uMenuChr

JEDNOTKA PRO VYTVÁŘENÍ ZNAKOVÝCH UŽIVATELSKÝCH MENU

Příručka uživatele a programátora

***SofCon*®spol. s r.o.**
Střešovická 49
162 00 Praha 6
tel/fax: +420 220 180 454
E-mail: sofcon@sofcon.cz
www: http://www.sofcon.cz

Verze dokumentu 1.10

Informace v tomto dokumentu byly pečlivě zkontrolovány a SofCon věří, že jsou spolehlivé, přesto SofCon nenese odpovědnost za případné nepřesnosti nebo nesprávnosti zde uvedených informací.
SofCon negarantuje bezchybnost tohoto dokumentu ani programového vybavení, které je v tomto dokumentu popsané. Uživatel přebírá informace z tohoto dokumentu a odpovídající programové vybavení ve stavu, jak byly vytvořeny a sám je povinen provést validaci bezchybnosti produktu, který s použitím zde popsaného programového vybavení vytvořil.
SofCon si vyhrazuje právo změny obsahu tohoto dokumentu bez předchozího oznámení a nenese žádnou odpovědnost za důsledky, které z toho mohou vyplynout pro uživatele.

Datum vydání: 16.05.2003
Datum posledního uložení dokumentu: 16.05.2003
(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	1.XX	Su		První vydání
1.10	2.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000. Zrušená proměnná tMenuChr.Term. Doplněný popis tMenuChr.SetUpKeyChar. Přejmenované funkce InChVerify → InVerifyChar a NotInChVerify → NotInVerifyChar. Opravené hlavičky funkci EditXXXHex a ExitXXXBin. Doplněný funkce PrepareEditStr, SetEditBinCursor, PlacePosXXX, edXXX.

1.2. Účel dokumentu

Tento dokument slouží jako popis knihovny jednotky pro vytváření znakových uživatelských menu uMenuChr.

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro čtení tohoto dokumentu je potřeba seznámit se s manuálem uTermChr, uATerm a uAMenu,.

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

Tato jednotka implementuje rozšíření objektu tAMenu pro práci se znakovými menu. Zděděné procedury zde nejsou popsány, jejich popis lze najít v manuálu k jednotce uAMenu. Rozšíření spočívá v doplnění funkcí pracujících nad textovým bufferem.

Objekt znakového menu **tMenuChr** pracuje pouze se znakovými řetězci popisujícími obsah obrazovky menu a přiřazení fontu. Tyto řetězce předává objektu terminálu a ten je v procesu obnovy obrazovky předává objektu displeje. Až objekt displeje provádí pomocí grafické knihovny interpretaci těchto definičních řetězců pomocí přednastavených fontů do videostránky displeje a tu přenáší na fyzický displej.

Pole fontů musí být uživatelem vytvořeno vně objektu grafického menu i objektu terminálu a ukazatel na ně se předává při inicializaci objektu displeje.

4. Popis konstant a typů

```
cVerNo = např. $0251; { BCD formát }
cVer   = např. '02.51,07.08.2003';
```

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

```
pMMenuChr          =     ^tMMenuChr;
tMMenuChr          =     procedure(P: pMenuChr);
```

tMMenuChr je typ procedura s parametrem ukazatel na objekt **tMenuChr.** Díky tomuto parametru může procedura přímo přistupovat k metodám a položkám objektu.

```
tMenuEditStr       =     String[100];
```

tMenuEditStr je typ pro řetězec používaný při editaci v objektu **tMenuChr**.

