TERM 06A

Sériový terminál

Příručka uživatele

Střešovická 49 , 162 00 Praha 6, e-mail: s o f c o n @ s o f c o n . c z
tel./fax : 220 610 348 / 220 180 454 , http :// w w w . s o f c o n . c z

Obsah:

Další přílohy

Terminál SCN 185.01 Sestava desky list 0

1. Úvod

Terminál **TERM 06A** je vylepšenou verzí terminálu Term06. Jedinými odlišnostmi jsou: jas podsvícení nelze měnit, zvukový výstup je typu otevřený kolektor nebo volitelně digitální výstup a Term06A lze rozšířit na Term06A/IO. **TERM 06A** je určen ke sledování a zadávání údajů přicházejících po sériové komunikační lince. Je vhodný pro průmyslové použití a uplatní se všude tam, kde je třeba sledovat a modifikovat technologické parametry nebo ovládat chod řídicích systémů, strojů a přístrojů. Pro zobrazení údajů slouží alfanumerický LCD displej o velikosti 2 řádky po 16 znacích. Displej je díky podsvětlení čitelný i ve tmě. Klávesnice je membránová s definovaným stiskem tlačítek. Sériová komunikační linka pracuje s rozhraním RS485 a umožňuje připojit více zařízení na jednu síť. Napájení terminálu je přivedeno kabelem spolu s komunikačními vodiči. Dodávané programové vybavení umožňuje zobrazovat přijmutá data na displeji a vysílat kódy stisknutých kláves. Uživatel může použitím vlastního programového vybavení vytvořit z terminálu ovládací a řídicí stanici schopnou komunikovat po sériové komunikační lince s nadřazeným systémem. Terminál TERM 06A v síťové verzi umožňuje připojení do sítě typu MASTER - SLAVE, která sestává až z 15 zařízení komunikujících s nadřazeným systémem formou zpráv, dle síťového komunikačního protokolu firmy SofCon, který je popsán níže.

Řídicí sekvence terminálu TERM 06A jsou shodné s terminálem TERM 01.

TERM 06A je svým mechanickým provedením určen k vestavbě, např. do panelu zařízení.

2. Popis

2.1 Popis HW

Terminál TERM06A je řízen interní procesorovou jednotkou. Její základ tvoří procesor AT89C51 nebo AT89C52, tedy klon populárního mikrořadiče i8051, ve standardním zapojení s krystalovým oscilátorem. Mikrořadič obsahuje paměť programu o velikosti 4 KB (8 KB pro AT89C52) přímo na čipu. Periferní obvody (klávesnice, LCD displej, sériová EEPROM, akustický měnič, watchdog, obvody sériové linky) jsou připojeny na brány procesoru.

Pro uložení konfiguračních údajů terminál nabízí sériovou paměť EEPROM o kapacitě 128 x 8 bit, jejíž obsah lze změnit v konfiguračním módu (Setup), z klávesnice nebo po sériové lince. Paměť obsahuje prostředky zabraňující náhodnému přepsání v ní uložených dat.

Pro zabezpečení chodu mikrořadiče obsahuje terminál jednoduchý hlídací obvod typu Watch-Dog. Tento obvod musí být neustále nastavován změnou úrovně výstupního signálu běžícím programem. Při nesplnění této podmínky, např. havarování programu, je generován signál RESET, který uvede procesor do výchozího stavu.

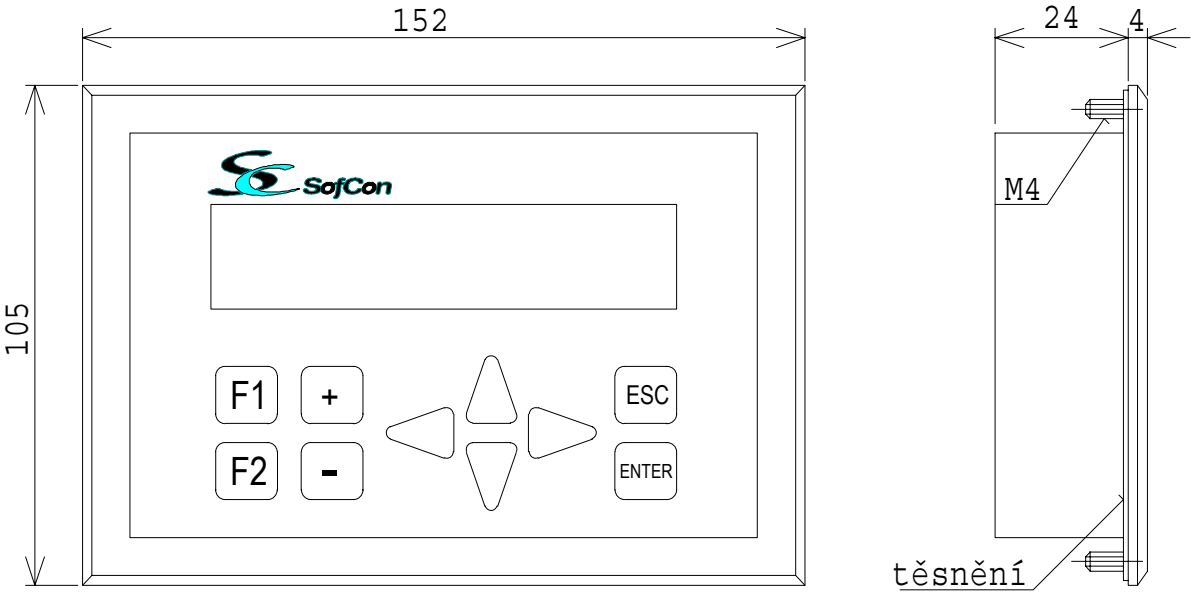
Alfanumerický displej LCD je připojen na bránu P2. Displej obsahuje vestavěný řadič, který zajišťuje zobrazení znaků a zprostředkovává pohyb kurzoru. LCD je typu DATA IMAGE CM1621 (nebo ekvivalent s řadičem Hitachi HD44780U), s vestavěným generátorem pevné sady znaků. Řízení LCD se provádí čtením a zápisem do instrukčních a datových registrů řadiče. Je použit displej transflexní s přisvětlením LED diodami. Intenzitu přisvětlení lze nastavit v konfiguraci (Setup) na jednu z osmi úrovní. Kontrast zobrazení lze korigovat trimrem R21.

Klávesnice je membránová s reliéfním hmatníkem. Tvořena je soustavou horizontálních a vertikálních vodičů, připojených ke konektoru X1. V místech křížení jsou na hmatníku výstupky, při jejichž stisku se vodiče propojí. Stisk tlačítka procesor vyhodnocuje postupným vysíláním logických úrovní z vývodů portů P0.0, P0.2, P0.4 až P0.6 a čtením odezvy na portech P0.1 a P0.3. Jako stisknuté vyhodnotí nejvýše jedno tlačítko.

Zvukovou signalizaci zajišťuje piezoelektrický měnič. Zvuk je generován periodickou změnou logické úrovně na portu P3.6.

Zelená LED dioda, připojená k vývodu P3.3 indikuje chod řídicího programu. Bliká s frekvencí asi 1 Hz.

Výstupní budiče sériové komunikační linky RS485 (směr přenosu) se ovládají logickou úrovní na portu P3.2.



Obr. 1 Základní rozměry terminálu TERM06A

2.2 Přehled rozložení kláves

Maticová klávesnice 10 tlačítek (2 × 5)

F1	+	↑	→	ENTER
F2	-	←	↓	ESC

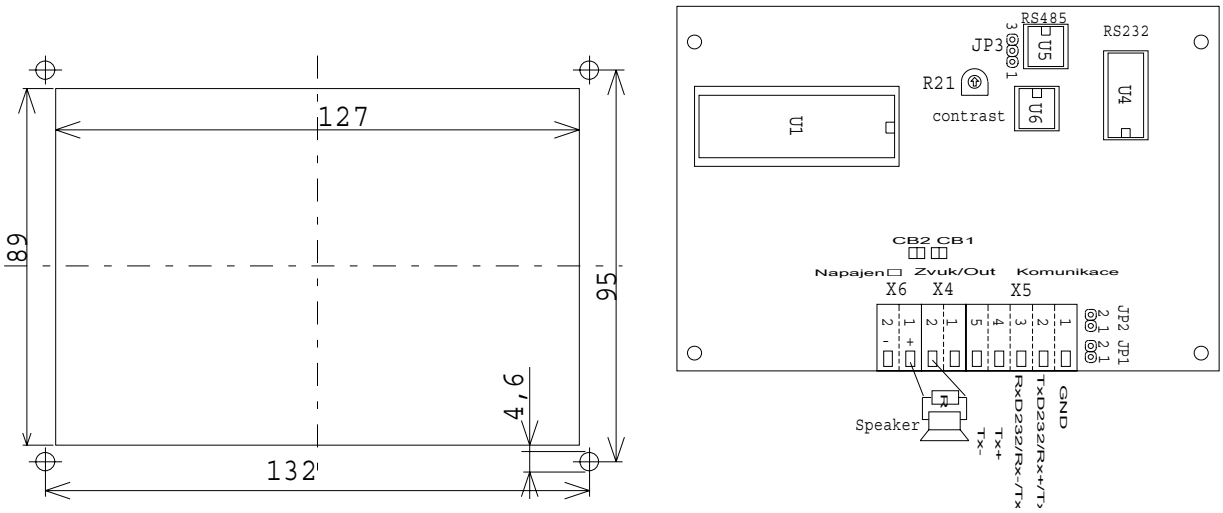
Přiřazení sepnutí kontaktů na konektoru X1 ke stisknutému tlačítku:

