T9Drv

OVLADAČE TERMINÁLU TERM 09 A TERM 90 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 1.00

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: 15.05.2004
Datum posledního uložení dokumentu: 18.05.2004
(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	Bin	15.05.2004	První vydání

1.2. Účel dokumentu

Tento dokument slouží jako popis knihovny T10Drv, která je součástí balíku vizualizačních knihoven pro jednotku 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 knihovny T9Drv

Knihovna T9Drv obsahuje ovladače klávesnice a správce komunikačního kanálu pro terminály Term 90 a Term 09 a ovladač znakového displeje terminálu Term 09 fy. SofCon pro použití s vizualizačními knihovnami pro jednotku KIT.

3.2. Terminál Term 90 a Term 09

Terminál Term 90 je terminál s barevným TFT grafickým displejem s rozlišením 640x480 pixelů. Terminál Term 09 je terminál se znakovým displejem 4x20 znaků. Terminály jsou dále vybaveny membránovou klávesnicí.

3.2.1. Ovladač klávesnice

Knihovna T9Drv obsahuje třídu **TT9KeybDriver**, která implementuje ovladač klávesnice terminálů Term 90 a Term 09. 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.

Ovladač umožňuje zároveň řízení zvukové signalizace kláves terminálů pomocí metod **SetKeyBeep** a **GetKeyBeep**.

3.2.1.1. Podporované kódy kláves

Ovladač klávesnice **T9KeybDriver** podporuje kódy speciálních kláves uvedené v následující tabulce. Jednotlivé konstanty s prefixem kb_ jsou popsány v dokumentaci ke knihovně IoDrv.

kbF1	kbAltF1	kbEnter	1
kbF2	kbAltF2	kbEsc	2
kbF3	kbAltF3	kbDel	3
kbF4	kbAltF4	kbNumLock	4
kbF5	kbAltF5	kbLeft	5
kbF6	kbAltF6	kbRight	6
kbF7	kbAltF7	kbUp	7
kbF8	kbAltF8	kbDown	8
kbF9	kbAltF9	‘+’	9
kbF10	kbAltF10	‘-‘	0
		‘*‘	‘.’

3.2.1.2. Mobilová klávesnice

Abychom umožnili vkládání písmen pomocí této klávesnice, byl vyvinut ovladač, který se chová jako klávesnice na mobilních telefonech.

3.2.2. Ovladač displeje terminálu Term 09

Knihovna T9Drv obsahuje třídu **TT9DispDriver**, která implementuje ovladač znakového displeje terminálu. Ovladač vychází z abstraktní třídy **TCachedDisplayDriver** implementované v knihovně IoDrv.

3.2.3. Použití ovladačů terminálů

3.2.3.1. Terminál Term 09

Následující příklad ukazuje, jak vytvořit základní komponentu aplikace **TApplication** pro terminál Term 09.

```
var
  App : PApplication;
...
begin
...

  App :=
    New( PApplication, Init (
      New( PInputDriver, Init(
        New( PT9KeybDriver, Init ),
        nil
      )),
      New( PT9DispDriver, Init ),
      @g_AppSettings
    ));
...
```

Proměnná g_AppSettings obsahuje nastavení ovladače displeje a klávesnice terminálu. Tato proměnná by měla být umístěna v non-volatilní paměti.

3.2.3.2. Terminál Term 90

Následující příklad ukazuje, jak vytvořit základní komponentu aplikace **TApplication** pro terminál Term 90.

```
var
  App : PApplication;
...
begin
...

  App :=
    New( PApplication, Init (
      New( PInputDriver, Init(
        New( PT9KeybDriver, Init ),
        nil
      )),
      New( PVESADriver, Init( 640, 480 ) ),
      @g_AppSettings
    ));
...
```

3.2.3.3. Typy klávesnic

V knihovně jsou předdefinovány tři typy klávesnic. Rozdíl je pouze ve jméně konstruktoru. Máme-li zájem použít pouze čísla, použijeme konstruktor Init. Chceme-li použít i písmena, použijeme konstruktor InitMobil. Pro terminál T90 přichází v úvahu i varianta s konstruktorem InitMobilCZ, která nabízí i znaky s českou diakritikou.
Pokud máte zájem o jinou definiční množinu znaků na klapkách, podívejte se na referenční kapitolu k objektu TT9KeybDriver.

