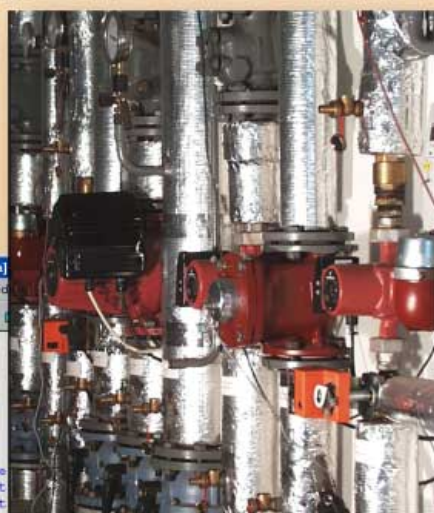


microPEL

MaR.LIB

KNIHOVNA SIMPLE4 PRO TVORBU APLIKACÍ MĚŘENÍ A REGULACE NA PLC MICROPEL 03.2005



The screenshot displays the StudioWin software interface. The main window shows a project tree on the left with 'Projekt PříkladMaR' selected. The central area contains a ladder logic program for a boiler control system, including sections for menu control, initialization, and various boiler parameters like temperature and pressure. The bottom status bar shows 'PříkladMaR' and 'PříkladMaR'.

On the right side of the screenshot, there is a photograph of a complex industrial control system, likely a boiler or steam engine, featuring various pipes, valves, and electrical components.

MaR.LIB

V1.0

Knihovní funkce v jazyce SIMPLE4 pro tvorbu aplikací v oblasti měření a regulace (MaR).

uživatelský manuál - edice 06.2005

2. verze dokumentu

změny proti 1. verzi dokumentu 03.2005 :

opraven popis poruchového slova na str.35

MaR.LIB V1

© Ing. Jaroslav Kurzweil 2005

MICROPEL s.r.o. 2005

všechna práva vyhrazena

kopírování publikace dovoleno pouze bez změny textu a obsahu

<http://www.micropel.cz>

OBSAH

1. Možnosti knihovny MaR.LIB	5
Parametrizace	5
2. Používání knihovny	6
2.1. Obsah distribuce, instalace	6
2.2. Začlenění knihovny do projektu	6
2.3. Prostředky a zdroje, použité funkcemi MaR	7
2.4. Další omezení	8
3. Princip funkce knihovny MaR	9
3.1. Pracovní blok dat	9
3.2. Připojování vstupů a výstupů, parametry	10
3.3. Generování a zpracování poruch	11
4. Seznam knihovnických funkcí	13
Popis pracovních bloků dat	13
4.1. Globální proměnné definované v knihovně MaR.LIB	14
Proměnné "MaR_sec" a "MaR_blik"	14
Předdefinované texty	14
4.2. Spojovací funkce	14
MaRZapisPar, MaRZapisParBit	15
MaRPrectiPar, MaRPrectiParBit	16
MaRTiskniPar	16
MaREditPar	17
MaRZpracujTep	18
4.3. Funkce pro vyhodnocování poruch	20
MaRPoruchovka	20
MaRDispPoruchy	23
4.4. Regulační funkce	26
MaRUT	27
MaREkviterm, MaRDispEkviterm	28
MaRKaskada	31
MaRPlynKotle	33
MaRHavStavy	37
MaRVentilator	40
MaRVZT	42
MaRTUV	49

MaRDrevokotel	51
4.5. Časové programy	54
MaRDispKalendar	56
MaRLineProfily	57
MaRLineBit, MaRLineHodnota	58
MaRSpravaProfilu	59
MaRStavKalendareB, MaRStavKalendareW	59
4.6. Pomocné funkce	60
MaRAI, MaRAO	60
MaRHavSig	62
MaRTiskVentil	63
MaRTiskTep	64
MaRTiskCas	64
MaREditCas	64
MaREditTep	64
MaRTiskDatum	65
MaRMaxOf	65
MaRMinOf	65
5. Parametrizace	66

1. Možnosti knihovny MaR.LIB

Pomocí knihovny je možné na automatech MICROPEL velmi jednoduše a efektivně vytvořit kompletní aplikaci MaR, včetně komfortního ovládacího rozhraní pro uživatele.

Knihovna řeší zejména různá zařízení plynové kotelny a vzduchotechniky (postupně přibývají další funkce) a také obzvláště výhodným způsobem nabízí centrální zpracování poruch a následné vygenerování sumárních poruchových stavů. Tyto sumární poruchové stavy mohou být využity pro blokování jednotlivých zařízení (to většinou nabízejí samy příslušné řídicí podprogramy, které se pro danou funkci jen parametrují), nebo pro vyhledávání poruch poruchovou signalizací (i tuto funkci knihovna nabízí, stačí ji jen použít a vhodně naparametrovat) nebo pro jakékoli speciální úkoly. Kromě těchto sumárních stavů jsou v automatu dostupné "výpisy" jednotlivých poruch s časem a datem jejich posledního výskytu. Navíc je k dispozici automaticky vytvářený archiv poruchových stavů, který standardně vytváří 200 chronologicky řazených záznamů poruchových událostí s datem, časem a popisem události.

Knihovní funkce počítají i tvorbou rozsáhlejších aplikací pro více automatů v síti a se vzájemnou komunikací mezi nimi (lze např. v síti rozdělit nebo naopak centralizovat katalog poruch apod.).

Parametrizace

Chování celého systému vytvořeného pomocí MaR.LIB není dáno jen zápisem programu, ale též nastavením mnoha funkčních parametrů. Tyto parametry jsou uloženy v datové paměti v oblasti zásobníku (STACK). Pro komfortní a přehledné nastavení parametrů všech knihovních funkcí pro celou aplikaci slouží připravená šablona ve formě XLS souboru pro program Microsoft® Excel. V tomto souboru je již vše připraveno pro kompletní nastavení všech potřebných dat a z něj se i provádí vlastní zatažení parametrů do paměti automatu. Výhodou tohoto řešení (krom vysoké přehlednosti) je i možnost jednoduché archivace vytvořených souborů parametrů a jejich pohodlné editace. Nutnou podmínkou pro zdárné oživení aplikace MaR je tedy instalace programu Microsoft® Excel na počítači.

2. Používání knihovny

Pro dokonalé využití knihovnických funkcí je nutná znalost programování v jazyce SIMPLE4 a alespoň základní znalost používání vývojového prostředí StudioWin. Pro úspěšné naparametrizování celé aplikace je dále nutná instalace programu MS Excel a znalost jeho ovládání. Protože knihovna **MaR.LIB** používá k vytvoření ovládacího uživatelského rozhraní funkce knihovny **MenuLIB**, je třeba instalovat do prostředí StudioWin obě tyto knihovny.

Proto, pokud teprve chcete začít s programováním automatů MICROPEL, doporučujeme před studiem knihovny nejprve prostudovat příručku k jazyku SIMPLE4 a zkusit si napsat ve vývojovém prostředí StudioWin několik jednoduchých příkladů. Kromě znalosti programování v SIMPLE4 je potřebná ještě alespoň základní znalost tvorby ovládacího menu pomocí knihovny MenuLIB.

Pro snazší pochopení 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 **MaR1.lib** ještě několik kompletních ukázkových projektů s touto knihovnou a je tam rovněž i knihovna **MenuLIB**.

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

Pro instalaci a snadné použití knihovny je třeba vývojové prostředí StudioWin verze min. 6.902 (s vylepšenou podporou pro instalaci a používání knihoven).

2.2. Začlenění knihovny do projektu

Před použitím funkcí knihovny je třeba ji zapojit do projektu.

Pro zdárnou funkci knihovny MaR je třeba společně s ní začlenit do projektu i knihovnu MenuLIB.

V pracovním okně projektu (na levé straně v prostředí StudioWin) lze začlenit soubor knihovny buď do složky knihovnických modulů celého projektu (nahore), nebo konkrétního automatu. Obě knihovny postupně 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 MaR

Protože knihovna používá víceméně stejné prostředky jako knihovna MenuLIB, platí pro její používání podobná omezení, jako pro knihovnu MenuLIB. Funkce knihoven MaR a MenuLIB jsou součástí výsledného uživatelského programu pro automat v jazyce SIMPLE4 a představují ucelený, logicky provázaný systém. Pro korektní funkci takového systému 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 řídicích bitů: TEN7, TOE7, TPA7, TDM7). Je to na druhou stranu ale jediný časovač, který systém knihoven využívá.

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 MaR nebo MenuLIB, zahlásí překladač duplicitní definici. V takovém případě je třeba zvolit nějaký jiný název.

Pro přehlednost začínají veškeré názvy proměnných i funkcí knihovny MaR písmeny **MaR** a knihovny MenuLIB písmeny **Me**.

Je tedy nejjednodušší volit názvy symbolů tak, aby nezačínaly na **MaR**, **mar**, **MAR**, **Me**, **me**, **ME** ... (překladač nerozlišuje velikost písmen).

Klávesnice

Kromě výše uvedených názvů jsou v knihovně MenuLIB 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 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 knihovnických funkcí (a pravděpodobně i celého programu).*

2.4. Další omezení

V uživatelském programu lze použít vždy jen jeden systém menu.

Bližší informace ke stavbě ovládacího rozhraní pomocí knihovny MenuLIB jsou obsaženy v uživatelské příručce k této knihovně.

3. Princip funkce knihovny MaR

V této kapitole popíšeme koncepci systému a společné charakteristiky všech funkcí. Teprve v konkrétním popisu jednotlivých funkcí budou uváděny případné výjimky proti následujícímu popisu.

Řídící procedury pro svoji práci potřebují tyto druhy dat:

- ☐ **vstupy a výstupy**
- ☐ **parametry**
- ☐ **vnitřní stavy**

Všechna tato data jsou pro každou knihovní funkci seřazena do souvislého bloku wordů (nazvěme ho např. "**Pracovní blok dat**") a umístěna v paměti automatu zvané zásobník (STACK). Pracovní blok dat může programátor umístit kamkoli do zásobníku. Pracovní blok má pro každou knihovní funkci jinou velikost i strukturu a jeho popis je předmětem popisu konkrétní funkce.

Dále všechny procedury či funkce generují "**Poruchové slovo**" a to ukládají rovněž na programátorem stanovenou adresu v zásobníku. Jednotlivá poruchová slova mají délku 4B (zaberou tedy dva wordy) a jsou v zásobníku ukládána bezprostředně za sebou za účelem dalšího zpracování. Specializovaná knihovní funkce pak může tato slova centrálně zpracovávat, vyhlášovat do systému poruchy a havárie a vést historii poruch.

Proto je do každé knihovní funkce třeba předat dva parametry:

- ☐ **AdrPar** - adresu zásobníku, od které začíná výše uvedený pracovní blok dat
- ☐ **AdrPor** - adresu zásobníku, kam bude funkce ukládat poruchové slovo.

Volání funkcí pak obvykle vypadá takto:

MaRxx(AdrPor,AdrPar)

kde MaRxx je identifikátor funkce (všechny identifikátory knihovny začínají zkratkou MaR, identifikátory proměnných této knihovny začínají vždy MaR_).

3.1. Pracovní blok dat

Jak bylo výše uvedeno, v pracovním bloku jsou data těchto tří kategorií:

vstupy a výstupy, parametry, vnitřní stavy

Vnitřní stavy

O vnitřní stavy se starají funkce samy, programátor se jimi nemusí zabývat. Naopak nešetrný zásah do těchto dat může mít za následek nesprávnou funkci, nebo dokonce kolaps příslušné knihovní funkce.

Parametry

Lze je definovat na začátku programu (např. konstrukcí if RESET then ...), nebo je měnit programem a tím měnit chování systému, nebo si jich v programu vůbec nevšimat. Parametrizaci lze totiž provést kdykoli přímým přístupem do STACKu - nejlépe použitím parametrizační aplikace pro MS EXCEL[®], která tvoří příslušenství k této knihovně.

Modifikaci dat v zásobníku je možno samozřejmě provést přímo ve vývojovém prostředí StudioWin, ve sledovači se zobrazeným zásobníkem. Tuto variantu však ponechme jako úplně krajní, vyžaduje totiž přesný přehled o umístění dat v zásobníku a nese s sebou větší riziko chyby.

Pozn.: Hranice mezi parametry a vstupy/výstupy knihovních funkcí není úplně jednoznačná a za určitých podmínek může být vstup dat de-facto parametrem. Některé vstupy knihovní funkce nemusí být realizovány, protože vlastní zařízení je jednodušší (např. motor ventilátoru není vybaven termokontaktem). V tuto chvíli se nezapojený vstup pro termokontakt stane vlastně parametrem (jeho nastavení do 1 simuluje trvale spojený termokontakt a funkce pak pracuje tak, jako by měla tuto ochranu vyřazenou).

Vstupy/výstupy

Vstupy a výstupy dat do a z knihovních funkcí jsou buď analogové nebo digitální. Analogové vstupy a výstupy jsou ve struktuře dat reprezentovány proměnnou typu word umístěnou na konkrétním místě v zásobníku (umístění od AdrPar je dáno popisem parametrů konkrétní funkce). Digitální vstupy a výstupy jsou obvykle sdruženy do jedné proměnné (bývá to většinou hned první proměnná pracovního bloku určená adresou AdrPar a často také obsahuje digitální parametry systému).

Před voláním určité knihovní funkce je tedy nutné naplnit její vstupy a posléze, když procedura proběhne, se použijí k dalšímu zpracování její výstupy. Do vstupů funkcí nejčastěji kopírujeme hodnoty fyzických vstupů automatu (dle povahy buď analogové nebo digitální), nebo výstupy jiných knihovních funkcí (tak se funkce mohou "propojovat" mezi sebou), nebo obecně jakékoli jiné smysluplné hodnoty. Výstupy funkcí se analogicky připojují buď na výstupy automatu, nebo se použijí jako data pro další zpracování, nebo se některé z nich v tom kterém případě nemusí použít vůbec.

3.2. Připojování vstupů a výstupů, parametry

Všechny vstupy a výstupy mají přiřazena svá symbolická jména - identifikátory. Pod těmito jmény se s nimi pracuje pomocí tzv. spojovacích funkcí - viz dále v seznamu funkcí. Těžištěm práce programátora při používání této knihovny je vyjasnit si počet a druh použitých knihovních funkcí a uspořádat zásobník. Pro každé použití knihovní funkce je třeba v něm vyhradit místo pro pracovní blok dat (**AdrPar**) a pro poruchové slovo (**AdrPor**). Poruchová slova od všech použitých funkcí na sebe musí navazovat. Nakonec je třeba ještě vyhradit prostor pro informace o konkrétních poruchách - viz dále.

3.3. Generování a zpracování poruch

Poruchová slova

Každá regulační funkce (až na několik výjimek) obsluhuje své poruchové slovo o délce **2x WORD** (tedy 4B) na zásobníku (na adrese dané parametrem **AdrPor**). Regulační funkce aktualizuje spodních 12 bitů (bity 0.-11.) prvního wordu na adrese AdrPor. Těchto 12 bitů umožňuje hlášení 12-ti různých poruch. Je-li třeba hlásit více poruch, je třeba vyhradit více poruchových slov (u popisů funkcí je to vždy uvedeno). Poruchová slova od každého volání řídicí funkce jsou v zásobníku uložena bezprostředně za sebou. Horní 4 bity a další word jsou dány parametrizací v aplikaci pro MS Excel.

UPOZORNĚNÍ: V inicializační části programu je nutné nastavit tyto globální proměnné, které využívá funkce pro centrální správu poruch:

MaR_ZacatekPor = začátek bloku poruchových slov, tedy nejmenší použitá AdrPor

MaR_Poc4B = celkový počet použitých poruchových slov

Parametry poruch

K poruchovým slovům je třeba vytvořit na zásobníku místo pro uložení parametrů a stavů jednotlivých poruch. Každá porucha (tedy každý z 12 bitů každého poruchového slova) potřebuje na tyto informace **2x WORD** na zásobníku. To je tedy **24x WORD** (tedy 48B) na každé poruchové slovo. Celý blok všech těchto parametrů je uložen v zásobníku od adresy dané globální předdefinovanou proměnnou **MaR_AdrVyskyt**.

UPOZORNĚNÍ: Tuto proměnnou je nutné v inicializační části programu nastavit (opět nutné pro funkci centrální správy poruch)!

Samotný systém zpracování poruch zabere ještě **2x WORD** (tedy 4B) od adresy **MaR_AdrVyskyt**. Pro parametry poruch musíme tedy stanovit nějakou adresu ve STACKu, tu nastavit do proměnné **MaR_AdrVyskyt** a počítat od této adresy s volným místem na zásobníku v celkové délce: **(2 + (celkový_počet_poruchových_slov x 24)) x WORD**

Příklad

Vypadá to komplikovaně a tak si to ukážeme na následujícím příkladu. Použijeme fiktivní knihovní funkce **Fa** a **Fb**. Funkci **Fa** použijeme dvakrát (např. pro řízení dvou různých motorů). Funkce **Fa** bude potřebovat 10 parametrů, **Fb** 15 parametrů. Komentáře jsou uváděny vždy za středníkem, dle syntaxe jazyka SIMPLE. Nejprve si uspořádáme zásobník. Parametry pro první **Fa** budou začínat na adrese 0, parametry druhé **Fa** budou začínat na adrese 10 a parametry **Fb** budou začínat na adrese 20. Pak určíme místo pro poruchová slova. Použité funkce **Fa** (první), **Fa** (druhá) a **Fb** budou mít **AdrPor** na adresách 35, 37 a 39 (každé poruchové slovo zabírá 2 wordy). Nakonec zbývá jen určit **MaR_AdrVyskyt=41** (zabere 74 wordů: 2 systémové + 3x24 poruchových). Celkem jsme spotřebovali prvních 115 položek zásobníku (STACK 0...STACK 114), volný STACK pro libovolné další použití je tedy od adresy 115. A teď již můžeme psát program...

```

; Používání funkcí MaR.LIB - rozvržení adres, napojení vstupů/výstupů
; Pozn.: v projektu musí být připojeny knihovny MaR.lib a Menu2.lib

; Umístění objektů Fa1,Fa2,Fb a jejich poruchových slov v zásobníku
Const AdrFa1=0,AdrFa2=10,AdrFb=20,PorFa1=35,PorFa2=37, PorFb=39

; Zde začíná hlavní smyčka programu
If RESET then begin
  MaR_AdrVyskyt=41      ;inicializace globálně definovaných proměnných
  MaR_ZacatekPor=35
  MaR_Poc4B=3
end

; ----- Objekt s funkcí Fa (první)
MaRZapisParBit (AdrFa1,MarFaVstup1,X1)      ;připojení dig. vstupů X1,X2
MaRZapisParBit (AdrFa1,MarFaVstup2,X2)
MaRZapisPar (AdrFa1,MarFaVstup3,I8)          ;připojení analogového vstupu I8
MaRfa (PorFa1,AdrFa1)                       ;vlastní volání funkce Fa
O9= MaRPrectiPar (AdrFa1,MarFaVystup1)      ;připojení analog. výstupu O9
Y0= MaRPrectiParBit (AdrFa1,MarFaVystup2)   ;připojení dig.výstupů Y0,Y1
Y1= MaRPrectiParBit (AdrFa1,MarFaVystup3)

; ----- Objekt s funkcí Fa (druhý)
MaRZapisParBit (AdrFa2,MarFaVstup1,X3)      ;připojení dig. vstupů X3,X4
MaRZapisParBit (AdrFa2,MarFaVstup2,X4)
MaRZapisPar (AdrFa2,MarFaVstup3,I9)          ;připojení analogového vstupu I9
MaRfa (PorFa2,AdrFa2)                       ;vlastní volání funkce Fa
O10= MaRPrectiPar (AdrFa2,MarFaVystup1)     ;připojení analog. výstupu O10
Y2= MaRPrectiParBit (AdrFa2,MarFaVystup2)   ;připojení dig.výstupů Y2,Y3
Y3= MaRPrectiParBit (AdrFa2,MarFaVystup3)

; ----- Objekt s funkcí Fb (druhý)
MaRZapisParBit (AdrFb,MarFbVstup1,X5)      ;připojení dig. vstupů X5,X6
MaRZapisParBit (AdrFb,MarFbVstup2,X6)
MaRZapisPar (AdrFb,MarFbVstup3,I10)         ;připojení analogového vstupu I10
MaRfb (PorFb,AdrFb)                        ;vlastní volání funkce Fb
Y4= MaRPrectiParBit (AdrFb,MarFbVystup2)   ;připojení dig.výstupů Y4,Y5
Y5= MaRPrectiParBit (AdrFb,MarFbVystup3)
O11= MaRPrectiPar (AdrFb,MarFbVystup1)     ;připojení analog. výstupu O11

; ----- Centrální zpracování poruch
MaRPoruchovka      ;knihovná funkce, která zpracuje všechny poruchy
                  ;(tedy od obou Fa i od Fb). Je bez parametrů.

RESET=0           ;RESET nulovat až na konci smyčky - nulovat se ale musí!
End

```

Uvědomíme-li si, že jsme právě "naprogramovali" tři samostatné regulátory včetně havarijních funkcí, je uvedený příklad velmi krátký a jednoduchý. Samozřejmě zbývá ještě vše naparametrizovat, ale k tomu později.

