PLC programozás az IEC 1131-3 szabvány szerint

Jancskárné Anweiler Ildikó
főiskolai docens
PTE – PMMFK
Műszaki Informatika Tanszék

Tartalomjegyzék

Az IEC-1131-3 szabvány

Az IEC-1131-3 szabvány (International Electronical Comission : 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 deklarálása

Az IEC-1131-3 szabvány az információk tárolására és feldolgozására *változókat* használ. Vannak olyan PLC-rendszerek, amelyekben a változókat *merker*eknek (német nyelterület) ill. *flag*eknek (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 grafikusan is megadható (be- és kimeneti paraméterek).

Példa egy tipikus változódeklarációra

```
VAR_INPUT                 (*bemeneti változó*)
        kapcsoló : BOOL;  (*bináris érték*)
END_VAR
VAR_OUTPUT                (*kimeneti változó*)
        fordszam : REAL;  (*valós érték*)
END_VAR
VAR_RETAIN                          (*lokális változó, elemmel pufferelt*)
        motorsz : INT;              (*előjeles egészszám*)
        Motornev : STRING[10];      (*karakterlánc*)
        Veszki : %IX2.0 : BOOL      (*a bemeneti periféria 2.0-s bitje*)
END_VAR
```

A programszervezési egység (POU) törzse

A programszervezési egység törzse a deklarációs részt követi, a PLC által végrehajtandó parancsok leírása, jellemzően az alább felsorolt valamelyik programozási nyelv szintaktikájának megfelelően.

Az IEC-1131-3 szabványban ajánlott programozási nyelvek

- *utasításlista*
  jellemzője:
  - gépközeli, akkumulátorra épülő, assembly típusú nyelv;
  - a német utasításlista nyelven alapul (Anweisungsliste: AWL);
  - soronként egy parancs a megengedett;
  - a legtöbb fejlesztői környezet biztosítja a használatát.

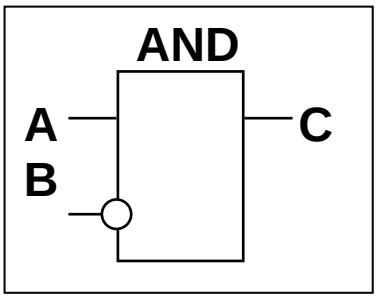
```
LD      A
ANDN    B
ST      C
```

- *létradiagram*
  jellemzője:
  - az észak-amerikai programozási stíluson alapul, az USA-ban szabványos; elektromos rajzjeleket használja;
  - standardizált relé-készlet és létraprogramozási szimbólumok.

```
 A   B              C
-| |--|/|----------( )
```

- *funkcióterv*
  jellemzője:
  - Európában elterjedten használt grafikus programozási nyelv;

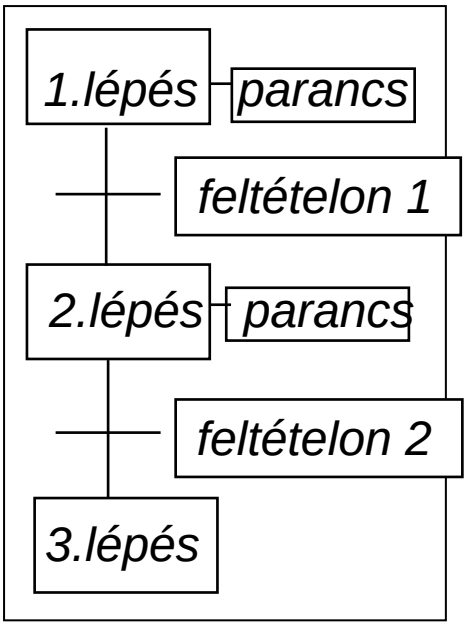
- a programelemek, mint blokkok összeköthetők, hasonlóan a logikai áramköri rajzokhoz;
- olyan alkalmazásokban használják, amelyek vezérlőkomponensek közötti adat vagy információáramlást tartalmaznak.



- *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 estén, milyen beavatkozás engedélyezhető a vezérelt folyamatban. Az IEC-1131-3 szabvány a vezérlő algoritmus strukturálásában hangsúlyozza a programtervezési technika jelentőségét.



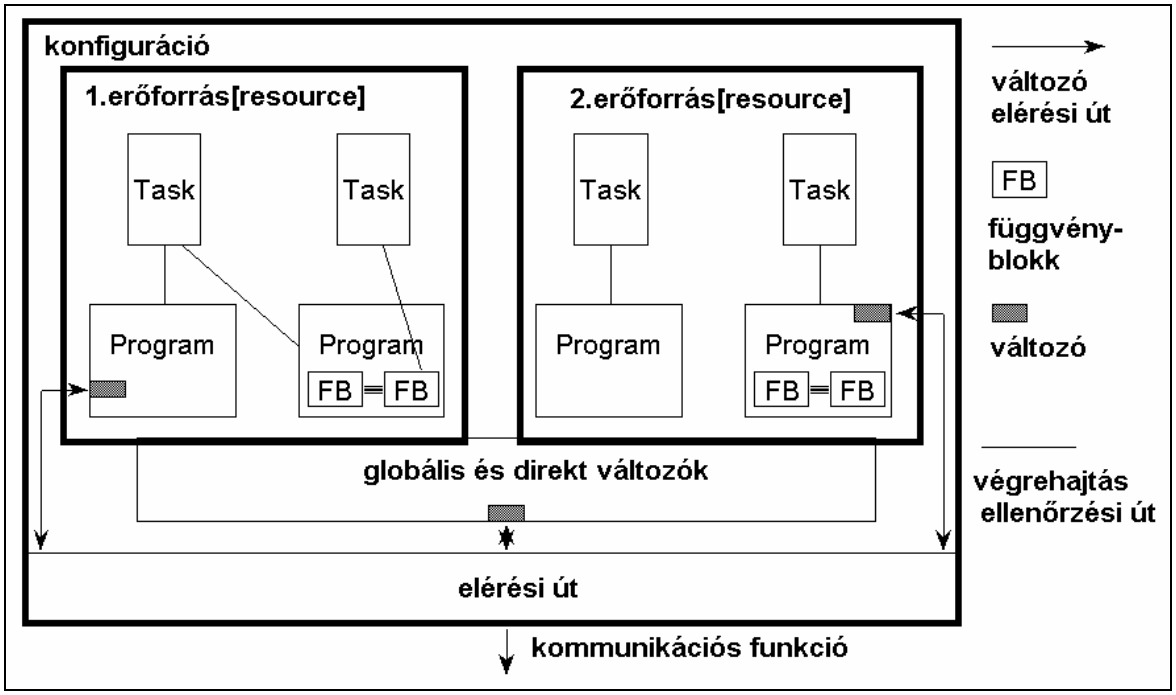
Az IEC programozói környezet

A legtöbb fejlesztőrendszer biztosítja az alábbi feltételeket:

- grafikus programozói felület;
- több ablakos rendszer;
- egérműveletek;
- legördülő menü;
- beépített hypertextes helpfunkció;
- szoftveres ellenőrzés a tervezés során.

Erőforrás elosztás

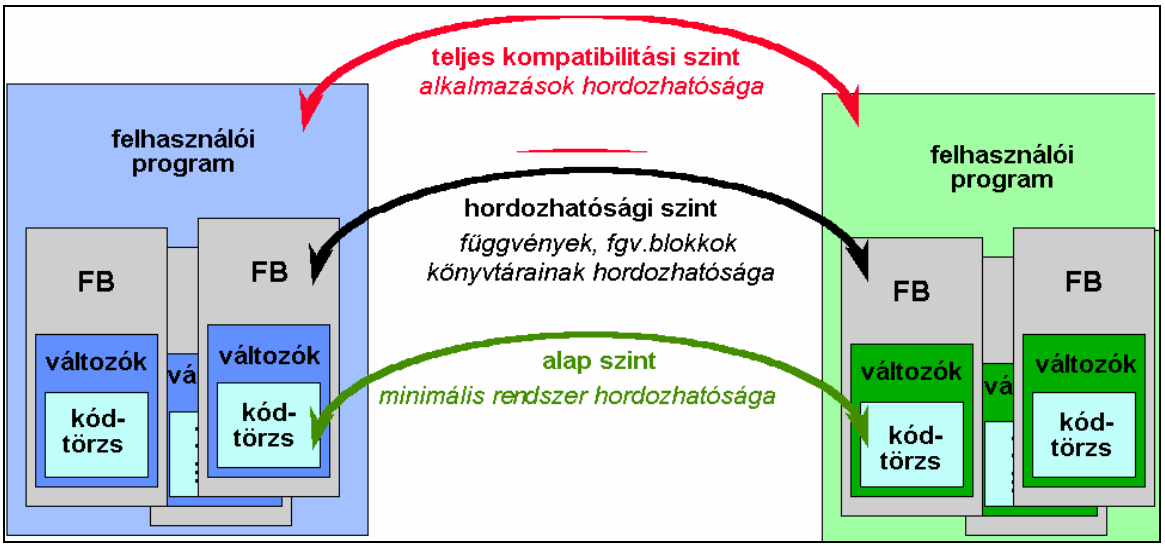
Egy általános vezérlő berendezés (PLC) több központi egységet (CPU-t) vagy speciális processzorokat tartalmazhat, amelyeket az IEC-1131-3 szabvány erőforrásoknak (*resources)* nevez. Egy erőforráson több taszk futhat, amelyek a prioritásuk, vagy a végrehajtás típusa (ciklikus, periodikus, interrupt) alapján különböznek egymástól. A programokat taszkokhoz rendeljük, ez eredményezi a futásidőbeli programot. Egy program több taszkhoz is hozzárendelhető (instancálás).



1. ábra Az IEC-1131-3 szabvány szerinti szoftvermodell

A PLC–projektet azokból a szervezési egységekből építhetjük fel, amelyeket a gyártó mellékelt ill. a felhasználó programozott. A felhasználói programokból könyvtár képezhető, amelynek tesztelt egységei más projektekbe is átmásolhatók.

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

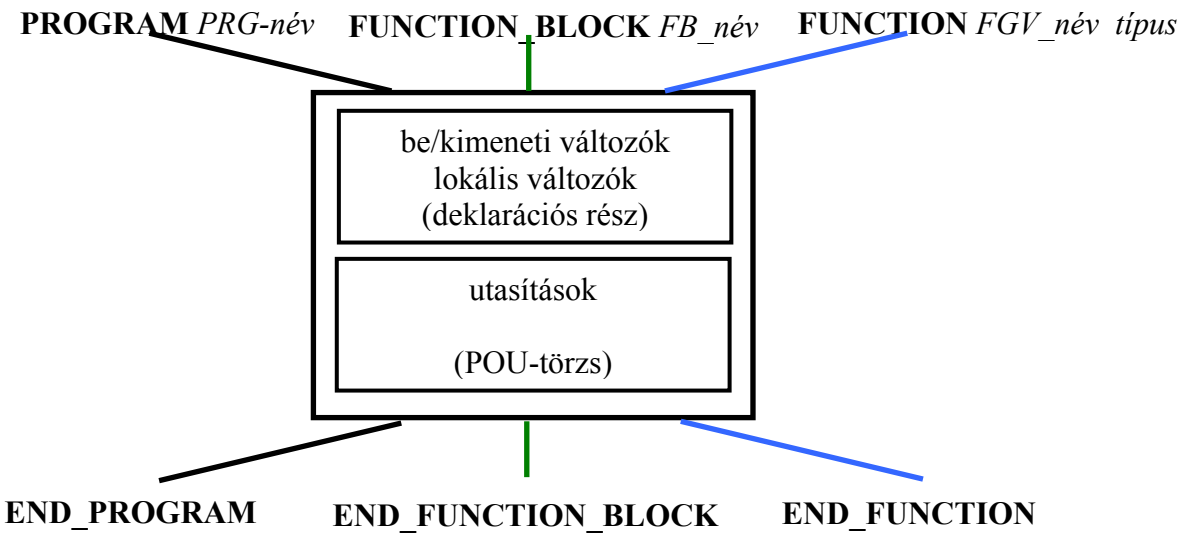
POU típus	kulcsszó	jelentés
program	PROGRAM	főprogram a PLC-perifériák hozzárendeléseivel, globális változókkal
függvényblokk	FUNCTION_BLOCK	építőelem be- és kimeneti változókkal, a leggyakrabban használt POU típus
függvény	FUNCTION	A PLC műveletek készletének kibővítésére szolgáló függvény

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 függnek 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

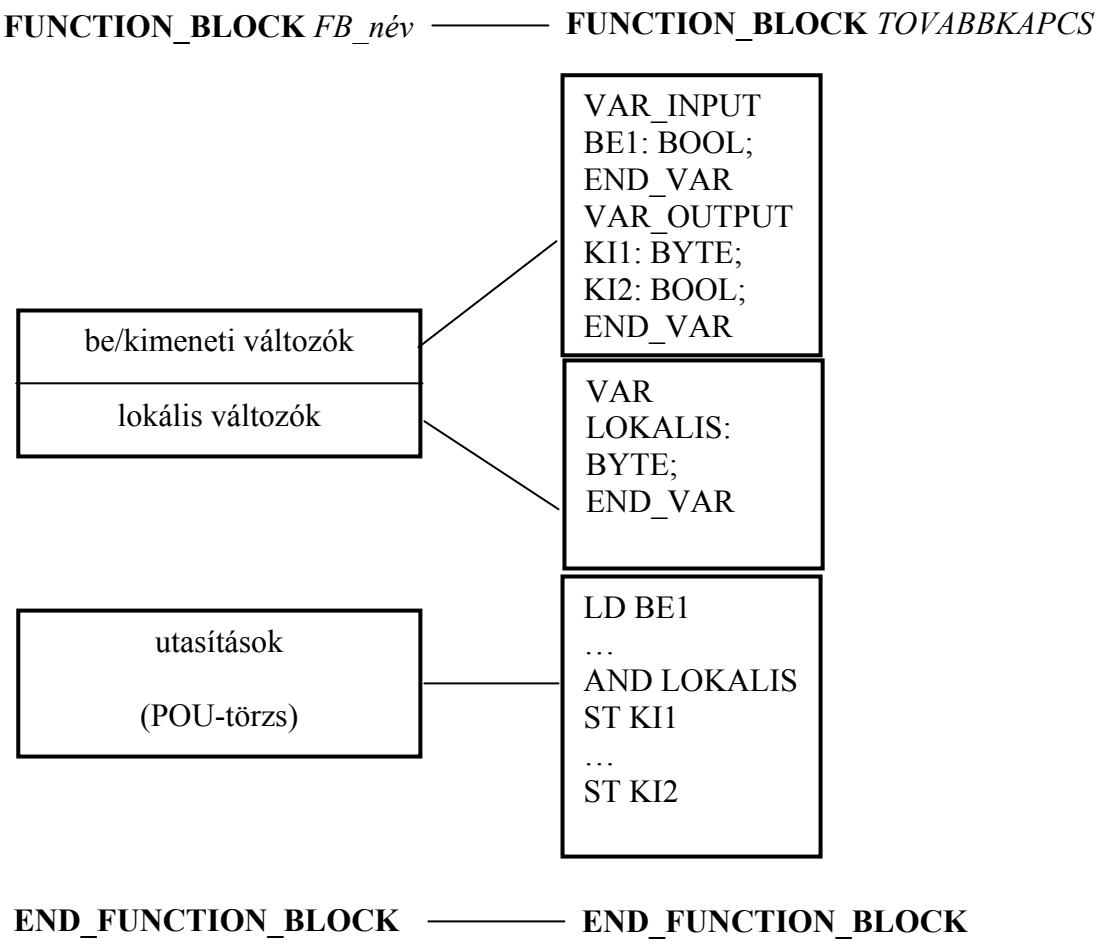
Egy POU az alábbi ábrán látható részekből épül(het) fel.

- 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)



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 változótípusoknak 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 értékként 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 adatcsere 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 *external* típusú változót egy másik POU-ban mint *global* változót deklarálták. 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 *global* változót a POU-n belül deklarálják és a külső POU-kban mint *external* 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 deklarálás 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 változókhoz 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 egészíthetik 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 változóit 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

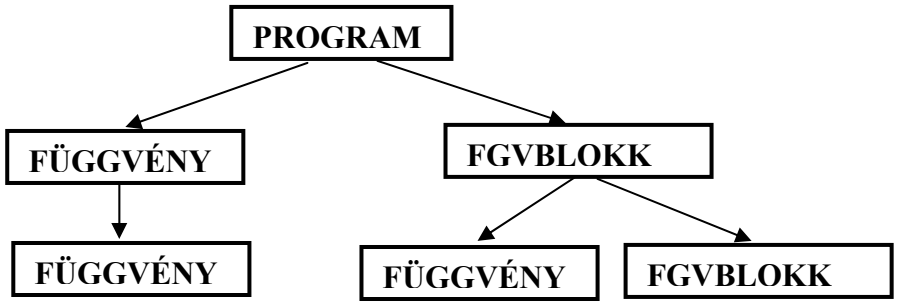
A függvényblokk és a függvény alprogramok, a PROGRAM főprogram. Múlti-taszkos operációs rendszerben egymással párhuzamosan több főprogram is futtatható.

A program sajátosságai:

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

Kis rendszereknél a program feladata az is, hogy külön konfigurációs fájl nélkül biztosítsa a PLC-perifériák változókhoz rendelését. A lehetőségek operációs rendszertől és kiépítettségtől (gyártótól) függőek. Azonos programot több taszkhoz 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 címkeként. 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 SRTING változókban 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
karaktersor	’’ ’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 karaktersorozat, maximális hossza implementációfüggő. Különböző programelemek, változók, címkék, származtatott adattípusok, konfigurációk, erőforrások azonosítására szolgáló felhasználó által adott nevek.

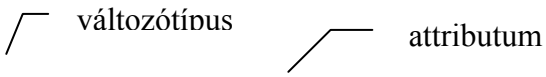
Példák felhasználói nevekre:

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

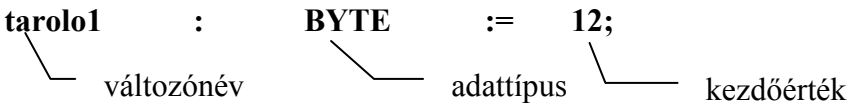
Változók és adattípusok

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

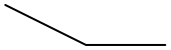
A változódeklaráció legfontosabb elemei



VAR_OUTPUT RETAIN



END_VAR



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 egészszám**	**előjel nélküli egész**	**valós**	**időpont, időtartam, dátum, karaktersor**
BOOL BYTE WORD DWORD LWORD	INT SINT DINT LINT	UINT USINT UDINT ULINT	REAL LREAL	TIME DATE TIME_OF_DAY DATE_OF_TIME 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	$[-2^{63},\ldots,+2^{63}-1]$	
USINT	unsign. short integer	8	[0,…,+255]	0
UINT	unsigned integer	16	[0,…,+65 535]	0
UDINT	unsign. double int.	32	$[0,\ldots,+2^{32}-1]$	0
ULINT	unsign. long integer	64	$[0,\ldots,+2^{64}-1]$	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 típusdefiniálást 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

<table>
<tr><th colspan="5">ANY</th></tr>
<tr><th rowspan="2">ANY_BIT</th><th colspan="3">ANY_NUM</th><th rowspan="2">ANY_DATE</th><th rowspan="2"></th></tr>
<tr><th colspan="2">ANY_INT</th><th>ANY_REAL</th></tr>
<tr><td>BOOL
BYTE
WORD
DWORD
LWORD</td><td>INT
SINT
DINT
LINT</td><td>UINT
USINT
UDINT
ULINT</td><td>REAL
LREAL</td><td>DATE
TIME_OF_DAY
DATE_OF_TIME</td><td>TIME
STRING</td></tr>
</table>

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, merkerek.) 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 bájtcím gépfüggő (Összesen mennyi be/kimenet ill. merker definiálható.) Gyakran előírás, hogy a szó csak páros bájtcímen kezdődhet. (Ne felejtsük el, hogy közvetlen címzésű változókat csak a főprogramban lehet deklarálni!)

Példa közvetlen címzésű változók deklarálására

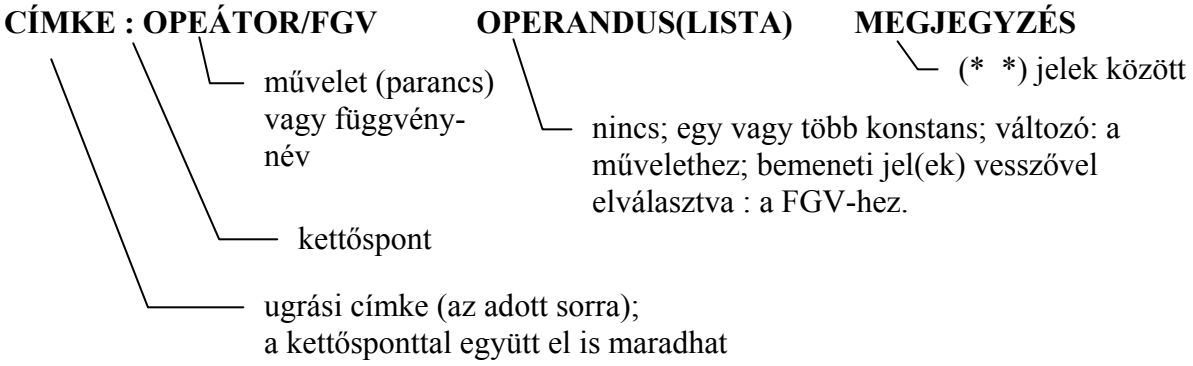
```
VAR
        (*közvetlen ábrázolású változók*)
        AT%IW6      : WORD;
        AT%QD4      : DINT;
        (*szimbolikus nevű, közvetlen címzésű változók*)
        INP_BYTE    AT%IB0;
        OUT_WORD AT%QW0;
END_VAR

…
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 kommentárkezdetként. 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 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 alapműveletet, 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	V vagy U	feltétel nélküli ugrás	címkére
JMPC JMPCN	U	CR-függő feltételes ugrás	címkére
CAL	V	feltétel nélküli hívás	FB
CALC CALCN	U	CR-függő feltételes hívás	FB
RET	V vagy U	feltétel nélküli visszatérés	FGV-ből, FB-ból
RETC RETCN	U	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 deklarálása:

```
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 másodszori 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-késleltetéses 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 adatttípusú 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:

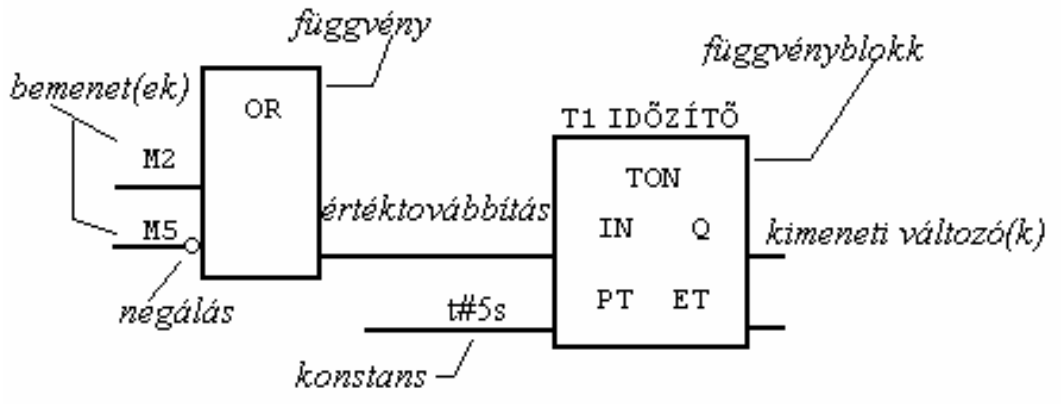
```
CAL fgvblk1(par1:= t#20ms, par2:=BE)
```

(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. Karaktersorozat 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**
Típuskonvertáló függvények		
*_TO_** (ANY)	ANY	típuskonverziók elemi adattípusok között
TRUNC (ANY_REAL)	ANY_INT	REAL szám egészét adja
Numerikus függvények		
ABS (ANY_NUM)	ANY_NUM	abszolút értékképzés
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	arckoszinusz fgv.
ATAN (ANY_REAL)	ANY_REAL	arctangens fgv.
Aritmetikai függvények (IN1,IN2)		
ADD (ANY_NUM, ANY_NUM)	ANY_NUM	összeadás
ADD (TIME, TIME)	TIME	időösszegzés
ADD (TOD, TIME)	TOD	időösszegzés
ADD (DT, TIME)	DT	időösszegzés
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
Léptető függvények (IN1,N)		
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
Bitsoros függvénye k (IN1,IN2)		
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
Kiválasztó függvények (IN1,IN2)		
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)
Összehasonlító függv. (IN1,IN2)		
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
Karaktersor függvények (IN1,IN2)		
LEN (STRING)	INT	karaktersor hossza
LEFT (STRING,L)	STRING	karaktersor balról
RIGHT (STRING,L)	STRING	karaktersor jobbról
MID (STRING,L,P)	STRING	karaktersor középről
CONCAT (STRING,STRING)	STRING	karaktersor összefűzés
INSERT (STRING,STRING,P)	STRING	karaktersor beszúrás
DELETE (STRING,L,P)	STRING	karaktersor törlés
REPLACE(STRING,STRING,L,P)	STRING	karaktersor csere
FIND (STRING,STRING)	INT	pozíció keresés

A táblázat rövidítéseinek magyarázata

bemenet	**jelentés**	**adattípus**
N	a léptetendő bitek száma	UINT
L	balpozíció a karaktersoron belül	ANY_INT
P	pozíció a karaktersoron 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**
R/S tárolók		
SR (S1,R)	Q	SET domináns
RS (R,S1)	Q	RESET domináns
Élkiértékelők		
R_TRIG (CLK)	Q	felfutó él felismerése
F_TRIG (CLK)	Q	lefutó él felismerése
Számlálók		
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
Időzítők		
TP (IN,PT)	Q,ET	impulzusadó
TON (IN,PT)	Q,ET	bekapcsolás-késleltetéses időzítő
TOF (IN,PT)	Q,ET	kikapcsolás-késleltetéses időzítő
RTC (EN,PDT)	Q,CDT	valósidejű 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

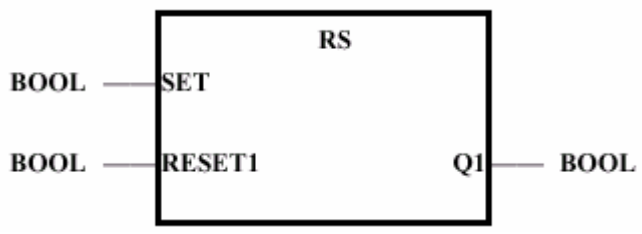
A standard függvényblokkok kimeneti értéke az első futtatás során nulla, kivéve a valósidejű órát.

A standard függvényblokkok bemeneti paraméterei kulcsszónak minősülnek. 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

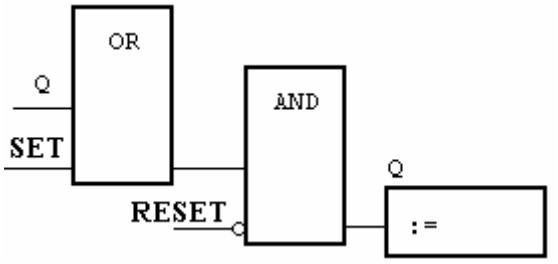
RS tároló

Funkciótervbeli jelölése:



6. ábra RS-tároló

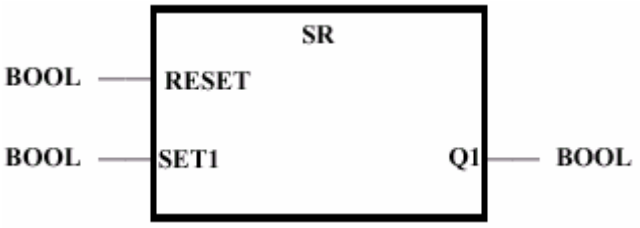
RESET domináns flip-flop. Ha a SET és RESET jel egyidejűleg 1 értékű, a RESET jel határozza meg a kimenetet, vagyis Q1=0. Az RS függvényblokk az alábbi funkciótervvel leírható algoritmus szerint működik:



7. ábra Az RS-tároló belső algoritmusa

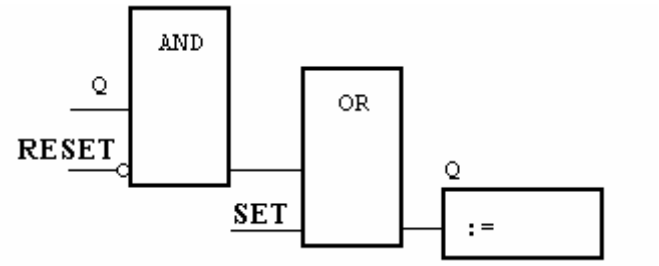
SR tároló

Funkciótervbeli jelölése:



8. ábra SR-tároló

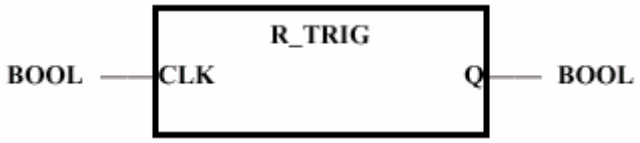
SET domináns flip-flop. Ha a SET és RESET jel egyidejűleg 1 értékű, a SET jel határozza meg a kimenetet, vagyis Q1=1. Az SR függvényblokk az alábbi funkciótervvel leírható algoritmus szerint működik:



9. ábra Az SR-tároló belső algoritmusa

Felfutó él detektálása: az R_TRIG függvényblokk

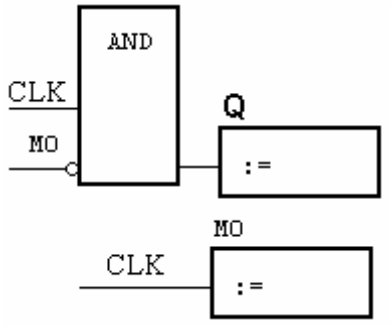
Ábrázolás funkciótervben:



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 algoritmusa funkciótervben:



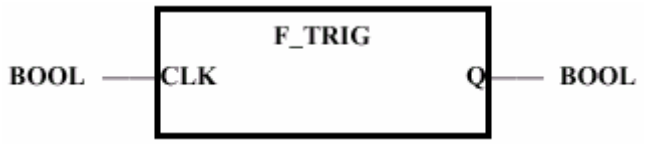
11. ábra Impulzus felfutó élre

Utasításlistában:

```
LD    CLK
ANDN  M0
ST    Q
LD    CLK
ST    M0 .
```

Lefutó él detektálása: az F_TRIG függvényblokk

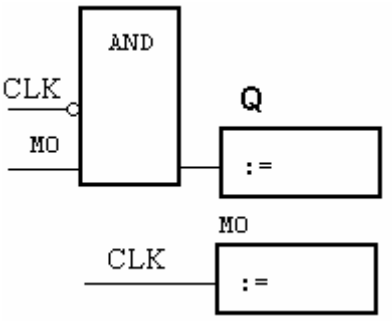
Ábrázolás funkciótervben:



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 algoritmusa funkciótervben:



13. ábra Impulzus lefutó élre

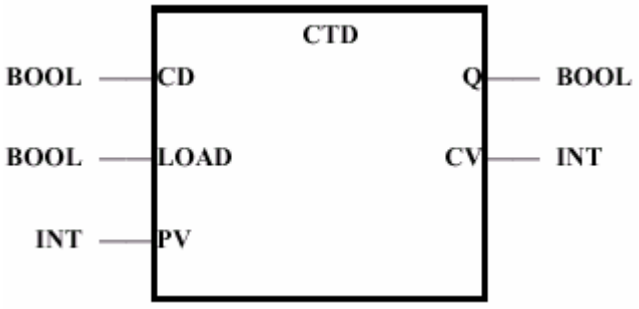
Utasításlistában:

```
LDN   CLK
AND   M0
ST    Q
LD    CLK
ST    M0 .
```

A számlálók

CTD (Count Down) lefelé számláló

Ábrázolása funkciótervben:



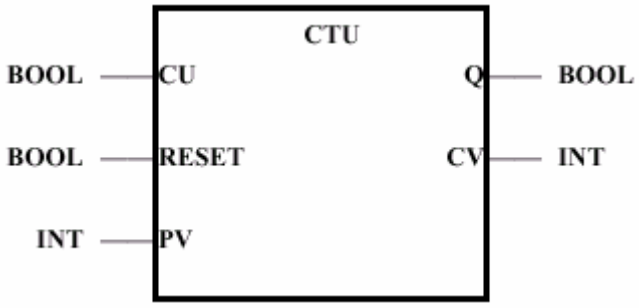
14. ábra A lefelé számláló

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 élére 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:



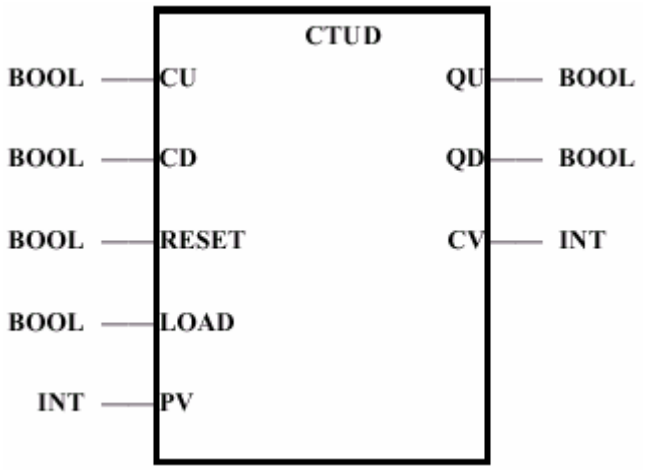
15. ábra A felfelé számláló

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≥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:

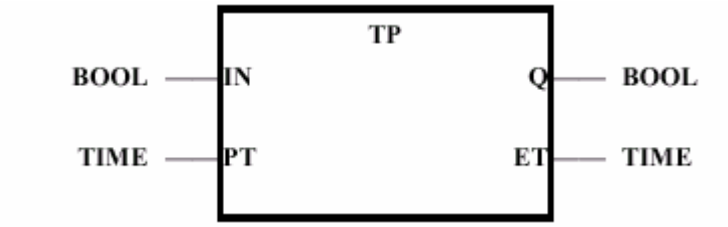
jelölés	jelentés
CU	A CU bemeneten megjelenő jel felfutó élre a számláló értékét eggyel növeli.
CD	A CD bemeneten megjelenő jel felfutó élre a számláló értékét eggyel csökkenti.
RESET	A RESET bemeneten lévő jel felfutó élére a számláló értékét 0-val teszi egyenlővé (a számláló kezdeti értékének beállítására).
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≥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 elsajátíttatása, ezért az alábbiakban csak a szabványban rögzített időzítőket mutatjuk be. Gyártótól és típustól függően az időzítők palettája sokkal szélesebb is lehet.

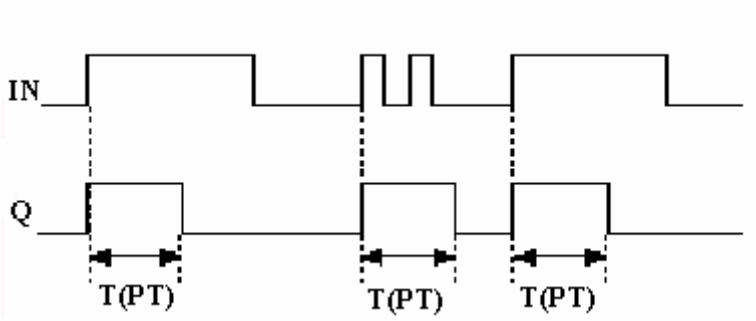
Impulzus időzítő (TP = Time Pulse)

Funkciótervbeli jelölése:



17. ábra Az impulzus időzítő funkciótervbeli ábrázolása

Az időzítő viselkedését bemutató idődiagram:



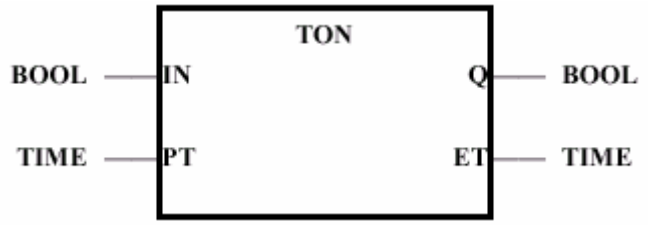
18. ábra Az impulzus időzítő idődiagramja

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.

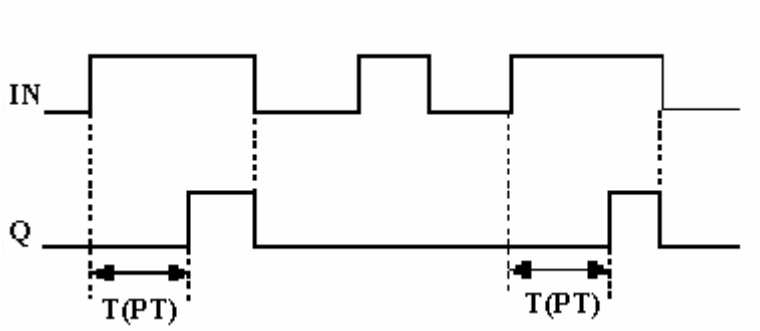
Bekapcsolás-késleltetéses időzítő

Funkciótervbeli jelölése:



19. ábra A bekapcsolás-késleltetéses időzítő funkciótervbeli ábrázolása

Az időzítő viselkedését bemutató időidiagram:



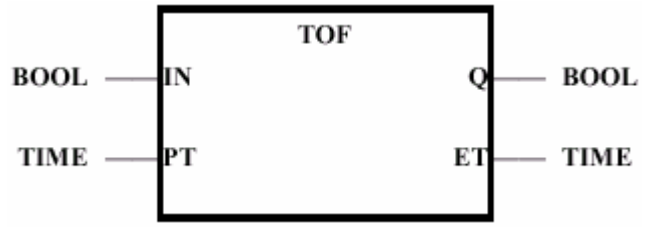
20. ábra A bekapcsolás-késleltetéses időzítő időidiagramja

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 időidiagram szerint.
ET	Az indítás óta eltelt idő. Értéke nem lehet nagyobb PT-nél.

