Začínáme s Term10

PŘÍRUČKA 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 1.20

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: 07.04.2004
Datum posledního uložení dokumentu: 07.04.2004
(Datum vydání a posledního uložení dokumentu musí být stejné)

Obsah :

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00		Ce		První vydání.
1.10		Tu	05.06.2003	Úprava dokumentu dle ISO9000. Nový obrázek terminálu. Doplnění kapitoly „Vytváření menu-obrazovek v grafické vrstvě“.
1.20		Wil	07.04.2004	Úprava grafického vzhledu dokumentu.

1.2. Účel dokumentu

Tento dokument slouží jako úvod do programování aplikací s vizualizací na terminálu TERM10.

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro čtení tohoto dokumentu není potřeba číst žádný další manuál, ale je potřeba orientovat se v používání programového vybavení SofCon.

2. Termíny a definice

Používané termíny a definice jsou popsány v samostatném dokumentu „Termíny a definice“.

3. Úvod

Tato příručka je určena pro seznámení s programováním **terminálu TERM10** firmy ***SofCon*** jako řídícího systému s lokálním ovládáním a vizualizací. Zaměřuje se především na práci s jednotkami pro vytváření **komfortního grafického uživatelského rozhraní**, které tvoří **hierarchický systém tzv. menu-obrazovek**. Na několika příkladech se čtenář postupně seznámí se základními principy funkce uživatelského rozhraní a s nástroji, které má při programování k dispozici. Měl by tak být **velmi rychle schopen vytvářet jednoduché uživatelské programy**. V textu příručky jsou uvedeny odkazy na příklady, jejichž kompletní zdrojové texty jsou k dispozici v příloze. Příklady jsou koncipovány a popisovány tak, aby měl čtenář možnost v nich provádět změny, a získané poznatky si tak ihned ověřit v praxi. V případě hlubšího zájmu je možné najít podrobné informace v referenčním popisu programového vybavení ***SofCon***.

Systém terminálu je postaven na bázi stavebnice Kit firmy ***SofCon***, a proto lze využít všech výhod, které tato stavebnice poskytuje. Jádrem může být buď **procesor V40 nebo 386EXR, oba programově kompatibilní s 8086**. Aplikační programy jsou implementovány **v jazyce Turbo Pascal** a lze je zpočátku **ladit v počítači IBM-PC**. V této fázi vývoje je možné využít všech možností, které poskytují Turbo Pascal a Turbo Debugger. Zároveň je možné fyzický **terminál TERM10 simulovat na PC** (LCD displej je nahrazen okénkem na obrazovce v grafickém režimu a místo klávesnice terminálu je použita klávesnice PC). Přídavné moduly lze připojit k desce PC-KitV40, a tak mít k dispozici celý reálný systém. Příručka se však zaměřuje výhradně na obsluhu terminálu, a proto nejsou přídavné moduly v příkladech používány. K seznámení s programovým vybavením tudíž zpočátku postačí pouze počítač PC.

Pro další fázi vývoje a ladění programu v terminálu TERM10 je k dispozici nástroj **ReTOS-Debugger**. Ten slouží k vytváření binárního obsahu paměti EPROM a případně k jeho zavedení po sériové lince do paměti FLASH-EPROM v terminálu. ReTOS-Debugger dále umožňuje on-line ladění, sledování a modifikaci chodu aplikačního programu v terminálu z PC.

Programové vybavení je založeno na využití **operačního systému reálného času ReTOS**. Systém poskytuje možnost vytváření **paralelních programů** a je implementován jako samostatná jednotka v Turbo Pascalu. Úloha je tak rozdělena do několika paralelních procesů, kde každý z nich řeší její relativně samostatnou část.

K vytváření aplikačních programů je nezbytná znalost programovacího jazyka Turbo Pascal. Přestože je programové vybavení implementováno pomocí objektově orientovaného programování, lze základní a jednoduché aplikace vytvořit i bez jeho hlubší znalosti. Doporučuje se též alespoň stručné seznámení s o.s. ReTOS.

4. Úloha uživatele při vytváření aplikačního programu

Pro tvorbu aplikačních programů je připraveno několik programových jednotek, které zajišťují obsluhu terminálu a hierarchického systému menu-obrazovek. Jednotky deklarují mimo jiné objekt terminálu a objekt menu. Po jejich inicializaci jsou spuštěny dva paralelní procesy. První periodicky volá metody objektu terminálu pro komunikaci s fyzickým hardware terminálu a druhý periodicky volá metody

objektu menu pro obsluhu systému menu. Úkolem uživatele při vytváření aplikačního programu je pak pouze **definovat jednotlivé menu-obrazovky a jejich vzájemné propojení.** K tomu má k dispozici řadu prostředků, se kterými ho tato příručka postupně seznamuje.

Každá menu-obrazovka je implementována jako uživatelem naprogramovaná **definiční procedura**. Odkazy na definiční procedury jednotlivých menu-obrazovek jsou uloženy do pole. Podle indexu právě aktuální menu-obrazovky je z pole vybírána příslušná definiční procedura a ta je periodicky volána z kontextu procesu obsluhujícího systém menu. Po každé změně aktuální menu-obrazovky jsou datové struktury menu naplněny implicitními hodnotami. Tím je dáno standardní chování menu-obrazovky. Uživatel pak v definiční proceduře **definuje pouze změny oproti tomuto standardnímu nastavení.**

5. První menu pro TERM10

Zpočátku budeme pro vytváření aplikačních programů pro terminál TERM10 využívat jeho **simulaci na počítači PC**. První programy budou spouštět pouze proces pro terminál a proces pro systém menu. V reálném případě je pak možné spouštět ještě řadu dalších procesů pro řízení, regulaci atd.

Hned zpočátku se pokusíme vytvořit jednoduchý systém složený ze dvou menu-obrazovek. Hlavní program bude vypadat takto (soubor **T10DEMO1.PAS** přílohy):

```
program T10Demo1;
{$I Sets.inc}
uses
  Menu1         { Uzivatelske menu                                  }
  Kernel,       {Operacni system ReTOS                              }
  uInit,         {Inicializace                                      }
  uExit;        { Ukonceni                                          }

begin
  PrgInit;                  { Inicialzace                           }
  UserMenu;                 { Spusteni uzivatelskeho menu           }
  SetPriority(1,254);       { Nastaveni priority v o.s.RETOS        }
  repeat                    { Smycka hlavniho procesu               }
    Wait(4);                { pouze ceka }
  until FlgEnd;             { na nastaveni priznaku ukonceni
                              programu   }
  PrgExit;                  { Ukonceni programu a HW                }
end.
```

Hlavní program využívá čtyři jednotky, kde jednotka **Kernel** obsahuje **jádro operačního systému ReTOS,** jednotka uInit inicializaci a uExit ukončení programu. Jednotka **Menu1** obsahuje vlastní **uživatelské menu**. Soubor Sets.inc obsahuje nastavení přepínačů pro překlad.

Procedury PrgInit a PrgExit slouží k inicializaci a ukončení programu, UserMenu inicializuje uživatelské menu. Během inicializace menu je spuštěn proces pro terminál a proces pro menu. Hlavní program dále setrvá ve smyčce, dokud není nastaven příznak ukončení programu.

Jednotka **Menu1** obsahující vlastní uživatelské menu (soubor **MENU1.PAS** přílohy):

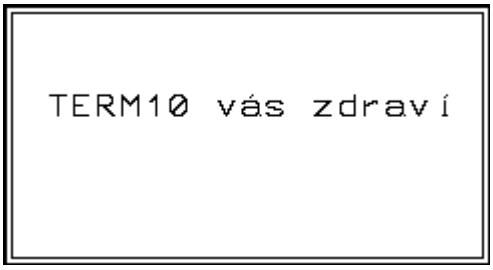
```
unit Menu1;
{$I Sets.inc}
interface
procedure UserMenu;     { spousti uzivatelske menu }
{=====================================================================}
implementation
uses
  BMPTCh,           { data pouzite bitmapy - Tynsky chram           }

  F8x8c,            { data fontu 8x8                                }
  F12x16c,          { data fontu 12x16                              }
  G240x128,         { Grafika 240 x 128 pixelu                      }
  uAMenu,           { Abstraktni Menu                               }
  uMenuChr,         { Znakove Menu                                  }
  uMenuGr,          { Graficke Menu                                 }
  uATerm,           { Abstraktni terminal                           }
  uTermT10,         { Terminal TERM10                               }

 {$IFDEF SimPC}     { Je zapnut simulator TERM10 na obrazovce PC ?  }
  uSimT10,          { Simulator TERM10 na obrazovce a klavesnici PC }
 {$ELSE}
  uDispT10,         { Displej terminalu TERM10                      }
  uKeybT10,         { Klavesnice terminalu TERM10                   }
 {$ENDIF}
  uInit;            { Inicializace - pro priznak FlEnd              }

{---------------------------------------------------------------------}
const
    DispStr : String='';    { String pro vypis textu na display     }
    HlpStr  : String='';    { String pro help                       }
    GraphStr: String='';    { String pro grafiku                    }
    BkBMP   : Integer=-1;   { Bitmapa pozadi                        }
{---------------------------------------------------------------------}
type                { Vyctovy typ jednotlivych menuobrazovek        }
  tSelector=(tMPozdrav,
             tMKonec
            );
const               { Konstanta prvni a posledni menu-obrazovky     }
  BegMenu = tMPozdrav;
  EndMenu = tMKonec;
{---------------------------------------------------------------------}
{           M e n u - o b r a z o v k y                             }
{---------------------------------------------------------------------}
procedure pMPozdrav(P: pMenuGr); far;   { TERM10 vas zdravi         }
begin
  with P^ do
  begin
    if ChangeMenu then      { Doslo k zmene Menu-obrazovky ?        }
    begin
      BkBmp     :=-1;       { Nastaveni pozadi -1= prazdne          }
                            { Formatovany text pro vypis            }
      DispStr   :=zESC+ 'F1,20,40;' +'TERM10 vas zdravi';
                            { Formatovany text pro help             }
      HlpStr    :=zESC+'F2,60,12;'  +'NÁPOVĚDA';
      GraphStr  :='R(2,2,237,125)';
    end;
  end;
end;
{---------------------------------------------------------------------}
procedure pMKonecThird(P: pMenuGr);far;{Procedura pro ukonceni      }
begin          { Doslo-li k stisku [Q] nastavi se priznak ukonceni  }
  if p^.ActChar='Q' then FlgEnd:=True;
end;
```

```
procedure pMKonec(P: pMenuGr); far;   { Tynsky chram , Quit         }
begin
  with P^ do
  begin
    if ChangeMenu then      { Doslo k zmene Menu-obrazovky ?        }
    begin
      BkBMP:=0;             { Nastaveni pozadi 0 = Tynsky chram     }
                            { Formatovany text pro vypis            }
      DispStr:=zESC+'F0,190,110;'  +'Quit';
                            { Formatovany text pro help             }
      HlpStr :=zESC+'F0,40,45;'    +'Stisk Q ukonci program';
                            { Grafika - nic                         }
      GraphStr:='';
         AddTerminateChar(['Q']);{Pridani ukoncovaci klavesy        }
                            { Instalace procedury pro ukonceni      }
      SetUserThird(Addr(pMKonecThird));
    end;
  end;
end;
{-------------------------------------------------------------------}
{-------------------------------------------------------------------}
const                       { Pole procedur menu-obrazovek          }
ProcPtrArray  :array[tSelector]of tMMenuGr =
             (
             pMPozdrav,
             pMKonec
             );
{-------------------------------------------------------------------}
const               { Adresa terminalu TERM10 v IO prostoru         }
 {$ifdef SimPC}
  AdrTerm =$300;
 {$else}
  {$ifdef VerPC}
  AdrTerm =$300;
  {$endif}
 {$endif}

 {$ifdef VerMc}
  AdrTerm =$2300;
 {$endif}
                    { Nastaveni parametru terminalu                 }
  sParTerm='CMODE=2 CCHAR='+#$80+' REF=ON';
                    { Definice procesu Tick  - viz o.s.RETOS        }
 TermName = 'TICK'; { Jmeno procesu                                 }
  TermStk   = 8000; { Velikost zasobniku procesu                    }
  TermSPrio = 99;   { Staticka priorita                             }
  TermDPrio = 254;  { Dynamicka priorita                            }

                    { Definice procesu Menu                         }
  Menuname  ='MENU';{ Jmeno procesu                                 }
  MenuStk = 8000;   { Velikost zasobniku procesu                    }
  MenuSPrio = 101;  { Staticka priorita                             }
  MenuDPrio = 254;  { Dynamicka priorita                            }
{-------------------------------------------------------------------}

procedure UserMenu;
var pMyMenu :pMenuGr;{Ukazatel na objekt uzivatelskeho menu         }
    pMyTerm:pTermT10;{Ukazatel na objekt terminalu                  }
begin
 {$IFDEF SimPC}     { Je zapnut simulator TERM10 na obrazovce PC ?  }
                    { Inicializace simulatoru terminalu TERM10 }
MyTerm:=New(pTermT10,Init(New(PSimDispT10,Init(nil,20,6,AdrTerm,true,
            0,0,2,2,265,165,False)),
            New(PSimKeybT10,Init(nil,20,AdrTerm,FlgEnd)),
            AdrTerm,nil,nil));
 {$ELSE}            { Neni zapnut simulator TERM10 na obrazovce PC  }
```

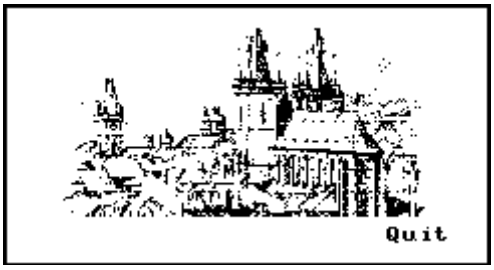
```
                    { Inicializace terminalu TERM10                 }
pMyTerm:=New(pTermT10,Init(New(PDispT10,Init(nil,20,6,AdrTerm,True)),
             New(PKeybT10,Init(nil,20,AdrTerm)),
             AdrTerm,nil,nil));
 {$ENDIF}
                    { Nastaveni parametru terminalu                 }
  pMyTerm^.SetDispParam(sParTerm);
                    { Inicializace menu                             }
  pMyMenu:=New(pMenuGr,Init(@ProcPtrArray,
                            ord(EndMenu),
                            DispStr,
                            HlpStr,
                            GraphStr,
                            @BkBMP,
                            pMyTerm,
                            '',
                            2));
                    { Spusteni menu                                 }
  InitRunMenu(pMyMenu,
              MenuName,MenuStk,MenuSPrio,MenuDPrio,
              TermName,TermStk,TermSPrio,TermDPrio,
              2,2,
              edNop,edNop,
              FlgEnd);

end;

{-----------------------------------------------------------------------}

procedure InitializeBMPs;
var Lng:word;

                    { Prirazeni adresy bitmapy s Tynskym chramem    }
begin               { do pole ukazatelu na uzivatelske bitmapy.     }
  User_BkBMPs[0].AssignBMPAddr(GetBMPAddr_BMPTCH(Lng),Lng);
end;

{-----------------------------------------------------------------------}
const
 { Identifikatory typu pouzitych fontu }
    cF8x8c    = 0;
    cF12x16n  = 1;

{-----------------------------------------------------------------------}

procedure InitializeAppFonts;   { Inicializace fontu                }
var   PomLng  :word;
begin
  { Inicializace pole zobrazovacich procedur }
  with User_FontArray[cF8x8c   ] do
  begin
    P:=pViewFont08x08;
    U:=GetFontAddr_F8x8c(PomLng);
  end;

  with User_FontArray[cF12x16n ] do
  begin
    P:=pViewFont12x16;
    U:=GetFontAddr_f12x16c(PomLng);
  end;
end;

Begin
  InitializeAppFonts;           { Inicializace bitmap               }
  InitializeBMPs;               { Inicializace fontu                }
End.
```

Voláním procedury **UserMenu** došlo k inicializaci objektů terminálu a menu a k volání procedury **InitRunMenu**, která spouští procesy pro terminál a pro menu.