5. Popis objektu tMenuChr

```
pMenuChr           =     ^tMenuChr;
tMenuChr           =     object(tAMenu);
```

5.1. Položky

```
MenuEditStr        :     tMenuEditStr;
```

EditString je řetězec používaný pro editaci editačními procedurami, které vyvolává metoda **Run**.

```
WaitTick           :     word;
```

WaitTick je interval čekání při testování klávesnice editačními procedurami, které vyvolává metoda **Run**.

```
KeyDir             :     boolean;
```

Pokud je proměnná **KeyDir** rovna **true**, je možno používat klávesy ‘8’ a ‘2’ jako šipky nahoru a dolů.

```
PlaceString        :     ^string;
```

PlaceString je ukazatel na řetězec, se kterým pracují metody **Place...**

```
PlaceStringSize    :     byte;
```

PlaceStringSize je délka řetězce, se kterým pracují metody **Place... .** Pokud by byl text zapisovaný metodami delší než hodnota této proměnné, požadovaná operace se neprovede.

```
InsertMode        :     boolean;
```

Pokud je **InsertMode** rovno **true**, je text metodami **Place...** vkládán, v opačném případě je původní obsah přepsán.

```
SetupKeyChar      :     Char;
```

V proměnné **SetupKeyChar** je uložen znak pro vstup do SetUpTerminal.

5.2. Metody

5.2.1. Init

```
constructor Init(Menu:pMenu; EndIndex:word;var DisplayString,
                 HelpString:String; Terminal:pTermChr;
                 ParTerm:tParamStr; WaitMenu:Word);
```

Konstruktor **Init** provádí základní inicializace proměnných objektu **tMenuChr**. **Menu** je ukazatel na pole ukazatelů na definiční procedury typu **tMMenuChr**. **EndIndex** je index poslední procedury v poli ukazatelů **Menu^**. **DisplayStr** je řetězec, do kterého se bude ukládat text určený k zobrazení na displej. **HelpStr** je řetězec, do kterého se bude ukládat text určený pro nápovědu k dané menu-obrazovce. **Terminal** je ukazatel na objekt znakového terminálu. **ParTerm** je řetězec, který bude předán pro inicializaci znakového terminálu **Terminal**. **WaitTick** je počet tiků operačního systému reálného času, na které se pozastavuje editace při čekání na stisk klávesy.

5.2.2. Done

```
destructor Done;
```

Destructor **Done** uvolňuje paměť alokovanou pro dynamické položky objektu **tMenuChr** a ruší objekt terminálu.

5.2.3. InitParam

```
procedure InitParam;
```

InitParam vyvolává metodu svého předchůdce (viz dokumentace k jednotce uAMenu), rozšiřuje a modifikuje nastavení proměnných:

UserFirst je nastaveno na metodu objektu tMenuChr.mUserFirst.

DisplayMenu je nastaveno na metodu objektu tMenuChr.mDisplayMenu.

UserSecond je nastaveno na proceduru edMenu.

NextMenu je nastaveno na metodu objektu tMenuChr.mNextMenu.

Do množiny ukončovacích znaků **TerminateChar** jsou vloženy prvky **zUp**, **zDn**, **zCR**, **zEsc**.

Do množiny potvrzovacích znaků **VerifyChar** je vložen prvek **zCR**.

5.2.4. mUserFirst

```
procedure mUserFirst;
```

Metoda **mUserFirst** přiřazuje množině ukončovacích znaků pro editaci objektu terminálu **Term** množinu ukončovacích znaků objektu **tMenuChr.**

5.2.5. mDisplayMenu

```
procedure mDisplayMenu;
```

Metoda **mDisplayMenu** vypisuje data na displej terminálu **Term**. Bližší informace viz popis metody **WriteS** v manuálu k jednotce terminálu.

5.2.6. mNextMenu

```
procedure mNextMenu;
```

Metoda **mNextMenu** se liší od stejnojmenné metody objektu **tAMenu** pouze implementačně. Pro bližší definici viz manuál k jednotce **uAMenu**.

5.2.7. InVerifyChar

```
function InVerifyChar : Boolean;
```

Metoda **InVerifyChar** vrací **true**, je-li přijatý znak prvkem množiny potvrzovacích znaků **VerifyChar**.

5.2.8. NotInVerifyChar

```
function NotInVerifyChar : Boolean;
```

Metoda **NotInVerifyChar** navrací **true**, není-li přijatý znak prvkem množiny potvrzovacích znaků **VerifyChar**.

