Ismerkedés a PLC-vel

Juhász Róbert

(kizárólag oktatási célra)[1]

1 A tananyag szabadon felhasználható a közoktatásban a szerző megjelölésével.

Programozható Logikai Vezérlők

1. PLC történelem

A Programozható logikai vezérlő (PLC - Programmable Logic Controller) tulajdonképpen egy speciális ipari számítógép, amelyet digitális vagy analóg jelekkel történő vezérlési folyamatokra fejlesztettek ki üzemi, gyári környezetben való használatra. Eredetileg a régi relés vezérlések kiváltására hozták létre az autóiparban. Napjainkban a PLC már gyakorlatilag minden iparágban megtalálható, sőt a „civil" életben is rendkívül sok helyen alkalmazzák[2].

A PLC-k korai története az 1960-as évekig nyúlik vissza, amikor is a vezérlőberendezések relés logikákból álltak. Ezekben az időben a vezérlőtermek falait rengeteg relé, csatlakozó és még több vezeték foglalta el. Az ilyen rendszereknek rengeteg hibájuk volt, hogy csak néhányat említsünk:

- Rugalmatlan a rendszer: nincs lehetőség a bővítésre, valamint egyes paraméterek gyors megváltoztatására szükség esetén
- A hibajavítás – amely általában az eloxidált érintkezőknek és a laza kötéseknek volt köszönhető – egy rémálom volt, amelyet még csak súlyosbított, hogy a dokumentációkat nem frissítették, így a kapcsolási rajzok és a konkrét megvalósítás köszönő viszonyban se álltak egymással[3].

A kor mérnökei és technikusai nap mint nap szembesültek ezekkel a problémákkal. Akkoriban járta ez a mondás: „Öt órába telik megkeresni azt a hibát, amit aztán öt perc alatt ki lehet javítani."

1. ábra: Korabeli vezérlőszekrények

1968-ban Bill Stone, aki a General Motors automata sebességváltó üzemének volt a mérnöke ismertette a problémát a cég egyik konferenciáján. Stone felvázolt egy tervezési kritériumrendszert

2 Például intelligens épületvezérlések (böngészőnk keresősávjába beírva nagyon sok olyan oldalt találhatunk, amely az intelligens épületfelügyeletekkel foglalkozik).

3 Nem tudom elégszer hangsúlyozni, de a dokumentáció most is nagyon fontos, nem csak a hardvert kell leírással ellátnunk, hanem a megírt PLC programunkat is, és a bekövetkezett változásokat is nyomon kell követni!

a GM fejlesztőmérnökeinek egy úgynevezett „Szabvány Gép Vezérlőről". A kritériumoknak megfelelően kifejlesztett korai modell nem csak arra volt képes, hogy megszüntesse az – amúgy is megbízhatatlan – relés vezérlők cseréjéből adódó költségeket amelyek jelentkeztek a modellváltáskor, hanem hogy:

- Növelje az elektronikus áramkörök arányát 90%-ra a mechanikus alkatrészekkel szemben.
- Csökkentse az állásidőt azáltal, hogy könnyebb lett a karbantartás és a programozás a létradiagram felhasználásával.
- Tegye lehetővé a jövőbeli továbbfejlesztést, a moduláris felépítést és így az egyes alkatrészek cseréjét és bővítését.
- Legyen képes ipari környezetben is működni (szennyeződés, nedvesség, rázkódás, elektromágneses zavarok).
- Tartalmazzon funkcionálisan teljes logikát, kivéve az adatredukciós funkciókat.

Ezen specifikáció mentén a GM felkért 4 vezérlő berendezéseket gyártó céget egy prototípus elkészítésére:

- Allen-Bradley
- Digital Equipment Corporation (DEC)
- Century Detroit
- Bedford Associates

A javaslatban megfogalmazottak alapján a Digital Equipment készített egy „mini-számítógépet" a GM számára. Ezt végül elutasították több okból kifolyólag (pl. a statikus memóriája komoly korlátozó tényezőnek bizonyult).

Az Allen-Bradley – akinek már komoly tapasztalata volt az elektronika és a vezérlés területén – reagált a kihívásra a legjobban. Remélve, hogy eleget tesznek az összes elvárásnak az Allen-Bradley vállalat 5 hónap alatt elkészítette a prototípust, amelyet PDQ-II-nek (Program Data Quantizier) neveztek. Ez azonban túl nagy, túl összetett és nehezen programozható volt. A második próbálkozás, a PMC (Programmable Matrix Controller) már kisebb és könnyebben programozható volt, de még nem felelt meg teljes egészében a támasztott követelményeknek.

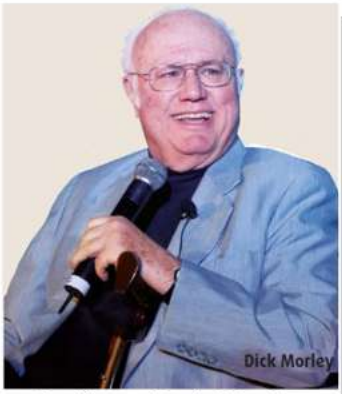
Eközben a Bedford Associates-nél Richard (Dick) Morley, Mike Greenberg, Jonas Landau, George Schwenk és Tom Boissevain már dolgoztak egy olyan egység tervén, amely moduláris és strapabíró kialakítású volt, valamint az egyes eszközöket közvetlen memória címzéssel lehetett

elérni[4]. A Bedford csapat 084-nek nevezte el ezt a vezérlőt, mégpedig azon okból kifolyólag, hogy ez volt a cég 84. projektje. Miután a vállalat pénzügyi problémákba ütközött a csapat úgy döntött, hogy létrehoznak egy új céget, amelyet Modicon-nak (Modular DIgital CONtroller)[5] neveztek el, s az Allen-Bradley-vel szoros együttműködésben elkészítik a vezérlőt. A Modicon-nál végül megalkották a 084-et, amelyet most már a Programmable Controller (PC) nevet kapta. Végül 1969-ben, amikor a GM-nél bemutatták a Modicon 084-es szilárdtest relén alapuló sorrendi vezérlőt a Bedford és a Modicon megnyerte a pályázatot. A Modicon 084 három egységre tagolódott, amely tartalmazta a processzor kártyát, a memóriát, a logikai egységet, amelyet létradiagramban lehetett programozni. A vezérlő egy teljesen zárt házba lett beszerelve, és nem tartalmazott Be/Ki kapcsolót sem. A rendszernek nem volt hűtőventilátora sem, a külső levegő egyáltalán nem jutott be a belsejébe, megakadályozva ezzel a szennyeződést és a korróziót. Morley úgy képzelte el, hogy ha egy kamion alá kötöznék és végigautóznák Texast és Alaszkát, az is ki kell, hogy bírja. A másik követelmény az volt, hogy elviselje a mikrohullámú sugárzást és a klimatizálatlan helyiséget is minden karbantartás nélkül.

2. ábra: AB 1774 PLC

1971-re Odo Struger és Ernst Dummermuth az Allen-Bradley mérnökei egy koncepció kidolgozásába kezdtek, amely már képes volt teljesíteni az ügyfelek azon igényeit is, amelyet a PMC-vel nem sikerült. Ez az új eszköz a Bulletin 1774 PLC nevet kapta. Az Allen-Bradley nevezte el ezt az eszközt „Programmable Logic Controller", azaz PLC-nek. A PLC terminológia lett ezután ipari szabvány különös tekintettel arra, hogy PC (Personal Computer) a személyi számítógépek elnevezésévé vált[1]

Végül álljon itt egy kis érdekesség a PLC történetéből. 2008-ban a PLC 40 éves évfordulóján Dick Morley a PLC atyja így ír visszaemlékezésében[2]: „A leginkább arra emlékszem, hogy másnapos voltam. Azért voltam másnapos, mert Újév reggele volt[6]. Ekkor már két hetes késében voltam, hogy javaslatot tegyek egy új rendszerre, de úgy gondoltam túl fáradt vagyok ehhez. Így ahelyett, hogy a javaslatot tettem volna kitaláltam – Istennek bizony ezen a napon – a programozható vezérlőt." Morley előtt kitisztult a kép, hogy milyen is legyen ez a programozható vezérlő:



3. ábra: Dick Morley

4 Az ötlet nagyszerűsége abban áll, hogy a különböző egységeket (pl. bemeneti modul) nem külön csatolófelületeken, terminálokon, stb. keresztül érjük el, hanem egyszerűen csak egy memóriabeli címre kell hivatkoznunk!

5 A cég ma is létezik Schneider Electric néven

6 1968. január elseje

- Nincs megszakítás a folyamatok számára
- Közvetlen elérés a memóriából
- Nincs szoftveres kiszolgálás az ismétlődő feladatoknál
- Robusztus felépítés
- Lassú (ez tévedés volt, amire Morley később rájött)[7]
- Programnyelv[8]

2. PLC-k az iparban

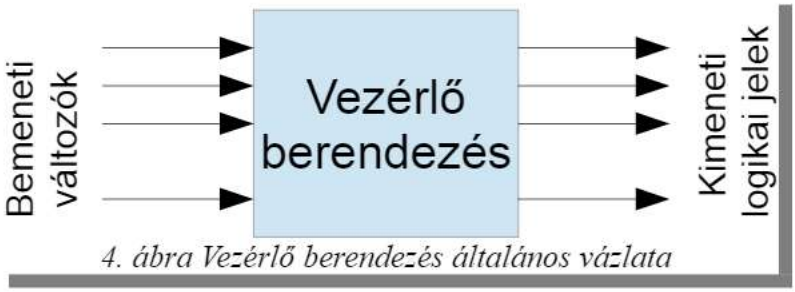
Az előzőekben néhány dolgot már áttekintettük a vezérléstechnika történetéből. A teljesség kedvéért érdemes megemlíteni, hogy a mechanikus vezérlésekkel itt most nem foglalkozunk, illetve az elektromechanikus és PLC-s vezérléseken kívül létezik más alternatíva is. Az elektronika fejlődésével egyre bonyolultabb és megbízhatóbb áramköröket készítettek, amelyek alkalmasak voltak az elektromechanikus vezérlések kiváltására. Megjelentek a nyomtatott áramkörök[9], amelyek kiküszöbölték a vezetékes összekötésből eredő hibákat, hiszen az alkatrészek közötti összeköttetést nem nekünk kell elvégezni, hanem a nyák rajzolat adja. A nyomtatott áramkör jelentősége a tranzisztor feltalálása után mutatkozott meg, hiszen a megelőző technikák (elektroncső) nem igényelték ezt a nagy méret miatt. Ugyanilyen jelentőséggel bírt az integrált áramkörök feltalálása[10] is, hiszen a kapcsolásunkat immár nem diszkrét elemekből kellett összeraknunk. Az egyes alkatrészek közötti összekötések az integrált áramkörön belül valósultak meg, így a nyomtatott áramkörön csökkent a huzalozások száma, ami megbízhatóbbá tette a kapcsolást. Ezen kívül ez a technológia már lehetővé tette, hogy az egyes áramkörök egyre kisebbek legyenek. Ekkora már készültek olyan integrált áramkörök, amelyek egymagukban képesek voltak egy eszköz vezérlésére. Ezeket BOÁK-nak (Berendezés Orientált Áramkörök) nevezték. Későbbiekben megjelent a mikroprocesszor, a mikroszámítógép és a mikrovezérlő, amelyek szintén alkalmasak vezérlési és szabályozási feladatokra.

2.1. A PLC fogalma

Először is vizsgáljunk meg egy egyszerű diszkrét elemekből álló vezérlést[11]. Ebben az estben bizonyos számú érzékelő elem kétállapotú információkat szolgáltat a működési feltételek

7 Megemlíthetjük, hogy a PLC-k ma sem tartoznak a leggyorsabb vezérlőeszközök közé.
8 A létradiagram néhány hónap múlva született meg
9 Paul Eisler 1943-ban szabadalmaztatta módszerét, amellyel lehetővé vált vezető sávok kialakítása üvegszálas erősítésű nem-vezető alapanyagra felvitt rézfólián
10 Jack Kilby 1958 Texas Instruments
11 Az egyszerűség kedvéért kombinációs logikai hálózatról beszéljünk, a sorrendit egyelőre ne tárgyaljuk.

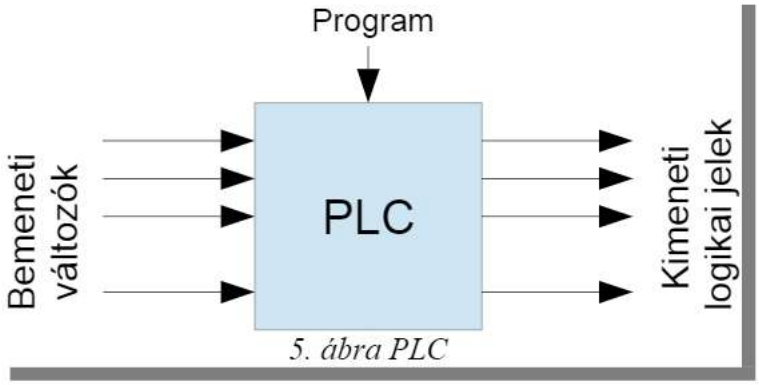
teljesüléséről. A vezérlőberendezés a benne megvalósított logikai algoritmus alapján működteti a végrehajtó szerveket. Van tehát valamennyi bemeneti változó, van valahány kimeneti változó és köztük egy berendezés, amely a logikai függvényeket megvalósítja. Ez tulajdonképpen megfelel a logikai hálózatok modelljének.



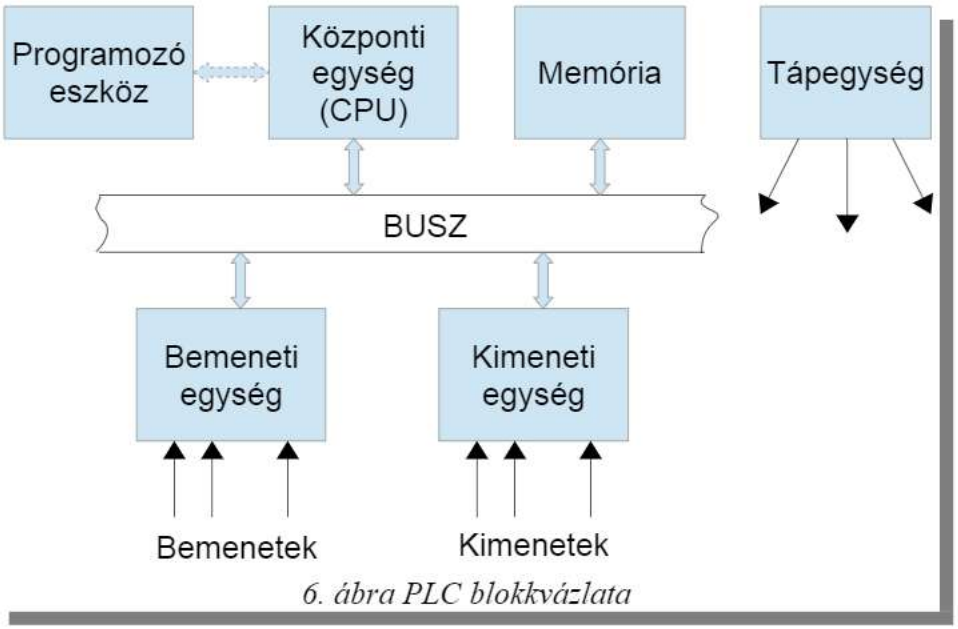
4. ábra Vezérlő berendezés általános vázlata

Az ipari alkalmazások nagy többségében szükség lehet arra, hogy a vezérlőberendezés rugalmas legyen. Ugyanis sok esetben lehet feltétel, hogy egy gyártmány opcionális tulajdonságokkal rendelkezzen és így igény szerinti eltérések legyenek az egyes berendezések között. Másik lehetőség, amikor a már működő berendezést kell módosítani pl. az alkalmazásának megváltozása miatt. Ilyen eset fordulhat elő, amikor egy automatikus gyártósort átállítanak egy másik termékre. Mindkét estben belátható, hogy a módosításoknak milyen hatása van a berendezés árára. Az első példában vagy minden egyedi igényhez külön készítik el a vezérlést, vagy minden gépet felkészítenek az összes lehetséges opció megvalósítására, és így felesleges részeket is beépítenek. A második példában szereplő gyártósor átállítás lehet, hogy olyan mérvű változtatásokat kíván, ami miatt a teljes vezérlőberendezést egyszerűbb kicserélni, mint a régit átalakítani.

A fentiekből érzékelhető, hogy olyan vezérlőberendezésre lenne szükség, amely könnyen és lehetőség szerint olcsón alakítható az igényekhez. A megoldást, mint ahogy fentebb láttuk az elektronika gyors léptékű fejlődése hozta meg. Az is világosan látszik, hogy a problémát egy olyan berendezés tudja megoldani, amely programozható, s így rugalmasan viszonyul az igényekhez. A programozhatóság azt jelenti, hogy a felhasználó a működtető program módosításával, vagy cseréjével ugyanazt a berendezést teljesen más feladat elvégzésére teszi alkalmassá. Olyan célszámítógépre van tehát szükség, amely rendelkezik megfelelő be- és kimeneti csatlakozási felülettel és benne az ezek közti logikai kapcsolat programozással valósítható meg. Az ilyen eszközöket programozható logikai vezérlőknek nevezzük (PLC, Programmable Logic Controller).



5. ábra PLC



6. ábra PLC blokkvázlata

2.2. A PLC-vel szemben támasztott követelmények

A PLC történetével kapcsolatban már találkozhattunk azzal, hogy milyen követelményeket támasztottak megjelenésekor az új eszközzel szemben. Ezek az elvárások jórészt ugyanazok maradtak néhány kiegészítéssel. Vegyük sorra a bővített kritériumokat:

- A felhasználó által könnyen programozható legyen.
- Rendelkezzen olyan csatlakozási felülettel, amely jelszintjeiben, védettségében (zárlat, túlfeszültség), a be- és kimenetek számában illeszkedik a felhasználási környezetbe.
- Működési sebessége tegye lehetővé a vezérelt rendszer biztonságos felügyeletét.
- Mérete lehetőleg kicsi legyen a könnyű beépíthetőség miatt.
- Ipari körülmények között is megfelelően működjön. Elsősorban viselje el az ingadozó hőmérsékletet és hálózati feszültséget, legyen védett a rezgéssel és a szennyeződéssel szemben.
- Könnyen lehessen a helyszínen diagnosztizálni és a hibát elhárítani.
- Olcsó legyen[12].

2.3. A PLC általános felépítése

Mivel a PLC tulajdonképpen egy speciális számítógép, ezért felépítése hasonló. A PLC fő egységei a következőek:

- CPU (központi egység)

12 A mai PLC-kre sajnos ez nem jellemző!

- Memória
- Be- és kimeneti egységek

A mikroszámítógéphez hasonlóan az egységeket buszrendszer köti össze. Ez teszi lehető, hogy a rendszer bővíthető legyen, hiszen így a sínrendszerre több elemet is felfűzhetünk. A buszrendszer itt is 3 részre tagolódik. A címbusz végzi azon egységek kiválasztását, amelyikkel a CPU kommunikálni szeretne, az adatbuszon közlekednek a hozzá tartozó adatok, a vezérlőbuszon pedig a rendszer vezérlőjelei és a különböző állapotjelek találhatók. A legtöbbször a be- és kimenetek számát szükséges bővíteni. Fontos tudni, hogy a bővíthetőséget a CPU címtartománya határozza meg. PLC-t gyártanak moduláris és kompakt kivitelben is. Az előbbi tetszés szerint bővíthető – természetesen a CPU címtartományán belül – be- és kimeneti egységekkel, analóg modulokkal, hajtásszabályozással, kommunikációs perifériákkal, és így tovább. A kompakt PLC esetében az össze részegység egybe van építve. Azt hihetnénk, hogy ezt nem lehet bővíteni, azonban több típus esetén a be- és kimenetek száma (de csak ez) növelhető.

2.3.1. A PLC-k fő egységei

A következőkben tekintsük át a PLC-k főbb egységeit részletesen.

- Központi egység

Ez látja el a PLC működtetését, benne történik a felhasználói program feldolgozása. Kezeli a be- és kimeneti modulokat, illetve az egyéb perifériákat. A központi egység lényegében a mikroprocesszort és az azt kiszolgáló áramköröket foglalja magában. Szokás a központi egységet CPU-nak is nevezni. Ide kapcsolódnak a programozó készülékek és az egyéb külső adatátviteli vonalak.

7. ábra S7-300 CPU

- Memória

Elsősorban a felhasználói programok tárolására szolgál, tehát írható-olvasható memória. Leggyakrabban CMOS RAM-ot, EEPROM-ot, vagy FLASH memóriát használnak. CMOS RAM esetén akkumulátorral védik a cellákat felejtés ellen. Természetesen az akkumulátor meghibásodása esetén a program elvész. Ennek megakadályozására használják azt a módszert is, hogy a felhasználói programot lementik valamilyen EPROM cellába, s a PLC bekapcsolásakor mindig visszaírják onnan az operatív memóriába. Ezen kívül egy másik memória tartalmazza a PLC rendszerprogramját is. Ezek kezdetben csak olvasható (ROM) memóriák voltak, manapság FLASH típusúak, így a PLC operációs rendszerét könnyű

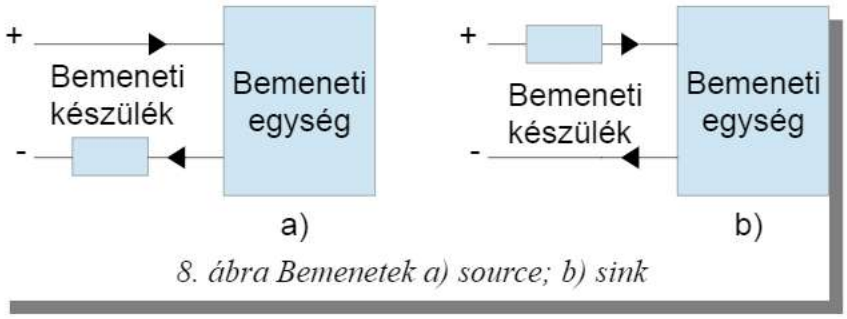
frissíteni.

- Be- és kimeneti egység

Bemeneti egység: fogadja a külvilág felől érkező jeleket, biztosítja a megfelelő elválasztást és a bemeneti jelszintet illeszti a PLC belső jelszintjéhez. Moduláris felépítésnél egy modul általában 8 vagy 16 bemenetet tartalmaz. Létezik olyan modul is, amelyik vegyesen tartalmaz be- és kimenetet is. A bemeneti egység lehet:

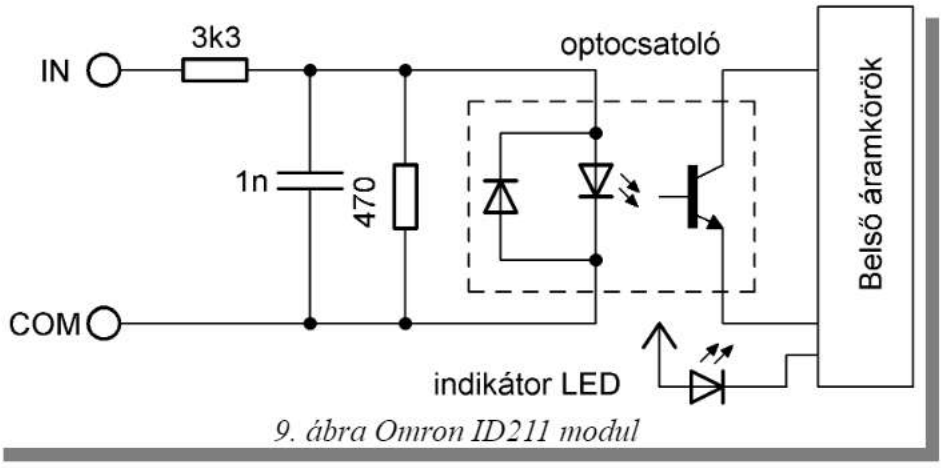
- Diszkrét (digitális)
- Impulzus számláló
- Analóg

A bemenet lehet source (forrás) vagy sink (nyelő) típusú. Ezt az alábbi ábra szemlélteti:



8. ábra Bemenetek a) source; b) sink

A bemeneti jelszintet illeszteni kell a PLC belső jelszintjéhez, ami általában TTL kompatíbilis és a galvanikus leválasztást is el kell végezni, ezért általában a bemeneten optocsatolót alkalmaznak, ami mindkettőt megoldja egyszerre. A 9. ábrán erre láthatunk példát.



9. ábra Omron ID211 modul

A modulok 12/24 V AC/DC jelek fogadására alkalmasak általában, de létezik olyan is amely a hálózati 203 V-os jelszintet is képes fogadni. Analóg jelek fogadásához a bemeneten A/D átalakítóra van szükség, hiszen a PLC belül digitális jelekkel dolgozik. Ezek a bemenetek általában 4 analóg jel fogadására képesek. Feszültség bemenet estén a tartomány lehet:

- 0÷5 V

- 0÷10 V
- ±10 V

Áram esetén:

- 0÷20 mA
- 4÷20 mA

Az áramjel különösen akkor hasznos, ha az érzékelő és a PLC között nagy a távolság, hiszen a vezeték hosszából származó ellenállás ilyenkor nem befolyásolja a mérést. A 4 mA minimum pedig azt a célt szolgálja, hogy kimutatható legyen a vezeték szakadás.

10. ábra. Omron A/D modul

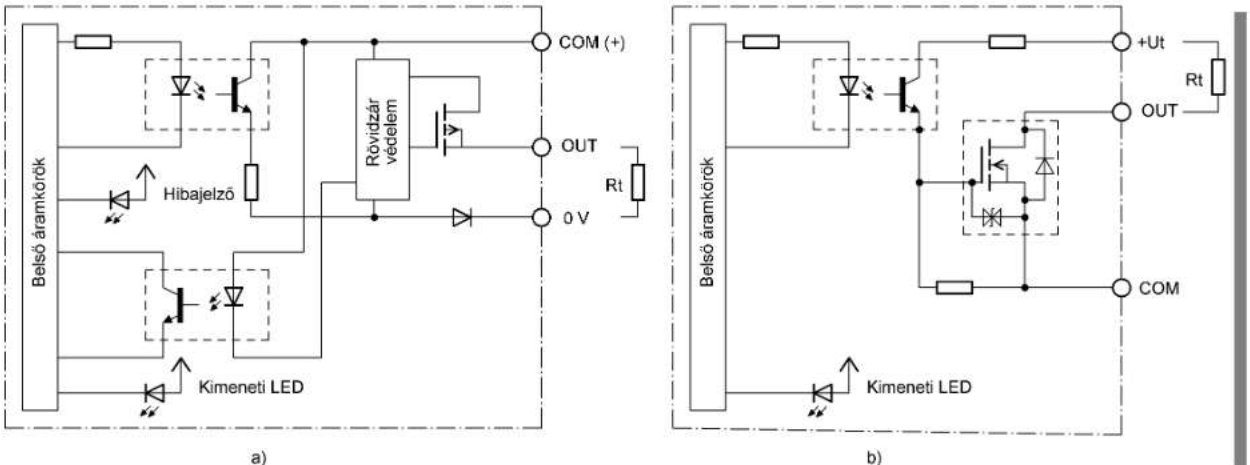
Kimeneti egység: továbbítja megfelelő jelszinten a kimeneti vezérléseket. A kimeneti egység lehet:

- Diszkrét (digitális)
- Analóg

Fizikai megvalósítás szempontjából:

- Relés
- Félvezetős

A kimeneteknél meg kell említenünk, hogy a kiválasztásuknál nagyon fontos a terhelhetőség figyelembe vétele, különösképpen a félvezetős típusoknál. A relés kimeneteknél ezen kívül figyelembe kell venni a maximális kapcsolásszámot is. A kimenetek is lehetnek source vagy sink típusúak.



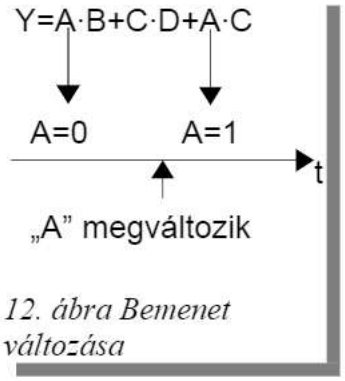
11. ábra Omron a) OD204 (source); OD203 b) (sink) modulok

Tápegység: Az ipari PLC-k tápegységével szemben az a legfontosabb követelmény, hogy szélsőséges körülmények között is biztosítsák a megfelelő tápellátást. Kellő terhelhetőséggel kell rendelkeznie, hiszen előfordulhat, hogy a modulok számának változtatásával erősen megváltozik a terhelése. Mivel a PLC-ben információ feldolgozás folyik, ezért különös gondot kell fordítani a megfelelő zavarszűrésre és árnyékolásra. A rossz tápegység számtalan „rejtélyes" hiba forrása lehet. A PLC gyártók természetesen tápegységet is készítenek a PLC-hez. Napjainkban a legtöbb tápegység kapcsolóüzemű és így a bemeneti feszültségük is széles tartományban változhat (100-240 VAC). Léteznek olyan tápegységek is, amelyek „nyers" 24 VDC feszültséget igényelnek.

2.4. PLC-k működése

A PLC-k működésének megértése nagyon fontos lesz a programírás folyamán. Tapasztalataim azt mutatják, hogy számos hiba adódik abból, hogy a programozó nincs tisztában a PLC működésével. Nézzük végig tehát hogyan is működik a PLC! A PLC fogadja a külvilág felől érkező jeleket. Ezek változása aszinkron módon bármikor bekövetkezhet. Mint megállapítottuk a PLC tulajdonképpen egy célszámítógép, így működése is ennek megfelelő. A megírt program utasításait a memóriából sorban kiolvasva végrehajtja. A programban viszont nem kötött az utasítások sorrendje, gondoljuk pl. egy egyszerű kombinációs hálózatra, ahol az ÉS[13] műveletben a változók sorrendje tetszőleges lehet.

Ezeket figyelembe véve a sorrendi végrehajtás alatt előfordulhat, hogy az egyik bemeneti változó értéke a program futása közben megváltozik. Amennyiben ez a változó a programban több helyen is szerepel – márpedig ez elég gyakori – akkor az két különböző értékkel lesz figyelembe véve. Ez így természetesen hibás, hiszen a kimeneten olyan bemeneti kombinációra kapunk választ, ami nem is létezhet, hiszen az „A" változó nem vehet fel egyszerre kétféle értéket!

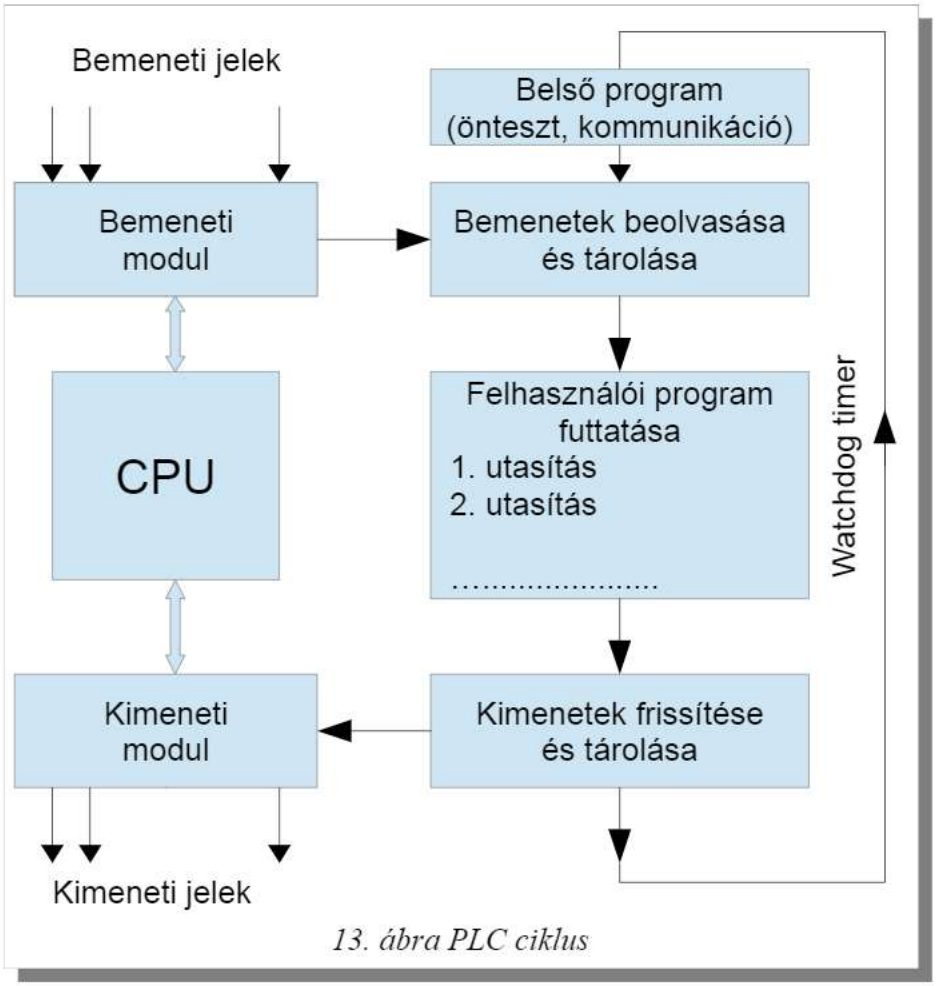


12. ábra Bemenet változása

A hiba kiküszöbölésére a PLC-ben azt a módszert választották, hogy a program futása alatt a bemeneteket és a kimeneteket befagyasztják. A program végrehajtása ciklikusan történik. A ciklus elején a PLC mintavételezi a bemeneteket és eltárolja egy regiszterben, így a teljes ciklus alatt ezeket veszi figyelembe. A bemeneten hiába történik változás, az csak a következő ciklusban lesz érvényes. Ezután lépésről-lépésre végrehajtja a programot, majd az eredményt a kimeneti regiszterbe írja. Ilyen módon a kimeneten is legközelebb akkor lesz változás, ha a ciklus lefut. Mindezekből az következik, hogy a ciklus alatt a bemeneten történő változásokat a PLC nem veszi figyelembe, csak a következő ciklus elején. Amennyiben a változás rövidebb impulzus, mint a

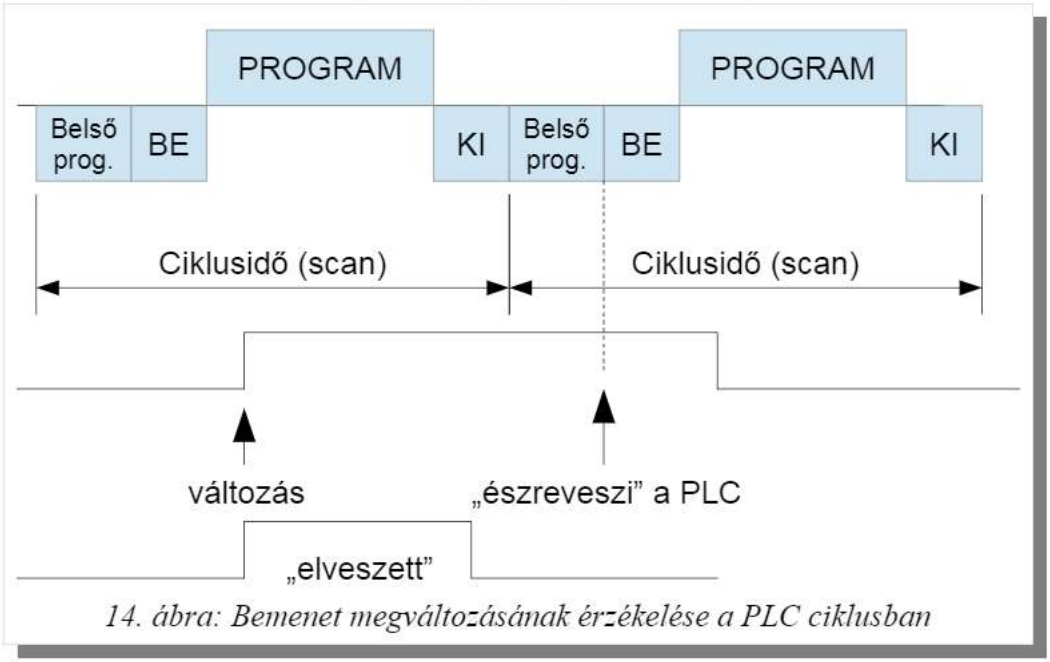
13 Ismételjük át a Boole-algebrát!

ciklusidő, akkor az elveszik.



13. ábra PLC ciklus

Ennek elkerülése érdekében vannak olyan bemenetek, amelyek ezt a rövid impulzust a következő ciklus kezdetéig eltárolják, vagy megszakításos eljárással dolgozzák fel.[14] Ez a



14. ábra: Bemenet megváltozásának érzékelése a PLC ciklusban

14 A PLC programozásnál általában ezeket a bemeneteket azonnali frissítésűnek nevezik.

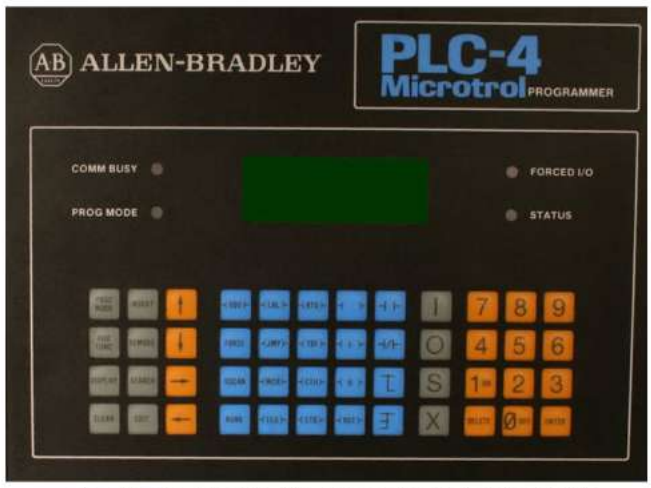
mintavételezés a legrosszabb esetben a változás észlelésében 1 ciklusidőnyi késést okoz. Természetesen a kimeneteken sem történik változás a ciklus alatt, az előző ciklus végén felvett állapot marad fenn.

Ezeket a sajátosságokat figyelembe kell venni a PLC kiválasztásakor, hiszen ezek az idők a program hosszának függvényében változnak. A számításhoz a PLC gyártók megadják az 1 kB hosszú programra vonatkozó ciklusidőt. Az alkalmazásánál figyelembe kell venni, hogy a vezérelt folyamat maximum mekkora időkésést visel el.

Gyakori hibaként szokott felmerülni – tapasztalataim szerint – hogy egy kimenetre a programozók gyakran több feltételt adnak meg külön-külön. Pl. az Y kimenet akkor kapcsol be, ha A és B változó igaz, vagy C és D igaz: Y=AB, illetve Y=CD. Attól függően, hogy melyiket írtuk a programban később, a kimenetre csak ez a változás kerül, hiszen a ciklusban sorrendi végrehajtás van és a kiírás csak a program végén hajtódik végre! A helyes megoldás Y=AB+CD! Gondoljuk csak végig, a kombinációs hálózatoknál is úgy írjuk fel a függvényt, hogy a mintermeket VAGY kapcsolatba hozzuk egymással. Amit meg kell jegyezzünk, hogy egy kimenetre csak egy függvényt írhatunk fel! A legtöbb PLC program ezt jelzi is, sőt le sem engedi tölteni az ilyen függvényt, de nem mindegyik! Ez alól a SET és a RESET utasítás lehet kivétel.[15]

2.5. PLC-k programozása

A PLC-k programozására az idők folyamán sokféle programnyelv alakult ki. A legősibb a létradiagram volt, amit napjainkban is előszeretettel alkalmaznak. Ezen kívül a programozó eszközök is sokat fejlődtek a PLC kialakulása óta. Ma már természetesnek vesszük, hogy számítógép segítségével írjuk a programokat, de gondoljunk csak bele az akkori körülményekbe



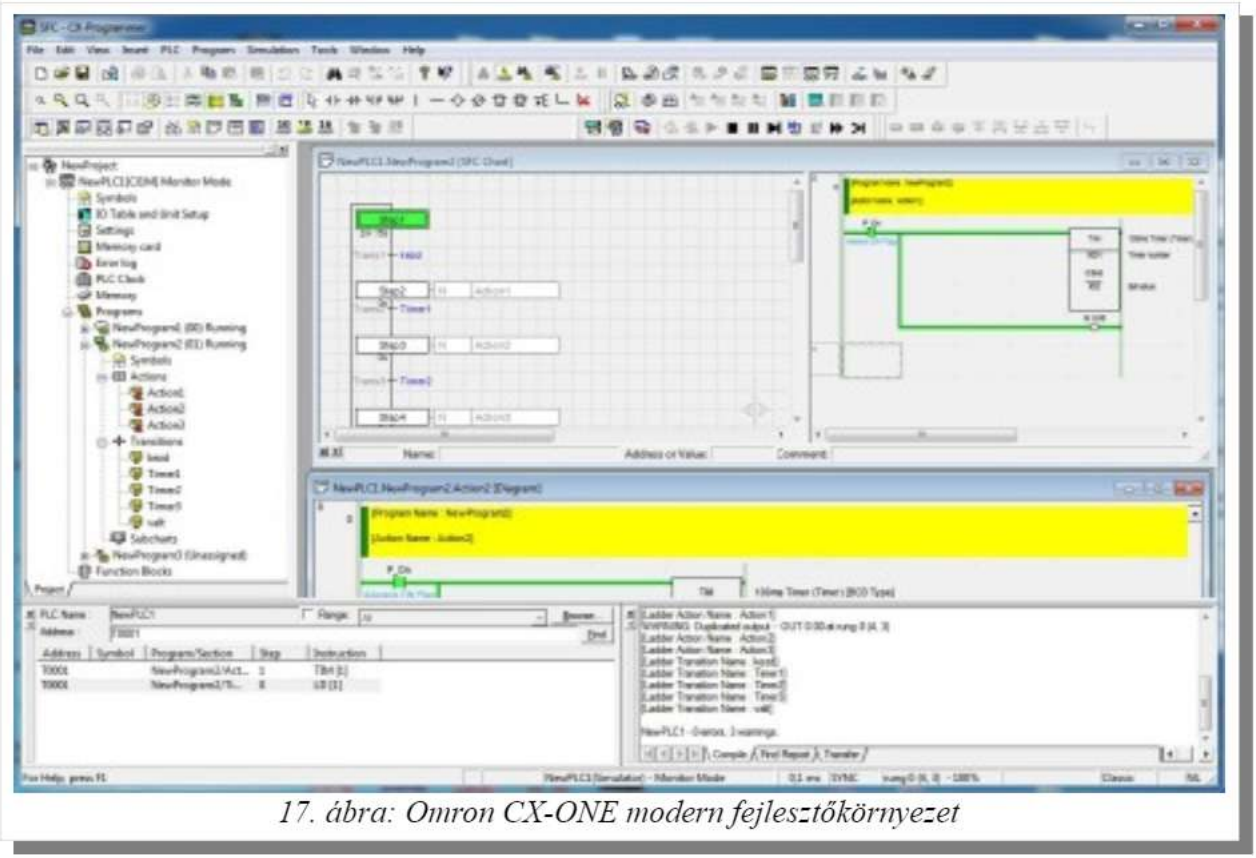
15. ábra: Korai AB programozó készülék

16. ábra: LOGO! 230RCE

(1971-ben találták fel az első mikroprocesszort)[16]. A kezdetekben különböző külső programozó

15 Azonban ez is az adott PLC-től függ!
16 Intel 4004, bővebben információk: itt.

készülékeket használtak a program PLC-be töltéséhez. Ilyen eszközök ma is léteznek modernebb kivitelben. Egyszerűbb esetekben a programozó készüléket egybe építik a PLC-vel[17]. Természetesen ezeknél is van ma már lehetőség a számítógépes programbevitelre. A beépített programozó akkor lehet hasznos például, ha csak valamilyen paramétert és nem a programot szeretnénk módosítani.



17. ábra: Omron CX-ONE modern fejlesztőkörnyezet

Vizsgáljuk meg a következőkben, hogy milyen programozási nyelveket használhatunk napjainkban. Természetesen, mint bárhol a műszaki életben itt is vannak szabványok annak érdekében, hogy egységes és érthető legyen a világon mindenki számára az adott programnyelv[18]. Az ide vonatkozó szabvány az IEC-61131 nevet viseli[19]. A programozást a 3. rész tartalmazza: IEC-61131-3. IEC 61131-3 – nem csak azokat a PLC programozási nyelveket írja le, hanem a PLC projektek létrehozására is kínál átfogó koncepciókat és irányelveket.

2.5.1. Programszervezési egységek

Az IEC 61131-3 Programszervezési Egységeknek (POU – Program Organisation Units) nevezi azokat a blokkokat, amiből a PLC programok, projektek felépülnek. A korábbi szabványokhoz képest ez 3-ra redukálta a blokkok számát az egyszerűbb használhatóság érdekében. A következő POU típusok léteznek:

17 Siemens LOGO! PLC.

18 Sajnos a tapasztalatok azt mutatják, hogy a szoftverkészítők nem igazán olvassák ezeket a szabványokat, hiszen óriási a kuszaság ezen a területen. Az egyes gyártók teljesen eltérő terminológiákat használnak sokszor!

19 Az első ide vonatkozó szabvány az IEC 848 (DIN 40 719-6 funkció blokk diagram) volt 1977-ben .

POU típusa	Szabványos név	Jelentés
Program	PROGRAM	Főprogram be- és kimenetek hozzárendelésével, globális változókkal, elérési utakkal.
Függvény blokk	FUNCTION_BLOCK	Blokk be- és kimeneti változókkal (a leggyakrabban használt POU típus).
Függvény	FUNCTION	A PLC alapműveletek kibővítésére szolgáló blokk.

1. táblázat: POU típusok

A 3 POU típus fő jellemzői:POU típusok

- Függvény (FUN): hozzá lehet rendelni paramétereket, de nincs statikus változója (memóriája). Azonos paraméterekkel meghívva mindig ugyanazt az eredmény adja vissza.
- Függvény blokk (FB): paraméterezhető és rendelkezik statikus változóval (memória). Az FB (pl. számláló vagy időzítő) kimenetén visszaadott érték nem csak a paraméterektől, hanem a belső (VAR) és külső változók (VAR_EXTERNAL) értékétől is függ. A blokk megtartja a visszaadott értéket a következő függvényhívásig.
- Program (PROG): Tulajdonképpen ez a főprogram. A program összes változójához hozzárendelt fizikai címet (pl. I/O címek) itt vagy ettől magasabb szinten (konfigurációs beállítások) kell deklarálni. Minden más tekintetben úgy viselkedik, mint egy FB.

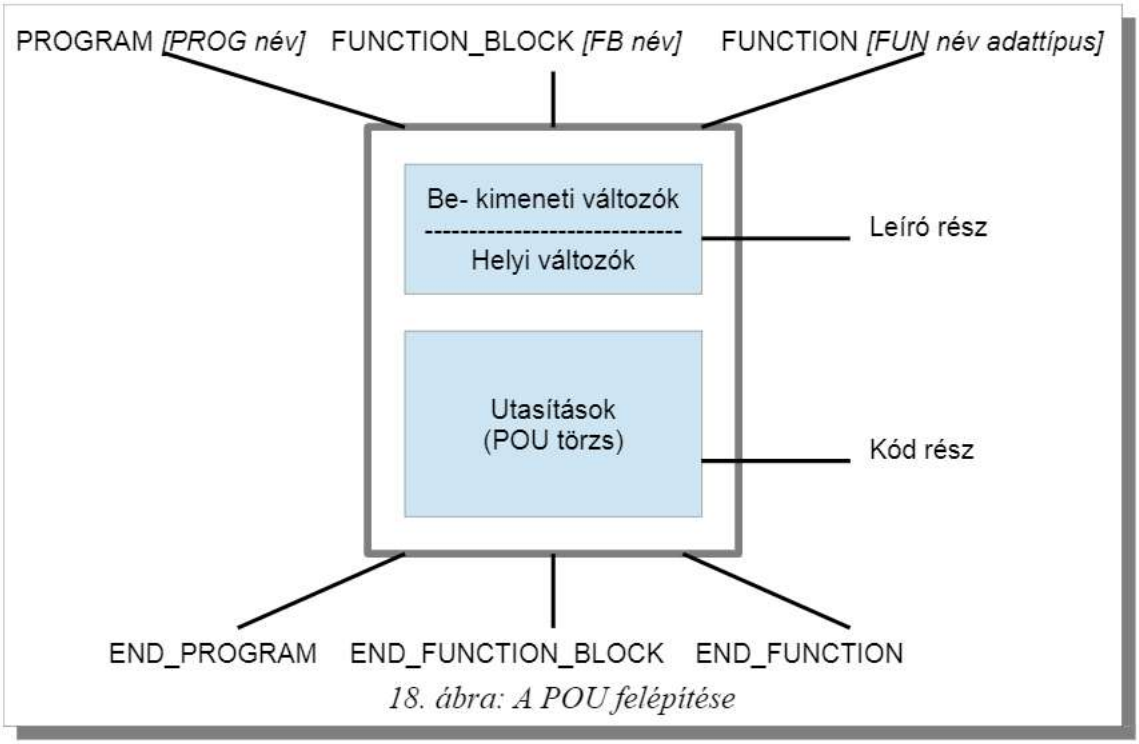
Minden POU lezárt teljes egységként viselkedik, ezért a fordítóprogram önállóan is le tudja fordítani. Azonban ha a POU egy másik blokkot is hív, akkor ezekre az információkra is szüksége van.

2.5.1.1. A POU felépítése

A POU felépítése a 18. ábrán látható. Részei a következők:

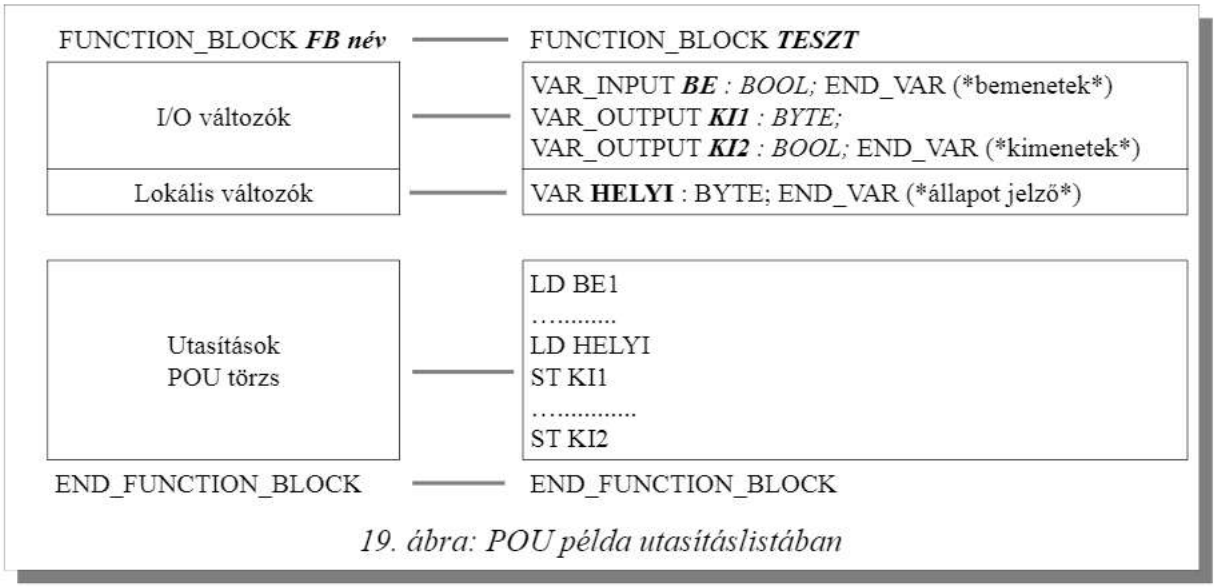
- POU típusa és neve
- Leíró rész a változók deklarálásával
- POU törzs az utasításokkal

A leíró rész definiálja az összes változót, amelyet a POU használ. Megkülönböztetünk olyan változókat, amelyek a POU helyi változói és olyanokat, amik kívülről is látszanak (lokális és globális változók). A kód rész (POU törzs) tartalmazza a logikai áramkör vagy algoritmus programját, amelyet a kívánt programozási nyelven megírtunk.



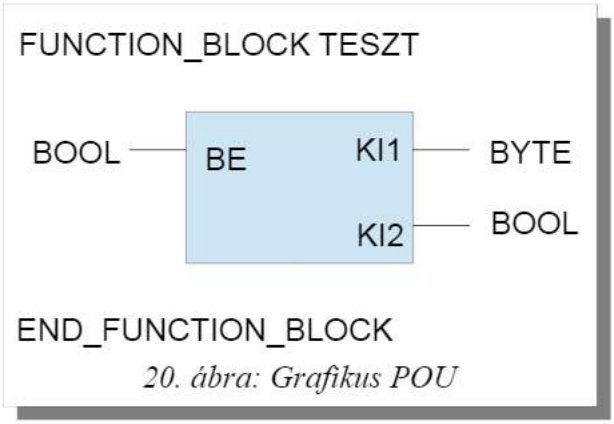
18. ábra: A POU felépítése

Példa:



19. ábra: POU példa utasításlistában

Bal oldalon láthatóak a POU elemek, jobboldalon pedig a megírt utasításlistás program. Az FB tartalmaz egy bemeneti paramétert ***BE***, két visszatérési értéket (***KI1*** és ***KI2***), valamint egy helyi változót: ***HELYI***. Az LD (LOAD - betöltés) és az ST (STORE - tárolás) operátorok (mnemonikok) foglaltak az utasításlista számára.



20. ábra: Grafikus POU

A következő ábrán a ***TESZT*** nevű

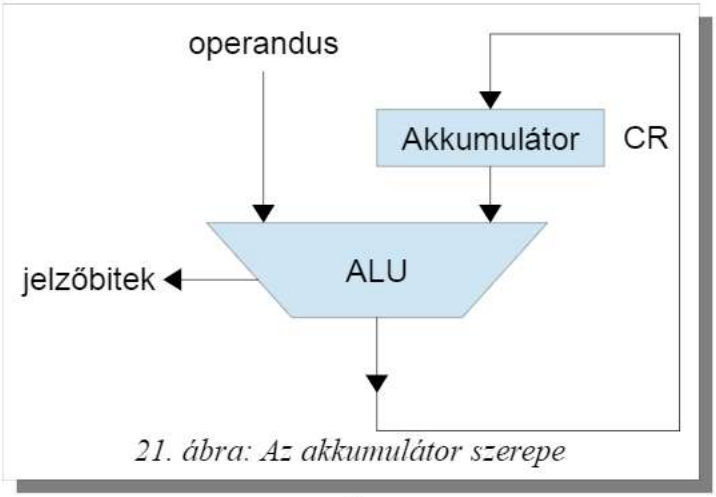
függvényblokk grafikus megjelenését láthatjuk. Ebben az esetben az FB lokális változója (**HELYI**) nem látszik.

2.5.2. Programnyelvek

Az IEC 61131-3 a következő programnyelveket definiálja:

- Szöveges programnyelvek
  - Utasításlista (IL – Instruction List)
  - Strukturált szöveg (ST – Structured Text)
- Grafikus programnyelvek
  - Létradiagram (LD – Ladder Diagram)
  - Funkció Blokk Diagram (FBD – Function Block Diagram)
  - Szekvenciális folyamatábra (SFC – Sequential Function Chart)

A programnyelvek bemutatása előtt érdemes megismerkedni néhány alapvető elvvel. A PLC is tulajdonképpen egy speciális mikroszámítógép, ennek megfelelően úgynevezett egycímes gép[20]. Egy utasításban mindig csak egy operandusra hivatkozhatunk. Két operandus estén az egyik operandus mindig az akkumulátorban van és a művelet elvégzése után ide kerül az eredmény is. Az operandus behozása az akkumulátorba a LOAD utasítással történik, míg az eredmény kiírására a STORE utasítást használjuk. PLC-k esetén az akkumulátor aktuális értékére a CR (Current Result) szimbólummal hivatkozunk. Érdemes megemlíteni, hogy létezik olyan PLC is, amelyben több akkumulátor található. A műveletek elvégzése után állítódnak a jelzőbitek (FLAG) is (Zero, Carry, stb.), így az összehasonlító utasításokat (<, >, stb.) közvetlenül a művelet elvégzése után kell használnunk.



21. ábra: Az akkumulátor szerepe

A programnyelvek ismertetése a teljesség igénye nélkül készült, ugyanis ennek terjedelme meghaladná a lehetőségeket. Azt is érdemes tisztázni, hogy a különböző gyártók által készített programozó szoftverek jelentősen különböznek egymástól és az IEC szabványtól is eltérhetnek (pl. a Siemens S7 típusú PLC-k szoftverei 3-nál több POU-t használnak). Az egyes fejlesztőkörnyezetek

20 Ezekről bővebb információ itt található: http://plc.mechatronika.hu/piclei/mikrovez_okt_honlap.pdf

finomságait az adott programon belül kell kitapasztalnunk, azonban ha erős alapokkal rendelkezünk, akkor könnyű eligazodnunk a szoftverek dzsungelében!

Közös elemek a programnyelvekben:

- ·TYPE...END_TYPE;
- ·VAR...END_VAR;
- ·VAR_INPUT...END_VAR;
- ·VAR_OUTPUT...END_VAR;
- ·VAR_IN_OUT...END_VAR;
- ·VAR_EXTERNAL...END_VAR;
- ·FUNCTION...END_FUNCTION;
- ·FUNCTION_BLOCK...END_FUNCTION_BLOCK;
- ·PROGRAM...END_PROGRAM;
- ·STEP...END_STEP;
- ·TRANSITION...END_TRANSITION;
- ·ACTION...END_ACTION.

2.5.2.1. Utasításlista (IL)

Az utasításlista az assembly-hez hasonló alacsonyszintű programozási nyelv, felépítése nagyon hasonló hozzá. Az utasításoknak, a változóknak és a program különböző belépési pontjainak szimbolikus neveket adhatunk. Az utasítások neveit itt is mnemonikokkal (itt gyakran operátornak nevezik) rövidítik. A program ugyanúgy sorokból áll, mint az assembly, felépítése a következő:

Címke	Művelet	Operandus	Megjegyzés
START:	LD ANDN ST	%IX1 %MX5 %QX2	(*nyomógomb*) (*tiltó feltétel*) (*motor be*)

2. táblázat: Az utasításlista mezői

```
VAR
        Operandus1, Operandus2, Eredmeny : INT :=0;
END_VAR
CIMKE1:  LD      Operandus1  (* CR←Operandus1, aminek a kezdőértéke 0 *)
         ADD     10          (* CR←CR+10, azaz CR=10 *)
         ST      Eredmeny    (* Eredmeny←CR, a CR értéke változatlan *)
         GT      0           (* CR>0? – igen, akkor CR:= IGAZ* )
         JMPC    CIMKE2      (* ugrás a CIMKE2-re ha CR = IGAZ, CR nem változik *)
         ADD     Operandus2  (* CR←CR+Operandus 2– Hiba!!! – az adattípus rossz  *)
CIMKE2:  LD      X1          (* CR←X1 *)
         MUL(    X2          (* CR←X2 *)
          SUB(   X3          (* CR←X3 *)
           ADD   X4          (* CR←X3+X4 *)
          )              (* CR←X2–(X3+X4) *)
```

```
)                (* CR←X1·(X2–(X3+X4)) *)
ST        Y      (* CR értéke nem változik *)
```

A fenti példából látható, hogy a műveletek elvégzésének sorrendje itt kötött. Egyik részről itt is működik az a szabály, amit a matematikából ismerünk, hogy vannak magasabb rendű műveletek (precedencia szabály), illetve itt most egy összeget kellett kivonni. Azért, hogy helyes eredményt kapjunk ezekben az esetekben zárójelet kell alkalmazni. A példából kiderül, hogy a nyitó zárójel – (– a mnemonikok végén tulajdonképpen egy LOAD utasításnak felel meg. Ilyenkor az akkumulátorba a megcímzett operandus kerül. Ehhez az egymásba ágyazáshoz természetesen több

S. sz.	Operátor	Módosítás	Operandus típusa	Leírás
1.	LD	N	*	A CR felveszi az operandus értékét (LOAD)
2.	ST	N	*	Elmenti a CR a megadott címre (STORE)
3.	S R		BOOL BOOL	Ha CR=1, akkor az operandus értéke 1 lesz Ha CR=1, akkor az operandus értéke 0 lesz
4.	AND	N, (, N(	BOOL	ÉS
5.	OR	N, (, N(	BOOL	VAGY
6.	XOR	N, (, N(	BOOL	kizáró-VAGY
7.	ADD	(	*	Összeadás (ADDition)
8.	SUB	(	*	Kivonás (SUBtraction)
9.	MUL	(	*	Szorzás (MULtiplication)
10.	DIV	(	*	Osztás (DIVision)
11.	GT	(	*	Összehasonlítás: **CR>operandus** (Greater Than)
12.	GE	(	*	Összehasonlítás: **CR>operandus** (Greater than) or Equal
13.	EQU	(	*	Összehasonlítás: **CR=operandus** (Equal)
14.	NE	(	*	Összehasonlítás: **CR<>operandus** (Not Equal)
15.	LE	(	*	Összehasonlítás: **CR<=operandus** (Lesser than Equal)
16.	LT	(	*	Összehasonlítás: **CR<operandus** (Lesser Than)
17.	JMP	C, CN	LABEL	Ugrás a megadott címkére (JuMP)
18.	CAL	C, CN	NAME	Függvényblokk hívása (CALl)
19.	RET	C, CN		Visszatérés a hívott rutinból (FB vagy FUN) (RETurn)
20.	)			A függőben maradt műveletek lezárása
* – ezek az operátorok vagy felülírják az adattípust az akkumulátorban (tetszés szerinti típusra), vagy meg kell egyezzenek az akkumulátor adattípusával				

3. táblázat: Az utasításlista operátorai (mnemonikok)

akkumulátor szükséges! Léteznek olyan PLC-k, ahol erre nincs lehetőség, ilyenkor az egyes részeredményeket nekünk kel elmenteni egy belső változóba (M – Marker). Az utasításlista

operátorai a 3. táblázatban, a standard függvényblokkok pedig a 4. táblázatban találhatóak.

S. sz.	Operátor	Funkció blokk (FB)	Leírás
1.	S1, R	SR	Set domináns RS tároló.[21]
2.	S, R1	RS	Reset domináns RS tároló.
3.	CLK	R_TRIG	Felfutó él detektálása.
4.	CLK	F_TRIG	Lefutó él detektálása.
5.	CU, R, PV	CTU	Előreszámláló.
6.	CD, LD, PV	CTD	Hátraszámláló.
7.	IN, PT	CTUD	Előre-hátra számláló.
8.	IN, PT	TP	Impulzus időzítő.
9.	IN, PT	TON	Bekapcsolást késleltető.
10.	IN, PT	TOF	Kikapcsolást késleltető.

4. táblázat: Standard függvényblokkok az IL-ben

Példák FB hívásra különféle módokon:

VAR C10 : CTU; END_VAR

1. Hívás a bemeneti paraméterek felsorolásával:
   CAL C10(CU:=%I10, R:= %I11, PV:=15)
   ….....
2. Hívás Load és Store utasításokkal (paraméter hozzárendelés):
   LD 15
   ST C10.PV
   LD %I10
   ST C10.CU
   LD %I11
   ST C10.R
   CAL C10
   .....
3. Bemeneti operátorok segítségével (implicit):
   LD 15
   PV C10
   LD %I1
   CU C10
   LD %I11
   R C10
   CAL C10
   .....

A kimenetek hozzárendelése mindhárom példa esetén ugyanaz:

```
LD      C10.Q
ST      %QX1          (* a számláló kimenete kapcsolja %QX1 kimenetet *)
LD      C10.CV
ST      AktualisErtek (* A számláló tartalma az AktualisErtek változóba kerül, amelyet
                         INT típusúként kell megadni *)
```

21 Az S és az R bemenet értéke egyszerre nem lehet 1, ha ez mégis előfordulna a PLC-ben valamelyik állapotot előnyben részesítik.

2.5.2.2. Strukturált szöveg (ST)

Ez a programozási nyelv a magasszintű szöveges nyelvek közé tartozik (pl. C nyelv) annak minden előnyével és hátrányával. A programozó szempontjából kényelmesebbnek tűnik használata, a fejlesztési idő lerövidülhet. A magasszintű absztrakció miatt a lefordított kód sokkal redundánsabb[22] lesz, mint az utasításlistában írt. Alkalmazásánál figyelembe kell venni a PLC sajátosságait (pl. egy rosszul megírt ciklus Watchdog hibát okozhat).

A strukturált szöveg operátorait a következő táblázat foglalja össze:

S. sz.	Szimbólum	Művelet	Példa
1.	(kifejezés)	Zárójeles műveletek	(X+Y)*(X-Y)
2.	Fgv_név (argumentumok)	Függvény kiértékelés	LN(A), MAX(X, Y)
3.	**	Exponens	X**Y
4. 5.	– NOT	Ellentett Komplemens	–10, –X NOT (X > Y)
6. 7. 8.	* / MOD	Szorzás Osztás Modulo	X * Y X / Y 13 MOD 10 (Eredmény: 3)
9. 10.	+ –	Összeadás Kivonás	X + Y X – Y
11.	<, >, <=, >=	Összehasonlítás	T#1h > T#30m (Eredmény: TRUE)
12. 13.	= <>	Egyenlőség Egyenlőtlenség	T#1d = T#24h (Eredmény: TRUE)
14.	AND vagy &	Logikai ÉS	(X >Y) AND (X < Z)
15.	XOR	Logikai kizáró-VAGY	TRUE XOR FALSE
16.	OR	Logikai VAGY	TRUE OR FALSE

5. táblázat: ST operátorok

22 Ez azzal jár, hogy a program lassabb lesz és több helyet foglal a memóriában. Nem véletlen, hogy a PLC-k, mikrovezérlők, stb. belső memóriája kB-ról MB-ra növekedett. Próbálkozzon csak meg valaki assembly-ben 8 kB memóriát teleírni, el fog csodálkozni rajta, hogy milyen nehéz!

Az ST nyelv elemei (6. táblázat):

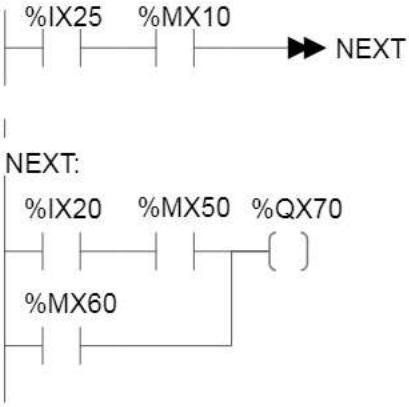
S. sz.	Szerkezet	Példa
1.	Hozzárendelés	A:=B; CV:=CV+1; Y:=SIN(X); D:=INT_TO_REAL(C)
2.	FB hívás FB kimenet használata	My_TMR(IN:=%IX5, PT:=T#300ms); A:=My_TMR.Q;
3.	RETURN	RETURN;
4.	IF	D:=B*B-4*A*C; IF D < 0.0 THEN NROOTS:=0; (* nincs megoldás *) ELSIF D = 0.0 THEN (* egy megoldás *) NROOTS := 1; X1 := –B / (2.0 * A); ELSE (* két megoldás *) NROOTS := 2; X1 := (–B + SQRT(D)) / (2.0 * A); X2 := (–B – SQRT(D)) / (2.0 * A); END_IF;
5.	CASE	*ERROR:=0;* *XW:=BCD_TO_INT(Y);* *CASE XW OF* *1,4: DISPLAY := TEKST1;* *2: DISPLAY := TEKST2;* *Y := SIN(Z);* *3,5..10: DISPLAY := STATUS (XW – 3);* *ELSE DISPLAY := ''; (* XW tartományon kívül(1..10) *)* *ERROR := 1;* *END_CASE;*
6.	FOR	*J := 101;* *FOR I := 1 TO 100 BY 2 DO* *IF WORDS(I) = 'KEY' THEN* *J := I;* *EXIT;* *END_IF;* *END_FOR;*
7.	WHILE	J := 1; WHILE J <= 100 AND WORDS(J) <> 'KEY' DO J := J+2; END_WHILE;
8.	REPEAT	J := –1; REPEAT J := J+2; UNTIL J = 101 OR WORDS(J) = 'KEY' END_REPEAT;
9.	EXIT	EXIT;

6. táblázat: Az ST nyelv utasításai

2.5.2.3. Létradiagram (LD)

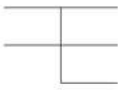
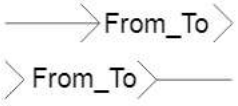
A létradiagrammal már a bevezetőben megismerkedhettünk, ez volt a PLC-k első jellegzetes programnyelve, s még ma is nagy népszerűségnek örvend a gyártók és a programozók körében. A legtöbb PLC-nek ez a fő programozási nyelve. Mivel a kezdetekben a relés logikák kiváltása céljából készült, ezért nagyban hasonlít az áramutas rajzra. Természetesen a program grafikája és a felhasználható elemek száma nagyon sokat fejlődött az idők folyamán.

Vezérlésátadó utasítások:

S. sz.	Szimbólum/Példa	Magyarázat
1.		Feltétel nélküli ugrás a CIMKE1-re.
2.		Feltételes ugrás a NEXT címkére. Ugrási cél.
3.		Visszatérés a POU-ból.
4.		Feltételes visszatérés a POU-ból.

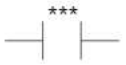
7. táblázat: Vezérlésátadó szimbólumok létradiagramban

Tápsínek és összekötő elemek:

S. sz.	Szimbólum	Leírás
1.		Baloldali tápsín (vízszintes csatlakozással).
2.		Jobboldali tápsín (vízszintes csatlakozással).
3.		Vízszintes összekötő.
4.		Függőleges csatlakozás (vízszintes csatlakozással.
5.		Csatlakozó. Összeköttetés folytatása.

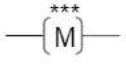
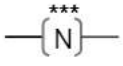
8. táblázat: Tápsínek és összekötő elemek

Érintkezők (kontaktusok):

Érintkező	Szimbólum	Leírás
Statikus kontaktus		Alaphelyzetben nyitott érintkező. A kontaktus záródik, ha a hozzárendelt logikai változó (***) értéke 1.
		Alaphelyzetben zárt érintkező. A kontaktus záródik, ha a hozzárendelt logikai változó (***) értéke 0.
Élfigyelő kontaktus		Felfutó élt figyelő bemenet. A kontaktus záródik, ha a hozzárendelt logikai változó 0-ról 1-re vált.
		Lefutó élt figyelő bemenet. A kontaktus záródik, ha a hozzárendelt logikai változó 1-ről 0-ra vált.

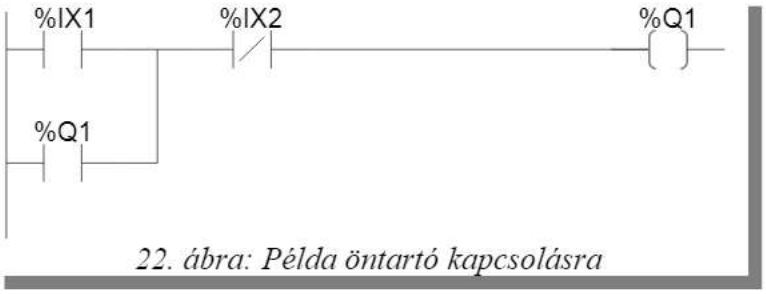
9. táblázat: Érintkező, kontaktusok

Kimenetek (relék):

Kimenet	Szimbólum	Leírás
Normál kimenet		A kimenet logikai változó (***) a baloldalán lévő vezeték állapotát veszi fel és átmásolja a jobboldali vezetékre.
		A jobboldali vezeték a baloldali állapotát veszi fel, míg a hozzárendelt változó az inverzét.
Reteszelt kimenet		Kimenet 1-be billentése (SET). Amennyiben a baloldali vezeték értéke 1 lesz a hozzárendelt logikai változó értéke 1-be billen és megőrzi az állapotát a RESET utasításig.
		Kimenet 0-ba billentése (RESET). Amennyiben a baloldali vezeték értéke 1 lesz a hozzárendelt logikai változó értéke 0-ba billen és úgy is marad a SET utasításig.
Emlékező kimenet		Emlékező (memória) kimenet.
		Emlékező SET kimenet.
		Emlékező RESET kimenet.
Élvezérelt kimenet		Felfutó élt figyelő kimenet. A hozzárendelt logikai változó értéke 1 lesz, ha a baloldali vezeték állapota 0-ról 1-re vált. A baloldali vezeték állapota átmásolódik a jobb oldalra.
		Lefutó élt figyelő kimenet. A hozzárendelt logikai változó értéke 1 lesz, ha a baloldali vezeték állapota 1-ről 0-ra vált. A baloldali vezeték állapota átmásolódik a jobb oldalra.

10. táblázat: Kimenetek

Példa:

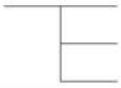
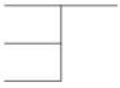


22. ábra: Példa öntartó kapcsolásra

2.5.2.4. Funkció blokk diagram (FBD)

Szintén grafikus programozási nyelv. Leginkább a digitális technikában megszokott kapcsolási rajhoz hasonlít, hiszen szimbólumai ugyanolyanok, vagy ahhoz hasonlóak.

Összeköttetések (huzalozás) az FBD-ben:

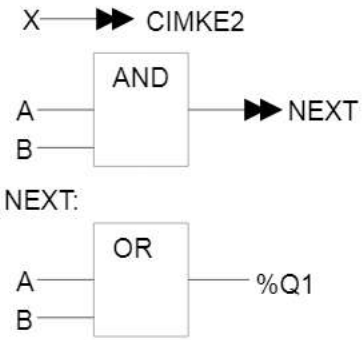
S. sz.	Szimbólum	Leírás
1.		Vízszintes összekötő vezeték.
2.		Függőleges leágazás vízszintes kapcsolódással (elosztás).
3.		Tiltott összeköttetés (huzalozott VAGY)*.
4.		Vezetékek keresztezése összekötés nélkül**.

* Normál esetben a logikai kapuk kimenetei nem köthetők egymással össze, hiszen ha pl. az egyik kimenete 1-ben, a másiké pedig 0-ban van, akkor a két kimenet között akkora áram folyhat, ami tönkre teszi a kapu végtranzisztorait.

** Az elektronikában ez nem használatos, az összekötést csomóponttal jelölik. Ezen ok miatt a legtöbb FBD program is az elektronikában megszokott csomópontot használja!

11. táblázat: Összekötő elemek az FBD-ben

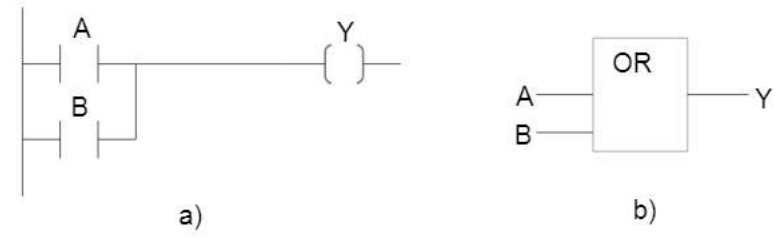
Vezérlésátadó utasítások az FBD-ben:

S. sz.	Szimbólum	Leírás
1.		
2.		Feltételes ugrás (ha X=1) a CIMKE1-re. Példa a feltételes ugrásra: a feltétel akkor teljesül, ha minkét bemenet (A és B) értéke logikai 1. Ugrási cél (NEXT címke jelöli)
3.		Feltételes visszatérés függvényből vagy függvény blokkból
4.	END_FUNCTION END_FUNCTION_BLOCK	Feltétel nélküli visszatérés függvényből Feltétel nélküli visszatérés függvény blokkból

12. táblázat: Vezérlésátadó utasítások FBD-ben

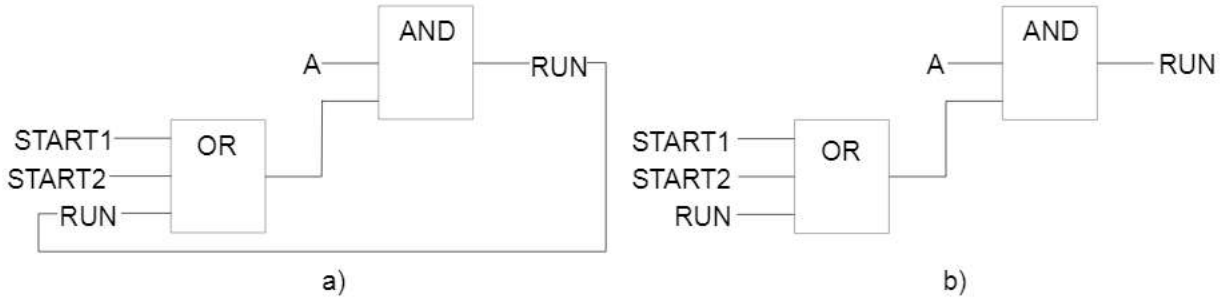
Példák:

- Logikai VAGY kapcsolat



23. ábra: Logikai függvények: a) "huzalozott VAGY" (LD); b) VAGY kapu (FBD)

- FBD hálózat visszacsatolással[23]



24. ábra: a) közvetlen visszacsatolás b) közvetett visszacsatolás

2.5.2.5. Szekvenciális folyamatábra (SFC)

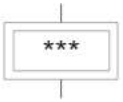
Ebben az estben tulajdonképpen egy folyamatábrát kell készítenünk. A program különböző szekvenciákra (lépésekre) bomlik. Az egyes lépésekből (step) más lépésekbe eljutni valamilyen átmenet (transition) árán lehetséges. Az egyes lépésekhez tartozó elvégzendő feladatokat akcióknak nevezzük (action). A programhoz tartozó feltételeket és akciókat a már megismert programnyelv valamelyikével írhatjuk meg (ST, LD, stb.). A folyamatábrát megszerkeszthetjük grafikusan[24] vagy megírhatjuk szövegesen. A lépések között van egy kitüntetett lépés (initial step), ahonnan a program indul. Az éppen aktív szekvenciáról az úgynevezett lépésmarker (X) ad felvilágosítást. Alapvetően egy feltétel teljesülésekor az előző lépés inaktív lesz (a hozzá tartozó lépésmarker nulla lesz), a következő aktív. Ez alól csak a szimultán elágazás jelent kivételt, ilyenkor párhuzamosan fut több ág. A következőkben tekintsük át az SFC legfontosabb elemeit.

---

23 Létezik olyan PLC szoftver, ahol a kimenetet nem tudjuk visszacsatolni a bemenetre. Ilyenkor egy belső változón (marker) keresztül tehetjük ezt meg.

