České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra měření

TESTOVACÍ MODUL

Vedoucí práce: Ing. Jan Fischer, CSc. Autor: Peter Palaj

Praha 2009

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne .....

podpis

Poděkování

Děkuji Ing. Janu Fischerovi, CSc., Ing. Zdeňkovi Rozehnalovi za cenné a podnětné připomínky a informace k této bakalářské práci a Ing. Ondrejovi Pribulovi za poskytnutí a přizpůsobení programu zobrazovače přímo k Testovacímu modulu.

Děkuji Ing. Kateřině Daníčkové a panu Pavlovi Kandrikovi za gramatické korektury této bakalářské práce.

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra měření

Akademický rok **2007-2008**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	**Peter Palaj**
Obor:	**Kybernetika a měření**
Název tématu česky:	**Testovací modul**
Název tématu anglicky:	**Testing Module**

Zásady pro vypracování:

Navrhněte a realizujte testovací modul s mikropočítačem řady Atmel - ATmega, případně i jiným mikropočítačem, pro použití při výuce mikroprocesorové techniky a programovatelných logických obvodů.

Modul bude poskytovat statické úrovně i testovací průběhy logických signálů podle nastavení z PC. Dále bude číst stav na vstupech a tuto informaci přenášet prostřednictvím rozhraní RS232 na nadřazené PC, kde se zobrazí.

Modul bude také umožňovat funkci velmi jednoduchého logického analyzátoru, číslicového voltmetru a digitálního osciloskopu se zobrazením výsledků na PC.

Vytvořte potřebné programové vybavení pro řídicí procesor v modulu i programy pro nadřazené PC.

Seznam odborné literatury:

[1] Vedral, J., Fischer, J.: Elektronické obvody pro měřicí techniku. Vydavatelství ČVUT, Praha 1999

[2] Atmel: ATmega 8-data sheet (www.atmel.com/.../doc2486.pdf)

[3] Atmel: 8-bit AVR Instruction set (www.atmel.com/.../doc0856.pdf)

Vedoucí bakalářské práce:	Ing. Jan Fischer, CSc.
Datum zadání bakalářské práce:	11. února 2008
Platnost zadání do[1]:	24. ledna 2009

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická -4-
L.S.

Prof. Ing. Vladimír Haasz, CSc.
vedoucí katedry

Doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 11.2.2008

[1] Platnost zadání je omezena na dobu dvou následujících semestrů.

Anotace

Výsledkem této práce je v assembleru naprogramovaný mikroprocesor, který umožňuje měřit logické a napěťové úrovně, generovat frekvenci, logické úrovně a specielní signály. Přípravek komunikuje s obsluhou pomocí sériového portu, přičemž obsluha přes počítač může z klávesnice nastavit požadované parametry a na monitoru sledovat naměřené a nastavené údaje. Cílem přípravku je usnadnění práce na cvičeních se zaměřením na mikroprocesorovou techniku.

Annotation

The result of this work is a microprocessor programmed in assembler, allowing to measure logical and voltage levels and to generate frequencies, logical levels and special signals. The microprocessor communicates with operator via serial port, while the values can be set directly from PC. The input and output data are displayed on the PC monitor. The purpose of this appliance is to ease work during practical lessons focusing on microprocessor technology.

OBSAH

Seznam obrázků

1 Úvod

Cílem Českého vysokého učení technického v Praze, oboru Kybernetika a měření, není studenty jenom naučit teoretické znalosti oboru, ale taky snaha prakticky studenty seznámit s danou problematikou.

„Mikroprocesory v přístrojové technice (MIP)“ a „Návrh řídící části mikroprocesorů (NRM)“ jsou předměty, které studenta naučí teoretické znalosti z mikroprocesorové techniky a logických obvodů. Navíc si studenti na cvičeních s těmito obvody mohou prakticky vyzkoušet zapojení různých funkčních celků, kdy mikroprocesorem řídí další obvody, nebo zapojit měřící přístroje, například ohmmetr nebo voltmetr.

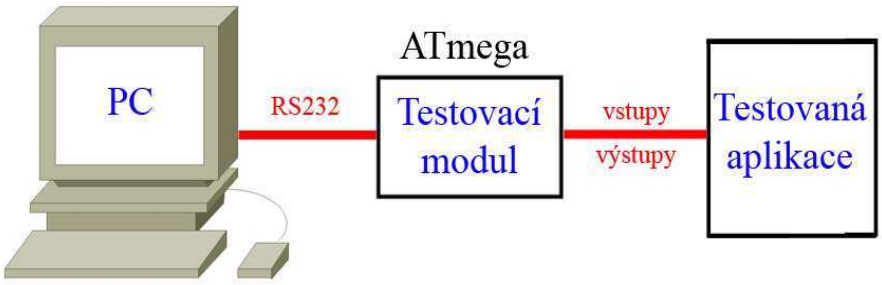
Cílem této bakalářské práce je vytvořit univerzální modul, který by ulehčil práci na cvičeních předmětu MIP a NRM. Při práci studenti často potřebují různé generátory, pomocné signály, měření logických úrovní, napěťových úrovní a podobně. Testovací modul, který je výsledkem této bakalářské práce, umožní studentům všechno zmíněné použít jednoduchým připojením naprogramovaného mikroprocesoru ke své aplikaci. Navíc po připojení k počítači mohou sledovat výpis měřených hodnot v programu hyperterminálu, který je součástí všech operačních systémů Windows, nebo v zobrazovači, který přímo ukáže naměřené děje graficky.

2 Problematika a její řešení

Jak již bylo zmíněno v úvodu, cílem této bakalářské práce je vytvořit přípravek, který by ulehčil studentům práci na cvičeních se zaměřením na „*mikroprocesorovou techniku*“ a „*programovatelné logické obvody*“. Před samotným začátkem řešení problematiky vývoje tohoto přípravku, bylo nezbytné se zamyslet nad možnostmi pro samotný použitý hardware a následně vyvinout a přizpůsobit potřebné funkce tak, aby výsledkem byl přípravek, jehož použití by bylo co nejjednodušší z hlediska jeho zapojení k testované aplikaci a jeho ovládání bylo co nejintuitivnější.

Při zamyšlení se nad prvně zmíněným problémem, tj. „jaký hardware použít“ se po krátkých úvahách a zhodnocení možností dospělo k závěru, že nejvhodnější pro tuto práci je použití samotného mikroprocesoru. Pro mnohé funkce, které byly navrženy, by z možných řešení mohlo být i použití kombinačních či sekvenčních logických obvodů. Pro jiné funkce by ale toto řešení buďto nebylo možné, nebo by bylo komplikované. Taky pro požadavek komunikace s nadřazeným PC by tato metoda nebyla vhodná.

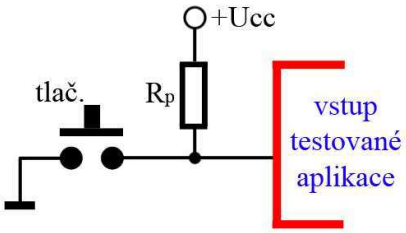
Mikroprocesorů různých typů a výrobců je v dnešní době na trhu mnoho. Vybrány byly mikroprocesory řady Atmel – ATmega, konkrétně ATmega8, kterého poměr cena/výkon je vynikající. Mikroprocesor ATmega8 se vyrábí v provedení SMD (32 vývodů) a v provedení PDIP28. Z hlediska využití pro tuto práci je ideální použít provedení PDIP, protože odpadá nutnost pájet mikroprocesor do plošného spoje. Zamáčknutím mikroprocesoru do kontaktního pole mohou k němu následně studenti přivádět potřebné přívody. Další výhodou je, že při poškození mikroprocesoru se jeho výměna provede vytažením z kontaktního pole.



Obr. 2.1 Blokové znázornění testování pomocí Testovacího modulu

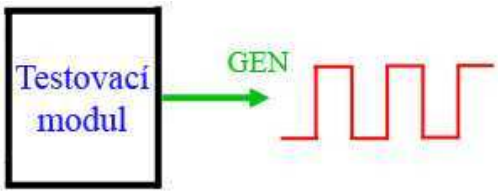
Dalším zmíněným problémem bylo vymyšlení samotných funkcí použitých v testovacím modulu. Inspirace pro použité funkce vycházely ze samotných studijních plánů předmětů MIP a NRM. Jako příklad uveďme úkol, kdy studenti mají pomocí GAL-u navrhnout sekvenční

čítač. Aby jej studenti mohli otestovat, musí na vstup čítače přivést pulsy, způsobující čítání čítače. Nejjednodušší varianta se nabízí použití tlačítka. Tlačítko z důvodů mechanických odskoků při stlačení, kdy místo jednoho žádaného pulsu vygeneruje pulsů více, pro použití v přístrojích, u kterých je snímání vstupů rychlé není vhodné. Ještě horší variantou je přivádět na vstup přímo logické úrovně pomocí kabelu.



Obr. 2.2 Zapojení tlačítka s pull-up rezistorem

Ideální možností je přivádět na čítač pulsy, které bude generovat příslušná funkce v mikroprocesoru. Vyřešit takový problém bylo jednoduché, stačí když na výstupním pinu mikroprocesoru měníme logickou úroveň, přičemž se mezi samotnými změnami úrovní udělá časová mezera. Tím získáme jednoduchý ***„GENERÁTOR PULSŮ“***. Požadavky pro rychlosti pulsů, jsou za různých situací jiné. Proto bylo ve funkci generátoru vyvedeno více vývodů s různými frekvencemi generovaných pulsů. Tyto vývody jsou označeny zkratkou ***„GEN“*** a hodnota za ní udává frekvenci generovaných pulsů.



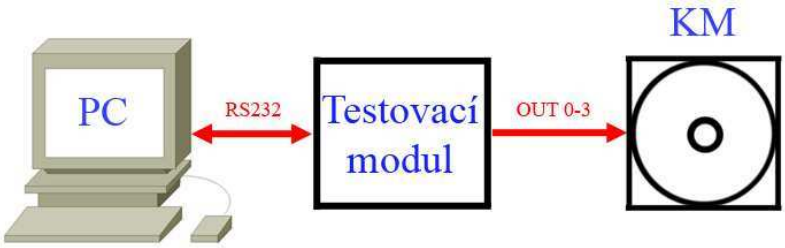
Obr. 2.3 Blokové schéma generátoru frekvence

Mějme situaci, že potřebujeme zjistit, zda-li nám čítač čítá přiváděné impulsy správně. Samozřejmě se to dá zjistit vícenásobným stláčením tlačítka, což je ale nepohodlné. Ano, řekneme si, že pro takový test může postačovat zmáčknout tlačítko pouze dvakrát a můžeme dojít k závěru, že čítač počítá. Co když je ale naprogramován špatně a při naplnění registrů přeteče a přitom neudělá potřebné opatření? V případě osmi-bitového registru by jsme již museli tlačítko zmáčknout minimálně 256 krát. Proto byla navržená funkce ***„BURST“***, která po zadání požadovaného počtů pulsů tyto pulsy vygeneruje.

Výstupem čítače lze vhodným zapojením řídit např. přepínání výstupů multiplexorů. Takové řízení může být v praxi užitečné, proto byly pro testovací modul navrženy funkce,

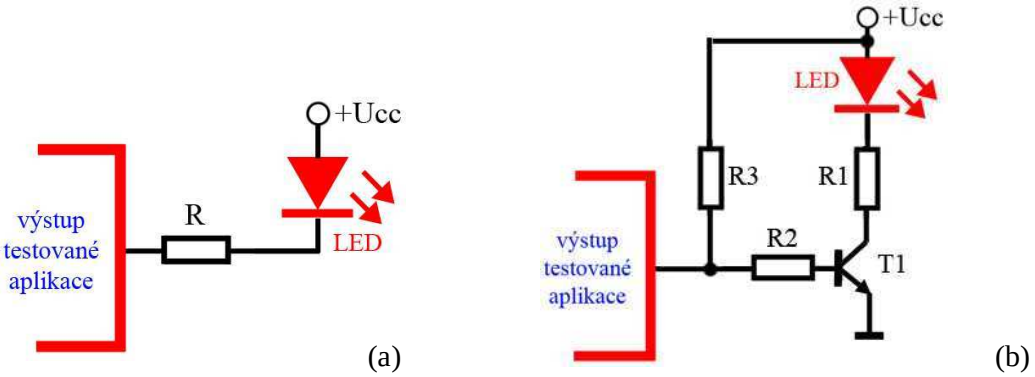
simulující čítání čítače vpřed i vzad, a to funkce ***„INKREMENTACE“*** a funkce ***„DEKREMENTACE“***, kterých hodnota se požadavkem z klávesnice počítače zvýší, případně sníží o jedna.

Dalším úkolem pro studenty je vytvoření přípravku, který by umožňoval dekódování výstupů inkrementálního snímače. Takový snímač se nachází například ve starších myších k počítači. K samotnému zjištění, zda-li je dekodér naprogramován studentem správně je potřeba daný signál přivést na vstupy dekodéru. Funkce generující tento signál je naprogramována v přípravku testovacího modulu a jmenuje se ***„SIMULÁTOR INKREMENTÁLNÍHO SNÍMAČE“,*** ve schématech dále označována jako ***„MOU“***. Podobnou problematikou je řízení krokového motoru. Student navrhne a sestrojí řídící část, kterou následně připojí k samotnému motoru. Může se stát, že se motor nerozeběhne. Chyba je v navrženém řízení, nebo je poškozen motor, nebo výkonová elektronika přivedená na jeho vynutí? Pro testy tohoto typu může být použita funkce testovacího modulu nazvaná ***„ŘÍZENÍ KROKOVÉHO MOTORU“***, u které se z klávesnice počítače ovládá směr rotace motoru v osmitaktním režimu.



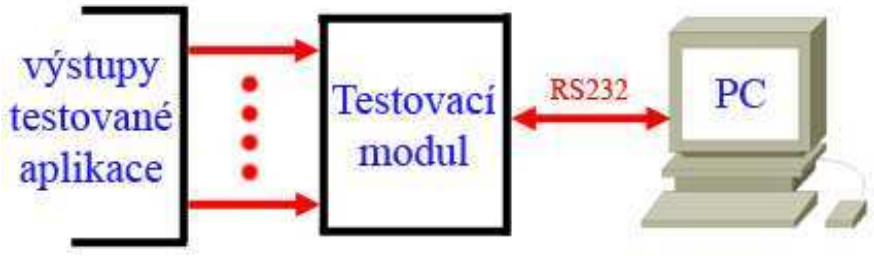
Obr. 2.4 Blokové schéma zapojení pro řízení krokového motoru

Ke zjištění stavu čítače, aktuálního stavu řízení krokového motoru apod., musíme načíst stav daných výstupů. K tomu mohou sloužit připojené LED diody s předřazenými rezistory.



Obr. 2.5 Zapojení LED diody (a) s přímím buzením (b) se spínacím tranzistorem

Když máme měřených výstupů více, nenastává pouze problém s nutností zapojování více LED + rezistorů. Každá součástka, tedy také mikroprocesor, má definovaný maximální zatěžovací proud, z čehož plyne, že zapojení více LED dle obr. 2.5(a) nemusí být bezpečné, proto by se muselo použít zapojení uvedené na obr. 2.5(b). Aby se studenti nemuseli trápit se zapojováním hodně součástek, pro testovací modul byla navržena funkce ***„LOGICKÝ ANALYZÁTOR“***, která snímá logické úrovně na svých vstupech a náměr zobrazí v nadřazeném PC.



Obr. 2.6 Blokové schéma logického analyzátoru

V začátcích programování této práce byla vypracována funkce logického analyzátoru, která vykonávala měření na vstupech pouze rychlostí, danou výpisem řádku obsahujícího náměry do nadřazeného PC (rychlost přepisování těchto hodnot je 30krát za sekundu). Později byla tato funkce vylepšena o možnost rychlého snímání, s ukládáním náměrů do vnitřní paměti mikroprocesoru.

