Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

Ústav elektroenergetiky

IGNITION1

Mikroprocesorové zapalování pro kogenerační jednotku

Technická dokumentace

Funkční vzorek

Ing. Petr Baxant, Ph.D.

Brno **2008**

Anotace

Elektronické zapalování pro kogenerační jednotku řízené mikrorocesorem řady 8051 s možností konfigurace předstihu a energie jiskry a buzením zapalovací lišty Felicie.

Tento materiál obsahuje výsledky výzkumu financovaného Ministerstvem školství, mládeže a tělovýchovy České republiky v rámci projektu č. MSM0021630516.

Klíčová slova: Zapalování, mikroprocesor

Annotation

Electronic ignition controled by microcontroler 8051 family with posibility of spark advance and energy regulation.

Keywords: Ignition, microcontroler

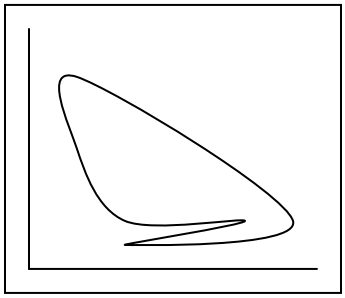
Verze: 2008-12-05

Obsah

Úvod

Zapalovací systém zážehového motoru je jedním z nejdůležitějších systémů ovlivňujících chod spalovacího motoru, jeho účinnost a výkon. Je proto velice důležité dodržet předepsané podmínky pro efektivní zapálení palivové směsi. Možnosti, jak tyto podmínky dodržet, jsou velice ohraničené a zařízení, které toto splňuje, musí mít možnost v plném rozsahu se těmto požadavkům přizpůsobit.

Využití spalovacího motoru v kogeneračních jednotkách umožňuje experimentovat s novými přístupy v jeho řízení a optimalizaci jeho chodu. Jako každý tepelný motor, je i reálný spalovací motor odlišný od svého teoretického ideálu. Jeho funkce a především účinnost budou hodně záviset na tom, jak moc se blíží k teoretickým předpokladům. Jelikož u tepelného stroje je nejpodstatnější uzavřený tepelný cyklus a jeho energetická účinnost, bude nás zajímat, jak můžeme přispět k jeho zdokonalení. Na obrázku 1. je patrný teoretický průběh tepelného cyklu u zážehového motoru. Energie tohoto cyklu je maximální při dodržení základních požadavků na kvalitu palivové směsi a jejím přesném míchání a kvalitním směšování.



Obr. 1 Tepelný oběhový cyklus znázorněný v P-V diagramu

Pokud nebudeme uvažovat kvalitu seřízení uzavíracích zařízení, netěsnosti motoru, mazání, chlazení apod., pak průběh hoření palivové směsi a tím celého tepelného cyklu, lze ovlivnit už v podstatě pouze dvěma faktory a to okamžikem zapálení a dobou trvání resp. energií zapalovací jiskry. Pomineme-li technickou realizaci zapalovacího systému (konstrukce zapalovacích svíček, indukčních vysokonapěťových transformátorů) můžeme požadované parametry měnit v zásadě pouze vhodným časováním zapínacího a vypínacího cyklu indukční cívky.

K tomu je bezpodmínečně nutné zajistit přímou vazbu polohy klikového hřídele motoru a zapalovacího systému časování s vyloučením mechanických vůlí, které jsou běžné u klasického systému s mechanickým rozdělovačem a rozvodným systémem. Je nutné si uvědomit, že stroj pracuje ve odlišných pracovních podmínkách s jiným palivem a jinými výkonovými poměry než je běžně používán v automobilu. Motor v kogenerační jednotce pracuje ve velice stabilních podmínkách a je tudíž nutné zajistit i stabilní funkce motoru.

Jak již bylo řečeno, správné zapálení palivové směsi je velice důležitým ukazatelem provozních vlastností motoru. Použití plynného paliva znamená jistou odlišnost od běžného systému a je třeba s tímto počítat. Při nedodržení správného načasování mohou nastat tyto situace:

- *k zapálení dojde příliš pozdě:* směs nestihne shořet všechna, čímž se nevyužije dodaná energie v palivu na expanzi plynu v pracovním cyklu motoru. Snižuje se výkon a klesá účinnost motoru.
- *k zapálení dojde příliš brzy:* směs začne expandovat ještě před horní úvratí, která odděluje fázi komprese a expanze. Rázově zvýšený tlak v kompresní části cyklu má za následek brždění motoru a tím snižování jeho výkonu. Energie vynaložená na brždění jde na úkor energie užitečné, o kterou je motor ochuzen v pracovním cyklu. V tomto případě se navíc zvyšuje kompresní tlak působící na píst, ojnici a všechna ložiska, čímž se zvyšuje jejich opotřebení a klesá životnost kritických částí motoru.

Podobný účinek ovšem menšího dopadu má i energie jiskry. Elektrická energie vytvářející oblouk (jiskru), který zapaluje směs, se dá opět ovlivnit vhodným časováním. Nízká energie většinou znamená malý počáteční impulz a pomalé hoření směsi, naopak velká energie znamená silný popudový impulz, který může vyvolat prudké hoření směsi a nevhodné rozložení pracovního tlaku na píst v kompresní části motorového cyklu. Oba provozní parametry se navzájem ovlivňují a jejich vhodné sladění může zajistit optimální a klidný chod stroje s maximálním využitím paliva a minimálním opotřebením motoru.

Je zřejmé, že zapalovací systém nelze zcela oddělit od okolí a že výsledné chování motoru bude závislé i na ostatních podmínkách (vlastnosti palivové směsi, vstupní teplota směsi, kvalita plynného paliva, opotřebení motoru atd.). Ideální systém by se měl na tyto parametry adaptovat a přizpůsobit své chování aktuálním podmínkám. To však vyžaduje velice složitý systém s komplikovaným řízením, které lze realizovat pouze s využitím elektronických obvodů s mikroprocesorovým řízením a systémem adaptivní regulace.

Obecný popis a funkce

Elektronické zapalování je navrženo pro připojení zapalovacího systému motoru Škoda Felicie, které je popsáno dále. Řídící obvod umožňuje budit dvoufázový koncový stupeň impulzy s úrovní 0 a 5V s jejich plně programovatelným časováním.

Obvod využívá jednočipového mikrořadiče AT89C2051 s taktovacím kmitočtem 24MHz, který zajišťuje všechny výkonné funkce obvodu zapalování. Chod mikroprocesoru hlídá nezávislý Watchdog obvod, který v případě nekorektního chodu programu zajistí restartování mikroprocesoru a znovu naběhnutí systému.

Proměnné, které lze uživatelsky měnit, se ukládají do elektricky mazatelné sériové paměti EEPROM, která zajišťuje uložení proměnných i v případě vypnutého napájení.

Vstupní informace obvod získává z čidla, které je popsáno dále. Uživatelsky je možné některé proměnné nastavit pomocí třítlačítkové klávesnice připojené ke dvěma vstupům mikroprocesoru. Výstupní signál je oddělen od procesoru výstupními budícími tranzistory, které zajišťují dostatečné buzení koncových budičů indukčních cívek.

Provozní informace a hlášení jsou předávána v sérové podobě s časováním hodinovým signálem. Data je možné zobrazit na odpovídajícím displeji

Technické parametry

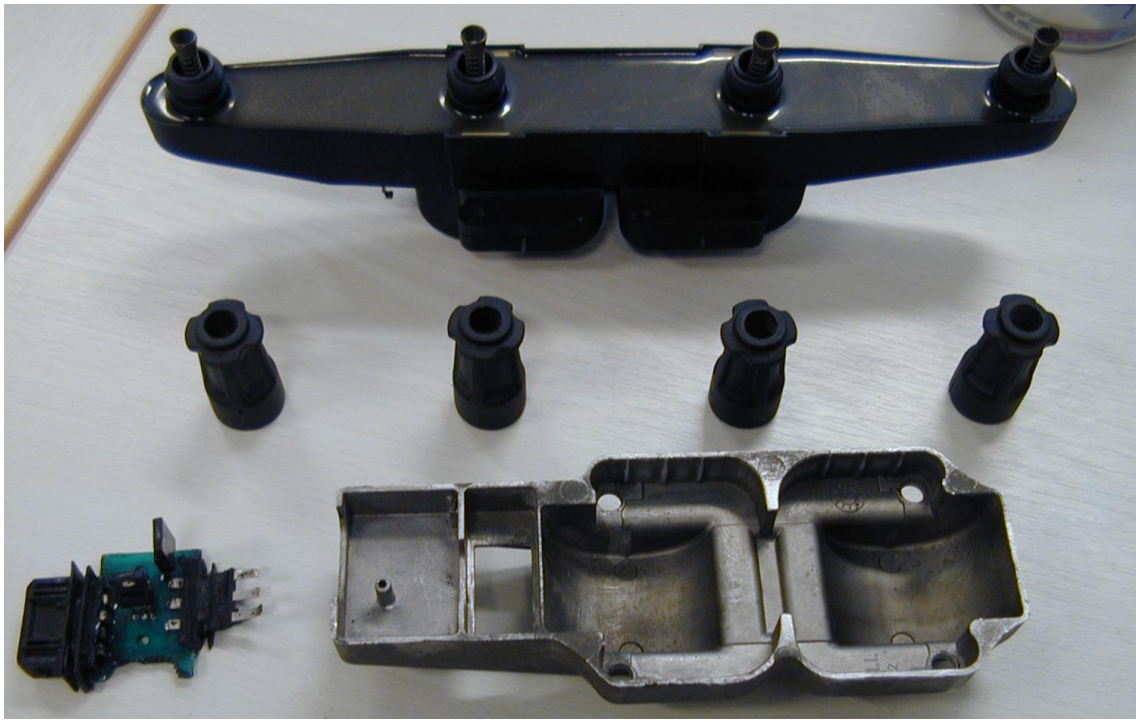
Napájecí napětí	5V, 12V
Procesor	Atmel AT89C2051-24MHz
Paměť	93C24 - EEPROM 8x128b
Výstupní pulz	4,5 - 5V, s proměnnou délkou a fází
Nastavení předstihu	-30 - +20 stupňů
Nastavení délky pulzu	0 - 4 ms
Umístění čidla	90 - 180 stupňů před horní úvratí 1. válce
Vstupní impulz	sestupná hrana 5 → 0V z čidla otáček
Maximální otáčky	nastavitelné, max. 5000 min^{-1}
Minimální otáčky	nastavitelné, min. 1000 min^{-1}

Popis zapalovacího systému motoru Škoda Felicie

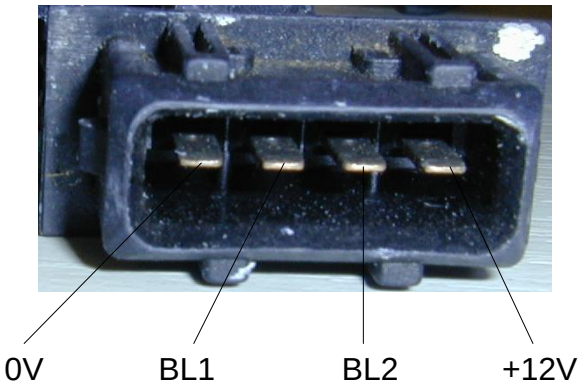
Zapalovací souprava (lišta) je konstruována jako jeden kus, který se připojuje přímo na zapalovací svíčky a připojení napájení a buzení je realizováno přes jediný čtyřpinový konektor. Uvnitř lišty je dvojitá indukční vysokonapěťová cívka s dvěmi oddělenými sekundárními vinutími a dvěmi primárními vinutími se společným vývodem. Buzení cívek zajišťují speciální koncové budiče VB921ZVFI v pouzdře TO-220, určené pro tento druh obvodů. Jedná se v podstatě o integrovaný obvod, zajišťující omezení primárního napětí i proudu s jednoduchým buzením napěťovou úrovní 4,5 - 5,5V. Bližší informace a technické parametry jsou patrné z katalogových listů v příloze.

