

uATerm

JEDNOTKA PRO PRÁCI S ABSTRAKTNÍM 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>

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 popsáno. 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	5
1.1. Revize dokumentu	5
1.2. Účel dokumentu	5
1.3. Rozsah platnosti	5
1.4. Související dokumenty	5
2.Termíny a definice	5
3.Úvod	6
4.Popis konstant a typů	6
5.Popis objektů	8
5.1. Objekt tAKeyb	8
5.1.1. Pole	8
5.1.2. Metody	8
5.1.2.1. Init constructor	8
5.1.2.2. Done destructor	9
5.1.2.3. SetKeybParam	9
5.1.2.4. GetKeybParam	9
5.1.2.5. FlushKeyb	9
5.1.2.6. ReadKeybBuf	9
5.1.2.7. ResultKeybParam	9
5.1.2.8. KeyPressed	9
5.1.2.9. ReadKey	9
5.1.2.10. ShowKey	9
5.1.2.11. InsertKey	9
5.1.2.12. KeybStatus	10
5.1.2.13. Tick	10
5.1.2.14. KTick	10
5.2. Objekt TCursorAttrReaderWriter	10
5.2.1. Pole	10
5.2.2. Metody	10
5.2.2.1. Init konstruktor	10
5.3. Objekt tADisp	10
5.3.1. Pole	11
5.3.2. Metody	11
5.3.2.1. Init constructor	11
5.3.2.2. Done destructor	11
5.3.2.3. SetDispParam	11
5.3.2.4. GetDispParam	12
5.3.2.5. ResultDispParam	12
5.3.2.6. SuspendRefresh	12
5.3.2.7. ReleaseRefresh	12
5.3.2.8. RefreshSuspended	12
5.3.2.9. GetDispBufPtr	12
5.3.2.10. GetDispBufSize	12
5.3.2.11. WriteCtrlBuf	13
5.3.2.12. WriteCharBuf	13
5.3.2.13. WriteCharBufAsStr	13
5.3.2.14. SetCursorAttr	13
5.3.2.15. SetCursorAttrAsStr	13

5.3.2.16.	FillTrBufWith_CursorAttr	13
5.3.2.17.	Tick	13
5.3.2.18.	UpdateDisplay	13
5.3.2.19.	InvalidateDisplay	13
5.3.2.20.	ShowCursorIfBlink	13
5.3.2.21.	HideCursorIfBlink	13
5.3.2.22.	DTick	14
5.3.2.23.	DTickRefreshScr	14
5.4.	Objekt tATerm	14
5.4.1.	Pole	14
5.4.2.	Metody	15
5.4.2.1.	Init constructor	15
5.4.2.2.	Done destructor	15
5.4.2.3.	SetTermParam	15
5.4.2.4.	GetTermParam	15
5.4.2.5.	ResultTerm	15
5.4.2.6.	SetTDispParam	15
5.4.2.7.	GetTDispParam	15
5.4.2.8.	ResultTDispParam	16
5.4.2.9.	SetTKeybParam	16
5.4.2.10.	GetTKeybParam	16
5.4.2.11.	ResultTKeybParam	16
5.4.2.12.	SetTChnParam	16
5.4.2.13.	GetTChnParam	16
5.4.2.14.	ResultTChnParam	16
5.4.2.15.	FlushKbd	16
5.4.2.16.	KeyPressed	16
5.4.2.17.	ReadKey	16
5.4.2.18.	ShowKey	17
5.4.2.19.	InsertKey	17
5.4.2.20.	KeybStatus	17
5.4.2.21.	Suspend_DispRefresh	17
5.4.2.22.	Release_DispRefresh	17
5.4.2.23.	Beep	17
5.4.2.24.	ShowFirstScreen	17
5.4.2.25.	SetupTerminal	17
5.4.2.26.	StoreTerminalSetup	17
5.4.2.27.	LoadTerminalSetup	17
5.4.2.28.	Tick	18
5.4.2.29.	DTick_GetDisplayData	18
5.4.2.30.	SendDataToRemoteTerm	18
5.4.2.31.	ReceiveDataFromRemoteTerm	18

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	We		První vydání
1.10	2.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000. Opravená hlavička tADisp.GetDispParam. Doplněný popis fce tADisp.ShowCursorIfBlink a HideCursorIfBlink. Doplněná proměnná tATerm.Echar. Upravený typ ParamStr → tParamStr. Uvedený objekt TCursorAttrReaderWriter

1.2. Účel dokumentu

Tento dokument slouží jako popis knihovny abstraktního terminálu uATerm.

