LG GLOFA GM7U ÉS A GMWIN 4.0 PROGRAM HASZNÁLATA

készítette

Német Krisztián

Tartalomjegyzék

1 Bevezető...

1.1 Bemutatás

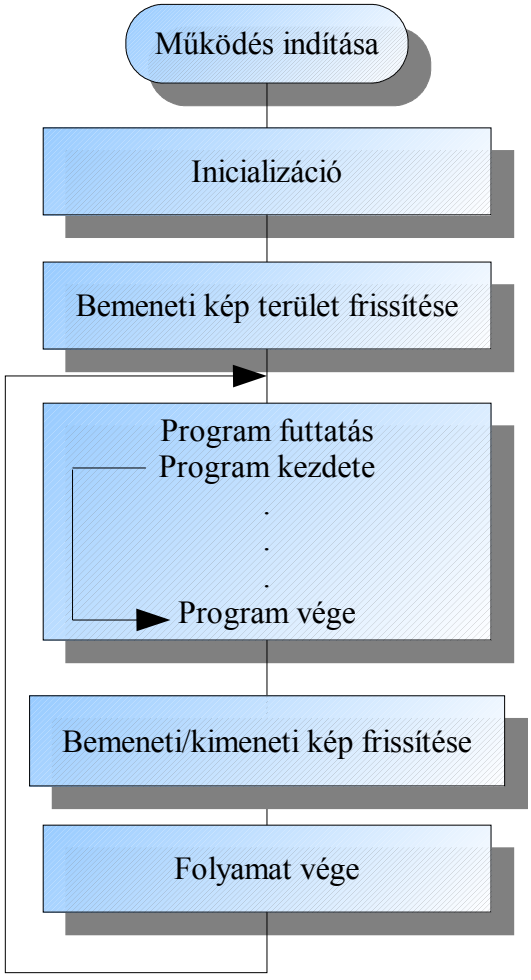
Just to be clear, this section of this template has been created in order to explain some layout elements and, therefore, may be easily deleted when you'll have written your document. This template can be used for whatever kind of documentation or work that needs a complex layout, i.e relations, researches, studies and, why not, books. With few modifications, you can get a template that fulfills your own needs.

1.2 A PLC működési sajátosságai

Működései eljárások

Ciklikus működés

A PLC program sorra az elejétől a végéig hajtja végre a parancsokat lépésről lépésre. Ezt a folyamatot nevezzük szken-nek, illetve a sorrendi feldolgozást ciklikus végrehajtásnak. A PLC ciklikus végrehajtása addig folytatódik, amíg a körülmények meg nem változnak megszakítás feldolgozásra a programfutás során



Az inicializációs lépés csak a készülék feszültségrehelyezésekor, illetve reseteléskor fut le és a következő lépésekből áll:

I/O reset

Hibakeresés

Adatok törlése

I/O címek és típusok meghatározása

A **bemeneti kép terület frissítése** lépés során a bemenetek állapotának a kép területbe történő olvasására kerül sor még a feldolgozás előtt.

A **program futtatás** során a program lefut az első lépéstől az utolsóig.

Bemeneti/kimeneti kép frissítése lépés során a tartalom a kimeneti képben tárolódik

A **folyamat vége** a befejező lépése 1 szkennek, a következő lépésekből tevődik össze:
Hiba keresés
A timer, counter, stb. programelemek jelenlegi értékének (present value)megváltoztatása
Adat kommunikációs lépések lebonyolítsa
Az üzemmód kapcsoló állapotának ellenőrzése.

Idővezérelés

Az idő alapú megszakítás vezérlési eljárás során, a vezérlés nem ismétlődésesen kerül feldolgozásra, hanem minden előre beállított időközökben. A GM7U sorozat esetében az időköz intervalluma 0,001 -től 4 294 967 29 másodpercig terjedő lehet. Ez a vezérlésmód használatos egy állandó ciklus vezérlésére.

Megszakítás vezérlés

A már létező PLC program megszakítható, ha egy művelet végrehajtása sürgőssé válik. Azt a jelet, amely a CPU-t a sűrgősségi állapotról tájékoztatja, megszakító jelnek nevezzük. A GM7U CPU-nak háromfajta megszakításve üzemmódja létezik. Ezek a belső, külső és nagysebességű számláló megszakító jel eljárások

Műveletvégrehajtás pillanatnyi feszültség zavar esetén

Pillanatnyi feszültség zavar akkor lép fel, amikor a tápegység bemeneti feszültsége a megengedett érték alá esik. Ha ez 10 ms alatt lezajlik, a CPU működési állapota változatlan marad. Ha túllépi a 10 ms -ot, akkor a CPU befejezi a feldolgozást, minden kimenetet kikapcsol. Amint a tápellátás helyreáll, a működési rend automatikusan helyreáll.

Szken idő

A feldolgozási időt a 0. lépéstől a következő 0. lépésig szken időnek nevezik. A szken idő a következőkből tevődik össze:

SZKEN IDŐ = SZKEN PROGRAM FELDOLGOZÁSI IDŐ + TASK FROGRAM FELDOLGOZÁSI IDŐ + PLC BELSŐ FELDOLGOZÁSI IDŐ

SZKEN PROGRAM FELDOLGOZÁSI IDŐ: Azon felhasználói program feldolgozási ideje, amely nincs meghatározva task programban
TASK FROGRAM FELDOLGOZÁSI IDŐ: Egy szken alatt a mmegszakításos programok teljes feldolgozási ideje
PLC BELSŐ FELDOLGOZÁSI IDŐ: Hibakeresési idő + I/O frissítési idő + Belső adatfeldolgozási idő + Kommunikációs szolgáltatásokra fordított idő

A szken idő a következő rendszer flag(állapotjelző bit) terülketeken van tárolva:
- _SCAN_MAX: Legnagyobb szken idő (egység: 1 ms)
- _SCAN_MIN: Legkisebb szken idő (egység: 1 ms)

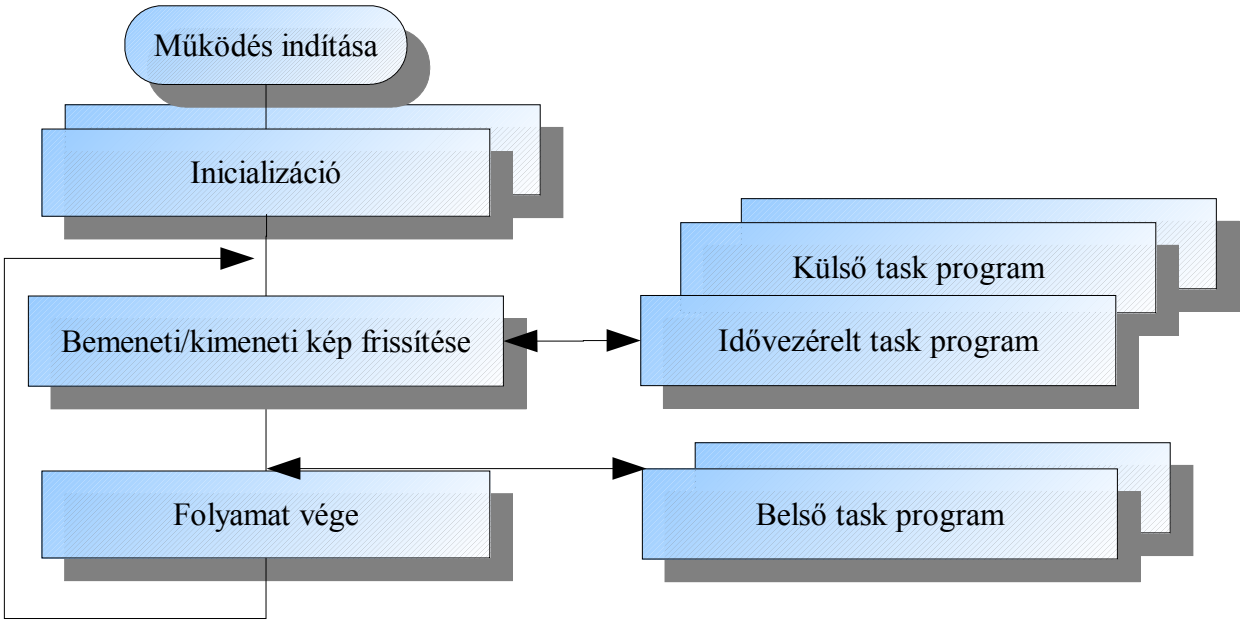
- _SCAN_CUR: Aktuális szken idő (egység: 1 ms)

A program működéséről

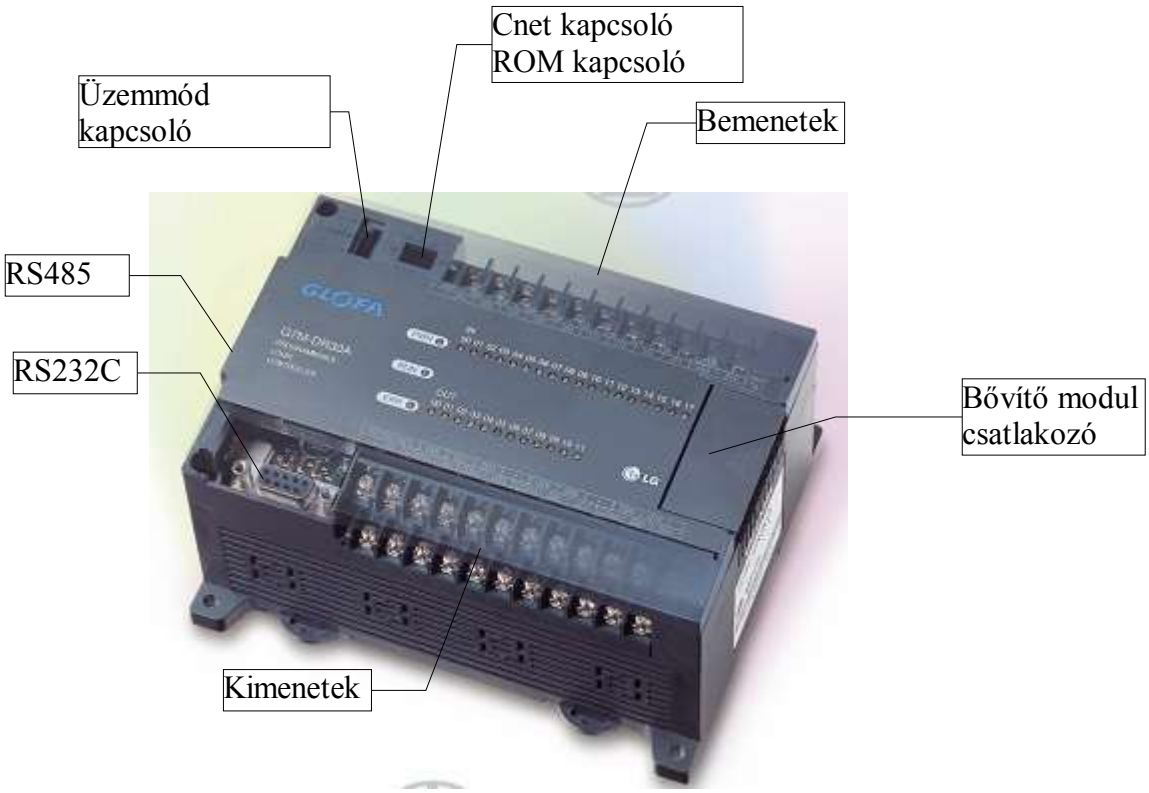
Egy program azokból a függvényelemekből tevődik össze, melyek egy egyéni irányítás megvalósítását lehetővé teszik. A program a CPU RAM memóriájában, vagy az EPROM memóriában tárolódik. A függvényelemek a következő szerint csoportosíthatók:

- Inicializációs program: Akkor fut le, amikor a feszültségre helyezzük a készüléket, illetve ha RUN működésmódba kapcsolunk. Beállítja a főegység, illetve a bővítő modulok paramétereit.

- Szken programban: Feldolgozza az állandóanismétlődő jeleket, minden szken alkalmával.
- Idővezérelt task program:
- Megszakításos program: Ha egy gyors feldolgozás kerül meghívásra belső vagy külső megszakítás által
- Nagysebességű számláló által megszakított program: akkor fut le, amikor a nagysebességű számláló összehasonlító kimeneti feltétele teljesül.



