uTerm10S

JEDNOTKA PRO IMPLEMETACI TERMINÁLU TERM10 VE FUNKCI PODŘÍZENÉ PERIFERIE

Příručka uživatele a programátora

***SofCon*®spol. s r.o.**
Střešovická 49
162 00 Praha 6
tel/fax: +420 220 180 454
E-mail: sofcon@sofcon.cz
www: http://www.sofcon.cz

Verze dokumentu 1.10

Informace v tomto dokumentu byly pečlivě zkontrolovány a SofCon věří, že jsou spolehlivé, přesto SofCon nenese odpovědnost za případné nepřesnosti nebo nesprávnosti zde uvedených informací.
SofCon negarantuje bezchybnost tohoto dokumentu ani programového vybavení, které je v tomto dokumentu popsané. Uživatel přebírá informace z tohoto dokumentu a odpovídající programové vybavení ve stavu, jak byly vytvořeny a sám je povinen provést validaci bezchybnosti produktu, který s použitím zde popsaného programového vybavení vytvořil.
SofCon si vyhrazuje právo změny obsahu tohoto dokumentu bez předchozího oznámení a nenese žádnou odpovědnost za důsledky, které z toho mohou vyplynout pro uživatele.

Datum vydání: 16.05.2003
Datum posledního uložení dokumentu: 16.05.2003
(Datum vydání a posledního uložení dokumentu musí být stejné)

Upozornění:
V dokumentu použité názvy výrobků, firem apod. mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Obsah :

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Če		První vydání
1.10	2.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky pro implementaci terminálu Term10 ve funkci podřízené periferie.

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro čtení tohoto dokumentu je potřeba seznámit se s manuálem uATerm, uCharBuf, uTermT10, uKeybT10 a uDispT10.

Popis formátu verze knihovny a souvisejících funkcí je popsán v manuálu LibVer.

2. Termíny a definice

Používané termíny a definice jsou popsány v samostatném dokumentu Termíny a definice.

3. Úvod

Jednotka **uTerm10S** implementuje funkce terminálu TERM10 jako podřízené periferie. Kódy stisknutých tlačítek vysílá po komunikačním kanálu. Zobrazení na displeji je řízeno daty přijatými z komunikačního kanálu. Přijímaná data představují ESC sekvence popisující grafický obsah obrazovky.

Zděděné metody zde nejsou popsány, jejich popis je možno najít v dokumentaci k jednotkám **tATerm, tTermChr, uTermGr** a **uTermT10**.

4. Popis konstant a typů

```
cVerNo = např. $0251; { BCD formát }
cVer   = např. '02.51,07.08.2003';
```

Číslo verze jednotky v BCD tvaru a v textové podobě včetně datumu změny.

```
tEscSwitch =(escsw_Err
             escsw_None,
             escsw_Text,
             escsw_Cursor,
             escsw_Style,
             escsw_Graphic,
             escsw_BkGround,
             escsw_Led,
             escsw_Out
             escsw_Sound);
```

Typ **tEscSwitch** je výčtový typ stavů automatu pro dekódování přijaté ESC sekvence.

```
tT10ScreenRec=record
                TextStr:string;
                CursorStr:string[20];
                StyleStr:string;
                GraphicStr:string;
                BkGroundStr:string[20];
              end;
```

Typ **tT10ScreenRec** představuje jednu sadu přijímaných obrazových dat.

```
tT10ScreenData=array [0..1] of tT10ScreenRec;
```

Typ tT10ScreenData je používán pro buffer na sady obrazových dat. Do jedné sady jsou data přijímána z komunikačního kanálu a z druhé sady jsou data zobrazována.

5. Popis objektu tTermT10S

Objektový typ **tTermT10S** je potomkem typu **tTermT10**. Implementuje funkce terminálu TERM10 jako podřízené periferie připojené k aplikaci komunikačním kanálem.

