TECHNICKÁ UNIVRZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 - Elektrotechnika a informatika

Studijní obor: 2612R011 - Elektronické informační a řídící systémy

Časomíra pro závěrečné zkoušky

Clock for final exams

Bakalářská práce

Autor: **Jiří Mastík**

Vedoucí práce: doc. Ing. Zdeněk Plíva, Ph.D.

Konzultant: doc. Ing. Ivan Doležal, CSc.

V Liberci 25. 5. 2009

Prohlášení

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

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé práce a prohlašuji, že **souhlasím** s případným užitím mé práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do její skutečné výše).

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

Datum

Podpis

Poděkování

Děkuji vedoucímu bakalářské práce doc. Ing. Zdeňkovi Plívovi, Ph.D. a konzultantovi bakalářské práce doc. Ing. Ivanu Doležalovi, CSc. za účinnou odbornou pomoc a další cenné rady při tvorbě této bakalářské práce.

Abstrakt

Cílem této bakalářské práce je vytvořit časomíru, která by svým vzhledem, funkcemi a ovládáním byla použitelná u státních závěrečných zkoušek. V textu práce jsou stručně popsány možné metody řízení sedmisegmentového LED displeje, jenž časomíra využívá k zobrazování času. Dále jsou zde popsány: tvorba přesného časového intervalu, schémata, požadavky na časomíru, program pro mikrokontrolér Atmega32 a základní charakteristika tohoto mikrokontroléru, který tvoří logický základ časomíry. Jelikož má být časomíra využívána hlavně u státních závěrečných zkoušek je kladen velký důraz na vzhled krabičky časomíry, a proto obsahuje tato práce také popis výroby plechové krabičky pro tuto aplikaci. Časomíra je ovládána pomocí speciálně upravené fóliové klávesnice. Napájení časomíry je řešeno pomocí dvanáctivoltového spínaného zdroje.

Klíčová slova

Časomíra, Atmega32, závěrečné zkoušky, sedmisegmentový LED displej

Abstract

The aim of this bachelor's thesis is to produce a clock whose design, functions and method of control would make it suitable for use at the final graduation exams. This essay briefly discusses possible methods of operating the seven-segment LED display used by the clock to show time. The text further includes: description of creating an accurate time interval, diagrams, requirements posed on the clock, a programme for the Atmega32 microcontroller which serves as the logical core of the clock and also a basic description of it. As the clock is expected to be used mainly at the final graduation exams, the design of the clock box is of big importance. Therefore this text also describes the production of a sheet-metal box for this application. The clock is controlled by an adapted foil keypad and is powered by a 12V switching power supply.

Keywords

Clock, Atmega32, final exams, seven-segment LED display

Obsah

Seznam obrázků

Úvod

Cílem této bakalářské práce je navrhnout a zkonstruovat funkční vzorek časomíry, která by odpočítávala od uživatelem nastavené časové hodnoty do nuly. Časomíra má mít hlavní využití u státních závěrečných zkoušek, kde má informativním charakterem napomáhat k dodržování časového harmonogramu těchto zkoušek, při kterých je velice těžké pro komisi souběžně sledovat uplynulý čas a soustředit se při tom na jejich průběh. Nicméně časomíra má zároveň sloužit informativním charakterem i zkoušenému studentovi a to hlavně při prezentaci diplomové či bakalářské práce. Tímto informativním charakterem by měla studenta posunout k důležitějším částem jeho prezentace a zakončit tak obsahově rozsáhlou prezentaci v rozumném časovém intervalu s ohledem na další studenty v pořadí.

V první části textu je řešena problematika řízení sedmisegmentového displeje, který časomíra využívá. Dále jsou zde řešeny požadavky, které by časomíra měla splňovat vzhledem k využití u státních závěrečných zkoušek (vzhled, ovládání, funkce) a být tak uživatelsky příjemná. V druhé části bakalářské práce je již řešeno navržené zapojení, konstrukce krabičky, program, architektura a popis použitého mikrokontroléru, nastavení časovače tohoto mikrokontroléru pro přesné vytváření časového intervalu, který určuje přesnost časomíry. V přílohách práce jsou umístěny obrázky, nákresy, program, schémata a seznam součástek.

1 Možná řešení elektronických nastavitelných hodin

1.1 Způsoby zapojení a řízení displeje

Různé řešení elektronických hodin se v základu liší hlavně použitým displejem. Jako nejlepší zobrazovací prvek pro tuto aplikaci byl zvolen sedmisegmentový led displej (obr. 1.1). Samozřejmě by se mohl zdát zastaralý v době, ve které se nejčastěji používají různé LCD displeje a to hlavně díky tomu, že každý LCD displej má většinou integrovaný řadič. Z tohoto důvodu je pak poměrně jednoduché je připojeným mikrokontrolérem řídit a zapisovat na ně různé znaky, které daný displej podporuje.

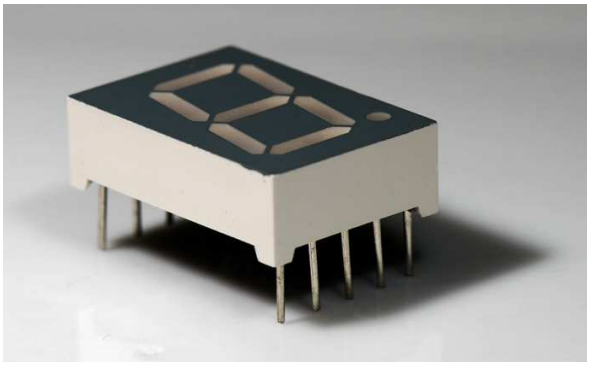
Sedmisegmentový LED displej byl vybrán z důvodu potřebnosti zobrazovat pouze čísla. Tento displej by měl také dát svou čitelnosti a velikostí dostatečný pozorovací komfort případným uživatelům.

Následující text stručně popisuje několik způsobů řízení sedmisegmentového displeje, které se dají použít pro nastavitelné hodiny a byly uvažovány jako možné řešení řízení sedmisegmentového displeje časomíry pro závěrečné zkoušky.

Sedmisegmentový displej se dá řídit v základu dvěma způsoby:

a) *Staticky*
b) *Dynamicky (multiplexně)*

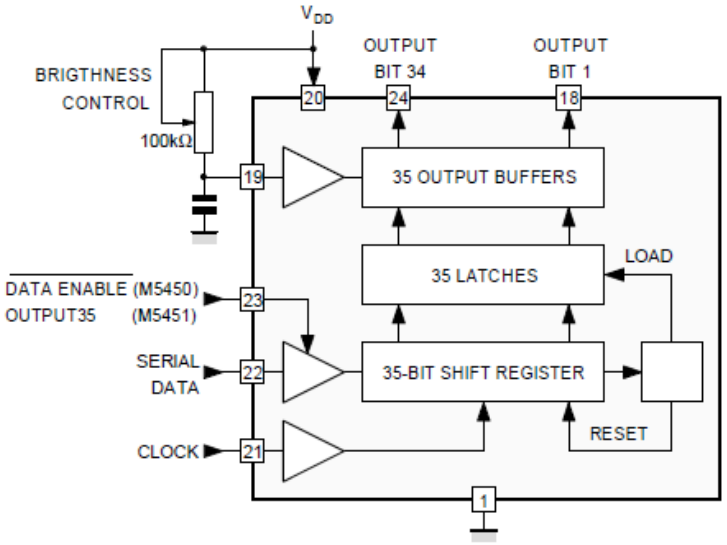
Oba způsoby mají samozřejmě svoje výhody a nevýhody, co se týče stability, náročnosti na počet vývodů mikrokontroléru a na jeho výpočetní zatížení.



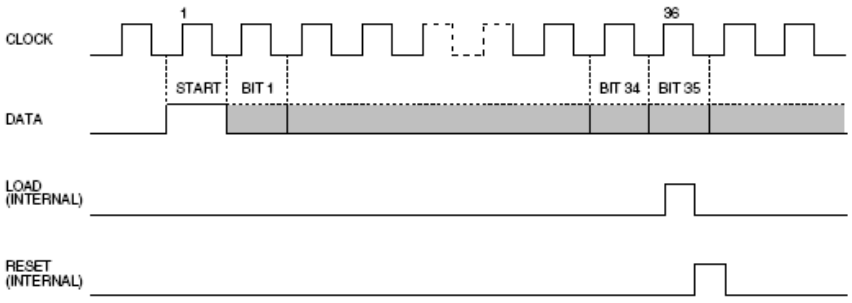
Obr. 1.1: Sedmisegmentový displej (převzato z [1])

1.1.1 Statické způsoby řízení displeje

Statické řízení sedmisegmentového displeje se, jak už název napovídá, nezaobírá problémy se stabilitou zobrazovaných dat, ale je náročné na počet vývodů řídícího prvku. K řízení se nejčastěji v praxi používají posuvné registry nebo řadiče, které mají většinou v základu také posuvné registry, pouze obsahují případné výkonové přizpůsobení k dosažení většího jasu, obvody pro řízení jasu atd.. Jednotlivé řadiče se liší maximálním proudem, kterým dokážou budit jednotlivé segmenty. Dále se také liší maximální velikostí sedmisegmentového displeje, který je na ně možné připojit. Tyto řadiče se pak ovládají sériovým vstupem viz. např. obvody M5451 a M5450. Oba tyto řadiče se liší pouze počtem výstupů. M5451 má 35 výstupů a M5450 pouze 34 výstupů, ale obsahuje oproti M5451 vstup DATA ENABLE. Funkci M5451 a M5450 popisuje blokové schéma (obr. 1.2) a datový formát komunikace (obr. 1.3).



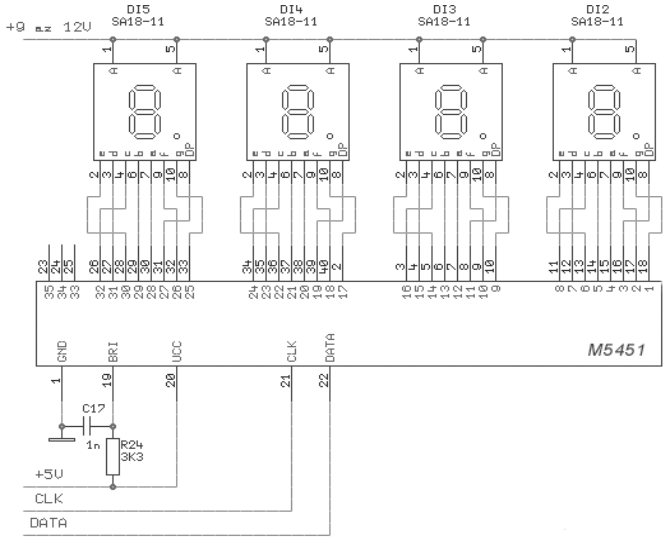
Obr. 1.2: Blokové schéma M5451-M5450 (převzato z [2])



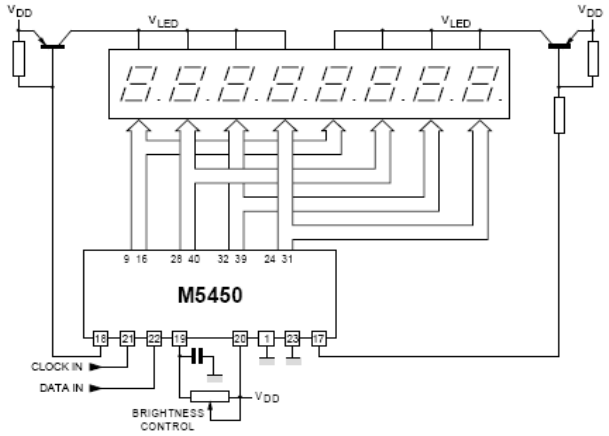
Obr. 1.3: Datový formát komunikace M5451-M5450 (převzato z [2])

Jak je vidět na datovém formátu k zápisu dat do řadiče je nutné nejdříve vyslat startbit a poté následuje 35 datových bitů (M5451). Následně jsou data zkopírována interním impulzem load do záchytného registru.

Mezi výhody, které plynou při využití tohoto řadiče, patří ovládání pomocí dvou vstupů řadiče, což klade minimální nároky na řídící mikrokontrolér, co se týče výstupů. Je také možná regulace jasu, řízení čtyř nebo pěti sedmisegmentů v případě M5451 (bez využití tečky sedmisegmentového displeje) ve statickém režimu (obr. 1.4) a až osm displejů v duplexním režimu (obr. 1.5). Nevýhoda je, že ve statickém režimu nelze řídit více jak pět displejů na jedné sériové lince, tzn. chybí možnost zřetězení.

