# uTermGr

## JEDNOTKA PRO VYTVÁŘENÍ GRAFICKÝCH TERMINÁLŮ

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 |
| | | | | |

## 1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky pro vytváření grafických terminálů.

## 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, uCharBuf, uATerm a uTermChr.

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 obsahuje deklarace základních typů, konstant a proměnných pro práci s obecným grafickým terminálem. Je zde implementován objektový typ **tTermGr**, který rozšiřuje funkci obecného znakového terminálu na obecný grafický terminál.

Výpis textů je rozšířen o možnost volby textových stylů (umístění a font), jsou implementovány funkce vytvoření stylů z ESC sekvencí, funkce správy deskriptorů grafických objektů (bod, přímka, kružnice ap.) a funkce správy deskriptoru pozadí. Objekt pracuje s indexy do polí uživatelských bitmap a fontů a textovými popisy grafického obsahu obrazovky. Tyto indexy a textové popisy jsou předávány objektu displeje. Ten provádí jejich grafickou interpretaci do videostránek a přenáší je na fyzický displej.

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

# 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.

```
cNumFonts=20;
```

Konstanta **cNumFonts** udává maximální počet uživatelských fontů. Pole uživatelských fontů je deklarováno v jednotce implementující grafické operace do konkrétní videopaměti.

```
BkBmpsCount=20;
```

Konstanta **BkBmpsCount** udává maximální počet uživatelských bitmap. Pole uživatelských bitmap je deklarováno v jednotce implementující grafické operace do konkrétní videopaměti.

```
MaxTextStylesCount=20;
```

Konstanta **MaxTextStylesCount** udává maximální počet textových stylů na jedné obrazovce.

```
tVideoRop=(b_OR,b_NOT,b_XOR);
```

**tVideoRop** je výčtový typ grafických operací. Význam grafických operací je v následující tabulce:

| | | |
|---|---|---|
| b_OR | operace OR - objekt je vykreslen černě | |

| | | |
|---|---|---|
| b_NOT | operace NOT - objekt je vykreslen bíle | |
| b_XOR | operace XOR - objekt je vykreslen jako inverse pozadí | |

```
tGrObj_Type = (grobj_None,
               grobj_Point,
               grobj_Line,
               grobj_Rectangle,
               grobj_FilledRect,
               grobj_Circle);
```

**tGrObj_Type** je výčtový typ grafických objektů. Je používán jako selektor v typu **tGraphicProperty**.

```
tGraphicProperty = record
  case GrObj_Type :tGrObj_Type of
    grobj_Point     :(X,Y         :integer);
    grobj_Line,
    grobj_Rectangle,
    grobj_FilledRect:(X1,Y1,X2,Y2 :integer);
    grobj_Circle    :(CX,CY,R     :integer);
end;
```

**tGraphicProperty** je typ pro popis grafického objektu. Implementované grafické objekty a význam položek variantního záznamu je v následující tabulce:

| druh gr.objektu | identifikátor | význam položek záznamu |
|---|---|---|
| bod | grobj_Point | X,Y - horizontální a vertikální souřadnice bodu |
| úsečka | grobj_Line | X1,Y1 - souřadnice začátku úsečky<br>X2,Y2 - souřadnice konce úsečky |
| obdélník | grobj_Rectangle | X1,Y1 - souřadnice jednoho z rohů obdélníka<br>X2,Y2 - souřadnice protilehlého rohu obdélníka |
| vyplněný obdélník | grobj_FilledRect | X1,Y1 - souřadnice jednoho z rohů obdélníka<br>X2,Y2 - souřadnice protilehlého rohu obdélníka |
| kružnice | grobj_Circle | CX,CY - souřadnice středu kružnice<br>R - poloměr kružnice |

```
tTextStyleProperty = record
  FontIndex      :integer;                                    }
  OrigX          :integer;
  OrigY          :integer;
```

```
  TxtRop          :tVideoROP;
end;
```

