uAMenu

JEDNOTKA PRO VYTVÁŘENÍ 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)

Upozornění:
V dokumentu použité názvy výrobků, firem apod. mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Obsah :

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Su		První vydání
1.10	2.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000. Opravený typ EditWaitTime. Doplněné proměnné EditBinCursor a Term. Opravená hlavička fce Init. Doplněný popis fce edNop.

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky pro vytváření uživatelských menu.

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.

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

Knihovna uAMenu obsahuje abstraktní objekt pro tvorbu a obsluhu uživatelských menu. Po vytvoření dědice objektu tAMenu pro konkrétní terminál, se tvorba menu skládá z vytvoření souboru procedur, které mění vlastnosti a chování objektu tAMenu. Jedné položce menu odpovídá jedna procedura, která nastaví objekt tAMenu. (text na displeji, text nápovědy, povolené klávesy pro ukončení menu a jim odpovídající odkazy na další menu, atd.) Tuto činnost procedura provádí voláním metod objektu resp. nastavením některých proměnných objektu, případně připojením dalších výkonných procedur do cyklu metody Run. Programátor místo vyplňování tabulek pro každou položku menu, vlastně procedurou nastavuje chování objektu.

Při inicializaci objektu **tAMenu** se předá pole ukazatelů na jednotlivé definiční procedury menu. Metoda **Run**, která zajišťuje běh menu, pak pracuje nad tímto polem a přechází od položky k položce podle toho, jak jsou nastaveny odkazy uživatelem a podle přijatých znaků od operátora. Při zpracování menu se nejprve provede implicitní nastavení menu a uživatel pouze provádí změny oproti standardu. Při zpracování volá metoda **Run** šest procedurálních proměnných a to v tomto pořadí:

"definiční procedura"	- požadované nastavení
UserFirst	- první "uživatelská" akce
DisplayMenu	- zobrazení menu
UserSecond	- druhá "uživatelská" akce, např. editace
NextMenu	- podle ukončovacího znaku nastaví další menu
UserThird	- třetí "uživatelská" akce

Při nestandardních situacích lze použít tzv. menu přerušení (Menu Interrupts). Při běhu procedury NextMenu se tyto přerušení vyhodnotí a nastaví se menu pro zpracování daného přerušení.

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.

```
cStackSize        =     30;
```

cStackSize určuje hloubku zásobníku pro menu.

```
cPrimEditWidth    =     15;
```

cPrimEditWidth je konstanta, kterou se nastavuje při **Init**u délka výpisu čísla. Pokud není **EditSemiLogReal** rovno **true**, pak se při výpisu typu **Real** v exponenciálním tvaru použije toto číslo jako počet znaků.

```
cEditValWidth     =     10;
```

cEditValWidth je konstanta, kterou se nastavuje při **Init**u proměnná **EditValWidth**, která určuje celkový počet míst při výpisu číselných typů.

```
cEditValDecimals  =     5;
```

cEditValDecimals je konstanta, kterou se nastavuje při **Init**u proměnná **EditValDecimals**, která určuje počet míst za desetinou čárkou při výpisu čísla typu **Real.**

```
tNum                =     (nByte, nWord, nInteger, nLongInt, nReal);
```

tNum je výčtový typ určující typ čísla při editaci pomocí procedury **edCislo**.

```
tIndCtrlTab         = (
      iBeg,i00,iUp,iDn,iCr,iYes,iNo,iLe,iRi,
      iCall1,iCall2,iCall3,iCall4,iCall5,
      iCall6,iCall7,iCall8,iCall9,iCall10,... ,iCall40,
      iRet,
      iJmp1,iJmp2,iJmp3,iJmp4,iJmp5,
      iJmp6,iJmp7,iJmp8,iJmp9,iJmp10,... ,iJmp40,
      iEnd
);
```

tIndCtrlTab je seznam indexů do tabulky **CtrlTab**.

```
tSetIndCtrlTab      =     set of tIndChtrlTab;
```

tSetIndCtrlTab je množina prvků **tIndCtrlTab**.

```
tTransTab           =     array[Char] of tIndCtrlTab;
```