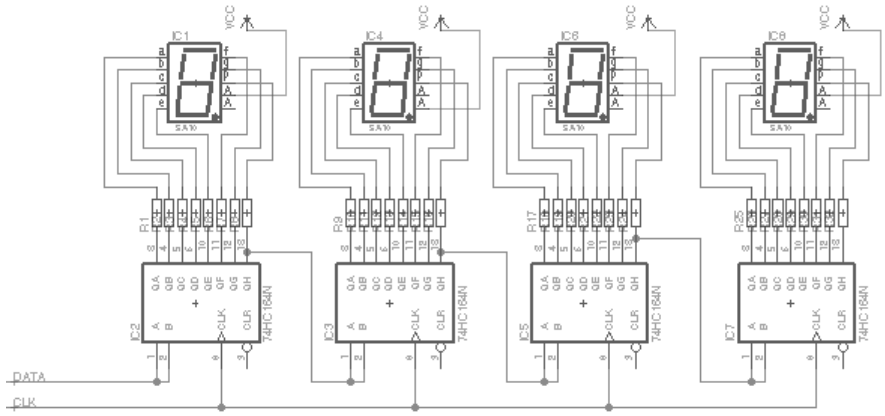


Obr. 1.4: Zapojení s M5451 (převzato z [3])



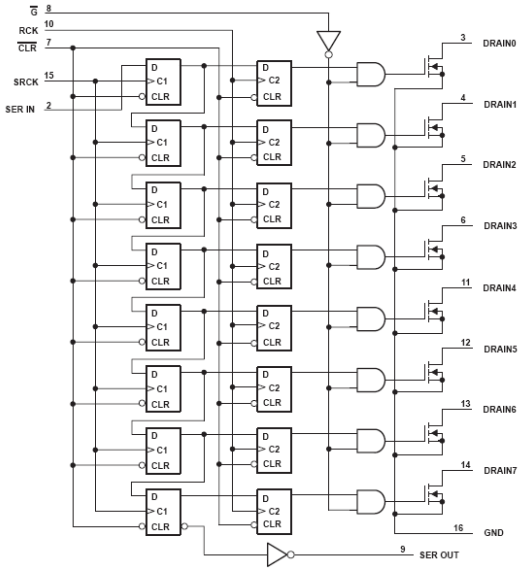
Obr. 1.5: Zapojení M5450 pro duplexní provoz (převzato z [2])

Dalším často používaným způsobem ke statickému řízení displeje, hlavně v amatérské praxi, je využití posuvných registrů 74HC164. Výhodou je možné neomezené zřetězení posuvných registrů a tím i neomezený počet možných displejů. Další výhoda je, že k řízení všech registrů postačí pouze jedna sériová linka. Hlavní nevýhody jsou velké množství integrovaných obvodů a rezistorů omezujících proud segmenty (obr. 1.6) a také absence záchytných registrů u posuvného registru 74HC164. Poslední problém lze například vyřešit nahrazením obvodu 74HC164 za obvod TPIC6C595 (obr. 1.7) a adekvátním upravením celkového schématu pro tento obvod.



Obr. 1.6: Statické řízeni displeje s využitím 74HC164

Obvod TPIC6C595 je kromě záchytného registru vybaven na výstupech DMOS tranzistory, což umožňuje spínat proudy až 100 mA, čímž lze dosáhnout vysokého jasu displeje.

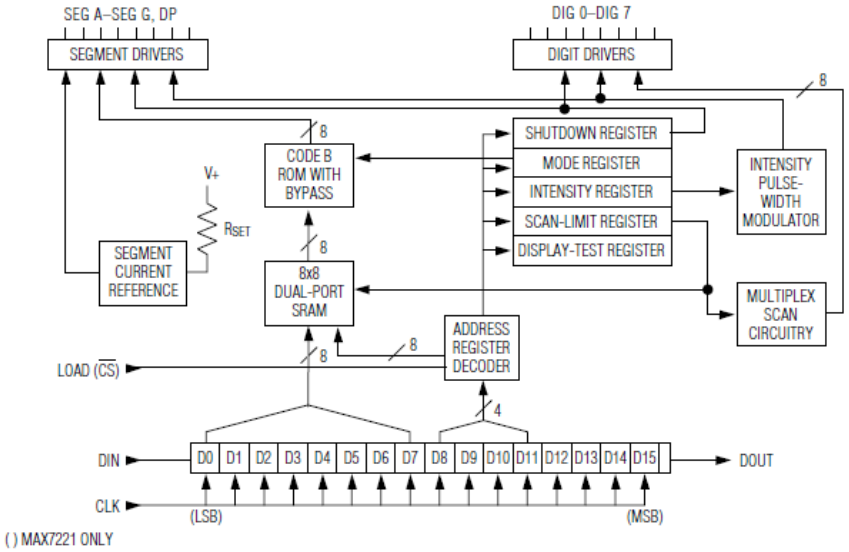


Obr. 1.7: Vnitřní zapojení TPIC6C595 (převzato z [4])

1.1.2 Dynamické způsoby řízení displeje

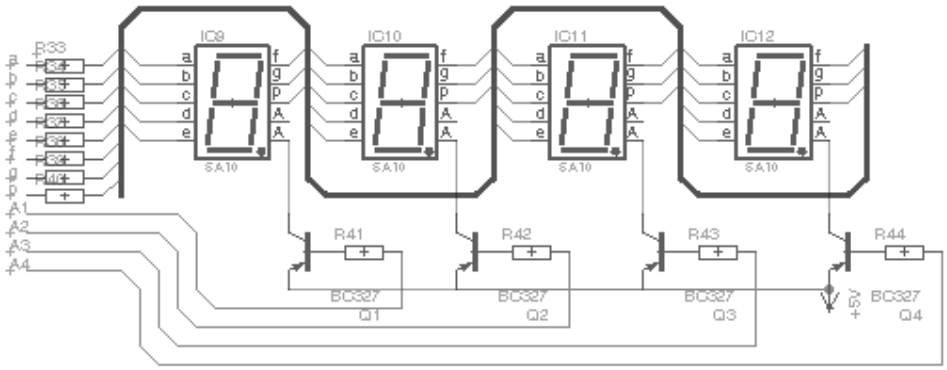
Dynamické, neboli multiplexní řízení displeje využívá nedokonalosti lidského oka, které má definovaný takzvaný „kritický kmitočet blikání“. Tento kmitočet je pro lidské oko přibližně 50 Hz. To znamená, že požadavek pro jednotlivé sedmisegmenty displeje je, aby frekvence obnovování zobrazovaných dat byla větší jak 50 Hz. Multiplexní řízení se také potýká s problémem zmenšující se střídy v závislosti na zvětšování počtu sedmisegmentů, což má za následek snížení jasu. Tato skutečnost se v praxi řeší zvětšením proudu jednotlivých segmentů. Velkou výhodou multiplexního řízení je minimální počet potřebných integrovaných obvodů k řízení. Samozřejmě i jako ve statickém řízení displeje lze využít řadič nebo často používanou metodu s využitím bipolárních tranzistorů, která je velice využívaná a postačí pro většinu aplikací.

Na obr. 1.8 je znázorněno blokové schéma obvodu MAX7221. Tímto obvodem je možno řídit jas jak připojeným externím rezistorem, tak i programově. Velkými výhodami tohoto obvodu je možnost adresování jednotlivých sedmisegmentů, řízení přes sériovou SPI sběrnici a rychlost zápisu dat, kdy hodinový signál může mít frekvenci až 10 Mhz, možnost zapojení až osmi sedmisegmentů se společnou katodou a možnost zřetězení s dalším MAX7221, které umožní řídit až šestnáct sedmisegmentů. Nevýhoda obvodu MAX7221 je, že buzení sedmisegmentů je prováděno proudovými zdroji, které dokážou sice vyvinout proud až 40 mA, ale doporučuje se využívat tento obvod pouze pro dvouvoltové displeje, jelikož proud při zvětšování zátěže rychle klesá, což limituje velikost displeje.



Obr. 1.8: Blokové schéma MAX7221 (převzato z [5])

Hojně používaným zapojením je zapojení, které využívá přímé řízení sedmisegmentů pomocí mikrokontroléru (obr. 1.9). K tomuto řízení se často u náročnějších aplikací používá speciální mikrokontrolér, jelikož multiplexní řízení displeje je poměrně výpočetně náročné. Programově bývá displej nejčastěji řízen v obsluze přerušení mikrokontroléru od časovače. Takto je možné zaručit periodickou obnovu dat na jednotlivých sedmisegmentech. Nevýhoda zapojení, jak je vidět z obr. 1.9 je větší náročnost na vývody mikrokontroléru. Tato nevýhoda se však dá jednoduše odstranit zapojením posuvných registrů do obvodu řízení segmentů, případně i anod. To s sebou nese ovšem větší výpočetní zatížitelnost, která plyne z neustálého plnění posuvných registrů v obsluze přerušení.



Obr. 1.9: Multiplexní řízení displeje

Upravené zapojení z obr. 1.9 je vzhledem k jednoduchosti řešení použito pro aplikaci časomíra pro závěrečné zkoušky. Důvod této volby plyne z jisté svobody k možnosti upravení obvodu z obr. 1.9 a tím možnost například použít i větší sedmisegmenty a také z malé náročnosti programu řídícího běh hodin, takže není problém řídit jedním mikrokontrolérem i displej hodin. K této volbě přispěl také fakt nepotřebnosti regulace jasu, které umožňují inteligentní řadiče a to z důvodu, že časomíra se bude výhradně používat ve dne za normálního osvětlení.

V kapitole tři jsou kompletní schémata pro časomíru pro závěrečné zkoušky. Obsahují upravené zapojení z obr. 1.9 pro větší sedmisegmenty a větší proud segmenty, který by byl na obr. 1.9 limitován maximálním proudovým zatížením výstupů mikrokontroléru.

2 Požadavky na časomíru pro státní závěrečné zkoušky

Pro časomíru bylo stanoveno několik základních požadavků, co se týče ovládání, vzhledu a velikosti. Tyto požadavky vycházejí hlavně z použití časomíry pro státní závěrečné zkoušky, i když jiné použití se nevylučuje.

Tyto požadavky jsou:

- Kompaktní a přijatelný design
- Snadné a rychlé ovládání
- Dva dostatečně veliké displeje
- Odpočítávání od nastavené hodnoty času
- Upozorňování na blížící se konec odpočítávání

Vzhledem k těmto základním požadavkům bylo zvoleno ovládání pomocí upravené fóliové klávesnice, velikost sedmisegmentových displejů 57 mm pro zobrazení minut a 38 mm pro zobrazení sekund, což umožňuje dostatečný pozorovací komfort. Dva displeje jsou zvoleny pro snadnou pozorovatelnost displeje časomíry komisí na straně jedné a studentem na straně druhé. Časomíra by samozřejmě měla z čistě praktických důvodů obsahovat upozornění na blížící se konec odpočítávání, například prezentace bakalářské práce, ovšem ne rušivým způsobem jako je využití například zvukových signálů, ale spíše využití světelných signálů, které jednoduše každý bez zbytečného vyrušení akceptuje a případně jim přizpůsobí další průběh výkladu.

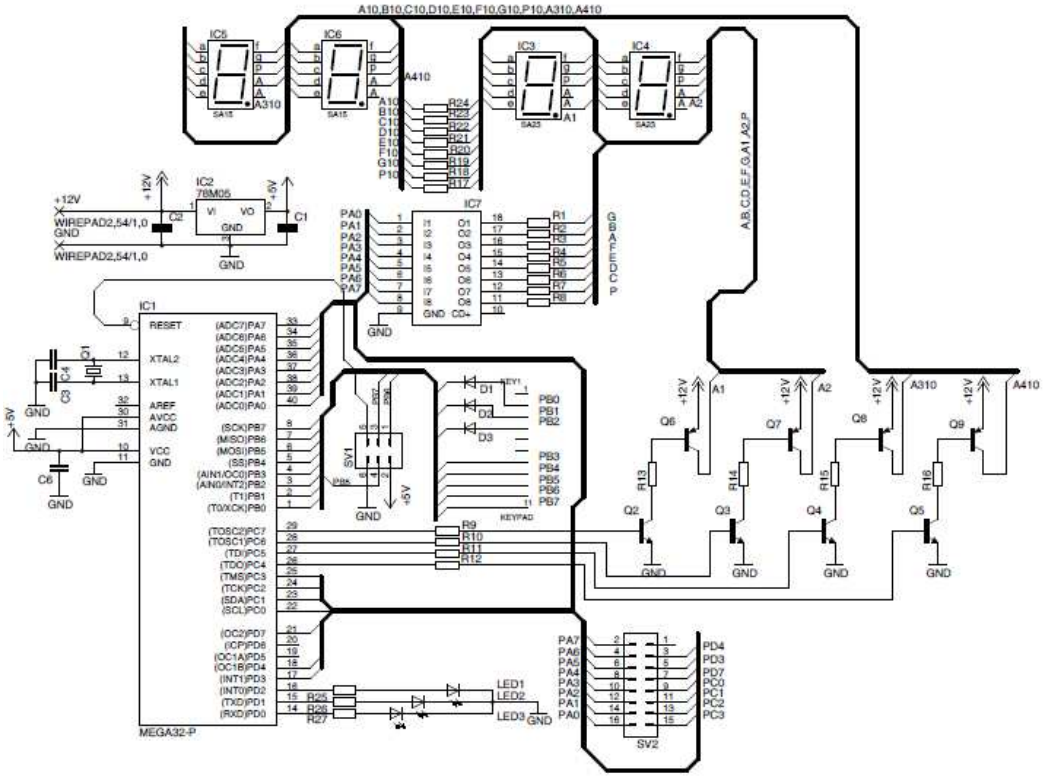
Co se týče konkrétního ovládání, funkcí, designu a konstrukce časomíry, bude popsáno v následujících kapitolách.

3 Schéma a popis zapojení časomíry

Základním požadavkem na časomíru je využití dvou displejů. To znamená, že časomíra se skládá ze dvou desek plošných spojů, které jsou vzájemně propojeny.

