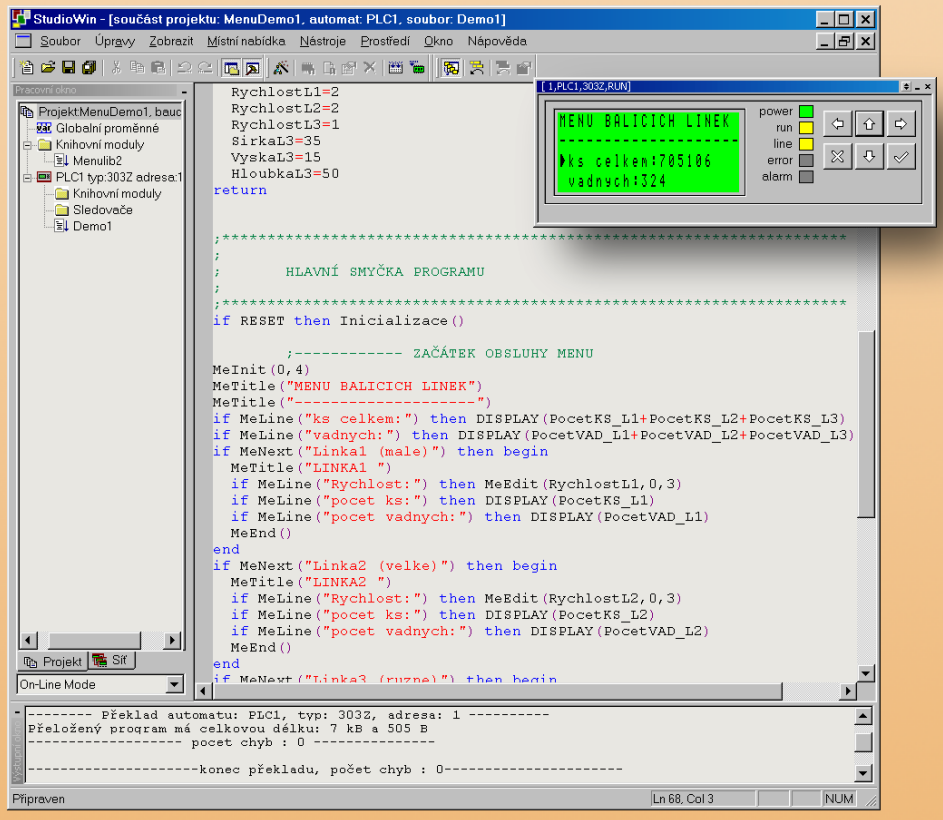
microPEL

MenuLIB

KNIHOVNA SIMPLE4 PRO TVORBU UŽIVATELSKÉHO ROZHRANÍ NA PLC MICROPEL

02.2005



MenuLIB
V2.0

Knihovní funkce v jazyce SIMPLE4 pro snadnou tvorbu uživatelského ovládacího rozhraní ve stylu nabídkového menu na operátorských panelech automatů MICROPEL.

uživatelský manuál - edice 02.2005

1. verze dokumentu

http://www.micropel.cz

OBSAH

1. Možnosti knihovny MenuLIB

Pomocí knihovny je možné na automatech MICROPEL velmi jednoduše a efektivně vytvořit ovládací rozhraní pro uživatele ve stylu nabídkového menu se stromovou strukturou. Tento způsob je velmi rozšířený u všemožných výrobků spotřební elektroniky, je intuitivní a snadno pochopitelný. Knihovna má tyto základní vlastnosti:

- ❑ Libovolný počet položek v menu i počet samostatných vnořených menu
- ❑ Až 10 úrovní zanoření do dalších samostatných menu
- ❑ Vestavěné editory hodnot pro všechny datové typy, vč. typu float
- ❑ Zobrazení i editace přepínačů stavů s využitím jejich textových názvů
- ❑ Podpora pro snadnou tvorbu nápovědy, nebo zobrazení dlouhých bloků textu
- ❑ Jednoduché zobrazování i editace tabulek, seznamů a polí hodnot

Knihovna není žádný speciální nástroj, celé menu se programuje v jazyce SIMPLE4 a je tedy součástí výsledného programu pro automat. Je tedy možno i veškeré vazby a fungování celého systému menu přizpůsobit konkrétním potřebám dané aplikace.

Program s využitím knihovny MenuLIB je velmi krátký a přehledný. Tvar jeho zápisu již sám o sobě velmi přesně odpovídá struktuře výsledného vytvořeného menu.

Velmi zajímavou a důležitou vlastností je **nezávislost na velikosti displeje** (resp. na počtu jeho řádků). Počet řádků se nastavuje jen jedním parametrem a lze jej dynamicky měnit (není-li např. žádoucí, aby menu zabralo celý displej).

Ovládání

Celý systém menu je postaven tak, aby pracoval a poskytoval všechny funkce na displejích s libovolným počtem řádků. Samozřejmě, čím více je řádků a znaků, tím větší je komfort a přehlednost při ovládání takového zařízení obsluhou. Proto pro obsáhlejší a složitější systém menu doporučujeme raději použít automaty typů PES-K, nebo MPC303 se 4-řádkovým displejem.

Pro veškerý pohyb v menu, vybírání položek a editaci hodnot slouží 6 základních kláves (šipka nahoru/dolů/vlevo/vpravo, ESC a ENT), které jsou na všech klávesnicích automatů MICROPEL.

Struktura menu

Ovládací menu je vlastně seznam položek, ve kterém se pohybujeme šipkami nahoru a dolů. Posouváme tím po řádcích ukazovátko a označujeme položky (blikající šipka vlevo před položkou). Co položka, to jeden řádek displeje. Je-li seznam krátký, vejde se celý na obrazovku displeje. Je-li seznam delší, pak se po dosažení horního řádku posouvá celá obrazovka po seznamu nahoru a analogicky po dosažení dolní pozice naopak dolů. Takto listujeme celým seznamem bez ohledu na délku seznamu a počet řádků

displeje (musíme mít k dispozici alespoň jeden řádek). Stisk klávesy **ENT** na označené položce může v závislosti na druhu položky způsobit :

- ❑ přechod do editačního režimu a možnost změny hodnoty parametru
- ❑ vstup do dalšího vnořeného menu
- ❑ vyvolání obecně jakékoliv naprogramované akce, procedury, funkce...
- ❑ nic, pokud se jedná o položku čistě informačního charakteru

Klávesa **ESC** má ve všech situacích význam návratu, resp. stornování činnosti. Klávesou **ESC** opouštíme editaci parametru, vracíme se z vnořeného menu zpět do předcházejícího atd..

Nadpis

V záhlaví menu může být umístěn jeden nebo i více nadpisů. Může informovat obsluhu o tom, v jakém menu nebo nastavení se právě nachází, dávat stručnou nápovědu apod. Nadpis je stále v záhlaví. Při listování v menu rolují položky menu na zbývajících řádcích pod ním.

Položky menu

Položka (1 řádek displeje) je základní prvek menu a může mít několik podob:

a) položka typu ŘÁDEK

Výpis textu a za ním např. zobrazení hodnoty nějaké proměnné. Vhodný pro přehledné a okomentované zobrazení řady hodnot.

b) položka typu ŘÁDEK s připojeným editorem

Poskytuje navíc možnost editace hodnoty po stisku klávesy ***ENT***. Potvrzení a zápis hodnoty se provede stiskem ***ENT***, stornování editace a zachování staré hodnoty stiskem ***ESC***. Editorů je několik druhů - editory čísel, nebo editory bitů a stavů, pracující s textovým vyjádřením hodnoty stavu. Podrobný popis ovládání a možností editorů viz dále ve stati ***Editory hodnot***.

c) položka typu DALŠÍ MENU

Položka po stisku ***ENT*** způsobí přechod do dalšího, vnořeného menu. Je to samostatný seznam položek, který funguje jako další samostatné menu (může mít svůj vlastní nadpis, může obsahovat všechny typy položek - tedy i vstupy do dalších vnořených menu). Stisk klávesy **ESC** způsobí návrat zpět do předchozí úrovně a na stejnou pozici. Systém menu dovoluje realizovat až 10 úrovní menu, vnořených do sebe.

2. Používání knihovny

Pro dokonalé využití knihovních funkcí je nutná znalost programování v jazyce SIMPLE4 a alespoň základní znalost používání vývojového prostředí StudioWin. Proto, pokud teprve chcete začít s programováním automatů MICROPEL, doporučujeme před studiem knihoven MenuLIB nejprve prostudovat příručku k jazyku SIMPLE4 a zkusit si napsat ve vývojovém prostředí StudioWin několik jednoduchých příkladů.

Pro snazší pochopení všech funkcí knihovny je zároveň s knihovnou dodáváno několik ukázkových příkladů ve formě samostatných projektů, spustitelných jak na reálném automatu, tak na simulátoru v prostředí StudioWin.

2.1. Obsah distribuce, instalace

V balíčku knihovny je kromě vlastního souboru **menu2.lib** ještě několik kompletních ukázkových projektů s touto knihovnou. Knihovna MenuLIB je od StudioWin verze 6.902 již distribuována jako součást instalace vývojového prostředí StudioWin.

Samostatný balíček s knihovnou obsahuje instalátor, který implicitně nabízí kopírování knihovny do přednastavené složky knihoven a demonstračních projektů do přednastavené složky projektů (vytvoří se při instalaci prostředí StudioWin). Samozřejmě lze knihovnu i ukázkové příklady kopírovat kamkoli jinam.