**tTextStyleProperty** je typ pro popis textového stylu. Význam položek záznamu je v následující tabulce:

| položka | význam |
|---|---|
| FontIndex | Index do pole uživatelských fontů. |
| OrigX | Horizontální souřadnice levého horního rohu textu. |
| OrigY | Verikální souřadnice levého horního rohu textu. |
| TxtRop | Grafická operace, kterou se bude text vypisovat. |

```
tTextStyleCount   = 0..MaxTextStylesCount;
```

```
tTextStylesIndex  = 0..MaxTextStylesCount-1;
```

Výčtový typ **tTextStylesIndex** je používán pro indexování jednotlivých stylů v seznamu stylů.

```
PStyleMap         = ^TPStyleMap;
TPStyleMap        = array[0..$fff0 div SizeOf(TTextStylesIndex)] of
                                                    TTextStyleCount;
```

Typ **TPStyleMap** představuje mapu stylů. Přiřazuje každé pozici znakového bufferu index stylu, který má být pro výpis příslušného znaku použit.

```
TTextStyleList    = array[TTextStylesIndex] of TTextStyleProperty;
```

Typ **TTextStyleList** představuje seznam stylů. Přiřazuje indexům jednotlivých použitých stylů popis stylu.

```
TGrDataChfFlag    = (Chf_StyleMap,
                     Chf_TextStyle,
                     Chf_GraphicDscr,
                     Chf_BkGroundIndex);
```

Výčtový typ **TGrDataChfFlag** je výčet příznaků změn v datech popisujících grafický obsah obrazovky. Je použit pro tvorbu typu **TGrDataChfFlags**.

```
TGrDataChfFlags   = set of TGrDataChfFlag;
```

Typ **TGrDataChfFlags** je množina příznaků změn v datech popisujících grafický obsah obrazovky.

# 5. Popis objektu tGraphicPropReaderWriter

```
tGraphicPropReaderWriter = object(tPropertyReaderWriter);
```

Objektový typ **tGraphicPropReaderWriter** slouží k dekódování deskriptorů grafických objektů z textové podoby.

## 5.1. Proměnné

```
Property   :tGraphicProperty;
```

**Property** je popis přečteného grafického objektu.

```
Rop        :tVideoROP;
```

**Rop** je grafická operace, kterou má být objekt vykreslen (implicitně b_OR).

```
MaxX       :integer;
```

**MaxX** je maximální velikost horizontální souřadnice.

```
MaxY       :integer;
```

**MaxY** je maximální velikost vertikální souřadnice.

## 5.2. Metody

### 5.2.1. Init

```
constructor Init(NewMaxX,NewMaxY:integer);
```

Konstruktor **Init** zinicializuje předka objektu, proměnné **MaxX** a **MaxY** nastaví podle parametrů konstruktoru, proměnnou **Property** naplní obsahem "žádný objekt" a proměnnou **Rop** nastaví na implicitní grafickou operaci (b_OR).

### 5.2.2. ReadProperty

```
procedure ReadProperty(var j:integer;const S:string);virtual;
```

Metoda **ReadProperty** překrývá abstraktní metodu předka objektu. Metoda přečte a dekóduje jeden grafický deskriptor ze stringu **S**. Deskriptor je čten od pozice dané parametrem **j**. Popis přečteného deskriptoru je uložen do proměnné **Property**. Proměnná **Rop** je zároveň naplněna požadovanou grafickou operací. Dojde-li při dekódování k chybě je nastavena proměnná **Err** na true, při úspěšném čtení je nastavena na false. V parametru **j** je zároveň vracena koncová poloha čtení v řetězci **S**.