Na obr. 3.1 je zobrazeno schéma plošného spoje pro stranu, kde bude prováděno ovládání. Jak už bylo řečeno, je použito dynamické řízení displeje za využití bipolárních tranzistorů, které řídí anody sedmisegmentů. Z důvodu využití sedmisegmentů SA23-12SRWA (čtyři LED diody na segment) pro zobrazování minut je časomíra napájena pomocí dvanáctivoltového spínaného zdroje. Zobrazování sekund je realizováno sedmisegmenty SA15-11EWA (dvě LED diody na segment). Modelové označení spínaného zdroje je EAP-12-12 a maximální proud tohoto zdroje je jedna ampéra.

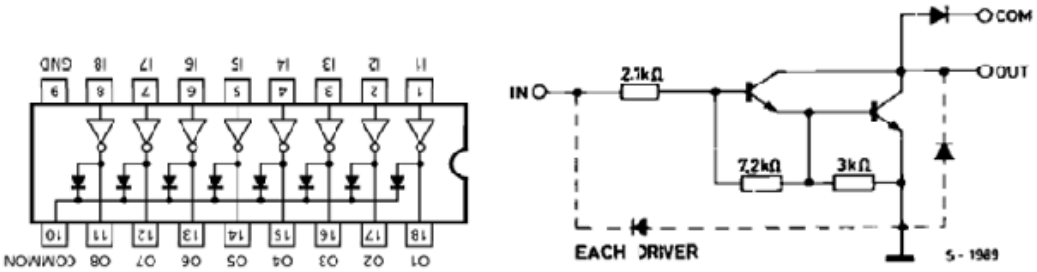
Z důvodů napájení displeje dvanácti volty je tedy nutno spínat anody sedmisegmentů pomocí dvojice tranzistorů, které pracují ve spínacím režimu. Ve schématu je patrné spínání tranzistoru NPN (BC817-16SMD) mikrokontrolérem (Atmega32) a následné spínání tranzistoru PNP (BC807-16SMD). Tato dvojice je nutná kvůli logickým úrovním na výstupech mikrokontroléru, kterými nelze přímo ovládat tranzistor PNP v tomto zapojení.



Obr. 3.1: Schéma zapojení pro stranu ovládání

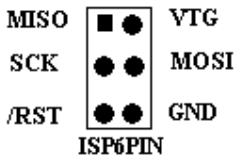
Na portu B mikrokontroléru je připojena klávesnice v maticovém zapojení. Klávesnice má pět řádků a tři sloupce. Diody, které jsou připojeny ke sloupcům, slouží proti zkratu výstupů mikrokontroléru, který může nastat při stisku dvou tlačítek v jednom řádku matice klávesnice.

K dosažení většího jasu displeje je do obvodu zařazen obvod ULN2803A, který umožňuje zvětšení proudu tekoucího jednotlivými segmenty na 40 mA, což by mělo v konečném důsledku dát postačující jas displeje. Obvod ULN2803A se v praxi využívá k buzení krokových motorů, LED displejů atd., vnitřní zapojení viz. obr. 3.2. ULN2803A obsahuje osm Darlingtonových tranzistorů s otevřeným kolektorem, každý může spínat proud až 500 mA a napětí do 50 V. Napětí na vstupech nesmí překročit 30V, vstupy jsou kompatibilní s TTL a CMOS logikou.



Obr. 3.2: Obvod ULN2803A a jeho vnitřní zapojení (převzato z [6])

Zapojení na obr. 3.1 obsahuje také šestipinový konektor, který umožňuje připojit ISP programátor. Zapojení konektoru koresponduje s popisem a se zapojením na obr. 3.3. Využití ISP programování umožňuje programovat mikrokontrolér přímo v daném zapojení a odpadá tak neustálé vyndávání a zandávání mikrokontroléru z patice pří odlaďování programu.

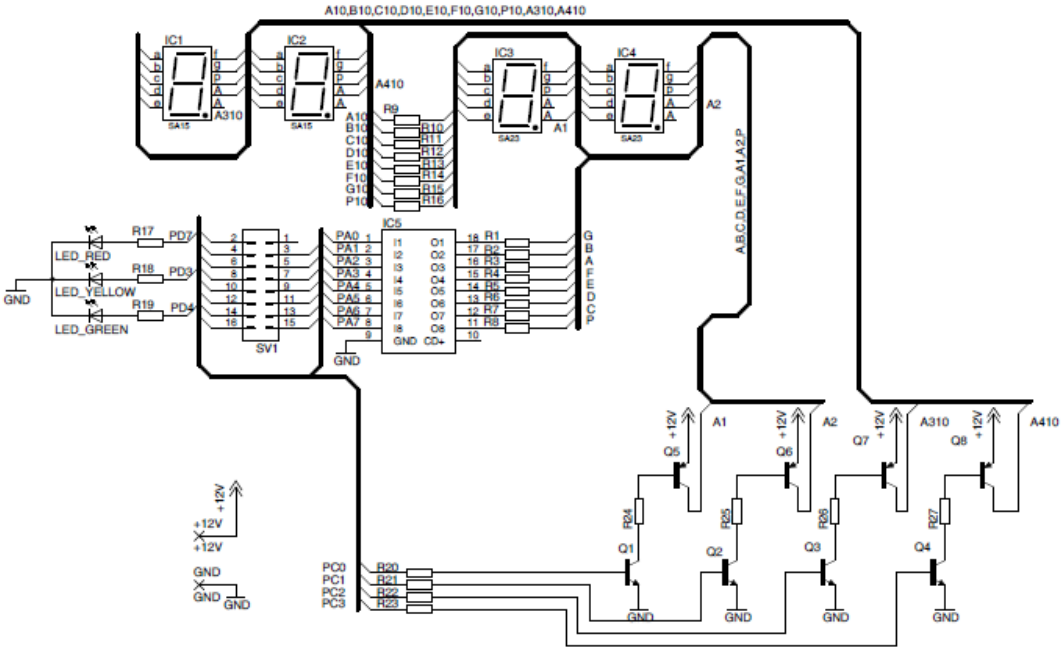


1	MISO
2	VTG (napájení z cílové desky)
3	SCK
4	MOSI
5	/RST
6	GND

Obr. 3.3: ISP (in-circuit serial programing) konektor

Na obr. 3.4 je znázorněno schéma plošného spoje obsahující displej pro pozorování ze strany studenta, tento plošný spoj je spojený s plošným spojem obsahujícím řídící mikrokontrolér pomocí šestnáctižilového plochého vodiče.

Samozřejmě zapojení obsahuje obdobné základní prvky k buzení displeje jako výše uvedené zapojení (obr. 3.1).



Obr. 3.4: Schéma zapojení pro stranu určenou zkoušenému

Plošné spoje, které přísluší k výše uvedeným schématům, jsou stejně jako schémata vytvořeny v prostředí EAGLE CAD 5.30 a pro ilustraci jsou zmenšené obrázky obou plošných spojů umístěny do přílohy D. Obě DPS využívají rozlité mědi připojené na GND.

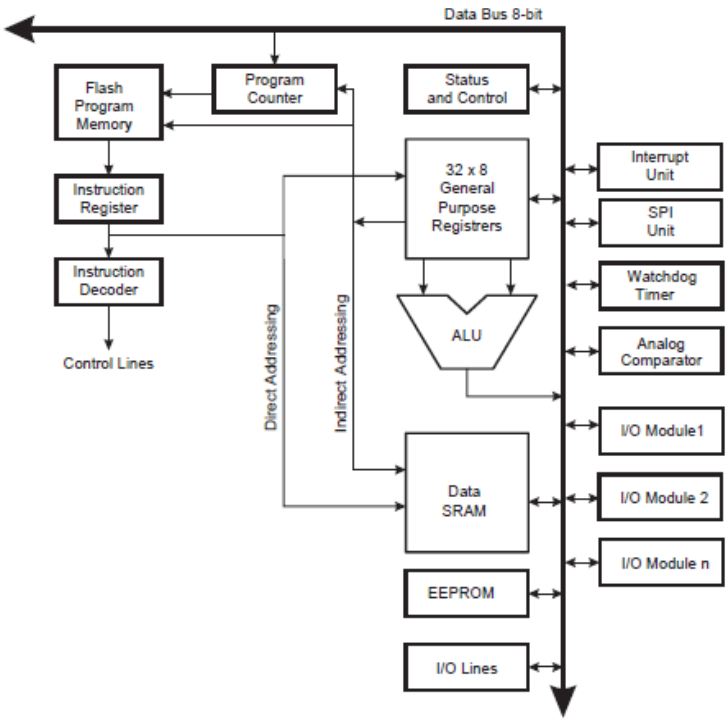
4 Mikrokontrolér

4.1 Architektura AVR

Mikrokontrolér Atmega32 určený k řízení časomíry patří do velice v dnešní době oblíbené rodiny mikrokontrolérů s architekturou AVR (obr. 4.1). Rodina AVR využívá architekturu RISC a je plně přizpůsobena pro efektivní programování v jazyce C. Rodina AVR se ovšem se svým počtem instrukcí 131 u výkonnějších typů velmi přibližuje CPU s architekturou CISC. Všechny ostatní charakteristiky, jako stejná bitová šířka instrukcí a zpracování instrukcí v jednom hodinovém cyklu jsou naopak typické vlastnosti architektury RISC. Dá se tedy konstatovat, že rodina mikrokontrolérů AVR má v sobě zakomponované výhody z obou architektur [10].

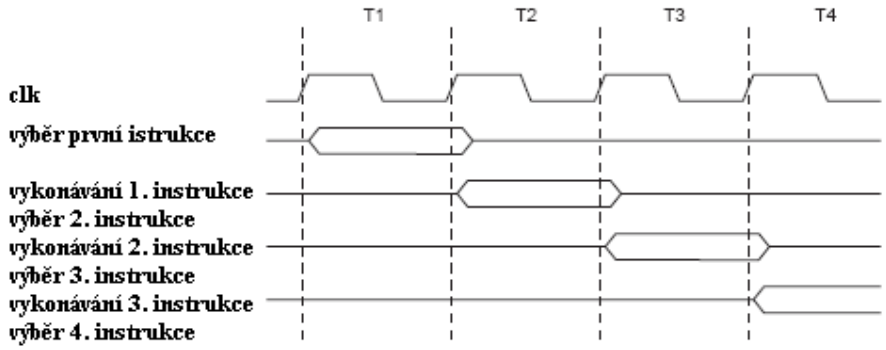
Instrukční kód má šířku 16 bitů a je tedy dostatečně velký, aby dokázal v jednom instrukčním slově pojmout jak instrukci, tak i operand. Jednostupňové zřetězení instrukcí neboli pipeline, podporuje zpracování instrukčního slova, přečtení, interpretaci a provedení v jednom hodinovém cyklu [10].

Rodina AVR využívá 32 univerzálních pracovních registrů. Tato skutečnost tedy umožňuje se vyhnout zbytečnému a zdlouhavému přemísťování obsahu registrů u aritmetických operací jako tomu je např. u mikrokontrolérů s jádrem 8051.



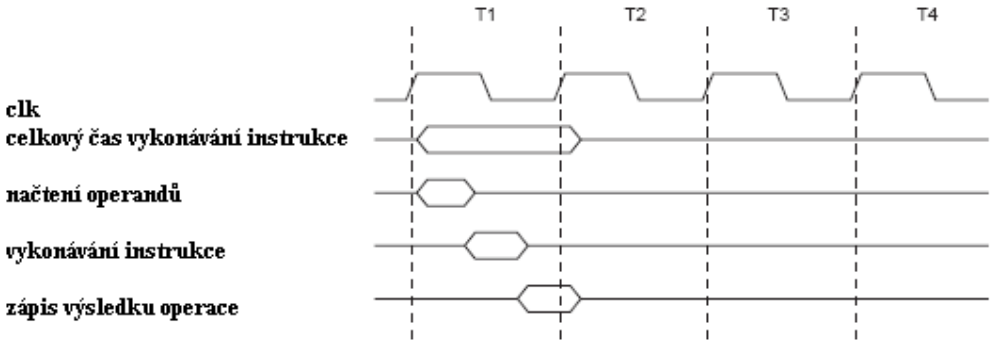
Obr. 4.1: Architektura AVR (převzato z [8])

Jak již bylo výše uvedeno mikrokontroléry AVR využívají zřetězené vykonávání instrukcí neboli pipelining. Konkrétně celý průběh probíhá tak, že nejdříve se z programové paměti vybere první instrukce. V následujícím taktu se tato instrukce vykonává a při tom se z paměti vybírá následující instrukce (prefetch) [9]. Celou situaci popisuje časový diagram na obr. 4.2.



Obr. 4.2: Pipelining a prefetch mikrokontroléru AVR (převzato z [8])

Na obr. 4.3 je časový průběh jednocyklového vykonávání aritmeticko-logické instrukce, kdy v jednom cyklu hodin jsou načteny oba operandy, vykonána instrukce a následně výsledek uložen do cílového registru.



Obr. 4.3: Jednocyklové vykonávání aritmeticko-logické instrukce (převzato z [8])

Naprostá většina instrukcí mikrokontrolérů AVR má délku 16 bitů, což zjednodušuje realizaci dekodéru instrukcí a spolu výše popsanými schopnostmi dává mikrokontrolérům AVR velký výpočetní výkon.