2 A készülék bemutatása



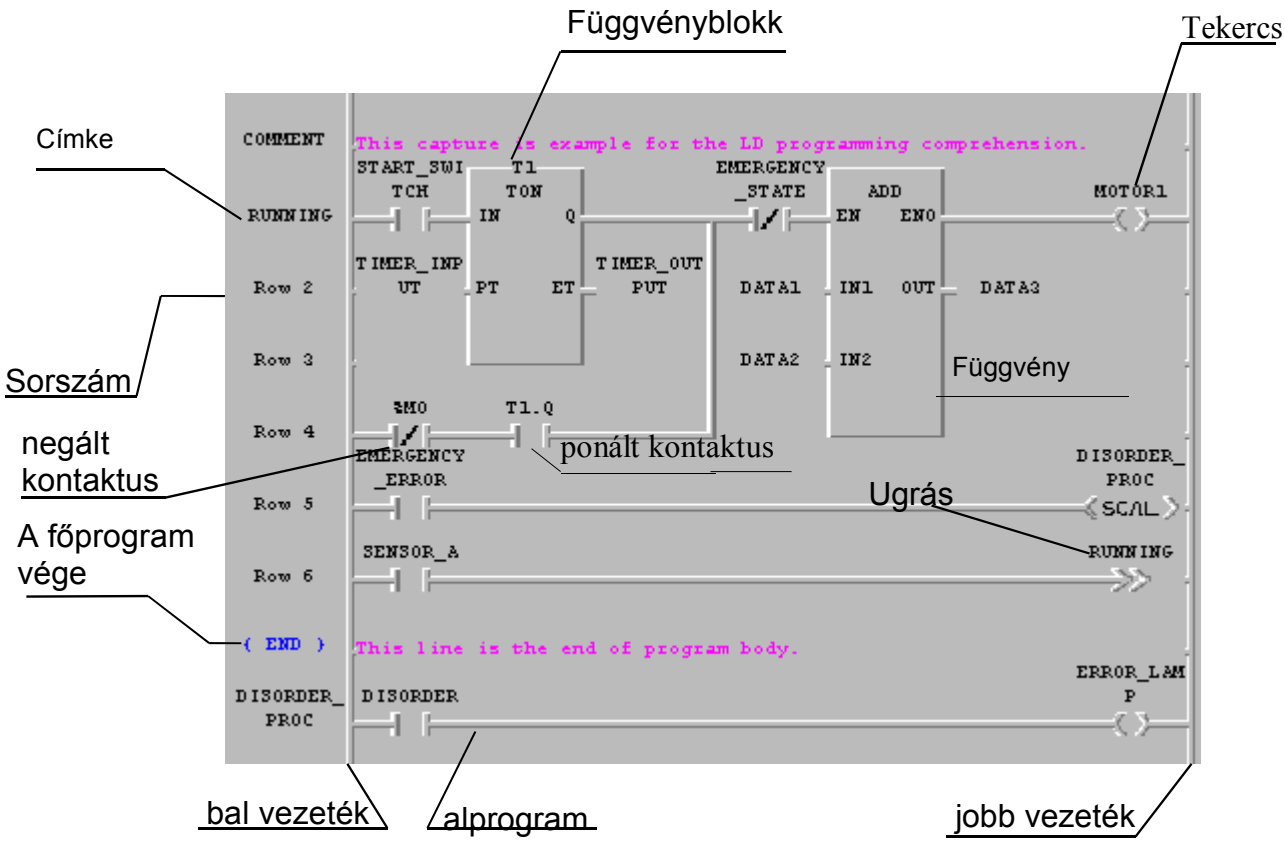
3 A programfelület áttekintése

A továbbiakban bemutatásra kerül egy komplex példa, mely átfogó képet nyújt az eszköztár nyújtotta lehetőségekről, majd az elemeket fogjuk egyenként áttekinteni.

Szerkesztés a Ladder programmal

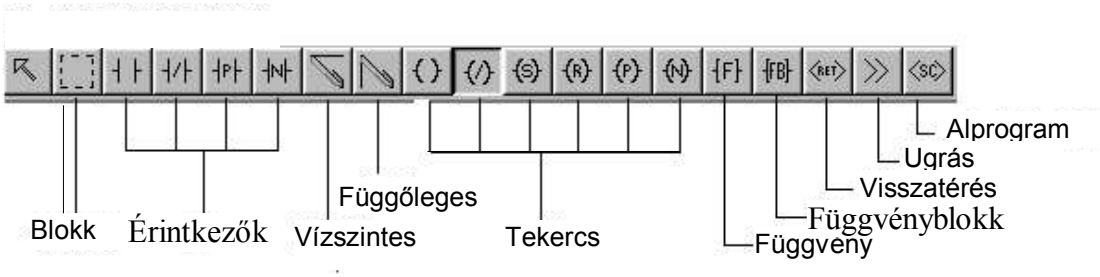
A szerkesztés eszközeinek összefoglalása egy komplex példa segítségével

Az LD program PLC programot hoz létre olyan grafikus szimbólumokkalm mint például érintkezők, tekercsek, melyek a relé logikai ábrákon használatosak. Az alábbi képen a , **'Rung Comment – létrafok megjegyzés '** a megjegyzés az aktuális létrafokra vonatkozóan. A létrafok egy olyan folytonos vonalakból álló egység melyek függőlegesen kapcsolódnak össze. Ilyen például a képen látható létrafok, amely az első sortól az negyedikig terjed(ROW1- ROW4). Az ötödik sor(ROW5) újab sornak számít.



3.1 Az eszköztár (Toolbar)

Az eszköztárban(toolbar) találhatók mindazon elemek, melyek az LD programunk fejlesztéséhez szükségesek.



Táblázat formájában

Ikon	Billentyűparancs	Jelentés
	F2	Ponált kontaktus
	F3	Negált kontaktus
	Shift-F1	Felfutó élre bekapcsoló kontaktus
	Shift-F2	Lefutó élre bekapcsoló kontaktus
	F6	Tekercs
	F7	Fordított tekercs
	Shift-F3	Bekapcsolt/Reteszelt tekercs
	Shift-F4	Törlő/Kireteszelt tekercs
	Shift-F5	Pozitív váltzást érzékelő tekercs
	Shift-F6	Negatív változást érzékelő tekercs
	F4	Vízszintes vonal
	F5	Függőleges vonal
	F8	Függvény
	F9	Függvényblokk
	Shift-F7	End (vége) parancs hívása a fő- vagy alprogram számára
	Shift-F8	Ugrás parancs hívása (az LD program elágazása) - ugrá egy címkére
	Shift-F9	Alprogram hívása

Kontaktus létrehozása

Válassza ki a megfelelő kontaktustt( ,  ,  ,  ,  , )az eszköztárról (T**oolbox**).

- Az LD programablakban mozdítsuk az egeret a célhelyre, majd kattintsunk a bal egérgombbal.
- Amikor rajzolunk egy vonalat( ), miután az egérkurzort a megfelelő helyről elmozdítottuk, tartsuk lenyomva a bal egérgombot és húzzuk a mutatót a létrehozni kívánt vonal utján

- Mozgassa a kurzort arra a helyre, ahová a kontaktust szeretné elhelyezni
- Ezután tegye le a kontaktust az F2:ponált,F3:negált,Shift+F1:pozitív él,Shift+F2:negatív él gombok segítségével.
- szükség esetén: vizszintes vonal: F4; Függőleges vonal: F5

Tekercs létrehozása

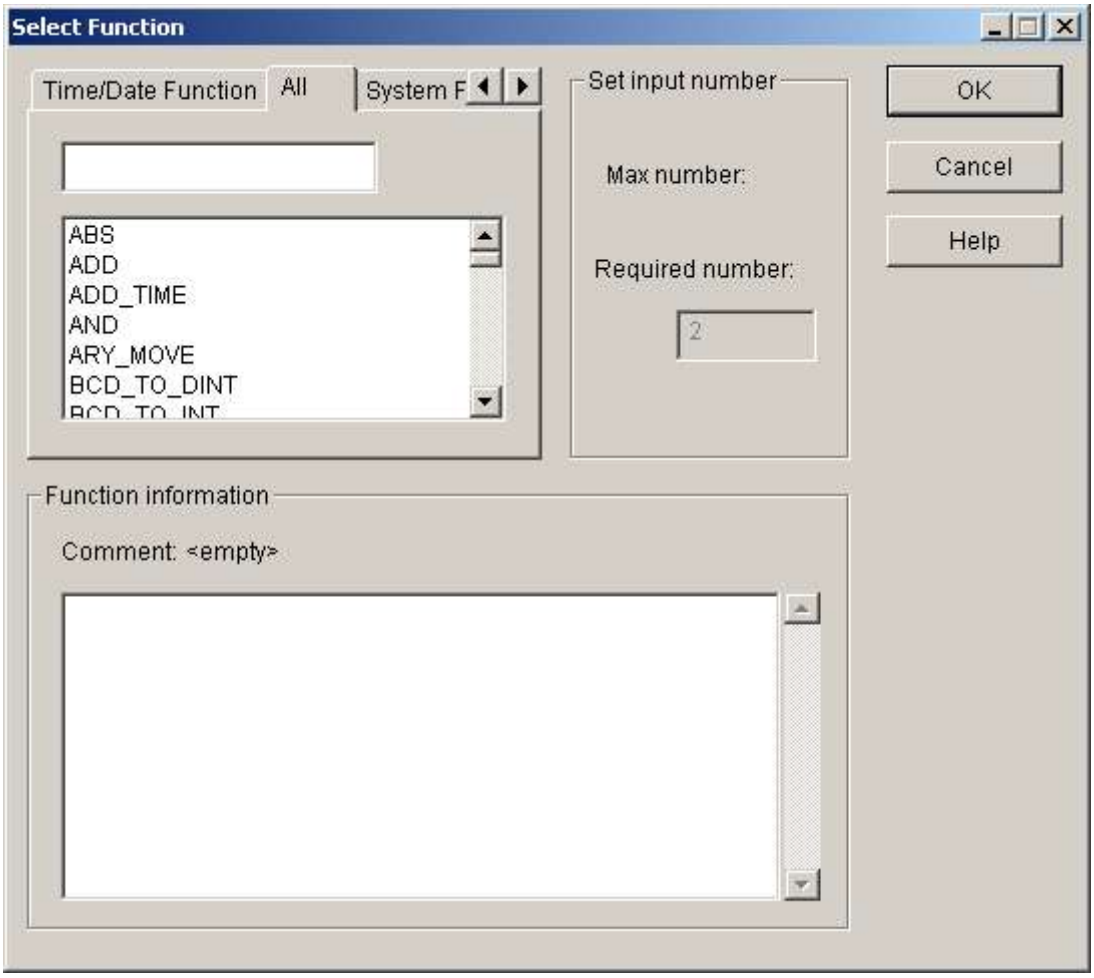
- Válassza ki a megfelelő tekercset( ,  ,  ,  ,  , )az eszköztárról(**Toolbox)**.
- Az LD programablakban mozdítsuk az egeret a célhelyre, majd kattintsunk a bal egérgombbal.

- Az LD programablakban mozdítsuk az egeret a célhelyre, majd kattintsunk a bal egérgombbal.
- Ezutántegye le a kontaktust az F6:ponált, F7:negált, Shft-F3:reteszelt, Shft-F4:kireteszelt, Shft-F5:felfutó élre kapcsoló, Shft-F6:negatív élre kapcsoló gombok segítségével.

Függvény létrehozása

- Válassza ki az {F} ikont az ezköztárból(**Toolbox)**.
- Az LD programablakban mozgassa a kurzort a kívánt helyre, és kattintson a bal egérgombbal.

- Mozgassa a kurzort oda, ahová a kurzort leszeretné tenni.
- Tegye ki a függvényt az F8 gombbal!



Válassza ki a függvénycsoportot fenti fülek közül

- Majd a függvényt a listából
- Majd kattintson a bal egérgombbal az Ok gomra.