5.2.9. EnableKeyDir

```
procedure EnableKeyDir;
```

Metoda **EnableKeyDir** nastavuje zpracovávání kláves 2, 8 jako šipky. Toto přepnutí je použito u terminálů bez samostatných kurzorových kláves.

5.2.10. DisableKeyDir

```
procedure DisableKeyDir;
```

Metoda **DisableKeyDir** nastavuje zpracovávání kláves 2, 8 jako čísel.

5.2.11. edStr

```
procedure edStr;
```

Statická metoda **edStr** provádí editaci řetězce **MenuEditStr** v předem nastaveném editačním okně.

5.2.12. DisplayStr

```
procedure DisplayStr;
```

Metoda **DisplayStr** zobrazí řetězec **MenuEditStr** jako při editaci, avšak editaci neprovádí. Testuje pouze příjem znaku **F1**, a v případě jeho přijetí z klávesnice zobrazí nápovědu k dané menu-obrazovce.

5.2.13. ToFirstChar

```
procedure ToFirstChar(var Stri:string);
```

Metoda **ToFirstChar** doplní řetězec **Stri** o sekvenci, která smaže obrazovku, vypíše původní text a posune kursor na první znak, který není mezera.

5.2.14. PrepareEditStr

```
procedure PrepareEditStr(var Stri:String; SkipBlank: Boolean);
```

Metoda **PrepareEditStr** připraví řetězec pro editaci, tj. doplní ho mezerami na šířku okna. Pokud je nastaven příznak **SkipBlank** začíná editace na prvním neprázdném řádku. Pokud není příznak **SkipBlank** nastaven, začíná editace v levém horním rohu okna.

5.2.15. SetEditBinCursor

```
procedure SetEditBinCursor(Position: Byte);
```

Metoda **SetEditBinCursor** nastaví pozici kurzoru pro editaci proměnné v binární podobě na pozici **Position**.

5.2.16. ToEndChar

```
procedure ToEndChar(var Stri:string);
```

Statická metoda **ToEndChar** doplní řetězec **Stri** o sekvenci, která vypíše původní text a posune kursor za poslední znak řetězce.

5.3. Metody pro editaci řetězce

Všechny tyto procedury jsou určeny pro jednodušší práci s textovými řetězci. Řetězec, se kterým budou všechny operace pracovat se nastaví voláním procedury **SetDestString**. Toto nastavení se zruší zavoláním procedury **ClearDestString**. Mód vkládání (resp. přepisování) se nastaví voláním procedury **SetInsertMode** (resp.**SetOverMode**). Metody **PlaceXY...** jsou určeny k vložení (resp. přepsání) nového textu do řetězce původního. První dva parametry jsou vždy X-ová a Y-ová souřadnice místa, kde má být nový text umístěn a další parametry specifikují nový text. Tyto procedury jsou určeny pro práci s řetězci **DisplayStr** a **HelpStr**.

5.3.1. SetDestString

```
procedure SetDestString(var S:String; Len:Byte);
```

Tato procedura určuje řetězec, se kterým budou všechny procedury **PlaceXY...** pracovat. Proměnná **Len** určuje maximální délku řetězce. Pokud je délka při zápisu delší než **Len**, daná operace se neprovede. Procedura si zjistí šířku displeje z objektu terminálu, a proto je možné používat Y-ovou souřadnici.

5.3.2. ClearDestString

```
procedure ClearDestString;
```

Tato procedura anuluje parametry určené procedurou **SetDestString**.

5.3.3. SetInsertMode

```
procedure SetInsertMode;
```

Nastaví mód pro vkládání nového textu do starého.

5.3.4. SetOverMode

```
procedure SetOverMode;
```

Nastaví mód pro přepis starého textu novým.

5.3.5. PlaceXYString

```
procedure PlaceXYString(X,Y:Byte; S:tEditString);
```

Umístí řetězec **S** na pozici X, Y.