F1	1/6	+	2/6	↑	3/6	↓	7/6	ENTER	5/4
F2	1/4	-	2/4	←	3/5	→	7/4	ESC	5/6

3. Instalace

Terminál se připojuje prostřednictvím konektorů X6 (napájení), X5 (RS232 a RS485), a X4 (akustický měnič - Speaker). Jako konektory jsou standardně osazeny stiskací svorky (Wago), na objednávku lze provést osazení jiného typu konektoru, odpovídajícího rozměrově otvorům v plošném spoji a krytu terminálu. Napájecí napětí terminálu musí odpovídat parametrům podle kapitoly 7.1.

Jako budič sériové linky RS232 je použit obvod MAX232 nebo ekvivalent (obvod U4), pro sériovou linku typu RS485 tvoří budiče obvody 75176 nebo ekvivalentní (U5, U6). Tyto dva druhy budičů se osazují do objímek a nesmí být osazeny současně! Pro variantu RS485 dvoudrát není nutné osazovat obvod U6.



Obr. 2 Montážní rozměry a rozmístění propojek a konektorů.

3.1 Nastavení propojek

Na základní desce terminálu je několik propojek, které slouží k nastavení některých specifických vlastností.

JP1 Připojuje zakončovací odpor 220R mezi vodič RX/TX+ a napájecí napětí +5V.

JP2 Připojuje zakončovací odpor 220R mezi vodič RX/TX- a zem.

JP3 Přepíná mód RS 485: JP3/1-2 dvoudrát, JP3/2-3 čtyřdrát.

3.2 Zapojení přívodů

kabel RS232

signál	pin konektoru X5	PC - 9 pin	PC - 25 pin
RxD	3	3	2
TxD	2	2	3
GND	1	5	7

kabel RS485

signál	pin konektoru X5
Rx/Tx+	2
Rx/Tx-	3
Tx+	4
Tx-	5
GND	1

napájení

signál	pin konektoru X6	barva svorky
+ (Vss)	1	oranžová
- (GND)	2	modrá

digitální výstup/ akustický měnič – speaker (s paralelním odporem 4k7)

signál	pin konektoru X4	CB
Dout0+	1	CB1 ON
Dout0-	2	CB2 Off

signál	pin konektoru	CB
Sp1	X6-1 (+24V)	CB1 Off
Sp2	X4-2	CB2 ON

4. Programové vybavení verze ASCII terminálu

Řídicí program je napsán v instrukčním kódu mikrořadičů typu i51 a je uložen v interní paměti PROM. Zajišťuje veškeré funkce terminálu. V této paměti jsou také uloženy kódy klávesnice, všechny konstantní zprávy, které terminál zobrazuje nebo odesílá (např. uživatelské logo) a implicitní konfigurační konstanty.

Standardně je dodáváno programové vybavení, které umožňuje:

Přijímat znaky ze sériové komunikační linky a zobrazovat je na displej LCD.

Na komunikační linku vysílat kódy stisknutých tlačítek.

Měnit některé vlastnosti terminálu buď z klávesnice v servisním módu SETUP, nebo pomocí ESC sekvencí z komunikační linky.

Dále je popsáno, jak jsou určeny vlastnosti jednotlivých částí (klávesnice, displej) a funkce standardního řídicího programu (inicializace, Setup).

4.1 Displej

Přijímané znaky v rozsahu 20h až 7Fh jsou zobrazovány ve významu ASCII, další jako semigrafické. Některé znaky jsou rozpoznávány jako řídící, a to přímo nebo jako součást ESC sekvence.

Význam řídicích znaků je v následující tabulce:

Řídicí funkce	Kód znaku		
	H	D	CTRL
HOME	C7	199	
→	09 CD	09 205	I
←	08 CB	08 203	H
↑	0B C8	11 200	K
↓	0A D0	10 208	J
LF	0A	10	-
CR	0D	13	M
CR + LF	-	-	-
HOME + clr	1E	30	^
BELL	07	07	G
ESC	1B	27	[

Pozn:	HOME	kurzor do levého horního rohu
	HOME + clr	smazání displeje, kurzor do levého horního rohu
	CR	kurzor na začátek řádku
	LF	kurzor o řádek dolů, jestliže byl na posledním, tak rolování
	→	kurzor o jeden znak doprava
	←	kurzor o jeden znak doleva
	↑	kurzor o řádek nahoru
	↓	kurzor o řádek dolů
	BELL	pípnutí
	ESC	start Escape sekvence (speciální řídicí a nastavovací funkce)

Tabulka ESC sekvencí:

Řídicí funkce	Vstupní ESC sekvence
Kurzor na řádek (r) a sloupec (c) r = ⟨0,3⟩, c = ⟨0,15⟩	ESC Y (r) (c)
Mazání od kurzoru do konce řádku	ESC K
Mazání od kurzoru do konce stránky	ESC J
↑	ESC A
↓	ESC B
→	ESC C
←	ESC D
HOME	ESC H
Kurzor OFF	ESC 1
Kurzor ON blikající	ESC 2
Kurzor ON neblikající	ESC 3
Dotaz na verzi terminálu	ESC V
Zápis klíče (k) k = bin. číslo ⟨0,255⟩ *)	ESC ! N (k)
Čtení klíče	ESC ! n
Report (přihlášení)	ESC ! r
Konfigurace terminálu (viz kap. 6.3)	ESC S ...

Pozn.: *) Tyto funkce trvají až 15 milisekund.
⟨…⟩ označuje binární číslo

Kurzor OFF	Vypnutí kurzoru.
Kurzor ON	Zapnutí (zviditelnění) kurzoru
Verze terminálu	Vyšle se řetězec znaků s označením verze programového vybavení.
Klíč	Jeden znak, který se může použít pro identifikaci terminálu.
Report	Pevně naprogramovaný řetězec znaků, který je z terminálu vyslán touto funkcí nebo při Resetu (zapnutí napájení).

4.2 Klávesnice

Při stisku tlačítka klávesnice je vyslán odpovídající kód znaku na sériovou komunikační linku. Při trvalém držení je v činnosti opakovací funkce AUTOREPEAT, to znamená, že znak je zvýšenou rychlostí trvale opakován. Při každém odeslání kódu klávesy se ozve krátké pípnutí.

Kód znaků je určen kódovací tabulkou, jejíž změny lze specifikovat při objednání terminálu. Protože tato tabulka je uložena v interní paměti programu, není možné ji měnit bez přeprogramování procesoru. Tabulka znaků klávesnice představuje funkci jednoznačně přiřazující stisknutému tlačítku vysílaný kód znaku. Programové vybavení standardně obsahuje kódy DEC, ENH DEC (pozměněné funkční kódy), IBM PC (ASCII a funkční klávesy podle IBM PC dvouznakové) a USER. Není však problém je nahradit libovolným kódem (jeden nebo dva odeslané znaky na stisk klávesy). Kódy USER standardně odpovídají SCAN CODE, podle specifikace uživatele mohou být libovolné jiné.

Pro vnitřní reprezentaci znaků je používán tzv. SCAN CODE tlačítek. Ten odpovídá pořadovým číslům tlačítek tak, jak jsou řídicím programem snímána.

Překódovávací tabulka pro 10 tlačítkovou klávesnici (dekadicky):

Klávesa		→	↓	Enter	ESC	↑	←	+	-	F1	F2
SCAN CODE	1 byte	01	02	03	04	05	06	07	08	09	10
DEC	1 byte	09	10	13	27	11	08	43	45	00	00
ENH.DEC	1 byte	205	208	13	27	200	203	43	45	187	188
IBM PC	1-2 byty	00 77	00 80	13	27	00 72	00 75	43	45	00 59	00 60
USER	1 byte	01	02	03	04	05	06	07	08	09	10

Všechny kódy kláves jsou umístěny v interní paměti programu. Je možné je změnit přepsáním obsahu na adresách dle kapitoly 6.4

4.3 Časové nároky na zpracování znaků a funkcí

Některé řídicí funkce (např. rolování řádků nahoru) jsou časově náročné a proto, aby se ani při nejvyšších přenosových rychlostech neztratily přijímané znaky, je terminál vybaven vyrovnávací pamětí pro dočasné uložení přijímaných znaků. Některé funkce, při kterých je zapisováno do konfigurační EEPROM, trvají i desítky milisekund. Po těchto příkazech je nutno alespoň 100 ms do terminálu nic neposílat, jinak budou znaky, které se již nevejdou do přijímacího bufferu ztraceny. Doby, potřebné pro provedení funkcí jsou v příslušných odstavcích uvedeny poznámkou.

