

**PLC programozás az  
IEC 1131-3  
szabvány szerint**

**Jancskárné Anweiler Ildikó**  
főiskolai adjunktus  
PTE – PMMFK  
Műszaki Informatika Tanszék



## Tartalomjegyzék

<b>Az IEC-1131-3 szabvány</b> .....	<b>6</b>
<b>A programszervezési egységek felépítése</b> .....	<b>6</b>
A változók deklarálása .....	6
Példa egy tipikus változódeklarációra.....	7
A programszervezési egység törzse .....	7
Az IEC-1131-3 szabványban ajánlott programozási nyelvek .....	7
<b>Az IEC programozói környezet</b> .....	<b>9</b>
<b>Erőforrás elosztás</b> .....	<b>9</b>
<b>A programszervezési egységekről részletesen</b> .....	<b>11</b>
A programszervezési egység részei.....	11
Példa a programszervezési egység felépítésére (függvényblokk).....	12
Deklaráció .....	12
Változótípusok .....	13
A szervezési egységek kapcsolódási felületeinek jellegzetességei .....	13
A formális paraméter és a visszatérési érték értelmezése .....	13
Példa a FB formális paramétereinek belső és külső értelmezésére .....	14
<b>A függvényblokk</b> .....	<b>15</b>
Hordozhatóság és objektum orientáltság.....	15
A függvényblokkban használható változótípusok.....	15
<b>A függvény</b> .....	<b>15</b>
A függvény változótípusai és a függvényérték .....	16
<b>A program</b> .....	<b>16</b>
<b>Nyelvi elemek, adattípusok, változók</b> .....	<b>17</b>
Egyszerű nyelvi elemek .....	17
Foglalt kulcsszavak .....	17
A különböző adattípusok számbázisra ábrázolása.....	17
A konstansok áttekintése .....	18
A felhasználó által definiálható nevek, címkék .....	18
<b>Változók és adattípusok</b> .....	<b>19</b>
A változódeklaráció legfontosabb elemei .....	19
<b>Adattípusok</b> .....	<b>19</b>
Elemi adattípusok.....	19
Származtatott adattípusok .....	20
Általános adattípusok .....	21
<b>A változóattribútumok</b> .....	<b>21</b>
Példa az attribútumok használatára .....	22
<b>Közvetlen címzésű változók</b> .....	<b>22</b>
Példa közvetlen címzésű változók deklarálására .....	23
<b>A szervezési egység törzsrésze</b> .....	<b>24</b>
Az utasításlista.....	24
Az akkumulátor .....	24
Műveletek, parancsok.....	24
Módosító operátorok .....	25
<b>A műveletek csoportosítása</b> .....	<b>26</b>
Műveletek logikai (BOOL) változókkal .....	26
Műveletek általános (ANY) adattípussal .....	26
Ugró és hívóutasítások (programszervezési utasítások).....	26
<b>A függvények és a függvényblokkok használata</b> .....	<b>27</b>
A függvények hívása.....	27

Példa függvényhívásra .....	27
Példa műveletre .....	28
Példa standard függvény hívására .....	28
A függvényblokk hívása.....	28
Példa a felhasználói függvényblokk hívására .....	29
<b>Programtervezés funkciótervben.....</b>	<b>31</b>
<b>A standard függvények .....</b>	<b>32</b>
A standard függvényblokkok be- és kimeneti paramétereinek értelmezése és adattípusa...	35
RS tároló.....	35
SR tároló.....	36
Felfutó él detektálása: az R_TRIG függvényblokk.....	36
Lefutó él detektálása: az F_TRIG függvényblokk .....	37
<b>A számlálók.....</b>	<b>38</b>
CTD (Count Down) lefelé számláló.....	38
CTU (Count Up) felfelé számláló .....	38
CTUD (Count Up-Down) fel-le számláló .....	39
<b>Az időzítők .....</b>	<b>40</b>
Impulzus időzítő (TP = Time Pulse) .....	40
Bekapcsolás-késleltetési időzítő .....	41
Kikapcsolás-késleltetési időzítő.....	41
<b>A PLC konfigurálása .....</b>	<b>43</b>
<b>A PLC projekt felépítése.....</b>	<b>43</b>
A konfiguráció összetevői.....	44
A CONFIGURATION jellemzői .....	44
A RESOURCE jellemzői .....	44
A TASK és a futó program .....	45
Példa TASK deklarációra.....	46
<b>PÉLDATÁR.....</b>	<b>47</b>
Összerendelési táblázat: .....	48
Funkcióterv.....	50
Utasításlista .....	51
Létradiagram .....	52
<b>Követővezérlés tervezése döntési táblázattal .....</b>	<b>53</b>
Stanolás.....	53
Összerendelési táblázat .....	54
A döntési táblázat .....	54
A redukált függvénytáblázat .....	54
Létradiagram .....	55
A program utasításlistája.....	55
Gyakorló feladat Szivattyúk vezérlése .....	56
<b>Követővezérlés tárolással.....</b>	<b>57</b>
<b>Tárolótartályrendszer: feltöltés vezérlése.....</b>	<b>57</b>
Összerendelési táblázat .....	57
Funkcióterv.....	58
Kérdések:.....	59
Gyakorló feladat: Gyárkapu vezérlése .....	60
Összerendelési táblázat .....	60
Összerendelési táblázat .....	61
Megoldás .....	62
Utasításlista .....	62

Funkcióterv.....	62
Gyakorló feladat: utasításlista elemzése I. ....	63
<b>Követővezérlés impulzus időzítővel .....</b>	<b>64</b>
Kétkezes reteszelés.....	64
Összerendelési táblázat .....	64
A szűkített függvénytáblázat.....	64
Funkcióterv.....	65
A program utasításlistája.....	65
Vészjelzés.....	66
Összerendelési táblázat .....	66
Funkcióterv.....	67
Utasításlista .....	67
Gyakorló feladat: utasításlista elemzése II.....	68
<b>Követővezérlés időzítőkkal .....</b>	<b>69</b>
Szállítószalagok együttes vezérlése .....	69
Összerendelési táblázat .....	70
Funkcióterv.....	70
Utasításlista .....	72
Gyakorló feladat: Szállítószalag vezérlése.....	75
Összerendelési táblázat .....	75
Munkadarabok átmeneti tárolása .....	76
Összerendelési táblázat .....	76
Funkcióterv.....	77
Utasításlista .....	77
Tisztítóberendezés elektro-pneumatikus vezérlése .....	78
Összerendelési táblázat .....	78
Funkcióterv.....	79
Utasításlista .....	79
Gyakorló feladat: utasításlista elemzése III. ....	80

## Az IEC-1131-3 szabvány

Az IEC-1131-3 szabvány (International Electrotechnical Commission : [www.plcopen.org](http://www.plcopen.org)) a programozható logikai vezérlőberendezések (továbbiakban: PLC = Programmable Logic Controller) programozási nyelvére és a PLC-projektek felépítésére tartalmaz előírásokat.

A felhasználói program legkisebb, önállóan kezelhető szoftveregysége az ún. programszervezési egység, továbbiakban a POU (Program Organisation Unit).

A POU típusai: a függvény, a függvényblokk és a program, a sorrendnek megfelelően növekvő funkcionalitással. A függvény azonos bemenetekre mindig ugyanazt az eredményt, függvényértéket adja. A függvényblokknak ezzel szemben saját adatterülete (memóriája) van, melynek segítségével képes az előző állapotok információira „emlékezni” (ez az ún. instancképzés). A kimeneti értékeket így a bemeneteken kívül a tárolt adatok is befolyásolhatják, az előző állapotok függvényében más-más eredményt produkálva. A programok jelentik a felhasználói program legmagasabb hierarchia szinten lévő egységét, a programok biztosítják a többi POU-nak is a PLC-perifériákhoz való hozzáférés lehetőségét.

Megkülönböztethetünk standard, gyártó-specifikus és felhasználó által készített (felhasználói) programszervezési egységeket. Az IEC-1131-3 szabvány előírja a leggyakrabban előforduló standard függvények (pl.: aritmetikai, összehasonlító függvények) ill. standard függvényblokkok (pl.: időzítők, számlálók) hívási felületét és viselkedését.

### A programszervezési egységek felépítése

Minden POU két részből tevődik össze: a deklarációs részből és a programtörzsből.

### A változók deklarációja

Az IEC-1131-3 szabvány az információk tárolására és feldolgozására *változókat* használ. Vannak olyan PLC-rendszerek, amelyekben a változókat *merkek*nek (német nyeltesület) ill. *flagek*nek (angol) nevezik. A szabvány szerint a változók memóriaterületen történő elhelyezéséről már nem a programkészítőnek kell gondoskodnia, vagyis az ún. abszolút tárolási címet már nem kell manuálisan megadni. A fejlesztőrendszer feladata a változóhoz az adattípusának megfelelő méretű tárolóterület hozzárendelése. Előfordulhatnak azonban olyan esetek is, amikor szükségessé válhat a pontos memóriacím ismerete (pl. soros kommunikáció). A szabvány megengedi a felhasználónak a közvetlen memóriacím kijelölését, azzal az ajánlással, hogy ez csak a *program* típusú szervezési egység deklarációs részében történjen.

Az IEC-1131-3 szabvány több adattípust előre definiál (BOOL, BYTE, INTEGER stb.), amelyek a bitek számában, az előjelek kezelésében stb. különbözhetnek egymástól. Lehetőség van felhasználói adattípusokat is deklarálni (struktúrák, mezők).

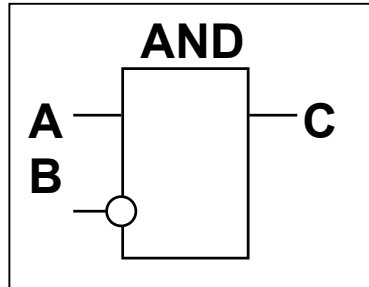
A változót hozzárendelhetjük elemmel védett fizikai címhez is, (remanens memória) így áramkimaradás esetén megőrzi értékét.

A változó érvényessége attól függ, hogy hol deklarálják. Így megkülönböztetnek globális és lokális változókat.

A POU deklarációs része szöveges formátumú és független az alkalmazott programozási nyelvtől. Egy részük grafikusán is megadható (be- és kimeneti paraméterek).



- a programelemek, mint blokkok összeköthetők, hasonlóan a logikai áramköri rajzokhoz;
- olyan alkalmazásokban használják, amelyek vezérlőkomponensek közötti adat vagy információáramlást tartalmaznak.



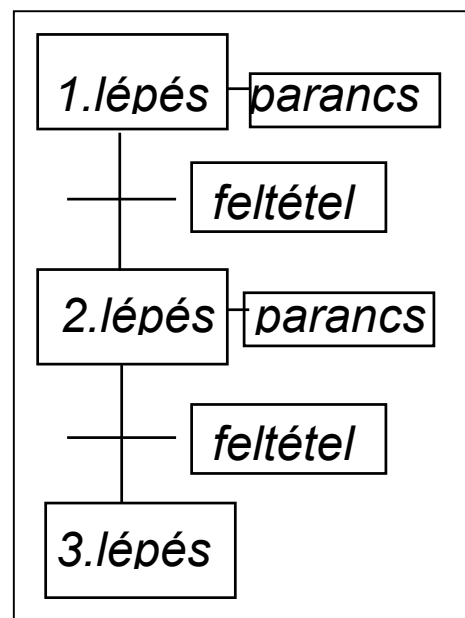
- *strukturált nyelv*  
jellemzője:

- PASCAL-ra emlékeztető, magas szintű, blokkszervezésű nyelv;
- megengedi az összetett utasításokat is;
- támogatja a ciklikus végrehajtást (REPEAT-UNTIL; WHILE-DO); támogatja a feltételes végrehajtást (IF-THEN-ELSE; CASE);
- a függvényeket (SQRT(), SIN()).

**C:= A AND NOT B**

- *lefutó nyelv: állapotgráf, léptetőlánc*  
jellemzője:

a vezérlési feladat sorosan és párhuzamosan végrehajtandó lépések sorozataként tervezhető. A léptetőlánc szemléletesen mutatja be a program lefutását, miközben megadja, hogy mely időpontban, milyen feltételek teljesülése esetén, milyen beavatkozás engedélyezhető a vezérelt folyamatban. Az IEC-1131-3 szabvány a vezérlő algoritmus strukturálásában hangsúlyozza a programtervezési technika jelentőségét.

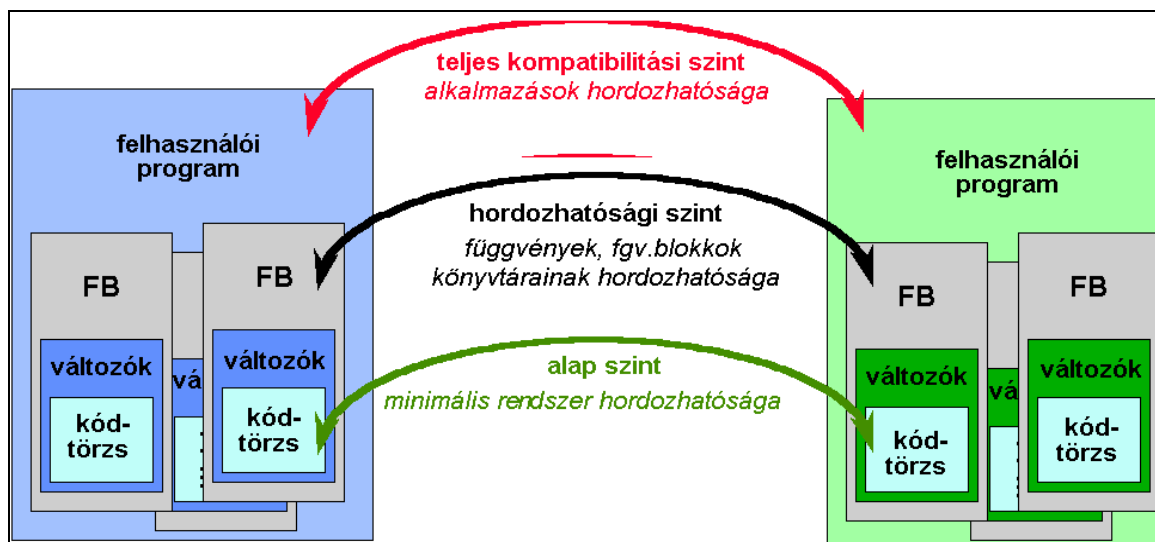


### Az IEC programozói környezet

A legtöbb fejlesztőrendszer biztosítja az alábbi feltételeket:



Az IEC-1131-3 szabvány támogatja a felhasználók törekvését a hordozhatóságra, azaz hogy amennyire lehetséges a függvények, függvényblokkok hardverfüggetlenek legyenek. A 2. ábrán láthatjuk a felhasználói programok lehetséges kompatibilitási szintjeit. Az, hogy egy fejlesztőrendszer melyik kompatibilitási szintet biztosítja, megmutatja azt is, hogy mennyiben felel meg a szabvány előírásainak.



2. ábra Kompatibilitási szintek

### A programszervezési egységekről részletesen

POU típus	kulcsszó	jelentés
program	PROGRAM	főprogram a PLC-perifériák hozzárendeléseivel, globális változókkal
függvényblokk	FUNCTION_BLOCK	építőelem be- és kimeneti változókkal, a leggyakrabban használt POU típus
függvény	FUNCTION	A PLC műveletek készletének kibővítésére szolgáló függvény

**Függvény (FGV):** paraméterezhető POU statikus változók nélkül (emlékezet nélkül), amely azonos bemeneti paraméterekre mindig azonos eredményt szolgáltat.

**Függvényblokk (FB):** paraméterezhető POU statikus változókkal, azonos bemeneti értékekre adott kimeneti értékek függenek a belső ill. globális változók memóriában tárolt értékeitől.

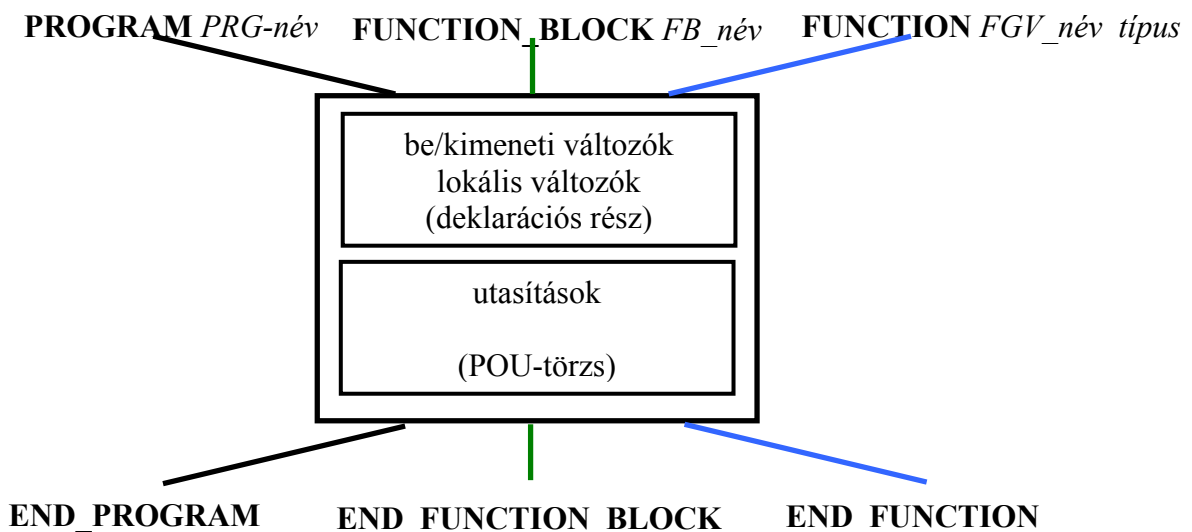
**Program (PRG):** főprogramként szolgál. Minden olyan változót itt kell deklarálni, amelyet fizikai címekhez akarunk rögzíteni (pl. a PLC be- és kimeneteihez). Egyébként olyan, mint a FB.

Mindegyik POU saját, lezárt tulajdonságokkal rendelkezik és a compiler a többi POU-tól függetlenül képes lefordítani. A fordítónak egyébként szüksége van minden információra azokról a programelemekről (prototípusok), amelyeket az adott POU hív. A lefordított POU-k később a LINK eljárással fűzhetők össze egységes programmá.

### A programszervezési egység részei

Egy POU az alábbi ábrán látható részekből épül(het) fel.

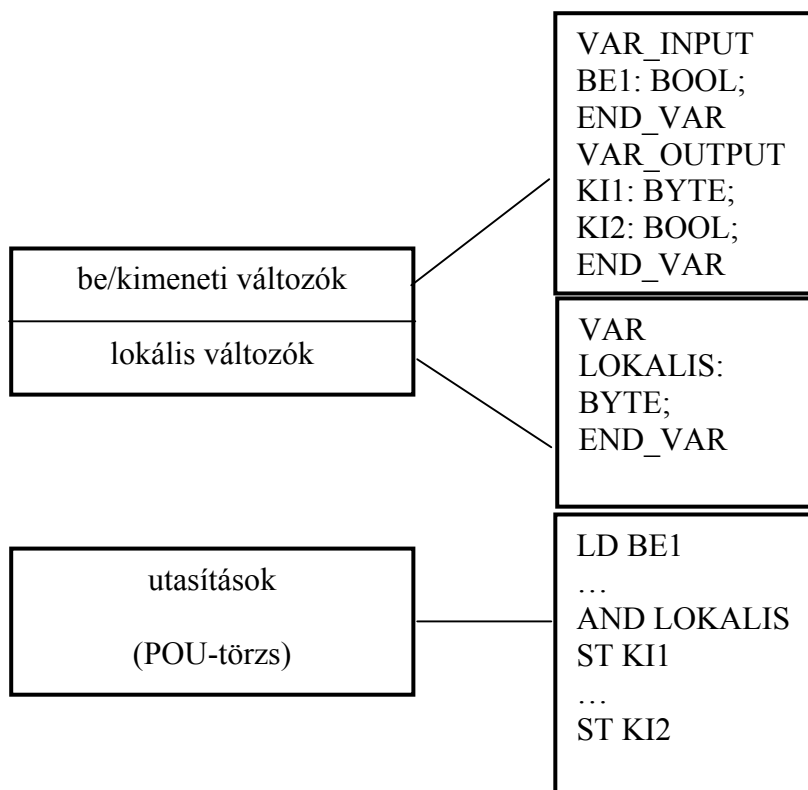
- A POU – típus megadása a *POU-név*-vel (és az adattípus is a FGV-eknél).
- Deklarációs rész a változódeklarálásokkal.
- POU-törzs az utasításokkal.



3. ábra A POU részei

### Példa a programszervezési egység felépítésére (függvényblokk)

FUNCTION\_BLOCK *FB\_név* ——— FUNCTION\_BLOCK *TOVABBKAPCS*



END\_FUNCTION\_BLOCK ——— END\_FUNCTION\_BLOCK

### Deklaráció

Az IEC-1131-3 szabvány a változókat a felhasználói adatok inicializálására, további feldolgozására és köztes tárolására használja. Ezeket a változókat minden POU elején deklarálják. A deklaráció megadja, hogy a változó milyen adattípusba tartozik, illetve milyen egyéb sajátosságokkal rendelkezik.

A deklaráció a változó típusoknak megfelelő blokkokra tagolódik. A deklarációblokk (VAR\_\* ... END\_VAR között) meghatározza a változó típusát, több változót is tartalmazhat.

A blokkok sorrendje, azonos változóra vonatkozó gyakorisága tetszőleges, illetve implementációfüggő, a szabvány nem rögzíti.

## Változótípusok

### A változótípusok engedélyezett használata

változótípus	engedélyezett a használat		
	PROGRAM	FUNCTION BLOCK	FUNCTION
VAR	igen	igen	igen
VAR_INPUT	igen	igen	igen
VAR_OUTPUT	igen	igen	nem
VAR_IN_OUT	igen	igen	nem
VAR_EXTERNAL	igen	igen	nem
VAR_GLOBAL	igen	nem	nem
VAR_ACCESS	igen	nem	nem

Látható, hogy a függvényeknél van a legnagyobb korlátozás, csak lokális és bemeneti változói lehetnek. A számítás eredményét a függvényértékben adják vissza, amely az AKKU-ban képződik.