tTransTab je tabulka transformující znak přijatý z klávesnice na jeden z příznaků z tabulky **tIndCtrlTab**.

```
tCtrlTab            =     array[tIndCtrlTab] of word;
```

tCtrlTab je tabulka indexovaná typem **tIndCtrlTab** a obsahuje indexy procedur menu.

```
cTransTab : tTransTab = (...);
```

cTransTab je konstanta pro inicializaci tabulky **TransTab**, sloužící pro transformaci přijatých ukončovacích znaků (z klávesnice) od obsluhy na index do tabulky **CtrlTab**, která určuje index následující definiční procedury menu. Konstanta **cTransTab** inicializuje tabulku **TransTab** takto:

Klávesa	Index	Hodnota
CR	$0D	iCr
ESC	$1B	iRet
‘A’	$41	iYes
’a’	$61	iYes
’Y’	$59	iYes
’y’	$79	iYes
‘N’	$4E	iNo
’n’	$6E	iNo
Up Arrow	$C8	iUp
Left Arrow	$CB	iLe
Right Arrow	$CD	iRi
Down Arrow	$D0	iDn
ostatní	-	i00

```
cCtrlTab    : tCtrlTab  = ( ... );
```

cCtrlTab je konstanta obsahující nulové hodnoty a je použita pro vynulování tabulky **CtrlTab.**

```
pAMenu                  =     ^tAMenu;
pMenu                   =     ^tMenu;
pMMenu                  =     ^tMMenu;
tSetChar                =     set of Char;
pString                 =     ^tString;
```

```
tStackArray             =     array[0..cStackSize-1] of Word;
```

tStackArray je pole sloužící jako zásobník pro menu.

```
tMMenu                  =     procedure(P: pAMenu);
```

tMMenu je procedura s parametrem *ukazatel na objekt **tAMenu.*** Díky tomuto parametru může procedura přímo přistupovat k metodám a proměnným objektu.

```
tMenu             =     array[0..16000] of tMMenu;
```

tMenu je pole s ukazateli na definiční procedury jednotlivých položek menu. Díky parametru ukazatele na objekt **tAMenu**, dokáží tyto procedury přímo přistupovat k metodám a proměnným objektu. Nad tímto polem pracuje metoda **RUN**.

5. Popis objektu tStackMenu

```
pStackMenu        =     ^tStackMenu;
tStackMenu        =     object(tObject)
```

Objekt **tStackMenu** obsahuje nástroje pro ovládání zásobníku pro menu. Do objektu se ukládají indexy definičních procedur menu, tak jak jsou menu do sebe vnořena.

5.1. Proměnné

```
StackArray        :     tStackArray;
```

StackArray je zásobník pro ukládání návratových adres pro pohyb v menu.

```
StackPointer      :     byte;
```

StackPointer je index poslední položky z pole **StackArray** a nabývá hodnot 0..**cStackSize-1**.

5.2. Metody

5.2.1. Init constructor

```
constructor Init;
```

Konstruktor **Init** nastaví **StackPointer** na nulovou hodnotu.

5.2.2. PushMenu

```
procedure PushMenu(A : word); virtual;
```

Procedura uloží na zásobník word **A** a zvýší hodnotu proměnné **StackPointer.**

5.2.3. CondPushMenu

```
procedure CondPushMenu(A : word); virtual;
```

Procedura uloží na zásobník word **A** pouze v případě, že tato hodnota na zásobníku uložena není. Takto je zajištěno, že se zásobník při opakovaném volání nezvětšuje.

5.2.4. PopMenu

```
function PopMenu : word;
      virtual;
```

Funkce vybere ze zásobníku word a vrátí ho jako svoji funkční hodnotu a sníží hodnotu proměnné **StackPointer**.

5.2.5. CondPopMenu

```
function CondPopMenu(A : word) : word; virtual;
```

Procedura vybere word **A** ze zásobníku, pokud je tato hodnota na zásobníku uložena. To umožňuje čištění zásobníku, pokud se na danou úroveň menu dostaneme jinak než pomocí "**Return**".

5.2.6. TopMenu

```
function TopMenu : word; virtual;
```

Funkce vrátí poslední položku jako svoji funkční hodnotu, ale nevybírá ji. Funkce nemění hodnotu proměnné **StackPointer**.

