Norma pro řídicí systémy IEC 61 131

1. Úvod

Norma IEC 61 131 pro programovatelné řídící systémy má sedm základních částí a představuje souhrn požadavků na moderní řídící systémy. Je nezávislá na konkrétní organizaci či firmě a má širokou mezinárodní podporu. Jednotlivé části normy jsou věnovány jak technickému tak programovému vybavení těchto systémů.

V ČR byly přijaty jednotlivé části této normy pod následujícími čísly a názvy:

ČSN EN 61 131-1 Programovatelné řídicí jednotky - Část 1: Všeobecné informace
ČSN EN 61 131-2 Programovatelné řídicí jednotky - Část 2: Požadavky na zařízení a zkoušky
ČSN EN 61 131-3 Programovatelné řídicí jednotky - Část 3: Programovací jazyky
ČSN EN 61 131-4 Programovatelné řídicí jednotky - Část 4: Podpora uživatelů
ČSN EN 61 131-5 Programovatelné řídicí jednotky - Část 5: Komunikace
ČSN EN 61 131-7 Programovatelné řídicí jednotky - Část 7: Programování fuzzy řízení

V Evropské unii jsou tyto normy přijaty pod číslem EN IEC 61 131.

2. Základní myšlenky normy IEC 61 131-3

Norma IEC 61 131-3 je třetí částí skupiny norem řady IEC 61 131. Dělí se na dvě základní části:
- společné prvky
- programovací jazyky

2.1. Společné prvky

2.1.1. Typy dat

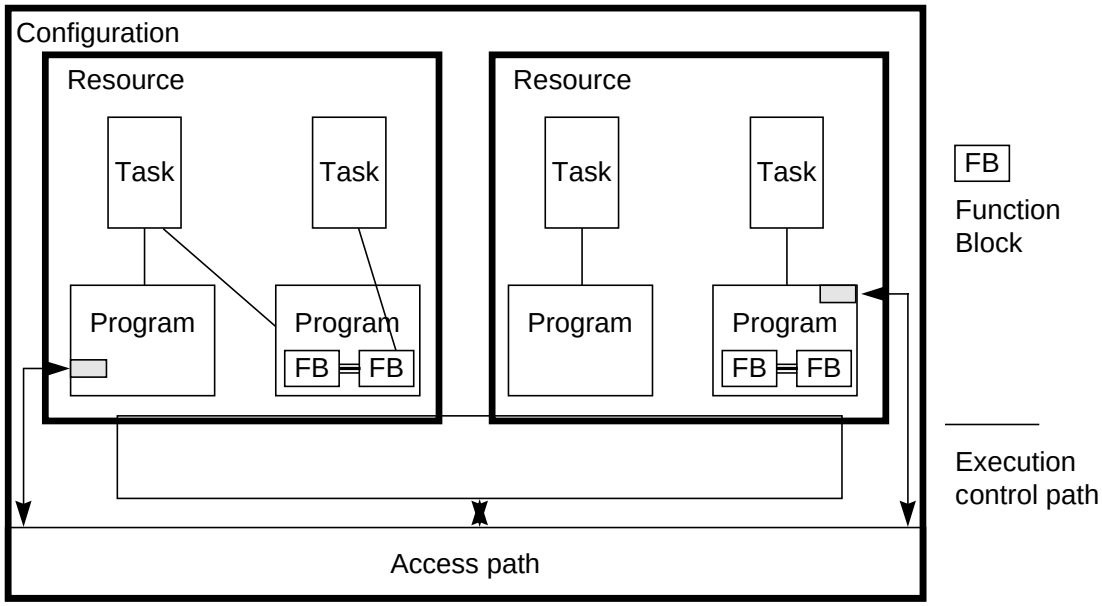
V rámci společných prvků jsou definovány typy dat. Definování datových typů napomáhá prevenci chyb v samém počátku tvorby projektu. Je nutné definovat typy všech použitých parametrů. Běžné datové typy jsou **BOOL**, **BYTE**, **WORD**, **INT** (Integer), **REAL**, **DATE**, **TIME**, **STRING** atd.
Z těchto základních datových typů je pak možné odvozovat vlastní uživatelské datové typy, tzv. odvozené datové typy. Tímto způsobem můžeme např. definovat jako samostatný datový typ analogový vstupní kanál a opakovaně ho používat pod definovaným jménem.