3.2.4. Správce komunikačního kanálu

Pro nastavování komunikačních parametrů je určen správce komunikačního kanálu, který umožňuje nižší úroveň řízení.

4. Reference

4.1. Konstanty

Vzhledem k přítomnosti vstupního zařízení – čipového klíče – jsou zde definovány dvě konstanty a to:

```
const
  evDallas = $1000;   { udalost pritomnosti cipoveho klice }
  nmDallas = $0100;   { oznameni cipoveho klice }
```

Konstanta evDallas je dále registrována jako ohnisková událost (focusedEvent), tedy její šíření je shodné s ostatními ohniskovými událostmi, jako např. klávesnice.

4.2. Typy

4.2.1. Typ TDallasParam

Tento typ je určen k přenášení identifikace čipového klíče dallas. Je deklarován jako:

```
TDallasParam = array[0..7] of byte;
```

4.2.2. Typy TREQuest a TSREQ

TREQuest je výčtový typ jednotlivých požadavků na spodní desku terminálu (s procesorem 89c2052). Pro uchování množiny aktuálních příznaků je deklarován typ TSREQ, který je množinovým typem z TREQuest.

```
{ jednotlive priznaky pozadavku na spodni desku terminalu }
TREQuest = ( REQ_readDallas,
             REQ_STOP,
             REQ_TEST,
             REQ_swVER,
             REQ_SaveSetup,
             REQ_T6Status,
             REQ_readError,
             REQ_changeComm,
             REQ_writeBackLight,
             REQ_writeMisc,
             REQ_changeDNode
           );
{ typ mnoziny priznaku pozadavku na spodni desku terminalu }
TSREQ = set of TREQuest;
```

4.3. Třídy

4.3.1. Třída TT9KeybDriver

Třída **TT9KeybDriver** implementuje ovladač klávesnice terminálu Term 09 / 90. Tato třída vychází z bázové třídy pro implementaci ovladačů klávesnic **TKeyboardDriver** (viz. dokumentace ke knihovně IoDrv)

```
{------------------------------------------------------------------}
{ TT9KeybDriver                                                    }
{------------------------------------------------------------------}
{ Ovladace klavesnice terminalu Term09 a Term90 }

PT9KeybDriver = ^TT9KeybDriver;
TT9KeybDriver = object( TKeyboardDriver )
public
  { true = cisla }
  NumLockState : boolean;
  { false = velka }
  LowAlphaState : boolean;

  { znaky = znaky.up['1'], znaky.down['.'] }
  znaky : record
            up,dn : array ['.'..'9'] of TString32;
          end;

  { konstruktor tridy - obycejne klavesnice }
  constructor Init;
  { konstruktor tridy - mobiloidni klavesnice }
  constructor InitMobil;
  { konstruktor tridy - mobiloidni klavesnice ceska verze }
  constructor InitMobilCz;

  { Inicializace zarizeni, volat ihned po vytvoreni instace }
```

```
    function  Initialize: Boolean; virtual;
    { Uvoleni alokovanych hw a sw prostredku zarizeni, volat
    { pred zrusenim instance }
    procedure Finalize; virtual;
    { Tik ovladace klavesnice }
    procedure Tick; virtual;
    { Metoda pro ziskani posledni vstupni udalosti. Volani metody
    { zpusobi vyjmuti udalosti z fronty  }
    procedure GetEvent( var Event: TEvent ); virtual;

    { Zahozeni vsech hodnot z kruhove fronty radice }
    procedure Flush;

    { Pipnuti pri stisku klavesy, True = pipnout }
    procedure SetKeyBeep( AValue: Boolean ); virtual;
    function  GetKeyBeep: Boolean; virtual;

    { definovani znaku na klapkach }
    procedure SetKeyPadString(const a : char; const sUP,sDN :
TString32 );

  private
    { Vraci kod nejpozdeji stisknute klavesy }
    function GetKey: Byte;
  end;
```

4.3.1.1. Položka TT9KeybDriver.NumLockState

Položka **NumLockState** obsahuje stav klávesnice. Pokud má hodnotu **true**, chová se klávesnice jako pouze klávesnice číselná. Má-li hodnotu **false**, potom je možné použít tuto klávesnici k vkládání dalších znaků (jako například písmen abecedy atp.) . Položka je inicializovaná metodou initialize a určena pouze pro čtení i pro zápis.

```
NumLockState : Boolean;
```

4.3.1.2. Položka TT9KeybDriver.LowAlphaState

Položka **LowAlphaState** indikuje velikost písmen při zapnuté mobilové klávesnici. Hodnota true znamená malá písmena, hodnota false znamená velká písmena.

```
LowAlphaState : Boolean;
```

4.3.1.3. Položka TT9KeybDriver.Znaky

Jedná se o záznam obsahující jednotlivé znaky kláves mobilové klávesnice. Je nastavován konstruktorem a případně modifikován níže uvedenou metodou TT9KeybDriver.SetKeyPadString.