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 uCharBuf.

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 **uATerm** obsahuje abstraktní objekt **tATerm**, implementující obecné chování displeje a klávesnice v abstraktních objektech **tADisp** a **tAKeyb**. Definiuje virtuální rozhraní pro všechny objekty, které se na objekt terminálu odkazují. Metody objektu **tATerm** umí nastavovat a zjišťovat stav virtuální klávesnice, displeje a komunikačního kanálu. Ty potom využívají dědici těchto objektů, kteří si je dodefinují, popř. přepíší tak, aby jim vyhovovaly a aby pracovaly se skutečnou klávesnicí, displejem a komunikačním kanálem. **tAKeyb** definuje rozhraní abstraktní klávesnice, obsluhu jejího bufferu a objekt **tADisp** implementuje abstraktní displej, umí nastavit parametry kurzoru a rozměry plochy displeje. Většina metod je abstraktních a musí je dodefinovat dědici objektu.

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.

V jednotce jsou předdefinovány znakové a řetězcové konstanty, které jsou využívány především při zpracovávání řetězců pro výpis na displej a znaků přicházejících z klávesnice, od uživatele. Předdefinované znaky:

zBell	- znak pro Bell
zLf	- line feed
zCr	- carriage return
zBs	- backspace
zHt	- horizont. tabulátor
zTab	- znak pro tabulátor
zSBs	- výmaz řádku doleva do konce = rychlý BS
zEsc	- znak pro ESC
zSpace	- znak pro mezeru
zDel	- znak Del pro výmaz znaku
zUp	- cursor up
zDn	- cursor down
zLe	- cursor left
zRi	- cursor right
zFx	- znaky funkčních kláves F1-F10
zHelp	- znak pro nápovědu (F1)
zHome	- znak pro Home
zPgUp	- znak pro PgUp
zPgDn	- znak pro PgDn
zEnd	- znak pro End
zIns	- znak pro Insert
zClrScr	- znak pro Smazání obrazovky
zCursorPc	- znak pro kurzor simulátoru
zCursor	- znak pro kurzor terminálu TERM01
zStart	- znak pro klávesu Start na klávesnici TERM10
zStop	- znak pro klávesu Stop na klávesnici TERM10
zShiftEnter	- znak pro vstup do Terminal Setup

Předdefinované řetězce:

TimeoutStr - je string, který se přesune do bufferu klávesnice při vypršení timeoutu
BellStr - řetězec pro Bell
HomeStr - řetězec pro Home

Esc sekvence je řetězec, ve kterém je možné definovat vzhled obrazovky. Vlastní Esc sekvence a metody, které je vypisují na obrazovku jsou implementovány v jednotkách pro práci se znakovým a grafickým terminálem. Jejich základní znaky jsou však definovány již v jednotce uATerm. Řídící znaky "Escape" sekvencí:

```

esc_ScrBegin      = '{';  { ESC char begin screen  string }
esc_ScrEnd        = '>';  { ESC char end   screen  string }
esc_GraphicBegin  = '(';  { ESC char begin graphic string }
esc_GraphicEnd    = ')';  { ESC char end   graphic string }
esc_BkGroundBegin = '<';  { ESC char begin BkGround string }
esc_BkGroundEnd   = '>';  { ESC char end   BkGround string }
esc_StylesBegin   = '[';  { ESC char begin styles  string }
esc_StylesEnd     = ']';  { ESC char end   styles  string }
esc_CursorBegin   = '@';  { ESC char begin styles  string }
esc_CursorEnd     = '&';  { ESC char end   styles  string }
esc_LedBegin      = '!';  { ESC char begin led    string }
esc_LedEnd        = '?';  { ESC char end   led    string }
esc_SoundBegin    = '*';  { ESC char begin sound  string }
esc_SoundEnd      = '/';  { ESC char end   sound  string }
esc_OutBegin      = ''';  { ESC char begin out   string }
esc_OutEnd        = '"';  { ESC char end   out   string }

