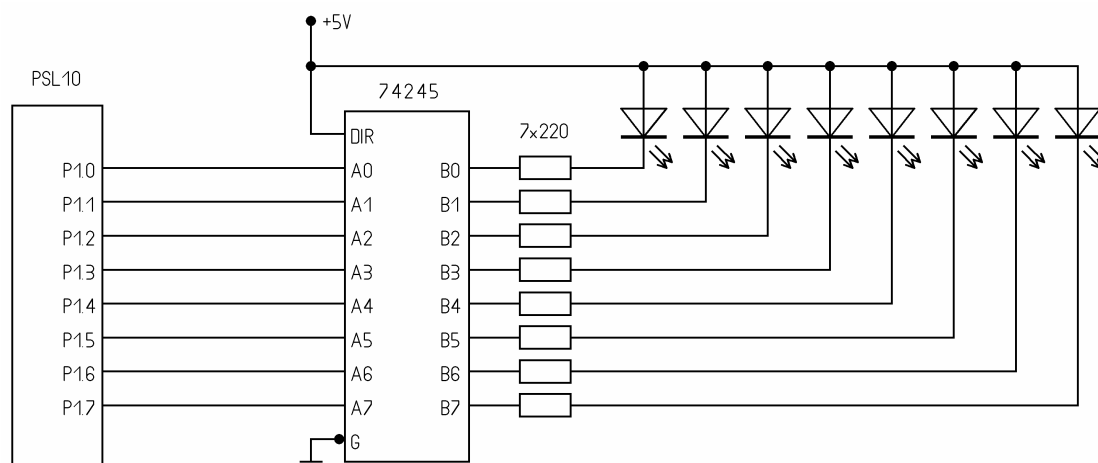


1.	MODUL LED	2
2.	MODUL JEDNOMÍSTNÉHO STATICKÉHO DISPLEJE	3
3.	MODUL DVOJMÍSTNÉHO STATICKÉHO DISPLEJE.....	5
4.	MODUL DIP.....	7
5.	MODUL REPRO.....	8
6.	MODUL PIEZO	9
7.	MODUL KLÁVESNICE	10
8.	MODUL TLAČÍTEK	12
9.	MODUL TŘÍMÍSTNÝ DYNAMICKÝ DISPLEJ	14
10.	MODUL OSMIMÍSTNÉHO DYNAMICKÉHO DISPLEJE	15
11.	MODUL MATICOVÉHO DISPLEJE.....	16
12.	MODUL PRO OVLÁDÁNÍ KROKOVÉHO MOTORU (4 CÍVKY).....	18
13.	MODUL PRO OVLÁDÁNÍ KROKOVÉHO MOTORU (2 CÍVKY).....	20
14.	MODUL ČTYŘMÍSTNÉHO DISPLEJE S REGISTRY 74595	22
15.	MODUL D/A PŘEVODNÍKU S POSTUPNOU APROXIMACÍ.....	24
16.	MODUL LCD DISPLEJ.....	26
17.	MODUL PRO MĚŘENÍ FREKVENCE A PERIODY	28
18.	MODUL PŘIJÍMAČE DÁLKOVÉHO OVLÁDÁNÍ SONY.....	30
19.	MODUL REGULACE JASU ŽÁROVKY – TRIAKEM.....	32
20.	MODUL REGULACE JASU ŽÁROVKY – TRANZISTOREM.....	35
21.	MODUL OBVODU REÁLNÉHO ČASU (IIC).....	37
22.	MODUL OBVODU REÁLNÉHO ČASU DS 1302	40
23.	MODUL PAMĚTI EEPROM (IIC)	44
24.	MODUL PRO SERIOVOU KOMUNIKACI.....	45
25.	MODUL PRO SÉRIOVOU KOMUNIKACI – VYSÍLAČ	47
26.	MODUL PRO SÉRIOVOU KOMUNIKACI - PŘIJÍMAČ.....	48

1. Modul LED

Modul LED se skládá z osmi LED diod, které jsou připojené přes budič sběrnice k portu procesoru. Obvod 74245 je obousměrný budič sběrnice a slouží k posílení výstupního proudu portu procesoru. Vstup G je „output enable“, který umožňuje uvést výstup do stavu vysoké impedance. Vstup DIR určuje směr toku dat (která z bran A,B je vstupní a která výstupní). Diody jsou zapojené se společnou anodou, to znamená, že svítí na logickou nulu. Spínání logickou nulou se u procesoru provádí ze dvou důvodů:

1. Po resetu jsou na portech log. 1
2. Výstupní proud je v log.0 vyšší než v log.1

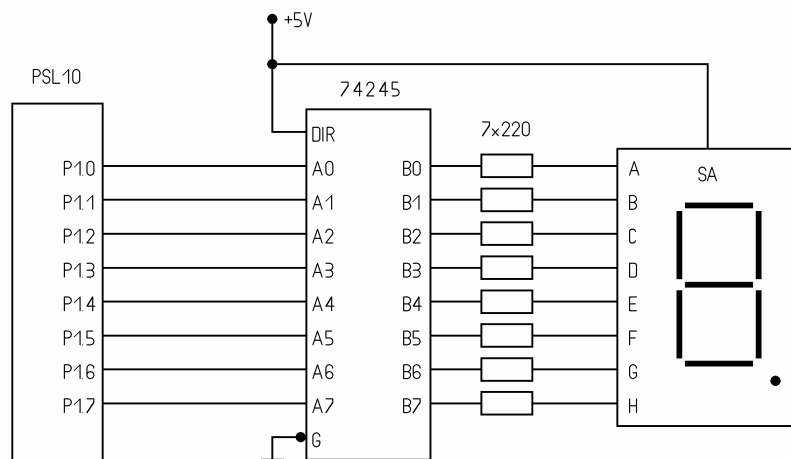


Programy pro modul LED:

1. Program *LED1* provádí blikání diod s frekvencí 1Hz. Doba trvání podprogramu se vypočítá $T=2 \cdot R1 \cdot R2 \cdot R3$.
2. Program *LED2* rozsvěcuje po jedné sekundě různé kombinace diod. V programu je možno přidávat a měnit řádky „MOV P3,.....“.
3. Program *LED3* rozsvěcuje po jedné sekundě různé kombinace diod, které jsou uloženy v tabulce. Program bude posouvat svítící bod vlevo a vpravo.

2. Modul jednomístného statického displeje

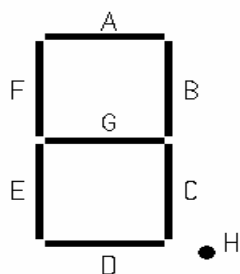
Jedná se o sedmisegmentový displej se společnou anodou, který je přes budič připojen k portu procesoru. Svítí segmenty na nichž je logická nula.



Zapojení jednotlivých bitů portu je patrné z obrázku:

P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
H	G	F	E	D	C	B	A

Označení jednotlivých segmentů:

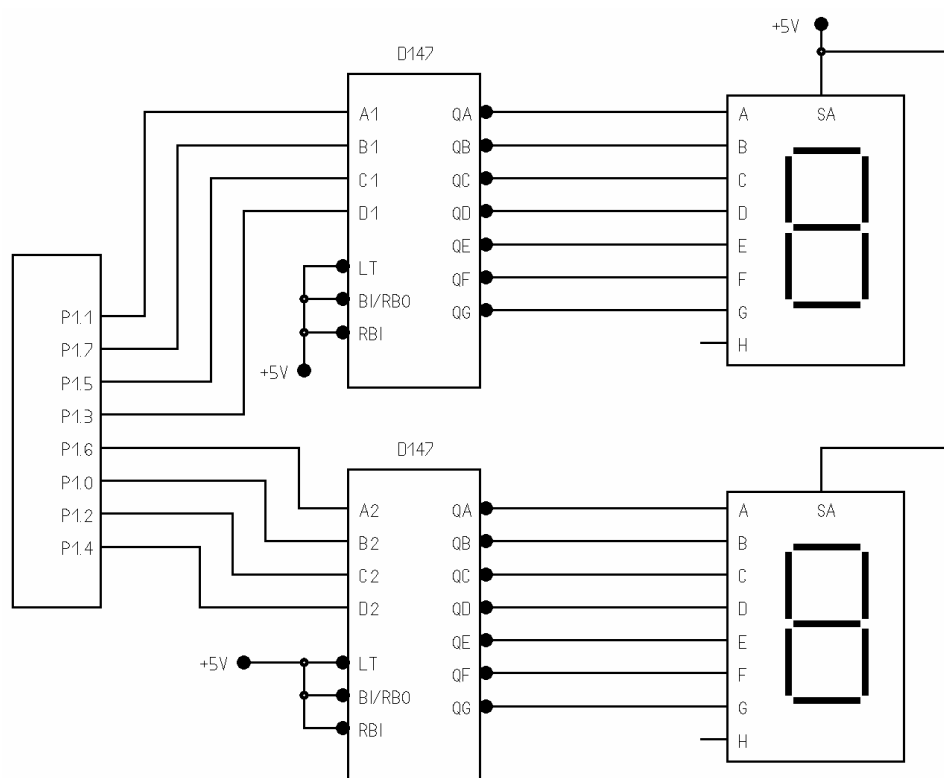


Programy pro modul statický displej:

1. Program *SDISP1* zobrazuje na displeji po jedné sekundě vybrané znaky.
2. Program *SDISP2* již obsahuje podprogram *SDISP*, který bude používán ve spojení s dalšími programy. Podprogram převede binární hodnotu proměnné *CISLO* do kódu pro sedmisegmentovku a pošle na displej.

3. Modul dvojmístného statického displeje

Stejně jako modul jednomístného statického displeje je i tento modul připojen k jednomu portu. Jak je tedy možné, že u obou je obsazeno všech osm bitů a tento modul má dvakrát tolik míst? U jednomístného modulu je každý segment připojen na jeden bit portu. To znamená, že můžeme na sedmsegmentovce zobrazit libovolný znak ze všech 256 možných kombinací. U dvojmístného displeje jsou sedmsegmentovky připojené přes dekodér. Do každého dekodéru jsou připojené čtyři bity portu (viz. Tabulka). Do dekodéru můžeme přivést číslo 0 – 15 a na výstupu dekodéru může být pouze 16 kombinací (16 různých znaků). Většinou to bývají znaky 0 – 9, a dalších šest znaků, např. A, B, C, D, E, F, -, prázdný znak (zhasnutý displej) apod. Navíc nejsou zapojené desetinné tečky.



Připojení vstupů dekodérů na port není ve správném pořadí a číslo, které se má zobrazit bude nutné před vysláním na port upravit pomocí bitových přesunů. Přesné zapojení je v tabulce:

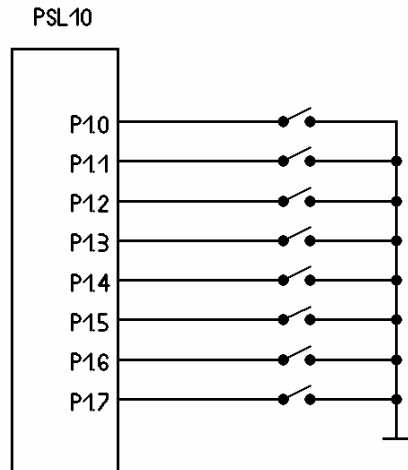
P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
B1	A2	C1	D2	D1	C2	A1	B2

Programy pro modul dvojmístný statický displej:

1. Program *SDISP2A* postupně zobrazí na jedné a pak na druhé sedmisegmentovce všechny znaky odpovídající číslům 0 – 15.
2. Program *SDISP2B* po sekundě inkrementuje čísla na displeji od 0 do 59.

4. Modul DIP

Tento modul obsahuje osm dipů (přepínačů), které spojují jednotlivé bity portu s úrovní logická nula.



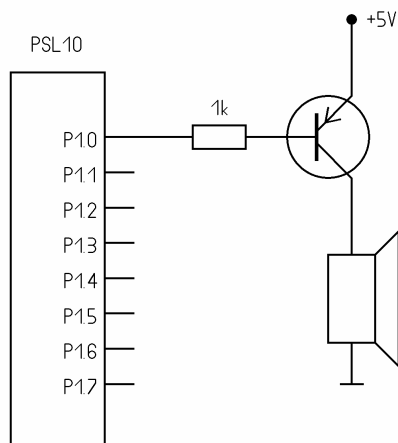
V poloze *ON* je dip sepnutý a bit portu je spojen s logickou nulou.

Program pro modul DIP

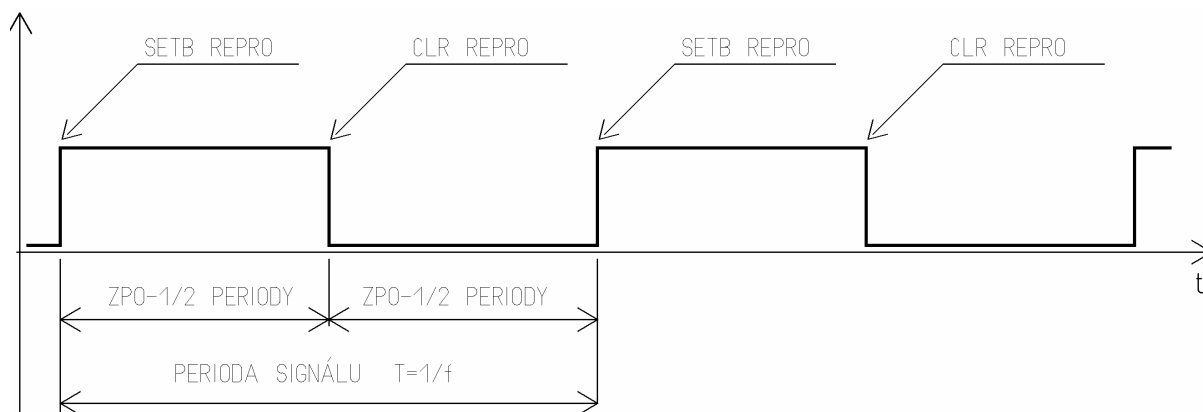
1. Program *DIP* testuje dip na bitu P1.0 a je-li $P1.0 = \text{log } 0$, zhasne všechny diody na modulu LED připojeném k P3 a je-li $P1.0 = \text{log } 1$, rozsvítí všechny diody na modulu LED připojeném k P3.