5.3.6. PlaceXYByte

```
procedure PlaceXYByte(X,Y:Byte; B:Byte; Len:Byte);
```

Zkonvertuje byte **B** na řetězec o délce **Len** a umístí jej na pozici X, Y.

5.3.7. PlaceXYWord

```
procedure PlaceXYWord(X,Y:Byte; W:Word; Len:Byte);
```

Zkonvertuje word **W** na řetězec o délce **Len** a umístí jej na pozici X, Y.

5.3.8. PlaceXYInteger

```
procedure PlaceXYInteger(X,Y:Byte; I:Integer; Len:Byte);
```

Zkonvertuje integer **I** na řetězec o délce **Len** a umístí jej na pozici X, Y.

5.3.9. PlaceXYLongInt

```
procedure PlaceXYLongInt(X,Y:Byte; L:LongInt; Len:Byte);
```

Zkonvertuje longint **L** na řetězec o délce **Len** a umístí jej na pozici X, Y.

5.3.10. PlaceXYReal

```
procedure PlaceXYReal(X,Y:Byte; R:Real; Len:Byte; Flt:Byte);
```

Zkonvertuje reálné číslo **R** na řetězec o délce **Len** a počtu desetinných míst **Flt** a umístí jej na pozici X, Y. Pokud je **Len** rovno **Flt**, bude výsledek v pohyblivé desetinné čárce, v opačném případě bude desetinná čárka pevná.

5.3.11. PlaceXYRealExp

```
procedure PlaceXYRealExp(X,Y:Byte; R:Real; Len:Byte);
```

Zkonvertuje reálné číslo **R** na řetězec o délce **Len** v exponenciálním tvaru a umístí jej na pozici X, Y.

5.3.12. PlaceXYByteHex

```
procedure PlaceXYByteHex(X,Y:Byte; B:Byte);
```

Zkonvertuje byte **B** na řetězec v hexadecimálním zápisu a umístí jej na pozici X, Y.

5.3.13. PlaceXYWordHex

```
procedure PlaceXYWordHex(X,Y:Byte; W:Word);
```

Zkonvertuje word **W** na řetězec v hexadecimálním zápisu a umístí jej na pozici X,Y.

5.3.14. PlaceXYLongIntHex

```
procedure PlaceXYLongIntHex(X,Y:Byte; L:LongInt);
```

Zkonvertuje longint **L** na řetězec v hexadecimálním zápisu a umístí jej na pozici X,Y.

5.3.15. PlaceXYByteBin

```
procedure PlaceXYByteBin(X,Y:Byte; B:Byte);
```

Zkonvertuje byte **B** na řetězec v binárním zápisu a umístí jej na pozici X,Y.

5.3.16. PlaceXYWordBin

```
procedure PlaceXYWordBin(X,Y:Byte; W:Word);
```

Zkonvertuje word **W** na řetězec v binárním zápisu a umístí jej na pozici X,Y.

5.3.17. PlaceXYLongIntBin

```
procedure PlaceXYLongIntBin(X,Y:Byte; L:LongInt);
```

Zkonvertuje longint **L** na řetězec v binárním zápisu a umístí jej na pozici X,Y.

5.3.18. PlaceXYSetBin

```
procedure PlaceXYSetBin(X,Y:Byte; S:LongInt; Len:Byte);
```

Zkonvertuje množinu **S** na řetězec v binárním zápisu s počtem bitů **Len** a umístí jej na pozici X,Y.

5.3.19. PlacePosString

```
procedure PlacePosString(P: Byte; S:tMenuEditStr);
```

Procedura zajistí vložení řetězce **S** do editačního řetězce **PlaceString** od pozice **P**. Pokud je výsledný řetězec delší než maximální povolená délka, je zkrácen původní řetězec **PlaceString**. Pokud není nastavený příznak InsertMode, řetězec **S** přepíše část řetězce **PlaceString** od pozice **P**.