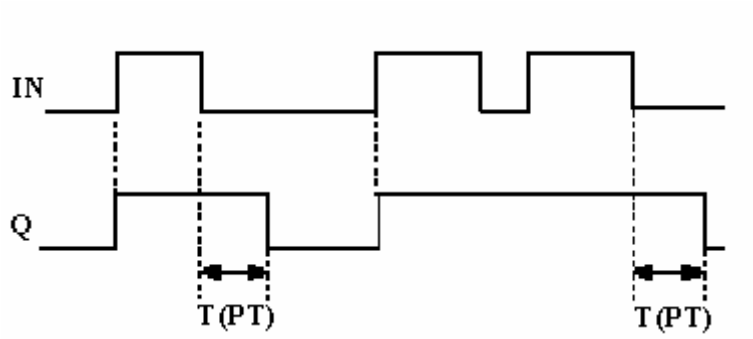
Kikapcsolás-késleltetéses időzítő

Funkciótervbeli jelölése:



21. ábra A kikapcsolás-késleltetéses időzítő funkciótervbeli ábrázolása

Az időzítő viselkedését bemutató idődiagram:



22. ábra A kikapcsolás-késleltetéses időzítő idődiagramja

A be/kimeneti jelek értelmezése:

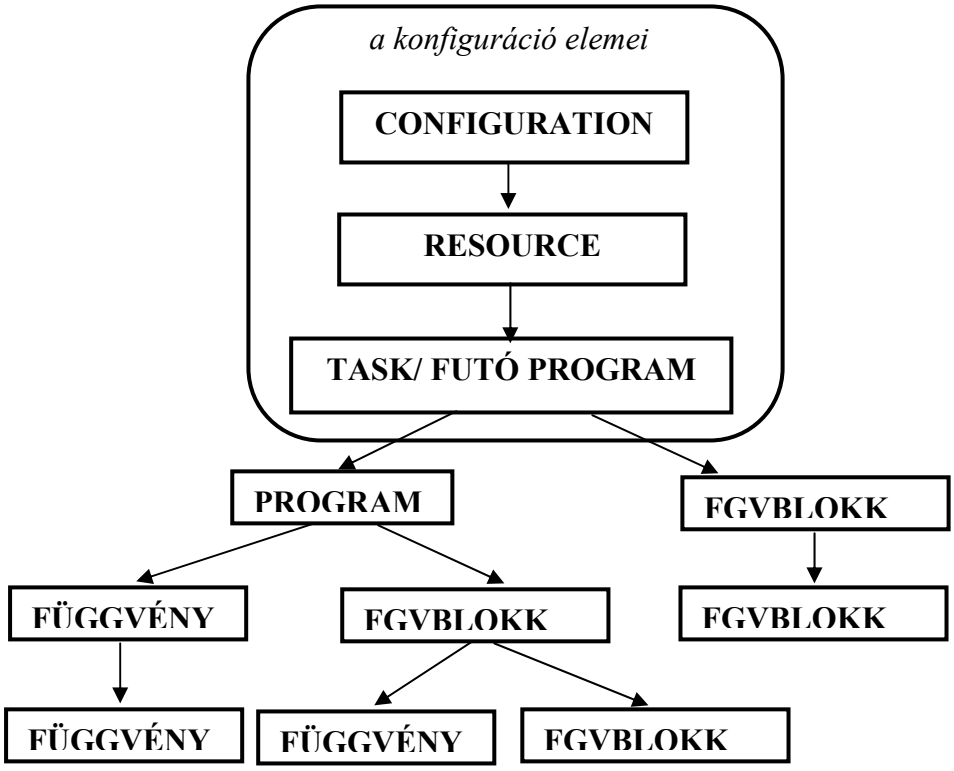
jelölés	jelentés
IN	Az IN bemeneten jelentkező felfutó élre a kimenet 1-re vált.
PT	Az IN bemenet lefutó éle után PT ideig a Q kimeneten még fenntartandó az 1 jel.
Q	Az időzítő kétállapotú kimenete. Beállítása az idődiagram szerint.
ET	A lefutó él óta eltelt idő. Értéke nem lesz negatív.

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 futtasási sajátságokat 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 sajátosságaikat, 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_szszalag ON    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.

TASK-paraméter	jelentés
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 (preemtive 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-preemtive 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.

PÉLDATÁR

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 vezérlőalgoritmus 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 **zöld**et mutat.
- Ha két ventilátor működik, a lámpa **sárgá**t jelez.
- Ha kettőnél kevesebb ventilátor működik, **piros** jelzést kell adni.

Összerendelési táblázat

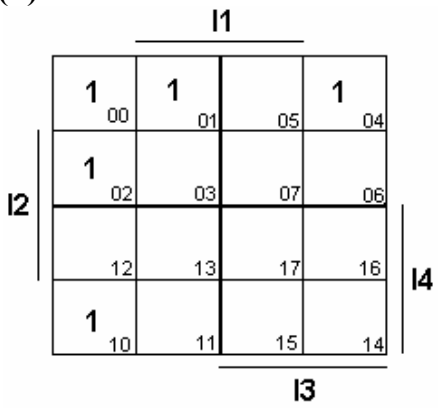
Bemenetek	Jel	Logikai hozzárendelés	Cím
1. áramlásjelző	I1	1. ventilátor üzemel: I1=1	I0.0
2. áramlásjelző	I2	2. ventilátor üzemel: I2=1	I0.1
3. áramlásjelző	I3	3. ventilátor üzemel: I3=1	I0.2
4. áramlásjelző	I4	4. ventilátor üzemel: I4=1	I0.3
Kimenetek			
Piros lámpa	P	világít, ha: P=1	Q0.2
Sárga lámpa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

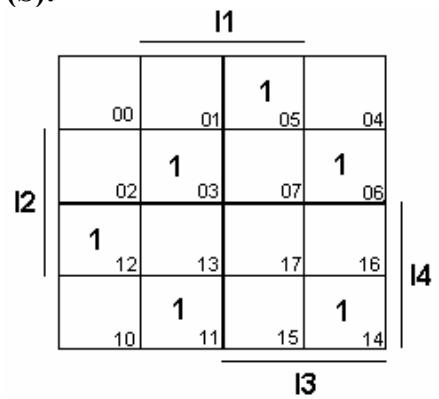
Karno-tábla

Piros (P):



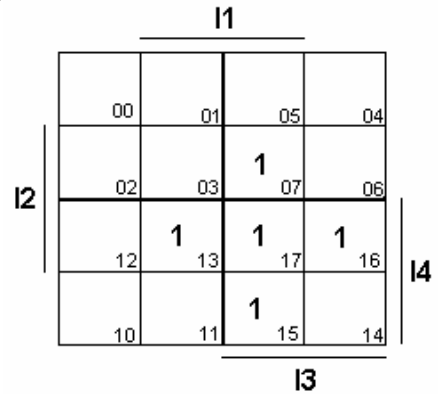
$\bar{I}1\,\bar{I}2\,I4$ V
$\bar{I}1\,\bar{I}2\,\bar{I}3$ V
$\bar{I}1\,I3\,\bar{I}4$ V
$\bar{I}2\,I3\,I4$

Sárga (S):



$\bar{I}1\,\bar{I}2\,I3\,I4$ V
$I1\,\bar{I}2\,\bar{I}3\,I4$ V
$\bar{I}1\,I2\,\bar{I}3\,I4$ V
$\bar{I}1\,I2\,I3\,\bar{I}4$ V
$I1\,I2\,\bar{I}3\,\bar{I}4$ V
$I1\,\bar{I}2\,I3\,\bar{I}4$

Zöld (Z):

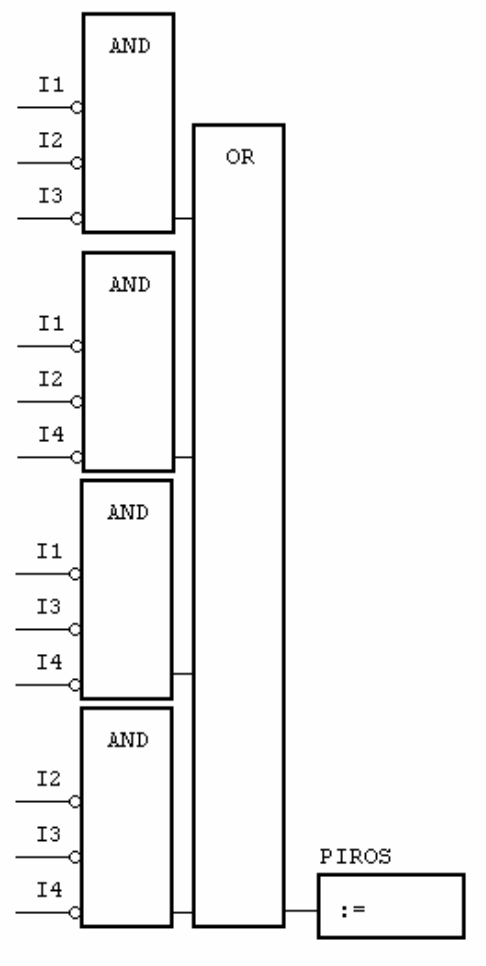


$I1\,I3\,I4$ V
$I2\,I3\,I4$ V
$I1\,I2\,I3$ V
$I1\,I2\,I4$

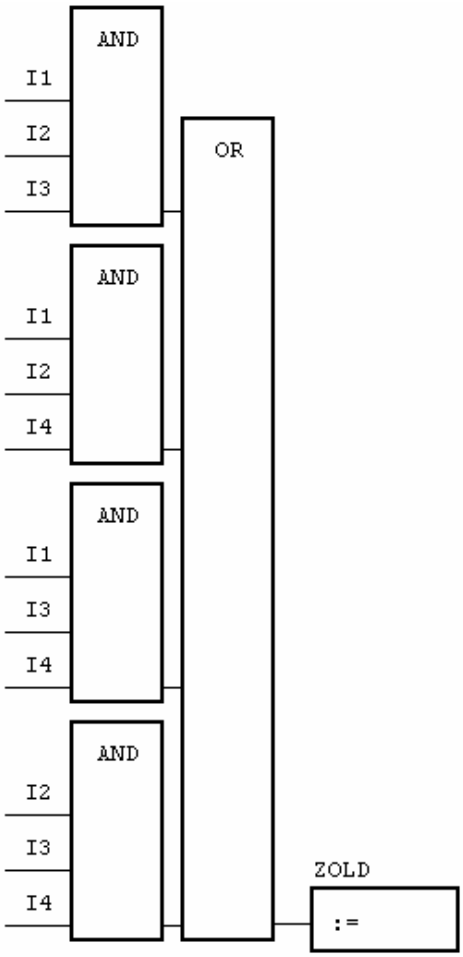
Mivel egy lámpának mindig világítania kell, elegendő, ha a kapcsolási feltételeket csak két lámpára írjuk meg, a harmadik pedig akkor lesz igaz, ha a másik kettő hamis. Mivel a sárga logikai függvénye a leghosszabb, ezért legyen: $S=\bar{P}\&\bar{Z}$

Funkcióterv

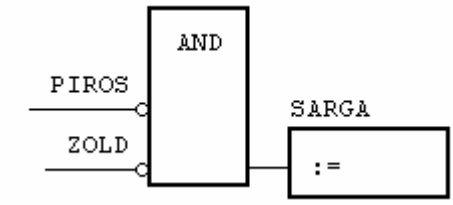
Piros lámpa világít:



Zöld lámpa világít:



Sárga lámpa világít:



Utasításlista

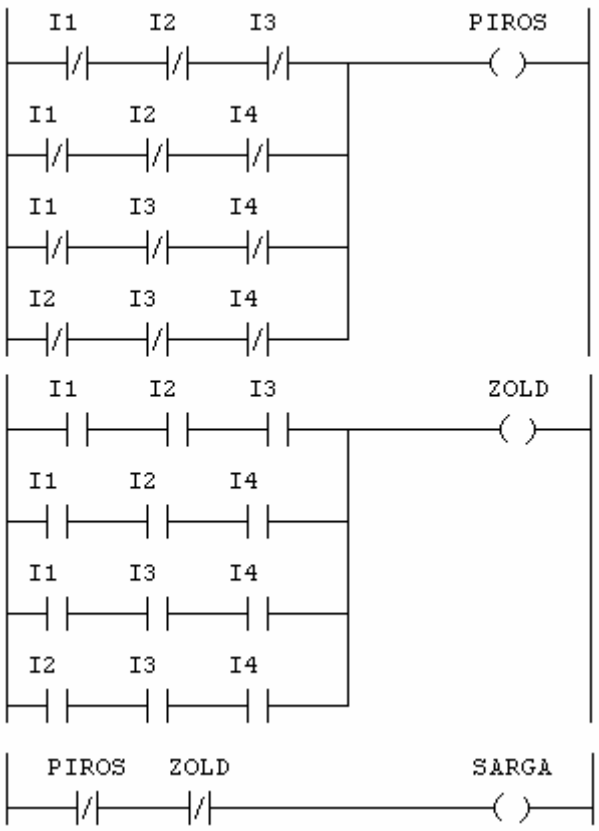
```
PROGRAM SZELLOZ
VAR
        I1 AT %I0.0.0.0.0:    BOOL;
        I2 AT %I0.0.0.0.1:    BOOL;
        I3 AT %I0.0.0.0.2:    BOOL;
        I4 AT %I0.0.0.0.3:    BOOL;
        PIROS       AT        %Q0.0.0.0.2:
        BOOL;
        SARGA        AT       %Q0.0.0.0.1:
        BOOL;
        ZOLD      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

<table>
<tr><th></th><th>Problémaleírás</th><th colspan="5">Szabályok</th><th>Egyéb-ként</th></tr>
<tr><th></th><th></th><th>R1</th><th>R2</th><th>R3</th><th>...</th><th>Rn</th><th></th></tr>
<tr><td>Feltételek</td><td>1.bemenő változó
2.bemenő változó
.
.
n.bemenő változó</td><td colspan="5">Feltétel vagy esetleírások szabályok megadásával.
(Az olyan bemeneti jelkombinációra, amelyre nincs szabály, az EGYÉB oszlop vonatkozik!)</td><td></td></tr>
<tr><td>Következ-mények</td><td>1.kimenő változó
2.kimenő változó
.
.
n.kimenő változó</td><td colspan="5">A feltételektől függő következmények (akciók) jelölése.</td><td></td></tr>
</table>

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 össszerendelés	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
Kimenetek			
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

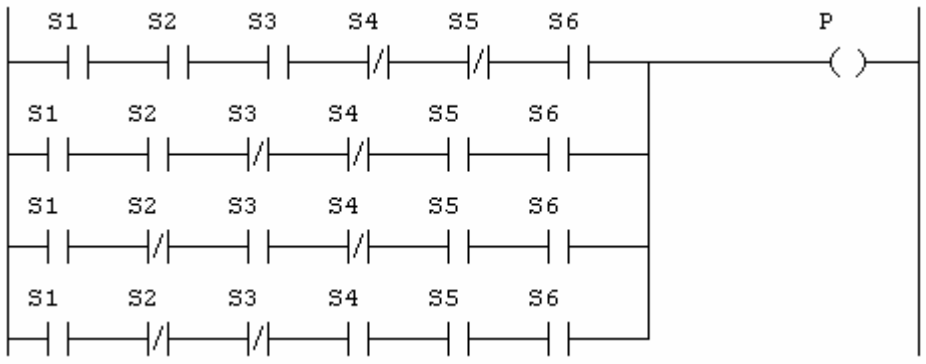
A redukált függvénytáblázat

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:

$P = S6\overline{S5}S4S3S2S1 \vee S6\overline{S5}\overline{S4}S3S2S1 \vee S6\overline{S6}S5\overline{S4}S3S2S1 \vee S6\overline{S5}\overline{S4}S3S2S1$

Létradiagram

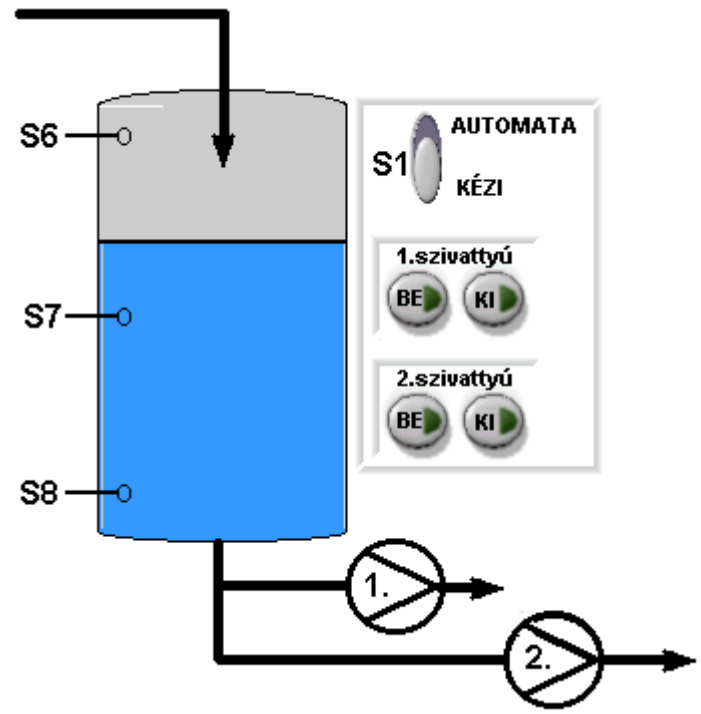


A program utasításlistája

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

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 fölötte 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 szivattyúkhoz tartozó be- ill. kikapcsoló nyomógombokkal.

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

Emelkedő folyadékszintnél:

S8 és **S7** között az 1. sz. szivattyú működjön;

S7 felett mindkét szivattyú kapcsoljon be.

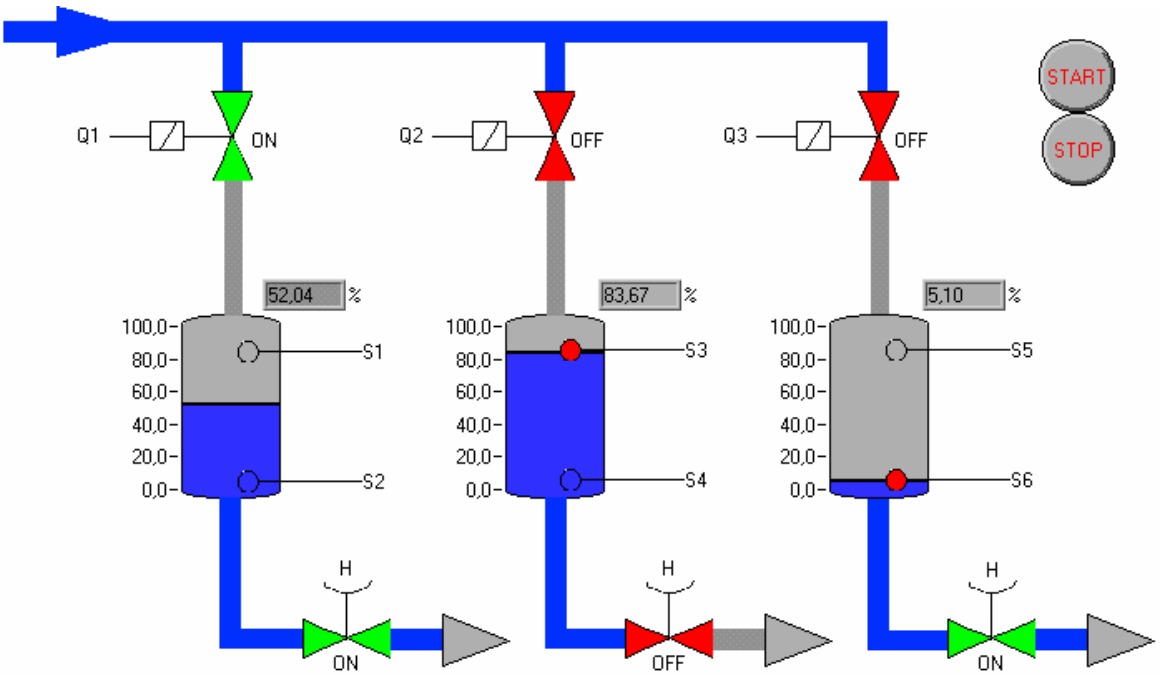
Csökkenő folyadékszintnél, ha **S8** szintérzékelő jelez, mindkét szivattyú álljon le.

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 tárolót töltsön fel. A tartály feltöltése akkor fejeződik be, ha a tele jel megérkezik. A tartályokat kézi szeleppel ürítik.

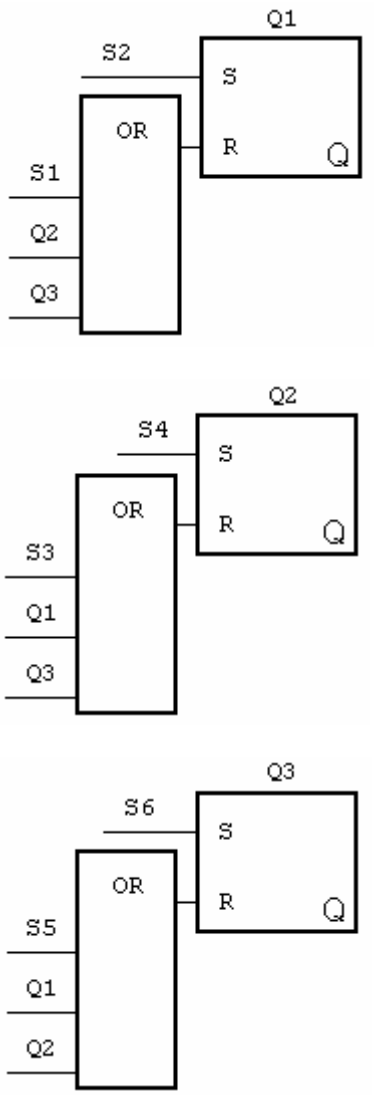


24. ábra Tárolótartályok feltöltésének vezérlése

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 :

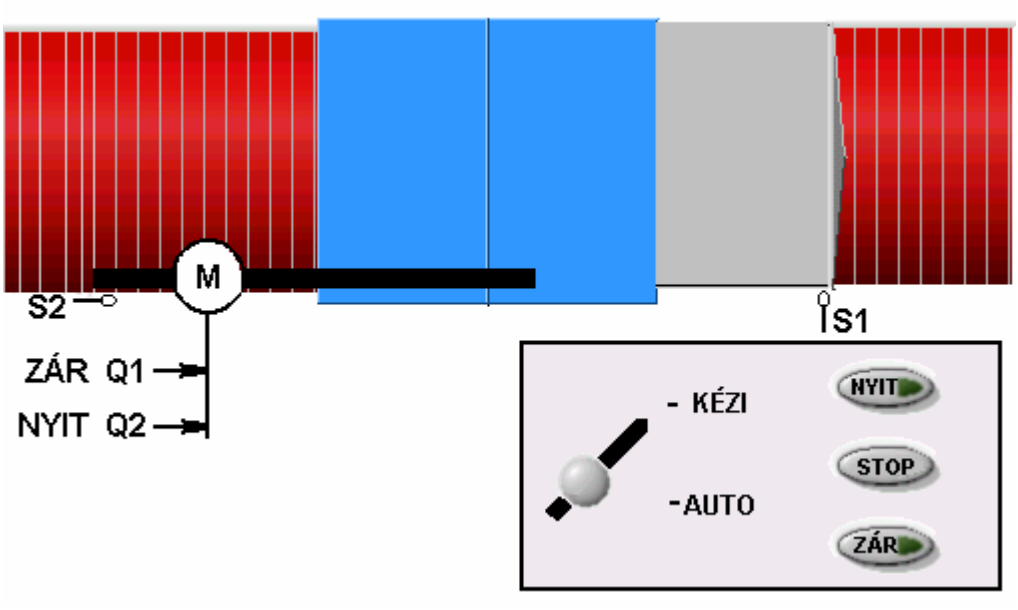
```
PROGRAM PR3TART
VAR
        S1 AT %I0.0.0.0.0:
        BOOL;
        S2 AT %I0.0.0.0.1:
        BOOL;
        S3 AT %I0.0.0.0.2:
        BOOL;
        S4 AT %I0.0.0.0.3:
        BOOL;
        S5 AT %I0.0.0.0.4:
        BOOL;
        S6 AT %I0.0.0.0.5:
        BOOL;
        Q1 AT %Q0.0.0.0.0:
        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 töltse 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 reteszeljék egymást a kapcsolási oldalon is. A kapu véghelyzeteit végállás-kapcsolók (**S1**: a kapu zárva, **S2**: a kapu nyitva) jelzik.



25. ábra Gyárkapu vezérlése

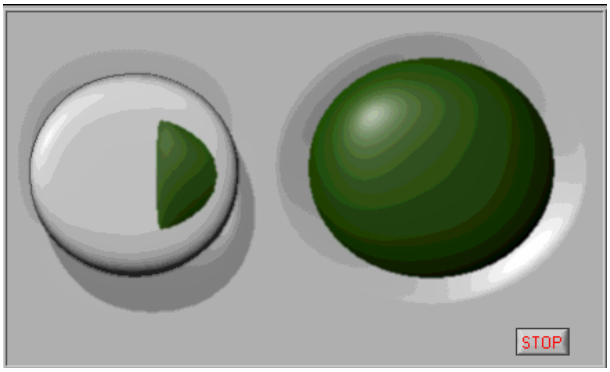
A kapusfülkében helyezték el a kapu kezelői pultját. A kaput kézi ill. automata üzemmódban lehet nyitni/zárni. A kívánt működés automata üzemmódban: a gomb rövid idejű benyomásával a kapu a véghelyzetig folyamatosan nyílik, illetve záródik. A művelet a STOP gomb benyomásával bármikor megszakítható. A 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
NYIT nyomógomb	NYIT	benyomva:	NYIT=1	I0.3
STOP nyomógomb	STOP	benyomva:	STOP=0	I0.4
ZÁR nyomógomb	ZAR	benyomva:	ZAR=1	I0.5
Kimenetek				
zárás irányba kapcsoló relé	Q1	behúzva:	Q1=1	Q0.1
nyitás irányba kapcsoló relé	Q2	behúzva:	Q2=1	Q0.2

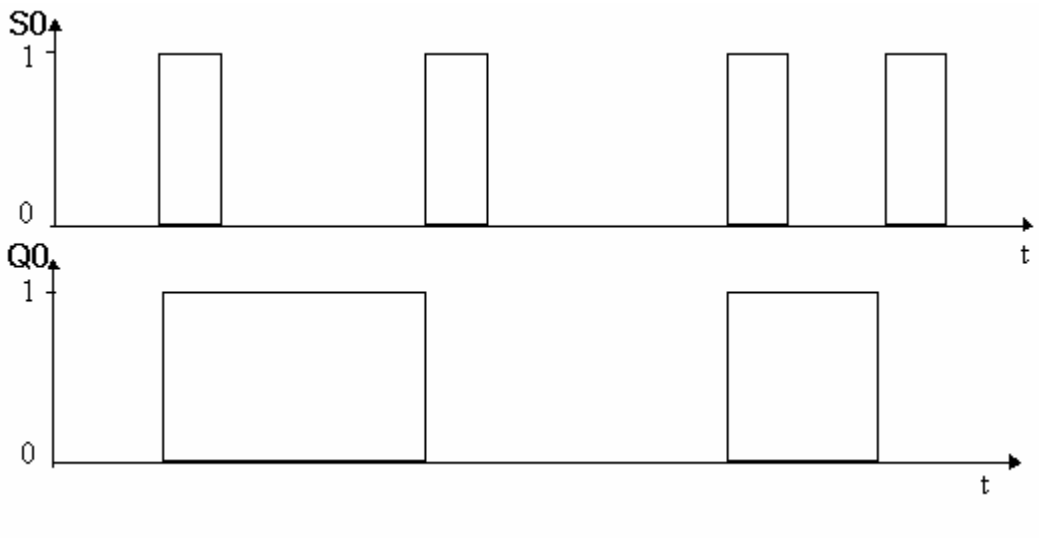
Impulzuskapcsoló

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

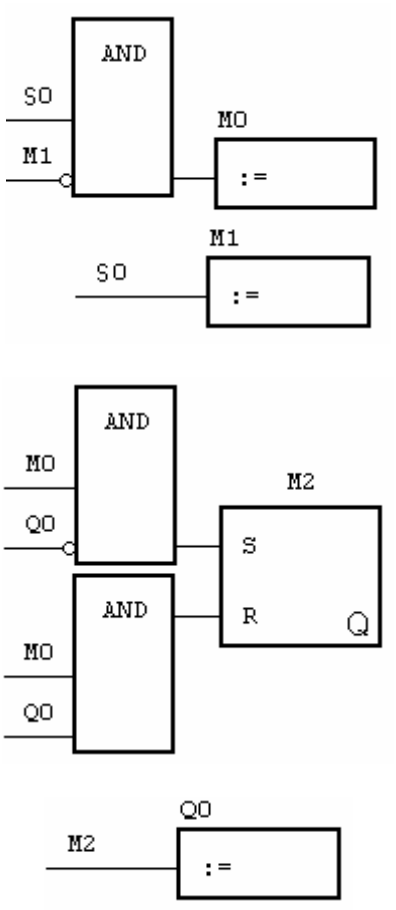
Bemenet	Jel	Logikai összerendelés	Cím
Nyomógomb	S0	benyomva: S0=1	I0.0
Kimenet			
Jelzőlámpa	Q0	világít: Q0=1	Q0.0

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
ANDN    M1
ST      M0

LD      S0
ST      M1

LD      M0
ANDN    Q0
S       M2
LD      M0
AND     Q0
R       M2

LD      M2
ST      Q0
END_PROGRAM
```

Funkcióterv

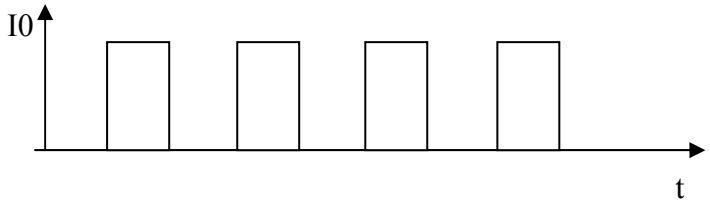


Gyakorló feladat: utasításlista elemzése I.

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:

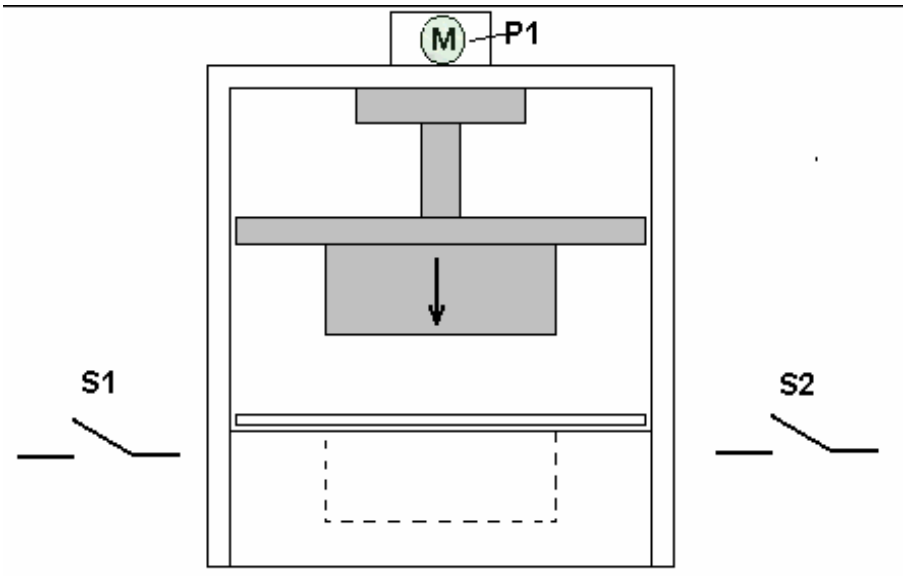


A kimenőjel időbeli alakulása a bemenőjel függvényében (megoldandó feladat!):

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

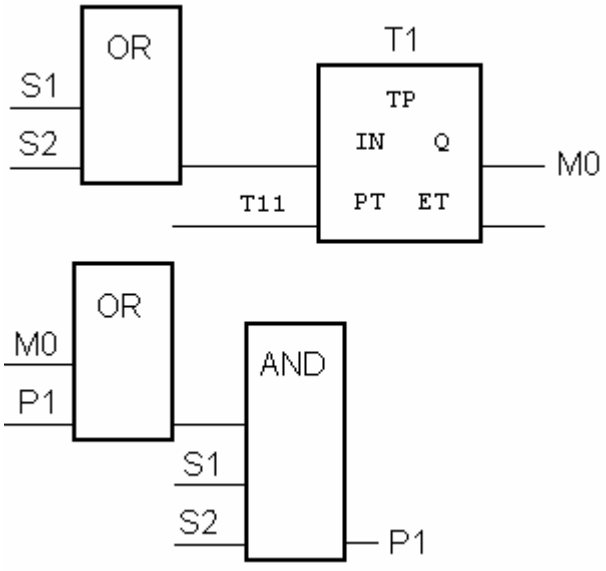
Összerendelési táblázat

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
minden egyéb estben				**0**

Funkcióterv

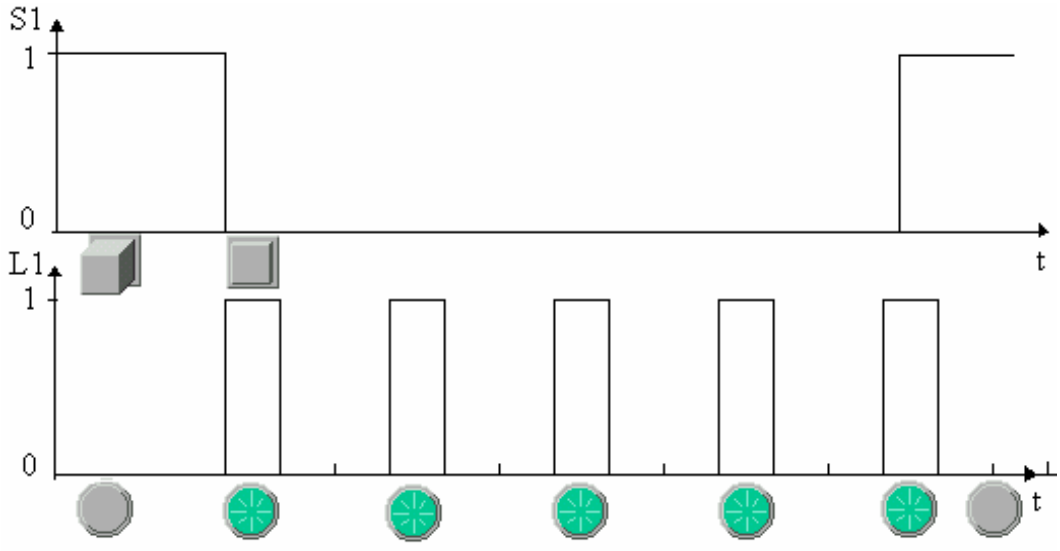


A program utasításlistája

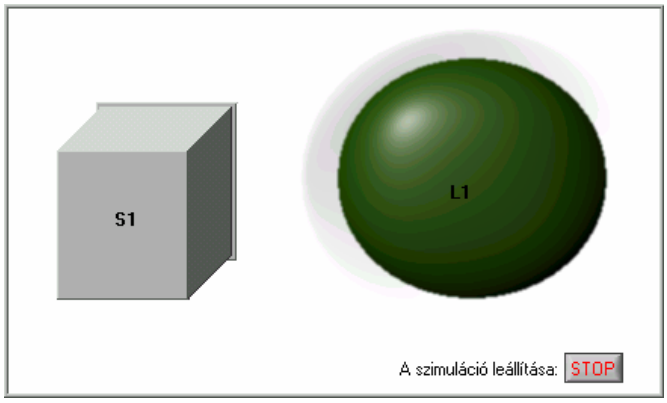
```
PROGRAM ketkret
VAR
        S1 AT %I0.1 : BOOL ;
        S2 AT %I0.2 : BOOL ;
        P1 AT %Q0.1 : BOOL ;
END_VAR

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



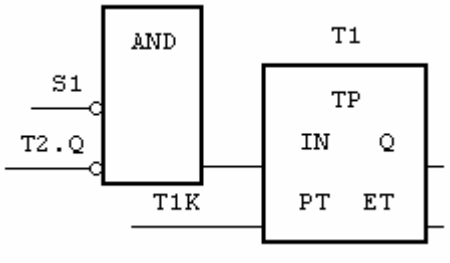
30. ábra A kívánt működést szimuláló program frontpanelképe