Függvényblokkban nem lehet globális változót deklarálni, ez csak a programban (ill. az a fölötti hierarchiaszinteken lévő programozási elemekben) megengedett.

### A szervezési egységek kapcsolódási felületeinek jellegzetességei

Azzal, hogy a POU változóit változótípusokhoz rendeljük, meghatározzuk azok lehetséges kapcsolatát a többi POU-val, vagyis a csatlakoztatási változók és a lokális változók adatkörét is. A POU-kapcsolódási felülete lehet:

- hívási felület: formális paraméterek (be ill. Be/kimeneti paraméter)
- visszatérési érték: kimeneti érték vagy függvényérték
- globális csatlakozási felület: globális/externális változókkal.

A formális paraméterek helyébe a POU hívásakor az ún. aktuális paraméterek kerülnek.

### A formális paraméter és a visszatérési érték értelmezése

**Formális paraméter: (VAR\_INPUT):** az aktuális paraméter átadása értéként történik, azaz nem maga a változó, hanem csak a kópiája adódik át a hívott POU-nak. Így a feldolgozás a hívó POU-ban lévő változót nem módosítja.

**Formális paraméter: (VAR\_IN\_OUT):** az aktuális paraméter, mint mutató kerül átadásra. Így tulajdonképpen maga a változó kerül átadásra, értéke a POU-ban módosítható.

**Visszaadott érték (VAR\_OUTPUT):** a hívott POU nem adja át a változót, csak az értéke olvasható ki a POU futása után. A további feldolgozás (a hívó POU-ban) nem befolyásolja a (hívott POU-ban) tárolt változót.

Abban az esetben, ha nagymennyiségű adatot, vagy adatstruktúrát akarunk átadni a hívott programszervezési egységnek, célszerű a VAR\_IN\_OUT változótípus használata, mivel így nem történik többszörös tárterület foglalás.

A formális paramétereknek és a visszatérési értéknek az a különleges tulajdonsága tehát, hogy a hívó programban is láthatók és hivatkozhatunk rájuk anélkül, hogy deklaráltuk volna őket. A POU-k adatsere felületét ezért igyekezzünk jól dokumentálni. A be- és kimeneti változók védettek a nemkívánatos felülírástól.

**A változótípusok hozzáférési lehetőségeinek összefoglaló táblázata**

változótípus	hozzáférési jogosultság		értelmezés
	külső	belső	
VAR	-	I O	A lokális változó csak a POU-n belül látható, dolgozható fel.
VAR_INPUT	I	O	A bemeneti változó a hívó programban látható és írható, a POU-n belül csak olvasható.
VAR_OUTPUT	O	I O	A kimeneti változó a hívó programban látható és ott csak olvasható, A POU-n belül írható és olvasható is.
VAR_IN_OUT	I O	I O	A POU-n belül és kívül is írható – olvasható.
VAR_EXTERNAL	I O	I O	Az <i>external</i> típusú változót egy másik POU-ban mint <i>global</i> változót deklaráltak. Így minden POU-ban elérhető, és mint lokális változó módosítható. Az új értéket megőrzi a POU futása után is.
VAR_GLOBAL	I O	I O	A <i>global</i> változót a POU-n belül deklarálják és a külső POU-kban mint <i>external</i> változó deklarálható és használható. A POU-n belül úgy viselkedik, mint egy lokális változó.
VAR_ACCESS	I O	I O	Globális változó a konfigurációban. Az erőforrások közötti kommunikációs csatorna deklarálására szolgál. A POU-n belül mint globális változó kezelhető.

I = írható O = olvasható

**Példa a FB formális paramétereinek belső és külső értelmezésére**

FUNCTION\_BLOCK Fbketto

```

VAR_INPUT
    bemenet : BYTE;
END_VAR
VAR_OUTPUT
    kimenet : BYTE;
END_VAR
VAR
    lokalis: BYTE;
END_VAR

```

```

...
LD    bemenet
AND  lokalis
ST    kimenet
...

```

END\_FUNCTION\_BLOCK

FUNCTION\_BLOCK FBegy

```

VAR
    peldaFB: Fbketto;
END_VAR
...
LD    48
ST    peldaFB.bemenet
CAL  peldaFB
LD    peldaFB.kimenet

```

```

...
END_FUNCTION_BLOCK

```

## A függvényblokk

Az IEC-1131-3 szabvány legfontosabb szoftvereleme. A strukturált programírás hatékony eszköze. Programból vagy függvényblokkból hívható és függvényt vagy függvényblokkot hívhat. A függvényblokk fogalmát tulajdonképpen kétféle értelemben használják. Az egyik értelmezés a függvényblokkot, mint típust jelenti, ezt kapjuk a FB megírásával. A másik megjelenési formája a deklaráció segítségével egyediesített (*instance*) függvényblokk. Az egyediesítés során a függvényblokk-típusban meghatározott méretű adatterületet a fordító lefoglalja az egyedi FB számára, így annak saját, önálló adatterülete lesz. Az a FB\_név tehát, amelyet a FB írás során a FB-nak adunk, típusazonosítóként szolgál a deklarációs részben, a FB hívása az egyedi névvel történik. (lásd a fenti példában Fbketto - peldaFB )

A függvényblokkot abban a POU-ban, amelyben hívni akarjuk, deklarálnunk kell, mégpedig annyiszor, ahány egymástól különböző felhasználást akarunk. Ezáltal biztosíthatjuk a megfelelő, egymástól elkülönült és védett adatterület lefoglalását, amely adatterület az egyediesített FB „emlékezeteként” működik. Itt tárolja a rendszer a FB be- és kimeneti ill. lokális változóit. Ez vonatkozik a standard és a felhasználói függvényblokkokra is. (Mivel ez statikus tárfoglalást jelent, nagy adatblokkokkal dolgozó függvényblokk igen sok helyet foglalhat le. Tervezik ezért a VAR\_DYN ... END\_VAR típusú deklarációt.)

## Hordozhatóság és objektum orientáltság

Az alábbi korlátozásokat a hordozhatóság, a platformfüggetlenség biztosítása miatt rögzítették:

- közvetlen fizikai címet lokális változóhoz nem rendelhetünk,
- adatokhoz kizárólag a csatlakozási felületként deklarált változókon keresztül juthat,
- a függvényblokkban globális változók nem deklarálhatók.

## A függvényblokkban használható változótípusok

A függvényblokknak tetszőleges számú, vagy semennyi be/kimeneti paramétere lehet, ill. lokális és externális változókat is felhasználhat.

A VAR\_INPUT, VAR\_OUTPUT, VAR\_IN\_OUT és VAR\_EXTERNAL típusú változókat a hívó program is látja, hivatkozni rájuk az *egyedi\_FB\_név.változónév* formátummal lehet. A bemeneteknek a FB hívása előtt adhatunk értéket, a kimeneteket a FB hívása után kérdezhetjük le

## A függvény

Rendszeresen ismétlődő feladatokhoz célszerű függvényeket alkalmazni. A függvény több hívási paramétert tartalmazhat, a végrehajtás eredménye pedig egyetlen kimeneti változóban helyezkedik el, mely lehet egyetlen adat, de lehet akár többelemű, tömb típusú is.

A függvény azonos bemeneti paraméterekre mindig azonos eredményt szolgáltat, függetlenül attól, hogy hányszor, ill. mely időpillanatban történt a hívása. Nagyszámú, gyakran használt függvényt standardizáltak, azaz tulajdonságait, számítási algoritmusát, hívási paraméterlistáját a szabványban rögzítették. Ezt a gyűjteményt egészíthetjük ki egy adott projektben a gyártó-specifikus és a felhasználó által készített függvények

### A függvény változótípusai és a függvényérték

A függvénynek egy vagy több (tetszőleges számú) bemeneti paramétere lehet, de csak egy értéket adhat vissza, ez a függvényérték. A függvényérték tetszőleges adattípusú lehet, akár származtatott adattípus is. A lokális változót nem lehet RETAIN-nel pufferelni.

A függvények érvényességi területe globális, azaz minden POU részére rendelkezésre áll, nem kell külön a hívó POU-ban deklarálni.

A függvény hívása a függvény nevének megadásával és a bemeneti adatok teljes paraméterátadásával történik.

A paraméterátadás során az elsőként deklarált bemeneti változót beírjuk az AKKU-ba, a többi változót a függvény hívási sorában, a függvény neve után, egymás között vesszővel elválasztva, felsoroljuk.

### A program

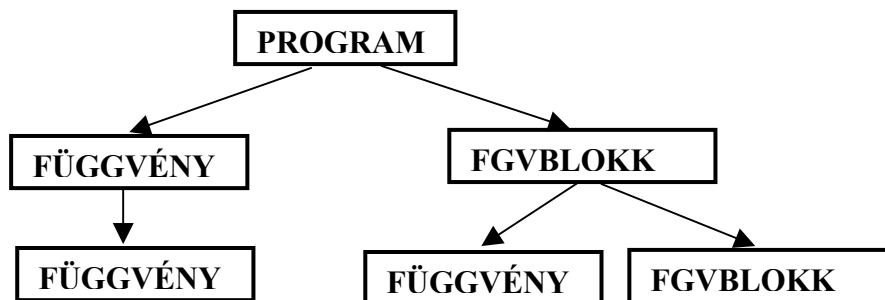
A függvényblokk és a függvény alprogramok, a PROGRAM főprogram. Multi-taszkos operációs rendszerben egymással párhuzamosan több főprogram is futtatható.

A program sajátosságai:

- a FB-hoz képest lehetővé teszi:
  - közvetlen (direkt) címzésű változók használatát,
  - globális változó deklarálását.
- a programot a PLC-konfiguráción belül taszkhhoz rendeljük, a programot explicit más POU nem hívhatja.

Kis rendszereknél a program feladata az is, hogy külön konfigurációs fájl nélkül biztosítsa a PLC-perifériák változókhöz rendelését. A lehetőségek operációs rendszertől és kiépítettségtől (gyártótól) függőek. Azonos programot több taszkhhoz is hozzárendelhetünk, ezt az ún. konfigurációs rendszerben definiálhatjuk.

A szervezési egységek lehetséges hívási kapcsolatát mutatja be a következő ábra:



4. ábra A függvény és a függvényblokk hívási lehetőségei

A rekurzív hívás nem megengedett!

## **Nyelvi elemek, adattípusok, változók**

### **Egyszerű nyelvi elemek**

Minden PLC programozási nyelv tulajdonképpen alapvető, tovább nem bontható nyelvi elemek sokaságából épül fel. Ezen nyelvi elemekből áll össze a változódeklaráció, az utasítássorok, végezetül az egész program. A nyelvi elemek lehetnek:

- különleges jelentéssel bíró karakterek: (, ), +, -, \*, \$, :, =, #, space
- kulcsszavak: a programnyelv „szavai”
- különböző adattípusok számábrázolására szolgáló karakterkombinációk
- a felhasználó által definiált nevek, címkék.

### **Foglalt kulcsszavak**

A kulcsszavak a szabvány által leírt és egyértelmű jelentéssel bíró standard nevek, amelyek nem használhatók a felhasználó által definiált változók neveiként vagy címkéiként. Ilyenek:

- elemi adattípusok nevei,
- standard függvények nevei,
- standard függvényblokkok nevei,
- standard függvények bemeneti paramétereinek a nevei,
- standard függvényblokkok be/kimeneti paramétereinek a nevei,
- az utasítások, parancsok nevei.

### **A különböző adattípusok számábrázolása**

A számábrázoláshoz előírt helyesírási konvenció tartozik. A konstanson belül szóközök alkalmazása helyett megengedett térelválasztónak az aláhúzás jel. (A szóközök csak a STRING változóknál használhatók!)

### A konstansok áttekintése

adattípus	példa	jelentés
bináris	0, 1	1 bit
bool	FALSE, TRUE	bool-algebrai kifejezés
byte	11, 16#0B, 2#0000_1011	11 decimális, hexadecimális és kettes számrendszerben
egész szám	-13 45165 vagy 45_165 +125	egész szám: -13 egész szám: 45 165 egész szám: 125
valós	13.12 123.45 0.123 -1.23E-3	valós szám: 13,12 valós szám: 123,45 valós szám: 0,123 valós szám:0,00123
karaktorsor	'' 'SZTRING'	üres sztring sztring
időtartam	T#12.3ms vagy TIME#12.3ms t#2h 7m 19s	12,3ms időtartam  2 óra 7 perc 19 másodperc időtartam
dátum	DATE#2001-12-31 vagy D#1995-12-31	dátum: 2001 12. 31.
napi idő	TOD#12:16:14.56 vagy TIME_of_DAY#12:16:14.16	időpont: 12óra 16perc, 14,56másodp
dátum és időpont	DT#2001-12-31-12:16:14.56 v. DATE_AND_TIME#2001-12-31- 12:16:14.56	dátum és idő együtt: 2001 12. 31 12óra 16perc, 14,56másodperc

### A felhasználó által definiálható nevek, címkék

Karakterrel vagy aláhúzás jellel kezdődő alfanumerikus karaktorsorozat, maximális hossza implementációfüggő. Különböző programelemek, változók, címkék, származtatott adattípusok, konfigurációk, erőforrások azonosítására szolgáló felhasználó által adott nevek.

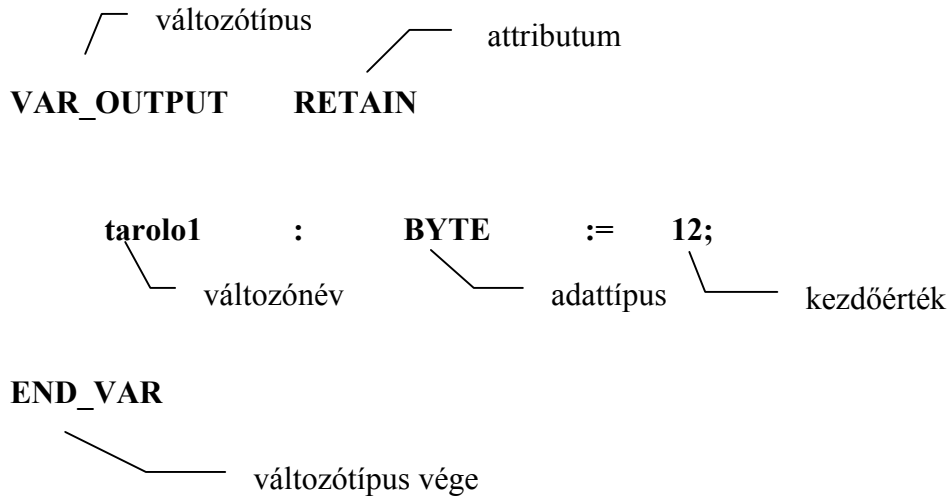
#### Példák felhasználói nevekre:

megengedett nevek	nem megengedett nevek
VALT2	2VALT
SZELEP3X7	3X7
VESZ_KI	VÉSZ KI
_3kevero	3keverő

## Változók és adattípusok

A változók segítségével történik a felhasználó-specifikus adatterületek adattípus által meghatározott méretű helyfoglalása és azonosítása.

### A változódeklaráció legfontosabb elemei



A változónév: betűvel vagy aláhúzás jellel kezdődő, kis- és nagybetűk, számok, aláhúzások sorozatából áll, max. 64 karakter hosszú. Nem tartalmazhat: szóközt, ékezetes betűket és kulcsszavakat. A kis- és nagybetűk között nincs megkülönböztetés.

A kezdeti értékadás := operátorral lehetséges.

A deklarációs sor végét ; jelzi. Megjegyzéseket, kommentárokat (\* \*) zárójelek között írhatunk.

## Adattípusok

### Elemi adattípusok

bináris/ bitminta	előjeles egésszám	előjel nélküli egész	valós	időpont, időtartam, dátum, karakter sor
BOOL	INT	UINT	REAL	TIME
BYTE	SINT	USINT	LREAL	DATE
WORD	DINT	UDINT		TIME_OF_DAY
DWORD	LINT	ULINT		DATE_OF_TIME
LWORD				STRING

Az elemi adattípusokat az adott kulcsszavak egyértelműen meghatározzák. A kezdeti értékek a := hozzárendelési operátorral adhatók meg. Amennyiben nincs kezdeti (inicializálási) értékadás, a változók a default értékeket veszik fel.

Az alábbi táblázatban összefoglaltuk a standard elemi adattípusok jellemzőit.

adattípus	értelmezés	hossz (bit)	értékkészlet	default érték
BOOL	kétértékű bináris szám	1	[0,1]	0
BYTE	bitsorozat 8bit	8	[0,...,16#FF]	0
WORD	bitsorozat 16bit	16	[0,...,16#FFFF]	0
DWORD	bitsorozat 32bit	32	[0,...,16#FFFF FFFF]	0
LWORD	bitsorozat 64bit	64	[0,...,16# FFFF FFFF FFFF FFFF]	
SINT	short integer	8	[-128,...,+127]	0
INT	integer	16	[-32 768,...,+32 767]	0
DINT	double integer	32	[-2 147 483 648,...,+2 147 483 647]	0
LINT	long integer	64	$[-2^{63}, \dots, +2^{63}-1]$	
USINT	unsign. short integer	8	[0,...,+255]	0
UINT	unsigned integer	16	[0,...,+65 535]	0
UDINT	unsign. double int.	32	$[0, \dots, +2^{32}-1]$	0
ULINT	unsign. long integer	64	$[0, \dots, +2^{64}-1]$	0
REAL	real; valós szám,	32	+/-3,4 E+/-38	0
LREAL	long real	64		0
TIME	időtartam			T#0s
DATE	dátum formátum: YYYY-MM-TT			D#0001-01-01
TIME_OF_DAY	időpont formátum: HH:MM:SS			TOD#00:00:00
DATE_AND_TIME	dátum és idő			DT 0001-01-01-00:00:00
STRING	változó hosszúságú karakterlánc			'' (üres)

### Származtatott adattípusok

A származtatott adattípusokat az elemi adattípusokból lehet új, a felhasználó által megadott kulcsszóval előállítani. Típusdeklarációnak is nevezik. Az ilyen típusdefiníciók a PLC-projektben globálisan felhasználhatók, a programozónak lehetősége van a feladatmegvalósításhoz illeszkedő adatstruktúra kialakítására. A típusdefiníálást a TYPE... END\_TYPE kulcsszavak határolják.

Ide sorolhatók:

- az egyedi felhasználónévvel ellátott, esetenként korlátozott értéktartományú változók;
- az azonos adattípusú elemi változóból álló, ARRAY kulcsszóval definiált tömbök;
- az adatstruktúrák: a magas szintű programnyelvekhez hasonlóan, a STRUCT .... END\_STRUCT kulcsszavak között deklarált hierarchikus változók.

## Általános adattípusok

Az elemi adattípusok hierarchikus csoportba foglalására az IEC-1131-3 szabvány ún. általános adattípusokat definiál. Ezek az adattípusok az ANY rövidítéssel kezdődnek, pl.: az összes egészszám adattípus (integer: INT) összefoglaló neve az ANY\_INT lesz. A legáltalánosabb, bármely elemi adattípust elfogadó az ANY paraméter.

Deklarációban az ANY-vel kezdődő adattípus nem használható!

### Az általános adattípus

ANY					
ANY_BIT	ANY_NUM			ANY_DATE	TIME STRING
	ANY_INT		ANY_REAL		
BOOL	INT	UINT	REAL	DATE	
BYTE	SINT	USINT	LREAL	TIME_OF_DAY	
WORD	DINT	UDINT		DATE_OF_TIME	
DWORD	LINT	ULINT			
LWORD					

A standard függvények és függvényblokkok be/kimeneti paramétertípusainál találkozhatunk az összefoglaló nevekkel, és azt jelzi, hogy az adott függvény(blokk) többféle elemi adattípussal is meghívható. Ez az ún. függvényátlapolási technika.

Az ANY-vel kezdődő adattípus felhasználói függvényben ill. függvényblokkban nem megengedett, illetve a szabvány nem rögzíti.

## A változóattribútumok

- RETAIN : elemmel pufferelt adatterületen tárolt változók. Melegindítás esetén megőrzik előző értéküket.
- CONSTANT : állandó értékű változó.
- R\_EDGE, F\_EDGE felfutó- ill. lefutó-élhez rendelt változó.
- READ\_ONLY, READ\_WRITE. írásvédett ill. írható/olvasható változó.

### A változótípusokhoz rendelhető attribútumok összefoglaló táblázata

változótípus	RETAIN	CONSTANT	R_EDGE, F_EDGE	READ_ONLY, READ_WRITE
VAR	+	+	-	-
VAR_INPUT	-	-	-	-
VAR_OUTPUT	+	-	-	-
VAR_IN_OUT	-	-	-	-
VAR_EXTERNAL	-	-	-	-
VAR_GLOBAL	+	+	-	-
VAR_ACCESS	-	-	-	+

A READ\_WRITE attribútum csak a VAR\_ACCESS típusú változó jelölésére engedélyezett.

### Példa az attribútumok használatára

```

VAR_OUTPUT RETAIN
    puffer1 : BYTE;
END_VAR
VAR_INPUT
    LEFUTO : BOOL F_EDGE;
END_VAR
VAR_CONSTANT
    allando1 : BYTE:= 16#FF;
END_VAR
    
```

### Közvetlen címzésű változók

A fizikai címek közvetlenül is megszólíthatók a programban. (Bemenetek, kimenetek, belső változó, merkek.) Ez történhet:

- közvetlen (direkt) ábrázolású változóval
- szimbolikus nevű, közvetlen (direkt) címzésű változóval.

Az ilyen változók deklarálása az **AT** kulcsszóval és a fizikai cím megadásával történik. A címek felépítése az alábbi táblázat szerinti.

A közvetlen címeket hierarchikus címeknek is szokták nevezni, % jellel kezdődnek, amelyet egy betű követ: **I** (bemenet), **Q** (kimenet) vagy **M** (változó, merker). Az ezt követő betű a cím hosszára ad információt. Az **X** bitcím elhagyható.

közvetlen PLC-címek				magyarázat
%				kezdőjel
	I Q M			bemenet kimenet merker
		SEMMI X B W D L		bit bit (opcionális) bájt szó duplaszó hosszú szó
			v,w,x,y,z	hierarchikus cím, jobbról balra nő a hierarchiaérték. A hossza és interpretálása gyártófüggő. Pl.: z-bit, y-bájt, x-modul, w-vonal, v-erőforrás
például:				
%	I	W	0.0.1.2	1. modul, 2. bájt
%	Q	D	0.0.3.0	3-ik modul, 0. bájt
%	M		0.0.5.2.0	5. modul, 2. bájt, 0. bit
%	M	X	0.0.5.2.0	5. modul, 2. bájt, 0. bit
%	I		0.0.1.0.4	1. modul, 0. bájt, 4. bit
%	Q	B	0.0.0.1.4	0.erőforrás,0.vonal,0-ik modul, 1. bájt, 4. bit

A bitcím 0..7 között változhat. A bájt cím gépfüggő (Összesen mennyi be/kimenet ill. merker definiálható.) Gyakran előírás, hogy a szó csak páros bájt címen kezdődhet. (Ne felejtsük el, hogy közvetlen címzésű változókat csak a főprogramban lehet deklarálni!)