Testovacímu modulu byla naprogramována taky funkce ***„VOLTMETR“***. Často se při zapojování nebo testování různých aplikací, může vyskytnout nutnost zjistit připojované napěťové úrovně a zamezit tak situacím, kdy nesprávné připojení by mohlo z důvodů nedostatečného napájení fungovat nesprávně, nebo naopak větším napětím poškodit zapojené obvody. Tuto funkci lze taky použít například po otestování naprogramovaného řízení pomocí PWM, kdy je potřeba zjišťovat napěťový stav za filtračním členem v závislosti na řízené šířce impulsů. V případě potřeby měření napěťových úrovní měnících se v čase, má testovací modul v sobě naprogramovanou funkci ***„OSCILOSKOP“***.

Obr. 2.7 Blokové schéma měření voltmetrem a osciloskopem

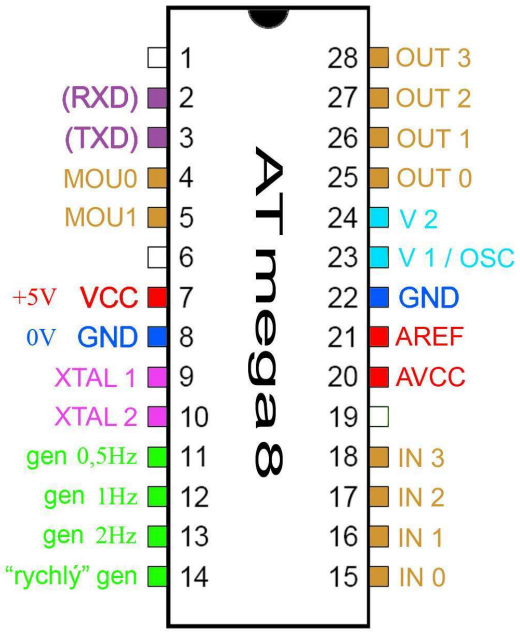
3 Popis přípravku

V průběhu programování „*Testovacího modulu*“ vzniklo postupně více verzí programu. V první verzi se jednalo pouze o jednoduché využití vstupně/výstupních portů pro funkce logického analyzátoru, logického generátoru a generátoru frekvence. Již tahle první verze byla používána na cvičeních předmětu NRM. V dalších verzích programu byly dopracovány funkce voltmetru, rychlého logického generátoru s ukládáním do vnitřní paměti, osciloskopu a funkce pro generaci signálů zadaných obsluhou.

Modul má tyto funkce:

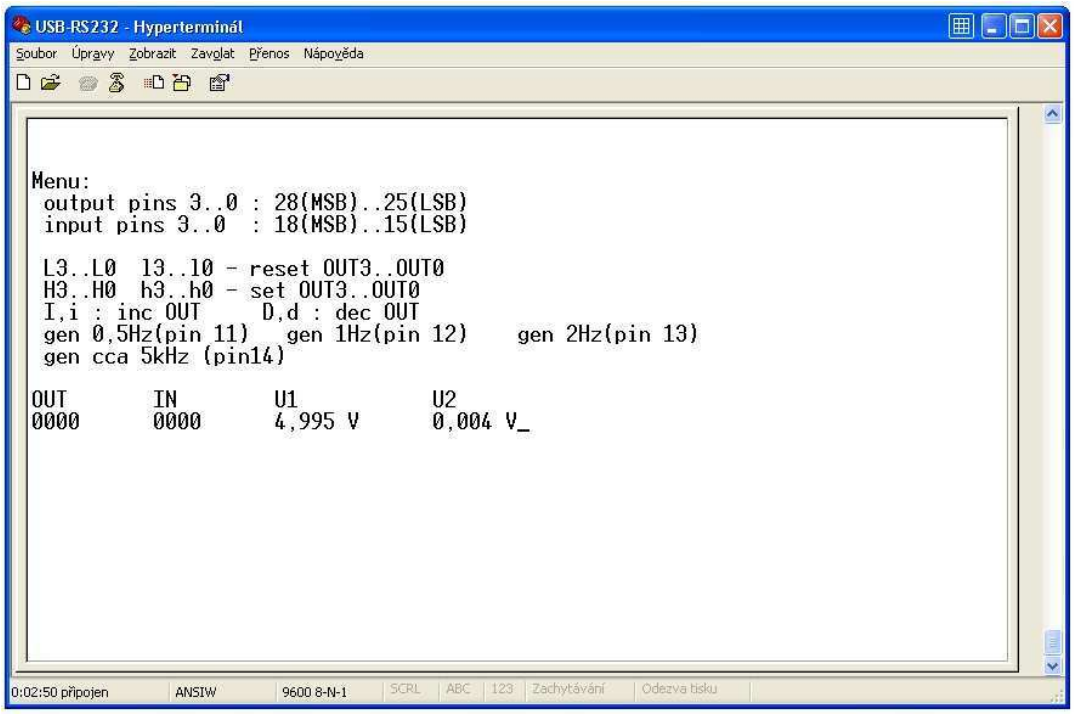
- poskytuje čtyř-bitové statické úrovně logických signálu podle nastavení z PC
- generátor obdélníkového signálu logických úrovní o frekvencích : 0,5Hz, 1Hz, 2Hz a taky „rychlý gen“ (3kHz nebo 17kHz podle verze použitého software)
- jednorázová nebo kontinuální simulace inkrementálního snímače
- generace pulsů pro řízení krokového motoru
- rychlý generátor frekvence (až 4MHz) s možností nastavení generované frekvence a funkcí pro kontinuální generaci, nebo generaci zadaného počtu pulsů (je naprogramován a napálen v samostatném mikroprocesoru)
- 4 bitový logický analyzátor s výpisem náměrů v hyperterminálu
- rychlý logický analyzátor s ukládáním náměrů do vnitřní paměti
- dvoukanálový číslicový voltmetr
- digitální osciloskop

Přípravek je naprogramován a napálen do mikroprocesoru ATmega8. Rozložení použitých vstupů a výstupů je možno vidět na obr. 3.1 .



Obr. 3.1 Popis pinů Testovacího modulu mikroprocesoru ATmega8 (poslední verze)

Výpis hlavního menu a naměřených výsledků a taky komunikaci s uživatelem zabezpečuje program Hyperterminál, který byl vybrán záměrně, protože jej obsahují všechny verze Windows. Tím nejsou kladeny žádné velké podmínky pro použitý počítač a jeho software. Komunikace mezi počítačem a Testovacím modulem probíhá přes sériový port RS232.



Obr. 3.2 Ukázka běhu Testovacího modulu v programu Hyperterminál

4 Software „*TESTOVACÍ MODUL*“

Program „Testovacího modulu“ byl napsán v **ASSEMBLERU AVR STUDIA** a přímo přizpůsoben pro mikroprocesor ATmega8. AVR Studio je software firmy **ATMEL**. Jedná se o freeware, který je možné volně stáhnout ze stránek www.atmel.com

4.1 Metoda programování *Testovacího modulu*

Většina funkcí *Testovacího modulu* byla naprogramována metodou *Stavového programování*. Tato metoda programování se v předmětech MIP a NRM nepoužívá. Princip spočívá v tom, že všechny procesy naprogramované touto metodou musí mít ve svých pro to definovaných proměnných uložené číslo stavu, ve kterém se aktuálně nachází a ze kterého pak přechází na další stav. V praxi to znamená, že program obsahující své podprogramy běží ve smyčce. Každý jednotlivý podprogram se podle své stavové proměnné buď vykoná nebo nikoli. Žádný podprogram se nesmí zacyklit. V průběhu jedné smyčky se prochází všemi podprogramy a tím je zaručená funkce všech podprogramů, které jsou touto metodou naprogramovány. Jako názorná ukázka stylu programování této metody je, když si představíme, že ve vyšším programovacím jazyce neumožníme příkazy zpětných skoků a přerušení. Tím může program běžet ve smyčce pouze jedním směrem.

V prvních verzích *Testovacího modulu* ještě tato metoda nebyla použita. Příjem znaků byl vyřešen pomocí přerušení, kdy mikroprocesor po zjištění příjmu znaku udělal skok v programu na obsluhu přerušení příjmu znaku. Při běžném používání se program jevil jako plně funkční, avšak v případě extrémních podmínek docházelo k neočekávanému běhu programu. Extrémní podmínka se dala nasimulovat například velice rychlým zadáváním znaků z klávesnice počítače. Přerušení mikroprocesoru je výhodné použít v případech, kdy se vstupní podmínka přerušení neočekává a je velice náhodná a málo častá. To není ale případ *Testovacího modulu*, který přímo komunikuje s uživatelem pomocí příkazů z počítače. Metodou využívající přerušení po zjištění příjmu znaku sice mikroprocesor reagoval rychleji na obsluhu přijatého znaku, ale kvůli možnému chybnému běhu programu byla ve finálních verzích programu použitá metoda *stavového programování*.

Z hlediska měření je žádoucí, aby programová smyčka proběhla co nejrychleji a tím byla zaručená co nejvyšší možná vzorkovací frekvence pro měření. Z toho vyplývá, že nesmí

docházet k žádnému zacyklení v některém z podprogramů, ani ke zpětným skokům v programu. Výjimkou jsou podprogramy *osciloskopu* a *rychlého analyzátoru*, které jsou zacykleny. V průběhu hlavní programové smyčky se hlídá, zda-li nebyl zadán požadavek pro běh některé z funkcí a poté se provádí jejich volání. V programu jsou pro potřeby dělení použity externí aritmetické knihovny, které nesplňují bez-cyklový běh podprogramu a za různých vstupních podmínek netrvají stejně dlouhou dobu. Proto musí být na konci hlavní programové smyčky vložen podprogram, který dopočítá běh hlavní programové smyčky vždy na konstantní hodnotu. Tato hodnota musí být větší, než maximální možná hodnota běhu smyčky bez použití podprogramu dopočtu. Touto hodnotou je možné nastavit požadovanou dobu běhu hlavní programové smyčky a tím danou vzorkovací frekvenci. Po dosažení nastavené hodnoty čítače se podprogram dopočtu ukončí, čítač se vynuluje a skočí se na začátek hlavní programové smyčky. Tím je zaručená vždy stejná vzorkovací frekvence.

Čas pro výpis jednoho desetibitového znaku (1x start bit, 8x data, 1x stop bit) přes sériový port do programu Hyperterminálu se vypočte dle vztahu:

$$t_{zn} = \frac{1}{BAUD} \times 10 \qquad (4.1)$$

To znamená, že při nastavené rychlosti 9600Bd (viz kap. 4.2 - nastavení parametrů USART) je čas potřebný pro vyslání jednoho znaku na sériový port $t_{zn} = 10x(1/9600) = 1{,}04ms$. Jeden takt mikroprocesoru pracujícího na hodinovém kmitočtu 8MHz je 125ns. Pro vyslání jednoho znaku je v tomto případě potřeba 8333 taktů mikroprocesoru. Kdyby měl být znak vysílán v zacyklené smyčce, bylo by to z hlediska rychlosti běhu programu neefektivní. Vysílání znaku je v mikroprocesoru řešeno hardwarově a tím je možnost plynulého běhu programu i během vysílání znaku. Vysílání celé věty, obsahující větší počet znaků je vyřešeno *metodou stavového programování*. To znamená, že v jednom průběhu hlavní programové smyčky je vysílán pouze jeden znak. Nato je inkrementována pomocná proměnná. V dalším běhu hlavní programové smyčky je nejdříve otestováno, zda-li již byl vysílaný znak úspěšně odeslán. Když ano, dle pomocné proměnné je zjištěno, který znak byl vysílán naposled, a následně je vyslán znak následující. To se opakuje, až jsou vypsány všechny potřebné znaky. To umožňuje časové nezatížení smyčky. Vyslání všech znaků najednou je použito pouze při vypisování hlavního menu po zapnutí mikroprocesoru, kdy nejsou ještě funkce měření a generování Testovacího modulu aktivní.

Podprogram voltmetru, u kterého je pro získání výsledku naměřeného napětí potřeba vícenásobného dělení, využívá podobně jako u vysílání více znaků toho, že v jednom průběhu

hlavní programové smyčky je uskutečněno pouze jedno dělení. Tím není programová smyčka v době, kdy je funkce voltmetru aktivní extrémně zatěžována. Přípravek obsahuje dvoukanálové měření napětí. Kvůli rychlosti je měření kanálů střídáno, vždy je měřen pouze jediný kanál, měření dalšího kanálu je započato až po dokončení měření kanálu předchozího.

Funkci osciloskopu a funkci rychlého analyzátoru zabezpečují podprogramy, které se spustí za předpokladu splnění všech jejich vstupních podmínek startu. Tyto podprogramy jsou zacykleny a při jejich běhu jsou odpojeny všechny ostatní funkce Testovacího modulu. Zacyklený běh je podmínkou pro získání maximální možné vzorkovací frekvence těchto funkcí. Následná aktivace možností běhu všech funkcí je umožněna až po ukončení podprogramů osciloskopu nebo rychlého analyzátoru.

Další funkcí testovacího modulu je generátor. V hlavní programové smyčce se nacházejí podprogramy, umožňující generaci různých frekvencí a průběhů. Maximální frekvence pro tyto generátory je dána rychlostí běhu hlavní programové smyčky. Proto byl přidělán další program, umožňující generaci frekvencí až 4MHz. Tento program není součástí hlavní programové smyčky Testovacího modulu a je naprogramován v jiném mikroprocesoru.

4.2 Inicializace mikroprocesoru

Samotný software „Testovacího modulu“ začíná inicializací mikroprocesoru. V první řadě je zde potřeba nastavit typ mikroprocesoru, na kterém program poběží. Dále pak nastavit potřebné parametry, nezbytné k správnému běhu programu.

Nastavení „STACK POINTER-u“

Stack pointer je ukazatel vrcholu zásobníku. Mikroprocesor jej využívá pro ukládání návratových adres při skocích do jiných částí programu. Pracuje jako paměť typu LIFO, tj. poslední uložená informace se z paměti vybírá jako první.

Nastavení parametrů „USART“ (Universal Synchronous and Asynchronous serial Receiver and Transmitter) a **povolení vysílače a přijímače**

Tady je potřebné nastavit přenosovou rychlost. V našem případě se jedná o přenosovou rychlost 9600 Bd. V případě potřeby přeprogramování přenosové rychlosti USART-u je nutné změnit hodnotu konstanty „speed“, dle vztahu :

$$speed = \frac{fosc}{16 \cdot BAUD} - 1 \qquad (4.2)$$

kde $\mathbf{f_{osc}}$ = 8.000.000 Hz je taktovací frekvence mikroprocesoru a **BAUD** = 9600 je požadovaná hodnota přenosové rychlosti.

Pro přenosovou rychlost 9600Bd je tedy konstanta speed = 51 .

Nastavení „TIMER“

Mikroprocesor ATmega8 obsahuje 3 čítače-časovače, dva jsou osmi-bitové a jeden je šestnácti-bitový. U čítačů je potřeba nastavit dělící poměry cyklů čítání, vůči taktovací frekvenci mikroprocesoru. K dispozici jsou dělící poměry 8, 64, 256 a 1024.

Dále jsou na začátku programu v sekci inicializace nastaveny konstanty vyskytující se v programu. Změna těchto konstant je nutná při přepisování programu pro jiný typ mikroprocesoru, nebo pro pouhou změnu některé z funkcí. Pak není nutné přepisovat tyto konstanty v celém kódu programu, ale jenom přepsat konstantu v sekci inicializace. Je to velmi výhodné, jelikož délkou se stává program méně přehledný a při přepisování by mohlo snadno dojít k chybě, která by při překladu programu překladačem nebyla vyhodnocena jako chyba programu, protože by se jednalo o chybu logickou. Následkem toho by docházelo k neočekávanému běhu programu, přičemž hledání takové chyby je hodně náročné.

Konstantami jsou nadefinované taky vstupně/výstupní porty. Tím je docíleno, že pro zapojení mikroprocesoru do nějaké aplikace, lze snadno přizpůsobit vstupně/výstupní piny tak, aby to co nejlépe vyhovovalo pro rozložení ostatních součástek a usnadňovalo práci obsluze.