4.3.1.4. Konstruktor TT9KeybDriver.Init

Konstruktor **Init** provádí inicializaci instance třídy. Klávesy jsou pouze numerické bez ohledu na stav NumLock.

```
constructor Init;
```

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

4.3.1.5. Konstruktor TT9KeybDriver.InitMobil

Konstruktor **InitMobil** provádí inicializaci instance třídy. Klávesy jsou numerické a jsou k dispozici standardní písmena abecedy (velká i malá). Přepínání se provádí klávesou NumLock v cyklickém pořadí -> čísla -> VELKÁ PÍSMENA -> čísla -> malá písmena -> .

```
constructor InitMobil;
```

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

4.3.1.6. Konstruktor TT9KeybDriver.InitMobilCZ

Konstruktor **InitMobilCZ** provádí inicializaci instance třídy. Klávesy jsou numerické a jsou k dispozici standardní písmena abecedy (velká i malá) včetně písmen s českou diakritikou. Přepínání se provádí klávesou NumLock v cyklickém pořadí -> čísla -> VELKÁ PÍSMENA -> čísla -> malá písmena -> .

```
constructor InitMobilCZ;
```

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

4.3.1.7. Metoda TT9KeybDriver.Initialize

Metoda **Initialize** provádí inicializaci hardware řadiče klávesnice terminálu Term 09 či 90.

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

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací hodnotu True v případě úspěšné inicializace řadiče klávesnice.

Poznámky:

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

Metoda vyprázdní buffer klávesnice a nastaví režim NumLock a režim malých písmen

(LowAlphaState).

4.3.1.8. Metoda TT9KeybDriver.Finalize

Metoda Finalize je určena k regulérnímu ukončení práce s klávesnicí. Je vhodné ji vyvolávat před zrušením instance.

```
procedure Finalize; virtual;
```

4.3.1.9. Metoda TT9KeybDriver.Tick

Metoda **Tick** provádí krok automatu klávesnice implementovaného ovladačem klávesnice.

```
procedure Tick; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

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

Metoda provádí periodický přepis položky NumLockState příslušné spodní vrstvy a tak správně signalizuje stav LED diody NumLock. Kromě toho také vyvolává metodu Tick od správce komunikačního kanálu.

4.3.1.10. Metoda TT9KeybDriver.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).

Kódy kláves, které ovladač **TT9KeybDriver** může vrátit jsou popsány v kapitole 3.2.1.1. Kromě těchto „nativních" znaků může ovladač vracet klávesy definované pro mobilovou klávesnici.

4.3.1.11. Metoda TT9KeybDriver.Flush

Metoda **Flush** provádí vyprázdnění bufferu řadiče klávesnice.

```
procedure Flush;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

4.3.1.12. Metoda TT9KeybDriver.SetKeyBeep

Metoda **SetKeyBeep** zapíná resp. vypíná zvukovou signalizaci při stisku klávesy terminálu.

```
procedure SetKeyBeep( AValue: Boolean ); virtual;
```

Parametry:

AValue	Pokud je parametr AValue nastaven na hodnotu True, zvuková signalizace bude zapnuta. V opačném případě bude vypnuta.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

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

4.3.1.13. Metoda T9KeybDriver.GetKeyBeep

Metoda **GetKeyBeep** vrací aktuální stav zvukové signalizace při stisku klávesy

terminálu.

```
function  GetKeyBeep: Boolean; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací hodnotu True, pokud je zvuková signalizace při stisku klávesy terminálu zapnuta. V opačném případě vrací hodnotu False.

Poznámky:

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

4.3.1.14. Metoda T9KeybDriver.SetKeyPadString

Tato metoda provádí definování znaků mobilové klávesnice.