4.2 Mikrokontrolér Atmega32

Mikrokontrolér Atmega32 (obr. 4.4) byl vybrán hlavně z důvodu dostatečného počtu vstupně/výstupních portů, které umožňují řídit oba displeje, připojit ovládací klávesnici a stavové diody a také díky výše uvedeným vlastnostem jeho jádra AVR.

Základní charakteristika mikrokontroléru Atmega32 [8]:

- 131 instrukcí v instrukčním souboru
- 32x8 pracovních registrů
- Čtyři vstupně/výstupní porty
- Možnost připojení krystalu až 16 Mhz
- Datová paměť 2 KB, typu SRAM
- Programová paměť 32 KB
- Datová paměť 1024 B, typu EEPROM
- Velké množství periferií
- Power-on reset, Brown-out detekce
- Šest sleep módů
- Zabudovaný RC oscilátor

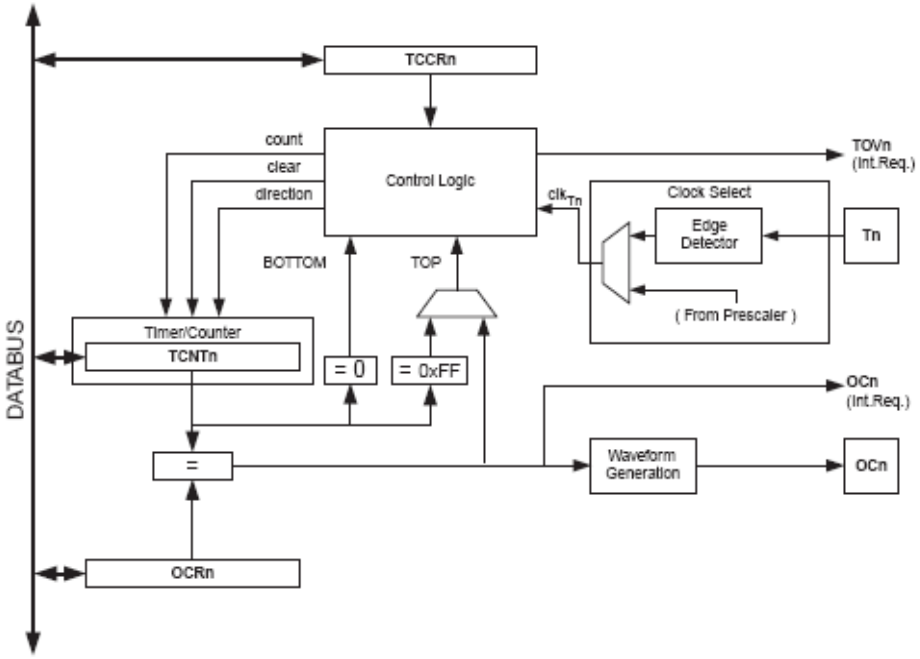
V příloze A je vyznačeno blokové schéma mikrokontroléru Atmega32 se všemi jeho periferiemi.

Obr. 4.4: Atmega32 v pouzdru PDIP (převzato z [7])

5 Generování časového intervalu

5.1 Čítač/časovač 0

Mikrokontrolér Atmega32 obsahuje tři čítače/časovače, dva jsou osmibitové a jeden šestnáctibitový. Pro vytvoření časového intervalu je využitý osmibitový čítač/časovač 0 čítající impulzy dělené děličkou k dosažení času jedné milisekundy. Z důvodu nastavení přesného časového intervalu je k mikrokontroléru připojen externí krystal generující frekvenci 5,12 Mhz. Odměření jedné sekundy probíhá pak programově čítáním počtu přerušení, které jsou generovány pomocí osmibitového čítače/časovače 0. Obr. 5.1 blokově popisuje zapojení čítače/časovače 0.



Obr. 5.1: Blokové schéma čítače/časovače0 (převzato z [8])

Časovač tedy může čítat impulzy odvozené od kmitočtu hodinového signálu mikrokontroléru a načítáním určitého počtu impulzů lze tedy odměřit jistý časový interval, jehož přesnost je závislá na přesnosti frekvence zdroje hodinového signálu.

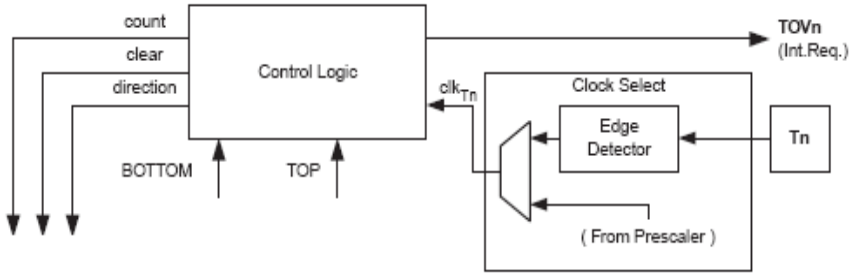
Základní vlastnosti čítače/časovače 0 [9]:

- Obecně použitelný osmibitový čítač/časovač
- Možnost připojení desetibitové předděličky pro hodinový signál
- Možnost generování přerušení v závislosti na přetečení nebo shodě s porovnávacím registrem

- Může pracovat jako generátor kmitočtu nebo čítač vnějších událostí
- Může být použit jako PWM generátor

Na obrázku 5.1 je zobrazeno blokové schéma čítače/časovače 0. Registr OCR0 interpretuje komparační registr k registru časovače TCNT0. Registr TCNT0 je tedy registr obsahující reálný stav časovače/čítače. Signály TOV0 a OCF0 jsou příznaky přerušení. Nastavení příznaku TOV0 signalizuje přerušení od přetečení čítače/časovače0 a příznak OCF0 signalizuje přerušení, které je požadováno od shody hodnoty registru časovače TCNT0 a komparačního registru OCF0. Oba tyto signály jsou obsaženy v registru přerušení TIFR [9].

Čítač/časovač 0 může čítat impulzy odvozené od pracovního kmitočtu mikrokontroléru nebo čítat impulzy přivedené na vstup vnějších hodin T0. Vybraný signál následně vstupuje do řídícího bloku (*control logic*).



Obr. 5.2: Řídící signály čítače/časovače (převzato z [8])

Na obr. 5.1 a na výřezu z něj na obr. 5.2 je definováno několik důležitých signálů potřebných k řízení čítače/časovače 0 [8].

- **Count** (čítání) – inkrementace nebo dekrementace obsahu TCNT0 o 1
- **Direction** (směr) – volba mezi inkrementací nebo dekrementací
- **Clear** (nulování) – nulování obsahu TCNT0
- **Clk_{T0}** – hodinový signál
- **Top** (vrchol) – signalizuje dosažení vrcholu časovače. To znamená, že TCNT0 = 0xff nebo bylo dosaženo shody TCNT0 a OCR0
- **Bottom** (dno) – signalizuje dosažení nulové hodnoty u TCNT0

5.2 Nastavení časovače pro časomíru

Pro časomíru je z hlediska jednoduchosti použit čítač/časovač 0 v režimu CTC (*Clear Timer on Compere Match*). Obr. 5.3 symbolizuje řídící registr čítače/časovače 0.

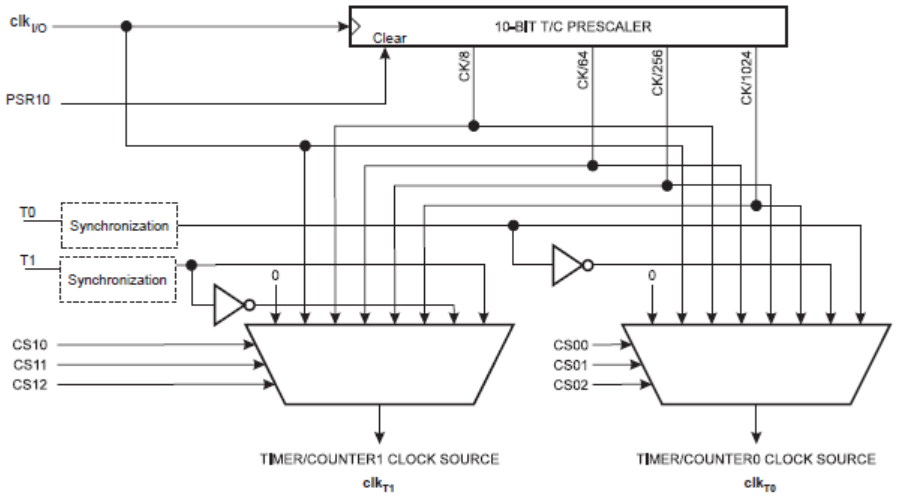
Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Obr. 5.3: Řídící registr čítače/časovače 0 (převzato z [8])

CTC režim se nastaví nastavením bitů v registru TCCR0; WGM01 do jedničky WGM00 do nuly. Funkce režimu CTC spočívá ve vynulování reálné hodnoty časovače, když se registr reálné hodnoty časovače TCNT0 dostane do stejné hodnoty jako má porovnávací registr OCR0. Při dosažení vrcholu TCNT0, který je dán hodnotou OCR0, se nastaví příznak OCF0 a je vyvoláno přerušení. Toto přerušení musí být samozřejmě povoleno [9].

Význam a nastavení dalších bitů registru TCCR0

- **COM00 a COM01** (režim OC vývodu) – řídí chování OC vývodu. Jsou tedy nastaveny pro odpojení OC vývodu od čítače/časovače 0, tzn. **COM00 = 0** a **COM01 = 0**
- **CS00 až CS02** – výběr hodin a předděličky. Jako zdroj je zvolen hodinový signál s předděličkou 64 (obr. 5.4), tzn. **CS00 = 1**, **CS01 = 1**, **CS02 = 0**
- **FOC0 –** slouží k vynucení OC výstupu, tzn. **FOC = 0**



Obr. 5.4: Předdělička mikrokontroléru Atmega32 (převzato z [8])

5.2.1 Výpočet hodnoty OCR0

Hodnota porovnávacího registru, která by měla být určena jako celé číslo z důvodu přesnosti odměřeného času, konkrétně jedné milisekundy. Z této podmínky byl tedy vybrán krystal o hodnotě frekvence 5,12 Mhz. Tato frekvence umožňuje přesné nastavení porovnávacího registru OCR0 a není nutné přistupovat k zaokrouhlování, které by vedlo ke kumulujícímu se předbíhání nebo zpožďování časomíry. Níže uvedený vzorec (5.1) a dosazení do něj (5.2) přesně určuje hodnotu porovnávacího registru OCR0.

$$OCR0 = \frac{f \cdot t}{p} - 1 \tag{5.1}$$

$$OCR0 = \frac{5{,}12 \cdot 10^6 \cdot 0{,}001}{64} - 1 = 79 \tag{5.2}$$

f – frekvence hodinového signálu, t – požadovaný čas, p – předdělička, OCR0 – porovnávací registr

Jak je patrné ze vzorce kmitočet neboli hodinový signál, který generuje krystal je dělen předděličkou (obr. 5.4) nastavenou na dělení hodinového signálu 64. Využití této předděličky umožňuje odměřit osmibitovým čítačem/časovačem 0 požadovaný časový interval při frekvenci hodinového signálu 5,12 Mhz.

6 Program

Program pro mikrokontrolér je napsán pomocí jazyka C s využitím kompilátoru avr-gcc a knihovny avr-libc. Obě tyto části jsou obsaženy ve volně dostupném balíku WinAVR. Program byl vyvíjen v AVR Studiu 4 od firmy Atmel. Kompletní verze programu využívajícího knihovny dodávané s kompilátorem je umístěna do přílohy G.

Celý program je z hlediska přehlednosti rozdělen do funkčních bloků, které činí spolu s využitím vyššího programovacího jazyka C dobrou čitelnost programu. Program je dále z velké části koncipován jako stavový automat, který mění svoje stavy v závislosti na stisknutí určitého tlačítka klávesnice.