Pro instalaci a snadné použití knihovny je třeba vývojové prostředí StudioWin verze min. 6.902 (již obsahující aktuální verzi knihovny). Instalaci ze samostatného balíčku lze tedy doporučit jen tehdy, není-li žádoucí nebo potřebné instalovat zároveň aktuální verzi vývojového prostředí.

2.2. Začlenění knihovny do projektu

Před použitím funkcí knihovny je třeba ji zapojit do projektu. V pracovním okně projektu (na levé straně v prostředí StudioWin) ji lze začlenit buď do složky knihovních modulů celého projektu (nahoře), nebo konkrétního automatu. Knihovnu přidáme do složky kliknutím pravého tlačítka myši nad příslušnou složkou a výběrem volby "Vložit soubor" (nebo klávesou "Insert" na této složce). Na vložení souboru se otevře dialog s navigačními tlačítky a výběrem souboru.

Vložení se provede buď tlačítkem "Otevřít" - vloží se odkaz na soubor, nebo tlačítkem "Kopírovat do projektu" - vloží se kopie souboru.

a) Vložení odkazu na soubor

Do projektu se uloží jen odkaz na soubor. Vlastní knihovní soubor existuje jen na cílovém místě (typicky např. ve sdílené centrální složce knihoven). Pokud je použit ve více projektech, pak změna tohoto souboru, resp. jeho nová verze (např. nové verze jednotlivých funkcí, opravy chyb apod.) se projeví ve všech projektech, které mají tuto knihovnu vloženou odkazem (tedy samozřejmě až po překladu). Výhodou je neustálá

aktuálnost použité knihovny ve všech projektech po každé instalaci nové verze knihovny do sdílené složky.

b) Vložení kopie souboru

Do projektu se fyzicky zkopíruje soubor a ten se také bude používat při překladu projektu. Instalace nové verze knihovny do centrální složky knihoven nebo do jiných projektů pak nemá vliv na tento projekt. Nevýhodou je nemožnost automatického přebírání nových verzí knihoven do projektů (musely by se pokaždé ručně zkopírovat do složky projektu), výhodou je zase větší jistota, že případná zavlečená chyba nebo nekompatibilita nové verze neohrozí stávající projekt. ***Pozn.:*** pro bezpečné uložení a zaarchivování nebo přenos projektů je však určena funkce "Export projektu", která na zadané místo uloží projekt i se všemi knihovnami, které jsou k překladu třeba.

2.3. Prostředky a zdroje, použité funkcemi MenuLIB

Protože menu sestavené z funkcí MenuLIB je součástí uživatelského programu pro automat v jazyce SIMPLE4 (a knihovna je rovněž kompletně napsána v SIMPLE4), využívá stejné prostředky automatu a jazyka jako uživatelský program. Pro korektní funkci menu je navíc nutné, aby pro něj některé prostředky zůstaly ***úplně vyhrazeny !***

Časovač T7

Nejdůležitějším zdrojem, který nesmí být v uživatelském programu nikde použit, je časovač T7 (nikde v uživatelském programu se nesmí použít ani časovač, ani žádný z jeho řídících bitů: TEN7, TOE7, TPA7, TDM7). Je to na druhou stranu ale jediný časovač, který systém knihoven využívá (jeho obsluha v knihovně je naprogramována tak, aby další chystané knihovny, které budou pro jazyk SIMPLE4 k dispozici, jej mohly rovněž využít a nezabíraly tak další volné zdroje).

V programu pro automat lze tedy použít pouze časovače T0 až T6.

Uživatelský znak č.7

Znakové displeje automatů MPC300 a PES-K umožňují vytvoření osmi uživatelských znaků (viz popis obsluhy displeje/klávesnice v manuálu SIMPLE4). Menu využívá poslední pozici této sady (znak s kódem 7) pro zadefinování plné ukazovací šipky.

V programu pro automat lze tedy použít pouze uživatelské znaky 0 až 6.

Názvy proměnných a funkcí

V systému knihoven je krom "veřejných" použito ještě mnoho interních proměnných a funkcí. Jejich názvy už není možno použít pro proměnné nebo funkce, definované v uživatelském programu. Je tedy třeba počítat s tím, že při zadefinování nových proměnných nebo procedur s názvem, který je již použit uvnitř knihovny MenuLib, zahlásí překladač duplicitní definici. V takovém případě je třeba zvolit nějaký jiný název.

Pro přehlednost začínají veškeré názvy proměnných i funkcí knihovny MenuLIB písmeny ***Me***. Je tedy nejjednodušší volit názvy symbolů tak, aby nezačínaly písmeny ***Me***, ***me*** nebo ***ME*** (překladač nerozlišuje velikost písmen).

Klávesnice

Kromě názvů začínajících ***Me*** jsou v knihovně ještě zadefinovány symbolické názvy pro hodnoty kódů stisknutých kláves v proměnné KBCODE. Jsou to :

KB_LEFT, KB_RIGHT, KB_ESC, KB_ENT, KB_UP, KB_DOWN

V aplikačním programu nedoporučujeme využívat klávesnici (tedy proměnnou KBCODE) v jiných částech programu mimo menu. Musí-li to být, tak jen s maximální obezřetností. Mohlo by docházet k rozporům v logice ovládání takového zařízení. Výjimkou jsou jen klávesy F1..F3 na automatech PES-K, které knihovna MenuLIB vůbec nepoužívá.

Bit RESET

Pro počáteční inicializaci knihoven MenuLIB po zapnutí automatu je využíván systémový bit RESET, který se po zapnutí nastaví do 1 a nulovat jej musí uživatelský program. Aby systém knihoven mohl tento bit správně využít, musí být RESET vynulován až na konci programu, nejlépe až před závěrečným příkazem **end** na konci programové smyčky. Tak bude zajištěno, že při prvním průchodu programovou smyčkou bude bit RESET=1 (a to po celou dobu průchodu) a při dalším běhu již bude stále RESET=0.

UPOZORNĚNÍ: Bude-li bit RESET nulován už na začátku programu, nebo naopak nebude-li v programu vynulován vůbec, bude to mít za následek naprostou nefunkčnost systému menu (a pravděpodobně i celého programu).

2.4. Další omezení

V uživatelském programu lze použít vždy jen jeden systém menu.

To je dáno vyhrazením jedné pevné sady pracovních proměnných pro běh celého systému knihoven. Nelze tedy vytvořit např. dvě úplně samostatné stromové struktury menu a volit mezi nimi nějakou logickou podmínkou.

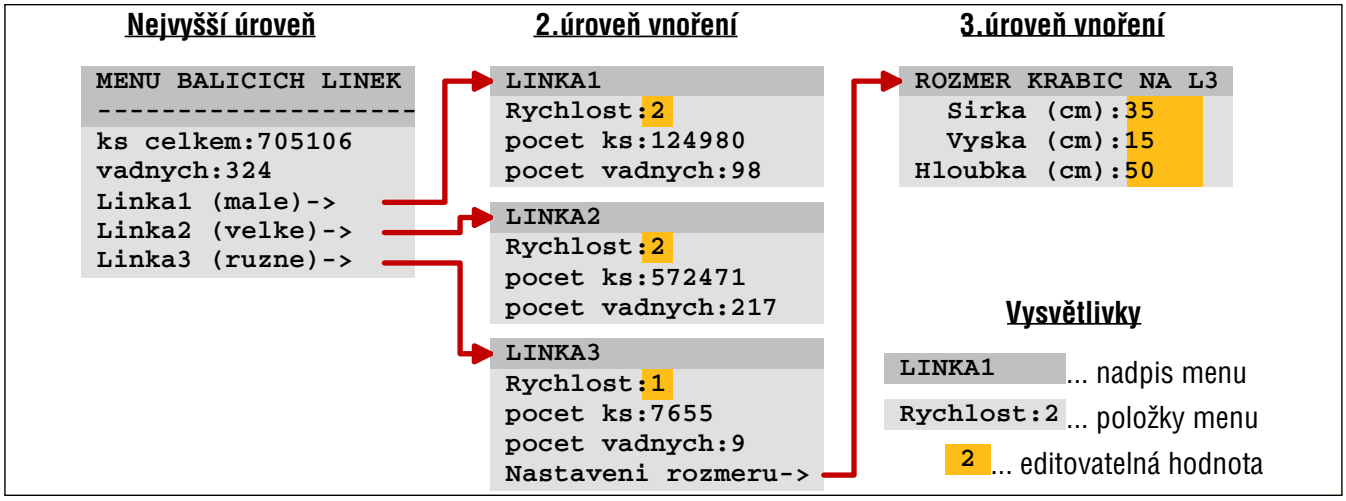
Má-li program fungovat tak, aby např. za určitých situací bylo celé menu "vyměněno" za jiné (anebo jen jeho části), lze to prakticky vždy řešit v rámci jedné struktury menu buď knihovními funkcemi pro větvení a vnořování menu, nebo použitím konstrukcí jazyka SIMPLE4 pro větvení podle stavu logických podmínek (konstrukce if-then-else, nebo switch-case).

