PCDrv

OVLADAČE SIMULÁTORŮ TERMINÁLŮ NA PC PRO VIZUALIZAČNÍ KNIHOVNY PRO JEDNOTKU KIT

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: 23.09.2005
Datum posledního uložení dokumentu: 23.09.2005
(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	1.00	Cr	21.01.2004	První vydání
1.10	5.XX	Net	23.09.2005	Přidán driver pro Touch 51

1.2. Účel dokumentu

Tento dokument slouží jako popis knihovny PCDrv, která je součástí balíku vizualizačních knihoven LIBV pro řídicí jednotky KIT.

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 IoDrv a Controls.

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

3.1. Účel dokumentu PCDrv

Knihovna **PCDrv** slouží ke spuštění aplikace určené pro terminály fy SofCon v simulačním prostředí na PC. Obsahuje ovladače klávesnice (**TPCKeybDriver**), myši (**TPCMouseDriver**) a dva ovladač videokarty první v monochromatickém VGA režimu s rozlišením 640x480 bodů (**TVGAMonoDriver**) a druhý ovladač VGA karty v některém z 256 barevných režimů SVGA karty (**TVESADriver**).
Dokument je též určen pro knihovnu **T51Drv**, která knihovnu **PCDrv** rozšiřuje o funkce řízení jasu dotykového displeje a ovladač video karty (**T51VESADriver**) řídící jednotky Touch 51.

3.1.1. Ovladač klávesnice

Knihovna PCDrv obsahuje třídu **TPCKeybDriver** (viz kapitola 4.1.3), která implementuje ovladač standardní klávesnice připojené k PC. Ovladač vychází z abstraktní třídy **TKeyboardDriver** implementované v knihovně IoDrv.

Ovladač implementuje metodu **GetEvent**, která vrací událost evKeyDown s kódem klávesy umístěné na začátku fronty řadiče klávesnice.

3.1.1.1. Podporované kódy kláves

Ovladač klávesnice **TPCKeybDriver** podporuje kódy speciálních kláves uvedené v následující tabulce. Jednotlivé konstanty s prefixem kb_ jsou popsány v dokumentaci ke knihovně IoDrv.

kbF1	kbCtrlF1	kbAltA	kbAltU	kbInsert
kbF2	kbCtrlF2	kbAltB	kbAltV	kbDelete
kbF3	kbCtrlF3	kbAltC	kbAltW	kbBackSpace
kbF4	kbCtrlF4	kbAltD	kbAltX	kbTab
kbF5	kbCtrlF5	kbAltE	kbAltY	kbEsc
kbF6	kbCtrlF6	kbAltF	kbAltZ	kbShiftTab
kbF7	kbCtrlF7	kbAltG	kbAlt0	kbEnter
kbF8	kbCtrlF8	kbAltH	kbAlt1	kbLeft
kbF9	kbCtrlF9	kbAltI	kbAlt2	kbRight
kbF10	kbCtrlF10	kbAltJ	kbAlt3	kbUp
kbShiftF1	kbAltF1	kbAltK	kbAlt4	kbDown
kbShiftF2	kbAltF2	kbAltL	kbAlt5	kbHome
kbShiftF3	kbAltF3	kbAltM	kbAlt6	kbEnd
kbShiftF4	kbAltF4	kbAltN	kbAlt7	kbPageUp
kbShiftF5	kbAltF5	kbAltO	kbAlt8	kbPageDown
kbShiftF6	kbAltF6	kbAltP	kbAlt9	
kbShiftF7	kbAltF7	kbAltQ		
kbShiftF8	kbAltF8	kbAltR		
kbShiftF9	kbAltF9	kbAltS		
kbShiftF10	kbAltF10	kbAltT		

3.1.2. Ovladač myši

Knihovna PCDrv obsahuje třídu **TPCMouseDriver** (viz kapitola 4.1.1), která implementuje ovladač standardní myši připojené k PC. Ovladač vychází z abstraktní třídy **TMouseDriver** implementované v knihovně IoDrv. Tento ovladač nahrazuje v simulačním prostředí na PC ovladač dotykového panelu terminálu.

