Tartalomjegyzék

1. Bevezető

Az IEC61131-3 széles körben elterjedt ipari vezérlés programozási szabvány. Ez a szabvány a tervezést és a működtetést harmonizálja össze a programozási felületen keresztül. Az IEC61131-3 elfogadott iránymutatás a legtöbb PLC gyártó számára.
A KINCO-K3 PLC programozó szoftvere, a KincoBuilder, felhasználó barát, hatékonyan kezelhető, kezelése könnyen elsajátítható.

KincoBuilder program a következő tulajdonságokkal rendelkezik:

- IEC61131-3 szabvány szerint készült
- Támogatott hagyományos programnyelvek: IL (utasítás lista), LD (létra diagram)
- Strukturált programozást tesz lehetővé
- Megszakítási rutinok kezelése
- Alprogramok a könnyebb áttekinthetőség érdekében
- Közvetlen címek és szimbolikus nevek is használhatók
- Felhasználóbarát és hatékony fejlesztői környezet
- Változtatható hardver konfiguráció, meghatározható hardver paraméterek

1.2. Általános megnevezések

- Főmodul (CPU body)
  A felhasználói programot tárolja a CPU modul belső memóriája, és a letöltést követően végre ciklikusan végre is hajtja azt. Mindeközben a CPU végrehajt egy önellenőrzést is: ellenőrzi a CPU megfelelő működéséhez a memória területeket és a bővítő modulokat.

- Bővítő modulok, kommunikáció
  A kiegészítő modul a központi egység bővítésére használható. A kiegészítő busz csatlakoztatja össze a főmodult és a kiegészítő modulokkal, a kapcsolatot 16 eres kábel biztosítja. Az adat-buszt, cím-buszt és a kiegészítő modulhoz szükséges tápellátást is biztosítja a kommunikációs busz.

- KincoBuilder
  Programozó szoftver a KINCO-K3 PLC-hez, az IEC61131-3 szabványnak megfelelően, jelenleg LD és IL nyelven érhető el, melyek kényelemes és hatékony programfejlesztést tesznek lehetővé.

- CPU firmware
  Ez az „operációs rendszere” a központi egységnek, mely a Flash memóriában található. A PLC-ben található programok végrehajtását is végzi.

- Felhasználói program
  Más néven felhasználói projekt vagy alkalmazási program, mely a felhasználó által írt speciális vezérlési funkciót tartalmazza. A felhasználói program a CPU-ba való letöltése után, az FRAM-ban tárolódik. Bekapcsolás után a főmodul kiolvassa az FRAM-ból a RAM-ba.

- Főprogram és végrehajtása
  A központi egység feladatok sorát hajtja végre ciklikusan. A főprogram a felhasználói program része. A központi egységben a főprogram ciklusonként egyszer fut le. A központi egységben egy főprogram futtatható.

- Soros vonal
  A főmodulban elérhető soros kommunikációs port, mely kommunikálhat Modbus RTU protokoll szerint (Slave-ként egyes típusoknál Master-ként is) vagy szabadon programozható protokollként. Szabad protokoll esetén ASCII és bináris protokollok is támogatottak, alkalmazásával egyedi kommunikációs protokoll is kialakítható.

- I/O memória terület
  Tartalmazza a bemeneti és a kimeneti memória területet. A programciklus elején a bemeneti jelek bekerülnek a bemeneti memória területére; a ciklus végén az értékeket a kimeneti memória területre teszi majd onnan közvetlenül a kimenetre kerülnek.

- Nem felejtő memória
  KincoBuilder hardver konfigurációjában, meghatározható négy nem felejtő memóriaterület, melyek értékei megmaradnak a tápfeszültség elvétele után is. A CPU tápellátásának megszűnésekor, a pillanatnyi adatok a RAM-ban megmaradnak a szuper kondenzátornak köszönhetően, a nem felejtő terület megtartja az adatot a következő bekapcsolásig. Normál hőmérsékleten 72 óráig képes megtartani az adatokat.

- Adat kimentés
  Adat kimentés az a folyamat, amikor néhány adatot kiírunk az EEPROM-ba vagy az FRAM-ba állandó tárolás céljából. Megj.: minden típusú hosszú távú memóriának van egy bizonyos élettartama, pl.: az EEPROM-ot 100 000-szer lehet újraírni, a FRAM-ot 10 billiószor.

1.3. Telepítés

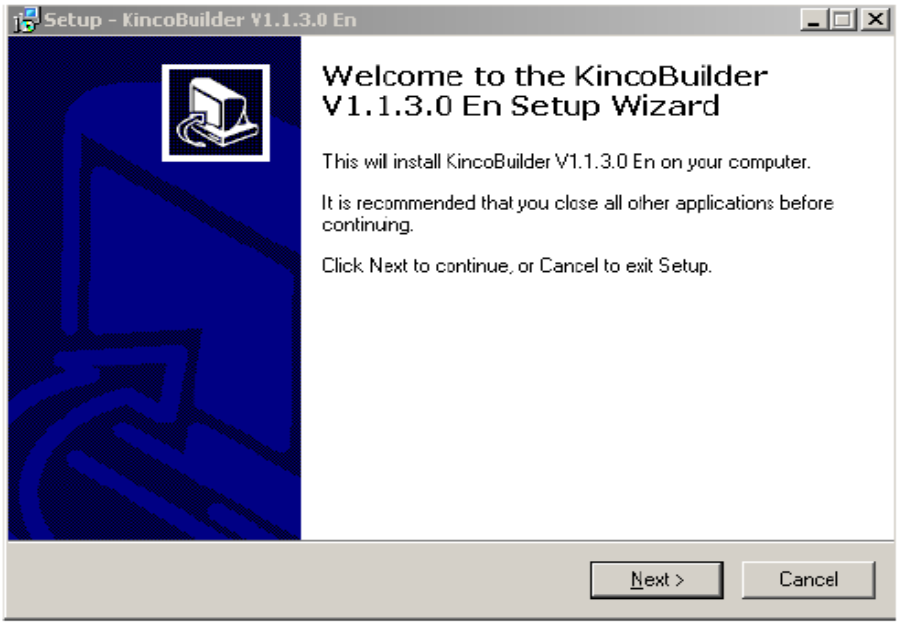
1.3.1. Rendszerkövetelmények

- CPU: 133 MHz vagy nagyobb
- Merevlemez: legalább 10Mbyte szabad terület
- RAM: 32M vagy nagyobb
- Billentyűzet, egér, soros port
- 256 vagy több színű VGA, 1024*768
- Operációs rendszer: Angol verziós Windows NT4.0 (vagy frissebb verzió)/ Windows 2000/Windows XP

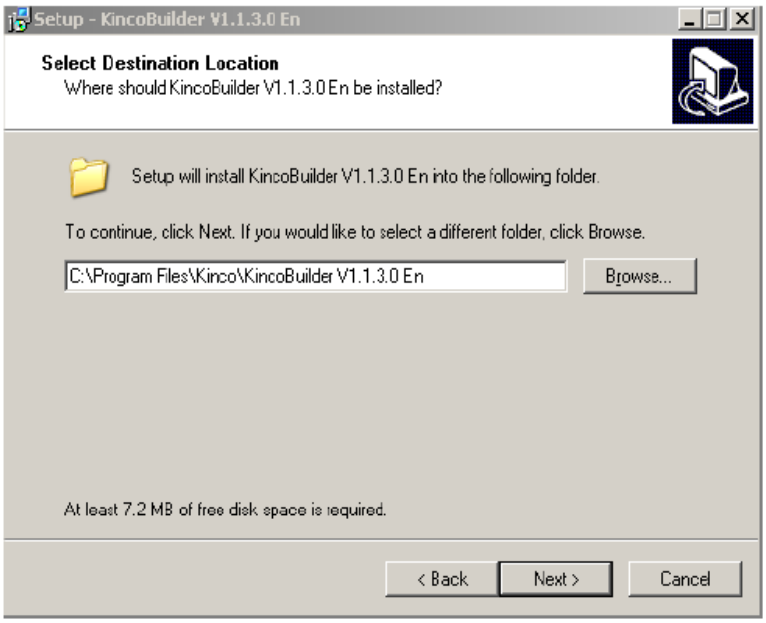
1.3.2. Telepítés/Törlés

Amennyiben egy korábbi verziója a KincoBuilder-nek már telepítve van, törölje mielőtt belekezd a telepítésbe.
A „Cancel” gombra kattintva bármikor megszakíthatja a telepítést.

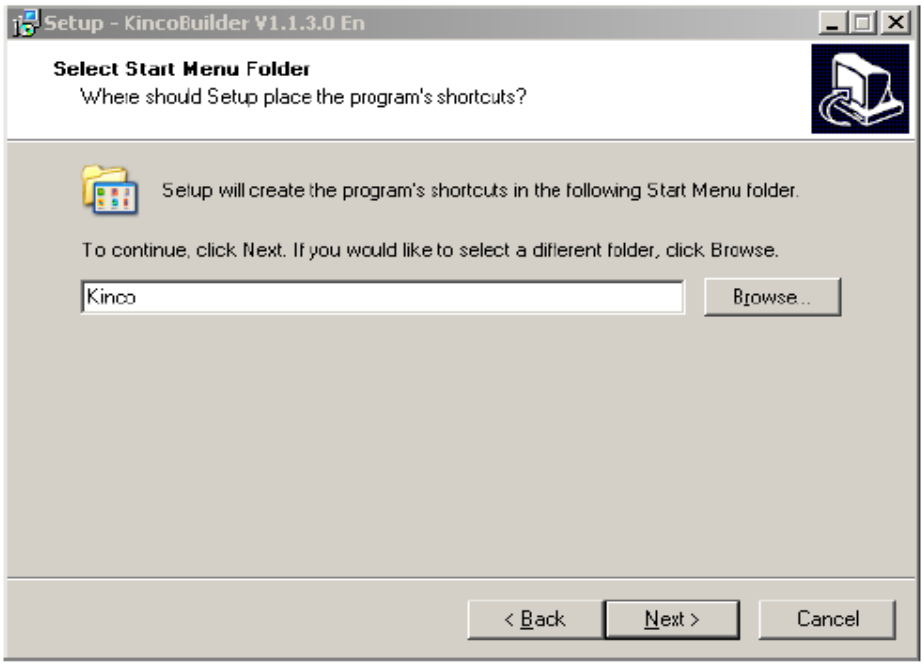
- Futtassuk a KincoBuilderVxxxx_setup.exe (xxxx a verziószámot jelöli) a telepítés megkezdéséhez.



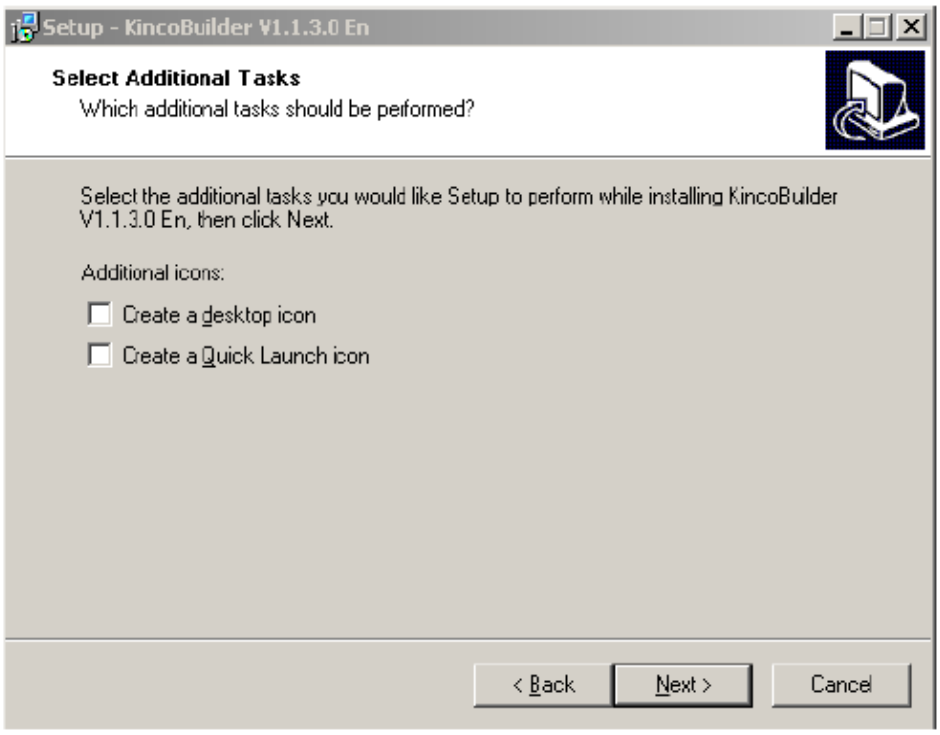
- Kattintsunk a Next gombra, hogy kiválaszthassuk a mappát ahova telepíteni szeretnénk. Választhatjuk a program által felajánlott mappát, de módosíthatjuk is amennyiben szükséges.



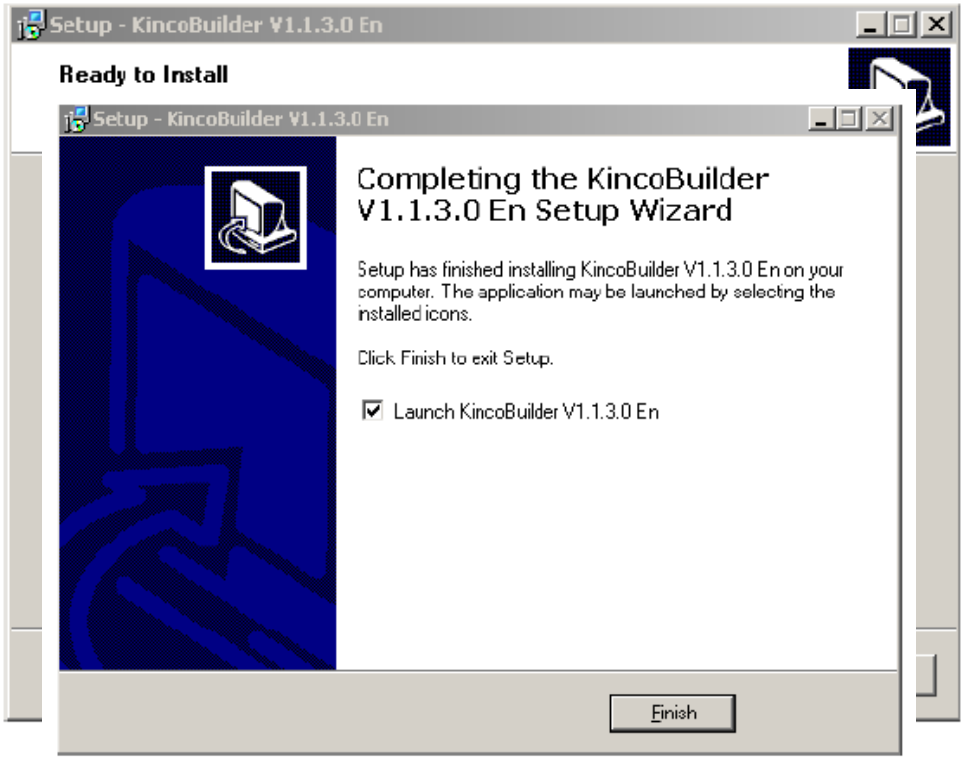
- Kattintsunk a Next gombra, hogy kiválasszuk a Start Menü-ben milyen nevű mappába legyen elérhető a program, az alapértelmezett a „Kinco” mappa.



- A Next gombra kattintva kiválaszthatjuk, hogy legyen e az asztalon parancsikon, illetve gyors elérési ikon.



- A Next gombra kattintva a program készen áll a tényleges telepítésre.



- Az Install gomb megnyomásával a programunk feltelepítődik számítógépünkre. A telepítés befejeztével a következő kép jelenik meg.

- A Finish gombra kattintva befejezhetjük a telepítést.

A program törléséhez zárjuk be a programunkat.

A program törléséhez két lehetőség közül választhatunk.

- Kattintsunk a [Start] gombra, válasszuk a [Minden program], majd a [Kinco] mappába válasszuk az „Uninstall KincoBuilder Vxxxx „ menüt.
  A KincoBuilder így automatikusan törölni fogja magát.

- Kattintsunk a [Start] gombra, válasszuk a [Vezérlopult] menüt.
  Nyissuk meg a „Programok telepítése és törlése” menüt.
  Válasszuk ki a KincoBuilder Vxxxx programot és kattintsunk az [Eltávolítás] gombra.

2. Programozási fogalmak

Ebben a fejezetben megismerkedhetünk a KINCO-K3 típusú PLC programozási alapfogalmaival a KincoBuilder segítségével és az IEC61131-3 szabvány néhány alapfogalmával, amelyek segítségére lehetnek bármilyen IEC61131-3 szabványú szoftver használatában.

2.1.POU (program szervezési egység)

A blokkokat amiből a projektek és programok épülnek POU-nak hívják az IEC61131-3 szabvány szerint. A név arra utal, hogy a POU a legkisebb, független szoftver egység mely a programkódot tartalmazza.

A következő három POU-t határozza meg az IEC61131-3:

- Program
  kulcsszó: PROGRAMME
  Ez a típusú POU tartalmazza a főprogramot, és végrehajtja azt. A programnak ki és bemeneti paraméterei egyaránt lehetnek.

- Funkció
  kulcsszó: FUNCTION
  A Funkcióknak bemeneti paraméterei és visszatérési értékei egyaránt lehetnek . A Funkció mindig ugyanazt az értéket adja, ha ugyanazzal a bemeneti paraméterrel hívjuk meg.

- Funkció blokk
  kulcsszó: FUNCTION_BLOCK
  A funkció blokkot rövidítve: FB, mostantól így használjuk a leírásban.
  A FB rendelkezik kimenetekkel/bemenetekkel és van statikus változója, és a statikus változó megjegyzi az előző állapotát. Az FB kimenete függ a statikus változó állapotától is, ha ugyanolyan paraméterekkel hívjuk meg.

A felhasználói projekt tartalmaz POU-kat, amelyeket a gyártó biztosít vagy a felhasználó állít elő. POU-k meghívhatják egymást paraméterekkel vagy azok nélkül, ez elősegíti a szoftver részeinek újrahasználását. A rekurzív hívások tilosak, az IEC61131-3 szabvány egyértelműen előírja, hogy POU-k nem tudják meghívni egymást direkt vagy indirekt módon.

2.2. Adattípusok

Az adattípusok meghatározzák egy adat bitszámát, az érték tartományát és az alapértelmezett kezdeti értékét. Minden változó adattípusát meg kell határozni a felhasználói programban.
Az alap adattípusok az IEC61131-3 szabványban meghatározottak, ennek következtében ezen adattípusok használata a PLC programozásban egységes és nyitott.
A KINCO-K3 által támogatott adattípusok a következő táblázatban látható.

Kulcsszó	Típus	Mérete (bit)	Érték tartomány	Alapértelmezett kezdeti érték
BOOL	Logikai érték	1	Igaz, hamis	HAMIS
BYTE	8 bit hosszú bit string	8	0~255	0
WORD	16 bit hosszú bit string	16	0~65,535	0
DWORD	32 bit hosszú bit string	32	0~4,284,967,295	0
INT	Előjeles egész szám	16	-2^{15}~$(2^{15}-1)$	0
DINT	Előjeles dupla egész szám	32	-2^{31}~$(2^{31}-1)$	0
REAL	Lebegőpontos szám ANSI/IEEE 754--1985	32	$1{,}18*10^{-38}$~$3{,}40*10^{38}$, $-3{,}40*10^{-38}$~$-1{,}18*10^{-38}$	0.0

2.3 Azonosítók

Az azonosítók betűk, számok és aláhúzás karakterek sorozata, de csak betűvel vagy aláhúzás karakterrel kezdődhet. (IEC61131-3)

2.3.1. Azonosítók meghatározása

A következő alapelveket kell betartani azonosítok meghatározásánál:

- Betűvel vagy aláhúzás karakterrel kell kezdődnie, aztán következhetnek számok vagy betűk vagy aláhúzás karakterek.
- Az azonosítók nem kis-nagy betű érzékenyek. Például az azonosítóknál az abc, ABC vagy az aBC egyenértékű.
- Az azonosítók hosszát a programozó környezet határozza meg. A KincoBuilder-ben a maximum hossz 16 karakter.
- Kulcsszavak nem használhatók, mint azonosítók, azok IEC61131-3 szabvány szerint foglaltak a programozói környezet számára.