## 5.3. Syntaxe deskriptorů grafických objektů:

| objekt | syntaxe | příklad |
|---|---|---|
| bod | P(<OrigX>,<OrigY>{,<Rop>}) | P(30,10)<br>P(30,10,2) |
| úsečka | P(<$OrigX_1$>,<$OrigY_1$>,<$OrigX_2$>,<$OrigY_2$>{,<Rop>}) | L(0,100,50,120)<br>L(0,10,5,120,1) |
| obdélník | P(<$OrigX_1$>,<$OrigY_1$>,<$OrigX_2$>,<$OrigY_2$>{,<Rop>}) | R(10,10,50,20)<br>R(10,10,50,20,0) |
| vyplněný obdélník | P(<$OrigX_1$>,<$OrigY_1$>,<$OrigX_2$>,<$OrigY_2$>{,<Rop>}) | F(20,10,50,20)<br>F(20,10,50,20,0) |
| kružnice | P(<OrigX>,<OrigY>,<Radius>{,<Rop>}) | C(100,50,30)<br>C(100,50,30,2) |
| <Rop> = 0\|1\|2 | 0 = b_OR, 1=b_NOT, 2=b_XOR | |

# 6. Popis objektu tStylePropertyReaderWriter

```
TStylePropReaderWriter = object(TPropertyReaderWriter)
```

Objektový typ **tStylePropReaderWriter** slouží k dekódování textových stylů z textové podoby.

## 6.1. Proměnné

```
StyleMapPos  :integer;
```
Proměnná **StyleMapPos** obsahuje pozici přečteného stylu ve znakovém bufferu.

```
Property     :TTextStyleProperty;
```
Proměnná **Property** obsahuje popis přečteného stylu.

```
MaxX       :integer;
```
**MaxX** je maximální velikost horizontální souřadnice.

```
MaxY       :integer;
```
**MaxY** je maximální velikost vertikální souřadnice.

## 6.2. Metody

### 6.2.1. Init

```
constructor Init(NewMaxX,NewMaxY:integer);
```
Konstruktor **Init** zinicializuje předka objektu, proměnné **MaxX** a **MaxY** nastaví podle parametrů konstruktoru, proměnnou **Property** naplní popisem implicitního stylu (implicitní font 0, souřadnice 0,0, operace b_OR), a proměnnou **StyleMapPos** na počátek znakového bufferu (0).

### 6.2.2. ReadProperty

```
procedure ReadProperty(var j:integer;const S:string);virtual;
```
Metoda **ReadProperty** překrývá abstraktní metodu předka objektu. Metoda přečte a dekóduje popis jednoho stylu ze stringu **S**. Popis stylu je čten od pozice dané parametrem **j**. Popis přečteného stylu je uložen do proměnné **Property**. Proměnná **StyleMapPos** je zároveň naplněna požadovanou polohou stylu ve znakovém bufferu. Dojde-li při dekódování k chybě je nastavena proměnná **Err** na true, při úspěšném čtení je nastavena na false. V parametru **j** je zároveň vracena koncová poloha čtení v řetězci **S**.

### 6.2.3. WriteProperty

```
procedure WriteProperty(DestBuf:pointer;DestSize:word);virtual;
```
Metoda **WriteProperty** překrývá abstraktní metodu předka objektu. Metodě je předán ukazatel na buffer **DestBuf**, který slouží k zápisu popisu stylu v textové podobě a velikost bufferu **DestSize**. Do bufferu je zapsán popis stylu z proměnné **Property**, který akceptuje nastavení polohy ve znakovém bufferu z proměnné **StyleMapPos**. Není-li velikost výstupního bufferu dostatečná je nastavena proměnná **Err** na true, při úspěšném zápisu je nastavena na false.