Pokud je terminálu zasílán obsah celého displeje najednou, je vhodné při komunikačních rychlostech větších než 9600 Bd mezi zasílanými zprávami ponechat prodlevu alespoň 10 ms na zpracování přijatých znaků a jejich zobrazení na displeji.

5. Programové vybavení síťové verze terminálu

Řídicí program je napsán v instrukčním kódu mikrořadičů typu i51 a je uložen v interní paměti PROM. Zajišťuje veškeré funkce terminálu. V této paměti jsou také uloženy kódy klávesnice, všechny konstantní zprávy, které terminál zobrazuje nebo odesílá (např. uživatelské logo) a implicitní konfigurační konstanty.

Standardně je dodáváno programové vybavení, které umožňuje:

Přijímat zprávy ze sériové komunikační linky a vykonávat funkce dle jejich znění.

Odesílat zprávy masteru jako odpovědi na přijaté zprávy.

Měnit některé vlastnosti terminálu buď z klávesnice v servisním módu SETUP, nebo pomocí ESC sekvencí ve zprávě s identifikátorem C1h.

Dále je popsáno, jak jsou určeny vlastnosti jednotlivých částí (klávesnice, displej) a funkce standardního řídicího programu (inicializace, Setup).

5.1 Displej

Jsou-li ve zprávě s identifikátorem C1h znaky v rozsahu 20h až 7Fh jsou zobrazovány ve významu ASCII, další jako semigrafické. Některé znaky jsou rozpoznávány jako řídící, a to přímo nebo jako součást ESC sekvence.

Význam řídicích znaků je shodný s ASCII terminálem.

Tabulka ESC sekvencí:

Řídicí funkce	Vstupní ESC sekvence
Kurzor na řádek (r) a sloupec (c) r = ⟨0,3⟩, c = ⟨0,15⟩	ESC Y (r) (c)
Mazání od kurzoru do konce řádku	ESC K
Mazání od kurzoru do konce stránky	ESC J
↑	ESC A
↓	ESC B
→	ESC C
←	ESC D
HOME	ESC H
Kurzor OFF	ESC 1
Kurzor ON blikající	ESC 2
Kurzor ON neblikající	ESC 3
Zápis klíče (k) k = bin. Číslo ⟨0,255⟩ *)	ESC ! N (k)
Konfigurace terminálu (viz odst. **Setup**)	ESC S ...

Pozn.: *) Tyto funkce trvají až 15 milisekund.

⟨…⟩ označuje binární číslo

Kurzor OFF	Vypnutí kurzoru.
Kurzor ON	Zapnutí (zviditelnění) kurzoru
Klíč	Jeden znak, který je použít pro identifikaci terminálu – TERM_ID

5.2 Klávesnice

Při stisku tlačítka klávesnice je uložen odpovídající kód znaku do bufferu klávesnice a po dotazu je v odpovědi odeslán jeho obsah. Při trvalém držení je v činnosti opakovací funkce AUTOREPEAT, to znamená, že znak je zvýšenou rychlostí trvale ukládán do bufferu.

Kódovací tabulka je shodná s ASCII verzí terminálu. Viz. kapitola 4.2

5.3 Popis komunikace

Terminál komunikuje s nadřazeným systémem pomocí zpráv jejichž formát je pevně dán a je popsán komunikačním protokolem firmy SofCon formátu DF0 uvedeným v příloze. Komunikační protokol popisuje komunikační vrstvy fyzické, linkové a síťové.

Fyzická vrstva je tvořena rozhraním RS485 pro které je v programovém vybavení firmy SofCon s.r.o vytvořena komunikační knihovna ChnCom.

Linková vrstva popisuje základní rámec přenášených dat. Obsahuje definici řídících znaků, způsob adresace v komunikační síti, zabezpečení přenášených dat, délku bloků a zajištění transparence dat. Všechny přenosy se uskutečňují pomocí zpráv mající níže popsaný rámec, zabezpečení a transparentnost. Pro tuto linkovou vrstvu je v programovém vybavení firmy SofCon s.r.o vytvořena komunikační knihovna ChnPrt.

Síťová vrstva popisuje sémantiku přenosu libovolných zpráv oběma směry po síti Master - Slave resp. Master - množina Slave. Definuje formáty a obsahy identifikačních a datových polí, způsoby potvrzování, zabezpečení opakování atd.

Komunikace probíhá formou dotazů a odpovědí. Master vyšle terminálu zprávu, jejíž identifikátor terminálu slouží jako příkaz, který po přijetí zprávy terminál začne okamžitě vykonávat. Po vykonání příkazu obdrží master od terminálu odpověď. Po té může master vyslat další zprávu, nebo oslovit jiné zařízení. Pokud odpověď od terminálu master nedostane ve stanovené lhůtě, je třeba zaslat dotaz znova.

5.4 Typy zpráv

Terminál může pracovat ve dvou režimech činnosti: v normálním režimu a v režimu lokální editace. Typy zpráv, které jsou pak mezi masterem a terminálem zasílány se dle těchto režimů liší. Zpráva, která má identifikátor jiný než níže uvedený, je ignorována. Pokud je zpráva zasílána na adresu NODE=0, pak je přijímána všemi zařízeními na síti, ale žádný z nich na ni neodpovídá. Terminál však bude akceptovat pouze dva typy zpráv zaslaných na adresu NODE=0. Zprávu s identifikátorem Cmd_disp v normálním režimu a Cmd_snm v režimu lokální editace.

5.4.1 Normální režim

V tomto režimu jsou mezi masterem a terminálem zasílány tyto dvojice zpráv:

Master → Terminál		Terminál → Master	
Identifikátor	Funkce	Identifikátor	Funkce
Cmd_disp C1h	Zápis obsahu zprávy na displej	Cmd_key C2h	Zaslání obsahu bufferu klávesnice
Cmd_gsts C5h	Dotaz na stav terminálu	Cmd_sts C6h	Stav terminálu
Cmd_slm C7h	Přechod do režimu lokální editace	Cmd_sts C6h	Stav terminálu
Cmd_geb C9h	Čtení obsahu editačního bufferu	Cmd_eb CAh	Obsah editačního bufferu

5.4.2 Režim lokální editace

V tomto režimu jsou mezi masterem a terminálem zasílány tyto dvojice zpráv:

Master → terminál		Terminál → Master	
Identifikátor	Funkce	Identifikátor	Funkce
Cmd_gsts C5h	Dotaz na stav terminálu	Cmd_sts C6h	Stav terminálu
Cmd_geb C9h	Čtení obsahu editačního bufferu	Cmd_eb CAh	Obsah editačního bufferu
Cmd_snm CBh	Přechod do normálního režimu	Cmd_eb CAh	Obsah editačního bufferu

5.4.3 Popisy zpráv přijímaných terminálem

- Zápis obsahu zprávy na displej - identifikátor **Cmd_disp**

CDATA

Pole	Význam
CDATA	Data vypisovaná na displej nebo řídící kódy a ESCape sekvence. Délka <0,40> bytů

- Dotaz na stav terminálu – identifikátor **Cmd_gsts**

CDATA

Pole	Význam
CDATA	Pole není využito.

- Přechod do režimu lokální editace – identifikátor **Cmd_slm**

CDATA				
POS	EDM	EDDATA		
		NUM	LL	HL

Pole	Význam	
POS	Poloha editované položky na displeji <0,31>, 1 byte	
EDM	Byte určující editační mód: 00h … Editace stringu – zobrazované znaky leží v rozsahu <20h-7Fh> 01h … Editace stringu – zobrazované znaky leží v rozsahu <30h-39h> číslice v ASCII 02h … Editace stringu – zobrazované znaky leží v rozsahu <40h-5Ah> velká písmena v ASCII 10h … Editace čísla (4 byte)	
EDDATA	Dle režimu editace (EDM) má pole EDDATA význam: ♦ pro EDM <00h,02h> má EDDATA <1,16> bytů, je to přímo editovaný řetězec ♦ pro EDM = 10h má EDDATA délku 12 bytů s tímto významem:	
	NUM	Inicializační hodnota – 4 byty, musí platit LL ≤ NUM ≤ HL
	LL	Dolní mez editovaného čísla – 4 byty – editované číslo bude větší nebo rovno. Musí platit: LL ≥ 00h
	HL	Horní mez editovaného čísla – 4 byty – editované číslo bude menší nebo rovno. Musí platit: HL ≤ 3B 9A CA 00 h = 1e9 d

- Čtení obsahu editačního bufferu – identifikátor **Cmd_geb**

CDATA

Pole	Význam
CDATA	Pole není využito.

