Střední škola informatiky, elektrotechniky a řemesel

Rožnov pod Radhoštěm

Dokumentace k maturitní práci

nf osciloskop s výstupem na TV

Autor:
Radim Pechal, S 4.
2006/2007

Konzultant práce:
Ing. Jiří Král

I. nf osciloskop s výstupem na TV

1. Úvod

Osciloskop patří bezesporu mezi základní měřící přístroje v elektrotechnice. Osciloskopy se vyrábějí ve dvou základních provedeních - analogové a digitální. Amatérská konstrukce analogového osciloskopu je poměrně složitá a nákladná, naopak konstrukce digitálního osciloskopu je vzhledem k dostupnosti mikrokontrolérů mnohem snazší.

Jedním ze základních prvků každého osciloskopu je zobrazovací jednotka. U analogových osciloskopů se můžeme setkat s osciloskopickými obrazovkami. Moderní digitální osciloskopy používají různé displeje, které bývají často velmi nákladné. Proto jsou některé konstrukce pojaty jako moduly, které se dají připojit k počítači. Data se pak zobrazují pomocí nainstalovaného software na monitoru počítače. Nevýhodou takové koncepce je požadavek na počítač, tedy větší energetická náročnost zařízení a také vyšší cena.

Snahou této konstrukce bylo vytvořit osciloskop, který by byl nezávislý na počítači, zároveň aby jeho displej nabízel dostatečný prostor pro zobrazování naměřených dat. Po zvážení dostupných možností byla zvolena jako zobrazovací jednotka obrazovka standardního televizoru. Výhoda televizoru je v tom, že nabízí dostatečné rozlišení, snadnou komunikaci se zařízením a všeobecnou dostupnost.

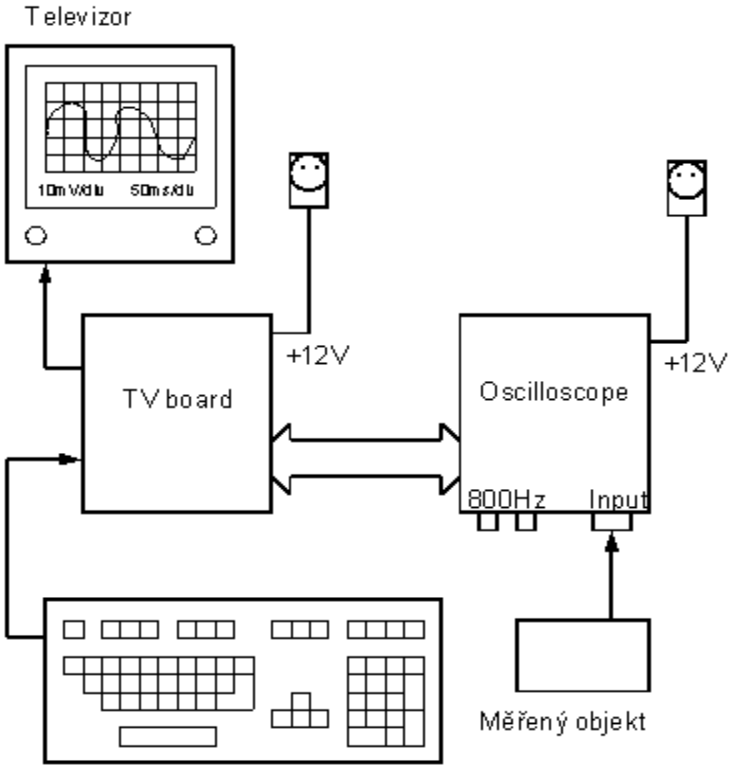
Většina digitálních osciloskopů se skládá z rychlého AD převodníku, paměti FISO[1] a z řídícího obvodu. Tato koncepce je standardní a osvědčená, dostupnost součástkové základny umožňuje tyto bloky zahrnout do jediné součástky - mikrokontroléru.

Výhodou této koncepce je jednoduchost a nízké náklady, nevýhodou je nižší vzorkovací frekvence osciloskopu. Protože cílem konstrukce bylo vytvořit osciloskop, který umožní měření v oblasti nízkofrekvenčních kmitočtů, je možné toto omezení přijmout.

Výsledkem je nf osciloskop, který dokáže snímat signál s vzorkovací frekvencí 500 kHz.

2. Blokový popis jednotlivých celků

Celkové zařízení se dělí na dvě části.



obr. 1.Koncepce zařízení

První část zařízení, kterou jsem nazval modul TV terminálu. Jeho úkolem je zajistit komunikaci mezi uživatelem a osciloskopem. Hlavním úkolem TV terminálu je zobrazování naměřených dat na TV obrazovce.

1 Paměť FISO (fast in slow out) je schopna ve velmi krátkém časovém intervalu uložit data, která lze po té z paměti načíst.

Komunikaci uživatele s osciloskopem umožňuje klávesnice pro osobní počítače kompatibilní s IBM AT.

Druhou částí je osciloskop, který slouží k samotnému měření signálu. Toto zařízení obsahuje vstupní obvody pro úpravu měřeného signálu a mikrokontrolér, který obsahuje AD převodník pro převod napětí z analogové podoby do digitální. Dále umožňuje úpravu získaných dat a jejich odeslání k zobrazení.

II. Modul TV terminálu

1. Úvod

Snahou bylo vytvořit zařízení, které by se chovalo jako terminál s výstupem na televizní obrazovku. U terminálu se data nejen přijímají, ale jsou také vysílána. Proto je k modulu kromě TV připojena také klávesnice. Volil jsem klávesnici kompatibilní s IBM AT. Tento typ klávesnice je velmi levný a běžně dostupný.

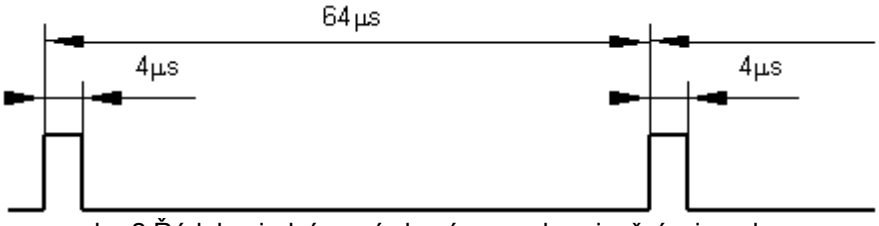
TV obrazovka má dostatečné rozlišení pro zobrazování znaků, ale také dostatečné rozlišení pro základní grafiku. Funkce standardního znakového terminálu bylo nutné doplnit o jednoduché příkazy, které umožňují vykreslovat na obrazovku čáry, obdélníky a další.

Vývoj tohoto zařízení si vyžádal i doplnění některých speciálních funkcí určených pro osciloskop. Přesto výsledné zařízení nabízí univerzální využití i v jiných typech přístrojů a zařízení.

2. Popis televizního signálu

Pro komunikaci s televizorem TV terminál generuje černobílý video signál, který časově odpovídá normě PAL. Data přicházející na obrazovku jsou rozkládána do jednotlivých snímků a jednotlivých řádků. Generovaný signál využívá lineárního řádkování. Při odesílání dat na obrazovku se musí ve správných časových intervalech vkládat synchronizační impulsy.

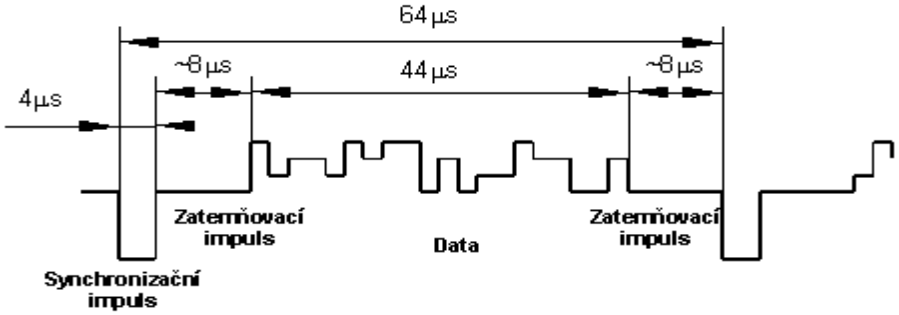
Snímkové synchronizační impulsy udávají okamžik, kdy bude zahájeno vykreslování nového snímku a elektronový paprsek by se vrací zpět do levého horního rohu obrazovky. Snímková frekvence (počet snímků za sekundu) je 50Hz, s touto periodou také musí přicházet sekvence tří řádků se snímkovým synchronizačním impulsem.



obr. 2.Řádek s jedním snímkovým synchronizačním impulsem

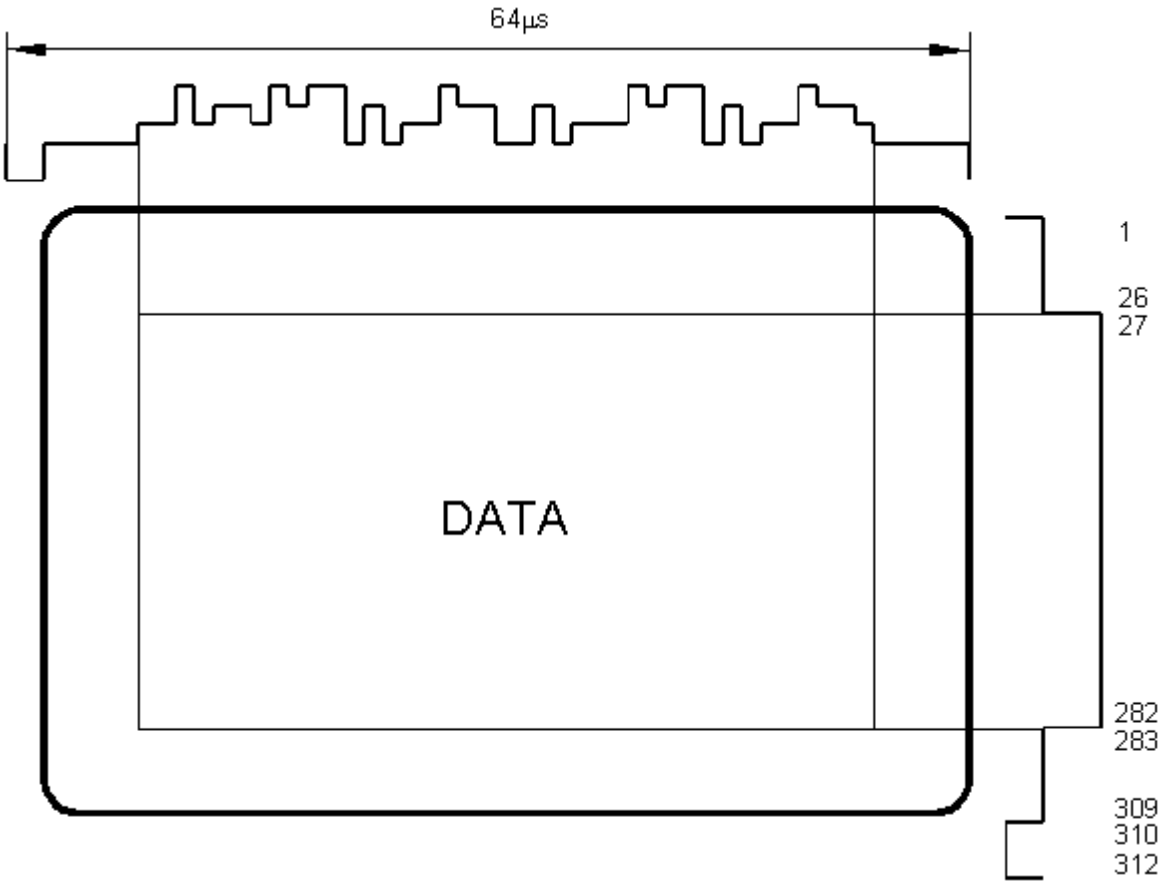
Řádkové synchronizační impulsy udávají okamžik, kdy se začíná vykreslovat jeden řádek a elektronový impuls se vrátí na začátek řádku. Samotný synchronizační impuls potom trvá 4μs a je definován úrovní signálu na 0 V.

Jeden řádek začíná řádkovým synchronizačním impulsem, potom následuje tzv. zatemňovací impuls ~8μs, je to doba, kdy nejsou vysílána žádná data a čeká se, než se paprsek posune na začátek zobrazované plochy. Poté se začnou vysílat data. Ty se vysílají 44μs. Následně řádek dobíhá opět zatemňovacím impulsem.



obr. 3.Časový průběh jednoho řádku

Průběh jednoho snímku je složen z ~26 prázdných řádků. Během těchto řádků se nezobrazují data, protože tyto řádky můžou být deformovány zaoblením obrazovky. Potom následuje 256 datových řádků po kterých se odvysílá ~27 prázdných řádků. Následují 3 řádky snímkových synchronizačních impulsů. Celkový počet řádků tedy odpovídá 312 řádkům, které jsou definovány v normě PAL..



obr. 4.Časový průběh jednoho snímku

3. Způsob generování signálu

Jako základ zařízení je použit mikrokontrolér ATmega 8515. Program mikrokontroléru jde rozdělit na dvě části. První je hlavní smyčka, která zpracovává příchozí znaky a ukládá jejich podobu do externí paměti a druhá část je obsluha přerušení, které generuje video signál.

Aby se obraz nevytrhával, musí přicházet přerušení přesně každých 64μs (doba jednoho řádku). Na začátku se vždy vygeneruje řádkový synchronizační impuls (pokud se nemá řádek se snímkovým synchronizačním impulsem). Počká se přibližně 8μs (zatemňovací impuls) a začnou se vysílat data. Po odvysílání dat se přerušení ukončí.

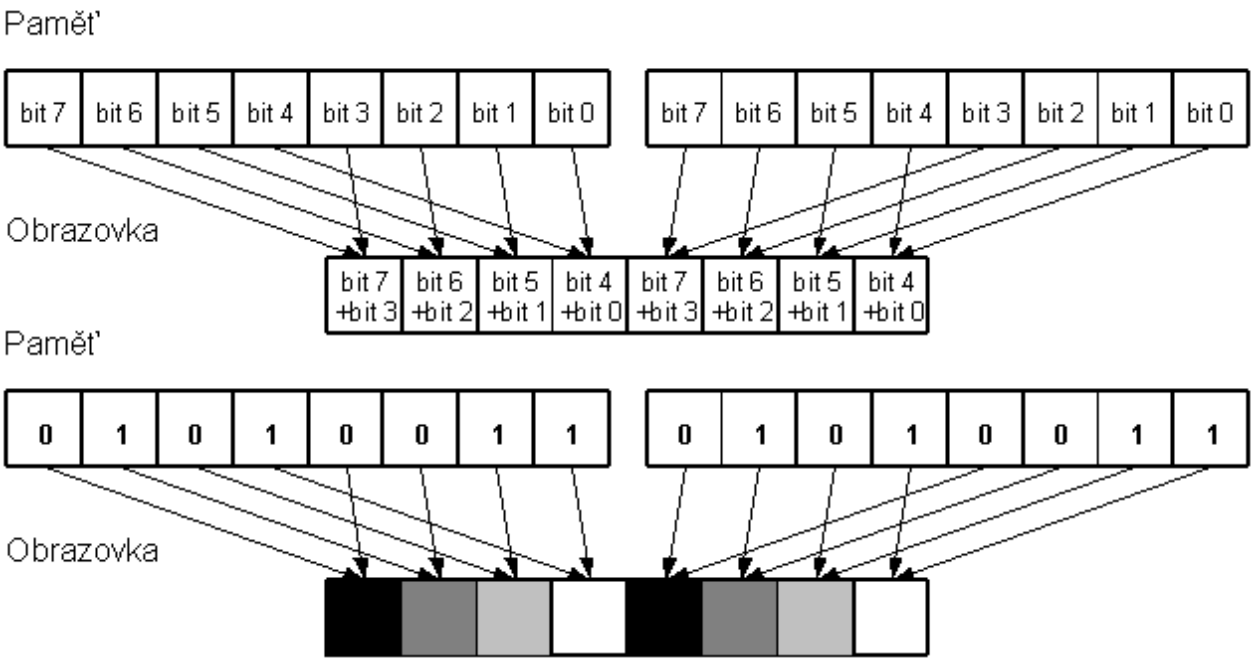
Ukázalo se, že problémem je různá doba instrukcí v hlavním programu. Instrukce mohou trvat 1 až 4 instrukční cykly[2]. Přerušení nastává až po ukončení dané instrukce, a proto začátek přerušení se při používání 12 MHz krystalu u procesoru může lišit až o 333 ns což je již viditelné okem.

Proto program využívá dvě přerušení. První přerušení přichází každých 64μs (doba jednoho řádku). V tomto „přípravném“ přerušení jsou rychlé instrukce, které trvají 1 až 2 cykly, které slouží k ukládání přijatých znaků do fronty (viz. dále)

Během tohoto přerušení přichází „hlavní“ přerušení, které odesílá data a synchronizační impulsy.

Rozlišení obrazovky je 256 x 256 pixelů. Zařízení umožňuje zobrazovat data na obrazovce ve čtyřech barvách. Proto je každý pixel na obrazovce reprezentován dvěma bity v paměti. Jeden Byte zobrazený na obrazovce se tak skládá ze dvou Byte vybraných z paměti.

2 Jeden cyklus procesoru se rovná periodě kmitů krystalu procesoru, při 12 MHz krystalu je perioda kmitů ~83 ns.

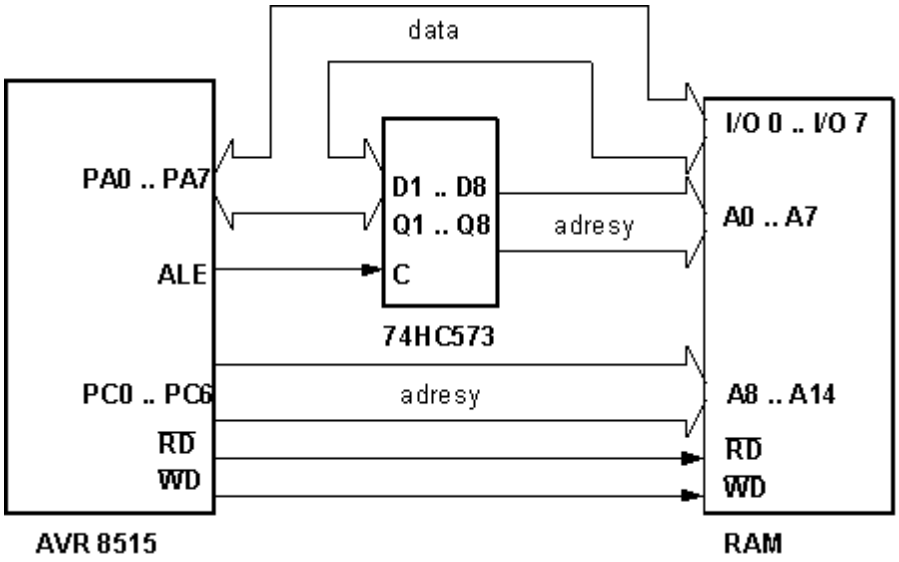


obr. 5.Způsob uložení barvy jednoho pixelu v obrazovce

Jeden řádek na obrazovce je potom reprezentován 32 Byte, což je 64 Byte v paměti. Data pro vykreslení celé obrazovky potom zabírají 16 384 Byte[3]. Takové množství dat by se do mikrokontroléru Atmega 8515 nevlezlo, proto je k němu připojena externí paměť, která slouží pro uložení zobrazovaných dat.

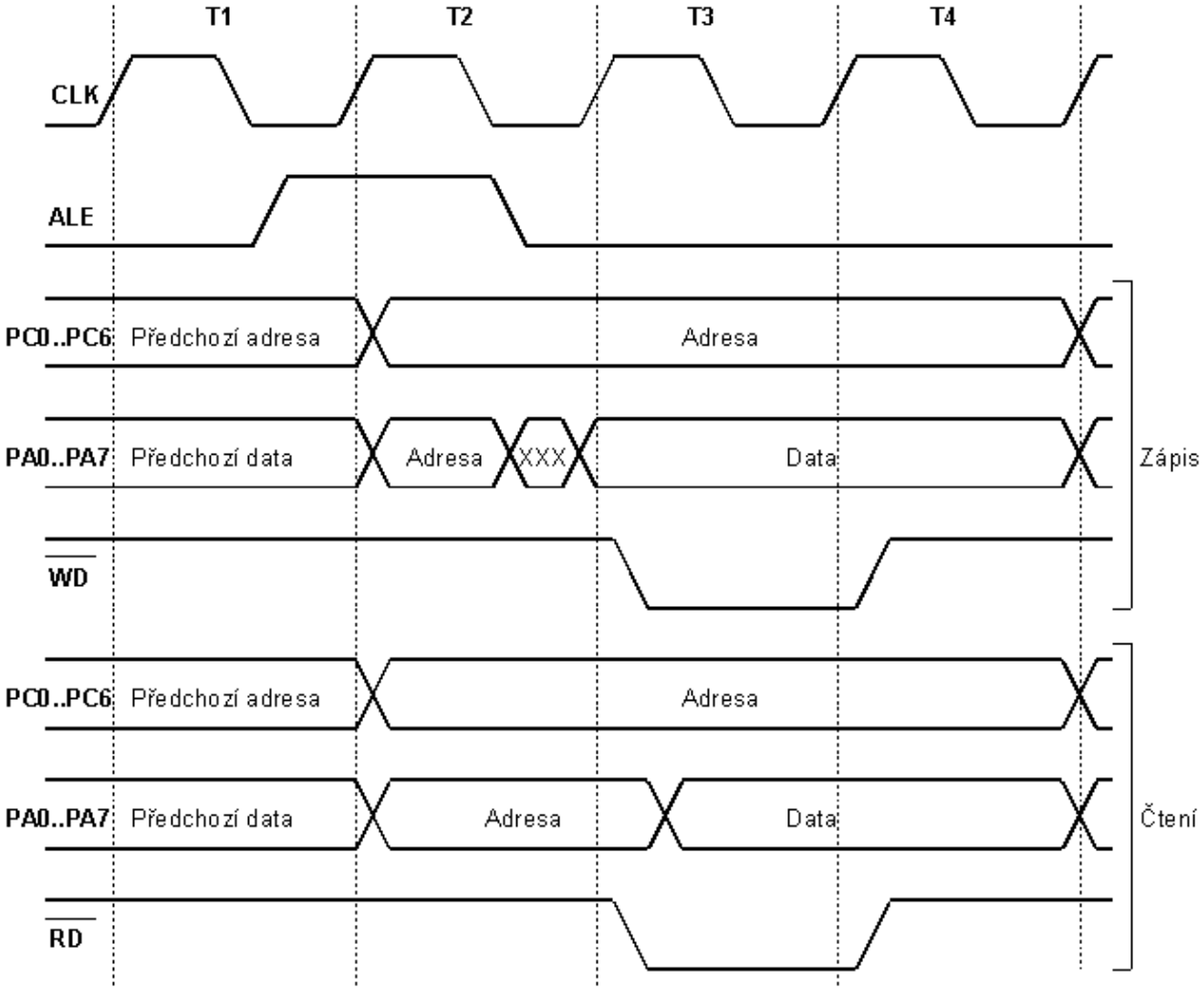
Z ekonomických důvodů byla vybrána 32 KB paměť. Tato paměť potřebuje pro adresování paměťového prostoru 15 vývodů a pro přenos dat dalších 8. Takové řešení by neúměrně zabíralo mnoho vývodů mikrokontroléru, proto se využívá záchytný paměťový registr typu D, který umožňuje 8 dolních bitů adresy uložit do tohoto registru a potom tyto piny může mikrokontrolér využít pro práci s daty.

3 Větší použitá paměť umožnila ukládat zobrazovaná data pro dva nezávislé obrazy.



obr. 6.Schematické znázornění připojení externí paměti

Na portu PA se nejdříve objeví spodní část adresy. Následuje ALE impuls, který zapíše spodní bity adresy do D registru (IO 74HC573). Na portu PC se nastaví horní část adresy. Nyní se můžou nastavit data při zápisu a zapsat impulsem WD nebo pomocí impulsu RD číst data. Uvedené zapojení je převzato z [4.]



obr. 7.Časový diagram zápisu do externí paměti *(zdroj [4.])*

Na jednom řádku se zobrazuje 256 pixelů během 44μs. To znamená, že vždy po 170 ns se musí odeslat další pixel. Což odpovídá zhruba dvěma instrukčním cyklům. Mikrokontrolér není schopný takto rychle odesílat data, proto se data vysílají ze dvou posuvných registrů 74HC166.

Z důvodu úspory vývodů a času, jsou tyto registry připojeny na stejnou sběrnici jako externí paměť. Výhodou je, že mikrokontrolér sám vygeneruje signálem WD impuls pro zápis do posuvných registrů. Během ukládání dat, když se zrovna nezobrazuje, jsou registry trvale nulovány a neodesílají žádná data na obrazovku.