4.3 Hlavní programová smyčka

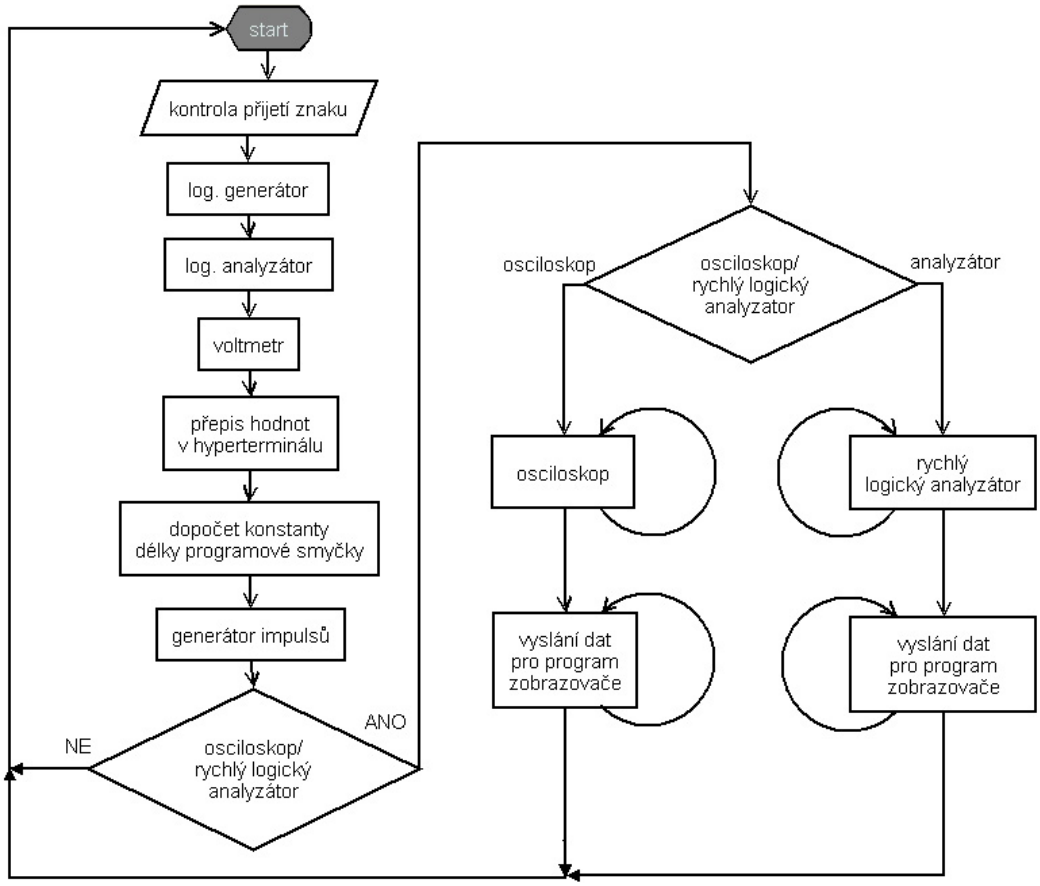
Všechna nastavení provedena v sekci inicializace Testovacího modulu jsou nastaveny až do doby resetování mikroprocesoru. Po skoku do hlavní programové smyčky se totiž z této smyčky již nevyskočí a vykonávají se cyklicky všechny funkce hlavní programové smyčky.

Na začátku hlavní programové smyčky je vynulována proměnná definující přijatý znak z klávesnice počítače a následně se zkoumá, zda-li nebyl přijat nový znak. Když ano, je uložen do proměnné, kterou vyhodnocují podprogramy a na základě ní se rozhodují, zda-li mají vykonat svou činnost, nebo nikoli.

Po zjištění příjmu znaku se volají jednotlivé podprogramy funkcí „rcall“. Funkce „rcall“ zavolá podprogram, který se provede a následně funkcí „ret“ se vrátí pomocí návratové adresy

stack pointeru za místo, ze kterého byl volán. Tento typ programování má velkou výhodu v tom, že samotná hlavní smyčka programu je složena jenom z volání jednotlivých podprogramů, je tedy přehledná. Navíc, v případě potřeby přidání nebo odebrání nějaké funkce, stačí dopsat, popřípadě smazat, volání dalšího podprogramu, aniž by se porušily funkce již naprogramované. Po ukončení volání všech potřebných podprogramů, hlavní smyčka programu skočí na svůj začátek a vše se opakuje. Situaci znázorňuje obr 4.1 .

Časová délka běhu hlavní programové smyčky (bez použití podprogramu dopočtu) je závislá na tom, které a kolik podprogramů jsou aktuálně aktivní. Největší časové zpoždění zapříčiňuje volání 32bitové knihovny pro dělení v podprogramu voltmetru. V tomto případě trvá průběh hlavní smyčkou přibližně 1250 hodinových taktů mikroprocesoru. Současně mohou být aktivní také jiné podprogramy, proto je předimenzován podprogram dopočtu na větší hodnotu, konkrétně 1333 hodinových taktů, což dává frekvenci pro průběh hlavní programové smyčky 6 kHz. Při negaci výstupního pinu „rychlý gen“ po každém průběhu smyčkou, je tedy perioda generované frekvence 3kHz. V případě přidání další funkce běžící v hlavní programové smyčce se zvýší počet hodinových taktů potřebných pro její průběh. V podprogramu dopočtu pak stačí změnit hodnotu čítače, ve které čítání ukončí.



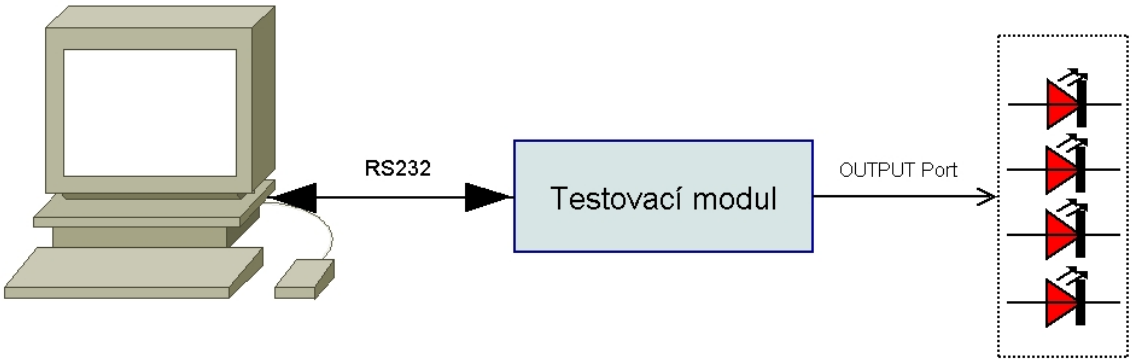
Obr. 4.1 Blokové schéma (vývojový diagram) hlavní programové smyčky

4.4 Podprogramy

4.4.1 Podprogramy pracující se čtyř-bitovými statickými úrovněmi

Podprogramy pro statické čtyř-bitové úrovně používají čtyři bity vstupně/výstupních portů. Tyto bity lze v programu nastavit na kterékoli volné piny mikroprocesoru a kdykoli při přeprogramování měnit pomocí konstant „gen0, gen1, gen2, gen3“ a tím přizpůsobit tyto výstupní bity Testovacího modulu pro nejefektivnější využití v dané aplikaci, pro kterou bude Testovací modul použit. Při změně těchto výstupních bitů v rámci jiného portu je potřeba navíc nastavit, o který jiný port se bude jednat.

Aktuální stav vstupů a výstupů je možné sledovat v okně hyperterminálu, nebo přímo pomocí logické sondy na pinech mikroprocesoru. Samozřejmě je taky možné připojit LED diody a zkoumat stav všech čtyř bitů najednou. K LED diodám je potřeba připojit rezistor takový, aby nedošlo k poškození LED, nebo k poškození mikroprocesoru samotného. Mikroprocesor má výrobcem stanoven maximální možný zatěžovací proud pro jednotlivé vstupně-výstupní porty a taky celkový možný proud tekoucí mikroprocesorem [3]. V případě překročení těchto hodnot může dojít k nevratnému poškození mikroprocesoru. Výhodnější z hlediska jednoduššího zapojení je sledovat vstupy a výstupy pomocí okna hyperterminálu.



Obr. 4.2 Blokové schéma zapojení výstupního portu

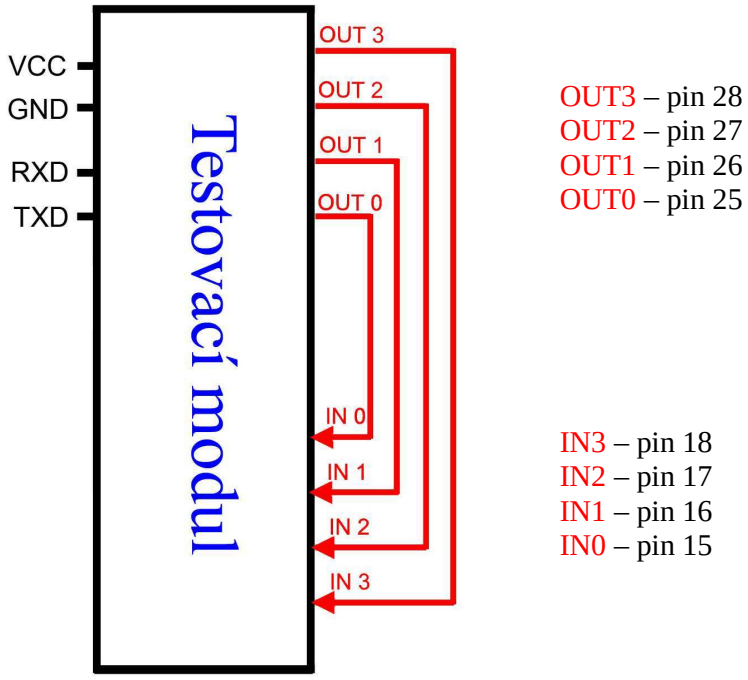
Přednastavená hodnota čtyř-bitového výstupu OUT0 – OUT3 je „0000“. Tato hodnota je nastavena po zapnutí nebo po resetování mikroprocesoru. Před samotnou operací některých z funkcí pro práci se 4-bitovými statickými úrovněmi musí dojít k získání aktuálních hodnot čtyř-bitového výstupu. Testovací modul tuto informaci získává nasnímáním hodnot, které si mikroprocesor přečte rovnou ze vstupně/výstupních portů. Je taky možnost, aby byla tato

informace přečtena přímo z registrů vstupně/výstupních portů. Při správné funkci Testovacího modulu a správném zapojení je z hlediska pozorovaných výsledků jedno, kterou metodou je získání informace o stavu výstupů uděláno. K rozdílným výsledkům při snímání výstupů by ale mohlo náhle dojít v případě zkratu na některém z výstupních pinů mikroprocesoru při metodě snímání vstupně/výstupních registrů. Naprogramováním Testovacího modulu výše uvedenou metodou dokáže obsluha situaci zkratu na výstupním pinu odhalit, popřípadě zjistit nefunkčnost kteréhokoli ze zobrazovaných výstupních pinů mikroprocesoru.

Funkce podprogramů pracujících se čtyř-bitovými statickými úrovněmi se o svoje výstupní piny mikroprocesoru OUT0-OUT3 dělí s funkcí „řízení krokového motoru“. V případě běhu této funkce jsou funkce podprogramů pracujících se čtyř-bitovými statickými úrovněmi zablokovány. Vysvětlení proč tomu tak musí být je vysvětleno v čl. 4.4.3.2 .

Výstup funkcí inkrementace a dekrementace se chová podobně jako jednoduchý čtyř-bitový čítač s čítáním vpřed nebo vzad. Tím je možné tyto funkce využít pro simulaci čtyř-bitového čítače. Taky je možné využití výstupů těchto funkcí například k řízení multiplexorů a demultiplexorů.

Otestovat funkce ovlivňující výstupní piny mikroprocesoru lze přímo jednoduchým zapojením těchto pinů k vstupním pinům logického analyzátoru (popis logického analyzátoru viz čl. 4.4.2). Zapojení je možno vidět na obr. 4.3 .



Obr. 4.3 Jednoduchý test funkcí pracujících s výstupními piny mikroprocesoru

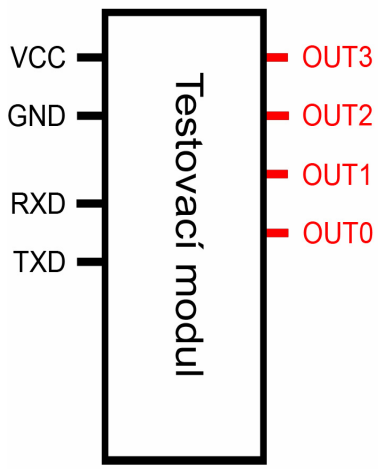
4.4.1.1 Inkrementace

Po volání „rcall *increment*“ se nejdříve zjistí, zda-li je provádění funkce inkrementace možné, což znamená, jestli nejsou zrovna obsazeny výstupy OUT0-OUT3 pro funkci řízení krokového motoru. V případě, že řízení krokového motoru je zablokováno se zjistí, zda-li přijatý znak z klávesnice vyhovuje podmínce pro běh podprogramu inkrementace. Konkrétně se jedná o znaky „i“ a „I“. V případě splnění příjmu těchto znaků se prozkoumá aktuální stav I_n na výstupním portu určeném pro tuto funkci a následně se provede inkrementace daného stavu. V opačném případě se z podprogramu inkrementace vyskočí.

Jelikož funkce inkrementace používá pro svůj výstup čtyři bity, v případě stavu 1111 se výstup neinkrementuje klasicky osmi-bitově, ale nastaví se na něm hodnota 0000, která je následující po přetečení čtyřbitového čísla 1111.

$$I_{n+1} = I_n + 1$$

I_n	OUT3	OUT2	OUT1	OUT0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

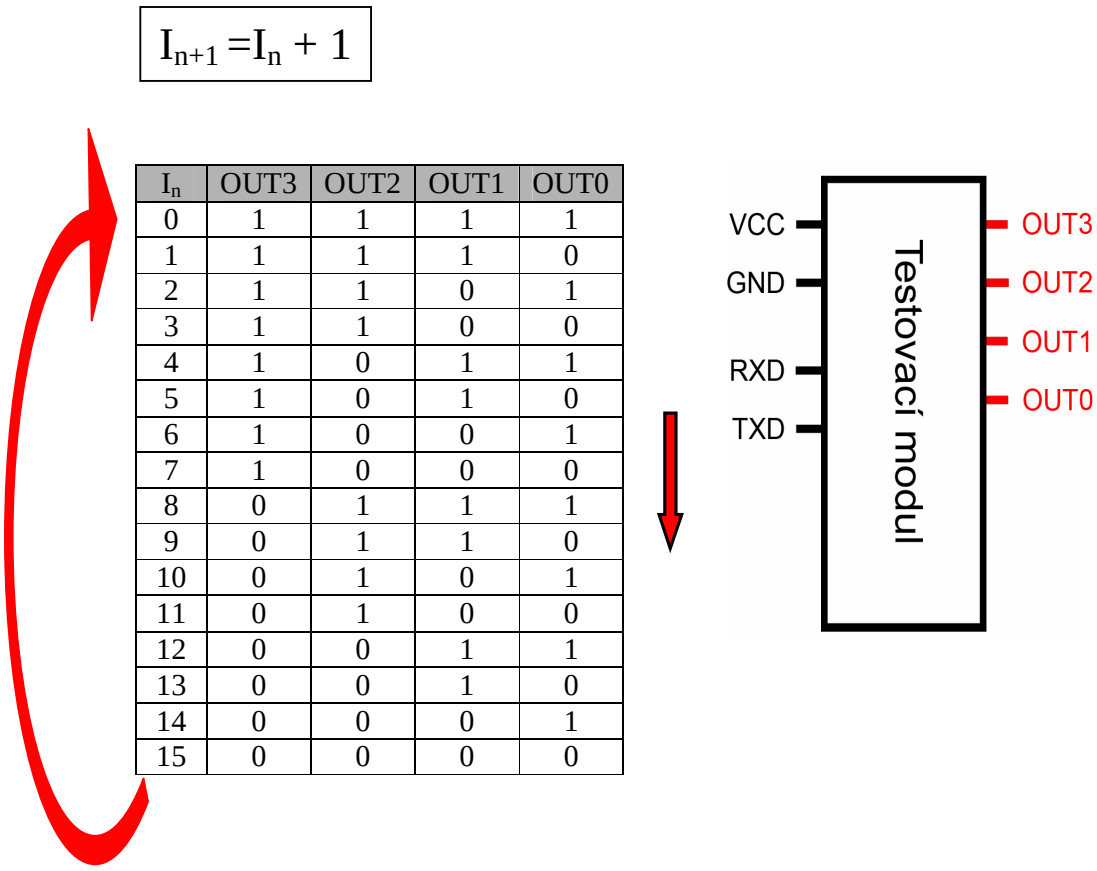


Obr. 4.4 Tabulka stavů výstupních pinů mikroprocesoru při inkrementaci

4.4.1.2 Dekrementace

Stejně jako u podprogramu inkrementace se nejdříve zjistí stav zapnutí řízení krokového motoru a podle toho se z podprogramu dekrementace buďto vyskočí, nebo běží dál. Na rozdíl od podprogramu inkrementace se po volání „rcall *decrement*“ ověří vstupní podmínka příjmu znaků „d“ a „D“. Při nesplnění této podmínky se z podprogramu dekrementace vyskočí.

