uTermT05

TERMINÁL TERM05

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: 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

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	We	16.05.2003	První vydání

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky .

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro lepší pochopení tohoto dokumentu je vhodné se seznámit se s manuálem uATerm, uTermChr, uCharBuf, ChnVirt a uString.

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

Terminál TERM05 je malý průmyslový terminál s membránovou klávesnicí a alfanumerickým podsvíceným LCD displejem se dvěmi řádky o šestnácti znacích. Vzhledově je terminál identický s terminálem Term06, ale není samostatným celkem. Terminál je nedílnou součástí řídícího systému. Display a klávesnice je k procesorové desce KitV40 nebo Kit386 připojen přes PBus. Program pro obsluhu terminálu je součástí celkového řídícího programu. Terminál snímá znaky z klávesnice a ukládá je do fronty přijatých znaků. Text ze zobrazovacího bufferu, se zobrazí na displeji. Text je většinou vytvářen pomocí nařízeného systému menu v řídicím programu.

Tato jednotka implementuje objektové typy pro práci s terminálem TERM05. Je určena pro použití v software řídícího systému, kterého součástí je terminál TERM05.

V tomto dokumentu nejsou popsány proměnné a metody zděděných objektových typů. Jejich popis lze nalézt v dokumentaci jednotek **uATerm, uTermChr** a **uCharBuf.**

4. Popis konstant a typů

```
const
  cVerNo = např. $0102; { BCD formát }
  cVer   = např. '01.02,28.04.2003';
```

Číslo verze jednotky v BCD tvaru a v textové podobě včetně datumu změny.

```
const
  Primar     =1;    {Primární znaky z klávesnice po krátkém stisku}
  Sekundar   =2;    {Sekundární znaky z klávesnice po dlouhém stisku}
  zPlusPlus  =#$AB; {znak '+' s nahozeným nejvyšším bitem}
  zMinusMinus=#$AD; {znak '-' s nahozenym nejvyšším bitem}
  kNone  = #$00;    {žádná známá klávesa}

  Res_Ok          = $00; { bez chyby }
  Res_SetParamErr = $01; { chyba při zadávání parametrů }
```

Konstanty chyb výsledek KResult pro inicializaci klávesnice a výsledek operací nad sériovým kanálem terminálu, pokud je inicializován.

```
type
  tAChar   = array[0..$fff0] of Char;
  pAChar   = ^tAChar;
```

5. Objektové typy

5.1. tADispT05

```
pADispT05 = ^tADispT05;
tADispT05 = object(tADisp)
    constructor Init(TermOwner:pATerm;CharColls,CharRows:byte);
    procedure mFLLight (b:byte); virtual;
    procedure DispClrScr;
end;
```

Mezi hlavní objekt displeje **tDispT05** a objekt abstraktního displeje **tADisp** z jednotky uATerm je zařazený objekt **tADispT05**. Tento objektový typ implementuje metody **mFlLight** a **DispClrScr**, které volá nadřízený objekt terminálu a které objekt abstraktní ho displeje **tADisp** nemá. Objekt **tADispT05** je vhodný jako rodičovský objekt pro simulátory displeje v simulátorech terminálu. Tyto simulátory fungují například na monitoru PC a chovají se jako reálný terminál Term05. Používají se zejména na ladění programu.

5.1.1. Init

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

TermOwner je ukazatel na nadřízený objekt terminálu. **CharColls** je počet sloupců displeje. **CharRows** je počet řádek displeje.

5.1.2. mFlLight

```
procedure mFLLight  (b:byte); virtual;
```

Metoda nastavuje v nařízeném terminálu proměnnou **StLight**, která charakterizuje úroveň přisvětlení displeje. Proměnná se zadává v intervalu <0;3>. U reálného displeje B=0 způsobí zhasnutí přisvětlení. Zbytek metody je prázdný, poněvadž tento objekt ještě nepracuje s reálným displejem.

5.1.3. DispClrScr

```
procedure DispClrScr;
```

Metoda je prázdná a slouží pouze tomu, aby ji dědicové v simulátorech měli definovanou a mohla v nich být volaná.