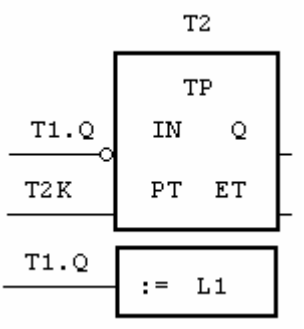
Összerendelési táblázat

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** jelváltása 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
```

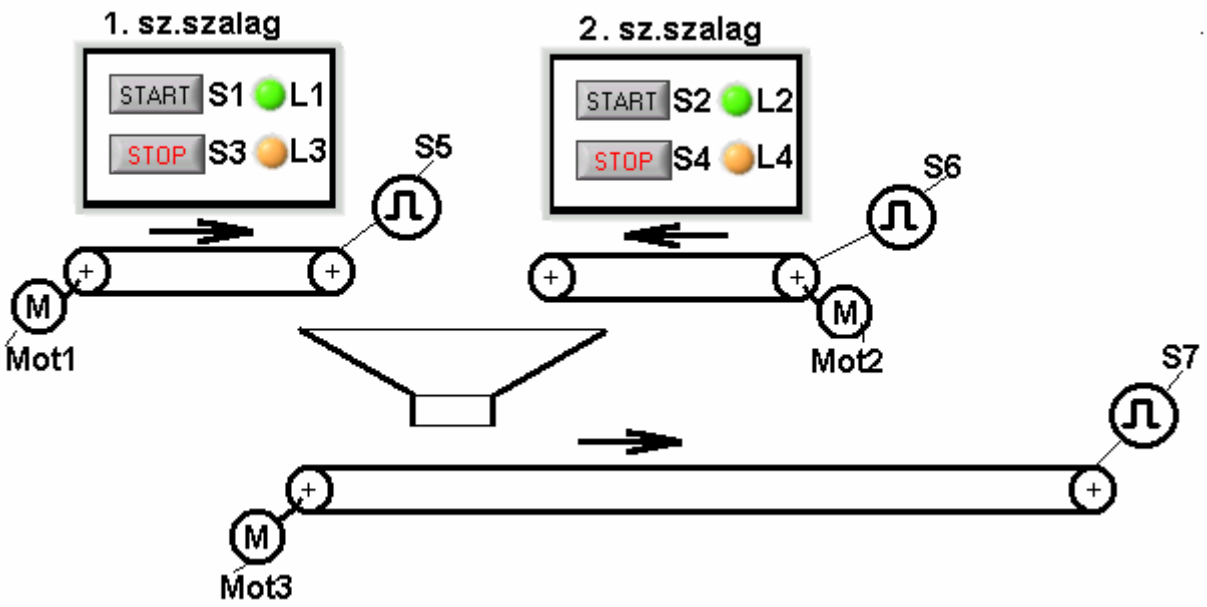
A bemenőjel időbeli változása:

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

Követővezérlés időzítőkkel

Szállítószalagok együttes vezérlése

A kikapcsolás-késleltetéses 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 vezérlőalgoritmus 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:

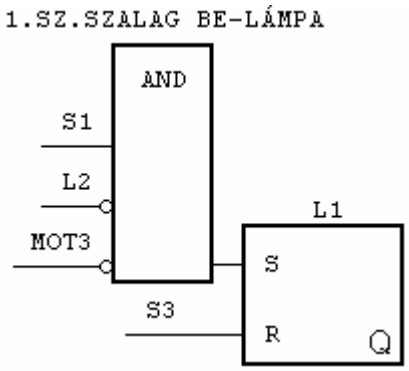
- Az 1. és 2. szállítószalagok kézi nyomógombokkal kapcsolhatók be/ki (**S1, S2, S3, S4**).
- Az üzemállapotokat jelzőlámpákkal kell visszajelezni (**L1, L2, L3, L4**).
- Az 1. és 2. szállítószalag nem működhet egyidejűleg.
- A 3. szállítószalagnak mindig működnie kell, ha az 1-t vagy a 2-t elindították.
- 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.

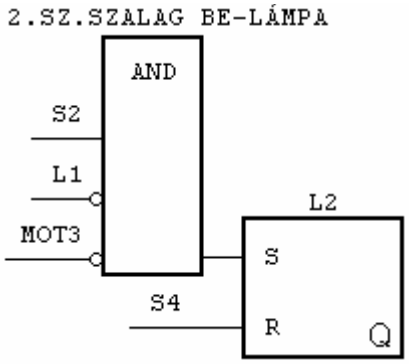
- A hibajelzést (villogást) a megfelelő szállítószalag **STOP** nyomógombjának megnyomásával lehet nyugtázni.

Összerendelési táblázat

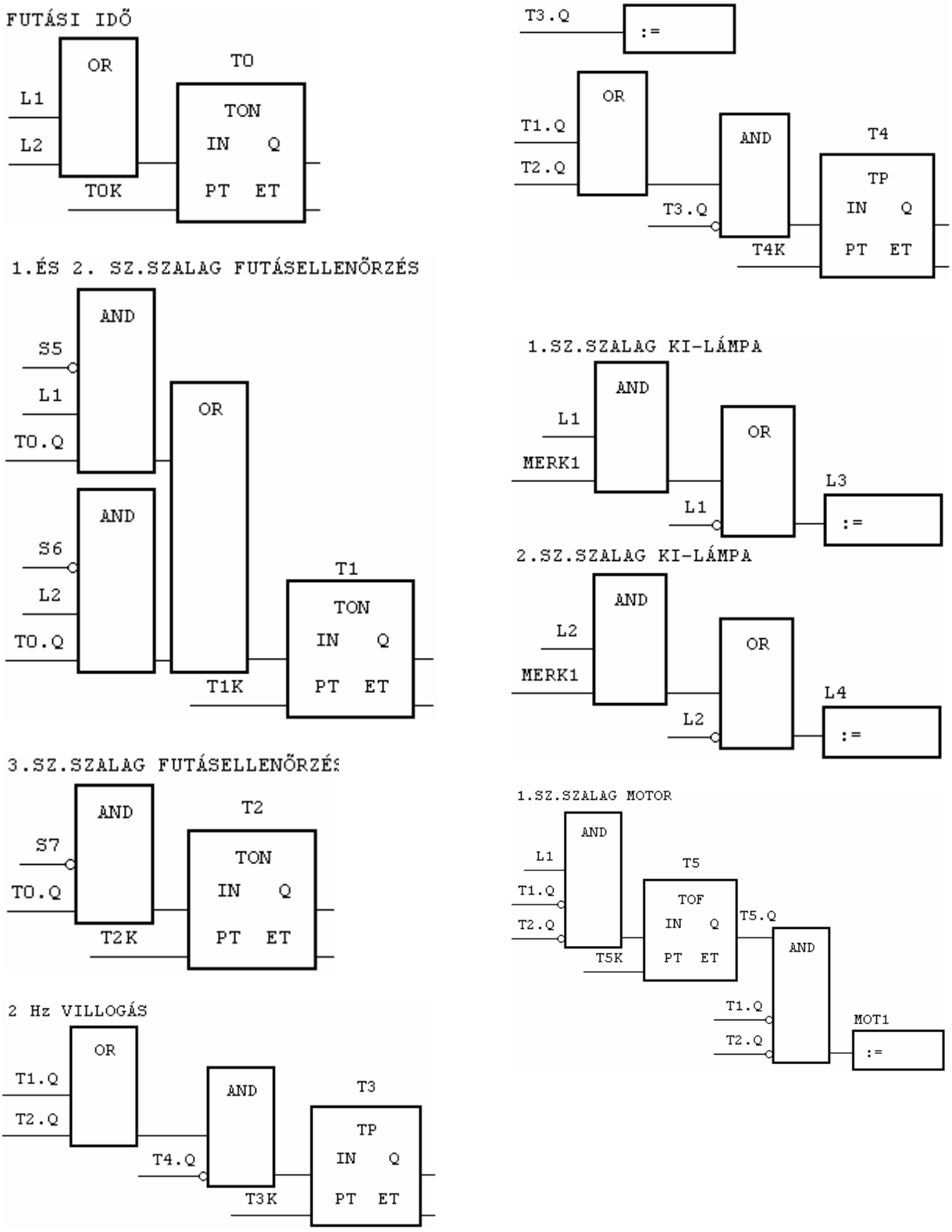
Bemenetek	Jel	Logikai összerendelés	Cím
1. sz.szalag BE-kapcsolás nyomógomb	S1	benyomva: S1=1	I0.0
2. sz.szalag BE-kapcsolás nyomógomb	S2	benyomva: S2=1	I0.1
1. sz.szalag KI-kapcsolás nyomógomb	S3	benyomva: S3=1	I0.2
2. sz.szalag KI-kapcsolás nyomógomb	S4	benyomva: S4=1	I0.3
1. sz.szalag fordulatjelző	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

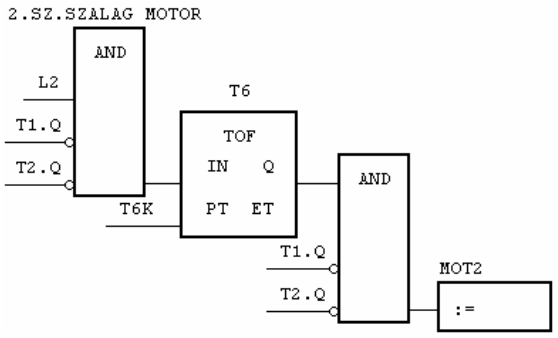
Funkcióterv

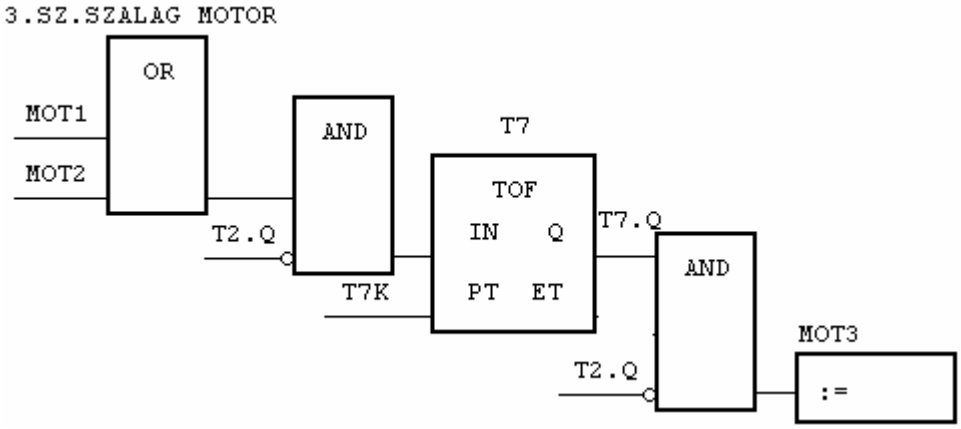




A rugalmasabb adatmódosítás biztosításának érdekében az időzítők időállandóit a deklarációs részben rögzítettük, a funkciótervben is a változóneveket tüntettük fel.







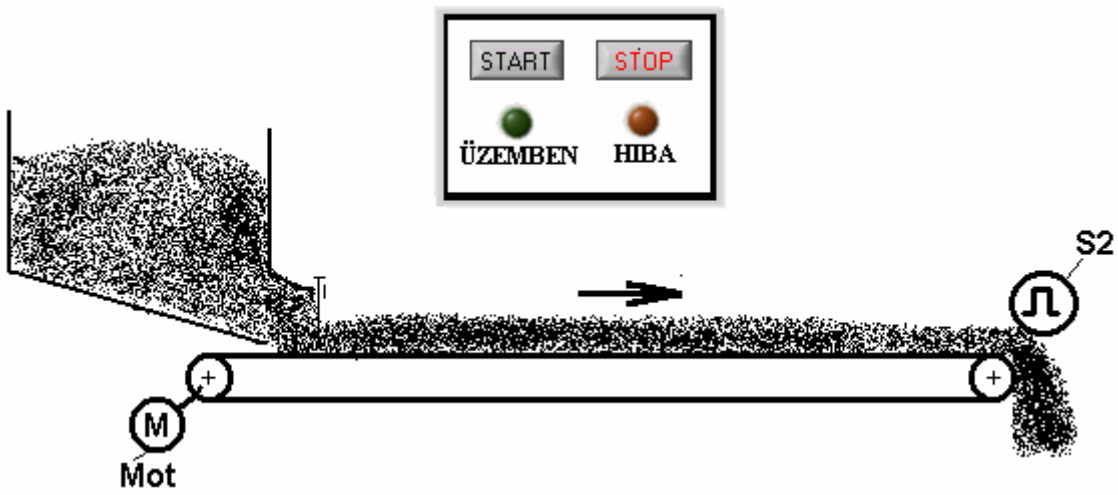
Utasításlista

```
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
        ANDN   L2
        ANDN   MOT3
        S      L1
        LD     S3
        R      L1
        (*2.SZ.SZALAG BE-LÁMPA*)
        LD     S2
        ANDN   L1
        ANDN   MOT3
        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
        ANDN   T4.Q
        ST     T3.IN
        LD     T3K
        ST     T3.PT
        CAL    T3
        LD     T3.Q
        ST     MERK1
        LD     T1.Q
        OR     T2.Q
        ANDN   T3.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
        ANDN   T1.Q
        ANDN   T2.Q
        ST     T5.IN
        LD     T5K
        ST     T5.PT
        CAL    T5

        LD     T5.Q
        ANDN   T1.Q
```

```
ANDN T2.Q
ST   MOT1
(*2.SZ.SZALAG MOTOR*)
LD   L2
ANDN T1.Q
ANDN T2.Q
ST   T6.IN
LD   T6K
ST   T6.PT
CAL  T6
LD   T6.Q
ANDN T1.Q
ANDN T2.Q
ST   MOT2
(*3.SZ.SZALAG MOTOR*)
LD   MOT1
OR   MOT2
ANDN T2.Q
ST   T7.IN
LD   T7K
ST   T7.PT
CAL  T7
LD   T7.Q
ANDN T2.Q
ST   MOT3
END_PROGRAM
```

Gyakorló feladat: Szállítószalag vezérlése



32. ábra Szállítószalag motor vezérlése

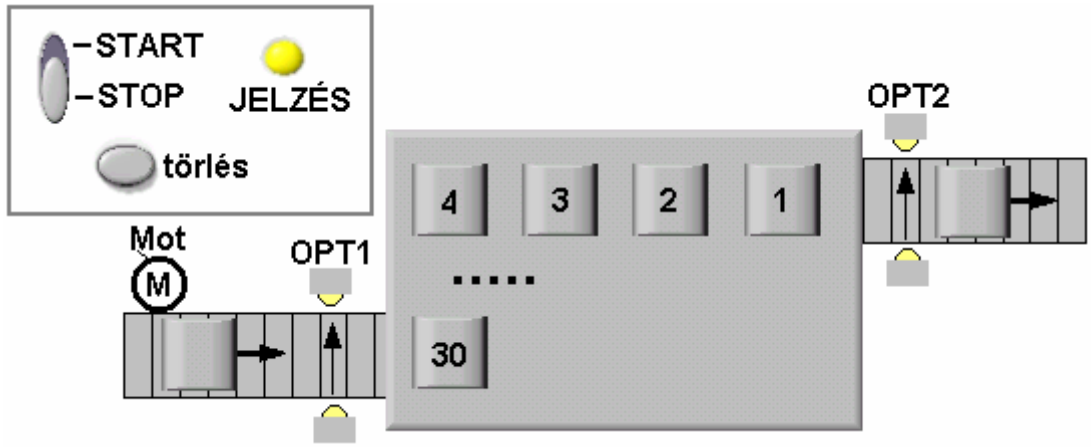
A szállítószalagot a **Mot** jelzésű motor működteti. Amíg a szalag megfelelően fut, az **S2** jele 10 Hz frekvenciajel. Probléma esetén (pl. szalagszakadás stb.) az **S2** jeladó folyamatosan 0 értéket ad. Ha bekapcsolt motor mellett nem jelentkezik az impulzusjel, le kell állítani a motort, és a **HIBA** 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	START	benyomva: START=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 üzemkezdetkor, üres tároló mellett, a törlőgomb benyomásával lehetséges.

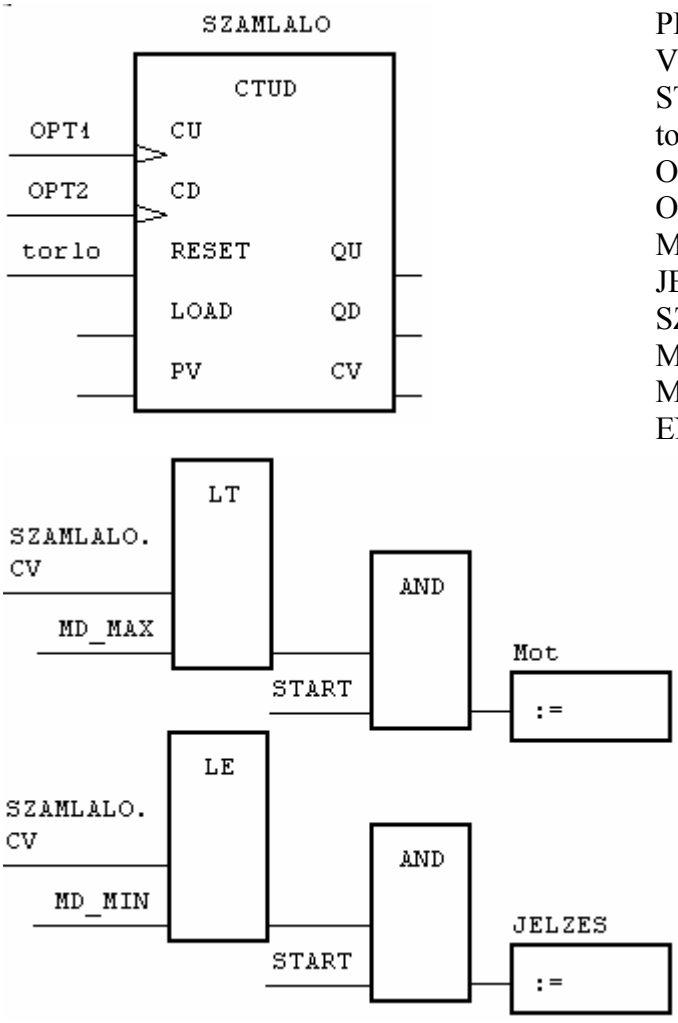


33. ábra Átmeneti munkadarab-tároló

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



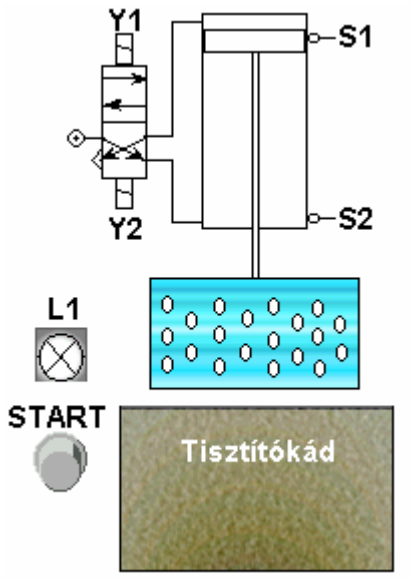
Utasításlista

```
PROGRAM mdtarol
VAR
START AT %I0.0.0.0.0:      BOOL;
torlo AT %I0.0.0.0.1: BOOL;
OPT1 AT %I0.0.0.0.2:       BOOL;
OPT2 AT %I0.0.0.0.3:       BOOL;
Mot AT %Q0.0.0.0.0:BOOL;
JELZES AT %Q0.0.0.0.1:     BOOL;
SZAMLALO: CTUD;
MD_MAX:    INT := 30;
MD_MIN:    INT := 10;
END_VAR
```

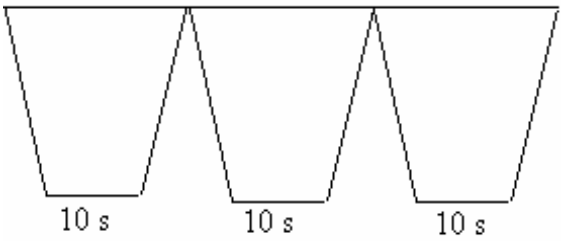
```
(*SZÁMLÁLÓ*)
LD      OPT1
ST      SZAMLALO.CU
LD      OPT2
ST      SZAMLALO.CD
LD      torlo
ST      SZAMLALO.RESET
CAL     SZAMLALO

(*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



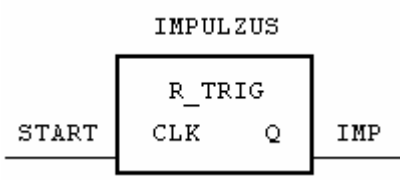
35. ábra A tisztítási ciklus idődiagramja

Összerendelési táblázat

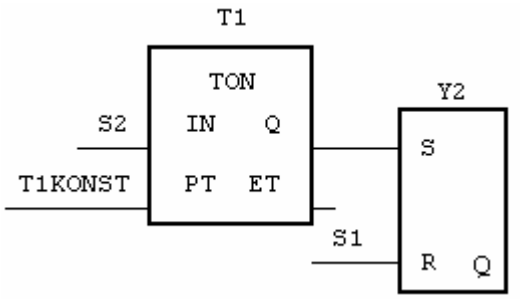
Bemenetek	Jel	Logikai összerendelés	Cím
BE/KI kapcsoló	START	benyomva: START=1	I0.0
felső végálláskapcsoló	S1	jelez, ha: S1=1	I0.1
alsó végálláskapcsoló	S2	jelez, ha: S2=1	I0.2
Kimenetek			
Munkahenger le	Y1	működtetve: Y1=1	Q0.0
Munkahenger fel	Y2	működtetve: Y2=1	Q0.1
Lámpa	L1	világít, ha: L1=1	Q0.2

Funkcióterv

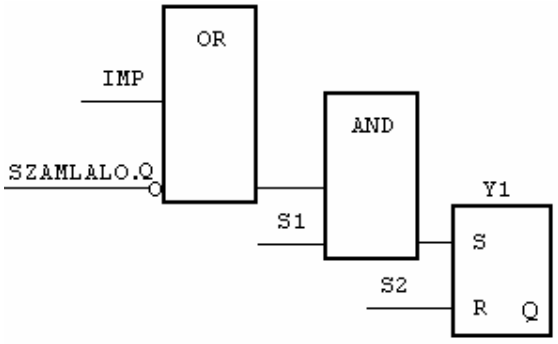
Bekapcsolási impulzus:



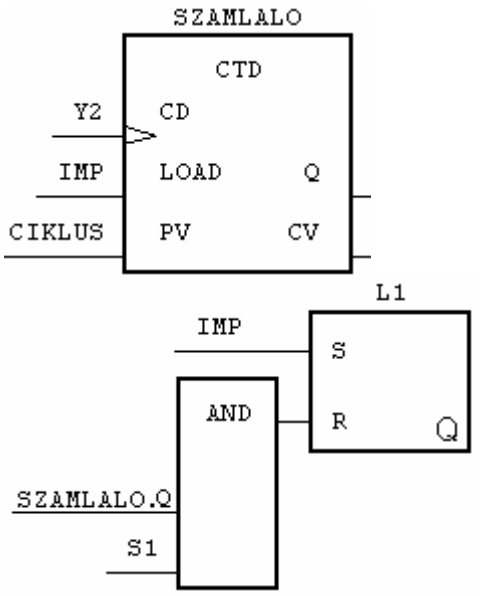
Kosár fel:



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!):

Jancskárné Anweiler Ildikó

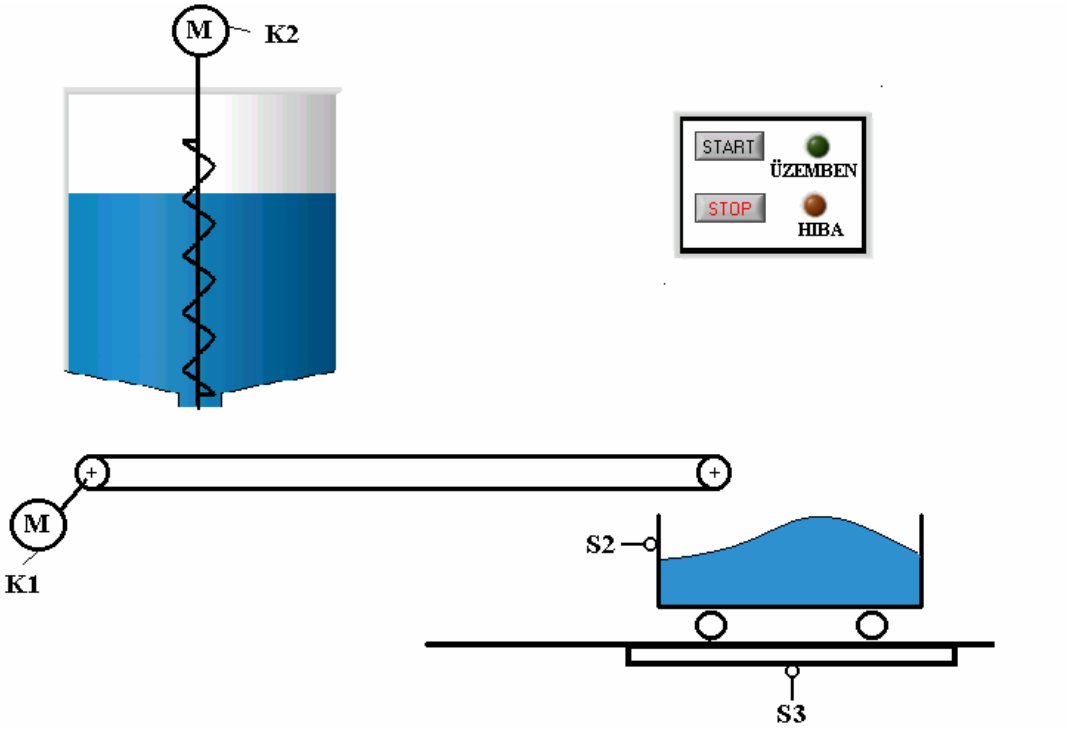
PLC programozás II. rész

KÉZIRAT!

Követővezérlési feladatok megoldása állapotgráf segítségével

Vagontöltő berendezés

A vagont a szilárdanyag tárolóból (siló), adagolócsiga és szállítószalag segítségével töltik fel. Az adagolást a **STRART** gomb megnyomásával engedélyezik. A **START** jel csak akkor hatásos, ha a vagon töltési helyzetben van (**S2** jelez). Ekkor az adagolandó anyag feltorlódásának elkerülése érdekében először a szállítószalagot kell elindítani, és **3s**-ig üresen járatni. Az idő letelte után bekapcsolható az adagolócsiga motorja is. Ha megtelt a vagon, vagy a vagon elmozdult a töltési pozíciójából, vagy megnyomták a **STOP** gombot, az adagolócsigát azonnal le kell állítani. Ekkor a szállítószalag még **5s**-ig bekapcsolva marad, hogy teljesen leürüljön. Újabb adagolást a **START** gomb ismételt benyomásával lehet elindítani.



36. ábra Vagontöltés

Összerendelési táblázat

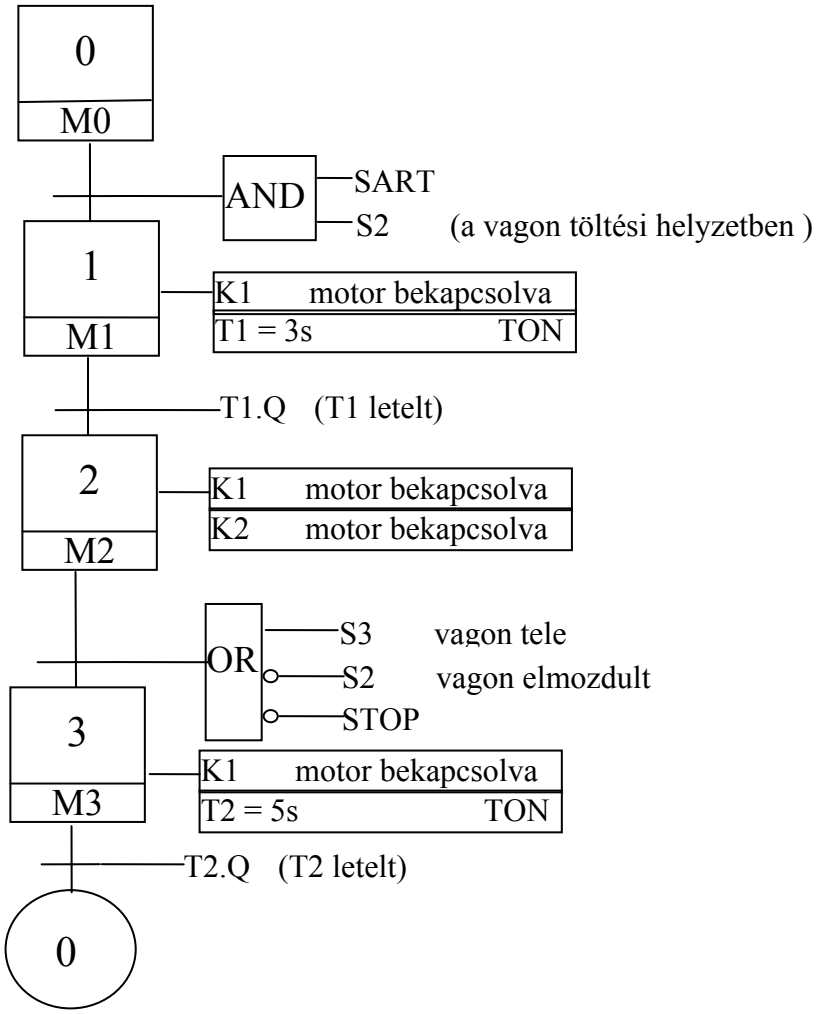
Bemenetek	Jel	Logikai összerendelés		Cím
START nyomógomb	START	benyomva:	START=1	I0.0
STOP nyomógomb	STOP	benyomva:	STOP=0	I0.1
Rámpaérzékelő	S2	a vagon töltési pozícióban:	S2=1	I0.2
Súlyérzékelő	S3	a vagon tele:	S3=1	I0.3
Kimenetek				
Szállítószalag motor	K1	bekapcsolva:	K1=1	Q0.0
Adagolócsiga motor	K2	bekapcsolva:	K2=1	Q0.1

A vezérlés állapotai

- **Alapállapot (M0):** semmit sem működik, a vezérlés a **START** jelre vár
- **(M1):** Adott felfutási ideig csak a szállítószalag motorja van bekapcsolva
- **(M2):** Mindkét kimenetet (szállítószalag és adagolócsiga) működteti a vezérlés
- **(M3):** Adott leállítási ideig csak a szállítószalag működik

Az állapotok közötti kapcsolatokat, egyik állapotból a másikba történő átlépés feltételeit állapotgráf segítségével írjuk le.

Állapotgráf

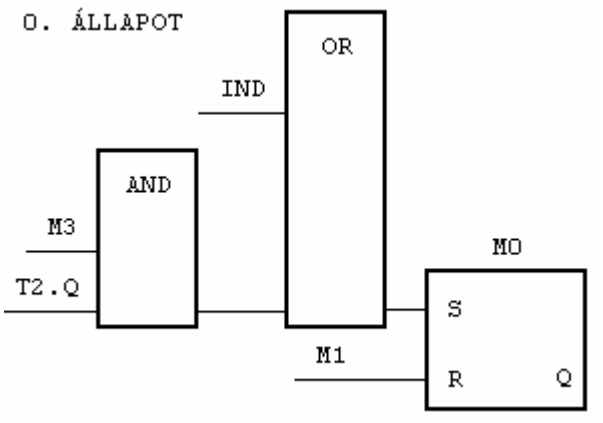


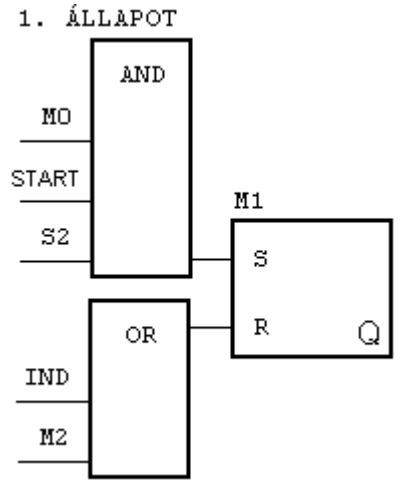
Az állapotgráf funkciótervbe történő átírásának szabályai

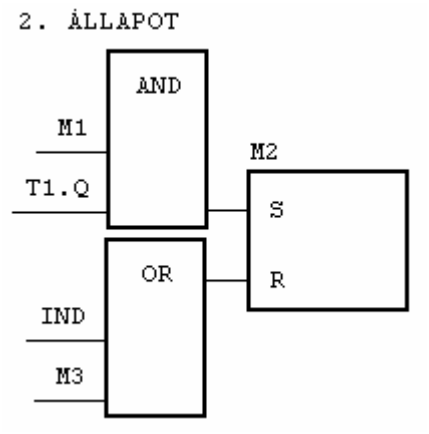
- Minden állapothoz hozzárendelünk egy RS-tárolót.
- Ha a tárolókat nem-remanens memóriaterületre címezzük, elegendő a 0. állapotot (**M0**) az ún. indító impulzussal beállítani, a többi tároló értéke az újraindításkor úgy is törlődik. Ha remanens memóriaterületen tároljuk az állapotokat, az indító impulzus segítségével az összes állapot tárolóját (kivéve a 0. állapot) törölni („resetelni") kell!

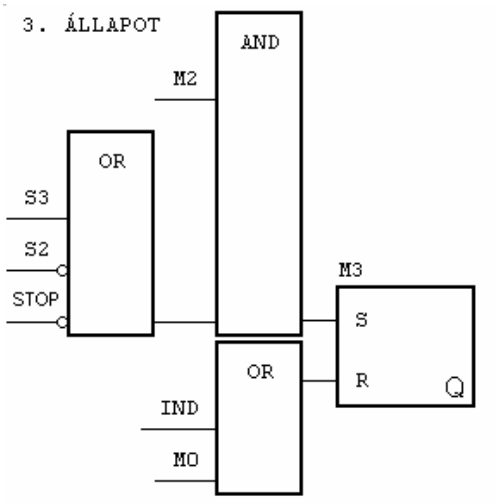
- Gondoskodnunk kell arról, hogy a vezérlésben mindig csak egy állapot legyen aktív. Ezt úgy tudjuk biztosítani, hogy az állapotok tárolóit (set-oldal) az őt megelőző állapot és az átváltás feltétele állítja be és az őt követő állapot törli (reset-oldal).
- Elágazás előtti állapotot az összes őt követő állapot törölheti (vagy-kapcsolat). Ha az elágazásban az átváltások feltételei egyszerre, egy időpillanatban teljesülhetnek, az ágak között prioritási sorrendet kell megállapítanunk. A magasabb prioritású állapot reteszeli az elágazásban lévő nála alacsonyabb prioritású állapotokat.
- Ha két állapot hurokba kerül, a hurokban lévő állapotokat az őket követő állapot és az átváltás feltétele együttesen törlik.
- A kimenetek azon állapotok VAGY-kapcsolataként írhatók fel, amelyekhez hozzárendeltük őket. Az esetleges plusz feltételeket (retesz feltételeket) ÉS feltételként hozzáillesztjük.

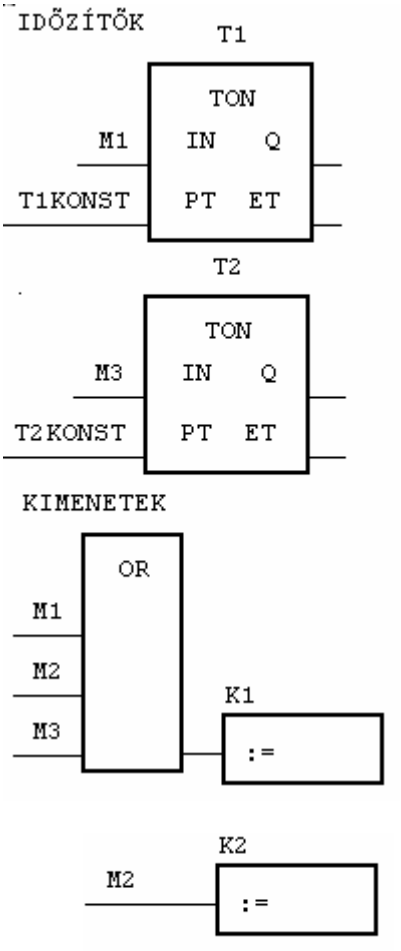
Funkcióterv











Utasításlista

```
PROGRAM PRVAGTLT
VAR
        START AT %I0.0.0.0.0:     BOOL;
        STOP AT %I0.0.0.0.1:      BOOL;
        RAMPA AT %I0.0.0.0.2:     BOOL;
        SULY AT %I0.0.0.0.3:      BOOL;
        MOTOR1 AT %Q0.0.0.0.0:    BOOL;
        MOTOR2 AT %Q0.0.0.0.1:    BOOL;
        FGVBL:      VAGTOLT;
        IMPULZUS:   BOOL;
END_VAR

        (*INDÍTÓ IMPULZUS*)

        LD      ISA
        PLC_Message
        ST      IMPULZUS

        (*FÜGGVÉNYBLOKK HÍVÁSA*)
```

```
        CAL   FGVBL(IND:=IMPULZUS, START:=START, STOP:=STOP, S2:=RAMPA,
        S3:=SULY)

        (*KIMENETEK TÁROLÁSA*)

        LD    FGVBL.K1
        ST    MOTOR1
        LD    FGVBL.K2
        ST    MOTOR2
        END_PROGRAM
```

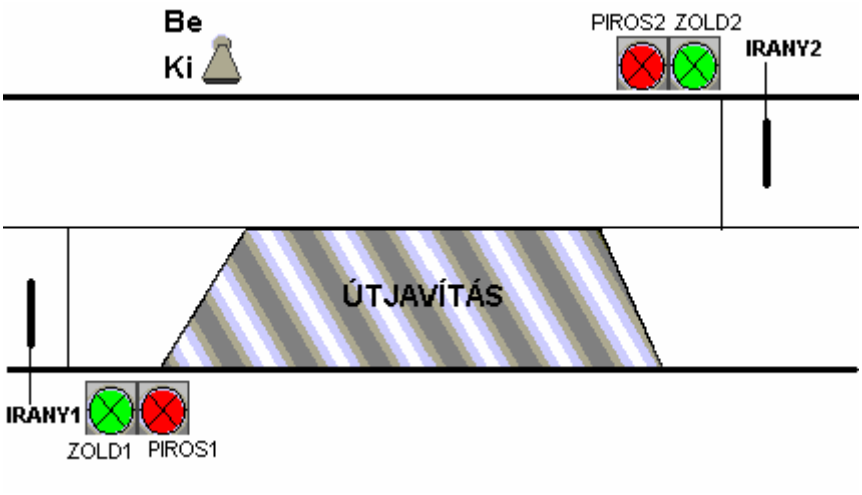
Az állapotgráfot függvényblokkban írtuk meg.

```
FUNCTION_BLOCK VAGTOLT

VAR_INPUT
        START:BOOL;
        STOP: BOOL;
        S2:   BOOL;
        S3:   BOOL;
        IND:  BOOL;
END_VAR

VAR_OUTPUT
        K1:   BOOL;
        K2:   BOOL;
END_VAR

VAR
        M0:   BOOL;
        M1:   BOOL;
        M2:   BOOL;
        M3:   BOOL;
        T1:   TON;
        T2:   TON;
END_VAR

VAR CONSTANT
        T1KONST:   TIME := T#3s;
        T2KONST:   TIME := t#5s;
END_VAR

        (*0. ÁLLAPOT*)
        LD    IND
        OR(   M3
        AND   T2.Q
        )
        S     M0
        LD    M1
        R     M0

(*1. ÁLLAPOT*)
LD    M0
AND   START
AND   S2
S     M1
LD    IND
OR    M2
R     M1
```

```
(*2. ÁLLAPOT*)
LD    M1
AND   T1.Q
S     M2
LD    IND
OR    M3
R     M2

(*3. ÁLLAPOT*)
LD    M2
AND   S3
ORN   S2
ORN   STOP
S     M3
LD    IND
OR    M0
R     M3
(*IDŐZÍTŐK*)

CAL   T1(IN := M1,PT :=T1KONST)
CAL   T2(IN := M3,PT :=T2KONST)

(*KIMENETEK*)

LD    M1
OR    M2
OR    M3
ST    K1

LD    M2
ST    K2

END_FUNCTION_BLOCK
```

Útjavítást jelző lámpa

Útjavítás miatt egy bekötőutat adott útszakaszon egysávosra kell szűkíteni. Mivel napközben igen nagy a forgalom, jelzőlámpákat állítottak fel a szűkítés végpontjain. A vezérlőberendezés bekapcsolásakor mindkét jelzőlámpa pirosat mutat. Ha az egyik irányú érzékelő jelez, a megfelelő lámpát 10s múlva zöldre váltja. A zöld fázist kb. 20s-ig tartani kell, mielőtt a másik érzékelő jelzése mindkét lámpát pirosra váltja. 10s múlva a másik irány lesz zöld. Ha egyik érzékelő sem jelez, a lámpajelzés az előző állapotában marad. A berendezést csak valamely irány zöldfázisa után lehet kikapcsolni.



37. ábra Útlezárás

Összerendelési táblázat

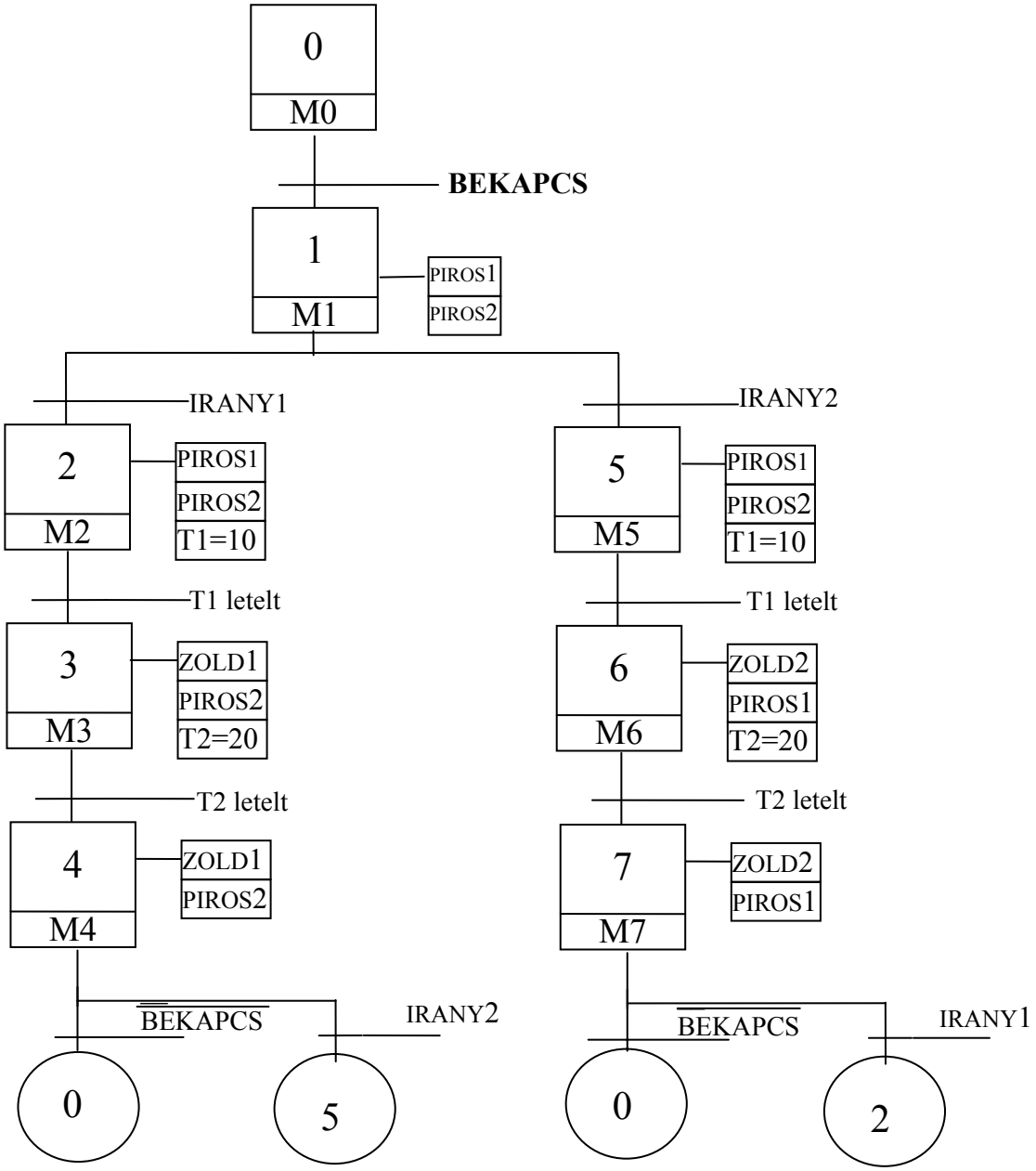
Bemenetek	Jel	Logikai összerendelés	Cím
BE/KI kapcsoló	BEKAPCS	bekapcsolva: BEKAPCS=1	I0.0
1. irány érzékelő	IRANY1	jelez: IRANY1=1	I0.1
2. irány érzékelő	IRANY2	jelez: IRANY2=1	I0.2
Kimenetek			
1. lámpa zöld	ZOLD1	világít: ZOLD1=1	Q0.0
2. lámpa zöld	ZOLD2	világít: ZOLD2=1	Q0.1
1. lámpa piros	PIROS1	világít: PIROS1=1	Q0.2
2. lámpa piros	PIROS2	világít: PIROS2=1	Q0.3

A vezérlés állapotai

1. Alapállapot: semmit sem működtet, a bekapcsolási jelre vár
2. Mindkét lámpa piros
3. Adott ideig mindkét lámpa piros, jármű vár az 1. irányból
4. Adott ideig mindkét lámpa piros, jármű vár a 2. irányból
5. Adott ideig 1. lámpa piros, 2. lámpa zöld
6. Adott ideig 2. lámpa piros, 1. lámpa zöld
7. lámpa piros, 2. lámpa zöld
8. lámpa piros, 1. lámpa zöld

Az állapotok közötti kapcsolatokat, az egyik állapotból a másikba történő átlépés feltételeit állapotgráf segítségével írjuk le.