5.3.20. PlacePosXXX

```
procedure   PlacePosByte      (P: Byte; B: Byte; Len: Byte);
procedure   PlacePosWord      (P: Byte; W: Word; Len: Byte);
procedure   PlacePosInteger   (P: Byte; I: Integer; Len: Byte);
procedure   PlacePosLongint   (P: Byte; L: Longint; Len: Byte);
procedure   PlacePosReal      (P: Byte; R: Real; Len: Byte; Flt:
                              Byte);
procedure   PlacePosRealExp   (P: Byte; R: Real; Len: Byte);
procedure   PlacePosByteHex   (P: Byte; B: Byte);
procedure   PlacePosWordHex   (P: Byte; W: Word);
procedure   PlacePosLongintHex(P: Byte; L: Longint);
procedure   PlacePosByteBin   (P: Byte; B: Byte);
procedure   PlacePosWordBin   (P: Byte; W: Word);
procedure   PlacePosLongIntBin(P: Byte; L: LongInt);
procedure   PlacePosSetBin    (P: Byte; S: Longint; Len: Byte);
```

Tyto procedury používají proceduru **PlacePosString**. Nejdříve se převede číselná hodnota na řetězec typu **tMenuEditStr** (v závislosti na typu funkce) a pak se získaný řetězec vloží do editačního řetězce **PlaceString** od pozice **P**.

5.4. Metody pro editaci

Všechny tyto procedury převedou proměnnou, která je dána jako parametr na řetězec. Tento řetězec dají editovat metodě **edStr** a po úspěšném skončení editace jej zase převedou na původní typ, zkontrolují rozsahy a uloží do proměnné. Pokud došlo k chybě převodu, vezme procedura původní obsah a začne editovat znovu. Metoda **edStr** umí vypsat po stisku klávesy F1 Nápovědu a to poskytuje i ostatním procedurám. Pokud obdrží ukončovací znak, který není znakem potvrzovacím (tzn. není obsažen v množině **VerifyChar)**, tak se proměnná nepřepisuje, ale pouze se ukončí editace.

Pokud je nastaven režim **flUpDn** (režim pro malé terminály s omezenou klávesnicí, viz UtermChr), nastavují tyto procedury zároveň parametr **edType** (viz UtermChr) podle typu předávaného řetězce (binární čísla, hexa čísla, celá čísla, text, písmena). Nastavení proměnné **edType** je využíváno pro výběr sady povolených znaků při tomto režimu editace, kdy pomocí šipek nahoru/dolů měníme znak na pozici kurzoru za jeho následníka/předchůdce v dané sadě znaků. S výjimkou procedury **EditLetters**, která je k dispozici pouze pro režim **FlUpDn**=true, jsou všechny editační procedury použitelné v obou režimech editace.

Chování editačních metod se modifikuje proměnnou **EditEnter**. Je-li **EditEnter** rovna **true**, metody pouze vypíší proměnnou pomocí metody **DisplayStr** do editačního okna v předepsaném formátu. Kursor je potlačen a metoda čeká na vstup znaku z množiny **TerminateChr**. Během čekání na znak se dynamicky obnovuje výpis proměnné na obrazovce. Tzn. pokud se hodnota proměnné změní,

projeví se to na výpisu. Pokud obdrží znak z množiny **VerifyChar**, objeví se kursor, hodnota proměnné se zmrazí a začne vlastní editace. Pokud jde o jiný znak z **TerminateChr**, metoda ukončí činnost; novou hodnotu proměnné nenastavuje a ukončovací znak vrátí v proměnné **ActChar**. Na znak **F1** i v tomto čekání vypisuje nápovědu. Po úspěšné editaci zakončené znakem z množiny **VerifyChar** opět přechází do módu čekání. V tomto módu lze, bez nebezpečí přepsání, prohlížet obsahy proměnných.

Je-li **EditEnter** rovno **false**, editační metody bez čekání na znak z množiny **VerifyChar** přejdou rovnou do módu editace a po ukončení editace opouštějí metodu. Toto nastavení je implicitní.

Při editaci celočíselných a reálných typů lze pomocí proměnné **EditReject0** řídit, zda se budou při hodnotě proměnné rovné 0 předkládat k editaci nuly, nebo prázdný řádek.