Dvojice těchto budičů je umístěna na desce plošného spoje společně se vstupním a výstupním konektorem. Deska je vložena do hliníkového pláště a zalita krycí a těsnící hmotou LUKOPREN. Chlazení budičů zajišťuje hliníkový kryt cívek kontaktující s pouzdrem přes tepelně vodivou vazelínu.

Vývody cívky jsou bodově svařeny s výstupním páskovým konektorem a spoj je zakryt a zalit izolační hmotou. Uspořádání vývodů vstupního konektoru je na obrázku.



Obr. 3 Zapalovací lišta

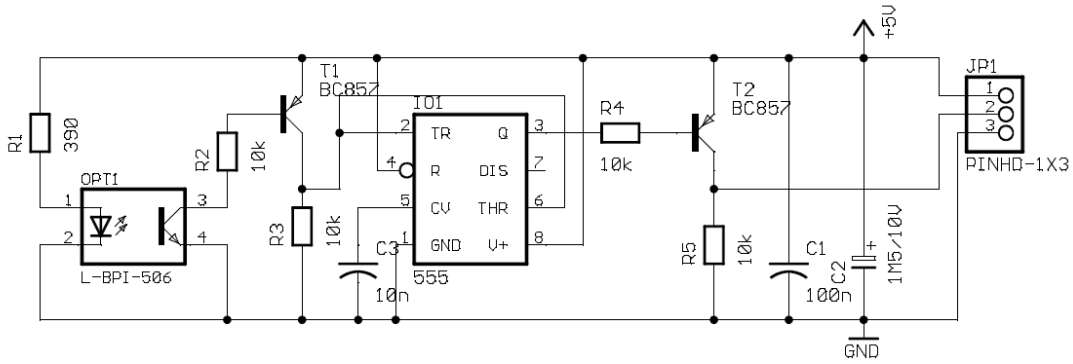


Obr. 4 Zapojení vývodů vstupního konektoru

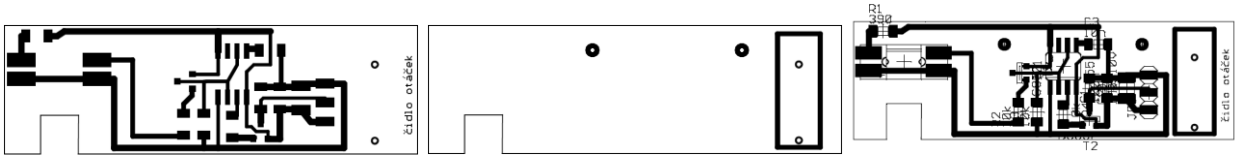
Čidlo otáček a polohy

Funkce čidla otáček a polohy je integrována do jediného obvodu. Jedná se o poměrně jednoduchý obvod s optickou snímací hlavou, tedy bez jakékoliv mechanické vazby. Optická závora je schopna detekovat i velice úzké a strmé hrany, což zabezpečuje velkou přesnost nastavení. Poloha čidla přitom nemusí být přesně definována, konečná poloha se ukládá do paměti procesoru a je v podstatě nastavitelná v širokém rozsahu úhlů.

Vlastní obvod sestává z několika jednoduchých částí. V první řadě je to optozávora i infračerveným senzorem. Závora detekuje předměty procházející pracovní štěrbinou. Signál je dále zesilován a tvarován Schmidtovým klopným obvodem. Poté je signál invertován. Výstup je aktivní v úrovni L, v době kdy je detekován předmět. Klidová úroveň výstupního signálu je H. Čidlo je napájeno 5V z hlavního obvodu zapalování.



Obr. 5 Schéma zapojení optického čidla polohy



Obr. 6 Plošný spoj čidla polohy, horní a spodní strana a schéma osazení

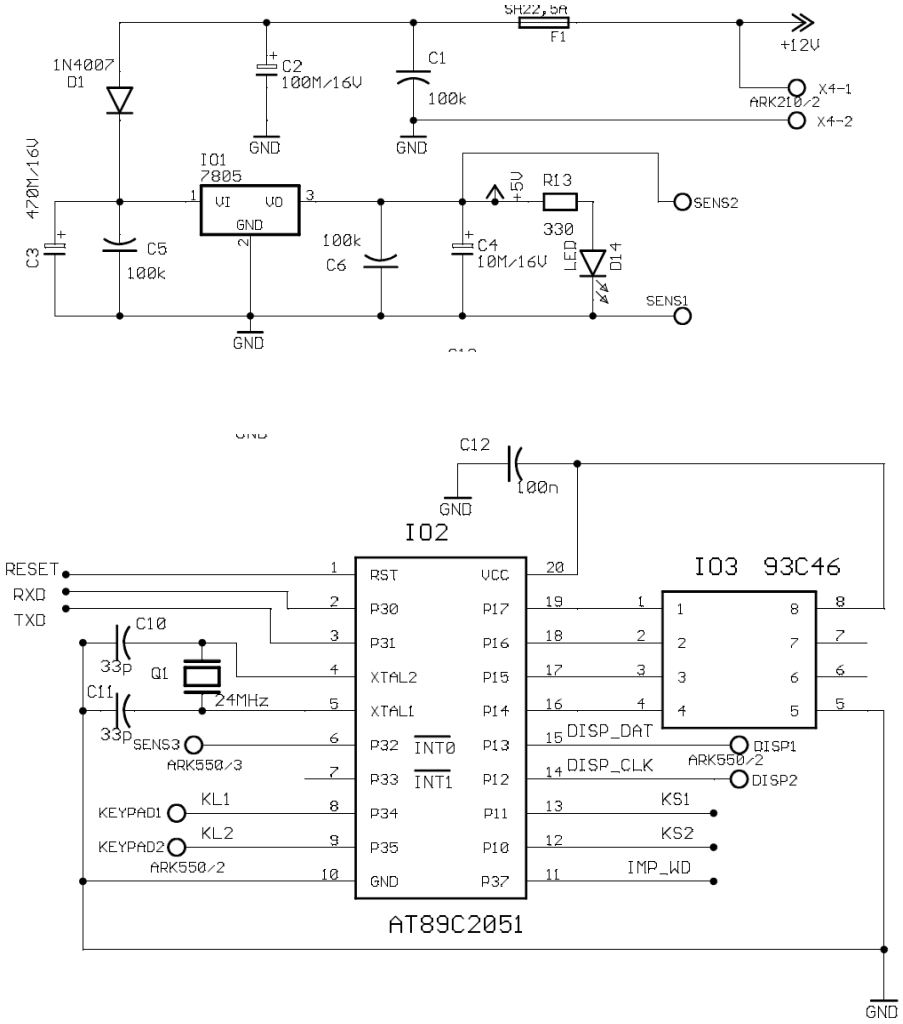
Obvod elektronického zapalování

Obvod je koncipován prakticky jako jednoduchý mikropočítač, který má vlastní napájení 5V, hlídací obvod WATCHDOG a mikroprocesor. Mikroprocesor vyhodnocuje příchozí impulsy od čidla polohy v rámci obsluhy přerušení INT0 a provádí potřebný výpočet časování pro buzení

koncového stupně a tím pádem i pro zapalovací cívky. Vlastní inteligence obvodu je tedy v programu mikroprocesoru.

Komunikaci s programem zajišťuje externí klávesnice a displej integrovaná do samostatného zařízení.

K procesoru je připojena navíc elektricky mazatelná paměť EEPROM, do které je možné uložit uživatelsky nastavené hodnoty, které zde zůstanou i po vypnutí napájení a budou použity při dalším startu. Signály RxD a TxD je možné využít pro nadřazenou komunikaci, např. s osobním počítačem přes linku RS232 přes příslušný konvertor na úrovně signálů TTL logiky.

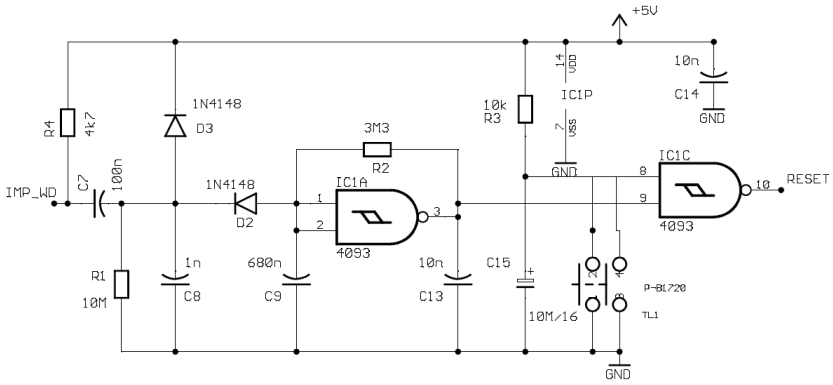


Obr. 7 Obvody napájení a procesorové části zapalování

Obvod WATCHDOG

Obvod WATCHDOG je samoběžný multivibrátor s možností resetování. Jedná se o obvod, který musí být vlastním programem procesoru neustále obsluhován a tím pádem deaktivován, resetován. Počet obsluh je asi 10 za sekundu. Pokud se stane, že program nebo sám procesor

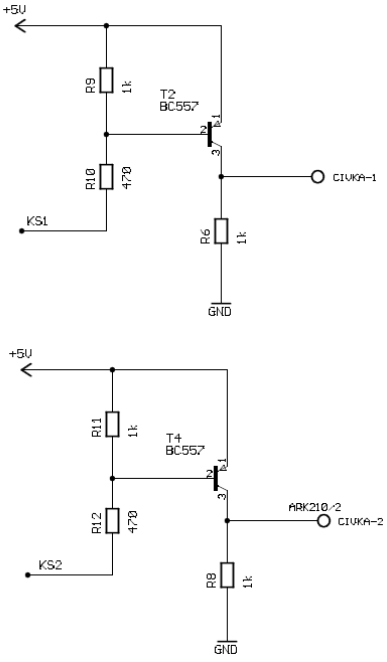
nějakým způsobem selže, obvod watchdogu provede automatický RESET procesoru, čímž jej nastavý zpět na začátek a zajistí tak opět pravidelnou obsluhu watchdog obvodu. Obvod je integrován na stejné desce, jako celé zapalování.