5.1. Proměnné

```
FlgEsc:Byte;
```

Proměnná **FlEsc** obsahuje stav automatu pro dekódování přijímaných ESC sekvencí.

```
EscSwitch:tEscSwitch;
```

Proměnná **ESCSwitch** obsahuje stav automatu pro dekódování přijímaných ESC sekvencí.

```
ScrData:tT10ScreenData;
```

Proměnná **ScrData** implementuje vyrovnávací buffer pro dvě sady obrazových dat. V jedné sadě jsou data určená k zobrazení a do druhé jsou data přijímána z komunikačního kanálu. Po přijetí kompletní sady se funkce obou sad se vymění.

```
ScrRecDataIx:Byte;
```

Proměnná **ScrRecDataIx** obsahuje index sady do které jsou obrazová data přijímána.

```
ScrRdyDataIx:Byte;
```

Proměnná **ScrRdyDataIx** obsahuje index sady ze které jsou obrazová data zobrazována.

```
ScrDataRdy:Boolean;
```

Proměnná **ScrDataRdy** obsahuje příznak, že byla přijata kompletní obrazová data, která ještě nejsou zobrazena.

```
LedStr:string[40];
```

Proměnná **LedStr** implementuje buffer pro příjem ESC sekvence s definicí stavu signalizačních LED.

```
OutStr:string[40];
```

Proměnná **OutStr** implementuje buffer pro příjem ESC sekvence s definicí satvu optovýstupů.

```
SoundStr:string[40];
```

Proměnná **SoundStr** implementuje buffer pro příjem ESC sekvence s definicí stavu zvukové signalizace.

```
SetupKeyChar:Char;
```

Proměnná **SetupKeyChar** obsahuje kód klávesy pro vstup do setupu.(implicitně SHIFT-ENTER)

```
KbdSendBuff:array[0..20]of Char;
```

Proměnná **KbdSendBuff** představuje buffer pro vysílání kódů stisknutých kláves.

5.2. Metody

5.2.1. Init

```
constructor Init(ADisp:pADisp;AKeyb:pAKeyb;NAddr:Word;
                 AChnTerm:pChnVirt;AChnRecBuff:pointer);
```

Konstruktor **Init** inicializuje objekt **tTerm10** a nastavuje proměnné objektu na počáteční hodnoty. Nastavuje stav automatu pro dekódování přijímané ESC sekvence do výchozího stavu (**FlESC** a **EscSwitch**), nuluje buffer obrazových dat **ScrData**, index bufferu dat pro zobrazení **ScrRdyDataIx** a index bufferu dat pro příjem **ScrRecDataIx**, příznak přijetí dat **ScrDataRdy** nastavuje na false a proměnnou **SetupKeyChar** na implicitní hodnotu **zShiftEnter**.

5.2.2. Tick

```
procedure Tick;virtual;
```

Metoda **Tick** je periodicky volána z kontextu procesu "Term" a zajišťuje tak činnost terminálu. Provádí tyto činnosti:

Přijímá data z komunikačního kanálu a předává je metodě ReceiveDataFromRemoteTerm.

Testuje příznaky stisku tlačítek "START" a "STOP" v objektu klávesnice. (**Keyb^.FlStart** a **Keyb^.FlStop**). Bylo-li některé z tlačítek stisknuto je do bufferu klávesnice vložen příslušný znak **zStart** nebo **zStop**. Ty jsou pak přeneseny spolu s kódy ostatních tlačítek do vzdáleného terminálu implementovaného objektem **tTermT10R**. Zde jsou znaky **zStart** a **zStop** z posloupnosti kódů kláves opět vyňaty a nastaveny na jejich základě příslušné příznaky.

Není-li buffer klávesnice prázdný je vyslán jeho obsah metodou **SendDataToRemoteTerm** do vzdáleného terminálu.

Obnoví stav signalizačních LED voláním **LedSign(0,0)** a je-li povolena obnova stavu optovýstupů (**FlRereshOut**), obnoví stav optovýstupů voláním **WriteOut(0,0).**

Podle stavu funkce sreen saver nastaví voláním metody **FlLight** jas výbojky podsvětlující displej.

5.2.3. DTick_GetDisplayData

```
function  DTick_GetDisplayData:Boolean;virtual;
```

Metoda **DTick_GetDisplayData** slouží k předání obrazových dat z objektu terminálu do objektu displeje. Objektu displeje jsou předána data z proměnné **ScrData** nacházející se pod indexem **ScrRdyDataIx** (data určená k zobrazení). Metoda testuje zda jsou k dispozici přijatá dosud nezobrazená data (příznak **ScrDataRdy**). Pokud ano jsou data předána objektu displeje, příznak nastaven na false a metoda vrací true. V opačném případě vrací metoda false.