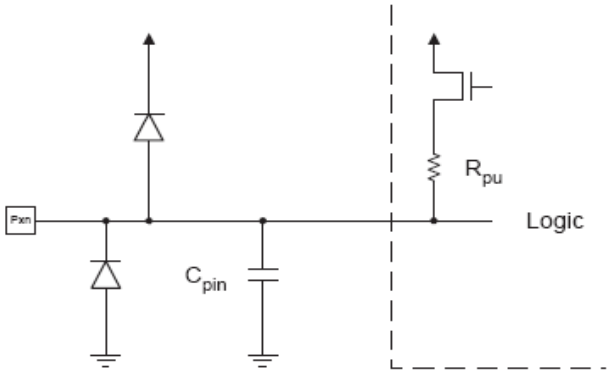
Níže uvedený text udává a popisuje důležité funkce programu pro mikrokontrolér Atmega32, který je řídícím prvkem časomíry.

```
Void anodes(uint8_t numblik,uint8_t anodes1,uint8_t anodes2,uint8_t digit);
```

Funkce má za úkol dynamicky řídit sedmisegmentový displej. Umožňuje blikání jednotlivých digitů na displeji z pohledu nastavujícího, zhasnutí celého displeje nebo blikání celého displeje z obou stran časomíry. Funkce je volána z obsluhy přerušení od čítače/časovače 0 každou jednu milisekundu. Je také využito paralelního řízení jednotlivých digitů displeje. To znamená, jsou-li zapnuty oba displeje, tak se jednotlivé digity spínají současně, např. minuty na obou stranách časomíry. Má to ovšem nevýhodu v proudu až 560 mA (svítí všechny segmenty kromě tečky), který je odebírán ze zdroje pro funkci displeje. Zároveň je udržena poměrně vhodná střída 1:3, která dává spolu s vyvoláním této funkce cca každou milisekundu obnovovací frekvencí displeje cca 250 Hz a dostatečný jas displeje pří proudu 40 mA na segment. Parametry funkce se samozřejmě mění v závislosti na tom, který digit nebo digity mají právě svítit.

```
Void keypad_buttons (void);
```

Funkce keypad_buttons testuje stisk jednotlivého tlačítka klávesnice. Tlačítka klávesnice jsou zapojeny do matice, což neumožňuje stisk více tlačítek najednou, ale je méně náročná na počet vývodů mikrokontroléru. Dále jsou na vstupech pro řádky využity interní pull-up rezistory mikrokontroléru viz. obr. 6.1. Funkce ve svém těle přepíná mezi sloupci a testuje řádky a následně při vyhodnocení stisku tlačítka se provádí odrušení. Odrušení probíhá cca jedenáct milisekund. Tato funkce je také volána v obsluze přerušení mikrokontroléru od čítače/časovače 0. Odpadá tak neustálé volání funkce v různých částech programu.



Obr 6.1: Znázornění pull-up rezistoru na vstupu

void settings (**void**);

Funkce settings slouží k výběru jednoho ze tří uložených časů v časomíře. Je volána z hlavní smyčky programu na základě stisknutí tlačítka SET a pouze jen je-li odpočítávání času zastaveno. Ve funkci je používáno často nepřímé adresování z důvodu zkrácení kódu programu.

Void settime (**uint8_t** *psec,**uint8_t** *pmin,**uint8_t** eeadress1,**uint8_t** eeadress2,**uint8_t** eeadress3);

Settime funkce má za úkol nastavení vždy jednoho ze tří možných časů časomíry. Funkce je volána z funkce settings a opět se v ní využívá nepřímého adresování. Nastavené hodnoty minut a sekund jsou pak uloženy do proměnných prostřednictvím pointerů, které má funkce ve svých parametrech. Nastavené hodnoty času jsou také uloženy do paměti EEPROM, která si uchovává svoje data i po odpojení napájení.

Void decode2(**uint8_t** seconds,**uint8_t** minutes);

Decode2 je funkce, která převádí počet sekund a minut na jednotky minut a sekund a na desítky minut a sekund. Ve svém obsahu tyto převedené hodnoty ukládá do určených proměnných, ty jsou pak dále dekódovány ve funkci k řízení displeje prostřednictvím pole a následně zapsány na PORTA mikrokontroléru. PORTA pak ovládá jednotlivé segmenty právě svítícího digitu.

7 Krabička pro časomíru

Krabička určuje výrazně svým vzhledem celkový dojem z časomíry a bylo tedy nutné podle toho k tomu přistoupit. Vzhledem k tomu, že na trhu není dostupná žádná krabička, která by se i po určitých úpravách dala využít, bylo nutné vytvořit krabičku přímo na míru této aplikace s přihlédnutím na dostupné technologie výroby.

Krabička je vyrobená z plechu o tloušťce 0,8 mm. Plech byl zvolen hlavně z důvodu odolnosti, možnosti ohýbání a možnosti snadné povrchové úpravy oproti jiným materiálům. Základ krabičky je vyřezán laserem až na některé otvory jako otvor pro vodiče klávesnice, otvory pro šroubky a otvor pro napájecí konektor, které byly vytvářeny dodatečně. Nákres krabičky je vytvořený v prostředí AutoCad 2009 a je umístěn do přílohy B.

Využití laseru při řezání plechu s sebou nese výhody vysoké přesnosti a nulové mechanické opotřebení výrobku, které vzniká například u mechanického stříhání nebo řezání.

Krabička se skládá ze dvou částí viz. příloha B.1 a B.2, které byly naohýbány na požadovaný tvar. Vzniklé nerovnosti v oblasti spojů jsou dorovnány karosářským tmelem. Obě části krabičky byly opatřeny lakem v podobě černé metalízy a poté, k dosažení požadovaného lesku, bezbarvým lakem. Do otvorů pro displeje jsou vlepeny červené filtry, které mají za úkol zvýšit čitelnost displeje při oslnění ostrým světlem. Plní také funkci optického sjednocení jednotlivých sedmisegmentů. Celý design hotové časomíry pak tvoří kompaktní celek.

V F.2 je vyobrazena krabička s nalepenou foliovou klávesnicí s původním označením kláves. Toto označení se ovšem neslučovalo s funkcemi časomíry, a tak bylo nutné vytvořit přelepku, která by tyto funkce svými popisy zastupovala. Přelepka obr. 7.4 byla vytvořena na zakázku podle návrhu vytvořeném v prostředí CorelDraw X4.

Obr. 7.1: Přelepka klávesnice (zmenšený obrázek návrhu)

8 Ovládání a funkce časomíry

8.1 Funkce časomíry

Hned po zapnutí časomíry se časomíra uvede do režimu výběru a nastavení času, od kterého se má odpočítávat. Je možné nastavit tři tyto časy z důvodu časově rozdílných částí státní závěrečné zkoušky. Každému nastavenému času odpovídá jedna LED dioda nad displejem. Po odstartování zvoleného času je možné odpočítávání kdykoliv zastavit a zase odstartovat. V režimu, kdy je časomíra zastavená je možné vyvolat nastavení jednotlivých časů nebo resetovat poslední nastavený. Je třeba také říci, že nastavené hodnoty jsou uloženy do paměti EEPROM mikrokontroléru, takže jsou zachovány i po odpojení napájení. To umožňuje připravit si časomíru před zkouškami a nezdržovat se pak nastavováním, i když je poměrně rychlé. Časomíra má také funkci upozorňování na blížící se konec zkoušky nebo prezentace v podobě semaforu zhotoveného z LED diod, které se zapínají v závislosti na čase, který zbývá do konce. Tyto časy nelze na časomíře nastavit, lze je nastavit pouze úpravou programu mikrokontroléru. Časový formát zapínání diod popisuje tabulka 8.1. Dalším upozornění, které časomíra obsahuje je upozornění, že odpočítávání došlo do nuly. Tento stav je signalizován na obou stranách blikajícím displejem.

Čas [min]	Barva
t > 5	zelená
2 < t ≤ 5	žlutá
0 ≤ t ≤ 2	červená

Tab. 8.1: Časové spínání led diod

8.2 Ovládání časomíry

Ovládání časomíry je odvozeno od požadavku snadného a rychlého ovládání. Následující text popisuje ovládání funkcí pomocí tlačítek klávesnice.

Tlačítko SET slouží v režimu výběru nastaveného času k vyvolání režimu nastavení času. Režim výběru nastaveného času je indikován blikající LED diodou umístěnou nad displejem. Po stisku tlačítka v tomto režimu se vyvolá režim nastavení času, při kterém nepřerušovaně svítí LED nad displejem a přerušovaně svítí digit nejvíce vlevo, tedy digit interpretující desetiminutovou číslici. Následně je možnost zadat hodnotu z numerické klávesnice. Zároveň je v tomto režimu displej na protější straně vypnutý. Při opětovném stisku SET je aktuální hodnota času uložena do paměti EEPROM a následuje návrat do režimu výběru nastaveného času. Režim výběru

nastaveného času lze tímto tlačítkem vyvolat jen, je-li odpočítávání zastaveno tlačítkem START/STOP nebo časový údaj časomíry dospěl do nuly. Režim výběru nastaveného času se také vyvolá automaticky při připojení napájení k časomíře.

Tlačítko ŠIPKA je funkční pouze v režimu výběru nastaveného času anebo v režimu nastavení času. V režimu výběru času lze tímto tlačítkem přepínat mezi nastavenými časy. Přepínání je signalizováno blikajícími LED diodami nad displejem. V režimu nastavení času má za úkol toto tlačítko přepínat mezi jednotlivými digity displeje (zleva doprava) podobně jako tomu je u digitálních budíků.

Tlačítko RST má pouze jednu funkci a funguje pouze v režimu, kdy je časomíra zastavená a není v režimech nastavení a výběru nastaveného časů. To znamená, že dioda nad displejem plně svítí a nebliká digit displeje. Při stisku tohoto tlačítka se znovu nastaví na časomíře poslední odstartovaný čas z režimu výběru nastaveného času nebo režimu nastavení času.

Tlačítko START/STOP, jak už název napovídá, má funkci odstartování zvoleného času. Odstartovat odpočítávání lze z režimu výběru času a nastavení času (v tomto režimu se při stisku START/STOP uloží aktuální hodnota do paměti EEPROM). Běží-li pak odpočítávání, lze tímto tlačítkem časomíru zastavit a opět odstartovat od času, který je aktuálně na displeji. Stisk tohoto tlačítka provází také zhasnutí obou displejů, které trvá 250 ms. Je to z důvodu použité klávesnice, která má téměř nulový zdvih tlačítek a tímto je tedy stisk signalizován. Při odstartování tedy časomíra čeká 250 ms a až pak se začne odpočítávat. Při zastavování časomíry je tento děj vykonáván opačně. Tímto tlačítkem lze i vypnout blikání displejů na obou stranách časomíry, když časomíra dospěla do nuly.

Závěr

Navržená časomíra splňuje požadavky na ovládání a vzhled. Díky velikosti obou displejů je její využití všestranné. Dá se například využít pro různé amatérské sportovní akce, i když by mohla vadit její imobilita vzniklá závislostí na síťové zásuvce. Možná větší uplatnění než u státních závěrečných zkoušek by mohla časomíra najít při různých zkouškách, přijímacích pohovorech a testech.

Co se týče rozdílnosti v jasu a odstínu zobrazovaných sekund od minut, je tento jev zapříčiněn rozdílností typů použitých segmentových displejů. Ale v konečném dojmu tento jev nijak neovlivňuje dojem z časomíry jako celku. Spíše přispívá ke zdůraznění minut jako důležitějšího údaje.

LED diody, které by měly upozorňovat na blížící se konec odpočítávání, se zdají v konečném důsledku zbytečné a to hlavně v porovnání s displejem, který svým jasem a velikostí strhává pozornost na sebe, nicméně splňují požadavek na rychlou informaci o zbývajícím čase.

Je třeba také říci, že časomíra byla od samého počátku konstruována pro napájení ze spínaného zdroje, aby se minimalizovala její případná průběžná údržba. Nicméně je možné zvolit v budoucnu i napájení z akumulátorů s dostatečnou kapacitou; vzhledem k uvažované spotřebě by bylo nutno řešit jejich snadnou a rychlou výměnu. Nároky na akumulátory z hlediska napětí, při zachování proudu jednotlivými segmenty a tím tedy i jasu displeje, lze zmírnit upravením zapojení časomíry a to odpovídajícím zmenšením velikosti hodnoty odporu rezistorů, které jsou přímo připojeny na výstupy obvodu ULN2803a. Nároky na proudový odběr lze například zmenšit změnou programu mikrokontroléru tím, že nebude využíváno paralelního spínání sedmisegmentových displejů nebo pomocí zmenšení proudu jednotlivých segmentů displeje. Pak je ovšem nutno počítat se snížením jasu displeje časomíry.