Obr. 8 Obvod WATCHDOG

Koncové stupně

Koncové stupně pro buzení zapalovací lišty jsou velmi jednoduché, tvoří prakticky jen posilovací tranzistory, které jsou schopny vybudit koncové tranzistory v zapalovací liště.

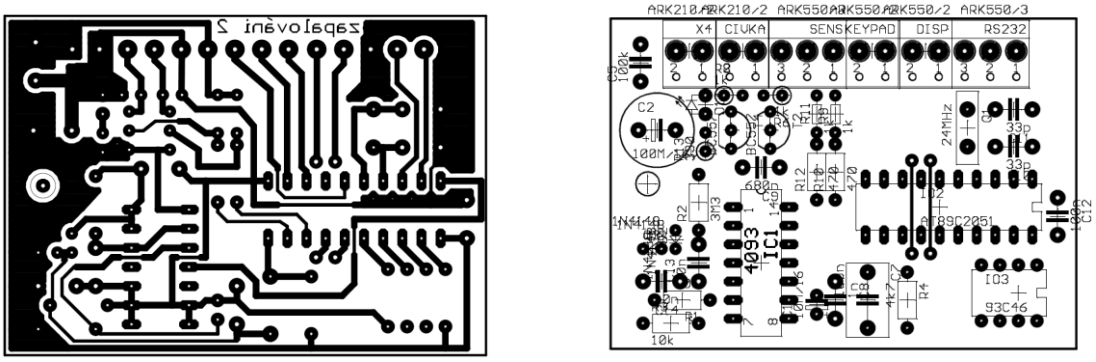


Obr. 9 Koncový stupeň obvodu zapalování

Vlastní výroba vzorku

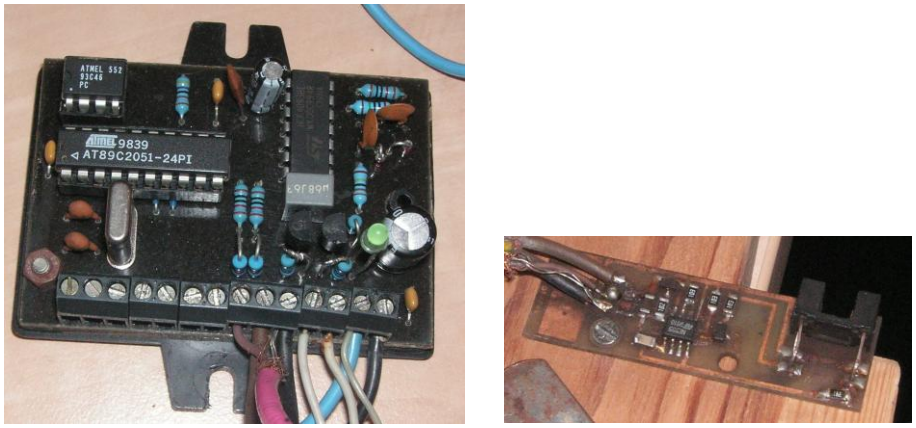
Elektronické zapalování sestává ze třech, částí: čidlo otáček (polohy), hlavního obvodu a univerzální klávesnice, displeje. Zde popíšeme konstrukci hlavní elektronické části, neboť vlastní čidlo otáček je velmi jednoduché a není třeba jej blíže popisovat a jeho konstrukce je zřejmá z popisu v kapitole dříve.

Plošný spoj je navržen jako jednostranný s tím, že je třeba použít dvou propojek z horní strany součástek. Plošný spoj je vyobrazen na následujícím obrázku včetně schématu osazení.



Obr. 10 Deska plošných spojů a schéma osazení

Osazená deska je opatřena svorkovnicí pro připojení dalších obvodů. Výslednou podobu zapojeného obvodu ukazují následující obrázky.



Obr. 11 Foto dokumentace (obvod zapalování vlevo, optické čidlo vpravo)

Obslužný program – softwarové vybavení

Jádro funkčnosti obvodu zapalování obstarává program mikroprocesoru, který je koncipován jako jednoduchý automat, který prochází dílčí moduly vykonávající určitou činnost. Součástí hlavního běhu je tedy zejména:

- obsluha klávesnice a displeje
- výpočet časování spínání a vypínání jednotlivých tranzistorů
- obsluha watchdog obvodu
- obsluha přerušení pro záznam pulsů od čidla otáček
- výpočet systémových proměnných, které se používají pro řízení
- obsluha elektricky mazatelné paměti (ukládání a načítání dat)

Program je zdokumentován v podobě komentářů přímo ve zdrojovém kódu. Program je napsán v assembleru pro procesory řady 51 a zkompilován pro daný procesor Atmel 89C2051. Výpis programu je v příloze.