3. Programování systému menu

Nejprve krátký příklad v jazyce SIMPLE4 (knihovní funkce MenuLIB vyznačeny tučně):

```
MeInit(0,4)              ;  (ZAČÁTEK OBSLUHY MENU)
MeTitle("MENU BALICICH LINEK")
MeTitle("--------------------")
if MeLine("ks celkem:") then DISPLAY(PocetKS_L1+PocetKS_L2+PocetKS_L3)
if MeLine("vadnych:") then DISPLAY(PocetVAD_L1+PocetVAD_L2+PocetVAD_L3)
if MeNext("Linka1 (male)") then begin
     MeTitle("LINKA1 ")
     if MeLine("Rychlost:") then MeEdit(RychlostL1,0,3)
     if MeLine("pocet ks:") then DISPLAY(PocetKS_L1)
     if MeLine("pocet vadnych:") then DISPLAY(PocetVAD_L1)
     MeEnd()
end
if MeNext("Linka2 (velke)") then begin
     MeTitle("LINKA2 ")
     if MeLine("Rychlost:") then MeEdit(RychlostL2,0,3)
     if MeLine("pocet ks:") then DISPLAY(PocetKS_L2)
     if MeLine("pocet vadnych:") then DISPLAY(PocetVAD_L2)
     MeEnd()
end
if MeNext("Linka3 (ruzne)") then begin
     MeTitle("LINKA3 ")
     if MeLine("Rychlost:") then MeEdit(RychlostL3,0,3)
     if MeLine("pocet ks:") then DISPLAY(PocetKS_L3)
     if MeLine("pocet vadnych:") then DISPLAY(PocetVAD_L3)
     if MeNext("Nastaveni rozmeru") then begin
          MeTitle("ROZMER KRABIC NA L3")
          if MeLine("  Sirka (cm):") then MeEdit(SirkaL3,30,50)
          if MeLine("  Vyska (cm):") then MeEdit(VyskaL3,10,45)
          if MeLine("Hloubka (cm):") then MeEdit(HloubkaL3,40,70)
          MeEnd()
     end
     MeEnd()
end
MeEnd()                  ; (KONEC OBSLUHY MENU)
```

Tento program realizuje ovládací menu s touto strukturou:



Pozn.:

Kompletní a funkční zdrojový text příkladu je v souboru DEMO1.STP.

Poznámka k demonstračním ukázkám zdrojových textů:

Volání procedur nebo funkcí bez parametrů je uváděno vždy s prázdnými závorkami za názvem, i když to z hlediska jazyka není nutné. Zvyšuje to přehled v názvech (je na první pohled vidět, že se jedná o proceduru a ne např. nastavení bitu apod.).

3.1. Vytváření programu menu

Celý systém menu musí vždy začínat funkcí **MeInit** a končit funkcí **MeEnd**. Funkci **MeInit** je nutné použít na začátku celého menu a nesmí být použita už nikde jinde. Naopak funkce **MeEnd** se používá častěji, na konci každého samostatného bloku menu.

Nejjednodušší menu může být jednoúrovňové, ve formě jednoho souvislého bloku položek s jedním začátkem (**MeInit**) a jedním ukončením (**MeEnd**).

Složitější menu (jako např. v naší ukázce) může být rozvětveno pomocí odkazů do samostatných bloků menu v dalších vnořených úrovních. Větvení zajišťují funkce **MeNext**. Počet položek a bloků menu není omezen, limitován je jen počet úrovní menu na max. 10 vnoření. Každý takový blok může mít nadpis (vytvořený funkcí **MeTitle**) a musí být zakončen funkcí **MeEnd**.

Jak je zřejmé ze zápisu programu, není nutno předem znát délku menu nebo nastavovat někde počet položek. Vše počítají knihovní funkce zcela automaticky a průběžně za chodu programu. Stačí jen seřadit za sebe požadované položky menu a vymezit konec každého jejich bloku zavoláním funkce **MeEnd**.

Volání všech funkcí menu nemusí být v kompaktním souvislém bloku jako je tomu v naší ukázce (i když je to asi nejrozumnější). Části systému menu mohou být rozptýleny mezi ostatními částmi uživatelského programu. Nicméně logický sled volání jednotlivých funkcí menu v průběhu vykonávání programové smyčky musí odpovídat výše uvedeným konvencím.

Při tvorbě složitějších menu můžeme pro zpřehlednění programu využít volání procedur a funkcí. V našem příkladu např. část týkající se posledního menu "NASTAVENI ROZMERU" může být přesunuta do procedury. Zápis by pak mohl vypadat třeba takto:

```
     .....
     if MeLine("pocet vadnych:") then DISPLAY(PocetVAD_L3)
     if MeNext("Nastaveni rozmeru") then RozmerKrabic()
     MeEnd()
end
MeEnd()                ; (KONEC OBSLUHY MENU)
```

Dále můžeme pro optimalizaci programu z našeho příkladu přesunout položky "Rychlost", "pocet ks" a "pocet vadnych", které se opakují na začátku několika bloků, do procedury a volat ji s příslušnými parametry.

Zde je ukázka upravené části programu:

```
...
if MeNext("Linka1 (male)") then begin
     UvodniBlok("LINKA1 ",RychlostL1,PocetKS_L1,PocetVAD_L1)
     MeEnd()
end
if MeNext("Linka2 (velke)") then begin
     UvodniBlok("LINKA2 ",RychlostL2,PocetKS_L2,PocetVAD_L2)
     MeEnd()
end
if MeNext("Linka3 (ruzne)") then begin
     UvodniBlok("LINKA3 ",RychlostL3,PocetKS_L3,PocetVAD_L3)
     if MeNext("Nastaveni rozmeru") then RozmerKrabic()
     MeEnd()
end
MeEnd()               ; (KONEC OBSLUHY MENU)
```

3.2. Jak knihovní funkce pracují

Protože blok položek může být delší než je počet řádků na displeji, jsou vždy podle momentální situace některé položky zobrazeny a některé ne. Pozice položek na displeji se mění v závislosti na rolování seznamu po displeji pomocí šipek nahoru a dolů. Navíc, pokud je aktivní nějaký vnořený blok položek, musí být veškeré zobrazování na displej vypnuto až k tomuto momentálně aktivnímu bloku.

Knihovní funkce pro zobrazení nadpisů a položek menu samy řídí zobrazování a pozici na displeji tak, jak je třeba podle logiky obsluhy menu. Než totiž program při svém běhu projde všemi funkcemi od začátku menu až k aktivnímu bloku položek, ve kterém se právě pohybuje obsluha, je zobrazení kompletně "vypnuto" a program funkcemi "propadává" co nejrychleji a bez jakékoliv akce. Po obsloužení aktivních položek se po dosažení nejbližší funkce **MeEnd** opět zobrazování vypne a všechny zbývající knihovní funkce menu až do konce programové smyčky proběhnou naprázdno a rychle.

Podobně pracuje i větvení struktury menu pomocí funkcí **MeNext**. Každá funkce **MeNext** funguje jako výhybka, která při vnoření do nižší úrovně neustále směruje běh programu do postranní větve. Zobrazování je však vypnuté a funkce propadávají naprázdno až do dosažení právě aktivních položek v aktuální úrovni vnoření. Pak se obslouží položky, které jsou právě na displeji a po dosažení nejbližšího **MeEnd** opět vše končí a funkce menu probíhají naprázdno až do konce programové smyčky.

Aby bylo možné do položek i nadpisů kromě základního textu snadno začlenit další prvky, poskytují funkce informaci o tom, zda je možno tisknout na displej. Pokud ano, vytisknou předaný text a nastaví systémovou proměnnou **POSITION** (udávající pozici na displeji, kam se bude tisknout) přesně za konec tohoto textu. Stačí potom jen otestovat povolení k tisku (stavový bit nebo výstupní hodnotu, podle konkrétního typu funkce) a dotisknout za položku cokoliv pomocí standardní systémové funkce **DISPLAY**. Další možností je zařazení některé knihovní funkce pro editaci hodnot. Ty jsou již přímo uzpůsobeny k tomu, aby se daly napojit na knihovní funkce pro vytváření položek.

3.3. Stavové proměnné

Informace o povolení k tisku, stavu menu, vnoření apod. předávají funkce buď svojí výstupní hodnotou (lze tedy rovnou volání funkce použít ve vyhodnocovací části konstrukce if-then) nebo do stavových proměnných definovaných v knihovně. Tyto proměnné odrážejí stav knihovní funkce, jež právě proběhla, volání dalších funkcí knihovny MenuLIB je může opět změnit nebo anulovat.

UPOZORNĚNÍ

Stavové proměnné jsou funkcemi menu nastavovány, ale i používány pro řízení chodu. Lze je číst, vyhodnocovat, používat k větvení programu, ale zápis do nich může vyvolat vážné poruchy funkce menu !

Základní stavové proměnné jsou tyto a týkají se stavu položky menu:

bit	**Me_DISP**	=1: povolení k tisku (položka se zobrazuje, lze za ni tisknout)
bit	**Me_SELECT**	=1: položka zobrazena a označena ukazovátkem (blikající šipka)
bit	**Me_PRESS**	=1: na označené položce byla navíc stisknuta klávesa ENT

Stavová proměnná, kterou naopak může programátor nastavovat, je tato:

bit	**Me_SYMBOLOFF**	=1: vypnutí tisku šipky ve funkcích MeNext (po RESETu je nulový, tedy šipka se tiskne)

3.4. Systémové proměnné

Funkce knihovny MenuLIB ovlivňují i některé systémové proměnné.