Állapotgráf

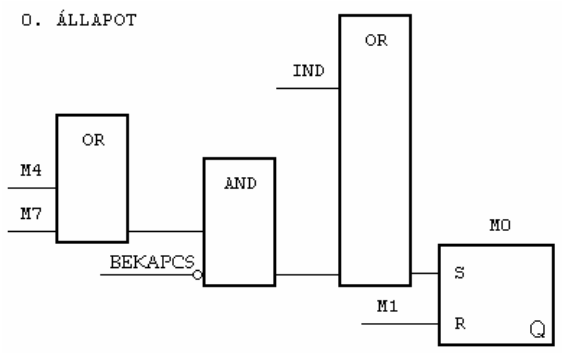


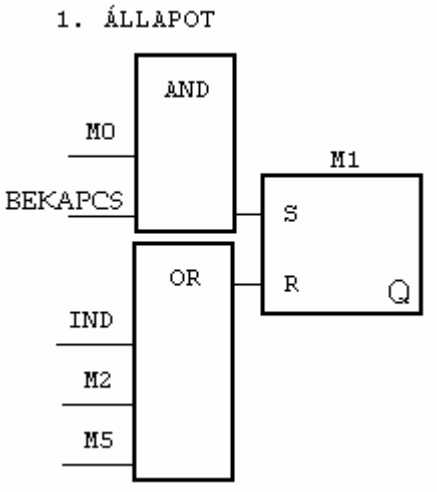
IRANY1 és **IRANY2** egyidejűleg is jelezhet. (Mindkét irányból érkezhet egyszerre jármű.) Mivel egyszerre csak egy állapot lehet „aktív" -egyszerre csak egy iránynak lehet zöld jelzése- , el kell dönteni, hogy melyik irány élvezzen elsőbbséget. Legyen **IRANY1**-nek elsőbbsége, ekkor **M2** reteszeli **M5**-t. (Az **5. állapotot** a **2.** RESET-eli.)

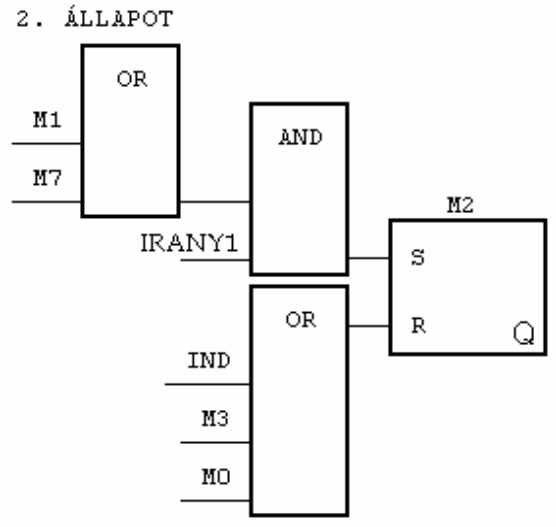
A **4.** és **7. állapot** után ismét kölcsönös reteszelés van. Itt a **0. állapot**nak van előnye. **M0** **M2**-t és **M5-**t is RESET-eli.

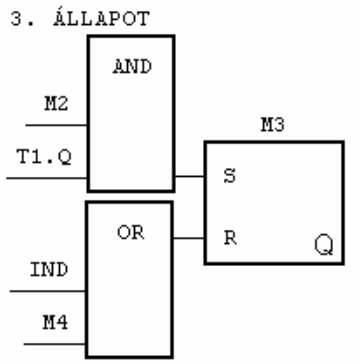
Az állapotgráf átírása funkciótervbe illetve utasításlistába

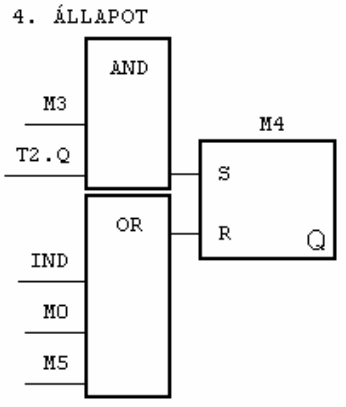
Funkcióterv

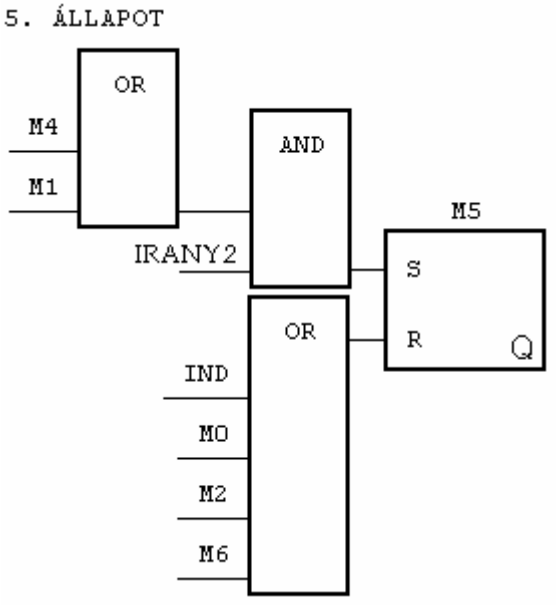


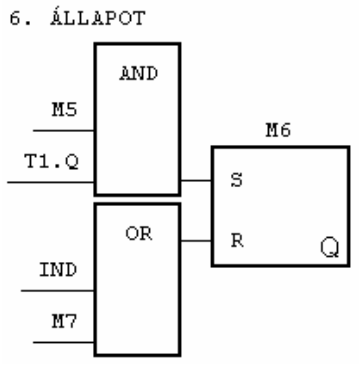


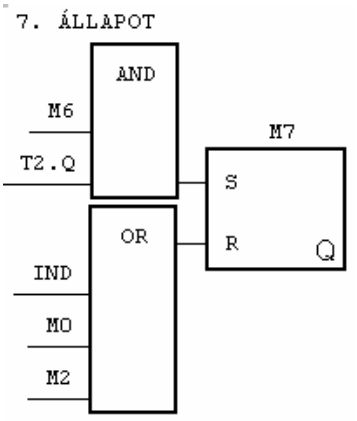


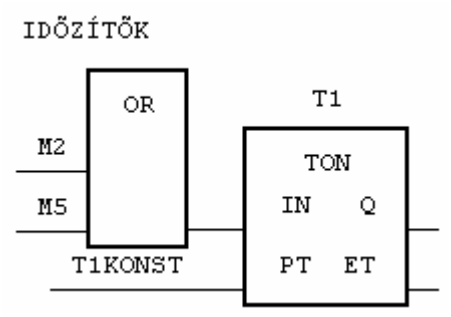


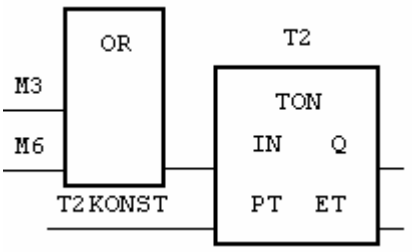


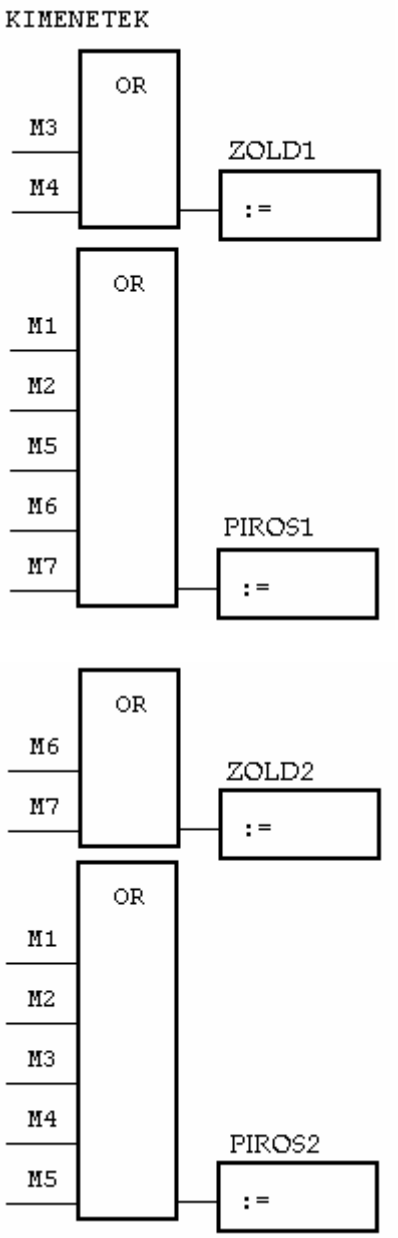



6. ÁLLAPOT
AND
M5
T1.Q
M6
S
R
Q
OR
IND
M7


7. ÁLLAPOT
AND
M6
T2.Q
M7
S
R
Q
OR
IND
M0
M2


IDŐZÍTŐK
OR
M2
M5
T1KONST
T1
TON
IN Q
PT ET


OR
M3
M6
T2KONST
T2
TON
IN Q
PT ET


KIMENETEK
OR
M3
M4
ZOLD1
:=
OR
M1
M2
M5
M6
M7
PIROS1
:=
OR
M6
M7
ZOLD2
:=
OR
M1
M2
M3
M4
M5
PIROS2
:=

Utasításlista

```
PROGRAM PRUTLEZAR

VAR_GLOBAL
        BEKAPCS AT %I0.0.0.0.0:  BOOL;
        IRANY1 AT %I0.0.0.0.1:   BOOL;
        IRANY2 AT %I0.0.0.0.2:   BOOL;
        ZOLD1 AT %Q0.0.0.0.0:    BOOL;
        PIROS1 AT %Q0.0.0.0.1:   BOOL;
        ZOLD2 AT %Q0.0.0.0.2:    BOOL;
        PIROS2 AT %Q0.0.0.0.3:   BOOL;
END_VAR

VAR
        FGVBL:      UTLEZ1;
        IMPULZUS:   BOOL;
END_VAR

         (*INDÍTÓ IMPULZUS*)
        LD      ISA
        PLC_Message
        ST      IMPULZUS

        (*FÜGGVÉNYBLOKK HÍVÁSA*)
        CAL     FGVBL(IND:=IMPULZUS)

        END_PROGRAM

FUNCTION_BLOCK UTLEZ1

VAR_INPUT
        IND:    BOOL;
END_VAR

VAR
        M0:     BOOL;
        M1:     BOOL;
        M2:     BOOL;
        M3:     BOOL;
        M4:     BOOL;
        M5:     BOOL;
        M6:     BOOL;
        M7:     BOOL;
        T1:     TON;
        T2:     TON;
END_VAR

VAR constant
        T1KONST:    TIME := t#10s;
```

```
        T2KONST:    TIME := t#20s;
END_VAR

VAR_EXTERNAL
        BEKAPCS:    BOOL;
        IRANY1:     BOOL;
        IRANY2:     BOOL;
        ZOLD1:      BOOL;
        PIROS1:     BOOL;
        ZOLD2:      BOOL;
        PIROS2:     BOOL;
END_VAR

        (*0. ÁLLAPOT*)
        LD      IND
        OR(     M4
        OR      M7
        ANDN    BEKAPCS
        )
        S       M0
        LD      M1
        R       M0

        (*1. ÁLLAPOT*)
        LD      M0
        AND     BEKAPCS
        S       M1
        LD      IND
        OR      M2
        OR      M5
        R       M1

        (*2. ÁLLAPOT*)
        LD      M1
        OR      M7
        AND     IRANY1
        S       M2
        LD      IND
        OR      M3
        OR      M0
        R       M2

        (*3. ÁLLAPOT*)
        LD      M2
        AND     T1.Q
        S       M3
        LD      IND
        OR      M4
        R       M3

        (*4. ÁLLAPOT*)
        LD      M3
        AND     T2.Q
        S       M4
        LD      IND
        OR      M0
        OR      M5
        R       M4

        (*5. ÁLLAPOT*)
        LD      M4
        OR      M1
        AND     IRANY2
        S       M5
        LD      IND
        OR      M0
        OR      M2
        OR      M6
        R       M5

        (*6. ÁLLAPOT*)
        LD      M5
        AND     T1.Q
        S       M6
        LD      IND
        OR      M7
        R       M6

        (*7. ÁLLAPOT*)
        LD      M6
```

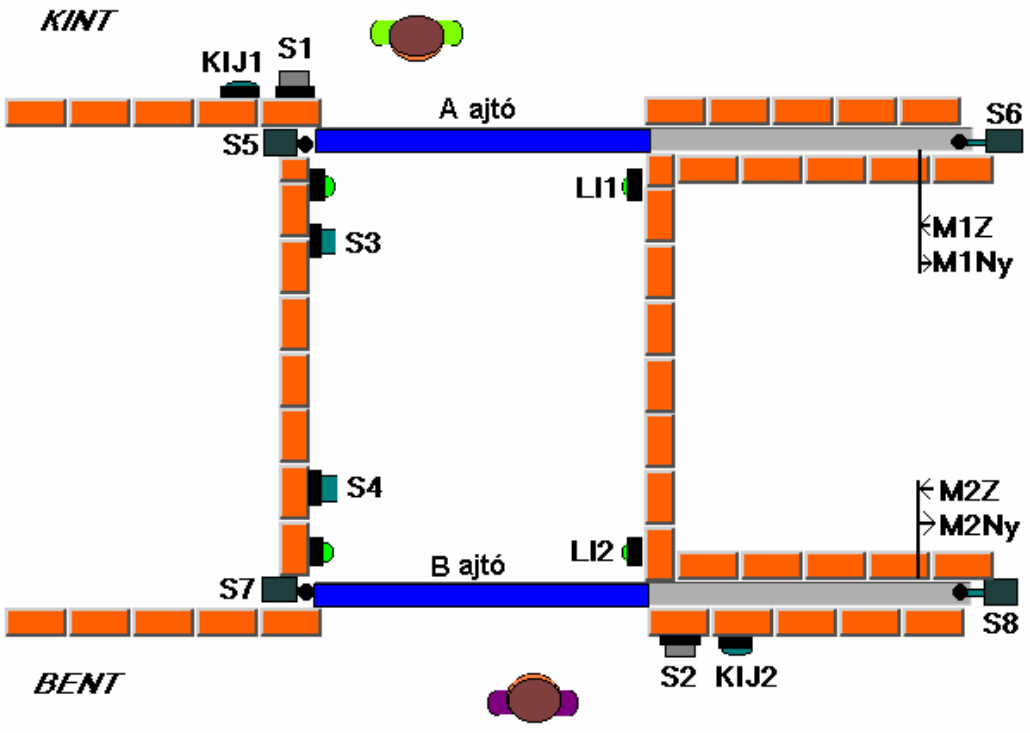
```
AND   T2.Q
S     M7
LD    IND
OR    M0
OR    M2
R     M7

(*IDŐZÍTŐK*)
LD    M2
OR    M5
ST    T1.IN
LD    T1KONST
ST    T1.PT
CAL   T1

LD    M3
OR    M6
ST    T2.IN
LD    T2KONST
ST    T2.PT
CAL   T2

(*KIMENETEK*)
LD    M3
OR    M4
ST    ZOLD1

LD    M1
      OR    M2
OR    M5
OR    M6
OR    M7
ST    PIROS1

LD    M6
OR    M7
      ST    ZOLD2

LD    M1
OR    M2
OR    M3
OR    M4
OR    M5
      ST    PIROS2
END_FUNCTION_BLOCK
```

Jelek állapotgráfon kívüli feldolgozása

Előfordul sok esetben, hogy a jelforrás, a jeladó, (pl. nyomógomb) csak rövid ideig szolgáltat jelet a PLC bemenetén. Ha a vezérlést állapotgráffal oldjuk meg, és a vezérlés éppen olyan állapotban van, amelynek követőállapota egy másik vezérlőjel megjelenésére vár, akkor a rövid időre megjelenő jelzést elveszíthetjük. Ennek elkerülésére **szükséges a rövid ideig ható jeleket az állapotgráfon kívül eltárolnunk**. (Ez az ún. **előfeldolgozás**.)

Zsilipajtók vezérlése

Egy helyiséget por- és szennyeződésmentesen kell tartani, így a bejárathoz kiépítettek egy zsilipkamrát, „**A**” és „**B**” tolóajtóval. Egyszerre, egy időben mindig csak az egyik ajtó lehet nyitva. A zsilipen az áthaladást kívülről az **S1** vagy **S2** nyomógombokkal lehet kezdeményezni. Kívülről belülre **S1** gomb megnyomása nyitja az „**A**” ajtót. Miután „**A**” kinyílt (S6 jelez), még 3 s-ig nyitva marad, majd becsukódik. Ha „A” ajtó becsukódott (S5 jelez), „**B**” ajtó automatikusan kinyílik, 3 s-ig nyitva marad, majd becsukódik. A másik irányból az áthaladás hasonló módon, fordított sorrendben, **S2** gomb megnyomására „**B**” majd „**A**” ajtó egymás után automatikusan nyílik és záródik. A nyomógomb melletti jelzőlámpa mutatja, hogy a vezérlés észrevette a gomb benyomását. Mindkét ajtónál végállás-kapcsolók jelzik az ajtó nyitott ill. zárt helyzetét. Az ajtókhoz tartozik egy-egy optikai érzékelő, ha a fényút megszakad, az ajtó zárását nem lehet megkezdeni, illetve, ha már záródik, azonnal vissza kell nyitni. Hasonlóan zárás közben vissza kell nyitni az ajtókat akkor is, ha az „**A**” ajtónál **S1** vagy **S3**, a „**B**” ajtónál **S2** vagy **S4** gombot megnyomják. 3s-os várakozás után a zárás ismét kezdeményezhető. A zsilipben a biztonság miatt elhelyeztek két nyomógombot, (**S3** és **S4**), amelyekkel a hozzájuk tartozó ajtók nyitását szükség esetén kezdeményezni lehet (pl. ha valaki véletlenül bennrekedt a zsilipben, mert úgy lépett be egy éppen nyitott zsilipajtón, hogy előzőleg nem nyomta meg az **S1** vagy **S2** gombot.)

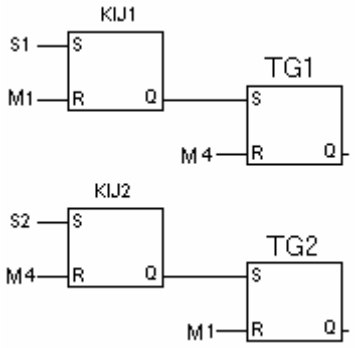


38. ábra Zsilipajtó

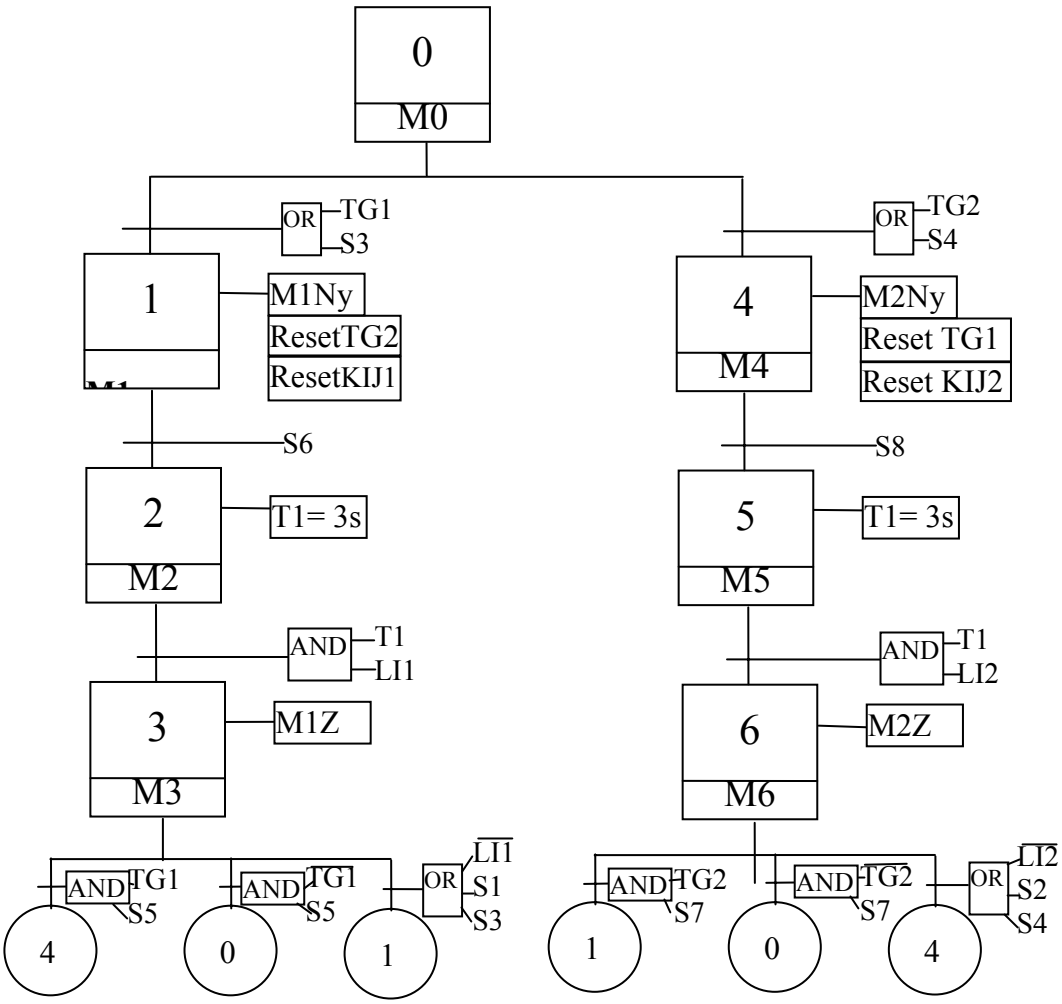
Összerendelési táblázat

Bemenetek	Jel	Logikai összerendelés	Cím
A ajtó külső nyomógomb	S1	**benyomva:** S1=1	I0.0
B ajtó külső nyomógomb	S2	benyomva: S2=1	I0.1
A ajtó belső nyomógomb	S3	benyomva: S3=1	I0.2
B ajtó belső nyomógomb	S4	benyomva: S4=1	I0.3
A ajtó csukva	S5	jelez, ha: S5=1	I0.4
A ajtó nyitva	S6	jelez, ha: S6=1	I0.5
B ajtó csukva	S7	jelez, ha: S7=1	I0.6
B ajtó nyitva	S8	jelez, ha: S8=1	I0.7
A ajtó optikai érzékelő	LI1	ha a fényút megszakad: LI1=0	I1.0
B ajtó optikai érzékelő	LI2	ha a fényút megszakad: LI2=0	I1.1
Kimenetek			
A ajtó elektromotor nyitás irányba	M1Ny	működtetve: M1Ny=1	Q0.0
A ajtó elektromotor zárás irányba	M1Z	működtetve: M1Z=1	Q0.1
B ajtó elektromotor nyitás irányba	M2Ny	működtetve: M2Ny=1	Q0.2
B ajtó elektromotor zárás irányba	M2Z	működtetve: M2Z=1	Q0.3
A oldali visszajelző lámpa	KIJ1	világít, ha: KIJ1=1	Q0.4
B oldali visszajelző lámpa	KIJ2	világít, ha: KIJ2=1	Q0.5

Rövid ideig ható jelek feldolgozása az állapotgráfon kívül



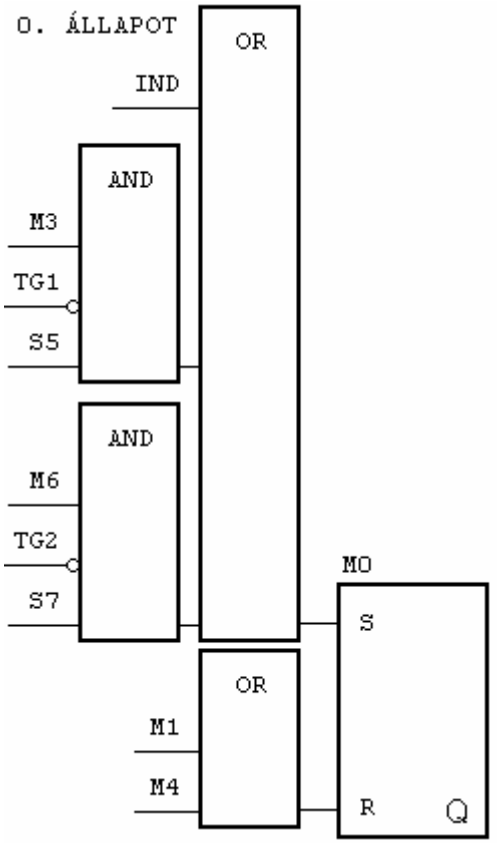
Az állapotgráf

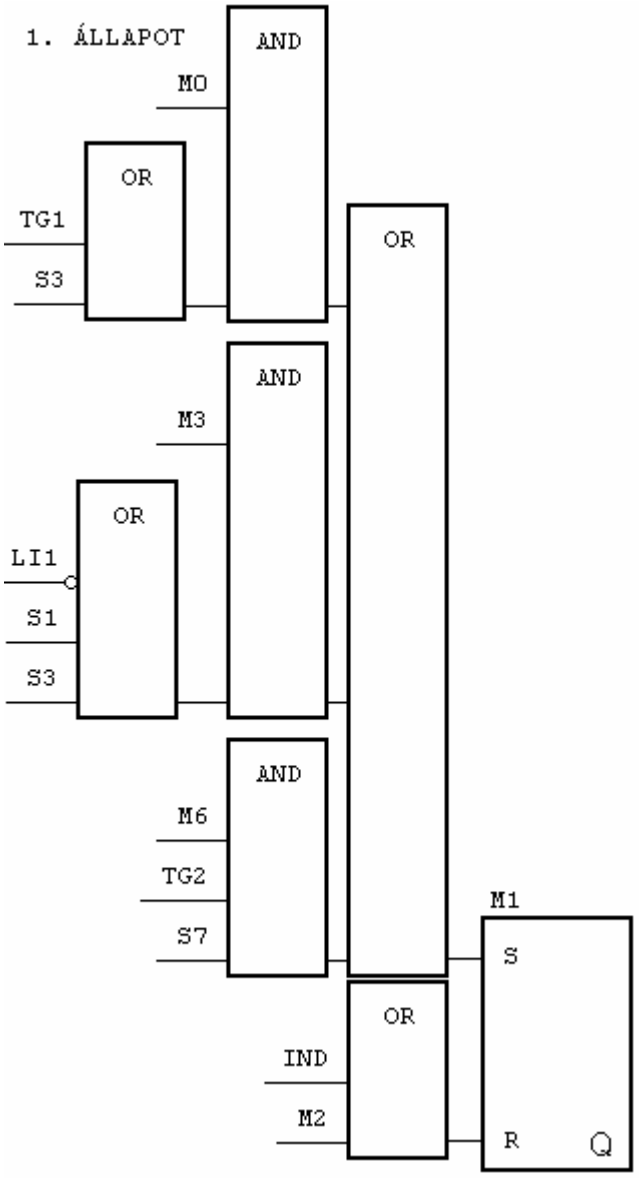


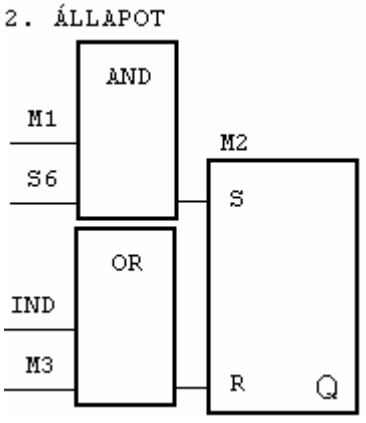
Az állapotgráf átírása funkciótervbe illetve utasításlistába

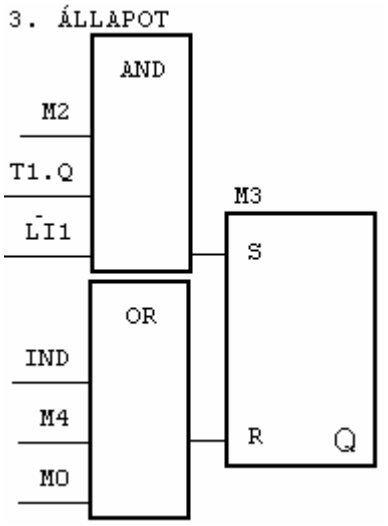
Az állapotgráf elágazásokat tartalmaz. Az elágazásokra vonatkozó szabályok figyelembevételével, rövid elemzés után észrevehetjük, hogy elegendő **M1 - M4** (vagyis a belépés – kilépés) közötti elsőbbséget meghatározni, ezáltal mindhárom elágazás problémája megoldódik. Legyen pl. a belépésnek elsőbbsége: ekkor **M1** reteszeli **M4**-et.

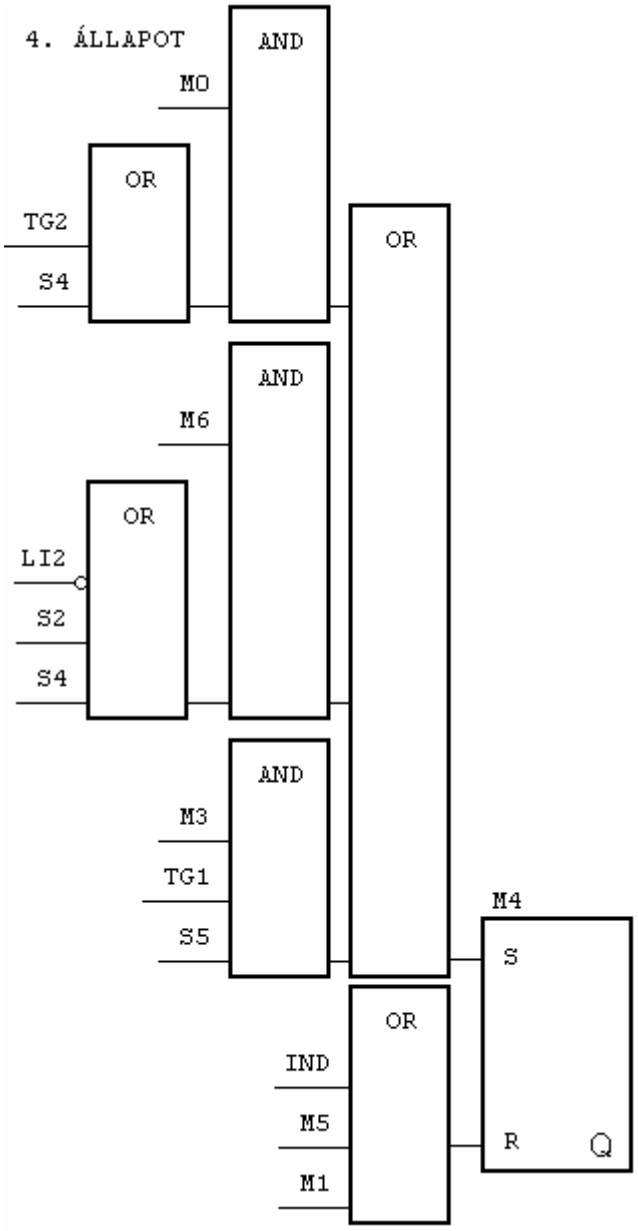
Funkcióterv

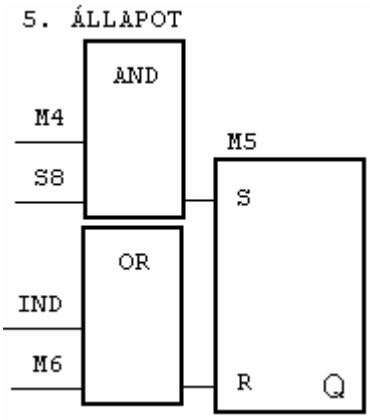


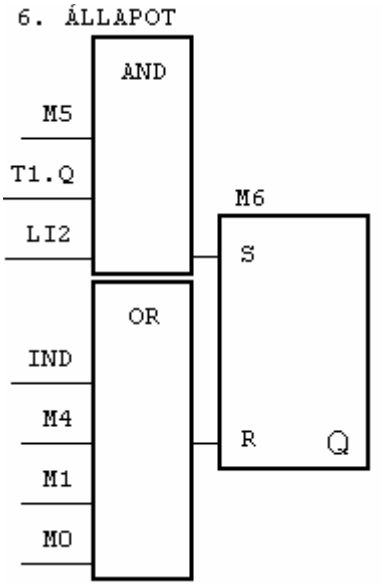

1. ÁLLAPOT
AND
M0
OR
TG1
S3
OR
AND
M3
OR
LI1
S1
S3
AND
M6
TG2
S7
M1
S
OR
IND
M2
R
Q

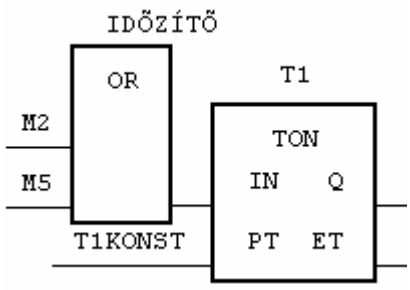

2. ÁLLAPOT
AND
M1
S6
M2
S
OR
IND
M3
R
Q

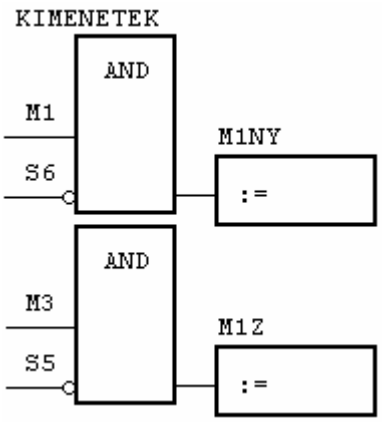

3. ÁLLAPOT
AND
M2
T1.Q
LI1
M3
S
OR
IND
M4
MO
R
Q

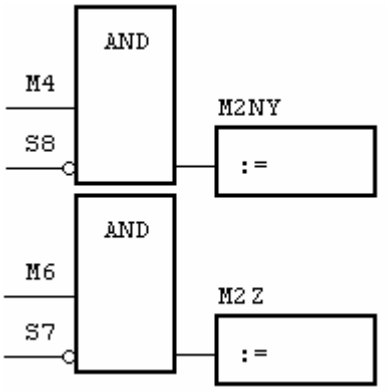

4. ÁLLAPOT
AND
MO
OR
TG2
S4
OR
AND
M6
OR
LI2
S2
S4
AND
M3
TG1
S5
M4
S
OR
IND
M5
M1
R
Q


5. ÁLLAPOT
AND
M4
S8
M5
S
OR
IND
M6
R
Q


6. ÁLLAPOT
AND
M5
T1.Q
LI2
M6
S
OR
IND
M4
M1
M0
R
Q


IDŐZÍTŐ
OR
T1
M2
M5
TON
IN
Q
T1KONST
PT
ET


KIMENETEK
AND
M1
S6
M1NY
:=
AND
M3
S5
M1Z
:=



A függvényblokk listája

```
FUNCTION_BLOCK ZSLPALL

VAR_INPUT
        S1:     BOOL;
        S2:     BOOL;
        S3:     BOOL;
        S4:     BOOL;
        S5:     BOOL;
        S6:     BOOL;
        S7:     BOOL;
        S8:     BOOL;
        LI1:    BOOL;
        LI2:    BOOL;
        TG1:    BOOL;
        TG2:    BOOL;
        IND:    BOOL;
END_VAR

VAR_OUTPUT
        M1NY:           BOOL;
        M1Z:    BOOL;
        M2NY:           BOOL;
        M2Z:    BOOL;
        M1:     BOOL;
        M4:     BOOL;
END_VAR

VAR
        M0:     BOOL;
        M2:     BOOL;
        M3:     BOOL;
        M5:     BOOL;
        M6:     BOOL;
        T1:     TON;
END_VAR
```

```
VAR constant
        T1KONST:    TIME := t#3s;
END_VAR

        (*0. ÁLLAPOT*)

        LD      IND
        OR(     M3
        ANDNTG1
        AND     S5
        )
        OR(     M6
        ANDNTG2
        AND     S7
        )
        S       M0
        LD      M1
        OR      M4
        R       M0

        (*1. ÁLLAPOT*)
        LD      M0
        AND(    TG1
        OR      S3
        )
        OR(     M3
        AND(    LI1
        NOT
        OR      S1
        OR      S3
        )
        OR(     M6
        AND     TG2
        AND     S7
        )
        )
        S       M1
        LD      IND
        OR      M2
        R       M1

        (*2. ÁLLAPOT*)
        LD      M1
        AND     S6
        S       M2
        LD      IND
        OR      M3
        R       M2

        (*3. ÁLLAPOT*)

        LD      M2
        AND     T1.Q
        AND     LI1
        S       M3
        LD      IND
        OR      M4
        OR      M0
        R       M3

        (*4. ÁLLAPOT*)

        LD      M0
        AND(    TG2
        OR      S4
        )
        OR(     M6
        AND(    LI2
        NOT
        OR      S2
        OR      S4
        )
        OR(     M3
        AND     TG1
        AND     S5
        )
        )
        S       M4
        LD      IND
        OR      M5
        OR      M1
        R       M4

        (*5. ÁLLAPOT*)

        LD      M4
        AND     S8
        S       M5
        LD      IND
        OR      M6
        R       M5

        (*6. ÁLLAPOT*)

        LD      M5
        AND     T1.Q
```

```
        AND   LI2
        S     M6
        LD    IND
        OR    M4
        OR    M1
        OR    M0
        R     M6

        (*IDŐZÍTŐ*)

        LD    M2
        OR    M5
        ST    T1.IN
        LD    T1KONST
        ST    T1.PT
        CAL   T1

        (*KIMENETEK*)

        LD    M1
        ANDN  S6
        ST    M1NY

        LD    M3
        ANDN  S5
        ST    M1Z

        LD    M4
        ANDN  S8
        ST    M2NY

        LD    M6
        ANDN  S7
        ST    M2Z

        END_FUNCTION_BLOCK
```

A főprogram listája

```
PROGRAM przsilip
VAR
        A_KULSO_NYG AT %I0.0.0.0.0:   BOOL;
        B_KULSO_NYG AT %I0.0.0.0.1:   BOOL;
        A_BELSO_NYG AT %I0.0.0.0.2:   BOOL;
        B_BELSO_NYG AT %I0.0.0.0.3:   BOOL;
        A_ZARVA AT %I0.0.0.0.4: BOOL;
        A_NYITVA AT %I0.0.0.0.5:      BOOL;
        B_ZARVA AT %I0.0.0.0.6: BOOL;
        B_NYITVA AT %I0.0.0.0.7:BOOL;
        A_OPT AT %I0.0.0.1.0:     BOOL;
        B_OPT AT %I0.0.0.1.1:     BOOL;
        A_NYITAS AT %Q0.0.0.0.0:      BOOL;
        A_ZARAS AT %Q0.0.0.0.1: BOOL;
        B_NYITAS AT %Q0.0.0.0.2:      BOOL;
        B_ZARAS AT %Q0.0.0.0.3: BOOL;
        KIJ1 AT %Q0.0.0.0.4:      BOOL;
        KIJ2 AT %Q0.0.0.0.5:      BOOL;
        TG1:   BOOL;
        TG2:   BOOL;
        IMPULZUS:  BOOL;
        FGVBL:     ZSLPALL;
END_VAR
```

```
(*INDÍTÓ IMPULZUS*)
LD      ISA
PLC_Message
ST      IMPULZUS

(*JELFELDOLGOZÁS AZ ÁLLAPOTGRÁFON KÍVÜL*)
LD      A_KULSO_NYG
S       KIJ1
LD      FGVBL.M1
R       KIJ1
LD      KIJ1
S       TG1
LD      FGVBL.M4
R       TG1

LD      B_KULSO_NYG
S       KIJ2
LD      FGVBL.M4
R       KIJ2
LD      KIJ2
S       TG2
LD      FGVBL.M1
R       TG2

(*FÜGGVÉNYBLOKK HÍVÁSA*)

CAL     FGVBL(IND:=IMPULZUS,
        S1:=A_KULSO_NYG,
        S2:=B_KULSO_NYG,
        S3:=A_BELSO_NYG,
        S4:=B_BELSO_NYG,
        S5:=A_ZARVA,
        S6:=A_NYITVA,
        S7:=B_ZARVA,
        S8:=B_NYITVA,
        LI1:=A_OPT,
        LI2:=B_OPT,
        TG1:=TG1,
        TG2:=TG2
        )

(* KIMENETEK *)
LD      FGVBL.M1NY
ST      A_NYITAS
LD      FGVBL.M2NY
ST      B_NYITAS
LD      FGVBL.M1Z
ST      A_ZARAS
LD      FGVBL.M2Z
ST      B_ZARAS
END_PROGRAM
```

Gyakorló feladat: utasításlista elemzése IV.