Vlastní menu-obrazovky jsou implementovány pomocí procedur **pMPozdrav** a **pMKonec**. PMPozdrav zobrazí nápis "TERM10 vás zdraví" na orámované obrazovce (Obr. A) a pMKonec obrázek Týnského chrámu s nápisem "Quit" (Obr. B). Z úvodní menu-obrazovky lze přejít do druhé stiskem klávesy **"šipka dolů"** a z druhé do první stiskem klávesy **"šipka nahoru".** V obou menu-obrazovkách lze vyvolat **nápovědu** stiskem klávesy **"F1".** Návrat z nápovědy je možný stiskem klávesy "ENTER" nebo "ESC". Stiskem klávesy "Q" v druhé menu-obrazovce se program ukončí.

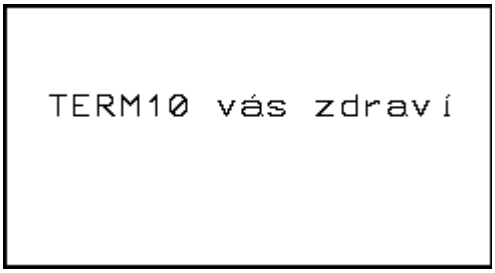


obr. A - pMPozdrav



obr. B - pMKonec

Vzhled menu-obrazovky vzniká skládáním tří grafických stránek. Jejich obsah definuje uživatel přiřazením hodnot do proměnných **DispStr, GraphStr** a **BkBmp**. Proměnná DispStr definuje **textový výstup**, proměnná GraphStr definuje **grafický výstup** a proměnná BkBmp obsahuje index do pole **uživatelských bitmap,** a tím definuje **grafické pozadí** menu-obrazovky. Index -1 přiřazený do BkBmp v první menu-obrazovce má význam prázdného pozadí. Obsah jednotlivých grafických stránek ukazují Obr. C až Obr. H:

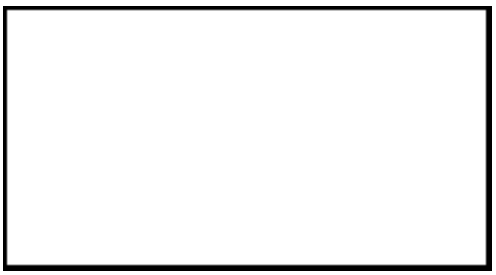


obr. C - pMPozdrav DispStr

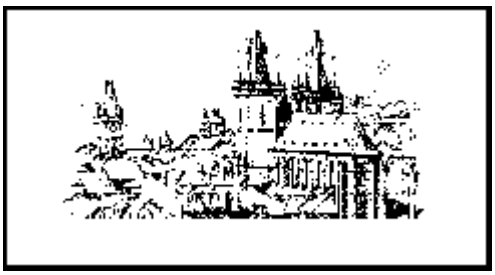
obr. D - pMKonec DispStr

obr. E - pMPozdrav GraphStr

obr. F - pMKonec GraphStr



obr. G - pMPozdrav BkBmp



obr. H - pMKonec BkBmp

K **formátování textového výstupu** v proměnné DispStr lze použít řídících znaků a ESC sekvencí. Terminál TERM10 je schopen emulovat znakový terminál. Rozměry rastru znakového terminálu (počet řádků a sloupců) se zadávají jako parametry constructoru objektu displeje v těle procedury UserMenu:

```
... Init(nil,20,6, ...
```

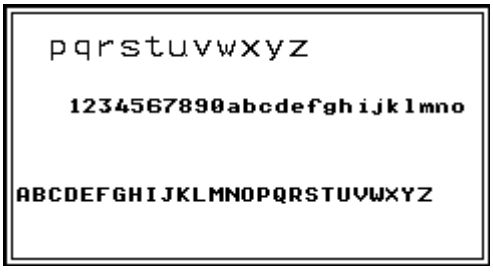
- tedy 6 řádků po 20 znacích. Celkový počet znaků v rastru (řádky × sloupce) nesmí přesáhnout 255. Implicitně je okénko znakového terminálu umístěno v levém horním rohu obrazovky a k výpisu je použit implicitní font. K **pohybu ve znakovém rastru** lze používat **řídící znaky a ESC sekvence**. Pomocí ESC sekvence lze též **okénko znakového terminálu přesunout** a **zvolit font pro výpis**. Příkladem výpisu ve znakovém rastru je i nápis "TERM10 vás zdraví" na úvodní menu-obrazovce. Jedná se však pouze o jednořádkový text. Pro příklad výpisu ve znakovém rastru na více řádků modifikujeme úvodní menu-obrazovku:

```
...
      DispStr:=zEsc+'F0,40,45;'+ '1234567890abcdefghijklmno'+
                  +zDn+          'pqrstuvwxyz'+
                  +zCr+zLf+      'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
...
```

obr. I - výpis ve znakovém rastru

Vzhled menu-obrazovky v tomto případě je na Obr. I. Pomocí ESC sekvence je okénko přesunuto na grafické souřadnice x = 40 a y = 45 pixelů a pro výpis je zvolen font s indexem 0. Po vypsání písmene "o" pokračuje výpis o znak níže (zDn) a po vypsání písmene "z" o řádek níže a na jeho začátku (zCr, zLf). Pokud zvolíme styl výpisu (umístění a font) více než jedenkrát, **výpis ve znakovém rastru se neprovádí**. Pro příklad opět modifikujeme úvodní menu-obrazovku:

```
...
      DispStr:=zEsc+'F0,30,45;'+ '1234567890abcdefghijklmno'+
               zEsc+'F1,20,10;'+ 'pqrstuvwxyz'+
               zEsc+'F0,5,90;' + 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
...
```



obr. J - výpis bez znakového rastru

Vzhled menu-obrazovky po této modifikaci je na Obr. J. Styl výpisu je zvolen celkem třikrát. Je možné si všimnout, že ačkoli je počet tištěných velkých písmen abecedy větší než délka řádku znakového rastru, jsou zobrazena na řádku jediném.

Grafický výstup v proměnné GraphStr může obsahovat **objekty**

vektorové grafiky jako bod - P(X,Y), úsečka - L(X1,Y1,X2,Y2), kružnice - C(X,Y,R), prázdný obdélník - R(X1,Y1,X2,Y2) a plný obdélník - F(X1,Y1,X2,Y2). Souřadnice grafických objektů se zadávají v jednotkách pixel. Levý horní roh displeje má souřadnice (0,0). Souřadnice X pak roste zleva doprava a souřadnice Y shora dolů. Pro příklad možností grafického výstupu modifikujeme obsah proměnné GraphStr:

```
...
GraphStr:='R(2,2,237,125)P(50,20)L(10,80,150,120)C(200,100,20)F(20,11
0,70,120)';
...
```

obr. K - možnosti grafického výstupu

Vzhled menu-obrazovky po této modifikaci je na Obr. K.

Podrobný popis syntaxe DispStr a GraphStr je v příloze na konci příručky.

Proměnná **HlpStr** slouží k **výpisu nápovědy**. Formátování je stejné jako u DispStr. Text nápovědy je standardně zobrazen po stisku klávesy "F1" a k návratu slouží klávesa "ESC" nebo "ENTER".

K implicitnímu nastavení datových struktur menu dochází vždy po změně aktuální menu-obrazovky. Není proto nutné provádět v těle definiční procedury při každém jejím volání uživatelská nastavení. Z tohoto důvodu je jejich provádění podmíněno nastavením proměnné **ChangeMenu** na hodnotu true. K němu dochází právě jen po **změně aktuální menu-obrazovky**. Vyřazení uživatelských nastavení z podmínky if ChangeMenu nemá sice sémantický důsledek, avšak zbytečně prováděné opakovaní téhož vede k plýtvání časem procesoru. Chceme-li však, aby k určitým změnám docházelo nepřetržitě, je vyřazení z podmínky nezbytné (např. modifikace DispStr při zobrazení proměnné veličiny).

Typ **tSelector** je výčtový typ menu-obrazovek a **ProcPtrArray** je pole procedurálních proměnných (ukazatelů na vstupní body procedur) jednotlivých menu-obrazovek.

V inicializační části jednotky je volána procedura pro inicializaci použitých fontů (InitializeAppFonts) a procedura pro inicializaci použitých bitmap (InitializeBmps).

6. Vytvoření uživatelské bitmapy a fontu

Pozadí menu-obrazovek může být tvořeno uživatelskými bitmapami. Aby byly bitmapy pro program v terminálu TERM10 dostupné, je třeba mít uložen v paměti jejich binární obraz. Proto **vytvoříme pro každou bitmapu jednotku v Turbo Pascalu**. Jednotka musí obsahovat binární obraz příslušné bitmapy a zároveň umožnit zjištění adresy, kde je uložen. Aby byl binární obraz bitmapy uložen do paměti EPROM, je v této jednotce umístěn do těla funkce. Jeho adresu pak získáme jako návratovou hodnotu při volání této funkce. K převodu souboru *.BMP na zdrojový text jednotky *.PAS je určen program **CRLCDBMP.EXE** pod o.s. Windows. Po spuštění převodního programu zvolíme příslušné rozměry bitmapy (240 × 128 LCD pixelů) a otevřeme soubor se zdrojovou bitmapou *.BMP (Open). Zdrojová bitmapa

*.BMP musí být černobílá a mít správné rozměry. Nakonec uložíme zdrojový text jednotky *.PAS (Save).

Obdobným způsobem lze ze souboru Windows fontu *.FNT vytvořit programem **CRLCDFNT.EXE** zdrojový text jednotky obsahující příslušný font.

7. Přidání další menu-obrazovky

V dalším příkladu (soubory **T10DEMO2.PAS** a **MENU2.PAS** přílohy) se pokusíme první menu rozšířit o další menu-obrazovku. K tomu je třeba rozšířit typ **tSelector** o její identifikátor,

```
type                { Vyctovy typ jednotlivych menuobrazovek        }
  tSelector=(tMPozdrav,
             tMPridana,
             tMKonec
            );
```

napsat její **definiční proceduru**,

```
procedure pMPridana(P: pMenuGr); far;   { Pridana menu-obrazovka    }
begin
  with P^ do
  begin
    if ChangeMenu then      { Doslo k zmene Menu-obrazovky ?        }
    begin
      BkBmp     :=1;        { Nastaveni pozadi                      }
                            { Formatovany text pro vypis            }
      DispStr   :=zESC+ 'F2,60,5;'  +'Pridana'+
                  zESC+ 'F0,60,40;' +'menu-obrazovka';
                            { Formatovany text pro help             }
      HlpStr    :=zESC+'F1,60,12;'  +'NÁPOVĚDA'+
                  zESC+'F0,20,40;'  +'K pridane menu-obrazovce';
      GraphStr  :='R(1,1,238,126)L(60,35,180,35)';
    end;
  end;
end;
```

a doplnit ji do pole **ProcPtrArray**.

```
const                       { Pole procedur menu-obrazovek          }
ProcPtrArray  :array [tSelector] of tMMenuGr =
             (
             pMPozdrav,
             pMPridana,
             pMKonec
             );
```

obr. L - přidaná menu-obrazovka

Vzhled přidané menu-obrazovky je na Obr. L. Bude zobrazovat nápis "Přidaná menu-obrazovka" při použití dvou typů fontů, čáru podtrhující slovo "Přidaná" a obrázek, který bude tvořit pozadí. Bude zařazena mezi dvě stávající menu-obrazovky. Protože používáme další typ fontu a další uživatelskou bitmapu, je třeba zařadit jejich jednotky.

```
uses
BMPHR,              { data pouzite bitmapy - Ruce a pres. hodiny    }
F16x32c,            { data fontu 16x32                              }
```

v proceduře **InitializeBMPs** zařadit bitmapu do pole uživatelských bitmap,

```
User_BkBMPs[1].AssignBMPAddr(GetBMPAddr_BMPHR(Lng),Lng);
```

a v proceduře **InitializeAppFonts** zařadit font do pole zobrazovacích procedur.

```
const
cF16x32n  = 2;
.......
  with User_FontArray[cF16x32n ] do
  begin
    P:=pViewFont16x32;
    U:=GetFontAddr_f16x32b(PomLng);
  end;
```

Po spuštění programu bude možné **přecházet** mezi všemi třemi menu-obrazovkami pomocí kláves **"šipka nahoru"** a **"šipka dolů",** ke každé bude možno vyvolat **nápovědu** klávesou **"F1"** a program **ukončit** stiskem klávesy **"Q"** v poslední menu-obrazovce.