5. Modul REPRO

Modul obsahuje reproduktor, který je připojen přes tranzistor k bitu P1.0.



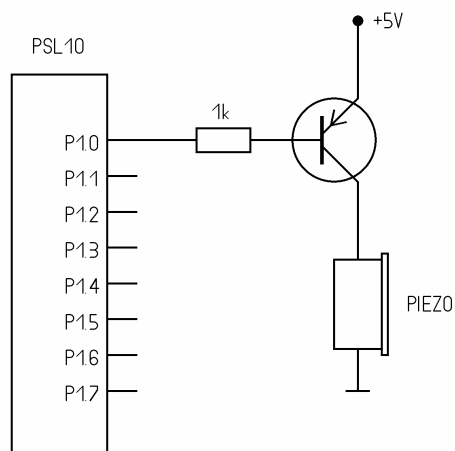
Aby reproduktor vydával zvuk, je nutné rozpohybovat jeho membránu a frekvence kmitů membrány pak odpovídá frekvenci zvuku. Membrána se rozkmitá, jestliže připojíme reproduktor na obdélníkový signál. Ten budeme vytvářet na portu P1.0 tím, že ho uvedeme do stavu log. 0, počkáme polovinu periody (ZPO), potom do stavu log. 1 a znovu počkáme polovinu periody.



Programy pro modul REPRO

1. Program *REPRO1* bude nepřetržitě generovat tón zvolené frekvence.
2. Program *REPRO2* bude zvolenou dobu generovat tón zvolené frekvence. Tento program vygeneruje tón 1kHz po dobu 1 sekundy, dále tón 200 Hz po dobu jedné sekundy a zastaví se na řádku JMP \$. Podprogramy tónů lze libovolně přidávat a měnit jejich frekvenci a délku. Frekvence se mění délkou zpoždění a délka tónu registry R2, R3.

6. Modul PIEZO



Připojení piezosirény k mikroprocesoru umožňuje velmi snadné generování zvuku. Stačí připojit piezosirénu k stejnosměrnému napětí. Nevýhodou ovšem je, že frekvence vydávaného tónu je stále stejná.

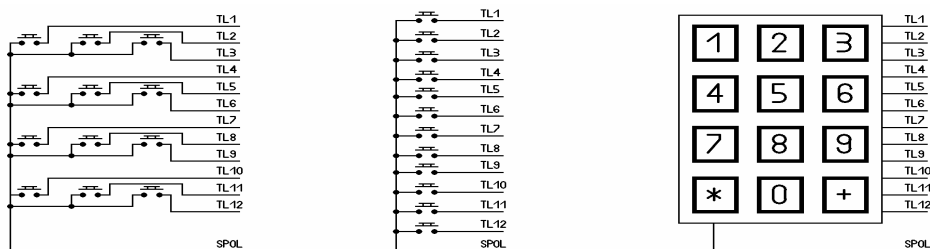
Programy pro modul PIEZO

1. Program *PIEZO1* třikrát krátce a dvakrát dlouze pípne
2. Program *PIEZO2* pro generování libovolného počtu pípnutí libovolné délky. Délka pípnutí se volí registrem R6 (x 100 ms), počet registrem R7.

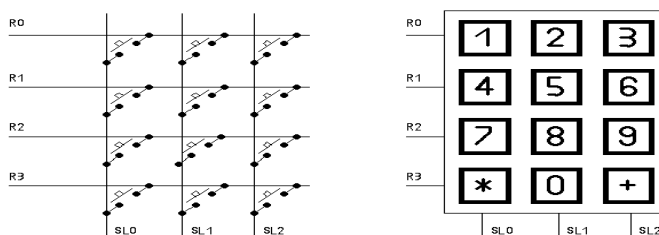
7. Modul KLÁVESNICE

Přímé připojení klávesnice k procesoru je možno provést dvěma způsoby:

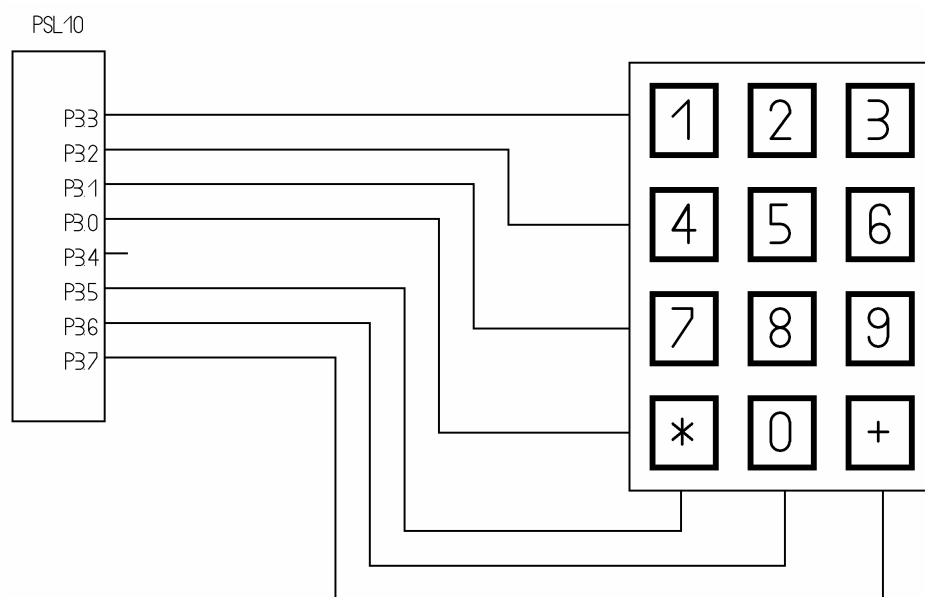
1. Každé tlačítko připojíme k jednomu bitu portu. Toto řešení je jednoduché, program pro čtení této klávesnice bude jednoduchý, ale např. 16-ti tlačítková klávesnice nám zabere celé dva porty. Zapojení bude podobné jako u modulu *DIP* a také program bude podobný.



2. Klávesnici zapojíme do matice, takže každé tlačítko bude spínat sloupcový a řádkový vodič. 16-ti tlačítkovou klávesnici připojíme k jednomu portu. Čtení klávesnice je pak složitější. Nejprve pošleme log 0 na první sloupec a testujeme čtyři řádkové vodiče. Jestliže je stisknuta některá klávesa z prvního sloupce, objeví se na patřičném řádkovém vodiči log 0. Jinak jsou na všech řádkových vodičích log 1. Toto opakujeme pro druhý a třetí sloupec.



Modul klávesnice je zapojen podle obrázku:

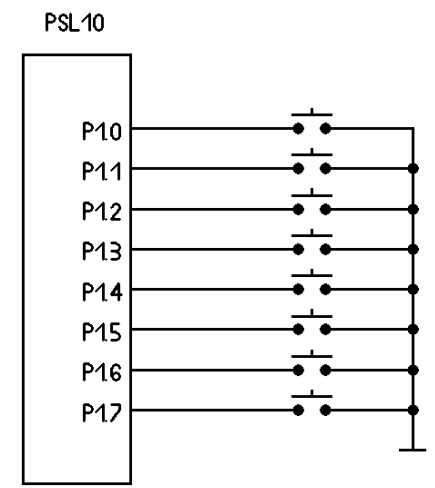


Programy pro čtení klávesnice

1. Program *KLAV1* čte klávesnici a číslo stisknuté klávesy zobrazí na modulu jednomístného statického displeje. Program obsahuje podprogram pro čtení klávesnice, který bude používán u dalších modulů. Tento podprogram při držení klávesy v proměnné KL vrací číslo klávesy.
2. Program *KLAV2* čte klávesnici a při stisku libovolné klávesy inkrementuje číslo na modulu jednomístného statického displeje. Program obsahuje upravený podprogram z předchozího příkladu, který vrací v proměnné KL číslo stisknuté klávesy pouze při prvním čtení klávesnice po stisku. Jinak řečeno – po stisku klávesy při prvním čtení klávesnice je v KL číslo stisknuté klávesy, při dalších čteních klávesnice se chová jako by nebylo stisknuto nic, teprve po uvolnění klávesy a dalším stisku je v KL číslo stisknuté klávesy.

8. Modul tlačítek

V tomto modulu je každé tlačítko připojeno k jednomu bitu portu, jak bylo popsáno výše. Toto zapojení použijeme v případě, že máme dostatek volných bitů na portech.



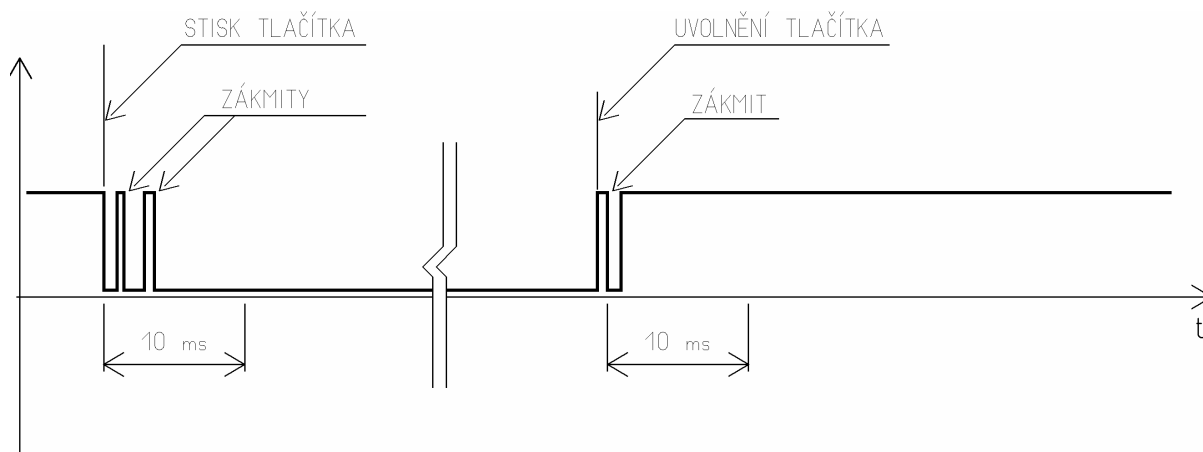
Testování tlačítek lze provádět dvěma způsoby:

1. Akce se bude provádět neustále, pokud bude tlačítko stisknuté (test log0).
2. Akce se provede při stisku tlačítka (test sestupné hrany).

V druhém případě je nutné dávat pozor na „zákmity“. Zákmit je jev, který vzniká při stisku nebo uvolnění tlačítka. V tomto okamžiku dochází k odskočení kontaktu a tím k několika sestupným a nástupným hranám (vše trvá několik milisek.). Potom při jednom stisku tlačítka může načíst několik impulsů, stejně tak při uvolnění tlačítka.

Odstranit tento jev je možné následující úpravou programu pro test tlačítka:

1. Po zjištění sestupné hrany čekáme 10ms
2. Testujeme tlačítko znovu a pokud není stisknuto, jedná se zákmit



Pokud dojde ke stisku tlačítka, tak během čekací doby zákmity odezní. Vzhledem k tomu, že stisk tlačítka kratší, než 10 ms není možný, tak po této době bude tlačítko stisknuté a zaznamenáme platnou sestupnou hranu. Pokud dojde při uvolnění k zákmitu, po 10 ms již tlačítko stisknuté nebude a proto sestupnou hranu ignorujeme.

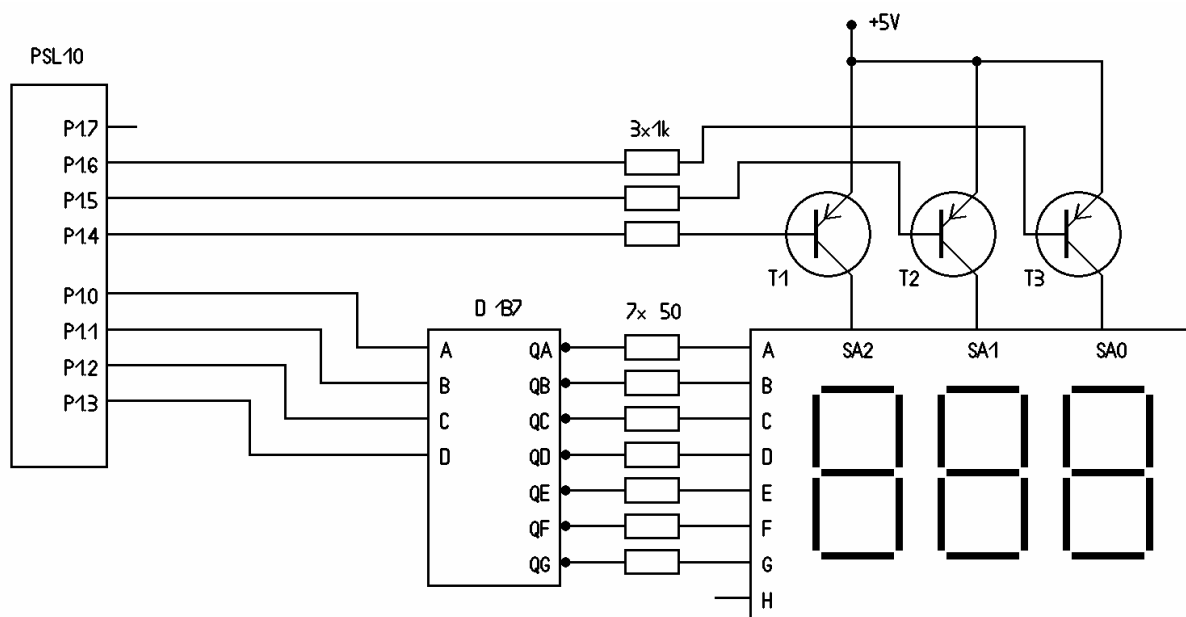
Programy pro modul tlačítek:

1. Program *TLA1* testuje tlačítka na portu P3 a je-li stisknuté tlačítko P3.0, inkrementuje číslo na jednomístném stat. displeji po dobu stisku každých 100 ms.
2. Program *TLA1* testuje tlačítka na portu P3 a při každém stisku tlačítka P3.0 inkrementuje číslo na jednomístném stat. displeji (bez ošetření zákmitů).
3. Program *TLA1* testuje tlačítka na portu P3 a při každém stisku tlačítka P3.0 inkrementuje číslo na jednomístném stat. displeji (s ošetřením zákmitů).