2.3.2. Azonosítók használata

A KincoBuilder-ben a következő nyelvi elemekre lehet alkalmazni

- Programnév, funkció neve, FB neve
- Változó neve
- Címke, stb.

2.4. Állandók

Az állandó egy előre meghatározott érték a programban. Használhatjuk szám, karakterlánc vagy idő megadására, melyek értéke a program futása során nem változtatható. A konstans fontos jellemzője adattípusa és értéke. A KINCO-K3 által támogatott állandók tulajdonságai és a példák a következő táblázatban láthatóak.

Adattípus	Formátum[1]	Érték tartomány	Példa
BOOL	Igaz, hamis	Igaz, hamis	HAMIS
BYTE	B#szám	B#0~B#255	B#129
BYTE	B#2#bináris szám	B#0~B#255	B#2#10010110
BYTE	B#8#oktális szám	B#0~B#255	B#8#173
BYTE	B#16# hexa szám	B#0~B#255	B#16#3E
WORD	W#szám	W#0~W#65535	W#39675
WORD	2#bináris szám	W#0~W#65535	2#100110011
WORD	W#2#bináris szám	W#0~W#65535	W#2#110011
WORD	8#oktális szám	W#0~W#65535	8#7432
WORD	W#8#oktális szám	W#0~W#65535	8#174792
WORD	16#hexa szám	W#0~W#65535	16#6A7D
WORD	W#16#hexa szám	W#0~W#65535	W#16#9BFE
DWORD	DW#szám	DW#0~DW#4294967295	DW#547321
DWORD	DW#2#bináris szám	DW#0~DW#4294967295	DW#2#10111
DWORD	DW#8#oktális szám	DW#0~DW#4294967295	DW#8#76543
DWORD	DW#16#hexa szám	DW#0~DW#4294967295	DW#16#FF7D
INT	Szám	-32768~32767	12345
INT	I#szám	-32768~32767	I#-2345
INT	I#2#bináris szám[2]	-32768~32767	I#2#1111110
INT	I#8#oktális szám[2]	-32768~32767	I#8#16732
INT	I#16#hexa szám[2]	-32768~32767	I#16#7FFF
DINT	DI#szám	DI#-2147483647~DI#2147483647	DI#8976540
DINT	DI#2#bináris szám[2]	DI#-2147483647~DI#2147483647	DI#2#101111
DINT	DI#8#oktális szám[2]	DI#-2147483647~DI#2147483647	DI#8#126732
DINT	DI#16#hexa szám[2]	DI#-2147483647~DI#2147483647	DI#16#2A7FF
REAL	Tizedes pontos számok	$1.18*10^{-38}$~$3.40*10^{38}$, $-3.40*10^{38}$~$1.18*10^{-38}$	1.0,-243.456
REAL	xEy	$1.18*10^{-38}$~$3.40*10^{38}$, $-3.40*10^{38}$~$1.18*10^{-38}$	-2.3E-23

[1] Nem kis/nagy betű érzékeny, mindegy, hogy W#234 vagy w#234.

[2] Az INT és DINT bináris, oktális és hexadecimális ábrázolása mindkét esetben két komplemenses ábrázolás, vagyis MSB a jelzőbit: negatív a szám ha az MSB 1, és pozitív ha az MSB 0.
Pl.: I#16#FFFF = -1,I#7FFF = 32767,I#8000 = -32768, stb.

2.5. Változók

Ellentétben az állandókkal, a változók olyan adatot határoznak meg amely a program futása során változhat, pl.: olyan adat amely kapcsolatot teremt a bemenet, kimenet vagy PLC memóriája között. (IEC61131-3)
A változókat használják adatok inicializálására, eltárolására és feldolgozására. A változók adattípusát deklaráláskor meg kell határozni. A változók tárolási helyét meghatározhatjuk magunk vagy a programozó környezet automatikusan is lefoglalhatja.

2.5.1. Deklaráció

A változókat használat előtt deklarálni kell. Deklarálhatjuk a POU-n kívül és használhatjuk globálisan, mint interfész paraméter vagy helyi változó a POU-ban. A változók különféle változó típusokra vannak osztva deklarációs szempontból.
A következő táblázatban található a KINCO-K3 által támogatott változó típusok. A táblázatban a „Belső" jelzi, hogy a változó olvasható és írható a POU-n belül amelyben deklarálva van, a „Külső" jelzi, hogy a változó látható, olvasható és írható a POU hívása nélkül.

Változó típusa	Belső	Külső	Leírás
VAR	---	Olvasható/írható	Helyi változó Csak a POU-n belüli hozzáférés.
VAR_INPUT	Írható	Olvasható	Interfész bemeneti változó, pl: formális bemeneti paraméter Írható a POU meghívásakor, de csak olvasható azon belül.
VAR_OUTPUT	Olvasható	Olvasható/írható	Kimeneti változó, amely a POU visszatérési értéke. Csak olvasható POU meghívásakor, de írható és olvasható a saját POU-ján belül.
VAR_IN_OUT	Olvasható/írható	Olvasható/írható	Interfész be és kimeneti változó, pl: formális be és kimeneti paraméter VAR-INPUT és a VAR_OUTPUT tulajdonságaival rendelkezik
VAR_GLOBAL	Olvasható/írható	Olvasható/írható	Globális változó. Olvasható és írható az összes POU-n belül.

2.5.2. Változók deklarálása a KincoBuilderben

Minden változó típust táblázatos formában kell deklarálni, a KincoBuilder program szigorúan ellenőrzi a bevitt adatokat.
A globális váltózókat a Globális változó táblában kell deklarálni, míg a többi változót a megfelelő POU-ban található változó táblában. Minden POU-nak külön változó táblázata van. Ha ugyanazt a változó nevet használjuk a helyi és a globális szinten, akkor a helyi elsőbbséget élvez a saját POU-ján belül.

2.5.3 Változók automatikus ellenőrzése

Programozás közben a KincoBuilder ellenőrzi, hogy a változó használatakor megfelelő adattípushoz legyen rendelve. Például, ha egy REAL értéket WORD változóhoz rendel, vagy egy VAR_INPUT változó módosítva van a POU-n belül, akkor a KincoBuilder jelez, hogy módosítsa a programot.

2.6. PLC memória-hozzáférés módjai

A KINCO-K3 különböző memória területeken tárol információkat. A felhasználók kényelme érdekében a KINCO-K3 kétféle memóriacímzési módszert biztosít a területek elérése érdekében:

- Direkt címzés
- Indirekt címzés, mutató használatával

2.6.1. Memória típusok és jellemzőik

A KINCO-K3 PLC memóriaterületei különböző felhasználás céljából, különböző memória területeket biztosít, különböző karakterisztikákkal. A részleteket a következő táblázatban található.

I	
Leírás	DI (digitális bemenet) memória terület A KINCO-K3 az összes DI csatornát olvassa minden ciklus elején és beírja az értékeket az I területre.
Hozzáférési mód	Használható típusok: bit, byte, word, double word
Hozzáférési jog	Csak olvasható
Egyéb	Felülbírálható, állapotát tápfeszültség elvételekor nem jegyzi meg
Q	
Leírás	DO (digitális kimenet) memória terület. Minden ciklus végén a KINCO-K3 kiírja a Q területen található értékeket, a fizikai DO csatornákra.
Hozzáférési mód	Használható típusok: bit, byte, word, double word
Hozzáférési jog	Olvasható/írható
Egyéb	Felülbírálható, állapotát tápfeszültség elvételekor nem jegyzi meg
AI	
Leírás	AI (analóg bemenet) memória terület. A KINCO-K3 beolvassa az összes AI csatornát a ciklus elején, és az analóg értéket (áram, feszültség) 16-bites digitális értékké alakítja, majd az AI területen eltárolja.
Hozzáférési mód	Használható típusok: word (adat típusa INT)
Hozzáférési jog	Olvasható
Egyéb	Felülbírálható, állapotát tápfeszültség elvételekor nem jegyzi meg
AQ	
Leírás	AO (analóg kimenet) memória terület. A ciklus végén, a KINCO-K3 az AQ területen tárolt 16-bites digitális

	értéket átkonvertálja kimeneti jellé és kiírja azokat az AO csatornákra.
Hozzáférési mód	Használható típusok: word (adat típusa INT)
Hozzáférési jog	Olvasható/írható
Egyéb	Felülbírálható, állapotát tápfeszültség elvételekor nem jegyzi meg
HC	
Leírás	Gyors-számláló memória terület, a gyors-számlálók aktuális értékének a tárolására alkalmazható
Hozzáférési mód	Használható típusok: double word (adat típusa DINT)
Hozzáférési jog	Olvasható
Egyéb	Nem bírálható felül, állapotát tápfeszültség elvételekor nem jegyzi meg
V	
Leírás	Változó terület, relatív nagy terület, nagyméretű adatok tárolására.
Hozzáférési mód	Használható típusok: bit, byte, word, double word
Hozzáférési jog	Olvasható/írható
Egyéb	Felülbírálható, állapotát tápfeszültség elvételekor megjegyezheti
M	
Leírás	Belső memória terület. Összehasonlítva a V területtel, az M terület gyorsabban elérhető, elsősorban bit műveletek elvégzésére
Hozzáférési mód	Használható típusok: bit, byte, word, double word
Hozzáférési jog	Olvasható/írható
Egyéb	Felülbírálható, állapotát tápfeszültség elvételekor megjegyezheti
SM	
Leírás	Rendszer memória terület, rendszer adatok tárolására. Néhány cím olvasható a rendszer állapotának meghatározásához, némely pedig írható rendszerfunkciók módosításához
Hozzáférési mód	Használható típusok: bit, byte, word, double word
Hozzáférési jog	Olvasható/írható
Egyéb	Nem bírálható felül, állapotát tápfeszültség elvételekor nem jegyzi meg
L	
Leírás	Helyi változó terület. A KincoBuilder automatikusan kiosztja az L területet a helyi változók és be/ kimeneti változók számára. **Nem javasolt a közvetlen hozzáférés az L területhez.**
Hozzáférési mód	Használható típusok: bit, byte, word, double word
Hozzáférési jog	Olvasható/írható
Egyéb	Nem bírálható felül, állapotát tápfeszültség elvételekor nem jegyzi meg

2.6.2. Direkt memória címzés

- Közvetlenül megcímzett változó
  IEC61131-3 szabvány és egy megadott formátum szerint speciális karakter használatával. A közvetlen címzéshez a „%” karaktert kell használni, ezt követi a méret meghatározás, majd a tényleges cím, amennyiben szükséges elválasztó elemnek „.” használható. Például: %QB7 meghatározza a hetedik kimeneti byte-ot.

- Szimbolikus változó
  A szimbolikus változó esetén a közvetlen címzéssel ellentétben a fizikai címhez egy változó név rendelhető, mely segít az egyértelmű azonosításban. Célszerű, ha a változó neve utal a funkciójára is, így a programkód könnyebben értelmezhető lesz. A KincoBuilder programban a változó létrehozható a globális változó táblában, vagy a POU-hoz tartozó változó táblában.

2.6.2.1. Közvetlen változó címzés

A KincoBuilder programban használható változó címzéseket a következő táblázat tartalmazza. A táblázatban „x” vagy „y” decimális számokat jelent.

- I (digitális bemeneti memória) terület

Bites címzés	Formátum	**%I**x.y
	Leírás	x: változó bájt címe y: bitszám, 0~7.
	Adat típus	BOOL
	Példa	%I0.0 %I0.7 %I5.6
Byte-os címzés	Formátum	**%IB**x
	Leírás	x: változó bájt címe
	Adat típus	BYTE
	Példa	%IB0 %IB1 %IB5
Szavas (word) címzés	Formátum	**%IW**x
	Leírás	x: változó bájt kezdőcíme Mivel egy szó mérete 2 bájt, x-nek páros számnak kell lennie.
	Adat típus	WORD, INT
	Példa	%IW0 %IW2 %IW4
Dupla szavas (double world) címzés	Formátum	**%ID**x
	Leírás	x: változó bájt kezdőcíme Mivel a duplaszó mérete 4 bájt, x-nek páros számnak kell lennie.
	Adat típus	DWORD, INT
	Példa	%ID0 %ID4

- Q (digitális kimeneti memória)terület

Bites címzés	Formátum	**%Q**x,y
	Leírás	x: változó bájt címe y: bitszám, 0~7.
	Adat típus	BOOL
	Példa	%Q0.0 %Q0.7 %Q5.6
Byte-os címzés	Formátum	**%QB**x
	Leírás	x: változó bájt címe
	Adat típus	BYTE
	Példa	%QB0 %QB1 %QB5
Szavas (word) címzés	Formátum	**%QW**x
	Leírás	x: változó bájt kezdőcíme Mivel egy szó mérete 2 bájt, x-nek páros számnak kell lennie.
	Adat típus	WORD, INT
	Példa	%QW0 %QW2 %QW4
Dupla szavas (double world) címzés	Formátum	**%QD**x
	Leírás	x: változó bájt kezdőcíme Mivel a duplaszó mérete 4 bájt, x-nek páros számnak kell lennie.
	Adat típus	DWORD, INT
	Példa	%QD0 %QD4 /QD12

- AI (analóg bemeneti memória) terület

Szavas (word) címzés	Formátum	**%AIW**x
	Leírás	x: változó bájt kezdőcíme Mivel egy szó mérete 2 bájt, x-nek páros számnak kell lennie.
	Adat típus	INT
	Példa	%AIW0 %AIW2 %AIW12

- AQ (analóg kimeneti memória) terület

Szavas (word) címzés	Formátum	**%AQW**x
	Leírás	x: változó bájt kezdőcíme Mivel egy szó mérete 2 bájt, x-nek páros számnak kell lennie.
	Adat típus	INT
	Példa	%AQW0 %AQW2 %AQW12

- M (belső memória) terület

Bites címzés	Formátum	**%M**x,y
	Leírás	x: változó bájt címe y: bitszám, 0~7.
	Adat típus	BOOL
	Példa	%M0.0 %M0.7 %M5.6
Byte-os címzés	Formátum	**%MB**x
	Leírás	x: változó bájt címe
	Adat típus	BYTE
	Példa	%MB0 %MB1 %MB10
Szavas (word) címzés	Formátum	**%MW**x
	Leírás	x: változó bájt kezdőcíme Mivel egy szó mérete 2 bájt, x-nek páros számnak kell lennie.
	Adat típus	WORD, INT
	Példa	%MW0 %MW2 %MW4
Dupla szavas (double world) címzés	Formátum	**%MD**x
	Leírás	x: változó bájt kezdőcíme Mivel a duplaszó mérete 4 bájt, x-nek páros számnak kell lennie.
	Adat típus	DWORD, INT
	Példa	%MD0 %MD4 %MD12

- V (változó memória) terület

Bites címzés	Formátum	**%V**x,y
	Leírás	x: változó bájt címe y: bitszám, 0~7.
	Adat típus	BOOL
	Példa	%V0.0 %V0.7 %V5.6
Byte-os címzés	Formátum	**%VB**x
	Leírás	x: változó bájt címe
	Adat típus	BYTE
	Példa	%VB0 %VB1 %VB10
Szavas (word) címzés	Formátum	**%VW**x
	Leírás	x: változó bájt kezdőcíme Mivel egy szó mérete 2 bájt, x-nek páros számnak kell lennie.
	Adat típus	WORD, INT
	Példa	%VW0 %VW2 %VW4

Dupla szavas (double world) címzés	Formátum	**%ID**x
	Leírás	x: változó bájt kezdőcíme Mivel a duplaszó mérete 4 bájt, x-nek páros számnak kell lennie.
	Adat típus	DWORD, INT
	Példa	%VD0 %VD4 %VD12
Lebegőpontos (real) címzés	Formátum	**%VR**x
	Leírás	x: változó bájt kezdőcíme Mivel a lebegőpontos szám mérete 4 bájt, x-nek páros számnak kell lennie.
	Adat típus	REAL
	Példa	%VR0 %VR4 %VR1200

- SM (rendszermemória) terület

Bites címzés	Formátum	**%SM**x,y
	Leírás	x: változó bájt címe y: bitszám, 0~7.
	Adat típus	BOOL
	Példa	%SM0.0 %SM0.7 %SM5.6
Byte-os címzés	Formátum	**%SMB**x
	Leírás	x: változó bájt címe
	Adat típus	BYTE
	Példa	%SMB0 %SMB1 %SMB10
Szavas (word) címzés	Formátum	**%SMW**x
	Leírás	x: változó bájt kezdőcíme Mivel egy szó mérete 2 bájt, x-nek páros számnak kell lennie.
	Adat típus	WORD, INT
	Példa	%SMW0 %SMW2 %SMW4
Dupla szavas (double world) címzés	Formátum	**%SMD**x
	Leírás	x: változó bájt kezdőcíme Mivel a duplaszó mérete 4 bájt, x-nek páros számnak kell lennie.
	Adat típus	DWORD, INT
	Példa	%SMD0 %SMD4 %SMD12

- L terület (Megj.: Nem javasolt az L terület közvetlen címzése)

Bit címzés	Formátum	**%L**x,y
	Leírás	x: változó bájt címe y: bitszám, 0~7.
	Adat típus	BOOL
	Példa	%L0.0 %L0.7 %L5.6
Byte címzés	Formátum	**%LB**x
	Leírás	x: változó bájt címe
	Adat típus	BYTE
	Példa	%LB0 %LB1 %LB10
World címzés	Formátum	**%LW**x
	Leírás	x: változó bájt kezdőcíme Mivel egy szó mérete 2 bájt, x-nek páros számnak kell lennie.
	Adat típus	WORD, INT
	Példa	%LW0 %LW2 %LW4
Double world címzés	Formátum	**%LD**x
	Leírás	x: változó bájt kezdőcíme Mivel a duplaszó mérete 4 bájt, x-nek páros számnak kell lennie.
	Adat típus	DWORD, DINT, REAL
	Példa	%LD0 %LD4 %LD12

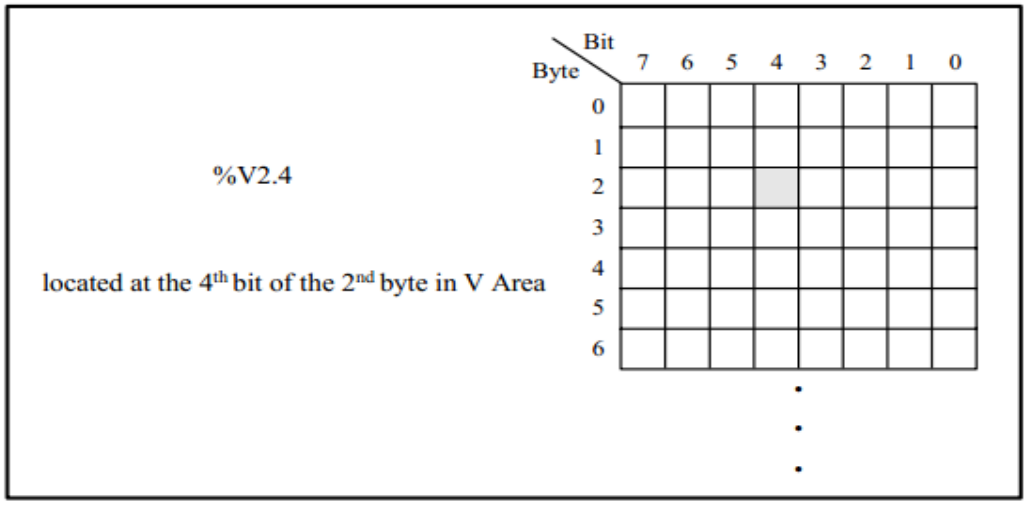
- HC (gyorsszámláló memória) terület

Double world címzés	Formátum	**%HC**x
	Leírás	x: nagy sebességű számláló száma
	Adat típus	DINT
	Példa	%HC0 %HC1

2.6.2.2. Címkiosztás a direkt címzés és a PLC memóriaterület között

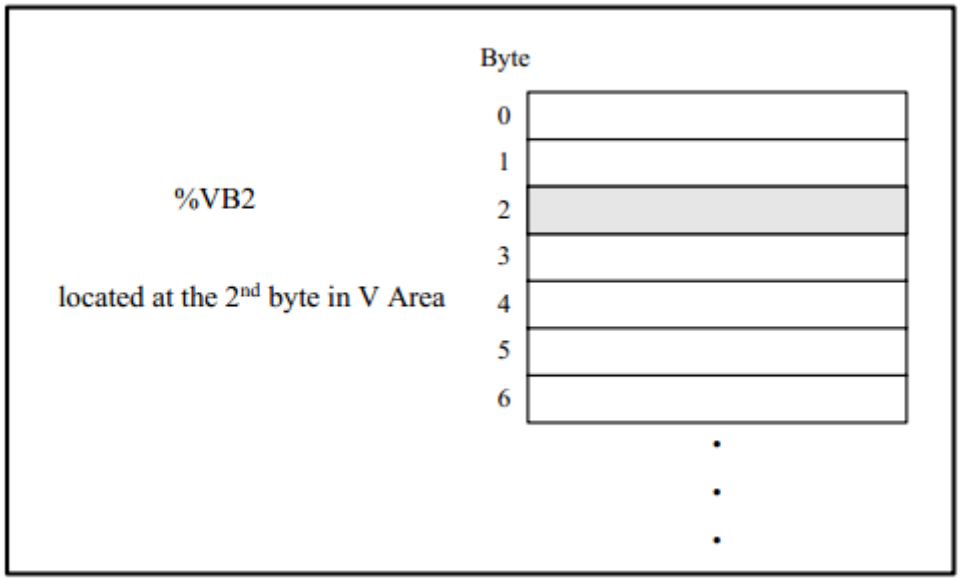
Minden direkt cím a PLC egy megadott memória területére hivatkozik. A következő példában látható, hogyan hivatkozik a direkt címzett változó a PLC memóriaterületére.