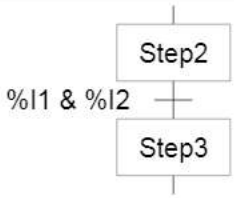
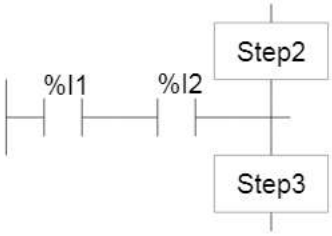
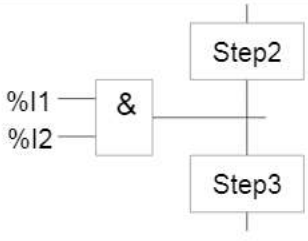
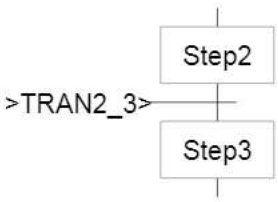
24 A grafikus jóval áttekinthetőbb, szimulátorban nagyon jól követhetőek az egyes lépések.

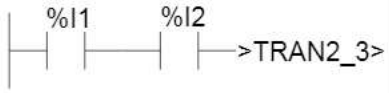
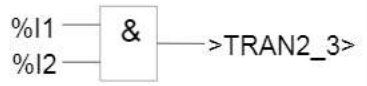
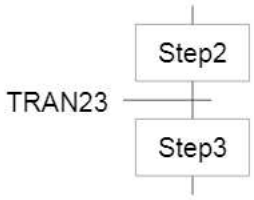
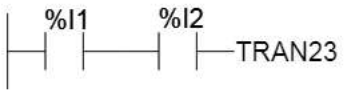
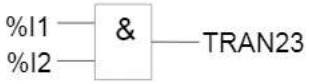
Lépések és feltételek:

S. sz.	Grafikus	Szöveges	Leírás
1.		STEP ***: (* lépés törzs *) END_STEP	Lépés a **** a lépés nevét jelenti
		INITIAL_STEP ***: (* lépés törzs *) END_STEP	Kezdő lépés A felső csatlakozási vonal nem szükséges, ha nincs előző lépés
2.	***.X		Lépésmarker (általános formula).
			Lépésmarker: közvetlen csatlakozású logikai változó (***.X) a jobb oldalon.
3.	***.T		Eltelt idő (általános formula). ***.T , idő (TIME) típusú változó

13. táblázat: Lépések és feltételek SFC-ben

Átmenetek és átmeneti feltételek grafikusan:

S. sz.	Példa	Leírás
1.		Előző lépés Feltétel ST nyelven Következő lépés
2.		Előző lépés Feltétel létradiagramban (LD) Következő lépés
3.		Előző lépés Feltétel létradiagramban (LD) Következő lépés
4.		Csatlakozás használatával: Előző lépés Átmenet csatlakoztatása Következő lépés

S. sz.	Példa	Leírás
4. a		Feltétel LD-ben készítve.
4. b		Feltétel FBD-ben.
5.		Elnevezés használatával: Előző lépés Átmenet csatlakoztatása Következő lépés
5. a		Feltétel LD-ben.
5. b		Feltétel FBD-ben.
5. c	*TRANSITION TRAN23:* *LD %I1* *AND %I2* *END_TRANSITION*	Feltétel IL-ben.
5.d	*TRANSITION TRAN23:=* *%I1 & %I2;* *END_TRANSITION*	Feltétel STL-ben.

14. táblázat: Átmenetek és feltételek grafikusan

Átmenetek és átmeneti feltételek szövegesen:

STEP STEP2:

... (Declaration of STEP2 *)*

END_STEP

(Declaration of transition from STEP2 to STEP3 *)*

TRANSITION FROM STEP2 TO STEP3:=

(Transition condition in ST *)*

%I1 & %I2;

(Transition condition in IL *)*

LD %I1

AND %I2

END_TRANSITION (End of transition declaration *)*

STEP STEP3:

*... (*Declaration of STEP3 *)*

END_STEP

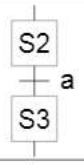
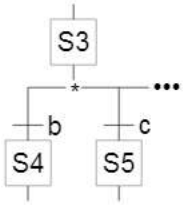
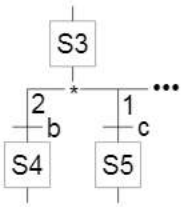
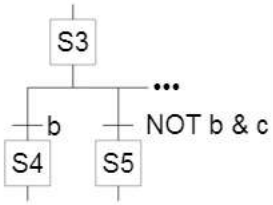
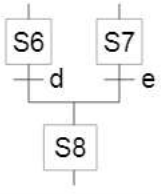
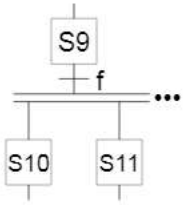
Többszörös megadás:

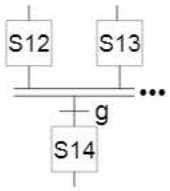
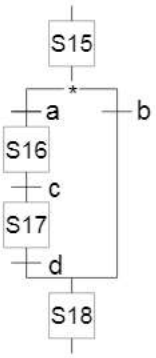
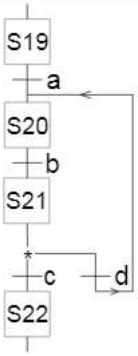
TRANSITION FROM (S1, S2, S3) TO (S4,S5):=

(Boolean expression for transition conditions *)*

END_TRANSITION

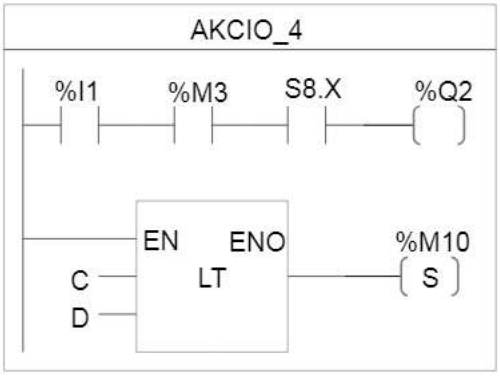
Szekvenciák (lépések) felépítése:

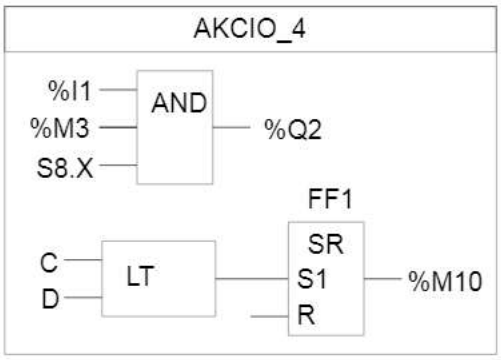
S. sz.	Példa	Szabály
1.		*Egyszerű szekvencia:* A lépések sorban egymás után következnek. Példa: az S2-ből akkor lépünk csak az S3-ba, ha az S2 aktív és a feltétel (a) igaz.
2. a		*Elágazás prioritással:* Az elágazások a vízszintes vonal alá kapcsolódnak, amelyek különböző ágakat jelölnek. Az elágazásban a * jelölés azt jelenti, hogy az ágak prioritása balról jobbra tart (jobboldali a magasabb). Példa: S3-ból S4-be akkor lépünk, ha S3 aktív és a „b” feltétel igaz. S3-ból S5-be viszont csak akkor juthatunk, ha S3 aktív, a „c” feltétel igaz és „b” hamis!
2. b		*Elágazás prioritással megadással:* Az elágazások itt is a vízszintes vonal alá kapcsolódnak. Az ágak prioritását a felhasználó adja meg, a legkisebb sorszámú ág prioritása a legnagyobb. Példa: S3-ból S5-be akkor lépünk, ha S3 aktív és a „c” feltétel igaz. S3-ból S4-be viszont csak akkor juthatunk, ha S3 aktív, a „b” feltétel igaz és „c” hamis!
2. c		*Elágazás prioritás nélkül:* Ebben az esetben nincs prioritási sorrend az ágak között. A felhasználónak kell gondoskodnia arról, hogy a feltételek kölcsönösen kizárják egymást. Példa: S3-ból léphetünk S4-be, ha S3 aktív és „b” igaz; illetve léphetünk S5-be ha S3 aktív, valamint „b” hamis és „c” igaz.
3.		*Szekvenciák egyesítése:* A vízszintes vonal alatt az ágak újra egyesülnek. Példa: Az S6 ágon akkor lépünk S8-ra, ha S6 aktív és a „d” feltétel igaz. Az S7 ágon akkor lépünk S8-ra, ha S7 aktív és az „e” feltétel igaz.
4.		*Szimultán elágazás:* Egy átmenettel egyszerre több párhuzamos ágat is futtathatunk, amelyek a dupla vízszintes vonal alatt találhatók. Példa: S9-ből lépünk S10, S11, … lépésbe, ha S9 aktív és az „f” feltétel igaz. Ezután a két ág egymástól függetlenül fut le.

S. sz.	Példa	Szabály
5.		*Szimultán szekvenciák egyesítése:* A dupla vonal feletti szimultán ágak újraegyesülnek egy átmenet hatására. Példa: az S12, S13, … ágakból az S14 szekvenciára lép a program akkor, ha a dupla vonal felső részére csatlakozó összes lépés aktív és a közös „g” feltétel igaz.
6. a 6. b 6. c		*Szekvenciák átlépése:* Ez egy speciális fajtája az elágazásnak (lásd 2-es pont). Ebben az esetben 1 vagy több ág nem tartalmaz lépést. A 6. a, 6. b és 6. c jellemzői a 2. a, 2. b és 2. c esetekével azonos. Példa (6. a): Az S15-ből az S18-ba akkor lépünk, ha S15 aktív és az „a” feltétel hamis a „b” pedig igaz. Ilyenkor az S16 és S17 szekvencia kimarad (átlépjük).
7. a 7. b 7. c		*Hurok szekvencia:* Ez is egy speciális fajtája az elágazásnak (lásd 2-es pont). Ilyenkor egy vagy több ág visszatér az előző lépésbe. A 7. a, 7. b és 7. c jellemzői a 2. a, 2. b és 2. c esetekével azonos. Példa (7. a): Az S21-ből az S20-ba lépünk, ha a „c” feltétel hamis és a „d” feltétel igaz. Ebben az esetben az S20 és S21 lépés ismétlődik.

15. táblázat: Szekvenciák felépítése

Akciók megadása:

S. sz.	Példa	Leírás
1.		Grafikus deklaráció LD-ben.

S. sz.	Példa	Leírás
2.		Grafikus deklaráció FBD-ben.
3.	*ACTION AKCIO_4:* *%Q2 := %I1 & %M3 & S8.X;* *FF1(S1 := (C < D));* *%M10 := FF1.Q;* *END_ACTION*	Szöveges deklaráció ST-ben.
4.	*ACTION AKCIO_4:* *LD S8.X* *AND %I1* *AND %M3* *ST %Q2* *LD C* *LT D* *S1 FF1* *LD FF1.Q* *ST %M10* *END_ACTION*	Szöveges deklaráció IL-ben.
5.		Akció kizárólag SFC elemekkel.

16. táblázat: Akciók megadása

3. Mellékletek

Irodalomjegyzék

1: Vanessa Romero Segovia and Alfred Theorin, History of Control History of PLC and DCS, 2012,

2: Alison Dunn, The father of invention: Dick Morley looks back on the 40th anniversary of the PLC , 2008, http://www.automationmag.com/features/the-father-of-invention-dick-morley-looks-back-on-the-40th-anniversary-of-the-plc.html