cyklus je ukončen v případě, že jsou si výsledky obou čtení rovny, toto je ochrana proti zákmitům tlačítek klávesnice. Na řádku 40 dojde k dekódování sloupce, tedy hodnoty 0, 1, 2, 4, 8 se dle tabulky sloupce převedou na hodnoty 0, 1, 2, 3, 4. Pokud je stisklá klávesa tak se na řádku 42. z kombinace sloupce a řádku vypočítá pozice a tedy kód stisklé klávesy. Tato část kódu je podrobněji popsána v [14]. S tímto kódem jsem měl problém, že při stisku klávesy mnohdy vypsal více než jeden znak, proto jsem na konec funkce přidal podmínku, která hlídá, jestli nedochází ke stisku stejné klávesy. Z tohoto důvodu potřebuji globální proměnnou predKod jejíž obsah zůstane zachován i mimo běh této funkce a je v ní uložen kód předchozí stisklé klávesy. Pro výpis dvou stejných znaků je zapotřebí uvolnění klávesy.
Funkce klávesnice vrací proměnnou „kod“, proto musí končit příkazem RETURN. Funkce vrací hodnotu 0 až 15 jako kód stisklé klávesy a hodnotu -1 pokud není stisklá žádná klávesa.

Povel (řízení displeje)

```
73) void povel (unsigned char p)
74) {
75) RS=0;
76) P2=p;
77) E=1;
78) cekej(1);
79) E=0;
80) }
```

Způsob komunikace s displejem je znázorněn v tab. 14 a v tab. 15 jsou uvedeny významy proměnných z tab. 14. Znak * značí, že na daném bitu v daný okamžik nezáleží. Čas udává minimální potřebný čas pro zpracování odeslané operace displejem. Aby displej věděl, že jde o povel a ne data, musí být signál RS v log. 0 a přítomná hodnota požadovaného příkazu na portu, v tomto případě P2, tato hodnota se funkci předává jako parametr p. Po sestupné hraně signálu E displej přečte stav svých vstupů a provede požadovanou operaci. Vzhledem k tomu, že ne všechny příkazy pro displej mají stejnou dobu zpracování displejem, není součástí funkce povel závěrečná časová prodleva a musí za touto funkcí vždy následovat funkce čekej, nebo čekej2.

Vypiš znak

```
81) void vypisZnak (unsigned char kod)
82) {
83) unsigned char a;
84) a=kurzor;
85) RS=1;
86) P2 = kod;
87) E=1;
88) cekej(1);
89) E=0;
90) kurzor=kurzor++;
91) if (kod<48)
92)    kod=48;
93) if (a<5)
94)    obsahD[a]=kod-48;
```

```
95) if (kurzor>63)
96)    obsahD[a-59]=kod-48;
97) cekej(40);
98) return;
99) }
```

Do řádku 66 je funkce principiálně stejná s funkcí povel a také vychází z tab. 14 a 15. Po odeslání znaku, který se funkci předává jako parametr „kod“, dojde k inkrementaci proměnný kurzor, což představuje posun o jeden znak dopředu. Podmínka na řádku 91 zabrání přetečení do záporných čísel. Do datového pole „obsahD“ ukládám pouze data zobrazená na displeji, která budu později potřebovat, tím dochází k podstatnému snížení paměťových nároků pro uložení daného pole. O to se starají podmínky na řádcích 93 a 95. Pro vypsání znaku na displej je vždy stejná doba zpracování a tak se funkce čekaní volá hned z těla funkce vypiš znak a není ji díky tomu potřeba za touto funkcí uvádět samostatně.

Inicializace displeje

```
100)    void inicializace()
101)    {
102)    cekej2(30);
103)    povel(56);
104)    cekej(40);
105)    povel(15);
106)    cekej(40);
107)    povel(1);
108)    cekej2(2);
109)    povel(6);
110)    cekej(40);
111)    return;
112)    }
```

Pro správnou funkci displeje musím displeji nejdříve dát vědět, v kterém režimu si přeji, aby pracoval. Tomuto prvotnímu nastavení se říká inicializace a skládá se z dané sekvence příkazu. Nejdříve musím počkat na ukončení vnitřního restartu displeje po přivedení napájení. Výchozím nastavením displeje po restartu je osmibitová komunikace a tak tento požadavek displeji sdělovat nemusím. Řádek 103 šířka komunikace, počet řádků a velikost fontu. Řádek 105 zapnutí displeje a nastavení kurzoru. Řádek 107 smaže displej a nastaví kurzor na 1. pozici. Řádek 109 posuv kurzoru a displeje. Přesný význam parametru povelu vis tab. 14 a tab. 15.

Smaž znak

```
113)    void smazZnak()
114)    {
115)    vypisZnak(32);
116)    kurzor--;
117)    povel(128+kurzor);
118)    cekej(40);
119)    }
```

Chování funkce již zcela popisuje její název. Pracuje tak, že na místo mazaného znaku zapíše prázdný znak. Vzhledem k tomu, že znak mažu, abych jej mohl opravit, následně se funkce vrátí zpět na přepisované pole, tím že dekrementuji hodnotu kurzoru a tu použiji jako adresu místa nastavení kurzoru, vis tab. 14.

