ON-LINE ZMĚNA PROGRAMU

Úvod

On-line změna programu je vlastnost centrální jednotky PLC Tecomat, která umožňuje provádět úpravy uživatelského programu bez zastavení řízení technologie, tj. bez nutnosti odstavit řízenou technologii při úpravách PLC programu. Tato vlastnost dává programátorovi systému Tecomat možnost provádět úpravy změny PLC programu takzvaně za chodu. Odpovědnost za správnost prováděných úprav je samozřejmě na programátorovi systému. Centrální jednotka PLC ve spolupráci s programovacím prostředím Mosaic zajišťuje bezpečné provedení změn v jednom okamžiku tak, aby plynulost řízení nebyla ohrožena.

On-line změnu programu je možno provádět v sytému TC700 s centrálními jednotkami CP-7001 a CP-7002 od verze SW v4.1. Podpora on-line změny programu je integrována v programovacím prostředí Mosaic od verze v1.5.10. Chování při on-line změně si lze také vyzkoušet se simulátorem PLC v prostředí Mosaic.

Základní princip činnosti

Pro vysvětlení základní principu použijeme následující příklad. Předpokládejme, že PLC Tecomat řídí technologii, jejíž odstavení znamená značnou ekonomickou ztrátu např. vypalovací pec a programátor má za úkol upravit PLC program. V této chvíli je vcelku lhostejné, zda se bude jednat o opravu chybného algoritmu řízení nebo přidání nové funkce např. pro vypalování dalšího sortimentu výrobků. Program pro PLC je prostě třeba upravit a řízení pece se nesmí ani na okamžik zastavit. On-line změna programu nabízí řešení této situace. Programátor provede příslušné úpravy PLC programu a centrální jednotka PLC zajistí přepnutí ze starého na nový program tak, že n-tý cyklus výpočtu je kompletně proveden podle původního programu a následující cyklus se provede podle nového programu. Centrální jednotka zároveň zajistí potřebné činnosti spojené se změnami proměnných tak, aby plynulost řízení nebyla narušena.

Možnosti on-line změn

V rámci on-line změny může programátor PLC systému Tecomat upravovat následující části programu:

- kód programu, tzn. libovolné úpravy všech částí programu (funkcí, funkčních bloků, programů) včetně vkládání nových POU resp. jejich vypouštění
- úpravy rozhraní POU, tj. změny vstupních a výstupních proměnných POU včetně jejich přidávání a vypouštění
- úpravy proměnných, tj. vkládání a vypouštění všech typů proměnných (lokálních i globálních) resp. změna proměnných jako např. změna rozměru pole
- úpravy datových typů, např. změny ve strukturách, přidávání nových datových typů a vypouštění nepoužitých datových typů
- úpravy velikosti remanentní zóny

Následující úpravy nelze v rámci on-line změn programu provádět:

- změny hw konfigurace systému, např. přidávání IO modulů nebo změna typu IO modulu
- změny nastavení IO modulů

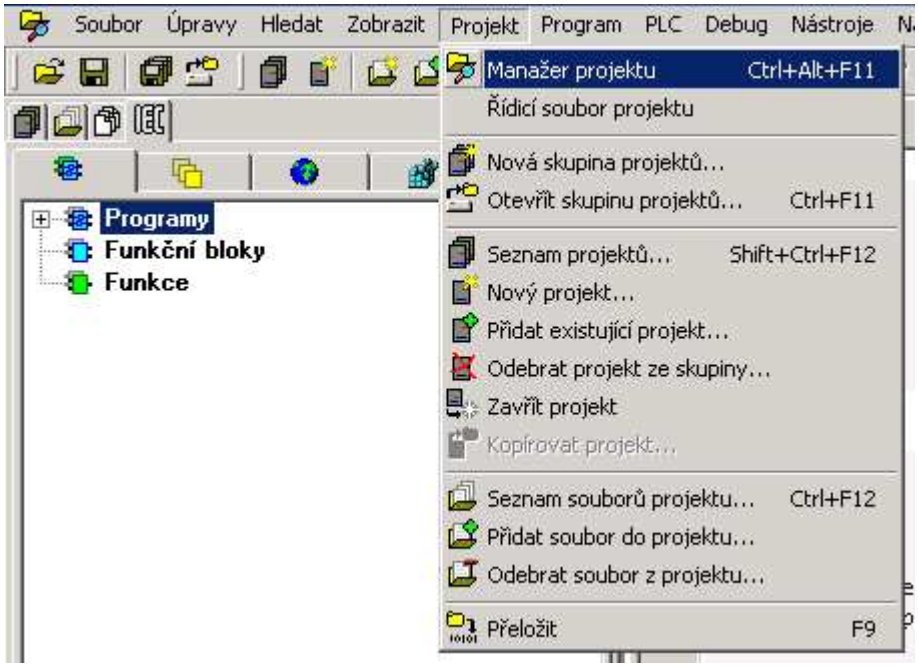
- změny v nastavení komunikačních parametrů pro sériové kanály
- změny v síti PLC

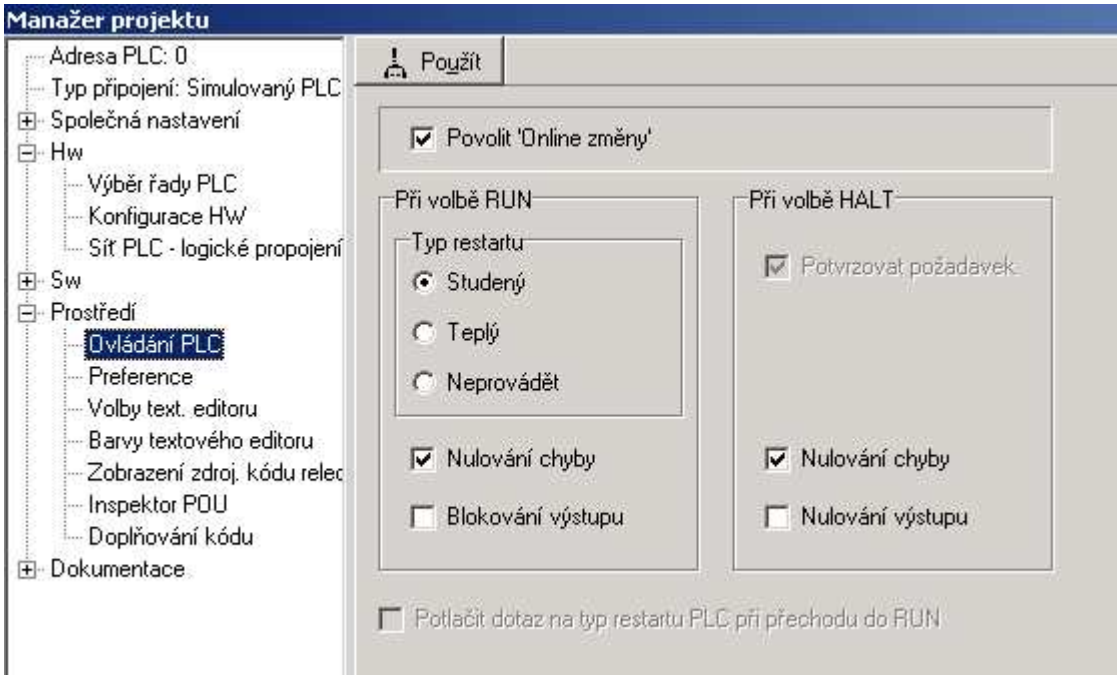
Zapnutí podpory on-line změn v prostředí Mosaic

On-line změny programu lze zapnout následujícím postupem:

- zvolit v menu Projekt | Manažer projektu (CTRL+ALT+F11)
- ve stromu projektu vybrat uzel Prostředí | Ovládání PLC
- vybrat položku „Povolit on-line změny“

Postup zapnutí podpory on-line změn ukazují následující obrázky.