4. Seznam knihovních funkcí

Následuje 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 se distribuují i ukázkové soubory a funkční příklady, které lze přeložit a spustit na reálném automatu nebo simulátoru v prostředí StudioWin.

Popis pracovních bloků dat

Většina funkcí pracuje, jak bylo uvedeno, s pracovním blokem dat umístěným na zásobníku. Struktura tohoto bloku je u každé takové funkce popsána tímto způsobem:

Struktura pracovního bloku dat na adrese **AdrPar** :

celková délka pracovního bloku (x WORD) :		8	
identifikátor	význam	typ	adresa
vstupy			
MaRxxAI	analogový vstup funkce	W	1
MaRxxDI	digitální vstup funkce	bit	0?1
výstupy			
MaRxxAO	analogový výstup z funkce	W	2
MaRxxDO	digitální výstup z funkce	bit	0?2
parametry			
MaRxxAP	analogový parametr funkce	W	3
MaRxxAP	analogový parametr funkce	B	4 (H)
MaRxxDP	digitální parametr funkce	bit	0?3
	vnitřní stavy	W	5...7

Nejdůležitější informací je vždy symbolický identifikátor vstupu, výstupu nebo parametru (tučně ve sloupci "**identifikátor**") - pod tímto názvem se s danou položkou pracuje pomocí spojovacích funkcí. S parametry se v programu pracuje poměrně zřídkla, protože pro parametrizaci hotového programu je připravena aplikace pro MS Excel (popis na konci příručky). Sloupec "**význam**" vysvětluje funkci dané položky. Poměrně důležitý je sloupec "**typ**" - udávající, zda jde o položku digitální (typu BIT), nebo analogovou (typu WORD, příp. BYTE). Sloupec "**adresa**" jen pro úplnost udává relativní umístění položky vzhledem k počáteční adrese **AdrPar** bloku dat na zásobníku.

Umístění bitů je značeno konvencí SIMPLE4, tj. W?N - kde W je rel. adresa příslušného wordu v němž je umístěn dotyčný bit a N je pořadí bitu ve wordu. Typy byte (používané jen v parametrech) jsou umísťovány po 2 do jedné pozice typu word a značí se (H) - horní byte, nebo (L) - dolní byte.

Vstupy, výstupy, parametry a analogové typy (word/byte) jsou pro přehlednost barevně odlišeny.

4.1. Globální proměnné definované v knihovně MaR.LIB

Proměnné "MaR_sec" a "MaR_blik"

V knihovně je globálně definovaná proměnná typu bit: **MaR_sec**. Systém knihovny zajišťuje, že bit **MaR_sec** je každou sekundu po dobu právě jednoho průchodu hlavní programové smyčky aktivní. Lze toho využít pro konstrukci neomezeného počtu různých časovačů pracujících s krokem 1 sekundy, ale i k zajištění toho, aby se např. v daném čase nějaká část programu vykonala právě jednou.

Další užitečnou proměnnou typu bit je **MaR_blik**. S periodou asi 1,3s mění svůj stav z 0 do 1 a zpět. Jak název napovídá, lze jej použít např. pro "blikání" různých výstupů.

Předdefinované texty

V knihovně jsou definována tři pole textů pro popis bitových stavů při zobrazování a editaci proměnných typu bit.

```
Table string [2] MaRVypZap=("VYP", "ZAP")
```

```
Table string [2] MaRNeAno=("NE ", "ANO")
```

```
Table string [2] MaRPorOK=("POR", "OK ")
```

4.2. Spojovací funkce

Tvoří jakési "pojivo" mezi jednotlivými výkonnými regulačními funkcemi.

Ke všem vstupům, výstupům i parametrům použité knihovní funkce se můžeme samozřejmě dostat díky znalosti absolutních adres. To je však velmi náročné a nepřehledné. Proto mají parametry, vstupy i výstupy všech funkcí zavedeny své speciální jedinečné názvy - tzv. **identifikátory** (což jsou ve skutečnosti konstanty, definované uvnitř v knihovně). V konstantách jsou kromě adresy umístění zakódovány i další vlastnosti:

- a) typ parametru (word, byte nebo bit)
- b) umístění parametru ve struktuře **Pracovního bloku dat** pro danou funkci
- c) další vlastnosti (např. standardní mód tisku aj.)

Spojovací funkce, sloužící k napojování vstupů a výstupů knihovních funkcí na fyzické vstupy/výstupy automatu nebo na vnitřní proměnné v programu, pracují přímo s těmito identifikátory. Programátor se tedy nemusí zabývat dohledáváním umístění požadovaných dat, parametrů, vstupů a výstupů ve struktuře každého pracovního bloku dat, použije jednoduše jen předdefinovaný symbolický název.

Spojovací funkce:

function word MaRPrectiPar (AdrPar, IDPar)	<i>vytažení parametru</i>
function bit MaRPrectiParBit (AdrPar, IDPar)	<i>vytažení parametru typu bit</i>
subroutine MaRZapisPar (AdrPar, IDPar, Hodnota)	<i>zápis parametru</i>
subroutine MaRZapisParBit (AdrPar, IDPar, Stav)	<i>zápis parametru typu bit</i>
subroutine MaRTiskniPar (AdrPar, IDPar)	<i>zobrazení parametru na displej</i>
subroutine MaREditPar (AdrPar, IDPar, Min, Max)	<i>editace parametru na displeji</i>
subroutine MaRZpracujTep (AdrPar, IDPar, Hodnota)	<i>konverze parametru typu "teplota"</i>

kde:

AdrPar (word) je adresa umístění prac.bloku dat funkce v zásobníku (STACK)
IDPar (word) je identifikátor zvoleného parametru
Hodnota (word) je zapisovaná hodnota typu word
Stav (bit) je zapisovaný stav digitální proměnné typu bit

Pozn.: Abychom předešli případným kolizím datových typů je pro čtení nebo zapisování hodnoty přímo v názvech funkcí odlišena operace s typem "bit".

MaRZapisPar , MaRZapisParBit

Zápis hodnoty do položky v pracovním bloku dat

```
subroutine MaRZapisPar(word AdrPar : word IDPar : word Hodnota)
subroutine MaRZapisParBit(word AdrPar : word IDPar : bit Stav)
```

Předávané parametry

1.parametr (word): adresa (umístění) pracovního bloku dat v zásobníku
2.parametr (word): symbolický identifikátor parametru
3.parametr (word, bit): nová hodnota parametru

Použití

Umožňuje snadný zápis hodnoty do požadovaného parametru, vstupu či výstupu funkce v pracovním bloku dat. Stačí znát jen umístění pracovního bloku dat v zásobníku (většinou zadefinováno v úvodu programu pomocí symbolicky pojmenovaných konstant) a symbolický identifikátor požadovaného parametru (jsou uvedeny vždy u popisu jednotlivých regulačních funkcí). Používá se pro napojení vstupů funkcí na fyzické vstupy automatu, nebo pro přenos hodnot výstupů jedné funkce do vstupů funkcí dalších (viz úvodní příklad pro ilustraci).

Příklad použití

Vezměme např. knihovní regulační funkci MaRUT, popsanou dále. Nyní se podíváme na jeden její parametr. Takto je popsán v tabulce parametrů:

identifikátor	význam	adresa umístění
MaRUTepZad	žádaná teplota topné vody	AdrPar + 2

Popis znamená, že žádaná teplota topné vody je umístěna na adrese `AdrPar+2`, (kde `AdrPar` je adresa umístění začátku pracovního bloku dat funkce ve stacku) a její identifikátor má jméno **MaRUTepZad**.

Dejme tomu, že okruh UT, který regulujeme funkcí `MaRUT`, nazveme `UT1` a v zásobníku bude mít parametry (pracovní blok dat) umístěné od adresy 100. Pak pro přehlednost můžeme na začátku programu definovat konstantu:

Const **UT1** = 100

Dále předpokládejme, že hodnotu žádané teploty topné vody dostává automat po síťové proměnné `D32` a v tabulce globálních síťových proměnných (v projektu ve StudioWin) jsme `D32` zadefinovali symbolickým názvem **UT1TepZad**. Nyní už můžeme elegantně zapsat tuto síťovou proměnnou do struktury pracovního bloku dat `UT1`:

MaRZapisPar(UT1, MaRUTepZad, UT1TepZad)

MaRPrectiPar , MaRPrectiParBit

Vyčtení položky z pracovního bloku dat

```
function word MaRPrectiPar(word AdrPar : word IDPar)
function bit MaRPrectiParBit(word AdrPar : word IDPar)
```

Předávané parametry

- 1.parametr (word): adresa (umístění) pracovního bloku dat v zásobníku
- 2.parametr (word): symbolický identifikátor parametru

Výstupní hodnota

výstup (word, bit): hodnota vyčteného parametru

Použití

Umožňuje snadné vytažení požadovaného parametru, vstupu či výstupu funkce z pracovního bloku dat. Stačí znát jen umístění pracovního bloku dat v zásobníku (většinou zadefinováno v úvodu programu pomocí symbolicky pojmenovaných konstant) a symbolický identifikátor požadovaného parametru (jsou uvedeny vždy u popisu jednotlivých regulačních funkcí).

MaRTiskniPar

Zobrazení položky z pracovního bloku dat na displeji

```
subroutine MaRTiskniPar(word AdrPar : word IDPar)
```

Předávané parametry

- 1.parametr (word): adresa (umístění) pracovního bloku dat v zásobníku
- 2.parametr (word): symbolický identifikátor parametru

Použití

Umožňuje jednoduše tisknout hodnotu požadovaného parametru na displeji automatu, vstupu či výstupu funkce z pracovního bloku dat.

Příklad použití

Opět použijeme pro ilustraci parametr obsahující žádanou teplotu a vytiskneme jej na displej automatu:

```
; TISK PARAMETRU TYPU TEPLOTA  
; předpokládáme vřazení zobrazované položky do centrálního systému menu  
; (programování ovládacího menu viz dokumentace knihovny MenuLIB)  
...  
if MeLine("Tep.Zad.UT1: ") then MaRTiskniPar(UT1, MaRUTTepZad)  
...
```

Díky předdefinovanému identifikátoru **MaRUTTepZad** tisková procedura **MaRTiskniPar** sama zvolí formát zobrazené hodnoty a vytiskne i jednotku. Např. při hodnotě proměnné D32=3332 bude na displeji vytištěno:

Tep.Zad.UT1: +60.0C

Teploty jsou ukládány v desetinách Kelvina, stejně jako je tomu u analogových vstupů pro měření teplot. Číslo 3332 tedy znamená 333.2K, což je +60.0°C.

Pozn.:

Funkce MeLine je z knihovny MenuLIB. Ve všech uváděných příkladech se vždy předpokládá použití knihovních funkcí MenuLIB pro tvorbu ovládacího uživatelského rozhraní ve stylu nabídkového menu. Proto jsou veškeré zobrazovací i editační funkce v příkladech uváděny vždy v kontextu s příslušnými funkcemi MenuLIB. Některé komplexnější funkce knihovny MaR již přímo v sobě obsahují hotovou část nabídkového menu.

MaREditPar

Editace položky z pracovního bloku dat na displeji

```
subroutine MaREditPar(word AdrPar : word IDPar : word Min : word Max)
```

Předávané parametry

- 1.parametr (word): adresa (umístění) pracovního bloku dat v zásobníku
- 2.parametr (word): symbolický identifikátor parametru
- 3.parametr (word): dolní limit pro editaci hodnoty
- 4.parametr (word): horní limit pro editaci hodnoty

Použití

Chceme-li žádanou teplotu editovat, bude příslušná část menu vypadat takto:

```
; EDITACE PARAMETRU TYPU TEPLOTA  
; předpokládáme vřazení zobrazované položky do centrálního systému menu  
; (programování ovládacího menu viz dokumentace knihovny MenuLIB)  
...  
if MeLine("Tep.Zad.UT1: ") then MaREditPar(UT1, MaRUTTepZad, 3032, 3432)  
...
```

Dvě konstanty udávají minimální a maximální editovatelnou hodnotu - v případě teploty je rozměr čísla chápán v desetinách Kelvina (čísla 3032 a 3432 tedy znamenají +30...+70°C). Procedura sama podle předaného identifikátoru rozpozná typ editovaného parametru a zvolí správně formát i jednotky.

V knihovně existuje tato procedura ve dvou modifikacích (pod stejným názvem) - pro word a pro bit. Je-li parametrem k editaci bit, můžeme tuto funkci volat bez parametrů min a max.

MaRZpracujTep

Funkce pro vstupní zpracování teplotních veličin

```
subroutine MaRZpracujTep(word AdrPar : word IDPar : word Hodnota)
```

Předávané parametry

- 1.parametr (word): adresa (umístění) pracovního bloku dat v zásobníku
- 2.parametr (word): symbolický identifikátor parametru, kam se запиše výsledek
- 3.parametr (word): hodnota měřené teploty (např. přímo analogový vstup apod.)

Použití

V oboru MaR se neustále zabýváme teplotami čehokoli a proto je vhodné teploty (resp. analogové vstupy pro Pt-čidla nebo analogové teploměry) zpracovávat jednotným způsobem. Jedná se zejména o vyhlazení jejich průběhu (hodnoty přímo měřené analogovými vstupy neustále driftují o min. $\pm 0.1^{\circ}\text{C}$ vlivem kolísání měřené veličiny, nepřesností digitálního převodu, rušení apod.) a kontroly smysluplnosti předávané hodnoty (vhodné pro kontrolu připojení/nepřipojení, poruchy nebo absence čidla).

Procedura **MaRZpracujTep** časový průběh teploty volitelným způsobem zatlumí, vyhodnotí i smysluplnost předávané hodnoty a výsledek uloží na danou adresu do zásobníku. Neodpovídá-li měřená teplota rozsahu běžných teplot, ukládá procedura do zásobníku poslední smysluplnou hodnotu a stavem 1 v nejvyšším bitu zapisovaného wordu signalizuje poruchu čidla. Všechny knihovní funkce s tímto formátem teploty pracují a na základě tohoto nejvyššího bitu vyhlásují poruchu příslušného čidla a přitom jsou schopny dál pracovat s poslední smysluplnou hodnotou. Samozřejmě i jakoukoli poruchu čidla lze naparametrovat jako havárii a třeba od ní okruh odstavit.

Je třeba ještě připomenout, že analogové vstupy pro Pt-čidla signalizují hodnotami 65532...65535 jednak poruchu, jednak stav bezprostředně po zapnutí automatu po dobu ustálení měřicích vstupů (typicky asi 5-10s).

Smysluplná hodnota zpracovávané teploty je chápána jako číslo v intervalu 1000..30999 (tedy 100,0...3099,9K, což odpovídá rozsahu -173,2...+2226,7 °C). Není-li teplota v tomto rozmezí, je generován již zmíněný "poruchový bit".

Typický příklad volání (vyhodnocení Pt-čidla připojeného na vstup pro čidla Pt100):

`MaRZpracujTep(AdrPar, IDPar, I5)`

Vlastnosti zatlumení se dají nastavit dvěma globálními parametry typu byte:

MaR_TZpracovani určuje periodu zpracování teploty ve sekundách

(při inicializaci je nastaven na hodnotu 4).

MaR_KTlumeni koeficient zatlumení - čím je větší, tím je i zatlumení větší

(při inicializaci je nastaven na hodnotu 2).

Pokud přednastavené parametry nevyhovují, je možné je na začátku programu změnit. Pokud by se vyskytla potřeba různé vstupy automatu zatlumit různým způsobem, je možné tyto parametry nastavit před každým voláním funkce **MaRZpracujTep**.

Pozn.:

Pro připojení čidel jiného typu než Pt100 a na jiné analogové vstupy (které nedávají přímý údaj o teplotě) je vhodné veličinu typu teplota připravit funkcí MaRAI. Funkce realizuje transformaci hodnoty analogového vstupu a je popsána na konci v podkapitole "Pomocné funkce".

4.3. Funkce pro vyhodnocování poruch

Sem patří jednak funkce, která centrálně zpracovává všechny poruchy a havárie generované v systému, jednak funkce pro zobrazování seznamu poruch na displeji. Protože vyhodnocování a zpracování poruch se týká víceméně všech řídicích a regulačních funkcí této knihovny, jsou funkce pro správu poruch popsány jako první.

MaRPoruchovka

Centrální správce poruch

```
subroutine MaRPoruchovka()
```

Předávané parametry

Této proceduře se nepředávají žádné parametry, pracuje totiž s globálně definovanými proměnnými, které se inicializují na začátku aplikačního programu (viz např. náš úvodní příklad - úsek "if RESET then....").

Popis funkce

Při každém proběhnutí hlavní smyčky programu **MaRPoruchovka** zpracuje jeden poruchový stav. Funguje zhruba takto: zjistí v parametrech jestli má vůbec daný poruchový stav vyhodnocovat a případně jak, dále pak zkontroluje zda se změnil stav příslušné poruchy. V případě, že se stav změnil, uloží do vymezeného prostoru informací o novém stavu, zapíše čas a datum tohoto posledního výskytu a do chronologického seznamu poruch učiní záznam o události. Nakonec, když po patřičném počtu proběhů hlavní smyčky vyhodnotí všechny požadované poruchy, vygeneruje globálně definované poruchové bity:

MaR_HavA, MaR_HavB, MaR_HavC, MaR_Por, MaR_JinyStav, MaR_PorNew

Každému vyhodnocovanému poruchovému stavu lze totiž pomocí parametrů přiřadit význam a podle toho jej bude **MaRPoruchovka** vyhodnocovat. Jedná se o čtyři identifikační bity, které lze u každé poruchy nastavit: **HavA, HavB, HavC** a **Por**.