6. Popis objektu tAMenu

```
pAMenu      =     ^tAMenu;
tAMenu      =     object(tObject)
```

Objekt **tAMenu** je abstraktním typem pro práci s uživatelskými menu. Hlavní činnost vykonává procedura **Run**, která pracuje nad polem procedur, které definují chování menu (resp. změny oproti standardnímu chování).

6.1. Proměnné

```
UserFirst         :     tMMenu;     {1. uživatelská procedura}
DisplayMenu       :     tMMenu;     {zobrazení textu na Displej}
UserSecond        :     tMMenu;     {2. uživatelská procedura}
NextMenu          :     tMMenu;     {Detekce přijatého znaku
                                     a nastavení řídicí proměnné}
UserThird         :     tMMenu;     {3. uživatelská procedura}
```

UserFirst, DisplayMenu, UserSecond, NextMenu, UserThird je pět procedurálních proměnných. K těmto proměnným se přistupuje pomocí metod **Set(název_proměnné)** a **Get(název_proměnné)**. Metoda **Run** je v pořadí, v jakém jsou zapsány, vyvolává pro každou položku menu. Definiční procedury menu tyto procedurální proměnné plní pomocí funkcí **Set(název_proměnné)** ukazateli na objekty, nebo procedurami.

```
MenuDispStr       :     pString;
```

MenuDispStr je ukazatel na řetězec pro výpis textu na displej.

```
MenuHelpStr       :     pString;
```

MenuHelpStr je ukazatel na řetězec pro nápovědu.

```
MenuArray         :     pMenu;
```

MenuArray je ukazatel na pole ukazatelů na definiční procedury menu.

```
StackMenu         :     tStackMenu;
```

StackMenu je objekt typu **tStackMenu**, sloužící jako zásobník pro ukládání indexů vnořených menu.

```
ChangeMenu      :   Boolean;
```

ChangeMenu je proměnná indikující změnu menu.

```
ActMenu         :   Word;
```

ActMenu je index aktuálního menu v poli **MenuArray**^.

```
OldMenu         :   Word;
```

Pokud se menu změnilo, tato proměnná obsahuje index předchozí procedury do pole **MenuArray**^.

```
PreMenu         :   Word;
```

PreMenu je index menu v poli **MenuArray**^, které bylo zpracováno v přechozím průběhu metody **Run**.

```
EndMenu         :   Word;
```

EndMenu je index poslední položky v poli **MenuArray**^.

```
TransTab        :   tTransTab;
```

Transformační tabulka **TransTab** transformuje přijaté znaky pro vstup do tabulky **CtrlTab**, která určuje index následující procedury v poli **MenuArray**^.

```
CtrlTab         :   tCtrlTab;
```

V tabulce **CtrlTab** jsou uvedeny indexy následujících procedur menu v poli **MenuArray**^. Standardní interpretace v tabulce pro znak šipka nahoru, šipka dolů, šipka vlevo, šipka vpravo, Escape a Enter se tvoří pro každou položku menu vždy znovu metodou **InitParam**. Další interpretace a přiřazení ke konkrétnímu znaku se definuje voláním přiřazovací metody v definiční proceduře menu. Pokud je hodnota v tabulce **CtrlTab** rovna **i00** tak se index definiční procedury menu nemění a při dalším cyklu **RUN** se opět zobrazí stávající menu. Pokud je hodnota v tabulce rovna **iCall1** až **iCall10**, potom se index stávající procedury uloží do **StackMenu** a obsahem **CtrlTab** se naplní proměnná **ActMenu**. V příštím kole **RUN** se vyvolá definiční procedura s indexem **ActMenu**. Pokud je hodnota v tabulce rovna **iRet** tak se **ActMenu** naplní indexem z vrcholu **StackMenu**. Menu se vrátí k poslední položce odkud jsme použili jeden z indexů **iCall1** až **iCall10**. Pokud je hodnota v tabulce neznámá, tak se provede **SetAktual(CtrlTab[TransTab[ActChar])**.

```
TransUpCase     :   boolean;
```