### **Példa közvetlen címzésű változók deklarálására**

VAR

(\*közvetlen ábrázolású változók\*)

AT%IW6 : WORD;

AT%QD4 : DINT;

(\*szimbolikus nevű, közvetlen címzésű változók\*)

INP\_BYTE AT%IB0;

OUT\_WORD AT%QW0;

END\_VAR

...

LD INP\_BYTE

BYTE\_TO\_WORD

ST OUT\_WORD

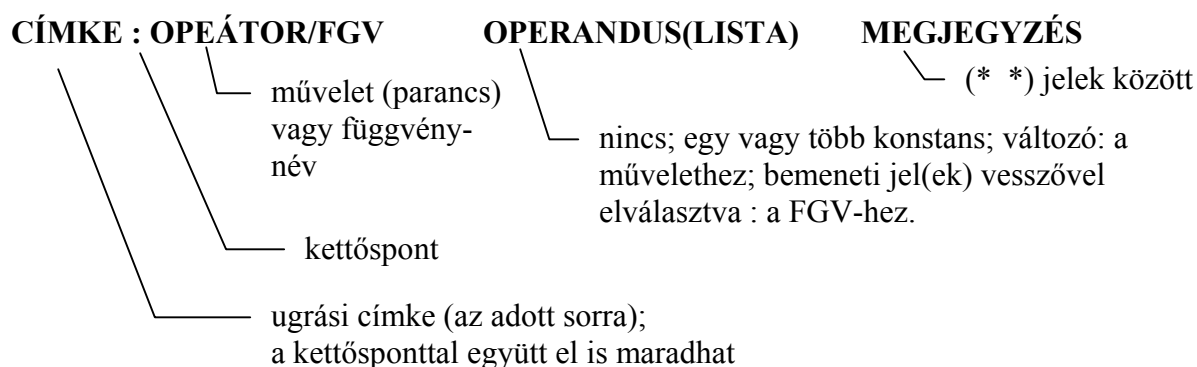
....

## A szervezési egység törzsrésze

### Az utasításlista

Sororientált nyelv: egy parancs egy sor.

A parancssor elemei:



Soronként egy megjegyzés megengedett. A pontosvessző (;) az utasításlistában nem használható sem határoló karakterként, sem kommentárkezdésként. A művelet (parancs) és az operandus között legalább egy szóközt kell hagyni. Nem kötelező a tabulátor használata.

### Az akkumulátor

Az assembly nyelvek gyakran indulnak ki egy fizikailag is meglévő processzor-akkumulátorból. Az utasításlistás nyelv szintén ismeri ezt az akkumulátort, amelynek CURRENT RESULT (CR), aktuális eredmény a neve, de nem úgy kezeli, mint egy fix hosszúságú regisztert. A fordító (compiler) gondoskodik arról, hogy rendelkezésre álljon a feldolgozandó adattípusnak megfelelő hosszúságú adatterület (akku-tároló). Más assembly nyelvektől eltérően, nincs külön speciális státuszbit. Az összehasonlítások eredménye (igaz/hamis, 0/1), a CR –ben képződik. A feltételes ugrás vagy hívás a CR értékétől függ.

Szintaktikai hibát okoz, ha különböző adattípusok között akarunk műveletet végrehajtani, vagyis, ha a CR adattípusa más, mint az operandus adattípusa.

Egy művelet a CR értékét :

- beállítja (B),
- módosítja (M),
- változatlanul hagyja (V),
- nem definiálja (U).

A következő fejezet táblázata mutatja az elemi műveletek fenti műveleti csoportba sorolását is.

### Műveletek, parancsok

Az alábbiakban összefoglaljuk az utasításlista műveleteit. Ezek közül néhányat ún. módosító operátorokkal is kibővíthetünk.

## Módosító operátorok

A módosító operátorok új jelentést adnak a műveleteknek.

- negálás  $\bar{N}$   
a parancs végrehajtása előtt az operandust negálja.
- zárójel  $( )$   
segítségével a CR értékét egy utasítássorozat eredményével hozhatjuk kapcsolatba. A zárójelek egymásba ágyazhatók.

Pl.:

```
LD    1
ADD(  2
ADD(  3
ADD   4
)
)
ST    valt1
```

- a művelet feltételes végrehajtása  $C$   
vannak olyan műveletek, amelyek eredménye logikai érték. Ha ez igaz, az utasítást végrehajtja, ha nem a program a műveletet „átugorja”, és a következő sorral folytatja a futását.

Pl.:

```
LD    valt1 }
GT    20   } CR=1, ha valt1>20, egyébként CR=0.
JMPC  B2
JMP   TOVABB
```

B2 : ....

...

TOVABB : ..

...

## A műveletek csoportosítása

### Műveletek logikai (BOOL) változókkal

művelet	műveletcsoport	értelmezés
LD LDN	B	betöltés a CR-be
AND ANDN AND( ANDN(	M	„és” kapcsolat a CR és az operandus között
OR ORN OR( ORN(	M	„vagy” kapcsolat a CR és a művelet operandusa között
XOR XORN XOR( XORN(	M	„kizáró-vagy” kapcsolat a CR és a művelet operandusa között
ST STN	V	CR értékének /negáltjának tárolása az operandusban
S	V	operandus beállítása igaz (1) értékre, ha CR=igaz
R	V	operandus beállítása hamis (0) értékre, ha CR=igaz
)	V	a zárójeles művelet vége

Megjegyzés: a legtöbb fejlesztői rendszer kibővíti a fenti műveleteket azonos névvel, de standard függvényként ANY\_BIT adattípusra. Ezzel biztosítják, hogy azonos műveleti névvel, szóhosszúságú adatokra is alkalmazható a parancs. A felhasználónak nem kell különbséget tennie, hogy alapl műveletet, vagy standard függvényt hív.

### Műveletek általános (ANY) adattípussal

művelet	műveletcsoport	értelmezés
LD	B	AZ OPERANDUS CR-be töltése
ST	U	CR értékének tárolása az operandusban
ADD ADD(	M	az operandus értékét hozzáadja a CR-hez
SUB SUB(	M	az operandus értékét levonja a CR-ből
MUL MUL(	M	az operandus értékével szorozza a CR-t
DIV DIV(	M	az operandus értékével osztja a CR-t
GT GT(	M	CR > operandus? igen:CR=1, nem: CR=0.
GE GE(	M	CR >= operandus? igen:CR=1, nem: CR=0.
EQ EQ(	M	CR = operandus? igen:CR=1, nem: CR=0.
NE NE(	M	CR ≠ operandus? igen:CR=1, nem: CR=0.
LE LE(	M	CR <= operandus? igen:CR=1, nem: CR=0.
LT LT(	M	CR < operandus? igen:CR=1, nem: CR=0.
)	U	a zárójeles művelet vége

### Ugró és hívóutasítások (programszervezési utasítások)

művelet	műveletcsoport	értelmezés
JMP JMPC JMPCN	V vagy U U	feltétel nélküli ugrás CR-függő feltételes ugrás } címkére
CAL CALC CALCN	V U	feltétel nélküli hívás CR-függő feltételes hívás } FB
RET RETC RETCN	V vagy U U	feltétel nélküli visszatérés CR-függő feltételes visszatérés } FGV-ből, FB-ból
FGV_név	M	függvényhívás

A fenti táblázatban lévő műveletek operandusai címkék ill. egyedi FB\_nevek.

## A függvények és a függvényblokkok használata

### A függvények hívása

A függvények hívása utasításlistában a függvénynév megadásával történik. Az aktuális paramétereket vesszővel elválasztva fűzzük hozzá. A paraméterátadás úgy történik, hogy az elsőként deklarált bemeneti változót beírjuk az AKKU-ba, a többi változót a függvény hívási sorában, a függvény neve után, vesszővel elválasztva soroljuk fel. A függvények érvényességi területe globális, nem kell külön deklarálni.

A függvénynek pontosan egy kimeneti paramétere van, amely a CR-be kerül. Ez úgy lehetséges, hogy a függvénytörzsben van egy olyan tárolási utasítás, amely a függvénynévvel azonos nevű változónak ad értéket. Ezt a változót a fordító automatikusan generálja, a deklarációs részben nem kell a felhasználónak külön definiálnia.

### Példa függvényhívásra

A függvény deklarációja:

```
FUNCTION felhasznaloi : INT
VAR_INPUT
    fgvpar1, fgvpar2, fgvpar3: INT;
END_VAR

LD    fgvpar1
ADD   fgvpar2
ADD   fgvpar3
ST    felhasznaloi (*visszatérési érték*)
END_FUNCTION
```

A függvény hívása:

```
...
VAR
    par1: INT :=10;
    par2: INT :=20;
    par3: INT :=30;
    eredm: INT;
END_VAR

LD    par1
felhasznaloi    par2, par3
(*második hívás*)
felhasznaloi    par2, par3
ST    eredm
...
```

A másodszeri hívás után az **eredm** változóban tárolt érték: 110.

Gyakran nem is vesszük észre, hogy nem műveletet, hanem egy standard függvény hívását tartalmazza az utasítássor. Ennek felismerése a fordító feladata.

### Példa műveletre

```
Var
    valt1: BOOL;
END_VAR

LD TRUE
AND valt1
```

### Példa standard függvény hívására

```
Var
    valt1: WORD;
END_VAR

LD 16#77F
AND valt1
```

### A függvényblokk hívása

A függvényblokk a CAL vagy a CALC/CALCN paranccsal hívható. Az IEC-1131-3 szabvány a FB-hívás háromféle szintaktikáját engedi meg:

- hívás a bemeneti paraméterek zárójelbe zárt listájával;
- hívás előtt a bemeneti paramétereknek a megfelelő címre tárolásával;
- implicit hívás a bemeneti paraméterek, mint operátorok felhasználásával.

A harmadik módszer csak a standard függvényblokkoknál alkalmazható. (Ilyenkor a rendszer képes a standard függvényblokkok bemeneteit mint műveleteket (parancsokat) értelmezni. Erre csak kevés fejlesztői rendszer van felkészítve.)

Az alábbi példában egy standard függvényblokk, a bekapcsolás-késleltetési időzítő szabvány szerinti három lehetséges hívását mutatjuk be.

Az időzítő deklarálása:

```
VAR
    indit, ki : BOOL :=0; (*indit: futásengedélyező – input, ki: kimenet*)
    idozito1: TON; (*standard FB TON deklarálás egyedi néven*)
    ertek: TIME; (*idő adattípusú változó*)
END_VAR
```

A függvényblokk hívása:

1. módszer	2. módszer	3. módszer
(*paraméterátadás*)	LD t#500ms ST idozito1.PT LD indit ST idozito1.IN	LD t#500ms PT idozito1 LD indit
CAL idozito1(IN:=indit, PT:= t#500ms)	CAL idozito1	IN idozito1

A kimeneti paraméterek kiértékelése mindhárom módszernél azonos:

```
LD idozito1.Q
ST ki
LD idozito1.ET
ST ertek
```

A deklarációs rész és a kimenetek kiolvasása mindhárom módszernél azonos. Különbség a bemeneti paraméterátadásban és a FB-hívásban van.

### Példa a felhasználói függvényblokk hívására

Lássunk egy példát felhasználói függvényblokk hívására is. A függvényblokknak csak a deklarációs részét adjuk meg, a FB-törzsnek a példa szempontjából nincs jelentősége.

A függvényblokk:

```
FUNCTION_BLOCK Fblokk
  VAR_INPUT
    par1: TIME;
    par2: WORD;
    par3: INT;
  END_VAR
  .....(*utasítások sorozata*)
END_FUNCTION_BLOCK

PROGRAM progr1
  VAR_GLOBAL
    fgvblk1: Fblokk;
    globvalt : INT;
  END_VAR
  VAR
    BE: WORD AT %IW4;
  END_VAR
  .....

END_PROGRAM
```

Hívások:

1. módszer:

CAL fgvblk1(par1:= t#20ms, par2:=BE, par3:=globvalt)

vagy:

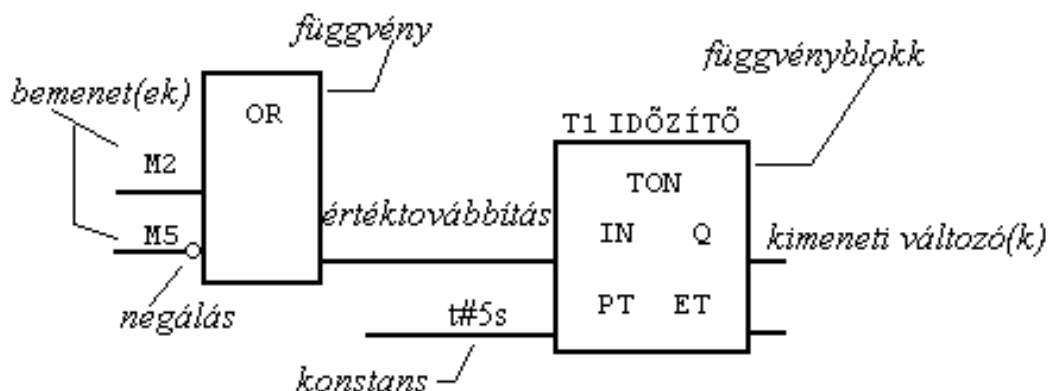
CAL fgvblk1(par1:= t#20ms, par2:=BE)

(A hiányzó formális paraméter aktuális értéke az első híváskor a kezdeti érték vagy a default érték, későbbiekben az utolsó hívás értéke.)

2. módszer:

```
LD    t#20ms
ST    fgvblk1.par1
LD    BE
ST    fgvblk1.par2
LD    globvalt
ST    fgvblk1.par3
CAL   fgvblk1
```

## Programtervezés funkciótervben



5. ábra Példa a funkcióterv elemeire

A funkciótervben a vezérlőalgoritmust grafikus objektumok kapcsolatrendszereként építjük fel. Az objektumok lehetnek:

- összekapcsolások;
- ugrások grafikus jelei;
- függvények és függvényblokkok hívása;
- csatlakoztatások.

A funkcióterv logikus, jól áttekinthető programtervet eredményez, melyben a hibafelismerés, program-módosítás lényegesen egyszerűbb, mint az utasításlistában. A funkciótervben az információ áramlás irányát balról jobbra és felülről lefelé, sorosan, lépésenként kell elképzelnünk, azaz a funkcióterv nem analóg egy áramköri tervvel. A korszerű fejlesztőrendszerek lehetővé teszik a vezérlőalgoritmus funkciótervvel történő leírását, de ezt fordításkor mindig átkonvertálják utasításlistába. Nem felejtkezhetünk el tehát arról, hogy bár korszerűbb programtervezési módszerrel dolgozunk, a gépen futó programunk időben sorban egymás után következő információ-feldolgozó gépi parancsok sorozata, amely nem felel meg egy digitális áramkör párhuzamosan futó áramjeleinek.

## A standard függvények

Az IEC-1131-3 szabvány a standard függvényeket az alábbi nyolc csoportba foglalja:

1. Típusváltásra szolgáló függvények (adattípus konvertálása).
2. Numerikus függvények.
3. Aritmetikai függvények.
4. Bitsorozat függvények (léptető és bitsoros logikai függvények).
5. Összehasonlító és kiválasztó függvények.
6. Karakterorozat feldolgozó függvények (sztring-műveletek).
7. TIME adattípus speciális függvényei.
8. Számlálóval kapcsolatos speciális függvények.

Az alábbi táblázat a fenti felosztásnak megfelelően csoportosított standard függvények jellemzőit tartalmazza.

standard függvény (a bemeneti paraméterekkel)	függvény- érték adattípusa	jelentés
<i>Típuskonvertáló függvények</i>		
*_TO_** (ANY)	ANY	típuskonverziók elemi adattípusok között
TRUNC (ANY_REAL)	ANY_INT	REAL szám egészét adja
<i>Numerikus függvények</i>		
ABS (ANY_NUM)	ANY_NUM	abszolút érték képzés
SQRT (ANY_REAL)	ANY_REAL	négyzetgyök
LN (ANY_REAL)	ANY_REAL	természetes alapú logaritmus
LOG (ANY_REAL)	ANY_REAL	10-es alapú logaritmus
EXP (ANY_REAL)	ANY_REAL	exponens
SIN (ANY_REAL)	ANY_REAL	szinusz fgv.
COS (ANY_REAL)	ANY_REAL	koszinusz fgv.
TAN (ANY_REAL)	ANY_REAL	tangens fgv.
ASIN (ANY_REAL)	ANY_REAL	arcszinusz fgv
ACOS (ANY_REAL)	ANY_REAL	arccoszinusz fgv.
ATAN (ANY_REAL)	ANY_REAL	arctangens fgv.
<i>Aritmetikai függvények (IN1, IN2)</i>		
ADD (ANY_NUM, ANY_NUM)	ANY_NUM	összeadás
ADD (TIME, TIME)	TIME	időösszegezés
ADD (TOD, TIME)	TOD	időösszegezés
ADD (DT, TIME)	DT	időösszegezés
MUL (ANY_NUM, ANY_NUM)	ANY_NUM	szorzás
MUL (TIME, ANY_NUM)	TIME	időszorzás
SUB (ANY_NUM, ANY_NUM)	ANY_NUM	kivonás
SUB (TIME, TIME)	TIME	időkivonás
SUB (DATE, DATE)	TIME	időkivonás
SUB (TOD, TIME)	TOD	időkivonás
SUB (TOD, TOD)	TIME	időkivonás
SUB (DT, TIME)	DT	időkivonás
SUB (DT, DT)	TIME	időkivonás
DIV (ANY_NUM, ANY_NUM)	ANY_NUM	osztás

standard függvény (a bemeneti paraméterekkel)	függvény-érték adattípusa	jelentés
DIV (TIME, ANY_NUM)	TIME	időosztás
MOD (ANY_NUM, ANY_NUM)	ANY_NUM	maradékértéket adó osztás
MOVE(ANY_NUM,ANY_NUM)	ANY_NUM	egyenlőség
<i>Léptető függvények (IN1,N)</i>		
SHL (ANY_BIT, N)	ANY_BIT	bitmintát adott értékkel balra tolni
SHR (ANY_BIT, N)	ANY_BIT	bitmintát adott értékkel jobbra tolni
ROL (ANY_BIT, N)	ANY_BIT	bitmintát adott értékkel balra forgatni
ROR (ANY_BIT, N)	ANY_BIT	bitmintát adott értékkel jobbra forgatni
<i>Bitsoros függvények (IN1,IN2)</i>		
AND (ANY_BIT, ANY_BIT)	ANY_BIT	bitsoros ÉS összekapcsolás
OR (ANY_BIT, ANY_BIT)	ANY_BIT	bitsoros VAGY összekapcsolás
XOR (ANY_BIT, ANY_BIT)	ANY_BIT	bitsoros XOR összekapcsolás
NOT (ANY_BIT, ANY_BIT)	ANY_BIT	bitsoros NEGÁLÁS
<i>Kiválasztó függvények (IN1,IN2)</i>		
SEL (G,ANY, ANY)	ANY	bin. kiválasztás1 –2-ből
MAX (ANY,ANY)	ANY	maximum
MIN (ANY,ANY)	ANY	minimum
LIMIT (MN,ANY,MX)	ANY	korlátozás
MUX (K,ANY, ANY)	ANY	multiplexer (1-N-ből)
<i>Összehasonlító függv. (IN1,IN2)</i>		
GT (ANY,ANY)	BOOL	nagyobb, mint
GE (ANY,ANY)	BOOL	nagyobb, egyenlő
EQ (ANY,ANY)	BOOL	egyenlő
LT (ANY,ANY)	BOOL	kisebb, mint
LE (ANY,ANY)	BOOL	kisebb, egyenlő
NE (ANY,ANY)	BOOL	nem egyenlő
<i>Karakter sor függvények (IN1,IN2)</i>		
LEN (STRING)	INT	karakter sor hossza
LEFT (STRING,L)	STRING	karakter sor balról
RIGHT (STRING,L)	STRING	karakter sor jobbról
MID (STRING,L,P)	STRING	karakter sor középről
CONCAT (STRING,STRING)	STRING	karakter sor összefűzés
INSERT (STRING,STRING,P)	STRING	karakter sor beszúrás
DELETE (STRING,L,P)	STRING	karakter sor törlés
REPLACE(STRING,STRING,L,P)	STRING	karakter sor csere
FIND (STRING,STRING)	INT	pozíció keresés

A táblázat rövidítéseinek magyarázata

bemenet	jelentés	adattípus
N	a léptetendő bitek száma	UINT
L	balpozíció a karakter soron belül	ANY_INT
P	pozíció a karakter soron belül	ANY_INT
G	a kiválasztandó elem a 2 db bemenetből	BOOL
K	a kiválasztandó elem N db bemenetből	ANY_INT
MN	minimális érték a limitáláshoz	ANY
MX	maximális érték a limitáláshoz	ANY

## Standard függvényblokkok

Az IEC-1131-3 szabvány számos függvényblokkot definiál, ezzel biztosítva, hogy rendelkezésére álljanak a legfontosabb, tárolási tulajdonsággal rendelkező függvényblokkok.

A szabványban leírt függvényblokkok az alábbi öt csoportba sorolhatók:

1. Bistabil elemek (flip-flopok, R/S-tárolók).
2. Élkiértékelők.
3. Számlálók.
4. Időzítők.
5. Kommunikációs függvényblokkok.

Az alábbi táblázatban összefoglaljuk a a standard függvényblokkok jellemzőit, kivéve a kommunikációs függvényblokkokat. A kommunikációs függvényblokkokat a szabvány külön fejezete írja le (IEC 1131-5), ezt a gyártók általában saját függvényblokkokkal is kiegészítik, tárgyalásával jelen munkában nem foglalkozunk.

a függvényblokk neve és a bemeneti paraméterek	kimeneti érték(ek)	jelentés
<i>R/S tárolók</i>		
SR (S1,R)	Q	SET domináns
RS (R,S1)	Q	RESET domináns
<i>Élkiértékelők</i>		
R_TRIG (CLK)	Q	felfutó él felismerése
F_TRIG (CLK)	Q	lefutó él felismerése
<i>Számlálók</i>		
CTU (CU,R,PV)	Q,CV	felfelé számláló
CTD (CD,LD,PV)	Q,CV	lefelé számláló
CTUD (CU,CD,R,LD,PV)	QU,QD,CV	fel/le-számláló
<i>Időzítők</i>		
TP (IN,PT)	Q,ET	impulzusadó
TON (IN,PT)	Q,ET	bekapcsolás-késleltetési időzítő
TOF (IN,PT)	Q,ET	kikapcsolás-késleltetési időzítő
RTC (EN,PDT)	Q,CDT	valósídejű óra

**A standard függvényblokkok be- és kimeneti paramétereinek értelmezése és adattípusa**

bemenet/kimenet	jelentés	adattípus
R	RESET jel bemenet	BOOL
S	SET jel bemenet	BOOL
R1	RESET domináns	BOOL
S1	SET domináns	BOOL
Q	kimenet (kétállapotú)	BOOL
CLK	(ütem) bemenet (Clock)	BOOL
CU	(Count Up) számlálás felfelé bemeneti impulzus	R_EDGE
CD	(Count Down) számlálás lefelé bemeneti impulzus	R_EDGE
LD	(Load) számlálóérték betöltése bemenet	INT
PV	(Preset Value) számlálóérték	INT
QD	lefelészámlálás kimenete (Down) =1, ha CV=0	BOOL
QU	felfelészámlálás kimenete (Up) =1, ha CV≥PV	BOOL
CV	Aktuális számlálóérték (Current Value)	INT
IN	időzítő indítása (INput)	BOOL
PT	időérték (Preset Time)	TIME
ET	az indítástól eltelt idő (Elapsed Time)	TIME
PDT	dátum/időérték (Preset Date and Time)	DT
CDT	aktuális dátum/időérték (Current Date and Time)	DT

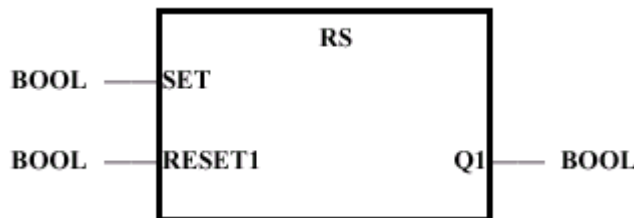
A standard függvényblokkok kimeneti értéke az első futtatás során nulla, kivéve a valósídejű órát.

A standard függvényblokkok bemeneti paramétereit kulcsszónak minősítik. A standard függvényblokkokat a programkészítés során úgy tudjuk felhasználni, hogy a deklarációs részben egy egyedi névhez mint **FB-típust** rendeljük hozzá. A POU-törzsben ezen egyedi névvel dolgozunk. A paraméterátadás a függvényblokkoknál tárgyalt módon lehetséges.

**Tárolók**

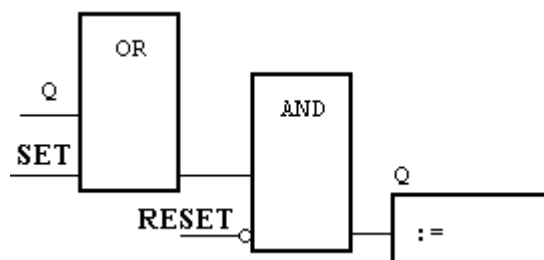
**RS tároló**

Funkciótervbeli jelölése:



6. ábra RS-tároló

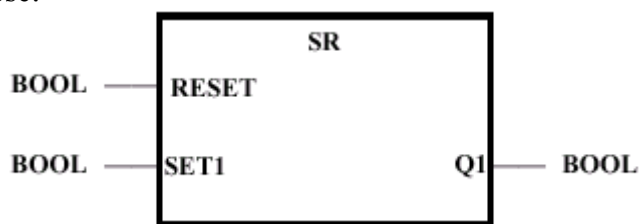
RESET domináns flip-flop. Ha a SET és RESET jel egyidejűleg 1 értékű, a RESET jel határozza meg a kimenetet, vagyis Q1=0. Az RS függvényblokk az alábbi funkciótervvel leírható algoritmus szerint működik:



7. ábra Az RS-tároló belső algoritmus

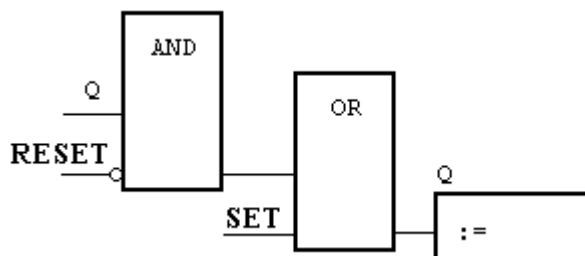
### SR tároló

Funkciótervbeli jelölése:



8. ábra SR-tároló

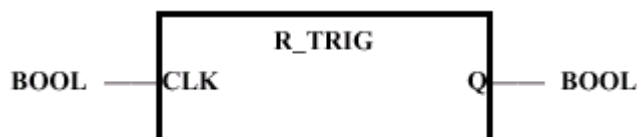
SET domináns flip-flop. Ha a SET és RESET jel egyidejűleg 1 értékű, a SET jel határozza meg a kimenetet, vagyis  $Q1=1$ . Az SR függvényblokk az alábbi funkciótervvel leírható algoritmus szerint működik:



9. ábra Az SR-tároló belső algoritmus

### Felfutó él detektálása: az R\_TRIG függvényblokk

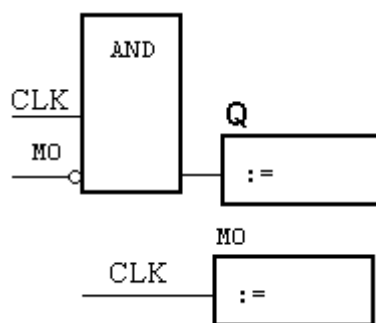
Ábrázolás funkciótervben:



10. ábra A felfutó él detektálása

A **Q** kimenet abban a programciklusban 1, amelyben a **CLK** bemeneti változó értéke 0-ról 1-re vált.

A függvényblokk algoritmus funkciótervben:



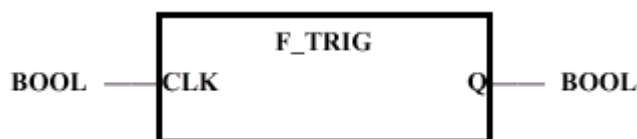
11. ábra Impulzus felfutó élre

Utasításlistában:

LD	CLK
ANDN	M0
ST	Q
LD	CLK
ST	M0 .

**Lefutó él detektálása: az F\_TRIG függvényblokk**

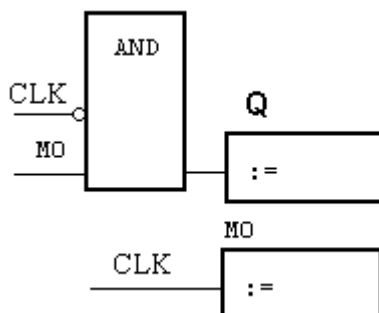
Ábrázolás funkciótervben:



12. ábra A lefutó él detektálása

A **Q** kimenet abban a programciklusban 1, amelyben a **CLK** bemeneti változó értéke 1-ről 0-ra vált.

A függvényblokk algoritmus funkciótervben:



13. ábra Impulzus lefutó élre

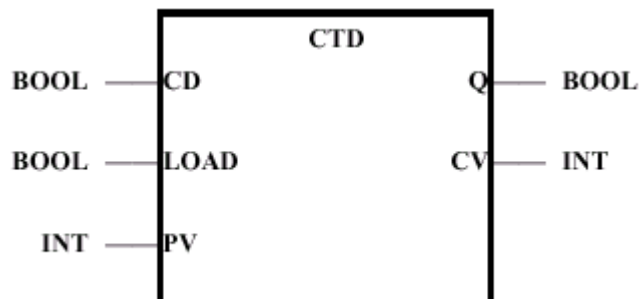
Utasításlistában:

LDN	CLK
AND	M0
ST	Q
LD	CLK
ST	M0 .

## A számlálók

### CTD (Count Down) lefelé számláló

Ábrázolása funkciótervben:



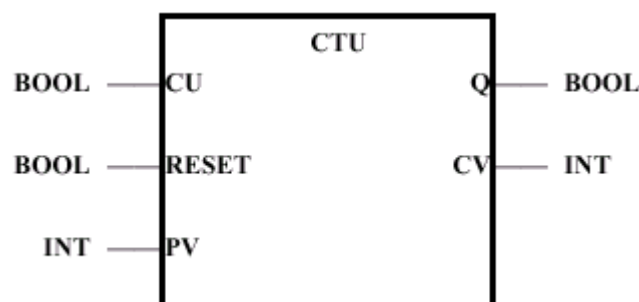
14. ábra A lefelé számláló

A be/kimeneti jelek értelmezése:

jelölés	jelentés
CD	A CD bemeneten megjelenő jel felfutó élre a számláló értékét eggyel csökkenti.
LOAD	A LOAD bemeneten lévő jel felfutó élre a számláló értékét PV-vel teszi egyenlővé (a számláló kezdeti értékének beállítása).
PV	A számláló kezdeti értéke. Alapértelmezés=0.
Q	A számláló kétállapotú kimenete. Értéke=1, ha a számláló értéke: CV=0.
CV	A számláló értéke.

### CTU (Count Up) felfelé számláló

Ábrázolása funkciótervben:



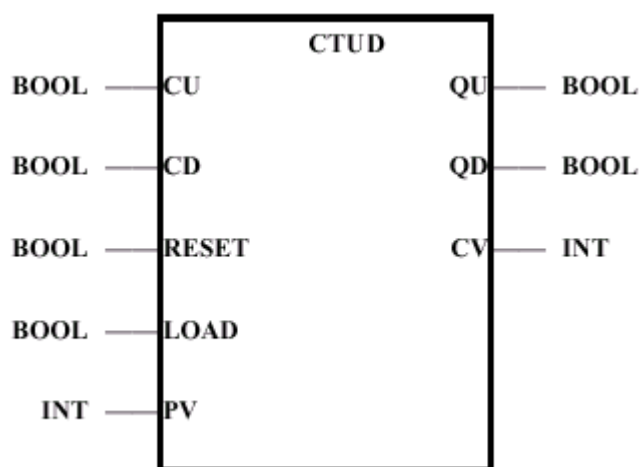
15. ábra A felfelé számláló

A be/kimeneti jelek értelmezése:

jelölés	jelentés
CU	A CU bemeneten megjelenő jel felfutó élre a számláló értékét eggyel növeli
RESET	A RESET bemeneten lévő jel felfutó élére a számláló értékét 0-val teszi egyenlővé (a számláló kezdeti értékének beállítása).
PV	A számláló felső határértéke.
Q	A számláló kétállapotú kimenete. Értéke=1, ha a számláló értéke: $CV \geq PV$ .
CV	A számláló értéke.

### CTUD (Count Up-Down) fel-le számláló

Ábrázolása funkciótervben:



16. ábra A fel/le számláló

A be/kimeneti jelek értelmezése:

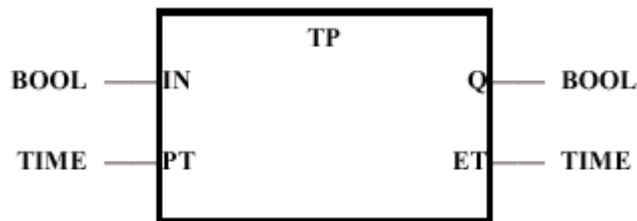
jelölés	jelentés
CU	A CU bemeneten megjelenő jel felfutó élre a számláló értékét eggyel növeli.
CD	A CD bemeneten megjelenő jel felfutó élre a számláló értékét eggyel csökkenti.
RESET	A RESET bemeneten lévő jel felfutó élére a számláló értékét 0-val teszi egyenlővé (a számláló kezdeti értékének beállítására).
LOAD	A LOAD bemeneten lévő impulzus jelre a számláló értékét PV-vel teszi egyenlővé (a számláló kezdeti értékének beállítása).
PV	A számláló kezdeti értéke. Alapértelmezés=0.
QU	A számláló kétállapotú kimenete. Értéke=1, ha a számláló értéke: $CV \geq PV$ .
QD	A számláló kétállapotú kimenete. Értéke=1, ha a számláló értéke: $CV = 0$ .
CV	A számláló értéke.

## Az időzítők

Célunk az általános, géptől független programfejlesztés elsajátítása, ezért az alábbiakban csak a szabványban rögzített időzítőket mutatjuk be. Gyártótól és típustól függően az időzítők palettája sokkal szélesebb is lehet.

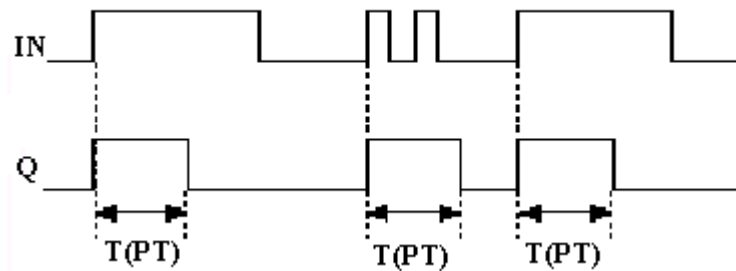
### Impulzus időzítő (TP = Time Pulse)

Funkciótervbeli jelölése:



17. ábra Az impulzus időzítő funkciótervbeli ábrázolása

Az időzítő viselkedését bemutató idődiagram:



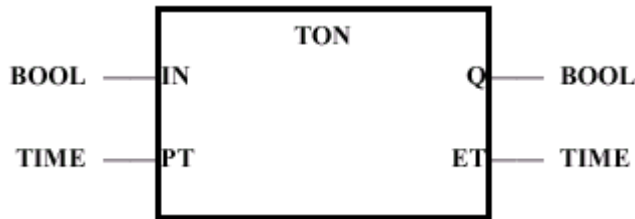
18. ábra Az impulzus időzítő idődiagramja

A be/kimeneti jelek értelmezése:

jelölés	jelentés
IN	Az IN bemeneten jelentkező felfutó élre indul az időzítés.
PT	A kimeneten megjelenő impulzus időtartamát állítja be. PT értékét a FB mindig csak IN felfutó élre kérdezi le. Köztes módosítása nincs hatással.
Q	Az időzítő kétállapotú kimenete. Beállítása az idődiagram szerint.
ET	Az indítás óta eltelt idő.

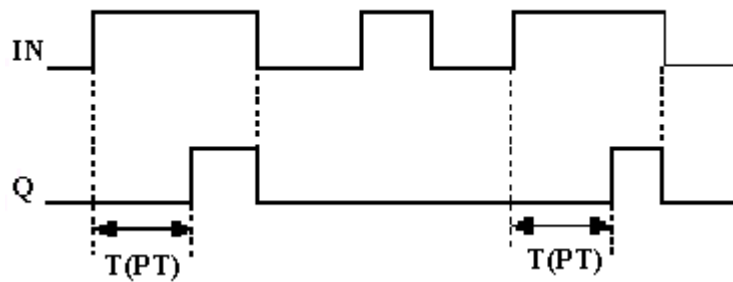
### Bekapcsolás-késleltetési időzítő

Funkciótervbeli jelölése:



19. ábra A bekapcsolás-késleltetési időzítő funkciótervbeli ábrázolása

Az időzítő viselkedését bemutató idődiagram:



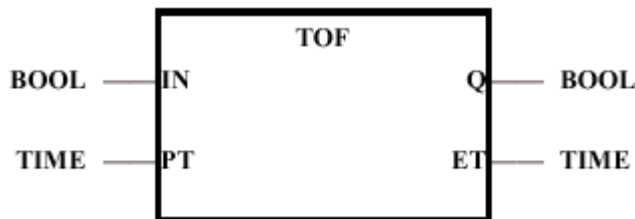
20. ábra A bekapcsolás-késleltetési időzítő idődiagramja

A be/kimeneti jelek értelmezése:

jelölés	jelentés
IN	Az IN bemeneten jelentkező felfutó élre indul az időzítés.
PT	A kimeneten megjelenő jel késleltetésének idejét adja meg.
Q	Az időzítő kétállapotú kimenete. Beállítása az idődiagram szerint.
ET	Az indítás óta eltelt idő. Értéke nem lehet nagyobb PT-nél.

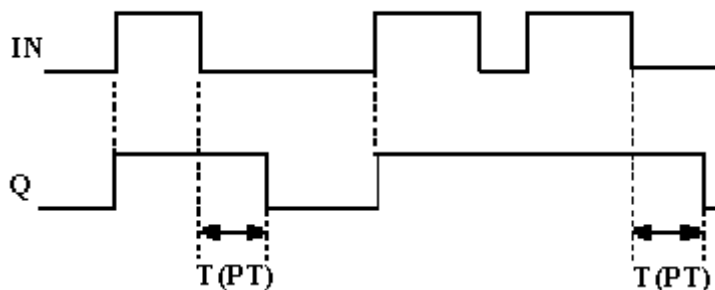
### Kikapcsolás-késleltetési időzítő

Funkciótervbeli jelölése:



21. ábra A kikapcsolás-késleltetési időzítő funkciótervbeli ábrázolása

Az időzítő viselkedését bemutató idődiagram:



22. ábra A kikapcsolás-késleltetési időzítő idődiagramja

A be/kimeneti jelek értelmezése:

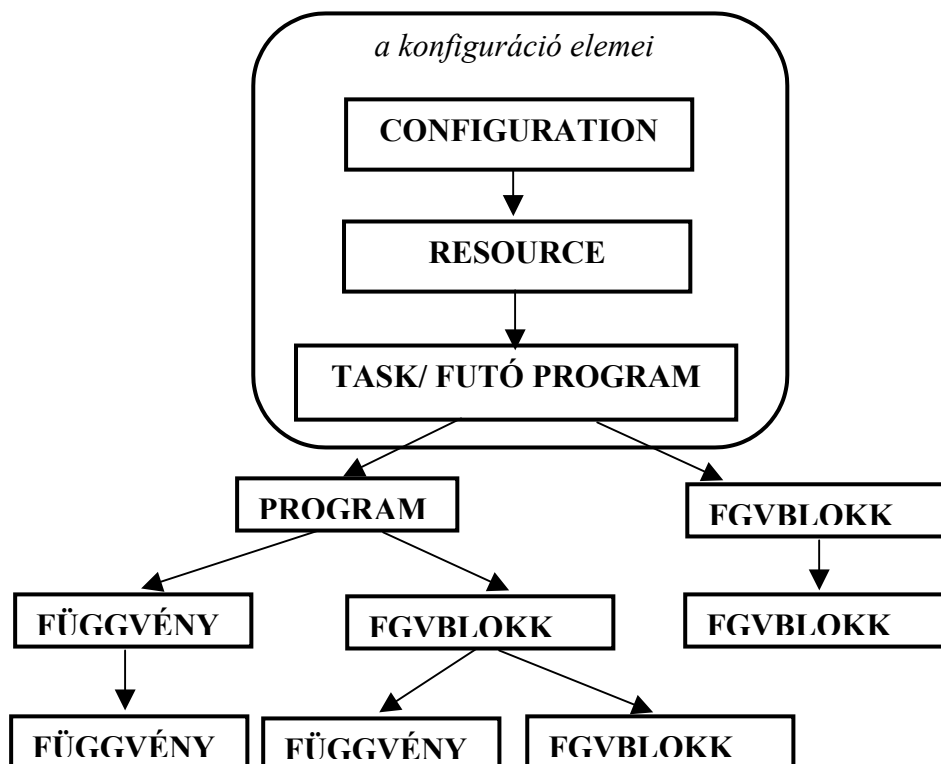
jelölés	jelentés
IN	Az IN bemeneten jelentkező felfutó élre a kimenet 1-re vált.
PT	Az IN bemenet lefutó éle után PT ideig a Q kimeneten még fenntartandó az 1 jel.
Q	Az időzítő kétállapotú kimenete. Beállítása az idődiagram szerint.
ET	A lefutó él óta eltelt idő. Értéke nem lesz negatív.

## A PLC konfigurálása

Az IEC-1131-3 szabvány ajánlása szerint a strukturált szoftvermodell biztosítja a felhasználói programok könnyebb áttekinthetőségét, egyenkénti szintaktikai ellenőrzését, hordozhatóságát. Ebben a fejezetben az IEC-1131-3 szabvány azon konfigurációs elemeit ismertetjük, amelyek a programszervezési egységek összehangolásának fontos segédeszközei. Itt definiáljuk a programok futási tulajdonságait, a kommunikációs kapcsolatokat és a hardver összerendeléseket. A mai modern operációs rendszerek a PLC oldaláról támogatják ezeket a konfigurációs elemeket. Egy CPU például több programot is tud egyszerre futtatni (multi-taszking).

## A PLC projekt felépítése

A PLC-projekt, amelyet egy jól körülhatárolható irányítási feladat megvalósítására hoznak létre, az alábbi ábrán látható hierarchikus felépítéssel jellemezhető. Láthatjuk, hogy az előző fejezetekben tárgyalt programszerkezet fölötti hierarchiaszinteken megjelenik a taszk (TASK) a futó programmal, az erőforrás (RESOURCE) és a konfiguráció (CONFIGURATION).



A PLC projekt felépítése az IEC-1131-3 szabvány szerint

A POU-kból képezik a hívási hierarchiát, a konfigurációs elemek pedig arra szolgálnak, hogy ezekhez a POU-khoz futtatói sajátosságokat és hardverelemeket rendeljenek hozzá. Részletezve:

- a programok és függvényblokkok futási jellemzőit,
- a kommunikációs kapcsolatokat,
- a programváltozók leképezését a PLC hardvercímeire.

## A konfiguráció összetevői

A konfigurációs elemek határozzák meg a PLC-rendszer valós összetevőit:

- a konfigurációt:* a PLC-rendszert, mint egy keretbe épített, akár több (elosztott) központi egységgel bíró, folyamatközeli (gépegység szintű) irányítórendszert.
- az erőforrást:* (esetleg multitaskingot lehetővé tevő) CPU-t.
- a taszkot:* a programok és program típusú függvényblokkok futási sajátosságait. (A PLC program egyediesítése.)
- a futó programot:* a programból ill. függvényblokkból és a TASK-ból képzett egységet.

A CPU főprogramja egy PROGRAM típusú POU.

A főprogramokhoz és a függvényblokkokhoz hozzárendeljük a futási sajátosságait, mint pl. a periodikus végrehajtást, prioritási szintet. A futó program egy rögzített (lezárt) tulajdonságokkal rendelkező programegység, amely természetesen egy adott CPU-n képes csak futni.

## A CONFIGURATION jellemzői

Az IEC-1131-3 szabvány a CONFIGURATION elemet használja arra, hogy a PLC rendszer erőforrásait (RESOURCE) összefogja és biztosítsa közöttük az adat és információcserét.

A konfiguráció részei:

### **CONFIGURATION** *konfiguráció-név*

Típusdefiníciók  
Globális deklarációk  
RESOURCE-deklaráció  
ACCESS-deklaráció

### **END\_CONFIGURATION**

A konfigurációban deklarált típusokat, globális változókat az egész projekt látja és használhatja. (Több CPU is.) A konfigurációk közötti adatcserét a VAR\_ACCESS segítségével hozhatjuk létre. Léteznek ezen kívül egyéb, konfigurációk közötti kommunikációt biztosító függvények is, ezek az IEC-1131-5 szabvány írja le.

A konfigurációra példa:

```
CONFIGURATION PLC_gep1
VAR_GLOBAL ... END_VAR
RESUORCE CPU_szszalagON CPU_001 END_RESOURCE
RESUORCE CPU_henger ON CPU_002 END_RESOURCE
VAR_ACCESS ... END_VAR
```

## A RESOURCE jellemzői

A RESOURCE deklarálás biztosítja a TASK-ok hozzárendelését a PLC-rendszer fizikai erőforrásaihoz.

. Az erőforrás részei:

**RESOURCE** *erőforrás-név* **ON** erőforrás

Globális deklarációk

TASK-deklaráció

**END\_RESOURCE**

Az *erőforrás-név* lesz a PLC-CPU szimbolikus neve. A RESOURCE-ban deklarált globális változók csak az adott CPU-n belül láthatók és használhatók.

Az erőforráson belül rendeljük hozzá a TASK-hoz a program típusú POU-t.

A konfiguráció és az erőforrás nem tartalmaz parancs részt, csak deklarációs része van.

Az erőforrás deklarációra példa:

```
RESOURCE CPU_szszalag ON CPU_001
TASK ...
PROGRAM ... WITH ...
END_RESOURCE
RESOURCE CPU_henger ON CPU_002
TASK ...
PROGRAM ... WITH ...
END_RESOURCE
```

### A TASK és a futó program

A TASK definíció feladata a program és függvényblokkjainak futási sajátosságait rögzíteni. Régebbi PLC-rendszerekben szokásos volt speciális blokkok megadása (pl. szervezői blokk), amelyek rögzített futtatási sajátosságokkal rendelkeztek. Ezeket tölthette fel utasításokkal a felhasználó, ha ciklikus vagy megszakítás/esemény feldolgozást kívánt. A TASK bevezetésével ezen tulajdonságokat expliciten és gyártótól függetlenül lehet megfogalmazni. Ezáltal a programok jobban dokumentálhatók és könnyebben várakoztathatók.

TASK deklarálásra példa:

**TASK** *task-név* (task-tulajdonságok)

**PROGRAM** *program-név* **WITH** *task-név* : *progr-név* (PROGRAM – csatlakoztatás)

A futásidejű program neve a *program-név* lesz. Ez tulajdonképpen egy *progr-név* típusú POU instancálása, egyediesítése. A (PROGRAM –csatlakoztatás) adja meg a formális paramétereknek megfelelő aktuális paraméterek listáját.

A TASK lehetséges tulajdonságait a következő táblázatba foglaltuk össze.

<b>TASK-paraméter</b>	<b>jelentés</b>
SINGLE	A paraméterhez rendelt jel emelkedő éle indítja el a program egyszeri lefutását.
INTERVAL	Ha ez a paraméter nem egyenlő nullával, akkor a TASK-hoz rendelt program ciklikusan fut. Ez a paraméter szolgál a ciklusidő megadására és túllépésének ellenőrzésére.
PRIORITY	A TASK-hoz rendelt program prioritását adja meg az erőforráson egyidejűleg futó többi programhoz viszonyítva.

A prioritás hatása attól függ, hogy a PLC operációsrendszere milyen módon szabályozza több TASK feldolgozását. (Tehát implementációfüggő.) Általában kétféle feldolgozási mód lehetséges. Az egyik szerint (preemptive scheduling) a futó taszk azonnal megszakad, ha egy magasabb prioritású taszk futni akar. A másik módszer a taszk a futását nem szakítja meg, az lefut. Ezután a rendszer a várakozó taszkok közül a legnagyobb prioritásút indítja el. (non-preemptive scheduling) Mindkét eljárás célja, hogy a legmagasabb prioritású taszknak adja át az erőforrás felügyeletét.

#### **Példa TASK deklarációra**

```
TASK T_gyors      (INTERVAL:=t#8ms, PRIORITY:=1);
PROGRAM berendezes WITH T_gyors :
  progrA(szabpar:=%MW3,szabert:=hibakod)
```

```
TASK T_megszakit (SINGLE := trigger, PRIORITY:=1);
PROGRAM berendezes WITH T_megszakit : progrB
```

Kis PLC rendszerekben (egy erőforrás, egyetlen futtatható programmal) a konfiguráció szerepét teljesen átveheti a főprogram. A programban deklaráljuk a rendszerben szükséges globális változókat, a közvetlen leképezésű és a szimbolikus változókat. A futási tulajdonságokat a fejlesztőrendszer ill. a PLC képességei (implicit) behatárolják, beállítják.

## PÉL DATÁR

Az IEC-1131-3 szabvány rövid ismertetése után, a jegyzet további fejezeteiben példaprogramokon keresztül ismerkedünk meg a PLC programozásának technikájával. Az irányított technológiai folyamattal meglévő folyamatos jelkapcsolat és a sajátos felhasználói programfuttatás (jellemzően ciklikus feldolgozás) a programozótól, a klasszikus programfejlesztésnél megszokottól kissé eltérő látásmódot, gondolkodásmódot kíván. A példaprogramok sorával ezt a problémafelismerő és megoldó képességet szeretnénk a hallgatókban kifejleszteni. A példák a nehézségüknek megfelelő sorrendben követik egymást. A feladatok egy-egy kiemelt téma ismertetését, begyakoroltatását célozzák, nem törekedtünk minden esetben a teljes technológiai folyamatnak, ill. az összes biztonságtechnikai előírásnak megfelelő vezérlőalgorithmus kidolgozására. Az esettanulmányokhoz a legtöbb ötletet a [8] irodalomból vettük. A programokat Az IEC-1131-3 szabványnak megfelelően, az S40 programfejlesztői rendszerben készítettem és a Klöckner–Moeller cég PS4-341-MM1 programozható vezérlőjén teszteltem.

## Követővezérlések

### Szellőztetés felügyelete

Egy mélygarázsba 4 db szellőztetőt építettek be. A szellőztetés felügyeletét a szellőzővezetékekben lévő áramlásjelzők látják el. A garázs bejáratánál a szellőztetéstől függően jelzőlámpa engedélyezi a behajtást.

#### Jelzések:

- Ha négy, vagy három ventilátor működik, ezek gondoskodnak a megfelelő szellőzésről, és a lámpa **zöldet** mutat.
- Ha két ventilátor működik, a lámpa **sárgát** jelez.
- Ha kettőnél kevesebb ventilátor működik, **piros** jelzést kell adni.

#### Összerendelési táblázat:

Bemenetek	Jel	Logikai hozzárendelés	Cím
1. áramlásjelző	I1	1. ventilátor üzemel: I1=1	I0.0
2. áramlásjelző	I2	2. ventilátor üzemel: I2=1	I0.1
3. áramlásjelző	I3	3. ventilátor üzemel: I3=1	I0.2
4. áramlásjelző	I4	4. ventilátor üzemel: I4=1	I0.3
Kimenetek			
Piros lámpa	P	világít, ha: P=1	Q0.2
Sárga lámpa	S	világít, ha: S=1	Q0.1
Zöld lámpa	Z	világít, ha: Z=1	Q0.0

#### A függvénytáblázat:

OKT	I4	I3	I2	I1	P	S	Z
00	0	0	0	0	1	0	0
01	0	0	0	1	1	0	0
02	0	0	1	0	1	0	0
03	0	0	1	1	0	1	0
04	0	1	0	0	1	0	0
05	0	1	0	1	0	1	0
06	0	1	1	0	0	1	0
07	0	1	1	1	0	0	1
10	1	0	0	0	1	0	0
11	1	0	0	1	0	1	0
12	1	0	1	0	0	1	0
13	1	0	1	1	0	0	1
14	1	1	0	0	0	1	0
15	1	1	0	1	0	0	1
16	1	1	1	0	0	0	1
17	1	1	1	1	0	0	1

## Karno-tábla

Piros (P):

		I1				
		1	1	0	1	
I2	1	1	0	0	1	
	1	0	0	1	0	
	0	0	1	1	0	
	1	0	1	0	1	I4
		I3				

$\bar{I}1\bar{I}2\bar{I}4$  V  
 $\bar{I}1\bar{I}2\bar{I}3$  V  
 $\bar{I}1\bar{I}3\bar{I}4$  V  
 $\bar{I}2\bar{I}3\bar{I}4$

Sárga (S):

		I1				
		0	1	1	0	
I2	0	0	1	0	1	
	0	1	0	1	0	
	1	0	0	1	0	
	0	1	0	1	1	I4
		I3				

$\bar{I}1\bar{I}2\bar{I}3\bar{I}4$  V  
 $I1\bar{I}2\bar{I}3\bar{I}4$  V  
 $\bar{I}1\bar{I}2\bar{I}3\bar{I}4$  V  
 $\bar{I}1\bar{I}2\bar{I}3\bar{I}4$  V  
 $I1\bar{I}2\bar{I}3\bar{I}4$  V  
 $I1\bar{I}2\bar{I}3\bar{I}4$

Zöld (Z):

		I1				
		0	1	0	1	
I2	0	0	0	1	0	
	0	1	0	1	0	
	1	0	1	1	0	
	0	1	0	1	1	I4
		I3				

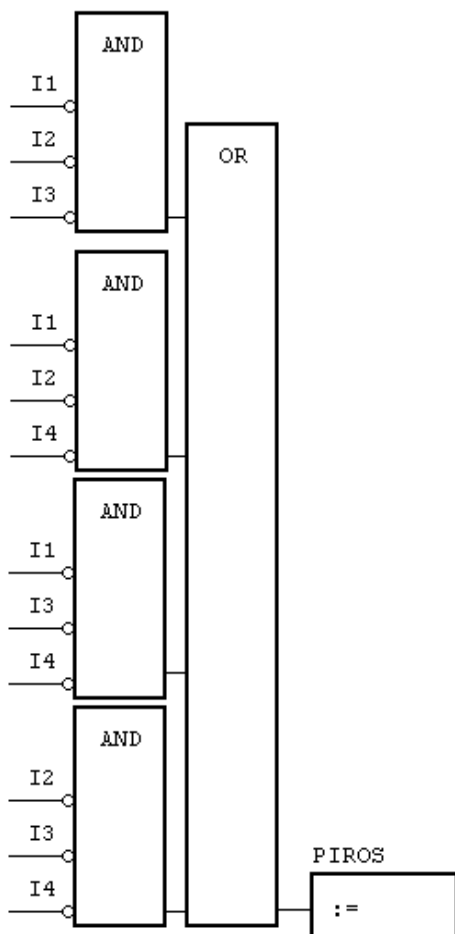
$I1\bar{I}3\bar{I}4$  V  
 $I2\bar{I}3\bar{I}4$  V  
 $I1\bar{I}2\bar{I}3$  V  
 $I1\bar{I}2\bar{I}4$

Mivel egy lámpának mindig világítania kell, elegendő, ha a kapcsolási feltételeket csak két lámpára írjuk meg, a harmadik pedig akkor lesz igaz, ha a másik kettő hamis. Mivel a sárga logikai függvénye a leghosszabb, ezért legyen:

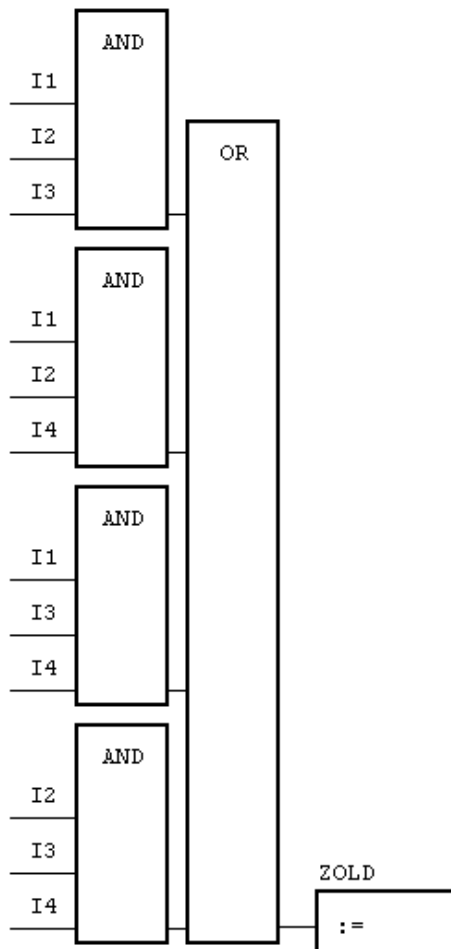
$$S = \bar{P} \& \bar{Z}$$

### Funkcióterv

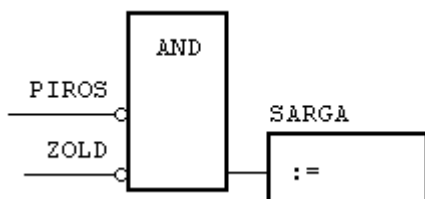
#### Piros lámpa világít:



#### Zöld lámpa világít:



#### Sárga lámpa világít:



## Utasításlista

PROGRAM SZELLOZ

VAR

```
I1 AT %I0.0.0.0.0:  BOOL;
I2 AT %I0.0.0.0.1:  BOOL;
I3 AT %I0.0.0.0.2:  BOOL;
I4 AT %I0.0.0.0.3:  BOOL;
PIROS      AT      %Q0.0.0.0.2:
BOOL;
SARGA      AT      %Q0.0.0.0.1:
BOOL;
ZOLD       AT      %Q0.0.0.0.0:
BOOL;
```

END\_VAR

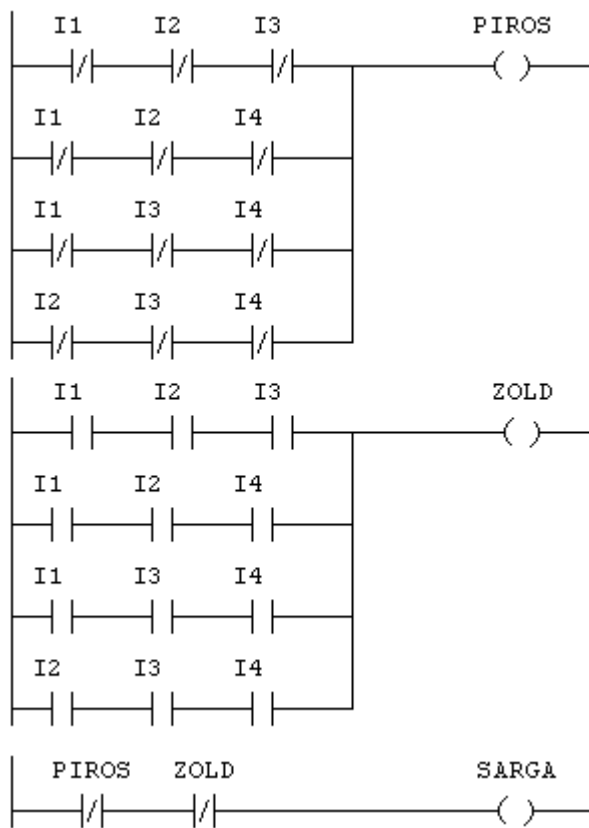
```
LDN      I1
ANDN     I2
ANDN     I3
OR(      I1
NOT
ANDN     I2
ANDN     I4
)
OR(      I1
NOT
ANDN     I3
ANDN     I4
)
OR(      I2
NOT
ANDN     I3
ANDN     I4
)
ST       PIROS

LD       I1
AND      I2
AND      I3
OR(      I1
AND      I2
AND      I4
)
OR(      I2
AND      I3
AND      I4
)
ST       ZOLD

LDN      PIROS
ANDN     ZOLD
ST       SARGA
END_PROGRAM
```

## Létradiagram

A programtörzs létradiagramban ábrázolva:



### Követővezérlés tervezése döntési táblázattal

A be- és kimeneti változók közötti kapcsolatot döntési táblázat segítségével is felírhatjuk. (DIN 66241). A döntési táblázat a döntési feladatok táblázatos leírása. Viszonylag kevés döntési szabállyal leírható vezérlési feladatoknál célszerű alkalmazni. A táblázat két fő részre osztható: a feltételrészre és a következmény részre.

	Problémaleírás	Szabályok					Egyéb- ként
		R1	R2	R3	...	Rn	
<b>Feltételek</b>	1.bemenő változó 2.bemenő változó . n.bemenő változó	Feltétel vagy esetleírások szabályok megadásával. (Az olyan bemeneti jelkombinációra, amelyre nincs szabály, az EGYÉB oszlop vonatkozik!)					
<b>Következmények</b>	1.kimenő változó 2.kimenő változó . n.kimenő változó	A feltételektől függő következmények (akciók) jelölése.					

Jelállapotok: **0** : hamis  
**1** : igaz  
 - : nincs jelentősége a feltételnek az adott szabályban.

A függvénytáblázattól csak a változók és következményeik elrendezésében különbözik, így a döntési táblázat fogalmilag nem jelent új leírási módot. Alkalmazásának előnye akkor jelentkezik, ha a vezérlési feladat visszavezethető kombinációs hálózatra és nincs szükség a lehetséges bemeneti jelkombinációk mindegyikére. A döntési táblázattal leírt vezérlési feladat a függvénytáblázathoz hasonlóan transzformálható át vezérlőprogrammá. Az alábbi vezérlési feladat példa a döntési táblázat használatára. 6 db bemenőjel esetén  $2^6=64$  a lehetséges bemenőjel-kombinációk száma. Egy ilyen nagyméretű igazságtáblázat nehezen tekinthető át, nehezen kezelhető.

### Stancolás

A gép hengere csak az alábbi feltételek esetén működtethető:

1. A két kézi nyomógomb egyidejűleg lenyomva (most nincs kétkezes reteszelési előírás).
2. A védőrács zárva (leeresztve) és a lábkapcsoló benyomva.
3. A védőrács zárva és a két kézi nyomógomb közül az egyiket benyomták.

Ezen kívül mindhárom esetben szükséges még, hogy a készüléket már bekapcsolták és a kivágóminta a helyén van.

### Összerendelési táblázat

Bemenetek	Jel	Logikai összerendelés	Cím
BE - kapcsoló	S1	bekapcsolva: S1=1	I0.0
1. kézi nyomógomb	S2	benyomva: S2=1	I0.1
2. kézi nyomógomb	S3	benyomva: S3=1	I0.2
Lábnyomógomb	S4	benyomva: S4=1	I0.3
Védőrács	S5	Védőrács leeresztve: S5=1	I0.4
Kivágóminta	S6	Kivágóminta a helyén: S6=1	I0.5
<b>Kimenetek</b>			
Préshenger	P	leeresztve: P=1	Q0.0

### A döntési táblázat

	Problémaleírás		Szabályok				Egyéb- ként
			47	63	65	71	
	BE - kapcsoló	S1	1	1	1	1	
	1. kézi nyomógomb	S2	1	1	0	0	
	2. kézi nyomógomb	S3	1	0	1	0	
	Lábnyomógomb	S4	0	0	0	1	
	Védőrács	S5	0	1	1	1	
	Kivágóminta	S6	1	1	1	1	
	Préshenger	P	1	1	1	1	0

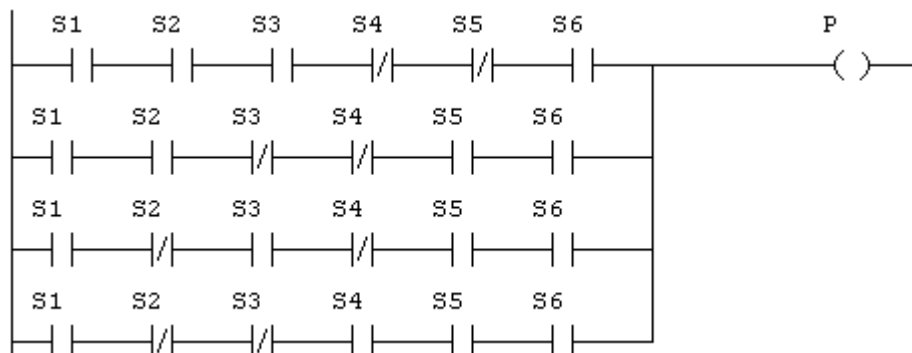
### A redukált függvénytáblázat

S6	S5	S4	S3	S2	S1	P
1	0	0	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	0	1	1
1	1	1	0	0	1	1

A diszjunktív normál forma:

$$P = \overline{S6}S5\overline{S4}S3S2S1 \vee S6S5\overline{S4}S3S2S1 \vee \overline{S6}S5\overline{S4}S3S2S1 \vee S6S5S4S3S2S1$$

## Létradiagram



## A program utasításlistája

PROGRAM STANC

VAR

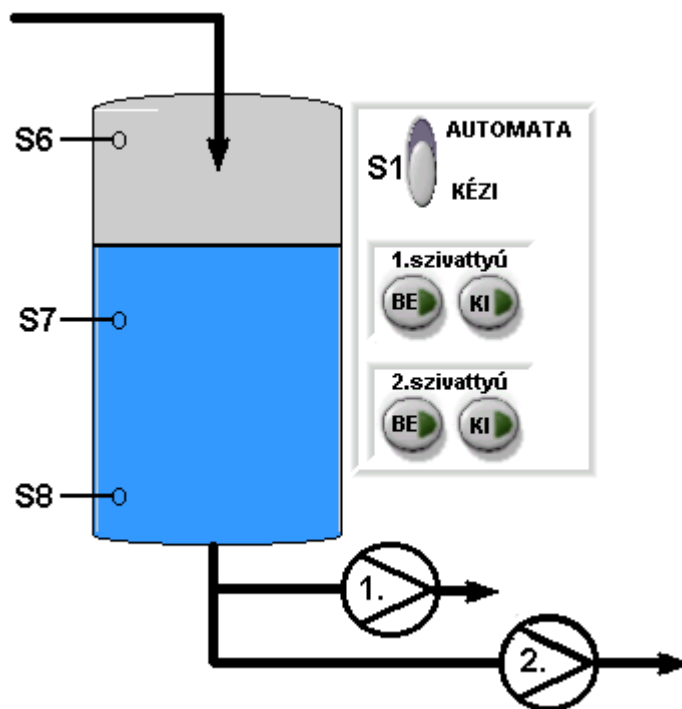
S1 AT %I0.0.0.0.0: BOOL;  
 S2 AT %I0.0.0.0.1: BOOL;  
 S3 AT %I0.0.0.0.2: BOOL;  
 S4 AT %I0.0.0.0.3: BOOL;  
 S5 AT %I0.0.0.0.4: BOOL;  
 S6 AT %I0.0.0.0.5: BOOL;  
 P AT %Q0.0.0.0.0: BOOL;

END\_VAR

LD(	S1	OR(	S1
AND	S2	ANDN	S2
AND	S3	ANDN	S3
ANDN	S4	AND	S4
ANDN	S5	AND	S5
AND	S6	AND	S6
)		)	
OR(	S1	ST	P
AND	S2	END_PROGRAM	
ANDN	S3		
ANDN	S4		
AND	S5		
AND	S6		
)			
OR(	S1		
ANDN	S2		
AND	S3		
ANDN	S4		
AND	S5		
AND	S6		
)			

## Gyakorló feladat Szivattyúk vezérlése

A technológiai berendezés egy átmeneti folyadéktároló, a belépő folyadékáram mennyisége időben változhat. A tartályban 3 db szintérzékelőt építettek be, a felső kettő akkor ad jelet, ha a folyadékszint az érzékelőt elérte vagy föllette van, az alsó pedig akkor ad jelet, ha a folyadékszint alatta van. A tartály a kilépő vezetékbe épített két db szivattyúval üríthető le.



23. ábra Szivattyúk vezérlése

A vezérlésnek kézi és automata üzemmódot is kell biztosítania.

**Kézi üzemmódban** ( $S1=1$ ) a szivattyúkat a kezelőszemély működtetheti a szivattyúkhöz tartozó be- ill. kikapcsoló nyomógombokkal.

**Automata üzemmódban** ( $S1=0$ ) a vezérlésnek kell megakadályoznia a folyadék túlfolyását.

Emelkedő folyadékszintnél:

**S6** és **S7** között az 1. sz. szivattyú működjön;

**S6** felett mindkét szivattyú kapcsoljon be.

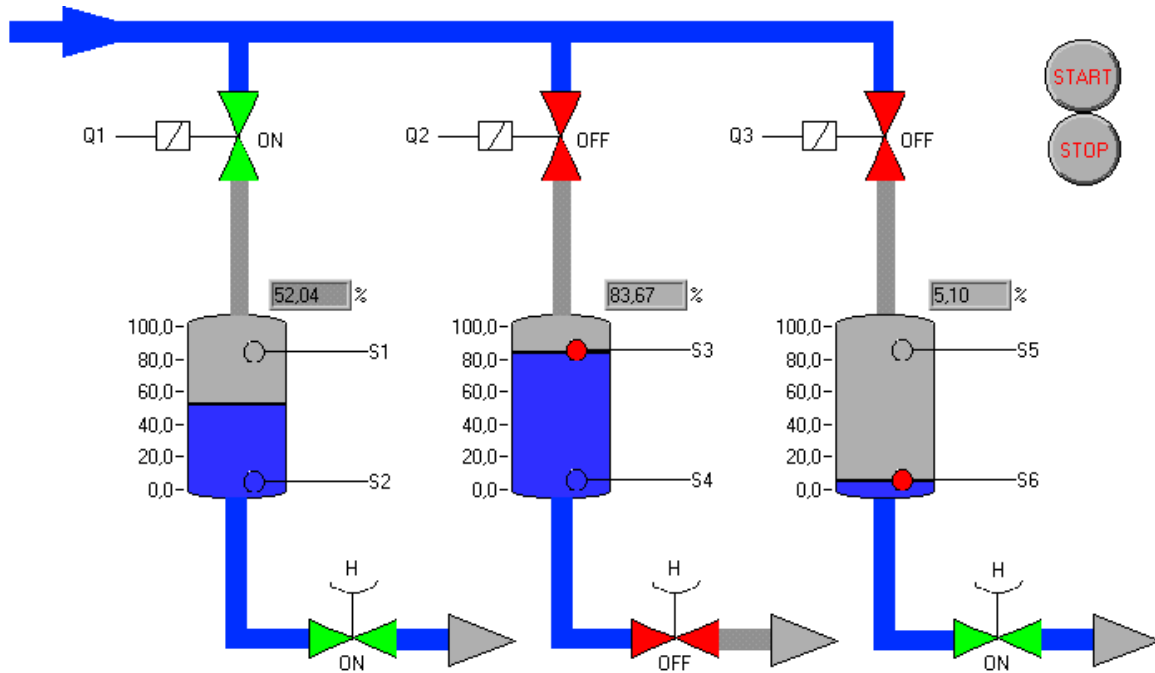
Csökkenő folyadékszintnél, ha **S8** szintérzékelő jelez, mindkét szivattyú álljon le.

**Feladat:** összerendelési táblázat, funkcióterv, utasításlista.

## Követővezérlés tárolással

### Tárolótartályrendszer: feltöltés vezérlése

Három tárolótartály tele állapotát az **S1, S3, S5** jeladók, az üres jelet az **S2, S4, S6** jeladók szolgáltatják az előbbi sorrendben. A vezérlésnek gondoskodnia kell arról, hogy üres jelzésnél egyszerre csak egy tartályt töltsön fel. A tartály feltöltése akkor fejeződik be, ha a tele jel megérkezik. A tartályokat kézi szeleppel ürítik.

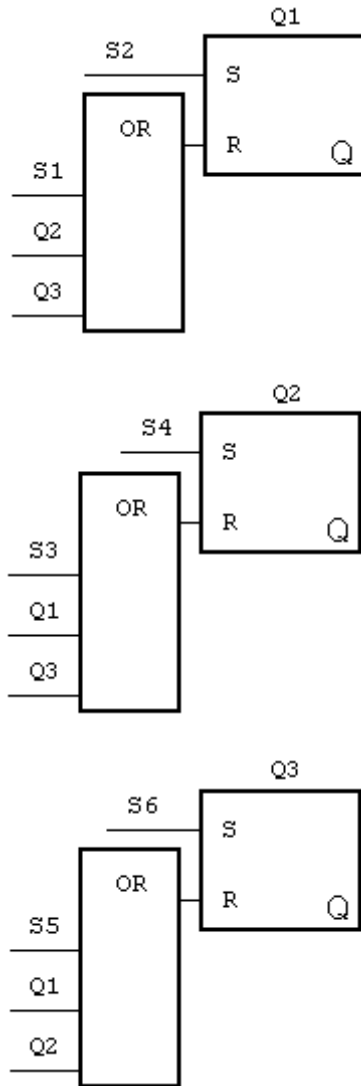


24. ábra Tárolótartályok feltöltésének vezérlése

### Összerendelési táblázat

Bemenetek	Jel	Logikai összerendelés	Cím
1. tartály tele	S1	A tartály tele, ha: S1=1	I 0.0
2. tartály tele	S3	A tartály tele, ha: S3=1	I 0.2
3. tartály tele	S5	A tartály tele, ha: S5=1	I 0.4
1. tartály üres	S2	A tartály üres, ha: S2=1	I 0.1
2. tartály üres	S4	A tartály üres, ha: S4=1	I 0.3
3. tartály üres	S6	A tartály üres, ha: S6=1	I 0.5
Kimenetek			
1. tartály mágnesszelep	Q1	A szelep nyitva, ha: Q1=1	Q0.0
2. tartály mágnesszelep	Q2	A szelep nyitva, ha: Q2=1	Q0.1
3. tartály mágnesszelep	Q3	A szelep nyitva, ha: Q3=1	Q0.2

**Funkcióterv**



**Utasításlista :**

```

PROGRAM PR3TART
VAR
    S1 AT %I0.0.0.0.0:
    BOOL;
    S2 AT %I0.0.0.0.1:
    BOOL;
    S3 AT %I0.0.0.0.2:
    BOOL;
    S4 AT %I0.0.0.0.3:
    BOOL;
    S5 AT %I0.0.0.0.4:
    BOOL;
    S6 AT %I0.0.0.0.5:
    BOOL;
    Q1 AT %Q0.0.0.0.0:
    BOOL;
    Q2 AT %Q0.0.0.0.1:
    BOOL;
    Q3 AT %Q0.0.0.0.2:
    BOOL;
END_VAR

    LD    S2
    S     Q1
    LD    S1
    OR    Q2
    OR    Q3
    R     Q1

    LD    S4
    S     Q2
    LD    S3
    OR    Q1
    OR    Q3
    R     Q2

    LD    S6
    S     Q3
    LD    S5
    OR    Q1
    OR    Q2
    R     Q3

END_PROGRAM
    
```

**Kérdések:**

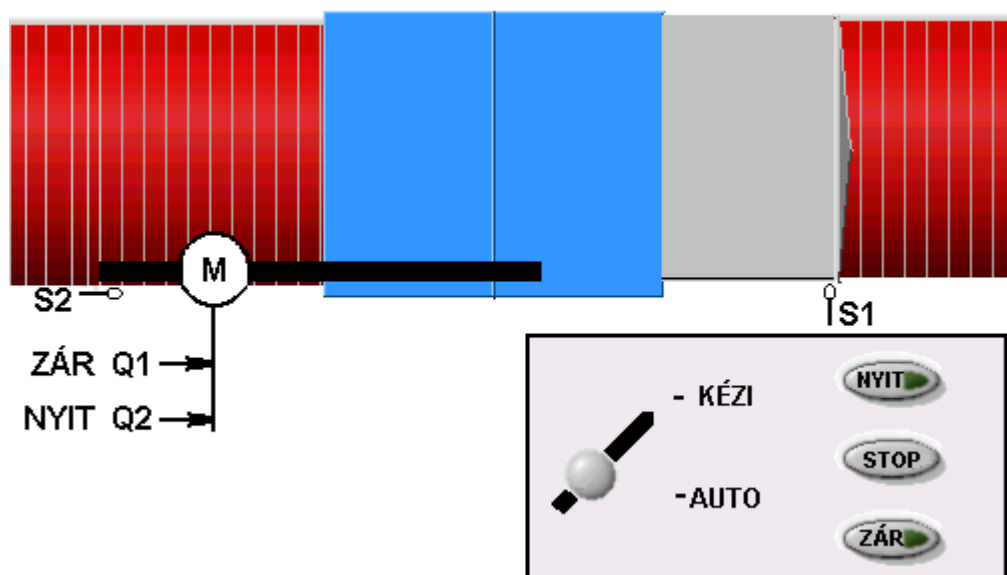
Ha egyszerre több tartály is üres jelzést ad, a fenti program milyen sorrendben fogja feltölteni őket?

Hogyan módosítaná a programot, ha az lenne a feladat, hogy a leürülés sorrendjében töltsse fel a tartályokat?

Hogyan módosítaná a programot, ha a start/stop jelet is figyelembe kellene vennie, azaz csak akkor ellenőrizze a szintjelzőket és működtesse a szelepeket, ha a START gombot benyomták?

### Gyakorló feladat: Gyárkapu vezérlése

Egy gyárkaput a kapusfülkéből elektromotorral működtetnek. Az elektromotort két teljesítménykapcsolóval lehet a nyitás illetve zárás irányba kapcsolni. **Q1**: balra, a kapu kinyílik. **Q2** jobbra, a kapu záródik. A két relét nem lehet egyidejűleg kapcsolni, kölcsönösen reteszelve egymást a kapcsolási oldalon is. A kapu véghelyzeteit végállás-kezelők (**S1**: a kapu zárva, **S2**: a kapu nyitva) jelzik.



25. ábra Gyárkapu vezérlése

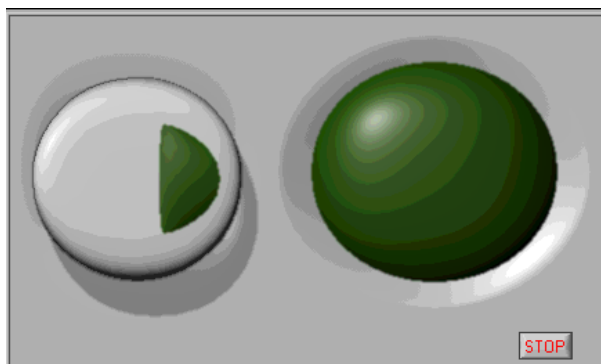
A kapusfülkében helyezték el a kapu kezelői pultját. A kaput kézi ill. automata üzemmódban lehet nyitni/zárni. A kívánt működés automata üzemmódban: a gomb rövid idejű benyomásával a kapu a véghelyzetig folyamatosan nyílik, illetve záródik. A művelet a STOP gomb benyomásával bármikor megszakítható. A vezérlést úgy kell megoldani, hogy ha a motor az egyik irányba működteti a kaput, a másik irányba átváltani csak a STOP benyomása után lehessen. Ha a kapu véghelyzetbe ér, a motor leáll. Kézi üzemmódban a motor addig nyitja vagy zárja a kaput, amíg a megfelelő gombot lenyomva tartják és a kapu még nem érte el a véghelyzetét.

### Összerendelési táblázat

Bemenetek	Jel	Logikai összerendelés	Cím
a kapu zárva	S1	jelez, ha :	S1=1 I0.0
a kapu nyitva	S2	jelez, ha :	S2=1 I0.1
AUT/KÉZI váltókapcsoló	A_K	AUTOMATA, ha :	A_K=1 I0.2
<b>NYIT nyomógomb</b>	NYIT	benyomva:	NYIT=1 I0.3
STOP nyomógomb	STOP	benyomva:	STOP=0 I0.4
ZÁR nyomógomb	ZAR	benyomva:	ZAR=1 I0.5
Kimenetek			
nyitás irányba kapcsoló relé	Q1	behúzza:	Q1=1 Q0.1
zárás irányba kapcsoló relé	Q2	behúzza:	Q2=1 Q0.2

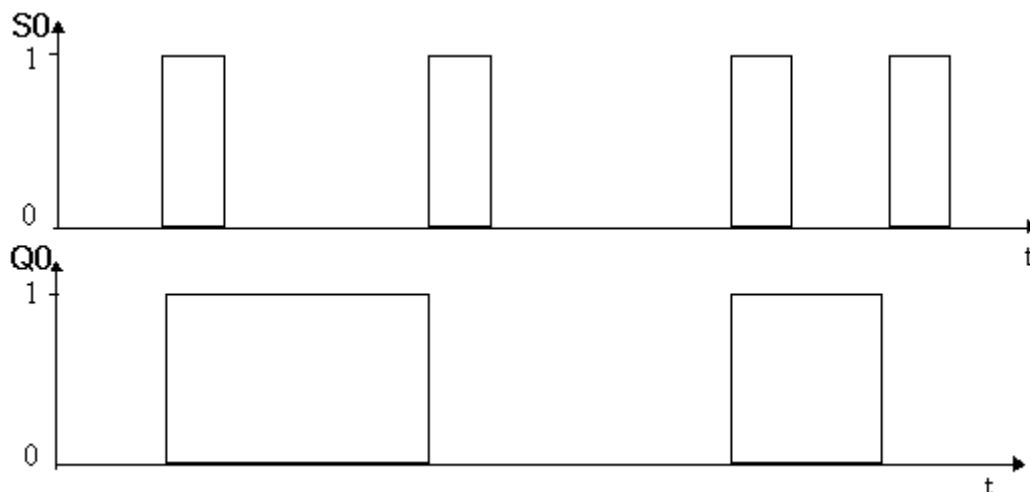
## Impulzuskapcsoló

Egy jelzőlámpa (Q0) az S0 nyomógomb (rövid idejű) megnyomására bekapcsol. Ha az S0 gombot ismételten megnyomják, a lámpa kialszik.



26. ábra A kívánt működést szimuláló program frontpanelképe

Idődiagram:



27. ábra Az impulzuskapcsoló idődiagramja

A bemeneti jelen fellépő emelkedő él (0-1 átmenet) a kimenet állapotváltozását okozza.

### Összerendelési táblázat

Bemenet	Jel	Logikai összerendelés	Cím
Nyomógomb	S0	benyomva: S0=1	I0.0
Kimenet			
Jelzőlámpa	Q0	világít: Q0=1	Q0.0

## Megoldás

### Utasításlista

```

PROGRAM NYGLAMPA
VAR
    S0 AT %I0.0: BOOL;
    Q0 AT %Q0.0:
    BOOL;
    M0:   BOOL;
    M1:   BOOL;
    M2:   BOOL;
END_VAR

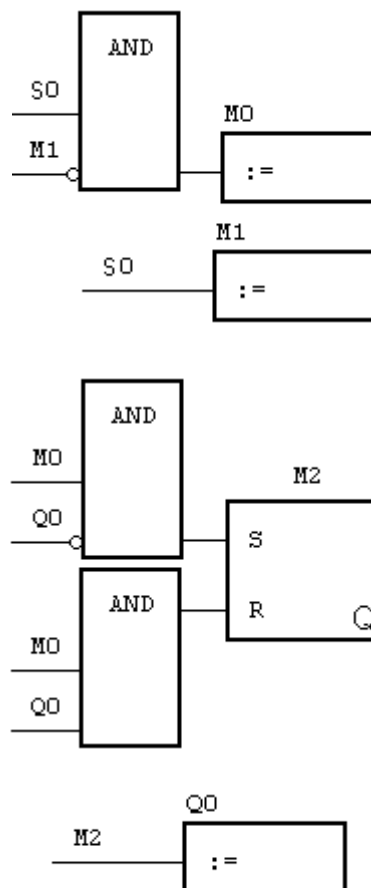
LD    S0
ANDNM1
ST    M0

LD    S0
ST    M1

LD    M0
ANDNQ0
S     M2
LD    M0
AND  Q0
R     M2

LD    M2
ST    Q0
END_PROGRAM
    
```

### Funkcióterv



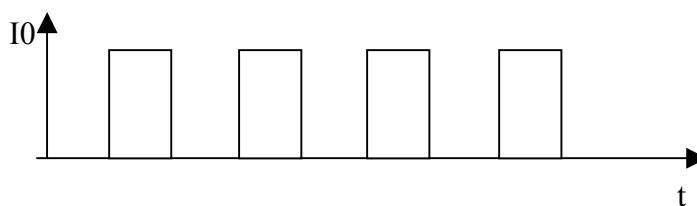
### Gyakorló feladat: utasításlista elemzése I.

**Feladat:** Írja át az alábbi utasításlistát funkciótervbe, majd próbálja meg elemezni a bemenőjel és a kimenőjel kapcsolatát!

```
FUNCTION_BLOCK FGVBLOKK
VAR_IN_OUT
  PAR1: BOOL;
END_VAR
LDN PAR1
ST PAR1
END_FUNCTION_BLOCK
```

```
PROGRAM ELEMZ1
VAR
  I0 AT %I0.0.0.0.0: BOOL;
  Q0 AT %Q0.0.0.0.0: BOOL;
  M0: BOOL;
  FGVB:FGVBLOKK;
END_VAR
LD I0
ANDNM0
CALC FGVB (PAR1:=Q0)
LD I0
ST M0
END_PROGRAM
```

A bemenőjel időbeli változása:



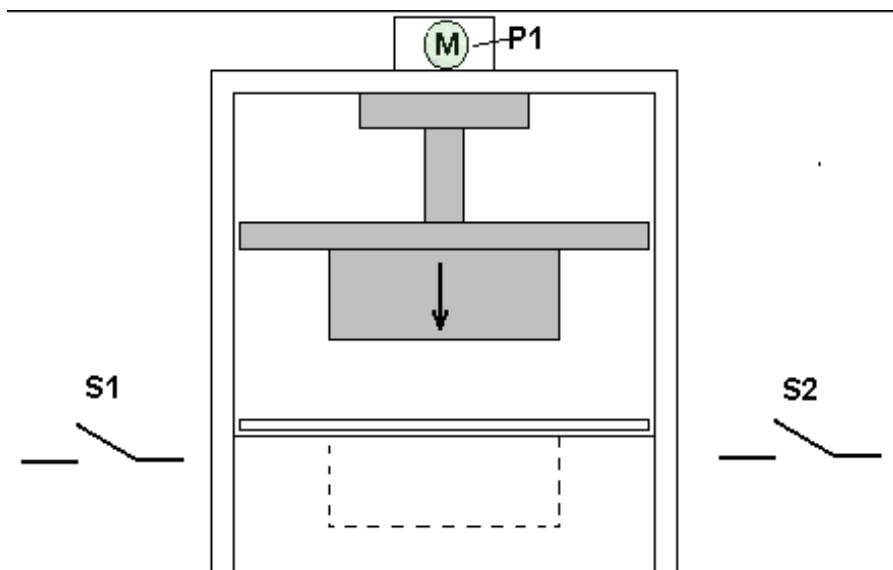
A kimenőjel időbeli alakulása a bemenőjel függvényében (megoldandó feladat!):



## Követővezérlés impulzus időzítővel

### Kétkezes reteszelés

A balesetveszély elkerülése végett egy présgép működtetését az ún. „kétkezes reteszeléssel” kell biztosítani. A prés csak akkor engedhető le, ha a kezelő az **S1** és **S2** nyomógombot adott időn belül (0,1s) egyszerre nyomja le. A két nyomógombot egymástól megfelelő távolságra kell elhelyezni. Nem engedélyezhető a présművelet, ha az egyik vagy a másik nyomógomb folyamatosan be van nyomva. (Pl.: kitámasztják). Ugyanígy, az excenter feletti nyomás azonnal megszűnik, ha abbahagyják a nyomógombok működtetését. Egy préselési művelet után a prés a kiindulási (felső) helyzetbe kerül és ott is marad, csak a két nyomógomb újbóli, 0,1s-on belüli lenyomása eredményez újabb műveletet.



28. ábra Stancológép

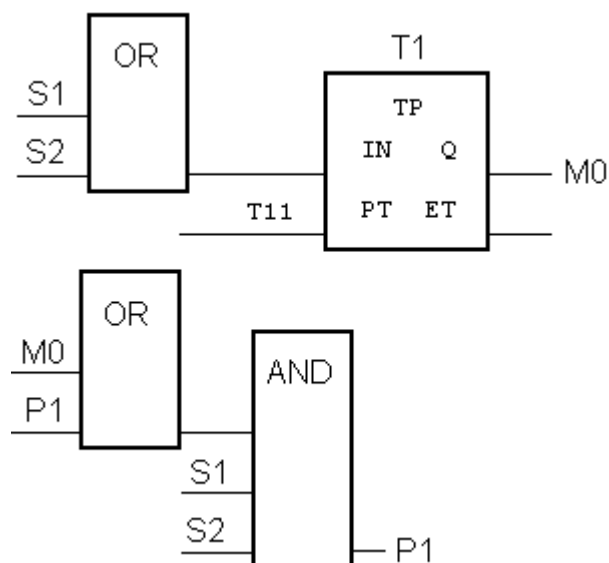
### Összerendelési táblázat

Bemenetek	Jel	Logikai összerendelés	Cím
Baloldali nyomógomb	S1	benyomva: S1=1	I0.1
Jobboldali nyomógomb	S2	benyomva: S2=1	I0.2
Kimenet			
Prés	P1	működtetve: P1=1	Q0.1

### A szűkített függvénytáblázat

P1előző értéke	T1 időzítő	S1	S2	P1
0	1	1	1	1
1	0	1	1	1
1	1	1	1	1
<b>minden egyéb estben</b>				<b>0</b>

## Funkcióterv



## A program utasításlistája

PROGRAM ketkret

VAR

S1 AT %I0.1 : BOOL ;  
 S2 AT %I0.2 : BOOL ;  
 P1 AT %Q0.1 : BOOL ;

END\_VAR

VAR

T1 : TP ;  
 M0 : BOOL ;

END\_VAR

VAR CONSTANT

T11 : TIME := T#0.1S ;

END\_VAR

LD S1  
 OR S2  
 ST T1.IN

LD T11  
 ST T1.PT

CAL T1

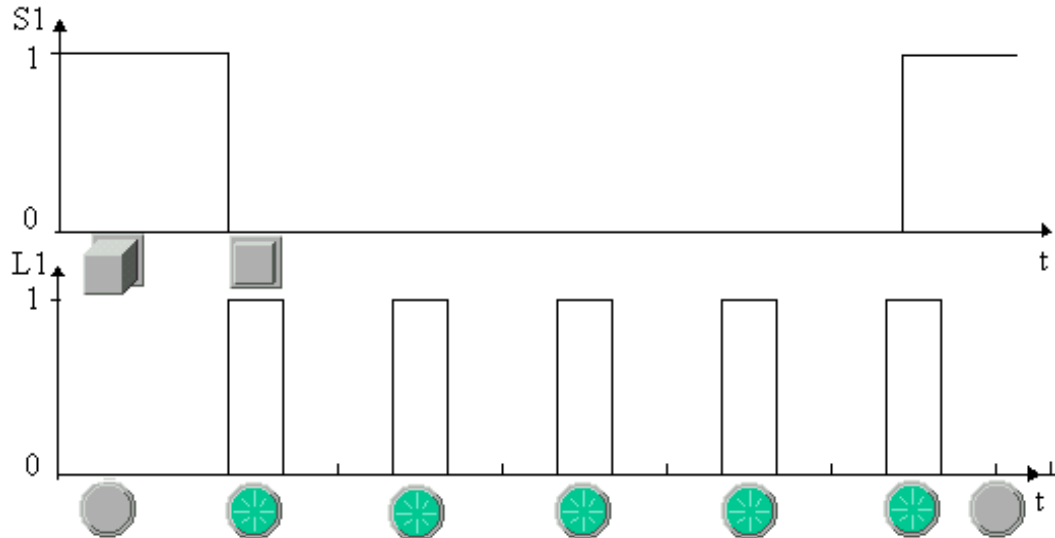
LD T1.Q  
 ST M0

LD M0  
 OR P1  
 AND S1  
 AND S2  
 ST P1

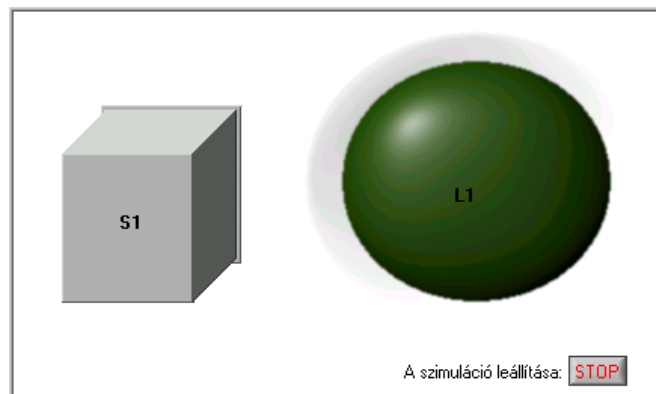
END\_PROGRAM

## Vészjelzés

Készítendő 1 Hz frekvenciájú vészjelzés, amely egy **S1** kapcsoló működtetésére a kimeneten (**L1** jelzőlámpa) azonnal „1”-jellel indul, az impulzus:szünet arány 1:2. Ha a kapcsolót átkapcsolják, az utolsó teljes ütemciklus befejeztével megszakad az ütemgenerálás.



29. ábra Idődiagram



30. ábra A kívánt működést szimuláló program frontpanelképe

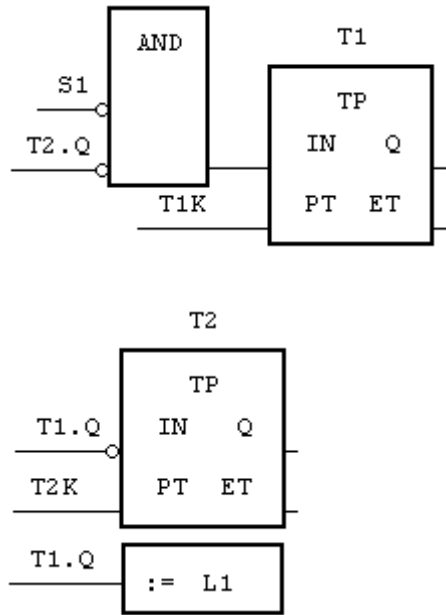
## Összerendelési táblázat

Bemenet	Jel	Logikai összerendelés	Cím
Nyomógomb	S1	benyomva: S1=0	I0.0
Kimenet			
Jelzőlámpa	L1	világít, ha: L1=1	Q0.0

A megoldáshoz két db impulzus időzítő (**T1**, **T2**) szükséges, amelyek felváltva működnek. Az egyik időzítő kétállapotú kimenetének **1**→**0** jelvéltása indítja a másik időzítőt.

A **T1** időzítő bináris kimenete megegyezik az ütemgenerátor **L1** kimenetével.

## Funkcióterv



## Utasításlista

```

ROGRAM PRVESZJ
VAR
    VESZJEL AT %I0.0.0.0.0:
    BOOL;
    LAMPA AT %Q0.0.0.0.0:
    BOOL;
    FGVBL:    VESZJ;
END_VAR
CAL FGVBL(S1:=VESZJEL)
LD  FGVBL.L1
ST  LAMPA
END_PROGRAM

FUNCTION_BLOCK VESZJ

VAR_INPUT
    S1:    BOOL;

END_VAR

VAR_OUTPUT
    L1:    BOOL;
END_VAR

VAR
    T1:    TP;
    T2:    TP;
    T1K:   TIME := t#0.33S;
    T2K:   TIME := t#0.66S;
END_VAR
    
```

```

LDN  S1
ANDN T2.Q
ST   T1.IN
LD   T1K
ST   T1.PT
CAL  T1
LDN  T1.Q
ST   T2.IN
LD   T2K
ST   T2.PT
CAL  T2
LD   T1.Q
ST   L1
END_FUNCTION_BLOCK
    
```

### Gyakorló feladat: utasításlista elemzése II.

Írja át az alábbi utasításlistát funkciótervbe, majd próbálja meg elemezni a bemenőjel és a kimenőjel kapcsolatát, ha a bemenőjel 1-ről 0-ra vált, és ott is marad!

```
PROGRAM ELEMZ2
VAR
    I0 AT %I0.0.0.0.0:  BOOL;
    Q0 AT %Q0.0.0.0.0:  BOOL;
    M1:  BOOL;
    M2:  BOOL;
    T1:  TON;
END_VAR
LD  I0
ORN M1
ST  T1.IN
LD  t#1s
ST  T1.PT
CAL T1
LD  T1.Q
ST  M1

LDN I0
AND M1
S   M2
LD  M1
AND Q0
R   M2
LD  M2
ST  Q0
END_PROGRAM
```

A bemenőjel időbeli változása:



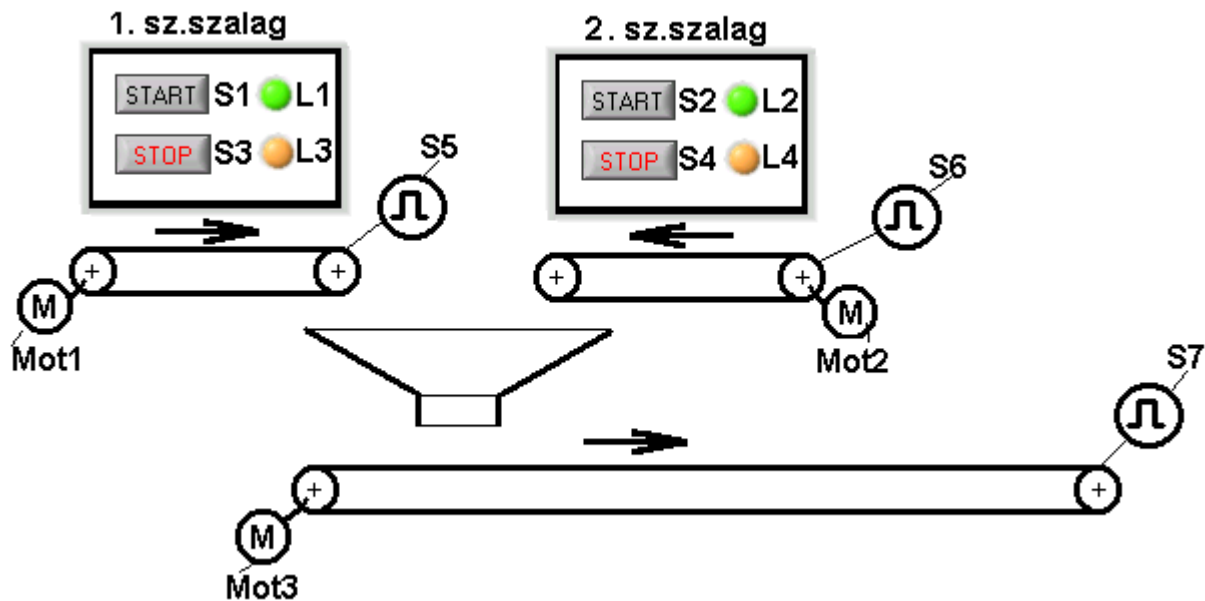
A kimenőjel időbeli alakulása a bemenőjel függvényében (megoldandó feladat!):



## Követővezérlés időzítőekkel

### Szállítószalagok együttes vezérlése

A kikapcsolás-késleltetési időzítő tipikus alkalmazására láthatunk példát a következő feladat megoldásában, ahol a szilárd anyag feltorlódását elkerülendő, a szállítószalagokat a kikapcsolási jel után még adott ideig működtetjük, hogy leürüljenek. A vezérlőalgorithmus ún. heurisztikus megoldású, és már meglehetősen bonyolult. Áttekintése, és így esetleges módosítása sem olyan egyszerű.



31. ábra Szállítószalagok vezérlése

A vezérlési feladat a szállítószalagok motorjainak működtetése az alábbi feltételek szerint:

- Az 1. és 2. szállítószalagok kézi nyomógombokkal kapcsolhatók be/ki (S1, S2, S3, S4).
- Az üzemállapotokat jelzőlámpákkal kell visszajelezni (L1, L2, L3, L4).
- Az 1. és 2. szállítószalag nem működhet egyidejűleg.
- A 3. szállítószalagnak mindig működni kell, ha az 1-t vagy a 2-t elindították.
- Ha az 1. vagy a 2. szállítószalagot a megfelelő STOP gombbal kikapcsolják, a szalagok még 2s-ig futnak, hogy a rajtuk lévő anyag leürülhessen. Ugyanezen okból a 3. szállítószalag a STOP benyomása után még 6s-ig fut.
- Az S5, S6, S7 felügyelők 10 Hz-es impulzusjellel jelzik a szalagok működését (forgás). Ha az impulzusjel megszakad, a jeladó kimenete folyamatosan 0 (hamis). Az indítás után 3s-ig a felügyelők jeleit nem kell kiértékelni. (Felfutási idő.)
- Ha az 1. vagy 2. szállítószalag jeladójának jele megszakad, a szállítószalag motorját azonnal ki kell kapcsolni, a 3. szállítószalagot pedig le kell üríteni, majd azt is le kell állítani. Eközben a Ki-jelzőlámpa (L3 vagy L4) 2 Hz frekvenciával villog.
- Ha a 3. szállítószalag jelzője ad folyamatos 0 jelet, minden motort azonnal le kell állítani, és be kell kapcsolni a hibajelzés villogását.

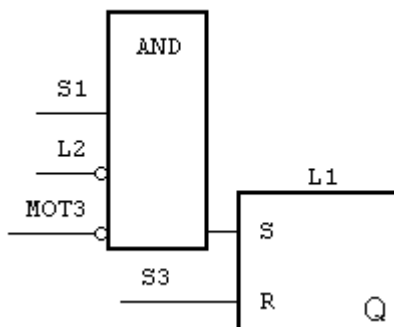
- A hibajelzést (villogást) a megfelelő szállítószalag **STOP** nyomógombjának megnyomásával lehet nyugtázni.

### Összerendelési táblázat

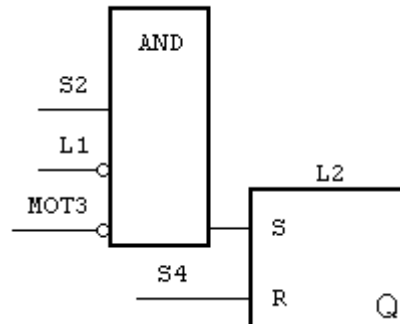
Bemenetek	Jel	Logikai összerendelés	Cím
1. sz.szalag BE-kapcsolás nyomógomb	S1	benyomva: S1=1	I0.0
2. sz.szalag BE-kapcsolás nyomógomb	S2	benyomva: S2=1	I0.1
1. sz.szalag KI-kapcsolás nyomógomb	S3	benyomva: S3=1	I0.2
2. sz.szalag KI-kapcsolás nyomógomb	S4	benyomva: S4=1	I0.3
1. sz.szalag fordulatjelző	S5	impulzus: S5=1	I0.4
2. sz.szalag fordulatjelző	S6	impulzus: S6=1	I0.5
3. sz.szalag fordulatjelző	S7	impulzus: S7=1	I0.6
Kimenetek			
1. sz.szalag működtetést jelző lámpa	L1	világít, ha: L1=1	Q0.0
2. sz.szalag működtetést jelző lámpa	L2	világít, ha: L2=1	Q0.1
Jelzőlámpa: 1. sz.szalag kikapcsolva	L3	világít, ha: L3=1	Q0.2
Jelzőlámpa: 2. sz.szalag kikapcsolva	L4	világít, ha: L4=1	Q0.3
1. sz.szalag motor	MOT1	működtetve: Mot1=1	Q0.4
2. sz.szalag motor	MOT2	működtetve: Mot2=1	Q0.5
3. sz.szalag motor	MOT3	működtetve: Mot3=1	Q0.6

### Funkcióterv

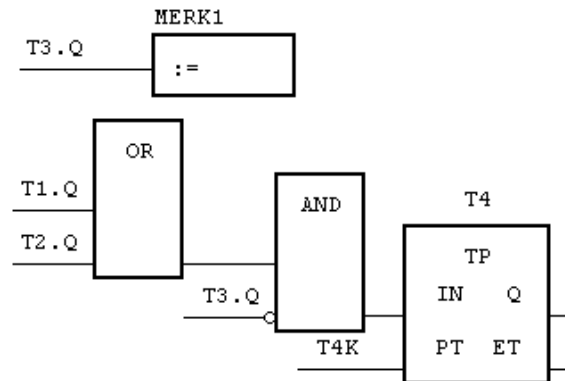
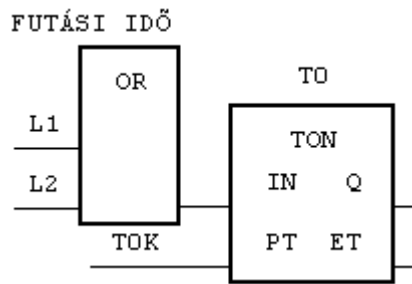
1.SZ.SZALAG BE-LÁMPA



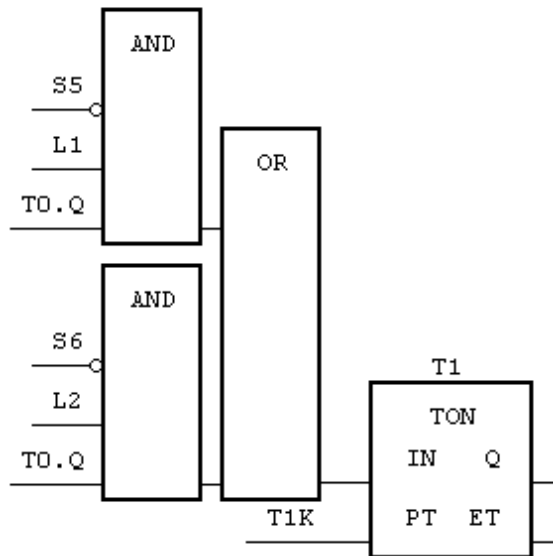
2.SZ.SZALAG BE-LÁMPA



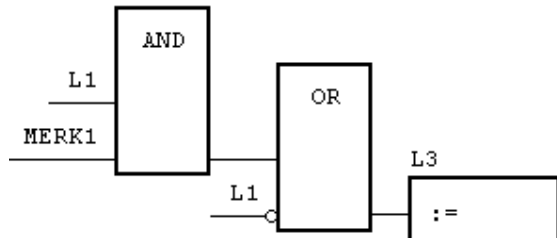
A rugalmasabb adatmódosítás biztosításának érdekében az időzítők időállandóit a deklarációs részben rögzítettük, a funkciótervben is a változóneveket tüntettük fel.



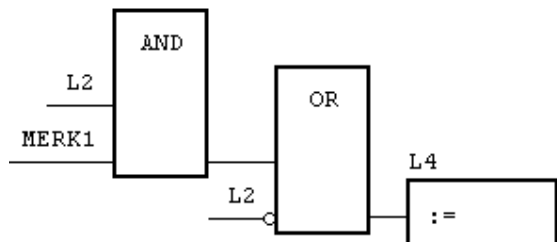
1.ÉS 2. SZ.SZALAG FUTÁSELLENŐRZÉS



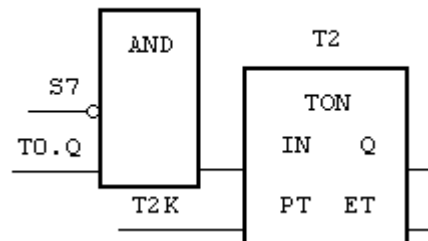
1.SZ.SZALAG KI-LÁMPA



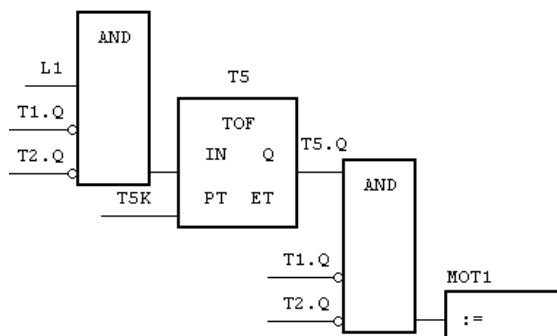
2.SZ.SZALAG KI-LÁMPA



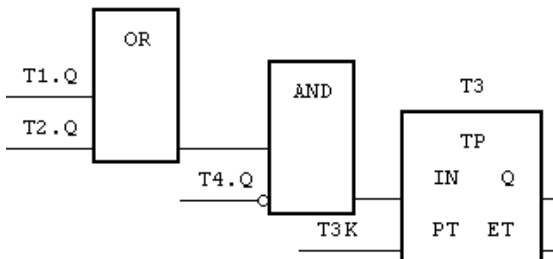
3.SZ.SZALAG FUTÁSELLENŐRZÉS:

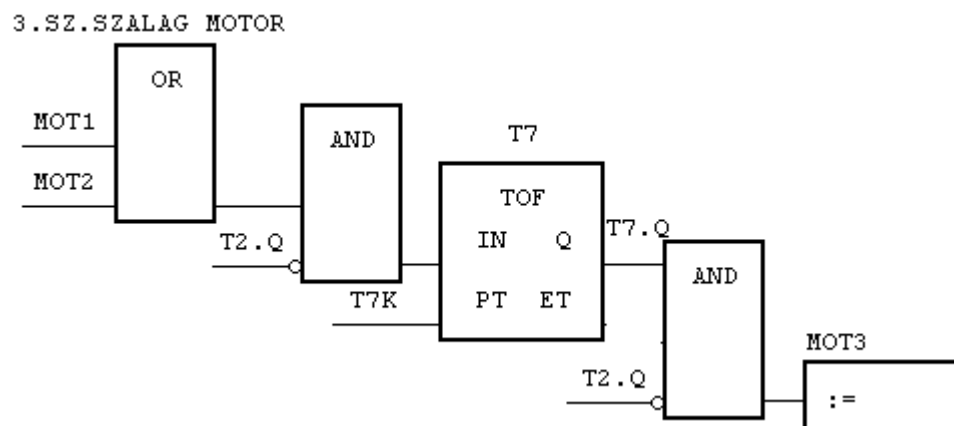
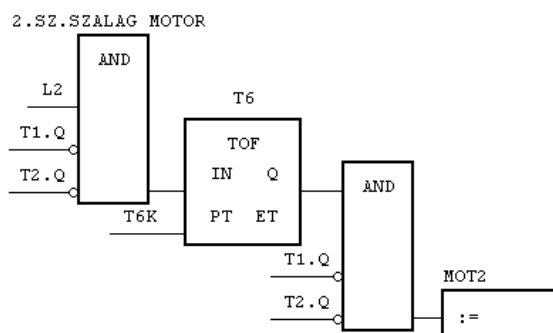


1.SZ.SZALAG MOTOR



2 Hz VILLOGÁS





### Utasításlista

#### PROGRAM SZSZALAG3

#### VAR

```

S1 AT %I0.0.0.0.0:  BOOL;
S2 AT %I0.0.0.0.1:  BOOL;
S3 AT %I0.0.0.0.2:  BOOL;
S4 AT %I0.0.0.0.3:  BOOL;
S5 AT %I0.0.0.0.4:  BOOL;
S6 AT %I0.0.0.0.5:  BOOL;
S7 AT %I0.0.0.0.6:  BOOL;
MOT1 AT %Q0.0.0.0.0:  BOOL;
MOT2 AT %Q0.0.0.0.1:  BOOL;
MOT3 AT %Q0.0.0.0.2:  BOOL;
L1 AT %Q0.0.0.0.3:  BOOL;
L2 AT %Q0.0.0.0.4:  BOOL;
L3 AT %Q0.0.0.0.5:  BOOL;
L4 AT %Q0.0.0.0.6:  BOOL;
T0:  TON;
T0K:  TIME := t#3s;
T1:  TON;
T1K:  TIME := t#120ms;
T2:  TON;
T2K:  TIME := t#120ms;
T3:  TP;
T3K:  TIME := t#250ms;
T4:  TP;
    
```

```

T4K: TIME := t#250ms;
MERK1:   BOOL;
T5:     TOF;
T5K:    TIME := t#2s;
T6:     TOF;
T6K:    TIME := t#2s;
T7:     TOF;
T7K:    TIME := t#6s;
END_VAR

(*1.SZ.SZALAG BE-LÁMPA*)
LD  S1
ANDNL2
ANDNMOT3
S  L1
LD  S3
R  L1
(*2.SZ.SZALAG BE-LÁMPA*)
LD  S2
ANDNL1
ANDNMOT3
S  L2
LD  S4
R  L2
(*FUTÁSI IDŐ*)
LD  L1
OR  L2
ST  T0.IN
LD  T0K
ST  T0.PT
CAL  T0
(*1.ÉS 2. SZ.SZALAG
FUTÁSELLENŐRZÉS*)
LDN  S5
AND  L1
AND  T0.Q
OR(  S6
NOT
AND  L2
AND  T0.Q
)
ST  T1.IN
LD  T1K
ST  T1.PT
CAL  T1
(*3.SZ.SZALAG
FUTÁSELLENŐRZÉS*)
LDN  S7
AND  T0.Q
ST  T2.IN

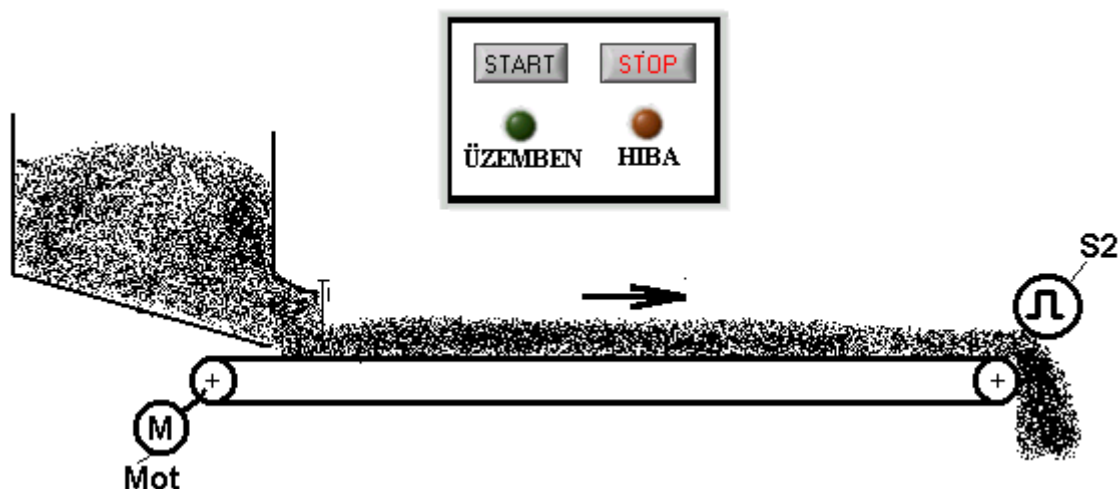
LD  T2K
ST  T2.PT
CAL  T2
(*2 Hz VILLOGÁS*)
LD  T1.Q
OR  T2.Q
ANDNT4.Q
ST  T3.IN
LD  T3K
ST  T3.PT
CAL  T3
LD  T3.Q
ST  MERK1
LD  T1.Q
OR  T2.Q
ANDNT3.Q
ST  T4.IN
LD  T4K
ST  T4.PT
CAL  T4
(*1.SZ.SZALAG KI-LÁMPA*)
LD  L1
AND  MERK1
ORN  L1
ST  L3
(*2.SZ.SZALAG KI-LÁMPA*)
LD  L2
AND  MERK1
ORN  L2
ST  L4
(*1.SZ.SZALAG MOTOR*)
LD  L1
ANDNT1.Q
ANDNT2.Q
ST  T5.IN
LD  T5K
ST  T5.PT
CAL  T5

LD  T5.Q
ANDNT1.Q

```

```
ANDNT2.Q
ST  MOT1
(*2.SZ.SZALAG MOTOR*)
LD  L2
ANDNT1.Q
ANDNT2.Q
ST  T6.IN
LD  T6K
ST  T6.PT
CAL T6
LD  T6.Q
ANDNT1.Q
ANDNT2.Q
ST  MOT2
(*3.SZ.SZALAG MOTOR*)
LD  MOT1
OR  MOT2
ANDNT2.Q
ST  T7.IN
LD  T7K
ST  T7.PT
CAL T7
LD  T7.Q
ANDNT2.Q
ST  MOT3
END_PROGRAM
```

**Gyakorló feladat: Szállítószalag vezérlése**



**32. ábra Szállítószalag motor vezérlése**

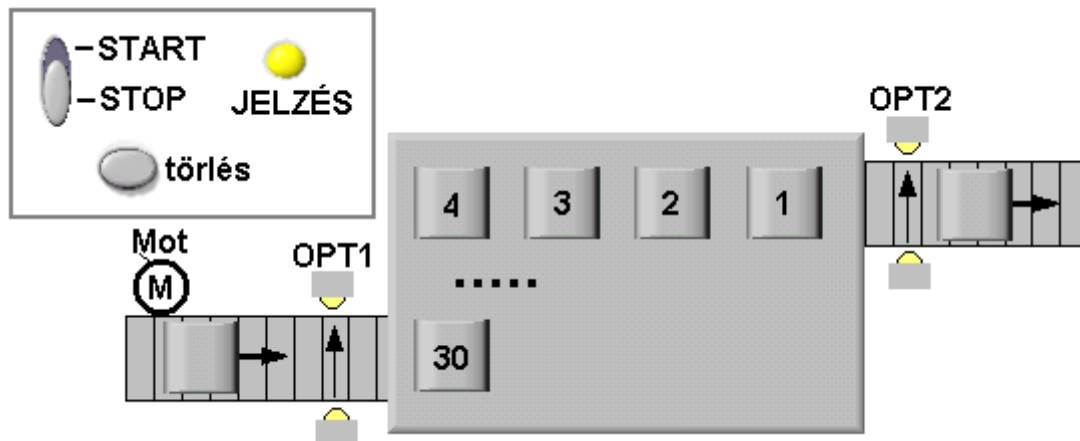
A szállítószalagot a **Mot** jelzésű motor működteti. Amíg a szalag megfelelően fut, az **S2** jele 10 Hz frekvenciajel. Probléma esetén (pl. szalagszakadás stb.) az **S2** jeladó folyamatosan 0 értéket ad. Ha bekapcsolt motor mellett nem jelentkezik az impulzusjel, le kell állítani a motort, és a **HIBA** jelzőlámpa 2Hz-es frekvenciával villog. A szalag indítása a **START** nyomógombbal történik, leállítása illetve a hibajel nyugtázása **STOP** nyomógombbal lehetséges. Üzem közben az **ÜZEMBEN** jel folyamatosan világít. Indítás után 5s-ig nem kell figyelembe venni **S2** jelét.

**Összerendelési táblázat**

Bemenetek	Jel	Logikai összerendelés	Cím
Sz.szalag BE-kapcsolás ny.gomb	SART	benyomva: SART=1	I0.0
Sz.szalag KI-kapcsolás ny.gomb	STOP	benyomva: STOP=0	I0.1
Sz.szalag futásjelző	S2	impulzus: S2=1	I0.2
Kimenetek			
Szállítószalag motor	Mot	működtetve: Mot=1	Q0.0
ÜZEMBEN jelzőlámpa	UZEMBEN	világít, ha: UZEMBEN =1	Q0.1
HIBA jelzőlámpa	HIBA	világít, ha: HIBA =1	Q0.2

### Munkadarabok átmeneti tárolása

Egy szerelési útvonalon a munkadarabok feltorlódásának elkerülésére átmeneti tároló asztalt építenek be. A munkadarabok beérkezését és kiadását optikai érzékelők jelzik, melyek impulzusait egy számlálóba vezetjük. Ha a tárolóban lévő munkadarabok száma eléri a maximumot (30 db), le kell állítani a bejövő szalag továbbító motorját. Ha a munkadarabok száma a tárolóban 10 alá csökken, (alsó határérték), a vezérlés bekapcsolja a jelzőlámpát. A számláló törlése üzemenkzdetkor, üres tároló mellett, a törlőgomb benyomásával lehetséges.

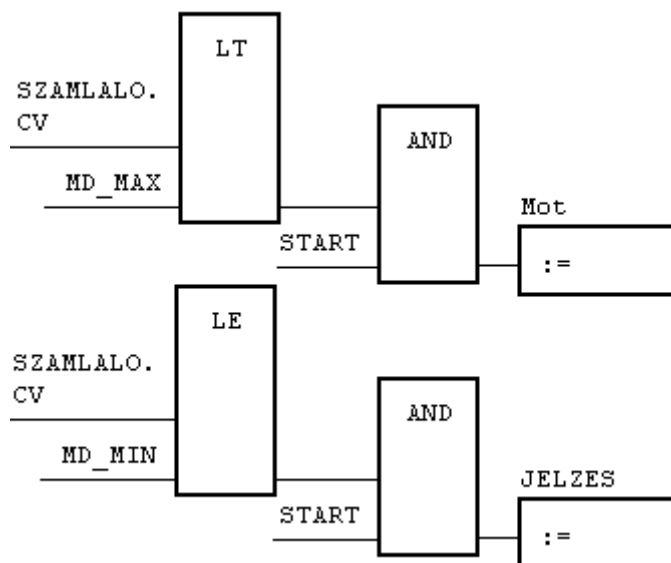
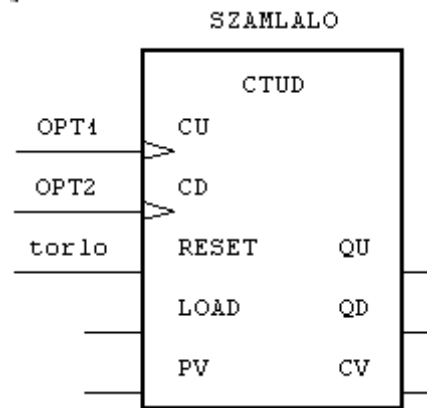


33. ábra Átmeneti munkadarab-tároló

### Összerendelési táblázat

Bemenetek	Jel	Logikai összerendelés	Cím
BE/KI kapcsoló	START	bekapcsolva: START=1	I0.0
Törlőgomb	torlo	benyomva: torlo=1	I0.1
Belépés optikai érzékelője	OPT1	jelez, ha: OPT1=1	I0.2
Kilépés optikai érzékelője	OPT2	jelez, ha: OPT2=1	I0.3
Kimenetek			
Szállítószalag motorja	Mot	működtetve, ha: Mot=1	Q0.0
Jelzőlámpa	JELZES	világít, ha: JELZES =1	Q0.1

## Funkcióterv



## Utasításlista

```
PROGRAM mdtarol
VAR
START AT %I0.0.0.0:    BOOL;
torlo AT %I0.0.0.1:   BOOL;
OPT1 AT %I0.0.0.2:    BOOL;
OPT2 AT %I0.0.0.3:    BOOL;
Mot AT %Q0.0.0.0:     BOOL;
JELZES AT %Q0.0.0.1:  BOOL;
SZAMLALO: CTUD;
MD_MAX:  INT := 30;
MD_MIN:  INT := 10;
END_VAR
```

(\*SZÁMLÁLÓ\*)

```
LD  OPT1
ST  SZAMLALO.CU
LD  OPT2
ST  SZAMLALO.CD
LD  torlo
ST  SZAMLALO.RESET
CAL  SZAMLALO
```

(\*ÖSSZEHASONLÍTÁS <30\*)

```
LD  SZAMLALO.CV
LT  MD_MAX
AND  START
ST  Mot
```

(\*ÖSSZEHASONLÍTÁS <=10\*)

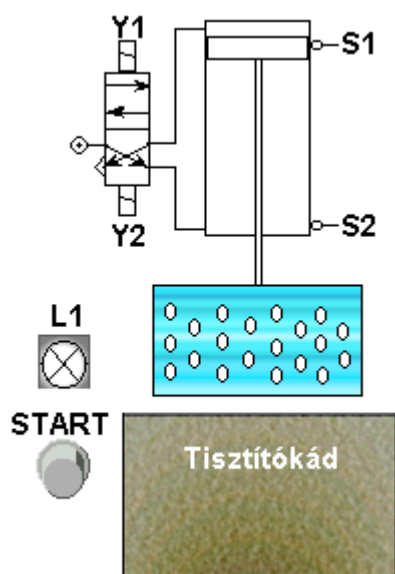
```
LD  SZAMLALO.CV
LE  MD_MIN
AND  START
ST  JELZES
```

END\_PROGRAM

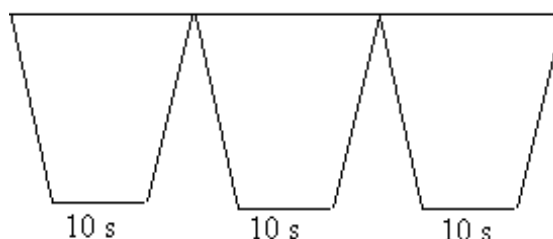
### Tisztítóberendezés elektro-pneumatikus vezérlése

Egy tisztítóberendezés tartóeleme (kosár) pneumatikus munkahenger segítségével engedhető le a tisztítóoldatba és emelhető fel csepegtetési ill. cserélési állapotba. A munkahengerre a működtető levegőt a 4/2 utas elektromágneses szelep segítségével kapcsoljuk a megfelelő irányba.

A feladat: háromszori leengedés és felemelés után a kiindulási helyzetbe kell vinni a dugattyút. Eközben mindig 10 s-ig a tisztítóoldatban kell maradnia a tartókosárnak. A tisztítóciklus **START** nyomógomb megnyomásával indítható. Az **L1** lámpa a tisztítási ciklus alatt folyamatosan világít.



34. ábra Tisztítóberendezés



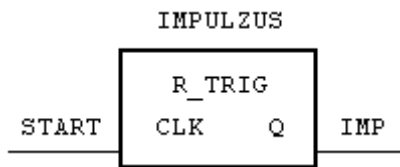
35. ábra A tisztítási ciklus idődiagramja

### Összerendelési táblázat

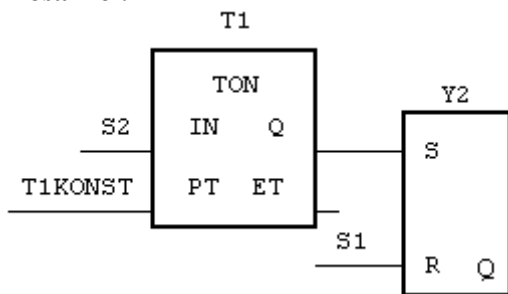
Bemenetek	Jel	Logikai összerendelés	Cím
BE/KI kapcsoló	START	benyomva: START=1	I0.0
felső végálláskapcsoló	S1	jelez, ha: S1=1	I0.1
alsó végálláskapcsoló	S2	jelez, ha: S2=1	I0.2
Kimenetek			
Munkahenger le	Y1	működtetve: Y1=1	Q0.0
Munkahenger fel	Y2	működtetve: Y2=1	Q0.1
Lámpa	L1	világít, ha: L1=1	Q0.2

## Funkcióterv

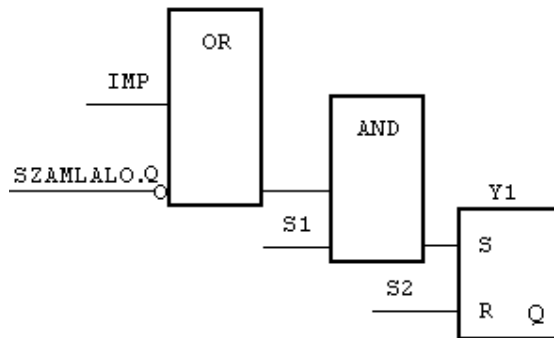
### Bekapcsolási impulzus:



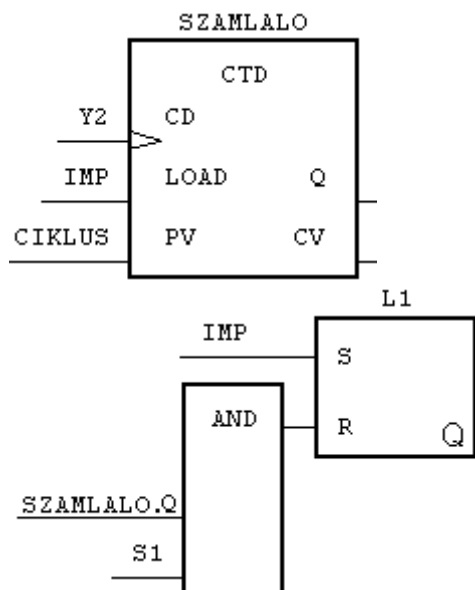
### Kosár fel:



### Kosár le:



### A számláló és a működést jelző lámpa:



## Utasításlista

PROGRAM TISZTIT

VAR

```
START AT %I0.0.0.0.0:
  BOOL;
S1 AT %I0.0.0.0.1:   BOOL;
S2 AT %I0.0.0.0.2:   BOOL;
Y1 AT %Q0.0.0.0.0:   BOOL;
Y2 AT %Q0.0.0.0.1:   BOOL;
L1 AT %Q0.0.0.0.2:   BOOL;
T1:   TON;
T1KONST:   TIME := t#10s;
SZAMLALO:   CTD;
CIKLUS:   INT := 3;
IMPULZUS:   R_TRIG;
IMP:   BOOL;
```

END\_VAR

(\*SART IMPULZUS\*)

```
CAL IMPULZUS(CLK :=START|
  IMP := Q)
```

(\*KOSÁR FEL\*)

```
CAL T1(IN := S2,PT :=T1KONST)
```

```
LD T1.Q
```

```
S Y2
```

```
LD S1
```

```
R Y2
```

(\*KOSÁR LE\*)

```
LD IMP
```

```
ORN SZAMLALO.Q
```

```
AND S1
```

```
S Y1
```

```
LD S2
```

```
R Y1
```

(\*SZÁMLÁLÓ\*)

```
CAL SZAMLALO(
  CD := Y2,
  LOAD := IMP,
  PV := CIKLUS
)
```

```
LD IMP
```

```
S L1
```

```
LD SZAMLALO.Q
```

```
AND S1
```

```
R L1
```

```
END_PROGRAM
```

### Gyakorló feladat: utasításlista elemzése III.

**Feladat:** Írja át az alábbi utasításlistát funkciótervbe, majd próbálja meg elemezni a bemenőjel és a kimenőjel kapcsolatát, ha a bemenőjel 1-ről 0-ra vált, és ott is marad!

```
PROGRAM elemz3
VAR
BE AT %I0.0.0.0.0:BOOL;
KI AT %Q0.0.0.0.0:BOOL;
T1: TP;
T2: TP;
T3: TON;
C1: CTUD;
M0: BOOL;
M1: BOOL;
M2: BOOL;
M3: BOOL;
M4: BOOL;
END_VAR
VAR constant
T1K: TIME := T#1S;
T2K: TIME := T#1S;
T3K: TIME := T#11S;
END_VAR
LDN BE
AND M4
ST M0
LD BE
ST M4
LD M0
OR M3
ST C1.LOAD
LD 5
ST C1.PV
LD M1
ST C1.CD
CAL C1

LDN C1.QD
ANDNM2
ST T1.IN
LD T1K
ST T1.PT
CAL T1
LD T1.Q
ST M1
LDN M1
ANDNBE
ST T2.IN
LD T2K
```

```
ST T2.PT
CAL T2

LD T2.Q
ST M2
LD C1.QD
ANDNBE
ST T3.IN
LD T3K
ST T3.PT
CAL T3
LD T3.Q
ST M3
LD M1
ST KI
END_PROGRAM
```

A bemenőjel időbeli változása:



A kimenőjel időbeli alakulása a bemenőjel függvényében (megoldandó feladat!):