V případě, že daná zpracovávaná porucha bude mít nastavené parametry **HavB** a **Por** a bude aktivní, pak **MaRPoruchovka** po zpracování všech poruch nastaví právě globální proměnné **MaR_HavB** a **MaR_Por**. V případě, že zpracovávaná aktivní porucha nemá nastavený žádný z uvedených čtyř parametrů, nastaví **MaRPoruchovka** globálně deklarovaný bit **MaR_JinyStav**.

Možnost kvitování poruchy

V případě, že budeme chtít provozovat regulované zařízení s nějakým aktivním stavem typu "Por" a přitom budeme chtít tuto poruchu např. opticky signalizovat, můžeme využít globálně deklarovaného poruchového bitu **MaR_PorNew**. Ten je aktivován při objevení nové poruchy, ale je možné jej shodit krátkodobým nastavením dalšího předdefinovaného bitu **MaR_Kvitovani**. V tuto chvíli se všechny právě aktivní poruchové stavy odstíní, tzn. bit **MaR_PorNew** bude procedurou vynulován a nastaven bude až aktivováním nového poruchového stavu. Výhoda spočívá v tom, že systém umožní opticky

signalizovat poruchu, tu (ačkoliv je stále aktivní) může obsluha kvitovat, signalizace zhasne a je připravena signalizovat výskyt další poruchy. Bit **MaR_Kvitovani** je třeba pro správnou funkci nastavit na čas, který potřebuje **MaRPoruchovka** k vyhodnocení všech poruchových stavů, a po ukončení procesu kvitování je nutné ho v programu opět nulovat!

Struktura záznamu poruch

Každá regulační knihovní funkce generuje poruchové slovo o délce 4 byty resp. 2 wordy. Ve skutečnosti poruchové slovo zabírá tuto délku, ale funkce generuje jen spodních 12 bitů (bity 0. až 11.) prvního wordu, tedy té proměnné kam ukazuje **AdrPor**. Každý z těchto bitů představuje jeden poruchový stav. Jaký má daný bit význam je popsáno v popisu příslušné konkrétní funkce. Je-li daný bit nastaven, pak představuje aktivní poruchu. Horní čtyři bity této proměnné (bity 12. až 15.) jsou interpretovány jako 4-bitové číslo (0..15). Toto číslo udává kolik poruch bude MaRPoruchovka zpracovávat. Některé funkce generují poměrně málo poruch (např. 4) a MaRPoruchovka by zbytečně dlouho zpracovávala ty nepoužité. Tento parametr tedy umožní zpracovávat jen tolik poruchových bitů (počítáno od nejnižšího) kolik je třeba a zbývající přeskočit. Následující proměnná ve stacku (na adrese **AdrPor+1**) určuje způsob, jakým bude porucha interpretována na displeji - to využívá zejména procedura MaRDispPoruchy pro zobrazování stavu a historie poruch na displeji (viz dále).

Parametry poruch

Jsou postupně ukládány od adresy **MaR_AdrVyskyt+2** (proměnné na adresách **MaR_AdrVyskyt** a **MaR_AdrVyskyt+1** jsou vnitřní stavy systému). Každému poruchovému bitu na příslušné pozici (od adresy **AdrPor**) jsou přiřazeny dva wordy, v adekvátním pořadí. Pokud je v prvním z nich uloženo číslo 65535 (neboli 0xFFFF neboli samé jedničky) pak není tato porucha vyhodnocována. Jinak jsou v tomto wordu uloženy kromě systémových informací datum posledního výskytu a parametrizační bity dané poruchy:

?0 je HavC

?1 je HavB

?2 je HavA

?3 je Por

?4 je PozBlok

Význam prvních čtyř bitů jsme probrali, nový je zde **PozBlok** (Požadavek Blokování). Pro některé poruchy se totiž hodí, když jejich vznikem se poruchový stav zablokuje a odblokování pak lze provést aktivací globálního předdefinovaného bitu **MaR_Deblok**. Hodí se to například u čidla plynu, kdy i krátkodobá zvýšená koncentrace plynu umožní kotelnu trvale odstavit a vynutit tak příchod obsluhy. (čidla plynu obvykle tuto funkci umí také, ale tady je např. výhoda centrálního debloku všech zablokovaných stavů z jednoho místa). Umístění jednotlivých parametrizačních bitů nemusí programátor detailně znát, neboť pro parametrizaci lze použít parametrizační aplikaci pro MS Excel, která je popsána v závěru této příručky.

Pro úplnost v druhém wordu náležejícímu k danému poruchovému bitu je uložen čas vzniku a další systémové bity.

Poté, co je systém takto podrobně popsán, se vrátíme ještě jednou k fungování havarijního systému jako celku. Jednotlivé řídicí funkce reagují přímo na některé poruchy (např. při výpadku jističe čerpadla funkce automaticky čerpadlo vypne), ale sama neodstaví regulované zařízení jako celek nikdy (až na výjimky). Odstavení příslušného okruhu lze naparametrovat právě od globálních bitů **MaR_HavA**, **MaR_HavB** a **MaR_HavC**.

Systém pak funguje následujícím způsobem: Regulační funkce např. vyhodnotí od difference tlaku chybu ventilátoru. **MaRPoruchovka** zjistí, že je naparametrována jako např. "HavA" a po vyhodnocení všech poruch nastaví globální bit **MaR_HavA**. Regulační funkce identifikuje nastavený bit **MaR_HavA** a ve svých parametrech nalezne informaci, že od bitu **MaR_HavA** má odstavit zařízení - a odstaví jej. Výhodou je, že všechny tyto funkční vazby jsou uloženy v parametrech, nemusí se tedy vůbec programovat a i při chodu programu je možné tyto vazby měnit přímým zásahem do dat v zásobníku (STACK). Nevýhodou je, že samotný STACK je poměrně nepřehledný a k tomu, aby do něj někdo zasahoval, musí mít naprosto jasnou představu. Doporučujeme proto již při tvorbě programu si vést samostatnou dokumentaci o umístění dat v zásobníku, aby kdykoli v budoucnu byla usnadněna orientace v programu. Je vhodné dokumentovat i nastavené parametry. Pro tento účel je připravena již zmiňovaná aplikace pro MS Excel. Použitím této aplikace lze nejen parametry vytvořit a archivovat, ale umožňuje i archivovat STACK kdykoli v průběhu chodu automatu. Parametrizace viz poslední kapitola.

Síťový provoz

Zajímavá situace z hlediska vyhodnocování poruch nastává při použití více automatů v síti. Je zřejmé, že proceduru **MaRPoruchovka** je možné v každém automatu použít jen jednou. A teď nastává nepřehledné množství variací. Nejčastěji bývá v konfiguraci více PLC jeden hlavní (tj. s displejem a tlačítky - např. MPC303) a další podřízené (např. MPC301). Zde je několik možností řešení poruchových stavů:

- a) Jednotlivé regulační funkce jsou dle potřeby umístěné v jednotlivých PLC a tam také generují poruchová slova. Tato slova jsou pomocí síťových proměnných odesílána do řídicího PLC, který je vyhodnotí, vygeneruje poruchové bity a ty zase pomocí síťových proměnných odešle ostatním automatům. Ty pak těmito síťovými bity definují svoje globální poruchové bity a na jejich základě odstavují jimi kontrolované části zařízení.
- b) Jednotlivé regulační funkce jsou dle potřeby umístěné v jednotlivých PLC a tam také generují poruchová slova. Tam jsou ovšem vyhodnocována lokální procedurou **MaRPoruchovka** a tím pádem se chovají značně autonomně. I tak je možné posílat poruchová slova do řídicího PLC pro lepší evidenci nebo je možné naopak řídicímu PLC zpřístupnit poruchové stavy podřízených automatů. Takovou proceduru však není možné univerzálně předpřipravit a je třeba ji napsat na míru pro konkrétní řešení případ.

- c) Další možností je kombinace obou předchozích modelů. Znamenalo by to, že v každém automatu by **MaRPoruchovka** vyhodnocovala poruchy, ale např. **MaR_HavC** by byla vyčleněna pro "globální poruchu" systému generovanou řídicím PLC a předávanou po síti. Po proběhnutí "lokálních" poruchových procedur **MaRPoruchovka** v jednotlivých PLC by se v každém podřízeném automatu vygenerovaný **MaR_HavC** přepsal síťovou globální poruchou. **MaRPoruchovka** v řídicím PLC pak může vyhodnocovat jen poruchová slova vybraných funkcí.

Samozřejmě jsou i další možnosti a vše je v rukách programátora. Takové využití klade vysoké nároky na správu uspořádání zásobníku, na promyšlené předávání dat a na správnou parametrizaci všech parametrů funkcí i poruch ve všech automatech.

Ještě jednou k popsané variantě a) :

Parametry jednotlivých regulačních funkcí musíme samozřejmě umístit do zásobníku automatu, ve kterém jsou používány. Tyto funkce ukládají na příslušnou **AdrPor** maximálně 12 spodních bitů. Tato slova resp. jen těchto 12 bitů předáme po síti do řídicího PLC, kde je umístíme na správné místo ve STACKu a **MaRPoruchovka** je tam vyhodnotí. To tedy znamená, že parametry poruch funkcí použitých v podřízených automatech budou ukládány v řídicím automatu. V případě b) by se vlastně jedna porucha vyhodnocovala ve dvou automatech a musela by mít také dvoje parametry (nejspíš shodné, ale lze si představit i záměrně odlišné vyhodnocování jedné poruchy ve dvou automatech).

MaRDispPoruchy

Zobrazení databáze poruch

```
subroutine MaRDispPoruchy()
```

Předávané parametry

Nepředávají se žádné parametry, procedura pracuje podobně jako **MaRPoruchovka** s globálně definovanými proměnnými.

Popis funkce

Procedura umožňuje prohlížení databáze poruch, tvořené knihovní procedurou **MaRPoruchovka** (viz předchozí stať). Procedura poskytuje tři druhy výpisu poruch, seřazené v kompletním hotovém menu, které procedura nabízí na displeji automatu.

Zvolíme-li "**VSECHNY PORUCHY**", můžeme šipkami vlevo/vpravo listovat seznamem všech poruch. U každé poruchy se zobrazí čas a datum jejího posledního vzniku (pokud ještě nikdy nevznikla, může se na displeji objevit nesmyslný údaj).

Zvolíme-li "**AKTIVNI PORUCHY**", objeví se ve výpisu pouze v danou chvíli aktivní (nebo zablokované) poruchy. Bliká-li na první pozici prvního řádku hvězdička, znamená to, že tento poruchový stav je zablokován a stiskem klávesy ENT lze tuto poruchu individuálně odblokovat. Pokud tato hvězdička svítí trvale, pak nejde odblokovat, protože je stále

aktivní. Svítící tečka na této pozici signalizuje povolené blokování dané poruchy a zároveň stav, kdy porucha není aktivní ani zablokována.

Zvolíme-li "**SEZNAM PORUCH**" vstoupíme do již zmiňovaného chronologického seznamu poruchových událostí, obsahujícího cca 200 záznamů. Po vstupu do tohoto menu je vždy zobrazován poslední záznam. Není-li v seznamu žádný záznam nebo stisknutím šipek vlevo (resp.vpravo) se posuneme mimo zaznamenané stavy menu automaticky ze zobrazování vyskočí o jednu úroveň zpět. Při zobrazování zaznamenaného poruchového stavu se na první pozici prvního řádku vždy objeví buď "+" nebo "-". "+" signalizuje vznik poruchového stavu a "-" signalizuje vymizení poruchy. Chronologický seznam poruch se neukládá do zásobníku, ale do datové struktury definované přímo v knihovně. To znamená, že při změně a následném novém zatažení SW do automatu si automat toto pole definuje na jiném místě a starý seznam tím pádem po provedené změně nebude dostupný. Tento seznam lze mazat procedurou MaRSmazSeznam. Pro správnou funkci zobrazování (resp. pro automatické vyhodnocení začátku a konce seznamu) je nutné tuto proceduru alespoň jednou po zatažení SW zavolat. Není vhodné to činit pomocí bitu RESET, neboť seznam by byl vždy po výpadku napájení automatu smazán. Mazání lze zařadit např. do servisního menu, nebo jej podmínit nastavením nějakého "servisního" bitu.

Identifikace zdroje poruchy

Funkce MaRDispPoruchy poskytuje do systému užitečnou globální proměnnou **MaR_CPor** (typu word). Znamená číslo poruchy, přesněji řečeno pořadí právě zobrazované poruchy. Např. pro celkem 4 generovaná poruchová slova (4x12 poruch) bude proměnná nabývat hodnot 0...47. Pokud není žádná porucha zobrazována, MaR_CPor=65535 (0xFFFF). Například je-li zobrazována první porucha prvního poruchového slova (na adrese MaRZacatekPor), MaR_CPor=0. Je-li zobrazována první porucha druhého poruchového slova (daná bitem (MaRZacatekPor+2)?0), MaR_CPor=12.

Druhá položka typu word ve stacku (na adrese **AdrPor**+1) určuje způsob, jakým bude porucha interpretována na displeji. Přesněji řečeno identifikuje knihovní funkci, která poruchové slovo vytvořila a tím určí použitou skupinu názvů poruch. Opět to vypadá komplikovaně a tak navážeme na úvodní příklad. Představme si, že první funkce **Fa** obsluhuje přívodní ventilátor, druhá funkce **Fa** obsluhuje odtahový ventilátor. Funkce **Fb** se týká např. kotelny. V tomto pořadí nechť také generují svá poruchová slova.

Příklad

Zdefinujeme si např. tabulku textů "NazvySkup", která obsahuje pojmenování těchto tří zpracovávaných objektů. Využijeme-li výše uvedenou proměnnou MaR_CPor, potom při zobrazení např. havárie ventilátoru můžeme zobrazit i informaci o skupině, které se havárie týká (protože skupina, resp. poruchové slovo, obsahuje vždy 12 poruch, dělíme MaR_CPor číslem 12 a dostáváme tak pořadové číslo skupiny, resp. poruchového slova, které je právě zpracovááno).

```

; zdefinování názvů skupin:
table string[3] NazvySkup=("PRIVOD","ODTAH","KOTELNA")
; začlenění zobrazování poruch do menu:
MeInit(0,4)
if MeNext ("PORUCHY") then begin
  MaRDispPoruchy
  if MaR_CPor<36 then Display(NazvySkup[MaR_CPor/12])
end
MeEnd

```

Zobrazení na displeji pak vypadá takto:

```

HAV. PRIVOD
VENTILATORU
V CASE    ZE DNE
14:50     23. 4.

```

Na displej se tiskne "HAV.", protože tato porucha je parametrem určena jako např. HavA, "PRIVOD" tiskne řádek programu, který vyhodnocuje číslo právě zpracovávaného poruchového slova (MaR_CPor/12). "VENTILATORU" se tiskne z tabulky poruchových textů, kterou obsahuje samotná knihovna s pomocí automaticky uloženého wordu na adrese AdrPor+1 a pořadí příslušného poruchového bitu.

Pozn.:

MaRDispPoruchy správně nastaví POSITION (pro dotisk názvu okruhu). Pokud chceme tisknout i název poruchy (pro knihovní funkce jsou automaticky předdefinované) musíme po tisku názvu okruhu posunout POSITION na další řádek a dotisknout název poruchy. K tomu se výborně hodí konstrukce:

```

switch MaR_CPor
  case 0:  Display("text")
  ....
end

```

4.4. Regulační funkce

Ve všech regulačních funkcích, jež jsou založeny na oblíbené PID rovnici, je použit (a dostatečně se osvědčil) pouze PI regulátor. Všechny knihovní procedury, které ovládají jakékoli servopohony, nabízejí výstupy jak digitální (pro "trojbodová" serva) tak analogové a nikde není (většinou) parametr, kterým by bylo třeba upřesnit typ pohonu. Záleží jen na programátorovi, jaké výstupy z funkce připojí na příslušné výstupy automatu a regulátor si pak automaticky najde svůj pracovní bod. Je nutné si v případě použití trojbodového servopohonu uvědomit, že hodnota na analogovém výstupu funkce (v případě trojbodového pohonu nepoužitá) kopíruje pohyb skutečného trojbodového serva jen velmi přibližně a že od hodnoty tohoto výstupu nelze odvozovat skutečnou polohu serva.

Dále uvedeme parametry, které mají všechny regulátory stejně uspořádané:

použitá zkratka	význam	jednotka
I	integrační parametr	0.01
P	proporcionální parametr	0.01
T	perioda jednotlivých zásahů v sekundách	s
N	necitlivost (je-li celkový zásah menší než tento parametr*10, zásah se neprovádí)	0,1s resp. 0,1%

Parametrizační program již sám automaticky nabízí standardní hodnoty pro dané použití regulátoru v konkrétních funkcích. Použijeme-li servopohon s dobou přeběhu 100s, pak by se měl chovat stejně ve verzi analogové i trojbodové. Jinými slovy: ***analogový výstup 0..10V je adekvátní zásahu trojbodového serva velikosti 100s.***

Na závěr důležité upozornění:

K časování všech funkcí systém knihovny využívá časovač T7. Ten s krokem 10ms čítá stále dokola dopředu a je možné ho využít i k jiným účelům než ho využívá knihovna. V žádném případě se však nesmí do jeho chodu jakkoli zasahovat. Byly by tím zasaženy všechny regulační funkce!!! Některé regulační funkce navíc využívají i reálný čas automatu, zejména proměnnou SECOND. Přenastavení hodin se samozřejmě nijak neuplatní, ale například v hlavní smyčce poněkud nesmyslně volaný příkaz Second=0 by chod regulátorů mohl ovlivnit.

MaRUT

Regulace topného okruhu

```
subroutine MaRUT(word AdrPor : word AdrPar)
```

Předávané parametry

- 1.parametr (word): adresa (umístění) bloku poruch v zásobníku
- 2.parametr (word): adresa (umístění) pracovního bloku dat v zásobníku

Struktura pracovního bloku dat na adrese AdrPar :

celková délka pracovního bloku (x WORD) :			13	
identifikátor		význam	typ	adresa
	vstupy			
	MaRUTTepOkruhu	teplota topné vody okruhu UT	W	1
	MaRUTTepZad	žádaná teplota topné vody	W	2
	MaRUTJistic	k.k.(kontrolní kontakt) jističe čer.	bit	0?4
	MaRUTHavTermostat	havarijní termostat přehřátí okruhu	bit	0?3
	výstupy			
	MaRUTVentil	směšovací ventil	W	3
	MaRUTCerpadlo	čerpadlo	bit	0?2
	MaRUTOtvira	trojbodový ventil otvírání	bit	0?0
	MaRUTZavira	trojbodový ventil zavírání	bit	0?1
	parametry			
	MaRUTI	I - integrační parametr	W	4
	MaRUTP	P - proporcionální parametr	W	5
	MaRUTT	T - perioda zásahů	W	6
	MaRUTN	N - necitlivost	W	7
	MaRUTTlumeni	Zatlumení havárie přehřátí (s)	W	8
	MaRUTVentilHavA	zavřít ventil od MaR_HavA	bit	0?8
	MaRUTVentilHavB	zavřít ventil od MaR_HavB	bit	0?9
	MaRUTVentilHavC	zavřít ventil od MaR_HavC	bit	0?10
	MaRUTCerpadloHavA	vypnout čerpadlo od MaR_HavA	bit	0?11
	MaRUTCerpadloHavB	vypnout čerpadlo od MaR_HavB	bit	0?12
	MaRUTCerpadloHavC	vypnout čerpadlo od MaR_HavC	bit	0?13
	MaRUTProtacet	požadavek na týdenní protočení čerpadla	bit	0?14
		vnitřní stavy	W	9...12

Poruchové slovo na adrese AdrPor :

celkový počet poruchových slov (x 2 x WORD) :		1	
porucha od:		typ	adresa
čerpadla		bit	0?0
čidla teploty		bit	0?1
přehřátí		bit	0?2
havárie přehřátí		bit	0?3

Popis funkce

Obě teploty jsou očekávány v desetínách Kelvina. Žádanou teplotu je možné předávat například z níže popsané knihovní funkce **MaRVypoctiEkv**. V případě, že žádaná teplota okruhu je nulová, funkce zastaví čerpadlo a zavře ventil. Digitální vstup **MaRUTJistic** (jistič čerpadla) musí být sepnutý (podmínka pro chod čerpadla). Na rozpojení digitálního vstupu **MaRUTHavTermostat** (používá se typicky u podlahového vytápění) funkce reaguje okamžitým uzavřením ventilu, spuštěním čerpadla a vyhlášením poruchy přehřátí. Pokud je tento vstup rozpojený déle než určuje parametr **MaRUTtlumeni**, vyhlásí systém havarii přehřátí. Týdenní minutové protočení čerpadla je realizováno vždy v úterý ve tři hodiny a tři minuty.

