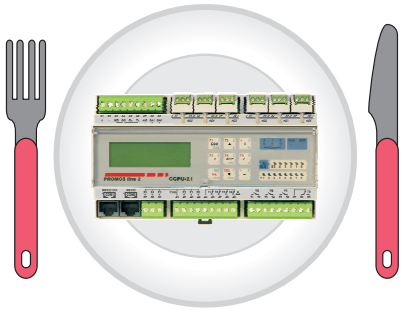
Kuchařka

Komunikace výrobků ELSACO

XDM-14A, SAM-01, SAM-02, SBI-11/12, SBIO-11/12, SBO-11/12, SAIO-12



Jaselská 177, 280 02 KOLÍN 3
tel./fax: 321 727 753

Verze 1.02 03/2005

23.3.2005
Účelová publikace ELSACO

ELSACO, Jaselská 177, 280 02 Kolín 3
Tel./fax/modem: 321 727 753 / 321 727 759

Internet : **www.elsaco.cz**

Připomínky : vacek@elsaco.cz

Obsah kuchařky

1 Panelový číslicový indikátor XDM-14A

1.1 Úvodem

Přístroj XDM má implementován jednoduchý protokol SAM. Protokol je vskutku jednoduchý, je dost dobře popsán v manuále. Přesto vás poprvé XDMko či protokol SAM potrápí.

Musíte totiž nejdříve pochopit princip chování přístroje po jeho zapnutí.

V době těsně po **každém** zapnutí přístroje (cca 1,5 sec) očekává přístroj sekvenci tří po sobě jdoucích znaků ESC (escape, kód 27 [dekadicky]). Tyto znaky očekává na svém sériovém kanálu, nastaveném těsně po každém zapnutí na

2400 Bd

bez parity

1 stop bit

a adresa přístroje je **00** !

Pokud je dostane, přechází přístroj do tzv. konfiguračního režimu.

Pokud je nedostane, pak přístroj provede příkazy, které má po konfiguračním režimu (po konfiguraci) uložené v paměti EEPROM. Tím se po každém zapnutí přístroj nakonfiguruje podle toho, co má v této paměti uloženo, tj. teprve potom platí pro přístroj např. jiná komunikační adresa (než 00 po zapnutí), jiná komunikační rychlost (než 2400 Bd po zapnutí), atd. ...

Při prvním seznamování se s přístrojem a s protokolem SAM doporučuji udělat pár pokusů na stole. Nakonec stejně musíte nějakou konfiguraci přístroje zvolit a uložit do paměti EEPROM, aby se přístroj pak choval podle toho jak potřebujete (přinejmenším bude nutné zvolit jeho komunikační adresu, případně rychlost).

1.1.1 Propojení PC s XDM

V praxi budete používat obdobné přístroje na lince RS-485 jednak proto, že dovoluje připojení přístroje na větší vzdálenost (do 1200m) , jednak také proto, že vám tato linka umožňuje připojit na ni více přístrojů a ty pak adresovat.

Proto k PC připojíte nejprve převodník RS-232/RS-485, např. stolní převodník SLC-67, obj.č. EI5067.40, který je užit v tomto konkrétním příkladě.

Propojení s PC je standardním propojovacím kabelem pro RS-232 s konektory CANON DB9.

Propojení linky RS-485 je v následující tabulce:

převodník - DB15		svorkovnice XDM-14A	
4	-RxTxD	2	-RxD
		4	-TxD
11	+RxTxD	1	+RxD
		3	+TxD

Toto propojení doporučujeme s ohledem na další kapitoly provést s mezisvorkovnicí - tzv. čokoládou ap.

Pro úplnost popíšeme propojení slovně - na XDM propojíme svorky 2+4 a z nich vedeme vodič na první svorku čokolády, do které připojíme i vodič ze svorky 4 konektoru DB15 převodníku SLC-67. Dále na XDM propojíme svorky 1+3 a z nich vedeme vodič na druhou svorku čokolády, do které připojíme vodič ze svorky 11 konektoru DB15 převodníku SLC-67.

XDM napájíme podle štítku a popisu svorek na přístroji, např. sv. 9 = GND a sv. 10 = + 12V.

1.1.2 Terminál

Po fyzickém zapojení přístroje na napájení a komunikační linku RS-485 přes převodník k PC, musíme vybrat vhodný PC program pro odesílání a příjem znaků mezi PC a přístrojem. Těmto programům se obecně říká TERMINAL.

Na www.elsaco.cz je ke stažení program TERMINAL.EXE, který je funkční i ve Windows 2000, Windows XP ... Umožňuje využít kterýkoliv COM vašeho PC, zvolit parametry COMu, ... Odesílané i přijímané znaky zobrazuje v příslušných oknech programu, lze odesílat předvolené sekvence (dávky) znaků či odesílat znaky ze souboru. Předesíláme, že je možno zvolit kterýkoliv program TERMINAL, na který jste zvyklý a který umíte spolehlivě ovládat.

Poznámka: výrobce nedávno na svých stránkách zveřejnil beta verzi programu TESTER pro konfiguraci a testování výrobků s komunikací. Součástí TESTERu je i HELP, chystá se do něj i terminál.

Na obr. 1 vidíte nastavení programu TERMINAL.EXE pro konfiguraci přístroje XDM-14A. Všimněte si vpravo dole rámečku s označením TRANSMIT MACROS a definice jednotlivých maker. Jako M1 vidíte sekvenci tří znaků ESC (kód znaku ESC = 027 dekadicky). M2 pak s kódem 013 nahrazuje klávesu ENTER - čili zakončení běžného ASCII příkazu, které se také označuje jako <CR>. M3 je pak předdefinováno pro stanici s adresou 07 pro nápis AHOJ i s ukončovacím znakem příkazu <CR>, tj. to #013. Všimněte si, že jednotlivá makra lze odeslat jednak kliknutím na tlačítko M1/M2/M3 nebo horkou klávesou F1 pro M1, F2 pro M2 a F3 pro M3 (toho lze velmi vhodně využívat např. pro přechod do konfiguračního módu - stiskem F1). Pokud chcete makro opakovat, stačí nadefinovat čas opakování v okénku vedle tlačítka a zaškrtnout na konci řádku tuto volbu pro opakování.

1.1.3 Přechod do konfiguračního režimu

V TERMINAL.EXE zkontrolujte či zeditujte nastavení COMu tak, aby bylo nastaveno:

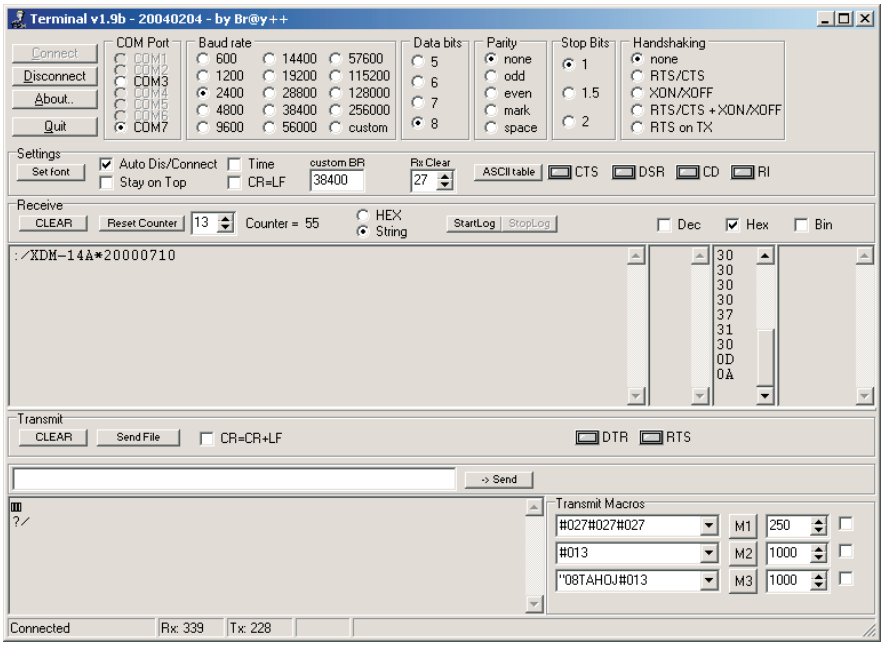
komunikační rychlost (Baud rate) na **2400** Bd

Data bits na **8**

Parity na **none**

Stop Bits na **1**

Handshaking na **none**



Obr. 1 Okno programu TERMINAL.EXE při počátku konfigurace XDM-14A

Dále zkontrolujte, zda zvolený COM PC má odezvu v převodníku. To stačí psát do levého dolního okna nějaké znaky (třeba párkrát zmáčknout klávesu ESC), a na LEDkách převodníku musíte vidět, že probliknou.

V programu TERMINAL.EXE je vhodné zaškrtnout volbu AUTO DIS/CONNECT. Tak po spuštění tohoto programu je program "připojen" ke zvolenému COMu PC, který tím ovládá. Vlevo nahoře v okně programu zešedne tlačítko CONNECT, čímž je dáno na vědomí, že program s COMem pracuje. Pokud jste neměli volbu AUTO DIS/CONNECT zaškrtnutou, musíte kliknout na tlačítko CONNECT by zešedlo.

Tím by mělo být vše přichystáno pro přechod do konfiguračního režimu.

Vypněte nyní pouze přístroj (XDM-14A). Po jeho zapnutí stačí stisknout horkou klávesu F1 pro spuštění makra M1, které pošle tři znaky ESC na linku.

V levém dolním okně pro TRANSMIT se 3x ESC zviditelní jako tři nalepené obdélníčky nastojato - také paznaky :-).

V okně RECEIVE (nad oknem TRANSMIT) se objeví dvojtečka jako potvrzení přechodu do konfiguračního režimu.

Poznámka:

Netrpělivé povahy mohou po prvním neúspěchu zkusit stiskat klávesu F1 víckrát po sobě nebo dokonce opakování M1 zvolit opakované zaškrtnutím této volby vpravo na konci řádku pro M1 (lze volit i čas opakování - na obrázku 1 zvoleno 250 ms).,

A jak poznáme, že zvolený režim žije?

Pošlete (viz obrázek 1) z okna TRANSMITE příkaz pro zjištění jména jednotky:

```
?/
```

To provedete tak, že kurzor musíte mít umístěn v okně TRANSMIT (pokud tam není, klikněte tam myší). A pak stačí z klávesnice PC zadat uvedené dva znaky, tj. "?/". S každým zadáním znaku vidíte na LEDkách, že komunikace probíhá. Po druhém znaku dojde k výpisu v okně RECEIVE.

Za dvojtečku, která tam je od přechodu do konfigurace, se vypíše odpověď:

```
/XDM-14A*20000710
```

takže přístroj vrací znak /, za něj označení přístroje XDM-14A a znak *, po kterém následuje datum verze firmwaru v přístroji.

Klinutím na tlačítka CLEAR u obou oken (TRANSMIT i RECEIVE) vyčístíte obě okna.

Do okna RECEIVE zadejte příkaz

```
??
```

(dva otazníky), tj. příkaz pro výpis příkazů, které jsou uložené v EEPROM.

Výpis vidíte v okně TRASMIT - viz obrázek 2.

A z těchto příkazů poznáte, zda byl přístroj konfigurován a jak, nebo pokud neobsahuje nic (jen !), pak je platné základní nastavení (jako tu první vteřinu po zapnutí, tj. 2400 Bd, bez parity, 1 stopbit, adresa 00). Pokud nějaké příkazy v tomto výpisu vidíte, musíte je podle manuálu rozlousknout a tak se dozvědět adresu přístroje, komunikační rychlost atd...

Ale nakonec proč bychom to louskali a podřizovali se?! Daleko jednodušší je nachystat si požadovanou konfiguraci do souboru příkazů a původní konfiguraci v přístroji prostě přepsat vlastní !

A nachystat si potřebné příkazy do souboru je spolehlivější a pak je vyslat je jednodušší a rychlejší, než je posílat po jednotlivých znacích z okna RECEIVE nebo po jednotlivých příkazech z řádku vlevo od tlačítka -> SEND (pozor v případě odesílacího řádku musíte doplňovat ukončení příkazu <CR> pomocí makra M2, nejlépe pomocí horké klávesy F2).

1.1.4 Konfigurace XDM-14A

Uvažujme o připojení XDM-14A k CCPU-21, programování aplikace pak v ProgWinu, kde programová smyčka bude prováděna pravděpodobně 1x za jednu nebo tři vteřiny. Proto v konfiguraci XDM využijeme příkazu W pro nastavení soft WatchDog Timeru, a to na 10 sec, Z technického manuálu o protokolech vyčteme, že příkaz má obecný tvar

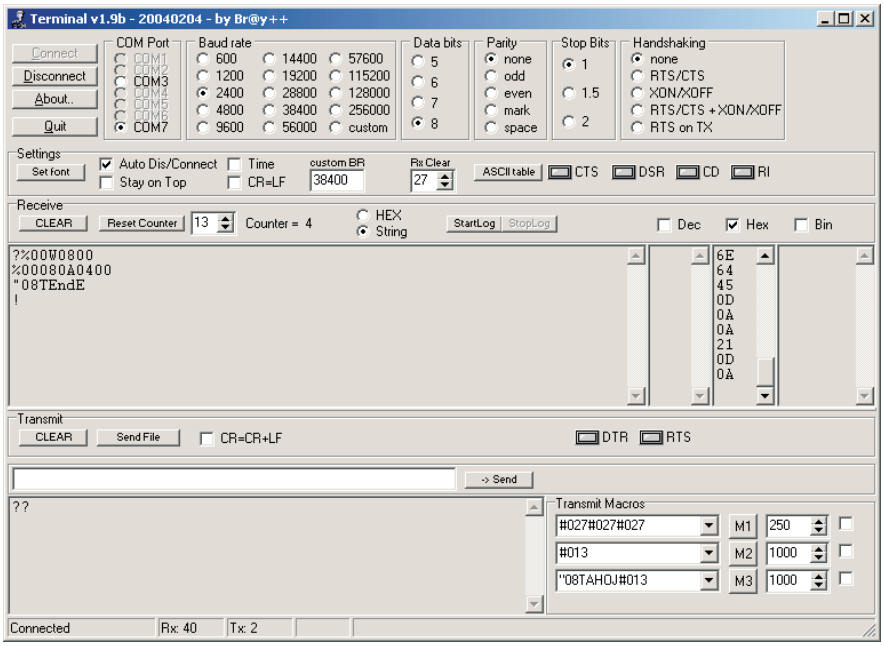
```
%aaWnnnn<CR>
```

10 sec je 10000 ms a v hexa = 2710

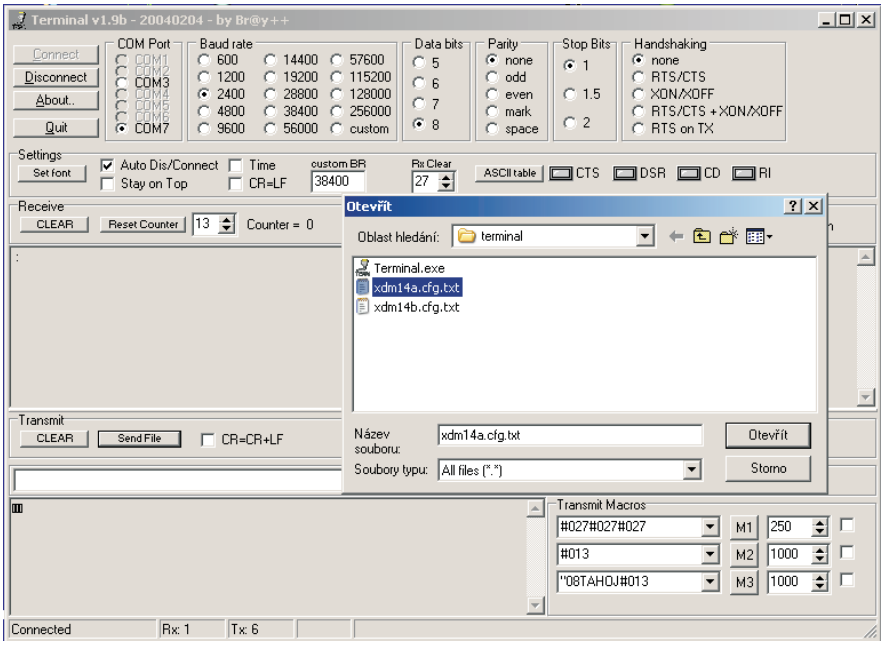
Pokud příkaz použijeme jako první příkaz v souboru, bude adresa stanice ještě 00, proto bude vypadat takto:

```
%00W2710
```

Dalším příkazem bude příkaz pro nastavení komunikačních parametrů, obecně:



Obr. 2 Výpis konfiguračních příkazů z EEPROM



Obr. 3 Výběr konfiguračního souboru pro odeslání jeho obsahu po lince do přístroje

A abychom poznali, že přístroj prošel předchozími konfiguračními příkazy, dejme na závěr příkaz:

"07TdISP

Ten zobrazí nápis **dISP** na přístroji po jeho zapnutí a po konfiguraci, a to do doby než budou data pro zobrazení přepsána novými, případně pokud nebudou přepsána do 10 sec, bude nápis nahrazen čtyřmi pomlčkami (vlastnost zobrazovače po definici pomocí W - viz manuál).

Příkaz W jsme vybrali proto, abychom se v případě poruchy komunikace s přístrojem, dozvěděli díky výpisu na displeji:

\- - - -

o tom, že komunikace neprobíhá zdárně.

Posledním příkazem v souboru by měl být příkaz pro ukončení konfiguračního režimu, a to je:

!

vykřičník. Shrnuto - příkazy:

%00W2710
%00070A0600
"07TdISP
!

uložíme do souboru, který má např. název **xdm14a.cfg** a vypneme pouze přístroj, který budeme konfigurovat (tento soubor je ke stažení na www.elsaco.cz, tak jako ostatní zde potřebné programy či soubory).

COM PC nastavíme v TEMINAL.EXE podle obr. 1, tj. 2400 Bd, bez parity, 1 stopbit.

Po zapnutí přístroje stiskneme na klávesnici PC klávesu F1 (pro makro M1), čímž vyšleme do přístroje 3x ESC.

V horním okně RECEIVE dvojtečka ohlásí přechod do konfiguračního režimu.

Klikneme na tlačítko SEND FILE, je zobrazeno dialogové okno pro určení souboru, jehož obsah chceme odeslat do přístroje viz obr. 3. Vybereme náš soubor a klinutím na tlačítko OTEVŘÍT odešleme jeho obsah po komunikační lince do přístroje.

Po příjmu souboru odpoví přístroj znakem !, který uvidíme za dvojtečkou v okně RECEIVE terminálu.

Proběhne konfigurace, což poznáme dle nápisu DISP na displeji, který cca po 10 sec nahradí pomlčky.

%aannttccff<CR>

kde

aa = původní adresa 00

nn = nová adressa 07

tt = prodleva odpovědi 0A = 10ms

cc = komun.rychlost 06 = 9600 Bd

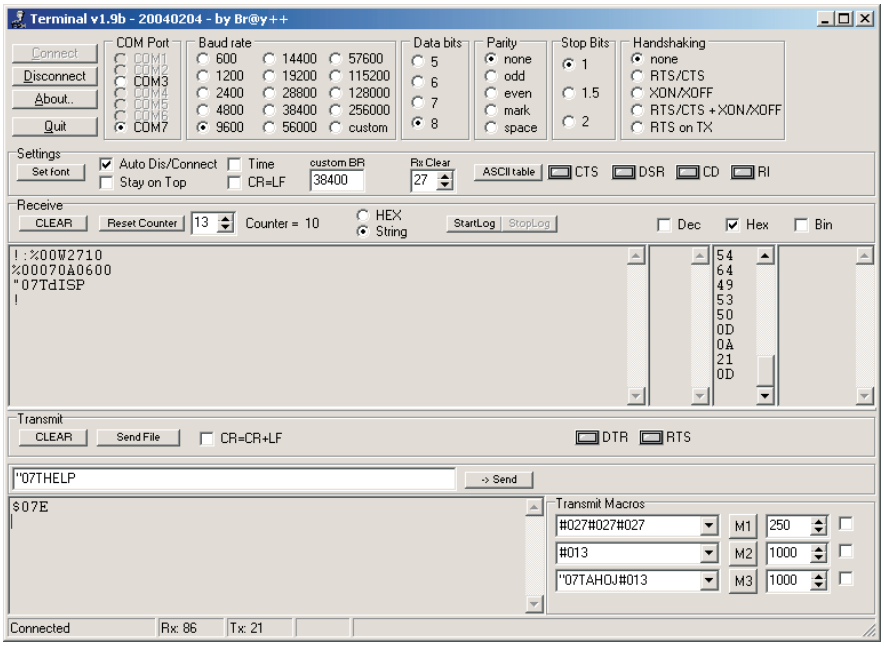
ff = konfigurační bajt 00,

tj. vše zde ovladatelné vypnuto

(viz zmíněný manuál)

Pak bude náš příkaz pro nastavení komunikace vypadat takto:

%00070A0600



Obr. 4 Vyčtení konfiguračních příkazů v normálním režimu příkazem E

Pak můžeme v okně terminálu pro definici komunikačních parametrů změnit rychlost na 9600 Bd a vyslat makro M3:

```
"07TAHOJ#013
```

Tím je zobrazen na 10 sec nápis AHOJ, který je po uplynutí nastaveného času příkazem W opět nahrazen pomlčkami.

Pokud budeme makro M3 posílat opakovaně vždy do 10 sec, bude nápis AHOJ na displeji zobrazován po tuto dobu nepřetržitě. Stačí zaškrtnout v řádku M3 políčko vpravo.

Snad toto základní seznámení s protokolem SAM spolu s uvedenými pokusy stačilo... V dalším textu počítáme s modulem XDM-14A, který je nakonfigurován podle předchozího popisu. Komunikační rychlost 9600 Bd, bez parity, 1 stopbit a adresa 07.

1.2 Práce s XDM-14A v ProgWinu

Jak připojit XDMko k sériovému kanálu centrály a jak do něj posílat změřený / vypočtený údaj z aplikace si ukážeme na dalším příkladu.

1.2.1 Fyzické zapojení

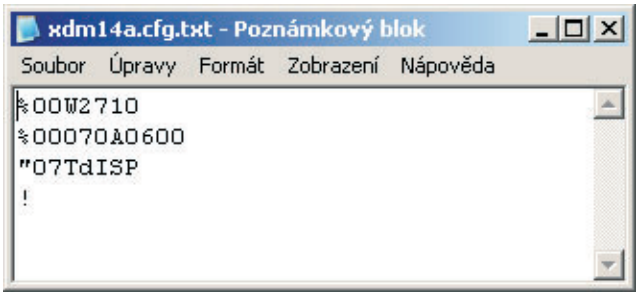
Pokud v praxi potřebujeme k centrále připojit pouze jednotku/y XDM, použijeme sériovou linku RS-422/485 s galvanickým oddělením, tedy COM1 (platí pro centrály CCPU-02/03/21). Příklad zapojení XDM k PLC naleznete v manuálu, zde se zase zaměříme na pokus "na stole" s odposlechem, aby se nám lépe ladil aplikační SW.

COM1 u centrál CCPU-03 a CCPU-21 je vyveden stejně, a to na 8pinový konektor RJ45 s tímto rozložením signálů:

COM1 RS-422	
1	+360R
2	+TxD
3	-TxD
4	GND
5	GND
6	-RxD
7	+RxD
8	-360R

Signály RxD a TxD jsou naštěstí rozvrženy na konektoru tak, že je v podstatě jedno, z které strany počítáte číslo pinu od 1. Skutečnost umístění pinu 1 ověříte měřením napětí na pinech 1 a 8, na které je vyvedeno napětí 5V DC přes odpory 360 ohmů (určeno pro připojení zakončovacích odporů linky).

V Technickém manuálu Centrální jednotky naleznete obrázky umístění konektoru RJ45 i tabulky zapojení signálů. Piny 1 a 8 jsou na obrázku označeny, pro úplnost dodáváme, že na obrázku je konektor centrály (tedy ten pevný zaletovaný do tištěného spoje) a ne konektor na kabelu! Jednoznačně doporučujeme všechny vodiče vedoucí z konektoru RJ45 na kabelu před připojením na další svorkovnici prozvonit, ušetříte se tím hodně času.



*Obr. 5 Konfigurační soubor **xdm14a.cfg***

Abychom mohli komunikační zprávy na lince RS-485 odposlouchávat na PC programem TERMINAL.EXE, musíme propojit linkou RS-485 centrálu, XDMko a převodník RS-485, připojený k PC.

Převodník RS232/RS485 je k PC připojen jako v předchzí kapitole standardním kabelem RS232.

Z předchozí kapitoly máme rovněž připojen přístroj XDM-14A ke konektoru DB15 převodníku přes pomocnou svorkovnici - např. přes tzv. čokoládu.

Pak stačí z centrály vyvést z COMu1 kabel, žíly z pinů 3+6 spojit a připojit na první svorku čokolády - spojíme tak -RxTxD všech zařízení na lince.

Dále z téhož kabelu z COMu1 centrály spojit žíly z pinů 2+7 a připojit na druhou svorku čokolády - spojíme tak +RxTxD všech zařízení na lince.

	COM1 centrála	čokoláda	XDM-14A	převodník
-RxTxD	3+6	1	2+4	4
+RxTxD	2+7	2	1+3	11

Ideální by bylo, kdyby převodník RS232/RS485 byl k PC připojen na jednom COMu PC a jiný COM PC byl věnován překladům z ProgWinu do centrály. Nebo použít stolní PC a notebook - prostě dva PC a každý aby dělal "svou" práci.

Problém je totiž v tom, že pokud budete používat jeden COM v PC k překladům a střídat ho k užití na odposlech přes převodník, bude třeba přestavit jeho komunikační parametry (rychlost). Pak se občas stane, že vám komunikace zmrzne a budete muset restartovat někdy PC, někdy centrálu, ...

1.2.2 Projekt s XDM

Máme za úkol naprogramovat aplikaci, ve které bude měřená teplota vody zobrazována nejen na displeji regulátoru, ale i na zobrazovači XDM-14A, který bude připojen na RS-485, na COM1 centrály CCPU-21.

Když spustíte ProgWin a prohlédnete si knihovnu pro PL2, XDM nenaleznete. Znáte však jeho komunikační protokol, proto nebude tak složité jej oprogramovat pomocí modulu SERIALCOMM.

Všimněme si nejprve parametrů modulu SERIALCOMM. Parametr PRIORITA ponecháme nulový, pokud bude potřeba, tak se k němu vrátíme po ladění v RUN režimu ProgWinu.

Parametr rychlost nadefinujme na 2, aby SERIALCOMM byl zpracováván programovou smyčkou každou vteřinu.

Parametr BAUDRATE nadefinujeme na 55, protože podle HELPu ProgWinu tak určíme komunikační rychlost pro vysílání i příjem na 9600 Bd - na tuto rychlost jsme XDMko nakonfigurovali dříve.

Další komunikančí parametry volíme opět podle toho, jak jsme XDMko prve nadefinovali:

PARITY = 0, tj. bez parity

FORMAT = 88, tj. osmibitová zpráva pro vysílání i pro příjem.

Parametr TIMEOUT určuje časový limit pro dobu mezi znaky v ms. TIMEOUT pro celou zprávu je pak 256 * meziznakový TIMEOUT. Přečtěte si v HELPu jak je komunikováno v praxi na lince, pokud je v projektu více modulů SERIALCOMM.

Máme-li užit SERIALCOMM ve vteřinové programové smyčce a prvním průchodem smyčkou je vyslán na linku požadavek a teprve druhým průchodem programovou smyčkou je zpracován příjem, tj. zpráva, kterou XDMko odpovídá. Proto musí celkový TIMEOUT být větší než 1000 ms, podle výše uvedeného tedy alespoň 1000/256, což po zaokrouhlení nahoru dává pro meziznakový TIMEOUT 4 ms. Parametr TIMEOUT nastavíme na 5.

Později můžete udělat v projektu pokus a nastavit TIMEOUT menší než 4 a uvidíte, že na výstupu ERROR *(Výstup ERROR [vpravo nahoře] je aktivován v případě, že na výzvu modulu SERIALCOMM nepřijde do doby TIMEOUT odpověď nebo při nesprávném formátu přijaté zprávy)* se objeví chybová 1.

Parametr COMNR dáme = 1, protože chceme použít linku RS-422/485 s GO.

Nyní se vrátíme ke komunikační zprávě.