POSITION

Při běhu a na výstupu funkcí zobrazujících na displej je nastavována proměnná **POSITION** tak, aby spolupráce se systémem menu byla co nejjednodušší.

KBREPEN, KBREPEAT, KBDELAY

Po startu automatu (když RESET=1) se nastaví režim automatického opakování kláves při delším podržení klávesy (tzv. "auto-repeat") a proměnné s ním související:

KBREPEN=1, **KBREPEAT**=10, **KBDELAY**=50

(aut.opakování povoleno, opakování po 100ms, počáteční prodleva 500ms)

Nastavení provádí funkce **MeInit** při resetu. Toto nastavení je ve většině případů užitečné. Pro chod menu však není podmínkou, je tedy možné po prvním proběhnutí **MeInit** tyto proměnné jakkoliv přestavit, nebo auto-repeat úplně vyřadit.

KBCODE

Proměnná poskytující kód stisknuté klávesy. Je stěžejní pro chod menu, v některých případech ji funkce menu i nulují nebo nastavují.

4. Seznam knihovních funkcí

Podrobný popis všech funkcí. Ke každé funkci je uveden řádek s její přesnou definicí v jazyce SIMPLE4, dále případné ukázky použití ve formě zdrojových textů SIMPLE4. S knihovnami MenuLIB se distribuují i ukázkové soubory a funkční příklady, které lze přeložit a spustit na reálném automatu nebo simulátoru v prostředí StudioWin.

4.1. Základní funkce

Základní stavební kameny každého menu. Slouží k vytváření položek, větvení a zajištění chodu systému menu. Způsob jejich použití je patrný z úvodního příkladu.

MeInit

Inicializace a začátek menu, nastavení pozice a délky

```
subroutine MeInit(byte StartLine : byte MenuLen)
```

Předávané parametry

1.parametr (byte) udává řádek displeje, kde menu začíná (0..3)

2.parametr (byte) udává počet řádků displeje, na kterých se menu rozvíjí (1..4)

Použití

Procedura se stará o chod menu, v programu se použije ***právě jednou***, na začátku celého menu. V každém průchodu programové smyčky se musí nejprve volat procedura **MeInit** a pak teprve vše, co souvisí se systémem menu.

Vše se bude zobrazovat jen od zadaného řádku a na zadaný počet řádků na displeji. Ostatní řádky zůstávají netknuty. Např. **MeInit(1,2)** na automatu MPC303 vytvoří menu na prostředních 2 řádcích displeje, horní a dolní řádek zůstane volný.

Po zapnutí automatu provádí **MeInit** úvodní inicializaci při nastaveném bitu RESET. Je tedy nutné, aby uživatelský program nuloval bit RESET až na konci programu a aby procedura **MeInit** nebyla nějakou podmínkou odstavena, když je RESET=1.

MeEnd

Zakončení bloku menu

```
subroutine MeEnd()
```

Použití

Procedura je bez parametrů. Musí být použita k zakončení každého celistvého bloku menu započatého funkcí **MeNext** a též na konci celého menu. Celá struktura menu je tedy uzavřena párem procedur **MeInit - MeEnd**.

Platí zásada, že ke každému volání **MeInit** a **MeNext** musí na konci příslušného bloku existovat zakončení **MeEnd**. Používání procedury je patrné z úvodního příkladu.

MeTitle

Vytvoření nadpisu bloku menu

```
function bit MeTitle(const string txt)
```

Předávané parametry

1.parametr (const string): text nadpisu

Výstupní hodnota

výstup (bit): =0 ... zákaz zobrazení, =1 ... zobrazení povoleno

Použití

Rezervuje řádek na displeji (odshora) pro nadpis, předaný parametrem jako textový řetězec. Řetězce lze vytvořit i do předdefinované tabulky konstrukcí "table string ..." a do funkce jako parametr předat odkaz na tabulku s indexem (i s proměnným indexem).

Funkce je vždy na začátku bloku menu, teprve za ni se pak řadí jednotlivé položky (viz úvodní příklad). Použití funkce je nepovinné, blok položek menu může být i bez nadpisu. Je možné použít i více funkcí **MeTitle** za sebou, každá vytvoří text na jeden řádek - vznikne tak víceřádkový nadpis.

Nadpis se neposouvá po displeji tak, jako položky. Je stále na stejném místě. Čím více řádků se použije pro nadpis, tím méně jich zbyde na zobrazení seznamu položek. Aby byla zajištěna funkčnost seznamu položek, musí zbýt alespoň jeden řádek pro zobrazení seznamu položek. Funkce **MeTitle** má zabudovanou ochranu a pokud by po ní už neměl zůstat volný řádek, nadpis se nevytvoří.

Výstupní hodnotu není nutno využít. Není-li třeba další dotisk za text nadpisu, volá se funkce jako procedura - viz příklad.

Příklady použití

```
     ; A) jednoduchý nadpis
MeTitle("STAV LINKY:")
if MeLine("pocet kusu:") then DISPLAY(PocetKS)
...
     ; B) dvouřádkový nadpis
MeTitle("STAV LINKY:")
MeTitle("-----------")
if MeLine("pocet kusu:") then DISPLAY(PocetKS)
...
     ; C) nadpis s využitím dotisku další informace
if MeTitle("STAV LINKY C.") then begin
  DISPLAY(CisloLinky)
  DISPLAY(":")    ; a ještě dvojtečku, aby to lépe vypadalo
end
if MeLine("pocet kusu:") then DISPLAY(PocetKS)
```

MeLine

Vytvoření položky menu typu ŘÁDEK

```
function byte MeLine(const string txt)
```

Předávané parametry

1.parametr (const string): text položky

Výstupní hodnota

výstup (byte): =0 ... zákaz zobrazení, >0 ... zobrazení povoleno

Použití

MeLine vytvoří v menu položku s textem, který je předán parametrem. Před vytištěným textem je vlevo na displeji rezervován 1 znak pro případnou blikající šipku, označující aktuální pozici "ukazovátka" do seznamu položek.

Bezprostředně za vytištěný text lze buď standardními prostředky jazyka SIMPLE4 dotisknout další informace nebo zařadit některý z editorů proměnných pro snadnou změnu hodnot proměnných přímo z menu (knihovny MenuLib jich nabízejí několik).

Příklady použití

```
     ; A) využití MeLine k pouhému generování textového řádku
     ; (tvorba nápovědy apod.)
MeLine("----- NAPOVEDA -----")
MeLine("spusteni dopravniku")
MeLine("provedeme vyberem")
MeLine("polozky SPUSTIT")
MeLine("v menu DOPRAVNIKY !")
MeEnd()

     ; B) položka s proměnným textem podle předdefinované tabulky "Tabl"
     ; (výběr textu je dán proměnnou "Rezim") :
     ; zadefinovani tabulky textu :
table string[4] Tabl = ("Linka stoji","Linka jede","Porucha","Odstavka")
     ; volani s vyberem textu z "Tabl" podle hodnoty indexu "Rezim" :
MeLine(Tabl[Rezim])

     ; C) dotisk čísla za text položky :
if MeLine("pocet ks:") then DISPLAY(PocetKS_L1)

     ; D) dotisk různých textů z tabulky za základní text MeLine
     ; podle stavu proměnné "Rezim" :
table string[4] TabStavu = ("Stoji","Jede","Porucha","Odstavka")
if MeLine("cinnost: ") then DISPLAY(TabStavu[Rezim])

     ; E) dotisk pomocí další funkce :
if MeLine("dnes je: ") then TiskniDatum()

     ; F) začlenění editoru hodnoty (popisy editorů budou dále)
if MeLine("zadana teplota:") then MeEdit(ZadTepl,-10,30)
```

Stavové proměnné

Funkce **MeLine** při svém běhu nastavuje tyto stavové proměnné:

bit **Me_DISP**, bit **Me_SELECT** a bit **Me_PRESS**

(popis stavových proměnných byl uveden v předchozí kapitole)

Aby se daly rovnou na výstupu funkce zjistit všechny podstatné informace, kopíruje **MeLine** aktuální stavové bity do své **výstupní hodnoty** na pozice bitů 0., 1. a 2. :

MeLine?0 = Me_DISP *(povoleno zobrazení na displej)*

MeLine?1 = Me_SELECT *(položka navíc vybrána ukazovátkem)*

MeLine?2 = Me_PRESS *(na položce navíc právě stisknuta klávesa ENT)*

Logicky platí, že pokud je bit **Me_DISP** v nule (položka není zrovna na displeji), jsou nulové i ostatní bity. Pokud je tedy nastaven **Me_SELECT** nebo **Me_SELECT** a **Me_PRESS**, musí být současně nastaven i **Me_DISP**. Proto tedy k určení, zda je daný řádek na displeji, stačí zjednodušené porovnání výstupu funkce ve stylu nula/nenula.

```
; A) zjednodušený test na nenulovost   :
if MeLine("pocet kusu") then DISPLAY(PocetKS)

; B) totéž porovnání, zapsané jiným způsobem:
if MeLine("pocet kusu")<>0 then DISPLAY(PocetKS)

; C) test na hodnotu 0. bitu výstupu funkce (informace Me_DISP):
if MeLine("pocet kusu")?0 then DISPLAY(PocetKS)

; D) použití stavového bitu namísto výstupní hodnoty:
MeLine("pocet kusu")
if Me_DISP then DISPLAY(PocetKS)
```