Pokud je **TransUpCase** rovno **true,** tak se všechny přijaté znaky z proměnné **ActChar** převedou na velká písmena. To se provede ještě před jejich vstupem do konverzní tabulky **TransTab**.

```
ClrScreen       :   boolean;
```

Pokud je **ClrScreen** rovno **true,** tak se v metodě **DisplayMenu** před výpisem řetězce **MenuDisplayStr**^ smaže obrazovka. Pokud je FALSE tak se pouze přesune kurzor do počátečního stavu (levý horní roh), aby se obrazovka přepisovala od začátku.

```
ActChar         :   Char;
```

ActChar je znak přijatý z klávesnice.

```
TerminateChar   :   tSetChar;
```

TerminateChar je množina ukončovacích znaků menu, které se dekódují a na jejichž základě program přejde k jiné položce menu.

```
VerifyChar      :   tSetChar;
```

Ze všech ukončovacích znaků z **TerminateChar** úspěšně ukončují editaci pouze znaky obsažené v množině **VerifyChar**. Ostatní znaky editaci přeruší a nedojde k přepisu výsledku do **EditVal^.** Implicitně se do **VerifyChar** nastavuje Enter.

```
EditWBeginX       :     byte;       {počátek editačního okna X}
EditWBeginY       :     byte;       {počátek editačního okna Y}
EditWWidth        :     byte;       {šířka editačního okna}
EditWHeight       :     byte;       {výška editačního okna}
```

EditWBeginX, **EditWBeginY**, **EditWWidth**, **EditWHeight** určují velikost editačního okna ve znacích pro metodu **Edit**. Souřadnice levého horního rohu displeje jsou [0,0].

```
EditVal           :     pointer;
```

EditVal je ukazatel na proměnnou, která se bude editovat v metodě **Edit**.

```
EditEnter         :     boolean;
```

Pokud se **EditEnter** rovno **true**, tak se při editaci proměnná pouze vypíše na displej v předepsaném formátu a čeká se na znak **VerifyChar**, nebo řídicí znak z **TerminateChar**. Po znacích z množiny **VerifyChar** se objeví kursor a začne editace. Po ukončení editace znaky z množiny **VerifyChar** se opět přejde do zobrazovacího módu, ve kterém se přijímají znaky. Pokud je **EditEnter** rovno **false**, při editaci se proměnná rovnou edituje.

```
EditSemiLogReal   :     boolean;
```

Když je **EditSemilogReal** rovno **true**, tak metoda **Edit** vypisuje reálná čísla s exponentem na **PrimEditWidth** znaků.

```
EditValWidth      :     byte;
```

EditValWidth určuje u numerických typů celkový počet číslic (u typu real včetně desetinné čárky).

```
EditValDecimals   :     byte;
```

EditValDecimals určuje u typu real počet znaků za desetinou čárkou, pokud není výpis v exponenciálním formátu.

```
EditLowLimit      :     real;
EditHiLimit       :     real;
```

EditLowLimit, **EditHiLimit** je horní a dolní mez čísel pro editaci. Při editaci metoda Edit nedovolí dosáhnout větší hodnoty čísel než **EditHiLimit** a menší hodnoty než je **EditLowLimit**.

```
EditMultipl       :     real;
```

EditMultipl je multiplikativní konstanta pro editaci čísla. Před editací se číslo vynásobí **EditMultipl** a po editaci vydělí **EditMultipl**.

```
EditNumElementsSet:     byte;
```

Číslo **EditNumElementsSet** určuje mohutnost množiny editované metodou **Edit**. Toto číslo může nabývat hodnoty <1..16>.

```
EditReject0       :     boolean;
```

Když je **EditReject0** rovno **true,** při editaci čísla s hodnotou 0 metoda **edit** nulu nevypíše a editace se začíná s prázdným editačním řetězcem.

```
EditOk            :     boolean;
```

Provede-li se úspěšná editace, nastaví se **EditOk** na **true**. Tato proměnná je příznakem o úspěšnosti či neúspěšnosti editace po jejím ukončení.

```
EditWaitTime      :     Longint;
```