V projektu měříme zadanou teplotu vody analogovým vstupem In1 na centrále CCPU-21. Změřená hodnota je předána na In1 centrály, odkud je vedena na vstup IN modulu SCALE, který má za úkol přepočítat měřený rozsah a výsledek předat na výstup OUT. Odtud je výstup veden na vstup Tx1 horního modulu SERIALCOMM.

Při použití čidla Ni1000 nebo Pt100 není nutno používat modul SCALE. Při testování na stole byl použit odporový analogový vstup s potenciometrem pro simulaci a ladění...

Tak máme na Tx1 přivedenu hodnotu, kterou potřebujeme zapracovat do komunikační zprávy pro XDMko.

Z manuálu pro XDM-14A vyčteme jak má vypadat zpráva pro zobrazení čísla na displeji zobrazovače:

"aaT<číslo><CR>

(viz zmíněný manuál)

aa = adresa XDMka,

tj. po předchozím nastavení 07

<číslo> může mít maximálně 4 znaky a desetinnou tečku, měříme teplotu vody, použijme tedy formát čísla XXX.X

Přečtěme si HELP k modulu SERIALCOMM pro formátování zprávy, kterou chceme vyslat jako požadavek do XDMka:

příkazy apz, bpz, cpz, dpz slouží pro vyslání reálného čísla ze vstupů Tx0..3 jako ASCII znaky na linku

a pro Tx0, b pro Tx1, c pro Tx2, d pro Tx3

p = počet znaků celkem včetně desetinné tečky

z = počet cifer za desetinnou tečkou

maximálně 3 desetinná místa a celkem 8 cifer

je-li cifer pro přenos méně než je udáno ve formátu, jsou zleva doplněny mezerami

při překoročení formátu je na linku vysláno EE.EE (počet E podle formátu)

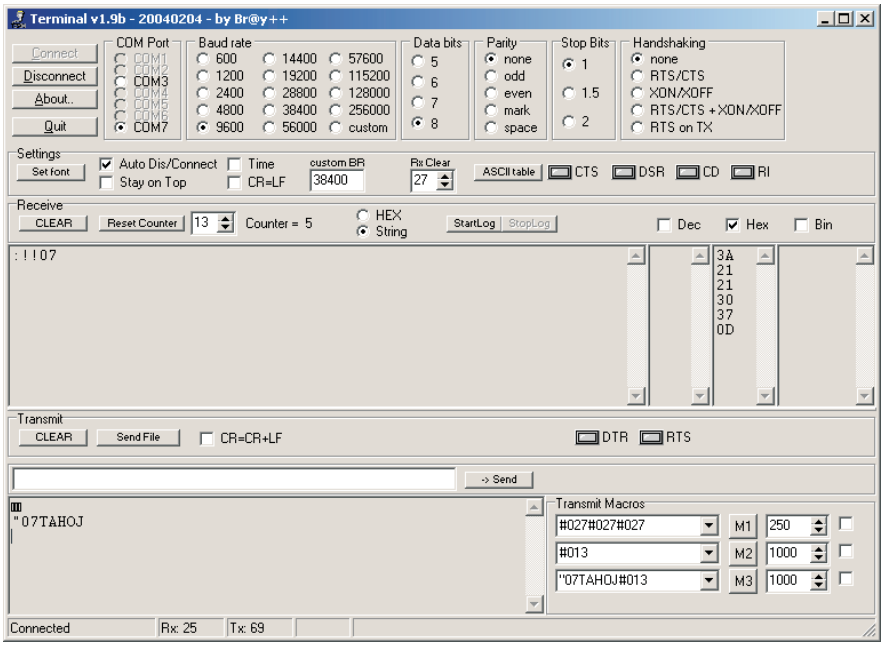
např. a52 pro vyslání hodnoty reálného čísla ze vstupu Tx0 jako ASCII znaky ve tvaru 12.45

Vypadá to na použití formátu

b51

protože jsme signál přivedli na vstup Tx1, celkem bude zobrazeno 5 znaků včetně desetinné tečky a bude nám stačit jedno desetinné místo.

Celý formátovací řetězec začíná uvozovkami, pokračuje adresou přístroje (XDMka), pak následuje znak T, za ním číslo a na závěr znak <CR> pro konec příkazu. Podle HELPu pevné znaky, které chceme na



Obr. 6 AHOJ po konfiguraci XDM-14A

linku poslat, máme uvádět ve tvaru HEXA a velkými písmeny.

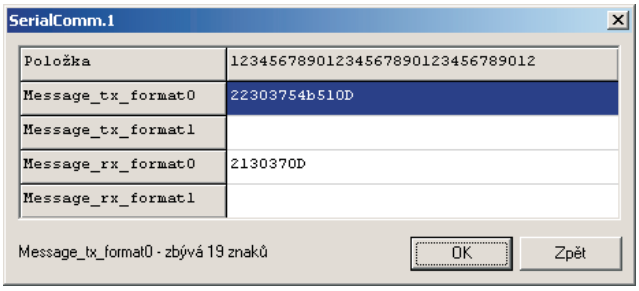
uvozovky = 22h

adresa 07 = 30h 37h

znak T = 54h

číslo formátem = b51

<CR> = 0Dh



Obr. 7 Definice zprávy pro vyslání naměřené hodnoty

Z manuálu XDM14-A lze vyčíst odpověď na tuto zprávu při správně provedeném příkazu:

!aa<CR>

Proto ji zadáme jako očekávanou odpověď do stejného modulu SERIALCOMM.

! = 21h

aa = 07 = 30h 37h

<CR> = 0Dh

Tím bychom měli mít obslouženo zobrazování teploty vody podle předchozího zadání.

A protože se chceme něco naučit, zkusme si zobrazování jetě dál vylepšit. Ať se na displeji střídá nápis VODA s měřeným údajem.

To znamená, že potřebujeme uskutečnit další komunikační relaci se stejným přístrojem (na stejné lince se stejnou adresou ...). Použijeme k tomu další modul SERIALCOMM v projektu ProgWinu. Jeho parametry nastavíme stejně jako u prvního, jen mu změníme formátováním zprávu, kterou vysílá.

Použijeme stejný typ zprávy:

"aaT<číslo><CR>

ve které <číslo> zkusíme nahradit nápisem VODA. Z manuálu XDM-14A lze totiž vyčíst, že jako <číslo> lze zobrazovat jakékoliv ASCII znaky, které jsou na sedmisegmentovém displeji interpretované.

V praxi doporučujeme vyzkoušet čitelnost (čtivost) takového nápisu přímo na displeji.

Znaky, které chceme vyslat přímo na linku opět zadáváme ve tvaru hexa:

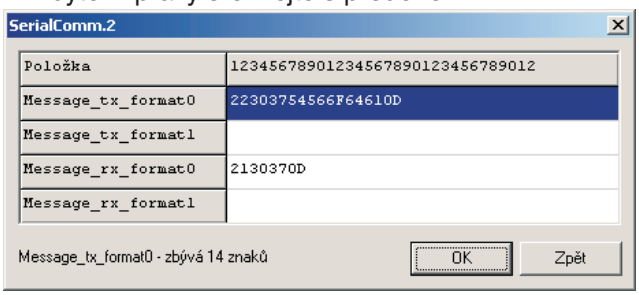
V = 56h

o = 6Fh

d = 64h

a = 61h

Zbytek zprávy srovnejte s předchozí.



Obr. 8 a pro vylepšení nápis Voda

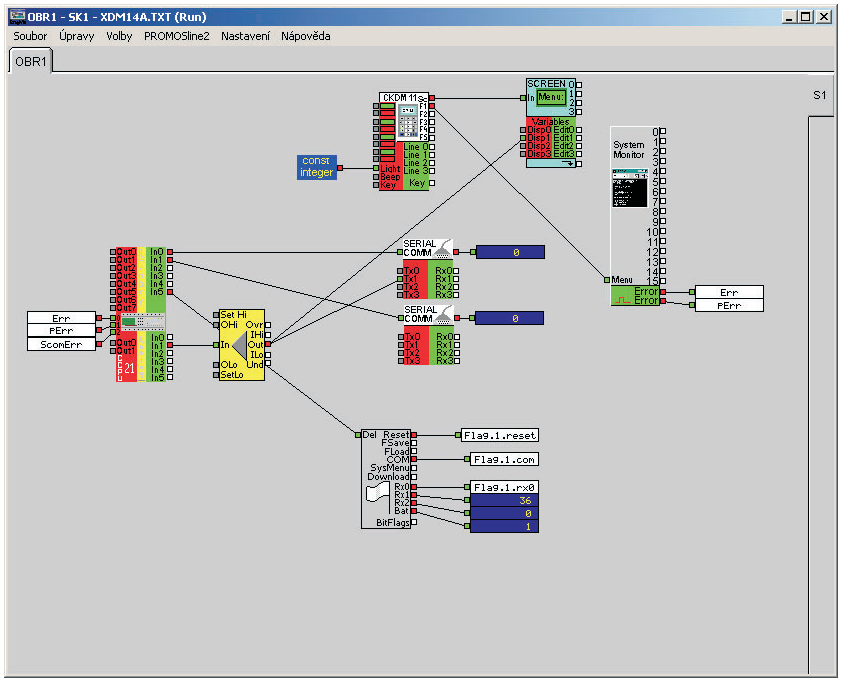
Odpověď na tuto zprávu je stejná jako na předchozí (pokud je výzva přijata a správně zpracována v XDM).

Pokud zpráva nebude zpracována správně, odpovídá XDMko trochu jinak (a sice ?aa<cr>), což modul SERIALCOMM vyhodnotí jako nesouhlas s očekávanou odpovědí (neodpovídá zadanému formátu pro odpověď) a výstup ERROR (vpravo nahoře na modulu SERIALCOMM) vystaví do jedničky.

Tím bychom měli mít projekt nachystán k překladu, zkusme se zamyslet, jak se bude chovat komunikace mezi centrálou a XDMkem.

Na linku vysíláme v podstatě dvě výzvy pro XDMko, jednu pro zobrazení hodnoty, druhou pro zobrazení nápisu VODA. Moduly SERIALCOMM dáme do programové smyčky, která je obsluhována 1x za vteřinu (parametr RYCHLOST=2). Pokud by byly zprávy vyslány těsně po sobě, viděli bychom prakticky na displeji až tu druhou. Pokud jsme si však přečetli HELP, jsme o trochu chytřejší:

Bude-li modulů SERIALCOMM v projektu více, bude se při správně nastavené prodlevě parametrem TIMEOUT vykonávat průchodem programovou smyčkou jen jeden, a to postupně po sobě podle umístění v projektu na ploše schéma, resp. podle priority modulu.



Obr. 9 Projekt s komunikací pro XDM-14A

Modul SERIALCOMM nejprve vyšle požadavek (první průchod programovou smyčkou) a potom očekává odpověď (druhý průchod programovou smyčkou), a to i v případě, že není formátovacím řetězcem předepsána. Proto pokud potřebujeme docílit rychlého vysílání samostatných zpráv modulem SERIALCOMM, musíme počítat s tím, že zpráva bude vyslána vždy ob jednu programovou smyčku, tj. např. při rychlost=3 až po 200 ms. Bude-li takto více modulů serialcomm správně po sobě navazovat, proběhne vysílání z každého z nich už po 100ms, ale při přechodu z "posledního" na "první" bude zase až za 200ms.

Protože máme zvoleny RYCHLOST=2, tj. vteřinovou programovou smyčku, a použity dva moduly SERIALCOMM pro jeden přístroj, lze očekávat, že prvním průchodem smyčkou bude zobrazen text VODA v případě, že "jeho" SERIALCOMM bude mít pro jistotu parametr PRIORITA=1, Druhým průchodem bude očekávána odpověď na VODA a zároveň zobrazena hodnota druhým modulem SERIALCOMM, který má pro jistotu parametr PRIORITA=2. Tento SERIALCOMM je poslední, proto průchodem třetí smyčkou se očekává pouze odpověď na poslední zprávu (vyslanou hodnotu) a teprve čtvrtým průchodem se vše opakuje.

Tak bude nápis VODA zobrazen jednu vteřinu a HODNOTA dvě vteřiny.

1.2.3 Překlad

Máme na stole vše zapojeno podle předchozích textů, projekt nahrán v ProgWinu (všechny soubory jsou ke stažení na www.elsaco.cz, a to jak tato Kuchařka, tak projekt, ...).

PC COM propojen s COM0 centrály, můžeme udělat překlad.

Pokud je vše zapojeno správně (XDMko včetně napájení a na lince RS-485 podle předchozích tabulek), aplikace se rozběhne, na displeji ovládacího panelu vidíte (krom data a času na prvním řádku) ve třetím řádku měřenou hodnotu teploty vody (občerstvována každých 100 ms, protože moduly před SCREENem jsou rovněž s parametrem RYCHLOST=3) a na zobrazovači XDM-14A se střídá nápis VODA (1 sec) se zobrazovanou hodnotou teplty vody (2 sec).

Ale co když na XDMku nebude nic nebo jen pomlčky?

Pak se musíme zamyslet jednak nad zapojením vodičů, jednak nad konfigurací XDM.

Pokud jsou vidět alespoň pomlčky, znamená to, že displej je aspoň správně napájen, ale nedostává komunikační zprávu pro zobrazení nebo ta zpráva je zmršená.

Potřebujeme tedy zprávy na RS-485 odposlouchávat a víme, že máme vlastně vše k tomu nachystáno.

Na lince je převodník, stranou RS-485 napojen na naši komunikační linku, po které létají zprávy (mezi centrálou a XDM), definované v ProgWinu moduly SERIALCOMM.

Stranou RS232 je převodník připojen k PC, ve kterém běží program TERMINAL.EXE, pomocí kterého jsou zprávy odposlouchávány a zobrazovány.

Ideální je, pokud tento program běží na jiném PC než na kterém provádíme překlady z ProgWinu. Nebo alespoň na jiném COMu.... O tom bylo psáno dříve, pokud použijete jedn COM pro překlady i pro odposlech, bývají občas potíže s přepínáním komunikačních vlastností tohoto COMu. Ještě by se dalo tomuto problému aspoň při ladění pomoci tím, že bychom zvolili pro překlad i pro linku RS-485 stejné komunikační parametry, tj. zejména rychlost.

Tady na papíře všechno zdárně funguje, podívejme se druhým PC na odposlech zpráv, tedy na to jak mají vypadat.

Mějme spuštěn ve druhém PC program TERMINAL.EXE, převodník zapnut a na lince. V TERMINALu nezapomeňte překontrolovat nastavení COMu PC, případně jej poopravit na:

COM Port - ten, který na PC k tomuto užíváte
Baud rate = 9600
Data bits = 8
Parity = none
Stop Bits = 1
Handshaking = none

A dodávám, že při připojení na linku musí být šedivé tlačítko vlevo nahoře CONNECT, není-li, pak na něj klikněte.

Pak bychom měli v okně RECEIVE propříjem zpráv číst:

```
"07TVoda
!07
"07T 41.4
!07
```

Ten třetí řádek berte samozřejmě s rezervou, protože nevím, co zrovna vám bude naměřeno.

Vidno na obrázku 13.

Pokud se chete přesvědčit, že je na linku vysílán i znak <CR>, tj. 0Dh, stačí v TERMINALu zaškrtnout vpravo volbu HEX a ve sloupečku pod HEX budou vypisovány znaky ve tvaru HEXA. Po každé zprávě je časová prodleva, stačíte si tedy všimnout, že ve sloupečku HEX je na konci vždy OD.

No a pokud jste si vše doporučované nestahovali z webu, či děláte další pokusy s XDMkem a nějak vás to neposloudá, máte díky odposlechu šanci chybu dohledat. V okně RECEIVE vidíte, zda na l inku opravdu posíláte to, co chcete. Pokud ano a stejně vám to nefunguje, vidíte odpověď na svůj požadavek. Podle ní pak třeba poznáte, že XDMko zprávě nerozumí a je tedy možné, že jste ji stvořili špatně. Proto se vracíte k manuálu, čtete pozorněji, zprávu opravíte ... nebo nahradíte jinou.

Doporučuji pak zvolit nějakou jednodušší zprávu, případně ji vyslat přímo z TERMINALu ap. Lze totiž vytáhnout konektor s kabelem z centrály z COMu1, nechat tak zobrazovač připojen k PC s TERMINALem a znovu zkusit poslat jednoduchou zprávu pro ujištění se, že XDMko funguje správně...

K tomuto účelu můžeme opět nadefinovat v TERMINALu nápis AHOJ - viz obrázek 6.

1.3 Jiné TERMINALy

Pokud vám program TERMINAL.EXE, který je ke stažení na našich stránkách, nevyhovuje, můžete samozřejmě používat jiný, třeba ten, na který jste zvyklý. Každý program má své výhody a nevýhody, proto zkuste TERMINALů více a vyberte si pro své používání ten, který vám nejvíce vyhovuje.

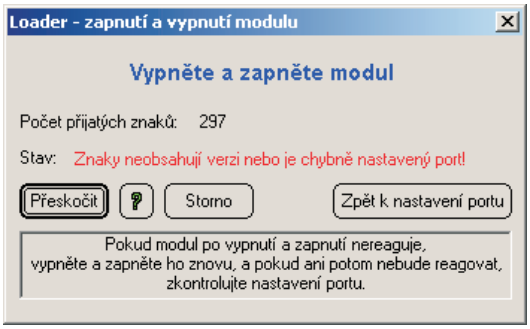
1.3.1 LOADER.EXE

ELSACO dodává (a na webu je opět volně ke stažení) program LOADER.EXE pro nahrávání firmware do jednotek systému PROMOS (pro UpGrade FW ap.). I tento program lze použít pro odposlech na komunikační lince RS-485, či jím vyslat na linku zprávu.

Po spuštění nabízí LOADER definici PC COMu, který budeme mít připojen na stranu RS-232 převodníku na RS-485.

Po případné editaci parametrů na námi požadované (jsou zobrazeny na obr. 10), schválíme klávesou OK a dostaneme se dál na specialitu tohoto programu, který byl ušit pro periferní jednotky PL2 - program očekává po zapnutí jednotky speciální zprávu. Proto nám toto očekávání napíše v dialogovém okně (obr. 11).

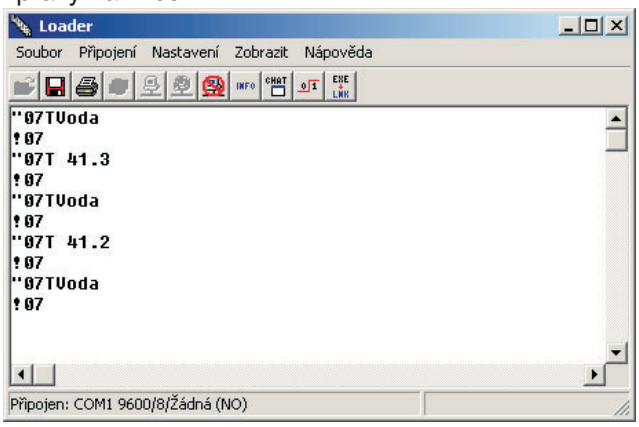
Obr. 10 Definice PC COMu pro LOADER



Obr. 11 LOADER vyžaduje spec. zprávu

A nabízí nám volbu PŘESKOČIT, kterou samozřejmě zvolíme my.

Potom se dostáváme do hlavního okna programu LOADER, ve kterém můžeme číst odposlouchávané zprávy na lince.

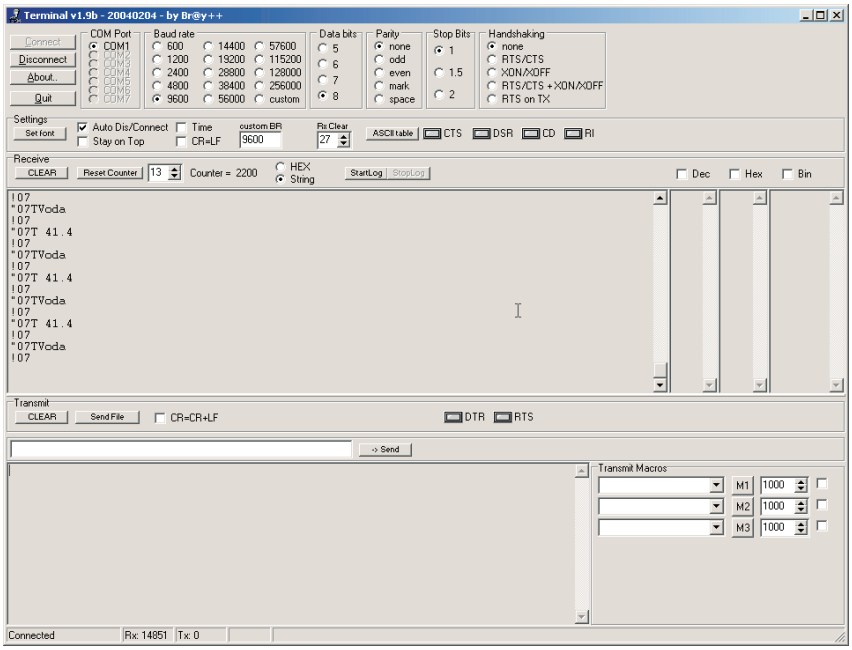


Obr. 12 Hlavní okno LOADERu

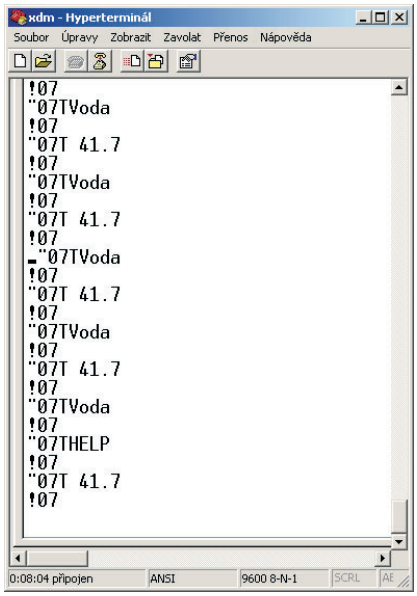
LOADER si můžete dál "osahat", obsahuje další užitečné volby. Kliknete-li na volbu CHAT, dostanete k dispozici příkazový řádek (obr.), do kterého lze zadávat příkazy, které chcete odeslat na linku. Pod příkazovým řádkem můžete zaškrtnout další užitečné volby pro tento režim, např. se nám bude hodit doplnění každé zprávy znakem <CR> pro konec zprávy.

A tak můžete v okně číst odposlouchávané zprávy na lince, ba dokonce nějakou zprávu zadat do příkazového řádku (vlevo od tlačítka ODESLAT) a vnutit ji klknutím na ODESLAT na stejnou linku, na kterou zprávy posílá centrála.

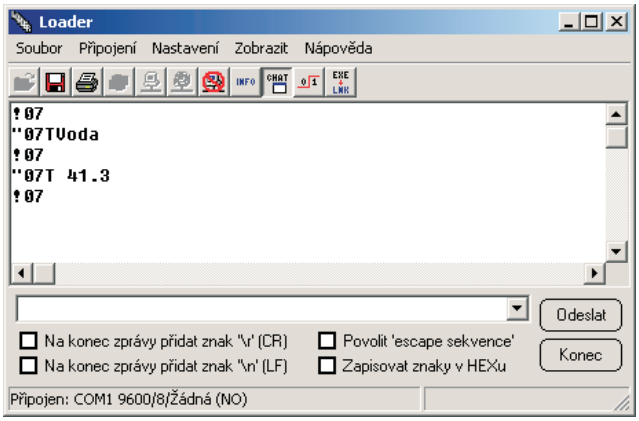
Pokud se vám to podaří, spatříte na krátkou dobu na XDMku třeba i nápis HELP, pokud potřebný příkaz správně zadáte.



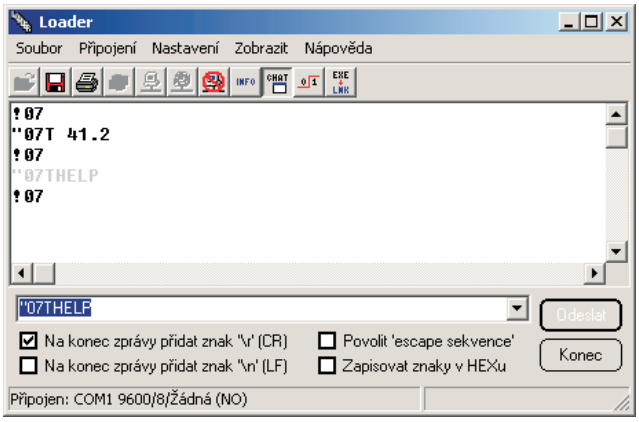
Obr. 13 Odposlech komunikace centrála <-> XDM14A na lince RS-485



Obr. 15 Okno Hyperterminálu



Obr. 14 Okno LOADERu s příkazovým řádkem



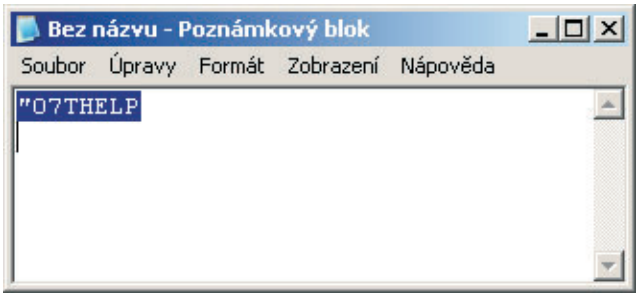
Obr. 17 LOADER - odeslání příkazu na linku

A v hlavním okně LOADERu uvidíte šedě vypsaný příkaz, který jste odeslali z příkazového řádku a za ním potvrzení (!07) - viz obr.

1.3.2 HYPERTERMINAL

Po "troše" práce s konfigurací lze k odposlechu na lince použít i HYPERTERMINAL dodávaný se systémem Windows XP.

Horší je to však u něj s vkládáním a odesíláním zpráv na linku. Připadalo mi nejjednodušší otevřít Poznámkový blok, v něm si zprávu nachystat včetně <CR>, zkopírovat do klipboardu pomocí CTRL+C a pak přenést do HYPERTERMINALU volbou Úpravy a Vložit, nebo při správné konfiguraci užití klávesy CTRL pro Windows přímo CTRL+V (obr. 16).

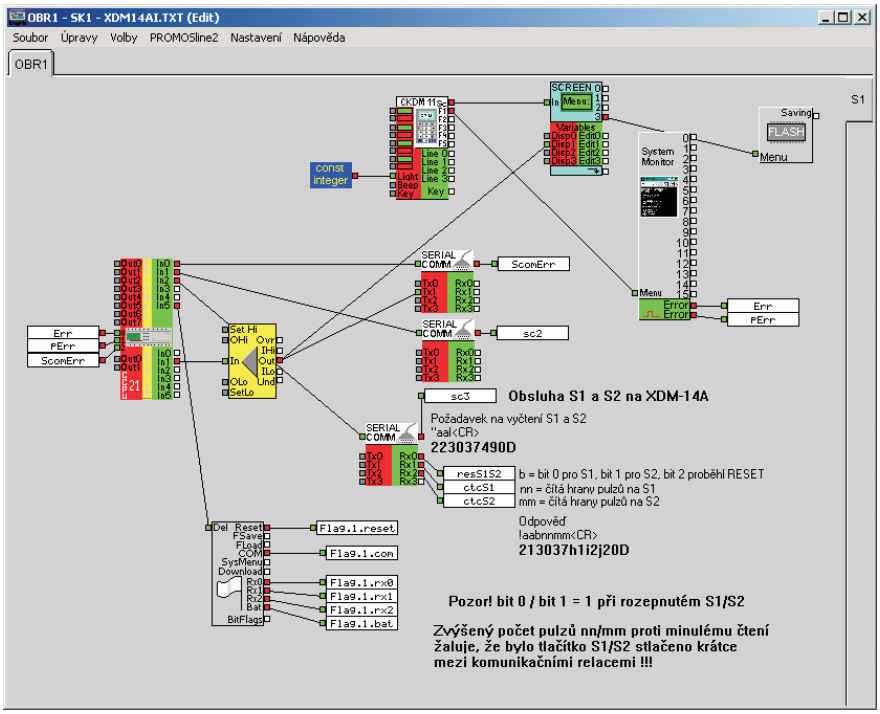


Obr. 16 Přichystání příkazu v Poznámkovém bloku

1.4 Čtení S1/S2 na XDM-14A

Vypadá to sice neuspořádaně, ale nepředpokládal jsem, že čtení stavu logických vstupů S1 a S2 bude v praxi příliš časté. A ani jsem manuál nedopsal a už jsem musel poskytovat OnLineHelp :-)

K XDM-14A lze přidat dvě tlačítka, která se zapojují vždy jednou stranou na svorku 7 - GND, druhou stranou pak pro první tlačítko S1 na svorku 5 a pro druhé tlačítko S2 na svorku 6. Již ze zapojení plyne, že svorky 5/6 tlačítky S1/S2 přizemňujeme (naměříte na nich



Obr. 18 Projekt i pro S1/S2

necelých 5V, tj. logickou jedničku), takže při nestlačených S1/S2 vyčteme "1" !