Proměnné
Proměnné mohou být přiřazeny explicitně k hardwarovým adresám (např. vstupům, výstupům) pouze v konfiguracích, zdrojích nebo programech. Tímto způsobem je dosaženo vysokého stupně hardwarové nezávislosti a možnosti opakovaného využití softwaru na různých hardwarových platformách.

Oblast působnosti proměnných je běžně omezena pouze na tu programovou organizační jednotku, ve které byly deklarovány (proměnné jsou v ní lokální). To znamená, že jejich jména mohou být používána v jiných částech bez omezení. Tímto opatřením dojde k eliminaci řady dalších chyb. Pokud mají mít proměnné globální působnost, např. v rámci celého projektu, pak musí být jako globální deklarovány (**VAR_GLOBAL**). Aby bylo možné správně nastavit počáteční stav procesu nebo stroje, může být parametrům přiřazena počáteční hodnota při startu nebo studeném restartu.

2.1.2 . Struktura programu (konfigurace, zdroje a úlohy)

Pro lepší pochopení může posloužit softwarový model, který je definován v normě a znázorněn na obrázku 1.



Obrázek 1 Softwarový model programu

***Konfigurace* (Configuration)**
Definuje nejvyšší úroveň celého softwarového řešení určitého problému řízení. Konfigurace je závislá na konkrétním řídicím systému, včetně uspořádání hardwaru, jako jsou například typy procesorových jednotek, paměťové oblasti přiřazené vstupním a výstupním kanálům a charakteristiky systémového programového vybavení (operačního systému).

***Zdroje* (Resource)**
V rámci konfigurace můžeme pak definovat jeden nebo více tzv. zdrojů. Na zdroj se můžeme dívat jako na nějaké zařízení, které je schopno vykonávat IEC programy.

***Úloha* (Task)**
Uvnitř zdroje můžeme definovat jednu nebo více tzv. *úloh* (Task). Úlohy řídí provádění souboru programů a/nebo funkčních bloků. Tyto jednotky mohou být prováděny buď periodicky nebo po vzniku speciální spouštěcí události, což může být např. změna proměnné.
Programy jsou vystavěny z řady různých softwarových prvků, které jsou zapsány v některém z jazyků definovaném v normě.
Často je program složen ze sítě funkcí a funkčních bloků, které jsou schopny si vyměňovat data. Funkce a funkční bloky jsou základní stavební kameny, které obsahují
datové struktury a algoritmus.

2.1.3. Programové organizační jednotky

Funkce, funkční bloky a programy jsou v rámci normy IEC 61 131 nazývány společně *programové organizační jednotky* (Program Organization Units, někdy se pro tento důležitý a často používaný pojem používá zkratka POUs).

Funkce
IEC 61 131-3 definuje standardní funkce a uživatelem definované funkce. Standardní funkce jsou např. **ADD** pro sčítání, **ABS** pro absolutní hodnotu, **SQRT** pro odmocninu, **SIN** pro sinus a **COS** pro cosinus. Jakmile jsou jednou definovány nové uživatelské funkce, mohou být používány opakovaně.

