

22

22. TVORBA UŽIVATELSKÝCH INSTRUKCÍ A MAKER

Od verze překladače PLC 6.041 je umožněno si definovat a používat vlastní instrukce pro překlad PLC programu. Rozvoj uživatelských instrukcí může být definován jak na úrovni jazyka TECHNOL, tak na úrovni assembleru 386 a vyšším. Uživatelské instrukce mohou přebírat formální parametry a mohou si definovat vlastní lokální proměnné a návěští.

Uživatelské instrukce mohou být definovány v samostatném souboru, který se připojuje ke zdrojovému textu v době překladu.

22.1 Připojování externích definičních souborů

Uživatelské instrukce, symbolické identifikátory chyb a informačních hlášení (viz. Nastavování chyb – kapitola 14.) nebo různá makra, mohou být definovány v samostatných souborech, které se připojují ke zdrojovému textu v době překladu pomocí instrukce `T_INCLUDE`.

instrukce	<code>T_INCLUDE</code>	
funkce	<code>T_INCLUDE</code>	připojení definičního souboru
syntax	<code>T_INCLUDE</code>	<code>file</code>
parametr	<code>"file"</code>	název souboru

Připojení definičního souboru ke zdrojovému textu. Parametr `"file"` je název souboru, který může obsahovat absolutní cestu. Pokud název žádnou cestu neobsahuje, bude se hledat ve stejném adresáři, kde se nachází zdrojový PLC program.

Pokud je uveden název souboru v apostrofech (`'`) a neobsahuje absolutní cestu, předpokládá se umístění v systémovém adresáři SYSTEM.

Je zvykem umísťovat instrukce `T_INCLUDE` hned na začátek zdrojového programu a v názvech pro definiční soubory používat příponu `INC` .

Definiční soubory mohou obsahovat definice symbolických konstant (chyb), definice maker a uživatelských

instrukcí a nesmějí obsahovat přímý výkonný instrukční kód (kromě kódu definovaného v makrech).

Příklad:

```
T_INCLUDE    VXR50.INC           ;Definiční soubor VXR50.INC se bude hledat
                                   ;v adresáři, kde se nachází zdrojový PLC
                                   ;program (VXR50.PLC) .

T_INCLUDE    VXR50\VXR50.INC     ;Umístění definičního souboru VXR50.INC
                                   ;v podadresáři VXR50 adresáře, kde je
                                   ;umístěn zdrojový PLC program.
```

22.2 Definice uživatelských instrukcí a maker

Rozvoj uživatelských instrukcí může být definován jak na úrovni jazyka TECHNOLOG, tak na úrovni assembleru. Uživatelské instrukce mohou přebírat formální parametry a mohou si definovat vlastní lokální proměnné a návěští.

instrukce	DEF_T_MACRO	
funkce	DEF_T_MACRO	začátek definice instrukce (makra)
syntax	DEF_T_MACRO	name [par1, par2, ...]
1.parametr	"name"	jméno makra
2.parametr	"par1"	formální parametry

Instrukce **DEF_T_MACRO** označuje začátek definice uživatelského makra, nebo uživatelské instrukce.

První parametr „**name**“ je povinný a udává název makra nebo instrukce. Pod tímto názvem se potom makro nebo instrukce volá pro její vykonání, přičemž se automaticky provede rozvoj makra podle definice.

Další parametry jsou formální parametry makra nebo instrukce a jejich počet závisí od konkrétní implementace. Formální parametry slouží pro předávání skutečných proměnných do rozvoje makra nebo instrukce při jejím výkonu a mohou to být například konstanty, bitové proměnné a různé datové proměnné.

Volání uživatelských maker a instrukcí se provede prostým voláním podle názvu makra a výčtem skutečných parametrů:

```
name par1, par2
```

Definice maker mohou být do sebe vnořovány, takže z těla jednoho makra možno volat jiné makro.

instrukce	END_T_MACRO	
funkce	END_T_MACRO	konec definice instrukce (makra)
syntax	END_T_MACRO	[modif]

Instrukce **END_T_MACRO** označuje konec definice uživatelského makra, nebo uživatelské instrukce.

Instrukce nemusí mít žádný parametr. Pokud instrukce má parametry, jedná se o seznam řídicích příznaků, které slouží pro dodatečné upřesnění uživatelské instrukce. Příznaky upřesňují „debugovatelnost“, práci se zásobníkem při závorkových operacích a konverzi pro předání parametrů. Popis jednotlivých příznaků bude uveden dále u instrukci (APPEND_T_MACRO) v části „Řízení uživatelských instrukcí“.