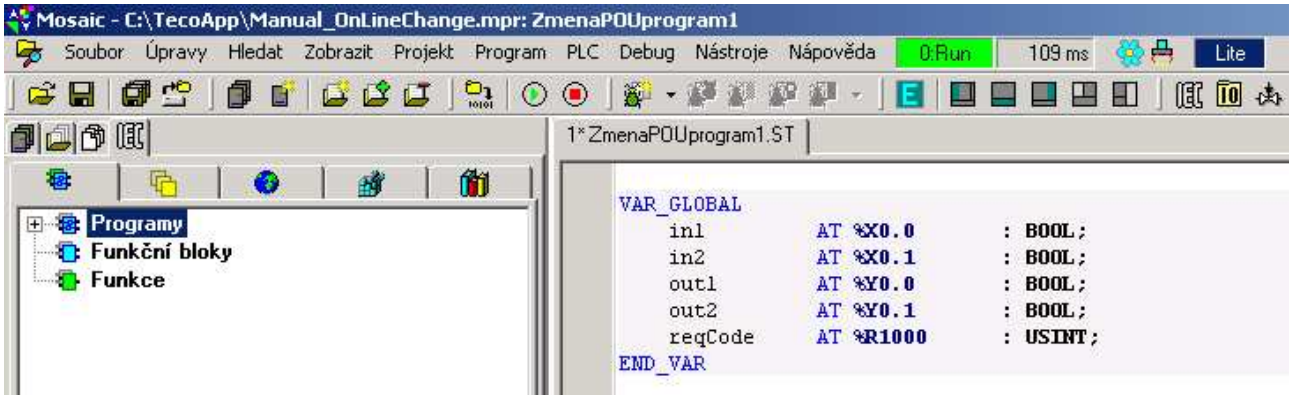


Pokud centrální jednotka PLC nepodporuje on-line změny, položka povolující on-line změny bude šedivá a zaškrtávací box nepůjde vybrat.

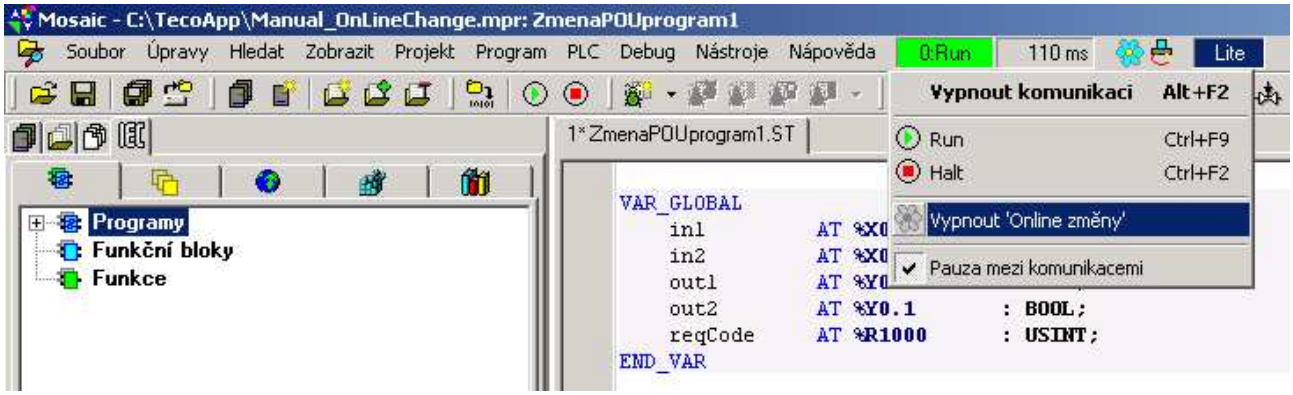
Pokud jsou on-line změny zapnuty, je také automaticky zapnutá volba „Potvrzovat požadavek při volbě HALT“, což znamená že přechod PLC do režimu HALT je nutno potvrdit ve zvláštním dialogu a nelze tedy zastavit řízení pouze stiknutím ikony HALT nebo klávesovou zkratkou CTRL+F2. Dále nelze potlačit dotaz na typ restartu poté, co se poprvé nahraje nový program do PLC. Dialog s dotazem na typ restartu je zobrazen vždy když se poprvé nahrává nový program do PLC (tj. v PLC ještě není žádný program, proti kterému by bylo možno vyhodnotit změny).

Signalizace podpory on-line změn v prostředí Mosaic

Zapnutá podpora on-line změn je v prostředí Mosaic signalizovaná v liště Menu ikonou se symbolem květiny  . Pokud je ikona barevná, podpora on-line změn je zapnutá. Je-li ikona květiny šedivá, on-line změny jsou vypnuté a každá změna v programu povede na zastavení řízení při nahrávání nového programu do PLC.



On-line změny lze také zapnout resp. vypnout z menu, které se objeví po stisknutí levého tlačítka myši nad ikonou on-line změn nebo nad ikonou se symbolem připojeného PLC jak ukazuje následující obrázek.



Stejně jako v předchozím postupu, lze on-line změnu zapnout pouze v případě, že centrální jednotka PLC tyto změny podporuje. V opačném případě je volba neaktivní (šedivá) a nelze ji vybrat.

1. On-line změny programu v jazyce ST

V této kapitole se budeme věnovat on-line změnám v programu, který je napsán v jazyce ST.

Zahájení programování PLC se zapnutými on-line změnami

Poté co zapneme podporu on-line změn programu, je samozřejmě potřeba napsat první verzi programu, úspěšně tento program přeložit (např. F9) a nahrát jej do centrální jednotky PLC (např. CTRL+F9). Před nahráváním programu do PLC zkontroluje prostředí Mosaic zda v PLC existuje nějaká předchozí verze nahrávaného programu. Pokud neexistuje (např. jedná se o úplně nový program a novou centrální jednotku), pak se program nahraje do PLC standardním postupem. To znamená, že se centrální jednotka nastaví do režimu HALT, výstupy PLC se zablokují, nahraje se program do PLC a centrální jednotka přejde do režimu RUN se zadaným typem restartu. Poté je centrální jednotka PLC ve spolupráci s programovacím prostředím Mosaic připravena akceptovat změny v programu bez zastavení řízení.

Pro další výklad zvolíme následující příklad. Předpokládejme, že program v PLC zapíná a vypíná chlazení podle údaje o teplotě centrální jednotky. Proměnná *temp_CP7002* udává teplotu ve stupních Celsia (systémový registr S36). Výstup *cool* je zapnutý pokud teplota překročí 50 stupňů.