```
    procedure SetKeyPadString(const a : char; const sUP,sDN :
TString32 );
```

Parametry:

a	- označuje klapku, o kterou se jedná (jedna z ['.','0'..'9'])
sUP	- definuje jednotlivé znaky klapky pro variantu velkých písmen
sDN	- definuje jednotlivé znaky klapky pro variantu malých písmen

Příklad:

```
SetKeyPadString( '2', 'ABC2ÁČ', 'abc2áč' );
```

4.3.1.15. Metoda T9KeybDriver.GetKey

Jedná se o vnitřní metodu vybírající z nižší vrstvy nejstarší klávesu. Metoda je interní a aplikačně se nemá a ani nemůže vyvolávat.

```
    function GetKey: Byte;
```

4.3.2. Třída TT9DispDriver

Třída **TT9DispDriver** implementuje ovladač displeje terminálu Term09. Tato třída vychází z bázové třídy pro implementaci ovladačů displejů s vyrovnávací pamětí **TCachedDisplayDriver** (viz. dokumentace ke knihovně IoDrv)

```
PT9DispDriver = ^TT9DispDriver;
TT9DispDriver = object( TCachedDisplayDriver )
                  Cache : pointer;     { Kopie videopameti }
```

```
                public

                  constructor Init;

                  destructor Done; virtual;

                  { HW inicializace displeje }
                  function  Initialize: Boolean; virtual;

                  { HW deinicializace displeje }
                  procedure Finalize; virtual;

                  { Tik ovladace displeje }
                  procedure Tick; virtual;

                  procedure EndDraw; virtual;

                private

                  { Metoda pro prekopirovani casti obsahu Surface
na displej }
                  procedure Refresh( const R: TRect ); virtual;
                end;
```

4.3.2.1. Položka TT9DispDriver.Cache

Položka **Cache** obsahuje ukazatel na vyrovnávací paměť pro videodata zapisovaná do řadiče displeje alokovanou na hromadě. Tato položka je inicializovaná konstruktorem instance a je určena pouze pro čtení.

```
Cache      : Pointer;
```

4.3.2.2. Konstruktor TT9DispDriver.Init

Konstruktor **Init** provádí inicializaci instance třídy.

```
constructor Init;
```

Poznámky:

Konstruktor alokuje vyrovnávací paměť pro videodata zapisovaná do řadiče displeje. Vytvoří instanci textového kurzoru a kreslícího povrchu (viz. dokumentace ke knihovně IoDrv).

4.3.2.3. Destruktor TT9DispDriver.Done

Destruktor **Done** provádí uvolnění prostředků alokovaných konstruktorem.

```
destructor Done; virtual;
```

Parametry:

Destruktor nemá žádné parametry.

Návratové hodnoty:

Destruktor nevrací žádnou hodnotu.

Poznámky:

Destruktor uvolní z paměti vyrovnávací paměť pro videodata a vytvořené instance textového kurzoru a kreslícího povrchu.

4.3.2.4. Metoda TT9DispDriver.Initialize

Metoda **Initialize** provádí inicializaci hardware řadiče displeje terminálu Term09.

```
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 provede inicializaci řadiče displeje a do paměti řadiče displeje přepíše aktuální stav vyrovnávací pamětí (mezery = prázdné znaky).

4.3.2.5. Metoda TT9DispDriver.Finalize

Metoda **Finalize** provádí deinicializaci hardware řadiče displeje terminálu Term09.

```
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).

4.3.2.6. Metoda TT9DispDriver.Tick

Metoda **Tick** provádí jeden krok automatu ovladače displeje.

```
procedure Tick; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **Tick** volá metodu **Tick** svého pra-předka, tj. třídy **TDisplayDriver** (viz. dokumentace ke knihovně IoDrv) a navíc přidává kód pro periodický přepis řádků na displej (refresh). Dále tiká správcem komunikačního kanálu.

4.3.2.7. Metoda TT9DispDriver.Refresh

Metoda **Refresh** slouží k překopírování části kopie videopaměti do paměti řadiče displeje.

```
procedure Refresh( const R: TRect ); virtual;
```

Parametry:

R	Obdélník ohraničující oblast videopaměti, kterou je potřeba přenést do paměti řadiče displeje.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **Refresh** předefinovává abstraktní metodu **Refresh** předka, tj. třídy **TCachedDisplayDriver** (viz. dokumentace ke knihovně IoDrv). Z důvodu vysoké komunikační režie je přesouván 1 nebo 2 celé řádky v jedné zprávě.