1.4.1 Relace pro čtení S1/S2

Pozor na starší manuály (chyba $aaI<CR>). Relace obecně vypadá takto:

"aaI<CR>

aa = adresa modulu

Odpověď při chybě

?aa<CR>

Správná odpověď

!aabnnmm<CR>

pro **b**

- bit 0 = stav vstupu S1
- bit 1 = stav vstupu S2
- bit 2=0 -> proběhl RESET

nn a **mm** indikují počet hran modulo 256 detekovaných na vstupech S1 (**nn**) / S2 (**mm**)

Vyhodnocení hodnot **nn**/**mm** v aplikačním programu umožňuje detekovat i krátké stlačení **S1**/**S2** mimo komunikační relaci. Stačí kontrolovat jejich nárůst mezi relacemi => bylo sepnuto.

1.4.2 Jak na S1/S2 v ProgWinu

Pro vyčtení nadefinujeme do dalšího modulu SERIALCOMM, který bude mít komunikační parametry stejné jako moduly SERIALCOMM předchozí, požadavek s adresou 07 (projekt XDM14AI.TXT):

"07I<CR>

223037490D

To by neměl být problém. Pro formát odpovědi se musíme trochu zamyslet. Očekáváme příjem celkem pěti bajtů, lepší je však je už v modulu SERIALCOMM rozebrat podle obecné odpovědi na tři části. Očekáváme jednak příjem jednoho bajtu b pro vyčtení stavů S1/S2, jednak příjem dvou dvoubajtů nn a mm pro vyčtení počtu napočítaných hran (pulzů) od S1 a S2.

Proto pomocí HELPu:

- příkaz hn až kn (malá písmena h, i, j, k s cifrou) = přečte n znaků ve formátu ASCII hexa a uloží jako hodnotu integer do výstupu Rx0..3

použijeme příslušný formátovací řetězec pro příjem:

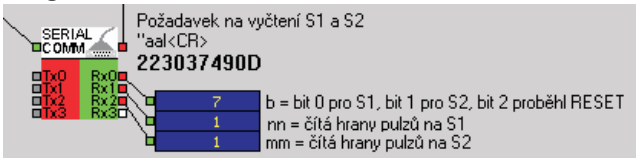
!aabnnmm<CR>

213037h1i2j20D

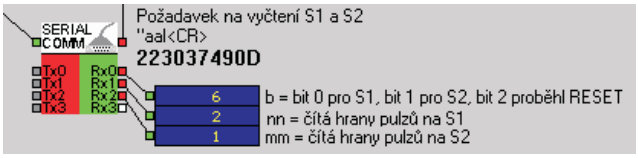
a na výstupu **Rx0** budeme očekávat vyčtený stav vstupů **S1** / **S2**, na výstupu **Rx1** hodnotu **nn** a na výstupu **Rx2** hodnotu **mm**.

Viz obrázek 18, spodní modul SERIALCOMM a poznámky v projektu. Nezpomeňte, že stav S1 i S2 (nakonec i RESET jako bit 2) je v klidu vždy jedničkový, takže na Rx0 očekávejte v klidu hodnotu 7!

Pokud chcete však vyhodnocovat stisk S1/S2 prakticky, vycházejte z hodnot nn a mm, a to z jejich navýšení. Činnost S1/S2 si vyzkoušejte v RUN režimu v ProgWinu:



A po stlačení a podržení S1:



Po těchto pokusech si již zajisté dál poradíte...

2 Modul SAM-02 (4x DI/CTC + 4x DO-relé)

2.1 Úvodem

Modul SAM-02 slouží pro vzdálené připojení (po RS-422/485) 4 logických (digitálních/binárních) vstupů (s možností využití režimu čítání pro každý vstup) a 4 relé. Na komunikační lince je užit firemní protokol SAM, podobný protokolu ADAM.

V době těsně po **každém** zapnutí přístroje (cca 1,5 sec) očekává přístroj sekvenci tří po sobě jdoucích znaků ESC (escape, kód 27 [dekadicky]). Tyto znaky očekává na svém sériovém kanálu, nastaveném těsně po každém zapnutí na

2400 Bd

bez parity

1 stop bit

a adresa přístroje je **00** !

Pokud je dostane, přechází přístroj do tzv. konfiguračního režimu.

Pokud je nedostane, pak přístroj provede příkazy, které má po konfiguračním režimu (po konfiguraci) uložené v paměti EEPROM. Tím se po každém zapnutí přístroj nakonfiguruje podle toho, co má v této paměti uloženo, tj. teprve potom platí pro přístroj např. jiná komunikační adresa (než 00 po zapnutí), jiná komunikační rychlost (než 2400 Bd po zapnutí), atd. ...

2.1.1 Propojení PC se SAM-02

Protože se jedná o stejný komunikační protokol a linku RS-485, bude mnoho věcí stejných jako u XDM-14A v předchozí kapitole. Proto čtěte kap. 1.1.1 a SAM-02 připojte k RS-485 obdobně, a to podle této tabulky:

převodník - DB15		svorkovnice SAM-02	
4	-RxTxD	4	-RxD
		2	-TxD
11	+RxTxD	5	+RxD
		3	+TxD

Napájení modulu SAM-02 zapojte na jeho svorky 6 (-) a 7 (+), hodnotu napájení volte podle údaje na štítku modulu.

Správně zapojený modul zkusíme nakonfigurovat, a to obdobným způsobem jako XDM-14A.

2.1.2 Přechod do konfiguračnígo režimu

Poznámka: výrobce nedávno na svých stránkách zveřejnil program TESTER pro konfiguraci a testování výrobků s komunikací. Součástí TESTERu je i HELP.

V PC spustíme TERMINAL, nastavíme komunikační parametry pro konfiguraci přístrojů s protokolem SAM vždy takto:

komunikační rychlost (Baud rate) na **2400** Bd

Data bits na **8**

Parity na **none**

Stop Bits na **1**

Handshaking na **none**

Zkontrolujeme, zda máme zvolen v PC správný COM, a to tak, že zkusmo napíšeme libovolný znak do okna TERMINALu pro vysílání. LEDky na převodníku RS232/485 musí bliknout.

Na obr. 1 sice vidíte nastavení programu TERMINAL.EXE pro konfiguraci přístroje XDM-14A, ale toto nastavení je stejné pro všechny přístroje s protokolem SAM, tedy i pro SAM-02 (myšleny komunikační parametry a makro M1). Všimněte si vpravo dole rámečku s označením TRANSMIT MACROS a definice jednotlivých maker. Jako M1 vidíte sekvenci tří znaků ESC (kód znaku ESC = 027 dekadicky). A opět právě tuto sekvenci potřebujeme pro přechod SAMu do konfiguračního režimu.

SAM-02 máme vypnut a připojen podle kap. 2.1.1 na RS-485. TERMINAL v PC spuštěn a nastaven podle předchozího. Víme, že klávesou F1 odešleme sekvenci 3x ESC na RS-485, proto zapneme SAM-02 a stiskneme klávesu F1.

V horním okně TERMINALu pro příjem (RECEIVE) se objeví dvojtečka, která nám oznamuje, že jsme se právě dostali do konfiguračního režimu.

Pokud máte s přechodem do konf. režimu potíže, zkontrolujte celé zapojení, ověřte činnost převodníku a vše opakujte i s tím, že klávesu F1 můžete během 1,5 sec, kdy na ni SAM čeká, stisknout vícekrát.

Pokud dvojtečku v okně RECEIVE spatříte, napište do okna pro vysílání znaků (TRANSMIT) příkaz

```
/?
```

pro výpis jména přístroje a verze FW v něm. Výsledek vidíte i na obr. 19:

```
/SAM-02*20000710
```

Příkaz

```
??
```

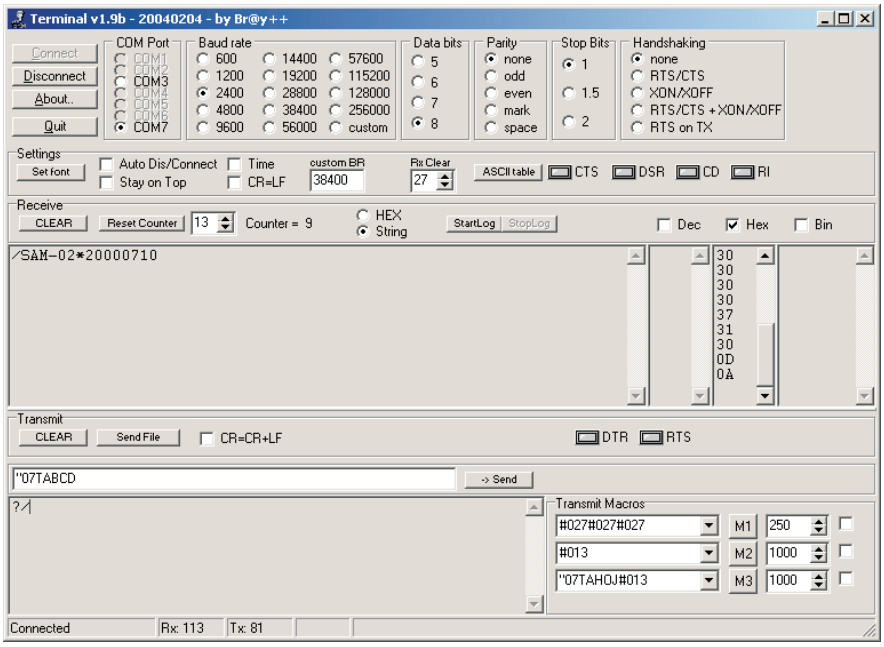
vypíše všechny konfigurační příkazy uložené v jeho paměti EEPROM/FLASH. Viz obr. 20.

Z těchto příkazů poznáte, zda byl přístroj konfigurován a jak, nebo pokud neobsahuje nic (jen !), pak je platné základní nastavení (jako tu první vteřinu po zapnutí, tj. 2400 Bd, bez parity, 1 stopbit, adresa 00). Pokud nějaké příkazy v tomto výpisu vidíte, musíte je podle manuálu rozlousknout a tak se dozvědět adresu přístroje, komunikační rychlost atd...

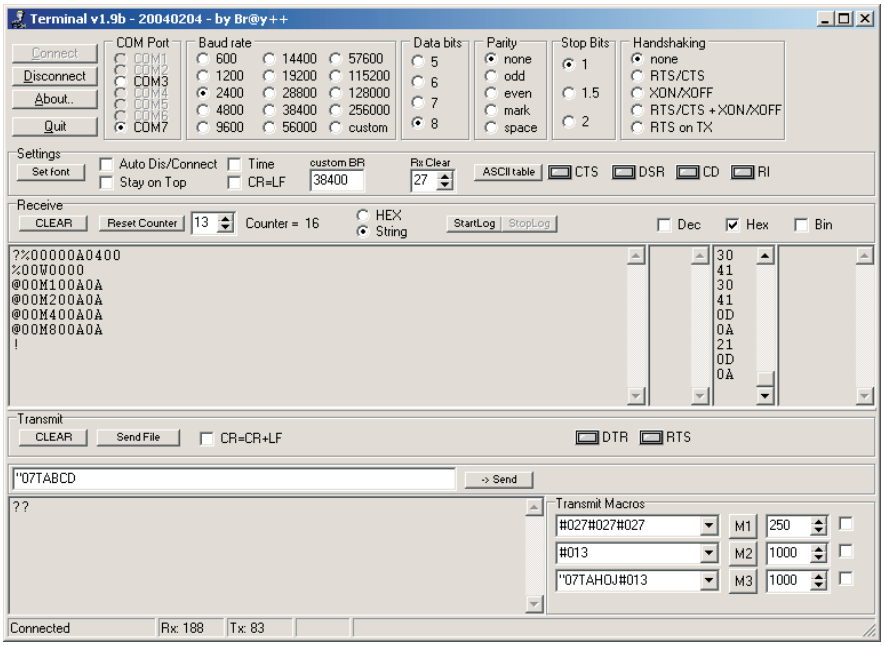
Ale nakonec proč bychom to louskali a podřizovali se?! Daleko jednodušší je nachystat si požadovanou konfiguraci do souboru příkazů a původní konfiguraci v přístroji prostě přepsat vlastní !

2.1.3 Konfigurace SAM-02

Nejprve se musíme zamyslet, nač budeme SAM-02 používat a z toho by pak mělo vyplynout, co všechno budeme potřebovat nakonfigurovat. Asi by to chtělo začít manuálem protokolu SAM s ohledem na modul



Obr. 19 Dotaz na jméno přístroje v konfiguračním režimu i s odpovědí



Obr. 20 Výpis obsahu paměti EEPROM v SAM-02

SAM-02 a z něj se dozvědět, co všechno se dá na SAM-02 konfigurovat.

Určitě budeme chtít nastavit komunikační parametry, zvolme

adresu modulu 06

komunikační rychlost 9600 Bd

prodlevu na odpověď 10 ms

ostatní = 0

K tomu slouží podle manuálu příkaz

%aannttccff<CR>

který podle uvedených požadavků bude vypadat:

%00060A0600<CR>

Pak budeme chtít modul použít na akci, kterou budeme programovat v ProgWinu, s modulem SAM-02 bude komunikovat centrála, komunikaci určíme v PW modulem SERIALCOMM, který bude programově obsluhován každou vteřinu.

I z tohoto důvodu nastavme pro SAM-02 WatchDog Timer na 10 sec. Pokud bude prakticky do SAM-02 pomocí komunikace po dobu kratší než je zvolených 10 sec nastavován stav logických výstupů - relé, bude LED s označením **Blok** zhasnuta, pokud bude komunikace pro nastavení výstupů přerušeno, LED Blok se po 10-ti vteřinách po ztrátě komunikace rozsvítí.

Příkaz pro definici WatchDogu:

%aaWnnnn<CR>

upravíme na

%06W2710<CR>

protože SAM-02 má adresu 06 a 10 000 ms vyjádřeno hexa je 2710.

Dále má modul SAM-02 4 logické vstupy, jejichž obvody a FW umožňují nastavit vstupní filtry, a to každý zvlášť. Lze nadefinovat polaritu čítací hrany, minimální délku pulzu v úrovni H i L, a to příkazem

@aaMbphhll<CR>

b = bity odpovídající vstupům

bit 0 -> vstup X0

bit 1 -> vstup X1

bit 2 -> vstup X2

bit 3 -> vstup X3

Tento příkaz použijeme pro každý vstup, lišit se budou určitě právě b-éčkem.

p = polarita čítací hrany

0 - vzestupná

1- sestupná

A opět každý bit odpovídá jednomu vstupu, čili bitem 0 nastavujeme vstup X0, ...

hh = minimální požadovaná šířka pulzu v úrovni H v ms, 01-FF odpovídá 1 - 255 ms, 00 = 256 ms

ll = minimální požadovaná šířka pulzu v úrovni L v ms, 01-FF odpovídá 1 - 255 ms, 00 = 256 ms

Zapíšeme tedy definici pro čítání vzestupnou hranou a pro šířku H/L = 10ms/10ms takto:

@06M100A0A

@06M200A0A

@06M400A0A

@06M800A0A

A jestli jsme četli manuál pro Komunikační protokol SAM pořádně, tak by se nemělo dát pro SAM-02 nastavovat nic dalšího.

Všechny tyto příkazy uložme do souboru SAM-02a.cfg:

```
%00060A0600
%06W2710
@06M100A0A
@06M200A0A
@06M400A0A
@06M800A0A
!
```

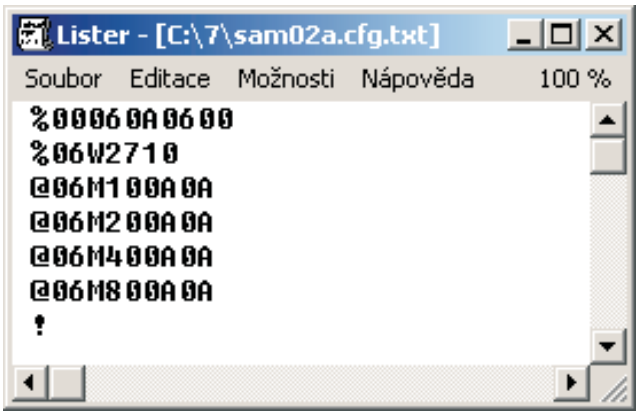
Na konci nezapomeňte na !, který ukončuje konfigurační režim.

Ještě bych upozornil na skutečnost, že pokud setrváte v konfiguračním režimu nečinně příliš dlouho, tj. pokud 5 minut nepřijme SAM-02 žádný znak, přechází automaticky do normálního provozního režimu.

Pokud tedy čtete tuto Kuchařku a zároveň k ní ještě např. manuál Komunikační protokol SAM a zároveň děláte s modulem pokusy, může uvedených 5 minut beze znaku uplynout.... Pak musíte přejít do konfiguračního režimu znovu po zapnutí modulu ...

Jestli ještě jste v konfiguračním režimu poznáte snad pouze pomocí příkazů /? a ??. Ty mají odezvu. Pozor, jiné příkazy se budou zapisovat do paměti bez odezvy, provádíte nebo kazíte tak konfiguraci modulu.

Pokud nemáte jistotu, co se vlastně s modulem děje, je nejvhodnější si nachystat uvedený (nebo obdobný) soubor s konfigurací pro SAM-02, modul vypnout a pokračovat podle dalšího popisu.



COM PC nastavíme v TEMINAL.EXE podle obr. 1, tj. 2400 Bd, bez parity, 1 stopbit.

Po zapnutí přístroje stiskneme na klávesnici PC klávesu F1 (pro makro M1), čímž vyšleme do přístroje 3x ESC.

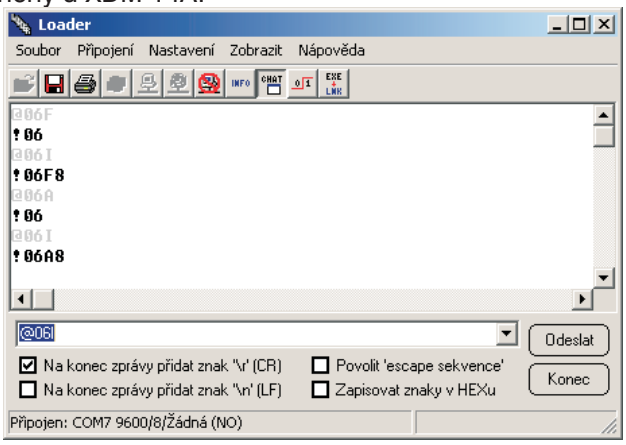
V horním okně RECEIVE dvojtečka ohlásí přechod do konfiguračního režimu.

Klineme na tlačítko SEND FILE, je zobrazeno dialogové okno pro určení souboru, jehož obsah chceme odeslat do přístroje (pro XDM-14A viz obr. 3 - nyní vybereme soubor pro SAM-02). Vybereme soubor pro SAM-02 a klinutím na tlačítko OTEVŘÍT odešleme jeho obsah po komunikační lince do přístroje.

Po příjmu souboru odpoví přístroj znakem !, který uvidíme za dvojtečkou v okně RECEIVE terminálu.

Konfigurace je hotova.

Nyní můžeme ověřit, že se SAM-02 hlásí pod novou adresou a na rychlosti 9600 Bd. K tomu lze použít jednak příkazy, které vypíší nastavení ($062<CR>), obsah EEPROMky ($06E<CR>), ... Lze však použít i příkazy pro čtení stavu vstupů či ovládání výstupů modulu SAM-02. Můžeme k tomu používat dál TERMINAL.EXE nebo jiný terminál, např. LOADER.EXE, zmíněný u XDM-14A.



V LOADERu zadáváme příkazy přes řádek CHATu, odesíláme je tlačítkem ODESLAT. Výhodou je, že v jednom okně vidíme jak znaky, které vysíláme (šedě), tak znaky, kterými modul odpovídá.

Na obrázku vidíte, že byl nejprve užit příkaz

```
@06F
```

pro sepnutí všech 4 relé modulu (odpověď !06 značí vše OK), poté příkaz

```
@06I
```

pro vyčtení stavu vstupů. Odpověď

```
!06F8
```

znamená OK, F = všechna 4 relé sepnuta, 8 pak, že je vstup X3 (tj. bit 3 s váhou 8) v jedničce, ostatní vstupy v nule.

Podobně lze sepnout třeba 2 relé (váhy bitů 8 + 2 = Ahexa) a zpět přečíst stav výstupů i vstupů.

Pro úplnost - čtení okamžitých stavů je s operačním znakem "I", čtení filtrované hodnoty vstupů provádíme pomocí "Y".

Stav čítačů lze vyčíst příkazem

```
@aaPb<CR>
```

b = opět bitově jednotlivé vstupy

b=F pro všechny 4

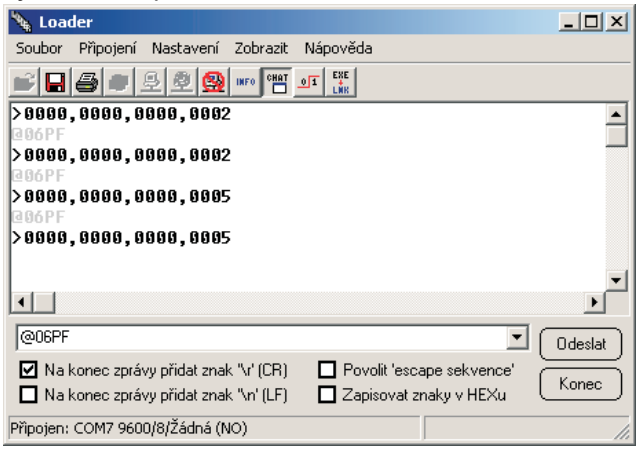
pak

```
@06PF<CR>
```

vrátí stavy všech čítačů ze SAM-02, např.

```
>0000,0000,0000,0002
```

Bližší opět v manuálu Komunikační protokol SAM, vyzkoušeno pomocí LOADERu - viz další obrázek.



2.2 SAM-02 v ProgWinu

V této kapitole si ukážeme jak připojit modul SAM-02 k centrále systému PL2 a jak ho programově ovládat pomocí ProgWinu.

2.2.1 Fyzické připojení

Pokud v praxi potřebujeme k centrále připojit pouze modul/y SAM, použijeme sériovou linku RS-422/485 s galvanickým oddělením, tedy COM1 (platí pro centrály CCPU-02/03/21). Zapojení modulů SAM k PLC je obdobné jako připojování modulů XDM-14A k PLC - viz kap. 1.2.1. Zde se zase zaměříme na pokus "na stole" s odposlechem, aby se nám lépe ladil aplikační SW.

COM1 u centrál CCPU-03 a CCPU-21 je vyveden stejně, a to na 8pinový konektor RJ45 s tímto rozložením signálů:

COM1 RS-422	
1	+360R
2	+TxD
3	-TxD
4	GND
5	GND
6	-RxD
7	+RxD
8	-360R

Signály RxD a TxD jsou naštěstí rozvrženy na konektoru tak, že je v podstatě jedno, z které strany počítáte číslo pinu od 1. Skutečnost umístění pinu 1 ověříte měřením napětí na pinech 1 a 8, na které je vyvedeno napětí 5V DC přes odpory 360 ohmů (určeno pro připojení zakončovacích odporů linky).

V Technickém manuálu Centrální jednotky naleznete obrázky umístění konektoru RJ45 i tabulky zapojení signálů. Piny 1 a 8 jsou na obrázku označeny, pro úplnost dodáváme, že na obrázku je konektor centrály (tedy ten pevný zaletovaný do tištěného spoje) a ne konektor na kabelu! Jednoznačně doporučujeme všechny vodiče vedoucí z konektoru RJ45 na kabelu před připojením na další svorkovnici prozvonit, ušetříte se tím hodně času.

Abychom mohli komunikační zprávy na lince RS-485 odposlouchávat na PC programem TERMINAL.EXE, musíme propojit linkou RS-485 centrálu, SAM a převodník RS-485, připojený k PC.

Převodník RS232/RS485 je k PC připojen jako v předchozích kapitolách standardním kabelem RS232.

Z předchozí kapitoly máme rovněž připojen modul SAMO-02 ke konektoru DB15 převodníku přes pomocnou svorkovnici (přes tzv. čokoládu).

Pak stačí z centrály vyvést z COMu1 kabel, žíly z pinů 3+6 spojit a připojit na první svorku čokolády - spojíme tak -RxTxD všech zařízení na lince.

Dále z téhož kabelu z COMu1 centrály spojit žíly z pinů 2+7 a připojit na druhou svorku čokolády - spojíme tak +RxTxD všech zařízení na lince.

	COM1 centrála	čokoláda	SAM-02	převodník
-RxTxD	3+6	1	2+4	4
+RxTxD	2+7	2	3+5	11

Ideální by bylo, kdyby převodník RS232/RS485 byl k PC připojen na jednom COMu PC a jiný COM PC byl věnován překladům z ProgWinu do centrály. Nebo použít stolní PC a notebook - prostě dva PC a každý aby dělal "svou" práci.

Problém je totiž v tom, že pokud budete používat jeden COM v PC k překladům a střídat ho k užití na odposlech přes převodník, bude třeba přestavit jeho komunikační parametry (rychlost). Pak se občas stane, že vám komunikace zmrzne a budete muset restartovat někdy PC, někdy centrálu, ...

2.2.2 Projekt s modulem SAM-02

Vytvořme projekt, který bude umět základní ovládání modulu SAM-02. Bude umět vyčíst stav jeho 4 logických vstupů (po filtraci vstupního signálu, což je dáno jednak konfigurací modulu a při vyčítání operačním znakem Y a ne I), dále bude umět ovládat 4 relé na modulu. Sice jednoduché, ale pro pochopení postačující.

Modul SAM-02 nenaleznete v knihovně ProgWinu, naleznete však opět modul SERIALCOMM, pomocí kterého dokážene do modulu SAM-02 poslat požadavek na vyčtení stavů nebo na ovládání relé pomocí komunikační relace.

Takže se nejdříve zaměříme na modul SAM-02. Jeho konfiguraci jsme provedli v předchozí kap. 2.1.3. Modul má tedy adresu 06, počítá s komunikační rychlostí 9600 Bd, s 8mibitovými daty, bez parity a s jedním stopbitem.

Pro nastavení stavů výstupů (relé) slouží relace:

@aan<CR>

kde **n** je číslo, odpovídající po bitech výstupům Y. Bit 0 pro Y0, ... bit 3 pro Y3.

Dosadíme-li adresu **06**, bude požadavek na nastavení výstupů/relé na modulu SAM-02 vypadat:

@06n<CR>

kde místo **n** dosadíme číslo podle požadavku na sepnutí příslušných relé.

Budeme-li chtít sepnout jenom relé Y3, bude číslo **n** mít hodnotu pouze váhy bitu 3, což je 8.

Pokud budeme chtít sepnout relé Y0 a Y3 současně, sečteme pro číslo n váhy příslušných bitů 0 a 3, tj. 1+8=9 a v relaci pošleme n=9 (jako hexabajt 39).

Protože budeme chtít číslo **n** měnit, musíme pro modul SERIALCOMM nalézt jednak vhodný formát pro definici požadavku, který chceme vyslat, jednak zabezpečit vytvoření hodnoty čísla **n** a tuto hodnotu přivést na některý ze vstupů modulu SERIALCOMM.

Pro školní příklad stačí vytvářet číslo **n** pomocí generátoru pulzů PPG a čítače CNT. Viz zapojení v projektu, který je stažení na www.elsaco.cz. Tam hodnota na výstupu CNT nabývá postupně 0 - 15 (modulo 1).

V praxi přiveďte logické signály postupně na modul BINtoINT, tím přiřadíte každému logickému signálu bitovou váhu a výsledné číslo **n** získáte na výstupu **Int**.

Hodnotu čísla n přivedeme na modul SERIALCOMM, a to na vstup Tx0.