Írja át az alábbi utasításlistát funkciótervbe, majd próbálja meg felrajzolni az állapotgráfot.

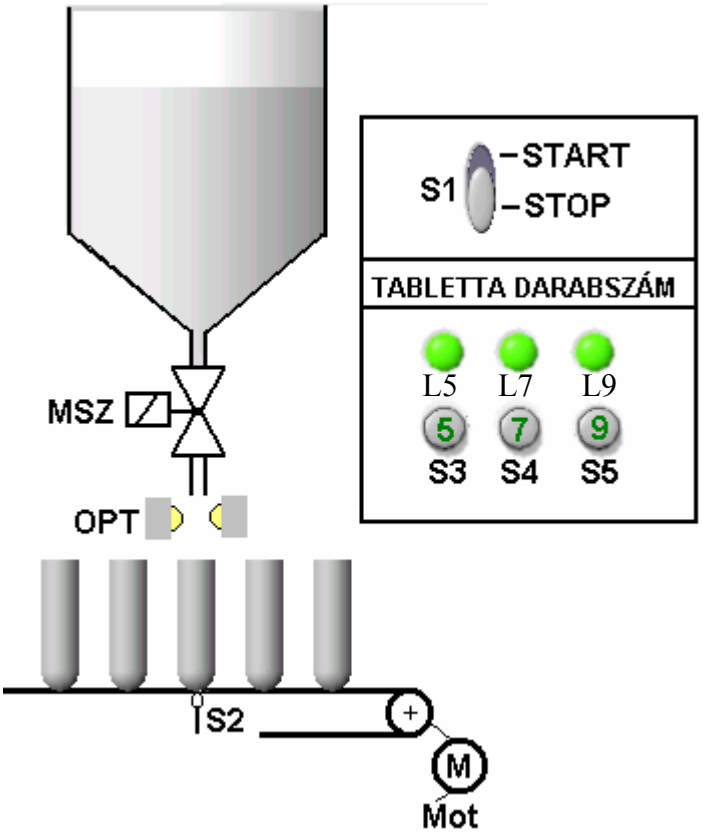
```
PROGRAM UTLELEM
VAR
        M60 : BOOL ;
        M61 : BOOL ;
        M0 : BOOL ;
        M1 : BOOL ;
        M2 : BOOL ;
        M3 : BOOL ;
        M4 : BOOL ;
        M5 : BOOL ;
        M6 : BOOL ;
END_VAR
VAR
        I1 AT %I0.1 : BOOL ;
        I2 AT %I0.2 : BOOL ;
        I3 AT %I0.3 : BOOL ;
        I4 AT %I0.4 : BOOL ;
        Q1 AT %Q0.1 : BOOL ;
        Q2 AT %Q0.2 : BOOL ;
        Q3 AT %Q0.3 : BOOL ;
END_VAR

LDN     M60
ST      M61
S       M60

LD      M61
OR(     I1
NOT
AND(    M3
OR      M5
OR      M6
)
)
S       M0
LD      M1
R       M0

LD      M0
AND     I1
S       M1
LD      M2
OR      M4
OR      M6
R       M1

LD      M1
AND     I2
ANDN    I4
S       M2
LD      M3
R       M2
LD      M2
AND     I3
S       M3
LD      M0
OR      M4
R       M3
LD      M1
ANDN    I2
AND     I4
OR(     M3
AND     I4
)
S       M4
LD      M5
OR      M0
R       M4
LD      M4
AND     I3
S       M5
LD      M0
OR      M6
R       M5
LD      M1
AND     I1
ANDN    I2
ANDN    I4
OR(     M5
ANDN    I2
)
S       M6
LD      M0
R       M6

LD      M1
OR      M2
OR      M3
OR      M4
OR      M6
ST      Q1
LD      M2
OR      M3
ST      Q2
LD      M4
OR      M5
ST      Q3
END_PROGRAM
```

Komplex vezérlési feladat számlálóval

A számlálókat használó vezérlési feladatokban is szisztematikusabb feladatmegoldást jelent az állapotgráf bevezetése.

Tablettaadagoló berendezés vezérlése

Egy előtárolóból bizonyos számú tablettát kell kémcsövekbe adagolni. A bekapcsolás után a kezelő a megfelelő nyomógomb működtetésével kiválasztja a kívánt darabszámot. A szalagmotor bekapcsol, és a kémcsövet töltési pozícióba továbbítja (**S2** jelez). A motor kikapcsol és a mágnesszelep nyitásával a tabletták beleesnek a kémcsőbe. A tablettákat az optikai érzékelő segítségével számolhatjuk. A megfelelő számú tabletta beesése után a szelep zár, a szalag továbbmegy és a következő kémcsőt hozza töltési helyzetbe. Ez a műveletsor ismétlődik, amíg a berendezést le nem állítják. A kívánt darabszámot működés közben is bármikor megváltoztathatják a kiválasztó nyomógomb benyomásával. A változtatás csak a következő kémcsőre érvényes, az éppen töltés alatt lévő kémcsőbe még annyi tabletta kerül, amennyivel a töltése kezdődött. A kikapcsoláskor az éppen folyamatban lévő adagolást még befejezi a vezérlés.



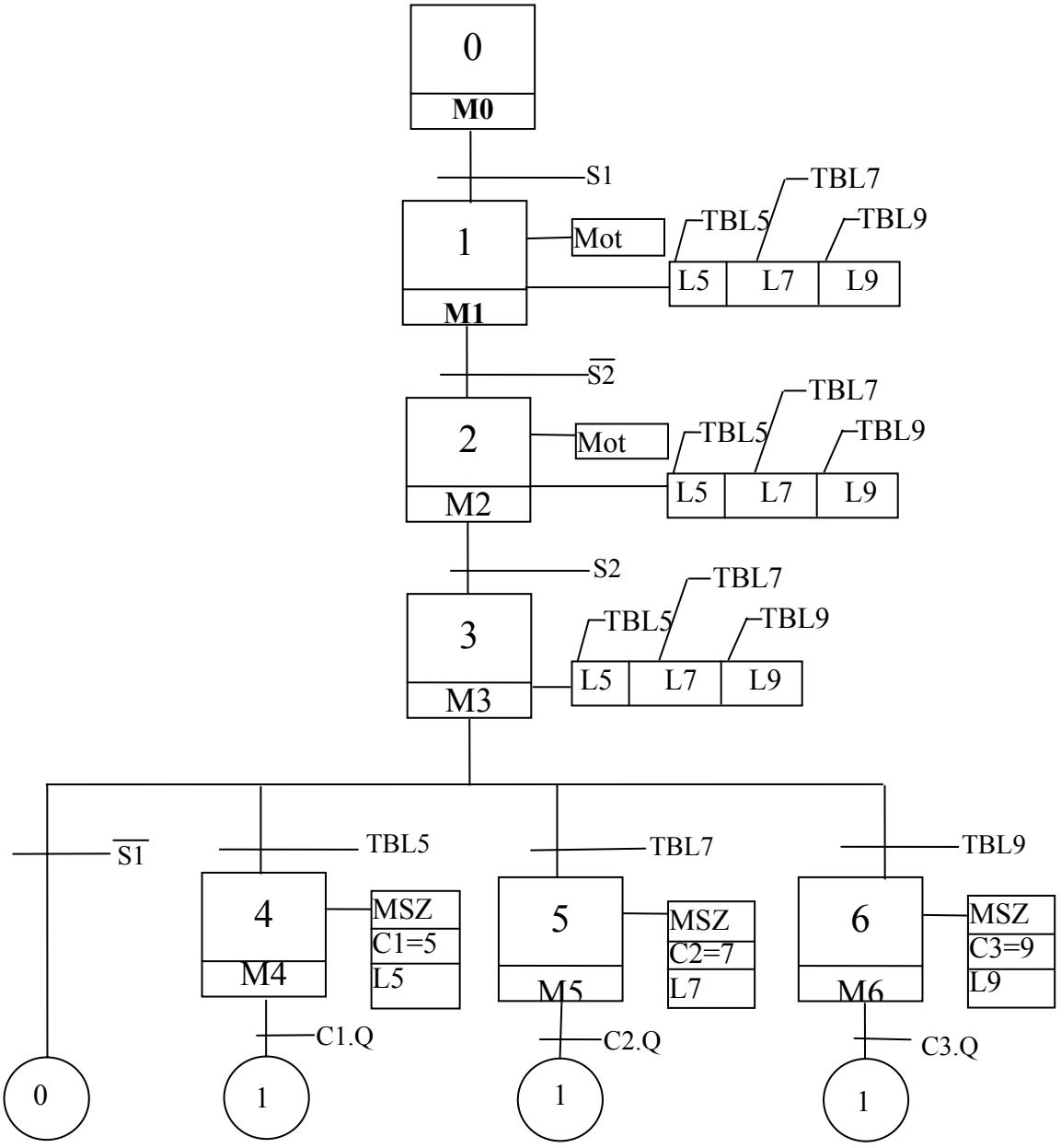
39. ábra Tablettaadagoló berendezés

A megoldásban a tabletta darabszámok kölcsönösen reteszelik egymást. Az egyik darabszámról a másikra átváltást mindig biztosítani kell, ezért szükséges a darabszámok tárolása, ezt RS – tárolókkal, ún. jelelőfeldolgozással, az állapotgráfon kívül valósítjuk meg. A készülék kikapcsolása törli a tárolót.

Összerendelési táblázat

Bemenetek	Jel	Logikai összerendelés	Cím
BE/KI kapcsoló	S1	**bekapcsolva:** S1=1	I0.0
A kémcső töltési pozícióban	S2	jelez, ha: S2=1	I0.1
5 DB TBL kiválasztó	S3	benyomva: S3=1	I0.2
7 DB TBL kiválasztó	S4	benyomva: S4=1	I0.3
9 DB TBL kiválasztó	S5	benyomva: S5=1	I0.4
optikai érzékelő	OPT	jelez, ha: OPT=1	I0.5
Kimenetek			
Motor	Mot	bekapcsolva: Mot=1	Q0.0
Szelep	MSZ	nyitva, ha: MSZ=1	Q0.1
5 TBL jelzőlámpa	L5	világít. ha: L5=1	Q0.2
7 TBL jelzőlámpa	L7	világít. ha: L7=1	Q0.3
9 TBL jelzőlámpa	L9	világít. ha: L9=1	Q0.4

Az állapotgráf

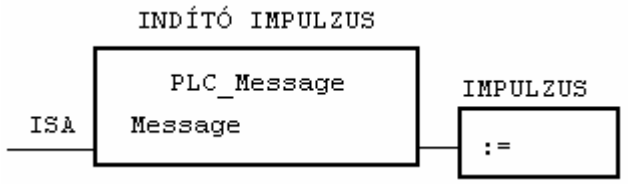


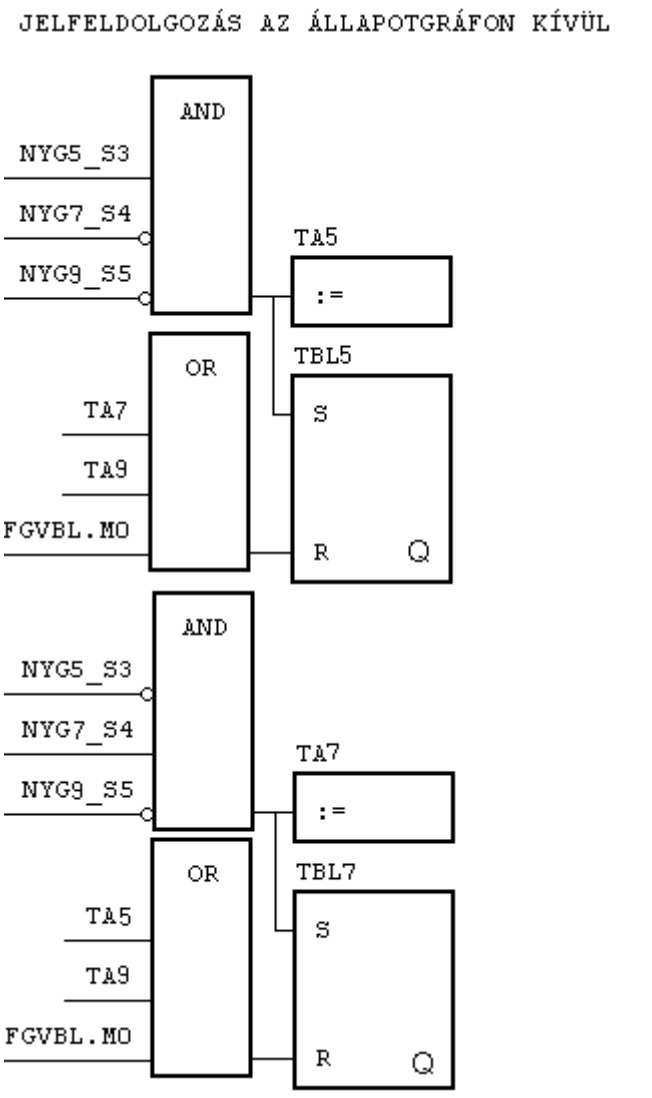
A vezérlőalgoritmus felépítése

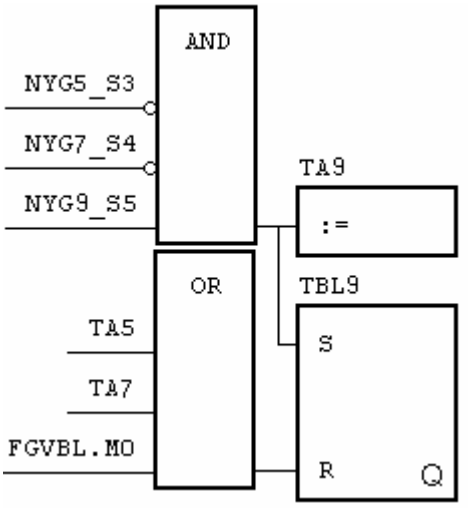
A vezérlést egy főprogram és egy függvényblokk segítségével valósítjuk meg. A főprogramban a ki/bemeneti jelek deklarálása mellett, még a függvényblokk-hívás előtt, feldolgozzuk a rövid ideig ható jeleket. Esetünkben ilyen jelnek számít a tabletta darabszám kiválasztó nyomógomb. A függvényblokk az állapotgráfnak felel meg. A számlálók és a kimenetek kiszámítása is a függvényblokkban történik, a kimenetek beállítását a főprogram végzi.

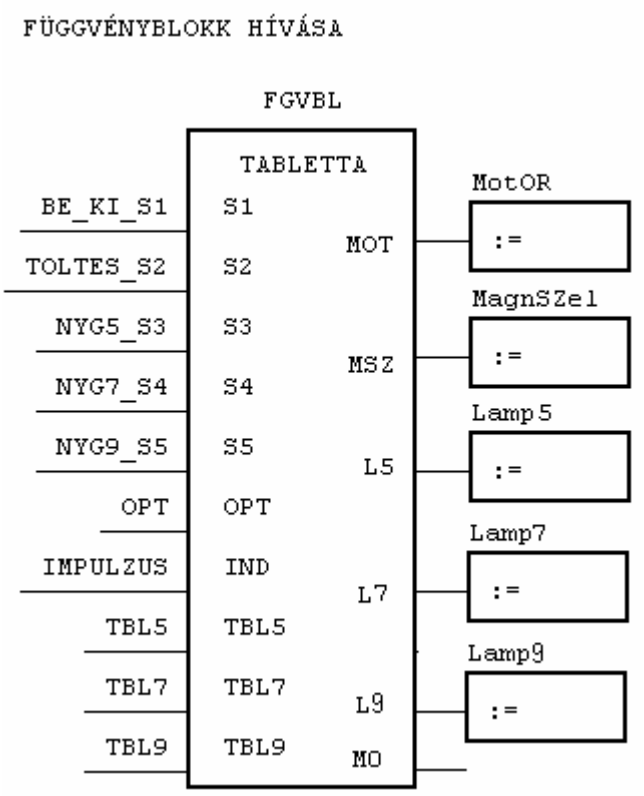
A főprogram

Első programciklus lekérdezése:









A funkcióterv átírása utasításlistába

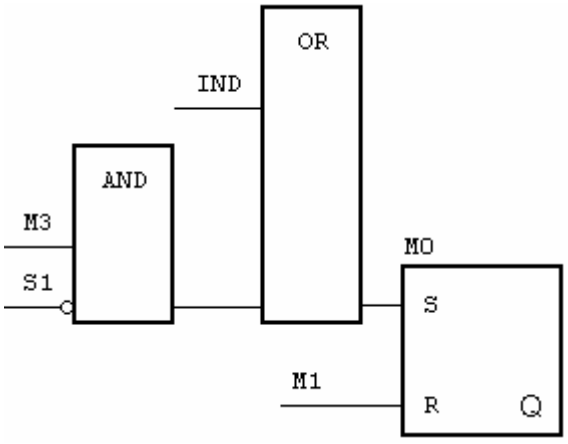
```
PROGRAM PRTABL
VAR
        BE_KI_S1 AT %I0.0.0.0.0:  BOOL;
        TOLTES_S2 AT %I0.0.0.0.1:         BOOL;
        NYG5_S3 AT %I0.0.0.0.2:   BOOL;
        NYG7_S4 AT %I0.0.0.0.3:   BOOL;
        NYG9_S5 AT %I0.0.0.0.4:   BOOL;
        OPT AT %I0.0.0.0.5: BOOL;
        MotOR AT %Q0.0.0.0.0:     BOOL;
        MagnSZel AT %Q0.0.0.0.1:  BOOL;
        Lamp5 AT %Q0.0.0.0.2:     BOOL;
```

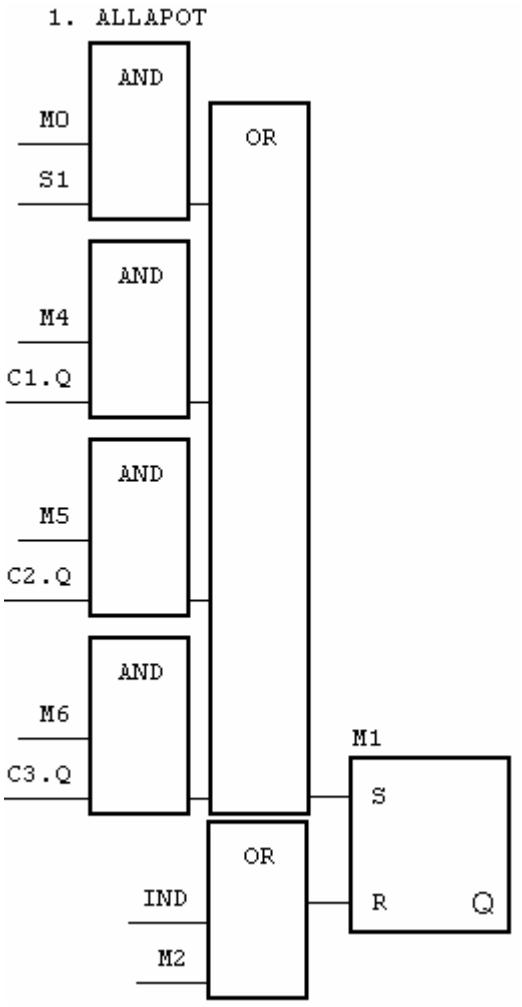
```
        Lamp7 AT %Q0.0.0.0.3:     BOOL;
        Lamp9 AT %Q0.0.0.0.4:     BOOL;
        TA5:   BOOL;
        TA7:   BOOL;
        TA9:   BOOL;
        TBL5: BOOL;
        TBL7: BOOL;
        TBL9: BOOL;
        IMPULZUS:  BOOL;
        FGVBL:       TABLETTA;
END_VAR

        (*INDÍTÓ IMPULZUS*)
        LD     ISA
        PLC_Message
        ST     IMPULZUS

        (*JELFELDOLGOZÁS AZ ÁLLAPOTGRÁFON KÍVÜL*)
        LD     NYG5_S3
        ANDN NYG7_S4
        ANDN NYG9_S5
        ST     TA5
        S      TBL5
        LD     TA7
        OR     TA9
        OR     FGVBL.M0
        R      TBL5

        LDN   NYG5_S3
        AND   NYG7_S4
        ANDN NYG9_S5
        ST     TA7
        S      TBL7
        LD     TA5
        OR     TA9
        OR     FGVBL.M0
        R      TBL7

        LDN   NYG5_S3
        ANDN NYG7_S4
        AND   NYG9_S5
        ST     TA9
        S      TBL9
        LD     TA5
        OR     TA7
        OR     FGVBL.M0
        R      TBL9
```

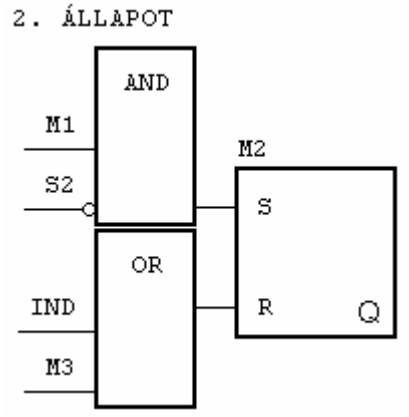
```
(*FÜGGVÉNYBLOKK HÍVÁSA*)

CAL   FGVBL(IND:=IMPULZUS,
      S1:=BE_KI_S1,
      S2:=TOLTES_S2,
      S3:=NYG5_S3,
      S4:=NYG7_S4,
      S5:=NYG9_S5,
      OPT:=OPT,
      TBL5:=TBL5,
      TBL7:=TBL7,
      TBL9:=TBL9
      )

(* KIMENETEK *)

LD    FGVBL.MOT
ST    MotOR

LD    FGVBL.MSZ
ST    MagnSZel

LD    FGVBL.L5
ST    Lamp5

LD    FGVBL.L7
ST    Lamp7

LD    FGVBL.L9
ST    Lamp9
END_PROGRAM
```

A függvényblokk

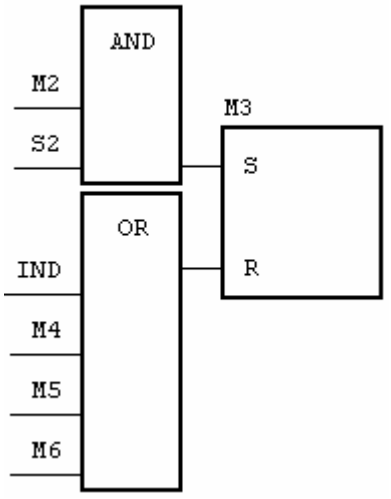
Funkcióterv



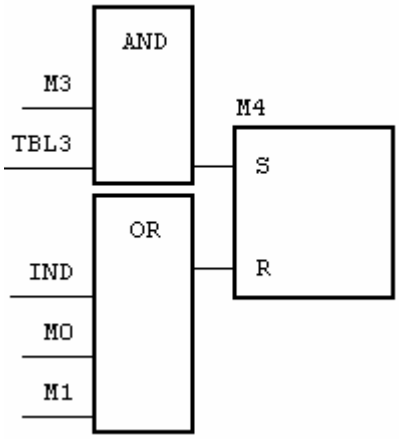

1. ALLAPOT
AND
M0
S1
AND
M4
C1.Q
AND
M5
C2.Q
AND
M6
C3.Q
OR
M1
S
R
Q
OR
IND
M2


2. ÁLLAPOT
AND
M1
S2
M2
S
OR
IND
R
Q
M3

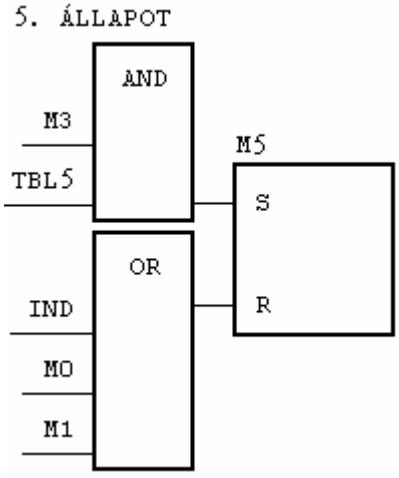
3. ÁLLAPOT

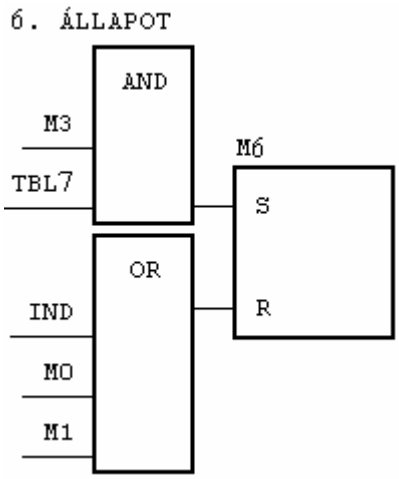


4. ÁLLAPOT

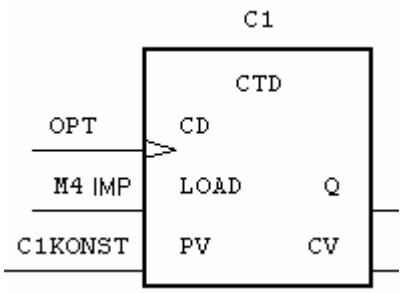


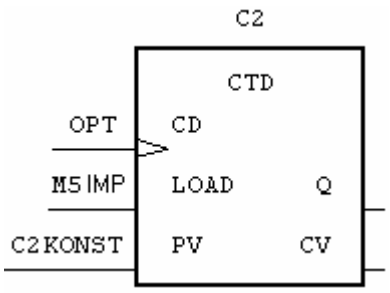
5. ÁLLAPOT

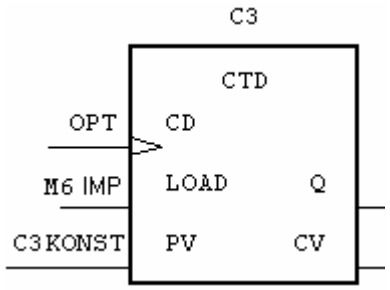


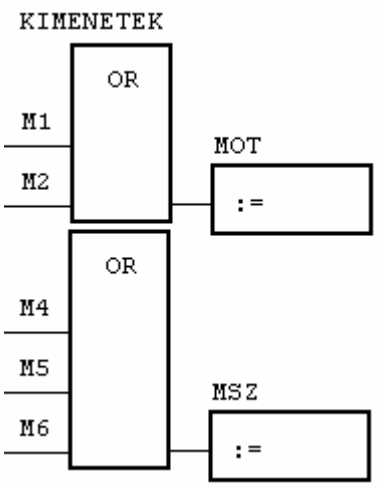

6. ÁLLAPOT
AND
M3
TBL7
M6
S
OR
IND
M0
M1
R

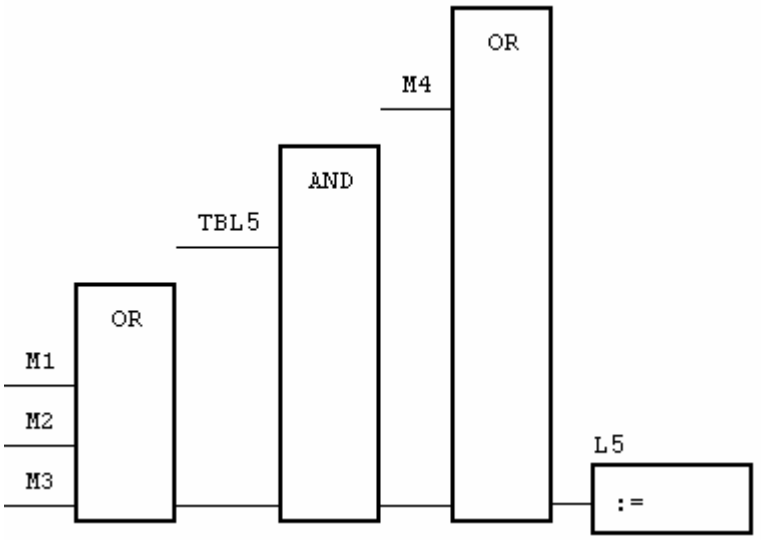
A számlálók:

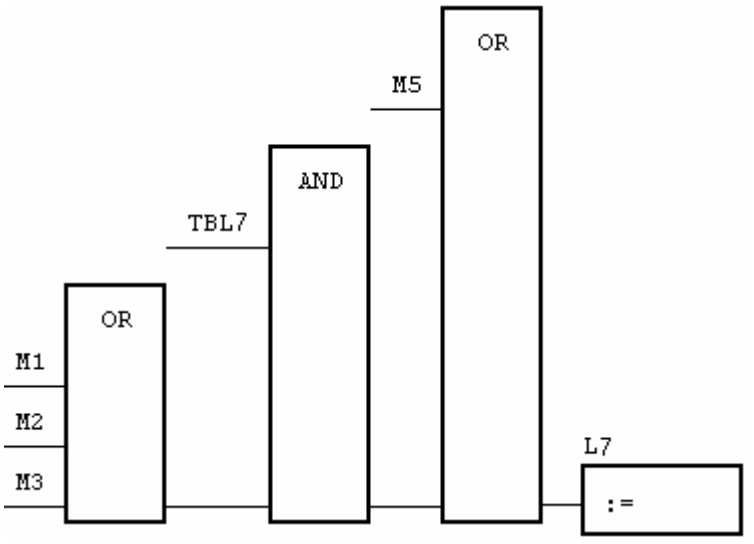

C1
CTD
OPT
CD
M4 IMP
LOAD
Q
C1KONST
PV
CV

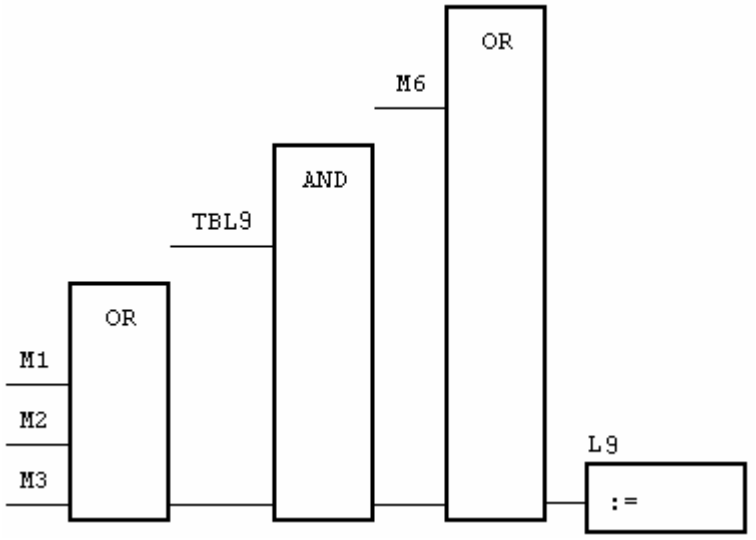

C2
CTD
OPT
CD
M5 IMP
LOAD
Q
C2 KONST
PV
CV


C3
CTD
OPT
CD
M6 IMP
LOAD
Q
C3 KONST
PV
CV


KIMENETEK
OR
M1
M2
MOT
:=
OR
M4
M5
M6
MSZ
:=


OR
M4
AND
TBL5
OR
M1
M2
M3
L5
:=


OR
M5
AND
TBL7
OR
M1
M2
M3
L7
:=



A függvényblokk utasításlista

```
FUNCTION_BLOCK TABLETTA
VAR_INPUT
        S1:     BOOL;
        S2:     BOOL;
        S3:     BOOL;
        S4:     BOOL;
        S5:     BOOL;
        OPT:    BOOL;
        IND:    BOOL;
        TBL5: BOOL;
        TBL7: BOOL;
        TBL9: BOOL;
END_VAR
VAR_OUTPUT
        MOT:  BOOL;
        MSZ:  BOOL;
        L5:     BOOL;
        L7:     BOOL;
        L9:     BOOL;
        M0:     BOOL;
END_VAR
VAR
        M1:     BOOL;
        M2:     BOOL;
        M3:     BOOL;
        M4:     BOOL;
        M5:     BOOL;
        M6:     BOOL;
        C1:     CTD;
END_VAR

VAR constant
        C1KONST:    INT := 5;
```

```
END_VAR

VAR
        C1IMP:          R_TRIG;
        C2:     CTD;
END_VAR

VAR constant
        C2KONST:    INT := 7;
END_VAR

VAR
        C2IMP:          R_TRIG;
        C3:     CTD;
END_VAR

VAR constant
        C3KONST:    INT := 9;
END_VAR

VAR
        C3IMP:          R_TRIG;
END_VAR

        (*0. ÁLLAPOT*)
        LD      IND
        OR(     M3
        ANDNS1
        )
        S       M0
        LD      M1
        R       M0

        (*1. ÁLLAPOT*)
        LD      M0
        AND     S1

        OR(     M4
        AND     C1.Q
        )
        OR(     M5
        AND     C2.Q
        )
        OR(     M6
        AND     C3.Q
        )
        S       M1
        LD      IND
        OR      M2
        R       M1

        (*2. ÁLLAPOT*)
        LD      M1
        ANDNS2
        S       M2
        LD      IND
        OR      M3
        R       M2

        (*3. ÁLLAPOT*)
        LD      M2
        AND     S2
        S       M3
        LD      IND
        OR      M4
        OR      M5
        OR      M6
        R       M3

        (*4. ÁLLAPOT*)
        LD      M3
        AND     TBL5
        S       M4
        LD      IND
        OR      M0
        OR      M1
        R       M4
```

```
(*5. ÁLLAPOT*)
LD    M3
AND   TBL7
S     M5
LD    IND
OR    M0
OR    M1
R     M5

(*6. ÁLLAPOT*)
LD    M3
AND   TBL9
S     M6
LD    IND
OR    M0
OR    M1
R     M6

(*SZÁMLÁLÓK*)
LD    OPT
ST    C1.CD
LD    M4
ST    C1IMP.CLK
CAL   C1IMP
LD    C1IMP.Q
ST    C1.LOAD
LD    C1KONST
ST    C1.PV
CAL   C1

LD    OPT
ST    C2.CD
LD    M5
ST    C2IMP.CLK
CAL   C2IMP
LD    C2IMP.Q
ST    C2.LOAD
LD    C2KONST
ST    C2.PV
CAL   C2

LD    OPT
ST    C3.CD
LD    M6
ST    C3IMP.CLK
CAL   C3IMP
LD    C3IMP.Q
ST    C3.LOAD
LD    C3KONST
ST    C3.PV
CAL   C3

(*KIMENETEK*)

LD    M1
OR    M2
ST    MOT

LD    M4
OR    M5
OR    M6
ST    MSZ

LD    M4
OR(   TBL5
AND(  M1
OR    M2
OR    M3
)
)
ST    L5

LD    M5
OR(   TBL7
AND(  M1
OR    M2
OR    M3
)
)
ST    L7

LD    M6
OR(   TBL9
AND(  M1
OR    M2
OR    M3
)
)
ST    L9
END_FUNCTION_BLOCK
```

Ütemvezérelt lefutóvezérlések

Közlekedési lámpa vezérlése

Egy közlekedési lámpa **3 időegység**ig **piros**, ahol a **3. időegységben** a piros mellett a **sárga** lámpa **is** világít. Ezután **4 időegység zöld** fázis következik. A ciklus **1 időegység sárgá**val zárul.
Az időegység legyen **5** s.

Összerendelési táblázat

Bemenetek	Jel	Logikai hozzárendelés	Cím
BE/KI kapcsoló	S0	bekapcsolva: S0=1	I0.0
Kimenetek			
Piros. lámpa	P	világít, ha: P=1	Q0.0
Sárga lámpa	S	világít, ha: S=1	Q0.1
Zöld. lámpa	Z	világít, ha: Z=1	Q0.2

Megoldás 1. változat

Egy ciklus 8 időegységből áll. Az időütemet egy ütemadó adja, amelynek igen egyszerű a programja: az időzítő kimenőjelével vezérelt UTEM-merker legyen 0 állapotú. A merker negáltját az indítójellel (S0) együtt egy bekapcsolás-késleltetéses időzítő IN bemenetéhez kapcsoljuk. Bekapcsoláskor elindul az időzítő, és a kimenetén 5s elteltével megjelenik az 1 jel, amely csak egyetlen ciklusideig „él”, mert a következő ciklusban az indítófeltétele nullára vált. Ez a ciklus ismétlődik mindaddig, amíg S0-al ki nem kapcsoljuk a vezérlést.
Az ütemgenerátor impulzusai egy számlálót inkrementálnak. Ha a számláló értéke eléri a 8-at, a számlálót RESET-elni kell. A számláló érétke megadja az ütemszámot, amely segítségével a kimenőjelek beállíthatók.