- Bites címzés



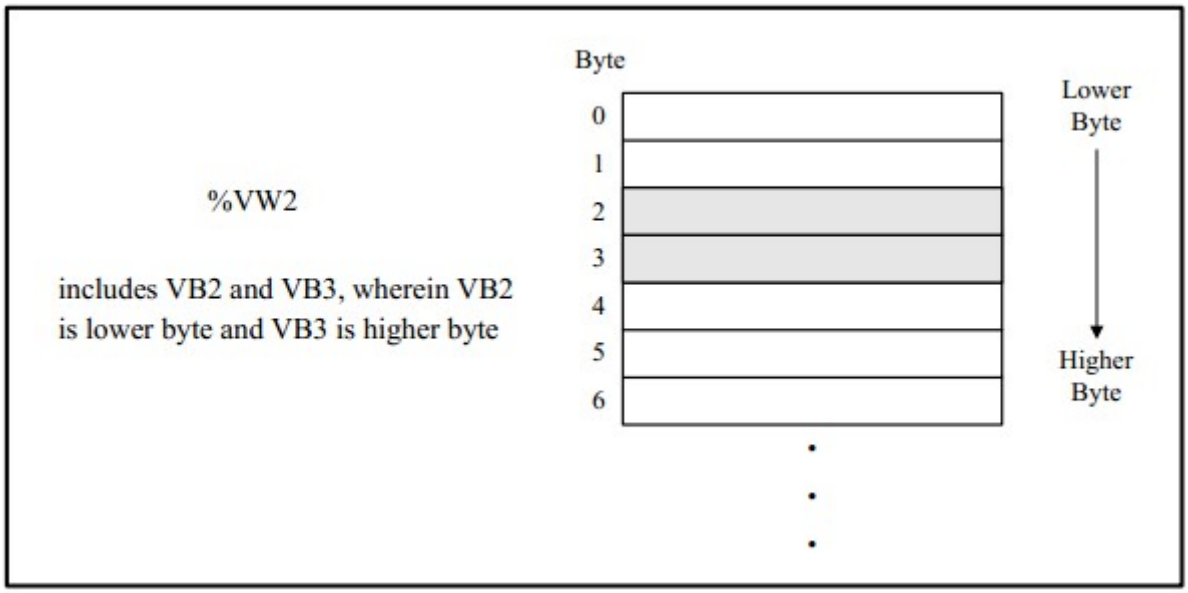
(%V2.4 a 2 bájt 4. bitjét jelenti a V területen található)

- Byte-os címzés



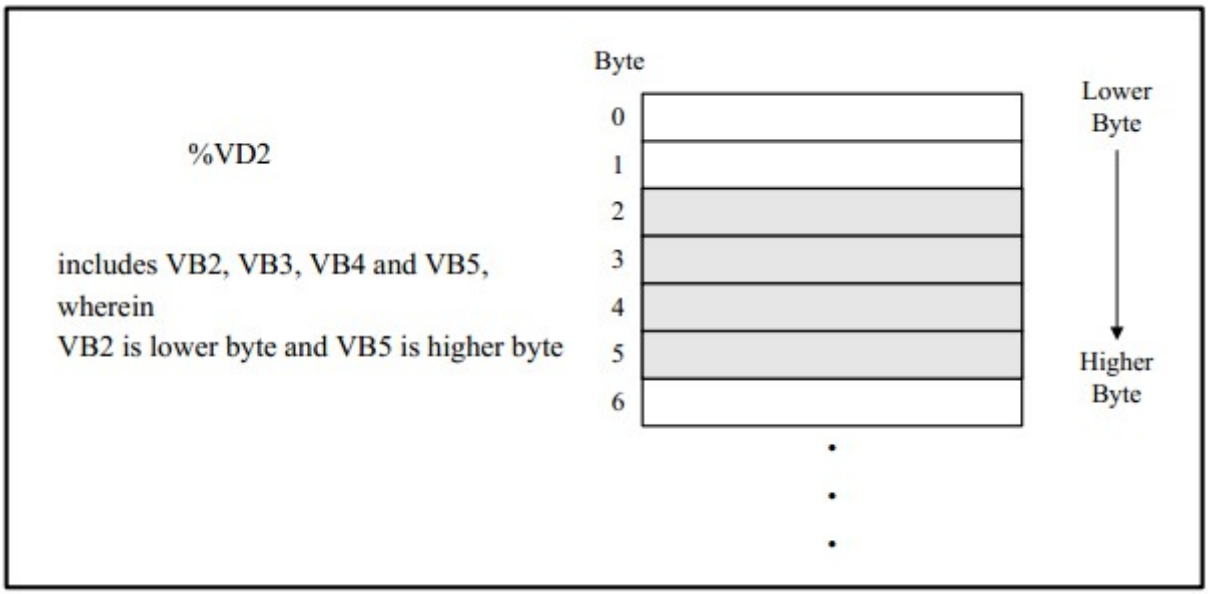
(%VB2 a 2.bájtja, a V területen található)

- Szavas címzés



(%VW2 magába foglalja a VB2-t és VB3-t, amelyben VB2 az legkisebb helyiértékű, VB3 a legnagyobb helyiértékű bájt)

- Dupla szavas címzés



(%VD2 magába foglalja VB2-t, VB3-t, VB4-t és VB5-t, amelyben VB2 a legkisebb helyiértékű, VB5 a legnagyobb helyiértékű bájt)

2.6.3. Indirekt címzés

A mutató egy double word típusú változó, ami tárolja a fizikai memória címet. Az indirekt címzéshez mutatók használata szükséges, mellyel megcímezhető a fizikai memória terület.
A KINCO-K3 PLC esetén csak a V terület címezhető indirekt módon.

Megjegyzés: Csak a CPU306Ex és a CPU308 támogatja az indirekt címzési metódust.

2.6.3.1. Mutató létrehozása

Az indirekt adateléréshez a memóriaterületen elsőként létre kell hozni egy mutatót. Ehhez a '&' cím operátort kell használnunk, pl.: &VB100 áll a VB100 fizikai címén.

Például:

(*létrehozunk egy mutatót (VD204) ami a VW2 re mutat, azaz a fizikai címe a VW2-nek a VD204-en tárolódik*)

```
MOVE    &VW2,%VD204
```

2.6.3.2. Adathozzáférés mutató használatával

'*' karaktert kell használni, mint mutató operátort. Megadjuk a '*'-ot az elé a mutató elé, amely hivatkozik arra a direkt címzett változóra, amelyre a mutató mutat. Amikor a mutatót operandusként használjuk az utasításban, figyeljünk az utasítás operandusának adattípusára.
Például:

```
LD      %SM0.0
MOVE    &VB0,%VD200     (*Létrehozunk egy mutatót (VD200) ami a VB0-ra mutat*)
MOVE    *VD200,%VB10    (*VB0 értéke VB10-be kerül. A VD200 mutató a VB0-ra*)
                        (*mutat*)
```

2.6.3.3. Mutató értékének módosítása

A mutató egy 32-bites változó, és ennek az értékét módosíthatjuk az ADD és SUB, stb. utasításokkal.
Amikor a mutató értékét egyel növeljük/csökkentjük, az a direkt cím is amelyre a mutató mutat növekedni/csökkenni fog 1 byte-tal. Tehát amikor változtatjuk a mutató értékét, figyelni kell a változó adattípusára, melyre a mutató hivatkozik.

- Ha a mutató byte-os típusú változóra mutat, a mutató értékét bármilyen kétszavas (double integer) számmal változtathatjuk.
- Ha a mutató szavas vagy integer típusú változóra mutat, a mutató értékét 2 többszörösére változtathatjuk.
- Ha a mutató dupla szavas (DWORD), DINT vagy lebegőpontos (REAL) típusú változóra mutat, a mutató értékét 4 többszörösére változtathatjuk.

2.6.3.4. Megjegyzés a mutatók használatához

- A mutató érvényességéről a felhasználói programnak kell gondoskodnia. A mutató rugalmasan alkalmazható, de használatával körültekintően kell bánni. Ha a mutató egy érvénytelen címre mutat, az nem várt eredményhez vezethet.
- A KINCO-K3 csak az egyszintű mutatót és címet támogatja, a többszintű mutató és cím érvénytelen. A következő példa érvénytelen:

  MOVE &VB4,*VD44

2.6.3.5. Példa

```
(*Network0*)

LD      %SM0.0
MOVE    &VW0,%VD200     (*Létrehozunk egy mutatót (VD200) ami a VW0.-ra mutat*)
MOVE    *VD200,%VW50    (*VW0 értéke bekerül VW50-be. VD200-as mutató a VW0-ra
                    0   mutat*) (*tehát a VD200 a VW0-t jelenti*)
ADD     DI#2,%VD200     (*A mutató értékét kettővel növeljük, így az VW2-re mutat.*)
MOVE    *VD200,%VW52    (*VW2 értékét VW52-be tesszük.*)
```

2.6.4. Elérhető memória tartományok

A KINCO-K3 több fajta CPU modullal rendelhető, melyek memória kiosztása eltérő egymástól. A programban bizonyosnak kell lenni, hogy az összes megadott memória cím az alkalmazott CPU-nak megfelelő.

		CPU304	CPU304EX, CPU306	CPU306EX, CPU308
I	Size	2 bytes	8 bytes	32 bytes
	Bit address	%I0.0 --- %I1.7	%I0.0 --- %I7.7	%I0.0 --- %I31.7
	Byte address	%IB0、IB1	%IB0 --- %IB7	%IB0 --- %IB31
	Word address	%IW0	%IW0 ---% IW6	%IW0 ---% IW30
	Double-word address	———	%ID0 --- %ID4	%ID0 --- %ID28
Q	Size	2 bytes	8 bytes	32 bytes
	Bit address	%Q0.0 --- %Q0.7	%Q0.0 --- %Q7.7	%Q31.0 --- %Q31.7
	Byte address	%QB0	%QB0 --- %QB7	%QB0 --- %QB31
	Word address	———	%QW0 --- %QW6	%QW0 --- %QW30
	Double-word address	———	%QD0 --- %QD4	%QD0 --- %QD28
AI	Size	0	32 bytes	64 bytes
	Word address	———	%AIW0 --- %AIW30	%AIW0 --- %AIW62
AQ	Size	0	32 bytes	64 bytes
	Word address	———	%AQW0 -- %AQW30	%AQW0 -- %AQW62
HC	Size	8 bytes	24 bytes	
	Word address	%HC0，%HC1	%HC0 --- %HC5	
V	Size	2048 bytes	4096 bytes	
	Bit address	%V0.0 --- %V2047.7	%V0.0 ---%V4095.7	
	Byte address	%VB0 --- %VB2047	%VB0 --- %VB4095	
	Word address	%VW0 --- %VW2046	%VW0 --- %VW4094	
	Double-word address	%VD0 --- %VD2044	%VD0 --- %VD4092	
	REAL address	%VR0 --- %VR2044	%VR0 --- %VR4092	
M	Size	32 bytes		
	Bit address	%M0.0 --- %M31.7		
	Byte address	%MB0 --- %MB31		
	Word address	%MW0 --- %MW30		

	Double-word address	%MD0 --- %MD28
SM	Size	300 bytes
	Bit address	%SM0.0 --- %SM299.7
	Byte address	%SMB0 --- %SMB299
	Word address	%SMW0 --- %SMW298
	Double-word address	%SMD0 --- %SMD296
L	Size	272 bytes
	Bit address	%L0.0 --- %L271.7
	Byte address	%LB0 --- %LB271
	Word address	%LW0 --- %LW270
	Double-word address	%LD0 --- %LD268

(Bit address=bites címzés, Byte address=Byte-os cím, Word address=Szavas cím, Double-word=Dupla szavas cím)

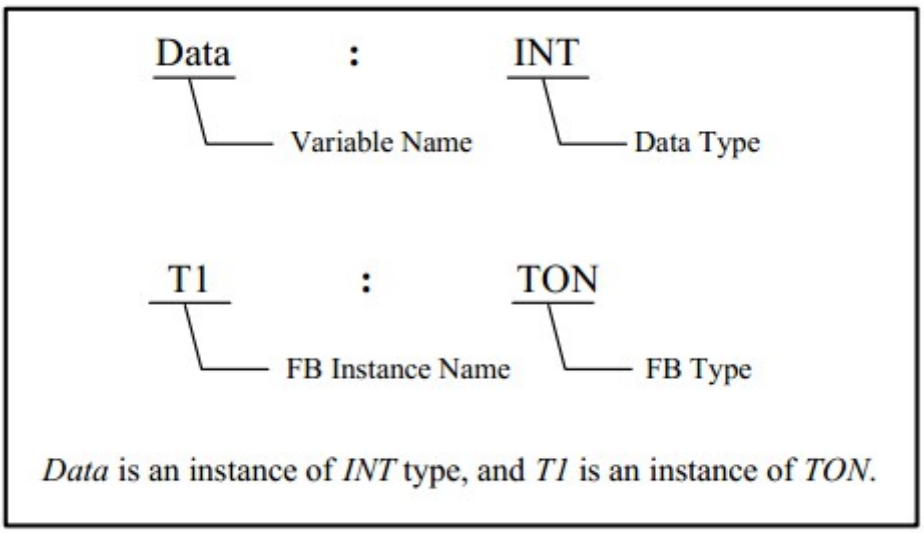
2.6.5 Funkcióblokkok és változóik

2.6.5.1. Alap funkcióblokkok az IEC61131-3 szabványban

- Időzítők: TP --- impulzus időzítő; TON --- bekapcsolás késleltető; TOF --- kikapcsolás késleltető
- Számlálók: CTU --- felfele számláló; CTD --- lefele számláló; CTUD ---- fel-le számláló
- Bistabil elemek: SR --- Set a meghatározó; RS --- Reset a meghatározó
- Élváltás figyelő: R_TRIG --- felfutó él figyelő; F_TRIG --- lefutó él figyelő

2.6.5.2. Funkcióblokk változók

A "FB deklarálás" fontos része az IEC61131-3 szabványnak.
A deklarálás során a programozó változókat hoz létre (deklarál) megadott névvel és adat típussal.
A deklarálást követően a változó a programból elérhető.
A FB változót is deklarálni kell mint egy változót. Deklarálás után (mint a változó) a FB-ot használhatjuk azon a POU-n belül ahol deklaráltuk.
A következő példa egy bekapcsolás késleltető blokk deklarációját mutatja be:



A „Data” változó INT típus, míg a T1 névvel ellátott funkcióblokk TON típusú.

2.6.5.3. FB változó memória területe

A KINCO-K3 PLC-nél a funkcióblokkok fix memória területet használhatnak, melyek leírását a következő táblázat tartalmazza.

T	
Leírás	Időzítő memória terület, a TON, TOF és TP változói számára. Itt tárolódnak a státusz bitek, és az aktuális értékeik az időzítőknek
Elérési mód	Státuszbit és az aktuális értéke az időzítőnek közvetlen elérhető.
Elérési jog	Olvasás
Egyéb	Nem bírálható felül, állapotát tápfeszültség elvételekor nem jegyzi meg
C	
Leírás	Számláló memória terület, a CTU, CTD és CTUD változói számára. Itt tárolódik a státusz bit, és az aktuális értéke minden számlálónak.
Elérési mód	Státuszbit és az aktuális értéke az időzítőnek közvetlen elérhető.
Elérési jog	Olvasható
Egyéb	Nem bírálható felül, állapotát tápfeszültség elvételekor megjegyzi
RS	
Leírás	RS memória terület, az RS változói számára. Itt tárolódnak a státuszbitek az RS értékeinek.
Elérési mód	Státuszbit közvetlen elérhető.
Elérési jog	Olvasható
Egyéb	Nem bírálható felül, állapotát tápfeszültség elvételekor nem jegyzi meg
SR	
Leírás	SR memória terület, az SR változói számára. Itt tárolódnak a státuszbitek az SR értékeinek.
Elérési mód	Státuszbit közvetlen elérhető.
Elérési jog	Olvasható
Egyéb	Nem bírálható felül, állapotát tápfeszültség elvételekor nem jegyzi meg

2.6.6. FB változó használata

A funkcióblokkhoz (FB-hez) kapcsolódó változót deklarálni kell használat előtt.

- T: időzítő memória terület

Formátum	**T**x
Leírás	x: decimális szám, jelzi az időzítő számát
Adattípus	BOOL --- időzítő státusz bitje INT --- időzítő aktuális értéke **T**x mindkettő változóhoz hozzáférést biztosít. KincoBuilder programban ha BOOL operandusként szerepel akkor a státuszbit érhető el vele, és INT operandusként pedig az időzítő aktuális értéke.
Példa	T0, T5, T20

- C: számláló memória terület

Formátum	**C**x
Leírás	x: decimális szám, jelzi a számláló számát
Adattípus	BOOL --- számláló státusz bitje INT --- számláló aktuális értéke **C**x mindkettő változóhoz hozzáférést biztosít. A KincoBuilder programban ha BOOL operandusként szerepel akkor a státuszbit érhető el vele, és INT operandusként pedig az időzítő aktuális értéke.
Példa	C0, C5, C20

- RS memória terület

Formátum	**RS**x
Leírás	x: decimális szám, jelzi az RS számát
Adattípus	BOOL --- RS státusz bitje
Példa	RS0, RS5, RS10

- SR memória terület

Formátum	SRx
Leírás	x: decimális szám, jelzi az SR számát
Adattípus	BOOL --- SR státusz bitje
Példa	SR0, SR5, SR10

2.6.7. FB változó memória terület

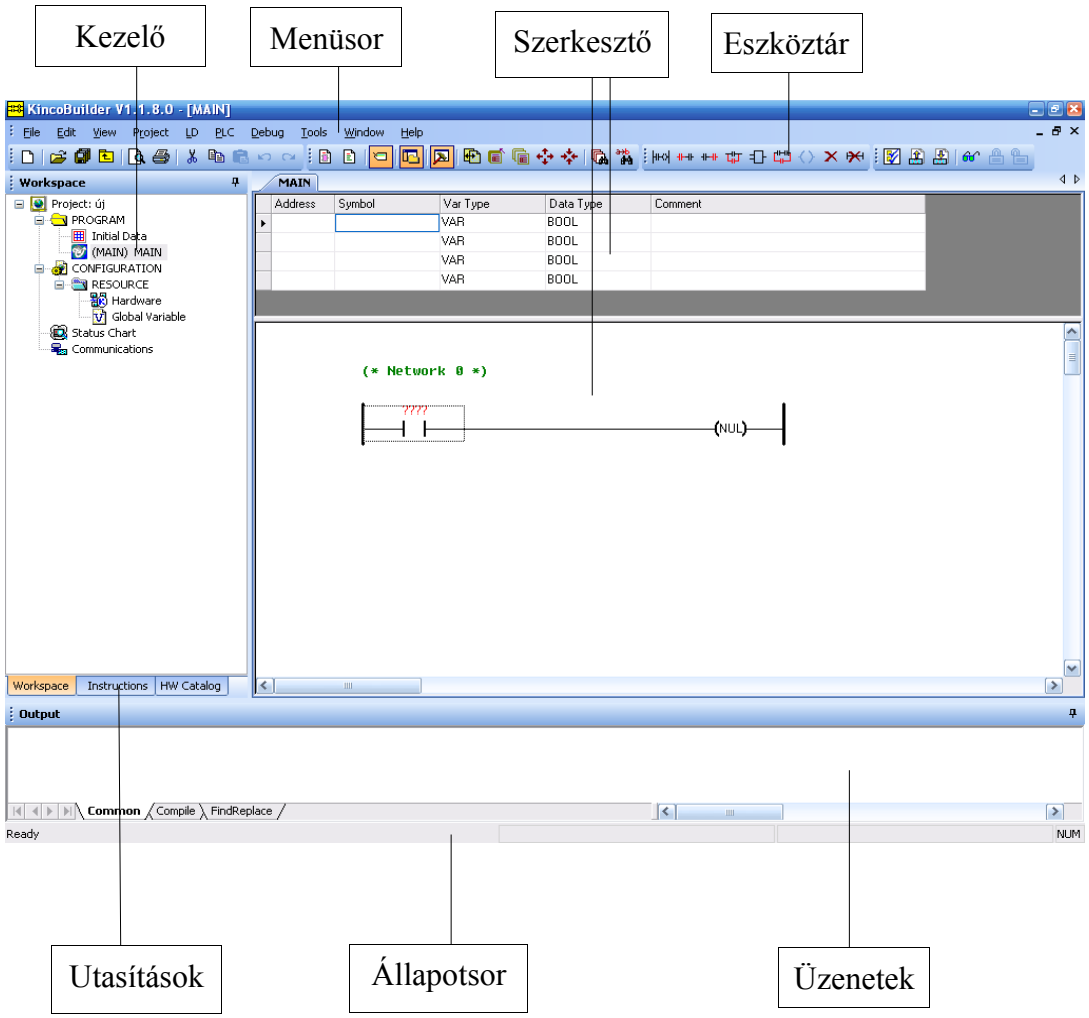
A funkció blokkok számára a különböző típusú PLC-k, a hardvertől függő, különböző méretű memória területet biztosítanak. A használható memória tartományokat a következő táblázat tartalmazza.

		CPU304	**CPU304EX, CPU306**	**CPU306EX, CPU308**
T	Méret	64	128	256
	Tartomány	T0 --- T63	T0 --- T127	T0 --- T255
	Felbontás	T0 --- T3: 1ms T4 --- T19: 10ms T20 --- T63: 100ms	T0 --- T3: 1ms T4 --- T19: 10ms T20 --- T127: 100ms	T0 --- T3: 1ms T4 --- T19: 10ms T20 --- T255: 100ms
	Max időzítés	32767*tartomány	32767*tartomány	32767*tartomány
C	Méret	64	128	256
	Tartomány	C0 --- C63	C0 --- C127	C0 --- C255
	Max számlálás	32767	32767	32767
RS	Méret	-------		32
	Tartomány	-------		RS0 --- RS31
SR	Méret	-------		32
	Tartomány	-------		SR0 --- SR31

3. A KincoBuilder program használata

3.1. A KincoBuilder felhasználói felülete

A program elindítását követően a következő képernyő jelenik meg



- Menü: Ez tartalmazza a KincoBuilder vezérlő parancsait.
- Eszköztár: Gyakran használt parancsok, eszközök gyors elérését teszi lehetővé.
- Állapotsor: Itt láthatóak az aktuális állapot információk.
- Kezelő: Projekthez kapcsolódó objektumok találhatók benne, mint a főprogram és az esetleges alprogramok, globális változók vagy a hardware konfiguráció stb.
- Szerkesztő: Itt található a változó tábla (Variable Table) és a programszerkesztő (IL vagy LD nyelven). A programszerkesztőben készül a vezérlő program, valamint a változó tábla tartalmazza a lokális változókat és a POU be és kimeneti változóit.
- Utasítások: A programban elérhető LD és IL utasítások találhatók a listában.
- Üzenetek: az ablakban különböző típusú információk láthatóak. A fülek segítségével választhatunk a nézetek között: "Compile" ablakban a legutolsó fordítás információi, a "Common" ablakban az utolsó műveletre vonatkozó információk láthatók.

3.2. Program létrehozása a KincoBuilderrel

3.2.1. Projekt összetevők

A programozás megkezdése előtt konfigurálni kell a vezérlőt, definiálni a szimbolikus változókat, és megírni a POU-kat, stb.. A KincoBuilderben ezek az összetevők a képernyő bal oldalán találhatók a „Workspace” részt kiválasztva. Azok az elemek amelyek ”Opcionális” megjelölésűek azokat nem feltétlen kell alkalmazni a projektben, akár mellőzhetőek is.

PROGRAM	Initial Data (Opcionális)	Kezdeti értéket rendelhet szám értékekhez BYTE, WORD, DWORD, INT, DINT és REAL változóként a V memória területen. A CPU modul végrehajtja az inicializálást egyszer, a bekapcsoláskor, majd futtatja a ciklikus végrehajtást.
	Main Program (Főprogram)	A program ciklikusan kerül végrehajtásra, ciklusonként egyszer. Egy projekten belül csak egy főprogram lehetséges.
	Megszakítás rutin (Interrupt routin) (Opcionális)	Meghatározott események kezdeményezhetnek megszakítást, ilyenkor kerül meghívásra a megszakítás rutin. Megszakítás esetén a főprogram futása megszakad, lefut a megszakítási rutin, majd a központi egység visszatér a ciklikus főprogram futtatásához. 16 megszakítás esemény hozható létre a projektben.
	Szubrutinok (Subroutines) (Opcionális)	A szubrutinok csak akkor hajtódnak végre, ha egy megszakítás vagy a főprogram hivatkozik rájuk. Segítségükkel könnyebben áttekinthető programszerkezet alakítható ki. Legnagyobb előnyük, hogy újrafelhasználhatóak, egyszer kell csak megírni a programrészt a szubrutinba és utána annyiszor hajtható végre ahányszor csak szükséges. 16 szubrutin hozható létre a projektben.
CONFIGURATION	Hardware	Itt lehet beállítani a használt KINCO-K3 modulokat, meghatározni a címeket, funkció paramétereket, stb..
	Global variables (Opcionális)	Itt lehet deklarálni a projektben szükséges globális változókat.

3.2.2. A projekt elérési útja a merevlemezen

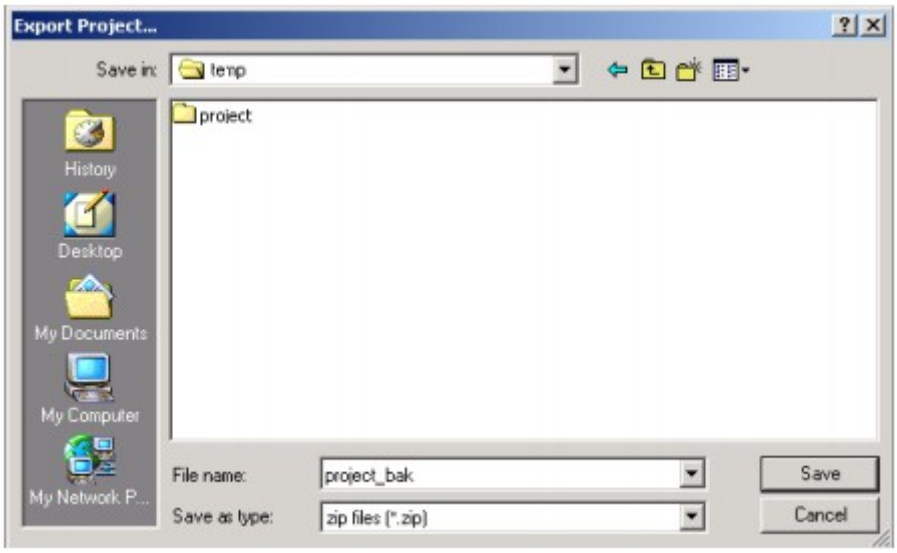
Amikor létrehozunk egy projektet, a KincoBuilder először megkérdezi, hogy a merevlemezen hol tárolja a projektet. A megadott könyvtárban létrehoz egy állományt (".kpr" kiterjesztéssel). Mindemellett létrejön a kiválasztott helyre egy, a projekttel azonos nevű mappa, amelyben a projekt programfájljai, változók fájljai és más átmeneti fájlok kerülnek. Például, ha létrehoz egy "example" nevű projektet a "c:\temp" könyvtárban, a projekt elérési útvonala "c:\temp\example.kpr", és a többi fájl a "c:\temp\example" mappába kerül.

3.2.3. Projekt importálás és exportálás

A KincoBuilder-ben a menüben található a [File] → [Import] és a [File] → [Export] parancsok lehetőséget adnak a projekt importálására és exportálására.

- **[Export...]**
  Összetömörít az összes, az aktuális projekthez tartozó fájlt egy mentési állományba (".zip" kiterjesztéssel).

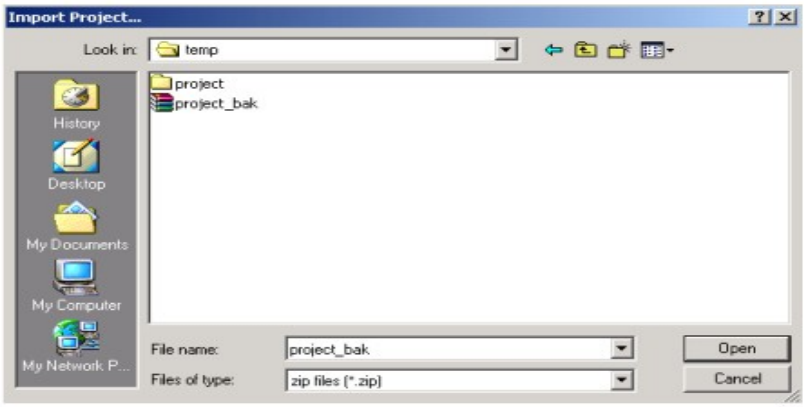
1. Válassza a [File] → [Export] gombot.
   Megjelenik az "Export Project..." ablak



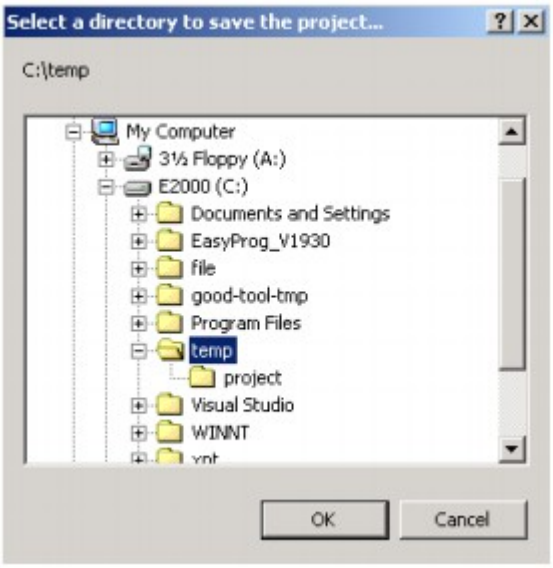
2. Adjuk meg az elérési utat, írjuk be a fájlnevet és kattintsunk a [Save] gombra, az exportálás befejezéséhez.

- **[Import...]**
  Projekt importálás során egy, előzőleg a programmal mentett állomány (".zip" kiterjesztéssel) megnyitása hajtható végre.

1. Kiválasztjuk a [File] → [Import] gombot.
   Megjelenik az "Import Projekt..." párbeszédablak:



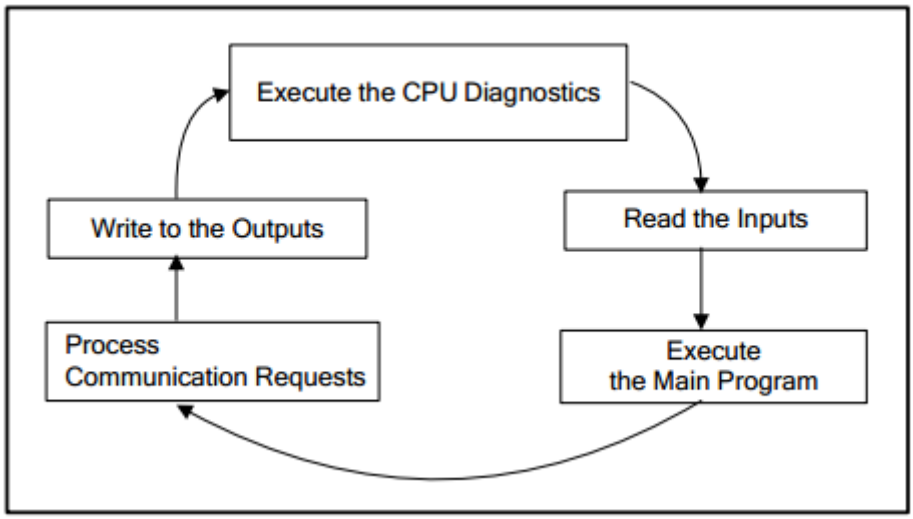
2. Kiválasztjuk a mentett fájlt és az [Open] gombra kattintunk.
   A következő párbeszédablak jelenik meg, ahol meg kell határozni a mappát, ahova a program kicsomagolja a projektet.



3. Kiválasztjuk a mappát és az [OK] -ra kattintunk, és a projekt a kiválasztott mappába kerül, és a szerkesztő megnyitja azt.

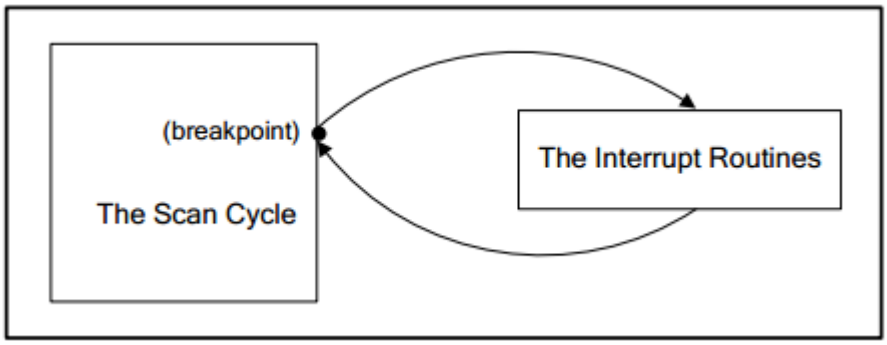
3.3. A központi egység ciklikus működése

A CPU modul feladatok sorozatát hajtja végre folyamatosan és ciklikusan. A központi egység a megszakítás rutint vagy rutinokat és a főprogramot futtatja ciklikusan. A főprogram ciklusonként egyszer fut le, a megszakítás rutin pedig csak abban az esetben, ha a megadott megszakítási esemény létrejön.
A központi egység a következő ábra szerint hajtja végre a feladatokat:



- CPU ellenőrzése (Executing the CPU diagnostics): A CPU modul öntesztelést végez, ellenőrzi a memória területet, és a kiegészítő modulokat.
- Bemenetek olvasása (Read the inputs): A KINCO-K3 PLC beolvassa az összes fizikai bemenetet és ezeket az értékeket beírja a bemeneti memória területre.
- Felhasználói program végrehajtása (Executing the user program): A CPU modul folyamatosan végrehajtja a főprogramban lévő utasításokat és frissíti az érintett memória területeket.
- Kommunikációs kérések feldolgozása (Processing communikation reguests)
- Kimenetek írása (Writing to the outputs): A KINCO-K3 kiírja a kimeneti memória területen tárolt értékeket a fizikai kimeneti csatornákra.

A megszakítások a ciklikus működés során bármikor végrehajtódhatnak. Ha megszakítás jön létre, a CPU átmenetileg megszakítja a ciklikus működését és lefuttatja a megszakítási rutint. Amikor a megszakítási rutin befejeződik, a ciklus működése a megszakítási ponttól folytatódik.



3.4. Számítógép és a KINCO-K3 PLC összekapcsolása

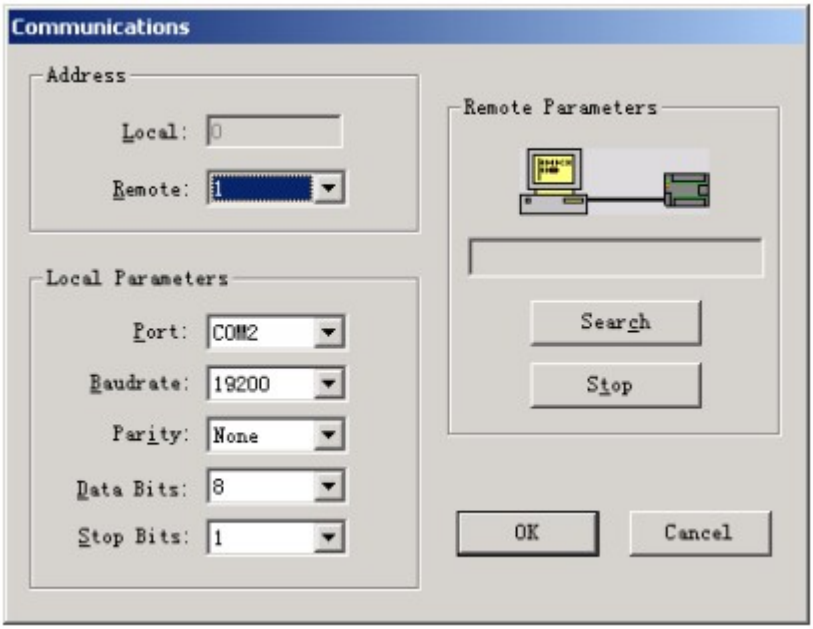
A központi egységen található RS232 vagy RS482 soros kommunikációs port biztosítja a kommunikációt a többi eszközzel, beleértve a számítógéppel történő kommunikációt. A legegyszerűbb csatlakozási megoldás a központi egység RS-232 portja, a megfelelő programozó kábel alkalmazásával.

1. Nyissuk meg a KincoBuilder programot, nyissunk meg egy már meglevő projektet vagy hozzunk létre egy újat.
   Kösse össze a számítógép soros portját a CPU modul soros portjával a programozó kábel segítségével.
   Megjegyzés: az RS232 kapcsolat megbontása nem javasolt, amíg valamelyik eszközt ki nem kapcsoljuk (CPU modult vagy a számítógépet), ellenkező esetben sérülhet a port.

2. Konfiguráljuk a számítógép soros portját.
   Megjegyzés: A kommunikáció csak abban az esetben működik, ha a központi egység paraméterei megegyeznek a soros vonal beállításával.

a) Válasszuk ki a [Tools] → [Communications...] gombot, vagy kattintsunk duplán a [Communications] feliratra a Manager ablakban, vagy jobb egérgombbal kattintsunk a [Communication] feliratra aztán válasszuk az [Open] gombot a felugró ablakban.

Megjelenik a "Communications" párbeszédablak:



b) Válasszunk a központi egység címét a [Remote] legördülő listából; Válasszuk azt a COM portot amit a számítógép használ a [Port] listában; Állítsuk be a választott COM port paramétereit ([Baudrate], [Parity], [Data Bits], [Stop Bits]) a CPU portjának megfelelően, és kattintsunk az [OK] gombra, hogy elfogadjuk és mentsük a beállításokat.

Ha nem ismerjük a CPU port kommunikációs paramétereit, hogyan tudhatjuk meg?
Két módon lehetséges:

- Válasszuk a számítógépen használt portot, majd kattintsunk a [Search] gombra, hogy a KincoBuilder automatikusan felismerje a CPU modul kommunikációs paramétereit. Ez néhány másodperctől néhány percig is eltarthat. Ha befejezte a keresést, a KincoBuilder automatikusan beállítja a megfelelő paramétereket a számítógépen.

- Kapcsoljuk ki a tápellátását a CPU modulnak; Állítsuk a műveleti kapcsolót STOP állásba; majd kapcsoljunk rá tápfeszültséget, és most a CPU port az alapértelmezett beállításokat használja: Station number, 1; Baudrate, 19200; None parity check; 8databits; 1 stop bit. Beállíthatjuk a számítógép soros COM portját ezen paraméterek alapján.
Megjegyzés: Ne változtassuk meg a kapcsoló pozícióját ameddig módosítjuk a CPU kommunikációs paramétereit.

3. Miután kész a kommunikáció paramétereinek beállítása, tudjuk programozni a KINCO-K3 PLC-t.

3.5. CPU kommunikációs paraméterek módosítása

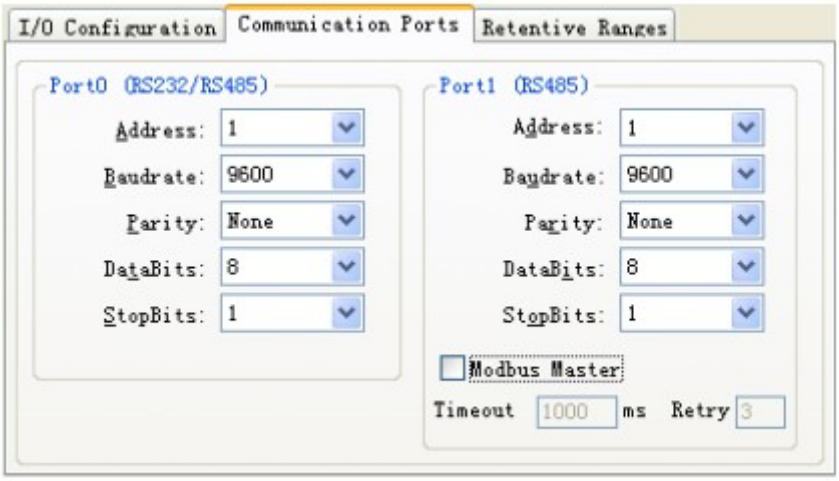
Miután csatlakoztattuk a CPU modult a számítógéphez, módosítani tudjuk a kommunikációs paramétereket a KincoBuilder programmal.

(1) Először megnyitjuk a "Hardware" ablakot a következő módokon:

- Dupla kattintással a [Hardware] feliratra a Manager ablakban;
- Jobb gombbal kattintva a [Hardware] feliratra majd az [Open...] gombra kattintva a felugró ablakban.

A hardware ablak felső részében egy részletes listát találunk a PLC modulokról táblázatban, ezt konfigurációs táblázatnak (Configuration Table) hívjuk. A konfigurációs táblázat a ténylegesen beállított konfigurációt mutatja.
A hardware ablak alsó része a konfigurációs táblázatban kiválasztott modul paramétereit mutatja, és ezt paraméter ablaknak hívjuk (Parameters Window).

(2) Válasszuk ki a CPU modult a konfigurációs táblázatban, aztán a [Communication Ports] fület a paraméter ablakban. Most már tudjuk módosítani a kommunikációs paramétereket, ahogy itt láthatjuk:



(3) Miután módosítottuk a beállításokat le kell tölteni a CPU modulba.
Megjegyzés: A konfigurált paraméter nem fog életbe lépni ha nem töltjük le!

3.7. Példa: Projekt létrehozása lépésről lépésre

A kezdőknek a könnyebb érthetőség kedvéért a következőkben egy egyszerű példát mutatunk be, és lépésről lépésre létrehozunk és nyomon követünk egy projektet.

A következő projektet hozzuk létre:

- Projekt neve: „Example”;
- Hardware: KINCO-K306-24DT CPU modul;
- Vezérlő logika: Q0.0---Q0.7 állapotának ciklikus váltása. A jobb struktúra miatt két POU-t használunk: a szubrutin „Demo” vezérlő logikát tartalmazza, a fő program neve „Main” ahol a „Demo” meghívásra kerül.

1. Először nyissuk meg a KincoBuildert.

2. A KincoBuilder program néhány alapértelmezett beállítása módosítható, ha szükséges.

- Válasszuk a [Tools] → [Options...] gombot
  Az "Options" párbeszédablak felajánlja a változtatható beállításokat, pl.: az alapértelmezett programozási nyelv, stb. Ezeket a beállításokat automatikusan elmenti.

3. Hozzunk létre egy új projektet, melynek kétféle módja lehetséges:

- Válasszuk a [File] → [Nex project...] gombot

- Kattintsunk a  ikonra az eszköztárban.
  A "New projekt..." párbeszédablakban megadjuk az új projekt nevét és kiválasztjuk a mappát, majd a [Save] gombra kattintunk, és kész az új projektünk.

Ebben a példában a "D:\temp" könyvtárat választottuk a projekt mappának, a projektet pedig "Example"-nek neveztük el.

4. Változtassuk meg a hardver konfigurációt, bár a programozás során erre a későbbiekben is lehetőségünk nyílik. A hardver konfiguráció fontos a projekt szempontjából, ezért javasolt ezt beállítani elsőnek.

   Nyissuk meg a "Hardware" ablakot ezen módok valamelyikével:
- Dupla kattintás a [Hardware] szövegre a Manager ablakban;
- Jobb egérgombbal kattintsunk a [Hardware] szövegre és a felugró ablakban válasszuk az [Open] gombot.

Ebben a példában a KINCO-K306-24DT az alkalmazott modul, alapértelmezett paramétereivel.

5. Hozzuk létre a példaprogramokat.

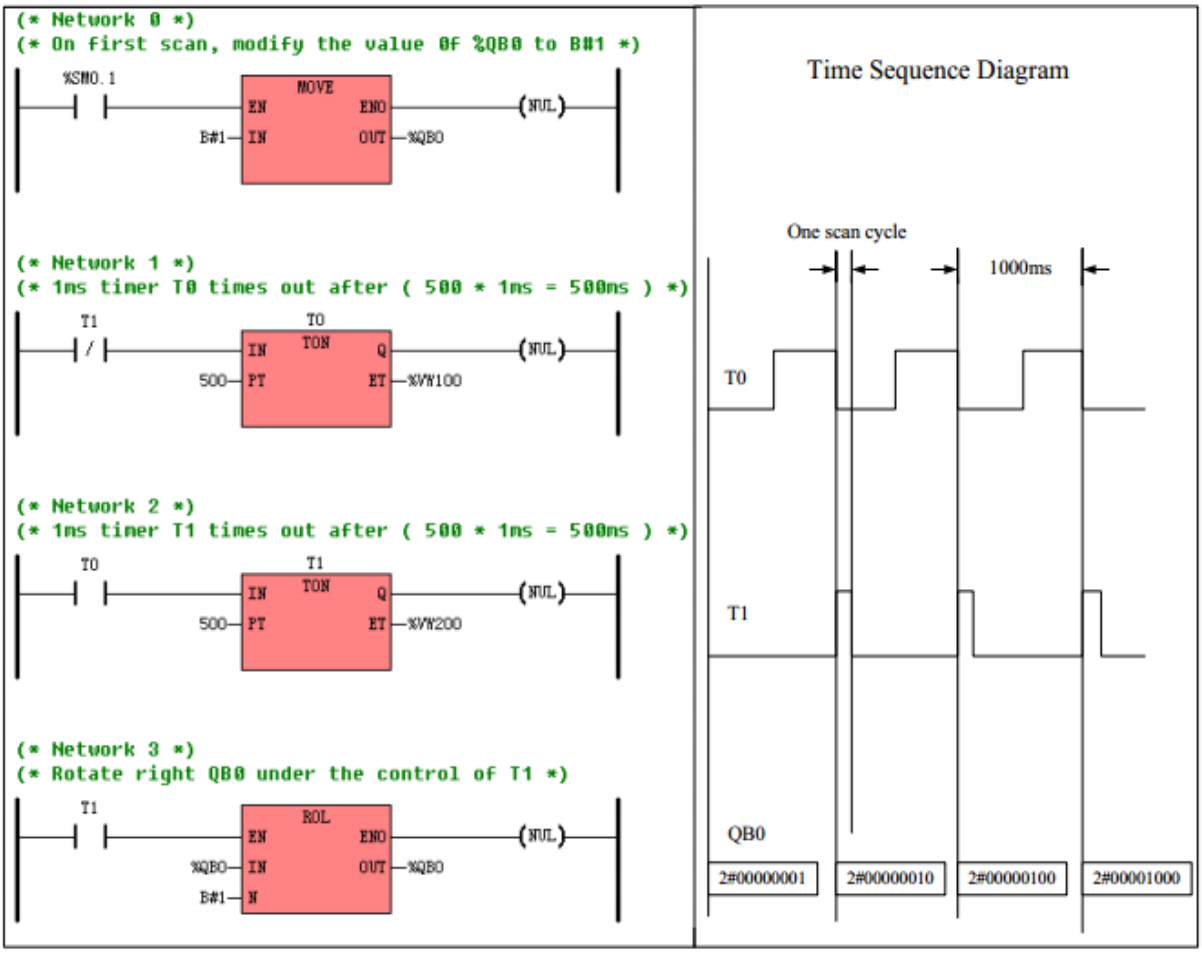
A KincoBuilder IL (utasítás lista) és LD (létra diagram) programozási nyelvet alkalmaz.
Kiválaszthatjuk a [Project] → [IL] vagy a [Projekt] → [LD] menüknél az aktuális POU programozási nyelvét.
Példánkban, a főprogramot "Main" és a szubrutint "Demo" LD nyelven írtuk.

a) Főprogram
Amikor létrehozunk egy új projektet a KincoBuilder automatikusan létrehoz egy üres főprogramot, amit "MAIN" -nek hív.

b) Szubrutin létrehozható több különböző módon:

- Válasszuk a [projekt] → [New Subrutine] gombot
- Kattintsunk a  ikonra az eszköztáron
- Jobb egérgombbal a [PROGRAM] szövegre a Manager ablakban, aztán válasszuk a [New Subrutine] gombot a felugró ablakban.

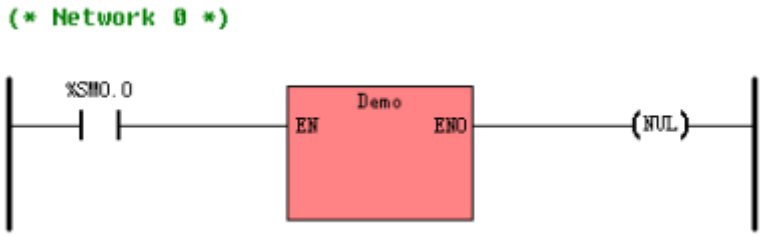
Így létrehoztuk az új szubrutint, amelynek az alapértelmezett neve "SBR_0". A szubrutin a következő blokkokból építhető fel:



Miután befejezte a program bevitelét, módosítható a szubrutin neve a következőképpen: Zárjuk be a szubrutin ablakot; Jobb egérgombbal kattintsunk a "(SBR00) SBR_0" szövegre a Manager ablakban, aztán válasszuk a [Rename] gombot a felugró ablakban és módosítsuk a nevet "Demo", vagy válasszuk a [Properties...] gombot, és módosítsa a "Property" párbeszédablakban.

c) Főprogram módosítása

Most, hogy befejeztük a "Demo" szubrutint, térjünk vissza a főprogramba és adjuk hozzá a következő utasítást, mely a DEMO szubrutint hívja meg.



6. Miután kész vagyunk az egész projekttel, le kell fordítanunk. Amikor lefordítjuk a projektet a KincoBuilder automatikusan menti a módosításokat. A következő módokon lehet elindítani a fordítást:

- Válasszuk a [PLC] → [Compile All] gombot
- Kattintsunk a  ikonra az eszköztáron
- Használjuk az F7 gyors-gombot

A "Compile" fül az "Output Window-ban" mutatja az utolsó fordítási üzeneteket. Az esetleges hibáknál kattintsunk kétszer a hibaüzenetre a "Compile" ablakban, és a program a hibás részre ugrik. A programban levő hibákat ki kell javítani, hogy a program fordítása sikeres legyen.

7. Most már letölthetjük a programunkat. Megjegyzés: ha szükséges módosíthatjuk a kommunikációs paramétereit a számítógép soros portján a [Communications] párbeszédablakban.

   A program letöltése a következő módokon végezhető el:

- Válasszuk a [PLC] → [Download...] menüt
- Kattintsunk a  ikonra
- Használjuk az F8 gyors-gombot

Ha a CPU modul RUN módban van a párbeszéd ablak figyelmeztet, hogy a letöltés idejére átkapcsol STOP módba, ehhez válassza a YES gombot.

A projekt letöltését követően a CPU modul RUN módba vált, és a jelző ledek Q0.0---Q0.7 villogni fognak.

8. A program futása nyomon követhető, ha kiválasztjuk a [Debug] → [Monitor] menüpontot vagy a  ikonra kattintunk az eszköztáron, és így a KincoBuilder megmutatja az összes használatban lévő változó értékét, amit a program használ.

A CPU modul leállításához kapcsoljuk a vezérlő kapcsolót STOP állásba vagy válasszuk a [Debug] → [Stop] menüpontot.

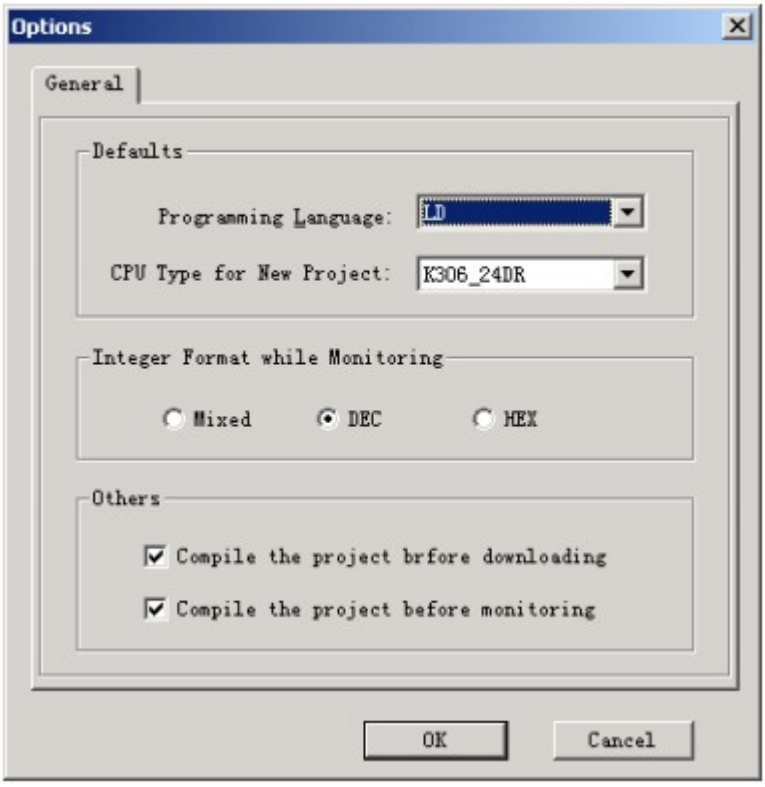
4. KincoBuilder használata – alap funkciók

Ez a fejezet részletesen bemutatja a KincoBuilder programot, beleértve az alkalmazható funkciókat és a műveleti lépéseket. Az alap fogalmaktól kiindulva, ez a fejezet segíthet jobban megérteni a KincoBuilder programot.

4.1. Szoftver beállítások konfigurálása

Szükségünk lehet néhány alap beállítás módosítására a KincoBuilder programban, pl.: az alapértelmezett programozási nyelv és a CPU típusának beállítása az új projekthez. A KincoBuilder automatikusan elmenti a beállításokat.

Válasszuk a [Tools] → [Options...] menüt, és a következő párbeszédablak jelenik meg:



General fül

- Defaults
  - Programming Language:
    Válasszuk ki a programozási nyelvet az új projekthez, IL vagy LD.
  - CPU Type for New Projects:
    Válasszuk ki az alapértelmezett CPU típust. Új projekt létrehozásakor ezt a típust rendeli a hardware konfigurációhoz a program.

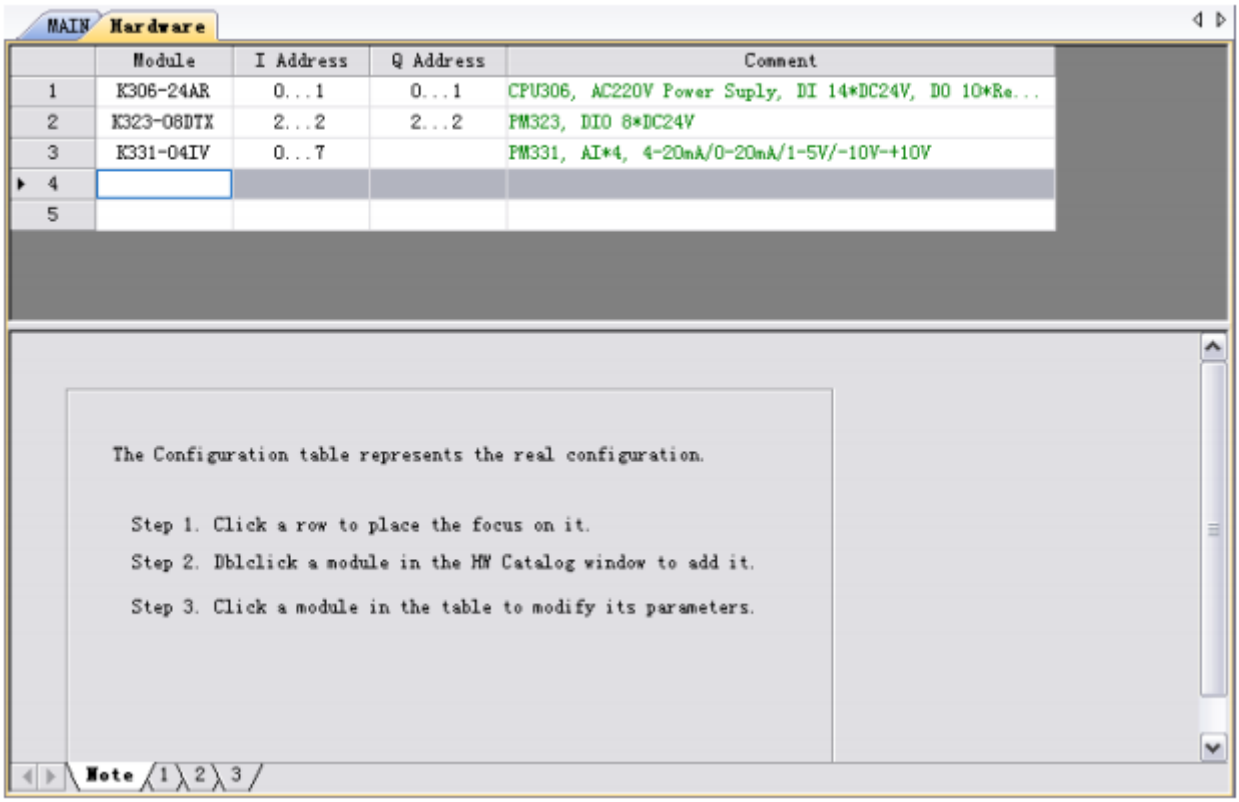
- Integer Format While Monitoring
  Válasszuk ki az egész számok megjelenési formátumát, monitor módban.
  Mixed: Az INT és DINT értékek decimális formában láthatóak, a BYTE, WORD és DWORD értékek hexadecimális formában jelennek meg.
  DEC: Minden egész szám decimálisan jelenik meg.
  HEX: Minden egész szám hexadecimálisan jelenik meg.

- Others

- Compile the projekt before downloading:
  Ha ez ki van választva, a KincoBuilder automatikusan lefordítja az aktuális projektet letöltés előtt.
- Compile the projekt before monitoring:
  Ha ez ki van választva, a KincoBuilder automatikusan lefordítja a programot monitor üzemmódba kapcsolás előtt.

4.2. Hardware konfiguráció

A programozás első lépéseként javasolt elkészíteni a hardver konfigurálását. Az új projekt létrehozásakor az alapértelmezett központi egységet a program hozzáadja a hardver konfigurációhoz. A központi egység típusa a későbbiekben módosítható. A következő ábrán a "Hardware" ablak látható, mely két részre bontható.



- Konfigurációs táblázat
  A felső részen lévő táblázat a hardver ablaknak a PLC felépítését mutatja. A konfigurációs táblázat a valós konfigurációt kell, hogy mutassa: a modulokat olyan sorrendben kell a táblázatban feltüntetni, ahogy azok a valóságban egymáshoz kapcsolódnak.

- Paraméter ablak
  Az alsó része a hardver ablaknak a konfigurációs táblázatban kiválasztott modul paramétereit mutatja.

A hardver konfiguráció addig nem lép érvénybe, amíg azt le nem töltjük a CPU modulba.

4.2.1. Hardver ablak megnyitása

A "Hardware" ablakot a következő módokon lehet megnyitni:
- Dupla kattintással a [Hardware] szövegre a Manager ablakban.
- Jobb egérgombbal a [Hardware] szövegre, aztán az [Open] gombra kattintva a felugró ablakban.

4.2.2. Modul hozzáadása/eltávolítása

- Modul hozzáadása

  Modult a következő lépésekkel lehet hozzáadni:
  - A konfigurációs táblán az egyik sorra kattintunk. Ha már tartalmaz modult a kiválasztott sor, először ki kell azt törölni.
  - A PLC katalógus ablakban lévő modulok valamelyikére dupla kattintással a kijelölt sorba teszi a modult a konfigurációs táblázatban.

  Az első sorba csak a CPU modul helyezhető, az összes többibe lehet a kiegészítő modulokat. Két modul között nem lehet üres sort hagyni. Ha üres sort hagyunk a KincoBuilder program egy hibaüzenetet fog adni ha el akarjuk menteni, vagy le akarjuk fordítani a projektet.
  A CPU304, CPU306 és a CPU308 típusoknál előre meg van határozva a maximum I/O csatornák száma. Ha a hozzáadott modulok csatornáinak száma nagyobb mint a megengedett maximális, a KincoBuilder nem engedélyezi további modulok hozzáadását, és hibaüzenetet küld mentéskor vagy fordításkor.

- Modul eltávolítása
  A következő lépésekkel lehet modult eltávolítani:
  - Kattintsunk a törölni kívánt modulra a konfigurációs táblán, majd használjuk a Del gombot.
  - Jobb egérgombbal a törölni kívánt modulra, majd a [Remove] gombot választjuk a felugró ablakban.

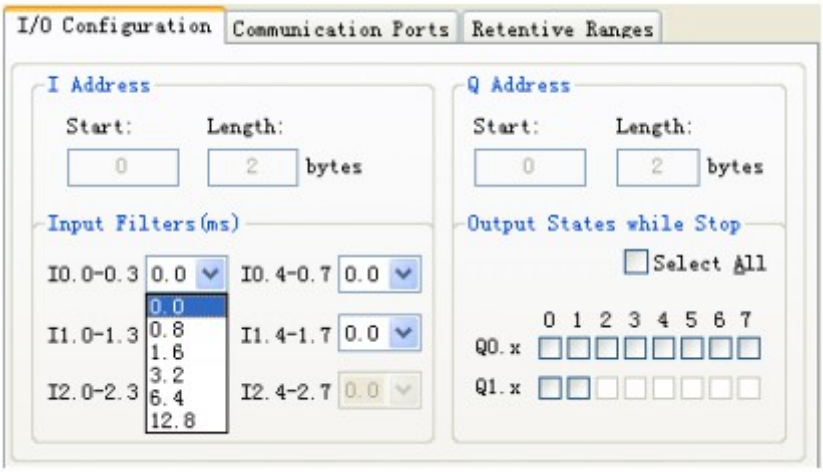
4.2.3. Modul paraméterek beállítása

Ha elrendeztük a moduljainkat a konfigurációs táblázatban, akkor beállíthatjuk a paramétereiket. A konfigurációs táblázatban jelöljük ki a beállítani kívánt modult, a képernyő alsó részén megjelenik a hozzá kapcsolódó paraméter ablak.

Megjegyzés: A modulok címei között, ha azonos memória területeket érintenek (I, Q, AI vagy AQ) nem lehet átfedés!

4.2.3.1. CPU paraméterei

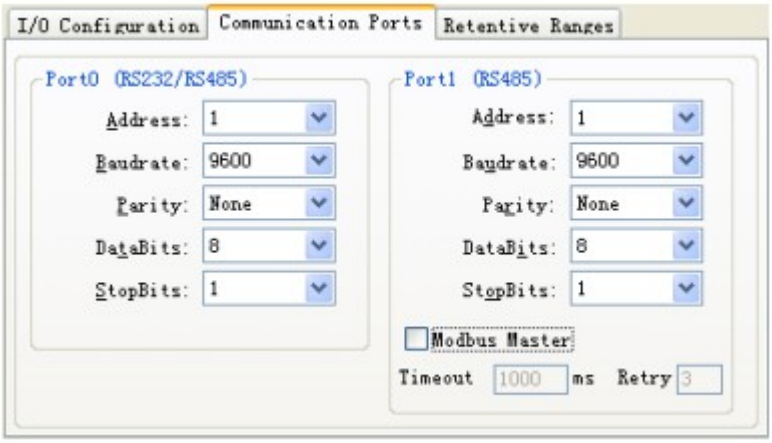
(1) [I/O Configuration] fül
Itt lehet beállítani a CPU I/O paramétereit, az ábrán látható módon:



- Input: CPU digitális csatornáinak a konfigurálása.
  - I Address: a DI csatorna kezdő bájt címe az I területen. Ez fixen 0.
  - Input Filters: bemeneti szűrő értékének megadása (ms). Alkalmazásával kiszűrhetők a bemeneti zajok. A bemeneti érték csak abban az esetben változik meg, ha a jel értéke a megadott ideig változatlan marad.

- Output: DO csatorna konfigurálása.
  - Q Address: a DO csatorna kezdő bájt címe a Q területen. Ez fixen 0.
  - Output States while STOP: kimenet állapota ha a CPU megáll. Az alapértelmezett érték, ha megáll a CPU, akkor az összes kimenet kikapcsolt állapotba kerül.

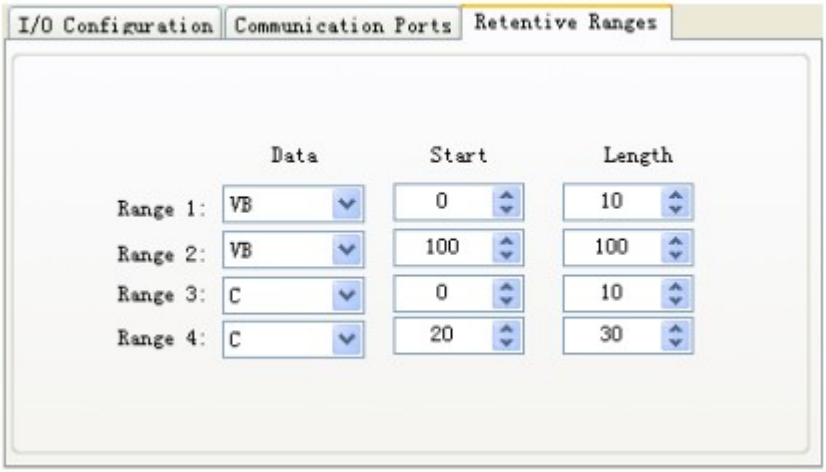
(2) [Communication Ports] fül
Itt lehet a CPU modul soros kommunikációs portjainak (Port0 és Port1) paramétereit beállítani.



- Port0
  - Address: A PLC címe, Ez lesz az eszköz Modbus RTU slave címe is.
  - Baudrate: Válasszuk ki a kívánt átviteli sebességet.
  - Parity: Válasszuk ki a kívánt paritást
  - DataBits: Válasszuk ki az adatbitek számát (8)
  - StopBits: Stopbitek száma.

- Port1: RS485-ös port. Néhány CPU típusnak csak egy soros portja van (Port0), és a Port1 nem elérhető. A kommunikációs paraméterek beállításakor, hasonlóan kell eljárni mint a PORT0 esetében. További paraméterek:
  - Modbus Master: Ha a jelölőnégyzet ki van választva a Port1 Modbus RTU master-ként működik.
  - Timeout: A Modbus master időtúllépését lehet beállítani.
  - Retry: Beállíthatjuk, hogy hányszor próbáljon a master újra kommunikálni a slave-vel, ha nem érkezik válasz.

(3) [Retentive Ranges] fül
Itt lehet definiálni a nem felejtő memória tartományokat a RAM területen, hogy kikapcsolás után is megmaradjanak a szükséges adatok.



- Range 1, Range 2, Range 3, Range 4
  - Data
    V terület vagy Számláló (C) terület.
    Számlálók esetében csak az aktuális érték tárolható el.
  - Start
    A nem felejtő memória kezdőcíme
  - Lenght
    A nem felejtő memória mérete, byte-ban megadva.

A fenti ábrán a következő nem felejtő memória tartalmak vannak megadva:

- Range 1 (%VB0 --- %VB9),
- Range 2 (%VB100 --- %VB199),
- Range 3 (C0 --- C9)
- Range 4 (C20 --- C49)

4.2.3.2. DI modul paraméterei

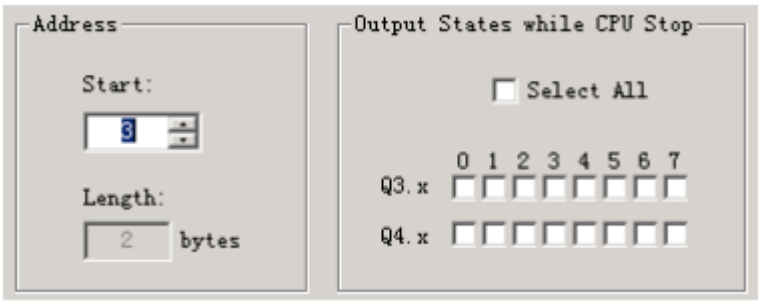
A DI modul paramétereit a következő módon lehet állítani:

Start
A kezdő bájt címe az I területnek.

Length
A modul címterületének hossza.
Ez az érték fix, függ a modul DI csatornáinak számától.

A fenti ábra mutatja, hogy a modulnak 8 DI csatornája van, melyek 1 byte-ot foglalnak a memóriában. A modul által használt címtartomány %IB3 címen kezdődik, tehát a bemeneti pontok %I3.0 --- %I3.7 címek között tárolódnak.

4.2.3.3. DO modul paraméterei



Start
A kezdő bájt címe a Q területnek.

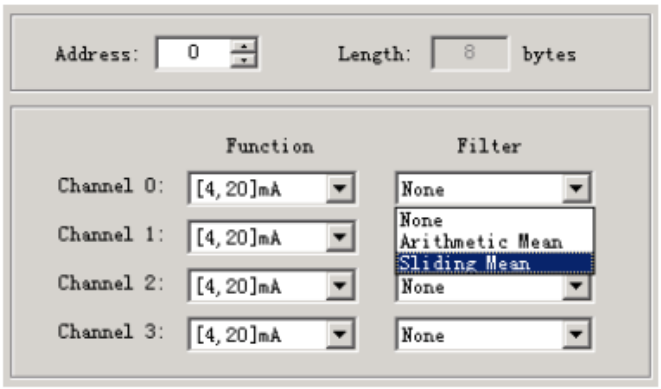
Length
A modul címterületének hossza.
Ez az érték fix, függ a modul DO csatornáinak számától.

A fenti ábrán, a modulnak 16 DO csatornája van, melyek 2 byte-ot foglalnak a memóriában. A modul által használt címtartomány %IQB3 címen kezdődik, tehát a bemeneti pontok %I3.0 --- %I4.7 címek között tárolódnak.

Kimenetek állapota STOP üzemmódban

- Itt lehet beállítani, a digitális kimenet állapotát, ha a CPU megáll. Ha kipipáljuk a jelölő négyzetet a kimenet ON (1) állapotba kerül ha a CPU leáll. Az alapértelmezett állapot az OFF (0).

4.2.3.4. AI modul paraméterei



Address
A kezdő bájt címe a AI területnek, a modulon található analóg bemenetek címzése innen kezdődik. Minden egyes analóg bemenet 2 byte-ot foglal a memóriában. A kezdőcímnek párosnak kell lennie.

Length
A modul címterületének hossza.
Ez az érték fix, függ a modul AI csatornáinak számától.

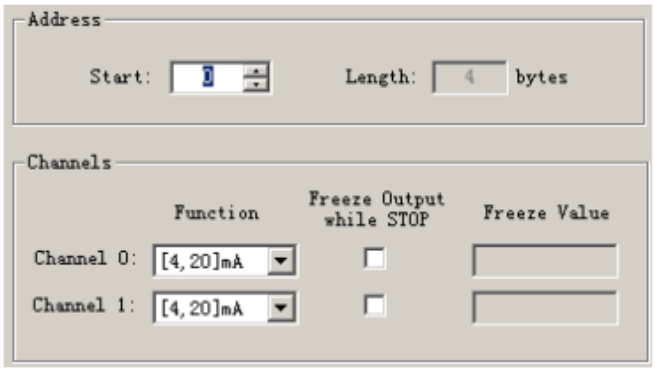
Function
Csatorna bemeneti típusa, pl.:4.20mA, 1-5V, stb.

Filter
Analóg bemeneti csatorna szoftveres szűrése. Gyorsan változó analóg jelek esetén a szűrő segíthet a mérendő mennyiség stabilizálásában. *Ha a vezérlés megköveteli a gyorsan változó jelek kezelését, javasolt a bemeneti szűrő kikapcsolása.*

- No: bemeneti szűrő kikapcsolása
- Arithmetic Mean (számtani közép): az érték a minták számtani közepe lesz.
- Sliding Mean (csúszó átlag): az értéket a definiált minták csúszó átlaga határozza meg

A fenti ábra mutatja, hogy a modulnak 4 AI csatornája van, és az %AIW0 címen kezdődik, tehát a bemeneti értékek a memóriában a %AIW0 --- %AIW6 címek között találhatók.

4.2.3.5. AO modul paraméterei



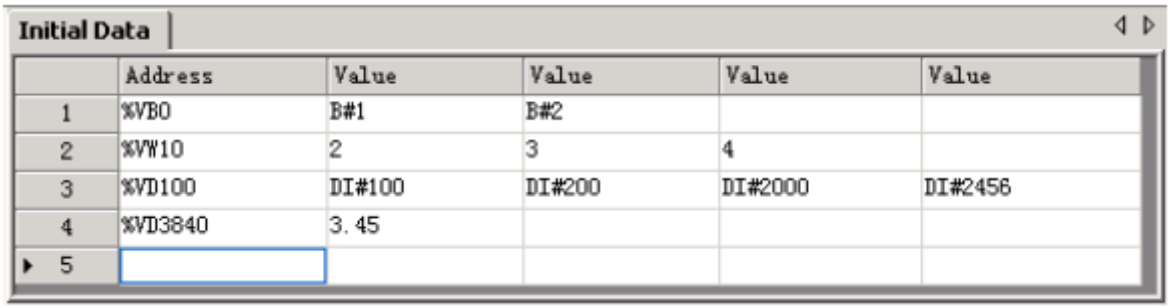
Address
A kezdő bájt címe a AO területnek, a modulon található analóg kimenetek címzése innen kezdődik. Minden egyes analóg kimenet 2 byte-ot foglal a memóriában. A kezdőcímnek párosnak kell lennie.

Length
A modul címterületének hossza.
Ez az érték fix, függ a modul AO csatornáinak számától.

A fenti ábra mutatja, hogy a modulnak 2 AO csatornája van, és az %AQW0 címen kezdődik, tehát a kimeneti értékeke a memóriában %AQW0 --- %AQW2 címek között találhatók.
Freeze Output while STOP: kimeneti csatorna értékének beállítása egy fix értékre, ha a központi egység STOP állapotba kerül. A kívánt érték a **Freeze Value** pontban adható meg.

A kezdeti érték táblázat

Az Initial Data Table-ben a V terület változóinak adhatunk számszerű kezdő értékeket BYTE, WORD, INT, DINT és REAL típusokhoz. A CPU modul bekapcsoláskor a megadott címekre betölti a megadott értékeket, majd megkezdődik a ciklikus végrehajtás.


Initial Data

	Address	Value	Value	Value	Value
1	%VB0	B#1	B#2		
2	%VW10	2	3	4	
3	%VD100	DI#100	DI#200	DI#2000	DI#2456
4	%VD3840	3.45			
5					

A kezdeti érték táblázat megnyitása

- Dupla kattintással az [Initial Data] feliratra a Manage ablakban.
- Jobb egérgombbal az [Initial Data] feliratra aztán az [Open] gombra a felugró ablakban.

Cellák szerkesztése

Rákattintunk a cellára amit szerkeszteni szeretnénk, és már be is tudjuk vinni a kívánt adatot, majd a billentyűzet nyilaival tudunk más cellákat kijelölni és szerkeszteni.
Amikor egy celláról megszűnik a kijelölés akkor a benne lévő tartalom rögzül. Az ENTER billentyű segítségével is tudjuk rögzíteni a bevitt adatot.
A helytelen adat piros színűre változik.

4.3.3. Kezdeti érték táblázat kitöltése

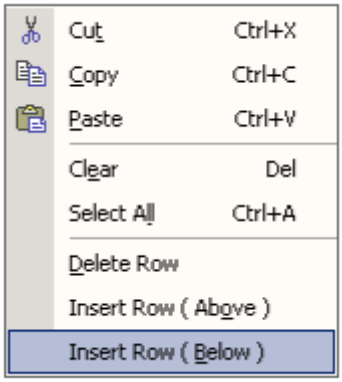
A táblázatnak 5 oszlopa van; Egy Adress (cím) oszlop és 4 Value (érték).

1. Adja meg a címet az Adress oszlopba
2. Adja meg értéket vagy értékeket a a Value oszlopba vagy oszlopokba. Több érték megadása esetén a KincoBuilder a következőképpen jár el, a fenti táblázatból a példa:
   - A „%VB0” cím értéke B#1 lesz, „%VB1” címre pedig B#2 kerül
   - A második sorban %VW10, %VW12, %VW14 értéke 2, 3, 4

Ahogy a fenti ábrán látjuk, az első sor mutatja, hogy a B#1 mutat a %VB0-ra és a B#2 mutat a %VB1-re; a második sorban pedig a 2,3 és 4 a %VW10-re,%VW12-re illetve a %VW14-re mutatnak.

Rendezés: az **Address** fejlécre kattintva lehet rendezni a táblát

Új sor beillesztése



Kattintson a táblázat egy sorára az egér jobb gombjával, ekkor megjelenik a bal oldalt látható felugró menü.

Delete Row:
A kijelölt sor törlése.

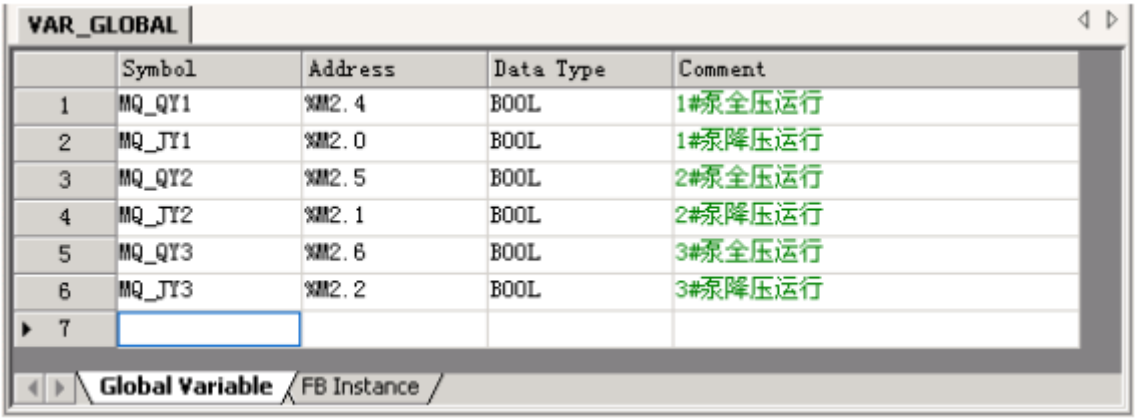
Insert Row (Above):
A kijelölt sor fölé beszúr egy új sort.

Insert Row (Below):
A kijelölt sor alá beszúr egy új sort.

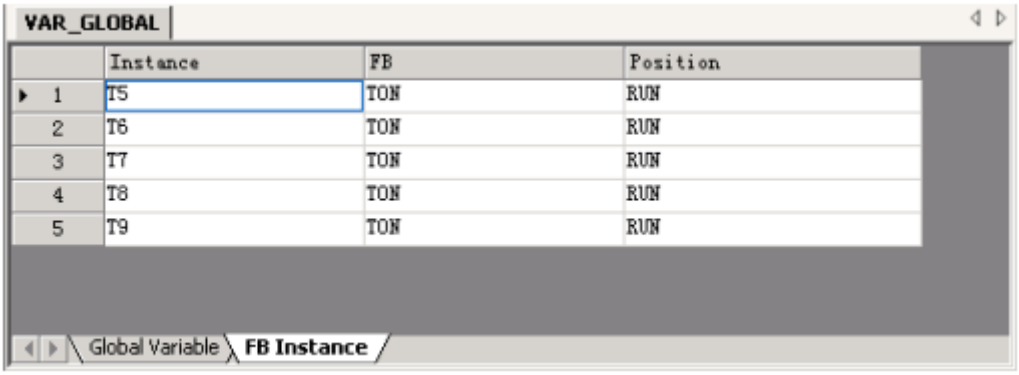
4.4. A Globális változó táblázat

A globális változó táblázat két részből áll: a Global Variable fülből és a FB Instance fülből.

- A Global Variable fül
  Itt lehet deklarálni a globális változókat, mint a következő ábrán látható:



- FB Instance fül



A funkcióblokkok működéséhez szükséges FB Instance változókat a KincoBuilder generálja, így a táblázatban megjelenő adatok nem módosíthatók.

4.4.1. Globális változó táblázat megnyitása

Három módon lehet megnyitni a globális változó táblát:

- Dupla kattintással a [Global Variable] szövegre a Manager ablakban.
- Jobb egérgombbal a [Global Variariable] szövegre kattintva, majd az [Open] gombot választva.
- Válasszuk a [Project] → [Global Variable] almenüt.

4.4.2. Globális változók deklarálása

A táblázatnak 5 oszlopa van: Symbol (megnevezés), Adress (cím), Data Type (adattípus) és Comment (megjegyzés).

1. Nyissuk meg a globális változó ablakot és válasszuk a Global Variable fület.
2. Adjuk meg a létrehozni kívánt változó nevét a Symbol oszlopban
3. Adjuk meg a változó címét az Address oszlopban
4. Válasszuk ki a változó adattípusát
5. Opcionálisan megjegyzést fűzhetünk a létrehozott változókhoz, mely általában a funkciójára utal.

Ha deklarálunk egy globális változót, akkor a programban a változóra a szimbolikus nevével hivatkozhatunk, nem szükséges a cím ismerete.

A globális változó táblázatot ugyan úgy lehet használni mint a korábban említett kezdeti érték táblázatot.

4.5. A keresztreferencia táblázat

Ez a táblázat mutatja az összes projektben használt változót, továbbá azt, hogy melyik POU-ban találhatók és azon belül melyik Network-ben. A Cross Reference táblázat hasznos eszköz ha szeretnénk tudni egy szimbólum nevet vagy címét, és hogy hol használjuk. A táblázat tartalma az első fordítás után generálódik és minden fordítás után frissül.

A következő ábrán látjuk a Cross Reference táblázatot:

Cross Reference

Index	Address	Symbol	POU	Position	Read/Write
0	%M1.3		MAIN	Network 0	Read
1	%M1.3		MAIN	Network 1	Read
2	%M1.4		MAIN	Network 0	Read
3	%M1.4		MAIN	Network 1	Read
4	%M10.0		MAIN	Network 0	Write
5	%M10.0		MAIN	Network 1	Write

Cross Reference

A keresztreferencia táblázatban, egy elemre duplán kattintva a program kapcsolódó része megjelenik.

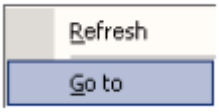
- Address: A változó címe
- Symbol: A változó neve, amennyiben a címhez tartozik szimbolikus név
- POU: a programegység, ahol a változóra hivatkozás történik
- Position: a hivatkozás helye a POU-n belül
- Read/Write: hozzáférés módja, vagyis az adott helyen a változót írja vagy olvassa-e a program.

4.5.1. A keresztreferencia táblázat megnyitása

- Válasszunk a [Project] → [Cross Reference] menüpontot

- Kattintsunk a  ikonra az eszköztárba.
- Alt+C billentyűkombinációval.

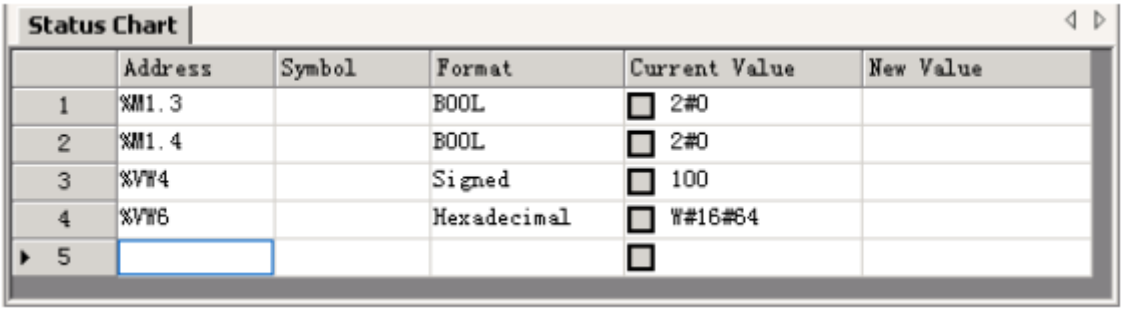
A megjelenő táblázatban a jobb egérgombbal kattintsunk bármelyik sorba és a következő menü fog megjelenni:



- Refresh: Frissíti a táblázatot.
- Go to: A program azon részére ugrik ahol a kijelölt elem található.

4.6. Változók állapota táblázat

A változó állapot táblázat segítségével nyomon követhetők a projektben használt változók aktuális állapotai, valamint megadhatók a kényszerített (Force) értékek is, miután letöltöttük a projektet a PLC-re.



	Address	Symbol	Format	Current Value	New Value
1	%M1.3		BOOL	2#0	
2	%M1.4		BOOL	2#0	
3	%VW4		Signed	100	
4	%VW6		Hexadecimal	W#16#64	
5					

- Address: Adjuk meg a monitorozni és kényszeríteni kívánt címet
- Symbol: A megadott címhez kapcsolódó szimbolikus név jelenik meg
- Format: Válasszuk ki az aktuális és az új felülbírálási érték megjelenítési formátumát (BOOL;REAL;Signed,Unsigned,Hexadecimal vagy Binary)
- Current value: Az éppen aktuális értéket mutatja
- New Value: Adjuk meg a kényszerített értéket, melyet a megadott változó felvesz, így a programból kapott érték felülírásra kerül.

A változóállapot táblázat csak abban az esetben mutatja az aktuális értékeket, ha a programozó környezetet Monitor módba kapcsoltuk, mely a [Debug] → [Monitor] paranccsal, vagy az eszköztárról érhető el.

A KincoBuilder program csak a projektben használt változókat tudja monitorozni. Ha egy olyan változót adunk meg a táblázatban amit nem használunk, a Current Value és a New Value értéke nem jelenik meg.

4.6.1. Státusz táblázat megnyitása

- Dupla kattintással a [Status Chart] szövegre a Manager ablakban.
- Jobb egérgombbal a [Status Chart] szövegre kattintva, majd az [Open] gombot választva.
- Válasszuk a [Debug] → [Status Chart] menüpontot

4.7 Jelszavas védelem

A KINCO-K3 PLC lehetőséget biztosít jelszavas védelem beállítására, ily módon védhető a központi egységben levő alkalmazás. A jelszó csak az aktuális műveletre érvényes. Amennyiben újra jelszóval védett funkciót szeretnénk elérni, ismét meg kell adnunk a jelszót. A jelszó, betűk, számok és aláhúzás karakterek kombinációi lehetnek és érzékeny a kis és nagy betűkre. Maximum 8 karakter hosszú lehet.

4.7.1. Jelszavas védelem szintjei

A KINCO-K3 PLC 3 szintű védelmet biztosít:

- Level 1: Teljes hozzáférés, nincs korlátozás az elérhető funkciókban
- Level 2: Részleges hozzáférés, jelszó szükséges a program letöltéshez
- Level 3: Minimális hozzáférés, jelszó szükséges a program le és feltöltéshez is

4.7.2. Jelszó és védelmi szint módosítása

Válasszuk a [PLC] → [Password...] menüt a "Password" ablak megnyitásához:

- Old password (régi jelszó megadása)
  Amennyiben a központi egységen már egyszer be volt állítva a jelszavas védelem, itt adható meg a szükséges jelszó, ha nem volt beállítva akkor üresen kell hagyni a mezőt.

- New Privileges (új védelmi szint)
  Új védelmi szintet és jelszót rendelhetünk a központi egységhez
  - Védelmi szint választható: level 1, level 2 vagy level 3.
  - New password (új jelszó): Adja meg az új jelszót
  - Confirm (jelszó megerősítése): Új jelszó ismételt megadása

A beállítások befejezését követően, kattintsunk az [Apply] gombra, így az elvégzett beállítások a központi egységbe mentésre kerülnek.

4.7.3. Az elfelejtett jelszó visszanyerése

Ha elfelejtjük a jelszót, törölnünk kell a CPU memóriáját. Válasszuk a [PLC] → [Clear...] menüt a CPU memóriájának törléséhez.
Törlés után az összes adat ami a CPU-n volt, beleértve a felhasználói programot, a konfigurációs adatokat, és a jelszót, törlődni fog. A CPU a gyári állapotba áll vissza kivéve a valós idejű órát.
Alapértelmezett kommunikációs paraméterek: station number 1, baudrate 9600, no parity, 8 data bits, 1 stop bit.

5. Programozás a KincoBuilder programmal

A KincoBuilder program az IL (utasítás lista) és az LD (létra-diagram) programozási nyelveket támogatja. Ez a fejezet részletesen leírást ad erről a két programozási nyelvről, bemutatja a fontos szintaktikáit és szabályait az IL és LD nyelveknek.

A KincoBuilder program a két programozási nyelvnek megfelelően két szerkesztőt kínál: az IL szerkesztőt és az LD szerkesztőt. Bizonyos korlátozásokkal a megírt POU-k átváltása lehetséges a másik nyelvre- Válasszuk a [Project] → [IL] vagy [Project] → [LD] menüt a programozási nyelv átkapcsolásához.

5.1. Utasítás lista (IL) programozás

5.1.1. Áttekintés

Az IL egy alacsony szintű programozási nyelv, ami nagyon hasonlít az assembly nyelvhez, jellemzője az egyszerű utasítás lista, melyet számos PLC gyártó alkalmaz világszerte.
Az IL hasonlít a gépi kódra, és nagyon gyakorlatias nyelv, így az IL a gyakorlott programozóknak hasznos.

5.1.2. Szabályok

5.1.2.1. Utasítások

Az IL program utasítások sorozatát tartalmazza. Minden utasításnak új sorban kell kezdődnie, és egy utasítást kell tartalmaznia. Az operandusok opcionálisak, és szóközzel vagy vesszővel kell elválasztani őket. A kommenteket a sor végén helyezhetjük el csillagozva és zárójelben. Üres sorokat is elhelyezhetünk az utasítás listában a könnyebb áttekinthetőség érdekében.

A következő ábra bemutatja az IL nyelv formátumát:

- label (címke)
  Opcionális. Ugrást arra használhatjuk, hogy az IL program egyik sorára ugorjunk.
- Operator (függvény, utasítás)
- Operands (adat)
- Comment (megjegyzés)
  Opcionális. Csak egy komment alkalmazható egy sorban.

Példa:

```
(* NETWORK 0 *)
begin: (* címke, programon belüli ugrásra használható *)
LD %I1.0
TP T2, 168 (* ha %I1.0 igaz, T2 időzítő elindul. T2 egy TP típusú változó *)
```

5.1.2.2. Aktuális eredményt tartalmazó változó

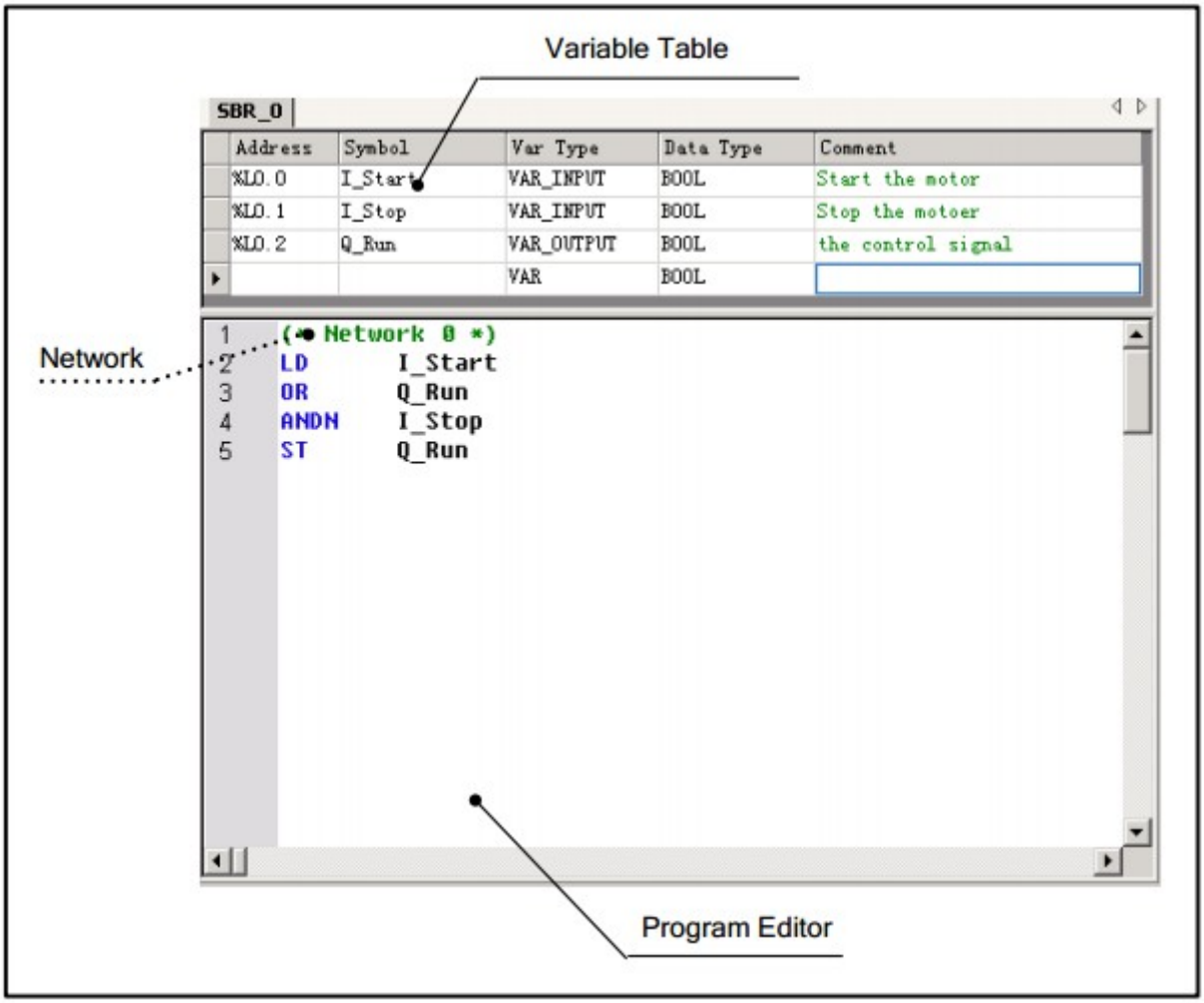
Az IL programban alkalmazható egy "Current Result (CR)" univerzális akkumulátor regiszter. Az elvégzett logikai művelet eredményét a program a CR-ben tárolja.
A KincoBuilder programban az operátorok csoportosíthatók aszerint, hogy milyen hatással vannak a CR regiszterre.

Csoport	Hatása CR-re	Példa
C	CR létrehozása	LD, LDN
P	Művelet eredménye a CR-be kerülhet	Bites műveletek, komparálás, stb.
U	CR-t változatlanul hagyja	ST, R, S, JMP stb.

IEC61131-3 szabványban nincsenek definiálva a fenti csoportok, ezért ez a rész eltérhet más programozói környezetektől.

5.1.3. IL szerkesztő a KincoBuilderben

Az IL szerkesztő felépítése a következő ábrán látható:



Az IL szerkesztő két részre bontható:

- A változó táblázat (Variable Table): deklarálhatjuk a helyi változókat és a POU be és kimeneti paramétereit.
- A program szerkesztő (Program Editor): a vezérlőprogram helye

5.1.3.1. Új programrész (network) hozzáadása

Használja valamelyik megoldást a következők közül:

- Ctrl+Q billentyűzetkombináció
- Jobb egérgombbal kattintsunk a program szerkesztőbe és válasszuk az [Insert Network] opciót a felugró menüben

- Egy programrészben csak egy címke (label) használható

```
(* NETWORK 0 *)

MRun: (* címke megadása*) Csak egy címkét lehet megadni!
```

- Minden programrésznek „C” csoportos utasítással kell kezdődnie, a vége pedig „P” vagy „U” csoportos utasítás lehet.

```
(* NETWORK 0 *)

LD %M3.5 (*program kezdetén LD utasítás, „C” utasítás csoport  *)
… …        (*további utasítások *)
ST %Q2.3  (*programrész vége, „U” utasítás csoport*)
```

- Előző programrész kiegészítése, címkével

```
(* NETWORK 0 *)

MRun: (* címke megadása*)
LD %M3.5 (*program kezdetén LD utasítás, „C” utasítás csoport  *)
… …        (*további utasítások *)
ST %Q2.3  (*programrész vége, „U” utasítás csoport*)
```

Az IL szerkesztő automatikusan megformázza a bevitt utasításokat, és ellenőrzi a helyességüket. Amennyiben egy utasítás nem megfelelő, a sor elején piros kérdőjel (?) jelenik meg.

5.1.3.2. IL nyelvű mintaprogram

```
(* NETWORK 0 *)
LDN %M0.0
TON T0, 1000       (*T0 indítása T1 kimenetével, időzítés: 1000*1ms *)
ST %M0.1
LD %M0.1
TON T1, 1000       (*T1 indítása T0 kimenetével, időzítés: 1000*1ms *)
ST %M0.0
LD %M0.1
ST %Q0.0           (* Négyszögjel 2s periódusidővel, %Q0.0 kimenetre *)
```

5.1.3.3. Online monitor mód

Miután a [Debug] → [Monitoring] menüt kiválasztottuk, az IL szerkesztő átvált monitor üzemmódba, melyben a program szerkesztése nem lehetséges.
Monitor módban a programszerkesztő nézet két részre oszlik, a jobb oszlopban látható a program a bal oldalon pedig a programban szereplő változók láthatók.

5.1.4. IL program konvertálása LD programra

Válasszuk a [Project] → [LD] menüt, hogy az IL nyelven készített programot LD nyelvre alakítsa át a program.
Nem minden IL programot lehet LD-re konvertálni; a sikeres művelethez a következő feltételeknek kell teljesülniük:

1. Nem lehet hiba az IL programban.
2. Az IL program forrásának szigorúan meg kell felelni a következő szabályoknak:
   - Minden programrésznek (network-nek) "C" csoportba tartozó utasítással kell kezdődnie; vagy csak egy címke lehet a network-ben.
   - Az az utasítás amivel a programrész kezdődik csak egyszer használható a network-ben.
   - Minden programrészt "P" vagy "U" csoportba való utasítással kell befejezni.

5.2. LD programozás
5.2.1. Áttekintés

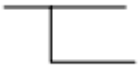
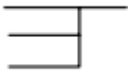
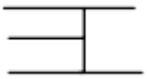
A LD (létra diagram) az egyik legkedveltebb grafikai programozási nyelv a PLC programozásban. LD programozási nyelv alapja a tradicionális relés logika, de mellette az IEC LD nyelv lehetőséget biztosít a felhasználó által definiált funkció blokk és funkciók használatára.

A következő egyszerű program LD programnyelvben készült.

Az LD programozás során szabványos grafikai elemekkel készíthetjük el a vezérlőprogramot. Az LD programozás során a programot a képernyőn levő két vezetősín közé kell elkészíteni. A bal oldali sín folyamatosan bekapcsolt állapotúnak tekinthető. Ha egy programsorban megadott feltételek teljesülnek, a kapcsolat létrejön a bal és jobboldali sín között.

5.2.3. Szabványosított grafikai szimbólumok

1) Összeköttetések (link)
   Vízszintes és függőleges összeköttetések valósíthatóak meg az LD programozás során, melyekkel soros vagy párhuzamos összeköttetések valósíthatók meg. Az összekötő elemek logikai 0 vagy 1 állapotot vehetnek fel.

Szimbólum	Név	Leírás
	Vízszintes összekötő	A vízszintes összekötő elem egy vonalként jelenik meg a szerkesztőben. A bal oldalán található elem állapotát átviszi a jobb oldalt található elemhez.
	Függőleges összeköttetések	A függőleges összeköttetés jobb oldala a következő állapotokat veheti fel • Kikapcsolt (OFF): ha a bal oldalt levő összes bemenet állapota kikapcsolt • Bekapcsolt (ON): ha a bal oldalt levő egy vagy több bemenet állapota bekapcsolt Amennyiben a függőleges összeköttetés több kimenettel rendelkezik a jobb oldalon, az összes kimenet azonos értéket vesz fel.
		