Posun do leva

```
120)    void posunLeva()
121)    {
122)
123)
124)      do
125)        {
126)        povel(127+kurzor);
127)        cekej(40);
128)        kurzor--;
129)        }
130)        while (kurzor==71 || kurzor==74 || kurzor==77);
131)
132)    }
```

Funkce svým chováním opět odpovídá svému jménu a posune kurzor o jedno pole do leva. Dekrementace hodnoty kurzoru je zajištěna snížením konstanty 128 na 127, vis tab. 14 a 15. Pokud se kurzor zastaví na pozici, která se ve vytvářené tabulce na displeji nevypisuje, cyklus DO WHILE zopakuje posun ještě jedenkrát.

Posun do prava

```
133)            void posunPrava()
134)            {
135)            povel(129+kurzor);
136)            cekej(40);
137)    kurzor=kurzor++;
138)    }
```

Stejný princip jako u posuvu doleva, ale zde inkrementuji. Přeskok volných míst řešen jinak, popíši později.

Posun o řádek níž

```
139)    void niz()
140)    {
141)    if(kurzor<16)
142)      {
143)      povel(192+kurzor);
144)      cekej(40);
145)      kurzor=kurzor+64;
146)      }
147)    }
```

Posun řešen stejným způsobem jako u posuvu kurzoru vpravo a vlevo. Po přesunu na druhý řádek je potřeba zvýšit hodnotu kurzoru o 64 vzhledem k tomu, že první pozice 2. řádku má adresu 64.

Posuv o řádek výš

```
148)  void vys()
149)  {
150)  if(kurzor>63)
151)    {
152)    povel(64+kurzor);
153)    cekej(40);
154)    kurzor=kurzor-64;
155)    }
156)  }
```

Obrácený princip posuvu o řádek níž.

Pozdrav

```
157)  void pozdrav()
158)  {
159)  code unsigned char
      poz[]={78,69,80,82,79,80,65,68,69,74,84,69,80,65,78,73,67,69,33};
160)  unsigned char i;
161)  for (i=0;i<19;i++)
162)    {
163)      if (i==12)
164)      {
165)      povel(201);
166)      cekej(40);
167)      }
168)    vypisZnak(poz[i]);
169)    }
170)  }
```

Jde jen o vypsání pozdravu při spuštění zařízení, podobně jako to dělají například mobilní telefony. V proměnné „poz“ je v ASCII kódu uložen nápis NEPROPADEJTE PANICE, který následně cyklus na řádku 161 vypíše na displej. Vzhledem k tomu, že je pozdrav delší než řádek displeje, po vypsání prvního slova pozdravu se druhé slovo začne vypisovat od poloviny druhého řádku, o což se stará podmínka na řádku 163. Pozdrav je z mé oblíbené knihy Stopařův průvodce po galaxii a zvolil jsem ho z důvodu, že v knize se používá úplně ke stejnému účelu, jako pozdrav při spuštění důmyslného zařízení elektronické knihy Stopařův průvodce po galaxii. Druhým důvodem pravděpodobně bude, že jsem si jí během tvorby tohoto programu potřeboval připomínat.

Kategorie

```
148)  void kategorie (unsigned char z)
149)  {
```

```
150)    code unsigned char
   kat[][2]={{79,83},{75,77},{75,90},{67,77},{67,90},{67,50},{67,87},{72,108}};
151)    unsigned char i;
152)    unsigned char v;
153)    if(kurzor<3)
154)      {
155)      z=z-48;
156)      if(z<=7)
157)        {
158)        povel(128);
159)        kurzor=0;
160)        cekej(40);
161)        for(i=0;i<2;i++)
162)          {
163)          v=kat[z][i];
164)          vypisZnak(v);
165)          cekej(40);}
166)           }
167)       obsahD[0]=z;
168)       povel(131);
169)       kurzor=3;
170)       cekej(40);
171)       }
172)     }
173)    }
```

Vzhledem k tomu, že na ovládací klávesnici jsou pouze číslice, má tato funkce za úkol číselný kód kategorie, vis tab. 6, převést na zkratku kategorie. Zkratka obsahuje dva alfanumerické znaky, které jsou uloženy v dvourozměrném poli konstant „kat“ deklarovaném na řádku 150. Funkce na řádku 153 nejdříve otestuje, zda je znak zapisován do části displeje určenému pro kategorii závodu, pokud ano, na řádku 155 převede číselný znak na hodnotu odpovídající zapisovanému číslu, pokud ne, celá funkce končí. Na řádku 156 podmínka testuje, jestli zadaná hodnota nepřesahuje rozsah kategorií. Pokud je vše v pořádku, na řádku 158 nastavím kurzor na první pozici displeje a cyklem na řádku 161 vypíši dva znaky proměnné odpovídající pořadí kategorie a proměnné „i“ postupně vypíše první a druhý znak. Po vypsání kategorie se kód kategorie na řádku 166 uloží do datového pole „obsahD“, kde zůstane pro pozdější použití. Před ukončením funkce dojde k nastavení kurzoru na pozici 3, tedy na čtvrté pole displeje, kde se předpokládá následné zadání pořadí jízdy.