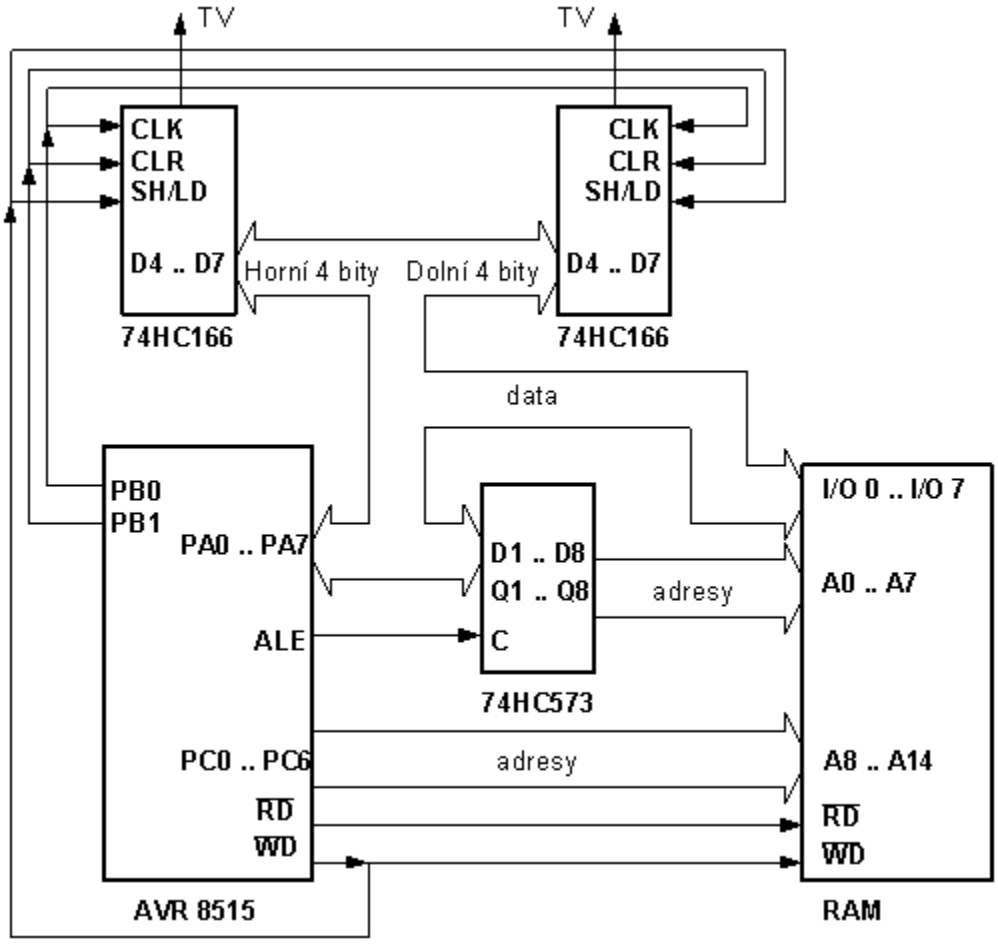
Signál pro vstup hodin posuvných registrů se vytváří přímo v mikrokontroléru. Jedná se o časovač T0, který je nastaven tak, aby na pinu PB.0 vytvářel obdélníkový signál s frekvencí 6MHz.

Povolování vysílání posuvných registrů je připojeno na pin PB.1.

Posuvné registry vysílají postupně bity od nejvyššího po nejnižší. Vzhledem k tomu, že se z každého registru odesílají pouze 4 bity, je na vstupy D7 … D4 jednoho registru přiveden signál z PA7 … PA4 a na vstupy druhého registru D7 … D4 přiveden signál PA3 … PA0. Ostatní vstupy posuvných registrů jsou uzemněny.

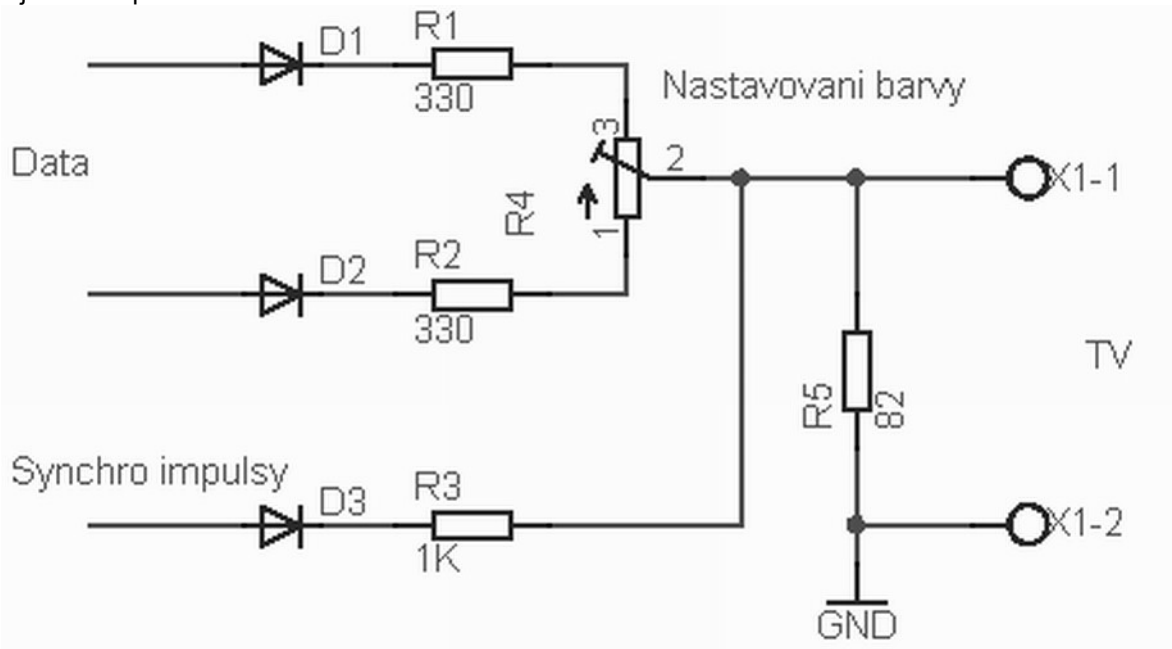
V okamžiku kdy se mají data odesílat na obrazovku tak mikrokontrolér vybere data z paměti. Uloží data do paměti (současně i do registrů) a zvedne hodnotu registru ukazatele do paměti.

Celá tato posloupnost trvá právě 8 instrukčních cyklů, což je doba 4 pixelů na obrazovce.



obr. 8.Schématické znázornění připojení výstupních posuvných registrů

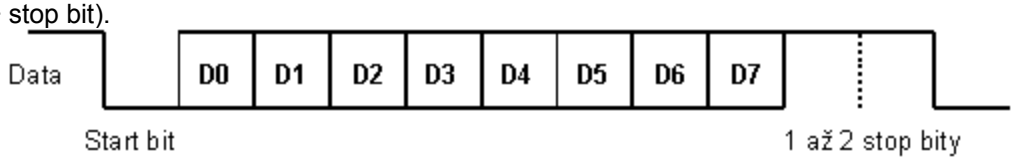
Výsledný video signál se musí složit ze synchronizačních impulsů a dat tak, aby při synchronizačních impulsech bylo na výstupu napětí 0V, při vykreslování černého bodu 0,33V a při bílém bodu 1V. Výsledný signál je složen pomocí rezistorů a diod.



obr. 9.Převodník pro výstupní video signál.

4. Příjem dat

Pro komunikaci se TV terminálem je využita sériová asynchronní komunikace. Zařízení pracuje v duplexním[4] režimu. Pro odeslání jednoho Bytu informace je třeba odeslat minimálně 10 bitů (start bit + 8bitů informace + stop bit).



obr. 10.Asynchronní sériový přenos

Parametry komunikace:
- rychlost: 9600 Baud[5]
- počet bitů: 8
- parita: bez parity
- počet stop bitů: 1 až 2, doporučovány 2
- konektor: CANNON 9 M

Zařízení může fungovat pouze textově, to znamená, že přijaté 8 bitové číslo je převedeno na znak podle ASCII[6] tabulky. Ze speciálních znaků jsou zpracovávány znaky uvedené v tab. 1.

Číslo znaku		Název znaku	Funkce
DEC	HEX		
8	8	BS	Vymaže poslední znak
9	9	TAB	Zarovná kurzor na nejbližší vyšší pozici s násobkem 6
10	A	LF	Posune kurzor na další řádek
12	C	FF	Vymaže obrazovku a nastaví kurzor na počáteční pozici
13	D	CR	Posune kurzor na začátek dalšího řádku

tab. 1.Speciální znaky

V textovém režimu má jeden znak velikost 6 x 8 pixelů. Na řádek je možné umístit až 42 znaků Obrazovka pojme celkem 32 textových řádků. Textová pozice je číslována od 0 – pozice [0,0] je v levém horním rohu.

Zařízení umožňuje také grafické funkce volané pomocí tzv. Esc sekvencí. Grafické rozlišení je 256 x 256 pixelů, souřadnice [0,0] je v levém horním rohu. Esc sekvence byly zvoleny speciálně pro toto zařízení a neodpovídají žádnému standardu.

Kód	Parametry	Funkce
[ESC] + [TAB]		Zařízení umožňuje díky dostatečné paměti mít uloženy dvoje data pro obrazovku. Tento kód přepne zobrazení a zápis na nyní nezobrazovanou část externí paměti.
[ESC]+[B]	[P] .. '0','1'	Pokud [P] = '0' nastaví znaky na neprůhledné (znak kolem se vytiskne na černý podklad 6 x 8 pixelů). Pokud [P] = '1' nastaví znaky na průhledné (znak přetiskne své pozadí).
[ESC]+[C]	[C] .. 0-255	Podle bitů [C] nastaví barvu se kterou u vybraných funkcí a u znaků provádí logický součin jednotlivých bitů ukládaného Byte do paměti.
[ESC]+[G]	[X] .. 0-255 [N] .. 0-255 N*[Y] .. 0-255	Tato funkce je připravena pro osciloskop. Na pozici [X,Y_0] začne vykreslovat spojitý graf tak, že pro každou zadanou vertikální pozici [Y_N] vykreslí čáru spojující body [X,Y_N] a [X-1,$Y_{N=1}$] *(pokud X+N>255 neprovede nic; použije aktuální barvu).*
[ESC]+[H]	[X] .. 0-15	Podle hodnoty [X] nastaví celkové horizontální posunutí obrazu na obrazovce - nastaví dobu prvního zatemňovacího impulsu *(podle spodních 4 bitů čísla [X])*

4 V jednom časovém okamžiku se mohou data přenášet mezi komunikujícími zařízeními dvěma směry.
5 Baud = počet odeslaných bitů za sekundu
6 Pozn.: zařízení neobsahuje kompletní znakovou sadu ASCII

Kód	Parametry	Funkce
[ESC]+[L]	$[X_1]$ .. 0-255 $[Y_1]$ .. 0-255 $[X_2]$ .. 0-255 $[Y_2]$ .. 0-255	Vykreslí úsečku z bodu $[X_1,Y_1]$ do bodu $[X_2,Y_2]$ *(použije aktuální barvu)*
[ESC]+[M]	$[A_H]$..128-255 $[A_L]$ .. 0-255 [N] .. 0-255 N*[D] .. 0-255	Od adresy externí paměti $[A_L,A_H]$ uloží postupně [N] Byte dat $[D_0$ .. $D_N]$ *(adresa se automaticky inkrementuje; pokud je [N] rovno nule, příjme se 256 hodnot).*
[ESC]+[O]	[Y] .. 0-15	Podle hodnoty [Y] nastaví celkové Vertikální posunutí obrazu na obrazovce - nastaví počet prvních zatemňovacích řádků *(podle spodních 4 bitů čísla [Y]).*
[ESC]+[P]	[X] .. 0-255 [Y] .. 0-255	Vykreslí bod na pozici [X,Y] *(použije aktuální barvu).*
[ESC]+[Q]		Softwarově restartuje zařízení.
[ESC]+[R]	[X] .. 0-255 [Y] .. 0-255 $[d_X]$ .. 0-255 $[d_X]$ .. 0-255	Vykreslí obdélník z bodu [X,Y] o horizontální délce $[d_x]$ a vertikální délce $[d_y]$ *(pokud by se měl obdélník vykreslit mimo obrazovku nevykreslí jej, použije aktuální barvu).*
[ESC]+[S]		Vymaže obrazovku a nastaví pozici kursoru na [0,0]
[ESC]+[T]	[X] .. 0-255 [Y] .. 0-255 $[d_x]$ .. 0-255	Vykreslí vodorovnou čáru z bodu [X,Y] o délce $[d_x]$ *(při $(X+d_x)>255$ čáru nevykreslí; použije aktuální barvu)*
[ESC]+[U]	[X] .. 0-255 [Y] .. 0-255 $[d_Y]$ .. 0-255	Vykreslí svislou čáru z bodu [X,Y] o délce $[d_Y]$ *(při $(Y+d_Y)>255$ čáru nevykreslí; použije aktuální barvu)*
[ESC]+[V]	[P] .. '1','2'	Zařízení umožňuje díky dostatečné paměti mít uloženy dvoje data pro obrazovku. Tento kód přepne zobrazení dat na část 1. ([P] = '1'), nebo na část 2. ([P] = '2')
[ESC]+[W]	[P] .. '1','2'	Zařízení umožňuje díky dostatečné paměti mít uloženy dvoje data pro obrazovku. Tento kód přepne zápis dat na část 1. ([P] = '1'), nebo na část 2. ([P] = '2')
[ESC]+[X]	[X] .. 0-41	Nastaví horizontální textovou pozici na [X]
[ESC]+[Y]	[Y] .. 0-31	Nastaví vertikální textovou pozici na [Y]
[ESC]+[Z]	[X] .. 0-234 [Y] .. 0-226 [Z] .. ' '-'~'	Vykreslí 4x větší znak [Z] na pozici [X,Y] (*použije aktuální barvu).*

tab. 2.Esc sekvence

5. Odesílání dat

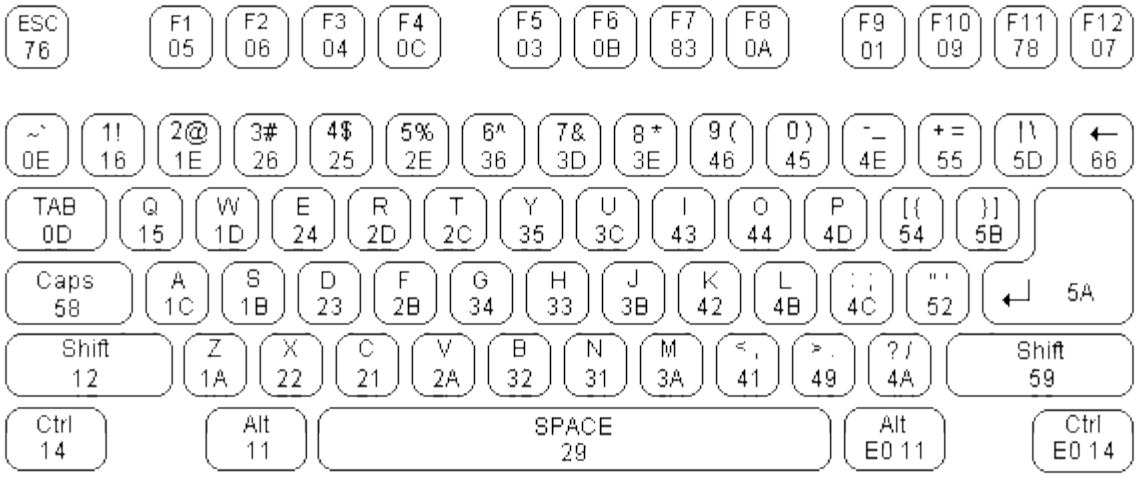
Pro zadávání výstupních dat z TV terminálu slouží klávesnice kompatibilní s IBM AT, která se obvykle připojuje k PC. Požadavek byl, aby veškerá komunikace s cílovým zařízením probíhala pomocí sériové komunikace. Proto bylo třeba použít převodník mezi scan kódy klávesnice a kódem ASCII.

Softwarová implementace převodníku do ATmega 8515 již nebyla možná. Vzhledem k časové vytíženosti nezvládal mikrokontrolér v reálném čase obsluhovat klávesnici. Proto byl modul doplněn o samostatný mikrokontrolér ATtiny 13, jehož úkolem je snímání scan kódů klávesnice a jejich převod na ASCII kód a odesílání pomocí sériové komunikace.

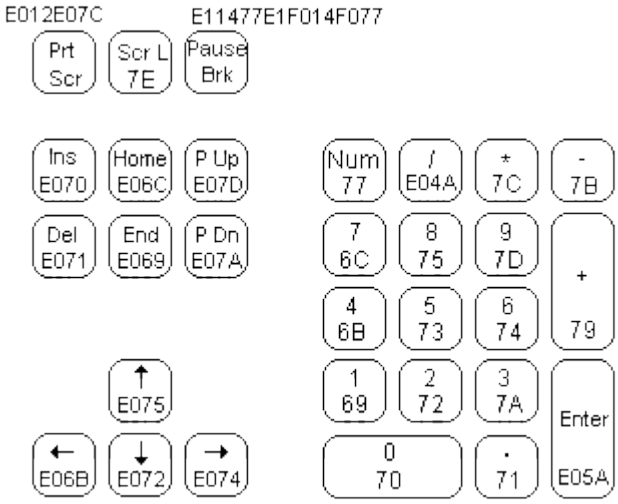
Obvod pro obsluhu klávesnice není nijak vázán na obsluhu TV obrazovky a lze jej úplně vynechat.

Program pro ATtiny 13 jsem neprogramoval, ale převzal jsem jej z oficiálních stránek firmy Atmel, kde byl tento problém již vyřešen [1.] případně obdobný program pro mikrokontrolér HC705JA můžeme nalézt na [2.]

Hlavní funkcí programu mikrokontroléru je převod scan kódu klávesnice na ASCII kód, komunikační části byly mírně upraveny.



obr. 11.Mapa scan kódů klávesnice *(zdroj [2.])*

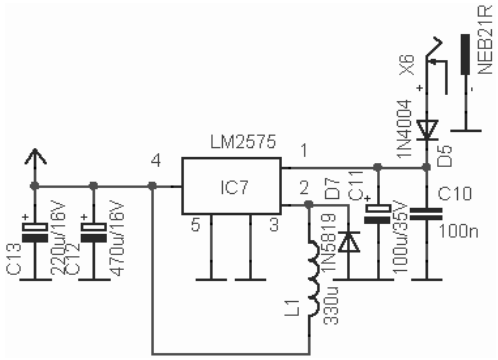


obr. 12.Mapa scan kódů druhé části klávesnice *(zdroj [2.])*

Vstup klávesnice je realizován přes konektor DIN6 mini na který je možné připojit redukci na konektor DIN 5. Výstup ze zařízení je realizován přes stejný konektor, který se využívá pro vstup dat.

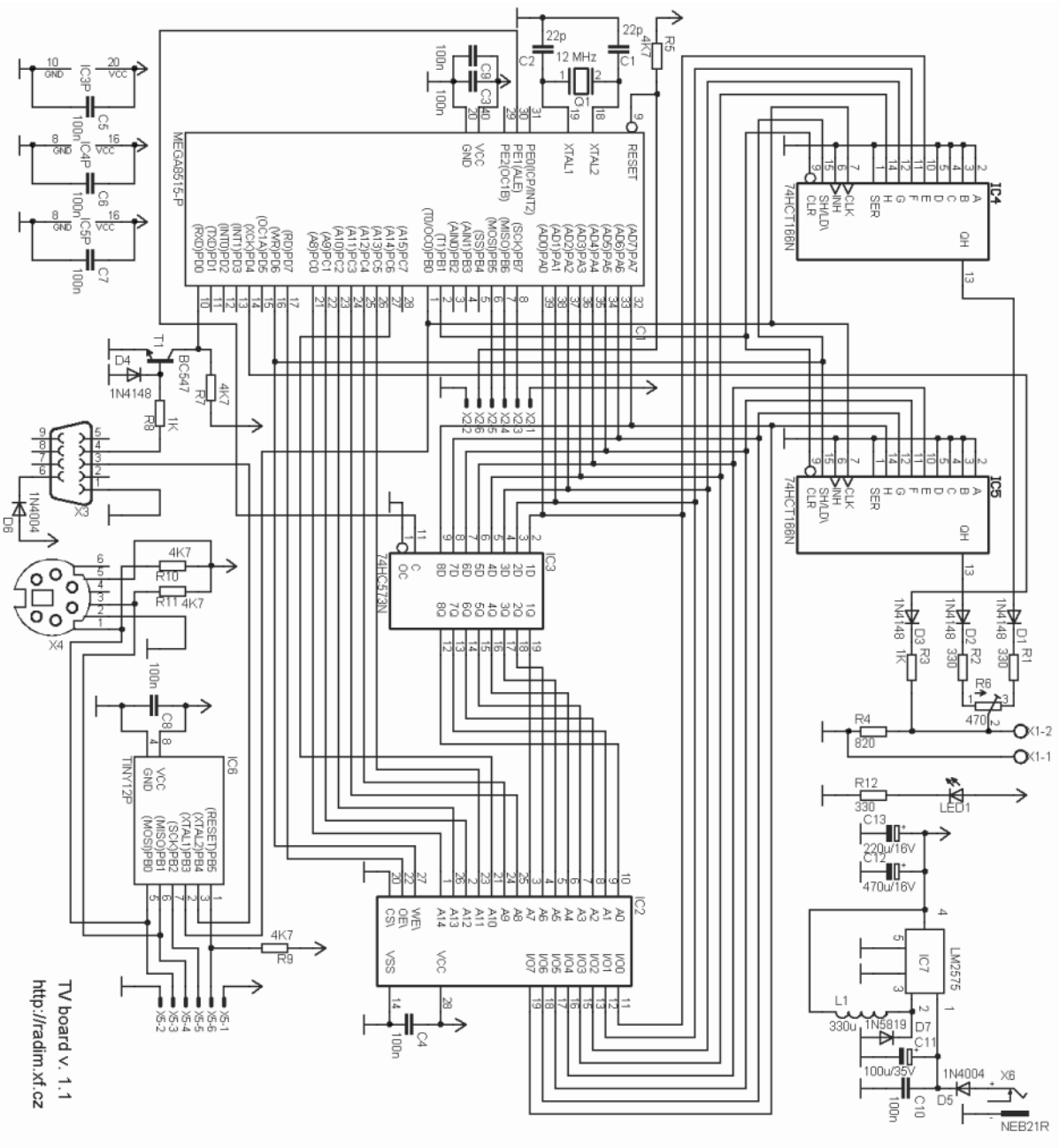
6. Napájení

Napájecí napětí k zařízení se přivádí přes konektor NEB 21 R. Velikost napájení by měla být 9 … 15V stejnosměrných. Maximální proudový odběr by neměl přesáhnout 0,3 A. Ke stabilizaci napětí na +5V se využívá LM 2575. Jedná se o spínaný zdroj, tento obvod má proto menší ztrátový výkon než klasické stabilizátory řady 78xx.



obr. 13.Schéma stabilizátoru napětí

7. Schéma zapojení



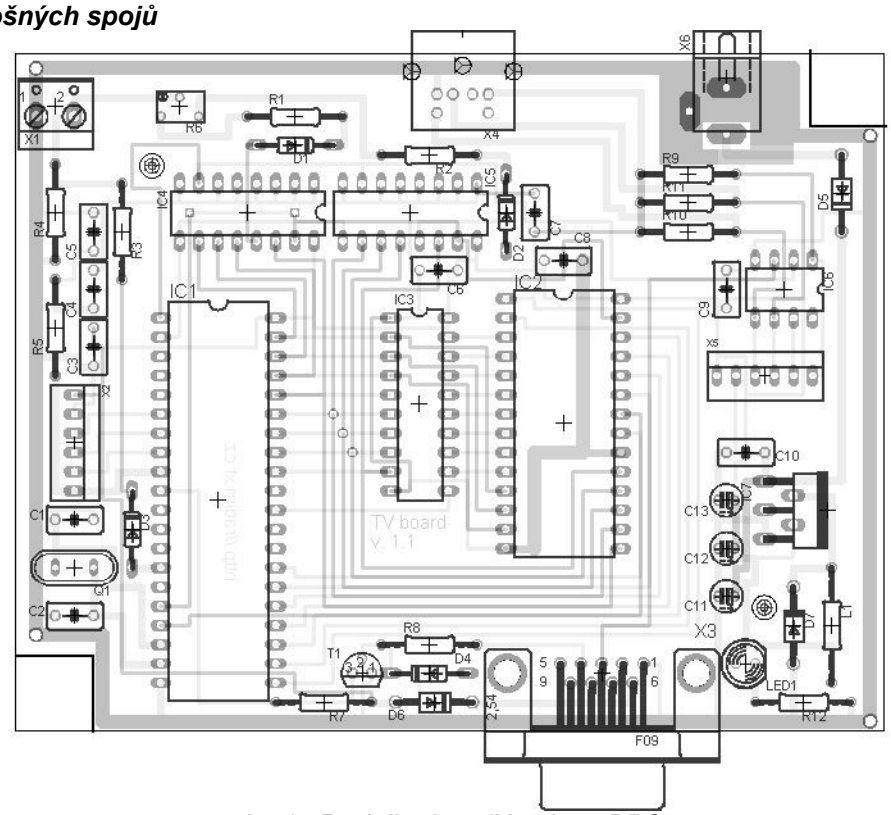
obr. 14.Schéma zapojení

8. Seznam použitých součástek

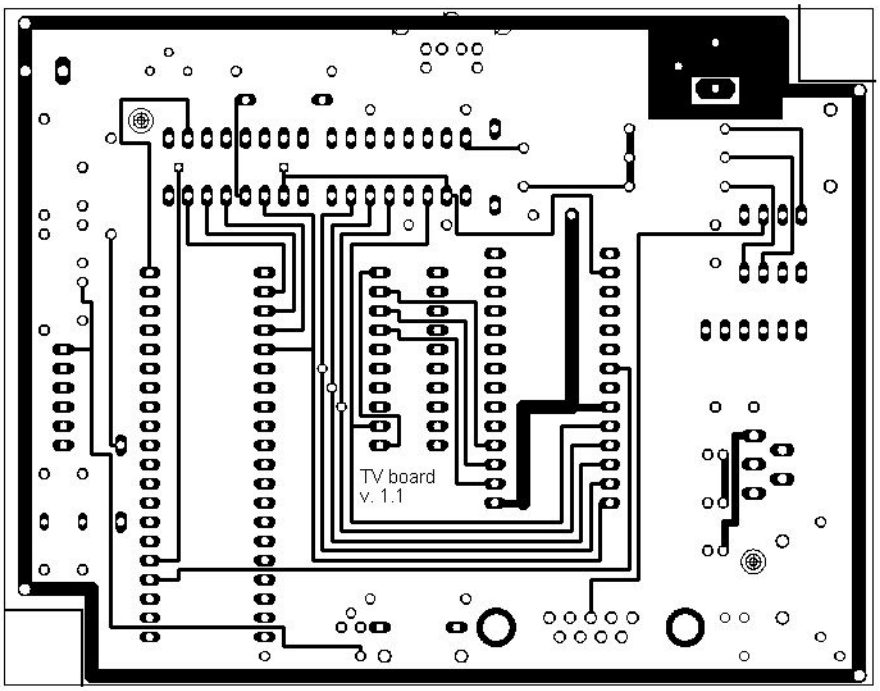
Označení součástky	Hodnota	Označení součástky	Hodnota
C1, C2	22pF	L1	330μH
C3, C4, C5, C6, C7, C8, C9, C10	100nF	LED1	LED 5mm
C11	100μF/35V	Q1	12MHz
C12	470μF/16V	R1, R2, R12	330Ω
C13	220μF/16V	R3, R7	1KΩ
D1, D2, D3, D4	1N4148	R4	820Ω
D5, D6	1N4004	R5, R8, R9, R10, R11	4K7Ω
D7	1N5819	R6	470Ω
IC1	ATmega 8515	T1	BC 547
IC2	AMIC A6253082	X1	W237-102
IC3	74HC573	X2, X5	PFH 02-06-P
IC4, IC5	74HC166	X3	CANNON 9 M
IC6	ATtiny13	X4	DIN 6 mini F DPS
IC7	LM2575	X6	NEB21R

tab. 3.Seznam použitých součástek TV terminál

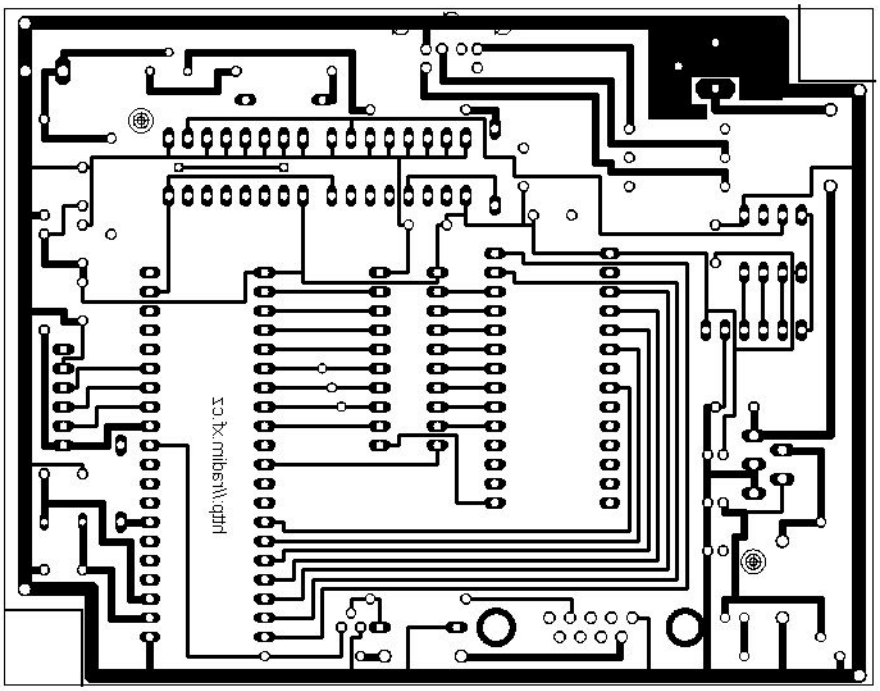
9. Deska plošných spojů



obr. 15.Rozložení součástek na DPS



obr. 16.Pohled na DPS ze strany součástek



obr. 17.Pohled na DPS ze strany spojů

10.Možnosti využití modulu TV terminálu

V současné době si mnoho lidí pořizuje nové televizory ať již s plochou obrazovkou nebo připravené pro příjem digitálního vysílání. Tím amatéři dostávají k dispozici řadu televizorů, které se již nepoužívají. Tyto přístroje lze použít jako velkoplošné zobrazovače pro různá amatérská zařízení.