Pokud je aplikace spuštěna přímo v prostředí MS-DOS (ne v režimu MS-DOS MS Windows) je nutné, aby byl před spuštěním aplikace nainstalován rezidentní ovladač myši.

3.1.3. Ovladače displeje

Knihovna PCDrv obsahuje dva ovladače displeje. Ovladače vychází z abstraktní třídy **TDisplayDriver** implementované v knihovně IoDrv.

Pro simulaci terminálů s monochromatickým displejem lze vystačit s ovladačem VGA karty v monochromatickém režim, tj. ovladačem **TVGAMonoDriver** (viz kapitola 4.1.4). Tento ovladač umožňuje přepnutí videokarty do monochromatického režimu s rozlišením 640x480 bodů. Tento ovladač funguje jak v DOSu, tak ve všech verzích MS Windows.

Pro simulaci terminálů s barevným displejem je potřeba použít ovladač SVGA karty s VESA BIOSem, tj. ovladač **TVESADriver** (viz kapitola 4.1.5). Tento ovladač umožňuje přepnutí videokarty do libovolného z podporovaných režimů s 8 bity na pixel. Ovladač funguje spolehlivě v DOSu a prostředí MS Window 95 a 98. Pod Windows 2000 ovladač nefunguje a pod Windows XP funguje pouze v některých případech (potřeba vyzkoušet). V případě, že tento ovladač nelze použít, lze k aplikaci přilinkovat knihovnu emulátoru terminálu (viz. knihovna TEDrv).

Knihovna T51Drv obsahuje ovladač displeje řídící jednotky Touch51, tento ovladač vychází z ovladače TVESADriver, umožňuje přepnutí videokarty do libovolného z podporovaných režimů s 8 bitovou barevnou hloubkou. Ovladač je určen pro prostředí DOS v reálném nebo chráněném režimu procesoru.

3.1.4. Použití ovladačů simulátoru terminálu

Následující příklad ukazuje, jak vytvořit základní komponentu aplikace **TApplication** v simulačním prostředí na PC pomocí knihovny PCDrv.

```
var
  App : PApplication;

{$ifndef COLOR}
  { Monochromaticky rezim 640x480 bodu }
  App :=
    New( PApplication, Init (
      New( PInputDriver, Init(
        New( PPCKeybDriver, Init ),
        New( PPCMouseDriver, Init ),
      )),
      New( PMonoVGADriver, Init( 640, 480 ) ),
      @g_AppSettings
```

```
    ));

{$else}
  { Barevny rezim rezim 640x480 bodu a 256 barev }
  App :=
    New( PApplication, Init (
      New( PInputDriver, Init(
        New( PPCKeybDriver, Init ),
        New( PPCMouseDriver, Init ),
      )),
      New( PVESADriver, Init( 640, 480 ) ),
      @g_AppSettings
    ));
{$endif}
```

Následující příklad ukazuje, jak vytvořit základní komponentu aplikace **TApplication** pro Touch51 pomocí knihovny T51Drv.

```
var
  App : PApplication;

  App :=
    New( PApplication, Init (
      New( PInputDriver, Init(
        nil,
        New( PTouchPanelDriver, Init ),
      )),
      New( PT51VesaDriver, Init( 800, 600 ) ),
      @g_AppSettings
    ));
```

Proměnná g_AppSettings obsahuje nastavení ovladače displeje a klávesnice terminálu.

4. Reference

4.1. Třídy

4.1.1. Třída TPCMouseDriver

Třída **TPCMouseDriver** implementuje myši PC. Tato třída vychází z bázové třídy pro implementaci ovladačů myší **TMouseDriver** (viz. dokumentace ke knihovně IoDrv)

```
  PPCMouseDriver = ^TPCMouseDriver;
  TPCMouseDriver = object( TMouseDriver )
  public
    ButtonCount   : Byte;
    Buttons       : Byte;
    Position      : TPoint;

    constructor Init;
    function  Initialize: Boolean; virtual;
    procedure Finalize; virtual;
    procedure GetEvent( var AEvent: TEvent ); virtual;
    procedure SetDblClickDelay( AValue: Integer ); virtual;
    function  GetDblClickDelay: Integer; virtual;
```

```
    procedure SetDblClickArea( AValue: Integer ); virtual;
    function  GetDblClickArea: Integer; virtual;
    procedure SetRepeatDelay( AValue: Integer ); virtual;
    function  GetRepeatDelay: Integer; virtual;
    procedure SetRepeatRate( AValue: Integer ); virtual;
    function  GetRepeatRate: Integer; virtual;
  end;
```