Ukončovací znaky platné editace jsou v množině **VerifyChar**. Pozor, tato množina nemusí obsahovat stejné prvky jako množina **TerminateChr**.

Jelikož se editace provádí pomocí editační metody **ReadLnH** použitého objektu pro práci s terminálem, jsou editační schopnosti plně závislé na této metodě. Obecně lze říci, že se používají editační klávesy šipka nahoru, dolů, doprava, doleva, Home a End. Podle nastavení Insert se pracuje buď v přepisovacím, nebo vkládacím režimu. Pokud je řídicí znak znakem ukončovacím, nemůže pochopitelně fungovat pro řízení kursoru, poněvadž ukončí editaci. Klávesa Del maže znak pod kursorem, klávesa BackSpace maže znak vlevo od kursoru. Pro bližší specifikaci viz manuál k použité jednotce terminálu.

Editační procedury nenastavují okno pro editaci proměnné. To musí uživatel v definiční proceduře menu udělat sám. Metodou **SetEditWin** (viz manuál k jednotce **uAMenu**) nastavíme okno a v něm se objeví editovaná proměnná.

Všechny editační procedury nastavují podle výsledku editace proměnnou **flEditOk**. Tím oznamují, že editace byla úspěšně dokončena, nebo byla předčasně ukončena.

Editační procedury používají následující funkce, které byly zachovány ve veřejné části jen z důvodu zpětné kompatibility. Zavoláním editační funkce se zpravidla nastaví parametry pro editaci a pomocí funkce **SetUserSecond()** se nastaví **UserSecond** na funkci **edXXX**.

```
procedure edString       (P: pAMenu);
procedure edMenu         (P: pAMenu);
procedure edByteHex      (P: pAMenu);
procedure edWordHex      (P: pAMenu);
procedure edLongintHex   (P: pAMenu);
procedure edByteBin      (P: pAMenu);
procedure edWordBin      (P: pAMenu);
procedure edSetBin       (P: pAMenu);
procedure edWait         (P: pAMenu);
procedure edWaitPress    (P: pAMenu);
procedure edPassWord     (P: pAMenu);
procedure edReturn       (P: pAMenu);
procedure edCislo        (P: pAMenu);
procedure edByteDec      (P: pAMenu);
procedure edWordDec      (P: pAMenu);
procedure edInteger      (P: pAMenu);
procedure edLongint      (P: pAMenu);
procedure edReal         (P: pAMenu);
```

5.4.1. EditString

```
procedure EditString(var S:String)
```

Metoda **EditString** edituje řetězec **S**. Metoda umožňuje vkládat do řetězce veškeré znaky Asci, které terminál vygeneruje. V režimu **FlUpDn**=true se do stringu při výměně znaků po stlačení Up, Dn postupně dosazují veškeré tištitelné znaky, viz **uTermChr**.

5.4.2. EditLetters

```
Procedure EditLetters(var S:String)
```

Metoda **EditLetters** edituje řetězec **S**. Metoda se dá použít pouze v režimu **FlUpDn**=true. Metoda vkládá do řetězce pouze číslice a písmena, viz **uTermChr**.

5.4.3. EditMenu

```
procedure EditMenu
```

Metoda **EditMenu** pouze testuje klávesnici. Přijme-li znak obsažený v množině ukončovacích znaků, přejde podle něj k další položce menu. Na znak **zF1** vypíše Help. **EditMenu** je implicitní mód činnosti, který nastavuje **InitParam**.

5.4.4. EditByte

```
procedure EditByte(var B:Byte; LoLim,HiLim:Byte; Width:Byte)
```

Metoda **EditByte** edituje byte **B** dekadicky, horní a dolní mez se určuje proměnnými **LoLim** a **HiLim**. Celkový počet znaků proměnné se určuje proměnnou **Width**.

5.4.5. EditWord

```
procedure EditWord(var W:Word; LoLim,HiLim:Word; Width:Byte)
```

Metoda **EditWord** edituje word **W** dekadicky, horní a dolní mez se určuje proměnnými **LoLim** a **HiLim**. Celkový počet znaků proměnné se určuje proměnnou **Width**.