Vzrůstající možnosti amatérů používat ve svých konstrukcích různé typy mikrokontrolérů přinášejí možnost využít tento modul jako výstupní zobrazovací jednotku pro zařízení s potřebou zobrazovat větší objem dat. Lze si představit tento televizní modul jako výstupní část například u řady měřících zařízení jako například složitější voltmetry, měřiče kapacity, indukčnosti, záznamníky dat, měřiče charakteristik atd.

Celkové nízké náklady na zhotovení tohoto modulu otevírají mnoha amatérům možnost doplnit svůj výrobek a efektní způsob zobrazení dat.

III. Modul osciloskopu

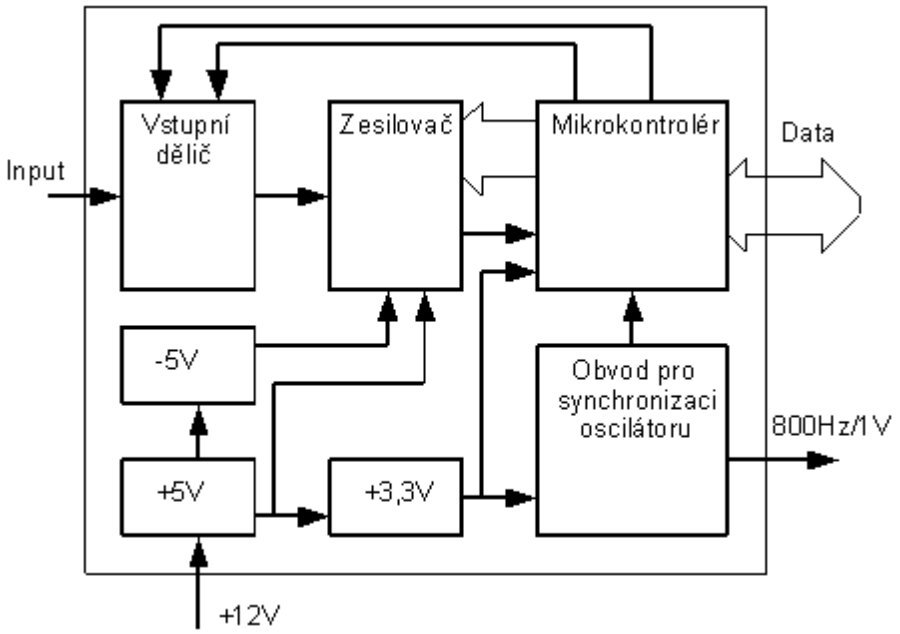
1. Úvod

Cílem zde popsaného osciloskopu je zobrazovat průběhy napěťových signálů s frekvencí až 50 kHz. Osciloskop byl zároveň vyvíjen pro komunikaci s výše popsaným TV terminálem. Naměřená data se odesílají přes sériovou linku do připojeného zařízení.

Digitální osciloskop umožňuje naměřená data zaznamenávat a také zobrazit nejen grafický průběh, ale i konkrétní hodnotu napětí u dané křivky.

Osciloskop byl koncipován s cílem jednoduché konstrukce, přesto s důrazem na kvalitu měřených dat s minimální chybou měření.

Obvod by bylo možné připojit i k počítači. Nevýhodou je, že v současné době není vytvořen žádný program pro zpracování přenesených dat.



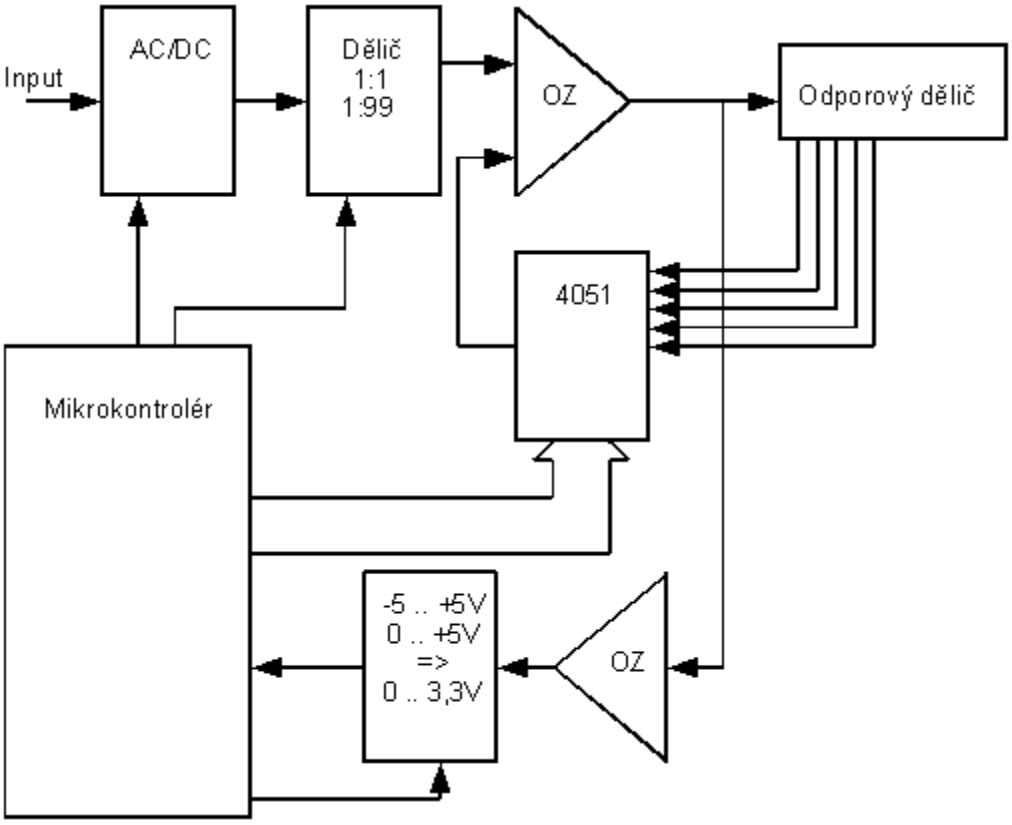
obr. 18.Blokové schéma jednotlivých částí osciloskopu

2. Vstupní obvod

Vstupní obvod patří mezi nejdůležitější částí osciloskopu. Jeho úkolem je připravit měřený signál pro AD převodník. Použitý AD převodník mikrokontroléru je schopný měřit napětí v rozsahu 0 až 3,3 V (viz. dále). Proto je potřeba napětí zesílit, či podělit tak aby co nejlépe vyhovovalo zvolenému rozsahu a tím dosažení co největší přesnosti měření.

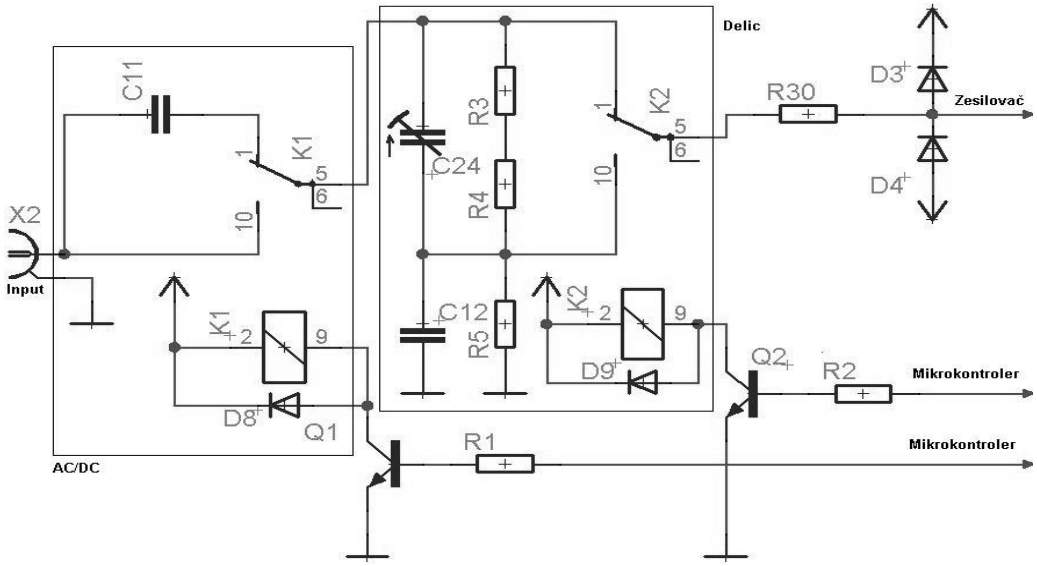
Řešení některých prvků vstupního děliče vychází z článku [3.]. Oproti zde uváděnému zapojení je jinak zapojen vstupní multiplexer a celá vstupní část je zjednodušena.

Na vstupu obvodu je vazební kondenzátor, kterým se může na vstupu osciloskopu oddělit stejnosměrná složka signálu. Zařazení či nezařazení kondenzátoru do obvodu řídí mikrokontrolér.



obr. 19.Blokové schéma vstupní obvod osciloskopu

Další částí je vstupní dělič neboli atenuátor. Úkolem děliče je při napětí s amplitudou vyšší než 2 V podělit vstupní signál 1:99. Díky tomu lze osciloskopem měřit střídavé napětí s amplitudou až 200 V. Volba dělení signálu 1:1 či 1:99 se opět ovládá programově mikrokontrolérem. Vstupní dělič je kmitočtově kompenzován. Pro nastavení kmitočtové kompenzace je využit otočný kondenzátorový trimr C24 pomocí něhož lze kmitočtovou kompenzaci dostavit s konkrétním kusem připojené osciloskopické sondy. Celkové zapojení vstupního děliče udává vstupní parametry osciloskopu – 1 MΩ / 35 pF, což je srovnatelné s některými staršími typy analogových osciloskopů.



obr. 20.Detailní schéma vstupního děliče

Následuje zesilovač signálu. Slouží k zesílení malých signálů. Zesilovač má proměnné zesílení, které lze nastavit mikrokontrolérem. Základem je odporový dělič, který je připojen ve zpětné vazbě operačního

zesilovače OZ. Odporový dělič obsahuje pět trimrů na jejichž běžcích lze nastavit takovou úroveň signálu, aby zesílení operačního zesilovače bylo 50, 20, 10, 5, 2 a 1. K přepínání úrovní je určen analogový multiplexer typu 4051.Operační zesilovač je zapojen jako běžný neinvertující zesilovač.

V zapojení [3.] byl multiplexer zapojen tak, že přepínal jednotlivé rezistory k výstupu operačního zesilovače. Nevýhodou tohoto zapojení je, že při nastavování zesílení operačního zesilovače se uplatňuje také vnitřní odpor multiplexeru. Poroto jsem volil jiné zapojení, kdy je multiplexer zapojen mezi odporový dělič a vstup operačního zesilovače. Odpor multiplexeru je v tomto zapojení zanedbatelný proti vstupnímu odporu operačního zesilovače.

Patřičně upravený signál jde do oddělovacího operačního zesilovače. Tento operační zesilovač není pravým sledovačem neboť má malé zesílení. Upravuje signál tak, aby AD převodník mikrokontroléru již měřil reálné hodnoty a nebylo nutné měřená data programově přepočítávat. Toto zesílení lze doladit pomocí trimru ve zpětné vazbě zesilovače.

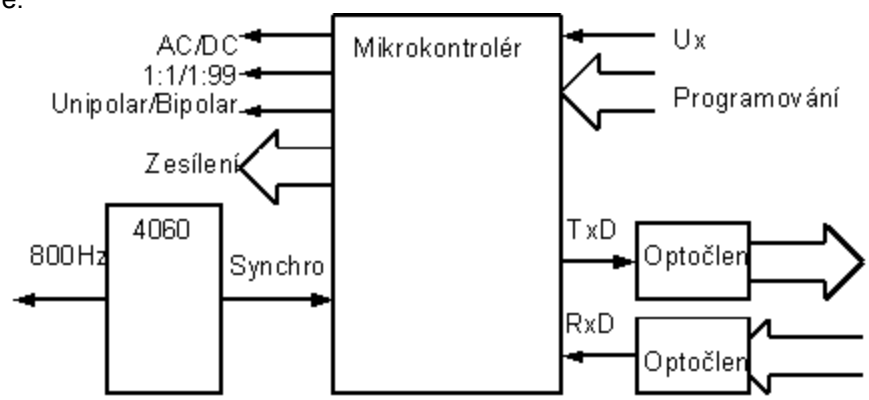
Posledním prvkem je obvod pro posouvání napěťových úrovní. Jak již bylo zmíněno v úvodu je mikrokontrolér schopen měřit napětí v rozsahu 0 až 3,3V. Proto je potřeba aby obvod napětí v rozsahu -5 až 5V zmenšil a posunul. Tento obvod rovněž nabízí možnost měřit data v unipolární či bipolární variantě. Standardní je bipolární režim, při kterém se zobrazuje jak kladná, tak záporná část signálu. Pokud chceme ovšem osciloskop použít pro sledování číslicových signálů, nepotřebuje zobrazovat zápornou část dat. Proto je lepší, když osciloskop je schopný oříznout zápornou část dat (pokud ji data mají) a nulu posunou níž a tím zvětšit rozlišení dat. Přepínání mezi bipolárním a unipolárním režimem opět řídí mikrokontrolér pomocí spínacího tranzistoru T1.

3. Řídící obvod

V současné době je na trhu mnoho mikrokontrolérů od různých výrobců. Pro osciloskop jsou nejdůležitější vlastnosti AD převodníku daného mikrokontroléru. Požadované rozlišení (8 bitů) splňují téměř všechny. Kritickým parametrem je však rychlost převodu. Při výběru z dostupných typů mikrokontrolérů se jako nejlepší ukázal typ MC9S08QG8 od firmy Freescale.

Na jedno měření AD převodníku u tohoto typu je třeba 17 bus instrukcí (při osmibitovém rozlišení a kontinuálním převodu). Jedna bus instrukce je dána dvěma impulsy oscilátoru. Pro buzení mikrokontroléru jsem zvolil vnitřní oscilátor mikrokontroléru, který je potřeba na začátku nastavit na hodnotu 17 MHz. Tím je získána minimální doba převodu 2 μs.

Toto mírně překračuje povolované parametry avšak při osmibitovém převodu pracuje AD převodník naprosto spolehlivě.



obr. 21.Periferie připojené k mikrokontroléru

Mikrokontrolér dále slouží k ovládání zvolených módů při měření. Spíná pomocí tranzistorů relé, která nastavují režimy AC/DC, 1:1/1:99, Unipolar/Bipolar (viz. předchozí kapitola). Mikrokontrolér zajišťuje také sériovou komunikaci. Při sériové komunikaci jsou data oddělena pomocí optočlenů, aby se galvanicky oddělilo celé zařízení od dalších přístrojů.

4. Nastavení vnitřního oscilátoru řídícího obvodu

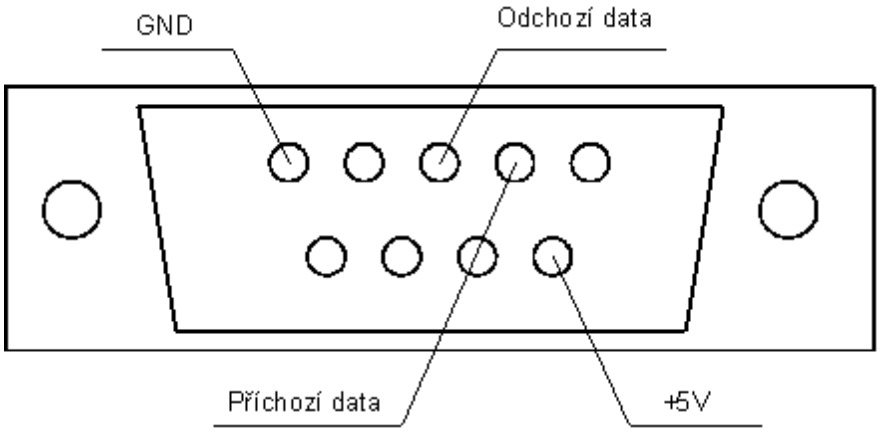
Po spuštění programu mikrokontroléru se frekvence vnitřního oscilátoru nastaví na 17 MHz. K tomu slouží synchronizační funkce, která postupně nastavuje trimovací registr a spouští zkušební smyčku. Úkolem zkušební smyčky je provést daný počet instrukcí a během nich počítat příchozí impulsy, které jsou generovány pomocí externího integrovaného obvodu typu 4060 s připojeným krystalem.

Výpočtem vyšlo a empiricky potvrzeno bylo, že by se mělo provést 2360 cyklů aby se oscilátor dostal na frekvenci 17 MHz. Na přesné nastavení trimovacího registru má funkce 50 opakování, během kterých by se měl trimovací registr nastavit na přesnou hodnotu.

Integrovaný obvod 4060 se zároveň používá pro generování kontrolního kmitočtu 800Hz, který je vyveden na výstupní svorku. Tento kontrolní kmitočet se může použít pro kontrolu funkčnosti osciloskopu a pro případné nastavování kmitočtové kompenzace s připojenou osciloskopickou sondou.

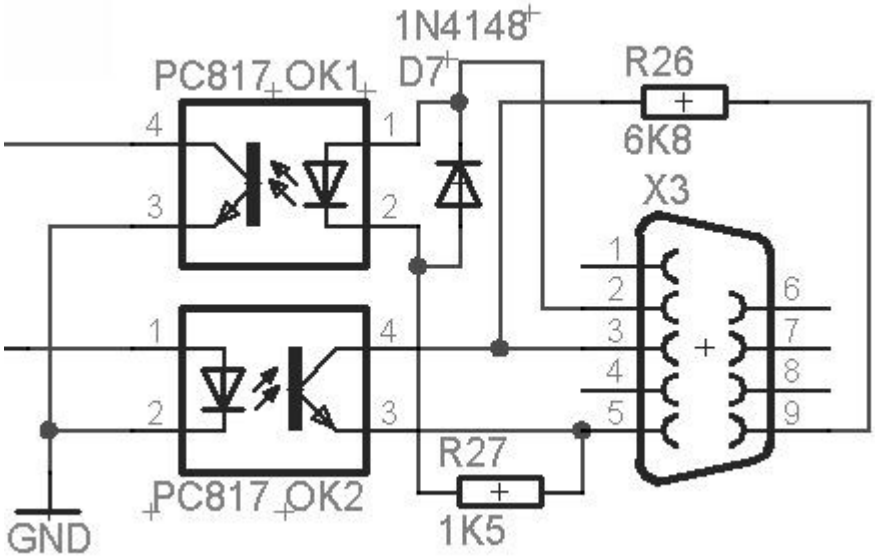
5. Sériová komunikace

Naměřená data i data sloužící k ovládání osciloskopu se přenášejí pomocí sériové linky. Pro ovládání osciloskopu stačí přenášet pouze znaky v ASCII kódu. Jednotlivé znaky jsou uvedeny v helpu, který je součástí programu osciloskopu (viz. následující kapitola).



obr. 22.Zapojení datového konektoru u osciloskopu

Parametry sériové komunikace jsou shodné s TV terminálem. Aby byla zajištěna funkce výstupního oddělení sériové komunikace je potřeba zajistit napájení výstupního optočlenu. Proto je nezbytné aby na vývod číslo 9 na konektoru CANNON bylo přivedeno napětí. Pokud je osciloskop připojen k TV terminálu, je na vývod 9 přivedeno +5 V. U jiných zařízení musíme toto zajistit například softwarově. Pro osobní počítače IBM je potřeba nastavit správnou polaritu signálu označovaného jako RING.



obr. 23.Detail galvanického oddělení sériové komunikace

6. Módy měření

Osciloskop umožňuje nastavovat různé módy měření. Módy se mění pomocí znaků, které přichází po sériové lince. Změnu a nastavení jednotlivých módů udává následující tabulka:

<table>
<tr><th>Klávesa</th><th>Funkce</th></tr>
<tr><td>← →</td><td>Nastavení časové základny. Klávesa ← snižuje dobu jednoho dílku, klávesa → zvyšuje dobu dílku.
Jednotlivé rozsahy časové základny:
50 μs / dílek; 100 μs / div; 200μs/div; 500μs/div; 1ms/div; 2ms/div; 5ms/div; 10ms/div; 20ms/div; 50ms/div; 100ms/div; 200ms/div; 500ms/div; 1s/div; 2s/div;</td></tr>
<tr><td>↑↓</td><td>Nastavení zesílení signálu. Klávesa ↓ zvyšuje zesílení signálu, klávesa ↑ zmenšuje zesílení signálu.
Jednotlivé zesílení a nastavení vstupního odporového děliče:
<table>
<tr><th>Bipolární režim V/div</th><th>Unipolární režim V/div</th><th>Zesílení signálu</th><th>Nastavení vstupního děliče</th></tr>
<tr><td>10m</td><td>5m</td><td>50</td><td>1:1</td></tr>
<tr><td>20m</td><td>10m</td><td>25</td><td>1:1</td></tr>
<tr><td>50m</td><td>25m</td><td>10</td><td>1:1</td></tr>
<tr><td>100m</td><td>50m</td><td>5</td><td>1:1</td></tr>
<tr><td>200m</td><td>100m</td><td>2,5</td><td>1:1</td></tr>
<tr><td>500m</td><td>250m</td><td>1</td><td>1:1</td></tr>
<tr><td>1</td><td>500m</td><td>50</td><td>1:99</td></tr>
<tr><td>2</td><td>1</td><td>25</td><td>1:99</td></tr>
<tr><td>5</td><td>2,5</td><td>10</td><td>1:99</td></tr>
<tr><td>10</td><td>5</td><td>5</td><td>1:99</td></tr>
<tr><td>20</td><td>10</td><td>2,5</td><td>1:99</td></tr>
<tr><td>50</td><td>25</td><td>1</td><td>1:99</td></tr>
</table>
tab. 4.Zesílení jednotlivých rozsahů</td></tr>
<tr><td>S</td><td>Nastavování synchronizace. Přepíná mezi režimy No synchro, Synchro Down, Synchro Up. Při zapnuté synchronizaci se začne měřit až v okamžiku, kdy se dosáhne napěťové úrovně definované triggerem.
No synchro – vypnutá synchronizace
Synchro Down – synchronizace na sestupnou hranu
Synchro Up – synchronizace na vzestupnou hranu</td></tr>
<tr><td>U, u, D, d</td><td>Nastavení úrovně tirggeru pro synchronizaci.
U – zvednutí úrovně triggeru o dílek
u – zvednutí úrovně triggeru o 0,2 dílku
S – snížení úrovně triggeru o dílek
s – snížení úrovně triggeru o 0,2 dílku</td></tr>
<tr><td>space</td><td>Zobrazení posledních měřených hodnot k analyzování – tzv. Hold. Při módu Hold se objeví kurzor, kterým lze pohybovat a zobrazovat hodnotu napětí signálu v daném bodě (pouze při bipolárním režimu).</td></tr>
<tr><td>< ></td><td>Klávesy pro posuv kurzoru v módu Hold.</td></tr>
<tr><td>P</td><td>Přepínání mezi bipolárním a unipolárním režimem.</td></tr>
<tr><td>A</td><td>Přepínání mezi režimy AC / DC.</td></tr>
<tr><td>F</td><td>Tato klávesa přepíná mezi kontinuálním režimem Free run a jednotným sejmutím signálu Single shot. Režim Single shot po zmáčknutí startovní klávesy (enter) začne měřit, případně vyčká na synchronizační úroveň.</td></tr>
</table>

Klávesa	Funkce
enter	Spouštění měření v režimu **Free run**.
C	Pro komunikaci s TV terminálem je vhodné mít možnost **vycentrovat obraz**. Při zmáčknutí tlačítka C se zapne režim při kterém je možné vycentrovat obraz na televizoru. Po vycentrování se potvrdí klávesou enter.
H	Zobrazí **Help** který zobrazuje informace o funkčních klávesách.

tab. 5.Funkční klávesy osciloskopu

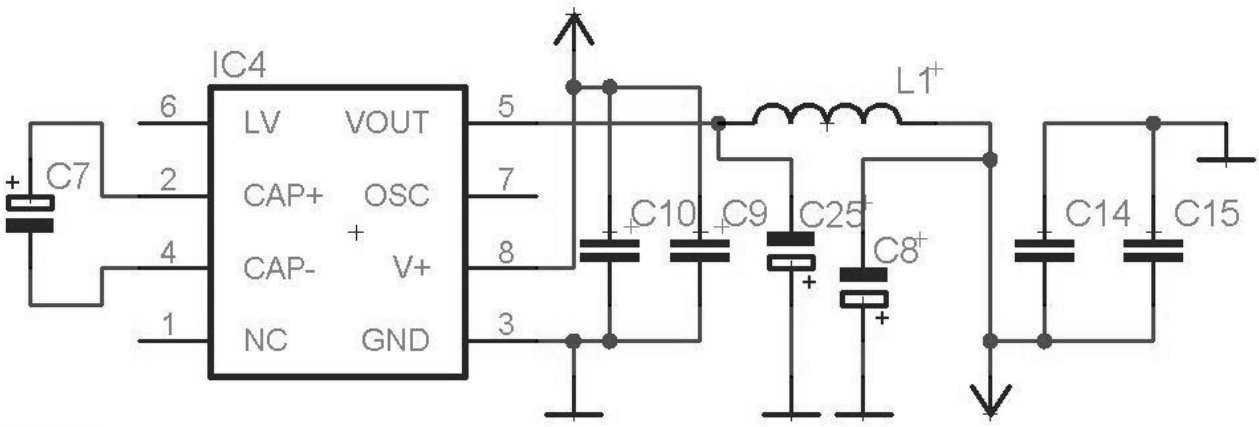
7. Napájení

V zapojení je potřeba celkově tří úrovní napětí. Logické obvody jsou napájeny napětím 3,3 V. Operační zesilovače a multiplexer vyžadují napětí +5 V a -5 V.