Funkční bloky

Na funkční bloky se můžeme dívat jako na integrované obvody, které reprezentují hardwarové řešení specializované řídicí funkce. Obsahují algoritmy i data, takže mohu zachovávat informaci o minulosti, (tím se liší od funkcí). Mají jasně definované rozhraní a skryté vnitřní proměnné, podobně jako integrovaný obvod nebo černá skříňka. Umožňují tím jednoznačně oddělit různé úrovně programátorů nebo obslužného personálu. Klasickými příklady funkčního bloku jsou např. regulační smyčka pro teplotu nebo PID regulátor.
Jakmile je jednou funkční blok definován, může být používán opakovaně v daném programu, nebo v jiném programu, nebo dokonce i v jiném projektu. Je tedy univerzální a mnohonásobně použitelný.
Funkční bloky mohou být zapsány v libovolném z jazyků definovaném v normě. Mohou být tedy plně definovány uživatelem. Odvozené funkční bloky jsou založeny na standardních funkčních blocích, ale v rámci pravidel normy je možno vytvářet i zcela nové zákaznické funkční bloky.
Interface funkcí a funkčních bloků je popsán stejným způsobem: Mezi deklarací označující název bloku a deklarací pro konec bloku je uveden soupis deklarací vstupních proměnných, výstupních proměnných a vlastní kód v tzv. těle bloku.

Programy
Na základě výše uvedených definic lze říci, že program je vlastně sítí funkcí a funkčních bloků. Program může být zapsán v libovolném z jazyků definovaných v normě.

2.2. Programovací jazyky
V rámci standardu je definováno pět programovacích jazyků. Jejich sémantika i syntaxe je přesně definována a neponechává žádný prostor pro nepřesné vyjadřování. Zvládnutím těchto jazyků se tak otevírá cesta k používání široké škály řídicích systémů, které jsou na tomto standardu založeny. Programovací jazyky se dělí do dvou základních kategorií:

Textové jazyky

IL - Instruction List - jazyk seznamu instrukcí
Textový jazyk nejvíce připomínající assembler

ST - Structured Text - jazyk strukturovaného textu
Je velmi výkonný vyšší programovací jazyk, který má kořeny ve známých jazycích Pascal a C. Obsahuje všechny podstatné prvky moderního programovacího jazyka, včetně větvení (**IF-THEN-ELSE** a **CASE OF**) a iterační smyčky (**FOR**, **WHILE** a **REPEAT**). Tyto prvky mohou být vnořovány. Tento jazyk je vynikajícím nástrojem pro definování komplexních funkčních bloků, které pak mohou být použity v jakémkoliv jiném programovacím jazyku.