V HELPu ProgWinu nalezneme nápovědu pro modul SERIALCOMM:

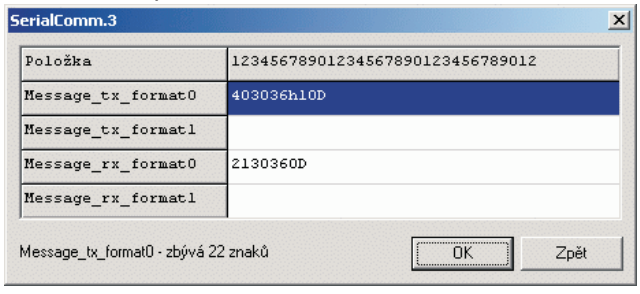
- příkazy hn, in, jn, kn slouží pro vyslání integer hodnoty ze vstupů Tx0..3 jako n znaků ve formátu ASCII hexa

Podle toho upravíme zápis požadované relace tak, že místo čísla **n** dosadíme formát **h1** (jedná se o jeden bajt a vstup Tx0). Nesmíme opomenout, že zápis formátu do dialogového okna modulu SERIALCOMM provedeme ve formátu ASCII hexa. Na následujících řádcích vyjádříme zápis nejprve ASCII, poté ASCII hexa:

@06h1<CR>

403036h10D

Znaky h1 zachováme, protože se jedná o formátovací řetězec pro modul SERIALCOMM:



A pro pořádek doplníme do parametru Message_rx_format0 formát zprávy pro očekávanou odpověď. Ta podle manuálu má mít obecný tvar:

!aa<CR>

což ve tvaru ASCII hexa vypadá takto:

2130360D

Pokud bude vrácena jiná zpráva, bude vystaven chybový výstup modulu SERIALCOMM (vpravo nahoře) do jedničky. Jinak setrvá v nule.

Tímto způsobem ovládneme výstupy modulu SAM-02. Vytvořme nyní pomocí dalšího modulu SERAILCOMM další relaci pro vyčtení stavů filtrovaných vstupů. Způsob filtrování vstupů (10 ms) jsme určili konfigurací modulu SAM-02. Pro vyčtení filtrovaných vstupů použijeme podle manuálu Komunikančí protokol SAM operační znak Y:

@aaY<CR>

V odpovědi však dostaneme nejen požadovaný stav vstupů, ale i vyčtený stav výstupů:

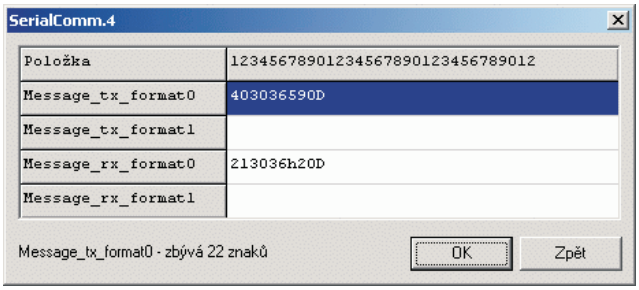
!aaoi<CR>

kde vyšší bajt **o** určuje aktuální stav výstupů (opět po bitech, tj. bit 0 odpovídá stavu Y0, ... bit 3 pak stavu Y3) a nižší bajt **i** odpovídá vyčtenému stavu filtrovaných vstupů, a to opět po bitech, tj. bit 0 odpovídá stavu vstupu X0,... bit 3 pak X3.

Požadavek bude tedy vypadat:

@06Y<CR>

403036590D



Obr. 21 Formát pro vyčtení stavů vstupů

A podle HELPu zformátujeme i odpověď:

- příkaz hn až kn (malá písmena h, i, j, k s cifrou) = přečte n znaků ve formátu ASCII hexa a uloží jako hodnotu integer do výstupu Rx0..3

Jen připomeňme, že nyní očekáváme 2 bajty (oi). Proto obecný tvar upravíme:

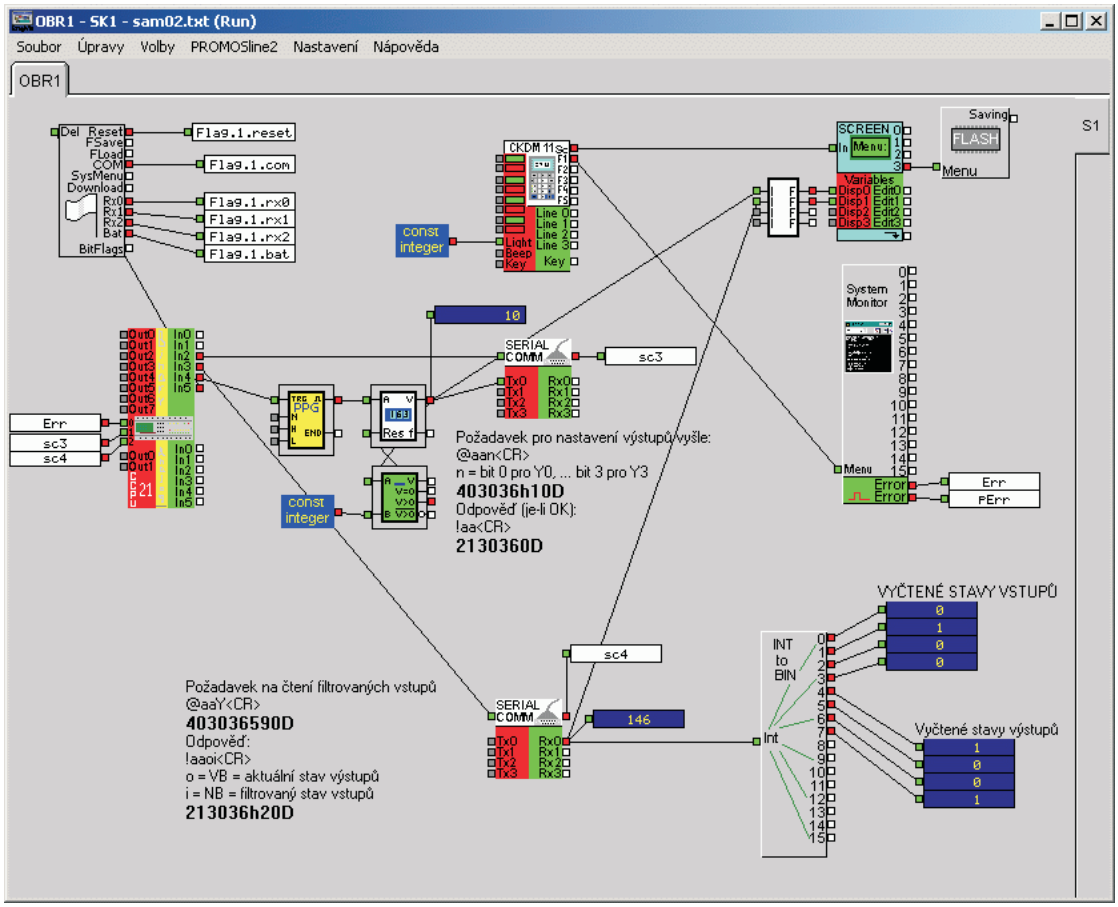
!06h2<CR>

213036h20D

Zkušební projekt máme nachystán z webu, parametry SERIALCOMMu vysvětleny, můžeme přejít k překladu a zkoušení. Ještě jen poznámka pro ty, které zaráží, že v projektu je užita centrála CCPU-21, kterou nemají zrovna k dispozici. Stačí ji v projektu nahradit jinou, případně vazby na její logické vstupy nahradit konstantami typu integer (ty lze v RUNu ovládat visualmodulem TLACITKO).

2.2.3 Překlad, ladění

Prohlédněte si projekt, na spodní modul SERIALCOMM pro příjem stavu vstupů / výstupů je napojen modul **INTtoBIN** a na jeho výstupech získáváme přímo stavy vstupů **X0..3**, resp. výstupů **Y0..3**. Proveďte překlad a pokud bude vše OK, můžete v RUN režimu ProgWinu sledovat stavy i/o na SAM-02.



Obr. 22 Projekt pro modul SAM-02

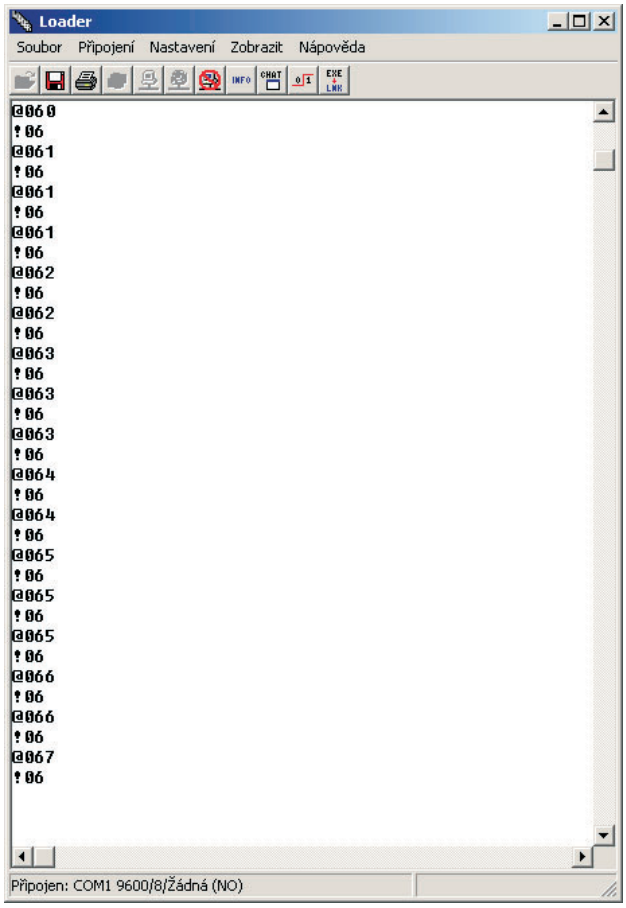
Pokud přece jenom něco nefunguje, zkontrolujte nastavení komunikačních parametrů v modulech SERIALCOMM, zda jejich definice odpovídá naší dřívější konfiguraci (9600 Bd, 8 datových bitů, 1 stopbit, bez parity).

Můžeme si opět pomoci odposlechem linky RS-485, např. pomoci LOADERu. Relace jednoho modulu SERIALCOMM lze zakazovat/povolovat logickým signálem na vstupu STROBE (u modulu vlevo nahoře).

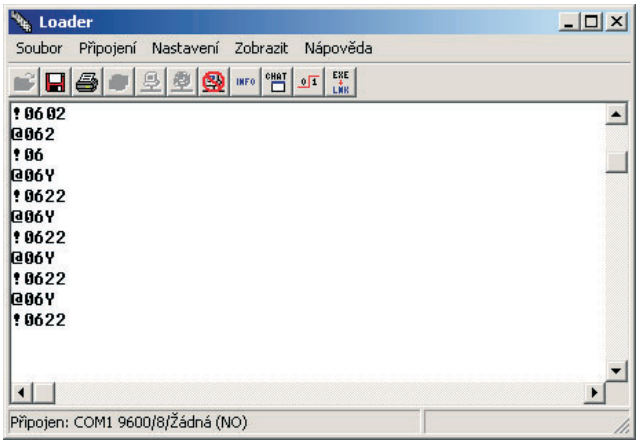
Nechejme povolenou činnost hornímu modulu SERAILCOMM, který ovládá 4 relé modulu SAM-02. V okně LOADERu bychom měli vidět následující komunikaci - viz obr. 23.

Ve vašem konkrétním případě zjistíte, zda vám na lince probíhají vůbec njaké relace, případně jaké, zkontrolujete nastavení COMu, případně zda požadavek vypadá podle definice. Je-li vše OK pak ještě sledujte, zda odpověď je podle manuálu na počátku s ! (to by bylo OK) nebo s ? (přístroj odpovídá chybou).

Podobně lze zkusit i druhou relaci samostatně, a to přivedením jedničky na STROBE pro horní SERIALCOMM a nuly na STROBE pro dolní SERIALCOMM. Pak budou na lince RS-485 pouze relace pro vyčtení stavů i/o (s operačním znakem Y) a odpověď na ni. To je vidět na obrázku 24.

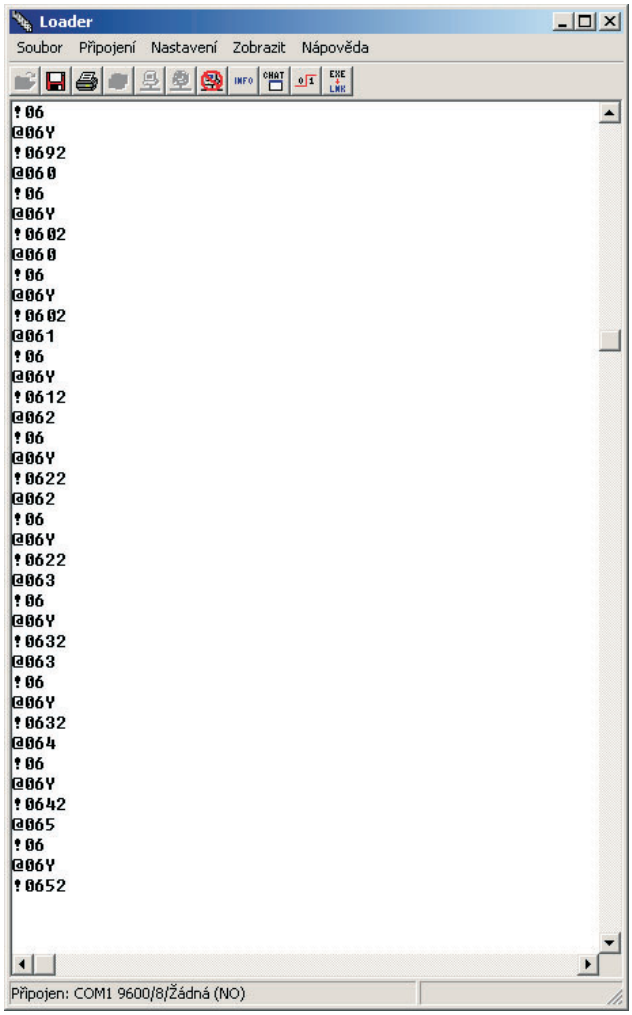


Obr. 23 Relace SAM-02 pro ovládání OUTů



Obr. 24 Relace SAM-02 pro čtení stavů i/o

Pokud na oba vstupy STROBE přivedete nulu, budou v činnosti oba moduly SERIALCOMM a relace na lince budou vypadat např. jako na obrázku 25.



Obr. 25 Relace SAM-02 - oba druhy

3 Modul SAM-01 (4x Ain)

3.1 Úvodem

Modul SAM-01 slouží pro vzdálené připojení (po RS-422/485) 4 analogových vstupů, jejichž typ a rozsah se volí výměnnou odporovou sítí, tzv. domečkem (staršího typu - jako u modulů PAI-01, viz katalog). Na komunikační lince je užit firemní protokol SAM, podobný protokolu ADAM.

V době těsně po **každém** zapnutí přístroje (cca 1,5 sec) očekává přístroj sekvenci tří po sobě jdoucích znaků ESC (escape, kód 27 [dekadicky]). Tyto znaky očekává na svém sériovém kanálu, nastaveném těsně po každém zapnutí na

2400 Bd

bez parity

1 stop bit

a adresa přístroje je **00** !

Pokud je dostane, přechází přístroj do tzv. konfiguračního režimu.

Pokud je nedostane, pak přístroj provede příkazy, které má po konfiguračním režimu (po konfiguraci) uložené v paměti EEPROM. Tím se po každém zapnutí přístroj nakonfiguruje podle toho, co má v této paměti uloženo, tj. teprve potom platí pro přístroj např. jiná komunikační adresa (než 00 po zapnutí), jiná komunikační rychlost (než 2400 Bd po zapnutí), atd. ...

3.1.1 Propojení PC se SAM-01

Protože se jedná o stejný komunikační protokol a linku RS-485, bude mnoho věcí stejných jako u XDM-14A, resp. SAM-02 v předchozích kapitolách. SAM-01 má pro připojení komunikační linky a napájení stejně zapojené konektory jako SAM-02. Proto zapojení k lince RS-485 proveďte obdobně, a to podle této tabulky:

převodník - DB15		svorkovnice SAM-01	
4	-RxTxD	4	-RxD
		2	-TxD
11	+RxTxD	5	+RxD
		3	+TxD

Napájení modulu SAM-01 zapojte na jeho svorky 6 (-) a 7 (+), hodnotu napájení volte podle údaje na štítku modulu.

Poznámka: v případě předchozího připojení modulu SAM-02 k PC, stačí přendat konektory ze svorek 1-7 na modul SAM-01.

3.1.2 Přechod do konfiguračnígo režimu

Poznámka: výrobce nedávno na svých stránkách zveřejnil program TESTER pro konfiguraci a testování výrobků s komunikací. Součástí TESTERu je i HELP.

V PC spustíme TERMINAL, nastavíme komunikační parametry pro konfiguraci přístrojů s protokolem SAM vždy takto:

komunikační rychlost (Baud rate) na **2400** Bd

Data bits na **8**

Parity na **none**

Stop Bits na **1**

Handshaking na **none**

Zkontrolujeme, zda máme zvolen v PC správný COM, a to tak, že zkusmo napíšeme libovolný znak do okna TERMINALu pro vysílání. LEDky na převodníku RS232/485 musí bliknout.

Na obr. 1 sice vidíte nastavení programu TERMINAL.EXE pro konfiguraci přístroje XDM-14A, ale toto nastavení je stejné pro všechny přístroje s protokolem SAM, tedy i pro SAM-01 (myšleny komunikační parametry a makro M1). Všimněte si vpravo dole rámečku s označením TRANSMIT MACROS a definice jednotlivých maker. Jako M1 vidíte sekvenci tří znaků ESC (kód znaku ESC = 027 dekadicky). A opět právě tuto sekvenci potřebujeme pro přechod SAMu do konfiguračního režimu.

SAM-01 máme vypnut a připojen podle kap. 3.1.1 na RS-485. TERMINAL v PC spuštěn a nastaven podle předchozího. Víme, že klávesou F1 odešleme sekvenci 3x ESC na RS-485, proto zapneme SAM-01 a stiskneme klávesu F1.

V horním okně TERMINALu pro příjem (RECEIVE) se objeví dvojtečka, která nám oznamuje, že jsme se právě dostali do konfiguračního režimu.

Pokud máte s přechodem do konf. režimu potíže, zkontrolujte celé zapojení, ověřte činnost převodníku a vše opakujte i s tím, že klávesu F1 můžete během 1,5 sec, kdy na ni SAM čeká, stisknout vícekrát.

Pokud dvojtečku v okně RECEIVE spatříte, napiště do okna pro vysílání znaků (TRANSMIT) příkaz

```
/?
```

pro výpis jména přístroje a verze FW v něm:

```
/SAM-01*20000710
```

Po příkazu

```
??
```

vypíše všechny konfigurační příkazy uložené v jeho paměti EEPROM/FLASH, např.:

```
%00050A0600
%05W0000
%05R0+00.00+10.00
%05R1+00.00+10.00
%05R2+00.00+10.00
%05R3+00.00+10.00
!
```

Z těchto příkazů poznáte, zda byl přístroj konfigurován a jak, nebo pokud neobsahuje nic (jen !), pak je platné základní nastavení (jako tu první vteřinu po zapnutí, tj. 2400 Bd, bez parity, 1 stopbit, adresa 00). Pokud nějaké příkazy v tomto výpisu vidíte, musíte je podle manuálu rozlousknout a tak se dozvědět adresu přístroje, komunikační rychlost atd...

Ale nakonec proč bychom to louskali a podřizovali se?! Daleko jednodušší je nachystat si požadovanou konfiguraci do souboru příkazů a původní konfiguraci v přístroji prostě přepsat vlastní !

3.1.3 Konfigurace modulu SAM-01

Začněte manuálem protokolu SAM s ohledem na modul SAM-01 a z něj se snažte dozvědět, co všechno se dá na SAM-01 konfigurovat.

Určitě budeme chtít nastavit komunikační parametry, zvolme

adresu modulu **05**

komunikační rychlost **9600 Bd**

prodlevu na odpověď **10 ms**

ostatní = **0**

K tomu slouží podle manuálu příkaz

`%aannttccff<CR>`

který podle uvedených požadavků bude vypadat:

`%00050A0600<CR>`

Pak budeme chtít modul použít na akci, kterou budeme programovat v ProgWinu, s modulem SAM-01 bude komunikovat centrála, komunikaci určíme v PW modulem SERIALCOMM, který bude programově obsluhován každou vteřinu. Modul nemá žádné výstupy, budeme z něj jen číst naměřené hodnoty, proto můžeme WatchDog zakázat (nebudeme občerstvovat v modulu SAM-01 žádná data).

Příkaz pro definici WatchDogu:

`%aaWnnnn<CR>`

upravíme na

`%05W0000<CR>`

protože SAM-01 má adresu 05 a 0000 WD zakazuje.

Po prostudování komunikačního manuálu zjistíme, že je nutno pro SAM-01 nastavit rozsahy pro jednotlivé analogové vstupy. K tomu slouží relace typu:

`%aaRn±ddd.d±hhh.h<CR>`

aa = adresa modulu

n = číslo vstupu (0÷3 pro SAM-01)

±ddd.d a **±hhh.h** dolní a horní hranice rozsahu

Čísla musí být vždy zadána 4 ciframi se znaménkem, obě čísla musí mít desetinnou tečku na stejné pozici. Např.:

```
+.0000+.9999
-0.500+0.500
+00.00+10.00
-100.0+100.0
+0000.+1000.
```

Nastavená dolní hranice bude odpovídat měřené analogové hodnotě při čísle 0 z převodníku a horní hranice nejvyššímu číslu z převodníku. Nastavené hranice rozsahů musí být v souladu s použitou konfigurační odporovou sítí příslušného vstupu.

A když jsme ve zjišťování všech možných komunikačních relací s modulem SAM-01, budeme v praktickém vyčítání změřených analogových hodnot tyto vyčítat příkazem typu:

`#aan<CR>`

aa = adresa modulu

n = číslo vstupu (0÷3 pro SAM-01)

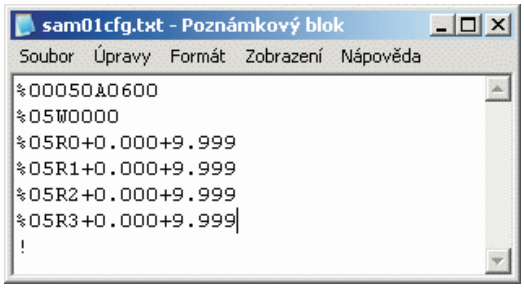
Tentokrát nás bude velmi zajímat i odpověď:

`>+123.4<CR>`

S poznámkou, že analogová hodnota je vrácena vždy v takovém formátu, jak byly zadány dolní a horní hranice rozsahu.

Nyní připravme soubor pro konfiguraci. O komunikačních parametrech jsme již hovořili (aa=05, 9600Bd, ...), zvolme ještě rozsahy měřených hodnot. Dejme tomu, že všechny 4 domečky jsou typu AIPU-10 s rozsahem 0 - 10V (pro měření napětí). AD převodník je 12tibitový, tj. 4096 kroků, tj. 0,00244 V na jeden krok. Pak má smysl nadefinovat meze na 3 desetinná místa, musíme použít v definici vždy znaménko a 4 cifry:

`+0.000+9.999`

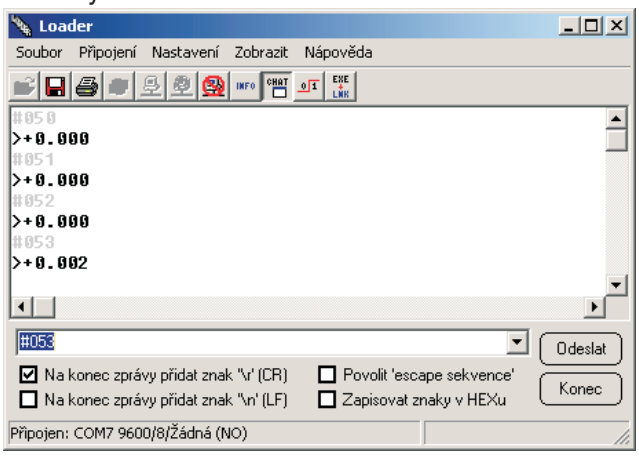


Obr. 26 Konfigurace modulu SAM-01

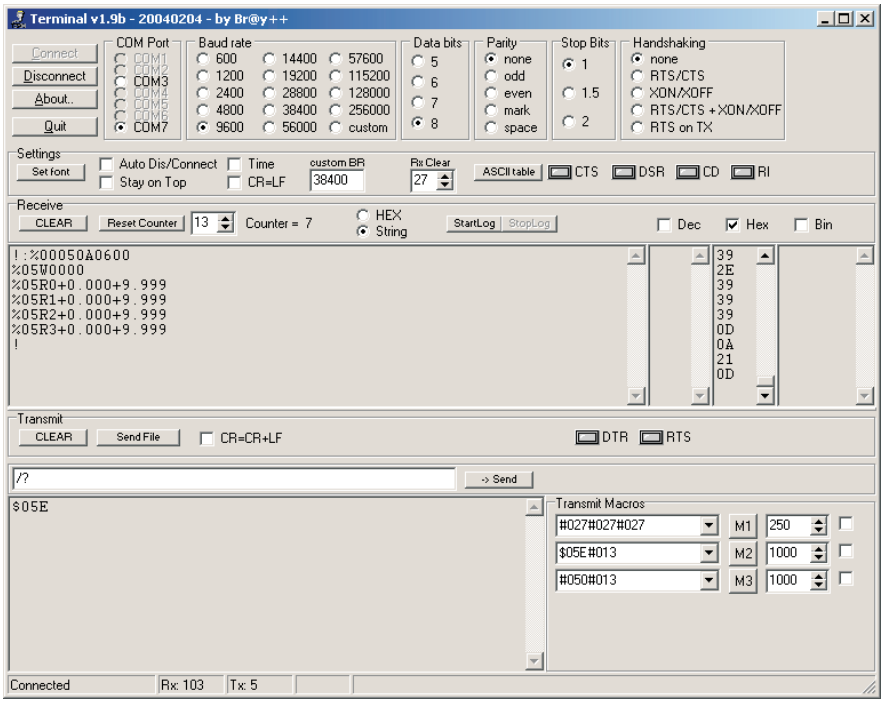
Nachystáme si konfigurační soubor podle obr. 26 a pomocí programu TERMINAL.EXE jej po komunikační lince odešleme do SAM-01.

Po konfiguraci je lepší SAM-01 vypnout a znovu zapnout (napájení) a příkazem si ověřit, že se nám konfigurace zdařila. Viz obrázek 28.

Také lze nakonfigurovaný SAM-01 vyzkoušet třeba pomocí LOADERu příkazem pro vyčtení analogové hodnoty.



Obr. 27 Vyčtení hodnot ze SAM-01



Obr. 28 Kontrola konfigurace SAMu-01

SAM-01 poslouchá, naprogramujeme ho v ProgWinu.

3.2 SAM-01 v ProgWinu

S modulem SAM-01 pracujeme v ProgWinu obdobně jako s předchozími moduly pomocí komunikačního modulu SERIALCOMM, pomocí kterého nadefinujeme relace, které mají na komunikační lince probíhat a pomocí kterých pak modulu SAM-01 vydáme příkaz a následně od něj čteme a vyhodnocujeme jeho odpověď.

3.2.1 Fyzické připojení

Jako SAM-02.

3.2.2 Projekt s modulem SAM-01

Požadavek na vyčtení změřené analogové hodnoty z modulu SAM-01 vypadá obecně:

#aan<CR>

Prakticky musíme dát 4 požadavky, každý pro jeden vstup a očekávat po každém požadavku odpověď. Protože po každém požadavku následuje odpověď, použijeme vždy pro jeden požadavek a odpověď na něj jeden modul SERIALCOMM. Požadavky tedy budou vypadat následovně:

#050<CR>
#051<CR>
#052<CR>
#053<CR>

Podle HELPu ProgWinu, protože přesně víme, jak požadavek bude vypadat, využijeme formát pro vysílání:

- přímo hexadecimální číslo 00 až FF (velkými písmeny), např. 3F. Tento znak je vyslán na linku.

Vyjádříme-li požadavky podle HELPu, zapíšeme pak do parametru **Message_tx_format0** jednotlivých modulů SERIALCOMM:

233035300D
233035310D
233035320D
233035330D

Ve všech případech očekáváme odpověď stejného formátu

>+0.123<CR>

První znak "**>**" čteme modulem SERIALCOMM jako povinně očekávaný, proto zadáváme do formátovacího řetězce hexahodnotu **3E**.