EditWaitTime je počet čekacích smyček metody **Edit** v modu edWait.

```
EditNum           :     tNum;
```

EditNum je příznak typu čísla při editaci.

```
EditBinCursor     : Byte;
```

V parametru **EditBinCursor** je počáteční poloha kursoru při editacích binárních typu.

```
MenuInt           :     set of 0..7;
```

MenuInt je množina příznaků přerušení normální posloupnosti menu. Je-li nastavena 0 odskakuje program na menu s indexem **EndMenu**. Je-li nastavena 7 odskakuje program na menu s indexem **EndMenu**-7. 0 má nejvyšší prioritu, 7 má prioritu nejnižší.

```
MenuIntEnable     :     Boolean;
```

Při **MenuIntEnable** rovno **false** jsou MenuInterrupty zakázány, neakceptují se. Při obsluze přerušení se **MenuIntEnable** automaticky nastavuje na **false** a při návratu z obsluhy přerušení na **true**.

```
MenuIntCnt        :     Byte;
```

V proměnné **MenuIntCnt** se automaticky počítá počet vnořených volání v obsluze přerušení pro menu. Při návratu se čítač snižuje o 1 a při posledním návratu se povoluje **MenuIntEnable**.

```
MenuIntChar : Char;
```

Do proměnné **MenuIntChar** se při obsluze přerušení ukládá **ActChar**. Po ukončení přerušení se **ActChar** obnovuje na původní hodnotu.

```
PrimEditWBeginX   :     Byte;
PrimEditWBeginY   :     Byte;
PrimEditWWidth    :     Byte;
PrimEditWHeight   :     Byte;
PrimEditEnter     :     Boolean;
PrimEditWidth     :     Byte;
PrimEditClrScreen :     Boolean;
```

Tyto proměnné slouží jako implicitní nastavení pro proceduru **InitParam**, ve které se plní proměnné stejného jména bez předpony Prim.

```
Term              : pTermChr;
```

Parametr **Term** obsahuje ukazatel na objekt terminálu.

6.2. Metody

6.2.1. Init constructor

```
constructor Init(Menu: pMenu; EndIndex: Word;
                 var DisplayString, HelpString: String);
```

Konstruktor nastaví základní proměnné objektu **tAMenu** a vytvoří objekt **StackMenu** pro ukládání vnořených indexů menu. Pak je zavolána metoda **InitParam**. **Menu** je ukazatel na pole ukazatelů na procedury typu **tMMenu**. **EndIndex** je index poslední položky menu v poli **Menu^**. **DisplayString** je ukazatel na řetězec, do kterého se bude ukládat řetězec zobrazovaný metodou **DisplayMenu**. **HelpString** je ukazatel na řetězec, do kterého se bude ukládat řetězec pro nápovědu. Příklad:

```
type
   tSelector : (
                  tMPrvníMenu,
                  tDruhéMenu
               );
```

```
const
   BegMenu = tMPrvníMenu;
   EndMenu = tDruhéMenu;
const
   DispString : string = '';
   HlpString  : string = '';

procedure pMPrvníMenu(P: pAMenu); forward;
procedure pMDruhéMenu(P: pAMenu); forward;

const
   MenuArray : array [tSelector] of tMMenu =
               (
                  pMPrvníMenu,
                  pMDruhéMenu
               );
var
   MyMenu : tAMenu;

...
MyMenu.Init(@MenuArray, ord(EndMenu), DispStr, HlpStr);
...
```

Typ **tSelector** je použit pro indexaci tabulky s definičními procedurami menu. Procedury **pPrvníMenu** a **pDruhéMenu** jsou vlastní definiční procedury menu a musí být deklarovány jak **far** a jejich jediným parametrem je ukazatel na objekt menu, který je zde pro přístup k metodám objektu. V tomto případě bude mít vždy hodnotu ukazatel na **MyMenu**. Konstanty **BegMenu** a **EndMenu** ukazují na první a poslední definiční proceduru menu. Proměnné **DispString** a **HlpString** budou v definičních procedurách menu plněny řetězci, které popisují vzhled obrazovky a nápovědy k ní. Proměnná **MenuArray** je pole definičních procedur menu. **MyMenu** je vlastní objekt menu.