## 6.3. Syntaxe textového popisu stylu

[<StyleMapPos>, <FontIndex>, <OrigX>, <OrigY> {,<Rop>}]

Význam jednotlivých položek je následující:

| | |
|---|---|
| <StyleMapPos> | Poloha stylu ve znakovém bufferu. |
| <FontIndex> | Index do pole uživatelských fontů. |
| <OrigX> | Horizontální souřadnice levého horního rohu výpisu. |
| <OrigY> | Vertikální souřadnice levého horního rohu výpisu. |
| <Rop> | Grafická operace použitá pro výpis stylem.<br>0 = b_OR, 1=b_NOT, 2=b_XOR |

# 7. Popis objektu tGraphicDataMan

```
PGraphicDataMan =^TGraphicDataMan;
TGraphicDataMan = object(TObject)
```

Objekt **tGraphicDataMan** slouží ke správě dat popisujících obsah grafické obrazovky.

## 7.1. Proměnné

```
DisplayOwner          :pADisp;
```

Proměnná **DisplayOwner** obsahuje ukazatel na instanci objektu displeje, který vlastní příslušnou instanci objektu **tGraphicDataMan**.

```
CharRasterWidth    :integer;
CharRasterHeight   :integer;
```

Proměnné **CharRasterWidth** a **CharRasterHeight** obsahují rozměry znakového rastru, při výpisu textu ve znakovém rastru.

```
PixelRasterWidth   :integer;          { rozměr rastru v pixelech }
PixelRasterHeight  :integer;          { rozměr rastru v pixelech }
```

Proměnné **PixelRasterWidth** a **PixelRasterHeight** obsahují rozměry rastru displeje v pixelech.

```
StyleMapPtr        :PStyleMap;
```

Proměnná **StyleMapPtr** obsahuje ukazatel na mapu stylů. Mapa stylů přiřazuje každé pozici ve znakovém bufferu index stylu, který má být použit pro výpis příslušného znaku.

```
StyleMapSize       :word;
```

Proměnná **StyleMapSize** obsahuje alokovanou velikost mapy stylů.

```
UsedStylesCount    :TTextStyleCount;
```

Proměnná **UsedStylesCount** obsahuje počet použitých stylů.

```
UsedStylesList     :TTextStyleList;
```

Proměnná **UsedStylesList** obsahuje seznam použitých stylů. Přiřazuje jednotlivým indexům stylů jejich popis.

```
DscrGraphicStr     :string;
```

Proměnná **DscrGraphicStr** obsahuje deskriptory grafických obrazců obrazovky.

```
BkGroundIndex       :integer;
```

Proměnná **BkGroundIndex** obsahuje index použité podkladové bitmapy. Index -1 má význam prázdného pozadí.

```
GrDataChfFlags      :TGrDataChfFlags;
```

Proměnná **GrDataChfFlags** obsahuje množinu příznaků změn v grafických datech obrazovky.

## 7.2. Metody

### 7.2.1. Init

```
constructor Init(Owner:pADisp;
           CharRasterW,CharRasterH,PixRasterW,PixRasterH:integer);
```

Konstruktor **Init** inicializuje objekt. Nastaví proměnné **DisplayOwner, CharRasterWidth, CharRasterHeight, PixelRasterWidth** a **PixelRasterHeight** podle parametru předaných do konstruktoru. Alokuje paměť pro mapu stylů podle velikosti znakového rastru, smaže styly a deskriptory grafických objektů a nastaví prázdné pozadí.

### 7.2.2. Done

```
destructor Done;virtual;
```

Destruktor Done uvolní paměť alokovanou pro mapu stylů a zruší instanci objektu.

### 7.2.3. ClearStyles

```
procedure ClearStyles;
```

Metoda **ClearStyles** smaže mapu stylů a seznam stylů a nastaví příznaky změn, došlo-li ke změnám v datech.

### 7.2.4. Set_GrDataChfFlg

```
procedure Set_GrDataChfFlg(Flg:TGrDataChfFlag;FlgVal:Boolean);
```

Metoda **Set_GrDataChfFlg** nastaví příznak změny grafických dat předaný v parametru **Flg**. Nastavení se provede pouze je-li parametr **FlgVal** = true.