Dalších 6 znaků (včetně znaménka a desetinné tečky) odpovídá hodnotě, vyjádřené ASCII a pro to existuje podle HELPu vyjádření **an..dn**:

*- příkaz **an** až **dn** (malá písmena a, b, c, d s cifrou) = přečte **n** znaků jako ASCII číslo a uloží jako reálné číslo na výstup **Rx0..3** (př. **b6** přečte následujících 6 znaků jako ascii řetězec [např. +23.56],konvertuje na reálné číslo a uloží na výstup **Rx1**)*

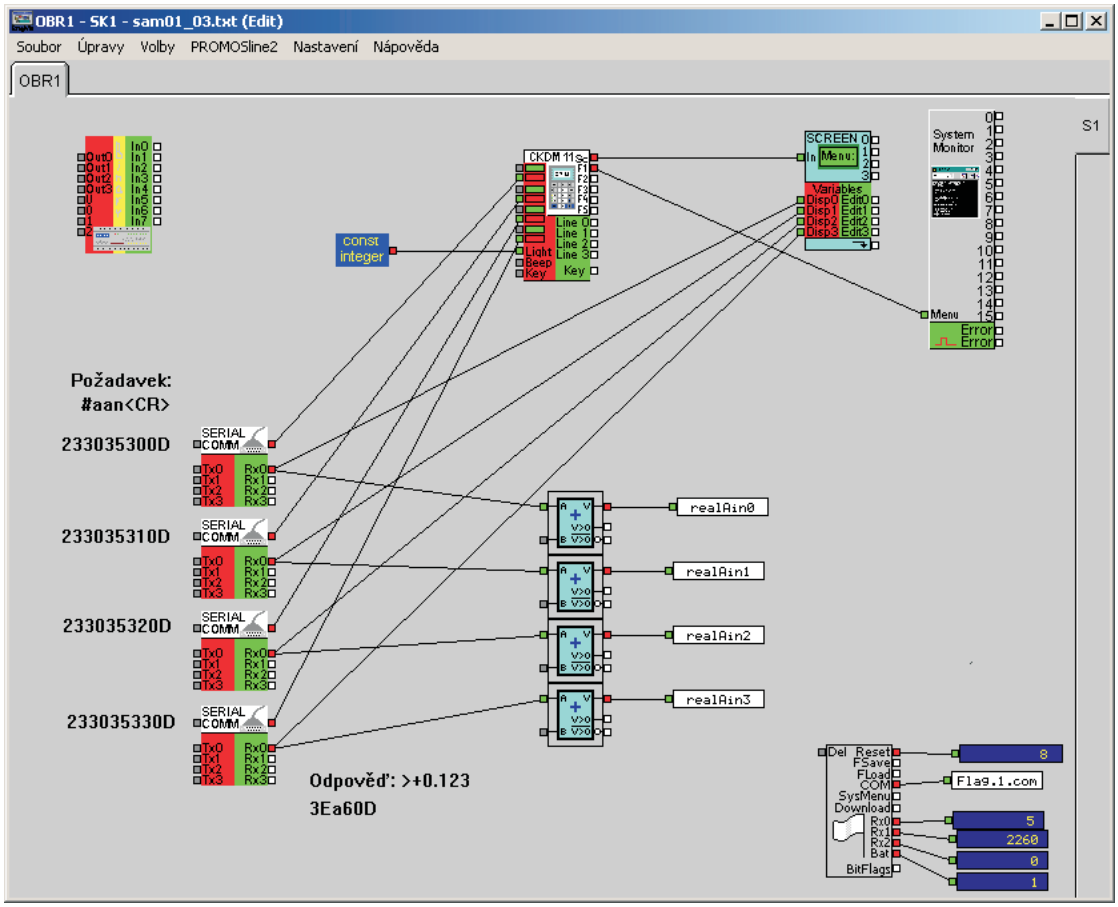
Parametr **Message_rx_format0** pro příjem takové zprávy zapíšeme:

3Ea60D

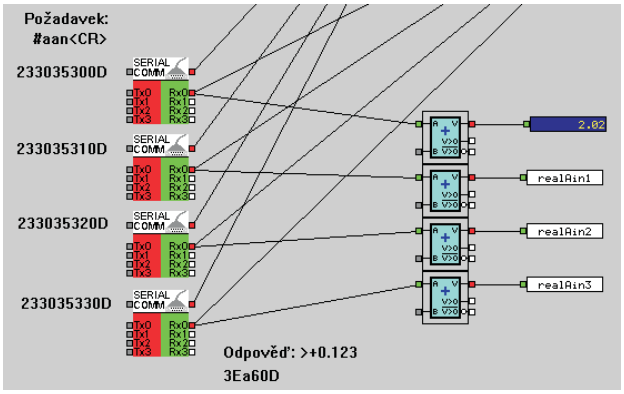
Pak bude z přijaté zprávy hodnota šesti ASCII znaků (včetně znaménka a desetinné tečky) převedena na reálné číslo a bude v ProgWinu k dispozici na výstupu Rx0 každého modulu SERIALCOMM (jeden modul pro požadavek na vyčtení z jednoho analogového vstupu a pro odpověď na tento požadavek).

Nyní víme, jak bude vypadat každý komunikační požadavek a odpověď na něj, proto můžeme projekt začít kreslit. Pokud budete studovat stažený projekt, lépe nyní porozumíte parametrům SERIALCOMMu.

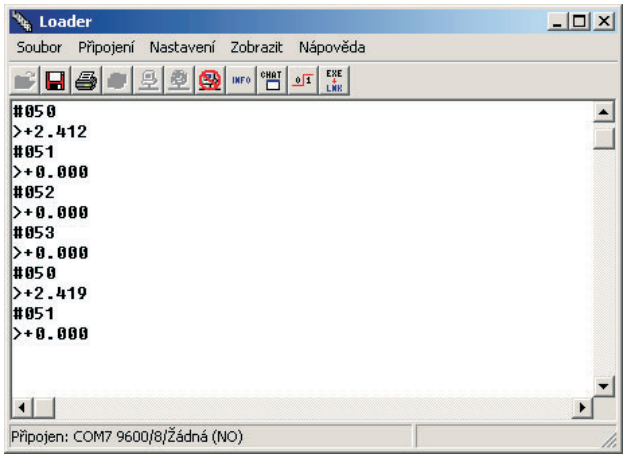
V projektu budou 4 moduly SERIALCOMM podle výše popsaného s příslušnými formáty pro relace. Výstupy Rx0 těchto modulů lze spojit s piny modulu SCREEN pro zobrazování přijatých hodnot. Pokud bychom však chtěli tyto hodnoty vidět přímo na výstupech Rx0, nesmíme zapomenout, že tentokrát převádíme přijaté ASCII znaky na reálné číslo a že hodnotu reálného čísla v RUN režimu ProgWinu uvidíme správně zobrazenou až za některým modulem (ve staženém projektu SAM01_03.TXT to jsou moduly APLUS, a to na jejich výstupech napojených modulech SCROUT s popisem **realAin0..3**) pro reálná čísla. Toto můžete vidět na obrázcích 29 a 30.



Obr. 29 Projekt pro modul SAM-01



Obr. 30 Režim RUN a zobrazení hodnoty



Obr. 31 Odposlech SAM-01 při ladění

3.2.3 Překlad, ladění

Projekt SAM01_03.TXT z webu je pro centrálu CCPU-03, která má USB. Pro překlady a ladění komunikace je to velmi vhodná centrála, protože překlady a RUN režim běží na USB a bez kolizí lze sledovat komunikaci na RS-485 odposlechem po COMu PC pomocí TERMINALu nebo LOADERu (a jim podobným programům). Spolehlivě a bez přepínání či přendávání kabelů...

Provedeme překlad, spustíme LOADER, ve kterém nastavíme COM PC na 9600Bd, 8 bitů bez parity. Pokud je vše OK, sledujeme v okně LOADERu probíhající komunikaci podle obrázku 31.

Tak můžeme najednou vidět v ProgWinu v RUN režimu vyčítanou hodnotu na **realAin0** a v okně odposlechu pak všechny relace na lince.

Pokud v ProgWinu nemáme očekávané, v okně LOADERu posoudíme vypisované relace a podle nich poznáme, která je chybná. V tom případě hledáme chybu v definici formátu zpráv.

Pokud máme v SAM-01 v některých pozicích stejné domečky, lze trojice svorek z čidel mezi sebou prohodit a usuzovat tak na případnou chybu zapojení čidla.

4 Modul SBI-11/12 (16x DI)

4.1 UpG nové verze FW

Základním rozpoznání starého a nového firmware v jednotce SBI-11/12 je signalizace LEDkou s označením RUN (viz čelní štítek).

Jednotka nemusí mít připojenu komunikační linku, stačí ji připojit na napájení. Pokud se LEDka RUN rozsvítí červeně, jedná se o jednotku bez označení verze FW a je nutno FW aktualizovat.

Pokud se po připojení napájení rozsvítí LEDka RUN žlutě, jedná se o jednotku s FW verze 2.00 a vyšší. Pro takovou platí technický manuál **Komunikační protokoly jednotek PL2**.

4.1.1 Postup při UpG FW

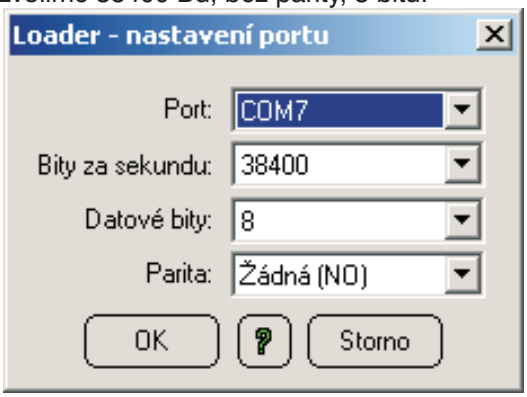
Je-li nutno provést UpG FW, postupujeme podle následujícího popisu.

V centrále spustíme aplikační program, který posílá na komunikační linku RS-485 protokolem epsnet libovolné relace s adresou jednotky, kterou chceme UpG. Že komunikace probíhá poznáte podle toho, že LEDka RUN bude blikat zeleně při každé zprávě.

Prakticky lze na RS-485 přivádět libovolnou relaci, ale častěji než 1x za sekundu (jedná se o udržení watchdogu jednotky). V tomto případě lze probíhající relace zkontrolovat odposlechem, ne na LEDce RUN jednotky).

Jednotku vypneme, sundáme průhledné víčko a čelní panel (štítek), a vlevo od modrých přepínačů DIL (vpravo od něj jsou LEDky) zasuneme do speciálního konektoru kabel s redukcí na RS-232 (ladicí adaptér LSI-11 - pro LOGIC a UpG FW periferií). Adaptér se zasune tak, aby byl LED diodami nahoru - tj. směrem k okraji desky.

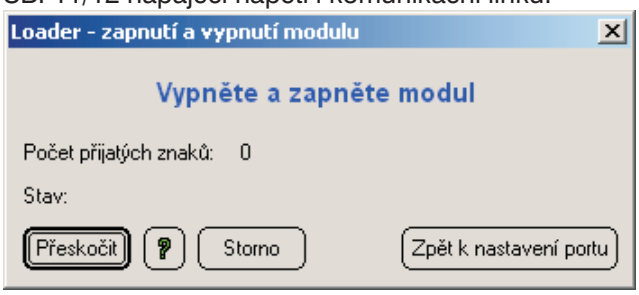
V počítači spustíme LOADER.EXE, nastavíme COM PC, zvolíme 38400 Bd, bez parity, 8 bitů.



Obr. 32

Po volbě OK jsme vyzváni k zapnutí modulu SBI-11/12.

Doporučujeme použít INCO-02 (připojovací svorkovnice; slouží k vyvedení signálů z konektoru PFL10 na 4 šroubovací svorky [+Un, GND, H/+, L/-]). Na označené svorky pro napájecí napětí (pro SBI-11/12 v rozsahu 10 až 30 V DC) přivedete vodiče ze zdroje. Zasunutím INCO-02 na PFL10 přivedeme na jednotku SBI-11/12 napájecí napětí i komunikační linku.

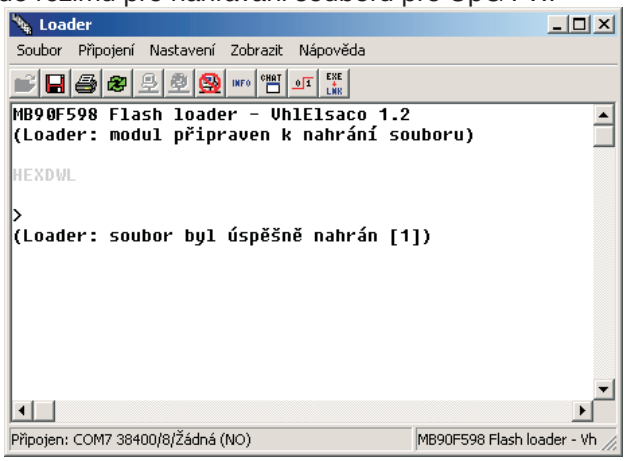


Obr. 33

Pokud modul SBI-11/12 položíte před sebe tak, aby byly nápisy na štítku ke čtení (modré DILy vlevo nahoře), zasunete propojku INCO-02 do pravého konektoru PFL-10.

Jednodušší a spolehlivější ovládání napájecího napětí docílíte vřazením vypínače do vodiče, který vede ze zdroje, a to před svorku +Un svorkovnice INCO-02.

Zapnutí napájení jednotky LOADER pozná a přejde do režimu pro nahrávání souboru pro UpG FW.



Obr. 34

Nejprve se do hlavního okna LOADERu vypíší první řádky, hlásající připravenost k nahrávání souboru:

```
MB90F598 Flash loader - VhlElsaco 1.2
(Loader: modul připraven k nahrání souboru)
```

Pak už jen volíte SOUBOR / OTEVŘÍT (nebo příslušnou ikonu úplně vlevo), zvolíte pro nahrávání soubor SBIEPS.MHX a schválíte nahrávání. Objeví se dlaší okno s bargrafem, ve kterém lze sledovat průběh nahrávání, po jehož skončení posledně vyvolané okno zmizí a v hlavním okně LOADERu se objeví zpráva, že soubor byl úspěšně nahrán.

Vyjmete nahrávací kabel s převodníčkem, vypnete a znovu zapnete modul. LEDka RUN problikne a rozsvítí se žlutě. Tím máte zkontrolováno, že se UpG zdařil (před ním svítila tato LEDka červeně).

Vyjmete INCO-02, osadíte čelní panel a průhledné vyklápěcí víčko zpět. UpG FW je hotov.

Při dalším zapínání modulu si všimněte, že krátce probliknou i zelené LED pro signalizaci logických stavů na vstupních svorkách modulu. Na nich lze v tomto krátkém probliknutí nyní odečíst verzi FW.

```
7 6 5 4 3 2 1 0 [označení na štítku]
8 4 2 1 8 4 2 1 [význam pro odečtení]
```

V mém případě problikly LED 5 a 1, což odpovídá označení verze FW **2.2**.

Zkrácení času při manipulaci s LOADERem lze docílit vytvořením jeho zástupce na ploše.

4.2 Fyzické propojení

Z předchozích pokusů s komunikací máme propojen COM počítače s převodníkem RS-232/RS-485 s čokoládou a s COMem1 centrály. Připojíme k takto vysvorkované lince RS-485 i propojku INCO-02, a to podle následující tabulky.

	COM1 centrála	čokoláda	INCO-02	převodník
-RxTxD	3+6	1	L/-	4
+RxTxD	2+7	2	H/+	11

U propojky INCO-02 dejte pozor na správnou polaritu signálů RS-485. Signál **+TxRxD** je vyveden z pinu **5** konektoru PFL10 jednotky a na svorce propojky INCO-02 je označen jako **H**/**+**.

Signál **-TxRxD** je vyveden z pinu **6** a je označen jako **L**/**-**.

Pokud modul SBI-11/12 položíte před sebe tak, aby byli nápisy na štítku ke čtení (modré DILy vlevo nahoře), zasunete propojku INCO-02 do pravého konektoru PFL-10.

Propojka INCO-02 obsahuje 4 svorky, Dvě z nich, popsané GND a Un, slouží pro přívod napájecího napětí modulu (pro SBI-11/12 v rozsahu 10 - 30 V DC).

Dále je možné zapojit i některý ze vstupů SBI-11/12 pro otestování funkce modulu, zde postupujte podle katalogu.

Máme-li k dispozici SBI-12, přivedeme napájecí napětí galvanicky oddělených vstupů na svorky 11 a 9, resp. 31 a 29, pak stačí zkratovat svorky pro každý jednotlivý vstup, abychom na něm dostali logickou jedničku.

Protokol epsnet je natolik složitý, že nemá smysl se pokoušet moduly SBI, SBIO, SBO či SAIO připojovat pouze k PC a pomocí terminálu se pokoušet s nimi komunikovat. Zvolené připojení linky RS-485 i k PC slouží spíše ke zběžnému odposlechu na lince. Samozřejmě lze pomocí technického manuálu určit jak konkrétně mají jednotlivé zprávy vypadat, ale jsou tak složité, že nemá smysl se pokoušet je vytvářet ručně. Jednodušší je naprogramovat komunikaci v ProgWinu a zprávy pak nanejvýš odposlechem zkontrolovat. Pro případné přenastavení komunikačních parametrů linky RS-485 slouží program NASTAV.EXE (ke stažení na www.elsaco.cz).

4.3 SBI + komunikační parametry

Základní komunikační parametry modulu SBI pro epsnet, nastavené z výroby, jsou:

38400 Bd

sudá parita (even)

1 stop bit

adresu modulu nastavte v sestavě PL2 jako jedinečnou na DILech modulu

4.4 ProgWin a SBI

Protože budeme pokračovat programováním komunikace s modulem SBI v ProgWinu, nastavíme pro epsnet v PW centrálu s adresou 1 a pro modul SBI adresu 2 (tu nastavíme na DILech SBIčka).

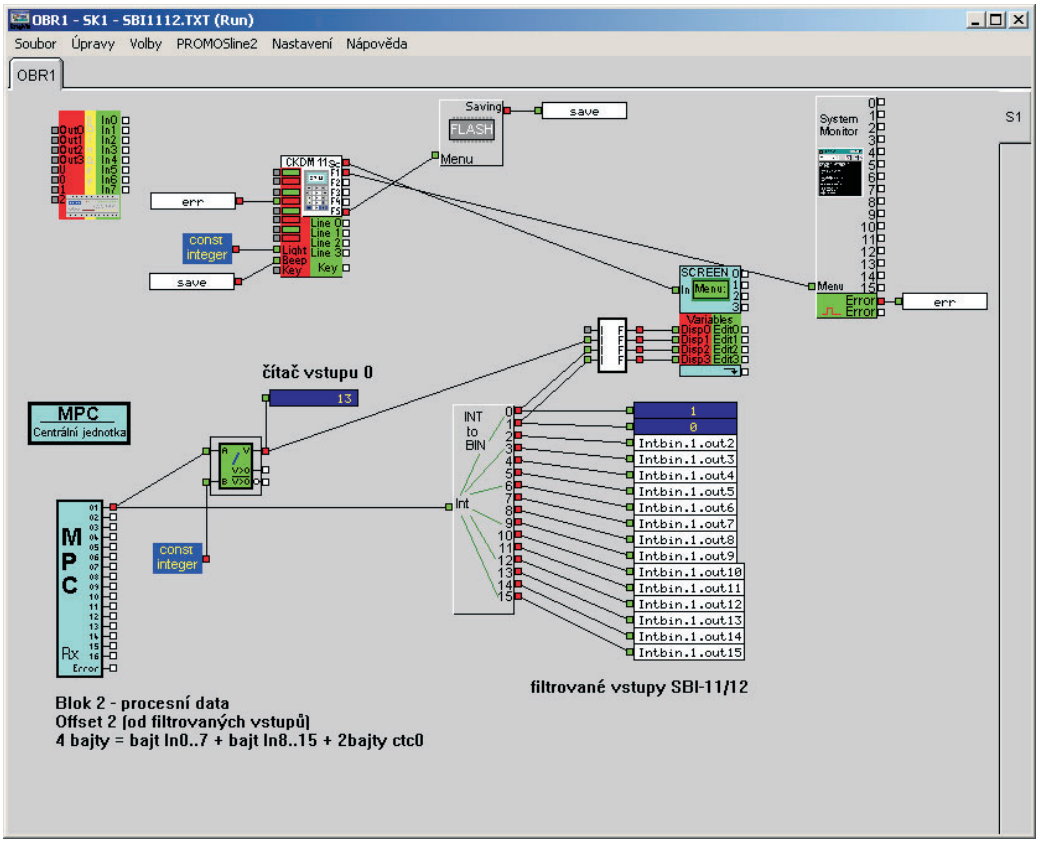
Pro naprogramování v ProgWinu použijeme moduly pro komunikaci protokolem epsnet, a to PWPB_MAIN a PWPB_RX.

Modul **PWPB_MAIN** definuje společné parametry pro všechny přijímací (pwpb_rx) a vysílací (pwpb_tx) moduly. Zvolme tyto hodnoty parametrů:

```
priorita = 0
rychlost = 2
kanal =1
comrychlost = 38400
parita = 2 (sudá)
mezera =0
prodleva = 10
odezva = 100
maxtoken = 500
adresa = 1 (adresa centrály na epsnetu)
maxadresa = 2
```

Definicí parametrů modulu PWPB_RX určíme, která data budeme po epsnetu z modulu SBI přenášet. Podle technického manuálu **SBI-11**/**12**, případně technického manuálu **Komunikační protokoly jednotek PL2**, jsou procesní data uložena v bloku 2, a to jednotlivé položky s tímto offsetem:

Offset	Položka
0 0x00	newin 0 (vstupy 0÷7)
1 0x01	newin 1 (vstupy 8÷15)
2 0x02	filtered 0 (vstupy 0÷7)
3 0x03	filtered 1 (vstupy 8÷15)
4 0x04	čítač impulsů vstupu 0
6 0x06	čítač impulsů vstupu 1
8 0x08	čítač impulsů vstupu 2
10 0x0A	čítač impulsů vstupu 3
12 0x0C	čítač impulsů vstupu 4
14 0x0E	čítač impulsů vstupu 5
16 0x10	čítač impulsů vstupu 6
18 0x12	čítač impulsů vstupu 7
20 0x14	čítač impulsů vstupu 8
22 0x16	čítač impulsů vstupu 9
24 0x18	čítač impulsů vstupu 10
26 0x1A	čítač impulsů vstupu 11
28 0x1C	čítač impulsů vstupu 12
30 0x1E	čítač impulsů vstupu 13
32 0x20	čítač impulsů vstupu 14
34 0x22	čítač impulsů vstupu 15



Obr. 35

36 0x24 čítač impulsů vstupu 0
38 0x26 čítač impulsů vstupu 1
40 0x28 čítač impulsů vstupu 2
42 0x2A čítač impulsů vstupu 3
44 0x2C čítač impulsů vstupu 4
46 0x2E čítač impulsů vstupu 5
48 0x30 čítač impulsů vstupu 6
50 0x32 čítač impulsů vstupu 7
52 0x34 čítač impulsů vstupu 8
54 0x36 čítač impulsů vstupu 9
56 0x38 čítač impulsů vstupu 10
58 0x3A čítač impulsů vstupu 11
60 0x3C čítač impulsů vstupu 12
62 0x3E čítač impulsů vstupu 13
64 0x40 čítač impulsů vstupu 14
66 0x42 čítač impulsů vstupu 15
68 0x44 měřič periody vstupu 0
70 0x46 měřič periody vstupu 1
72 0x48 měřič periody vstupu 2
74 0x4A měřič periody vstupu 3
76 0x4C měřič periody vstupu 4
78 0x4E měřič periody vstupu 5
80 0x50 měřič periody vstupu 6
82 0x52 měřič periody vstupu 7
84 0x54 měřič periody vstupu 8
86 0x56 měřič periody vstupu 9
88 0x58 měřič periody vstupu 10
90 0x5A měřič periody vstupu 11
92 0x5C měřič periody vstupu 12
94 0x5E měřič periody vstupu 13
96 0x60 měřič periody vstupu 14
98 0x62 měřič periody vstupu 15

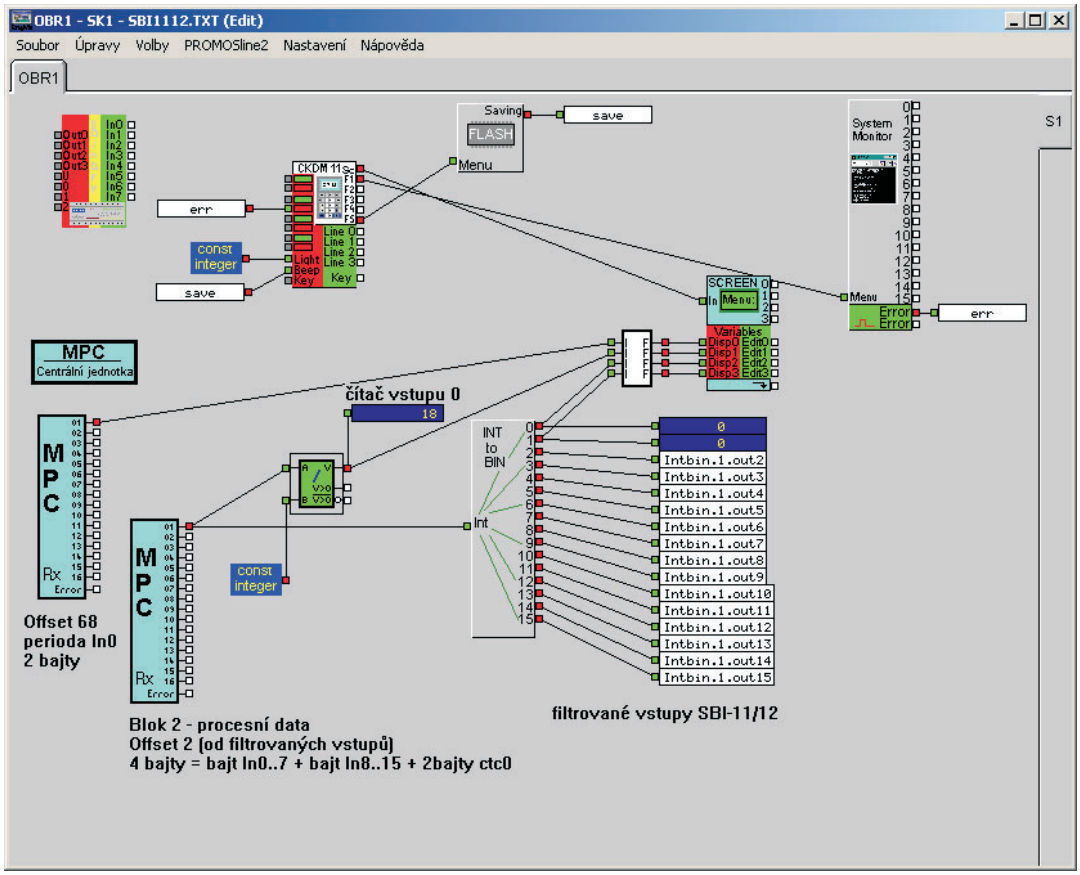
Chtějme z modulu SBI vyčíst stavy filtrovaných vstupů a hodnotu prvního čítače impulzů (vstupu 0). Pak do modulu PWPB_RX zadáme tyto parametry:

```
priorita = 0
rychlost = 2
perioda = 1000
adresa = 2
blok = 2 (procesní data)
offset = 2 (od filtrovaných vstupů)
delka = 4 (4 bajty dat)
initout1..16 = 0
```

Projekt je na obrázku 35. Pokud se zamyslíte nad všemi možnostmi získání dat z modulu SBI, zdá se vyčítání stavů čítačů či hodnot naměřených period pro jednotlivé vstupy složité. Pokud však nepotřebujete kvanta těchto dat, ale např. jen pár z těchto hodnot, nabízí se nadefinovat pro vyčtení jediné zvolené hodnoty přímou relaci. Zvolme si pro takový příklad vyčtení změřené periody pro vstup 0.