9. Modul třímístný dynamický displej

Dynamický displej využívá nedokonalosti lidského oka, které není schopno zaznamenat změny častěji než 50-krát za sekundu. Kvůli snížení počtu bitů potřebných k připojení displeje k procesoru, se zapojí displej dle obr. a zobrazování znaků na jednotlivých sedmissegmentovkách se provádí postupně. Nejdříve na datové vodiče přivedeme číslo pro sedmissegmentovku č1. Potom sepne tranzistor T1 přivedením log 0 na bit P1.4. Tím se rozsvítí znak na sedmissegmentovce č1. Dále čekáme 2 ms, zhasneme tím, že rozepneme tranzistory T1. Přivedeme na datové vodiče číslo pro sedmissegmentovku č2 a sepne tranzistor T2. Opět čekáme 2 ms atd. Vše se opakuje pro třetí sedmissegmentovku a potom znovu pro první. Vzhledem k tomu, že se toto provádí velmi rychle a proud jednotlivých segmentů je trojnásobný (svítí třetinu doby), blikání displeje není vidět.

Výhodou dynamického displeje je malé množství vodičů potřebných k připojení k procesoru. Pro srovnání – statický displej s dekodéry osmimístný potřebuje $8 \times 4 = 32$ vodičů, kdežto dynamický displej vystačí se sedmi vodiči. Nevýhodou ovšem je složitý program pro obsluhu a potřeba neustálého zapisování dat na displej.

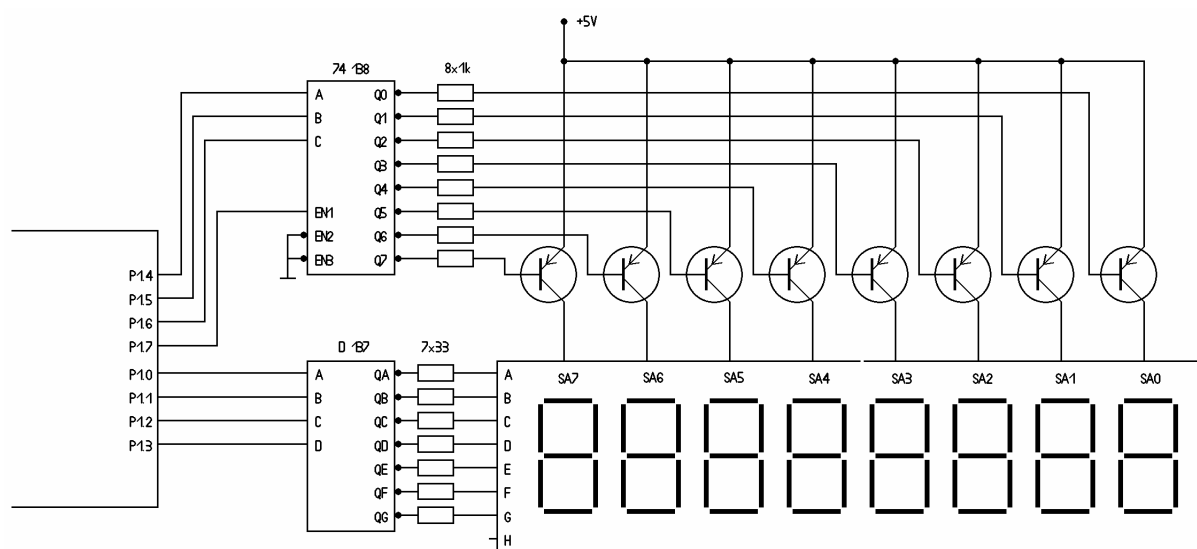


Programy pro modul třímístného dynamického displeje:

1. Program *DDISP3A* zobrazí na displeji čísla zapsané v proměnných *JED*, *DES*, *STA*.
2. Program *DDISP3B* čte klávesnici (port P3) a čísla stisknutých kláves zobrazuje na SEG0 (sedmissegmentovka vpravo) čísla na displeji posouvá vlevo.

10. Modul osmimístného dynamického displeje

Tento displej pracuje na podobném principu jako třímístný dynamický displej. Rozdíl je pouze v počtu míst a v adresování míst. U třímístného displeje je každý tranzistor připojen na jeden bit portu, u osmimístného jsou tranzistory připojeny na výstupy dekodéru 1z8 a k adresování je tedy potřeba méně bitů portu.

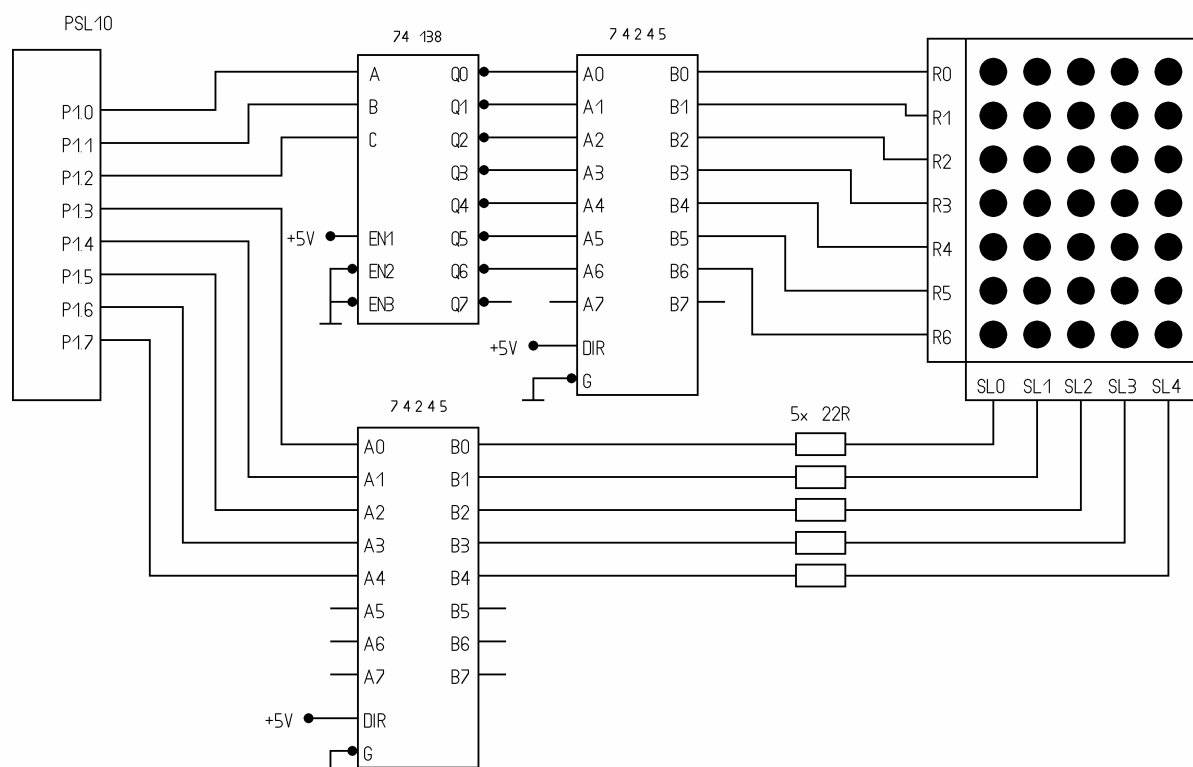


Programy pro osmimístný dynamický displej:

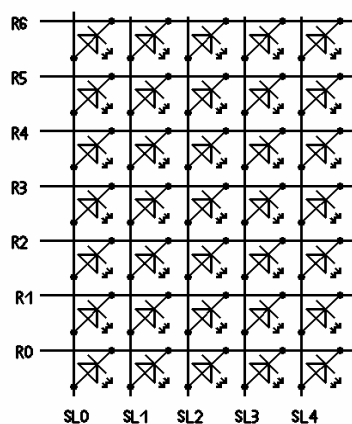
1. Program *DISP8A* zobrazí čísla zapsaná v proměnných *JED*, *DES*,.....*DTI*.
2. V programu *DISP8B* je zadán čas (sekundy, minuty, hodiny) a od tohoto času poběží na displeji hodiny (program využívá časové přerušení).

11. Modul maticového displeje

Jedná se o displej složený z 35 LED diod uspořádaných do obdélníku 5 x 7. Tento displej určitě nepůjde připojit k procesoru přímo, protože procesor nemá 35 bitů na portech. Takže musíme využít možnost ovládat maticový displej dynamicky. Zvolíme variantu, ve které budou na sloupce posílána data (5 LED diod) a jednotlivé řádky (7) budou postupně spínané dekodérem 1 z 8, přičemž jeden vývod dekodéru zůstane volný.



Zapojení diod v maticovém displeji je následující:

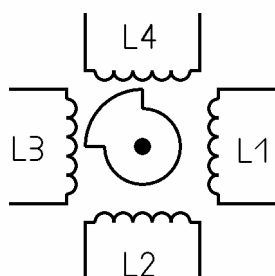


Programy pro maticový displej:

1. Program *MATIC1* zobrazí na displeji písmeno „A“.
2. Program *MATIC2* zobrazí na displeji bod a pomocí tlačítek 2, 4, 5, 6, 8 je možno bodem pohybovat (tento program využívá časové přerušení).

12. Modul pro ovládání krokového motoru (4 cívky)

Krokový motor se výrazně liší od ostatních motorů. Hlavní rozdíl je v tom, že u krokového motoru je možno přesně nastavit polohu rotoru, v této poloze rotor zajistit a navíc při přetížení a zastavení rotoru nedojde ke zničení motoru. Přesná funkce je patrná z obrázku:



Rotor krokového motoru je v podstatě magnet a stator je složen ze čtyř cívek (ne u všech kr. motorů). Toto je velmi zjednodušené, ale pro vysvětlení funkce to postačí. Jestliže cívkou protéká proud, rotor se k této cívce pootočí. Budeme-li spínat postupně cívky L1, L2, L3, L4, bude se rotor otáčet ve směru ručiček hodinových. Budeme-li spínat postupně cívky L4, L3, L2, L1, bude se rotor otáčet opačným směrem. Rychlost otáčení je dána rychlostí přepínání jednotlivých cívek. Nutno dodat, že těchto čtveřic cívek je na statoru KM několik. Čím více, tím bude menší krok otáčení. Zůstane-li cívka trvale sepnuta, rotor bude v této poloze fixován určitou silou.

Otáčení rotoru je možno provádět třemi způsoby:

1. S jednou sepnutou fází – toto je způsob, který byl popsán výše
2. Se dvěma sepnutými fázemi - spínají se zároveň dvě cívky L1+L2, L2+L3, L3+L4, L4+L1, L1+L2, atd. Tento způsob má oproti předchozímu dvojnásobný moment, ale také dvojnásobnou spotřebu.
3. S polovičním krokem – toto je kombinace obou předchozích způsobů. Postup spínání – L1, L1+L2, L2, L2+L3, L3, L3+L4, L4, L4+L1, L1, atd. Tento způsob má poloviční krok otáčení.

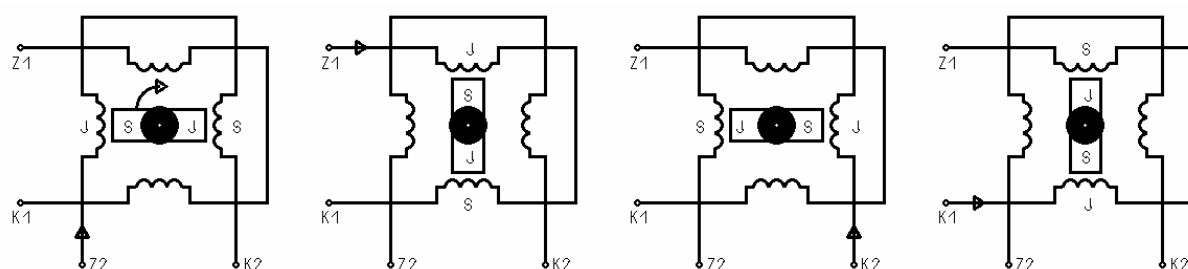
Modul je zapojen dle obrázku a spínání jednotlivých cívek se provádí přivedením log. nuly na bity P3.1, P3.3, P3.5, P3.7.



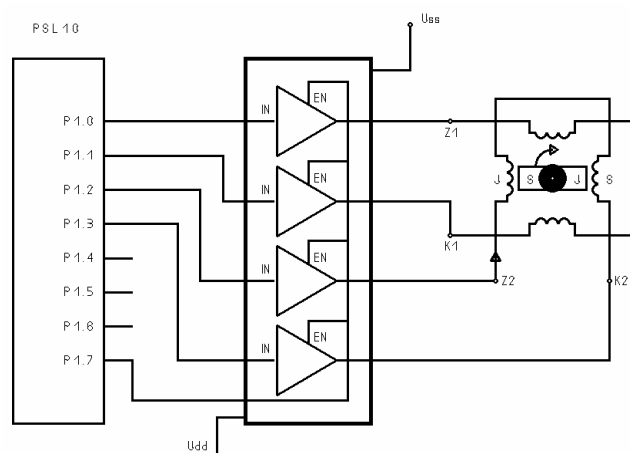
- 19

13. Modul pro ovládání krokového motoru (2 cívky)

Pro tento motor platí vše co bylo o krokovém motoru napsáno v předchozí kapitole. Narazíme zde ale na jeden problém – jak je možno pomocí dvou cívek určit směr otáčení? K tomu jsou potřeba minimálně tři cívky. U motoru se dvěma cívkami záleží na směru proudu. To znamená, že nejdříve sepne cívku L1, přičemž proud protéká od Z1 do K1, potom sepne cívku L2, proud protéká od Z2 do K2, dále cívku L1, proud protéká od K1 do Z1 a nakonec cívku L2, proud protéká od K2 do Z2. V podstatě to znamená, že čtyři polohy získáme připojením každé cívky na obě možnosti polarity napětí.