5.2.4. SendDataToRemoteTerm

```
procedure SendDataToRemoteTerm(TrBufPtr:pBuf;
                               TrBufSize:word);virtual;
```

Metoda **SendDataToRemoteTerm** slouží k odvysílání kódů stisknutých kláves přečtených z bufferu klávesnice Vyčká na vyprázdnění vysílače a vyšle data na komunikační kanál. Parametrem **TrBufPtr** je předáván ukazatel na buffer s daty a parametrem **TrBufSize** délka dat v bufferu. Není-li prázdný buffer klávesnice, je metoda **SendDataToRemoteTerm** volána metodou **Tick** k vyslání kódů stisknutých kláves na komunikační kanál.

5.2.5. ReceiveDataFromRemoteTerm

```
procedure ReceiveDataFromRemoteTerm(RecBufPtr:pBuf;
          RecBufSize:word);virtual;
```

Metodě **ReceiveDataFromRemoteTerm** jsou předávána data přijatá z komunikačního kanálu. Parametrem **RecBufPtr** je předán ukazatel na buffer s

přijatými daty a parametrem **RecBufSize** délka dat v bufferu. Metoda provádí dekódování ESC sekvencí přijatých dat. Je-li sestavena kompletní sada dat, jsou na konci metody voláním **Disp^.UpDateDisplay** data zobrazena.

5.2.6. ResetScreenData

```
procedure ResetScreenData;
```

Metoda **ResetScreenData** nastaví automat pro dekódování přijaté ESC sekvence do výchozího stavu. Je volána v případě chyby příjmu dat z komunikačního kanálu.

5.2.7. NewScreenData

```
procedure NewScreenData;
```

Metoda **NewScreenData** smaže buffer přijímaných obrazových dat. Je volána metodou **ReceiveDataFromRemoteTerm** při přijetí začátku obrazových dat.

5.2.8. ShowScreen

```
procedure ShowScreen;
```

Změnou indexů **ScrRecDataIx** a **ScrRdyDataIx** metoda **ShowScreen** vymění funkce obou sad bufferu obrazových dat a nastaví příznak přijetí kompletní sady obrazových dat **ScrDataRdy** na true. Je volána metodou **ReceiveDataFromRemoteTerm** po ukončení příjmu kompletní sady obrazových dat.

5.2.9. NewLedData

```
procedure NewLedData;
```

Metoda **NewLedData** smaže buffer pro příjem dat s definicí stavu signalizačních LED **LedStr**. Je volána metodou **ReceiveDataFromRemoteTerm** při příjmu začátku dat s definicí stavu signalizačních LED.

5.2.10. NewOutData

```
procedure NewOutData;
```

Metoda **NewOutData** smaže buffer pro příjem dat s definicí stavu optovýstupů OutStr. Je volána metodou **ReceiveDataFromRemoteTerm** při příjmu začátku dat s definicí stavu optovýstupů.

5.2.11. NewSoundData

```
procedure NewSoundData;
```

Metoda **NewSoundData** smaže buffer pro příjem dat s definicí stavu zvukové signalizace **SoundStr**. Je volána metodou **ReceiveDataFromRemoteTerm** při příjmu začátku dat s definicí stavu zvukové signalizace.

5.2.12. ShowLed

```
procedure ShowLed;
```

Metoda **ShowLed** dekóduje ESC sekvenci s daty definujícími stav signalizačních LED a voláním zděděné metody **LedSign** zapíše data na výstup LED. Je volána metodou **ReceiveDataFromRemoteTerm** po ukončení příjmu dat s definicí stavu optovýstupů.

5.2.13. DoOut

```
procedure DoOut;
```

Metoda **DoOut** dekóduje ESC sekvenci s daty definujícími stav optovýstupů a voláním zděděné metody **WriteOut** zapíše data na optovýstupy. Je volána metodou **ReceiveDataFromRemoteTerm** po ukončení příjmu dat s definicí stavu optovýstupů.

5.2.14. DoSound

```
procedure DoSound;
```

Metoda **DoSound** dekóduje ESC sekvenci s daty definujícími stav zvukové signalizace a podle stavu volá buď zděděnou metodu **BellOn** nebo **BellOff**. Je volána metodou **ReceiveDataFromRemoteTerm** po ukončení příjmu dat s definicí stavu zvukové signalizace.