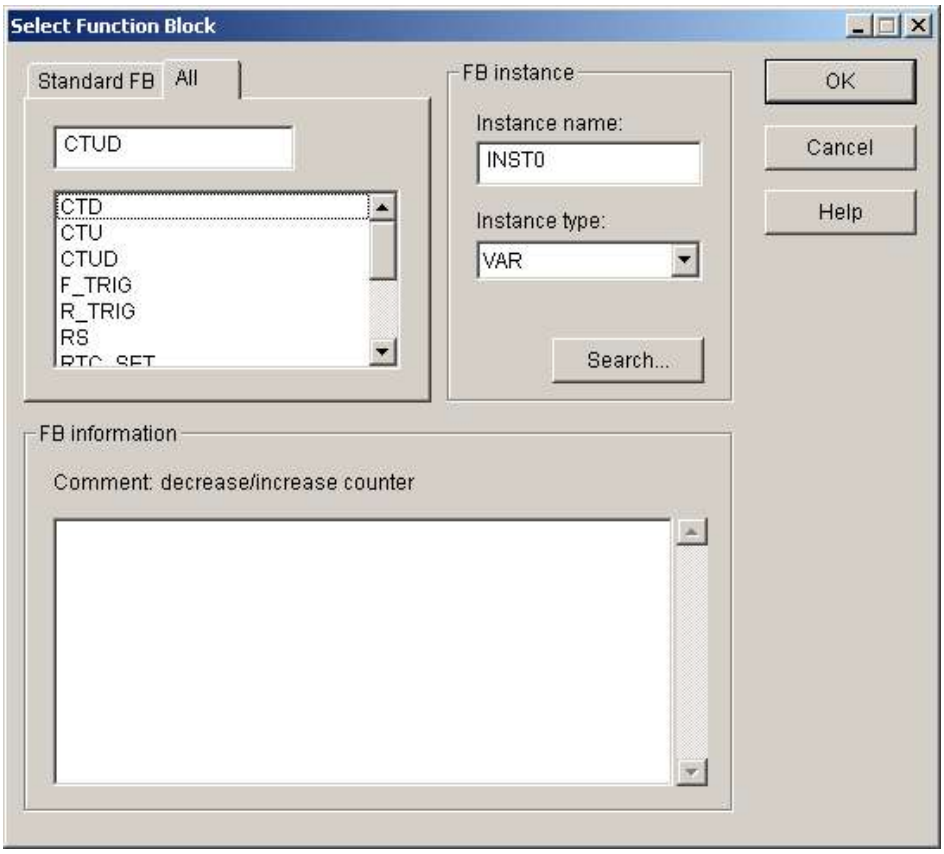
Függvényblokk létrehozása

Válassza ki az ikont az ezköztárból(Toolbox).

- Az LD programablakban mozgassa a kurzort a kívánt helyre, és kattintson a bal egérgombbal.

- Mozgassa a kurzort oda, ahová a kurzort leszeretné tenni.

Tegye le a függvényt az F9 gomb lenyomásával.



- A baloldali listából válassza ki a függvényblokk csoportot!
- Majd adja meg a péda nevet(instance name) és típust (type)
- Nyomja meg az OK gombot

Return(Visszatérés)

Válassza ki a gombot az eszköztárról (toolbox)

- Az LD programablakban mozgassa a kurzort a kívánt helyre, és kattintson a bal egérgombbal.

- Mozgassa a kurzort oda, ahová a kurzort leszeretné tenni.
- Tegye le a függvényt az (Shift-F7) gomb lenyomásával.

Ugrás és Címke

Az LD programban az ugrás (Jump) parancs segítségével a program egy tetszőleges helyére ugorhatunk át. A cél kijelölése címke (label) segítségével történik. A címkének mindig a nulladik oszlopban pozícióban kell állnia.

1) Ugrás (Jump)

- Válassza ki az ikont az eszköztárból (**Toolbox)**.
- Majd tegye le a kívánt helyre (vigye oda a kurzort majd kattintson a bal egérgombbal)

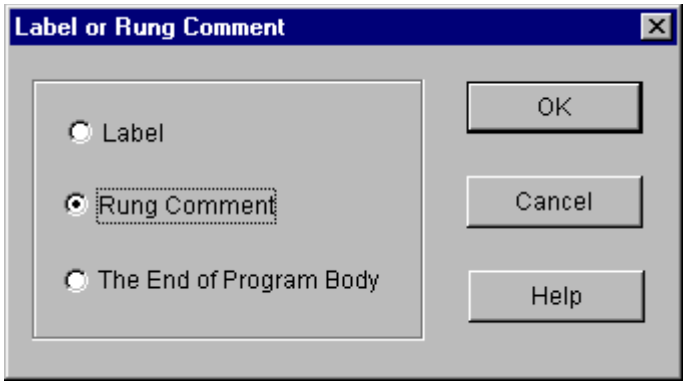
- Vigye a kurzort a megfelelő helyre
- Majd nyomja le a Shft-F8 billentyűkombinációt

2) Címke (Label)

- Válassza ki az ikont az eszköztárból (**Toolbox)**.
- Az LD programablakban kattintson duplán a bal egérgombbal a megfelelő sor első oszlopában majd kattintson egyet szinén a bal egérgombbal
- Válassza ki a **Label/Rung Comment/The End of Program Body** fület.
- Addjon meg a címkenevet az Add Label (címke hozzáadása) dialógusablakban (Maximum 16 karakter)

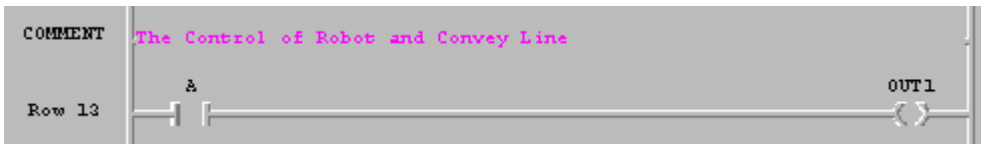
Létrafok megjegyzés (Rung Comment)

- Válassza ki az ikont az eszköztárból(**Toolbox)**.
- Az LD programablakban vigye az egérkurzort a megcímkézendő létrafok 0. oszlopára (a Row 0,1,2,... szövegre) és kattintson duplán a bal egérgombbal. De ha egy létező megjegyzést akar áthelyezni, akkor kattintson rá először duplán, majd szimplán s bal egérgombbal.



- Válassza ki a létrafok megjegyzés (**Rung Comment)** a megjelenő dialógusablakban.
- A megjelenő létrafok megjegyzés (**Rung Comment**) dialógusablakba írhatja be mgjegyzését. (Maximum 170 karakter)

- Az LD programablakban vigye az egérkurzort a megcímkézendő létrafok 0. oszlopára (a Row 0,1,2,... szövegre) majd nyomja le az Enter gombot billentyűzetén
- Válassza ki a létrafok megjegyzés (**Rung Comment)** a megjelenő dialógusablakban.
- A megjelenő létrafok megjegyzés (**Rung Comment**) dialógusablakba írhatja be mgjegyzését. (Maximum 170 karakter)



Alprogram hívás(Subroutine Call)

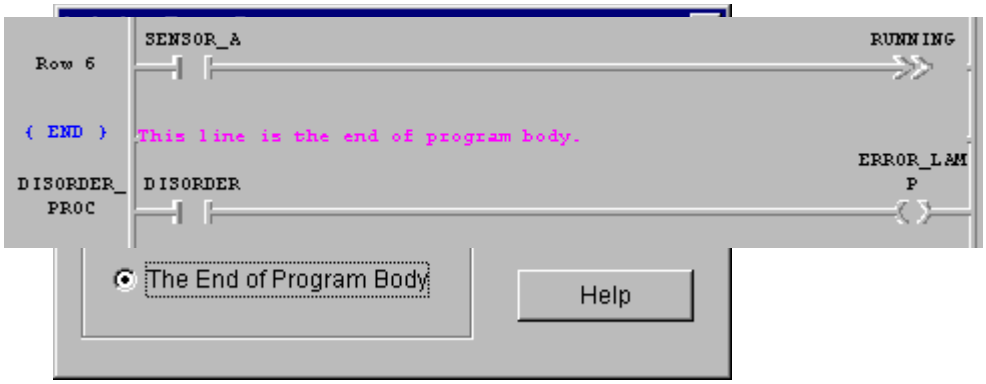
- Válassza ki az  ikont az eszköztárról(**Toolbox)**.
- Mozgassa az egérkurzort a célhelyre, majd tegye le az elemet úgy , hogy kattint a bal egérgombbal!

- Az LD programablakban vigye a kurzort, ahová az alprogramot elkívánja helyezni
- **Nyomja le a** Shft+F9 billentyűkombinációt
- Duplán kattintva a kihelyezett ikonra kiválaszthatjuk a megjelenő listából az alprogramot

Főprogram vége (End of Main Program)

- Válassza ki az  ikont az eszköztárról(**Toolbox)**.
- Kattintson duplán azon sor 0. oszlopára(a Row1,2,3... stb.-re)
- Válassza ki a megjelenő ablakban(**The end of Program Body)** a **Label** or **Rung Comment** -ot.

- Mozgassa a kurzort azon sor 0. oszlopára(a Row1,2,3... stb.-re) ahol a főprogram vége legyen
- Üsse le az Enter billentyűt
- Válassza ki a megjelenő ablakban(**The end of Program Body)** a **Label** or **Rung Comment** -ot.



◆

Blokk szerkesztése (Edit a Block)

Mielőtt blokkot szerkesztenénk, a blokknak szánt tartományt kell meghatározni. Ezt hasonlóm ahhoz, amikor vágólapra másolás majd beillesztés céljából kijelölünk egy szövegrészt dokumentumunkban

- Válassza ki az  ikont az eszköztárbó!
- Az LD programablakban válassza ki a kijelölés kezdőpozícióját
- Majd tartsa nyomva a bal egérgombot és húzza a kurzort a kijelölés végpontjába és engedje el az egérgombot!

- Az LD programablakban válassza ki a kijelölés kezdőpozícióját a nyilak segítségével!
- Majd a **Shift** billentyű nyomvatartása mellett jelölje ki a blokkot a nyilak segítségével!

Kivágás (Cut)

Miután kijelöltük a blokkot

- Válasszuik ki az **Edit-Copy(Ctrl+X,)** parancsot!

Másolás(Copy)

Miután kijelöltük a blokkot

- Válasszuik ki az **Edit-Copy(Ctrl+C,)** parancsot

Beillesztés (Paste)

After copying or cutting a block,

- Vigye a kurzort a beillesztés helyére!
- Válasszuik ki az **Edit-Paste(Ctrl+V,)** parancsot!

Törlés (Delete)

Miután kijelöltük a blokkot

- Válasszuik ki az **Edit-delete(Delete,)**parancsot!

4 A Ladder programozás alapjai

4.1 Ponált és negált kontaktus

Alapvetően két érintkezőtípust különböztetünk meg:

- Ponált: bekapcsolásakor átengedi a rövidrezár
- Negált: bekapcsolásakor szakadásként működik

Változók:

- direkt változók % jellel kezdődnek és az adat jelekkel folytatódnak. A jelölésük a következők szerint történik: % [Elhelyezkedés jel] [Méret jel] n1.n2.n3
- szimbolikus változók (névvel meghatározott változó)

A PLC bemeneteihez, illetve kimeneteihez a következő direkt változók rendelhetők (csak az alap-egység GM7 és GM7U eseteit vizsgáljuk):

Elhelyezkedés jele

No.	Jel	Jelentés
1	I	Bemenet helye
2	Q	Kimenet helye
3	M	Memória helye

Méret jele

No.	Jel	Jelentés
1	X	1 bit méret
2	None	1 bit méret
3	B	Byte (8 bit) méret
4	W	Word (szó) (16 bit) méret
5	D	Double Word (kettős szó) (32 bit) méret
6	L	Long Word (hosszú szó) (64 bit) méret

Pozíció számok

No.	I, Q	M *
n1	Alaplap sorszáma (0-tól kezdődik)	n1 értéke a [méret jel] –től függ (0-tól kezdődik)
n2	Kártyahely sorszáma (0-tól kezdődik)	n1 változó n2. bitje (0-tól kezdődik)
n3	n3 értéke a [méret jel] –től függ (0-tól kezdődik)	Nincs használva

** A belső változóknál nem értelmezhető az alaplap és kártyahely sorszáma.*

Példa:

%QX3.1.4 vagy **%Q3.1.4** : Kimenet, bit (1 bit) 3. alaplap, 1. kártyahely, 4. kimenet (1 bit)

%IW2.4.1 : Bemenet, szó (16 bit) 2. alaplap, 4. kártyahely, 1. szó (16 bit)

%MD48 : Memoria, dupla szó (32 bit), 48. memoria rekesz (32 bit)

%MW40.3 : Memoria szó (16 bit) 40. memoria rekesz *

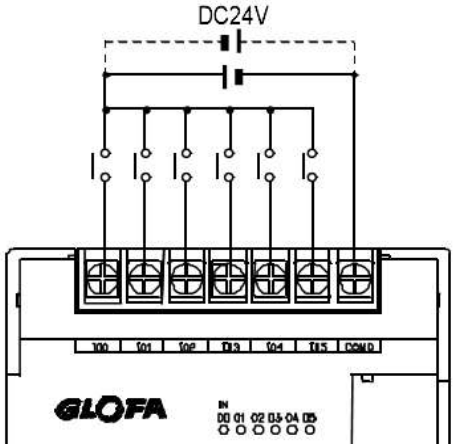
**A belső változóknál nem értelmezhető az alaplap és kártyahaly sorszáma.*

- ▷ Kis betű nem használható a jelölésnél.
- ▷ Ha nem használunk méret jelet akkor bit-es változóként értelmezi a program (lásd táblázat).
- ▷ Közvetlen változók (Direct variable) meghatározás nélkül is használhatók.