MaREkviterm, MaRDispEkviterm

Výpočet žádané teploty a editace ekvitermní křivky

```
function word MaREkviterm(word AdrPar)
subroutine MaRDispEkviterm(word AdrPar)
```

Předávané parametry

1.parametr (word): adresa (umístění) pracovního bloku dat v zásobníku

Výstupní hodnota

Funkce **MaREkviterm** svůj hlavní výsledek činnosti (tedy žádanou teplotu topného okruhu) ukládá jednak do pracovního bloku dat, jednak jej poskytuje i svojí výstupní hodnotou (typu word, v desetínách Kelvina), lze ji tedy přiřadit kam je třeba i přímo bez použití spojovacích funkcí. Samozřejmě je možné (a pro přehlednost i vhodné) používat **MaREkviterm** tak jako ostatní regulační funkce, bez využití výstupní hodnoty a brát výstup z pracovního bloku dat.

Popis

Tyto dvě knihovní funkce doplňují regulaci topného okruhu.

Obě funkce obracejí tentýž pracovní blok dat, proto jsou zde společně. **MaRDispEkviterm** je samostatné menu displeje pro zobrazení a editaci ekvitermní křivky. **MaREkviterm** počítá dle této křivky žádanou teplotu topného okruhu, např. pro MaRUT.

Struktura pracovního bloku dat na adrese AdrPar :

celková délka pracovního bloku (x WORD) :		9	
identifikátor	význam	typ	adresa
vstupy			
MaREkvitermTepVenk	venkovní teplota	W	4
MaREkvitermTepZadProstor	kvazi-žádaná prostorová teplota	W	7
MaREkvitermUtlum	požadovaný útlum topné vody	W	5
MaREkvitermZap	požadavek na chod okruhu	bit	2?0
výstupy			
MaREkvitermTepZad	žádaná teplota topné vody okruhu	W	6
parametry			
MaREkvitermMinus15	ekviterm.teplota při Tvenk=-15 °C (ve °C)	B	0 (H)
MaREkvitermMinus5	ekviterm.teplota při Tvenk=-5 °C (ve °C)	B	0 (L)
MaREkvitermPlus5	ekviterm.teplota při Tvenk=+5 °C (ve °C)	B	1 (H)
MaREkvitermPlus15	ekviterm.teplota při Tvenk=+15 °C (ve °C)	B	1 (L)
MaREkvitermVypOd	teplota Tvenk pro vypnutí okruhu (ve °C)	B	2 (H)
MaREkvitermTepMax	omezení žádané teploty okruhu (v 0.1K)	W	3
MaREkvitermEditMax	omezení editovaných teplot (ve °C)	B	8 (L)

Pozn.: Termín "**kvazi-žádaná**" teplota je použit proto, že se jedná o ekvitermní regulaci, tedy bez prostorového čidla. Hodnotou tohoto vstupu tedy pouze měníme vztažný bod a posouváme ekvitermní křivku (viz.dále).

Poruchové slovo

Tyto funkce negenerují žádné poruchy.

Popis funkce

Čtyři první parametry určují ekvitermní křivku. Ekvitermní křivka je tvořena libovolnou nadvakrát zalomenou přímkou. Jednotlivé hodnoty venkovní teploty, pro které lze nastavit požadovaná teplota topné vody, jsou pevně dané (-15°C, -5°C, 5°C, 15°C). Nastavovat je může uživatel v menu MaRDispEkviterm v mezích od 10°C až do hodnoty dané parametrem MaREkvitermEditMax.

Parametrem "MaREkvitermVypOd" je možné nastavit teplotu 10°C...30°C. Dosáhne-li venkovní teplota této nastavené hodnoty, celý okruh se automaticky vypne.

Do parametru "MaREkvitermTepMax" je nutné zadat (např.při parametrizaci) omezení pro výpočet žádané teploty v desetinách Kelvina. Toto omezení se sice neprojeví při editaci hodnot ve funkci MaRDispEkviterm, ale uplatní se až při samotném výpočtu žádané teploty funkcí "MaREkviterm". To se hodí zejména u okruhu podlahového vytápění, kde tímto parametrem určíme strop žádané teploty a ani laické nesprávné nastavení ekvitermní křivky nezpůsobí katastrofu.

Do parametru "MaREkvitermTepVenk" uložíme zpracovanou venkovní teplotu. Jako venkovní teplotu můžeme použít například vážený průměr venkovní teploty čidla na severní a čidla na jižní straně. Můžeme tím optimalizovat vytápění okruhu podle jeho situování v budově.

Do parametru "MaREkvitermTepZadProstor" je nutné uložit kvazi-žádanou prostorovou teplotu. Je to normovaná teplota, vztažená k základní hodnotě 20 °C. Jejím nastavením můžeme posunout ekvitermní křivku. Při nastavení 2932 (20 °C) je posun křivky nulový. Větší hodnota žádané prostorové teploty způsobí posun křivky směrem k vyšším teplotám topné vody a obráceně. Posun je vypočítáván tak, aby při správně nastavené ekvitermní křivce způsobil požadované zvětšení prostorové teploty. Regulační funkce očekává parametr kvazi-žádané prostorové teploty v mezích 2782..3082 (tj. 5 °C..35 °C). Je-li parametr mimo tento rozsah, uplatní se omezení tohoto intervalu.

Do parametru MaREkvitermUtlum zapíšeme požadovaný útlum UT. Tento parametr asi nejčastěji použijeme ve spojení s kalendářem (nebo je možné vytvořit kalendář pro kvazi-žádanou prostorovou teplotu). Parametr útlumu může být ve stupních Celsia (je-li menší než 100) nebo může být předán jako teplota v destínách Kelvina. Útlumem pak pro funkci bude rozdíl od teploty 0C (tj. od hodnoty 2732). Ať kladný či záporný - vždy bude interpretován správně jako útlum. Výhodou je to, že v kalendáři lze použít jakýkoli formát editace hodnoty útlumu a výstup kalendáře nakopírovat bez jakýchkoli úprav do funkce MaRUT jako MaREkvitermUtlum.

Vypnutím MaREkvitermZap je možné okruh odstavit. Při venkovní teplotě menší než 5 °C si funkce MaREkviterm tento bit sama zapíná. Tento bit je možné např. připojit v aplikaci na manuální přepínač léto-zima.

Funkce MaREkviterm pak z dat popisujících ekvitermní křivku, venkovní teploty, požadovaného útlumu a kvazi-žádané prostorové teploty vypočítá žádanou teplotu topného okruhu (typicky pro regulační funkci MaRUT). Funkce hodnotu vrací a zároveň ji uloží do parametru "MaREkvitermTepZad".

MaRDispEkviterm

Je samostatné menu displeje, zobrazuje a umožňuje editaci ekvitermní křivky. Tato funkce používá vytváření menu pomocí knihovny MenuLIB. Doporučený způsob volání je tento:

```
If MeNext("Editace Ekvitermu") then MaRDispEkviterm(AdrPar)
```

Kde parametr **AdrPar** (typu word) je umístění pracovního bloku dat (stejně jako pro MaREkviterm). Samotná funkce v sobě obsahuje ukončovací příkaz menu "MeEnd" a proto se už nemusí za jejím voláním "MeEnd" používat.

Samotný displej vytvořený funkcí MaRDispEkviterm vypadá takto:

```
VYPNOUT OD:      20C
EKV.TEP.PRI -15C:70C
EKV.TEP.PRI  -5C:60C
EKV.TEP.PRI +5C:50C
```

Pokud nechceme využít automatické vypínání okruhu, stačí nastavit parametr **MaREkvitermVypOd** na hodnotu větší než 30°C. Vypínání okruhu se pak již prakticky neuplatní a první řádek výše uvedeného menu automaticky přestane být zobrazován.

MaRKaskada

Určení počtu zapnutých kaskádních stupňů (pro plynové kotle apod.)

```
subroutine MaRKaskada(word AdrPor : word AdrPar)
```

Předávané parametry

- 1.parametr (word): adresa (umístění) bloku poruch v zásobníku
- 2.parametr (word): adresa (umístění) pracovního bloku dat v zásobníku

Struktura pracovního bloku dat na adrese AdrPar :

celková délka pracovního bloku (x WORD) :			13	
identifikátor		význam	typ	adresa
	vstupy			
	MaRKaskadaTepVystup	regulovaná teplota (x 0,1K)	W	0
	MaRKaskadaTepZadana	žádaná teplota (x 0,1K)	W	1
	výstupy			
	MaRKaskadaVystup	počet zapnutých stupňů	W	2
	parametry			
	MaRKaskadaStupnuMax	počet stupňů kaskády celkem	W	3
	MaRKaskadaTBlokovani	interval blokování zásahu (x 1s)	W	4
	MaRKaskadaPJZ	pásmo jediného zásahu (x 0,1 °C)	W	5
	MaRKaskadaHystereze	hystereze (x 0,1 °C)	W	6
	MaRKaskadaTDifTep	interval pro výpočet diferenciálu teploty (x 1s)	W	7
		vnitřní stavy	W	8...12

Poruchové slovo na adrese AdrPor :

celkový počet poruchových slov (x 2 x WORD) :		1	
porucha od:		typ	adresa
čidla teploty		bit	0?2

Popis funkce

Sleduje směr vývoje regulované teploty a počítá dobu od svého posledního zásahu (přidání či ubrání stupně). Přidá další stupeň právě tehdy, když:

- a) regulovaná teplota je menší než rozdíl žádané teploty a hystereze
- b) regulovaná teplota neroste
- c) od minulého zásahu uplynula doba větší než interval blokování zásahu

Nepřidá další stupeň pokud po uplynutí doby blokování od posledního zásahu nepřesáhne regulovaná teplota interval daný rozkmitem teploty v tomto blokovacím intervalu, maximálně však v intervalu (žádaná+hystereze, žádaná-pásma jediného zásahu).

Tato regulační funkce neobsahuje PID-regulátor (neosvědčil se).

Analogicky se chová v opačném směru tj. při ubírání stupňů. Logika této regulace počítá s nepříjemným faktem, že samotné spuštění dalšího kotle trvá obvykle 20s a ve chvíli kdy se zapne kotlové čerpadlo dalšího kotle dochází naopak k prudkému poklesu na společném výstupu kotlů. Účinek sepnutí dalšího stupně se tak na čidle společné teploty primárního kotlového okruhu projeví se zpožděním minimálně 90 sekund. Proto parametrizačním programem doporučený interval blokování je cca 120-150s. Jsou-li parametry dobře nastaveny, funguje v praxi tento regulátor po ustálení teplot v jednotlivých okruzích tak, že v intervalu 8..15 minut cyklicky připíná a odpíná jediný stupeň kaskády. Kolísání teploty v primárním okruhu podle nastavených parametrů a vlastností systému bývá asi $\pm 3-4^{\circ}\text{C}$. V parametrizačním programu jsou vždy automaticky nabízeny doporučené hodnoty.

Tuto funkci lze dobře použít i pro řízení elektrokotlů. Její výhodou je to, že jednotlivé stupně připíná postupně a vždy sleduje účinek, který připojení daného stupně způsobilo. V tomto případě je však nutné nastavit interval blokování na podstatně kratší čas.

Tato funkce generuje jednu jedinou poruchu a ta je proto přiřazena do poruch generovaných knihovní funkcí **MaRHavKotPlyn**. Proto se obě tyto funkce mohou volat se stejnou adresou AdrPor a o jedno poruchové slovo se podělí.

MaRPlynKotle

Ovládání max.6 vícestupňových kotlů s návazností na řízení kaskády

subroutine MaRPlynKotle(word AdrPor : word AdrPar)

Předávané parametry

- 1.parametr (word): adresa (umístění) bloku poruch v zásobníku
- 2.parametr (word): adresa (umístění) pracovního bloku dat v zásobníku

Struktura pracovního bloku dat na adrese AdrPar :

celková délka pracovního bloku (x WORD) :		17 ... 42	
identifikátor	význam	typ	adresa
vstupy			
MaRPlynKotleKaskada	požadovaný počet zapnutých stupňů	W	3
MaRPlynKotleTepKotle1	teplota kotle 1.	W	12
MaRPlynKotleJistic1	jistič čerpadel kotle 1. (NC)	bit	1?8
MaRPlynKotleChod1	chod kotle 1. (NO)	bit	2?0
MaRPlynKotlePor1	porucha kotle 1. (NC)	bit	2?8
MaRPlynKotleTepKotle2	teplota kotle 2.	W	17
MaRPlynKotleJistic2	jistič čerpadel kotle 2. (NC)	bit	1?9
MaRPlynKotleChod2	chod kotle 2. (NO)	bit	2?1
MaRPlynKotlePor2	porucha kotle 2. (NC)	bit	2?9
MaRPlynKotleTepKotle3	teplota kotle 3.	W	22
MaRPlynKotleJistic3	jistič čerpadel kotle 3. (NC)	bit	1?10
MaRPlynKotleChod3	chod kotle 3. (NO)	bit	2?2
MaRPlynKotlePor3	porucha kotle 3. (NC)	bit	2?10
MaRPlynKotleTepKotle4	teplota kotle 4.	W	27
MaRPlynKotleJistic4	jistič čerpadel kotle 4. (NC)	bit	1?11
MaRPlynKotleChod4	chod kotle 4. (NO)	bit	2?3
MaRPlynKotlePor4	porucha kotle 4. (NC)	bit	2?11
MaRPlynKotleTepKotle5	teplota kotle 5.	W	32
MaRPlynKotleJistic5	jistič čerpadel kotle 5. (NC)	bit	1?12
MaRPlynKotleChod5	chod kotle 5. (NO)	bit	2?4
MaRPlynKotlePor5	porucha kotle 5. (NC)	bit	2?12
MaRPlynKotleTepKotle6	teplota kotle 6.	W	37
MaRPlynKotleJistic6	jistič čerpadel kotle 6. (NC)	bit	1?13

identifikátor	význam	typ	adresa
MaRPlynKotleChod6	chod kotle 6. (NO)	bit	2?5
MaRPlynKotlePor6	porucha kotle 6. (NC)	bit	2?13
výstupy			
MaRPlynKotleCerpadlo1	chod čerpadla kotle 1.	bit	1?0
MaRPlynKotleMotohod1	doba chodu kotle 1. (hod)	W	14
MaRPlynKotleStupen1	chod stupně 1.	bit	0?0
MaRPlynKotleStupen2...11	(chod stupně 2.-11.)	bit	
MaRPlynKotleStupen12	chod stupně 12.	bit	0?11
MaRPlynKotleCerpadlo2	chod čerpadla kotle 2.	bit	1?1
MaRPlynKotleMotohod2	doba chodu kotle 2. (hod)	W	19
MaRPlynKotleCerpadlo3	chod čerpadla kotle 3.	bit	1?2
MaRPlynKotleMotohod3	doba chodu kotle 3. (hod)	W	24
MaRPlynKotleCerpadlo4	chod čerpadla kotle 4.	bit	1?3
MaRPlynKotleMotohod4	doba chodu kotle 4. (hod)	W	29
MaRPlynKotleCerpadlo5	chod čerpadla kotle 5.	bit	1?4
MaRPlynKotleMotohod5	doba chodu kotle 5. (hod)	W	34
MaRPlynKotleCerpadlo6	chod čerpadla kotle 6.	bit	1?5
MaRPlynKotleMotohod6	doba chodu kotle 6. (hod)	W	39
parametry			
MaRPlynKotlePKotlu	počet kotlů	B	4 (H)
MaRPlynKotlePStupnu	počet supňů každého kotle	B	4 (L)
MaRPlynKotleDobeh	doběh čerpadel (x 1s)	W	6
MaRPlynKotleTepMax	maximální teplota kotlů (x 0,1K)	W	7
MaRPlynKotleTlumeni	Zatlumení poruch chod kotle (x 1s)	W	8
MaRPlynKotlePoradi	Pořadí 1. spínaného kotle (0..poč.kotlů-1)	W	9
MaRPlynKotleBlokovat	blokovat poruchy kotlů	bit	5?0
MaRPlynKotleHavA	odstavit kotle od MaR_HavA	bit	5?1
MaRPlynKotleHavB	odstavit kotle od MaR_HavB	bit	5?2
MaRPlynKotleHavC	odstavit kotle od MaR_HavC	bit	5?3
MaRPlynKotleProtacet	týdenní protáčení čerpadel	bit	5?4
MaRPlynKotle1Cerpadlo	požadavek na chod min. 1 čerpadla	bit	5?5
MaRPlynKotleStridat	požadavek na denní střídání pořadí kotlů	bit	5?6
	vnitřní stavy	W	různě

* NO = Normally Open (v klidu rozepnuto), NC = Normally Closed (v klidu sepnuto)

Pozn.: Šedé položky pro kotle 2...6 nemusí být použity (lze použít libovolný počet kotlů od 1 do 6) a délka pracovního bloku dat je pak podle toho různá - viz dále v textu.

Poruchové slovo na adrese AdrPor :

celkový počet poruchových slov (x 2 x WORD) :		2 ... 3	
porucha od:		typ	adresa
	teplotního čidla kotle 1	bit	0?0
	teplotního čidla kotle 2	bit	0?1
	teplotního čidla kotle 3	bit	0?2
	teplotního čidla kotle 4	bit	0?3
	teplotního čidla kotle 5	bit	0?4
	teplotního čidla kotle 6	bit	0?5
	kotle 1	bit	2?0
	kotle 1 chod	bit	2?1
	kotle 1 přehřátí	bit	2?2
	čerpadla kotle 1	bit	2?3
	kotle 2	bit	2?4
	kotle 2 chod	bit	2?5
	kotle 2 přehřátí	bit	2?6
	čerpadla kotle 2	bit	2?7
	kotle 3	bit	2?8
	kotle 3 chod	bit	2?9
	kotle 3 přehřátí	bit	2?10
	čerpadla kotle 3	bit	2?11
	kotle 4	bit	4?0
	kotle 4 chod	bit	4?1
	kotle 4 přehřátí	bit	4?2
	čerpadla kotle 4	bit	4?3
	kotle 5	bit	4?4
	kotle 5 chod	bit	4?5
	kotle 5 přehřátí	bit	4?6
	čerpadla kotle 5	bit	4?7
	kotle 6	bit	4?8
	kotle 6 chod	bit	4?9
	kotle 6 přehřátí	bit	4?10
	čerpadla kotle 6	bit	4?11

Popis funkce

Tato regulační funkce je specifická variabilním počtem parametrů, poruch, vstupů a výstupů. Vše závisí na počtu řízených kotlů. Je možno nakonfigurovat maximálně 6 jednostupňových nebo dvoustupňových kotlů nebo 4 třístupňové nebo 3 čtyřstupňové nebo 2 pětistupňové nebo teoreticky jen jeden max. dvanáctistupňový kotel. Je-li počet kotlů nejvýše tři, pak se generují vždy 2 poruchová slova (celkem tedy 4 wordy neboli 8B) jeli kotlů více pak generuje vždy 3 poruchová slova (celkem tedy 6 wordů neboli 12B) Vše závisí na parametru počet kotlů, od tohoto parametru se vše odvozuje.