- Přechod do normálního režimu – identifikátor **Cmd_snm**

CDATA

Pole	Význam
CDATA	Pole není využito.

5.4.4 Popis zpráv zasílaných terminálem

- Zaslání obsahu bufferu klávesnice– identifikátor **Cmd_key**

CDATA

Pole	Význam
CDATA	Kódy stisknutých kláves. <1,8> bytů

- Zaslání obsahu editačního bufferu – identifikátor **Cmd_eb**

CDATA

Pole	Význam
CDATA	obsah EDDATA - význam viz Cmd_slm - viz výše. <1,16> bytů

- Stav terminálu – identifikátor **Cmd_sts**

CDATA 3 byty				
KONF1	KONFB			VER
	B7 (MSB)	B6	…	B0 (LSB)

Pole	Význam	
KONF1	Konfigurační byte popsaný v části SETUP viz níže.	
KONFB	Konfigurační byte KONF3 popsaný v části SETUP doplněný o dva nejvyšší bity mající následující význam:	
	B7	Bit určující režim terminálu. Je-li bit B7=1, pak je terminál v režimu lokální editace. Je-li bit B7=0, pak je terminál v normálním režimu.
	B6	Bit určující způsob ukončení lokální editace. Je-li bit B6=1, pak lokální editace byla opuštěna stiskem klávesy ESC nebo byla ukončena z MASTERU zprávou s identifikátorem CBh. Je-li bit B6=0, pak lokální editace byla korektně ukončena stiskem klávesy ENTER.
VER	Verze firmware. 1 Byte.	

5.5 Časové nároky na zpracování zpráv

Terminál po obdržení zprávy začne vykonávat danou funkci a v této době nepřijímá žádné další zprávy. V tomto stavu setrvává až do odeslání odpovědi masteru. Proto je třeba zajistit, aby master neoslovoval, do doby než obdrží odpověď od terminálu, žádné zařízení na síti. Pokud master neobdrží ani po 100ms od terminálu odpověď, může považovat zprávu zaslanou terminálu za nedoručenou a zaslání zprávy opakovat.

6. SETUP terminálu

Softwarová konfigurace (SETUP) se uskutečňuje z klávesnice přes menu, které je vyvoláno přidržením klávesy F2 při zapnutí terminálu, nebo dálkově po komunikační lince ESC sekvencemi - ESC S. Konfiguraci lze kdykoliv uložit do paměti EEPROM. Po zapnutí napájení se konfigurace automaticky obnovuje čtením EEPROM. V případě neplatného obsahu EEPROM, nebo není-li EEPROM přítomna, je načtena implicitní konfigurace z paměti programu. Platnost dat v EEPROM je hlídána identifikačním znakem a kontrolním součtem.

Parametry, které se nastavují v režimu SETUP:

Zdroj konfigurace	**implicitní**, uživatelská (EEPROM)
Přenosová rychlost kom. linky	300, 600, 1 200, 2 400, 4 800, **9 600**, 19 200, 57 600 Bd
Verifikace přenosu	sudá parita, **lichá parita**, žádná
Počet stop bitů	**jeden**, dva
Podoba kurzoru	neblikající, blikající, **neviditelný**
Identifikační klíč terminálu	*ASCII verze*: hodnota <0,255> (implicitně znak "6") *síťová verze*: hodnota <0,15> (implicitně 15)
Intenzita podsvětlení	hodnota v intervalu 0 – 7 (implicitně **7**, tedy maximální)
Time-out pro zhasnutí podsvětlení	trvale vypnuto, 10, 30, 60, 120, 180, 240 s **trvale zapnuto**

Tučně je uvedena implicitní konfigurace terminálu.

6.1 Implicitní konfigurace

Je určena obsahem 4 bajtů (KONF1, KONF2, KONF3, TERM_ID), které jsou uloženy v interní paměti procesoru na adresách dle kapitoly 6.4. Konfiguraci lze měnit bez překladu řídicího programu. Při nastavení zámku konfigurace je možné ji změnit pouze pomocí ESC sekvence.

KONF1 = ZEXXKKCC, kde:

Z = zámek konfigurace z klávesnice	E = zdroj konfigurace	KK = typ kódování klávesnice	CC = podoba kurzoru
0 odemčeno	0 implicitní	00 IBM PC	00 neviditelný
1 zamčeno	1 uživatelská	01 DEC	01 viditelný
		10 ENH DEC	1x blikající
		11 uživatelská	

XX - rezerva (musí být 00), x – nezáleží na hodnotě

KONF2 = XRSPPBBB, kde:

S = počet stop bitů	P = parita	BBB = přenosová rychlost
0 1 bit	0x bez parity	000 300 Bd
1 2 bity	10 lichá (odd)	001 600 Bd
	11 sudá (even)	010 1 200 Bd
		011 2 400 Bd
		100 4 800 Bd
		101 9 600 Bd
		110 19 200 Bd
		111 57 600 Bd

KONF3 = XXLLLIII, kde:

```
LLL = doba svícení              III = intenzita podsvětlení
000 -   0 s                     000 - vypnuto
001 -  10 s                     ...
...                             011 - vypnuto
110 - 180 s                     100 - maximální jas
111 - time-out vypnut           ...
XX - rezerva (musí být nula)    111 - maximální jas
```

TERM_ID = binární kód identifikačního klíče terminálu, u síťové verze slouží jako adresa terminálu na síti

6.2 SETUP z klávesnice

Do režimu Setup z klávesnice se vstupuje přidržením klávesy F2 při startu terminálu (po zapnutí napájení). V tomto režimu lze měnit všechna nastavení terminálu včetně parametrů přenosové linky, které není dovoleno měnit jiným způsobem. Pomocí šipek nahoru a dolů lze rolovat v textové nabídce, kde se v horním řádku displeje zobrazují jednotlivé položky Setup a ve spodním jejich aktuální nastavení. Pomocí šipek → ← lze měnit nastavení dané položky. Mód konfigurace lze opustit několika způsoby:

- Kdykoliv klávesou Enter – terminál pokračuje ve startovací sekvenci s nově nastavenou konfigurací. Konfigurace se neukládá, ale je terminálem používána až do dalšího restartu (vypnutí napájení). Lze ji uložit pomocí escape sekvence ESC-S-W.
- Kdykoliv klávesou ESC – nastavená konfigurace se neukládá, terminál provede restart a nastartuje s původní konfigurací.
- Zvolením položky Save setup a stisknutím klávesy Enter – dojde k uložení konfigurace do paměti EEPROM a restartu terminálu pro správné načtení nové konfigurace.
- Po uplynutí prodlevy 20 s bez stisku klávesy – dojde k restartu terminálu a načtení původní konfigurace.

Případná chyba při ukládání konfigurace je indikována akusticky a nápisem na displeji.

Pokud je nastaven zámek konfigurace z klávesnice, na displeji se před vstupem do setupu objeví hláška „Setup is locked – read only“. Po stisku jakékoli klávesy je zpřístupněn setup bez možnosti změny konfigurace. Opustit jej lze výše uvedenými způsoby, vyjma uložení konfigurace. Tato funkce je určena pro kontrolu nastavení terminálu.

6.3 SETUP po komunikační lince

Podobně jako z klávesnice lze měnit konfiguraci po komunikační lince. K tomu slouží příkazy ESC S.

Konfigurace	ESC sekvence
Intenzita podsvětlení n = ⟨0,7⟩ viz KONF3	ESC S I (n)
Prodleva pro vypnutí podsvětlení n = ⟨0,7⟩ viz KONF3	ESC S L (n)
Report **) n = ⟨0,1⟩ viz KONF1	ESC S P (n)
Podoba kurzoru n = ⟨0,3⟩ viz KONF2	ESC S C (n)
Zámek SETUP z klávesnice n = ⟨0,1⟩ viz KONF1	ESC S Z (n)
Zápis konfigurace do EEPROM *)	ESC S W
Čtení původní konfig. z EEPROM *)	ESC S R
Volba kódování klávesnice n = ⟨0,3⟩ viz KONF1	ESC S K (n)

Pozn.: *) Tyto funkce trvají až 100 ms.

**) Dostupné jen ve verzi ASCII

⟨…⟩ - rozsah udáván jako binární hodnota, hodnoty mimo uvedený rozsah jsou oříznuty.