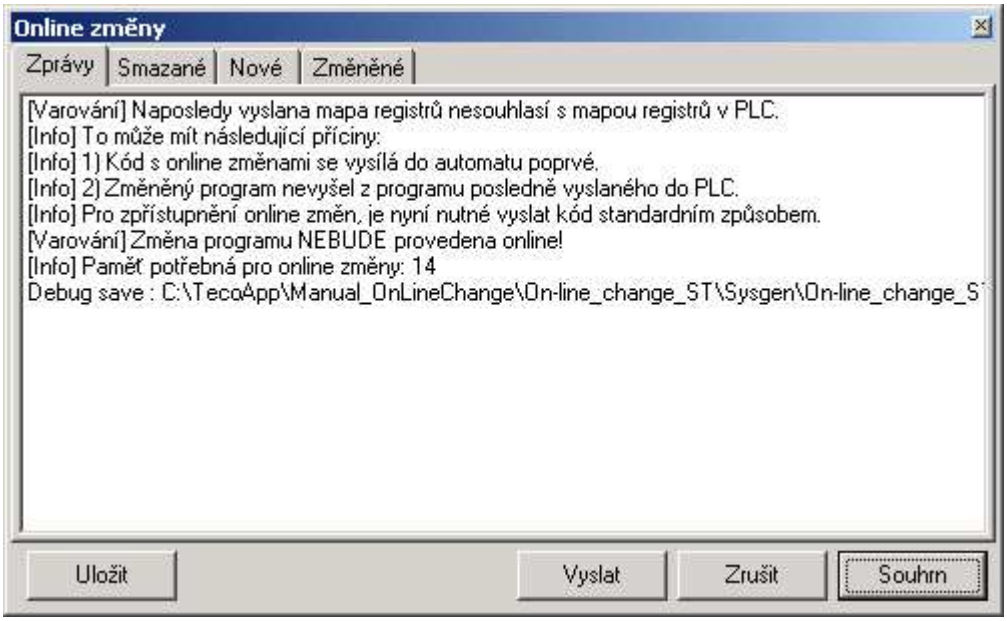
```
VAR_GLOBAL
  temp_CP7002    AT %S36      : SINT;   // CPU temperature
  cool           AT %Y0.0     : BOOL;   // cooling
END_VAR

PROGRAM Prog1

  CASE temp_CP7002 OF
    0..50   : cool := false;
    51..127 : cool := true;
  END_CASE;
END_PROGRAM

CONFIGURATION ExampleOnLineChange
  RESOURCE CPM
    TASK FreeWheeling(Number := 0);
    PROGRAM main WITH FreeWheeling : Prog1 ();
  END_RESOURCE
END_CONFIGURATION
```

Přeložíme program (F9) a vyšleme jej do centrální jednotky (CTRL+F9). Předtím, než se kód programu začne vysílat, objeví se dialog s informacemi o on-line změnách.

Pokud zvolíme „Vyslat“ kód programu bude vyslán standardním způsobem, protože se jedná o nový program. To znamená, že centrální jednotka přejde do režimu HALT (kde jsou typicky zablokované výstupy), nahraje se nový program, provede se restart a poté centrální jednotka přejde do režimu RUN.

Pokud zvolíme „Zrušit“ kód programu se nevyšle a centrální jednotka zůstane beze změny s původním programem a v původním režimu.

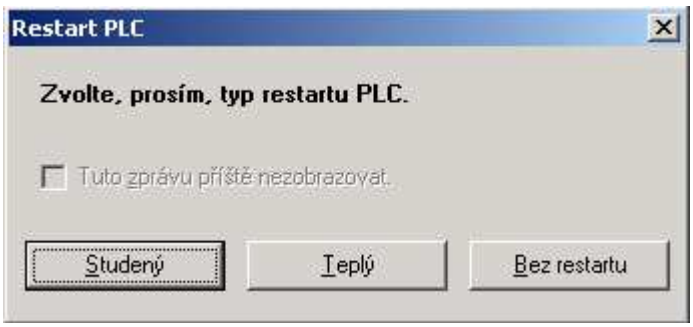
Volba „Podrobnosti“ umožňuje zobrazit doplňkové informace.



Volby „Vyslat“ a „Zrušit“ mají stejný význam, jako v předchozím dialogu. Volba „Souhrn“ provede návrat k předchozímu dialogu. Volba „Uložit“ uloží veškeré informace o on-line změně do textového souboru.

Záložka „Zprávy“ obsahuje souhrnnou informaci o on-line změně. Záložky „Smazané“, „Nové“ a „Změněné“ nesou informace o změnách proměnných a našem případě jsou prázdné, protože zahajujeme on-line změnu (nahráváme nový program).

Vyšleme tedy nový kód do PLC volbou „Vyslat“. Poté se zobrazí dialog s volbou restartu.



Po volbě restartu přejde centrální jednotka do režimu RUN. Od této chvíle je možné upravovat program bez zastavení řízení (tedy on-line).

Změny v kódu programu

Předpokládejme, že předcházející program je potřeba doplnit o ovládání topení. Pokud bude teplota měřená centrální jednotkou meší než 0 stupňů Celsia, zapneme výstup pro ovládání topení.

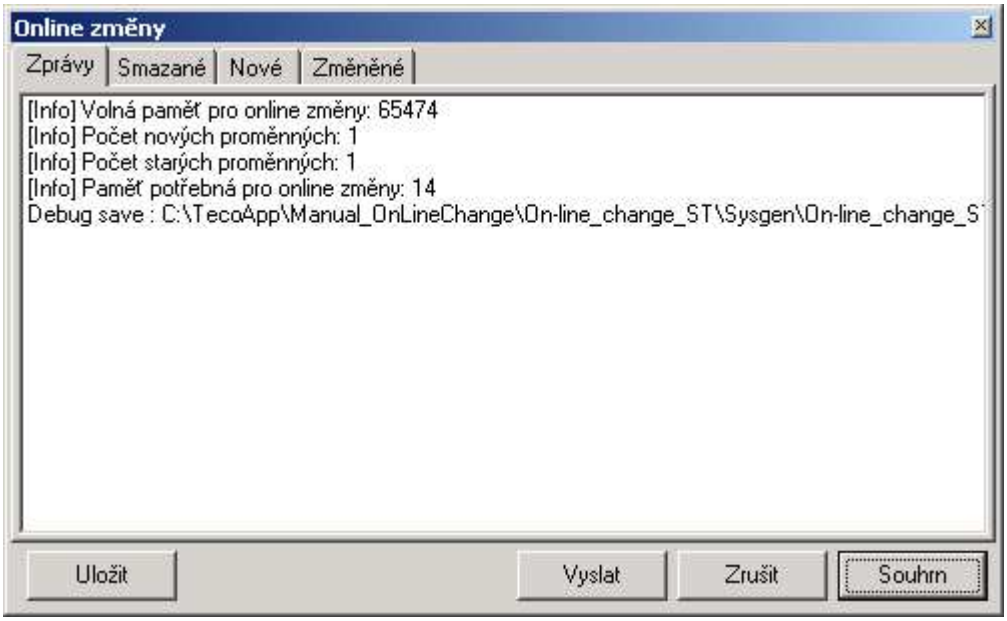
První věc, kterou si musíme uvědomit, je stav oken editoru v prostředí Mosaic. Po nahrání nového programu a přechodu do režimu RUN jsou všechna okna editoru ve stavu DEBUG, což znamená, že není možno editovat text. Tento stav signalizuje ikona  v levém dolním rohu editorového okna. Přepnutí do stavu EDIT lze provést klinutím levého tlačítka myši na této ikoně nebo horkou klávesou ALT+F6. Stav EDIT je signalizován ikonou  .