```

typy:

```

type
  tBuf    = array[0..$fff0] of Char;
  pBuf    = ^tBuf;

```

tBuf je typ použitý pro buffer abstraktní klávesnice a displeje. Definuje pole, do kterého se zapisují znaky, které buď přišly ze skutečné klávesnice, nebo se mají vypsat na displej. Typ je základem rozhraní abstraktní klávesnice a displeje.

```

tCharPoint = record
  X,Y : byte;
end;

```

tCharPoint je typ znakového bodu displeje.

```

tCharRect = record
  X,Y,W,H : byte;
end;

```

tCharRect je typ znakové obdélníkové plochy.

```

tCharWin = object
  X,Y : byte; kurzoru.
  Rect : tCharRect;
  procedure AssignCursorPos(cx,cy:byte);
end;

```

tCharWin je typ obsahující umístění a rozměry znakového okna.

```

tSetChar = set of Char;

tEvBeep = (evBeep_Off,      {vypnutí  pípnutí}
           evBeep_On,      {zapnutí  pípnutí}
           evBeep_OnOff);  {krátké pípnutí }

tCursorMode = (CursorOff,   {vypnutý kurzor }
               CursorOn,    {zapnutý kurzor }
               CursorBlink); {blikající kurzor}

tDispBufState = (DispSts_Clear,      { zobrazovaná část bufferu }
                 DispSts_Accepted,   { obsahuje samé mezery }
                 DispSts_Modified);  { buffer obsahuje data, která }
                                     { jsou již zobrazena }
                                     { obsah bufferu byl změněn }

```

tEdit je typ obsahující umístění a rozměry znakového okna.

```

tEdit = (edStrings, edBinNum, edDecNum, edHexNum, edASCII,
edLetters, edUnsDec, edRealNum);
{definuje typ editovaného znaku pro UpDn mod editace}
{edStrings-[$3A..$60]} {edBin-[' ', '0', '1']}
{edDecNum-[' ', '-', '0'..'9']} {edHexNum - [' ', '0'..'9', 'A'..'F']}
{edLetters - [' ', 'A'..'Z', '0'..'9']} {edUnsDec - [' ', '0'..'9']}
{edRealNum - [' ', '-', '+', 'E', '0'..'9']}

```

5. Popis objektů

5.1. Objekt tAKeyb

Objekt **tAKeyb** je abstraktní objekt a až jeho dědicové provádějí napojení na fyzickou klávesnici. Pracuje s bufferem virtuální klávesnice typu **tBuf**. Objekt jej umí mazat, naplňovat a číst z něj znaky. O naplnění bufferu virtuální klávesnice znaky ze skutečné klávesnice se postarají až konkrétní dědici objektu **tAKeyb**, a to přepsáním metody **Ktick**, která je periodicky volána z metody **Tick**. Objekt dále umí nastavovat parametry klávesnice. Je také využíván objektem abstraktního terminálu **tATerm** pro práci s klávesnicí.

```

pAKeyb = ^tAKeyb;
tAKeyb = object(tObject)

```

5.1.1. Pole

OwnerTerm je reference na terminál, který tento objekt vlastní.

```
OwnerTerm : pATerm;
```

5.1.2. Metody

5.1.2.1. Init constructor

```
constructor Init(TermOwner:pATerm;Len:word);
```


Při inicializaci se zadá ukazatel na vlastníka objektu a délka vyrovnávacího bufferu klávesnice, tj. max. počet znaků, které bude moci buffer obsahovat. Buffer je vytvořen ve volné paměti.

