TECHNICKÁ UNIVERZITA V LIBERCI

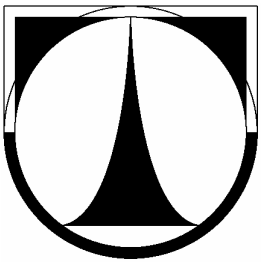
Fakulta strojní

DIPLOMOVÁ PRÁCE

2006 **M a r t i n V Í T E K**

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta strojní



DIPLOMOVÁ PRÁCE

Programovatelný PLC dle IEC 61 131

Martin Vítek
2006

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Katedra aplikované kybernetiky Studijní rok: 2005/06

ZADÁNÍ DIPLOMOVÉ PRÁCE

Jméno a příjmení: **Martin V Í T E K**

Studijní program: **M2301 Strojní inženýrství**

Obor: **3902T021 Automatizované systémy řízení ve strojírenství**

Zaměření: **Automatizace inženýrských prací**

Ve smyslu zákona č. 111/1998 Sb. o vysokých školách se Vám určuje diplomová práce na téma:

Programovatelný PLC dle IEC 61 131

Zásady pro vypracování:

1. Seznamte se s problematikou programování PLC dle IEC 61 131.
2. Prověřte možnosti a způsoby implementace programování PLC s využitím IEC 61 131.
3. Navrhněte hardwarové řešení.
4. Vytvořte ukázkovou aplikaci.
5. Vyhodnoťte výsledky softwarového a hardwarového návrhu PLC programovatelného dle IEC 61 131.

Platnost zadání diplomové práce je 15 měsíců od výše uvedeného data (v uvedené lhůtě je třeba podat přihlášku ke SZZ). Termíny odevzdání diplomové práce jsou určeny pro každý studijní rok a jsou uvedeny v harmonogramu výuky.

Fakulta
Strojní
Technická univerzita v Liberci

Katedra aplikované kybernetiky

Studijní rok 2005/2006

Studijní program: M2301 Strojní inženýrství

Studijní obor: 3902T021 Automatizované systémy řízení ve strojírenství

Zaměření: Automatizace inženýrských prací

Programovatelný PLC dle IEC 61 131

PLC in compliance with IEC 61 131

Jméno autora: Martin Vítek

Vedoucí diplomové práce: Prof. Ing. Miroslav Olehla, CSc.

Rozsah práce:
Počet stran: 60
Počet příloh: 4

Datum odevzdání:
25.5.2006

MÍSTOPŘÍSEŽNÉ PROHLÁŠENÍ

Místopřísežně prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury.

V Liberci dne 25.5.2006 ……

Martin Vítek

PODĚKOVÁNÍ

Na tomto místě bych chtěl poděkovat Prof. Ing. Miroslavu Olehlovi, Csc., který mi umožnil zpracovat toto téma. Dále děkuji rodinným příslušníkům za podporu a trpělivost, kterou projevili.

ANOTACE:

Cílem diplomové práce je navrhnout a odzkoušet na ukázkové aplikaci jednoduchý programovatelný automat. Jeho realizace je v souladu s mezinárodní normou IEC 61131, kterou z větší části převzala i Česká republika. V teoretické části je seznámení s normou IEC 61131, s pojmy a programovacími jazyky definovanými v této normě.

V praktické části je vytvořena ukázková aplikace s použitím hardwarového vývojového prostředí firmy Beck IPC GmbH a softwarového prostředí firmy 3S – Smart Software Solution GmbH.

ANNOTATION:

The objective of this thesis is to design and test a simple PLC by means of an exemplary application. The PLC design corresponds with the international IEC 61131 standard which was partially adopted by the Czech Republic. The theoretical part of the thesis gives information about the IEC 61131 standard, basic terminology and programming languages defined in this standard.

The practical part contains an exemplary application created with the use of the Beck IPC GmbH hardware development environment and the 3S – Smart Software Solution GmbH software environment.

SEZNAM POUŽITÝCH ZKRATEK

ČSN	Česká Státní Norma
EN	Europen Norm
FB	Function Block
FBD	Function Block Diagram
FUN	Function
HTML	Hyper Text Transfer Protocol
I2C	Inter Integrated Circuit
IL	Instruction List
LD	Ladder Diagram
IEC	International Elektrotechnical Commission
PLC	Programmable Logic Controller
POU	Program Organization Unit
SFC	Sequential Function Chart
ST	Structured Text

OBSAH

1 Úvod

V dnešní době je automatizace stále častější ve všech technologických procesech i jiných lidských činnostech. Automatizace prošla velkým vývojem od jednoúčelových mechanických strojů až po univerzální elektronická zařízení. S masovým nasazením univerzálních automatizačních prvků se snižuje cena těchto zařízení, díky tomu jsou dostupná i v těch nejjednodušších strojích ve všech možných oborech.
Standardizace v této oblasti přinesla ještě více masovější nasazení těchto technických zařízení a umožnila realizace projektů v co možná nejkratším čase. Další přínos standardu je v možnosti využití zařízení různých výrobců bez nutnosti změny uživatelského programu nebo jen v minimální míře.

Pro realizaci ukázkové aplikace byl zvolen vývojový kit DK50 od německé firmy BECK GmbH a softwarové prostředí CoDeSys od německé firmy 3S-Smart Software Solution. K tomuto kitu byly připojeny další periferie a to tlačítka, display, větráček, čidlo teploty v podobě integrovaného obvodu komunikujícího přes sběrnici I2C a rezistory jako zdroj tepla. Pomocí těchto prostředků byl navržen a odzkoušen jednoduchý programovatelný automat jako regulátor teploty. Pomocí webového rozhraní je možné měnit požadovanou hodnotu teploty, sledovat její průběh a je možné sledovat zapínání akčních členů.

2 PLC v automatizaci

Programovatelný automat označovaný zkratkou PLC (Programmable Logic Controller) je uživatelsky programovatelný řídící systém přizpůsobený pro řízení různých průmyslových a technologických procesů nebo zařízení. Původně nahrazovaly programovatelné automaty reléovou logiku. Postupně zastávaly funkci regulační, monitorovací s využitím binárních i analogových vstupů a výstupů. V současné době lze spojovat PLC do rozsáhlých systémů s centrálním řízením umožňující řídit velké technologické procesy.

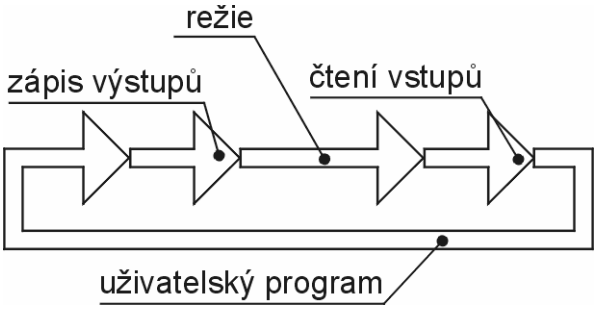
Existují různé systémy od mikro PLC, které mají pár binárních vstupů a výstupů, až po modulární PLC jednotky, umožňující obsloužit stovky vstupů a výstupů jak binárních, tak analogových. V praxi se můžeme setkat s programovatelnými automaty od různých výrobců, např.: H+B, Matsushita, Mitsubishi, Saia, Festo, ABB, Allen-Bradley, Teco.
Jejich konstrukce umožňuje použití i v nejtěžším průmyslovém prostředí na rozdíl od běžných PC.

2.1 Princip funkce PLC

Všechny programovatelné automaty se skládají z CPU (centrální procesorová jednotka), systémové paměti, uživatelské paměti, skupiny vstupně výstupních jednotek a to buď binárních, analogových nebo jejich kombinací, komunikačního rozhraní pro ovládání a programování programovatelného automatu, případně pro spojení jednotlivých automatů mezi sebou a pro připojení na vizualizaci .

Programovatelné automaty snímají pomocí senzorů stavy systému a pomocí akčních členů ovlivňují systém dle řídícího programu. CPU programovatelného automatu vykonává dva programy. Jednak systémový, který je součástí PLC jako tzv. firmware a uživatelský, který řeší úlohu specifikovanou uživatelem. Cyklus běhu programu v PLC je na obrázku 2.1 – 1. V uživatelském software není nutné vracet řízení programu zpět na začátek, o to se většinou stará systém PLC. Doba cyklu programu v PLC má být co nejkratší, vzhledem ke zpracování časově náročnějších úloh. Proto volíme algoritmus uživatelského programu co možná nejefektivnější. Jakékoli cykly zadržující běh programu jsou nežádoucí a mohou

vést k překročení maximální doby cyklu běhu programu a sytém toto může hlásit jako chybu překročení doby cyklu. Režie je doba, kterou potřebuje PLC pro obsloužení systémových registrů, ošetření sériové komunikace RS232 a TCP/IP a obsluhu dalších systémových úloh. Po té následuje načtení vstupních hodnot, které se uloží do registrů a během provádění uživatelského programu se nemění. Po té se provede uživatelský program. Následuje zapsání výstupních hodnot. Je nutné si uvědomit, že jakákoli změna výstupních proměnných během uživatelského programu se reálně neprojeví a hodnoty výstupu jsou přepsány až po poslední instrukci uživatelského programu.



Obr. 2.1 -1 Cyklus PLC

V počátcích každý výrobce automatizační techniky měl svoji vlastní koncepci, v jejímž rámci poskytoval svým zákazníkům vybavení pro jednotlivý typ systému. Zákazník tím byl nucen v rámci zachování kompatibility svého systému pokračovat s rozšiřováním zařízení téhož výrobce.

Bloková struktura programovatelného automatu je znázorněna na obr. 2.1 - 2. Jednotlivé bloky uvedené na obrázku mají následující funkce:

CPU – (centrální procesorová jednotka) zpracovává informace, tj. provádí systémový a uživatelský program na základě vyčtených údajů z periferních jednotek.

I/O jednotka vstupů a výstupů snímá hodnoty vstupních veličin buď binárních nebo analogových, které konvertuje na digitální, ukládá je do paměti a ovládá výstupy buď binární v podobě spínacích tranzistorů nebo relé a analogových výstupů a to buď napěťových nebo proudových.

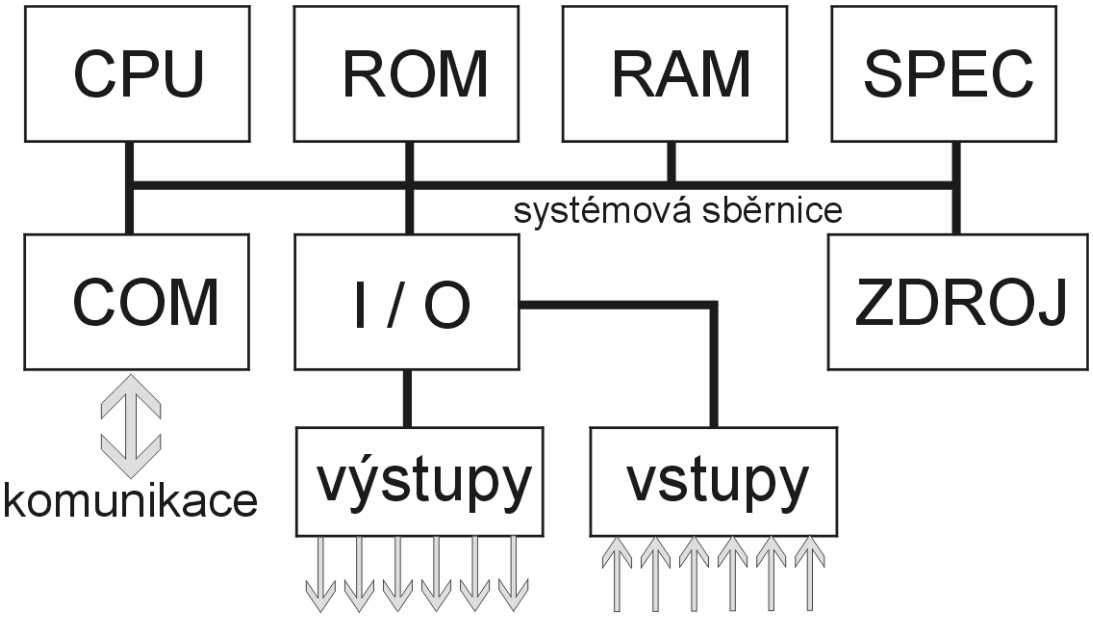
RAM (read access memory) paměť s možností čtení a zápisu, slouží pro proměnné a registry nutné pro chod programu.

ROM (read only memory) paměť pouze pro čtení. V ní je převážně uložen systémový program.

COM – komunikace, zajišťuje komunikaci automatu s okolím.

ZDROJ – zajišťuje napájení automatu.

SPEC – speciální funkce jako např. hodiny reálného času, časovače, čítače, sekvenční registry.



Obr. 2.1 – 2 blokové schéma PLC

2.2 Standardizace, IEC 1131 nebo IEC 61131 ?

V minulosti bylo učiněno mnoho pokusů sjednotit programování PLC automatů. Například německá norma DIN 40 719-6 a další národní specifikace. Žádná tato norma ale neměla nadnárodní platnost. Až norma IEC 1131 (International Elektrotechnical Comission) byla prvním oficiálním mezinárodním standardem. V současné době je platné značení normy IEC 61 131 místo původního 1131. Proč se změnilo číslování z 1131 na 61131? International Elektrotechnical Comission (IEC) je celosvětový standardizační orgán. Téměř všechny země na světě mají svůj vlastní národní standardizační úřad. Tyto úřady se domluvily na přijetí schválených a publikovaných standardů IEC. V místních

publikacích byl tento standard uveřejňován pod lokálním číslem. Často toto číslování nemělo žádnou spojitost s číslováním dle IEC. Aby došlo ke sjednocení číslování domluvily se standardizační úřady na novém čísle a to 61131. Tato změna proběhla při novém vydání standardů IEC.

Mezinárodní standard IEC 1131 byl vydán v roce 1993 a od jeho vydání se stal široce přijímaným. Dnes je jako takový celosvětově uznávaným standardem pro programování a projektování průmyslových programovatelných automatů. I přesto je zde několik důvodů, proč musí být standard upraven. Jedním z důvodů je, že od roku 1993 bylo získáno velké množství praktických zkušeností, kterými bylo odhaleno spousta nesrovnalostí a rozporů. Mnoho uživatelů navrhovalo úpravy a vylepšení, které byly zavedeny v opravách a dodatcích patřících k normě. Navíc požadavky na průmyslové řídící systémy a jejich strojírenské aplikace se během let podstatně změnily. Nejdůležitější změna nastala v pohledu na PLC z úzce centralizovaného systému na distribuovaný systém s nejrůznějšími způsoby komunikace mezi sebou a mezi nadřazeným systémem.

Nové vydání normy bylo publikováno jako mezinárodní standard v roce 2003. Tato nová verze zahrnuje i všechna vylepšení a opravy doposud publikované jako dodatky. Norma se nadále vyvíjí pod tíhou nových informačních technologií a poznatků z oboru. V Evropské unii byla tato norma přijata pod číslem EN IEC 61131.

2.3 Norma IEC 61131, EN 61131, ČSN EN 61131

Norma IEC 61131 pro programovatelné řídící systémy má pět základních částí. Ty představují soubor požadavků pro moderní řídící systémy. Jednotlivé části normy jsou členěny dle technických programových vybavení řídících systémů. Evropská unie převzala z velké části normu IEC 61131 pod označením EN 61131. V České republice je norma vedena pod označením ČSN EN 61131.

Jednotlivé části normy jsou:

- IEC 61131 – 1 Všeobecné informace

Tato část mezinárodní normy EN 61131 představuje první část souboru norem z oblasti programovatelných řídících jednotek a jejich přidružených vnějších jednotek, která by

měla být používána společně s dalšími sedmi částmi. Tato 1. část stanovuje definice a identifikuje základní vlastnosti vztahující se k výběru a aplikaci programovatelných řídicích jednotek a jejich přidružených vnějších jednotek. Mezinárodní norma obsahuje 18 stran anglického textu. V EU je norma převzata pod označením EN 61131-1 a v ČR je převzata pod označením ČSN EN 61131-1 v původním anglickém jazyce.