V další ukázce použijeme informaci o stisku klávesy ENT na vybraném řádku (např. pro přímé nastavení bitu nebo vykonání nějaké akce):

```
; PROVEDENÍ AKCE PO STISKU KLÁVESY "ENT" NA POLOŽCE
; A) využití bitu Me_PRESS
MeLine("MOTOR SPUSTIT")
if Me_PRESS then VYSTUP_MOTOR=1
MeLine("MOTOR ZASTAVIT")
if Me_PRESS then VYSTUP_MOTOR=0
...

; B) testování 2.bitu výstupní hodnoty funkce MeLine
if MeLine("SPUSTIT MOTOR")?2 then VYSTUP_MOTOR=1
if MeLine("ZASTAVIT MOTOR")?2 then VYSTUP_MOTOR=0
...

; C) kombinované využití informací o zobrazení i stisku ENT
; jednak pro tisk aktuálního stavu, jednak pro provedení akce
if MeLine("MOTOR ") then begin
  if VYSTUP_MOTOR then DISPLAY("ZASTAVIT") else DISPLAY("SPUSTIT")
  if Me_PRESS then !VYSTUP_MOTOR       ; invertovat výstup při stisku
end
...
```

MeNext

Vytvoření položky typu DALŠÍ MENU

```
function bit MeNext(const string txt)
```

Předávané parametry

1.parametr (const string): text položky

Výstupní hodnota

výstup (bit): =0 ... normální běh, =1 ... vnoření do podmenu

Použití

Položka **MeNext** se používá pro vnořování do dalších úrovní menu. Při běžném listování v menu položka **MeNext** zobrazuje na řádku text, předaný jako parametr. Za textem je přidán symbol šipky, avizující možnost přechodu do dalšího menu. Dojde-li na této položce ke stisku klávesy ENT, aktivuje se další úroveň menu a funkce dává na svém výstupu hodnotu 1. K návratu zpět do vyšší úrovně dojde po stisku klávesy ESC.

MeNext tvoří jakousi výhybku ve struktuře menu. Předpokládá se, že program menu použije funkci **MeNext** k rozvětvení běhu programu na dvě části (konstrukcí if-then). Není-li podmínka splněna (**MeNext**=0), zůstává program ve stávající úrovni. Je-li **MeNext**=1, znamená to přechod do další vnořené úrovně.

Příklad použití

```
MeTitle("HLAVNI UROVEN")        ; hlavní blok menu
if MeNext("vstup do 2.urovne") then begin
  MeTitle("2.UROVEN")           ; blok menu 2.úrovně
  if MeNext("vstup do 3.urovne") then begin
    MeTitle("3.UROVEN")         ; blok menu 3.úrovně
    MeEnd()
  end
  MeLine("jsme v 2.urovni")     ; tady pokračuje blok 2.úrovně
  MeEnd()
end
...                             ; tady už pokračuje hlavní blok menu
```

K uvedenému příkladu:

Jsme např. v hlavní úrovni, listujeme v menu, všechny funkce **MeNext** v hlavní úrovni dávají hodnotu 0 a program nezabíhá do žádných větví, podmíněných funkcemi **MeNext**. Vyberme položku vytvořenou funkcí **MeNext**("vstup do 2.urovne") a stiskněme ENT. Dojde k vnoření - funkce **MeNext**("vstup do 2.urovne") od tohoto okamžiku nezobrazuje text, dává každém průchodu hodnotu 1 a všechny předcházející funkce jsou uvnitř "odstaveny". Veškeré zobrazení a ovládání teď začíná funkcí **MeTitle**("2.UROVEN") a končí na konci bloku funkcí **MeEnd**. Analogicky dalším stiskem ENT na položce dané funkcí **MeNext**("vstup do 3.urovne") se vnoříme ještě dále, tato funkce **MeNext** bude dávat od tohoto okamžiku výstupní hodnotu 1 a "zapnuty" budou až funkce následující za ní. Každý stisk ESC pak vrací hodnotu poslední aktivované výhybky **MeNext** do stavu 0 a vrací řízení do předchozí úrovně.

Stavové proměnné

Funkce nastavuje stavové proměnné: bit **Me_DISP**, bit **Me_SELECT**

Má-li být na řádku s položkou typu **MeNext** zobrazena ještě další informace, hodnota, apod., je třeba k tomu použít stavový bit **Me_DISP**. K položce typu **MeNext** nelze přidat editor, protože stisk ENT je na této položce vyhrazen pro vnoření do dalšího menu.

Jak bylo uvedeno, za text položky tiskne **MeNext** symbol šipky. Tisk další informace by tedy následoval až za šipkou. Nastavením stavové proměnné **Me_SYMBOLOFF** do 1 je možno vypnout tisk šipky ve funkcích **MeNext**. Chceme-li být precizní, můžeme pak dále popsanou funkcí **MeDispArrow** dodatečně šipku dotisknout.

Pozn.: Stavové bity se nastavují jen tehdy, je-li **MeNext**=0. V opačném případě se jedná o vnoření a stavové bity budou záviset na stavu dalšího menu.

```
; Dotisk stavu kotle (TOPI/NETOPI) přímo na řádek s položkou MeNext
; Test na Me_DISP a podmíněné provedení tisku je třeba provést
; až za konstrukcí "if MeNext.. then ..", protože pokud je MeNext=1,
; nezobrazuje se vlastní text položky, ale vnořené menu
Me_SYMBOLOFF=1            ; vypnutí symbolu šipky
if MeNext("KOTEL 2, ") then begin
  MeTitle("OKRUH KOTLE 2")
  if MeLine("Teplota: ") then DISPLAY(Kot2_Teplota)
  if MeLine("Vykon:   ") then DISPLAY(Kot2_Vykon)
  MeEnd()
end
if Me_DISP then begin
  if Kot2_ZAP then DISPLAY("TOPI  ") else DISPLAY("NETOPI")
end
Me_SYMBOLOFF=1            ; opětné zapnutí symbolu šipky
```

4.2. Editory hodnot

Slouží k editaci proměnných. Je jich několik typů pro různé případy. Jsou přizpůsobeny k snadnému začlenění do položek menu, tvořených funkcí **MeLine**. V každé položce může být jen jeden editor, počet položek s editory nicméně není nijak omezen.

Všechny editory mají jednotnou filozofii ovádání. Při běžném procházení položkami v menu zobrazuje každý editor hodnotu proměnné, která je mu předávána jako parametr (pracuje v ***zobrazovacím režimu***). Při stisku klávesy ENT na vybrané položce s editorem přejde dotyčný editor do ***editačního režimu*** a v zobrazované hodnotě se v levé části objeví blikající kurzor (ostatní editory zůstávají v ***zobrazovacím režimu***). Šipky nyní neovládají pohyb v menu, ale editaci. Šipky nahoru/dolů mění hodnotu, u číselných editorů **MeEdit** navíc šipky vlevo/vpravo přecházejí po řádech čísla. Editaci lze ukončit klávesou ESC nebo ENT. ESC stornuje všechny provedené změny a editovaná proměnná si podrží původní stav. ENT potvrdí nový stav a editovaná proměnná se přepíše novou hodnotou.

POZN.: Hodnota měnící se během editace je udržována ve vnitřní proměnné editoru. Předaná editovaná proměnná se tedy v průběhu editace nemění. K její eventuální změně dojde jen tehdy, je-li editace potvrzena klávesou ENT.

MeEdit

Číselný editor proměnných
pro datové typy BYTE, WORD, INT, LONGWORD, LONGINT a FLOAT

```
subroutine MeEdit(var byte iodata : byte emin : byte emax)
subroutine MeEdit(var word iodata : word emin : word emax)
subroutine MeEdit(var int iodata : int emin : int emax)
subroutine MeEdit(var longword iodata : longword emin : longword emax)
subroutine MeEdit(var longint iodata : longint emin : longint emax)
subroutine MeEdit(var float iodata : float emin : float emax)
```

Předávané parametry

1.parametr (var data_typ): editovaná proměnná

2.parametr (data_typ): dolní limit editované hodnoty

3.parametr (data_typ): horní limit editované hodnoty

Všechny 3 parametry jsou stejného typu, editory jsou k dispozici pro datové typy **byte**, **word**, **int**, **longword**, **longint**, **float**. Název procedury je stále stejný, správný druh vybere překladač podle typu parametrů, které jsou do procedury předány.

Funkce a ovládání editoru

Kromě jednotného způsobu ovládání, popsaného v úvodu k editorům, mají funkce **MeEdit** ještě několik specifických vlastností.

Šipkami vpravo/vlevo lze přecházet po řádech čísla (je-li číslo vícemístné). Šipkami nahoru/dolů se mění číslice na daném řádu vždy o 1, přičemž funguje přenos do vyšších i nižších řádů. Stručně řečeno: Stojí-li kurzor na jednotkách, pak opakovaný stisk nebo držení šipky nahoru/dolů neustále přičítá/odčítá jedničku. Podobně stojí-li kurzor na desítkách, přičítáme/odčítáme desítku. U znaménkových typů zvětšování/zmenšování hodnoty plynule pracuje i přes nulu. Je-li nastaveno zobrazení se znaménkem, je možno posunout kurzor až na znaménko a šipkou nahoru/dolů měnit znaménko.