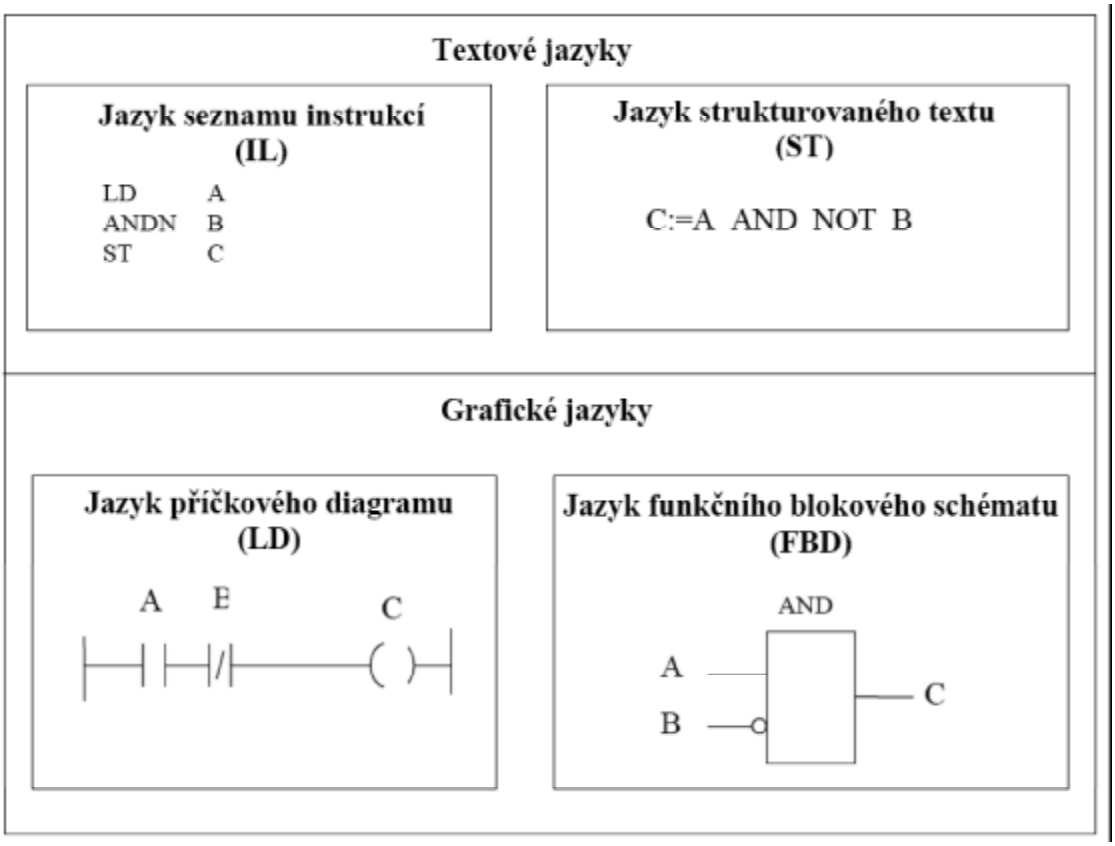
Grafické jazyky

LD - Ladder Diagram - jazyk příčkového diagramu (jazyk kontaktních schémat).
Má původ v USA. Je založen na grafické reprezentaci reléové logiky.

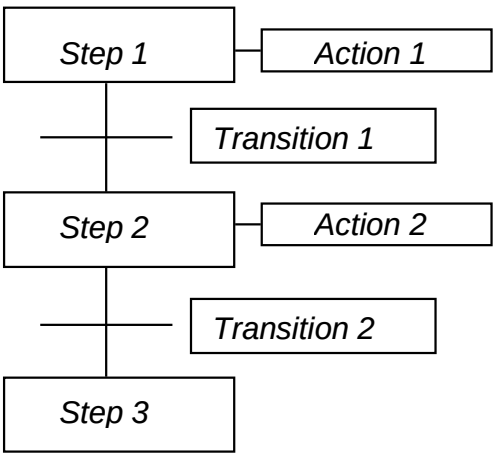
FBD - Function Block Diagram - jazyk funkčního blokového schématu
je velmi blízký procesnímu průmyslu. Vyjadřuje chování funkcí, funkčních bloků a programů jako soubor vzájemně provázaných grafických bloků, podobně jako v elektronických obvodových diagramech. Je to určitý systém prvků, které zpracovávají signály.

SFC - Sequential Function Chart – jazyk sekvenčních diagramů podobný Petriho sítím. Algoritmus je tvořen posloupností kroků (Steps), oddělených přechodovými podmínkami (Transitions). Ke krokům mohou být připojeny akce (Actions).

Pro první představu je na Obr.2 stejná logická funkce, a to součin proměnné A a negované proměnné B s výsledkem ukládaným do proměnné C, vyjádřen ve čtyřech základních programovacích jazycích, příklad struktury programu vytvořeného v jazyku SFC vidíme na Obr. 3.



Obr. 2. Ukázka programovacích jazyků



Obr. 3. Ukázka programovacího jazyka SFC

3. Základní stavební bloky programu

Základním pojmem při programování podle normy IEC 61 131-3 je termín ***Programová Organizační Jednotka*** nebo zkráceně ***POU*** *(Program Organisation Unit)*. Jak vyplývá z názvu, POU je nejmenší nezávislá část uživatelského programu. POU mohou být dodávány od výrobce řídícího systému nebo je

může napsat uživatel. Každá POU může volat další POU a při tomto volání může volitelně předávat volané POU jeden nebo více parametrů.
Existují tři základní typy POU :

funkce *(function, FUN)*
funkční blok *(function block, FB)*
program *(program, PROG)*

3.1.Funkce

Je nejjednodušší POU, její hlavní charakteristikou je to, že pokud je volána se stejnými vstupními parametry, musí produkovat stejný výsledek (funkční hodnotu). Funkce může vracet pouze jeden výsledek.