Modul krokového motoru je zapojen dle obr., všechny čtyři vývody cívek jsou připojeny na výkonové budiče. Každý budič má TTL vstup a výkonový výstup. Je-li na vstupu log. 0, je na výstupu U_{dd} , je-li na vstupu log. 1, je na výstupu napětí U_{ss} .



Cívkou tedy bude protékat proud tehdy, bude-li na jednom vodiči U_{dd} a na druhém vodiči U_{ss} . Na portu procesoru tento stav nastane tehdy, budou-li na P1.0 a P1.1 rozdílné logické úrovně

(podobně na P1.2 a P1.3). Pro řízení s jednou sepnutou fází budeme na port posílat následující data:

1. 1XXX 0001B
2. 1XXX 0100B
3. 1XXX 0010B
4. 1XXX 1000B

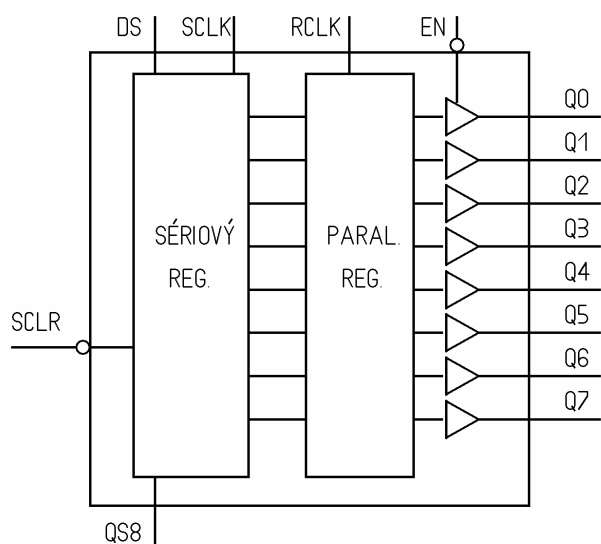
Pro druhý směr samozřejmě v opačném pořadí. Na bit P1.7 je připojen ENABLE všech budičů. Bude-li v log. 0, budou všechny výstupy na úrovni U_{dd} .

Program pro modul krokového motoru (2 cívky):

1. Program *KROKMOT2A* provádí otáčení rotoru o jednu otáčku střídavě doleva a doprava.

14. Modul čtyřmístného displeje s registry 74595

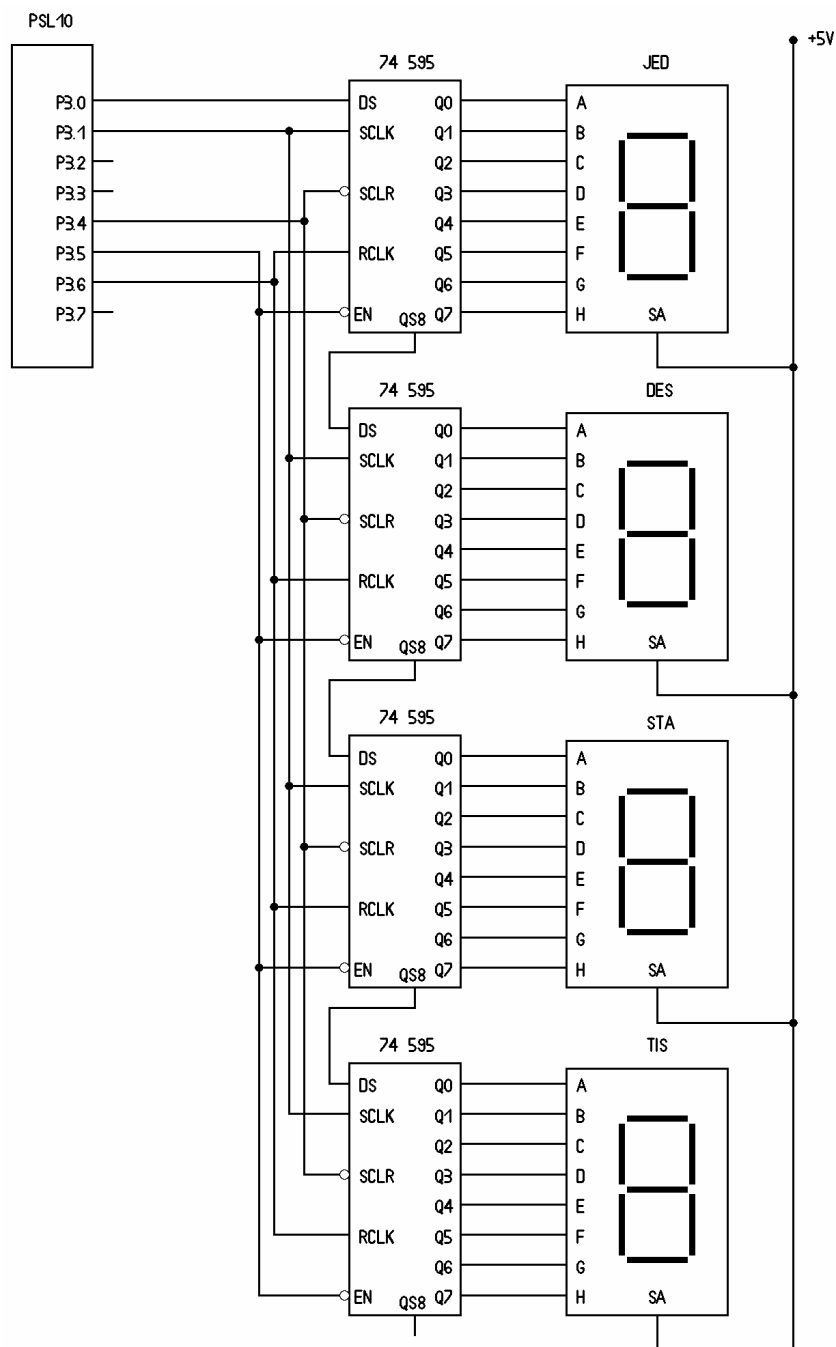
Registr 74595 se skládá z osmibitového sériového registru, osmibitového paralelního registru a třístavového výstupního budiče. Sériový registr má hodinový vstup SCLK, datový vstup DS, resetovací vstup SCLR a datový výstup QS8. Hodinový vstup nástupnou hranou posouvá data v sériovém registru ze vstupu DS. Poslední bit dat je na výstupu QS8 a tento výstup slouží k propojení s dalším obvodem. Vstup SCLR provede resetování všech bitů sériového registru. Všechny osm bitů sériového registru je připojeno na vstupy paralelního registru. Přepis do paralelního registru se provádí nástupnou hranou na hodinovém vstupu RCLK. Tím dostaneme data na vstupy výstupních budičů. Tyto budiče jsou ovládané vstupem EN. Bude-li EN ve stavu log1, budou všechny výstupy ve stavu vysoké impedance. V opačném případě budou na výstupech data z paralelního registru.



Na následujícím obrázku je zapojení celého modulu. Displej je zapojený bez dekodérů, to znamená, že je možno zobrazit libovolný znak, ale číslo, které chceme zobrazit musíme převést v programu podle následující tabulky:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
H	D	C	G	B	A	F	E

Takže budeme-li mít čtyři proměnné (JED, DES, STA, TIS), musíme každou proměnnou převést do kódu pro sedmissegmentovku podle tabulky a vyslat jako sériová data. Nejdříve se vysílá proměnná TIS. Po vyslání všech čtyř proměnných provedeme nástupnou hranu na bitu RCLK a tím se přepíší data ze sériových registrů do paralelních. Bity SCLR- log1, EN- log0.



Programy pro modul čtyřmístný displej s registry 74595:

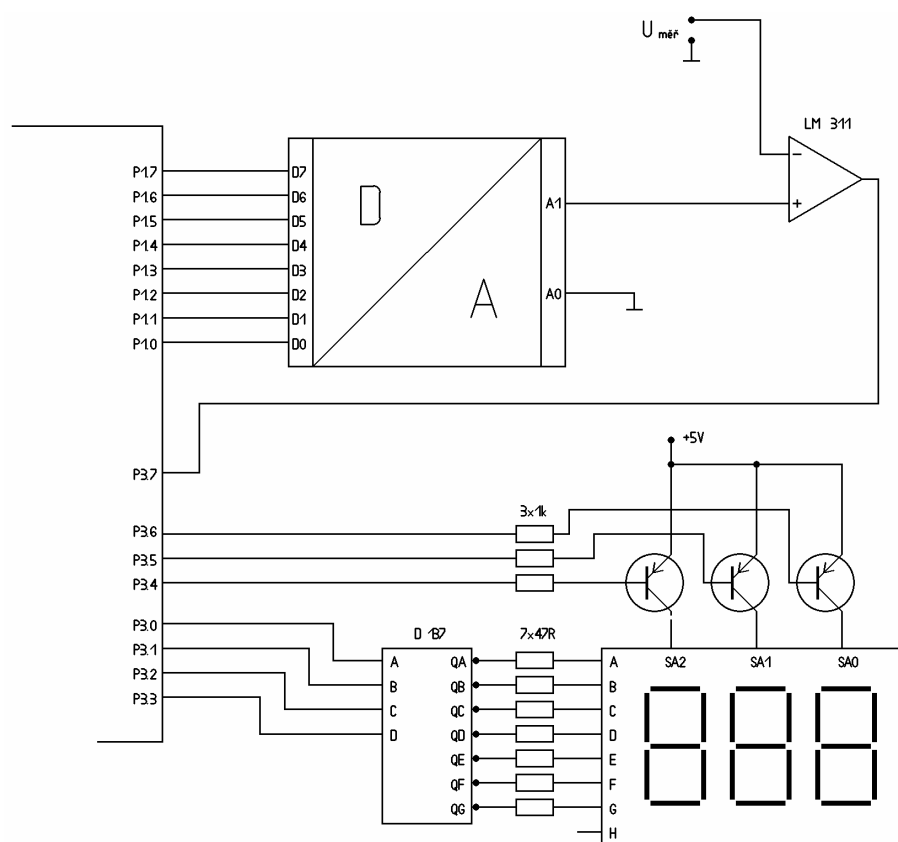
1. Program *DISP595A* zobrazí na displeji čísla z proměnných JED, DES, STA, TIS. Vyslání dat se provádí programově.
2. Program *DISP595B* zobrazí na displeji čísla z proměnných JED, DES, STA, TIS. Vyslání dat se provádí pomocí sériového kanálu (mód 0).

15. Modul D/A převodníku s postupnou aproximací

D/A převodník s postupnou aproximací se skládá z A/D převodníku, komparátoru a z aproximačního registru, který zde nahrazuje procesor. Při převodu procesor generuje čísla (v našem případě 8 bitů – čísla 0-255), která D/A převodník převede na analogovou hodnotu. Tato je přiváděna do komparátoru, kde se porovnává s měřenou hodnotou.

Pro názornost zvolíme rozsah měřeného napětí 0-2.55V. číslu 1 tedy odpovídá napětí 0.01V, číslu 2 napětí 0.02V, číslu 100 napětí 1.00V atd. Měřené napětí má hodnotu 0.999V.

Takže nejdříve přivedeme číslo 128, čímž rozdělíme interval 0-255 na dvě stejné části. Číslo 128 je binárně 1000 0000b, to znamená, že BIT 7 má úroveň log. 1.



Odpovídající napětí porovnáme s měřeným napětím. Bude-li měřená napětí větší než 1.28V (měřená napětí je z intervalu 1.28 – 2.55V), pak bude na výstupu komparátoru log 0 (log. 0 na P3.7) a BIT 7 zůstane v log. úrovni 1. V našem případě má měřená napětí hodnotu 0.999V, tedy nižší než 1.28V, na výstupu komparátoru bude log.0 a proto BIT 7 uvedeme do stavu log. 0. Tímto jsme vybrali interval 0-127. Nyní tento interval rozpůlíme tím, že uvedeme do stavu log. 1 BIT 6 (0100 0000b). Tomuto číslu odpovídá napětí 0.64V, měřená napětí je vyšší, na P3.7 je log. 0, BIT 6 zůstane v log1. Tedy dále budeme půlit interval 0.64 – 1.27. Dále jen zkratkovitě:

BIT 5 – log.1(0110 0000b) – tomu odpovídá napětí 0.96V, měřená je vyšší, P3.7 je v log.0, BIT 5 zůstane v log. 1.

BIT 4 – log. 1 (0111 0000b) – 1.12V, měřené je nižší, P3.7 je v log. 1, BIT 4 do log. 0
BIT 3 – log. 1 (0110 1000b) – 1.04V, měřené je nižší, P3.7 je v log. 1, BIT 3 do log. 0
BIT 2 – log. 1 (0110 0100b) – 1.00V, měřené je nižší, P3.7 je v log. 1, BIT 2 do log. 0
BIT 1 – log. 1 (0110 0010b) – 0.98V, měřené je vyšší, P3.7 je v log. 0, BIT 1 zůstane v log. 1
BIT 0 – log. 1 (0110 0011b) – 0.99V měřené je vyšší, P3.7 je v log. 0, BIT 0 zůstane v log. 1

Výsledné číslo je 0110 0011b – 99 a tomu odpovídá napětí 0.99 V. Tedy nejbližší nižší než je měřené napětí.

Program pro modul A/D převodník s postupnou aproximací:

1. Program *APROX* provádí převod napětí z intervalu $(-0.80V) - (+0.80V)$ na čísla 0 – 255. Napětí $-0.80V$ odpovídá číslo 0, napětí 0V odpovídá číslo 127, napětí $+0.80V$ odpovídá číslo 255.