Odešli

```
174)    void odesli (unsigned char p)
175)    {
176)    SBUF = p;
177)    while(TI!=1);
178)    TI=0;
179)    cekej(10);
180)    }
```

Funkce „odešli“ slouží k odeslání jednoho bajtu přes sériové rozhraní UART. Odeslání se provádí vložením jednobajtové proměnné do registru SBUF na řádku 175. Pro správné odeslání bajtu je zapotřebí počkat na odeslání bajtu, což zajišťuje cyklus WHILE na řádku 176 tím, že neustále testuje, jestli nedojde k nastavení bitu TI v registru SCON, který indikuje vyprázdnění zásobníku. Tento bit se nastavuje hardwarově, ale nulovat se musí programově, zde na řádku177. Funkce končí krátkou prodlevou k ukončení přechodových dějů na lince.

Ulož

Funkce ulož se v podstatě skládá ze dvou, ulož a vyhledej. Pro přehlednost vysvětlím princip obou funkcí odděleně, i když v pragramu jsou dohromady.

```
181)    void uloz()
182)    {
183)    unsigned char i;
184)    unsigned char kod;
185)    unsigned char a;
186)    unsigned char b;
187)    unsigned int adr;
188)    if (kurzor!=67)
        Ulož záznam
        else
        Vyhledej záznam
```

Proměnná „i“ slouží k řízení cyklů for. Proměnná „kod“ slouží k zakódování ukládaných dat do jednoho bajtu. Proměnné „a, b“ jsou pomocné a slouží k uložení mezivýsledků při kódování. Šestnáctibitová proměnná „adr“ slouží pro uchování adresy ve vnější paměti dat ukládaného záznamu. Na řádku 187 testuji zda se nenacházím na pozici displeje, která by znamenala, že neukládám, ale chci hledat.

Ulož záznam

```
189)    kod=obsahD[0];
190)    if (kod==7)
191)      {
192)      a=obsahD[9];
193)      if (a>3)
194)        a=0;
195)      a=a<<5;
196)      kod=kod|a;
197)      }
198)    a=obsahD[3];
199)    if (a<3 && a>0)
200)      a=a-1;
201)    else
202)      a=0;
203)    a=a<<7;
204)    kod=kod|a;
205)    adr=adresy[0];
206)    adr++;
```

```
207)      zaznamy[adr]=kod;
208)      adr++;
209)      kod=obsahD[5];
210)      if (kod<3)
211)        kod=kod*100;
212)      a=obsahD[6];
213)      if (a<10)
214)        kod=kod+(10*a);
215)      a=obsahD[7];
216)      if (a<10)
217)        kod=kod+a;
218)      zaznamy[adr]=kod;
219)      adr++;
220)      povel(192);
221)      kurzor=64;
222)      cekej(40);
223)      if(obsahD[0]==7)
224)        {
225)         if (obsahD[9]>=3)
226)           {
227)           obsahD[9]=1;
228)           if (kod<255)
229)           kod++;
230)           }
231)           else
232)           kod=1;  else
233)         obsahD[9]++;
234)         }
235)         else
236)         {
237)         obsahD[9]=0;
238)         if (kod<254)
239)           kod++;
240)         else
241)           kod=1;
242)         }
243)      vypisZnak((kod/100)+48);
244)      kod=kod%100;
245)      vypisZnak((kod/10)+48);
246)      vypisZnak((kod%10)+48);
247)      vypisZnak(32);
248)      if (obsahD[9]!=0)
249)        vypisZnak((obsahD[9]+48));
250)      else
251)        vypisZnak(32);
252)      kod=0;
253)      for(i=0;i<4;i++)
254)        {
255)        a=obsahD[10+(3*i)];
```

```
256)          if (a>9)
257)            a=0;
258)          a=a*10;
259)          b=obsahD[11+(3*i)];
260)          if (b>9)
261)            b=0;
262)          a=a+b;
263)          if (a==1)
264)            a=0;
265)          if (a>2)
266)             {
267)        if (a>5 && a<50)
268)                a=2;
269)             else
270)                a=3;
271)             }
272)          a=a<<(2*i);
273)          kod=kod|a;
274)          }
275)        zaznamy[adr]=kod;
276)        adresy[0]=adr;
277)        cekej(100);
278)        povel(197);
279)        kurzor=69;
280)        cekej(40);
281)        for (i=0;i<11;i++)
282)          {
283)          vypisZnak(32);
284)          obsahD[10+i]=0;
285)          }
286)        povel(197);
287)        kurzor=69;
288)        cekej(40);
289)        }
```

Na řádku 188 načtu z „obsahD“ kód kategorie, následně otestuji, jestli nejde o kód hlídek, pak by bylo potřeba do kódu uložit i číslo jezdce a na řádku 194 se posune o 5 bitů výše, následně na řádku 195 logicky přičte ke kódu „kod“, tím dvoubitová hodnota pořadí jezdce v hlídce v kódu obsadí 6 a 5 bit. Na 197 řádku načte z „obsahuD“ pořadí jízdy, po otestování rozsahu proměnné se převede na rozsah 0 a 1 a tento jeden bit se následně na řádku 203 uloží na místo 7. bitu ve slově „kod“. Tím je dokončeno zakódování prvního ukládaného bajtu a z datového pole adresy je do proměnné „adr“ načtena adresa posledního uloženého záznamu. Datové pole adresy je uloženo v externí paměti dat EEPROM z důvodu, že i po restartu systému zde zůstane zachována adresa posledního platného uloženého záznamu a tím nehrozí, že by po restartu došlo k přepisování již uložených záznamů. Proměnná „adr“ se inkrementuje, čímž se získá adresa prvního volného místa v paměti. Následně se „kod“ uloží do datového pole záznamy na řádku 207, čímž dojde k uložení prvního bajtu záznamu do paměti EEPROM a proměnná „adr! se opět inkrementuje, tím je vygenerována adresa pro uložení dalšího bajtu záznamu.