Poznámka:

Často se definice maker nezaobejde bez použití instrukcí assembleru, které se budou kombinovat se standardními instrukcemi v TECHNOLu. V tomto případě je nutné znát několik pravidel. Fyzická reprezentace bitu v RLO registru je bit s váhou 40h v AH registru mikroprocesoru. Datový registr odpovídá registru ECX. Nedoporučuje se používat SI s ESI registr, protože se nezachová jeho obsah ve standardních instrukcích TECHNOL. Lepší je nepočítat se zachováním obsahů registrů, když jsou mezi naše instrukce vkládány standardní instrukce TECHNOL.

Příklad:

```

DEF_T_MACRO      ERRNUM
                  EQUI      ERR_VR1,    4512h      ; chyba 1.12.45
                  EQUI      ERR_VR2,    4612h      ; chyba 1.12.46

                  EQUI      DD123,      123

END_T_MACRO

```

22.3 Formální parametry a lokální symboly maker

Makro obsahuje při své definici formální parametry. Formální parametry slouží pro předávání skutečných proměnných do rozvoje makra nebo instrukce při jejím výkonu (rozvoji makra).

Kromě formálních parametrů, může makro běžně používat všechny globální a lokální proměnné, které jsou v okamžiku výkonu makra k dispozici.

Když je potřeba při definici makra použít některý z formálních parametrů pro instrukce TECHNOL, je nutné použít před názvem formálního parametru prefix: “.TMAC “. Tento prefix způsobí, že instrukce TECHNOLu přebere formální parametr tak, aby došlo ke správné náhradě skutečného parametru v okamžiku výkonu makra s ohledem na její název a typ. Prefix “.TMAC “ se doporučuje psát jako první před případnými dalšími prefixy.

Příklad:

Příklady použití formálních parametrů:

```

DEF_T_MACRO      POKUS      PAR1, PAR2, PAR3
                  LOD        TMAC.PAR3      ;načte PAR3 podle jeho typu
                  LDR        TMAC.PAR1      ;načte bit PAR1
                  LO         -TMAC.PAR2     ;log. OR s negací bitu PAR2
                  WR         TMAC.PAR3.PAR1 ;zápis bitu na adresu PAR3
                                      ;s váhou PAR1
                                      ;složitější adresace bitu)
                  LDR        ALFA          ;načte globální bit ALFA
                  LO         TMAC.PAR2     ;log. OR s bitem PAR2
                  STO1       TMAC.BYTE.PAR3 ;podmíněný zápis do PAR3
                                      ;typ je změněn prefixem BYTE

END_T_MACRO

;Volání makra:
;ALFA a BETA jsou bitové proměnné a BUNX je datová proměnná

POKUS      ALFA, BETA, BUNX      ;Volání uživatelského makra

```

Makro může ve svém rozvoji definovat vlastní návěští a vlastní data. Když by makro potom bylo v programu použito vícekrát, došlo by ke chybě překladu následkem vícenásobné definice symbolů. Pro odstranění tohoto problému slouží instrukce **T_LOCAL**.

instrukce	T_LOCAL
-----------	----------------

funkce	T_LOCAL	definice lokálních symbolů makra
syntax	T_LOCAL	sym1, [sym2, sym3, ...]

Instrukce **T_LOCAL** musí být umístěná bezprostředně za instrukcí pro začátek definice makra **DEF_T_MACRO** a může být použita vícekrát. Instrukce **T_LOCAL** se používá pro specifikování lokálních symbolů v rámci makra. Lokálními symboly mohou být návěští, datové a bitové proměnné, které jsou použity jen v rozvoji makra. Instrukci je nutno použít vždy, kdy takové symboly jsou v rámci makra definovány a kdy se předpokládá vícenásobné použití makra (instrukce) ve zdrojovém kódu.

Bitové a datové proměnné deklarované v makru musí mít lokální charakter a proto se musí definovat v modulu **DATA_LOCAL**, včleněném přímo v makru (viz. Popis modulů – Kapitola 5., Struktura PLC programu). Pro definici datových proměnných možno použít instrukci **DS** a pro definici bitových proměnných možno použít instrukci **DFM**. Jediná výjimka je, že v instrukci **DFM** musí být povinně definováno všech osm bitů.