16. Modul LCD displej

Modul LCD displeje obsahuje řadič a dvojřádkový LCD displej, kde na každém řádku lze zobrazit 16 znaků. Každý znak je složen z 35 bodů organizovaných do matice 5x7. Znaky, které lze zobrazit jsou uvedeny v tabulce v katalogu GM ELEKTRONIC a zadávají se pomocí ASCII kódu. Řadič HD44780 od firmy HITACHI nabízí velké množství funkcí. Po připojení řadiče na napájecí napětí je nutné provést počáteční nastavení řadiče a naprogramovat tyto funkce. Počátečním nastavením řadiče se myslí způsob komunikace – čtyřbitový nebo osmibitový, nastavení počtu řádků, nastavení fontů. Programované funkce jsou – pohyb kurzoru vlevo nebo vpravo, posun textu vlevo nebo vpravo, zapnutí, vypnutí nebo blikání kurzoru.

Toto nastavení se provádí v podprogramu LCDINI. Jakým způsobem se provádí nastavení není nutné rozebírat, neboť v něm nebudeme provádět změny, pouze si vysvětlíme, jak bude displej fungovat.

- Komunikaci nastavíme čtyřbitovou, to znamená, že osmibitová data nebo instrukce budeme přesouvat nadvakrát po čtyřech bitech.
- Nastavíme posun kurzoru vpravo, to znamená, že po zapsání znaku na určité místo se další znak zapíše o jednu pozici doprava.
- Kurzor vypneme, to znamená, že nebude vidět.
- Font nastavíme na 5x7, protože toto uspořádání má připojený displej.

Popis řídicích bitů:

- RS – bit P1.5 - je-li RS=0 vysílaný byt je instrukce (smazání displeje, přechod na začátek řádku, přechod na jiný řádek), RS=1 vysílaný byt jsou data, která se zapíší na displej
- E – bit P1.4 - sestupná hrana na tomto bitu zahájí přenos dat nebo instrukce
- VDD – přes tranzistor na bit P1.7 - P1.7=0 zapojí napájení řadiče
- VO – regulace kontrastu - není připojen, je vhodné připojit na jezdec trimru zapojeného mezi 0V a +5V

Program pro ovládání displeje je tvořen několika podprogramy. Voláním těchto podprogramů se provedou tyto funkce:

- LCDINI – podprogram pro počáteční nastavení řadiče, volá se jen po resetu
- RAD1, RAD2 – přesun kurzoru na řádek 1 (řádek 2), pozice je uložena v Acc
- CLRLCD – smazání displeje
- ZAPBLIK – zapne blikání znaku na kterém je kurzor
- VYPBLIK – vypne blikání znaku
- ZAPKUR – zapne zobrazování kurzoru (čára pod znakem)
- VYPKUR – vypne zobrazování kurzoru
- VYSDATA – zapíše na aktuální pozici displeje znak, jehož ascii kód je v ACC
- VYSINST – provede instrukci, která je uložena v ACC

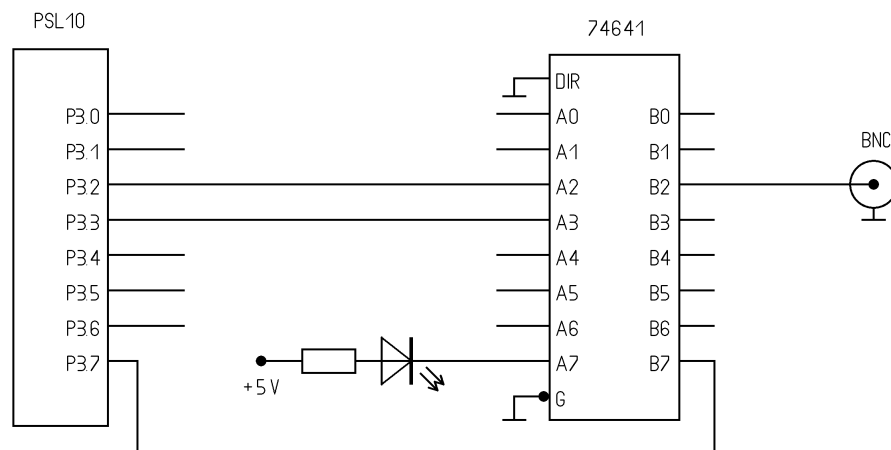
Programy pro modul LCD displej:

1. Program *LCD1* vypisuje postupně po 300ms všechny možné znaky.

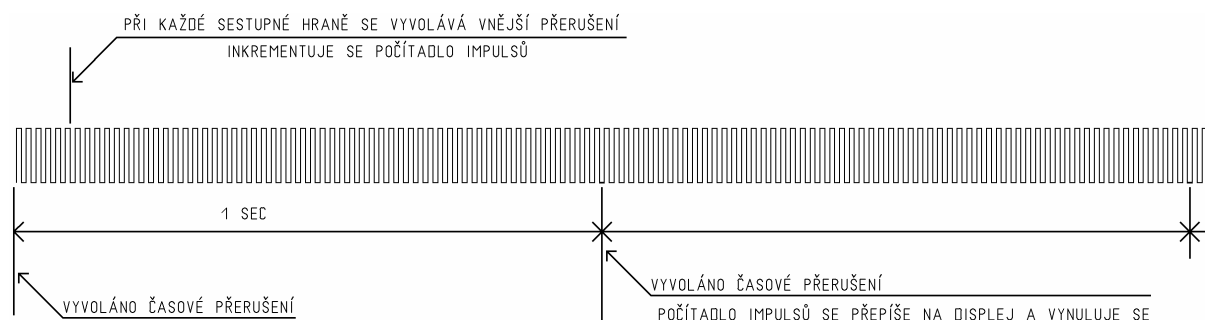
2. Program *LCD2* napíše na displej na první řádek nápis „LCD DISPLEJ“ a na řádek druhý nápis „2 radky 16 znaku“.
3. Program *LCD3* vypíše na první řádek nápis „Cidlo 1“ a na řádek druhý nápis „Teplota: “XY,Z „°C“. X,Y,Z jsou proměnné, do kterých je možno zapsat teplotu (je zde naznačen převod čísla na ascii kód).

17. Modul pro měření frekvence a periody

Modul pro měření frekvence a periody umožňuje připojení signálu TTL na bit P3.2 (vstup vnějšího přerušení). Vstup modulu je tvořen BNC konektorem pro jednodušší připojení generátoru TTL signálu. Na bit P3.7 je připojena LED dioda, která může sloužit pro signalizaci vyvolávání přerušení tím, že bude P3.7 negovat v každém přerušení.



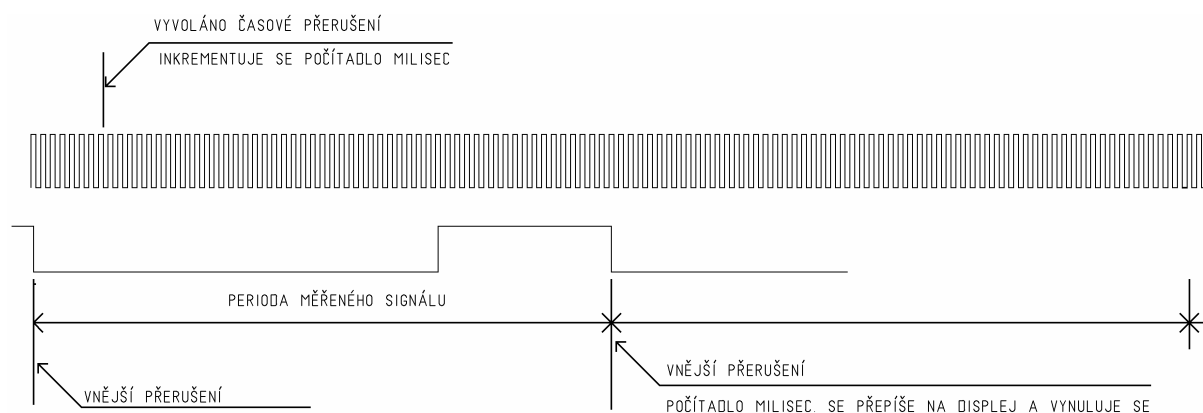
Při měření frekvence se provádí počítání impulsů v podprogramu pro vnější přerušení. Po uplynutí jedné sekundy se napočítané impulsy přepíše na displej, proměnné se vynulují a začne počítání znovu.



Měření periody se používá pro signály s malou frekvencí, kde by měření frekvence muselo být prováděno dlouhou dobu. Např. při měření frekvence 5,25Hz při době měření 1sec

získáme číslo 5, což je velmi nepřesné a pro získání čísla 5,2 musíme dobu měření prodloužit na 10 sec.

Při měření periody se počítají krátké časové intervaly (1ms). Po příchodu sestupné hrany se počet intervalů přepíše na displej, proměnné se vynulují a začne počítání znovu.

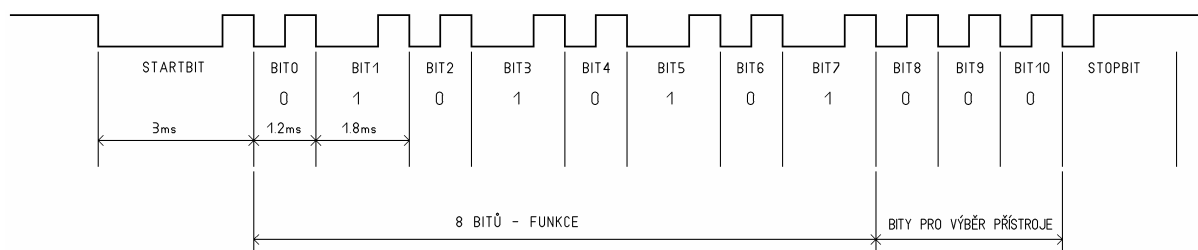


Programy pro modul měření frekvence:

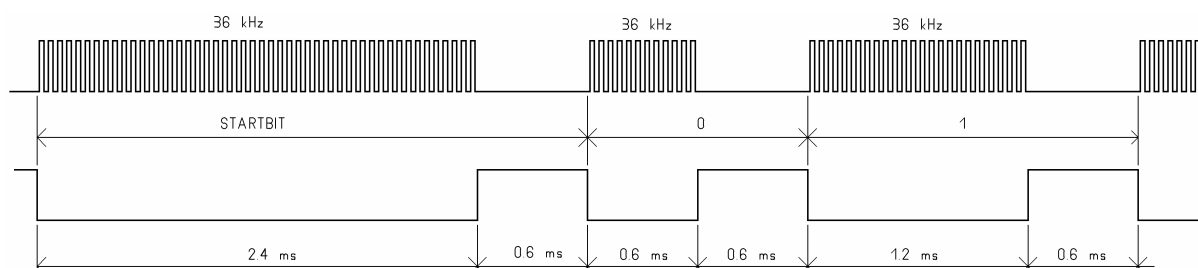
1. Program *MFREK* měří frekvenci signálu a zobrazuje na osmimístném displeji připojeném na port P1.
2. Program *MPER* měří periodu signálu a zobrazuje na osmimístném displeji připojeném na port P1.

18. Modul přijímače dálkového ovládání SONY

Dálkové ovládání SONY vysílá 13-ti bitový kód. První bit je startbit, dále se vysílá 8 bitů dat, 3 bity pro výběr přístroje a stopbit.



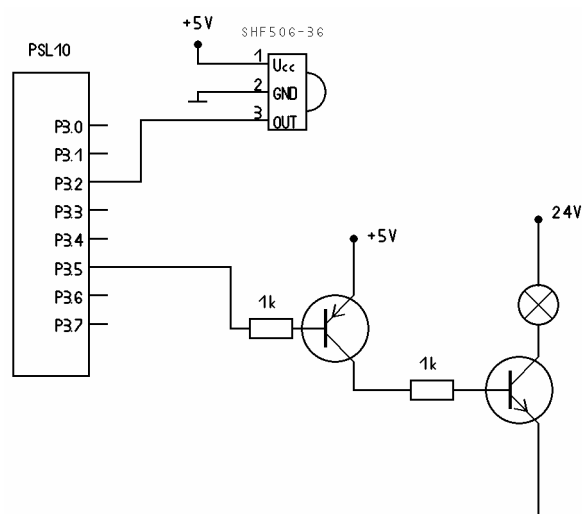
Na horním obrázku je celý kód, který začíná startbitem dlouhým 3ms, dále následuje osm bitů funkce tlačítka (např. PLAY, POWER, STOP apod.). V tomto případě je vysílána kombinace 01010101, následují tři bity pro výběr přístroje, pro který je povel určen (CD, TUNER, TV, VIDEO apod.). V tomto případě je vysílána kombinace 000.



Na tomto obrázku jsou detailně zobrazeny všechny tři typy vysílaných bitů. V horní části je zobrazen průběh z vysílače. Na začátku každého bitu se vysílá IR záření s frekvencí 36 kHz u startbitu 2.4ms, u nuly 0.6ms a u jedničky 1.2ms. Dále následuje doba po kterou se nevysílá a ta je 0.6ms. V dolní části obrázku je průběh za infrapřijímačem s tvarovačem. To znamená, že z vysílaného signálu je odfiltrována nosná frekvence 36kHz a signál je invertován a upraven pro logiku TTL.

Při přijímání a dekódování signálu budeme sledovat sestupné hrany, k čemuž nám poslouží vnější přerušovací vstup (INT0). Nejdříve budeme ignorovat impulsy pokud nebudou vzdáleny 3ms (STARTBIT) a od této chvíle budeme zjišťovat čas mezi impulsy a zapisovat nuly a jedničky podle toho, bude-li čas 1.2ms nebo 1.8ms. Toto je možno provést několika způsoby. V programu DOSONY1 nastavíme časovač na 50μs, v proměnné CAS počítáme, kolikrát dojde k přetečení časovače mezi sestupnými hranami, z TABDO vyhodnotíme o který typ bitu se jedná (STARTBIT, 0,1) a podle proměnné POCBIT čekáme na STARTBIT, nebo zapisujeme jedničky a nuly do proměnné KOD.