3.2. Funkční blok

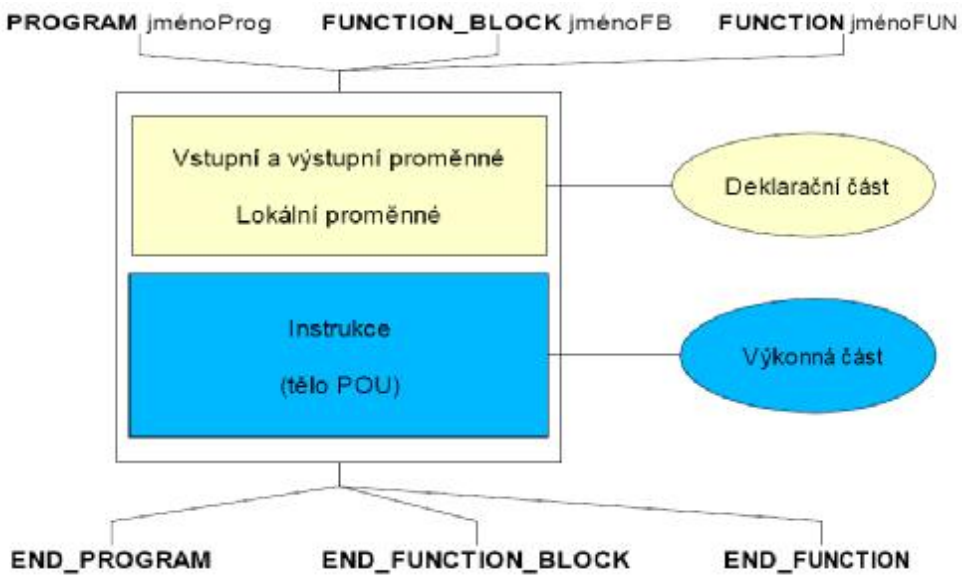
Dalším typem POU je ***funkční blok***, který si na rozdíl od funkce, může pamatovat některé hodnoty z předchozího volání (např. stavové informace). Ty pak mohou ovlivňovat výsledek. Hlavním rozdílem mezi funkcí a funkčním blokem je tedy schopnost funkčního bloku vlastnit paměť pro zapamatování hodnot některých proměnných. Tuto schopnost funkce nemají a jejich výsledek je tedy jednoznačně určen vstupními parametry při volání funkce.
Funkční blok může také (na rozdíl od funkce) vracet více než jeden výsledek.

3.3. Program

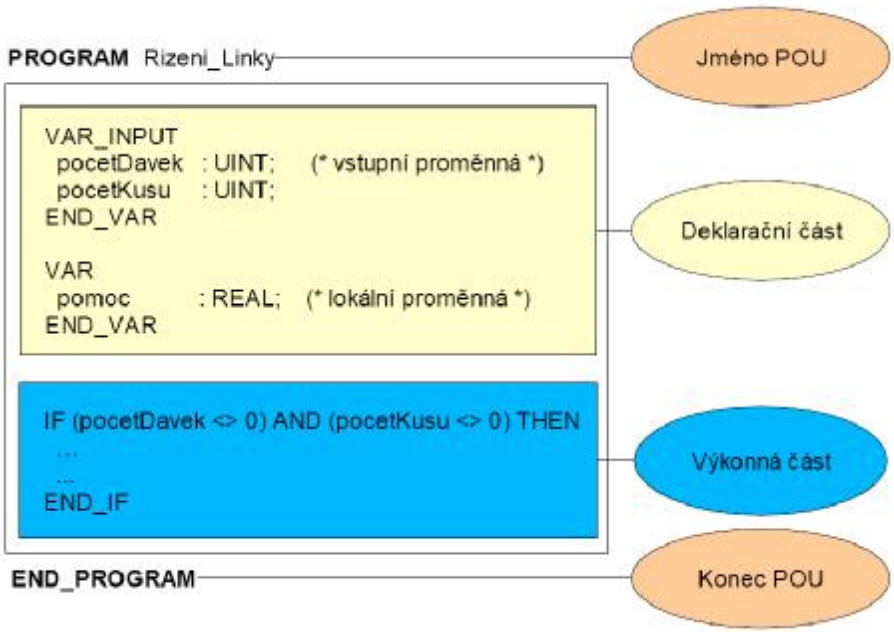
Posledním typem POU je ***program***, který představuje vrcholovou programovou jednotku v uživatelském programu. Centrální jednotka PLC může zpracovávat více programů a programovací jazyk ST obsahuje prostředky pro definice spouštění programů (v jaké periodě vykonávat program, s jakou prioritou, apod.).