5.1.2.2. Done destructor

```
destructor Done; virtual;
```

Destruktor uvolňuje buffer pro znaky přijaté z klávesnice z paměti.

5.1.2.3. SetKeybParam

```
procedure SetKeybParam(const S:tParamStr); virtual;
```

Metoda nastaví parametry klávesnice.

5.1.2.4. GetKeybParam

```
function GetKeybParam(const KeyStr:tParamStr):tParamStr; virtual;
```

Metoda navrátí nastavení klávesnice.

5.1.2.5. FlushKeyb

```
procedure FlushKeyb; virtual;
```

Metoda vyprázdní vyrovnávací buffer klávesnice.

5.1.2.6. ReadKeybBuf

```
function ReadKeybBuf(DstBuf:pBuf;DstSize:word):word; virtual;
```

Metoda předá vyrovnávací buffer, vrací jeho délku.

5.1.2.7. ResultKeybParam

```
function ResultKeybParam: Boolean; virtual;
```

Metoda navrátí výsledek inicializace klávesnice; False => v pořádku.

5.1.2.8. KeyPressed

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

Metoda vrací true, je-li ve vyrovnávacím bufferu klávesnice k dispozici znak.

5.1.2.9. ReadKey

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

Metoda vybere z vyrovnávacího bufferu klávesnice znak a dodá ho jako svou funkční hodnotu. Je-li buffer prázdný, navrátí znak #0.

5.1.2.10. ShowKey

```
function ShowKey: Char; virtual;
```

Metoda navrátí hodnotu znaku, který je ve vyrovnávacím bufferu klávesnice, ale z vyrovnávacího bufferu jej nevyjme. Je-li buffer prázdný, navrátí znak #0.

5.1.2.11. InsertKey

```
procedure InsertKey(Key: Char); virtual;
```

Metoda vloží znak **Key** na konec vyrovnávacího bufferu klávesnice.

5.1.2.12. KeybStatus

function KeybStatus: Word; virtual;

Metoda vrací status klávesnice - stlačené klávesy, soutisky.

5.1.2.13. Tick

procedure Tick; virtual;

Metoda volá periodicky **KTick**, čímž se zabezpečí pravidelný přenos znaků z klávesnice do vyrovnávacího bufferu. Je volána z procesu "TICK" a pomocí flagu **KTickRunning** je ochráněn vícenásobný vstup do procedury **KTick**. Skutečnost, že se činnost **KTick** někdy neprovede, nesmí být na závadu.

5.1.2.14. KTick

procedure KTick; virtual;

Metodu musí dodefinovat dědic - měla by zabezpečit dotaz na konkrétní klávesnici a přesun stisknutých znaků klávesnice do vyrovnávacího bufferu.

5.2. Objekt TCursorAttrReaderWriter ---

Objekt popisuje nastavení atributů kurzoru.

5.2.1. Pole

A_CursorBufPos udává offset pozice kurzoru vůči počátku znakového bufferu.

A_CursorBufPos : Word;

A_ShowCursor určuje zda má být kurzor viditelný.

A_ShowCursor : Boolean;

5.2.2. Metody

5.2.2.1. Init konstruktor

constructor Init;

Konstruktor **Init** inicializuje objekt, nastavuje jeho proměnné. Offset pozice kurzoru je nastaven na nulu a je nastaven neviditelný kurzor.

5.3. Objekt tADisp ---

Objekt **tADisp** je virtuální objekt a až jeho dědicové provádějí napojení na fyzický displej. Obsahuje buffer abstraktního displeje, který bude vypsán na skutečný displej. Buffer je typu **tBuf**. Objekt umí do bufferu zapisovat znaky, vyplnit jej znakem kurzoru a mazat jej. Pravidelné obnovování displeje, tj. zápis bufferu na displej zajišťuje metoda **Tick**, která volá metodu **Dtick**. Ta, pokud je třeba, zavolá metodu pro obnovení displeje **DTickRefreshScr**, která obsluhuje pouze předání všech potřebných dat pro přepsání displeje, sama však displej neobnoví, to bude muset zajistit metoda dědice objektu, který bude mít toto dodefinováno. Fyzický zobrazovač

může používat buď vlastní kurzor, pak dědicové metod ovládající kurzor musí umět tento ovládat, nebo používat kurzor generovaný objektem **tADisp**. V tom případě se fyzický kurzor potlačí a nastaví se viditelný kurzor v objektu **tADisp**. Tento může být stále viditelný, blikající, nebo neviditelný. Objekt umí dále nastavovat parametry displeje a atributy kurzoru.

```
pADisp = ^tADisp;
tADisp = object(tObject)
```