8. Hierarchická struktura a propojení menu-obrazovek

Až doposud byly námi vytvořené menu-obrazovky jednoduše řazeny za sebou. Programové vybavení má však mocné nástroje k vytváření hierarchických struktur s vnořenými menu-obrazovkami. Nejčastější příčinou změny aktuální menu-obrazovky je **stisk některé klávesy** terminálu.

8.1. Prostředky pro správu pohybu v hierarchické struktuře

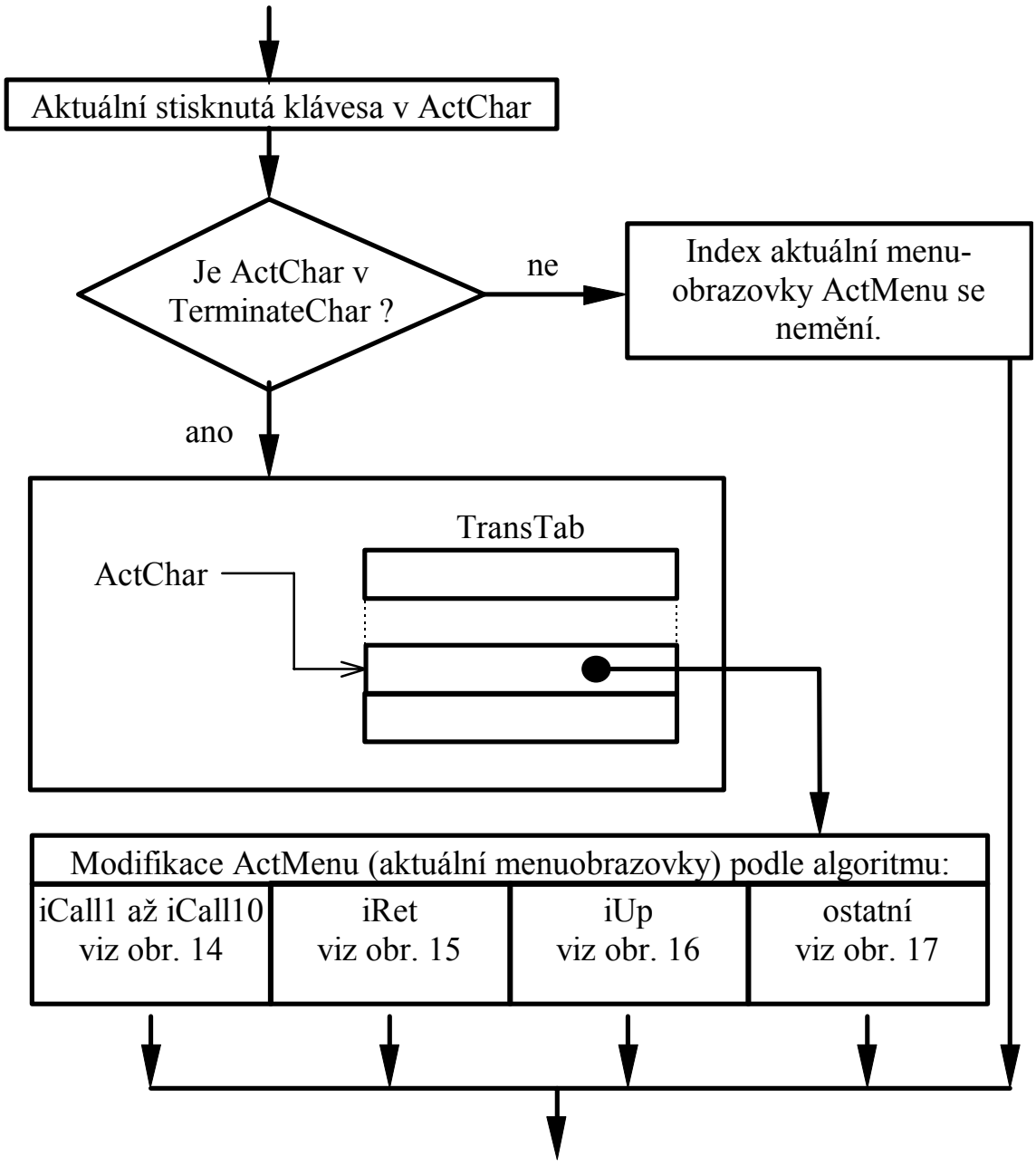
Pro správu pohybu v hierarchické struktuře má objekt menu několik prostředků. Jsou to tabulky **TrasTab** a **CtrlTab**, množina **TerminateChar**, zásobník **StackMenu** a metody pro modifikaci obsahu těchto datových struktur. Jejich obsah definuje index následující menu-obrazovky, pokud dojde ke stisku některé klávesy.

Kromě zásobníku StackMenu jsou datové struktury po změně aktuální menu-obrazovky naplněny **implicitními hodnotami**. Tím je dáno **standardní chování** menu-obrazovky. Pokud chce uživatel toto chování změnit, postačí, když v definiční proceduře **změní tyto implicitní hodnoty**. Modifikaci datových struktur neprovádí přímo, ale prostřednictvím k tomu určených metod objektu menu.

Množina **TerminateChar** obsahuje klávesy, jejichž stisk může vést ke změně aktuální menu-obrazovky. Pokud se aktuální stisknutá klávesa uložená v proměnné **ActChar** nenachází v této množině, aktuální menu-obrazovka se nemění. Tabulka **TransTab** převádí ukončovací klávesu na příkaz, který definuje další postup při určení indexu následující menu-obrazovky. Podle něj jsou dále využívány tabulka CtrlTab a zásobník StackMenu. Tabulka **CtrlTab** přiřazuje jednotlivým příkazům indexy následujících menu-obrazovek a zásobník **StackMenu** slouží k ukládání indexu aktuální menu-obrazovky pro návrat z volání vnořené menu-obrazovky.

8.2. Algoritmus určení indexu následující menu-obrazovky

Algoritmus určení indexu následující menu-obrazovky je znázorněn na Obr. M.



obr. M - algoritmus určení indexu následující menu-obrazovky

Aktuální stisknutá klávesa je uložena v proměnné **ActChar**. Je otestováno, zda se tato klávesa nachází množině ukončovacích kláves **TerminateChar**. Pokud ne, index aktuální menu-obrazovky se nemění. Pokud ano, je obsah ActChar použit jako index do tabulky **TransTab**. Z této tabulky je získán příkaz, který rozhoduje o dalším pokračování algoritmu určení indexu následující menu-obrazovky.

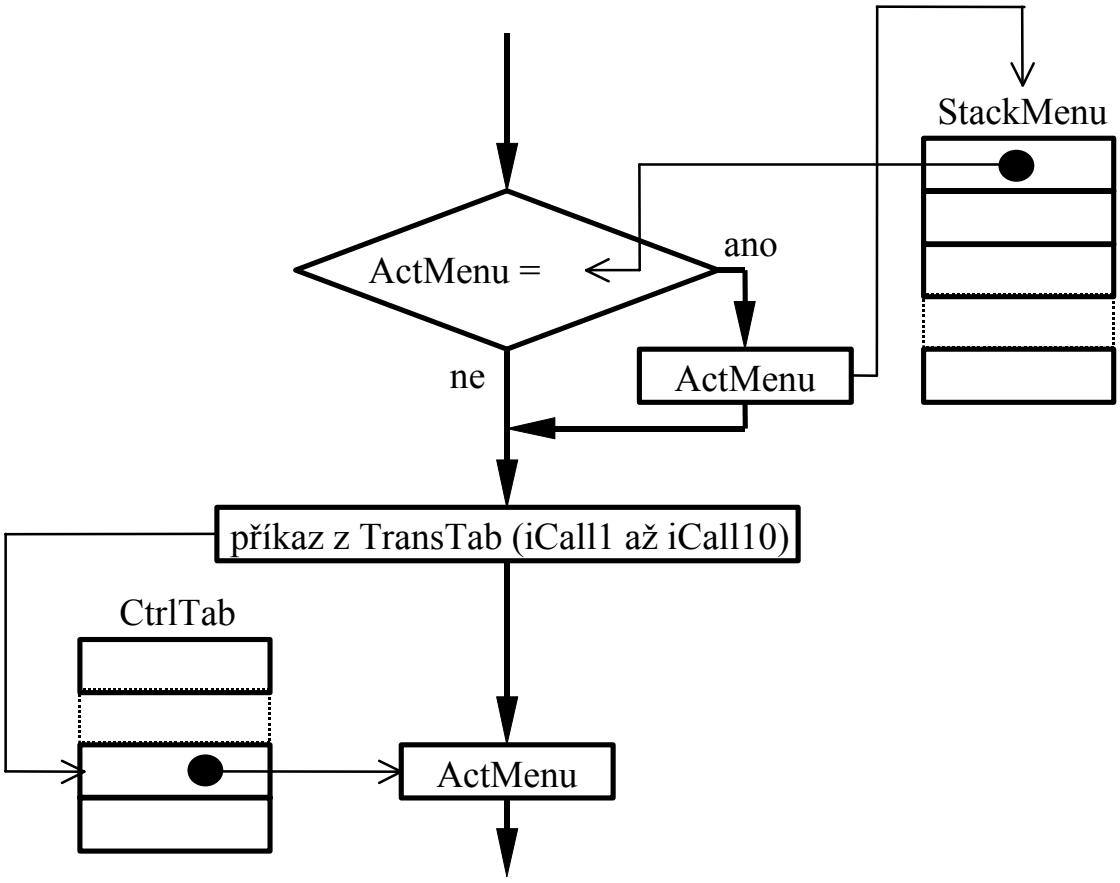
Množina **TerminateChar implicitně** obsahuje prvky **zCr, zEsc, zUp a zDn**, tedy klávesy "ENTER", "ESC", "šipka nahoru" a "šipka dolů". K její modifikaci jsou určeny metody **AddTerminateChar** (přidání ukončovacích kláves),

SubTerminateChar (ubrání ukončovacích kláves), AddSubTerminateChar (funkční obdoba obou předchozích) a **SetTerminateChar** (nastavení ukončovacích kláves).

Tabulka **TrasTab** je implicitně plněna strukturovanou konstantou, jejíž podrobný obsah lze nalézt v referenčním popisu programového vybavení ***SofCon***. Zde uvádíme jen některá nastavení. Pro klávesu zCr ("ENTER") je implicitní nastavení iCr, pro zEsc ("ESC") iRet, pro zUp ("šipka nahoru") iUp a pro zDn ("šipka dolů") iDn. K modifikaci tabulky TrasTab slouží metody **SetTransTab** (nastavení TransTab) a **SetTrnsCtrlTab** (nastavení TransTab a CtrlTab - viz dále).

8.3. Algoritmus určení indexu následující menuobrazovky pro iCall1 až iCall10

Pokračování algoritmu určení indexu následující menu-obrazovky v případě, že je z tabulky TransTab vyzvednuta některá z hodnot iCall1 až iCall10 je znázorněno na Obr. N.



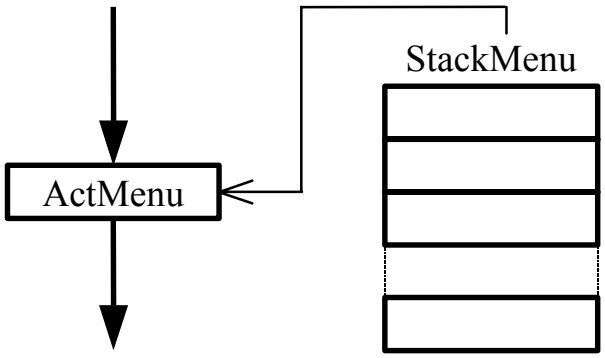
obr. N - algoritmus určení indexu následující menu-obrazovky pro iCall1 až iCall10

V tomto případě se jedná o volání vnořené menu-obrazovky. Je zjištěno, zda se index aktuální menu-obrazovky nachází na vrcholu zásobníku **StackMenu**. Pokud ne, uloží se index aktuální menu-obrazovky do zásobníku. Hodnota ze zásobníku může pak být použita pro návrat. Podmíněné uložení staré hodnoty indexu aktuální menu-obrazovky je použito proto, aby nedocházelo k ukládání téže staré hodnoty indexu aktuální menu-obrazovky v případě, že došlo k návratu jinou cestou než příkazem iRet (viz dále). Pak je použit příkaz iCall1 až iCall10 z tabulky **TransTab** jako index do

tabulky **CtrlTab**. Hodnota z tabulky CtrlTab se stává novým indexem aktuální menu-obrazovky.

Implicitní nastavení tabulky CtrlTab pro příkazy iCall1 až iCall10 je rovno hodnotě indexu aktuální menu-obrazovky. Pokud by uživatel explicitně neprovedl její nastavení v těle definiční procedury, ke změně menu-obrazovky by nedošlo. Byla by pouze uložena hodnota indexu aktuální menu-obrazovky do zásobníku (došlo by vlastně k volání sebe sama). K nastavení hodnot v tabulce CtrlTab slouží metody **SetCtrlTab** (nastavení CtrlTab) a **SetTransCtrlTab** (společné nastavení TransTab a CtrlTab).