5.2. tDispT05

```
type
  pDispT05 = ^tDispT05;
  tDispT05 = object(tADispT05)
    vAddr :word;
    constructor Init(TermOwner:pATerm; CharColls, CharRows:byte;
                     Adresa:word; EnIniHWProc:Boolean);
    procedure InitDisplay;                virtual;

    destructor  Done;                     virtual;
    procedure DoneHWProc;                 virtual;
    procedure mFLLight  (b:byte);         virtual;
    procedure DispClrScr;
    procedure DTickRefreshScr;            virtual;
    procedure MoveDispCur(X,Y:byte);      virtual;
    procedure WriteStr(const S:string);   virtual;
   private
    function  GetDispReady : Boolean;     virtual;
    procedure WriteCmd(Data:byte);        virtual;
    procedure WriteChar(Zn:char);         virtual;
  end;
```

Objekt **tDispT05** obsahuje výkonné metody pro ovládání reálného displeje terminálu Term05.

5.2.1. Init

```
constructor Init(TermOwner:pATerm; CharColls, CharRows:byte;
                 Adresa:word; EnIniHWProc:Boolean);
```

TermOwner je ukazatel na nadřízený objekt terminálu. **CharColls** je počet sloupců displeje. **CharRows** je počet řádek displeje. **Adresa** je bázová adresa PBusu, na který je displej připojen. Je-li **EnIniHWProc** = true, při initu se provede hardwarové nastavení displeje.

5.2.2. InitDisplay

```
procedure InitDisplay;    virtual;
```

Procedura provede základní HW nastavení displeje.

5.2.3. Done

```
destructor  Done;   virtual;
```

Metoda zhasne podsvícení displeje a vymaže displej. Destruuje objekt displeje.

5.2.4. DTickRefreshScr

```
procedure DTickRefreshScr;    virtual;
```

Metoda provede kompletní obnovu zobrazení displeje včetně nastavení podsvícení a polohy kursoru. (Tuto metodu používá objekt menu pro obnovu zobrazení dat.)

5.2.5. MoveDispCur

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

Metoda přesune HW pozici kurzoru na pozici ve sloupci X zleva a na řádek Y shora. Pozice [1,1] je vlevo nahoře.

5.2.6. WriteStr

```
procedure WriteStr(const S:string);   virtual;
```

Metoda vypíše na aktuální pozici kurzoru řetězec S. S je „Pascalského formátu“.

5.3. tKeybT05

```
pKeybT05=^tKeybT05;
tKeybT05=object(tAkeyb)
  FlBell         :Boolean;
  FlBeepKey      :Boolean;
  K_FlRepeat     :Boolean;
  K_NBegRep      :Word;
  K_NTickRep     :Byte;
  K_FlChangeScan :Boolean;
  K_NbChange     :Word;
  K_BeepTicks    :Byte;
  K_BaseAddr     :Word;
  K_AddrDataOut  :Word;
  K_AddrEnbOut   :Word;
```

```
    K_AddrDataIn   :Word;
    K_AddrEnbIn    :Word;
    K_Result       :Byte;
    K_KonvrtTab    :array[Primar..Sekundar,1..2,1..32]of char;

   constructor Init(TermOwner:pATerm;Len:Word;Adresa:word);
   procedure   mBellOn;  virtual;
   procedure   mBellOff;     virtual;
   procedure   mBeepKeyOn;   virtual;
   procedure   mBeepKeyOff;  virtual;
   procedure   mReptOn;
   procedure   mReptOff;
   procedure   mChangeOn;
   procedure   mChangeOff;
   procedure   ScanKeys;virtual;
   procedure SetKeybParam(const S:tParamStr); virtual;
  function  GetKeybParam(const KeyStr:tParamStr):tParamStr;virtual;
  function  ResultKeybParam: Boolean; virtual;
end;
```

Objekt **tKeybT05** detekuje metodou **ScanKeys** stisk kláves a přiřazuje jim znakové konstanty. Nejlépe je spouštět **ScanKeys** periodicky každých 10ms z přerušení časovače. Při podržení klávesy objekt umí generovat její opakovaný stisk. Po dlouhém stisku začne generovat jinou klávesu (rychlejší). Například místo znaku ← generuje znak home. Rychlost opakování, doba kdy začne opakovat a doba na změnu klávesy jsou nastavitelné například řetězcem v metodě **SetKeybParam** . Generované znaky se vybírají z pole **K_KonvrtTab**.