Použijeme další modul PWPB_RX (s instancí 2) a v něm nadefinujeme:

```
priorita = 0
rychlost = 2
perioda = 1000
adresa = 2
```



Obr. 36

```
68 08 08 68 02 01 6C 0B 02 44 00 02 C2 16 68 05 05 68 01 02 08 FF FF 09 16 68 08 08 68 02 01 6C 0B
02 02 00 04 82 16 68 07 07 68 01 02 08 00 00 E5 00 F0 16 DC 02 01 DC 02 01 DC 02 01 DC 02 01 68 08
08 68 02 01 6C 0B 02 44 00 02 C2 16 68 05 05 68 01 02 08 FF FF 09 16 68 08 08 68 02 01 6C 0B 02 02
00 04 82 16 68 07 07 68 01 02 08 00 00 E5 00 F0 16 DC 02 01 DC 02 01 DC 02 01 DC 02 01 68 08 08 68
02 01 6C 0B 02 44 00 02 C2 16 68 05 05 68 01 02 08 FF FF 09 16 68 08 08 68 02 01 6C 0B 02 02 00 04
82 16 68 07 07 68 01 02 08 00 00 E5 00 F0 16 DC 02 01 DC 02 01 DC 02 01 DC 02 01 68 08 08 68 02 01
6C 0B 02 44 00 02 C2 16 68 05 05 68 01 02 08 FF FF 09 16 68 08 08 68 02 01 6C 0B 02 02 00 04 82 16
68 07 07 68 01 02 08 00 00 E5 00 F0 16 DC 02 01 DC 02 01 DC 02 01 DC 02 01 68 08 08 68 02 01 6C 0B
02 44 00 02 C2 16 68 05 05 68 01 02 08 FF FF 09 16 68 08 08 68 02 01 6C 0B 02 02 00 04 82 16 68 07
07 68 01 02 08 00 00 E5 00 F0 16 DC 02 01 DC 02 01 DC 02 01 DC 02 01 68 08 08 68 02 01 6C 0B 02 44
00 02 C2 16 68 05 05 68 01 02 08 FF FF 09 16 68 08 08 68 02 01 6C 0B 02 02 00 04 82 16 68 07 07 68
01 02 08 00 00 E5 00 F0 16 DC 02 01 DC 02 01 DC 02 01 DC 02 01 68 08 08 68 02 01 6C 0B 02 44 00 02
C2 16 68 05 05 68 01 02 08 FF FF 09 16 68 08 08 68 02 01 6C 0B 02 02 00 04 82 16 68 07 07 68 01 02
08 00 00 E5 00 F0 16 DC 02 01 DC 02 01 DC 02 01 DC 02 01 68 08 08 68 02 01 6C 0B 02 44 00 02 C2 16
68 05 05 68 01 02 08 FF FF 09 16 68 08 08 68 02 01 6C 0B 02 02 00 04 82 16 68 07 07 68 01 02 08 00
00 E5 00 F0 16 DC 02 01 DC 02 01 DC 02 01 DC 02 01 68 08 08 68 02 01 6C 0B 02 44 00 02 C2 16 68 05
05 68 01 02 08 FF FF 09 16 68 08 08 68 02 01 6C 0B 02 02 00 04 82 16 68 07 07 68 01 02 08 00 00 E5
00 F0 16 DC 02 01 DC 02 01 DC 02 01 DC 02 01 68 08 08 68 02 01 6C 0B 02 44 00 02 C2 16 68 05 05 68
01 02 08 FF FF 09 16 68 08 08 68 02 01 6C 0B 02 02 00 04 82 16 68 07 07 68 01 02 08 00 00 E5 00 F0
16 DC 02 01 DC 02 01 DC 02 01 DC 02 01 68 08 08 68 02 01 6C 0B 02 44 00 02 C2 16 68 05 05 68 01 02
08 FF FF 09 16 68 08 08 68 02 01 6C 0B 02 02 00 04 82 16 68 07 07 68 01 02 08 00 00 E5 00 F0 16 DC
02 01 DC 02 01 DC 02 01 DC 02 01 68 08 08 68 02 01 6C 0B 02 44 00 02 C2 16 68 05 05 68 01 02 08 FF
FF 09 16 68 08 08 68 02 01 6C 0B 02 02 00 04 82 16 68 07 07 68 01 02 08 00 00 E5 00 F0 16 DC 02 01
DC 02 01 DC 02 01 DC 02 01 68 08 08 68 02 01 6C 0B 02 44 00 02 C2 16 68 05 05 68 01 02 08 FF FF 09
16 68 08 08 68 02 01 6C 0B 02 02 00 04 82 16 68 07 07 68 01 02 08 00 00 E5 00 F0 16
```

Obr. 37

```
blok = 2 (procesní data)
offset = 68 (pro periodu vstupu 0)
delka = 2 (2 bajty dat)
initout1..16 = 0
```

Tak získáme jinou komunikační relací změřenou hodnotu periodu na prvním vstupu X0 bez toho, že bychom ji museli z většího bloku dat nějak oddělovat a přepočítávat.

Takto upravený projekt je na obrázku 36.

4.5 Odposlech na lince

Na obrázku 37 můžete sledovat odposlech komunikační linky RS-485 pomocí TERMINALu (na obrázku vidíte pouze data z přijímacího okna).

A pokud máte náladu data šťourat, stačí vzít technický manuál pro SBI-11/12 a v něm nalistovat kapitolu **1.7.2 Blok 2 - procesní data** a podle popisu rozlousknout aspoň počátek zobrazovaných dat.

Na obrázku 37 začínají data blokem:

```
68 08 08 68 02 01 6C 0B 02 44 00 02 C2 16
SD2     SD2R      FC    BLK      LEN    ED
   LE LER  DA SA  kod   offset  FCS
```

Popis jednotlivých bajtů si rozkódujte podle uvedeného manuálu, pak zjistíte, že stanice 1 žádá stanici 2 o data z bloku 2, offset 44h=68d, a sice o 2 bajty dat. Podle našeho zadání již tušíme, že je to zrovna požadavek na hodnotu změřené periody impulzů na vstupu 0. Za touto relací by měla být odpověď na ni:

```
68 05 05 68 01 02 08 FF FF 09 16
SD2     SD2R      FC data  FCS
   LE LER   DA SA             DE
```

Ano, stanice 2 odpovídá stanici 1 a na dotaz posílá hodnotu FF FF, tj. 65535 ms.

Následuje pravděpodobně další požadavek

```
68 08 08 68 02 01 6C 0B 02 02 00 04 82 16
SD2     SD2R      FC   BLK offset   FCS
  LE LER    DA SA    kod        LEN    ED
```

A po rozkódování vidíme, že se jedná opravdu o požadavek stanice 1 na stanici 2, a to o data z bloku 2, s offsetem 2, což je čtení filtrovaných vstupů (2 bajty) a hodnoty prvního čítače (vstupu 0, další 2 bajty).

Následuje samozřejmě odpověď:

```
68 07 07 68 01 02 08 00 00 E5 00 F0 16
SD2     SD2R      FC vstupy      FCS
   LE LEN   DA SA          čítač    ED
```

2 bajty vstupů s hodnotou 0000 říkají, že jsou všechny vstupy v 0, 2 bajty E5 00 (NB VB), tj. E5h říkají, že čítač napočítal do 229.

Další zprávou je tzv. zpráva TOKEN, kterou se snaží centrála s adresou 1 předat tzv. token (oprávnění pro vysílání) další stanici, tj. stanici s adresou 2, tj. modulu SBI.

```
DC 02 01
SD4
   DA SA
```

Viz str. 15 Technického manuálu **Komunikační protokoly jednotek PL2**.

Z uvedeného plyne, že pokud připojujete k centrále sériové moduly, je velmi vhodné zvolit adresu 1 pro epsnet přímo pro centrálu a potom postupně modulo 1 (+1) adresovat další moduly. V modulu PWPB_MAIN pak zadávejte parametr MAXADRESA o hodnotě =2.

4.6 Dodatek platný od verze FW 3.008

Od FW 3.008 (tj. od 1.12.2004) jsou v knihovně ProgWinu zařazeny komunikační moduly pro MPC komunikaci (epsnet) s jednotkami SBI-11/12, SBO-11/12, SBIO-11/12 a SAIO-11/12. Pokud vám v projektu stačí nabízené i/o knihovních modulů (což asi v praxi bude téměř vždy), nemusíte pro vyčtení i/o ze sériových modulů používat způsob, popsaný v této a následujících kapitolách (kap. 4 až 7). Podrobnější informace o tomto jednodušším způsobu - viz kap. 8.

Zároveň byl od FW 3.008 odstraněn jistý nedostatek v MPC komunikaci, týkající se provozu **monomaster**. V kap. 4 až 7 je uvedeno, že parametr MAXADRESA modulu PWPB_MAIN máte plnit hodnotou 2 (protože při plnění nižší hodnotou se FW až bořil).

Od FW 3.008 definujte pro provoz **monomaster** "adresy" stanice na kanálu, kde probíhá MPC komunikace se sériovými moduly parametry:

ADRESA = 1

(adresa vlastní stanice na MPC)

MAXADRESA = 1

(poslední adresa stanice MASTER na MPC)

Další adresy na MPC volte pro jednotlivé stanice typu slave - sériové moduly 2, 3, 4, ...

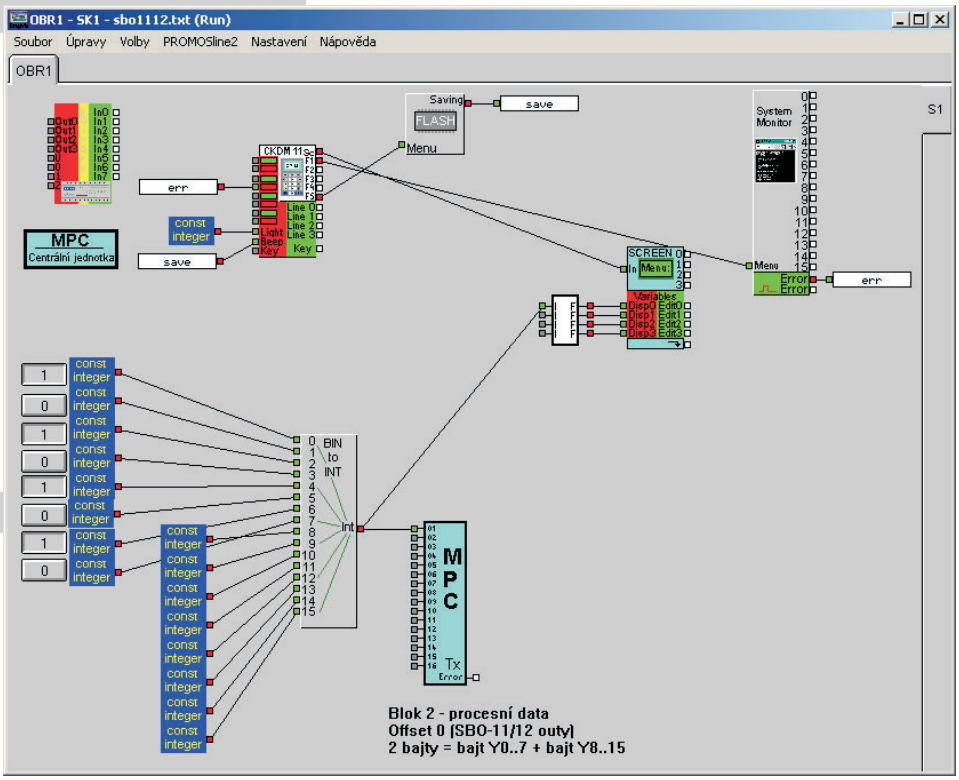
Výsledkem bude to, že se nebudou generovat relace pro předávání tokenu, tj. zmizí v předchozím příkladu relace

```
DC 02 01
```

5 Modul SBO-11/12 (12x DO - relé)

5.1 Úvodem

Pro jednotku SBO platí ohledně FW a zapojení na sériovou linku totéž, co pro předchozí modul SBI-11/12. Pro základní pokusy s jednotkou lze propojku INCO-02 zasunout do jednotky SBO-11 / 12 (předpokládáme napájení všech modulů 24 V DC) a máte sestavu centrála, SBOčko a PC přichystánu k dalšímu.



Obr. 38

5.2 SBO + komunikační parametry

Základní komunikační parametry modulu SBO pro epsnet, nastavené z výroby, jsou:

38400 Bd

sudá parita (even)

1 stop bit

adresu modulu nastavte v sestavě PL2 jako jedinečnou na DILech modulu (2)

5.3 ProgWin a SBO

Protože budeme pokračovat programováním komunikace s modulem SBO v ProgWinu, nastavíme pro epsnet v PW centrálu s adresou 1 a pro modul SBO adresu 2 (tu nastavíme na DILech SBOčka).

Pro naprogramování v ProgWinu použijeme moduly pro komunikaci protokolem epsnet, a to PWPB_MAIN a PWPB_TX.

Modul **PWPB_MAIN** definuje společné parametry pro všechny přijímací (pwpb_rx) a vysílací (pwpb_tx) moduly. Zvolme tyto hodnoty parametrů:

```
priorita = 0
rychlost = 2
kanal =1
comrychlost = 38400
parita = 2 (sudá)
mezera =0
prodleva = 10
odezva = 100
maxtoken = 500
adresa = 1 (adresa centrály na epsnetu)
maxadresa = 2
```

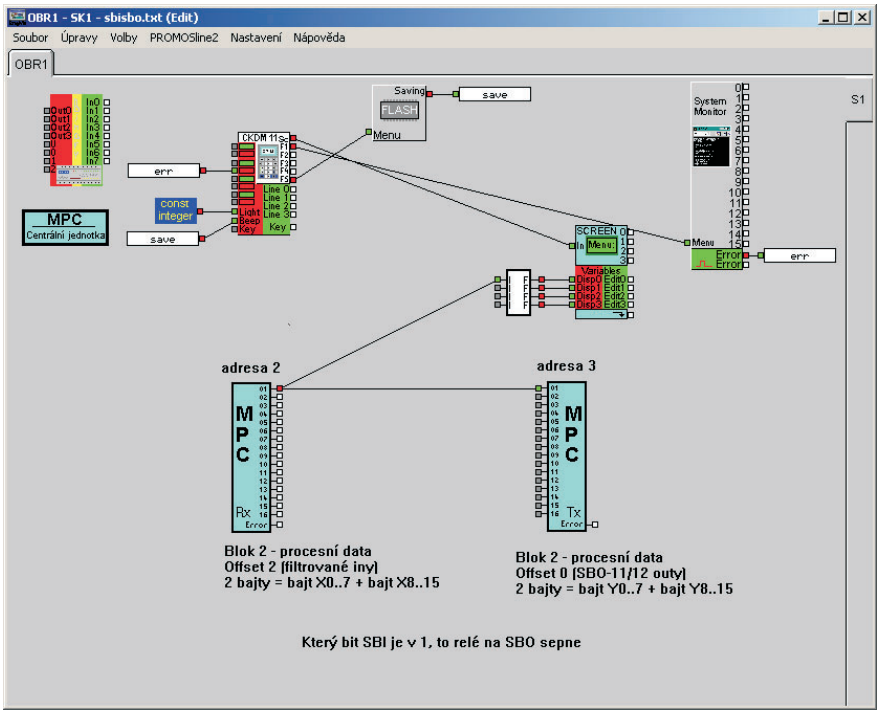
Definicí parametrů modulu **PWPB_TX** určíme, která data budeme po epsnetu do modulu SBO přenášet (chceme do něj zapsat dva bajty, podle jejichž stavu jednotlivých bitů dojde k sepnutí příslušných relé, resp. rozsvícení LED na panelu jednotky). Podle technického manuálu **SBO-11/12**, případně technického manuálu **Komunikační protokoly jednotek PL2**, jsou procesní data uložena v bloku 2, a to jednotlivé položky s tímto offsetem:

Offset	Položka
0=0x00	outs (výstupy 0÷7)
1=0x01	outs (výstupy 8÷15)

Proto nadefinujeme parametry modulu PWPB_TX takto:

```
priorita = 0
rychlost = 2
perioda = 1000
adresa = 2
blok = 2
offset = 0
délka = 2
```

V projektu pak budeme přivádět integer hodnotu na vstup 01 modulu PWPB_TX, ta bude vyslána po komunikační lince protokolem epsnet do jednotky SBO a podle ní na jednotce budou ovládána relé.



Obr. 39

Projekt SBO1112.TXT má pro každé relé CONST_INTEGER, případně ovládanou visual modulem tlačítko pro hodnoty 0 a 1, výstupy konstant jsou vedeny do vstupů modulu BINtoINT a tak je z nich vytvořeno integer číslo, jehož hodnota se skládá z vah jednotlivých bitů, které ovládají jednotlivá relé jednotky SBO.

Po překladu v RUN režimu ProgWinu lze změnou hodnot konstant spínat/rozepínat jednotlivá relé jednotky SBO a odladit tak správnou funkci.

Projekt je na obrázku 38 a je ke stažení na našich stránkách v ZIPu s tímto manuálem.

5.4 Projekt s SBI i SBO

Vytvořme jednoduchý projekt s oběma moduly, které zatím známe. SBI ať čte logické stavy, podle kterých ať SBO spíná jednotlivá relé tak, aby si jednotlivé bity odpovídaly.

V projektu použijeme modul PWPB_MAIN se stejnými parametry jako obvykle, dále pak modul PWPB_RX, který vyčte filtrované stavy vstupů z modulu SBI a vyčtené 2 bajty (pro 16 vstupů) pošleme do modulu PWPB_TX, který tyto bajty vyšle do jednotky SBO, kde bude jimi ovládat jednotlivá relé, resp. signalizační LEDky na panelu SBO (16 LED, 12 z nich odpovídá jednotlivým relé, 4 pouze signalizační).

Modul **PWPB_MAIN** definuje společné parametry pro všechny přijímací (pwpb_rx) a vysílací (pwpb_tx) moduly. Zvolme tyto hodnoty parametrů:

```
priorita = 0
rychlost = 2
kanal =1
comrychlost = 38400
parita = 2 (sudá)
mezera =0
prodleva = 10
odezva = 100
maxtoken = 500
adresa = 1 (adresa centrály na epsnetu)
maxadresa = 2
```

Chceme z modulu SBI vyčíst stavy filtrovaných vstupů. Pak do modulu PWPB_RX zadáme tyto parametry:

```
priorita = 0
rychlost = 2
perioda = 1000
adresa = 2
blok = 2 (procesní data)
offset = 2 (od filtrovaných vstupů)
delka = 2
initout1..16 = 0
```

Parametry modulu PWPB_TX nadefinujeme takto:

```
priorita = 0
rychlost = 2
perioda = 1000
adresa = 3
blok = 2
offset = 0
délka = 2
```

Projekt je jednak na obrázku 39, jednak je opět součástí ZIP balíčku na webu.

6 Modul SBIO-11/12 (8x DI + 8x DO - relé)

6.1 Úvodem

Pro jednotku SBIO platí ohledně FW a zapojení na sériovou linku totéž, co pro předchozí moduly SBI a SBO. Pro základní pokusy s jednotkou lze propojku INCO-02 zasunout do jednotky SBIO-11/12 (předpokládáme napájení všech modulů 24 V DC) a máte sestavu centrála, SBIOčko a PC přichystánu k dalšímu.

6.2 SBIO + komunikační parametry

Základní komunikační parametry modulu SBIO pro epsnet, nastavené z výroby, jsou:

38400 Bd

sudá parita (even)

1 stop bit

adresu modulu nastavte v sestavě PL2 jako jedinečnou na DILech modulu (2)

6.3 ProgWin a SBIO

Protože budeme pokračovat programováním komunikace s modulem SBIO v ProgWinu, nastavíme pro epsnet v PW centrálu s adresou 1 a pro modul SBIO adresu 2 (tu nastavíme na DILech SBIOčka).

6.3.1 Čtení vstupů, ovládání výstupů

Pro naprogramování v ProgWinu použijeme moduly pro komunikaci protokolem epsnet, a to PWPB_MAIN, PWPB_RX pro vyčtení logických stavů a PWPB_TX pro ovládání relé..

Modul **PWPB_MAIN** definuje společné parametry pro všechny přijímací (pwpb_rx) a vysílací (pwpb_tx) moduly. Zvolme tyto hodnoty parametrů:

```
priorita = 0
rychlost = 2
kanal =1
comrychlost = 38400
parita = 2 (sudá)
mezera =0
prodleva = 10
odezva = 100
maxtoken = 500
adresa = 1 (adresa centrály na epsnetu)
maxadresa = 2
```

Definicí parametrů modulů PWPB_RX a PWPB_TX určíme, která data budeme po epsnetu z modulu SBI přenášet a která budeme do něj zapisovat. Modulem PWPB_RX lze vyčítat stavy nefiltrovaných i filtrovaných logických vstupů, hodnoty čítačů impulzů jednotlivých vstupů a hodnoty naměřených period opět pro každý vstup zvlášť. Modulem PWPB_TX lze ovládat 8 relé.

Podle technického manuálu **SBIO-11/12**, případně technického manuálu **Komunikační protokoly jednotek PL2**, jsou procesní data uložena v bloku 2, a to jednotlivé položky s tímto offsetem:

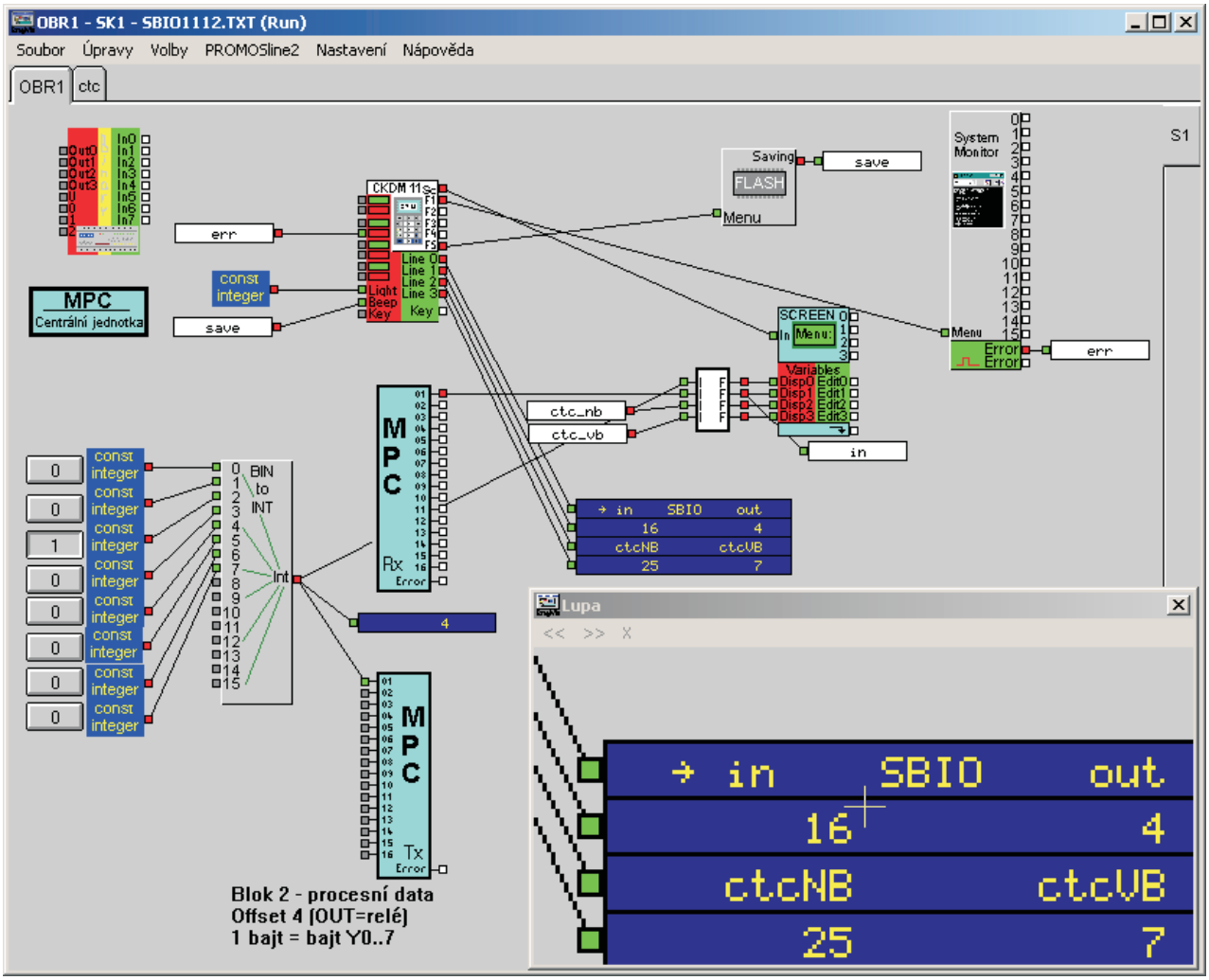
Offset	Položka
0 0x00	newin 0 (vstupy 0÷7)
2 0x02	filtered 0 (vstupy 0÷7)
4 0x04	outs (výstupy 0÷7)
6 0x06	čítač impulsů vstupu 0
8 0x08	čítač impulsů vstupu 1
10 0x0A	čítač impulsů vstupu 2
12 0x0C	čítač impulsů vstupu 3
14 0x0E	čítač impulsů vstupu 4
16 0x10	čítač impulsů vstupu 5
18 0x12	čítač impulsů vstupu 6
20 0x14	čítač impulsů vstupu 7
38 0x26	čítač impulsů vstupu 0
40 0x28	čítač impulsů vstupu 1
42 0x2A	čítač impulsů vstupu 2
44 0x2C	čítač impulsů vstupu 3
46 0x2E	čítač impulsů vstupu 4
48 0x30	čítač impulsů vstupu 5
50 0x32	čítač impulsů vstupu 6
52 0x34	čítač impulsů vstupu 7
70 0x46	měřič periody vstupu 0
72 0x48	měřič periody vstupu 1
74 0x4A	měřič periody vstupu 2
76 0x4C	měřič periody vstupu 3
78 0x4E	měřič periody vstupu 4
80 0x50	měřič periody vstupu 5
82 0x52	měřič periody vstupu 6
84 0x54	měřič periody vstupu 7

Chtějme z modulu SBIO vyčíst stavy filtrovaných vstupů. Pak do modulu **PWPB_RX** zadáme tyto parametry:

```
priorita = 0
rychlost = 2
perioda = 1000
adresa = 2
blok = 2 (procesní data)
offset = 2 (od filtrovaných vstupů)
delka = 1 (1 bajt dat)
initout1..16 = 0
```

Osmice stavů vstupů je uložena v jednom bajtu, proto stačí číst jen tento jeden bajt.

Pokud bychom chtěli číst hodnoty čítačů či period, použijeme k tomu další modul PWPB_RX s obdobnými parametry, pouze posuneme offset podle definice bloku 2, bloku procesních dat. Rovněž podle počtu přenášených hodnot upravíme parametr DELKA, a to s tím, že jedna hodnota je ve dvou bajtech a s tím, že musíme počítat, že na výstupech PWPB_RX získáváme hodnoty čtyřbajtové. Lepší bude, když tomu bude-



Obr. 40

me věnovat v této kapitole další její část. Nyní chtějme z jednotky SBIO přečíst 8 filtrovaných vstupů a zapsat do ní povel pro ovládání 8 relé.

Proto nadefinujeme parametry modulu **PWPB_TX** takto:

```
priorita = 0
rychlost = 2
perioda = 1000
adresa = 2
blok = 2
offset = 4
délka = 1
```

V projektu pak budeme přivádět integer hodnotu na vstup 01 modulu PWPB_TX, ta bude vyslána po komunikační lince protokolem epsnet do jednotky SBIO a podle ní na jednotce budou ovládána relé.

Projekt je na obrázku . OUTy - relé - ovládáme pomocí visual tlačítek v RUNu ProgWinu, vysílanou hodnotu vidíme jak na ovládacím panelu CKDM-11, tak v RUN režimu v modulech SCROUT. Obdobně vidíme i vyčítanou hodnotu, odpovídající stavům logických vstupů.

6.3.2 Čtení čítačů a period

Rozšíříme náš projekt v ProgWinu nejprve o čtení hodnot čítačů pulzů pro logické vstupy modulu SBIO. V tabulce pro offsety (předchozí kapitola) nalezneme, že první čítač (vstupu X0) má offset = 6. Pokud tedy budeme chtít vyčítat čítače od vstupu 0, použijeme v dalším modulu PWPB_RX parametr OFFSET = 6, pokud bychom chtěli vyčítat až od čítače vstupu X4, byl by OFFSET = 14. Další parametr DELKA nastavíme podle požadovaného počtu vyčítaných čítačů. Jedna hodnota je v poli dat uložena jako dvoubajtový integer. Modul PWPB_RX vyčte souvislý tok bajtů od zadaného OFFSETu dlouhý podle DELKA a na výstupní piny tyto bajty předává po čtveřicích!