V datovém poli „obsahD“ na šesté pozici, vzhledem k tomu, že datová pole jsou číslována od 0, tak je adresa této konstanty 5, je uložena první číslice startovního čísla, tedy stovky. Tato proměnná je na řádku 210 kontrolována podmínkou if, jestli není zadané číslo mino rozsah jednoho bajtu. Je-li číslo maximálně do hodnoty 2, vynásobí se toto číslo konstantou 100. To samé se provede s proměnnými „obsahD [6] a [7]“, kde jsou uloženy desítky a jednotky startovního čísla, ty se uloží do pomocné proměnné „a“, zde se již kontrolují čísla na celý rozsah 0 až 9 a následně se přičtou k proměnné „kod“, čímž se převede znakový obsah displeje na číselnou hodnotu odpovídající zapsanému číslu, která se následně uloží do záznamů na odpovídající místo paměti. Po uložení se proměnná „adr“ opět navýší o 1 pro použití s posledním ukládaným bajtem. Následně na řádku 220 dojde k přesunutí kurzoru na první pozicí druhého řádku a na řádku 222 ke kontrole, jestli nejde o závod hlídek, pokud ano, zkontroluje se zda nejede poslední závodník hlídky, pokud ne, navýší se pořadí jezdce v hlídce o 1, jde-li o posledního jezdce hlídky, nastaví se pořadí jezdce zpět na 1. jezdce a k navýšení startovního čísla na následující startovní číslo. V případě kdy by projelo startovní číslo 255, je nastaveno opět číslo 1. Kategorie se automaticky nemění, neb pořadí kategorií se na jednotlivých závodech může lišit. Nejde-li o hlídku, dojde rovnou k navýšení startovního čísla a pořadí jezdce je nulové. Od řádku 244 do řádku 252 dojde k vypsání nového startovního čísla na displej.

Po vynulování proměnné „kod“ se na řádku 254 spustí cyklus, který postupně čte datové pole na adresách 10 (desítky) a 11(jednotky), stejným způsobem 13 a 14, 16 a 17 a na závěr 19 a 20. Výsledná hodnota trestných bodů je uložena do pomocné proměnné a převedena na kód dle tab. 5. Tak, aby se případné překlepy zaokrouhlily směrem k nejbližší správné hodnotě, vis řádky 264 až 271. Na řádku 273 dojde k bitovému pootočení o příslušnou hodnotu odpovídající pozici čtené brány a k logickému příčtení proměnné a k proměnné „kod“, čímž se postupně všechny průjezdy uloží do proměnné „kod“ a cyklus skončí. Kód se opět uloží na příslušnou adresu EEPROM. Tímto je celý záznam uložen a proměnná „adr“ se uloží do datového pole adresy[0], jako poslední uložený záznam. Tím, že k uložení posledního záznamu dojde až po uložení posledního ukládaného bajtu, odkazuje tato adresa vždy na poslední platný, tedy kompletní záznam.

Od řádku 279 dojde k vymazání displeje na pozicích trestných bodu a ke snulování trestných bodů v datovém poli „obsahD“. Kurzor se nastaví na první pozici zápisu trestných bodů. Na řádku 278 je čekací cyklus, který je nutný vložit při přechodu mezi komunikací s displejem a s pamětí, jinak dochází k chybám v datovém přenosu.

Vyhledání záznamu

```
291)    else
292)    {
293)    kod=obsahD[5];
294)    if (kod<3)
295)      kod=kod*100;
296)    a=obsahD[6];
297)    if (a<10)
298)      kod=kod+(10*a);
299)    a=obsahD[7];
300)    if (a<10)
301)      kod=kod+a;
302)    i=1;
303)    for(adr=(adresy[0]-1);adr>=3 && i==1;adr=adr-3)
```

```
304)          {
305)           b=zaznamy[adr];
306)           if (b==kod)
307)              {
308)              a=zaznamy[adr-1];
309)              b=a&7;
310)              if (b==obsahD[0])
311)                 {
312)                 b=a&128;
313)                  b=b>>7;
314)                  b++;
315)                  if (b==obsahD[3])
316)                  {
317)                  b=a&96;
318)                  b=b>>5;
319)                  if (b==obsahD[9])
320)                  {
321)              a=zaznamy[adr+1];
322)              cekej(100);
323)              povel(197);
324)              kurzor=69;
325)              cekej(40);
326)              for (i=0;i<4;i++)
327)                 {
328)                 b=3;
329)                 b=b<<(i*2);
330)                 kod=a&b;
331)                 kod=kod>>(i*2);
332)                 if (kod>2)
333)                    kod=50;
334)                 vypisZnak((kod/10)+48);
335)                vypisZnak((kod%10)+48);
336)                if (i!=3)
337)                   vypisZnak(32);
338)                }
339)              i=0;
340)             }
341)           }
342)         }
343)        }
344)       }
345)       if (i==1)
346)         {
347)          povel(197);
348)          kurzor=69;
349)          cekej(40);
350)           vypisZnak(78);
351)          }
352)        }
```

```
353)      }
```