Podprogram dekrementace pracuje stejně jako podprogram inkrementace s rozdílem, že se nejedná o inkrementaci aktuálního stavu na výstupním portu, ale o jeho dekrementaci. V případě zjištění hodnoty 0000 se jako následující hodnota po dekrementaci nastaví hodnota 1111.



I_n	OUT3	OUT2	OUT1	OUT0
0	1	1	1	1
1	1	1	1	0
2	1	1	0	1
3	1	1	0	0
4	1	0	1	1
5	1	0	1	0
6	1	0	0	1
7	1	0	0	0
8	0	1	1	1
9	0	1	1	0
10	0	1	0	1
11	0	1	0	0
12	0	0	1	1
13	0	0	1	0
14	0	0	0	1
15	0	0	0	0

Obr. 4.5 Tabulka stavů výstupních pinů mikroprocesoru při dekrementaci

4.4.1.3 Nastavení logického stavu

Je možné na kterémkoli ze čtyř-bitových výstupů nastavit požadovanou logickou úroveň. Uživatel si nejdříve zvolí, zda-li bude nastavovat úroveň logické nuly (klávesou „l“,“L“), nebo úroveň logické jedničky (klávesou „h“,“H“). Poté číslicí 0 až 3 zvolí požadovaný bit, který se následně překlopí do požadované logické úrovně. Při nastavování stejné logické úrovně na vícero bitech, není nutno pokaždé zadávat logickou úroveň. Mikroprocesor si pamatuje posledně zadanou, stačí pak zadávat hodnoty jednotlivých bitů (číslicemi 0 až 3). Ukázku možného zadávání příkazů pro funkci nastavení logického stavu ukazuje tabulka na obr.4.7. Po vykonání změny se podprogram nastavení logického stavu ukončí a vrátí se zpět za místo, odkud byl volán.

Po zapnutí nebo po resetování mikroprocesoru je příznak pro zadávání stavu logické úrovně přednastaven pro nastavení hodnoty logické nuly. Taky u této funkce je pro její vykonání nutné, aby zrovna nebyla aktivní funkce řízení krokového motoru.

Klávesy pro nastavení logického stavu

„L“ nastaví příznak pro zadávání stavu logické nuly (přednastavená hodnota po zapnutí)

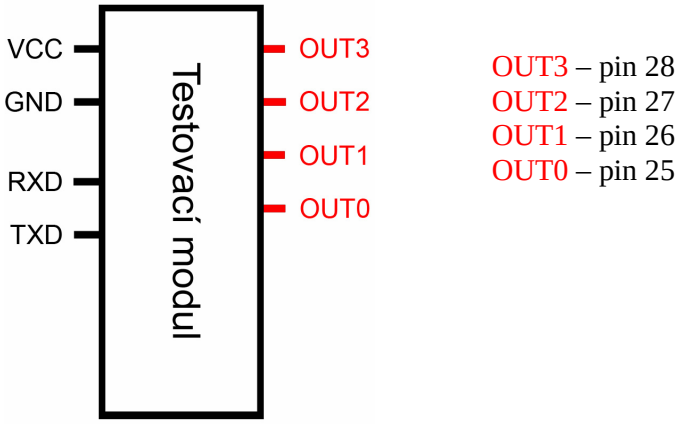
„H“ nastaví příznak pro zadávání stavu logické jedničky

„0“ na pinu OUT0 se nastaví posledně zadaný příznak logické úrovně

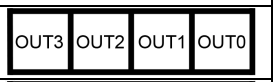
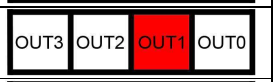
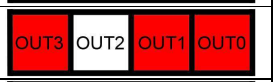
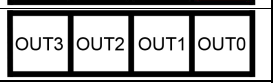
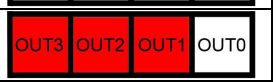
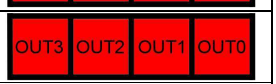
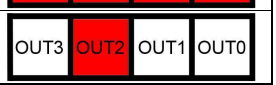
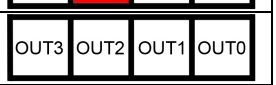
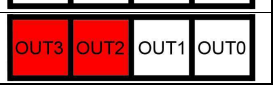
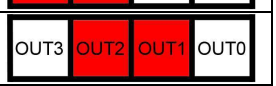
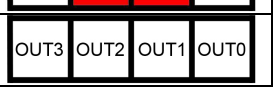
„1“ na pinu OUT1 se nastaví posledně zadaný příznak logické úrovně

„2“ na pinu OUT2 se nastaví posledně zadaný příznak logické úrovně

„3“ na pinu OUT3 se nastaví posledně zadaný příznak logické úrovně



Obr. 4.6 Označení výstupních pinů Testovacího modulu

Zadaný příkaz z klávesnice	Aktuální stav	
		- počáteční stav
H 1		- nastaven bit 1
H 3 0		- k bitu 1 nastaven taky bit 0 a 3
L 1 3		- vynulovány bity 1 a 3
0		- vynulován bit 0
H		- nastavení příznaku logické 1
1 3 2		- nastaveny bity 1,2,3 (na pořadí nezáleží)
0		- nastaven bit 0
L		- nastavení příznaku logické 0
0 3 1		- nulování bitů 0,1,3
3 0 2 1		- nulování bitů 0,1,2,3 (nulování nulového bitu je možné)
L H 2 3		- důležité je, který příznak je nastaven jako poslední, v tomto případě log.1
L 3 H 1		- nulování bitu 3 a nastavení bitu 1
L 0 1 5 3 4 2		- nulování bitů 0,1,2,3 (zadané čísla neoznačující žádný bit jsou ignorovány)

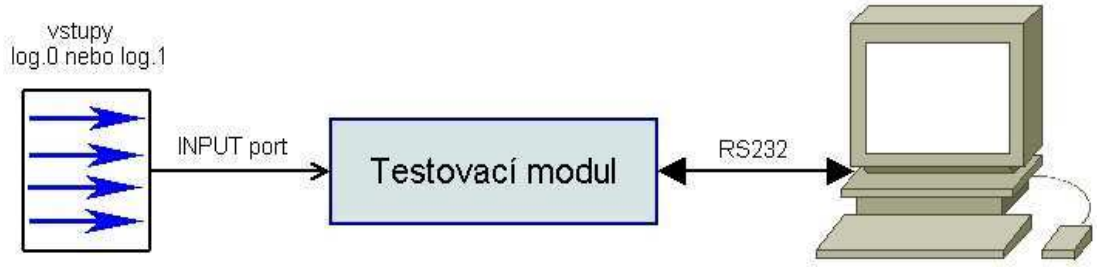
 - znázorňuje hodnotu log.0 na výstupu

 - znázorňuje hodnotu log.1 na výstupu

Obr. 4.7 Tabulka znázorňující možné zadávání příkazů pro funkci nastavení logického stavu

4.4.2 Logický analyzátor

Možnost měřit logické úrovně čtyř-bitově a následně je zobrazit v programu Hyperterminálu umožňuje funkce *logický analyzátor*. Ta provádí jednoduché snímání logických hodnot na čtyřech vstupních pinech mikroprocesoru „**IN0 - IN3**“. Tyto čtyři bity lze naprogramovat, popřípadě přeprogramovat libovolně na kterékoli volné piny mikroprocesoru dle požadavků obsluhy. V programu Testovacího modulu jsou naprogramované vstupy logického analyzátoru označené konstantami „ana0, ana1, ana2, ana3“. Při přeprogramování těchto vstupních bitů v rámci jiného portu je potřeba navíc nastavit, o který jiný port se bude jednat.



Obr. 4.8 Blokové schéma zapojení vstupního portu

Naměřené hodnoty logického analyzátoru jsou vypisovány v řádku aktuálních hodnot v programu Hyperterminálu. Výpis naměřeného řádku se děje přibližně 30krát za vteřinu. Bylo by proto neefektivní z hlediska pomalého vypisování naměřených hodnot, měřit tyto hodnoty rychleji. Proto je vzorkovací frekvence logického analyzátoru dána rychlostí přepisování řádku aktuálních hodnot do programu Hyperterminál. K nasnímání nových hodnot vstupů totiž dochází jednou za výpis řádku aktuálních naměřených hodnot. Taková snímací rychlost je pro mnohé aplikace dostačující. V případě potřeby vyšší rychlosti snímání, se musí použít funkce „rychlého logického analyzátoru“, který má samostatně běžící smyčku, podobně jako má samostatně běžící smyčku funkce osciloskopu.

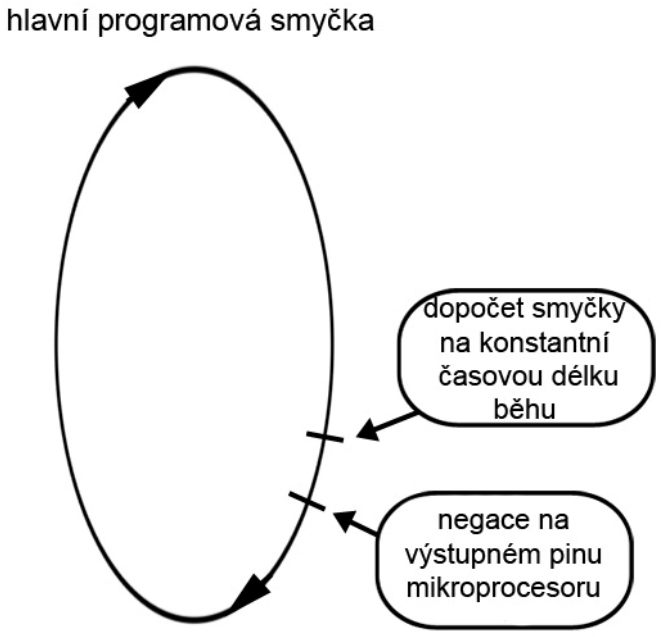
Mikroprocesory použité pro přípravek Testovacího modulu využívají technologii CMOS. Z hlediska snímání vstupů je důležité, aby byl snímaný vstup připojen na některou z logických úrovní. Při nezapojení některé z logických úrovní na pinu nakonfigurovaném jako vstupní totiž dochází k náhodné a neočekávané měřené vstupní hodnotě, způsobené vnějším šumem z prostředí. Pouhé mávnutí rukou může způsobit překlopení logického stavu. Je proto důležité při programování ošetřit vstupní piny mikroprocesoru vnitřními pull-up rezistory. Tím jsou vstupní piny přes rezistory připojeny k napájecímu napětí a jsou překlopeny do

stavu logické jedničky i při nepřipojeném vstupu. Při testování byl použit hardware, kterým byly měřené vstupy logického analyzátoru připojeny k přepínačům, které dodávaly na svých výstupech hodnotu logické nuly nebo logické jedničky. Tím byla závada způsobená nepřipojenými pull-up rezistory objevena až při používaní jedné z prvních verzí na cvičeních předmětu NRM a následně poté opravena.

Pro běh logického analyzátoru není potřeba ze strany obsluhy zadávat žádné příkazy z klávesnice. Po zapnutí mikroprocesoru se po výpisu menu a následně po skočení do hlavní programové smyčky započne měření logickým analyzátorem automaticky. Funkci logického analyzátoru nelze za běhu programu zrušit, neaktivní se dočasně stane pouze při skoku do podprogramů funkcí osciloskopu nebo rychlého logického analyzátoru.

4.4.3 Generátor pulsů

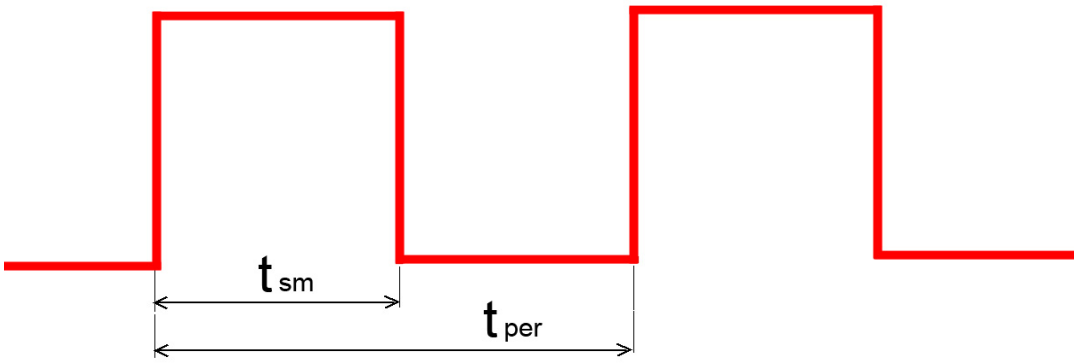
Hlavní programová smyčka obsahuje ve svém konci dopočet pro konstantní časovou délku svého běhu. Když po této funkci budeme pokaždé negovat výstup jednoho bitu na některém portu mikroprocesoru, získáme generátor impulsů s maximální možnou frekvenci pro danou smyčku. Ten je vyveden na výstupním pinu mikroprocesoru a je možné ho využít pro aplikace vyžadující rychlé vstupní pulsy.



Obr. 4.9 Ukázka principu funkce jednoduchého generování pulsů

Když dále pomocí čítače vezmeme jenom každý n-tý puls generátoru, vznikne nám velice jednoduchým způsobem generátor pulsů s n-násobně menším kmitočtem vůči hlavnímu kmitočtu. Ten lze nastavit a zabudovat natvrdo na stálou hodnotu, nebo umožnit obsluze pomocí kláves počítače zvolit požadovanou hodnotu, kterou bude následně mikroprocesor generovat na některém ze svých výstupů. Podmínkou je, že se musí jednat o celočíselný poměr k nejvyšší generované frekvenci. Je možnost přímo přednastavit několik hodnot a přiřadit jim klávesové zkratky, které by bylo možné najít po výpisu „help“, nebo požádat obsluhu o zadání číselné hodnoty pro následující generovanou frekvenci.

Generovanou frekvenci je možné použít jako hodinový signál pro různé testované aplikace. Generovaný signál je možné vidět na následujícím obrázku, kde je taky vidět časové změny hran závislých na běhu hlavní programové smyčky.