4.1.1.1. Položka TPCMouseDriver.ButtonCount

Položka **ButtonCount** obsahuje počet tlačítek myši. Tj. hodnotu od 1 do 3. Položka je nastavena automaticky při volání metody **Initialize** a je určena pouze pro čtení.

```
ButtonCount   : Byte;
```

4.1.1.2. Položka TPCMouseDriver.Buttons

Položka **Buttons** obsahuje aktuální stav tlačítek myši. Položka je aktualizována automaticky a je určena pouze pro čtení.

```
Buttons       : Byte;
```

Položka obsahuje kombinaci příznaků mbLeft, mbRight a mbCenter pro levé, pravé a prostřední tlačítko myši (viz. dokumentace ke knihovně IoDrv).

4.1.1.3. Položka TPCMouseDriver.Position

Položka **Position** obsahuje aktuální souřadnice myši. Položka je aktualizována automaticky a je určena pouze pro čtení.

```
Position      : TPoint;
```

4.1.1.4. Konstruktor TPCMouseDriver.Init

Konstruktor **Init** provádí inicializaci instance třídy.

```
constructor Init;
```

Parametry:

Konstruktor nemá žádné parametry.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor nastaví parametry ovladače na implicitní hodnoty, tj.

Oblast dvojkliku	2 pixely
Zpoždění detekce dvojkliku	300 ms
Zpoždění před opakováním	500 ms
Perioda opakování	150 ms

4.1.1.5. Metoda TPCMouseDriver.Initialize

Metoda **Initialize** provádí inicializaci hardware myši.

```
function  Initialize: Boolean; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací hodnotu True v případě úspěšné inicializace hardware myši.

Poznámky:

Metoda **Initializace** předefinovává metodu **Initialize** bázové třídy **TMouseDriver** (viz. dokumentace ke knihovně IoDrv).

Metoda provede detekci a inicializaci rezidentního ovladače myši.

4.1.1.6. Metoda TPCMouseDriver.Finalize

Metoda **Finalize** provádí deinicializaci hardware myši.

```
procedure Finalize; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu

Poznámky:

Metoda **Finalize** předefinovává metodu **Finalize** bázové třídy **TMouseDriver** (viz. dokumentace ke knihovně IoDrv).

Metoda obnoví parametry rezidentního ovladače myši, které byly nastaveny před voláním metody **Initialize**.

4.1.1.7. Metoda TPCMouseDriver.GetEvent

Metoda **GetEvent** předá nejstarší událost a odstraní ji z fronty událostí.

```
procedure GetEvent( var AEvent: TEvent ); virtual;
```

Parametry:

AEvent	Po provedení metody je do parametru AEvent uložena událost

typu evMouseXXX a jsou vyplněny položky Buttons a Pos. V případě, že ve fronta událostí ovladače myši je prázdná, pak je vyplněna pouze položka Code hodnotou evNothing.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Ovladač myši generuje následující uvedené v tabulce níže. U všech typů událostí jsou vyplněny položky Pos a Buttons struktury **TEvent**.

Událost	Popis události
evMouseDown	Stisk tlačítka myši
evMouseUp	Uvolnění tlačítka myši
evMouseMove	Pohyb ukazatele myši (tlačítka mohou být stisknuta i uvolněna)
evMouseDbl	Dvojklik tlačítka myši
evMouseRep	Automatické opakování stisku tlačítka

4.1.1.8. Metoda TPCMouseDriver.SetDblClickDelay

Metoda **SetDblClickDelay** slouží pro nastavení maximálního časového intervalu mezi dvěmi kliknutími detekovanými jako dvojklik.