8.4. Algoritmus určení indexu aktuální menu-obrazovky pro iRet



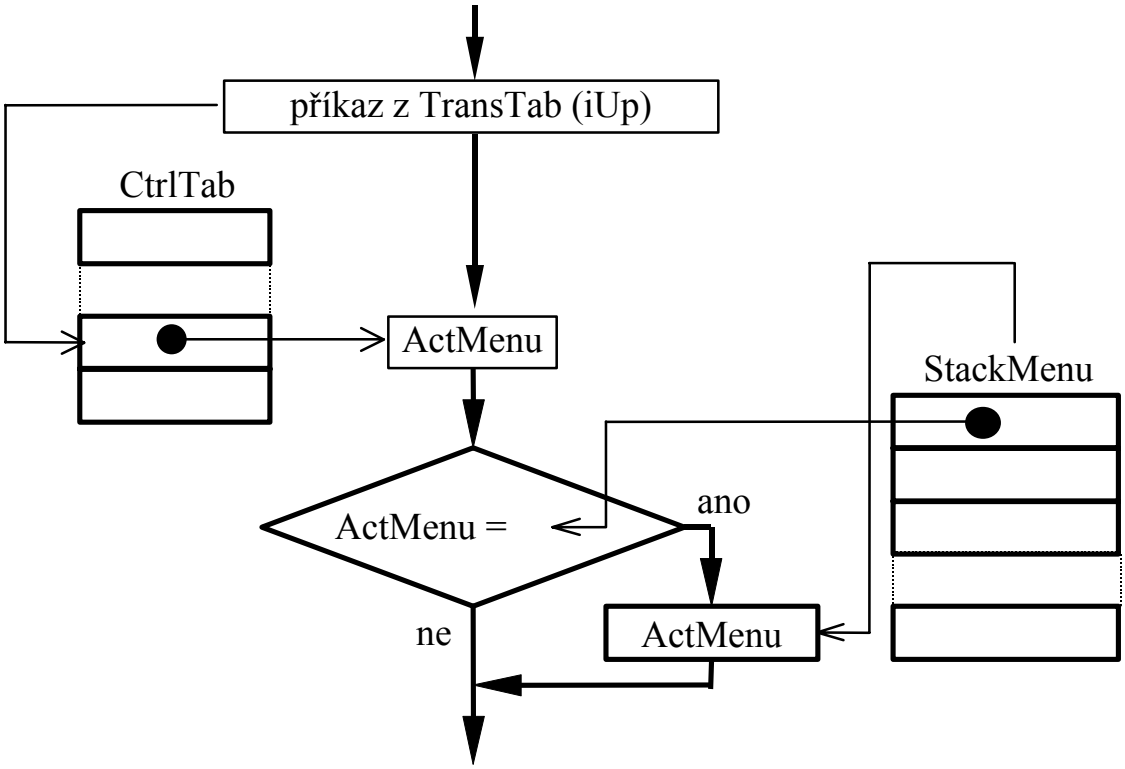
obr. O - algoritmus určení indexu aktuální menu-obrazovky pro iRet

Pokračování algoritmu určení indexu následující menu-obrazovky v případě, že je z tabulky TransTab vyzvednuta hodnota iRet je znázorněno na Obr. O.

V tomto případě se jedná o návrat z vnořené menu-obrazovky. Nová hodnota indexu aktuální menu-obrazovky je vyzvednuta ze zásobníku **StackMenu.**

8.5. Algoritmus určení indexu aktuální menu-obrazovky pro iUp

Pokračování algoritmu určení indexu následující menu-obrazovky v případě, že je z tabulky TransTab vyzvednuta hodnota iUp je znázorněno na Obr. P.



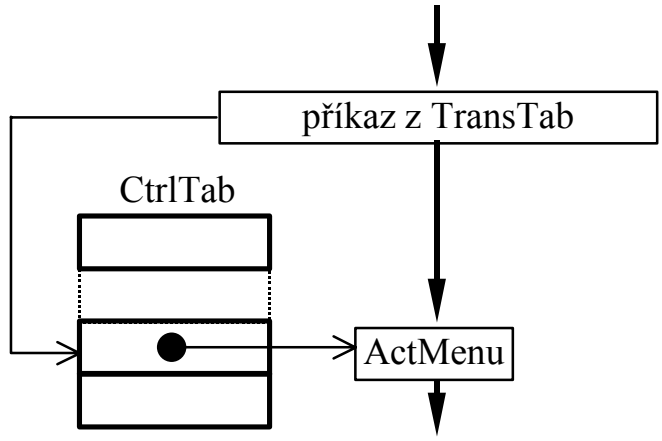
obr. P - algoritmus určení indexu aktuální menu-obrazovky pro iUp

V tomto případě se příkaz **iUp** z tabulky **TransTab** použije jako index do tabulky **CtrlTab**, ze které se vyzvedne nová hodnota indexu aktuální menu-obrazovky. Pokud se tato hodnota zároveň nachází na vrcholu zásobníku, je ze zásobníku vyzvednuta. Podmíněné vyzvednutí hodnoty ze zásobníku umožňuje návrat z vnořené menu-obrazovky v případě, že příkazem iUp přecházíme do menu-obrazovky, ze které byla vnořená menu-obrazovka otevřena.

Jak už bylo zmíněno, je příkaz iUp přiřazen v tabulce TransTab implicitně klávese zUp ("šipka nahoru"). Tabulka CtrlTab je pro příkaz iUp implicitně plněna hodnotou indexu aktuální menu-obrazovky - 1. Standardně se tedy při stisku klávesy **"šipka nahoru"** přechází na menu-obrazovku, jejíž definiční procedura je v poli ProcPtrArray zařazena **před** definiční procedurou aktuální menu-obrazovky.

8.6. Algoritmus určení indexu aktuální menu-obrazovky pro ostatní příkazy

Pokračování algoritmu určení indexu následující menu-obrazovky v případě, že je z tabulky TransTab jiná hodnota než iCall1 až iCall10, iRet a iUp je znázorněno na Obr. Q.



obr. Q - algoritmus určení indexu aktuální menu-obrazovky pro ostatní příkazy

V těchto případech je hodnota příkazu z tabulky **TransTab** použita jako index do tabulky **CtrlTab**, ze které je vyzvednuta nová hodnota indexu aktuální menu-obrazovky.

Pro všechny tyto příkazy, kromě iDn, je tabulka CtrlTab implicitně plněna hodnotou indexu aktuální menu-obrazovky. Pro iDn je tabulka CtrlTab implicitně plněna hodnotou indexu aktuální menu-obrazovky + 1. Jak už bylo zmíněno, je příkaz iDn implicitně přiřazen v tabulce TransTab klávese zDn ("šipka dolů"). Standardně se tedy při stisku klávesy **"šipka dolů"** přechází na menu-obrazovku, jejíž definiční procedura je v poli ProcPtrArray zařazena **za** definiční procedurou aktuální menu-obrazovky.

8.7. Příklad

V následujícím příkladu (soubory **T10DEMO3.PAS** a **MENU3.PAS** přílohy) přidáme další menu-obrazovku, kterou bude možné vyvolat z kterékoli jiné menu-obrazovky kromě úvodní, a vrátit se zpět tam odkud byla vyvolána. Zároveň zajistíme, abychom se při stisku klávesy "šipka nahoru" v menu-obrazovce "Přidaná" nevraceli na úvodní menu-obrazovku, ale přešli na poslední menu-obrazovku a při stisku klávesy "šipka dolů" na poslední menu-obrazovce přešli na menu-obrazovku "Přidaná". V nové menu-obrazovce "Sledování" budeme vypisovat dvě měnící se veličiny. Její vzhled je na Obr. R.

obr. R - menu-obr. "Sledování"

V jednotce Menu tedy provedeme následující změny. Doplníme **definiční proceduru** nové menu-obrazovky:

```
procedure pMSledovani(P: pMenuGr); far; { Menu-obrazovka sledovani  }
begin
  with P^ do
  begin                      { Formatovany text pro vypis           }
```

```
    DispStr   :=zESC+ 'F0,10,10;'  +'Velicina typu real:'+
                 zESC+ 'F4,30,20;'  +RealStrDec(SledRealVel,5,2)+
                 zESC+ 'F0,10,70;'  +'Velicina typu'+
                 zESC+ 'F0,10,80;'  +'     integer:'+
                 zESC+ 'F3,160,50;' +IntegerStrDec(SledIntVel,2);

    if ChangeMenu then      { Doslo k zmene Menu-obrazovky ?        }
    begin
      BkBmp     :=-1;       { Nastaveni pozadi                      }
                            { Formatovany text pro help             }
      HlpStr    :=zESC+'F1,60,12;'  +'NÁPOVĚDA'+
                  zESC+'F0,20,40;'  +'ukazka sledovani velicin'+
                  zESC+'F0,20,50;'  +'navrat klavesou [ESC]';
      GraphStr  :='R(1,1,238,126)C(183,80,40)';
    end;
  end;
end;
```

Samozřejmě je třeba doplnit o novou menu-obrazovku typ **tSelector:**

```
type                { Vyctovy typ jednotlivych menuobrazovek        }
  tSelector=(tMPozdrav,
             tMPridana,
               tMSledovani,
             tMKonec
            );
```

a novou menu-obrazovku doplnit do pole **ProcPtrArray:**

```
const                       { Pole procedur menu-obrazovek          }
ProcPtrArray  :array [tSelector] of tMMenuGr =
             (
             pMPozdrav,
             pMPridana,
               pMSledovani
             pMKonec
             );
```

V poli ProcPtrArray byla nová menu-obrazovka zařazena za "Přidanou", a proto je třeba zajistit, abychom z "Přidané" menu-obrazovky nepřešli po stisku klávesy "šipka dolů" do nové menu-obrazovky, ale až do menu-obrazovky "Konec". Neboť tabulka TransTab je pro klávesu "šipka dolů" implicitně nastavena na hodnotu iDn, postačí nastavit hodnotu v tabulce CtrlTab. Definiční proceduru úvodní menu-obrazovky tedy doplníme:

```
SetCtrlTab([iDn],Word(tMKonec));
```

Stejně tak musíme dolnit definiční proceduru **pMKonec** pro klávesu "šipka nahoru":

```
SetCtrlTab([iUp],Word(tMPridana));
```

Z menu-obrazovky "Přidaná" chceme přecházet stiskem klávesy "šipka nahoru" do poslední menu-obrazovky. Proceduru **pMPridana** tedy dále doplníme:

```
SetCtrlTab([iUp],Word(tMKonec));
```

Dále chceme stiskem klávesy "S" vyvolat novou menu-obrazovku "Sledování". Je tedy třeba přidat klávesu "S" do množiny ukončovacích kláves **TerminateChar**:

```
AddTerminateChar(['S']);  { Pridani ukoncovaci klavesy            }
```

a nastavit **TransTab** a **CtrlTab**:

```
SetTransCtrlTab(['S'],iCall1,Word(tMSledovani));
```

Protože bude TransTab naplněna hodnotou iCall1, bude se při případném přechodu ukládat index aktuální menu-obrazovky do zásobníku menu-obrazovek **StackMenu.**

Podobně chceme z poslední menu-obrazovky přecházet po stisku klávesy "šipka dolů" do menu-obrazovky "Přidaná" a po stisku "S" otevřít novou menu-obrazovku "Sledování".

```
AddTerminateChar(['Q','S']);{ Pridani ukoncovaci klavesy        }
                            { Instalace procedury pro ukonceni      }
SetUserThird(Addr(pMKonecThird));
                            { Prechod na spravnou menu-obr. pri Dn  }
SetCtrlTab([iDn],Word(tMPridana));
                            { Prechod na menu-obr.sledovani pri 'S' }
SetTransCtrlTab(['S'],iCall1,Word(tMSledovani));
```

Tabulka TransTab je implicitně plněna pro klávesu "ESC" hodnotou iRet, a proto bude po jejím stisku v menu-obrazovce "Sledování" nová hodnota indexu aktuální menu-obrazovky vyzvednuta ze zásobníku menu-obrazovek **StackMenu**. Dojde tak k návratu do menu-obrazovky, ve které byla menu-obrazovka "Sledování" otevřena.

V těle definiční procedury **pMSledovani** je třeba vyřadit z podmínky **if ChangeMenu** přiřazení hodnoty do proměnné DispStr, neboť chceme zobrazovat měnící se veličiny a hodnotu je třeba aktualizovat i v době, kdy nedochází ke změně menu-obrazovky.

Funkce **RealStrDec** a **IntegerStrDec** slouží k formátovanému převodu čísel na řetězce a jsou deklarovány v jednotce NumToStr, kterou doplníme za klauzuli uses:

```
uses
    NumToStr,
```

Sledované veličiny jsou uloženy v globálních proměnných **SledRealVel** a **SledIntVel**.

```
const
    SledRealVel : Real=10.5;{ Sledovana velicina real              }
    SledIntVel : Integer=221;{Sledovana velicina integer           }
```

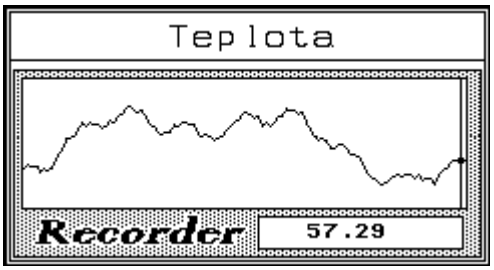
Změny těchto veličin budeme simulovat ve smyčce hlavního procesu. Hlavní program bude tedy doplněn:

```
repeat                  { Smycka hlavniho procesu                  }
  if SledRealVel>120    { simuluje zmeny sledovanych velicin       }
    then SledRealVel:=-120
    else SledRealVel:=SledRealVel+11.24;

  if SledIntVel>93
    then SledIntVel:=1
    else SledIntVel:=SledIntVel+3;

    Wait(4);            { a ceka                                   }
  until FlgEnd;         { na nastaveni priznaku ukonceni programu  }
```

Zároveň byly v tomto příkladu instalovány v těle procedury InitializeAppFonts dva nové uživatelské fonty používané v nové menu-obrazovce.