Nyní můžeme upravit program. Přidáme definici pro výstup topení *heat* a do programu doplníme jeho ovládání. Doplněné části mají světle tyrkysový podklad.

```
VAR_GLOBAL
  temp_CP7002     AT %S36       : SINT;    // CPU temperature
  cool            AT %Y0.0      : BOOL;    // cooling
  heat            AT %Y0.1      : BOOL;    // heating
END_VAR

PROGRAM Prog1

  CASE temp_CP7002 OF
    0..50    : heat := false; cool := false;
    51..127 : heat := false; cool := true;
    ELSE
              heat := true;  cool := false;
  END_CASE;
END_PROGRAM
```

Dále je postup standardní: přeložit upravený program a vyslat kód do PLC. A protože máme zapnuté on-line změny a v centrální jednotce existuje předchozí verze našeho programu, tak nový program bude akceptován centrální jednotkou bez zastavení řízení. Před vysláním programu do PLC se opět zobrazí dialog s informacemi o provedených on-line změnách. Tentokrát bude vypadat následovně.

Dialog nás informuje o tom, že změny byly provedeny pouze v kódu programu. Tuto skutečnost lze ověřit v podrobnostech.



Změny v proměnných programu

Pro ukázku této vlastnosti doplníme program z předchozího odstavce o proměnnou, která bude evidovat počet případů, kdy byla teplota centrální jednotky mimo meze <0,50>. Do lokálních proměnných programu tedy přibyde proměnná pro počet chybových stavů *errCounter* typu *USINT*. Další přidaná proměnná se jmenuje *tmp*, je instancí funkčního bloku *R_TRIG* a slouží pro vyhodnocení změny signálů *heat* a *cool*. Doplněné části mají opět světle tyrkysový podklad.

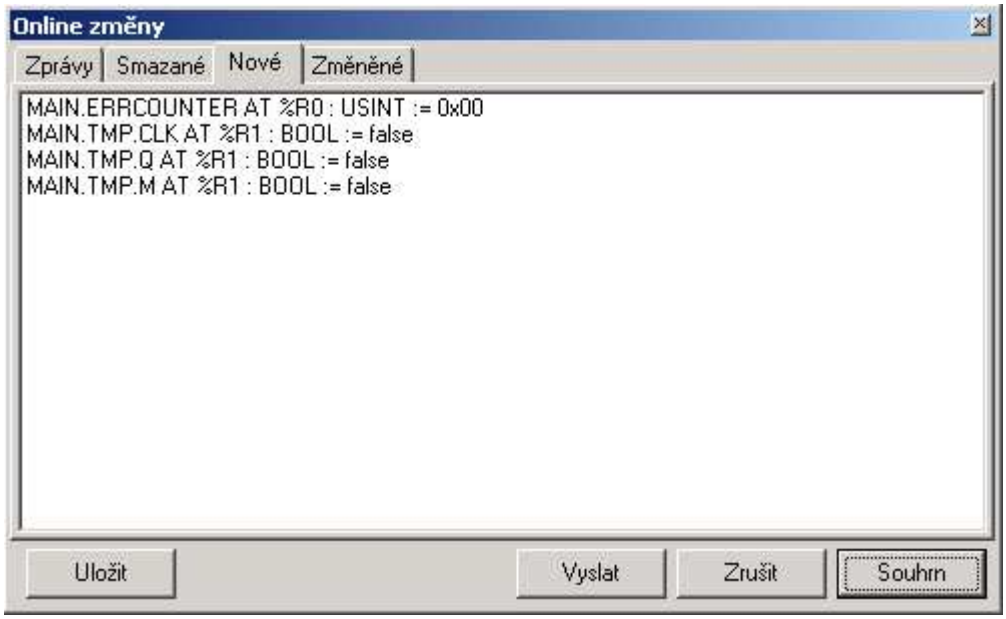
```
VAR_GLOBAL
  temp_CP7002     AT %S36       : SINT;    // CPU temperature
  cool            AT %Y0.0      : BOOL;    // cooling
  heat            AT %Y0.1      : BOOL;    // heating
END_VAR
```

```
PROGRAM Prog1
  VAR
    errCounter   : USINT;
    tmp          : R_TRIG;
  END_VAR

  CASE temp_CP7002 OF
    0..50   : heat := false; cool := false;
    51..127 : heat := false; cool := true;
    ELSE
              heat := true;  cool := false;
  END_CASE;

  tmp( CLK := heat OR cool);
  IF tmp.Q THEN  errCounter := errCounter + 1;  END_IF;
END_PROGRAM
```

Před vysláním přeloženého kódu se zobrazí dialog se souhrnem provedených změn.

Z tohoto dialogu je vidět, že tato změna přidala do programu celkem 4 nové proměnné a jedna proměnná z původního programu byla vypuštěna. To může na první pohled vypadat nelogicky, neboť jsme žádnou proměnnou z programu nevypouštěli a přidávali jsme pouze dvě nové proměnné *errCounter* a *tmp*. Vysvětlení najdeme v podrobnostech.

Pokud POU Program nemá deklarovanou žádnou lokální proměnnou, zakládá překladač v prostředí Mosaic automaticky alespoň jednu prázdnou proměnnou s názvem *DUMMY*. V okamžiku, kdy do programu přidáme alespoň jednu vlastní proměnnou, proměnná *DUMMY* ztrácí smysl a překladač ji vypustí. To vysvětluje záhadu vypuštěné proměnné. Nesoulad v počtu nových proměnných nám také objasní okno podrobnosti.



Z tohoto okna je vidět, že proměnné *errCounter* opravdu odpovídá jedna nová proměnná zatímco pod proměnnou *tmp* se ve skutečnosti skrývají proměnné tři, neboť jde o instanci funkčního bloku typu *R_TRIG*. Každá instance tohoto funkčního bloku pak bude obsahovat proměnné *CLK*, *Q* a *M* typu BOOL.

Dále je v tomto okně uvedeno, jakou inicializační hodnotou byly nové proměnné po svém založení naplněny. Obecně platí, že nové proměnné jsou naplněny inicializační hodnotou, kterou uvede programátor v deklaraci proměnné. Pokud není inicializace v deklaraci proměnné uvedena, pak se proměnná naplní implicitní inicializační hodnotou pro příslušný datový typ.