Obr. 4.10 Průběh generovaného signálu Testovacím modulem

t_{sm} čas jednoho průběhu hlavní programové smyčky

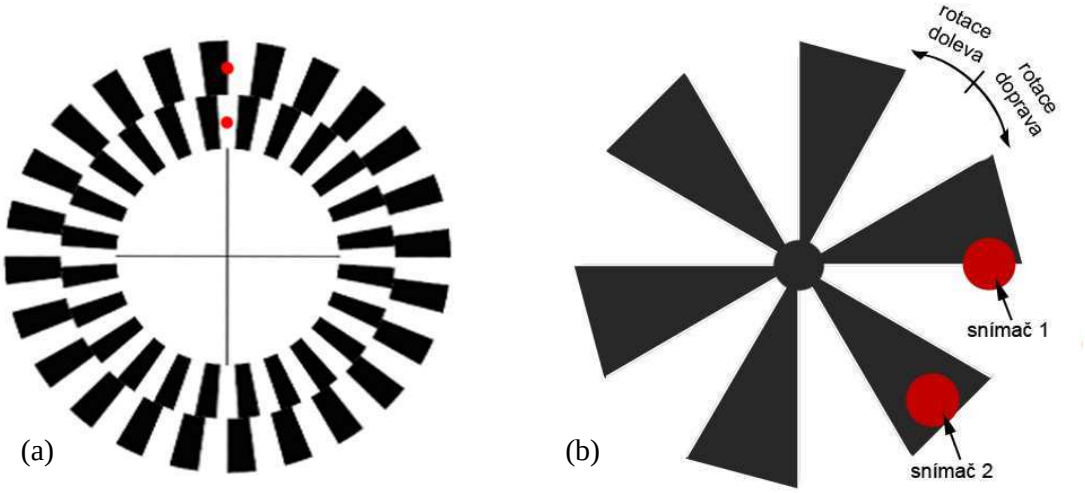
t_{per} čas jedné periody generovaného signálu = 2 průběhy hlavní programové smyčky

Některé aplikace, nebo požadovaná měření, vyžadují často mnohem složitější generované signály, než jen pouhý hodinový kmitočet. Testovací modul v sobě obsahuje funkce pro generaci speciálních signálů, které jsou popsány v následujících kapitolách.

4.4.3.1 Simulace „inkrementálního snímače“

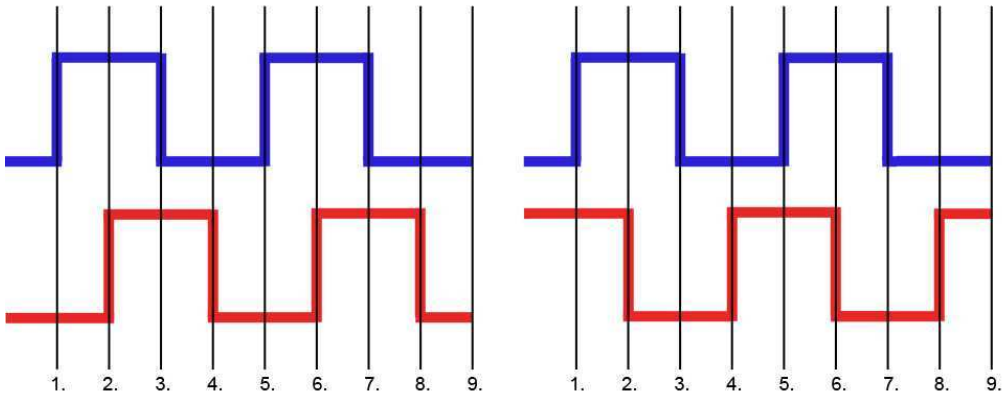
Při měření otáček, když po obvodu hřídele je vyznačen daný počet světlých a tmavých míst pro vyhodnocení optickým snímačem, je velice snadné zjistit, kolikrát se hřídel otočí za předpokladu otáčení stále jedním směrem. Simulování tohoto případu je velice snadné, v tomto případě stačí pouhý generátor hodinových signálů.

V případě, že dochází ke změnám otáčení, je potřeba vhodným způsobem zajistit možnost detekce těchto změn. K možné ukázce této problematiky stačí použití dvou, od sebe o polovinu štěrbiny pootočených snímacích ploch, přičemž obě snímací plochy vyhodnocují optické snímače, umístěny v jedné rovině (viz obr. 4.11a). Přesná konstrukce takového přípravku by ale v praxi byla velice obtížná, proto se používá pouze jedna snímací plocha, kterou snímají dva optické snímače posunuty vůči sobě. Situaci znázorňuje obr. 4.11b .



(a) (b)

Obr. 4.11 Ukázky přípravků pro funkci inkrementálního snímače



(a) (b)

Obr. 4.12 Signály inkrementálního snímače (a) rotace doprava (b) rotace doleva

Po zapnutí, nebo po resetu „Testovacího modulu“, je funkce simulátoru inkrementálního snímače neaktivní a jeho výstupy jsou ve stavu logické nuly. Do chodu jej obsluha uvede až zvolením požadovaného směru otáčení. Pro oba směry otáčení má obsluha na výběr dva režimy. Režim „single mode“ vykoná po jednom stisku klávesy „q“ nebo „w“ překlopení výstupu do následujícího stavu. Tento stav zůstane na daném výstupu až do doby, dokud obsluha neaktivuje následující stav, neukončí generování klávesou „e“ nebo „E“, nebo nedojde k resetu mikroprocesoru. Každým dalším stiskem klávesy „q“ nebo „w“ se provede překlopení výstupu do následujícího stavu. Druhý režim umožňuje kontinuální běh simulátoru inkrementálního snímače, který se aktivuje po stisku klávesy „Q“ nebo „W“. Kdykoli je možná změna směru kontinuálního běhu. Simulátor inkrementálního snímače nacházející se v režimu kontinuálního běhu generuje ve svých výstupech pulsy až do doby, dokud jej obsluha nezastaví ukončením generování, a to klávesou „x“ nebo „X“, změnou na „single mode“, nebo pokud nedojde k resetu mikroprocesoru. Rychlost změny na následující stav v režimu kontinuálního běhu je dána rychlostí běhu hlavní programové smyčky.

Výstupy funkce simulujícího inkrementální snímač jsou vyvedeny na pinech mikroprocesoru „4, 5“ s označením **„MOU0, MOU1“**.

Klávesy pro ovládání generátoru simulujícího inkrementální snímač :

„**q**“ – rotace doleva jednorázová - „single mode“

„**Q**“ – rotace doleva kontinuální - „continual mode“

„**w**“ – rotace doprava jednorázová - „single mode“

„**W**“ – rotace doprava kontinuální - „continual mode“

„**e**“, „**E**“ – ukončení generování a vynulování výstupů

Funkci simulátoru inkrementálního snímače mohou studenti využít pro „umělé zavedení rotace“ při testování své aplikace, která má dekódovat signály skutečného inkrementálního snímače.

4.4.3.2 Řízení krokového motoru

Mezi specielní generované signály, které poskytují funkce testovacího modulu, patří i signály pro řízení rotace krokového motoru. Konkrétně se jedná o řízení krokového motoru v osmitaktním režimu. Pro výstupní signály této funkce jsou použity výstupy „**OUT0-OUT3“**, které jsou sdíleny společně s funkcemi pracujícími se čtyř-bitovými statickými úrovněmi.

Při řízení krokového motoru jsou přesně definovány signály (viz obr. 4.13), které se přivádí na výkonovou část krokového motoru (viz obr. 4.14). V případě nedodržení posloupností těchto signálů by docházelo k špatnému řízení motoru. Proto je potřeba softwarově oddělit řízení krokového motoru od funkcí pracujících se čtyř-bitovými statickými úrovněmi, aby nenarušovaly plynulé řízení krokového motoru. Kdyby tomu tak nebylo, obsluha by kromě ovládání motoru pro to určenými klávesami, mohla taky klávesami pro změnu statických úrovní na výstupu úplně změnit stav pulsů pro krokový motor. Využití tohoto opatření v případě běhu krokového motoru znemožňuje běh ostatních funkcí pracujících se stejnými výstupy, které jsou využívány pro řízení krokového motoru a naopak.

Po zapnutí nebo restartu mikroprocesoru je hodnota pro rozhodování přidělení výstupních pinů OUT0-OUT3 přednastavena pro použití funkcí pracujících se čtyř-bitovými statickými úrovněmi. Ty se stlačením klávesy „u“ nebo „U“ deaktivují, stavová proměnná se prohodí ve prospěch řízení krokového motoru a na výstupech OUT0-OUT3 se nastaví hodnota odpovídající prvnímu stavu z osmistavové množiny řídících pulsů. Poté je možné klávesami „p“ nebo „P“ a „o“ nebo „O“ ovládat směr otáčení krokového motoru doprava nebo doleva. Při deaktivaci řízení krokového motoru stlačením klávesy „z“ nebo „Z“ se výstupy vynulují a umožní se běh funkcí pracujících se čtyř-bitovými statickými úrovněmi.

Klávesy určené k řízení krokového motoru :

„p“, „P“ - rotace krokového motoru doprava

„o“, „O“ - rotace krokového motoru doleva

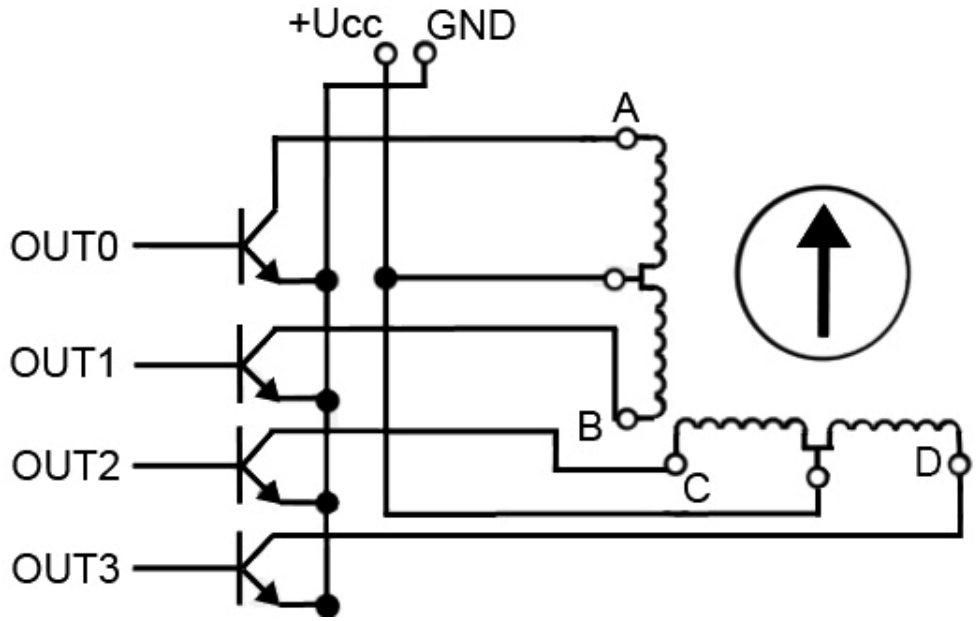
„u“, „U“ - uvolnění výstupních pinů krokovému motoru a jeho aktivace

„z“, „Z“ - deaktivace krokového motoru

krok	OUT0	OUT1	OUT2	OUT3
0	1	0	0	0
1	1	0	1	0
2	0	0	1	0
3	0	1	1	0
4	0	1	0	0
5	0	1	0	1
6	0	0	0	1
7	1	0	0	1

Obr. 4.13 Stavy výstupů při osmitaktním řízení krokového motoru

Výstupy mikroprocesoru nelze přímo připojit na vinutí krokového motoru, protože na to není výkonově stavěný. Je potřeba použít výkonové členy, například tranzistory, na jejichž vstupy se přivedou signály z mikroprocesoru. Možné zapojení krokového motoru ukazuje následující obrázek.



Obr. 4.14 Zapojení vynutí krokového motoru

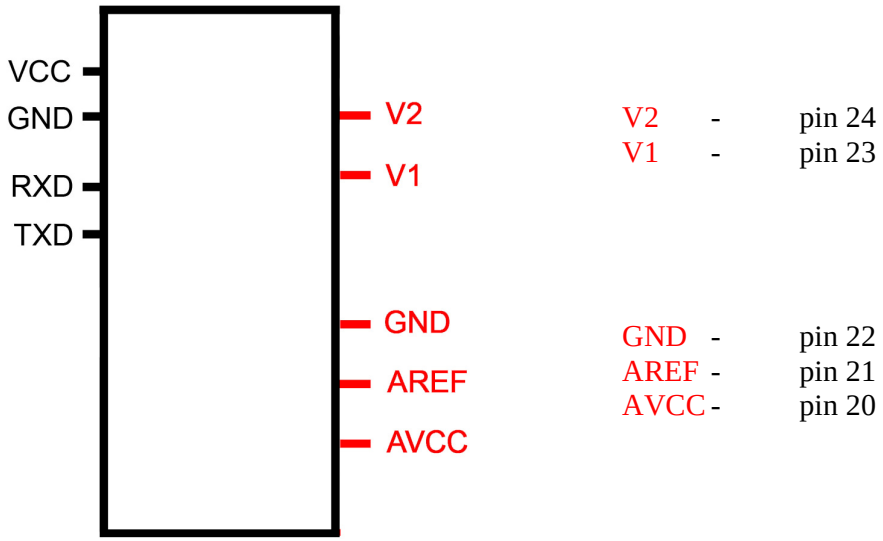
4.4.4 Voltmetr

Mikroprocesor ATmega8 má v sobě zabudovaný analogově digitální převodník (dále jen AD převodník). Jedná se o deseti-bitový AD převodník, pracující metodou postupné aproximace. AD převodník je hardwarově připojen na analogový multiplexor, který je u mikroprocesoru ATmega8 šesti-vstupový. U AD převodníku je možnost využít zabudovanou vnitřní napěťovou referenci, nebo připojit externí napěťovou referenci na pro to vyhrazených pinech mikroprocesoru. Přípravek Testovacího modulu využívá ve funkci voltmetru externí napěťovou referenci, které hodnota pro získání správného výsledku musí být **+5V**. Tu je nezbytné připojit na pin „AREF“ mikroprocesoru (viz obr. 4.15). Výsledek AD převodu (**AD_{vysl}**) se vypočte podle vztahu :

$$AD_{vysl} = 1024 \times \frac{U_{VST}}{U_{REF}} \qquad (4.3)$$

kde „**U_{VST}**“ je vstupní měřené napětí a „**U_{REF}**“ je referenční napětí přivedeno na pinu AREF.

V Testovacím modulu jsou pro měření napětí použity dva kanály. Měřené napětí se přivádí na vstupy „**V1**“ a „**V2**“. V případě potřeby je možné Testovací modul doplnit o více než dvoukanálové měření napětí, ale nastává problém s nedostatkem volných pinů mikroprocesoru. Čtyři ze šesti pinů možných pro zapojení vstupů AD převodníku, jsou totiž obsazeny funkcemi pracujícími se čtyř-bitovými statickými úrovněmi.



Obr. 4.15 Označení pinů potřebných pro správný běh voltmetru

Rychlost měření napětí je dána rychlostí AD převodníku. Ten pro dosažení maximální přesnosti, potřebuje hodinové pulsy mikroprocesoru pomocí vlastní předděličky AD převodníku snížit pro svoje účely na méně než 200 kHz. Převod AD převodníku trvá 13 hodinových cyklů. (Poznámka : při prvém zapnutí AD převodníku udělá mikroprocesor tzv. rozšířený převod, který je nutný pro inicializaci a minimalizování chyby offsetu. Rozšířený převod trvá 25 hodinových cyklů). Rychlost měření při maximální přesnosti je tedy přibližně 15 000 vzorků za sekundu. Funkce voltmetru ale kvůli potřebě vypisování naměřeného výsledku v terminálu rychlostí 30krát za sekundu tak rychlé měření nevyužívá.

Aby použití vícekanálového měření napětí časově moc nezatěžovalo rychlost průběhu hlavní smyčky, je v rámci běhu jedné smyčky AD převodníkem zpracováván pouze jeden kanál voltmetru. V běhu programu se kanály voltmetru střídají. Přepínání kanálu je zabezpečeno softwarově, kdy měření druhého kanálu je započato až po úspěšném odměření a uložení prvního kanálu. Multiplexor AD převodníku je přepínán, kdy za pomocí na to určené proměnné si pamatuje, který kanál byl měřen naposled a v dalším cyklu se posune.

Po odměření všech, v našem případě dvou kanálů, se až do doby vypsání hodnot v programu Hyperterminál zablokuje funkce voltmetru. Je to velice potřebné opatření. Mějme pro příklad případ, kdy voltmetr naměří „4,000“ V. Při vypisování hodnot se nejdříve vypíše číslo „4“. Při nezablokování měření voltmetru se udělá nový náměr např. „3,995“ V. V následujícím běhu smyčky by se v tomhle případě nevypsala „0“ ale „9“. To by znamenalo, že místo vypsání hodnoty „4,0xx“ V by se vypsala hodnota „4,9xx“ V.