Za zvláštní zmínku stojí popis jednotlivých poruch. U každého kotle je vyhodnocováno celkem pět poruch. Poruchy čidla teploty a čerpadla nepotřebují komentář. Porucha kotle je odvozena od poruchového signálu z kotle.

Některé kotle mohou vykazovat poměrně nepříjemnou vlastnost spočívající v tom, že kotel signalizuje poruchu pouze v případě, že je požadavek na chod kotle (*v některých projektech jsou takto uměle realizovány i poruchy např. od čerpadel - pak nezbyvá, než upravit přímo zapojení a kontrolní kontakty jističů čerpadel přivést přímo na vstupy automatu*). Při požadavku chod vykáže kotel poruchu, na ni zareaguje regulátor odstavením kotle, porucha zmizí a vše se opakuje až do zničení ovládacích relé. Proto je možné zablokovat výskyt poruch kotlů již v této řídicí funkci (nikoli až v MaRPoruchovka) hned prvním digitálním parametrem této funkce. Odblokování lze v tomto případě provést jen pomocí globálního bitu "MaR_Deblok" a tento bit musí být nastaven do doby, než **MaRPlynKotle** zpracuje příslušný zablokovaný kotel. Každým voláním totiž funkce zpracuje vždy jen jeden stupeň celé kaskády.

Dále je vyhodnocována porucha kotle-chod. Kotle mají totiž často další nepříjemnou vlastnost, že svoji poruchu odvozují od stavu své automatiky a existuje stav, kdy automatika kotle se rozhodne kotel nespouštět, ale přesto nevyhlásí poruchu kotle (například když provozní nebo havarijní termostat kotle kotel odstaví). To, že je něco v nepořádku pak odvodíme pouze z toho, že po požadavku na chod dalšího kotle do dané doby (nastavitelné příslušným parametrem, obvykle cca 30s) se neobjeví signál chod kotle.

Poslední poruchou je přehřátí kotle. Tato porucha je vyhodnocována překročením teploty na čidle teploty kotle nastavenou hodnotu (opět příslušný parametr). Možná je to zbytečné, protože kotel je vždy chráněn svým jak provozním tak havarijním termostatem a kaskáda se reguluje čidlem teploty na společném výstupu všech kotlů. Tím pádem jsou na jednotlivých kotlích i čidla teploty zbytečná a zde jsou zapracována jen proto, že jsou takto většinou projektována.

Ještě k parametru pořadí spínaného kotle. Tímto parametrem můžeme určit kotel zařazený do kaskády v prvním pořadí. Pokud ale příslušným parametrem povolíme denní střídání pořadí kotlů, funkce **MaRPlynKotle** si tento parametr sama cyklicky mění a tím realizuje střídání pořadí kotlů.

K parametru požadavek na chod min. 1 čerpadla: Představme si, že je v topném systému několik ekvitermních okruhů. S tím, jak roste venkovní teplota se jednotlivé

okruhy postupně automaticky vypínají a i zbývající okruhy odebírají čím dál tím méně tepla. Pak se může snadno stát, že i jeden stupeň jediného kotle vytopí primární okruh a po uplynutí doby doběhu kotlového čerpadla by se zastavil primární okruh. V tu chvíli čidlo teploty na společném výstupu kotlů neukazuje reprezentativní hodnotu a kaskádní regulace přestane fungovat. Proto je vhodné, když v programu kontrolujeme chod jednotlivých ekvitermních okruhů a tento bit nulujeme až tehdy, když jsou zastaveny všechny sekundární okruhy. Pak se zastaví celá kotelna.

K naplnění požadavku na protáčení čerpadel dochází vždy v úterý ve tři hodiny a tři minuty, pořadí kotlů se případně mění vždy ve dvě hodiny v noci.

Přiřazení stupňů k jednotlivým kotlům je logicky za sebou. Máme-li např. dva dvoustupňové kotle, pak 1.stupni 1.kotle patří 0. bit wordu na adrese AdrPar (?0), 2.stupni 1.kotle patří (?1), 1.stupni 2.kotle patří (?2) a 2.stupni 2. kotle náleží bit (?3).

MaRHavStavy

Havarijní stavy plynové kotelny, automatické dopouštění, ovládání HUP

```
subroutine MaRHavStavy(word AdrPor : word AdrPar)
```

Předávané parametry

1.parametr (word): adresa (umístění) bloku poruch v zásobníku

2.parametr (word): adresa (umístění) pracovního bloku dat v zásobníku

Struktura pracovního bloku dat na adrese AdrPar :

celková délka pracovního bloku (x WORD) :		10	
identifikátor	význam	typ	adresa
	vstupy		
MaRHavStavyTepVenk	venkovní teplota (x 0,1K)	W	1
MaRHavStavyTepVenkJih	venkovní teplota - jih (x 0,1K)	W	2
MaRHavStavyTlak	čidlo tlaku (x)	W	3
MaRHavStavyPlyn1	únik plynu 1.stupeň (NO)	bit	0?0
MaRHavStavyPlyn2	únik plynu 2.stupeň (NO)	bit	0?1
MaRHavStavyZaplaveni	zaplavení (NO)	bit	0?2
MaRHavStavyMaxTepKot	přehřátí prostoru kotelny (NO)	bit	0?3
MaRHavStavyMaxTepPO	přehřátí primárního okruhu (NO)	bit	0?4
MaRHavStavyMinTlak	havarijní minimální tlak (NO)	bit	0?5
MaRHavStavyStop	nouzové odstavení kotelny (NO)	bit	0?6
MaRHavStavyMaxTlak	havarijní maximální tlak (NO)	bit	0?7
MaRHavStavyMPTlak	minimální provozní tlak (NO)	bit	0?8

výstupy				
MaRHavStavyDopousteni	dopouštění systému	bit	0?9	
MaRHavStavyHUP	HUP	bit	0?15	
parametry				
MaRHavStavyTlakZad	žádaný min. tlak (x)	W	4	
MaRHavStavyTDopousteni	interval dopouštění (x 1s)	W	5	
MaRHavStavyTKontroly	interval kontroly dopouštění (x 1min)	W	6	
MaRHavStavyTlakHavMin	havarijní minimální tlak (x)	W	7	
MaRHavStavyTlakHavMax	havarijní maximální tlak (x)	W	8	
MaRHavStavyHavA	odstavit od MaR_HavA	bit	0?14	
MaRHavStavyHavB	odstavit od MaR_HavB	bit	0?13	
MaRHavStavyHavC	odstavit od MaR_HavC	bit	0?12	
MaRHavStavyBlokD	blokovat dopouštění od poruchy dopouštění	bit	0?11	
	vnitřní stavy	W	9	

* NO = Normally Open (v klidu rozepnuto), NC = Normally Closed (v klidu sepnuto)

Poruchové slovo na adrese AdrPor :

celkový počet poruchových slov (x 2 x WORD) :			1
porucha od:			typ adresa
	venkovního čidla		0?0
	venkovního čidla jih		0?1
	čidla anuloidu (rezervováno pro poruchu od "MaRKaskada" !)		0?2
	únik plynu 1.stupeň		0?3
	únik plynu 2.stupeň		0?4
	zaplavení		0?5
	přehřátí prostoru kotelny		0?6
	přehřátí primárního okruhu		0?7
	havarijní minimální tlak		0?8
	nouzové odstavení kotelny		0?9
	havarijní maximální tlak		0?10
	porucha dopouštění		0?11

Popis funkce

Všechny digitální vstupy této funkce mají charakter NO (tj. normaly open resp. aktivní porucha je reprezentována jedničkou). Jednotky u všech veličin s fyzikálním významem tlaku mají uveden rozměr (x). To znamená jednotku kterou zobrazí příslušný analogový vstup automatu - rozsah 0..1000 odpovídá 0...10V a to odpovídá tlaku dle převodní charakteristiky použitého čidla. Proto tedy blíže neurčený rozměr (x). Jednotlivé havarijní vstupy jsou seřazeny do poruchových stavů a vlastní havarijní funkci je třeba realizovat prostřednictvím funkce **MaRPoruchovka** a globálních poruchových bitů.

Funkce automatického dopouštění

Funkce je připravena jak na analogové nebo kontaktní snímání parametrů tlaku. Dopouštění je spouštěno buď poklesem tlaku analogového čidla pod žádaný minimální tlak nebo sepnutím vstupu "min. provozní tlak" (Jen ty vstupy, které programátor "zapojí" budou funkční. Je možné zapojit jak analogový vstup, tak k tomu i např. havarijní tlakový spínač dohromady). V tuto chvíli začne vlastní dopouštění, které vždy trvá po dobu nastavenou parametrem "interval dopouštění". To proto, že v okamžiku otevření dopouštěcího solenoidu vznikají v systému tlakové rázy a měření tlaku v okamžiku dopouštění není reprezentativní. Ihned po skončení dopouštění začíná interval kontroly "nedopouštění". Ten se nastavuje příslušným parametrem v minutách. Kdyby v tomto intervalu došlo k dalšímu požadavku na dopouštění, vyhlásí systém poruchu dopouštění (pravděpodobně uniká voda a systém upozorňuje na příliš časté dopouštění). Od této poruchy je digitálním parametrem možné zablokovat dopouštění. To může zabránit vytopení prostoru s prasklým topením, pokud se však systém napouští poprvé, je časté dopouštění normální. Proto můžeme tento parametr nastavit např. až po týdenním provozu (třeba i automaticky programem automatu).

Ještě k použití dopouštěcího solenoidu: ten se používá naprosto běžně a skrývá v sobě jedno nebezpečí. V případě výpadku dodávky vody tlačí voda v systému topení na solenoid obráceně, ten na to nebývá konstruován a začne propouštět. Je proto nutné do automatické větve dopouštění vždy dávat zpětnou klapku (a na to se většinou zapomíná). Pro větší spolehlivost je také vhodné na dopouštění použít místo solenoidu ventil se servopohonem. Zpětná klapka by stejně neměla chybět, neboť automatika při poklesu tlaku ventil otvírá a v případě výpadku dodávky vody by se voda ze systému naopak vypustila do rozvodu vody. Pozor, většinou dochází v tuto chvíli ke kontaminaci rozvodu pitné vody !

Nakonec ještě nejúspornější varianta automatického dopouštění: pro jeho provoz stačí provozní tlakový spínač. Od něj se řídí dopouštění a vyhlásuje se porucha dopouštění. V případě, že žádné jiné čidlo není k dispozici, je možné pro lepší orientaci obsluhy přejmenovat poruchu dopouštění na havarijní minimální tlak. To uděláme jednoduše tak, že ihned po volání této funkce přesuneme na **AdrPor** poruchový bit poruchy dopouštění na pozici havarijního minimálního tlaku.

HUP je ovládán výhradně kombinací havarijních parametrů a globálních poruchových bitů. Nastavená kombinace blokuje HUP i dopouštění současně.

MaRVentilator

Zajištění spouštění a chodu ventilátoru, kontrola zanesení filtru

subroutine MaRVentilator(word AdrPor : word AdrPar)

Předávané parametry

- 1.parametr (word): adresa (umístění) bloku poruch v zásobníku
- 2.parametr (word): adresa (umístění) pracovního bloku dat v zásobníku

Struktura pracovního bloku dat na adrese AdrPar :

celková délka pracovního bloku (x WORD) :			8	
identifikátor		význam	typ	adresa
	vstupy			
	MaRVentilatorPoz	požadavek chodu ventilátoru (NO)	bit	0?0
	MaRVentilatorDif	tlaková difference na ventilátoru (NO)	bit	0?2
	MaRVentilatorFiltr	diference zanesení filtru (NC)	bit	0?3
	MaRVentilatorJistic	jistič motoru (NC)	bit	0?4
	MaRVentilatorFM	porucha FM motoru (NC)	bit	0?5
	MaRVentilatorTK	termokontakt (NC)	bit	0?6
	výstupy			
	MaRVentilatorChod	chod motoru	bit	0?1
	parametry			
	MaRVentilatorTZap	interval synchronizovaného spouštění	B	1 (H)
	MaRVentilatorOZap	okamžik synchronizovaného spouštění	B	1 (L)
	MaRVentilatorTlumeni	tlumení diferencí (s)	W	2
	MaRVentilatorOdstaveni	odstavení od TK (s)	W	3
	MaRVentilatorHavA	odstavit od MaR_HavA	bit	7?0
	MaRVentilatorHavB	odstavit od MaR_HavB	bit	7?1
	MaRVentilatorHavC	odstavit od MaR_HavC	bit	7?2
	MaRVentilatorRFiltr	nulovat Resetem poruchu filtru	bit	7?3
	MaRVentilatorRMotor	nulovat Resetem poruchu motoru	bit	7?4
	MaRVentilatorRTK	nulovat Resetem odstavení od TK	bit	7?5
	MaRVentilatorARFiltr	automatický reset zanesení filtru	bit	7?6
		vnitřní stavy	W	4...6

* NO = Normally Open (v klidu rozepnuto), NC = Normally Closed (v klidu sepnuto)

Poruchové slovo na adrese AdrPor :

celkový počet poruchových slov (x 2 x WORD) :		1	
porucha od:		typ	adresa
poruchy		bit	0?0
jističe motoru		bit	0?1
chyba FM		bit	0?2
TK (termokontakt)		bit	0?3
Porucha motoru		bit	0?4
Odstavení od TK		bit	0?5
Zanesený filtr		bit	0?6

Popis funkce

Základní kontrolní funkcí je kontrola chodu ventilátoru pomocí čidla tlakové difference zapojeného přes ventilátor. Pokud je ventilátor vypnutý, funkce očekává rozepnutý kontakt, je-li ventilátor zapnutý, očekává kontakt sepnutý. V jakémkoli jiném případě se načítá čas trvání tohoto stavu a při překročení doby tlumení diferencí je vyhlášena porucha motoru a vypnut motor. Pokud sestava není vybavena touto diferencí (motor ventilátoru je chráněn výhradně FM), je třeba přenášet na tento vstupní bit v pracovním bloku dat výstupní bit chod motoru.

Odstavení motoru způsobí automaticky také rozepnutý termokontakt. Prodloužit odstavení ventilátoru od rozepnutí termokontaktu je možné nastavením příslušného parametru. To pomůže zabránit ničivému cyklu: zapnutí, přehřátí, krátké vypnutí a to celé dokola. Nastavíme-li parametr odstavení na hodnotu větší než 60000, je odstavení trvalé. Zrušit tuto odstávku lze nastavením bitu **MaR_Deblok** nebo RESET (je-li nastaven příslušný dig. parametr).

Porucha filtru je vyhlášena překročí-li doba sepnutí difference filtru dobu tlumení diferencí. Odstranit ji lze buď pomocí bitu **MaR_Deblok** nebo RESET (je-li nastaven příslušný dig. parametr) nebo je možné aktivovat bit automatického resetu filtru. Porucha filtru pak sama zmizí pokud při chodu jednotky po dobu dvacetinásobku parametru tlumení diferencí difference ani jednou nesepe. Pro použitý algoritmus nesmí být tlumení diferencí větší než 3000 (obvyklá doporučená hodnota je 30).

Od poruchy FM není přímo odstaven ventilátor (FM jej vypne sám), ale v tomto případě nevznikne porucha ventilátoru (protože fyzicky nemůže běžet).

Nejzajímavější částí je možnost synchronizovaného startu ventilátorů pro eliminaci nadměrného rozběhového proudu při současném spuštění více ventilátorů současně. K tomu slouží parametry MaRVentilatorTZap a MaRVentilatorOZap. Parametr MaRVentilatorTZap určuje periodu spouštění všech motorů v sekundách, MaRVentilatorOZap určuje sekundu sepnutí daného motoru. Příklad: máme celkem šest různých motorů, knihovni funkci MaRVentilator tedy použijeme celkem šestkrát. Spouštět jednotlivé motory budeme po dvou sekundách, perioda tj. MaRVentilatorTZap všech

synchronizačních proměnných bude tedy 12. MaRVentilatorOZap pro jednotlivé motory bude postupně 0, 2, 4, 6, 8, 10. V případě, že MaRVentilatorTZap bude 0, pak se motory spínají okamžitě. Synchronizované spouštění je odvozeno od hodin automatu. Při ovládání motorů z různých automatů je proto třeba ještě zajistit naprosto shodný reálný čas všech automatů, resp. jeho synchronizaci.

MaRVentilator nemá výstup na ovládání příslušné klapky. Předpokládá se ovládání shodné s chodem ventilátoru. Má-li ventilátor zavřenou klapku (ta se otvírá pomalu), je motor odlehčen (tj. spotřebovává menší proud), ale na velikost rozběhového proudu zavřená klapka nemá prakticky vliv.

