uTermChr

JEDNOTKA PRO PRÁCI SE ZNAKOVÝM TERMINÁLEM

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)

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. Doplněná proměnná fCharBufEmpty. Doplněný popis funkce WriteLnSXY.

1.2. Účel dokumentu

Tento dokument slouží jako popis knihovny jednotky pro práci se znakovým terminálem uTermChr.

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 ChnVirt a uATerm.

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 **uTermChr** obsahuje jediný objekt **tTermChr**, který je potomkem objektu **uATerm**, objektu implementující abstraktní terminál. **tTermChr** definuje jeho abstraktní metody a rozšiřuje o metody pracující se znakovým terminálem, jako je mazání a výpis na displej, ukládání a načítání obsahu displeje, pohyb kurzoru po displeji.

Zděděné metody jsou popsány v dokumentaci k jednotce **uATerm**.

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.

Jednotka deklaruje několik konstant určujících stav automatu pro editaci a stav automatu pro výpis nápovědy :

```
S_ReadLnReady   = $0000; { je k dispozici string }
S_ReadLnBegin   = $0001; { počátek Read          }
S_ReadLnNoReady = $0002; { není ready            }
S_ReadLnHelp    = $0003; { je Help               }
S_HelpReady     = $0000; { je ukončen help       }
S_HelpBegin     = $0001; { počátek help          }
S_HelpNoReady   = $0002; { help není dokončen    }
```

5. Popis objektu tTermChr

Objekt **tTermChr** je potomek objektu **tATerm** a definuje především jeho abstraktní metody jako **Tick** a **DTick_GetDisplayData**, které zajišťují předávání dat z klávesnice do bufferu klávesnice a z bufferu displeje na displej, a metody, které pracují s fyzickým znakovým displejem, dále metodu Keypressed a Readkey pracující s bufferem klávesnice a metodu **Done**.

```
pATermChr = ^tTermChr;
tTermChr = object(tATerm)
```

5.1. Pole

Všechny popisované proměnné slouží výhradně pro vnitřní použití a uživatel by je neměl využívat ani měnit.

Proměnná **EditEnd** je množina znaků pro ukončení editace. Implicitní nastavení je [zCr, zEsc].

```
EditEnd           :tSetChar;
```

Proměnná **flTickExecute** je nastavena na true po dokončení **Keyb^.Tick** a informuje tak proces "Menu" o sejmutí stavu klávesnice.

```
flTickExecute     :Boolean;
```

Proměnná **flIns** obsahuje informaci o režimu vkládání-přepis. Implicitní nastavení je true = vkládání.

```
flIns             :Boolean;
```

Proměnná **TermCharSize** vlastní údaje o velikosti znakového okna terminálu.

```
TermCharSize      :tCharRect;
```

Proměnná **DispCursorWin** vlastní informace o pozici kurzoru ve znakovém okénku terminálu.

```
DispCursorWin     :tCharWin;
```

Proměnná **DispShowCursor** obsahuje informaci o viditelnosti kursoru. Implicitní nastavení je false = kursor neviditelný.

```
DispShowCursor    :Boolean;
```

Proměnná **WinAct** určuje velikost a pozice kurzoru aktuálního znakového okna.

```
WinAct            :tCharWin;
```

Proměnná **flEsc** obsahuje stav automatu pro dekódování ESC sekvencí.

```
flEsc             :byte;
```

Proměnná **flEdit** určuje, jestli se text bude při editaci rolovat.

```
flEdit            :Boolean;
```

Proměnná **flUpDn** určuje význam kurzorových kláves při editaci. Při **flUpDn**=true se při editaci vyměňují jednotlivé znaky na pozici kursoru při stlačení Up, nebo Dn. Tento mód se používá u terminálů s omezeným rozsahem klávesnice např. Term06 a Term05.

```
flUpDn            :Boolean;
```

Proměnná **flTerm06** určuje specifické chování pro Term06.

```
flTerm06          :Boolean;
```

Proměnná **edType** určuje typ právě editovaného řetězce. Má význam pro editaci v režimu **flUpDn**=true.

```
edType            :tEdit;
```

Proměnná **TermCharBuf** je ukazatel na znakový buffer terminálu.

```
TermCharBuf       :pBuf;
```

Proměnná **TermCharBufSize** velikost paměti alokované **TermCharBuf^**.

```
TermCharBufSize   :word;
```