Napájecí napětí k zařízení se přivádí přes konektor NEB 21 R. Velikost napájení by měla být 9 … 15 V stejnosměrných. Maximální proudový odběr by neměl přesáhnout 0,2 A. Je nutné, aby byl zdroj napětí galvanicky oddělený od elektrické sítě.

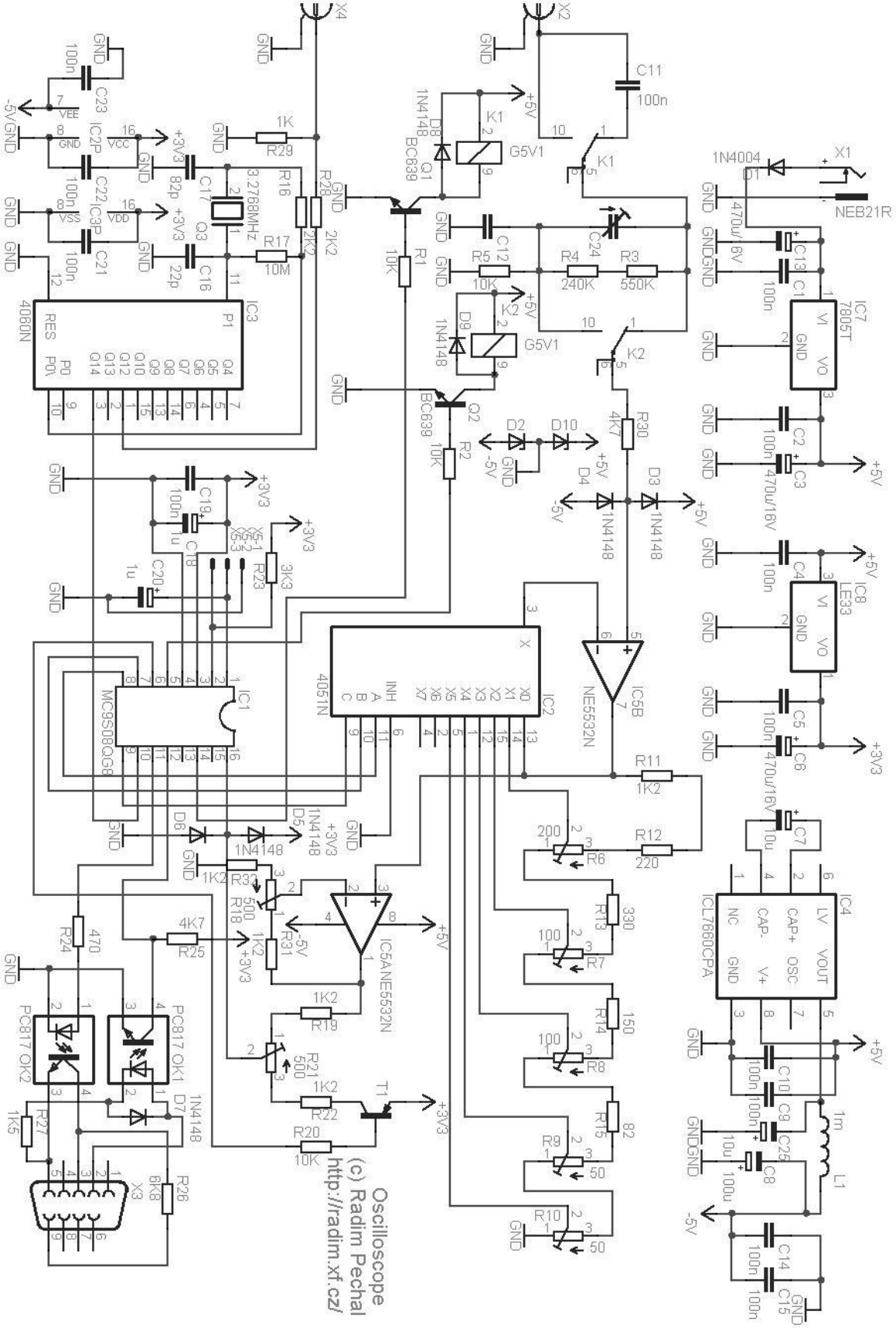
Napájecí napětí se přivede přes ochrannou diodu, aby se zabránilo přepólování napětí. Následně se napájení pomocí stabilizátoru 7805 upraví na +5 V. Toto napětí je potom přiváděno na stabilizátor typu LE33, který vytvoří napětí 3,3 V pro napájení logických obvodů. Napájecí napětí +5 V je také přiváděno na měnič typu ICL7660, který mění polaritu napětí na -5V.

U měniče ICL7660 se objevil problém se zvlněným výstupním napětím. Ukázalo se, že -5 V bylo tak zvlněné, že pronikalo do vstupních operačních zesilovačů a projevovalo se to na měřeném signálu. Proto bylo třeba na výstup měniče připojit filtr tvořený kondenzátorem C8 a tlumivkou L1.



obr. 24.Detail schéma zapojení měniče

8. Schéma zapojení



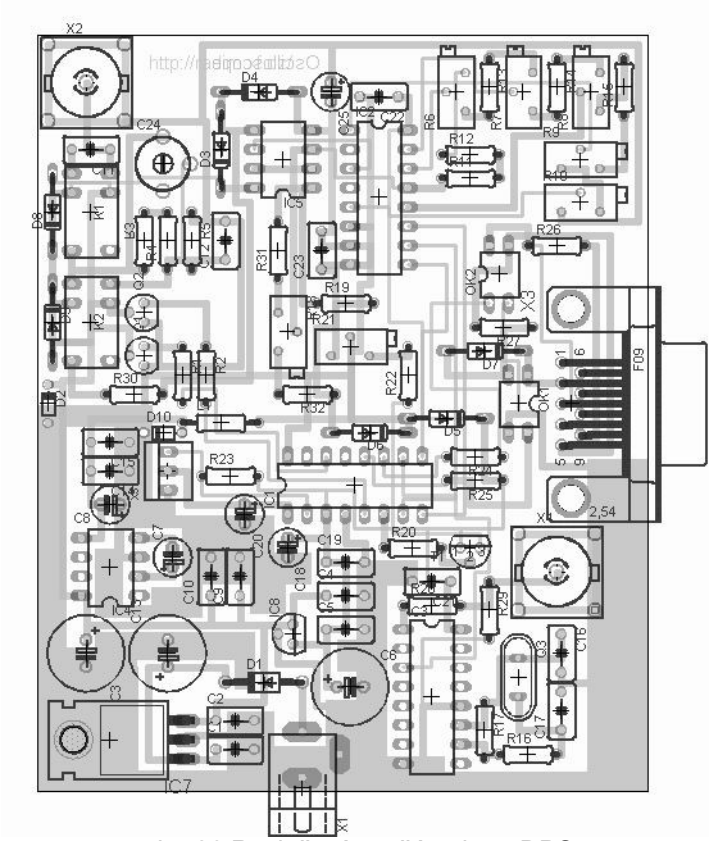
obr. 25.Schéma zapojení osciloskopu

9. Seznam součástek

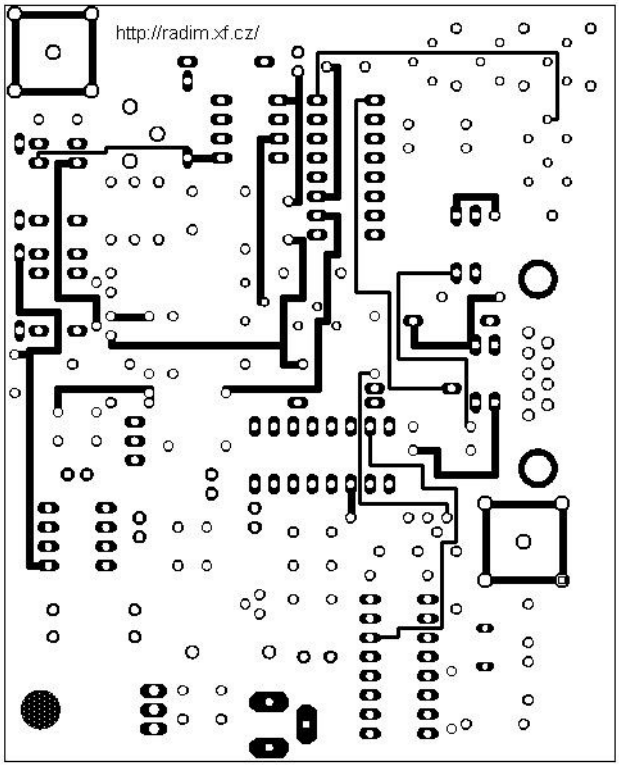
Označení součástky	Hodnota	Označení součástky	Hodnota
C1, C2,C4, C5, C9, C10, C11, C14, C15, C19, C21, C22, C23	100n	R1, R2, R5, R20	10KΩ
C3, C6, C13	470u/16V	R3	550KΩ
C7, C8, C25	10u/16V	R4	240KΩ
C12, C25	100u/16V	R6	200Ω
C16	22p	R7, R8	100Ω
C17	82p	R9, R10	50Ω
C18, C20	1u/16V	R11, R19, R22, R31, R32	1K2Ω
C24	C-TRIMM3-40pF	R12	220Ω
D1	1N4004	R13	330Ω
D2, D10	BZX55/5V1	R14	150Ω
D3, D4, D5, D6, D7, D8, D9	1N4148	R16, R28	2K2Ω
IC1	MC9S08QG8	R17	10MΩ
IC2	4051	R18, R21	TRIMM 500Ω
IC3	4060	R23	3K3Ω
IC4	ICL7660	R24	470Ω
IC5	NE5532N	R26	6K8
IC7	7805	R27	1K5
IC8	LE33	R29	1K
IC5	IC5	R15	82Ω
T1	BC560	R25, R30	4K7Ω
K1, K2	G5V1	X1	NEB21R
L1	1mH	X2,X4	R141426
OK1, OK2	PC817	X3	F09HP
Q1, Q2	BC639	X5	22-23-2031

tab. 6.Seznam použitých součástek osciloskopu

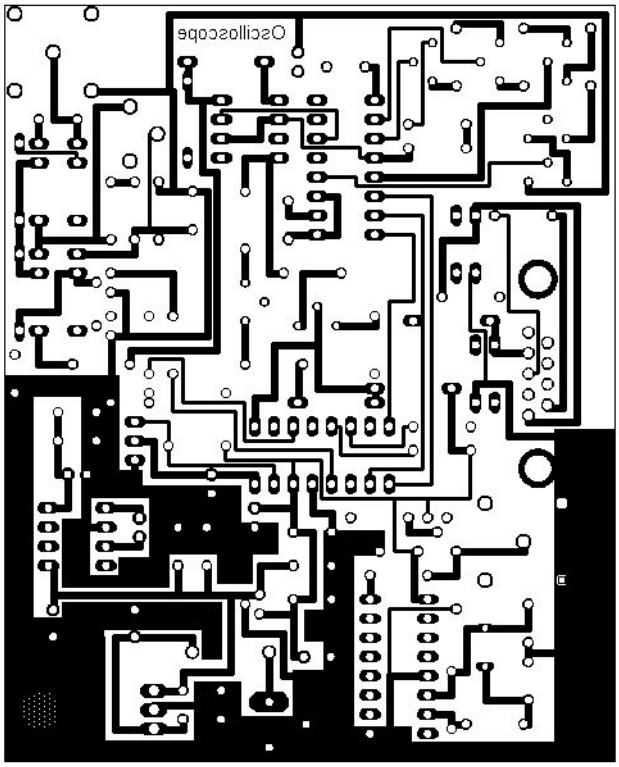
10.Deska plošných spojů



obr. 26.Rozložení součástek na DPS



obr. 27.Pohled na DPS ze strany součástek



obr. 28.Pohled na DPS ze strany spojů

11. Shrnutí vlastností osciloskopu

Parametry popsaného osciloskopu jsou limitovány především použitým mikrokontrolérem a jeho AD převodníkem. Pokud by měli být parametry výrazně lepší musela by být konstrukce zcela jiného charakteru s využitím rychlých AD převodníků a rychlé vyrovnávací paměti. Taková konstrukce by však byla neúměrně složitější. Pro cíl, který byl stanoven na počátku návrhu tj. osciloskop pro nf signály však daný mikrokontrolér a AD převodník vyhovují a účel splňují.

Při školním měřením jsem se setkal s tvrzením, že odečítat data z osciloskopu lze s přesností přibližně 10%. Při správném zkalibrování osciloskopu můžeme dosáhnout mnohem lepší přesnosti. Výhodou u tohoto digitálního osciloskopu je možnost procházet změřený signál bod po bodu a odečítat naměřené napětí.

Nevýhodou, kterou tento osciloskop trpí podobně jako jiné digitální osciloskopy je tzv. aliasing. Jedná se o zobrazování chybného průběhu při shodě vzorkovací frekvence s násobky měřené periody.

Pro používání v běžné amatérské praxi nabízí tento osciloskop jednoduchou a levnou možnost doplnění amatérského pracoviště o kvalitní měřící přístroj.

IV. Citace

[1.] Interfacing the PC AT keyboard, http://www.atmel.com/dyn/resources/prod_documents/doc1235.pdf, ©ATMEL corporation 2002
[2.] http://www.beyondlogic.org/keyboard/keybrd.htm ©Craig Peacock 1999-2005
[3.] Praktická elektronika A Radio, ročník XI/2006, číslo 10, str. 9 – 13
[4.] ATmega 8515 datascheet,www.atmel.com/dyn/resources/prod_documents/doc2512.pdf, ©ATMEL corporation 2002

V. Použité prameny

Belza J.: Operační zesilovače pro obyčejné smrtelníky, BEN – technická literatura, Praha 2004
ISBN: 8073000601
Dietmeier U.: Vzorce pro elektroniku, BEN – technická literatura, Praha 1999
ISBN: 8086056538
Jedlička P.: Přehled obvodů řady CMOS 4000 díl I. 4000 … 4099, BEN – technická literatura, Praha 1994
Vít V.: Základy televizní techniky; SNTL; Praha 1987
Praktická elektronika A Radio, ročník XI/2006, číslo 10
Katalog elektronické součástky, stavebnice a moduly; Elektronika Zdeněk Krčmář 2007
http://www.beyondlogic.org/keyboard/keybrd.htm ©Craig Peacock 1999-2005
http://www.lancos.com/prog.html
ATmega 8515 datascheet,www.atmel.com/dyn/resources/prod_documents/doc2512.pdf, ©ATMEL corporation 2002
ATtiny 13 datascheet, http://www.atmel.com/dyn/resources/prod_documents/doc2535.pdf, ©ATMEL corporation 2002
LM2575 datascheet, http://www.ortodoxism.ro/datasheets2/9/0o5u8fuw6uw86g2qtrc3178eexcy.pdf, ©ON Semiconductor
74HC573 datascheet, http://www.ortodoxism.ro/datasheets/philips/74HC_HCT573_CNV_2.pdf © Philips Semiconductors 1990
74HC166 datascheet, http://www.ortodoxism.ro/datasheets/philips/74HC_HCT166_CNV_2.pdf, © Philips Semiconductors 1990
Interfacing the PC AT keyboard, http://www.atmel.com/dyn/resources/prod_documents/doc1235.pdf, ©ATMEL corporation 2002
Switching diode 1N4148/1N4150/1N4448/1N914B, http://www.ortodoxism.ro/datasheets/rohm/1n4148.pdf ©ROHM
NE5532, NE5532A,SA5532,SA5532A Dual low-noise operational, http://www.ortodoxism.ro/datasheets/texasinstruments/ne5532.pdf ©Texas Instruments Incorporated 2005
Switched-Capacitor Voltage Converters, ©Maxim Integrated Products 1994
MC9S08QG8,MC9S08QG4 Data Scheet, http://freescale.com, ©Freescale Semiconductor, Inc. 2005