Příloha 1 - Výpis řídícího programu

```
; ------------------------------------------------------------------------------
; ZAPAL.ASM     v.1.2
;
; program pro procesor AT80C2051 obvodu zapalování
; optimalizovano pro 2kB FLASH EPROM, jsou vypusteny nektere funkce puvodniho programu
; zkracene menu na nejdulezitejsi funkce.
; P. Baxant 24.10.2008
; ------------------------------------------------------------------------------

; ************************ popis funkce programu ***********************************************

; Výstupy KS1 a KS2 ovládají dvojci koncových tranzistorů v PNP provedení. Ty potom
; budí vlastní stupně zapalovací lišty. Výstupy jsou tedy aktivní v L.
; Každý tranzistor ovládá jednu cívku, vždy střídavě.
; Cívka L1 je připojena na válce 1 a 4
; Cívka L2 je připojena na válce 2 a 3
;
; Další dva výstupy (DIS_DAT,DIS_CLK) slouží k plnění čtyřmístného displeje sériovými daty.
; Dva vstupy (UP,DOWN)cslouží k připojení jednoduché klávesnice se třemi tlačítky.
; Čtyři výstupy (DO,DI,SK,CS) jsou použity pro komunikaci s pamětí EEPROM.
; Vstup senzor (SENS) slouží k připojení čidla otáček. Vyvolává přerušení INT0 a je aktivní v L
; Vstupy pro sériovou komunikaci (RX, TX) jsou zatím nevyužity. Je však možné je použít pro rozšíření.
;
; Výstup IMP_WD slouží k obsluze vnějšího watchdog obvodu, který musí být programem obsluhován
; v intervalu méně než asi 0.5 s. Neobsloužení znamená reset programu.
;
; Program využívá oba vnitřní čitače-časovače. Oba čitače-časovače pracují v módu
; 16-ti bitového čitače (mód 1). Čitač 0 je použit jako hlavní měřicí člen. Čítá s přesností
; krystalu a krokem 0.5us při 24MHz hodinového kmitočtu. Při 3000 ot/min je tak schopen detekovat
; úhel s přesností 0.54 úhlových vteřin, což je 0,009 stupňů.
; Druhý čítač 1 je použit na generování výstupních impulsů pro koncové stupně. Zároveň generuje
; impulsy pro hlídací obvod watchdog.
;
; Program obsahuje rutinu obsluhy sériové paměti FLASH EEPROM, kde se ukládají uživatelsky
; volitelné konstanty, které se po restartu opět nahrávají do operační paměti.
;
; ********************************************************************************************
;
;       Definice konstant
;
LTIME           SET     100d            ;doba nutna pro dlouhe stisknuti 2s
STIME           SET     50d             ;doba nutna pro kratsi stisknuti 1s
CODE            SET     10111101b       ;overovaci kod
CHARGE           SET     16d              ;cas nabijeni v 1/10 ms
A_S_L            SET     58h              ;(uhel umisteni cidla - 100)*100 (180 stupnu)
A_S_H            SET     1Bh
A_P_L            SET     0DCh             ;uhel predstihu (15 stupnu)
A_P_H            SET     05h
POZI_MAX SET     05d             ;maximalni pocet polozek v menu 1 + 1
ASMINL           SET     0D0h
ASMINH           SET     07h
ASMAXL           SET     4Ch
ASMAXH           SET     1Dh
APMINL           SET     00h
APMINH           SET     00h
APMAXL           SET     0B8h
APMAXH           SET     0Bh
TCHMAX           SET     35d
TCHMIN           SET     2d


;
;
;       Definice vstupů a výstupů
;
;Návěští instr.   operand          poznámka

RX              BIT     P3.0            ;vysílač
TX              BIT     P3.1            ;přijímač
SENS            BIT     P3.2            ;vstup od čidla otáček
UP              BIT     P3.4            ;klávesa Up
DOWN            BIT     P3.5            ;klávesa Down
IMP_WD          BIT     P3.7            ;impulsy pro watchdog

KS1             BIT     P1.1            ;koncový stupeň 1
KS2             BIT     P1.0            ;koncový stupeň 2
```

```
DIS_CLK          BIT     P1.2            ;hodiny display
DIS_DAT          BIT     P1.3            ;data display
DO              BIT     P1.4            ;serial data output
DI              BIT     P1.5            ;serial data input
SK              BIT     P1.6            ;shift clock
CS              BIT     P1.7            ;93C46 chip select

;       Definice paměti (bitova oblast zacina na adrese 20H)

NEW_DATA BIT     00H             ;nová data v čitači otáček
NEW_PRUM BIT     01H             ;nový průměr
NEW_OT           BIT     02H             ;nové otáčky (ot/min)

K_UP            BIT     03H             ;priznak stisknute klavesy UP
K_DOWN          BIT     04H             ;priznak stisknute klavesy DOWN
K_MODE          BIT     05H             ;priznak stiskleho MODE tlacitka
KPLT            BIT     06H             ;priznak urcujici dlouhe stisknuti tlacitka
KPST            BIT     07H             ;priznak dele stisknuteho tlacitka
K_GO_UP         BIT     08H             ;priznak povoleni tlacitka
K_USE           BIT     09H             ;priznak pouziti stiskleho tlacitka
K_PRESS         BIT     0AH             ;priznak stisku jakekoliv klavesy
K_PRESSP        BIT     0BH             ;priznak stisku klavesy v predchozim cyklu
SW_END           BIT     0CH             ;priznak ukonceni casoveho useku
OT_OK            BIT     0DH             ;priznak spravnych otacek
OT_HI            BIT     0EH             ;priznak vysokych otacek
OT_LO            BIT     0FH             ;priznak nizkych otacek
K_UP_T           BIT     10H             ;docasny priznak stiskle klavesy
K_DOWN_T BIT     11H
K_MODE_T BIT     12H
KPOK             BIT     13H             ;priznak potvrzeni spravne stiskle klavesy
DECIM1           BIT     14H
DECIM2           BIT     15H
DECIM3           BIT     16H
OT_STOP          BIT     17H
WRITE_OK BIT     18H
STO_OK           BIT     19H             ;zmena dat
STORE            BIT     1AH             ;priznak ukladani
STO_STRT BIT     1BH             ;zacatek nahravani
STO_END          BIT     1CH             ;konec nahravani
RECALL           BIT     1DH             ;priznak nahravani
RCL_OK           BIT     1EH             ;priznak spravneho natazeni dat z EEPROM
CODE             SET     10011011b       ;kodovy poznavaci klic

;       Deklarace osmibitových proměnných

                ORG     29H
N_L:            DS      1               ;nejnižší byte počítadla periody
N_H:            DS      1               ;vyšší byte počítadla periody
N_C:            DS      1               ;nejvyšší byte počítadla periody

; tři byty při frekvenci procesoru 24MHz (0.5 us čitač) pojmou 2(24)=16777216 cyklů
; tj. min. 8 sekund.

N_P_0:          DS      1               ;pamet pro soucet prumerne hodnoty periody
N_P_1:          DS      1               ;
N_P_2:          DS      1               ;
N_P_3:          DS      1               ;
N_P_L:          DS      1               ;pamet pro skutecnou prumernou hodnotu periody
N_P_H:          DS      1               ;
N_P_C:          DS      1               ;
OT_L:           DS      1               ;
OT_H:           DS      1               ;otacky vypoctene z prumeru
TMP_N_C: DS      1               ;
POC_IMP: DS      1               ;pocitadlo impulsu (otacek)

POCSEG:         DS      1               ;pocitadlo segmentu
T_PRESS: DS      1               ;pocitadlo doby stiskle klavesy
MENU:           DS      1               ;pamet menu
POZ_MENU:        DS      1               ;pozice v aktivnim menu
PAUSE:          DS      1               ;pocitadlo pauzy
DISPLAY: DS      1               ;byte displaye 0-199 zobrazuje cislo, 200-255 spec znaky

TMP_0:          DS      1               ;docasna pamet pro vypocty deleni a nasobeni
TMP_1:          DS      1
TMP_2:          DS      1
TMP_3:          DS      1
OP_0:           DS      1               ;pamet pro operandy
OP_1:           DS      1
OP_2:           DS      1
OP_3:           DS      1
```

```
D_0:            DS      1               ;pamet pro delitel
D_1:            DS      1
D_2:            DS      1
MUL_16_L:       DS      1               ;pamet pro nasobitel
MUL_16_H:       DS      1
T_ALFA_L:       DS      1               ;pamet pro uhel ALFA
T_ALFA_H:       DS      1
T_WAIT_L:       DS      1               ;pamet pro cekani v useku 2
T_WAIT_H:       DS      1
T_CHAR_L:       DS      1               ;pamet pro nabijeci cas v useku 1 a 3
T_CHAR_H:       DS      1
DATEPROM:       DS      1               ;pamet pro prenos dat z pameti EEPROM
ADREPROM:       DS      1               ;adresa pameti EEPROM
USEK:           DS      1               ;pamet pro casovy usek
DIS_TIME:       DS      1               ;pamet pro cas zobrazeni ruznych zprav
ADRRAM:         DS      1               ;ukazatel pameti RAM

DIS_0           EQU     10H             ;pozice pro ukladani kodu zobrazovaných znaku displaye
DIS_1           EQU     11H
DIS_2           EQU     12H
DIS_3           EQU     13H

ALFA_S_L EQU    18H             ;pamet pro uhel senzoru
ALFA_S_H EQU    19H
ALFA_P_L EQU    1AH             ;pamet pro uhel predstihu
ALFA_P_H EQU    1BH
T_CH            EQU     1CH             ;pamet pro dobu nabijeni civky
CODE_KEY EQU    1DH             ;kodovy klic pro overovani integrity pameti

;-------------------------------------------------------------------------------
;vlastni program

                ORG     00H
                jmp     RESET           ;skok na zacatek programu

                ORG     03H
                jmp     INT_0
                reti                    ;obsluha preruseni pri detekci impulsu cidla

                ORG     0BH
                jmp     T0_OF           ;skok na obsluhu preruseni preteceni casovace 0

                ORG     13H
                reti

                ORG     1BH
                jmp     T1_OF
                reti                    ;skok na obsluhu preruseni preteceni casovace 1

                ORG     23H
                reti                    ;skok na obsluhu seriove linky

;rutiny obsluh preruseni
;-------------------------------------------------------------------------------
                ORG     25H             ;rutina obsluhy preruseni pri preteceni casovace 0
T0_OF:
                push    ACC
                push    PSW
                clr     C
                mov     A,TMP_N_C
                add     A,#1d
                subb    A,#0d
                mov     TMP_N_C,A
                pop     PSW
                pop     ACC
                reti

;-------------------------------------------------------------------------------
                                        ;rutina obsluhy preruseni pri preteceni casovace 1
T1_OF:
                clr     TR1             ;zastav casovac
                setb    NEW_DATA ;informuj program o pripadnych novych datech o case
                jnb     OT_OK,NO_SW     ;preskoc, pokud nejsou otacky v poradku, casuje se pouze interne
                call    SWITCH          ;proved prislusne prepnuti tranzistoru a posun usek
                clr     NEW_DATA ;a zrus informaci o novych datech, slo pouze o prepnuti
NO_SW:
                reti

;-------------------------------------------------------------------------------
INT_0:
```

```
                clr     TR0             ;zastav citac
                mov     N_L,TL0
                mov     N_H,TH0
                mov     N_C,TMP_N_C     ;ulož délku periody
                mov     TMP_N_C,#0
                mov     TH0,#0
                mov     TL0,#11d ;znovu inicializuj čitač
                setb    TR0             ;a spusť jej znova
                setb    NEW_DATA ;informuj program o nových datech
                reti


; ***********************    hlavni program po RESETu    *********************************

; cast inicializace vnitrnich promennych


RESET:
                mov     SP,#60H         ;nastaveni stackpointeru
                mov     TMOD,#00010001b ;nastaveni casovacu do modu1
                mov     TCON,#00000101b ;nastaveni preruseni na detekci sestupne hrany
                mov     IE,#10001010b   ;povol preruseni timery
                setb    EX0               ;povol preruseni od INT0
                setb    PX0               ;nastav nejvyssi prioritu tomuto preruseni
                setb    TR0               ;spust casovac 0
                clr     TR1               ;casovac 1 zastaven
                setb    KS1               ;vypni oba koncove stupne
                setb    KS2
                clr     OT_OK             ;otacky zatim povazuj za neplatne
                setb    OT_STOP           ;otacky stop


; prednastaveni velicin a pameti