- IEC 61131 – 2 Požadavky na zařízení a zkoušky

Specifikuje požadavky a příslušné zkoušky pro programovatelné řídicí jednotky (PLC) a jejich přidružená zařízení (např. programovací a odlaďovací nástroje (PADT), rozhraní člověk-stroj (HMI), atd.), určená k řízení a ovládání zařízení a průmyslových procesů. Mezinárodní evropská norma obsahuje 120 stran anglického textu včetně textu EN. V EU je norma převzata pod označením EN 61131-2 a v ČR je norma harmonizovaná pod označením ČSN EN 61131-1 v českém jazyce.

- IEC 61131 – 3 Programovací jazyky.

Tato 3 část mezinárodní normy 61131 specifikuje syntaxi a sémantiku programovacích jazyků, určených pro programovatelné řídicí jednotky, jak je definováno v EN 61131, část 1. Definuje základní sadu programových prvků, syntaktická a sémantická programovací pravidla pro nejběžněji užívané programovací jazyky, včetně grafických jazyků LD (Ladder Diagram) a FBD (Functional Block Diagram) a textových jazyků IL (Instruction List) a ST (Structured Text). Dále zahrnuje popis prostředků pomocí kterých mohou výrobci rozšířit nebo přizpůsobit tuto základní sadu svým programovatelným automatům. Funkce programových vstupů, zkoušení, řídicího systému atd. jsou specifikovány v EN 61131, část 1. Evropská norma obsahuje 3 strany anglického textu a mezinárodní norma obsahuje 219 stran anglického textu. V EU je norma převzata pod označením EN 61131-3 a v ČR je norma schválená k přímému používání pod označením ČSN EN 61131-3 v českém jazyce a obsahuje pouze předmluvu bez originální normy.

- IEC 61131 – 4 Podpora uživatelů.

Technická zpráva poskytující základní přehled informací a pravidel pro užívání standardů pro koncového uživatele programovatelných automatů. V ČR tato norma zatím neexistuje.

- IEC 61131 – 5 Komunikace.

Tato část 61131 stanovuje komunikační hlediska programovatelných řídicích jednotek. Určuje vzájemné možnosti komunikace jakéhokoli zařízení s programovatelnou řídicí jednotkou. Stanovuje chování programovatelné řídicí jednotky při poskytování služeb jiným zařízením i služeb poskytovaných aplikačním programem programovatelné řídicí jednotky vyžádaných jiným zařízením. Chování programovatelné řídicí jednotky jako serveru, nebo klienta, je specifikováno nezávisle na konkrétním komunikačním podsystému. Definuje datovou komunikaci mezi programovatelnými automaty a jinými elektronickými systémy používajícími MMS (Manufacturing Message Specification) v souladu s mezinárodním standardem ISO/IEC 9506. V EU je norma převzata pod označením EN 61131-5 a v ČR je norma schválená k přímému používání pod označením ČSN EN 61131-5 v českém jazyce a obsahuje pouze předmluvu bez originální normy.

- IEC 61131 – 6 Rezervováno pro budoucí využití.

- IEC 61131 – 7 Programování FUZZY řízení.

Tato část 61131 definuje jazyk pro programování aplikací fuzzy řízení využívajících programovatelných řídicích jednotek. Poskytuje základní prostředky pro integraci aplikací fuzzy řízení do programovacích jazyků programovatelných řídicích jednotek podle IEC 61131-3 a možnosti výměny programů fuzzy řízení mezi různými programovacími systémy. Příloha A obsahuje stručný úvod do teorie fuzzy logiky a fuzzy řízení. Přebíraná evropská norma má 3 strany, původní IEC norma 113 stran. V EU je norma převzata pod označením EN 61131-7 a v ČR je norma schválená k přímému používání pod označením ČSN EN 61131-7 v českém jazyce a obsahuje pouze předmluvu bez originální normy.

- IEC 61131 – 8 Pravidla pro použití a implementaci programovacích jazyků.

Tato část 61131 definuje a popisuje softwarové vývojové průvodce pro programovací jazyky definované v části 3. V ČR tato norma zatím neexistuje.

2.3.1 Společné prvky

V rámci společných prvků jsou definovány typy dat. Běžné datové typy jsou Bool, Byte, Word, Integer, Real, Date, Time_of_Day, String. Z těchto typů je možné dále odvozovat uživatelské datové typy.

Proměnné mohou být přiřazeny k hardwarovým adresám pouze v konfiguracích, zdrojích nebo programech. Tím je umožněna hardwarová nezávislost. Proměnné jsou přístupné pouze v té programové jednotce, kde byly deklarovány (lokální proměnné). To znamená, že můžeme použít stejný název proměnné i v jiné programové jednotce a tyto proměnné nebudou na sobě závislé. Pro platnost ve všech jednotkách projektu můžeme deklarovat tzv. globální proměnnou. Při deklaraci proměnných můžeme ihned přiřadit inicializační hodnotu.

2.3.2 Konfigurace

Celý projekt řešení problému zastřešuje tzv. konfigurace. Konfigurace je závislá na daném hardware. V konfiguraci se definuje zdroj, případně více zdrojů. Ty reprezentují zařízení, které vykonává programy dle IEC 611131. Zdroje obsahují jednu nebo více úloh. Ty řídí provádění programů nebo funkčních bloků. Úlohy jsou prováděny periodicky nebo na základě daných událostí. Programy jsou sestaveny z prvků definovaných v normě ve formě jazyků. Programy jsou členěny na funkce resp. funkční bloky.

2.3.3 Programové organizační jednotky

Funkce, funkční bloky a programy jsou v normě nazývány společně programovými organizačními jednotkami (Program Organization Units), většinou jsou označované jako POUs. POU mohou být dodávány od výrobce nebo je může napsat uživatel. Existují tři základní typy POU:

- Funkce (function, FUN)
- Funkční blok (function block, FB)
- Program (program, PROG)

2.3.4 Funkce

Nejjednodušší POU je funkce. Funkce může vracet pouze jeden výsledek. Může se volat s několika parametry a opakovaně. Je velmi podobná programovacímu jazyku Pascal.

2.3.5 Funkční bloky

Na funkční bloky se můžeme dívat jako na integrované obvody, které řeší různé specializované funkce. Na rozdíl od funkcí si mohou pamatovat minulé hodnoty. Obsahují různě složité algoritmy. Mají definované rozhraní, vnitřní proměnné. Používají se pro zpřehlednění programování a možnost opakovaného používání stejně jako funkce. Příkladem jsou čítač, časovač, PID regulátor a další, mohou být dodávané jako knihovny, které jsou součástí vývojového prostředí. Funkční bloky v rámci jednoho projektu mohou být napsány v libovolném jazyce definovaném v normě.

2.3.6 Program

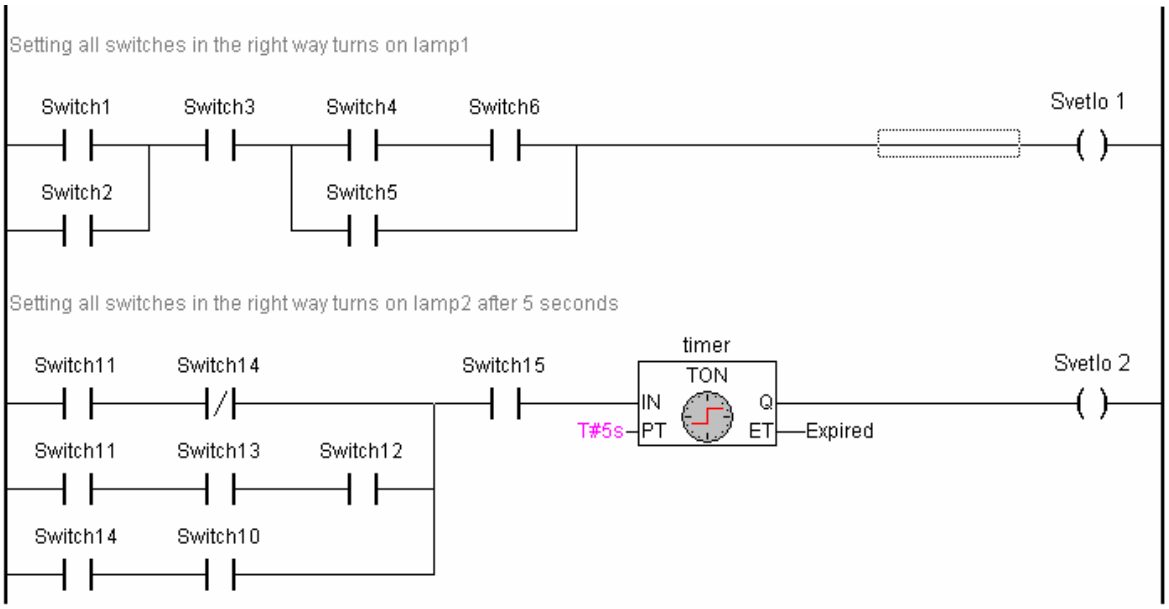
Program je sítí funkcí a funkčních bloků. Opět může být napsán v libovolném z jazyků definovaných v normě.

2.3.7 Programovací jazyky

V normě IEC 61131-3 jsou zařazeny čtyři programovací jazyky. Jejich sémantika a syntaxe je normou přesně definována. Jsou to: LD, FBD, IL, ST. Zvládnutím těchto jazyků se nám otevírá cesta pro rychlou realizaci požadované aplikace pomocí hardware různých výrobců, kteří přijali tuto normu. Jako další jazyky se uvádějí SFC, CFC, nejsou však zařazeny přímo mezi jazyky, ale jsou to tzv. společné prvky tvořící nadstavbu pro strukturování celé aplikace.

2.3.7.1 LD (Ladder Diagram) – jazyk příčkového diagramu

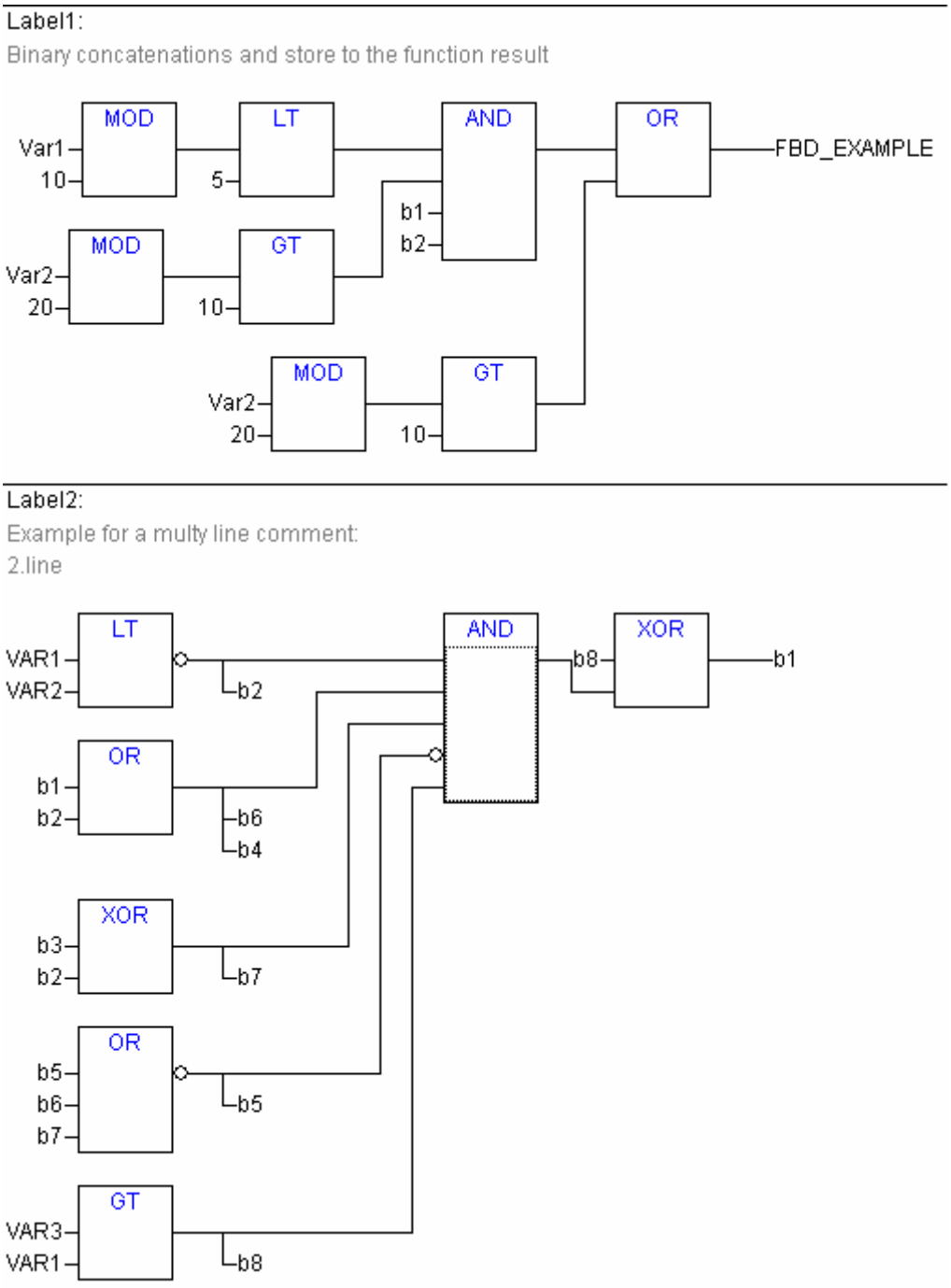
Patří mezi tzv. grafické jazyky. Je někdy nazýván jazykem kontaktních schémat. Je založen na grafické reprezentaci reléové logiky. Organizační jednotka programu je vyjádřena sítí propojených grafických prvků. Síť v jazyce LD je zleva i zprava ohraničena svislými čarami, které se nazývají levá a pravá napájecí sběrnice. Mezi nimi je tzv. příčka, která může být rozvětvena. Každý úsek příčky, vodorovný nebo svislý, může být ve stavu *on* nebo *off* . Do příček mohou být včleněny kontakty (spínací, rozpínací apod.), cívky a funkce nebo funkční bloky. Na obrázku 2.3 – 1 je ukázka programu v jazyce příčkového diagramu.



Obr. 2.3 – 1 Ukázka jazyku příčkového diagramu

2.3.7.2 FBD (Function Block Diagram) – jazyk funkčního blokového schématu

FBD vyjadřuje chování funkcí, funkčních bloků a programů jako soubor vzájemně provázaných grafických bloků podobně jako v elektrotechnických obvodech. Jde o systém prvků, které zpracovávají signály. Často se zde používají standardní funkční bloky, jako jsou např. bistabilní prvky, prvky pro detekci náběžné a sestupné hrany, čítače, časovače a komunikační bloky. Příklad je na obrázku 2.3 - 2.