6.4 Důležité konfigurační konstanty

Dále jsou uvedeny konfigurační konstanty a jejich adresy v interní paměti programu. Pro jejich snadné nalezení v binárním souboru jsou před nimi textové řetězce. Má-li být implicitní konfigurace odlišná od standardní (kap. 7.2) je třeba obsah těchto konstant změnit.

Uloženo za řetězcem "CONFIG:" (délka 4byty)

KONF1 - implicitní konfigurace

KONF2 - implicitní konfigurace přenosového kanálu

KONF3 – implicitní konfigurace

TERM_ID - identifikační klíč terminálu, adresa terminálu v případě síťového protokolu. V ASCII verzi aktivní odpověď terminálu na dotaz ESC ! n. Normálně je 'klíč' zapisován ESC ! N sekvencí do EEPROM. Není-li EEPROM z nějakého důvodu dostupná, bere se odpověď z tohoto bytu.

Uloženo za řetězcem "REPORT:"

Report (přihlášení), řetězec 1 až 21 bajtů, poslední bajt musí být nulový!

Tento řetězec se vyšle po resetu terminálu (zapnutí napájení). Znaky jsou odesílány dokud program nenarazí na hodnotu 00 (vysílá se jako ukončovací znak řetězce). Ve standardním programovém vybavení je jako Report naprogramován řetězec "SofCon TERM06 Ready". Tato funkce je implementována pouze v ASCII verzi terminálu.

Uloženo za řetězcem "ULOGO:"

Uživatelské logo, zobrazené po zapnutí na displeji. Délka přesně 16 znaků. Ve standardním programovém vybavení je řetězec " S o f C o n ".

Uloženo za řetězcem "USER KEY:_"

0AAh Délka přesně 10 bytů. Uživatelsky definované kódy klávesnice, ve standardním programovém vybavení odpovídají scan kódu.

! Upozornění !
V případě, že uživatel provede přeprogramování interní paměti terminálu, ztrácí tím záruku na programové vybavení přístroje.

6.5 Verze programového vybavení

Programové vybavení je dodáváno ve dvou verzích. V síťové verzi a ve verzi pro ASCII terminál.

Po resetu terminálu (zapnutí napájení) se pro kontrolu na 1 vteřinu rozsvítí všechny body displeje, poté je na vteřinu zobrazeno zákaznické logo, buď "SofCon" nebo speciální zákaznické, doplněné číslem verze programového vybavení, např. "TERM06 SCG185.01". Poté se na okamžik objeví logo "SofCon".

Při vstupu do režimu SETUP je zobrazeno datum verze HW a SW. Po komunikační lince se lze na verzi programového vybavení dotázat zasláním zprávy s identifikátorem C5h viz výše nebo ESC sekvencí ESC V ve verzi softwarového vybavení ASCII.

7. Základní technické údaje

7.1 Provozní podmínky

Zařízení je konstruováno jako elektrický předmět třídy III podle ČSN EN 33 0600

Provoz	nepřetržitý
Prostředí	průmyslové, neklimatizované, bez agresivních plynů a par, stupeň znečistění 2
EMC	Zařízení třídy A podle ČSN EN 55 022 určené pro průmyslové prostředí, emise podle ČSN EN 50 081-2, odolnost podle ČSN EN 61000-6-2 (ČSN EN 50 082-2)
Provozní teplota okolí	0 až 50°C
Relativní vlhkost vzduchu	40 až 95% při 25°C
Atmosférický tlak	80 až 107 kPa
Pracovní vibrace	max. 0,15 mm při 55Hz
Napájení	ze zdroje malého bezpečného napětí (PELV) podle ČSN 332000-4, nestabilizované, stejnosměrné, 12 - 35 V

7.2 Technické parametry

Rozměry	154 × 105 × 40 mm
Hmotnost	0,3 kg
Krytí	IP 20, přední panel IP 65
Napájecí proud	100 mA při rozsvíceném displeji, napájení 12 V
(při RS232)	180 mA při zvýšeném jasu, napájení 12 V
(při RS485 + 30mA)	50 mA při rozsvíceném displeji, napájení 24 V
	90 mA při zvýšeném jasu, napájení 24 V
displej	LCD transflexní s přisvětlením LED (čitelný i ve tmě), žlutozelený
	2 řádky × 16 znaků, velikost znaku 8x5 mm, písmo latinka
klávesnice	membránová s definovaným stiskem, 10 tlačítek
kódování	**ENH DEC**, USER (SCAN), DEC nebo IBM PC lze nastavit v Setup
	volitelně: uživatelské kódování místo SCAN kódů (nutno specifikovat přiřazení kódů)
komunikace	asynchronní, plný duplex
rozhraní	volitelně: **RS232**, RS485 dvoudrát nebo čtyřdrát pro ASCII verzi
	RS485 dvoudrát nebo čtyřdrát pro síťovou verzi
přenosová rychlost	300, 600, 1200, 2400, 4800, **9600**, 19200, 57600 Bd
počet inform. bitů	8 + 1 paritní (pokud je nakonfigurován)
počet stop bitů	**1**, 2
zabezpečení přenosu	**parita lichá**, sudá, bez parity
vyrovnávací paměť	40 bytů pro zprávu
procesor	AT89C51 resp. AT89C52
frekvence krystalu	11,0592 MHz
konfigurační paměť	sériová EEPROM
zvuková signalizace	**bez zvukové signalizace**
	volitelně: piezoelektrický měnič

Pozn.: V případě více možností jsou standardní vlastnosti a nastavení **zvýrazněny**. Rozhraní je nutno specifikovat v objednávce, ostatní vlastnosti je možno nastavit v SETUPu.

8. Objednávání

Na objednávce nutno specifikovat požadovaný typ přístroje:

- rozhraní RS232, ASCII verze
- rozhraní RS485, ASCII verze
- rozhraní RS485, síťová verze

Příklad objednávky 1:

Terminál TERM 06A, provedení standardní ASCII terminál RS232.

9. Přílohy

9.1 Popis komunikačního protokolu firmy *SofCon® s.r.o.*

V dokumentu jsou popsány komunikační vrstvy fyzické, linkové a síťové pro přenos zpráv mezi paralelně běžícími procesy používanými ve firmě *SofCon® s.r.o.* .

Fyzická vrstva popisuje, přes která rozhraní je možno komunikaci provést.

Linková vrstva popisuje základní rámec přenášených dat. Obsahuje definici řídících znaků, způsob adresace v komunikační síti, zabezpečení přenášených dat a zajištění transparence dat.

Síťová vrstva popisuje sémantiku přenosu libovolných zpráv oběma směry po síti Master - Slave resp. Master - množina Slave. Definuje formáty a obsahy identifikačních a datových polí, způsoby potvrzování, zabezpečení opakování, ...

V poslední kapitole jsou pak omezení jednotlivých zpráv při komunikaci s KIT-BUILDER a při programování v PASCALu.

Fyzická vrstva

Fyzická komunikační vrstva může být tvořena rozhraním RS232, RS485, RS422, telefonním modemem, GSM modemem, radiovým modemem a jiným.

Pro tyto fyzické vrstvy jsou v programovém vybavení firmy SofCon vytvořeny komunikační knihovny ChnCom, ChnCom2, ChnV40, ChnComM, ChnV40M, ChnRacom.

Linková vrstva

Linková vrstva definuje pravidla přenosu mezi dvěma uzly, délku bloků, způsob zabezpečení dat zajištění transparence přenášených dat. Je zde popsán linkový protokol používaný ve firmě *SofCon® s.r.o.* .

Základní struktura

Všechny přenosy se uskutečňují pomocí zpráv mající níže popsaný rámec, zabezpečení a transparentnost.

Pro tuto linkovou vrstvu je v programovém vybavení firmy SofCon vytvořena komunikační knihovna ChnPrt.

	Rámec zprávy						
délka v bytech	1	1	1	2	LEN	2	1
obsah	SOH	DNODE	SNODE	LEN	DATA	CRC	ETX

SOH počátek zprávy, 001h
DNODE fyzická adresa příjemce v síti <0,255>
Je-li adresa rovna 0, pak přijímají všichni ale na zprávu žádný neodpovídá.
SNODE fyzická adresa odesílatele v síti <0,255>
LEN délka zprávy DATA v bytech <0, 32000>
DATA vlastní datová zpráva
CRC zabezpečení dat CRC16 = x16+x15+x2+1
ETX konec zprávy, 003h

Pravidla

1) Přenosový rámec je shodný při komunikaci Master - Slave i Slave - Master
2) Zbytek po dělení je generován z SOH, DNODE, SNODE, LEN a DATA.

3) Pole SNODE je plněno číslem uzlu, který zprávu odesílá.

4) U polí LEN a CRC je nižší byte je zasílán první.

5) Je-li adresa adresáta DNODE rovna 0, pak přijímají všichni účastníci na komunikační lince, ale na zprávu žádný neodpovídá.