U datových typů FLOAT (jsou-li zobrazeny v exponenciálním tvaru) lze dokonce posunout kurzor do oblasti exponentu a editovat tak separátně i exponent.

Při nastavování hodnoty směrem nahoru nebo dolů funguje dolní a horní limit, hodnotu je možno editovat pouze v zadaných mezích. V zobrazovacím režimu je proměnná interpretována tak, jak je, bez ohledu na limity. Po vstupu do editačního režimu se zobrazovaná hodnota přizpůsobí nastaveným limitům, nicméně editaci je pak nutno potvrdit stiskem ENT, aby se změna skutečně přepsala do editované proměnné.

Použití

Editor je uzpůsoben k začlenění do položky typu **MeLine**. Editor sám přejde ze zobrazovacího do editačního režimu při detekci bitu **Me_PRESS**. Editovanou proměnnou (1.parametr) editor po potvrzení editace sám změní. To je možné díky předání parametru odkazem (pozn.: klíčové slovo **var** v deklaraci hlavičky procedury). Nevýhodou tohoto jinak elegantního řešení je komplikace při editaci síťových proměnných - viz dále.

Formát zobrazení

Tvar zobrazení čísla je dán nastavením systémové proměnné **FORMAT**. Lze tedy editovat čísla zobrazená ve všech formátech, podporovaných automaty MICROPEL, tedy např. i v hexadecimálním tvaru. Nevhodná volba formátu však může přinášet komplikace.

Chceme-li např. editovat znaménkové proměnné i s možností snadné změny znaménka, měl by mít FORMAT nastaveno zobrazení znaménka "**+**". Pak je možné kdykoliv kurzorem najet na znaménko. Není-li tomu tak, pak u záporných hodnot znaménko je a lze se na něj kurzorem dostat, ale u kladných hodnot ne.

Další komplikace mohou být u proměnných typu FLOAT. Aby byla možná jejich bezproblémová editace, měl by být FORMAT nastaven na takový počet desetinných míst, který má být editovatelný (jen tak je zajištěno, že požadovaná desetinná místa budou vždy zobrazena a tudíž přístupná k editaci). Má-li být u proměnných typu float editovatelný i exponent (vhodné pro změny čísel ve velkých rozsazích), je třeba nastavit FORMAT na exponenciální zobrazení čísla.

Obecně při editaci proměnných typu FLOAT bychom neměli používat automatický formát (nebo základní univerzální nastavení FORMAT=0). Tak, jak se mění velikost čísla, mění se automaticky i způsob jeho vyjádření a to je pro editaci nevhodné.

Příklad použití

```
; A) Jednoduché začlenění editoru do položky MeLine
; (celočíselná znaménková proměnná, 1 des.místo, limity -50.0/+50.0)
FORMAT=0x1201
if MeLine("Korekce(sec): ") then MeEdit(KorekceCasu, -500, 500)

; B) Totéž, ale s dotiskem jednotek až za číslem
if MeLine("Korekce: ") then begin
  FORMAT=0x1241           ; rezervovat pevně 4 znaky pro číslo
  MeEdit(KorekceCasu, -500, 500)
  DISPLAY(" sec")
end
```

Síťové proměnné

Protože editovaná proměnná je předávána odkazem, aby ji editor mohl sám změnit, ***není bohužel možné procedurou MeEdit přímo zpracovávat síťové proměnné*** (síťovou proměnnou totiž nelze předat odkazem). Tento nepříjemný nedostatek lze obejít například použitím pomocné proměnné zhruba takto:

```
; Editace síťové proměnné  "Posun" (umístěné např. na D32)
var word pomocny          ; založení pomocné proměnné typu word

if MeLine("Posunuti: ") then begin
  pomocny=Posun           ; přesunutí síťové proměnné do pomocné
  MeEdit(pomocny, 0, 100) ; editor pracuje s pomocnou proměnnou
  if pomocny<>Posun then Posun=pomocny
                          ; zápis do síťové proměnné, došlo-li ke změně
end
```

MeSelector

Editor indexu pro textové zobrazení stavů

```
function byte MeSelector(var byte iodata : byte min : byte max)
```

Předávané parametry

1.parametr (var byte): editovaná proměnná
2.parametr (byte): dolní limit
3.parametr (byte): horní limit

Výstupní hodnota

výstup (byte): průběžný stav hodnoty

Použití

Pro práci s proměnnými, které nabývají omezeného množství hodnot (editovaná proměnná i limity jsou typu byte). Typickým příkladem jsou proměnné, popisující nějaký stav, přičemž možných stavů je definované množství a mají nějaké slovní pojmenování.

Editor sám o sobě kromě pozice pro kurzor nezobrazuje nic, jen korektně nastavuje proměnnou POSITION a poskytuje na výstupu průběžnou editovanou hodnotu, tak jak se mění během editace. Předpokládá se použití této výstupní hodnoty jako indexu do tabulky textů, které se pak zobrazují standardní funkcí DISPLAY. Během editace se pak na displeji namísto čísel objevují textové názvy, odpovídající jednotlivým hodnotám editované proměnné. Vlastní zápis do editované proměnné (té, která se předává do funkce jako 1.parametr) se ale samozřejmě provede až po potvrzení klávesou ENT.

Při tvorbě textů je třeba počítat s tím, že editor vytvoří na displeji jeden prázdný znak (před tiskem textu) pro případný blikající kurzor, označující režim editace.

Ovládání

Editor má jednotný způsob ovládání, popsaný v úvodu k editorům, změna hodnoty se provádí pouze šipkami nahoru/dolů. Je tu jedna odlišnost proti číselným editorům - po dosažení dolního/horního limitu se hodnota nezastaví, ale "točí" se dokola. Tedy při podržení šipky všechny názvy neustále defilují za sebou kolem dokola.

Příklad použití

Mějme např. proměnnou ***Rezim***, která nabývá hodnot 0,1,2,3 a ty odpovídají stavům VYPNUTO, KONST, UTLUM a EKVITERM. Uvedený příklad zobrazuje text z tabulky, kde jako index do tabulky je použita rovnou výstupní hodnota funkce **MeSelector** a tak je v zobrazovacím i editačním režimu umožněno zobrazení textových názvů odpovídajícím jednotlivým hodnotám proměnné Rezim. ***Pozn.:*** Bude-li výstupní hodnota větší než velikost tabulky, operační systém automatu automaticky nastaví index na 0.

```
; Editace proměnné "Rezim", která nabývá těchto stavů:
; VYPNUTO, KONST, UTLUM, EKVITERM
                  ; definice textových názvů jednotlivých stavů :
table string[4] NazvyStavu = ("VYPNUTO","KONST","UTLUM","EKVITERM")
                  ; zobrazení textů na displeji
if MeLine("rezim UT: ") then DISPLAY(NazvyStavu[MeSelector(Rezim,0,3)])
```

MeBitSelector

Editor bitu s textovým zobrazením stavů

```
function bit MeBitSelector(bit bb : const Me_tbl2txt txt)
```

Předávané parametry

1.parametr (bit): bit k editaci

2.parametr (table string[2]): tabulka textů o 2 prvcích - názvy stavů

Výstupní hodnota

výstup (bit): pokyn k invertování editovaného bitu

Použití

Pro editaci bitu. Stav bitu je na displeji zobrazen ve formě textu, proto se jako parametr do funkce, kromě editovaného bitu, předává i odkaz na pole dvou textů (kde na položce 0. je text pro hodnotu bitu=0 a na položce 1. text pro bit=1). ***Pozn.:*** datový typ Me_tbl2txt, uvedený v deklaraci funkce je vytvořen jen kvůli možnosti předání konstantního pole do funkce a není třeba se s ním nijak zabývat. Dvojici požadovaných textů je třeba zadefinovat standardní konstrukcí:

table string[2] *název_pole* = ("*text_0*", "*text_1*")

Při tvorbě textů je třeba počítat s tím, že editor vytvoří na displeji jeden prázdný znak (před tiskem textu) pro případný blikající kurzor, označující režim editace.

Protože bit nelze do funkce předávat odkazem, není tento editor schopen sám provést změnu bitu předaného na místě 1.parametru. K tomuto účelu dává funkce **MeBitSelector** na výstupu pokyn k invertování tohoto bitu. Výstup bude mít hodnotu 1 jen tehdy, pokud bude editace potvrzena klávesou ENT a pokud tím zároveň dojde i ke změně původní hodnoty bitu. Proto se bitový editor musí v programu použít zcela specifickým způsobem, ukázaným níže v příkladu použití.

Ovládání

Editor má jednotný způsob ovládání, popsaný v úvodu k editorům, změna hodnoty se provádí pouze šipkami nahoru/dolů. Protože editor bitu zná jen dva stavy, při stisku jakékoliv šipky se hodnota překlápí.

Příklad použití

```
; Editace bitu "VETRANI", který nabývá stavů: "VYPNUTO" a "ZAPNUTO"

                  ; definice textových názvů jednotlivých stavů :
table string[2] VypZapText = ("VYPNUTO","ZAPNUTO")

                  ; použití bitového editoru
if MeLine("vetrani: ") then begin
  if MeBitSelector(VETRANI,VypZapText) then !VETRANI
end
...
```

MeSetRTC

Nastavení reálného času automatu

```
subroutine MeSetRTC()
```

Použití

Umožňuje kompletní nastavení všech registrů reálného času automatu obsluhou z klávesnice. Tuto proceduru je vhodné začlenit do všech programů, které nějakým způsobem používají reálný čas. Obvod RTC v automatu má samozřejmě omezenou přesnost a po nějakém čase se jeho údaje rozejdou se skutečností. Pokud není korekce RTC řešena jinak (např. z osobního počítače), je třeba dát obsluze k dispozici alespoň tento prostředek.