5.3.1. Pole

OwnerTerm je reference na terminál, který tento objekt displeje vlastní.

```
OwnerTerm          : pATerm;
```

DispCharSizeX a **DispCharSizeY** určují počet znakových sloupců a řádků displeje.

```
DispCharSizeX      : byte;
```

DispShowBufOfs Ofset počátku znakového bufferu se znaky ke zobrazení.

```
DispShowBufOfs     : word;
```

5.3.2. Metody

5.3.2.1. Init constructor

```
constructor Init(TermOwner:pATerm;CharColls,CharRows:byte;
NewDispShowBufOfs:word);
```

Při inicializaci se ukazateli na vlastníka objektu **OwnerTerm** přiřadí **TermOwner**. Odkazy **CharColls** a **CharRows** říkáme, kolik má mít vytvářený displej sloupců a řádek. Pokud tyto překračují hranice **MaxDispColumns** nebo **MaxDispRows**, tj. maximální počet sloupců a řádků, nastaví se tyto maximální hodnoty. Dále se určí místo, od kterého začínají v bufferu vlastní znaky pro výpis na displej, to určujeme v parametru **NewDispShowBufOfs**. Buffer je vytvořen ve volné paměti. Metoda dále nastaví vlastnosti kurzoru **CursorVisible**, která říká, jestli je kurzor viditelný a **ShowCursor**, která určuje, zda se má kurzor ukazovat. Obě jsou implicitně nastaveny na false.

5.3.2.2. Done destructor

```
destructor Done; virtual;
```

Metoda uvolní buffer displeje z paměti.

5.3.2.3. SetDispParam

```
procedure SetDispParam(S: ParamStr); virtual;
```

Tato metoda nastavuje proměnné objektu **tADisp**. String **S** obsahuje seznam parametrů. Oddělovačem mezi parametry je mezera. V objektu **tADisp** jsou implementovány tyto parametry:

REF=aaa

Parametrem definujeme, zda je povoleno nebo zakázáno automatické obnovování obsahu obrazovky. aaa může nabývat hodnot **‘OFF’** nebo **‘ON’**.

CCH=bbb

Parametrem definujeme znak, který je zobrazován jako kurzor. bbb může být libovolný znak.

CMO=ccc

Parametr definuje typ kurzoru. ccc může nabývat hodnoty 0 pro neviditelný kurzor, hodnoty 1 pro kurzor trvale svítící nebo hodnoty 2 pro blikající kurzor. Tento typ kurzoru je poté generován objektem **tADisp**. Při neúspěšném pokusu nastavení parametrů se přiřadí **ParamErr** true.

5.3.2.4. GetDispParam

```
function GetDispParam(KeyStr: tParamStr): tParamStr; virtual;
```

Metoda navrácí v **ParamStr** nastavení displeje. Znak pro kurzor se ve stringu objeví jako znak. Nemusí být zobrazitelný, nebo tišitelný. Pomocí parametru **KeyStr** je možné omezit počet parametrů, které funkce vrací. Pokud je **KeyStr** prázdný, budou vráceny všechny parametry, jinak pouze parametry uvedené v **KeyStr**.

5.3.2.5. ResultDispParam

```
function ResultDispParam: Boolean; virtual;
```

Vrátí hodnotu **ParamErr**, tj. úspěšnost nastavení parametrů displeje; false = v pořádku.

5.3.2.6. SuspendRefresh

```
procedure SuspendRefresh;
```

Dočasně potlačí provádění Display refresh.