Nyní si představme následující situaci. Proměnná *errCounter* je typu USINT a maximální hodnota této proměnné může být 255. Dejme tomu, že je to příliš málo pro tento případ. Změníme tedy datový typ proměnné *errCounter* z typu USINT na typ UDINT. Pro lepší orientaci je v tomto textu změněný řádek vyznačen fialově. Zbytek programu se nezměnil.

```
VAR_GLOBAL
  temp_CP7002     AT %S36       : SINT;    // CPU temperature
  cool            AT %Y0.0      : BOOL;    // cooling
  heat            AT %Y0.1      : BOOL;    // heating
END_VAR

PROGRAM Prog1
  VAR
    errCounter   : UDINT;
    tmp          : R_TRIG;
  END_VAR
```

```
  CASE temp_CP7002 OF
    0..50   : heat := false; cool := false;
    51..127 : heat := false; cool := true;
    ELSE
              heat := true;  cool := false;
  END_CASE;

  tmp( CLK := heat OR cool);
  IF tmp.Q THEN  errCounter := errCounter + 1;  END_IF;
END_PROGRAM
```

Dále předpokládejme, že před provedením změny má proměnná *errCounter* nenulovou hodnotu. Úkolem on-line změny je samozřejmě zachovat všechny aktuální hodnoty proměnných v programu a to jak v případech změny datového typu proměnné, tak v případě změny umístění proměnné v paměti PLC. Oba případy při této změně nastaly. Proměnná *errCounter* změnila datový typ a proměnná *tmp* změnila umístění v paměti.

Celkovou informaci podává opět dialog souhrn změn.

Žádné proměnné nepřibyly ani neubyly, leč u všech čtyřech došlo ke změnám. V podrobnostech jsou popsány změny datového typu, změny umístění proměnných v paměti se nevypisují.

Popsané změny zařídí centrální jednotka PLC ve spolupráci s prostředím Mosaic automaticky po vyslání kódu programu z Mosaicu do PLC. Proměnné si prostě ponechají hodnoty nezávisle na tom, jestli se měnil jejich datový typ nebo umístění v paměti.

Stejným způsobem, jako jsme postupovali u změn lokálních proměnných (VAR ... END_VAR), můžeme postupovat i proměnných globálních (VAR_GLOBAL ... END_VAR resp. VAR_GLOBAL RETAIN ... END_VAR). Lze tedy přidávat proměnné, ubírat proměnné a měnit datový typ proměnných

Změny lokálních proměnných na proměnné globální

Příkladem tohoto typu změny může být požadavek na změnu lokální proměnné *errCounter* z našeho příkladu na proměnnou globální např. z důvodu zálohování hodnoty této proměnné při výpadku napájení. Zdá se, že bude stačit pouze změna původní deklarace lokální proměnné

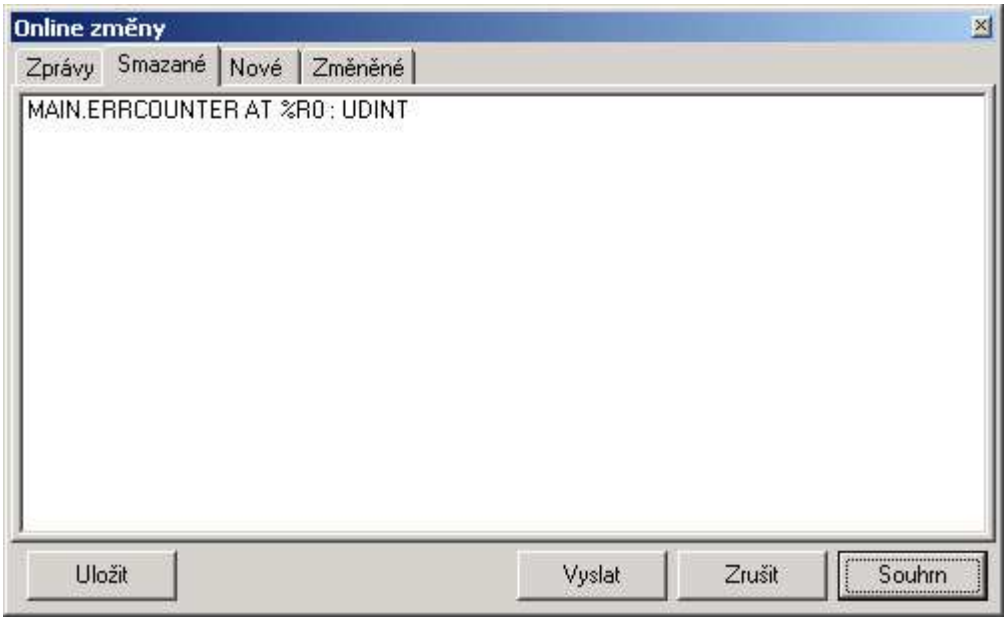
```
PROGRAM Prog1
  VAR
    errCounter  : UDINT; ;   // local variable
    tmp         : R_TRIG;
  END_VAR
```

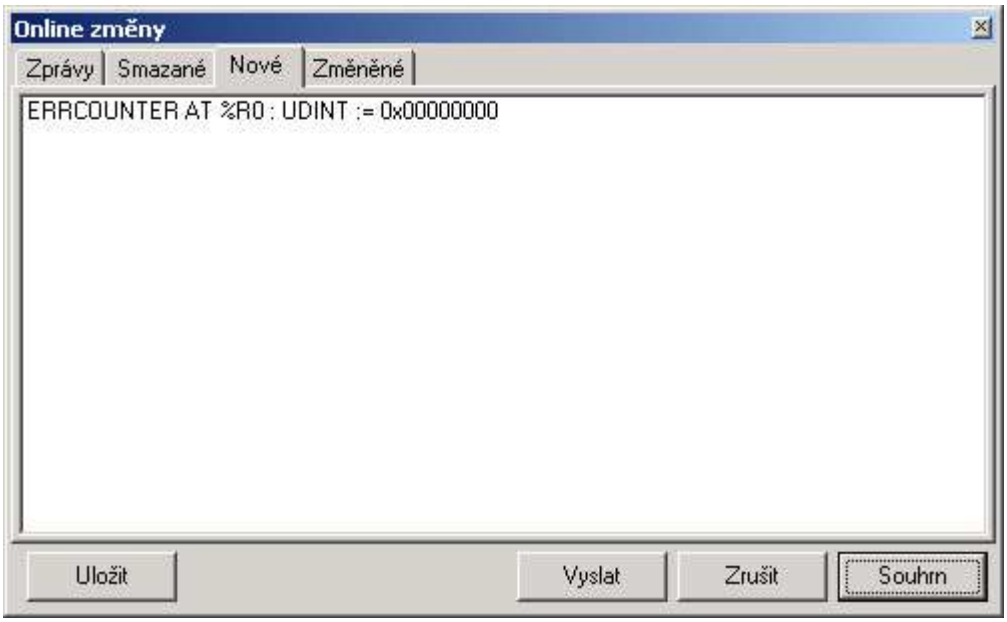
na novou deklaraci proměnné globální

```
VAR_GLOBAL RETAIN
  errCounter  : UDINT;     // global variable
END_VAR

PROGRAM Prog1
  VAR
    tmp         : R_TRIG;
  END_VAR
```

V tomto případě zdání bohužel klame. Varováním budiž souhrnný dialog o provedených změnách.