V případě, že se kurzor nachází na 4. pozici druhého řádku, tedy na adrese 67, požaduji po systému vyhledání konkrétního záznamu. Na řádcích 293 až 301 stejným způsobem jako při ukládání načtu z „obsahD“ startovní číslo. Na řádku 303 začíná hledací cyklus, který začíná předposledním záznamem, kde se nachází poslední uložené startovní číslo. Cyklus postupuje s krokem -3, vzhledem k tomu, že záznam se skládá ze 3B proměnná „adr“ bude vždy odpovídat paměťovému místu s uloženým startovním číslem. Od posledního záznamu pokračuji z důvodu, že oprava se ukládá vždy na konec záznamů a tedy platný záznam by měl být vždy ten poslední v řadě. Startovní číslo se načte do proměnné „b“ a v podmínce na řádku 306 se porovná s proměnou „kod“, tedy s hledaným startovním číslem. Pokud se čísla neshodují cyklus pokračuje k následujícímu startovnímu číslu. Pokus se startovní číslo shoduje s hledaným, načte se z EEPROM paměti kódové slovo obsahující kód kategorie, ten je obsažen v prvních 3 bitech, proto se získá tím, že se daný bajt logicky vynásobí bajtem 00000111B. Druhému bajtu se říká maska. Pokud se shoduje kategorie s hledanou kategorií z toho samého kódového slova, se maskou 10000000B vymaskuje na řádku 312 pořadí jízdy v závodě, které se ještě kvůli porovnání musí na řádku 313 přesunout na první bit proměnné „b“ a pokračuje se stejným způsobem dál, jestli se jízda shoduje, porovná se ještě pořadí jezdce v hlídce, pokud se o hlídku nejedná, je jeho pořadí nulové. Když vše sedí, získal jsem v proměnné „adr“ adresu hledaného záznamu a můžu začít s výpisem hledaného záznamu na displej. Dekódování trestných bodů začíná na řádku 326 a je to obrácený postup kódování při ukládání, stejně jako převod na textový řetězec na řádcích 334 a 335. Po vypsání trestných bodů se snuluje proměnná „i“ a tím dojde k ukončení cyklu. Pokud se hledaný záznam nenajde, cyklus se ukončí dojetím na konec, vlastně na začátek záznamů a nedojde ke snulování proměnné „i“, když nedojde ke snulování „i“ dojde k vypsání „N“ na displej, podmínka na řádku 345. „N“ jako nenalezeno.

Odešly záznam

```
354)      void odesliZ()
355)      {
356)      unsigned int start;
357)      unsigned int stop;
358)      unsigned char pom;
359)      stop=adresy[0];
360)      start=adresy[1];
361)      cekej2(1);
362)      DE=1;
363)      EA=0;
364)      if (stop>start)
365)        {
366)         start++;
367)         for (; start<=stop;start++)
368)            {
369)             pom=zaznamy[start];
370)             odesli(pom);
371)            }
372)         adresy[1]=stop;
373)         odesli(8);
```

```
374)      }
375)    else
376)      odesli(8);
377)    DE=0;
378)    EA=1;
379)    }
```

Do proměnné stop se načte adresa posledního uloženého záznamu a do proměnné start adresa posledního odeslaného záznamu. Pak se počká, až se převodník přepne zpět na příjem a nastavením signálu DE na 1 se vysílač MAX485 přepne na vysílaní. Snulováním bitu EA v registru IE se zakážou veškerá přerušení a nedojde k příjmu náhodného bajtu vlivem přechodových dějů na sběrnici. Podmínka na řádku 364 zjistí, jestli jsou data k odeslání, pokud ano, postupně načítá data z paměti a odesílá prostřednictvím funkce odešli. Po odeslání všech, do této doby neodeslaných dat, přepíše adresu posledního odeslaného záznamu a následně bajt o hodnotě 8, který znamená, že byl přenos ukončen. Ukončovacím bajtem je hodnota 8 z důvodu, že tato hodnota jinak nastat nemůže, jde o rezervní hodnotu z trestných bodů, kdybych ji chtěl využit pro plánované DNF, nesměla by se pouze vyskytovat na pozici 5. brány. Pokud nejsou data k odeslání, pošle se pouze ukončovací bajt. Na závěr se opět přepne MAX485 na příjem (DE=0) a povolí se zpět přerušení programu.

Brány

```
380)    void brany()
381)    {
382)    unsigned char i;
383)    unsigned char a;
384)    povel(133);
385)    kurzor=5;
386)    cekej(40);
387)    for (i=1;i<5;i++)
388)      {
389)      cekej(100);
390)      a=cislaB[i];
391)      cekej(100);
392)      vypisZnak((a/10)+48);
393)      vypisZnak((a%10)+48);
394)      povel(129+kurzor);
395)      kurzor=kurzor+1;
396)      cekej(40);
397)      }
398)    povel(128);
399)    kurzor=0;
400)    cekej(40);
401)    }
```

Funkce slouží k vypsání čísel bran daného postu na 1. řádek displeje od adresy 5. Čísla bran jsou uložena v EEPROM v datovém poli „cislaB“. Všechny použité principy, kterými funkce brány pracuje, jsem popsal již výše.