Zajištění transparence přenášených dat

Při vysílání zprávy jsou jednotlivé byte dále zakódovány takto:

SOH - zakódován jako dvojice DLE,SOH -> 010h,001h

ETX - zakódován jako dvojice DLE,ETX -> 010h,003h

ostatní jednotlivé byte zprávy (DNODE, SNODE, LEN, DATA, CRC), mající hodnotu DLE jsou zakódovány jako DLE,DLE -> 010h,010h.

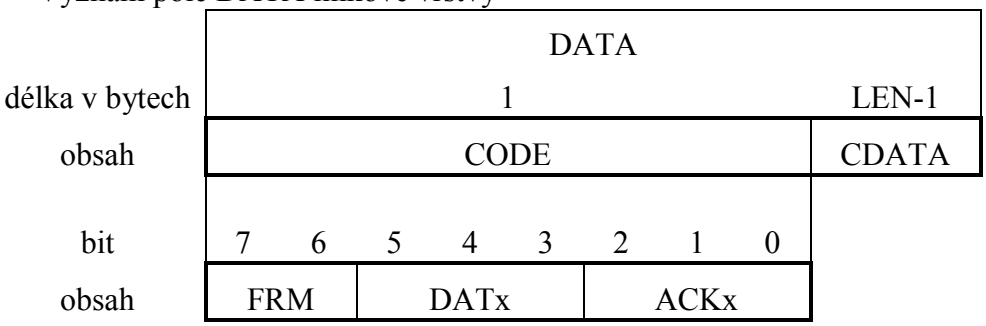
Tím je umožněno zkonstruovat přijímač tak, aby byl v proudu znaků schopen poznat počátek zprávy (vždy kombinace DLE,SOH). Jako další byte zprávy následuje DNODE=adresa přijímače. (ve výjimečném případě DNODE=DLE zakódovaném jako DLE,DLE). Pokud zpráva není určena pro příslušný přijímač, může být kompletně příjem ignorován až do opětovného nalezení počátku zprávy.

Síťová vrstva

Základní struktura

Protokol musí být schopen přenášet obousměrně zprávy CDATA mezi dvěma komunikujícími procesy.

Význam pole DATA linkové vrstvy



význam pole FRM

FRM		
hodnota	název	formát přenášených dat
00	TST	test spojení, data CDATA nejsou obsažena
01	DF1	data formát 1, bez logických adres
10	DF2	data formát 2, s logickými adresami
11	DF0	data formát 0, FRM(11)+DATx+ACKx ve významu identifikátoru zprávy

význam pole DAT

DAT		
hodnota	název	číslo zprávy
000	DAT0	není zasílána žádná zpráva CDATA
001 až 110	DATx	je zasílána zpráva CDATA číslo 1až 6, vyžadováno ACKx
111	DAT7	je zasílána zpráva CDATA číslo 7, nevyžadováno ACK

Pro číslování zpráv platí :

NEWčíslo = (OLDčíslo mod 6) + 1

význam pole ACK

ACK		
ACK	název	číslo potvrzení ACK
000	ACK0	neposíláno žádné potvrzení
001 až 110	ACKx	posíláno potvrzení ACK na došlou zprávu č.1až 6
111	NACK	posíláno negativní potvrzení

Síťová verze terminálu komunikuje síťovým protokolem pouze ve formátu DF0.

Pravidla

1) Komunikace na komunikační síti je MASTER - SLAVE.
2) MASTER zasílá rámce, SLAVE zasílá rámce jako odpovědi.
3) SLAVE komunikuje pouze jedním typem formátu dat DF0
4) Při zaslání odpovědi SLAVE uzlem je pole DNODE linkové vrstvy naplněno hodnotou přijatou v poli SNODE.

Obsah datové části CDATA

Při formátu DF0 přenášených dat zajišťuje aplikační proces naplnění obsahu datové části DATA a adresu cílového uzlu DNODE.

Data formát 0

Protokol formátu 0 - DF0 musí být schopen přenášet zprávy CDATA mezi dvěma uzly sítě, z nichž jeden je ve funkci MASTER a druhý ve funkci SLAVE. Není zde zavedeno potvrzování zasílaných a přijímaných rámců (komunikace na úrovni linkové vrstvy). Potvrzování a zabezpečení může být realizováno přímo komunikujícím procesem v MASTER uzlu.

	CDATA formát 0 - DF0
délka v bytech	LEN-1
obsah	REC

Identifikátor zprávy je dán polem CODE = FRM(11) + DATx + ACKx. Vyjadřuje zároveň 2 informace :

- typ rekordu přenášených dat (vyjadřuje počet, délku a sémantický význam jednotlivých položek data-rekordu).
- sémantiku, která se má provést po příjmu zprávy (test komunikace, žádost o data, zaslání dat).

REC vlastní přenášená data. Význam jednotlivých položek rekordu a jejich celkový počet je určen konkrétní hodnotou typu TREC.

Délka REC je dána velikostí LEN-1, sémanticky musí souhlasit s předpokládanou délkou v závislosti na použitém identifikátoru zprávy.

Příklady zpráv

Přenosový rámec - linková vrstva

Zaslání rámce z uzlu 5 do uzlu 6, pole DATA obsahuje 6 byte „1,2,3,4“

SOH	DNODE	SNODE	LEN		DATA	CRC		ETX
			LO	HI		LO	HI	
001h	6	5	4	0	1,2,3,4	?	?	003h

9.2 Algoritmus výpočtu CRC16

9.2.1 Pro mikroprocesory řady 8051

```
;funkce CRC16 zajistuje vypocet zbytku po deleni generujicim polynomem

;uzivatel musi zajistit vynulovani promenne Residue pri vypoctu CRC
;pro novy balik dat

;naroky na pamet
;2B RAM - ulozeni zbytku (Residue)
;Stack - 2B navratova adresa + 1B na prubezne ukladani

;!!Residue data 32h    ;zbytek po deleni (word)

;procedura pro vypocet CRC-16
;parametry predavany v dvojici ACC (Data:Byte) a Residue (Zbytek:Word)
CRC16:

        push acc   ;@emr
        push 00
        push 01

        xrl A,Residue  ; xor al,byte ptr es:[bx].Residue
        mov R1,A
        clr C
        rlc A          ; add dl,dl
        xrl A,R1       ; xor dl,al
        mov R0,A
        mov A,R1
        clr C
        rlc A          ; add al,al    ;nastaveni carry
        mov R1,#2      ; mov dh,2

        jc Nc1         ; jc  Nc1
        mov R1,#0      ; mov dh,0
Nc1:
        mov C,P        ; or  al,al
        mov A,R1
        jnc Nc2        ; jpe Nc2
        xrl A,#3       ; xor dh,3
Nc2:
        mov R1,A       ; mov al,dh
        clr C
        rrc A          ; rcr al,1
        push ACC
        mov A,R0
        mov R0,#6
Adding: clr C
        rlc A
        xch A,R1
        rlc A
        xch A,R1
        djnz R0,Adding
        mov R0,A
        pop ACC        ; 6 * add dx,dx

        orl A,R0       ; or  dl,al
        xrl A,Residue+1       ; xor dl,byte ptr es:[bx+1].Residue
        mov Residue,A
        mov Residue+1,R1; mov es:[bx].Residue,dx

        pop 01
        pop 00
        pop acc

        ret
```

9.2.2 Pro mikroprocesory řady 8086 (jazyk Turobo Pascal + Assembler 8086)

```
unit Crc16;
interface
uses
  Objects;
```

```
          {=======================================================}
          {                                                       }
          {  unit uCrc16 - jednotka pro vypocet Crc16              }
          {                                                       }
          {  (C)1992 Vladimir Kastner, Na Vlcovce 6, Praha 6      }
          {                                                       }
          {=======================================================}

{ (C) P.Tesar, T.Pecina, LP Praha, December 1989 }
{ Vypocet CRC-16 (IEEE MICRO 83)                 }
{ generacni polynom = x16 + x15 + x2 + 1         }
{ nejnizsi bit B je prvni prijaty nebo zasilany  }

type
  pCrc16 = ^ tCrc16;
  tCrc16 = object(tObject)
    Residue : Word;                 { x15 je umisten v bitu B0 }
    constructor Init;               { vytvoreni objektu }
    procedure SetResidue(Res: Word); { definuje hodnotu zbytku po deleni }
    function  GetResidue: Word;     { navrati hodnotu zbytku po deleni }
    procedure MakeCrc(B: Byte);     { vypocet zbytku a jeho navraceni }
  end;

implementation

constructor tCrc16.Init; { vytvoreni objektu }
begin
end;

procedure tCrc16.SetResidue(Res: Word); { definuje hodnotu zbytku po deleni }
begin
  Residue:=Res;
end;

function tCrc16.GetResidue: Word; { navrati hodnotu zbytku po deleni }
begin
  GetResidue:=Residue;
end;

procedure tCrc16.MakeCrc(B: Byte); { vypocet zbytku a jeho navraceni }
label Nc1,Nc2;
begin
  asm
    mov al,B
    les bx,Self
    xor al,byte ptr es:[bx].Residue
    mov dl,al
    add dl,dl
    xor dl,al
    add al,al
    mov dh,2
    jc  Nc1
    mov dh,0
  Nc1:
    or  al,al
    jpe Nc2
    xor dh,3
Nc2:
    mov al,dh
    rcr al,1
    add dx,dx
    add dx,dx
    add dx,dx
    add dx,dx
    add dx,dx
    add dx,dx
    or  dl,al
    xor dl,byte ptr es:[bx+1].Residue
    mov es:[bx].Residue,dx
  end;
end;
end.
```