VI. Obsah

VII. Přílohy

1. Program pro Atmega 8515

```
;obrazovka.asm v. 1.00 ze dne 12. 3. 2007
;pro programovani zatrhnout security bity: SUT1; BODEN, CKOPT
.include        "m8515def.inc"
.equ            TABULKA         =0x800
.equ            UVOD            =0x700
.def            REG0            =R0
.def            REG1            =R1
.def            ZACOBR1         =R2
.def            ZACOBR2         =R3
.def            MASKA           =R4
.def            BARVA           =R5
.def            TEMP0           =R6
.def            ZAPISBUF        =R7
.def            ZACPIS          =R8
.def            POZX1           =R9
.def            POZY1           =R10
.def            POZX2           =R11
.def            POZY2           =R12
.def            CTIBUF          =R13
.def            TYPRAD          =R14
.def            POSUV           =R15
;spodni 4 bity posuv X, horni bity posuv Y
.def            TEMP            =R16
.def            TEMP2           =R17
.def            TEMP3           =R18
.def            TEMP4           =R19
.def            TEMP5           =R20
.def            INTTEMP         =R21
.def            RADEK           =R22
.def            COUNTER         =R23
.def            ZNAK            =R24
.def            STAV            =R25
;u STAV.0 = kterou obrazovku zobrazuju (0 .. prvni; 1 .. druhou)
;       1 = pomocny priznak "zaporny Y" pouzivany pri zobrazeni usecky
;       1 = pomocny priznak "zaporny Y" pouzivany pri zobrazeni grafu
;       2 = vstupni priznak pro SAVEZNAK (0 .. dokresli; 1 .. prekresli)
; registry 26 až 31 jsou X,Y,Z!!!
.org            0x0000
                rjmp            RESET
;*********************************************************************
;preruseni od casovace pro synchronizacni impulsy a zobrazeni
.org            0x0004
                rjmp            CEKAM
.org            0x0005
;ulozi STATUS do zasobniku
MAININT:        in              INTTEMP,SREG    ;1
                push            INTTEMP         ;2
                sbic            PINB,0
                rjmp            TSNIMSYN
;synchronizace zapisoveho impulsu
TSNIMSYN:       sbrc            TYPRAD,3        ;1/2
                rjmp            SNIMEK          ;2
                sbrc            TYPRAD,0        ;1/2
                rjmp            ZATEMNI1        ;2
                sbrc            TYPRAD,2        ;1/2
                rjmp            ZATEMNI2        ;2
;synchro impuls
                cbi             PORTD,4         ;2
                ldi             INTTEMP,16
DELL4uSD:       dec             INTTEMP
                brne            DELL4uSD
                sbi             PORTD,4
;test zatemnovacich radku
;zobrazeni datoveho radku
```

```
                    mov     INTTEMP,POSUV
                    andi    INTTEMP,0x0F
                    subi    INTTEMP,0xF0
DELLXuS:            nop                             ;1
                    dec     INTTEMP                 ;1
                    brne    DELLXuS                 ;2
;zapnuti hodin pro posuvne registry
                    sbi     PORTB,1                 ;2
;data na obrazovku
                    ld      INTTEMP,X               ;4 usec     1-5
                    st      X+,INTTEMP              ;4 usec
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X               ;6-10
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X               ;11 - 15
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X               ;16-20
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X               ;21-25
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X               ;26-30
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X
                    st      X+,INTTEMP
                    ld      INTTEMP,X               ;31-35
```

```
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X       ;36-40
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X       ;41-45
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X       ;46-50
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X       ;51-55
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X       ;56-60
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X       ;61-64
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
ld      INTTEMP,X
st      X+,INTTEMP
nop
nop
nop
```

```
;zastav posuvne registry
                cbi     PORTB,1
;testuju preteceni citace radku pres rozsah obrazovky
                sbrc    STAV,0
                rjmp    DRUHAOBR
;obsluha prvni obrazovky
                sbrs    XH,6
                rjmp    TEST256RAD
                clr     XL
                ldi     XH,0x80
                rjmp    TEST256RAD
;obsluha druhe obrazovky
DRUHAOBR:       sbrc    XH,7
                rjmp    TEST256RAD
                clr     XL
                ldi     XH,0xC0
                rjmp    TEST256RAD
;je 256 radku hotovo?
TEST256RAD:     inc     RADEK
                brne    UKONCI
                lsl     TYPRAD
UKONCI:         rjmp    INTEND
;zatemnovaci radky horni
ZATEMNI1:       nop
;synchro impuls
                cbi     PORTD,4         ;2
                ldi     INTTEMP,16
DELL4uSDA:      dec     INTTEMP
                brn     DELL4uSDA
                sbi     PORTD,4
                inc     RADEK
                mov     INTTEMP,POSUV
                andi    INTTEMP,0xF0
                swap    INTTEMP
                subi    INTTEMP,0xDE
                cp      RADEK,INTTEMP
                brne    INTEND
                clr     RADEK
                lsl     TYPRAD
                rjmp    INTEND
;zatemnovaci radky spodni
ZATEMNI2:
;synchro impuls
                cbi     PORTD,4         ;2
                ldi     INTTEMP,16
DELL4uSDB:      dec     INTTEMP
                brne    DELL4uSDB
                sbi     PORTD,4
                inc     RADEK
                mov     INTTEMP,POSUV
                andi    INTTEMP,0xF0
                swap    INTTEMP
                subi    INTTEMP,0x16
                neg     INTTEMP
                cp      RADEK,INTTEMP
                brne    INTEND
                clr     RADEK
                lsl     TYPRAD
                rjmp    INTEND
;osetreni snimkoveho synchro impulsu
SNIMEK:         sbi     PORTD,3
                sbi     PORTD,4         ;po dobu 4uS "1" pak "0"
                ldi     INTTEMP,16
DELL4uSU:       dec     INTTEMP
                brne    DELL4uSU
                cbi     PORTD,4
                inc     RADEK           ;osetreni citace radku
                cpi     RADEK,0x3
                brne    INTEND
                clr     RADEK
```

```
                        ldi             INTTEMP,0x1
                        mov             TYPRAD,INTTEMP
                        sbrs            STAV,0
                        mov             XH,ZACOBR1          ;pocatecni misto pro vyber z pamneti
                        sbrc            STAV,0
                        mov             XH,ZACOBR2
                        clr             XL
                        cbi             PORTD,3
;konec interuptu pro vsechny rezimy
INTEND:                 pop             INTTEMP
                        out             SREG,INTTEMP
                        reti
;*************************************************************************
;synchronizacni cekani na odstraneni vlivu delky instrukci
CEKAM:                  sei
                        push            TEMP                ; 2
                        in              TEMP,SREG           ; 1
                        push            TEMP                ; 2
                        sbis            UCSRA,RXC           ; 2 test prichoziho znaku
                        rjmp            CEKAMC              ; 2
                        inc             ZAPISBUF            ; 1
                        cp              ZAPISBUF,CTIBUF     ; 1
                        breq            CEKAMSUB            ; 1
                        push            ZL                  ; 2
                        push            ZH                  ; 2
                        dec             ZAPISBUF            ; 1
                        in              TEMP,UDR            ; 1
                        mov             ZL,ZAPISBUF         ; 1
                        ldi             ZH,0x1              ; 1
                        nop
                        nop
                        nop
                        nop
                        nop
                        nop
                        nop
                        nop
                        nop
                        nop
                        nop
                        nop
                        nop
                        nop
                        nop
                        st              Z,TEMP              ; 2
                        inc             ZAPISBUF            ; 1
                        pop             ZH                  ; 2
                        pop             ZL                  ; 2
                        pop             TEMP
                        out             SREG,TEMP
                        pop             TEMP                ; 2
                        reti
CEKAMSUB:               dec             ZAPISBUF            ; 1
                        nop
                        nop
                        nop
                        nop
                        pop             TEMP
                        out             SREG,TEMP
                        pop             TEMP                ; 2
                        reti
CEKAMC:                 nop
                        nop
                        nop
                        nop
                        nop
                        nop
                        nop
                        nop
                        nop
```

```
                        nop
                        pop             TEMP
                        out             SREG,TEMP
                        pop             TEMP                    ; 2
                        reti
;*************************************************************************
;inicializace promennych po startu
RESET:
;inicializace ukazatele zasobniku na konec adresy RAM
                        ldi             TEMP,low(RAMEND)
                        out             SPL,TEMP
                        ldi             TEMP,high(RAMEND)
                        out             SPH,TEMP
;inicializace vystupu portu  A-E
                        ldi             TEMP,0xFF
                        out             DDRA,TEMP
                        ldi             TEMP,0xFF
                        out             DDRB,TEMP
                        ldi             TEMP,0xFF
                        out             DDRC,TEMP
                        out             DDRD,TEMP
                        out             DDRE,TEMP
;nastaveni ostatnich promennych
                        clr             ZAPISBUF
                        clr             CTIBUF
                        clr             RADEK                   ;pocitadlo a typ zobrazovanych
radku
                        ldi             TEMP,0xFF
                        mov             BARVA,TEMP
                        clr             STAV
;0x1 horni zat. 0x2 data, 0x4 dolni zat. 0x8 synchro
                        ldi             TEMP,0x1
                        mov             TYPRAD,TEMP
;nastaveni pozice "kurzoru"
                        clr             POZX1
                        clr             POZY1
                        clr             POZX2
                        clr             POZY2
;inicializace komunikace s externi pameti
                        ldi             TEMP,0xC0
                        out             MCUCR,TEMP
                        ldi             TEMP,0x44
                        out             EMCUCR,TEMP
                        ldi             TEMP,0x40
                        out             SFIOR,TEMP
                        ldi             TEMP,0x80
                        mov             ZACOBR1,TEMP
                        mov             ZACPIS,TEMP
                        ldi             TEMP,0xC0
                        mov             ZACOBR2,TEMP
;uvodni vymazani obrazovky
                        cbi             PORTB,2
                        ldi             TEMP,0xC0
                        mov             ZACPIS,TEMP
                        rcall           CLRSCR
                        ldi             TEMP,0x80
                        mov             ZACPIS,TEMP
                        rcall           CLRSCR
;inicializace seriove komunikace
                        ldi             TEMP,0x4D               ;nastaveni rychlosti prenosu na 9600
                        out             UBRRL,TEMP
                        clr             TEMP
                        out             UBRRH,TEMP
                        ldi             TEMP,0x10
                        out             UCSRB,TEMP
                        ldi             TEMP,0x86
                        out             UCSRC,TEMP
;nacteni posunu obrazovky
EEPROMR:                sbic            EECR,EEWE
                        rjmp            EEPROMR
```

```
                        clr             TEMP
                        out             EEARH,TEMP
                        out             EEARL,TEMP
                        sbi             EECR,EERE
                        in              POSUV,EEDR
;*********************************
;nastaveni casovace pro preruseni
;casovac pro preruseni po 64us
                        ldi             TEMP,0x02
                        out             OCR1AH,TEMP
                        ldi             TEMP,0xFD
                        out             OCR1AL,TEMP
;casovac pro uvodni sesynchronizovani
                        ldi             TEMP,0x00
                        out             OCR1BH,TEMP
                        ldi             TEMP,0x0F
                        out             OCR1BL,TEMP
;nastaveni rezimu casovace
                        ldi             TEMP,0x00
                        out             TCCR1A,TEMP
                        ldi             TEMP,0x9
                        out             TCCR1B,TEMP
                        ldi             TEMP,0x60
                        out             TIMSK,TEMP
;nastaveni casovace pro hodiny posuvnych registru
                        clr             TEMP
                        out             TCNT0,TEMP
                        out             OCR0,TEMP
                        ldi             TEMP,0x19
                        out             TCCR0,TEMP
;povoleni preruseni
                        sei
;*********************************
;uvodni text na obrazovce
                        ldi             ZL,0x0
                        ldi             ZH,0xE
                        rcall           VYPISTXT
                        ldi             TEMP,0x0F
                        mov             BARVA,TEMP
                        rcall           VYPISTXT
                        clr             BARVA
                        com             BARVA
                        rcall           VYPISTXT
                        ldi             TEMP,0x0F0
                        mov             BARVA,TEMP
                        rcall           VYPISTXT
                        ldi             TEMP,0x0F
                        mov             BARVA,TEMP
                        rcall           VYPISTXT
                        clr             BARVA
                        com             BARVA
                        rcall           VYPISTXT
                        ldi             TEMP,0x30
PLNTEMP2:               ldi             TEMP2,0xFF
PLNTEMP3:               ldi             TEMP3,0xFF
WAITCYKL:               dec             TEMP3
                        brne            WAITCYKL
                        dec             TEMP2
                        brne            PLNTEMP3
                        dec             TEMP
                        brne            PLNTEMP2
                        rcall           CLRSCR
                        clr             POZX1
                        clr             POZY1
;*************************************************************************
;*                      Hlavni smycka                                    *
;*************************************************************************
LOOP:                   rcall           GETCH
                        cpi             ZNAK,0x1B
                        brne            NOESC
```

```
                rjmp    ESC
NOESC:          rcall   WRITE
                rjmp    LOOP
;tady zpracuju ESC prikaz
ESC:            rcall   GETCH
                ldi     TEMP,0xDF
                and     ZNAK,TEMP
                cpi     ZNAK,0x09       ;TAB
                breq    ESCTABP
                cpi     ZNAK,0x42       ;"B"
                breq    ESCBP
                cpi     ZNAK,0x43       ;"C"
                breq    ESCCP
                cpi     ZNAK,0x47       ;"G"
                breq    ESCGP
                cpi     ZNAK,0x48       ;"H"
                breq    ESCHP
                cpi     ZNAK,0x4C       ;"L"
                breq    ESCLP
                cpi     ZNAK,0x4D       ;"M"
                breq    ESCMP
                cpi     ZNAK,0x4F       ;"O"
                breq    ESCOP
                cpi     ZNAK,0x50       ;"P"
                breq    ESCPP
                cpi     ZNAK,0x51       ;"Q"
                breq    ESCQP
                cpi     ZNAK,0x52       ;"R"
                breq    ESCRP
                cpi     ZNAK,0x53       ;"S"
                breq    ESCSP
                cpi     ZNAK,0x54       ;"T"
                breq    ESCTP
                cpi     ZNAK,0x55       ;"U"
                breq    ESCUP
                cpi     ZNAK,0x56       ;"V"
                breq    ESCVP
                cpi     ZNAK,0x57       ;"W"
                breq    ESCWP
                cpi     ZNAK,0x58       ;"X"
                breq    ESCXP
                cpi     ZNAK,0x59       ;"Y"
                breq    ESCYP
                cpi     ZNAK,0x5A       ;"Z"
                breq    ESCZP
                rjmp    LOOP
ESCTABP:        rjmp    ESCTAB
ESCBP:          rjmp    ESCB
ESCCP:          rjmp    ESCC
ESCGP:          rjmp    ESCG
ESCHP:          rjmp    ESCH
ESCLP:          rjmp    ESCL
ESCMP:          rjmp    ESCM
ESCOP:          rjmp    ESCO
ESCPP:          rjmp    ESCP
ESCQP:          rjmp    ESCQ
ESCRP:          rjmp    ESCR
ESCSP:          rjmp    ESCS
ESCTP:          rjmp    ESCT
ESCUP:          rjmp    ESCU
ESCVP:          rjmp    ESCV
ESCWP:          rjmp    ESCW
ESCXP:          rjmp    ESCX
ESCYP:          rjmp    ESCY
ESCZP:          rjmp    ESCZ
;*********************
;vodorovna cara
ESCT:           rcall   VEMXY
                mov     TEMP2,YL
                rcall   UPRAVXY
```

```
                rcall   GETCH           ;ZNAK .. delka cary
                tst     ZNAK
                breq    ESCTEND
                ldi     TEMP,0x4
                sub     TEMP,TEMP3
                mov     TEMP3,TEMP
                clr     TEMP            ;TEMP .. registr pro prvni 4 bity
                add     TEMP2,ZNAK      ;kontrola preteceni
                brcc    ESCTOK
ESCTEND:        rjmp    LOOP
ESCTOK:         tst     TEMP3
                breq    ESCTC
                tst     ZNAK
                breq    ESCTOKA
                dec     TEMP3
                dec     ZNAK
                sec
                rol     TEMP            ;postupne nasouva nahoru 0
                rjmp    ESCTOK
ESCTOKA:        dec     TEMP3
                clc
                rol     TEMP
                rjmp    ESCTOK
ESCTC:          mov     TEMP2,TEMP
                swap    TEMP
                or      TEMP,TEMP2
                and     TEMP,BARVA
                ld      TEMP2,Y
                or      TEMP2,TEMP
                st      Y+,TEMP2        ;konec 1. bytu
                clr     TEMP
ESCTL:          cpi     ZNAK,0x4
                brlo    ESCTK
                ld      TEMP2,Y
                or      TEMP2,BARVA
                st      Y+,TEMP2
                subi    ZNAK,0x4
                rjmp    ESCTL           ;konec prostredni casti
ESCTK:          tst     ZNAK
                breq    ESCTUK
                sec
                ror     TEMP
                dec     ZNAK
                rjmp    ESCTK
ESCTUK:         mov     TEMP2,TEMP
                swap    TEMP
                or      TEMP,TEMP2
                and     TEMP,BARVA
                ld      TEMP2,Y
                or      TEMP2,TEMP
                st      Y,TEMP2
                rjmp    LOOP
;*******************
;svisla cara
ESCU:           rcall   VEMXY
                rcall   UPRAVXY
                mov     TEMP,ZNAK
                rcall   GETCH           ;ZNAK .. delka cary
                add     TEMP,ZNAK       ;kontrola preteceni
                brcc    ESCUOK
ESCUEND:        rjmp    LOOP
ESCUOK:         ldi     TEMP,0x88       ;TEMP .. maska pozice bitu
ESCUL:          tst     TEMP3           ;upraveno podle zbytku v TEMP3
                breq    ESCUC
                lsr     TEMP
                dec     TEMP3
                rjmp    ESCUL
ESCUC:          and     TEMP,BARVA      ;do TEMP pridana barva
ESCUML:         tst     ZNAK
                breq    ESCUEND
```

```
                    ld          TEMP3,Y         ;TEMP2 .. nactena hodnota
                    or          TEMP3,TEMP
                    st          Y+,TEMP3
                    adiw        Y,0x3F
                    dec         ZNAK
                    rjmp        ESCUML
;*******************
;pevny zapis do pameti
ESCM:               rcall       GETCH
                    mov         YH,ZNAK
                    rcall       GETCH
                    mov         YL,ZNAK
                    rcall       GETCH
                    mov         COUNTER,ZNAK
ESCMLOOP:           dec         COUNTER
                    rcall       GETCH
                    ori         YH,0x80
                    st          Y+,ZNAK
                    tst         COUNTER
                    brne        ESCMLOOP
ESCMEND:            rjmp        LOOP
;*******************
;nastaveni posunu obrazu v horizontalnim smeru
ESCH:               rcall       GETCH
                    andi        ZNAK,0x0F
                    ldi         TEMP,0xF0
                    and         POSUV,TEMP
                    or          POSUV,ZNAK
                    rjmp        ULOZEEP
;*******************
;nastaveni posunu obrazu v horizontalnim smeru
ESCO:               rcall       GETCH
                    andi        ZNAK,0x0F
                    swap        ZNAK
                    ldi         TEMP,0x0F
                    and         POSUV,TEMP
                    or          POSUV,ZNAK
ULOZEEP:            sbic        EECR,EEWE
                    rjmp        ULOZEEP
                    clr         TEMP
                    out         EEARH,TEMP
                    out         EEARL,TEMP
                    out         EEDR,POSUV
                    cli
                    sbi         EECR,EEMWE
                    sbi         EECR,EEWE
                    sei
                    rjmp        LOOP
;*******************
;nastaveni pruhlednosti/nepruhlednosti znaku
ESCB:               rcall       GETCH
                    cpi         ZNAK,0x30
                    breq        ESCB0
                    cpi         ZNAK,0x31
                    breq        ESCB1
                    rjmp        LOOP
ESCB0:              andi        STAV,0xFB
                    rjmp        LOOP
ESCB1:              ori         STAV,0x04
                    rjmp        LOOP
;*******************
;restart zarizeni
ESCQ:               rjmp        RESET
;*******************
;prehozeni ploch
ESCTAB:             sbrs        STAV,0
                    rjmp        ESCTAB2
                    andi        STAV,0xFE
                    ldi         TEMP,0x80
                    mov         ZACPIS,TEMP
```

```
                    rjmp    LOOP
ESCTAB2:            ori     STAV,0x01
                    ldi     TEMP,0xC0
                    mov     ZACPIS,TEMP
                    rjmp    LOOP
;*******************
;nastaveni zobrazovane plochy
ESCV:               rcall   GETCH
                    cpi     ZNAK,0x31
                    breq    ESCV1
                    cpi     ZNAK,0x32
                    breq    ESCV2
                    rjmp    LOOP
ESCV1:              ldi     TEMP,0x8
ESCVL1:             cp      TYPRAD,TEMP
                    brne    ESCVL1
                    andi    STAV,0xFE
                    rjmp    LOOP
ESCV2:              ldi     TEMP,0x8
ESCVL2:             cp      TYPRAD,TEMP
                    brne    ESCVL2
                    ori     STAV,0x01
                    rjmp    LOOP
;*******************
;nastaveni plochy pro zapis
ESCW:               rcall   GETCH
                    cpi     ZNAK,0x31
                    breq    ESCW1
                    cpi     ZNAK,0x32
                    breq    ESCW2
                    rjmp    LOOP
ESCW1:              ldi     TEMP,0x80
                    mov     ZACPIS,TEMP
                    rjmp    LOOP
ESCW2:              ldi     TEMP,0xC0
                    mov     ZACPIS,TEMP
                    rjmp    LOOP
;*******************
;vykresleni spojitych bodu
ESCG:               rcall   GETCH
                    mov     YL,ZNAK             ;YL .. aktualni X souradnice
                    rcall   GETCH
                    tst     ZNAK
                    breq    ESCGEND
                    mov     COUNTER,ZNAK    ;COUNTER .. počet vykreslovanych hodnot
                    mov     TEMP,YL             ;kontrola preteceni pres obrazovku v X
                    add     TEMP,COUNTER
                    brcs    ESCGEND
                    rcall   GETCH               ;YH .. "stara" Y souradnice
                    mov     YH,ZNAK
                    rcall   UPRAVXY
                    ldi     TEMP,0x88           ;TEMP .. maska pozice bitu
ESCGL:              tst     TEMP3               ;upraveno podle zbytku v TEMP3
                    breq    ESCGLOOP
                    lsr     TEMP
                    dec     TEMP3
                    rjmp    ESCGL
ESCGEND:            rjmp    LOOP
ESCGLOOP:           dec     COUNTER
                    tst     COUNTER
                    breq    ESCGEND
                    mov     TEMP2,ZNAK          ;TEMP2 .. zaloha stare Y pozice
                    rcall   GETCH               ;ZNAK .. aktualni Y souradnice
                    cp      TEMP2,ZNAK          ;rozskok, podle pozic
                    brlo    ESCGD
                    mov     TEMP3,TEMP2         ;TEMP3 .. delka 1. svisle cary
                    sub     TEMP3,ZNAK
                    clr     TEMP4
                    lsr     TEMP3
                    rol     TEMP4               ;TEMP4 .. zbytek TEMP3 / 2
```

```
                add     TEMP4,TEMP3     ;TEMP4 .. delka 2. svisle cary
                mov     TEMP0,TEMP
                and     TEMP0,BARVA
ESCGU1:         ld      TEMP5,Y
                or      TEMP5,TEMP0
                st      Y,TEMP5
                tst     TEMP3
                breq    ESCGU2
                dec     TEMP3
                sbiw    Y,0x20
                sbiw    Y,0x20
                rjmp    ESCGU1
ESCGU2:         lsr     TEMP
                brcc    ESCGU4
                ldi     TEMP,0x88
                adiw    Y,0x1
ESCGU4:         mov     TEMP0,TEMP
                and     TEMP0,BARVA
ESCGU3:         tst     TEMP4
                breq    ESCGLOOP
                dec     TEMP4
                ld      TEMP5,Y
                or      TEMP5,TEMP0
                st      Y,TEMP5
                sbiw    Y,0x20
                sbiw    Y,0x20
                rjmp    ESCGU3
ESCGD:          mov     TEMP3,ZNAK      ;TEMP3 .. delka 1. svisle cary
                sub     TEMP3,TEMP2
                clr     TEMP4
                lsr     TEMP3
                rol     TEMP4           ;TEMP4 .. zbytek TEMP3 / 2
                add     TEMP4,TEMP3     ;TEMP4 .. delka 2. svisle cary
                mov     TEMP0,TEMP
                and     TEMP0,BARVA
ESCGD1:         ld      TEMP5,Y
                or      TEMP5,TEMP0
                st      Y,TEMP5
                tst     TEMP3
                breq    ESCGD2
                dec     TEMP3
                adiw    Y,0x20
                adiw    Y,0x20
                rjmp    ESCGD1
ESCGD2:         lsr     TEMP
                brcc    ESCGD4
                ldi     TEMP,0x88
                adiw    Y,0x1
ESCGD4:         mov     TEMP0,TEMP
                and     TEMP0,BARVA
ESCGD3:         tst     TEMP4
                breq    ESCGLOOP
                dec     TEMP4
                ld      TEMP5,Y
                or      TEMP5,TEMP0
                st      Y,TEMP5
                adiw    Y,0x20
                adiw    Y,0x20
                rjmp    ESCGD3
;*******************
;nastaveni barvy
ESCC:           rcall   GETCH
                mov     BARVA,ZNAK
                rjmp    LOOP
;*******************
;vykresleni usecky
ESCL:           rcall   GETCH
                mov     TEMP4,ZNAK      ;TEMP4 .. "X1"
                rcall   GETCH
                mov     TEMP5,ZNAK      ;TEMP5 .. "Y1"
```

```
                rcall   VEMXY           ;YL .. "X2"; YH .. "Y2"
                cp      YL,TEMP4
                brcs    ESCLSOURY
;pokud je X1>X2, prohodim oba body, aby v YL/H byl pocatecni bod
                mov     TEMP,YL
                mov     YL,TEMP4
                mov     TEMP4,TEMP
                mov     TEMP,YH
                mov     YH,TEMP5
                mov     TEMP5,TEMP
ESCLSOURY:      ldi     TEMP,0x02       ;test, zda je Y1<Y2 - pak je v Y ose prirustek
                or      STAV,TEMP       ;jinak je ubytek => priznak "zaporne  Y"
                cp      TEMP5,YH
                brcs    ESCLDELTA
                ldi     TEMP,0xFD
                and     STAV,TEMP
ESCLDELTA:      sub     TEMP4,YL        ;TEMP4 .. "dX"
                sbrs    STAV,1
                rjmp    ESCLSUB
                mov     TEMP,YH
                sub     TEMP,TEMP5      ;TEMP5 .. "dY"
                mov     TEMP5,TEMP
                rjmp    ESCLN
ESCLSUB:        sub     TEMP5,YH
ESCLN:          mov     TEMP3,TEMP5     ;TEMP3 .. "N" .. pocet pixelu
                cp      TEMP5,TEMP4
                brcc    ESCLKROK
                mov     TEMP3,TEMP4
ESCLKROK:       tst     TEMP4           ;test deleni nulou
                breq    ESCLX0DIV
                mov     TEMP,TEMP3      ;krokX:=N div dx
                clr     TEMP2
ESCLKROKL1:     sub     TEMP,TEMP4
                brlo    ESCLKROK2
                inc     TEMP2           ;TEMP2 .. krokX
                rjmp    ESCLKROKL1
ESCLKROK2:      add     TEMP,TEMP4
                mov     REG0,TEMP       ;REG0 .. zbytekX
                rjmp    ESCLX0DIE
ESCLX0DIV:      ldi     TEMP2,0xFF
                clr     REG0
ESCLX0DIE:      tst     TEMP5           ;test deleni nulou
                breq    ESCLY0DIV
                mov     TEMP,TEMP3      ;krokY:=N div dy
                clr     TEMP0
ESCLKROKL2:     sub     TEMP,TEMP5
                brlo    ESCLZAPOR
                inc     TEMP0           ;TEMP0 .. krokY
                rjmp    ESCLKROKL2
ESCLZAPOR:      add     TEMP,TEMP5
                mov     REG1,TEMP       ;REG1 .. zbytekY
                rjmp    ESCLY0DIE
ESCLY0DIV:      clr     TEMP0
                dec     TEMP0
                clr     REG1
ESCLY0DIE:      cp      REG0,TEMP2      ;porovnani zbytkuX a krokuX
                brcs    ECSLNODELX
                mov     TEMP,TEMP3      ;zbytekX:=N div zbytekX
                mov     COUNTER,REG0
                clr     REG0
ESCLZBL1:       sub     TEMP,COUNTER
                brlo    ESCLZB2
                inc     REG0            ;REG0 .. pomocny krok zbytkuX
                rjmp    ESCLZBL1
ECSLNODELX:     ldi     TEMP,0xFF
                mov     REG0,TEMP
ESCLZB2:        cp      REG1,TEMP0      ;porovnani zbytkuY a krokuY
                brcs    ECSLNODELY
                mov     TEMP,TEMP3      ;zbytekY:=N div zbytekY
                mov     COUNTER,REG1
```

```
                clr     REG1
ESCLZBL2:       sub     TEMP,COUNTER
                brlo    ESCLINCI
                inc     REG1            ;REG1 .. pomocny krok zbytkuY
                rjmp    ESCLZBL2
ECSLNODELY:     ldi     TEMP,0xFF
                mov     REG1,TEMP
ESCLINCI:       clr     COUNTER         ;COUNTER .. pocitadlo krokuX
                clr     ZNAK            ;ZNAK .. pocitadlo krokuY
                clr     ZL              ;ZL .. pocitadlo zbytkuX
                clr     ZH              ;ZH .. pocitadlo zbytkuY