Nastavení

```
402)    void nastaveni (unsigned char a)
403)    {
404)    a=a-48;
405)    cislaB[brana]=a;
406)    brana++;
407)    cekej(100);
408)    if (brana>5)
409)      {
410)       SM2=1;
411)       brana=0;
412)       brany();
413)       }
414)    }
```

Tato funkce uloží přijaté číslo závodní brány z PC do paměti EEPROM v záznamovém zařízení do datového pole „cislaB“. Po přijetí páté, tedy poslední brány, nastavení bitu SM2 povolí víceprocesorovou komunikaci, tedy k přerušení programu dojde jen tehdy, když přijatý devátý bit po sériové lince bude roven log. 1. Po nastavení tohoto bytu přerušení programu vyvolá jen příjem příkazu a žádná jiná data již přerušení nevyvolají. Dojde ke snulování globální proměnné „brana“ a přes funkci „brany“ k vypsání čísel bran na displej. Princip víceprocesorové komunikace vysvětlím později. Tento způsob jsem zvolil z důvodu, že číslo konkrétní brány se v komunikačním protokolu nepřenáší, jen číslo postu a pořadí brán, z kterých si počítač číslo brány dosadí. Tento způsob nastavení zaručí, že jak v zařízení, tak v PC budou čísla bran zadána stejně.

Smaž paměť

```
415)    void smazP()
416)    {
417)    adresy[0]= 3;
418)    adresy[1]= 3;
419)    }
```

Funkce skutečně paměť EEPROM nemaže, ale nastaví adresy poslední zapsaný a poslední odeslaný záznam na výchozí hodnoty a tak následně dojde k přepsání paměti.

Stáhni paměť

```
420)    void stahniP()
421)    {
422)    unsigned char d;
423)    unsigned int adr;
424)    unsigned int adrPom;
425)    cekej2(1);
426)    DE=1;    //Povoli vysilani
427)    EA=0;    //Zakaze preruseni
428)    for (adr=1;adr<6;adr++)
429)      {
```

```
430)        d=cislaB[adr];
431)        odesli(d);
432)       }
433)     adrPom = adresy[0];
434)     for (adr=0;adr<=adrPom;adr++)
435)       {
436)        d=zaznamy[adr];
437)        odesli(d);
438)       }
439)     odesli(8);
440)     DE=0;
441)     EA=1;
442)     }
```

Tato funkce stáhne všechny záznamy z paměti EEPROM, tedy uložená čísla závodních bran a všechny uložené záznamy o průjezdech danými branami. O způsobu realizace se již zmiňovat nebudu, vychází z práce s pamětí, kterou jsem v této kapitole již popsal.

Příjem

```
443)     void prijem() interrupt 4
444)     {
445)     unsigned char a;
446)     a=SBUF;
447)     RI=0;
448)     RB8=0;
449)     if (brana!=0)
450)       nastaveni(a);
451)     else
452)       {
453)       if (brana ==0)
454)          {
455)           if (a==71 )
456)              {
457)               SM2=0;
458)               brana=1;
459)              }
460)          if (a==72 )
461)             odesliZ();
462)          if (a==73 )
463)             stahniP();
464)           if (a==74 )
465)             smazP();
466)          }
467)       }
468)     }
```

Funkce příjem je tvořena obsluhou přerušení od sériové linky. První, co se musí při vyvolání přerušení udělat, je uložit obsah registru SBUF, aby nedošlo k přepsání dat dalším přijatým bajtem.

S výchozím nastavením je bit SM2 nastaven a k přerušení může dojít jen tehdy, je-li po přijetí stop-bitu i bit RB8 = 1., tímto bitem jsou odlišena data od příkazů. Hodnota příkazu je dána pořadovým číslem záznamového zařízení, které udává desítky dané hodnoty příkazu a číselným kódem příkazu, které udává jednotky daného příkazu, například příkaz 1 k záznamovému zařízení má hodnotu 71, jak je vidět ve výše uvedeném zdrojovém kódu. Dojde-li k přijetí tohoto kódu a je-li globální proměnná brana = 0, která zajistí, aby nedošlo k vyvolání příkazu i příjetím čísla brány se shodným číslem, dojde k vypnutí meziprocesorové komunikace a přerušení od této chvíle vyvolá kterýkoli přijatý stop-bit. A dojde k nastavení proměnné brana na 1. Od této chvíle se každý přijatý bajt bere jako číslo závodní brány a bude zpracován funkcí nastavení.

O příkazech 72, 73 a 74 se již podrobněji zmiňovat nebude, jejich funkce je zřejmá z již napsaného.