9.2.3 Pro mikroprocesory řady 8080 a Z80 (Assembler 8080)

```
;POPIS CINNOSTI:
;PODPROGRAM CRCF JE EFEKTIVNI REALIZACI RUTINY
;PRO VYPOCET CRC-16 (CYCLIC REDUNDANCY CHECK).
;GENERACNI POLYNOM:
;       X^16 + X^15 + X^2 + 1
;PRIRAZENI BITU:
; ADR             ADR+1          ADR+2
; MSB        LSB   MSB       LSB   MSB          LSB
; X^16.......X^23  X^8.......X^15  X^0..........X^7
;
;ZAKLADNI VLASTNOSTI CRC JE:
;       NECHT CRC(AAAA,BBBB)=CCDD
;
;KDE AAAA ... POCATECNI ADRESA OBLASTI PAMETI
;    BBBB ... KONCOVA ADRESA   OBLASTI PAMETI
;    CCDD ... CRC-16 Z OBLASTI AAAA-BBBB VCETNE
;
;POTOM, ZAPISEME-LI NA ADRESU  BBBB+1 BYTE DD
;                A NA ADRESU  BBBB+2 BYTE CC
;
;MUSI VYJIT: CRC(AAAA,BBBB+2)=0000H
;
;CRC-16 POUZIVAME PRI KONTROLE OBSAHU PAMETI EPROM
;PRO VYPOCET CRC Z OBLASTI PAMETI JE TREBA PODPROGRAM
;CRCF UZAVRIT VNEJSI SMYCKOU PRES ADRESY OBLASTI.
;
;VSTUPNI REGISTRY: A  - BYTE
;                  BC - CRC Z PREDCHOZIHO KROKU
;VYSTUPNI REGISTRY:BC - CRC VYPOCTENE V PROVEDENEM KROKU
;ZMENENE REGISTRY: A, FLAGS, HL, BC
;
;PRIKLAD: PRO SAPI -1
        ORG     7000H
ADRPOC  EQU     1000H   ;NAPRIKLAD
ADRKON  EQU     1FFFH   ;NAPRIKLAD
HILO    EQU      136H   ;SAPI-1 MONITOR

        LXI     H,ADRPOC
        LXI     D,ADRKON
        LXI     B,0000H ;CRC=0000
CYKL:   MOV     A,M
        PUSH    H
        CALL    CRCF    ;VYPOCET CRC-16
        POP     H
        CALL    HILO    ;HL+1 : DE SET CY DLE POROVNANI
        JNC     CYKL
;       ...             BC=CRC-16 Z OBLASTI
;                       ADRPOC -- ADRKON
;

;PODPROGRAM PRO VYPOCET CRC-16
; ( IEEE MICRO 83 )
;------------------------------
CRCF:   XRA     C
        MOV     L,A
        MOV     H,A
        ADD     A
        XRA     L
        MOV     L,A
        MOV     A,H
        ADD     A
;=======================================
;       MVI     A,0     ;ORIGINAL 8080
;       JPE     CRCF1
;       MVI     A,03H
;CRCF1: JNC     CRCF2
;       XRI     02H
;CRCF2: MOV     H,A
;=======================================
; UPRAVA PRO 8080 & Z80
        MVI     H,10B
        JC      CRCF1
        MVI     H,00B
CRCF1:  ORA     A
        JPE     CRCF2
        MVI     A,11B
        SUB     H
```

```
        MOV     H,A
CRCF2:  MOV     A,H     ;CY=0
;---------------------------------------

        RAR
        DAD     H
        DAD     H
        DAD     H
        DAD     H
        DAD     H
        DAD     H
        ORA     L
        XRA     B
        MOV     C,A
        MOV     B,H
        RET

        END
```

9.3 Příklady použití síťové verze Term06

Příklady využívají komunikační knihovny firmy SofCon - ChnVirt, ChCom a ChnPrt.

9.3.1 Příklad komunikace v síti Master (PC), a 2xSlave (TERM06)

```
Program Primitivni_master_NetTerm06;
{ PC node 1
  TERM06 node 2 a 3
}

uses
    uString,
    ChnVirt,
    ChnCom,
    ChnPrt,
    Crt;

const
     BaudRate = 9600;

     delay1 = (1/BaudRate)*11*1000;

     delay2 = (1/BaudRate)*11*20*1000;

     ParamStr1 : tParamStr = 'NAM=PRT LSB=500 NOD=1 DNO=2 NAM=COM COM=3 ADD=$0110 '
                           + 'IRQ=5 BD=9600 BIT=8 PAR=N STO=1 LRB=1000';
     ParamStr2 : tParamStr = 'NAM=PRT LSB=500 NOD=1 DNO=3 NAM=COM COM=3 ADD=$0110 '
                           + 'IRQ=5 BD=9600 BIT=8 PAR=N STO=1 LRB=1000';

type
   tmess = array [0..65500] of Byte;

var
   Chn1,Chn2       : pChnVirt;
   SMess1,SMess2   : ^tmess;
   RMess1,RMess2   : ^tmess;
   LRMess1,LRMess2 : Word;
   LSMEss1,LSMEss2 : Word;
   res             : byte;
   OKSend          : Boolean;
   i,j             : LongInt;

begin

     New(SMess1);
     New(RMess1);
     New(SMess2);
     New(RMess2);

     { vytvoreni instance komunikacniho objektu Chn1 }

     Chn1 := ChnCollection^.ChNewInit(ChnPrt.cName);
     Chn2 := ChnCollection^.ChNewInit(ChnPrt.cName);

repeat

     with Chn1^ do
     begin

       { Nastavni parametru komunikace }
       ChSetParam(ParamStr1);
       if ChResult <> res_OK then halt;

       { Otevreni komunikacniho kanalu }
       ChOpen;
       repeat
         if ChResult <> res_Ok then halt { Chyba otevreni kanalu };
       until ChReady = CHS_Open;
       if ChResult <> res_Ok then halt;

       { Definice prijimaciho bufferu }
       ChReceiveBuffer(RMess1,SizeOf(tmess));
       if ChReceiveResult <> res_OK then halt; { Chyba otevreni bufferu }
```

```
  { Pripojeni kanalu }
  ChConnect;
  repeat
    if ChResult <> res_Ok then halt { Chyba pripojeni kanalu }
  until ChReady = CHS_Connect;
  if ChResult <> res_Ok then halt;
end;

  { Priprava zpravy k odeslani na NODE 2 }
  SMess1^    :=  { Command,Data  }
  LSMess1    :=  { Delka zpravy  }

with Chn1^ do
begin
  OKSend := True;
  repeat

      { Vyslani zpravy do terminalu NODE 2 }
      if ChSendReady = CHS_SendReady then
        begin
          ChSend(SMess1,LSMess1);
          { Cekani na odvysilani zpravy }
          repeat
            if ChSendResult <> res_Ok then halt; {Chyba pri odesilani zpravy}
          until ChSendReady = CHS_SendReady;
          if ChSendResult <> res_Ok then halt;
          OKSend := True;
        end
      else halt;
      { Cekani na odezvu }
      i := 0;
      while (ChReceiveReady <> CHS_ReceiveReady ) AND OKSend do
      begin
        inc (i);
        if ChReceiveResult <> res_OK then WriteLn('Chyba pri cekani na zpravu');
        if i > 50 then
          begin
            OKSend := False;
            Delay (round(delay2));
          end;
        delay(round(delay1));
      end;

  until OKSend;

  { Prijem zpravy }
  ChReceive(LRMess1);
  if ChReceiveResult <> Res_Ok then halt; {Chyba pri prijmu zpravy}

  with Chn1^ do
  begin
    ChDisconnect;
    ChClose
  end;

  { Zpracovani prijate zpravy }

  ...

end;

with Chn2^ do
begin

  { Nastavni parametru komunikace }
  ChSetParam(ParamStr2);
  if ChResult <> res_OK then halt;

  { Otevreni komunikacniho kanalu }
  ChOpen;
  repeat
    if ChResult <> res_Ok then halt; {Chyba otevreni kanalu}
  until ChReady = CHS_Open;
  if ChResult <> res_Ok then halt; {Chyba otevreni kanalu}

  { Definice prijimaciho bufferu }
  ChReceiveBuffer(RMess2,SizeOf(tmess));
  if ChReceiveResult <> res_OK then halt; {Chyba pripojeni bufferu}

  { Pripojeni kanalu }
```

```
        ChConnect;
        repeat
          if ChResult <> res_Ok then halt; {Chyba pripojeni kanalu}
        until ChReady = CHS_Connect;
        if ChResult <> res_Ok then halt; {Chyba pripojeni kanalu}
      end;

        { Priprava zpravy k odeslani na NODE 3 }
        SMess2^    :=  { Command,Data  }
        LSMess2    :=  { Delka zpravy  }

      with Chn2^ do
      begin
        OKSend := True;
        repeat
            { Vyslani zpravy do terminalu 1 }
            if ChSendReady = CHS_SendReady then
              begin
                ChSend(SMess2,LSMess2);
                { Cekani na odvysilani zpravy }
                repeat
                  if ChSendResult <> res_Ok then halt; {Chyba pri odesilani zpravy}
                until ChSendReady = CHS_SendReady;
                if ChSendResult <> res_Ok then halt;
                OKSend := True;
              end
            else halt;
            { Cekani na odezvu }
            i := 0;
            while (ChReceiveReady <> CHS_ReceiveReady ) AND OKSend do
            begin
              inc (i);
              if ChReceiveResult <> res_OK then WriteLn('Chyba pri cekani na zpravu');
              if i > 50 then
                begin
                  OKSend := False;
                  Delay (round(delay2));
                end;
              delay(round(delay1));
            end;

        until OKSend;

        { Prijem zpravy }
        ChReceive(LRMess2);
        if ChReceiveResult <> Res_Ok then halt; {Chyba pri prijmu zpravy}

        with Chn2^ do
        begin
          ChDisconnect;
          ChClose
        end;

        { Zpracovani prijate zpravy }

        ...

      end;



Until ... ;


with Chn1^ do
      begin
        { Zruseni instance Chn1 }
        Dispose(Chn1,Done);
      end;


with Chn2^ do
      begin
        { Zruseni instance Chn2 }
        Dispose(Chn2,Done);
      end;

       Dispose(SMess1);
       Dispose(RMess1);
       Dispose(SMess2);
       Dispose(RMess2);
```

```
end.
```