ESCLOOP:        tst     TEMP3
                breq    ESCLEND
                dec     TEMP3
                inc     COUNTER
                inc     ZNAK
                inc     ZL
                inc     ZH
                cp      COUNTER,TEMP2
                brne    ESCLL2
                inc     YL
                clr     COUNTER
ESCLL2:         cp      ZNAK,TEMP0
                brne    ESCLL3
                clr     ZNAK
                sbrc    STAV,1
                dec     YH
                sbrs    STAV,1
                inc     YH
ESCLL3:         cp      ZL,REG0
                brne    ESCLL4
                inc     YL
                clr     ZNAK
ESCLL4:         cp      ZH,REG1
                brne    ESCLL5
                clr     ZH
                sbrc    STAV,1
                dec     YH
                sbrs    STAV,1
                inc     YH
ESCLL5:         push    TEMP2           ;vykresleni bodu
                push    TEMP3
                push    TEMP4
                push    YL
                push    YH
                rcall   UPRAVXY
                rcall   POINT
                pop     YH
                pop     YL
                pop     TEMP4
                pop     TEMP3
                pop     TEMP2
                rjmp    ESCLOOP
ESCLEND:        rjmp    LOOP
;*******************
;vykresleni plneho obdelniku
ESCR:           rcall   VEMXY
                rcall   GETCH
                mov     TEMP,ZNAK       ;TEMP4 .. velikost strany X
                mov     TEMP4,TEMP
                add     TEMP,YL
                brcc    ESCRY
                mov     TEMP4,YL
                com     TEMP4
ESCRY:          rcall   GETCH
                mov     TEMP,ZNAK       ;TEMP5 .. velikost strany Y
                mov     TEMP5,TEMP
                add     TEMP,YH
                brcc    ESCRC
                mov     TEMP5,YH
```

```
                com     TEMP5
ESCRC:          rcall   UPRAVXY         ;TEMP3 .. zbytek (posun v ramci Bytu)
                mov     ZH,YH
                mov     ZL,YL
                mov     TEMP,TEMP3      ;uprava zbytku v TEMP3 na "4-zbytek"
                ldi     TEMP3,0x04      ;budeme posouvat doleva o tolik pixelu
                sub     TEMP3,TEMP
                clr     TEMP2
                sub     TEMP4,TEMP3     ;TEMP4 .. zmenseni velikosti strany o zbytek
                brcc    ESCRZZ
                add     TEMP4,TEMP3
                sub     TEMP3,TEMP4
                clr     TEMP0
ESCRMA:         subi    TEMP4,0x01
                brlo    ESCRMB
                sec
                rol     TEMP0
                rjmp    ESCRMA
ESCRMB:         subi    TEMP3,0x01
                brlo    ESCRMC
                lsl     TEMP0
                rjmp    ESCRMB
ESCRMC:         clr     TEMP4
                clr     TEMP2
                mov     TEMP,TEMP0
                swap    TEMP
                or      TEMP0,TEMP
                rjmp    ESCRLOOP
ESCRZZ:         subi    TEMP4,0x04
                brlo    ESCRZZK
                inc     TEMP2
                rjmp    ESCRZZ          ;TEMP2 .. pocet Bytu
ESCRZZK:        ldi     TEMP,0x04
                add     TEMP4,TEMP      ;TEMP4 .. konecny zbytek
                clr     TEMP0
                mov     TEMP,TEMP3
ESCRZZA:        subi    TEMP,0x01
                brlo    ESCRZZAK
                sec
                rol     TEMP0
                rjmp    ESCRZZA
ESCRZZAK:       mov     TEMP,TEMP0
                swap    TEMP
                or      TEMP0,TEMP      ;TEMP0 .. maska pro prvni Byte
                clr     TEMP
ESCRKZ:         subi    TEMP4,0x01
                brlo    ESCRZKK
                sec
                ror     TEMP
                rjmp    ESCRKZ
ESCRZKK:        mov     TEMP4,TEMP
                swap    TEMP
                or      TEMP4,TEMP      ;TEMP4 .. maska konecneho Byte
ESCRLOOP:       subi    TEMP5,0x01
                brlo    ZPETLOOP
                mov     YL,ZL
                mov     YH,ZH
                adiw    Z,0x20
                adiw    Z,0x20
                ld      TEMP,Y
                mov     MASKA,TEMP0
                com     MASKA
                and     TEMP,MASKA
                mov     MASKA,TEMP0
                and     MASKA,BARVA
                or      TEMP,MASKA
                st      Y+,TEMP
                mov     TEMP,TEMP2
ESCRSTRED:      subi    TEMP,0x01
                brlo    ESCREND
```

```
                        st          Y+,BARVA
                        rjmp        ESCRSTRED
ESCREND:                ld          TEMP,Y
                        mov         MASKA,TEMP4
                        com         MASKA
                        and         TEMP,MASKA
                        mov         MASKA,TEMP4
                        and         MASKA,BARVA
                        or          TEMP,MASKA
                        st          Y,TEMP
                        rjmp        ESCRLOOP
ZPETLOOP:               rjmp        LOOP
;*******************
;vykresleni jednoho bodu
ESCP:                   rcall       VEMXY
                        rcall       UPRAVXY
                        rcall       POINT
                        rjmp        LOOP
POINT:                  ldi         TEMP2,0x88
                        and         TEMP2,BARVA
                        ldi         TEMP4,0x77
ESCPLOOP:               tst         TEMP3
                        breq        ESCPCON
                        lsr         TEMP2
                        sec
                        ror         TEMP4
                        dec         TEMP3
                        rjmp        ESCPLOOP
ESCPCON:                ld          TEMP,Y
                        and         TEMP,TEMP4
                        or          TEMP,TEMP2
                        st          Y,TEMP
                        ret
;*******************
;vymazani obrazovky
ESCS:                   rcall       CLRSCR
                        sbrc        ZACPIS,6
                        rjmp        ESCSB
                        clr         POZX1
                        clr         POZY1
                        rjmp        LOOP
ESCSB:                  clr         POZX2
                        clr         POZY2
                        rjmp        LOOP
;*******************
;nastaveni x pozice kursoru
ESCX:       rcall           GETCH
                        cpi         ZNAK,0x2A
                        brcs        ESCX2
                        rjmp        LOOP
ESCX2:                  ldi         TEMP,0x06
                        mul         TEMP,ZNAK
                        sbrs        ZACPIS,6
                        mov         POZX1,REG0
                        sbrc        ZACPIS,6
                        mov         POZX2,REG0
                        rjmp        LOOP
;*******************
;nastaveni y pozice kursoru
ESCY:                   rcall       GETCH
                        cpi         ZNAK,0x20
                        brcs        ESCY2
                        rjmp        LOOP
ESCY2:                  ldi         TEMP,0x08
                        mul         TEMP,ZNAK
                        sbrs        ZACPIS,6
                        mov         POZY1,REG0
                        sbrc        ZACPIS,6
                        mov         POZY2,REG0
                        rjmp        LOOP
```

```
;*******************
;vypise velky znak
ESCZ:           rcall       GETCH
                cpi         ZNAK,0xEB
                brcc        ESCZK2
                mov         YL,ZNAK
                rcall       GETCH
                cpi         ZNAK,0xE3
                brcc        ESCZK1
                mov         YH,ZNAK
                lsr         YH
                ror         YL
                lsr         YH
                ror         YL
                sbrs        STAV,0
                ldi         TEMP,0x80
                sbrc        STAV,0
                ldi         TEMP,0xC0
                add         YH,TEMP
                rcall       GETCH
                ldi         TEMP,8          ;nastaveni zacatku znaku v tabulce znaku
                mul         TEMP,ZNAK
                ldi         TEMP,0x10
                add         R1,TEMP
                movw        ZL,R0
                ldi         TEMP,7              ;presouvam znaky
DALSIRADEK:     ldi         COUNTER,4
                lpm         TEMP2,Z+            ;vyber z tabulky znaku 1 Byte
YBITY:          clr         TEMP4
                sbrc        TEMP2,7
                mov         TEMP4,BARVA
                st          Y+,TEMP4
                clr         TEMP4
                sbrc        TEMP2,6
                mov         TEMP4,BARVA
                st          Y+,TEMP4
                clr         TEMP4
                sbrc        TEMP2,5
                mov         TEMP4,BARVA
                st          Y+,TEMP4
                clr         TEMP4
                sbrc        TEMP2,4
                mov         TEMP4,BARVA
                st          Y+,TEMP4
                clr         TEMP4
                sbrc        TEMP2,3
                mov         TEMP4,BARVA
                st          Y+,TEMP4
                adiw        Y,59                ; tady bylo YL
                dec         COUNTER
                brne        YBITY
                dec         TEMP
                brne        DALSIRADEK
                rjmp        LOOP
ESCZK2:         rcall       GETCH
ESCZK1:         rcall       GETCH
                rjmp        LOOP
;*****************************************************************************
;podprogramy
;*****************************************************************************
;ceka na prichod znaku, ulozi ho do ZNAK
;vstup dat: ---
;pouziva: ---
;vola: ---
GETCH:          cp          ZAPISBUF,CTIBUF     ; 1
                breq        GETCH               ; 1
                push        ZL                  ; 2
                push        ZH                  ; 2
                mov         ZL,CTIBUF           ; 1
                ldi         ZH,0x1              ; 1
```

```
                    ld      ZNAK,Z          ; 2
                    inc     CTIBUF          ; 1
                    pop     ZH              ; 2
                    pop     ZL              ; 2
                    ret
;vypise na obrazovku znak a upravi POZX, POZY
;vstup dat: ZNAK, POZX, POZY
;pouziva: TEMP,
;vola: SAVEZNAK
FORMFEED1:          sbrc    ZACPIS,6
                    rjmp    FORMFEED1A
                    clr     POZX1
                    clr     POZY1
                    rjmp    CLRSCR
FORMFEED1A:         clr     POZX2
                    clr     POZY2
                    rjmp    CLRSCR
WRITE:              sbrc    ZACPIS,6
                    rjmp    WRITE2
;OBRAZOVKA1
WRITE1:             cpi     ZNAK,0x8        ;testuju BACKSPACE
                    breq    BACKSPACE1
                    cpi     ZNAK,0x0C       ;testuju FORMFEED
                    breq    FORMFEED1
                    cpi     ZNAK,0x9        ;testuju TAB
                    breq    TAB1
                    cpi     ZNAK,0xA        ;testuju LineFeed
                    breq    KONECFCE
                    cpi     ZNAK,0xD        ;testuju ENTER
                    breq    ENTER1
                    rcall   SAVEZNAK
; test, zda zbyva misto pro posun na radku 6 je zapsany znak + na dalsi volny
                    ldi     TEMP,14
                    add     TEMP,POZX1
                    brcc    PRICTIPOZ1
; neni misto, musim posunou radek na dalsi
ENTER1:             clr     POZX1
                    ldi     TEMP,8
                    add     POZY1,TEMP
; pokud jsem se nedostal na 248, muzu ukoncit
                    brcc    ENDWRITE1
; musim pouzit nejstarsi radek
                    clr     YL              ;mazu nejstarsi zobrazovany radek
                    mov     YH,ZACOBR1
                    clr     TEMP
                    clr     TEMP2
CLRBYTE1:           st      Y+,TEMP2
                    st      Y+,TEMP2
                    dec     TEMP
                    brne    CLRBYTE1
; radek je vymazany, musim odrolovat obrazovku
                    inc     ZACOBR1         ;roluju obrazovku 2x
                    inc     ZACOBR1
; a kontroluju, zda nejsem pres C000hex
                    sbrs    YH,6
                    rjmp    SETPOZY1
                    ldi     TEMP,0x80
                    mov                     ZACOBR1,TEMP
; ukazatel Y nechavam na plne obrazovce = 248 linka
SETPOZY1:           ldi     TEMP,248
                    mov     POZY1,TEMP
ENDWRITE1:          ret
PRICTIPOZ1:         ldi     TEMP,6
                    add     POZX1,TEMP
KONECFCE:           ret
;obsluha tabulatoru
TAB1:               ldi     TEMP,40         ;testuju zda je misto pro posledni TAB
                    add     TEMP,POZX1
                    brcc    TABRUN1
                    ret
```

```
TABRUN1:        clr     TEMP
                ldi     TEMP2,36
TABLOOP1:       add     TEMP,TEMP2
                mov     TEMP3,POZX1
                sub     TEMP3,TEMP
                brcc    TABLOOP1
                mov     POZX1,TEMP
                ret
;obsluha backspace
BACKSPACE1:     mov     TEMP,POZX1
                subi    TEMP,0x06
                brcc    BACKRUN1
                ret
BACKRUN1:       mov     POZX1,TEMP
                push    STAV
                andi    STAV,0xFB
                rcall   SAVEZNAK
                pop     STAV
                ret
;*****************************
;OBRAZOVKA2
FORMFEED2:      sbrc    ZACPIS,6
                rjmp    FORMFEED2A
                clr     POZX1
                clr     POZY1
                rjmp    CLRSCR
FORMFEED2A:     clr     POZX2
                clr     POZY2
                rjmp    CLRSCR
WRITE2:         cpi     ZNAK,0x8        ;testuju BACKSPACE
                breq    BACKSPACE2
                cpi     ZNAK,0xC        ;testuju FORMFEED
                breq    FORMFEED2
                cpi     ZNAK,0x9        ;testuju TAB
                breq    TAB2
                cpi     ZNAK,0xA        ;testuju LineFeed
                breq    KONECFCE
                cpi     ZNAK,0xD        ;testuju ENTER
                breq    ENTER2
                rcall   SAVEZNAK
; test, zda zbyva misto pro posun na radku 6 je zapsany znak + na dalsi volny
                ldi     TEMP,14
                add     TEMP,POZX2
                brcc    PRICTIPOZ2
; neni misto, musim posunou radek na dalsi
ENTER2:         clr     POZX2
                ldi     TEMP,8
                add     POZY2,TEMP
; pokud jsem se nedostal na 248, muzu ukoncit
                brcc    ENDWRITE2
; musim pouzit nejstarsi radek
                clr     YL              ;mazu nejstarsi zobrazovany radek
                mov     YH,ZACOBR2
                clr     TEMP
                clr     TEMP2
CLRBYTE2:       st      Y+,TEMP2
                st      Y+,TEMP2
                dec     TEMP
                brne    CLRBYTE2
; radek je vymazany, musim odrolovat obrazovku
                inc     ZACOBR2         ;roluju obrazovku 2x
                inc     ZACOBR2
; a kontroluju, zda nejsem pres FFFF hex
                sbrc    YH,7
                rjmp    SETPOZY2
                ldi     TEMP,0xC0
                mov                     ZACOBR2,TEMP
; ukazatel Y nechavam na plne obrazovce = 248 linka
SETPOZY2:       ldi     TEMP,248
                mov     POZY2,TEMP
```

```
ENDWRITE2:      ret
PRICTIPOZ2:     ldi     TEMP,6
                add     POZX2,TEMP
                ret
;obsluha tabulatoru
TAB2:           ldi     TEMP,40         ;testuju zda je misto pro posledni TAB
                add     TEMP,POZX2
                brcc    TABRUN2
                ret
TABRUN2:        clr     TEMP
                ldi     TEMP2,36
TABLOOP2:       add     TEMP,TEMP2
                mov     TEMP3,POZX2
                sub     TEMP3,TEMP
                brcc    TABLOOP2
                mov     POZX2,TEMP
                ret
;obsluha backspace
BACKSPACE2:     mov     TEMP,POZX2
                subi    TEMP,6
                brcc    BACKRUN2
                ret
BACKRUN2:       mov     POZX2,TEMP
                push    STAV
                andi    STAV,0xFB
                rcall   SAVEZNAK
                pop     STAV
                ret
;vybere s tabulky a ulozi do externi pamneti znak
;vstup dat: Z(pozice pro vyber z tabulky), Y(pozice pro ulozeni do ext pamneti),
;               ZNAK, POZX, POZY
;pouziva: TEMP, TEMP2, TEMP3, TEMP4, MASKA
SAVEZNAK:       clr     TEMP3
                sbrc    ZACPIS,6        ;rozhodnuti pro kterou obrazovku se ma kreslit
                rjmp    SAVEOBR2
                mov     YL,POZX1
                mov     YH,POZY1
                rjmp    SAVEZNAKU
SAVEOBR2:       mov     YL,POZX2
                mov     YH,POZY2
SAVEZNAKU:      lsr     YH              ;Y .. Y div 4
                ror     YL
                lsr     YH
                ror     YL
                ror     TEMP3           ;TEMP3.7 .. priznak pro posun
                ldi     TEMP,0x00       ;MASKA .. maska pro prvni Byte
                sbrc    TEMP3,7
                ldi     TEMP,0xCC
                mov     MASKA,TEMP
                ldi     TEMP,0x33
                sbrc    TEMP3,7
                ldi     TEMP,0x00
                mov     TEMP3,TEMP      ;TEMP3 .. maska pro druhy Byte
;uprava pozice pro ulozeni do pameti
                sbrc    ZACPIS,6
                rjmp    SAVEINI2
                add     YH,ZACOBR1
; ted musi testovat, zda nejsem pres rozsah 16kB video pameti
                sbrs    YH,6
                rjmp    SETPAM
                ldi     TEMP,0xBF
                and     YH,TEMP
                rjmp    SETPAM
SAVEINI2:       add     YH,ZACOBR2
;ted musi testovat, zda nejsem pres rozsah 16kB video pameti
                sbrc    YH,7
                rjmp    SETPAM
                ldi     TEMP,0xC0
                or      YH,TEMP
;vyber znaku z tabulky + priprava adresy do Z
```

```
SETPAM:         ldi     TEMP,8          ;nastaveni zacatku znaku v tabulce znaku
                mul     TEMP,ZNAK
                ldi     TEMP,0x10
                add     R1,TEMP
                movw    ZL,R0
                ldi     TEMP,0x8
                mov     TEMP0,TEMP      ;presouvam znaky - pocitadlo osmi radku
;zde zacina smycka
LOADZNAK:       lpm     TEMP2,Z+            ;TEMP2 .. zobrazovana data
                andi    TEMP2,0xFC
                sbrc    TEMP3,0
                rjmp    LOADZNAKA
                lsr     TEMP2               ;korekce dat v TEMP2 podle posunuti
                lsr     TEMP2
LOADZNAKA:      mov     TEMP4,TEMP2
                andi    TEMP4,0x0F          ;TEMP4 .. data pro druhy Byte
                mov     TEMP,TEMP4
                swap    TEMP
                or      TEMP4,TEMP
                andi    TEMP2,0xF0          ;TEMP2 .. data pro prvni Byte
                mov     TEMP,TEMP2
                swap    TEMP
                or      TEMP2,TEMP
                ld      TEMP,Y
                sbrs    STAV,2
                and     TEMP,MASKA
                and     TEMP2,BARVA
                or      TEMP,TEMP2
                st      Y+,TEMP             ;ulozeni prvniho Bytu
                ld      TEMP,Y
                sbrs    STAV,2
                and     TEMP,TEMP3
                and     TEMP4,BARVA
                or      TEMP,TEMP4
                st      Y+,TEMP             ;ulozeni druheho Bytu
                adiw    Y,0x3E
                dec     TEMP0
                brne    LOADZNAK
                ret
;podprogram pro vymazani obrazovky
;vstup dat: -
;pouziva: TEMP, TEMP2, TEMP3, Y
CLRSCR:         mov     YH,ZACPIS           ;pocatecni misto pro vyber z pamneti
                sbrc    ZACPIS,6
                mov     ZACOBR2,YH
                sbrs    ZACPIS,6
                mov     ZACOBR1,YH
                ldi     YL,0x00
                clr     TEMP3
                ldi     TEMP2,64
DATA1:          clr     TEMP                ;provede 64 znaku x 256 radku
DATA2:          st      Y+,TEMP3            ;tam kde ukazuje Y sype 0
                dec     TEMP
                brne    DATA2
                dec     TEMP2
                brne    DATA1
                ret
;podprogram pro nasteni souradnic X a Y,
;vystup dat: Y .. adresa pameti; TEMP3 .. zbytek po deleni 4 (posun v Bytu)
;vstup dat: -
;pouziva: TEMP3, Y, GETCH
VEMXY   :       rcall   GETCH
                mov     YL,ZNAK
                rcall   GETCH
                mov     YH,ZNAK
                ret
UPRAVXY:        clr     TEMP3
                lsr     YH
                ror     YL
                ror     TEMP3
```

```
                        lsr             YH
                        ror             YL
                        rol             TEMP3
                        rol             TEMP3
                        sbrs            ZACPIS,6
                        ldi             TEMP,0x80
                        sbrc            ZACPIS,6
                        ldi             TEMP,0xC0
                        add             YH,TEMP
                        ret
;****************
;slouzi k vypsani textu na obrazovku
;vstup dat: Z .. ukazatel do tabulky na text ukonceny 0x00
;vystup dat: -
;pouziva: ZNAK, Z,
VYPISTXT:               lpm             ZNAK,Z+
                        tst             ZNAK
                        breq            VYPISTXTE
                        push            ZL
                        push            ZH
                        rcall           WRITE
                        pop             ZH
                        pop             ZL
                        rjmp            VYPISTXT
VYPISTXTE:              ret
;*******************************************************************************
;uvodni vypis
.org                    UVOD
.dw                     0x6C41                          ;Al
.dw                     0x6870                          ;ph
.dw                     0x6E61                          ;an
.dw                     0x6D75                          ;um
.dw                     0x7265                          ;er
.dw                     0x6369                          ;ic
.dw                     0x2620                          ;&
.dw                     0x6720                          ;g
.dw                     0x6172                          ;ra
.dw                     0x6870                          ;ph
.dw                     0x6369                          ;ic
.dw                     0x5420                          ;T
.dw                     0x2056                          ;V
.dw                     0x6964                          ;di
.dw                     0x7073                          ;sp
.dw                     0x616C                          ;la
.dw                     0x0D79                          ;y
.dw                     0x4900                          ;I
.dw                     0x706E                          ;np
.dw                     0x7475                          ;ut
.dw                     0x203A                          ;:
.dw                     0x5352                          ;RS
.dw                     0x3332                          ;23
.dw                     0x2C32                          ;2,
.dw                     0x3920                          ;9
.dw                     0x3036                          ;60
.dw                     0x2030                          ;0
.dw                     0x6442                          ;bd
.dw                     0x0D2C                          ;,
.dw                     0x2020
.dw                     0x2020
.dw                     0x2020
.dw                     0x3820                          ;8
.dw                     0x6220                          ;b
.dw                     0x7469                          ;it
.dw                     0x2C73                          ;s,
.dw                     0x6E20                          ;n
.dw                     0x206F                          ;o
.dw                     0x6170                          ;pa
.dw                     0x6972                          ;ri
.dw                     0x7974                          ;ty
.dw                     0x202C                          ;,
```

```
.dw     0x200D
.dw     0x2020
.dw     0x2020
.dw     0x2020
.dw     0x2032          ;2
.dw     0x7473          ;st
.dw     0x706F          ;op
.dw     0x6220          ;b
.dw     0x7469          ;it
.dw     0x2073          ;s
.dw     0x320D          ;2
.dw     0x3635          ;56
.dw     0x3278          ;x2
.dw     0x3635          ;56
.dw     0x7020          ;p
.dw     0x7869          ;ix
.dw     0x6C65          ;el
.dw     0x202C          ;,
.dw     0x6F66          ;fo
.dw     0x7275          ;ur
.dw     0x6320          ;c
.dw     0x6C6F          ;ol
.dw     0x726F          ;or
.dw     0x2073          ;s
.dw     0xB200          ;
.dw     0xB200          ;
.dw     0xB200          ;
.dw     0x0D00          ;
.dw     0x280D          ;(
.dw     0x2963          ;c)
.dw     0x5220          ;R
.dw     0x6461          ;ad
.dw     0x6D69          ;im
.dw     0x5020          ;P
.dw     0x6365          ;ec
.dw     0x6168          ;ha
.dw     0x2C6C          ;l,
.dw     0x7220          ;r
.dw     0x6461          ;ad
.dw     0x6D69          ;im
.dw     0x702E          ;.p
.dw     0x6365          ;ec
.dw     0x6168          ;ha
.dw     0x406C          ;l@
.dw     0x6573          ;se
.dw     0x6E7A          ;zn
.dw     0x6D61          ;am
.dw     0x632E          ;.c
.dw     0x0D7A          ;z
.dw     0x7468          ;ht
.dw     0x7074          ;tp
.dw     0x2F3A          ;:/
.dw     0x722F          ;/r
.dw     0x6461          ;ad
.dw     0x6D69          ;im
.dw     0x782E          ;.x
.dw     0x2E66          ;f.
.dw     0x7A63          ;cz
.dw     0x0000          ;
;tabulka znaku
.org    TABULKA
.dw     0xF8F8          ;netisknutelny znak
.dw     0xF8F8
.dw     0xF8F8
.dw     0x00F8
.dw     0xF8F8          ;netisknutelny znak
.dw     0xF8F8
.dw     0xF8F8
.dw     0x00F8
.dw     0xF8F8          ;netisknutelny znak
```

```
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0x0000    ;backspace
.dw    0x0000
.dw    0x0000
.dw    0x0000
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
.dw    0x00F8
.dw    0xF8F8    ;netisknutelny znak
.dw    0xF8F8
.dw    0xF8F8
```

```
.dw     0x00F8
.dw     0xF8F8          ;netisknutelny znak
.dw     0xF8F8
.dw     0xF8F8
.dw     0x00F8
.dw     0xF8F8          ;netisknutelny znak
.dw     0xF8F8
.dw     0xF8F8
.dw     0x00F8
.dw     0xF8F8          ;netisknutelny znak
.dw     0xF8F8
.dw     0xF8F8
.dw     0x00F8
.dw     0xF8F8          ;netisknutelny znak
.dw     0xF8F8
.dw     0xF8F8
.dw     0x00F8
.dw     0xF8F8          ;netisknutelny znak
.dw     0xF8F8
.dw     0xF8F8
.dw     0x00F8
.dw     0xF8F8          ;netisknutelny znak
.dw     0xF8F8
.dw     0xF8F8
.dw     0x00F8
.dw     0xF8F8          ;netisknutelny znak
.dw     0xF8F8
.dw     0xF8F8
.dw     0x00F8
.dw     0xF8F8          ;netisknutelny znak
.dw     0xF8F8
.dw     0xF8F8
.dw     0x00F8
.dw     0xF8F8          ;netisknutelny znak
.dw     0xF8F8
.dw     0xF8F8
.dw     0x00F8
.dw     0xF8F8          ;netisknutelny znak
.dw     0xF8F8
.dw     0xF8F8
.dw     0x00F8
.dw     0xF8F8          ;netisknutelny znak
.dw     0xF8F8
.dw     0xF8F8
.dw     0x00F8
.dw     0xF8F8          ;netisknutelny znak
.dw     0xF8F8
.dw     0xF8F8
.dw     0x00F8
.dw     0x0000          ;
.dw     0x0000
.dw     0x0000
.dw     0x0000
.dw     0x2020          ;!
.dw     0x2020
.dw     0x0020
.dw     0x0020
.dw     0x5050          ;"
.dw     0x0050
.dw     0x0000
.dw     0x0000
.dw     0x5050          ;#
.dw     0x50F8
.dw     0x50F8
.dw     0x0050
.dw     0x7020          ;$
.dw     0x70A0
.dw     0x7028
.dw     0x0020
.dw     0xD8D0          ;%
```

```
.dw	0x2010
.dw	0x9840
.dw	0x0018
.dw	0x5020		;&
.dw	0x6050
.dw	0x90A8
.dw	0x0068
.dw	0x2020		;´
.dw	0x0040
.dw	0x0000
.dw	0x0000
.dw	0x2010		;(
.dw	0x4040
.dw	0x2040
.dw	0x0010
.dw	0x2040		;)
.dw	0x1010
.dw	0x2010
.dw	0x0040
.dw	0x2000		;*
.dw	0x70A8
.dw	0x20A8
.dw	0x0000
.dw	0x2000		;+
.dw	0xF820
.dw	0x2020
.dw	0x0000
.dw	0x0000		;,
.dw	0x0000
.dw	0x4040
.dw	0x0080
.dw	0x0000		;-
.dw	0xF800
.dw	0x0000
.dw	0x0000
.dw	0x0000		;.
.dw	0x0000
.dw	0x0000
.dw	0x0040
.dw	0x0800		;/
.dw	0x2010
.dw	0x8040
.dw	0x0000
.dw	0x8870		;0
.dw	0xA898
.dw	0x88C8
.dw	0x0070
.dw	0x6020		;1
.dw	0x2020
.dw	0x2020
.dw	0x0070
.dw	0x8870		;2
.dw	0x3008
.dw	0x8040
.dw	0x00F8
.dw	0x08F8		;3
.dw	0x3010
.dw	0x8808
.dw	0x0070
.dw	0x3010		;4
.dw	0x9050
.dw	0x10F8
.dw	0x0010
.dw	0x80F8		;5
.dw	0x08F0
.dw	0x8808
.dw	0x0070
.dw	0x4038		;6
.dw	0xF080
.dw	0x8888
```

```
.dw	0x0070
.dw	0x08F8		;7
.dw	0x2010
.dw	0x4040
.dw	0x0040
.dw	0x8870		;8
.dw	0x7088
.dw	0x8888
.dw	0x0070
.dw	0x8870		;9
.dw	0x7088
.dw	0x1008
.dw	0x00E0
.dw	0x0000		;:
.dw	0x0000
.dw	0x0040
.dw	0x0040
.dw	0x0000		;;
.dw	0x0020
.dw	0x2020
.dw	0x0040
.dw	0x2010		;<
.dw	0x8040
.dw	0x2040
.dw	0x0010
.dw	0x0000		;=
.dw	0x00F8
.dw	0x00F8
.dw	0x0000
.dw	0x2040		;>
.dw	0x0810
.dw	0x2010
.dw	0x0040
.dw	0x8870		;?
.dw	0x1008
.dw	0x0020
.dw	0x0020
.dw	0x8870		;@
.dw	0x6808
.dw	0xA8A8
.dw	0x0070
.dw	0x5020		;A
.dw	0x8888
.dw	0x88F8
.dw	0x0088
.dw	0x88F0		;B
.dw	0xF088
.dw	0x8888
.dw	0x00F0
.dw	0x8870		;C
.dw	0x8080
.dw	0x8880
.dw	0x0070
.dw	0x48F0		;D
.dw	0x4848
.dw	0x4848
.dw	0x00F0
.dw	0x80F8		;E
.dw	0xF080
.dw	0x8080
.dw	0x00F8
.dw	0x80F8		;F
.dw	0xF080
.dw	0x8080
.dw	0x0080
.dw	0x8070		;G
.dw	0x8080
.dw	0x8898
.dw	0x0078
.dw	0x8888		;H
```

```
.dw 0xF888
.dw 0x8888
.dw 0x0088
.dw 0x2070 ;I
.dw 0x2020
.dw 0x2020
.dw 0x0070
.dw 0x0808 ;J
.dw 0x0808
.dw 0x8888
.dw 0x0070
.dw 0x9088 ;K
.dw 0xC0A0
.dw 0x90A0
.dw 0x0088
.dw 0x8080 ;L
.dw 0x8080
.dw 0x8080
.dw 0x00F8
.dw 0xD888 ;M
.dw 0xA8A8
.dw 0x8888
.dw 0x0088
.dw 0x8888 ;N
.dw 0xB8C8
.dw 0x8898
.dw 0x0088
.dw 0x8870 ;O
.dw 0x8888
.dw 0x8888
.dw 0x0070
.dw 0x88F0 ;P
.dw 0xF088
.dw 0x8080
.dw 0x0080
.dw 0x8870 ;Q
.dw 0x8888
.dw 0x90B8
.dw 0x0068
.dw 0x88F0 ;R
.dw 0xF088
.dw 0x90B0
.dw 0x0088
.dw 0x8870 ;S
.dw 0x7080
.dw 0x8808
.dw 0x0070
.dw 0x20F8 ;T
.dw 0x2020
.dw 0x2020
.dw 0x0020
.dw 0x8888 ;U
.dw 0x8888
.dw 0x8888
.dw 0x0070
.dw 0x8888 ;V
.dw 0x5088
.dw 0x2050
.dw 0x0020
.dw 0x8888 ;W
.dw 0xA888
.dw 0xA8A8
.dw 0x0050
.dw 0x8888 ;X
.dw 0x2050
.dw 0x8850
.dw 0x0088
.dw 0x8888 ;Y
.dw 0x2050
.dw 0x2020
```

```
.dw     0x0020
.dw     0x08F8          ;Z
.dw     0x2010
.dw     0x8040
.dw     0x00F8
.dw     0x80E0          ;[
.dw     0x8080
.dw     0x8080
.dw     0x00E0
.dw     0x8000          ;\
.dw     0x2040
.dw     0x0810
.dw     0x0000
.dw     0x0838          ;]
.dw     0x0808
.dw     0x0808
.dw     0x0038
.dw     0x8870          ;^
.dw     0x0000
.dw     0x0000
.dw     0x0000
.dw     0x0000          ;_
.dw     0x0000
.dw     0x0000
.dw     0x00F8
.dw     0x2020          ;'
.dw     0x0020
.dw     0x0000
.dw     0x0000
.dw     0x0000          ;a
.dw     0x0870
.dw     0x8878
.dw     0x0078
.dw     0x8080          ;b
.dw     0xC8B0
.dw     0x8888
.dw     0x00F0
.dw     0x0000          ;c
.dw     0x8070
.dw     0x8880
.dw     0x0070
.dw     0x0808          ;d
.dw     0x9868
.dw     0x8888
.dw     0x0078
.dw     0x0000          ;e
.dw     0x8870
.dw     0x80F8
.dw     0x0070
.dw     0x4830          ;f
.dw     0xE040
.dw     0x4040
.dw     0x0040
.dw     0x7800          ;g
.dw     0x8888
.dw     0x0878
.dw     0x0070
.dw     0x8080          ;h
.dw     0xC8B0
.dw     0x8888
.dw     0x0088
.dw     0x0020          ;i
.dw     0x2060
.dw     0x2020
.dw     0x0070
.dw     0x0010          ;j
.dw     0x1030
.dw     0x9010
.dw     0x0060
.dw     0x8080          ;k
```

```
.dw	0xA090
.dw	0xA0C0
.dw	0x0090
.dw	0x2060		;l
.dw	0x2020
.dw	0x2020
.dw	0x0070
.dw	0x0000		;m
.dw	0xA8D0
.dw	0x88A8
.dw	0x0088
.dw	0x0000		;n
.dw	0xC8B0
.dw	0x8888
.dw	0x0088
.dw	0x0000		;o
.dw	0x8870
.dw	0x8888
.dw	0x0070
.dw	0x0000		;p
.dw	0x88F0
.dw	0x80F0
.dw	0x0080
.dw	0x0000		;q
.dw	0x9868
.dw	0x0878
.dw	0x0008
.dw	0x0000		;r
.dw	0xC8B0
.dw	0x8080
.dw	0x0080
.dw	0x0000		;s
.dw	0x8070
.dw	0x0870
.dw	0x00F0
.dw	0x4040		;t
.dw	0x40E0
.dw	0x4840
.dw	0x0030
.dw	0x0000		;u
.dw	0x8888
.dw	0x9888
.dw	0x0068
.dw	0x0000		;v
.dw	0x8888
.dw	0x5088
.dw	0x0020
.dw	0x0000		;w
.dw	0x8888
.dw	0xA8A8
.dw	0x0050
.dw	0x0000		;x
.dw	0x5088
.dw	0x5020
.dw	0x0088
.dw	0x0000		;y
.dw	0x8888
.dw	0x0878
.dw	0x0070
.dw	0x0000		;z
.dw	0x10F8
.dw	0x4020
.dw	0x00F8
.dw	0x2010		;{
.dw	0x4020
.dw	0x2020
.dw	0x0010
.dw	0x2020		;|
.dw	0x2020
.dw	0x2020
```

```
.dw             0x0020
.dw             0x1020                  ;}
.dw             0x0810
.dw             0x1010
.dw             0x0020
.dw             0xB068                  ;~
.dw             0x0000
.dw             0x0000
.dw             0x0000
.DSEG           ; Start data segment
.dw             0x0707
```