Parametry:

AValue Délka časového intervalu v milisekundách.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

4.1.1.9. Metoda TPCMouseDriver.GetDblClickDelay

Metoda **GetDblClickDelay** vrací hodnotu maximálního časového intervalu mezi dvěmi kliknutími detekovanými jako dvojklik.

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací délku časového intervalu v milisekundách.

Poznámky:

4.1.1.10. Metoda TPCMouseDriver.SetDblClickArea

Metoda **SetDblClickArea** slouží k nastavení velikosti oblasti dvojkliku, tj. maximální vzdálenosti v bodech mezi pozicema dvou kliknutí, detekovanými jako dvojklik.

Parametry:

AValue Počet pixelů.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Optimální hodnota velikosti oblasti dvojkliku závisí na rozměrech bodu displeje. Obvykle se pohybuje mezi 4 až 8 body.

4.1.1.11. Metoda TPCMouseDriver.GetDblClickArea

Metoda **GetDblClickArea** vrací velikost oblasti dvojkliku, tj. maximální vzdálenosti v bodech mezi pozicema dvou kliknutí, detekovanými jako dvojklik

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací počet bodů.

Poznámky:

4.1.1.12. Metoda TPCMouseDriver.SetRepeatDelay

Metoda **SetRepeatDelay** slouží k nastavení zpoždění generování události evMouseRep při držení tlačítka myši na jednom místě po delší dobu.

Parametry:

AValue Délka časového intervalu v milisekundách.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

4.1.1.13. Metoda TPCMouseDriver.GetRepeatDelay

Metoda **GetRepeatDelay** vrací nastavené zpoždění generování události evMouseRep při držení tlačítka myši na jednom místě po delší dobu.

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací délku časového intervalu v milisekundách.

Poznámky:

4.1.1.14. Metoda TPCMouseDriver.SetRepeatRate

Metoda **SetRepeatRate** slouží k nastavení periody generování události evMouseRep při držení tlačítka myši na jednom místě po delší dobu.

Parametry:

AValue	Délka časového intervalu v milisekundách.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

4.1.1.15. Metoda TPCMouseDriver.GetRepeatRate

Metoda **GetRepeatRate** vrací nastavenou periodu generování události evMouseRep při držení tlačítka myši na jednom místě po delší dobu.

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací délku časového intervalu v milisekundách.

Poznámky:

4.1.2. Třída TStdVGACursor

Třída **TStdVGACursor** implementuje ovládání standardního kurzoru myši realizovaného rezidentním ovladačem myši v prostředí MS-DOS. Tento ovladač využívá ovladač **TVGAMonoDriver** (viz kapitola 4.1.4).

```
PStdVGACursor = ^TStdVgaCursor;
TStdVGACursor = object( TCursor )
public
  procedure Show; virtual;
  procedure Hide; virtual;
end;
```

4.1.2.1. Metoda TStdVGACursor.Show

Metoda **Show** zobrazí kurzor myši na displeji.

```
procedure Show; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda volá službu pro zobrazení kurzoru rezidentního ovladače myši v prostředí MS-DOS.

Metody **Show** a **Hide** je možné volat rekurzivně. Pro zobrazení kurzoru po n-tém volání metody **Hide** je nutné zavolat metodu Show n-krát.

4.1.2.2. Metoda TStdVGACursor.Hide

Metoda **Hide** skryje kurzor myši.

```
procedure Hide; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda volá službu pro zobrazení kurzoru rezidentního ovladače myši v prostředí MS-DOS.

Metody **Show** a **Hide** je možné volat rekurzivně. Pro zobrazení kurzoru po n-tém volání metody **Hide** je nutné zavolat metodu **Show** n-krát.