Použitá literatura

[1] Wikipedia. File: Seven segment 02 Pengo.jpg. [cit. 5.3.2009] URL: <en.wikipedia.org/wiki/File:Seven_segment_02_Pengo.jpg>

[2] SGS-Thomson Microelectronics. *Led display drivers.* SGS-Thomson Microelectronics 1994. [cit. 5.3.2009]. URL: <www.grifo.com/PRESS/DOC/ST/M5450_51.pdf>

[3] HW server. Řešení buzení segmentových displejů mikroprocesoru. HW server 2000. [cit.5.3.2009]. URL: <hw.cz/Teorie-apraxe/Navrhy-vyvojare/ART516-Reseni-buzeni-segmentovych-displeju-mikroprocesoru.html>

[4] Texas Instruments. *Power logic 8-bit shift register*. Texas Instruments 2005. [cit. 5.3.2009]. URL:<www.gme.cz/_dokumentace/dokumenty/955/955-044/dsh.955-044.1.pdf >

[5] Maxim Integrated Products. *Serially interfaced, 8-digit LED display drivers*. Maxim Integrated Products 2003. [cit. 5.3.2009]. URL: <datasheets.maximic.com/en/ds/MAX7219-MAX7221.pdf >

[6] Texas Instruments. *Darlington transistor array*. Texas Instruments 2006 [cit. 5.3.2009] URL: <focus.ti.com/lit/ds/symlink/uln2803a.pdf>

[7] Wikimedia. File: Atmel atmega32 mikrokontrolleri.jpg. [cit. 5.3.2009]. URL: <commons.wikimedia.org/wiki/File:Atmel_atmega32_mikrokontrolleri.jpg>

[8] Atmel Corporation. *8-bit AVR microcontroller with 32K bytes in-system programmable flash*, Atmel Corporation 2004. [cit. 5.3.2009]. URL:<www.gme.cz/_dokumentace/dokumenty/432/432-180/dsh.432-180.1.pdf>

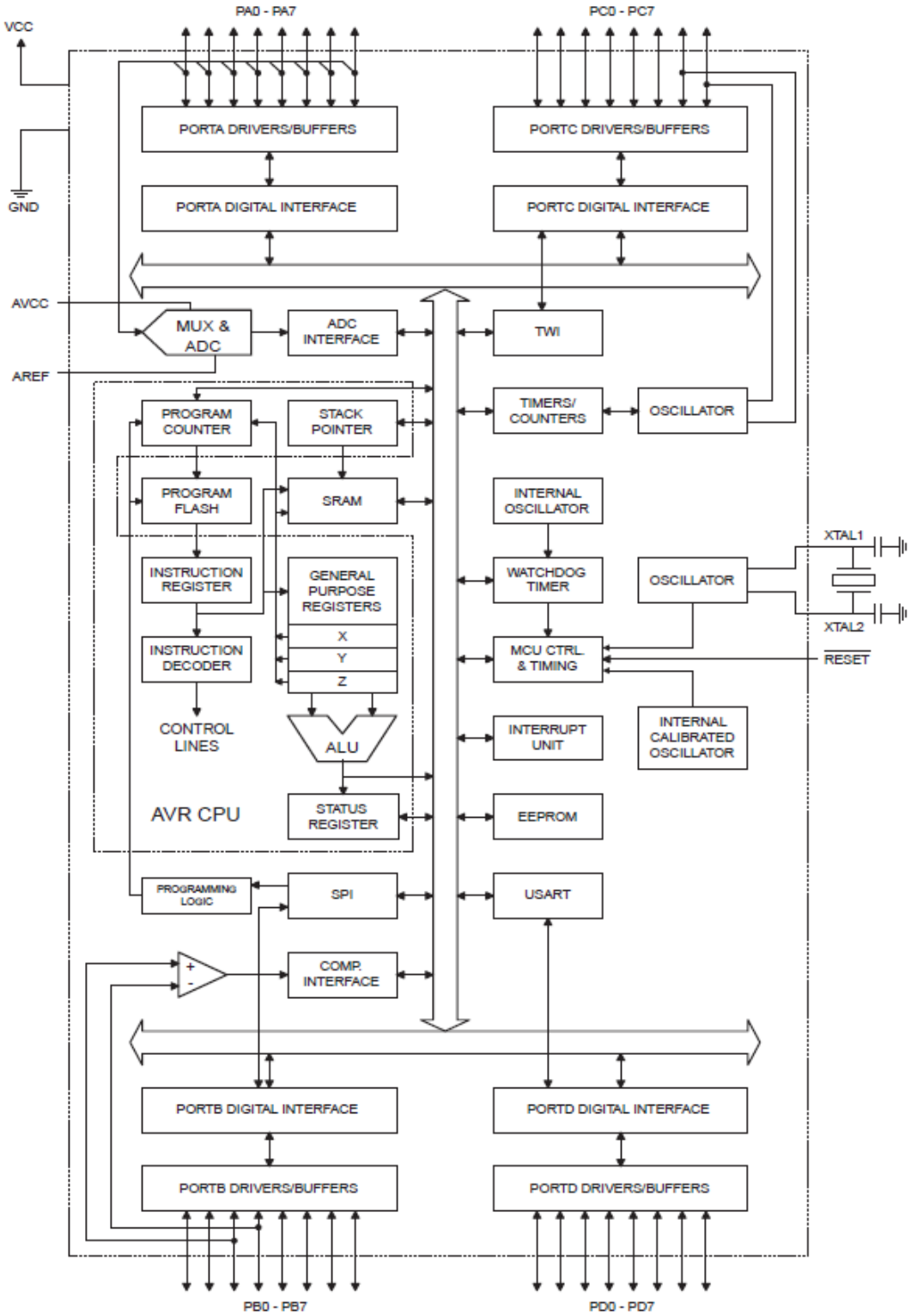
[9] MATOUŠEK, David. *Práce s mikrokontroléry Atmel Atmega16.* BEN – technická literatura, Praha 2006, ISBN 80-7300-174-8

[10] MANN, Burkhard. *C pro mikrokontroléry.* BEN – technická literatura, Praha 2003, ISBN 80-7300-077-6

Seznam příloh

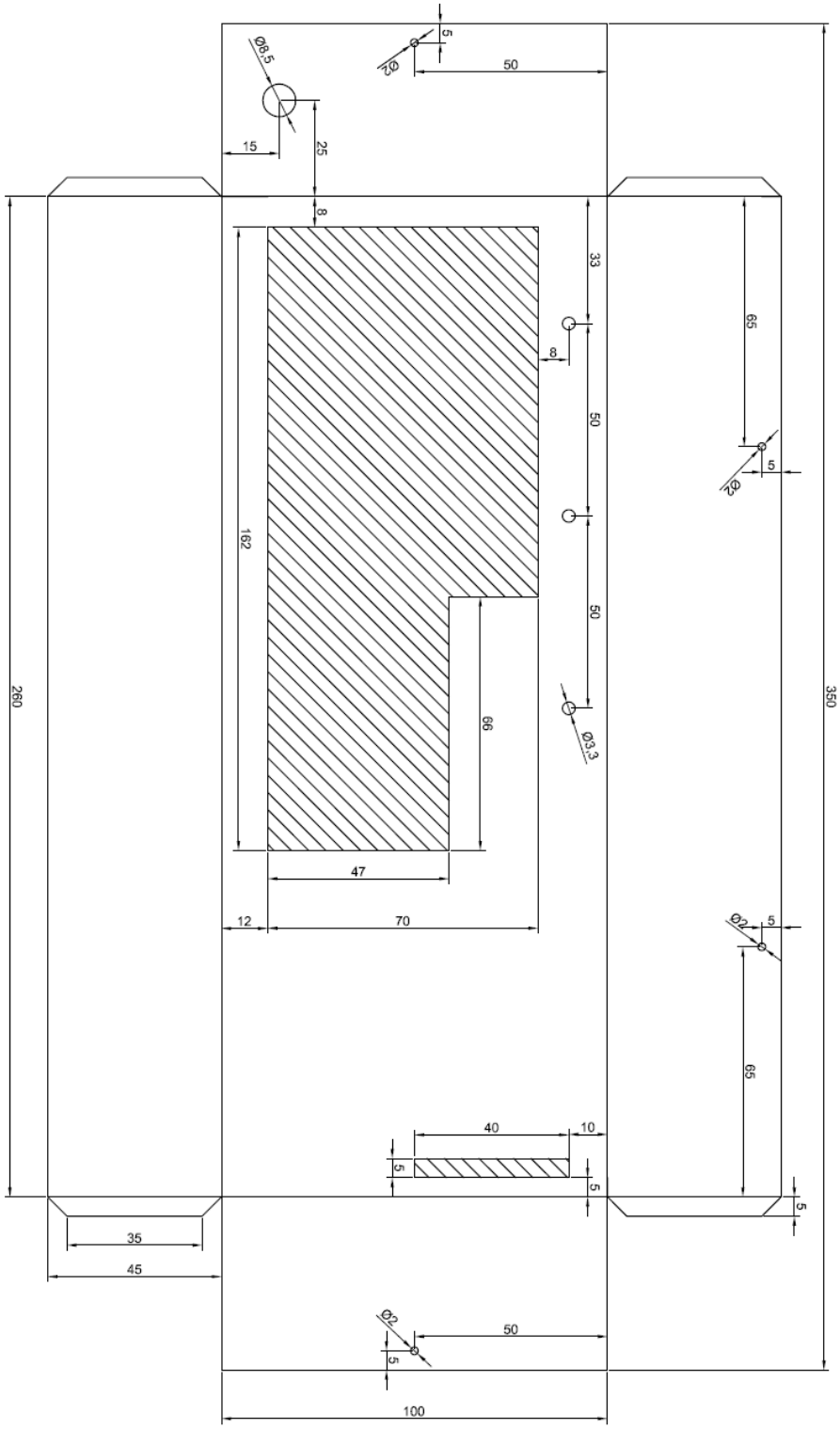
A Blokové schéma Atmega32

Blokové schéma převzato z [8].