Proměnná **dTab** určuje velikost tabelátoru.

```
dTab              :byte;
```

Proměnná **fCharBuffEmpty** je nastavená na true pokud je prázdný znakový buffer terminálu, tj. obsahuje samé mezery.

```
fCharBuffEmpty   :boolean;
```

Proměnná **TimeOutKeyb** určuje timeout pro klávesnici v milisekundách. Po vypršení timeoutu se do bufferu klávesnice vloží string **TimeOutStr**. Pro hodnotu **TimeOutKeyb** = -1 se nebude vypršení timeoutu kontrolovat. Proměnná není veřejná.

```
TimeOutKeyb      :longint;
```

5.2. Metody

5.2.1. Init constructor

```
constructor Init(NewDisp:pADisp;NewKeyb:pAKeyb;NewChnTerm:pChnVirt;
                 NewChnRecBuf:pointer);
```

Konstruktor inicializuje abstraktní terminál **tATerm** voláním **tATerm.Init** a inicializuje pole objektu, např. základní okna displeje a ukončovací znaky editace a helpu. Parametrem **NewDisp** je předáván ukazatel na instanci objektu displeje, parametrem **NewKeyb** ukazatel na instanci objektu klávesnice, parametrem **NewChnTerm** ukazatel na instanci objektu komunikačního kanálu terminálu a parametrem **NewChnRecBuf** ukazatel na buffer alokovaný pro příjem znaků z komunikačního kanálu.

5.2.2. Done destructor

```
destructor Done; virtual;
```

Destruktor uvolňuje buffer terminálu pro editaci, metoda volá **tATerm.Done**.

5.2.3. KeyPressed

```
function KeyPressed: Boolean; virtual;
```

Metoda vrací true, je-li ve vyrovnávacím bufferu klávesnice k dispozici znak. Mimo to kontroluje vypršení timeoutu **TimeOutKeyb** a v případě kladné odezvy se smaže buffer klávesnice a naplní se stringem **TimeOutStr**.

5.2.4. ReadKey

```
function ReadKey: Char; virtual;
```

Metoda vybere z vyrovnávacího bufferu klávesnice znak a dodá jej jako svou funkční hodnotu. Je-li buffer prázdný, navrátí znak #$0. Ve svém těle volá metodu **Keyb^.Readkey**, poté přiřadí **flTickExecute** true.

5.2.5. Tick

```
procedure Tick; virtual;
```

Metoda **Tick** je volána periodicky z kontextu procesu "TICK". Volá **Keyb^.Tick**, čímž se zabezpečí pravidelný přenos znaků z klávesnice do vyrovnávacího bufferu a **Disp^.Tick**, zajišťující obnovování displeje.

Metoda se modifikuje podle proměnné **flUpDn**. Při **flUpDn**=true se při editaci vyměňují jednotlivé znaky na pozici kursoru při stlačení Up, nebo Dn podle typu editované proměnné, který je uložený v proměnné **edType**. Tento mód se používá u terminálů s omezeným rozsahem klávesnice např. Term06 a Term05.

5.2.6. DTick_GetDisplayData

```
function tTermChr.DTick_GetDisplayData:Boolean;
```

Touto metodou požaduje objekt displeje všechna data od objektu terminálu. V této abstraktní verzi je to pouze znakový buffer a pozice kurzoru. Je volána z kontextu procesu "Tick".

5.2.7. ShowDispCursor

```
procedure ShowDispCursor; virtual;
```

Metoda nastaví kurzor na viditelný.

5.2.8. HideDispCursor

```
procedure HideDispCursor; virtual;
```

Metoda nastaví kurzor na neviditelný.

5.2.9. TestDispCursor

```
function TestDispCursor : boolean; virtual;
```

Metoda vrátí nastavení kurzoru.

5.2.10. GetWindow

```
procedure GetWindow(var X,Y,W,H:Byte); virtual;
```

Metoda **GetWindow** vrací v parametrech **X, Y, W, H** souřadnice a velikost aktuálního okna.

5.2.11. SetWindow

```
procedure SetWindow(X,Y,W,H:Byte); virtual;
```

Metoda **SetWindow** nastaví podle parametrů **X, Y, W, H** souřadnice a velikost aktuálního okna.