Desetibitové měření AD převodníku, tj. hodnoty v desítkové soustavě od 0 do 1023, se musejí přepočíst dle konstanty, která je určena z referenčního napětí na číslo, které bude prezentováno ve výpisu jako výsledek měření ve voltech. Výsledné napětí se určí dle vztahu :

$$U_{vysl} = U_{REF} \times \frac{AD_{vysl}}{1024} \qquad (4.4)$$

Pro potřebné dělení program Testovacího modulu využívá dvě externí knihovny, z nichž jedna umožňuje šestnácti-bitové a druhá třicetidvou-bitové dělení. Tyto podprogramy knihoven jsou hodně zacyklené a netrvají vždy stejně dlouhou dobu. Doba jednoho dělení se u šestnácti-bitové knihovny pohybuje od 235 do 251 cyklů a u třicetidvou-bitové knihovny od 528 do 688 cyklů. Samotný výsledek naměřeného napětí se vypočte následovně:

1. deseti-bitový náměr AD převodníku se vynásobí hodnotou 5000 (v případě maximálně naměřené hodnoty AD převodníku, tj. hodnoty 1023, se získá hodnota 1023x5000=5115000, která je po přepočtu do šestnáctkové soustavy rovná 4E.0C.78)
2. výsledek kroku 1. se vydělí hodnotou 1023 - v tomto kroku je potřeba využití třicetidvou-bitové knihovny pro dělení (jinde v programu již použitá není)
3. výsledek kroku 2. se vydělí hodnotou 10 a zbytek po dělení se uloží do paměti jako „TISÍCINA“
4. výsledek kroku 3. se vydělí hodnotou 10 a zbytek po dělení se uloží do paměti jako „SETINA“
5. výsledek kroku 4. se vydělí hodnotou 10, výsledek po dělení se uloží do paměti jako „VÝSLEDEK“ a zbytek po dělení jako „DESETINA“

Před samotným zobrazením výsledku změřeného napětí, vypočteném ve výše zmíněných krocích, je nutno vypočtené čísla „0“-„9“ transformovat na hodnoty znaků „0“-„9“ v ASCII. Mezi hodnotami „výsledek“ a „desetina, setina, tisícina“ se musí vložit znak desetinné čárky.

Podprogram voltmetru je z celého software Testovacího modulu z hlediska potřebného času ke zpracování výsledků nejvíce náročný. Aby časově nezatěžoval hlavní programovou smyčku celou dobou svého trvání, je softwarově ošetřen tak, aby v průběhu jedné hlavní programové smyčky vykonával pouze jeden z pěti výše označených kroků. Tento problém je vyřešen inkrementací pomocné proměnné zabezpečující postupné přepínání těchto kroků, kdy každý krok hlídá aktuální stav této proměnné a v případě rovnosti s ní se provede.

Po ukončení odměru a přepočtu náměru, na výsledek reprezentující naměřené napětí ve voltech, se tento výsledek uloží to paměti pod hodnotou výsledku aktuálně běžícího kanálu voltmetru. Úplně stejně se provede měření, přepočet a uložení výsledku druhého kanálu voltmetru. Tyto naměřené hodnoty jsou v době výpisu řádku naměřených hodnot vysílány na sériový port.

Při využití deseti-bitového rozlišení AD převodníku a referenčního napětí +5V je chyba měření přibližně 5mV. Přesnost samotného náměru je závislá na přesností referenčního napětí, kterého hodnota může být negativně ovlivňována vnějším rušením. K mikroprocesoru je proto potřeba připojovat blokovací kondenzátory!

4.4.5 Osciloskop

Funkci osciloskopu zabezpečuje podprogram, který je zacyklen. Z toho vyplívá, že ostatní funkce jsou po dobu běhu osciloskopu neaktivní. Aby došlo k samotnému spuštění podprogramu osciloskopu, je potřeba splnit dvě podmínky. První podmínkou je nastavení stavové proměnné pro rozhodování mezi aktivací funkce osciloskopu, nebo rychlého logického analyzátoru, kterou zadává obsluha z klávesnice počítače stiskem klávesy „s“ nebo „S“ ve prospěch osciloskopu. Stavová proměnná pro toto rozhodování je po zapnutí, nebo restartu testovacího modulu, přednastavena pro běh podprogramu osciloskopu. Rozhodování mezi během funkcí osciloskopu, nebo funkcí rychlého logického analyzátoru je nutné z důvodu, protože samotné spouštění obou těchto funkcí je shodné a to příjmem znaku „0x03“ USART-em, což je druhou nutnou podmínkou pro běh osciloskopu.

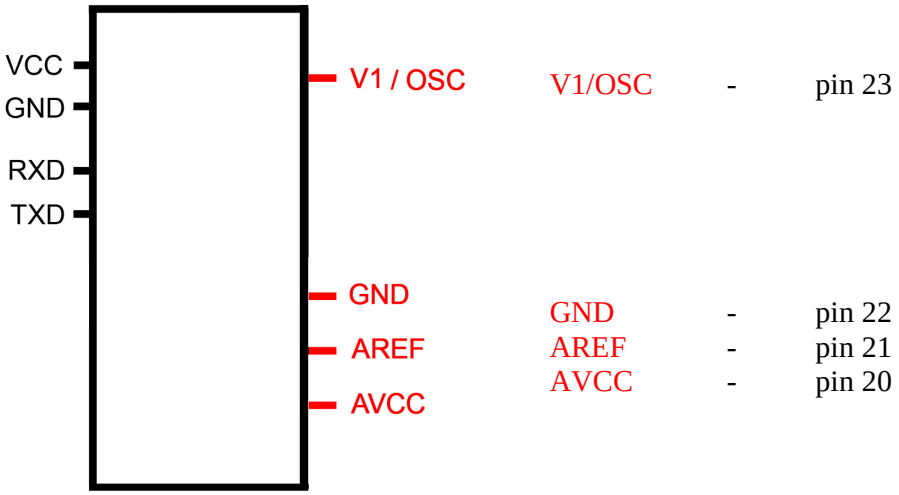
Funkce osciloskopu, podobně jako funkce voltmetru, využívá pro své náměry zabudovaný AD převodník. Na rozdíl od voltmetru, využívá osciloskop pouze osmi-bitové rozlišení AD převodníku. Deseti-bitové rozlišení AD převodníku totiž potřebuje pro svoje měření za pomocí předděličky snížit rychlost hodinového signálu na méně než 200 kHz. Osmi-bitové rozlišení, pro svůj běh využívající rychlosti hodinového signálu až do frekvence 1MHz, tak může své náměry měřit pětinásobně rychleji. Je to ale na úkor přesnosti. Náměr osciloskopu je kvůli menšímu rozlišení zatížen čtyřikrát větší chybou vůči náměru voltmetru. Pro měření AD převodníkem funkce osciloskopu využívá napěťovou referenci přímo od napájení AD převodníku (pin AVCC). Na rozdíl od všech ostatních funkcí testovacího modulu je u osciloskopu použito přerušení (konkrétně přerušení po ukončení AD převodu).

V případě zjištění ukončení náměru AD převodníku se běh podprogramu osciloskopu přeruší a skočí se do obslužné rutiny přerušení, kde se náměr AD převodníku uloží do vnitřní paměti a ukazatel na místo v paměti se inkrementuje. Poté se začne provádět nový náměr. To se opakuje až do doby, kdy je softwarově zjištěno naplnění poslední buňky vnitřní paměti. Velikost paměti mikroprocesoru ATmega8 je 1kByte, což v ideálním případě umožní osciloskopu uložit 1024 náměrů. Některé Byty datové paměti jsou ale využity pro pomocné proměnné, proto je počet možných náměru nižší, konkrétně Testovací modul provádí „900“ náměrů. Po ukončení měření se všechny náměry vyšlou po sériové lince do počítače, kde se náměr zobrazí graficky v programu zobrazovače „NRM Display“ (viz kap. 4.5).

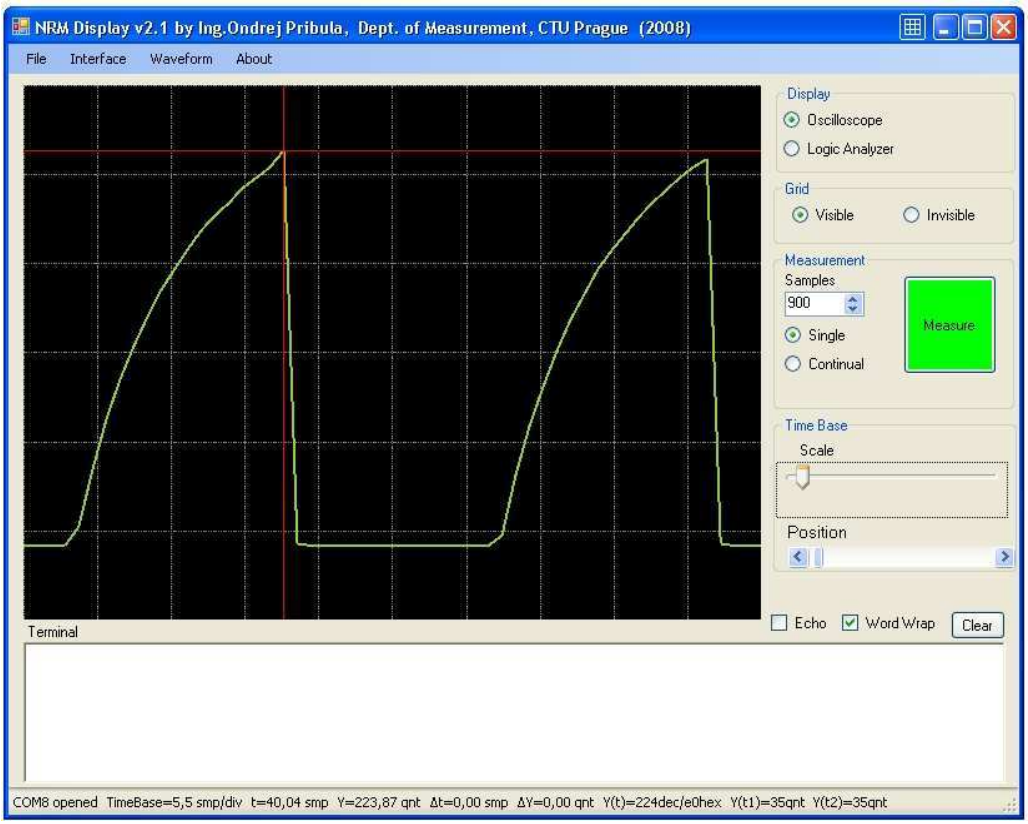
Z důvodu šetření pinů mikroprocesoru, je pro funkci osciloskopu použit stejný pin, jako pro funkci voltmetru (kanál 0 – pin23). Osciloskop je naprogramován pouze jednokanálově.

Náměry uložené ve vnitřní datové paměti mikroprocesoru neodpovídají skutečné naměřené hodnotě. Kdyby tomu tak mělo být, musela by se použít aritmetika pro přepočet náměru (čísla od 0 do 255) na číslo odpovídající napětí měřeného signálu. To by ale způsobilo příliš velké zdržení smyčky osciloskopu. Navíc, měření jednoho vzorku je osmi-bitové. Aby se po přepočtu nesnížila přesnost ukládaného náměru, musel by se náměr násobit konstantou větší než 1, čímž by se z něj stal více než osmi-bitový náměr. To by se ale negativně projevilo v počtu uložených náměru do vnitřní paměti, protože jeden náměr by už nezabíral pouze jeden bajt paměti, ale dva. Do vnitřní paměti by se tedy mohlo uložit maximálně 512 náměrů. Přepočet na správnou hodnotu napětí se tedy neprovádí, v programu zobrazovače je náměr kvantován od hodnoty 0qnt do hodnoty 255qnt.

Po připojení Testovacího modulu k počítači má tedy obsluha na výběr komunikaci v programu „Hyperterminál“, nebo pomocí zobrazovače „NRM Display“. Současné připojení obou těchto programů na stejný COM-port ale není možné. Současně může komunikovat s daným COM-portem pouze jediný program. Přepínání mezi programy s připojením na daný COM-port je komplikované. Před změnou je totiž potřeba zrušit volání daného COM-portu v aktuálně využívaném programu a zavolat COM-port druhým programem. Pro tyto požadavky byl program zobrazovače, který ve své první verzi ještě neměl funkci terminálu, přepracován Ing. Ondrejom Pribulou na verzi, která již v sobě funkci terminálu má. Tím se stává program zobrazovače ideální pro použití k testovacímu modulu.



Obr. 4.16 Potřebné vstupy pro měření osciloskopem



Obr. 4.17 Ukázka náměru osciloskopu v programu „NRM Display“

4.4.6 Rychlý logický analyzátor

Funkce logického analyzátoru má na svém začátku stejně jako funkce osciloskopu zjištění stavové proměnné pro rozhodování mezi aktivací funkce osciloskopu, nebo rychlého logického analyzátoru. Obsluha umožní měření rychlého logického analyzátoru zmáčknutím kláves „a“ nebo „A“. Následně se zkoumá přijetí bajtu 0x03, což je druhá nezbytná podmínka ke startu této funkce. Poté se již cyklicky začne provádět samotné měření logických stavů na vstupech „**IN0 - IN3**“.

Funkce rychlého logického analyzátoru má ze všech funkcí testovacího modulu nejrychlejší vzorkovací frekvenci, která je dána rychlostí průběhu vnitřní zacyklené smyčky. V této smyčce se odečte stav na vstupech IN0-IN3, uloží se do vnitřní datové paměti a inkrementuje se ukazatel místa paměti. Další činností je zjištění, zda-li náměr není ukládán do posledního místa v paměti. Protože adresace je šestnácti-bitová, musí se test dělat nadvakrát. V případě rovnosti vyššího adresovacího bajtu paměti s konstantou danou vyšším

bajtem adresy poslední paměťové buňky, se začne provádět také kontrola nižšího bajtu adresace. Tím se ale zpomalí průběh smyčkou o tuto kontrolu. Proto je potřeba zabezpečit, aby i v případě, kdy není prováděn test nižšího bajtu adresace, byla navíc držena smyčka o dobu trvání testu nižšího bajtu, která se vícekrát provede instrukcí, mající za úkol pouze pozdržení jednoho hodinového taktu. Tím se zabezpečí, že vzorkovací kmitočet rychlého logického analyzátoru bude za každé vstupní podmínky související s adresovací hodnotou paměti stejně dlouhý.

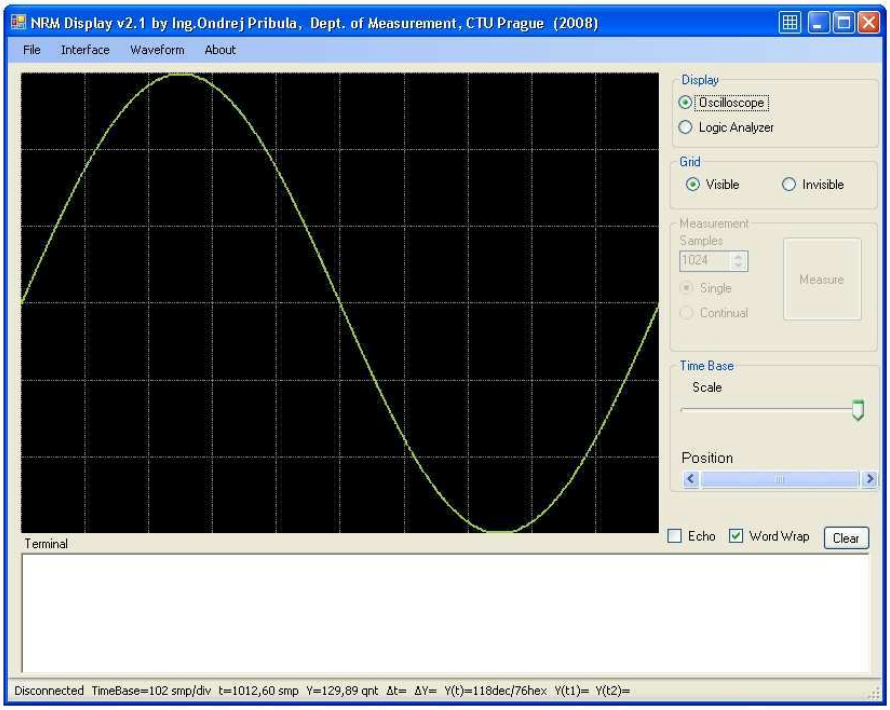
Z hlediska funkčnosti měření by již zmíněné kroky po zacyklení postačovaly k samotnému měření. Výhodná je ale taky funkce změny vzorkovacího kmitočtu, proto se ve smyčce nachází další smyčka, která vzorkovací kmitočet umožňuje zpomalit. Konstanta pro zpomalení smyčky je přednastavena na minimální hodnotu zdržení. Jeden získaný odměr, tj. jeden průběh smyčkou analyzátoru, trvá 14 hodinových taktů. Nejrychlejší vzorkovací frekvence rychlého logického analyzátoru je tedy přibližně 570kHz.