Obr. 2.3 – 2 Ukázka programu v jazyku funkčního blokového schématu

2.3.7.3 IL (Instruction List) – jazyk seznamu instrukcí

Textový jazyk IL označovaný také jako jazyk pokynů, trochu připomíná assembler. (Assembler je jazyk symbolických adres. Z hlediska úrovně je to nejnižší programovací jazyk. Převádí textové reprezentace instrukcí na čísla strojového kódu procesoru.) Programová organizační jednotka je složena ze sekvence instrukcí, z nichž každá začíná na novém řádku a může obsahovat návěští ukončené dvojtečkou; operátor, který může být doplněn modifikátorem, někdy také komentářem. Pomocí modifikátorů se vyjadřují negace, podmíněnost a nepodmíněnost instrukce skoků, volání a návratů a priorita. Viz obrázek 2.3 – 3.

```
(* Calculate sinus of r1 and  multiply with 1000  *)
LD    r1
SIN
MUL      1000
ST    sinus
(* Calculate cosinus of r1 and  multiply with 1000  *)
LD    r1
COS
MUL      1000
ST    cosinus

(* increment r1 *)
LD    r1
ADD      0.1
ST    r1
```

Obr. 2.3 – 3 Ukázka programu v jazyku seznamu instrukcí

2.3.7.4 ST (Structured Text) – jazyk strukturovaného textu

ST je výkonný vyšší programovací jazyk, který má kořeny v jazycích Pascal a C. Syntaxe jazyka je dána povolenými výrazy a příkazy. Vyhodnocením výrazu vyjde hodnota v některém z definovaných datových typů. Výraz se skládá z operátorů a operandů. Operandem může být konstanta, proměnná, funkce nebo jiný výraz. Operátory pro jazyk ST jsou definovány pro sedmnáct typů operací (vyhodnocení, funkce, negace, násobení, boolovské funkce AND, XOR, OR apod.). Je definováno deset typů příkazů

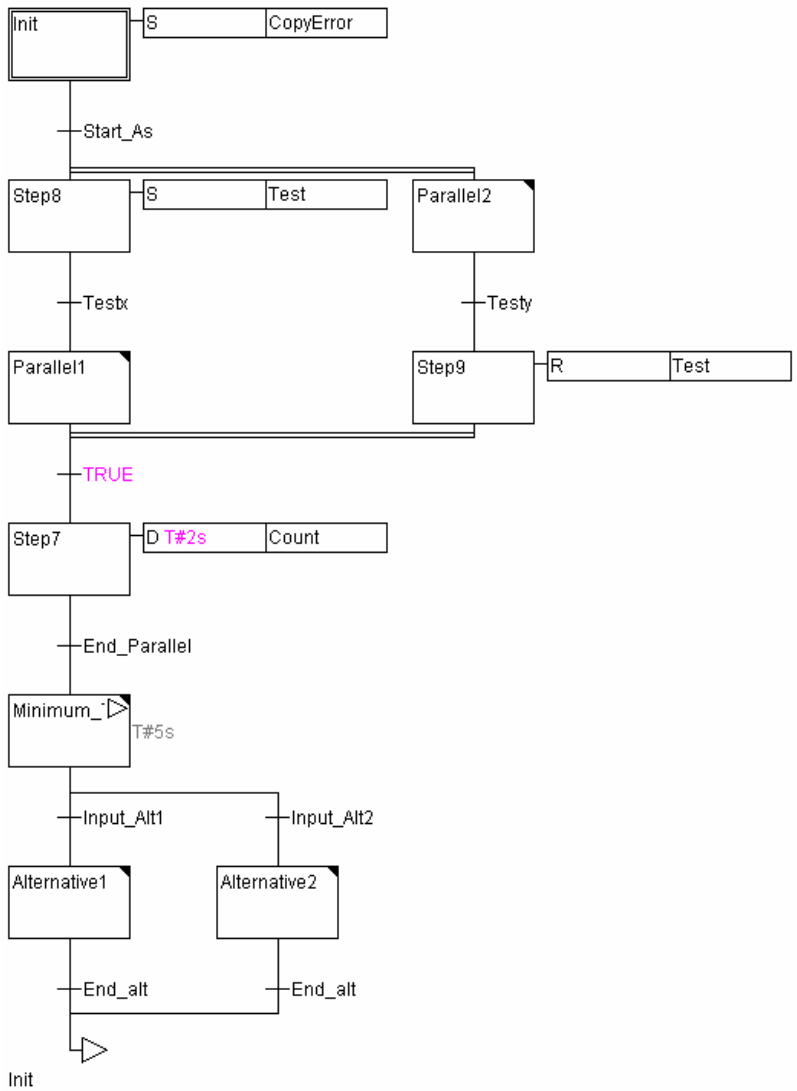
(přiřazení, vyvolání funkce, návrat, výběr apod.). Příkazy jsou odděleny středníkem a může jich být více na jednom řádku. Jazyk ST je vhodným nástrojem pro definování komplexních funkčních bloků, které pak mohou být použity v libovolném programovacím jazyku. Viz obrázek 2.3 - 4.

```
run_string:='Start';
IF NOT run THEN
    RETURN;
END_IF;
run_string:='Stop';

rot := rot + offset;

IF (yVal < 0) THEN
    yVal := yVal + offset;
    bottom := yVal + offset;
ELSE
    IF(xVal < 470) THEN
        xVal := xVal+offset;
    END_IF
    IF (bottom > -250) THEN
        bottom := bottom -offset;
    END_IF
END_IF

i := 0;

IF xVal >= 470 THEN
    wait := wait +1;
    IF (wait >= 10 AND wait <= 20) THEN
        inv := FALSE;
        inv2 := TRUE;
    ELSE
        inv := TRUE;
        inv2 := FALSE;
    END_IF
    IF (wait > 20) THEN
        scale := scale - 50;
    END_IF
    IF (wait > 41) THEN
        wait := 0;
        xVal := 0;
        yVal := -250;
        scale := 1000;
    END_IF
END_IF

WHILE i < 6 DO
    stripes[i] := stripes[i]+offset;
    IF(stripes[i] >= 540) THEN
        stripes[i] := 0;
    END_IF
    i := i + 1;
END_WHILE
```

Obr. 2.3 – 4 Ukázka programu v jazyku strukturovaného textu

2.3.7.5 SFC (Sequential Function Chart) – sekvenční funkční diagram

SFC popisuje sekvenční chování řídícího programu. Grafická reprezentace se převádí přímo do souboru výkonných řídících prvků. SFC strukturalizuje vnitřní organizaci programu a umožňuje rozložit úlohu řízení na zvládnutelné části a zachovat přitom přehled o chování celku. Sekvenční diagram se skládá z kroků a přechodů. Každý krok reprezentuje daný stav systému a má přiřazen blok akcí. Přechod je spojen s podmínkami, které musí být splněny. Každý blok nebo přechod může být naprogramován v libovolném jazyce daném normou i v SFC. Ukázka je na obrázku 2.3 - 5.



Obr. 2.3 – 5 Ukázka programu v jazyku sekvenčního funkčního diagramu

2.4 Jazyk ST podrobně

Syntaxe popisuje prvky, které je možné využít při programování v jazyce ST a způsob, jakým se mohou vzájemně kombinovat. Sémantika vyjadřuje význam jednotlivých prvků.

2.4.1 Prvky

Program v jazyce ST se skládá z prvků. Jsou to:

- Klíčová slova
- Identifikátory
- Literáty
- Oddělovače
- Komentáře

2.4.2 Klíčová slova

Klíčová slova jsou zvýrazněna jinou barvou nebo jsou psána tučně pro zpřehlednění struktury programu. Klíčová slova jsou standardní identifikátory. Jejich přesný tvar je popsán v normě IEC 61131-3. Nesmějí se používat jinak než je jejich původní význam, tj. např. pro názvy proměnných. Mohou se psát jak s malými, tak s velkými písmeny nebo jejich kombinací. Ke klíčovým slovům patří:

- Jména standardních funkcí
- Jména standardních funkčních bloků
- Jména standardních parametrů standardních funkcí
- Jména vstupních a výstupních parametrů standardních funkčních bloků
- Jména elementárních datových typů
- Prvky jazyka

Komentáře bývají důležitou součástí dokumentace programu. Překladačem jsou komentáře ignorovány. Existují dva typy komentářů:

- Obecný komentář
- Řádkový komentář

Obecný komentář začíná dvojicí znaků „(*“ a končí „*)“. Může obsahovat jakékoli znaky a může být dlouhý i přes několik řádků. Řádkový komentář začíná dvojicí znaků“//“ a končí koncem řádku.

2.4.3 Identifikátory

Identifikátor je řetězec malých nebo velkých písmen, číslic nebo podtržítka, který lze použít pro:

- Jména úloh
- Jména funkcí, funkčních bloků a programů
- Jména odvozených datových typů
- Jména konstant
- Jména proměnných

Identifikátor nesmí začínat číslicí, nesmí se používat národní znaky a nesmí obsahovat mezery. Znak podtržítko se nesmí po sobě opakovat. Maximální délka je 64 znaků.

Platné identifikátory	Neplatné identifikátory
BEH_MOTORU	BĚH_MOTORU
_zap_cerpadlo	__zap_cerpadlo (dve podtržítka vedle sebe)
ventil_183_otevri	Ventil_54.2.3_otevri
DF_47	DF 47
AD45b	AD$45б
PT5	5PT
Rele_4_Zap, RELE_4_ZAP, rele_4_ZAP	Stejné identifikátory

Obr. 2.4 – 1 Příklady identifikátorů

2.4.4 Literály

Vnější reprezentace dat v ST jazyce sestává z:

- Numerických literátů
- Řetězců znaků
- Časových literátů

2.4.4.1 Numerické literály

Numerický literát je definován jako číselná konstanta v jakékoli soustavě (nejen desítkové). Dělí se na INTEGER a REAL.

2.4.4.2 Literály řetězce

Řetězec znaků je posloupnost znaků uvozených jednoduchou uvozovkou. Může být prázdný. Řetězce znaků se používají pro výměnu textů mezi PLC nebo mezi PLC a jinými komponentami. Dále se používají řetězce při programování textů pro zobrazení na displejích PLC nebo na zobrazovacích panelech určených pro obsluhu.

2.4.4.3 Časové literáty

Existují dva druhy časových literátů, absolutní čas a datum a údaj o trvání. Absolutním časem se mohou řídit děje závislé na tomto údaji, např. noční a denní proud atd. Údaj o trvání může informovat o délce nějakého děje nebo události. Příklady časových literálů jsou v následující tabulce na obrázku 2.4 – 2.

Popis	Příklady
Doba trvání	T#37m, t#56m44s, TIME#587ms, t#458.45ms, T#7h_45m_20s
Datum	D#2006-03-28, date#2005-12-20
Denní čas	TOD#08:55:45.09, TIME_OF_DAY#11:28:55.48
Datum a denní čas	DATE_AND_TIME#2004-11-20->13:45:55.14, DT#2001-05-30-01:12:45.22

Obr. 2.4 – 2 Příklady časových literátů

2.4.5 Datové typy

Dle normy IEC 61131-3 jsou pro ST jazyk definovány tyto skupiny datových typů:

- Elementární (předdefinované datové typy)
- Rodové (pro příbuzné skupiny datových typů)
- Uživatelské (odvozené datové typy)

2.4.5.1 Elementární datové typy

Jsou charakteristické obsazením paměti v bitech, případně rozsahem hodnot. Přehled podporovaných datových typů je uveden v následující tabulce na obrázku 2.4 - 3:

Klíčové slovo	Datový typ	bitů	Rozsah hodnot
BOOL	Booleovské číslo	1	0,1
SINT	Short integer, krátké celé číslo	8	-128 až +127
INT	Integer, celé číslo	16	-32 768 až +32767
DINT	Double integer, celé číslo dvojnásobná délka	32	-2147483648 až +2147483647
USINT	Unsigned short integer, celé číslo bez znaménka krátké	8	0 až 255
UINT	Unsignet integer, celé číslo bez znaménka	16	0 až 65535
UDINT	Unsigned double integer, celé číslo bez znaménka dvojnásobná délka	32	0 až 4294967295
REAL	Real, Číslo v pohyblivé řádové čárce	32	1.175494351e-38F až 3.402823466e+38F
LREAL	Long real, číslo v pohyblivé řádové čárce dvojnásobná délka	64	2.2250738585072014e-308 až 1.7976931348623158e+308
TIME	Duration, trvání času		Max.: 49d17h2m47s295ms
DATE	Pouze datum		1970-00-00 až 2106-02-06
TIME_OF_DAY, TOD	Denní čas		00:00:00 až 23:59:59.999
DATE_AND_TIME, DT	Absolutní čas		1970-00-00-00:00:00 až 2106-02-06-06:28:15
STRING	Proměnlivě dlouhý string		Max.: 255 znaků
BYTE	Sekvence 8 bitů	8	0 až 255
WORD	Sekvence 16 bitů	16	0 až 65535
DWORD	Sekvence 32 bitů	32	0 až 4294967295

Obr. 2.4 – 3 Podporované datové typy dle IEC 61131 – 3

2.4.5.2 Rodové datové typy

Rodové datové typy vyjadřují celou skupinu datových typů. Jsou uvozeny prefixem ANY, např. ANY_INT znamená všechny datové typy INT, SINT, DINT, UINT, USINT, UDINT.

Přehled je v následující tabulce na obrázku 2.4 - 4:

<table>
<tr><th colspan="5">ANY</th></tr>
<tr><th>ANY_BIT</th><th colspan="3">ANY_NUM</th><th>ANY_DATE</th><td rowspan="2">TIME
STRING
uživatelské</td></tr>
<tr><td rowspan="4">BOOL
BYTE
WORD
DWORD</td><th colspan="2">ANY_INT</th><th>ANY_REAL</th><td rowspan="3">DATE
TIME_OF_DATE</td></tr>
<tr><td>INT
SINT
DINT</td><td>UINT
USINT
UDINT</td><td>REAL
LREAL</td><td></td></tr>
</table>

Obr. 2.4 – 4 Rodové datové typy

2.4.5.3 Uživatelské datové typy

Odvozené datové typy mohou být deklarovány pomocí konstrukce TYPE….END_TYPE. Tyto odvozené typy se mohou používat spolu s elementárními datovými typy v deklaracích proměnných. Deklarace nového datového typu může být provedena výčtem hodnot nebo jako dílčí rozsah v rozmezí dolní a horní uvedené meze. Může to být také pole nebo struktura.

```
TYPE
        Kniha  : (volna, pujcena, zamluvena);
        Teplota: INT;
        Teploty : ARRAY [1..5] OF Teplota;

        Kontakt:
          STRUCT
            Jmeno       : STRING;
            Prijmeni    : STRING;
            Muz         : BOOL;
            Vaha        : INT;
          END_STRUCT;
END_TYPE
```

Obr. 2.4 – 5 Příklad deklarace odvozených datových typů

2.4.5.4 Inicializace datového typu

Elementární datové typy mají definované počáteční hodnoty. Jsou to nuly, prázdný string a u DATA je to D#1970-01-01 a u DATE_AND_TIME DT#1970-01-01-00:00:00. Počáteční hodnota se dá přiřadit přímo v deklaraci přiřazovacím operátorem „:=“.

2.4.6 Proměnné

Dle IEC 61131-3 jsou proměnné prostředkem k identifikaci datových objektů, jejichž obsah se může měnit. Ty představují vstupy, výstupy nebo paměť PLC. Proměnná může být deklarována buď elementárním nebo uživatelským datovým typem.

2.4.6.1 Jednoduché proměnné

Jsou definovány jako proměnné, které reprezentují jednoduchý datový prvek jednoho z elementárních datových typů nebo uživatelského datového typu. Reprezentace proměnných může být symbolická nebo přímá (tj. přiřazení prvku dat k fyzickým pozicím na vstupech nebo výstupech a paměti PLC). Pro symbolickou reprezentaci proměnných se používají identifikátory. Pro přímou reprezentaci se používá pro uvození speciální znak procento „%“, prefix umístění a prefix šíře dat. Za těmito znaky následuje jeden nebo více znaků typu UINT oddělených tečkami.

Prefix umístění	Význam
%I	Vstup
%Q	Výstup
%M	Paměťové místo

Prefix šíře dat	Velikost
X	1 bit
Žádný	1 bit
B	1 byte (8 bitů)
W	1 word (16 bitů)
D	1 doble word (32 bitů)

Příklady přímo reprezentovaných proměnných:

%QX5.2 – výstupní bit číslo 2 ve skupině 5
%IB3 – vstupní byte číslo 3

2.4.6.2 Složené proměnné

Složené proměnné jsou tyto:

- Pole
- Strukturovaná proměnná

Pole je množina datových prvků stejného datového typu, na které je možné se odkázat pomocí jednoho nebo více indexů uzavřených v závorkách a oddělených čárkami. Maximální počet indexů je 4 a maximální rozsah indexů odpovídá typu INT.