5.2.12. GetWindowE

```
procedure GetWindowE(var X,Y,W,H:Byte); virtual;
```

Metoda **GetWindowE** vrací v parametrech **X, Y, W, H** souřadnice a velikost editačního okna.

5.2.13. SetWindowE

```
procedure SetWindowE(X,Y,W,H:Byte); virtual;
```

Metoda **SetWindowE** nastaví podle parametrů **X, Y, W, H** souřadnice a velikost editačního okna.

5.2.14. GetWindowH

```
procedure GetWindowH(var X,Y,W,H: Byte); virtual;
```

Metoda **GetWindowH** vrací v parametrech **X, Y, W, H** souřadnice a velikost okna pro výpis nápovědy.

5.2.15. SetWindowH

```
procedure SetWindowH(X,Y,W,H: Byte); virtual;
```

Metoda **SetWindowH** nastaví podle parametrů **X, Y, W, H** souřadnice a velikost okna pro výpis nápovědy.

5.2.16. ClrScr

```
procedure TestDispCursor; virtual;
```

Metoda smaže aktuální okno displeje.

5.2.17. GotoXY

```
procedure GotoXY(X, Y : byte); virtual;
```

Metoda nastaví kurzor na souřadnice X, Y v aktuálním okně displeje.

5.2.18. RolUp

```
procedure RolUp; virtual;
```

Metoda odroluje aktuální okno obrazovky o řádek nahoru.

5.2.19. MoveRi

```
procedure MoveRi(X, Y: Byte);
```

Metoda vloží do místa **X,Y** v aktivním okně mezeru a posune veškerý další text v okně o jeden znak vpravo. Znak z poslední pozice aktuálního okna je ztracen.

5.2.20. MoveLe

```
procedure MoveLe(X, Y: Byte);
```

Metoda zruší v aktivním okně znak v pozici **X,Y** a posune veškerý další text v okně o jeden znak vlevo. Poslední pozice v okně je doplněna mezerou.

5.2.21. IncXY

```
function IncXY(var X, Y: Byte):Boolean; virtual;
```

Metoda **IncXY** vypočte souřadnice následující pozice od pozice X,Y v aktuálním okně. Metoda respektuje znakový rastr a velikost okna. Metoda vrací true je-li vypočtená pozice mimo hranice aktuálního okna.

5.2.22. DecXY

```
function DecXY(var X, Y: Byte):Boolean; virtual;
```

Metoda **DecXY** vypočte souřadnice předchozí pozice od pozice X,Y v aktuálním okně. Metoda respektuje znakový rastr a velikost okna. Metoda vrací true je-li vypočtená pozice mimo hranice aktuálního okna.

5.2.23. WhereX

```
function WhereX:Byte; virtual;
```

Metoda **WhereX** vrací jako funkční hodnotu horizontální pozici kursoru v aktuálním okně.

5.2.24. WhereY

```
function WhereY:Byte; virtual;
```

Metoda **WhereY** vrací jako funkční hodnotu vetikální pozici kursoru v aktuálním okně.

5.2.25. UkBuff

```
function UkBuff(X,Y: Word): Word; virtual;
```

Metoda vypočte aktuální lineární pozici ve znakovém bufferu **TermCharBuf^** jako funkci pozice **X,Y** v aktuálním okně.

5.2.26. ReadChar

```
function ReadChar:Char; virtual;
```

Metoda **ReadChar** vrací jako funkční hodnotu znak ležící pod kursorem v aktuálním okně.

5.2.27. ReadString

```
function ReadString:String; virtual;
```

Metoda **ReadString** vrací jako funkční hodnotu řetězec znaků ležících za kursorem do konce řádku v aktuálním okně. Koncové mezery jsou z výstupního řetězce vypuštěny.

5.2.28. ClrRow

```
procedure ClrRow; virtual;
```

Metoda **ClrRow** smaže v aktuálním okně řádek na kterém leží kursor.

5.2.29. ClrEnd

```
procedure ClrEnd; virtual;
```

Metoda **ClrEnd** smaže v aktuálním okně řádek na kterém leží kursor a všechny následující řádky až do konce aktuálního okna.

5.2.30. ClrRight

```
procedure ClrRight; virtual;
```

Metoda **ClrRight** smaže v aktuálním okně znak pod kursorem a všechny následující znaky až do konce řádku.