4.3.2.8. Metoda TT9DispDriver.EndDraw;

Pro správnou práci terminálu Term09 bylo potřeba tuto metodu předefinovat.

4.3.3. Třída TT9CommDriver

Třída **TT9CommDriver** implementuje správce komunikačního kanálu pro desku terminálu Term 09 a 90 (s procesorem 89c2052). Z deklarace je zde pouze uvedena část zajímavá pro uživatele

```
  PT9CommDriver = ^TT9CommDriver;
  TT9CommDriver = object( TObject )
                  private
```

```
                    ...
                    { komunikacni kanal }
                    Chn : PChnVirt;
                    { Prijata zprava a jeji delka }
                    RMess : array [0..199] of byte;
                    LRMess : word;
                    { Odesilana zprava a jeji delka }
                    SMess : array [0..199] of byte;
                    LSMess : word;
                    ...
                    {PRT nastavitelne param}
                    { NODE, DNODE }
                  public
                    NOD, DNO : byte;
                    {COM nastavitelne param}
                    COM, IRQ, STO : byte;
                    BD  : word;
                    PAR : char; {'N','E','O'}

                    { Fronta klaves }
                    KeybQueue : array[0..15] of byte;
                    KeybFirst, KeybLast : byte;

                    { obraz bajtu MISC - WD|NL|PIP|REV|xxxx}
                    Misc : byte;

                  public
                    { vnitrni stav automatu ticku , 0 = normalni,
jinak reconnecting }
                    State : byte;
                    { pozadavky na vyslani zprav }
                    REQ : TSREQ;

                    { 8 byte identifikator Dallasu }
                    DallasParam : TDallasParam;

                    { inicializace }
                    procedure Initialize;
                    { finalizace }
                    procedure Finalize;
                    { vyzvedava klv. udalosti }
                    procedure GetKeyCode( var ACode: Byte );
                    { prijima zpravu pro display }
                    procedure DisplayMess( ptr:pointer; len : byte );
                    { tikaci procedura }
                    procedure Tick;
                    { vraci status komunikace se spodni deskou }
                    function  GetStatus : byte;
                    { manualni zadost o reinicializaci kanalu }
                    procedure ReInit;
                    { zapis vystupu - vsech 4 bran }
                    procedure WriteOut( const a, b, c, d : byte );
                    { zapis LED diod klaves F1-F8 }
                    procedure WriteFxLED( const A : byte );
                    { zapis podsvetleni trvanlivych tlacitek }
                    procedure WriteBtConfirm( const A : byte );
                    { zapis dig. vystupu }
                    procedure WriteDigOut( const A : byte );
                    { zapis na vystupni P-BUS }
                    procedure WritePBusOut( const A : byte );
                    { cteni vstupu }
                    function  ReadIn : word;
                    { cteni dig. vstupu }
                    function  ReadDigIn : byte;
                    { cteni vstupniho P-BUSu}
```

```
                    function  ReadPBusIN : byte;
                    { podsvetleni }
                    procedure BackLight( const ATime : byte; const
AValue: byte );
                    { num-lock }
                    procedure NumLock( const AEnable : boolean );
                    { watch-dog }
                    procedure WatchDog( const AEnable : boolean );
                    { pipani - zajistuje si klavesnice }
                    procedure SetKeyBeep( const AEnable: boolean );
                    { staly rev - xxx }
                    procedure Noise( const AEnable : boolean );
                    { parametry PRT }
                    { nastav cislo node }
                    function  SetNode( const ANode : byte ): boolean;
                    { nastav cislo dnode }
                    function  SetDNode( const ADNode : byte ):
boolean;
                    { parametry COM }
                    { cislo COM portu mastera (PC/Kita) }
                    function  SetCOM( const ACOM : byte ): boolean;
                    { cislo IRQ }
                    function  SetIRQ( const AIRQ : byte ): boolean;
                    { pocet stop-bitu }
                    function  SetSTO( const ASTO : byte ): boolean;
                    { baud rate }
                    function  SetBD( const ABD : word ): boolean;
                    { parita }
                    function  SetPAR( const APAR: char ): boolean;

                    procedure DoChangeComm;
                  end;
```