Strukturovaná proměnná je deklarovaná typem, který je specifikován jako struktura dat. Prvek strukturované proměnné je reprezentován dvěma nebo více identifikátory nebo místy v poli, které jsou odděleny tečkou. První identifikátor představuje jméno celé struktury a následující identifikátory představují sekvenci jmen prvků, které vedou až ke konkrétnímu typu struktury. Maximální počet úrovní není omezen. Omezení představuje maximální délka identifikátoru 64 znaků.

2.4.6.3 Inicializace proměnných

Po restartu programu může každá z proměnných nabýt jedné z následujících hodnot:

- Hodnotu, kterou měla proměnná v okamžiku zastavení konfiguračního prvku (retain value)
- Uživatelem specifikovanou počáteční hodnotu
- Předdefinovanou počáteční hodnotu (default)

Uživatel může deklarovat proměnnou pomocí kvalifikátoru RETAIN, aby se uchovala její poslední hodnota. To platí pouze pro globální proměnné.

2.4.6.4 Deklarace proměnných

Každá POU v ST jazyku má mít na začátku deklarační část. Ta specifikuje datové typy používané v programové organizační jednotce. Deklarační část má textovou podobu a používá jedno z klíčových slov VAR, VAR_INPUT, VAR_OUTPUT. Následuje jedna nebo více deklarací proměnných oddělených středníkem a deklarace je ukončena klíčovým slovem END_VAR. Platnost deklarací je lokální pro danou POU. Explicitně se mohou předávat parametry přes vstupní proměnné (VAR_INPUT), resp. výstupní (VAR_OUTPUT). Globální proměnné společné pro všechny POU použité v projektu se deklarují vně POU s použitím klíčového slova VAR_GLOBAL.

2.4.6.5 Přiřazení počáteční hodnoty

Konstrukce VAR….END_VAR umožňuje přiřazení počáteční hodnoty proměnných pomocí příkazu přiřazení „:=“. Např.:

VAR

Počet : **INT** := 100;

END_VAR

2.4.7 Funkce

Pro účely programovacích jazyků pro PLC je funkce definována jako programová organizační jednotka, která po provedení vygeneruje jeden datový element. Funkce neobsahují žádnou vnitřní stavovou informaci, tzn. Že volání funkce se stejnými argumenty vytvoří vždy stejný výsledek.

2.4.7.1 Standardní funkce

Jazyk ST disponuje skupinou standardních funkcí, které jsou využitelné ve všech POU. Některé funkce jsou rozšiřitelné, tzn. že mohou mít proměnný počet vstupů, který u těchto funkcí není omezen.

Standardní funkce jsou rozděleny do několika základních skupin:

- Funkce pro konverzi typu
- Numerické funkce jedné proměnné
- Numerické funkce více proměnných
- Funkce nad řetězcem bitů, rotace bitů
- Funkce nad řetězcem bitů, booleovské funkce
- Funkce výběru
- Funkce porovnání
- Funkce nad řetězcem znaků
- Funkce s typy datum a čas
- Funkce nad datovými typy výčet

V následujícím přehledu je specifikace standardních funkcí jazyka ST. Sloupec označený „Př.“ udává, jestli je funkce přetížená a sloupec označený „Roz.“ udává, jestli je funkce rozšiřitelná.

Standardní funkce, skupina konverze typu:

Standardní funkce, skupina konverze typu					
Jméno funkce	**Datový typ vstupu**	**Datový typ výstupu**	**Popis funkce**	**Př.**	**Roz.**
…_TO_...	ANY	ANY	Konverze datového typu uvedeného na prvním místě na datový typ uvedený na druhém místě	+	-
TRUNC	ANY_REAL	ANY_INT	Ořezání	+	-
BCD_TO…	ANY_BIT	ANY_INT	Převod z BCD kódu na celé číslo	+	-
…TO_BCD	ANY_INT	ANY_BIT	Převod z celého čísla na kód BCD	+	-

Standardní funkce, skupina numerické funkce jedné proměnné

Standardní funkce, skupina numerické funkce jedné proměnné				
Jméno funkce	**Datový typ vstupu / výstupu**	**Popis funkce**	**Př.**	**Roz.**
ABS	ANY_NUM / ANY_NUM	Absolutní hodnota	+	-
SQRT	ANY_REAL / ANY_REAL	Odmocnina	+	-
LN	ANY_REAL / ANY_REAL	Přirozený logaritmus	+	-
LOG	ANY_REAL / ANY_REAL	Desítkový logaritmus	+	-
EXP	ANY_REAL / ANY_REAL	Přirozená exponenciální funkce	+	-
SIN	ANY_REAL / ANY_REAL	Sinus úhlu v radiánech	+	-
COS	ANY_REAL / ANY_REAL	Kosinu úhlu v radiánech	+	-
TAN	ANY_REAL / ANY_REAL	Tangens úhlu v radiánech	+	-
ASIN	ANY_REAL / ANY_REAL	Arcus sinus	+	-
ACOS	ANY_REAL / ANY_REAL	Arcus kosinus	+	-
ATAN	ANY_REAL / ANY_REAL	Arcus tangens	+	-

Standardní funkce, numerické funkce více proměnných

Standardní funkce, skupina numerické funkce více proměnných					
Jméno funkce	**Datový typ vstupu / výstupu**	**symbol**	**Popis funkce**	**Př.**	**Roz.**
ADD	ANY_NUM / ANY_NUM	+	Součet	+	+
MUL	ANY_REAL / ANY_REAL	*	Součin	+	+
SUB	ANY_REAL / ANY_REAL	-	Rozdíl	+	-
DIV	ANY_REAL / ANY_REAL	/	Podíl	+	-
MOD	ANY_REAL / ANY_REAL		Modulo	+	-
EXPT	ANY_REAL / ANY_REAL	**	Umocnění	+	-
MOVE	ANY_REAL / ANY_REAL	:=	Přesunutí, přiřazení	+	-

Standardní funkce nad řetězcem bitů, rotace bitů

Standardní funkce, funkce nad řetězcem bitů, rotace bitů				
Jméno funkce	**Datový typ vstupu / výstupu**	**Popis funkce**	**Př.**	**Roz.**
SHL	ANY_BIT / ANY_BIT	Posun vlevo, zprava doplněno nulami	+	-
SHR	ANY_BIT / ANY_BIT	Posun vpravo, zleva doplněno nulami	+	-
ROR	ANY_BIT / ANY_BIT	Rotace vpravo, zleva odrotovaný bity	+	-
ROL	ANY_BIT / ANY_BIT	Rotace vlevo, zprava orotované bity	+	-

Standardní funkce nad řetězcem bitů, booleovské operace

Standardní funkce nad řetězcem bitů, booleovské operace					
Jméno funkce	**Datový typ vstupu / výstupu**	**symbol**	**Popis funkce**	**Př.**	**Roz.**
AND	ANY_BIT / ANY_BIT	&	Log. Součin	+	+
OR	ANY_BIT / ANY_BIT		Log. Součet	+	+
XOR	ANY_BIT / ANY_BIT		exkluzivní součet	+	+
NOT	ANY_BIT / ANY_BIT		Negace	+	-

Standardní funkce, skupina funkce výběru

Standardní funkce, skupina funkce výběru				
Jméno funkce	**Datový typ vstupu / výstupu**	**Popis funkce**	**Př.**	**Roz.**
SEL	BOOL, ANY, ANY / ANY	Binární výběr	+	-
MAX	ANY / ANY	Maximum	+	+
MIN	ANY / ANY	Minimum	+	+
LIMIT	MN, ANY, MX / ANY	Omezovač	+	-

Standardní funkce, skupina porovnání

Standardní funkce, skupina porovnání				
Jméno funkce	**Datový typ vstupu / výstupu**	**Popis funkce**	**Př.**	**Roz.**
GT	ANY / BOOL	Klesající sekvence	+	+
GE	ANY / BOOL	Monotónní sekvence	+	+
EQ	ANY / BOOL	Rovnost	+	+
LE	ANY / BOOL	Monotónní sekvence směrem nahoru	+	+
LT	ANY / BOOL	Vzrůstající sekvence	+	+
NE	ANY / BOOL	Nerovnost	+	-

Standardní funkce, skupina funkcí nad řetězcem znaků

Standardní funkce, skupina funkcí nad řetězcem znaků				
Jméno funkce	**Datový typ vstupu / výstupu**	**Popis funkce**	**Př.**	**Roz.**
LEN	STRING / INT	Délka řetězce	+	-
LEFT	STRING, ANY_INT / INT	L znaků zleva přesunout do výstupního řetězce	+	-
RIGHT	STRING, ANY_INT / INT	L znaků zprava přesunout do výstupního řetězce	+	-
MID	STRING, ANY_INT / INT	L znaků od P znaku přesunout do výstupního řetězce	+	-
CONCAT	STRING / STRING	Připojení jednotlivých vstupních řetězců do výstupního	-	+
INSERT	STRING, STRING, ANY_INT / INT	Vložení řetězce druhého vstupu do řetězce prvního vstupu počínaje od P pozice	+	+
DELETE	STRING, ANY_INT, ANY_INT / STRING	Smazání L znaků ze řetězce vstupu počínaje od P pozice	+	+
REPLACE	STRING, STRING, ANY_INT, ANY_INT / STRING	Náhrada L znaků řetězce prvního vstupu znaky řetězce druhého vstupu, vkládání od P pozice	+	+
FIND	STRING, STRING / INT	Nalezení pozice prvního znaku prvního výskytu řetězce z druhého vstupu v řetězci prvního vstupu	+	+

Standardní funkce, skupina s typy datum a čas

Standardní funkce, skupina s typy datum a čas						
Jméno funkce	**symbol**	**IN1**	**IN2**	**OUT**	**Př.**	**Roz.**
ADD	+	TIME TOD DT	TIME TIME TIME	TIME TOD DT	+	-
SUB	-	TIME DATE TOD TOD DT DT	TIME DATE TIME TOD TIME DT	TIME TIME TOD TIME DT TIME	+	-
MUL	*	TIME	ANY_NUM	TIME	+	-
DIV	/	TIME	ANY_NUM	TIME	+	-
Funkce konverze typu						
DATE_AND_TIME_TO_TIME_OF_DAY, DAT_TO_TIME, DATE_AND_TIME_TO_DATE, DAT_TO_DATE						
TOD – TIME_OF_DATE, DT – DATE_AND_TIME						

2.4.8 Funkční bloky

Funkční blok v jazyku ST je organizační jednotka, která vygeneruje jednu nebo více hodnot. Z funkčních bloků lze vytvářet násobné kopie tzv. instance. Každá instance má přiřazený identifikátor (jméno instance) a datovou strukturu, která obsahuje její vstupní, vnitřní a výstupní proměnné. Všechny hodnoty proměnných v této datové struktuře se uchovávají od jednoho provedení funkčního bloku k dalšímu provedení. Vyvolání funkčního bloku se stejnými argumenty tedy nemusí vždy vést ke stejnému výsledku na rozdíl od funkce. Jakýkoli funkční blok může být použit v deklaraci dalšího funkčního bloku. Rozsah působnosti funkčního bloku je lokální pro danou programovou organizační jednotku, pokud není deklarován jako globální. Zvenku jsou přístupné pouze vstupně - výstupní instance. Vnitřní proměnné jsou z hlediska uživatele skryté. Přiřazení hodnoty výstupní proměnné z venku není možné, tu přiřazuje pouze funkční blok sám. Přiřazení hodnoty vstupu je možné kdekoli v rámci příslušné POU, převážně se tak děje při samotném volání funkčního bloku.

2.4.8.1 Deklarace funkčního bloku

- Klíčová slova pro deklaraci funkčních bloků jsou FUNCTION_BLOCK…END_FUNCTION_BLOCK
- Funkční blok může mít více než jeden výstupní parametr, deklarovaný textově konstrukcí VAR_OUTPUT…END_VAR
- Hodnoty proměnných, které jsou předávány funkčnímu bloku pomocí konstrukce VAR_IN_OUT nebo VAR_EXTRENAL mohou být modifikovány zevnitř funkčního bloku.
- V deklaraci vstupních proměnných funkčního bloku mohou být použity kvalifikátory R_EDGE a F_EDGE. Tyto kvalifikátory označují funkci detekce hran na Booleovských vstupech. Tím je vyvolána implicitní deklarace funkčního bloku R_TRIG nebo F_TRIG.
- Konstrukce definovaná pro inicializaci funkcí se používá i pro deklaraci defaultních hodnot vstupů funkčního bloku a pro počáteční hodnoty jeho vnitřních a výstupních proměnných.

2.4.9 Standardní funkční bloky

V normě IEC 61131-3 jsou podrobně definovány standardní funkční bloky. Jsou rozděleny do následujících skupin:

- Bistabilní prvky
- Detekce hrany
- Čítače
- Časovače

Přehled standardních funkčních bloků

Jméno standard. funkčního bloku	Jméno vstupního parametru	Jméno výstup. parametru	Popis
Klopné obvody			
SR	S1, R	Q1	Dominantní nastavení (sepnutí)
RS	S, R1	Q1	Dominantní mazání (vypnutí)
Detekce hrany			
R_TRIG	CLK	Q	Detekce náběžné hrany
F_TRIG	CLK	Q	Detekce sestupné hrany
Čítače			
CTU	CU, R, PV	Q, CV	Dopředný čítač
CTD	CD, LD, PV	Q, CV	Zpětný čítač
CTUD	CU, CD, R, LD, PV	QU, QD, CV	Reverzibilní čítač
Časovače			
TP	IN, PT	Q, ET	Pulzní časovač
TON (T—0)	IN, PT	Q, ET	Zpoždění náběžné hrany
TOF (0—T)	IN, PT	Q, ET	Zpoždění sestupné hrany
RTC	EN, PDT	Q, CDT	Hodiny reálného času

Názvy, významy a datové typy proměnných používané u standardních funkčních bloků.

Název vstupu, výstupu	význam	Datový typ
R	Resetovaní vstup	BOOL
S	Setovací vstup	BOOL
R1	Dominantní resetovaní vstup	BOOL
S1	Dominantní setovací vstup	BOOL
Q	Výstup	BOOL
Q1	Výstup klopných obvodů	BOOL
CLK	Hodinový signál	BOOL
CU	Vstup pro dopředné čítání	BOOL
CD	Vstup pro zpětné čítání	BOOL
LD	Nastavení předvolby čítače	BOOL
PV	Předvolba čítače	INT
QD	Výstup zpětného čítače	BOOL
QU	Výstup dopředného čítače	BOOL
CV	Aktuální hodnota čítače	INT
IN	Vstup časovače	BOOL
PT	Předvolba časovače	TIME
ET	Aktuální hodnota časovače	TIME
PDT	Předvolba datumu a času	DT
CDT	Aktuální hodnota datumu a času	DT

2.4.10 Programy

Podle normy IEC 61131 je program definován jako „logický souhrn prvků programovacích zařízeních jazyků a konstrukcí nutných pro zamýšlené zpracování signálů, které je vyžadováno pro řízení stroje nebo procesu systémem programovatelného automatu“. Jinak řečeno funkce a funkční bloky lze přirovnat k podprogramům a POU program je hlavní program. Deklarace a používání programů je identické s deklarací a používání funkčních bloků.

Odlišnosti programů od funkčních bloků:

- Deklarace programu je pomocí klíčových slov PROGRAM…END_PROGRAM
- Programy mohou být instanciovány pouze v rámci zdrojů, funkční bloky mohou být instanciovány pouze v rámci programů nebo jiných funkčních bloků
- Programy mohou volat funkce a funkční bloky. Volání programů z funkcí nebo funkčních bloků není možné.