5.2.31. ClrLeft

```
procedure ClrLeft; virtual;
```

Metoda **ClrLeft** smaže v aktuálním okně znaky od začátku řákdku, na kterém leží kursor, až po znak pod kursorem.

5.2.32. SaveTerminalScr

```
procedure SaveTerminalScr; virtual;
```

Metoda uloží obsah aktuálního okna obrazovky terminálu. Používá se před výpisem nápovědy.

5.2.33. LoadTerminalScr

```
procedure LoadTerminalScr; virtual;
```

Metoda obnoví obsah aktuálního okna obrazovky terminálu uložený metodou **SaveTerminalScr**.

5.2.34. GetTerminalSize

```
procedure GetTerminalSize(var W,H: Byte); virtual;
```

Metoda **GetTerminalSize** vrátí v parametrech **W, H** šířku a výšku znakového rastru terminálu.

5.2.35. WriteS

```
procedure WriteS(S : string); virtual;
```

Metoda vypíše text předaný v parametru **S** od aktuální pozice kurzoru v aktuálním okně displeje. Text předávaný metodě může obsahovat Esc sekvence a řídící znaky. Řídící znaky a Esc sekvence jsou interpretovány následujícím způsobem:

znak , Esc sekvence	**význam**
zDel	Vymaže znak pod kursorem a všechny znaky za kursorem posune o jednu pozici do prava. Na poslední místo je doplněn znak mezera.
zBs	Vymaže znak o jednu pozici před kursorem a všechny znaky za kursorem posune o jednu pozici doprava. Na poslední místo je doplněn znak mezera.
zHt	Doplní příslušný počet mezer k nejbližší pozici tabulátoru. V případě potřeby text odroluje.

znak , Esc sekvence	**význam**
zLf	Přesune výpis na následující řádek. V případě potřeby text odroluje.
zCr	Přesune výpis na začátek řádku.
zDn	Přesune výpis na následující řádek.
zRi	Přesune o znak doprava.
zLe	Přesune o znak doleva.
zUp	Přesune výpis na předchozí řádek.
zHome	Přesune výpis na začátek řádku.
zEnd	Přesune výpis na konec řádku.
zPgUp	Je-li aktuální pozice výpisu za aktuálním oknem přesune výpis na první řádek.
zPgDn	Je-li aktuální pozice výpisu za aktuálním oknem přesune výpis na poslední znak aktuálního okna.
zClrScr	Vymaže okno a přesune výpis na začátek.
zIns	Změní režim vkládání-přepis.
zEsc Y <XPos> <Ypos>	Přesune výpis na pozici <XPos>,<YPos>, kde <Xpos> a <Ypos> jsou znaky, jejichž ordinální hodnota udává souřadnice výpisu.

5.2.36. WriteSXY

```
procedure WriteSXY(X, Y : byte; S : string); virtual;
```

Metoda **WriteSXY** přesune kursor na pozici **X, Y** a vypíše text předaný parametrem **S** v aktuálním okně displeje stejně jako metoda **WriteS**.

5.2.37. WriteLnS

```
procedure WriteLnS(S: String); virtual;
```

Metoda **WriteLnS** vypíše text předaný v parametru **S** od aktuální pozice kurzoru v aktuálním okně displeje obdobně jako metoda **WriteS** a přesune kursor na začátek následujícího řádku.

5.2.38. WriteLnSXY

```
procedure WriteLnSXY(X, Y : byte; S: String); virtual;
```

Metoda **WriteLnSXY** přesune kurzor na pozici **X, Y** a vypíše text předaný v parametru **S** v aktuálním okně displeje obdobně jako metoda **WriteS** a přesune kursor na začátek následujícího řádku.

5.2.39. ReadLnH

```
procedure ReadLnH(var S : string; H : string); virtual;
```

Metoda **ReadLnH** nastaví na počátek automat pro editaci a voláním metody **RTick** provede první krok automatu. Další kroky automatu se provádí voláním metody **RTick** prostřednictvím metod **ReadLnResult** a **ReadLnReady** z kontextu procesu "Menu". Parametr **S** představuje editovaný string a parametr **H** obsahuje string pro výpis nápovědy.

5.2.40. ReadLn

```
procedure ReadLn(var S: String); virtual;
```

Metoda **ReadLn** pracuje stejně jako metoda **ReadLnH** s tím rozdílem že řetězec pro výpis nápovědy je prázdný.