### 7.2.5. Res_GrDataChfFlg

```
procedure Res_GrDataChfFlg(Flg:TGrDataChfFlag);
```

Metoda **Res_GrDataChfFlg** smaže příznak změny grafických dat předaný v parametru **Flg**.

### 7.2.6. QGrDataChfFlg

```
function  QGrDataChfFlg(Flg:TGrDataChfFlag):Boolean;
```

Metoda **QGrDataChfFlg** vrací true, je-li nastaven příznak předaný v parametru **Flg**.

### 7.2.7. PutCharStyleToMap

```
procedure PutCharStyleToMap(CharIndex:byte);
```

Metoda **PutCharStyleToMap** provede zápis indexu posledního stylu do mapy stylů pro pozici předanou parametrem **CharIndex**. Dojde-li ke změně v mapě stylů nastaví příznak změny v mapě stylů.

### 7.2.8. PutNewStyle

```
procedure PutNewStyle(var NewStyle:TTextStyleProperty);
```

Metoda **PutNewStyle** zapíše nový styl s popisem předaným v parametru **NewStyle** do seznamu stylů, zvýší počet použitých stylů o 1 a nastaví příznak změny stylů.

### 7.2.9. AssignFriedObj

```
procedure AssignFriendObj(pStyleMan:PGraphicDataMan);
```

Metoda **AssignFrienObj** slouží k převzetí dat popisujících obsah grafické obrazovky z jiné instance tohoto objektu. Ukazatel na instanci objektu, ze které mají být data převzata je předán jako parametr **pStyleMan**. Je převzata mapa stylů, seznam stylů, deskriptory grafických objektů a index bitmapy pozadí. Podle potřeby nastaví příznaky změn v datech popisujících grafický obsah obrazovky.

### 7.2.10. AssignGraphicDscr

```
procedure AssignGraphicDscr(const GrStr:string);
```

Metoda **AssignGraphicDscr** zapíše deskriptory grafických objektů předané v parametru **GrStr** do proměnné **DscrGraphicStr**. Dojde-li ke změně, je nastaven příznak změny deskriptorů grafických objektů.

### 7.2.11. AssignBkGroundIndex

```
procedure AssignBkGroundIndex(Index:integer);
```

Metoda AssignBkGroundIndex zapíše index předaný jako parametr do proměnné BkGroundIndex. Dojde-li ke změně, je nastaven příznak změny grafického pozadí.

### 7.2.12. AssignBkGroundDscr

```
procedure AssignBkGroundDscr(const BkStr:string);
```

Metoda **AssignBkGroundDscr** má stejnou funkci jako **AssignBkGroundIndex**, ale index bitmapy pozadí je předán v textové podobě jako parametr **BkStr**.

### 7.2.13. AssignStylesDscr

```
procedure AssignStylesDscr(const StylesStr:string);
```

Metoda **AssignStylesDscr** vypní nově na základě textové popisu předaného parametrem **StylesStr** mapu stylů a seznam stylů.

## 7.2.14. FillTrBufWith_Text

```
function FillTrBufWith_Text(DestBuf:pointer;DestSize:word):word;
```

Metoda **FillTrBufWith_Text** předává data ze znakového bufferu vlastníka instance (objektu displeje) do bufferu, na který ukazuje ukazatel předaný v parametru **DestBuf**. Parametr **DestSize** udává alokovanou velikost cílového bufferu. Metoda vrací počet předaných znaků do cílového bufferu.

## 7.2.15. FillTrBufWith_TextStr

```
function FillTrBufWith_TextStr(DestBuf:pointer;DestSize:word):word;
```

Metoda **FillTrBufWith_TextStr** má stejnou funkci jako **FillTrBufWith_Text**. Do cílového bufferu však nepředává mezery nacházející se na konci znakového bufferu displeje.