5.3.2.7. ReleaseRefresh

```
procedure ReleaseRefresh;
```

Uvolní provádění Display refresh, pokud bylo povoleno.

5.3.2.8. RefreshSuspended

```
function RefreshSuspended: Boolean;
```

Vrací stav provádění Display refresh.

5.3.2.9. GetDispBufPtr

```
function GetDispBufPtr: pointer;
```

Vrací pointer na počátek zobrazované části znakového bufferu.

5.3.2.10. GetDispBufSize

```
function GetDispBufSize: word;
```

Vrací délku zobrazované části znakového bufferu.

5.3.2.11. WriteCtrlBuf

```
procedure WriteCtrlBuf(CtrlBuf:pointer;CtrlBufSize:word);
```

Zápis do řídicí části bufferu.

5.3.2.12. WriteCharBuf

```
procedure WriteCharBuf(CharBuf:pointer;CharBufSize:word);
```

Zápis do znakové části bufferu.

5.3.2.13. WriteCharBufAsStr

```
procedure WriteCharBufAsStr(CharStr:String);
```

Zápis do znakové části bufferu.

5.3.2.14. SetCursorAttr

```
procedure SetCursorAttr(CX,CY:byte;NewShowCursor:Boolean);
```

Nastavení pozice a zobrazení znakového kurzoru.

5.3.2.15. SetCursorAttrAsStr

```
procedure SetCursorAttrAsStr(CursorStr:string);
```

Nastavení kurzoru jako String.

5.3.2.16. FillTrBufWith_CursorAttr

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

Naplní Buffer atributy kurzoru.

5.3.2.17. Tick

```
procedure Tick; virtual;
```

Metoda volá metodu **DTick**. Sama je volána z procesu terminál "Tick" a díky ní je zabezpečeno pravidelné obnovování displeje. To jest přenášení obsahu videopaměti do bufferu virtuálního displeje a jeho zobrazení na displej.

5.3.2.18. UpDateDisplay

```
procedure UpDateDisplay;
```

Metoda způsobí přímé zobrazení dat na displej.

5.3.2.19. InvalidateDisplay

```
procedure InvalidateDisplay;
```

Metoda způsobí co nejrychlejší zobrazení dat na displej.

5.3.2.20. ShowCursorIfBlink

```
procedure ShowCursorIfBlink;
```

Pomocí této metody se zobrazí kurzor a vynuluje časovač pro blikání kurzoru.

5.3.2.21. HideCursorIfBlink

```
procedure HideCursorIfBlink;
```

Pomocí této metody se skryje kurzor a vynuluje časovač pro blikání kurzoru.

5.3.2.22. DTick

```
procedure DTick; virtual;
```

Metoda zajišťuje blikání kurzoru a výpis na displej, dále brání reentrantnímu provádění a volání metody **DtickRefreshScr**.

5.3.2.23. DTickRefreshScr

```
procedure DTickRefreshScr; virtual;
```

Metoda pro zobrazení obsahu videopaměti na displeji. Zkopíruje videopaměť do výstupního bufferu **BuffRef**[^]. Tento buffer má na začátku **RefOfs** volných znaků pro nastavení fyzického zobrazovače. Je například možné přidat zde sekvenci pro příkaz Home. Pokud se má zobrazovat kurzor z tohoto objektu, tak metoda na místo kurzoru zkopíruje do výstupního bufferu znak pro interpretaci kurzoru. V případě blikání kurzoru metoda v nastavené periodě jednou kurzor zobrazí a podruhé ne; metoda musí být pro další činnost dodefinována dědici.