Bemenetek:

Bemenet	Változó
I00	%IX0.0.0
I01	%IX0.0.1
I02	%IX0.0.2
I03	%IX0.0.3
I04	%IX0.0.4
I05	%IX0.0.5
I[n]	%IX0.0.n

A bemenetek bekötése a belső tápegység felhasználásával úgy történik, hogy összekötjük COM0 bemenetet a 24G jelű csatlakozóval, illetve a 24V csatlakozót bevezetjük abba a kapcsolóba, avagy szenzorba amit működtetni szeretnénk, majd onnan a használni kívánt bemenetre (például I00).



Kimenetek:
A kimenetek esetében a COM és a Q csatlakozók közt van kontaktus, ha a hozzátartozó változó logikai 1 értéket kap ennek értelmében úgy köthetjük rá a kapcsolandó áramkört, avagy beavatkozó egységet. Például a Q00 kimenethez a %Q0.0.0 direkt változó tartozik, a kontaktus a Q00 és COM0 érinkezők közt értendő.

Kimenet	Változó
Q00	%QX0.0.0
Q01	%QX0.0.1
Q02	%QX0.0.2
Q03	%QX0.0.3

Kimenet	Változó
Q04	%QX0.0.4
Q05	%QX0.0.5
Q[n]	%I0.0.n

Első program

Készítsünk programot , ami egyszerű kapcsolóként működik, tehát egy kapcsoló (ez legyen I0 helyre bekötve) bekapcsolásakor ráküldi a jelet egy kimenetre (Q0).

1. Első lépésben készítsünk egy új LD programot első program néven a GMWIN szoftver használatáról szóló fejezetben leírtak szerint.

2. Ezután a toolbar eszköztárról helyezzük ki egy ponált kontaktust, illetve egy tekercset (kimenet)  
    1. Kattintsunk 1x az ikonra
    2. Majd 1x a célhelyére
3. Definiáljuk a be-, illetve a kimenetek jelölésére szolgáló változókat. Ehhez:
    1. kattintsunk duplán egyaránt a ponált kontaktusra, illetve a tekercs kimenetre
    2. A megjelenő ablakba rögtön be is írhatjuk a direkt változókat
        1. kontaktus esetében: %IX0.0.0
        2. tekercs esetében: %QX0.0.0

A munkánk helyességét a compile menü compile parancsával ellenőrízhetjük. Ha hibát (error) tartalmaz, akkor kiírja, hány hiba van, illetve piros vonal jelenik meg a hiba helyén. A figyalmeztetések (warnings) a programban hagyott fölösleges változókról adnak tájékoztatást.

Próbáljuk ki programunkat a PLC-n is! Az Online menüben található Connect+Write+Run+Monitor On parancsra kattintással (vagy Ctrl+R gyorsbillentyű, avagy  ikon).

Végezetül mentsük el programunkat Project--> Save Program -->Save!

Mindehhez előfeltétel, hogy a PLC csatlakoztatva legyen a számítógéphez az RS232C (COM1 vagy COM2) portra a GMWIN programban ez a port legyen megadva (alapértelmezés COM1), illetve a port a számítógép BIOS -ában be legyen kapcsolva (alapértelmezés szerint be van kapcsolva). Ezen kívül a PLC kapcsolója középső állásban álljon (PAUSE/REM)

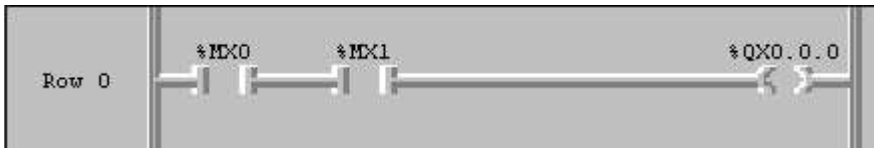
Második program

Készítsünk programot ami két kapcsolóból álló ÉS logikai kapuként működik. Legyen a két kapcsoló elhelyezkedési helye a memóriában (%MX0, %MX1), illetve a kimenet Q0.

1. Első lépésben készítsünk egy új LD programot második program néven a GMWIN szoftver használatáról szóló fejezetben leírtak szerint.

2. Ezután a toolbar eszköztárról (avagy az F2 billentyű megnyomásával) helyezzük ki egymás mellé az első sorba (ROW0) a ponált kontaktusokat, illetve egy tekercset (kimenet) 
   (a tekercs gyorsbillentyűje: F6)

3. Definiáljuk a be-, illetve a kimenetek jelölésére szolgáló változókat. Ehhez:
   1. kattintsunk duplán egyaránt a ponált kontaktusra, illetve a tekercs kimenetre
   2. A megjelenő ablakba rögtön be is írhatjuk a direkt változókat
      1. kontaktusok esetében: %MX0, illetve %MX1
      2. tekercs esetében: %QX0.0.0



Harmadik program

Készítsünk programot ami akkor zár egy tekercset, amikor nem érkezik rá jel!

1. Első lépésben készítsünk egy új LD programot harmadik program néven a GMWIN szoftver használatáról szóló fejezetben leírtak szerint.

2. Ezután a toolbar eszköztárról (avagy az F3 billentyű megnyomásával) helyezzük ki egy negált kontaktust az első sorba (ROW0), illetve egy tekercset (kimenet)

(a tekercs gyorsbillentyűje: F6)

3. Definiáljuk a be-, illetve a kimenetek jelölésére szolgáló változókat. Ehhez:
    1. kattintsunk duplán egyaránt a negált kontaktusra, illetve a tekercs kimenetére
    2. A megjelenő ablakba rögtön be is írhatjuk a direkt változókat

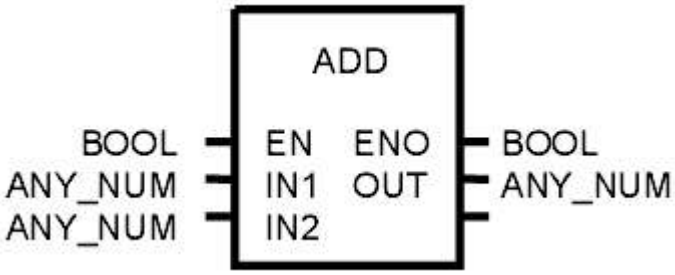
        kontaktus esetében: %IX0.0.0

        tekercs esetében: %QX0.0.0

4.2 Aritmetikai függvények

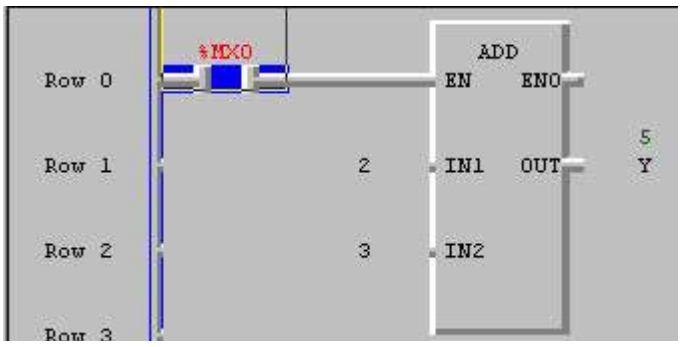
A függvények létrehozásával kapcsolatban a 3.1 fejezet “Függvény létrehozása” pontjában kaphat felvilágosítást. Az összes függvény esetén EN bemenet a függvény lefutásának a feltétele, illetve az esetek többségében EN0-ra EN értéke egyszerűen átmásolódik, avagy EN0 1 értéket vesz fel abban az esetben, ha nincs hiba.IN(1..8) a bemeneteket jelöli melyekre különböző típusú változókat kell csatolni, OUT pedig a függvény végeredménye OUT=f(IN1,IN2,...IN16), ha EN=1.

ADD függvény

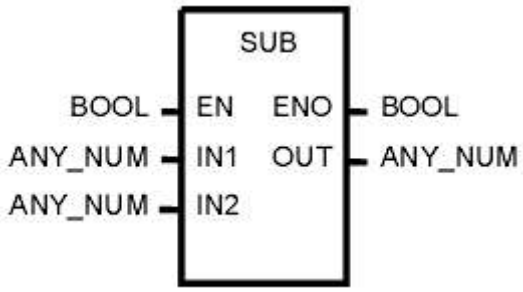


Az ADD függvény segítségével numerikus értékeket(IN1, IN2,..., I8) addhatunk össze(OUT), amennyiben az EN bemenetre logikai egy érkezik. Figyelni kell arra, hogy IN1, IN2 és OUT azonos típusú legyen. A bemenetek száma nyolcig bővíthető.

Negyedik program – az ADD függvény bemutatása

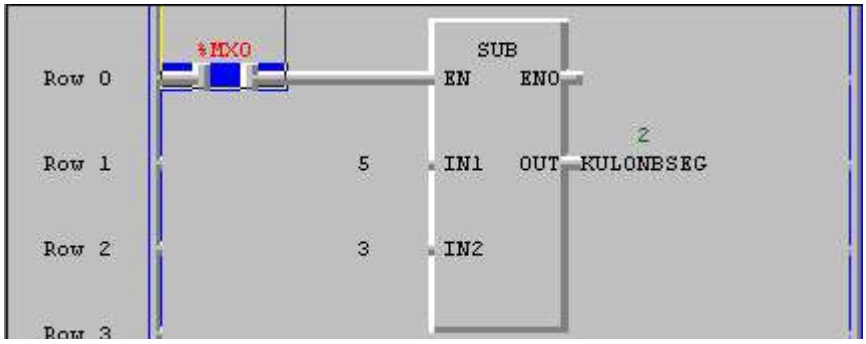


SUB (kivonás) függvény

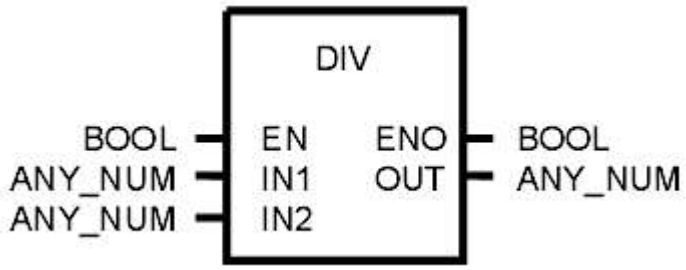


Az ADD függvény segítségével numerikus értékeket(IN1, IN2) vonhatunk ki egymásból(OUT), amennyiben az EN bemenetre logikai 1 érkezik. Figyelni kell arra, hogy IN1, IN2 és OUT azonos típusú szám legyen. EN0 abban az esetben 1, ha semmi hiba nem jelentkezik. OUT = IN1 - IN2

Ötödik program – a SUB függvény bemutatása



DIV függvény



A DIV függvény segítségével numerikus értékeket(IN1, IN2) oszthatunk el egymással; eredmény: OUT, amennyiben az EN bemenetre logikai egy érkezik. Figyelni kell arra, hogy IN1, IN2 és OUT számlegyen. OUT = IN1/IN2.

Hatodik program – a DIV függvény bemutatása

Input(IN1) : VALUE1(INT) = 300(16#012C)

(IN2): VALUE2(INT) = 100(16#0064)

Output(OUT) : OUT_VAL(INT) = 3(16#3)

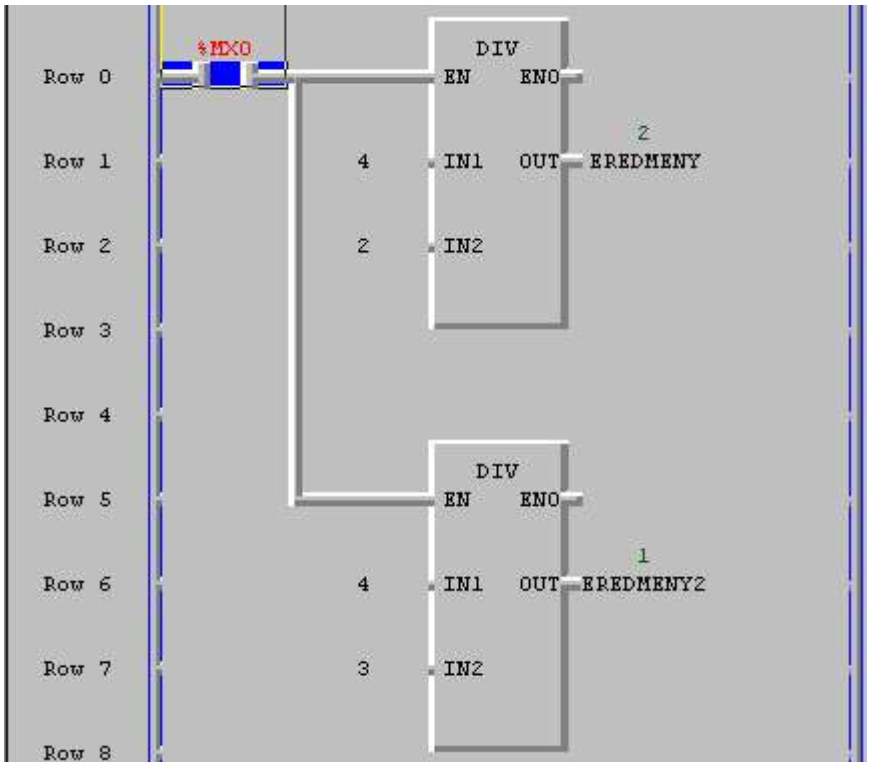
0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