2.4.10.1 Výrazy

Výraz je konstrukce, ze které se po vyhodnocení vygeneruje hodnota jednoho z definovaných datových typů. Výraz je složen z operátorů a operandů. Operandem může být literát, proměnná, volání funkce nebo jiný výraz. Operátory jazyka ST jsou vyjmenovány v následující tabulce a seřazeny od nejvyšší priority k nejnižší:

Operátor	Operace
()	Závorky
**	Umocňování
-	Znaménko
NOT	Doplněk
*	Násobení
/	Dělení
MOD	Modulo
+	Sčítání
-	Odčítání
<, >, <=, >=	Porovnání
=	Rovnost
<>	Nerovnost
&, AND	Booleovské AND
XOR	Booleovské exkluzivní OR
OR	Booleovské OR

2.4.10.2 Příkazy

ST jazyk obsahuje podstatné prvky moderního programovacího jazyka, včetně větvení IF, THEN, ELSE, CASE OF a iterační smyčky FOR, WHILE, REPEAT. Všechny tyto prvky mohou být vnořovány. Tento jazyk je velmi vhodný pro definování komplexních funkčních bloků. Algoritmus v jazyce ST lze rozdělit na jednotlivé příkazy. Příkazy se používají pro výpočet a přiřazení hodnot, řízení toku programu a pro volání a ukončení POU. Část příkazu, která vypočítává hodnotu je výraz. Výrazy produkují hodnoty nezbytné pro provádění příkazů. Příkazy jsou ukončeny středníkem. Seznam příkazů jazyka ST je v následující tabulce:

Příkaz	Popis	Příklad	Poznámka
:=	Přiřazení	C := 135	Přiřazení hodnoty vypočtené na pravé straně do identifikátoru na levé straně
	Volání funkčního bloku	InstanceFB(A:=5, B:= 7);	Volání funkčního bloku s předáváním parametrů
IF	příkaz výběru	IF A > 7 THEN B := 1; ELSE B := 0; END_IF;	Výběr alternativy
CASE	Příkaz výběru	CASE num OF 1 : A := 10; 2 : A := 20; ELSE A := 0; END_CASE;	Výběr více alternativ

FOR	Iterační příkaz smyčka FOR	FOR i := 0 TO 10 BY 2 DO
		j := j +1;
		END_FOR;
			Vícenásobná smyčka bloku příkazů s počáteční a koncovou podmínkou

WHILE	Iterační příkaz smyčka WHILE	WHILE i > 0 DO
		j := j +1;
		END_WHILE;
			Vícenásobná smyčka s podmínkou ukončení smyčky

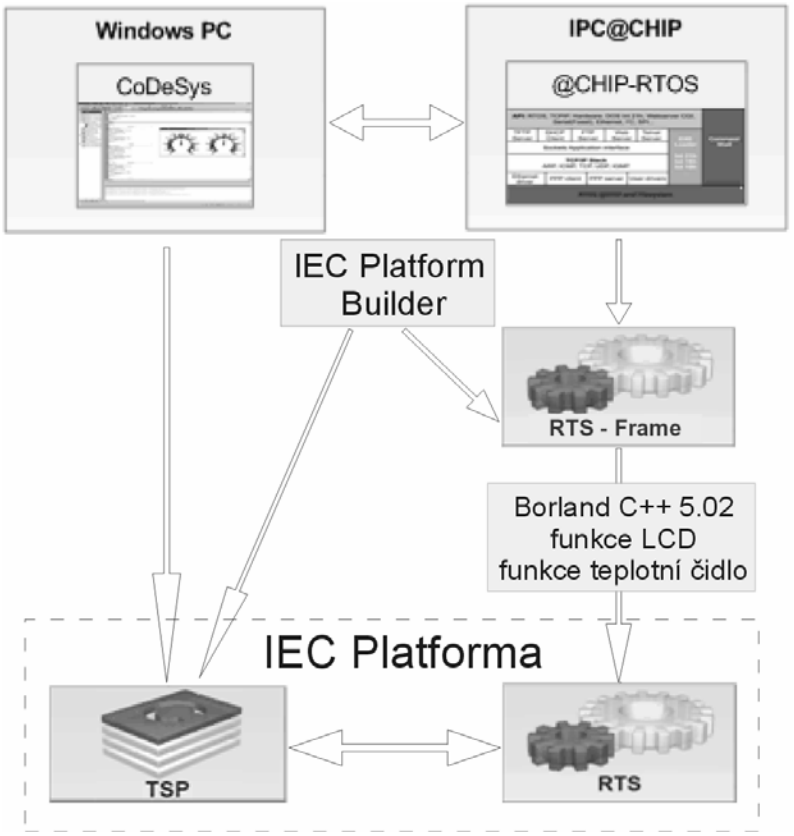
REPEAT	Iterační příkaz smyčka REPEAT	REPEAT
		j := j +1;
		UNTIL k < 15
		END_REPEAT;
			Vícenásobná smyčka s podmínkou ukončení smyčky na konci

EXIT	Ukončení smyčky	EXIT;	Předčasné ukončení iteračního příkazu
RETURN	Návrat	RETURN;	Opuštění právě vykonávané POU a návrat do volající POU
;	Prázdný příkaz	;	

3 Praktická část

3.1 Software

Pro vývoj aplikací pro PLC je nutné softwarové vybavení umožňující překlad kódu jazyků definovaných v IEC 61131-3. Takovýto software musí být uzpůsoben pro danou hardwarovou platformu PLC. Různorodost procesorů používaných pro PLC je velmi rozmanitá a sjednocení pod jeden systém je velmi obtížné. Firma 3S – Smart Software Solutions GmbH vyvinula softwarový nástroj CoDeSys, který je v maximální míře univerzální z hlediska aplikace jazyků definovaných v IEC 61131-3 do různého hardware. Univerzálnost je zabezpečena pomocí tzv. Targets, což jsou moduly portující výstupy z CoDeSysu do hardware. Pro svoje procesory SC13, SC123 a SC143 vytvořila firma BECK IPC GmbH nástroj SDK IEC Platform Builder, tvořící Target pro tyto procesory. Vazby mezi PC a PLC jsou znázorněny na obrázku 3.1 – 1.



Obr. 3.1 – 1 Softwarové vazby mezi PC a PLC

3.1.1 SDK IEC Platform Builder

Firma Beck IPC GmbH umožnila pomocí software SDK IEC Platform Builderu spojení svých procesorů SC13, SC123 a SC143 s vývojovým prostředím CoDeSys. Ten generuje TSP modul pro CoDeSys a RTS Frame pro procesor SC13.

V tomto software je nutné nastavit:

- Použití webové vizualizace (ANO)
- Typ použitého procesoru (SC13)
- Podpora REAL datového typu (ANO)
- Pracovní disk (B)
- Použití TCP/IP (port 1200)
- Použití RS232 (port 1)
- Rozsah paměťových oblastí
- Seznam použitých knihoven pro CoDeSys
- Vstupní a výstupní proměnné
- Poměr časů přidělených pro úlohu RTOSu a IEC

Výsledkem SDK Platform Builderu jsou adresáře TSP a RTS. V adresáři TSP jsou soubory nutné pro implementaci hardware do CoDeSysu a soubor install.bat, který po spuštění vytvoří „Target“ pro CoDeSys. V adresáři RTS jsou soubory nutné pro vytvoření souboru myrts.exe, který bude spouštěn v procesoru SC13. Je to soubor myrts.c plus knihovny nutné k překladu. V překladači Borland C/C++ je nutné soubor myrts.c editovat a dopsat funkce pro obsluhu hardware. V našem případě to jsou funkce pro zápis a čtení z displaye a čtení z AD7416 (teplotní čidlo). Jsou to tyto funkce:

- ReadTemp(void) – vyčtení teploty z AD7416
- LCD_ReadByte(unsigned int rs) – čtení byte z LCD
- LCD_WriteByte(unsigned int rs, unsigned char value) zápis byte na LCD

- LCD_Init(void) – inicializace LCD
- LCD_WriteStr(char *str) – zápis řetězce na LCD
- LCD_PosXY(int x, int y) – posun kurzoru na pozici X,Y

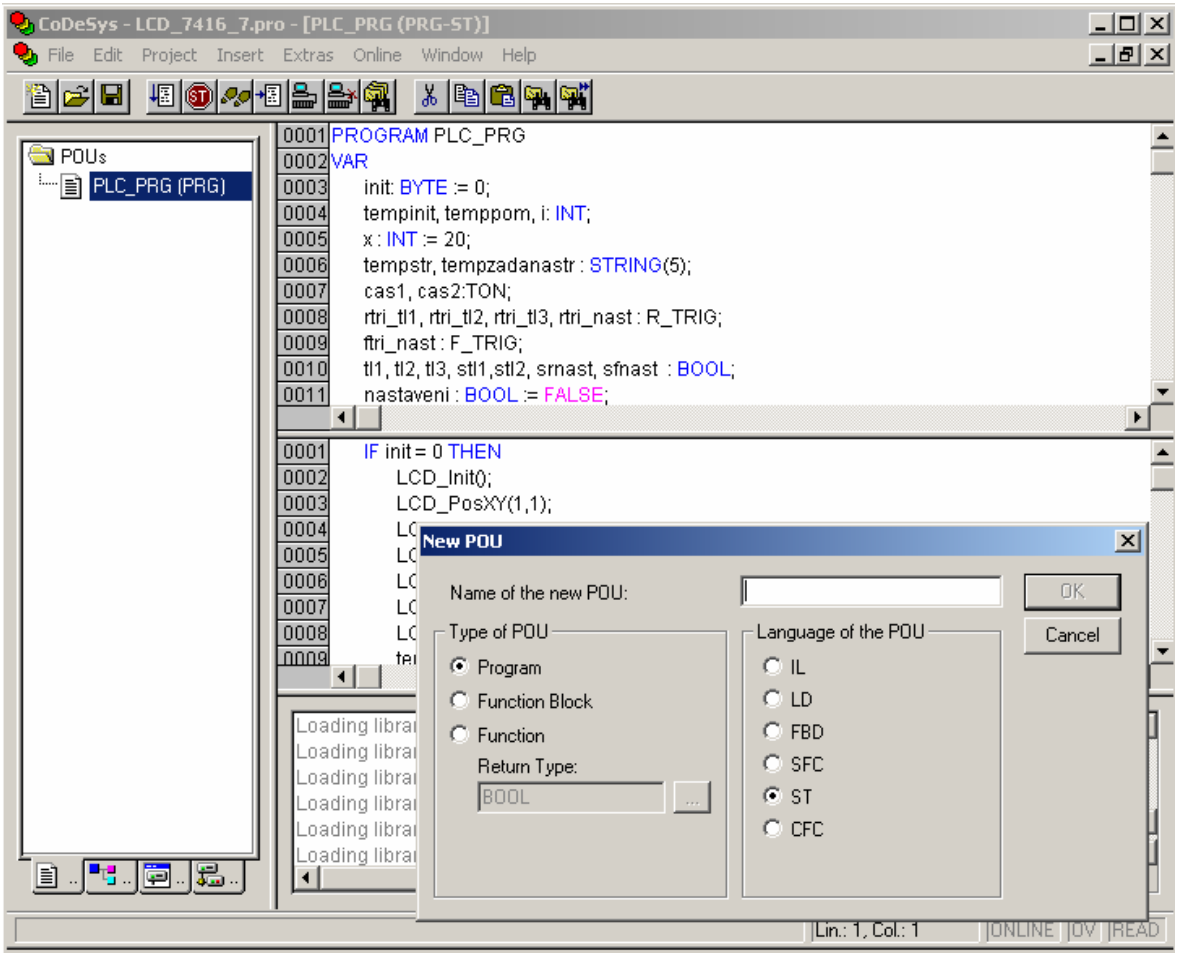
Těla funkcí jsou v příloze a na přiloženém CD.

3.1.2 CoDeSys

Pro vývoj software pro PLC dle IEC 61131 je dostupný software CoDeSys (Controlled Development System) od německé firmy 3S – Smart Software Solutions GmbH. Lze ho volně stáhnout na webových stránkách výrobce. Je to jedno z nejvýkonnějších vývojových prostředí - robusní nástroj umožňující programování ve všech standardizovaných jazycích dle IEC a obsahující všechny potřebné komponenty pro implementaci do PLC. Umožňuje pohodlné ladění vyvíjené aplikace pomocí krokování, vkládání breakpointů a při běhu software v PLC zobrazuje aktuální hodnoty všech proměnných. Na obr. 3.4 – 2 je základní obrazovka software CoDeSys se zobrazením vkládání nové POU. Tento software je schopný spolupracovat s množstvím různých hardwarových platforem a procesorů (na straně PLC). Jsou to:

- ARM
- PowerPC
- 68xxx
- 8051
- 80x86/Pentium
- 80C16x
- Hitachi SH 2/3/4
- Hitachi H8
- Motorola ColdFire
- 80186
- Infineon TriCore
- Texas Instrument DSP TMS32028x

CoDeSys kombinuje možnosti vyšších programovacích jazyků jako C a Pascal. Umožňuje jednoduchou manipulaci a funkčnost programovacího systému PLC. Kompletní balíček obsahuje manuál a online pomoc v návaznosti na programovací systém. Je dostupný v angličtině, francouzštině a němčině. Runtime systém, programovací systém a generátor kódu jsou důsledně propojeny pro minimalizaci adaptačního času a nákladů. Praktické a přátelské prostředí urychluje učení se systémem. Funkce jako autodeklarace, autoformát a kontextová nápověda zjednodušuje použitelnost CoDeSysu. Všechny funkce mohou být iniciovány pomocí klávesnice. Přirozená tvorba kódu pro všechny obvyklé procesory garantuje optimální využití kontrolního systému. Inteligentní Algoritmus umožňuje překlad ve velmi krátkém čase i velkým projektům s tisícem globálních proměnných. Spojení mezi CoDeSysem a IEC platformou slouží TSP (Target Support Package). Tato aplikace zajistí přístup mezi CoDeSysem jako vývojovým prostředím PLC a platformou IEC. V TSP jsou obsaženy hardwarové možnosti PLC dle vyvíjené aplikace, tudíž už při projektování aplikace je nutné znát počet vstupních a výstupních proměnných, rozsah paměťových adres, zda daná aplikace bude využívat webové vizualizace a jaké další periferie budou používané v aplikaci. Na straně modulu DK50, resp. procesoru SC13 běží operační systém RTOS, dodávaný firmou Beck GmbH. Na tomto operačním systému je nutné spustit RTS (Run Time System) který spolupracuje s platformou IEC. RTS je nutné upravit pro danou aplikaci, tj vepsat všechny potřebné funkce pro obsluhu periferií, v našem případě zápis a čtení z LCD displaye a čtení z teplotního senzoru.