2. Program pro MC9S08QG8

```
/*************************************************************\
*                                                             *
*                        Osciloskop                           *
*                                                             *
* (c) Radim Pechal, radim.pechal@seznam.cz                    *
* http://radim.xf.cz/                                         *
*                                                             *
* v. 1.0 z data 4. 3. 2007                                    *
*                                                             *
\*************************************************************/
#include <hidef.h>
#include "derivative.h"
volatile byte _NVICSTRM @0x0000FFAF;
#define NVICSTRM _NVICSTRM
#define hi_baud 9600
#define fei1_spd 8500000/16/hi_baud
byte pole[250];                                  // Pole hodnot namerenych AD0
byte trigger=0x64;                               // Trigrovaci uroven AD prevodniku
byte stav=0x06;                                  // Nastaveni osciloskopu
// stav.0 hrana 0=vzestupna; 1=sestupna
// stav.1 0=unipolar 1=bipolar
// stav.2 0=synchro 1=free
// stav.3 0=mereni 1=hold
// stav.4 0=free run 1=one shot
// stav.5 0=DC 1=AC
byte timebase=7;
// timebase == 0 ..  50us/div
// timebase == 1 .. 100us/div
// timebase == 2 .. 200us/div
// timebase == 3 .. 500us/div
// timebase == 4 ..   1ms/div
// timebase == 5 ..   2ms/div
// timebase == 6 ..   5ms/div
// timebase == 7 ..  10ms/div
// timebase == 8 ..  20ms/div
// timebase == 9 ..  50ms/div
// timebase ==10 .. 100ms/div
// timebase ==11 .. 200ms/div
// timebase ==12 .. 500ms/div
// timebase ==13 ..   1 s/div
// timebase ==14 ..   2 s/div
byte gain=6;
// gain == 0 ..  10mV/div  zesileni 50   vstup  1 : 1
// gain == 1 ..  20mV/div  zesileni 25   vstup  1 : 1
// gain == 2 ..  50mV/div  zesileni 10   vstup  1 : 1
// gain == 3 .. 100mV/div  zesileni  5   vstup  1 : 1
// gain == 4 .. 200mV/div  zesileni  2.5 vstup  1 : 1
// gain == 5 .. 500mV/div  zesileni  1   vstup  1 : 1
// gain == 6 ..   1 V/div  zesileni 50   vstup  1 : 100
// gain == 7 ..   2 V/div  zesileni 25   vstup  1 : 100
// gain == 8 ..   5 V/div  zesileni 10   vstup  1 : 100
// gain == 9 ..  10 V/div  zesileni  5   vstup  1 : 100
// gain ==10 ..  20 V/div  zesileni  2.5 vstup  1 : 100
// gain ==11 ..  50 V/div  zesileni  1   vstup  1 : 100
byte Xhold;                                      // Promenna urcujici X pozici pro Hold
byte screen=0;                                   // Pomocna promenna pro Free
byte mergain;
byte mertimebase;
byte obrX=0xF;
byte obrY=0xF;
void feedCOP() {
    if (PTAD_PTAD5 == 1)
        SRS = 1;
    if (SPMSC1_LVDF == 1) {
        PTBD = PTBD^0x10;
        SPMSC1_LVDACK = 1;
    }
```

```
}
// Funkce pro prepinani mezi vnitnimi hodinami
void ics_intclk(byte C1, byte C2) {
    byte i;
    if (NVICSTRM != 0xFF) {
        ICSTRM = NVICSTRM;                          // nacteni trimovaciho registru
    }
    ICSC1 = C1;
    for (i=0x10;i>0;i--){
        feedCOP();
    }
    ICSC2 = C2;
     while (ICSC1_CLKS != ICSSC_CLKST) {}
}
void SendChar(char znak) {
    byte i;
    i = SCIS1;                                       // Nastartovani serioveho prenosu
    SCID = znak;                                     // Odeslani znaku
    while(!SCIS1_TDRE){                              // Ocekavani pro odeslani znaku
        feedCOP();
    };
    while(!SCIS1_TC){
        feedCOP();
    };
}
void SendMsg(char msg[]) {
    byte ix=0;
    byte dummy;
    byte nxt_char;
    dummy = SCIS1;
    nxt_char = msg[ix++];
    while(nxt_char != 0x00) {
        SCID = nxt_char;
        nxt_char = msg[ix++];
        while(!SCIS1_TDRE){
            feedCOP();
        };
    }
    while(!SCIS1_TC){
        feedCOP();
    };
}
// *******************************************************
// *                FUNKCE PRO CEKANI                    *
// *******************************************************
void Cekej10ms(void) {
    byte i,j;
    for(i=0; i<19; i++)
        for(j=0; j<255; j++) asm{
            STA    SRS
        }
}
void Cekej20ms(void) {
    byte i,j;
    for(i=0; i<39; i++)
        for(j=0; j<255; j++) asm{
            STA    SRS
        }
}
void Cekej100ms(void){
    byte i,j;
    for(i=0; i<196; i++)
        for(j=0; j<255; j++) asm{
            STA    SRS
        }
}
// *******************************************************
// *                FUNKCE PRO MERENI                    *
// *******************************************************
byte ADquickSynchro() {
```

```
   word ukazat;
   byte counter1, counter2, counter3;
   counter1 = 0;
   counter2 = 0;
   counter3 = 10;
// Pretypuji adresu pole (prekladac v ASM casti to odmital)
   ukazat=(unsigned int)pole;
    if (stav & 0x1)
      asm{
      ; Zakaz preruseni
            CLI
      ; Priprav ukazatel na zacatek pole
            LDHX   ukazat
      ; Cekani na komparacni uroven pri vzestupne hrane
            LDA    trigger
      WAITL:  STA    SRS
            DBNZ   counter1,WAITLA
            DBNZ   counter2,WAITLA
            DBNZ   counter3,WAITLA
            JMP    ERTRIG
      WAITLA: CMP    ADCRL
            BLO    WAITL
      ; ... a pak vyskocit nad komparacni uroven
      WAITH:  STA    SRS
            DBNZ   counter1,WAITHA
            DBNZ   counter2,WAITHA
            DBNZ   counter3,WAITHA
            JMP    ERTRIG
      WAITHA: CMP    ADCRL
            BHS    WAITH
      }
      else
      asm{
      ; Zakaz preruseni
            CLI
      ; Priprav ukazatel na zacatek pole
            LDHX   ukazat
      ; Cekani na komparacni uroven pri sestupne hrane
            LDA    trigger
      WAITSH: STA    SRS
            DBNZ   counter1,WAITSHA
            DBNZ   counter2,WAITSHA
            DBNZ   counter3,WAITSHA
            JMP    ERTRIG
      WAITSHA:CMP    ADCRL
            BHS    WAITSH
      ; ... a pak spadnout pod komparacni uroven
      WAITSL: STA    SRS
            DBNZ   counter1,WAITSLA
            DBNZ   counter2,WAITSLA
            DBNZ   counter3,WAITSLA
            JMP    ERTRIG
      WAITSLA:CMP    ADCRL
            BLO    WAITSL
      }
      asm {
      ; "A" registr pouziju jako pocitadlo
       MERENI: LDA    #$FA                          ; Pocitadlo nastaveno na 250
      ; A zaznamenam data (po 17 Bus cyklech) do pameti
      LOOP:   MOV    ADCRL,X+   ; 5 Bus cycles
            NOP              ; 1 (1)
            NOP              ; 1 (2)
            NOP              ; 1 (3)
            NOP              ; 1 (4)
            NOP              ; 1 (5)
            NOP              ; 1 (6)
            NOP              ; 1 (7)
            NOP              ; 1 (8)
            DBNZA   LOOP     ; 4 Bus cycles
      ; Povolim preruseni
```

```
            SEI
        ; Ulozeni navratove hodnoty do counter1
            LDA     #0x0
            JMP     KONEC
        ERTRIG: LDA     #0x1
        KONEC:  STA     counter1
        }
    return(counter1);
}
byte ADnormalSynchro(byte speed) {
    word ukazat;
    byte i, counter1, counter2, counter3;
    counter1 = 0;
    counter2 = 0;
    counter3 = 10;
// Pretypuji adresu pole (prekladac v ASM casti to odmital)
    ukazat=(unsigned int)pole;
    if (stav & 0x1)
        asm{
        ; Zakaz preruseni
            CLI
        ; Priprav ukazatel na zacatek pole
            LDHX   ukazat
        ; Cekani na komparacni uroven pri vzestupne hrane
            LDA    trigger
        WAITL:  STA    SRS
            DBNZ   counter1,WAITLA
            DBNZ   counter2,WAITLA
            DBNZ   counter3,WAITLA
            JMP    ERTRIG
        WAITLA: CMP    ADCRL
            BLO    WAITL
        ; ... a pak vyskocit nad komparacni uroven
        WAITH:  STA    SRS
            DBNZ   counter1,WAITHA
            DBNZ   counter2,WAITHA
            DBNZ   counter3,WAITHA
            JMP    ERTRIG
        WAITHA: CMP    ADCRL
            BHI    WAITH
        }
        else
        asm{
        ; Zakaz preruseni
            CLI
        ; Priprav ukazatel na zacatek pole
            LDHX   ukazat
        ; Cekani na komparacni uroven pri sestupne hrane
            LDA    trigger
        WAITSH: STA    SRS
            DBNZ   counter1,WAITSHA
            DBNZ   counter2,WAITSHA
            DBNZ   counter3,WAITSHA
            JMP    ERTRIG
        WAITSHA:CMP    ADCRL
            BHI    WAITSH
        ; ... a pak spadnout pod komparacni uroven
        WAITSL: STA    SRS
            DBNZ   counter1,WAITSLA
            DBNZ   counter2,WAITSLA
            DBNZ   counter3,WAITSLA
            JMP    ERTRIG
        WAITSLA:CMP    ADCRL
            BLO    WAITSL
        }
        asm {
        ; "A" registr pouziju jako pocitadlo
            LDA    #$FA      ; Pocitadlo nastaveno na 250
        ; A zaznamenam data (po 17 Bus cyklech) do pameti
        LOOPL:  MOV    ADCRL,X+                    ; 5 Bus cycles
```

```
            STA   i                         ; 5
            LDA   speed                     ; 5 (15)
            NOP                             ; 1 (1)
            NOP                             ; 1 (2)
            NOP                             ; 1 (3)
            NOP                             ; 1 (4)
            NOP                             ; 1 (5)
            NOP                             ; 1 (6)
            NOP                             ; 1 (7)
            NOP                             ; 1 (8)
            NOP                             ; 1 (9)
            DECA                            ; 1
            BEQ   LOOPLC                    ; 3 (25)
        LOOPS:  NOP             ; 1 (1)
            NOP                             ; 1 (2)
            NOP                             ; 1 (3)
            NOP                             ; 1 (4)
            NOP                             ; 1 (5)
            NOP                             ; 1 (6)
            NOP                             ; 1 (7)
            NOP                             ; 1 (8)
            NOP                             ; 1 (9)
            NOP                             ; 1 (10)
            NOP                             ; 1 (11)
            NOP                             ; 1 (12)
            NOP                             ; 1 (13)
            DBNZA   LOOPS                   ; 4
        LOOPLC: LDA     i                   ; 5
            DBNZA   LOOPL                   ; 4 Bus cycles (34)
        ; Povolim preruseni
            SEI
        ; Ulozeni navratove hodnoty do counter1
            LDA     #0x0
            JMP     KONEC
        ERTRIG: LDA     #0x1
        KONEC:  STA     counter1
        }
    return(counter1);
}
Bool ADhigher(void){
    byte i;
    for (i = 0; i<10; i++){
        if (ADCRL<trigger) return(0);
        asm{
            STA   SRS
            STA   SRS
            STA   SRS
            STA   SRS
            STA   SRS
        }
    }
    return(1);
}
Bool ADlower(void){
    byte i;
    for (i = 0; i<10; i++){
        if (ADCRL>trigger) return(0);
        asm{
            STA   SRS
            STA   SRS
            STA   SRS
            STA   SRS
            STA   SRS
        }
    }
    return(1);
}
byte ADslowSynchro(void){
    byte i;
    word counter;
```

```
    counter = 10000;
    if (stav & 0x01){
        while ((ADhigher())&&(counter>0)) counter--;
        if (counter) while ((ADlower())&&(counter>0)) counter--;
    } else {
        while ((ADlower())&&(counter>0)) counter--;
        if (counter) while ((ADhigher())&&(counter>0)) counter--;
    }
    if (counter){
// Nastaveni casovace
        MTIMSC = 0x30;
        MTIMMOD = 0x6A;
        MTIMCLK = 0x06;
        MTIMSC = 0x00;
        for (i = 0; i<250; i++){
            pole[i] = ADCRL;
            while (!(MTIMSC & 0x80)) SRS = 42;
            MTIMSC = 0;
        }
         MTIMSC = 0x30;
    }
    return(!((byte)counter));
}
byte ADveryslowSynchro(byte t){
    byte i,n;
    word counter;
// Nastaveni zobrazovani nezobrazovane obrazovky
    if (!(stav & 0x08)){
        SendChar(0x1B);
        SendChar('V');
        if (screen == 1) SendChar('1');
        else SendChar('2');
    }
    Cekej20ms();
    SendChar(0x1B);
    SendChar('G');
    SendChar(0x03);
    SendChar(0x0FA);
    counter = 10000;
    if (stav & 0x01){
        while ((ADhigher())&&(counter>0)) counter--;
        if (counter) while ((ADlower())&&(counter>0)) counter--;
    } else {
        while ((ADlower())&&(counter>0)) counter--;
        if (counter) while ((ADhigher())&&(counter>0)) counter--;
    }
    if (counter){
// Nastaveni casovace
    MTIMSC = 0x30;
    MTIMMOD = 0x85;
    MTIMCLK = 0x07;
    MTIMSC = 0x00;
    for (i = 0; i<250; i++){
        pole[i] = ADCRL;
        if (stav & 0x2){
            if (pole[i]>(byte)200) SendChar(10);
            else SendChar((byte)((byte)200-(byte)pole[i])+(byte)10);
        } else{
            n = ((pole[i] * 158)>>8) * 2;
            if (n>(byte)200) SendChar(10);
            else SendChar((byte)((byte)200-(byte)n)+(byte)10);
        }
            for (n = 0; n<t ; n++){
                while (!(MTIMSC & 0x80)) SRS = 42;
                MTIMSC = 0;
            }
        }
        MTIMSC = 0x30;
    } else {
        if (stav & 0x01) for (i = 0; i<250; i++) SendChar(210);
```

```
        else for (i = 0; i<250; i++) SendChar(10);
        SendMsg("No Synchro");
    }
    return((byte)counter);
}
byte ADSynchro(void) {
    byte i;
    switch (timebase){
        case  0 : i = ADquickSynchro();      break;
        case  1 : i = ADnormalSynchro(1);    break;
        case  2 : i = ADnormalSynchro(4);    break;
        case  3 : i = ADnormalSynchro(9);    break;
        case  4 : i = ADnormalSynchro(19);   break;
        case  5 : i = ADnormalSynchro(49);   break;
        case  6 : i = ADnormalSynchro(99);   break;
        case  7 : i = ADnormalSynchro(199);  break;
        case  8 : i = ADslowSynchro();       break;
        case  9 : i = ADveryslowSynchro(1);  break;
        case 10 : i = ADveryslowSynchro(2);  break;
        case 11 : i = ADveryslowSynchro(4);  break;
        case 12 : i = ADveryslowSynchro(10); break;
        case 13 : i = ADveryslowSynchro(20); break;
        case 14 : i = ADveryslowSynchro(40); break;
    }
    return i;
}
// Nejrychlejsi sejmuti signalu
void ADquickFree() {
    word ukazat;
// Pretypuji adresu pole (prekladac v ASM casti to odmital)
    ukazat=(unsigned int)pole;
    asm {
    ; Zakaz preruseni
                CLI
   ; Priprav ukazatel na zacatek pole
                LDHX   ukazat
   ; "A" registr pouziju jako pocitadlo
   MERENI: LDA    #$FA                    ; Pocitadlo nastaveno na 250
   ; A zaznamenam data (po 17 Bus cyklech) do pameti
   LOOP:   MOV    ADCRL,X+                ; 5 Bus cycles
           NOP                            ; 1 (1)
           NOP                            ; 1 (2)
           NOP                            ; 1 (3)
           NOP                            ; 1 (4)
           NOP                            ; 1 (5)
           NOP                            ; 1 (6)
           NOP                            ; 1 (7)
           NOP                            ; 1 (8)
           DBNZA   LOOP                   ; 4 Bus cycles
    ; Povolim preruseni
           SEI
    }
}
// Sejmuti signalu se spomalenim podle konstanty
void ADnormalFree(byte speed) {
    word ukazat;
    byte i;
// Pretypuji adresu pole (prekladac v ASM casti to odmital)
    ukazat=(unsigned int)pole;
    asm {
    ; Zakaz preruseni
                CLI
    ; Priprav ukazatel na zacatek pole
                LDHX   ukazat
    ; "A" registr pouziju jako pocitadlo
                LDA    #$FA                ; Pocitadlo nastaveno na 250
    ; A zaznamenam data (po 17 Bus cyklech) do pameti
     LOOPL:  MOV    ADCRL,X+                ; 5 Bus cycles
                STA    i                   ; 5
                LDA    speed               ; 5 (15)
```

```
                NOP                                     ; 1 (1)
                NOP                                     ; 1 (2)
                NOP                                     ; 1 (3)
                NOP                                     ; 1 (4)
                NOP                                     ; 1 (5)
                NOP                                     ; 1 (6)
                NOP                                     ; 1 (7)
                NOP                                     ; 1 (8)
                NOP                                     ; 1 (9)
                DECA                                    ; 1
                BEQ    LOOPLC                           ; 3 (25)
    LOOPS:  NOP                                         ; 1 (1)
                NOP                                     ; 1 (2)
                NOP                                     ; 1 (3)
                NOP                                     ; 1 (4)
                NOP                                     ; 1 (5)
                NOP                                     ; 1 (6)
                NOP                                     ; 1 (7)
                NOP                                     ; 1 (8)
                NOP                                     ; 1 (9)
                NOP                                     ; 1 (10)
                NOP                                     ; 1 (11)
                NOP                                     ; 1 (12)
                NOP                                     ; 1 (13)
                DBNZA   LOOPS                           ; 4
    LOOPLC: LDA     i                                   ; 5
                DBNZA   LOOPL                           ; 4 Bus cycles (34)
    ; Povolim preruseni
                SEI
    }
}
void ADslowFree(void){
    byte i;
// Nastaveni casovace
    MTIMSC = 0x30;
    MTIMMOD = 0x6A;
    MTIMCLK = 0x06;
    MTIMSC = 0x00;
    for (i = 0; i<250; i++){
        pole[i] = ADCRL;
        while (!(MTIMSC & 0x80)) SRS = 42;
        MTIMSC = 0;
    }
    MTIMSC = 0x30;
}
void ADveryslowFree(byte t){
    byte i,n;
// Nastaveni zobrazovani nezobrazovane obrazovky
    if (!(stav & 0x08)){
        SendChar(0x1B);
        SendChar('V');
        if (screen == 1) SendChar('1');
            else SendChar('2');
    }
    Cekej20ms();
    SendChar(0x1B);
    SendChar('G');
    SendChar(0x03);
    SendChar(0x0FA);
// Nastaveni casovace
    MTIMSC = 0x30;
    MTIMMOD = 0x85;
    MTIMCLK = 0x07;
    MTIMSC = 0x00;
    for (i = 0; i<250; i++){
        pole[i] = ADCRL;
        if (stav & 0x2){
            if (pole[i]>(byte)200) SendChar(10);
                else SendChar((byte)((byte)200-(byte)pole[i])+(byte)10);
        } else{
```

```
        n = ((pole[i] * 158)>>8) * 2;
        if (n>(byte)200) SendChar(10);
            else SendChar((byte)((byte)200-(byte)n)+(byte)10);
    }
    for (n = 0; n<t ; n++){
            while (!(MTIMSC & 0x80)) SRS = 42;
            MTIMSC = 0;
        }
    }
    MTIMSC = 0x30;
}
void ADFree(void) {
    switch (timebase) {
        case  0 : ADquickFree();      break;
        case  1 : ADnormalFree(1);    break;
        case  2 : ADnormalFree(4);    break;
        case  3 : ADnormalFree(9);    break;
        case  4 : ADnormalFree(19);   break;
        case  5 : ADnormalFree(49);   break;
        case  6 : ADnormalFree(99);   break;
        case  7 : ADnormalFree(199);  break;
        case  8 : ADslowFree();       break;
        case  9 : ADveryslowFree(1);  break;
        case 10 : ADveryslowFree(2);  break;
        case 11 : ADveryslowFree(4);  break;
        case 12 : ADveryslowFree(10); break;
        case 13 : ADveryslowFree(20); break;
        case 14 : ADveryslowFree(40); break;
    }
}
// *********************************************************
// *                   FUNKCE GRAF                         *
// *********************************************************
void HoldValue(byte X);
void Ramec(void);
void Graf(void) {
    byte i=0;
    word n;
// Nastaveni zapisu na nezobrazovanou obrazovku a vymalovani Ramce
    SendChar(0x1B);
    SendChar('W');
    if (screen == 1) {
        SendChar('2');
        screen = 0;
    }
    else {
        screen = 1;
        SendChar('1');
    }
    Ramec();
// Provedeni mereni
    if (!(stav & 0x08)){
        if (stav & 0x4) ADFree();
            else i=ADSynchro();
        mertimebase=timebase;
        mergain=gain;
    }
// Nastaveni zobrazovani nezobrazovane obrazovky pokud se merilo
    if ((!(stav & 0x08))&&(timebase<9)){
        SendChar(0x1B);
        SendChar('V');
        if (screen == 1) SendChar('1');
            else SendChar('2');
    }
    Cekej20ms();
// Vykresleni hodnot
    if (((!i)&&(timebase<9))||(stav & 0x08)) {
        SendChar(0x1B);
        SendChar('G');
        SendChar(0x03);
```

```
        SendChar(0x0FA);
        for (i=0 ; i<0x0FA ; i++) {
            if (stav & 0x2){
            if (pole[i]>(byte)200) SendChar(10);
                else SendChar((byte)((byte)200-(byte)pole[i])+(byte)10);