Připojení tzv. přijímací kostky (výše zmíněný infrapřijímač s tvarovačem) je naznačeno na obr., kde je slouží jako přijímač DO pro řízení jasu žárovky napájené ss napětím.



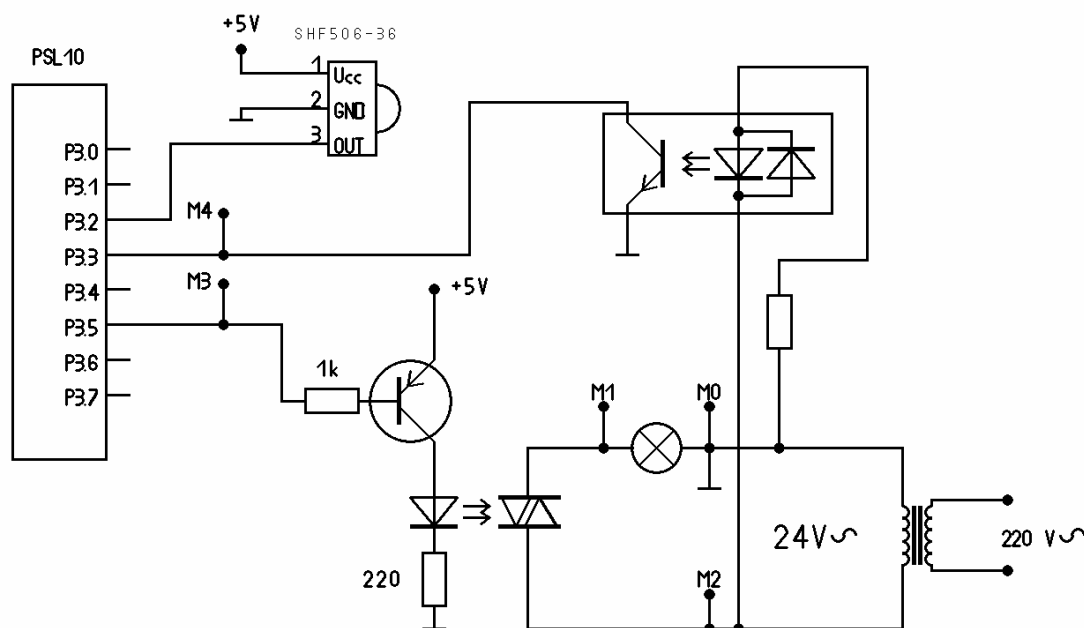
A v tomto obrázku přijímací kostka slouží jako přijímač DO pro řízení jasu žárovky napájené ss napětím.

Programy pro moduly DO:

1. Program *DOSONY1* čte dálkové ovládání a kódy zobrazuje na modulu LED (P1).
2. Program *DOSONY2* reaguje na dva povely DO, na jeden povel modul LED rozsvítí, na druhý povel modul LED zhasne.

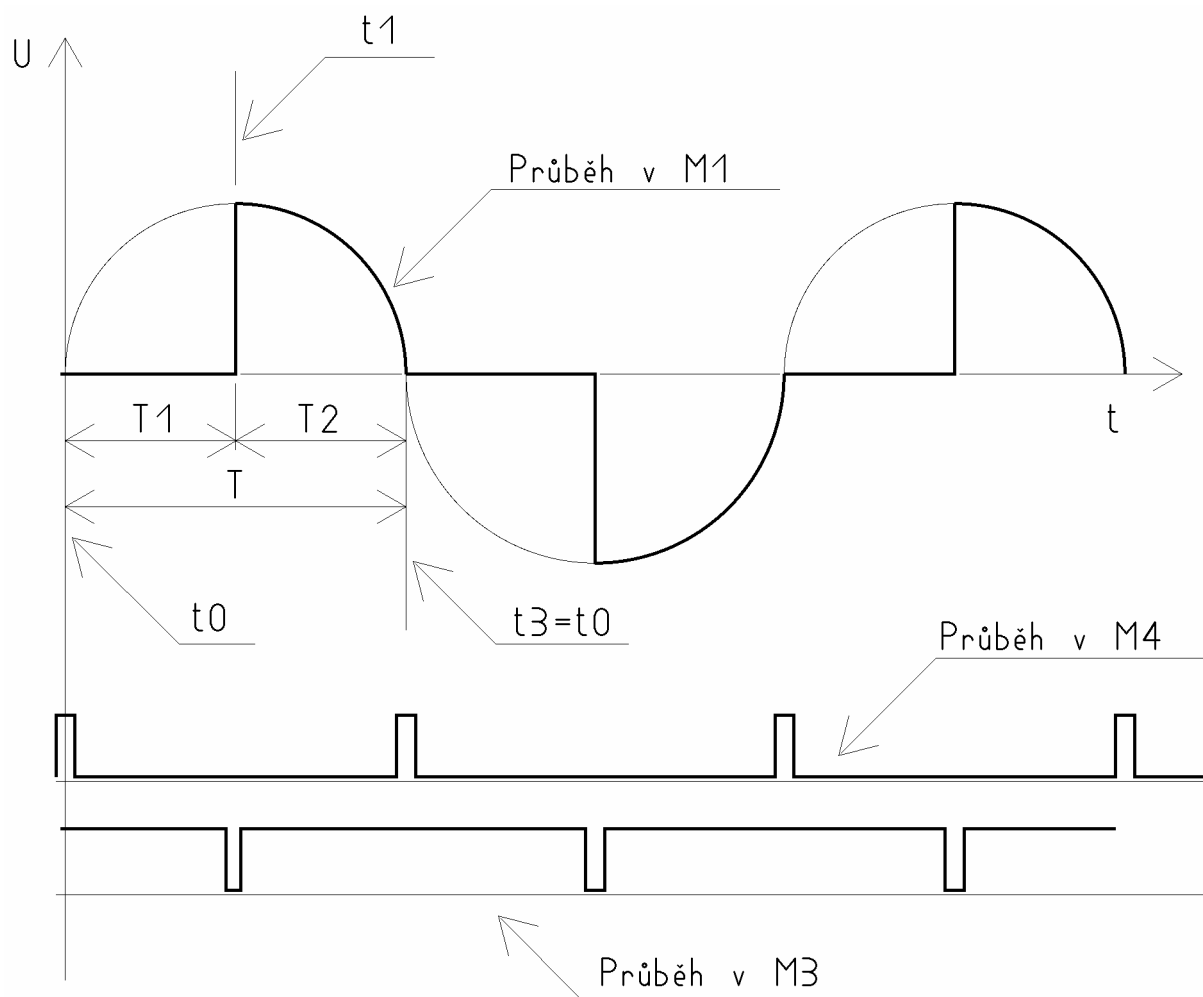
19. Modul regulace jasu žárovky – triakem

Zapojení modulu je na obr.:



Pro regulaci jasu žárovky je nutné měřit průchod napětí nulou. K tomu slouží optočlen s antiparalelním zapojením diod. Tento optočlen měří oba dva průchody nulou (při přechodu z kladné půlvlny do záporné i naopak). Tranzistor optočlenu je připojen na bit INT1. Žárovka je spínaná triakem, který je spínán optotriakem, ten je spínán tranzistorem připojeným na bit P3.5. Optotriak zajistí galvanické oddělení procesoru od střídavého obvodu.

Z obvodu jsou vyvedeny měřicí body M0 – M4. Mezi body M0 – M2 naměříme úplný sinusový průběh, mezi body M0 – M1 napětí regulované, mezi body M0 – M3 spínací impulsy triaku a mezi M0 – M4 impulsy průchodu nulou.



Na obr. je zobrazen průběh napětí na žárovce při 50% jasu. Regulace se provádí řízením spínacího úhlu (v našem obrázku mu odpovídá čas $T1$). Čas $T1$ můžeme měnit od 0 do 10 ms. Je-li $T1=0\text{ms}$, bude jas 100%, bude-li $T1=10\text{ms}$, bude jas 0%.

Průchod nulou vyvolává vnější přerušení EXT1. Od tohoto okamžiku odpočítáme čas $T1$, potom sepneme triak (log. 0 na P3.5). Po 100 μs na P3.5 log.1, triak zůstane sepnutý do průchodu nulou.

Toto lze řešit několika způsoby. Nejjednodušší bude při vyvolání vnějšího přerušení spustit časovač a přednastavit ho na hodnotu, která odpovídá požadovanému jasu. Nabízí se možnost číst hodnotu pro časovač z tabulek (jedna pro TH a druhá pro TL). Potom máme možnost použít libovolný počet kroků a provést linearizaci regulace (použijeme-li krok stejné délky, bude regulace kolem středu půlvlny pomalejší než u okrajů půlvlny). Nebo máme možnost využít jednodušší variantu, ve které proměnnou JAS budeme měnit od čísla 255 – maximální jas do čísla 224 – minimální jas. Tuto hodnotu budeme přímo přednastavovat do registru TH. Přednastavení 224 při použití krystalu 11 059 200 Hz odpovídá času 8,8 ms. Musíme uvažovat, že vnější přerušení se vyvolá několik stovek μs po začátku půlperiody (způsobeno snímacím optočlenem), proto neodpovídá maximální přednastavení 10ms. Navíc regulace kolem krajů půlperiody je zanedbatelná.

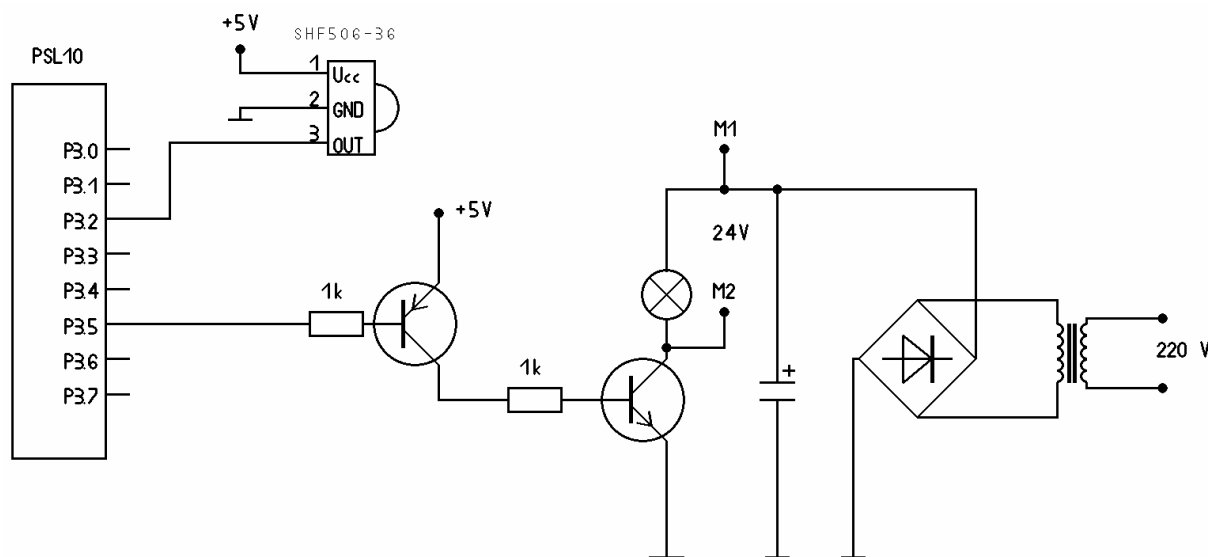
Při vyvolání časového přerušení provedeme sepnutí triaku na několik μs , vypneme časovač a čekáme na další vnější přerušení.

Programy pro modul regulace jasu triakem:

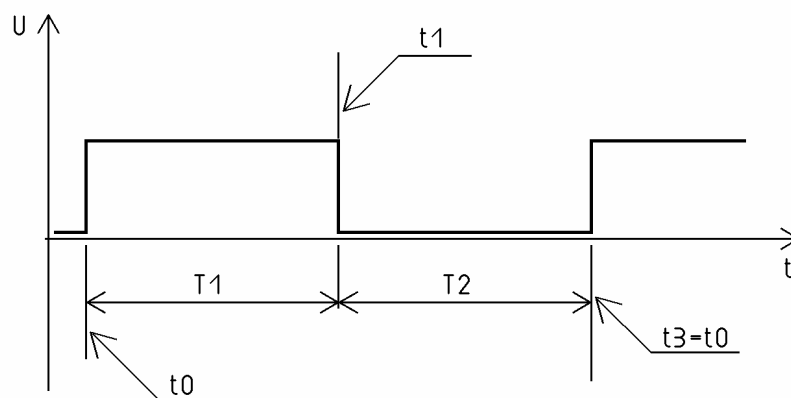
1. Program *REGTR11* čte dálkové ovládání SONY a pomocí tlačítek VOL+ a VOL- se rozsvítí a zhasne žárovka.
2. Program *REGTR12* čte klávesnici, pomocí které se tlačítky 1 a 3 reguluje jas, tlačítky 4 a 6 nastavuje přímo min. a max. hodnota jasu. Regulace jasu se provádí jednodušší variantou z výše popsaných.
3. Program *REGTR13* čte dálkové ovládání SONY a pomocí tlačítek VOL+ a VOL- se reguluje hodnota jasu. Regulace jasu se provádí jednodušší variantou z výše popsaných.

20. Modul regulace jasu žárovky – tranzistorem

Zapojení modulu je na obr.:



Regulaci jasu žárovky budeme provádět podobným způsobem jako v předchozí kapitole. Nejdříve sepneme tranzistor (bit P3.5 do log.0, čas t_0), proměnnou JAS (224 – 255) vložíme do TH, spustíme časovač. Po vyvolání přerušení od časovače vypneme tranzistor (P3.5 do log.1, čas t_1), vložíme do TH proměnnou JAS1 ($JAS1 = 224 + 256 - JAS$). Po vyvolání přerušení (čas $t_3 = t_0$) sepneme tranzistor atd. V tomto případě je možné měnit počet kroků (dolní hranici 224 je možno posouvat), ale je nutné dodržet podmínku, aby frekvence $f = 1 / (T_1 + T_2)$ byla vyšší než 50 Hz.