Obr. 3.4 – 2 Základní obrazovka software CoDeSys

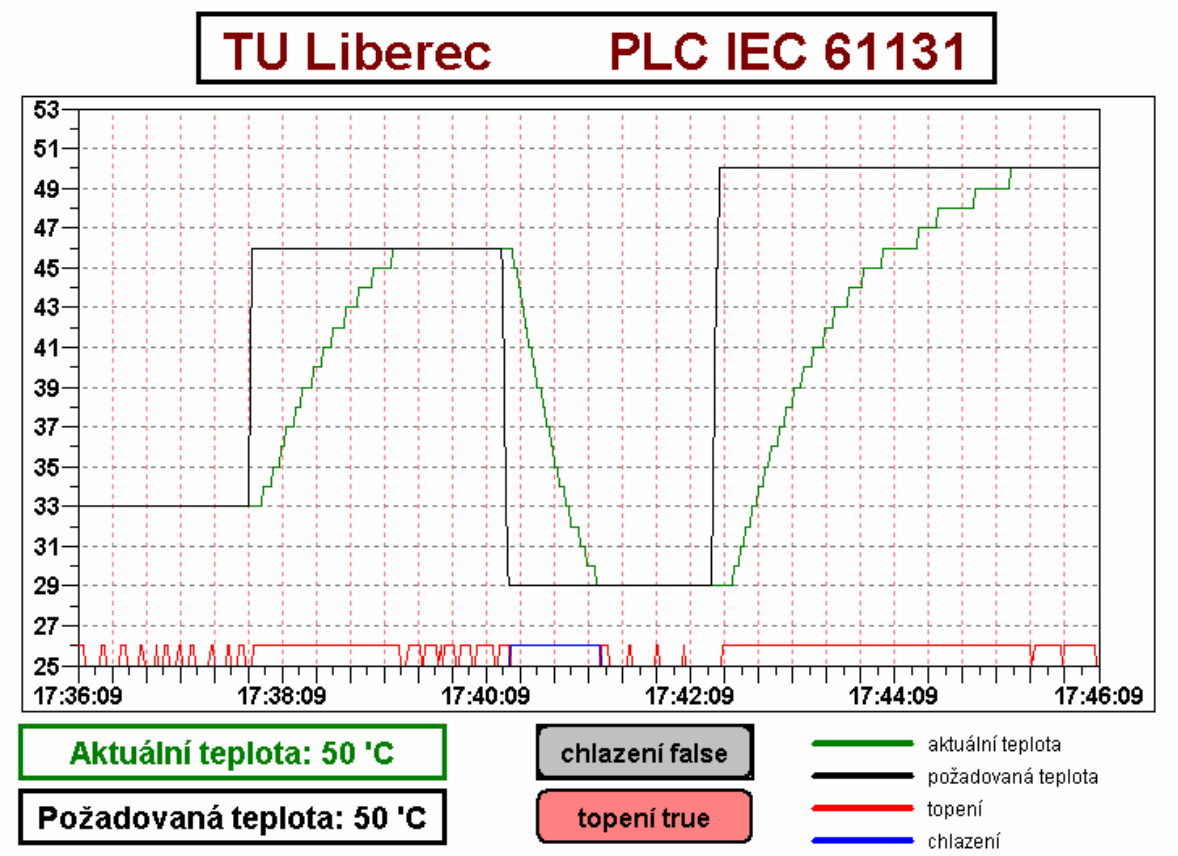
3.1.3 Struktura projektu v CoDeSysu

Celý projekt se ukládá do souboru s příponou *.pro. První vytvořená POU se automaticky jmenuje PLC_PRG (lze ji přejmenovat). Proces, který je pak spouštěn v samotném PLC začíná právě v této základní POU. Další POU (programy, funkční bloky, funkce) jsou přístupné právě z této základní POU. Při tvoření nového projektu je nejprve nutné vybrat „Target“. Vytvoření „Target“ je popsáno v kapitole 3.1 - 1. Potom se vytvoří první programová POU (v jakémkoli jazyce definovaném v IEC 61131-3) a pak eventuelně další POU i v jiných jazycích, než první programová POU. Pro ladění je možné využít simulační mód, pomocí něhož se nasimuluje PLC a nahraje se do něj testovaný projekt. Po spuštění lze sledovat běh programu, měnit manuálně vstupní hodnoty a sledovat jak

lokální, tak výstupní proměnné, jestli odpovídají očekávaným hodnotám. Lze též využít vkládání tzv. breakpointů, na kterých se běh programu zastaví a mohou se kontrolovat hodnoty proměnných, které by nebylo možno kontrolovat za běhu. Dále je možno využít k ladění projektu krokování. Všechny tyto nástroje zefektivňují vývoj požadovaného projektu ke splnění zadané úlohy.

3.1.4 Vizualizace

CoDeSys umožňuje velmi efektivně vytvořit web-vizualizaci. Musí ho samozřejmě podporovat námi programovaný PLC a musí být aktivován již při tvoření „Target“ v IEC Platform Builderu. CoDeSys generuje popis v XML formátu z vizualizačních dat které se nahrají do PLC společně s JAVA-Appletem. Tato data mohou být zobrazena přes TCP/IP a webový prohlížeč. Tím je vizualizace přístupná na jakékoli platformě umožňující zobrazování HTML a Javy. Vytvořená vizualizace k ukázkové aplikaci je patrná na obrázku 3.4 – 3. Je zde vidět změna požadované teploty na tři různé hodnoty a průběh regulované veličiny, dále zapínání topení a chlazení.

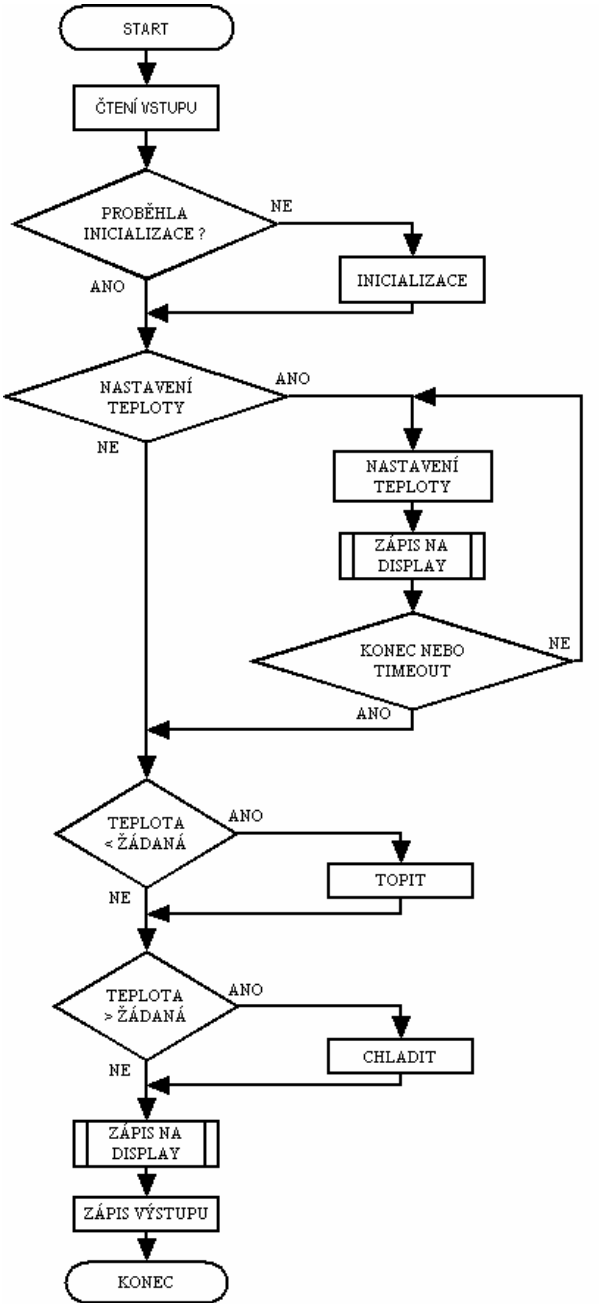


Obr. 3.4 – 3 Ukázka vizualizace ve webovém prohlížeči řešené úlohy

3.1.5 Program regulace teploty

Program, který demonstruje regulaci teploty systému pomocí teplotního čidla AD7416, rezistoru (jako zdroje tepla) a větráčku (pro chlazení) byl napsán v jazyce ST. Regulace je jednoduchá třístavová. Soustava se buď ohřívá rezistorem, chladí větráčkem nebo není ovlivněna. Důsledkem této regulace je kolísání teploty okolo požadované hodnoty. Hystereze pro ohřev je nastavena pouze na 0,2 °C. Díky tak malé hysterezi je frekvence spínání vysoká. To není na závadu, protože spínání proudu do ohřívacího rezistoru je pomocí tranzistoru a nehrozí opotřebení mechanických části jako např. u relé. Kolísání teploty je proto minimální. U chlazení je hystereze nastavena na 0,4 °C. Větší hodnota hystereze u chlazení než u topení je právě z důvodu použití mechanického větráčku, kde by mohlo být časté spínání na závadu. Program umožňuje nastavení požadované hodnoty v rozmezí 27-53 °C. Nastavení se provádí stiskem nastavovacího

tlačítka SET a pomocí tlačítek ↑ a ↓ se nastaví požadovaná hodnota. Pro opuštění menu nastavení se opět použije tlačítko SET nebo se program automaticky vrátí do režimu řízení, pokud není stisknuto jakékoli tlačítko po dobu 5 sekund. Celý program je napsán zhruba na 130 řádcích. Z toho je patrné, jak efektivně lze řešit regulační úlohu pomocí jazyka ST. Vývojový diagram je na obrázku 3.1 - 1. Celý kód je v příloze a na připojeném CD.



Obr. 3.1 – 1 vývojový diagram software PLC

3.2 hardware

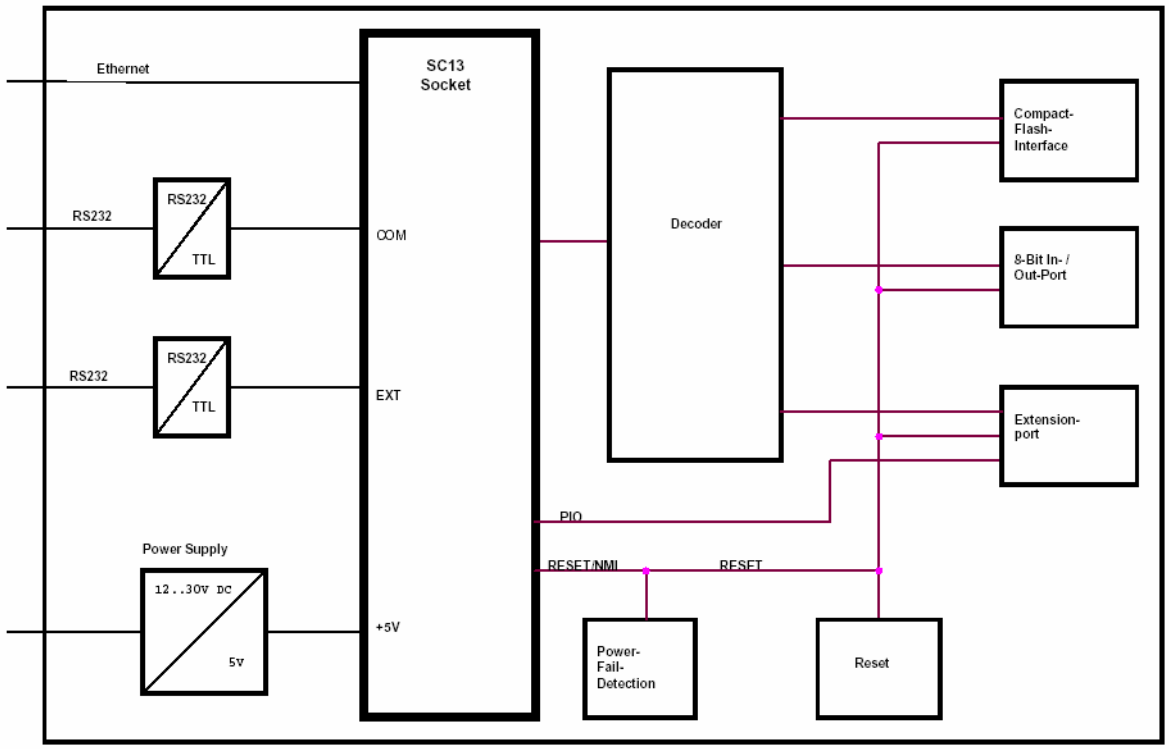
Pro realizaci byl vybrán kit DK51 od německé firmy BECK IPC GmbH. Ten obsahuje modul DK50, potřebnou kabeláž a síťový adaptér. Dále jsou zde instalační CD s Borland C++ Development Suite a softwary firmy BECK pro vývoj aplikací. Modul DK50 obsahuje embedded procesor SC13 a další podpůrné obvody:

- Ethernet rozhraní, ethernet transformátor, LED indikující přenos, konektor RJ45
- 2xRS232 rozhraní, převodník z úrovní TTL na RS232
- Napájecí zdroj DC/DC převodník z 12-30V na 5V
- Detektor chyb v napájení
- Resetovací tlačítko
- Konektor pro CF (compact flash) rozšiřující paměť
- 8 digitálních TTL vstupů
- 8 digitálních výstupů TTL s indikačními LED

Specifikace procesoru SC13 je v následující tabulce:

Procesor	SC186/40 MHz
Paměť RAM	512 KB
Paměť Flash	512 KB
Seriál	2x TTL, 4-wire with DMA
Ethernet	10/100BaseT
I/O	14 PIO, Intel AD-Bus
Napájení/spotřeba	5V/300 mA (typicky)
Operační systém	RTOS
Pouzdro	DIL32 (22 x 44 x 9,5 mm)

Funkci modulu DK50 přibližuje následující blokové schéma:



Obr. 3.2 – 2 Blokové schéma modulu DK50

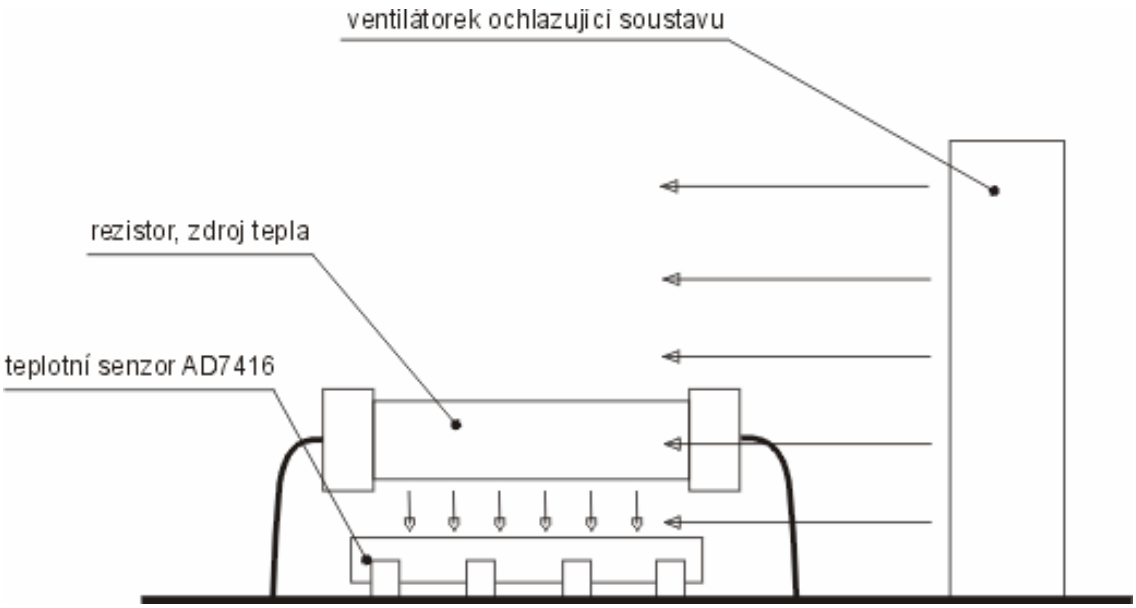
Procesor obsahuje API (Application Programmer Interface) s těmito funkcemi:

- RTOS – operační systém na bázi DOSu, umožňující spouštět až 35 úloh, souborový systém pro interní ramdisk, interní flashdisk a externí disk.
- TCP/IP rozhraní s http webserverem, FTP serverem, telnetem, DHCP klientem, TFTP serverem, SNMP podporou a UDP
- Možností spouštění 12 DOSových aplikací
- Příkazový řádek dostupný přes telnet
- Podpora I2C
- Podpora sériové komunikace
- Webserver CGI

Dále je k procesoru dodávaná C knihovna. Ta umožňuje lépe využít hardware procesoru SC13 při programování aplikací v jazyce C.