5.3.1. FlBell

Při true je bzučák terminálu trvale zapnutý. Proměnná se nastavuje odpovídající metodou.

5.3.2. FlBeepKey

Při true je zapnuté pípání při stisku klávesy. Proměnná se nastavuje odpovídající metodou.

5.3.3. K_FlRepeat

Při true je povoleno opakování klávesy při dlouhém stisku. Proměnná se nastavuje odpovídající metodou.

5.3.4. K_NbegRep

Proměnná nastavující počet průchodů metodou **ScanKeys** (počet tiků), než začne opakování klávesy. Proměnná se nastavuje textovym řetězcem metodou **SetKeybParam**.

5.3.5. K_NtickRep

Každých **K_NtickRep** po **K_NbegRep** průchodů metodou **ScanKeys** při držení stisknuté klávesy metoda vyrobí opakované stisknutí klávesy. Proměnná se nastavuje textovym řetězcem metodou **SetKeybParam**.

5.3.6. K_FlChangeScan

Při true je povolena změna klávesy po dlouhém stisku. Proměnná se nastavuje odpovídající metodou.

5.3.7. K_NbChange

Po **K_NbChange** průchodů metodou **ScanKeys** při držení stisknuté klávesy se přepne generace znaku na znak z druhé sady **K_KonvrtTab** . Proměnná se nastavuje textovym řetězcem metodou **SetKeybParam**.

5.3.8. K_BeepTicks

Proměnná nastavuje počet průchodů metodou **ScanKeys**, po kterou se generuje písknutí na stisk klávesy. Proměnná se nastavuje textovym řetězcem metodou **SetKeybParam**.

5.3.9. K_Result

V této proměnné je nastavený výsledek operace **SetKeybParam** a otevírání sériové komunikace, pokud je použita. Proměnná nabývá hodnot **Res_Ok** = bez chyby a **Res_SetParamErr** = chyba při zadávání parametru.

5.3.10. K_KonvrtTab

V tabulce **K_KonvrtTab** jsou uložené znaky, které jsou generovány při stisku kláves. Index **primar** a **sekundar** označuje první a druhou generovanou sadu znaků. Při krátkém stisku se generují znaky z primární sady, po **K_NbChange** průchodech metodou **ScanKeys** se generují znaky ze sekundární sady. Druhý index je adresa sloupce membránové klávesnice, na který se přivádí napětí. Třetí index je adresa čtené řádky klávesnice.

V konstruktoru Init se tabulka nastavuje na dále uvedené hodnoty. Hodnoty znakových konstant jsou uvedeny v manuálu uATerm a některé nestandardní znaky jsou definovány v této jednotce.

```
Primar     =1;    {Primární znaky z klávesnice po krátkém stisku}
Sekundar   =2;    {Sekundární znaky z klávesnice po dlouhém stisku}
zPlusPlus  =#$AB; {znak '+' s nahozeným nejvyšším bitem}
zMinusMinus=#$AD; {znak '-' s nahozenym nejvyšším bitem}
kNone      =#$00; {žádná známá klávesa}


K_KonvrtTab[Primar,1,1]  :=zF2; {znak F2}
K_KonvrtTab[Primar,1,2]  :='-';
K_KonvrtTab[Primar,1,4]  :=zLe; { znak šipka doleva}
K_KonvrtTab[Primar,1,8]  :=zCr;
```

```
  K_KonvrtTab[Primar,1,$10]:=zDn; {znak šipka dolů}
  K_KonvrtTab[Primar,2,1]  :=zF1; {znak F1}
  K_KonvrtTab[Primar,2,2]  :='+';
  K_KonvrtTab[Primar,2,4]  :=zUp; {znak šipka nahoru}
  K_KonvrtTab[Primar,2,8]  :=zEsc;
  K_KonvrtTab[Primar,2,$10]:=zRi; {znak šipka doprava}

  K_KonvrtTab[Sekundar,1,1]  :=zF2;
  K_KonvrtTab[Sekundar,1,2]  :=zMinusMinus;
  K_KonvrtTab[Sekundar,1,4]  :=zHome;
  K_KonvrtTab[Sekundar,1,8]  :=zCr;
  K_KonvrtTab[Sekundar,1,$10]:=zPgDn;
  K_KonvrtTab[Sekundar,2,1]  :=zF1;
  K_KonvrtTab[Sekundar,2,2]  :=zPlusPlus;
  K_KonvrtTab[Sekundar,2,4]  :=zPgUp;
  K_KonvrtTab[Sekundar,2,8]  :=zEsc;
  K_KonvrtTab[Sekundar,2,$10]:=zEnd;
```