Hlavní program

```
466)    void main()
467)    {
468)    signed char z;
469)    unsigned char s;
470)    code unsigned char tab[]={49,50,51,10,52,53,54,1,55,56,57,2,3,48,4,13};
471)    inicializace();
472)    pozdrav();
473)    cekej2(60);
474)    cekej2(30);
480)    povel(1);
481)    cekej2(2);
482)    TH1=152;
483)    TMOD=32;
484)    TR1=1;
485)    SCON=240;
486)    PCON=128;
487)    ES=1;
488)    EA=1;
489)    RB8=0;
490)    TB8=0;
491)    brana=0;
492)    brany();
493)    DE=0;
494)    povel(128);
495)    cekej(40);
496)    kurzor=0;
497)    while(1)
498)      {
499)       z=klavesnice();
500)       if (z!=-1)
501)          {
502)           s=tab[z];
503)           if (s>31)
504)              {
```

```
505)                vypisZnak(s);
506)                kategorie(s);
507)                }
508)             else
509)                {
510)                if (s==1)
511)                    smazZnak();
512)                if (s==3)
513)                    posunLeva();
514)                if (s==4)
515)                   posunPrava();
516)                if(s==10)
517)                   uloz();
518)                if(s==13)
519)                    niz();
520)                if(s==2)
521)                    vys();
522)                }
523)              if(kurzor>3 && kurzor<64)
524)                  {
525)                   kurzor=64;
526)                   povel(192);
527)                   cekej(40);
528)                  }
529)               if(kurzor>79)
530)                  {
531)                   kurzor=79;
532)                   povel(207);
533)                   cekej(40);
534)                  }
535)                if (kurzor==71 || kurzor==74 || kurzor==77)
536)                    vypisZnak(32);
537)           }
538)         }
539)       }
```

Deklarace bajtových proměnných „z“ pro kód klávesy a do „s“ ukládám převedený kód klávesy na ASCII kód přes dekódovací tabulku „tab“, uloženou v datovém poli.

Inicializace je nezbytná pro nastavení displeje, vis funkce inicializace.

Pozdrav vypsání pozdravu, vis funkce pozdrav. Po vypsání následuje časová prodleva pro přečtení pozdravu a následně smazání displeje.

Nastavení čítače/časovače je provedeno v registrech TH1 a TMOD a to výběrem funkce časovače 1. Zapsáním hodnoty 32 do registru TMOD dojde k nastavení pouze bitu M1 pro daný časovač 1, vis tab. 17, tím vyberu režim časovače 1, vis tab. 18. Od časovače 1 je pak odvozena přenosová rychlost sériové linky.

Časový interval odměřovaný časovačem 1 je dán hodnotou 152 v registru TH1. Tato hodnota se vypočítá podle požadované přenosové rychlosti podle vzorce 1.

Nastavením bitu TR1 dojde ke spuštění časovače v registru TCON, který slouží pro řízení čítače /časovače.

Nastavením registru SCON na hodnotu 240, tedy 11110000B, vis tab. 21.
Nastavením bitů ES a EA dojde k povolení přerušení od sériové linky, vis tab. 1.

7	6	5	4	3	2	1	0
EA	-	ET2	ES	ET1	EX1	ET0	EX0

Tab 1 Registr IE

EA – povolení všech přerušení
ET2 – povolení přerušení od čítače/časovače 2 nebo T2EX
ES – povolení přerušení od sériového kanálu
ET1 – povolení přerušení od čítače/časovače 1
EX1 – povolení přerušení od INT1 (vnější přerušení 1)
ET0 – povolení přerušení od čítače/časovače 0
EX0 – povolení přerušení od INT0 (vnější přerušení 0)

Bity RB8 a TB8 slouží k nastavení vysílaného a přijímaného devátého bitu sériovou linkou, vis tab. 19.

Funkce brány po spuštění systému vypíše čísla bran uložená v EEPROM, tedy po restartu systému dojde k vypsání čísel bran, funkce je volána i po novém nastavení pozice postu.

DE = 0 povolí příjem RS485.

Následně dojde k nastavení kurzoru a první pozici displeje a k nulování proměnné kurzor. Tím je systém nastaven a připraven k použití.

Hlavní obslužný program záznamového zařízení začíná instrukcí WHILE(1), jde o nekonečný cyklus, který neustále testuje, jestli nedošlo ke stisknutí klávesnice funkcí klávesnice. V případě stisku klávesy zpracuje požadavek následujícím způsobem.

Po získání kódu klávesy podmínka na řádku 495 zjistí, jestli byla stisknutá klávesa, pokud ano, dojde k převedení kódu klávesy na ASCII kód znaku a nebo příkazu, vis datové pole dekódovací tabulky „tab“. Příkazům jsem přiřadil kód nižší než 32 a tak následující podmínka snadno rozezná, jde-li o znak a vypíše jej, nebo jde-li o příkaz, ke kterému je potřeba vybrat odpovídající funkci.