4.3.3.1. Inicializace, průběh činnosti a ukončení

K základním metodám této třídy patří:

```
  procedure Initialize;
  procedure Finalize;
  procedure Tick;
  function  GetStatus : byte;
```

Po spuštění programu je standardně komunikační kanál uzavřen. Otvírá se až registrováním prvního vyššího komunikačního objektu (klávesnice, displej, aplikace, …). Registrace se provádí metodou Initialize. Komunikační kanál se standardně zavírá odregistrováním posledního vyššího komunikačního objektu. Odregistrování provádí metoda Finalize.
Pro správnou funkci je třeba opakovaně vyvolávat metodu Tick (není potřeba vyvolávat s nějakou pravidelnou přesností).
Zjištění stavu spodní desky provádí metoda GetStatus.

4.3.3.2. Rozhraní pro klávesnici

Rozhraní pro klávesnici je tvořeno třemi metodami.

```
  procedure GetKeyCode( var ACode: Byte );
  procedure NumLock( const AEnable : boolean );
  procedure SetKeyBeep( const AEnable: boolean );
```

První z nich je metoda GetKeyCode. V parametru navrací kód klávesy. Pokud není žádná klávesa k dispozici, vrací hodnotu 0.

Druhá, jak název napovídá, určuje stav LED diody na tlačítku NumLock. Hodnota **true** znamená svícení.
Poslední z této kategorie je metoda SetKeyBeep. Tato určuje, zdali mají tlačítka vydávat po stisku zvukovou ozvěnu. Hodnota **true** zapíná a hodnota **false** vypíná tuto odezvu.

4.3.3.3. Rozhraní pro displej

Pro displej je určena pouze jedna metoda. Jedná se o metodu DisplayMess.

```
procedure TT9CommDriver.DisplayMess( ptr : pointer; len : byte );
```

Parametry:

ptr = ukazatel na zprávu
len = počet byte zprávy

Poznámky:
Metoda si ve vlastní režii okopíruje zprávu.

4.3.3.4. Aplikační rozhraní

Aplikační rozhraní je tvořeno skupinou metod provádějící zbývající akce příslušející právě tomuto terminálu. Jedná se o tyto metody:

```
procedure WriteOut( const a, b, c, d : byte );
procedure WriteFxLED( const A : byte );
procedure WriteBtConfirm( const A : byte );
procedure WriteDigOut( const A : byte );
procedure WritePBusOut( const A : byte );

function  ReadIn : word;
function  ReadDigIn : byte;
function  ReadPBusIN : byte;

procedure Noise( const AEnable : boolean );

procedure BackLight( const ATime : byte; const AValue: byte );

procedure WatchDog( const AEnable : boolean );
```

Metoda WriteOut zapisuje všechny výstupy. Spíše než tuto metodu je vhodné vyvolávat metody pro jednotivé výstupy, což jsou WriteFxLED (podsvětlení diod kláves F1-F8), WriteBtConfirm (podsvětlení tlačítek s vysokou životností Alt-F1 – Alt-F8). Zápis hodnoty na digitální 24V výstupy se provádí metodou WriteDigOut a nakonec výstupní brána P-BUSu je ovládána metodou WritePBusOut.

Metoda ReadIn čte všechny dig. vstupy. Spíše než tuto metodu je vhodné vyvolávat metodu ReadIn pro čtení stavu digitálních 24V vstupů a metodu ReadPBusIN pro čtení vstupní brány P-BUSu.

Na terminálu je možné trvale zapnout zvukovou signalizaci (vhodné pro případ havárie/…). To se provádí metodou Noise, kde vstupní parametr **true** určuje výstražný nepřetržitý zvuk, zatímco **false** tento nepřetržitý zvuk vypíná.

Pro nastavení intenzity podsvětlení displeje a doby vypnutí podsvětlení je určena metoda BackLight. Prvním parametrem je číslo určující dobu vypnutí a druhým je číslo určující intenzitu podsvětlení.

WatchDog- zatím není implementován.

Do aplikačního rozhraní taktéž patří čtečka čipových klíčů Dallas. Ta byla integrována do knihoven tak, že při příchodu čipového klíče je vyvolána (ohnisková) událost evDallas. Zpracování této události je na programátorovi aplikace. Nutno dodat, že platný čipový klíč je v proměnné DallasParam a pro vhodné reagování na tuto událost byla vytvořena procedura DallasHandle (viz 4.5.1).