/ (DIV)

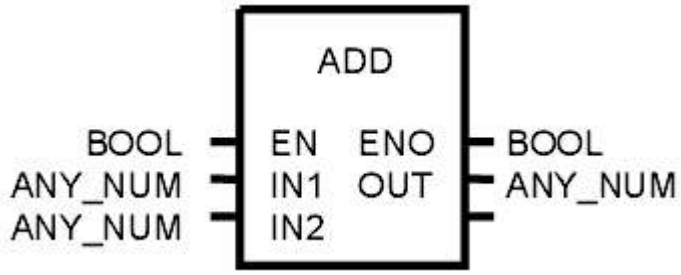
0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

↓

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1



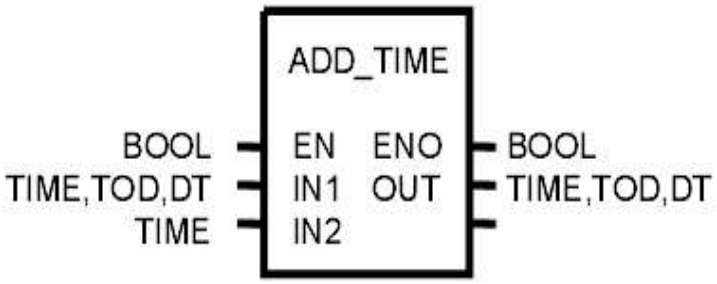
MUL függvény



Az ADD függvény segítségével numerikus értékeket(IN1, IN2,..., I8) addhatunk össze(OUT), amennyiben az EN bemenetre logikai egy érkezik. Figyelni kell arra, hogy IN1, IN2 és OUT azonos típusú legyen. A bemenetek száma nyolcig bővíthető.

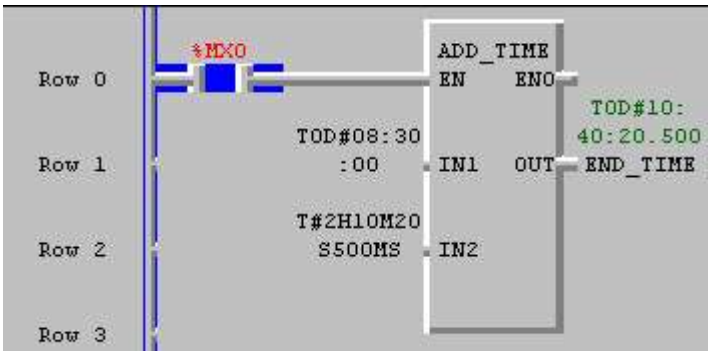
Hetedik program – a MUL függvény bemutatása

ADD_TIME függvény

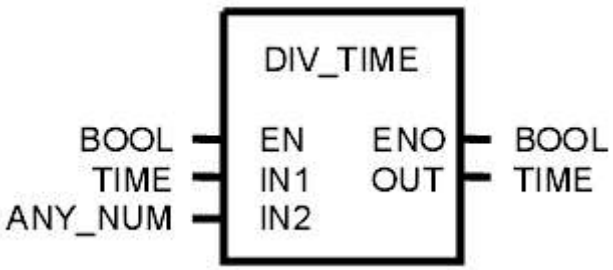


Az ADD_TIME függvény segítségével az (IN1..IN8) bemenetre adott TIME, TOD illetve DT típusú változókat addhatjuk össze, eredmény: OUT. Természetesen csak akkor jelenik meg az eredmény a kimeneten, ha az EN bemenetre logikai 1 érkezik. OUT = IN1+IN2+...+IN8

Nyolcadik program – az ADD_TIME függvény bemutatása

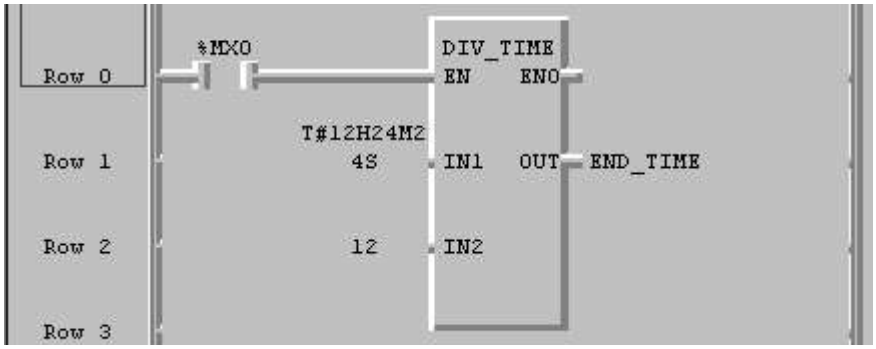


DIV_TIME függvény

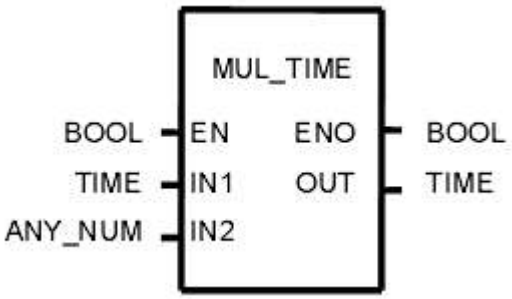


A DIV_TIME függvény segítségével az IN1bemenetre adott TIME, TOD illetve DT típusú változót oszthatjuk el az IN2 -re adott számmal, eredmény: OUT (TIME) . Természetesen csak akkor jelenik meg az eredmény a kimeneten, ha az EN bemenetre logikai 1 érkezik. OUT = IN1/IN2

Kilencedik program – a DIV_TIME függvény bemutatása

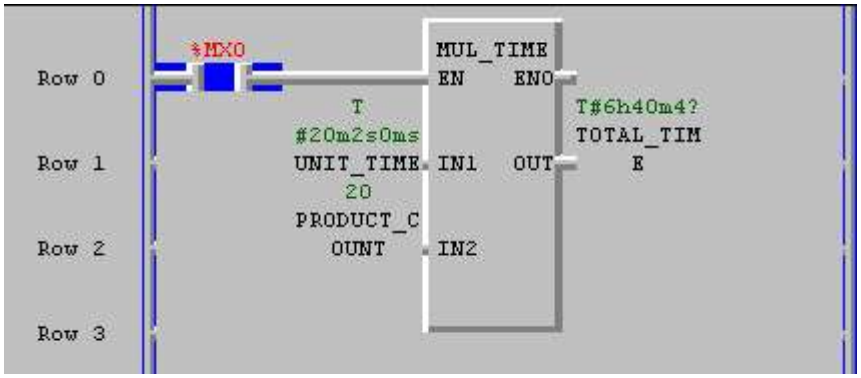


MUL_TIME függvény



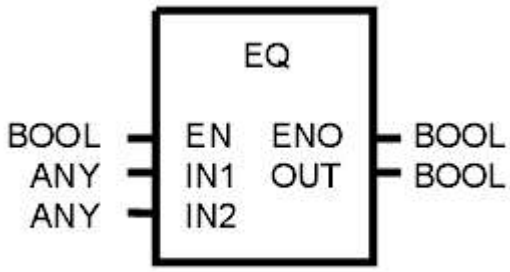
A MUL_TIME függvény segítségével az IN1bemenetre adott TIME, TOD illetve DT típusú változót szorozhatjuk meg az IN2 -re adott számmal, eredmény: OUT (TIME) . Természetesen csak akkor jelenik meg az eredmény a kimeneten, ha az EN bemenetre logikai 1 érkezik. OUT = IN1*IN2

Tizedik program – a MUL_TIME függvény bemutatása



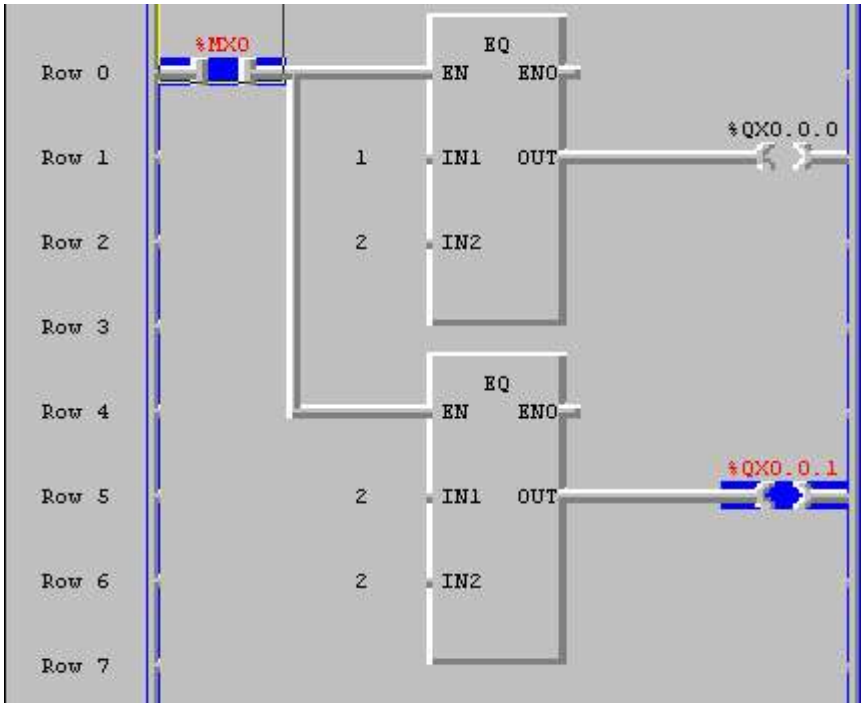
4.3 Összehasonlító függvények

EQ függvény

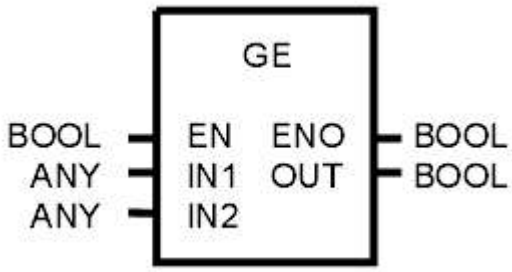


Az EQ függvény segítségével bármely (IN1..IN8) bemenetekre adott változókat hasonlíthatunk össze, eredmény: OUT (Bool) , amennyiben az EN bemenetre logikai egy érkezik. Figyelni kell arra, hogy IN1, IN2, INn(n egész szám) azonos típusú legyen. A bemenetek száma nyolcig bővíthető. EN0 EN értékét veszi fel.

Tizenegyedik program – a EQ függvény bemutatása

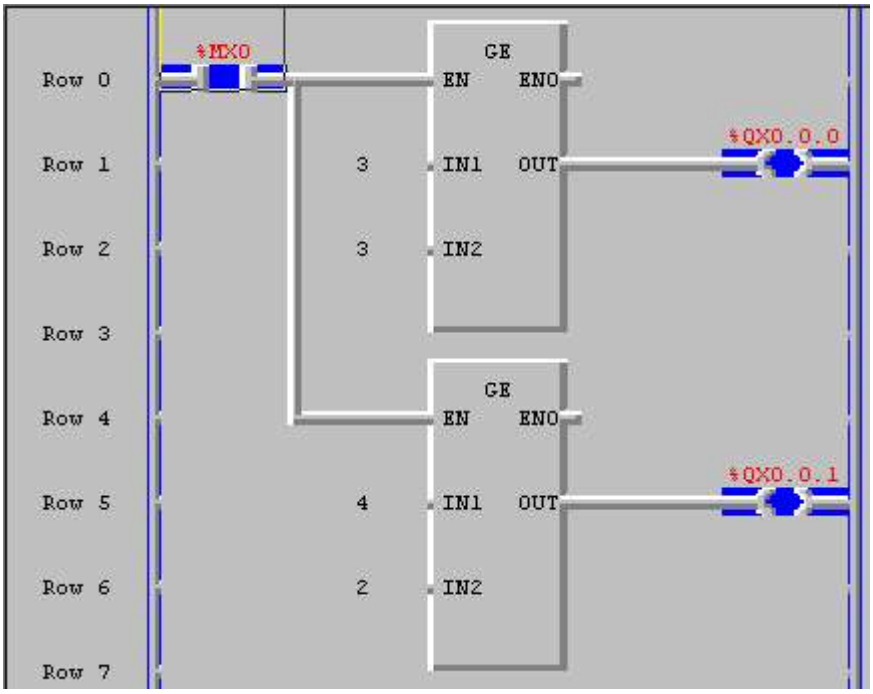


GE függvény

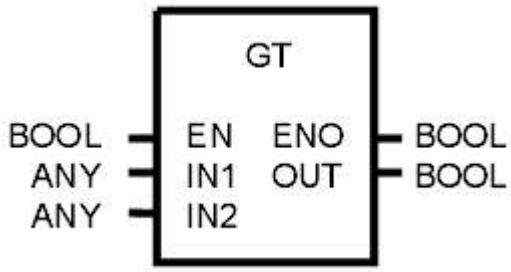


A GE függvény segítségével numerikus értékeket(IN1, IN2,..., IN8) hasonlíthatunk össze. A GE függvény valójában nem más mint a “Nagyobb vagy egyenlő” reláció függvényváltozata. a bemenetekre(IN1..IN8) érekező változók bármilyen, de azonos típusúak lehetnek. A kimenet Bool típusú. Tehát ha EN=1 és IN1≥N2≥...≥INn (n egész szám max. 8), akkor OUT=1. Minden más esetben OUT=0