4.1.3. Třída TPCKeybDriver

Třída **TPCKeybDriver** implementuje klávesnice PC. Tato třída vychází z bázové třídy pro implementaci ovladačů klávesnic **TKeyboardDriver** (viz. dokumentace ke knihovně IoDrv)

```
PPCKeybDriver = ^TPCKeybDriver;
TPCKeybDriver = object( TKeyboardDriver )
public
  procedure GetEvent( var AEvent: TEvent ); virtual;
end;
```

4.1.3.1. Metoda TPCKeybDriver.GetEvent

Metoda **GetEvent** předá nejstarší událost a odstraní ji z fronty událostí ovladače klávesnice.

```
procedure GetEvent( var AEvent: TEvent ); virtual;
```

Parametry:

AEvent	Po provedení metody je do parametru AEvent uložena událost typu evKeyDown a jsou vyplněny položky KeyCode, CharCode. Položka VirtKey je nastavena na 0. V případě, že nebyla stisknuta žádná klávesa je vyplněna pouze položka Code hodnotou evNothing.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **GetEvent** předefinovává metodu **GetEvent** bázové třídy **TKeyboardDriver** (viz. dokumentace ke knihovně IoDrv).

Všechny kódy kláves, které ovladač **TT10KeybDriver** může vrátit jsou popsány v kapitole 3.1.1.1.

4.1.4. Třída TVGAMonoDriver

Třída **TVGAMonoDriver** implementuje ovladač VGA videokarty v monochromatickém režimu. Tato třída vychází z bázové třídy pro implementaci ovladačů displejů **TDisplayDriver** (viz. dokumentace ke knihovně IoDrv)

```
PVGAMonoDriver = ^TVGAMonoDriver;
TVGAMonoDriver = object( TDisplayDriver )
public
  OldVideoMode : Byte;

  constructor Init( AWidth, AHeight: Integer );
  function  Initialize: Boolean; virtual;
```

```
    procedure Finalize; virtual;
  end;
```

4.1.4.1. Položka TVGAMonoDriver.OldVideoMode

Položka **OldVideoMode** obsahuje číslo režimu videokarty, který je nastaven při volání metody **Initialize**, před nastavením požadovaného grafického režimu. Tato položka je určena pouze pro čtení.

```
OldVideoMode : Byte;
```

4.1.4.2. Konstruktor TVGAMonoDriver.Init

Konstruktor **Init** provádí inicializaci instance třídy.

```
constructor Init( AWidth, AHeight: Integer );
```

Parametry:

AWidth	Požadovaný počet bodů na řádku.
AHeight	Požadovaný počet bodů ve sloupci.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor **Init** provede inicializaci kreslícího povrchu a textového kurzoru.
Požadovaný počet bodů na řádku a ve sloupci daný parametry AWidth a AHeight nemá vliv na volbu grafického režimu. Vždy je nastaven režim 640x480x1. Parametry pouze omezují rozměry oblasti do které se bude kreslit.

4.1.4.3. Metoda TVGAMonoDriver.Initialize

Metoda **Initialize** provádí inicializaci ovladače displeje a VGA karty.

```
function  Initialize: Boolean; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací hodnotu True v případě úspěšné inicializace řadiče displeje.

Poznámky:

Metoda **Initializace** předefinovává metodu **Initialize** bázové třídy **TDisplayDriver** (viz. dokumentace ke knihovně IoDrv).

Metoda nastaví grafický režim videokarty 640x480x1 a vytvoří instanci kurzoru myši **TStdVGACursor** (viz kapitola 4.1.2).

4.1.4.4. Metoda TVGAMonoDriver.Finalize

Metoda **Finalize** provádí deinicializaci ovladače displeje a VGA karty.

```
procedure Finalize; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu

Poznámky:

Metoda **Finalize** předefinovává metodu **Finalize** bázové třídy **TDisplayDriver** (viz. dokumentace ke knihovně IoDrv).

Metoda vrátí původní režim videokarty, který byl nastaven před voláním metody **Initialize**.