5.3.11. Init

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

Init je konstruktor objektu klávesnice. **TermOwner** je ukazatel na nadřízený objekt terminálu, **Len** je délka bufferu stisknutých znaků z klávesnice a **Adresa** je bázová adresa PBusu, na který je klávesnice připojena.

5.3.12. mBellOn

```
procedure   mBellOn;   virtual;
```

Metoda zapne nepřetržitý zvukový signál.

5.3.13. mBellOff

```
procedure   mBellOff;   virtual;
```

Metoda vypne nepřetržitý zvukový signál.

5.3.14. mBeepKeyOn

```
procedure   mBeepKeyOn;  virtual;
```

Metoda zapne pípání po stisku klávesy.

5.3.15. mBeepKeyOff

```
procedure   mBeepKeyOff;  virtual;
```

Metoda vypne pípání po stisku klávesy.

5.3.16. mReptOn

```
procedure   mReptOn;
```

Metoda povoluje opakovanou generaci znaku při dlouhém držení stisknuté klávesy.

5.3.17. mReptOff

```
procedure   mReptOff;
```

Metoda zakazuje opakovanou generaci znaku při dlouhém držení stisknuté klávesy.

5.3.18. mChangeOn

```
procedure   mChangeOn;
```

Metoda povoluje změnu generované klávesy po dlouhém držení stisknuté klávesy. Je ji třeba držet déle, než **K_NbChange** průchodů metodou **ScanKeys.**

5.3.19. mChangeOff

```
procedure   mChangeOff;
```

Metoda zakazuje změnu generované klávesy po dlouhém držení stisknuté klávesy.

5.3.20. ScanKeys

```
procedure   ScanKeys; virtual;
```

Metoda provede jeden krok testu klávesnice. Při detekci stisknuté klávesy pomocí metody **InsertKey** z **uAtermu** zařadí znak z tabulky **K_KonvrtTab** do fronty přijatých znaků z klávesnice. Metoda počítá dobu stisknutí klávesy v počtech průchodů. Metoda filtruje zákmity membránové klávesnice. Metoda nepodporuje soutisk více kláves s výjimkou soutisku Enter a šipka nahoru. Tato kombinace generuje znak **zShiftEnter**, který bývá nastaven na vyvolání programu SetUp terminálu.

Při držení stisknuté klávesy a při povoleném opakování metoda po **K_NbegRep** průchodech začne opakovat generaci znaku z primární tabulky **K_KonvrtTab** s periodou **K_NtickRep** průchodů. Je–li povolena změna generované klávesy, po **K_NbChange** průchodech metoda vybere znaky ze sekundární sekce tabulky **K_KonvrtTab** a opět je bude opakovat s periodou **K_NtickRep**.

Uživatel by měl metodu periodicky volat minimálně každých 55 ms, ale lépe každých 10 ms.