Procedura **MeSetRTC** je již hotové předpřipravené samostatné menu, které je možno kamkoliv začlenit a které má několik položek na nastavení roku, měsíce, data, hodin, minut, sekund a dne v týdnu. Jako nadpis menu se zobrazuje aktuální stav RTC.

Procedura nemá žádné vstupní parametry a nedává ani žádnou výstupní hodnotu. Uvnitř procedury je na konci již obsaženo volání ukončovače **MeEnd** na zakončení bloku menu. Předpokládá se použití ve formě vnořeného menu, začleněného do položky typu **MeNext**.

Příklad použití

```
; Začlenění menu pro nastavení RTC :

...
if MeNext("REALNY CAS: ") then MeSetRTC()
...
```

4.3. Zobrazovací funkce

Zobrazení číselných hodnot i textů z tabulek zajistíme standardní funkcí DISPLAY a nastavením požadovaného formátu zobrazení v proměnné FORMAT:

```
; Zobrazení čísla (float, fix point, 3 des.místa) :
FORMAT=0x1403
if MeLine("koeficient: ") then DISPLAY(Koef)
FORMAT=0
...
; Zobrazení stavu (textově, z tabulky, podle stavu proměnné Rezim) :
if MeLine("rezim: ") then DISPLAY(NazvyStavu[Rezim])
```

Dále je popsáno několik funkcí pro složitější případy zobrazení, kdy nestačí pouhý jednoduchý tisk pomocí funkce DISPLAY. Jsou to funkce pro zobrazení data/času, zobrazení stavu bitu v textové formě a ovládání tisku symbolu šipky u funkcí **MeNext**.

Funkce zobrazují na displej od pozice dané momentálním nastavením proměnné POSITION a nijak nesouvisejí s funkcemi běhu menu, lze je tedy využít i mimo menu, v libovolném uživatelském programu.

MeDispBitText

Zobrazení stavu bitu textem

```
subroutine MeDispBitText(bit bb : const Me_tbl2txt txt)
```

Předávané parametry

1.parametr (bit): bit pro zobrazení

2.parametr (table string[2]): tabulka textů o 2 prvcích - názvy stavů

Použití

Zobrazení stavu bitu ve formě textu.

Typ předávaných parametrů, jejich význam a zadefinování tabulky textů jsou přesně stejné jako u výše popsaného editoru **MeBitSelector**. Je tedy možné využít předdefinované tabulky dvojic textů jak pro **MeDispBitText**, tak pro **MeBitSelector**.

Příklad použití

```
; Zobrazení stavu bitu SMER :
                  ; použití bitového editoru
table string[2] Smer_Text = ("VLEVO","VPRAVO")
                  ; použití zobrazovače stavu bitu v položce:
if MeLine("smer pohybu: ") then MeDispBitText(SMER,Smer_Text)
...
```

MeDispArrow

Zobrazení symbolu šipky

```
subroutine MeDispArrow()
```

Použití

Vytiskne na displej symbol šipky na aktuální pozici POSITION. Pokud se v programu používají funkce **MeNext** s vypnutou šipkou (**Me_SYMBOLOFF**=1) a s příp. dotiskem dalších informací za text položky, může být vhodné za ně šipku dotisknout. To se provede pouhým zavoláním této funkce.

Příklad použití

```
; Doplněný příklad, uvedený u popisu funkce MeNext:

Me_SYMBOLOFF=1            ; vypnutí symbolu šipky
if MeNext("KOTEL 2, ") then begin
  MeTitle("OKRUH KOTLE 2")
  ....
  MeEnd()
end
if Me_DISP then begin
  if Kot2_ZAP then DISPLAY("TOPI  ") else DISPLAY("NETOPI")
  MeDispArrow()           ; dotisknutí šipky
end
Me_SYMBOLOFF=1            ; opětné zapnutí symbolu šipky
```

MeDispRTC

Zobrazení reálného času

```
subroutine MeDispRTC(byte mode)
```

Předávané parametry

1.parametr (byte): mód zobrazení

Použití

Zobrazení reálného času. Mód zobrazení svými prvními pěti bity určuje, co se na displeji zobrazí. Není-li nastaveno zobrazení času ani data, nezobrazí se nic. V tabulce je uvedena funkce jednotlivých bitů i ukázky možných módů zobrazení :

mód zobrazení	
bit	funkce
4.	zobrazení data
3.	zobrazení času
2.	datum včetně roku
1.	rok v dlouhém formátu
0.	čas včetně sekund

mód	ukázka	mód	ukázka
0		24	27.09. 18:05
8	18:05	25	27.09. 18:05:30
9	18:05:30	28	27.09.04 18:05
16	27.09.	29	27.09.04 18:05:30
20	27.09.04	30	27.09.2004 18:05
22	27.09.2004	31	27.09.2004 18:05:30

Příklad použití

```
; A) Použití reálného data/času přímo v nadpisu menu
if MeTitle("KOTELNA ") then MeDispRTC(24)   ; tisk krátkého data/času

; B) Příklad zobrazení úplně mimo menu, např. na volný nevyužitý řádek
POSITION=120      ; tisk na 4.řádek
MeDispRTC(31)     ; tisk kompletní informace

; C) Začlenění do položky menu
if MeLine("Dnes je: ") then MeDispRTC(22)   ; tisk data a roku
```

4.4. Funkce pro zobrazování seznamů

Všechny dosud popsané funkce počítají s bloky menu vytvořenými z jednotlivých různých položek typu **MeLine** nebo **MeNext**, volaných po sobě. Kolik položek napíšeme v programu za sebou, tolik jich pak v menu je.

Často je však třeba vytvořit dlouhý seznam nebo dokonce tabulku položek stejného typu, ale s předem neznámým nebo proměnným počtem položek (například seznam prvků pole, které se liší jen indexem). Tuto úlohu řeší zde popsané funkce.

MeTable

Vytvoření řady položek

```
subroutine MeTable(var word iy : word ysize)
```

Předávané parametry

1.parametr (var word): proměnný index osy Y (index do seznamu)

2.parametr (word): počet položek v ose Y (délka seznamu)

Použití

Řazení položek za sebou a separátní pojmenování každé položky menu textovým názvem, tak jak bylo ukázáno v úvodním příkladu, vyhovuje pro menší počet navzájem různorodých položek. Potřebujeme-li však vytvořit velmi dlouhý blok stejných nebo velmi podobných položek (chceme například editovat pole padesáti proměnných stejného typu), je to tímto způsobem nešikovné nebo dokonce neproveditelné.

Procedura **MeTable** tvoří jakýsi "opakovač" položek. Používá se na začátku seznamu položek (až za případným nadpisem **MeTitle**) a za ní může být jen jedna jediná položka - buď typu **MeLine** anebo **MeNext**. Pak musí následovat ukončovač **MeEnd**.

Jediné, co takto vzniklou řadu položek navzájem odlišuje, je proměnný index. Tak, jak procházíme šipkami nahoru/dolů, mění procedura **MeTable** proměnnou na místě 1.parametru - index osy Y (je to analogické se svislým pohybem ve sloupci tabulky).

Procedura připraví v každém průchodu programu podmínky pro zpracování jedné položky v následné funkci **MeLine** nebo **MeNext**. Podmínky se cyklicky mění tak, aby se na všechny dostupné řádky displeje vytiskly patřičné položky. ***Např.:*** jsou k dispozici 3 řádky displeje a zobrazuje se tabulka od 7. do 9. řádku - pak se volá **MeLine** nebo **MeNext** s indexem 7 pro 1.řádek, 8 pro 2.řádek, 9 pro 3.řádek.. a tak stále dokola.

Příklad použití

```
; Editace jednoduchého seznamu o 10 položkách
;-------------------------------------------------------------
var word[10] CasyOhrevu   ; zadefinování pole s parametry časů
var word Index            ; index do seznamu

if RESET then Index=0     ; počáteční inicializace
             ; samostatné vnořené menu s tabulkou:
if MeNext("Nastaveni parametru: ") then begin
  MeTitle("DOBA OHREVU")
  MeTable(Index, 10)      ; vytvoření seznamu o 10 položkách
  if MeLine("proces ") then begin
    DISPLAY(Index)
    DISPLAY(": ")
    MeEdit(CasyOhrevu[Index],0,600)  ; editace parametru v rozmezí 0-600
  end
  MeEnd()
end
...
```

Použitím doplňku **MeXShift** (viz dále) snadno vznikne dvourozměrná tabulka, v níž je možný pohyb všemi směry po řádcích i sloupcích. V tabulce funguje i vnoření do dalších menu, použijeme-li položky **MeNext**. Ve vnořeném menu pak můžeme použít cokoli, tedy i další tabulku a pracovat tak s tabulkou tabulek....