                mov     POC_IMP,#128d    ;nastaveni poctu impulsu ze kterych se pocita prumer
                mov     T_PRESS,#0d      ;nuluj dobu stisku klavesy



                mov     MENU,#1d ;nastav menu 1
                mov     POZ_MENU,#0      ;nastav vychozí pozici v menu
                mov     USEK,#0
                mov     DIS_TIME,#50d
                mov     CODE_KEY,#CODE

                call    READ
                clr     C
                mov     A,CODE_KEY
                subb    A,#CODE
                jz      LOAD_OK
                call    DEFAULT
LOAD_OK:
                clr     DECIM1
                clr     DECIM2
                clr     DECIM3
                clr     STORE
                clr     RECALL
                clr     RCL_OK
                clr     STO_STRT
                clr     STO_END
                clr     STO_OK

                jmp     NECEKEJ

;-------------------------------------------------------------------------------

; **********************     vykonna cast programu    *********************************

; po prijmuti impulsu je nutne stanovit delku periody, vypocitat casove useky pro nabijeni civky
; obslouzit display a klavesnici.


CEKEJ:
                jnb     NEW_DATA,CEKEJ   ;cekej ve smycce na impuls a nova data o otackach

                clr     NEW_DATA
                cpl     IMP_WD           ;obsluz watchdog

                call    SP_TEST          ;test otacek
```

```
                jb      OT_OK,MAIN1      ;pokud jsou otacky spravne pokracuj
NECEKEJ:
                call    OFFMODE          ;jinak prejdi do vypnuteho stavu
                jmp     MAIN2
MAIN1:
                call    C_PRUM_P ;spocitej prumenou periodu popr. otacky
                call    CALC_TIM ;spocitej casove useky pro casovac 1
                call    USEK0            ;nastaveni casovace pro prvni usek
MAIN2:
                call    READ_KEY ;obsluha klavesnice
                call    MENU_SEL ;obsluha menu
                call    NADISP           ;zobraz pripravena data

                jnb     STORE,MAIN3
                call    WRITE
MAIN3:
                jnb     RECALL,MAIN4
                call    READ
MAIN4:

                jmp     CEKEJ            ;skoc a cekej na dalsi impuls

;-------------------------------------------------------------------------------------------------------
USEK0:
; inicializace casovace 1 - je nutno posunout jeho hodnotu o hodnotu nacitanou v citaci 0

                clr     TR0              ;zastav citac
                mov     R1,TH0           ;sejmi stav citace 0
                mov     A,TL0
                add     A,#12            ;koriguj zastaveni citace prictenim konstanty zdrzeni citace
                mov     TL0,A            ;vrat do citace
                mov     R0,A             ;a uschovej pro dalsi vypocet
                mov     A,R1             ;
                addc    A,#00            ;pricti C k hornimu bytu
                mov     TH0,A            ;uloz do citace
                mov     R1,A             ;a uschovej
                setb    TR0              ;spust opet citac

; sejmuta hodnota casovace 0 je v reg. paru R0,R1

                mov     A,T_ALFA_L       ;priprav hodnotu casu uhlu alfa, urcujici polohu cidla
                add     A,R0             ;
                mov     TL1,A            ;vysledek do citace 1
                mov     A,T_ALFA_H       ;
                addc    A,R1
                mov     TH1,A            ;vysledek do citace 1
                clr     SW_END           ;nuluj priznak konce casovani
                setb    TR1              ;spust citac 1
                mov     USEK,#1          ;dalsi bude uz usek 1
                ret

;-------------------------------------------------------------------------------------------------------

SWITCH:
                push    ACC
                mov     A,USEK

                cjne    A,#1,SW2
                clr     KS1              ;sepni koncovy stupen 1 (zahajeni nabijeni)
                mov     TL1,T_CHAR_L     ;napln citac nabijeci konstantou pro 2. usek
                mov     TH1,T_CHAR_H
                setb    TR1              ;a spust jej
                mov     USEK,#2          ;prepni na dalsi usek pro pristi cyklus
                sjmp    SW5

SW2:
                cjne    A,#2,SW3
                setb    KS1              ;vypni koncovy stupen 1 (zapal jiskry 1,4)
                mov     TL1,T_WAIT_L     ;napln citac cekaci konstantou pro 3. usek
                mov     TH1,T_WAIT_H     ;
                setb    TR1              ;a spust jej
                mov     USEK,#3          ;prepni na dalsi usek pro pristi cyklus
                sjmp    SW5

SW3:
                cjne    A,#3,SW4 ;cekej ve smycce na skonceni useku 3
                clr     KS2              ;sepni koncovy stupen 2 (zahajeni nabijeni)
                mov     TL1,T_CHAR_L     ;napln citac nabijeci konstantou pro 4.usek
                mov     TH1,T_CHAR_H
                setb    TR1              ;a spust jej
```

```
                mov     USEK,#4          ;prepni na dalsi usek pro pristi cyklus
                sjmp    SW5

SW4:
                cjne    A,#4,SW5 ;cekej ve smycce na skonceni useku 4
                setb    KS2              ;vypni koncovy stupen 2 (zapal jiskry 2,3)
                clr     TR1              ;zastav casovac
                mov     USEK,#0          ;prepni na dalsi usek pro pristi cyklus
                setb    SW_END           ;nastav bit informujici hlavni program o ukonceni casovani

SW5:
                pop     ACC
                ret
;-------------------------------------------------------------------------------------------------------


; ***************************   podprogramy    ************************************

;-------------------------------------------------------------------------------------
OFFMODE:                        ;vypnuti koncovych stupnu a nastaveni programove smycky na 20 ms
                setb    KS1              ;vypni oba tranzistory
                setb    KS2
                mov     TL1,#0C0h
                mov     TH1,#063h        ;nastav citac 1 tak aby cital 20 ms - 50 Hz
                setb    TR1              ;a spust jej
                clr     A                ;vynuluj pamet prumeru otacek
                mov     N_P_0,A
                mov     N_P_1,A
                mov     N_P_2,A
                mov     N_P_3,A
                mov     POC_IMP,#128d
;-------------------------------------------------------------------------------------

C_PRUM_P:

; podprogram pro vypocet prumerne hodnoty periody

                mov     A,N_P_0
                add     A,N_L
                mov     N_P_0,A
                mov     A,N_P_1
                addc    A,N_H
                mov     N_P_1,A
                mov     A,N_P_2
                addc    A,N_C
                mov     N_P_2,A
                mov     A,N_P_3
                addc    A,#00
                mov     N_P_3,A

                djnz    POC_IMP,PR_NO

                mov     A,N_P_0
                rlc     A
                mov     A,N_P_1
                rlc     A
                mov     N_P_L,A
                mov     A,N_P_2
                rlc     A
                mov     N_P_H,A
                mov     A,N_P_3
                rlc     A
                mov     N_P_C,A
                clr     A                ;vynuluj pamet prumeru otacek
                mov     N_P_0,A
                mov     N_P_1,A
                mov     N_P_2,A
                mov     N_P_3,A
                mov     POC_IMP,#128d
                setb    NEW_PRUM

PR_NO:
                jnb     NEW_PRUM,No_Div

                mov     D_0,N_P_L
                mov     D_1,N_P_H
                mov     D_2,N_P_C
                mov     OP_0,#00h
                mov     OP_1,#0Eh
                mov     OP_2,#27h
```

```
                mov     OP_3,#07h       ;120 000 000/N

                call    Div_16
                mov     OT_L,OP_0
                mov     OT_H,OP_1
                clr     NEW_PRUM
No_Div:

                ret

;-------------------------------------------------------------------------------
SP_TEST:
;podprogram na test otacek.
;Pokud jsou otacky prilis nizke, nastavuje se bit OT_LO
;Pokud jsou otacky prilis vysoke, nastavuje se bit SP_HI
;Pokud jsou otacky spravne nastavuje se bit OT_OK


                mov     A,N_L           ;test na vysoke otacky (>4000 min-1)
                clr     C
                subb    A,#30h
                mov     A,N_H
                subb    A,#75h
                mov     A,N_C
                subb    A,#00
                mov     OT_HI,C

                clr     C               ;test na nizke otacky (<2000 min-1)
                mov     A,#60h
                subb    A,N_L
                mov     A,#0EAh
                subb    A,N_H
                mov     A,#00
                subb    A,N_C
                mov     OT_LO,C
                orl     C,OT_HI
                cpl     C
                mov     OT_OK,C


                mov     A,#80h          ;test na stop otacky (<1 min-1)
                subb    A,N_L
                mov     A,#084h
                subb    A,N_H
                mov     A,#1Eh
                subb    A,TMP_N_C
                mov     OT_STOP,C

                ret
;-------------------------------------------------------------------------------
SIG_HL:
;podprogram pripravuje pro zobrazeni zpravu o otackach vysokych ci nizkych
                mov     DIS_3,#22d      ;S
                mov     DIS_2,#21       ;P.

                jnb     OT_LO,SIG1
                mov     DIS_1,#19       ;L
                mov     DIS_0,#17       ;o
SIG1:
                jnb     OT_HI,SIG2
                mov     DIS_1,#18       ;H
                mov     DIS_0,#23       ;i
SIG2:
                jnb     OT_STOP,SIG3
                mov     DIS_2,#24d      ;t
                mov     DIS_1,#00d      ;O
                mov     DIS_0,#20d      ;P
SIG3:
                ret
;-------------------------------------------------------------------------------
CALC_TIM:
;podprogram vypocte vsechny casy pro casovani nabijeni a vybijeni obou civek
;Cas T_ALFA je prvni cas mezi impulsem cidla a zacatkem nabijeni civky 1
;Cas T_CHAR je cas nabijeni obou civek (zatim pevne nastaven na 3,2ms)
;Cas T_WAIT je cas cekani mezi vypnutim civky 1 a zapnutim civky 2

; vypocet casu T_CHAR

                mov     A,T_CH          ;konstanta nabijeni
                mov     B,#200          ;
```

```
                mul     AB              ;
                mov     T_CHAR_L,A              ;schovej
                mov     T_CHAR_H,B              ;schovej


; vypocet casu T_ALFA
; T_ALFA=(ALFA_S-ALFA_P)xN/36000-T_CHAR
;
                clr     C               ;nejprve rozdíl ALFA_S-ALFA_P
                mov     A,ALFA_S_L
                subb    A,ALFA_P_L      ;
                mov     R0,A            ;uschovej vysledek
                mov     A,ALFA_S_H
                subb    A,ALFA_P_H
                mov     R1,A            ;uschovej vysledek
                mov     A,R0
                add     A,#10h          ;pricti jeste 10000
                mov     R0,A            ;protoze v promenne ALFA_S se uchovaji jen desitky a jednotky
                mov     A,R1            ;kvuli uspore mista
                addc    A,#27h
                mov     R1,A

                mov     OP_0,N_L ;napln pameti s operandy
                mov     OP_1,N_H
                mov     R2,N_L
                mov     R3,N_H
                mov     OP_2,#0d
                mov     OP_3,#0d
                mov     MUL_16_L,R0
                mov     MUL_16_H,R1

                call    Mul_16          ;a vynasob je
                mov     R0,OP_0
                mov     R1,OP_1
                mov     R2,OP_2
                mov     R3,OP_3


                mov     D_0,#0A0h
                mov     D_1,#8Ch
                mov     D_2,#0h         ;nastav delitele (36 000 = 8CA0h)

                call    Div_16          ;vydel
                mov     R0,OP_0
                mov     R1,OP_1
                mov     R2,OP_2
                mov     R3,OP_3

; ted zbyva odecet hodnoty T_CHAR