4.1.5. Třída TVESADriver

Třída **TVESADriver** implementuje ovladač SVGA videokarty v režimu s 256 barvami na pixel. Tato třída vychází z bázové třídy pro implementaci ovladačů displejů **TDisplayDriver** (viz. dokumentace ke knihovně IoDrv)

```
PVESADriver = ^TVESADriver;
TVESADriver = object( TDisplayDriver )
public
  OldVideoMode : Word;

  constructor Init( AWidth, AHeight: Integer );
  function  Initialize: Boolean; virtual;
  procedure Finalize; virtual;
end;
```

4.1.5.1. Položka TVESADriver.OldVideoMode

Položka **OldVideoMode** obsahuje číslo režimu videokarty, který je nastaven při volání metody **Initialize**, před nastavením požadovaného grafického režimu. Tato položka je určena pouze pro čtení.

```
OldVideoMode : Word;
```

4.1.5.2. Konstruktor TVESADriver.Init

Konstruktor **Init** provádí inicializaci instance třídy.

```
constructor Init( AWidth, AHeight: Integer );
```

Parametry:

AWidth	Požadovaný počet bodů na řádku.
AHeight	Požadovaný počet bodů ve sloupci.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor **Init** pouze nastaví implicitní parametry ovladače displeje a uloží požadovanou počet pixelů grafického režimu. Parametry AWidth a AHeight je možné nastavit podle následující tabulky:

AWidth	AHeight
320	240
640	480
800	600
1024	768

Inicializace požadovaného režimu se provede až při volání metody **Initialize**. Některé grafické karty nemusí podporovat všechny uvedené režimy. Spolehlivě je podporován obvykle režim 640x480. Pokud požadovaný režim není podporován metoda **Initialize** vrátí hodnotu False (a knihovna Controls vyvolá runtimeovou chybu 241).

4.1.5.3. Metoda TVESADriver.Initialize

Metoda **Initialize** provádí inicializaci ovladače displeje a SVGA karty.

```
function  Initialize: Boolean; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací hodnotu True v případě úspěšné inicializace řadiče displeje.

Poznámky:

Metoda **Initializace** předefinovává metodu **Initialize** bázové třídy **TDisplayDriver** (viz. dokumentace ke knihovně IoDrv).

Metoda uloží aktuální režim videokarty do položky OldVideoMode a pokusí se inicializovat požadovaný grafický režim, tj. režim s rozlišením daným parametry

konstruktoru s 8bit barevnou hloubkou. Pokud vše proběhne v pořádku vytvoří instanci grafického povrchu (třída **T8BPPDrawSurface**), textového kurzoru (třída **TGraphicCaret**) a kurzoru myší (**T8BPPCursor**), viz. dokumentace ke knihovně IoDrv.

Barevná paleta je nastavena podle palety v knihovně StdPal8 (viz. dokumentace ke knihovně StdPal8).

4.1.5.4. Metoda TVESADriver.Finalize

Metoda **Finalize** provádí deinicializaci ovladače displeje a SVGA karty.

```
procedure Finalize; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu

Poznámky:

Metoda **Finalize** předefinovává metodu **Finalize** bázové třídy **TDisplayDriver** (viz dokumentace ke knihovně IoDrv).

Metoda vrátí původní režim videokarty, který byl nastaven před voláním metody **Initialize**.

4.1.6. Třída TT51VESADriver

Třída **TT51VESADriver** implementuje ovladač SVGA videokarty v režimu s 256 barvami na pixel. Tato třída vychází z třídy TVESADriver, rozšiřuje ji o metody pro řízení jasu displeje řídící jednotky Touch 51.

```
PT51VESADriver = ^TT51VESADriver;
TT51VESADriver = object( TVESADriver )
public
  function  GetBrightness:Integer; virtual;
  procedure SetBrightness(Avalue:Integer); virtual;
end;
```

4.1.6.1. Funkce TT51VESADriver. GetBrightness

Metoda **GetBrightness** vrací aktuální hodnotu nastavení jasu displaje Touch51.

```
function  GetBrightness:Integer; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Hodnota jasu v rozsahu 0 (nejnižší jas) až 255 (nejvyšší jas).

Poznámky:

4.1.6.2. Funkce TT51VESADriver. SetBrightness

Metoda **SetBrightness** nastavuje aktuální jas displaje Touch51.

```
procedure SetBrightness(Avalue:Integer); virtual;
```

Parametry:

Avalue	Hodnota jasu v rozsahu 0 (nejnižší jas) až 255 (nejvyšší jas).

Návratové hodnoty:

Metoda nevrací žádnou hodnotu

Poznámky: