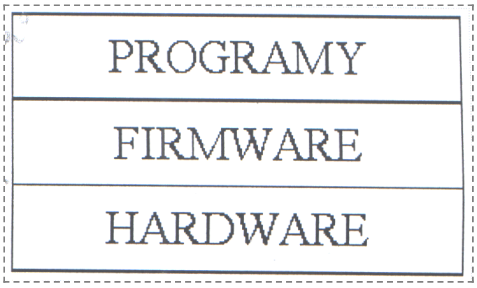
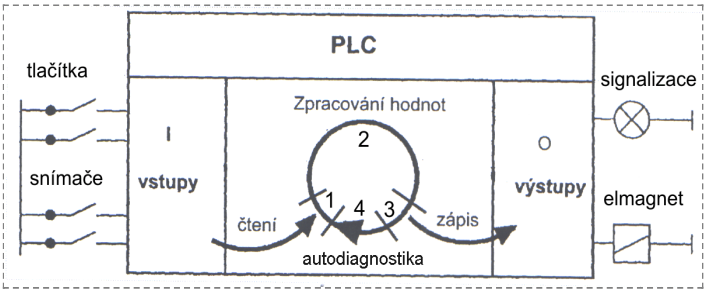
23. Software a programování PA

Vrstvy software	Princip práce PA
	

Vrstvy software:

↔	**Firmware**	↔	**systémový SW -** má funkci jako OS u PC - spouštění uživatelského programu, obsluha vstupů a výstupů, obsluha chyb, správa paměti
		↔	je závislý na HW (každý výrobce má svůj), výrobce poskytuje pravidelný upgrade
↔	**Programy**	↔	**aplikace - uživatelský SW** – programy pro konkrétní řízení pomocí PA - jsou vytvořené programováním

- Pozn. **SoftPLC** - software pro simulaci PA na PC (emulace PA - virtuální PA - PC se "tváří" jako PA), slouží k vývoji, ověření funkčnosti a výuce aplikačních programů pro PA

Princip práce PA:

↔	řídící program je zpracováván **cyklicky** („pořád dokola“)
↔	**Perioda programu** = doba jednoho pracovního cyklu – řádově v ms (záleží na délce programu, rychlosti CPU)

V každém cyklu probíhá:

1.	**Načtení** aktuálních **vstupních hodnot** do paměti vstupů - analogové a digitální vstupní signály poskytují snímače, tlačítka, spínače
2.	**Zpracování hodnot** v paměti podle aktuálního programu
3.	**Nastavení výstupních** (řídících) **hodnot** v paměti výstupů - analogové a digitální výstupní signály pro signální prvky, stykače, elektromagnetické ventily, které řídí akční prvky (motory, válce)
4.	**Autodiagnostika PLC** - kontrola chyb, také signál pro obvod **watchdog** (kontrolující běh programu - při "zaseknutí" programu resetuje PA) + komunikace s ostatními zařízeními

23.1. Programování PA

=	**tvorba** uživatelských **programů** pro požadovanou funkci PA
↔	liší se podle výrobce PA – způsob výrobce popisuje v dokumentaci
↔	programování probíhá většinou na **PC** s **vývojovým softwarem/prostředím** jako tzv. **projekt** - např. Siemens Step 7 (TIA portal)

Fáze tvorby programu

1.	**Rozbor zadání (analýza)**	↔	technologické schéma - náčrt, výkres - fyzické uspořádání řízeného systému
		↔	schéma řízení - vývojový diagram - algoritmus řízení
		↔	přiřazení vstupů a výstupů na PA konkrétním zařízením (zapojení snímačů, motorů)
2.	**Zápis programu**	↔	v editoru ve formě grafického nebo textového **programovacího jazyka** (viz další kapitola) + tvorba **dokumentace** programu
3.	**Překlad (kompilace)**	↔	z programovacího jazyka do strojového kódu procesoru PA - vznikne spustitelný kód
4.	**Ladění (debug)**	↔	**kontrola výpisu** kritických chyb (errors) a varování (warnings)
		↔	**odstranění** chyb

		↔	**simulace** - kontrola správné funkce programu na monitoru PC
5.	**Přenos kódu**	↔	PA musí být se stavu **STOP** (viz LED na PA)
		↔	**load** (download/upload) - nahrání přeloženého kódu do operační paměti PA - přes LAN, USB, flash disk
6.	**Testování a provoz**	↔	po zapojení vstupů a výstupů se PA uvede do stavu **RUN**

23.2. Programovací jazyky

↔	jazyk si volí programátor podle typu úlohy a podle svých zkušeností a dovedností

Rozdělení:

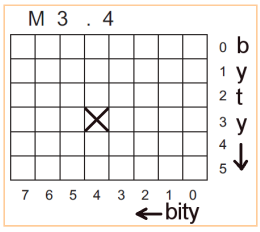
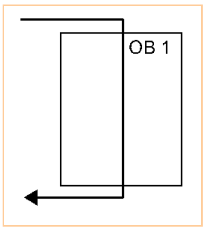
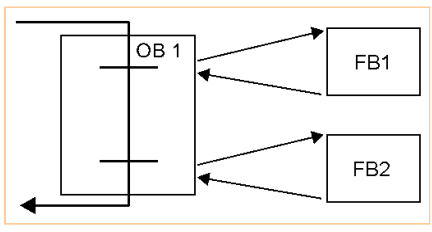
↔	**textové programovací jazyky**	↔	program tvoří posloupnost **textových příkazů** (instrukcí - významových slov)	
		↔	rozdělují se na jednodušší **seznam příkazů** (STL - statement list) a **vyšší programovací jazyky** (C, Pascal)	
↔	**grafické programovací jazyky**	↔	program tvoří posloupnost **grafických symbolů** – jsou přehlednější, druhy:	
			↔	**příčkový diagram** (Ladder)
			↔	**funkční bloky** (FBD - Function Block Diagram)
			↔	**sekvenční schéma** (SFC - Sequential Function Chart, GRAFCET)

23.2.1. Základní typy proměnných v PA Siemens S7-1200

- **Bit** je nejmenší jednotka informace - může mít hodnotu 0 nebo 1 (stav binárního signálu)
- **Byte** je skupina 8 bitů

Proměnná	Význam
Ix.y	Hodnota binárního vstupu, kde x.y je adresa vstupu - x je pořadové číslo bytu a y je pořadové číslo bitu (počítá se od 0) - např. I0.0 je první vstup (první bit prvního bytu)
Qx.y	Hodnota binárního výstupu - např. Q0.5 je šestý vstup (šestý bit prvního bytu)
Mx.y	Pomocné proměnné (tzv. markery) pro ukládání mezivýsledků

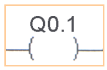
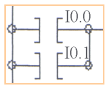
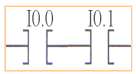
- Místo číselných adres lze používat i **slovní jména** (PLC tags) pro lepší orientaci v programu - např. Start, Stop

Proměnná v paměti	Hlavní program	Program s podprogramy
		

- Program musí obsahovat **hlavní organizační blok** (main OB), který se spouští jako první (z něho se mohou spouštět další bloky - podprogramy)

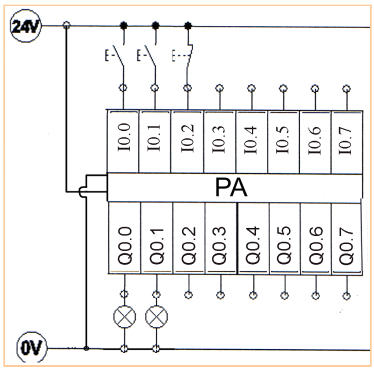
23.2.2. Příčkový diagram (LAD - Ladder diagram)

↔	program je tvořen posloupností příček žebříku (ladder) - také se říká žebříčkový diagram, příp. kontaktní nebo liniové schéma	
↔	diagram vychází z kontaktního elektrotechnického schématu zapojení vodičů, spínačů a řízeného "spotřebiče":	
↔	vlevo a vpravo jsou **napájecí sběrnice**	
↔	mezi nimi jsou vlevo **vstupy** v **hranatých** závorkách (spínací NO kontakty)	
↔	nebo rozpínací NC kontakty (mají negované hodnoty)	

↔	vpravo jsou **výstupy** v **kulatých** závorkách (akční prvky)	
↔	spínače svázané logickým součtem **OR** se kreslí **paralelně**	
↔	spínače svázané log. součinem **AND** se kreslí **sériově** (za sebou)	

Příklady programů:

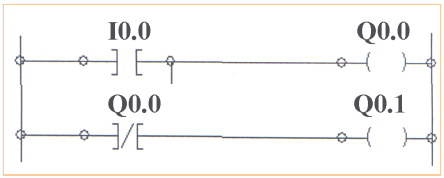
Zapojení vstupů a výstupů



Program 1.1a pro rozsvícení žárovky jedním tlačítkem:

I0.0	Q0.0
0	0
1	1

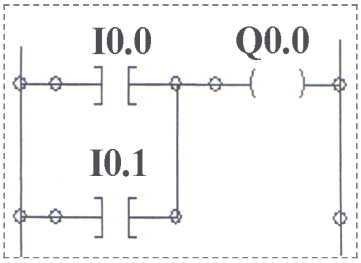
Program 1.1b pro negované svícení žárovek:



I0.0	Q0.0	Q0.1
0	0	1
1	1	0

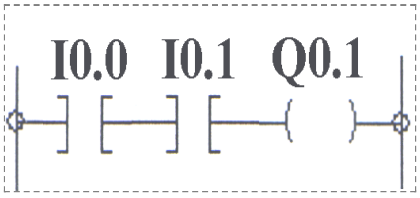
Program 1.2 pro rozsvícení žárovky dvěma tlačítky (logické funkce):

Logický součet OR (paralelní zap.)



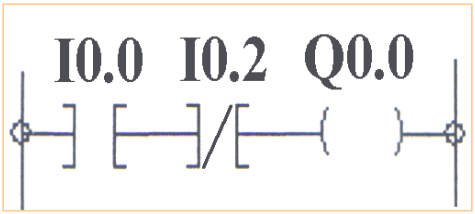
I0.0	I0.1	Q0.0
0	0	0
0	1	1
1	0	1
1	1	1

Logický součin AND (sériové zap.)



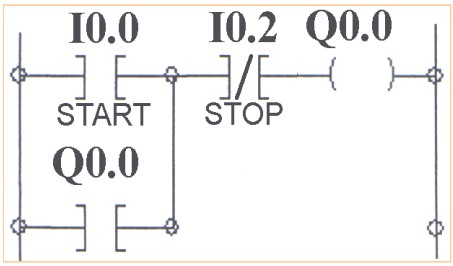
I0.0	I0.1	Q0.1
0	0	0
0	1	0
1	0	0
1	1	1

Program 1.3 pro zhasnutí žárovky rozpínacím kontaktem I0.2



I0.0	I0.2	Q0.0
0	0	0
0	1	0
1	0	1
1	1	0

Program 1.4 pro svícení žárovky i po uvolnění tlačítka I0.0

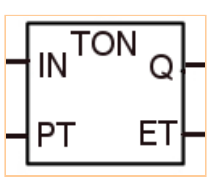


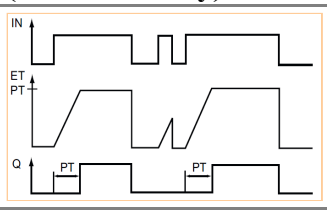
23.2.3. Funkční bloky

↔	obdélníkový **blok** symbolizuje určitou složitější **funkci** - např. čítač nebo časovač
↔	zleva bloku jsou **vstupy**, vpravo **výstupy**
↔	programátor může naprogramovat nové funkční bloky podle potřeby

Příklady funkčních bloků:

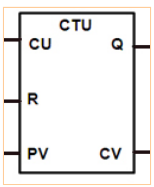
Časovač TON (timer on delay)

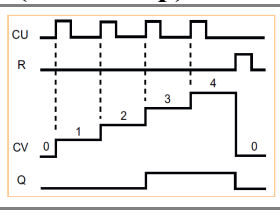




Časovač se spustí po zapnutí IN a po nastaveném čase PT (preset time) nastaví výstup Q na 1 (ET je uběhnutý čas)

Čítač CTU (counter up)



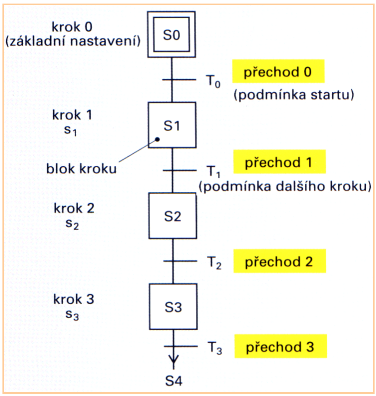


Každý náběh CU navýší aktuální počet CV, po dosažení PV se Q zapne, R slouží k vynulování CV

23.2.4. Sekvenční schéma

↔	používá se pro grafický popis **časové posloupnosti** jednotlivých akcí v rámci sekvenčního řízení strojů - také Sequential Function Chart, GRAFCET, GRAPH
↔	je tvořeno svislou sekvencí kroků (posloupností - steps) - graficky v obdélnících
↔	přechod na následující krok (transition) je podmíněn splněním podmínky - graficky vodorovnými čarami

Sekvenční schéma obecně



Příklad řízení vrtačky