3.3 Regulovaná soustava

Pro ukázkovou aplikaci byl zvolen systém regulace teploty. Regulovaná soustava se skládá z teplotního čidla, rezistoru jako zdroje tepla, větráčku pro chlazení. Ten spočívá v ohřívání měřícího teplotního čipu rezistorem nebo chlazení větráčkem. Dále je použit další rezistor pro ohřívání čipu jako simulace neměřené poruchy. Soustava je na následujícím obrázku:



Obr. 3.3 – 1 Schéma regulované soustavy

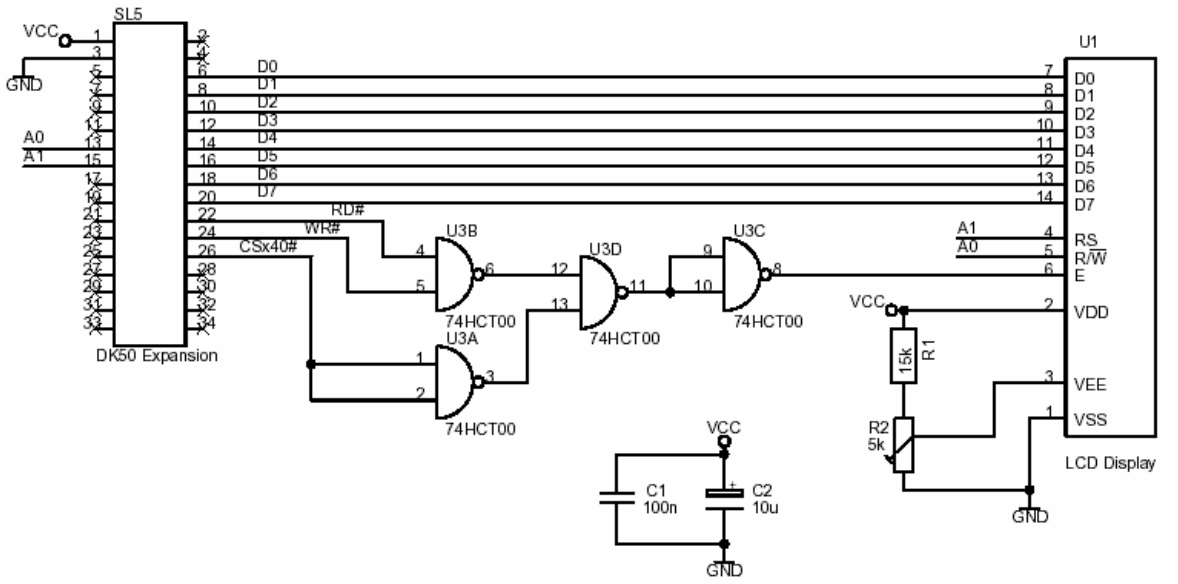
3.4 Periferie

3.4.1 Display

Pro zobrazování byl zvolen LCD display EL2004A. Je to 4 řádkový display s 20 znaky na řádek s možností podsvícení. Display má 8 bitovou paralelní sběrnici a další signály popsané v následující tabulce:

Číslo vývodu	Signál	úroveň	Poznámka
1	VSS	0V	Zem napájení
2	VDD	5V ± 5%	Napájecí napětí
3	VEE	0-5V	Řízení kontrastu display
4	RS	TTL	Řídící registr H-data, L-řídící kódy
5	R/W	TTL	Čtení / zápis
6	E	TTL	Povolovací signál
7	DB0	TTL	Data bit 0
8	DB1	TTL	Data bit 1
9	DB2	TTL	Data bit 2
10	DB3	TTL	Data bit 3
11	DB4	TTL	Data bit 4
12	DB5	TTL	Data bit 5
13	DB6	TTL	Data bit 6
14	DB7	TTL	Data bit 7
15	LED+	4,2V	Napájení podsvícení
16	LED-	0	Zem podsvícení

Připojení display EL2004A k modulu DK50 je na následujícím elektronickém schématu:



Obr. 3.4 – 1 Schéma připojení displaye El2004A k DK50

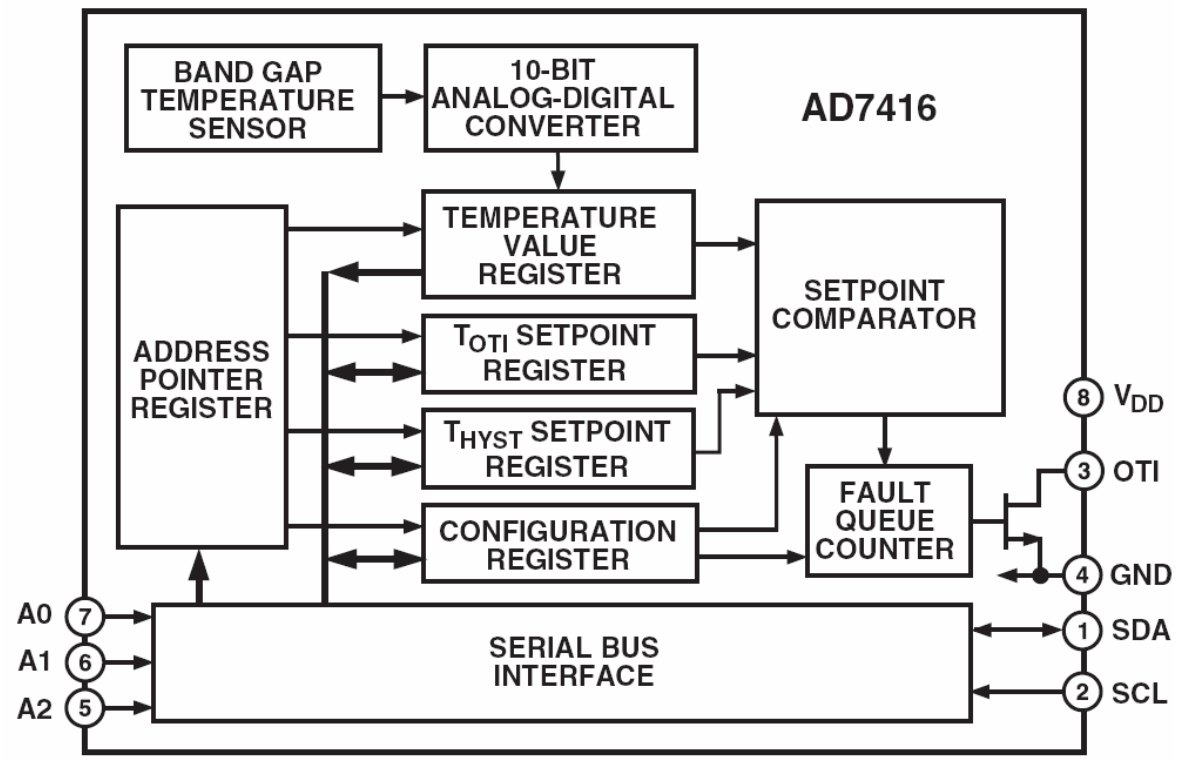
3.4.2 Teplotní čidlo

Pro měření teploty byl vybrán integrovaný obvod AD7416 firmy Analog Device. Charakteristické rysy obvodu:

- 10 bitový A/D převodník
- Teplotní čidlo integrované na čipu obvodu s rozsahem měření -40°C až +125°C

- Široký rozsah napájení 2,7 – 5,5V
- I2C sériová sběrnice
- Chyba měření ±2°C
- Rozlišení 0,25°C

Blokové schéma integrovaného obvodu AD7416 je na obrázku 3.4-2.



Obr. 3.4-2 blokové schéma AD7416

Obvod je primárně určen pro automatizaci, průmyslové aplikace a pro osobní počítače. I2C adresa obvodu AD7416 je | 1 | 0 | 0 | 1 | A2 | A1 | A0 | R/W | binárně. V našem případě jsou adresové vodiče A2, A1, A0 zapojeny na logickou úroveň High, adresa je tedy 9E hexadecimálně. Teplota je uložena v 16 bitovém registru. Významových je pouze 10 bitů. Vyčítání teploty z obvodu je na adrese 9E hexadecimálně dvou bytů po sobě. Význam jednotlivých bitů je v tabulce na obrázku 3.4 -3. Garantovaný teplotní rozsah měření je od -40°C do +125°C. Formát teploty v 10 bitech je v tabule na obrázku 3.4 – 4.

BYTE 1								BYTE 0							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
B9	B8	B7	B6	B5	B4	B3	B2	B1	B0	Bez významu					

Obr. 3.4 – 3 Formát teploty v 16bitovém registru

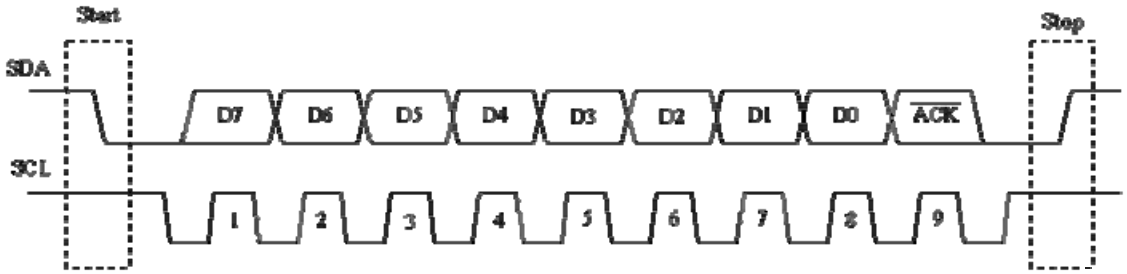
Teplota	Digitální výstup									
	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
-128 °C	1	0	0	0	0	0	0	0	0	0
-125°C	1	0	0	0	0	0	1	1	0	0
-100°C	1	0	0	1	1	1	0	0	0	0
-75°C	1	0	1	1	0	1	0	1	0	0
-50°C	1	1	0	0	1	1	1	0	0	0
-25°C	1	1	1	0	0	1	1	1	0	0
-0,25°C	1	1	1	1	1	1	1	1	1	1
0°C	0	0	0	0	0	0	0	0	0	0
+0,25°C	0	0	0	0	0	0	0	0	0	1
+10°C	0	0	0	0	1	0	1	0	0	0
+25°C	0	0	0	1	1	0	0	1	0	0
+50°C	0	0	1	1	0	0	1	0	0	0
+75°C	0	1	0	0	1	0	1	1	0	0
+100°C	0	1	1	0	0	1	0	0	0	0
+125°C	0	1	1	1	1	1	0	1	0	0
+127°C	0	1	1	1	1	1	1	1	0	0

Obr. 3.4 – 4 Datový formát teploty

3.4.2.1 I2C

I2C sběrnice je produktem firmy Philips, který představila před více jak 20 lety. Používá se pro komunikaci dvou až 128 zařízení (7 bitů adresa, 1 bit čtení/zápis, nově se zavádí 10 bitová adresa, tj. 1024 zařízení) v režimech Master / Slave. Rychlost sběrnice je 100kb/s – standardní mód, 400kb/s rychlý mód 3,4Mb/s ultra rychlý mód. Jde o prostředek pro obousměrnou komunikaci mezi různými integrovanými obvody a moduly, jako např. vstupně výstupní expandery, A/D a D/A převodníky, řadiče displayů, senzory teploty, napěťové a kmitočtové syntézy, hodiny reálného času, paměti a další. Vodič, jimž jsou přenášena data, bývá označován SDA (Serial Data), vodič, přenášející hodinový signál se označuje SCL (Serial Clock). Přenos dat může být zahájen pouze pokud na

sběrnici neprobíhá žádná komunikace. Obvody mají pevnou část adresy 4 bity SA0-SA4 a proměnnou 3 bity DA0-DA2, které umožňují nastavit různou adresu u stejných typů obvodů. Díky tomu můžeme zapojit 8 stejných obvodů na jednu sběrnici. Potvrzení příjmu dat probíhá po každém přeneseném bytu jedním bitem. Tento bit vysílá přijímač Slave. Master pro ten účel generuje jeden hodinový puls a vysílač musí zajistit během něho na SDA úroveň H, aby mohl úroveň na vodiči SDA přečíst. A to všechno i v případě, že je masterem (vysílačem SCL) modul přijímače. Potvrzovací signál na vodiči SDA musí mít úroveň L po celou dobu, kdy je hodinový puls, vsouvaný pro přenos potvrzovacího pulsu, na úrovni H. MASTER přijímač (generující hodinové pulsy) musí generovat potvrzovací signál za každým přeneseným bytem dat, s výjimkou posledního bytu. Aby označil konec přenášení dat slave-vysílači, master nepotvrzuje poslední byte zprávy, kterou taktoval přijímač. Slave pak přestaví úroveň na vodiči SDA na H, aby umožnil masteru, kterým je v tomto případě přijímač, generovat signál STOP. Master může také generovat signál STOP během potvrzovacího signálu a řádně tak ukončit přenos dat.