Příklad:

Definice lokálních dat

```
DEF_T_MACRO POKUS3      PAR1, PAR2, PAR3
    T_LOCAL      BUN_M1, BUN_M2, BUN_BIT      ;lokální symboly makra
    T_LOCAL      BIT0,BIT1,BIT2,BIT3,BIT4,BIT5,BIT6,BIT7

    DATA_LOCAL
        BUN_M1:      DS      1      ;lokální bajtová proměnná
        BUN_M2:      DS      2      ;lokální wordová proměnná
        BUN_BIT:     DFM BIT0,BIT1,BIT2,BIT3,BIT4,BIT5,BIT6,BIT7
    DATA_LOCAL_END
```

Příklad:

První a druhý parametr makra jsou bitové proměnné a třetí parametr je datová proměnná typu WORD

```
DEF_T_MACRO POKUS4      PAR1, PAR2, PAR3
    T_LOCAL      NAVM      ;lokální návěští
    T_LOCAL      BIT0,BIT1,BIT2,BIT3,BIT4,BIT5,BIT6,BIT7 ;lokální symboly

    DATA_LOCAL
        DFM BIT0,BIT1,BIT2,BIT3,BIT4,BIT5,BIT6,BIT7 ;lokální bity
    DATA_LOCAL_END

    LDR          -TMAC.PAR1      ;čtení negace formálního bitu PAR1
    LO           TMAC.PAR2      ;log. OR s formálním bitem PAR2
    LA           -ALFA          ;log. AND s globálním bitem ALFA
    WR           BIT0           ;zápis do lokálního bitu makra BIT0
    JL0          NAVM           ;podmíněný skok
    LDR          TMAC.PAR2      ;čtení formálního bitu PAR3
    FL1          1,BIT1        ;podmíněný zápis do lokálního bitu BIT1
    NAVM:        ;lokální návěští makra
    LOD          TMAC.PAR3      ;čtení z formálního parametru (word)
END_T_MACRO
```

;Volání makra:

```
POKUS4      ALFA, BETA, BUNX      ;Volání uživatelského makra
```

22.4 Řízení uživatelských instrukcí

Mezi další možnosti řízení uživatelských instrukcí patří možnost nastavení ladění, konverzí a práce se zásobníkem. Také je umožněno tzv. přetěžování základních instrukcí jazyka TECHNOLOG uživatelskými instrukcemi.

instrukce	APPEND_T_MACRO
-----------	-----------------------

funkce	APPEND_T_MACRO	řízení uživatelské instrukce
syntax	APPEND_T_MACRO	name, alias, [modif]

Instrukce **APPEND_T_MACRO** slouží pro připojení názvu k rezervovaným názvům překladače TECHNOLOG a pro nastavení příznaků. Tato instrukce se samotná používá hlavně pro přetěžování názvů instrukcí a vzhledem k její speciálnějšímu významu se budeme hlavně zabývat seznamem příznaků, které jsou v ní uvedeny. Tento seznam se také používá v parametrech instrukce **END_T_MACRO**, kde je jeho hlavní použití. Příznaky jsou odděleny čárkou.

Přehled nastavování příznaků:

1.parametr		2.parametr		3.parametr	
Vztah k zásobníku log.instrukcí		Konverze vstup.parametrů		Nastavování breakpointů (DEBUG)	
T_NORMAL*	Nemá vztah k zásobníku	C_0*	Bez konverze	D_OFF*	Instrukce nemá povolen breakpoint
T_BEGIN	Vyprázdnění zásobníku	C_1	Změna závorek na řetězců _op, _cl, ...	D_ON	Instrukce má povolen breakpoint
T_END	Koncová instrukce, podobně jako WR.				
T_PUSH	Uložení obsahu RLO do zásobníku, podobně jako LDR.				
T_POP	Vybrání RLO ze zásobníku, jako samotné LO, LA.				

Implicitní nastavení pro instrukce je T_NORMAL, C_0, D_OFF.

Pokud v ukončovací instrukci definice makra **END_T_MACRO** žádné parametry neuvedeme, instrukce nebude mít žádný vztah vzhledem k zásobníku, nebude mít konverzi parametrů a nebude mít povolen breakpoint.

Příklad:

Pro konec definice makra:

Uživatelská instrukce má být typu koncové instrukce (WR, FL1,...) , nemá mít konverzi a je bez ladění:

```
END_T_MACRO      T_END, C_0, D_OFF
```

```
APPEND_T_MACRO   ALFA, BETA, T_END, C_0, D_OFF
```

instrukce	CONTROL_T_MACRO
------------------	------------------------

funkce **CONTROL_T_MACRO** **řízení uživatelských instrukcí**

syntax **CONTROL_T_MACRO** **par**

Instrukce **CONTROL_T_MACRO** slouží pro řízení vykonávání všech uživatelských instrukcí a maker. Instrukce má jeden parametr, kterým je řídicí klíčové slovo. Instrukce může být v programu použita vícekrát.

instrukce	parametr	význam
CONTROL_T_MACRO	POS*	(implicitní nastavení) Uživatelské instrukce se provádí až po rozdekódování standardních instrukcí TECHNO Lu (posprocesor). V tomto případě se nedá použít přetěžování standardních instrukcí.
	PRE	Uživatelské instrukce se provádí před rozdekódováním standardních instrukcí TECHNO Lu (preprocesor). V tomto případě je možno použít přetěžování standardních instrukcí.