6.2.2. InitParam

```
procedure InitParam; virtual;
```

Tato metoda je volaná metodou **Run** jako první, pokud došlo ke změně aktuálního menu. Nastavuje proměnné ovlivňující chování menu: proměnné **UserFirst**, **DisplayMenu**, **UserSecond** a **UserThird** nastaví tak, aby neprováděly žádnou činost. **NextMenu** je nastaveno na metodu **mNextMenu** (viz níže). Tabulky **tTransTab** a **tCtrlTab** jsou inicializovány jim příslušnými konstantami **cTransTab** a **cCtrlTab**, přičemž do tabulky **tCtrlTab** jsou doplněny odkazy na aktuální menu pro indexy i00, iCr, iLe a iRi. Pokud menu není nastaveno na první položce je nastaven odkaz na předchozí menu pro index iUp. Obdobně pro nastavení následujícího menu. Poté jsou nastaveny některé proměnné ovlivňující chování editace na konstantní hodnoty.

6.2.3. Run

```
procedure Run; virtual;
```

Metoda **Run** zajišťuje vlastní běh menu. Pracuje nad polem **MenuArray**, ve kterém jsou uloženy odkazy na definiční procedury menu a podle definice menu a přijatých znaků od operátora se rozhoduje, jaká činnost bude provedena. (Přechod do dalšího menu, editace čísla apod.) Pokud bylo od minulého průchodu metodou změněno aktuální menu, je zavolána metoda **InitParam**. Poté metoda volá definiční proceduru menu, ve které uživatel nastaví proměnné objektu, definuje textové řetězce

popisující menu, modifikuje standardní chování menu při stisku kláves, může modifikovat procedurální proměnné (**UserFirst**, **UserSecond**, **UserThird**) a případně může volat i další procedury. Dále jsou postupně volány tyto procedury:

UserFirst, první uživatelská procedura, ve které je možno modifikovat menu ještě před zobrazením na displej.

DisplayMenu, v této proceduře se zobrazí menu podle nastavení v **MenuDispStr**.

UserSecond, druhá uživatelská procedura. Obvykle se zde provádí editace.

NextMenu pohyb mezi jednotlivými menu (viz níže).

UserThird je poslední uživatelská procedura, ve které je možno modifikovat parametry před koncem zpracování menu.

6.2.4. mNextMenu

```
procedure mNextMenu;
```

Touto metodou je implicitně nastavena proměnná **NextMenu**, Na začátku se zjistí, zda-li není nastaven příznak přerušení pro menu, tzn. **MenuInt** <> [], a je povolené přerušení pro menu, tzn. **MenuIntEnable** rovno **true**. Pokud ano, uschová se aktuální znak přijatý z klávesnice, zakáže se přerušení pro menu, vyčistí se příznak daného přerušení a jako další menu se nastaví zpracování přerušení, tzn. jako další menu se vyvolá (**EndMenu -** číslo_přerušení). Menu obsluhující přerušení mohou být do sebe vnořena. Tímto je zpracováno přerušení.

Pokud nenastalo žádné přerušení pro menu, nebo přerušení nejsou povolena, tak se nejprve zjistí, jestli je přijatý znak v množině ukončovacích znaků **TerminateChar** a pokud ano, transformuje se pomocí tabulky **TransTab** na prvek z množiny **tIndCtrlTab**.

Pokud je tento prvek z množiny <**iCall0**, **iCall40**> a index menu tohoto prvku v tabulce **CtrlTab** je menší nebo roven poslednímu indexu menu **EndMenu**, vyvolá se toto menu jako vnořené, tzn. po návratu z něj bude menu nastaveno na nynější. Toto je zajištěno pomocí zásobníku pro menu **StackMenu**.

Pokud je tento prvek roven **iRet**, znamená to, že se vracíme z vnořeného volání menu a proto se další menu určí pomocí zásobníku pro menu **StackMenu**.

Pokud je tento prvek roven **iUp**, je zde možnost, že se vracíme z vnořeného menu. Tato možnost je otestována a v případě této varianty se náležitě upraví zásobník pro menu.