5.3.21. SetKeybParam

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

Metoda slouží na nastavení parametrů počtu průchodů metodou **ScanKeys** na opakování a změny kláves. V **Initu** jsou tyto parametry nastaveny na hodnoty, které vyhoví při vyvolávání metody **ScanKeys** po 10ms. Při jiné periodě, nebo při požadavku na jiné chování klávesnice, můžeme tyto parametry přestavit. Vstupním parametrem metody je řetězec **S**, který může mít například tento obsah, který odpovídá implicitnímu nastavení:

```
'REP=75 CNG=300 DRP=30 LBP=4'
```

REP je počet tiků do opakování klávesy. **CNG** je počet tiků do změny klávesy. **DRP** je dělič tiků pro opakování klávesy. **LBP** je počet tiků délky pískání.

5.3.22. GetKeybParam

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

Funkce z důvodu kompatibility má vstupní parametr **KeyStr** typu string. Tento string může mít libovolný obsah, například být prázdný. Jako svou hodnotu vrací string s nastavením klávesnice odpovídajícím metodě **SetKeybParam**.

5.3.23. ResultKeybParam

```
function  ResultKeybParam: Boolean;
```

Funkce vrací jako svou hodnotu, zda je výsledek operace **SetKeybParam**=**Res_OK**. Hodnota true tedy značí, že operace proběhla v pořádku.

5.4. tTermT05

```
type
  tOnSetupT05  = procedure(P: pTermT05);
type
  pTermT05 = ^tTermT05;
  tTermT05 = object(tTermChr)
    StLight               :Byte;
    StDelay               :Byte;
    FlgDark               :Boolean;
    TimeLastKeyPressed    :Integer;
    OnSetupTerminal       :tOnSetupT05;
    ShareOutputImage      :byte;
    constructor Init(NewDisp :pADisp; NewKeyb: pAKeyb;
                     NewChnTerm: pChnVirt; NewChnRecBuf: pointer);
    procedure Tick;                              virtual;
    procedure FLLight  (B: Byte);                virtual;
    procedure SetDelay (B: Byte);                virtual;
    procedure BellOn;                            virtual;
    procedure BellOff;                           virtual;
    procedure BeepKeyOn;                         virtual;
    procedure BeepKeyOff;                        virtual;
    function  KeyPressed:Boolean;                virtual;
    procedure SetupTerminal;                     virtual;
    procedure SetOnSetupTerminal(P : Pointer); virtual;
  end;
```

Objekt **tTerm05** je vrcholovým objektem této jednotky. Je dědicem znakového terminálu a obsahuje tedy všechny potřebné funkce pro spolupráci s objektem pro tvorbu obrazovek menu.

5.4.1. StLight

V této proměnné je zaznamenáno nastavení podsvícení displeje terminálu.

5.4.2. StDelay

V této proměnné je nastavena prodleva v minutách od posledního stlačení klávesnice, kdy se zhasne podsvícení displeje. Při nastavení na 0 se nezhasíná. Proměnnou nastavuje odpovídající metoda.

5.4.3. FlgDark

Příznak zhasnutí displeje (true = zhasnuto).

5.4.4. TimeLastKeyPressed

V této proměnné je zaznamenán čas (minuty*60+sec) posledního stisku klávesy. Slouží pro automatické zhasnutí podsvícení displeje.

5.4.5. OnSetupTerminal

Adresa procedury, která se vola při vyvolání Setup, pokud je adresa nastavena.

5.4.6. ShareOutputImage

Obraz společných výstupu na Pbus pro display a klávesnici. Jednotlivé objekty zapisují do tohoto obrazu a ten se jednou naráz zapisuje do HW.

5.4.7. Init

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

Parametr **NewDisp** je ukazatel na objekt displeje, parametr **NewKeyb** je ukazatel na objekt klávesnice. Parametr **NewChnTerm**: je ukazatel na komunikační objekt a **NewChnRecBuf** je ukazatel na přijímací buffer komunikačního kanálu. Pokud nastavíme ukazatele komunikačního kanálu na nil, žádný kanál se neinicializuje. V opačném případě se kanál se s dodaným nainicializovaným kanálem provede open a connect a rámci metody tick se komunikační kanál obsluhuje.

Obsluha komunikačního kanálu je v objektu zejména z důvodu kompatibility. Mělo by to smysl, pokud bychom vytvářeli samostatně existující terminál zobrazující znaky přijaté po sériové komunikaci a odesílající znaky sejmuté z klávesnice. Takový terminál je již existující Term06, který je řešen jiným hardwarem.