9. Uživatelská bitmapa v paměti RWM a Zapisovač

Uživatel může vytvářet vlastní bitmapy, které budou obsahovat pozadí menu-obrazovek (např. prázdnou tabulku, ilustrační obrázek nebo symboly atd.). Neboť jsou binární obrazy těchto bitmap uloženy v paměti EPROM terminálu, nelze za běhu programu měnit jejich podobu. Někdy však může být účelné podobu bitmap za běhu programu modifikovat. Nevystačíme přitom pouze s možností jejich vyměňování prostřednictvím volby indexu v poli ukazatelů na uživatelské bitmapy. V takovém případě je vhodné **bitmapu uložit do paměti RWM**. Tam může být kdykoli modifikována a to i bez ohledu na to, jestli je použita v právě aktuální menu-obrazovce. Bitmapu v paměti RWM implementuje objekt **tGraphicArea** deklarovaný jednotce uGrObj. Ve stejné jednotce najdeme i objekt **tGrRecorder**, který slouží ke kreslení pohyblivých grafů ve stylu zapisovače.



obr. S - zapisovač

Do demonstračního programu nyní přidáme menu-obrazovku se zapisovačem zapisujícím měnící se veličinu (soubory **T10DEMO4.PAS** a **MENU4.PAS** přílohy). Vzhled této menu-obrazovky je na Obr. S. Změny zapisované veličiny budeme simulovat ve smyčce hlavního programu. Zapisovač poběží stále, bez ohledu na aktuálně vybranou menu-obrazovku. Novou menu-obrazovku zařadíme za menu-obrazovku "Sledování". Bude možné do ní přejít z menu-obrazovky "Sledování" klávesou "šipka dolů". Při dalším stisku klávesy "šipka dolů" se vrátíme zpět do menu-obrazovky "Sledování". Téhož bude možné dosáhnout klávesou "šipka nahoru". Z těchto obou menu-obrazovek vznikne vnořené menu. Z něj se bude možné vrátit stiskem klávesy "ESC" na vyšší úroveň hierarchie do menu-obrazovky odkud byla menu-obrazovka "Sledování" klávesou "S" otevřena. Definiční procedura nové menu-obrazovky:

```
procedure pMZapisovac(P: pMenuGr); far; { Menu-obrazovka se
zapisovacem }
begin
  with P^ do
    begin                    { Formatovany text pro vypis           }
    DispStr   :=zESC+ 'F1,80,5;'     +'Teplota'+
                zESC+ 'F0,150,108;'  +RealStrDec(Teplota,5,2);
    if ChangeMenu then       { Doslo k zmene Menu-obrazovky ?       }
    begin
      BkBmp     :=2;         { Nastaveni pozadi - bitmapa v RWM     }
                             { Formatovany text pro help            }
      HlpStr    :=zESC+'F1,60,12;'  +'NÁPOVĚDA'+
                  zESC+'F0,20,40;'  +'Ukazka zapisovace';
      GraphStr  :='R(1,1,238,126)L(1,25,238,25)';
                             { Prechod na spravnou menu-obr. pri Dn  }
      SetCtrlTab([iDn],Word(tMSledovani));
    end;
  end;
end;
```

Proměnnou DispStr je vypisován pouze nápis "Teplota" a její číselná hodnota. Proměnnou GraphStr je kreslen rámeček kolem obrazovky a čára oddělující nápis

"Teplota". Vše ostatní je obsahem pozadí. Volenému indexu 2 pozadí je přiřazena bitmapa v paměti RWM. Tvoří ji instance objeku tGraphicArea. Spolu s instancí objektu tGrRecorder je deklarována v sekci interface. Sekci interface je třeba dále doplnit o klauzuli uses s jednotkou uGrObj, která deklaruje tyto objekty.

```
interface
uses GO240128;
var MyGrArea : tGraphicArea;{ Bitmapa v pameti RWM                  }
    MyGrRecorder:tGrRecorder;{Zapisovac                             }
```

Proměnná Teplota bude obsahovat zapisovanou teplotu.

```
const
    Teplota  : Real=0;   { Zapisovana teplota                    }
```

Objekty **MyGrArea** a **MyGrRecorder** je třeba inicializovat v proceduře InitializeBMPs.:

```
  MyGrArea.Init(240,128);
  MyGrArea.LoadBkBMP(GetBMPAddr_BMPZap(Lng),Lng);
  MyGrRecorder.Init(MyGrArea);
  MyGrRecorder.DefRecorderRect(8,35,223,99);
  User_BkBMPs[2].AssignBMPAddr(Addr(MyGrArea),Lng);
```

Nejprve je inicializován objekt **MyGrArea** a je do něj natažena bitmapa obsahující pozadí zapisovače. Data bitmapy se nachází v jednotce BMPZap. Pak je inicializován objekt MyGrRecorder, kterému je předán odkaz na grafickou plochu. Jsou definovány rozměry okénka zapisovače. Okénko zapisovače musí být ve směru x zarovnáno na celé byte - tedy X1 = n × 8 a X2 = m × 8 - 1. Adresa objektu MyGrArea je pak uložena na zvolené místo pole ukazatelů na uživatelské bitmapy.

Do plochy MyGrArea můžeme nyní jednak vykreslovat vektorové grafické objekty (bod , úsečka , kružnice atd.) a jednak kreslit pohybující se graf pomocí objektu **MyGrRecorder**. Aby ke kreslení docházelo neustále, nazávisle na aktuální menu-obrazovce, budeme ho provádět spolu se simulací změn teploty ve smyčce hlavního programu:

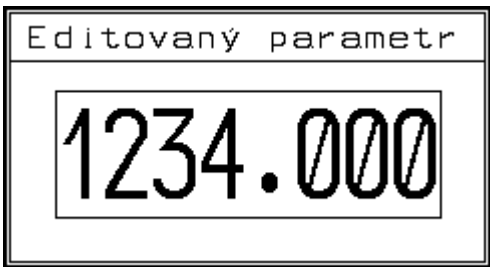
```
...
  repeat                { Smycka hlavniho procesu                 }
...
                        { teplota pro zapisovac                   }
    GetTime(Hour,Minute,Second,Sec100);{Zjisteni casu             }
    Time:=10*Sec100+1000*Second;      { Vypocet casu v ms         }
                      { simulace menici se teploty              }
    Teplota:=70+40*sin(2*3.14159*Time/15000)
               +20*sin(2*3.14159*Time/5000)
               +10*sin(2*3.14159*Time/1500)
               +random(10);
                        { zjisteni rozdilu casu od min. kroku zap.  }
    if Time>=LastTime then DifTime:=Round((Time-LastTime)/cCasMer)
                  else DifTime:=Round((Time+60000-LastTime)/cCasMer);
    LastTime:=Time;
                         { souradnice Y zapisovace v pixelech       }
    YZap:=MyGrRecorder.My2-Round(Teplota/cTeplMer);
    LockKernel;          { Pocatek chranene sekce                   }
    if DifTime>0 then
    begin
     MyGrRecorder.WrDifXToRecorder(DifTime,YZap);{Krok zapisovace   }
    mFillBox(224,35,228,99,b_NOT,MyGrArea.Img);{smaz.osy s pisatkem }
```

```
    mDrawLine(226,35,226,99,b_OR,MyGrArea.Img);{kresleni osy        }
   mDrawBox(225,Yzap-1,227,YZap+1,b_OR,MyGrArea.Img);{kresleni pis. }
     mDrawLine(224,YZap,228,Yzap,b_OR,MyGrArea.Img);
    end;
    UnLockKernel;        { Konec chranene sekce                     }
.........
```

Jak je vidět, je při každém průchodu smyčkou zjištěn čas a vypočten časový rozdíl od minulého průchodu. Časový rozdíl a teplota jsou pak převedeny na vyjádření v počtu pixelů. Vlastní krok zapisovače je proveden voláním metody tGrRecorder.WrDifXToRecorder.Ta posune stávající obsah okénka o zadaný počet pixelů a vykreslí úsečku tvořící nově vykreslenou část průběhu veličiny. Pohybující se pisátko je pak kresleno pomocí procedur pro grafické operace přímo do grafické plochy. Procedury **LockKernel** a **UnLockKernel** ohraničují chráněnou sekci, ve které jádro operačního systému ReTOS nepřepne do jiného procesu.

10. Editace

Objekt menu umožňuje uživateli instalovat do systému menu **uživatelské procedury**. Jsou to procedury UserFirst, User Second a UserThird. První procedura se provádí po definiční proceduře menu-obrazovky před zobrazením, druhá se provádí po zobrazení před určením indexu následující menu-obrazovky a třetí se provádí na závěr cyklu procházeného systémem menu. Instalaci třetí procedury používáme už od prvního příkladu k ukončení programu. Na místo druhé procedury lze instalovat editační proceduru.



obr. T - editace

Systém menu má implementovány procedury pro editaci všech základních číselných typů v různých formátech a stringů včetně několika dalších speciálních procedur (editace hesla, čekání zadanou dobu atd.). Pokud je uživatel postaven před problém editovat na některé menu-obrazovce nějaký parametr, postačí mu v těle definiční procedury zavolat příslušnou metodu objektu menu. V dalším rozšíření příkladu (soubory **T10DEMO5.PAS** a **MENU5.PAS** přílohy) přidáme menu-obrazovku, kde bude možno editovat jeden parametr typu real. Její vzhled je na Obr. T. Definiční procedura menu-obrazovky:

```
procedure pMEditace(P: pMenuGr); far;   { Menu-obrazovka s editaci  }
begin
  with P^ do
  begin
    if ChangeMenu then      { Doslo k zmene Menu-obrazovky ?        }
    begin
      BkBmp     :=-1;       { Nastaveni pozadi                      }
                            { Formatovany text pro vypis            }
      DispStr   :='12345678'+{Pro editacni okenko                    }
                  zESC+ 'F1,10,5;'  +'Editovany parametr'+
                  zESC+ 'F3,24,40';{ Umisteni editacniho okenka     }
                            { Formatovany text pro help             }
      HlpStr    :=zESC+'F1,72,12;'  +'NÁPOVĚDA'+
                  zESC+'F0,60,40;'  +'Po stisku ENTER'+
                  zESC+'F0,40,50;'  +'lze editovat hodnotu'+
                  zESC+'F0,44,60;'  +'parametru v rozsahu'+
                  zESC+'F0,72,70;'  +'100 az 5000.';
```

```
      GraphStr  :='R(1,1,238,126)L(1,25,238,25)R(24,40,216,104)';
      Term^.Flins:=false;   { Nastaveni rezimu prepisovani          }
      SetEditWin(1,1,8,1);  { Nastaveni editacniho okenka           }
                            { Editace realneho cisla                }
      EditReal(Parametr,100,5000,8,3);
      AddTerminateChar(['S']);{Pridani ukoncovaci klav.            }
                            { Prechod na menu-obr.sledovani pri 'S' }
      SetTransCtrlTab(['S'],iCall1,Word(tMSledovani));
    end;
  end;
end;
```

Hodnota editovaného parametru je v proměnné Parametr. Voláním metody **EditReal** se na místo druhé uživatelské procedury (UserSecond) instaluje procedura pro editaci čísel typu real a nastaví se parametry editace: **horní a dolní mez platné hodnoty, délka znakové reprezentace a počet desetinných míst**. Metodou **SetEditWin** se nastaví poloha a rozměry editačního okénka. Nastavením proměnné FlIns objektu terminálu přejdeme do režimu přepis (implicitně je vkládání). Styl výpisu editačního stringu je třeba nastavit na konci DispStr a na jeho začátek vložit příslušný počet znaků, na které je nastaveno editační okénko.

Nová menu-obrazovka je zařazena do řetězce na nejvyšší úrovni hierarchie a lze z ní také otevřít stiskem "S" menu-obrazovku "Sledování".

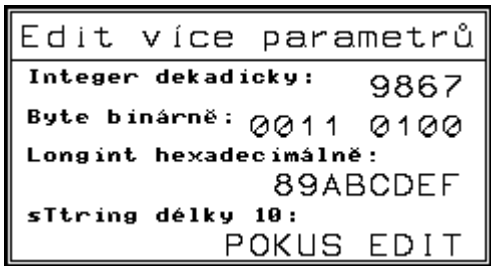
Definiční proceduru úvodní menu-obrazovky jsme dále upravili o volání metody **EditWaitPress**.

```
EditWaitPress(2000);      { 2000ms ceka a pak posle zDn }
```

Touto metodou je na místo druhé uživatelské procedury instalována čekací procedura. Tato procedura, pokud není stisknuta některá klávesa z množiny ukončovacích kláves, po zadané době vloží do proměnné ActChar znak zDn. Tím vyvolá akci, jako by byla stisknuta klávesa "šipka dolů". Pokud tedy nestiskneme žádné tlačítko, zůstane úvodní menu-obrazovka zobrazena pouze dvě sekundy. Po této době se přejde do následující menu-obrazovky "Přidaná".

11. Vícenásobná editace na jedné obrazovce

V praxi je často vhodné editovat na jedné obrazovce více parametrů. Volbu editovaného parametru provádí obsluha stiskem příslušného tlačítka (např. "A","B","C"... ,"1","2","3"... , funkčních kláves "Fx" nebo zvýrazněného písmene v názvu parametru ap.). Tuto situaci lze nejlépe implementovat tak, že vytvoříme příslušný počet menu-obrazovek se stejným vzhledem. V každé z nich bude editován jeden z parametrů. K tomu ještě vytvoříme jednu menu-obrazovku, kde budou všechny parametry pouze zobrazeny. Editační menu-obrazovky budou volány ze zobrazovací menu-obrazovky po stisku příslušné klávesy. Vzhledem k tomu, že je vzhled všech menu-obrazovek shodný, budí výsledek dojem editace více parametrů na jedné obrazovce.



obr. U- vícenásobná editace

V příkladu zařadíme za stávající menu-obrazovku Editace menu-obrazovku s editací čtyř parametrů a to typu integer v dekadickém tvaru, byte v binárním tvaru, longint v hexadecimálním tvaru a string (soubory **T10DEMO6.PAS** a **MENU6.PAS** přílohy). Vzhled menu-obrazovky je na Obr. U.