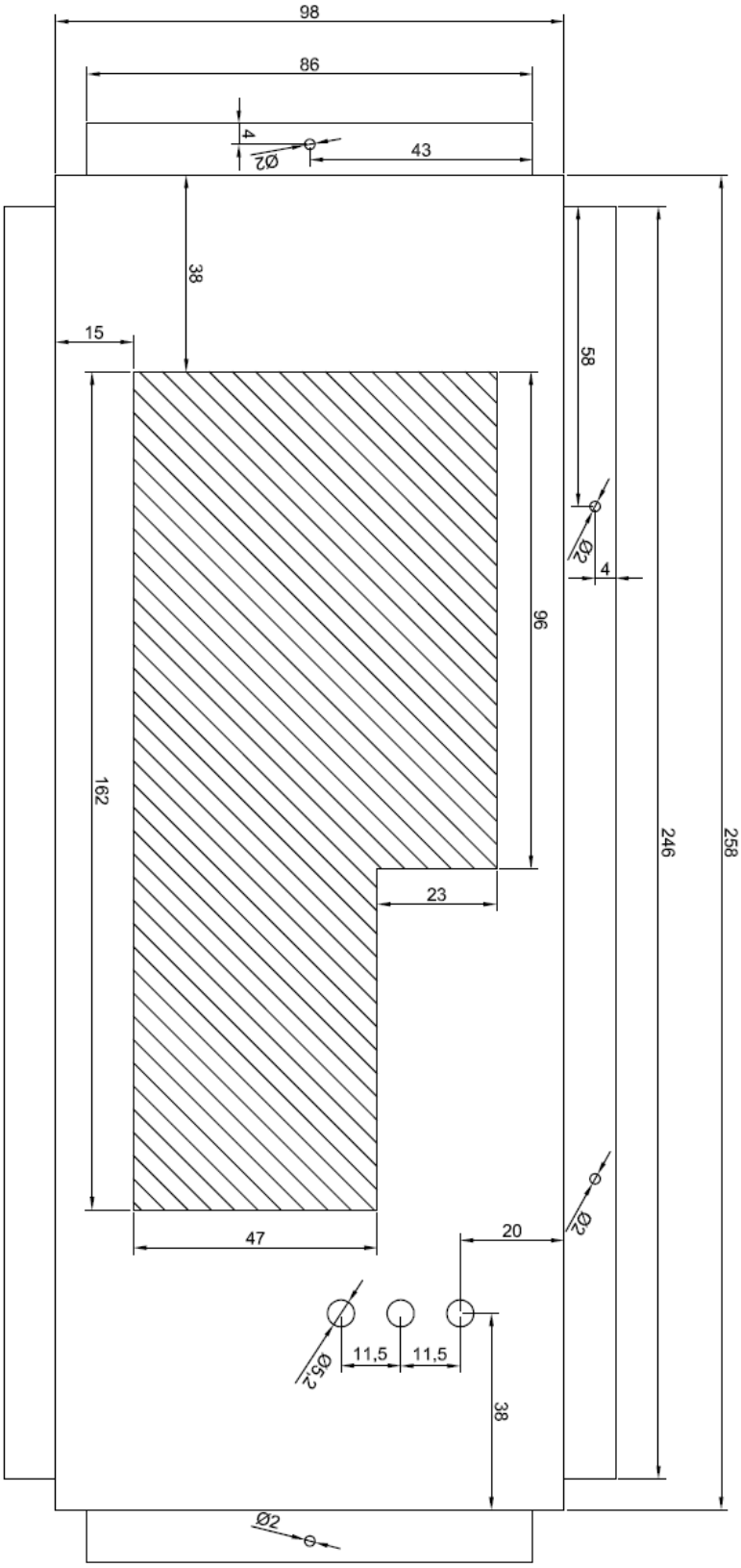


B Nákresy krabičky

B.1 Nákres části A krabičky

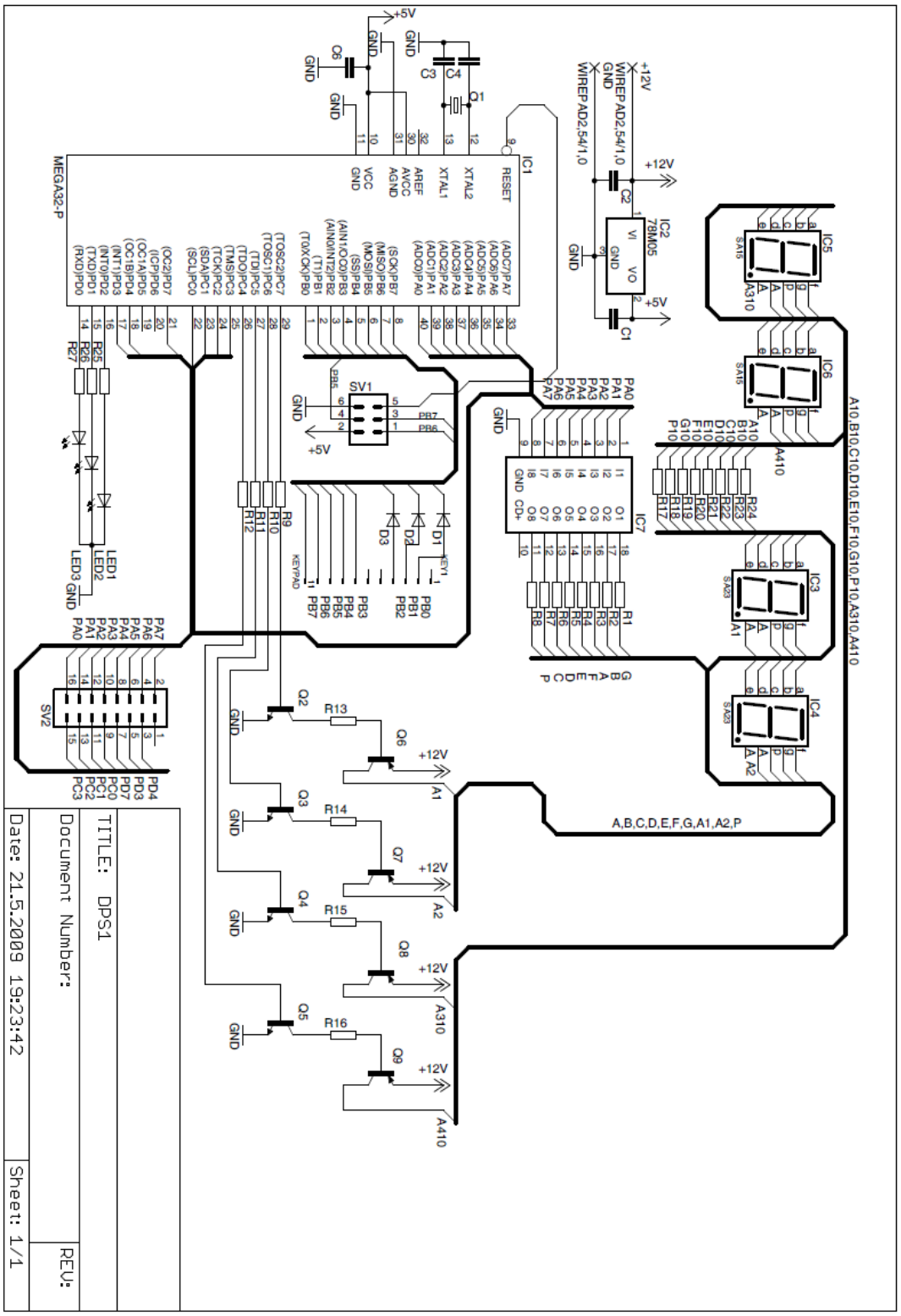


B.2 Nákres části B krabičky

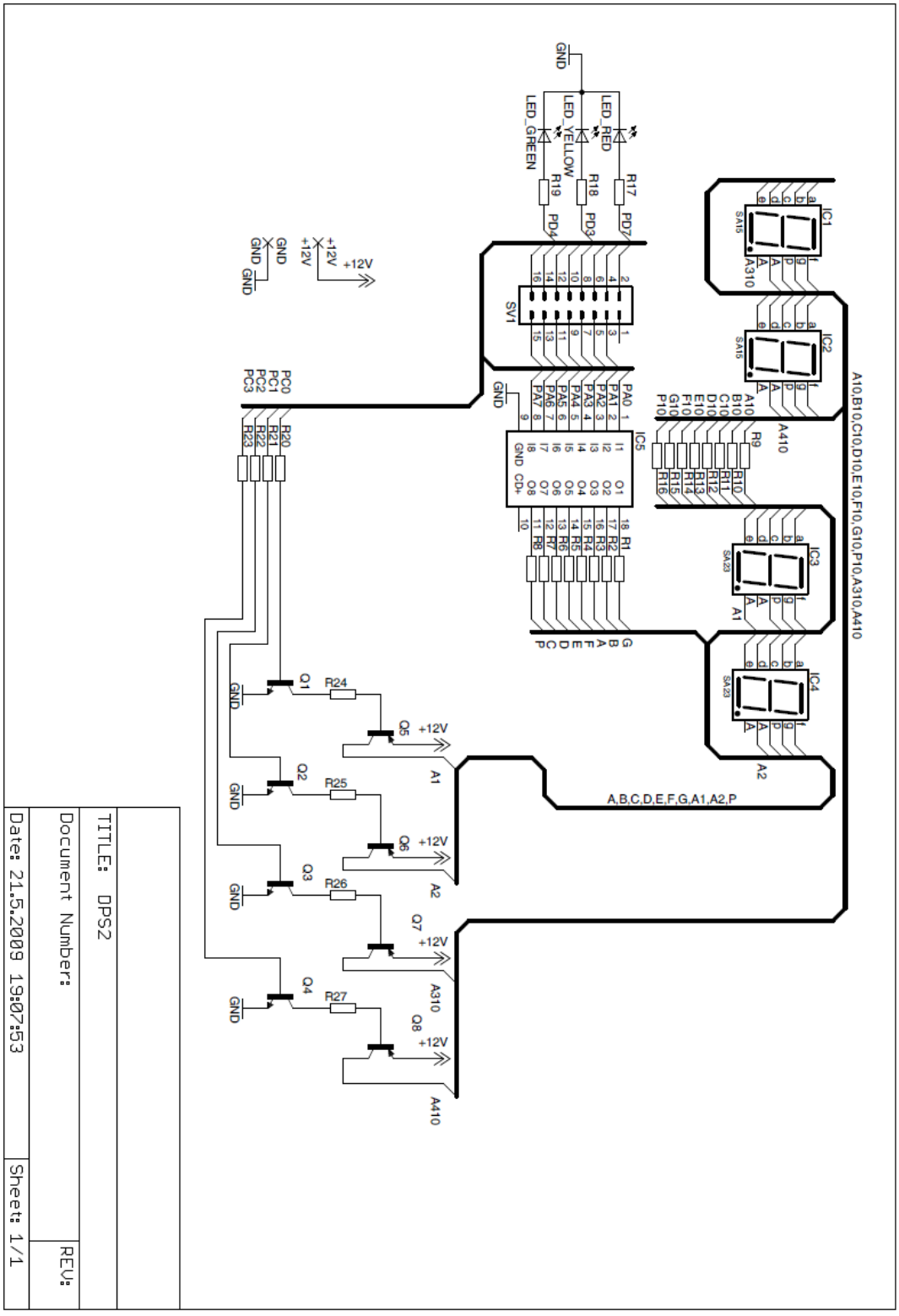


C Schémata zapojení

C.1 DPS1 (strana s ovládáním)

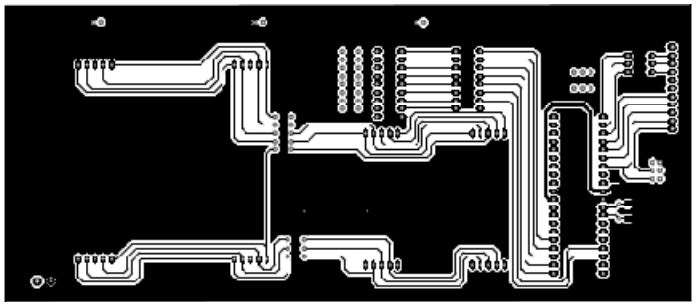


C.2 DPS2 (strana určená zkoušenému)

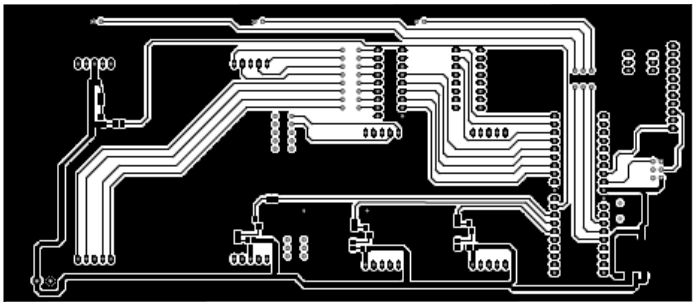


D Desky plošných spojů

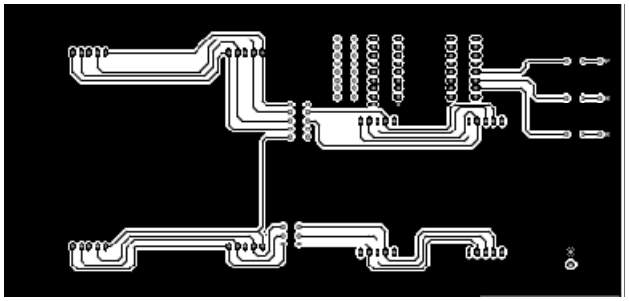
D.1 DPS1 (strana s ovládáním) - TOP



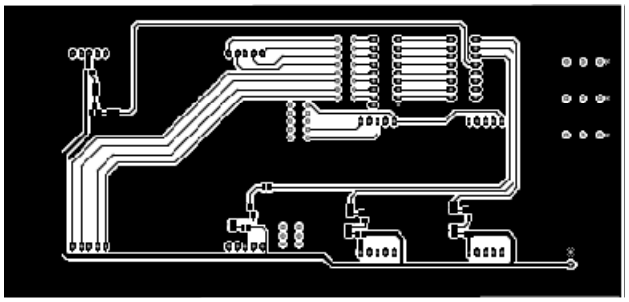
D.2 DPS1 (strana s ovládáním) – BOTTOM



D.3 DPS2 (strana určená zkoušenému) - TOP



D.4 DPS2 (strana určená zkoušenému) - BOTTOM



E Seznam součástek

DPS1 Označení	DPS2 Označení	Hodnota	Pouzdro	Popis
C1		100N	SMD - 0805	Keramický kondenzátor
C2		220N	SMD - 0805	Keramický kondenzátor
C3,C4		15pF	SMD - 0805	Keramický kondenzátor
C6		1M	SMD - 0805	Keramický kondenzátor
R1 - R7	R1-R7	75R	0204	Metalizovaný rezistor
R8	R8	180R	0204	Metalizovaný rezistor
R9-R12	R20-R23	10K	0204	Metalizovaný rezistor
R13-R16		2K2	0204	Metalizovaný rezistor
R17	R16	32R	0204	Metalizovaný rezistor
R18-R24	R9-R15	82R	0204	Metalizovaný rezistor
R25-R27	R17-R19	150R	0204	Metalizovaný rezistor
Q1		5,12 Mhz	HC49U-V	Krystal
Q2-Q5	Q1-Q4	BC817-16SMD	SOT23-BEC	Tranzistor NPN
Q6-Q9	Q5-Q8	BC807-16SMD	SOT23-BEC	Tranzistor PNP
IC1		Atmega32	DIL40	Mikrokontrolér
IC2		78M05	TO252	Stabilizátor napětí
IC3,IC4	IC3,IC4	SA23-12SRWA		Sedmisegment červený 56,9 mm
IC5,IC6	IC1,IC2	SA15-11EWA		Sedmisegment červený 38 mm
IC7	IC5	ULN2803A	DIL18	Darlington. pole
LED1-LED3		Červená	LED3MM	Červena LED 3mm
SV1			ML6L	Konektor 6 pinový
SV2	SV1		DIL16	Konektor do DPS pro plochý kabel
D1-D3		1N4148	DO35-7	Dioda
	LED_RED	Červená	LED5MM	Červena LED 5mm
	LED_YELLOW	Žlutá	LED5MM	Žlutá LED 5mm
	LED_GREEN	Zelená	LED5MM	Zelená LED 5mm

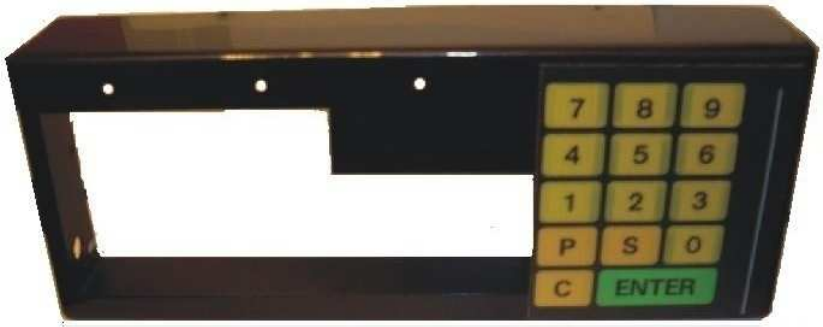
Ostatní součástky

Název	Počet [ks]	Popis
AWG28-16H	1	Plochý kabel 16 x licna 0,15 mm (délka cca 100mm)
DC-power central 2 mm	1	Napájecí konektor do panelu
Vrut	4	Vrut 2,2 x 6,5 mm (sešroubování krabičky)
Filtr displeje	2	Dodávaný ke krabičce KM-85
Zdroj EAP-12-12	1	Spínaný zdroj DC 12V, 1A