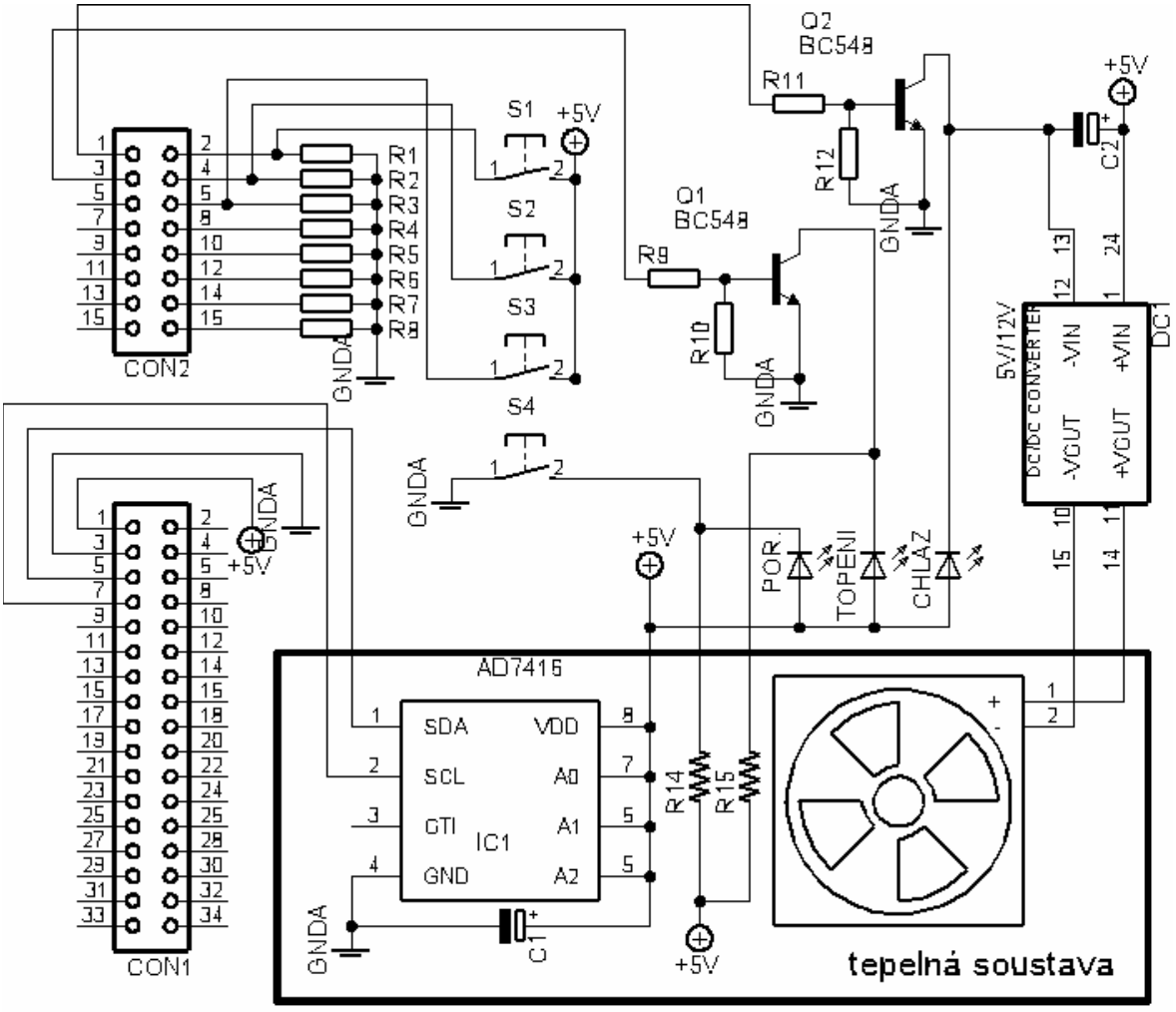


Obr. 3.4 – 5 Ukázka komunikace I2C s označením start a stop sekvence

Obr. 3.4 – 6 Ukázka I2C navazující komunikace více bytů

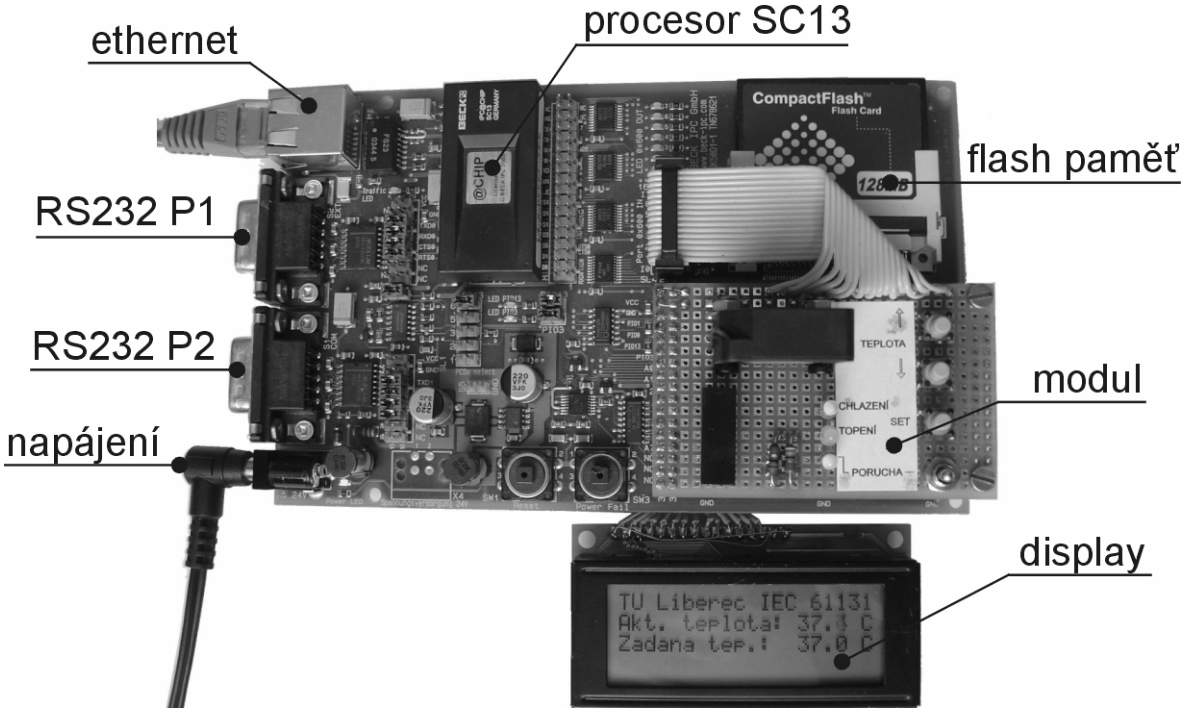
3.4.3 Tlačítka, simulace topení, větráček

Modul DK50 má 8 binárních vstupů a 8 binárních výstupů. 3 vstupy byly použity pro 3 tlačítka. Jedno tlačítko slouží pro vstup do nastavovacího menu, dvě pro nastavování požadované teploty. 2 výstupy byly zesíleny pomocí diskrétních bipolárních tranzistorů. K jednomu výstupu byl připojen rezistor jako zdroj tepla a ke druhému přes konvertor 5VDC/12VDC byl připojen větráček pro chlazení soustavy. DC/DC konvertor byl použit z důvodu nedostupnosti 12V napájení na modulu DK50. Pro simulaci neměřené poruchy je přidán ještě jeden rezistor jako další zdroj tepla. Elektrické schéma připojení systému k modulu DK50 je na obrázku 3.4 – 7.



Obr. 3.4 – 7 Elektrické schéma připojení regulované soustavy k modulu DK50

Celkový pohled na modul DK50 s připojeným displayem, modulem simulace teplotního řízení je na následujícím obrázku:



Obr. 3.4 – 8 Modul DK50 s připojenými periferiemi

4 Závěr

Výsledek diplomové práce je funkční PLC schopné řídit ukázkovou soustavu. Soustava je tvořena teplotním čidlem, zdrojem tepla v podobě rezistoru a větráčkem pro chlazení soustavy. Soustava je relativně jednoduchá, s malou tepelnou kapacitou a rychlou odezvou. Proto postačuje pro řízení dvoustavový, resp. třístavový regulátor. Pro komunikaci s uživatelem je k PLC připojen displej pro zobrazení aktuální teploty soustavy a zobrazování nastavení požadované teploty. K zadávání požadované teploty a ke vstupu do nastavovacího režimu slouží tři tlačítka. Dále je v PLC naprogramovaná vizualizace. Ta je v podobě JAVA Appletu. Pro zobrazení postačuje jakýkoli webový prohlížeč s Javou. Vizualizace zobrazuje číselně žádanou hodnotu a regulovanou veličinu a také tyto hodnoty zobrazuje v grafu v 10 minutovém časovém okně. Dále se zobrazuje stav akčních prvků a to zapnutí topení a zapnutí větráčku.

Při řešení bylo třeba řešit mnoho překážek. Jedny z nejdůležitějších byla připojení displaye EL4002A a integrovaného obvodu AD7416. Bylo nutné prostudovat jejich funkce a pochopit jejich komunikaci s okolím. Napsat funkce v jazyce C, které umožní komunikaci s vývojovým prostředím CoDeSys a tím umožní komunikaci s tímto systémem. Dále bylo nutné naučit se ovládat vývojové prostředí CoDeSys, nastudovat standard IEC 61131-3 a naučit se aspoň v potřebné míře programovací jazyky standardizované v této normě pro napsání ukázkové aplikace. Rád bych proto zde poděkoval Ing. Tomáši Víznerovi za podněty a znalosti, které umožnily realizaci této práce.

Další vylepšení při aplikaci v reálných podmínkách vidím v použití vhodnější regulace pro dané podmínky, např. PI nebo PID regulátoru. Dále je možné zlepšit vizualizaci např. pro zadávání žádané hodnoty. Potom by bylo možné ovládat a monitorovat řízený proces na dálku pomocí internetu.

5 Seznam literatury

[1] ĎAĎO,S.-KREIDL,M.: Měřicí převodníky fyzikálních veličin. ČVUT, Praha 1980.

[2] Šmejkal, L. Martinásková, M.: PLC a automatizace 1 - základní pojmy, úvod do programování, BEN – technická literatura, Praha 1999, ISBN 80-86056-58-9

[3] Martinásková, M Šmejkal, L.: Řízení programovatelnými automaty, ČVUT Praha 1998, ISBN 80-01-02925-5

[4] Martinásková, M Šmejkal, L.: Řízení programovatelnými automaty II, ČVUT Praha 2000, ISBN 80-01-02096-7

[5] Martinásková, M Šmejkal, L.: Řízení programovatelnými automaty II, ČVUT Praha 2000, ISBN 80-01-02096-7

[6] M Šmejkal, L.: Programovací jazyk ST pro PLC Tecomat, TXV 003 21.01 2005

[7] Analog Devices: Datasheet AD7416, 2004

[8] USER MANUAL FOR PLC Programming with CoDeSys 2.3, 3S – Smart Software Solutions GmbH 2005

[9] The CoDeSys Visualization, Suplement to the User Manual for PLC Programming with CoDeSys 2.3, 3S – Smart Software Solutions GmbH 2005

[8] CoDeSys@CHIP-SDK Manual, Easy Creating IEC 61131-3 compatible IPC@CHIP board controllers, Beck IPC GmbH 2005

6 Seznam příloh

Funkce komunikace LCD a AD7416 s CoDeSysem

```
/*****************************************************************************
* LCD Defines
*****************************************************************************/
#define LCD_BASE 0x640 // LCD base adress  0x140~pcs1 0x640~pcs6 flash pouz.pcs6

/*****************************************************************************
* LCD Prototypes + 7416
*****************************************************************************/
int ReadTemp(void);
int LCD_Init(void);
int LCD_ReadByte(unsigned int rs);
int LCD_WriteByte(unsigned int rs, unsigned char  value);
int LCD_PosXY(int x, int y);
int LCD_WriteStr(char *str);

/*****************************************************************************
* Functions
*****************************************************************************/
/*****************************************************************************
* ReadTemp AD7416
*****************************************************************************/
int ReadTemp(void)
{
  int byte0=0;
  int byte1=0;
  int temp=0;

  I2C_receive_char (0x9E, &byte0, 1);//pevne nastavena adr 9Ehex pro A0,A1,A2=1
  temp = byte0;
  temp <<= 2;            // *4 v CoDeSysu se musi vydelit 4
  I2C_receive_char (0x9E, &byte1, 0); // je to proto, aby se preneslo pomoci INT
  byte1 >>= 6;                        // rozliseni 0.25C * 4 = cele cislo
  temp += byte1;
  I2C_release();
  return(temp);
}

/*****************************************************************************
* LCD_ReadByte
*****************************************************************************/
int LCD_ReadByte(unsigned int rs)
{
  // read byte LCD
  // LCD read: A0 = 1
  // A1 = register select
  int value=0;

  rs = (((rs & 0x0001) << 1) | 0x0001);
  value=inp (LCD_BASE+rs);
  return(value);
}

/*****************************************************************************
* LCD_WriteByte
*****************************************************************************/
int LCD_WriteByte(unsigned int rs, unsigned char  value)
{
  // write byte to the LCD
  // write: A0 = 0
  // A1 = segister select
  // rs = segister select: 1=displaydata / 0=command
  // value = write byte
  do {} while ((LCD_ReadByte(0) & 0x0080) == 0x0080); // kontrola pripraven
  rs = (rs & 0x0001) << 1;
  outp(LCD_BASE+rs,value);
  return(0);
}

/*****************************************************************************
* LCD_Init
*****************************************************************************/
int LCD_Init(void)
{
  pfe_enable_bus(0xFF,1);
  pfe_enable_pcs(LCD_BASE >> 8);

  LCD_WriteByte(0, 0x01); // smazani displaye,
  LCD_WriteByte(0, 0x02); // cursor na zacatek
  LCD_WriteByte(0, 0x06); // mod nastaveni
  LCD_WriteByte(0, 0x0C); // display on/off
  LCD_WriteByte(0, 0x38); // funkce nastaveni
}
```

```
/*****************************************************************************
* LCD_WriteStr
*****************************************************************************/
int LCD_WriteStr(char *str)
{
  while (*str!=0)
  {
    LCD_WriteByte(1,*(str++));
  }
  return(0);
}

/*****************************************************************************
* LCD_PosXY
*****************************************************************************/
int LCD_PosXY(int x, int y)
{
  // nastavi kurzor na X,Y
  // pro 4 radkovy display, 20 znaku na radek
  // radek 1:  x
  // radek 3:  0x40 + x
  // radek 2:  20 + x
  // radek 4:  0x40 + 20 + x
  // x = 1..20
  // y = 1..4
  const int X_MAX = 20; // max
  int pos=0;
  x--;
  switch (y)
  {
      case 1:  pos=x;
               break;
      case 2:  pos=0x40 + x;
               break;
      case 3:  pos=X_MAX + x;
               break;
      case 4:  pos=0x40 + X_MAX + x;
               break;
      default: pos=10;
               break;
  }
  LCD_WriteByte(0,0x80|pos);
  return(0);
}
```

Program PLC

```
PROGRAM PLC_PRG
VAR
	init: BYTE := 0;
	tempinit, temppom, i: INT;
	x : INT := 20;
	tempstr, tempzadanastr : STRING(5);
	cas1, cas2:TON;
	rtri_tl1, rtri_tl2, rtri_tl3, rtri_nast : R_TRIG;
	ftri_nast : F_TRIG;
	tl1, tl2, tl3, stl1,stl2, srnast, sfnast  : BOOL;
	nastaveni : BOOL := FALSE;
	tempmin: INT := 27;
	tempmax : INT := 53;
	tempread: REAL;
END_VAR
(* @END_DECLARATION := '0' *)
	IF init = 0 THEN
		LCD_Init();
		LCD_PosXY(1,1);
		LCD_WriteStr('TU Liberec IEC 61131');
		LCD_PosXY(1,2);
		LCD_WriteStr('Akt. teplota:');
		LCD_PosXY(1,3);
		LCD_WriteStr('Zadana tep.:');
		tempzadanastr := INT_TO_STRING(tempzadana);
		LCD_PosXY(15,3);
		LCD_WriteStr(tempzadanastr);
		LCD_PosXY(17,3);
		LCD_WriteStr('.0 C');
		init:=1;
	END_IF;
	temppom := ReadTemp();
	IF i < x THEN
		tempinit := tempinit+temppom;
		i:=i+1;
	ELSE
		temp:=(INT_TO_REAL(tempinit))/x/4;
		tempstr := REAL_TO_STRING(temp);
		tempint := REAL_TO_INT(temp-0.5);
		tempinit:=0;
		i:=0;
	END_IF;

	tl1:=BYTE_TO_BOOL(D_in AND 2#1);
	tl2:=BYTE_TO_BOOL(D_in AND 2#10);
	tl3:=BYTE_TO_BOOL(D_in AND 2#100);

	rtri_tl3(CLK:=tl3);

	IF rtri_tl3.Q THEN
		nastaveni := NOT nastaveni;
	END_IF;

		IF nastaveni THEN                    (*nastaveni*)
			rtri_tl1(CLK:=tl1);
			rtri_tl2(CLK:=tl2);
			tempzadanastr:=INT_TO_STRING(tempzadana);
			LCD_PosXY(8,4);
			LCD_WriteStr(tempzadanastr);
			LCD_PosXY(10,4);
			LCD_WriteStr('.0 C');
			IF  rtri_tl1.Q THEN
				tempzadana:=tempzadana+1;
				IF tempzadana > tempmax THEN
					tempzadana := tempmax;
					LCD_PosXY(15,4);
					LCD_WriteStr('MAX !');
				ELSE
					LCD_PosXY(15,4);
					LCD_WriteStr('     ');
				END_IF;
				cas2(IN:=FALSE);
			END_IF;
			IF  rtri_tl2.Q THEN
				tempzadana:=tempzadana-1;
				IF tempzadana < tempmin THEN
					tempzadana := tempmin;
					LCD_PosXY(15,4);
```

```
                                LCD_WriteStr('MIN !');
                        ELSE
                                LCD_PosXY(15,4);
                                LCD_WriteStr('     ');
                        END_IF;
                        cas2(IN:=FALSE);
                END_IF;

                cas2(PT:= T#5s, IN:=TRUE);
                IF cas2.Q THEN
                        nastaveni := FALSE;
                END_IF;
        ELSE                    (* cteni teploty a regulace*)
                cas2(IN:=FALSE);
                LCD_PosXY(15,2);
                LCD_WriteStr(tempstr);
                LCD_PosXY(19,2);
                LCD_WriteStr(' C');
                LCD_PosXY(15,3);
                LCD_WriteStr(tempzadanastr);
                LCD_PosXY(17,3);
                LCD_WriteStr('.0 C');
                IF (temp+0.1)<tempzadana THEN
                        warm := TRUE;
                END_IF;
                IF (temp+0.0)>tempzadana THEN
                        warm := FALSE;
                END_IF;
                IF (temp-0.3) > tempzadana THEN
                        cold := TRUE;
                END_IF;
                IF (temp-0.2) < tempzadana THEN
                        cold := FALSE;
                END_IF;

        END_IF;

    rtri_nast(CLK:=nastaveni);
    srnast :=rtri_nast.Q;
    ftri_nast(CLK:=nastaveni);
    sfnast :=ftri_nast.Q;

    IF srnast  THEN
            LCD_Init();
            LCD_PosXY(1,1);
            LCD_WriteStr('nastav pozadovanou');
            LCD_PosXY(1,2);
            LCD_WriteStr('teplotu pomoci sipek');
    END_IF
    IF sfnast  THEN
            LCD_Init();
            LCD_PosXY(1,1);
            LCD_WriteStr('TU Liberec IEC 61131');
            LCD_PosXY(1,2);
            LCD_WriteStr('Akt. teplota:');
            LCD_PosXY(1,3);
            LCD_WriteStr('Zadana tep.:');
    END_IF

    D_out:=(BOOL_TO_BYTE(cold)+2*BOOL_TO_BYTE(warm));
    coldgrf:=BOOL_TO_INT(cold)+25;
    warmgrf:=BOOL_TO_INT(warm)+25;
END_PROGRAM
```