; jelikoz vsak je jiz hodnota T_CHAR nastavena tak, aby citac cital tolik kroku kolik je T_CHAR
; staci od teto prednastavene hodnoty jeste odecist vysledek deleni OP_0 a OP_1. Tim ziskame
; hodnotu od ktere ma citac citat tak, aby nacital T_ALFA hodnot.

                mov     A,OP_0          ;nacti hodnotu vysledku deleni nizsi byte
                clr     C               ;nuluj carry
                subb    A,T_CHAR_L      ;odecti T_CHAR nizsi byte
                mov     T_ALFA_L,A      ;uloz do konecneho vysledku
                mov     A,OP_1          ;stejne udelej i s vyssim bytem
                subb    A,T_CHAR_H
                mov     T_ALFA_H,A

                clr     C               ;nyni nastav hodnotu T_ALFA pro inkrementujici citac
                mov     A,#0d           ;tj. od maximalni hodnoty odecti T_ALFA
                subb    A,T_ALFA_L
                mov     T_ALFA_L,A
                mov     A,#0d
                subb    A,T_ALFA_H
                mov     T_ALFA_H,A


; postedni cas je cas T_WAIT. Je to cas mezi vypnutim civky 1 a zapnutim civky 2.
; Vypocte se jako polovina periody otacky - doba nabijeni. Polovinu periody ziskame
; rotaci cisla o jeden bit doprava.

                clr     C
                mov     A,N_H           ;prvni budeme rotovat horni byte
                rrc     A               ;rotuj pres C
                mov     R1,A            ;vysledek si uloz do R1
                mov     A,N_L           ;priprav dolni byte
                rrc     A               ;a rotuj
```

```
                mov     R0,A

; podobne jako v predchozim vypoctu staci od doby T_CHAR odecist tento vysledek a dostaneme
; hodnotu prednastaveni citace takovou, ktera zajisti citani po dobu T_WAIT

                mov     A,R0            ;nacti hodnotu vysledku deleni nizsi byte
                clr     C               ;nuluj carry
                subb    A,T_CHAR_L      ;odecti T_CHAR nizsi byte
                mov     T_WAIT_L,A      ;uloz do konecneho vysledku
                mov     A,R1            ;stejne udelej i s vyssim bytem
                subb    A,T_CHAR_H
                mov     T_WAIT_H,A

                clr     C               ;opet priprav hodnotu pro inkrementujici citac
                mov     A,#0d
                subb    A,T_WAIT_L
                mov     T_WAIT_L,A
                mov     A,#0d
                subb    A,T_WAIT_H
                mov     T_WAIT_H,A


                clr     C               ;opet priprav hodnotu pro inkrementujici citac
                mov     A,#0d
                subb    A,T_CHAR_L
                mov     T_CHAR_L,A
                mov     A,#0d
                subb    A,T_CHAR_H
                mov     T_CHAR_H,A


; nyni je treba  jeste korigovat hodnoty tak, aby zpozdeni v hlavnim programu nezpusobovalo pridavne
; chyby casu. Je nutne tedy doby zkratit o zpozdeni v programu. To se provede nikoliv odectenim
; ale pricenim zpozdeni ke vsem konstantam v pametech T_CHAR, T_ALFA a T_WAIT.


                ret                     ;navrat z podprogramu


;-------------------------------------------------------------------------------------
Mul_16:
; rutina nasobi 16 bitu x 16 bitu
                mov     TMP_3,#0 ;nuluj docasnou pamet
                mov     TMP_2,#0
; generuj nejnizsi byte vysledku
                mov     B,OP_0
                mov     A,MUL_16_L
                mul     AB
                mov     TMP_0,A         ;uschovej vysledek nizsi byte
                mov     TMP_1,B         ;uschovej vyssi byte
; generuj dalsi vyssi byte
                mov     B,OP_1
                mov     A,MUL_16_L
                mul     AB
                add     A,TMP_1         ;pricti predchozi vyssi byte
                mov     TMP_1,A
                mov     A,B             ;vem horni cast vysledku nasobeni
                addc    A,TMP_2         ;a pricti prenos C
                mov     TMP_2,A         ;uloz
                jnc     ML_LOOP         ;jestlize neni prenos, skoc dal
                inc     TMP_3           ;jestlize je prenos, dej carry i do TMP_3
ML_LOOP:
                mov     B,OP_0
                mov     A,MUL_16_H      ;ber dalsi spodni cast operandu
                mul     AB
                add     A,TMP_1         ;pricti predchozi vysledek
                mov     TMP_1,A         ;uloz
                mov     A,B             ;vem horni cast vysledku nasobeni
                addc    A,TMP_2         ;a pricti predchozi vysledek vcetne carry
                mov     TMP_2,A         ;uloz
                jnc     ML_LOOP2 ;jestli neni prenos skoc
                inc     TMP_3           ;prenes prenos i do TMP_3
ML_LOOP2:
; nyni se generuje treti byte
                mov     B,OP_2
                mov     A,MUL_16_L
                mul     AB
                add     A,TMP_2
```

```
                mov     TMP_2,A
                mov     A,B
                addc    A,TMP_3
                mov     TMP_3,A
; druha polovina
                mov     B,OP_1
                mov     A,MUL_16_H
                mul     AB
                add     A,TMP_2
                mov     TMP_2,A
                mov     A,B
                addc    A,TMP_3
                mov     TMP_3,A
;nyni se dokonci nejvyssi cast vysledku
                mov     B,OP_3
                mov     A,MUL_16_L
                mul     AB
                add     A,TMP_3
                mov     TMP_3,A
; dalsi horni cast jiz nepocitej protoze vysledek muze byt maximalne 32 bitu
                mov     B,OP_2
                mov     A,MUL_16_H
                mul     AB
                add     A,TMP_3
                mov     TMP_3,A
; zde vypocet konci
                mov     OP_0,TMP_0      ;napln vysledek do vystupniho operandu
                mov     OP_1,TMP_1
                mov     OP_2,TMP_2
                mov     OP_3,TMP_3

                ret

;-------------------------------------------------------------------------------
BIN2DEC:
;podprogram pro prevod cisla z binarniho do dekadickeho tvaru.

                mov     R2,#00

                mov     A,R1
                mov     B,#10d
                div     AB
                mov     R3,A
                mov     A,R0
                anl     A,#0F0H
                orl     A,B
                swap    A
                mov     B,#10d
                div     AB
                swap    A
                orl     A,R3
                swap    A
                mov     R3,A
                mov     A,R0
                swap    A
                anl     A,#0F0H
                orl     A,B
                swap    A
                mov     B,#10d
                div     AB
                mov     DIS_0,B
                swap    A
                mov     R2,A
                mov     A,R3
                mov     B,#10d
                div     AB
                mov     R3,A
                mov     A,R2
                anl     A,#0F0H
                orl     A,B
                swap    A
                mov     B,#10d
                div     AB
                swap    A
                orl     A,R3
                swap    A
                mov     DIS_1,B
                mov     B,#10d
                div     AB
                mov     DIS_2,B
```

```
                mov     DIS_3,A
                ret

;-------------------------------------------------------------------------------
; podprogram dělí 32 bitový OP registr hodnotou 16 bitového registru D_1, D_0
Div_16:

                mov     R7,#0
                mov     R6,#0
                mov     R5,#0
                mov     TMP_0,#0
                mov     TMP_1,#0
                mov     TMP_2,#0
                mov     TMP_3,#0 ;vynuluj paměti mezivýsledků
                mov     R2,D_2
                mov     R1,D_1          ;načti dělitele
                mov     R0,D_0
                mov     R4,#32          ;nastav počítadlo bitů
;dělicí smyčka
Div_loop:
                call    Shift_D         ;vysuň dělence a vrať MSB v C
                mov     A,R5
                rlc     A
                mov     R5,A            ;
                mov     A,R6            ;nasuň carry do LSB částečného výsledku
                rlc     A
                mov     R6,A
                mov     A,R7
                rlc     A
                mov     R7,A

;nyní testuj zda R7:R6:R5 >= R2:R1:R0
                jc      Can_sub         ;můžeš odečítat
                clr     C
                mov     A,R7            ;odečti R2 od R7 a zjisti zda R2 < R7
                subb    A,R2            ;A = R7 - R2, carry je nastaveno když R7 < R2
                jc      Cant_sub ;nemůžeš odečítat

;v této chvíli je R6 > R1 nebo R6 = R1
                jnz     Can_sub         ;skoč když je R6 > R1

;nyní testuj když R7 = R1 zda R6 >= R0
                clr     C
                mov     A,R6
                subb    A,R1            ;A = R6 - R1, carry je nastaveno když R6 < R1
                jc      Cant_sub

                jnz     Can_sub         ;skoč když je R5 > R0

;nyní testuj když R5 = R0 zda R5 >= R0
                clr     C
                mov     A,R5
                subb    A,R0            ;A = R5 - R0, carry je nastaveno když R5 < R0
                jc      Cant_sub

Can_sub:
;odečti dělitele od částečného výsledku

                clr     C
                mov     A,R5
                subb    A,R0            ;A = R5 - R0
                mov     R5,A
                mov     A,R6
                subb    A,R1            ;A = R6 - R1 - C
                mov     R6,A
                mov     A,R7
                subb    A,R2            ;A = R7 - R1 - C
                mov     R7,A
                setb    C               ;nasuň do výsledku 1
                sjmp    Quot
Cant_sub:
                clr     C               ;jinak nasuň 0

Quot:
                call    Shift_Q
                djnz    R4,Div_loop     ;je už smyčka celá, ne opakuj
                mov     OP_0,TMP_0
                mov     OP_1,TMP_1
                mov     OP_2,TMP_2
                mov     OP_3,TMP_3
```

```
                ret

Shift_D:
;posun dělence o jeden bit vlevo a vrácení MSB v C

                clr     C
                mov     A,OP_0
                rlc     A
                mov     OP_0,A
                mov     A,OP_1
                rlc     A
                mov     OP_1,A
                mov     A,OP_2
                rlc     A
                mov     OP_2,A
                mov     A,OP_3
                rlc     A
                mov     OP_3,A
                ret

Shift_Q:

                mov     A,TMP_0
                rlc     A
                mov     TMP_0,A
                mov     A,TMP_1
                rlc     A
                mov     TMP_1,A
                mov     A,TMP_2
                rlc     A
                mov     TMP_2,A
                mov     A,TMP_3
                rlc     A
                mov     TMP_3,A
                ret


;-------------------------------------------------------------------------------
NADISP:
;program dekoduje obsah pameti DIS_0 až DIS_3 a výsledek odesílá primo na display pomoci
;podprogramu VYSLI. V A tedy musi byt pripraveny data pro display v seriové podobe.
;Posledni segmentovka se musi vysilat jako prvni

                mov     DPTR,#TABZNAK   ;do datapointeru adresa pocatku tabulky znaku


                mov     A,DIS_0          ;nacti znak k dekodovani
                clr     C
                call    VYSLI

                mov     A,DIS_1          ;nacti znak k dekodovani
                mov     C,DECIM1
                clr     DECIM1
                call    VYSLI

                mov     A,DIS_2          ;nacti znak k dekodovani
                mov     C,DECIM2
                clr     DECIM2
                call    VYSLI

                mov     A,DIS_3          ;nacti znak k dekodovani
                mov     C,DECIM3
                clr     DECIM3
                call    VYSLI

                ret


;-------------------------------------------------------------------------------
VYSLI:
;podprogram vysílá data na display. Je vzat kód v A, kde jsou nastaveny
;svítící a nesvítící segmenty

                movc    A,@A+DPTR       ;dekoduj kod v A podle dekodovaci tabulky a uloz zpet do A