Definiční procedura menu-obrazovky se zobrazením parametrů:

```
procedure pMEditVice(P: pMenuGr); far;{ Menu-obrazovka s editaci vice
p.}
begin
  with P^ do
  begin
    if ChangeMenu then      { Doslo k zmene Menu-obrazovky ?        }
    begin
      BkBmp     :=-1;       { Nastaveni pozadi                      }
                            { Formatovany text pro vypis            }
      DispStr   :=zESC+'F1,6,5;'   +'Edit vice parametru'+

                  zESC+'F0,10,30;' +'Integer dekadicky:' +
                  zESC+'F1,180,30;'+IntegerStrDec(VicEdInt,4)+

                  zESC+'F0,10,50;' +'Byte binarne:' +
                  zESC+'F1,120,50;'+ByteStrBin(VicEdByte)+

                  zESC+'F0,10,70;' +'Longint hexadecimalne:' +
                  zESC+'F1,132,80;'+LongIntStrHex(VicEdLong)+

                  zESC+'F0,10,100;' +'sTtring delky 10:' +
                  zESC+'F1,108,110;'+VicEdString;
                            { Formatovany text pro help             }
                                 {12345678901234567890123456789O}
      HlpStr    :=zESC+'F1,72,12;'  +'NÁPOVĚDA'+
                  zESC+'F0,52,40;'  +'Stisknete I,B,L,T'+
                  zESC+'F0,28,50;'  +'pro editaci prislusneho'+
                  zESC+'F0,80,60;'  +'parametru.';
      GraphStr  :='R(1,1,238,126)L(1,25,238,25)';
      Term^.Flins:=false;   { Nastaveni rezimu prepisovani          }
                            { Pridani ukoncovaci klavesy            }
      AddTerminateChar(['S','I','B','L','T']);
                            { Prechod na menu-obr.sledovani pri 'S' }
      SetTransCtrlTab(['S'],iCall1,Word(tMSledovani));
      SetTransCtrlTab(['I'],iCall2,Word(tMEditViceI));
      SetTransCtrlTab(['B'],iCall3,Word(tMEditViceB));
      SetTransCtrlTab(['L'],iCall4,Word(tMEditViceL));
      SetTransCtrlTab(['T'],iCall5,Word(tMEditViceT));
      SetCtrlTab([iDn],Word(tMKonec)) {Dn                           }
    end;
  end;
end;
```

V menu-obrazovce zobrazíme všechny čtyři parametry včetně doprovodného textu. O zvolené klávesy "I","B","L","T" rozšíříme množinu ukončovacích kláves a nastavíme TransTab a CtrlTab na volání příslušných editačních menu-obrazovek. Je též třeba zajistit správné přechody při stisku "šipka dolů" a "šipka nahoru". Je vidět, že i v této menu-obrazovce jsme zachovali možnost volat stiskem "S" menu-obrazovku "Sledování".

Čtyři editační menu-obrazovky budou velmi podobné. Pro menu-obrazovku s editací čísla integer bude definiční procedura:

```
procedure pMEditViceI(P: pMenuGr); far;{ Menu-obrazovka s editaci
vice p.}
begin
  with P^ do
  begin
    if ChangeMenu then      { Doslo k zmene Menu-obrazovky ?        }
    begin
      BkBmp     :=-1;       { Nastaveni pozadi                      }
                            { Formatovany text pro vypis            }
      DispStr   :='1234'+   { Pro editacni okenko                   }
                  zESC+'F1,6,5;'   +'Edit vice parametru'+
                  zESC+'F0,10,50;' +'Byte binarne:' +
                  zESC+'F1,120,50;'+ByteStrBin(VicEdByte)+
                  zESC+'F0,10,70;' +'Longint hexadecimalne:' +
                  zESC+'F1,132,80;'+LongIntStrHex(VicEdLong)+
                  zESC+'F0,10,100;' +'sTtring delky 10:' +
                  zESC+'F1,108,110;'+VicEdString+

                  zESC+'F0,10,30;' +'Integer dekadicky:' +
                  zESC+'F1,180,30;';
                            { Formatovany text pro help             }
                                 {12345678901234567890123456789O}
      HlpStr    :=zESC+'F1,72,12;'  +'NÁPOVĚDA'+
                  zESC+'F0,8,40;'   +'Zadej cislo -9999 az 9999.';
      GraphStr  :='R(1,1,238,126)L(1,25,238,25)';
      EditEnter :=false;
      SetEditWin(1,1,4,1);  { Nastaveni editacniho okenka           }
                            { Editace integer                       }
      EditInteger(VicEdInt,-9999,9999,4);
                            { Nastaveni ukoncovacich klaves         }
      SetTerminateChar([zCr,zESC]);
      SetTransTab([zCr],iRet);
    end;
  end;
end;
```

Na začátek DispStr jsou vloženy znaky pro editační okénko. Nastavení stylu editované veličiny je na konci DispStr. Volá se metoda **EditInteger** pro instalaci příslušné editační procedury a nastavení parametrů editace. Množina ukončovacích kláves TerminateChar je nastavena pouze na klávesy "ENTER" (zCR) a "ESC" (zESC). Jiné klávesy tedy nebude menu-obrazovka akceptovat jako ukončovací. Pro klávesu "ENTER" je nastavena TransTab na návrat z podmenu (iRet). Pro klávesu "ESC" je toto nastavení implicitní. Proměnná **EditEnter** je nastavena na false. Implicitní hodnota této proměnné je true. Pokud je hodnota EditEnter true, čeká menu-obrazovka před zahájením editace na stisk klávesy z množiny VerifyChar. Tato množina obsahuje klávesy pro zahájení a platné ukončení editace. Implicitně obsahuje jedinou klávesu, a to zCR (klávesa "ENTER"). Tento případ nastává u menu-obrazovky s jednou editací, kdy se editace zahajuje stiskem klávesy "ENTER". Pokud je hodnota proměnné EditEnter false, na klávesu z VerifyChar se nečeká a editace je zahájena ihned po vstupu do menu-obrazovky.

12. Vytváření menu-obrazovek v grafické vrstvě

V jednotce **G240x128** resp. **G128c64** jsou definovány funkce, které umožňují výpis textů a vykreslování geometrických objektů přímo VideoRWM. Jsou to tyto funkce:

Vymazání VideoRWM:
procedure mClear(var vVideoRWM :tVideoRWM);

Vykreslení bodu o souřadnicích [X,Y] operací Rop:
procedure mDrawPoint (X, Y : Integer; Rop : tVideoROP;
var vVideoRWM : tVideoRWM);

Vykreslení čáry o z body [X1,Y1] do bodu [X2,Y2] operací Rop:
procedure mDrawLine(X1, Y1 ,X2 ,Y2 : Integer; Rop : tVideoROP;
var vVideoRWM : tVideoRWM);

Vykreslení kružnici o středu [Xr,Yr] a poloměru R operací Rop:
procedure mDrawCircle(Xr, Yr, R : Integer; Rop : tVideoROP;
var vVideoRWM : tVideoRWM);

Vykreslení obdélníku s rohy [X1,Y1], [X2,Y2] operací Rop:
procedure mDrawBox(X1, Y1, X2, Y2 : Integer; Rop : tVideoROP;
var vVideoRWM : tVideoRWM);

Vykreslení vyplněného obdélníku s rohy [X1,Y1],[X2,Y2] operací Rop:
procedure mFillBox(X1, Y1, X2, Y2 : Integer; Rop : tVideoROP;
var vVideoRWM : tVideoRWM);

Vypsaní textového řetězce Str operací Rop fontem FIndex na souřadnice [X,Y]:
procedure mTextOutXY(X, Y : Integer; Str : tVideoStr; Rop : tVideoROP;
var vVideoRWM : tVideoRWM; Findex : Integer);

Parametr **Rop** určuje binární operaci mezi původním a novým obsahem VideoRWM, tedy zda bude použita operace NOT, OR nebo XOR. Podrobnější informace o uvedených funkcích je možné nalézt v dokumentaci G240128.

S výše uvedenými funkcemi je stále nutné používat pro editaci řetězec DispStr, ale ostatní řetězce (GraphStr a HlpStr) nemusí být vůbec použity. Na jedné obrazovce je možné umístit mnohem více informací a editovatelných údajů. Před skokem do editace se vymaže oblast VideoRWM, ve které bude probíhat samotná editace

Volbu editovaného údaje je vhodné implementovat například jako výběr pomocí určené klávesy (HotKey) nebo pomocí kurzoru, kterým se pohybuje po obrazovce klávesami šipek. Následující příklad ukazuje obě uvedené možnost – viz. soubory **T10Demo7.pas** a **Menu7.pas** přílohy.

V příkladu jsou definovány dvě oblasti pro přímé vykreslování do videoRWM:

```
var   { bitmapy       }
  BkArea   : tGraphicArea; { Pozadi beznych obrazovek  }
  HlpArea  : tGraphicArea; { Pozadi obrazovek s helpem }
```

Při inicializaci bitmap přiřadíme adresu s logem do pole ukazatelů na uživatelské bitmapy. Také provedeme inicializaci bitmap pro nápovědu a běžnou obrazovku.

```
procedure InitializeBitmaps;
var
  wSize: Word;
begin
  { Bitmapa s logem a nazvem programu }
  User_BkBMPs[cBmpLogo].AssignBMPAddr(GetBMPAddr_Logo(wSize), wSize);

  { Bitmapa pozadi beznych obrazovek }
  BkArea.Init(240, 128);
  wSize := SizeOf(BkArea);
  User_BkBMPs[cBmpBkArea].AssignBMPAddr(@BkArea, wSize);

  { Bitmapa pozadi obrazovek s helpem }
  HlpArea.Init(240, 128);
  wSize := SizeOf(HlpArea);
  User_BkBMPs[cBmpHelpArea].AssignBMPAddr(@HlpArea, wSize);

  { Vykresleni horni a spodni radky helpu }
  PrepareHelpBitmap;
end;
```

Zároveň je možné předpřipravit část bitmapy, která se v průběhu programu už nebude měnit, například vykreslení titulku do bitmapy s nápovědou.

```
procedure PrepareHelpBitmap;
begin
  HlpArea.ClearImage;
  { Horni radka }
  mTextOutXY(86, 3, 'Nápověda', b_OR, HlpArea.Img, 0);
  mFillBox (1, 1,  238, 12, b_XOR, HlpArea.Img);
  mDrawLine(1, 14, 238, 14,b_OR,HlpArea.Img);
  { Spodni radka }
  mTextOutXY(180, 117, 'Esc=zpět', b_OR, HlpArea.Img, 0);
  mDrawLine(1, 113, 238, 113, b_OR,HlpArea.Img);
  mFillBox (1, 115, 238, 126, b_XOR, HlpArea.Img);
end;
```

Spuštění uživatelského menu bude vypadat následovně (pro zjednodušení uvádíme jen část podmíněného překladu pro verzi se simulátorem terminálu na PC):

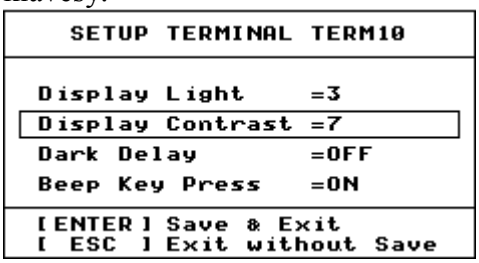
```
procedure UserMenu;
var
  pMyTerm : pTermT10;
  pMyMenu : pMenuGr;

begin
  InitializeAppFonts;
  InitializeBitmaps;

  DispStr      := '';
  HlpStr       := '';
  GraphStr     := '';
  HlpGraphStr  := '';
  BkBmp        := cBmpNone;
```

```
  HlpBmp       := cBmpNone;
  PMyTerm := New(pTermT10, Init(New(PSimDispT10,
             Init(nil,26,9,AdrTerm,true,0,0,2,2,265,165,False)),
             New(PSimKeybT10,Init(nil,20,AdrTerm,EndMenu)),
             AdrTerm,nil,nil));

  pMyTerm^.SetTDispParam(sParTerm); { Nastaveni parametru terminalu }
  pMyMenu:=New(pMenuGr,Init(@ProcPtrArray, ord(high(tSelector)),
           DispStr, HlpStr, GraphStr, @BkBmp, pMyTerm, '', 2));

  pMyMenu^.SetHelpVar(HlpGraphStr, HlpBmp);
  HlpBmp := cBmpHelpArea;

  InitSetUpT10(pMyTerm, @TermSetup, 4, 4, 15, 7);

  InitRunMenu(pMyMenu,
              MenuName,MenuStk,MenuSPrio,MenuDPrio,
              TermName,TermStk,TermSPrio,TermDPrio,
              2,2, edNop,edNop, EndMenu);   { Spusteni menu }

  pMyMenu^.MenuIntEnable := False;   { Zakaz MenuInterruptu }
  pMyMenu^.SetupKeyChar  := zF10;    { Nastaveni klavesy pro setup }
  pMyTerm^.FlIns         := false;   { Nastaveni rezimu prepisovani }
end;
```

Oproti předchozím příkladům je zde nové použití obrazovky **Nastavení terminálu**. Tato obrazovka je obsažena v jednotce **SetUpT10** a její inicializaci v tomto příkladu zajišťuje příkaz

```
InitSetUpT10(pMyTerm, @TermSetup, 4, 4, 15, 7);
```

pomocí kterého se nastaví jas a kontrast dislpeje a doba, za kterou se displej přepne do úsporného režimu. Poslední parametr povoluje zvukovou signalizaci při stisknutí klávesy.



Obrazovka nastavení terminálu

```
pMyMenu^.SetupKeyChar  := zF10;
```

Tato proměnná udává, jakou klávesou se vyvolá menu **Nastavení terminálu**.

Příklad obsahuje pouze pět menu-obrazovek. První z nich zajišťuje zobrazení loga. Po dvou sekundách případně po stisku klávesy ENTER se přejde do obrazovky Param1.

```
procedure pLogo(P: pMenuGr); far;
begin
  with P^ do
    if ChangeMenu then
    begin
      BkBmp := cBmpLogo;
      EditWaitPress(2000);  {po 2 sec nebo Enter prejde do tParam1}
      SetTransCtrlTab([zCr], iJmp1, (Word(tParam1)));
    end;
end;
```

Druhá obrazovka slouží pro výpis seznamu parametrů a jejich hodnot. Editovatelné parametry Params1[1..5] a Params2[1..5] jsou zobrazeny jen při prvním vstupu do procedury. Pokud je potřeba zobrazovat proměnný parametr (v našem příkladě například čas), je nutné při každém průběhu procedurou vymazat oblast, kam se tento parametr zapisuje a pak zapsat novou hodnotu parametru.

Jako ukončovací klávesy byly zvoleny PageUp a PageDn, které slouží k přechodu na obrazovku **Param2**. To je pohodlné v případě simulátoru terminálu, ale na skutečném terminálu Term10 to odpovídá kombinaci SHIFT + Up a SHIFT + Dn a je pohodlnější používat samostatné klávesy šipek. V našem příkladě jsou klávesy šipek použity k pohybu po menu **Param2**.

