



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

---

# **PLC programozási alapismeretek**

Szerkesztette:  
**Dr. Ferenczi István**

A tananyag elkészítését a „**NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen, az Északkelet-Magyarországi térség felemelkedéséért**” az **EFOP-3.5.1-16-2017-00017** számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.



**EFOP-3.5.1-16-2017-00017**

**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

**SZÉCHENYI**



MAGYARORSZÁG  
KORMÁNYA

**Európai Unió**  
Európai Szociális  
Alap



**BEFEKTETÉS A JÖVŐBE**



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

---

**Szerkesztő:  
Dr. Ferenczi István**

**Szerzők:  
Dr. Ferenczi István**

**Lektor:  
Ferenczi Ildikó**

Kézirat lezárva: 2018.05.20

**ISBN 978-615-5545-78-8**

**Kiadja a Nyíregyházi Egyetem**

# PLC programozási alapismeretek

## I. rész



**2018**



EFOP-3.5.1-16-2017-00017

**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

## TARTALOMJEGYZÉK

TARTALOMJEGYZÉK .....	5
1. BEVEZETÉS, ALAPFOGALMAK .....	7
1.1. A PLC-k felépítése .....	8
1.2. Bemeneti illesztő egység .....	11
1.2.1. Egyenáramú bemeneti csatornák .....	11
1.2.2. Váltakozó áramú bemeneti csatornák .....	12
1.3. Kimeneti illesztő egység .....	13
1.3.1. Relés kimenet .....	13
1.3.2. Egyenáramú tranzisztoros kimenet .....	14
1.3.3. Váltakozó áramú triakos kimenet .....	15
2. A PLC-K ALKALMAZÁSA .....	16
3. PROGRAMOZÁSI ALAPISMERETEK .....	21
3.1. Programozási eljárások .....	23
3.2. Létradiagramos programozás .....	25
3.3. Utasításlistás programozás .....	28
3.4. Az utasítások csoportosítása .....	29
3.4.1. Adatbetöltő utasítások (Load) .....	29
3.4.2. Műveletvégző utasítások .....	29
3.4.3. Tároló és adatmozgató utasítások .....	30
3.4.4. Értékadó és törlő utasítások .....	32
3.4.5. Veremkezelő utasítások .....	32
3.4.6. Ugró utasítások .....	33
3.4.7. Vezérlő utasítások .....	34
4. EGYSZERŰ VEZÉRLÉSEK PROGRAMOZÁSI MÓDSZEREI .....	36
4.1. Logikai kapcsolatok programozása .....	36
4.1.1. ÉS műveletek, AND, ANI utasítások .....	36
4.1.2 VAGY műveletek, OR, ORI utasítások .....	36
4.1.3. MERKER-ek és blokkutasítások használata, ANB és ORB utasítások .....	37
4.1.3. Keresztretesz kapcsolás programozása .....	38
4.1.4. A veremtár utasítások alkalmazása .....	40
4.1.5. Időzítők és számlálók .....	40



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

4.1.6. Bekapcsolási (meghúzási) késleltetés .....	41
4.1.6. Kikapcsolási (elengedési) késleltetés .....	42
4.1.7. Állapotmegőrző időzítők.....	43
4.1.8. Számlálók .....	45
4.1.9. Pergésmentesítés impulzusokkal (impulzusgenerálás) .....	46
4.1.10. A főszabályozó funkció (Master Control) használata .....	47
4.2. Alkalmazások .....	48
5. SORRENDI VEZÉRLÉSEK PROGRAMOZÁSA .....	54
5.1. Operátorok létre hozása, törlése .....	54
5.2. Az STL utasítás .....	55
5.2.1. Alkalmazási példák az STL utasításokhoz.....	55
5.2.2 A lépésvezérlés lefutási vázlata.....	56
5.2.3. A folyamatirányítás bemutatása egy folyamatábrán keresztül.....	57
5.3. A folyamatvezérlés programozása .....	58
5.3.1. Engedélyezett utasítások egy lépésszámszámra belül .....	58
5.3.2. Többszörösen lefedett kimenetek.....	59
5.3.3. A továbbkapcsolási feltétel visszavonási funkciója .....	60
5.3.4. Timer többszörös lefedése.....	61
5.3.5. Továbblépési feltétel impulzusjelre. ....	61
5.4. Egyszerű sorrendi vezérlés programozása .....	62
5.5. STL elágazások .....	63
5.5.1. Egyszerű folyamat.....	63
5.5.2. szelektív elágazás .....	65
5.5.3. párhuzamos elágazás .....	68
5.5.4. Szelektív és párhuzamos elágazások kombinációi.....	71
5.5.5. Üres státusz programozása .....	72
5.5.6. Ugró elágazások .....	74
5.6. Konkrét feladat sorrendi programozásra .....	76
6. GYAKORLÓ FELADATOK .....	79



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

## **1. BEVEZETÉS, ALAPFOGALMAK**

A PLC-k intelligens ipari vezérlő rendszerek, amelyek lehetővé teszik, hogy ugyanazon berendezés (hardver), a betöltött programnak megfelelően akár több vezérlési feladatot is megvalósíthasson. Ez igen fontos szempont, ha arra gondolunk, hogy a jelenlegi piaci igények megkövetelik, hogy egy termék, vagy egy terméket előállító technológia rugalmas legyen, a vevői igényeknek megfelelően. Vagyis ha egy termék előállítási technológiáján változtatni kell, ne kelljen feltétlenül új vezérlőt vásárolni, hanem elég csak a meglévő átprogramozása az új eljárásnak megfelelően. Ez az eljárás, a hagyományos huzalozott vezérlések esetén nem, vagy csak nagyon körülményesen oldható meg. Ezért a korszerű vezérlőrendszerek tervezői egyre inkább az egyre olcsóbbá váló PLC-ket alkalmazzák.

Elnevezésük az angolszász **Programmable Logic Controller** (Programozható Logikai Vezérlők) elnevezésből ered. Szokás még a német irodalom szerint **SPS** (Speicher-Programmierbar Steuerung), vagy **PEAS** (Programmierbar Eingang-Ausgang System)

Az első PLC-ket a hetvenes évek elején a Modicon cég fejlesztette ki az autóipar (General Motors) számára. Központi egységét huzalozott CPU alkotta, 1 kB memóriával, és 128 kimeneti/bemeneti csatornával rendelkezett. Az igazi fejlődés a mikroprocesszorok megjelenésével indult. Előbb a bitprocesszor, majd később a 8 illetve 16 bites processzorok alkották a központi egységet. Egyre több I/O csatornát tudtak kezelni, de növekedett a program-, illetve az operatív tár mérete is. A nyolcvanas évek közepétől megjelentek az olcsó, kompakt felépítésű ún. „mini” PLC-k, elsősorban nem ipari vezérlőrendszerekben. A nyolcvanas évek végétől a nagy ipari gyártósorok PLC-it hálózatba kapcsolták, ezáltal lehetővé vált ezen PLC-k integrálása a számítógépes folyamatirányításba, megvalósult a távfelügyelet vagy akár a távoli (remote) programozási mód is.



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

Kivitelezésük szerint a PLC-ket két nagy csoportra oszthatjuk: - kompakt felépítésű,  
- moduláris szerkezetű.

A kompakt felépítésű PLC-ket elsősorban egyszerűbb vezérlési feladatok végzésére készítették. Előnyük a viszonylag kis méret, egyszerű programozás. Hátrányuk, hogy bemeneti/kimeneti csatornáik száma és tulajdonsága adott, nem módosítható.

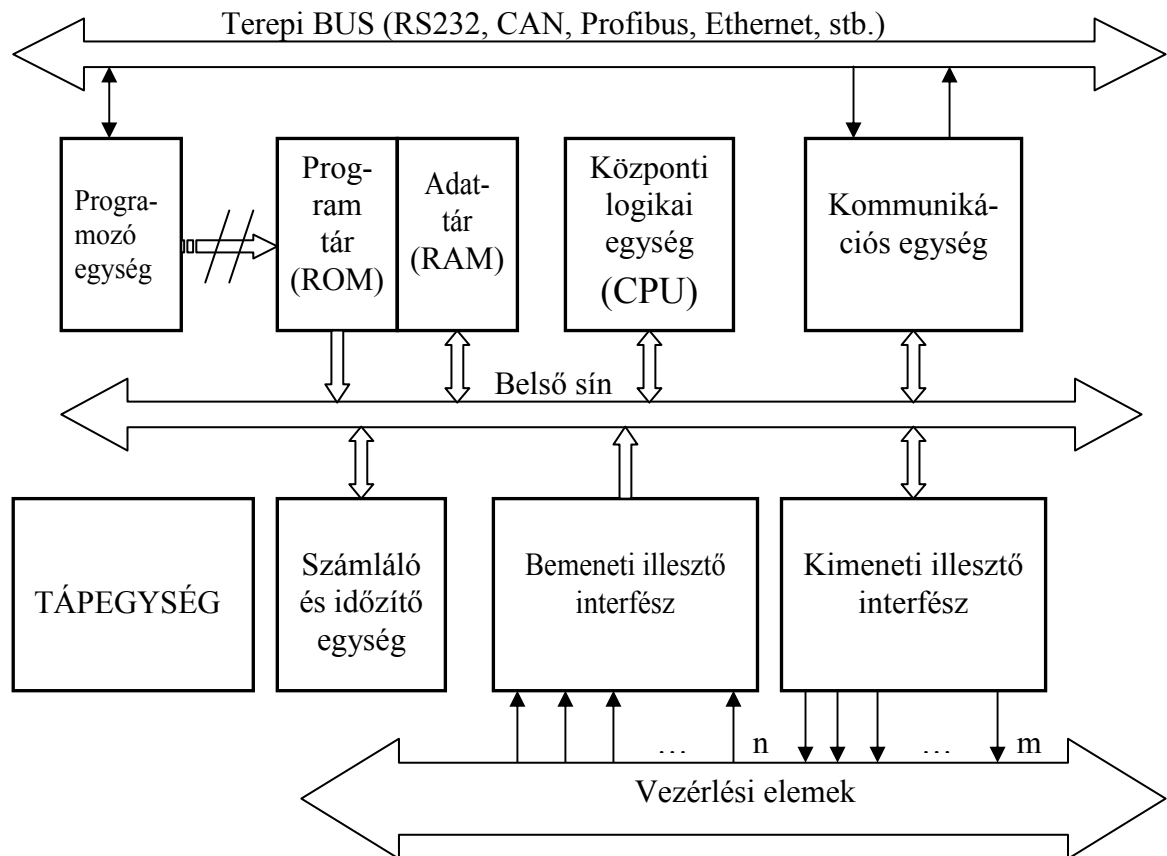
A moduláris szerkezetű PLC-k funkcionális egységei önállóak, modulok formájában kapcsolhatók egymáshoz, így a célnak megfelelően bemeneti/kimeneti csatornáinak száma bármikor módosítható. Ipari gyártósorok, gyártócellák, ipari folyamatirányító rendszerek vezérlési feladataira fejlesztették ki.

### **1.1. A PLC-k felépítése**

Akár kompakt felépítésű, akár moduláris szerkezetű a PLC, funkcionális felépítése hasonló. Tulajdonképpen a PLC egy speciális felépítésű számítógép, ezért leginkább a sínrendszeres megoldás jellemző rá. Fontosabb egységei (1.1. ábra):

- központi logikai egység CPU (Central Processor Unit)
- memóriaegység: - programmemória ( EPROM, EEPROM, FlashROM)  
- adatmemória (RAM)
- bemeneti illesztő egység (Input Unit)
- kimeneti illesztő egység (output Unit)
- kommunikációs egység
- számláló és időzítő egység
- tápegység





**1.1 ábra. A PLC felépítése**

A *központi logikai egység* feladata a programtárba betöltött vezérlési program utasításainak valós időben történő végrehajtása. Ennek érdekében a programvégrehajtás ciklikus működésű, azaz a másodperc töredéke alatt akár többször is lefut a program. A bemenetekre érkező állapotjeleket az utasításoknak megfelelően feldolgozza, majd az eredményeket a kívánt kimeneti csatornákra irányítja. Ezen kívül a CPU előállítja PLC belső funkcionális működését irányító vezérlőjeleket, biztosítja a különböző egységek szinkronizálását.

A *programmemória* tartalmazza egyrészt a PLC működését biztosító rendszerprogramot. Ez a PLC „operációs” rendszere. Ezt rendszerint a gyártó programozza, ennek módosítására nincs szükség, ezért ROM vagy EPROM típusú, csak olvasható tárhelyekbe teszik. Ugyancsak a programmemóriába kerül a felhasználói program. Ennek megvalósítása érdekében biztosítani



**EFOP-3.5.1-16-2017-00017**

**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

kell a kapcsolatot a tár és a programozó egység között. A programozó egység lehet csupán erre a célra kifejlesztett eszköz (ez inkább régebbi megoldás), vagy PC számítógép (laptop), amely valamilyen interfészen keresztül kapcsolódhat a programtárhoz. Betöltés és tesztelés után, ha további módosításokra nincs szükség a programozó egységet el is lehet távolítani. A felhasználói program is csak olvasható memóriába kerül, ez újabban FlashROM, vagy MMC kártya (Micro Memory Card), de kerülhet EPROM-ba is.

Az *adatmemóriába* kerülnek a bemeneti és kimeneti csatornák memória térképei, az utasítás végrehajtáshoz szükséges és a végrehajtás során keletkező átmeneti változók értékei, vagy olyan kimeneti értékek, amelyeket a program futása során több alkalommal is használ.

A *bemeneti* és *kimeneti* illesztők a PLC speciális funkcionális egységei. Ezeken keresztül kapcsolódik a PLC a vezérlési elemekkel. A bemeneti csatornákra bemeneti elemek, nyomógombok, kapcsolók, analóg és digitális érzékelők kerülhetnek. Mindegyik csatornának jól meghatározott címe van, amely alapján a CPU azonosítani tudja. A bemeneti csatornák száma ( $n$ ) rendszerint nagyobb, mint a kimeneti csatornáké ( $m$ ), mert egy vezérlés megvalósításához jóval több bemeneti elemre van szükség, mint beavatkozóra.

A kimeneti csatornákra kapcsolódnak a kimeneti elemek, vagyis a vezérlés beavatkozó szervei: relék, mágneskapcsolók, mágnes szelepek, lámpák, kijelzők stb. Ezek a csatornák is egyedi, jól meghatározott címmel rendelkeznek.

A jelenlegi korszerű PLC-k mindegyike rendelkezik valamilyen *kommunikációs* csatornával is, amelyen keresztül összekapcsolhatók egymással akár a terepi buszokon, akár ipari Ethernet hálózaton keresztül.

A PLC-knek saját, a vezérlés többi részétől független tápellátása van. Ezt egy belső *tápegység* biztosítja, amely speciális zavarssűrítő rendszerrel van ellátva, hogy a néha meglehetősen mostoha ipari körülmények között dolgozó CPU zavartalanul működhessen.

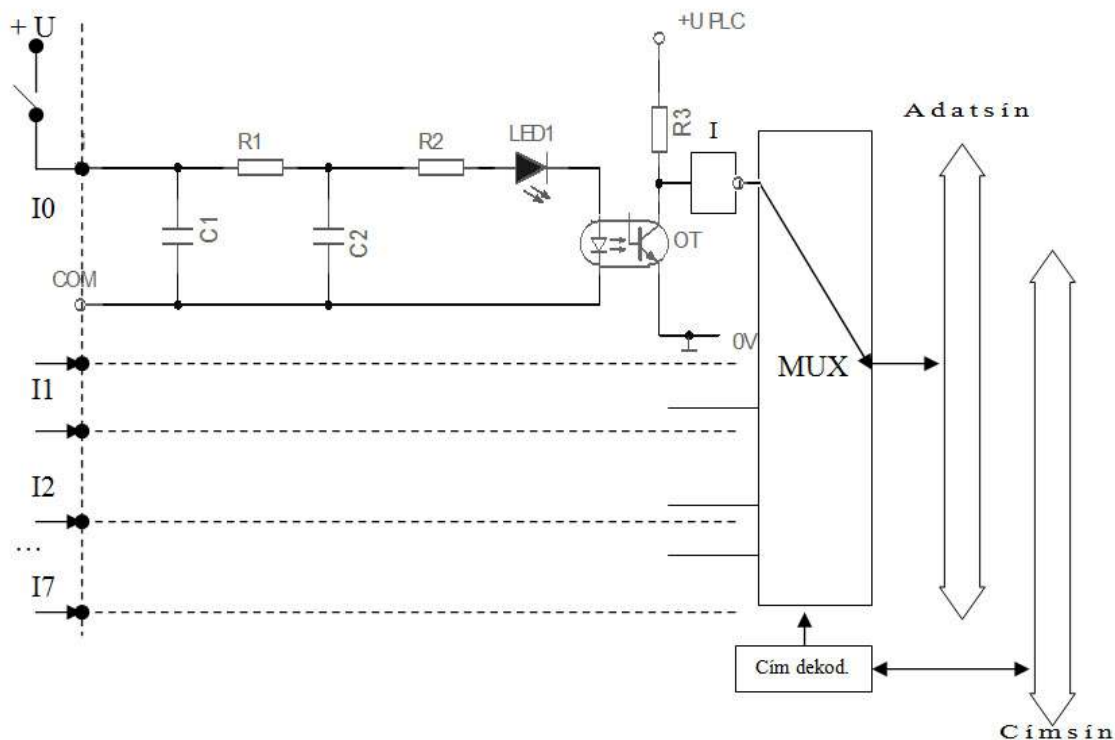
Mivel felhasználói szinten, megfelelő fejlesztő környezet birtokában, nem igazán van szükségünk a CPU működésének részletes ismeretére, annál inkább ismernünk kell a bemeneti és kimeneti illesztőket, mert ezeken keresztül kapcsolódunk a PLC-hez, a továbbiakban csak ezeknek az egységeknek a felépítésével foglalkozunk.

## 1.2. Bemeneti illesztő egység

A vezérlések bemeneti elemei igen sokfélék és változatosak. Ipari környezetben a táplálási feszültségek is többfélék: 230V, 110V, 24V váltakozó feszültség, 24V, 48V egyenfeszültség. Ennek megfelelően a bemeneti illesztők is más-más felépítésűek lehetnek. Ezen kívül számtalan analóg típusú bemeneti elem is van, amelyek speciális, un. analóg bemeneti csatornákat igényelnek. Nagyon fontos, hogy a bemeneti csatornák biztosítsák a PLC galvanikus elválasztását a vezérlés többi részétől. Ez azt jelenti, hogy nincs közös *testpont* a PLC tápfeszültsége és a vezérlési feszültségek között. Ily módon lehet csak megakadályozni, hogy esetleges külső üzemzavar esetén a PLC számára veszélyes, pl. 230V-os feszültség bekerülhessen a PLC-be.

### 1.2.1. Egyenáramú bemeneti csatornák [1]

Az egyenáramú bemeneti csatorna a leggyakrabban előforduló bemeneti típus. Egy ilyen csatorna elvi vázlatát láthatjuk az 1.2. ábrán.



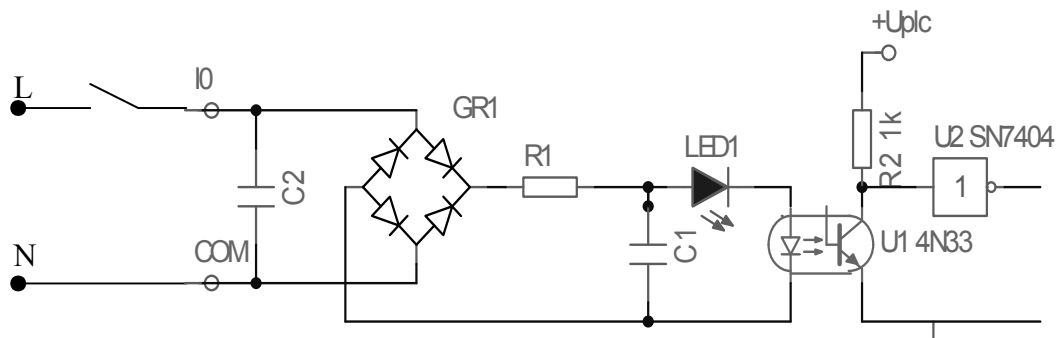
**1.2. ábra.** 8 bemenetű egyenáramú bemeneti egység elvi felépítése

Az R1, C1 és C2 elemek zajszűrő szerepet töltenek be, az R2 ellenállás pedig, a csatornaállapot jelző LED, illetve az optocsatoló áramkorlátját biztosítja. A cím alapján kiválasztott csatorna információja, az optocsatoló kimenetéről, a multiplexeren keresztül jut az adatsínre.

Elvi működés: feltételezzük, hogy a cím alapján az  $I_0$  bemeneti csatorna kerül kiválasztásra. Ha az idekapcsolt bemeneti elem, pl. érintkező nyitott állapotú, akkor az R1, R2, LED1 és optocsatoló diódáján keresztül áram nem halad. A csatoló fototranzisztora blokkolva van, kollektora az R3 ellenálláson keresztül a  $+U_{PLC}$  potenciálra kerül, azaz logikai 1-es, az I inverter után pedig 0 logikai szint kerül a multiplexer megcímzett bemenetére. Ha a bemeneti érintkező zárt állapotba kerül, akkor az R1, R2, LED1 és optocsatoló diódáján keresztül áthaladó áram hatására, a LED1 kigyullad, ezzel jelzi, hogy aktív a bemenet, az optocsatoló tranzisztora kinyit, kollektora 0 V potenciálra kerül, az I inverter után pedig logikai 1-es lesz a MUX megcímzett bemenetén. Láthatjuk, hogy az optocsatoló használatával megvalósul a bemeneten a galvanikus elválasztás is.

### 1.2.2. Váltakozó áramú bemeneti csatornák [1]

A 1.3. ábrán egy váltakozó áramú bemeneti csatorna elvi felépítését látjuk. A különbség lényegében annyi, hogy minden csatorna bemenetét egy hídval egyenirányítják, és megfelelő korlátozó ellenállásokat használnak. Természetesen a bemeneti zavarszűrőt (C2) is másképp kell méretezni, és biztosítani kell az egyenirányított feszültség szűrését is. (C1)



**1.3. ábra.** Váltakozó áramú bemeneti csatorna

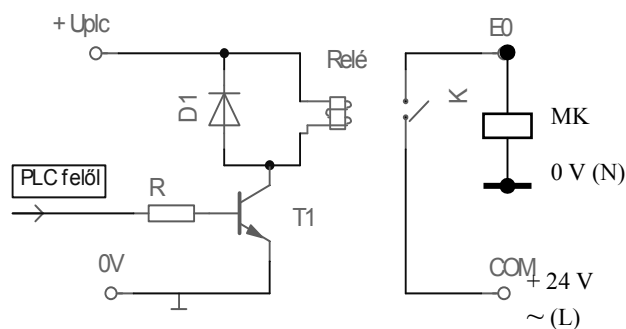
A további működés hasonló a már bemutatott egyenáramú csatornák működéséhez. A közös pont (N) ebben az esetben a váltakozó feszültség nullpontjához csatlakozik.

### 1.3. Kimeneti illesztő egység

A vezérléstechnikában a beavatkozó elemek igen sokfélék. Ennek megfelelően a PLC kimeneti csatornái is többféle megoldást biztosítanak. Alapvetően háromféle megoldást alkalmaznak kimeneti csatornák esetén: relés (univerzális), teljesítmény tranzisztoros (egyenáramú), és triakos (váltakozó áramú). Moduláris felépítésű PLC-k esetén akár mindhárom megoldás alkalmazható egyazon PLC kimeneteiként. A feszültségillesztésen kívül itt is a legfontosabb követelmény a galvanikus elválasztás biztosítása.

#### 1.3.1. Relés kimenet

A legegyszerűbb és leginkább elterjedt kimeneti csatorna. Egyaránt használható egyenáramú és váltakozó áramú beavatkozók esetén is. Mindegyik csatorna kimeneti fokozatára egy-egy REED relé kapcsolódik. Az érintkezők egyik végét még a PLC-n belül összekapcsolják egymással. Ez képezi a kimeneti közös (COM) pontot, melyet rendszerint a külső vezérlő feszültség pozitív potenciáljára, vagy váltakozó feszültség esetén a fázishoz kapcsolják. Az érintkezők másik vége rendre egy-egy kimeneti csatlakozási ponthoz vezet. (1.4. ábra)

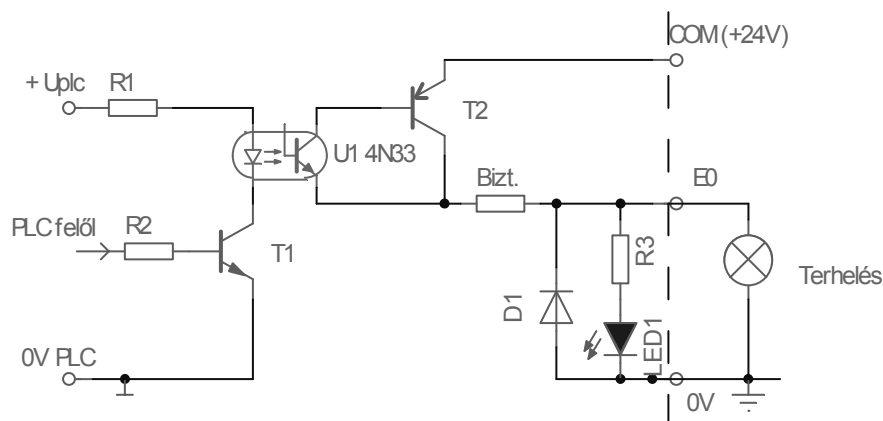


**1.4. ábra.** Relés kimeneti csatorna

A galvanikus elválasztáshoz itt nincs szükség optocsatolóra. Ezt a feladatot ebben az esetben a kimeneti relé biztosítja. Működés: Ha a PLC valamelyik, Ex kimeneti csatornát címzi meg és a PLC felől logikai 1-es érkezik, a T1 tranzisztor kinyit, kollektor potenciálja a 0V test potenciálra kerül, a relé behúzó, K érintkezője pedig zárja az MK, például mágneskapcsoló áramkörét. Ha logikai 0 érkezik a PLC felől, akkor a T1 tranzisztor, blokkolva marad, a relé tekercse nem kap áramot, a K érintkező pedig bontja az MK áramkörét.

### 1.3.2. Egyenáramú tranzisztoros kimenet [1]

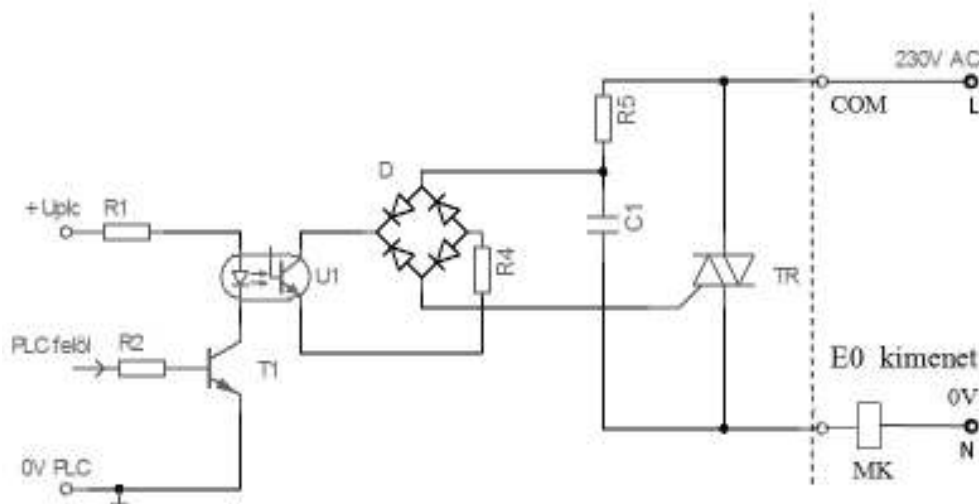
Galvanikus elválasztású, teljesítménytranzisztoros kimeneti fokozatot szemléltet az 1.5. ábra. A fokozat működési elve hasonló a már előzőekben bemutatott relés megoldáshoz. Attól függően, hogy milyen logikai szint érkezik a PLC demultiplexerje felől, a T2 tranzisztor zárja, illetve nyitja a kimenetre kapcsolt, jelen esetben a terhelést jelentő izzó áramkörét. A teljesítménytranzisztor rövidzár védelmére a biztosító látja el. A kimenet állapotának visszajelzésére rendszerint egy LED diódát használnak minden csatornánál. Mivel a kimenetre kapcsolt beavatkozók a legtöbb esetben induktív jellegűek (relé, mágneskapcsoló), fontos, hogy a kikapcsoláskor fellépő autóindukciós feszültséglököt elvezessük. Ezt a feladatot látja el a kimenetre ellenpárhuzamosan kapcsolt D1 dióda. A PLC I/O modulján rendszerint 8, 16 vagy 32 ilyen kimeneti csatornát helyeznek el.



**1.5. ábra.** Tranzisztoros kimeneti csatorna

### 1.3.3. Váltakozó áramú triakos kimenet [1]

Az 1.6. ábrán egy váltakozó áramú kimeneti csatorna (E0) elvi működési vázlatát láthatjuk. Az MK mágneskapcsoló tekercsének áramköre akkor záródik, amikor a PLC felől logikai 1-es érkezik, amely nyitja a T1 tranzisztort. Az optocsatoló kimenetére kapcsolt D híd, valamint az R5, C1 elemekből alkotott áramkör, gyújtja a TR triakot. Amikor a PLC felől logikai 0 érkezik, a tirisztor gyújtása megszűnik, majd a váltakozó feszültség nulla átmenetekor lezár, vagyis megszakítja az MK fogyasztó áramkörét. Mivel a megszakítás 0 volt közelében történik, a kikapcsoláskor fellépő nemkívánatos autóindukciós feszültség is minimális lesz.



**1.6. ábra.** Triakos felépítésű kimenet



EFOP-3.5.1-16-2017-00017

**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

## 2. A PLC-K ALKALMAZÁSA

Az adott vezérlési feladat megtervezésekor fontos lépés a PLC típusának és paramétereinek megválasztása. Abból kell kiindulnunk, hogy hány darab, milyen típusú bemeneti illetve kimeneti vezérlési elemeket kell alkalmazniuk. Figyelembe kell vegyük, hogy kapcsolódik-e a vezérlésünk más vezérlésekhez, vagy folyamatirányító rendszerhez. Olyan helyeken, ahol az alkalmazott technológia gyakran módosul, olyan PLC-t célszerű választani, amelynél a felhasználói program cseréje, nem okoz jelentősebb leállást. Ezen kívül fontos szempont lehet még, a biztonságos (üzembiztos) működés, ahol pedig szükséges a redundancia megvalósítása a 100%-os rendelkezésre állás érdekében.

A PLC kiválasztása után a rendelkezésre álló vezérlési dokumentációk és tervek alapján készíthető el a PLC programspecifikációja, programterve valamint a vezérlési program. A továbbiakban egy egyszerű példán keresztül szeretném bemutatni a PLC alkalmazásával kapcsolatos lépéseket, a tervezéstől egészen a felhasználói program elkészítéséig.

A meglehetősen egyszerű vezérlési feladat egy háromfázisú aszinkronmotor működtetésének és túláramvédelmének PLC-s vezérlése. A feladatból az is kiderül, hogy milyen egyszerűen lehet a vezérlést megváltoztatni, a huzalozott vezérléssel szemben.

Lássuk, milyen vezérlési elemekre van szükségünk:

- *bemeneti elemek:* - nyomógombok: - START



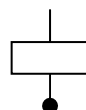
- STOP



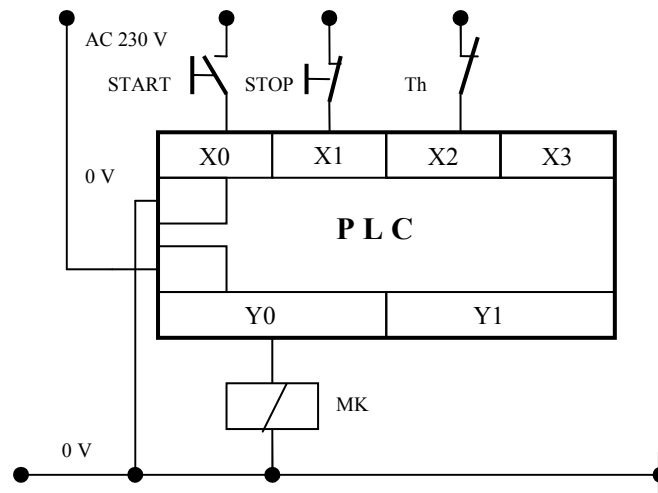
- hőrelé érintkezője: Th



- *kimeneti elem:* - mágneskapcsoló: MK







**2.1. ábra.** A PLC bekötési vázlata

Mivel a motor táplálása 3 fázisú feszültségről történik, célszerű a vezérlési feszültségként is ezt a feszültséget használni. Ezért olyan PLC-t választunk, amelynek bemeneti és kimeneti csatornái váltakozó áramúak, és legkevesebb 3 bemenete és 1 kimenete van. Mivel az IO csatornák száma, általában kettőnek a hatványa szokott lenni, ezért egy 4/2-es, AC 230V-os, kompakt PLC-t választunk. (2.1. ábra.)

A továbbiakban meg kell terveznünk a vezérlési programot. Ehhez szükséges a vezérlési állapotegyenleteknek a felírása. Minden kimeneti beavatkozó elemhez hozzá kell rendelni a működési feltételeket biztosító logikai összefüggéseket. A mi esetünkben ez a következőképpen hangzik: A motornak akkor kell működnie, amikor a START gombot megnyomjuk, nincs a STOP gomb benyomva és a hőrelé nincs működtetve. Azért, hogy a motor ne csak addig működjön, míg a START gombot nyomjuk, öntartást kell alkalmaznunk. A fentiek alapján felírhatjuk a következő vezérlési állapotegyenletet:

$$MK = (START + MK) \cdot \overline{STOP} \cdot \overline{Th} \quad (1)$$

A 2.1. ábrán láthatjuk, hogy a PLC-hez a vezérlési elemek jól meghatározott címekkel rendelkező IO portokhoz csatlakoznak, vagyis hozzárendeltünk minden egyes vezérlési elemhez egy-egy bemeneti illetve kimeneti címet.

START → X0

STOP → X1

Th → X2

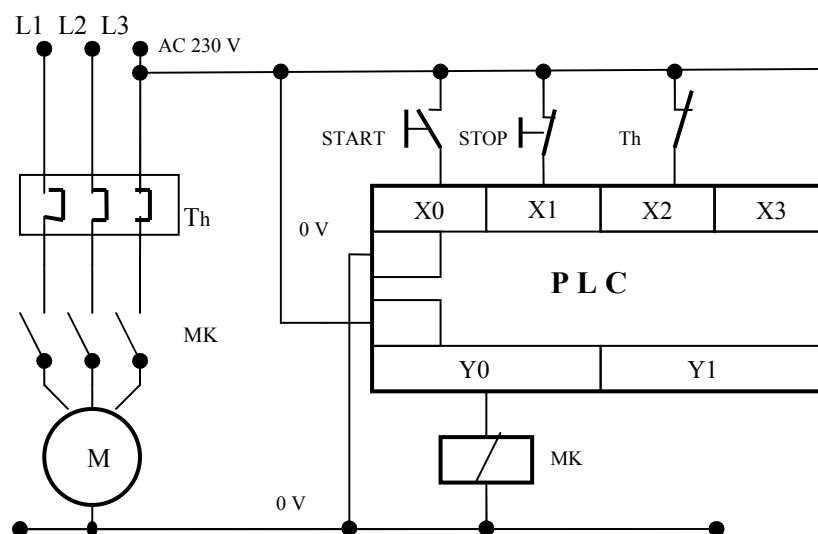
MK → Y0

Ennek megfelelően az előbbi összefüggés a következő módon is felírható:

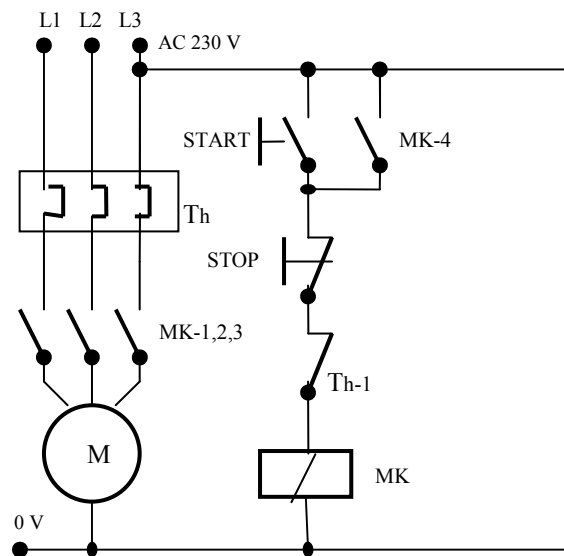
$$Y0 = (X0 + Y0) \cdot \overline{X1} \cdot \overline{X2} \quad (2)$$

A fenti összefüggést már csak valamilyen programozási módszerrel kell leprogramoznunk és betöltenünk a PLC programtárába.

Mielőtt azonban rátérnénk a programozási módszerek és eljárások ismertetésére, hasonlítsuk össze a PLC-s vezérlést a hagyományos huzalozott relés vezérléssel. A 2.2. ábra a teljes motorvezérlés PLC-s változatát látjuk, a 2.3. ábra pedig a huzalozott vezérlésű megoldást mutatja. Ennél az egyszerű vezérlésnél első ránézésre azt mondhatjuk, hogy semmivel sem lett egyszerűbb a kapcsolás. A teljesítmény rész változatlan marad, a motor táplálását továbbra is a mágneskapcsoló érintkezői biztosítják. Ugyanúgy megmaradtak a bemeneti elemek (START, STOP, hőrelé) és az MK beavatkozó is. Sőt még egy PLC-vel is bonyolódott a dolog, amit ráadásul még programozni is kell.



**2.2. ábra.** 3 fázisú aszinkronmotor működtetése PLC-s vezérléssel



**2.3. ábra.** 3 fázisú aszinkronmotor működtetése huzalozott vezérléssel.

Ilyen egyszerű vezérlésnél a PLC igazi előnyei nem teljesen nyilvánvalóak. Viszont ha egy bonyolultabb vezérlésre gondolunk, ahol nagyon sok, esetleg több száz vezérlési elemet kellene a működési logikák (vezérlési állapotegyenletek) alapján összehuzalozni, akkor már sokkal egyszerűbb dolgunk van, mert ezeket az elemeket, rendeltetésüktől függően csak a PLC bemeneti, illetve kimeneti pontjaihoz kell csatlakoztatnunk, és természetesen el kell készítenünk a vezérlési programot.

Az igazi előny azonban akkor mutatkozik meg, amikor a vezérlésben valamilyen módosítást kell alkalmaznunk, mert esetleg így kívánja meg a technológia. Az előbbi példánál maradva, tételezzük fel, hogy ez a motor egy szerszámgép főtengelyét hajtja. A munkadarab befogása hidraulikusan történik. Nem lenne szerencsés, ha véletlenül elindulna a forgás, még mielőtt a munkadarab beszorításra kerülne. Feltételezzük, hogy ezt a feladatot egy nyomásérzékelő végzi. Ez egy újabb bemeneti elemet jelent. Ha már az indítás feltételhez kötött, jó lenne tudni, hogy mikor teljesül a feltétel, vagyis valamilyen visszajelzést is szeretnénk. Ez még egy vezérlési elemet jelent, ezúttal egy beavatkozót, vagyis egy jelzőlámpát.

A fenti, viszonylag egyszerű módosítás elvégzése huzalozott vezérlés esetén nem is olyan könnyű feladat. A motorműködésre felírt vezérlési állapotegyenlet, az új feltételt bevezetve a következőképpen módosul:

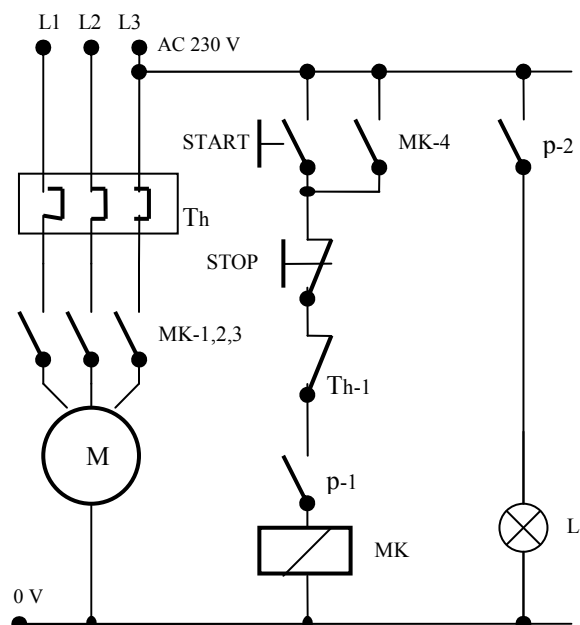
$$MK = (START + MK) \cdot \overline{STOP} \cdot \overline{Th} \cdot p \quad (3)$$

ahol  $p$ -vel a nyomásérzékelőt jelöltük. Hogy ezt megvalósíthassuk, meg kell szakítanunk a huzalozást valahol a hőrelé érintkezője és a mágneskapcsoló tekercse között, és bekössük sorosan a nyomásérzékelőt. (2.4. ábra)

A visszajelzés révén egy új kimenetünk is lesz, legyen ez az  $L$  lámpa, ennek is fel kell írunk az egyenletét. Mivel ez a nyomást hivatott jelezni ezért:

$$L = p$$

A megvalósításhoz a lámpával sorba kell kötni, ugyancsak a  $p$  érzékelő érintkezőjét, amely újabb huzalozást jelent.



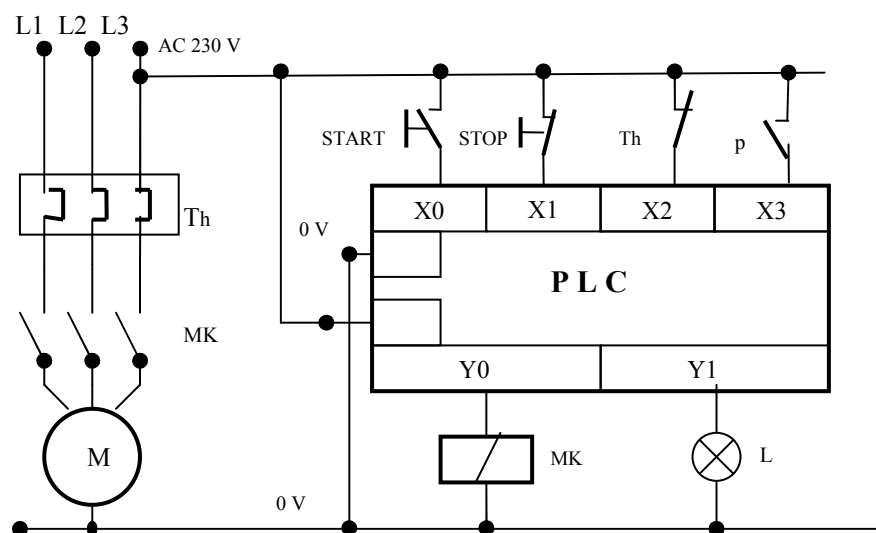
**2.4. ábra.** A módosított huzalozott vezérlés

Nyilvánvaló, hogy ha egy bonyolultabb vezérlést kell átalakítani, akkor sokkal több helyen kell újravezetkezni az áramköröket, amely sok esetben gyakorlatilag megvalósíthatatlan.

Ezzel szemben a PLC-s vezérlésnél jóval egyszerűbb a feladatunk. A *p* nyomásérzékelőt bekötjük egy szabad bemeneti csatornára (a mi esetünkben ez X3), az L lámpát pedig az Y1 kimeneti csatornára. Újraprogramozzuk a PLC-t az új vezérlésnek megfelelően:

$$\begin{aligned} Y0 &= (X0 + Y0) \cdot \overline{X1} \cdot \overline{X2} \cdot X3 \\ Y1 &= X3 \end{aligned} \quad (4)$$

A 2.5. ábrán ugyanannak a vezérlésmódosításnak a PLC-s változatát látjuk.



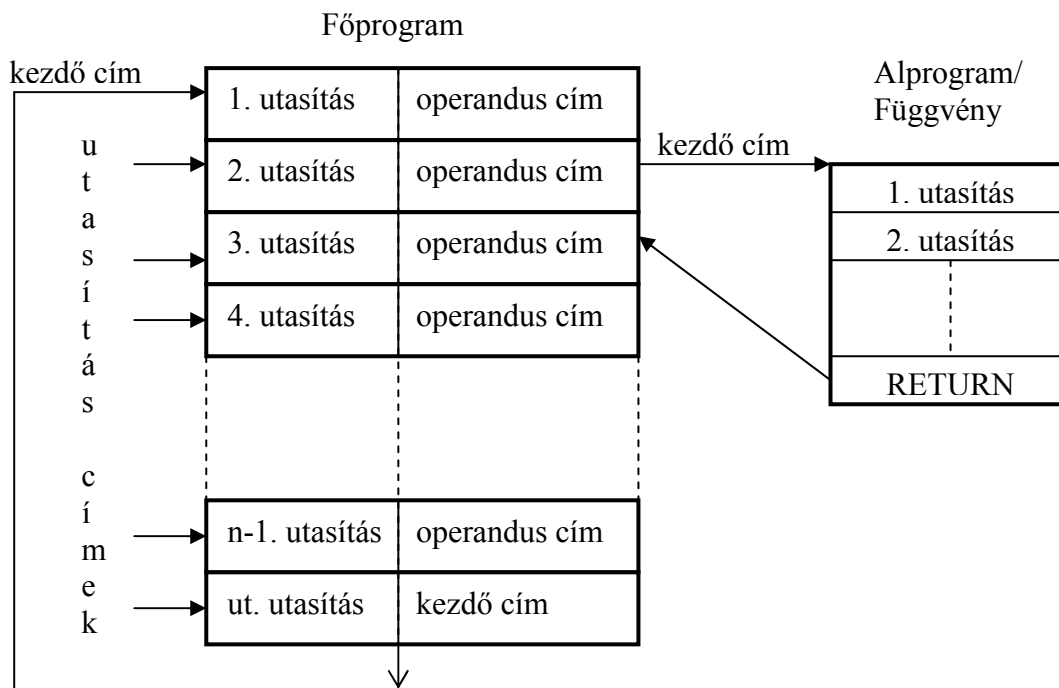
**2.5. ábra.** A módosított PLC-s vezérlés

### 3. PROGRAMOZÁSI ALAPISMERETEK

A vezérlési feladatok ellátására a PLC csak a felhasználói program elkészítése és betöltése után alkalmas. Míg a bitprocesszoros PLC-k vezérlési algoritmusának leprogramozását kizárólag, csak a felhasználói program biztosította, a mai korszerű mikroprocesszoros PLC-k alapszoftvere számos olyan funkciót tartalmaz, amelyeket alkalmazni lehet a felhasználói

program készítésekor. Ez nagymértékben megkönnyíti, főleg a bonyolult programrészek elkészítését.

Szerkezetileg a PLC program, akár csak bármilyen program a főprogramra épül. A főprogramon belül meghívhatók az alprogramok, illetve a függvények, vagy egyéb, a rendszerprogram által felkínált funkcióblokkok. Ezek használata nem kötelező, de a bonyolultabb programok készítését nagymértékben egyszerűbbé és áttekinthetőbbé teszi. A programok és alprogramok, valamint a függvények utasítások sorozatából állnak, melyeknek végrehajtása rendszerint időben egymásután történik, ha csak a programozó másképp nem rendelkezik. A programon belül minden utasítás a memóriában jól meghatározott címmel rendelkezik. Az alprogramok és függvények esetén ismernünk kell a kezdőcímet azért, hogy adott esetben hivatkozni tudjunk rájuk. (3.1. ábra.)



**3.1. ábra.** PLC program felépítése és elhelyezkedése a memóriában bitprocesszoros PLC-nél



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

Bájt szervezésű processzorok esetén a program elhelyezkedése a memóriában nagymértékben függ a rendszerprogramtól ezért az előbbitől eltérő lehet.

Betöltés után a program futása ciklikusan történik. A ciklusidő változó lehet. Ez nagymértékben függ a PLC processzorától, a program terjedelmétől és természetesen felhasznált bemeneti és kimeneti csatornák számától. Ahhoz, hogy a valós idejű feldolgozást biztosítani tudjuk, egyszerűbb vezérléseknél a ciklusidő rendszerint kisebb, mint 10 ms, de bizonyos esetekben ennél még kevesebb, akár 1 ms-nál kisebb is lehet.

Az utasítások szerkezetét tekintve, két fontosabb részt különböztetünk meg: műveleti rész (OPKÓD) és címrész (CÍM)

OPKÓD	CÍM
-------	-----

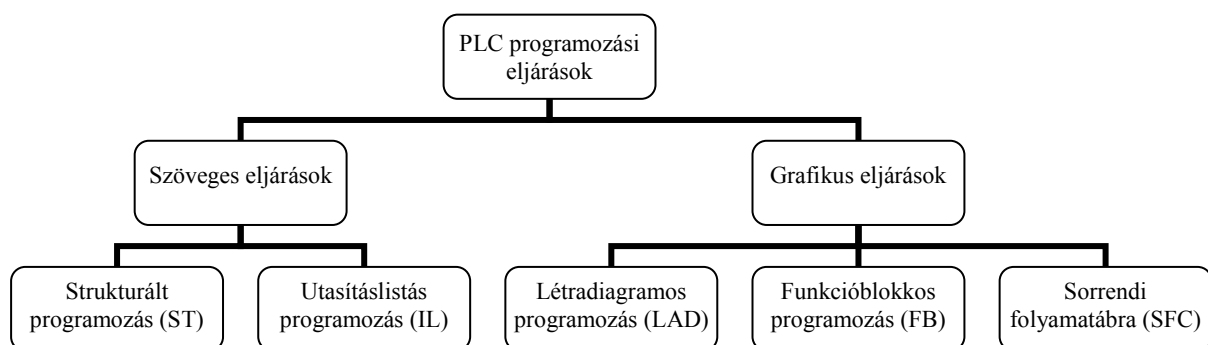
Az OPKÓD leírja, hogy milyen jellegű műveletet kell elvégezni, a CÍM, pedig megmutatja, hogy hol található az operandus, amivel a műveletet el kell végezni. A cím lehet I/O cím, memória cím, de lehet regiszter címe is. Az OPKÓD részt minden utasításnak tartalmaznia kell. A CÍM rész hiányozhat, ugyanis vannak olyan utasítások, amelyek nem operandusokra vonatkoznak, de vannak olyanok is, amelyek kettő vagy annál több címet tartalmaznak. Éppen ezért, az utasítás bonyolultságától függően, a memóriában elfoglalt helyük is lehet egy vagy több bájt hosszú. Például az A103H címen lévő 2 bájt hosszú utasítás, második bájtja elfoglalja az A104H memóriarekeszt is, így a soron következő utasítás címe A105H lesz.

### **3.1. Programozási eljárások**

Fejlődésük során a PLC-k, programozásukat illetően számos változáson mentek keresztül. A kezdeti szakaszban a programozás szinte kizárólag csak a programozó eszközről történt, amelyek legfeljebb csak a PLC gépi utasításainak a bevitelére voltak alkalmasak. Ez meglehetősen hosszú és fáradságos munkát igényelt a programozók részéről. Nagy előrelépést jelentett a programozásban a PC számítógépek elterjedése. Ez már lehetővé tette, megfelelő fordítóprogram birtokában, hogy a PLC programok valamilyen fejlettebb programozási

nyelven íródjanak. (Pascal, C, Basic stb.) Egy újabb lépés akkor következett, amikor megjelentek a grafikus felületű operációs rendszerek. Innentől kezdve jelentősen egyszerűsödtek és áttekinthetőbbé váltak a programozási eljárások.

Egy másik fejlődésbeli probléma az volt, hogy ezek a programozási eljárások gyártóspecifikusan alakultak ki, és fejlődtek mind a mai napig. Történtek ugyan szabványosítási kísérletek, meg is jelent az IEC 61131-3 szabvány, de ennek ellenére egy Modicon PLC-hez készített vezérlési program nem alkalmazható például egy Siemens, vagy egy Mitsubishi PLC esetén, de a szabvány segítségével lehetőség nyílt a különböző PLC programozási eljárásokat bizonyos mértékig kategorizálni és adott keretek közzé szorítani. Az IEC 61131-3 szabványnak megfelelően, a PLC programozási eljárásokat az alábbi struktúra szerint csoportosíthatjuk (3.2. ábra):



**3.2 ábra.** PLC programozási eljárások

Hogy mikor melyik programozási eljárást érdemes választani, az mindig attól függ, hogy milyen jellegű vezérlést szeretnénk PLC-re vinni, és természetesen attól is, hogy milyen fejlesztőkörnyezetet használunk. Kezdetben, amikor még az adott PLC-hez készített eszközről történt a programozás, ez a legtöbb esetben az utasításlistás (IL) mód volt. Az eljárás a PLC saját utasításkészletét használta. A programozási mód nagyon hasonlított az assembly programozási nyelvhez.

A másik, szöveges jellegű programozási eljárás akkor kezdett kialakulni, amikor a PC számítógépek megjelentek és számos fejlesztői környezet állt a programozók részére. Az eljárás lényege az, hogy valamilyen fejlett struktúrájú programozási nyelven (pl. PASCAL,





**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

BASIC vagy C) készítjük el a vezérlési programot, majd egy az adott PLC-hez készített fordítóprogram (compiler) segítségével átkonvertáljuk a PLC nyelvezetére. Az eljárás célja az volt, hogy a közismert programozási nyelveket ismerő fejlesztők képesek legyenek PLC programokat is készíteni. Ennek ellenére ez a módszer nem igazán terjedt el a PLC programozásban. A módszer hatékonysága természetesen attól függött, hogy mennyire jó fordítóprogramot sikerült előzőleg kifejleszteni az adott PLC-hez.

A grafikus eljárások közül a létradiagram volt az első, és talán a legelterjedtebb eljárás, amelyet elsősorban a bitprocesszor alapú PLC-nél alkalmaztak, de jelenleg is elterjedten használják a mai, fejlettebb PLC-knél is, főleg digitális jellegű vezérlési elemek esetén.

A funkcióblokkos programozás a huzalozott logikában használt szimbólumokból kialakított, meglehetősen hardverorientált nyelv. Egy funkcióblokk bal oldalán a bemenetek, jobb oldalán a kimenetek vannak feltüntetve. A jelfolyam iránya az előző fokozat kimenetétől a következő fokozat bemenete felé halad, a közöttük lévő logikai vagy funkcionális kapcsolat pedig a megcélzott kimeneti elem vezérlési állapotfunkcióját írja le.

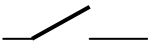
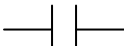


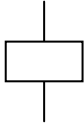
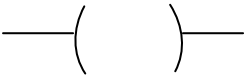
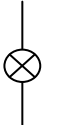

A sorrendi vezérlési eljárást leginkább a lefutó vezérlések programozásánál előnyös, ahol egy adott vezérlési szekvencia csak akkor indulhat, amikor az előző befejeződött.

A fenti eljárások közül, a továbbiakban a létradiagramos, az utasításlistás és sorrendi programozási eljárásokat ismertetjük, Mitsubishi FX típusú PLC-k esetén.

### **3.2. Létradiagramos programozás**

A létradiagram készítés elve már akkor megjelent, amikor még nem is léteztek a grafikus felületű fejlesztőkörnyezetek. A PLC-k megjelenésével sok esetben, a már meglévő relés, huzalozott vezérlési technikákat szerették volna PLC-sre cserélni. A szakemberek számára az adott berendezés vezérlésről csak az áramút-terv, a mechanikai és a villamos dokumentációk álltak rendelkezésre. Az áramút-terv a tápfeszültség két vezetéke között, időrendi sorrendben balról jobbra, több oldalon át elhelyezett vezérlési elemek közötti logikai kapcsolatok és áramköri rajzok összességét jelenti. Egy teljes vezérlés akár több száz oldalon keresztül tartalmazza az áramút-tervet. PLC programozás szempontjából a logikai felépítés úgy tűnik

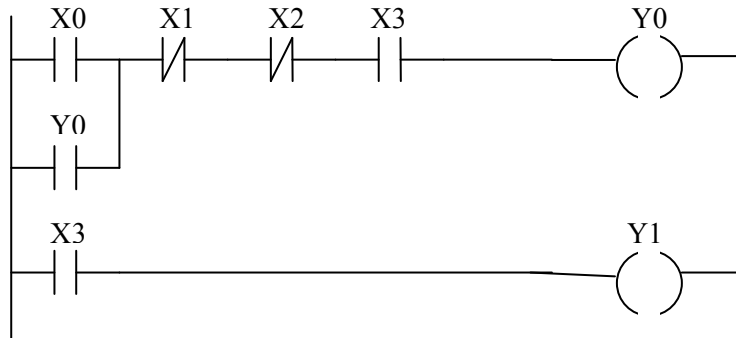
célszerűnek, hogy ha azok fentről lefelé következnek egymás után, úgy ahogy az utasítások is követik egymást. Így született meg a gondolat, hogy mi lenne, ha 90 fokkal balra elfordítanánk a teljes áramút-tervet, a vezérlési elemeket pedig egyszerűbb jelölésekkel látták el. Az alábbi táblázatban néhány fontosabb létradiagram szimbólumot mutatunk be.

Áramút-terv jelölés	Létradiagram szimbólum	Leírás
		Záró (normál nyitott állapotú) érintkező
		Bontó (normál zárt állapotú) érintkező
		Általános kimeneti elem. (relé, mágneskapcsoló, mágnesszelep tekercs, kijelző, időzítő, számláló)
		Speciális kimeneti funkció (SET, RESET, PULS, stb.)

**1. táblázat.** Gyakoribb létradiagram szimbólumok

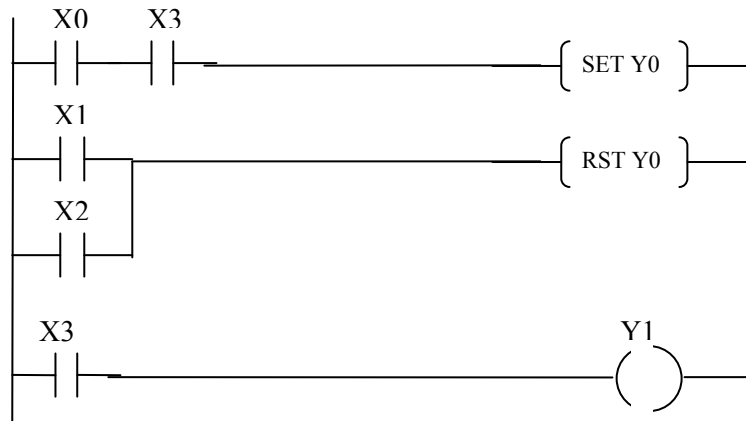
A létradiagramok a szimbólumokon kívül tartalmazzák még a huzalozást, valamint a jelöléseket, amelyek lehetnek I/O címek, memóriacímek (MERKER), regisztercímek, időzítők és számlálók címei. A létrának gyakorlatilag annyi „foka” lesz, ahány kimeneti elem, MERKER, időzítő, számláló, vagy regiszter van definiálva a program során, vagyis ahány vezérlési állapotegyenletet felírtunk.

A továbbiakban vizsgáljuk meg egy egyszerű, a már előzőekben ismerttetett vezérlés, létradiagramjának elkészítését. Induljunk ki a 2.4. ábra áramút-tervéből és alkalmazzuk a 1. táblázat szimbólumait, hozzárendelve a bekötés alapján (2.5. ábra) hozzárendelt I/O címeket.



**3.3. ábra.** A motorvezérlés létradiagramja öntartással

Az előbbi megoldás teljes egészében a huzalozott logikára építhető. A PLC adta lehetőségek ettől eltérő megoldást is támogatnak, amelyek ugyanarra az eredményre vezetnek. Az előbbi feladatot leprogramozhatjuk a következőképpen is: (3.4. ábra)



**3.4. ábra.** A motorvezérlés létradiagramja SET, RST utasítással

Ennél a megoldásnál öntartást nem kell biztosítanunk, mert a SET utasítás az Y0 kimenetet mindaddig logikai 1-sen tartja, amíg a RST utasítás feltételei nem teljesülnek, vagyis a STOP illetve hőrelé nem kapcsol. Ha a STOP gombot működtetjük, a RST utasítás, nullára állítja az Y0 kimenetet, mindaddig, amíg újra nem nyomjuk a START gombot. Amennyiben a leállás a hőrelé miatt következik be, a RST az X2-ön keresztül mindaddig fennmarad, ameddig a hiba



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

el nem járul. Ez alatt hiába nyomjuk a START gombot, a RESET nem oldódik fel, indítani nem tudunk.

Mindkét megoldás ugyanazt az eredményt adja. Hogy mikor melyiket használjuk, érdemes az adott helyzethez kiválasztani. A második esetben a motorműködtetési feladatot megosztottuk, indításra és megállításra, így összesen három kimeneti állapotot kaptunk, míg az első esetben egyetlen egy állapotba volt foglalva az indítás és megállítás is.

### **3.3. Utasításlistás programozás**

A létradiagramos programozás esetén, ugyan elkészítjük a PLC programot a grafikus felületen, a program betöltése és végrehajtása a PLC utasítások feldolgozása szerint fog történni. Vagyis a fejlesztő környezet a létradiagram alapján „legyártja” az utasításlistát. Ez kerül majd be a programtárba, és az itt lévő utasítások kerülnek feldolgozásra.

Minden utasítás feldolgozása alapvetően két jól meghatározható fázisra tagolható:

- utasítás lehívás (FETCH)
- utasítás végrehajtás (EXECUTION)

A lehívási fázisban, a programszámláló regiszter tartalma alapján a megcímzett utasítás az adatsínen keresztül bekerül a processzor utasításregiszterébe. A végrehajtási fázisban, az utasítás dekódoló értelmezi az utasítás műveleti részét (OPKÓD). A vezérlő egység ennek alapján előállítja a végrehajtáshoz szükséges vezérlőjeleket. Ezzel egy időben a címrész alapján kiválasztja az operandust. Az eredmény rendszerint az ACCUMULATOR regiszterbe kerül, és az állapotregiszter is az eredmény szerinti állapotra áll. Eközben a programszámláló inkrementál és lehívja a soron következő utasítást. Egy teljes utasítás feldolgozása a két ciklus során akár három, négy vagy több gépi ciklusidőt igényel, az utasítás bonyolultságától függően.



EFOP-3.5.1-16-2017-00017

**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

### 3.4. Az utasítások csoportosítása

#### 3.4.1. Adatbetöltő utasítások (Load)

Műveletvégzés szempontjából a processzoron belül az egyik legfontosabb regiszter az ACCUMULATOR (ACC). Ebbe a regiszterbe kerül mindig a műveletvégzésre váró egyik operandus, majd a művelet elvégzése után az eredményt tárolja. Főleg a régebbi PLC-knél, fontos tehát, hogy a konkrét műveletvégzés előtt az operandust betöltsük az ACC regiszterbe. Az utasítás általános formája:

##### LD cím

Ahol LD a műveleti kód, a cím pedig megmondja, hogy honnan szeretnénk adatot betölteni. A cím lehet tehát I/O csatorna vagy memória cím. Az utasítás hatására az adott címen lévő érték bekerül az ACC regiszterbe.

Példa, Mitsubishi FX PLC esetén:

**LD X0** ; X0[ ] → ACC (az X0 címen lévő adat bekerül az akkumulátorba.)

Bitcímzésnél a kiválasztott memória területet MERKER-eknek hívjuk, és M0, M1, M2 ... címeken érhetők el. (A PLC-től függ, hogy hány ilyen MERKER-t használhatunk). Ebben az esetben a cím egy memória cím lesz.

**LD M0** ; M0[ ] → ACC (az M0 címen lévő adat bekerül az akkumulátorba.)

Lehetőség van a megcímzett adat inverz betöltésére is. Ebben az esetben a műveleti kód LD helyett LDI lesz:

**LDI X0** X0[ ] → ACC (az X0 címen lévő adat negáltja bekerül az akkumulátorba.)

(A továbbiakban bemutatott utasítások is a Mitsubishi FX PLC utasításkészletébe tartoznak, de hasonlóak az utasítások a többi PLC esetében is)

#### 3.4.2. Műveletvégző utasítások

A műveletvégző utasítások közül elsősorban a logikai műveletvégzést emelném ki, mint például az ÉS, illetve a VAGY műveleteket. Az utasítás formátuma itt is OPKÓD és címrészből áll. A cím utalhat itt is I/O vagy memória címre.



**EFOP-3.5.1-16-2017-00017**

**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

ÉS logika esetén:

**AND** cím      jelentése: az adott címen lévő adat és az akkumulátor tartalma között  
ÉS logikai műveletet végez, az eredmény az akkumulátorba kerül.

Vagy:

**ANI** cím      amikor a címen lévő adat negáltja és az ACC tartalma között végez ÉS  
műveletet, az eredmény szintén az ACC-ba kerül.

Példa:

**AND** X1      ; X1[ ] · ACC → ACC

**ANI** X2      ; X2[ ] · ACC → ACC

Logikai VAGY művelet esetén:

**OR** cím      jelentése: Az adott címen lévő adat és az akkumulátor tartalma között  
VAGY logikai műveletet végez, az eredmény az akkumulátorba kerül.

Illetve:

**ORI** cím      amikor a címen lévő adat negáltja és az ACC tartalma között végez  
VAGY műveletet, az eredmény szintén az ACC-ba kerül.

Példa:

**OR** X1      ; X1[ ] + ACC → ACC

**ANI** X2      ; X2[ ] + ACC → ACC

*Figyelem!* A „·” és a „+” műveleti jelek nem szorzást vagy összeadást jelentenek, hanem ÉS  
és VAGY logikai műveleteket!

A PLC-k műveletvégző egysége, aritmetikai műveletek végzésére is alkalmas. Ilyenkor  
rendszerint nem bit, hanem bájt műveleteket definiálunk, adatainkat pedig a bájt műveletekre  
fenntartott adatregiszterekben (D0, D1, D2,...) helyezzük el. Ezekkel majd a II. részben  
foglalkozunk.

### **3.4.3. Tároló és adatmozgató utasítások**

Az eddigiek során láthattuk, hogy a keletkezett eredmény mindig az akkumulátorban van.  
Természetesen addig nem tudunk, új művelethez kezdeni (újabb adatot betölteni) amíg az



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

akkumulátor tartalmát el nem tároljuk valahová. A tárolás történhet kimeneti címre vagy memóriába. Az utasítás általános formája:

**OUT** cím        jelentése: az akkumulátor tartalmát a cím által meghatározott helyre elmentjük. A cím lehet közvetlenül kimeneti cím is, ám amennyiben a keletkezett eredményt a továbbiakban még más helyeken is szeretnénk használni, akkor célszerű memóriába is elmenteni.

Példa:

**OUT M1** ; ACC → M1

**OUT Y0** ; ACC → Y0

A bájtműveletek esetében, amikor az adatregiszterekkel dolgozunk akkor a MOV utasítást használjuk tárolásra. Ez kétcímes utasítás:

**MOV** cím1, cím2        jelentése: a cím1 tartalmát a cím2-re helyezzük, a cím1 tartalma változatlan marad, vagyis másolási művelet hajt végre.

Példa:

**MOV K100, D1** ; K100 → D1

Az eddig ismert utasítások segítségével, elkészíthetjük a már korábban ismert öntartásos motorvezérlés (3.2. ábra) utasításlistás programját.

**LD X0** ;X0[ ] → ACC

**OR Y0** ;Y0[ ] + ACC → ACC = Y0[ ] + X0[ ]

**ANI X1** ;X1[ ] · ACC → ACC = X1[ ] · (Y0[ ] + X0[ ])

**ANI X2** ;X2[ ] · ACC → ACC = X2[ ] · X1[ ] · (Y0[ ] + X0[ ])

**AND X3** ;X3[ ] · ACC → ACC = X3[ ] · X2[ ] · X1[ ] · (Y0[ ] + X0[ ])

**OUT Y0** ;ACC → Y0

**LD X3** ;X3[ ] → ACC

**OUT Y1** ;ACC → Y1

**END**



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

#### ***3.4.4. Értékadó és törlő utasítások***

Sok esetben előfordul, hogy egy MERKER-nek, vagy kimenetnek értéket kell adnunk, vagyis tartósan logikai 1-re kell állítsuk. Például ha azt szeretnénk, hogy egy nyomógombról működtetett kimenet a nyomógomb elengedése után is aktív maradjon, akkor az értékadó utasítást célszerű használni. Általános formája:

**SET** cím ; jelentése: a címen lévő adat értékét 1-re írjuk. A cím lehet kimeneti, vagy memória cím.

Példa:

**SET M0** ; M0 = 1

Az ily módon beállított kimenet, vagy MERKER mindaddig 1-esen marad, ameddig nem töröljük. A törlés általános formája:

**RST** cím ; jelentése: a címen lévő adat értékét 0-ra állítjuk.

Ebben az esetben is a cím lehet memória, vagy kimeneti cím, de lehet időzítő vagy akár számláló címe is.

Példa:

**RST M0** ; M0 = 0

#### ***3.4.5. Veremkezelő utasítások***

A veremtár az adatmemóriának (RAM) egy elkülönített része, amelynek kezelése speciális módon történik. Zsáknak (stack) is nevezik, mivel csak egy irányból érhető el. A megszakítások kiszolgálása előtt a processzor ide menti el az éppen aktuális állapot regisztereinek tartalmát, hogy a megszakítás kiszolgálása után újra elővehesse ezeket, hogy a megszakított programrész zavartalanul folytatódjon tovább. LIFO (Last In, First Out) típusú tár, azaz mindig az utoljára betett (beírt) adat vehető ki (olvasható) belőle először. Nem használ címezést, adminisztrációját a veremmutató regiszter (SP) végzi, amely mindig a verem tetejére mutat. (3.5. ábra.)

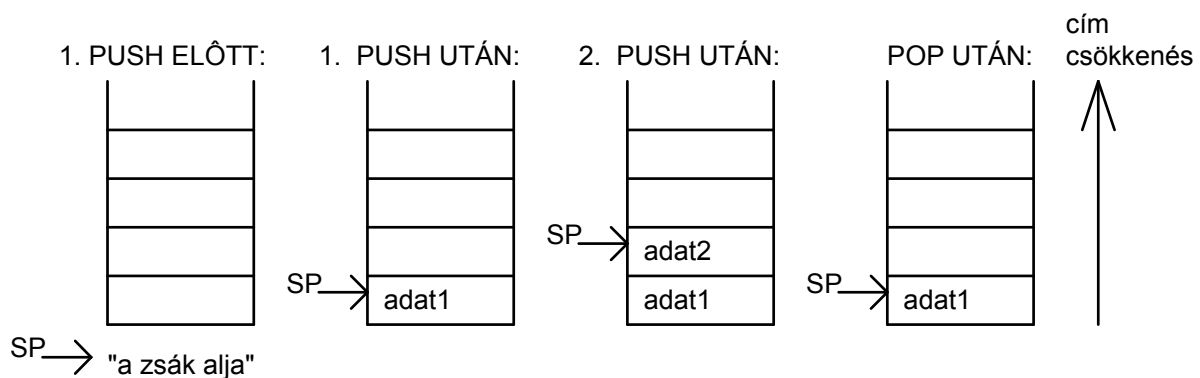




EFOP-3.5.1-16-2017-00017

**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

Assembly szintű programozás során a PUSH és POP utasításokkal írunk, illetve olvasunk a veremből. (Olvasás után a kiolvasott adat törlődik) Üres állapotban a SP a verem alját mutatja, ami megfelel a maximális kapacitásnak. Minden PUSH utasítás után a veremmutató dekrementál ( $SP = SP - 1$ ), POP utasítás után, pedig inkrementál ( $SP = SP + 1$ ). Ha megtelik a verem ( $SP = 0$ ), hibát (kivételt) generál.

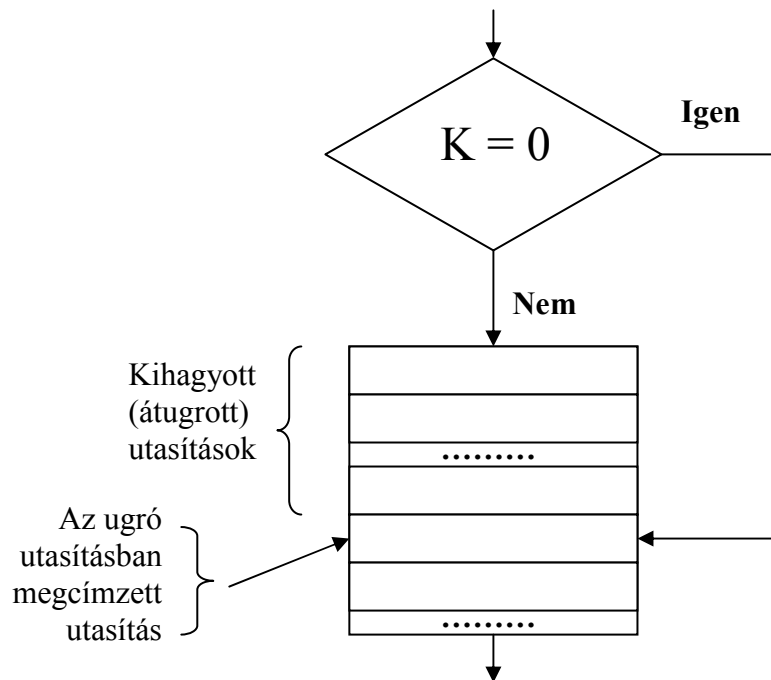


**3.5. ábra.** A veremtár kezelése

A Mitsubishi PLC-knél a felhasználó is használhatja a veremtárat. A verembe az **MPS** utasítással helyezünk (írunk) be adatot, és az **MPP** utasítással veszünk (olvassuk) ki. Ezen kívül még használható olvasásra az **MRD** utasítás, amely csak kiolvassa, de nem veszi ki az adatot a veremből. (A veremmutató értéke ilyenkor nem változik.)

#### **3.4.6. Ugró utasítások**

Az ugró utasításokat akkor használjuk, amikor a programon belül, olyan feltételek alakulnak ki, hogy bizonyos programrészek végrehajtása feleslegessé válik. Tegyük fel, hogy a programon belül egy összehasonlítást végzünk, amelynek eredménye igaznak bizonyul. Ilyenkor felesleges elvégeznünk a soron következő programrészletet, amely éppen arra hivatott, hogy ezt a feltételt megvalósítsa. (3.6. ábra.) Az ugró utasítások lehetnek feltételhez kötöttek, vagy feltétel nélküliek. Az utasításban szereplő címrész mindig annak az utasításnak a címét fogja jelölni, amely utasítással folytatódik a program futása.



**3.6. ábra.** Feltételhez kötött ugrás szemléltetése.

A Mitsubishi FX PLC-nél a feltétel nélküli ugrást használják, amelynek általános alakja:

**CJ** utasítás cím vagy címke                      jelentése: ugorj a címkézett utasításra!

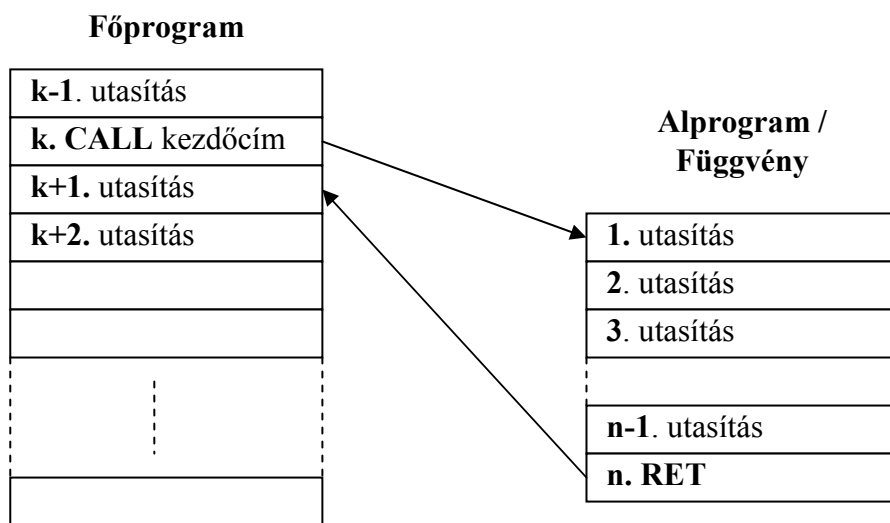
Az ugró utasítás feldolgozásakor a processzor vezérlője a programszámlálót (PC) az ugró utasításban szereplő címre állítja.

### 3.4.7. Vezérlő utasítások

A vezérlő utasítások programfolyamatokra vonatkoznak. Ilyen utasítások például a megszakításokat engedélyező vagy tiltó utasítások. Ezek rendszerint cím nélküli utasítások úgynevezett parancsok. Ide sorolható a program végét jelző **END** utasítás is, ami a programszámlálót automatikusan a kezdőcímre állítja, hogy biztosítva legyen a ciklikus működés. Vezérlő utasítás még az alprogramot vagy függvényt meghívó **CALL** utasítás is, melynek általános formája:

**CALL** cím                      jelentése: ugrás az alprogram kezdőcímére!

A CALL utasítás egyben a verembe helyezi a program folytatásához szükséges változók értékeit. Az alprogram vagy függvény végét a **RET** parancs jelzi, melynek hatására a programszámláló visszaáll a főprogrambeli következő utasítás címére, a regiszterekbe pedig visszakerülnek a veremtárból az elmentett változók. (3.7. ábra)



**3.7. ábra.** Alprogram vagy függvény hívása.



EFOP-3.5.1-16-2017-00017

„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”

## 4. EGYSZERŰ VEZÉRLÉSEK PROGRAMOZÁSI MÓDSZEREI

Az előző fejezetben már láthattunk egy példát az öntartó vezérlés programozására létradiagrammal és utasításlistában is. A továbbiakban néhány gyakran előforduló vezérlési helyzet programozási lehetőségeit ismertetem.

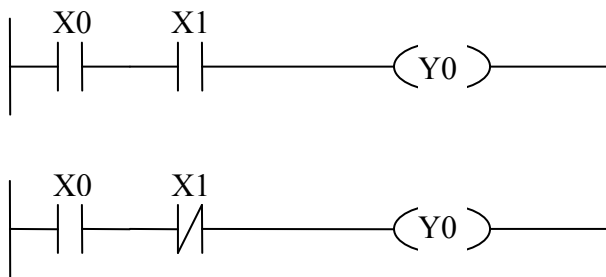
### 4.1. Logikai kapcsolatok programozása

#### 4.1.1. ÉS műveletek, AND, ANI utasítások

Az ÉS művelet egy áramkörben kettő vagy annál több elem soros kapcsolatának felel meg. A kimeneti feltétel akkor és csak akkor teljesül, ha mindegyik elem által jelölt feltétel igaz.

$$Y0 = X0 \cdot X1 \quad \text{vagy} \quad Y0 = X0 \cdot \overline{X1}$$

##### Létradiagram



##### Utasításlista

LD X0  
AND X1  
OUT Y0

LD X0  
ANI X1  
OUT Y0

#### 4.1.2 VAGY műveletek, OR, ORI utasítások

A VAGY művelet egy áramkörben kettő vagy annál több elem párhuzamos kapcsolatát jelenti. A kimeneti feltétel akkor teljesül, ha a feltételek közül legalább egy igaz.

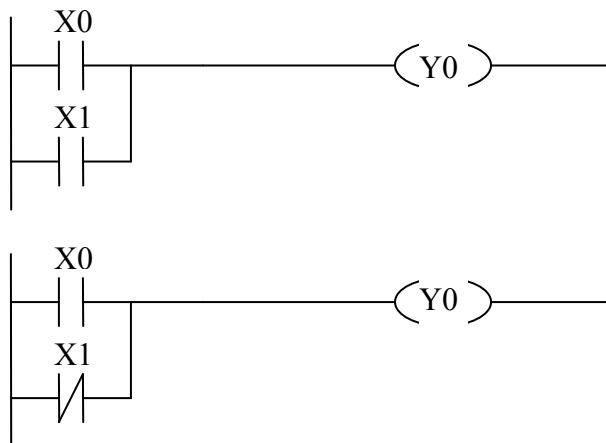
$$Y0 = X0 + X1 \quad \text{vagy} \quad Y0 = X0 + \overline{X1}$$



EFOP-3.5.1-16-2017-00017

**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

#### Létradiagram



#### Utasításlista

**LD X0  
OR X1  
OUT Y0**

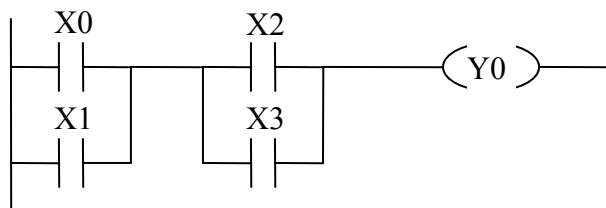
**LD X0  
ORI X1  
OUT Y0**

#### **4.1.3. MERKER-ek és blokkutasítások használata, ANB és ORB utasítások**

Tételezzük fel az alábbi vezérlési állapotegyenletet:

$$Y0 = (X0 + X1) \cdot (X2 + X3)$$

#### Létradiagram



#### Utasításlista

<i>1. megoldás</i>	<i>2. megoldás</i>
<b>LD X0</b>	<b>LD X0</b>
<b>OR X1</b>	<b>OR X1</b>
<b>OUT M0</b>	<b>LD X2</b>
<b>LD X2</b>	<b>OR X3</b>
<b>OR X3</b>	<b>ANB</b>
<b>AND M0</b>	<b>OUT Y0</b>
<b>OUT Y0</b>	<b>END</b>
<b>END</b>	

Az első megoldás szerint, miután az első zárójelre vonatkozó műveletsorozatot elvégeztük, kénytelenek vagyunk az ACC tartalmát az M0 MERKER-be menteni, hogy be tudjuk tölteni X2-t és elvégezhessük a második zárójelben lévő műveleteket, majd a keletkező eredmény és az M0 tartalmával még egy ÉS műveletet végzünk. A második megoldásban kihasználjuk a Mitsubishi PLC adta sajátos lehetőséget, amely szerint akár több egymás után következő



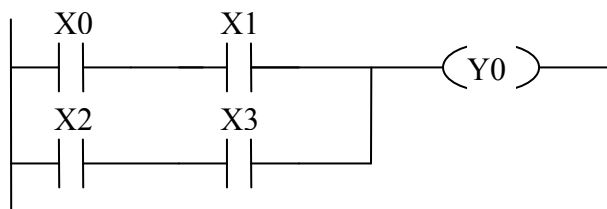
EFOP-3.5.1-16-2017-00017

**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

VAGY műveletsorozatot egyetlen egy utasítással (**ANB**), a blokkok között megvalósul az **ÉS** kapcsolat.

Hasonló az eljárás **ÉS** blokkok VAGY kapcsolat szerinti összekapcsolásakor is, ilyenkor az **ORB** utasítást használjuk. Legyen a vezérlési állapotegyenlet:  $Y0 = X0 \cdot X1 + X2 \cdot X3$

#### Létradiagram



#### Utasításlista

1. megoldás	2. megoldás
<b>LD X0</b>	<b>LD X0</b>
<b>AND X1</b>	<b>AND X1</b>
<b>OUT M0</b>	<b>LD X2</b>
<b>LD X2</b>	<b>AND X3</b>
<b>AND X3</b>	<b>ORB</b>
<b>OR M0</b>	<b>OUT Y0</b>
<b>OUT Y0</b>	<b>END</b>
<b>END</b>	

Az első megoldás szerint itt is **MERKER**-t használunk, majd az **M0** tartalmával és a második **ÉS** blokk között VAGY műveletet hajtunk végre. A második esetben az **ÉS** blokkokat az **ORB** utasítás foglalja egybe.

*Megjegyzés.* Maximálisan 8 blokk fűzhető össze egy **ANB** vagy **ORB** utasítással.

#### **4.1.3. Keresztretesz kapcsolat programozása**

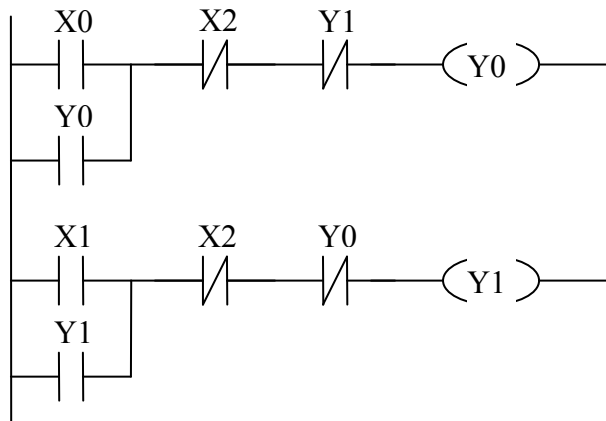
A vezérléstechnikában nagyon gyakran olyan megoldásokat kell alkalmazni, hogy amikor aktív egy adott kimenet, egy másik kimentet ne lehessen aktiválni mindaddig, míg az előző aktív, és fordítva. Ilyenek például az egymással ellentétes hatású kimenetek, forgásirány, előre-hátra, le-fel, stb. A 4.1. ábrán egy lehetséges megoldás programozását látjuk.

Legyen **Y0**, és **Y1** a két ellentétes irány kimenete, **X0** és **X1** a két irány kapcsolása, **X2** pedig a kikapcsolás. A kimenetekhez tartozó vezérlési állapotegyenletek:

$$Y0 = (X0 + Y0) \cdot \overline{X2} \cdot \overline{Y1}$$

$$Y1 = (X1 + Y1) \cdot \overline{X2} \cdot \overline{Y0}$$

### Létradiagram



### Utasításlista

```
LD X0
OR Y0
ANI X2
ANI Y1
OUT Y0
LD X1
OR Y1
ANI X2
ANI Y0
OUT Y1
END
```

**4.1. ábra.** Keresztretesz létradiagramja és utasításlistája

A létradiagramból jól látszik, hogy bármelyik irány csak akkor indítható, amikor a másik nem aktív.



EFOP-3.5.1-16-2017-00017

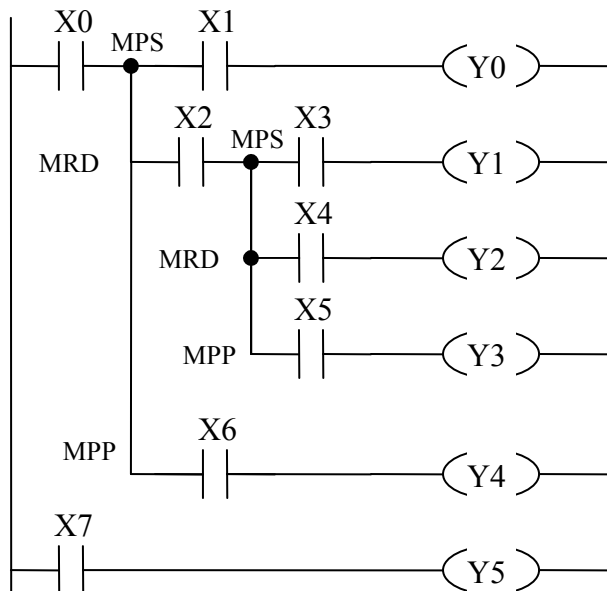
**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

#### 4.1.4. A veremtár utasítások alkalmazása

Az előző fejezetben bemutatott veremtár használat, nem csak programváltozók tárolására használható, hanem programozási célokra is.

Tekintsük az alábbi létradiagramos programszerkezetet (4.2. ábra)

Létradiagram:



Utasításlista

```
LD X0
MPS
AND X1
OUT Y0
MRD
AND X2
MPS
AND X3
OUT Y1
MRD
AND X4
OUT Y2
MPP
AND X5
OUT Y3
MPP
AND X6
OUT Y4
MPP
AND X7
OUT Y5
END
```

**4.2. ábra.** A veremtár használata

Az utasításlista alapján láthatjuk, hogy ahány MPS utasítást használunk, annyi MPP-nek is lennie kell, hogy ne maradjon semmi a veremben. Maximálisan 11 szinten történhet egymásba ágyazás.

#### 4.1.5. Időzítők és számlálók [2]

Folyamatvezérlés közben gyakran van szükségünk különböző időzítések, illetve késleltetések használatára. Huzalozott vezérléseknél erre a célra időreléket használtak. A PLC-s vezérléseknél ezekre már nincs szükség, mert felhasználhatjuk a PLC belső programozható időzítőit. A PLC időzítői valójában számlálók, amelyek a PLC belső 0,1 s illetve 0,01 s



alapidejű órajel impulzusait számolják az utasításban megadott értékig. Amikor a számláló eléri a beállított értéket, az időzítő kimenet bekapcsolódik, és mindaddig aktív marad, ameddig az időzítést kiváltó feltétel adott. Az időzítő utasítások általános formája a Mitsubishi PLC-nél: **Tcím Kszám**, ahol a **Tcím** a kiválasztott időzítő címét, a **Kszám** pedig egy konstans, amelyik megadja az időzítés időtartamát.

A Mitsubishi PLC-k típustól függően, többféle időzítőt tartalmaznak. Például az FX0 típusú PLC 32 darab (T0-T31) 100 ms-os, és 24 darab (T32-T55) 10 ms-os időzítőt tartalmaz. Ezek az időzítők 16 bitesek, ami azt jelenti, hogy  $K = 1 - 32767$  közötti értéket vehet fel, vagyis maximálisan 3276,7s-os (T0-T31), illetve 327,67s-os (T32-T55) késleltetést programozhatunk.

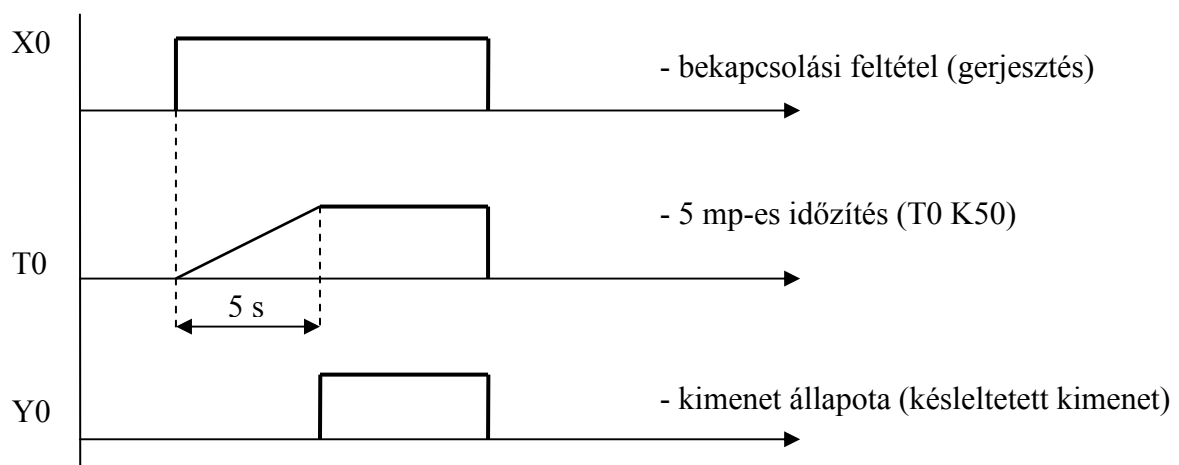
Példa:

T0 K50                      jelentése: a T0 időzítővel 5 másodperces késleltetés

A következő példákban néhány gyakrabban alkalmazott időzítési eljárást mutatunk be.:

#### 4.1.6. Bekapcsolási (meghúzási) késleltetés

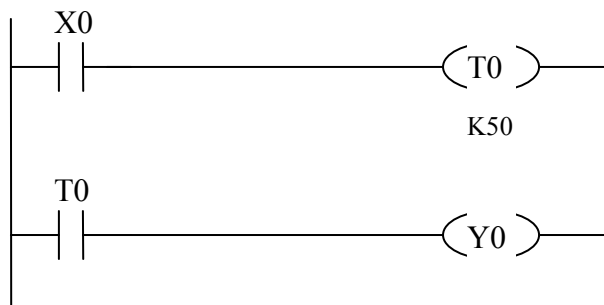
Olyankor alkalmazzuk, amikor az a célunk, hogy a kimenet a gerjesztés időpillanatától számítva, bizonyos idő eltelte után váljon aktívvá. (4.3. ábra)



**4.3. ábra.** Bekapcsolási késleltetés idődiagramja.

A 4.4. ábrán az előbbi késleltetés programvázlatát láthatjuk létradiagramos és utasításlistás formában:

Létradiagram:



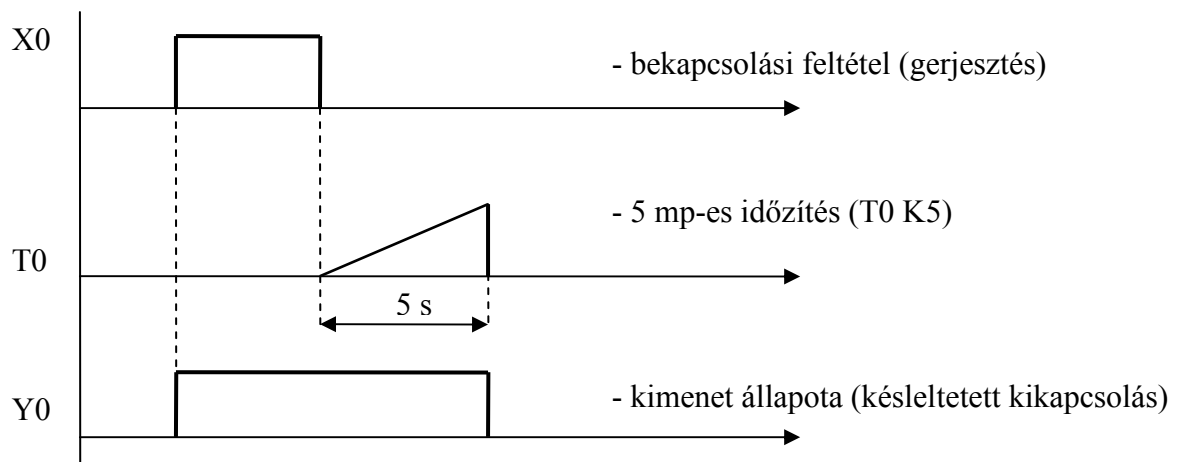
Utasításlista

```
LD X0
OUT T0 K50
LD T0
OUT Y0
END
```

**4.4. ábra.** Egy bekapcsolási késleltetés programja

#### 4.1.6. Kikapcsolási (elengedési) késleltetés

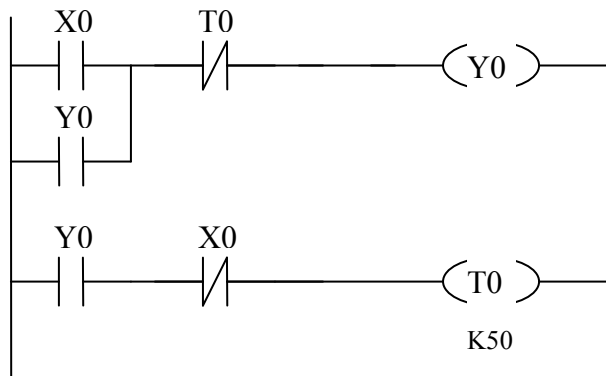
Kikapcsolási késleltetéskor a gerjesztés időpillanatában a kimenet aktívvá válik, viszont a gerjesztés megszűnte után, az időzítés időtartamáig még aktív marad. (4.5. ábra.)



**4.5. ábra.** Kikapcsolási késleltetés idődiagramja.

A késleltetés programvázlatát a 4.6. ábrán láthatjuk. A bemeneti feltételt közvetett módon biztosítanunk kell az időzítés időtartamáig, ezért kissé bonyolultabb a programozása.

#### Létradiagram



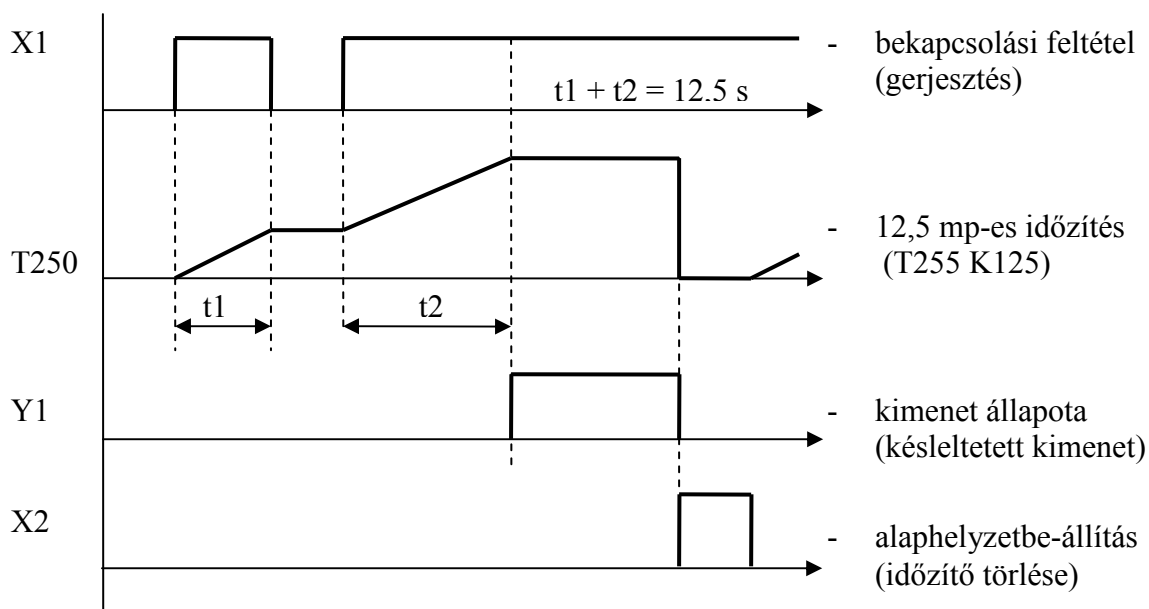
#### Utasításlista

```
LD X0
OR Y0
ANI T0
OUT Y0
LD Y0
ANI X0
OUT T0 K50
END
```

**4.6. ábra.** Egy bekapcsolási késleltetés programja

#### **4.1.7. Állapotmegőrző időzítők**

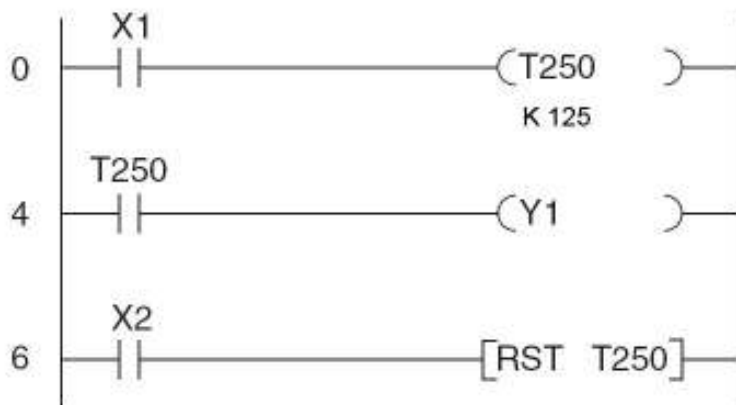
A Mitsubishi család fejlettebb FX1N, FX2N, FX3U vezérlői az alapidőzítőkön kívül még olyan speciális időzítőket is tartalmaznak, amelyek megőrzik a számláló pillanatnyi értékét, még akkor is, ha vezérlés közben az időzítőt kikapcsoljuk (megszüntetjük a gerjesztést). A pillanatnyi érték egy olyan memóriában tárolódik, amely megtartja a tárolt értéket még feszültségkiesés esetén is. A 4.7. ábrán az állapotmegőrző időzítő működési idődiagramját láthatjuk.



**4.7. ábra.** Állapotmegőrző időzítés elve

A következő ábrán az előbbi időzítés programrészletét mutatjuk be. Ilyen állapotmegőrző időzítést T250 és T255 között használhatunk összesen hat darabot. Például: **T250 K125** utasítás 12,5 másodperces állapotmegőrző időzítést valósít meg két lépésben.

#### Létradiagram



#### Utasításlista

0	LD	X1
1	OUT	T250 K125
4	LD	T250
5	OUT	Y1
6	LD	X2
7	RST	T250

**4.8. ábra.** Példa egy állapotmegőrző időzítés programozására



EFOP-3.5.1-16-2017-00017

**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

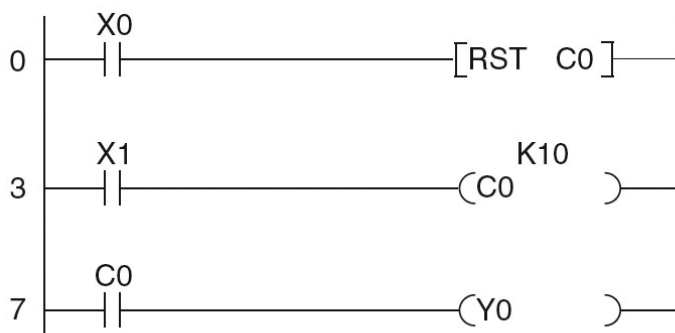
#### 4.1.8. Számlálók

Az FX család vezérlői belső számlálókkal is rendelkeznek, amelyeket számolási műveletekre használhatunk. Ezek a számlálók a bemenetekhez rendelt impulzusokat számolják. A számláló kimenete akkor válik aktívvá, amikor tartalma eléri a K paraméter által megadott értéket. Például, ha azt akarjuk, hogy a C0 számláló 10-ig számoljon, akkor az **C0 K10** utasítással adhatjuk meg.

Akárcsak az időzítőknél, a számlálóknál is a PLC típusától függően, többféle számlálót használhatunk. Ezek lehetnek, 16 bites előreszámlálók, 32 bites előre-hátraszámlálók, vagy állapotmegőrző számlálók. Például az FX0 típus 16 darab (C0 - C15) 16 bites (K = 1 – 32767), az FX0N 32 darab (C0 – C31) ugyancsak 16 bites számlálót, míg az FX1N 21 darab (C235 – C255) 32 bites nagysebességű állapotmegőrző számlálót is tartalmaz.

A 4.9. ábrán egy számlálási feladat programozását tanulmányozhatjuk.

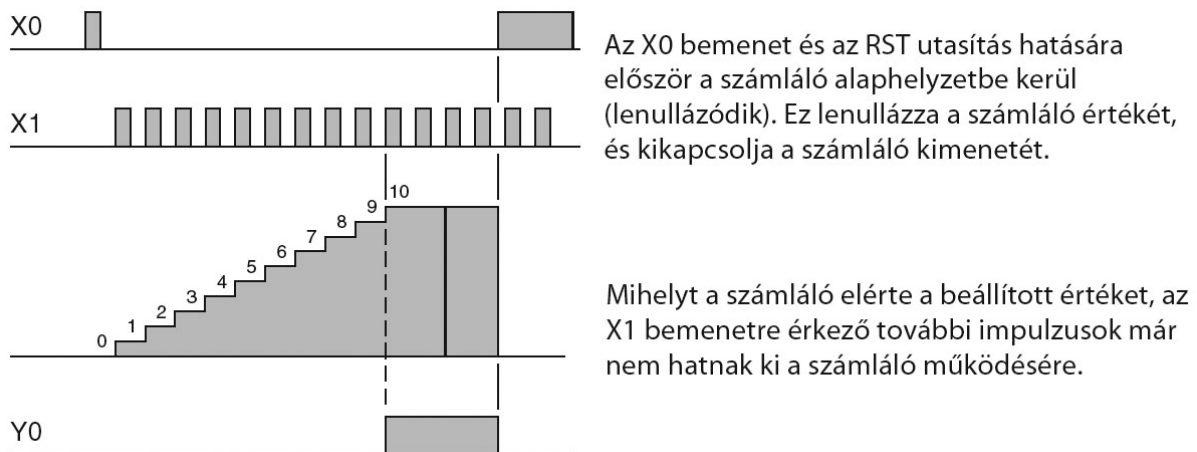
##### Létradiagram



##### Utasításlista

0	LD	X0	
1	RST	C0	
3	LD	X1	
4	OUT	C0	K10
7	LD	C0	
8	OUT	Y0	

**4.9. ábra.** Számláló programozása



**4.10. ábra.** A számláló működési vázlata (*Mitsubishi FX kézikönyv alapján*)

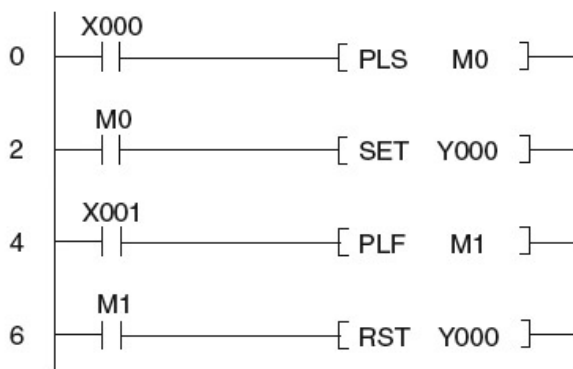
Az X1 bemenet mindegyik bekapcsolásakor a C0 számláló értéke eggyel nő. Amikor a számláló eléri a K paraméter által beállított értéket, kimenete 1-re vált és mindaddig megmarad, ameddig nem töröljük a tartalmát. (4.10. ábra)

Megjegyzés: Célszerű a számlálás megkezdése előtt is törölni.

#### **4.1.9. Pergésmentesítés impulzusokkal (impulzusgenerálás)**

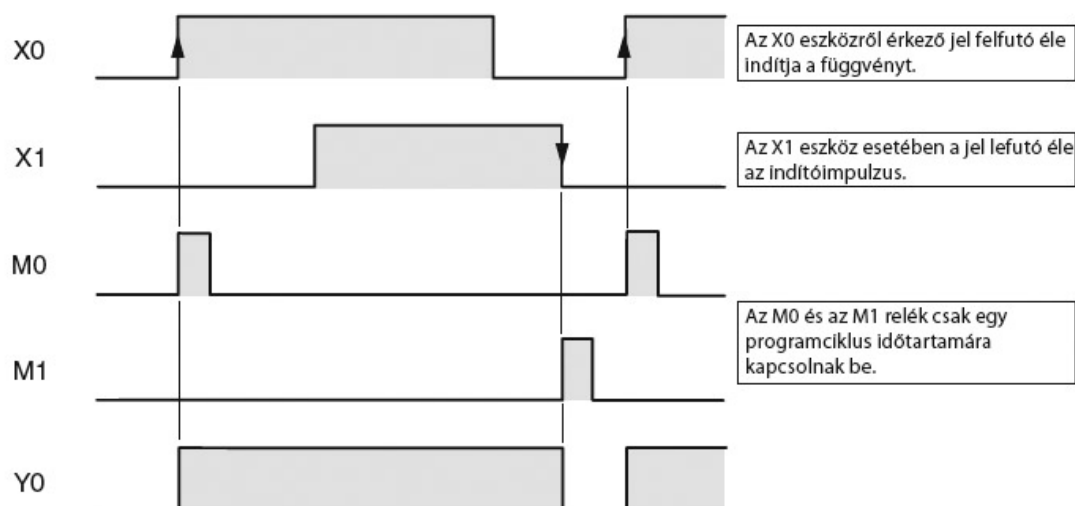
A valóságos érintkezők működtetésekor, a be- és kikapcsolás pillanatában számtalan nemkívánatos zaj keletkezik. Ha azt akarjuk, hogy a felmenő ( $0 \rightarrow 1$ ), illetve a lefutó élre ( $1 \rightarrow 0$ ) csak egy-egy jól meghatározott impulzus keletkezzen, akkor használjuk a PLS, illetve PLF utasításokat, az impulzusokat pedig MERKER-ekben tároljuk, amelyek csak egy ciklus idejéig maradnak 1-esen, abban a ciklusban amikor keletkeztek. A PLS utasítás a felmenő élre, a PLF, pedig a lefutó élre generál impulzust. (4.11. ábra)

Létradiagram



Utasításlista

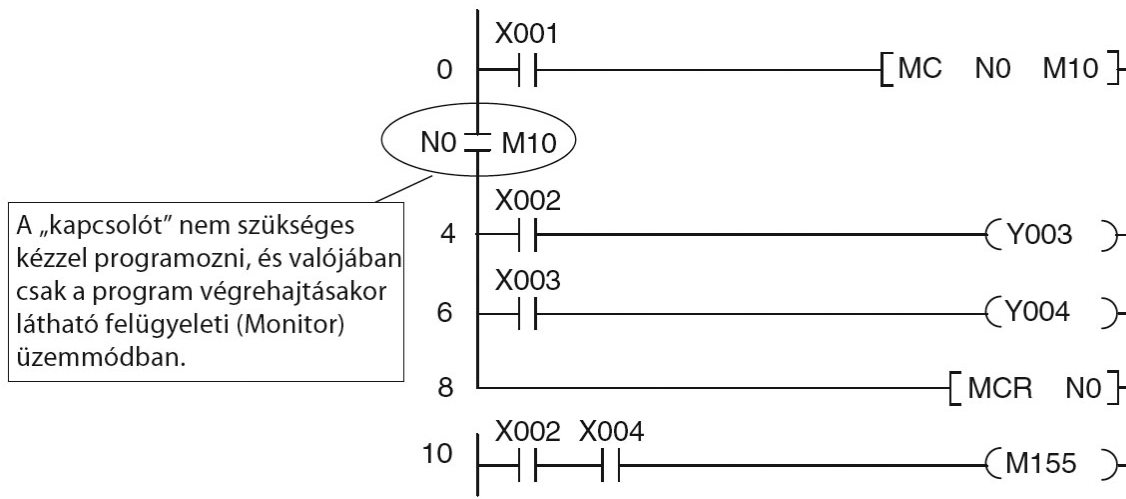
0	LD	X000
1	PLS	M0
2	LD	M0
3	SET	Y000
4	LD	X001
5	PLF	M1
6	LD	M1
7	RST	Y000



**4.11. ábra.** Az impulzusgenerálás programozása és idődiagramjai

**4.1.10. A főszabályozó funkció (Master Control) használata**

A főszabályozó funkció lehetőséget biztosít bizonyos programrészletek tetszőleges feltételhez kötötten aktívra tételére, illetve alaphelyzetbe állítására. Az aktiválást a Master Control Set (MC), deaktiválást a Master Control Reset (MCR) utasítások végzik. Létradiagramos ábrázolási módban az MC utasítás úgy működik, mint egy kapcsoló, aminek zárva kell lennie, ha azt akarjuk, hogy a programblokk végrehajtásra kerüljön. (4.12. ábra)



4.12. ábra. Példa a Master Control funkció használatra

## 4.2. Alkalmazások

### I. feladat

**Háromfázisú aszinkronmotor Y -> Δ indítása időzítéssel, keresztretesszel és hőrelés védelemmel.** A motor indítása és megállítása nyomógombokkal történjen. A hőrelé bontó (NC) érintkezővel rendelkezik. Induláskor a motor 5 s-ig csillagban van ( $K_Y$ ), majd átkapcsol deltába ( $K_\Delta$ ). A motor tekercseinek táplálása a K mágneskapcsoló érintkezőin keresztül történik. A kapcsolást úgy kell megvalósítani, hogy még véletlenül se fordulhasson elő, hogy mindkét üzemmód egyidőben működhessen. (keresztretesz)

- Programterv, vezérlési állapotegyenletek, változók címzése.
- Létradiagram elkészítése
- PLC bekötési vázlata.
- Utasításlistás programrészlet.



### I. feladat megoldása

A feladat változói és címkiosztása a következőképpen írható fel:

- bemeneti változók:

START → X0      indít  
STOP → X1      leállít  
Th → X2      hőrelé

- kimeneti változók:

K → Y0      motor táplálás  
KY → Y1      csillag kapcsolás  
KΔ → Y2      delta kapcsolás

Három kimeneti elemünk és egy időzítésünk van, vagyis négy vezérlési állapotegyenletet írhatunk fel:

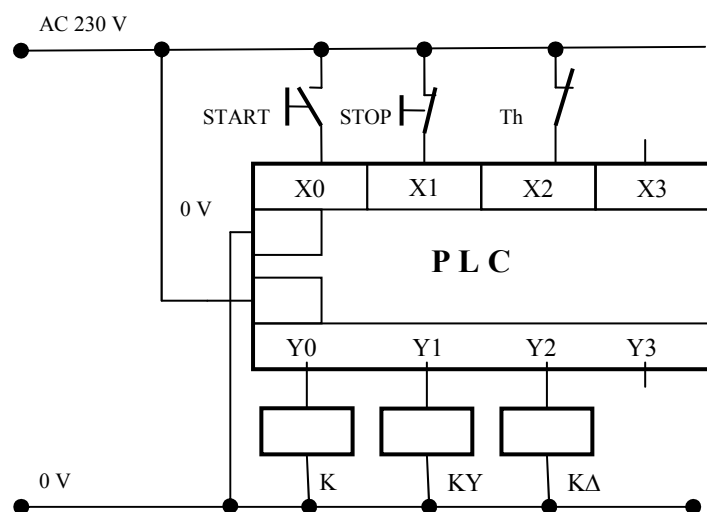
$$Y0 = (X0 + Y0) \cdot \overline{X1} \cdot \overline{X2}$$

$$Y1 = Y0 \cdot \overline{T0} \cdot \overline{Y2}$$

$$T0 = Y0$$

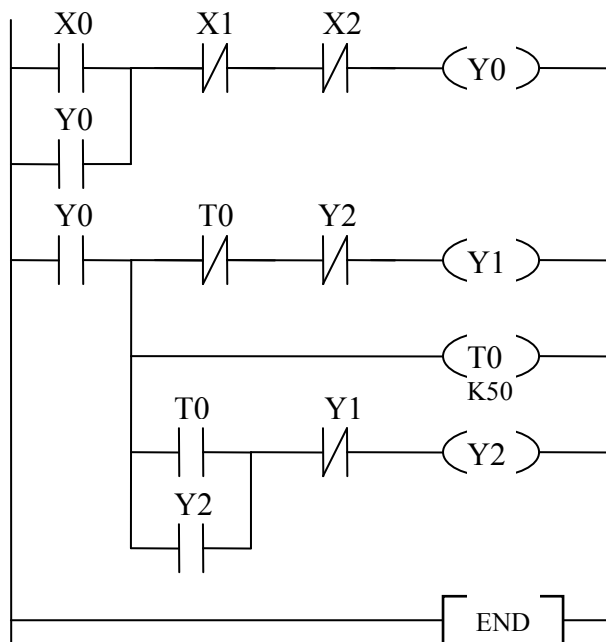
$$Y2 = Y0 \cdot (T0 + Y2) \cdot \overline{Y1}$$

A címkiosztás alapján megválaszthatjuk a PLC-t és elkészíthetjük a bekötési vázlatot (4.13. ábra.):



**4.13 ábra.** A PLC bekötési vázlata

Megjegyzés: A vezérlési állapotegyenletek alapján a *STOP* gomb és a *Th* hőrelé érintkezői egyaránt bontó érintkezők kell legyenek.



#### Utasításlista

```
LD X0
OR Y0
ANI X1
ANI X2
OUT Y0
LD Y0
MPS
ANI T0
ANI Y2
OUT Y1
MRD
OUT T0 K50
MPP
AND T0
OR Y2
ANI Y1
OUT Y2
END
```

## II. feladat

Egy szállítószalagon késztermékek haladnak a csomagoló egység felé. Minden tízedik termék után a szalag megáll 45 másodpercre, majd újraindul automatikusan. (A termékszámolás a szalag végén történik.) A szalagot lehet indítani nyomógombról is és bármikor megállítható a vészki kapcsolóról. A 45 másodperces megállást egy lámpa is jelzi.

- Programterv, vezérlési állapotegyenletek, változók címzése.
- Létradiagram elkészítése
- PLC bekötési vázlata.
- Utasítássoros programrészlet.



**EFOP-3.5.1-16-2017-00017**

**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

## II. feladat megoldása

Feltételezzük, hogy a szállítószalagot villanymotor hajtja, amelyet a  $K$  mágneskapcsoló működtet. Legyen a  $Be$  nyomógomb az indító,  $STOP$  a vészgomb,  $S$  a munkadarab-érzékelő,  $L$  pedig jelzőlámpa. A fentiek alapján tetszőlegesen a vezérlőelemekhez hozzárendelhetjük a megfelelő IO címeket.

- *bemeneti változók*

$Be \rightarrow X0$  - szalag indul

$STOP \rightarrow X1$  - vészgomb

$S \rightarrow X2$  - munkadarab érzékelő

- *kimeneti változók*

$K \rightarrow Y0$  - motorkapcsoló

$L \rightarrow Y1$  - kijelző

Ezekon kívül még van a számláló  $C0$  és az időzítő  $T0$ .

Felírhatjuk a vezérlési állapotegyenleteket:

$$Y0 = (X0 + T0 + Y0) \cdot \overline{C0} \cdot \overline{X1}$$

$$C0 = X2$$

$$T0 = C0$$

$$Y1 = C0$$

Amennyiben az indításhoz, megállításhoz a SET, RST utasításokat használjuk, akkor az első állapotegyenlet két részre bontható, a többi változatlan marad:

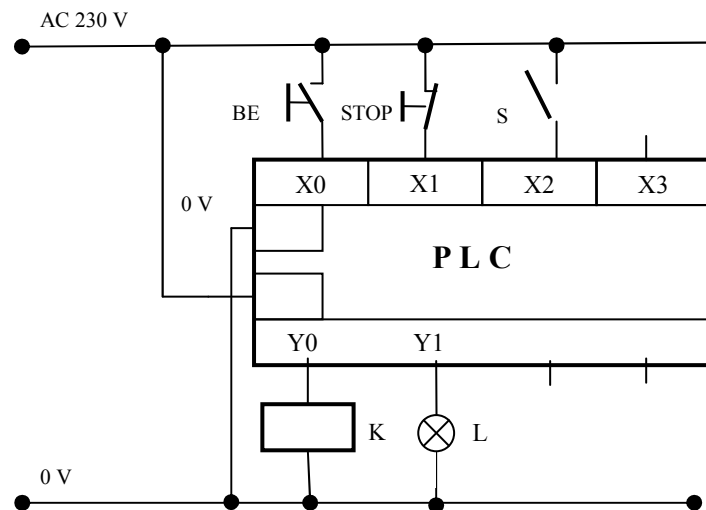
$$Y0 = X0 + T0$$

$$\overline{Y0} = C0 + X1$$

Mindkét megoldást ismertetjük.

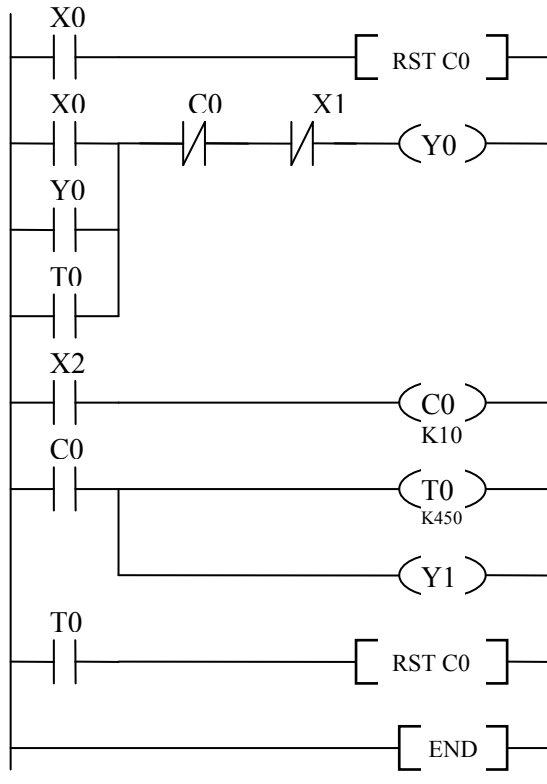
Megjegyzés: Indításkor, valamint az időzítés letelte után a számlálót törölni kell.

A címkiosztásnak megfelelően a PLC bekötési vázlata a következő ábrán látható (4.14. ábra):



**4.14. ábra.** A PLC bekötési vázlata

### Létradiagram



## Utasításlista

```
LD X0
RST C0
LD X0
OR Y0
OR T0
ANI C0
ANI X1
OUT Y0
LD X2
OUT C0 K10
LD C0
OUT T0 K450
OUT Y1
LD T0
RST C0
END
```

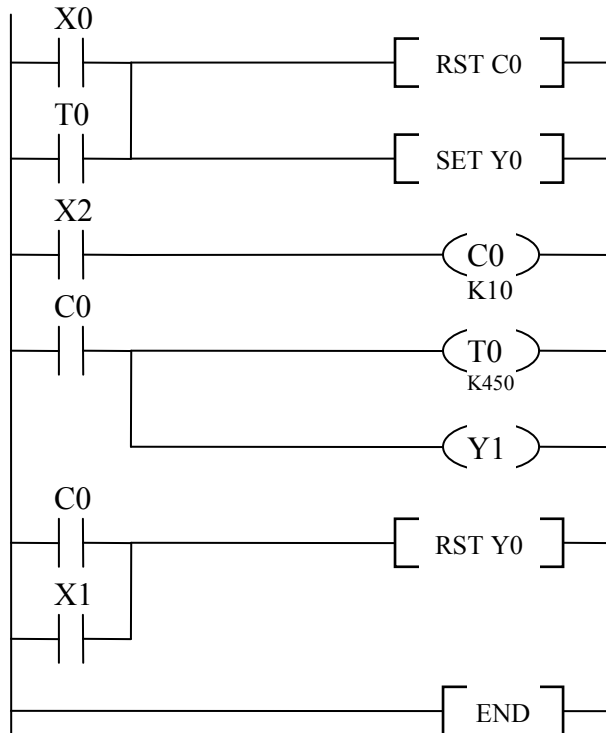


EFOP-3.5.1-16-2017-00017

**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

A második megoldás a SET, RST utasításoknak köszönhetően valamivel egyszerűbb lesz.

#### Létradiagram



#### Utasításlista

```
LD X0
OR T0
RST C0
SET Y0
LD X2
OUT C0 K10
LD C0
OUT T0 K450
OUT Y1
LD C0
OR X1
RST Y0
END
```



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

## **5. SORRENDI VEZÉRLÉSEK PROGRAMOZÁSA [2, 3]**

Sorrendi vezérléseknél (lefutó vezérlés) célszerű a sorrendi programozási eljárást alkalmazni. Ebben az esetben, a tervezési fázisban nem vezérlési állapotegyenleteket írunk fel, hanem ütemtervet (időtervet) készítünk. Mindegyik ütem a sorrendi vezérlési folyamat egy-egy lépését jelenti. Egyik lépésből a másikba átmenni csak a továbblépési feltételek teljesülése után lehetséges. A lépésekhez operátorokat rendelünk, majd ezeket műveletekkel (utasításokkal) töltjük fel. Az operátorok és a közöttük lévő továbblépési feltételek összessége alkotja a sorrendi folyamatábrát vagy más néven gráfcetet.

### **5.1. Operátorok létre hozása, törlése**

A Mitsubishi PLC-k sorrendi programozásakor kiinduló lépésként inicializáló operátorokat definiálhatunk. S0 és S9 között tetszőlegesen hozhatunk létre inicializáló operátort. Legalább egyet kötelezően kell használnunk. Azért, hogy indításkor automatikusan (külső beavatkozás nélkül) belépjünk ebbe az operátorba, továbblépési feltételként rendszerint egy rövid ideig tartó impulzust használunk. Ilyen impulzusokat a gyártó által készített speciális merkekben találunk. Pl. az M8002 egy ilyen pozitív impulzust tartalmaz. Az inicializáló operátorokban szokás leprogramozni a vezérlés alapbeállításait (paraméterei) vagy azokat a funkciókat, amelyeket csak kizárólag indítás után közvetlenül, de csak egyszer kell végrehajtani. Pl. a közlekedési lámpák indítása biztonsági okokból úgy történik, hogy bekapcsolás után egy bizonyos ideig minden irányban a sárga villogó üzemmód fut, majd utána beáll a normás működési lámpasorrend.

S10 és S999 között tetszőlegesen definiálhatunk munkaoperátorokat. (Mitsubishi FX0 PLC-nél csak S10 és S63 közötti operátorok használhatók.) Mindegyik lépéhez hozzárendelünk egy ilyen operátort. Az operátorok létre hozása a SET paranccsal történik. Pl. SET S11. Adott esetben, amikor egy adott operátorra már nincs szükség akár törölhetjük is. Operátort törölni a RST paranccsal lehet. Pl. RST S11.

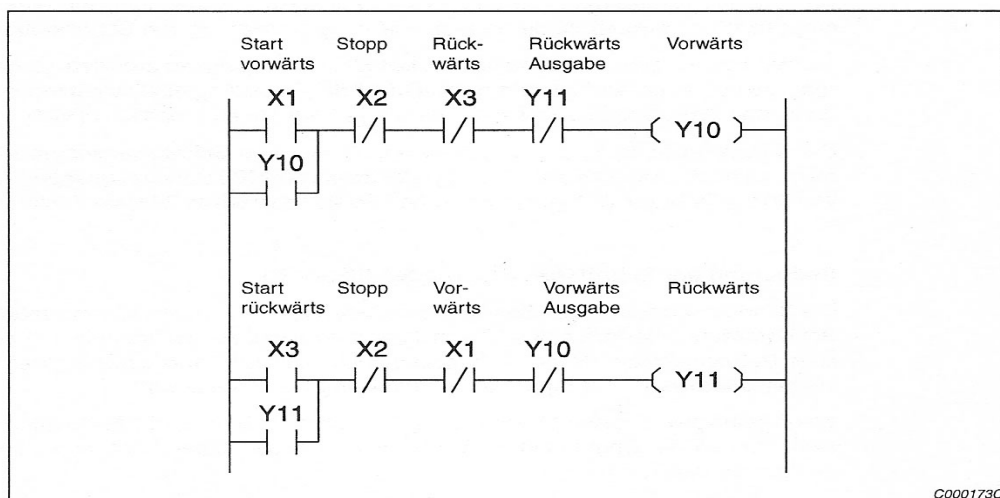
## 5.2. Az STL utasítás

Az STL utasítás egy általános PLC utasítás, amely a lépésvezérelt programozás alapját képezi. Az STL utasítással a már előzőleg létre hozott operátort megnyitjuk, majd az ezt követő utasításokkal és parancsokkal leprogramozzuk azokat a funkciókat, amelyeket az adott lépés alatt kell végre hajtania. Tovább lépésként definiáljuk a feltételeket és létre hozzuk a következő lépés operátorát. Pl. STL S11 utasítás azt jelenti, hogy a már előzőleg létre hozott S11 operátort megnyitjuk, vagyis belépünk az adott lépésbe.

Megjegyzés: Amikor elhagyunk egy lépést, az előző lépés kimenetei visszavonásra kerülnek, hacsak nem SET paranccsal voltak aktívvá téve. Ez a sorrendi vezérlések esetében előnyös, mert így elkerülhetjük a többszörösen lefedett kimeneteket, amelyek hibás működést okozhatnak.

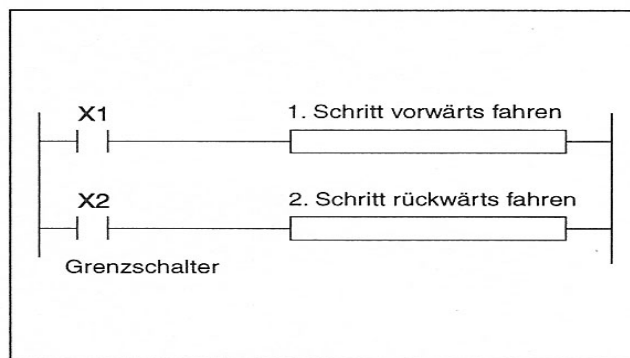
### 5.2.1. Alkalmazási példák az STL utasításokhoz

A hagyományos áramút-terven alapuló létradiagramos programozási mód abból áll, hogy a kimenethez hozzárendelt bemeneti kapcsolatok a vezérlési logikának megfelelően sorosan vagy párhuzamosan kapcsolódnak egymáshoz. A munkafolyamatok biztonságos működéséhez, valamint a nemkívánatos irányítási folyamatok, és azzal összefüggő téves funkciók ellen egy ilyen program nagyon sok keresztretesz feltételt kell tartalmazzon.



**5.2. ábra.** Alkalmazási példa keresztreteszeléssel (Mitsubishi FX kézikönyv alapján)

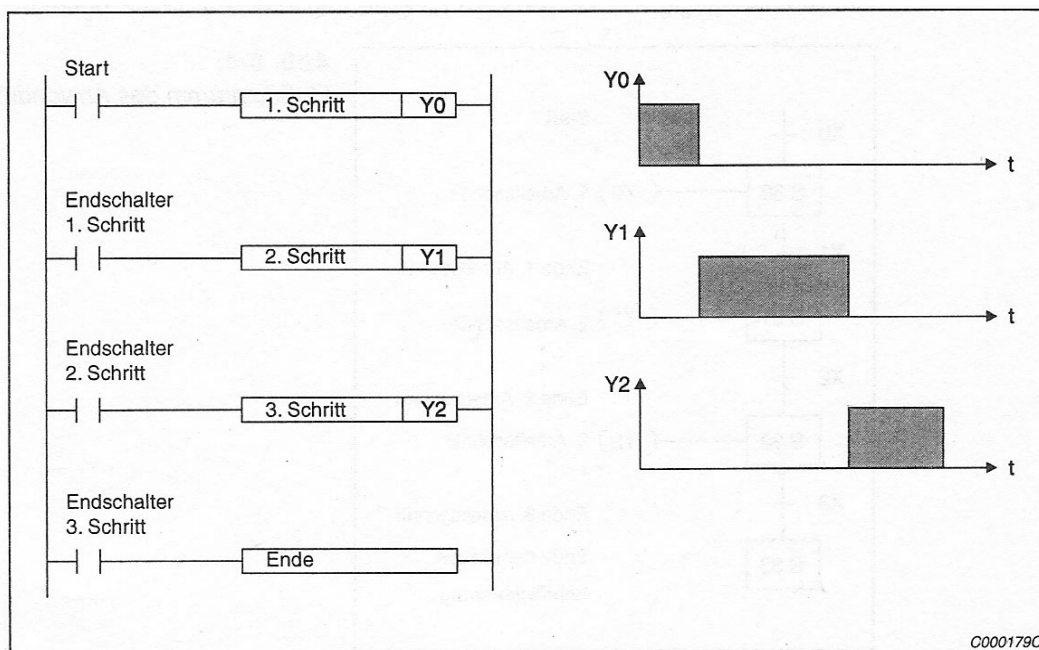
Lépésvezérlő utasítások alkalmazásakor, elhagyhatjuk a keresztreteszélést, ilyenkor az egymásnak ellentétes irányítási vonalak, mint pl. fel vagy le, előre vagy hátra stb, a program során bizonyos feltételek figyelembevételével történik.



**5.1. ábra.** Alkalmazási példa keresztreteszelés nélküli sorrendi vezérlésre

### 5.2.2 A lépésvezérlés lefutási vázlata

Egy rövid folyamattal, egy négy munkalépésből álló lépésirányítási rendszert jellemeztünk. A negyedik lépés befejezi a lépésirányítást.



**5.3 ábra.** A sorrendi vezérlő lefutási vázlata (Mitsubishi FX kézikönyv)

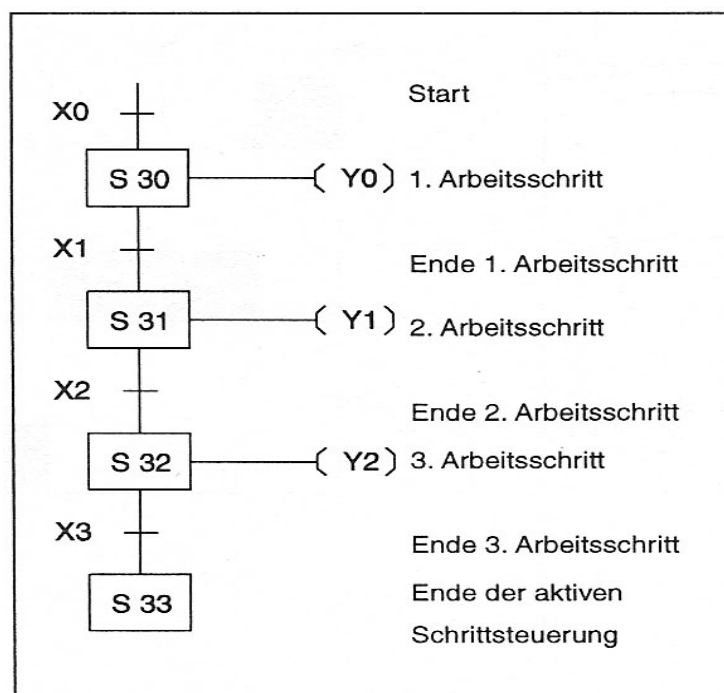


Az 5.3 ábrán az látható, hogy a második munkafolyamat akkor indul, amikor az első befejeződik és amikor a továbblépési feltételt jelentő (1.) kapcsoló bezáródik. Ez azt is jelenti, hogy az első lépés alatt alkalmazott operátorokat visszavonja. Jelen esetben az Y0 kimenet már nem lesz aktív miután elhagytuk az első operátort.

A második szakasz vége és a 2. nyomógomb aktiválása jelenti a harmadik munkafolyamat kezdetét. A negyedik lépésben a harmadik nyomógomb bekapcsolásával a lépésfolyamat befejeződik.

### 5.2.3. A folyamatirányítás bemutatása egy folyamatábrán keresztül

A következő ábrán (5.4. ábra) ugyanaz a folyamatirányítás látható csak folyamatábrán keresztül (IEC-szabvány). A PLC programban, egy lépésben megvalósuló folyamat független a későbbi realizálódástól.



**5.4 ábra.** Alkalmazási példa egy folyamatdiagramra (Mitsubishi FX kézikönyv alapján)



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

### **5.3. A folyamatvezérlés programozása**

- az STL utasítások a lépés operátorokkal együtt vannak alkalmazva. Az S lépésoperátort a következő utasításokkal lehet programozni: LD, LDI, AND, ANI, OR, ORI, OUT, SET, RST.
- folyamatvezérlés nélküli programban az S lépésoperátorokat, mint hagyományos Merkereket lehet alkalmazni.
- egy STL programtartomány kezdetén minden egyes lépésoperátort SET utasítással kell létrehozni.
- A létradiagramon az STL-kapcsolat a bal gyűjtősínen szerepel, és mint főkapcsolat látható.
- Ahogy az STL kapcsolatot visszavonjuk (kiléptünk belőle), a mellette lévő ágat többet nem tudja feldolgozni.
- A RET utasítás a sorrendi programozási mód végét jelenti. Az utasítás kötelezően az utolsó operátoron belül, utolsó utasításként kell szerepeljen. Ez nem feltétlenül jelenti a vezérlési program végét, utána még következhetnek további utasítások, amelyek nem részei a sorrendi programozásnak.
- Egy lépésoperátort a programozás során csak egyszer lehet egy STL utasítással programozni.
- Az STL utasítás nem lehet megszakításkor alkalmazni.
- Egy lépéssztátusz során nem lehet ugróutasítást alkalmazni.
- Az utolsó aktivált lépésoperátort egy RST utasítással kell visszavonni, vagy egy RET utasítással vissza az elejére.

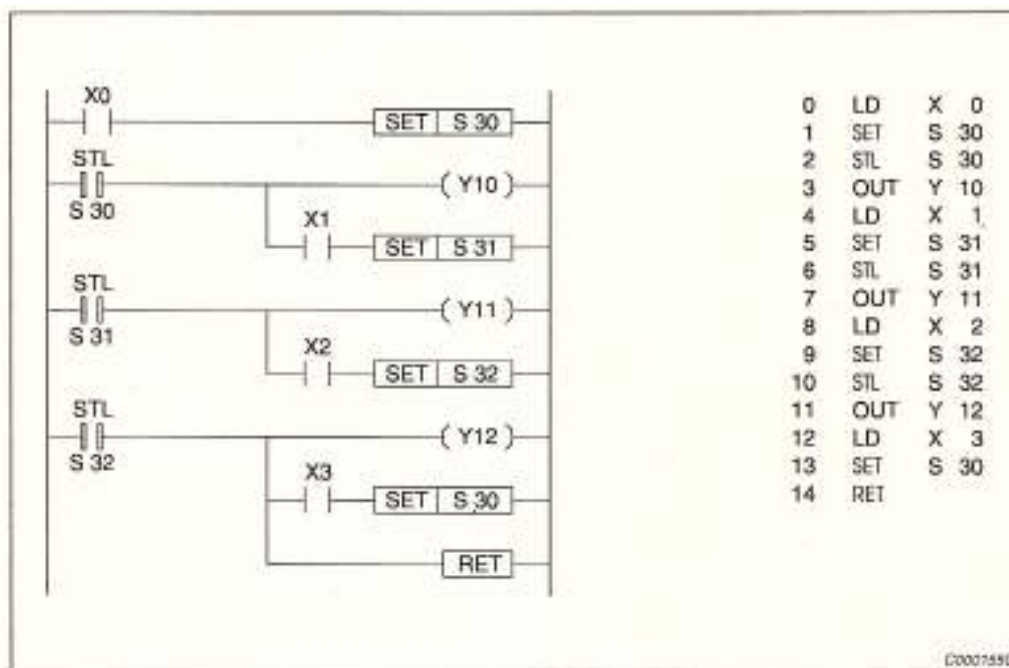
#### ***5.3.1. Engedélyezett utasítások egy lépéssztátuszon belül***

A következő táblázatban alapparancsok utasításai vannak felvezetve, amik az STL utasítások illetve az STL és RET utasítások között szabad alkalmazni

Állapot		Utasítások		
		LD, LDI, OUT, NOP, AND, ANI, SET, RST, OR, ORI, PLS, PLF	ANB, ORB, MPS, MRD, MPP	MC, MCR
Inicializálási helyzet		Engedélyezett	Engedélyezett	Nem engedélyezett
Programelágazás	Kimenetek	Engedélyezett	Engedélyezett	Nem engedélyezett
	Továbbkapcsolási feltételek	Engedélyezett	Nem engedélyezett	Nem engedélyezett

5.1 tábla. Engedélyezett utasítások egy lépéstatusznál

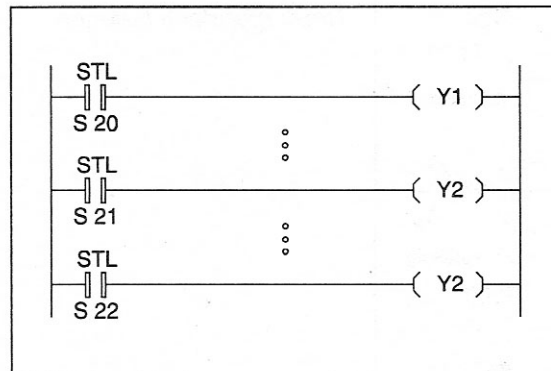
Az 5.5. ábrán egy programrészletet láthatunk, ahol jól látszik a lépések szerkezeti felépítése.



5.5 ábra. Példa STL, RET utasítások alkalmazása (Mitsubishi FX kézikönyv alapján)

### 5.3.2. Többszörösen lefedett kimenetek

Ugyanazokat a kimenet különböző STL utasításokkal illetve lépésoperátorral is el lehet érni, anélkül, hogy a leképzés értelmetlenné váljon.



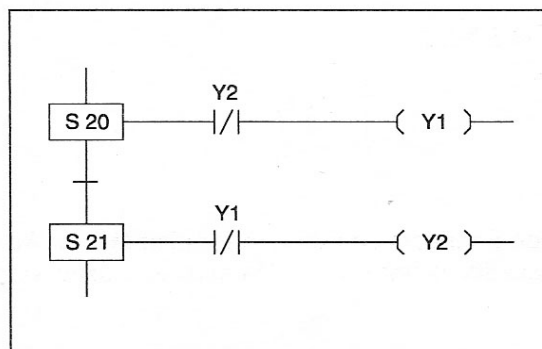
**5.6 ábra.** Többszörösen lefedett kimenetek

A fenti programrészletben ugyanaz a kimenet (Y2) különböző STL utasítással és illetve lépésoperátor S21, S22 is elérhető. Y2 be van kapcsolva, ha S21 vagy S22 aktív. Y2 ki van kapcsolva, ha S21 és S22 nem aktív. A dupla lefedettség ebben az esetben problémamentes, mert a 21 és 22 lépés nem egyszerre aktív.

### **5.3.3. A továbbkapcsolási feltétel visszavonási funkciója**

Ahogy valamelyik S státusz az STL utasításon keresztül megvalósul, az előző státusznak a továbbkapcsolási feltétele visszavonódik. Ez azt jelenti, hogy a programciklus alatt az aktuális és az azt követő státusz egy rövid időre együtt vannak jelen.

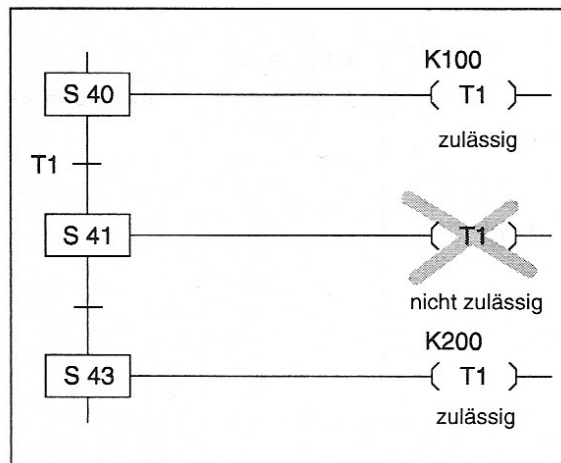
Javaslat: Ha az egymást követő operátoroknak nem szabad egyszerre aktívnak lenniük, javallott, az operátort keresztretesszel védeni, azért hogy az egyidejű bekapcsolt állapotot megakadályozzuk.



**5.7 ábra.** Keresztretesz beékelése,

#### 5.3.4. Timer többszörös lefedése

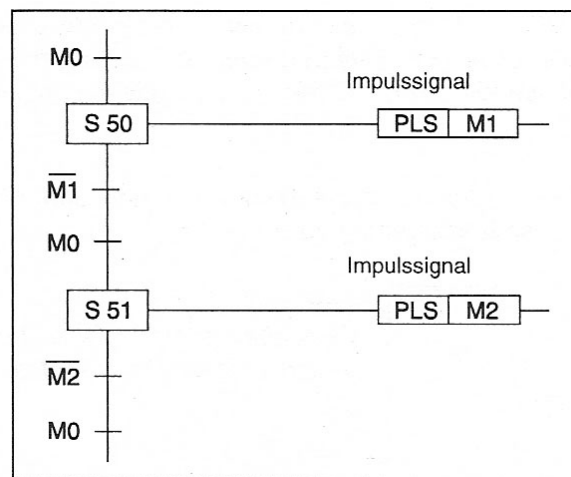
Egy programban a timer a lépésoperátorok alkalmazása közben többször is érintve lehet. Ugyanazt a timert azonban nem szabad két egymást követő lépésben alkalmazni.



**5.8. ábra** Timer többszörös lefedettsége (*Mitsubishi FX kézikönyv alapján*)

#### 5.3.5. Továbblépési feltétel impulzusjelre.

Az egymást követő lépések a továbblépési feltételei impulzusra is aktiválhatók. Ilyenkor egy impulzusutasítás alkalmazása (PLS utasítás) használható.

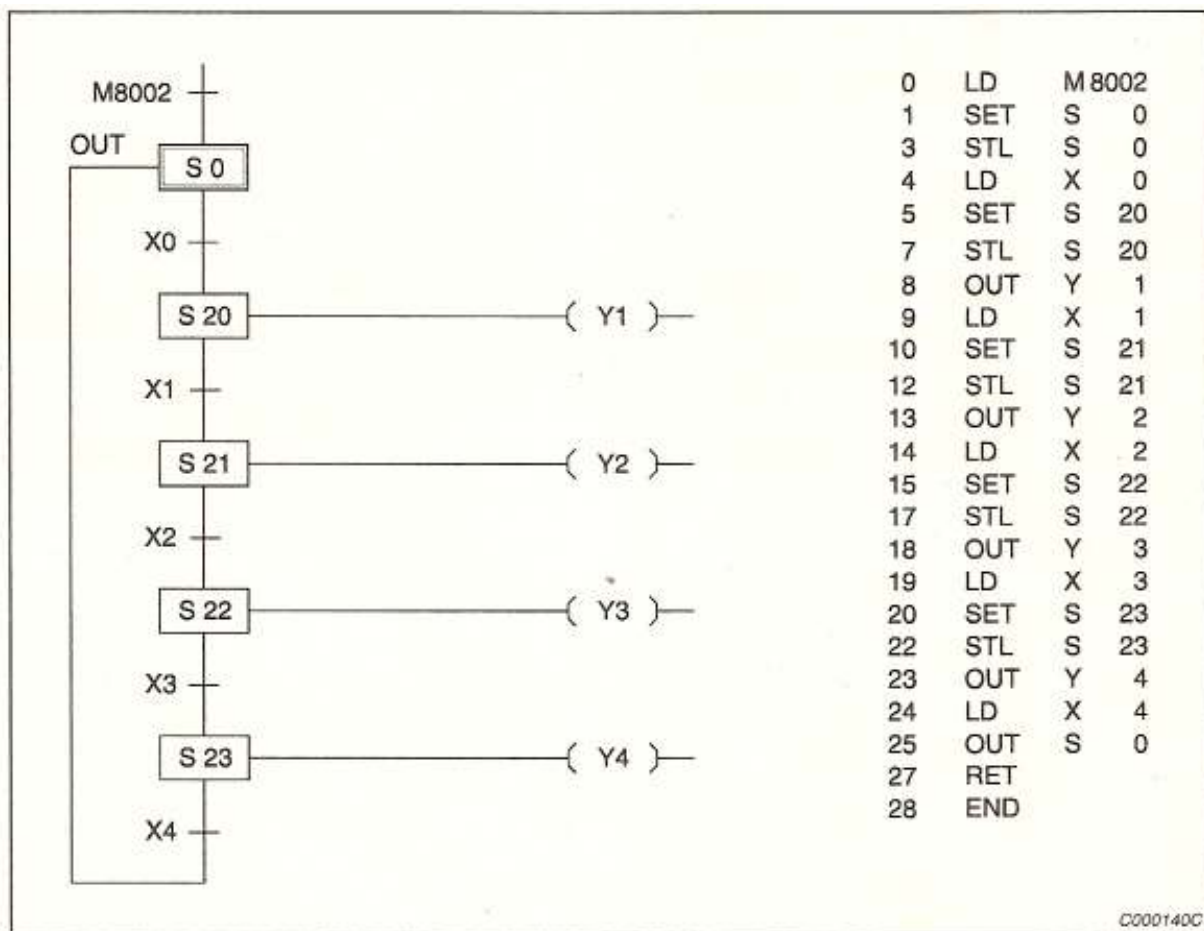


**5.9 ábra.** Továbblépési feltétel impulzusra (*Mitsubishi FX kézikönyv alapján*)

Az első M0 jel aktiválja az S50 lépésstatuszt és M1 bekapcsol. M1 megakadályozza a közvetlen aktiválódását a következő lépésnek. S51 csak akkor aktív, ha a következő M0 jel fennáll és M1 impulzus lejárt.

#### 5.4. Egyszerű sorrendi vezérlés programozása

Minden sorrendi vezérlés az inicializálással kezdődik. Ehhez S0-S9 operátorok állnak rendelkezésünkre. Az inicializáló operátorok segítségével különböző lépésfolyamatot tudunk az STL programozás során előállítani, azért hogy különböző üzemmódokat (kézi- automata üzemmód, nullpont) tudjunk realizálni.



**5.10 ábra.** Példa: lépésstatuszok inicializálása



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

---

Az M8002 (rövid ideig tartó pozitív impulzus) a PLC bekapcsoláskor egy meghatározott kezdeti állapotot hoz létre, amely az S0 operátoron belül valósul meg. A további lépéssztatuszok lépésfeltételeit a már leírt módon hajtja végre. Jelen esetben az inicializáló operátor nem tartalmaz egy műveletet sem, de jelenléte akkor is kötelező.

Azért, hogy a lépéslánc újrakezdését vagy megismétlését okozzuk, ugyancsak az S0-t kell bekapcsolni.

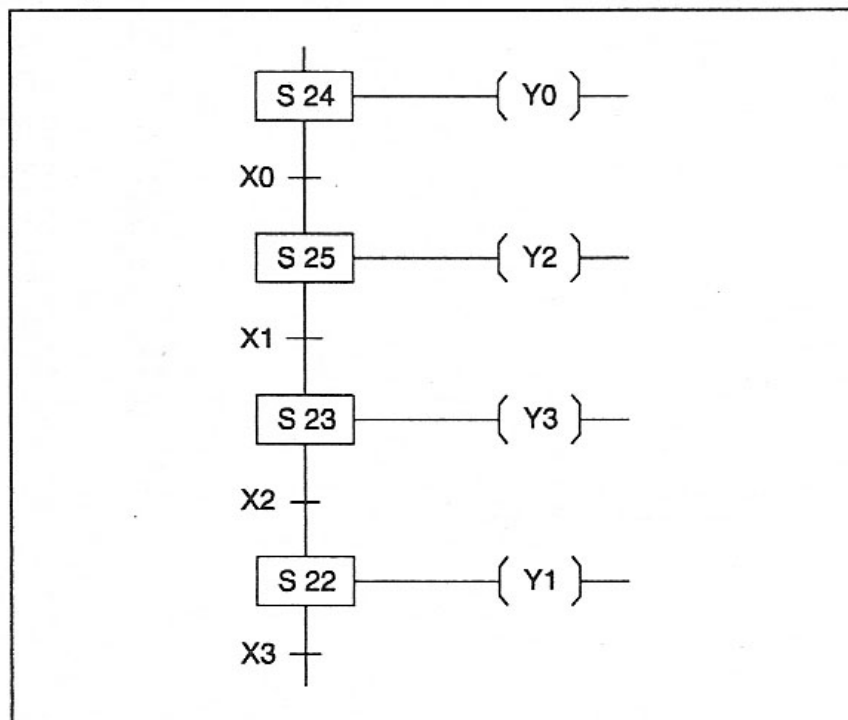
## **5. 5. STL elágazások**

A tároló programozású irányítás Mitsubishi FX családja különböző egymástól független státuszfolyamatokat és elágazásokat tud feldolgozni. Megkülönböztethetünk:

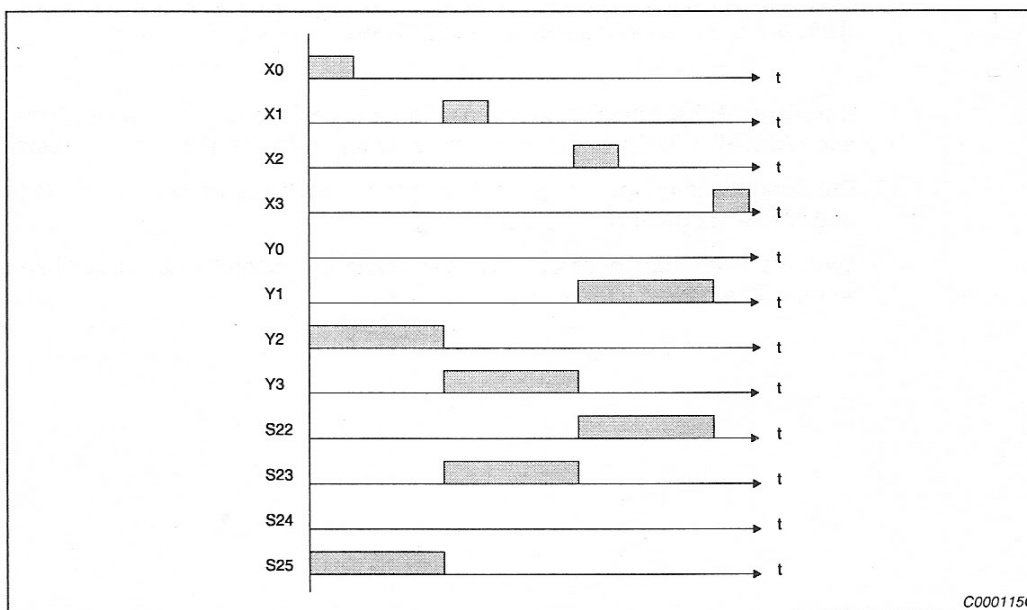
- egyszerű folyamat
- szelektív elágazás
- párhuzamos elágazás
- ugrárelágazás között

### ***5.5.1. Egyszerű folyamat***

Egyszerű folyamatnál a lépéssztatuszok egymás után vannak feldolgozva. A feldolgozás sorrendjét a lépéssztatuszok felállítása határozza meg és ezért független a lépéssztatusz címétől. Az 5.11. ábra egy ilyen egyszerű folyamat sorrendi vezérlését (gráfcetjét) mutatja. Az S24 operátor alatt az Y0 kimenet aktív mindaddig, míg a továbblépési feltétel (X0) nem teljesül. Ekkor válik aktívvá az S25. Az S24 operátor elhagyása után Y0 is törlődik anélkül, hogy erre külön utasítást adnánk.



5.11 ábra. Példa egy egyszerű folyamatra

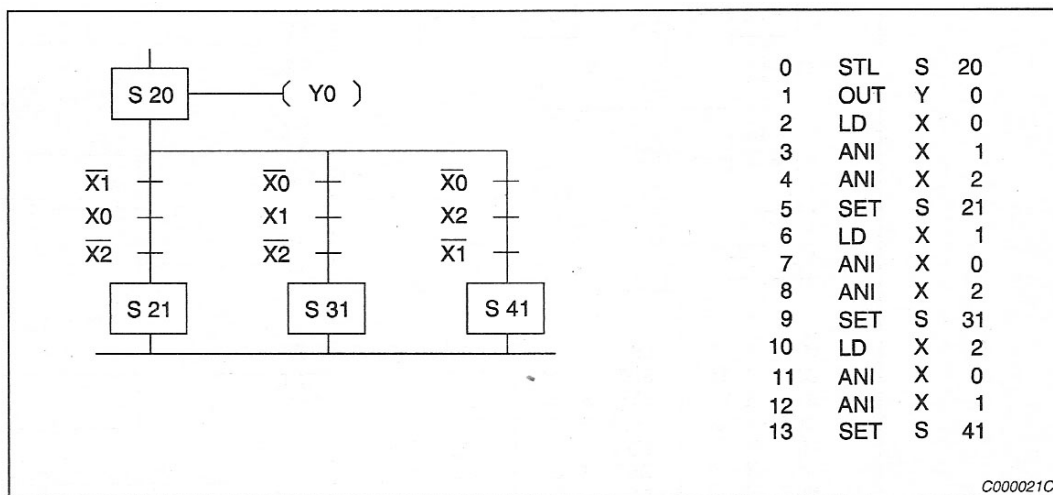


5.12. ábra. A sorrendi vezérlés időbeli lefutása (Mitsubishi FX kézikönyv alapján)



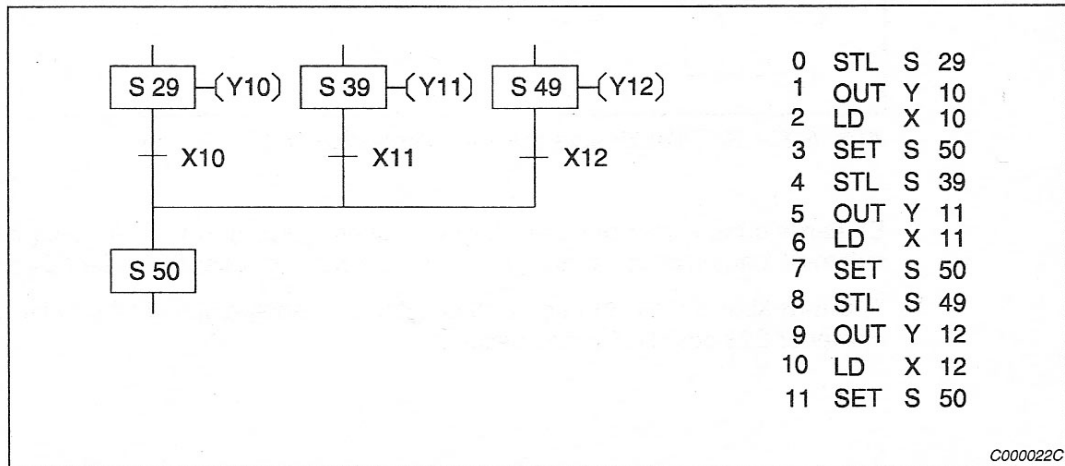
### 5.5.2. szelektív elágazás

A szelektív elágazásnál lehetőség van kettő vagy több státuszfolyamat közötti kiválasztásra az őket meghatározó feltételeknek megfelelően. Egy lépésoperátor a mindenkor belépési feltételeitől függően válik aktívvá. Ha az elágazásban a feltételek különbözőek, akkor ebben az esetben, egy időben csak egy út válik aktívvá. Maximum 8 elágazást definiálhatunk egy operátor programozása során. Az összes elágazások száma pedig nem lépheti túl a 16-ot. (5.13. ábra). Jól látható, hogy az S21, S31 és S41 operátorokba való belépés egyszerre egyidőben nem valósulhat meg.



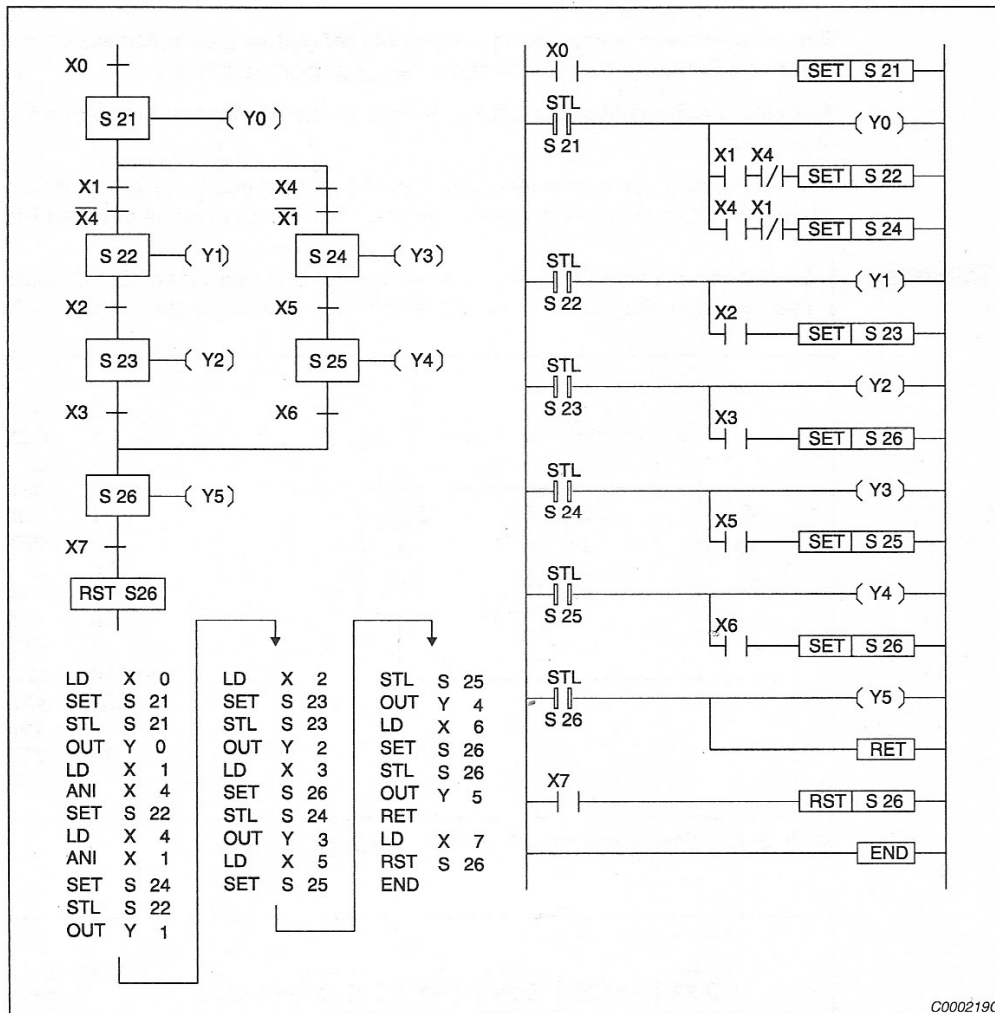
**5.13 ábra.** Szelektív elágazás folyamata

Ha a visszavezetés is feltételhez kötött (X10, X11, X12, 5.14. ábra) az S50 lépés csak az adott ágba teljesülő feltétel után válik aktívvá.



**5.14. ábra.** Szelektív elágazás visszavezetése

A következő példában az elágazást kiváltó feltételek egymást kizáró jellegűek. A két elágazás közül mindig csak egy lehet kiválasztva.

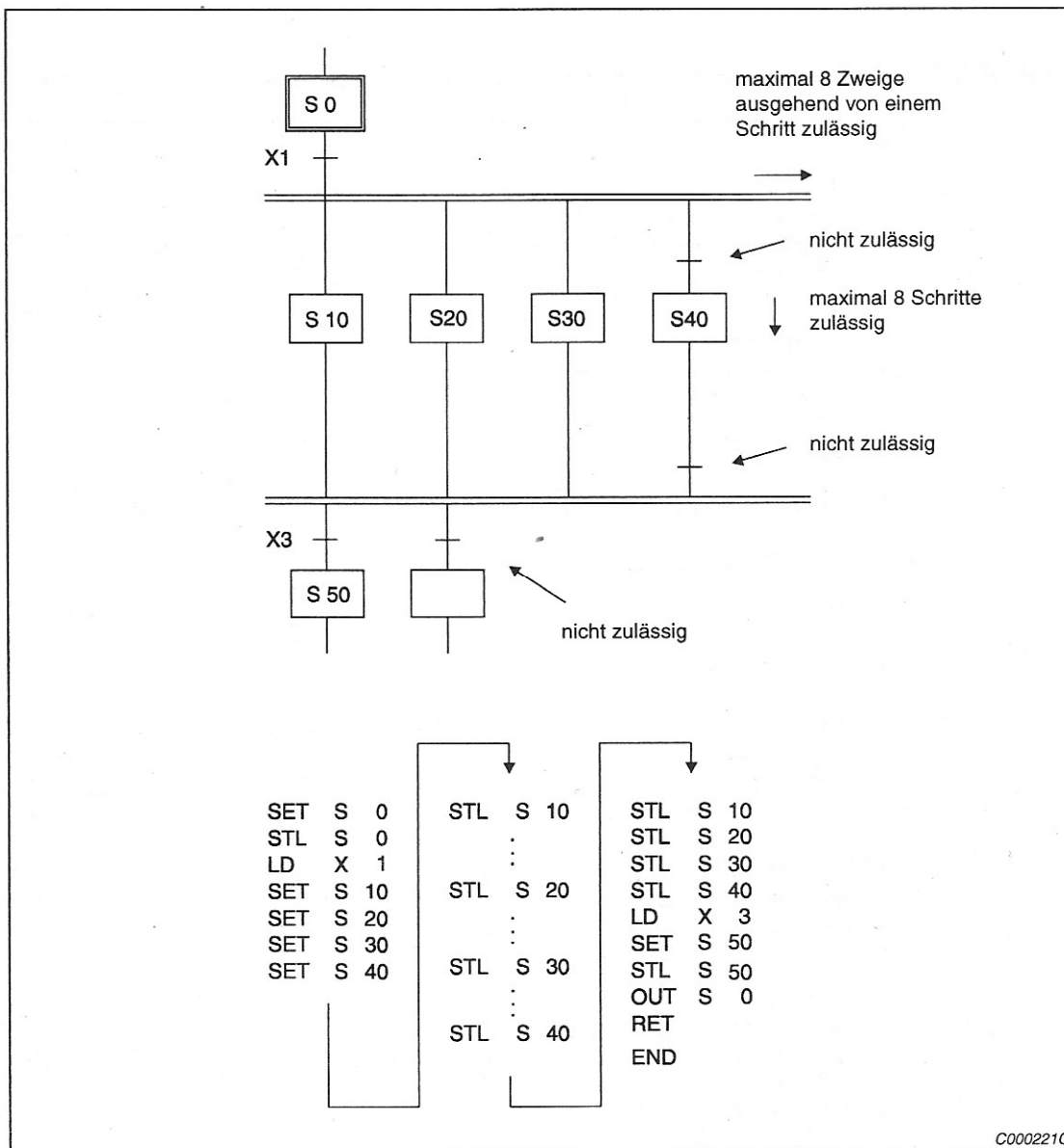


5.15 ábra. Sorrendi folyamatábra és utasításlista egy szelektív elágazásnak

Az  $X1 \cdot \overline{X4} + \overline{X1} \cdot X4$  feltétel azt eredményezi, hogy mindig csak egy funkciót lehet véghez vinni. Az S21 automatikusan visszavonódik, ha vagy a S22 vagy S24 aktív, illetve S26 az S23 vagy az S25-re aktív. Ennek megfelelően az S26-tól vagy az S23 vagy az S25 kerül visszavonásra.

### 5.5.3. párhuzamos elágazás

A párhuzamos elágazásnál kettő vagy több státuszfolyamat egyszerre van feldolgozva. A státuszon kívül egy elágazás több státuszfolyamatban történik. Az összes elágazások száma maximum 16 lehet. (5.16. ábra)



5.16 ábra. Engedélyezett párhuzamos elágazás (Mitsubishi FX kézikönyv alapján)



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

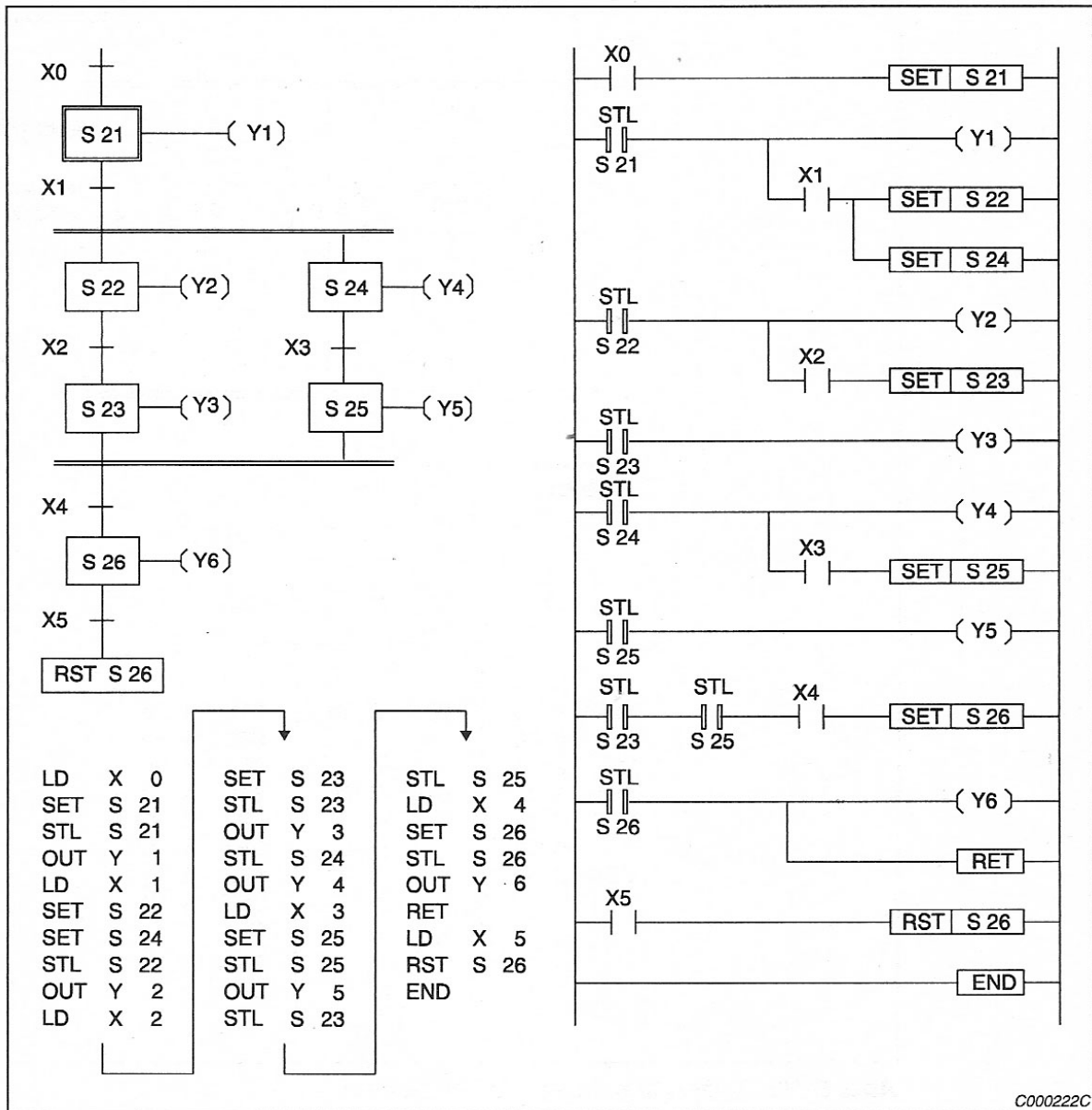
---

A mindenkor alkalmazott belépési feltételtől függően az elágazások a saját ösvényükben kerülnek végrehajtásra. Ellentétben a szelektív elágazással a párhuzamos elágazásnál több státusfolyamatot is lehet egyidőben aktív. A használt operátorok a párhuzamos lépéseknél csak akkor kerülnek visszavonásra, ha először az egymás mellett lévő lépéseket összekapcsolta és feldolgozta.

Megjegyzés: Az elágazás után és a visszavezetés előtt nem lehet kapcsolódás!

Egy párhuzamos elágazás maximum nyolc ágat tartalmazhat, amiktől minden ág maximum nyolc egymást követő lépésből állhat. Párhuzamos elágazás közben további szelektív elágazást már nem programozhatunk.

Az 5.17. ábrán egy párhuzamos elágazás teljes programrészletét látjuk.

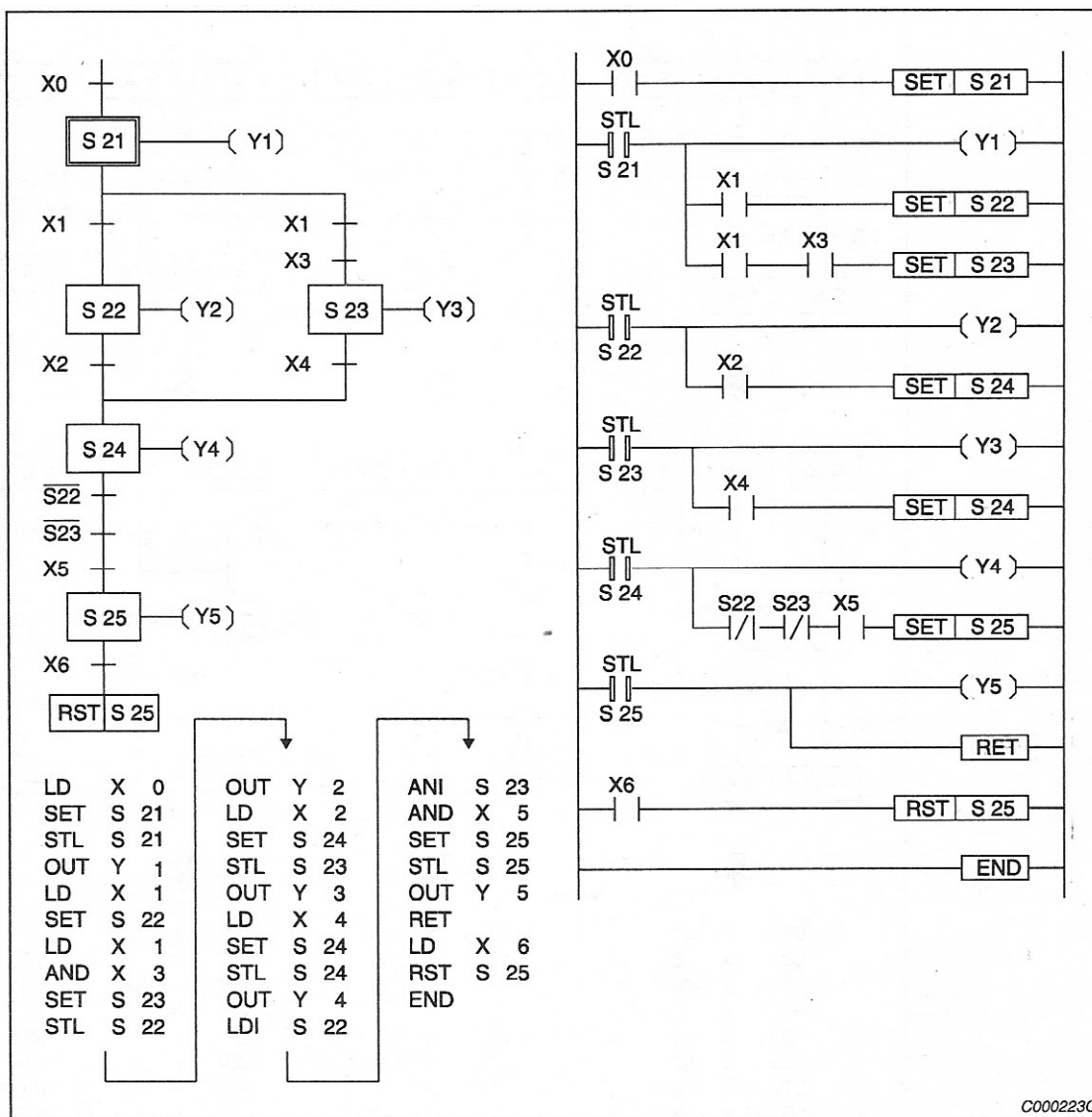


5.17. ábra. Párhuzamos elágazás (Mitsubishi FX kézikönyv alapján)

Az X1 feltétel mindkét ág végrehajtását elindítja (S22, S24), míg az S26 lépés X4-től függően csak az S23 és S25 megvalósulása után kivitelezhető.

#### 5.5.4. Szelektív és párhuzamos elágazások kombinációi

A szelektív és párhuzamos elágazás a programozás során kombinálható. Egy ilyen megoldást mutat az 5.18. ábra.



**5.18. ábra** Példa egy szelektív és párhuzamos elágazás kombinációjára

Ha van X3, a párhuzamos elágazás feltétele teljesült. Ha nincs, akkor végbemegy egy szelektív programfeldolgozás, ezért S24 az S22 után lefut. S24 csak akkor alkalmazható, ha



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

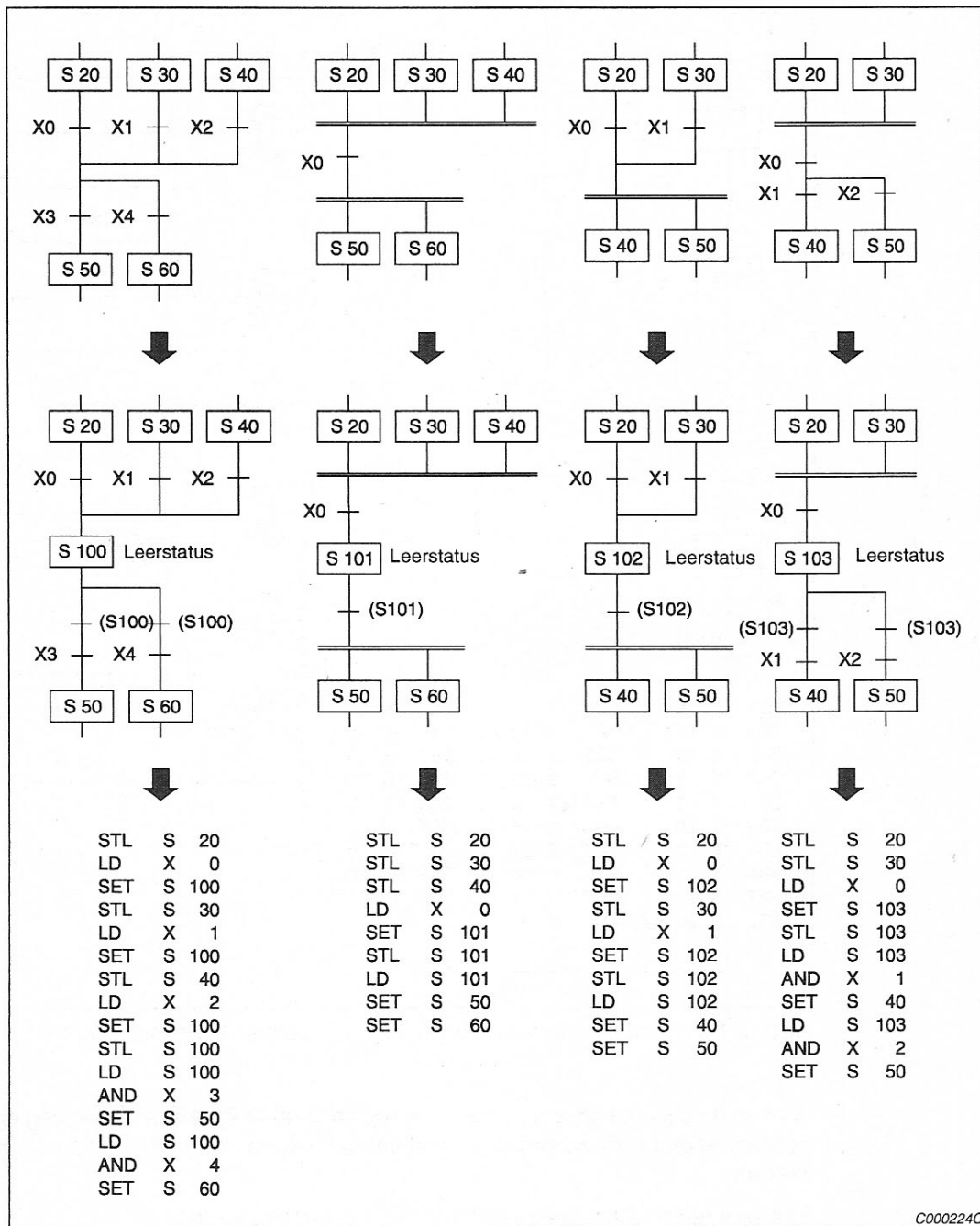
---

S22 vagy S23 vissza van vonva és teljesülnek a visszatérési feltételek (X2 vagy X4)  
Ugyanígy S25 is csak akkor lesz aktív, ha S22 és S23 visszavonásra került.

#### ***5.5.5. Üres státusz programozása***

Néhány lépésfolyamat végbemeneteléhez üres státuszok programozása szükséges. Ez a lehetőség egyrészt biztosítja egy elágazás visszavezetése utáni azonnali, feltétel nélküli, újabb elágazás bevezetését, másrészt egy jobb áttekinthetőséget biztosít a programfolyamatoknál. Az 5.19. ábrán az S100, S101, S102 és S103 operátorok ugyan jelen vannak, de nem tartalmaznak egyetlen egy műveletet sem.



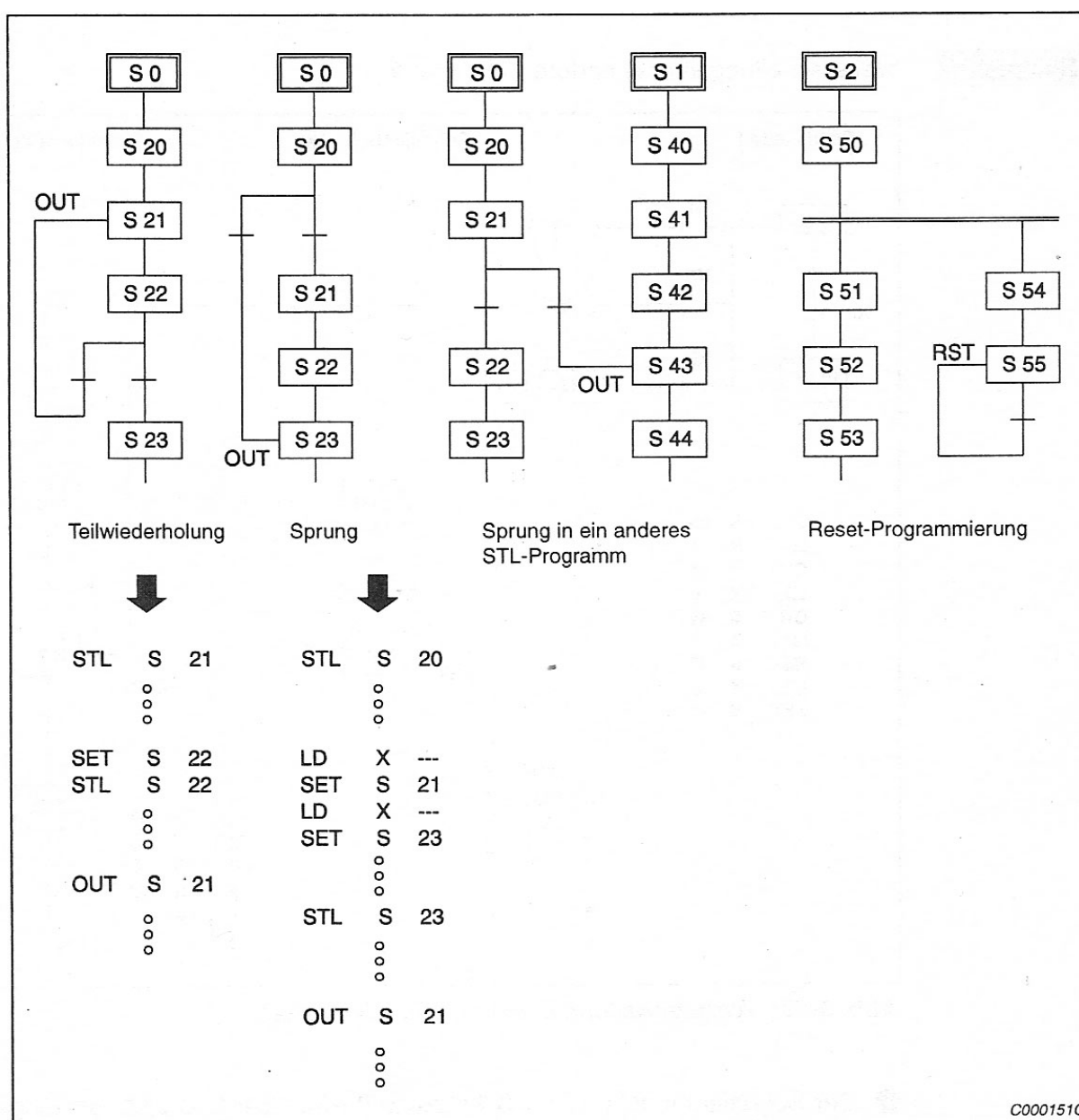


C000224C

5.19. ábra. Üres státusz programozása (Mitsubishi FX kézikönyv alapján)

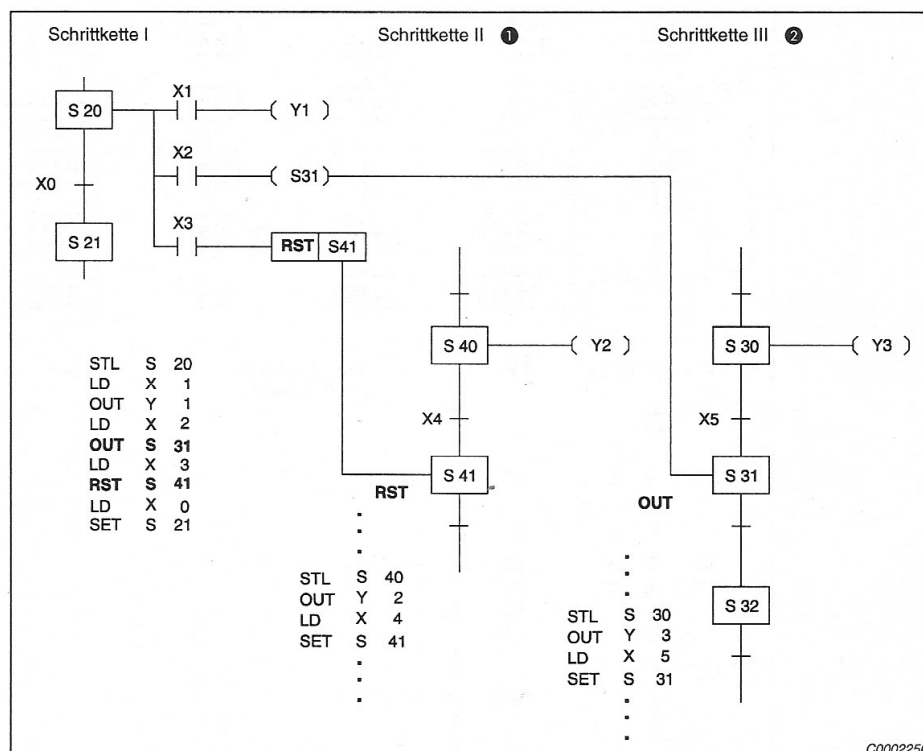
### 5.5.6. Ugró elágazások

Az ugró elágazások lehetőséget biztosítanak a résztartományok státuszsorrendjének az átugrására, vagy bizonyos programrészletek többszöri, egymás utáni futtatására.



**5.20. ábra.** Példa, különböző lehetőségekre ugráselágazáskor (Mitsubishi FX kézikönyv alapján)

Egy lépésláncnak a továbbkapcsolását egy másik lépésláncba SET utasítások helyett OUT utasítással is lehet programozni. Lásd az **OUT S31** utasítást az 5.21. ábrán bemutatott példa I lépésláncát. Láthatjuk, hogy az utasítás hatására a III. programrészlet S31 státuszára ugrottunk. Ennek az alternatívának nincs kihatása az irányítás belső programfeldolgozására.



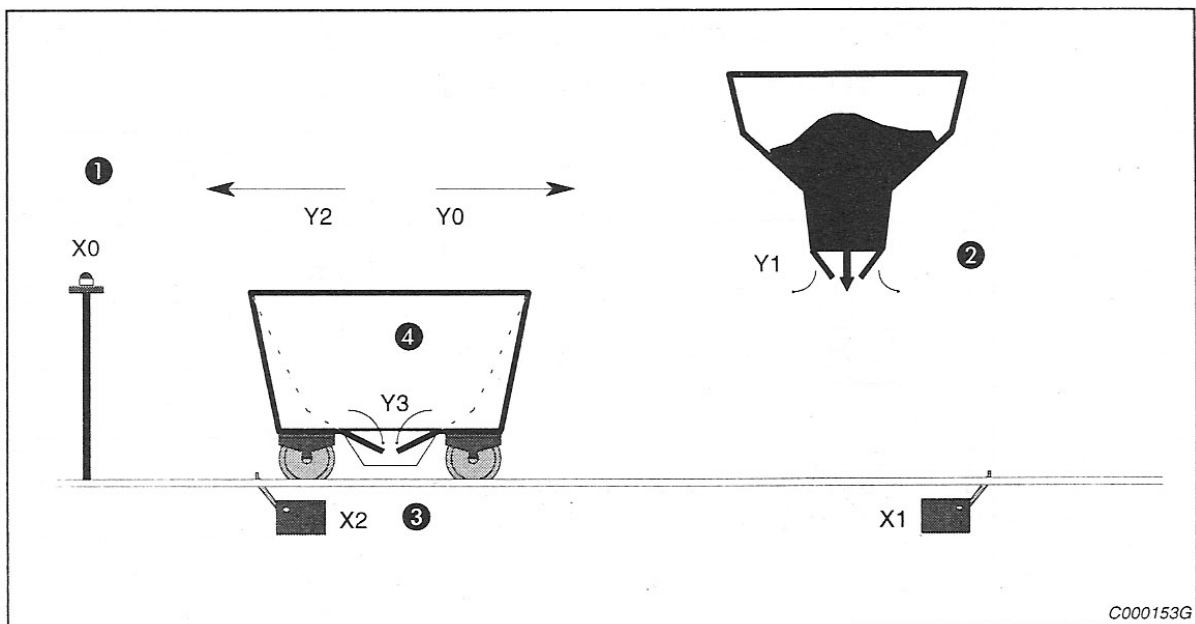
**5.21. ábra.** Továbbkapcsolódás egy másik lépésláncba.

Érdemes megfigyelnünk még, a következőket:

1. Az S41 lépéshelyzet a II. lépésláncban az S40-re és az X4 bemenetre van alkalmazva, viszont visszavonásra kerül, ha S20 és az X3 bejárat az I. lépésláncban be van kapcsolva. Ha a visszavonási folyamat befejeződött, a lépéslánc a S20 lépéshelyzetben található meg a továbbiakban, amelynek S41-től nincs befolyása.
2. A III. lépésláncban az S31 lépéshelyzet alkalmazva van, ha S20 és az X2 bemenet az I. lépésláncban aktív. S31 visszavonásra kerül, miután S32 továbbkapcsolták. Az S20-as lépés vissza van vonva, ha ugrottunk az S31-re.

## 5.6. Konkrét feladat sorrendi programozásra

Ebben a példában egy ömlesztett-áru konténert egy rögzített kocsihoz szállítják és az előre meghatározott helyen be-, illetve kirakodják.



5.22. ábra. Példa egy be- és kirakodás vezérlésre egy konténernél

1. az X0 startgomb hatására a kocsi a berakodási helyre megy és megáll az X1 végálláskapcsolónál.
2. a siló hét másodpercre rá nyílik (Y1)
3. a kocsi visszamegy (Y2) és megáll az X2 végálláskapcsolóra a kirakodás helyénél
4. a kocsi kirakodó ajtaja öt másodpercre kinyílik (Y3)

A feladat egyszerű lépéssorozatokból tevődik össze. Automatikusan indul, ezért az M8002 alapimpulzust, illetve az S0 inicializáló operátort használja indításra. A berakodás X0-ra indul.

Bemeneti elemek:

X2 → kirakodási helyzet

X0 → indítás

X1 → berakodási helyzet



**EFOP-3.5.1-16-2017-00017**

**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

*Kimeneti elemek:*

Y0 → kocsى jobbra

Y1 → siló ajtó nyit

T1 → siló ajtó nyitva 7s-ig.

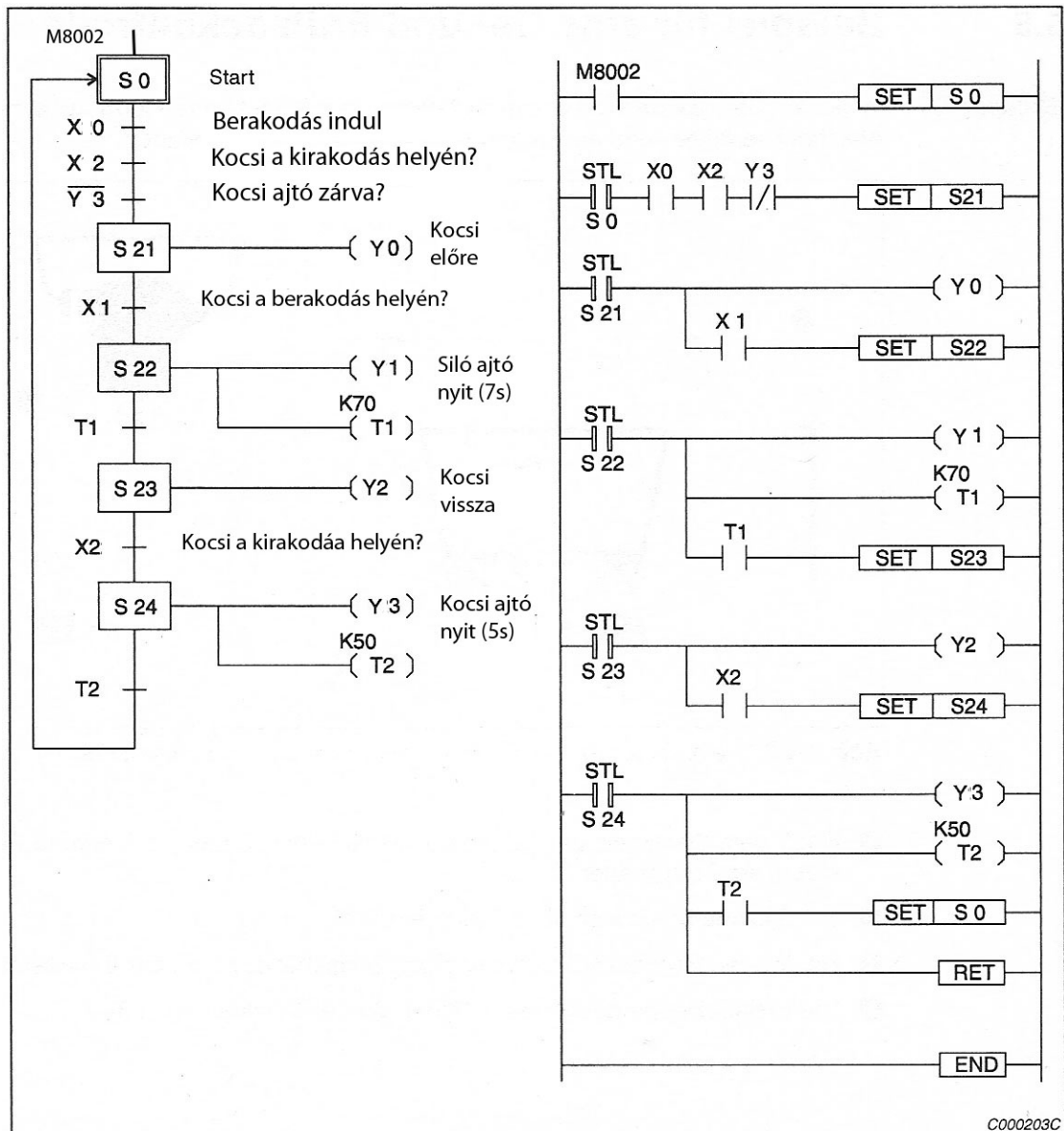
Y2 → kocsى balra

Y3 → kocsى ajtó nyit

T2 → kocsى ajtó nyitva 5s-ig

Az 5.23. ábrán láthatjuk a feladat folyamatábráját és létradiagramját. A feladat négy jól meghatározott lépésre tagolódik:

1. S21 → kocsى előre (Y0)
2. S22 → siló ajtó nyit 7 másodpercre (Y1 és T1)
3. S23 → kocsى vissza (Y2)
4. S24 → kocsى ajtó nyit 5 másodpercre (Y3 T2)



5.23. ábra. Az előző feladat megoldása (Mitsubishi FX kézikönyv alapján)

A létradiagram alapján elkészíthető az utasításlistás programrészlet is. Ezt a feladatot az olvasóra bízom!

(Készült a MITSUBISHI ELECTRIC **Melsec FX család** kezelési útmutatója alapján)



EFOP-3.5.1-16-2017-00017

**„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”**

## 6. GYAKORLÓ FELADATOK

### 1. feladat

**Háromfázisú aszinkron motor Y  $\rightarrow$   $\Delta$  indítása áramérzékeléssel, keresztretesszel és hőrelés védelemmel.** A motor indítása (I) és megállítása (S) nyomógombokkal történik, de csak akkor, ha előzőleg a hidraulikus rendszert (H), valamint a villamos áramköröket a START gombbal elindítottuk. Az olajnyomást a P szenzor érzékeli. A motor áramreléje (Ra) NO, a hőreléje (Th) NC érintkezővel rendelkezik. Induláskor a motor csillagban van (Ky) mindaddig, míg az áramrelé érintkezője nem jelzi az áram csökkenését, utána deltába kapcsol. ( $K_{\Delta}$ ) A motor tekercseinek táplálása a K mágneskapcsoló érintkezőin keresztül történik. A kapcsolást úgy kell megvalósítani, hogy még véletlenül se fordulhasson elő, hogy mindkét üzemmód egyidőben működhessen. (keresztretesz) A hőrelé kioldásakor, vagy a vészgomb megnyomásakor azonnal minden leáll és megszólal a vészjelző kürt (VK).

- Programterv, vezérlési állapotegyenletek, változók címezése.
- Létradiagram elkészítése
- PLC bekötési vázlata.
- Utasítássoros programrészlet.

### 2. feladat

**Háromfázisú aszinkron motor Y  $\rightarrow$   $\Delta$  indítása időzítéssel, keresztretesszel és hőrelés védelemmel.** A motor indítása és megállítása nyomógombokkal történjen. A hőrelé bontó (NC) érintkezővel rendelkezik. Induláskor a motor 5 s-ig csillagban van (Ky), majd átkapcsol deltába ( $K_{\Delta}$ ) A motor tekercseinek táplálása a K mágneskapcsoló érintkezőin keresztül történik. A kapcsolást úgy kell megvalósítani, hogy még véletlenül se fordulhasson elő, hogy mindkét üzemmód egyidőben működhessen. (keresztretesz)

- Programterv, vezérlési állapotegyenletek, változók címezése.
- Létradiagram elkészítése
- PLC bekötési vázlata.
- Utasítássoros programrészlet.

### 3. feladat

**Lépcsőházi világítás-automata és kapunyitó rendszer.** A világítást három különböző helyről lehet felkapcsolni nyomógombokkal ( $v_1, v_2, v_3$ ). A világítás időtartama 60 s. A kapunyitó elektromágnes két helyről ( $k_1, k_2$ ) nyomógombokkal lehet működtetni. Működtetés





**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

után 10 s-ig engedi az ajtó nyitását. Az ajtó nyitott állapota is a világítás felkapcsolásához vezet és bezárás után ugyancsak 60 s-ig kell működjön a világítás. A nyitott kapu állapotát lámpa jelzi a kapunyitó nyomógombok mellett.

- Programterv, vezérlési állapotegyenletek, változók címezése.
- Létradiagram elkészítése
- PLC bekötési vázlata.
- Utasítássoros programrészlet.

#### **4. feladat**

**Irányváltó kapcsolás keresztretesszel, végállás-kapcsolókkal és 3 s-os átkapcsolási késleltetéssel.** Az irányok bekapcsolása valamint a kikapcsolás nyomógombokkal (BAL, JOBB, STOP) kell történjen. Az irányváltás csak kikapcsolt helyzetben történhet. Ha ellentétes irány kerül bekapcsolásra, ez a állítás után 3 seltelte után indulhat és nem fordulhat elő még véletlenül sem, hogy egyszerre mindkét irány indítható legyen. A végállás-kapcsolók leállítják az adott irányt, ilyenkor csak az ellenkező irány működtethető.

- Programterv, vezérlési állapotegyenletek, változók címezése.
- Létradiagram elkészítése
- PLC bekötési vázlata.
- Utasítássoros programrészlet.

#### **5. feladat**

**Szállítószalag kézi és automatikus működtetése.** Kézi üzemmódban az indítás és megállítás nyomógombokról történik. Az üzemmódok közötti átkapcsolás csak leállított állapotban történhet. Automata üzemmódban a szállítószalag két végén található érzékelők ( $E_1$ ,  $E_2$ ) indítják, illetve állítják meg a szalagot.  $E_1$  elindítja,  $E_2$  pedig megállítja a szalagot, ha újabb érkező tárgyat nem érzékel  $E_1$ . Vagyis  $E_2$  csak akkor kapcsolhatja ki a szalagot, ha már az utolsó tárgy is elérkezett  $E_2$ -höz. A motor hőrelés védelemmel van ellátva, amely le kell állítsa a szalagot kézi és automata üzemmódban egyaránt. A hőrelé bekapcsolása egy visszajelző lámpán (VL) látható kell legyen.

- Programterv, vezérlési állapotegyenletek, változók címezése.
- Létradiagram elkészítése
- PLC bekötési vázlata.
- Utasítássoros programrészlet.



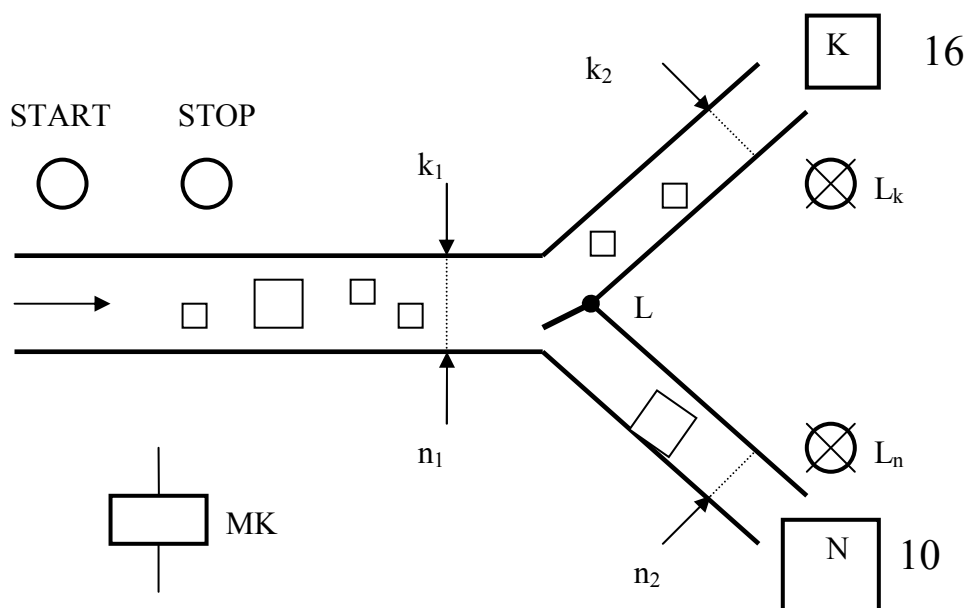
## 6. feladat

**Fénysorompó működtetése.** Kézi üzemmód: Nyitás (FEL), zárás (LE) nyomógombról történik és végállás-kapcsolók (F, L) állítják meg. Automata üzemmódban az érkezés érzékelő (Ee) nyitja, majd az áthaladás érzékelő (Ae) jelzése után 3 másodperces késleltetéssel indul a sorompó zárása. A nyitott és zárt állapotokat zöld és piros jelzőlámpák jelzik. A lámpa mindaddig piros, amíg sorompó egészen fel nem emelkedett. Itt átkapcsol a zöld lámpa és addig világít, míg a sorompó meg nem indul lefelé. (újra pirosra vált)

- Programterv, vezérlési állapotegyenletek, változók címezése.
- Létradiagram elkészítése
- PLC bekötési vázlata.
- Utasítássoros programrészlet.

## 7. feladat

Egy szállítószalagon kicsi és nagyméretű munkadarabokat válogatnak.  $k_1$  érzékeli a kicsiket,  $n_1$  pedig a nagyokat. Érzékelés után a szalag elágazik két irányba, az L terelőlemez pedig a kicsiket K irányba, a nagyokat N irányba tereli. A szalag végein dobozokba gyűjtik a munkadarabokat. A kicsiből 16, a nagyokból 10 darab kerülhet egy dobozba. A számlálást a  $k_2$  és  $n_2$  érzékelők segítik. ( $k_2$  a kicsiket,  $n_2$  a nagyokat) Amikor egy doboz megtelik, a szalag megáll 10 másodpercre (addig a dobozt kicserélik), és kigyullad az  $L_k$  (kicsi doboz megtelt), vagy az  $L_n$  (nagydoboz megtelt) lámpa, majd automatikusan újraindul. A lámpák csak addig világítanak, amíg áll a szalag. A szalagot az MK mágneskapcsoló által működtetett motor hajtja. A motor START gombra indul. Megállítani vészhelyzet esetén a STOP gombbal lehet. Az L terelő lemezt egy rugó ellenében működő elektromágnes működteti, amely alaphelyzetben (nincs működtetve) K (kicsik) irányba tereli a munkadarabokat.





**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

- Programterv, vezérlési állapotegyenletek, változók címzése.
- Létradiagram elkészítése
- PLC bekötési vázlata.
- Utasítássoros programrészlet.

## **8. feladat**

Közlekedési jelzőlámpa működtetése. Normál üzemmódban a lámparendszer bekapcsolás után azonnal indul. A PIROS lámpa 10 s-ig világít, majd követi a SÁRGA 1,5 s-ig, utána pedig a ZÖLD 12 s-ig. A zöld lámpát ismét a sárga követi, 2 s időtartamig, de úgy, hogy a zöld periódus végén, 1 s-ig mindkettő világít, majd ismét a piros következik. Üzemzavar esetén, (üzemzavar érzékelő aktív) a ciklikus működés megszakad, és SÁRGA-VILLOGÓ üzemmódba lép, amikor a sárga lámpa 1 s-ig világít, 1 s-ig sötét. A zavar megszűnése után automatikusan visszaáll a ciklikus üzemmód.

- Programterv, vezérlési állapotegyenletek, változók címzése.
- Létradiagram elkészítése
- PLC bekötési vázlata.
- Utasítássoros programrészlet.

## **9. feladat**

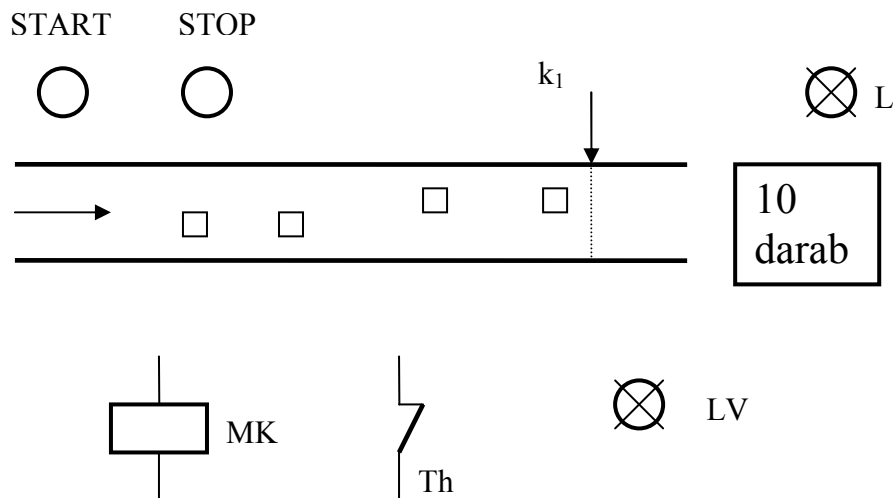
Egy kb. 100 m hosszú útszakasz félpályás lezárását, mindkét irányból piros és zöld jelzőlámpák irányítják. A zöld jelzés időtartama mindkét irányból 120 másodperc. Értelmszerűen, ha az egyik oldalon a zöld lámpa világít, akkor a másik oldalon piros a jelzés. Figyelembe kell veyük azonban azt a helyzetet, hogy a lezárt útszakasz megtételéhez kb. 30 másodperc szükséges. Ez azt feltételezi, hogy amikor a lámpák zöldről pirosra váltanak valamelyik oldalon, a másik oldalon még 30 mp.-ig ugyancsak piros kell legyen a jelzés, vagyis ezalatt mindkét lámpa pirosat mutat.

- Programterv, vezérlési állapotegyenletek, változók címzése.
- Létradiagram elkészítése
- PLC bekötési vázlata.
- Utasítássoros programrészlet.

## 10. feladat

Egy szállítószalag végén a kész termékek csomagolása történik. Egy dobozba 10 darab termék kerülhet. A  $k_1$  érzékelő vezérli a számlálást. Amikor a 10-edik darab is bekerült a dobozba, a szalag 20 másodpercre megáll, ameddig dobozt cserélnek, majd automatikusan újraindul. A leállás idején L lámpának világítania kell

A szalagot az MK mágneskapcsoló által működtetett motor hajtja. A motor kezdetben a START gombra indul. Megállítani vészhelyzet esetén a STOP gombbal lehet. A motor hőrelés védelemmel (Th) van ellátva, melynek kioldása esetén LV lámpa világít és meg kell álljon a motor!



- Programterv, vezérlési állapotegyenletek, változók címezése.
- PLC bekötési vázlata.
- Létradiagram elkészítése
- Utasítássoros programrészlet.

## 11. feladat

Egy prés gép villamos és hidraulikus rendszerét a K mágneskapcsoló működteti, amely a START nyomógombra indul, STOP vagy vészgombra megáll. A motor túláramvédelmét TH hőrelé biztosítja. A prés indítását biztonsági okokból úgy kell megoldani, hogy az egymástól 1,2 méter távolságra elhelyezett nyomógombokat (S1, S2) egyszerre, egyidőben legalább 1s-ig nyomva tartjuk (KÉTKÉZES INDÍTÁS). Ezen kívül figyelembe kell veyük, hogy alaphelyzetben (fent) van-e a prés, valamint a biztonsági fotocella (FB) nem jelez akadályt és van hidraulikus olajnyomás (p). A présfej le-fel mozgását KL és KF mágneskapcsolók által vezérelt hidraulikus szelepek működtetik. XF a fenti és XL a lenti helyzetérzékelők. A prés lenti helyzetben 1,2s-ig marad, majd automatikusan visszatér fenti alaphelyzetébe. Ha mozgás közben, bármikor a biztonsági fotocella akadályt jelez, vagy a vészkapcsolót benyomjuk, a



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

prés azonnal megszakít minden más műveletet, alaphelyzetbe áll, lekapcsolja a hidraulikát és kigyullad a sárga villogó vészjelző lámpa. Vészjelzés után újraindítani csak a veszély elhárítása és a RESET gomb megnyomása után lehet.

- Programterv, vezérlési állapotegyenletek, változók címzése.
- PLC bekötési vázlata.
- Létradiagram elkészítése
- Utasítássoros programrészlet.

## **12. feladat**

Egy ipari folyamat vízellátását az S mágnesszelepen keresztül egy T tartály biztosítja. A tartályba a vizet az M villanymotor által hajtott szivattyú tölti. A motor táplálása a KM mágneskapcsolón keresztül történik. A motornak akkor kell működnie, amikor az alacsony vízszint érzékelő (LV) bekapcsol, de a vízoszlop magassága még nem érte el a magas vízszint (HV) által érzékelt szintet, valamint indítható és megállítható kell legyen az MB illetve az MK nyomógombokból is. (Indítani nyilván csak akkor, amikor nincs tele a tartály, megállítani bármikor lehet.) A motor túláramvédelmét a TH hőrelé biztosítja. Biztonsági okokból a tartályban még van egy UT üres tartályérzékelő is, ennek érzékelési szintje az LV érzékelő alatt van. Az S szelepet nyitni és zárni az SB illetve az SK nyomógombokkal lehet, vagy automatikusan 90 másodpercenként 30 másodpercig működtetve. Nyitni csak akkor, amikor nem üres a tartály. Üzemzavar esetén (túláram, vagy üres tartály) a VL lámpa világít.

- Programterv, vezérlési állapotegyenletek, változók címzése.
- PLC bekötési vázlata.
- Létradiagram elkészítése
- Utasítássoros programrészlet.



**EFOP-3.5.1-16-2017-00017**

***„NYE-DUÁL- Új utakon a duális felsőoktatással a Nyíregyházi Egyetemen,  
az Északkelet-Magyarországi térség felemelkedéséért”***

---

## **FELHASZNÁLT SZAKIRODALOM**

- [1] Ajtonyi I., Gyuricza I.: Programozható irányítóberendezések, hálózatok és rendszerek, Műszaki könyvkiadó Budapest, 2002.
- [2] Mitsubishi Electric: A Melsec FX család, programozható logikai vezérlők kezelési útmutató, 2006.
- [3] Mitsubishi Electric: GX Developer, programozó és dokumentáló rendszer, 2009.