                jnc     NEXT_1
                orl     A,#00000001b     ;posledni bit v A nastav na 1
NEXT_1:
                cpl     A                ;invertuj bity pokud je pouzita spolecna anoda
                mov     R0,#8d           ;nastaveni poctu segmentu
NEXTSEG: rrc     A              ;posun o jednu pozici vpravo pres C
```

```
                clr     DIS_CLK         ;vynuluj hodiny pro SIPO
                mov     DIS_DAT,C       ;bit v C jsou data pro display
                setb    DIS_CLK         ;posun hodiny a zapis tak data do SIPO
                djnz    R0,NEXTSEG       ;dalsi segment

                ret                     ;navrat


;-------------------------------------------------------------------------------
READ_KEY:
;podprogram pro obsluhu klavesnice
;zjistuje se ktera klavesa je aktualne stiskla, popripade jak dlouho.
;nastavuje nasledujici bity
;K_UP            priznak stisknute klavesy UP
;K_DOWN         priznak stisknute klavesy DOWN
;K_MODE         priznak stiskleho MODE tlacitka
;KPLT           priznak urcujici dlouhe stisknuti tlacitka
;KPST           priznak dele stisknuteho tlacitka
;K_GO_UP        priznak uvolneni tlacitka
;K_PRESS        priznak stisku jakekoliv klavesy
;K_PRESSP       priznak stisku klavesy v predchozim cyklu


                mov     C,UP            ;zjisti stav klavesy UP
                cpl     C
                mov     K_UP_T,C ;a uloz do pametoveho bitu

                mov     C,DOWN          ;zjisti stav klavesy DOWN
                cpl     C
                mov     K_DOWN_T,C      ;a uloz do pametoveho bitu

                orl     C,K_UP_T ;pokud neni ani jedna klavesa stiskla je K_UP + K_DOWN = 0
                jc      KEY_1           ;kdyz je nejaka klavesa stiskla pak nema smysl testovat MODE
                clr     DOWN            ;nuluj stav na DOWN
                mov     R0,#20d         ;cekej 20us
KEY_X:          djnz    R0,KEY_X
                mov     C,UP            ;a zjisti, zda se nula objevi i na UP
                cpl     C
                mov     K_MODE_T,C      ;a uloz do pametoveho bitu
                setb    DOWN            ;nastav zpet do stavu H tlacitko DOWN
KEY_1:
                mov     C,K_MODE_T      ;
                orl     C,K_UP_T
                orl     C,K_DOWN_T
                mov     K_PRESS,C       ;nastav bit stiskle klavesy pri stisku jakekoliv

                jnc     KEY_2           ;kdyz neni nic stiskle tak bez dale

                mov     C,K_UP_T ;presun hodnoty z docasnych pameti
                mov     K_UP,C          ;do pameti indikatoru stisklych klaves
                mov     C,K_DOWN_T
                mov     K_DOWN,C
                mov     C,K_MODE_T
                mov     K_MODE,C

                mov     A,#1d           ;pricti jednicku
                add     A,T_PRESS       ;
                jc      KEY_5           ;a pri preteceni neukladej
                mov     T_PRESS,A       ;jinak uloz novy cas
KEY_5:
                clr     C
                mov     A,#4d
                subb    A,T_PRESS       ;testuj minimalni dobu stisku klavesy 4 takty cca 80ms
                mov     KPOK,C          ;klavesa skutecne stisknuta
                mov     A,#20d          ;kdyz je nejaka klavesa stiskla, tak testuj, zda je stiskla
                subb    A,T_PRESS       ;dele nez 20 cyklu (cca 1s)
                mov     KPST,C          ;pri prenosu ano pak nastav bit KPST- stisknuto kratsi cas
                mov     A,#40d          ;dele nez 40 cyklu (cca 2s)
                subb    A,T_PRESS
                mov     KPLT,C          ;nastav KPLT - stisknuto delsi cas


KEY_2:
                mov     C,K_PRESS       ;do C stav stiskle klavesnice
                jc      KEY_6           ;jestlize je neco stiskle preskoc
                mov     T_PRESS,#0d     ;jinak nuluj cas pocitadla casu stiskle klavesy
KEY_6:
                cpl     C
                anl     C,K_PRESSP      ;jestlize neni stiskla klavesa a v predchozim byla, C=1
                anl     C,KPOK          ;a jestlize byla stiskla dele nez 200 ms
```

```
                jnc     KEY_4           ;neni detekovano uvolneni tlacitka
                setb    K_GO_UP         ;nastav bit uvolneneho tlacitka
                jmp     KEY_7
KEY_4:
                clr     K_GO_UP         ;smaz priznak stiskleho tlacitka
KEY_7:
                mov     C,K_PRESS       ;a nastav soucasny stav stisku jako predchozi stav
                mov     K_PRESSP,C
                ret
;-------------------------------------------------------------------------------------------
MENU_SEL:
;podprogram vyberu menu

                mov     R7,MENU

;****************** M  E  N  U   1  ****************************

                cjne    R7,#1,M_2       ;menu 1 - zobrazuji se pouze otacky motoru

                jb      OT_OK,M_1_1
                call    SIG_HL          ;pokud nejsou spravne, tak nastav hlasku o chybnych otackach
M_1_1:
                jnb     OT_OK,M_1_2
                mov     R0,OT_L         ;pokud jsou spravne tak nastav hodnotu otacek
                mov     R1,OT_H
                call    BIN2DEC
M_1_2:
                jnb     K_GO_UP,M_1_3
                mov     MENU,#2d
                clr     K_GO_UP
                mov     POZ_MENU,#0d
M_1_3:


;****************** M  E  N  U   2  ****************************
; hlavni nabidkove menu, vyber ALFA.P ALFA.S t.Chr dEf SAVE LOAd

M_2:            cjne    R7,#2,M_3

                jnb     K_GO_UP,M_2_1
                clr     A
                mov     C,K_UP
                addc    A,POZ_MENU
                mov     C,K_DOWN
                subb    A,#0d
                jnc     M_2_3
                mov     A,#POZ1_MAX
M_2_3:          mov     POZ_MENU,A
                setb    C
                subb    A,#POZ1_MAX
                jc      M_2_4
                mov     POZ_MENU,#0d
M_2_4:
M_2_1:          mov     C,K_MODE
                anl     C,K_GO_UP
                jnc     M_2_5
                mov     A,#3d
                add     A,POZ_MENU
                mov     MENU,A
                mov     C,K_MODE
                anl     C,K_GO_UP
                anl     C,KPST
                jnc     M_2_5
                mov     MENU,#1d
M_2_5:
                mov     A,POZ_MENU
                mov     B,#4d
                mul     AB
                mov     R6,A            ; zapamatuj si
                mov     R5,#4d          ; pocet segmentovek na display
                mov     R0,#13h         ; adresa prvni segmentovky v pameti
                mov     DPTR,#TAB_HL

M_2_6:          mov     A,R6
                movc    A,@A+DPTR
                mov     @R0,A
                inc     R6
                dec     R0
                djnz    R5,M_2_6
```

```
;****************** M  E  N  U   3  ****************************
; zmena ALFA.P - uhel predstihu

M_3:            cjne    R7,#3,M_4

                mov     C,K_MODE
                anl     C,K_GO_UP
                jnc     M_3_1
                mov     MENU,#2
                mov     POZ_MENU,#0
M_3_1:
                mov     R0,ALFA_P_L
                mov     R1,ALFA_P_H
                mov     R3,#APMINL
                mov     R4,#APMINH
                mov     R5,#APMAXL
                mov     R6,#APMAXH
                mov     R2,#10d
                call    ZMENA
                mov     ALFA_P_L,R0
                mov     ALFA_P_H,R1

                call    BIN2DEC
                mov     DIS_0,DIS_1
                mov     DIS_1,DIS_2
                mov     DIS_2,DIS_3
                mov     DIS_3,#27d
                setb    DECIM1


;****************** M  E  N  U   4  ****************************
; zmena ALFA.S - uhel umisteni senzoru

M_4:            cjne    R7,#4,M_5

                mov     C,K_MODE
                anl     C,K_GO_UP
                jnc     M_4_1
                mov     MENU,#2
                mov     POZ_MENU,#1
M_4_1:
                mov     R0,ALFA_S_L
                mov     R1,ALFA_S_H
                mov     R3,#ASMINL
                mov     R4,#ASMINH
                mov     R5,#ASMAXL
                mov     R6,#ASMAXH
                mov     R2,#10d
                call    ZMENA
                mov     ALFA_S_L,R0
                mov     ALFA_S_H,R1

                call    BIN2DEC
                mov     DIS_0,DIS_1
                mov     DIS_1,DIS_2
                mov     DIS_2,DIS_3
                mov     DIS_3,#1d
                setb    DECIM1

;****************** M  E  N  U   5  ****************************
; zmena casu T_CHR cas nabijeni civky

M_5:            cjne    R7,#5,M_8

                mov     C,K_MODE
                anl     C,K_GO_UP
                jnc     M_5_1
                mov     MENU,#2
                mov     POZ_MENU,#2
M_5_1:
                mov     R0,T_CH
                mov     R1,#0d
                mov     R2,#1d
                mov     R3,#TCHMIN
                mov     R4,#0d
                mov     R5,#TCHMAX
                mov     R6,#0d
```

```
                call    ZMENA
                mov     T_CH,R0


                call    BIN2DEC
                mov     DIS_2,#27d
                mov     DIS_3,#27d
                setb    DECIM1

;****************** M  E  N  U   6  ****************************
; dEF. obnovuje predvolene (defaultni) hodnoty


M_8:            cjne    R7,#6,M_9

                mov     C,K_MODE
                anl     C,K_GO_UP
                jnc     M_8_1
                mov     MENU,#2
                mov     POZ_MENU,#3
M_8_1:
                call    DEFAULT
                mov     DIS_0,#14d      ;E
                mov     DIS_1,#31d      ;n
                mov     DIS_2,#17d      ;o
                mov     DIS_3,#26d      ;d
                djnz    DIS_TIME,M_9    ;cekej stanoveny cas
                mov     DIS_TIME,#50d   ;znovu nastav cas pro zpravu
                mov     MENU,#2         ;nastav navrat do menu 2
                mov     POZ_MENU,#3



;****************** M  E  N  U   7  ****************************
; SAVE uklada hodnoty do pameti.

M_9:            cjne    R7,#7,M_10

                mov     C,K_MODE
                anl     C,K_GO_UP
                jnc     M_9_1
                mov     MENU,#2
                mov     POZ_MENU,#4
M_9_1:
                mov     CODE_KEY,#CODE
                setb    STORE           ;nastav priznak ukladani
                jnb     STO_END,M_9_2   ;dokud se data neulozi preskoc na konec
                mov     DIS_0,#19d      ;L
                mov     DIS_1,#01d      ;I
                mov     DIS_2,#10d      ;A
                mov     DIS_3,#15d      ;F
                jnb     STO_OK,M_9_3    ;preskoc dokud se data neulozi spravne
                mov     DIS_0,#14d      ;E
                mov     DIS_1,#31d      ;n
                mov     DIS_2,#17d      ;o
                mov     DIS_3,#26d      ;d
                djnz    DIS_TIME,M_9_3  ;cekej stanoveny cas
                mov     DIS_TIME,#50d   ;znovu nastav cas pro zpravu
                mov     MENU,#2         ;nastav navrat do menu 2
                mov     POZ_MENU,#4
                clr     STO_OK          ;nuluj priznak spravneho ulozeni
                clr     STO_END

M_9_3:          clr     STORE
                clr     STO_STRT

M_9_2:



;****************** M  E  N  U   8  ****************************
; LOAD natazeni dat z pameti EEPROM

M_10:           cjne    R7,#8,M_11

                mov     C,K_MODE
                anl     C,K_GO_UP
                jnc     M_10_1
                mov     MENU,#2
```

```
                mov     POZ_MENU,#5
M_10_1:
                setb    RECALL
                jnb     RCL_OK,M_11
                mov     DIS_0,#14d      ;E
                mov     DIS_1,#31d      ;n
                mov     DIS_2,#17d      ;o
                mov     DIS_3,#26d      ;d
                clr     RECALL
                djnz    DIS_TIME,M_11   ;cekej stanoveny cas
                mov     DIS_TIME,#50d   ;znovu nastav cas pro zpravu
                mov     MENU,#2         ;nastav navrat do menu 2
                mov     POZ_MENU,#5
                clr     RCL_OK

M_11:

                ret
;-----------------------------------------------------------------