## 7.2.16. FillTrBufWith_Styles

```
function FillTrBufWith_Styles(DestBuf:pointer;DestSize:word):word;
```

Metoda **FillTrBufWith_Styles** předává do cílového bufferu, na který ukazuje parametr **DestBuf** ESC sekvenci s popisem stylů grafické obrazovky. V parametru **DestSize** je předávána alokovaná velikost paměti pro cílový buffer. Metoda vrací počet znaků přenesených do cílového bufferu.

Syntaxe ESC sekvence předané do cílového bufferu:

<ESC>[ <StylesList> <ESC>]

<StylesList> = <StylesSpec> | <StylesSpec> <StylesList>

<StylesList> je textový popis jednoho stylu viz popis objektu **tStylesPropReaderWriter**.

## 7.2.17. FillTrBufWith_Graphic

```
function FillTrBufWith_Graphic(DestBuf:pointer;DestSize:word):word;
```

Metoda **FillTrBufWith_Graphic** předává do cílového bufferu, na který ukazuje parametr **DestBuf** ESC sekvenci s popisem stylů grafické obrazovky. V parametru **DestSize** je předávána alokovaná velikost paměti pro cílový buffer. Metoda vrací počet znaků přenesených do cílového bufferu.

Syntaxe ESC sekvence předané do cílového bufferu:

<ESC>( <GrObjectsList> <ESC>)

<GrObjectsList> = <GrObjectSpec> | <GrObjectSpec> <GrObjectList>

<GrObjectSpec> je textový popis jednoho grafického objektu viz popis objektu **tGraphicPropReaderWriter**.

## 7.2.18. FillTrBufWith_BkGround

```
function FillTrBufWith_BkGround(DestBuf:pointer;DestSize:word):word;
```

Metoda **FillTrBufWith_BkGround** předává do cílového bufferu, na který ukazuje parametr **DestBuf** ESC sekvenci s popisem stylů grafické obrazovky. V

parametru **DestSize** je předávána alokovaná velikost paměti pro cílový buffer. Metoda vrací počet znaků přenesených do cílového bufferu.

Syntaxe ESC sekvence předané do cílového bufferu:

<ESC> '<' <BkGroundIndext> <ESC> '>'

<BkGroundIndext> je ascii reprezentace zápisu indexu podkladové bitmapy.

# 8. Popis objektu tTermGr

pTermGr = ^tTermGr;
tTermGr = object(tTermChr)

Objektový typ **tTermGr** je potomkem typu **tTermChr**. Rozšiřuje obecný znakový terminál na obecný grafický terminál. Je určen k odvození objektů konkrétních grafických terminálů.

Nově je implementováno dekódování textových stylů z ESC sekvencí v textovém řetězci a metody pro zápis deskriptorů grafických objektů a zápis indexu uživatelské bitmapy pozadí.

## 8.1. Proměnné

```
TermDataManager:pGraphicDataMan;
```

Proměnná **TermDataManager** obsahuje ukazatel na instanci objektu uchovávající a spravující data, která popisují grafický obsah obrazovky.

```
Saved_TermDataManager:pGraphicDataMan;
```

Proměnná **Saved_TermDataManager** obsahuje ukazatel na instanci objektu uchovávající a spravující dočasně uložená grafická data obrazovky.

```
PixelWidth:integer;
PixelHeight:integer;
```

Proměnné **PixelWidth** a **PixelHeight** uchovávají rozměry rastru displeje v pixelech.

```
pHelpGrStr:^string;
```

Proměnná **pHelpGrStr** obsahuje ukazatel na řetězec s textovým popisem grafických objektů pro nápovědu. Konstruktorem **Init** je nastavena na nil.

```
pHelpBkBmp:^integer;
```

Proměnná **pHelpBkBmp** obsahuje ukazatel na proměnnou s indexem bitmapy pozadí nápovědy. Konstruktorem **Init** je nastavena na nil.