3.4. Struktura POU

Každá POU se skládá ze dvou základních částí : ***deklarační*** a ***výkonné***, jak je vidět na Obr.4. V deklarační části POU se definují proměnné potřebné pro činnost POU. Výkonná část pak obsahuje vlastní příkazy pro realizaci požadovaného algoritmu.
Příklad struktury POU typu **PROGRAM** vidíme na Obr. 5. Definice POU začíná klíčovým slovem **PROGRAM** a je ukončena klíčovým slovem **END_PROGRAM**. Tato klíčová slova vymezují rozsah POU. Za klíčovým slovem **PROGRAM** je uvedeno jméno POU.
Poté následuje deklarační část POU. Ta obsahuje definice proměnných uvedené mezi klíčovými slovy **VAR_INPUT** a **END_VAR** resp. **VAR** a **END_VAR**.
Na závěr je uvedena výkonná část POU obsahující příkazy jazyka ST pro zpracování proměnných. Texty uvedené mezi znaky **(*** a ***)** jsou poznámky (komentáře).



Obr. 4. Struktura POU



Obr. 5. Ukázka programu v jazyku ST

3.4.1. Deklarační část POU

Deklarační část POU obsahuje definice proměnných potřebných pro činnost POU.
Proměnné jsou používány pro ukládání a zpracování informací. Každá ***proměnná*** je definována ***jménem proměnné*** a ***datovým typem***. Datový typ určuje velikost proměnné v paměti a zároveň do značné míry určuje způsob zpracování proměnné. Pro definice proměnných jsou k dispozici standardní datové typy (**BOOL**, **BYTE**, **INT**, …). Použití těchto typů závisí na tom, jaká informace bude v proměnné uložena (např. typ **BOOL** pro informace typu ANO-NE, typ **INT** pro uložení celých čísel se znaménkem apod.). Uživatel má samozřejmě možnost definovat svoje vlastní datové typy.
Umístění proměnných v paměti PLC systému zajišťuje automaticky programovací prostředí. Pokud je to potřeba, může umístění proměnné v paměti definovat i uživatel.

Proměnné můžeme rozdělit podle použití na
globální a ***lokální***. Globální proměnné jsou definovány
vně POU a mohou být použity v libovolné POU (jsou viditelné z libovolné POU). Lokální
proměnné jsou definovány uvnitř POU a v rámci této POU mohou být používány (z ostatních POU nejsou viditelné).
A konečně proměnné jsou také používány pro předávání parametrů při volání POU. V těchto případech mluvíme o ***vstupních*** resp. ***výstupních*** proměnných.

Příklad deklarace proměnných v POU

```
FUNCTION_BLOCK  PrikladDeklaraceProm
VAR_INPUT                           (* vstupní proměnné *)
        logPodminka :   BOOL;       (* binární hodnota *)
END_VAR
VAR_OUTPUT                          (* výstupní proměnné *)
        vysledek : INT;             (* celočíselná hodnota se znaménkem *)
END_VAR
VAR                                 (* lokální proměnné *)
        kontrolniSoucet : UINT;     (* celočíselná hodnota *)
        mezivysledek : REAL;        (* reálná hodnota *)
END_VAR
(* tady bude výkonná část POU *)
END_FUNCTION_BLOCK
```