Zde vidíme, že ačkoliv jsme neměnili počet proměnných ani jejich pojmenování bude jedna proměnná z programu smazaná (vypuštěná) a jedna nová proměnná přibude. Důvod pro toto chování bude opět zřejmý z podrobných informací o on-line změnách.





Lokální proměnná *errCounter* a globální proměnná *errCounter* se ve skutečnosti nejmenují stejně. Plné jméno jakékoliv lokální proměnné se totiž skládá ze jména instance a jména proměnné. V našem případě se tedy lokální proměnná *errCounter* ve skutečnosti jmenuje *main.errCounter*, kde main je jméno instance programu. A protože se jména neshodují, zachází se s nimi při on-line změně jako se dvěmi různými proměnnými. Takže lokální proměnná *errCounter* bude vynechaná a zavede se nová globální proměnná. Důsledkem toho bude ztráta hodnoty *errCounter*, protože nově zavedená globální proměnná bude nainicializovaná hodnotou 0.

Jak tedy postupovat při změně lokální proměnné na proměnnou globální.Tuto situaci je nutné řešit ve dvou krocích. V prvním kroku pouze přidáme novou globální proměnnou *re-*

tainErrCounter. Z lokální proměnné *errCounter* uděláme proměnnou výstupní a její hodnotu přiřadíme do nově založené globální proměnné. Upravený program bude vypadat následovně.

```
VAR_GLOBAL
  temp_CP7002    AT %S36      : SINT;   // CPU temperature
  cool           AT %Y0.0     : BOOL;   // cooling
  heat           AT %Y0.1     : BOOL;   // heating
END_VAR

VAR_GLOBAL RETAIN
  retainErrCounter    : UDINT;
END_VAR

PROGRAM Prog1
  VAR
    tmp         : R_TRIG;
  END_VAR
  VAR_OUTPUT
    errCounter  : UDINT;
  END_VAR

  CASE temp_CP7002 OF
    0..50   : heat := false; cool := false;
    51..127 : heat := false; cool := true;
    ELSE
              heat := true;  cool := false;
  END_CASE;

  tmp( CLK := heat OR cool);
  IF tmp.Q THEN  errCounter := errCounter + 1;  END_IF;
END_PROGRAM

CONFIGURATION ExampleOnLineChange
  RESOURCE CPM
    TASK FreeWheeling(Number := 0);
    PROGRAM main WITH FreeWheeling : Prog1 (errCounter => retainErrCounter);
  END_RESOURCE
END_CONFIGURATION
```

Provedeme-li on-line změnu, výstupní proměnná *errCounter* si podrží hodnotu původní lokální proměnné *errCounter*, protože jak už víme, jejich skutečná jména jsou stejná a sice *main.errCounter*. Hodnota výstupní proměnné proměnné *errCounter* se v každém cyklu kopíruje do nové globální proměnné *retainErrCounter*.

Druhým krokem je vynechání výstupní proměnné *errCounter* a změna programu, kde musíme zpracování výstupní proměnné *errCounter* nahradit zpracováním globální proměnné *retainErrCounter*. Vynechat musíme také přiřazení výstupní proměnné *errCounter*, protože ta už neexistuje. Program bude vypadat následovně.

```
VAR_GLOBAL
  temp_CP7002    AT %S36      : SINT;   // CPU temperature
  cool           AT %Y0.0     : BOOL;   // cooling
  heat           AT %Y0.1     : BOOL;   // heating
END_VAR

VAR_GLOBAL RETAIN
  retainErrCounter    : UDINT;
```

```
END_VAR

PROGRAM Prog1
  VAR
    tmp          : R_TRIG;
  END_VAR

  CASE temp_CP7002 OF
    0..50   : heat := false; cool := false;
    51..127 : heat := false; cool := true;
    ELSE
              heat := true;  cool := false;
  END_CASE;

  tmp( CLK := heat OR cool);
  IF tmp.Q THEN  retainErrCounter := retainErrCounter + 1;  END_IF;
END_PROGRAM


CONFIGURATION ExampleOnLineChange
  RESOURCE CPM
    TASK FreeWheeling(Number := 0);
    PROGRAM main WITH FreeWheeling : Prog1 ();
  END_RESOURCE
END_CONFIGURATION
```

Odměnou za tento složitější postup je korektně provedená on-line změna programu.

Rizika při on-line změnách v jazyce ST

Přejmenování proměnné

Při práci s proměnnými je třeba mít na paměti, že překladač rozeznává on-line změny proměnných podle jmen proměnných a jejich datových typů. To platí i pro opravu jména proměnné. Překladač neposuzuje gramatickou správnost jména proměnné, pouze porovnává shodnost jmen proměnných mezi starým a novým programem.

```
VAR_GLOBAL
  temp_CP7002    : SINT;   // CPU temperature
END_VAR
```

3. On-line změny *.mos programu

On-line změny je možné používat i pro programy, které jsou napsány v tradičních instrukcích pro PLC Tecomat. Těmto změnám je věnovaná následující kapitola.

On-line změny programu v assembleru

Pro další výklad zvolíme stejný příklad jako pro popis změn v jazyce ST. Předpokládejme, že program v PLC zapíná a vypíná chlazení podle údaje o teplotě centrální jednotky. Proměnná *temp_CP7002* udává teplotu ve stupních Celsia (systémový registr S36). Výstup *cool* je zapnutý pokud teplota překročí 50 stupňů. Program bude vypadat následovně.

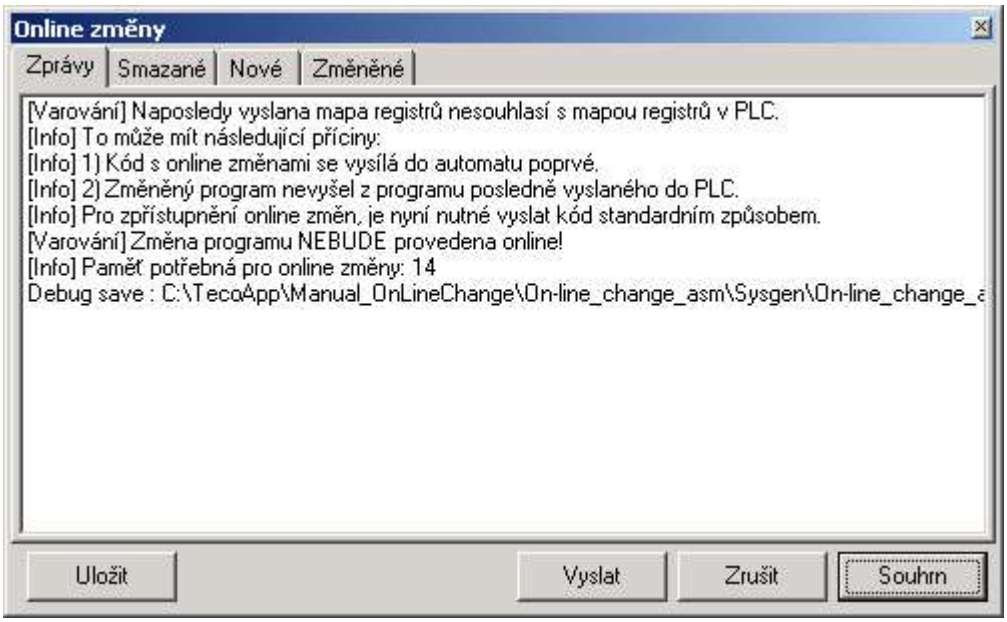
```
#def    temp_CP7002    %S36     ; CPU temperature
#def    cool           %Y0.0    ; cooling
;
P 0
        LD      temp_CP7002
        EXTB
        GTS     -1
        LD      temp_CP7002
        EXTB
        LTS     51
        OR
        NEG
        RES     cool
        ;
        LD      temp_CP7002
        EXTB
        GTS     50
        SET     cool
E 0
```

Zapneme on-line změny, přeložíme program (F9) a vyšleme jej do centrální jednotky (CTRL+F9). Předtím, než se kód programu začne vysílat, objeví se dialog s informacemi o on-line změnách.