## 8.2. Metody

### 8.2.1. Init

```
constructor Init(NewDisp:pADisp;NewKeyb:pAKeyb;
            NewChnTerm:pChnVirt;NewChnRecBuf:pointer;
            NewPixelWidth,NewPixelHeight:integer);
```

Konstruktor **Init** inicializuje znakový terminál vytvoří instance objektů **TermDataManager** a **Saved_TermDataManager** a nastaví proměnné **PixelWidth** a **PixelHeight** na hodnoty parametrů **NewPixelWidth** a **NewPixelHeight**.

### 8.2.2. Done

```
destructor Done; virtual;
```

Destruktor **Done** zruší existující instance objektů **TermDataManager** a **Saved_TermDataManager** a zruší znakový terminál.

### 8.2.3. WriteS

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

Metoda **WriteS** provádí zápis textového řetězce do znakového bufferu terminálu. Oproti znakovému terminálu je doplněna o funkci dekódování textových stylů z ESC sekvencí. V parametru **S** je předáván textový řetězec. Podle obsahu řetězce metoda vyplňuje znakový buffer terminálu a prostřednictvím **TermDataManager** vytváří seznam a mapu stylů.

### 8.2.4. SetWindowH

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

Metoda **SetWindowH** nastavuje okénko pro nápovědu. Překrytím původní metody se dosahuje nastavení okénka pro nápovědu vždy na celý znakový rastr displeje. Parametry předávané do metody jsou ignorovány.

### 8.2.5. WriteGraphicStr

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

Metoda **WriteGraphicStr** zapisuje prostřednictvím **TermDataManager** řetězec s deskriptory grafických objektů předaný parametrem **S**.

### 8.2.6. WriteBkGroundIndex

```
procedure WriteBkGroundIndex(Index:integer);virtual;
```

Metoda **WriteBkGroundIndex** zapisuje prostřednictvím **TermDataManager** index bitmapy pozadí předaný parametrem **Index**.

### 8.2.7. SaveTerminalScr

```
procedure SaveTerminalScr;virtual;
```

Metoda **SaveTerminalScr** slouží k dočasnému uložení dat popisujících obsah obrazovky. Je rozšířena o uložení grafických dat obrazovky. K uložení grafických dat slouží instance **Saved_TermDataManager**.

### 8.2.8. LoadTerminalScr

```
procedure LoadTerminalScr;virtual;
```

Metoda **LoadTerminalScr** slouží k obnovení dat popisujících obsah obrazovky po dočasném uložení. Je rozšířena o obnovu uložených grafických dat obrazovky. K uložení grafických dat slouží instance **Saved_TermDataManager**.

## 8.2.9. DisplayHelp

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

Metoda **DisplayHelp** vypíše na displej nápovědu. Je volána metodou **Htick**. Oproti předkovi **tTermChr** je metoda rozšířena o vykreslení grafických objektů a bitmapy pozadí nápovědy. Text nápovědy je předáván v parametru **H**, textový popis grafických objektů nápovědy je v proměnné, na kterou ukazuje ukazatel **pHelpGrStr** a index bitmapy pozadí nápovědy je v proměnné, na kterou ukazuje ukazatel **pHelpBkBmp**. Ukazatele **pHelpGrStr** a **pHelpBkBmp** lze nastavit metodou **TermSetHelpVar** tohoto objektu nebo metodou **SetHelpVar** objektu grafického menu **tMenuGr.**

Tyto ukazatele jsou implicitně nastaveny konstruktorem na nil. Pokud uživatel toto nastavení nezmění nebudou se grafické objekty a bitmapa pozadí v nápovědě zobrazovat.

## 8.2.10. TermSetHelpVar

```
procedure TermSetHelpVar(var HGrStr:string;
                         var HBkBmp:integer);virtual;
```

Metoda **TermSetHelpVar** slouží k nastavení ukazatelů na proměnné s textovým popisem grafických objektů a indexu bitmapy pozadí nápovědy. Tyto proměnné jsou předány v parametrech **HGrStr** a **HBkBmp**.