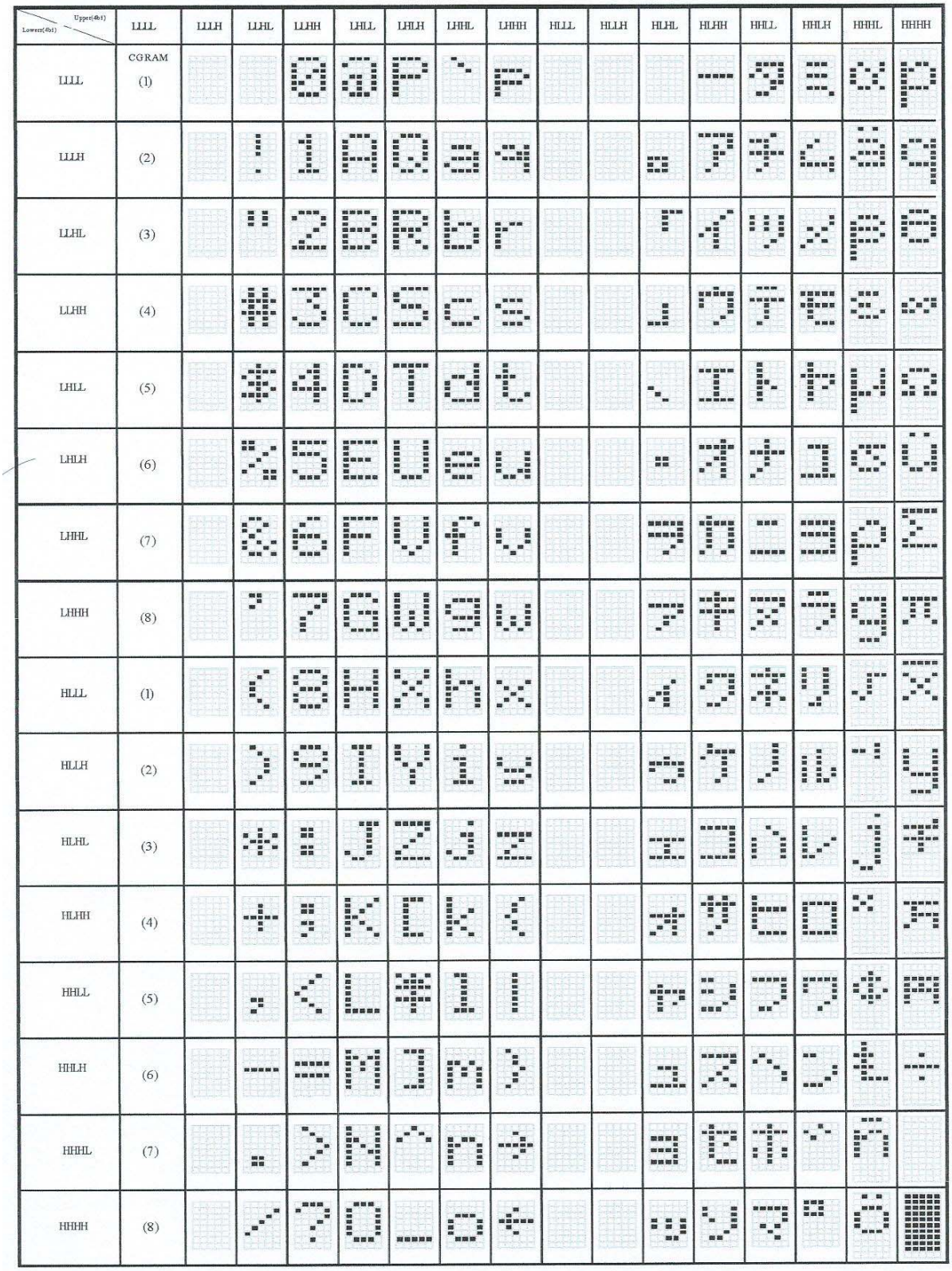
Podmínka na řádku 518 hlídá konec 1. řádku a v případě překročení konce řádku vrátí kurzor na první znak druhého řádku. První řádek je omezen jen ne 4 znaky, to proto, že další znaky jsou vymezeny k výpisu čísel závodních bran, které by obsluha přepisovat neměla. Kurzor se nastaví na druhý řádek z důvodu, že předpokládám, když už vyplním kategorii a jízdu, tak budu chtít pokračovat startovním číslem, které začíná právě na začátku druhého řádku.

Podmínka na řádku 524 hlídá poslední pozici displeje a v případě jejího překročení vrátí kurzor zpět na poslední pozici. Hlídat dorazy displeje je v tomto případě potřeba z důvodu, když by došlo k přechodu na řádek výš nebo níž mimo viditelnou oblast displeje, rozhodilo by se hlídání pozice kurzoru.

Poslední podmínka v těle programu zajišťuje přeskočení míst displeje, které slouží k oddělení sloupců na displeji pří pohybu vpravo.

Příloha 2

Uložení jednotlivých alfanumerických znaků v paměti LCD displeje



tab 2 Adresy jednotlivých znaků v paměti LCD displeje

Příloha 3

Obsah CD

Datasheet
Diplomová_práce
Kabelové_svazky
Plošný_spoj
Zdrojový_kód

Adresář Datasheet obsahuje katalogové listy hardwarových komponent.

Adresář Diplomová práce obsahuje kopii samotné diplomové práce ve formátu pdf.

Adresář Kabelové svazky obsahuje výkresy kabelových svazků pro záznamové zařízení ve formátu dwg.

Adresář Plošný spoj obsahuje návrh obvodového schéma záznamového zařízení a terminátoru záznamového zařízení vytvořeného v programu EAGLE, který je uložen v tomto adresáři v podadresáři EAGLE. Adresář dále obsahuje návrhy plošných spojů pro záznamové zařízení a terminátor záznamového zařízení, vytvořené v témže programu.

Adresář Zdrojový kód obsahuje soubor se samotným zdrojovým kódem programu pro záznamové zařízení v souboru s příponou „C“ a zkompilovaný soubor pro naprogramování záznamového zařízení ve formátu „HEX“. Dále v podadresáři Keil adresář obsahuje vývojové prostředí Keil µVision 4.