Teoreticky máme tedy dvě možnosti.

První spočívá v definici vždy jednoho modulu PWPB_RX pro vyčítání jen jedné hodnoty z jednoho čítače jako dvoubajt (DELKA=2). Tím však více zatížíme komunikaci.

Druhá možnost spočívá v tom, že jedním modulem PWPB_RX vyčteme třeba všechny čítače (tj. jednou komunikační relací, resp. jedním požadavkem a jednou odpovědí) a na výstupech modulu získaná data rozebereme, rozřadíme.

Ukážeme si druhou složitější možnost, a to alespoň pro dva sousední čítače - čili rozebrat jeden čtyřbajtový integer na dva dvoujbajtové. Pro více čítačů by se princip jen opakoval.

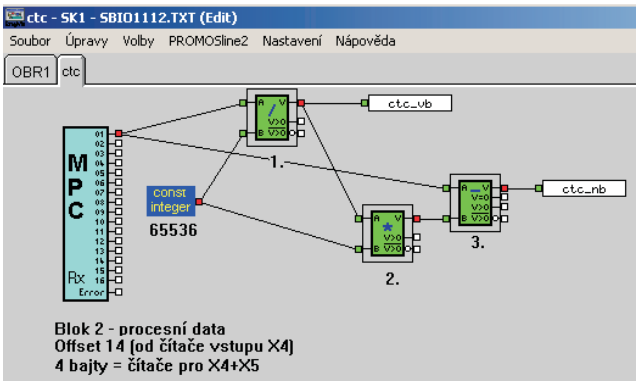
A chtějme vyčíst hodnoty čítačů pro X4 a X5. Pak parametry modulu PWPB_RX nadefinujeme takto:

```
priorita = 0
rychlost = 2
perioda = 1000
adresa = 2
blok = 2 (procesní data)
offset = 14 (od čítače vstupu X4)
delka = 4 (4 bajty dat)
initout1..16 = 0
```

Na obrázku 41 jak z dlouhé 4bajtové celočíselné hodnoty udělat dvě dvojbajtové. Dvoubajt má maximální hodnotu FFFFh, proto je ve výpočtu užito celočíselné dělení a násobení hodnotou 65536, tj. hodnotou prvního přetečení (FFFFh = 65535d). Pro to, aby výpočet probíhal správně, je třeba nadefinovat i prioritu "zelených" celočíselných hradel. Ta je na obrázku zdůrazněna pod hradly.

Projekt SBIO1112.TXT je opět ke stažení spolu s tímto manuálem.

Pro vyčtení naměřených hodnot period pulzů na jednotlivých vstupech X0...7 se jedná o stejný proncip jako u čítačů. Jediné co v definici parametrů dalšího případného modulu PWPB_RX změníme je OFFSET a DELKA. OFFSET nalezneme v tabulce offsetů, v ní vidíme, že pro první vstup X0 je pro periodu offset rovný 70, atd. Délku bloku dat volíme stejně jako u čítačů - po dvou bajtech pro každou hodnotu.



Obr. 41

7 Modul SAIO-12 (max. 12x AI, max. 6x AO)

7.1 Úvodem

Pro jednotku SAIO-12 platí ohledně FW a zapojení na sériovou linku totéž, co pro předchozí moduly SBI-11/12 a SBO-11/12. Pro základní pokusy s jednotkou lze propojku INCO-02 zasunout do jednotky SAIO-12 a máte sestavu centrála, SBOčko a PC přichystánu k dalšímu.

7.2 SAIO + komunikační parametry

Základní komunikační parametry modulu SAIO-12 pro epsnet, nastavené z výroby, jsou:

38400 Bd

sudá parita (even)

1 stop bit

adresu modulu nastavte v sestavě PL2 jako jedinečnou na DILech modulu (2)

7.3 ProgWin a SAIO-12

Protože budeme pokračovat programováním komunikace s modulem SAIO-12 v ProgWinu, nastavíme pro epsnet v PW centrálu s adresou 1 a pro modul SAIO-12 adresu 2 (tu nastavíme na DILech jednotky SAIO-12).

Pro naprogramování v ProgWinu pro čtení naměřených analogových hodnot použijeme moduly pro komunikaci protokolem epsnet, a to PWPB_MAIN a PWPB_RX.

Modul **PWPB_MAIN** definuje společné parametry pro všechny přijímací (pwpb_rx) a vysílací (pwpb_tx) moduly. Zvolme tyto hodnoty parametrů:

```
priorita = 0
rychlost = 2
kanal =1
comrychlost = 38400
parita = 2 (sudá)
mezera =0
prodleva = 10
odezva = 100
maxtoken = 500
adresa = 1 (adresa centrály na epsnetu)
maxadresa = 2
```

7.3.1 Analogové vstupy

Definicí parametrů modulu PWPB_RX určíme, která data budeme po epsnetu z modulu SAIO- 12 přenášet. Podle technického manuálu **SAIO-11/12**, případně technického manuálu **Komunikační protokoly jednotek PL2**, jsou procesní data uložena v bloku 2 a 3, proto si tentokrát musíme manuál přečíst pečlivěji. Pak zjistíme, že naměřené hodnoty, v mezích dle domečku a linearizované, jsou ve tvaru reálného čísla uloženy v bloku 2. Závěr bloku je pak po jednom bajtu věnován DA výstupům (integer 0-255).

Nás nyní zajímá to jednodušší, přečíst všech 12 měřených analogových hodnot. Budeme tedy počítat na epsnetu s blokem 2, offsetem 0 a s prvními 48 bajty, a to dle popisu v uvedených manuálech:

Offset	Položka
0 0x00	ad 0
4 0x04	ad 1
8 0x08	ad 2
12 0x0c	ad 3
16 0x10	ad 4
20 0x14	ad 5
24 0x18	ad 6
28 0x1c	ad 7
32 0x20	ad 8
36 0x24	ad 9
40 0x28	ad 10
44 0x2c	ad 11
48 0x30	da 0
49 0x31	da 1
50 0x32	da 2
51 0x33	da 3
52 0x34	da 4
53 0x35	da 5

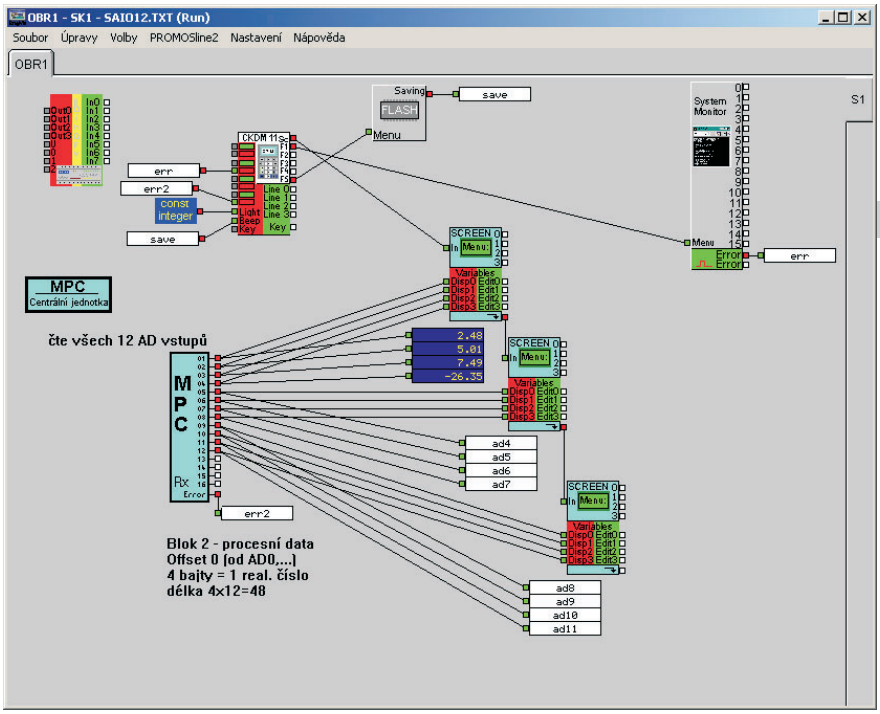
Pak do modulu PWPB_RX zadáme tyto parametry:

```
priorita = 0
rychlost = 2
perioda = 1000
adresa = 2
blok = 2 (procesní data)
offset = 0 (od AD0)
delka = 48 (4x12 bajtů dat)
initout1..16 = 0
```

V tomto případě pro čtení analogových hodnot to vypadá vcelku jednoduše, složitější to bude pro zápis DA hodnot pro analogové výstupy, ale o tom až za chvíli.

V projektu SAIO12.TXT (opět v ZIPu na webu) jsou čtené analogové hodnoty z jednotky SAIO-12 vedeny z modulu PWPB_RX jednak na modulky SCROUT pro zobrazení hodnot v RUN režimu ProgWinu, jednak na moduly SCREEN v kaskádě pro zobrazení na ovládacím panelu.

Výstup ERROR modulu PWPB_RX ovládá spodní červenou LEDku na ovládacím panelu a jak zjistíte, bude neustále svítit. Pokud vám to bude vadit, pamatujte, že výstup ERROR zůstává v 0 pouze při správné komunikaci s PWPB_RX a jen v případě, že bude komunikováno všech 64 možných bajtů. Protože nám pro zjištění 12 analogových hodnot stačí přenést 12 x 4 = 48 bajtů, můžeme zadat délku = 48, ale musíme



Obr. 42

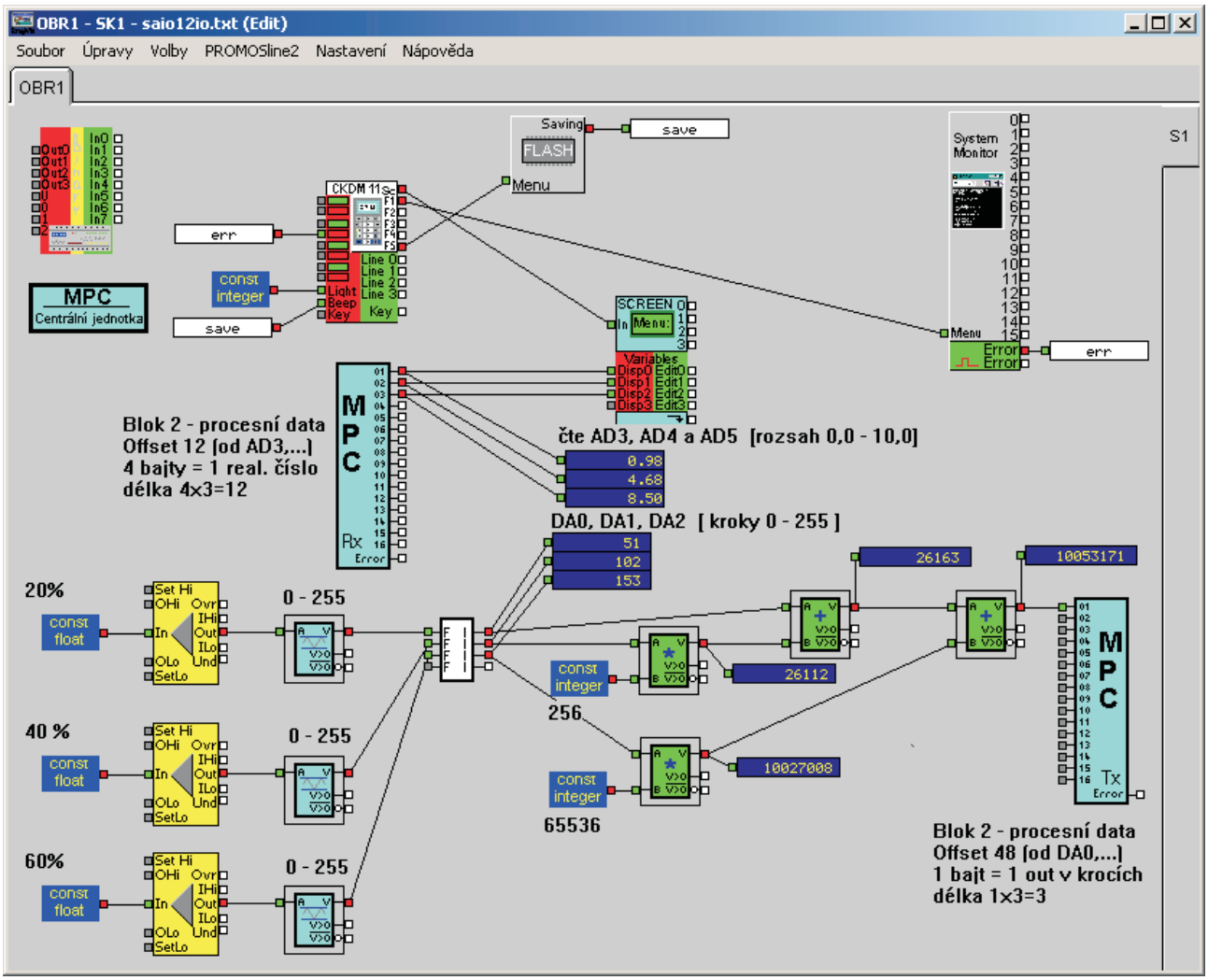
počítat s tím, že nemá smysl využívat zmíněného chybového hlášení na vástupu ERROR.

Zmíněný projekt je rovněž na obrázku 42.

7.3.2 Analogové výstupy

Nyní se podívejme na problematiku DA výstupů. V projektu, např. pro ovládání serva, získáme hodnotu DA výstupu, kterou chceme vyslat (zapsat) do jednotky SAIO-12. Dejme tomu, že chceme takto ovládat tři serva, tři DA výstupy.

Nejprve si musíme přečíst v tabulce offsetů, kde se v poli dat nachází DA výstupy a jak jsou definovány. Z tabulky a popisu je vcelku jasné, že pro jeden DA výstup je věnován jeden bajt a v rozsahu 0 až 255 kroků ovládá příslušný analogový výstup. Bude-li



Obr. 43

analogový výstup 0 - 10 V, bude odpovídat 0 krokům 0,0 V a 255 krokům hodnota 10,0 V. Pro tento převod je vhodné použití modulu SCALE. Na jeho výstupu však teoreticky může být i hodnota větší než 255 kroků, proto než s ní budeme pracovat dál, musíme výstup modulu SCALE "prohnat" modulem ALMT a omezit jej v rozsahu 0 - 255.

Budeme-li tedy chtít ovládat tři DA výstupy, a to DA0, DA1 a DA2, použijeme v projektu pro každý modul SCALE a modul ALMT.

Pak si musíme uvědomit, že modulem PWPB_TX budeme chtít jednou relací ovládat všechny 3 DA výstupy, tedy vyšleme trojbajt s tím, že nejnižší bajt bude odpovídat DA0, prostřední bajt DA1 a nejvyšší bajt DA2. Proto musíme hodnotu pro DA1 násobit 256, hodnotu DA2 násobit 65536 a výsledky sečíst, abychom dostali výslednou hodnotu velkého integer čísla, kterou nabídneme modulu PWPB_TX pro přenos do jednotky SAIO-12. To všechno je vidět na obrázku a v projektu SAIO12io.TXT.

V modulu **PWPB_TX** pak nadefinujeme parametry takto:

```
priorita = 0
rychlost = 2
perioda = 1000
adresa = 2
blok = 2
offset = 48 [od DA0]
délka = 3 [pro DA0, DA1, DA2]
```

V projektu pak vidíte, že jsou ještě čteny naměřené hodnoty AD3, AD4 a AD5. To jen pro úplnost, čtení analogů jsme viděli v předchozí kapitole. Tu jen upozornění, že je vhodné analogové výstupy umisťovat vždy od DA0 a všechny analogové výstupy plynule za sebou, teprve za nimi osazovat AD vstupy. Tím ušetříme celkový počet relací, protože jedním požadavkem zapíšeme hodnoty do výstupů a druhým přečteme vstupy.

8 Moduly SBxy-11/12 a SAIO-11/12 v knihovně ProgWinu

8.1 Úvod

V prosinci 2004 byla uvolněna verze FW 3.008 pro centrály systému PL2. Knihovna ProgWinu od této verze FW obsahuje ve skupině **KOMUNIKACE** knihovní moduly pro komunikaci se sériovými periferními jednotkami SBI-11/12, SBO-11/12, SBIO-11/12 a SAIO-11/12. Pokud jste se v předchozích kapitolách seznámili s MPC komunikací, vězte, že následný text bude pro vás srozumitelnější. To proto, že zmíněné nové knihovní moduly jsou vlastně zařazeny do MPC komunikace, a proto v projektu musí být při jejich použití použit rovněž modul **PWPB_MAIN**, pomocí kterého je nutno nadefinovat společné komunikační parametry pro všechny moduly **MPC komunikace** na stejné komunikační lince.

Jestliže jste četli popis MPC komunikace v HELPu či manuálu ProgWinu, začínáte tušit, že zařazením sériových modulů do MPC komunikace získáváte možnost komunikovat třeba i několika centrálami se stejnými periferními jednotkami (pokud budou na jedné komunikační lince).

V této kapitole se budeme zabývat napojením sériových modulů na centrálu systému PL2. Na centrály CCPU-02 a CCPU-03 běžně napojujeme CANovské periferní jednotky (většinou jsou tyto umístěny ve stejném rozvaděči jako centrála). Sériové periferní moduly k těmto centrálám budeme připojovat nejspíše z důvodů jejich větší vzdálenosti od centrály, tj. v případech, kdy budeme v technologii vytvářet jakési další hnízdo modulů (ty mohou být vzdáleny od centrály až 1200 metrů) proto, abychom např. ušetřili za dlouhou kabeláž k mnoha čidlům či akčním prvkům ap.

Použijeme-li v aplikaci centrálu **CCPU-21** (která nemá CAN pro připojení periferních modulů) a nastane potřeba rozšířit I/O centrály, nezapomínejme na možnost připojení sériových periferních jednotek SBxy-11/12 a SAIO-12 k lince RS-485 centrály na COMu1.

Ve všech případech, kdy připojujeme sériové periferní moduly k jedné centrále, volíme tedy MPC komunikaci typu **monomaster** - centrála bude stanicí ***master***, periferní jednotky budou stanice ***slave***. Centrála bude vysílat požadavek na data, periferní jednotky budou odpovídat...

Poznámka:

Centrála bude stanicí MASTER tehdy, bude-li parametr MAXTOKEN (pro maximální dobu držení tokenu) větší než nula.

TOKEN je pověření, které si stanice typu MASTER předávají postupně po adresách +1 mezi sebou. Stanice, která TOKEN právě vlastní je aktivní MASTER a po dobu MAXTOKEN má čas vyřídit svoje požadavky. Po této době předává TOKEN stanici s adresou +1.

Je definováno, že komunikaci začíná vždy MASTER s nejnižší adresou, proto stanice typu MASTER adresujeme

Výzva:

```
68 08 08 68 02 01 6C 0B 02 02 00 04 82 16
SD2 = 68h = pevná hodnota tohoto typu zprávy
   LE LER = opakovaná délka těla zprávy = 8 bajtů
         SD2R = 68h = pevná hodnota
            DA = 02 = adresa cílového modulu/stanice (komu)
               SA = 01 = adresa odesílatele výzvy
                  FC = 6Ch = řídící bajt rámce říká, že je to výzva o data
                     kód operace 0Bh = READN
                        BLK = 02 = číslo bloku
                            2bajtový offset v bloku VB NB = 2
                                 LEN = 4 = počet čtených bajtů
                                    FCS kontrolní součet
                                        ED pevná hodnota 16h, ukončovací znak
```

Odpověď:

```
68 07 07 68 01 02 08 00 00 00 00 0B 16
SD2
   LE LER
         SD2R
            DA = 01 adresa komu
               SA = 02 adresa kdo
                  FC = 08 řídící bajt říká, že jde o odpověď (na ř.b. 6Ch)
                     4 bajty dat, hodnota = 0
                                 FCS kontrolní součet
                                    ED pevná hodnota 16h, ukončovací znak
```

Popis komunikace pomocí klasických MPC modulů PWPB_TX, PWPB_RX

od adresy 1 postupně dále +1. (Jedná se o adresování pro MPC komunikaci, tedy o parametr ADRESA modulu PWPB_MAIN a ne o adresování centrály na hlavním kanále.)

Teprve potom přiřazujeme adresy periferním jednotkám, obecně stanicím typu SLAVE.

Poslední (nejvyšší) adresu stanice typu MASTER ukládáme do parametru MAXADRESA modulu PWPB_MAIN. (To zn. ve všech projektech MASTER stanic na téže komunikační lince; rovněž parametr ODEZVA musí mít všechny tyto stanice shodný ! - viz HELP.)

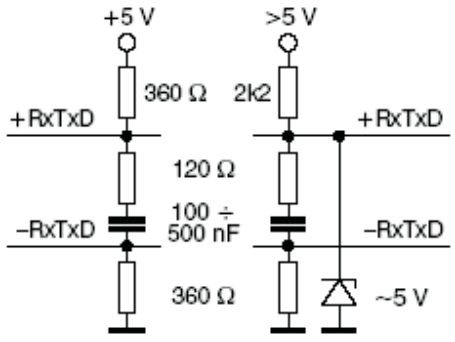
***Monomaster** režim navolíme tak, že u centrály s adresou 1 pro MPC komunikaci naplníme parametr MAXADRESA hodnotou 1. Tím nedojde ke generaci relací pro předávání tokenu - existuje jen jeden MASTER a není to potřeba.*

Připomeňme si (dole na str. 39), jak vypadala komunikace dle projektu a obr. 35, ve kterém vyčítáme jediným požadavkem 2 bajty filtrovaných vstupů a 2 bajty prvního čítače z jednotky SBI-11/12 (také projekt C3sbi12.txt v ZIPu).

8.2 Poznámka k lince RS-485

Většinou se lince RS-485 věnujeme až pokud nastanou problémy s komunikací. Dělejte to naopak - přečtěte si v manuálech, jak má být linka topologicky zapojena a ošetřena a pak se můžete zabývat pouze problémy s protokolem apod. Hodně starostí si tím ušetříte. Nejčastějším opomenutím bývá nesprávné zakončení linky; chcete si ověřit jak se např. pracuje s MPC komunikací na stole, naskládáte HW, propojíte linku dvěma vodiči a očekáváte zázraky. Vězte, že pokud i na stole propojíte zhruba víc jak tři moduly linkou RS-485 bez zakončení (při rychlosti 38 kB), nebude komunikace stoprocentní, v horších případech bude vykazovat pouze chyby.

Sběrnice potřebuje zakončení pouze na dvou místech, pokud se dodržuje správná topologie, je to první a poslední jednotka. Zakončení zajišťuje jednak impedanční přizpůsobení, jednak definici klidového stavu linky. Celé to řeší právě kombinace odporů na napájecí napětí spolu s odporem mezi vodiče RS485. Viz obr. 45.



Obr. 45

Vytahovací odpory, které používáme typicky 360R (ale není to pravidlem), jsou pro správnou komunikaci nutné, bez nich se nedá zajistit správný klidový stav linky.

Odpor mezi vodiče rozhraní kritický není, stejně obvykle málokdo používá kabely s definovanou impedancí - a tak se nedá mluvit o nějakém přizpůsobení. Navíc používané komunikační rychlosti nejsou žádný fofr, takže nějaké zákmity při změně úrovně nemají dopad na vlastní přenos dat, ale zvedají úroveň vyzařovaného rušení.

Co se dá na sběrnici změřit? Použitelné údaje lze získat pouze při zastavené komunikaci. Potom úroveň napětí na lince vůči GND určují především zakončovací odpory a pokud použijete 3x 360R, měli byste naměřit na +485 cca 3,3 V, na -485 cca 1,6 V, mezi dráty taky cca 1,6 V. Tyto údaje by se neměly moc měnit ani při připojování dalších jednotek.

Počet jednotek na RS-485 z HW hlediska závisí na použitých budičích a na odporech, které jsou používány na vstupech pro klidový stav. 32 jednotek není problém, dnešní budiče jich podporují 128.

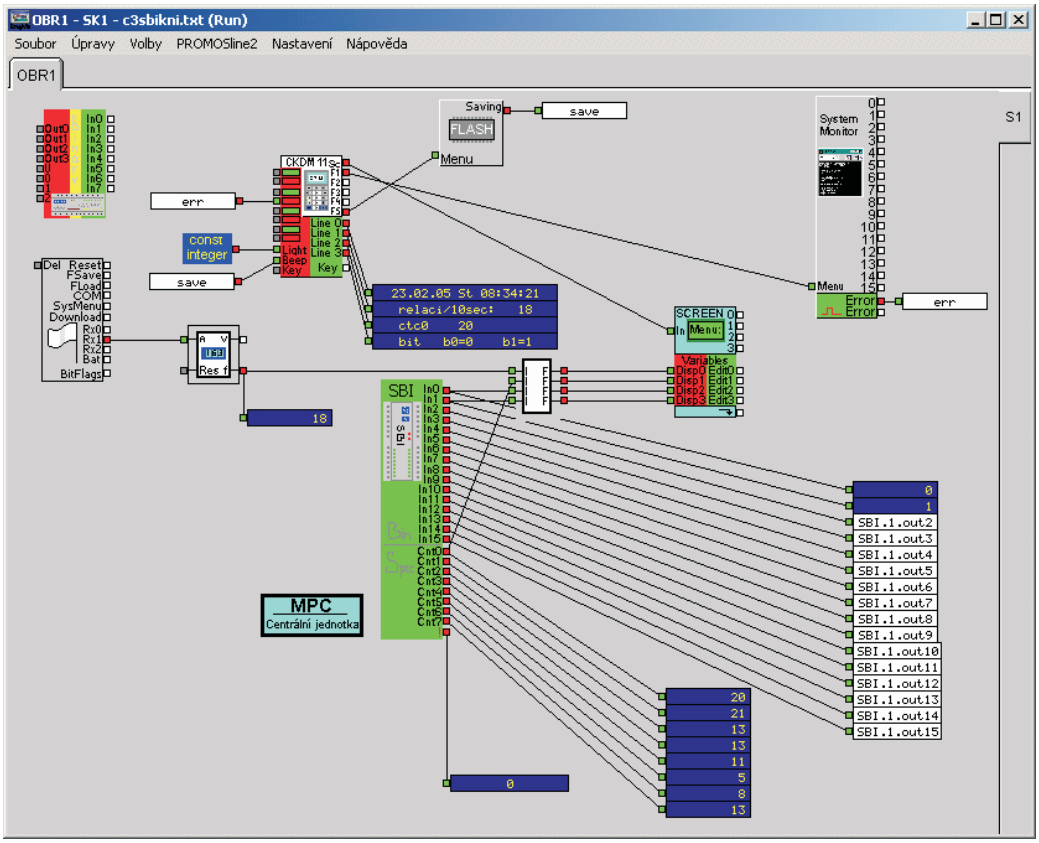
A jak poznáme, zda probíhá komunikace MPC / epsnet po lince RS-485 v pořádku? Každý komunikační modul má výstupní pin "!" a pokud je na něm hodnota = 1 znamená to, že došlo k zastarání dat modulu, tj. na poslední požadavek nepřišla správná odpověď. Proto je v projektu aplikace se sériovými moduly vhodné doplnit chybové hlášení o tomto stavu pomocí knihovních modulů HAVBIN. V našich cvičných projektech je výstup "!" vyveden pouze na modul SCROUT a v RUNu PW lze v něm spatřit požadovanou informaci 0/1 o správné/chybné komunikaci příslušného modulu. V některé z následujících kapitol projekt upravíme i o toto hlášení na displej ovládacího panelu.

8.3 SBI z knihovny PW

Na obr. č. 46 je cvičný projekt **C3SBIkni.TXT** ze ZIPu, který je přibalen k tomuto manuálu. Je použit knihovní modul SBI, pomocí kterého lze z jednotky SBI-11/12 vyčíst všech 16 logických vstupů **In0..15** a hodnoty prvních 8 čítačů **Cnt0..7**. Stavy vstupů a hodnoty čítačů lze v RUN režimu ProgWinu vyčítat do modulů SCR_OUT - viz obrázek.

Chceme-li v projektu použít knihovní modul pro sériovou jednotku/jednotky, musíme v prvé řadě použít modul PWPB_MAIN, který je na obrázku vidět jako blok MPC - centrální jednotka. Jím totiž řekneme, na kterém COMu centrály bude funkční MPC (epsnet) - tj. na který kanál centrály budou sériové jednotky připojeny. Pomocí jeho parametrů pak nadefinujeme další komunikační parametry zvoleného COMu, přitom můžeme vycházet z toho, že koupená sériová jednotka má defaultně pro RS-485 nastavenu komunikační rychlost 38400 Bd, sudou paritu.