MeXShift

Doplněk k proceduře MeTable - horizontální pohyb po sloupcích tabulky

```
subroutine MeXShift(var word ix : word xsize)
```

Předávané parametry

1.parametr (var word): proměnný index osy X (index sloupce)

2.parametr (word): počet položek v ose X (počet sloupců)

Použití

Doplněk k proceduře **MeTable**, umožňující rozšíření tabulky na více sloupců, tedy rozšíření z jednorozměrné tabulky na dvourozměrnou. Zařazuje se vždy těsně za **MeTable**, za **MeXShift** pak již následuje vytvoření položky **MeLine** nebo **MeNext**.

Šipkami vlevo/vpravo přecházíme mezi jednotlivými sloupci - procedura mění proměnnou na místě 1.parametru - index osy X. Parametr 2. nastavuje horizontální rozměr tabulky - počet sloupců. Procedury **MeTable** a **MeXShift** tedy společně zajistí ovládání indexů X i Y a pohyb po řádcích i sloupcích pomocí šipek nahoru, dolů, vlevo a vpravo.

Příklad použití

Pro řadu stejných procesů nastavujeme např.: dobu zapnutí, dobu vypnutí, dobu ohřevu :

```
; Editace tabulky parametrů (2-rozměrného pole) o velikosti 3x10 prvků
;-------------------------------------------------------------------------
    ; 1.index (0..2) vybírá zapnutí/vypnutí/ohřev, 2.index je č.procesu:
var word[3][10] Casy
var word TabX, TabY   ; proměnné pro indexy X a Y do tabulky
    ; pojmenování sloupců tabulky (pro lepší přehlednost):
table string[3] TextSloupce = ("ZAPNUTI", "VYPNUTI", "OHREVU")

if RESET then begin
  TabX=0          ; počáteční inicializace proměnných pro osy X a Y
  TabY=0
end
    ; vnořené menu pro editaci všech parametrů v tabulce:
if MeNext("Nastaveni parametru: ") then begin
  if MeTitle("DOBA ") then DISPLAY(TextSloupce[TabX])
  MeTable(TabY, 10)                ; vytvoření seznamu o 10 položkách
  MeXShift(TabX, 3)                ; vytvoření 3 sloupců tabulky
  if MeLine("proces ") then begin
    DISPLAY(TabY)
    DISPLAY(": ")
    MeEdit(Casy[TabX][TabY],0,600)
  end
  MeEnd()
end
```

MeScroll

Podpora pro rolování textů na displeji

```
function byte MeScroll(byte max : byte interval)
```

Předávané parametry

1.parametr (byte): maximální hodnota indexu

2.parametr (byte): interval automatického rolování (x 10 ms)

Výstupní hodnota

výstup (byte): proměnný index

Použití

Funkce je primárně určena pro výpis bloku textů na displej. Zařazuje se do samostatného bloku menu, ve kterém by kromě ní už neměly být žádné další položky.

Funkce využije zbývající volné řádky pro menu a cyklicky nastavuje pozici pro tisk na displej v proměnné POSITION tak, aby v každém průchodu programu ukazovala na jeden z volných řádků displeje. Zároveň poskytuje na výstupu proměnný index, odpovídající danému řádku. Tím je zajištěno, že na všechny řádky využitelné pro zobrazení položek, se zobrazí příslušný počet textových řádků.

Index se dále mění při stisku šipek nahoru/dolů (tak lze posunovat celým textem po displeji). Hodnota indexu, který může funkce na výstupu vygenerovat, je od nuly do hodnoty ***max*** (specifikováno 1.parametrem funkce).

Parametr 2. nastavuje režim automatického rolování - index se bude zvyšovat vždy po nastaveném časovém intervalu (v násobcích 10ms). Je-li 2.parametr nulový, automatické rolování je vypnuto. Pokud je během automatického rolování stisknuta nějaká klávesa, automatika se vypíná a od toho okamžiku již lze rolovat jen ručně.

Příklad použití

```
     ; zadefinování tabulky textů se 6 řádky (index do pole je tedy 0..5)
table string[6] BlokText = (
  "Ukazka rolovani",
  "blokoveho textu.",
  "Textove radky",
  "jsou umisteny",
  "v tabulce textu.",
  "--------------------")

if MeNext("S aut.rolovanim") then begin
     ; generování indexu s max. hodnotou 5 a aut.posuvem po 500ms
  DISPLAY(BlokText[MeScroll(5,50)])
  MeEnd()
end
if MeNext("S ruc.rolovanim") then begin
     ; generování indexu s max. hodnotou 5, bez aut.posuvu
  DISPLAY(BlokText[MeScroll(5,0)])
  MeEnd()
end
```

4.5. Pomocné funkce

Zde je zbytek funkcí, které nezapadají do žádné z předchozích kategorií. Představují různé užitečné pomůcky, využitelné při vytváření různých systémů ovládání pomocí knihoven MenuLIB.

MeSize

Průběžná změna pozice a délky menu

```
subroutine MeSize(byte StartLine : byte MenuLen)
```

Předávané parametry

1.parametr (byte) udává řádek displeje, kde menu začíná (0..3)

2.parametr (byte) udává počet řádků displeje, na kterých se menu rozvíjí (1..4)

Použití

Proceduru je možné zařadit kamkoliv do struktury programu menu (i vícekrát). Nastavuje pozici a délku menu podobně jako **MeInit**. Nastavené hodnoty platí do konce právě aktivního bloku menu nebo do dalšího volání **MeSize**. Vhodná např. tehdy, když je třeba některé části menu nastavit jiný rozměr, nebo na některé řádky zobrazovat informace pomocí vlastních programových konstrukcí (ne vždy se totiž způsob tisku daný funkcemi MenuLIB programátorovi hodí).

Příklad použití

Úryvek programu pro 4-řádkový displej v podmenu "***Katalog poruch***" lokálně zmenší rozměr menu ze 4 řádků jen na 2. Funkce ***ZobrazStavVZT***, volaná z tohoto menu, vypisuje informace na spodní 2 řádky. Protože je rozměr zvolen tak, aby byla zobrazena vždy jen 1 položka, budou k ní vždy zobrazeny ještě 2 řádky ze ***ZobrazStavVZT***. Šipkami pak budeme de-facto listovat ne po řádcích ale po celých obrazovkách.

```
     ; Procedura pro tisk informací z parametru "stav" na 3.a 4.řádek
subroutine ZobrazStavVZT(byte stav)
  POSITION=80
  if stav?0 then DISPLAY("TOPENI ") else DISPLAY("CHLAZENI ")
  if stav?1 then DISPLAY("V BEHU") else DISPLAY("STOJI")
  POSITION=120
  if stav?2 then DISPLAY("FILTR! ")
  if stav?3 then DISPLAY("PREHRATI!")
return

     ; Začátek hlavního menu
MeInit(0,4)       ; nastaven rozměr 4 řádky, odshora, tedy celý displej
if MeNext("Katalog poruch") then begin
  MeSize(0,2)     ; lokálně nastaveny jen horní 2 řádky, tedy 1.a 2.řádek
  MeTitle("KATALOG PORUCH")  ; nadpis na 1 řádek, jen 1 zbývá na položky
  if MeLine("VZT Prizemi") then ZobrazStavVZT(StavVZT_Prizemi)
  if MeLine("VZT Garaze") then ZobrazStavVZT(StavVZT_Garaze)
  if MeLine("VZT Kancelare") then ZobrazStavVZT(StavVZT_Kanc)
  MeEnd()
end   .......
```

MeWatch

Hlídání aktivity obsluhy z klávesnice

```
function bit MeWatch(word sec)
```

Předávané parametry

1.parametr (word) dovolený interval nečinnosti (v sekundách)

Výstupní hodnota

výstup (bit): =1 ... obsluha menu je aktivní, =0 ... neaktivní

Použití

Hodí se zejména k vytvoření "spořiče obrazovky", nebo zobrazení nějakých přehledových informací, když obsluha nějaký čas nesahá na klávesnici.

Po zapnutí automatu a po stisku každé klávesy se vždy nastaví interní časovač v knihovně MenuLIB na interval daný 1.parametrem. Výstup funkce **MeWatch** je 1. Pokud po celou dobu intervalu (v sekundách) není stisknuta klávesa, přejde výstup do hodnoty 0. Stiskem klávesy se opět vrací na 1.

Pozn.:

Pokud je **MeWatch** ve stavu 0 (tedy stav "obsluha neaktivní"), potom funkce **MeWatch** "spolkne" první stisk klávesy který přijde (nastaví syst. proměnnou KBCODE=0). To proto, aby první stisk klávesy ve stavu, kdy je například zhasnuté podsvícení displeje, neprovedl nějakou nechtěnou akci. Aby toto fungovalo, musí být ovšem volání funkce **MeWatch** zařazeno na začátku menu nebo před ním.

Příklad použití

```
; A) Zhasínání podsvícení displeje po delší nečinnosti
...
MeInit(0,4)
Y30=MeWatch(60)   ; vypne podsvícení po nečinnosti delší než 60s
...

; B) Zhasínání displeje a zobrazení jiné obrazovky po delší nečinnosti
...
MeInit(0,4)       ; Inicializace menu
if MeWatch(60) then Y30=1      ; v aktivním stavu je rozsvícený displej
else begin                     ; v neaktivním stavu je:
  ZobrazKlidovyObr()               ; zobrazeno něco na displeji
  Y30=0                            ; zhasnuté podsvícení
  MeEnd()                          ; odstaven celý zbytek funkcí menu
end
if MeNext....     ; začátek hlavního menu
...
```