Pokud zvolíme „Vyslat“ kód programu bude vyslán standardním způsobem, protože se jedná o nový program. To znamená, že centrální jednotka přejde do režimu HALT (kde jsou typicky zablokované výstupy), nahraje se nový program, provede se restart a poté centrální jednotka přejde do režimu RUN.

Pokud zvolíme „Zrušit“ kód programu se nevyšle a centrální jednotka zůstane beze změny s původním programem a v původním režimu.

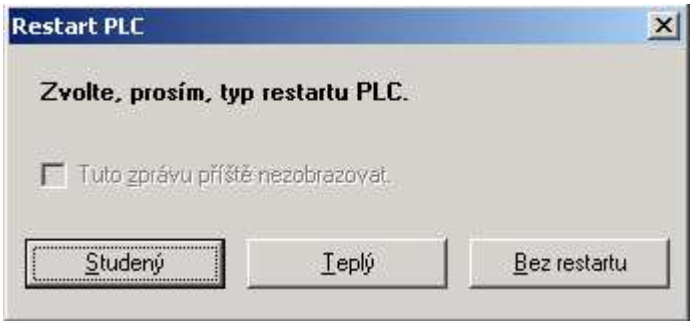
Volba „Podrobnosti“ umožňuje zobrazit doplňkové informace.



Volby „Vyslat“ a „Zrušit“ mají stejný význam, jako v předchozím dialogu. Volba „Souhrn“ provede návrat k předchozímu dialogu. Volba „Uložit“ uloží veškeré informace o on-line změně do textového souboru.

Záložka „Zprávy“ obsahuje souhrnnou informaci o on-line změně. Záložky „Smazané“, „Nové“ a „Změněné“ nesou informace o změnách proměnných a našem případě jsou prázdné, protože zahajujeme on-line změnu (nahráváme nový program).

Vyšleme tedy nový kód do PLC volbou „Vyslat“. Poté se zobrazí dialog s volbou restartu.



Po volbě restartu přejde centrální jednotka do režimu RUN. Od této chvíle je možné upravovat program bez zastavení řízení (tedy on-line).

Změny v kódu programu

Předpokládejme, že předcházející program je potřeba doplnit o ovládání topení. Pokud bude teplota měřená centrální jednotkou meší než 0 stupňů Celsia, zapneme výstup pro ovládání topení.

První věc, kterou si musíme uvědomit, je stav oken editoru v prostředí Mosaic. Po nahrání nového programu a přechodu do režimu RUN jsou všechna okna editoru ve stavu DEBUG, což znamená, že není možno editovat text. Tento stav signalizuje ikona  v levém dolním rohu editorového okna. Přepnutí do stavu EDIT lze provést klinutím levého tlačítka myši na této ikoně nebo horkou klávesou ALT+F6. Stav EDIT je signalizován ikonou  .

Nyní můžeme upravit program. Přidáme definici pro výstup topení *heat* a do programu doplníme jeho ovládání. Doplněné části mají světle tyrkysový podklad.

```
#def    temp_CP7002     %S36     ; CPU temperature
#def    cool            %Y0.0    ; cooling
#def    heat            %Y0.1    ; heating
;
P 0
        LD      temp_CP7002
        EXTB
        GTS     -1
        LD      temp_CP7002
        EXTB
        LTS     51
        OR
        NEG
        RES     cool
        RES     heat
        ;
        LD      temp_CP7002
        EXTB
        GTS     50
        SET     cool
        RES     heat
        ;
        LD      temp_CP7002
        EXTB
        LTS     0
        RES     cool
        SET     heat
E 0
```

Dále je postup standardní: přeložit upravený program a vyslat kód do PLC. A protože máme zapnuté on-line změny a v centrální jednotce existuje předchozí verze našeho programu, tak nový program bude akceptován centrální jednotkou bez zastavení řízení. Před vysláním programu do PLC se opět zobrazí dialog s informacemi o provedených on-line změnách. Tentokrát bude vypadat následovně.

Dialog nás informuje o tom, že změny byly provedeny pouze v kódu programu. Tuto skutečnost lze ověřit v podrobnostech.

Změny v proměnných programu

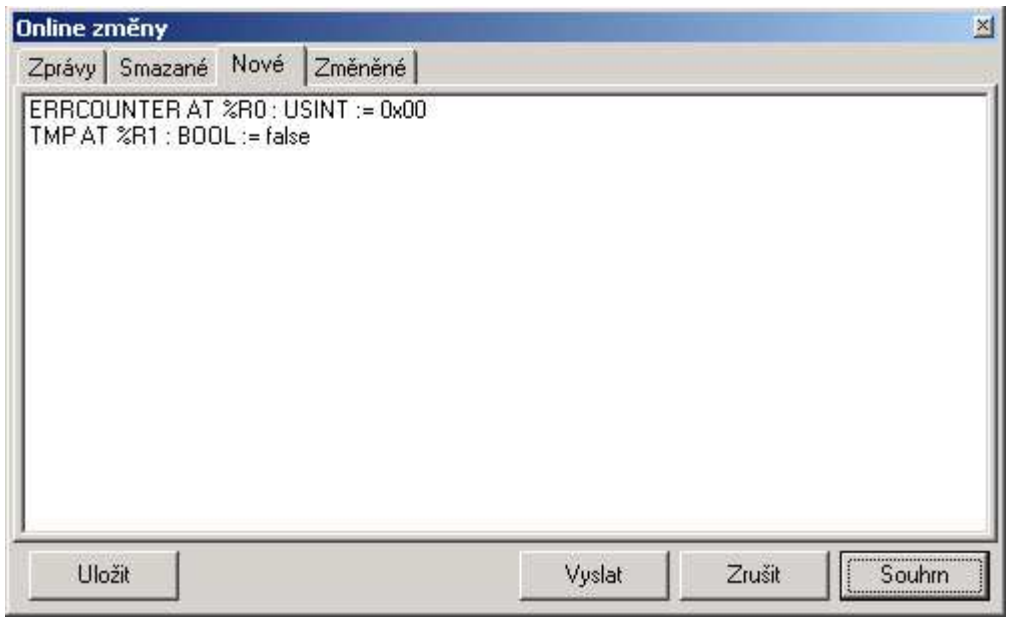
Pro ukázku této vlastnosti doplníme program z předchozího odstavce o proměnnou, která bude evidovat počet případů, kdy byla teplota centrální jednotky mimo meze <0,50>. V programu tedy přibyde nová proměnná pro počet chybových stavů *errCounter* typu *USINT*. Další přidaná proměnná se jmenuje *tmp a* je to pomocná proměnná pro instrukci *LET*, která slouží pro vyhodnocení změny signálů *heat* a *cool*. Doplněné části mají opět světle tyrkysový podklad.

```
#def    temp_CP7002   %S36     ; CPU temperature
#def    cool          %Y0.0    ; cooling
#def    heat          %Y0.1    ; heating
;
#reg    USINT         errCounter
#reg    BOOL          tmp
;
P 0
```

```
        LD      temp_CP7002
        EXTB
        GTS     -1
        LD      temp_CP7002
        EXTB
        LTS     51
        OR
        NEG
        RES     cool
        RES     heat
        ;
        LD      temp_CP7002
        EXTB
        GTS     50
        SET     cool
        RES     heat
        ;
        LD      temp_CP7002
        EXTB
        LTS     0
        RES     cool
        SET     heat
        ;
        LD      heat
        OR      cool
        LET     tmp
        AND     1
        LD      errCounter
        ADD
        WR      errCounter
E 0
```