V následující tabulce porovnáme parametry modulu PWPB_MAIN, a to defaultní (po položení z knihovny na plochu) s použitými ve jmenovaném cvičném projektu. Nesmíme zapomenout na to, že k centrále nebude připojena na MPC (epsnet) další centrála, tj. budeme nastavovat režim monomaster, kdy MASTER



Obr. č. 46

Výzva:

```
68 08 08 68 02 01 6C 0B 02 00 00 14 90 16
SD2
   LE LER
         SD2R
            adresa komu
               adresa od koho
                  řídící bajt
                     pro READN výzvu
                        číslo bloku zveřejněných dat
                           offset VB NB (vyšší bajt, nižší bajt)
                                 délka vyžadovaných dat
                                    kontrolní součet
                                       pevný ukončovací znak
```

Odpověď:

```
68 17 17 68 01 02 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0B 16
SD2
   LE LER
         SD2R
            komu
               od koho
                  řídící bajt 08 = odpověď na 6Ch
                     14h=20 znaků dat tj. ne i filtrované vstupy + prvních 8 ctc
```

Další stejná výzva pro SBI:

```
68 08 08 68 02 01 6C 0B 02 00 00 14 90 16
```

Odpověď s jinými daty:

```
68 17 17 68 01 02 08 80 00 80 00 19 00 0C 00 05 00 10 00 08 00 05 00 07 00 18 00 71 16
```

Obr. 47 Popis relací modulu SBI

bude centrála s MPC adresou 1 a další jednotka/jednotky na MPC bude/budou typu SLAVE. Ty budou následně adresouvány modulo 1, tj. od adresy 2 postupně dále...

PARAMETR	default	projekt
priorita	0	0
rychlost	1	1
kanal	2	1
comrychlost	9600	38400
parita	2	2
mezera	0	0
prodleva	10	10
odezva	100	100
maxtoken	500	500
adresa	1	1
maxadresa	8	1

Prakticky vidíte, že stačí nadefinovat zvolený COM, upravit komunikační rychlost a pro provoz monomaster zeditovat MAXADRESA=1.

Dokonce až tak nezáleží na parametru ***rychlost***, četnost komunikace nastavíte parametrem **perioda** u modulu SBI, což otestujeme za chvíli.

Nakonec nejjednodušší je přidržet se cvičného projektu, udělat překlad a odposlechnout relace, které pak můžeme porovnat s relacemi v předchozí kapitole, kdy modul SBI nebyl ještě v knihovně. Bude i nyni stačit na vyčtení logických vstupů a tolika čítačů z jednotky SBI-11/12 jediná relace? Ale i kdyby jich bylo víc, při dalším zkoumání a porovnávání projektů je vidět, že nebudeme muset dvoubajtové hodnoty čítačů vypočítávat ze získaných čtyřbajtů! A to přece ušetří práce a možné chybování! Jen se podívejte na obr. 35 nebo 41 jak jsme původně získávali hodnoty z čítačů impulzů jednotlivých vstupů. Teď je hodnota na výstupním pinu knihovního modulu...

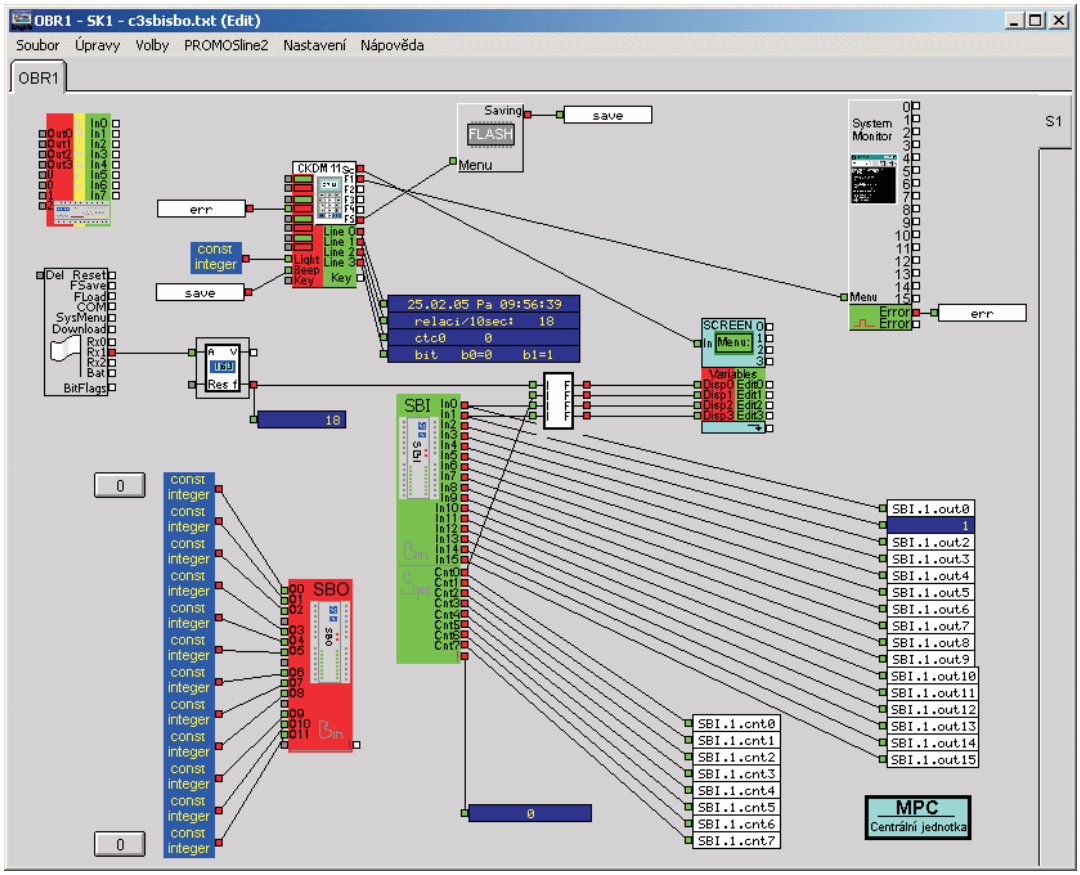
A na obr. 45 vidíme, že modul SBI z knihovny je nahrazen jedním požadavkem, po kterém následuje odpověď a v ní jak stavy vstupů. tak i prvních osmi čítačů. Proč to tak bylo možné zjistíte tehdy, budete-li obr. 45 srovnávat s manuálem pro SBI-11/12 a tam v kap. **1.7.2 Blok2 - složení procesních dat**, resp. jednodušeji zde v kap. 4.3.

V jednom z předchozích odstavců jsme upozorňovali na testování četnosti komunikačních relací na COMu1 (pro sériové moduly). K tomu je ve cvičném projektu využit modul FLAG, jeho výstup Rx1 (ten odevzdává do projektu hodnotu napočítaných přijatých zpráv na COMu1 s epsnetem) je napojen na hradlo CNT, které díky svým parametrům (rychlost=3, freqper=100) odevzdává na výstupu **f** počet relací za posledních 10 vteřin. Všimněte si, že v modulu SBI je parametr **perioda** = 1000. Na displeji v tomto případě čtete, že proběhne 18 až 19 relací za 10 sec. Zapněte RUN režim ProgWinu a zeditujte tuto hodnotu na 5000 ms. Hodnota se změní na 4 relace za 10 sec.

Vypadá to, že parametrem PERIODA u sériového modulu říkáme, jak často si má centrála žádat o jeho hodnoty. Je-li preioda 5000 ms = 5 sec, měl by být učiněn požadavek 2x za 10 sec. V HELPu PW si však přečteme, že parametr PERIODA udává maximální časovou periodu občerstvování dat - čili to napovídá, že systém to bude dělat častěji, asi počítá s možným nezdarem... Počítejme tedy, že systém zabezpečí zhruba dvojnásobek relací proti požadavku, který je dán hodnotou parametru PERIODA. Bude-li PERIODA = 5000 ms, měl by být počet relací za 10 sec zhruba 4, atd. To samozřejmě platí pokud se do nadefinované periody všechny probíhající relace "vejdou".

```
68 08 08 68 02 01 6C 0B 02 00 00 14 90 16
68 17 17 68 01 02 08 02 00 02 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 10 16
68 0A 0A 68 03 01 63 0C 02 00 00 02 00 00 77 16
SD2
   LE
      LE2
         SD2R
            CÍLOVÁ ADRESA
               ODESÍLATEL
                  ŘÍDÍCÍ BAJT PRO WRITEN
                     OPERACE WRITEN
                        ČÍSLO BLOKU DAT 02
                           OFFSET V BLOKU 00 00
                                 POČET ZAPISOVANÝCH BAJTŮ = 2
                                    ZAPISOVANÁ DATA
                                          FCS KONTROLNÍ SOUČET
                                             ZAKONČOVACÍ BAJT PEVNÁ HODNOTA 16h
E5
jednobajtová pozitivní odpověď, pevná hodnota E5h
```

Obr. 48 Popis relace modulu SBO



Obr. 49

8.4 SBO z knihovny PW

Z předchozího je vidět, že práce s komunikačními moduly z knihovny PW pro sériové jednotky bude mnohem jednodušší než s moduly PWPB_RX či PWPB_TX (pro tutéž práci, tj. pro vyčtení či zápis dat z či do sériových modulů). Přesto aspoň stručně a cvičně vyzkoušíme všechny knihovní moduly sériových jednotek.

Pro jednotky SBO-11/12 je v knihovně modul SBO. Pomocí něj pošleme do jednotky relaci pro zápis dat do jejího registru, podle kterého spínají relé modulu.

Doplníme předchozí projekt a uložíme jej jako C3SBISBO.TXT (opět je v ZIPu u tohoto manuálu).

Na obr. č. 49 vlevo dole vidíte jednoduché rozšíření původního projektu o modul SBO. V RUNu ProgWinu lze pak číst počet přijatých relací na COMu1 (MPC/epsnet) cca 18 pro PERIODA=1000 ms, dejte periody u modulů SBI a SBO = 5000 ms (více času) a dostanete očekávaný dvojnásobek zpráv.

Odposlech zpráv vidíte na str. 42, poslední popsané zprávy odpovídají příkazu centrály zapsat do SBO dva bajty dat a odpovědi OK.

K jednotlivým modulům CONST_INTEGER lze přiřadit VISUAL modul TLACITKO a pomoci něj v RUNu PW ovládat jednotlivá relé modulu SBO-11/12. Ve cvičném projektu jsou k dispozici pouze dvě tlačítka - pro první a poslední relé.

Parametry MPC v modulu PWPB_MAIN nebyly proti původnímu projektu měněny.

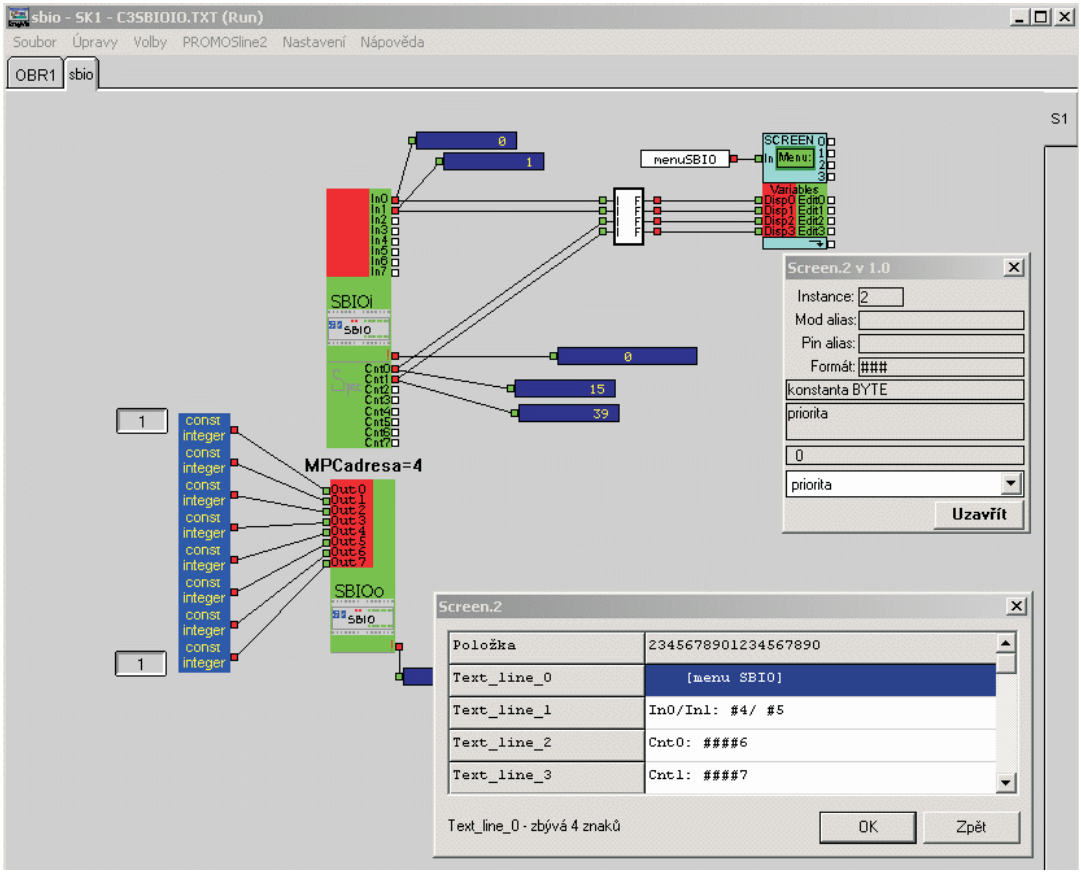
Z odposlechu je zřejmé, že modul SBO zařazuje do komunikace jednu relaci typu WRITEN a jednobajtovou odpověď (E5h) na ni.

8.5 SBIO z knihovny PW

Na obr. 51 je obrazovka "sbio" z připnutého projektu **C3sbioio.txt**, který je opět rozšířeným projektem z předchozího projektu. První co vás asi v projektu zaujme bude to, že pro sériovou jednotku SBIO-11/12 s MPC_adresou = 4 jsou použity dva moduly z knihovny (v projektu položeny pod sebou). První modul (modul **SBIOi**) pro čtení stavu vstupů a hodnot čítačů z jednotky SBIO-11/12 a druhý (modul **SBIOo**) pro zápis - definici stavů relé v jednotce. Už tento popis napovídá, že každý knihovní modul (nakonec je také ze skupiny KOMUNIKACE) zabezpečuje jeden typ komunikační relace na epsnetu. Více uvidíte v komentovaném odposlechu relací na obr. 49. Relace i komentář porovnávejte s popisem komunikačního protokolu epsnet v manuálu **Komunikační protokoly jednotek PL2** (kap.2.2), resp. s manuály jednotlivých jednotek či nejjednodušeji zde v kap. 6.3.1. Vidíte, že modul SBIOi odpovídá na požadavek centrály relací, ve které jsou předávány informace o vstupech (dokonce filtrované i nefiltrované), informace o stavu relé, a hodnoty čítačů. Vypadá to, že je zbytečně přenášeno všechno, ale ... jen jedinou relací, a tím je vlastně ušetřeno. Modul SBIOi pak na výstupní piny přepustí jen to, co je určeno k použití do projektu.

```
68 08 08 68 02 01 6C 0B 02 00 00 14 90 16
            příjemce SBI
               odesilatel centrála
                  6C 0B = READN = čtení z jednotky
                        bloku 2
                           s offsetem 0000
                                 14h=20 bajtů

68 17 17 68 01 02 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0B 16
            příjemce
               odesilatel
                  odpověď na READN
                     20 přečtených bajtů

68 0A 0A 68 04 01 63 0C 02 04 00 02 81 00 FD 16
            příjemce SBIO
               odesilatel centrála
                  63 0C = WRITEN = zápis do jednotky
                        do bloku 2
                           offset 0004 = výstupy 0-7
                                 zápis 2 bajtů
                                    zapsat word 0081, tj. sepnout první a poslední relé

E5
pozitivní odpověď na předchozí relaci

68 0A 0A 68 03 01 63 0C 02 00 00 02 00 00 77 16
            příjemce SBO
               odesilatel centrála
                  63 0C = WRITEN = zápis do jednotky
                        do bloku 2
                           offset 0000 = výstupy 0-16
                                 zápis 2 bajtů
                                    zapsat word 0000, tj. relé odpadnuta

E5
pozitivní odpověď na předchozí relaci

68 08 08 68 04 01 6C 0B 02 00 00 16 94 16
            příjemce SBIO
               odesilatel centrála
                  6C 0B = READN = čtení z jednotky
                        bloku 2
                           s offsetem 0000
                                 16h=22 bajtů

68 19 19 68 01 04 08 02 00 02 00 81 00 06 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 9B
16
            přijímá centrála
               z modulu SBIO
                  řídicí bajt pro odpověď na READN
                     word 0002 = nefiltrované vstupy 0-7, bit 2 = 1
                           word 0002 = filtrované vstupy 0-7, bit 2 = 1
                                 word 0081 = stav výstupů = relé 0+7 sepnuta
                                       word 0006 = hodnota Cnt0
                                             word 0003 = hodnota Cnt1
                                                ostatní Cnt2..7 nulové
```

Obr. 50 Komentované relace, v projektu jednotky SBI-12, SBO-12, SBIO-11



Obr. 51

Modul SBIOo pak zabezpečuje kratší relaci, která se týká zápisu dvou bajtů do registru jednotky, který ovládá relé jednotky. Přísně vzato - stačil by tu zápis jednoho bajtu pro 8 bitů - relé jednotky. Takto je relace obdobná relaci u modulu SBO, který má 12 relé (ale 16 LED = 16 bitů = 2 bajty).

K projektu jen tolik, že paramery základního modulu MPC - PWPB_MAIN - nebyly proti původnímu projektu měněny. Na obr. 52 je pak zobrazeno v RUNu ProgWinu menu ovládacího panelu pro jednotku SBIO-11/12, což můžete porovnat s definicí výpisu menu na obr. 51 vpravo dole.

8.6 SAIO z knihovny PW

I v případě jednotky SAIO-11/12 je situace v knihovně ProgWinu obdobná jednotce SBIO-11/12. V knihovně je k dispozici jeden modul SAIOi pro čtení změřených analogových hodnot a druhý knihovní modul SAIOo pro ovládání analogových výstupů. I když jsou jednotky SAIO-11 a SAIO-12 konstrukčně i FW odlišné, je umístění dat v paměti jednotky pro MPC komunikaci stejné, proto lze pro ně používat stejné knihovní moduly.

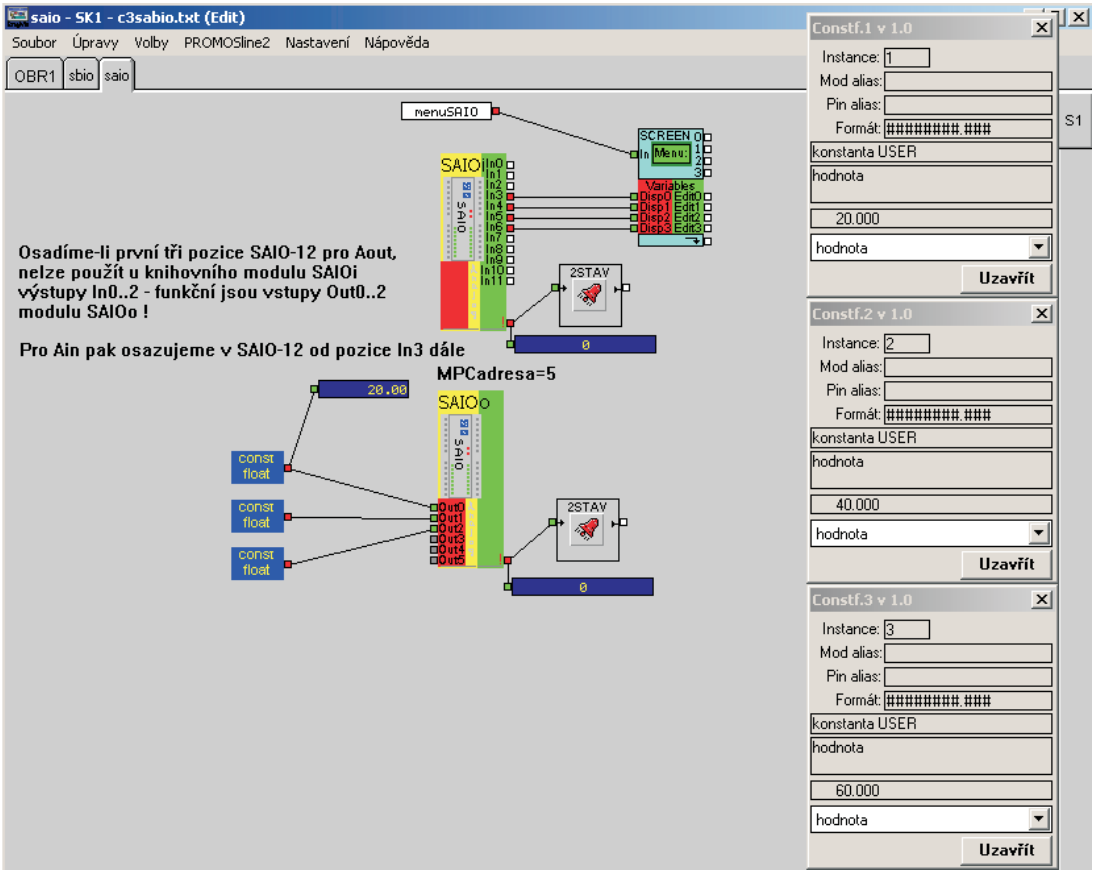
Předchozí projekt byl opět rozšířen, tentokrát o obrazovku **saio** a naleznete jej v ZIP balíčku s touto Kuchařkou pod názvem **C3SABIO.TXT**. V celém projektu byla doplněna chybová hlášení od všech sériových periferních modulů, ke každému byl přidán modul HAVBIN, ve kterém je definováno příslušné chybové hlášení. Pokud je v celé sestavě vše v pořádku, pak po překladu nebude na ovládacím panelu CKDM-11 svítit červená LED s popisem ERROR (podporovaná BEEpem). Pokud některý ze sériových modulů nebude komunikovat správně, bude tato LED svítit a po stisku klávesy F1 si můžete na displeji přečíst, který modul chybu komunikace hlásí. Pro odzkoušení stačí rozpojit "vláček" sériových modulů.

Obr. 52

Rovněž zase můžete zkontrolovat četnost relací, všechny sériové moduly mají nastaven parametr perioda = 5000 ms, takže za 10 sec můžete očekávat 24 relací. Po dvou z modulů SBIO a SAIO, po jedné z modulů SBI a SBO, tj. 6 požadavků + 6 odpovědí = 12 relací x 2 (dvojnásobek pro jistotu) = 24.

Pokud se ohlédneme zpět na předchozí projekty a odposlechy, lze konstatovat, že každý knihovní modul sériové jednotky zabezpečuje v MPC komunikaci jeden požadavek a jednu odpověď na něj.

V nových aplikacích již nepoužívejte jednotku SAIO-11, ale novější **SAIO-12**, se kterou se prakticky lépe pracuje díky novým výměným zásuvným modulkům (lépe přístupné i svorkovnice na nich). A protože analogové výstupy jsou na této jednotce použitelné pouze na prvních 6 pozicích, osazujte a používejte nejprve tyto první pozice pro analogové výstupy a teprve potom souvisle (nebo s vynecháním pozic pro rezervu analogových výstupů) zařaďte modulky pro



Obr. 53

analogové vstupy. Přitom pokud první pozice použijete pro analogové výstupy, pamatujte, že je nemůžete ani v projektu u modulu SAIOi použít pro analogové vstupy.

Např. v připnutém projektu jsou první tři pozice použity pro analogové výstupy, proto analogové vstupy jsou osazovány a zapojovány od další, tj. 4. pozice (první u SAIOi je In3).

Na obr. 53 je část projektu pro komunikaci s jednotkou SAIO-12. Změřené analogy jsou zobrazovány na displeji, pokud máte možnost je i měnit, neočekávejte na displeji rychlou odezvu - nová hodnota se musí nejprve vykomunikovat.

Z obrázku je patrné i nastavení hodnot pro analogové výstupy. V popsané relaci modulu SAIOo (s adresou 5) vidíte pro první analogový výstup hodnotu 33h, v projektu pak 20d. V projektu se udává hodnota v procentech v rozsahu 0 až 100, prakticky je však použit 8mibitový DA převodník (rozsah 0 - 255 kroků). Zkusme 20% přepočítat, 255 * 0,2 = 51d = 33h. A tuto hodnotu také v komunikančí relaci nalezneme.

Obdobně lze spočítat hodnoty pro další dva výstupy, v popisu relace je výpočet obrácený - ze tvaru hexa v krocích na tvar dekadický v procentech.

Výzvu a odpověď (relace) pro čtení analogových hodnot vidíte na další stránce. Oba typy relací (jak pro SAIOi, tak pro SAIOo) přenáší vždy maximální možný počet analogových vstupů nebo výstupů jednotky. Do projektu pak zapojujeme pouze piny osazených vstupů / výstupů.

```
68 0E 0E 68 05 01 63 0C 02 30 00 06 33 66 99 00 00 00 DF 16
           pro SAIOo
              od centrály
                 WRITEN
                       blok 2
                          offset 0030h=48d, tj na Aouty
                                6 bajtů pro 6x Aout
                                   Out0=33h=51d tj 51*100/255=20%
                                      Out1=66h=102d tj 10200/255=40%
                                         Out2=99h=153d tj 15300/255=60%
                                            Out3..5 = 00
E5
pozitivní odpověď na předchozí relaci
```

Popis relace modulu SAIOo

Výzva z centrály pro SAIOi:

```
68 08 08 68 05 01 6C 0B 02 00 00 30 AF 16
            pro SAIOi
               od centrály
                  READN
                        z bloku 2
                           offset 0000
                                 30h=48d bajtů dat, tj. 12 AD po 4 bajtech
```

Odpověď (pro přehlednost rozepsáno na víc řádků):

```
68 33 33 68 01 05 08
            pro centrálu
               od SAIOi
                  odpověď na READN

7A 3F FA 41   0B E6 12 43   2D 19 2D 41   2C 98 EB 40   B9 64 B9 3F   2F CC AE 40
dato1         dato2         dato3         dato4         dato5         dato6

45 5F B1 BF   77 8E 1E 43   68 2E 68 41   37 DD 36 43   BB 86 3A 42   93 D6 4A 43
dato7         dato8         dato9         dato10        dato11        dato12

E2 16
FCS=kontrolní součet
   ED=pevná hodnota ukončovacího znaku
```

Ostatní relace z projektu:

```
68 08 08 68 04 01 6C 0B 02 00 00 16 94 16
68 19 19 68 01 04 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0D 16
68 08 08 68 02 01 6C 0B 02 00 00 14 90 16
68 17 17 68 01 02 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0B 16
68 0A 0A 68 03 01 63 0C 02 00 00 02 00 00 77 16
E5
68 0E 0E 68 05 01 63 0C 02 30 00 06 33 66 99 00 00 00 DF 16
E5
68 0A 0A 68 04 01 63 0C 02 04 00 02 00 00 7C 16
E5
68 08 08 68 05 01 6C 0B 02 00 00 30 AF 16

68 33 33 68 01 05 08 A1 B9 00 42 4A 06 1A 43 AE 51 2E 41 EC 96 EB 40 B9 78 B9 3F EF CA AE
40 1C 92 B0 BF 62 85 19 43 E7 F5 66 41 5B 96 3A 43 85 25 45 42 D4 71 44 43 90 16

68 08 08 68 04 01 6C 0B 02 00 00 16 94 16
68 19 19 68 01 04 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0D 16
68 08 08 68 02 01 6C 0B 02 00 00 14 90 16
68 17 17 68 01 02 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0B 16
68 0A 0A 68 03 01 63 0C 02 00 00 02 00 00 77 16
E5
68 0E 0E 68 05 01 63 0C 02 30 00 06 33 66 99 00 00 00 DF 16
E5
68 0A 0A 68 04 01 63 0C 02 04 00 02 00 00 7C 16
E5
a 24 relací ... dokola ... dokola ...
```

Odposlech relací - projekt C3SABIO.TXT