3.4.2. Výkonná část PO*U*

Výkonná část POU následuje za částí deklarační a obsahuje příkazy a instrukce, které jsou zpracovány centrální jednotkou PLC. Ve výjimečných případech nemusí definice POU obsahovat žádnou deklarační část a potom je výkonná část uvedena bezprostředně za definicí začátku POU.
Příkladem může být POU, která pracuje pouze s globálními proměnnými, což sice není ideální řešení, ale může existovat.
Výkonná část POU může obsahovat volání dalších POU. Při volání mohou být předávány parametry pro volané funkce resp. funkční bloky.

Ukázka programu v jazyku ST

```
VAR_GLOBAL
// inputs
Start1 AT %X0.0,
Start2 AT %X0.1,
Stop1 AT %X0.2,
Stop2 AT %X0.3 : BOOL;
// outputs
Linka1 AT %Y0.0,
Linka2 AT %Y0.1 : BOOL;
END_VAR
FUNCTION_BLOCK fbStartMotoru
//-------------------------------------------------------------------------
VAR_INPUT
        start : BOOL;
        stop : BOOL;
END_VAR
VAR_OUTPUT
        vystup : BOOL;
END_VAR
        vystup := ( vystup OR start) AND NOT stop;
END_FUNCTION_BLOCK
```

```
PROGRAM Motory
//--------------------------------------------------------------------
VAR
	motor1 : fbStartMotoru;
	motor2 : fbStartMotoru;
END_VAR
	motor1( start := Start1, stop := Stop1, vystup => Linka1);
	motor2( start := Start2, stop := Stop2, vystup => Linka2);
END_PROGRAM
```

4. Společné prvky programovacích jazyků

Každý program pro PLC se skládá ze základních ***jednoduchých prvků***, určitých nejmenších jednotek, ze kterých se vytvářejí deklarace a příkazy. Tyto jednoduché prvky můžeme rozdělit na :

oddělovače (Delimiters),
identifikátory (Identifiers)
literály (Literals)
*klíčová slo*va (Keywords)
komentáře (Comments)

Oddělovače jsou speciální znaky (např. (,), =, :, mezera, apod.) s různým významem.
Identifikátory jsou alfanumerické řetězce znaků, které slouží pro vyjádření jmen uživatelských funkcí, návěští nebo programových organizačních jednotek (např. **Tepl_N1**, **Spinac_On**, **Krok4**, **Pohyb_dopr** apod.).
Literály slouží pro přímou reprezentaci hodnot proměnných (např. *0,1*; *84*; *3,79*; *TRUE* ; *zelena* apod.).
Klíčová slova jsou standardní identifikátory (např. **FUNCTION**, **REAL**, **VAR_OUTPUT**, apod.). Jejich přesný tvar a význam odpovídá normě IEC 61 131-3. Klíčová slova se nesmějí používat pro vytváření jakýchkoli uživatelských jmen. Pro zápis klíčových slov mohou být použita jak velká tak malá písmena resp. jejich libovolná kombinace.
Komentáře nemají syntaktický ani sémantický význam, jsou však důležitou částí dokumentace programu. Při překladu jsou tyto řetězce ignorovány, takže mohou obsahovat i znaky národních abeced. Překladač rozeznává dva druhy komentářů :

obecné komentáře řetězce znaků začínající dvojicí znaků **(*** a ukončené dvojicí znaků ***)**
řádkové komentáře začínající dvojicí znaků ***//*** a jsou ukončeny koncem řádku

5. Datové typy

Pro programování v některém z jazyků podle normy IEC 61 131-3 jsou definovány datové typy
elementární, předdefinované, (Elementary data types)
rodové datové typy (Generic data type) pro příbuzné skupiny datových typů.
odvozené *(uživatelské)* datové typy (Derived data type, Type definition).

5.1. Elementární datové typy

Elementární datové typy jsou charakterizované šířkou dat (počtem bitů) a případně i rozsahem hodnot. Přehled podporovaných datových typů je uveden v tabulce.