5.4.8. Tick

```
procedure Tick;   virtual;
```

Metoda volá zděděný Tick, který provádí volání metody **DtickRefreshScr** displeje. Detekce klávesnice musí být volána častěji a proto není do tiku zařazena. Pokud existuje komunikační kanál obslouží se v Ticku a jsou-li v něm znaky, zavolá se metoda **ReceiveDataFromRemoteTermTick**. Tato metoda je ovšem doposud abstraktní.

5.4.9. FLLight

```
procedure FLLight  (B: Byte); virtual;
```

Nastaví podsvětlení displeje.

5.4.10. SetDelay

```
procedure SetDelay (B: Byte); virtual;
```

Metoda nastaví prodlevu mezi posledním stlačením klávesnice a zhasnutím podsvětlení displeje. Nastaví-li se na nulu, ke zhasnutí nedochází.

5.4.11. BellOn

```
procedure BellOn; virtual;
```

Zapne trvale zvukovou signalizaci.

5.4.12. BellOff

```
procedure BellOff; virtual;
```

Vypne trvalou zvukovou signalizaci. Pokud je nastavené pípání při stisku kláves, tak ta nadále funguje.

5.4.13. BeepKeyOn

```
procedure BeepKeyOn;  virtual;
```

Zapne pípání na stisk klávesy.

5.4.14. BeepKeyOff

```
procedure BeepKeyOff;  virtual;
```

Vypne pípání na stisk klávesy.

5.4.15. KeyPressed

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

Je-li neprázdná fronta přijatých znaků z klávesnice, vrací true, jinak false.

5.4.16. SetupTerminal

```
procedure SetupTerminal; virtual;
```

Virtuální metoda volaná po stisku kláves generující SetUp terminálu. Pomocí metody **SetOnSetupTerminal** se sem přiřadí skutečný SetUp terminálu.

5.4.17. SetOnSetupTerminal

```
procedure SetOnSetupTerminal(P : Pointer); virtual;
```

Metoda přiřazuje SetUp terminálu do procedury **SetupTerminal**.

6. Příklad vytvoření terminálu včetně nadřízeného menu

```
{ Inicializace terminalu }
 {$ifdef SimTerm05}
  pMyTerm:=New(pTermT05,Init(
              New(pSimDispT05,Init(nil{pMyTerm},16,2, 3,3,16,2)),
              New(pSimKeybT05,Init(nil{pMyTerm},20,FlgEnd)),
              nil,nil));
 {$else}
    {$ifdef VerMc}
  pMyTerm:=New(pTermT05,Init(
```

```
              New(pDispT05   ,Init(nil{pMyTerm},16,2,AdrTerm,True)),
              New(pKeybT05,Init(nil{pMyTerm},20,$d210)),
              nil,nil));
    {$else}
  pMyTerm:=New(pTermT05,Init(
              New(pDispT05   ,Init(nil{pMyTerm},16,2,AdrTerm,True)),
              New(pKeybT05,Init(nil{pMyTerm},20,$0220)),
              nil,nil));
    {$endif}
 {$endif}


pMyMenu:=New(pMenuChr,Init(@ProcPtrArray,ord(EndMenu),DispStr,HlpStr,
pMyTerm,'',2));
  InitRunMenu(pMyMenu,MenuName,MenuStk,MenuSPrio,MenuDPrio,
              TermName,TermStk,TermSPrio,TermDPrio,
              2,2,          {wait ve volani Menuter2.Run;
Term01.Tick}
              edNop,edNop,
              FlgEnd);


  pMyMenu^.Term^.SetTDispParam(sParTerm);
  pMyMenu^.Term^.FlIns:=false;   {nastaveni prepisovaciho modu}

  InitSetUpT05(pTermT05(pMyMenu^.Term),@Glb^.Setup,3,2,6,1);
  pMyMenu^.SetupKeyChar:=zShiftEnter;

  pKeybT05(pMyMenu^.Term^.Keyb)^.mChangeOn;    {povol dlouhy stisk}
```