5.4. Objekt tATerm

Objekt **tATerm** je základním objektem jednotky uATerm. Jeho metody obsluhují fiktivní terminál, tj. virtuální displej, klávesnici a komunikační kanál. To znamená, že veškeré zápisy a editace displeje probíhají pouze v obrazové paměti a ta se periodicky blokově přenáší do fyzického zobrazovacího zařízení. To však mají definováno až dědici těchto objektů. Většina metod objektů **tADisp** a **tAKeyb** má svého jmenovce v objektu **tATerm**, který je volá. Uživatel proto může při přístupu k obrazovce a klávesnici používat pouze metody objektu **tATerm**.

```
pATerm = ^tATerm;
tATerm = object(tObject)
```

5.4.1. Pole

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

Disp je ukazatel na objekt **tADisp**, implementující abstraktní displej. Přes tento ukazatel se volají metody objektu **tADisp**.

```
Disp : pADisp;
```

Keyb je ukazatel na objekt **tAKeyb**, implementující abstraktní klávesnici. Přes tento ukazatel se volají metody objektu **tAKeyb**.

```
Keyb : pAKeyb;
```

ChnTerm je ukazatel na objekt **tChnVirt**, což je objekt nainicializovaného komunikačního kanálu.

```
ChnTerm : pChnVirt;
```

ChnRecBuf je pointer, ve kterém je uložena referenční adresa přijímacího bufferu komunikačního kanálu.

ChnRecBuf : pointer;

ParamErr je true, pokud nastala chyba parametru terminálu.

ParamErr : Boolean;

V Proměnné **EChar** je uložen znak, který se bude nahrazovat ESC.

Echar : Char;

5.4.2. Metody

5.4.2.1. Init constructor

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

NewDisp, **Keyb** a **NewChnTerm** jsou ukazatelé na instance objektů **tADisp**, **tAKeyb** a **tChnVirt**. Konstruktor je přiřadí referencím terminálu **Disp**, **Keyb** a **ChnTerm**, referencím **OwnerTerm** těchto objektů (mimo **NewChnTerm**^) přiřadí adresu objektu terminálu. **NewChnRecBuf** je adresa přijímacího bufferu komunikačního kanálu; ta se přiřadí pointeru **ChnRecBuf**.

5.4.2.2. Done destructor

```
destructor Done; virtual;
```

Destructor volá destruktory objektů Displeje a Klávesnice **Disp** a **Keyb**.

5.4.2.3. SetTermParam

```
procedure SetTermParam(S:tParamStr); virtual;
```

Metoda pro nastavení terminálu pomocí stringu s parametry. String může být 'RAT=x', kde x je číslo 1 až 255. Nastavuje proměnnou ParamErr, která označuje výsledek operace.

5.4.2.4. GetTermParam

```
function GetTermParam(KeyStr:tParamStr):tParamStr; virtual;
```

Metoda navrácí ve stringu nastavení terminálu. Jedná se o inverzní metodu k metodě SetTerm.

5.4.2.5. ResultTerm

```
function ResultTerm:Boolean; virtual;
```

Metoda navrátí výsledek operace (hodnotu ParamErr), False = v pořádku.

5.4.2.6. SetTDispParam

```
procedure SetTDispParam(S: tParamStr); virtual;
```

Metoda pro nastavení displeje. Ve svém těle volá metodu **Disp**^.**SetDisp** s parametrem S.

5.4.2.7. GetTDispParam

```
function GetTDispParam(KeyStr: tParamStr): tParamStr; virtual;
```

Metoda navrácí nastavení displeje. Ve svém těle volá metodu **Disp[^].GetDispParam**.

5.4.2.8. ResultTDispParam

```
function ResultTDispParam: Boolean; virtual;
```

Metoda navrátí výsledek inicializace displeje, False = v pořádku.

5.4.2.9. SetTKeybParam

```
procedure SetTKeybParam(S: tParamStr); virtual;
```

Metoda nastaví klávenici dle parametru **S**. Ve svém těle volá metodu **Keyb[^].SetKeyb** s parametrem **S**.

5.4.2.10. GetTKeybParam

```
function GetTKeybParam(KeyStr: tParamStr): tParamStr; virtual;
```

Metoda vrátí nastavení klávenice. Volá metodu **Keyb[^].GetKeyb**.

5.4.2.11. ResultTKeybParam

```
function ResultTKeybParam: Boolean;
```

Metoda navrátí výsledek inicializace klávesnice, False = v pořádku.