9.3.2 Ukázka režimu lokální editace terminálu

```
{$A+,B-,D+,E+,F+,G+,I+,L+,N-,O+,P-,Q-,R-,S+,T-,V+,X+,Y+}
{$M 16384,0,655360}
Program Test_Lokalni_Editace_NETTerm06;

uses
    Strings,
    uString,
    ChnVirt,
    ChnCom,
    ChnPrt,
    Crt;
const
     BaudRate = 9600;

     delay1 = (1/BaudRate)*11*1000;

     delay2 = (1/BaudRate)*11*20*1000;

     ParamStr1 : tParamStr = 'NAM=PRT LSB=500 NOD=1 DNO=2'+
                             ' NAM=COM COM=3 ADD=$0110 IRQ=5 BD=9600 BIT=8 PAR=N STO=1
LRB=1000';

type
   tmess = array [0..65500] of Byte;
   tdword = $00000000 .. $7FFFFFF;
   tdwrec = record
              B0,
              B1,
              B2,
              B3   : byte;
            end;

   tMessPtr = ^tmess;

var
   Chn1,Chn2       : pChnVirt;
   SMess1,RMess1   : tMessPtr;
   SMess0,RMess0   : tMessPtr;
   LRMess0,LSMess0 : Word;
   LRMess1,LSMess1 : Word;
   ML,MH,INum      : tdword;
   l               : byte;

procedure AddByteToMess(WrByte:Byte;WrMess:tMessPtr;var l:byte);
begin
  WrMess^[l] := WrByte;
  inc (l);
end;

procedure AddDWordToMess(WrWord:tdword;WrMess:tMessPtr;var l:byte);
var DW:tdwrec;
begin
  DW:=tdwrec(WrWord);
  WrMess^[l] := DW.B0;
  inc(l);
  WrMess^[l] := DW.B1;
  inc(l);
  WrMess^[l] := DW.B2;
  inc(l);
  WrMess^[l] := DW.B3;
  inc(l);
end;

begin

     New(SMess0);
     New(RMess0);
     New(SMess1);
     New(RMess1);

     l:=0;

     AddByteToMess($C1,SMess0,l);    { identifikator C1 }
```

```
AddByteToMess($1E,SMess0,l);    { smaz display }
AddByteToMess(88,SMess0,l);     { vypis X= }
AddByteToMess(61,SMess0,l);

LSMess0 := l;

WriteLn;
Write('Zadej dolni hranici pro editovane cislo: '); ReadLn(ML);
Write('Zadej horni hranici pro editovane cislo: '); ReadLn(MH);
Write('Zadej inicializacni hodnotu: ');             ReadLn(INum);

l:=0;

AddByteToMess($C7,SMess1,l);    { identifikator C7 - lokalni editace   }
AddByteToMess($02,SMess1,l);    { umisteni cisla na displeji }
AddByteToMess($03,SMess1,l);    { editace cisla }

AddDWordToMess(INum,SMess1,l);
AddDWordToMess(ML,SMess1,l);
AddDWordToMess(MH,SMess1,l);

LSMess1 := l;

{ vytvoreni instance komunikacniho objektu Chn1 }
Chn1:=ChnCollection^.ChNewInit(ChnPrt.cName);


with Chn1^ do
begin
  { Nastavni parametru komunikace }
  ChSetParam(ParamStr1);
  if ChResult <> res_OK then halt;

  { Otevreni komunikacniho kanalu }
  ChOpen;
  repeat
    if ChResult <> res_Ok then halt { Chyba otevreni kanalu };
  until ChReady = CHS_Open;
  if ChResult <> res_Ok then halt;

  { Definice prijimaciho bufferu }
  ChReceiveBuffer(RMess1,SizeOf(tmess));
  if ChReceiveResult <> res_OK then halt; { Chyba otevreni bufferu }

  { Pripojeni kanalu }
  ChConnect;
  repeat
    if ChResult <> res_Ok then halt { Chyba pripojeni kanalu }
  until ChReady = CHS_Connect;
  if ChResult <> res_Ok then halt;

  { Testovani stavu terminalu v pripade ze je v rezimu lokalni editace,
    tak ji ukoncit zpravou s identifikatorem CBh }
  { ... }

  { vyslani zpravy 0 do terminalu }
  if ChSendReady = CHS_SendReady then
    begin
      ChSend(SMess0,LSMess0);

      { cekani na odvysilani zpravy }
      repeat
        if ChSendResult <> res_Ok then halt; {Chyba pri odesilani zpravy}
      until ChSendReady = CHS_SendReady;
      if ChSendResult <> res_Ok then halt {Zprava neodeslana}
    end
      else halt; {Zprava neodeslana z duvodu nepripravenosti zarizeni}

      { Cekani na odezvu }
    while (ChReceiveReady <> CHS_ReceiveReady) do
    if ChReceiveResult <> res_OK then halt; {Chyba pri cekani na zpravu}

  { Prijem zpravy }
  ChReceive(LRMess0);
  if ChReceiveResult <> Res_Ok then halt; {Chyba pri prijmu zpravy}

  delay (round(Delay2));

  { vyslani zpravy 1 do terminalu }
  if ChSendReady = CHS_SendReady then
    begin
```

```
            ChSend(SMess1,LSMess1);

            { cekani na odvysilani zpravy }
            repeat
              if ChSendResult <> res_Ok then halt; {Chyba pri odesilani zpravy}
            until ChSendReady = CHS_SendReady;
            if ChSendResult <> res_Ok then halt; {Zprava neodeslana}
          end
            else halt; {Zprava neodeslana z duvodu nepripravenosti zarizeni}
            { Cekani na odezvu }
          while (ChReceiveReady <> CHS_ReceiveReady) do
          if ChReceiveResult <> res_OK then halt; {Chyba pri cekani na zpravu}

        { Prijem zpravy }
        ChReceive(LRMess1);
        if ChReceiveResult <> Res_Ok then halt; {Chyba pri prijmu zpravy}


        ChDisconnect;
        ChClose

      end;

        { Zpracovani prijate zpravy, testovani ukonceni editace ... }


with Chn1^ do
      begin
        { zruseni instance Chn1 }
        Dispose(Chn1,Done);
      end;

       Dispose(SMess1);
       Dispose(RMess1);

end.
```