Klíčové slovo	Anglicky	Datový typ	Bitů	Rozsah hodnot
BOOL	Boolean	Boolovské číslo	1	0,1
SINT	Short integer	Krátké celé číslo	8	–128 až 127
INT	Integer	Celé číslo	16	–32 768 až +32 767
DINT	Double integer	Celé číslo, dvojnásobná délka	32	–2 147 483 648 až +2 147 483 647
USINT	Unsigned short integer	Krátké celé číslo bez znaménka	8	0 až 255
UINT	Unsigned integer	Celé číslo bez znaménka	16	0 až 65 535
UDINT	Unsigned double integer	Celé číslo bez znaménka, dvojnásobná délka	32	0 až +4 294 967 295
REAL	Real (single precision)	Číslo v pohyblivé řádové čárce (jednoduchá přesnost)	32	±2.9E-39 až ±3.4E+38 Podle IEC 559
LREAL	Long real (double precision)	Číslo v pohyblivé řádové čárce (dvojnásobná přesnost)	64	Podle IEC 559
TIME	Duration	Trvání času	24d 20:31:23.647	
DATE	Date (only)	Datum	Od 1.1.1970 00:00:00	
TIME_OF_DAY nebo TOD	Time of day (only)	Denní čas	24d 20:31:23.647	
DATE_AND_TIME nebo DT	Date and time of day	„Absolutní čas“	Od 1.1.1970 00:00:00	
STRING	String	Řetězec	Max.255 znaků	
BYTE	Byte(bit string of 8 bits)	Sekvence 8 bitů	8	Není deklarován rozsah
WORD	Word (bit string of 16bits)	Sekvence 16 bitů	16	Není deklarován rozsah
DWORD	Double word (bit string of 32 bits)	Sekvence 32 bitů	32	Není deklarován rozsah

Inicializace elementárních datových typů

Důležitým principem při programování podle normy IEC 61 131-3 je, že všechny proměnné v programu mají inicializační (počáteční) hodnotu. Pokud uživatel neuvede jinak, bude proměnná inicializována implicitní (předdefinovanou, default) hodnotou podle použitého datového typu. Předdefinované počáteční hodnoty pro elementární datové typy jsou převážně nuly, u data je to D#1970-01-01.

5.2. Rodové datové typy

Rodové datové typy vyjadřují vždy celou skupinu (rod) datových typů. Jsou uvozeny prefixem **ANY**. Např. zápisem **ANY_BIT** se rozumí všechny datové typy uvedené v následujícím výčtu: **DWORD, WORD, BYTE, BOOL**. Přehled rodových datových typů je uveden v tabulce.

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

5.3. Odvozené datové typy

Odvozené typy, tzn. typy specifikované buď výrobcem nebo uživatelem, mohou být deklarovány pomocí textové konstrukce **TYPE…END_TYPE**. Jména nových typů, jejich datové typy, případně i s jejich inicializačními hodnotami, jsou uvedena v rámci této konstrukce. Tyto odvozené datové typy se pak mohou dále používat spolu s elementárními datovými typy v deklaracích proměnných. Definice odvozeného datového typu je globální, tj. může být použita v kterékoliv části programu pro PLC.

Příklad jednoduchých odvozených datových typů

```
TYPE
        TMyINT :        INT := 15; // jednoduchy odvozeny datovy typ
        TRoomTemp :     REAL := 20.0; // novy datovy typ s inicializaci
        THomeTemp :     TRoomTemp;
        TPumpMode :     ( off, run, fault); // novy typ deklarovany vyctem hodnot
END_TYPE
PROGRAM SingleDerivedType
VAR
        pump1Mode :     TPumpMode;
        display :       STRING;
        temperature :   THomeTemp;
END_VAR
CASE pump1Mode OF
        off :   display := 'Pump no.1 is off';
        run :   display := 'Pump no.1 is running';
        fault : display := 'Pump no.1 has a problem';
END_CASE;
END_PROGRAM
```