Tizenkettedik program – a GE függvény bemutatása

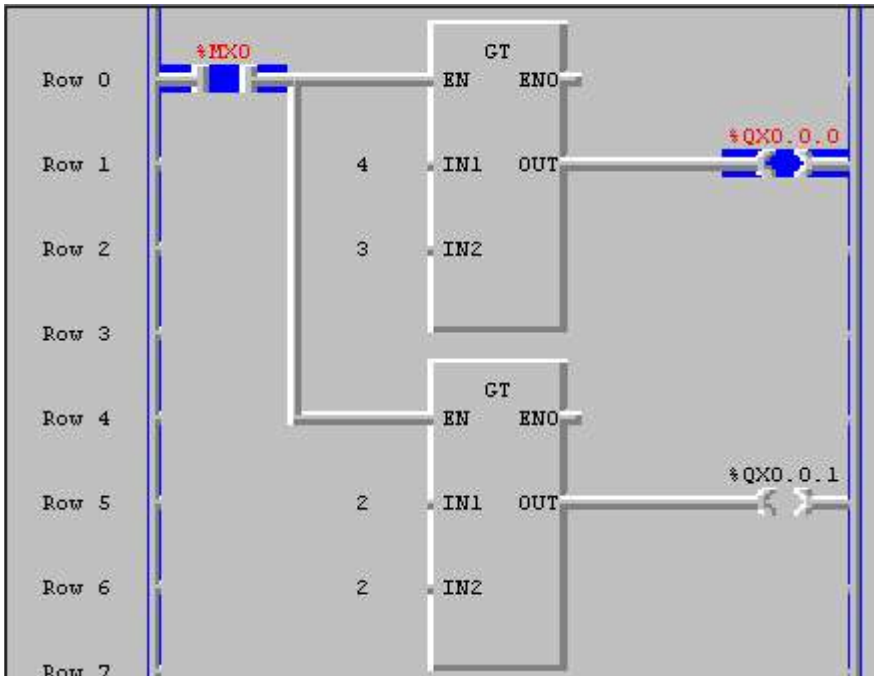


GT függvény

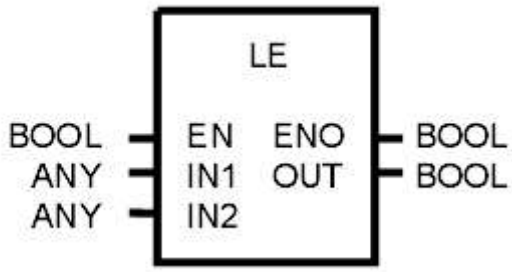


A GE függvény segítségével numerikus értékeket(IN1, IN2,..., IN8) hasonlíthatunk össze. A GE függvény valójában nem más mint a “Nagyobb” reláció függvényváltozata. a bemenetekre(IN1..IN8) érekező változók bármilyen, de azonos típusúak lehetnek. A kimenet Bool típusú. Tehát ha EN=1 és IN1>N2>...>INn (n egész szám max. 8), akkor OUT=1. Minden más esetben OUT=0

Tizenharmadik program – a GT függvény bemutatása

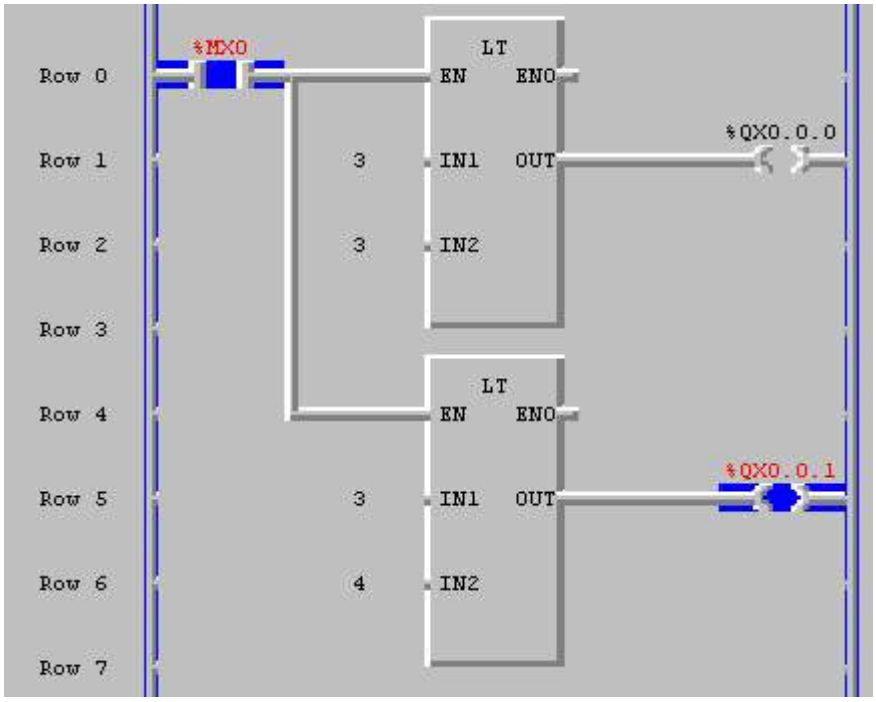


LE függvény

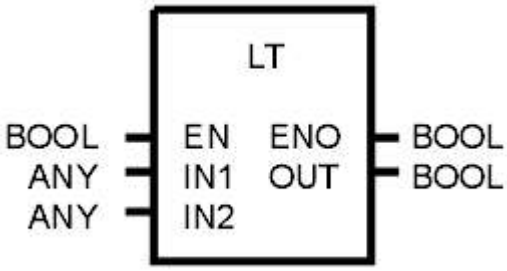


Az LE függvény valójában nem más mint a “Kisebb vagy egyenlő” reláció függvényváltozata. a bemenetekre(IN1..IN8) érekező változók bármilyen, de azonos típusúak lehetnek. A kimenet Bool típusú. Tehát ha EN=1 és IN1≤IN2≤...≤INn (n egész szám max. 8), akkor OUT=1. Minden más esetben OUT=0

Tizennegyedik program – az LE függvény bemutatása

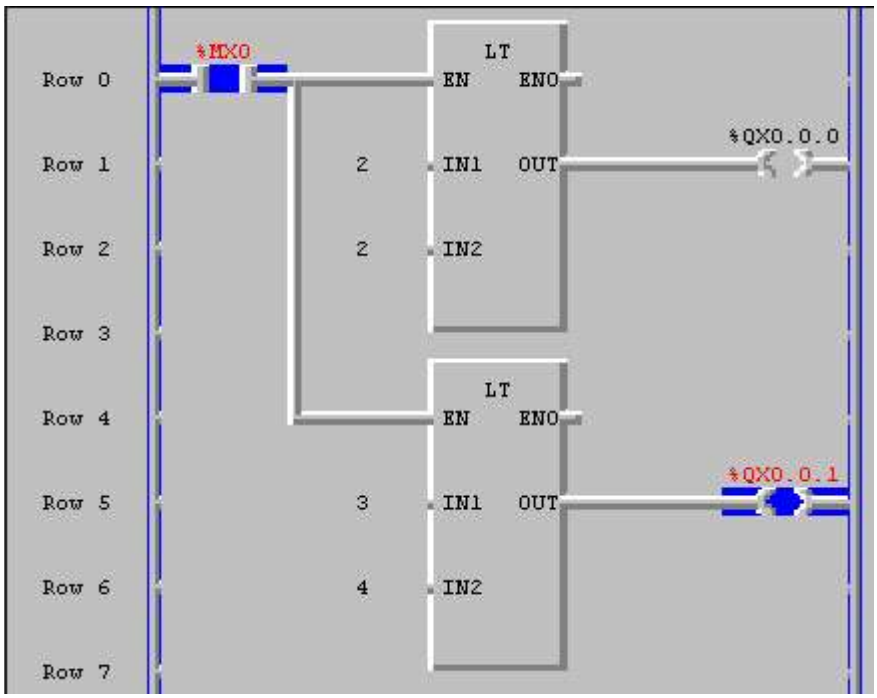


LT függvény

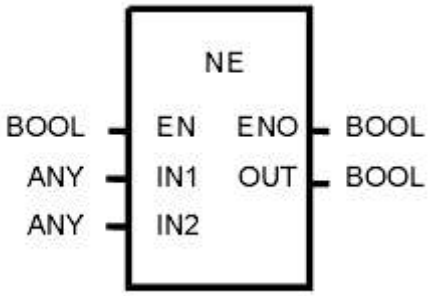


Az LT függvény valójában nem más mint a “Kisebb” reláció függvényváltozata. a bemenetekre(IN1..IN8) érekező változók bármilyen, de azonos típusúak lehetnek. A kimenet Bool típusú. Tehát ha EN=1 és IN1<IN2<...<INn (n egész szám max. 8), akkor OUT=1. Minden más esetben OUT=0

Tizenötödik program – a LT függvény bemutatása

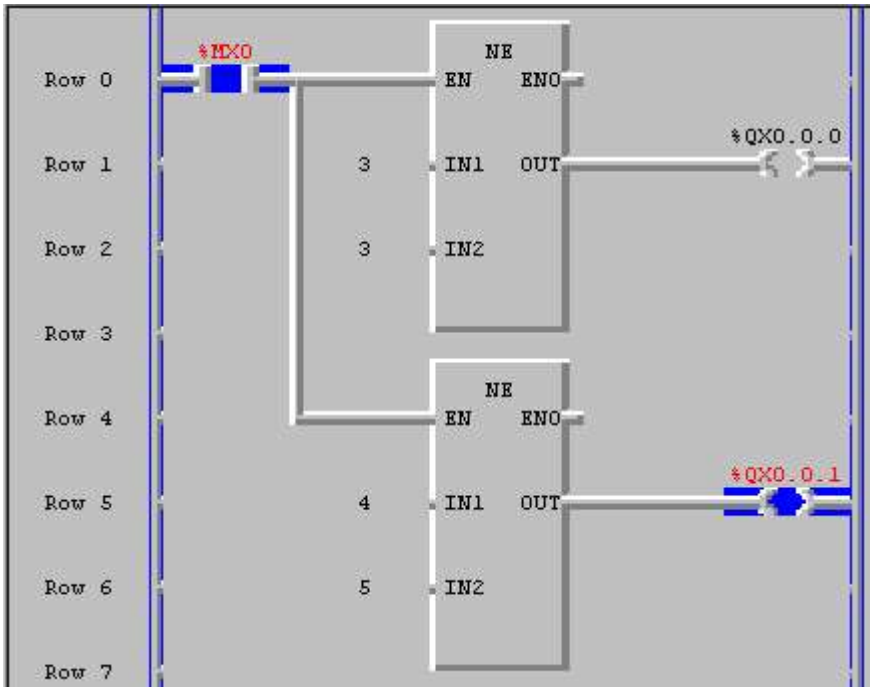


NE függvény



Az NE függvény valójában nem más mint a “nem egyenlő” reláció függvényváltozata. A bemenetekre(IN1..IN8) érekező változók bármilyen, de azonos típusúak lehetnek. A kimenet Bool típusú. Tehát ha EN=1 és IN1≠IN2≠...≠INn (n egész szám max. 8), akkor OUT=1. Minden más esetben OUT=0. EN0 értéke csak abban az esetben 0, ha nincs hiba.

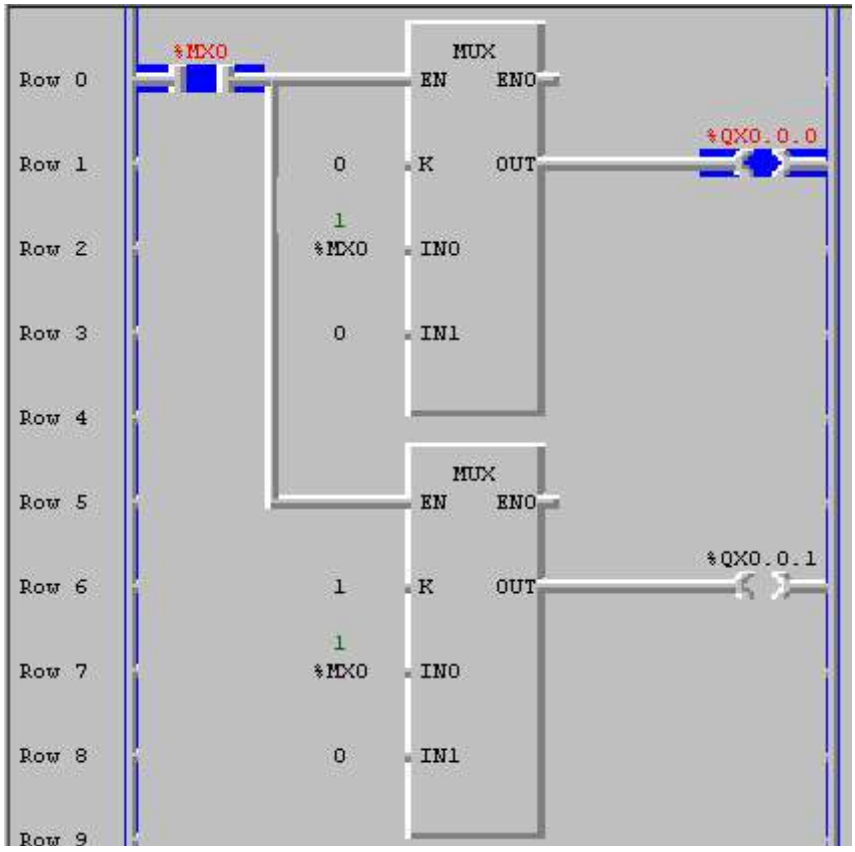
Tizenhatodik program – a NE függvény bemutatása



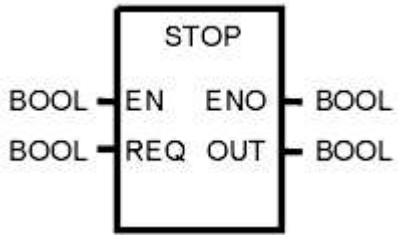
MUX (multiplexer) függvény

Az MUX függvény segítségével lehetőségünk van az (IN0... IN7) bemenetek közül a K(egész szám) bemenetre adott változóban megjelölt bemenetet kiválasztani – értékét átmásolni az OUT kimenetre. Az IN bemenetek, illetve OUT kimenet típusa bármi lehet. EN0 értéke csak abban az esetben 0, ha nincs hiba.

Tizenhetedik program – a MUX függvény bemutatása

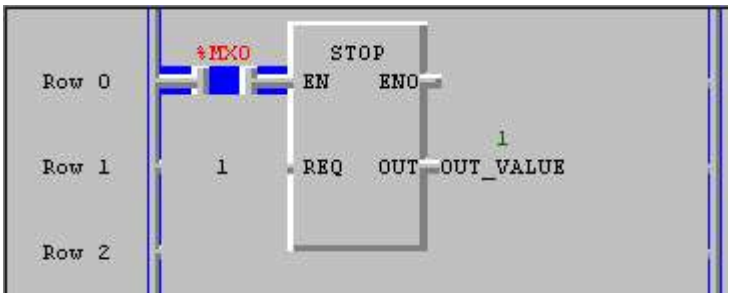


STOP függvény

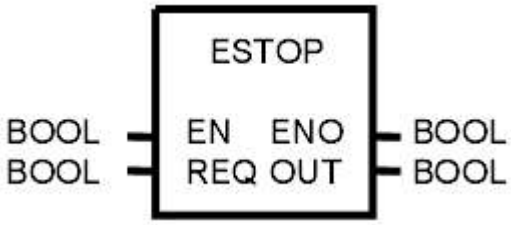


A STOP függvény akkor működik, ha az EN bemenetére logikai 1 érkezik. Ha a REQ bemenetre is logikai 1 érkezik, akkor a PLC STOP üzemmódba vált az utolsó utasítás végrahajtása után.

Tizennyolcadik program – a STOP függvény bemutatása

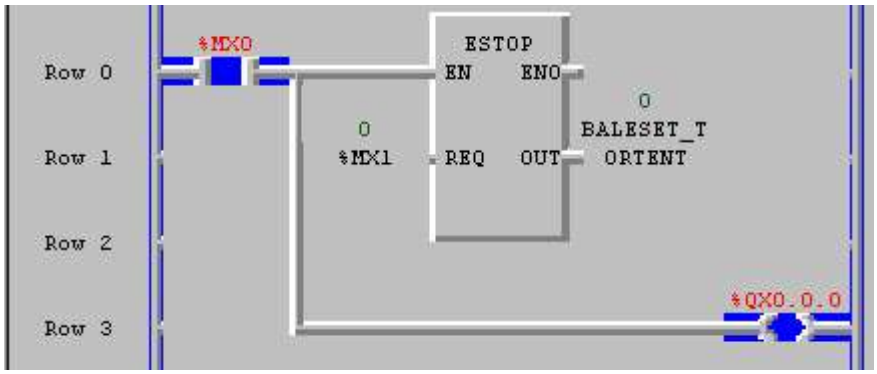


ESTOP függvény

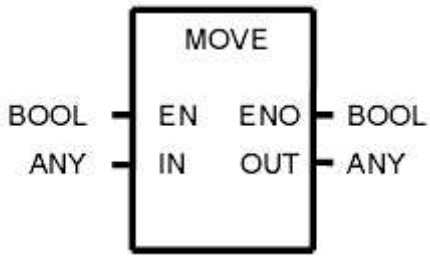


Az ESTOP függvény segítségével vészmegállás funkciót építhetünk be programunkba. Ha EN és REQ bemenetekreegyidejűleg logikai 1 érkezik, a program azonnyomban megáll, a PLC STOP üzemmódba kapcsol, illetve OUT értéke szintén logikai 1 lesz.

Tizenkilencedik program – a ESTOP függvény bemutatása

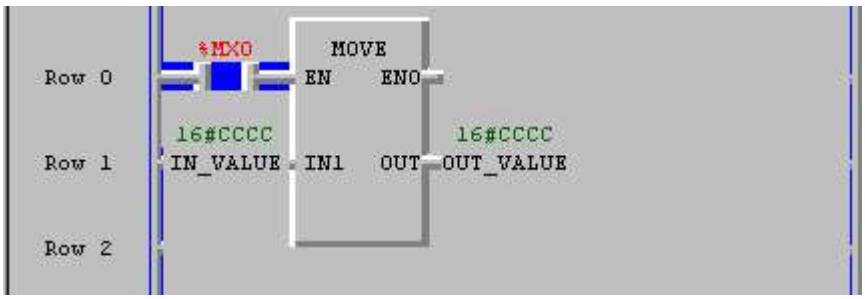


MOVE függvény



A Move függvény egyszerűen átadja az IN bemenetre érkező tetszőleges változó értékét az OUT (ugyanolyan típusú) kimenetre, ha EN-re logikai egy érkezik.

Huszadik program – a MOVE függvény bemutatása



NOT függvény

A NOT függvénny a negálás függvény. Segítségével biteket negálhatunk. Az OUT kimeneten az IN bemenetre adott bitek negáltja érkezik abban az esetben, ha EN-re logikai 1 érkezik. Pl: EN=1 IN=1100...1010 OUT=0011...0101

Huszonegyedik program – a NOT függvény bemutatása

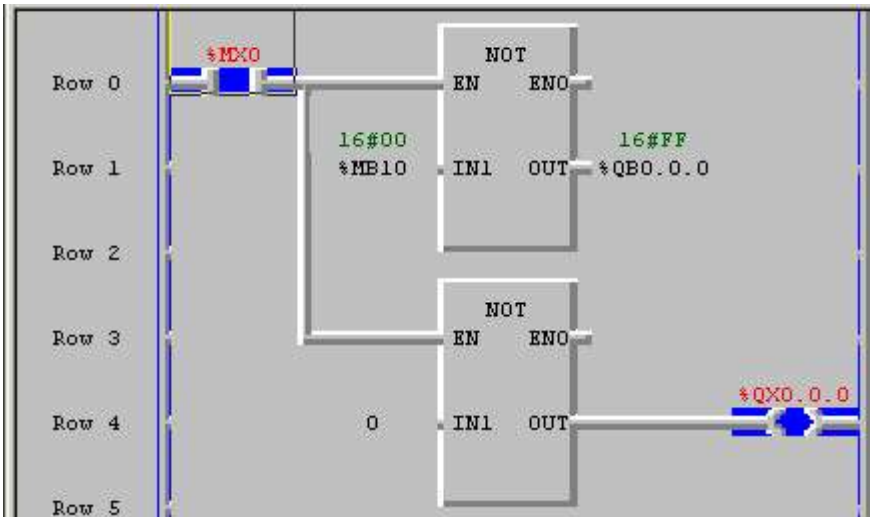
Input(IN1) : %MB10(BYTE) = 16#CC

Output(OUT) : %QB0.0.0(BYTE) = 16#33

1	1	0	0	1	1	0	0

(NOT)

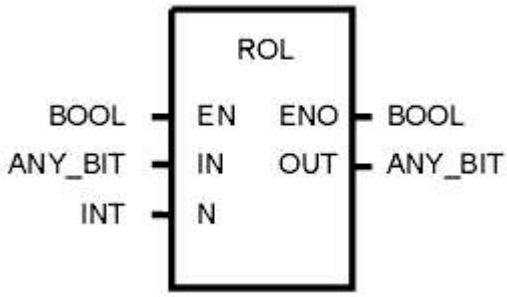
0	0	1	1	0	0	1	1



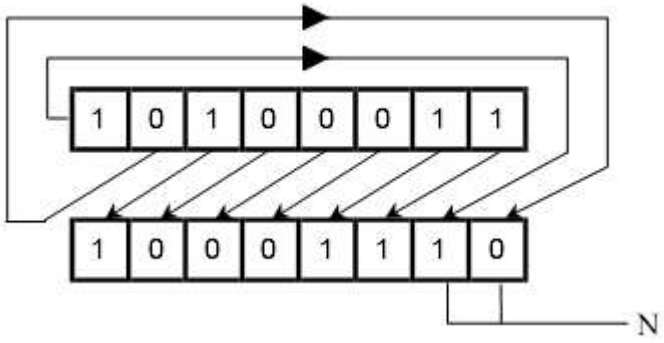
Megjegyzés:

Természetesen a BOOL változók is használhatók

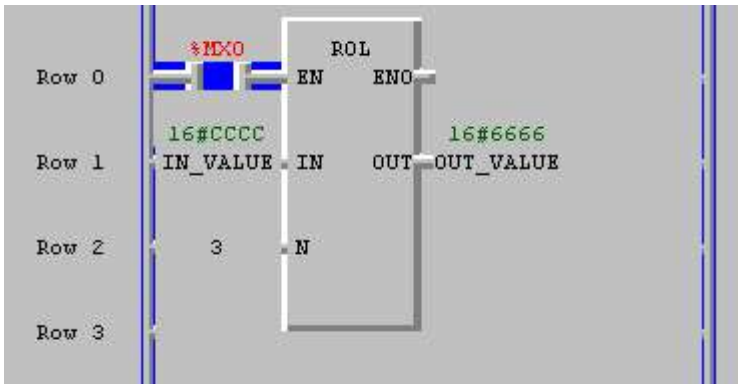
ROL függvény



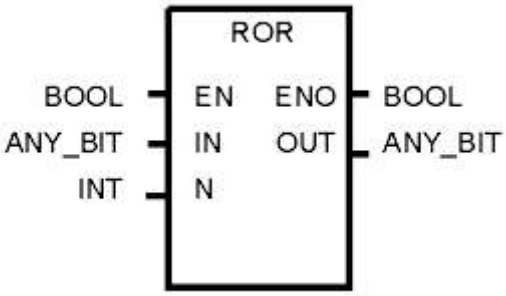
A ROL függvény annyiszor forgatja balra az IN -re érkező bitsorozatot, amennyi N értéke. Az eredmény az OUT kimenetre kerül. Csak akkor működik, ha az EN bemenetre logikai 1 érkezik.