MaRVZT

Regulace přívodní části VZT v konfiguraci s vodním ohříváčem, volitelně přímý odpar/vodní chlazení, deskový/rotační rekuperátor, směšování, trojbodový nebo analogový servopohon, bezpečnostní funkce, regulace na výtlak nebo prostor, rekuperace tepla i chladu

```
subroutine MaRVZT(word AdrPor : word AdrPar)
```

Předávané parametry

- 1.parametr (word): adresa (umístění) bloku poruch v zásobníku
- 2.parametr (word): adresa (umístění) pracovního bloku dat v zásobníku

Struktura pracovního bloku dat na adrese AdrPar :

celková délka pracovního bloku (x WORD) :			58	
identifikátor		význam	typ	adresa
	vstupy			
	MaRVZTTepNasavana	nasávaná teplota	W	1
	MaRVZTTepZpatecka	teplota zpátečky vodního ohříváče	W	2
	MaRVZTTepVytlak	teplota na výtlaku VZT	W	3
	MaRVZTTepProstor	teplota v prostoru	W	4
	MaRVZTTepZad	žádaná teplota	W	6
	MaRVZTMrazRek	namrzání rekuperátoru (NC)	bit	0?0
	MaRVZTPorChl	porucha chladicí jednotky (NC)	bit	0?1
	MaRVZTChodChl	chod chladicí jednotky (NO)	bit	0?2
	MaRVZTMrazChl	namrzání chladiče (NC)	bit	0?6
	MaRVZTJistic	jistič čerpadla ohříváče (NC)	bit	0?7
	MaRVZTMrazVO	mrazovka vodního ohříváče (NC)	bit	0?8
	MaRVZTPozChod	požadavek chodu VZT (NO)	bit	0?9
	MaRVZTChodChl2	chod chladicí jednotky 2.st. (NO)	bit	10?12

identifikátor	význam	typ	adresa
výstupy			
MaRVZTVentilVO	ventil vodního ohříváče	W	7
MaRVZTVentilChl	ventil vodního chlazení	W	8
MaRVZTRek	rekuperátor	W	9
MaRVZTChlOtvira	servo vodního chlazení otvírá	bit	0?3
MaRVZTChlZavira	servo vodního chlazení zavírá	bit	0?4
MaRVZTCerpadlo	čerpadlo ohříváče	bit	0?5
MaRVZTVOOtvira	servo ohříváče otvírá	bit	0?10
MaRVZTVOZavira	servo ohříváče zavírá	bit	0?11
MaRVZTChl1Chod	chlazení přímý odpar 1.st. chod	bit	0?12
MaRVZTChod	chod VZT (ventilátor)	bit	0?13
MaRVZTRekOtvira	rekuperátor otvírá	bit	0?14
MaRVZTRekZavira	rekuperátor zavírá	bit	0?15
MaRVZTMenicChod	chod měniče rekuperátoru	bit	0?15
MaRVZTTopi	stav VZT - topení (chlazení=0)	bit	10?10
MaRVZTChl2Chod	chlazení přímý odpar chod 2.st.	bit	10?14
parametry			
MaRVZTKPZ	koeficient prostorové závislosti (x 0,1)	W	11
MaRVZTTepMax	max. teplota na výtlaku (x 0,1K)	W	12
MaRVZTTepMin	min. teplota na výtlaku (x 0,1K)	W	13
MaRVZTI	I (integrační parametr) ohříváče	W	14
MaRVZTP	P (proporcionální parametr) ohříváče	W	15
MaRVZTT	T (perioda zásahů) ohříváče	W	16
MaRVZTN	N (necitlivost) ohříváče	W	17
MaRVZTTST	Časová konstanta systému topení (x 1s)	W	18
MaRVZTDobeh	Doběh čerpadla ohříváče (x 1s)	W	19
MaRVZTTOdmrazovani	Interval odmrazování rekuperátoru(x 1s)	W	20
MaRVZTChlI	I (integrační parametr) vodního chlazení	W	21
MaRVZTChlP	P (proporcionální parametr) vodního chlazení	W	22
MaRVZTChlT	T (perioda zásahů) vodního chlazení	W	23
MaRVZTChlN	N (necitlivost) vodního chlazení	W	24
MaRVZTBlok	Blokování chlazení/topení (x 1s)	W	25
MaRVZTDTepZap	Diference pro zapnutí přímého odparu (0,1 °C)	W	26
MaRVZTPasmoZap	Pásmo zapnutí PO (0,1 °C)	W	27

identifikátor	význam	typ	adresa
MaRVZTRekMax	Omezení max. směšování/rekuperace (0..100%)	W	28
MaRVZTTrekMax	Interval pro požadavek na max. rekuperaci	W	5
MaRVZTRekl	I (integrační parametr) rekuperátoru	W	45
MaRVZTRekP	P (proporcionální parametr) rekuperátoru	W	46
MaRVZTRekT	T (perioda zásahů) rekuperátoru	W	47
MaRVZTRekN	N (necitlivost) rekuperátoru	W	48
MaRVZTTServo	Doba přeběhu serva rekuperace (x 1s)	W	50
MaRVZTTlumeni	Tlumení poruchy chlazení chod (x 1s)	W	52
MaRVZTTChI2	Zatlumení 2.stupně PO (x 1s)	W	55
MaRVZTProtacet	požadavek na týdenní protočení čerpadla	bit	10?0
MaRVZTRekuperuj	povolit rekuperaci chladu	bit	10?1
MaRVZTPrefTop	preference topení před směšováním	bit	10?2
MaRVZTPO3min	přímý odpar- zajistit min. 3 min. pauzy	bit	10?3
MaRVZTPO20max	přímý odpar- zajistit max. 20 min. chod	bit	10?4
MaRVZTVChI	vodní chlazení	bit	10?5
MaRVZTHavA	blokovat od MaR_HavA	bit	10?6
MaRVZTHavB	blokovat od MaR_HavB	bit	10?7
MaRVZTHavC	blokovat od MaR_HavC	bit	10?8
MaRVZTChIBlok	blokovat poruchu chlazení	bit	10?9
MaRVZTRRek	rotační rekuperátor	bit	10?11
	vnitřní stavy	W	různé

* NO = Normally Open (v klidu rozepnuto), NC = Normally Closed (v klidu sepnuto)

Poruchové slovo na adrese AdrPor :

celkový počet poruchových slov (x 2 x WORD) :		1	
porucha od:		typ	adresa
čidla venkovní teploty		bit	0?0
čidla teploty na zpátečce ohřívače		bit	0?1
čidla teploty výtlaku VZT		bit	0?2
čidla prostorové teploty		bit	0?3
mrazovka		bit	0?4
prohřívání (chladný výměník)		bit	0?5
čerpadla		bit	0?6
namrzání chladiče		bit	0?7

porucha od:	typ	adresa
porucha chlazení	bit	0?8
namrzání rekuperátoru	bit	0?9
chlazení chod	bit	0?10
chlazení 2.st. chod	bit	0?11

Popis funkce

Čidlo nasávané teploty slouží ke dvěma účelům. Jednak je nezbytné pro regulaci rekuperace (nebo směšování), za druhé je od jeho hodnoty ovlivněno prohřívání vodního výměníku před startem. To konkrétně vypadá takto: je-li údaj tohoto čidla větší než 10°C pak požadovaná min. teplota zpátečky výměníku je jen 10°C (je-li větší než 5°C a menší než 10°C pak min. teplota výměníku je 20°C a je-li do 5°C pak regulační funkce požaduje alespoň 35°C na zpátečce). Teploty na které výměník prohříváme nejsou nijak velké. Praxe ukazuje, že když je výměník před startem prohřátý mnohem více, po zapnutí VZT se potrubím valí extrémně horký vzduch a regulaci je pak třeba zase krotit, aby úplně nezavřela ventil výměníku. Z uvedených informací plyne, že čidlo nasávané teploty není povinné (pokud není v sestavě rekuperátor) a je možné hodnotou tohoto parametru ovlivňovat režim startu jednotky. Je například možné přepínačem Léto/Zima měnit tento parametr v hodnotách 2932/2732 (tj. 20°C/0°C) (všechny teploty jsou v desetínách K). Je-li čidlo přítomné, je důležité jeho správné umístění. Čidlo je třeba umístit do sacího potrubí co nejblíže k venkovnímu prostoru ideálně do sluncem zastíněného místa hned za nasávací mřížkou (ve směru proudění vzduchu před vstupní klapku). Jako čidlo nasávané teploty lze také použít čidlo venkovní teploty, příp. tento údaj (pokud již v systému někde je) pouze převzít.

Čidlo na zpátečce vodního ohříváče je povinné. Má tři funkce: je od jeho hodnoty zajišťována protimrazová ochrana výměníku (ta je samozřejmě hlavně zajištěna kapilárním termostatem), zabezpečuje prohřívání výměníku a pomáhá zajistit stabilitu regulátoru.

Čidlo teploty na výtlaku je povinné, je to hlavní regulační čidlo. Je třeba jej umístit do místa, kde je v potrubí vzduch již promíchán (tedy ne hned za výměník).

Žádanou hodnotu teploty na výtlaku si funkce MaRVZT vypočítává takto:

$$\text{žádaná_výtlak} = \text{žádaná} + \text{koef.KPZ} * (\text{žádaná} - \text{prostorová}) / 10$$

kde:

žádaná_výtlak		vnitřní stav na adrese AdrPar+32
žádaná		MaRVZTTepZad
koef. KPZ		MaRVZTKPZ
prostorová		MaRVZTTepProstor

Je-li $\text{MaRVZTKPZ} = 0$, pak žádaná se stává žádanou na výtlaku tj. regulujeme na výtlak. Při regulaci na prostor se o koeficientem znásobenou odchylku prostorové od žádané teploty upravuje žádaná teplota výtlaku. Příklad: je-li žádaná=2932 (20°C), prostorová=2912 (18°C) a $\text{KPZ}=10$ (1,0 je interpretován v desetinách), pak:

$$\text{žadáná_výtlak} = 2932 + 10 * (2932 - 2912) / 10 = 2952 \text{ tj. } 22^\circ\text{C}.$$

Na žádanou teplotu na výtlaku se pak uplatní hodnoty omezení (MaRVZTTepMax resp. MaRVZTTepMin). To v praxi znamená, že skutečná teplota na výtlaku při extrémních požadavcích nepatrně kolísá právě kolem hodnoty tohoto omezení.

Čidlo teploty v prostoru je samozřejmě nepovinné (při regulaci na výtlak a není-li v sestavě rekuperátor, potom $\text{koef}=0$). Regulace rekuperátoru očekává, že prostorová teplota je táž, jako teplota na odtahu z prostoru. Z toho plyne požadavek na vhodné umístění čidla.

Jako čidlo namrzání rekuperátoru je očekáván snímač tlakové difference.

Podobně jako byla vyhodnocována porucha kotlů, je zde řešena porucha chladicí jednotky. Chladicí jednotka generuje poruchu a tu (pokud je závislá na požadavku chodu chladicí jednotky) je možné příslušným parametrem zablokovat. Podobně jako u kotlů je řešena také porucha chlazení chod. Ta je vyhlášena v okamžiku kdy od požadavku chlazení chod se do doby dané tlumením (MaRVZTTlumeni) neobjeví na příslušném vstupu funkce signál chlazení chod a nebo pokud při nepřerušeném požadavku chlazení chod "odpadne" chlazení chod (tj. automatika chladicí jednotky jednotku odstaví). Zde je situace komplikovaná, neboť různé chladicí jednotky mají různý stupeň zabezpečení svých funkcí. Často mají jednotky svoje vlastní čidlo namrzání výměníku (jehož signál nedávají externě k dispozici), většinou chrání svůj kompresor před opakovaným startem cca tříminutovým zpoždovacím členem atd. Opět nebývá většinou zcela jasné od jakých stavů výrobce zařízení generuje externí poruchu. Typicky namrzání výměníku bývá považováno za provozní stav, nicméně jeho příliš častý výskyt může signalizovat nějakou závadu. Jediný způsob, jak svéhlavé odstavování jednotky můžeme sledovat, poskytuje právě tato porucha: chlazení chod. V parametrech týkajících se právě chlazení (resp. přímého odparu) nalezneme parametry umožňující zajištění tříminutového odpočinku kompresoru a maximálně dvacetiminutového nepřetržitého provozu přímo funkcí MaRVZT . Je proto dobré vědět, jaká chladicí jednotka je použita a které bezpečnostní funkce má v sobě integrované (aby je funkce MaRVZT nezdvojovala) a aby bylo možné správně nastavit parametr tlumení poruchy chlazení chod (dána právě zpožděním rozběhu chladicí jednotky).

Všechny analogové výstupy mají rozsah 0...1000 a fungují tak, že ve stavu 0 očekávají ventil zavřený, ve stavu 1000 otevřený. Pro VZT se směřováním lze použít analogový výstup pro rekuperaci, hodnota 0 zde pak znamená nulové směřování tj. 100% venkovního vzduchu. Při hodnotě 1000 je vstupní klapka plně zavřena, směšovací plně otevřena. Parametrem omezení směšování (MaRVZTRekMax) nastavíme maximální hodnotu tohoto výstupu při chodu jednotky (pokud je jednotka vypnutá, objeví se zde hodnota 1000, aby se zavřela vstupní klapka).

Výstup "MaRVZTRekZavira" (tj.rekuperátor zavírá) pro deskový rekuperátor je pro rotační rekuperátor použit jako povel "MaRVZTMenicChod" (měnič chod). Proto oba identifikátory odkazují na stejný bit datové struktury. Použití rotačního rekuperátoru vyžaduje použití analogového výstupu. I zde je možné využít předcházejícího parametru omezení. Funkci pro rotační rekuperátor je nutné aktivovat příslušným bitem (MaRVZTRek).

Zajímavým bitem je výstupní stav (MaRVZTTopi). Ten signalizuje zda se celá jednotka nachází ve stavu topení nebo chlazení (resp. která z těchto činností je povolena). Jednotka vždy startuje v režimu topení. Režim si jednotka změní pokud po dobu blokování (AdrPar+25) nevykonává právě povolenou činnost a jeli změna regulací požadována. Je-li parametr blokování nulový nebo je-li venkovní teplota do 15°C, jednotka je trvale v režimu topení a nepovolí ani rekuperovat chlad.

Parametry I,P,T,N mají standardní význam ve všech použitých aplikacích. Jen u regulace ohřívače je skutečná regulace trochu složitější a počítá navíc pro zlepšení stability regulace s časovou konstantou systému topení (MaRVZTTST). Tato konstanta je určena zpožděním odezvy teploty na výtlaku na jednotkový skok serva. Tuto hodnotu je možné na VZT jednoduše hodinkami změřit a nemělo by do ní být započteno případné dopravní zpoždění topné vody od zdroje k třícestnému ventilu. Bývá asi 60 až 90 sekund podle velikosti výměníku, umístění čidla atd. Obsahuje-li jednotka za ohřívačem ještě chladicí výměník, je tato doba obvykle o cca. 30 sekund větší. Je vhodné hodnotu tohoto parametru zadávat o něco větší - cca 240s. VZT se pak sice bude pomaleji blížit k žádané teplotě výtlaku, ta ale po ustálení bude dlouhodobě stabilnější.

Chlazení analogové (vodní) je řízeno PI regulátorem s příslušnými standardními parametry P,T,N,I. Pro regulaci chlazení - přímého odparu je použit tento algoritmus: chlazení se zapne právě tehdy, když teplota na výtlaku je větší než žádaná teplota na výtlaku + difference pro zapnutí P.O. (MaRVZTDTEpZap). Pokud (od této teploty) teplota na výtlaku klesne o pásmo zapnutí P.O. (MaRVZTPasmoZap), tak se chlazení vypne. Pro řízení přímého odparu jsou protichůdné požadavky. Jednak z požadavku udržet teplotu na výtlaku co nejblíže žádané teplotě na výtlaku vyplývají hodnoty popsaných dvou parametrů co nejmenší. Pro "rozumné" spínání chladicí jednotky je dobré oba parametry volit zase větší. Doporučené parametry nabízí parametrizační program, ale zejména pásmo zapnutí musí programátor zvolit podle výkonových poměrů dané jednotky.

Druhý stupeň chlazení vychází z těchto parametrů jako první stupeň. Navíc je použit parametr "Kaskádní interval 2.stupně" PO (MaRVZTTChl2). Tímto parametrem je možné nastavit "kaskádní zpoždění 2.stupně" tj. 2. stupeň zapne o tento interval po 1.stupni pokud teplota na výtlaku neklesne pod úroveň požadavku na zapnutí chlazení. Obdobně se MaRVZT chová při vypínání. Kaskáda obou stupňů se chová takto:

- a) 1.stupeň se dle potřeby VZT zapíná a vypíná
- b) pokud 1.stupeň nepostačí, zapne se s kaskádním zpožděním 2.stupeň. Výkon chlazení je nadále regulován spínáním 1.stupně

- c) pokud 1. stupeň je vypnut a teplota na výtlaku je nižší než teplota pro vypnutí chlazení, rutina po uplynutí kaskádního zpoždění vypne i druhý stupeň.

Je důležité si uvědomit, že může být zapnutý 2.stupeň a vypnutý 1.stupeň. To může být někdy výhodné a jindy nikoliv. Pokud jsou jednotky prvního a druhého stupně na sobě zcela nezávislé, je to výhodné, MaRVZT může kontrolovat max. 20min chodu nezávisle obou jednotlivých stupňů. Pokud je jednotka kompaktní, je možné, že vyžaduje při zapnutí 2.stupně i chod 1.stupně. V takovém případě je nutné výstupy zkombinovat (pomocí and a or) tak, aby bylo vyhověno požadavkům výrobce chladicí jednotky. Vhodnou logikou je pak nutné "připojit" i vstupy chlazení-chod pro oba stupně a zajistit případné požadavky na 3min. pauzu jednotky.

Jen u servopohonu rekuperace je žádán parametr - doba přeběhu (MaRVZTTServo) v sekundách. MaRVZT jej využívá při výpočtu snížení rekuperace při namrzání rekuperátoru.

Při prvním rozeptnutí tlakové difference se sníží rekuperace o 50%. Tuto hodnotu zachová po dobu odmrazování rekuperace (MaRVZTTOdmrazovani) a pak vždy po tomtéž intervalu postupně po 10% se vrací k původní hodnotě. Pokud nedochází k namrzání rekuperátoru, je rekuperátor řízen standardním PD regulátorem. Rekuperátor je ale upřednostněn před topením tzn. od otvírání serva topení je po dobu požadavku na max. rekuperaci (MaRVZTTRekMax) rekuperace maximální (podobně to funguje i v režimu chlazení). Při použití směšování lze digitálním parametrem preferovat naopak topení výměníkem před směšováním. Použití tento parametr nalezne např. u VZT jídelny v době maximálního zatížení. Pak je možné hodinami (nebo použitím kalendáře) ovládat tento parametr a na dobu např. dvou hodin upřednostnit topení před rekuperací a zajistit tak pro danou dobu maximální výměnu vzduchu. Směšovat pak začne MaRVZT až v případě, že topení již nestačí k udržení požadované teploty.

Rekuperovat teplo je možné jen pokud prostorová teplota (na odtahu) je větší alespoň o 0,6°C než venkovní teplota. Pro rekuperování chladu platí podmínka obráceně.

Z poruch stojí za zmínku "prohřívání". Tímto bitem MaRVZT hlásí, že na zpátečce výměníku je vzhledem k venkovní teplotě nízká teplota, že se otvírá servo ohříváče a že ač chod VZT je požadován, není povolen. Protože je to spíše provozní stav, pro který není třeba aktivovat žádný bit určující poruchu nebo havárii. I pak bude tento stav v chronologickém seznamu poruch zaznamenán.

MaRTUV

Nabíjení TUV i řízení el.ohřevu, zprac. poruchy cirkulačního čerpadla.

```
subroutine MaRTUV(word AdrPor : word AdrPar)
```

Předávané parametry

- 1.parametr (word): adresa (umístění) bloku poruch v zásobníku
- 2.parametr (word): adresa (umístění) pracovního bloku dat v zásobníku

Struktura pracovního bloku dat na adrese AdrPar :

celková délka pracovního bloku (x WORD) :		7	
identifikátor	význam	typ	adresa
vstupy			
MaRTUVTep	teplota TUV	W	1
MaRTUVTepZad	žádaná teplota TUV	W	2
MaRTUVTepZadEl	žádaná teplota TUV pro el. ohřev	W	6
MaRTUVTepPrimar	teplota primárního okruhu (nabíjecí vody)	W	3
MaRTUVTerm	přehřátí TUV (NC)	bit	0?3
MaRTUVJisticNabC	jistič nabíjecího čerpadla (NC)	bit	0?4
MaRTUVJisticCirkC	jistič cirkulačního čerpadla (NC)	bit	0?11
výstupy			
MaRTUVCerpadlo	nabíjecí čerpadlo	bit	0?0
MaRTUVPoz	požadavek na teplou vodu	bit	0?1
MaRTUVEITop	el. ohřev	bit	0?2
parametry			
MaRTUVTepMax	teplota přehřátí TUV	W	4
MaRTUVHystereze	hystereze regulace	W	5
MaRTUVHavA	odstavit od MaR_HavA	bit	0?5
MaRTUVHavB	odstavit od MaR_HavB	bit	0?6
MaRTUVHavC	odstavit od MaR_HavC	bit	0?7
MaRTUVBlok	blokovat od přehřátí	bit	0?8
MaRTUVProtaceni	týdenní protáčení čerpadla	bit	0?9
MaRTUVBlokEl	blokovat el. ohřev od nabíjení	bit	0?10

* NO = Normally Open (v klidu rozepnuto), NC = Normally Closed (v klidu sepnuto)

Poruchové slovo na adrese AdrPor :

celkový počet poruchových slov (x 2 x WORD) :		1	
porucha od:		typ	adresa
čidla TUV		bit	0?0
nabíjecího čerpadla		bit	0?1
cirkulačního čerpadla		bit	0?2
přehřátí TUV		bit	0?3
čidla primárního okruhu		bit	0?4

Popis funkce

Je-li teplota TUV menší než žádaná teplota TUV, nastaví funkce MaRTUV požadavek na teplou vodu (MaRTUVPoz). Tento požadavek slouží pro určení žádané teploty primárního okruhu. Pokud je v primárním okruhu teplota alespoň o 7°C větší než teplota TUV, spustí se nabíjecí čerpadlo. Jeli teplota v primárním okruhu větší o méně než 5°C než teplota TUV, nabíjení se pro jeho malou efektivitu zastaví. Po nahřátí TUV na žádanou teplotu zvětšenou o hysterezi regulace je vynulován požadavek na teplou vodu a zastaveno nabíjení. Podobně funguje také regulace el. ohřevu. Kombinace obou ohřevů je myšlena takto: Žádaná teplota pro elektrický ohřev je cca o 5°C menší než žádaná teplota TUV (pro nabíjení). TUV je pak vždy napřed ohřívána topnou vodou a jen při jejím nedostatku (např. odstávce) se TUV nabíjí elektricky. Bitem (MaRTUVBlokEl) lze blokovat el. ohřev od chodu nabíjecího čerpadla. To je jediná vazba mezi oběma regulacemi.