Před vysláním přeloženého kódu se zobrazí dialog se souhrnem provedených změn.

Z tohoto dialogu je vidět, že tato změna přidala do programu celkem 2 nové proměnné. To jsou naše proměnné *errCounter* a *tmp*. Přesvědčit se můžeme v podrobnostech.



V tomto okně je uvedeno, jaké proměnné byly nově založeny a jakou inicializační hodnotou byly nové proměnné po svém založení naplněny. Proměnné založené pomocí direktivy *#reg* mají inicializační hodnotu 0.

Nyní si představme následující situaci. Proměnná *errCounter* je typu USINT a maximální hodnota této proměnné může být 255. Dejme tomu, že je to příliš málo pro tento případ. Změníme tedy datový typ proměnné *errCounter* z typu USINT (proměnná velikosti jeden byte, bez znaménka) na typ UDINT (proměnná velikosti čtyři byty, bez znaménka). Pro lepší orientaci je v tomto textu změněný řádek vyznačen fialově. Zbytek programu se nezměnil.

```
#def    temp_CP7002   %S36     ; CPU temperature
#def    cool          %Y0.0    ; cooling
#def    heat          %Y0.1    ; heating
;
#reg    UDINT         errCounter
#reg    BOOL          tmp
;
P 0
        LD      temp_CP7002
        ...
```

Dále předpokládejme, že před provedením změny má proměnná *errCounter* nenulovou hodnotu. Úkolem on-line změny je samozřejmě zachovat všechny aktuální hodnoty proměnných v programu a to jak v případech změny datového typu proměnné, tak v případě změny umístění proměnné v paměti PLC. Oba případy při této změně nastaly. Proměnná *errCounter* změnila datový typ a proměnná *tmp* změnila umístění v paměti.

Celkovou informaci podává opět dialog souhrn změn.

Žádné proměnné nepřibyly ani neubyly, leč u všech došlo ke změnám. V podrobnostech jsou popsány změny datového typu, změny umístění proměnných v paměti se nevypisují, pouze je evidován jejich počet.

Popsané změny zařídí centrální jednotka PLC ve spolupráci s prostředím Mosaic automaticky po vyslání kódu programu z Mosaicu do PLC. Proměnné si prostě ponechají hodnoty nezávisle na tom, jestli se měnil jejich datový typ nebo umístění v paměti.

Rizika při on-line změnách v assembleru

Direktiva #def

Překladač „neví nic“ o direktivách *#def* v programu. Tuto direktivu chápe překladač jako makro pro textovou náhradu. Co to znamená pro on-line změny v praxi ukáže následující příklad.

```
#def    SQ7     %X0.0    ; switch

P 0
        LD      SQ7
E 0
```

Ve výše uvedeném příkladu překladač nahradí při překladu programu všechny výskyty slova *SQ7* řetězcem *%X0.0*. Nic dalšího se nestane. To v praxi znamená, že změna direktivy *#def* není nijak evidovaná při on-line změně. Za kontinuitu obsahu proměnných, které jsou definovány pomocí direktivity *#def* tedy odpovídá programátor.

Přístup na absolutní adresy proměnných

Přístup na absolutní adresy proměnných v uživatelském programu rovněž není evidován v on-line změnách.

```
; old program
P 0
        LD      %R100    ; error_counter
E 0
```

Změníme-li výše uvedený program při on-line změně, změní se pouze kód programu.

```
; new program
P 0
        LD      %R200    ; error_counter
E 0
```

Proměnné %R100 a %R200 nejsou při on-line změně nijak ovlivněny, což znamená, že nový program může (a pravděpodobně také bude) pokračovat ve výpočtu s jinou hodnotou *error_counter* než program před provedením změny.

Pokud potřebujeme z nějakého důvodu přistoupit k proměnné na konkrétní adrese v paměti (např. z důvody vazby na vizualizační program) a chceme si zachovat výhody on-line změn, můžeme použít následující postup.

```
; old program
#reg byte   100,error_counter     ; == %R100
P 0
        LD      error_counter
E 0
```

Změníme-li výše uvedený program při on-line změně, proměnná *error_counter* si zachová původní hodnotu i po on-line změně programu.

```
; new program
#reg byte   200,error_counter     ; == %R200
P 0
        LD      error_counter
E 0
```

Proměnná tedy musí být deklarovaná pomocí direktivy *#reg*.

Časovače, čítače, posuvné registry

V instrukčním souboru systémů Tecomat jsou instrukce pro časovače, čítače a posuvné registry, které pro svoji funkci používají vnitřní pomocnou proměnnou. Ta je použita jako paměť náběžných hran čítaných vstupů apod. a není viditelná z uživatelského programu. Pro správnou funkci je však nezbytná. Vnitřní pomocnou proměnnou používají následující instrukce:

- časovače TON, TOF, RTO, IMP
- čítače CTU, CTD, CNT
- posuvné registry SFL, SFR

Pokud mají tyto instrukce fungovat správně při on-line změně, musí být jejich operand vyjádřen v programu symbolickým jménem proměnné. V tomto případě prostředí Mosaic ve spolupráci s centrální jednotkou PLC zajistí zachování hodnoty proměnné včetně vnitřní pomocné proměnné.

4. Seznam chyb při on-line změně

70	xx	xxxx	chyby hlášené centrální jednotkou při on-line změně (řízení technologie pokračuje s původním programem)
70	05	0000	chybná délka mapy nového uživatelského programu
70	06	0000	chybný zabezpečovací znak (CRC) mapy nového uživatelského programu
70	07	0000	chybný zabezpečovací znak (CRC) celého nového programu
70	09	0000	nový program je přeložen pro jinou řadu centrálních jednotek
70	24	0000	chybí seznam on-line změn
70	25	0000	seznam on-line změn má chybné CRC
70	31	rr pp	v definici I/Omodulu chybí inicializační tabulka
70	43	rr pp	v definici I/Omodulu překročeno max. číslo rámu

kde rr ... číslo rámu a pp ... pozice I/O modulu v rámu