Programy pro modul regulace jasu žárovky – tranzistorem:

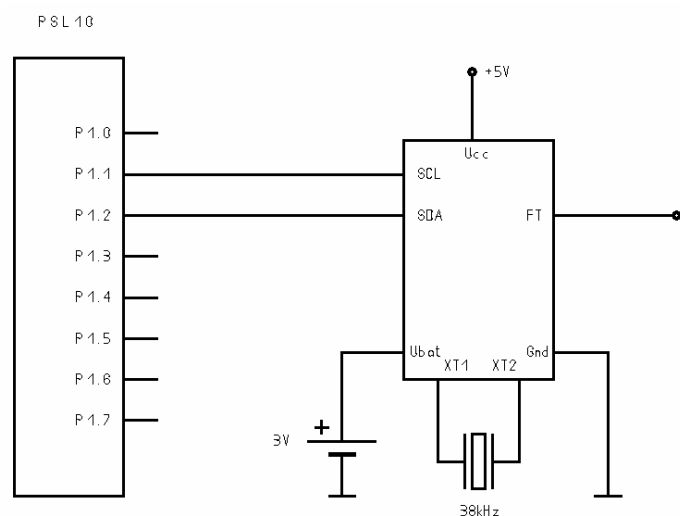
1. Program *REGTRAN1* čte klávesnici, pomocí které se tlačítky 1 a 3 reguluje jas, tlačítky 4 a 6 nastavuje přímo min. a max. hodnota jasu.
2. Program *REGTRAN2* čte dálkové ovládání SONY a pomocí tlačítek VOL+ a VOL- se reguluje hodnota jasu.

21. Modul obvodu reálného času (IIC)

Obvod reálného času MK 41T56 (dále jen RTC) je v podstatě 64 bytová paměť RAM typu CMOS ve které je dolních osm bytů vyhrazeno pro potřeby ukládání času (sec, min, hod), data (den v týdnu, den, měsíc, rok) a kalibraci. Obvod automaticky inkrementuje čas a datum. K obvodu je nutno připojit záložní baterii, krystal a napájení. Komunikace se provádí po sběrnici IIC, tedy po dvou vodičích (SDA, SCL).

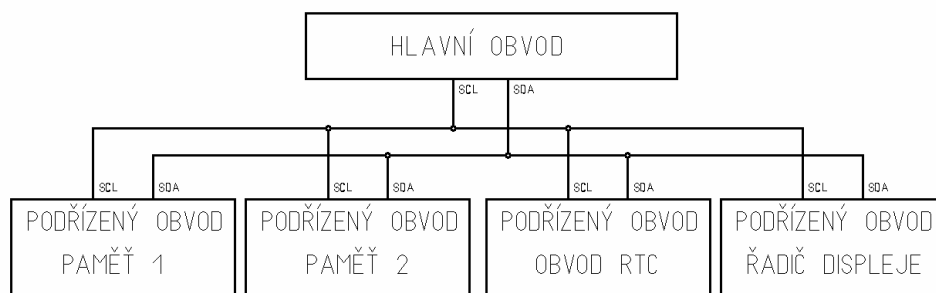
Adresa	D7	D6	D5	D4	D3	D2	D1	D0		
0	ST	DES. SEK			JED. SEK					
1	-----	DES. MIN			JED. MIN					
2	-----	-----	DES. HOD		JED. HOD					
3	-----	-----	-----	-----	-----	DEN V TÝDNU				
4	-----	-----	DES. DEN		JED. DEN					
5	-----	-----	-----	DMĚS	JED. MĚS					
6	DES. ROK				JED. ROK					
7	-----	-----	-----	KALIBRACE						

Připojení obvodu RTC k portu je na obr.:



Komunikace pomocí sběrnice IIC je sériová komunikace a některé typy procesoru 8051 mají sériový I/O který automaticky komunikuje tímto protokolem. V našem případě však budeme komunikaci provádět programově.

Při komunikaci IIC jsou propojeny vodiče SDA všech obvodů a stejným způsobem všechny vodiče SCL.



Hlavní obvod potom pro komunikaci s jedním z podřízených obvodů nejdříve tento obvod aktivuje vysláním jeho kódu a dále mu vyšle nebo od něj požaduje data.

Po aktivování obvodu je možné číst nebo vyslat i více bytů dat. My pro jednoduchost budeme komunikovat pro jedním bytu. Celá komunikace je následující.

Zápis jednoho bytu dat:

1. Vyšleme sedmibitový kód obvodu a jako osmý bit log. 0 (zápis) – 1101 0000b. Tím aktivujeme obvod, který se přepne na čekání na data (vyslání dat -přivedeme 1. bit dat na SDA a provedeme nástupnou a sestupnou hranu na SCL, přivedeme 2. bit dat atd.).
2. Vyšleme adresu, na kterou chceme data zapisovat (0 – 63).
3. Obvod RTC stále očekává data, proto můžeme přímo vyslat data.

Čtení jednoho bytu dat:

1. Vyšleme sedmibitový kód obvodu a jako osmý bit log. 0 (zápis) – 1101 0000b. Tím aktivujeme obvod, který se přepne na čekání na data.
2. Vyšleme adresu, ze které chceme data číst (0 – 63).
3. Vyšleme sedmibitový kód obvodu a jako osmý bit log. 1 (čtení) – 1101 0001b. Tím aktivujeme obvod, který se přepne na vysílání dat.
4. Přečteme 1 byt dat (po každé nástupné hraně na SCL se na SDA objeví jeden bit dat).

V programech pro modul RTC jsou dva základní podprogramy – CTENI a ZAPIS. Chceme-li zapsat do obvodu RTC 1 byt, provedeme následující tři instrukce:

1. `MOV ADRPRO,#20`
2. `MOV ADDRRTC,#0`
3. `CALL ZAPIS`

V ADRPRO je adresa v procesoru, ze které se mají data přesunout na adresu v ADDRRTC. Třetí instrukce provede přesun dat. V tomto případě z adresy 20 v procesoru se přesunou data na adresu 0 v RTC.

Chceme-li přečíst 1 byt z RTC provedeme následující tři instrukce:

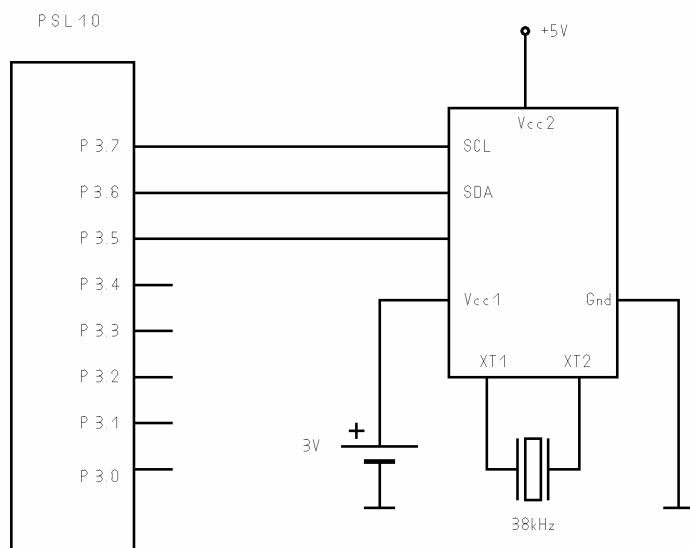
1. `MOV ADRPRO,#20`
2. `MOV ADDRRTC,#0`
3. `CALL CTENI`

Z adresy 0 v RTC se přesunou data na adresu 20 v procesoru.

Programy pro modul RTC:

1. Program *RTC1* zapíše čas do RTC a ten pak nepřetržitě čte a zapisuje na osmimístný displej.
2. Program *RTC2* zapíše čas a datum do RTC a ten pak čte a střídavě zobrazuje datum a čas.

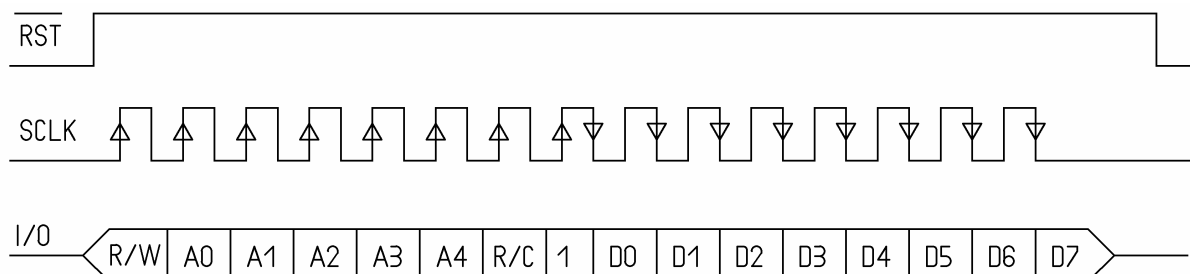
22. Modul obvodu reálného času DS 1302



Obvod DS 1302 pracuje na stejném principu jako obvod MK 41T56. Obsahuje 7 bytů pro čas a datum, dále 1 byt pro zakázání zápisu, 1 byt řízení nabíjení záložní baterie a 31 bytů paměti. Komunikace s obvodem se provádí pomocí třech bitů - RST, SCLK, I/O. Komunikace je odlišná od komunikace IIC. Provádí se po jednom bytu, nebo všechny byty najednou (BURST mód).

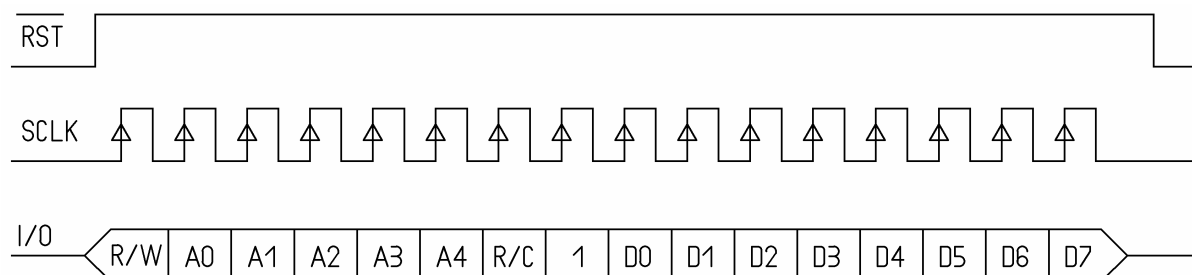
Při komunikaci po jednom bytu nejdříve vyšleme řídicí byt a potom provádíme čtení nebo zápis dat. Řídicí byt obsahuje bit R/W – čtení/zápis, pět bitů adresy, R/C – paměť/hodiny a log. jedničku.

Čtení: Nejdříve uvedeme bit RST do stavu log. 1, potom na bit I/O přivádíme řídicí byt a po každém bitu provedeme nástupnou a sestupnou hranu na SCLK. Po posledním bitu řídicího bytu (log. 1), se po každé sestupné hraně na SCLK bude na bitu I/O jeden bit dat. Po osmém bitu dat uvedeme RST do log. 0.



Zápis:

Nejdříve uvedeme bit RST do stavu log. 1, potom na bit I/O přivádíme řídicí byt a po každém bitu provedeme nástupnou a sestupnou hranu na SCLK. Stejným způsobem vysíláme osm bitů dat. Po osmém bitu dat uvedeme RST do log. 0.

**Burst mód:**

Čtení nebo zápis v burst módu se provádí stejným způsobem, pouze adresa je nahrazena číslem pro burst mód (11111), RST zůstává v log. 1 a provádí se čtení nebo zápis všech bytů dat najednou.

Tabulka registrů pro hodiny:

SEC	1	0	0	0	0	0	0	R/W		CH	10SEC				SEC	
MIN	1	0	0	0	0	0	1	R/W		0	10MIN				MIN	
HOD	1	0	0	0	0	1	0	R/W		12/24	0	10/AP	HOD	HOD		
DATUM	1	0	0	0	0	1	1	R/W		0	0	10 DATUM			DATUM	
MĚS	1	0	0	0	1	0	0	R/W		0	0	0	10M	MĚSÍC		
DEN V. T.	1	0	0	0	1	0	1	R/W		0	0	0	0	0	DEN V.T.	
ROK	1	0	0	0	1	1	0	R/W		10ROK				ROK		
ZÁPIS	1	0	0	0	1	1	1	R/W		WP	0	0	0	0	0	0
NABÍJENÍ	1	0	0	1	0	0	0	R/W		TCS				DS	RS	
BURST	1	0	1	1	1	1	1	R/W		DATA SEC ÷ NABÍJENÍ						

Bit CH =1 zastaví oscilátor a uvede obvod do režimu nízké spotřeby (nižší než 100 nA).

CH =0 oscilátor běží (po připojení k napájení není definován).

Bit 12/24 = 0 – režim 24h (bit 10/AP – desítky hodin).

12/24 = 1 - režim 12h (bit 10/AP – AM/PM).

- Bit WP = 0 zápis povolen
WP = 1 zápis zakázán
- Byt DEN V.T. je den v týdnu (1 ÷ 7).
- Byt NABÍJENÍ umožňuje zvolit nabíjecí proud, případně nabíjení vypnout.

Po vyslání řídicího slova pro BURST mód je možno přečíst postupně byty SEC ÷ NABÍJENÍ.

Tabulka registrů pro paměť:

RAM0	1	1	0	0	0	0	0	R/W		RAM DATA 0
RAM30	1	1	1	1	1	1	0	R/W		RAM DATA 30
BURST	1	1	1	1	1	1	1	R/W		RAM DATA 0 ÷ 30

Po vyslání řídicího slova pro BURST mód je možno přečíst postupně byty RAM0 ÷ RAM 30.

