PLC és SCADA rendszerek

Pletl Szilveszter, Kincses Zoltán

2014

A tananyag a TÁMOP-4.1.2.A/1-11/1-2011-0104 "A felsőfokú informatikai oktatás minőségének fejlesztése, modernizációja" c. projekt keretében a Pannon Egyetem és a Szegedi Tudományegyetem együttműködésében készült.

PLC és SCADA rendszerek

Egyetemi tananyag

SZTE
Pletl Szilveszter, Kincses Zoltán

Lektor:
Dr. Jeges Zoltán, főiskolai tanár

Tartalom

Bevezető

A PLC-k az ipari automatizálás egyik alappillérét képezik. A megoldások nagy hányada tartalmaz SCADA felületet is. A mérnökök és azon belül a mérnök-informatikusok nagy szerepet játszanak az egyes megoldások tervezése, kivitelezése és működtetése terén. A mérnök-informatikus szakemberek esetében elengedhetetlen az ipari informatika témakörébe tartozó kompetenciák megszerzése. A PLC-k és SCADA rendszerek tárgy a mérnök-informatikus alapszakos hallgatók számára a legtöbb felsőoktatási intézményben alapozó és kötelező tárgyként szerepel a tantervben. A témakörben számos, jó minőségű tankkönyv, jegyzet és példatár készült. Jelen jegyzet célja a hallgatók számára röviden bemutatni a szükséges elméletet. A PLC-k és SCADA rendszerek területe nagyon széles, az egyes elemek különböző mélységgel tárgyalhatók, jelen tankönyv a témakör lefedését tekintve nem törekszik a teljességre, az alapképzésben használható, nem túl mély elméleti tárgyalásmódot alkalmazza. Az első fejezet tárgyalja az irányítástechnika azon alapfogalmait, melyek elengedhetetlenül szükségesek a témakör, rendszertechnikán belüli betájolására. A második fejezet kitér a programozható vezérlők felosztására és architektúrájára. A harmadik fejezet foglalkozik a programozható vezérlők bemeneteivel és kimeneteivel. A negyedik fejezet tartalmazza az IEC 61131 szabvány lényeges elemeit. Az ötödik fejezet foglalkozik a PLC-k programozásával a programozási paradigmákkal. Az ipari hálózatok alapjait és a PLC-k hálózatba kötésének lehetőségeit a hatodik fejezet tartalmazza. A SCADA rendszerek helye és szerepe az integrált informatikai rendszeren belül a hetedik fejezetben kerül bemutatásra. Az OPC kommunikációs protokoll a nyolcadik fejezetben kapott helyet. Az esettanulmányok pedig az utolsó részben találhatók.

1. Irányítástechnikai alapfogalmak

A magyar nyelvű szakirodalom, az angol nyelvű irodalommal ellentétben az irányításokon belül megkülönböztet szabályozást és vezérlést. A szabályozások esetében az irányításhoz szükséges információ legalább egy részét a szabályozott jellemző érzékelésével nyerjük, vagyis van visszacsatolás az irányításon belül a szabályozott szakasz kimenetéről, míg a vezérlések esetében nincs visszacsatolás.

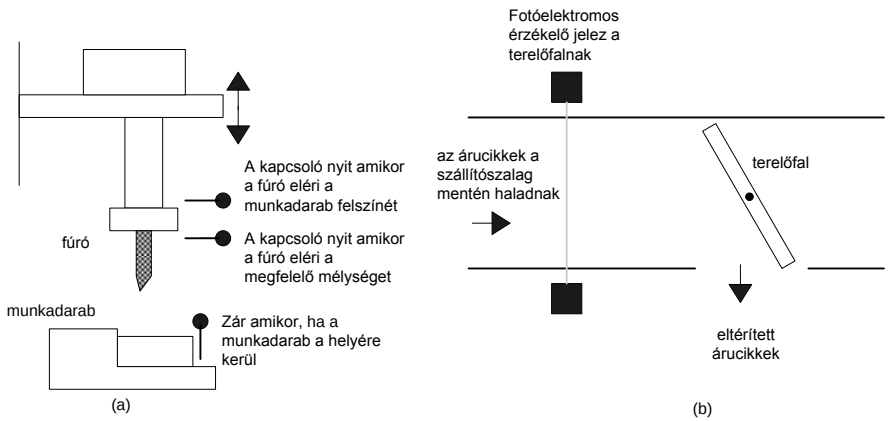
Tipikus irányítási feladatoknál szükség lehet bizonyos eseménysorozatok irányítására, vagy fenntartani bizonyos rendszerjellemző értékeket, vagy követni valamilyen előre meghatározott pályát. Így megkülönböztethetők: időterv-, lefutó- és követővezérlések.

Időterv-vezérlés esetén a rendszerbe való beavatkozást az idő vagy az időtől függő program befolyásolja. Időterv-vezérlést alkalmazunk például az utcai világítás vezérlésénél. Az irányító, azaz a vezérlőberendezés minden nap egy adott időben bekapcsolja, és egy adott időben kikapcsolja az utcai világítást függetlenül attól, hogy már sötét, vagy világos van–e. Ahhoz, hogy a be – vagy kikapcsolás idejét megváltoztassuk, a vezérlési programban bizonyos beavatkozást kell elvégeznünk.

A lefutó-vezérlésben az irányított rendszerben vezérlendő folyamat egyes szakaszai csak akkor kezdődhetnek meg, ha az előző szakaszok már lejátszódtak. Tehát a folyamattól függő egyes feltételek fennállása indítja meg a folyamat további szakaszát. Lefutó-vezérlést alkalmazunk például a klasszikus mosógépekben. Az indítógomb megnyomása után, feszültséget kap a vízbeeresztő elektromágneses szelep és megindul a víz a tárolóba. Ha a víz elérte a kívánt szintet, akkor bezárjuk a szelepet és megindítjuk a forgómotort stb…

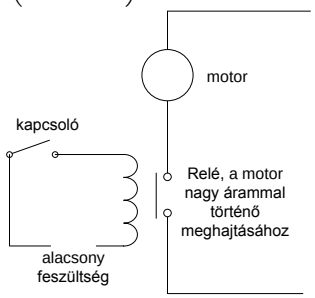
Követővezérlésben a beavatkozó szerv az irányító rendszertől előre megállapítható módon függő parancsot kap, és az irányított rendszert a kívánt irányba befolyásolja. A már említett utcai világítás esetében, az utca megvilágítását, mérjük. Ha ez egy érték alá vagy fölé esik, függetlenül attól, hogy hány óra van, a világítást bekapcsoljuk vagy kikapcsoljuk.

Egy automata fúróberendezés (1.1.a. ábra.) irányítási rendszere esetében például szükséges leengedni a fúrót, amikor a munkadarab a megfelelő pozícióban van, elkezdeni a fúrást, amikor a fúró eléri a munkadarabot, megállítani a fúrót, amikor a lyuk elérte a megfelelő mélységet, visszahúzni a fúrót és várni a következő munkadarab megérkezését, hogy az egész művelet újra elkezdődhessen. Egy másik irányítási rendszer lehet a futószalagon haladó mozgó árucikkek számát szabályozó, és a csomagoló részleg felé továbbító rendszer (1.1.b. ábra.). Egy ilyen irányítási rendszer esetében a rendszer bemenetei kapcsolók, melyek nyitott vagy zárt állapotban lehetnek, például ha a munkadarab elhalad egy kapcsoló előtt, amely ennek hatására zár, akkor ezzel jelezhető, hogy a munkadarab a megfelelő pozícióba érkezett. A vezérlő a bemenetek hatására a kimeneteihez kapcsolt eszközöket például motort, mely egy tárgyat a megfelelő pozícióba mozgat, egy szelepet, vagy egy fűtőszálat be vagy kikapcsolhat.



1.1. ábra. – Irányítási feladat példák, (a) automata fúrógép, (b) csomagoló rendszer.

Milyen formája lehet egy vezérlőnek? Az automata fúrógép esetében összeállíthatunk egy olyan áramkört, melyben a különböző kapcsolók, melyek a munkadarab pozícióját és a fúrás mélységét jelzik, indíthatják el a fúró motorját, vagy a mozgató rendszer szelepeit. Pontosabban egy kapcsoló zárásával egy relét aktiválhatunk, mely zárja a fúró motorjának áramkörét, mely ennek hatására elkezdi forgatni a fúrófejet (1.2. ábra).



1.2. ábra. – Egy vezérlési rendszer.

A fúrót vezérlő áramkör jelentősen kisebb energiával vezérli a nagyobb energiaigényű berendezést. Egy másik kapcsoló zárásával pedig egy olyan relét aktiválhatunk, amely egy pneumatikus vagy hidraulikus szelepre kapcsol áramot, amelynek hatására egy munkahenger a megfelelő pozícióba tolja a munkadarabot. Ez az elektromos áramkör csakis speciálisan az automata fúrógép vezérléséhez használható. Az automata csomagológép esetében is egy hasonló elvű áramkört készíthetünk, melyben a különböző kapcsolók hatására indulnak el az egyes motorok, vagy kapcsolnak be az egyes szelepek. Mindkét berendezésre jellemző az, hogy az érzékelők által szolgáltatott információ alapján hozott döntések következtében működésbe lépnek a beavatkozó szervek. Legegyszerűbb esetben a döntéseket hozó logika huzalozott technikával valósítható meg. A hagyományos irányítási rendszerekben az irányítási feladatot az egyes érzékelők (szenzorok) és beavatkozó szervek (aktuátorok) megfelelő összekapcsolásával oldhatjuk meg. Amikor az irányítási feladat megváltozik, akkor a vezetékezést is módosítani kell, például át kell huzalozni. Összetettebb feladatok megoldása esetében nagyszámú jelfogó, vagyis relé alkalmazására van szükség, esetükben a vezérlési logika megváltoztatása nehézkes.

A számítástechnika és az elektronika fejlődése lehetővé tette a mikroprocesszor alapú rendszerek alkalmazását az irányítástechnikában. A mikroprocesszoros megvalósítás során az érzékelők jelei bemeneti illesztő áramkörökön keresztül juttatják el az információt a vezérlési logikát megvalósító mikroprocesszorhoz, a kimeneti illesztő áramkörök pedig biztosítják a kapcsolatot a beavatkozó-szervek irányába. A széleskörű alkalmazhatóság egy speciális architektúra, a mikrovezérlő vagy mikrokontroller kialakulásához vezetett. A mikrokontroller, egyetlen lapkára integrált, általában vezérlési feladatokra optimalizált számítógép.

A mikrovezérlős megvalósítás biztosítja, hogy ahelyett hogy minden egyes irányítási feladathoz egy-egy külön áramkört vezetékeznénk össze, használhatjuk ugyanazt az alap rendszert is a különböző irányítási feladatokhoz. Egy ilyen rendszerben a vezérlési logikát program valósítja meg. A programban megadott utasítások egymás utáni elvégzése során valósul meg a vezérlés, mikrokontroller esetében a párhuzamos információfeldolgozás csak nagyon kis mértékben és különös odafigyelést követően valósítható meg. A bemenetek változásaira, a gyors műveletvégzésnek köszönhetően, a szekvenciális működés is azonnalinak tűnő kimeneti reakciót eredményez. A programozás elsősorban logikai és kapcsolási műveletek megvalósítását jelenti, mint például ha az A és a B esemény bekövetkezik, akkor kapcsoljuk be C-t. A mikrovezérlőhöz kapcsolt bemeneti eszközök, azaz az érzékelők (mint például kapcsolók) által szolgáltatott információk alapján a kimenetéhez kapcsolt eszközöket (mint például motorok, beavatkozók) irányíthatjuk a programozó által a mikrovezérlő memóriájába feltöltött program alapján. A vezérlési logikát megvalósító program tipikus szerkezete a következő:

```
IF A-kapcsoló zár
      Motor áramkör bekapacsol
IF B-kapcsoló zár
      Szelep áramkör bekapcsol
```

A programban található utasításokat megváltoztatva, ugyanazt a mikrovezérlő alapú rendszert használhatjuk az irányítási feladatok széles skálájának megoldásához. Példaként a modern háztartási mosógép is egy mikrovezérlőn alapú rendszer. A rendszer bemenete a mosógép kezelőfelületén található forgó kapcsoló, mellyel ki lehet választani a programot, az ajtó zárását és a vízszintet érzékelő kapcsoló, valamint a hőmérsékletérzékelő, mely a víz hőmérsékletét méri. A mikroprocesszor úgy lett felprogramozva, hogy ezen, bemenetek hatására bekapcsolja/kikapcsolja a mosódob motorját és szabályozza annak sebességét, nyissa/zárja a meleg víz vagy a vízleeresztő csapját, és nyissa/zárja a mosógép ajtajának zárszerkezetét.

Az ipari alkalmazások az irányítástechnikai rendszerek iránt mindig is speciális igényeket támasztottak. A mikrovezérlőkön alapuló irányítások a könnyebb programozhatóság, a jobb illeszthetőség, a robusztusság, a megbízhatóság irányába fejlődtek.

A Programozható Logikai Vezérlő vagy angolul a Programmable Logic Controller (PLV vagy PLC) egy olyan speciális mikrokontroller alapú vezérlő, mely a memóriájában eltárolt program alapján képes logikai, sorrendi, időzítési, számlálási és aritmetikai műveletek elvégzésére. A Programozható Logikai Vezérlő talán a legszélesebb körben alkalmazott eleme az ipari irányítási rendszereknek, segítségükkel főleg vezérlési funkciók valósíthatóak meg. A napjainkban forgalmazott PLC-k zöme azonban alkalmas szabályozási funkciók ellátására is.

Amint már említésre került, a történelmi fejlődés során a mikrokontrollerek és rajtuk keresztül a programozható vezérlők a logikai áramkörök relés megvalósítását váltották fel. Alkalmazásukkal, a relés (huzalozott) logikák esetében nélkülözhetetlen nagy mennyiségű vezetékezés szükségtelenné vált. Emellett a mikrokontrollerekkel szemben a PLC-k előnye a könnyű programozhatóság és szerelhetőség, a nagysebességű irányítás, a hálózatba kapcsolás lehetősége, az egyszerű hibakeresés és tesztelés, valamint a nagy megbízhatóság.

A relék esetében a különböző funkciók megvalósítása a relék megfelelő összevezetékezésével történik. Amikor a relés logika által vezérelt rendszer működésén változtatni kell, akkor a relék közötti vezetékezést kell módosítani. Szélsőséges esetben ez a teljes vezérlőtábla cseréjét jelenti, hisz nem gazdaságos, vagy lehetetlen az átvezetékezés. A programozható logikai vezérlő helyettesíti a hagyományos relés logikák esetében alkalmazott vezetékezések nagy részét. A programozható logikai vezérlők számos előnnyel rendelkeznek a hagyományos relés logikákkal szemben. A modern irányítási rendszerek, például rendszerek közötti illesztésekhez továbbra is alkalmaznak reléket, de már csak ritkán a logikai funkciók megvalósításához.

PLC-k számos előnnyel rendelkeznek a hagyományos relés logikákkal szemben:

- Alacsonyabb költség;
- Kompaktabb kivitelezés;

- Egyszerű bővíthetőség;
- Nagyobb megbízhatóság;
- Nagyobb rugalmasság;
- Kommunikációs lehetőség;
- Gyorsabb válaszidő;
- Egyszerűbb hibakezelés.

Az irányítási rendszerek esetében a költségeket egy rendszer teljes életciklusára érdemes számítani. Mindent figyelembe véve elmondható, hogy a PLC-k felhasználása költséghatékonyabb megoldást eredményez, mint a relés vagy mikrovezérlős megoldások. A PLC-ket alapvetően arra tervezték, hogy az ipari alkalmazások terén, költséghatékony megoldásként kiváltsák a relés és a hagyományos mikrovezérlős logikákat.

Az elektronika fejlődésének köszönhetően egyre kompaktabb, kisebb méretű, de több funkciót magába foglaló kivitelezésben érhetők el napjaink PLC-i.

A modularitást alkalmazó megvalósítások és a hálózatba kötés lehetősége egyszerű bővíthetőséget biztosítanak a modern vezérléseknél.

A nagy megbízhatóság egyrészt abban nyilvánul meg, hogy csökken a mozgó érintkezők száma, az illesztéseknél alkalmazott elektronikus alkatrészek és áramkörök megbízhatóbb kapcsolatokat nyújtanak a rendszerek között, valamint a modern vezérlőkben alkalmazott programok, rendszer szintű megoldások biztosítják a folyamatos, hibátlan működést.

Nagyobb rugalmassággal is bírnak ezek a rendszerek hisz egyszerűbb létrehozni és megváltoztatni a PLC programját, mint bevezetékezni és újravezetékezni egy áramkört. A PLC esetében a kapcsolatot a bemenetek és a kimenetek között a felhasználói program határozza meg.

A modern ipari automatizálási megoldások megkövetelik a rendszerek, rendszerelemek közötti kommunikációt. Az ipari hálózatok infrastruktúrát nyújtanak a PLC-k, hálózatba kötésének így biztosítva a kapcsolatot a felsőbb hierarchia szinten működő ipari számítógépek, a HMI eszközök és a PLC-k között. A kommunikációs csatornák felhasználásával megvalósítható: a hierarchikus irányítás, a redundáns irányítás, a felügyeleti ellenőrzés, az adatgyűjtés, az eszközök és folyamatok paramétereinek monitorozása, valamint a programok fel és letöltése.

A relés logikákhoz viszonyítva a PLC-k gyorsabb válaszidővel rendelkeznek. A PLC-ket nagy sebességű és valós idejű alkalmazásokhoz tervezték. A programozható logikai vezérlők működése valós idejű, ami azt jelenti, hogy ha a terepen valamilyen esemény bekövetkezik, akkor ennek hatására végrehajtódik valamilyen művelet, vagy egy kimenet megváltozik. A PLC-k ciklusideje a legtöbb rendszer esetében azonnalinak tűnő beavatkozást eredményez.

A PLC-k állandó belső diagnosztikával rendelkeznek, és a felhasználók könnyen megkereshetik és kijavíthatják a különböző szoftver és hardver hibákat. A hiba megkereséséhez és kijavításhoz a felhasználó megjelenítheti az irányítási programot a monitoron és valós időben nyomon követheti annak működését.

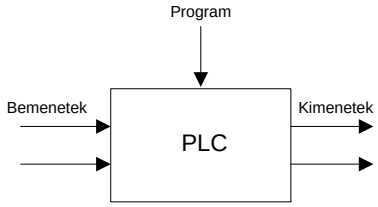
A PLC-k tervezése során az egyik fő szempont az volt, hogy ne csak a számítógépes programozó szakemberek legyenek képesek a programozásukra vagy a programjuk megváltoztatására. A létradiagram alapú programozhatóság nagyban hozzájárult a PLC-k széles körű elterjedéséhez.

A PLC-k több bemenetű és több kimenetű egységek, melyek tervezésénél figyelembe vették a szélsőséges, ipari körülmények közötti üzemeltetést. Így azok helytállnak a különböző hőmérsékleti és páratartalmi viszonyokban, nagyfokú tűrést mutatnak a mechanikai rezgésekkel és az elektromos zajokkal szemben. A PLC egy tipikus példája a valós idejű rendszereknek, hisz a PLC által irányított valós rendszer kimenete a bemeneti feltételektől függ. A programozható logikai vezérlő alapvetően egy speciális célú ipari számítógép, melyet különböző gépek irányítására terveztek. Mivel a PLC architektúrája ugyanazokon az elveken alapszik, mint az általános célú számítógépeké, így nem csupán az egyszerű relé szintű kapcsolási feladatok, hanem egyéb feladatok ellátására is alkalmas, mint például az időzítés, a számlálás, a számolás és az analóg jelek feldolgozása. Az általános célú számítógépekkel ellentétben a PLC-k úgy lettek

megtervezve, hogy képesek legyenek ipari környezetben működni, és speciális bemeneti/kimeneti interfésszel, valamint irányítási programozási nyelvvel rendelkezzenek.

2. A programozható vezérlők felosztása és architektúrája

A 2.1. ábrán látható, stilizált Programozható Logikai Vezérlő egy speciális mikrokontroller alapú automata, amely bemenetének és belső állapotának függvényében, a programja segítségével végzi el kimenete előállítását.



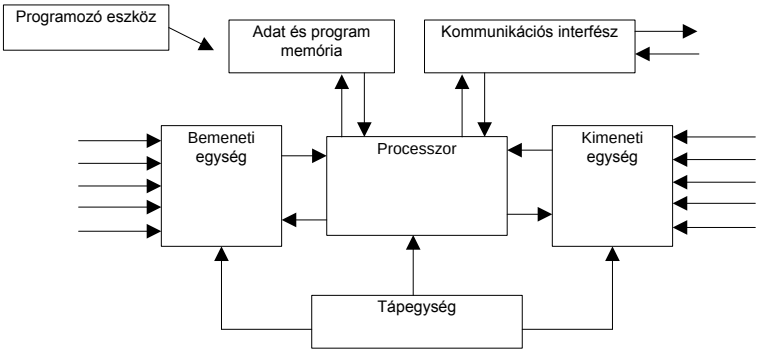
2.1. ábra. – A Programozható Logika Vezérlő stilizált ábrája.

A PLC-k és a számítógépek felépítése nagymértékben hasonló, azonban a számítógépek a különböző számítási feladatok elvégzésére, míg a PLC-k az ipari körülmények közötti irányítási feladatok ellátására lettek optimalizálva. Tehát a PLC-k:

- Úgy lettek megtervezve, hogy ellenálljanak a rezgéseknek, a különböző hőmérsékletű és páratartalmú külső környezetnek, és az elektromos zajoknak, zavaroknak.
- Rendelkeznek belső interfésszel a be- és kimeneti eszközök csatlakoztatásához.
- A könnyen megtanulható programozási nyelvük révén egyszerűen programozhatóak, hisz programozásukkor főleg logikai és kapcsolási műveletek implementálását kell végrehajtani.

Az első, huzalozott logikájú PLC-ket 1969-ben fejlesztették ki, és napjainkra igen széles körben elterjedtek az egyszerű kompakt felépítésű PLC-ktől egészen a moduláris rendszerekig, melyek nagyszámú be- és kimeneti vonalat, valamint analóg be- és kimeneti egységeket is tartalmaznak.

A PLC tipikusan egy olyan rendszer, melynek hat fő funkcionális része van, melyek a központi egység, a memória egység, a tápegység, a bemeneti/kimeneti egység, a kommunikációs interfész és a programozó eszköz, ahogyan ez a 2.2. ábrán is látható. A kompakt kivitelezésű esetben ezek az egységek egy tokozásban foglalnak helyet (2.3.a. ábra), míg a moduláris kivitel esetén külön modulok tartalmazzák az egyes funkcionális egységeket (2.3.b. ábra). A kompakt kivitelű PLC-k fix be- illetve kimenetek esetében alkalmazzák, melyek egy tokban tartalmaznak minden szükséges áramkört. A központi és az I/O terminál ugyanabban a tokban van, és az I/O terminálok fix számú be és kimeneti vonalat tartalmaznak. A fő előnye ennek a tokozásnak az alacsonyabb költség. A rendelkezésre álló I/O pontok száma változó és rendszerint van lehetőség a bővítésre újabb fix I/O egységek vásárlásával. Egy hátránya a fix I/O megoldásnak a flexibilitás hiánya, ugyanis a tokozás határozza meg az alkalmazható I/O-k típusát és számát. Egy másik hátránya ennek a megoldásnak, hogy ha ennek az egységnek egy része meghibásodik, akkor ez a teljes egység cseréjét igényli.



2.2. ábra. – A PLC felépítése.

A moduláris PLC konfiguráció rekeszekre bontható (2.3.b. ábra), ahova különálló modulok csatlakoztathatóak. Ez a tulajdonság növeli a rendelkezésre álló opciók számát és az egység flexibilitást. A gyártó által forgalmazott modulok közül választva tetszőleges kombinációk kialakíthatóak. Alapvetően egy moduláris vezérlő tipikusan egy rackbe illesztett tápegységet, központi modult, input/output modulokat és a programozáshoz és monitorozáshoz operátor interface-et tartalmaz. Amikor egy modult a rack-be csatlakoztatunk, akkor a rack hátoldalán lévő csatlakozópanelen keresztül kapja meg a tápellátást és éri el a különböző kommunikációs csatornákat is.

(a) *(b)*

2.3. ábra. – A PLC felépítése. Kompakt és moduláris.

A tápegység feladata a rendszer működéséhez szükséges energia biztosítása. A tápfeszültség általában szabványos értékű és frekvenciájú, a tápegység feladata illeszteni és kondicionálni azt a központi egység, a belső áramkörök és a bemeneti/kimenti modulok igénye szerint. A tápegység tipikusan a hálózati AC feszültséget átalakítja a rack-be csatlakoztatott modulok számára megfelelő DC tápfeszültséggé. Nagyméretű PLC rendszereknél a terepi eszközök tápellátását nem ez a tápegység, hanem valamilyen különálló AC vagy DC tápegység biztosítja. A kisméretű úgynevezett mikro PLC-k esetében azonban a terepi eszközök tápellátását is ez az egység biztosítja.

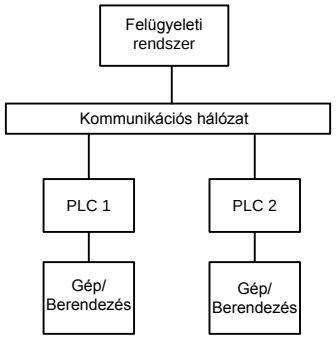
A központi logikai illetve feldolgozó egység vagy Central Processing Unit (CPU) tartalmazza a mikroprocesszort mely értelmezi a bementi jeleket és a memóriában tárolt programja, állapota, valamint a bemeneti jelek alapján a kimeneti jelek megfelelő beállításával végrehajtja az irányítási funkciókat.

A memória egység feladata egyrészt a CPU egység által végrehajtandó gépi program eltárolása, másrészt a belső állapotok, a bementekről érkező feldolgozandó és a kimenetekre küldendő feldolgozott adatok eltárolása. A memória mérete rendszerint K értékekben van

kifejezve, mint például 1K, 6K, 12K és így tovább. A memória elrendezésétől függően az 1K jelentése lehet 1024 bit, 1024 byte, vagy 1024 szó. Bár leggyakrabban a PLC memória kapacitását szavakban adjuk meg, mégis mielőtt a memória méretet pontosan össze tudnánk hasonlítani tudnunk kell, hogy egy szó hány bitből áll. A modern számítógépek esetében egy szó mérete 16, 32, 64 bit. Például ha egy PLC 8 bites szavakat használ, akkor egy 6K szó kapacitású memóriában 8x6x1024=49152 bit tárolható el, míg ha a PLC 32bites szavakat használ, akkor egy 6K szó kapacitású memóriában 32x6x1024=196608 bit tárolható el. A szükséges memória méretét is az adott alkalmazás határozza meg. A PLC-ben szükséges memória méretét egy adott alkalmazás esetében a használni kívánt I/O pontok száma, a vezérlési program mérete, adatgyűjtési követelmények, felügyeleti funkciók szükségessége és a jövőbeni bővítési igények határozzák meg.

A bemeneti/kimeneti egység valósítja meg az információáramlást a CPU és a csatolt bemeneti/kimeneti elemek között. Az I/O rendszer egy interfészt biztosít melyen keresztül a terepi eszközök a PLC-hez csatlakoztathatóak. Az interfész feladata a terepi eszközök felől érkező és a terepi eszközök felé küldött jelek kondicionálása. A bementi eszközök, mint például a nyomógombok, végállás kapcsolók és érzékelők fixen be vannak kötve a bemeneti terminálokba. A kimeneti eszközök, mint például a kis motorok, motorindítók, szelep-tekercsek, visszajelző lámpák, szintén fixen be vannak kötve a kimeneti terminálokba. Annak érdekében, hogy a be- és kimeneteket galvanikusan elválasszák a PLC belső elektronikájától optocsatolókat alkalmaznak. A be- és kimeneti eszközök csoportosíthatóak szabványos jeleik, jelszintjeik alapján. Ennek megfelelően három csoportot különböztethetünk meg, melyek a logikai, a digitális és az analóg eszközök. A logikai jeleket biztosító eszközök kimenetén 0 illetve 1 értékek jelenhetnek meg, hosszuk egy bit. A digitális eszközök alapvetően logikai eszközök, azonban a kimenetükön 0-ákból és 1-esekből álló jelsorozat, bitsorozat jelenik meg. Az analóg eszközök pedig olyan jeleket biztosítnak melyeknek értéke arányos az általuk monitorozott változó nagyságával. Például egy analóg hőmérsékletérzékelő olyan feszültség jelet biztosít, melynek nagysága arányos a mért hőmérsékletértékkel.

A kommunikációs interfész feladata, hogy valamilyen kommunikációs hálózaton keresztül küldjön és fogadjon adatokat más távoli PLC-ktől vagy egyéb intelligens eszközöktől. Ilyen kommunikációs műveletek lehetnek például az eszköz verifikáció, az adatgyűjtés, és a szinkronizálás két eszköz között. Egy kommunikációs blokkvázlat látható a 2.4. ábrán. A kommunikáció szabványos ipari kommunikációs szabványok szerint történik. A PLC gyártói igyekeznek támogatni minden fontosabb ipari kommunikációs szabványt.



2.4. ábra. – Alapvető kommunikációs modell.

A programozó eszköz segítségével kerül a megfelelő, gépi kódú program a PLC memóriájába. A kézi programozó eszközök egyszerűek és olcsóak. Segítségükkel könnyen felprogramozhatóak a PLC-k, valamint monitorozható a működésük. A PLC-k felprogramozásának és monitorozásának másik módja a személyi számítógépek használata. A személyi számítógépek a leggyakrabban használt programozási eszközök. A legtöbb PLC gyártó biztosít számítógépes programokat melyek segítségével elkészíthető, szerkeszthető,

monitorozható, letölthető a PLC programja. Egy PLC utasítás listája a PLC által támogatott utasításokat tartalmazza. Tipikusan a támogatott utasítások száma 15 utasítástól mely a kisebb PLC-kre jellemző, egészen a 100 utasításig terjedhet mely a nagyobb teljesítményű PLC-k sajátja. A számítógép monitorának nagyobb mérete miatt a PLC-n futó program könnyebben átlátható, monitorozható, mint a kézi eszközök esetében. A számítógépek a PLC központi egységével soros, párhuzamos vagy Ethernet kapcsolaton keresztül kommunikálnak. Ha a programozó eszköz nincsen használatban, akkor egyszerűen lecsatlakoztatható és eltávolítható.

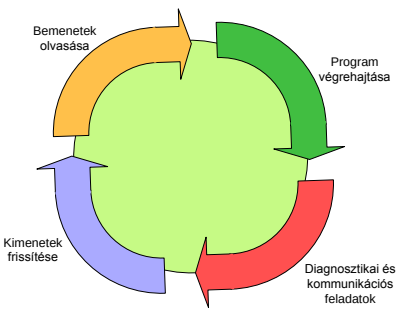
2.1. A központi egység belső architektúrája

A PLC esetében a CPU irányítja és dolgozza fel az összes műveletet. A modern PLC-k központi egysége mikroprocesszor vagy mikrokontroller alapú. Tipikus működési órajel frekvenciája 1Mhz-től terjedhet, melyet egy külső oszcillátor kristály állít elő. Ez az órajel határozza meg egyrészt a PLC működési sebességét, másrészt biztosítja az összes belső elem megfelelő időzítését és a szinkronizációt. A PLC-n belül az információ digitális jelek formájában került továbbításra belső buszokon keresztül. Ezeknek a buszoknak alapvetően négy típusát különböztetjük meg ezek az adat-, a cím-, a vezérlő- és a rendszerbusz. Az adatbusz feladata hogy adatokat továbbítson a részegységek között. A címbusz feladata az eltárolt adatok címének továbbítása. A vezérlőbusz feladata a különböző vezérlő információk továbbítása, míg a rendszerbusz a bemeneti/kimeneti port-ok és a bemeneti/kimeneti egység közötti információtovábbítást biztosítja.

A CPU belső felépítése az alkalmazott mikroprocesszor típusától függ. Általában a CPU tartalmaz:

- Aritmetikai és Logikai Egységet (ALU) mely az adat kezeléséért és az aritmetikai (például: összeadás, kivonás, szorzás) valamint a logikai (például ÉS, VAGY, NEM, Kizáró-VAGY) műveletek elvégzéséért felelős.
- Speciális és általános célú regisztereket, melyek a mikroprocesszor részét képezik és az irányítási program futtatása során keletkező illetve a futtatásához szükséges információkat tárolják. A speciális célú regiszterek közül kiemelhetők az utasítás regiszter, állapot regiszter, veremmutató.
- Vezérlő egységet, melynek feladata a műveletek időzítésének vezérlése.
- Sínrendszert, mely áll címsínből, adatsínből és vezérlősínből.

A központi egység a PLC agya. Egy tipikus PLC központi egység legalább egy mikroprocesszort tartalmaz, mely a különböző logikai és számítási funkciókat valósítja meg, valamint vezérli a kommunikációt az egyes modulok között. A központi egységnek a műveletek részeredményeinek eltárolására, szüksége van memóriára is. A PLC programja is egy EPROM, EEPROM vagy RAM memóriában kerül eltárolásra. A CPU vezérli a PLC által végrehajtott minden egyes műveletet. A PLC program egy ismétlődő folyamat részeként kerül végrehajtásra, ahogyan az a 2.5. ábrán látható.



2.5. ábra. – Egy tipikus PLC ciklus.

Egy tipikus PLC ciklus azzal kezdődik, hogy a CPU beolvassa a bemenetek státuszát. Ezután a PLC programja kerül végrehajtásra. A program végrehajtása után a PLC elvégezi a belső diagnosztikai és kommunikációs feladatokat. Végül a CPU frissíti a kimenetek státuszát. Ez a folyamat addig ismétlődik, amíg a PLC futási módban van.

A buszok biztosítják az útvonalakat a PLC-n belüli kommunikációhoz. Az információtovábbítás bináris formában az adott PLC bitszélességével megegyező nagyságú szavak formájában történik. Ahogyan az korábban is említésre került, alapvetően négy busztípust különböztetünk meg:

- Az adatbusz szállítja a CPU által végzett feldolgozáshoz szükséges adatokat. Egy 8-bites mikroprocesszor esetében a belső adat busz 8-bites, így 8-bites számok között képes műveleteket végezni, valamint az eredmény is 8-bites értékek formájában kerül meghatározásra.
- A címbusz feladata a különböző memóriahelyek címének továbbítása. Minden egyes memóriahelynek egyedi címe van, így a CPU ezt a címet felhasználva tud adatot kiolvasni a megfelelő memóriacímről, valamint ennek segítségével tud a megfelelő memóriacímre adatot írni. Ha a címbusz szélessége 8-bit, akkor ennek segítségével 2^8-on azaz 256 memóriahely címezhető meg, míg ha például 16-bites akkor már 65535 memóriahely címezhető a segítségével.
- A vezérlőbusz feladata, hogy a CPU által kiadott vezérlő információkat továbbítsa, mely segítségével informálhatja például a memóriát, hogy fogadja az adatokat a be- vagy kimenetei egységtől. Emellett a vezérlőbusz időzítési információkat is továbbít a rendszer egyes elemeinek szinkronizálásához.
- A rendszerbusz feladata a be- kimenti port-ok és a bemeneti/kimeneti egységek közötti kommunikáció lebonyolítása.

A PLC számos különböző memória típust tartalmaz melyek különböző funkcionalitással bírnak. Ilyen memóriák a:

- Rendszer memória mely egy Read-Only Memory (ROM), mely a PLC operációs rendszerét (firmware) valamint a működéséhez szükséges adatokat tárolja.
- Felhasználói programot tároló Random-Access-Memory (RAM).
- Adatokat tároló RAM. Ennek a memóriának az egyik részében a be- és a kimeneti eszközökhöz kapcsolódó státusz információk, valamint a portok értékei kerülnek eltárolására. Egy másik részében a számlálók és az időzítők értékei kerülnek eltárolásra, míg egy harmadik rész használható a felhasználói program futásához szükséges és a program által generált információk tárolására. Az adat RAM-ra gyakran úgy hivatkoznak, mint adat-tábla vagy regiszter-tábla.
- Erasable and Programmable Read-Only Memory (EPROM) mely olyan ROM, amely törölhető és újraprogramozható, így biztosítva a PLC firmware-jének frissíthetőségét.

Minden PLC tartalmaz valamekkora RAM-ot a felhasználó által létrehozott program és adatok tárolására. Az ebben a RAM-ban tárolt program és adatok a felhasználó által tetszőlegesen módosíthatóak. Annak érdekében, hogy a tápellátás kimaradásakor ne vesszen el a RAM-ban tárolt felhasználói program és az adatok a RAM tápellátását egy akkumulátoron keresztül biztosítják. Miután a program a RAM-ban kifejlesztésre kerül betölthető egy EPROM memória egységbe, ez által biztosítva a PLC program maradandóságát. Emellett vannak még ideiglenes pufferek, melyek a be- és a kimeneti portok értékeit tárolják.

2.2. PLC méret és alkalmazás

A PLC-k kategorizálásánál használt szempontok a funkcionalitás, a be- és kimenetek száma, a költség és a fizikai méret. Ezek közül a legfontosabb az I/O szám. Általában a nano a

legkisebb méret tipikusan 15 I/O pontnál kevesebb be- kimenettel rendelkezik. Ezt követik a mikor PLC-k (15-128 I/O pont), a közepes PLC-k (128-512 I/O), majd a nagy PLC-k (512 IO/ pont felett).

A PLC-k kiválasztásának egyik fő szempontja hogy a PLC minél jobban illeszkedjen az adott alkalmazáshoz. Általában nem tanácsos nagyobb PLC rendszert megvásárolni, mint azt az adott alkalmazás igényei diktálják. Fontos azonban figyelembe venni az adott alkalmazás jövőbeni igényeit is, hogy az adott PLC rendszer képes legyen ellátni az esetlegesen megnövekedett követelményeket is.

Alapvetően három fő alkalmazása lehetséges egy PLC-nek, melyek az úgynevezett egyfeladatos, többfeladatos, és a vezérlés menedzsment alkalmazások.

Az egyfeladatos alkalmazások esetében egy PLC vezérel egy folyamatot. Ebben az esetben a PLC egy külön álló egységként viselkedik, nincs összekapcsolva más PLC-kkel vagy számítógépekkel. A vezérelni kívánt folyamat mérete és bonyolultsága határozza, meg hogy milyen PLC-t válasszunk egy ilyen alkalmazásban.

Többfeladatos PLC alkalmazás esetében egy PLC több folyamat vezérléséért felelős. A megfelelő I/O pont szám a legfontosabb tényező a PLC kiválasztásakor egy ilyen alkalmazás esetében. Emellett, ha ez a PLC egy részegysége egy nagyobb rendszernek és kommunikálnia kell egy központi PLC-vel vagy számítógéppel, akkor biztosítani kell a megfelelő kapcsolatot az adat kommunikációs rendszerrel is.

A vezérlés menedzsment alkalmazások esetében a PLC feladata több más PLC vezérlése. Az ilyen alkalmazások esetében olyan PLC-kre van, szükség melyek elegendően nagy processzor kapacitással rendelkeznek, hogy kommunikáljanak más PLC-kkel és esetlegesen számítógépekkel. A vezérlés menedzsment feladatot ellátó PLC felügyeli a többi PLC-t a programjaik letöltésével és a feladataiknak meghatározásával. Az ilyen PLC-nek képesnek kell lennie kapcsolódni az összes többi PLC-hez, és a megfelelő címzést alkalmazva kommunikálni bármelyikkel, amelyikkel csak szükséges.

A single boksz, kompakt kivitelt tipikusan a kisméretű programozható vezérlők esetében alkalmazzák, és egy egység tartalmaz minden szükséges alkatrészt, azaz a tápegységet, a processzort, a memóriát, és a bemeneti/kimeneti egységet. Az ilyen PLC-knek tipikusan 6, 8, 12 vagy 24 bemenete és 4, 8, vagy 16 kimenete van, és a memóriájukban 300-1000 utasítás tárolható. Vannak olyan single box rendszerek melyek kapacitása bővíthető plusz bemenetekkel és kimenetekkel, kiegészítő bemeneti/kimeneti modulok csatlakoztatásával. A nagyobb bemenet/kimenet számmal rendelkező rendszerek rendszerint modulárisak, és rackbe helyezhetőek. A moduláris rendszerek külön egységként tartalmazzák a tápegységet, processzort, és a bemeneti/kimeneti egységeket. Az ilyen rendszerek könnyen bővíthetőek, az adott alkalmazáshoz szükséges egységekkel, mint például analóg bemeneti/kimeneti modul, szervomotor vezérlő, stb.

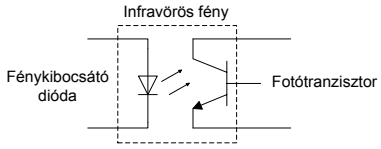
3. Bemenetek és kimenetek

A PLC architektúra bemeneti/kimeneti egysége biztosítja az interfészt a rendszer és a külvilág között. Minden egyes bemeneti/kimeneti pont egyedi címmel rendelkezik, melyet a CPU használ az egyes pontokhoz történő hozzáféréskor. A PLC programozásának fontos eleme a bemenetek és kimenetek pontos címeinek ismerete. A bemeneti/kimeneti egységek gyakran kezelhetők csoportokban, ilyenkor 8 vagy 16 bites szavak címzésére van lehetőség.

PLC-k esetében a legelterjedtebb I/O interface modul az 1 bit szóhosszúságú digitális I/O interface modul. Esetükben 1 vagy 0 logikai értékek értelmezhetők. Az ilyen típusú interfészek olyan terepi bemeneti eszközök csatlakoztatására képesek, melyek bekacsolt/kikapcsolt típusú jeleket biztosítanak, mint például a kapcsolók, nyomógombok vagy végállás kapcsolók. Hasonlóképpen olyan terepi kimeneti eszközök csatlakoztathatóak, mint például lámpák, jelfogók, tekercsek, motorindítók, melyek csak egyszerű bekacsolt/kikapcsolt kapcsolást igényelnek. A digitális I/O besorolás bit-elrendezésű be és kimenet. Ennél a típusú be és kimenetnél minden egyes bit önmagában egy teljes információs elem és megadja a státuszát valamely külső csatlakozásnak vagy a tápellátás jelenlétét a folyamat áramkörében.

Minden logikai valamint digitális I/O modulhoz valamilyen terepi feszültség forrás van csatlakoztatva. Mivel ezek a feszültségek nagyon különbözőek lehetnek amplitúdóban vagy típusban azért az I/O modulok különböző 24V –tól 230V-os, AC vagy DC névleges feszültség skálában érhetőek el.

A be/kimeneti csatornák rendelkeznek a megfelelő leválasztással jelkondicionálással, így az érzékelők és a beavatkozók közvetlenül hozzájuk kapcsolhatóak mindenféle járulékos illesztő elem felhasználása nélkül. A külvilág és a PLC közötti elektromos leválasztást leggyakrabban az úgynevezett optocsatolók segítségével valósítják meg. A 3.1. ábrán egy ilyen optocsatoló áramkör működési elvének bemutatása látható. Amikor egy digitális impulzus áthalad a LED diódán egy infravörös fényimpulzus generálódik. Ezt a fényimpulzust egy fotótranzisztor érzékeli, és egy feszültség impulzus generálódik a hozzá kapcsolt áramkörben. A távolság a LED és a fotótranzisztor között megfelelő galvanikus leválasztást biztosít.



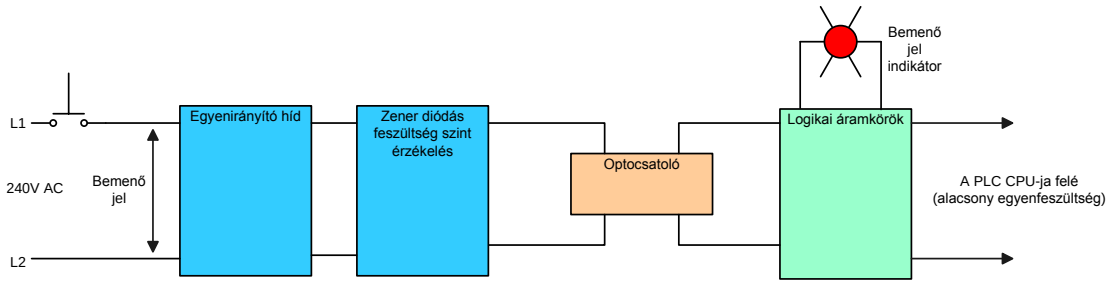
3.1. ábra. – Az optocsatoló működése.

3.1. Logikai bemeneti modul

A be/kimeneti egység bemeneti részén az optocsatolóban található fotótranzisztor által létrehozott digitális jel kompatibilis a PLC belső, például 5V-os DC feszültségével. Ezek a modulok alkalmasak bármilyen két állapotú bemeneti eszköz csatlakoztatására. A megfelelő jelkondicionálás és elektromos leválasztásnak köszönhetően a bemenő jelszintek széles skálája alkalmazható. Nagyobb PLC-k esetében a bemeneti jelek így lehetnek például 5V, 24V, 110V, vagy akár 240V logikai/digitális jelek. A kisebb PLC-k esetében tipikusan csak egy bemeneti jelszint például a 24V DC alkalmazható.

A 3.2. ábrán egy AC logikai bemeneti modul egy bemenetének blokk diagramja látható. A bemeneti áramkör alapvetően két szakaszból épül fel, melyek a táp és logika szakaszok. Egy optocsatoló biztosítja a megfelelő elektromos leválasztást a terepi eszközök áramköre és a PLC belső áramköre között. A bemeneti led be- és kikapcsolása jelzi a bemeneti eszköz aktuális

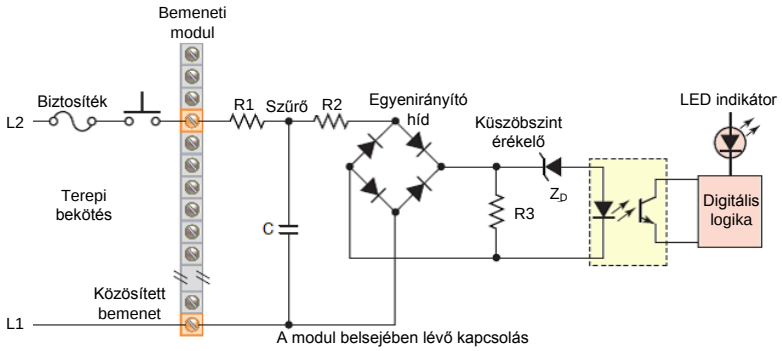
állapotát. A logikai áramkör dolgozza fel a digitális jelet a processzor számára. A belső PLC vezérlő áramkör tipikusan 5VDC feszültségszinten működik.



3.2. ábra. – Logikai AC bemeneti modul blokk diagram.

A 3.3. ábrán egy egyszerűsített diagramja látható a logikai AC bemeneti modul egy bemenetének. Az áramkör működése a következőképen foglalható össze:

- A bemeneti zajszűrő egy kondenzátorból és az R1 és R2 ellenállásokból áll. Feladatuk a bemeneten a kapcsolók által okozott lebegés és az elektromos interferencia kiszűrése;
- Amikor a nyomógomb zár 240VAC kerül az egyenirányító híd bemenetére;
- Ennek hatására az optocsatolóban található LED-en keresztül áram folyik, és elkezd világítani;
- A ZD Zener diódával állítható be a detektálható feszültség szint tresholdjának minimum értéke;
- A LED által kibocsájtott fény hatására a fotótranzisztor vezet, a fény galvanikus leválasztást biztosít a processzor felé;
- Az optocsatoló nem csupán a nagyobb bemenő AC feszültséget választja el a PLC belső logikájától, de megvédi azt a vezetékeken keletkező feszültség tranziensektől is. Emellett az optocsatoló segít csökkenteni az elektromos zajok hatását, melyek igen gyakran jelen vannak az ipari környezetben, és a logika hibás működését eredményezhetik;
- A hiba diagnosztikát az indikátor LED szolgálja, mely akkor van bekapcsolva, ha a bemeneti nyomógomb zárt állapotban van. Ez az indikátor hozzáköthető az optocsatoló mindkét bemenetéhez;
- Egy AC/DC típusú bemeneti modul használható mind az AC mind pedig a DC bemenetekhez, ilyenkor a bemeneti polaritás nem számít;
- A PLC bemeneti moduljában minden bemenet egymástól is el van választva, nincs közös bementi csatlakozás vagy olyan bemeneti csoport mely ugyanazon a közös csatlakozáson osztozik.



3.3. ábra. – A logikai AC bemeneti modul egy bemenetének vázlatos diagramja

A logikai bemeneti modul négy feladatot látnak el a PLC-s vezérlési rendszerben:

- Érzékeli, amikor jel érkezik a terepi eszköz felől.
- Átalakítja a bejövő jelet az adott PLC számára megfelelő feszültségszintű jellé
- Leválasztja a PLC-t bemeneti feszültségben vagy áramban megjelenő fluktuációktól.
- Jelet küld a processzornak jelezve, hogy melyik érzékelőtől érkezett a jelzés.

Több logikai bemenet, bit egységként való kezelése lehetővé teszi több bites információ együttes kezelését.

A bemenetek esetében fontos szempont a beállási sebesség, a gyártók gyakran megkülönböztetnek gyors be és kimeneti port-okat.

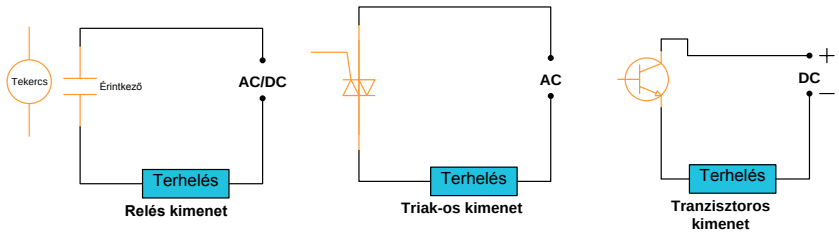
3.2. Logikai kimeneti modul

A be/kimeneti egység kimeneti része szintén 5V-os digitális jel. A logikai kimeneti modulok arra használhatóak, hogy a terepi eszközöket be illetve kikapcsolják. Ezek a modulok alkalmasak bármilyen két állapotú eszköz vezérlésére, mind AC mind pedig DC verzióban, valamint különböző feszültség és áram tartományokban kaphatóak. A megfelelő relés, tranzisztoros vagy triac-os jelkondicionálásnak köszönhetően a kimeneti csatornán megjelenő jel igen különböző lehet, például: egyenáramú 24V 100mA/110V 1A, illetve akár váltakozó áramú 240V 1A/2A. A nagyméretű PLC-k számos kimeneti jelszintet és formát támogathatnak, míg a kisebb PLC-k tipikusan csak egyet.

A kimenteknek alapvetően három típusát különbözethetjük meg, melyek a 3.4. ábrán látható: relés, tranzisztoros és triac-os kimenetek.

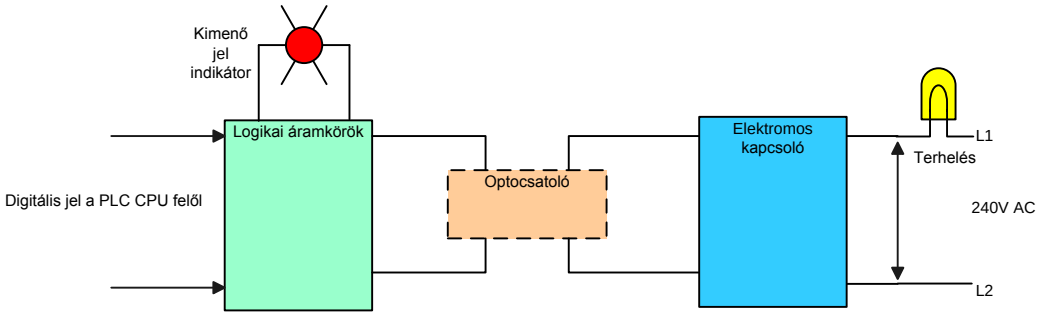
- A relés kimenet esetében relét használnak, a kimeneti jel kapcsolásához. Ezzel a megoldással a kimeneteken kapcsolható áramerősség akár néhány amper nagyságú is lehet. A relés kimet nemcsak nagy áramok kapcsolása miatt előnyös, hanem mert galvanikusan elválasztja a PLC-t a külső áramkörtől. A relés kimenetek azonban relatíve lassúak, kapcsolási sebességük kisebb mint 10Hz. Egyaránt alkalmasak AC és DC feszültség kapcsolására. Képesek ellenállni a nagy áramlökéseknek és feszültség tranzienseknek.
- A tranzisztoros kimenet esetében tranzisztort alkalmaznak a kimeneti jel kapcsolásához. A tranzisztoros kimenet segítségével lényegesen gyorsabb kapcsolás érhető el, mint a relés kimenet segítségével. Ez a kimenet típus csak DC feszültség kapcsolására alkalmas és könnyen tönkreteheti a nagy túláram és a nagy inverz feszültség. Ennek kivédésének érdekében vagy biztosítékot, vagy pedig valamilyen beépített túláram és túlfeszültség védelmet alkalmaznak. Optocsatolókat alkalmaznak az elektromos leválasztás biztosításához.

- A triac-os kimenet segítségével AC kimeneti jelek kapcsolhatóak. Itt is optocsatoló biztosítja az elektromos leválasztást. Csak AC jelek kapcsolhatóak, valamint ez a megoldás is érzékeny a nagy túláramokra, így itt is biztosítékokat alkalmaznak ennek kivédésére.



3.4. ábra. – Relés, tranzisztoros és triak-os kapcsolóelemek.

A 3.5. ábrán egy tipikus logikai kimeneti modul egy kimenetének blokk diagramja látható. A bemeneti modulhoz hasonlóan ez is két részből a táp és a logika részből épül fel, melyeket egy optocsatoló kapcsol egymáshoz. A kimeneti interfész úgy tekinthető, mint egy elektromos kapcsoló, mely be vagy kikapcsolja a kimeneti eszközt. A logikai áramkör határozza meg a kimenet állapotát. A kimeneti LED mutatja a kimeneti jel aktuális állapotát.



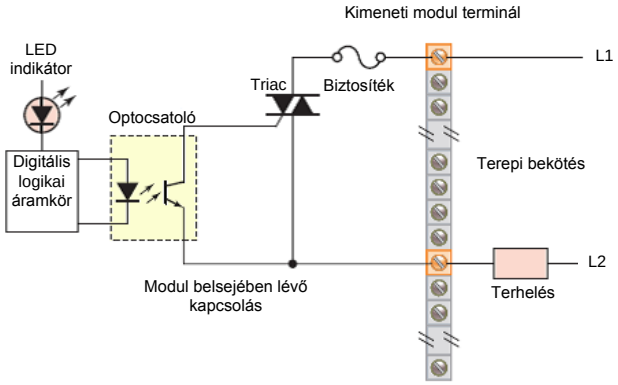
3.5. ábra. – Logikai AC kimeneti modul blokk diagramja

A logikai kimeneti modul egy kimenetének egyszerűsített diagramja látható a 3.6. ábrán. Az áramkör működése a következő:

- A helyes működés részeként a processzor digitális logikai áramköre a PLC-n futó programnak megfelelően beállítja a kimenet állapotát (logikai áramkör).
- Amikor a processzor egy kimeneti eszközt be akar kapcsolni akkor az optocsatolóban található LED-et bekapcsolja.
- A LED ennek hatására fényt bocsájt ki, mely hatására a fotótranzisztor elkezd vezetni. Ezen a ponton történik a terepi áramkörök galvanikus leválasztása a vezérlőtől.
- Ez trigger-eli a triak-ot, mely egy AC félvezető kapcsoló. Ennek eredményeként áram folyik a kimeneti terhelés felé.
- A triak-nak a bekapcsolt és kikapcsolt állapot helyet ALACSONY és MAGAS ellenállás szintjei vannak. A KI állapotában (MAGAS ellenállás) néhány milliampernyi áram továbbra is keresztülfolyik a triak-on.

A bemeneti interfészhez hasonlóan a kimeneti interfész is tartalmaz indikátor LED-et, mely a kimenet aktuális állapotának kijelzésére szolgál. Normál esetben biztosítékra is szükség van a kimeneti modulban, a modul rövidzár elleni védelmére. Bizonyos modulok a biztosíték állapotát

is visszajelzik. A triak nem használható DC terhelés kapcsolására. A hiba diagnosztikát szolgálja az indikátor LED, mely akkor van bekapcsolva, ha a PLC azt a parancsot adja, hogy a kimeneti terhelést be kell kapcsolni.

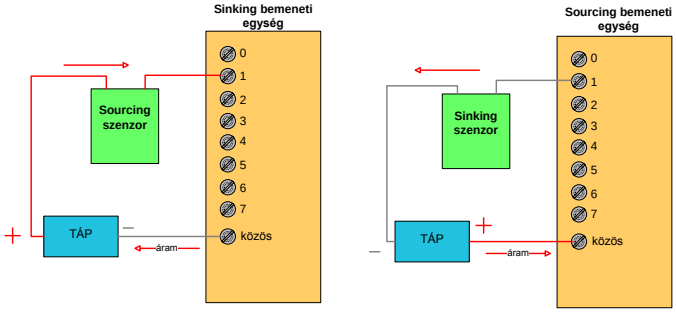


3.6. ábra. – A logikai kimeneti modul egy kimenetének egyszerűsített diagramja

Az egyes AC kimenetek által leadható áram a triak méretétől függően 1A vagy 2A lehet. A kimeneti modul védelme érdekében a specifikációban megadott áramértékeket soha sem szabad túllépni. A nagyobb terhelések vezérléséhez, mint például a nagy motorok, szabványos vezérlő relé kapcsolható a kimeneti modulhoz. A relé csatlakozói használhatóak nagyobb terhelések, vagy motorindítók vezérléséhez. Az ilyen megoldásoknál a relét közbülső relének hívják.

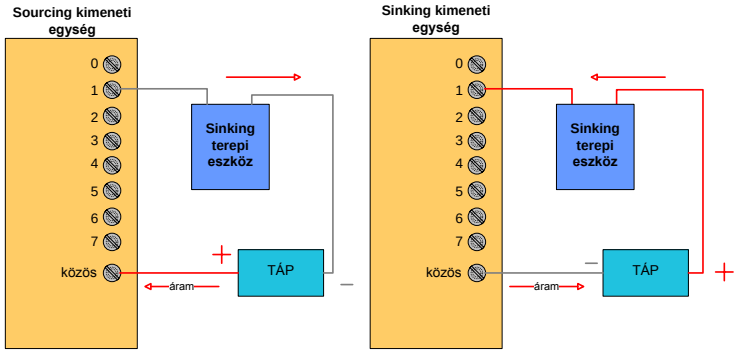
3.3. Sourcing és Sinking kapcsolások

A sourcing (forrás) és sinkink (nyelő) angol kifejezések arra utalnak, hogy a DC feszültségű eszközt hogyan kapcsoljuk össze a PLC-vel. Sourcing esetben az áram a konvencióknak megfelelően a pozitív potenciál felől folyik a negatív potenciál irányába, és a bemeneti eszköz áramellátását a bemeneti PLC modul biztosítja. Sinking esetben is a konvencióknak megfelelően az áram iránya a pozitív potenciáltól a negatív felé mutat, azonban itt a bemeneti eszköz biztosítja a bemeneti modul áramellátását 3.7. ábra.



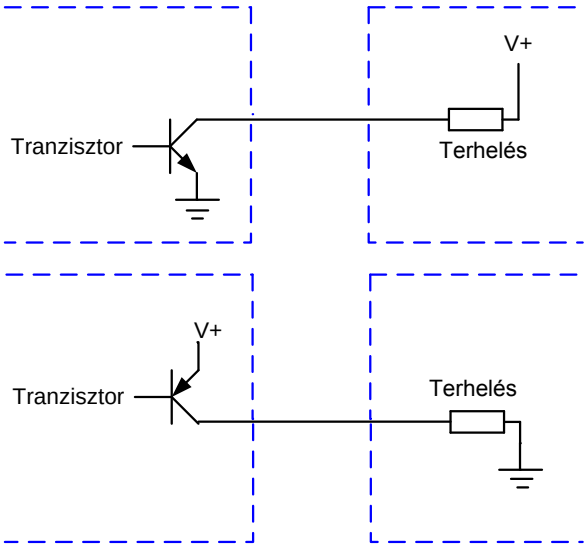
3.7. ábra. – Sinking és sourcing bemenetek.

Hasonló megfontolás alapján beszélhetünk a sourcing és a sinking típusú kimenetekről is (3.8. ábra).



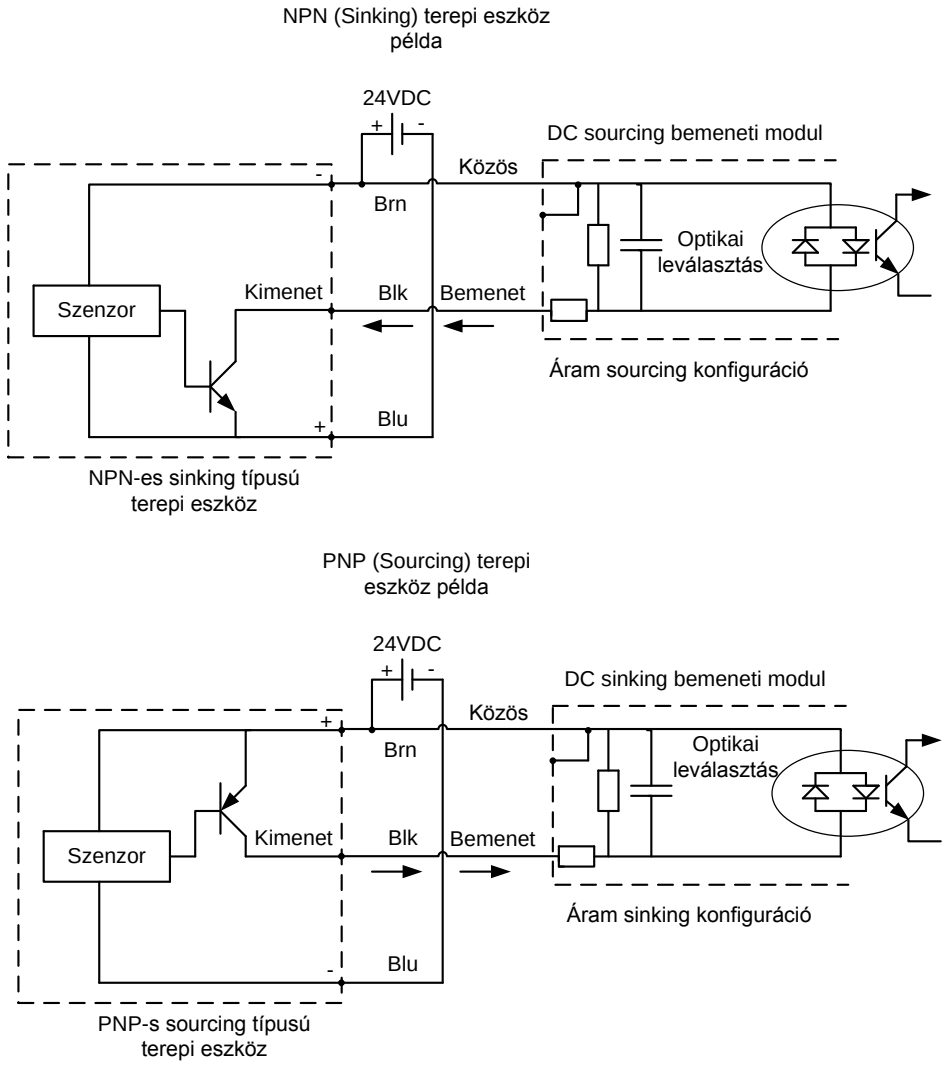
3.8. ábra. – Sinking és sourcing kimenetek.

A műszaki megoldás a nyitott kollektoros PNP vagy NPN típusú tranzisztorokkal kivitelezhető (3.9. ábra).



3.9. ábra. – Sourcing és sinking elvi megoldása.

Amennyiben a modul egy sourcing modul, akkor a be vagy kimeneti eszköznek sinking típusúnak kell lennie. Fordítva pedig, ha a modul sinking típusú, akkor a be vagy kimeneti eszköznek sourcing típusúnak kell lennie. Vannak olyan modulok is, amelyek esetében ki lehet választani, hogy sinking, vagy sourcing típusúként működjön, így bármilyen terepi eszköz csatlakoztatható hozzá.

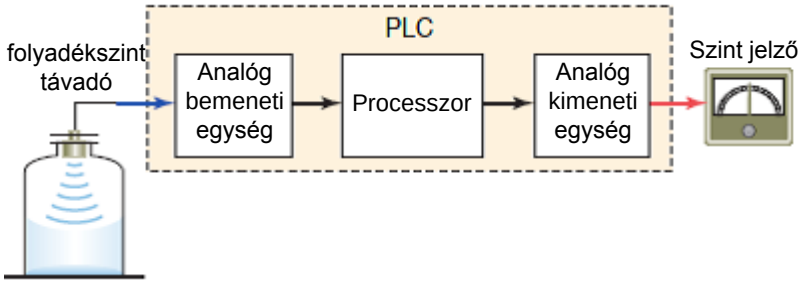


3.10. ábra. – Sourcing és sinking bemenetek és kimenetek.

3.4. Analóg I/O

A történelmi fejlődés folyamán a korai PLC-k csak logikai vagy digitális I/O interfészekkel rendelkeztek, így csak bekacsolt/kikapcsolt típusú eszközök kezelésére voltak alkalmasak. Ez a korlátozás azt jelentette, hogy a PLC csak részlegesen tudta vezérelni a különböző folyamatokat. Manapság azonban már rendelkezésre állnak mind digitális, mind pedig analóg interfészek, melyek segítségével a PLC-k gyakorlatilag minden irányítási folyamatban alkalmazhatóak.

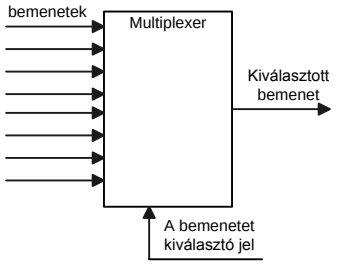
Az analóg eszközök fizikai mennyiségeket reprezentálnak, melyek végtelen számú értéket tartalmazhatnak. A tipikus analóg bemenet vagy kimenet lehet 0-20mA, 4-20mA vagy 0-10V szabványos értékű. A 3.11. ábrán az látható, hogy hogyan használható a PLC analóg be- és kimeneti modulja egy folyadék tartályban lévő folyadék szint mérésére, és kijelzésére. Az analóg bemeneti modul tartalmazza a folyadékszint távadó által küldött analóg feszültség vagy áram jelek fogasára alkalmas áramkört. Ez az analóg érték ebben a modulban átalakításra kerül egy digitális értékké, melyet már a PLC processzora is értelmezni tud. Az analóg kimeneti modulban található áramkör pedig digitális értékeket fogad a processzortól, és azt alakítja át analóg jellé mely vezérli a folyadékszint kijelző eszközt.



3.11. ábra. – Analóg be és kimenet a PLC-hez.

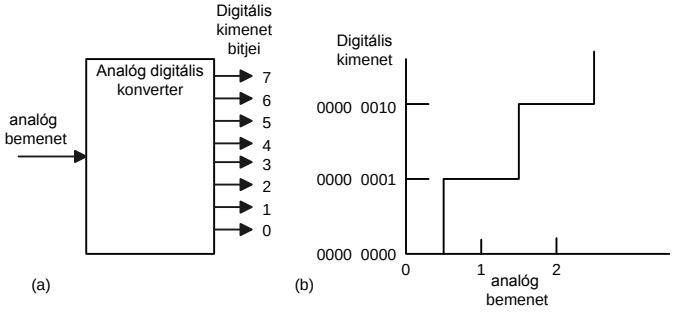
Az analóg bemeneti modulok tipikusan több bemeneti csatornával rendelkeznek, mely lehetőséget nyújt 4, 8, 16, eszköz PLC-hez csatlakoztatását. A két alapvető analóg bemeneti modul a feszültség és az áram bemenetű modulok. Az analóg érzékelők mérik a változó fizikai mennyiséget egy meghatározott tartományban és ennek megfelelően előállítanak egy megfelelő feszültség vagy áramjelet. A PLC analóg modulja által leggyakrabban mért fizikai mennyiségek a hőmérséklet, a sebesség, a szint, az áramlás, a súly, a nyomás és a pozíció. Például egy érzékelő mérhet hőmérsékletet a 0-500 fok tartományban és ennek megfelelően a kimenetén előállíthat egy feszültség jelet a 0-50mV tartományban.

Az analóg jeleket nem lehet közvetlenül a PLC bementére kötni, szükség van az analóg jelek digitálissá alakítására analóg-digitális átalakítók segítésével. Nem szükséges egy-egy analóg digitális átalakító minden egyes analóg jelhez, hisz a jelek multiplexálhatóak így egy átalakító több analóg jelet is képes kezelni (3.12 ábra). Az analóg bemeneti kártyák tipikusan 2, 4, 8, 16 analóg bemeneti csatorna kezelésére képesek.



3.12. ábra. – Analóg jelek multiplexelése.

A 3.13. ábra a részén egy analóg digitális átalakító működése látható. Egy analóg jelből 8 digitális vezetéken kiolvasható 8 bites digitális jel lesz. Egy ilyen 8-bites konverterrel 28 darab, azaz 256 különböző digitális értéket lehet előállítani az analóg jel reprezentálására (3.13.b. ábra).



3.13. ábra. – Analóg digitális átalakító működése

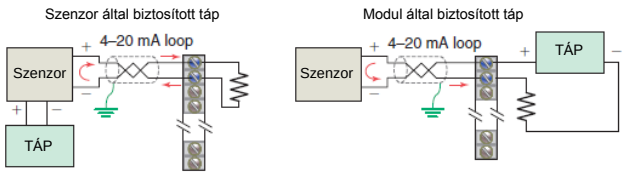
A kvantálási szint az egyes digitális értékek által reprezenált analóg feszültség értékek. A felbontás az a legkisebb változás az analóg jelben, amely egy bitnyi érték megváltozását jelenti a digitális jelben. Egy 8-bites ADC esetében, ha az analóg érték 0-10V között változik, akkor a digitális érték egy bitnyi megváltozásához 10/255V azaz 0.04V-nyi feszültségváltozás kell az analóg jelben. Ez azt jelenti, tehát hogy 0.03V-nyi feszültségváltozás az analóg jelben nem okoz változást a digitális jelben. Minél több bitet tartalmaz a digitális érték annál nagyobb a felbontása az ADC-nek azaz annál kisebb analóg feszültség változás is meg fog jelenni a digitális jelben. Ha például 10-bites ADC-t alkalmazunk, akkor 0-10V-os analóg jel esetében 10/1023V=0.01V analóg feszültségváltozás hatására is módosul a digitális jel egy bitje. Ha pedig egy 12-bites ADC-t alkalmazunk, akkor pedig már 10/4095=2.4mV-os analóg feszültségváltozás is egy bit módosulását fogja eredményezni az analóg jelben. Általános esetben egy n-bites ADC felbontása $1/(2^n-1)$. Az alábbi táblázatban látható egy 8-bitees analóg digitális konverter által a 0-10V-os analóg bejövő jelből előállított digitális értékek sorozata:

Analóg bemenet (V)	Digitalis kimenet
0,00	0000 0000
0,04	0000 0001
0,08	0000 0010
0,12	0000 0011
0,16	0000 0100
0,20	0000 0101
0,24	0000 0110
0,28	0000 0111
0,32	0000 1000

stb.

Hogy a fentieket illusztráljuk, képzeljünk el egy hőelemet, mely a PLC számára 0.5mV/C°-os kimeneti jelet szolgálta. Milyen pontosan tudja a PLC aktiválni a kimeneti eszközt, ha ezt a hőelemet egy 10-bites 0-10V-os ADC-n keresztül kapcsoljuk a PLC-hez? Egy 10-bites konverter esetében a 0-10V-os tartomány lefedésére 2^{10}=1024 darab digitális érték áll a rendelkezésünkre. Ebből következik, hogy 1 bit megváltozás 10/1023=0.01V azaz 10mV feszültségváltozás hatására jön létre. A pontosság, amivel a PLC a hőelem által szolgáltatott bemenetet felismeri +/-5mV vagyis +/-10fok.

Amikor feszültség jeleket vezetünk az analóg bemeneti modulhoz akkor a vezeték hossza csak korlátozott lehet, hiszen a túl hosszú vezetéken túl sok elektromos zaj gyűlhet össze, melyek jelentősen ronthatják az analóg modulhoz érkező feszültség jelek minőségét. Az áram jelek nem olyan érzékenyek a zajra, mint a feszültség jelek, így ezek továbbítására nincsenek távolsági korlátozások. Az áram bemenetű analóg input modulok általában 4-20mA vagy 0-20mA tartományban, lévő jeleket tudnak fogadni, de képesek a -20mA-20mA közötti áram jelek kezelésére is. Az áram jel továbbítására használt hurok tápellátását vagy az érzékelő vagy pedig az analóg modul szolgáltatja, ahogyan az a 3.14. ábrán is látható.

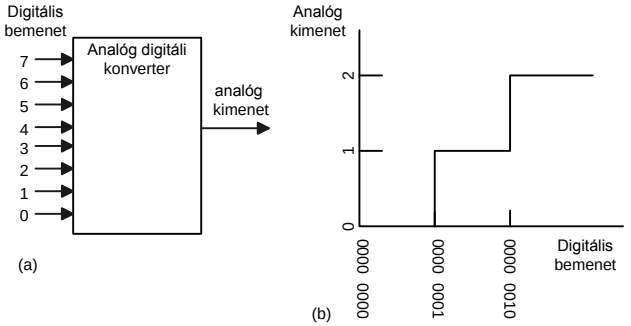


3.14. ábra. – : Érzékelő és modul által biztosított tápellátás.

Az analóg kimeneti interface a PLC processzorától érkező digitális jeleket fogadja, és azokat alakítja át ezzel arányos áram vagy feszültség értékké az analóg terepi eszközök vezérléséhez. A digitális jel analóg jellé történő átalakítását a digitális-analóg átalakítók (D/A, vagy DAC) végzik, melyek az analóg kimeneti modulok fő komponensei. A PLC által vezérelt analóg

kimeneti eszközök lehetnek például különböző műszerek, vezérlő szelepek, elektromos meghajtók és egyéb olyan eszközök, melyek analóg jeleket várnak.

A DAC bemenete egy digitális jel, melyet aztán ez az egység analóg jellé alakít. A 3.15. ábrán látható egy ilyen konverter alapvető felépítése és működése.



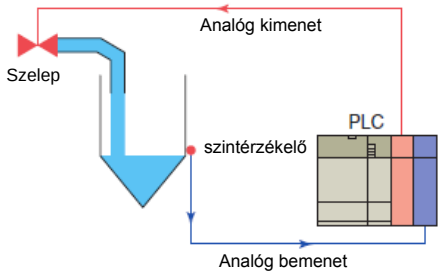
3.15. ábra. – : DAC vagy digitális analóg átalakító.

A 0-s pozícióban lévő bit hatására a kimeneten megjelenik egy bizonyos nagyságú impulzus. Az 1-es pozícióban lévő bit hatására a kimeneten az előbbinél kétszer nagyobb impulzus jelenik meg. A 2-es bit megváltozása hatására 4-szer akkor kimenet jön létre, mint az 1-es bit megváltozásakor és így tovább. Az összes bitpozícióhoz tartozó kimeneti impulzus összegéből kapjuk az analóg kimeneti jelet. Amikor a digitális érték változik, akkor a megváltozott bithez tartozó mennyiségnek megfelelően lépésenként változik az analóg kimenet értéke. Például ha egy 8-bites konvertert alkalmazunk, akkor a kimenet 2^8=256 analóg feszültségértékbeli lépést fog tartalmazni. Feltételezve hogy a kimeneti tartomány 0-10VDC, egy bit megváltozása 10/255V=0.04V változást jelent az analóg értékben.

Digitalis bemenet	Analóg kimenet (V)
0000 0000	0,00
0000 0001	0,04
0000 0010	0,08 + 0,00 = 0,08
0000 0011	0,08 + 0,04 = 0,12
0000 0100	0,16
0000 0101	0,16 + 0,00 + 0,04 = 0,20
0000 0110	0,16 + 0,08 = 0,24
0000 0111	0,16 + 0,08 + 0,04 = 0,28
0000 1000	0,32

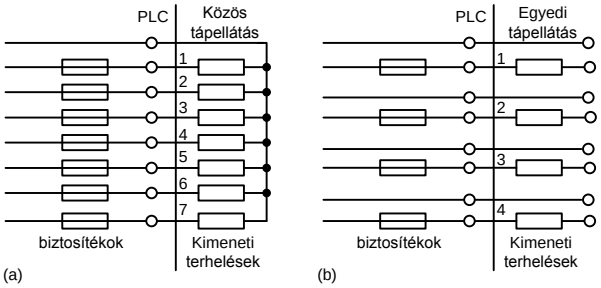
stb.

A 3.16. ábrán egy analóg I/O modul tipikus használata látható egy PLC-s irányítási rendszerben. Ebben az alkalmazásban a PLC egy szelep nyitottsági százalékának állításával vezérli, hogy mennyi folyadék kerüljön egy tároló tartályba. A PLC analóg kimenete szabályozza a szelepet. Kezdetben a szelep 100%-osan nyitva van. Amikor a folyadék szint eléri a beállított alapértéket a PLC processzora módosítja az analóg kimeneten megjelenő értéket, mely ennek megfelelően állítja a szelepet, hogy a folyadék szintet a kívánt értéken tartsuk. A szabályozási feladatokat ellátó PLC-k integer, fixpontos vagy lebegőpontos aritmetikai műveletek elvégzésére is alkalmasak.



3.16. ábra. – : Tipikus analóg I/O rendszer.

Az analóg kimeneti modulok rendszerint több különböző típusú analóg kimeneti jelet is képesek létrehozni, például 4-20mA, 0-5VDC, vagy 0-10VDC, és a megfelelő kimenet típus a modulon található kapcsolóval, vagy a modulhoz tartozó konfigurációs programmal adható meg. A moduloknak alapvetően két típusú lehet a kimenetük. Ez egyik típus esetén a modul minden kimenetének közös tápellátása van, a másik típus esetén pedig a modul minden kimenetének külön tápellátása van. A 3.17. ábrán látható ez a két kimeneti modul típus.



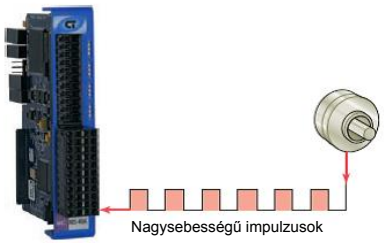
3.17. ábra. – : Közös és egyedi tápellátás.

3.5. Speciális I/O modulok

Az ipari alkalmazások terén számos tipikus műszaki megoldás ismétlődik. A PLC gyártók ezen megoldásokhoz specializált bemeneteket és kimeneteket fejlesztettek ki. Ezek a modulok az alábbiakban kerülnek bemutatásra.

3.5.1. Nagy sebességű számláló modul

A nagy sebességű számláló modult olyan alkalmazásokban használják, melyek olyan számlálási sebességet igényelnek, melyek felülmúlják a PLC létradiagram futtatási képességét. A nagysebességű számláló modulok segítségével impulzusokat számolhatunk (3.18. ábra), melyek különböző nagysebességű érzékelők, enkóderek, kapcsolók felől érkeznek. Ezek a modulok tartalmazzák a szükséges elektronikát, ahhoz, hogy a processzortól függetlenül képesek legyenek az impulzusok számlálására. A tipikus számlálási sebesség 0-100kHz között mozog, melye azt jelenti, hogy akár 100000 impulzus is megszámolható másodpercenként.



3.18. ábra. – : Nagy sebességű számláló modul.

3.5.2. Görgetőkerék modul

A görgető kerék modul arra használható, hogy numerikus információt szolgáltassunk a PLC-nek, mely befolyásolja a PLC által futtatott vezérlési program működését (3.19. ábra).

3.19. ábra. – : Görgetőkerék modul.

3.5.3. TTL modul

A TTL modul a TTL jelszintek küldésére és fogadására biztosít lehetőséget. Ennek a modulnak a segítségével, olyan eszközök is a PLC-hez kapcsolhatóak, melyek TTL szintű jeleket használnak a kommunikációhoz.

3.5.4. Enkóder-számláló modul

Az enkóder számláló modul arra használható, hogy egy enkóder által szolgáltatott jelet olvasson valós időben, és ezt eltárolja. A PLC processzora ezt az értéket tetszőleges időközönként olvashatja ki.

3.5.5. BASIC vagy ASCII modul

A BASIC vagy ASCII modul képes a felhasználó által elkészítetett BASIC vagy C nyelvű programot futtatni. Ezek a programok függetlenek a PLC processzorától, segítségével könnyen és gyorsan lehet interface-et létrehozni a PLC és egy idegen eszköz között. Tipikus alkalmazási a vonalkódolvasók, robotok, nyomtatók, megjelenítők.

3.5.6. Léptető motor modul

A léptető motor modul impulzus sorozatokat képes generálni, mely segítségével a hozzá kapcsolt léptető motorok vezérelhetőek.

3.5.7. BCD kimeneti modul

A BCD kimeneti modul segítségével lehetőség van arra, hogy a PLC olyan eszközöket működtessen, mely BCD kódolású jeleket fogad, mit például a 7-szegmenses kijelzők.

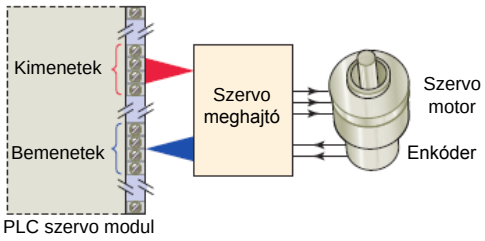
3.5.8. PID modul

A PID modul olyan folyamatirányítási feladatokban alkalmazható, melyek irányításához PID algoritmus szükséges. Az algoritmus lényegében egy komplex program alapján történő matematikai számítás. A PID modul segítségével ezek a komplex számítások elvégezhetőek, ezzel is tehermenetesítve a PLC processzorát. A fő alkalmazási területe olyan irányítási műveletek

elvégzése mely segítségével a hőmérséklet, az áramlás, a szint vagy a sebesség egy előre meghatározott értéken tarthatóak. Megjegyzendő, hogy több olyan PLC gyártó is van melyek által gyártott PLC-kben megtalálható a PID szabályzási funkció, és nem igényelnek ezen a feladatok ellátásához különálló modulokat.

3.5.9. Mozgás és pozíció vezérlő modul

A mozgás és pozíció vezérlő modulokat nagy sebességű és nagy pontosságú megmunkálási és csomagolási feladatok ellátására alkalmazzák (3.20. ábra). A pozícionálás, útmérés és fordulatszám, sebesség szerinti szabályozás egy modul esetében általában egy tengely mentén történik. Az intelligens pozíció és mozgás vezérlők menetesítik a PLC-t attól, hogy léptető vagy szervomotorokat vezéreljenek. Az ilyen rendszerek tartalmazzák a motorok meghajtásához szükséges teljesítmény elektronikát és átalakítják a PLC felől érkező jeleket a motor számára értelmezhető jelekké.



3.20. ábra. – : PLC szervo modul.

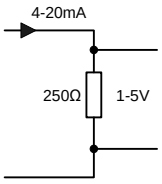
3.5.10. Kommunikációs modulok

A PLC-k gyakran rendelkeznek valamilyen szabványos soros kommunikációs vagy ipari hálózati kommunikációs lehetőséggel. A soros kommunikációs modulok segítségével lehetőség van más intelligens eszközökkel pont-pont közötti kapcsolatok létrehozására, és ezen a kapcsolaton keresztül adatok cseréjére. Ilyen kapcsolatokat rendszerint valamilyen számítógéppel, operátori állomással, folyamatirányítási rendszerekkel, vagy másik PLC-vel lehet létrehozni. A kommunikációs modul révén a PLC-k nagysebességű helyi hálózathoz kapcsolódhatnak.

3.6. Jelkondicionálás

Amikor digitális jelet szolgáltató érzékelőket kapcsolunk a bementi egységhez, fontos meggyőződni arról, hogy a feszültség szintek megfelelőek. Számos olyan érzékelő is van, amelyek analóg jeleket generálnak. Annak érdekében, hogy ne kelljen külön bemeneti csatornákat létrehozni minden egyes analóg jeltípushoz, melyet az analóg bemeneti modul kezelni képes, külső jelkondicionáló áramkörre van szükség, hogy ezeket a különböző analóg jeleket szabványos típusú analóg jellé alakítsuk, és így mindegyikhez ugyanazt a szabványos bemeneti csatornát használhassuk.

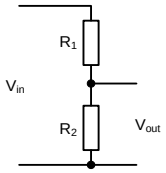
Egy gyakran alkalmazott megoldás látható a 3.18. ábrán, mely segítségével a 4-20mA-es analóg jelet 1-5V-os feszültségjellé alakítjuk, egy 250Ohm-os ellenállás alkalmazásával. Például, ha van egy olyan folyadékszint érzékelő mely 4-20mA-es jelet generál, ahol a tartályban lévő folyadék 0 szintjéhez a 4mA míg az 1m szintjéhez a 20mA tartozik, akkor ezt a jelet ennek a kapcsolásnak megfelelően át tudjuk alakítani 1-5V-os feszültségjellé. Az analóg áramjel esetében a 4mA-es áramértéket azért rendeljük a 0 szinthez, hogy meg tudjuk különböztetni azt, amikor a mért érték a 0 szinten van és azt, ha az érzékelő nem működik, vagy az érzékelő és a bementi egység közötti vezeték megszakadt. A 4mA sok analóg érzékelő működéséhez elegendő így azok számára nem szükséges külön tápegység.



3.18. ábra. – : Szabványos analóg jelátalakítás.

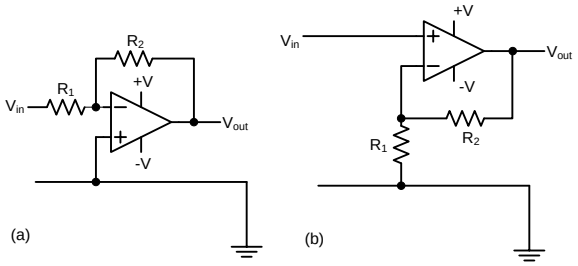
A feszültségosztó áramkör arra használható, hogy egy érzékelő által generált analóg feszültségjel szintjét a megfelelő tartományra csökkentse (3.19. ábra). Az áramkör kimenetén megjelenő feszültség a következőképpen alakul:

$$V_{out} = \frac{R_2}{R_1+R_2} V_{in}$$



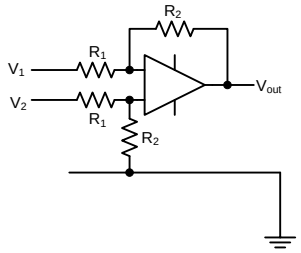
3.19. ábra. – : Feszültségosztó.

Az erősítős áramkörök felhasználhatóak arra, hogy az érzékelők által szolgáltatott analóg feszültségjelet a megfelelő tartományra növeljük. A 3.20. ábrán egy ilyen feladatot ellátó műveleti erősítős alapkapcsolás látható, invertáló és nem invertáló kapcsolásban. Invertáló kapcsolásban a kimenet: $V_{out} = -\frac{R_2}{R_1} V_{in}$. Nem invertáló kapcsolásban a kimenet: $V_{out} = \frac{R_1+R_2}{R_2} V_{in}$



3.20. ábra. – : Műveleti erősítős kapcsolások.

Gyakran differenciál erősítőt kell alkalmazni, hogy két bejövő feszültségjel között különbséget felerősíthessük. Ilyen eset lehet például a nyúlásmérő bélyeg Wheatstone-hídas alkalmazása, melyben a kimenet két feszültség különbsége. Emellett a differenciális jelek esetében is ilyen típusú erősítőt alkalmaznak a bemeneti egységben, hisz ez a kapcsolás csak a két feszültség közötti különbséget erősíti fel, a két vezetékre azonosan rászuperponálódott zavar jelet pedig elnyomja. Egy ilyen differenciális erősítő kapcsolás látható a 3.21. ábrán, mely esetben a kimeneti feszültség: $V_{out} = \frac{R_2}{R_1}(V_2 - V_1)$



3.21. ábra. – : Differenciális jelerősítő.

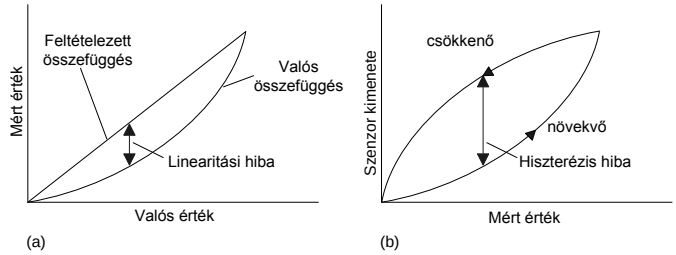
Az irodalomban megtalálhatók további jelkondicionálásra alkalmas kapcsolások. Alkalmazásukkor a jelhordozók átalakítása, a jelszintek illesztése mellett figyelembe kell venni a szélsőséges, ipari körülmények közötti működést, a zajos környezetet is.

3.7. Bemeneti eszközök

Az érzékelő (sensor) elnevezés egy olyan bemeneti eszközre utal, amely egy fizikai hatásra valamilyen kimenetet generál. Például a termoelem egy érzékelő, amely a hőmérséklet különbséget, elektromos kimenetté alakítja. A jelátalakító (transducer) egy olyan eszköz, amely egy fizikai jelet képes más formájú fizikai jellé átalakítani. Ebből következően az érzékelők gyakran jelátalakítók is, de vannak olyan eszközök is, amelyek csak jelátalakítók, mint például egy motor mely az elektromos bemenetet forgássá alakítja.

Az érzékelők, amelyek digitális/logikai azaz bekacsolt/kikapcsolt típusú kimenetet adnak egyszerűen hozzákapcsolhatóak a PLC bemeneteihez. Az olyan érzékelők jelét, amelyek analóg jeleket adnak, át kell alakítani digitális jellé mielőtt a PLC bementére kötjük. Az alábbiakban megnézünk néhány gyakran előforduló kifejezést, amely az érzékelők teljesítményének definiálására használnak.

Pontosság azt határozza meg, hogy a mérő rendszer vagy annak eleme által szolgáltatott mérési információ mennyire áll közel a valósághoz. Például egy hőmérsékletérzékelő pontossága lehet például +/- 0.1 fok. A mérés hibája a mért érték és a valós érték közötti eltérést jelenti. Mérési hiba számos forrásból származhat. Egyik ok lehet, a linearitási hiba mely abból származik, hogy az érzékelő mérési tartományában a bemenete és kimenete között nem lineáris a kapcsolat (3.22. ábra a). Másik ok lehet a hiszterézis hiba mely abból adódik, hogy az érzékelő kimenetén egy bizonyos ideig ugyanaz az érték jelenik meg akkor is, ha a mért érték csökken vagy növekszik (3.22. ábra b).

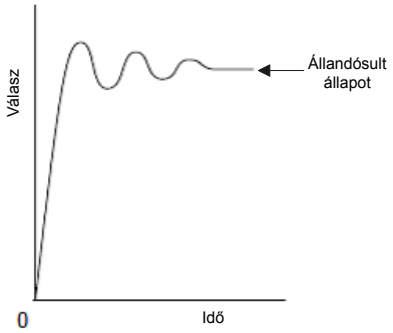


3.22. ábra. – A nemlinearitásból adódó hiba illusztrációja.

A mérési tartomány egy fontos jellemző. Egy rendszerben a változó értékek tartománya azt határozza meg, hogy a bemenetek milyen határok között változhatnak. Például egy hőmérsékletérzékelő ellenállás által érzékelhető tartomány -200 és +800 fok közé eshet.

A beállási idő. Ha egy érzékelő esetén a bemenet változik, akkor kell valamennyi idő, hogy az érzékelő kimenetén megjelenő jel stabilizálódjon (3.23. ábra). A válaszidő az az idő, amely

eltelik azalatt miközben a rendszer vagy egy elem bemenetén megjelenő érték hatására a kimeneten is megjelenik az ehhez az értékhez tartozó állandósult állapotbeli érték legalább 95%-a. A felfutási idő az az idő, amely eltelik, amíg az érzékelő kimenete eléri az állandósult állapotbeli érték valamely százalékát. Felfutási időként rendszerint azt az időt adják meg, amíg az állandósult állapot 10%-áról elér a kimenet a 90%-áig. A beállási idő az az idő, amely idő alatt a kimenet eléri az állandósult állapot körüli érték 2%-át



3.23. ábra. – A nemlinearitásból adódó hiba illusztrációja.

Az érzékenység megmutatja, hogy az érzékelő rendszer vagy annak egy elemének kimenete milyen mértékben változik, amennyiben a bemeneti mért érték egységnyit változik. Például egy hőelem érzékenysége 20uV/C°, ami azt jelenti, hogyha a mért hőmérséklet 1 fokot változik, akkor annak hatására a termoelem kimenete 20uV-ot változik.

A stabilitás egy mérőrendszer vagy eleme esetében azt jelenti, hogy a kimenet mennyire állandó, ha hosszabb időn keresztül ugyanazt a konstans értéket mérjük. A drift kifejezés azt jelenti, hogy a kimenet időben mennyit változik.

A megismételhetőség azt jelenti, hogy a mérőrendszer vagy annak egy eleme mennyire adja ugyanazt a kimenetet ugyanazon érték többszöri ismételt mérése esetében. A megismételhetőség hiányát gyakran a környezeti változások okozzák, úgy mint a hőmérséklet, vagy páratartalom változás. A megismételhetőségből adódó hibát a teljes kimeneti tartomány százalékában fejezik ki. Például egy nyomásérzékelő esetében a megismételhetőség a teljes tartomány +/-0.1%-a, ami azt jelenti hogy ha a teljes tartomány 20kPa akkor a megismételhetőségi hiba +/- 20Pa.

A megbízhatóság egy mérő rendszer vagy annak egy eleme esetében egy valószínűséget jelöl, hogy az eszköz meddig, milyen körülmények mellett fog egy meghatározott teljesítménnyel működni. A meghatározott teljesítmény általában a rendszer által adott pontosságot jelenti.

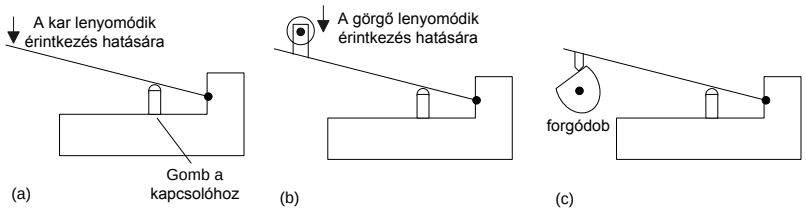
Az alábbiakban, a teljesség igénye nélkül bemutatásra kerül néhány tipikus bemeneti és kimeneti eszköz.

3.7.1. Mechanikus kapcsolók

A mechanikus kapcsolók kétállapotú bemeneti eszközök, melyek esetében megkülönböztetünk nyitott és zárt állapotot, működésük szerint valamilyen mechanikai erőhatás következtében érintkezőket nyitnak vagy zárnak. Ilyen mechanikai kapcsolók használhatóak egy munkadarab jelenlétének detektálására, azáltal hogy a munkadarab megnyomva a kapcsolót zárja annak érzékelőit. A munkadarab jelenlétét tehát a kapcsoló zárt állapota, a hiányát pedig a kapcsoló nyitott állapota jelzi.

A mechanikai kapcsolókat két típusban lehet megvásárolni a normál esetben nyitott (NO – Normally Open)és a normál esetben zárt (NC – Normally Closed) típusban. Az NO típusú kapcsolók mechanikai hatás hiányában nyitott állapotban vannak, és mechanikai behatásra kerülnek zárt állapotba. Az NC típusú kapcsolók pedig pont fordítva működnek, azaz mechanikai hatás hiányában zárt állapotban vannak, míg mechanikai behatásra nyitott állapotba kerülnek.

A mechanikus kapcsolók egy speciális fajtája a végállás kapcsolók. A végállás kapcsoló kifejezés olyan mechanikai kapcsolóra utal, mely segítségével érzékelni lehet egy mozgó rész elhaladását vagy jelenlétét. A végállás kapcsolókat rendszerint valamilyen görgős vagy görgő nélküli rúddal esetleg egy forgó dob segítségével lehet aktiválni és deaktiválni, ahogy az a 3.24. ábrán is látható.



3.24. ábra. – : Végállás kapcsolók.

3.7.2. Közelítéskapcsolók

A közelítéskapcsolók olyan eszközök melyek segítségével fizikai kontaktus nélkül lehet, egy objektum jelenlétét detektálni. Az ilyen kapcsolóknak számos formája van, vannak olyanok például, amelyek csak fémes objektumok detektálására alkalmasak.

Az örvényáram típusú közelítés kapcsolók egy tekercset tartalmaznak, melyen váltakozó áramot vezetnek keresztül, ezzel létrehozva egy váltakozó mágneses mezőt. Amikor egy fémes objektum közel kerül ehhez a kapcsolóhoz a váltakozó mágneses mező örvényáram indukálódik a fémes eszközben. A mágneses mező ezeknek az örvényáramoknak köszönhetően egy EMF-et (Electromotive Force), elektromotoros erőt generál mely visszahat a tekercsre és hatására változik a tekercs feszültsége. A tekercs feszültségének változása arányos a fémes tárgy érzékelőtől mért távolságával. A változó feszültség általában egy tranzisztort vezérel, mely a változó feszültség hatására ki vagy bekapcsolt állapotba kerül. Ezen eszközök tipikus hatótávolsága 0.5mm-20mm között van.

Egy másik közelítéskapcsoló típus a reed kapcsoló, vagy reed relé. A reed relé felépítése és működése egyszerű. Egy zárt, vákuumos vagy védőgázzal (nitrogéngáz) töltött üveg vagy műanyag csőben egy fémlapka pár található. Ez a lapka pár egy mágnesezhető mozgó érintkezőből és egy nem mágnesezhető, álló érintkezőből tevődik össze. Ha ezt az érzékelőt mágneses térbe helyezzük, akkor a mágnesezhető lapka a mágnese tér hatására elhajlik, és hozzáér a nem mágnesezhető lapkához, ezzel zárva az áramkört, és jelezve egy mágneses tárgy jelenlétét. Ezeket a kapcsolókat tipikusan ajtónyitás érzékelésre használják, például riasztó rendszerekben.

A közelítés kapcsolók azon típusa, amely képes fémes és nem fémes objektumokat is detektálni azok a kapacitív közelítés kapcsolók. A kondenzátor két egymástól bizonyos távolságra elhelyezett vezető lapkából (fegyverzet) épül fel. Minél kisebb a távolság annál nagyobb a kondenzátor kapacitása. A kapacitív közelítés kapcsoló lényegében egy fél kondenzátor, azaz a kondenzátor fegyverzetének csak egyik felét tartalmazza. A kondenzátor fegyverzetének másik fele a fémes tárgy melynek jelenlétét detektálni szeretnénk. Ebből következően az objektum jelenlétét a kapacitás változás alapján detektálhatjuk. A kapacitív közelítés kapcsoló alkalmas nem fémes tárgyak detektálására is, hisz a kapacitás a kondenzátor fegyverzetei között lévő dielektrikumtól is függ. Ebben az esetben a kondenzátor egyik fegyverzete az érzékelő a másik fegyverzete a föld a dielektrikum pedig a nem fémes detektálandó tárgy. A kapacitás változását fel lehet használni arra, hogy aktiváljunk egy elektronikus kapcsoló áramkört és azt ki és bekapcsoljuk. A kapacitív közelítés kapcsolók tipikusan 4-60mm-re lévő tárgyak detektálására alkalmasak.

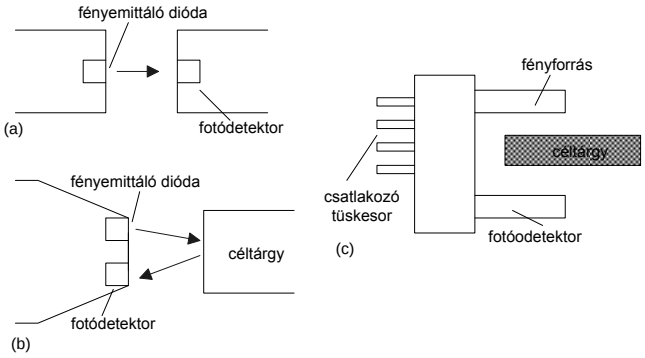
Egy negyedik típusa a közelítés kapcsolóknak az induktív közelítés kapcsoló, mely egy vasmagos tekercsből áll. Amikor ezt a tekercset egy fémes tárgy közelébe helyezzük, akkor megváltozik a tekercshez tartozó vasmag mennyisége, ezáltal megváltozik a tekercs induktivitása is. Az induktivitás változását egy rezonátorkör segítségével lehet monitorozni, hisz az áramkörben

folyó áram változik amennyiben egy fémes tárgy jelen van. Ez az áram felhasználható, hogy segítségével ki és be lehessen kapcsolni egy elektromos kapcsolót. Az induktív közelítéskapcsoló működési távolsága 2-15mm.

3.7.3. Fotoelektromos érzékelők és kapcsolók

A fotoelektromos érzékelők lehetnek transzmissziós és reflexiós típusúak. A transzmissziós típusú érzékelők esetén a detektálandó tárgynak meg kell szakítani a fénysugarat, hogy az ne érje el a detektort (3.25. ábra a). A reflexiós típusú érzékelők esetében pedig a detektálandó tárgyról verődik vissza a fénysugár a detektorhoz (3.25. ábra b). Mindkét esetben a fényforrás egy LED dióda. A detektor pedig a legtöbb esetben egy fototranzisztor vagy egy tranzisztor pár (Darlington-tranzisztor pár), mely növeli a detektor érzékenységét. A használt áramkörtől függ, hogy az érzékelő kimenete alacsony vagy magas lesz, amikor a fény eléri a tranzisztort. Az ilyen érzékelők általában egy U alakú tokozásban vannak és csak kis távolságon belül tipikusan 5mm-nél kisebb távolságra működnek (3.25. ábra c).

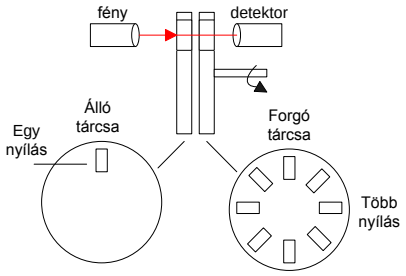
A detektor megvalósításának másik lehetősége a fotodióda. A fotodióda esetében is a használt áramkörtől függ, hogy ha a fotodiódát eléri a fénysugár, akkor annak hatására a kimenetén alacsony vagy magas érték jelenik meg. A detektor megvalósítására egy harmadik lehetőség a fényelektromos cella. A fényelektromos cella ellenállása függ a ráeső fény intenzitásától.



3.25. ábra. – : Fotoelektromos érzékelők.

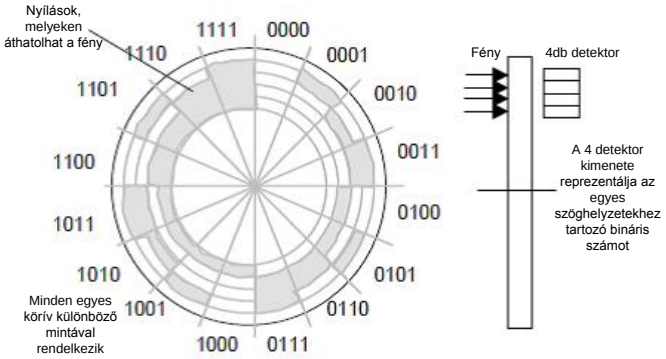
3.7.4. Enkóderek

Az enkóder elnevezés egy olyan eszközre utal, mely valamilyen lineáris vagy szögelmozdulás hatására digitális kimenetet generál. Az inkrementális enkóder valamilyen kezdő pozíciótól mért relatív lineáris vagy szögelmozdulást detektál, míg az abszolút enkóder az aktuális lineáris vagy szöghelyzetet adja meg. A 3.26. ábrán a szögelfordulást mérő inkrementális enkóder felépítése látható. Egy LED által kibocsájtott fénysugár áthaladva az enkóder korongon lévő nyíláson eléri a korong túloldalán lévő fototranzisztort, vagy fotodiódát. Amikor a korong forog, akkor a fénysugarat vagy átengedi, vagy pedig blokkolja az enkóder korong, ezáltal a foto érzékelő kimenetén egy pulzáló kimenet jelenik meg. Az impulzusok száma a koron szögelfordulásával a felbontása pedig a korongon lévő nyílások számával arányos. Például 60 nyílásos korong esetében a minimális detektálható szögelfordulás 6 fok (360°/60=6). Több egymáshoz képest eltolt nyílássorral több ezer nyílás is kiképezhető egy ilyen enkóder korongon, ezáltal sokkal nagyobb felbontást biztosítva. Ezzel az elrendezéssel nemcsak a szögelfordulás mértéke, de az iránya is meghatározható.



3.26. ábra. – : Inkrementális enkóder.

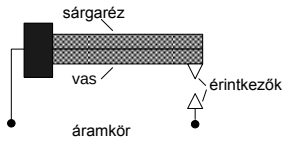
Az abszolút enkóder abban különbözik az inkrementális enkódertől, hogy az enkóder korongon speciális mintázatnak megfelelően vannak kialakítva a nyílások, és minden egyes mintázat egyértelműen definiál egy szöghelyzetet. A 3.27. ábrán látható enkóder korongon 4 darab koncentrikus körben helyezkednek el a nyílások és ennek megfelelően négy darab érzékelőt is tartalmaz ez az enkóder. A nyílások úgy vannak elhelyezve, hogy azok az egyes szöghelyzetekben egy-egy egymástól különböző bináris számot reprezentáljanak. Mivel 4 koncentrikus kör van így a nyílások 4 bites bináris számokat reprezentálnak, ennek megfelelően 24=16 különböző szögpozíciót lehet velük detektálni, tehát a felbontása 360°/16=22,5°. Az abszolút enkóderek tipikusan 10-12 pályát tartalmaznak, ennek megfelelően 10, 12 bites értékek formájában adják meg az enkóder korong aktuális pozícióját. A 10 pályás enkóder korong esetében 2^{10}=1024 szögpozíciót lehet detektálni, ami 360°/1024=0.35°-os felbontást biztosít. A szögérték, a hiba minimalizálása érdekében gyakran gray kódban van megadva.



3.27. ábra. – : Abszolút enkóder.

3.7.5. Hőmérsékletérzékelők

A legegyszerűbb hőmérsékletérzékelők kapcsoló típusúak, megvalósításuk úgynevezett bimetál elemekkel történik. A bimetál két különböző anyagú egymással összeragasztott fémlemezből épül fel (3.28. ábra). A két különböző fémnek eltérő a hőtágulási együtthatója. Ennek következtében, ha a hőmérséklet például emelkedik, akkor a fémlemezke pár elhajlik a kisebb hőtágulási együtthatójú fém irányába. Ha a hőmérséklet csökken, akkor pedig pont a fenti folyamat fordítottja játszódik le. Az a mechanikai mozgás felhasználható egy elektromos áramkör nyitására és zárására ezzel érzékelve azt, hogy a hőmérséklet egy előre definiált érték felett vagy alatt van. Ezek az eszközök általában nem túl pontosak, de igen gyakran használják őket. A PLC digitális bemenetére köthetőek.



3.28. ábra. – : Bimetál lemez.

A hőmérséklet érzékelők egy másik formája a hőellenállás (RTD – Resistive Temperature Detector). A fémek vagy félvezetők elektromos ellenállása változik a hőmérséklettel. A fémek esetében a leggyakrabban a platinát, nikkelt, nikkel ötvözetet használják, hisz ennek az ellenállása igen széles hőmérséklet skálán belül lineárisan változik a hőmérséklettel. A félvezetők, mint a termisztor nagy ellenállás változást mutatnak kis hőmérséklet-változás hatására is, azonban ez a változás nemlineáris. Az ilyen hőmérséklet érzékelőket általában a Wheatstone-híd egyik ágába beépítve szokták alkalmazni, és a híd kimenete tekinthető a mért hőmérsékletértéknek. Egy másik lehetőség, egy feszültségosztó alkalmazása, mely esetében egy fix ellenálláson eső feszültséget mérünk. A hőmérséklet változásával változik a hőellenállás ellenállása, azáltal a fix ellenálláson eső feszültség is. Mindkét esetben egy analóg kimeneti érték formájában kapjuk meg a mért hőmérséklet értéket.

3.7.6. Pozíció és elmozdulás érzékelők

A pozícióérzékelő egy olyan eszköz, amely képes megmérni a távolságot egy referencia pont és az aktuális cél pozíció között. Az elmozdulás érzékelő pedig egy olyan eszköz, mely képes meghatározni a távolságot a cél jelenlegi és korábbi pozíciója között.

Az ellenállás alapú lineáris és szöghelyzet érzékelőket széles körben alkalmazzák és relatíve olcsóak. Ezeket az eszközöket szokás lineáris és rotációs potenciométereknek is nevezni. Pozíció érzékelés megvalósítható még induktív és kapacitív elven működő érzékelők segítségével is. Nagyobb távolságok mérésére alkalmasak az ultrahang és lézerfény távolságmérők. Minden esetben a PLC analóg bemenetére köthető a távolsággal arányos kondicionált jel.

3.7.7. Nyúlásmérő bélyeg

Erő és nyomaték mérésére alkalmas eszköz, amit a gyakorlatban Wheatstone-hídba kötve szoktak alkalmazni. A PLC analóg bemenetér köthető a kondicionált jel.

3.7.8. Nyomás érzékelő

A leggyakrabban alkalmazott nyomásérzékelők diafragma alapúak. A diafragma egy vékony fém vagy műanyag lemezből áll, melynek a szélei le vannak rögzítve. Ha a diafragma két oldalán nyomáskülönbség lép fel, akkor a diafragma közepe elhajlik. Ezen elhajlás miatt fellépő megnyúlást lehet érzékelni nyúlásmérő bélyegekkel, vagy a lemez és egy vele párhuzamosan rögzített fix lemez által alkotott kondenzátor kapacitásának megváltozásával, vagy egy piezoelektromos kristály segítségével.

A nyomáskapcsolók olyan eszközök, melyek segítségével egy bizonyos nyomásérték elérésével nyitni/zárni lehet egy áramkört. A nyomáskapcsolók tipikus megvalósítása a diafragma és a tömlő alapú megvalósítás. A diafragma alapú nyomáskapcsolók kevésbé érzékenyek, de nagyobb nyomást képesek elviselni.

3.7.9. Folyadékszint érzékelő

A nyomásmérő érzékelők felhasználhatóak egy tartályban lévő folyadékoszlop magasságának mérésére. A folyadékoszlop magassága lineáris összefüggésben van az általa létrehozott nyomással. Ezen érzékelők a PLC-k analóg bemenetére kapcsolhatók.

Gyakran csak olyan érzékelőre van szükség, ami akkor jelez, ha a tárolandó folyadék szintje elér egy bizonyos szintet. Ebben az esetben egy olyan megoldás használható, mely esetében egy rúdon egy úszó helyezkedik el. Ha a folyadék szintje emelkedik, akkor az elkezdi

megemelni az úszót a hozzá csatlakoztatott rúddal együtt. A rúd másik végén egy kapcsoló van. Ha a folyadék elér egy bizonyos szintet, akkor a rúd zárja ezt a kapcsolót, ezzel jelezve, hogy a folyadékszint a kívánt értéket elérte. A szintérzékelő kapcsoló a PLC digitális bemenetére kapcsolható.

3.7.10. Folyadékok áramlásmérése

A folyadékok áramlásmérésének egyik gyakori módja, hogy egy szűkítő elemen áramoltatják keresztül a folyadékot, és mérik a szűkítő elem két oldala közötti nyomáskülönbséget.

3.7.11. Intelligens érzékelők

A beágyazott rendszerek alkalmazása a szenzortechnikában lehetővé tette, hogy az egyszerű terepi adatszolgáltatáson kívül számos egyéb, funkcióval, jellemzővel rendelkezzen egy érzékelő. Szinte kivétel nélkül, az intelligens érzékelők digitális kommunikációval rendelkeznek. Napjainkban nagy erőkkel történik az IO-link kommunikáció fejlesztése.

3.8. Kimeneti eszközök

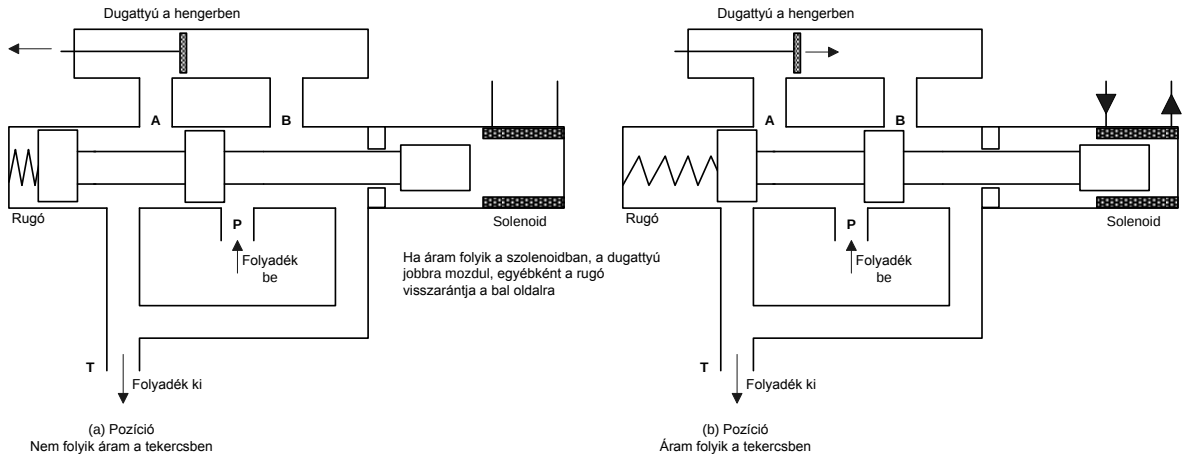
A PLC egyértelműen címezhető digitális és analóg kimenetekkel rendelkezik. A PLC digitális kimenete lehet relés, vagy optocsatolóval ellátott tranzisztoros, vagy pedig triak-os attól függően, hogy milyen eszközt szeretnénk hozzá csatlakoztatni, és a segítségével ki/be kapcsolni. Általában a PLC kimeneti csatornáján kiadott digitális jelet használhatjuk fel, hogy egy végrehajtó szervet és ennek segítségével egy folyamatot vezéreljünk. A végrehajtó szerv vagy aktuátor elnevezés egy olyan eszközre utal, amely elektromos jeleket alakít át valamilyen mechanikai műveletté, mely segítségével az adott folyamat irányítható.

3.8.1. Relék, jelfogók

A tekercsek alapját képezik számos beavatkozó szervnek, aktuátornak. A tekercsen átfolyó áram hatására keletkezett mágneses tér hat a környezetében jelen levő, mágnesezhető anyagokból készített tárgyakra. Egy ilyen elven működő aktuátor a relé, jelfogó vagy kontaktor. Tipikus használatkor, amikor a PLC kimenete bekapcsol, a relé tekercsén áram folyik keresztül, mely hatására létrejött mágneses mező magához vonzza a kapcsoló lemezt, amely ez által zárja a terepi eszköz áramkörét. Ezzel a megoldással, sokkal nagyobb terhelések kapcsolhatóak be, mint csupán egy PLC kimeneteit közvetlenül használva.

3.8.2. Irányvezérlő szelepek

A tekercsek aktuátorként történő felhasználásának egy másik példája a tekercs által működtetett szelep. A szelepek segítségével irányítani lehet nyomás alatt lévő pneumatikus vagy hidraulikus rendszer áramlási irányát, mely segítségével működtetni lehet például egy munkahengerben lévő dugattyút. A 3.29. ábrán egy ilyen megoldás látható, melyben egy tolattyú segítségével vezérelhető egy munkahengerben a dugattyú mozgása. A nyomás alatt lévő levegő hidraulika folyadék a P ponton jut a rendszerbe, míg a T ponton kerül vissza vagy a levegőbe, vagy pedig a hidraulikus rendszerbe. Amikor nem folyik áram a tekercsben (3.29.a. ábra) a hidraulikus folyadék vagy a nyomás alatt lévő levegő a dugattyú jobb oldalához jut és onnan a bal oldali ágon jut vissza a hidraulikus rendszerbe, vagy a levegőbe és a dugattyú balra mozdul. Ha áram folyik keresztül a tekercsen, akkor a szelep a hidraulika folyadék vagy nyomás alatt lévő levegő áramlási irányát megváltoztatja, és így az a dugattyú bal oldalához jut és a jobb oldali ágon hagyja el a rendszert (3.29.b ábra). A dugattyú ennek hatására jobbra mozdul. A dugattyú mozgását fel lehet használni például munkadarabok pozícionálására, vagy szállítószalagról történő leterelésre.



3.29. ábra. – : Solenoid tekercs által működtetet astabil szelep.

3.8.3. Motorok

Mozgásuk alapján megkülönböztetünk lineáris és forgó mozgást végző motorokat. A meghajtó energiától függően több típusú motor ismert. A PLC-s irányítások leggyakrabban a villamos hajtású motorokat alkalmazzák. A villamos hajtások lehetnek egyenáramúak és váltóáramúak. A váltóáramúak lehetnek szinkron és aszinkron hajtásúak, valamint egyfázisú és háromfázisú megoldások.

A DC motorokat megkülönböztethetjük a pólusok száma szerint vagy a kommutátoros és kefe nélküli kialakításuk alapján.

A klasszikus kommutátoros egyenáramú motor állandó mágnesekből álló állórészből áll, melyek között egy tekercselt forgórész található. Ahhoz, hogy a vezeték alkotta hurok elforduljon, a két végét egyenáramú áramforrásra kell kapcsolni, de úgy, hogy közben a vezeték elfordulhasson a saját tengelye körül. Ahhoz, hogy ez megoldható legyen, a vezető hurkot ún. kommutátorra csatlakoztatjuk, melyhez érintkező kefék kapcsolódnak. A kefék biztosítják az elektromos csatlakozást a kommutátorral, miközben az forog, így folyamatos a kapcsolat a vezető hurok és az áramforrás között. A hurokban folyó elektromos áram mágneses mezőt hoz létre, mely kapcsolatba lépve az állandó mágnes mezejével a hurkot elforgatja. A kommutátor fél fordulatonként felcseréli a tekercsben folyó áram irányát, ezzel biztosítva a forgó tekercs és az állandó mágneses mező közötti folyamatos taszítást, és ennek következtében a forgó tekercs folyamatos mozgását.

A kefe nélküli egyenáramú motor esetében nem a tekercs forog az állandó mágnes körül, hanem az állandó mágnes forog a tekercs körül. Egy ilyen motor esetében az áram irányának fél fordulatonkénti felcserélését elektromos áramkör segítségével oldják meg. Ezeknek a motoroknak sokkal jobb a hatékonysága, kisebb elektromos zajt generálnak, olcsóbbak és megbízhatóbbak, mint a kefés egyenáramú motorok.

Mindkét egyenáramú motor esetében a motor forgási sebességét a motor tekercsén átfolyó áram nagysága határozza meg. Mivel azonban fix feszültségű tápegységeket alkalmaznak a tekercsek áramellátáshoz, így a változó áramerősség előállításához külön elektromos áramkör szükséges. Ez az áramkör vezérli a motorra kapcsolt feszültség átlagértékét és ennek következtében az áram erősségét azáltal, hogy változtatja a konstans DC feszültség bekapcsolásának idejét. Ezt az eljárást impulzus szélesség modulációnak nevezzük (PWM – Pulse Width Modulation). A PLC így ennek az elektromos áramkörnek vezérlésével képes az egyenáramú motor nyomatékát változtatni.

Sok ipari folyamat esetében csupán arra van szükség, hogy a PLC be vagy kikapcsoljon egyenáramú motorokat. Ez egy egyszerű relé alkalmazásával is megoldható. A relé alkalmazásának alapszabálya a Lenz törvénye szerinti feszültség, diódával való semlegesítése. Vannak olyan ipari alkalmazások is amikor nem csak be vagy ki kell kapcsolni egy egyenáramú motort, de a

forgásának irányát is meg kell határozni. Ilyenkor fontos megoldani a motor fékezésekor keletkező energia elvezetését.

3.8.4. Léptető motorok

A léptető motor egy olyan egyenáramú motor, mely minden egyes digitális impulzus hatására azonos szögelfordulást produkál. Így ha egy impulzus 1.8°-os elfordulást eredményez, akkor 20 ilyen impulzus 36°-os elfordulást eredményez. A teljes 360°-os körbefordulához pedig 200 impulzusra van szükség. Ennek a működésnek köszönhetően ezek a motorok pontos pozícionálást tesznek lehetővé. A léptető motorok által leadható nyomaték értéke korlátos, így csak a maximális nyomatékuk alatti terhelések esetében alkalmasak a pontos pozícionálásra.

4. Az IEC 61131 szabvány

Az ipari automatizálás fejlődése szempontjából jelentős, hogy szereplői szabványokat követve fejlesszék gyártmányaikat. Az IEC (International Electrotechnical Commission) nemzetközi szabványügyi szervezet célul tűzte ki, hogy szabványokat fejlesszen ki az elektronikával és villamossággal kapcsolatban. A szervezet elérhetősége: http://www.iec.ch/ . A szervezet által kifejlesztett szabványok nagy hányada kapcsolatos az ipari alkalmazásokkal. Az IEC 61131 –es szabvány a programozható vezérlőkkel és a hozzá kapcsolható eszközökkel, mint például a programozást és nyomon-követést végző eszközökkel a PADT –ok (Programming And Debugging Tools) vagy az ember-gép kapcsolatot biztosító HMI (Human-Machine Interface) eszközökkel foglalkozik. A szabvány több részből tevődik össze. Az IEC 61131-1 a PLC és a hozzá kapcsolható perifériák általános dolgaival foglalkozik, megadja a szabványon belüli definíciókat és meghatározásokat. Az IEC 61131-2 meghatározza a szabvány alapvető követelményeit az eszközök iránt, megadja a megfeleltetés tesztjeit. A szabvány által előírt tesztek teljesítése több téren biztosítják a minimum követelményeknek való megfelelést, azaz: az eszközök funkcionalitásával, az elektromos illesztésekkel, a mechanikai tulajdonságokkal, a konstrukcióval, a környezetbe illesztéssel, a szervizeléssel, a biztonsággal és az elektromos zavarokkal kapcsolatban. Az IEC 61131-3 szintaktikai és szemantikai szempontból foglalkozik az eszközök programozásával. Az IEC 61131-4 információkat szolgál az eszközök végfelhasználói számára. Az IEC 61131-5 a kommunikáció üzeneteit határozza meg. Az IEC 61131-6 írja le a PLC kommunikációt a terepi eszközök és az irányítás felsőbb szintje felé. Az IEC 61131-7 a Fuzzy szabályozáshoz határoz meg programozási nyelvet. Az IEC 61131-8 a programozási nyelvek gyakorlati alkalmazásához ad iránymutatókat.

A továbbiakban a PLC-k programozása kerül részletes tárgyalásra.

5. A PLC-k programozása

Ez a fejezet áttekintést nyújt az IEC 61131-3 szabvány szerinti PLC programozási módszerekről és rávilágít a legfontosabb nyelvi elemekre. Az IEC szabványában nem csupán a PLC programozási nyelveket írja le, hanem átfogó elképzelést és útmutatót ad a PLC projektek elkészítésével kapcsolatban is.

Az új nyelvi koncepciók alapját a POU (Program Organisation Unit) alkotja. Amint a nevéből is kitűnik, a POU a felhasználói program legkisebb független szoftveregysége. A POU elvileg megfelel a különböző programozási rendszerek esetében használatos Block-oknak. A POU-k hívhatják egymást paraméteresen vagy paraméterek használata nélkül.

Alapvetően a POU-nak három fajtáját különböztetjük meg: a Függvényt (FUN), a Funkció Blokkot (FB), és a Programot (PROG). A felsorolás a funkcionalitás növekvő sorrendjében lett megadva. A legfőbb különbség a függvény és a funkció blokk között, hogy a függvény minden esetben ugyanazt a kimenetet produkálja, ugyanazon bemeneti paraméterek használata esetén, azaz nincsen memóriájuk. A funkció blokkoknak megvannak a saját adattárolásra szolgáló részei, azért emlékezhetnek a különböző státusz információkra. A program reprezentálja a PLC felhasználói programjának csúcsát, és képes elérni a PLC I/O-kat és más POU-k számára elérhetővé tenni azokat.

Az IEC 6131-3 szabvány előre definiálja a leggyakrabban használt szabványos függvények és funkció blokkok viselkedését és hívási interfészét.

Az IEC 61131-3 szabvány változókat használ, hogy eltároljon és feldolgozzon információkat. A hagyományos PLC rendszerekben a változók flag vagy bit memóriákhoz tartoztak. A felhasználónak már nem kell manuálisan definiálni a változók tárolási helyét, hisz ez a programozási rendszer által automatikusan végrehajtódik és mindegyiknek fix adattípusa van.

Az IEC 61131-3 számos adattípust definiál. Ezek eltérnek egymástól például a bitek számában, vagy az előjel használatában. Arra is lehetőség van, hogy új adattípusokat definiáljunk, mint például a struktúrák vagy a tömbök.

A változók hozzárendelhetőek egy adott I/O címhez és akkumulátorral (szünetmentes táplálás) védett területen is tárolhatóak az áramkimaradások elleni védelem érdekében.

A változóknak különböző formái lehetnek. Definiálhatóak a POU-n kívül és használhatóak az egész programban, deklarálhatóak, mint POU interface paraméter, vagy lehet egy POU-ban valamilyen lokális jelentésük. A deklarációs célok alapján a változók különböző változó típusokra oszthatóak. Minden a POU-ban használni kívánt változót a POU deklarációs részében deklarálni kell.

A kód rész vagy más néven az utasítás rész a PLC által végrehajtandó utasításokat tartalmazza. A POU vagy a szöveges vagy pedig a grafikus nyelvek egyikével programozható.

Minden egyes PLC több feldolgozó egységet tartalmazhat, mint például CPU-k vagy speciális processzorok. Ezeket erőforrásnak nevezi az IEC 61131-3 szabvány. Számos program futhat egy erőforráson. Ezek a programok különbözhetnek a prioritásuk vagy a futási módjuk alapján (periodikus/ciklikus vagy megszakításos). Minden program egy taszk-hoz van rendelve. Egy program akár több taszkhoz is hozzá lehet rendelve. A taszkok indítják a programok futását.

Mielőtt egy program feltölthető lenne egy PLC-re, több információt is meg kell adni annak érdekében, hogy biztosítsuk, hogy a taszk, amelyhez hozzá van rendelve a program az elképzelt tulajdonságokkal rendelkezzen, azaz:

- Melyik PLC típuson és melyik erőforráson fusson a program?
- Hogyan fusson a program, és mi legyen a program futtatásának prioritása?
- Szükség van a változók valós fizikai PLC címhez történő hozzárendelésére?
- Vannak-e globális vagy külső változó deklarációk, melyek más programokra hivatkoznak?

Ezeket az információkat a konfigurációs fájlban adhatók meg.

Az ipari automatizálás területén az IEC-61131-3 szabvány három szöveges és három grafikus programozási nyelvet biztosít a PLC-k programozásához.

A szöveges nyelvek a következők: az utasítás lista (IL – Instruction List), a struktúrált szöveg (ST – Structured Text) és a sorrendi folyamat ábra szöveges verziója (SFC – Sequential Flow Chart). A grafikus nyelvek: a létradiagram (LD - Ladder Diagram), a funkcióblokk diagram (FBD – Function Block Diagram) és a sorrendi folyamat ábra grafikus verziója (SFC – Sequential Flow Chart). A szöveges nyelvek kódrésze utasítások sorozatából épül fel. A grafikus nyelvek grafikai elemeket használnak annak érdekében, hogy megadhassuk velük a PLC kívánt viselkedését. Az elemeket összekötő vonalak, az úgynevezett csatlakozók jelzik az adatok áramlását az egyes funkcióblokkok között.

5.1. A Program Organisation Unit (POU)

A PLC projekt POU-kból épül fel, melynek egy részét a gyártó hozza létre, egy másik részét pedig a felhasználó. A jól megírt és letesztelt felhasználói programok segítségével POU könyvtárak hozhatóak létre, melyek aztán újra felhasználhatóak új projektekben. Az IEC 6113-3 támogatja a szoftver újra felhasználás ezen módját, azzal a kikötéssel, hogy ezek a függvények és funkció blokkok univerzálisak azaz hardver függetlenek legyenek amennyire csak lehetséges.

Az alábbi három POU típus vagy „blokk típus” került definiálására az új szabványban:

POU típus	**Kulcsszó**	**Jelentés**
Program	PROGRAM	A fő program mely tartalmazza az I/O-khoz, a globális változókhoz és az elérési utakhoz tartozó hozzárendeléseket
Funkció blokk	FUNCTION_BLOCK	Be és kimeneti változókat tartalmazó blokk mely a leggyakrabban használt POU típus
Függvény	FUNCTION	Függvény értékkel, be és kimenettel rendelkező blokk az alap PLC műveleti készlet kibővítésére

5.1. Táblázat – : Az IEC 61131-3 három alapvető POU típusa és a jelentésük.

Ez a három POU típus bizonyos tulajdonságokban eltér egymástól:

- Függvény (FUN). Ez egy olyan POU amelyhez rendelhetőek paraméterek, de nincs statikus változója (memória nélküli), mely azt jelenti hogy ha a FUN-t ugyanazokkal a paraméterrel hívjuk meg, akkor minden esetben ugyanazt a kimenetet kapjuk.
- Funkció blokk (FB). Olyan POU amelyhez rendelhetőek paraméterek, és van statikus változója (rendelkezik memóriával). Amikor egy FB-t többször meghívunk ugyanazokkal a bemeneti paraméterekkel, akkor a kimenet eltérő is lehet az FB aktuális állapotától függően. Tehát a kimenetet nem csak a bemenetek, hanem az FB belső változóinak értéke, továbbá a külső változóinak értékei is befolyásolják.
- Program (PROG). Ez a típusú POU reprezentálja a fő programot. A teljes program összes olyan változójának ebben a POU-ban, vagy ezen POU felett kell, deklarálva legyen, amely fizikai címhez van rendelve. Minden más szempontból ugyanúgy viselkedik, mint egy FB.

A PROG-nak és az FB-nek egyaránt lehetnek bemeneti és kimeneti paraméterei. A FUN-nak lehetnek bemeneti és kimeneti paraméterei és függvény értéke, mint visszatérési érték.

A POU egy egységbe zárt elem, mely a program más részeitől függetlenül fordítható. Azonban a fordításhoz a fordítónak szüksége van információkra a POU-ban meghívott további

POU-k hívási interfészével kapcsolatban. A lefordított POU-k később összefűzhetőek, hogy létrejöhessen a teljes program.

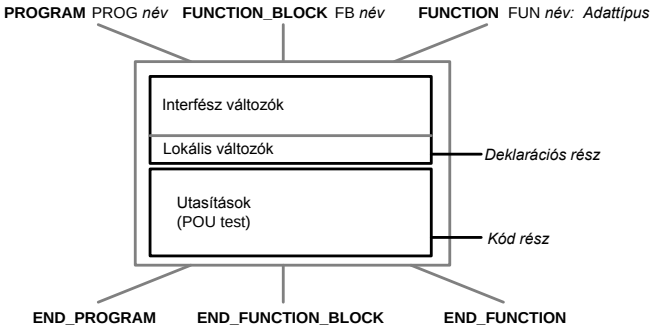
A POU neve a tejes projektben ismert, így egy név csak egyszer használható. A lokális szubrutinok, ahogy más nyelvekben sem az IEC 61131-3 szabványban sem engedélyezettek. Ebből következően, ha egy POU leprogramozásra kerül, akkor a nevét és a hívási interfészét ismerni fogja az összes többi POU a projektben, azaz a POU neve mindig globális.

A POU-k ezen függősége megkönnyíti az automatizálási feladatok széles körű modularizálását, és emellett a már implementált és tesztelt szoftver egységek újrahasznosítását.

5.1.1 A POU elemei

A POU a 5.1. ábrán látható elemekből épül fel:

- POU típus és név (és adat típus a függvények esetében),
- Deklarációs rész a változó deklarációval,
- POU test az utasításokkal.



5.1. ábra. – : A három POU típus általános struktúrája. PROG (jobbl), FB (közép) és FUN (bal).

A deklarációs részben kell definiálni az összes a POU-ban használt változót. Itt meg kell különböztetni a POU-n kívül is látható, valamint a csak a POU-n belül látható lokális változókat. A POU kód részén belül a logikai áramkör vagy algoritmus kerül leprogramozásra a kiválasztott programozási nyelven. A deklarációk és az utasítások grafikus és szöveges formában is leprogramozhatóak.

A deklarációs rész

Az IEC 61131-3 szabványban a változók arra használatosak, hogy inicializáljunk, feldolgozzunk, és tároljunk felhasználói adatokat. A változókat minden egyes POU elején deklarálni kell, azaz meg kell adni, hogy melyik speciális adattípushoz tartoznak.

```
(*Lokális változó*)
VAR          VarLocal: BOOL; END_VAR      (*lokális boolean változó*)
(*Hívási interfész: bemenő paraméterek*)
VAR_INPUT    VarIn: REAL;          END_VAR (*bemenő változó*)
VAR_IN_OUT   VarInOut: UINT; END_VAR      (*be és kimenő változó*)
(*Visszatérési értékek:kimenő változók*)
VAR_OUTPUT   VarOut:REAL;     END_VAR      (*kimenő változó*)
(*globális interface globális/külső változók és elérési utak*)
VAR_EXTERNAL     VarGlob: WORD;   END_VAR (*külső egy másik POU-ból*)
VAR_GLOBAL       VarGlob: WORD;   END_VAR (*globális más POU-k felé*)
VAR_ACCES        VarPath: WORD;   END_VAR (*konfiguráció elérési útvonala*)
```

5.1. példa. – : Különböző változók deklarálása.

Ahogy az 5.1. példában is látható a POU változók deklarációja egymástól elkülönülő szakaszokra van osztva a különböző változó típusokhoz. Minden egyes deklarációs blokk egy változó típushoz tartozik és egy vagy több változót tartalmaz. Ahogy az 5.2. példában is látható, ugyanahhoz a változó típushoz tartozó blokkok sorrendje és száma tetszőlegesen meghatározható vagy függhet attól, hogy a változók hogyan vannak használva az aktuális programozási rendszerben.

```
(*1-es blokk deklarációja*)
VAR         VarLocal1,VarLocal2,VarLocal3:BOOL; END_VAR
(*2-es blokk deklarációja*)
VAR_INPUT   VarIn1:REAL;                        END_VAR
(*3-as blokk deklarációja*)
VAR_OUTPUT  VarOut:INT;                         END_VAR
(*4-es blokk deklarációja*)
VAR         VarLocal4,VarLocal5:BOOL;           END_VAR
(*5-ös blokk deklarációja*)
VAR_INPUT   VarIn2,VarIn3:REAL;                 END_VAR
(*6-os blokk deklarációja*)
VAR_INPUT   VarIn4:REAL;                        END_VAR
```

5.2. példa. – : Példa a deklarációs blokkokra.

A 5.2. táblázat tartalmazza a különböző POU típusától függően a megengedett változótípusokat:

Változó típus	**Engedélyezett a:**		
	PROGRAM	*FUNCTION_BLOCK*	*FUNCTION*
VAR	igen	igen	igen
VAR_INPUT	igen	igen	igen
VAR_OUTPUT	igen	igen	igen
VAR_IN_OUT	igen	igen	igen
VAR_EXTERNAL	igen	igen	no
VAR_GLOBAL	igen	no	no
VAR_ACCESS	igen	no	no
VAR_TEMP	igen	igen	no

5.2. Táblázat – : A három különböző POU-ban használható változó típusok.

Ahogy az 5.2. táblázatban is látható, minden változó típus használható a programokban. A funkció blokkok nem hozhatnak létre más POU-k számára is elérhető globális változókat. Ez egy a programokban az erőforrásokban és a konfigurációkban lehetséges. Az FB-k a globális adatokat a VAR_EXTERNAL változó típuson keresztül érhetik el.

A függvények esetében van a legtöbb megkötés, mert csak lokális be és kimeneti változók használhatóak bennük. A számítások eredményét a függvény visszatérési értékének segítségével lehet megadni.

A lokális változók kivételével minden változó típus felhasználható arra, hogy importáljunk vagy exportáljunk adatokat a POU-ba és a POU-ból. Ez lehetővé teszi az adatcserét a POU-k között.

A POU interfész jellemzői

A POU interfész csakúgy, mint a POU-ban használt lokális adatterület annak érdekében került definiálásra, hogy a deklarációs blokkban a POU változókat a változó típusokhoz rendeljük. A POU interfész az alábbi szakaszokra osztható:

- Hívási interfész: formális paraméterek (bemeneti és be/kimeneti paraméterek);
- Visszatérési értékek: kimeneti paraméterek vagy függvény visszatérési értékek;
- Globális interfész globális/külső változókkal és elérési utakkal.

A POU hívási interfésze és a visszatérési értékek a grafikus nyelvekben is reprezentálhatóak.

A hívási interfész változóit formális paramétereknek is szokták nevezni. Amikor meghívunk egy POU-t, a formális paramétereket az aktuális paraméterekkel helyettesítjük, azaz aktuális változókat vagy konstansokat rendelünk hozzájuk.

Az 5.2. táblázatban található formális bemeneti paraméter (VAR_INPUT) esetében az aktuális paraméterek, mint értékek kerülnek átadásra a POU-nak, azaz a változó maga nem kerül használatra, csak annak egy másolata, vagyis érték szerinti átadás történik. Ez biztosítja azt, hogy ez a bemeneti változó nem változtatható meg a meghívott POU-n belül.

Formális be/kimeneti paraméter (VAR_IN_OUT), esetben az aktuális paraméter a meghívott POU-nek egy pointer formájában kerül átadásra, azaz a változó maga kerül felhasználásra. Ez azt jelenti, hogy a változó olvasható és módosítható is a meghívott POU által. Az ilyen változásoknak automatikus hatása van a POU-n kívül deklarált változókra. Ezt a megoldást referencia alapján történő hívásnak is nevezik.

Formális kimeneti paraméterek, visszatérési értékek (VAR_OUTPUT) nem kerülnek átadásra a meghívott POU-nak, hanem a POU biztosítja azokat, mint értékek. Ebből következően ezek nem részei a hívási interfésznek. A grafikus reprezentációban a VAR_INPUT és a VAR_IN_OUT változók együtt jelennek meg, de a szöveges nyelvekben, mint az IL vagy ST az értékek a POU hívás után kerülnek olvasásra.

Egy másik módszer, hogy értéket adjunk vissza hívó POU-nak a visszatérés érték, mely lehetőséget biztosít, hogy az értékek a hívó példány által kiolvashatóak legyenek. Ez biztosítja azt, hogy a POU kimenti paraméterei védettek legyenek a hívó POU-val szemben. Amikor egy PROGRAM típusú POU kerül, meghívásra a kimeneti paraméterek az aktuális paraméterekkel együtt az erőforrás által kerülnek biztosításra, és a további feldolgozáshoz a megfelelő változóhoz rendelődnek.

Amennyiben egy POU hívásban komplex tömbök vagy adatstruktúrák kerülnek alkalmazásra, mint változók, a VAR_IN_OUT használata egy sokkal effektívebb programot eredményez, mivel ebben az esetben nem magukat a változókat kell futási időben átmásolni a felhasználás helyére csak a hozzájuk tartozó pointereket. Azonban ebben az esetben ezek a változók nem védettek a meghívott POU által végrehajtott esetleges változásokkal szemben.

A formális paramétereknek és a visszatérési értékeknek megvan az a speciális tulajdonságuk hogy láthatóak az őket tartalmazó POU-n kívül, azaz a hívó POU használhatja explicit módon ezen változók neveit, hogy beállítsa a bemeneti értékeket.

Ez a lehetőség egyszerűbbé teszi a POU hívási interfészek dokumentálását és a paraméterek ki is hagyhatóak vagy a sorrendjük felcserélhető. Ebben a kontextusban a be és kimeneti változók is védettek a nem megengedett írás és olvasás ellen.

Az IEC 61131-3 szabvány a VAR_INPUT és VAR_OUTPUT bemeneti és kimeneti változók esetében igen széleskörű hozzáférési védelmet definiál. A bemeneti változók nem módosíthatóak az őket tartalmazó POU-n belül, a kimeneti paraméterek pedig nem módosíthatóak az őket tartalmazó POU-n kívül.

Kód rész

A kód rész közvetlenül a POU deklarációs részét követi és azokat az utasításokat tartalmazza, amelyeket a PLC-nek futtatnia kell. Az IEC 61131-3 5 programozási nyelvet biztosít az irányítási feladat alkalmazás orientált kialakítására.

Ahogy a programozási módszer nagyban különbözik az egyes nyelvek esetében, úgy az egyes nyelveket különböző feladatok megoldására, különböző területeken lehet felhasználni.

SFC	Sorrendi Funkció Ábra: A vezérlési feladat sorosan és párhuzamosan futtatható részekre bontására és emellett az ezen részekből felépítetett rendszer teljes vezérlésére alkalmas . Az SFC tisztán leírja a program végrehajtásának lépéseit, azáltal hogy definiálja azt hogy a vezérelt folyamat mely akciója van engedélyezve, letiltva, vagy végrehajtva. IEC 61131-3 az SFC fontosságát azzal hangsúlyozza ki, hogy úgy nevezi a SFC-t mint a PLC programok strukturálásának fő eszköze.
LD	Létradiagram: A Boole változók grafikus összeköttetése, egy áramkör geometriai nézete, hasonlóan a korábbi relés logikákhoz. A LAD nyelven megírt POU-k szakaszokra bonthatóak melyeket hálózatoknak nevezünk
FBD	Funkció Blokk Diagram: Aritmetikai, Boole, vagy egyéb funkcionális elemek és funkció blokkok grafikus kapcsolata. Az FBD nyelven megírt POU-k is a LAD-hoz hasonlóan hálózatokra bonthatóak. A Boole FBD hálózatokat gyakran LAD nyelven írják meg, és ugyanez igaz fordítva is.
IL	Utasítás Lista: Alacsony szintű gépi nyelv, melyet a legtöbb programozási rendszer támogat
ST	Strukturált Szöveg: Magas szintű nyelv különböző vezérlési feladatokhoz és komplex számításokhoz

5.3. Táblázat – : A programozási nyelvek felhasználási módjai

5.1.2. A Funkció Blokk

A PLC program strukturálásának legfőbb elemei a funkció blokkok. Az FB-ket a programból lehet hívni, és az FB hívhat függvényeket vagy újabb FB-ket.

Az FB-k példányosításának koncepciója nagy jelentőséggel bíz az IEC 61131-3 szabványban és az egyik alapvető megkülönböztető jegye az FB-knek a három POU típus között.

Azt a folyamatot, amikor a programozó úgy, hoz létre változókat, hogy megadja azok neveit és adattípusát a deklarációban példányosításnak nevezzük. A funkció blokkok is példányosíthatók mint a változók.

Az IEC 61131-3 szabványban a számlálást vagy időzítést végző blokkok esetében egy változó névre van szükség melyet a kiválasztott időzítő vagy számláló típus specifikációja követ. Azt a POU deklarációs részében kell deklarálni, és a programozási rendszer automatikusan generál belső abszolút számokat ezen FB változókhoz amikor a POU-t gépi kódra fordítja.

Ezeknek a változó neveknek a segítségével a PLC programozó különböző ugyanolyan típusú számlálót és időzítőt használhat anélkül, hogy ellenőriznie kellene a névütközéseket.

Az IEC 61131-3 szabványban bevezetett példányosítás egységesíti a gyártó függő és a felhasználó által definiált FB-k használatát. A példány nevek úgynevezett szimbolikus neveknek vagy szimbólumoknak felelnek meg a legtöbb PLC programozási rendszerben. Hasonlóan egy FB típus megfelel a hívási interfésznek.

A funkció blokk elnevezést gyakran két kicsit eltérő értelemben használják: egyrészt az FB példány szinonimájaként szolgál, másrészt pedig az FB típusra utal (magának az FB-nek a nevére). Jelen dokumentumban a funkció blokk elnevezés az FB típusra fog utalni.

Az FB példányok láthatóak és használhatóak azon POU-n belül, amelyben deklarálva lettek. Ha globálisként lettek deklarálva, akkor az össze POU használhatja azokat.

Másrészről a függvények mindig láthatóak az egész projektben, és bármelyik POU-ból hívhatóak anélkül, hogy bármilyen előzetes deklarációt igényelnének. Hasonlóan az FB típusok is ismertek az egész projektekben és bármelyik POU-ban használhatóak a példányok deklarálására.

Példányok, mint struktúrák

A példányosítás koncepciója, hogy egy változó struktúrát adott vissza eredményül, mely:

- leírja az FB hívási interfészét mint adat struktúra,
- egy időzítő vagy számláló státuszát tartalmazza,
- a hívó FB-k számára reprezentál egy eljárást.

Ennek köszönhetően egy FB hívásakor igen flexibilisen lehet megadni a paraméter hozzárendelést, ahogy ez az alábbi példában is látható:

```
VAR
    Counter: CTUD;   (*le/fel számláló*)
END_VAR
```

5.3. példa. – : Egy fel/le számláló deklarációja az IEC 61131-3 nyelvben.

Ezután a deklaráció után ennek a számlálónak a bemenete és a kimenete egy adat struktúra segítségével érhető el, mely az IEC 61131-3 szabványban implicit módon definiálva van. Azért hogy tisztábban lássuk, ezt a struktúrát nézzük meg az 5.4. példát, mely egy alternatív reprezentációját mutatja.

```
TYPE CTUD: (*adatstruktúra amely CTUD FB egy alternatív reprezentációja*)
    STRUCT
    (*bemenetek*)
    CU:          BOOL;    (*felfelé számol*)
    CD:          BOOL;    (*lefelé számol*)
    R:           BOOL;    (*reset*)
    LD:          BOOL;    (*betölt*)
    PV:          INT;     (*beállított érték*)
    (*kimenetek*)
    QU:          BOOL;    (*kimenet be*)
    QD:          BOOL;    (*kimenet ki*)
    CV:          INT;     (*aktuális érték*)
    END_STRUCT
END_TYPE
```

5.4. példa. – : Az 5.3. példában látható fel/le számláló adat struktúrájának alternatív reprezentációja.

A 5.4. példában található adat struktúra a szabványos CTUD FB formális paramétereit (hívási interface-ét) és visszatérési értékét mutatja. A hívó szemszögéből reprezentálja az FB-t. A POU lokális vagy külső változói rejtve maradnak.

Ez az adat struktúrát a programozói vagy futtatási környezet menedzseli, és segítségével könnyen elvégezhető az FB-k paraméter hozzárendelése, ahogy az a 5.5. példában is látható az IL programozási nyelv esetében.

```
LD      34
ST      Counter.PV      (*beállított érték, ameddig számol a számláló*)
LD      %IX7.1
ST      Counter.CU      (*felfelé számoljon*)
LD      %M3.4
ST      Counter.R       (*counter reset*)
CAL     Counter         (*FB meghívása aktuális paraméterekkel*)
LD      Counter.CV      (*aktuális érték lekérdezése*)
```

5.5. példa. – : Az 5.3. példában található fel/le számláló paraméterezése és meghívása.

Ebben a példában a Counter példányhoz a 34, %IX7.1 és a %M3.4 paraméterek vannak hozzárendelve, mielőtt a Counter a CAL utasítással meghívásra kerül. Az aktuális számláló érték ezután olvasható ki.

Ahogy azt a 5.5. példában is láthattuk az FB ki és bemeneteit az FB példány neve és egy elválasztó pont segítségével érhetjük el. Ez az eljárás használatos a struktúra elemeinél is.

A nem használt be és kimeneti paraméterek az FB-ben definiált kezdeti értéket veszik fel.

Példány, mint memória

Amikor több változót definiálunk ugyanolyan FB típusúnak, akkor minden egyes példányhoz egyfajta FB adat másolat keletkezik a PLC memóriájában. Ezek a másolatok tartalmazzák a lokális (VAR) és a be valamint a kimeneti változók (VAR_INPUT, VAR_OUTPUT) értékeit, de nem tartalmazzák a VAR_IN_OUT (ezek csak változókra mutató pointerek, nem változók) vagy VAR_EXTERNAL (globális változók) változók értékeit.

Ez azt jelenti, hogy egy-egy példány tárolhat lokális adatokat és be kimeneti paramétereket, azaz a példányok rendelkeznek példányváltozókkal, vagyis memóriával. Ez a memória fontos az olyan FB-k számára mint a flip-flop-ok vagy számlálók, mivel ezen FB-k viselkedése függ a hozzájuk tartozó flag-ek és számláló értékek státuszától. Ennek a memóriának minden egyes változója egy-egy memória területen kerül eltárolásra. Ez a memória terület csak és kizárólag az adott FB példányhoz tartozik. Ebből következően ennek a memóriának statikusnak kell lennie. Ez azonban azt is jelenti, hogy a verem sem használható a szokásos módon az ideiglenes változók menedzselésére.

Ez gyakorlatilag az olyan funkció blokkok esetében melyek olyan nagy adatterületeket kezelnek, mint a táblázatok vagy tömbök az FB példányok nagy statikus memória igényéhez vezethetnek.

IEC 61131-3 szabványnak ezért van egy úgynevezett VAR_TEMP változó típusa. Az olyan változókat kell ilyen típusúnak deklarálni, amelyek értékét nem kell megtartani két hívás között. Ebben az esetben a programozási rendszer egy dinamikus területet vagy vermet használ mely csak akkor érvényes, amikor a példány fut.

Emellett a be és kimeneti paraméterek nagy száma is memória igényes FB példányokhoz vezethet. A VAR_IN_OUT használata a VAR_IN és VAR _OUT helyett csökkentheti az ilyen FB példány memória igényét.

Az FB be és kimeneti változóinak írási/olvasási megkötéseit:

- Egy FB példány bemeneti paraméterei megtartják értéküket a következő hívásig. Ha az FB megváltoztathatja a saját bemeneti paramétereit, akkor ezek az értékek az FB példány következő hívásának hibáját eredményezhetik, mely nem detektálható a hívó POU által.
- Hasonlóan az FB példány kimeneti paraméterei is megtartják az értéküket két hívás között. Ha engedélyezzük a hívó POU számára, hogy megváltoztassa ezeket az értékeket, akkor ez azt eredményezheti a meghívott POU rosszul fogja tudni a saját kimeneteinek státuszát.

Mint a normál változók az FB példányok is védetté tehetőek a RETAIN kulcsszó alkalmazásával, azaz megtarthatják a saját lokális státusz információikat és a hívási interfész értékeit egy áramkimaradás alatt is.

Az FB példányok és az adatblokkok közötti kapcsolat

Egy hagyományos FB meghívása előtt, amely nem tartalmaz lokális memóriát, egy jól bevált gyakorlat az, hogy aktiválunk egy olyan adat blokkot mely például egy receptet vagy FB specifikus adatokat tartalmaz. Az FB-n belül az adat blokk egy lokális adatmemóriaként is szolgál. Ez azt jelenti, hogy a programozó a hagyományos FB-ket különálló példány adattal használhatja, azonban meg kell győződnie, az adatok FB-hez történő egyértelmű hozzárendeléséhez. Ez az adat szintén védett az FB hívások között, mivel az adat blokk globális osztott memória területen helyezkedik el.

Ez a fajta példányosítás csak a funkció blokkok esetében engedélyezett és nem alkalmazható a függvények esetében (FUNCTIONS).

A programok hasonlóan példányosíthatóak és hívhatóak a POU hierarchia legmagasabb szintjét képző Configuration-ban. Azonban ez a fajta példány különbözik az FB-k esetében bemutatott példányosítástól, abban hogy ez különböző taszk-okhoz hozzárendelt futási idejű programok létrehozását eredményezi.

Újrahasznosítható és objektum orientált FB-k

A funkció blokkok esetében számos megkötés van, melyek újrahasznosíthatóvá teszik őket a PLC programokban:

- A funkció blokkokban az olyan változók deklarálása lokális változóként, melyhez fix PLC hardver cím van rendleve tilos. Ez biztosítja hogy az FB hardver független legyen. A PLC címek használata mint globális változó a VAR_EXTERNAL típusban azonban lehetséges.
- Az FB-n belül a VAR_ACCESS vagy VAR_GLOBAL típusú változók deklarálása sem engedélyezett. A globális változók és így indirekt módon az elérési utak a VAR_EXTERNAL változó típus segítségével lehetséges.
- Külső adat csak a POU interfészen keresztül a paraméterek és külső változók segítéségével kerülhetnek átadásra az FB-nek.

Ezen tulajdonságok eredményeképpen a funkció blokkokra úgy is szoktak hivatkozni, mint egy zárt egység, mely azt jelzi, hogy ezek az FB-k univerzálisan használhatóak és mentesek, mindenféle mellék effektustól, mely egy igen fontos tulajdonsága a PLC programot alkotó részegységeknek. A lokális FB adatok és ebből következően az FB funkciók direkt módon nem támaszkodnak semmilyen globális változóra, I/O vagy rendszer szintű kommunikációs útvonalakra. Az FB-k az ilyen adat területeket csak indirekt módon az interfészükön keresztül képesek elérni.

Az FB definíciója: „A funkció blokk egy független zárt adat struktúra egy olyan algoritmussal, amely ezen az adaton dolgozik". Tehét az FB a kód és az adat egybezárása.

Az algoritmust az FB kódrésze tartalmazza. Az adat struktúra az FB példányhoz tarozik és „meghívható" ami nem lehetséges más adatstruktúrák esetében. Minden FB típusból tetszőleges számú példány képezhető, melyek mind függetlenek egymástól. Minden példánynak egyedi neve van saját adat területtel. Csakúgy mint az OOP szemléletmódban az osztályok és példányok.

Ennek köszönhetően az IEC 61131-3 úgy tekint a funkció blokkokra mint objektum orientált eszközökre, azonban ezek a tulajdonságok nem keverendőek össze a mai modern objektum orientált programozási nyelvek tulajdonságaival.

Összefoglalva az FB a saját adat területén dolgozik, mely tartalmazza a bemeneteit, kimeneteit és lokális változóit. A korábbi PLC programozási rendszerekben az Blokkok rendszerint globális adat területeken, mint flag-eken, osztott memóriákon, I/O-kon és adat blokkokon dolgoztak.

Változó típusok az FB-ben

A funkció blokknak tetszőleges számú be és kimenete, lokális illetve külső változói is lehetnek. Emellett vagy ennek egy alternatívájaként lehetőség van az egész FB, vagy akár a lokális és kimeneti változók védetté tételére is az FB deklarációs részében.

Az FB példány maga is védetté tehető a RETAIN kulcsszó használatával, de a bemeneti vagy be/kimeneti paraméterek nem deklarálhatóak, mint védett változók az FB deklarációs részében mivel ezeket a hívó POU adja át és ott kell azokat védetté tenni.

A VAR_IN_OUT esetében meg kell említeni, hogy a változókra mutató pointerek is védetté tehetőek a RETAIN módosító használatával. A hozzájuk tartozó értékek azonban elveszthetik az értéküket, ha azok nincsenek védetté téve a hívó POU-ban.

A hardverfüggetlenségnek köszönhetően a direkt reprezentált változók nem deklarálhatóak, mint lokális változók az FB-n belül. Az ilyen változók csak importálhatóak, mint globális változók a VAR_EXTERNAL használatával.

Az IEC 61131-3 szabványban a változó deklaráció egy speciális tulajdonsága az úgynevezett él vezérelt paraméterek. A szabvány két szabványos FB-t biztosít erre a célra, melyek az R_TRIG és az F_TRIG. Az él detektálás, mint változó típus attribútum csak a bemeneti változók esetében lehetséges.

5.1.3. Függvények

A függvények gyártó vagy alkalmazás specifikus kiterjesztései a PLC által végrehajtható műveletek halmazának.

Az alábbi egyszerű szabály igaz a függvényekre: ugyanazon bemeneti érték hatására mindig ugyanazt a kimeneti értéket és függvényértéket kapjuk, függetlenül attól, hogy milyen gyakran vagy mikor hívjuk meg a függvényt. Az FB-kkel ellentétben a függvényeknek nincs memóriájuk.

A függvények használhatók mint IL operátorok vagy egy ST kifejezésben mint operandusok. A függvények az FB típusokhoz hasonlóan szintén elérhetőek a teljes projektben, azaz a PLC program összes POU-ja által elérhetőek.

A PLC rendszerek alap funkcionalitásának egyszerűsítése és egységesítése céljából az IEC 61131-3 előre definiál egy gyakran használt függvényekből álló halmazt, melyek tulajdonsága, futási idejű viselkedése és hívási interfésze szabványosított.

A felhasználó által definiált függvények segítségével ez a gyűjtemény tovább bővíthető, akár eszköz vagy alkalmazás specifikus függvényekkel.

A függvények más POU típusokkal szemben számos megkötéssel rendelkeznek. Ezek a megkötések azért szükségesek, hogy biztosítható legyen a függvények teljes függetlensége, és hogy használhatóak legyenek kifejezéseken belül, mint például az ST nyelvben.

Változó típusok a függvényekben és a függvény érték

A függvényeknek tetszőleges számú bemeneti és kimeneti paramétere lehet, de pontosan csak egy függvény (visszatérési) értéke.

A függvény érték bármilyen adat típusú lehet, beleértve a származtatott adattípusokat is. Azaz egy egyszerű Boole érték vagy lebegőpontos érték ugyanúgy használható, mint egy tömb vagy valamilyen komplex sok elemmel rendelkező adat struktúra.

Mivel a függvény ugyanazon bemenet hatására mindig ugyanazon értékkel tér vissza, így a függvény nem tárol ideiglenes eredményeket, státusz információkat vagy belső adatokat két meghívása között, azaz a függvények memória nélkül működnek.

A függvények használhatnak lokális változókat a köztes értékek eltárolására, de ezek az értékek elvesznek, amikor a függvény futása véget ér. A lokális változók ebből következően nem deklarálhatóak védett változóként.

A függvények nem hívnak funkció blokkokat, mint időzítők, számlálók, vagy él detektálók, emellett a globális változók sem használhatóak a függvényekben.

A szabvány nem köti ki, hogy a PLC rendszerek hogyan kezeljék a függvényeket, és az aktuális értéküket egy tápellátás kimaradás után. Ebből következően az a POU felelős a szükséges értékek visszaállításáért, ami meghívta a függvényt. Minden olyan esetben, amikor valamilyen fontos adatot kell feldolgozni, használjunk inkább FB-t.

5.1.4. A program

A függvények és funkció blokkok szubrutinokból állnak, míg a PROGRAM típusú POU-k a PLC-k fő programját építik fel. A multitasking-ot támogató vezérlők képesek több főprogram egyidejű futtatására is. Ebből következően a PROGRAM-nak speciális tulajdonságai vannak az FB-hez képest.

Az FB tulajdonságai mellett a PLC programozó az alábbi tulajdonságokat is használhatja a PROGRAM esetében:

- A direkt reprezentált változók deklarálása, mely segítségével elérhetőek a PLC fizikai I/O címe engedélyezettek a PROGRAM-ban,
- A VAR_ACCESS és VAR_GLOBAL használata is lehetséges,
- A PROGRAM a Configuration-ban egy taszkhoz van rendelve, hogy egy futtaható program jöjjön létre, így a programok nem hívhatóak meg explicit módon más POU-ból.

A PLC I/O-k a PROGRAM-ban rendelhetőek hozzá a változókhoz, a direkt reprezentált vagy szimbolikus változók segítségével, mint globális vagy POU paraméterek.

Emellett a program leírja azt is, hogy milyen mechanizmusok használhatóak a kommunikációhoz és a más programokkal történő globális adatcseréhez. A VAR_ACCESS változó típus használható erre a célra.

Ezek a tulajdonságok használhatóak az erőforrás és a konfiguráció szintjén is.

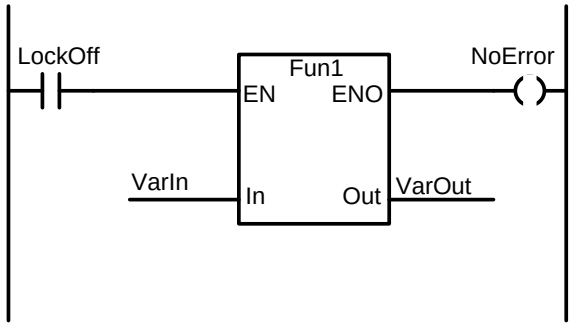
A POU PROGRAM széles funkcionalitásának köszönhetően lehetséges, hogy kisebb projektek esetében Configuration deklaráció nélkül dolgozzunk: a PROGRAM átveszi a program PLC hardverhez történő hozzárendelését.

Ezek a lehetőségek a programozási rendszer funkcionalitásától függenek.

A PROGRAM-ok speciális kezelését és futási idejű tulajdonságait a CPU számára a PROGRAM megfelelő taszk-hoz történő hozzárendelésével adhatjuk meg. A PROGRAM példányosítható így több taszk-hoz is hozzárendelhető és egy időben több is futtatható belőle a PLC-ben. Ez a példányosítás azonban különbözik az FB példányoktól.

Végrehajtás vezérlés az EN és ENO segítségével

A létradiagramban LAD, a függvényeknek van egy olyan speciális tulajdonsága, ami az IEC 61131-3 szabvány többi programozási nyelve esetében nem használható. Ebben a nyelvben a függvénynek van egy kiegészítő bemenete és kimenete. Ezek a Boole EN (Enable In) bemenet és a Boole ENO (Enable Out) kimenet.



5.6. példa. – : A függvény grafikus jelölése az EN/ENO paraméterekkel a LAD nyelvben.

Az 5.6 példa a Fun1 függvény hívásának grafikus reprezentációját mutatja az EN és ENO paraméterek használatával. Ebben a példában a Fun1 csak akkor fut le, ha az EN logikai TURE azaz 1 értéket kap, azaz a Lockoff kapcsoló zárva van. A POU hibamenetes lefutása után az ENO kimenet hasonlóan 1 lesz és a NoError változó értéke változatlan marad.

Az IEC 61131-3 szabvány nem kezeli normális függvény be és kimenetként az EN és ENO paramétereket. Az IEC 61131-3 szabvány programozási nyelvében nem használható az EN/ENO. Az FBD nyelven ennek a használata csak egy kiegészítő tulajdonság.

Amennyiben a programozási rendszer engedi az EN és ENO használatát, akkor az olyan POU-k melyekben használtuk ezeket csak nehezen konvertálhatóak szöveges formátumúvá. Ahhoz hogy ez lehetséges legyen az ST és IL nyelvekben, mint kulcsszavak szerepelniük kell, és le

kell ott generálódjanak. Más esetben az LD/FBD nyelven megírt függvény csak ezeken a nyelveken használható.

5.1.5. Függvények és funkció blokkok hívása

A POU-k kölcsönös meghívása a következők szerint:

- PROGRAM hívhat FB-t s FUN-t,
- FB hívhat FB-t,
- FB hívhat FUN-t,
- A POU hívása nem lehet rekurzív, azaz a POU nem hívhatja önmagát direkt vagy indirekt módon.

A programok és FB példányok hívhatnak FB példányokat. Függvények másrészt nem hívhatnak FB példányokat, hisz egyébként a függvények függetlensége nem garantált.

A Programok a Resource által hívhatóak.

A rekurzió

A rekurzív hívás nem megengedett. Az IEC 61131-3 világosan definiálja, hogy a POU-k nem hívhatják meg saját magukat, sem direkt sem pedig indirekt módon. Ha ez engedélyezve lenne, akkor azt jelentené, hogy a POU definiálhatná saját magát, a saját nevét használva a deklarációjában vagy meghívhatná saját magát a saját POU testében.

Ha a rekurzió engedélyezve lenne, akkor a programozási rendszer képtelen lenne meghatározni a maximálisan szükséges memória területet.

A rekurzió mindig helyettesíthető a megfelelő iteratív szerkezettel például program ciklusok készítésével.

A programozóknak vagy a programozási/PLC rendszernek ellenőriznie kell, hogy nincs-e rekurzív hívás. Ez az ellenőrzés végrehajtható a program készítésekor a POU hívási fa létrehozásával. Történhet rekurzív hívás akkor is, ha a függvény vagy FB példány nem hívja meg magát direkt módon. Ez akkor következhet be, ha kölcsönösen meghívják egymást az egységek.

Az ilyen típusú rekurziók szabály szerint nem lehetségesek az IEC 61131-3 szabványban. A hívási feltétel az alábbiaknak megfelelően van definiálva: ha a POU meghívásra kerül a POU A által, akkor az a POU és az összes a hívási hierarchiában alatta lévő nem használhatja a POU A nevet.

Kiterjeszthetőség és túlterhelés

A szabványos függvényeknek, mint az összeadásnak kettőnél több bemeneti paramétere is lehet. Ezt bemeneti kiterjesztésnek nevezik, és ez tisztábbá teszi ugyanazon függvény használatát több bemeneti paraméter esetén. Egy szabványos függvény vagy szabványos funkció blokk típus túlterhelt, ha ezek a POU-k különböző típusú bemeneti adatelemeken is dolgozhatnak.

Hívás formális paraméterekkel

Amikor egy FUN/FB meghívásra kerül, akkor a bemeneti paraméterek átadódnak a POU bemeneti változóinak. Ezeket a bemeneti változókat formális paramétereknek hívjuk. A bemeneti paramétereket pedig aktuális paramétereknek nevezzük.

Amikor meghívunk egy POU-t, akkor a formális paraméterek explicit módon definiáltak lehetnek, de az is lehet, hogy nincsenek definiálva. Ez a POU típusától (FUN, FB) és a POU hívásához használt programozási nyelvtől függ.

Az FB-kben és a PROGRAM-okban a formális paramétereket mindig explicit módon definiálni kell, a programozási nyelvektől függetlenül.

Az ST nyelvben a FUN meghívható a formális paraméterek megasása nélkül. Egy szabványos függvény számos formális paraméterének nincs neve. Ebből következően ez nem

jeleníthető meg a grafikus reprezentációban, és nem lehet explicit módon megadni a szöveges reprezentációban.

Hívás kihagyott vagy felcserélt bemeneti paraméterekkel

A függvények és funkció blokkok meghívhatóak még ha a bementi paraméter lista nem teljes vagy nem minden paraméterhez van érték rendelve.

Ha a bemeneti paramétereket kihagyjuk, akkor a használt formális paraméterek nevét explicit módon meg kell adni. Ez biztosítja azt hogy a programozási rendszer a megfelelő aktuális paramétert rendeli a megfelelő formális paraméterhez.

Amennyiben a paraméterek sorrendje a FUN/FB hívásban változik, akkor is fontos, hogy a formális paraméterek nevét explicit módon megadjuk. Ez azt jeleneti, hogy az összes formális paramétert specifikálni kell, és a paraméterek sorrendje nem releváns, vagy nincsenek formális paraméterek megadva és akkor minden aktuális paramétert a helyes sorrendben kell megadni. A formális paramétereket mindig specifikálni kell, amikor meghívunk egy FB-t, a függvények esetében ez azonban nyelv-függő.

A bemeneti változókhoz történő hozzárendelés kihagyható, ha a bemeneti változók inicializálása megtörténik a POU deklarációs részében. A hiányzó aktuális paraméter érték helyett ez az érték kerül felhasználásra. Ha nincs a felhasználó által definiált kezdeti érték, akkor az IEC 61131-3 szabványban foglalt szabványos adattípusok alapértelmezett értéke kerül felhasználásra. Ez biztosítja azt, hogy a bemeneti változóknak mindig lesz értéke.

Az FB-k inicializációja csak a példány első hívásakor történik meg. Ezután az előző hívásból származó értékek megmaradnak, hisz a példány adatok védettek.

Egy FB példány neve és komponensei is használhatóak, mint függvények aktuális paraméterei. Ez úgy tűnhet hogy ellent mond annak a kikötésnek, hogy a függvény ugyanazon bemenet hatására ugyanazt a kimenetet hozza létre, és nem hívhatnak FB-ket.

Ez azonban nem annyira ellentmondásos, mint amilyennek tűnik: Az átadott FB példány mint paraméter nem kerül meghívásra, de a bemeneti és kimeneti változói úgy vannak tekintve mint normális adat struktúra elemek.

A függvények és függvény értékek is használhatók mint aktuális paraméterek a függvényekben és a funkció blokkokban. A bemeneti változóknak ugyanaz az adat típusa, mint a függvényeknek és a függvény értékhez rendelődnek, amikor meghívásra kerül.

Az IEC 61131-3 nem ad semmilyen explicit utasítást erről a lehetőségről, így ezt implementáció függővé teszi.

Az FB példányok inicializálása

A funkció blokk példányok eltárolják a bemenet a kimenet és a belső változók státuszát. Ezt hívtuk memóriának a fentiekben. Az FB példányok emellett inicializálhatóak

5.1.6. Változók, adat típusok és közös elemek

Ebben a fejezetben bemutatjuk az IEC 61131-3 szabványban található programozási nyelvek esetében az alapvető közös nyelvi elemek szintaktikáját és szemantikáját.

A szintakszis leírja azokat a nyelvi elemeket, melyek felhasználhatóak az IEC 61131-3-ban definiált programozási nyelvek kapcsán valamint azt hogyan lehet azokat kombinálni egymással, a jelentésüket pedig a szemantika adja meg.

Az első bekezdés az egyszerű nyelvi elemekkel foglalkozik, mely a nyelv alap elemeit reprezentálja. Ezután az adat típus definíció és a változó deklaráció részletes bemutatása következik.

Egyszerű nyelvi elemek

Minden PLC program számos alapvető nyelvi elemből épül fel, melyek összefűzésével létrehozhatóak a deklarációk és/vagy az utasítások és végül a teljes program. Ezek az egyszerű nyelvi elemek feloszthatóak, mint:

- Elválasztók,
- Kulcsszavak,
- Literal-ok,
- Azonosítók.

Az elválasztó karakterek a vesszők, pontok, zárójelek, csillagok, egyenlő, mínusz, plusz karakterek és a pontosvessző.

A kulcsszavak szabványos azonosítók, melyek értelmezése és felhasználhatósága jól definiált az IEC 61131-3 szabványban.

Ezeket ebből következően nem lehet, mint felhasználó által definiált változók vagy egyéb nevekként felhasználni. A kis és nagybetűk használata nem szignifikáns a kulcsszavak esetében, azaz lehet azokat kis vagy nagybetűvel, esetleg ezek keverékeként is megadni. A jobb megkülönböztethetőség érdekében a kulcsszavakat ebben a jegyzetben nagybetűvel írjuk.

A fenntartott kulcsszavak a következők:

- Elemi adattípusok nevei,
- Szabványos függvények nevei,
- Szabványos funkció blokkok nevei,
- Szabványos függvények bemeneti paramétereinek nevei,
- Szabványos FB-k be és kimeneti paramétereinek nevei,
- A grafikus programozási nyelvekben az EN és ENO változók,
- Az IL nyelv operátorai,
- Az ST nyelv elemei,
- Az SFC nyelv nyelvi elemei.

A literálok változók értékeit reprezentálják. A formátum a változók adat típusától függ, mely ebből kifolyólag meghatározza a lehetséges értékhatárokat. Alapvetően három literál típust különböztetünk meg:

- Numerikus literál,
- Karakter sztring literál,
- Idő literálok.

A numerikus és az idő literálok tartalmazhatnak plusz alulvonás karaktereket, a jobb vizuális reprezentáció érdekében. A kis és nagybetű alkalmazásának nincs jelentősége.

A legjelentősebb egység egy időtartam literálban túlcsordulhat, például a t#127m_19s egy érvényes érték, és a programozási rendszer elvégzi a szükséges konverziót, a helyes reprezentáció érdekében, azaz t#2h_7m_19s. Míg az időtartam literál arra szolgál, hogy mérjük és feldolgozzuk a relatív eltelt időt, addig a többi idővel kapcsolatos literál az abszolút napi időt és dátumot adja meg. Az idő és dátum kifejezéséhez használt literálok, reprezentálhatóak rövid formában, vagy teljesen is kiírhatóak, az egyszerűbb olvashatóság érdekében.

A karakter sztring literálokat egyszerű gondolatjelek között adhatjuk meg. A dollár jel ($) prefixként használható, mely segítségével speciális karakterek adhatóak meg a sztring-en belül. A nem nyomtatható speciális karakterek a megjelenítéshez vagy nyomtatáshoz szükséges szöveg formázásához használhatjuk. A dollár jelek és a gondolta jelek sztringen belüli elhelyezéséhez ezeket is meg kell, előzze a dollár jel.

Az azonosítók alfanumerikus karakter sztringek, melyeket a PLC programozó arra használhat, hogy egyedi neveket adjon a változóknak, programoknak, stb. Az azonosítók betűvel, vagy egy alulvonás karakterrel kezdődhetnek, melyet tetszőleges számú betű, számjegy, és alulvonás karakter követhet. Nincs megkülönböztetés a kis és nagybetűs karakterek között, azaz

az EMERG_OFF változó megegyezik az emerg_off és az Emerg_Off változóval. A programozási rendszer ugyanazt a tárolási területet rendeli ezekhez az azonosítókhoz. Az azonosítók hosszát csak a programozási rendszer képességei befolyásolják. IEC 61131-3 szabványban csak az első 6 karakternek kell egyértelműnek lennie, azaz ezek szignifikánsak. Ha például egy programozási rendszerben egy azonosító 16 karakter lehet, akkor a programozónak meg kell győződnie arról, hogy az azonosító első 6 karaktere egyedi: A _DRILLTOOL_8 és a _DRILL az olyan rendszerekben, melyekben csak 6 szignifikáns hely van egy azonosítóban azonosnak tekinthetők. 32 vagy több szignifikáns helyet tartalmazó rendszerek általánosnak tekinthetőek.

A megjegyzések bármilyen olyan helyen alkalmazhatóak, ahol üres karaktereket is lehet alkalmazni, kivéve a sztring karakter literálokat. A megjegyzéseket (*…*) jelek között lehet megadni. A megjegyzéseket nem lehet egymásba ágyazni, és nincs szintaktikus vagy szemantikus jelentősége a deklarációkhoz vagy valamely IEC 61131-3 szabványban definiált nyelvhez.

A szabvány explicit módon tartalmaz pragmákat, melyeket tipikusan a programok automatikus előfeldolgozására és utófeldolgozására használatosak. Ezeket az elemeket zárójelek segítségével azonosíthatjuk. A szintaxisuk és a szemantikájuk a programozási rendszer implementációjától függ. Ebből következően ezek az elemek nincsenek a szabvány által definiálva. A pragmák minden olyan helyen használhatóak, ahol a megjegyzések.

Az adat típusok és a változók jelentései

A változók a POU deklarációs részében vannak deklarálva, azaz a tulajdonságaikkal együtt vannak megadva. A változók deklarációja független a választott programozási nyelvtől, és így egységes a teljes PLC projektben. A deklarációk alapvetően egy azonosítóból és a használt adat típussal kapcsolatos információkból állnak. A típus definíciók alkalmazás specifikus adat típusok és a teljes projektben érvényesek.

A hagyományos PLC programozás esetében hogy a PLC egy memória címét direkt elérhessük, olyan operandusokat kell használni, mint az M 3.1 (flag vagy 3.1-es memória bit) vagy az IW 4 (bemeneti szó 4). Ezek a címek egyrészt lehetnek a PLC központi egységének fő memóriájában, vagy például egy I/O modulban. A címeket tipikusan bit, byte, szó, vagy dupla szó formájában lehet elérni.

A fizikai címekkel megcímzett memória területek különböző céllal használhatóak a PLC programjában: mint egész vagy BCD érték, mint lebegőpontos szám, vagy mint számláló vagy időzítő érték, és így tovább. Ez azt jelenti, hogy a memória cellának van egy speciális adatformátuma minden esetben (8, 16, 32 bit). Ezek az adatformátumok általános esetben nem kompatibilisek egymással, és a programozónak kell tudnia, hogy a PLC címek egy programban milyen formátumban használhatóak.

Hibás program keletkezhet, ha nem helyes memória címet definiálunk, vagy a címet rossz adatformátumban használjuk.

A legtöbb PLC rendszer szimbólum, melyet az abszolút PLC címek ekvivalens helyettesítőjeként használhatunk, ezért be van vezetve, hogy egy sokkal olvashatóbb PLC programot biztosíthassunk. Minden címhez egy egyedi szimbolikus név van rendelve, egy hozzárendelési lista vagy szimbólum tábla használatával.

Az IEC 61131-3 nem csak a PLC címekhez használ változókat, hanem egységesen minden felhasználói adathoz a PLC programban, gyakorlatilag azokhoz az adatokhoz melyeknek nem kell egy speciális memória címhez vagy PLC címhez tartozniuk (általános változók).

A változók tulajdonságait az úgynevezett adat típusok hozzárendelésével határozhatjuk meg. Míg a változó név a változó tárolási területéhez tartozik, az adat típus azt mondja meg, hogy a változó milyen értékeket vehet fel.

Az adat típus meghatározza az olyan változó tulajdonságokat, mint a kezdeti érték, az értékek tartománya vagy a bitek száma. A változó deklarálása során adat típust rendelünk egy azonosítóhoz és ez által ismerté tesszük az adott és más POU-k számára.

Amikor hozzáférünk egy változóhoz, a programozási rendszer ellenőrizheti a változó típus specifikus használatát, azaz a változó az adat típusának megfelelően kerül feldolgozásra. Ez szignifikáns előny a korábbi PLC programozási rendszerekkel szemben, ahol ezek az ellenőrzések rendszer specifikusak voltak, és csak részlegesen lehetett elvégezni, ha egyáltalán el lehetett.

A típus ellenőrzés a PLC program fordítása során a programozási rendszer által automatikusan hajtódik végre. A programozót például figyelmeztetni lehet, ha például egy BYTE típusú változóhoz egy REAL típusú változó értékét rendeli.

Mivel a változó tulajdonsága az adat típus által van meghatározva, az adat formátum helytelen használata okozta hibák nagy száma ezáltal elkerülhetőek.

A POU számos változója esetében nem fontos, hogy a PLC mely memória területén kerülnek eltárolásra, amíg elegendő hely áll a rendelkezésünkre. A korábbi programozási rendszerek esetében egy explicit, manuális memória elosztást kellett alkalmazni, azonban ez könnyen hibákhoz vezetett különösen a komplex számítások és/vagy nagy memória területek esetében.

Az IEC 61131-3 szabványban alkalmazott változó koncepciót használva, az olyan változók, mint az általános változók, a fordítás során a megfelelő PLC memória területre kerülnek. A programozónak nem kell azzal foglalkoznia, hogy a változókat fizikai memória címekhez rendelje. Ez a folyamat hasonló mint amit a magas szintű programozási nyelvek esetében alkalmaznak.

Adat típusok

A hagyományos PLC programozás rendszerek olyan adat típusokat tartalmaznak, mint a lebegőpontos, a BCD, a számláló vagy időzítő értékek, melyeknek gyakran teljesen inkompatibilis a formájuk és a kódolásuk.

Például a lebegőpontos reprezentáció (REAL, vagy FLOAT, stb.) esetében 32 bites adat szavakat használnak, és különböző értéktartományokat alkalmaznak az egész és a törtrész esetében.

A legtöbb hagyományos programozási rendszer egységen használja a BIT, BYTE, WORD, vagy DWORD adat típusokat. Azonban még az egyszerű egész értékek esetében is kicsi, de annál megkülönböztetőbb különbségek vannak a PLC rendszerek és a gyártók között.

Ezért a legtöbb esetben az eltérő adattípusokat tartalmazó programok port-olása két rendszer között csak a program jelentős módosításával lehetséges.

Az IEC 61131-3 szabványnak köszönhetően azonban a leggyakoribb adat típusok szabványosítva lettek, így a PLC-k világában mindenütt azonosak. Ez az olyan szakemberek esetében fontos, akik több különböző PLC-vel is dolgoznak, hogy a feladatukat elláthassák. Az egységes adattípusok alkalmazása az első lépés afelé, hogy port-olható PLC programokat készíthessünk. Az IEC 61131-3-ban számos előre definiált szabványosított adat típus van, melyeket elemi adat típusoknak nevezünk. Ez elemi adattípusok az 5.4. táblázatban vannak összefoglalva.

Boole/bitsztring	**Előjeles egész**	**Előjel nélküli egész**	**Lebegőpontos (Valós)**	**Idő, Időtartam, dátum és karakter sztring**
BOOL	INT	UINT	REAL	TIME
BYTE	SINT	USINT	LREAL	DATE
WORD	DINT	UDINT		TIME_OF_DAY
DWORD	LINT	ULINT		DATE_AND_TIME
LWORD				STRING

5.4. Táblázat – : Elemi adat típusok.

Az elemi adat típusokat egyrészt a szélességükkel másrészt pedig az általuk felvehető lehetséges értékek tartományával jellemezhetjük. Mindkét értéket definiálja az IEC. A fent

leírtaktól kivételt képeznek az idő és adat sztring típusok, hisz ezek implementációfüggőek. A szabványban sem a BCD sem pedig a számláló adat típus nincsen definiálva. A BCD kód manapság már nem olyan fontos, mint korábban, ezért egyedileg kell az ilyen típusokat definiálni egy PLC rendszerben. A számláló értékek implementálhatóak, mint normál egész értékek, semmilyen specifikus formátum nem szükséges legalábbis az IEC 61131-3 szabványos számláló funkció blokkja esetében.

Az alap adat típusokat felhasználva a PLC programozók létrehozhatják a saját adat típusaikat. Ezt az eljárást származtatásnak vagy típus definíciónak nevezzük. Ennek segítségével a programozóknak lehetősége van hogy az adott feladat megoldásához legjobban illeszkedő adat modellt használhassák.

Ezek a típus definíciók globálisak egy PLC projektben. Az új névvel definiált adat típusokat származtatott adattípusoknak nevezzük, és ugyanúgy használhatóak a változók deklarálására, mint az elemi adat típusok. A szöveges leírás használható a típusok definiálásához. Az IEC 61131-3 nem tesz említést a grafikus reprezentációról.

Egy adott feladatot megvalósító programot sokkal hatékonyabban lehet megírni alkalmazás specifikus adattípusok használatával. A felhasználók vagy a gyártók az alábbiakban felsorolt célok érdekében hozhatnak létre, vagy definiálhatnak előre adattípusokat:

- A kezdeti érték eltér a szabványostól;
- Adat típusok tartományok és felsorolás definiálására;
- Többdimenziós tömbök;
- Komplex adat struktúrák.

Ezek a lehetőségek egymással is kombinálhatók és az IEC 61131-3 támogatja is ezt a lehetőséget. Kibővített tulajdonságok rendelhetőek az elemi adat típusokhoz, mint például egy meghatározott kezdeti érték, felsorolás, tartomány, tömb és a struktúra.

A „tömb” és a „struktúra” tulajdonságok alkalmazhatóak a származtatott adattípusok eseten is. Egy adott adattípus többszörös tömbje egy tömbdimenziós tömb típus alakít ki. A „tartomány” tulajdonság csak az egész elemi adattípus és annak leszármaztatott típusai esetében van definiálva az IEC 61131-3 szabványban. A felsorolás típus nem egy valódi értelemben vett származtatott adattípus mivel nem valamilyen elemi adattípusból van leszármaztatva. Azonban az IEC 61131-3 szabvány ilyen módon definiálja a felsorolás adattípust, és ha a programozási rendszer egész értékeket használ, hogy felsorolt adattípust hozzon létre, akkor ez lényegében a származtatás benyomását kelti.

```
TYPE
    Colour          :   (red,yellow,green);      (*felsorolás*)
    Sensor          :   INT(-56..128);           (*tartomány*)
    Measure         :   ARRAY[1..4] OF Sensor    (*tömb*)
    TestBench       :                            (*struktúra*)
        STRUCT
            Place   :   UINT;                    (*elemi adattípus*)
            Light   :   Colour:=red; (*felsorolás típusú változó kezdő
értékkel*)
            Meas1   :   Measure;                 (*tömb típusú változó*)
            Meas2   :   Measure;                 (*tömb típusú változó*)
            Meas3   :   Measure;                 (*tömb típusú változó*)
        END_STRUCT;
END_TYPE
```

5.6. példa. – : Elemi adattípusok tulajdonságának kibővítésére, mint leszármaztatott adattípus.

Az 5.6. példában a kibővített tulajdonságok használatára láthatunk példákat: a Colour egy közlekedési lámpa három színét tudja eltárolni, a Sensor a megengedett hőmérséklet tartományt tárolja, a Measure tömb pedig képes 45 darab mérési érték eltárolására. A TestBench pedig egy olyan adatstruktúra, amely elemi és származtatott adattípusokból épül fel. A tartomány és tömb

indexek ellenőrizhetőek mind statikusan (a programozási rendszer által), mind pedig dinamikusan (futási időben).

A tulajdonságok hozzárendelése segít a programírás közben vétett hibák detektálásában, mely eredményeképpen a létrejött programok biztonságosabbak és nő a program dokumentálhatóságának minősége.

A zárójelben található három elem a felsorolt adattípus esetében a programozó által egyszerűen megadható, mint név, semmilyen előzetes információ nem szükséges. Ennek következtében ezek a nevek, mint „szöveg konstans" értelmezhetőek.

A programozási rendszer automatikusan átkonvertálja egy megfelelő kóddá a red, yellow és a green konstansokat. Az értékek általában belsőleg, mint egész értékek vannak definiálva, elfedve ezeket a programozó elől. A programban a szín értékek nevei direkt módon használhatóak, mint konstansok. Az 5.6. példában a Lignt a red értéket kapja, mint kezdeti érték. A felsorolt adat típusok egyrészt egyszerűsítik a programozási rendszer által végzett automatikus ellenőrzést, másrészt pedig javítják a program olvashatóságát.

Ha a „tartomány" módosító definiálva van egy adattípusnál, mint ahogy az az 5.6. példában is látható a Sensor esetében, akkor hiba üzenet generálódik, ha a programozás vagy a futás során a változó ezen, tartományon kívül eső értéket kap.

A tartomány az ST nyelv esetében a CASE statement-nél is használható annak érdekében, hogy tartományfüggő feladatokat oldhassunk meg.

A tömbök a memóriában egymást követő azonos típusú adatokból épülnek fel. Egy elemre a tömbbeli elhelyezkedése alapján a megfelelő index-el hivatkozhatunk. A legtöbb PLC rendszer esetében hibaüzenetet kapunk, ha a futás közben a tömb határain kívül eső címmel szeretnénk a tömböt címezni.

Az FB egyedek tömbje nem engedélyezett az IEC 61131-3 esetében. Ez azonban egy igen fontos bővítése lehetne a szabványnak hisz így például könnyebben lehetne elérni azonos számlálókat és időzítőket.

Amellett hogy a tömböket lehet definiálni, mint adattípusokat, a tömbök direkt módon is definiálhatóak a váltózó deklarációban.

A STRUCT és az END_STRUCT kulcsszavak között megadva, hierarchikus adatstruktúrák hozhatóak létre. Ezek bármilyen elemi vagy származtatott adattípust tartalmazhatnak. Az FB egyedek nevei szintén nem engedélyezettek az adat struktúrákban, de ez is egy lehetséges kiterjesztése lehetne az IEC 61131-3-nak. Amennyiben egy részelem is egy adat struktúra, akkor egy hierarchikus adat struktúra jön létre. Ennek segítségével a PLC programozók optimálisan adaptálhatják az adat struktúráikat, hogy azok megfeleljenek az elvárásoknak.

Amikor egy adattípust deklarálunk a programozó megadhat kezdeti értékeket is, melyek automatikusan bekerülnek a megfelelő változó deklarációba. Amennyiben az IEC 61131-3-ban definiálttól eltérő kezdeti értéket adunk meg kezdeti értékként akkor, ezek értékek élveznek precedenciát a programozási rendszerben. A tömbök kezdeti értékének megadására érték hozzárendelések sorozatát hozhatjuk létre a megfelelő ismétlési faktor megadásával és a rész sorozatok zárójelbe tételével. A STRING adattípus esetében a kezdeti karakter sztringet definiálhatjuk egyszerű gondolatjelek között.

Az IEC 61131-3 úgynevezett generikus adattípusokat is definiál, annak érdekében, hogy hierarchikusan lehessen kombinálni az elemi adattípusokat különálló csoportokba. Ezek az adattípusok az ANY prefix-xel kezdődnek; például minden egész adattípus (INT) ANY_INT típusúnak nevezhető. Az IEC 61131-3 legtöbb szabványos függvénye nem csak egy, hanem több adattípus esetében is alkalmazató. Például az összeadás függvény (ADD) az összes egész típuson elvégezhető, tehát azt mondhatjuk, hogy az ADD támogatja az ANY_INT adattípust. A generikus adattípusok szabványos függvények leírására használhatóak, azért hogy meghatározhassuk, mely bemeneti és kimeneti változok esetében használható több adattípus. Ez lényegében a függvény overloading eljárás. Ha például a szabványos szorzás függvény (MUL) támogatja az ANY_NUM generikus adattípust, a programozási rendszer minden olyan adattípus

esetében engedélyezi a MUL függvény használatát, mely az ANY_INT és az ANY_REAL típusokban megtalálható, azaz használható előjeles és előjel nélküli egészekre, továbbá lebegőpontos számokra is. A felhasználó által definiált úgynevezett származtatott adattípusokat is lefedi az ANY típus. A direkt módon származtatott adattípusok esetében a változó generikus adattípusa megegyezik a származtatás forrásának adattípusával.

Változók

Ahogy azt már korábban is említettük a változókat az adattípussal együtt deklarálhatjuk. A változók tulajdonságai is ebben a deklarációs részben adhatóak meg, melyek az alábbiakat foglalják magukba:

- Egy elemi vagy származtatott adattípus tulajdonságai,
- Kezdeti értékekkel kapcsolatos kiegészítő információk,
- Tömb határokkal kapcsolatos kiegészítő információk,
- Azon deklarációs blokk változó típusa, melyben a változó deklarálva lett (attribútummal/módosítóval).

Amennyiben a változó deklarációjában megadtuk a RETAIN módosítót, akkor ez a változó a PLC elemmel védett memóriaterületén kerül eltárolásra.

A függvény blokkok példány nevének deklarációja egy speciális esete a változó deklarációnak: Egy FB példány nevet ugyanúgy kell deklarálni, mint egy változót, azzal a kivétellel, hogy az adattípus helyett az FB típus van megadva.

A definíciókhoz hasonlóan a kezdeti étékek és a tömbök a deklaráció során definiálhatóak, mint „névtelen" típusok. Az IEC 61131-3 szabvány nem ad ilyenre lehetőséget a felsorolás, a tartomány és a struktúra tulajdonságok esetében.

A népszerű PLC-s kifejezések, mint a bemenetek, a kimenetek, és a flag-ek speciális módon vannak kezelve az IEC változókkal kapcsolatos koncepciójában.

Ahhoz hogy direkt módon elérhetőek legyenek a PLC rendszer processzorának és az I/O moduloknak az egyes adat területei az IEC 61131-3 két lehetőséget kínál a programozó számára:

- Direkt reprezentált változók segítségével,
- Szimbolikus változók segítségével.

Az ilyen változók deklarálásakor a fizikai memória cím (PLC cím, például az I/O modul címe) az AT kulcsszóval adható meg. Ezek a címek egy „%" jellel kezdődnek melyet az I (bemenet), az O (kimenet), vagy az M (memória/flag) betűk követnek. Ezeket egy újabb betű követi, mely a PLC cím adathosszát határozza meg. Az „X" mely a bit címet jelöli ki is hagyható. Az I/O címekhez és a flag-ekhez történő adattípus hozzárendelés segítségével lehetőség nyílik arra, hogy a programozási rendszer ellenőrizze, hogy a változót helyesen érjük el. Például egy „AT %QD3 : DINT" változó nem érhető el véletlenül egy UINT vagy REAL típussal.

Az IEC 61131-3 szabvány a tömböket és struktúrákat többelemű változóknak nevezi. Ennek megfelelően az egyszerű változókat pedig egyelemű változóknak nevezi a szabvány. Egy tömb adat elemeire a tömb index []-ben történő megadásával hivatkozhatunk. A struktúra elemeire pedig úgy hivatkozhatunk, hogy megadjuk a struktúra nevét, melyet egy pont követ mely után adhatjuk meg az elérni kívánt struktúra adattagot.

A változók akkor kapják meg a kezdeti értéküket, amikor az erőforrás vagy a konfiguráció elindul. Mivel az elemi adattípusok előre definiált alapértelmezett kezdeti értékkel rendelkeznek, ezért az garantált, hogy minden változó valamilyen kezdeti értékkel lesz definiálva.

Amennyiben több lehetőség is van a változók kezdeti érték hozzárendelésére, akkor a legmagasabb prioritással rendelkező szabály kerül alkalmazásra a meleg vagy a hideg újraindítás esetén. Egy meleg újraindítás esetén a védett értéknek prioritása van a változó deklaráció vagy a típus definíció esetében megadott kezdeti értékekkel szemben. Egy adattípus esetén meghatározott alapértelmezett kezdeti érték csak akkor kerül felhasználásra, ha a változó nem védett és nem is lett a deklaráció során hozzá valamilyen kezdeti érték rendelve.

Ha a PLC egy meleg újraindítással kerül elindításra, akkor a védett változó az újraindítás (warm restart) előtti értékét fogja felvenni. Ezt a viselkedést meleg újraindulásnak (warm reboot) is szokták nevezni.

Egy hideg újraindítás után (a PLC felprogramozása után, vagy egy hiba miatt bekövetkezett leállás után), a változók vagy a típus definícióban megadott kezdeti éréküket veszik fel (előre definiált, vagy a felhasználó által definiált), vagy a változó deklaráció esetében megadott kezdeti értéket (felhasználó által definiált) veszik fel (Új Start).

A PLC be- és kimeneteinek valamint egyéb memóriaterületeinek kezdeti érték beállítása implementáció független.

A kezdeti értékek engedélyezettek minden változó típus esetében kivétel a VAR_IN_OUT és a VAR_EXTERNAL. A külső változók ott kerülnek inicializálásra ahol globális változóként lettek deklarálva (VAR_GLOBAL). A VAR_IN_OUT változók esetében az inicializálás nem engedélyezett, mivel ezek a változó típusok változókra mutató pointerek definiálnak, nem pedig magukat a változókat.

Az IEC 61131-3 attribútumokat vagy minősítőket definiál melyek segítségével kibővített tulajdonságok rendelhetőek a változókhoz:

- RETAIN Védett változó (akkumulátoros tápellátással rendelkező memóriaterület) ;
- NON_RETAIN Nem védett változó (akkumulátoros tápellátással nem rendelkező memóriaterület) ;
- CONSTANT Konstans változó (nem módosítható) ;
- R_EDGE Felfutó él;
- F_EDGE Lefutó él;
- READ_ONLY Írásvédett;
- READ_WRITE Írható és olvasható.

A RETAIN, NON_REATIN és a CONSTANT minősítőket az IEC 61131-3 szabvány szerint közvetlenül a változó típust meghatározó kulcsszó után kell írni. Ez azt jelenti, hogy ezek a módosítok a változó deklaráció teljes tartományára vonatkoznak, egészen az END_VAR kulcsszóig.

A többi négy attribútum vagy módosító külön-külön rendelhető egy-egy változóhoz, és nem kombinálható a másik három módosítóval.

A RETAIN kulcsszó azt jelöli, hogy a változó védett, azaz a tápellátás elvesztésekor is megtartja az értékét. A NON_RETAIN változók pedig pont az ellentettjei az előzőnek, azaz ezek a változók explicit módon nem védettek. A RETAIN és NON_RETAIN attribútummal nem rendelkező változók viselkedése a tápellátás elvesztése után implementációfüggő.

A RETAIN és/vagy NON_RETAIN attribútumok a VAR, VAR_INPUT, VAR_OUTPUT és a VAR_GLOBAL változók esetében engedélyezettek, és alkalmazhatóak egy funkció blokk példányában található változók, vagy programok, vagy struktúrák példányainak változói esetében. Azonban nem használhatóak különállóan a struktúra egy-egy elemére.

A CONSTANT módosító olyan változót jelöl, melynek értéke nem változhat a program futása során, azaz ezek a változók írásvédettként kezelendőek.

A CONSTANT módosító a VAR, VAR_EXTERNAL, VAR_GLOBAL változó típusok esetében használható. Az olyan konstansok melyek globálisként vannak deklarálva, a külső használat esetén is konstansként definiálandóak.

A RETAIN és a CONSTANT módosítók egyidejű használatának nincs értelme és nem is engedélyezett az IEC 61131-3 szabványban, mivel a konstansokat minden esetben vissza kell állítani a PLC-ben egy áramkimaradás esetében.

Az R_EDGE és az F_EDGE módosítók olyan Boole változókat jelölnek melyek csak a fel vagy a lefutó élek esetében érvényesek. Ezek a módosítók az IEC 61131-3 szabványban kizárólag csak a VAR_INPUT változók esetében alkalmazhatóak. Azonban az is elképzelhető

hogy ezt a módosítót a VAR és a VAR_GLOBAL változók esetében is alkalmazassuk. Az él detektálás az IEC 61131-3 egy szabványos funkció blokkja által kerül végrehajtásra.

A READ_ONLY és a READ_WRITE attribútumok kizárólag a VAR_ACCESS változók esetében használhatóak. Semmilyen más módosító nem engedélyezett a VAR_ACCESS változó esetében a konfigurációs szinten.

A POU deklarációs része – csakúgy, mint a kód rész – mind grafikus, mind pedig szöveges módon megadható. A grafikus reprezentáció a hívó interface-nek és a POU visszatérési értékének valamit a be és kimeneti értékének jobb vizualizációja érdekében alkalmazható.

Az IEC 61131-3 szabványban definiált grafikus lehetőségeket csak az egyelemű változók esetében lehet használni, és nem alkalmazhatóak a többelemű változók esetében, így ezeknél csak a grafikus reprezentáció jöhet számításba.

Az 5.5. táblázat azt mutatja, hogy mely változó típusok és a POU mely attribútumai reprezentálhatóak grafikusan, és ezzel együtt természetesen szövegesen is.

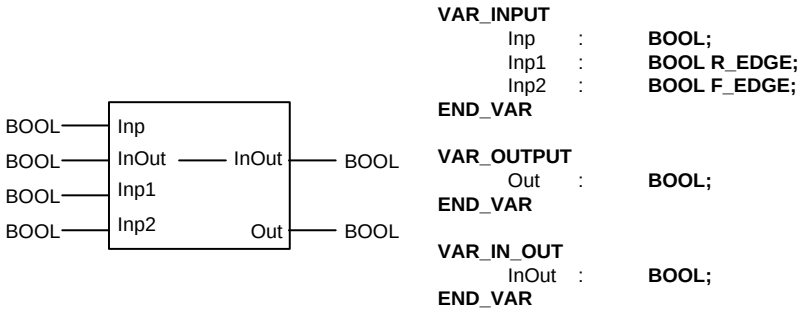
	Grafikus reprezentáció
Változó típus	
VAR	nem
VAR_INPUT	igen
VAR_IN_OUT	nem
VAR_OUTPUT	nem
VAR_EXTERNAL	nem
VAR_GLOBAL	nem
VAR_ACCESS	nem
Attribútum változó típussal	
RETAIN, NON_RETAIN	nem
CONSTANT	nem
R_EDGE, F_EDGE	igen[a]
READ_ONLY, READ_WRITE	nem

5.5. Táblázat – : Grafikusan reprezentálható változó típusok és módosítóik.

Az 5.7. példa egy grafikusan reprezentált deklarációs részt mutat, felhasználva az 5.5. táblázatban definiált lehetőségeket.

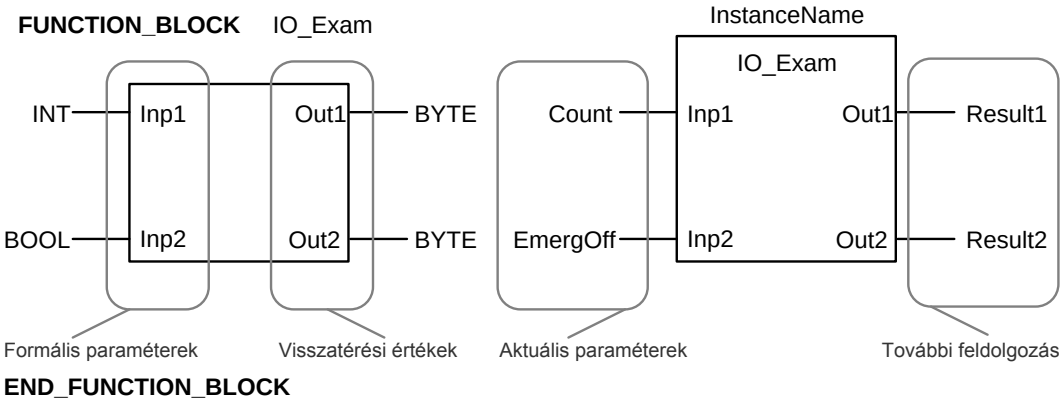
A RETAIN és a CONSTANT módosítók grafikus reprezentálásával az IEC 61131-3 szabvány nem foglalkozik, azonban az adott programozási rendszerben ez a lehetőség is implementálva lehet. A többi változó típus grafikus ábrázolása is elképzelhető.

Ahogy az az 5.7. példában is látszik a deklaráció grafikus reprezentációja a hívó interfészét, és a POU visszatérési értékét vizualizálja.



5.7. példa. – A deklarációs rész grafikus és szöveges reprezentációja.

A POU hívása is reprezentálható grafikusan. Ebben az esetben az aktuális paramétert kell hozzárendelni a formális paraméterhez, és a visszatérési értéket pedig a továbbiakban feldolgozni. Erre látható példa a 5.8. példában.



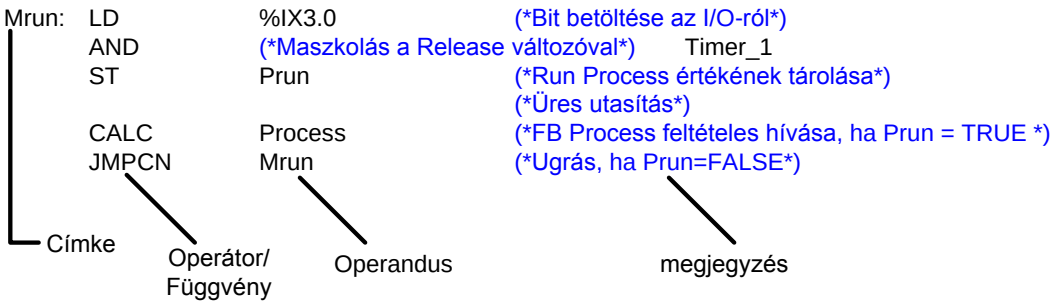
5.8. példa. – A POU hívásának grafikus reprezentálása.

5.2. Utasításlistás programozási nyelv (IL)

Az utasításlistás programozási nyelv (IL – Instruction List), egy hagyományos assembly szerű programozási nyelv. Az IL nyelv univerzálisan használható ezért gyakran köztes nyelvként is használják, amelyre más grafikus vagy szöveges nyelveket alakítanak át. Az IL nyelven az utasítások (instruction) operátorok vagy függvények és a hozzájuk tartozó egy vagy több operandusból épülnek fel. Az operátoroknak rendszerint egy a függvényeknek pedig egy vagy több operandusuk lehet.

5.2.1. Utasítás az IL nyelvben

Az IL egy sor-orientált nyelv. Az utasítás, azaz a PLC által futtaható parancs egyetlen sorba van megadva. Üres utasítások egy-egy üres sor formájában adhatóak meg. Az IL nyelvben; az operátor és az operandus közé elválasztó karakterként legalább egy szóköz karakter szükséges. A megjegyzések a (*…*) jelek között bárhova elhelyezhetők ahova az üres karakterek is.



5.9. példa. – Különböző IL utasítások.

A címke és a megjegyzés sorok az IL nyelvben csak opcionálisak. A megjegyzések nem csak a sorok végén helyezhetőek el, hanem a bárhova ahova üres karakter rakható, mint más programozási nyelvek esetében is.

A címkék azért szükségesek, mert csak a segítségükkel lehet a program futása során a programkód sorai között ugrásokat megvalósítani. Adhatunk címkét üres sornak is és ilyenkor az azt követő sorban található utasítás fog végrehajtódni.

A különálló operátorok/függvények írják le az adott műveletet, az operandusok pedig bementi paraméterek.

A megjegyzések mindegyik nyelvben azonosak, és két zárójel közötti csillag (*…*) segítségével adhatóak meg. Ezek általában a sorokban található utasítások értelmezéséhez tartalmazhatnak információkat.

Nincs előre definiált formátuma az operátoroknak vagy az operandusoknak, akármennyi üres karakterrel, vagy tabulátorral el lehet azokat választani. Az operátor első karaktere bármelyik oszlopban kezdődhet.

A pontosvessző „;” nem használható az IL nyelvben, sem mint “megjegyzés kezdődik”, sem pedig mint utasítás elválasztás.

5.2.2. Az univerzális akkumulátor (CR – Current Result)

A szabványos asszembler-ek rendszerint a processzor valamilyen valós hardver akkumulátorát használják a műveletek elvégzéséhez. Például az operátorban megadott operandus betöltődik az akkumulátorba, végrehajtódik rajta a művelet, melynek eredménye szintén az akkumulátorba kerül, majd innen az eredmény valamilyen memóriahelyen kerül eltárolásra.

Az IL nyelvben szintén van egy akkumulátor a CR (Current Result). A CR regiszternek nincs fixen meghatározott mérete, mint a valós hardver regisztereknek. Az IL nyelvi fordítók azt feltételezik, hogy ez a virtuális tároló elem tetszőleges méretben rendelkezésre áll. A mérete mindig az aktuálisan feldolgozandó operandustól függ. A CR által tárolható adat típus mindig annak megfelelően változik, hogy milyen az éppen aktuálisan feldolgozandó adat típusa.

Az IL nyelv esetében alkalmazott asszembler esetében nincsenek speciális státusz bitek. Egy logikai kifejezés kiértékelésének eredményeképpen kapott 0 (FALSE) vagy 1 (TRUE) érték a CR regiszterben tárolódik. A következő feltételes ugrások és függvényhívások a CR regiszterben tárolt TRUE vagy FALSE értékeknek megfelelően kerülnek végrehajtásra.

A CR regiszter lehet:

- Elemi adat típus;
- Származtatott adat típus;
- Funkció blokk típus.

Az IL esetében csak az a fontos hogy a két egymást követő művelet kompatibilis legyen, azaz például a CR adattípusának meg kell, egyezzen a következő utasítás adattípusával.

A műveletek csoportjainak hatása a CR regiszterre	Jelölés	Példa műveletek
Létrehozás	C	LD
Feldolgozás	P	GT
Változatlanul hagy	U	ST;JMPC
Nemdefiniáltra vált	-	CAL= funkcióblokk feltétel nélküli hívása

5.6. Táblázat – : A különböző operátor csoportok hatása a CR-re.

Az 5.6. táblázatban található lista a különböző operátor csoportok CR regiszterre gyakorolt hatását mutatják.

A változatlanul hagyja operátor csoport a CR regiszter tartalmát nem változtatja meg, az a következő utasítás pont úgy kapja meg a CR regisztert ahogyan az azt megelőző megkapta.

A nem definiáltra állítja operátor csoport esetében a CR regiszter nem használható a következő utasításban. Az első utasítás egy FB-ben így mindenképpen egy LD, JMP, FB hívásnak, vagy RET utasításnak kell, hogy legyen, mert ezek számára nem szükséges hogy a CR érvényes legyen.

Az IEC 61131-3 önmagában nem definiál operátor csoportokat, azonban a csoportok alkalmazásával, sokkal könnyebben megérthető, hogyan lehet az IL nyelv utasításait helyesen egymás után fűzni. Az IEC a CR regiszter hatását és kiértékelését csak alapvető szinten adja meg. Az olyan utasítások, mint az ADD és az OR esetében a CR értéke és típusa a műveletek elvégzése előtt és után pontosan le van írva. A szabvány azonban nem definiálja a CR tartalmát és típusát például egy feltétel nélküli ugrás utasítás esetében. A fenti táblázatban megadott operátor csoportok csupán segítségként szolgálnak a szabvány megértéséhez, és csupán a szabvány értelmezései, nem részei annak. Különbözőképpen lehetnek implementálva a különböző programozási nyelvekben.

5.2.3. Operátorok

Ebben az részben az IL nyelv operátorai kerülnek bemutatásra. Bizonyos műveleteknek vannak módosítói. Egy művelet az N vagy a C módosítóval ellátva kibővített jelentéssel bír.

A módosítók:

- N A műveletek tagadása;
- (Egymásba ágyazás zárójellel;
- C Feltételes utasítás végrehajtás.

Ezen módosítók jelentését az alábbi példa illusztrálja:

```
 1.: VAR         Var1: BOOL := FALSE;  END_VAR
 2.: LDN         FALSE  (*TRUE{BOOL}; ekvivalens LD 1-el*)
 3.: ANDN   Var1        (*TRUE{BOOL}*)
 4.: LD     Var1        (*Var1-et CR-be tölteni majd késleltetni*)
 5.: AND (  Var2        (*A zárójelben lévő kifejezés kiszámítása*)
 6.: OR     Var3        (*majd az ÉS művelet elvégzése CR-ben lévő Var1-el*)
 7.: )
 8.: ST     Var4             (*Eredmény tárolása*)
 9.: LD     1                (*1*)
10.: ADD (   2               (*2*)
11.:    ADD (    3           (*3*)
12.:        ADD     4        (*7*)
13.:    )                    (*9*)
14.: )                       (*10*)
15.: ST     Var5             (*az eredmény 10*)
```

5.10. példa. – Az operátorok használata.

A példában az N tagadási módosító használatánál a CR típusa és értéke a {}-ben látható. A „( )” zárójel módosító segítségével a CR regiszter és egy teljes utasítássorozat eredményén végezhetünk logikai műveletet. Amikor a „(” módosítóhoz ér a végrehajtás, akkor az operátor típusa, valamint a CR tartalma és típusa elmentésre kerül és egy új érték és típus kerül a CR-be, Amikor a „)” módosítóhoz érkezik a végrehajtás, akkor az elmentett érték visszatöltődik, és a módosított operátor és a CR aktuális értéke feldolgozásra kerül. Az eredmény a CR-ben tárolódik el. A szabvány nem határozza, meg hogy hogyan kerülnek a „(” használatakor az értékek elmentésre, de általában egy verem használható erre a célra. A példában, a Var4:=Var1 AND (Var2 OR Var3) esetében megfigyelhető a zárójel operátorok blokkonkénti végrehajtása. A példa utolsó része illusztrálja egy zárójelekkel beágyazott kifejezés kiszámítását, az eredmény a Var5 – ben kerül eltárolásra.

A műveletek feltételes végrehajtása alkalmazhatók bizonyos operátorok, mint például a GT egy logikai értéket generál, mely a CR regiszterben kerül eltárolásra. Ha ez az érték logikai igaz, akkor a következő utasítás hajtódik végre, egyébként átlépésre kerül a következő utasítás és az azt követő utasítás kerül végrehajtásra. Az irodalomban megtalálható az operátorok teljes áttekintése [4].

5.2.4. Függvények és funkció blokkok használata

A függvény hívása az IL nyelvben egyszerűen a függvény nevének a megadásával történik. Az aktuális paraméterek a függvény neve után zárójelben szerepelnek. A szintakszis megegyezik a több operandusos operátorok esetében alkalmazandó szintakszissal.

A formális paramétereknek is adhatunk értéket soronként a := jel segítségével, mely csak ebben az esetben használható.

A függvény első paramétere a CR regiszterben tárolt érték. Ebből következően ezt az értéket a függvény hívása előtt a CR-be kell beletölteni. Ezek alapján tehát a függvényhívás első operandusa a függvény második paramétere, és így tovább.

A függvény legalább egy kimeneti értékkel tér vissza. Ez az érték a CR-ben tárolódik és az adat típusát a függvény határozza meg. További visszatérési értékek paraméter hozzárendeléssel adhatóak meg. Ha egy függvényt paraméterek nélkül hívunk meg, a deklarációs sorozatot meg kell adjuk. A formális paraméter átadása esetében ez sorról sorra történik egy összefogó zárójelpárban. A programozási rendszer a függvény értéket a függvény nevével megegyező változóhoz rendeli. Ez a név automatikusan deklarálódik, és nem kell a felhasználónak a hívó FB-ben külön deklarálnia.

Az EN/ENO jelentését a korábbiakban adtuk meg. Az EN/ENO paraméterekkel rendelkező POU-t mindig formális paraméterekkel kell meghívni, nemcsak aktuális paraméterekkel.

Függvény hívás

```
VAR
        FirstFunPar: INT :=10;
        Par2: INT :=20;
        Par2: INT :=30;
END_VAR

LD      FirstFunPar
UserFun Par1, Par2
(*Második hívás:*)
UserFun Par1, Par2
ST      Sum
```

Függvény deklarálása:

```
FUNCTION        UserFun :INT
        VAR_INPUT
                FunPar1,FunPar2,FunPar3:INT;
        END_VAR

        LD      FunPar1
        ADD     FunPAr2
        ADD     FunPar3
        ST      UserFun (*Visszatérési érték elhagyható*)
        RET     (*szintén elhagyható*)
END_FUNCTION
```

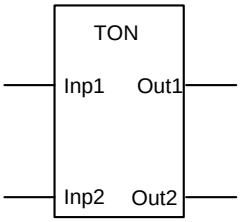
5.11. példa. – : A UserFun két egymást követő hívása.

Az 5.11. példa illusztrálja a UserFun két egymást követőhívását. Első esetben a FunPar1 értéke 10, a másodiknál 60. A FunPar2 20 a FunPar3 pedig 30. A számítás végén 120 tárolódik el a Sum-ban.

Egy funkcióblokk a CAL operátor segítségével hívható meg. Az IEC 61131-3 az IL nyelv esetében alapvetően három módszert határoz meg a FB híváshoz, éspedig:

- A híváskor adjuk meg az aktuális be és kimeneti paraméterek listáját egy zárójelben,
- Betöltjük és elmentjük az aktuális bemeneti paramétereket az FB hívás előtt,
- Implicit hívás, a bemeneti paramétereket mint operátorokat használva.

A harmadik módszer csak a szabványos FB-k esetében használható, nem használható a felhasználó által definiált FB-k esetében. Ebből következik, hogy az utolsó módszert ritkán alkalmazzuk a gyakorlatban.



5.2. ábra. A TON szabványos funkció blokkja két bemeneti és két kimeneti paraméterrel.

Az 5.2.ábrán látható TON FB három módszerrel való megvalósítását az 5.12. példa mutatja be.

```
VAR
  Rel, Out:     BOOL := 0; (* Release - Input; Output *)
  Time1:        TON;       (* Std - FB TON has the formal parameters *)
                           (* IN, PT (input) and Q, ET (output *)
  Value         TIME;      (* Set - Input *)
END_VAR
```

(* Method 1 *)	(* Method 2 *)	(* Method 3 *)
(* Supplying the parameters: *)	LD t#500ms ST Time1.PT LD Rel ST Time1.IN	LD t#500ms PT Time1 LD Rel
(* Call: *) CAL Time1 (IN:=Rel, PT:= t#500ms, Q=>Out, (* Output p. 1 *) ET=>VALUE (* Output p. 2 *))	CAL Time1	IN Time1
	(* Utilisation of the output parameters *) LD Time1.Q ST Out LD Time1.ET ST Value	

5.12. példa. – : Time1 a TON FB-ként van deklarálva. A bemeneti paraméterek, a Time1 meghívása, a kimeneti paraméterek kiértékelés is látható a példában.

A deklarációs rész és a kimeneti paraméterek kiértékelése megegyezik mindhárom módszer esetében, a bemeneti paraméterek megadásának módjában van csak különbség.

Ha egy programozási rendszer több, módszert használ, akkor a deklaráció, a kiértékelés és a paraméterek megadási módja tetszőlegesen keverhető.

Az 1-es eljárás módosításaként a hívás elvégezhető aktuális paraméterekkel is mint a függvényhívás esetében. Ebben az esetben az aktuális paraméterek sorrendben adhatóak meg, vesszővel elválasztva.

A 3-as módszert használva, a PT operátor inicializálja a PRESET időt. Az IN operátor elindítja a Time1 FB-t. A 3-as módszer nem változtatja meg a parancssorban meg nem címzett paramétereket, ezáltal azok korábbi értékei lesznek használva. A 3-as módszer használatakor elővigyázatosan kell eljárni, mert az FB meghívásakor nem mindig világos az, hogy mikor hívunk FB-t és mikor adunk meg bemeneti paramétereket.

5.3. Strukturált programozási nyelv (ST)

Az IL-hez hasonlóan a strukturált szöveg is az IEC 61131-3 szabvány egy szöveges nyelve. Az ST egy magas szintű nyelv, mert nem használ alacsony gépi szintű műveleteket, de számos absztrakt utasítást tartalmaz, melyek nagyon rövid formában adnak meg nagyon komplex funkciókat. Az ST magas szintű programozási nyelvben az utasítások (statement) különböző ST kulcsszavakból álló úgynevezett kifejezésekből (expression) épülnek fel, melyek vezérlik a program futását. A kifejezések operátorokból/függvényekből és a hozzájuk tartozó operandusokból épülnek fel, és futási időben értékelődnek ki.

Az ST nyelv előnyei az IL-hez képest:

- A programozási feladat nagyon tömör formában megoldható,
- A program könnyű átláthatósága az utasítás blokkok alkalmazásával,
- A vezérlési folyamat megadására kialakított szerkezetek.

Az előnyök magukban hordozzák a hátrányokat is:

- A gép kódra történő fordítás a felhasználó által nem követhető nyomon, hisz automatikusan történik.
- Az absztrakció magas szintje miatt elveszhet a hatékonyság (a lefordított programok hosszabbak és lassabbak lehetnek).

5.3.1. ST utasítások

Az ST nyelvű program, számos utasításból áll. Az utasítások (;)-val vannak elválasztva egymástól. Az IL-től eltérően az „end of line”, új sor, szintaktikailag space-ként, fehér karakterként van értelmezve. Az utasítás végét a „;” karakter jelenti. Ez azt jelenti, hogy egy utasítás több soron keresztül is folytatódhat, vagy több utasítás is megadható egy sorban. Hasonlóan, mint a PASCAL nyelvben.

A megjegyzések az IL nyelvhez hasonlóan a (*…*) jelek között adható meg, akár az utasításokon belül is, bárhol ahova fehér karakter rakható.

A:= B (*alap érték*) + C (*hőmérséklet*);

Bár az ST nyelv szabad formátumú, a kód olvashatósága érdekében a programozóknak célszerű a megfelelő tagolást, struktúrát alkalmazni. Az értelmes, leíró azonosítók és a megfelelő megjegyzések alkalmazása olvashatóvá teszi a kódot. Az ST utasítások az 5.7. táblázatban találhatóak.

Kulcsszó	Leírás	Példa	Magyarázat
:=	Hozzárendelés	d:=10	Egy kiszámított érték hozzárendelése a baloldalon álló azonosítóhoz
	Fb[a] hívása	FBName(	POU hívása a paramétereivel együtt ":=" bemeneti paraméterek, "=>" kimeneti paraméterek
		Par1:=10,	
		Par2:=20,	
		Par3=>Res);	
RETURN	Visszatérés	RETURN	Kilépés az aktuális POU-ból, vissza a hívó POU-hoz
IF	Kiválasztás	IF d < e THEN f:=1;	Boole kifejezések alapján történő alternatíva kiválasztás
		ELSIF d=e THEN f:=2	
		ELSE f:=3;	
		END_IF;	
CASE	Többszörös kiválasztás	CASE f OF	Utasítás blokk kiválasztása az f kifejezés értéke alapján
		1: g:=11;	
		2: g:=12;	
		ELSE g:=FunName();	
		END CASE;	
FOR	Itteráció (1)	FOR h:=1 TO 10 BY 2 DO	Egy utasítás blokk többszörös ciklusa start és stop feltétellel és egy inkrement értékkel
		f[h/2]:=h;	
		END _FOR	
WHILE	Itteráció (2)	WHILE m>1 DO	Egy utasítás blokk többszörös ciklusa a ciklus elején megadott leállási feltétellel
		n:=n/2;	
		END_WHILE;	
REPEAT	Itteráció (3)	REPEAT	Egy utasítás blokk többszörös ciklusa a ciklus végén megadott leállási feltétellel
		i:=i*j;	
		UNTIL i<1000	
		END_REPEAT;	
EXIT	Ciklus vége	EXIT;	Kilépés az iterációs utasítás -ból
;	Üres utasítás	;;	

5.7. Táblázat – : Az ST utasítások.

Az ST nyelvben, amint az más strukturált programozási nyelv esetén megszokott, nincs feltétel nélküli ugrásutasítás (GOTO). Ez nem funkcionális megkötés, hiszen az ugrásutasítás a nem strukturált programok sajátja. A PLC-s alkalmazások esetében a feltétel nélküli elágazás használata növeli a hatékonyságot, például akkor, amikor vészhelyzet esetén egy többszörösen beágyazott kifejezés végrehajtásából gyorsan ki kell lépni.

Az 5.13. példa. egy sztereo magnetofon vezérlését végző, ST nyelven íródott programot mutat be.

```
FUNCTION_BLOCK
    VAR_INPUT
        BalControl: SINT(-5..5);        (*Balance szabályzás -5, 5 egész tartományban*)
        VolControl: SINT(0..10);        (*Hangerő szabáylzás 0, 10 egész tartományban*)
        ModelType:  BOOL;               (*2 model típus: TRUE vagy FALSE*)
    END_VAR

    VAR_OUTPUT
        RightAmplif:    REAL;           (*Szabályzó a jobb erősítőhöz*)
        LeftAmplif:     REAL;           (*Szabályzó a bal erősítőhöz*)
        LED:            BOOL;           (*Figyelmeztető LED be:1; ki:FALSE*)
    END_VAR

    VAR_IN_OUT
        Critical:       BOOL;           (*Visszatérési érték*)
    END_VAR

    VAR
        MaxValue:       REAL := 26.0;   (*max. erősítő bemenet; meghatározott ideig aktív*)
                                        (*figyelmeztető led bekapcsolása*)
        HeatTime:       TON;            (*standard FB (time delay) a túlvezérlés
szabáylzására*)
        OverDrive:      BOOL;           (*túlvezérlés helyzete*)
    END_VAR

    (*Jobb erősítő vezérlése a hangerő és balance beállításoktól függően*)
    RightAmlif:= Norm(      LCtrlK:=VolControl,
                            BIK:=ABS(BalControl-5),
                            MType:= ModelType);

    (*Túlvezérlés?*)
    IF MAX(LeftAmpliff, RightAmpliff)>= MaxValue
    THEN
        Overdrive:=TRUE;
    ELSE
        Overdrive:=FALSE;
    END_IF;

    (*Túlvezérlés több mit 2 másodperce?*)
    HeatTime(IN:=Overdrive, PT:=T#2s);
    LED:=HeatTime.Q;
    IF HeatTime.Q= TRUE THEN
        Critical:=1;
    END_IF;
END_FUNCTION_BLOCK

FUNCTION
    VAR_INPUT
        BIK:    SINT;                   (*skálázható balance kontroll*)
        LCtrlK: SINT;                   (*Hangerő szabályzás*)
        MType:  BOOL;                   (*2 típus; TRUE vagy FALSE által leírva*)
    END_VAR
    TYPE
        CalType: REAL :=5.0; (*Adattípus speciális kezdőértékkel*)
    END_TYPE
    VAR
        Calib:  CalType;    (*Skálázási érték az erősítő kimenetére, 5-el inicializálva*)
    END_VAR
    (*Kiértékelni az egész számokat az erősítő számára függően a model típusától valamint a
beállítógomboktól*)
        Norm:=  SINT_TO_REAL(BIK) +     (*balance értéket venni*)
                Calib +                 (*hozzáadni a skálázási értéket*)
                SEL(G:=MType, IN0:= 4.0, IN1:=6.0) + (*Model típusától függő értékeket
hozzáadni*)
                SINT_TO_REAL(LCtrlK);   (*és a hangerő értékét hozzáadni*)
END_FUNCTION
```

5.13. példa. – Egy sztereo kazettás magnóhoz ST nyelvű vezérlőprogramja.

A vezérlő program a következő funkcionalitásokkal vértezi fel a magnetofont:

1. A hangszórók hangerő vezérlése az aktuális balance (integer érték -5 és +5 között) és hangerő (integer érték -10 és +10 között) beállításoknak megfelelően. Az erősítő kimenetének REAL adattípusúnak kell lennie.

2. Hangerővezérlés. Ha a hangerő meghalad egy előre meghatározott maximális értéket, egy figyelmeztető LED-et kell bekapcsolni. Emellett egy figyelmeztető üzenetet kell küldeni a hívó programnak.
3. A magnónak két modellje van, két különböző hangerő korláttal.

Amint a feladat megoldásából is kitűnik, az ST kód szintaktikailag és írási stílusában nagyon nagy hasonlóságot mutat a PASCAL nyelvel. Jelen jegyzet nem tárgyalja teljességében az ST nyelvet, az érdeklődő olvasó az irodalomban [4] megtalálhatja a nyelv részletes leírását.

5.4. Funkcióblokkos programozási nyelv (FBD)

A Funkció Blokk Diagram (FBD) eredetileg az analóg számítógépes modellezés területéről származik, ahol a rendszerek, elemzés és szintézis céljából blokkok segítségével került modellezésre. Időközben azonban az ipari vezérlők programozásának területén is széles körben elterjedt.

Az alábbi bekezdésben megadottak egyaránt vonatkoznak az FBD és a létradiagramos (LAD) programozási nyelvekre. A LAD vagy az FBD nyelven grafikus formában megadott POU is ugyanolyan részekből épül fel mint a szöveges nyelvek esetén:

- A POU kezdeti és bezáró része,
- A deklarációs rész,
- A kód rész.

A deklarációs rész lehet grafikus vagy szöveges. A legtöbb programozási rendszer támogat legalább egy szöveges nyelvet. A kód rész hálózatokra bontható. A hálózatok segítségével jól strukturálható a POU végrehajtásának menete.

A hálózat az alábbi részekből épül fel:

- Hálózati címke,
- Hálózati megjegyzés,
- Hálózati grafika.

A hálózati címke a hálózatok összekötését teszi lehetővé. Minden olyan hálózat, melybe egy másik hálózatból történő ugrással lépünk be egy a felhasználó által definiált alfanumerikus azonosítóval, vagy egy előjel nélküli decimális egésszel kezdődik. Ezeket az azonosítókat nevezzük hálózati címkének.

A programozási rendszerek rendszerint megszámozzák az egymást követő hálózatokat. Az egymást követő hálózatok számozása automatikusan frissül, ha egy új hálózat kerül beszúrásra. Ez egyszerűbbé és gyorsabbá teszi a hálózatok keresését (például a „GoTo network", vagy a hibás sor pozíciójának meghatározása esetén).

Az IEC 61131-3 szabvány csak a hálózati ugrás címkét definiálja. Ez a címke egy POU-n belüli lokális azonosító.

A legtöbb programozási rendszerben a hálózati címke és a hálózati grafika között lehetőség van megjegyzések elhelyezésére, ahogy azt a szöveges nyelvek esetében is láthattuk a (*…*) karakterek közötti szöveg megjegyzésnek tekintett.

Az IEC 61131-3 nem definiálja a megjegyzés fogalmát a grafikus nyelvek esetében, így nincs megkötés annak elhelyezésére vagy formátumára.

A hálózati grafika grafikus objektumokból épül fel, melyek felbonthatóak egyszerű grafikus elemekre, csatlakozásokra és összekötő vezetékekre, huzalokra. Az egyes grafikus elemek csatlakozásaikon keresztül tartják a kapcsolatot a külvilággal, rajtuk keresztül jutnak el az adatok a blokkokban meghatározott feldolgozásra. A feldolgozott adatok pedig a grafikus elemek kimeneti paramétereiben tárolódnak, készen a következő elem felé történő továbbításra.

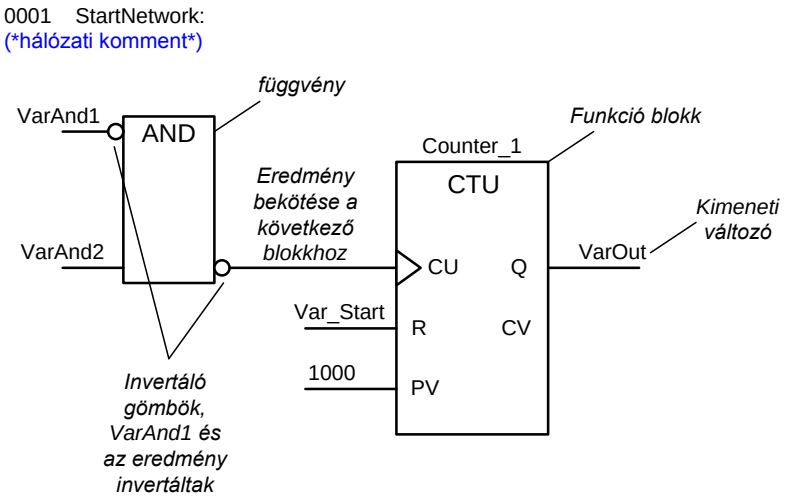
Az információáramlás irányának reprezentálásra az elemek közé vonalakat húzunk, melyek keresztezhetik egymást. A szabvány kétfajta keresztező vonalat definiál. Az egyik a keresztező vonal összeköttetés nélkül (vonalak melyek függetlenek egymástó), a másik pedig a

keresztező vonalak, melyek csomópontot képezve összekapcsolódnak vagy elágaznak (az információáramlás összekapcsolódik, vagy elágazik több elem felé).

Bizonyos programozási nyelvek korlátozzák a hálózat méretét, és nem lehetséges a horizontális navigálás. Ezeknél a rendszereknél a konnektor nevű konstrukció használható. A konnektor nem vezérlő vagy adat folyam elem, hanem az új sor grafikus megfelelője. Nagy hálózatok kialakításához a képernyő jobb oldalára egy névvel ellátott vonalat rajzolhatunk. Ez a vonal aztán a következő sorban a képernyő bal oldalán is megjelenik, jelezve a hálózat folytatódását. Ezt a névvel ellátott vonalat nevezzük konnektornak. A konnektorok nevei az adott POU-n belül lokális nevek. Ugyanabban a POU-ban egy konnektor nevet nem lehet hálózati címként vagy változó névként felhasználni. A konnektorok definiálhatóak a felhasználó által vagy a programozási rendszer által automatikusan. Bizonyos programozási rendszerekben ezeket a konnektorokat nem kell alkalmazni, mert nincs korlátozva a hálózat mérete. Minden egyes hálózat elem mely direkt vagy indirekt módon a konnektorok segítségével van összekötve ugyanahhoz a hálózathoz tartozik.

5.4.1. Hálózati architektúra az FBD nyelvben

Az FBD hálózat grafikus elemei műveletvégző elemekből és az azokat összekötő vízszintes és függőleges vezetékekből állnak. A felhasználható eszköztár tartalmaz úgynevezett forrás blokkokat, melyek csak kimenettel rendelkeznek, állandó vagy változó értékek forrásaként szerepelhetnek. A bemenetek és kimenetek maradhatnak nyitottak is, azaz előfordulhat, hogy nincsenek sehova bekötve.



5.14. példa. – : FBD hálózat elemei.

A 5.14. példában található hálózatban először meghatározzuk a VarAnd1 és a VarAnd2 azonosítójú változó közötti ÉS kapcsolatot majd az így kapott eredmény inverze a Counter1 CU nevű számláló bemenetére vezetjük (a Counter1 a CTU funkció blokk egy példánya). A Counter1.Q kimeneti paraméter a VarOut változóban kerül eltárolásra. A negált Boole bemeneteket vagy kimeneteket a „o” jelöljük. A funkcióblokk él vezérelt bemeneteit a „>” (felfutó él) és a „<” (lefutó él) jelekkel adhatjuk meg. Az IEC 61131-3 szabvány nem definiálja, hogy egy funkció blokk kimenete a blokk alján vagy tetején legyen elhelyezve. Az is implementációfüggő, hogy a változó nevek mindig az oldal bal oldali szélén vagy a blokk mellett, esetleg a csatlakozó vonalak felett van feltüntetve. A függvények és összeköttetések grafikus kombinációjával egyértelműen megadhatóak a logikai vagy aritmetikai kapcsolatok.

Az FBD-ben az alábbi grafikus objektumok fordulhatnak elő:

- Összeköttetések;
- Végrehajtást vezérlő grafikus elemek (mint a JMP) ;

- Függvény vagy funkció blokk hívását végző grafikus elemek (szabványos vagy a felhasználó által definiáltak) ;
- Konnektorok.

Az FBD-ben nincsen más grafikus elem, mint a LAD esetében például a tekercs. A következő bekezdésekben ezeket a grafikus elemeket fogjuk összefoglalni.

Az FBD-ben vízszintes és függőleges összekötő vezetékeket alkalmazhatunk. Egy vezeték több összeköttetéshez is tartozhat. Nem lehet egynél több kimenetet egy vagy több bementtel összekapcsolni. Ez inkonzisztenciát okoz, hisz nem lehet egyértelműen eldönteni, hogy melyik kimenet melyik bemenethez kapcsolódik. Ez csak a LAD esetében lehetséges. Az összeköttetéseken keresztül az elemi vagy származtatott adattípusok továbbíthatóak. Fontos, hogy a kimenti paraméter adattípusa, megegyezzen a bemeneti paraméter adattípusával, az automatikus típuskonverzió nem lehetséges.

A programban lefutása befolyásolható. Az FBD lehetőséget ad a POU-ból való kilépésre, valamint a címkére való ugrás lehetőséget biztosít a hálózatok feldolgozási sorrendjének megváltoztatására.

Grafikus objektum	Név	Magyarázat
1 —|RETURN|—	Feltétel nélküli visszatérés	A szabvány nem definiáls semmilyen speciális grafikus elemet. A felhasználó a hívó POU-hoz történő feltétel nélküli visszatérést egy IGAZ bemeneti paraméterrel rendelkező feltételes ugrással adhat meg. A vezérlés a hívó POU-hoz a POU végének elérésekor is visszaadódik
—|RETURN|—	Feltételes visszatérés	A bal oldali csatlakozás esetén ha a csatlakozáshoz kötött érték igaz akkor visszatérünk a hívó POU-hoz, ha pedig hamis, akkor marad a vezérlés a hívott POU-nál
1 —>> cím	Feltétel nélküli ugrás	Ugrás a cím címkével állatott hálózathoz
—>> cím	Feltételes ugrás	A bal oldali csatlakozás esetén ha a csatlakozáshoz kötött érték igaz akkor ugrunk a cím cimkével ellátott hálózathoz, egyébként nincs ugrás

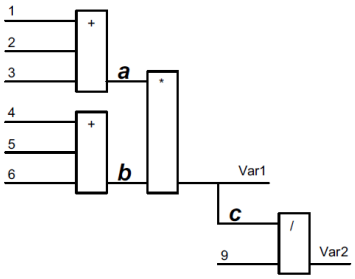
5.8. Táblázat – : Végrehajtást meghatározó grafikus primitívek az FBD-ben.

A funkcióblokkok (FB) és függvények hívásának grafikus reprezentációja hasonló. Az FB-knek és a függvényeknek számos be és kimeneti paramétere lehet. A formális paramétereket a blokkon belülre kell feltüntetni, azonban a túlterhelt függvényeknek nem lehetnek formális paraméter nevei. Az aktuális paraméter egy változó vagy konstans melyet a formális paraméter mellé a blokkon kívülre kell feltüntetni. Az információ továbbítás másik módja, hogy egyik blokk kimeneti paraméterét összekötjük egy másik blokk bemeneti paraméterével. Az FBD-ben nem kötelező hogy egy FB-nek legyen legalább egy Boole bemeneti és kimeneti paramétere. Ez a követelmény azonban a LAD esetében megvan. A függvény híváskor az EN/ENO be és kimeneti paraméterek használatára lehetőség van, de nem kötelező. Számos jelenlegi programozási rendszer nem tartalmazza ezt a tulajdonságot. Egy függvény visszatérési értéke a függvény nevéhez hozzárendelt érték segítségével adható meg. Ha a végrehajtás során több hozzárendelés is történik, akkor az utolsó hozzárendelés során megadott érték lesz a függvény visszatérési értéke. A függvény blokkjában a visszatérési értéket a blokk jobb oldalán kell feltüntetni formális paraméter nélkül. A további kimeneti paramétereket a könnyebb olvashatóság érdekében pedig formális paraméterekkel jelöljük. A VAR_IN_OUT deklaráció esetében egy vízszintes vezeték köti össze a formális be és kimeneti paramétereket.

5.4.2. A hálózat kiértékelése

A POU hálózatról-hálózatra fentről lefelé kerül kiértékelésre. Amennyiben szükséges megváltoztatni a sorrendet, akkor vezérlő utasításokkal ez megtehető. A kiértékelési sorrendben is vannak eltérések, hisz bizonyos rendszerekben az összes hálózat egy időben, vagy a számokkal meghatározott futtatási sorrendben történik. IEC 61131-3 a programozási rendszer gyártójának szabad kezet ad a hálózatok kiértékelésének sorrendjét illetően. A PLC-nek egy egyszerű hálózat kiértékelésekor az alábbi szabályokat kell figyelembe vennie:

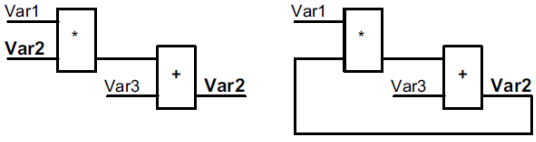
- Egy hálózati elem futtatása előtt az elem összes bemenetét ki kell értékelni;
- Egy hálózati elem kiértékelése addig tart, amíg az összes kimenete ki nem értékelődik;
- Egy hálózat kiértékelés addig nem fejeződik be amíg az összes elemének összes kimenete ki nem értékelődik.



5.15. példa. – : Értékmeghatározás egy FBD hálózatban. Az a, b és c értékek közbülső értékek.

Az 5.15. példában a kiértékelési sorrend a fenti szabályoknak megfelelően történik, azaz az osztás bemenetének (c vezeték) stabilnak kell lennie, mielőtt a Var2 kiszámítása megtörténhet. Ennek érdekében elsőként a szorzás függvény bemeneteit kell kiértékelni. A sorrend: az első és második összeadás kimeneteinek kiszámítása. Az összeadások eredménye a 6 (a érték) és 15 (b érték) érvényes bemeneti paraméterként, így már felhasználható a szorzásnál. A szorzást végző blokk eredménye, mint stabil bemenet, első operandusként felhasználható az osztás végző blokknál. Az osztó egy állandó (9), értéke a blokk második operandusa. Az osztás elvégzése után kapott 10-es eredmény a Var2 változóban kerül eltárolásra. A Var1 és Var2 értékének eltárolási sorrendje az IEC 61131-3 szabványban nincs meghatározva. A vezetékek összekapcsolhatnak bármilyen érvényes adat típust, mint az integer, a sztring mindaddig, amíg a vezetékek mindkét végén ugyanaz az adattípus van. Az ellenőrző folyamat megegyezik az IL nyelv esetében megismert CR regiszternél alkalmazott típusellenőrzéssel.

Az FBD-ben lehetőség van arra, hogy egy kimeneti paraméter értékét visszavezessük ugyanazon hálózat bemeneti paramétereként. A hurkot megvalósító vezetékeket visszacsatoló vezetékeknek, a hozzájuk tartozó értékeket pedig visszacsatolt változóknak nevezzük. Ismeretes, hogy a PLC program ciklikusan hajtódik végre. Az ilyen bemeneti változó a legelső kiértékelésekor a típushoz tartozó alapértelmezett értéket használjuk fel. Az első ciklus után pedig az értéke a hozzá tartozó kimenet előző ciklusbeli értéke lesz.



5.16. példa. – : Hálózat a Var2 változó két ekvivalens visszacsatolásával. A baloldalon implicit megvalósítás, a jobboldalon explicit megvalósítás látható.

5.5. Létradiagramos programozási nyelv (LD)

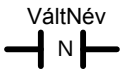
A kombinációs és szekvenciális hálózatok jelfogós megvalósításának dokumentálására alkalmazható az áramutas ábrázolás. Az áramutas ábrákat ismerő szakemberek könnyen átképezhetők a létradiagramos programozási nyelv a LAD (Ladder Diagram vagy LD) alkalmazására. A programnyelv kifejlesztését az is indokolta, hogy a logikai kapcsolások jelfogós megvalósítását az újabb technikai megoldásokban PLC váltja fel. Az FBD-hez hasonlóan az LD is hálózatot használ. A kiértékelés mindig fentről lefele egymás után sorrendben történik, amennyiben a elhasználó máshogy nem rendelkezik. Az áramutas ábrázolás esetében, csakúgy, mint az LD ábrázolásban feszültség mentes helyzetet tekintünk, az áram a huzalokban balról jobbra halad. Az LD kezdetben az alapvető Boole függvények és logikai jelek feldolgozására lett kifejlesztve. A hálózat két, a bal és a jobb oldalon elhelyezett úgynevezett tápvezeték között kerül kialakításra. A bal oldali tápvezeték a táp, és a rajta megjelenő érték a logikai 1-nek felel meg. A tápvezetékből folyó áram eléri az összes hozzá csatlakoztatott elemet. Az elemek a logikai állapotuknak megfelelően vagy engedik, vagy pedig nem engedik, hogy ez az áram elérje a hozzájuk csatlakoztatott következő elem bemenetét.

A LD hálózat az alábbi részekből épül fel:

- huzalozás,
- kontaktusok,
- tekercsek, logikai kimenetek,
- a végrehajtási sorrendet meghatározó grafikus elemek (ugrások),
- a függvényhívást és FB hívást végző grafikus elemek (szabványos vagy felhasználó által definiált POU),
- számlálók,
- időzítők,
- csatlakozók, ahogy azt az FBD esetében is láthattuk.

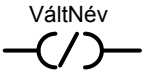
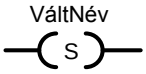
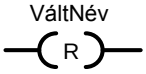
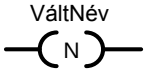
Az FBD-hez hasonlóan az LD hálózatban is alkalmazhatók vízszintes és függőleges huzalok az összeköttetések kialakításához. A programnyelv ismeri a csomópontokat és a keresztező vezetékeket.

A kontaktus, érintkező jellemzője, hogy nyitott állapotban nem engedi át a jelet, zárt állapotban pedig igen. A kontaktushoz tartozó változó nevét a kontaktus grafikus szimbóluma fölé kell írni. Minden változó adattípusa Boole. Ebből kifolyólag a hozzájuk tartozó összeköttetések csak Boole értékeket továbbíthatnak. Két típust különböztetünk meg, az egyik normális, amely alapállapotában nyitott (Normally Open, NO), a másik amely alapállapotában zárt (Normally Closed, NC).

Grafikus objektum	Név	Magyarázat
VáltNév 	Alaphelyzetben nyitott kontaktus	Jobb oldal := Bal oldal ÉS VáltNév (Átmásolja a bal oldal állapotát a jobb oldara ha a változó értéke IGAZ)
VáltNév 	Alaphelyzetben zárt kontaktus	Jobb oldal := Bal oldal ÉS NEM VáltNév (Átmásolja a bal oldal állapotát a jobb oldalra, ha a változó értéke HAMIS)
VáltNév 	Felfutó él ézékelő kontaktus	Jobb oldal := IGAZ ha a változó értéke az előző kiértékeléskor HAMIS **és** a változó aktuális értéke IGAZ **és** a bal oldal státusza IGAZ, egyébként HAMIS. (Átmásolja a bal oldal értékét a jobb oldalra , ha a változónak HAMIS -> IGAZ átmenete van
VáltNév 	Lefutó él ézékelő kontaktus	Jobb oldal := IGAZ ha a változó értéke az előző kiértékeléskor IGAZ **és** a változó aktuális értéke HAMIS **és** a bal oldal státusza IGAZ, egyébként HAMIS. (Átmásolja a bal oldal értékét a jobb oldalra , ha a változónak IGAZ -> HAMIS átmenete van

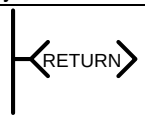
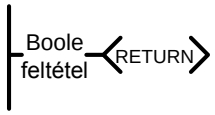
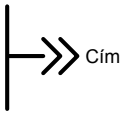
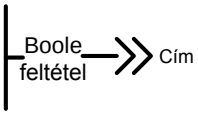
5.9. Táblázat – : Az érintkezők, kontaktusok IEC szabványos jelölése.

A tekercsek azok az elemek, melyek segítségével a változókhoz értékeket rendelhetünk. Az áramutas ábrához hasonlóan az LD esetében is az alapelv az, hogy logikai igaz érték esetén áram halad át a tekercsen és a hozzá tartozó jelfogó behúz, megmozgatva ezzel a jelfogó érintkezőit. Az LD programozási nyelv, az alapkapcsolás mellett lehetőséget ad összetettebb viselkedéssel bíró tekercsek alkalmazására is.

Grafikus objektum	Név	Magyarázat
	Tekrecs	Változó értéke := Bal oldal értéke (Átmásolja a bal oldal értékét a változóba)
	Negált tekercs	Változó értéke := NEM bal oldal értéke (Átmásolja a bal oldal negált értékét a változóba)
	Set tekrecs	Változó értéke := IGAZ ha a bal oldal értéke IGAZ egyébként nem változik a változó értéke
	Reset tekercs	Változó értéke := HAMIS ha a bal oldal értéke IGAZ egyébként nem változik a változó értéke
	Felfutó él érzékelő tekercs	Változó értéke := IGAZ ha a bal oldal értéke az előző kiértékeléskor HAMIS **és** a bal oldal aktuális értéke IGAZ, egyébként nem változik a váltzó értéke. (Átmásolja a bal oldal értékét a változóba , ha a változónak IGAZ -> HAMIS átmenete van)
	Lefutó él érzékelő tekercs	Változó értéke := IGAZ ha a bal oldal értéke az előző kiértékeléskor IGAZ **és** a bal oldal aktuális értéke HAMIS, egyébként nem változik a váltzó értéke. (Átmásolja a bal oldal értékét a változóba , ha a változónak HAMIS -> IGAZ átmenete van)

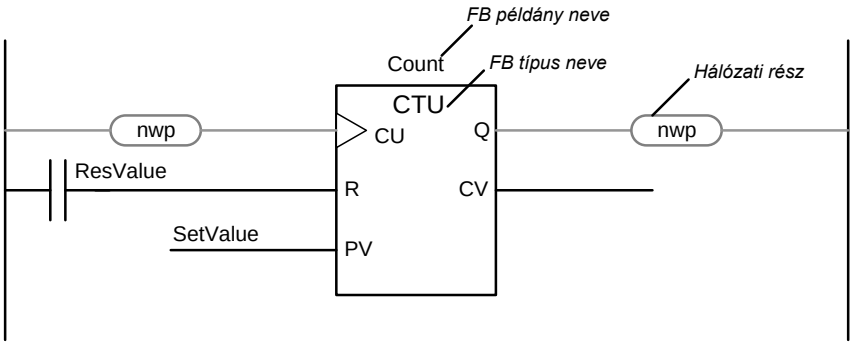
5.10. Táblázat – : A tekercsek IEC szerinti jelölése.

Annak érdekében, hogy megváltoztassuk a POU-k és a hálózataik normális végrehajtási sorrendjét a LAD nyelvben két lehetőség kínálkozik: az egyik a POU-ból történő kilépés, a másik pedig az ugrás egy meghatározott hálózathoz. Az 5.11. táblázat tartalmazza a műveleteket meghatározó jelöléseket.

Grafikus objektum	Név	Magyarázat
	Feltétel nélküli visszatérés	A POU elhagyása és visszatérés a hívó POU-hoz. A vezérlés automatikusan visszatér a havó programohoz ha a POU végére ér a végrehajtás
	Feltételes visszatérés	Ha a bal oldal értéke 1, akkor a POU leáll és visszatér a vezérlés a hívó POU-hoz, egyégként kimarad a lépés
	Feltétel nélküli ugrás	Ugrás a Cím címkével ellátott hálózatra
	Feltételes ugrás	Ha a bal oldal értéke 1, akkor ugrás a Cím címkével ellátott hálózatra, egyébként kimarad ez a lépés

5.11. Táblázat – : A végrehajtás vezérlések IEC szerinti grafikus szimbólumai.

Az LD nyelvben kétfajta POU hívása megengedett, melyek a függvény és az FB. A hívott POU-t egy négyszögletes blokkban adjuk meg. Az FB-knek és a függvényeknek számos be és kimeneti paramétere lehet. A függvényeknek egy vagy több visszatérési értékük lehet. A formális be és kimeneti paraméterek neveit a blokkon belül tüntetjük fel. A túlterhelt függvények esetében nincsenek formális paraméterek. A megfelelő aktuális paramétert a blokkon kívül a formális paraméterhez csatlakozó vezeték fölé írva adhatjuk meg. Itt is lehetséges az FBD nyelvhez hasonlóan, hogy a blokkok be és kimeneti paramétereit közvetlenül egymáshoz csatlakoztathassuk. A negálás és az élvezéreltség jelzése az FBD esetében bemutatottakhoz hasonlóan történik. Az FB paraméterei tetszőleges adattípusúak lehetnek, de mindenképpen kell legalább egy be és kimenet, melynek a típusa Boole, melyek direkt vagy indirekt módon csatlakoznak a bal és jobboldali tápvezetékhez. A szabványos FB-k esetében a Q kimenetet lehet erre a célra felhasználni.

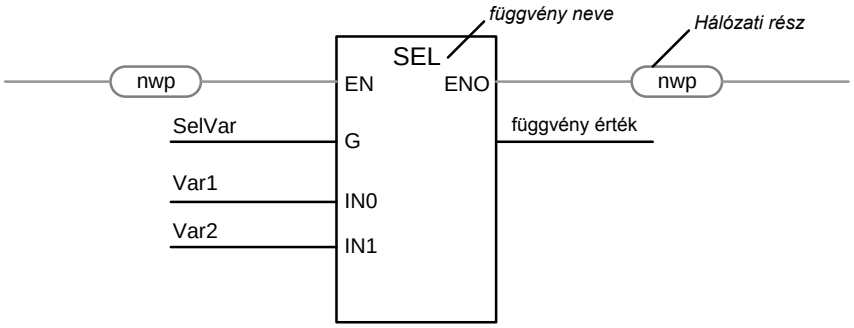


5.17. példa. – : A szabványos CTU FB hívása a LD hálózatban.

A SetValue változó a PV paraméterhez van rendelve. A ResVar értéke az R bemenetbe másolódik. A CV kimenet nincs bekötve. A bal oldali tetszőleges hálózati rész (nwp) Boole eredménye a CTU FB CU vezérlő bementére van csatlakoztatva. A kimeneti Q paraméter értéke a jobb oldali részhálózat (nwp) kiindulási értéke. Az 5.17 példában a bal oldali részhálózat nem

kötelező, hisz már van egy Boole csatlakozás a bal oldali tápvezetékhez (az R bemenet a ResVar-on keresztül). A jobb oldali részhálózat viszont mindenképpen szükséges, mert a CV egy integer típusú paraméter, mely nem csatlakoztatható a jobb oldali tápvezetékhez.

A függvények estében szükség van a speciális EN/ENO bemeneti és kimeneti paraméterekre, melyek a függvény végrehajtását vezérlik. Ha az EN=FALSE akkor a függvény nem kerül végrehajtásra és az ENO kimenete is FALSE lesz. Az ENO kimenet, mint hiba indikátor is felhasználható. Az EN és ENO be és kimeneteket direkt vagy indirekt módon, mindenképpen csatlakoztatni kell a bal és jobb oldali tápvezetékhez. Az ENO=FALSE esetében a függvény egyéb kimeneteinek értéke gyártó specifikus. A függvény további paramétereinek beállítása egy változó vagy hálózat által szintén implementációfüggő.

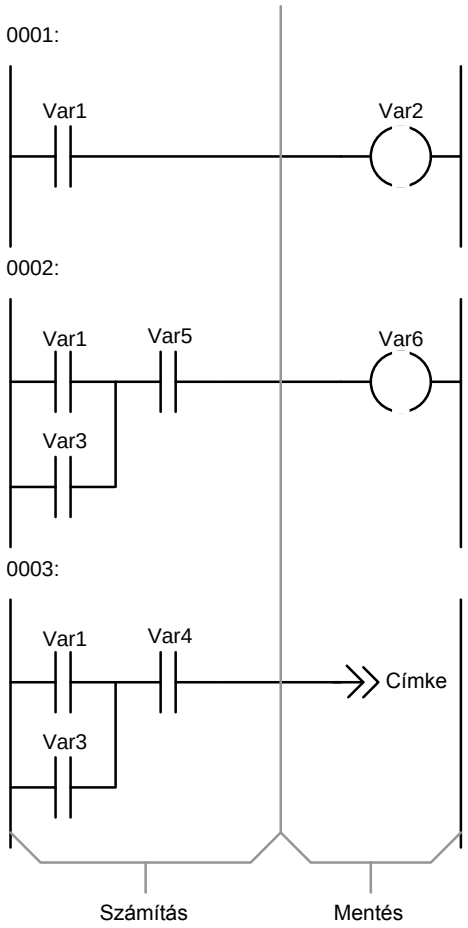


5.18. példa. – : A SEL szabványos függvény meghívása a LD nyelvben.

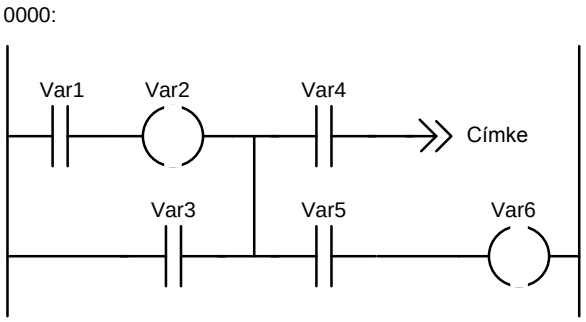
A felhasználó által definiált függvények esetében a függvény értéke egy változó értékének függvény nevéhez történő hozzárendelésével adható meg.

Az LD hálózat végrehajtási sorrendje megegyezik az FBD hálózat végrehajtási sorrendjével. Ezen a sorrenden a felhasználó az ugrás utasítások alkalmazásával változtathat. Tehát a hálózati eredmény kiértékelése a hálózat grafikus elrendezésétől és az összeköttetésektől függ. Egy egyszerű kombinációs hálózati alkalmazás esetén az elemek lehetnek sorosan és párhuzamosan kapcsolva. Soros kapcsolás esetén logikai ÉS kapcsolat valósul meg, párhuzamos kapcsolás esetén pedig logikai VAGY kapcsolat. A hálózatban az IEC 61131-3 szabvány engedélyezi egy érték hozzárendelés utáni további feldolgozását, ahogy az az 5.20. példában látható. Azonban a legtöbb programozási rendszer az alábbi mintát követi:

- Bal oldali rész: Logikai konstrukció változókkal, konstansokkal, és függvényekkel, melyek egy értéket számítanak ki (számítás) ;
- Jobb oldali rész: Az eredmények eltárolása (mentés).



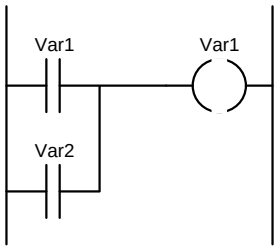
5.19. példa. – : LD hálózat struktúrája.



5.20. példa. – : Nem szokásos hálózati struktúra, mely funkcionalitása megegyezik az 5.19. példáéval.

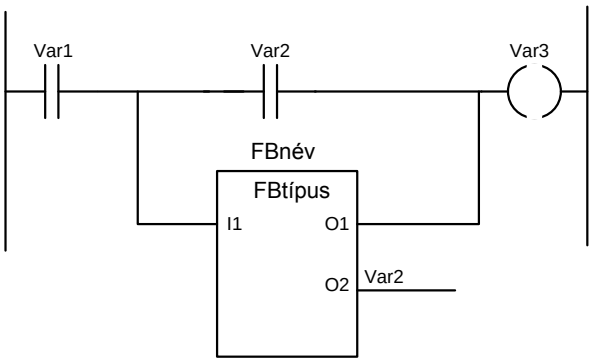
Az 5.20. példában megadott hálózat sokkal tömörebb, mint az 5.19. példában látható három hálózat, azonban az 5.19. példában megadott hálózat jobb átláthatóságot biztosít. Az LD a logikai kifejezések programozásának egyszerű módja. Amennyiben a konkrét feladat megoldása érdekében más adattípusok kiértékelésére, például Real vagy Integer is szükség mutatkozik, akkor előnyösebb, ha más programozási nyelvet használunk. Bizonyos fejlesztési rendszerek lehetővé teszik a kombinált programozást, legtöbb ilyen esetben a nem logikai számítások FBD hálózat formájában adhatók meg, míg a logikai kapcsolások LD-ben.

Ha egy változót arra használunk, hogy a számítás eredményét eltároljuk, majd ugyanezt az eredményt a számítás következő ciklusában ugyanazon hálózat bemeneteként használjuk fel, akkor a változót visszacsatolt változónak nevezzük.



5.21. példa. – : A Var1 visszacsatolt változó a LD nyelvben.

Az FBD-vel ellentétben az LD-ben csak implicit visszacsatolás megvalósítása lehetséges. Ennek következtében jobbról balra mutató explicit kapcsolatok nem adhatóak meg. Amennyiben az FB és a visszacsatolt változó ugyanazon hálózat részei, akkor ez különböző viselkedést eredményezhet különböző PLC rendszerek esetében:



5.22. példa. – : Hálózat mely FB-t és visszacsatolt változót (Var2) is tartalmaz. A végeredmény rendszerfüggő.

Az 5.22. példa szemlélteti a dilemmát, nem egyértelmű, hogy az FBName.O2 paraméter értékét a Var2-höz az előző vagy az aktuális ciklusban rendeljük hozzá. Az IEC 61131-3 nem tárgyalja ezeket az eseteket. A kétértelműségek kizárása érdekében, a hasonló esetekben hasznos az FB-t egy külön hálózatba tenni.

5.6. Sorrendi folyamatábra (SFC)

A PLC programok gyakran összetettek, a programozóknak minden esetben biztosítani kell a hibátlan, megbízható működést. Az IEC 61131-3 szabványban meghatározott sorrendi folyamatábra (Sequential Function Chart SFC) a modularitás biztosítása mellett lehetővé teszi áttekinthető, könnyen tesztelhető programok készítését. Alkalmazásával a komplex programok felbonthatók kisebb, könnyebben kezelhető részekre. Az SFC segítségével mind soros, mind pedig párhuzamos folyamatok irányítása megtervezhető, leírható. Az SFC nyelven megírt program egységeinek futását egyrészt statikus (program által definiált), másrészt pedig dinamikus (az I/O-k viselkedése) feltételek határozzák meg. Az SFC-ben a programozási egységek leírása valamelyik a korábbiakban már bemutatott IEC 61131-3 szabványos nyelven történhet. Az SFC mögött álló módszertan alapja a jól ismert Petri-hálók vagy a sorrendi módszertan.

Az automatizálás területén hamar igény mutatkozott egy, az állapotokat és az azok közötti átmeneti feltételeket leíró nyelvre. A franciák által kifejlesztett, szabványosított nyelv a Grafcet

volt. Ez a nyelv szolgált az IEC 848-as nemzetközi szabvány alapjaként. Az IEC 61131-3-ban definiált SFC nyelv integráltan tartalmazza mind a két módszertant.

Az egymás után következő lépéseket tartalmazó folyamatok leírására különösen alkalmas az SFC nyelvi struktúra. Vegyünk példaként egy kémiai folyamatot, ahol folyadékokat kell egy tartályba tölteni egészen addig, amíg a szintérzékelő nem jelzi, hogy a tartály tele van (első lépés). A folyadékadagoló szelepek elzárását követően elindul a keverőlapát (második lépés). Amint a keverésnek vége, a keverő motor leáll és a keverék kiürítésre kerül (harmadik lépés), majd az egész folyamat újraindul az elejétől.

Egy másik példa az egymást követő lépéseket tartalmazó folyamatokra az automata mosógép. A mosógép is lépésről lépésre hajtja végre a műveleteket, úgymint az előmosás, fő mosás, öblítés, és így tovább. Minden egyes lépés mögött egy-egy meghatározott algoritmus van. Amint egy lépés véget ér (melyet egy időzítő vagy érzékelő jelez), elindul a következő lépés, melyben más soros vagy párhuzamos vezérlési egységek aktiválódnak.

Az SFC elsődlegesen egy grafikus nyelv, de szöveges leírására is lehetőséget ad. Erre azért van szükség, hogy egyszerűbb legyen az információ cseréje a különböző programozási rendszerek között. Egy POU SFC nyelven történő megadására a grafikus verzió a jobb választás, mert ebben az esetben sokkal tisztábban lehet vizualizálni az egyes lépések közötti függőségeket és függetlenségeket.

Lehetséges, de nem szükséges, hogy az SFC segítségével strukturáljunk egy programot vagy FB-t. A függvények SFC-vel történő strukturálása nem lehetséges, mivel egy SFC alapú POU esetében statikus információkra van szükség, például védett állapotú információkra, melyek alkalmazása nem engedélyezett a függvényekben. Ha a POU SFC nyelven íródik, akkor teljes POU-t SFC-ben kell megadni.

A megjegyzések megadásának szabályai az SFC nyelvben megegyezik, a többi IEC 61131-3 szabványos nyelvvel. Szöveges formában a zárójel-csillag kombinációt (*….*) lehet alkalmazni mindenütt, ahol üres karaktert is lehet. A grafikus megadás esetében a megjegyzések írásához nincsenek szabályok definiálva, így ez implementációfüggő.

A többi grafikus nyelvhez hasonlóan az SFC nyelv esetében is a strukturálás alapja a hálózat. Egy POU-ban egy vagy több hálózat lehet. A hálózat elemeit lépéseknek és átmeneteknek nevezzük.

Egy lépés lehet aktív vagy inaktív. Amíg egy lépés aktív addig a hozzá tartozó utasítások ciklikusan futnak. Egy lépés státuszának (aktív vagy inaktív) meghatározását az úgynevezett átmenetek végzik, mely a következő elem közvetlenül a lépés alatt. Egy átmenetet átmeneti feltételek megadásával programozhatunk, melyek lényegében Boole kifejezések. Amikor a kifejezés igaz, akkor az aktív lépés inaktíválódni fog. Az átmenet alatt lévő egy vagy több vonal meghatározza, hogy mely lépés vagy lépések aktiválódnak, amikor az éppen aktuális lépés inaktíválódik.

A 5.3. ábrán egy SFC hálózatot láthatunk, mely lépésekből (négyszögletes dobozok), átmenetekből (vonalak azonosítókkal) és link-ekből áll. A diagram jobb oldala a konstrukció műveleti sorrendjét írja le.

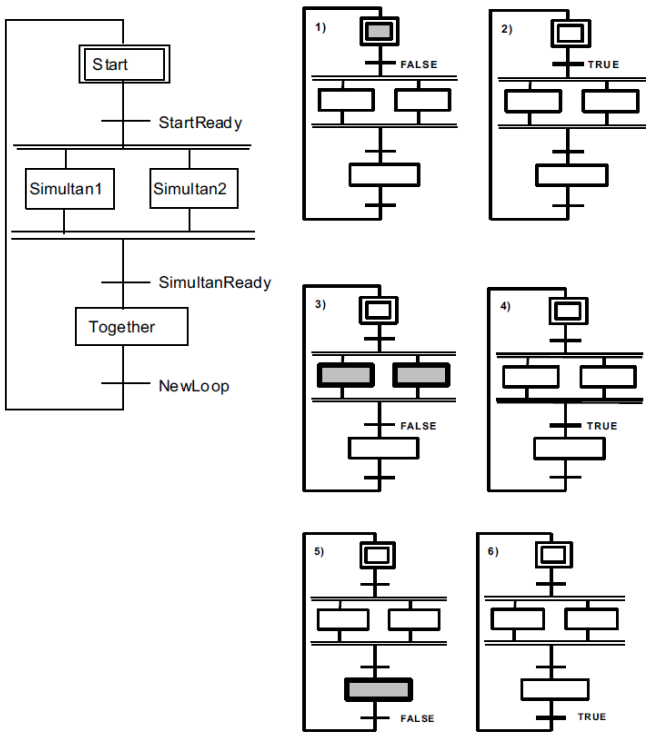
Amikor egy SFC-vel megadott POU-t meghívunk, akkor egy speciális lépés az úgynevezett kezdeti lépés aktiválódik. Ebben a példában a Start a kezdeti lépés. Az ehhez a lépéshez tartozó utasítások futnak le először. Amikor a StartReady Boole változó értéke FALSE-ról TRUE-ra változik a lépéshez tartozó utasítások végrehajtása vagy valamilyen I/O változás miatt, a Start lépés inaktíválódik. Ugyanebben az időben a következő Simultan1 és a Simultan2 lépések aktiválódnak, melyek az aktív átmenethez kapcsolódnak.

Minden ezekhez a lépésekhez tartozó utasítás végrehajtása elkezdődik. Ezután a következő átmenet értéke a SimultanReady kerül kiértékelésre. A Simultan lépések egészen addig futnak, amíg a SimultanReady értéke meg nem változik FALSE-ról TRUE-ra. Amikor ez bekövetkezik, akkor a két Simultan lépés inaktíválódik, és a Together lépés válik aktívvá. Ezután a

NewLoop nevű átment értéke vezérli a hálózat állapotát, és a Together lépés addig marad aktív, amíg a NewLoop változó értéke TRUE nem lesz.

Egy átmenet triggerelésével az aktív attribútum átkerül az aktív lépéstől az őt követő lépéshez vagy lépésekhez. Ebből következően ez az attribútum a lépések között vándorol. Ez az attribútum nem veszhet el, nem sokszorozódhat, vagy cirkulálhat a hálózatban.

Azokat a grafikus elemeket, melyeken ez az attribútum végighalad, a következő bekezdésben nézzük meg.



5.3. ábra – : A bal oldalon egy egyszerű SFC hálózat. A jobb oldalon a hálózat végrehajtása 1-6 lépéseken keresztül: A lépés (1) inaktív lesz, ha az alatta lévő átmenet TRUE lesz (2). Ez aktiválja a következő lépéseket (3) és így tovább. Amikor a végrehajtás eléri a hálózat végét, akkor a folyamat újra kezdődik a kezdeti lépéstől (1).

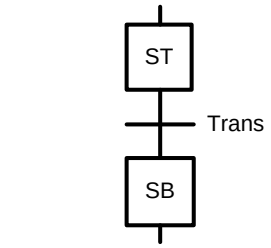
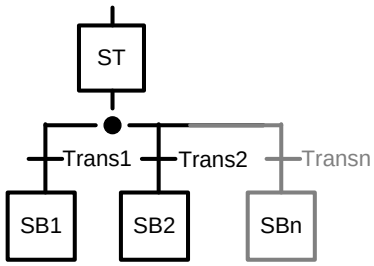
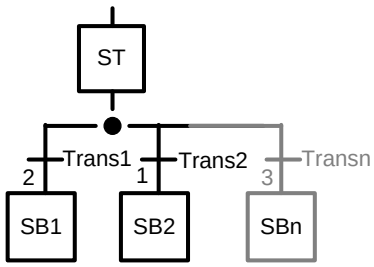
A lépéseknek és az átmeneteknek mindig váltakozniuk kell a hálózatban. Nem megengedett két azonos típusú elem összekapcsolása (lépés lépéssel vagy átmenet átmenettel). A átmenetnek lehet számos megelőző és követő lépése. A lépések és átmenetek egyszerű kombinációját szekvenciának nevezzük.

Egy lépés futtatása után lehetőség van több szekvencia közül egy kiválasztására, vagy több szekvencia egyidejű aktiválására. Ezen szekvenciákkal kapcsolatos szabályok az 5.12. táblázatban kerültek összefoglalásra.

Az 5.12.a. táblázatban megadott lehetséges SFC szekvenciák a következők:

- Egyszeres szekvencia: A lépés -> átmenet -> lépés sorozat váltakozása. Az ST deaktiválódik, ha a Trans értéke IGAZ. SB aktív lesz, ha a Trans IGAZ.
- Divergens útvonal: Pontosan egy szekvencia kiválasztása, a kiértékelés balról jobbra történik. Amíg ST aktív, az átmenetek balról jobbra kerülnek kiértékelésre. Az első IGAZ átmenet megszakítja az ST további futását és aktiválja a hozzárendelt allépést. Ez egy VAGY kapcsolásnak felel meg.
- Divergens útvonal a felhasználó által vezérelve: Ez lényegében megegyezik az előbbi divergens útvonallal, azzal a kivétellel, hogy itt a felhasználó precedenciát

határoz meg az átmenetek kiértékelésében. Ezt az átmenet mellé írt szám segítségével lehet megtenni, a kisebb szám jelenti a nagyobb prioritást.

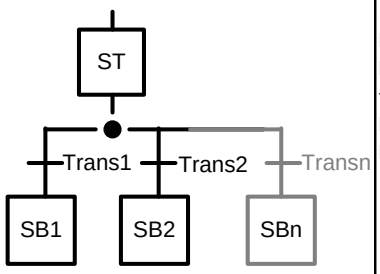
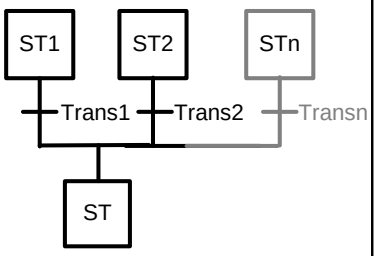
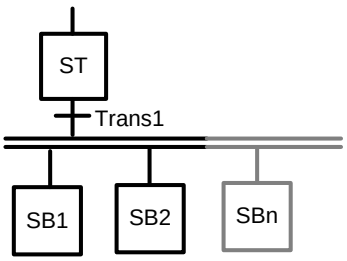
Grafikus objektum	Név és magyarázat
	**Egyszeres szekvencia:** A lépés -> átmenet -> lépés sorozat váltakozása. Az ST deaktiválódik, ha a Trans értéke IGAZ. SB aktív lesz, ha a Trans IGAZ
	**Divergens útvonal:** Pontosan egy szekvencia kiválasztása, a kiértékelés balról jobbra történik. Amíg ST aktív, az átmenetek balról jobbra kerülnek kiértékelésre. Az első IGAZ átmenet megszakítja az ST további futását és aktiválja a hozzárendelt allépést
	Divergens útvonal a felhasználó által vezérelve: Ez lényegében megegyezik az előbbi divergens útvonallal, azzal a kivétellel, hogy itt a felhasználó precedenciát határoz meg az átmenetek kiértékelésében. Ezt az átmenet mellé írt szám segítségével lehet megtenni, a kisebb szám jelenti a nagyobb prioritást.

5.12.a. Táblázat: Lehetséges SFC szekvenciák.

Az 5.12.b. táblázatban megadott lehetséges SFC szekvenciák a következők:

- Divergens útvonal a felhasználó által definiált prioritással: Ez lényegében megegyezik a divergens útvonallal, azzal a kivétellel, hogy itt az átmenetek precedencia nélkül értékelődnek ki. A felhasználónak kell arról gondoskodnia, hogy az átmenetek kizáróak legyenek, és minden esetben csak az egyik átmenet lehessen IGAZ;
- Szekvenciák konvergálása: A divergens útvonalak összekombinálódnak. Amikor valamelyik STn aktív, és a hozzá tartozó átmeneti feltétel IGAZ, akkor az STn inaktiválódik és az SB aktiválódik;

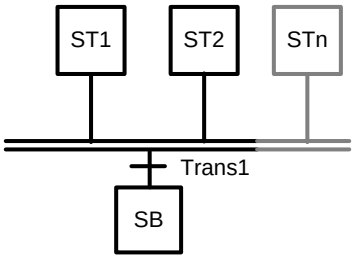
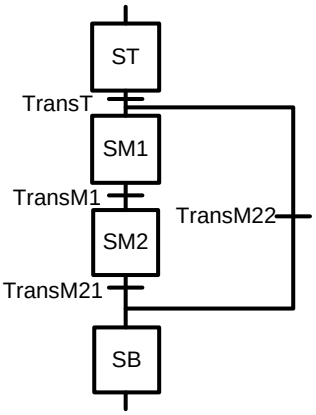
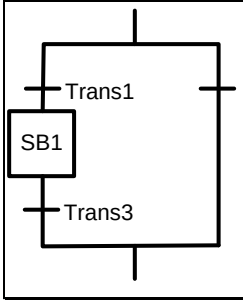
- Szimultán szekvenciák: Az összes csatlakoztatott lépés szimultán aktiválása. Az ST inaktiválódik, amikor a Trans IGAZ lesz, és az összes a Trans-hoz tartozó lépés aktiv lesz. Ez az újonnan aktiválódott szekvencia ezután párhuzamosan fut.

Grafikus objektum	Név és magyarázat
	Divergens útvonal a felhasználó által definiált prioritással: Ez lényegében megegyezik a divergens útvonallal, azzal a kivétellel, hogy itt az átmenekek precedencia nélkül értékelődnek ki. A felhasználónak kell arról gondoskodnia, hogy az átmenetek kizáróak legyenek, és minden esetben csak az egyik átmenet lehessen IGAZ
	**Szekvenciák konvergálása:** A divergens útvonalak összekombinálódnak. Amikor valamelyik STn aktív, és a hozzá tartozó átmeneti feltétel IGAZ, akkor az STn inaktiválódik és az SB aktiválódik
	**Szimultán szekvenciák:** Az összes csatlakoztatott lépés szimultán aktiválása. Az ST inaktiválódik, amikor a Trans IGAZ lesz, és az összes a Trans-hoz tartozó lépés aktiv lesz. Ez az újonnan aktiválódott szekvencia ezután párhuzamosan fut.

5.12.b. – : Táblázat: Lehetséges SFC szekvenciák.

Az 5.12.c. táblázatban megadott lehetséges SFC szekvenciák a következők:

- Szimultán szekvenciák konvergálása: A szimultán szekvenciákhoz tartozó különálló útvonalak újra egyesülnek. Amikor az összes STn aktív és a hozzájuk tartozó átmeneti feltételek IGAZ-ak, akkor az összes STn inaktiválódik, és az SB aktiválódik;
- Ciklikus szekvencia: Az átmenet visszatér egy megelőző lépéshez. A szekvancia kiválasztás megegyezik a divergens útvonalaknál leírtakkal;
- Kihagyás sekvencia: Ez mind a három divergens útvonal módban használható. Ez nem tartalmaz lépést, csak egy átmenetet: ezt hívják szekvencia kihagyásnak.

Grafikus objektum	Név és magyarázat
	**Szimultán szekvenciák konvergálása:** A szimultán szekvenciákhoz tartozó különálló útvonalak újra egyesülnek. Amikor az összes STn aktív és a hozzájuk tartozó átmeneti feltételek IGAZ-ak, akkor az összes STn inaktiválódik, és az SB aktiválódik
	**Ciklikus szekvencia:** Az átmenet visszatér egy megelőző lépéshez. A szekvancia kiválasztás megegyezik a divergens útvonalaknál leírtakkal.
	**Kihagyás sekvencia:** Ez mind a három divergens útvonal módban használható. Ez nem tartalmaz lépést, csak egy átmenetet: ezt hívják szekvencia kihagyásnak.

5.12.c. – : Táblázat: Lehetséges SFC szekvenciák.

Az IEC 61131-3 két hiba típust definiál, melyet el kell kerülni, amikor SFC nyelven programozunk. A nem biztonságos hálózat esetében lehetőség van a lépések kontrollálatlan és koordinálatlan aktiválására. Az elérhetetlen hálózat esetében pedig vannak olyan komponensek, amelyek sohasem lehetnek aktívak. Ideális esetben a programozási rendszer figyelmeztethet ezen, hibák kiküszöbölésére vagy a futó rendszer detektálhatja ezen hibák meglétét és jelezheti azokat.

6. Ipari hálózatok. A PLC-k hálózatba kapcsolása

A rendszertechnikában egyre nagyobb jelentősége van az egyes eszközök közötti kommunikációnak. A korszerű érzékelők, beavatkozók és vezérlők szinte kivétel nélkül rendelkeznek digitális kommunikációs interfész-el. Ezen interfészükön keresztül módosítható, frissíthető szoftverük, állíthatók paramétereik, lekérdezhetők állapotaik és folyamat adataik. Az informatikai erőforrások hálózatba kapcsolása minden esetben új dimenziót és problémákat nyit a megvalósítható lehetőségek, potenciális veszélyforrások, valamint a megbízhatóság terén. Nincs ez másképp az ipari hálózatok esetében sem. Az ipari fejlesztések igyekeznek: kihasználni a hálózatba kapcsolás nyújtotta lehetőségeket, a legkisebb mértékűre csökkenteni a veszélyeket és ezzel együtt növelni a megbízhatóságot. Az ipari hálózatok lehetőséget adnak: a hierarchikus, többszintű irányítási, felügyeleti rendszerek alkalmazására, ki/bemeneti szigetek távoli kialakítására, redundáns irányítás megvalósítására és erőforrások megosztására. A klasszikus irányítástechnikai megoldásokkal szemben a hálózatok alkalmazása számos előnyt jelent, melyek közül a legjelentősebbnek mondhatóak a: kisebb kábelezési költségek, a kisebb méretű kapcsolószekrények, a kisebb hagyományos technikai elem szükséglet, a kisebb ráfordítási és telepítési költség, a kisebb szervizköltségek, a rugalmas módosítási lehetőségek, valamint a nagyobb üzembiztonság.

Jelen fejezet nem terjed ki az adatátvitellel kapcsolatos főbb ismeretek bemutatására. Az irodalom [11] elég részletes betekintést nyújt, ebbe az ipari kommunikációs rendszerek megismeréséhez oly elengedhetetlen fontosságú témakörbe.

Az ipari hálózatok terén nagy hangsúlyt kap a szabványoknak való megfeleltetés, ami záloga a szélsőséges körülmények közötti hibátlan működésnek. Napjainkig számos, szabványosított protokoll látott napvilágot. Legtöbbjük az OSI (Open System Interconnection) hálózati modell fizikai, adatkapcsolati és alkalmazási szintjét valósítják meg. Az ipari automatizálás terén alkalmazott szabványok legtöbbje, az alkalmazási réteg fölé egy felhasználói réteget határoz meg, amely automatizálási funkciókat valósít meg. Néhány jelentős szabvány: a Foundation Fieldbus, a PROFIBUS, a PROFINET, a DeviceNet, a CAN és az IO-Link.

A Foundation Fieldbus (FF) nemzetközi szervezet vállalta föl, hogy kidolgozza a fildbusz szabványt és felügyeli a vele kapcsolatos tevékenységeket. A non-profit szervezet egy kissé módosította az IEC 61158-as szabványban levő: „A fildbus egy digitális, soros multidrop jellegű adatbusz az ipari irányítási és műszerezési feladatokhoz, mint például a távadók, beavatkozók és helyi vezérlők közötti kommunikáció biztonságos lebonyolításához.” meghatározást és a következőt javasolta: „A fildbusz digitális, kétirányú, multidrop jellegű kommunikációt jelent az intelligens mérő és irányító eszközök felé, amely LAN hálózatként szolgál a folyamatirányítás, a terepi I/O eszközök és a nagysebességű gyártásautomatizálási alkalmazások irányába.”. A konzorcium által adott meghatározás pontosabban fedi napjaink gyakorlatát.

Az IEC 61158-as szabványa 6 részben foglalkozik az ipari mérések és irányítások terén megvalósuló szabványos digitális kommunikációval. Az első rész áttekintést ad magáról a szabványról, a második a fizikai szinttel foglalkozik, a harmadikban kerülnek meghatározásra az adatkapcsolati szint szolgáltatásai, a negyedik írja le az adatkapcsolati protokollt, az ötödik foglalkozik az alkalmazási szint szolgáltatásaival, míg a hatodik határozza meg magát az alkalmazási protokollt.

Az FF H1 jelzésű kommunikáció elsősorban a folyamatirányításokban résztvevő terepi eszközök területén alkalmazható. A terepen meglévő csavart érpárokon keresztül 31.25 kbit/s–os digitális kommunikációt tesz lehetővé, hatékonyan kiváltva a hagyományos 4-20 mA–es analóg megoldásokat. Modern megoldások esetében optikai kábel is alkalmazható átviteli közegként. A megoldás alkalmazható robbanásveszélyes környezetben is.

Az FF High Speed Ethernet (HSE) elsősorban az irányítástechnikai gerincvezetékek kialakítására alkalmas. A 100 Mbit/s–os sebesség elsősorban PLC-k és folyamatirányítási számítógépek összekapcsolására alkalmas. A CSMA/CD közeghozzáférés és a redundanciára való törekvés hibatűrővé teszi a HSE hálózatot, így az felelősen alkalmazható a folyamatirányítás széles spektrumán.

A PROFIBUS vagyis a PROcess Field BUS egy gyártó-független ipari kommunikációs szabvány, melyet széleskörűen alkalmaznak az ipari folyamatok irányítása is a gyártásautomatizálás terén. Az IEC 61158 és az IEC 61784 szabványoknak köszönhetően illesztés nélkül kapcsolhatók hálózatba különböző gyártók eszközei.

Az architektúra fizikai szintje egyik megoldásaként RS485 vagy RS485-IS szabványt használ, amely viszonylag nagysebességű kommunikációt biztosítanak árnyékolt, csavart érpáron keresztül. Az RS485 nem alkalmazható robbanásveszélyes környezetben, míg az RS485-IS igen. A robbanásveszélyes környezetben is alkalmazható szabvány maximalizálja az csatornán folyó áramerősséget és a benne ható feszültséget. Az adatátviteli sebesség 9.6-Kbit/sec-tól 12000 Kbit/sec-ig terjed, az egy szegmensre csatlakoztatható eszközök száma 32. Egy másik megoldás szerint, az elektromágneses zavarok hatásának csökkentése és az áthidalni kívánt távolság növelése érdekében üvegszálas csatornán történik a kommunikáció. A harmadik lehetséges megoldása a fizikai szint megvalósításának a vezeték nélküli kapcsolat.

Az adatkapcsolati réteg mester-szolga alapú megoldást használ, ami elsősorban azt jelenti, hogy a szolga eszközök csak a mester megszólítására válaszolva kaphatnak időszeletet a buszon. A mester ciklikusan, egy lista alapján kérdezi le a szolgákat. Több mester csomópont esetében token-ring kapcsolat alakul ki a mesterek között, inicializáláskor kerül meghatározásra a mesterek száma, valamint minden mester esetében az azt megelőző és a következő mester címe. A szabvány foglalkozik a hibatűrő, redundáns megoldásokat tartalmazó topológiákkal is.

A szabvány több protokollt tartalmaz. A Profibus DP (Decentralized Periphery) elsősorban a PLC-k és a távoli perifériák közötti kommunikációhoz nyújt felületet. A Profibus PA (Process Automation) az adatátvitel megvalósítása mellett alkalmas tápellátás biztosítására is a terepi eszközök számára, akár robbanásveszélyes környezetben is. A protokoll lehetővé teszi a kommunikációban résztvevő eszközök blokk szintű leírását. A PROFIBUS-FMS (Fieldbus Message Specification) elsősorban a vezérlők és felügyelő eszközök közötti, magasabb szintű kommunikációt határozza meg.

Az IEC 61158 és az IEC 61784 –ben megfogalmazott PROFINET egy széleskörű bázis által meghatározott szabvány, amely valósidejű Ethernet-en alapul. A szabvány modularitásával támogatja a valós rendszerek által támasztott különleges igényeket, vannak alkalmazások, melyek esetében elengedhetetlen a nagyon pontos időzítés és a gyorsaság és vannak kevésbé igényes alkalmazások is. Az Ethernet alapú kommunikáció ugyan megengedné bármilyen ethernetes hálózati elem alkalmazását, de a valós idejűség biztosítása csak PRONET tanúsítvánnyal rendelkező eszközök használatával biztosítható. A 2000 –ben megkezdett fejlesztés eredményeként elsőként a PROFINET CBA (component-based automation) látott napvilágot. A szabvány az objektum orientált paradigma alkalmazásával, TCP/IP hálózaton keresztül tette lehetővé a gép-gép közötti kommunikációt. Az alapötlet az volt, hogy az összetett automatizálási rendszerek valójában autonóm egységek összességéből tevődnek össze. Az összetett rendszerekben az egyes komponensek ismétlődve, esetleg kis módosításokkal jelentkeznek. A protokoll által biztosított ciklusidő 50-100ms. A PROFINET IO (input output), a szabvány legutóbbi verziójában jelent meg. A szabvány két módszert kínál a valósidejű kommunikáció megvalósítására az egyik RT (real-time) a legnagyobb erőfeszítés paradigmát, míg az IRT (isochronous real-time) a lefoglalt időablak és sávszélesség paradigmát használja. A PROFINET IO alkalmazható az automatizáció teljes spektrumán, segítségével megvalósítható az eszközök, a vezérlők és az adatgyűjtők közötti kommunikáció is. Az adatátvitel ciklusideje néhány száz mikroszekundum.

A Robert Bosch GmbH által kifejlesztett CAN (Controller area network) kommunikációs szabvány immár a legnagyobb teret kapott szabvánnyá vált az automatizálás a gépjárműipar és az ipari kommunikációs oktatás terén. A fizikai átvitel busz alapú. Az elérhető legnagyobb átviteli sebesség 1Mbit/sec. A CAN sikerrel alkalmazható egyes kritikus szabályozások esetében is, mint például a fékrendszer blokkolásgátlója. A szabvány lényegében az OSI modell első két rétegét határozza meg, leginkább az adatkapcsolati réteget és kisebb mértékben a fizikai réteget.

Az Allen-Bradley cég, kifejezetten az ipari automatizálás céljára javasolta a CAN alapú magasabb réteget képviselő DeviceNet megoldás, ami az IEC 62026-3 alatt került szabványosításra.

A terepi eszközökkel, érzékelőkkel és beavatkozókkal való kommunikáció legfiatalabb képviselője az IO-Link. A főleg német székhelyű konzorciumtagok által kifejlesztett szabvány az intelligens érzékelők és beavatkozók kommunikációs szabványa. Topológiáját tekintve kétpontos megoldás, egy mester és egy terepi eszköz közötti kommunikáció. A mester több, 1-8 terepi eszközzel tartja a kapcsolatot és ugyancsak rendelkezik, legalább egy ipari hálózati interfész-el, melyen keresztül kapcsolódik az integrált automatizálási rendszerhez. A mester és a terepi eszközök közötti kommunikáció a kommunikációs sebesség meghatározásával kezdődik. A mester választ a 3 (4.8 kbit/sec, 38.4 kbit/sec, 230.4 kbit/sec) szabványos sebesség közül egyet. Fontos, hogy a kiválasztott, szabványos sebesség megbízható adatátvitelt eredményezzen. Az üzenetek között találhatók paraméterlekérdezési és beállítási üzenetek. Az érzékelő által szolgáltatott adat leolvasása vagy a beavatkozó eszköz értékének beállítása folyamatosan vagy a mester kezdeményezésével, ciklikusan történhet. A csomagok a mérési-beállítási értékek mellett tartalmazhatnak egyéb paramétereket is.

7. A SCADA helye és szerepe az integrált informatikai rendszerekben

Azon rendszereket melyek lehetővé teszik az ipari folyamatok vizualizációját, a vizuális felület általi beavatkozást, valamint az adatgyűjtést, SCADA rendszereknek nevezzük. A rövidítés az angol "Supervisory Control And Data Acquisition" kifejezésből ered, melynek jelentése "felügyeleti vezérlés és adatgyűjtést". A SCADA tehát egy felügyeleti rendszer, amely vizuális elemek segítségével szemlélteti az ipari folyamatok állapotait. A szemléltetésen kívül a SCADA rendszer segítségével a folyamatba való beavatkozás is lehetséges. Az ember-gép kapcsolatot megvalósító eszközök gyakran jelenítik meg a rendszer egy részét. Egy összetettebb ilyen rendszer magába foglalhatja a mérés, szabályozás és végrehajtás technikai elemeket, az őt ellátó energetikai rendszert, az egész kommunikációs hálózatot, valamint a mind ezek fölött elhelyezkedő számítógépes rendszert, amely a vizualizációt és naplózást végzi. A SCADA rendszerek struktúrájuk szerint a hibrid rendszerek közé tartoznak, mivel tartalmaznak analóg technológiai eszközöket (mérő, végrehajtó eszközök, stb.), számítógépes rendszereket, valamint a rajtuk futó szoftvereket is. A digitális technika fejlődésével a mérő és szabályozó technikai eszközök 80 százaléka mikro és mini számítógépes rendszereken alapulnak. A fő szerep azonban a legfelső szinten elhelyezkedő egymással hálózatban lévő számítógép rendszereknek jutott. Egy SCADA rendszer komplexitása több tényezőtől is függhet azonban egy jól megtervezett, kivitelezett és beállított rendszer hozzájárulhat a gyártási folyamat minőségének, megbízhatóságának javításához. Segíthet a technológiai folyamatban jelentkező hiba detektálásában, információt szolgáltathat a vállalatirányítási rendszernek. Az előnyök a következő pontokba szedhetők:

- Jelentős megtakarítás érhető el a gyártás során az erőforrások gazdaságos kihasználásával és a selejt termékek részarányának csökkentésében.
- Az optimális gyártási paraméterek megadása hozzájárul a termék minőségének javulásához.
- A technológiai eszközök állapotának folyamatos követésével a figyelmeztetések segítségével jelentős kiadáscsökkenés érhető el.
- Hiba elhárításkor a leállás idő minimalizálható és ezzel a veszteségek is csökkenthetőek. Információk nyerhetők ki a megelőző karbantartáshoz.
- A termelés jobb átláthatósága miatt a termeléstervezés és a nyersanyagkezelés sokkal pontosabb lehet.
- Könnyebb és gyorsabb alkalmazkodási lehetőség a piac szükségleteihez, akár egy új termék bevezetése során is.

A továbbiakban bemutatásra kerülnek a SCADA rendszerekben leggyakrabban alkalmazott elemek.

7.1. A SCADA rendszerhez tartozó elemek

Az ipari folyamatok és gyártások automatizálásában felhasznált elemek mindegyike helyet kap a SCADA rendszeren belül. A folyamat szinte minden esetben megjelenik a vizualizáció során. Az elemek főbb csoportjai a következők:

- Intelligens mérőeszközök;
- Adatgyűjtő rendszerek;
- Szabályozók;
- Programozható logikai vezérlők (PLC-k) ;
- Speciális munkaállomások ;
- Központi számítógépes rendszer;
- HMI eszközök.

A továbbiakban röviden bemutatásra kerülnek ezek a csoportok.

7.1.1. Intelligens mérőeszközök

Ezek az eszközök az irányítási rendszer és ezzel együtt a SCADA rendszer legalacsonyabb szintjén helyezkednek el. Számos fizikai mennyiség mérésére, jelenlét érzékelésére szolgálnak. A korszerű mérőeszközök (intelligens érzékelők) már beágyazott rendszerek. Ennek köszönhetően képesek az általános mérőfeladatok mellett más műveletek elvégzésére is, mint például szűrés, konvertálás, diagnosztika, valamint képesek különböző számítások elvégzésére is. Az intelligens mérőeszközök általában rendelkeznek valamilyen kommunikációs interfésszel (IO-Link, Ethernet, soros interfész) melyen keresztül megoszthatja a mért értékeket. Sok esetben támogatják a legelterjedtebb kommunikációs protokollokat is, mint pl. ProfiBus, Modbus, CanOpen, ProfiNet. Ezek az eszközök nem tartalmaznak nagyobb adattárolókat, mivel nem végeznek adatgyűjtést. Általában magas IP védettségi szintű kiszerelésben forgalmazzák őket, így közvetlen beépíthetőek akár a folyamat közelébe, vagy akár a vezérlőszekrénybe is. A több funkcióval rendelkező eszközök, paraméterkonfiguráció céljából rendelkeznek nyomógombokkal és kijelzővel is.

7.1.2. Adatgyűjtő rendszerek

Az adatgyűjtő rendszerek a SCADA rendszerben jelentkező nagy mennyiségű információk, mérési adatok tárolására, naplózására szolgálnak. Az ipari viszonyok közötti alkalmazás minden esetben megköveteli a megbízhatóságot és robosztusságot. Az ipari adatgyűjtő rendszerek ipari körülményekre lettek felkészítve, így ellenállnak a szélsőséges hőmérsékleti értékeknek, páratartalom széles körben való változásának, a mágneses és elektromágneses tér hatásainak. Az adatgyűjtő számítógépek, elnevezésükhöz híven az adatok tárolására és nem azok megjelenítésére és feldolgozására szolgálnak. Ezért, esetükben gyakran elegendő a minimális felhasználói interfész, amely általában pár nyomógombból és egy kijelzőből áll. Az adatgyűjtő számítógép megfelelő bemeneti csatornákkal, interfészekkel rendelkezik, hogy a terepi eszközök csatlakoztatása zavartalan legyen. Sok esetben tartalmaznak analóg-digitális (A/D) átalakítókkal ellátott bemeneti csatornákat is, melyekkel közvetlenül tárolni tudják az analóg jelet biztosító érzékelőkről érkező információkat is. A rendszerek összetettségétől függően a bemeneti csatornák száma néhánytól egészen pár százig változhat, ezért az adatgyűjtő számítógép architektúrája általában moduláris, így méretezhető az aktuális feladathoz és az alkalmazott érzékelő szükségleteihez. Az adatgyűjtő számítógép bemenetei által szolgáltatott értékek gyakran felhasználhatók a SCADA rendszer más elemei felé is. Az adatok, információk ebben az esetben valamilyen ipari kommunikációs csatornán tudnak eljutni a másik számítógépekhez. Az adatgyűjtő számítógép ebben az esetben valósidejű üzemmódban működik és a bemenetére érkezett értékeket valós időben küldi tovább az ipari hálózaton.

Az igazi szerepe az adatgyűjtő számítógépnek mégsem a valósidejű adat tükrözése, hanem a mérési adatok naplózása és tárolása. A mintavételezés periódusideje sok esetben nagy, akár percenkénti illetve óránkénti mintavételezés is lehetséges. Az adatgyűjtők ritkán illetve mondhatni egyáltalán nem tudnak beleavatkozni a folyamat működésébe. A legtöbb esetben a felhalmozott adatokat statisztikai modellek felépítésére, illetve minőségellenőrzési feladatok alátámasztására használják fel. A naplózott adatok alkalmasak a folyamat időbeni rekonstruálására, az előállt esetleges hiba okának felderítésére. Az adatok tárolására gyakran kerül alkalmazásra a round-robin adatbázis struktúra.

7.1.3. A szabályozók

A szabályozók a folyamatirányítás és a gyártásautomatizálás nélkülözhetetlen alkotóelemei. Összetett rendszerek esetében a SCADA betekintést enged a szabályozók paramétereibe, és mint olyan komoly felelősség terheli az automatizált rendszer minőséges működéséért. A folyamatba való beavatkozás történhet akár távolról a központi számítógép vizuális felületéről manuálisan, valamint előre megtervezett automatika beavatkozásával. Az első megoldása valójában kézi irányítás, és mint olyan inkább részmegoldásnak számít, a valódi cél az "igazi" automatizált folyamatirányítás. A digitális technológia valamint a mikroszámítógépek elterjedése lehetővé tette a legösszetettebb szabályozási technikák alkalmazását is. A már elméletben jól ismert szabályzási

algoritmusok kerültek a folyamatirányítási rendszerek gyakorlati megvalósításába, mint pl. Fuzzy szabályozók, adaptív szabályozók, valamint egyéb mesterséges intelligencián alapuló szabályzó algoritmusok. A SCADA rendszerek többváltozós összetett rendszerek esetében is képessé váltak egyszerre több szabályozási kör felügyeletére és menedzselésére.

A SCADA rendszerek szabályozó elemei között említhetjük meg az önálló szabályozókat is. Az önálló szabályozók olyan irányító eszközök, melyek már magukba foglalnak digitális-analóg (D/A), illetve analóg-digitális (A/D) átalakítókat, így közvetlenül a szabályozási körbe építhetőek. Ezek az eszközök a szabályozást a SCADA rendszertől függetlenül végezhetik, az csakis felügyelet és kézi vezérlés céljából kapcsolódhat hozzá. Ilyen szabályozók pl. különböző hőmérsékletszabályozó, szervo motorszabályozó, frekvenciaváltók, stb. Ezeket a szabályozókat általában programozni már nem tudjuk, esetükben csakis különböző üzemmódokat, paramétereket tudunk változtatni.

Az önálló szabályozók beállítási lehetőségei lehetnek:

- A folyamat kézi vezérlése;
- Kézi illetve automata (autotuning) paraméter beállítás;
- Vezérlési algoritmusváltás;
- Az alapjel típusának beállítása, határainak beállítása, stb.

Ezek a beállítások általában az eszközön található egyszerű kezelőfelülettel (nyomógombok, kijelző) vagy a központi számítógépről valamilyen kommunikációs protokoll segítségével valósíthatók meg.

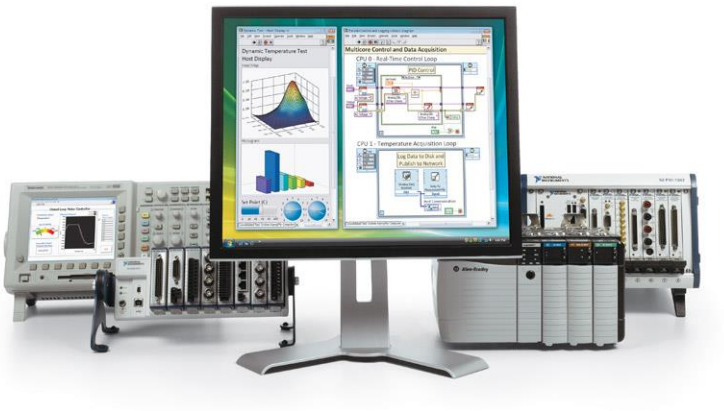
7.1.4. Programozható logikai vezérlők (PLC-k)

A folyamatirányítás legelterjedtebb számítógépes rendszere és SCADA rendszerek legfontosabb eszköze a PLC. Az előző fejezetek részletesen tárgyalták ezt az eszközt. A PLC szerepe a SCADA rendszerekben az információk és az adatok biztosítása a központi számítógép irányába. Az egyszerűbb és az összetettebb SCADA rendszereknél is egyaránt a központi számítógép legelterjedtebb adatszolgáltató eszköze. A PLC szerepének súlya a SCADA rendszerben mindig az adott alkalmazástól az adott folyamattól függ. A PLC a folyamat irányításához szükséges adatokat begyűjti a folyamatban levő érzékelőkről, jeladókról, mérőeszközökről. A vezérlési feladatok mellett a PLC képes ezeket a begyűjtött információkat továbbküldeni más eszközök számára is. A PLC hidat képez az ipari hierarchiában alsó szinten elhelyezkedő terepi eszközök és a felsőbb szinten elhelyezkedő eszközök között. A kommunikációs kapcsolat felhasználásával lehetőség nyílik a folyamatba való távoli beavatkozásra, a PLC-k "kimeneteinek" változtatására. A PLC esetében használt kommunikációs protokoll általában gyártófüggő. Az OMRON esetében például a FINS protokoll. A sikeres adatátvitel érdekében a SCADA rendszernek természetesen ismernie kell az adott protokollt.

7.1.5. Speciális munkaállomások

Az automatizálás területén gyakran kerülnek alkalmazásra diszlokált helyeken ipari számítógépek. Ezek a speciális munkaállomások (Field Point-ok) leginkább az adatgyűjtő számítógépekhez hasonlíthatók. Rendelkeznek ki és bemeneti modulokkal, melyeken keresztül adatokat gyűjthetnek be, megjelenítenek, illetve vezérléseket bonyolíthatnak le. A legfőbb különbség köztük és az adatgyűjtő számítógépek között, hogy rendelkeznek felhasználói felülettel. Az utóbbi években sok esetben előszeretettel alkalmazzák őket az adatgyűjtő számítógépes rendszerek helyett. A speciális munkaállomás állhat egy ipari számítógépből és a rá csatlakozó ki és bemeneti modulokból, valamint lehetnek kompakt kiszerelésű eszközök, ahol a számítógépházba már belekerültek a ki és bemeneti modulok is. Az ipari számítógép általában valamilyen valósidejű operációs rendszert futtatva végzi feladatát. A technológiai folyamat szükségétől illetve alkalmazási területtől függően különböző mérési, adatgyűjtő valamint újabban, felügyeleti feladatokat láthat el. A speciális munkaállomások alkalmazásának nagy úttörője a National Instruments (NI) cég. A legismertebb speciális munkaállomáson futtatott

programcsomag a LabVIEW. A LabVIEW a maga nemében egyedi alkalmazás, segítségével grafikus programozást alkalmazva speciális virtuális mérőműszereket hozhatunk létre a technológiai folyamathoz.



7.1. ábra – : NI ipari számítógép és eszközök.

7.1.6. Központi számítógépes rendszer

A központi számítógépes rendszer képviseli a SCADA rendszer legfontosabb elemét a felhasználói oldali számítógépet. Rajtuk keresztül valósul meg az ember-gép kapcsolat. Ezek a számítógépek magas felbontású képernyőn, egér, billentyűzet segítségével teremtenek kapcsolatot az ember és a folyamatban levő eszközök között. Általában a technológiai folyamattól távolabb eső zajmentes, légkondicionált helységben helyezkednek el. Magas felbontású vizuális elemek, ábrák, diagramok segítségével teszik valós időben követhetővé a technológiai folyamatok fázisait. A megjelenítés mellett lehetőség van a folyamatba való közvetlen beavatkozásra is. Az összetettebb rendszereknél lehetőség van különböző hibaelhárítási algoritmusok alkalmazására, amely az automatizált felügyeleti rendszer alapjait képezi. A központi számítógép platformja általában ipari PC, de a folyamat összetettségétől és fajtájától függően alkalmazhatók egyszerű PC számítógépek akár notebook számítógépek is. A központi számítógép legfontosabb eleme, maga a SCADA alkalmazás. A SCADA program legtöbb esetben egy programcsomag, amely biztosítja a szükséges eszközöket a vizuális megjelenítéshez, az adatbázisba mentéshez, valamint a meghajtókat a PLC vagy egyéb adatgyűjtő eszközök eléréséhez.

A vezérléstechnikai gyártó cégek általában saját fejlesztésű SCADA programcsomagokkal rendelkeznek. Nagymértékben megnehezíti a rendszerek tervezését az a tény, hogy az eszközök általában csakis a saját gyártójuk által fejlesztett programcsomaggal kompatibilisek. Számos törekvés van arra, hogy szabványosítsák a SCADA programcsomagokat, vagyis hogy különböző gyártótól származó eszközök egy SCADA rendszerrel legyenek felügyelhetők.

7.1.7. HMI eszközök

Az ember-gép kapcsolatot biztosító eszközök, a HMI-k (Human-machine interface) elterjedésükkel mind több és több esetben jelennek meg a SCADA rendszer elemeként. A magas felbontású, 65 ezer színt megjelenítő érintőképernyőnek köszönhetően a vizuális rendszerfelügyelet minden igényét ki tudják elégíteni. A korszerű HMI eszközök érintőképernyőjén keresztül a folyamatba való beavatkozás is lehetséges.

7.2. ábra – : Egyszerű és összetett HMI eszközök.

A 7.2. ábrán különböző összetettségű eszközök láthatók, a karakter alapú be/kiviteli eszköztől a saját operációs rendszerrel rendelkező grafikus megjelenítést és érintőképernyős bevitelt lehetővé tevő eszközökig. A HMI típustól függően felhasználhatjuk az egyszerűbb folyamat vizualizációjától, egészen az összetettebb SCADA rendszerekig. A megfelelő operációs rendszerrel rendelkező HMI eszközök a HMI gyártója által fejlesztett megjelenítő és kezelő program mellett szabványos SCADA szoftver futtatására is alkalmasak. Valamennyi HMI eszköz támogatja az ipari kommunikációs interfészek valamelyikét, így szabványos protokollokon keresztül létesíthet kapcsolatot PLC-kel. Ezen kapcsolat folytán érheti el a folyamat adatait. A HMI az ipari hierarchikus rendszerben általában a PLC fölött, az adatgyűjtő, naplózó rendszerekkel megegyező, vagy eggyel alacsonyabb szinten helyezkedik el. Egy jól megtervezett és kivitelezett HMI SCADA rendszer, helyi szinten általában teljes mértékben kielégítheti az ipari folyamat vizualizációs, felügyeleti igényét, sok esetben, mégis csak párhuzamosan használják a központi SCADA rendszerrel. A HMI gyakran a folyamatok közelében van elhelyezve és az aktuális részfolyamathoz tartozó információkat jelenítik meg, a gépet kezelő személyzet könnyebben, gyorsabban tud beavatkozni a folyamatba. Ennek köszönhetően a folyamatirányítási rendszerek minősége és megbízhatósága is nagyobb lett. A tervezés során mindenképp nagy hangsúlyt kell fektetni a biztonságra, arra, hogy csak avatott személy tudjon érzékeny adatokat változtatni, és hogy minden változtatásnak nyoma maradjon a rendszernaplóban.

A HMI kialakítása

Az első HMI-k még szöveges (karakteres) kijelezővel és nyomógombos billentyűzettel jelentek meg. Ezek az eszközök az információt textuális formában tudták megjeleníteni. A kezelő oldaláról az információt fólia billentyűzeten keresztül lehetett bevinni. Tipikusan Ilyen kialakítású HMI eszköz az OMRON NT11 típusú panelje 7.3.ábra.

7.3. ábra – : Az OMRON NT11 típusú HMI.

A numerikus és a navigáló gombokon kívül a kijelző alatt található még 4 funkciógomb is, amelyet előre beprogramozott utasításokra lehet felhasználni. Az NT11 HMI valójában még a

mai napig használatos eszköz egyszerűbb folyamatok, szivattyúállomások, kisebb célgépek felügyeletére. A szöveges kijelzőktől információkban gazdagabb képet tudnak nyújtani a monochrom grafikus kijelzők. Ezek általában két színben képesek megjeleníteni szimbólumokat, szöveget, képeket. Összetettebb vizualizációt is meg lehet velük valósítani, egyetlen negatívumuk a színszegény megjelenítés. A korszerű HMI eszközök színes grafikus TFT kijelzővel rendelkeznek. A HMI panelek több méretben érhetőek el. A képátmérőjük 4 inchtől akár 20 inchig is terjedhet. A nagyobb méretű kijelzők felbontása akár 1280x1024 is lehet 65 ezer színben. Az ilyen karakterisztikájú panelek képesek összetettebb rendszerek vizualizálására is. A részlet gazdag képük az érintőképernyő technológiával kombinálva teljes mértékben kielégíti a SCADA rendszerek minden követelményeit.

A HMI fontos tulajdonsága, hogy általában valamilyen IP szintű (legalább IP 65) por és folyadék elleni védelemmel rendelkeznek. Így olyan környezetbe is telepíthetőek ahol a feltételek más megoldások számára nem a legkedvezőbbek. Kialakítása szempontjából a HMI eszközök általában beépíthetőek, de léteznek mobil kivitelűek is. A mobil kivitel alatt értendő a mozgatható terminálos kivitel, ahol a kezelő panel vezetékkel, vagy akár vezeték nélküli tart kapcsolatot a rendszer többi elemével.

A HMI kommunikációs lehetőségei

A HMI eszközök gyártójuktól, típusuktól függően különböző protokollokat és kommunikációs interfészeket támogatnak. A HMI általában a PLC-vel kommunikálva frissíti a képernyőn megjelenített változók értékeit. Ez a PLC oldaláról jelenthet a folyamat szempontjából jelentős adatot, belső memória címén elhelyezkedő ki és bemeneti értéket, számlálót és időzítőt. A mester-szolga alapú kommunikáció során, üzemmódtól illetve protokolltól függően a HMI panelek működhetnek szolgaként is, azaz nem kezdeményeznek kommunikációt, hanem csak feldolgozzák, illetve megjelenítik azon adatokat melyek a PLC-től és más eszköztől érkeztek. A PLC-k felé való kommunikáció a HMI meghajtói teszik lehetővé. Elengedhetetlen, hogy a kiválasztott HMI a szabványos ipari kommunikációs megoldás mellett rendelkezzen a megfelelő alkalmazás szintű protokollal is.

A HMI mint vezérlő

Az egyszerűbb folyamatok irányítására, kisebb szabályozási feladatok elvégzésére, mára már elengedhetetlenné vált a PLC-k használata. Egy ilyen alkalmazást felügyeleti rendszerének és egyben vezérlésének a megvalósításához fejlesztették ki azokat a HMI eszközöket, melyek rendelkeznek a terepi eszközök iránt szükséges ki és bemenetekkel. Ezeket az eszközöket sokszor emlegetik OPLC néven, mint Operátor paneles PLC-k. Az OPLC a könnyű kezelhetőség érdekében a PLC-n kívül magába foglal egy grafikus érintés érzékeny LCD kijelzőt, vagy egy alfanumerikus illetve numerikus billentyűzetet. A kijelzőn a felhasználó által definiált kezelői utasítások és adatok jeleníthetőek meg. A programozható billentyűzeten keresztül a kezelői utasítások és adatok vihetők be. A 7.4. ábrán egy 5.7 színes, érintőképernyős kijelzővel, 24 programozható nyomógombbal rendelkező OPLC található, melyre 1000 I/O csatlakoztatható és melyhez vezetékes és vezeték nélküli kommunikációs modul szerelhető.

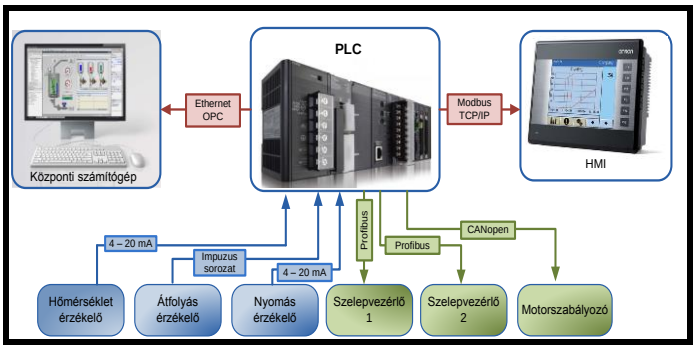
7.4. ábra – : Az UNITRONICS Vision 560.

Az OPLC-k a PLC-hez hasonlóan létradiagramos és a gyártóktól függően egyéb programnyelvekben programozhatóak. Az OPLC-k is rendelkeznek a legelterjedtebb kommunikációs interfészek legalább egyikével. A legtöbb esetben ki és bemeneti modulokkal vagy akár valamilyen kommunikációs interfésszel bővíthetőek. Amennyiben az OPLC erőforrásai megfelelnek az irányítandó, megjelenítendő folyamat igényeinek, akkor egyszerűbb fejlesztést biztosítanak, mint a különálló PLC, HMI alapú rendszerek.

7.2. A SCADA rendszerek struktúrája

A SCADA rendszerek struktúrái nagymértékben eltérhetnek egymástól, attól függően, hogy milyen összetettségű, kiterjedésű, típusú folyamatról van szó. A folyamattól függően beszélhetünk egy számítógépes SCADA rendszerről és több adatgyűjtő eszközt, több számítógépet tartalmazó rendszerről.

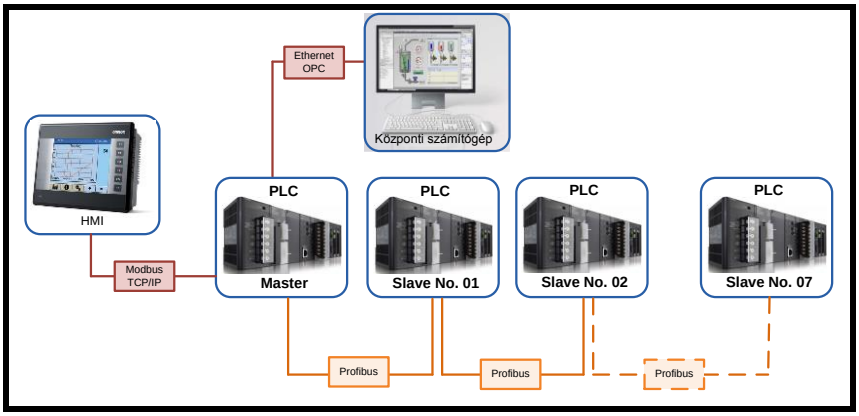
Megoldásában, architektúrájában legegyszerűbb SCADA rendszer állhat egy ipari számítógépből, amely valamilyen adatgyűjtő modul (field point) segítségével információt gyűjt a folyamatról. Egyszerű SCADA struktúrának számít az egy PLC-ből, egy HMI-ből és egy központi számítógépből álló rendszer is 7.5.ábra.



7.5. ábra – : Egyszerű SCADA architektúra.

A 7.5. ábra szerinti megoldásban a struktúrában a folyamat alsó szintjén elhelyezkedő érzékelőkkel és beavatkozókkal a PLC tartja a kapcsolatot. A PLC elsődleges feladata, hogy elvégezze a folyamatirányításhoz szükséges tevékenységeket. A terepi eszközök felé a kapcsolat valamilyen szabványos áramjel, feszültségjel, vagy kommunikációs interfész segítségével történik. A PLC szolgáltatja az érzékelők adatait, mint a központi számítógép, mint pedig a HMI irányába. A példában a számítógép felé OPC-n keresztül valósul meg az adatcsere, míg a HMI felé Modbuson. A PC illetve a HMI lekérdezi a PLC memóriából az érzékelőkhöz tartozó változók értékeit, majd ezeket felhasználva feldolgozzák, illetve megjelenítik azokat.

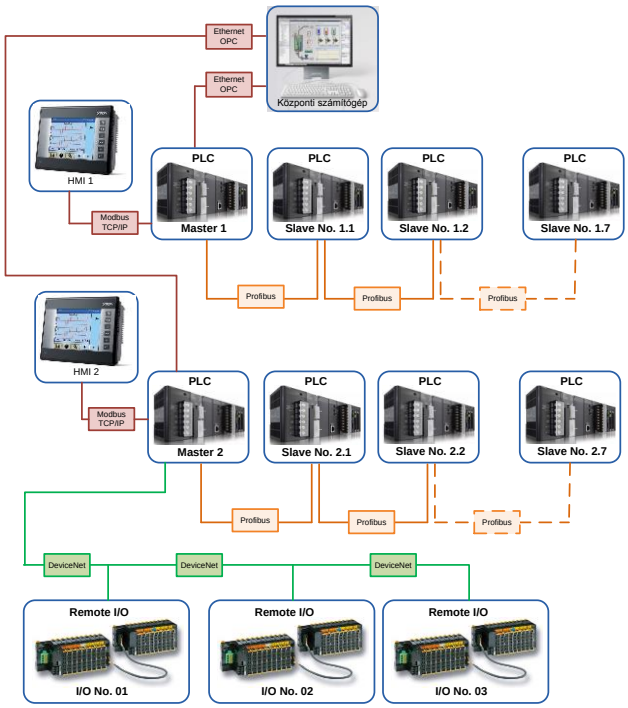
Sok esetben nem az előzőekben bemutatott egyszerű esetekről van szó. Legtöbbször a központi számítógép akár több eszközre is csatlakozhat. Egy ilyen példát láthatunk a 7.6. ábrán.



7.6. ábra – : A mester-szolga kommunikáció SCADA esetén.

Ebben az esetben a központi PC közvetlenül csakis egy PLC-re csatlakozik, ami azt kiszolgálja adatokkal. Ebben a kapcsolatban a mester eszköz a PC míg a szolga eszköz pedig a PLC. Valamilyen kommunikációs interfészen keresztül az első PLC kapcsolatban áll a többi PLC-vel ezért ebben a kommunikációban ő képviseli a mester szerepét. Így amikor a PC lekérdezné az adatot "Slave No. 1" PLC-től akkor azt először a "Mester" PLC-től kérdezi le amely pedig lekérdezi az említett PLC-től. Sok esetben a mester PLC folyamatosan lekérdezi az adatokat a szolga eszközöktől, így amikor a számítógép hozzáfordul az adatokért akkor az azonnal aktuális adatokkal válaszolhat.

Egy összetettebb struktúrájú SCADA rendszert láthatunk a 7.7. ábrán.

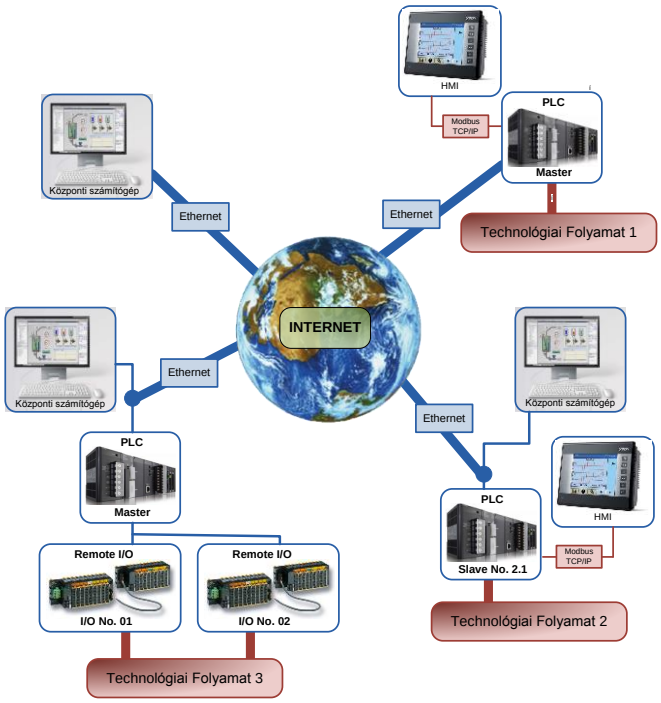


7.7. ábra – : Összetett SCADA architektúra.

A 7.7. ábrán láthatjuk, hogy számos új eszköz jelent meg a rendszerben. Ebben a megoldásban alkalmazásra kerültek a korszerű megoldásnak számító távoli ki és bemenetek

(Remote I/O), úgynevezett I/O szigetek, melyek valamilyen kommunikációs interfészen keresztül egy mester PLC-re kapcsolódva bővítik annak erőforrásait. A már előzőekben megismert PLC-k közötti mester-szolga architektúra itt is jelen van, azzal a különbséggel, hogy itt a folyamat összetettsége miatt egyszerre több mester PLC egység is szerepet kapott.

Az Internet elterjedésének köszönhetően lehetőség nyílt a folyamatok távoli helyekről való felügyeletére, beavatkozás elvégzésére a globális hálózaton keresztül. Megvalósítható az egységes felügyelet biztosítása olyan nagy és elosztott rendszerek esetében, ahol a termelés kihelyezve, több városban, országban zajlik. A 7.8. ábrán egy ilyen struktúrájú SCADA rendszert láthatunk.



7.8. ábra – : A globális hálózat és a SCADA.

A 7.8. ábra szerint a négy különböző helyszínen elhelyezkedő eszközök az internet segítségével teremtenek kapcsolatot egymással. A több helyen működő SCADA rendszerek közvetett módon nyernek információt a rendszerekről. Az internet alapjában véve nem biztosít biztonságos és megbízható kommunikációt. A SCADA rendszer tervezőinek feladata megoldani, hogy biztonságos technológiák és redundáns csatornák felhasználásával orvosolják a hiányosságokat. Az egyes helyszíneken, továbbra is a PLC-k kommunikálnak a helyi terepi eszközökkel. A PLC-k Ethernet interfész segítségével kapcsolódnak az internet felé. A CAD-CAM illetve a magasabb szintű gyártástervező, menedzselő szoftverek elterjedésével mind nagyobb szükség van az internet alapú felügyeleti rendszerekre is. Nem ritka az olyan megoldás sem ahol a folyamatirányításában, felügyeletében csakis távoli I/O modulokat alkalmaznak, melyek a világhálóra kapcsolódva egy távoli helyszínen lévő központi számítógéppel kommunikálnak. Ilyenkor, szinte kivétel nélkül VPN kapcsolaton keresztül zajlik a kommunikáció.

7.3. DCS rendszerek

A DCS (Distributed Control Systems) rendszerek, nagy és összetett rendszerek, folyamatok esetében alkalmazott folyamatirányító és felügyelő megoldások, melyek egy gyártó

termékeként egységes egészet képeznek. A DCS rendszerek jelen vannak a folyamatautomatizálás és a gyártásautomatizálás terén egyaránt. Alkalmazásuk jellemző a kőolaj-finomítók, vegyi üzemek, villamos energiatermelés, a gyógyszer-gyártás, élelmiszerek és italok gyártása, acélgyártás és papír gyártás esetében. Lényeges különbség a SCADA megvalósítással szemben, hogy gyártójuk zárt, kulcs a kézben rendszerként, a technológia teljes egészére forgalmazza őket. A DCS rendszerek fő jellemzője a nagy megbízhatóságot biztosító redundáns vezérlők, I/O egységek, valamint hálózati eszközök alkalmazása. A DCS rendszert alkalmazó beruházások általában gyorsan kivitelezhetők. Egy tipikus DCS rendszer a következőek elemekből épülhet fel:

- Mérnöki és felügyeleti munkaállomások;

A mérnöki munkaállomásokhoz általános felhasználású PC számítógépek tartoznak, melyeken a mérnökök tervezői munkát végezhetnek, valamint hozzáférésük van a felügyeleti rendszer beállításaihoz is. A felügyeleti munkaállomások rendelkeznek a szokványos SCADA felügyeleti elemekkel, mint a vizuális elemek, riasztások, trendek, grafikonok, stb.

- Folyamat archiváló rendszerek;

Ezen eszközök feladata a folyamatban levő összes adat, információ tárolása. Mögöttük, szinte minden esetben egy összetett adatbázis áll. Veszélyes üzem esetében fekete-dobozként is alkalmazhatók.

- SCADA adat server (SCADA Data Server – SDS) ;

AZ SDS átjárót képez az irányítás legfelsőbb szintjein és az alacsonyabb szinteken lévő eszközök között. Megvalósításuk általában valósidejű operációs rendszer, például QNX történik.

- FCU (Field Control Unit) ;

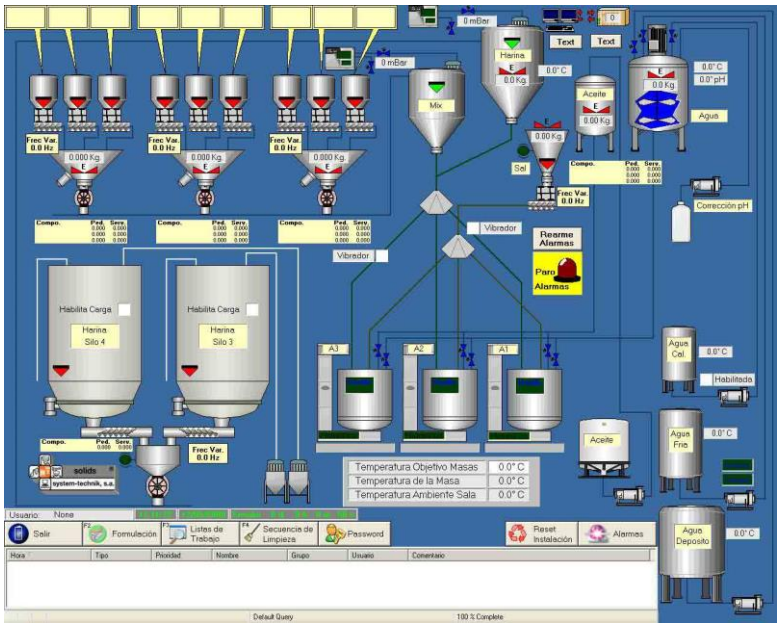
Az FCU segítségével valamilyen specifikus szabályozási feladat valósítható meg. Az implementálható algoritmusuk skálája a legegyszerűbb logikai leképezésektől az összetett numerikus számításokat igénylőig terjed. Sok esetben QNX operációs rendszerrel vannak támogatva amely lehetővé teszi a valósidejű I/O olvasást is. Ebben a szerepében valójában helyettesítheti a PLC-ket is. FCU eszközöket nem csakis kizárólag DCS rendszereknél alkalmaznak.

- RTU(Remote Terminal Unit) ;

Elsősorban távoli adatgyűjtés a fő feladatuk, de számos esetben távoli I/O szigetként is alkalmazhatóak. Sok esetben használják őket valamilyen vezeték nélküli adatgyűjtés, vezérlés kivitelezésénél.

7.4. A technológiai folyamat ábrázolása

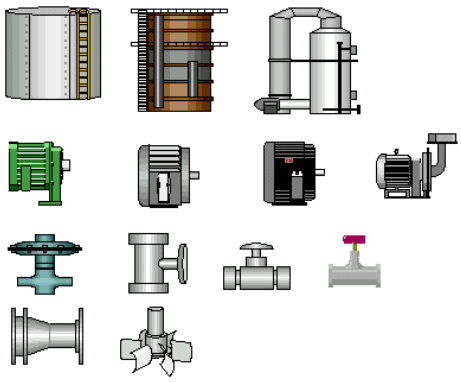
A SCADA által alkalmazott vizualizáció általában kétdimenziós képernyőn történik. A stilizált technológiai folyamatot összekapcsolt objektumok összessége képezi. A képernyő megtervezésénél sarkalatos szempont, hogy lényeg minden esetben szembeötlően ki legyen emelve. A 7.9. ábrán látható egy technológiai folyamat felügyeleti rendszerének az ábrázolására.



7.9. ábra – : Technológia vizualizációja és a SCADA.

Sok esetben a teljes technológiai folyamat részletesen nem ábrázolható egy képernyőn. Ilyenkor a jól elkülöníthető részegységeket egy kibontható elemként ábrázolják. A részegységek kifejtése a legmélyebben levő terepi eszközök vizuális megjelenéséig terjed. Egy alap, globális képernyő a legfőbb folyamatadatokkal együtt tartalmazza a kifejthető szegmenseket.

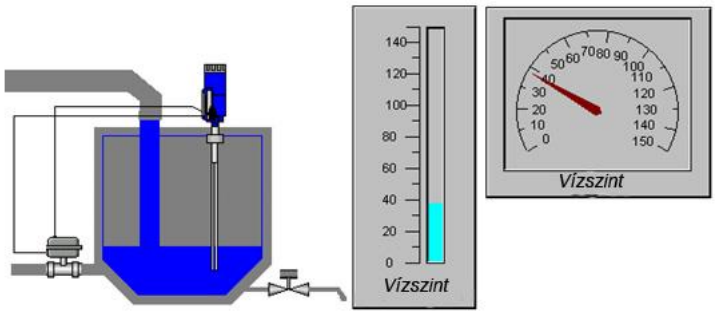
A vizualizáció elvileg bármely programnyelvben és környezetben történhet. Fontos, hogy a program rendelkezzen grafikus megjelenítési lehetőséggel és, hogy hozzáférjen a folyamat információihoz. Napjainkban a SCADA program fejlesztése objektum orientált paradigmát használó környezetekben történik. Számos, jól megépített környezet áll a fejlesztők rendelkezésére, néhány fontosabb: CX-Supervisor az OMRON-tól, SIMATIC WinCC V7 a SIEMENS-től és a Proficy HMI/SCADA (régebben iFIX) szoftver. A könyvtárak által nyújtott objektumok halmaza, a konkrét alkalmazás igényei szerint dinamikus bővíthető. Minden objektum esetében numerikusan megjeleníthetők a legfontosabb tulajdonságok értékei és a folyamatadatok, a kritikus állapotváltozásokat gyakran a megjelentetési szín változása, esetleg hangjelzés jelzi. A fejlesztési környezet gyártói igyekeznek egyre tetszetősebb megjelenítéseket alkalmazni.



7.10. ábra – : Néhány, gyakran használt SCADA objektum.

A folyamat dinamikus viselkedésének követését animációk segítik. Az animáció egyszerűbb esetben jelenthet színváltoztatást, fel-le, jobbra-balra irányú mozgást, forgást, stb. A

vizuális megjelenítés során a statikus szimbólumokon túl a dinamikus viselkedés is hordoz információt. Legalapvetőbb jelzésnek számít a valamilyen formájú színjelzés, vagy színváltoztatás. Ha az adott folyamaton, eszközön belül minden rendben van, akkor pl. zöld jel jelentheti a működést, a piros viszont a hibát. A villogó szimbólumok használata is kiváló hiba illetve riasztás indikátor lehet. Amennyiben az objektumhoz valamilyen felirat is társul, akkor lehetőség van a felirat különböző animálására illetve színezésére. A folyamatok vizualizálásának gyakran használt elemei a virtuális műszerek, melyek segítségével analóg vagy digitális mennyiségek értékeit tudjuk megjeleníteni. A hagyományos analóg mutatójú műszer virtuális megfelelője kiválóan alkalmas a folyamatban jelentkező jellemzők, például nyomás, hőmérséklet, értékeinek megjelenítésére. A másik fontos vizuális elemek a színoszlopok, melyek szintén analóg mennyiségek értékeit hivatottak megjeleníteni.



7.11. ábra – : Analóg mennyiség vizualizációja.

A 7.11. ábra egy tartályban levő folyadék szintjének egy lehetséges ábrázolását mutatja. A színoszlopok működhetnek úgy, mint egyszínű oszlop melynek magasságát az analóg mennyiség határozza meg vagy úgy, mint egy színoszlop melynek színének változása köthető az analóg mennyiség változásához. Színoszlopokkal legtöbb esetben valamilyen szintjelzést, hőmérsékletet szoktak vizualizálni.

A technológiai folyamatok vizualizációjakor nem csak a pillanatnyi értékek számítanak fontosnak, hanem a különböző statisztikai értékek, minimumok, maximumok is. A naplózott értékeket megjelenítő diagramok gyakran valamilyen esemény hatására jelennek meg, például egér kattintás az objektumon, egér mutató pozícionálása. A fejlesztő környezet által nyújtotta lehetőségek tárháza igény szerint bővíthető funkciók, objektumok hozzáadásával. A bővítések magas szintű programozási nyelven, esetleg sctript nyelven készíthetők el.

8. Az OPC kommunikációs protokoll

Egy ipari létesítményben számos olyan eszköz található, melyek egymás között információcserét valósítanak meg. Lehet itt szó PLC-PLC kapcsolatról, PLC-HMI kapcsolatról, PLC-SCADA rendszer kapcsolatról, komplex mérő és adatgyűjtő rendszerekről valamint egyéb magasabb szintű adatáramlásról. Napjainkban az ipari kommunikáció legnagyobb kihívása az információáramlás szabványosítása. Az ipari eszközöket forgalmazó cégek saját eszközeikhez általában fejlesztenek kommunikációs protokollokat is. Ezek a protokollok sok esetben csakis kizárólag egy gyártóhoz tartozó eszközök ismerik. A korszerű automatizálás elengedhetetlen része a számítógépes felügyeleti és mérő-adat gyűjtő-felügyeleti rendszerek (SCADA, HMI, stb.). Ezek a rendszerek az irányítási rendszerek legmagasabb szintjén helyezkednek el. Ahhoz hogy az információ a különböző gyártóktól származó, alsóbb szinteken elhelyezkedő eszközöktől (pl. PLC, adatgyűjtő egység) eljusson a legfelsőbb szinten lévő felügyeleti rendszerekhez, megfelelő szabványos kommunikációs protokoll alkalmazása szükséges. Külön nehézséget jelent amikor a legfelső szintről több rendszer is szeretne az alsó szinten lévő információkhoz hozzájutni. Az összetettebb rendszereknél ilyen esetekben komoly anyagi és időráfordítást igényel a kommunikációs hálózat kiépítése. Az OPC szabvány az ilyen problémákra kínál megoldást.

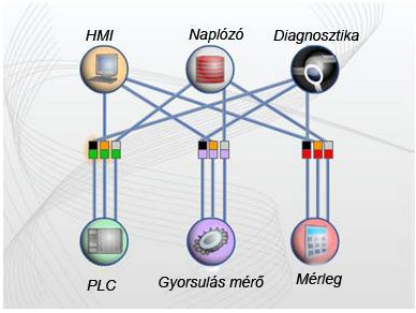
Az OPC rövidítés elejében az OLE for Process Control kifejezésből alakult ki, melyben az OLE az Object Linking and Embedding kifejezésre utal. Az OLE a Microsoft által kifejlesztett COM és DCOM technológiát alkalmazva a szoftverek közötti komponens és adatmegosztást teszi lehetővé. Későbbiekben az OLE a szerepét az ActiveX majd pedig a .net veszi át, így az ”OPC Foundation” 2011-ben megváltoztatta az OPC rövidítés jelentését Open Platform Communications-ra .

Az OPC célja, hogy az ipari infrastruktúra esetében, szabványos formában szolgáltassa a folyamatirányítás adatait. Az OPC alkalmazásával egy új réteg jelenik meg az automatizálási rendszer hierarchiájában. Az új réteg a PLC és fölötte elhelyezkedő felügyeleti rendszerek (SCADA, HMI, stb.) közé illeszkedik be, voltaképpen, mint egy kommunikációs hidat képezve közöttük. Ezzel megszűnik a közvetlen kapcsolat a PLC és a felügyeleti rendszerek között. Az OPC szabvány Szerver és kliens komponensből tevődik össze. Az OPC szerver része az, amely kapcsolatot létesít az alacsonyabb szintekkel és a felsőbb szinteken lévő klienseknek adatokat szolgáltat. Így valójában a mérő és adatgyűjtő, felügyeleti rendszerek OPC kliensek is egyben. A szerver kapcsolódni tud az alsóbb szinteken elhelyezkedő eszközökhöz úgy, hogy a kommunikáció történhet akár saját forgalmazói vagy bármilyen szabványos (Modbus, Profibus, Interbus) protokoll segítségével. Ehhez persze a szervernek ismernie kell ezeket a szabványokat. Az OPC igazi előnye a szerver-kliens közötti kapcsolatban rejlik. Ugyanis, egy szerverre több kliens is csatlakozhat, így többen hozzá tudnak férni a PLC-k, vagy más alsóbb szinten lévő eszközök által szolgáltatott adatokhoz.

Nézzünk egy példát illusztrációként ugyanarra a rendszerre OPC szabványt alkalmazva és nélküle. A példában egy aszfaltkeverő rendszer adatgyűjtő-felügyeleti rendszerét kell megvalósítani. A példában felhasznált eszközök az 8.1. ábrán láthatóak. Az alsóbb szinten található egy PLC, egy motorrezgéseket vizsgáló gyorsulás mérő, és egy magas terhelhetőségű mérleg. A felső szinten található egy HMI, a folyamat vizualizációhoz, egy folyamatnaplózó egység, és egy rendszerdiagnosztikai felügyeleti rendszer.

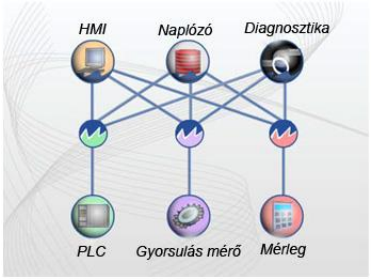
8.1. ábra – : Az aszfaltkeverőben felhasznált eszközök.

Tegyük fel, hogy a PLC Modbus protokoll segítségével kommunikál, a gyorsulás mérő CAN-bus használatával, míg a mérleg DDE (Dynamic Data Exchange) segítségével szolgáltat információt. Ahhoz, hogy a például a HMI hozzáférjen mindhárom eszköz által szolgáltatott információhoz, alkalmassá kell tenni mindhárom protokoll használatára. A másik két felső szinten elhelyezkedő eszköz még csak bonyolítja a helyzetet, mivel egyrészt nekik is ismerniük kell ezeket a protokollokat, másrészt viszont ezek is igényt tartanak az alsó szint által szolgáltatott adatokra. Így például az szerényebb erőforrásokkal rendelkező gyorsulásmérőnek ki kell szolgálnia mindhárom felsőbb eszközt, holott nem elsősorban ez lenne a feladata. A 8.2. ábrán láthatjuk az eszközök közötti kapcsolatokat.



8.2. ábra – : Az aszfaltkeverőben felhasznált eszközök.

A 8.2. szerinti esetben 9 különböző típusú kommunikációs kapcsolatot kell létrehozni. A megoldás sok esetben nem lehetséges, mivel a felső szinten lévő eszközök ritkán támogatják mindhárom kommunikációs megoldást. Érezhető, hogy szükség lenne valamiféle szabványosított közbülső szintre. A rendszerintegráció sokkal gyorsabb, minőségesebb és költséghatékonyabb lehet, amennyiben az eszközök közötti kommunikáció csak is egy protokoll felhasználásával történne. A 8.3. ábrán látható a feladat megoldása OPC szabvány rendszerbe illesztésével.



8.3. ábra – : Az aszfaltkeverőben felhasznált eszközök.

Minden alsóbb szinten lévő eszközhöz hozzárendelünk egy OPC szervert. Az OPC szervernek ismernie kell az adott eszköz kommunikációs protokollját. Az eszköz és a szerver

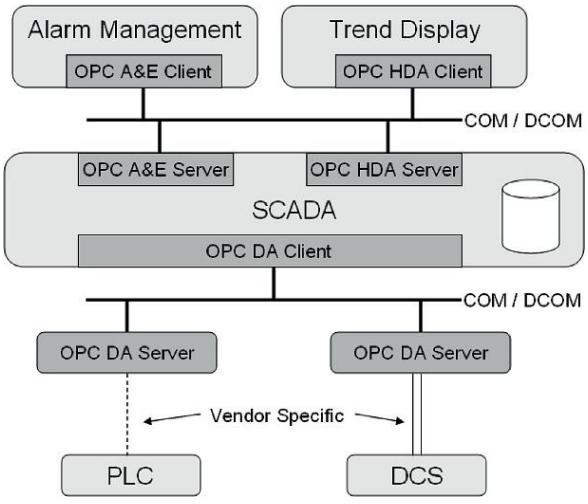
között megmarad az álltaluk alkalmazott saját protokoll. Valójában az OPC szabványosítás igazán csak felfelé történik. Az szerver a rá kapcsoló klienseket szolgálja ki a kért információval. A kliensek ebben az esetben: a HMI, egy folyamatnaplózó egység, és egy rendszer diagnosztikai felügyeleti rendszer. A kliensek az OPC szabvány szerint kérik le az adatokat. Ahhoz a szerverhez fordulnak, amelyiktől az információt szeretnék megkapni. A klienseknek nem kell ismerni az alsóbb szinten lévő eszközök, saját protokollját, csakis az OPC szabvány protokollját. Ez a megoldás gyors kommunikációt és egyszerű megvalósítást eredményez. Az alsó szinten lévő eszközöknek csak 1 szerverrel kommunikálnak így nincsenek túlterhelve az ismételt több irányból jövő lekérdezésektől. A 8.3. ábra szerint a folyamat működése során négyféle kommunikációs kapcsolat jön létre, három a szerverek és a hozzájuk tartozó eszközök között és egy az OPC szabvány szerint.

8.1. Az OPC adatmodellek

Az különböző ipari szükségletektől függően azaz, hogy milyen típusú adatokat szeretnénk az OPC szervertől lekérdezni 3 adatmodellt különböztetünk meg:

- Data Acces (DA)
- Alarm & Event (A&E)
- Historical Data Acces (HDA)

A legelterjedtebb adatmodell a Data Acces, mely az alsó szinten elhelyezkedő eszközökön valósidejű írást-olvasást tesz lehetővé. Az ipari OPC alkalmazások 99%-ban a DA-t alkalmazzák. Igazából a többi OPC interfész adatátvitele is a DA-ra alapul, amint ezt a 8.4. ábrán mutatja. Az OPC DA segítségével PLC memória területéhez tartozó változókat írhatunk, olvashatunk és figyelhetünk meg. Az adatfrissítés a kliens által definiált időközönként történik. Ha valami kommunikációs hiba miatt a szerver nem tudná elérni az eszközt, megtörténhet, hogy nem a legfrissebb adatot szolgáltatja a kliens felé. Ezt a problémát az OPC DA az adathoz csatolt időbélyeggel tudja orvosolni. Az időbélyeg mellett a server minőség osztályozást is végez. Az üzenetben benne van az adat, az időbélyeg, és az, hogy milyen pontosságú a felvételezett adat. Ha pontos, akkor good jelzőt, ha nem elérhető, a bad jelzőt és ha ismeretlen akkor pedig uncertain jelzővel jelöli az adat minőségét.



8.3. ábra – :Az OPC interfészek elhelyezkedése.

Mint már fent említettük az Alarm & Event (A&E) és a Historical Data Acces (HDA) adatmodellek is a DA-n alapulnak. Valójában OPC DA szerver szolgálja ki, mint az A&E server mind pedig a HDA servert. Az Alarm & Event interfész az OPC szabvány azon része, amely értesítések és riasztások generálására használható. Előre definiált események alkalmával a szerver

üzenetet küld a kliens felé, értesítve azt a megtörtént eseményről. Az esemény karakterisztikájától függően lehet értesítés vagy riasztás. A riasztás nagyobb prioritást, nagyobb fontosságú események jelzésére használják. A riasztás típusától függően lehetőség van nyugtázás kérésére is. Például egy nyersanyagtartály töltöttségi szintjét követjük A&E interfésszel, az 50% szinten lehet egy értesítést beállítanunk, míg a riasztást mondjuk 5% ra állíthatjuk. Ha nyugtázás kérés is be van állítva, akkor a riasztás mindaddig fennáll amíg azt le nem nyugtáztuk, még ha közben meg is szűnt a hiba. Az OPC Historical Data Acces a harmadik adatmodell, amely szintén az OPC DA szerverre kapcsolódva szolgáltatja a már eltárolt adatokat. A kliens ebben az esetben a már tárolt adatokat kérheti le egy meghatározott idő periódusra. Ez mellett lehetőség van lekérni egy vagy több változó értékét, amelyek egy bizonyos időbélyeghez tartoznak. Továbbá lehetőség van egy vagy több változóról származtatott adatok (pl. átlagérték, maximum, minimum, stb.) időszerinti lekérdezésére.

8.2. Az OPC jövője

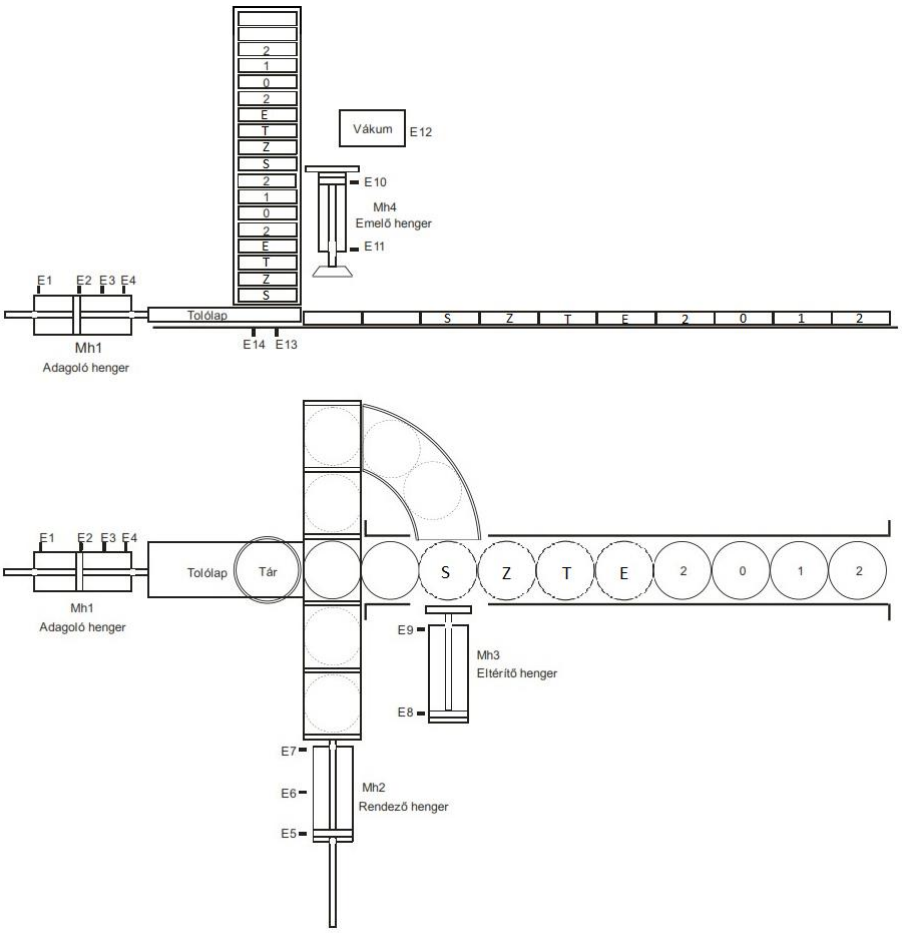
Az OPC mint láthattuk, nagyban hozzájárul az ipari kommunikáció szabványosításához. Az OPC szabvány egyetlen hátránynak tekinthető tulajdonsága az, hogy a COM/DCOM technológia miatt Microsoft, azaz Windows platformhoz kötött szabvány. Számos törekvés van afelé, hogy platform függetlenítsék az OPC szabványt. Így lehet csak igazán elterjedt és széles körben használható. Egy ilyen törekvés hatására született meg az OPC XML-DA szabvány ahol a COM/DCOM technológia HTTP/SOAP és Web Service technológiával van helyettesítve. Az XML-DA teljes mértékben átvette az OPC DA nyújtotta lehetőségeket. Ugyan úgy lehetőség van egy vagy több változó valósidejű írására, olvasására. Az XML-DA új ajtókat nyitott az OPC alkalmazások felé. Mivel ez a szabvány platform független így alkalmazni lehet számos olyan rendszernél ahol Windows felület nem elérhető. Számos alkalmazás szerepel, ugyanis az ipari eszközök palettáján melyeknél szükség lenne OPC implementációjára. Ipari implementációban használt beágyazott rendszerek a platformfüggetlenedésnek köszönhetően mind több estben alkalmaznak OPC-s megoldásokat. A legújabb irányvonalat képviselő OPC szabvány az OPC Unified Architecture az OPC UA.

9. Programozási példák, esettanulmányok

9.1. Korongválogató berendezés

Az alábbi berendezés a 2012-ben megrendezett FIOM Irányítástechnika versenyen került bemutatásra. Az eredeti feladatban a korongokon FIOM2012 felirat szerepelt, továbbá a kirakást háromszor kellett elvégezni. A gép ismertetését Dr. Gyeviki János vezetésével az SZTE Mérnöki Kara végezte.

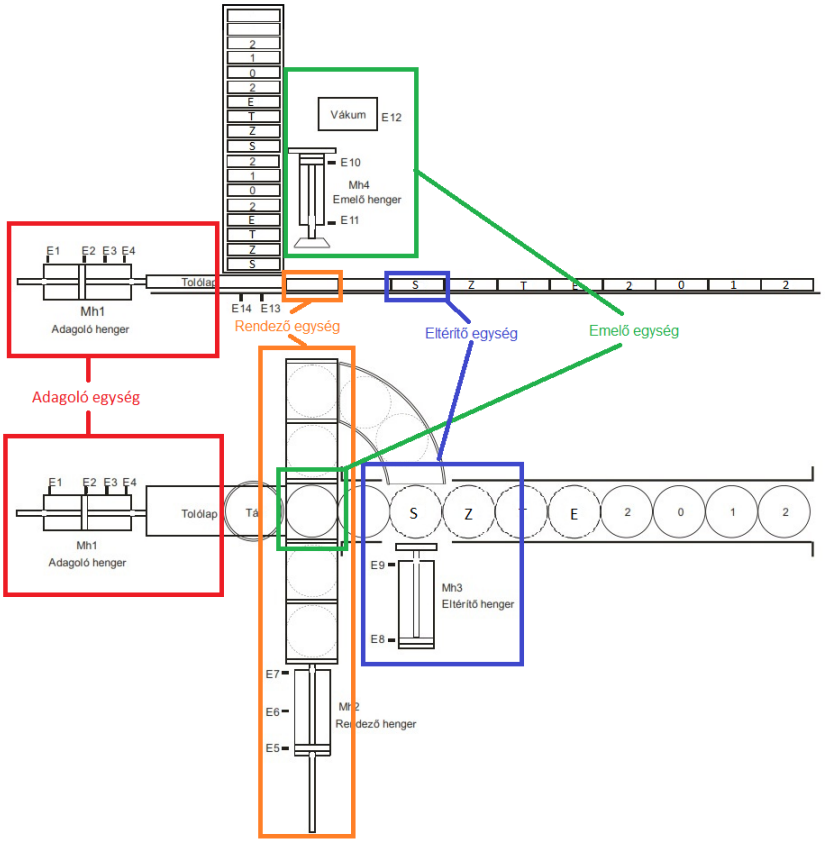
Adott a 9.1. ábrán látható korongválogató berendezés. A csőtárba betárolt 18 darab korongon lentről fölfele az SZTE2012 felirat olvasható kétszer, illetve 2 üres korong a tetején. Ekkor a vályú üres. A feladat az, hogy a korongok kitárolása és megfelelő rendezése után a vályúban a korongokon az SZTE2012 felirat legyen olvasható. Mindezt kétszer kell végrehajtani.



9.1. ábra – : A korongválogató berendezés oldal- illetve felülnézetből.

Mint látható a 9.1. ábrán, a korongválogató berendezés négy fő részből áll, ezek:
- Adagoló egység,
- Rendező egység,
- Eltérítő egység,
- Emelő egység.

A 9.2. ábrán ezeket külön jelölve tekinthetjük meg.

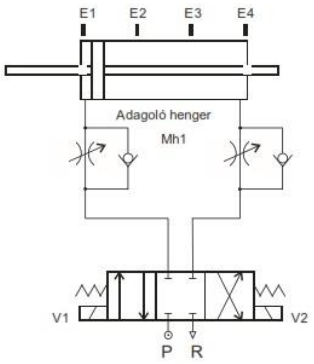


9.2. ábra – : A korongválogató berendezés egyes egységeinek jelölése.

Az alábbiakban tekintsük az egyes egységek pneumatikus felépítését és ismerjük meg a gép működését.

9.1.1. Az adagoló egység

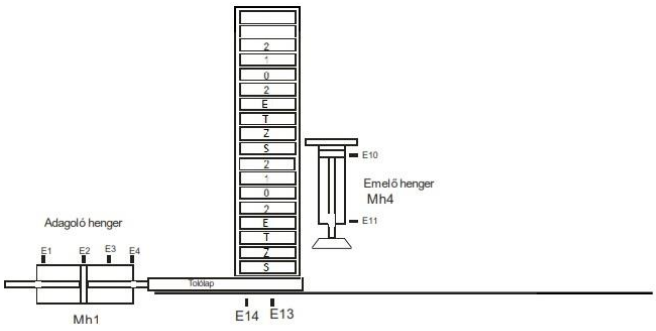
A 9.3. ábrán az adagoló egység pneumatikus felépítését láthatjuk. Egy 4/3-as szelep működését a V1 és V2 jelű elektromos, a PLC kimenetéről származó jelekkel vezéreljük. A V1 jelzésű szelepbemenetre feszültséget kapcsolva a dugattyú tengelyére erősített tolólap kifelé mozdul, tehát az ábra szerint a jobb oldalra indul el, amely majd a korongok kilökését segíti elő a tárból. A V2-esre feszültséget kapcsolva a mozgás iránya ellentétes. Az egyes érzékelők (E1, E2, E3, E4) jelzik a dugattyú aktuális helyzetét, így a megfelelő egységek elé tolhatjuk a korongokat.



9.3. ábra – : Adagoló egység pneumatikus felépítése.

Az adagoló egység a csőtárból a korongokat egy tolólap segítségével, egyesével tolja ki a gép többi egysége elé. Gyakorlatilag működése megegyezik az Mh1 nevű munkahenger működésével.

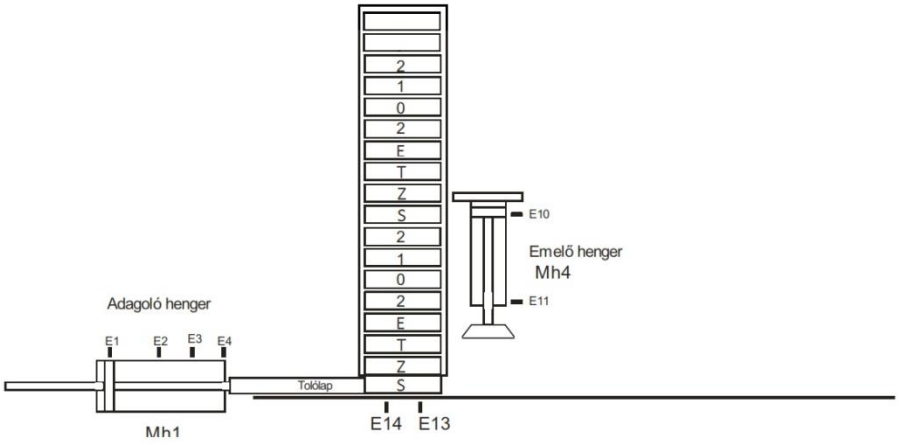
A 9.4. ábrán az adagoló egység kiindulási helyzetét láthatjuk.



9.4. ábra – : Adagoló egység pneumatikus felépítése.

A kiindulási helyzetet az E2 jelű érzékelő jelzi. Az adagolás a START nyomógomb megnyomása után indul. A START gomb a gépen szereplő záróérintkezős nyomógomb, amelynek hatására kell, hogy induljon a gép működése.

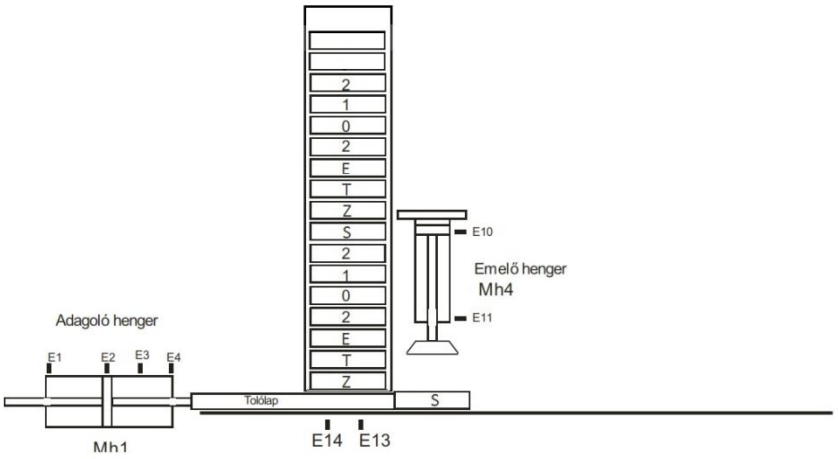
Az Mh1 jelű munkahenger dugattyújának hátsó véghelyzetbe történő mozgatásával folytatódik a vezérlés (V2=1, tehát a 3. ábra szerint a V2-es szelepvezérlő bemenetre feszültséget kapcsolunk.). Ekkor az első korong a tolólap elé esik, ahogy az az 9.5. ábrán látható.



9.5. ábra – : Az adagoló egység hátsó véghelyzeti állapota.

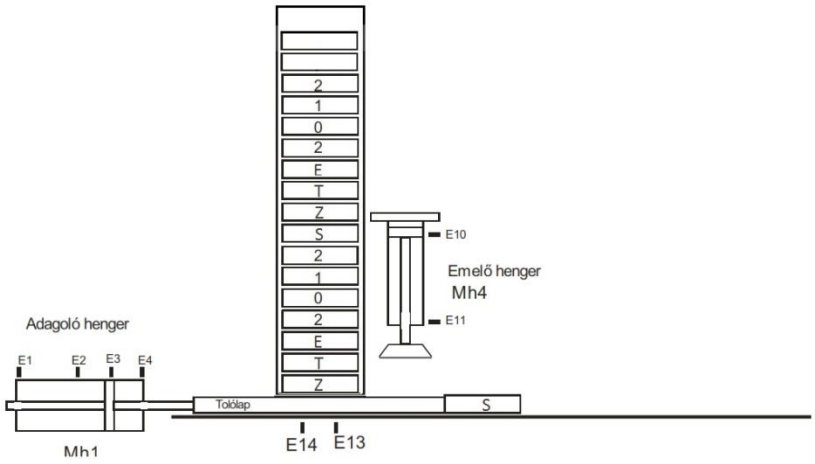
A korongok műanyagból készültek. A tolólap elé eső korongot az E13 jelű kapacitív közelítéskapcsoló érzékeli. Az E14 jelű induktív közelítéskapcsolónak ebben a feladatban nincs szerepe. A STOP nyomógomb megnyomásakor, amely egy bontóérintkezős kapcsoló a gépen, a még folyamatban lévő mozgás befejeződik, és a gép leáll. Ha a START jelű nyomógombot ismét megnyomjuk, a válogatás ott folytatódik, ahol azt megállítottuk. A gép működését a zöld jelzőlámpa folyamatos világítása jelzi, még a piros lámpa villogása a STOP gomb hatására bekövetkező gépleállást jelzi.

A dugattyú hátsó véghelyzetét az E1 jelű érzékelő jelzi. Ha a tolólapot egy osztásnyit mozdítjuk előre (V1=1, tehát a 3. ábra szerint most a V1-es bemenetre kapcsolunk feszültséget), az első korong a rendező egységbe, az emelő henger alá kerül, amelyet E2 jelű érzékelő jelez. Ezt az állapotot a 9.6. ábrán figyelhetjük meg.



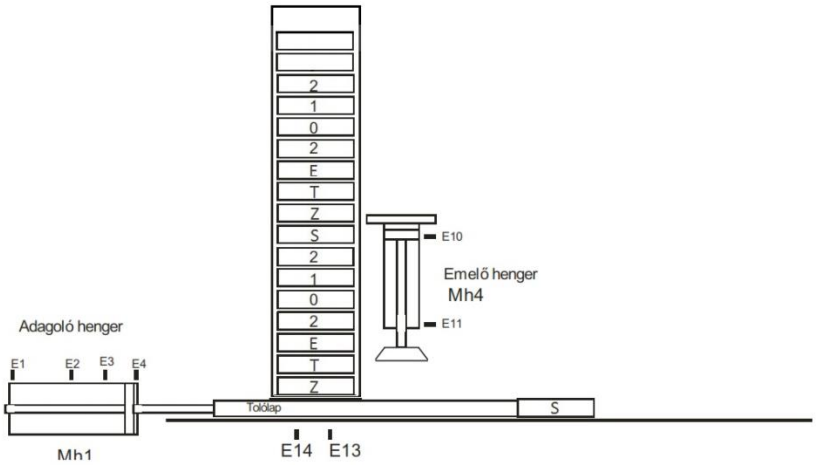
9.6. ábra – : Korong a rendező egységben.

A korongot áttolhatjuk az emelő henger alatt egy osztással akkor, ha a dugattyú mozgását az E3 jelű érzékelő jelére állítjuk meg. Ekkor figyelnünk kell a rendező asztal helyzetére is. A gép ezen pozíciójában a koronggal semmilyen művelet nem végezhető, tehát egy üres terület. Ez az állapotot a 9.7. ábrán figyelhetjük meg:



9.7. ábra – : Korong a rendező és eltérítő egység között.

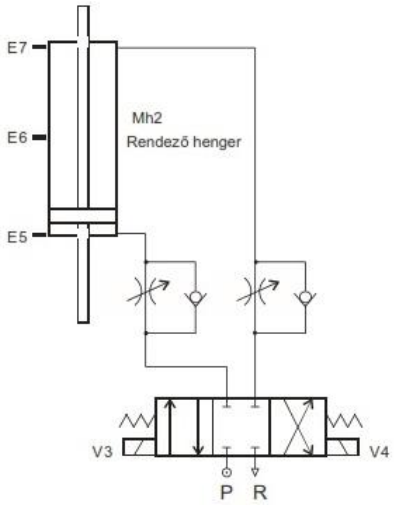
A 9.8. ábra a tolólap első véghelyzetét mutatja. Ekkor a korong az eltérítő egység elé kerül, melyet az E4 jelez.



9.8. ábra – : Korong az eltérítő egység előtt.

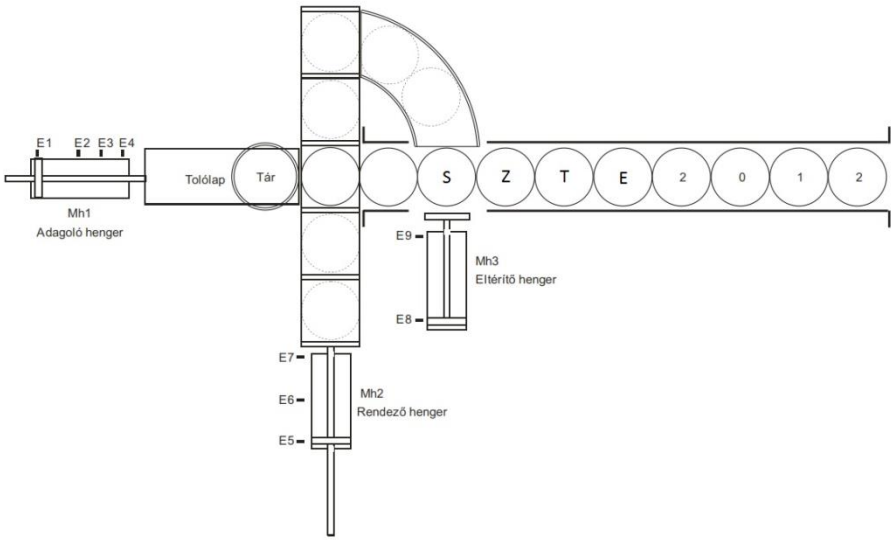
9.1.2. Rendező egység

A rendező egység működése a 9.9. ábrán látható. Nagyban hasonlít az adagoló egység működéséhez, mindössze annyiban tér el tőle, hogy itt a dugattyú csak három helyzetben állhat. A V3 és V4 jelű vezérlőbemenetekkel a dugattyú mozgását a 9.9. ábra szerint rendre felfelé illetve lefelé tudjuk irányítani. V3-ra feszültséget kapcsolva a dugattyú tengelyére erősítőtett rendező asztal mozgása a gép körívet tartalmazó oldala felé mozdul, míg V4-re feszültséget kapcsolva a mozgás ellentétes.



9.9. ábra. A rendező egység pneumatikus felépítése.

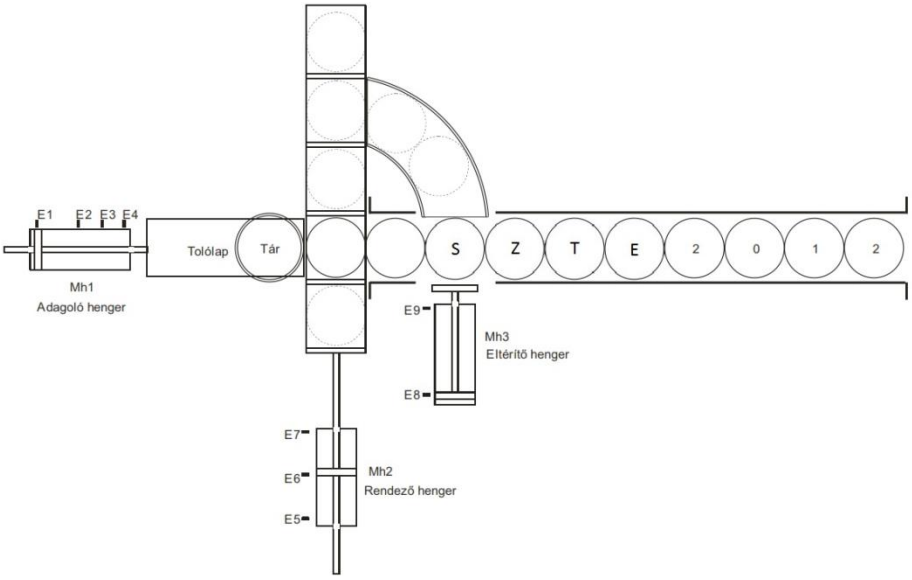
A dugattyú aktuális helyzetéről az E5, E6, E7 jelű érzékelők szolgálnak információval. Az Mh2 jelű munkahenger működtetésének feltétele, hogy az E1 vagy E2 érzékelő jelezzen (adagoló egység érzékelői, tehát a tolólap nem lehet kitolva, amennyiben a rendező asztalt szeretnénk mozgatni). Az alaphelyzet, amely az Mh2 munkahenger hátsó véghelyzete a 9.10. ábrán látható.



9.10. ábra – : A rendező egység hátsó véghelyzete.

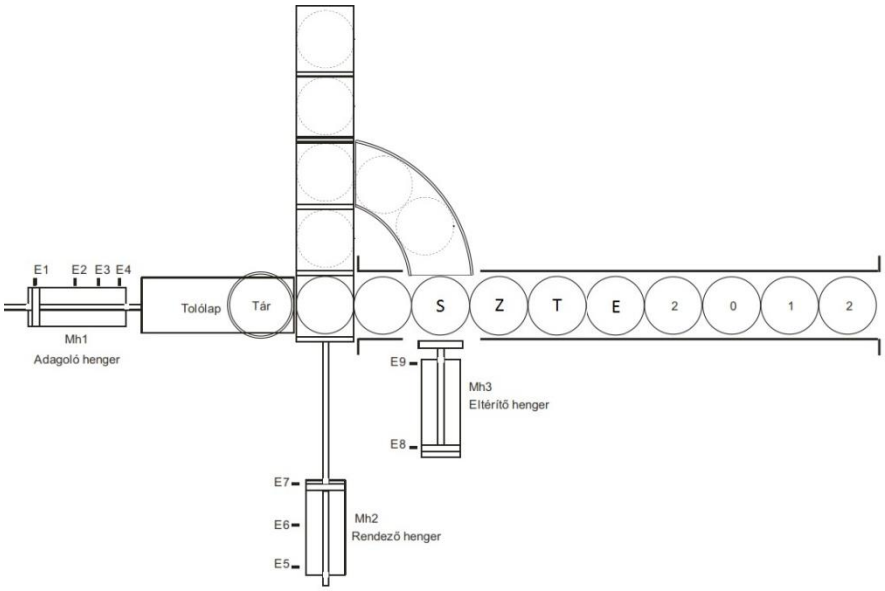
Mint már a 9.9. ábra tárgyalásakor láttuk, a dugattyút V3=1 jellel vezérelhetjük pozitív irányba, míg a hátramenet V4=1 jel kiadásával indítható el.

Ha a rendező asztalt, amely öt rekeszt tartalmaz, ahogy az a 9.2. ábrán látható, egy egységnyivel eltoljuk pozitív irányba (tehát a 9.9. ábra szerint V3=1), akkor az adagoló egység a következő rekeszbe tud korongot pakolni, vagy az emelő egység innen tud korongot felvenni vagy ide tud korongot letenni. Ezt az állapotot láthatjuk a 9.11. ábrán:



9.11. ábra – : A rendező egység dugattyúja a középső helyzetben.

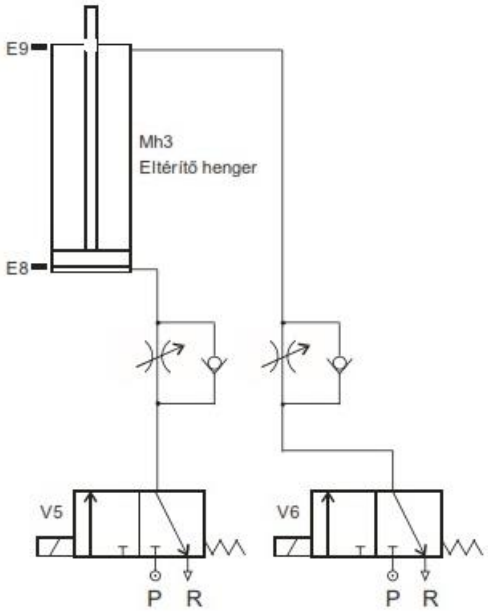
A rendező asztal felső véghelyzetét az E7 jelű érzékelő jelzi, ez a rendező asztal legalsó rekeszét teszi elérhetővé az emelő és adagoló egység számára, ezt az állapotot a 9.12. ábrát tekintjük meg.



9.12. ábra – : A rendező egység felső véghelyzete.

9.1.3. Eltérítő egység

A 9.13. ábrán látható a korongválogató berendezés eltérítő egységének pneumatikus felépítése. Mint látható, az eddigi egységektől eltérően, 3/2-es szelepek látják el a dugattyú mozgatását. Ennek okán érdemes észrevenni, hogy amennyiben egyik szelep egyik elektromos bemenetén sincs jel, akkor a hengerben, a dugattyú mindkét oldalán az atmoszferikus nyomás lesz jelen. Mivel az eltérítő egység feladata a korongoknak a gépen található körívbe lökése, ezért ez egy többnyire lassan elvégzendő feladat.

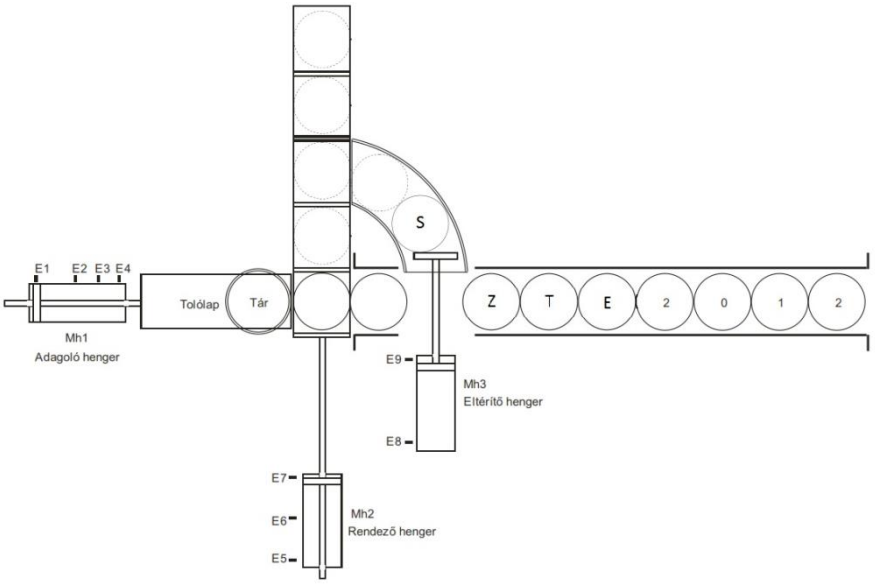


9.13. ábra – : Az eltérítő egység pneumatikus felépítése.

Ha azonban minden előzetes jeltartás nélkül a V5 vagy V6 jelű, a megfelelő szelepeket vezérlő elektromos bemenetekre feszültséget kapcsolunk, a pneumatikus nyomásforrásból (a 9.13. ábrán P-vel jelölt) a levegő nagy sebességgel és nyomással áramlik a dugattyú egyik oldalán a

hengerbe, amely a dugattyú gyors mozgását eredményezi, mert annak csak a külső nyomást (a 9.13. ábrán R-rel jelölt) kell leküzdenie. Ezért, majd a programban is látni fogjuk később, az eltérítő Mh3 nevű munkahenger esetében a vezérlés úgy történik, hogy egyik szelepen folyamatosan tartani kell a jelet, és csak irányváltáskor kell ezt a tartást levenni és a jelet az ellenkező szelepre adni. Ezzel elérhető, hogy a beáramló levegőnek a hengerben még fennálló nyomást kell legyőznie, ami az eltérítő egység lassabb működését eredményezi.

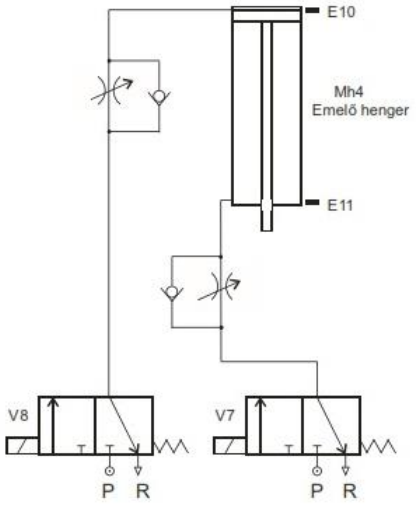
Ha a V5 jelű szelepvezérlő bemenetre feszültséget kapcsolunk, a dugattyú tengelyére erősített lap az eltérítő egység előtt álló korongot a körívbe tolja, ezt a helyzetet az E9 jelű érzékelő jelzi. Ezen állapotot figyelhetjük meg a 9.14. ábrán.



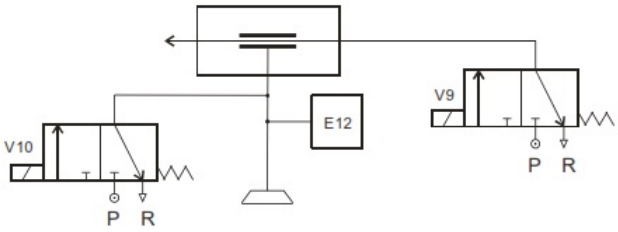
9.14. ábra – : Az eltérítő egység a felső véghelyzetében, korong a körívben.

Amikor a pálya megtelik (két darab korong van rajta) új korong érkezésekor az első korong visszatér a rendező egységbe, feltéve, hogy az íves pálya előtt álló rekesz üres. Ha a rekesz nem üres, a visszatérő korong kilöki a benne lévőt. Ez hibának számít. Mh3 működtetésének további feltétele az, hogy Mh2 pozícióban legyen. Ezt E5 vagy E6 vagy E7 jelzi (rendező egység, lásd: 9.9. ábra). Ha V6=1, tehát a 9.13. ábra szerint a V6 bementre kapcsolunk feszültséget, az eltérítő henger dugattyúja negatív irányba, vagyis visszafelé mozog, az eltérítő be lesz húzva.

9.1.4. Emelő egység



9.15.a. ábra – : Az emelő berendezés fel-le mozgását végző rész pneumatikus felépítése-

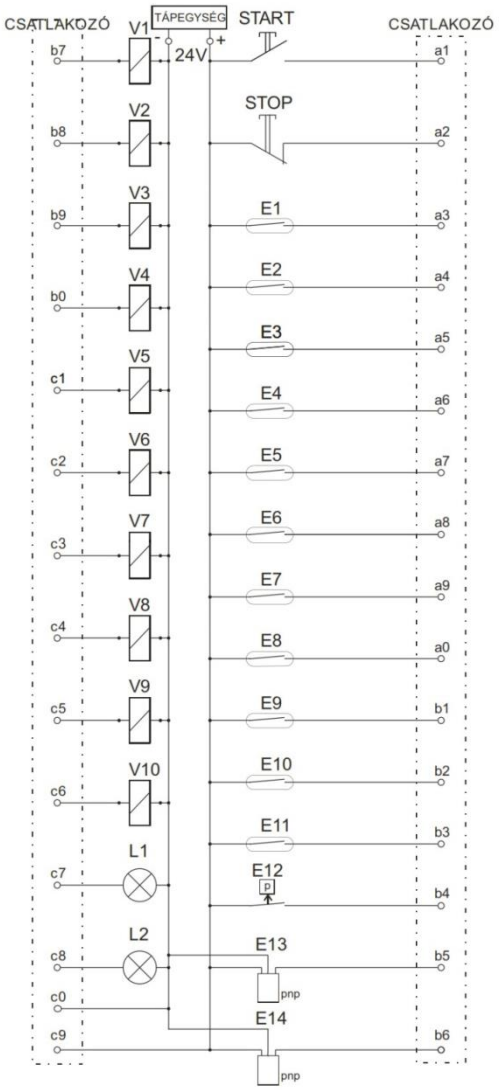


9.15.b. ábra – : Az emelő egység megfogó részének pneumatikus felépítése.

A 9.15.a. ábrán látható a berendezés emelő egységének pneumatikus felépítése. Az Mh4 nevű munkahengerben a dugattyú mozgását a V8 és V7 elektromos, szelepvezérlő bemenetekkel tudjuk megszabni, az ábra szerint rendre a dugattyú le ill. fel mozgása érhető el. Így a megfogó korong leengedhető egy korong megfogásához vagy elengedéséhez, illetve a megfogott korong felemelhető. A 9.15.b. ábrán a V9 nevű elektromos, szelepvezérlő bemenetre feszültséget kapcsolva nagy sebességgel levegő kezd áramlani. Ha egy korong hozzáér a megfogó részhez, akkor a nagy sebességű levegőáramlás hatására keletkező vákuumhatás megfogja az emelő berendezés alá kerülő korongot. Mindaddig kell a levegőt áramoltatni, ameddig tartani szeretnénk a korongot. Letevéskor, ha esetlegesen hiba történne, és a levegőáramlás megszűnése nem eredményezné a korong elengedését, akkor a V10 szelep a lerakást segíti, egy 0,5 szekundumos impulzussal.

A dugattyú két véghelyzetét az E10, illetve E11 szenzor jelzi. Ha a korong megfogása sikeresen megtörtént az E12 jelű vákuumkapcsoló jelet ad.

A bemenetek/kimenetek bekötése a 9.16. ábrán látható. A Vx | x∈[1,10] bemeneteket letárgyaltuk az előzőekben. Látható a szintén már tárgyalt START gomb, amely egy záróérintkező, illetve a STOP gomb, amely pedig egy bontóérintkező. Továbbá megfigyelhetők az E betűvel kezdődő jeladók, amelyek a dugattyúk helyzetéről illetve egyéb, már tárgyalt eseményekről szolgálnak információval. Az L1 és L2 jelölések pedig a zöld, ill. piros lámpákat szimbolizálják.



9.16 ábra – : A bemenetek és kimenetek bekötése

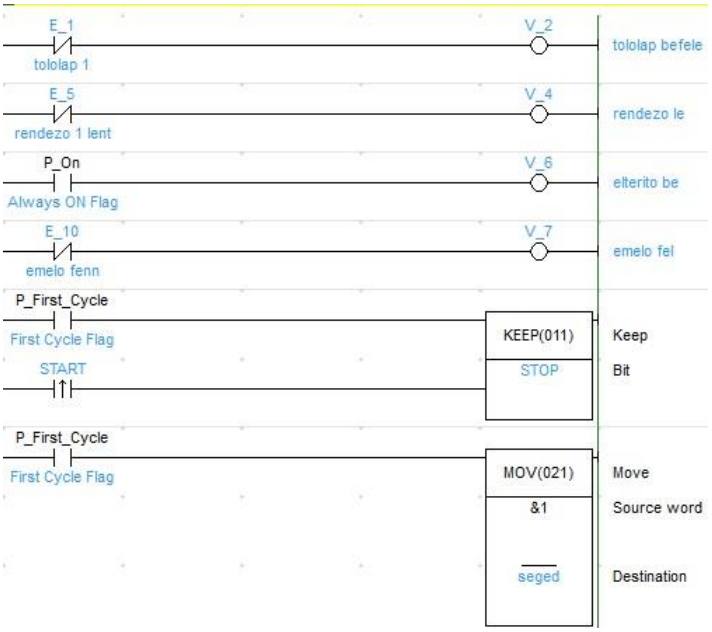
A versenyen a kiírt feladatra több megoldás is született. Tekintsük át a Szegedi Tudományegyetem Természettudományi és Informatikai Karát képviselő csapat megoldását:

A megoldáshoz használt PLC: OMRON CJ2M, CPU32, ID211 és OD212 be/kimeneti modulokkal. A feladat jellege lehetővé tette, hogy egy olyan programozási nyelvet használjunk, amely egységekre bontja a gép működésének folyamatát, és az egyes szakaszok csak az őket megelőző szakaszok után indulhatnak, mikor valamilyen feltételek beteljesülése megtörtént. Erre a legalkalmasabb, az IEC-61131-3-as szabvány szerint is támogatott a korábbiakban már bemutatott programozási nyelv, az SFC.

A feladat megoldásának első lépéseként fontos, hogy a gépet mindenképpen állítsuk a működéséhez szükséges alaphelyzetbe, mivel nem tudjuk, hogy milyen helyzetben hagyták, vagy esetleg valamilyen hiba miatt nem meghatározott pozícióban állt meg. Ennek érdekében a program belépési pontján, a kezdőállapotban (Initial step) gondoskodunk minden munkahenger alaphelyzetbe történő vezérléséről. Ezt a lépést láthatjuk a 9.17.a. ábrán:

9.17.a. ábra: A program kezdőállapota.

A kezdőállapot megkülönböztethető kettős keretéről (ahogyan az a könyv korábbi részében már láthattuk). A továbblépési feltétel, hogy minden dugattyú legyen véghelyzetében. Mivel a tolólapos munkahenger is elment E1-be, azaz első véghelyzetébe, ezért már korongleesés is történt, így figyelni kell az E13-as kapacitív közelítéskapcsoló jelét is, amely a leesett korong jelenlétét jelzi, lásd 9.1. ábra. Emellett a STOP gomb állapotát is figyelni kell, hogy ellenőrizzük, hogy nincs a gép stop állapotban. A kezdőállapothoz tartozó akció LAD nyelven lett megírva, ezt láthatjuk a 9.17.b. ábrán:



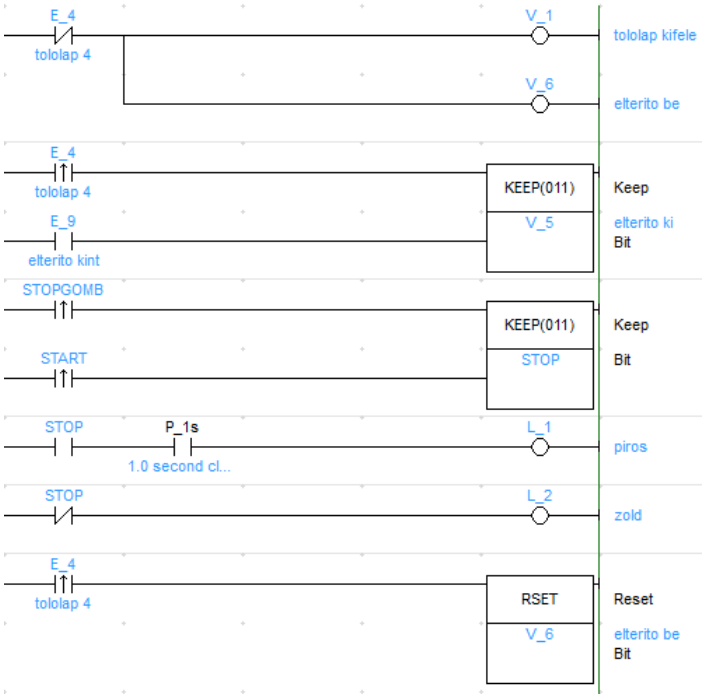
9.17.b ábra – : A kezdobeallitasok, LAD nyelven írt akció.

Látható, hogy bontóérintkezővel addig tartjuk az egyes szelepeket nyitva, még azok a levegő beáramlás segítségével a dugattyúkat a megfelelő jeladó elé nem tolják. Amennyiben az adott jeladó jelez, a szelepek működése leáll. Az eltérítő munkahenger működése kissé eltér a többitől, ugyanis őt úgy kell vezérelni, hogy minden irányváltási esetben a munkahenger azon felében, amerre a dugattyút mozgatni szeretnénk, kezdetben nyomást kell adnunk, hiszen ellenkező esetben a kitérítés túlságosan gyors lenne, lásd a 9.13. ábrát. Rögtön az első ciklusban (P_FirstCycle, amely csak a legelső ciklusban aktív kapcsoló) egy seged nevű segédváltozóba 1-et helyezünk, ez fogja majd számolni a kirakott SZTE2012-k számát (valójában eggyel többet mutat mindig, mert kezdetben 1-et tettünk a memóriába és ezt növeljük minden kirakáskor, ennek semmilyen speciális szerepe nincs, mindössze a megoldáskor ebben állapodtunk meg, hogy egytől kezdjük a számolást). Ugyanígy egy STOP bool típusú változó értékét is aktívan tartjuk, amíg meg nem nyomják a START gombot. Ha ez megtörtént, akkor a gép elindul, és állapotváltás következik be a programban is, hiszen az 9.17.a. ábráról leolvasható, hogy ekkor az átmeneti feltétel teljesül.

Átlépünk a második, immár teljesen átlagos állapotba. Ne feledjük, a tárból egy korong leesett, az S betűt tartalmazó. Ezt majd utoljára kell a vályúba tenni, mert azt SZTE2012 felirat vályúba helyezésekor ez a betű van legelöl, így a sorrendben ő az utoljára elhelyezendő, ezért valahova el kell raktározni a gépen belül, ahol nem zavarja a többi korong helyrerakását. Erre

alkalmas hely az eltérítő egységben lévő körív. El kell tolnunk tehát az S-t a eltérítő elé, majd annak segítségével a körívbe kell lökni, ezt csinálja a következő rész, lásd 9.18/a és 9.18/b ábra:

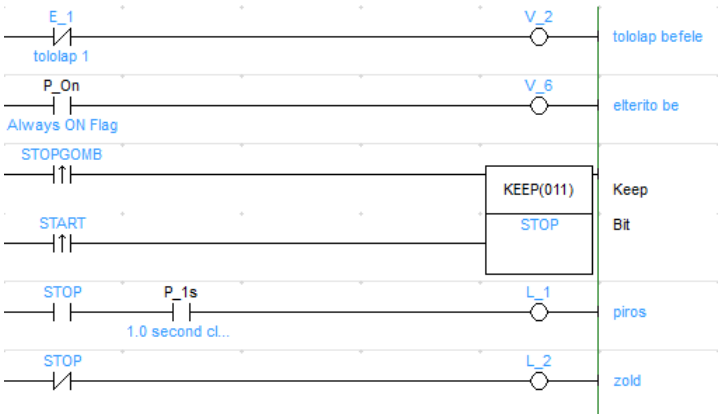
9.18.a. ábra – : elso_S_korivbe step.



9.18.b. ábra – : kezdobol_kiteritohoz_es_kiterit LAD akció.

Miután az S betűt tartalmazó korong az előző lépésben leesett, elkezdjük elmozgatni az eltérítő elé. Mindezt úgy tehetjük meg, hogy amíg az E4 (vagyis a tolólap kitolt véghelyzetét jelző) jeladó nem jelez, addig toljuk kifelé a lapot. A korong az eltérítő elé kerül, majd E4 felfutó élére elkezdjük mozgatni a kitérítő hengert (előtte még adunk a behúzási irányra is aktív értéket, amit szintén E4 felfutó élére kapcsolunk ki, a dugattyú lassabb mozgása érdekében, lásd a 9.13. ábrát). E9 jelzésére, vagyis ha a eltérítő kiért, vagyis ha a korong a körívbe került, levesszük a nyomást a kitérítésről. Teljesül a továbblépési feltétel, újabb lépés következik, amelyben visszamegyünk E1-es pozícióba a tolólappal, beolvassuk tehát a Z betűt, és a kitérítő hengert is visszavisszük az alapállapotába. Ezt láthatjuk a 9.19.a. és 9.19.b. ábrákon:

9.19/a. ábra: elso_Z_beolvas step.

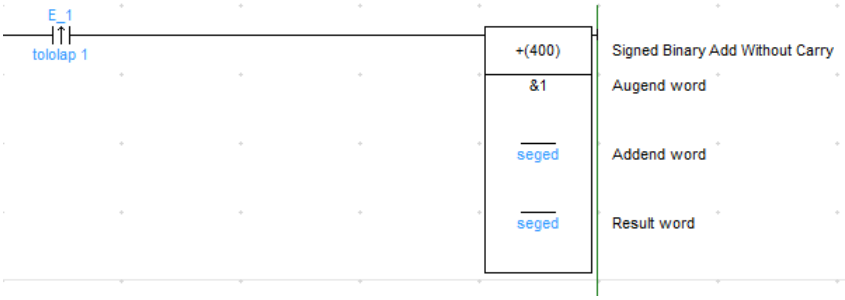


9.19.b. ábra – : E4_bol_vissza_betutolvas_kiterito_vissza LAD akció.

Látható a 9.19.a. ábrán, hogy átmeneti kritériumok, hogy a tolólap E1-es pozícióban legyen, a kitérítő E8-asban, vagyis behúzva, és korongnak is esnie kellett, továbbá nem nyomták meg a STOP gombot.

A következő lépésben a Z betűt tartalmazó korongot is el kell tolni, elvinni az eltérítő egység elé, eltéríteni, majd beolvasni a következő betűt (T) tartalmazó korongot:

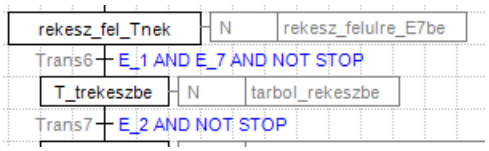
9.20.a. ábra – : Z-t tartalmazó korong eltérítése és T-t tartalmazó korong beolvasása.



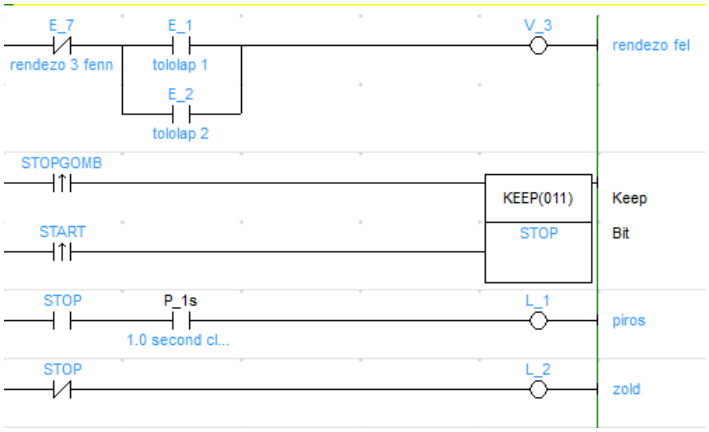
9.20.b. ábra – : Ciklusszamlalo LAD akció.

A ciklusszámláló azért kell, hogy megállapítsuk, hogy az innentől kezdődő rész hányszor futott le, ami valójában megmondja, hogy hányszor raktuk ki a feliratot. Látható egy Trans35 nevű becsatlakozás az átmenetkor, a 9.19.a. ábrán, T_beolvas előtt. Ez a program egy később szakaszából származó visszaugrás erre a részre, úgynevezett jump. Erre azért van szükség, hogy a program egy távoli pontjáról visszaugorhassunk ide, hogy az ettől kezdődő részt többször is végrehajthassuk. Így érhető el, hogy egy belső számlálót növelve és értékét figyelve megállapíthatjuk, hogy hányszor hajtottunk végre valamit, és eldönthetjük, hogy ismét végre szeretnénk-e hajtani.

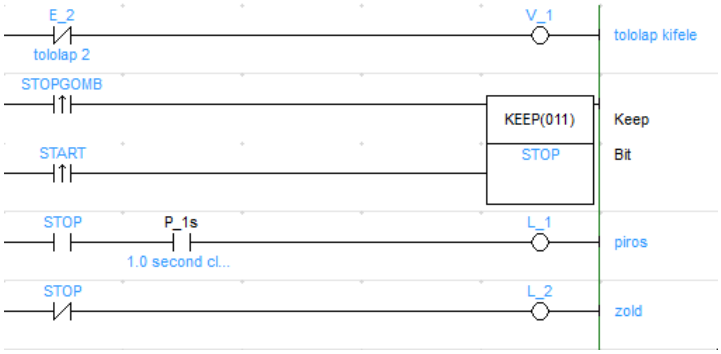
A körívben található az S és Z betűket tartalmazó korongok és a tárból kiolvastunk egy T-t, de még ez is akadályozná a 2012-es szövegrész kirakását, ezért a T-t és az E-t a rendező egység rekeszeiben fogjuk tárolni.



9.21.a. ábra – : rekesz_fel_Tnek és T_rekeszbe stepek.



9.21.b. ábra – : rekesz_felulre_E7be LAD akció.

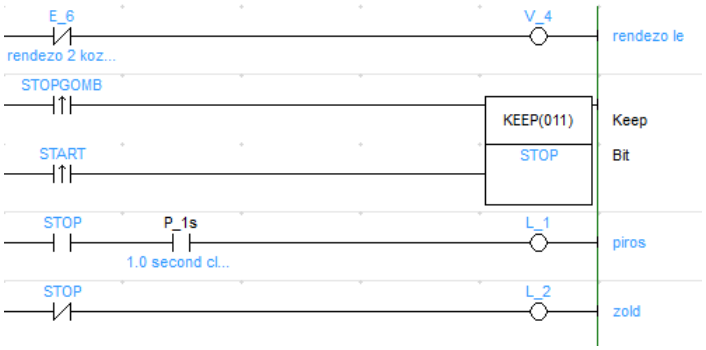


9.21.c. ábra – : tarbol_rekeszbe LAD akció.

A fent leírtakat úgy oldhatjuk meg, hogy a rendezőasztalt elmozdítjuk olyan pozícióba, hogy a legalsó rekeszbe pakolhassunk. Ezután betoljuk oda a T betűt tartalmazó korongot. A rekesz_felulre_E7be első sorában egy VAGY kapcsolatban lévő két érintkezőt láthatunk, ugyanis a rendezőasztalt csakis akkor szabad mozgatni, ha a tolólap nincs kitolva, vagyis vagy E1 vagy E2 jelez.

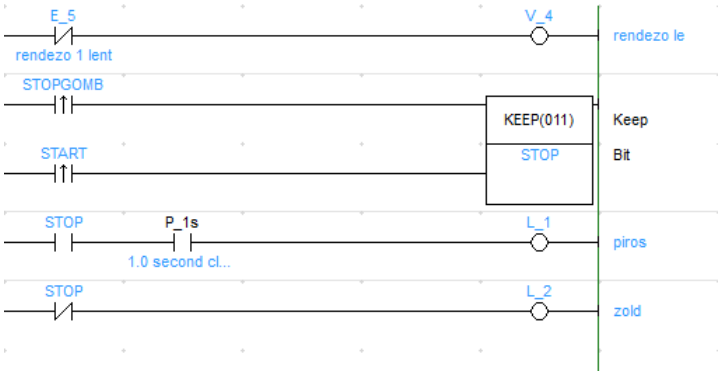
A továbbiakban E-t is hasonlóan mozgatjuk, csak a rendezőasztal alulról második tároló egységébe. Ezt láthatjuk a 9.22. ábrán.

9.22.a. ábra – : E betűt rekeszbe tévő programrészlet.



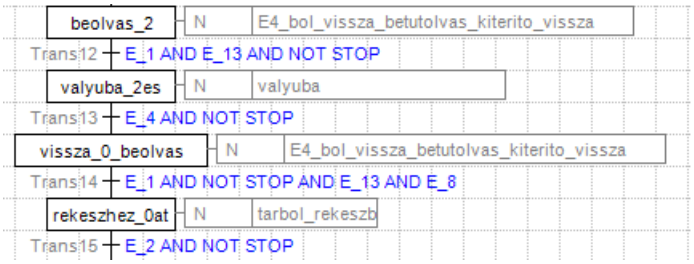
9.22/b. ábra: rekesz_kozep_fentrol LAD kódrészlet

Azért „közép” szerepel az akció nevében, mert ha belegondolunk, a rekesznek (rendező asztalnak) csak az alsó 3 része használható, hiszen a felső kettő elérhetetlen, mivel a munkahenger alsó pozíciójában is csak a 3. hely érhető el, tehát itt a „közép” a 3-ból a középsőre értendő.



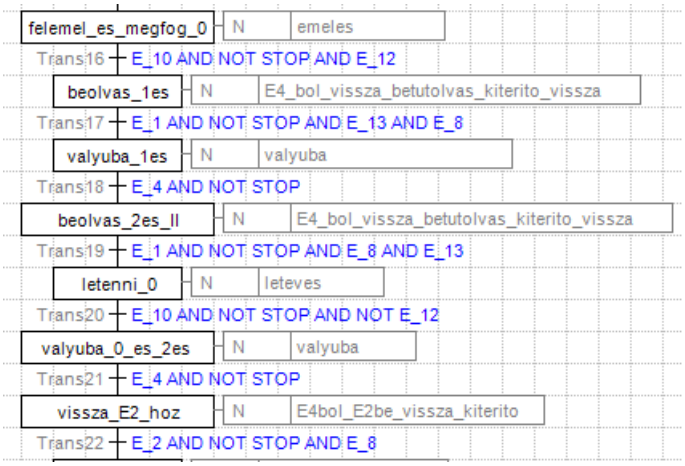
9.22.c. ábra – : rekesz_le LAD kódrészlet.

A továbbiakban beolvasunk egy 2-est, eltoljuk a vályúba, visszajövünk egy 0-sért és elmozgatjuk a rendező asztal egy rekeszébe. Ezt láthatjuk a 9.23. ábrán.

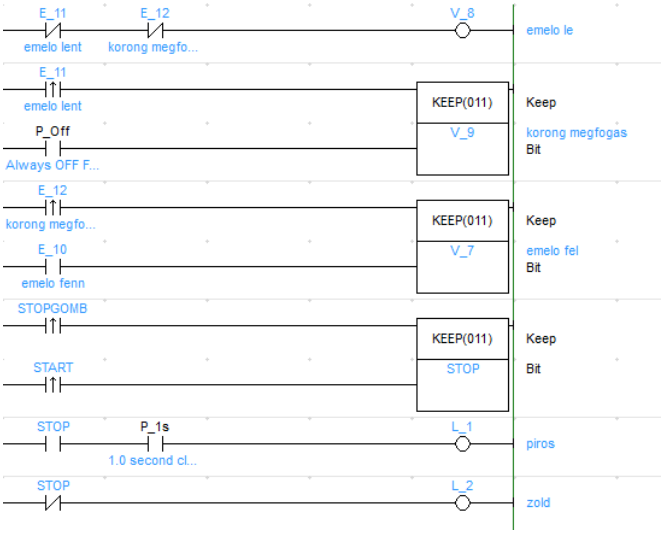


9.23. ábra – : Műveletek a 2-t tartalmazó koronggal és rendezőasztal mozgatása.

A következő lépésben annak érdekében, hogy a 2012 felirat kirakható legyen, a 0-t fel kell emelni, hogy 1-et eltolhassuk a vályúba, hiszen a tárban 01 sorrendben szerepelnek, és mivel a vályúban is erre van szükség, ha csak kitolnánk ezeket a korongokat, akkor a vályúban a 10 számsor jelenne meg, ami nem megfelelő. Ennek megvalósítása látható a 9.24. ábrán.



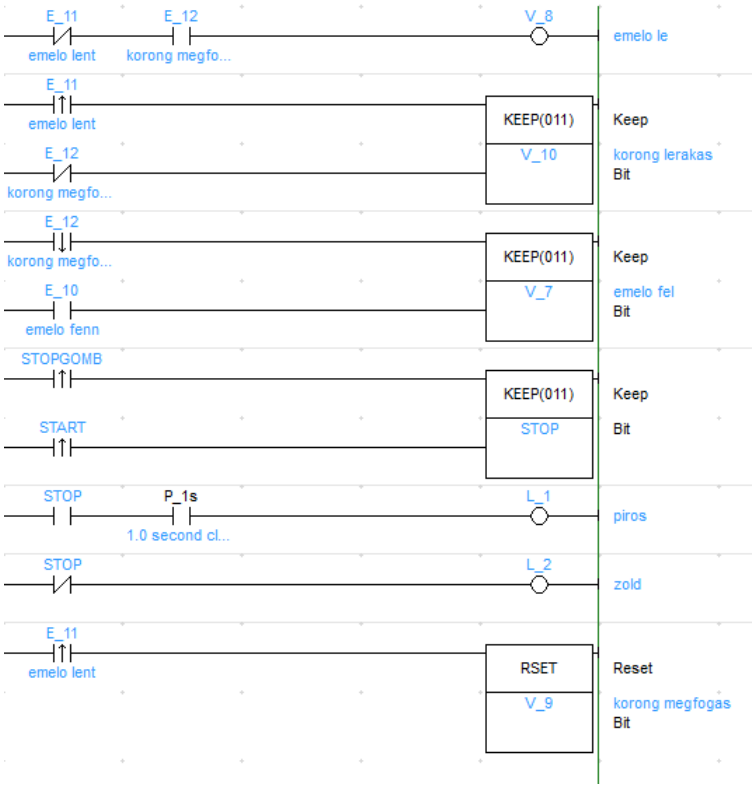
9.24.a. ábra – : Az SFC nyelven írt program egy részlete.



9.24.b. ábra – : Az emelést megvalósító LAD kódrészlet.

A 9.24.b. ábrán látható LAD leírásnak megfelelően, amíg az emelő nincs lent és a korong nincs megfogva, addig mozgassuk lefelé a megfogó részt. Amikor leért (E11 felfutó éle), indítsuk el a korong megfogását. Ameddig a korongot tartani szeretnénk, addig a V9-es szelepet is aktívan tartanunk kell. Lásd az emelő egység leírását, 9.15. ábra. Ezért szerepel a 9.24.b. ábrán a P_Off, mindig hamis (nyitott) érintkező (elég lenne egy SET V_9, de a példa kedvéért bemutattuk a P_Off működését) amely soha nem reseteli ebben az akcióban a megfogást. Mikor megfogtuk, amit az E_12 jelez, elkezdjük felemelni a korongot, amíg fel nem ért a felső véghelyzetébe.

Ezután nincs más dolgunk, beolvasni 1-es, eltenni a vályúba, beolvasni 2-est, letenni 0-t, és a kettőt együtt eltolni. A letevést megvalósító kódrészlet látható a 9.24.c. ábrán.

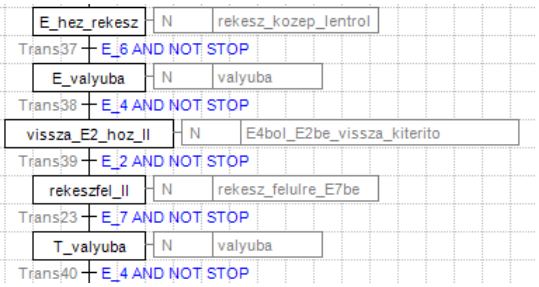


9.24.c. ábra – : A letevést megvalósító LAD kódrészlet.

A 9.24.c. ábrán látható, hogy ha a korong meg van fogva és nincs lent a megfogott korong, akkor engedjük lefelé a megfogót. Ha leért, elengedjük a V9-es szelepet, normál esetben a korong már ilyenkor is leesik, de ha a vákuum valamiért megmaradna, akkor egy kis segédlefúvás érkezik 0,5s-os impulzussal, ezért kell tartani V10-et, miután leért a megfogó. Ha minden rendben, a korong el lett engedve (E12 hamis), indulhat a felemelés immár megfogott korong nélkül. Ha felért a megfogó egység, a következő lépés következik a programban.

Eltoljuk a 2-est tartalmazó korongot, amely a tárból esett ki, mikor az előzőekben a tolólapot visszahúztuk E1-es pozícióba, és a 0-t tartalmazó korongot is eltoljuk, most tettük le az imént. A vályúban már a 2012 felirat szerepel.

Most a rekeszből ki kell vennünk az E-t tartalmazó korongot, eltolni a vályúba, majd a T-t tartalmazó korongot is kivesszük és eltoljuk a vályúba. Az ezt megvalósító SFC kódrészletet láthatjuk a 9.25. ábrán.



9.25. ábra – : E-t és T-t tartalmazó korongokat a vályúba toló SFC kódrészlet.

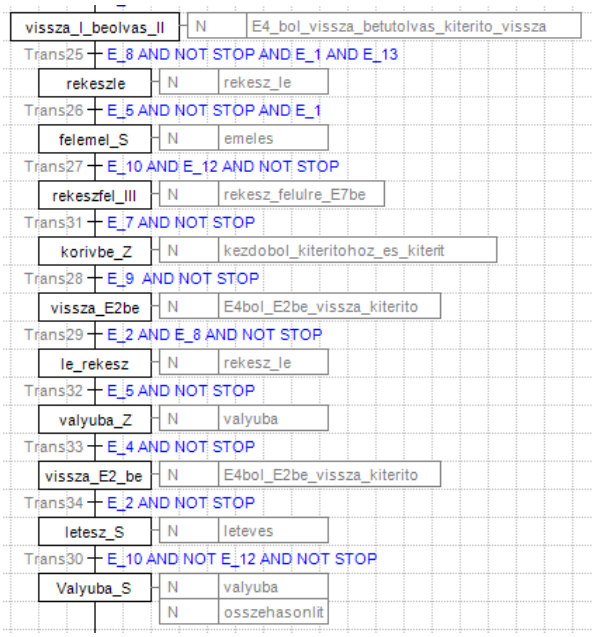
Innentől kezdve azt kell megoldani, hogy hogyan szedjük ki a körívből a benn maradt S-t és Z-t tartalmazó korongokat. Természetes a második, tárban lévő felirat olvasásával. Ha a körívbe újabb elemet lökünk, akkor az kilöki az S-t tartalmazó korongot a rendező legfelső

rekeszébe. Persze arról gondoskodnunk kell, hogy a rekesz (rendezőasztal) olyan pozícióban legyen, amelyből azután el is tudjuk mozgatni az oda belökött korongot. Ez a pozíció az alulról harmadik, vagyis középső tárolóegység. Tehát be kell olvasnunk újra egy S-t tartalmazó korongot, a rekeszt (rendezőasztalt) felső állapotba tolni (9.9. ábra), ide kerül majd a kilökött S betűt tartalmazó korong. A beolvasott S-t tartalmazó korongot eltoljuk az eltérítő elé, ott belökjük az eltérítő egység körívébe, és folytatjuk az algoritmust. (a rendező mozgatása már korábban megtörtént, mert T betűt tartalmazó korong feldolgozása után a rendezőasztal megfelelő pozícióban maradt, lásd a 9.25. ábrát). Az előzőekben tárgyalt funkciókat megvalósító SFC kódrészletet láthatjuk a 9.26. ábrán.

9.26. ábra – : S-t tartalmazó koronggal végzett műveleteket megvalósító SFC kódrészlet.

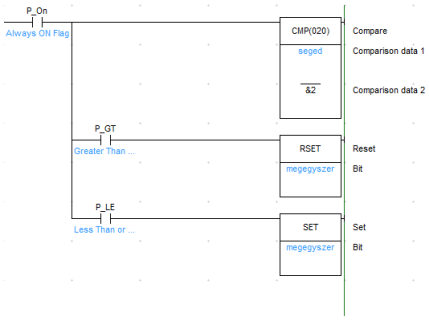
Nyilván valahogy Z-t tartalmazó korongot is ki kell szednünk a körívből, hiszen a vályúban ez kell, hogy előbb szerepeljen, mint az S betű. Ebben az esetben elég lenne csak beolvasni a következő Z-t és eltolni. Azonban ha azt szeretnénk, hogy a program tetszőlegesen sokszor lefuthasson, és a megfelelő szövegrész is számtalanszor kirakhassa, akkor olyan algoritmusra van szükség, amely számtalanszor ismétlődhet. Ennek érdekében azt kell csinálni, hogy a rendezőasztalon lévő S-t tartalmazó korongot felemeljük, majd az új Z-t tartalmazót beletesszük a körívbe. A rendezőasztalt elmozgatjuk alsó véghelyzetébe, eltoljuk a vályúba a Z betűt, majd letesszük S-t tartalmazó korongot vissza a rendezőasztalra. Ezután eltoljuk a vályúba. A vályúban már szerepel is az SZTE2012 felirat. A körívünkben az S, Z betűket tartalmazó korongokat találjuk, a rendezőasztalunk alsó véghelyzetében áll.

Emlékezzünk vissza (lásd a 9.20.a. ábrát), már volt egy ilyen esetünk. Innentől kezdve az algoritmus ismétli önmagát. A végén azért van szükség 2 üres korongra, hogy ezt a kettő korongot, amely az ismétlődéses rész végén a körívben marad, az S-t és a Z-t tartalmazó korongokat, ki tudjuk lökni az utolsó helyes felirat elkészítésének érdekében. A fentebb tárgyalt funkciókat megvalósító SFC kódrészlet a 9.27. ábrán látható.



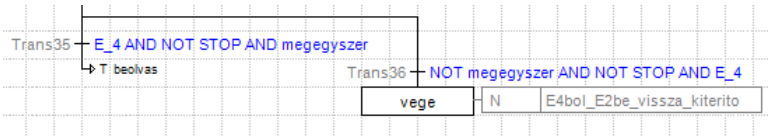
9.27.a. ábra – : A „SZ” szövegrész vályúba helyezését megvalósító SFC kódrészlet.

A 9.27.a. ábrán látható, hogy a végén szerepel egy összehasonlítás akció:



9.27.b. ábra – : Összehasonlítást végző LAD kódrészlet.

A 9.27.b. ábrán lévő kódrészlet ellenőrzi, hogy kiraktuk-e már kétszer a feliratot (ezesetben a seged=3), ha igen, akkor a még egyszer belső változót hamisra, egyébként igazra állítja. Eszerint a program vagy visszalép az fentiekben látott ugráshoz (lásd 9.20.a. ábra), vagy a vege állapotba lép. Ezt láthatjuk a 9.28. ábrán. A vege állapotból a gépnek nincs hová mennie, működése leáll.



9.28. ábra – : A program végén szereplő elágazás.

IRODALOM

[1] Hodossy László: Programozott vezérlések I. Széchenyi István Egyetem, Győr 2006.

[2] Ajtonyi I., Gyuricza I.:Programozható irányítóberendezések, hálózatok és rendszerek, Műszaki Könyvkiadó, Budapest, 2002.

[3] Ajtonyi István: PLC és SCADA-HMI rendszerek, AUT-INFO Kft. Miskolc, 2007

[4] http://webstore.iec.ch/ : IEC-61131-3-Programming Industrial Automation Systems_1.pdf

[5] Jeges Zoltán: "Irányítástechnika" – Jegyzet – Szabadkai műszaki szakfőiskola, Szabadka 2005.

[6] Zoltan Jegeš, Milan Adžić, Robert Marton: "Upravljanje primenom PLC uredjaja" – Jegyzet – Szabadkai műszaki szakfőiskola, Szabadka 2005.

[7] Frank D. Petruzella: "Programmable Logic Controllers", McGraw-Hill 2011

[8] John R. Hackworth, Frederick D. Hackworth: "Programmable Logic Controllers: Programming Methods and Applications", Pearson/Prentice Hall, 2004.

[9] W. Bolton: "Programmable Logic Controllers", 2006, Elsevier Newnes

[10] Wolfgang Mahnke, Stefan-Helmut Leitner, Matthias Damm: "OPC Unified Architecture", Springer, 2009

[11] Ajtonyi István: "Ipari kommunikációs rendszerek", AUT-INFO Kft. Miskolc, 2008

[12] Richard C. Dorf: "The Industrial Electronics Handbook, Industrial communication systems", Edited by Bogdan M. Wilamowski J. David Irwin, 2011 Taylor and Francis Group, LLC