F Částí krabičky

F.1 Část B krabičky

F.2 Část A krabičky



F.3 Krabička s displejem a původním označením kláves

G Program

```
/* ------------------------------------------------------------------------
 * Title:    Casomira pro zaverecne zkousky
 * Author:   Jiri Mastik
 * Date:     6.4.2009
 * Hardware: ATmega32
 * Software: WinAVR
 ------------------------------------------------------------------------*/
#include <util/delay.h>
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>
#define FALSE     0;
#define TRUE      1;
#define blink_dig0          0x01;
#define blink_dig3          0x08;
#define blink_all           0x0F;
uint8_t  digit0=0, digit1=0, digit2=0, digit3=0, dispdig=0, keypad=16,sec=0,min=0;
uint8_t   countdown, nokey=0,blinking=0,min1=0,sec1=0,min2=0,sec2=0,min3=0,sec3=0,cmin=0,csec=0;
uint8_t   blinkled=0,select=0,run=0;
uint16_t  timer0ticks=0, ledcounter=0;
//                          0      1      2      3      4      5      6      7      8      9      10
uint8_t  segments[11] = {0x7E, 0x42, 0x37, 0x67, 0x4B, 0x6D, 0x7D, 0x46, 0x7F, 0x6F, 0x00};
void settime (uint8_t *psec,uint8_t *pmin,uint8_t eeadress1,uint8_t eeadress2,uint8_t eeadress3);
void anodes(uint8_t numblik,uint8_t anodes1,uint8_t anodes2,uint8_t digit);
void keypad_buttons (void);
void ioinit (void);
void settings (void);
void decode2(uint8_t seconds,uint8_t minutes);
void LEDS (void);
//-------------------------------------------------------------------------------------------
ISR (TIMER0_COMP_vect)
{
        if (countdown)
        {
                if (min+sec)
                {
                        if (timer0ticks==999)
                        {
                                timer0ticks=0;
                                if(sec==0)
                                {
                                        sec=59;
                                        if (min) min--;
                                        decode2(sec,min);
                                }
                                else
                                {
                                        sec--;
                                        decode2(sec,min);
                                }
                                if ((sec==0)&&(min==5)) PORTD=((PORTD&0x07)|0x08);
                                if ((min==2)&&(sec==0)) PORTD=((PORTD&0x07)|0x80);
                        }
                        else timer0ticks++;
```

```
                }
                else
                {
                        blinking=blink_all;
                        countdown=FALSE;
                        timer0ticks=501;
                }
        }
        if (blinking)
        {
                if (timer0ticks!=1000) timer0ticks++;
                else timer0ticks=0;
        }
                PORTA = 0x00;
                PORTC = 0x00;
                switch (dispdig)
                                {
                                        case 0:
                                                anodes(0x00,0x10,0x18,digit0);
                                                dispdig++;

                                                break;

                                        case 1:
                                                anodes(0x01,0x20,0x24,digit1);
                                                dispdig++;
                                                break;
                                        case 2:
                                                anodes(0x02,0x40,0x42,digit2);
                                                dispdig++;
                                                break;
                                        case 3:
                                                anodes(0x03,0x80,0x81,digit3);
                                                dispdig=0;
                                                break;
                                }

        keypad_buttons();
        if (select) LEDS();
}
//-------------------------------------------------------------------------------------------
void ioinit (void)
{
        DDRA = 0xFF;
        PORTA =0x00;
        DDRC = 0xFF;
        PORTC = 0x00;
        DDRB=0b00000111;
        PORTB =0b11111000;
        DDRD = 0xFF;
        PORTD =0x00;
}
//-------------------------------------------------------------------------------------------
void LEDS (void)
{
        if(ledcounter!=500) ledcounter++;
        else
```

```
        {
                ledcounter=0;
                if(blinkled==0) PORTD=(~PORTD)&(0x04);
                if(blinkled==1) PORTD=(~PORTD)&(0x02);
                if(blinkled==2) PORTD=(~PORTD)&(0x01);
        }

}
//-------------------------------------------------------------------------------------------
void anodes(uint8_t numblik,uint8_t anodes1,uint8_t anodes2,uint8_t digit)
{
        if (bit_is_set(blinking,numblik))
        {
                if (blinking==0x0F) PORTC=anodes2;
                else PORTC=anodes1;
                if(timer0ticks>500) PORTA= segments[digit];
        }
        else
        {
                if(!blinking) PORTC=anodes2;
                else PORTC=anodes1;
                PORTA=segments[digit];
        }
}
//-------------------------------------------------------------------------------------------
void decode2(uint8_t seconds,uint8_t minutes)
{
        digit3=minutes/10;
        digit2=minutes%10;
        digit1=seconds/10;
        digit0=seconds%10;
        if(!digit3) digit3=10;
}
//-------------------------------------------------------------------------------------------
void settime(uint8_t *psec,uint8_t *pmin,uint8_t eeadress1,uint8_t eeadress2,uint8_t eeadress3)
{
        uint8_t *pint,keycopy;

        if(digit3==10)
        {
                digit3=0;
                timer0ticks=501;
        }
        else timer0ticks=0;
        blinking=blink_dig3;
        pint= & digit3;
        nokey=FALSE;
        for(;;)
        {
                        if (keypad==16) nokey=TRUE;
                        if (((keypad==15)&&(nokey))&&(blinking))
                        {
                                if (bit_is_set(blinking,0))
                                {
                                        blinking=blink_dig3;
                                }
                                else  blinking=(blinking>>1);
```

```
            timer0ticks=0;
            if (pint!= & digit0) pint--;
            else pint= & digit3;
            nokey=FALSE;
}
if(((keypad==11)&&(blinking))&&(nokey))
{
            blinking=0;
            nokey=FALSE;
            *psec=(digit1*10)+digit0;
            *pmin=(digit3*10)+digit2;
            if(!digit3) digit3=10;
            eeprom_busy_wait();
            eeprom_write_byte((uint8_t*)eeadress3,0xff);
            eeprom_write_byte((uint8_t*)eeadress1,*psec);
            eeprom_write_byte((uint8_t*)eeadress2,*pmin);
            eeprom_write_byte((uint8_t*)eeadress3,0x00);
            break;
}
if ((keypad==10)&&(nokey))
{
            nokey=FALSE;
            if(digit3==10) digit3=0;
            *psec=(digit1*10)+digit0;
            *pmin=(digit3*10)+digit2;
            if((*psec)||(*pmin))
            {
                        if(!digit3) digit3=10;
                        eeprom_busy_wait();
                        eeprom_write_byte((uint8_t*)eeadress3,0xff);
                        eeprom_write_byte((uint8_t*)eeadress1,*psec);
                        eeprom_write_byte((uint8_t*)eeadress2,*pmin);
                        eeprom_write_byte((uint8_t*)eeadress3,0x00);
                        run=TRUE;
                        break;
            }
}
if(((keypad==11)&&(!blinking))&&(nokey))
{
            nokey=FALSE;
            timer0ticks=0;
            blinking=blink_dig3;
            pint= & digit3;
}
if (blinking)
{
            if ((keypad<=9)&&(nokey))
            {
                        keycopy=keypad;
                        timer0ticks=501;
                        if (keycopy!=16)
                        {
                                    if (pint!=(& digit1)) *pint=keycopy;
                                    else
                                    {
                                                if (keycopy<=5) *pint=keycopy;
                                    }
```

```
                                                nokey=FALSE;
                                        }
                                }
                        }
        }
}
//-------------------------------------------------------------------------------------------------------
void settings (void)
{
        static  uint8_t adress, *time=& min1,*time2=& sec1;
        decode2(*time2,*time);
        select=TRUE;
        ledcounter=500;
        for(;;)
        {
                if((keypad==15)&&(nokey))
                {
                        if(blinkled==2) blinkled=0;
                        else blinkled++;
                        ledcounter=500;
                        if(time!=& min3)
                        {
                                time+=2;
                                time2=time+1;
                        }
                        else
                        {
                                time=& min1;
                                time2=& sec1;
                        }
                        decode2(*time2,*time);
                        if(adress!=6) adress+=3;
                        else adress=0;
                        nokey=FALSE;
                }
                if((keypad==11)&&(nokey))
                {
                        select=FALSE;
                        if(blinkled==0)  PORTD=0x04;
                        if(blinkled==1)  PORTD=0x02;
                        if(blinkled==2)  PORTD=0x01;
                        settime(time2,time,adress,adress+1,adress+2);
                        ledcounter=500;
                        select=TRUE;
                        nokey=FALSE;
                }
                if(keypad==16) nokey=TRUE;
                if(((keypad==10)||(run))&&((*time)||(*time2)))
                {
                        select=FALSE;
                        if(blinkled==0)  PORTD=0x04;
                        if(blinkled==1)  PORTD=0x02;
                        if(blinkled==2)  PORTD=0x01;
                        sec=*time2;
                        min=*time;
                        cmin=min;
                        csec=sec;
```

```
                        run=FALSE;
                        nokey=FALSE;
                        blinking=0;
                        break;
                }
        }
        TCCR0&=0xF8;
        TCNT0= 0x00;
        timer0ticks=0;
        PORTC=0x00;
        PORTA=0x00;
        countdown=TRUE;
        _delay_ms(250);
        TCCR0= 0x0B;
        if (((min>2)&&(min<5))||((min==5)&&(sec==0))||((min==2)&&(sec>0))) PORTD|=0x08;
        if ((min>5)||((min==5)&&(sec>0))) PORTD|=0x10;
        if (((min==2)&&(sec==0))||(min<2)) PORTD|=0x80;
}
//-----------------------------------------------------------------------------------------------------------
void keypad_buttons (void)
{
        uint8_t key;
        static uint8_t key2, debounce=0;

                key = 16;
                PORTB =0b11111101;
                PORTB =0b11111101;
                PORTB =0b11111101;
                if (bit_is_clear(PINB, 3)) key=8;
                if (bit_is_clear(PINB, 4)) key=5;
                if (bit_is_clear(PINB, 5)) key=2;
                if (bit_is_clear(PINB, 6)) key=15;
                PORTB =0b11111011;
                PORTB =0b11111011;
                PORTB =0b11111011;
                if (bit_is_clear(PINB, 3)) key=9;
                if (bit_is_clear(PINB, 4)) key=6;
                if (bit_is_clear(PINB, 5)) key=3;
                if (bit_is_clear(PINB, 6)) key=0;
                if (bit_is_clear(PINB, 7)) key=10;
                PORTB =0b11111110;
                PORTB =0b11111110;
                PORTB =0b11111110;
                if (bit_is_clear(PINB, 3)) key=7;
                if (bit_is_clear(PINB, 4)) key=4;
                if (bit_is_clear(PINB, 5)) key=1;
                if (bit_is_clear(PINB, 6)) key=12;
                if (bit_is_clear(PINB, 7)) key=11;

                if ((debounce)&&(debounce!=12)) debounce++;
                if ((key!=16)&&(!debounce))
                {
                        key2=key;
                        debounce=1;
                }
                if ((key2==key)&&(debounce==12))
                        {
```

```
                keypad = key;
                }
        if ((debounce==12)&&(key!=key2))
        {
                debounce=0;
                keypad=16;
        }
}
//-------------------------------------------------------------------------------------------------------------
int main (void)
{
        ioinit();
        nokey=FALSE;
        countdown=FALSE;
        OCR0 = 79;
        eeprom_busy_wait();
        if (eeprom_read_byte((uint8_t*)2)==0x00)
        {
                sec1=eeprom_read_byte((uint8_t*)0);
                min1=eeprom_read_byte((uint8_t*)1);
        }
        if (eeprom_read_byte((uint8_t*)5)==0x00)
        {
                sec2=eeprom_read_byte((uint8_t*)3);
                min2=eeprom_read_byte((uint8_t*)4);
        }
        if (eeprom_read_byte((uint8_t*)8)==0x00)
        {
                sec3=eeprom_read_byte((uint8_t*)6);
                min3=eeprom_read_byte((uint8_t*)7);
        }
        sei();
        TIMSK|=_BV(OCIE0);
        TCCR0= 0x0B;
        select=TRUE;
        settings();
        for(;;)
        {
                if (keypad==16) nokey=TRUE;
                if ((keypad==10)&&(nokey)&&(!countdown)&&(sec+min))
                {
                        TCCR0&= 0xF8;
                        TCNT0= 0x00;
                        timer0ticks=0;
                        PORTC=0x00;
                        PORTA=0x00;
                        blinking=0;
                        timer0ticks=0;
                        countdown=TRUE;
                        _delay_ms(250);
                        TCCR0= 0x0B;
                if(((min>2)&&(min<5))||((min==5)&&(sec==0))||((min==2)&&(sec>0))) PORTD|=0x08;
                        if ((min>5)||((min==5)&&(sec>0))) PORTD|=0x10;
                        if (((min==2)&&(sec==0))||(min<2)) PORTD|=0x80;
                        nokey=FALSE;
                }
                if((keypad==10)&&(nokey)&&(countdown))
```

```
                {
                        countdown=FALSE;
                        PORTD=(PORTD&0x07);
                        digit0=10;
                        digit1=10;
                        digit2=10;
                        digit3=10;
                        _delay_ms(250);
                        decode2(sec,min);
                        nokey=FALSE;
                }
                if((keypad==12)&&(!countdown)&&(nokey))
                {
                        min=cmin;
                        sec=csec;
                        decode2(sec,min);
                        blinking=0;
                        timer0ticks=0;
                        PORTD=(PORTD&0x07);
                        nokey=FALSE;
                }
                if((blinking==0x0F)&&(keypad==10))
                {
                        blinking=0;
                        nokey=FALSE;
                }
                if ((keypad==11)&&(!(countdown)))
                {
                        blinking=0;
                        PORTD=(PORTD&0x07);
                        nokey=FALSE;
                        settings();
                }
        }
}
//-------------------------------------------------------------------------------------------------------------------
```