Porucha přehřátí TUV je vyhlášována jak od rozepnutí havarijního termostatu (MaRTUVTerm), tak i od překročení mezní teploty (MaRTUVTepMax). Od této poruchy lze volitelně (MaRTUVBlok) přímo ve funkci MaRTUV blokovat nabíjení i el. ohřev.

MaRDrevokotel

Regulace kotle na dřevo

subroutine MaRDrevokotel(word AdrPor : word AdrPar)

Předávané parametry

- 1.parametr (word): adresa (umístění) bloku poruch v zásobníku
- 2.parametr (word): adresa (umístění) pracovního bloku dat v zásobníku

Struktura pracovního bloku dat na adrese AdrPar :

celková délka pracovního bloku (x WORD) :			20	
identifikátor		význam	typ	adresa
	vstupy			
	MaRDrevokotelTepZp	teplota zpátečky dřevokotle	W	2
	MaRDrevokotelTepOut	teplota výstupu DK tj. za reg. ventilem	W	3
	MaRDrevokotelTepOutZad	žádaná výstupní teplota	W	4
	MaRDrevokotelJisticOut	jistič čerpadla výstupu kotle (NC)	bit	0?0
	MaRDrevokotelTermK	přehřátí okruhu kotle (NC)	bit	0?1
	MaRDrevokotelTermOut	přehřátí výstupu kotle (NC)	bit	0?12
	MaRDrevokotelJisticK	jistič čerpadla okruhu kotle (NC)	bit	0?14
	výstupy			
	MaRDrevokotelVentil	ventil (je-li 0, pak kotel topí do zpátečky)	W	1
	MaRDrevokotelCerpadloOut	výstupní čerpadlo	bit	0?2
	MaRDrevokotelVentilator	ventilátor dřevokotle	bit	0?3
	MaRDrevokotelOtvira	ventil otvírá tj. kotel topí do primáru	bit	0?4
	MaRDrevokotelZavira	ventil zavírá tj. kotel topí do zpátečky	bit	0?5
	MaRDrevokotelCerpadloK	čerpadlo okruhu kotle	bit	0?6
	parametry			
	MaRDrevokotelTepZpZad	žádaná teplota zpátečky dřevokotle	W	5
	MaRDrevokotelIDTepMod	diference pro přepínání módu regulace	W	8
	MaRDrevokotelTBlok	blokování ventilátoru při přehřátí kotle	W	9
	MaRDrevokotelTepMax	mezní teplota přehřátí výstupu	W	10
	MaRDrevokotelI	I	W	11
	MaRDrevokotelP	P	W	12
	MaRDrevokotelT	T	W	13
	MaRDrevokotelN	N	W	14

identifikátor	význam	typ	adresa
MaRDrevokotelDobeh	doběh výstupního čerpadla	W	15
MaRDrevokotelHavA	odstavit od MaR_HavA	bit	0?7
MaRDrevokotelHavB	odstavit od MaR_HavB	bit	0?8
MaRDrevokotelHavC	odstavit od MaR_HavC	bit	0?9
MaRDrevokotelVypCer	vypnout při havárii čerpadla	bit	0?10
MaRDrevokotelZavVent	zavřít při havárii ventil	bit	0?11
MaRDrevokotelProtacet	týdenní protáčení čerpadel	bit	0?13
	vnitřní stavy	W	různé

* NO = Normally Open (v klidu rozepnuto), NC = Normally Closed (v klidu sepnuto)

Poruchové slovo na adrese AdrPor :

celkový počet poruchových slov (x 2 x WORD) :		1	
porucha od:		typ	adresa
	čidla zpátečky	bit	?0
	čidla výstupu	bit	?1
	přehřátí kotle	bit	?2
	přehřátí výstupu	bit	?3
	čerpadla výstupu	bit	?4
	čerpadla okruhu kotle	bit	?5

Popis funkce

Kompletní uvažovaná sestava zařízení je tato:

Okruh kotle (malý) :

dřevokotel - výstup - čerpadlo okruhu kotle - čtyřcestný ventil - zpátečka kotle

Primární okruh:

sběrač TV - čtyřcestný ventil - výstupní čerpadlo - rozdělovač TV

Čidla teploty:

na zpátečce kotle a na výstupu za čtyřcestným ventilem

Havarijní termostat:

na výstupu kotle před čtyřcestným ventilem a na výstupu za čtyřcestným ventilem.

Čidla teploty jsou povinná, termostaty nikoli, čerpadla dle doporučení výrobce kotle. Vlastní regulační algoritmus je poměrně složitý. Regulátor pracuje ve dvou módech: v prvním módu se reguluje na teplotu na zpátečce, ve druhém se reguluje na výstup.

V prvním módu se jednak kotel roztápí (prioritou je dosáhnout žádané teploty na zpátečce) a také se v něm dosahuje maximálního výkonu kotle (pokud není schopen dosáhnout žádané teploty na výstupu). V tomto módu je vždy zapnutý ventilátor kotle (pokud nedojde k přetopení kotle). Překročení teploty zpátečky žádanou teplotu zpátečky o více než 10°C způsobí v tomto režimu okamžité otevření ventilu. V druhém módu regulujeme na výstup. V tomto režimu je vypínáním ventilátoru kotle udržována teplota zpátečky na žádané hodnotě zvětšené o diferenci pro ovládání ventilátoru (MaRDrevokotelTepZpZad+ MaRDrevokotelDTepVent). Do prvního módu se regulace přepne právě tehdy, když teplota zpátečky klesne pod žádanou teplotu zpátečky nebo teplota výstupu klesne o více než diferenci pro přepínání módu regulace pod žádanou teplotu výstupu. Do druhého módu se regulace přepne právě tehdy, když na zpátečce bylo dosaženo žádané teploty a teplota na výstupu je větší než žádaná teplota na výstupu zvětšená o diferenci pro přepínání módu regulace. Doporučené parametry obsahuje parametrizační program.

Jediná "pevně zadrátovaná" bezpečnostní funkce spočívá v reakci na přetopení kotle (MaRDrevokotelTermK). Za prvé se otevře ventil a zapnou se čerpadla. Za druhé se vypne ventilátor kotle. Tento ventilátor je dále po vymizení přetopení kotle blokován po dobu danou parametrem (MaRDrevokotelTBlok). Toto blokování může být ukončeno signálem Reset nebo Deblok.

Další bezpečnostní funkce je třeba naparametrizovat nebo řešit jinými prostředky. Je třeba důkladně zvážit, co dělat při přehřátí primárního okruhu. Je nutné zvýšit odběr tepla z okruhu. Nabízí se dobít bojler TUV na maximální možnou mez, dále pak přebytečné teplo odvádět do okruhů UT případně VZT. Odstavení od jednotlivých havárií způsobí standardně otevření ventilu a zapnutí čerpadel. Pro realizaci jiné odstavovací funkce slouží bity (MaRDrevokotelVypCer a MaRDrevokotelZavVent).

Havarijní funkce je třeba důkladně promyslet v návaznosti na celý systém. Situace je značně komplikovanější než např. u plynových kotlů, které stačí při havárii odstavit a vypnout HUP.

4.5. Časové programy

Sem patří skupina funkcí pro realizaci uživatelských nastavení, závislých na datu, času a na týdenním cyklu - tzv. **kalendářů**.

Tyto funkce dohromady podporují tvorbu libovolného množství kalendářů. Základním stavebním prvkem kalendáře je denní profil. V samotném týdenním kalendáři pak uživatel každému dni v týdnu přiřadí právě jeden profil. Profil si uživatel nadefinuje v menu editace profilu. Každý profil umožňuje rozdělit den na nejvýše osm na sebe navazujících intervalů. V každém intervalu můžeme definovat stav spínače (tisknout např. VYP nebo ZAP) a žádanou hodnotu analogové veličiny v rozsahu 0..32767. Rozsah těchto hodnot je dán faktem, že pro analogovou žádanou hodnotu je vyhrazena proměnná word ve stacku, ale nejvyšší bit této proměnné obsadil právě zmíněný vypínač. Celkem je možné definovat maximálně 256 profilů. Ty jsou identifikovány svým pořadím tj. číslem 0..255. Programátor určí jaké profily budou dostupné pro konkrétní týdenní kalendář (podle typu ovládaného zařízení apod.). Editaci jednotlivých profilů lze pak také rozdělit podle použití v jednotlivých typech kalendářů a také určit rozsah, formát a jednotky editované veličiny daného profilu. Pro ovládání zařízení se pak používají funkce pro zjištění stavu příslušného kalendáře.

Pozn.: Funkce kalendářů a profilů nepoužívají standardní pracovní blok dat a negenerují žádné poruchy.

Příklad použití

Pro názornost budeme řešit následující příklad - potřebujeme vytvořit následující samostatné kalendáře (celkem 7):

- 1) kalendáře určující osvětlení tří pěstitelských kabin v režimu VYP/ZAP
- 2) kalendáře určující čas a dobu provětrávání těchto tří kabin v minutách
- 3) kalendář určující chod a žádanou teplotu přívodu VZT

První tři kalendáře (z bodu 1) budou využívat první čtyři profily (tj. profily 0..3) a umožní jen nastavení stavu VYP/ZAP. Další tři kalendáře (z bodu 2) budou využívat další čtyři profily (tj. profily 4..7) a umožní nastavovat čas v rozsahu 0..60 min. Kalendář pro VZT bude využívat další čtyři profily (tj. 8..11) a bude možné určit stav VYP/ZAP a pro stav ZAP bude navíc možné nastavit žádanou teplotu v rozsahu 0..30°C.

Samotný kalendář zabírá ve stacku prostor 7x word (pro každý den v týdnu se nastavuje žádaný profil). Můžeme tedy určit pro jednotlivé kalendáře adresy dat ve stacku takto. Začneme třeba od adresy 100. Pro zmíněných sedm kalendářů vycházejí tyto adresy: 100,107,114,121,128,135 a 142. STACK tedy bude obsazen až do adresy 148 a od položky 149 mohou být uložena data jednotlivých profilů. Každý profil spotřebuje 16x word a potřebujeme celkem 12 profilů. Adresy profilů 0..11 tedy budou:149, 165, 181, 197, 213, 229, 245, 261, 277, 293, 309, 325.

Nyní máme rozvržený zásobník (obsazen až do adresy 340) a můžeme psát program:

```

; umístění dat kalendářů na zásobníku:
Const KalOsvA=100, KalOsvB=107, KalOsvC=114
Const KalVetrA=121, KalVetrB=128, KalVetrC=135, KalVZT=142
; inicializace glob.proměnné určující umístění dat profilů na zásobníku:
if RESET then MaR_AdrProfily=149

; Menu editace jednotlivých kalendářů (celkem 7)
; (začlenění se do kompl.menu displeje, tvořeného z funkcí MenuLIB)
;-----
if MeNext("KALENDARE") then begin
  if MeNext("OSVETLENI A") then MaRDispKalendar("OSVETLENI A",KalOsvA,0,3)
  if MeNext("OSVETLENI B") then MaRDispKalendar("OSVETLENI B",KalOsvB,0,3)
  if MeNext("OSVETLENI C") then MaRDispKalendar("OSVETLENI C",KalOsvC,0,3)
  if MeNext("VETRANI A") then MaRDispKalendar("VETRANI A",KalVetrA,4,7)
  if MeNext("VETRANI B") then MaRDispKalendar("VETRANI B",KalVetrB,4,7)
  if MeNext("VETRANI C") then MaRDispKalendar("VETRANI C",KalVetrC,4,7)
  if MeNext("VZT") then MaRDispKalendar("VZT",KalVZT,8,11)
MeEnd()
end

; Menu editace jednotlivých typů profilů
;-----
if MeNext ("EDITACE PROFILU") then begin
  if MeNext ("PROFILY OSVETLENI") then begin
    if MaRLineProfily(0,3) then begin ; 4 profily osvětlení (0..3)
      MaRLineBit(TextVypZap) ; v profilu je bitová hodnota
      MeEnd()
    end
    MeEnd()
  end
  if MeNext ("PROFILY VETRANI") then begin
    if MaRLineProfily(4,7) then begin ; 4 profily větrání (4..7)
      Format=1
      MaRLineHodnota(0,60,0) ; v profilu je analog.hodnota
      if Me_DISP then Display("min")
      MeEnd()
    end
    MeEnd()
  end
  if MeNext ("PROFILY VZT") then begin
    if MaRLineProfily(8,11) then begin ; 4 profily VZT (8..11)
      MaRLineBit(TextVypZap)
      ; profil má bitovou (VYP/ZAP) i analogovou hodnotu (teplota)
      ; editace hodnoty je přístupná jen když je zapnutý bit:
      if MaRPrectiBit(MaR_Adr,15) then MaRLineHodnota(2832,3032,1)
      MeEnd()
    end
    MeEnd()
  end
  MeEnd()
end

; Hlavní funkce, zajišťující vyhodnocování všech profilů:
;-----
MaRSpravaProfilu(11)

```

K příkladu:

V menu "**KALENDARE**" přiřazujeme k jednotlivým dnům v týdnu pro jednotlivé části příslušné profily. Jednotlivým funkcím **MaRDispKalendar** se předkládají parametry určující úsek pole profilů, který je pro daný objekt relevantní (tedy pro všechna osvětlení se předává úsek profilů nastavujících osvětlení atd...)

V menu "**EDITACE PROFILU**" se pak editují jednotlivé profily. Menu se dělí na tři submenu podle vlastností, které od profilů očekáváme ("**PROFILY OSVETLENI**", "**PROFILY VETRANI**", "**PROFILY VZT**"). Parametry funkce **MaRLineProfily** jsou dolní a horní mez editovaných profilů v daném menu, tiskne se jen úvodní hlavička. Zde se určí jaký profil a jaký interval budeme editovat a lze zde posouvat časy jednotlivých intervalů. **MaRLineBit** edituje stav VYP/ZAP (používá pole textů "VYP","ZAP" z knihovny) a **MaRLineHodnota** edituje žádanou hodnotu s možností dalších nastavení (viz dále).

V editaci profilu VZT se nejprve edituje bit - hlavní "vypínač" vzduchotechniky (VYP/ZAP) a poté se vyhodnotí jeho stav a podmíněně se volá editor hodnoty profilu. Je-li bit ve stavu ZAP, zpřístupní se editace žádané teploty. Proměnná **MaR_Adr**, globálně definovaná knihovnou, ukazuje na word ve stacku editovaný funkcemi **MaRLineBit** a **MaRLineHodnota**. Proto se editace žádané teploty objeví jen když je VZT ve stavu ZAP.

MaRDispKalendar

Zobrazení a editace kalendáře

```
subroutine MaRDispKalendar(const string Titl : word Adr, ProfOd, ProfDo)
```

Předávané parametry

- 1.parametr (string): text nadpisu menu kalendáře
- 2.parametr (word): adresa (umístění) kalendáře v zásobníku
- 3.parametr (word): číslo prvního profilu použitého v kalendáři (profily 0..255)
- 4.parametr (word): číslo posledního profilu použitého v kalendáři (profily 0..255)

Popis

Funkce umožňuje přiřadit každému dnu v týdnu některý z dovolených profilů. Rozsah dovolených profilů pro tento kalendář je dán parametry ProfOd a ProfDo. Nastavovací menu kalendáře na displeji automatu má takovouto strukturu:

```
VZT
Pondeli -PROFIL: 8
Utery   -PROFIL: 8
Streda   -PROFIL: 8
Ctvrtek -PROFIL: 8
Patek    -PROFIL: 9
Sobota   -PROFIL:10
Nedele   -PROFIL:11
```

MaRLineProfily

Vytvoření menu pro editaci profilu

subroutine MaRLineProfily (word Min, Max)

Předávané parametry

- 1.parametr (word): první profil editovatelný v tomto menu (profily 0..255)
- 2.parametr (word): poslední profil editovatelný v tomto menu (profily 0..255)

Popis

Menu pracuje s blokem profilů na zásobníku, který je umístěn od adresy dané proměnnou **MaR_AdrProfily**. Je tedy nutné tuto adresu do **MaR_AdrProfily** při inicializaci celého programu (při nastaveném bitu RESET) správně nastavit (viz náš příklad v úvodu kapitoly).

Vlastní editace profilů obsahuje dva vnořené displeje. V prvním menu pouze šipkami doleva a doprava vybereme editovaný profil a šipkami dolů a nahoru vybereme editovaný interval. Stiskem klávesy ENT se dostaneme do vnořeného menu nastavení vlastních parametrů zvoleného intervalu. Menu mají takovouto strukturu:

1.menu (výběr intervalu)

```
PROFIL:  4
INTERVAL: 1
INTERVAL: 2
INTERVAL: 3
INTERVAL: 4
INTERVAL: 5
INTERVAL: 6
INTERVAL: 7
INTERVAL: 8
```

---->

podmenu (editace intervalu)

```
PROFIL: 4  INT:  6
OD: 0:00
DO: 23:40
ZMENIT STAV:  ZAP
HODNOTA:      +22,3 C
```

V menu nastavení intervalu ještě první tři řádky tiskne ještě funkce LineProfily. Na prvním řádku je zvolený profil a interval. Druhý řádek zobrazuje začátek daného intervalu. Čas "OD" je vždy shodný jako čas "DO" předchozího intervalu. Ve třetím řádku lze pro intervaly 1 až 7 nastavit čas "DO". Pro osmý interval je čas "DO" logicky vždy 24:00. To znamená, že aktivní bude zvolený interval právě tehdy když aktuální čas bude patřit do polouzavřeného intervalu <OD, DO). Jinými slovy čas "OD" patří do nastaveného časového intervalu, čas "DO" patří již do následujícího časového intervalu. Nastavování času "DO" směrem zpátky je omezeno časem "DO" předchozího intervalu. Při posouvání času "DO" směrem dopředu se automaticky posouvají všechny časy navazujících intervalů (a ty se pak návazně mohou rovněž posunout). Pokud např. 1. interval nastavíme : OD 0:00 DO: 24:00, tak žádné další intervaly se již neuplatní.