Po naměření možného počtu náměrů se smyčka měření ukončí a program skočí do smyčky vyslání dat. První vyslaný bajt musí být hodnota 0x03, aby se splnila komunikační podmínka zobrazovače pro odběr dat, které mají být zobrazeny v grafickém okně. Poté se postupně vysílají naměřené data z paměti, přičemž před samotným vysláním dat se vhodně upraví tak, aby ve svých osmi bitech obsahovaly pouze informaci ze vstupů IN0-IN3 a ostatní bity byly nastaveny do úrovně logické nuly. Po ukončení vysílání naměřených dat se z podprogramu rychlého analyzátoru vyskočí.

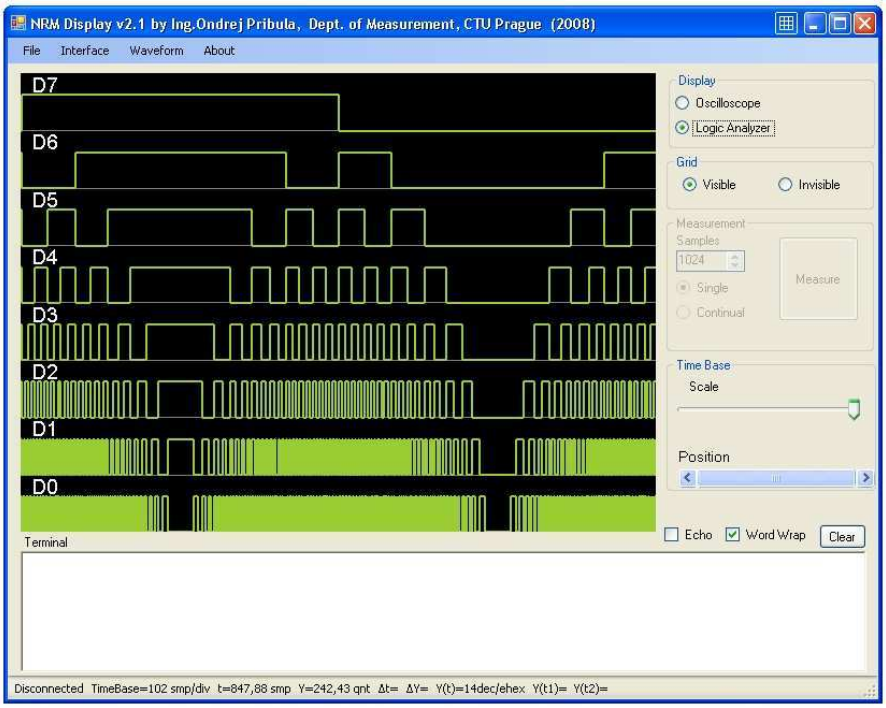
4.5 Zobrazovač „NRM Display“

Program „Zobrazovač“ je software, který vytvořil Ing. Ondrej Pribula. „Tento program umožňuje data, přijaté sériovou linkou, zobrazovat graficky. Taky umožňuje jednoduchou terminálovou komunikaci s připojeným zařízením. Pokud program zobrazovače zaregistruje příjem zobrazitelného znaku, nebo znaku CR nebo LF, vypíše ho v terminálovém okně. V případě příjmu nezobrazitelného znaku jiného než CR, LF nebo bajtu 0x03, změní se podkladová barva terminálu na červenou, což informuje uživatele, že pravděpodobně není něco s komunikaci v pořádku. Pokud je vybráno terminálové okno, znaky přijímané z klávesnice se odešlou sériovou linkou. Po přijetí znaku „03“ program předpokládá, že tento bajt bude následován dalšími bajty, jejichž hodnota se má zobrazit graficky. Program předpokládá, že počet těchto dat se rovná hodnotě zadané v editačním poli „Samples“. Pokud není splněna podmínka přijetí definovaného počtu bajtů do času nastaveného v menu

„Interface / TimeOut“, program upozorní obsluhu na nekonzistenci dat a přejde do definovaného počátečního stavu, tedy do stavu textových dat pro terminál. Stlačením tlačítka „Measure“ při současné volbě „Single“ program odešle bajt s hodnotou „03“ na sériový port. Pokud je zvolena možnost „Continual“, po každém správném balíku dat pro zobrazovač (bajt 0x03 + počet bajtů definovaný v „Samples“) program znovu vysílá bajt 0x03. “



Obr. 4.18 Ukázka programu NRM Display v režimu „Oscilloscope“



Obr. 4.19 Ukázka programu NRM Display v režimu „Logic Analyzer“

5 Hardware „Testovací modul“

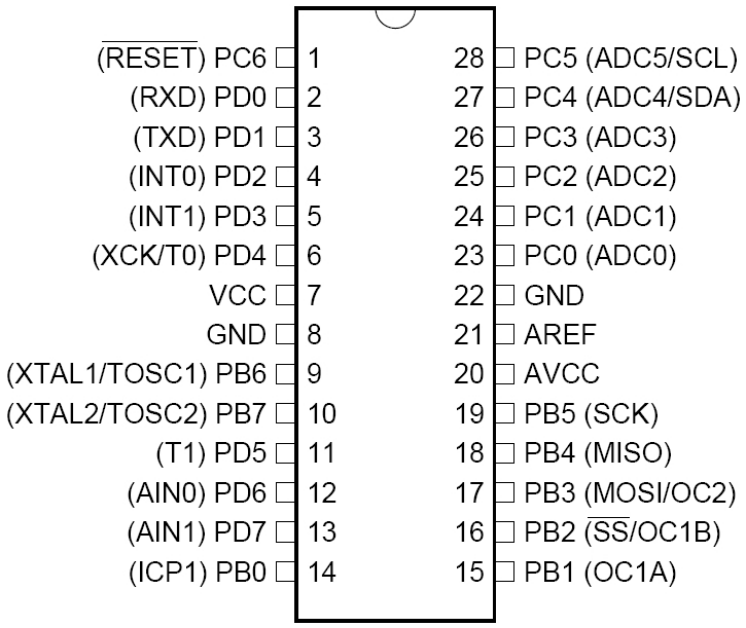
5.1 Mikroprocesor ATmega8

Jak již bylo zmíněno v kap. 2, zabývající se problematikou řešení přípravku Testovacího modulu, byl pro tuto práci vybrán mikroprocesor ATmega8 firmy Atmel.

ATmega8 je nízkopříkonový osmi-bitový mikroprocesor, který provádí své instrukce v jednom až tří hodinových cyklech.

Základní vlastnosti mikroprocesoru ATmega8 :

- instrukční soubor obsahuje 130 instrukcí
- obsahuje 32 registrů délky 8 bitů
- hodinový kmitočet až 16MHz, zabudovaný RC oscilátor až 8MHz
- tři vstupně/výstupní porty, z toho dva jsou osmi-bitové a jeden sedmi-bitový, celkem tedy 23 programovatelných vstupů/výstupů
- paměť programu je tvořena zabudovanou Flash, které kapacita je 8KB a počet přeprogramování je 10 000 cyklů
- 512 bajtů EEPROM, které počet možných přeprogramování je 100 000 cyklů
- datová paměť SRAM kapacity 1KB
- dva osmi-bitové a jeden šestnácti-bitový čítač/časovač
- tři PWM kanály
- šesti-kanálový deseti-bitový AD převodník
- jednotky USART, TWI, SPI
- jednotka Watchdog Timer
- napájecí napětí 2,7-5,5 V (ATmega8L) nebo 4,5-5,5 V (ATmega8)

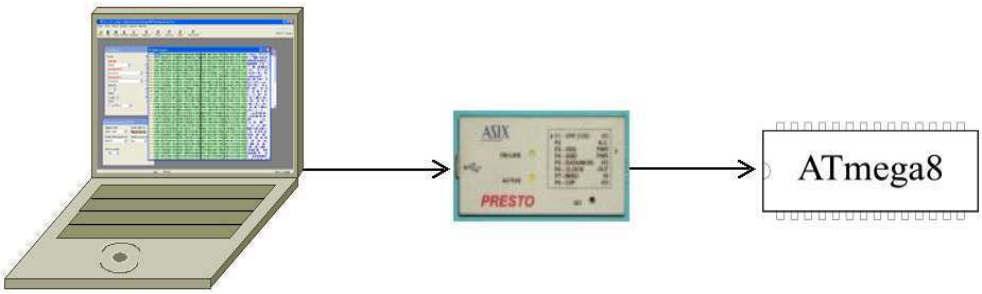


Obr. 5.1 Rozložení vývodů mikroprocesoru ATmega8 (PDIP 28)

5.2 Napálení programu do mikroprocesoru

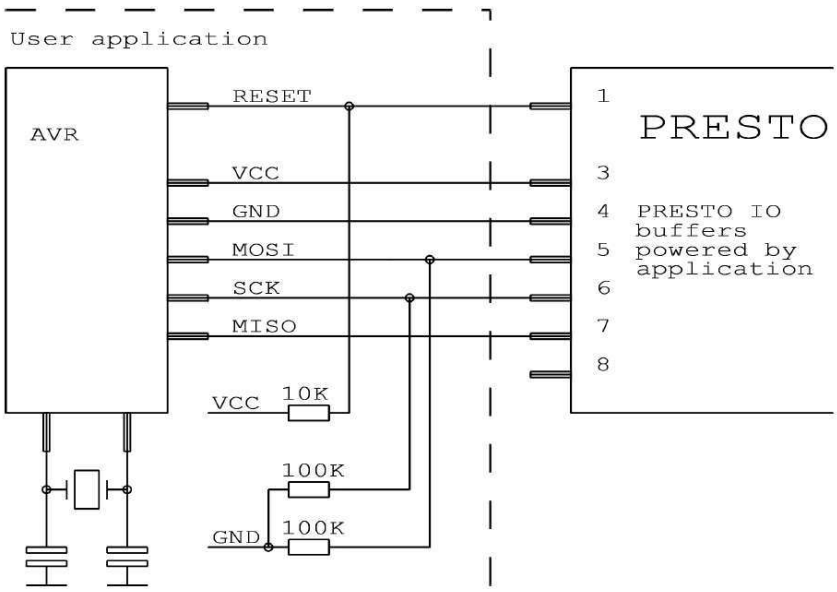
Mikroprocesory ATmega8 je možné programovat paralelně, nebo pomocí SPI rozhraní, tj. sériově. V dnešní době je na trhu značné množství různých programátorů a emulátorů, které jsou univerzální pro programování více druhů mikroprocesorů.

Ukázkový přípravek Testovacího modulu byl naprogramován sériově, pomocí programátoru „PRESTO“, k němuž je dodáván software „UP“. Vstup programátoru PRESTO se přes USB připojí k PC a výstup přes konektor ISP (In-System Programming) k programované součástce (viz obr. 5.2).



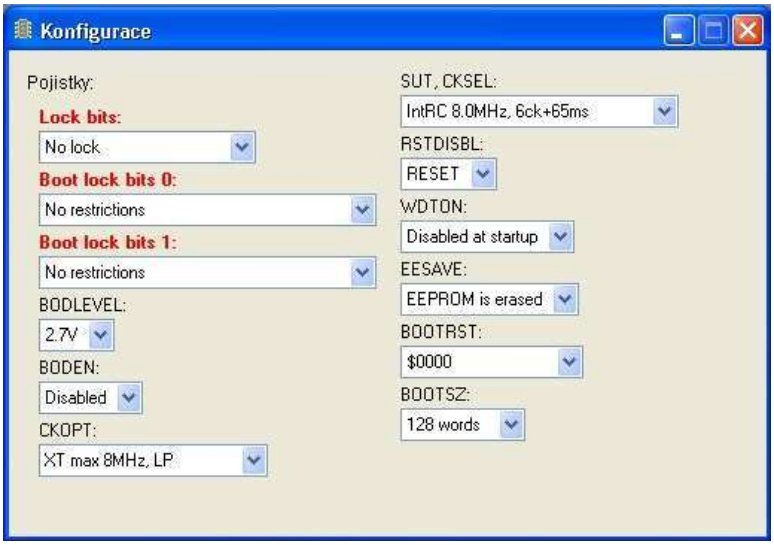
Obr. 5.2 Blokové schéma programování Testovacího modulu

K programovanému mikroprocesoru se musí připojit několik málo součástek podle schématu uvedeném ve firemním návodu (viz obr. 5.3). Při programování není možné využívat vnitřní RC-oscilátor, k mikroprocesoru se musí připojit externí krystal.



Obr. 5.3 Připojení programátoru PRESTO k programované součástce

Před samotným naprogramováním je potřeba pro konkrétní mikroprocesor nastavit několik parametrů, označovaných jako „pojistky“ (viz kap. 5.3). Ukázku nastavení pojistek je možné vidět na obr. 5.4 .



Obr. 5.4 Ukázka nastavení pojistek před naprogramováním mikroprocesoru ATmega8

5.3 Popis nastavených pojistek

Lock bits :

Nastavení „Lock bits“ umožňuje tři typy ochrany paměti :

1: není aktivována žádna ochrana

2: zákaz programování Flash a EEPROM

3: zákaz programování a čtení Flash a EEPROM – nastavení této ochrany zamezuje možnosti zpětné verifikace programu, brání tedy neoprávněnému kopírování navrženého programu.

Testovací modul nemá pojistkou „Lock bits“ aktivovanou žádnou ochranu !

Boot Lock bits 0 :

Umožňují nastavit omezení aplikační sekci Flash pro instrukce SMP (zápis do paměti programu) a LPM (čtení paměti programu).

Boot Lock bits 1 :

Umožňují nastavit omezení boot sekci Flash pro instrukce SMP (zápis do paměti programu) a LPM (čtení paměti programu).

Testovací modul nemá pojistkami „Lock bits 0, Lock bits 1“ aktivovanou žádnou ochranu !

BODLEVEL :

Umožní nastavit úroveň detektoru výpadku napájení. Na výběr jsou hodnoty „2,7V“a „4V“

BODEN :

Ovládá detektor výpadku napájení. Detektor může být aktivován nebo deaktivován

CKOPT :

Nastavení oscilátoru – pomocná volba spojená se synchronizačním zdrojem

CKSEL :

Volí zdroj synchronizace. Na výběr jsou vnější krystal nebo rezonátor, vnější nízkofrekvenční krystal, vnější RC oscilátor, kalibrovaný vnitřní RC oscilátor nebo vnější hodiny.