ZMENA:
;podpogram pro zmenu hodnoty dle stisku klavesy UP a DOWN

                mov     A,R0


                mov     C,K_PRESS
                anl     C,KPLT
                jnc     Z_3
                jmp     Z_4

Z_3:            jnb     K_GO_UP,Z_2
                clr     K_GO_UP

Z_4:            jnb     K_UP,Z_1

                mov     A,R5            ;test na dosazeni maximalni hodnoty
                setb    C               ;ulozene v paru R5,R6
                subb    A,R0
                mov     A,R6
                subb    A,R1
                jc      Z_2

                mov     A,R0            ;pokud jeste neni maximum dosazeno muzes pricist
                add     A,R2
                mov     R0,A
                mov     A,R1
                addc    A,#0d
                mov     R1,A

Z_1:
                jnb     K_DOWN,Z_2

                mov     A,R0            ;test na dosazeni minimalni hodnoty
                setb    C               ;ulozene v paru R3,R4
                subb    A,R3
                mov     A,R1
                subb    A,R4
                jc      Z_2

                mov     A,R0            ;pokud jeste neni minimum dosazeno muzes odecist
                subb    A,R2
                mov     R0,A
                mov     A,R1
                subb    A,#0d
                mov     R1,A
Z_2:
                ret
;-----------------------------------------------------------------
DEFAULT:
;podprogram pro natazeni predvolenych hodnot
                mov     ALFA_S_L,#A_S_L
                mov     ALFA_S_H,#A_S_H
                mov     ALFA_P_L,#A_P_L
                mov     ALFA_P_H,#A_P_H
                mov     T_CH,#CHARGE
                ret

;-----------------------------------------------------------------

;OBSLUHA PAMETI EEPROM
```

```
;-------------------------------------------------------------------------------
READ:
;podprogram ktery natahne data z pameti EEPROM 93C46
;

                mov     R0,#18H         ;nastav ukazatel RAM
                mov     ADREPROM,#00    ;nastav ukazatel EEPROM
r00:
                setb    CS              ;vyber EEPROM
                mov     DPL,#110b       ;start bit (1) a op kod (10) cteni
                mov     B,#3            ;delka kodu
                call    OUTDATA         ;vysli ven do EEPROM
                mov     DPL,ADREPROM    ;priprav adresu
                mov     B,#7            ;maximalni delka adresy 7 bitu
                call    OUTDATA         ;vysli ven do EEPROM
                call    INDATA          ;a precti vystup z pameti
                clr     CS              ;ukonci vyber EEPROM
                mov     A,DATEPROM      ;data z pameti do A
                mov     @R0,A           ;data z pameti uloz do RAM
                inc      R0
                inc     ADREPROM        ;posun na nasledujici bunku
                mov     A,ADREPROM      ;
                cjne    A,#7,r00          ;porovnej zda to neni posledni adresa
                setb     RCL_OK
                ret                     ;kdyz ano konec a navrat

;-------------------------------------------------------------------------------
WRITE:

;rutina ulozeni konstant do vnejsi pameti EEPROM 93C46

                jb       STO_STRT,w00     ;uz se zacalo ukladat, preskoc inicializaci
                call    EWEN            ;povol zapis do pameti
                mov     ADRRAM,#18H     ;nastav bazovou adresu RAM
                mov     ADREPROM,#00H   ;nastav bazovou adresu EEPROM
                clr     WRITE_OK
                setb     STO_STRT ;nastav bit startu ukladani

w00:
                setb    CS              ;aktivuj pamet
                mov     DPL,#101b       ;starbit a op kod
                mov     B,#3            ;delka kodu
                call    OUTDATA         ;vysli do EEPROM
                mov     DPL,ADREPROM
                mov     B,#7
                call    OUTDATA
                mov      R0,ADRRAM        ;nastav ukazatel
                mov     DPL,@R0
                mov     B,#8
                call    OUTDATA
                clr     CS              ;nuluj CS
                call    STATUS          ;cekej na zapis do pameti a zjisti stav
                orl     C,WRITE_OK      ;vysledek zapisu do pameti porovnej s predchozima
                mov     WRITE_OK,C
                inc     ADRRAM
                inc     ADREPROM
                mov     A,ADREPROM
                cjne    A,#7H,w01
                mov      C,WRITE_OK
                cpl      C
                mov      STO_OK,C
                setb     STO_END
                clr      STO_STRT
                call    EWDS            ;zakaz zapis do pameti

w01:
                ret                     ;navrat


;-------------------------------------------------------------------------------
EWEN:
;podprogram, ktery povoluje zapis do pameti EEPROM
;je nutny pred kazdym zapisem pokud je pamet ve stavu EWDS

                mov     A,CODE_KEY              ;do A kodovy klic
                cjne    A,#CODE,key_bad         ;porovnej ho s kodem
                sjmp    key_ok                  ;klic je dobry

key_bad:                                       ;klic je spatny
                jmp     RESET                   ;resetuj obvod
```

```
key_ok:

                setb    CS
                mov     DPTR,#(10011b SHL 5d)   ;op code
                mov     B,#10d
                call    OUTDATA
                clr     CS
                ret
;-------------------------------------------------------------------------------
EWDS:
;podprogram, ktery uzavre pristup do pameti pro zapis a mazani.


                setb    CS
                mov     DPTR,#(10000b SHL 5)
                mov     B,#10d
                call    OUTDATA
                clr     CS
                mov     CODE_KEY,#00
                ret
;-------------------------------------------------------------------------------
OUTDATA:
;podprogram vysle sekvenci dat ulozenou v DPTR o delce, ktera je ulozena v B
;nici DPTR a A

                push    B
                mov     A,B
                clr     C
                subb    A,#8d
                jc      ee6             ;sekvence mensi nez 8
                jz      ee5             ;sekvence rovna 8
                mov     B,A             ;sekvence vetsi nez 8
                clr     C
                subb    A,#8
                jc      ee2
                jc      ee9

                mov     A,DPH
                sjmp    ee4
ee2:
                push    B
                mov     A,DPH
ee3:
                rr      A
                djnz    B,ee3
                pop     B
ee4:
                call    SHOUT           ;ven do pameti
                mov     B,#8
ee5:
                mov     A,DPL
                sjmp    ee8
ee6:
                push    B
                mov     A,DPL
ee7:
                rr      A
                djnz    B,ee7
                pop     B
ee8:
                call    SHOUT
ee9:
                setb    DO              ;nechej pin plovouci
                pop     B
                ret
;-------------------------------------------------------------------------------
INDATA:
;podprogram nacte data z pameti

                setb    DO              ;pin DO jako plovouci
                push    B
                clr     SK              ;hodiny do L
                mov     B,#8d           ;nastav pocitadlo
ee30:
                setb    SK              ;hodiny vzestupna hrana
                nop
                nop                     ;cekej 1us
                mov     C,DO            ;nacti data
                rlc     A               ;nasun do A
                clr     SK
```

```
                djnz    B,ee30
                pop     B
                mov     DATEPROM,A
                ret
;-------------------------------------------------------------------------------
SHOUT:
;program na vysilani dat do EEPROM

ee50:
                clr     SK              ;hodiny na L
                rlc     A               ;vysun data z A
                mov     DI,C            ;posli je na pin DI
                nop                     ;cekej min 400 ns
                setb    SK              ;nastupnou hranou zapis
                djnz    B,ee50          ;opakuj B krat
                clr     SK              ;hodiny do L
                ret
;-------------------------------------------------------------------------------
STATUS:
;podprogram ktery overuje spravnost zapisu dat do pameti.
;vraci C nastavene kdyz je nejaka chyba
;
                push    B
                setb    DO              ;plovouci pin DO
                setb    CS              ;vyber pamet
                mov     B,#220d         ;220 x 50 us = 11 ms do teto doby musi pamet odpovedet
ee40:
                push    B               ;1 us
                mov     B,#47d          ;1 us
ee41:
                djnz    B,ee41
                pop     B
                jb      DO,ee42         ;testuj odpoved pameti
                djnz    B,ee40          ;kdyz neodpovida cekej
                setb    C               ;prekrocena doba cekani nastav error
                sjmp    ee43
ee42:           clr     C
ee43:           clr     CS              ;ukonci vyber
                pop     B
                ret

;-------------------------------------------------------------------------------

; tabulka hlasek menu 2
;               ORG     7BDh              ;pocatek tabulky v pameti
TAB_HL:
                DB      10d
                DB      19d
                DB      28d
                DB      20d

                DB      10d
                DB      19d
                DB      28d
                DB      22d

                DB      30d
                DB      12d
                DB      18d
                DB      29d

                DB      27d
                DB      13d
                DB      14d
                DB      28d

                DB      22d
                DB      10d
                DB      25d
                DB      14d

                DB      19d
                DB      00d
                DB      10d
                DB      13d


;dekodovaci tabulka pro prevod cisla na kod 7-segmentoveho displaye
;1 - segment sviti
;0- segment nesviti
```

```
; segmenty jsou v poradi ABCDEFGH

TABZNAK:

;                       binarni tvar            znak    kod
                DB      11111100b               ;0      0
                DB       01100000b              ;1      1
                DB       11011010b              ;2      2
                DB       11110010b              ;3      3
                DB       01100110b              ;4      4
                DB       10110110b              ;5      5
                DB      10111110b               ;6      6
                DB       11100000b              ;7      7
                DB       11111110b              ;8      8
                DB       11110110b              ;9      9
                DB       11101110b              ;A      10
                DB       00111110b              ;b      11
                DB       10011100b              ;C      12
                DB       01111010b              ;d      13
                DB       10011110b              ;E      14
                DB       10001110b              ;F      15
                DB       00000010b              ;-      16
                DB       00111010b              ;o      17
                DB       01101110b              ;H      18
                DB       00011100b              ;L      19
                DB       11001110b              ;P      20
                DB       11001111b              ;P.     21
                DB       10110110b              ;S      22
                DB       00100000b              ;i      23
                DB       00011110b              ;t      24
                DB       01111100b              ;U      25
                DB       01111010b              ;d      26
                DB       00000000b              ;       27
                DB       10001111b              ;F.     28
                DB       00001010b              ;r      29
                DB       00011111b              ;t.     30
                DB       00101010b              ;n      31


Konec:
end
```