Ütem	Lámpafázisok			A számláló értéke	ütemmerker
1	**P**			0	M1
2	**P**			1	M2
3	**P**	**S**		2	M3
4			**Z**	3	M4
5			**Z**	4	M5
6			**Z**	5	M6
7			**Z**	6	M7
8		**S**		7	M8

Utasításlista

```
PROGRAM KZLAMPA
VAR
        S0 AT %I0.0.0.0.0:      BOOL;
        PIROS AT %Q0.0.0.0.0:       BOOL;
        SARGA AT %Q0.0.0.0.1:       BOOL;
        ZOLD AT %Q0.0.0.0.2:        BOOL;
        SZAMLALO:   CTU;
        UTEM_MAX:  INT := 8;
        T1:     TON;
        UTEMIDO:    TIME := t#5S;
        M1:     BOOL;
        M2:     BOOL;
        M3:     BOOL;
        M4:     BOOL;
        M5:     BOOL;
        M6:     BOOL;
        M7:     BOOL;
        M8:     BOOL;
        UTEM:BOOL;
        USZAM:      INT;
END_VAR
        (*ÜTEMGENERÁTOR*)
        LD      S0
        ANDN UTEM
        ST      T1.IN
        LD      UTEMIDO
        ST      T1.PT
        CAL     T1
        LD      T1.Q
        ST      UTEM

        (*SZÁMLÁLÓ*)
        LD      UTEM
        ST      SZAMLALO.CU
        LD      SZAMLALO.CV
        GE      UTEM_MAX
        ORN     S0
        ST      SZAMLALO.RESET
        CAL     SZAMLALO
        (*ÜTEM-MERKEREK*)
        LD      SZAMLALO.CV
        ST      USZAM
        EQ      0
        ST      M1
        LD      USZAM
        EQ      1
        ST      M2
        LD      USZAM
        EQ      2
        ST      M3
        LD      USZAM
        EQ      3
        ST      M4
        LD      USZAM
        EQ      4
        ST      M5
        LD      USZAM
        EQ      5
        ST      M6
        LD      USZAM
        EQ      6
        ST      M7
        LD      USZAM
        EQ      7
        ST      M8

        (*PIROS LÁMPA*)
        LD      M1
        OR      M2
        OR      M3
        ST      PIROS

        (*SÁRGA LÁMPA*)
        LD      M3
        OR      M8
        ST      SARGA

        (*ZÖLD LÁMPA*)
        LD      M4
        OR      M5
        OR      M6
        OR      M7
        ST      ZOLD
END_PROGRAM
```

Megoldás 2. változat, utasításlista

```
PROGRAM KZLAMP2
VAR
        S0 AT %I0.0.0.0.0:      BOOL;
        PIROS AT %Q0.0.0.0.0:           BOOL;
        SARGA AT %Q0.0.0.0.1:           BOOL;
        ZOLD AT %Q0.0.0.0.2:            BOOL;
        UTGEN AT %QB0.0.0.1:            BYTE;
        FELFUTO:        R_TRIG;
        FGVBL:          FBKLAMP;
        FGVBL2:         FBSARGA;
END_VAR

        CAL FELFUTO(CLK:=S0)
        LD      S0
        JMPCN           VILLOG
        LD      FELFUTO.Q
        ST      FGVBL.INDIMP
        CAL FGVBL
        LD      FGVBL.P
        ST      PIROS
        LD      FGVBL.SA
        ST      SARGA
        LD      FGVBL.Z
        ST      ZOLD
        LD      FGVBL.UTEMB
        ST      UTGEN
        RET

VILLOG:
CAL     FGVBL2
LD      FGVBL2.P
ST      PIROS
LD      FGVBL2.SA
ST      SARGA
LD      FGVBL2.Z
ST      ZOLD
LD      FGVBL2.UTEMB
ST      UTGEN
RET
END_PROGRAM

FUNCTION_BLOCK FBKLAMP

VAR_INPUT
        INDIMP:         BOOL;
END_VAR

VAR_OUTPUT
        P:      BOOL;
        SA:     BOOL;
        Z:      BOOL;
        UTEMB:          BYTE;
END_VAR

VAR
        T1:     TON;
        MB:     BYTE;
        UTIMP:          BOOL;
        UTEMIDO:        TIME := t#1S;
END_VAR

        LD      INDIMP
        JMPCN           TOVABB
        LD      1
        ST      MB

TOVABB:
(*ÜTEMGENERÁTOR*)
LDN     UTIMP
ST      T1.IN
```

```
        LD      UTEMIDO
        ST      T1.PT
        CAL     T1
        LD      T1.Q
        ST      UTIMP
        (*FORGATÁS*)
        LD      UTIMP
        JMPCN           TOV2
        LD      MB
        ROL     1
        ST      MB
        (*KIMENETEK BEÁLLÍTÁSA*)
        TOV2:
        LD      MB
        ST      UTEMB
        LD      0
        ST      Z
        ST      P
        ST      SA

        LD      MB
        EQ      4
        ST      SA

        LD      MB
        LE      4
        ST      P
        RETC

        LD      MB
        EQ      128
        ST      SA
        RETC

        LD      1
        ST      Z

        RET


                                END_FUNCTION_BLOCK
```

```
FUNCTION_BLOCK FBSARGA

VAR_OUTPUT
        P:      BOOL;
        SA:     BOOL;
        Z:      BOOL;
        UTEMB:          BYTE;
END_VAR

VAR
        T1:     TP;
        T2:     TP;
        UTIDO1:         TIME := t#100MS;
        UTIDO2:         TIME := t#500MS;
END_VAR
```

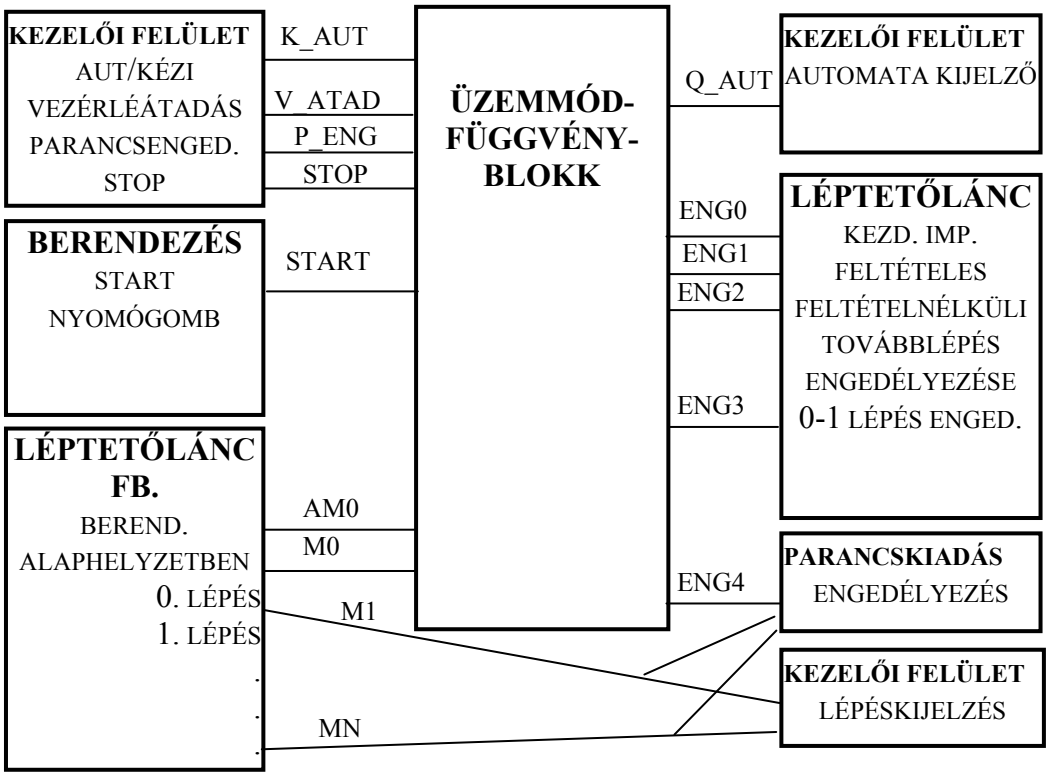
```
         (*VILLOGÁS KÉT
        IDŐZÍTŐVEL*)
        LDN     T2.Q
        ST      T1.IN
        LD      UTIDO1
        ST      T1.PT
        CAL     T1
        LDN     T1.Q
        ST      T2.IN
        LD      UTIDO2
        ST      T2.PT
        CAL     T2
        (*KIMENETEK BEÁLLÍTÁSA*)
        LD      T2.Q
        ST      SA
        LD      0
        ST      P
        ST      Z
        LD      0
        ST      UTEMB
        END_FUNCTION_BLOCK
```

Folyamatvezérelt lefutóvezérlések

Az üzemmód programrész (függvényblokk)

Az üzemmód függvényblokk feldolgozza a kezelői felületről és a berendezés felől jövő parancsokat, jelzéseket és engedélyező jelek formájában továbbítja a léptetőlánc felé. Visszajelzi ezen kívül az üzemmódot, és engedélyezi a kimenetek működtetését.

40. ábra A kezelői felület



2.ábra Az üzemmód programrész kapcsolata a többi programrésszel, illetve a kezelői felülettel és a berendezéssel

Az üzemmód függvényblokk

Készítsük el a fenti ábra előírásainak megfelelő függvényblokkot. A függvényblokk definiálásakor arra törekedtünk, hogy csak a konkrét technológiától független jeleket, a kezelői felület jeleit dolgozzuk fel. Azonos kezelői felület esetén így a későbbiekben változtatás nélkül felhasználhatjuk a függvényblokkot.

3. ábra A kezelői felület a függvényblokk változónév jelöléseivel

A függvényblokk utasításlistája:

```
FUNCTION_BLOCK UMODFGV
VAR_INPUT
        I1 : BOOL ;
        I2 : BOOL ;
        I3 : BOOL ;
        I4 : BOOL ;
        AM0 : BOOL ; (*BERENDEZÉS ALAPÁLLAPOTA*)
        M0 : BOOL ; (*0. LÉPÉS*)
        S0 : BOOL ; (*START a főprogramban*)
END_VAR
VAR_OUTPUT
        Q4 : BOOL ; (*AUTOMATA ÜZEMMÓD*)
        ENG0 : BOOL ; (*EN0 a főprogramban*)
        ENG1 : BOOL ; (*EN1 a főprogramban*)
        ENG2 : BOOL ; (*EN2 a főprogramban*)
        ENG3 : BOOL ; (*EN3 a főprogramban*)
        ENG4 : BOOL ; (*EN4 a főprogramban*)
END_VAR
VAR
        V_IMP : BOOL ;
        B11 : BOOL ;
        T_STOP : BOOL ;
END_VAR
        Impulzuskapcsoló a vezérlésátadás felfutó élére (V_IMP)
        LD      I2
        ANDN            B11
        ST      V_IMP
        LD      I2
        ST      B11

        Indító impulzus (ENG0)
```

```
LD      V_IMP
AND   AM0
ANDN          Q4
AND   I1
ANDN          M0
ST      ENG0
```

Automata üzemmód kijelzés (Q4) és feltételes léptetés engedélyező jele (ENG1)

```
LD      AM0
AND   V_IMP
AND   M0
S       Q4
LDN   I1
OR(    T_STOP
AND   M0
)
R       Q4
LD      Q4
ST      ENG1
```

Stop nyomógomb tárolása (T_STOP)

```
LD      Q4
ANDN          I4
S       T_STOP
LDN   Q4
R       T_STOP
```

Feltétel nélküli továbblépés engedélyező jele (ENG2)

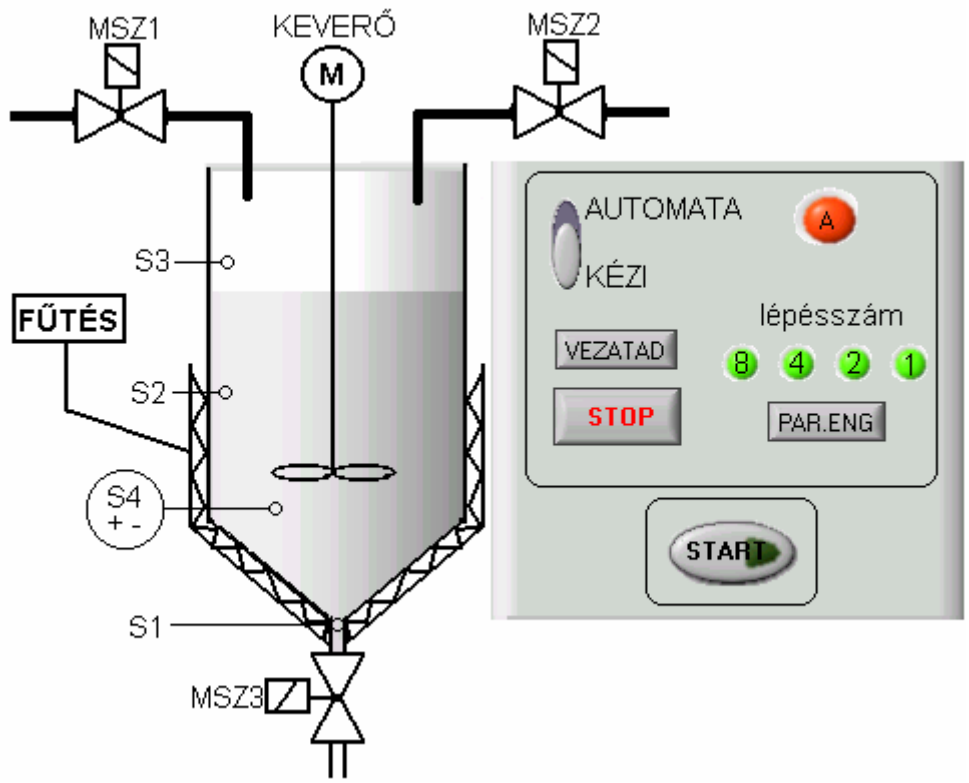
```
LD      V_IMP
ANDN          I1
ST      ENG2
```

Parancsengedélyezés (ENG4)

```
LD      S0
AND   AM0
ST      ENG3
LD      Q4
OR(    I3
ANDN          I1
)
ST      ENG4

END_FUNCTION_BLOCK
```

Szakaszos üzemű folyadékkeverő berendezés vezérlése

A technológiai feladat két különböző folyadék összeelegyítése és adott hőmérsékletre melegítése.



41. ábra Keverőtartály a kezelőtáblával

A berendezés szakaszos üzemben működik. Alapállapotban a tartály üres, a szelepek zárva. Automata üzemmódban a technológia az alábbi lépések sorozata:

START nyomógomb benyomására, a **MSZ1** jelű mágnesszelep nyit, az 1. folyadék beáramlik a tartályba. Ha **S2** szintérzékelő jelez, a **MSZ1** szelep zár, **MSZ2** nyit, bekapcsol a keverés, és beáramlik a 2. folyadék. Ha **S3** szintérzékelő jelez, **MSZ2** zár, a keverő mellett bekapcsol a fűtés. Ha a tartályban lévő folyadék hőmérséklete elérte a kívánt értéket, **S4** jelez, leáll a fűtés és a keverés, az **MSZ3** mágnesszelep nyit, a tartály leürül. Ha **S1** jelez, **START** jelre ismét indulhat elölről a folyamat.

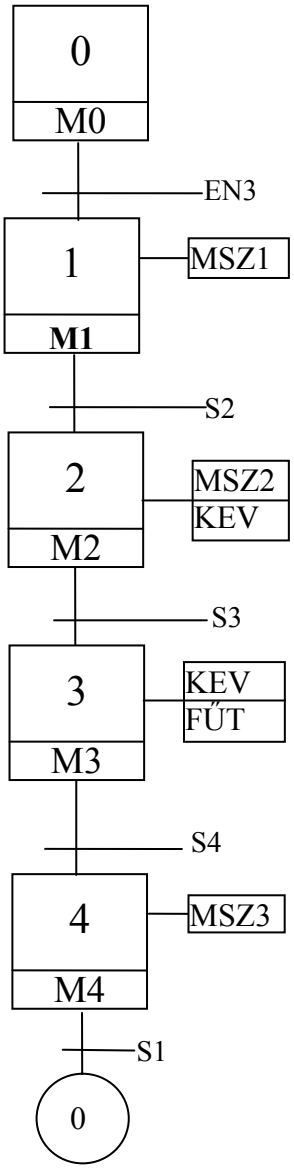
A berendezéshez tartozik egy olyan kezelői felület, amely biztosítja az üzemmód megválasztását, és különböző üzemmódokban a berendezés felügyeletét, a vezérlési lépések nyomon követését, a vezérlés tesztelését.

Összerendelési táblázat

A táblázatban felsoroljuk a technológiai jelek mellett a kezelői felület jeleit is.

Bemenetek	Jel	Logikai összerendelés	Cím
START	START	benyomva: START=1	I1.4
üres a tartály	S1	üres, ha: S1=1	I0.0
a tartály félig	S2	jelez, ha: S2=1	I0.1
a tartály tele	S3	jelez, ha: S3=1	I0.2
hőmérsékletérzékelő	S4	a mért hőmérséklet >= kívánt érték: S4=1	I0.3
Automata/kézi átkapcsoló	KAUT	automata: KAUT=1	I1.0
vezérlés átadás nyomógomb	VATAD	benyomva: VATAD=1	I1.1
parancsengedélyezés	PENG	benyomva: PENG=1	I1.2
STOP nyomógomb	STOP	benyomva: STOP=0	I1.3
Kimenetek	**Jel**	**Logikai összerendelés**	**Cím**
1. mágnesszelep	MSZ1	nyitva, ha: MSZ1=1	Q0.0
2. mágnesszelep	MSZ2	nyitva, ha: MSZ2=1	Q0.1
3. mágnesszelep	MSZ3	nyitva, ha: MSZ3=1	Q0.2
fűtés	FUT	bekapcsolva, ha: FUT=1	Q0.3
keverés	KEV	bekapcsolva, ha: KEV=1	Q0.4
lépéskijelző LED	LEP 0	világít, ha: LEP0=1	Q1.1
lépéskijelző LED	LEP1	világít, ha: LEP1=1	Q1.2
lépéskijelző LED	LEP2	világít, ha: LEP2=1	Q1.3
lépéskijelző LED	LEP3	világít, ha: LEP3=1	Q1.4
automata üzemmód jelzés	QAUT	világít, ha: LEP4=1	Q1.0

A léptetőlánc



A léptetőlánc funkciótervbe való átírásakor figyelembe vesszük az üzemmód függvényblokk engedélyező jeleit is.

:

A vezérlés függvényblokkjai	prototípus	feladata
üzemmód	UMODFB	üzemmód beállítása, engedélyező jelek
léptetőlánc	LEPTL	meghatározza a berendezés alapállapotát és beállítja a lépésmerkereket
lépéskijelzés	LEPKIJ	a lépéskijelző LED-ek beállítása
parancskiadás	PARKIAD	a kimenetek beállítása

A főprogram

A főprogramot utasításlistában adjuk meg.

A változódeklaráció:

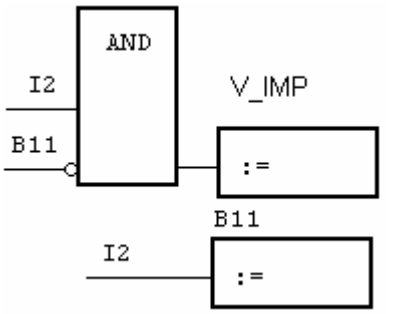
```
PROGRAM kevtart
VAR
        KAUT AT %I0.0.0.1.0:    BOOL;       (* =1 :AUTO =0 :KÉZI *)
        VATAD AT %I0.0.0.1.1:   BOOL;       (* =1 BENYOMVA *)
        PENG AT %I0.0.0.1.2:    BOOL;       (* =1 BENYOMVA *)
        STOP AT %I0.0.0.1.3:    BOOL;       (* =0 BENYOMVA *)
        START AT %I0.0.0.1.4:   BOOL;       (* =1 BENYOMVA *)
        QAUT AT %Q0.0.0.1.0:    BOOL;
        LEP1 AT %Q0.0.0.1.1:    BOOL;       (* LEPESkijelzés *)
        LEP2 AT %Q0.0.0.1.2:    BOOL;       (* LEPESkijelzés *)
        LEP3 AT %Q0.0.0.1.3:    BOOL;       (* LEPESkijelzés *)

        EN0:  BOOL;                         (*indító impulzus*)
        EN1:  BOOL;         (*feltételes továbblépés engedélyezése*)
        EN2:  BOOL; (*feltétel nélküli továbblépés engedélyezése*)
        EN3:  BOOL;                         (*0-1 lépés engedélyezése*)
        EN4:  BOOL;                         (*parancsengedélyezés*)
END_VAR

VAR
        S1 AT %I0.0.0.0.0:  BOOL;       (* 1.SZINT *)
        S2 AT %I0.0.0.0.1:  BOOL;       (* 2.SZINT *)
        S3 AT %I0.0.0.0.2:  BOOL;       (* 3.SZINT *)
        S4 AT %I0.0.0.0.3:  BOOL;       (* HOMERSEKLET *)

        MSZ1 AT %Q0.0.0.0.0:    BOOL;       (* 1.SZELEP *)
        MSZ2 AT %Q0.0.0.0.1:    BOOL;       (* 2.SZELEP *)
        MSZ3 AT %Q0.0.0.0.2:    BOOL;       (* 3.SZELEP *)
        FUT AT %Q0.0.0.0.3:     BOOL;       (* FUTES *)
        KEV AT %Q0.0.0.0.4:     BOOL;       (* KEVERO *)

        UZEM:       UMODFB;
        LANC:       LEPTL;
        KIJELZ:     LEPKIJ;
        KIMENET:    PARKIAD;
END_VAR

VAR_GLOBAL
        M0:   BOOL;     (* 0. lépés *)
        M1:   BOOL;     (* 1.lépés *)
        M2:   BOOL;     (* 2.lépés *)
        M3:   BOOL;     (* 3.lépés *)
        M4:   BOOL;     (* 4.lépés *)
        AM0:  BOOL;     (* berendezés alapállapota *)
END_VAR
```

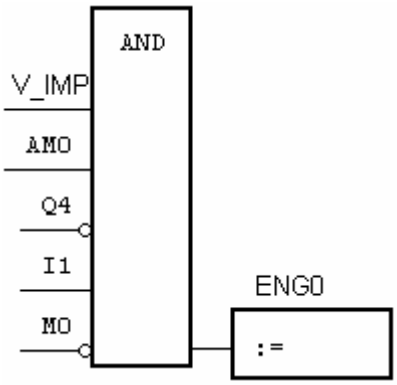
```
LD    KAUT
ST    UZEM.I1
LD    VATAD
ST    UZEM.I2
LD    PENG
ST    UZEM.I3
LD    STOP
ST    UZEM.I4
LD    START
ST    UZEM.S0
LD    AM0
ST    UZEM.AM0
LD    M0
ST    UZEM.M0

CAL   UZEM

LD    UZEM.Q4
ST    QAUT
LD    UZEM.ENG0
ST    EN0
LD    UZEM.ENG1
ST    EN1
LD    UZEM.ENG2
ST    EN2
LD    UZEM.ENG3
ST    EN3
LD    UZEM.ENG4
ST    EN4

CAL   LANC(ENG0:=EN0,ENG1:=EN1,
      ENG2:=EN2,ENG3:=EN3,
      S1:=S1,S2:=S2,
      S3:=S3,S4:=S4)

CAL   KIJELZ

LD    KIJELZ.Q0
ST    LEP1
LD    KIJELZ.Q1
ST    LEP2
LD    KIJELZ.Q2
ST    LEP3

CAL   KIMENET(ENG4:=EN4)

LD    KIMENET.Y1
ST    MSZ1
LD    KIMENET.Y2
```

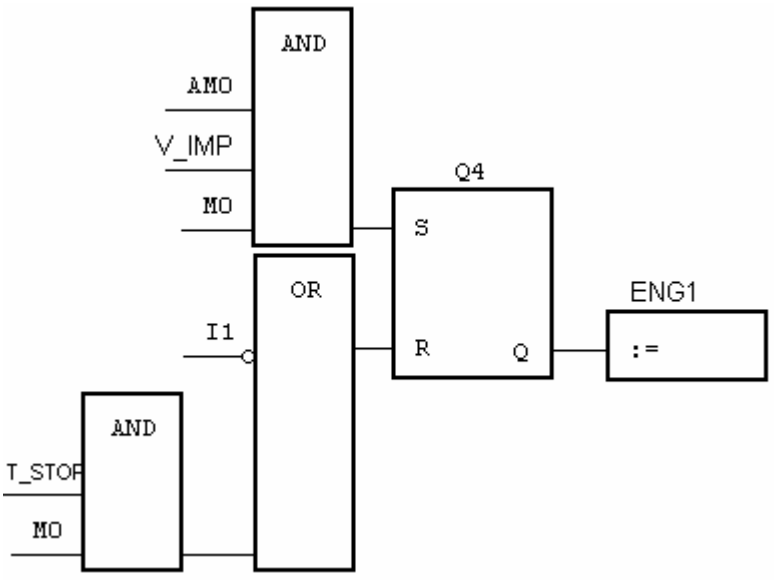
```
ST      MSZ2
LD      KIMENET.Y3
ST      MSZ3
LD      KIMENET.H
ST      FUT
LD      KIMENET.MOT
ST      KEV
END_PROGRAM
```

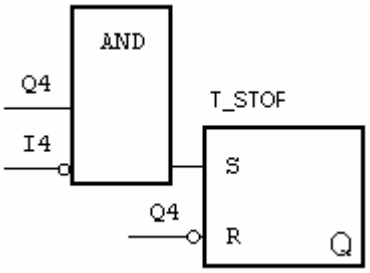
Az üzemmód függvényblokk

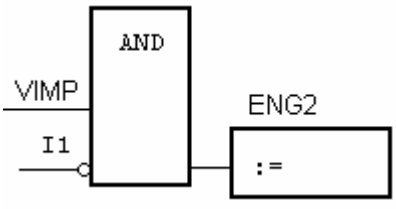
Funkcióterv

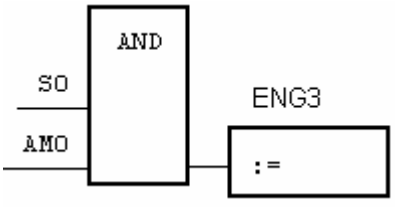


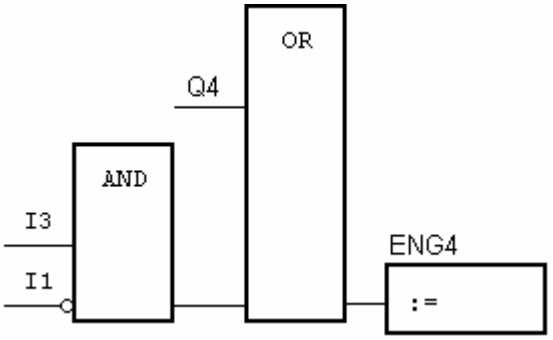





AND
Q4
I4
T_STOF
S
Q4
R
Q


AND
VIMP
I1
ENG2
:=


AND
SO
AMO
ENG3
:=

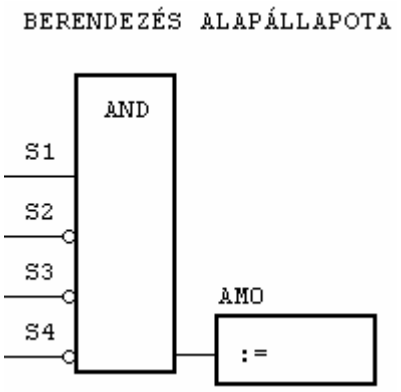

OR
Q4
AND
I3
I1
ENG4
:=

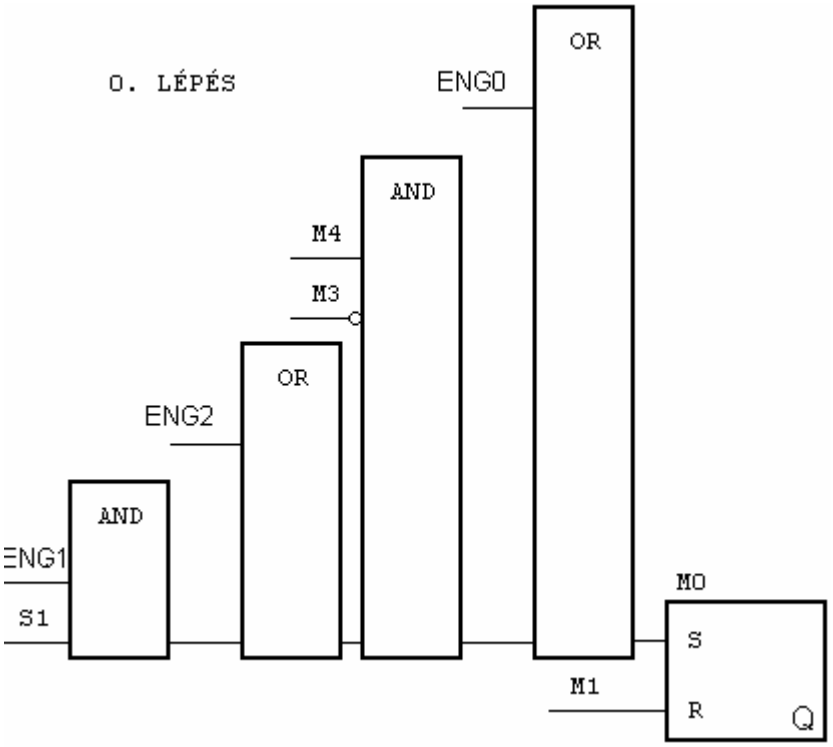
Utasításlista

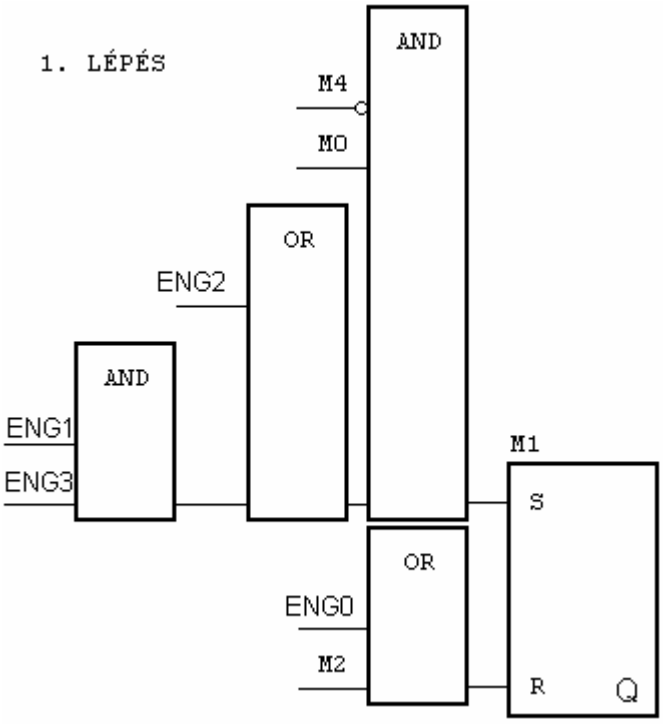
```
FUNCTION_BLOCK UMODFB
VAR_INPUT
        I1 : BOOL ;
        I2 : BOOL ;
        I3 : BOOL ;
        I4 : BOOL ;
        AM0 : BOOL ;
        M0 : BOOL ;
        S0 : BOOL ;
END_VAR
VAR_OUTPUT
        Q4 : BOOL ;
        ENG0 : BOOL ;
        ENG1 : BOOL ;
        ENG2 : BOOL ;
        ENG3 : BOOL ;
        ENG4 : BOOL ;
END_VAR
VAR
        V_IMP : BOOL ;
        B11 : BOOL ;
        T_STOP : BOOL ;
END_VAR

        LD      I2
        ANDN    B11
        ST      V_IMP

        LD      I2
        ST      B11

        LD      V_IMP
        AND     AM0
        ANDN    Q4
        AND     I1
        ANDN    M0
        ST      ENG0

        LD      AM0
        AND     V_IMP
        AND     M0
        S       Q4

        LDN     I1
        OR(     T_STOP
        AND     M0
        )
        R       Q4

        LD      Q4
        ST      ENG1
        LD      Q4
        ANDN    I4
        S       T_STOP

        LDN     Q4
        R       T_STOP

        LD      V_IMP
        ANDN    I1
        ST      ENG2

        LD      S0
        AND     AM0
        ST      ENG3

        LD      Q4
        OR(     I3
        ANDN    I1
        )
        ST      ENG4
END_FUNCTION_BLOCK
```

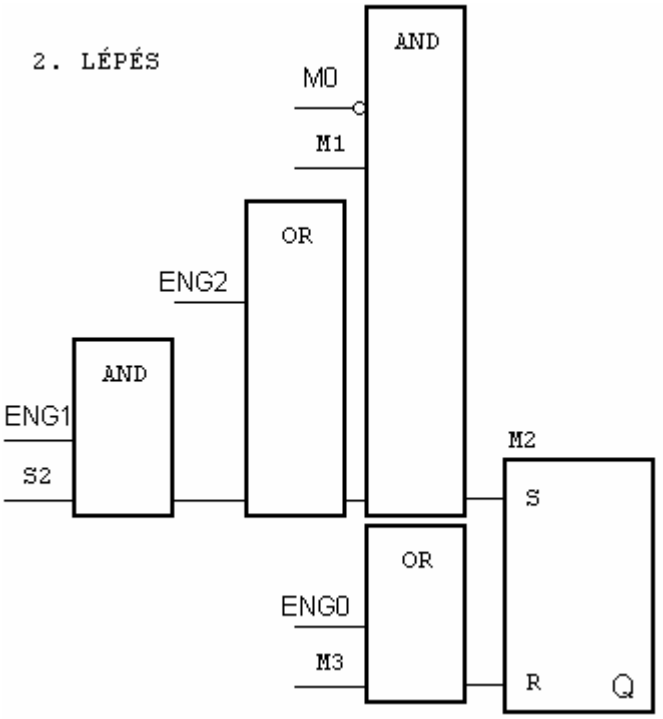
A léptetőlánc függvényblokk

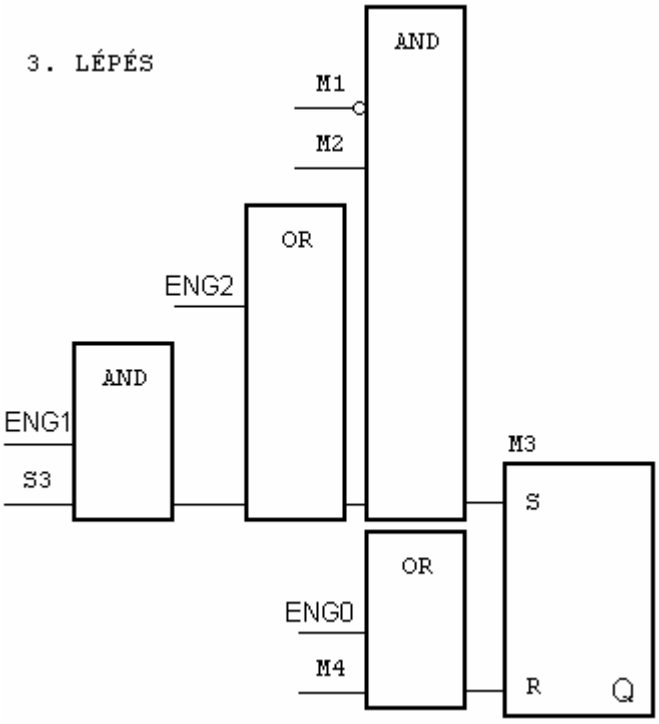
Funkcióterv

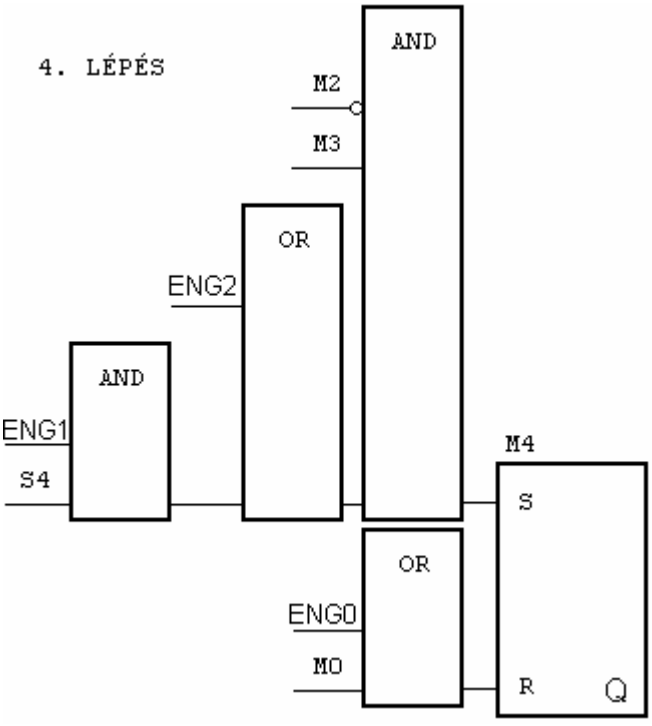





1. LÉPÉS
AND
M4
M0
OR
ENG2
AND
ENG1
ENG3
M1
S
OR
ENG0
M2
R
Q


2. LÉPÉS
AND
M0
M1
OR
ENG2
AND
ENG1
S2
M2
S
OR
ENG0
M3
R
Q


3. LÉPÉS
AND
M1
M2
OR
ENG2
AND
ENG1
S3
M3
S
OR
ENG0
M4
R
Q


4. LÉPÉS
AND
M2
M3
OR
ENG2
AND
ENG1
S4
M4
S
OR
ENG0
M0
R
Q

Utasításlista

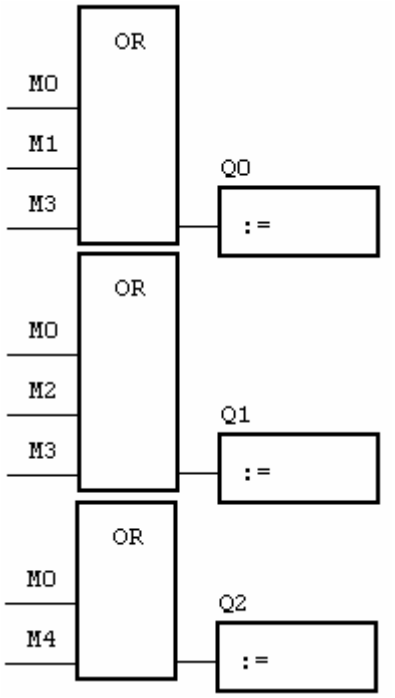
```
FUNCTION_BLOCK LEPTL
VAR_INPUT
        ENG0 : BOOL ;
        ENG1 : BOOL ;
        ENG2 : BOOL ;
        ENG3 : BOOL ;
        S1 : BOOL ;
        S2 : BOOL ;
        S3 : BOOL ;
        S4 : BOOL ;

END_VAR
VAR_EXTERNAL
        M0 : BOOL ;
        M1 : BOOL ;
        M2 : BOOL ;
        M3 : BOOL ;
        M4 : BOOL ;
        AM0 : BOOL ;
END_VAR

        (*BERENDEZÉS
        ALAPÁLLAPOTA*)

        LD      S1
        ANDN    S2
        ANDN    S3
        ANDN    S4
        ST      AM0

        (*0. LÉPÉS*)

        LD      ENG0
        OR      (       M4
                ANDN    M3
                AND     (       ENG2
                        OR      (       ENG1
                                AND     S1
                                )
                        )
                )
                S       M0



        LD      M1
        R       M0
```

```
        (*1. LÉPÉS*)

        LDN     M4
        AND     M0
        AND     (       ENG2
                OR      (       ENG1
                        AND     ENG3
                        )
                )
        S       M1
        LD      ENG0
        OR      M2
        R       M1
```

```
(*2. LÉPÉS*)

LDN   M0
AND   M1
AND   (       ENG2
      OR      (       ENG1
              AND     S2
              )
      )
S     M2
LD    ENG0
OR    M3
R     M2

(*3. LÉPÉS*)

LDN   M1
AND   M2
AND   (       ENG2
      OR      (       ENG1
              AND     S3
              )
      )
S     M3
LD    ENG0
OR    M4
R     M3

(*4. LÉPÉS*)

LDN   M2
AND   M3
AND   (       ENG2
      OR      (       ENG1
              AND     S4
              )
      )
S     M4
LD    ENG0
OR    M0
R     M4
END_FUNCTION_BLOCK
```

A lépéskijelzés függvényblokk

Funkcióterv



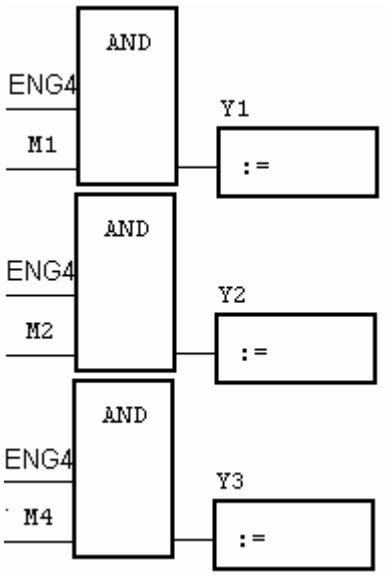
Utasításlista

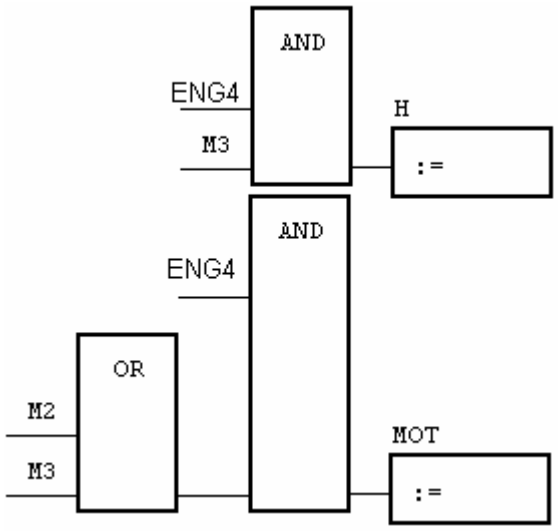
```
FUNCTION_BLOCK LEPKIJ
VAR_OUTPUT
        Q0 : BOOL ;
        Q1 : BOOL ;
        Q2 : BOOL ;
END_VAR
VAR_EXTERNAL
        M0 : BOOL ;
        M1 : BOOL ;
        M2 : BOOL ;
        M3 : BOOL ;
        M4 : BOOL ;
END_VAR
```

```
LD      M0
OR      M1
OR      M3
ST      Q0

LD      M0
OR      M2
OR      M3
ST      Q1

LD      M0
OR      M4
ST      Q2
END_FUNCTION_BLOCK
```

A parancskiadás függvényblokk

Funkcióterv





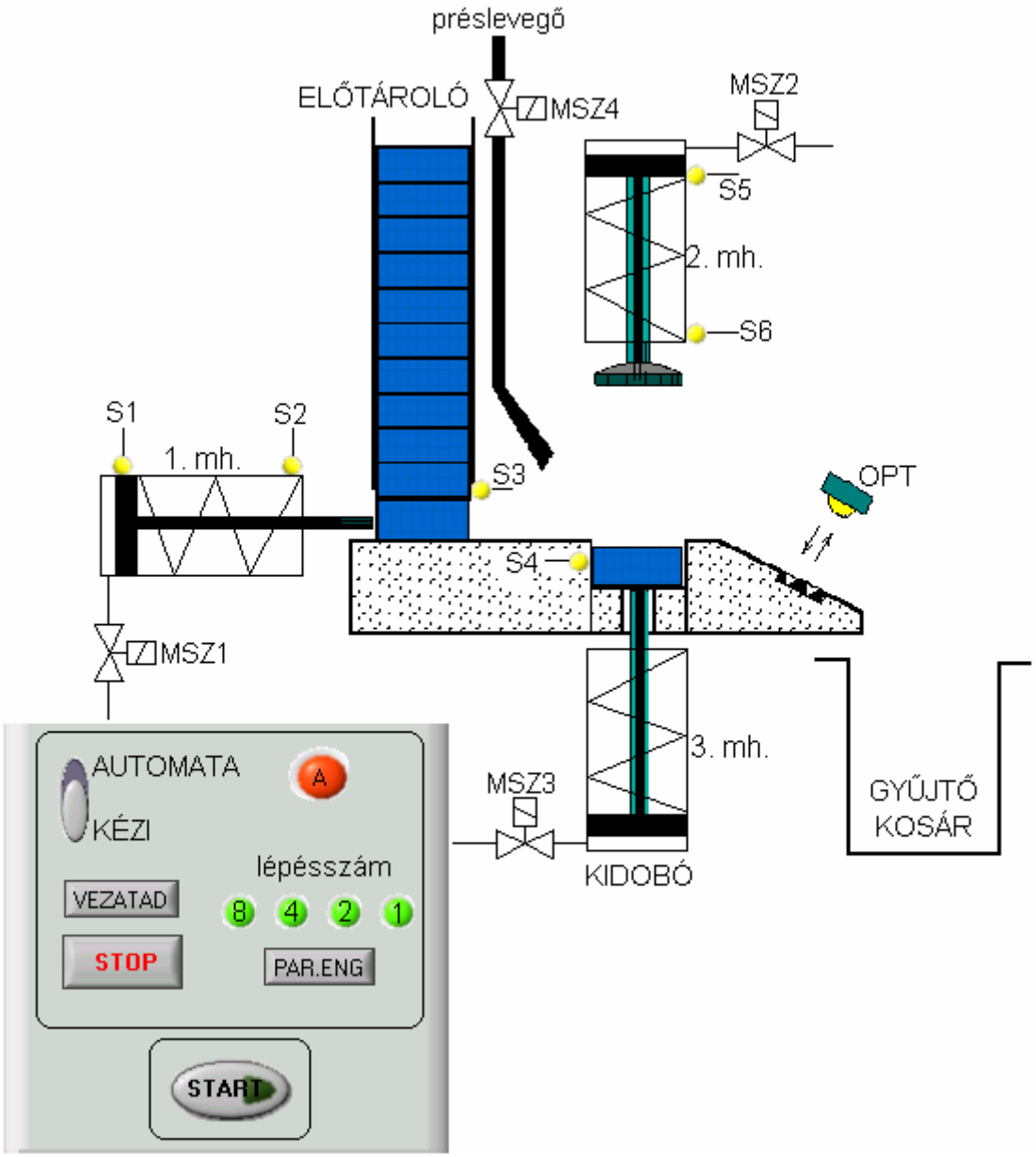
Utasításlista

```
FUNCTION_BLOCK PARKIAD
VAR_INPUT
        ENG4 : BOOL ;
END_VAR
VAR_OUTPUT
        Y1 : BOOL ;
        Y2 : BOOL ;
        Y3 : BOOL ;
        H : BOOL ;
        MOT : BOOL ;
END_VAR
VAR_EXTERNAL
        M1 : BOOL ;
        M2 : BOOL ;
        M3 : BOOL ;
        M4 : BOOL ;
END_VAR

        LD      ENG4
        AND     M1
        ST      Y1

        LD      ENG4
        AND     M2
        ST      Y2

        LD      ENG4
        AND     M4
        ST      Y3

        LD      ENG4
        AND     M3
        ST      H

        LD      ENG4
        AND(    M2
        OR      M3
        )
        ST      MOT
        END_FUNCTION_BLOCK
```

Példák lefutóvezérlésekre

Présgép vezérlése

A présberendezéssel az előtárolóból kivezetett munkadarabokba jelzést préselnek. A pneumatikus munkahengerek úgy vannak kiképezve, hogy csak egyik irányban szükséges a működtető levegő, a másik irányba (alaphelyzetbe) rugó tolja vissza.



42. ábra Présgép a kezelőtáblával

A préselési ciklus:
Az előtárolóból (**S3** jelzi, hogy van készenlétben új munkadarab) 1. munkahenger tolattyúja kitolja a munkadarabot a présformába. ha **S4** jelzi, hogy megfelelő pozícióba került a munkadarab, a préselő munkahenger rányomja a présmintát, 2s-ig lenyomva tartja. Ezután **MSZ2** mágnesszelep zár, a rugó visszaviszi eredeti pozíciójába a préskart (**S5** jelez). A kidobó

munkahenger és a terelőlevegő segítségével a munkadarab a gyűjtőkosárba kerül. Ha az **OPT** érzékelő jelzi, hogy a munkadarab áthaladt, **MSZ3** és **MSZ4** mágnesszelepek zárnak, és indulhat a folyamat előröl.

A préselés automata üzemmódban a **START** nyomógomb egyszeri lenyomására indul, és mindaddig fut ciklikusan, amíg van munkadarab az előtárolóban (ismételt végrehajtás engedélyezése!). (Az automata üzemmód természetesen a **STOP** gomb benyomásával a folyamatban lévő ciklus végén, a **Kézi** átkapcsolással pedig azonnal megszüntethető.)

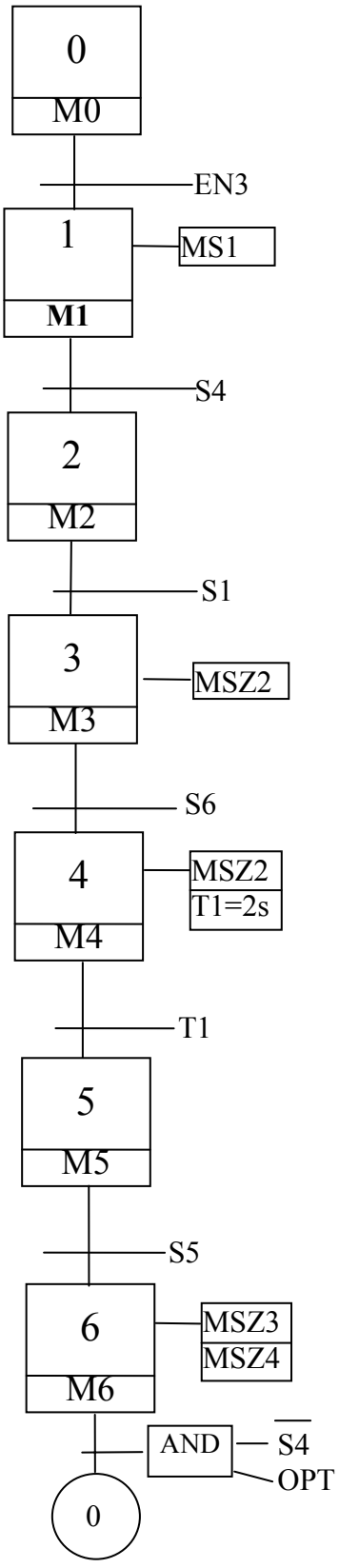
Összerendelési táblázat

A táblázatban felsoroljuk a technológiai jelek mellett a kezelői felület jeleit is.

Bemenetek	Jel	Logikai összerendelés	Cím
START	START	benyomva: START=1	I1.4
1. mh. hátsó pozíció	S1	jelez, ha: S1=1	I0.0
1. mh. első pozíció	S2	jelez, ha: S2=1	I0.1
munkadarab az előtárolóban	S3	jelez, ha: S3=1	I0.2
munkadarab a pésformában	S4	jelez, ha: S4=1	I0.3
2. mh. hátsó pozíció	S5	jelez, ha: S5=1	I0.4
2. mh. első pozíció	S6	jelez, ha: S6=1	I0.5
optoérzékelő	OPT	a fényút megszakad: OPT=1	I0.6
Automata/kézi átkapcsoló	KAUT	automata: KAUT=1	I1.0
vezérlés átadás nyomógomb	VATAD	benyomva: VATAD=1	I1.1
parancsengedélyezés	PENG	benyomva: PENG=1	I1.2
STOP nyomógomb	STOP	benyomva: STOP=0	I1.3

Kimenetek	Jel	Logikai összerendelés	Cím
1. mh. mágnesszelep	MSZ1	nyitva, ha: MSZ1=1	Q0.0
2. mh. mágnesszelep	MSZ2	nyitva, ha: MSZ2=1	Q0.1
3. mh. mágnesszelep	MSZ3	nyitva, ha: MSZ3=1	Q0.2
4. mh. mágnesszelep	MSZ4	nyitva, ha: MSZ4=1	Q0.3
lépéskijelző LED	LEP0	világít, ha: LEP0=1	Q1.0
lépéskijelző LED	LEP1	világít, ha: LEP1=1	Q1.1
lépéskijelző LED	LEP2	világít, ha: LEP2=1	Q1.2
lépéskijelző LED	LEP3	világít, ha: LEP3=1	Q1.3
automata üzemmód jelzés	QAUT	világít, ha: QAUT=1	Q1.4

Léptetőlánc



A léptetőlánc funkciótervbe való átírásakor figyelembe vesszük az üzemmód függvényblokk engedélyező jeleit is. Az üzemmód függvényblokkot változatlanul átvehetjük az előző feladatból. A lépéskijelzést ki kell bővítenünk 6 lépésszám kijelzésére. A léptetőláncot és a parancskiadást meg kell feleltetnünk az új technológiának, a főprogramban pedig deklarálnunk kell a megfelelő ki/bemeneti jeleket. A főprogramban a függvényblokkok hívását és az adatátadást aktualizálni kell.

A vezérlés függvényblokkjai	prototípus	feladata
üzemmód	UMODFB	üzemmód beállítása, engedélyező jelek
léptetőlánc	LEPTET	meghatározza a berendezés alapállapotát és beállítja a lépésmerkereket
lépéskijelzés	PRESLEP	a lépéskijelző LED-ek beállítása
parancskiadás	PRESPAR	a kimenetek beállítása

A főprogram

A főprogramot utasításlistában adjuk meg.

A változódeklaráció:

```
PROGRAM PRESPR

VAR
        KAUT AT %I0.0.0.1.0:      BOOL;     (* =1 :AUTO =0 :KÉZI *)
        VATAD AT %I0.0.0.1.1:     BOOL;     (* =1 BENYOMVA *)
        PENG AT %I0.0.0.1.2:      BOOL;     (* =1 BENYOMVA *)
        STOP AT %I0.0.0.1.3:      BOOL;     (* =0 BENYOMVA *)
        START AT %I0.0.0.1.4:     BOOL;     (* =1 BENYOMVA *)
        QAUT AT %Q0.0.0.1.0:      BOOL;
        LEP1 AT %Q0.0.0.1.1:      BOOL;     (* LEPESkijelzés *)
        LEP2 AT %Q0.0.0.1.2:      BOOL;     (* LEPESkijelzés *)
        LEP3 AT %Q0.0.0.1.3:      BOOL;     (* LEPESkijelzés *)
        EN0:  BOOL;       (* indító impulzus *)
        EN1:  BOOL;       (* feltételes továbblépés engedélyezése *)
        EN2:  BOOL;       (* feltétel nélküli továbblépés engedélyezése *)
        EN3:  BOOL;       (* 0-1 lépés engedélyezése *)
        EN4:  BOOL;       (* parancsengedélyezés *)
        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;
        OPT AT %I0.0.0.0.6: BOOL;
        MSZ1 AT %Q0.0.0.0.0:      BOOL;
        MSZ2 AT %Q0.0.0.0.1:      BOOL;
        MSZ3 AT %Q0.0.0.0.2:      BOOL;
        MSZ4 AT %Q0.0.0.0.3:      BOOL;
        UZEM:       UMODFB;
        LANC:       LEPTETP;
        KIJELZ:     PRESLEP;
```

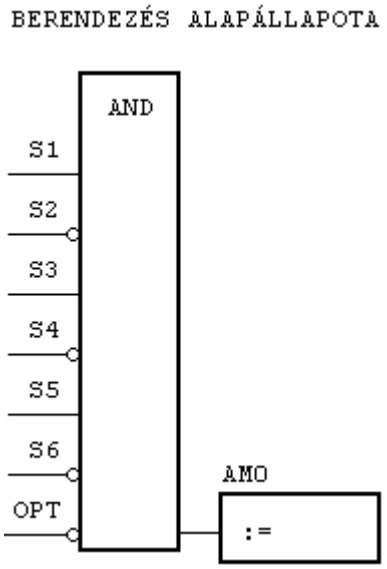
```
        KIMENET:    PRESPAR;
END_VAR

VAR_GLOBAL
        AM0:   BOOL; (* berendezés alapállapota *)
        M0:    BOOL; (* 0. lépés *)
        M1:    BOOL; (* 1.lépés *)
        M2:    BOOL; (* 2.lépés *)
        M3:    BOOL; (* 3.lépés *)
        M4:    BOOL; (* 4.lépés *)
        M5:    BOOL; (* 5.lépés *)
        M6:    BOOL; (* 6.lépés *)
END_VAR
```

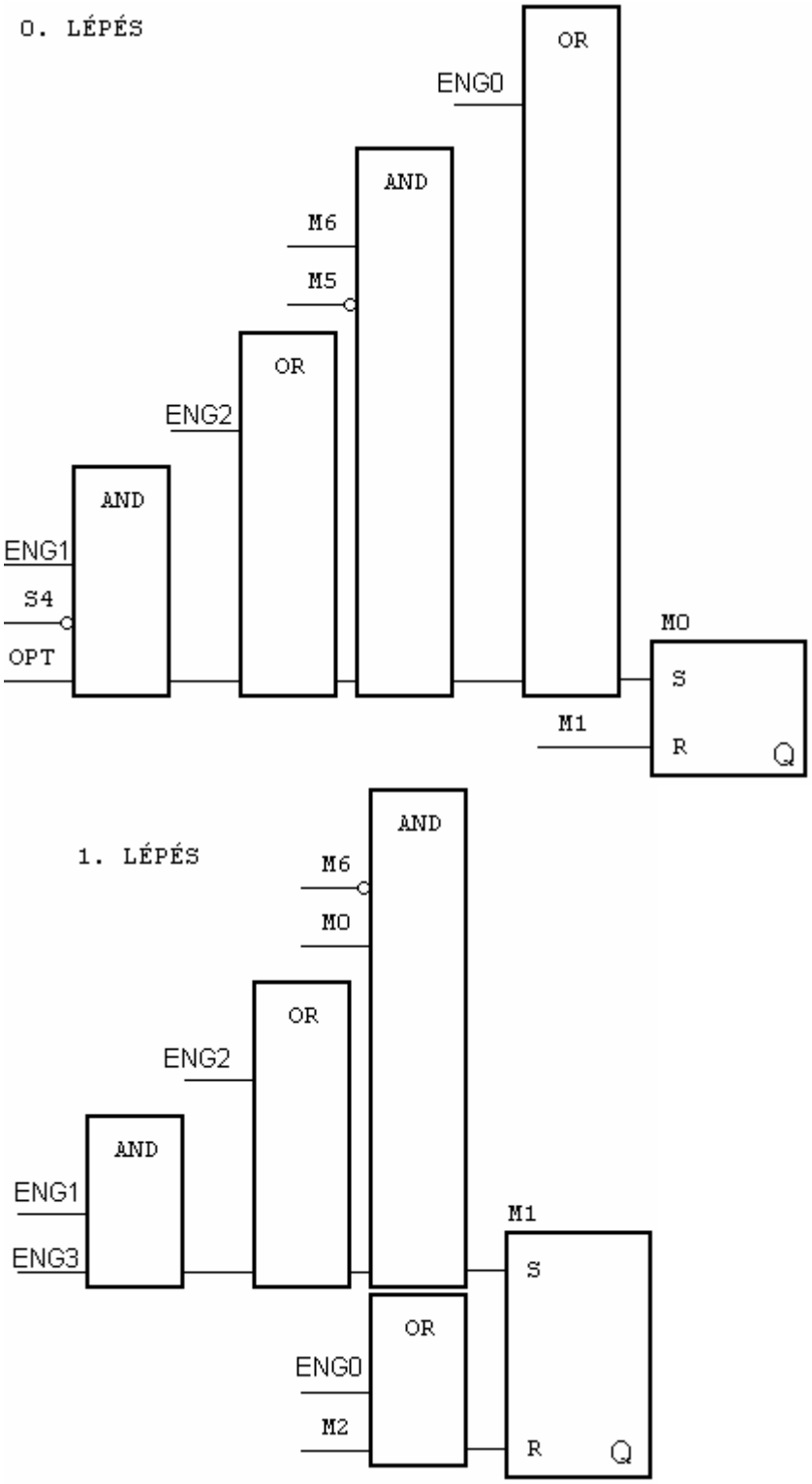
programtörzs:

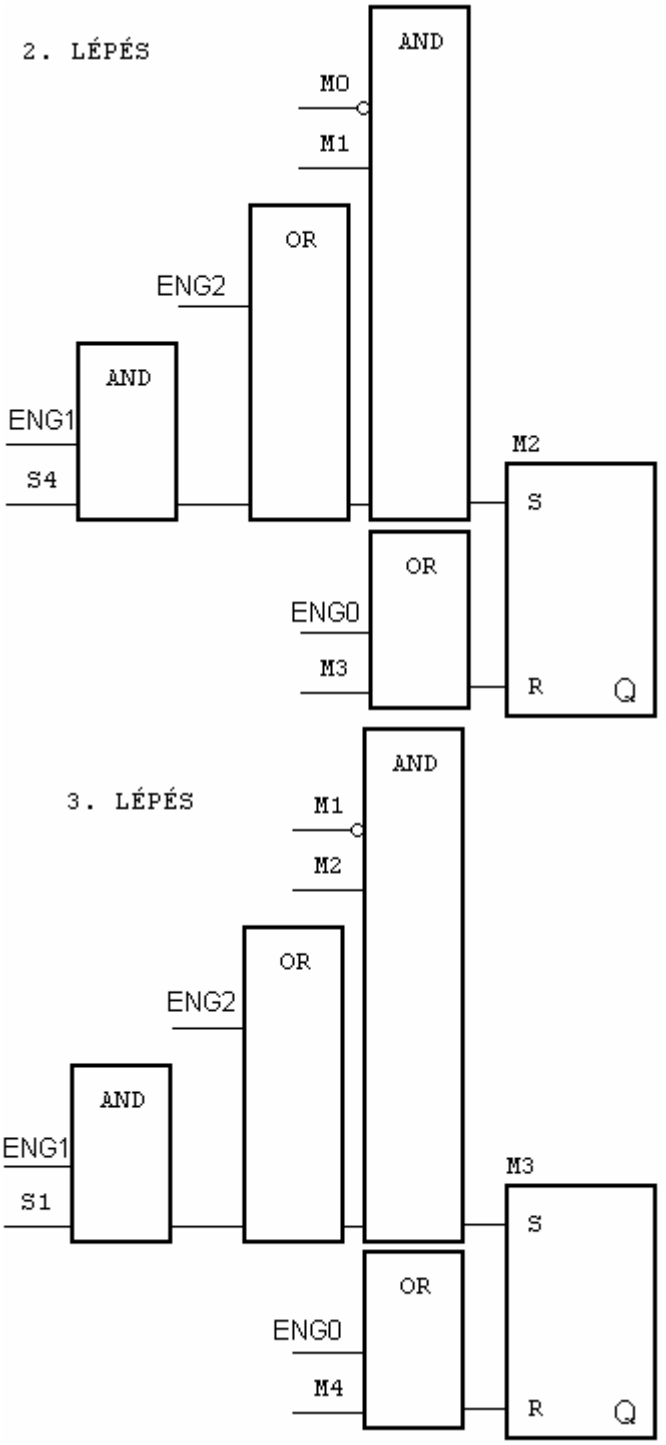
```
LD      KAUT
ST      UZEM.I1
LD      VATAD
ST      UZEM.I2
LD      PENG
ST      UZEM.I3
LD      STOP
ST      UZEM.I4
LD      START
ST      UZEM.S0
LD      AM0
ST      UZEM.AM0
LD      M0
ST      UZEM.M0

CAL     UZEM

LD      UZEM.Q4
ST      QAUT
LD      UZEM.ENG0
ST      EN0
LD      UZEM.ENG1
ST      EN1
LD      UZEM.ENG2
ST      EN2
LD      UZEM.ENG3
ST      EN3
LD      UZEM.ENG4
ST      EN4

CAL     LANC(ENG0:=EN0,ENG1:=EN1,
        ENG2:=EN2,ENG3:=EN3,
        S1:=S1,S2:=S2,
        S3:=S3,S4:=S4,
        S5:=S5,S6:=S6,
        OPT:=OPT)

CAL     KIJELZ
```

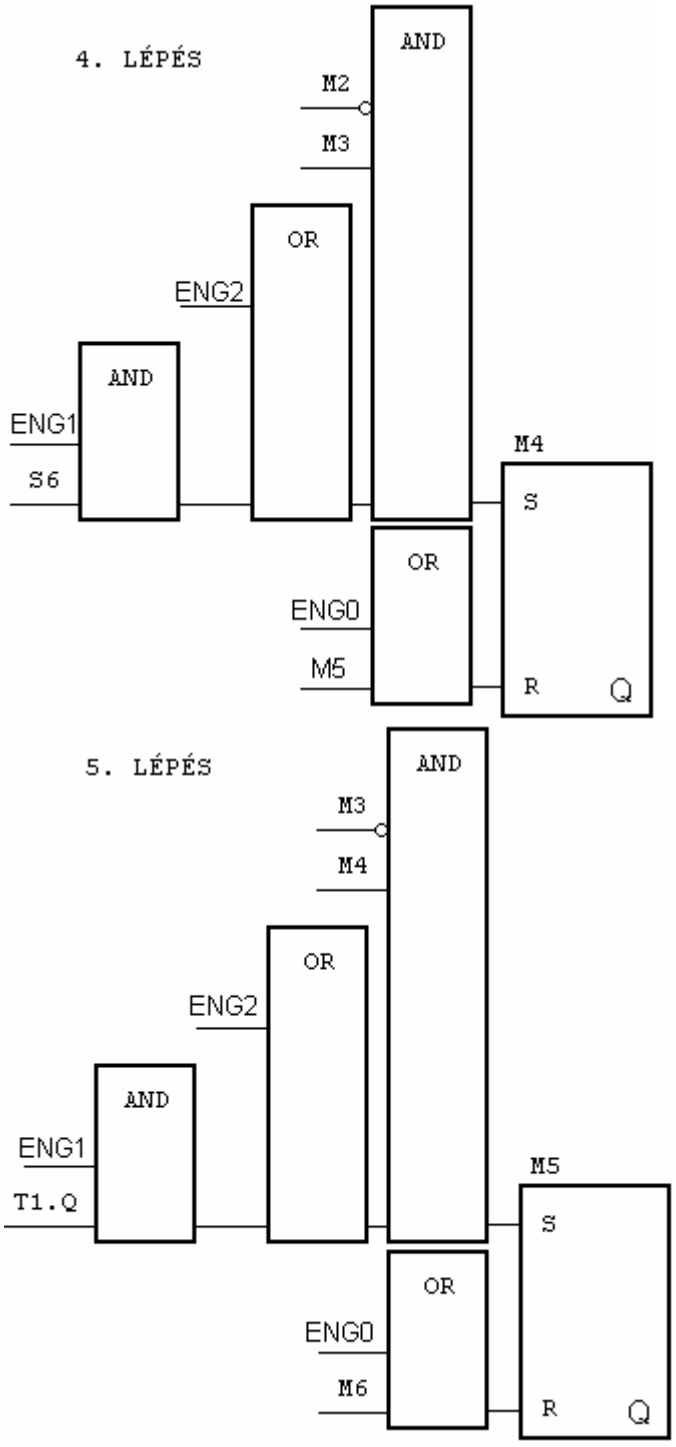
```
LD      KIJELZ.Q0
ST      LEP1
LD      KIJELZ.Q1
ST      LEP2
LD      KIJELZ.Q2
ST      LEP3

CAL     KIMENET(ENG4:=EN4)

LD      KIMENET.Y1
ST      MSZ1
LD      KIMENET.Y2
ST      MSZ2
LD      KIMENET.Y3
ST      MSZ3
LD      KIMENET.Y4
ST      MSZ4
END_PROGRAM
```

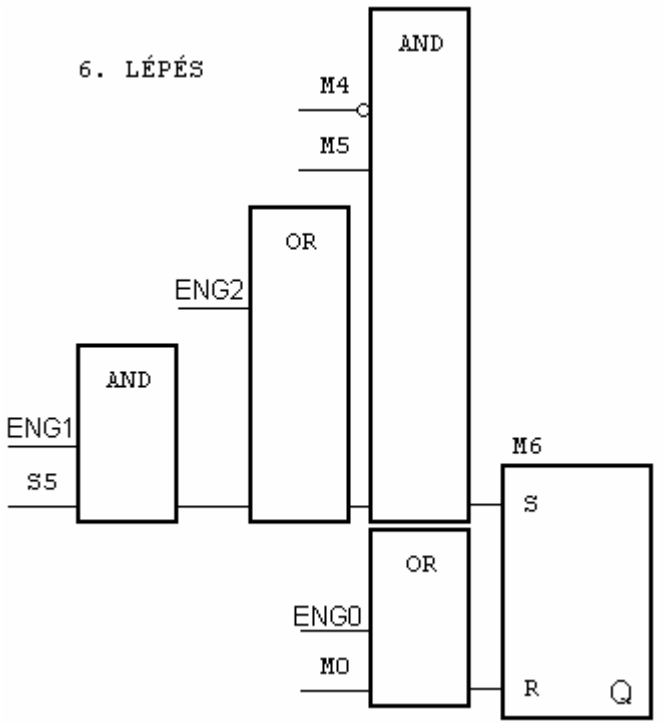
A léptetőlánc függvényblokk funkciótervben

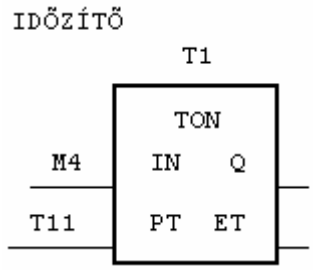



0. LÉPÉS
OR
ENG0
AND
M6
M5
OR
ENG2
AND
ENG1
S4
OPT
M0
S
M1
R
Q
1. LÉPÉS
AND
M6
M0
OR
ENG2
AND
ENG1
ENG3
M1
S
OR
ENG0
M2
R
Q


2. LÉPÉS
AND
M0
M1
OR
ENG2
AND
ENG1
S4
M2
S
OR
ENG0
M3
R
Q
3. LÉPÉS
AND
M1
M2
OR
ENG2
AND
ENG1
S1
M3
S
OR
ENG0
M4
R
Q


4. LÉPÉS
AND
M2
M3
OR
ENG2
AND
ENG1
S6
M4
S
OR
ENG0
M5
R
Q
5. LÉPÉS
AND
M3
M4
OR
ENG2
AND
ENG1
T1.Q
M5
S
OR
ENG0
M6
R
Q





Utasításlistában

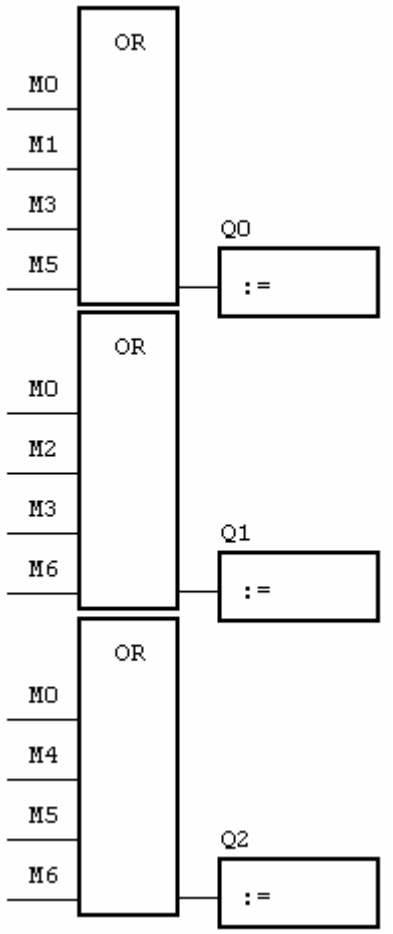
```
FUNCTION_BLOCK LEPTETP

VAR_INPUT
        ENG0: BOOL;
        ENG1: BOOL;
        ENG2: BOOL;
        ENG3: BOOL;
        S1:   BOOL;
        S2:   BOOL;
        S3:   BOOL;
        S4:   BOOL;
        S5:   BOOL;
        S6:   BOOL;
        OPT:  BOOL;
END_VAR

VAR
        T1:   TON;
        T11:  TIME := T#2s;
END_VAR
```

```
VAR_EXTERNAL
        AM0:  BOOL;
        M0:   BOOL;
        M1:   BOOL;
        M2:   BOOL;
        M3:   BOOL;
        M4:   BOOL;
        M5:   BOOL;
        M6:   BOOL;
END_VAR

(*BERENDEZÉS ALAPÁLLAPOTA*)

LD      S1
ANDN    S2
AND     S3
ANDN    S4
AND     S5
ANDN    S6
ANDN    OPT
ST      AM0

(*0. LÉPÉS*)

LD      ENG0
OR      (       M6
        ANDN    M5
        AND     (       ENG2
                OR      (       ENG1
                        ANDN    S4
                 AND    OPT
                        )
                )
        )
        S       M0



LD      M1
R       M0

(*1. LÉPÉS*)

LDN     M6
AND     M0
AND     (       ENG2
        OR      (       ENG1
                AND     ENG3
                )
```

```
        )
S       M1
LD      ENG0
OR      M2
R       M1

(*2. LÉPÉS*)

LDN     M0
AND     M1
AND     (       ENG2
        OR      (       ENG1
                AND     S4
                )
        )
S       M2
LD      ENG0
OR      M3
R       M2

(*3. LÉPÉS*)

LDN     M1
AND     M2
AND     (       ENG2
        OR      (       ENG1
                AND     S1
                )
        )
S       M3
LD      ENG0
OR      M4
R       M3

(*4. LÉPÉS*)

LDN     M2
AND     M3
AND     (       ENG2
        OR      (       ENG1
                AND     S6
                )
        )
S       M4
LD      ENG0
OR      M5
R       M4

(*5. LÉPÉS*)
```

```
LDN   M3
AND   M4
AND   (       ENG2
      OR      (       ENG1
              AND     T1.Q
              )
      )
S     M5
LD    ENG0
OR    M6
R     M5

(*6. LÉPÉS*)

LDN   M4
AND   M5
AND   (       ENG2
      OR      (       ENG1
              AND     S5
              )
      )
S     M6
LD    ENG0
OR    M0
R     M6

(*IDŐZÍTŐ*)

LD    M4
ST    T1.IN
LD    T11
ST    T1.PT
CAL   T1
END_FUNCTION_BLOCK
```

A lépéskijelzés függvényblokk funkciótervben

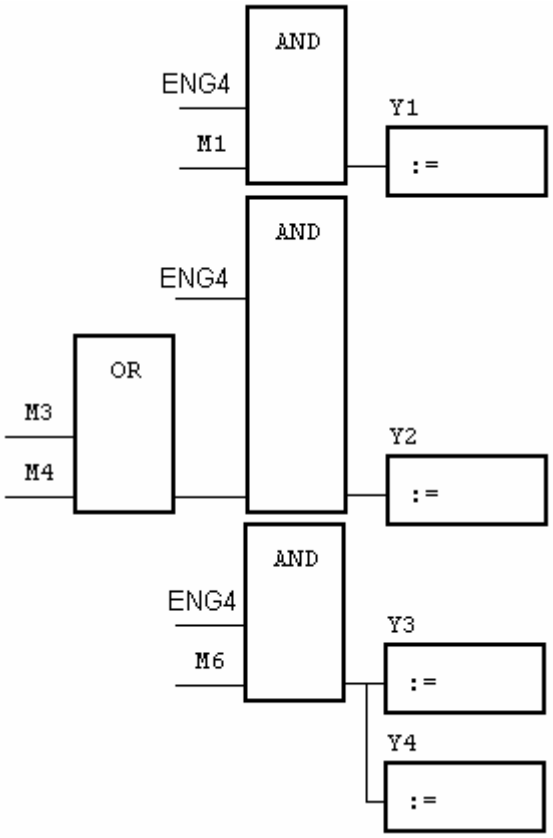


Utasításlistában

```
FUNCTION_BLOCK PRESLEP
VAR_OUTPUT
        Q0:     BOOL;
        Q1:     BOOL;
        Q2:     BOOL;
END_VAR
VAR_EXTERNAL
        M0:     BOOL;
        M1:     BOOL;
        M2:     BOOL;
        M3:     BOOL;
        M4:     BOOL;
        M5:     BOOL;
        M6:     BOOL;
END_VAR

LD      M0
OR      M1
OR      M3
OR      M5
ST      Q0
```

```
LD      M0
OR      M2
OR      M3
OR      M6
ST      Q1

LD      M0
OR      M4
OR      M5
OR      M6
ST      Q2
END_FUNCTION_BLOCK
```

A parancskiadás függvényblokk funkciótervben



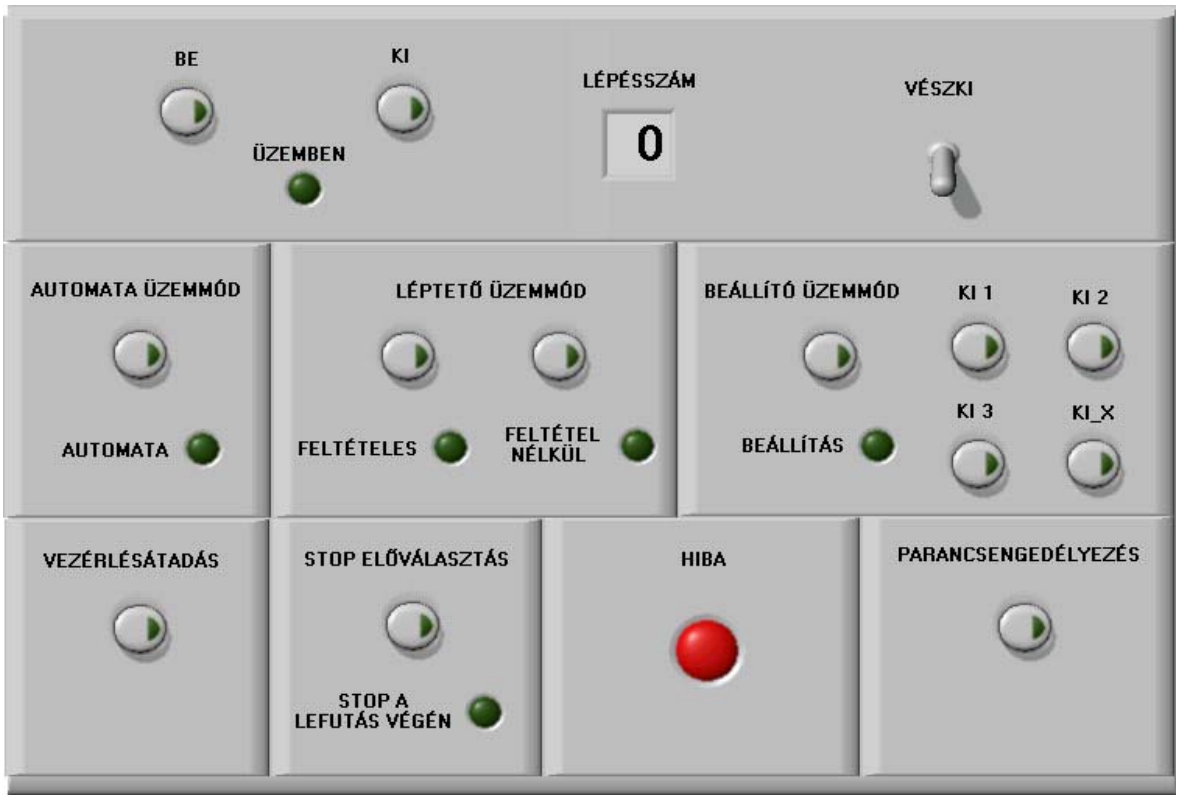
Utasításlistában

```
FUNCTION_BLOCK PRESPAR
VAR_INPUT
        ENG4 : BOOL ;
END_VAR
VAR_OUTPUT
        Y1 : BOOL ;
        Y2 : BOOL ;
        Y3 : BOOL ;
        Y4 : BOOL ;
```

```
END_VAR
VAR_EXTERNAL
        M1 : BOOL ;
        M2 : BOOL ;
        M3 : BOOL ;
        M4 : BOOL ;
        M5 : BOOL ;
        M6 : BOOL ;
END_VAR
LD      ENG4
AND     M1
ST      Y1

LD      ENG4
AND(    M3
OR      M4
)
ST      Y2

LD      ENG4
AND     M6
ST      Y3
ST      Y4
END_FUNCTION_BLOCK
```

Kezelői felület VÉSZKI-kapcsolóval, többféle üzemmód választásának lehetőségével

Elképzelhető, hogy egy szakaszos üzemű technológia levezérléséhez az előző példákban alkalmazott kezelői felület nem elégséges. Összetettebb technológiáknál biztosítanunk kell a be/ki kapcsolás mellett a vészkikapcsolás lehetőségét is. Az automata üzemmódon kívül feltételes és feltétel nélküli kézi léptető üzemmód is igény lehet. A berendezés alapálllapotának beállításához, vészleállás utáni beavatkozáshoz szükséges a beállító üzemmód. A technológia felől érkező hibajelzések, időtúllépések is leállíthatják az automata üzemmódot, hibajelzést adva. Egy ilyen kezelői felület egy lehetséges megvalósítása látható az alábbi ábrán.