4.3.3.5. Rozhraní pro nastavování komunikačního kanálu

Knihovny standardně navazují komunikaci se spodní deskou na rychlosti 28 800, bez parity s jedním stop bitem. Číslo node mastera je 1, číslo node slave je 31. Je využit standardní COM port 2 (tj. IRQ3)

Změny jednotlivých parametrů se různě promítají do skutečného stavu.
Metoda SetNode nastavuje číslo NODE mastera. Slave je stavěn tak, aby odpovídal tomu masterovi, který mu zprávu poslal, tudíž tato změna má vliv pouze na změnu proměnné Node posílané po komunikačním kanálu.

Metoda SetCOM nastavuje číslo COM portu. Povolené hodnoty jsou 1 až 8.
Metoda SetIRQ nastavuje číslo přerušení. Povolené hodnoty jsou 0 až 7.
Metoda SetSTO nastavuje počet stop bitů. Je možné nastavit 1 nebo 2 stop bity.
Metoda SetBD nastavuje komunikační rychlost. Jako parametr dostává požadovanou rychlost a na tu nastavuje komunikaci. Povolené hodnoty jsou: (57600, 28800, 19200, 14400, 11520, 9600, 4800, 2400).
Metoda SetPAR nastavuje paritu. Je možné komunikovat s žádnou, sudou či lichou paritou. (znakový parametr 'N', 'E', 'O')
Upozornění: neprázdná parita (tedy sudá, či lichá) se vzájemně vylučuje s dvěma stop bity. Máme-li zájem komunikovat s paritou, je třeba nastavit pouze jeden stop bit. (Důvodem je devítibitová komunikace ze strany spodní desky.)

Metoda SetDNode nastavuje číslo node spodní desky. Nastavením tohoto DNode dojde i k zápisu celého setupu spodní desky do EEPROM paměti.

Po nastavení všech požadovaných parametrů metodami SetXxx je třeba zavolat metodu DoChangeComm. Tato se postará o provedení změn. Nutno dodat, že změny nejsou provedeny ihned. Dále pak i pokud změníme komunikační rychlost a automaty se na této nedomluví, vracejí se na předchozí chodivé parametry.

4.3.3.6. Rozhraní pro low-level komunikaci s terminálovou deskou

Na nejnižší úrovni lze terminálové desce zasílat jednotlivé příkazy. Z hlediska programátora se jedná o tyto metody:

```
{ zaslani signalu stop }
procedure Stop52;
```

```
{ vycteni verze sw }
procedure SwVer52;
{ cteni statusu ala Term6 }
procedure T6Status;
{ zapis setupu do EEPROM 52 }
procedure SaveSetup52;
```

Metoda TT9CommDriver.Stop52 zasílá desce terminálu příkaz STOP, kterým si vynutí reinicializaci (reset) spodní desky.
Metoda TT9CommDriver.SwVer52 zasílá desce terminálu žádost o verzi software. Terminálová deska odpoví naplněním položek swVerByte, builtByte a wdtoutByte.
Metoda TT9CommDriver.T6Status zasílá požadavek na status terminálu kompatibilní s terminálem T6. Terminálová deska odpoví naplněním položek konf1, konf3 a fwver.
Metoda TT9CommDriver.SaveSetup52 zašle desce terminálu žádost o uložení aktuálního nastavení (konfigurace) do paměti EEPROM.

4.4. Proměnné

4.4.1. T9CommDriver

V knihovně je deklarována/definována právě jedna proměnná T9CommDriver typu TT9CommDriver. Jedná se o statickou proměnnou v datovém segmentu. Inicializační hodnoty jsou nastaveny použitím této jednotky (klauzulí **uses** T9Drv; v aplikaci), komunikační kanál je otevřen první registrací (voláním metody initialize).

4.5. Procedury

4.5.1. Procedura DallasHandle

Tato procedura je určena ke spolupráci s čtečkou čipových klíčů Dallas. Transformuje událost evDallas na oznámení nmDallas. Předpokládá se použití v procedurálních proměnných beforeHandle resp. afterHandle. Pokud chceme reagovat na čipové klíče kdykoliv, je vhodné tuto proceduru dát do proměnné `controls.application^.beforeHandle`. Máme-li zájem reagovat pouze na některých stránkách, je možné tuto proceduru přiřadit odpovídajícím stránkám.