Čtvrtý a pátý řádek již není v režii funkce **MaRLineProfily**, pro tisk těchto řádků slouží další funkce **MaRLineBit** a **MaRLineHodnota**. Je to proto, aby si programátor mohl upravit řádky s editací těchto parametrů profilu dle svých představ (vč. formátu tisku hodnot, příp. dotisku jednotek a dalších informací). Samozřejmě je možné použít jen řádek s editací bitu nebo jen řádek s editací hodnoty (viz náš úvodní příklad).

MaRLineBit, MaRLineHodnota

Vytvoření řádku menu s editací hodnoty v příslušném intervalu profilu

```
subroutine MaRLineBit (const Me_tbl2txt Stav)
subroutine MaRLineHodnota (word Min, Max : bit EditacevC)
```

Předávané parametry pro MaRLineBit

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

Předávané parametry pro MaRLineHodnota

1.parametr (word): dolní limit editované hodnoty

2.parametr (word): horní limit editované hodnoty

3.parametr (bit): formát: 0 = horní limit editované hodnoty

Popis funkce

MaRLineBit používá pro editaci bitu standardní editor bitu poskytovaný knihovnou MenuLIB. Do funkce se předává jako parametr předdefinovaná tabulka dvou textů, označujících stavy 0 a 1. Můžeme využít tabulku textů definovanou v knihovně, nebo si zadefinovat svoje vlastní texty.

Parametry funkce MaRLineHodnota jsou: dolní a horní mez editované proměnné a parametr typu bit, kde 0 znamená běžný způsob editace s nastavitelným formátem a možností dotisku jednotek a 1 znamená editaci teploty tj. v desetinách Kelvina se zobrazením v desetinách Celsia. Je to praktické a odpovídá to zavedenému standardu interpretace teploty v proměnné.

Obě funkce pracují s datovou proměnnou typu word právě vybraného intervalu (MaRLineBit edituje její nejvyšší bit a MaRLineHodnota spodních 15 bitů, jak bylo uvedeno v úvodu). Proměnná je ve STACKu v bloku profilů a její aktuální adresu nastavuje funkce MaRLineProfily do globální proměnné **MaR_Adr** (tuto adresu do zásobníku může využít i programátor při realizaci speciální logiky, jako např. v našem úvodním příkladu).

MaRSpravaProfilu

Správce kalendářů a profilů

```
subroutine MaRSpravaProfilu (byte MaxProfil)
```

Předávané parametry

1.parametr (byte): číslo max. profilu

Popis funkce

Zajišťuje průběžnou aktualizaci hodnot kalendáře podle času automatu i kontinuitu jednotlivých intervalů všech profilů. Je to základní funkce, která je nutná pro běh kalendářů. Vstupním parametrem je nejvyšší číslo použitého profilu - parametr je důležitý, neboť udává celkovou délku dat na zásobníku, kterou funkce bude zpracovávat - tedy pro celkem 12 zdefinovaných profilů předáváme parametr 11 (pole profilů se zpracovává vždy od čísla 0).

Volání funkce je třeba vřadit někam do hlavní smyčky programu, aby se provádělo neustále a bez podmínek.

Jedním voláním této funkce (tedy při každém průchodu hlavní smyčkou) se vždy zpracuje jeden interval jednoho profilu. Díky tomu se ani při velkém množství různých profilů chod programu nezpomaluje.

MaRStavKalendareB, MaRStavKalendareW

Přečtení stavu výstupu kalendáře

```
function bit MaRStavKalendareB (word Adr)
function word MaRStavKalendareW (word Adr)
```

Předávané parametry

1.parametr (word): adresa umístění příslušného kalendáře v zásobníku

Popis funkce

Funkce poskytují finální stav výstupního bitu (0/1) a výstupní hodnoty (0...32767) příslušného kalendáře. Výstup těchto funkcí pak slouží k realizaci časových průběhů regulovaných procesů podle nastavení jednotlivých profilů a jejich přiřazení jednotlivým dnům v kalendáři.

4.6. Pomocné funkce

MaRAI, MaRAO

Řízení rozsahu analogových vstupů a výstupů

```
function word MaRAI(word Min, Max, Input, TepMin, TepMax)
function word MaRAO(word Min, Max, Output)
```

Funkce MaRAI

Předávané parametry

word Min: minimální hranice vstupní hodnoty
word Max: maximální hranice vstupní hodnoty
word Input: vstupní hodnota
word OutMin: minimální hranice výstupní hodnoty
word OutMax: maximální hranice výstupní hodnoty

Výstupní hodnota

word Out: transformovaná vstupní hodnota

Popis funkce

Funkce MaRAI (Analog Input) slouží k transformaci rozsahu vstupních hodnot (nejčastěji analogového vstupu automatu) na požadovaný rozsah hodnot výstupních (tj. nejčastěji k výpočtu teploty). Není-li vstupní hodnota v zadaných mezích (Min, Max), dává funkce na výstupu 0. Nulovou hodnotu pak vyhodnotí regulační funkce jako chybu čidla.

Příklad

Dává-li např. analogový teploměr připojený na proudový vstup I16 proud v rozsahu 4..20mA při rozsahu teplot 0..100°C, pak můžeme funkcí MaRAI např. tímto způsobem připravit správnou hodnotu teploty pro funkci MaRZpracujTep:

```
MaRZpracujTep(AdrPar, IDPar, MaRAI(400, 2000, I16, 2732, 2832))
```

kde AdrPar je adresa pracovního bloku dat knihovny funkce pro kterou teplotu připravujeme a IDPar je identifikátor příslušného parametru.

Funkce MaRAO

Předávané parametry

word MinOut: minimální hranice výstupní hodnoty
word MaxOut: maximální hranice výstupní hodnoty
word Output: hodnota k transformaci na výstup

Výstupní hodnota

word Out: transformovaná vstupní hodnota

Popis funkce

Funkce MaRAO (Analog Output) slouží k přizpůsobení analogového výstupu knihovní procedury rozsahu analogového vstupu akčního prvku (nejčastěji servopohonu). Vstupní hodnota je vždy v rozsahu 0...1000 a je transformována na výstupní hodnotu v rozsahu MinOut...MaxOut.

Pro výstup platí ještě tyto podmínky:

Je-li Output=0, funkce vrací hodnotu 0, je-li $0 < \text{Output} < 1000$, pak funkce vrací hodnotu v rozsahu MinOut...MaxOut.

Příklad

Má-li např. servo připojené na analogový výstup O16 rozsah 2V..10V, pak:

$$O16 = \text{MaRAO}(200, 1000, \text{Output})$$

kde Output je výstup funkce, který potřebujeme transformovat.

Pokud potřebujeme chod serva "otočit", je možné funkci MaRAO použít takto:

$$O16 = \text{MaRAO}(1000, 200, \text{Output})$$

Potom je-li $0 \leq \text{Output} < 1000$, pak funkce vrací hodnotu v rozsahu 1000...200, je-li Output=1000, pak funkce vrací hodnotu 0.

MaRHavSig

Havarijní signalizace

```
subroutine MaRHavSig(word AdrPar)
```

Předávané parametry

word AdrPar: adresa (umístění) pracovního bloku dat v zásobníku

Struktura pracovního bloku dat :

celková délka pracovního bloku (x WORD) :			2	
identifikátor		význam	typ	adresa
	vstupy			
	MaRHavSigVAS	vypnutí akustické signalizace	bit	0?13
	výstupy			
	MaRHavSigOS	optická signalizace	bit	0?14
	MaRHavSigAS	akustická signalizace	bit	0?15
	parametry			
	MaRHavSigOSPor	svítit při "MaR_Por"	bit	0?0
	MaRHavSigOSPorNew	svítit při "MaR_PorNew"	bit	0?1
	MaRHavSigPOSHavA	blikat při "MaR_HavA"	bit	0?2
	MaRHavSigPOSHavB	blikat při "MaR_HavB"	bit	0?3
	MaRHavSigPOSHavC	blikat při "MaR_HavC"	bit	0?4
	MaRHavSigASHavA	trvale houkat při "MaR_HavA"	bit	0?5
	MaRHavSigASHavB	trvale houkat při "MaR_HavB"	bit	0?6
	MaRHavSigASHavC	trvale houkat při "MaR_HavC"	bit	0?7
	MaRHavSigPASHavA	přerušovaně houkat při "MaR_HavA"	bit	0?8
	MaRHavSigPASHavB	přerušovaně houkat při "MaR_HavB"	bit	0?9
	MaRHavSigPASHavC	přerušovaně houkat při "MaR_HavC"	bit	0?10
		vnitřní stavy	W	1

Pozn.: Jako vstupy dále využívá bity: MaR_HavA, MaR_HavB, MaR_HavC, MaR_Por a MaR_PorNew.

Popis funkce

Od globálních poruchových stavů odvozuje optickou a akustickou signalizaci.

Na výstupy z pracovního bloku přímo "připojíme" výstupy automatu pro světlenou a akustickou signalizaci. Na vstup lze připojit buď tlačítko "odstavení akustické signalizace" nebo třeba použít konstrukci:

```
if KBCode>0 then MaRZapisParBit(AdrPar, MaRHavSigVAS,1)
```

tzn. stiskem jakéhokoli tlačítka odstavit houkačku. Procedura si po odstavení houkačky tento bit sama nuluje, není třeba se o to v programu starat.

Funguje to tedy následovně:

Vznikne havarijní stav, spustí se akustická signalizace, přijde obsluha a odstaví stiskem tlačítka houkačku. Odstraní havárii a tím je již signalizace opět připravena k další funkci. Nastavením bitových parametrů zvolíme požadovanou funkci. Logika vychází z toho, že poruchu signalizujeme trvalým svitem, havárii akusticky a přerušovaně. Přerušovaná signalizace je vždy preferována před trvalou, tzn. nastane-li porucha a havárie současně, bude přednostně signalizována havárie. Procedura negeneruje žádné poruchové slovo a proto se volá jen s parametrem AdrPar.

Příklad použití

Ukázka začlenění funkce do programu, vč. kvitování klávesou a připojení výstupů (a zkusíme použít např. reléovou periférii EX-02, připojenou na síťové bity M64..M71). Optická signalizace budiž M64, akustická M65.

```
; zdefinování umístění pracovního bloku pro MaRHavSig na STACK :
const HavSig1=98
; vlastní program:
; kvitování signalizace libovolnou klávesou:
if KBCode>0 then MaRZapisParBit(HavSig1, MaRHavSigVAS,1)
; volání knihovní funkce:
MaRHavSig(HavSig1)
; připojení výstupů funkce např. na síťové bity:
M64=MaRPrectiParBit(HavSig1, MaRHavSigOS)
M65= MaRPrectiParBit(HavSig1, MaRHavSigAS)
```

Potřebujeme-li signalizovat více poruchových stavů, můžeme tuto knihovní funkci volat s různými parametry několikrát.

MaRTiskVentil

Zobrazení stavu trojbodového servopohonu

```
subroutine MaRTiskVentil(bit Otvira, Zavira)
```

Vytiskne na displej stav trojbodového ventilu. Do parametrů je možné napsat přímo výstupy automatu, kam je servo připojeno - např. MaRTiskVentil(Y0,Y1). Od nastavené pozice (POSITION) se na displej vytiskne jeden ze stavů:

"OTVIRA", "ZAVIRA" nebo "STOJI".

MaRTiskTep

Zobrazení teploty

```
subroutine MaRTiskTep(word tep)
```

Teplota je reprezentována hodnotou v desetínách Kelvina. Funkce tiskne takto reprezentovanou hodnotu v parametru tep ve formátu desetinného čísla ve stupních Celsia. Takto pracuje pouze pokud $110 < \text{tep} \leq 10000$. Je-li $\text{tep} < 111$, vytiskne se "**VYP**".

To je vhodné např. při tisku žádané hodnoty okruhu UT vypočtené již popsanou funkcí MaRVypoctiEkv. Je-li okruh vypnutý, funkce vrátí 0 a tiskne se automaticky "**VYP**".

MaRTiskCas

Tisk časových údajů

```
subroutine MaRTiskCas(word cas)
```

Čas bývá uvnitř knihovních funkcí (např. v profilech) interpretován jako minuta daného dne tj. hodnota 720 znamená 720. minutu dne, tj. čas 12:00. Parametrem této funkce je právě čas ve formátu minuty dne. Od nastavené pozice (POSITION) se na displej vytiskne čas ve formátu hodiny:minuty.

MaREditCas

Editor hodnot času

```
subroutine MaREditCas(var word iodata : int max)
```

Umožňuje, podobně jako funkce MeEdit z knihovny MenuLIB, komfortní editaci proměnné "iodata" až do omezení daného druhým parametrem. Čas v editované proměnné má formát času (viz výše popsaný formát typu minuta dne), ale vlastní editace je na displeji zobrazena ve formátu hodiny:minuty. Omezení v parametru max je ale pro přehlednost ve formátu Hodiny*100+Minuty. Pro omezení editovaného času např. do 23:00 bude parametr max=2300. Editovaná hodnota tak bude nejvýše 1380. Parametr max byl vlastně zabudován jen proto, aby umožnil editaci v rozmezí 0:00 až 23:59 nebo volitelně 0:00 až 24:00.

MaREditTep

Editor hodnot teploty

```
subroutine MaREditTep(var word iodata : int min : int max)
```

Podobně jako předchozí editor i tento umožňuje komfortní editaci proměnné "iodata" v mezích daných parametry. Teplota v editované proměnné má formát desetiny Kelvina, ale tiskne se a edituje v destínách Celsia.

MaRTiskDatum

Zobrazení data

```
subroutine MaRTiskDatum(word datum)
```

Vytiskne od nastavené POSTION datum ve formátu den.měsíc. Proměnná datum je ve formátu den*100+měsíc. Je-li datum >3112 , tiskne pouze mezery.

MaRMaxOf

Výběr maximální hodnoty

```
function word MaRMaxOf(word A, B, C, D)
```

Vrací nejvyšší z čísel A,B,C,D. Funkci lze volat i s nižším počtem argumentů.

MaRMinOf

Výběr minimální hodnoty

```
function word MaRMinOf(word A, B, C, D)
```

Vrací nejnižší z čísel A,B,C,D. Funkci lze volat i s nižším počtem argumentů.

5. Parametrizace

Všechny pracovní bloky dat (a též kalendáře, profily a poruchová slova) se všemi potřebnými parametry jsou uloženy v zásobníku. Je tedy potřebné nějakým přehledným způsobem vše naparametrizovat. To znamená vytvořit tabulku, která bude obsahovat všechny použité parametry a tuto tabulku pak zapsat do zásobníku automatu.

Proto je tu aplikace Stack1000.xls. Je to aplikace pro MS Excel, která umožní snadno provést všechny výše uvedené operace a navíc dokáže i vyčíst data ze zásobníku automatu zpět do tabulky. Aplikace se jmenuje Stack1000, protože pracuje s tabulkou velikosti 1000 (100x10) WORD. Pro odlišení novějších verzí index 1000 postupně zvyšujeme, ale základní vlastnost tj. práce s 1000 položkami zásobníku zůstává.

Aby aplikace správně fungovala, je třeba mít nainstalovanou sadu MS Office pro WindowsXP (nebo alespoň samotný MS Excel). Bohužel starší verze Excelu nepodporují všechny funkce, které tato aplikace používá. V programu Excel je nutné nastavit úroveň zabezpečení na "střední" a při spuštění aplikace povolit makra. Nakonec pro komunikaci s automatem je třeba spustit DDE server PESdde (při spuštění prostředí StudioWin je server spuštěn automaticky).

Vlastní aplikace má celkem 17 listů. Výjimečný je první list ("Zásobník"). Ten obsahuje právě onu již zmiňovanou tabulku 1000xWORD. Pomocí této tabulky "komunikuje" MS Excel s automatem. Zde je třeba nastavit adresu automatu a specifikovat umístění (adresu v zásobníku) první položky tabulky. Stiskem tlačítek "PřečtiStack" nebo "Zapiš Stack" pak dojde k vlastní výměně dat mezi aplikací a automatem.

Pracovní bloky dat

Další listy již obsahují příslušné parametry. Listy jsou řazeny podle typu regulačních funkcí, které se jimi parametrizují. Na daném listu jsou vždy doporučené hodnoty a v informačních oknech jsou vždy vypsány všechny parametry, které je nutné nastavit. Tyto listy jsou vybaveny tlačítkem "Kopíruj". Po jeho stisknutí dojde automaticky k přepsání parametrů pracovního bloku dat dané funkce do listu "Zásobník" od zvolené adresy. Parametry se přepíší do tabulky včetně barevného pozadí. To je jednak velmi přehledné (barvy pozadí parametrů souhlasí s barvami záložek jednotlivých listů, podle barvy pozadí poznáme typ knihovní funkce která parametry využívá) a jednak bezpečné (pole parametrů je barevně odlišeno a při přepisování již jednou zapsaných hodnot na to program upozorňuje). Výjimkou je parametrizace poruchových stavů (k tomu později).

Parametrizace tedy vypadá v praxi takto:

- a) Nastavíme v listu "Zásobník" adresu automatu a umístění počátku tabulky v zásobníku automatu.
- b) Zvolíme typ knihovní funkce, tj. otevřeme příslušný list.
- c) Zkontrolujeme podle informačních oken jednotlivé parametry pracovního bloku dat a případně je upravíme.

- d) Nastavíme adresu, od které je pracovní blok dat s parametry umístěn v zásobníku.
- e) Stiskneme tlačítko "Kopíruj" a zkontrolujeme v listu "Zásobník" přepsání parametrů.
- f) Body 2)..5) opakujeme pro naparametrizování všech pracovních bloků dat, které náš program potřebuje.
- g) připojíme automat k počítači a stisknutím tlačítka "Zapiš Stack" v listu "Zásobník" překopírujeme tabulku do automatu.

Listy příslušné k funkcím "KO" a "Profily" umožňují vícenásobné kopírování. Při nastavení příslušného parametru na daném listě na hodnotu větší než 1 dojde při kopírování k patřičnému rozmnožení. Jednotlivé bloky parametrů se nakopírují těsně za sebe. Je to výhodné, zrychlí to práci a ušetří starost s počítáním správných adres jednotlivých bloků parametrů.

Poruchy

Nakonec zbývá parametrizace poruchových stavů. Té patří poslední list ("Poruchovka"). V první etapě je nutné nastavit globální parametry : MaR_AdrVyskyt, MaR_Pocet4B a MaR_ZacatekPor v souladu s programem pro daný automat. Toto nastavení je nutné potvrdit stisknutím přepínače "Akceptuj nastavení globálních adres". Tím se uvolní další volby. V dalším kroku se parametrizují jednotlivé poruchy jednotlivých poruchových slov:

- a) Vybereme typ knihovní funkce, která vytvořila dané poruchové slovo a nastavíme parametry AdrPor a Počet zpracovávaných poruch. V zeleném okně se objeví konkrétní názvy daných poruch.
- b) Postupně vybíráme jednotlivé poruchy v zeleném okně a v hnědém okně přiřadíme každé poruše zvolené vlastnosti.
- c) Po každém nastavení konkrétní poruchy stiskneme tlačítko "Kopíruj". Každou dostupnou poruchu v zeleném poli je nutné naparametrovat tj. zatrhnout ji, vybrat vlastnosti a stisknout "Kopíruj".

Naparametrizujeme nejlépe všechny poruchy najednou a pak zkontrolujeme v listu "Zásobník" vlastní parametry. Pozor! "obarvené" parametry nemusí v listu zásobník od MaR_AdrVyskyt tvořit souvislý blok. To proto, že i poruchy, které nevyhodnocujeme, zabírají místo. Protože jsem některé ani neparametrovali, nemohou být v listu "Zásobník" barevně označeny.