43. ábra Kezelői felület

A megváltozott kezelői felület az üzemmód függvényblokk módosítását is megkívánja. Az alábbi utasításlista példa egy lehetséges megoldásra. A függvényblokkot úgy írtuk meg, hogy az engedélyező jelek, vagyis az illeszkedés a léptetőlánc és a parancskiadás függvényblokk felé változatlan marad, de alkalmassá tettük a kezelői felület felől érkező, megváltozott számú és funkciójú jelek fogadására illetve kiadására.

A függvényblokk utasításlistája

(Az előző feladatok üzemmód függvényblokkját az új függvényblokkra cseréve, azok az új kezelői felületről vezérelhetők. Természetesen a lépésszám-kijelzést a számjegyes kijelzőnek megfelelően módosítanunk kell.)

```
FUNCTION_BLOCK UMDOSSZ
VAR_INPUT
        I1:     BOOL;       (* VÉSZKI *)
        I2:     BOOL;       (* BE/KI, BE=1 *)
        I3:     BOOL;       (* AUTOMATA, BENYOMNVA=1  *)
```

```
        I4:     BOOL;       (* FELTÉTELES LÉPTETŐ ÜM , BENYOMNVA=1 *)
        I5:     BOOL;       (* FELTÉTEL NÉLKÜLI LÉPTETŐ ÜM., BENYOMNVA=1 *)
        I6:     BOOL;       (* BEÁLLÍTÓ ÜM., BENYOMNVA=1 *)
        I7:     BOOL;       (* VEZÉRLÉSÁTADÁS, BENYOMNVA=1 *)
        I8:     BOOL;       (* STOP, BENYOMNVA=1 *)
        I9:     BOOL;       (* PARANCS ENGEDÉLYEZÉS, BENYOMNVA=1 *)
        AM0:    BOOL;       (* BV: BERENDEZÉS ALAPÁLLAPOTA *)
        AM1:    BOOL;       (* BV: HIBA A TECHNOLÓGIÁBÓL *)
        M0:     BOOL;       (* 0. LÉPÉS MERKERE *)
        SX:     BOOL;       (* HIBAJEL A TECHNOLÓGIÁBÓL *)
        TX:     BOOL;       (* IDŐTÚLLÉPÉS A TECHNOLÓGIÁBÓL *)
END_VAR
VAR_OUTPUT
        Q1:     BOOL;       (* ÜZEMEL LED *)
        Q2:     BOOL;       (* AUTOMATA ÜZEMMÓD LED *)
        Q3:     BOOL;       (* FELTÉTELES LÉPTETŐ ÜZEMMÓD LED *)
        Q4:     BOOL;       (* FELTÉTEL NÉLKÜLI LÉPTETŐ ÜM. LED *)
        Q5:     BOOL;       (* BEÁLLÍTÓ ÜZEMMÓD LED *)
        Q6:     BOOL;       (* STOP ELŐJELZÉS LED *)
        Q7:     BOOL;       (* HIBA LED *)
        Q8:     BOOL;
        B0:     BOOL;       (* INDÍTÓ IMPULZUS *)
        B1:     BOOL;       (* FELTÉTELES TOVÁBBLÉPÉS ENGEDÉLYEZÉSE *)
        B2:     BOOL;       (* FELTÉTEL NÉLKÜLI TOVÁBBLÉPÉS ENGEDÉLYEZÉSE
*)
        B3:     BOOL;       (* 0-1 LÉPÉS ENGEDÉLYEZÉSE *)
        B4:     BOOL;       (* KIMENET-ENGEDÉLYEZÉS *)
END_VAR
VAR
        B10:    BOOL;
        B11:    BOOL;
        B12:    BOOL;
        B13:    BOOL;
        B14:    BOOL;
        B15:    BOOL;
        B16:    BOOL;
END_VAR


(*BE/KIKAPCSOLÁS IMPULZUS FELFUTÓ ÉLRE*)
LD      I2
ANDN    B11
ST      B10


LD      I2
ST      B11
```

```
(*ÜZEMEL KIJELZÉS*)
LD    B10
S     Q1


LDN   I2
ORN   I1
OR    AM1
R     Q1


(*AUTOMAT ÜZEM KIJELZÉS*)
LD    I3
S     Q2


LDN   Q1
OR    Q3
OR    Q4
OR    Q5
OR    B12
R     Q2


(*FELTÉTELES LÉPTETÉS ÜZEM KIJELZÉS*)
LD    I4
S     Q3


LDN   Q1
OR    I5
OR    Q5
OR    B12
R     Q3


(*FELTÉTEL NÉLKÜLI LÉPTETÉS ÜZEM KIJELZÉS*)
LD    I5
S     Q4


LDN   Q1
OR    I4
OR    Q5
OR    B12
R     Q4
```

```
(*STOP ELŐVÁLASZTÁS VISSZAJELZÉS*)
LD    I8
S     Q6


LDN   Q2
ANDN  Q3
ANDN  Q4
ANDN  Q5
R     Q6


LD    Q6
AND   M0
ST    B12


(*INDÍTÓ IMPULZUS*)
LD    Q1

ANDN  B13
ST    B0


LD    B0
S     B13


LDN   Q1
R     B13


(*VEZÉRLÉSÁTADÁS IMPULZUS FELFUTÓ ÉLRE*)
LD    I7
ANDN  B15
ST    B14


LD    I7
ST    B15


(*FELTÉTELES TOVÁBBLÉPTETÉS ENGEDÉLYEZÉSE*)
LD    Q2
OR    (     Q3
      AND   B14
      )
ST    B1
```

```
(*FELTÉTEL NÉLKÜLI TOVÁBBLÉPTETÉS ENGEDÉLYEZÉSE*)
LD    Q4
AND   B14
ST    B2


(*START-FELTÉTEL 0-1 LÉPTETÉS*)
LD    I7
S     B16


LDN   Q2
ANDN  Q3
OR    I8
R     B16


LD    AM0
AND   B1
AND   B16
ST    B3


(*PARANCSENGEDÉLYEZÉS*)
LD    Q2
OR    (     I9
      AND   (     Q3
            OR    Q4
            )
      )
ST    B4



(*HIBAJELZÉS*)
LD    Q2
AND   TX
OR    (     SX
NOT
      AND   (     Q2
            OR    Q3
            OR    Q4
            )
      )
S     AM1


LD    I7
R     AM1
```

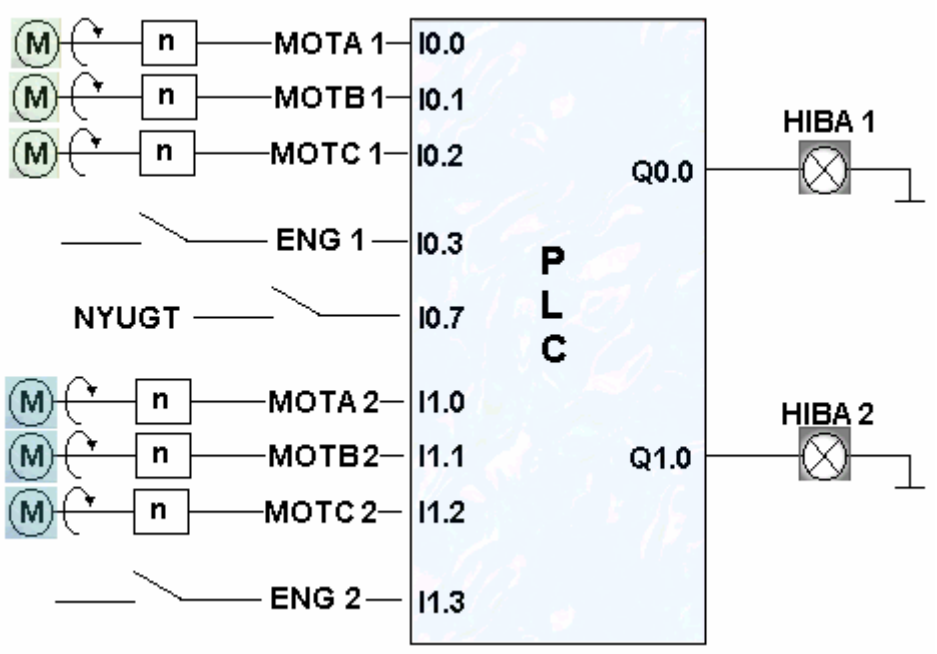
```
LD      AM1
ST      Q7
END_FUNCTION_BLOCK
```

Feladat: Írja át a fenti utasításlistát funkciótervbe és elemezze a működését!

Digitális vezérlések

Saját készítésű függvényblokk: Motorblokkok felügyelete

A feladat: 2 db, A,B,C motorból álló motorblokk felügyelete. Minden motorhoz tartozik egy fordulatszámjelző, amely folyamatos 1 jelet ad, ha a motor működik. Ha a motor leáll, az érzékelő-távadó kimenete 0-ra vált. A felügyelet akkor indul, vagyis a vezérlésnek akkor kell figyelni a motorok működését, ha a motorcsoporthoz tartozó engedélyező jelet a kezelőszemély bekapcsolta.



44. ábra Kettő, egyenként 3 db motorból álló motorblokk felügyelete

A hibajelzés feltételei:

1. eset: a háromból két motor 5 s-nál hosszabb ideig leáll. (Időkésleltetett jelzés).
2. eset: mindhárom motor leáll. (Azonnali jelzés).

A hibajelzés megszüntetése:

1. eset: a hibajelzés magától megszűnik, ha valamelyik motor újra indul, azaz ismét legalább két motor fut..
2. eset: a kezelőnek a hiba elhárítása után meg kell nyomnia a nyugtázó gombot is ahhoz, hogy a hibajelzés megszűnjön.

Megoldás: mivel minkét motorblokknál ugyanazt a feladatot kell megoldani, ezt függvényblokkban írjuk meg, és a két motorcsoporthoz külön-külön egyedi névvel deklaráljuk. A főprogramban kell gondoskodni a be/kimenőjelek fizikai címekhez rendeléséről és a függvényblokkok aktuális paraméterekkel történő hívásáról.

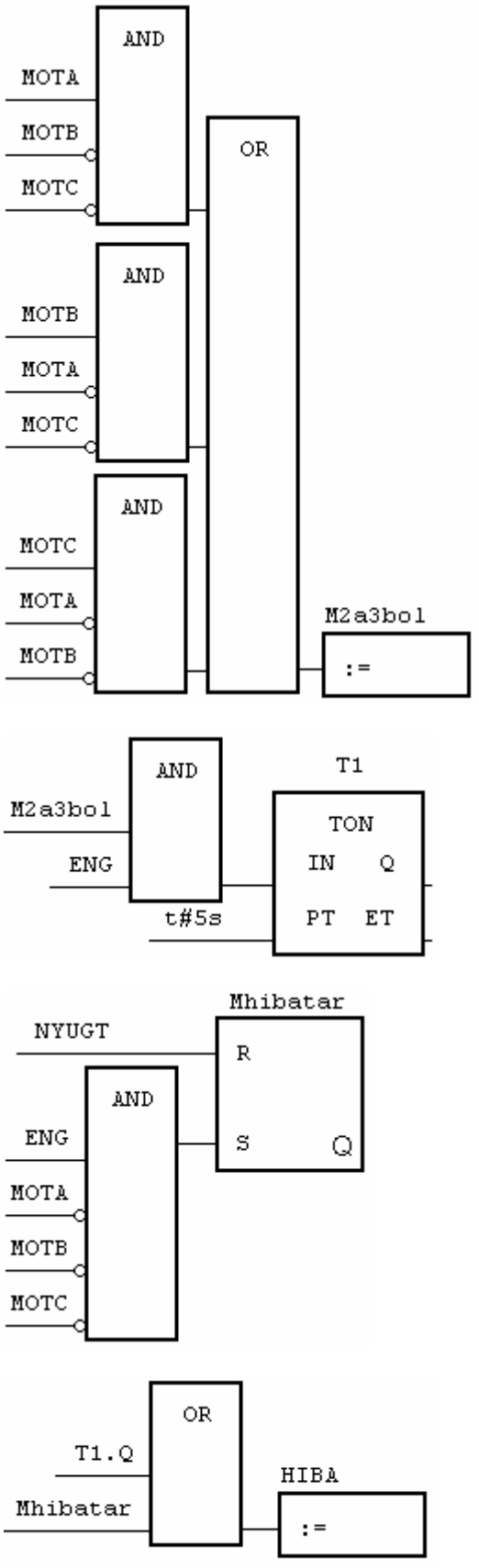
Összerendelési táblázat

Bemenetek	Jel	Logikai összerendelés	Cím
A1 motor fordulatszámjelző	MOTA1	A1 motor fut: MOTA1=1	I0.0
A2 motor fordulatszámjelző	MOTA2	A2 motor fut: MOTA2=1	I1.0
B1 motor fordulatszámjelző	MOTB1	B1 motor fut: MOTB1=1	I0.1
B2 motor fordulatszámjelző	MOTB2	B2 motor fut: MOTB2=1	I1.1
C1 motor fordulatszámjelző	MOTC1	C1 motor fut: MOTC1=1	I0.2
C2 motor fordulatszámjelző	MOTC2	C2 motor fut: MOTC2=1	I1.2
1. csop. engedélyezés kapcsoló	ENG1	1. blokk bekakcsolva: ENG1=1	I0.3
2. csop. engedélyezés kapcsoló	ENG2	2. blokk bekapcsolva: ENG2=1	I1.3
NYUGTÁZÁS nyomógomb	NYUGT	benyomva: NYUGT=1	I0.7
Kimenetek			
HIBAJELZÉS 1	HIBA1	világít, ha: HIBA1=1	Q0.0
HIBAJELZÉS 2	HIBA2	világít, ha: HIBA2=1	Q1.0

A függvényblokk formális paraméterei

A feladatból (input, output): **MOTA, MOTB, MOTC, ENG, HIBA**.
Elrejtett (lokális): T1 időzítő a késleltetéshez: (TON)
3 motor meghibásodás átmeneti tároló: **Mhibatar**
3-ból 2 kiesést jelző segédmerker: **M2a3bol**.

A függvényblokk funkciótervben

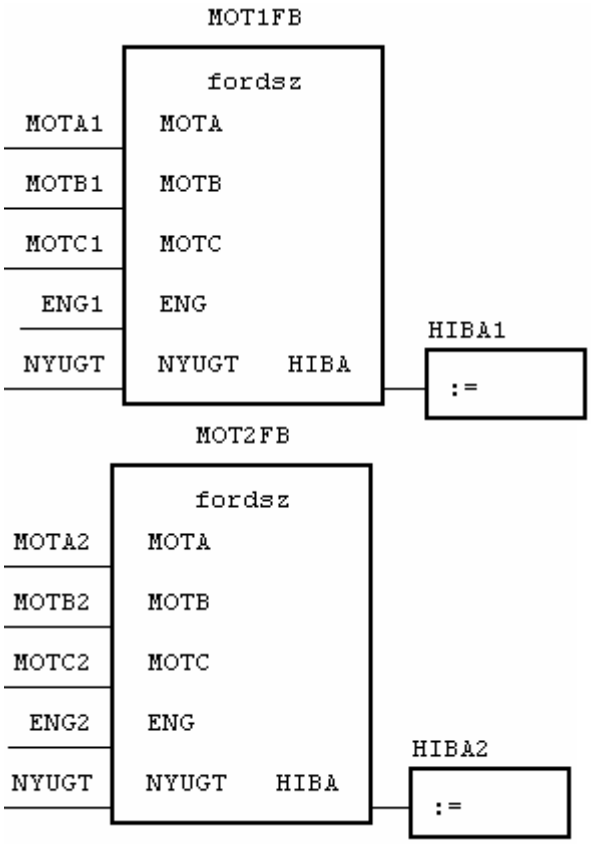


Utasításlista

```
FUNCTION_BLOCK fordsz
VAR_INPUT
        MOTA : BOOL ;
        MOTB : BOOL ;
        MOTC : BOOL ;
        ENG : BOOL ;
        NYUGT : BOOL ;
END_VAR
VAR_OUTPUT
        HIBA : BOOL ;
END_VAR
VAR
        Mhibatar : BOOL ;
        M2a3bol : BOOL ;
        T1 : TON ;
END_VAR

LD      MOTA
ANDN MOTB
ANDN MOTC
OR(     MOTB
ANDN MOTA
ANDN MOTC
)
OR(     MOTC
ANDN MOTA
ANDN MOTB
)
ST      M2a3bol

LD      M2a3bol
AND     ENG
ST      T1.IN
LD      t#5s
ST      T1.PT
CAL     T1

LD      NYUGT
R       Mhibatar

LD      ENG
ANDN MOTA
ANDN MOTB
ANDN MOTC
S       Mhibatar

LD      T1.Q
OR      Mhibatar
ST      HIBA
END_FUNCTION_BLOCK
```

Főprogram

Funkciótervben a függvényblokkok hívása az alábbi módon ábrázolható:



A program utasításlistája

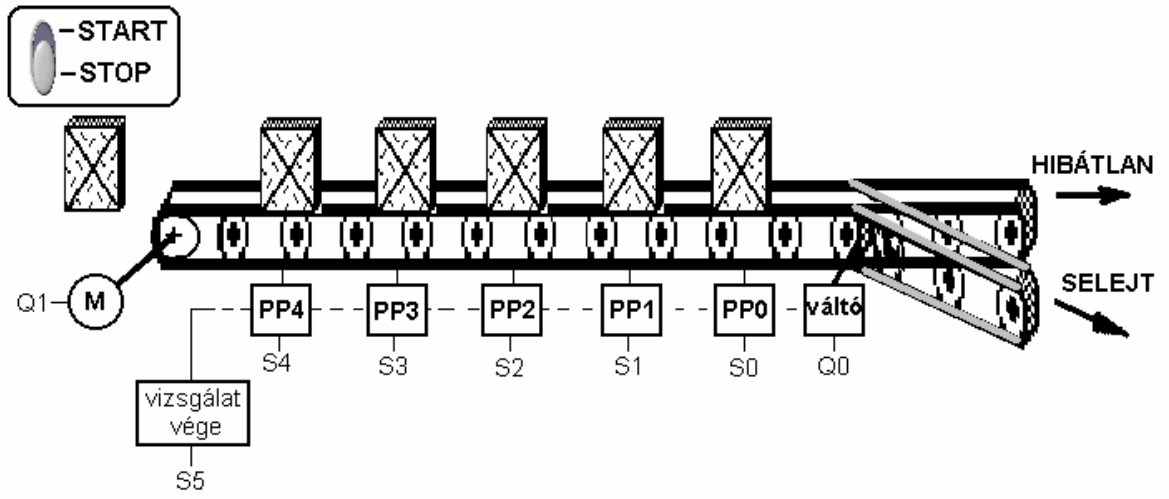
```
PROGRAM motorok

VAR
        MOTA1 AT %I0.0.0.0.0:     BOOL;
        MOTB1 AT %I0.0.0.0.1:     BOOL;
        MOTC1 AT %I0.0.0.0.2:     BOOL;
        ENG1  AT %I0.0.0.0.3:     BOOL;
        MOTA2 AT %I0.0.0.1.0:     BOOL;
        MOTB2 AT %I0.0.0.1.1:     BOOL;
        MOTC2 AT %I0.0.0.1.2:     BOOL;
        ENG2  AT %I0.0.0.1.3:     BOOL;
        HIBA1 AT %Q0.0.0.0.0:     BOOL;
        HIBA2 AT %Q0.0.0.0.1:     BOOL;
        MOT1FB:     FORDSZ;
        MOT2FB:     FORDSZ;
        NYUGT AT %I0.0.0.0.7:     BOOL;
END_VAR

CAL MOT1FB(MOTA:=MOTA1,MOTB:=MOTB1,MOTC:=MOTC1,
                ENG:=ENG1,NYUGT:=NYUGT)
LD      MOT1FB.HIBA
ST      HIBA1

CAL MOT2FB(MOTA:=MOTA2,MOTB:=MOTB2,MOTC:=MOTC2,
                ENG:=ENG2,NYUGT:=NYUGT)
LD      MOT2FB.HIBA
ST      HIBA2
END_PROGRAM
```

Gyakorló feladat: utasításlista elemzése V.

```
PROGRAM MASK1
VAR
        IB0 AT %IB0.0 : BYTE ;
        QB0 AT %QB0.0 : BYTE ;
END_VAR
VAR
        MB10 : BYTE ;
END_VAR
VAR
        IB1 AT %IB1.0 : BYTE ;
        QB1 AT %QB1.0 : BYTE ;
END_VAR
VAR
        MB5 : BYTE ;
END_VAR


LD MB10
XOR IB1
AND MB10
OR QB1
ST QB1

LD IB1
ST MB10
END_PROGRAM
```

Minőségellenőrzés

Egy gyártási folyamat végén a munkadarabokat minőségellenőrzésnek vetik alá. A vizsgáló berendezés 5 db, egymástól egyenlő távolságra lévő ellenőrzőegységből (próbapad) áll (**PP0..PP4**). Szállítószalag gondoskodik a munkadarabok továbbításáról. Egyik részegységtől a másikig 5 s alatt ér a munkadarab. A szállítószalag először a bekapcsolás jelre indul el, majd **S5** hatására, amely akkor jelez, ha az összes próbahelyen befejeződött az ellenőrzés. Ekkor a motor 5s-ig bekapcsol és egy vizsgálóhellyel továbblépteti a munkadarabokat. Ha a vizsgálandó munkadarab hibásnak bizonyul, a vizsgálóegység kimenetén **1**-es jel jelenik meg. A szalag végén a munkadarab a vizsgálatok eredményétől függően vagy a **HIBÁTLAN**, vagy a **SELEJT** irányba halad tovább. (Átváltás a **Q0** jellel.)



45. ábra Minőségellenőrző-sor vezérlése

Összerendelési táblázat

Bemenetek	Jel	Logikai összerendelés	Cím
START/STOP kapcsoló	START	bekapcsolva, ha: SART=1	I1.0
1. próba eredménye	PP0	hibás, ha : PP0=1	I0.0
2. próba eredménye	PP1	hibás, ha : PP1=1	I0.1
3. próba eredménye	PP2	hibás, ha : PP2=1	I0.2
4. próba eredménye	PP3	hibás, ha : PP3=1	I0.3
5. próba eredménye	PP4	hibás, ha : PP4=1	I0.4
vizsgálat vége	S5	továbbléptethet, ha : S5=1	I0.5
Kimenetek			
szalagváltó	Q0	selejt irány: Q0=1	Q0.0
sz.szalag motor	Q1	bekapcsolva, ha: Q1=1	Q0.1

Megoldás: A munkadarabok hibás voltát egy **HIBA** nevű változóban (bájt) tároljuk. Minden munkadarabhoz egy-egy bit tartozik, amelyet a munkadarab léptetésével együtt léptetünk tovább (jobbra). Amikor a munkadarab a vizsgálatok befejeztével a szalag végére kerül, a **HIBA** merkerbájt legkisebb helyiértékű bitje jelzi, hogy valamelyik próbapadon a munkadarab hibásnak bizonyult-e , tehát selejtes, vagy mindegyik vizsgálatnak megfelelt, így a hibátlan darabok közé kerülhet.
A vizsgálat befejeztével S5 jelet ad. Ennek felfutó éle indítja az alábbi vezérlőalgoritmust:

- **HIBA** adatmerker-bájtot 1 helyiértékkel jobbra léptetni;
- az aktuális bemenőjel-bájttal kiegészítjük a hibatárolót (**HIBA**);
- az **HIBA** legalacsonyabb helyiértékű bitjének megfelelően **Q0** kimenet (szalagváltó) beállítjuk vagy töröljük.

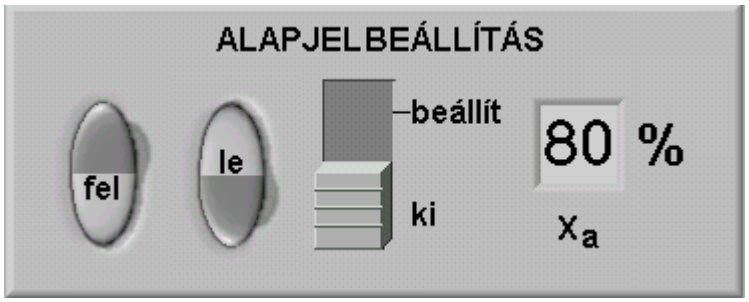
A vezérlőprogram

```
PROGRAM minellen
VAR
        START AT %I0.0.0.1.0:       BOOL;
        EREDM AT %IB0.0.0.0.0:   BYTE;
        KESZ AT %I0.0.0.1.1:        BOOL;
        Q0 AT %Q0.0.0.0.0:  BOOL;
        Q1 AT %Q0.0.0.0.1:  BOOL;
        HIBA AT %MB0.0.0.2.0:    BYTE;
        T1:     TP;
        KESZIMP:    R_TRIG;
        M1:     BOOL;
        M2:     BOOL;
        BEIMP:      R_TRIG;
END_VAR

VAR constant
        T11:    TIME := T#0.5S;
END_VAR
CAL   BEIMP(CLK:=START)

CAL   KESZIMP(CLK:=KESZ)

LD      KESZIMP.IMP
JMPCN       TOVABB

LD      HIBA
SHR   1
OR      EREDM
ST      HIBA
AND   1    (*MASZK*)
EQ      1
ST      Q0   (*SELEJT?*)

TOVABB:
LD      BEIMP.IMP
OR      KESZIMP.IMP
ST      T1.IN
```

```
LD    T11
ST    T1.PT
CAL   T1
LD    T1.Q
ST    Q1

END_PROGRAM
```

Alapjeladó

A kezelő egy szabályozás alapjelét 0..99% között változtathatja, a **fel** illetve **le** nyomógombok segítségével. A beállított alapjel visszajelzésre kerül. Írjuk meg az alapjel beállítását végző programrészletet. (A szabályozás most nem feladatunk, a kiszámított alapjelet más program használja fel.) Az alapjel csak akkor módosítható, ha előzőleg átkapcsolnak beállító üzemmódba. (A kapcsolót esetleg kulccsal is védhetik.)



46. ábra A kezelői felület

Összerendelési táblázat

Bemenetek	Jel	Logikai összerendelés	Cím
beállító kapcsoló	BE	beállítás, ha: BE=1	I1.0
alapjel növelése	FEL	benyomva : FEL=1	I0.0
alapjel csökkentése	LEF	benyomva : LEF=1	I0.1
Kimenet			
alapjel értéke	QXA	alapjel értéke: QXA=byte	Q0.1

Megoldás: A nyomógombok benyomásával T1 időzítő ütemjelére folyamatosan inkrementálható ill. dekrementálható az alapjel értéke 0 és 99 % között. Ha mindkét nyomógombot benyomják, nem történik semmi.

A vezérlőprogram

```
PROGRAM alapjALL
VAR
        FEL AT %I0.0.0.0.0: BOOL;
        LEF AT %I0.0.0.0.1: BOOL;
        BE AT %I0.0.0.1.0:   BOOL;
        QXA AT %QB0.0.0.0.0:       BYTE;
END_VAR

VAR constant
        T11:    TIME := T#0.25S;
END_VAR
```

```
VAR
        T1:     TON;
        ALAPJ:INT;
END_VAR

LD      BE
ANDNT1.Q
ST      T1.IN
LD      T11
ST      T1.PT
CAL     T1
LD      T1.Q
RETCN

LD      FEL
AND     LEF
RETC

LD      FEL
JMPC    NOVEL

LD      LEF
JMPC    CSOKK0
RET

NOVEL:
LD      ALAPJ
EQ      99
RETC

LD      ALAPJ
ADD     1
ST      ALAPJ
JMP     VEGE

CSOKK:
LD      ALAPJ
EQ      0
RETC
LD      ALAPJ
SUB     1
ST      ALAPJ

VEGE:
LD      ALAPJ
INT_TO_BYTE
ST      QXA
RET
END_PROGRAM
```

Tömbök használata a tároló nélküli követővezérlésekben

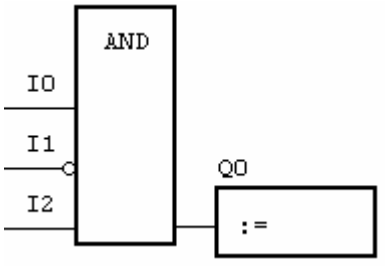
Ha a változók száma nem túl nagy, a tároló nélküli követővezérlés bemenő és kimenő változói közötti kapcsolat függvénytáblázattal leírható. A logikai függvényt nem egyszerűsítjük, hanem soronként felírjuk a függvénykódnak megfelelő digitális számot. Az így kapott vektort egy konstans tömbbe tároljuk. A bemeneteket és kimeneteket nem bitenként, hanem összefüggő bitcsoportként (bájt vagy szó, a változók számától függően) kezeljük. Az éppen aktuális bemeneti érték meghatározza a vektornak azt az elemét, amelynek tartalmát a kimenetre írhatjuk. A főprogramban a be és kimeneti bájtokat maszkoljuk, hogy a PLC be/kimeneteire esetlegesen rákötött egyéb jelek ne befolyásolják a kiolvasott értéket, csak azokat a biteket dolgozzuk fel és írjuk felül, amelyek az adott feladathoz tartoznak.

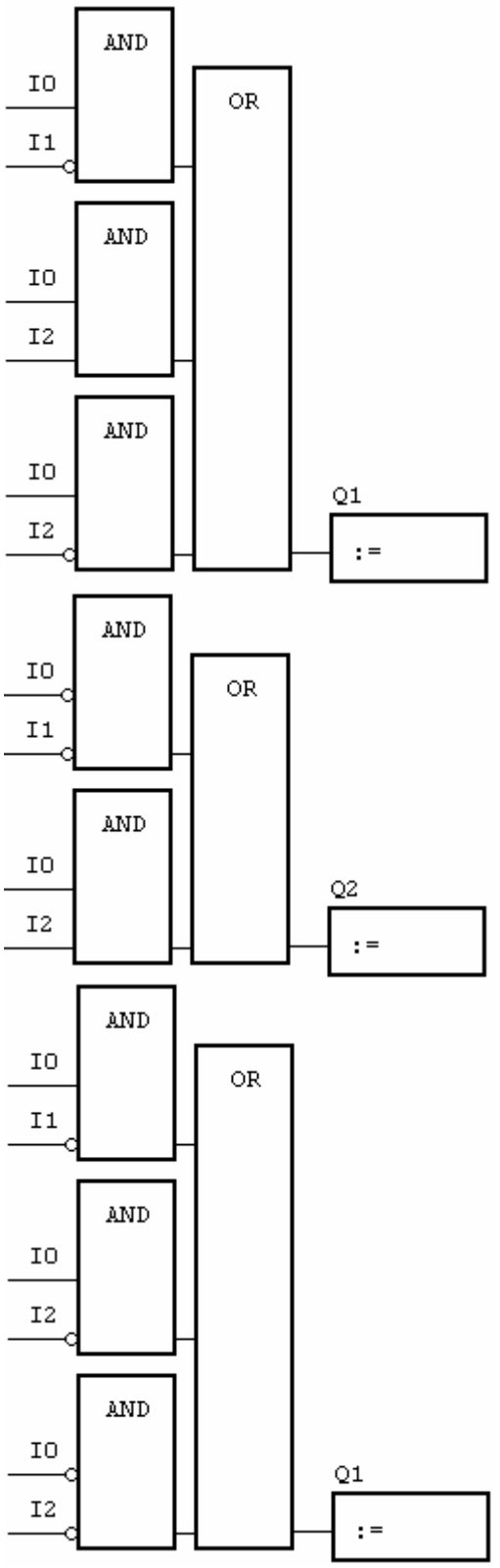
A program egyszerűen módosítható a tömb és a maszkok átírásával.

Összerendelési táblázat

Bemenetek	**Cím**
I0	I0.0
I1	I0.1
I2	I0.2
Kimenetek	
Q0	Q0.0
Q1	Q0.1
Q2	Q0.2
Q3	Q0.3

A be- és kimenetek közötti függvénykapcsolat





A függvénytáblázat

A függvénytáblázat felírásakor ügyeljünk arra, hogy a fizikai címeknek megfelelő növekvő sorrendben vegyük fel a be/kimeneti oszlopokat!

I2	I1	I0	Q3	Q2	Q1	Q0	KIMENET
0	0	0	1	1	1	0	14
0	0	1	1	0	1	0	10
0	1	0	0	0	1	0	2
0	1	1	1	0	0	0	8
1	0	0	0	1	0	0	4
1	0	1	1	1	1	1	15
1	1	0	0	0	0	0	0
1	1	1	0	1	1	0	6

A vezérlőalgoritmus

```
PROGRAM prkovvez
VAR
        IB0 AT %IB0.0.0.0: BYTE;
        QB0 AT %QB0.0.0.0:      BYTE;
        TABLA:      ARRAY[0..7] OF BYTE := [14,10,2,8,4,15,0,6];
END_VAR

VAR constant
        BEMASK:     BYTE := 2#00000111;
        KIMASK:     BYTE := 2#11110000;
END_VAR

VAR
        M0:    INT;
END_VAR

LD      IB0
AND   BEMASK
BYTE_TO_INT
ST      M0

LD      QB0
AND   KIMASK
OR      TABLA[M0]
ST      QB0
END_PROGRAM
```

Feladat: Módosítsa a fenti programot úgy, hogy a mélygarázs szellőzésfelügyletét valósítsa meg!

Tömbök használata ütemvezérelt lefutóvezérléseknél

Egy útkereszteződésben a főút és a mellékutak kereszteződését közlekedési lámpával irányítják. A megadott ütemdiagram alapján kell a piros, sárga, zöld fázist kapcsolgatni. Az időegység (ütemegység) 5s. A berendezést **S0** kapcsolóval lehet bekapcsolni.

Ütemdiagram

ütem:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0
P1										■	■	■	■	■	■	■	
S1									■							■	
Z1	■	■	■	■	■	■	■	■									■
P2	■	■	■	■	■	■	■	■	■	■						■	■
S2										■					■		
Z2											■	■	■	■			

Összerendelési táblázat

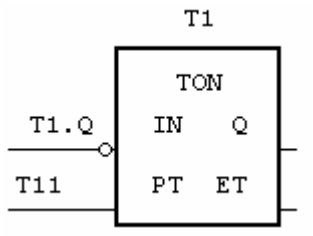
Bemenetek	Jel	Logikai összerendelés		Cím
BE/KI kapcsoló	S0	bekapcsolva:	S0=1	I0.0
Kimenetek				
1. lámpa (főút) piros	P1	világít:	P1=1	Q0.0
1. lámpa (főút) sárga	S1	világít:	S1=1	Q0.1
1. lámpa (főút) zöld	Z1	világít:	Z1=1	Q0.2
2. lámpa (mellékút) piros	P2	világít:	P2=1	Q0,3
2. lámpa (mellékút) sárga	S2	világít:	S2=1	Q0.4
2. lámpa (mellékút) zöld	Z2	világít:	Z2=1	Q0.5

A megoldáshoz felhasználunk egy bekapcsolás-késleltetéses időzítőt, az ütemjel generálására és egy számlálót az ütemek számlálására. A kimeneti byte értékeket egy tömbben tároljuk. A számláló értéke fogja megadni, hogy a tömb hányadik elemét írjuk ki a kimeneti byte-ba.

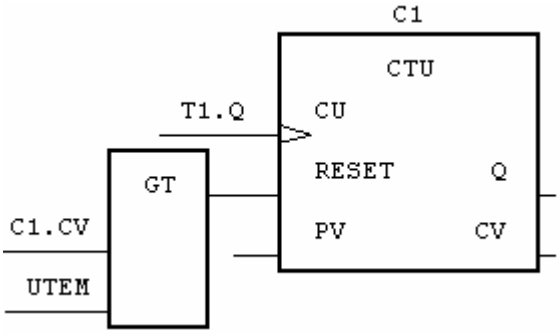
Ha S0=1, a kimenetek ütemezése:

számláló	Z2	S2	P2	Z1	S1	P1	kimeneti BYTE
0	0	0	1	1	0	0	**12**
1	0	0	1	1	0	0	**12**
2	0	0	1	1	0	0	**12**
3	0	0	1	1	0	0	**12**
4	0	0	1	1	0	0	**12**
5	0	0	1	1	0	0	**12**
6	0	0	1	1	0	0	**12**
7	0	0	1	1	0	0	**12**
8	0	0	1	0	1	0	**10**
9	0	1	1	0	0	1	**25**
10	1	0	0	0	0	1	**33**
11	1	0	0	0	0	1	**33**
12	1	0	0	0	0	1	**33**
13	1	0	0	0	0	1	**33**
14	0	1	0	0	0	1	**17**
15	0	0	1	0	1	1	**11**

Az ütemgenerátor



A számláló:



A vezérlőalgoritmus

```
PROGRAM TMBUTEM
VAR
        S0 AT %I0.0.0.0.0:  BOOL;
        QB0 AT %QB0.0.0.0:        BYTE;
```

```
        TABLA:      ARRAY[0..16] OF BYTE :=
[12,12,12,12,12,12,12,12,10,25,33,33,33,33,17,11,11];
        T1:    TON;
        C1:    CTU;
END_VAR

VAR constant
        T11:   TIME := t#2s;
        UTEM:       INT := 15;
END_VAR

VAR
        KIMASK:     BYTE := 2#11000000;
        M0:    BOOL;
        S0IMP:      BOOL;

END_VAR

LD     S0
JMPCN       KIKAPCS

LD     S0
ANDN   M0
ST     S0IMP
LD     S0
ST     M0


LDN    T1.Q
ST     T1.IN
LD     T11
ST     T1.PT
CAL    T1

LD     T1.Q
ST     C1.CU
LD     C1.CV
GT     UTEM
OR     S0IMP

ST     C1.RESET
CAL    C1

LD     QB0
AND    KIMASK
OR     TABLA[C1.CV]
ST     QB0
RET
KIKAPCS:
```

```
LD      0
ST      QB0
RET
END_PROGRAM
```

Feladatok:

A fenti program a kikapcsolás jel hatására azonnal lekapcsolja az összes lámpát. Hogyan módosítaná a programot, ha az lenne a feladata, hogy az adott ciklust még fejezze be?

Módosítsa úgy a fenti programot, hogy kikapcsolás után mindkét irányban a sárga lámpa villogjon 0,5 Hz frekvenciával!

Tervezzen reklámfényt, és az algoritmust a fenti példa szerint írja meg!

Irodalomjegyzék

1. LabView is a registered trademark of National Instruments Corporation
2. Beuschel, J.: Processsteuerungssysteme : Einführung in die Informationsverarbeitung in Automatisierungsanlagen – München; Wien: Oldenbourg, 1994
3. Infoteam OpenPCS programming System Ver. 4.1. User Manual 1. Edition infoteam Software GmbH, D-91088, Bubenreuth. 1996-2001. http://www.infoteam.de
4. Jakoby, W.: Automatisierungstechnik – Algorithmen und Programme: Entwurf und Programmierung von Automatisierungssystemen; Berlin; Heidelberg: Springer, 1997
5. Johns, K.- Tiegelkamp, M.: SPS-Programmierung mit IEC 1131-3 : Konzepte und Programmiersprachen, Anforderungen an Programmiersysteme, Entscheidungshilfen Berlin; Heidelberg: Springer, 1996
6. Sucosoft S 40 Programming Software: Language Elements for PS 4-150/-200/-300 and PS 416 02/00AWB 2700-1306 GB Bonn, Moeller GmbH, 2000
7. Sucosoft S 40 Programming Software: Training Guide 06/990AWB 27-137 GB Bonn, Moeller GmbH, 1999
8. Wellenreuther, G., Zastrow, D.: Steuerungstechnik mit SPS : Bitverarbeitung und Wortverarbeitung, Regeln mit SPS, Von der Steuerungsaufgabe zum Steuerungsprogramm; Braunschweig; Wiesbaden: Vieweg, 1995
9. Wellenreuther, G., Zastrow, D.: Lösungsbuch Steuerungstechnik mit SPS; Braunschweig; Wiesbaden: Vieweg, 1995