Při stisku klávesy ‘A’ .. ‘J’ se ve funkci **pParam1Third** zjistí který řádek se bude editovat (parametr **iEditLine**) a provede se skok do obrazovky **EditParam1**.

Obrazovka Param1

```
Const   {parametry editovatelne v okne Param1}
  Params1    : array[1..5] of integer     = (0,0,0,0,0);
  Params2    : array[1..5] of real        = (0,0,0,0,0);

procedure pParam1Third(P: pMenuGr); far;
begin
  if p^.ActChar in ['A'..'J'] then
  begin
    iEditLine := pos(p^.ActChar,'ABCDEFGHIJ');
    p^.StackMenu.CondPushMenu(word(tParam1));  {iCall}
    p^.SetActMenu(word(tEditParam1));
  end;
end;

procedure pParam1(P: pMenuGr); far;
var
  Hour, Min, Sec, W : word;
begin
 with p^ do
 begin
  if ChangeMenu then
  begin
   BkBmp := cBmpBkArea;
   BkArea.ClearImage;
   DrawScrTitle(35, 'Editace pomoci HotKey');
   DrawStatusBar('zvol: A B C D E F G H I J');

   mTextOutXY(30, 19, 'Param A', b_OR, BkArea.Img, 1);
   mTextOutXY(30, 28, 'Param B', b_OR, BkArea.Img, 1);
   mTextOutXY(30, 37, 'Param C', b_OR, BkArea.Img, 1);
   mTextOutXY(30, 46, 'Param D', b_OR, BkArea.Img, 1);
   mTextOutXY(30, 55, 'Param E', b_OR, BkArea.Img, 1);
   mTextOutXY(30, 64, 'Param F', b_OR, BkArea.Img, 1);
   mTextOutXY(30, 73, 'Param G', b_OR, BkArea.Img, 1);
```

```
   mTextOutXY(30, 82, 'Param H', b_OR, BkArea.Img, 1);
   mTextOutXY(30, 91, 'Param I', b_OR, BkArea.Img, 1);
   mTextOutXY(30,100, 'Param J', b_OR, BkArea.Img, 1);

   mTextOutXY(90,19,IntegerStrDec(Params1[1],0),b_OR,BkArea.Img,1);
   mTextOutXY(90,28,IntegerStrDec(Params1[2],0),b_OR,BkArea.Img,1);
   mTextOutXY(90,37,IntegerStrDec(Params1[3],0),b_OR,BkArea.Img,1);
   mTextOutXY(90,46,IntegerStrDec(Params1[4],0),b_OR,BkArea.Img,1);
   mTextOutXY(90,55,IntegerStrDec(Params1[5],0),b_OR,BkArea.Img, 1);
   mTextOutXY(90,64,RealStrDec(Params2[1],8,8),b_OR,BkArea.Img, 1);
   mTextOutXY(90,73,RealStrDec(Params2[2],8,8),b_OR,BkArea.Img, 1);
   mTextOutXY(90,82,RealStrDec(Params2[3],8,8),b_OR,BkArea.Img, 1);
   mTextOutXY(90,91,RealStrDec(Params2[4],8,8),b_OR,BkArea.Img, 1);
   mTextOutXY(90,100,RealStrDec(Params2[5],8,8),b_OR,BkArea.Img, 1);
   ClearHelpBitmap;
   DrawHelpText(1,'Stisknutim klavesy A - J se prejde');
   DrawHelpText(2,'do editace prislusneho parametru');

   SetTerminateChar([zPgUp, zPgDn]);
   SetTransCtrlTab([zPgUp, zPgDn], iJmp1, word(tParam2));
   SetUserThird(@pParam1Third);
  end;

  GetTime(Hour, Min, Sec, W);
  LockKernel;
  mFillBox(180, 19, 230, 26, b_NOT,BkArea.img);
  mTextOutXY(180,19, WordStrDec(Hour,0)+':'+
                     WordStrDec(Min,0)+':'+
                     WordStrDec(Sec,0), b_OR, BkArea.Img, 1);
  UnlockKernel;
 end;
end;
```

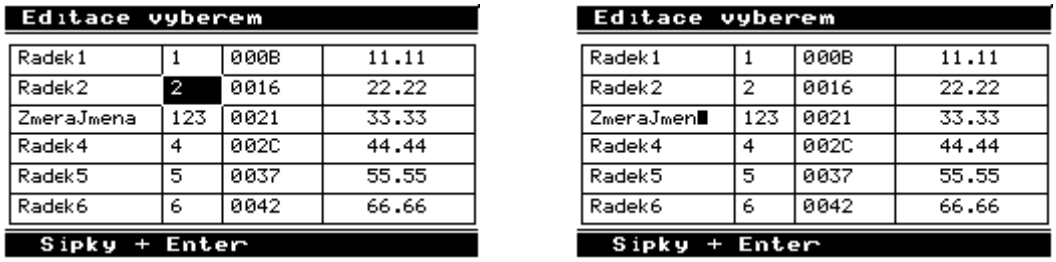
Ve funkci pEditParam1 se podle čísla editovaného parametru **iEditLine** vygeneruje **DispStr** (umístění kurzoru na začátku editace), nadefinuje se velikost editačního okénka, vymaže se oblast pro editaci, vykreslí se orámování kolem editovaného parametru a zavolá se příslušná editační funkce. Je možné změnit bitmapu s nápovědou.

```
procedure pEditParam1(P: pMenuGr); far;
var
  PosY : byte;
begin
  with P^ do
  begin
    if ChangeMenu then
    begin
      PosY:=10+9*iEditLine;

      DispStr:=zESC+'F1,90,'+ByteStrDec(PosY,0)+';';
      EditEnter:=false;
      ClearHelpBitmap;

      if iEditLine <=5 then
      begin
        SetEditWin(1,1,5,1);
        mFillBox(90, PosY, 118, PosY+7, b_NOT,BkArea.img);
        mDrawBox(88, PosY-1, 120, PosY+8, b_OR,BkArea.img);
        EditINteger(Params1[iEditLine],0,10000,0);

        DrawHelpText(1,'Zadejte cele cislo v rozsahu:');
        DrawHelpText(2,'0 .. 10000');
      end else
      begin
        SetEditWin(1,1,8,1);
```

```
        mFillBox(90, PosY, 138, PosY+7, b_NOT,BkArea.img);
        mDrawBox(88, PosY-1, 140, PosY+8, b_OR,BkArea.img);
        EditReal(Params2[iEditLine-5],-1000,1000,8,8);

        DrawHelpText(1,'Zadejte realne cislo v rozsahu:');
        DrawHelpText(2,'-1000 .. 1000');
      end;
      SetTerminateChar([zEsc,zCr]);
      SetTransTab([zEsc,zCr], iRet);
    end;
  end;
end;
```

Funkce **pParam2** slouží k zobrazení tabulky hodnot a editačního okénka (kurzoru). Funkce **pParam2Third** zajišťuji změnu pozice kurzoru na základě kláves šipek. Při změně pozice kurzoru je zároveň nastaven příznak ReDrawMenu, který zajišťuje překreslení tabulky hodnot ve funkci **pParam2**.

Ukončovací znak **zCr** způsobí přechod do editace zvoleného parametru.



Editace vyberem			
Radek1	1	000B	11.11
Radek2	2	0016	22.22
ZmeraJmena	123	0021	33.33
Radek4	4	002C	44.44
Radek5	5	0037	55.55
Radek6	6	0042	66.66
Sipky + Enter			

Editace vyberem			
Radek1	1	000B	11.11
Radek2	2	0016	22.22
ZmeraJmen	123	0021	33.33
Radek4	4	002C	44.44
Radek5	5	0037	55.55
Radek6	6	0042	66.66
Sipky + Enter			

Obrazovka Param2 – 1) výběr parametru pro editaci, 2) editace

```
type
  tLine = record   {jeden radek tabulky}
    ParamString : string[10];
    ParamByte   : byte;
    ParamWord   : word;
    ParamReal   : real;
  end;

const
 Table:array[1..6] of tLine=( {parametry editovatelne v okne Param2}
    (ParamString:'Radek1';ParamByte:1;ParamWord:11;ParamReal:11.11),
    (ParamString:'Radek2';ParamByte:2;ParamWord:22;ParamReal:22.22),
    (ParamString:'Radek3';ParamByte:3;ParamWord:33;ParamReal:33.33),
    (ParamString:'Radek4';ParamByte:4;ParamWord:44;ParamReal:44.44),
    (ParamString:'Radek5';ParamByte:5;ParamWord:55;ParamReal:55.55),
    (ParamString:'Radek6';ParamByte:6;ParamWord:66;ParamReal:66.66));

const
  xPos:array[0..4] of byte=(3,80,110,160,236); { rozdeleni tabulky }
  xLen:array[0..3] of byte=(75,18,58,74);

procedure pParam2Third(P: pMenuGr); far;
begin
  with P^ do
  begin
    case ActChar of
      zUp:
        begin
          if iEditPosY>1 then dec(iEditPosY)
          else iEditPosY := 6;
          ReDrawMenu := true;
        end;
```

```
      zDn:
        begin
          if iEditPosY<6 then inc(iEditPosY)
          else iEditPosY := 1;
          ReDrawMenu := true;
        end;
      zRi:
        begin
          if iEditPosX<4 then inc(iEditPosX)
          else iEditPosX := 1;
          ReDrawMenu := true;
        end;
      zLe:
        begin
          if iEditPosX>1 then dec(iEditPosX)
          else iEditPosX := 4;
          ReDrawMenu := true;
        end;
    end;
  end;
end;


procedure pParam2(P: pMenuGr); far;
var
  i : byte;

begin
 with P^ do
 begin
  if ChangeMenu or ReDrawMenu then
  begin
   ReDrawMenu:=false;
   LockKernel;
   BkArea.ClearImage;

   DrawScrTitle(10, 'Editace vyberem');
   DrawStatusBar('  Sipky + Enter');

   for i:=0 to 4 do
     mDrawLine(xPos[i], 20, xPos[i],110,b_OR,BkArea.img);
   for i:=0 to 7 do
     mDrawLine(4, 20+15*i, 236, 20+15*i,b_OR,BkArea.img);
   for i:=1 to 6 do
   begin
    mTextOutXY(xPos[0]+5,8+15*i,Table[i].ParamString,b_OR,
            BkArea.Img, 1);
    mTextOutXY(xPos[1]+5,8+15*i,ByteStrDec(Table[i].ParamByte,0),
            b_OR, BkArea.Img, 1);
    mTextOutXY(xPos[2]+5,8+15*i,WordStrHex(Table[i].ParamWord),b_OR,
            BkArea.Img, 1);
    mTextOutXY(xPos[3]+5,8+15*i,RealStrDec(Table[i].ParamReal,8,2),
            b_OR, BkArea.Img, 1);
   end;
   mFillBox(xPos[iEditPosX-1],5+15*iEditPosY,xPos[iEditPosX],
            20+15*iEditPosY,b_XOR,BkArea.img);
   UnLockKernel;
   if ChangeMenu then
   begin
    ClearHelpBitmap;
    DrawHelpText(1,'V tabulce se pohybuje pomoci klaves');
    DrawHelpText(2,'sipek, stisknutim ENTER je mozne prejit');
    DrawHelpText(3,'do editace');
   end;

   SetUserThird(@pParam2Third);
   SetTerminateChar([zCr, zPgUp, zPgDn]);
```

```
   SetTransCtrlTab([zCr], iCall1, word(tEditParam2));
   SetTransCtrlTab([zPgUp, zPgDn], iJmp1, word(tParam1));
  end;
 end;
end;
```

Funkce **pEditParam2** slouží k editaci tabulky parametrů **Table[]**. Je principálně stejná jako funkce **pEditParam1**.

```
procedure pEditParam2(P: pMenuGr); far;
begin
  with P^ do
    begin
      mFillBox(xPos[iEditPosX-1], 5+15*iEditPosY, xPos[iEditPosX],
            20+15*iEditPosY, b_NOT, BkArea.img);
      mDrawBox(xPos[iEditPosX-1], 5+15*iEditPosY, xPos[iEditPosX],
            20+15*iEditPosY,  b_OR, BkArea.img);
      DispStr:=zESC+'F2,'+ByteStrDec(xPos[iEditPosX-1]+5,0)+','+
            ByteStrDec(8+15*iEditPosY,0)+';';
      ClearHelpBitmap;
      case iEditPosX of
        1:begin  {editace stringu}
            SetEditWin(1,1,10,1);
            EditString(Table[iEditPosY].ParamString);
            DrawHelpText(1,'Zadejte text o delce max. 10 znaku');
          end;
        2:begin  {editace byte}
            SetEditWin(1,1,3,1);
            EditByte(Table[iEditPosY].ParamByte,0,255,0);
            DrawHelpText(1,'Zadejte cele cislo 0..255');
          end;
        3:begin  {editace word - hexa}
            SetEditWin(1,1,4,1);
            EditWordHex(Table[iEditPosY].ParamWord);
            DrawHelpText(1,'Zadejte cele cislo 0h..FFFFh');
          end;
        4:begin  {editace real}
            SetEditWin(1,1,8,1);
            EditReal(Table[iEditPosY].ParamReal,0,1E6,8,2);
            DrawHelpText(1,'Zadejte desetinne cislo 0..1E6');
          end;
     end;
    EditEnter:=false;
    SetTerminateChar([zEsc,zCr]);
    SetTransTab([zCr, zEsc], iRet);
    end;
end;
```

13. Zavedení aplikace do terminálu TERM10

Máme-li odladěnu aplikaci na počítači PC, můžeme ji zavést do terminálu TERM10. K tomu je třeba si uvědomit několik rozdílů mezi PC a KitV40.

Nejnižší vrstvu programového vybavení umožňující obsluhovat fyzická zařízení tvoří v KitV40 MCP BIOS, který má některé odlišnosti od BIOSu PC. Program pro PC je spouštěn z DOSu a může využívat jeho služeb. V KitV40 se DOS nezavádí, ale spouští se namísto toho přímo aplikační program. Ten pak může využívat pouze služby MCP BIOSu a popřípadě dalších přídavných BIOSů (např. grafické karty VGA). Z těchto důvodů je třeba při sestavování programu použít jiné knihovny, včetně systémové (obvykle v TURBO.TPL).