5.2.41. ReadLnResult

```
function ReadLnResult: Char; virtual;
```

Metoda **ReadLnResult** provede voláním metody **RTick** krok automatu pro editaci a v případě, že je editace ukončena navrátí hodnotu ukončovacího znaku. V opačném případě navrací hodnotu #$0.

5.2.42. ReadLnReady

```
function ReadlnReady:Word; virtual;
```

Metoda **ReadLnReady** provede voláním metody **RTick** krok automatu pro editaci a jako funkční hodnotu vrací stav automatu.

5.2.43. RTick

```
procedure RTick; virtual;
```

Metoda **RTick** provede krok automatu pro editaci. V prvním kroku automatu je uloženo aktuální okno a jako aktuální okno nastaveno editační okno, kam je vypsán editační řetězec nastavený metodou **ReadLn** nebo **ReadLnH**.

V dalších krocích jsou odebírány znaky z klávesnice a vypisovány na aktuální pozici kursoru. Znaky jsou do okna vpisovány metodou **WriteS**. Význam řídících znaků přijatých z klávesnice (zLe, zRi, zUp, zDn, zCr ...) je popsán v popisu metody **WriteS**. Je-li z klávesnice odebrán znak pro vyvolání nápovědy je vypsána nápověda voláním metod pro obsluhu automatu výpisu nápovědy (Help, HelpReady, HelpResult, Htick). Je-li z klávesnice odebrán některý znak z množiny **EditEnd** (znak pro ukončení editace) je automat ukončen a aktualizován editační řetězec.

Metoda **RTick** je chráněna proti vícenásobnému vstupu z kontextu více procesů.

5.2.44. Help

```
procedure Help(H: String); virtual;
```

Metoda **Help** nastaví na počátek automat pro výpis nápovědy a voláním metody **HTick** provede první kok automatu (výpis nápovědy). Parametr **H** obsahuje text nápovědy, který je vypsán do přednastaveného okna Help. Další kroky automatu (čekání na ukončovací znak nápovědy) jsou prováděny metodou **HTick** volanou prostřednictvím metod **HelpResult** a **HelpReady** z kontextu procesu "Menu".

5.2.45. HelpResult

```
function HelpResult: Char; virtual;
```

Metoda **HelpResult** provede voláním metody **HTick** krok automatu pro výpis nápovědy a v případě, že je nápověda ukončena navrací hodnotu ukončovacího znaku. V opačném případě navrací hodnotu #$0.

5.2.46. HelpReady

```
function HelpReady:Word; virtual;
```

Metoda **HelpReady** provede prostřednictvím volání metody **HTick** krok automatu pro výpis nápovědy a jako funkční hodnotu vrátí jeho stav.

5.2.47. HTick

```
procedure HTick; virtual;
```

Metoda **HTick** provede krok automatu pro výpis nápovědy. V prvním kroku automatu je uloženo aktuální okno, metodou SaveTerminalScr je uložen obsah aktuálního okna, jako aktuální okno je nastaveno okno pro výpis nápovědy a do něj je vypsán řetězec nápovědy nastavený metodou **Help** nebo **ReadLnH.**

V dalších krocích jsou odebírány znaky z klávesnice. Je-li z klávesnice odebrán některý znak z množiny znaků pro ukončení nápovědy (zCr, zEsc) je automat ukončen a obnoveno zobrazení v původním okně.

Metoda **HTick** je chráněna proti vícenásobnému vstupu z kontextu více procesů.

5.2.48. DisplayHelp

```
procedure DisplayHelp(var H:string); virtual;
```

Metoda **DisplayHelp** vypíše na displej nápovědu. Je volána metodou **Htick**. Parametr H obsahuje řetězec s textem nápovědy.

5.2.49. UpDnEnable

```
procedure UpDnEnable;
```

Metoda **UpDnEnable** změní význam kurzorových kláves při editaci. Šipky nahoru/dolů budou měnit znak na pozici kurzoru podle typu editovaného řetězce. Typ je udáván parametrem edType. Parametr edType nastavují funkce knihovny UmenuChr automaticky.

5.2.50. UpDnDisable

```
procedure UpDnDisable;
```

Metoda **UpDnDisable** vrátí význam kurzorových kláves do výchozího stavu.