Program pro zápis do registrů hodin:

- do proměnné ADR zapíšeme číslo adresy, do které chceme zapisovat
- do proměnné DAT zapíšeme data, která chceme zapisovat
- zavoláme podprogram WRBYTC a tím data zapíšeme

Program pro čtení z registrů hodin:

- do proměnné ADR zapíšeme číslo adresy, ze které chceme číst
- zavoláme podprogram RDBYTC a tím přečteme data z registru do proměnné DAT

Program pro čtení BURST hodin:

- zavoláme podprogram BURSTC
- data se ukládají postupně na adresy 30 ÷ 39 (30 – SEC,....39 – NABÍJENÍ)
- adresy 30 ÷ 39 je možno v programu BURSTC změnit

Program pro zápis do registrů paměti:

- do proměnné ADR zapíšeme číslo adresy, do které chceme zapisovat
- do proměnné DAT zapíšeme data, která chceme zapisovat
- zavoláme podprogram WRBYTR a tím data zapíšeme

Program pro čtení z registrů paměti:

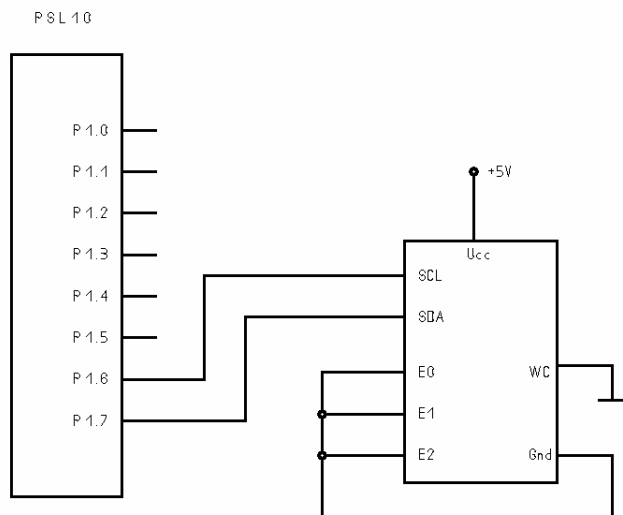
- do proměnné ADR zapíšeme číslo adresy, ze které chceme číst
- zavoláme podprogram RDBYTR a tím přečteme data z registru do proměnné DAT

Program pro čtení BURST paměti:

- zavoláme podprogram BURSTR
- data se ukládají postupně na adresy 50 ÷ 80 (50 – RAMDATA0...80 – RAMDAT30)
- adresy 50 ÷ 80 je možno v programu BURSTC změnit

23. Modul paměti EEPROM (IIC)

Paměť 24C02 má organizaci 256x8 bitů. Komunikace se provádí po dvou vodičích protokolem IIC. Zapojení paměti je na obr.:



Platí zde vše co bylo napsáno u obvodu RTC. Změní se samozřejmě kód, kterým se aktivuje obvod. Navíc tento kód obsahuje tři bity E0,E1,E2. Bity E0,E1,E2 najdeme též na třech vývodech paměti, kde je lze zapojit na log.0 nebo log.1. Podle toho se změní kód dané paměti. V našem případě jsou všechny tři bity zapojeny na log.0, v kódu tedy budou místo E0,E1,E2 tři nuly. Kód paměti je následující:

1	0	1	0	E2	E1	E0	R/W
1	0	1	0	0	0	0	R/W

Bity E0,E1,E2 lze zapojit osmi různými způsoby na úrovně log.0 a log.1, tím vznikne osm různých kódů a tedy lze připojit najednou na jednu sběrnici IIC osm pamětí. Vstup WC povoluje zápis do paměti.

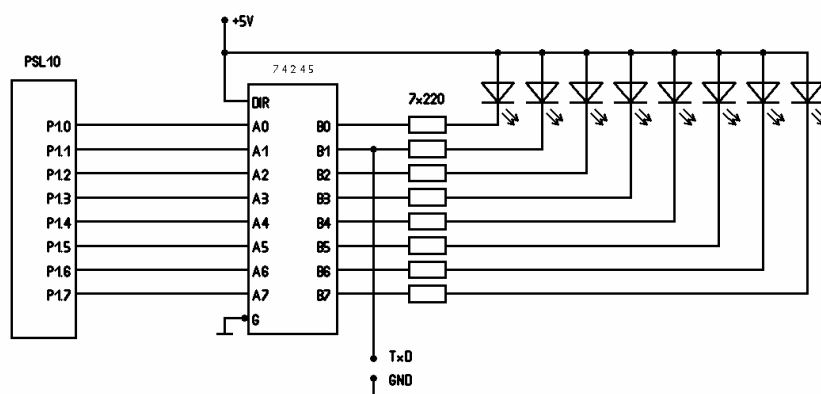
Ukládání a čtení dat se provádí stejným způsobem jako u obvodu RTC.

Program pro modul paměť EEPROM:

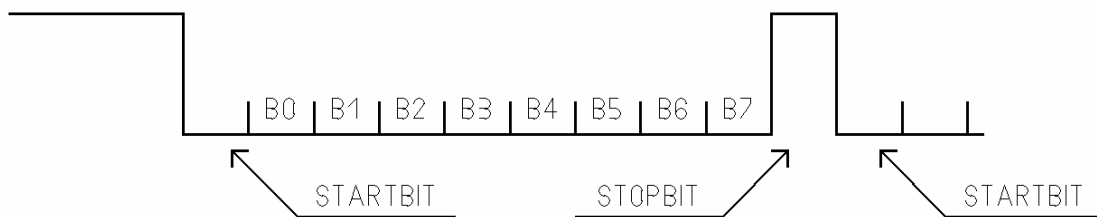
1. Program *PAM1* po resetu zobrazí na displeji pomlčky na 1 s, pak přečte data z adres 0-7 paměti EEPROM, zobrazí je na displeji a po 100 ms inkrementuje a zobrazuje na displeji. Provedeme-li reset modulu, inkrementování bude pokračovat od poslední hodnoty před resetem.

24. Modul pro seriovou komunikaci

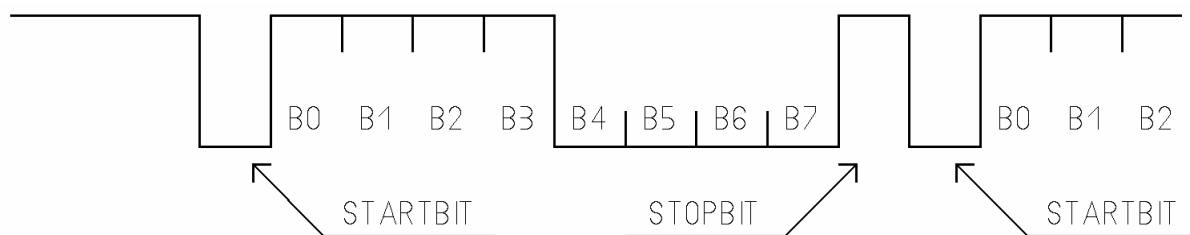
Tento modul je velmi jednoduchý a slouží pouze k vyvedení bitu TxD pro účely měření průběhu na sériovém kanálu osciloskopem. Mikroprocesor nabízí čtyři módy sériové komunikace. Mód 0 je synchronní (data jsou synchronizovaná hodinovými impulsy) a módy 1, 2, 3 jsou 8-mi a 9-ti bitový UART (vysílá se startbit, 8 nebo 9 bitů dat a stopbit).



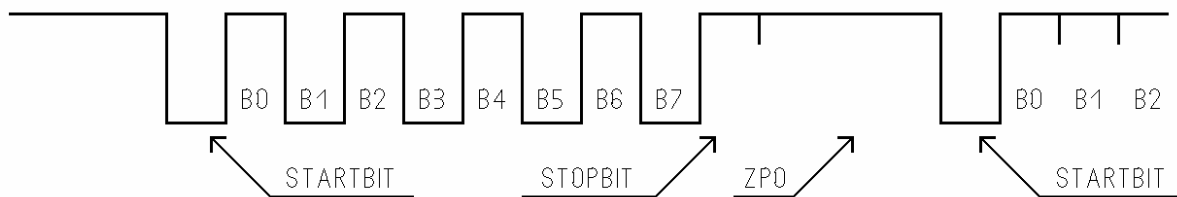
Na následujícím obrázku je osmibitový UART. Všechny bity (startbit, stopbit, datové bity) jsou stejně dlouhé. Při použití přenosové rychlosti 2400 bitů/s je doba vysílání jednoho bitu 416 μ s. Ihned po skončení stopbitu začíná startbit. Vysílá se deset bitů, takže vyslání jednoho bytu dat bude trvat 4,16 ms.



Na dalším obrázku je průběh, který získáme při vysílání čísla 00001111b při využití přerušení od sériového kanálu – vysílače (po odvyslání stopbitu se vyvolá přerušení ve kterém se znovu zapíše číslo 00001111b do SBUF a tím ihned vyšle).



Na posledním obrázku je průběh při vysílání čísla 01010101b, kdy mezi jednotlivými byty je časové zpoždění. Na tomto průběhu je možno dobře odečíst čas vysílání jednoho bitu.

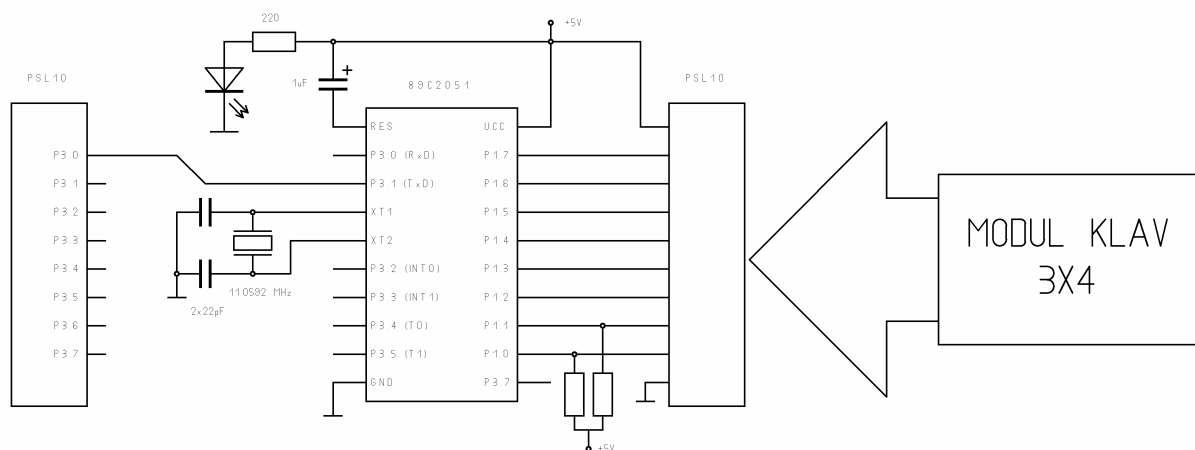


Programy pro modul sériové komunikace:

1. Program *SKOM1* vysílá nepřetržitě číslo 00001111b, využívá přerušení od vysílače.
2. Program *SKOM2* vysílá číslo 01010101b, kdy mezi vysílanými byty je časové zpoždění.

25. Modul pro sériovou komunikaci – vysílač

Na obrázku je modul vysílače. Připojuje se do emulátoru na port P3. Zároveň obsahuje jednu patici PSL10, do které se připojuje modul klávesnice. V procesoru v modulu je program, který čte klávesnici a při stisku tlačítka vyšle rychlostí 2400 bitů/s číslo klávesy po sériovém portu.

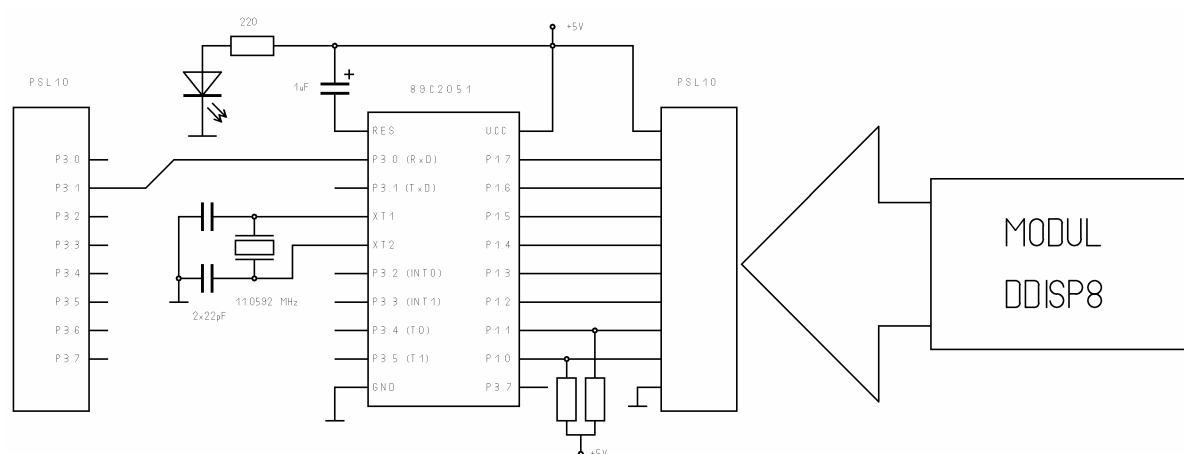


Program pro modul sériové komunikace – vysílač:

1. Program *SKOMDISP* přijímá data, která přicházejí po sériovém kanálu z modulu. Data (číslo klávesy) zobrazí na osmimístném displeji na pozici jednotek a ostatní čísla posune vlevo.

26. Modul pro sériovou komunikaci - přijímač

Na obrázku je modul přijímače. Připojuje se do emulátoru na port P3. Zároveň obsahuje jednu patici PSL10, do které se připojuje modul osmimístný dynamický displej. V procesoru v modulu je program, který čte data na sériovém kanálu rychlostí 2400 bitů/s číslo zobrazí na displeji na pozici jednotek, ostatní čísla posouvá vlevo.



Program pro modul sériové komunikace – přijímač:

1. Program *SKOMKLAV* čte klávesnici a při stisku klávesy vysílá číslo klávesy po sériovém kanálu rychlostí 2400 bitů/s (modul číslo přijme a zobrazí na displeji).