            } else{
            n = ((pole[i] * 158)>>8) * 2;
            if (n>(byte)200) SendChar(10);
                else SendChar((byte)((byte)200-(byte)n)+(byte)10);
            }
        }
        Cekej100ms();
    }else if (timebase<9) SendMsg("No Synchro");
// Vykresleni "hold" cary
    if (stav & 0x08) {
        SendChar(0x1B);
        SendChar('U');
        SendChar((byte)3+(byte)Xhold);
        SendChar((byte)10);
        SendChar((byte)200);
        if (stav & 0x02) HoldValue(pole[Xhold]);
    }
// Nastaveni zobrazovani nezobrazovane obrazovky pokud se nemerilo
    if (stav & 0x08){
        SendChar(0x1B);
        SendChar('V');
        if (screen == 1) SendChar('1');
            else SendChar('2');
    }
    Cekej20ms();
}
void HoldValue(byte X) {
    byte i;
    word Y,zbytek;
    SendChar(0x1B);
    SendChar('Y');
    SendChar(30);
    SendChar(0x1B);
    SendChar('X');
    SendChar(0);
    if (X>210) X=210;
    SendMsg("Voltage: ");
    if (X<100){
        SendChar('-');
        X = (unsigned)100 - (unsigned)X;
    } else X = (unsigned)X - (unsigned)100;
    zbytek = 0;
    switch (mergain){
        case  0 : Y = (X<<2)/10; break;
        case  1 : Y = (X<<3)/10; break;
        case  2 : Y = (X<<1);    break;
        case  3 : Y = (X<<2);    break;
        case  4 : Y = (X<<3);    break;
        case  5 : {
            Y = (X<<1)/100;
            zbytek = (X<<1)%100;
            }break;
        case  6 : {
            Y = (X<<2)/100;
            zbytek = (X<<2)%100;
            }break;
        case  7 : {
            Y = (X<<3)/100;
            zbytek = (X<<3)%100;
            }break;
        case  8 : {
            Y = (X<<1)/10;
            zbytek = (X<<1)%10;
            }break;
        case  9 : {
```

```
            Y = (X<<2)/10;
            zbytek = (X<<2)%10;
            }break;
        case 10 : Y = (X<<3)/10;  break;
        case 11 : Y = (X<<1);     break;
    }
    i=((byte)(Y/100));
    if (i) SendChar(i|0x30);
    i=((byte)((Y%10)%10))|0x30;
    if (Y>9) SendChar(i);
    i=((byte)(Y%10))|0x30;
    SendChar(i);
    if (zbytek) {
        SendChar('.');
        SendChar((zbytek/10)|0x30);
        SendChar((zbytek%10)|0x30);
    }
    if (mergain>4) SendMsg(" V");
        else SendMsg(" mV");
}
void ClrPozadi(void) {
    byte i=0;
// Nastaveni zapisu na nezobrazovanou obrazovku a vymalovani Ramce
    SendChar(0x1B);
    SendChar('W');
    if (screen == 1) {
        SendChar('2');
        screen = 0;
    }
    else {
        screen = 1;
        SendChar('1');
    }
    Ramec();
    SendChar(0x1B);
    SendChar('V');
    if (screen == 1) SendChar('1');
        else SendChar('2');
    Cekej20ms();
}
// *******************************************************
// *              UVODNI SYNCHRONIZACE           *
// *******************************************************
void Synchro(void) {
    byte horni,dolni,i,n,trimh,triml;
    word trim;
    char znak[4];
// Defaultni hodnota trimovaciho registru po resetu
    trimh=0x80;
    triml=0x00;
// Provede se 50 pokusu o nastaveni
    for (n=0; n<50; n++){
// Testovaci smycka v ASM pro jeden impuls 2.5 ms
    asm{
    ; Zakaz preruseni
                CLI
    ; Vynulovani registru
                CLRA
                CLRX
    ; Cekani na nulovy signal
    NULA:   BRSET   2,PTBD,NULA
    ; Cekani na vzestupnou hranu
    JEDNA:  BRCLR   2,PTBD,JEDNA
    ; Pocitaci smycka
    LOOP:   DBNZA   LOOPC
                INCX
    LOOPC:  BRSET   2,PTBD,LOOP
    ; Ulozeni navratovych hodnot
                STA     dolni
                STX     horni
```

```
    ; Povolim preruseni
                SEI
    }
// Prepocet namerene hodnoty
    trim=((word)0x100 - (word)dolni)+((word)horni)*0x100;
    if (trim == 2360) break;
// Test zda se ma trimovaci registr zvednout ci snizit a provedeni
// Spravna hodnota poctu pruchodu smyckou "LOOP" ma byt 2360
    if (trim>2360) {
        if (triml==1) {
            triml=0;
            ICSSC=0;
            trimh++;
            ICSTRM=trimh;
        } else{
            triml=1;
            ICSSC=1;
        }
    } else {
        if (triml==0) {
        triml=1;
        ICSSC=1;
        trimh--;
        ICSTRM=trimh;
    } else{
        triml=0;
        ICSSC=0;
        }
        }
    }
// Vymazani obrazovky
    SendChar(0x1B);
    SendChar('S');
    Cekej10ms();
// Vypis hlaseni
    if ((trim>2350) && (trim<2370)) SendMsg("Time calibration OK - ");
        else SendMsg("Error time calibration - ");
// Vypis namerene hodnty
    for (i=0; i<4; i++) {
        znak[3-i]=(trim%10)|0x30;
        trim=trim/10;
    }
    for (i=0; i<4; i++) SendChar(znak[i]);
}
// ********************************************************
// *                POZADI OBRAZOVKY                      *
// ********************************************************
void Ramec(void) {
    int i;
// Vymazani obrazovky
    SendChar(0x0C);
    Cekej10ms();
// Nastaveni barvy
    SendChar(0x1B);
    SendChar('C');
    SendChar(0x0F);
// Vykresleni krize
    for (i=0; i<51; i++){
        SendChar(0x1B);
        SendChar('U');
        SendChar(i*5+3);
        SendChar(108);
        SendChar(5);
    }
    for (i=0; i<41; i++){
        SendChar(0x1B);
        SendChar('T');
        SendChar(126);
        SendChar(i*5+10);
        SendChar(5);
```

```
    }
// Vykresleni vodorovnych car
    for (i=0; i<9; i++){
        SendChar(0x1B);
        SendChar('T');
        SendChar(3);
        SendChar(i*25+10);
        SendChar(250);
    }
// Vykresleni svislych car
    for (i=0; i<11; i++){
        SendChar(0x1B);
        SendChar('U');
        SendChar(i*25+3);
        SendChar(10);
        SendChar(200);
    }
    Cekej100ms();
// Nastaveni barvy na nejsvetlejsi
    SendChar(0x1B);
    SendChar('C');
    SendChar(0xF0);
// Vykresleni triggeru
    SendChar(0x1B);
    SendChar('U');
    SendChar((byte)3);
    if (stav & 0x02) {
        if (trigger>=100) {
            SendChar((byte)210 - (byte)trigger);
            SendChar((byte)trigger - (byte)100);
        } else{
            SendChar((byte)110);
            SendChar((byte)100 - (byte)trigger);
        }
    } else {
        SendChar((byte)210 - (byte)(((trigger * 158)>>8) * 2));
        SendChar(((trigger * 158)>>8) * 2);
    }
// Nastaveni barvy na nejsvetlejsi
    SendChar(0x1B);
    SendChar('C');
    SendChar(0xFF);
// Popisy
    SendMsg("Oscilloscope          http://radim.xf.cz");
    SendChar(0x0D);
// Nastaveni spodniho radku
    SendChar(0x1B);
    SendChar('Y');
    SendChar(27);
// Vypis casove zakladny
    SendMsg("Timebase: ");
    if (stav & 0x08) i=mertimebase;
        else i=timebase;
    switch (i){
        case  0 : SendMsg(" 50us/div"); break;
        case  1 : SendMsg("100us/div"); break;
        case  2 : SendMsg("200us/div"); break;
        case  3 : SendMsg("500us/div"); break;
        case  4 : SendMsg("  1ms/div"); break;
        case  5 : SendMsg("  2ms/div"); break;
        case  6 : SendMsg("  5ms/div"); break;
        case  7 : SendMsg(" 10ms/div"); break;
        case  8 : SendMsg(" 20ms/div"); break;
        case  9 : SendMsg(" 50ms/div"); break;
        case 10 : SendMsg("100ms/div"); break;
        case 11 : SendMsg("200ms/div"); break;
        case 12 : SendMsg("500ms/div"); break;
        case 13 : SendMsg("  1 s/div"); break;
        case 14 : SendMsg("  2 s/div"); break;
    }
```

```
// Vypis horizontalniho zesileni
    SendMsg("       Gain: ");
    if (stav & 0x08) i=mergain;
       else i=gain;
    if (stav & 0x02){
       switch (i){
          case  0 : SendMsg(" 10mV/div"); break;
          case  1 : SendMsg(" 20mV/div"); break;
          case  2 : SendMsg(" 50mV/div"); break;
          case  3 : SendMsg("100mV/div"); break;
          case  4 : SendMsg("200mV/div"); break;
          case  5 : SendMsg("500mV/div"); break;
          case  6 : SendMsg("  1 V/div"); break;
          case  7 : SendMsg("  2 V/div"); break;
          case  8 : SendMsg("  5 V/div"); break;
          case  9 : SendMsg(" 10 V/div"); break;
          case 10 : SendMsg(" 20 V/div"); break;
          case 11 : SendMsg(" 50 V/div"); break;
       }
    } else{
       switch (i){
          case  0 : SendMsg("  5mV/div"); break;
          case  1 : SendMsg(" 10mV/div"); break;
          case  2 : SendMsg(" 25mV/div"); break;
          case  3 : SendMsg(" 50mV/div"); break;
          case  4 : SendMsg("100mV/div"); break;
          case  5 : SendMsg("250mV/div"); break;
          case  6 : SendMsg("500mV/div"); break;
          case  7 : SendMsg("  1 V/div"); break;
          case  8 : SendMsg("2.5 V/div"); break;
          case  9 : SendMsg("  5 V/div"); break;
          case 10 : SendMsg(" 10 V/div"); break;
          case 11 : SendMsg(" 25 V/div"); break;
       }
    }
// Vypis druheho radku
    if (stav & 0x08) {
       SendMsg("HOLD");
       SendChar('\r');
    } else{
       if (stav & 0x02) SendMsg("Bipolar   ");
          else SendMsg("Unipolar  ");
       if (stav & 0x04) SendMsg("No Synchro   ");
          else if (stav & 0x01) SendMsg("Synchro  Up  ");
             else  SendMsg("Synchro Down ");
       if (stav & 0x10) SendMsg("    One shot ");
          else SendMsg("    Free run ");
       if (stav & 0x20) SendMsg("    AC");
          else SendMsg("    DC");
    }
// Vypis tretiho radku s Help
    SendChar(0x1B);
    SendChar('C');
    SendChar(0xF0);
    SendMsg("                              [H] Help");
    SendChar(0x1B);
    SendChar('C');
    SendChar(0xFF);
}
// *******************************************************
// *                INFORMATIVNI FUNKCE                  *
// *******************************************************
void Help(void){
    int i;
    SendChar(0x1B);
    SendChar(0x09);
    SendChar(0x1B);
    SendChar('S');
    Cekej100ms();
    SendMsg("\r Oscilloscope - Help\r\r");
```

```
    SendMsg("  /\\   - Gain + \r");
    SendMsg("  \\/   - Gain - \r\r");
    SendMsg("  ->   - Timebase + \r");
    SendMsg("  <-   - Timebase - \r\r");
    SendMsg("   S   - NoSynchro/SynchroDown/SynchroUp\r");
    SendMsg("   u   - Trigger + 0.2 div\r");
    SendMsg("   U   - Trigger +   1 div\r");
    SendMsg("   d   - Trigger - 0.2 div\r");
    SendMsg("   D   - Trigger -   1 div\r\r");
    SendMsg(" space - Hold data \r");
    SendMsg("   >   - Shift cursor right \r");
    SendMsg("   <   - Shift cursor left \r\r");
    SendMsg("   P   - Bipolar/Unipolar \r\r");
    SendMsg("   A   - AC/DC \r\r");
    SendMsg("   F   - Freerun/One shot \r");
    SendMsg(" enter - Start one shot \r\r");
    SendMsg("   C   - Center screen \r\r");
    SendMsg("   H   - Help \r\r");
    SendChar(0x1B);
    SendChar('U');
    SendChar(0);
    SendChar(0);
    SendChar(255);
    SendChar(0x1B);
    SendChar('U');
    SendChar(254);
    SendChar(0);
    SendChar(255);
    SendChar(0x1B);
    SendChar('T');
    SendChar(0);
    SendChar(0);
    SendChar(255);
    SendChar(0x1B);
    SendChar('T');
    SendChar(0);
    SendChar(255);
    SendChar(255);
    for (i=0; i<50; i++) Cekej100ms();
    SendChar(0x1B);
    SendChar(0x09);
}
void Znak(byte x, byte y, byte z){
    SendChar(0x1B);
    SendChar('Z');
    SendChar(x);
    SendChar(y);
    SendChar(z);
}
void Uvodtxt(void){
    byte i;
    SendChar(0x1B);
    SendChar('W');
    SendChar('2');
    Cekej20ms();
    SendChar(0x1B);
    SendChar('S');
    Cekej100ms();
    SendChar(0x1B);
    SendChar('V');
    SendChar('2');
    SendChar(0x1B);
    SendChar('C');
    SendChar(0xFF);
    Znak(100,36,'n');
    Znak(126,36,'f');
    SendChar(0x1B);
    SendChar('Y');
    SendChar(10);
    SendMsg("            OSCILLOSCOPE");
```

```
    SendChar(0x1B);
    SendChar('Y');
    SendChar(16);
    SendChar(0x1B);
    SendChar('C');
    SendChar(0x0F);
    SendMsg("\r  Timebase: 50us/div ..  2s/div ");
    SendMsg("\r  Gain:     10mV/div .. 50V/div");
    SendMsg("\r  Input:    1MOhm/30pF");
    SendChar(0x1B);
    SendChar('C');
    SendChar(0xFF);
    SendMsg("\r\r\r\r                        (c) Radim Pechal");
    SendMsg("\r                  radim.pechal@seznam.cz");
    SendMsg("\r                     http://radim.xf.cz");
    SendChar(0x1B);
    SendChar('U');
    SendChar(0);
    SendChar(0);
    SendChar(255);
    SendChar(0x1B);
    SendChar('U');
    SendChar(254);
    SendChar(0);
    SendChar(255);
    SendChar(0x1B);
    SendChar('T');
    SendChar(0);
    SendChar(0);
    SendChar(255);
    SendChar(0x1B);
    SendChar('T');
    SendChar(0);
    SendChar(255);
    SendChar(255);
    for (i=1; i<50; i++) Cekej100ms();
    SendChar(0x1B);
    SendChar('S');
    Cekej100ms();
}
// ********************************************************
// *              FUNKCE PRO CENTROVANI           *
// ********************************************************
void Centrovani(void){
    byte znak;
    int i;
    SendChar(0x1B);
    SendChar('W');
    if (screen == 1) {
        SendChar('2');
        screen = 0;
    }
    else {
        screen = 1;
        SendChar('1');
    }
// Vymazani obrazovky
    SendChar(0x0C);
    Cekej10ms();
    SendChar(0x1B);
    SendChar('C');
    SendChar(0x0F);
    for (i=1; i<4; i++){
        SendChar(0x1B);
        SendChar('U');
        SendChar(i*64);
        SendChar(0);
        SendChar(255);
        SendChar(0x1B);
        SendChar('T');
```

```
        SendChar(0);
        SendChar(i*64);
        SendChar(255);
    }
    SendChar(0x1B);
    SendChar('C');
    SendChar(0xFF);
    SendChar(0x1B);
    SendChar('U');
    SendChar(0);
    SendChar(0);
    SendChar(255);
    SendChar(0x1B);
    SendChar('U');
    SendChar(254);
    SendChar(0);
    SendChar(255);
    SendChar(0x1B);
    SendChar('T');
    SendChar(0);
    SendChar(0);
    SendChar(255);
    SendChar(0x1B);
    SendChar('T');
    SendChar(0);
    SendChar(255);
    SendChar(255);
    SendChar(0x1B);
    SendChar('X');
    SendChar(10);
    SendChar(0x1B);
    SendChar('Y');
    SendChar(15);
    SendMsg("Setting screen posicion\r");
    SendChar(0x1B);
    SendChar('X');
    SendChar(1);
    SendChar(0x1B);
    SendChar('Y');
    SendChar(30);
    SendMsg(" Accept [Enter] Storno [Esc]");
    SendChar(0x1B);
    SendChar('V');
    if (screen == 1) SendChar('1');
        else SendChar('2');
    Cekej20ms();
    while (znak !='\r'){
        SendChar(0x1B);
        SendChar('H');
        SendChar(obrX);
        SendChar(0x1B);
        SendChar('O');
        SendChar(obrY);
        while (!SCIS1_RDRF) {
            feedCOP();
        }
        znak = SCID;
        switch (znak){
            case '2': if (obrY <= 14) obrY++;break;
            case '8': if (obrY >=  1) obrY--;break;
            case '6': if (obrX <= 14) obrX++;break;
            case '4': if (obrX >=  1) obrX--;break;
        }
    }
}
// *******************************************************
// *              NASTAVENI PINU PRO GAIN                *
// *******************************************************
void UpravGain(void) {
    switch (gain){
```

```
        case 0  : {PTBD_PTBD5 = 0; PTBD_PTBD4 = 1; PTBD_PTBD3 = 1; PTBD_PTBD7 = 0;} break;
        case 1  : {PTBD_PTBD5 = 0; PTBD_PTBD4 = 0; PTBD_PTBD3 = 1; PTBD_PTBD7 = 0;} break;
        case 2  : {PTBD_PTBD5 = 1; PTBD_PTBD4 = 1; PTBD_PTBD3 = 0; PTBD_PTBD7 = 0;} break;
        case 3  : {PTBD_PTBD5 = 1; PTBD_PTBD4 = 0; PTBD_PTBD3 = 0; PTBD_PTBD7 = 0;} break;
        case 4  : {PTBD_PTBD5 = 0; PTBD_PTBD4 = 1; PTBD_PTBD3 = 0; PTBD_PTBD7 = 0;} break;
        case 5  : {PTBD_PTBD5 = 0; PTBD_PTBD4 = 0; PTBD_PTBD3 = 0; PTBD_PTBD7 = 0;} break;
        case 6  : {PTBD_PTBD5 = 0; PTBD_PTBD4 = 1; PTBD_PTBD3 = 1; PTBD_PTBD7 = 1;} break;
        case 7  : {PTBD_PTBD5 = 0; PTBD_PTBD4 = 0; PTBD_PTBD3 = 1; PTBD_PTBD7 = 1;} break;
        case 8  : {PTBD_PTBD5 = 1; PTBD_PTBD4 = 1; PTBD_PTBD3 = 0; PTBD_PTBD7 = 1;} break;
        case 9  : {PTBD_PTBD5 = 1; PTBD_PTBD4 = 0; PTBD_PTBD3 = 0; PTBD_PTBD7 = 1;} break;
        case 10 : {PTBD_PTBD5 = 0; PTBD_PTBD4 = 1; PTBD_PTBD3 = 0; PTBD_PTBD7 = 1;} break;
        case 11 : {PTBD_PTBD5 = 0; PTBD_PTBD4 = 0; PTBD_PTBD3 = 0; PTBD_PTBD7 = 1;} break;
    }
    Cekej20ms();
}
// *******************************************************
// *                    MAIN                             *
// *******************************************************
void main(void) {
    byte znak;
    EnableInterrupts;                           //Povoleni preruseni
// Inicializace
// Nastaveni vstupu/vystupu
    PTAPE = 0xFF;
    PTBPE = 0xFF;
    PTBDD_PTBDD7 = 1;                           //Pin pro zapnuti/vypnuti delice
    PTBD_PTBD7 = 1;
    PTBDD_PTBDD6 = 1;                           //Pin pro prepinani unipolar/bipolar
    PTBD_PTBD6 = 0;
    PTBDD_PTBDD5 = 1;                           //Pin pro nastaveni zesileni
    PTBD_PTBD5 = 0;
    PTBDD_PTBDD4 = 1;                           //Pin pro nastaveni zesileni
    PTBD_PTBD4 = 1;
    PTBDD_PTBDD3 = 1;                           //Pin pro nastaveni zesileni
    PTBD_PTBD3 = 1;
    PTADD_PTADD2 = 1;                           //Pin pro prepinani AC/DC
    PTAD_PTAD2 = 1;
    PTBDD_PTBDD2 = 0;                           //Pin se signalem na pocatecni synchro
// Nastaveni ICS na FEI mode
    ics_intclk(0x04,0x00);
// Nastaveni SCI pro 9600 baud
    SCIBD = fei1_spd;
    SCIC1 = 0x10;
    SCIC2 = 0x0C;
    SCIC3 = 0x40;
// Nastaveni A/D converter
    APCTL1 = 0x03;                              // Port PTA0 a PTA1 jako A/D
    ADCCFG = 0x00;                              // Rychlost a 8bit. prevod
    ADCSC1 = 0x20;                              // Continuous a kanal 0
    ADCSC2 = 0x00;                              // Vypnout komparaci
    ADCSC1 = 0x20;                              // Continuous a kanal 0
// Sesynchronizovani na 17MHz
    Synchro();
    Cekej100ms();
// Vypis uvodni obrazovky
    Uvodtxt();
// *******************************************************
// *              HLAVNI SMYCKA                          *
// *******************************************************
    for(;;) {
        if ((!(stav & 0x10))||(stav & 0x08)) Graf();
            else if (!(znak == '\r')) ClrPozadi();
        feedCOP();
// Obsluha Hold
        if (stav & 0x18) {
// Ocekavani klavesy pri Hold nebo pri One shot
            while (!SCIS1_RDRF) {
                feedCOP();
            }
        } else {
```

```
// Test zda neprisla klavesa
            if (!SCIS1_RDRF) {
                continue;
            }
        }
// Vyhodnoceni prijate klavesy
        znak = SCID;
        switch (znak) {
// Help
            case 'h' : case 'H' : Help(); break;
// Centrovani obrazovky
            case 'c' : case 'C' : Centrovani(); break;
// Prepinani AC/DC
            case 'a' : case 'A' : if (stav & 0x20) {
                PTAD_PTAD2 = 1;
                stav &= (byte)0xDF;
            } else{
                PTAD_PTAD2 = 0;
                stav |= (byte)0x20;
            }break;
// One shot / Free run
            case 'f' : case 'F' : if (stav & 0x10){
                stav &= 0xEF;
            } else {
                ClrPozadi();
                stav |= 0x10;
            }break;
            case '\r' : if ((stav & 0x10)&&(!((stav & 0x08)))) Graf(); break;
// Polarita signalu (Unipolar / Bipolar)
            case 'p' : case 'P' : if (stav & 0x2) {
                PTBD_PTBD6 = 1;
                stav &= (byte)0xFD;
                if (trigger>160) trigger=160;
            } else{
                PTBD_PTBD6 = 0;
                stav |= (byte)0x2;
            }break;
// Obsluha hold
            case ' ' : if (stav & 0x08) stav &= 0xF7;
                else stav |= 0x08; break;
            case ',' : if (Xhold > 0) Xhold -= 5; break;
            case '.' : if (Xhold < 250) Xhold += 5; break;
// Nastaveni Synchra
            case 'S' : case 's' : {
                if (stav & 0x04) stav &= 0xFA;
                else if (stav & 0x01) stav |= 0x04;
                    else stav |= 0x01;
            }break;
// Nastavovani zesileni
            case '8' : if (gain < 11) {
                gain++;
                UpravGain();
            } break;
            case '2' : if (gain > 0) {
                gain--;
                UpravGain();
            } break;
// Nastavovani casove zakladny
            case '6' : if (timebase < 14) timebase++; break;
            case '4' : if (timebase > 0) timebase--; break;
// Nastavovani Triggeru
            case 'U' : if (stav & 0x02) {
                if (trigger <= 175) trigger+=25;
            }
            else if (trigger <= 135) trigger+=25; break;
            case 'u' : if (stav & 0x02) {
                if (trigger <= 195) trigger+=5;
            }
            else if (trigger <= 155) trigger+=5; break;
            case 'D' : if (trigger >= 25) trigger-=25; break;
```

```
                case 'd' : if (trigger >= 5) trigger-=5; break;
// Spatne zadani
                default: SendMsg("\rInvalid selection");
            }
        }
}
// A to je vse :)
```