Na rozdíl od PC, kde se kód i data uživatelského programu nacházejí v paměti RWM, je v KitV40 třeba kód umístit do paměti EPROM. Kód uživatelského programu se proto nachází v adresním prostoru přídavných ROM modulů.

Používáme-li fyzický terminál TERM10 je třeba inicializovat příslušné objekty terminálu a nikoliv objekty jeho simulátoru na PC.

V KitV40 se data nacházejí v zálohované paměti RWM, a proto je třeba všechny proměnné před prvním použitím důsledně inicializovat hodnotou. Ačkoli je použití neinicializované proměnné obecně programátorskou chybou, nemusí se při ladění programu v PC projevit, neboť v PC je před spuštěním programu vynulován datový segment.

Program pro fyzický terminál sestavíme v jiném adresáři než jsme ladili program pro PC. Je třeba nastavit cesty na knihovny pro KitV40 a zkopírovat do tohoto adresáře soubor TURBO.TPL pro KitV40. V Option - Compiler zapneme generování informací pro ladění (nebo direktiva {$D+}) a v Option - Linker zapneme generování mapy v detailním tvaru. Ve zdrojových textech programů jsme použili podmíněné překlady tam, kde bylo třeba odlišných verzí pro PC a KitV40. Nastavení těchto přepínačů se nachází v souboru SETS.INC. Překlad pro KitV40 bude mít následující nastavení.

```
{ define SimPC}     { verse pocitac PC s monitorem a klavesnici }
{ define VerPc}     { verse Pc                                  }
{$define VerMc}     { verse Mc                                  }

{$define Deb}       { obecny debug na libovolne platforme       }
{$ifdef Deb}{-------------------------------------------------}
  {$ifdef SimPC}
    {$define DebGraph}
  {$endif}
  {$ifdef VerMc}    { verse pocitac KIT V40 + hardware          }
     { define Deb_Crt}
  {$endif}
{$endif} {-------------------------------------------------}
```

Spuštěním překladu vygenerujeme soubor *.EXE, který však nelze spouštět na PC. Z něj je pak třeba vygenerovat binární obsah paměti EPROM.

K vytvoření binárního obsahu paměti EPROM a jeho zavedení do paměti KitV40 slouží program ReTOS Debugger.

Po spuštění ReTOS Debuggeru nejprve zadáme jméno aplikace (jméno příslušného souboru *.EXE) pomocí příkazu Set file name v menu Place. Není - li k přístupná mapa aplikace *.MAP nebo není-li aktuální, zobrazí se chybové hlášení. V tom případě je třeba opustit ReTOS Debugger a program znovu přeložit s nastavením pro generování mapy. Pak přistoupíme k umístění aplikace do adresního prostoru KitV40. Provádí se příkazem Memory design v menu Place. Umístění aplikace se skládá z jednotlivých pamětí. Každá paměť má jméno, zaváděcí adresu, délku, seznam modulů a typ paměti (RAM nebo ROM). V Okně Memory list, které se otevře po zadání příkazu Memory design lze vkládat (Insert), rušit (Delete) nebo editovat (Edit) jednotlivé paměti. Po zadání Insert nebo Edit se otevře okno Edit Memory, ve kterém lze měnit nastavení pamětí. U každé paměti lze zadat její jméno (bude se shodovat s názvem souboru *.BIN generovaného pro ROM paměti), zaváděcí adresu Segment - zadává se v 16 bitových paragrafech a odpovídá tak hodnotě segmentačních registrů a délku paměti - zadává se v bytech. Paměti je třeba umísťovat do adresního prostoru na adresy, kam jsou mapovány fyzické paměti KitV40, přičemž je třeba mít na zřeteli, že veškerá paměť není k dispozici uživateli.

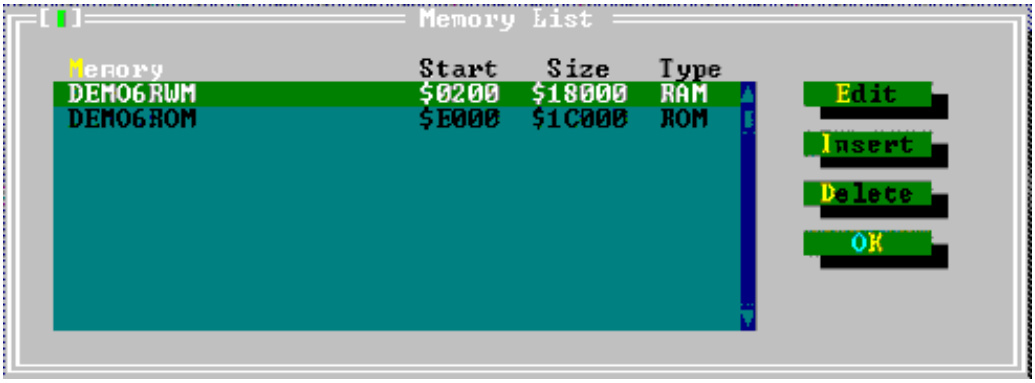
Paměti ROM představují externí ROM moduly. Oblast těchto modulů je prohledávána po spuštění počítače po 2kB blocích a je-li nalezen externí modul, provede se volání jeho inicializační části. Zaváděcí adresa ROM paměti musí tedy být celistvým násobkem 2kB (0080 pro hodnotu segmentačních registrů). Délka externího modulu je pak celistvým násobkem 512 bytů a je shora omezena na téměř 128kB (020000h).

Obsazení adresního prostoru KitV40:

000000h	000AFFh	RWM	Vektory přerušení, BIOS data...
000B00h	max.07FFFFh (podle velikosti osazené paměti RWM)		Zálohovaná paměť RWM pro uživatelský program
min.080000h (podle velikosti osazené pamětí EPROM)	09FFFFh	EPROM	Volný prostor paměti EPROM pro uživatelský program.
0A0000h	0C7FFFFh		Reservovaný prostor pro přídavné adaptéry (VGA).
0C80000h (podle velikosti osazené pamětí EPROM)	0FDFFFh	EPROM	Volný prostor paměti EPROM pro uživatelský program.
0FE000h	0FFFFFh		MCP BIOS

V okně Edit memory je zároveň zobrazován seznam modulů použitých v paměti a seznam dosud neumístěných modulů. Jednotlivé moduly umísťujeme do pamětí pomocí příkazu Insert, čímž příslušný modul přesuneme ze seznamu nepoužitých modulů do seznamu modulů použitých. Příkaz Delete přesune modul naopak a z paměti ho vyřadí. Moduly jsou datové (Data, Stack a Heap) a kódové (ostatní). Datové moduly se musí umístit do paměti RWM a kódové do paměti ROM. Zároveň je zobrazena celková délka použitých a neumístěných modulů a velikost zatím nevyužitého prostoru paměti. Je-li kapacita paměti překročena, je zobrazeno "over".

Příklad umístění aplikace do pamětí:



Pokud jsou všechny jednotky umístěné, můžeme vygenerovat binární obsah pamětí příkazem Generate v menu Place. Program vygeneruje příslušný počet souborů *.BIN, jejichž obsah je binárním obrazem jednotlivých pamětí ROM.

Tyto soubory je možné naprogramovat do paměti EPROM. Je však třeba do paměti také naprogramovat MCP BIOS tak, aby se nacházel na adrese 0FE000h v adresním prostoru pro KitV40.

14. Přílohy

14.1. Program pro generování kódu s BMP

Terminál TERM10 může používat pozadí menu-obrazovky definované černobílou bitovou mapou o rozměrech 240 x 128 pixelů. Bitovou mapu nakreslíme např. v programu PaintBrush, resp. Paint ve Win95, a uložíme jako monochrom BMP. Pomocí programu CRLCDBMP.EXE vytvoříme ze souboru *.BMP textový soubor obsahující zdrojový kód pascalské unit. Tuto unit poté použijeme v naší aplikaci.

V programu lze zadat komentář, který bude uložen do vytvářené pascalské unit. Dále lze nastavit horní mez relativního kompresního poměru, pro kterou je ještě použita komprese BMP, v opačném případě je BMP uložena nekomprimovaná.

Příkazy Menu:

Load	načtění souboru *.BMP
Save	uložení souboru jako unit *.PAS
SaveAs	uložení souboru jako unit Jméno.PAS

14.2. Program pro generování kódu s fontem

Program CRLCDFNT.EXE je určen pro převod souboru fontu ve formátu FNT do zdrojového tvaru pascalské unit s kódem definujícím font pro LCD display. Soubor *.FNT lze editovat ve Windows pomocí programu Borland Resource Workshop.

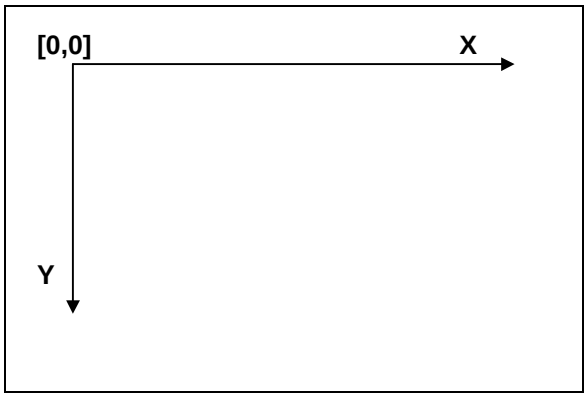
Program CRLCDFNT.EXE dále umí načtený soubor *.FNT zobrazit a poskytnout interaktivní preview pro zvolený LCD display.

Příkazy Menu:

Load	načtění souboru *.FNT
Save	uložení souboru jako unit *.PAS
SaveAs	uložení souboru jako unit Jméno.PAS

14.3. Grafický souřadný systém

Pro potřeby umísťování grafických objektů na display je zaveden souřadný systém s metrikou vyjádřenou v jednotkách pixel (tj. nejmenší zobrazitelný bod na display). Počátek souřadného systému leží v levém horním rohu display, osa X je orientována vodorovně směrem doprava, osa Y je orientována směrem dolů.



obr. V - Grafický souřadný systém

14.4. Syntaxe DisplayStr

Chceme-li používat při výpisu na terminál TERM10 různých fontů, musíme použít objekt tMenuGr. Jedním z parametrů constructoru objektu tMenuGr je ukazatel na proměnnou typu string, pomocí které je v těle uživatelské procedury menu definován tzv. DisplayStr pro danou obrazovku. Tento DisplayStr definuje zobrazovaný text na obrazovce. DisplayStr může obsahovat ESC-sekvence, které definují font a počáteční souřadnice pro výpis bezprostředně následujícího textu za ESC-sekvencí.

Není-li uvedena ani jediná ESC-sekvence, vypíše se text obsažený v DisplayStr v implicitním fontu uspořádán do znakové matice terminálu.

Je-li definována právě jedna ESC-sekvence, je text obsažený v DisplayStr vypsán ve fontu definovaném ESC-sekvencí do znakové matice, jejíž levý horní roh je zobrazen na souřadnici z ESC-sekvence.

V ostatních případech je text zobrazován dle ESC-sekvencí bez respektování znakové matice terminálu. Syntaxe zápisu ESC-sekvence do DisplayStr je následující:

zESC<číslo fontu>,<X>,<Y>{,<Rop>};

kde zESC je ASCII znak ESC a symboly uzavřené v < > označují číselné hodnoty zapsané znakově, tj. pomocí znaků číslic. Souřadnice <X>,<Y> v jednotkách pixel určují počátek výpisu bezprostředně následujícího textu za ESC-sekvencí.

kde <Rop> je operace nad video-pamětí při vykreslování grafického objektu
Rop=0 ... operace OR
Rop=1 ... operace NOT
Rop=2 ... operace XOR
Příklad zápisu DisplayStr:

```
.....
const zESC=$1B;
.....
DispStr:=zESC+'F1,10,10;'+'****** MENU 4 ******'+
                          'zadej string:       '+
                        '''xxxxxxxxxxxxxxxxxx''';
.....
```

Pro editační string je použit styl, který je definován jako poslední.

14.5. Syntaxe GraphicStr

Chceme-li používat grafické možnosti terminálu TERM10, musíme použít objekt tMenuGr. Jedním z parametrů constructoru objektu tMenuGr je ukazatel na proměnnou typu string, pomocí které je v těle uživatelské procedury menu definován descriptor grafických objektů pro danou obrazovku.

Při popisu syntaxe jsou použity tyto konvence:
symboly uzavřené v < > označují číselné hodnoty zapsané znakově
parametry uvedené ve složených závorkách { } jsou parametry volitelnými.

Syntaxe zápisu tohoto descriptoru je následující:

- **objekt bod P(<X>,<Y>{,<Rop>})**
  kde <X> je X-souřadnice bodu na display v jednotkách pixel
  kde <Y> je Y-souřadnice bodu na display v jednotkách pixel
  kde <Rop> je operace nad video-pamětí při vykreslování grafického objektu
  Rop=0 ... operace OR
  Rop=1 ... operace NOT
  Rop=2 ... operace XOR
- **objekt úsečka L(<X1>,<Y1>,<X2>,<Y2>{,<Rop>})**
  kde <X1> <Y1> jsou souřadnice počátečního bodu úsečky a
  kde <X2> <Y2> jsou souřadnice koncového bodu úsečky
- **objekt obdélník R (<X1>,<Y1 >,< X2>,<Y2>{,<Rop>})**
  kde <X1> <Y1> jsou souřadnice levého horního rohu,
  kde <X2> <Y2> jsou souřadnice pravého dolního rohu
- **objekt vyplněný obdélník F (<X1>,<Y1>,<X2>,<Y2>{,<Rop>})**
  kde <X1> <Y1> jsou souřadnice levého horního rohu,
  kde <X2> <Y2> jsou souřadnice pravého dolního rohu
- **objekt kružnice C(<CX>,<CY>,<R>{,<Rop>})**
  kde <CX><CY> jsou souřadnice středu kružnice,
  kde <R> je poloměr kružnice