Pro všechny ostatní hodnoty se index následujícího menu určí pomocí tabulky **CtrlTab**.

6.2.5. SetUserFirst, SetDisplayMenu, SetUserSecond, ...

```
procedure SetUserFirst(p: Pointer); virtual;
procedure SetDisplayMenu(p: Pointer); virtual;
procedure SetUserSecond(p: Pointer); virtual;
procedure SetNextMenu(p: Pointer);  virtual;
procedure SetUserThird(p: Pointer); virtual;
```

Metoda nastavuje danou proměnnou (**UserFirst, DisplayMenu ....**) na adresu procedury, předané jako parametr ve formátu: adresa nastavované procedury.

```
Př.: SetUserFirst(@edNop);
```

6.2.6. GetUserFirst, GetDisplayMenu, GetUserSecond, ...

```
procedure GetUserFirst: Pointer; virtual;
procedure GetDisplayMenu: Pointer; virtual;
procedure GetUserSecond: Pointer; virtual;
procedure GetNextMenu: Pointer; virtual;
procedure GetUserThird: Pointer; virtual;
```

Metody vrací hodnoty proměnných **UserFirst, DisplayMenu, ...** jako svoji funkční hodnotu.

6.2.7. SetActMenu

```
procedure SetActMenu(A : word); virtual;
```

Metoda nastavuje index aktuálního menu na **A**.

6.2.8. GetActMenu

```
procedure GetActMenu: word; virtual;
```

Metoda vrací jako svoji funkční hodnotu index aktuálního menu.

6.2.9. GetOldMenu

```
procedure GetOldMenu: word; virtual;
```

Metoda vrací jako svoji funkční hodnotu index předchozího menu.

6.2.10. SetTerminateChar

```
procedure SetTerminateChar(const SetChr : tSetChr); virtual;
```

Metoda nastavuje množinu ukončovacích znaků **TerminateChar** na množinu **SetChr**.

6.2.11. AddTerminateChar

```
procedure AddTerminateChar(const AddChr : tSetChr); virtual;
```

Metoda přidá do množiny ukončovacích znaků **TerminateChar** množinu **AddChr**.

6.2.12. SubTerminateChar

```
procedure SubTerminateChar(const SubChr : tSetChr); virtual;
```

Metoda odebere z množiny ukončovacích znaků **TerminateChar** množinu **SubChr.**

6.2.13. AddSubTerminateChar

```
procedure AddSubTerminateChar(const AddChr, SubChr:tSetChr); virtual;
```

Metoda přidá do množiny ukončovacích znaků **TerminateChar** množinu **AddChr** a odebere z ní množinu **SubChr**.

6.2.14. SetTransTab

```
SetTransTab(const SetChr: tSetChar; IndCtrlTab: tIndCtrlTab);virtual;
```

Metoda **SetTransTab** nastavuje tabulku **TransTab** v rozsahu **SetChar** na hodnoty určené v tabulce **IndCtrlTab**.

6.2.15. SetCtrlTab

```
SetCtrlTab(const SetIndCtrlTab: tSetIndCtrlTab; Menu: Word); virtual;
```

Metoda **SetCtrlTab** nastavuje tabulku **CtrlTab** v rozsahu **SetIndCtrlTab** na hodnotu **Menu**.

6.2.16. SetTransCtrlTab

```
SetTransCtrlTab(const SetChr: tSetChar; IndCtrlTab: tIndCtrlTab;
                Menu: Word);  virtual;
```

Metoda **SetTransCtrlTab** nastavuje tabulku **TransTab** v rozsahu **SetChar** na hodnoty určené v tabulce **IndCtrlTab** a tabulku **CtrlTab** na hodnotu **Menu**.

6.2.17. SetEditWin

```
procedure SetEditWin(WBeginX,WBeginY,WWidth,WHigh:byte);virtual;
```

Metoda **SetEditWin** nastavuje velikost editačního okna ve znacích. Levý horní roh má souřadnice [0,0].

7. Procedura edNop

```
procedure edNop(P: pAMenu);
```

Procedura **edNop** neobsahuje žádný kód. Používá se například při inicializaci objektu tAMenu pro funkce UserFirst, UserSecond a UserThird.