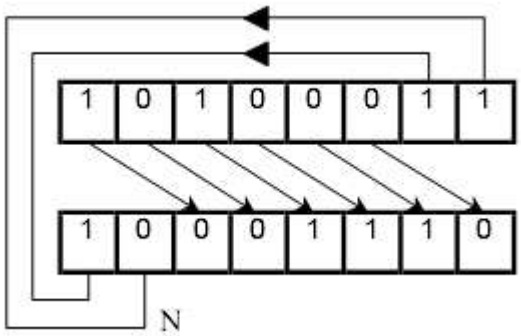
Huszonkettedik program – a ROL függvény bemutatása



ROR (jobbra forgatás) függvény

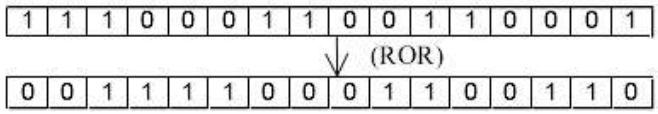


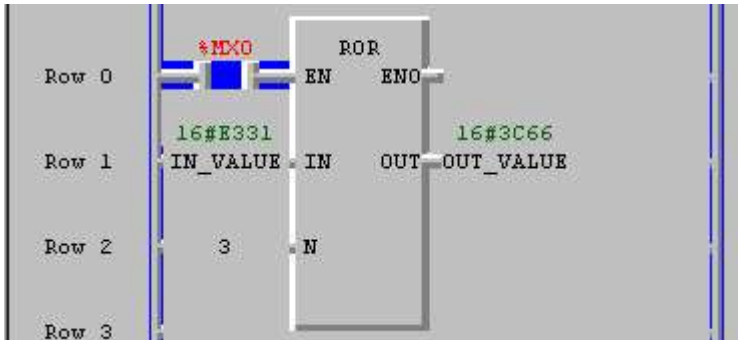
A ROR függvény annyiszor forgatja jobbra az IN -re érkező bitsorozatot, amennyi N értéke. Az eredmény az OUT kimenetre kerül. Csak akkor működik, ha az EN bemenetre logikai 1 érkezik.



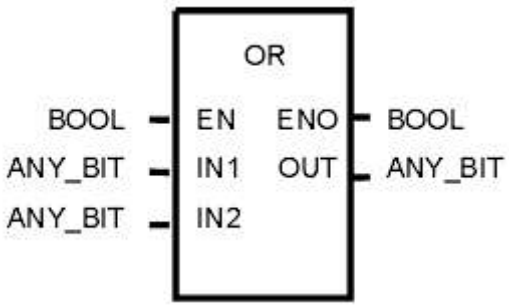
Huszonharmadik program – a ROR függvény bemutatása

Input (IN1) : IN_VALUE1(WORD) = 16#E331
(N) : 3
Output(OUT) :OUT_VALUE(WORD)=16#3C66



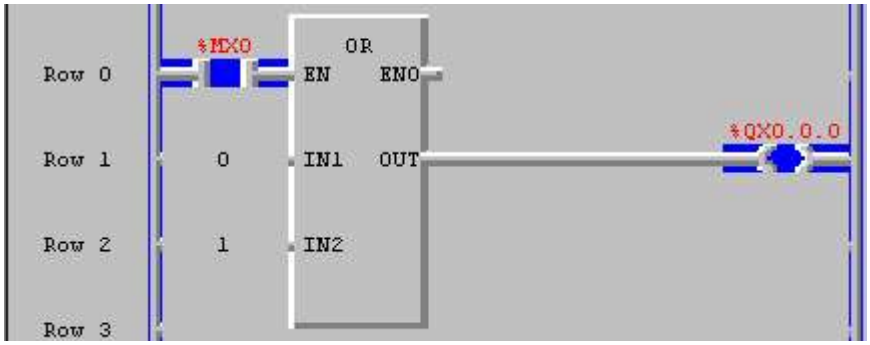


OR függvény

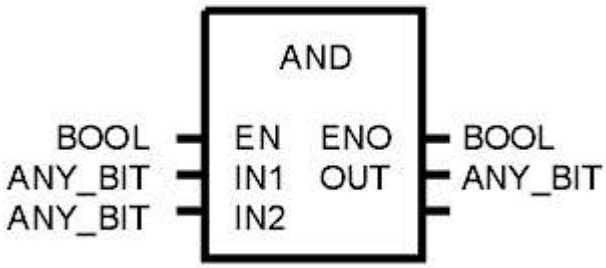


Az OR függvény segítségével logikai és kapcsolatot létesíthenünk az IN (minden Bit) bemenetek között. Az eredmény OUT- ra érkező bit lesz. A bemenetek száma nyolcig bővíthető. Csak akkor működik, ha az EN bemenetre logikai 1 érkezik!

Huszonnegyedik program – a OR függvény bemutatása

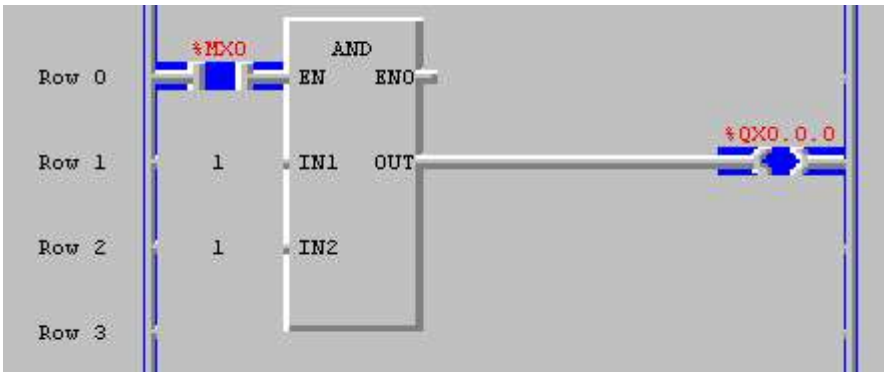


AND függvény

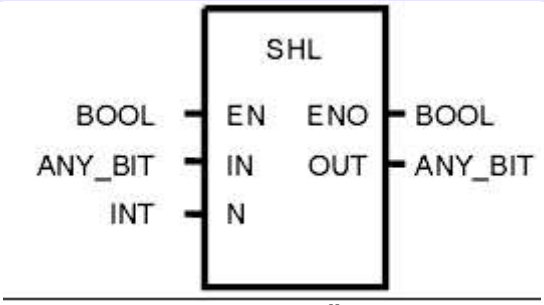


Az AND függvény segítségével logikai és kapcsolatot létesíthenünk az IN (minden Bit) bemenetek között. Az eredmény OUT- ra érkező bit lesz. A bemenetek száma nyolcig bővíthető. Csak akkor működik, ha az EN bemenetre logikai 1 érkezik!

Huszonötödik program – a AND függvény bemutatása



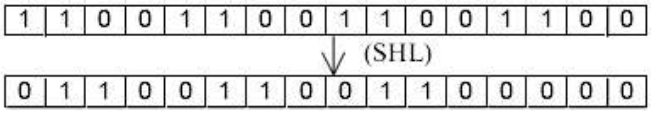
SHL függvény

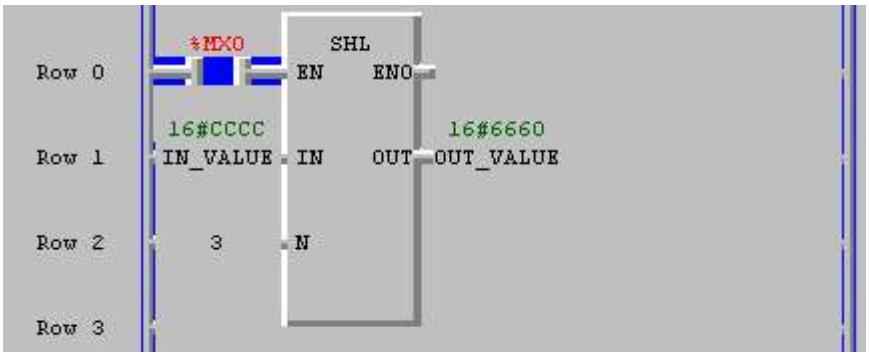


Az SHL függvény segítségével az IN bemenetre érkező biteket az N-re érkező egész szám értékével tolhatjuk balra. Az OUT kimenetre érkezik az ilymódon átalakitott bitsorozat.

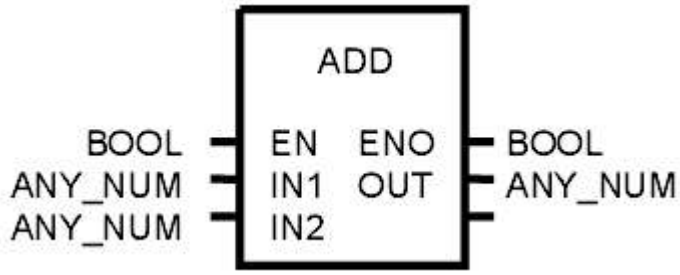
Huszonhatodik program – a SHL függvény bemutatása

Input(IN1) : IN_VALUE(WORD) = 16#CCCC
(N) : 3
Output(OUT) :OUT_VALUE(WORD) = 16#6660





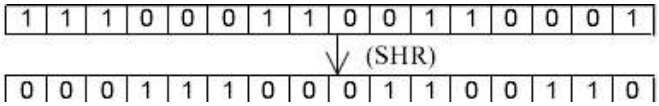
SHR függvény

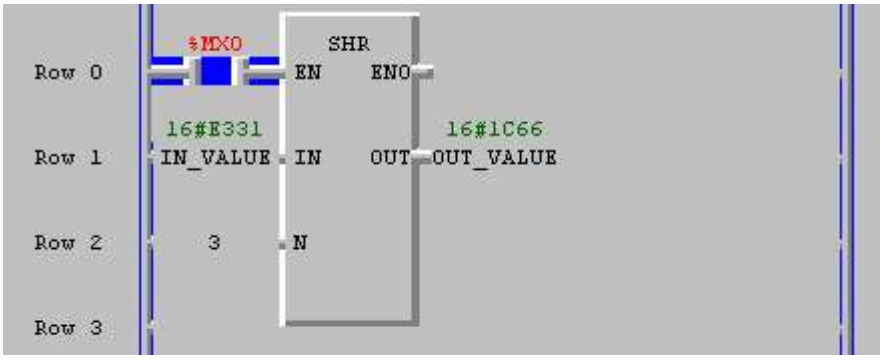


Az ADD függvény segítségével numerikus értékeket(IN1, IN2,..., I8) addhatunk össze(OUT), amennyiben az EN bemenetre logikai egy érkezik. Figyelni kell arra, hogy IN1, IN2 és OUT azonos típusú legyen. A bemenetek száma nyolcig bővíthető.

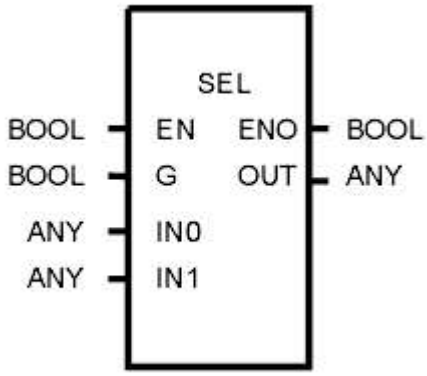
Huszonhetedik program – a SHR függvény bemutatása

Input (IN1): IN_VALUE(WORD) = 16#E331
(N) : 3
Output(OUT) : OUT VALUE(WORD) = 16#1C66



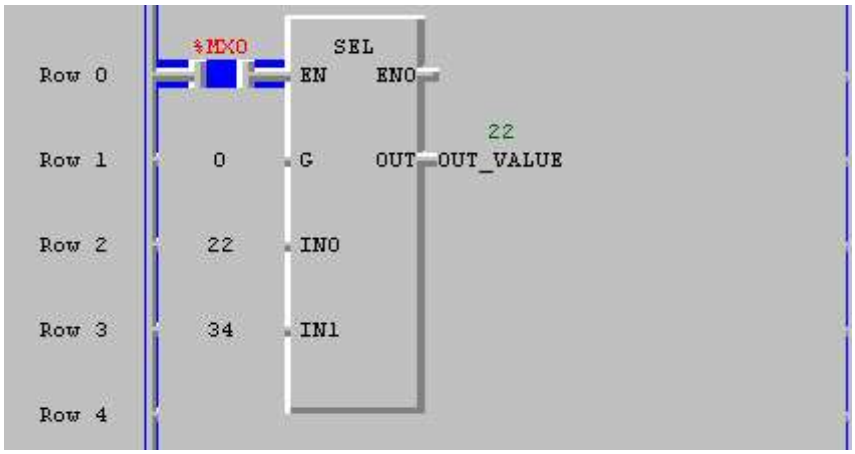


SEL függvény



A SEL függvénnyel kiválaszthatjuk(ráküldhetjük értékét OUT kimenetre) IN0, illetve IN1 bemenetet. Ha EN értéke logikai 1, illetve G értéke logikai 0, akkor IN0 kerül kiválasztásra. Abban az esetben, ha EN értéke logikai 1, illetve G értéke logikai 1, akkor IN1. Ha EN 0, akkor nem 0 érkezik a kimenetre.

Huszonnyolcadik program – a SEL függvény bemutatása



4.4 Funkcióblokkok