5.4.6. EditInteger

```
procedure EditInteger(var I:Integer; LoLim,HiLim:Integer; Width:Byte)
```

Metoda **EditInteger** edituje integer **I** dekadicky, horní a dolní mez se určuje proměnnými **LoLim** a **HiLim**. Celkový počet znaků proměnné se určuje proměnnou **Width**.

5.4.7. EditLongint

```
procedure EditLongInt(var L:LongInt; LoLim,HiLim:LongInt; Width:Byte)
```

Metoda **EditLongInt** edituje longint **L** dekadicky, horní a dolní mez se určuje proměnnými **LoLim** a **HiLim**. Celkový počet znaků proměnné se určuje proměnnou **Width**.

5.4.8. EditReal

```
procedure EditReal(var R:Real; LoLim,HiLim:Real; Width,Float: Byte)
```

Metoda **EditReal** edituje real **R** dekadicky, horní a dolní mez se určuje proměnnými **LoLim** a **HiLim**. Celkový počet znaků proměnné se určuje proměnnou **Width**. Počet znaků proměnné za desetinnou čárkou se určuje proměnnou **Float**.

5.4.9. EditRealExp

```
procedure EditRealExp(var R:Real; LoLim,HiLim:Real; Width:Byte)
```

Metoda **EditRealExp** edituje Real **R** v exponenciálním tvaru, horní a dolní mez se určuje proměnnými **LoLim** a **HiLim**. Celkový počet znaků proměnné se určuje proměnnou **Width**.

5.4.10. EditByteHex

```
procedure EditByteHex(var B:Byte)
```

Metoda **EditByteHex** edituje byte **B** hexadecimálně. Horní a dolní meze jsou implicitně nastaveny na maximální rozsah byte.

5.4.11. EditWordHex

```
procedure EditWordHex(var W:Word)
```

Metoda **EditWordHex** edituje word **W** hexadecimálně. Horní a dolní meze jsou implicitně nastaveny na maximální rozsah word.

5.4.12. EditByteBit

```
procedure EditByteBit(var B:Byte)
```

Metoda **EditByteBit** edituje byte **B** binárně. Znaky jiné než ‘1’ nebo ‘0’ jsou ze zpětné konverze vynechány. To dává možnost oddělovat výpis po čtyřech znacích mezerou. Pokud se zadá znaků ‘1’ a ‘0’ méně než 8, doplní se číslo zleva znaky ‘0’, pokud se zadá znaků více, použije se ke konversi posledních 8 znaků. Pro editaci se nastavuje přepisovací režim a po ní se vrací předchozí režim. Binárně vypisovaný byte se vejde do 9 znaků. Mezi horní a dolní čtveřicí bitů se do výpisu vkládá mezera. Číslo po editaci může mít opět mezi bity vkládané mezery pro lepší oddělení.

5.4.13. EditWordBit

```
procedure EditWordBit(var W:Word; LoLim,HiLim:Word)
```

Metoda **EditWordBit** edituje word **W** binárně, horní a dolní mez se určuje proměnnými **Lo** a **Hi**. Znaky jiné než ‘1’ nebo ‘0’ jsou ze zpětné konverze vynechány. To dává možnost oddělovat výpis po čtyřech znacích mezerou. Pokud se zadá znaků ‘1’ a ‘0’ méně než 16, doplní se číslo zleva znaky ‘0’, pokud se zadá znaků více, použije se ke konversi posledních 16 znaků. Pro editaci se nastavuje

přepisovací režim a po ní se vrací předchozí režim. Binárně vypisovaný word je dlouhý 18 znaků. V jednotlivých bytech se mezi horní a dolní čtveřici bitů vkládá mezera. Předpokládá se, že okno má šířku 9 znaků a je dvouřádkové.