5.4.2.12. SetTChnParam

```
procedure SetTChnParam(S: tParamStr); virtual;
```

Nastavuje parametry virtuálního komunikačního kanálu. Ve svém těle volá metodu **ChnTerm[^].ChSetParam**.

nastavení komunikace

5.4.2.13. GetTChnParam

```
function GetTChnParam(KeyStr: tParamStr): tParamStr; virtual;
```

Metoda navrátí nastavení komunikace.

5.4.2.14. ResultTChnParam

```
function ResultTChnParam: Boolean; virtual;
```

Metoda navrátí výsledek nastavení komunikačního kanálu, False = v pořádku.

5.4.2.15. FlushKbd

```
procedure FlushKbd; virtual;
```

Metoda vyprázdní vyrovnávací buffer klávesnice. Ve svém těle volá metodu **Kbd[^].FlushKbd**.

5.4.2.16. KeyPressed

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

Metoda navrácí true, je-li ve vyrovnávacím bufferu klávesnice k dispozici znak. Ve svém těle volá metodu **Keyb[^].KeyPressed**.

5.4.2.17. ReadKey

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


Metoda navrácí hodnotu stisknutého znaku. Ve svém těle volá metodu **Keyb^.ReadKey**.

5.4.2.18. ShowKey

```
function ShowKey: Char; virtual;
```

Metoda poskytne hodnotu stisknutého znaku, aniž by jí vyjmula z vyrovnávacího bufferu klávesnice. Ve svém těle volá metodu **Keyb^.ShowKey**.

5.4.2.19. InsertKey

```
procedure InsertKey(Key: Char);virtual;
```

Metoda vloží znak **Key** na konec vyrovnávacího bufferu klávesnice. Ve svém těle volá metodu **Keyb^.InsertKey**.

5.4.2.20. KeybStatus

```
function KeybStatus: Word; virtual;
```

Metoda navrátí status klávesnice. Ve svém těle volá metodu **Keyb^.KeybStatus**.

5.4.2.21. Suspend_DispRefresh

```
procedure Suspend_DispRefresh; virtual;
```

Dočasně zruší provádění Display refresh, volá metodu **Disp^.SuspendRefresh**.

5.4.2.22. Release_DispRefresh

```
procedure Release_DispRefresh; virtual;
```

Uvolní provádění Display refresh, pokud bylo povoleno. Ve svém těle volá metodu **Disp^.ReleaseRefresh**.

5.4.2.23. Beep

```
procedure Beep(EvBeep:tEvBeep); virtual;
```

Metoda je abstraktní. Má způsobit vygenerování zvukového znamení.

5.4.2.24. ShowFirstScreen

```
procedure ShowFirstScreen; virtual;
```

Procedura pro počáteční zobrazení inicializovaného terminálu. Musí být definována dědici.

5.4.2.25. SetupTerminal

```
procedure SetupTerminal; virtual;
```

Metoda je abstraktní. Má zajistit počáteční nastavení terminálu.

5.4.2.26. StoreTerminalSetup

```
procedure StoreTerminalSetup; virtual;
```

Metoda je abstraktní. Má uložit do zálohované paměti nastavení terminálu.

5.4.2.27. LoadTerminalSetup

```
procedure LoadTerminalSetup; virtual;
```

Metoda je abstraktní. Má obnovit ze zálohované paměti nastavení terminálu.

5.4.2.28. Tick

```
procedure Tick; virtual;
```

Abstraktní metoda pro periodické volání funkcí terminálu.

5.4.2.29. DTick_GetDisplayData

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

Touto metodou požaduje Disp předání všech dat pro zobrazení, vrací flag nových dat. Je také abstraktní.

5.4.2.30. SendDataToRemoteTerm

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

Abstraktní metoda pro posílání dat po komunikačním kanálu na vzdálený terminál.

5.4.2.31. ReceiveDataFromRemoteTerm

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

Abstraktní metoda pro převzetí obsahu přijímacího bufferu komunikace.