SUT :

Umožňuje nastavit volbu startovací prodlevy při probuzení mikroprocesoru

RSTDISBL :

Umožní nastavit, zda-li se bude pin1 mikroprocesoru ATmega8 používat pro RESET, nebo bude používán jako vstupně/výstupní pin

WDTON :

Umožňuje nastavení povolení funkce Watchdog Timer

EESAVE :

Je-li tato propojka naprogramována, je obsah EEPROM paměti zachován i po vykonání instrukce smazání čipu

BOOTRST :

Je-li tato propojka naprogramována, jsou vektory přerušení a resetu umístěny do boot loader sekce a ne do aplikační sekce Flash

BOOTSZ :

Umožňuje nastavení velikosti boot loader sekce. Na výběr jsou hodnoty {128, 256, 512 a 1024}

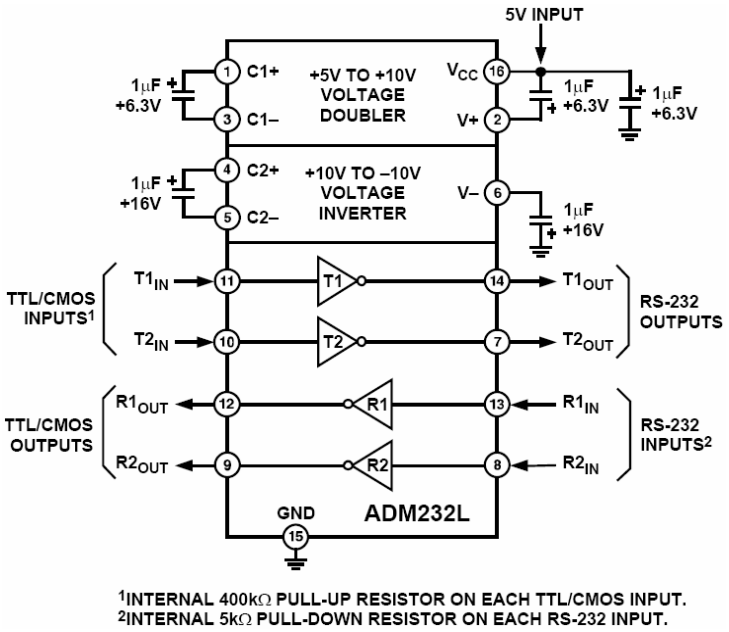
5.4 Oživení přípravku a připojení k počítači

Přípravek *Testovací modul* ke své správné funkci potřebuje připojení napájecího napětí „**+5V** a **0V**“ ze stabilizovaného zdroje (viz obr. 5.6). Podle nastavených pojistek „*SUT, CKSEL*“ (viz kap. 5.3) může přípravek pro generaci své taktovací frekvence využívat vnitřní RC oscilátor, nebo se může na vstupy mikroprocesoru „*XTAL1, XTAL2*“ připojit externí krystal se dvěma kondenzátory. Správné splnění zmíněných podmínek je plně dostačující k oživení přípravku.

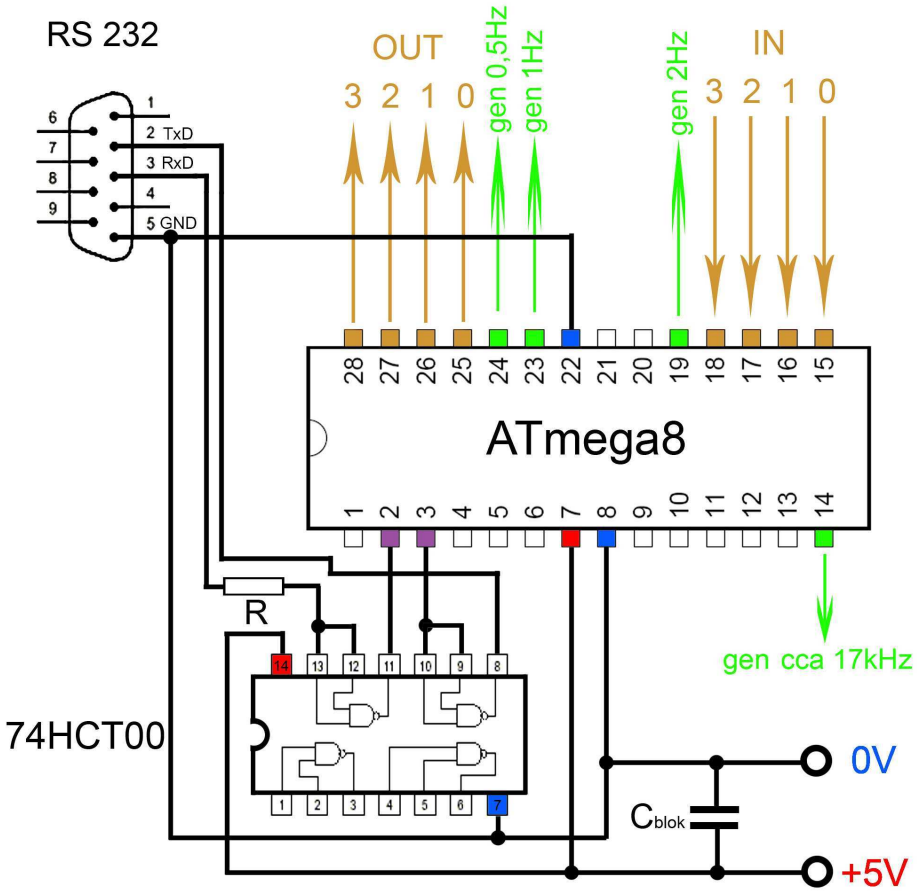
Zadání *Testovacího modulu* předpokládá co nejjednodušší zapojení mikroprocesoru. Byla proto využita možnost nastavení pojistek pro využití vnitřního RC oscilátoru, kde není potřeba pro běh mikroprocesoru použít žádné externí součástky. Je však vhodné, dokonce žádoucí, co nejblíže k vývodům mikroprocesoru pro napájecí napětí připojit vnější blokovací kondenzátor. Použití blokovacího kondenzátoru je nutné kvůli mikroprocesorem odebíraným proudovým špičkám. Doporučená hodnota blokovacího kondenzátoru je řádově v jednotkách až desítkách μF.

Rozhodnutí mezi použitím externího krystalu, nebo použitím vnitřního RC oscilátoru před naprogramováním je na samotném uživateli . Při použití externího krystalu je mnohem přesnější generace hodinového kmitočtu a mikroprocesory ATmega8 umožňují pracovat s vnějším připojeným krystalem až na frekvenci 16MHz. Použití vnitřního RC oscilátoru je výhodné z hlediska jednoduššího zapojení přípravku. To ale přináší nevýhodu méně přesné generace hodinového kmitočtu (výrobce udává nepřesnost ±3%) a také možnost maximální možné generace hodinového kmitočtu pouze 8MHz. RC oscilátor je ale výrobcem nakalibrován a nepřesnost ±3% při používání nastaveného USART-u není překážkou. Použít externí krystal je tedy nutné pouze v případě, že uživatel požaduje přesnou generaci frekvence mikroprocesorem.

Testovací modul komunikuje s počítačem přes sériový port. Aby byla komunikace možná, je potřebné zapojit vývody USART-u „*RXD, TXD*“ na vstup integrovaného obvodu ADM232 nebo MAX232 a jeho výstup připojit pomocí sériové linky k počítači (viz obr. 5.5). Integrovaný obvod ADM232 (MAX232) plní funkci nábojové pumpy využívající připojených kondenzátorů. Mikroprocesor využívá pro svůj běh pouze +5V a 0V. Sériová linka je standardem definovaná na úrovně „+3V až +15V“ pro logickou nulu a „-3V až -15V“ pro logickou jedničku. Tyto hodnoty nám správně zabezpečí jíž zmíněný integrovaný obvod ADM232 (MAX232). Při oživování je zde možnost toto, relativně náročnější zapojení, obejít jednodušším zapojením CMOS logického obvodu NAND (viz obr. 5.6). Není to ale z hlediska normy pro sériový port správné zapojení. Také nemusí ve všech případech fungovat správně.



Obr. 5.5 Zapojení integrovaného obvodu ADM232



Obr. 5.6 Ukázka možného zapojení Testovacího modulu (verze1) při oživování

Komunikaci *Testovacího modulu* s počítačem zabezpečuje program *Hyperterminál*. V programu Hyperterminál je potřeba nastavit některé parametry, aby byla komunikace mezi počítačem a Testovacím modulem úspěšná. Ty se nastaví po prvém spuštění, kdy se program Hyperterminál dotáže na název nového připojení. Poté je potřeba nastavit „Připojit pomocí“, kde se zadá COM-port, na kterém bude komunikace probíhat. Následně se nastaví „komunikační rychlost“, „počet datových bitů“, „parita“, „počet stop bitů“ a „řízení toku“. Pod zadaným názvem se komunikační nastavení uloží.

5.5 Nastavení programu Hyperterminál

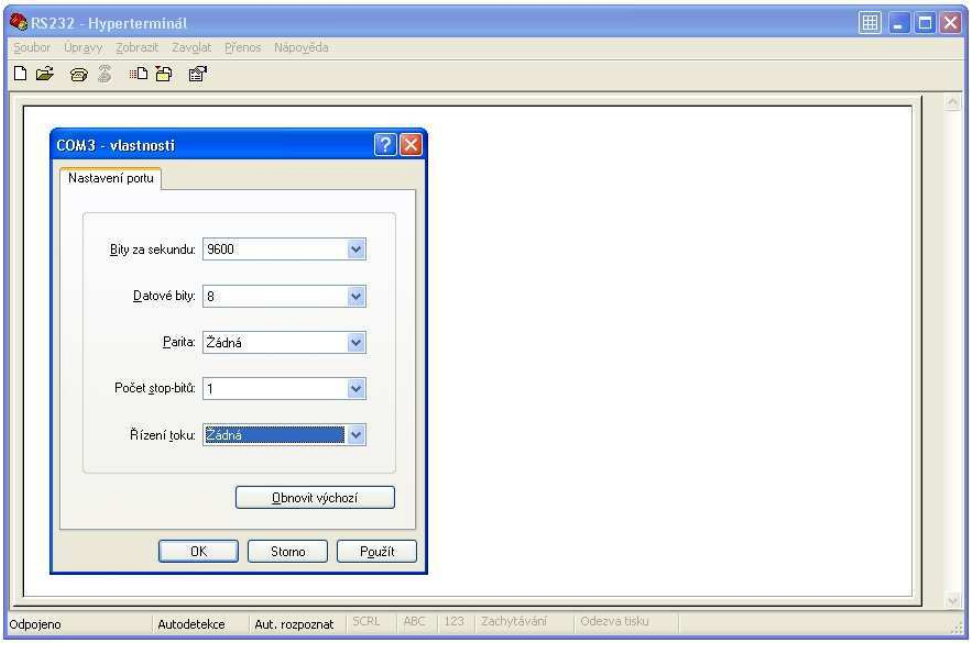
Program Hyperterminál se spouští v operačním systému Windows následovně:

„Start – všechny programy – příslušenství – komunikace – Hyperterminál“

Jak již bylo zmíněno, přípravek komunikuje s počítačem pomocí sériového portu. Uživatel při nastavování „Připojit pomocí“ zvolí COM-port, na který v počítači připojí sériový kabel. Novější počítače v dnešní době zpravidla sériový port neobsahují, nicméně se běžně dají sehnat převodníky USB-RS232. Po připojení převodníku k USB v počítači, je přiřazeno převodníku číslo COM-portu. Aby uživatel toto číslo zjistil, stačí když v *ovládacích panelech* Windows, podsložce *„Systém – hardware - správce zařízení – porty(COM a LPT)*“ vyhledá, který COM-port je přiřazen převodníku a v programu Hyperterminálu se daný COM-port nastaví pro komunikaci s *Testovacím modulem*.

Dále je potřeba nastavit :

- přenosová rychlost = 9600Bd
- počet datových bitů = 8
- parita = žádná
- stop bit = 1
- řízení toku = žádná



Obr. 5.7 Ukázka nastavení programu Hyperterminál

6 Klávesové zkratky používané v Testovacím modulu

Podprogramy pracující s výstupy OUT0-OUT3

„i, I“	**-**	inkrementace
„d, D“	**-**	dekrementace
„L“	-	nastaví příznak pro zadávání stavu logické nuly (přednastavená hodnota po zapnutí)
„H“	**-**	nastaví příznak pro zadávání stavu logické jedničky
„0“	**-**	na pinu OUT0 se nastaví posledně zadaný příznak logické úrovně
„1“	**-**	na pinu OUT1 se nastaví posledně zadaný příznak logické úrovně
„2“	**-**	na pinu OUT2 se nastaví posledně zadaný příznak logické úrovně
„3“	**-**	na pinu OUT3 se nastaví posledně zadaný příznak logické úrovně
„p, P“	-	rotace krokového motoru doprava
„o, O“	-	rotace krokového motoru doleva
„u, U“	-	uvolnění výstupních pinů krokovému motoru a jeho aktivace
„z, Z“	-	deaktivace krokového motoru

Simulace inkrementálního snímače – výstupy MOU0,MOU1

„q“	–	rotace doleva jednorázová - „single mode“
„Q“	–	rotace doleva kontinuální - „continual mode“
„w“	–	rotace doprava jednorázová - „single mode“
„W“	–	rotace doprava kontinuální - „continual mode“
„e, E“	–	ukončení generování a vynulování výstupů

Osciloskop / rychlý logický analyzátor

„s, S“	–	povolení funkce osciloskopu (přednastaveno po zapnutí nebo restartu)
„a, A“	–	povolení funkce rychlého logického analyzátoru

7 Závěr

Po zhodnocení funkčnosti přípravku „*Testovacího modulu*“ lze konstatovat, že přípravek funguje správně. Splnění cílů bakalářské práce, kromě programu pro nadřazené PC, bylo tedy úspěšně zvládnuto. Program „*Zobrazovač*“ v průběhu vypracování této bakalářské práce již vyvinul Ing. Ondrej Pribula. Proto byla snaha místo programování jiného zobrazovacího programu zaměřena na dopracování více různých generátorů pro Testovací modul.

Výsledkem této práce není nalezení něčeho, co ještě vymyšleno nebylo. Na trhu je velké množství měřících přístrojů a generátorů frekvence, pracujících navíc s mnohem větší přesností. V praxi se ale ukazuje, že často je potřeba k otestování své aplikace použít například pouze jediný puls, nebo naopak specielní posloupnost pulsů, což ani mnohé profesionální generátory nedokáží.

Testovací modul kromě různých generátorů umí měřit logické úrovně a napětí, což jej po jednoduchém připojení do testované aplikace dělá univerzálním testovacím přípravkem. V dnešní době je cena mikroprocesoru ATmega8 použitého pro tuto práci hodně nízká, pohybuje se řádově v desetikorunách, což se nedá porovnávat s cenou originálních měřících přístrojů a generátorů.

Snažil jsem se udělat přípravek Testovacího modulu takový, aby jeho použití a připojení k testovaným aplikacím bylo co nejjednodušší a ovládání intuitivní. Mnohdy se ale až v praxi ukáží různé nedostatky, nevhodné nebo nedostačující parametry funkcí, které je potřeba doladit či předělat, proto jsem přislíbil vedoucímu mé bakalářské práce Ing. Janu Fischerovi, že po zjištění nedostatků se je budu snažit odstranit. Doufám, že tento přípravek bude mít na cvičeních s mikroprocesorovou technikou pozitivní uplatnění a pro studenty bude přínosem spočívajícím v ulehčení práce při oživování a testování svých aplikací.

Můj zájem o mikroprocesorovou techniku se projevil již před léty, přesto jsem se s ní prakticky seznámil až při tvorbě bakalářské práce a osvojil jsem si ji natolik, že kromě aplikace v bakalářské práci, jsem ji použil také ve vlastních návrzích a v zaměstnání.

Čeština není můj rodní jazyk, ale jelikož studuji na české univerzitě a v České republice jsem už pět let, napsal jsem tuto bakalářskou práci v českém jazyce. Ještě jednou děkuji všem, kteří mi pomohli se zvládnutím české gramatiky při psaní této práce.

Seznam použité literatury :

[1] Vedral J., Fischer J.: Elektronické obvody pro měřící techniku, Vydavatelství ČVUT, Praha 2004, ISBN 80-01-02966-2

[2] Matoušek D.: Práce s mikrokontroléry ATMEL ATmega16, Nakladatelství BEN, Praha 2006, ISBN 80-7300-174-8

[3] Atmel: data sheet ATmega8 (http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf)

[4] Atmel: 8-bit AVR Instruction set (http://www.atmel.com/dyn/resources/prod_documents/DOC0856.PDF)

[5] stránky katedry měření ČVUT, FEL, NRM Display (http://measure.feld.cvut.cz/cs/vyuka/predmety/x38nrp/NRMDisplay)

Použitý software :

[a] Atmel AVR Studio 4

[b] Asix UP PRESTO

[c] zobrazovač NRM Display vytvořený Ing. Onrejom Pribulou

Seznam příloh na CD:

Program Testovacího modulu ve tvaru assembler source „testModul.asm“ a v přeloženém hexadecimálním tvaru „testModul.hex“

Program generátoru ve tvaru assembler source „generator.asm“ a v přeloženém hexadecimálním tvaru „generator.hex“