5.4.14. EditSetBit

```
procedure EditSetBit(var S:LongInt; Len:Byte)
```

Metoda **EditSetBit** edituje množinu **S** binárně. Mohutnost množiny se zadá proměnnou **Len**, která může být v rozsahu 1 až 16 včetně. Pokud je mohutnost množiny menší nebo rovna 12, tak se mezi každé čtyři bity od dolních bitů vkládá jedna mezera. Při zpětné konverzi z řetězce se jiné znaky než '0', '1', nebo ' ' nepřipouštějí. Mezery se však při zpětné konverzi vynechají. Kontroluje se, zda výsledná mohutnost množiny není větší než **Len**.

5.4.15. EditWait

```
procedure EditWait(WaitT:LongInt)
```

Metoda **EditWait** žádné znaky nepřijímá, nic nedělá pouze čeká **WaitT** milisekund. To umožňuje prostý výpis textu na displej a po uplynutí určené doby přechod do dalšího menu. Přechod do dalšího menu je podmíněn tím, že metoda simuluje zmáčknutí šipky dolů (**zDn**). Nastavením adresy pro pokračování po přijetí tohoto znaku se určí adresa dalšího menu.

5.4.16. EditWaitPress

```
procedure EditWaitPress(WaitT:LongInt)
```

Metoda **EditWaitPress** se chová podobně jako **EditWait**, ale při čekání testuje klávesnici. Přijme-li znak, chová se jako metoda **EditMenu**. Přijme-li znak, který není z **TerminateChr**, pouze se obnoví délka čekání.

5.4.17. EditPassword

```
procedure EditPassword(var PW:String)
```

Metoda **EditPassword** umožňuje neviditelné zadávání řetězce do proměnné **PW**. Při editaci nevypisuje znaky, ale pouze posouvá kursor. Řetězec převede na velká písmena. Při editaci nepřijímá řídicí znaky. Je-li nastaveno EditEnter na **true**, vkládání hesla začne až po stisku klávesy Enter. V režimu **flUpDn** lze použít šipky jako heslo.

5.4.18. EditReturn

```
procedure edReturn
```

Metoda **edReturn** provede rovnou, bez testování klávesnice, nebo čekání, návrat do nadřízeného menu. To může nahradit použití **UserThird** pro provedení nějaké akce po volbě. Použití je následující:

V menu přiřadíme libovolnému ukončovacímu znaku akci **iCall1** a do tabulky **TransTab[iCall1]** vložíme číslo menu s procedurou **EditReturn**. V tomto menu

provedeme akci, která odpovídá vybranému znaku a ihned se vracíme do původního menu.

6. InitRunMenu

```
procedure InitRunMenu(Menu:pMenuChr; MName:IdentType; MStack:Word;
                      MSPrio:Integer;  MDPrio:Byte; TName:IdentType;
                      TStack:Word; TSPrio:Integer; TDPrio:Byte;
                      MWaitTick,TWaitTick:Word; PUser1,PUser2:tMMenu;
                      var FlEnd:Boolean)
```

Procedura **InitRunMenu** vytvoří a spustí proces pro probíhání mezi jednotlivými položkami menu. **MName, MStack, MSPrio a MDPrio** je jméno, velikost zásobníku, statická a dynamická priorita procesu. Předtím je spuštěna uživatelská procedura **PUser1**, které je předán ukazatel na pole definičních procedur **Menu**.

Poté spustí inicializační proceduru objektu terminálu **TermTick** a předá ji parametry **TName, TStack, TSPrio, TDPrio a TWaitTick**. (viz manuál k jednotce terminálu).

Hlavní činnost, kterou procedura vykonává, je periodické vyvolávání metody **Run**, která zajišťuje zpracování menu-obrazovek. (viz manuál k jednotce **uAMenu**). V této smyčce je také volána druhá uživatelská procedura **PUser2** a procedura **Wait**, která čeká **MWaitTick** 'tiků' operačního systému reálného času **ReTOS**.

Tato smyčka je vykonávána do doby, než je boolovská proměnná rovna **true**. Poté se zruší proces se jménem **TName** a objekt **Menu**.