		

2) Kontaktus (contact)

A kontaktus az összeköttetések bal oldalán alkalmazható, a hozzárendelt kétállapotú változó állapotát viszi be a programba. A hozzárendelt változó értékén nem változtat.

Szimbólum	Név	Leírás
	Alap esetben nyitott kontaktus (NO)	A kontaktushoz rendelt kimenet bekapcsol, ha a bemeneti változó („***” jelölve) értéke logikai 1 szintet vesz fel. Ellenkező esetben a jobb oldali kimenet kikapcsolt állapotban marad.
	Alap esetben zárt kontaktus (NC)	A kontaktushoz rendelt kimenet bekapcsol, ha a bemeneti változó („***” jelölve) értéke logikai 0 szintet vesz fel. Ellenkező esetben a jobb oldali kimenet bekapcsolt állapotban marad.

3. Kimenetek (coil)

A kimenet a bal oldalt található bemeneti változó értékét rendeli egy megadott változóhoz.

Szimbólum	Név	Leírás
*** —( )—	Kimenet	A bal oldali bemenet állapotát rendeli egy megadott („***”) változóhoz.
*** —(/)—	Negált kimenet	A bal oldali bemenet negált állapotát rendeli egy megadott („***”) változóhoz. Például ha a bemenet kikapcsolt, akkor a kimenet bekapcsolt lesz.
*** —(S)—	Set kimenet	A hozzá kapcsolt boolean változó ON állapotba kapcsolja, amikor a bal oldalra ON állapot kerül és mindaddig tartja amíg a RESET kimenet nem törli.
*** —(R)—	Reset kimenet	A hozzá kapcsolt boolean változó OFF állapotba kapcsolja, amikor a bal oldali összekötőre ON állapot kerül és mindaddig tartja amíg a SET coil nem állítja vissza.

4. Programvezérlők

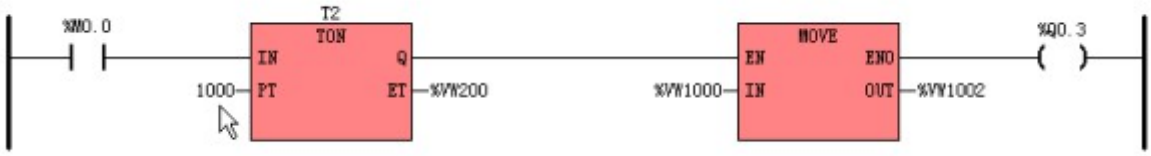
A programvezérlők a program utasítás végrehajtási sorrendjét módosíthatják.

Szimbólum	Név	Leírás
├─ (1) ──<RETURN>	Feltételes visszatérés	Megszakítási rutinban vagy alprogramban használható. A program végrehajtása visszaugrik a főprogramba, és onnan folytatja futtatást, ha a kifejezés bal oldala 1 (Igaz), hamis bemeneti feltételek esetén a következő utasítás kerül végrehajtásra ugrás nélkül.
├─>> Label	Feltétel nélküli ugrás	A program futása a megjelölt címke után folytatódik.
├─ (1) ─>> Label	Feltételes ugrás	A program futása a megjelölt címke után folytatódik, ha a bal oldali kifejezés igaz. Ellenkező esetben ugrás nélkül a soron következő utasítás kerül végrehajtásra.

5. Funkciók és funkció blokkok

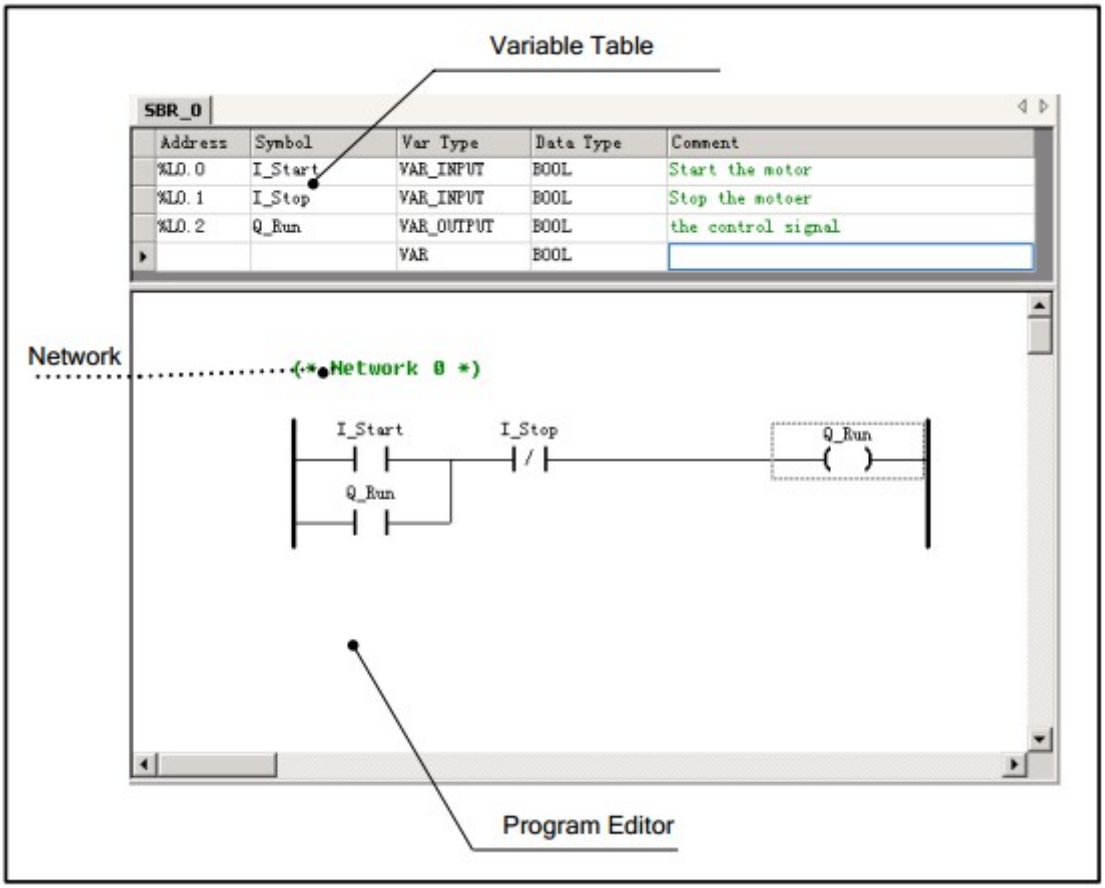
A funkciók vagy a funkció blokkok a programozás során négyzet formájában jelennek meg. A négyzet külsejéhez csatlakoztathatók a kapcsolódó változók, a csatlakozás elnevezése pedig a négyzet belsejében található. A blokkoknak legalább egy logikai be és kimenetük kell, hogy legyen, hogy a bal és jobb vezetősín közötti kapcsolat biztosítható legyen.

A funkciók az EN logikai bemenettel és ENO logikai kimenettel rendelkeznek. Az EN bemenet a funkció végrehajtását engedélyezi, ha EN = 1 akkor ENO is 1 értéket vesz fel, és a funkció végrehajtásra kerül. Amennyiben EN = 0 a funkció nem kerül végrehajtásra és ENO is 0 értéket vesz fel.



5.2.4. LD programszerkesztő

Ha létrehozunk vagy megnyitunk egy LD programot, akkor megnyílik a hozzá kapcsolódó programszerkesztő is.
Az LD szerkesztő felépítése a következő ábrán látható.



5.2.4.1. LD program korlátok

Maximálisan 200 programrész (Network) engedélyezett az LD programban.
Egy programrészben vízszintesen 32, függőlegesen pedig 16 elem helyezhető el. Amennyiben csak kontaktusok szerepelnek a programrészben, akkor maximálisan 31 kontaktus és 1 kimenet helyezhető el. Ha a programrész csak funkciókból vagy funkcióblokkokból áll, akkor 12 blokk mellett 1 kontaktus és 1 kimenet helyezhető el. Egy programrészben maximálisan 16 párhuzamos elágazás helyezhető el.
Párhuzamos elágazás kettő vagy több független funkció/funkció blokk között tilos.

5.2.4.3. LD programozás lépései

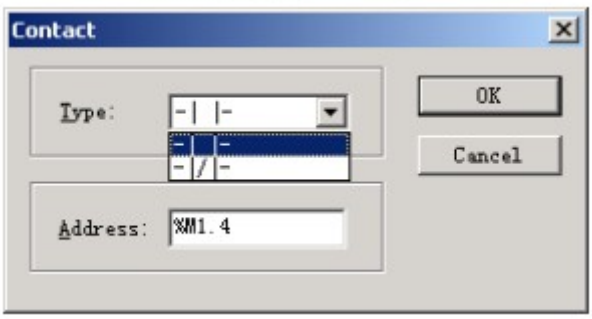
A következő leírás főként az egérrel elvégezhető műveletekkel foglalkozik.

1. A következő módszerekkel adhatunk hozzá új programrészt
   - Válasszuk az [LD] → [Network] menüt
   - Kattintsunk a  ikonra az eszköztáron
   - Ctrl+W
   - Jobb egérgombbal kattintsunk valamelyik elemre és a felugró ablakban válasszuk a [Network] menüt

Az új hozzáadott programrész a következőképpen jelenhet meg:

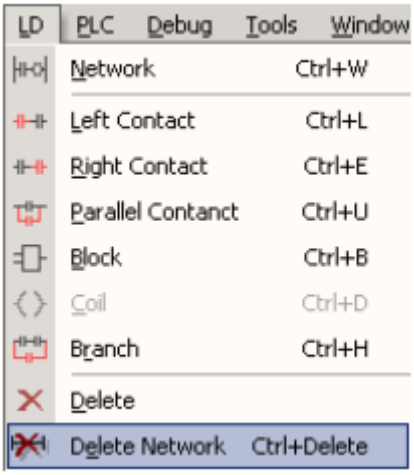
Dupla kattintással a programrész címkéjére (nevére) megnyitja a megjegyzés párbeszéd ablakot, és itt írhatunk kommentet vagy leírást a network-höz.

2) Amikor egy új utasítást adunk hozzá, annak a változója piros kérdőjelekként (????) jelenik meg, ezek a kérdőjelek azt jelzik, hogy nincs hozzárendelve változó, és hogy a deklarálást el kell végeznünk mielőtt lefordítjuk a programot.
Ha rákattintunk a változóra a felugró párbeszéd ablakban kiválaszthatjuk a változó típusát és beírhatjuk a címét. ENTER megnyomásával is elérhetjük ezt a párbeszédablakot.



3) Kattintsunk egy elemre és jelöljük ki mint a fenti példában, majd adjunk, hozzá egy másik elemet a következő módszerek valamelyikével:

- Használjuk az [LD] menüt vagy gyors-gombot:



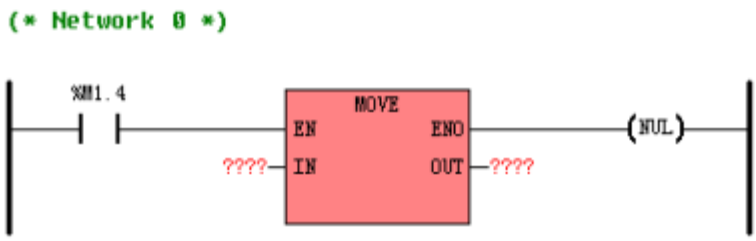
Left Contact: hozzáadunk egy kontaktust a kiválasztott elem bal oldalához
Right Contact: hozzáadunk egy kontaktust a kiválasztott elem jobb oldalához
Parallel Contact: hozzáadunk egy kontaktust a kiválasztott elemmel párhuzamosan.
Block: Funkciót, funkció blokkot vagy szubrutin adható hozzá
Coil: hozzáadunk egy kimenetet a kijelölt kimenettel párhuzamosan
Branch: Elágazást rajzolunk a kiválasztott elemmel párhuzamosan.
Delete: Töröljük a kiválasztott elemet.
Delete Network: Töröljük azt a programrészt, amelyben az elem ki van jelölve.

- A fenti funkciók elérhetők az eszköztáron is

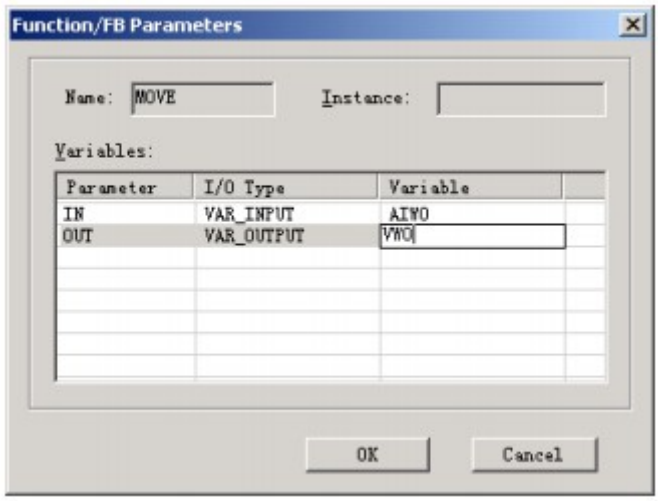
Kattintsunk a megfelelő gombra az eszköztáron, hogy kiválasszuk a helyes elemet.

- Elem elhelyezhető az Instructions fülön a megfelelő blokkra duplán kattintva.

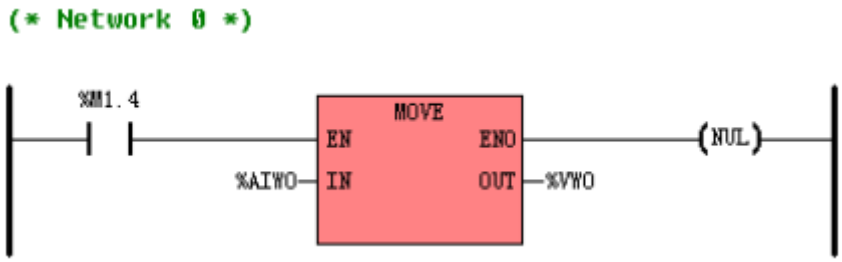
Egy "MOVE" blokkot hozzáadva a programrész a következőképpen jelenik meg.



4) Dupla kattintással egy blokkra a programban elérhetjük a blokk paraméter beállítására szolgáló párbeszédablakot.



Dupla kattintással bármelyik változóra a [Variable] oszlopba megadhatjuk a kívánt változót majd az Enter billentyűvel elmenthetjük a beállítást. KincoBuilder program ellenőrzi a változók típusát, és ha hibás visszautasítja azt.
A fenti képernyőn az OK gombra kattintva a módosított blokk a következőképpen jelenik meg:



5) Miután a fenti programrész elkészült, adhatunk hozzá új programrészt. Ha új programrészt ad hozzá, és az aktuális programrész címkéje van kijelölve, akkor az új network a kijelölt rész elé kerül, ellenkező esetben utána.

5.2.4.4. Online monitor mód

A [Debug] → [Monitor] menüpont választásakor az LD szerkesztő monitor módba kapcsol. Monitor üzemmódban a program adatai láthatók, a program szerkesztése monitor módban ez nem lehetséges.

6. KINCO-K3 Utasítás készlet

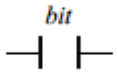
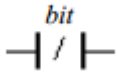
A KINCO-K3 utasítás készletének nagy része az IEC61131-3 szabvány szerint készült, az alap utasítások és funkcióblokkok a programban elérhetők. Emellett a programozói környezet nem hagyományos elemeket is tartalmaz, a felhasználói igényekhez igazodva.

6.1. Összefoglaló

Ebben a fejezetben részletesen bemutatjuk az utasításokat és speciális alkalmazási példákat.
Az utasítások IL és LD programnyelven is bemutatásra kerülnek.
LD programozási nyelv esetében, az EN és ENO operandusok nem kerülnek külön leírásra, mert funkciójuk az összes utasításnál megegyezik. EN engedélyező bemenetre egy logikai értéket (BOOL) kell csatlakoztatni, ezzel engedélyezhető a blokk működése. Az ENO kimenet bekapcsol, ha EN bemenetet engedélyezzük, segítségével a blokk logikai 1 szintje tovább vihető a tőle jobbra található elemek számára.

6.2 Bites logikai utasítások

6.2.1. Kontaktusok (digitális bemenetek)

	Név	Példa	Csoport	
LD	Alapesetben nyitott			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	Alapesetben zárt			
IL	LD	LD bit	C	
	LDN	LDN bit		
	AND	AND bit	P	
	OR	OR bit		
	ANDN	AND bit		
	ORN	ORN bit		

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
bit	Bemenet	BOOL	I, Q, V, M, SM, L, T, C, RS, SR,const.

- **LD**

  Amikor a hozzárendelt bit egyenlő 1-el, az alaphelyzetben nyitott kapcsoló bezár (on) és aktiválja a tőle jobbra levő elemet.

  Amikor a bit egyenlő 0-val, az alaphelyzetben zárt kapcsoló zárva marad (on) és a tőle jobbra levő elem aktiválódik.

- **IL**

Az alap helyzetben nyitott kapcsoló megjelenhet a programban mint LD, AND vagy OR utasítás.

Az LD utasítás betölti a bitet és a CR-t egyenlővé teszi vele.

Az AND utasítás ÉS kapcsolatba hozza a bitet és a CR-t, az eredmény CR-ben képződik

Az OR utasítás, VAGY kapcsolatba hozza a bitet és a CR-t, az eredmény CR-ben képződik

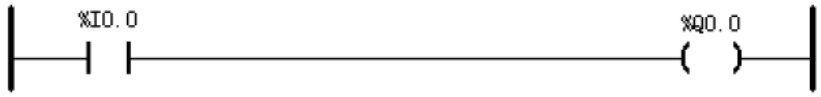
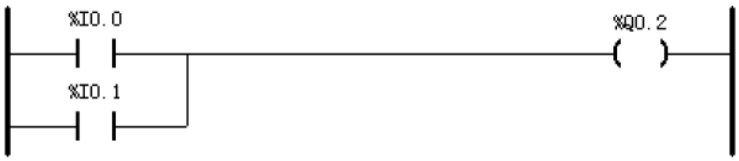
Az alap helyzetben zárt kapcsoló megjelenhet a programban mint LDN, ANDN vagy ORN utasítás.

Az LDN utasítás betölti a bit értékének negáltját és a CR-t egyenlővé teszi vele.

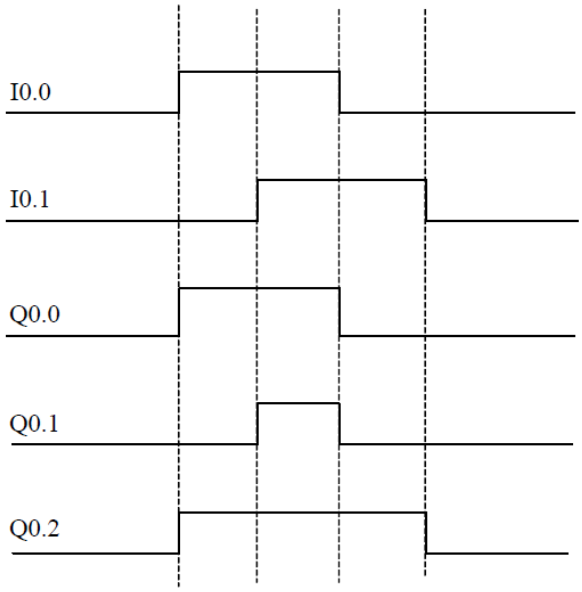
Az ANDN utasítás NEM-ÉS kapcsolatba hozza a bitet és a CR-t, az eredmény CR-ben képződik

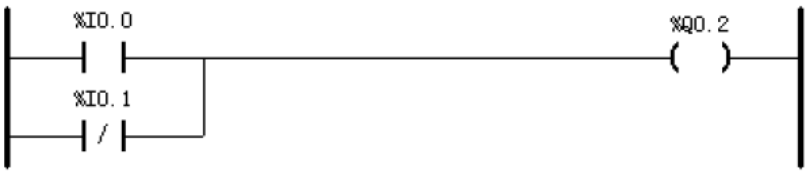
Az ORN utasítás, NEM-VAGY kapcsolatba hozza a bitet és a CR-t, az eredmény CR-ben képződik

- Példák

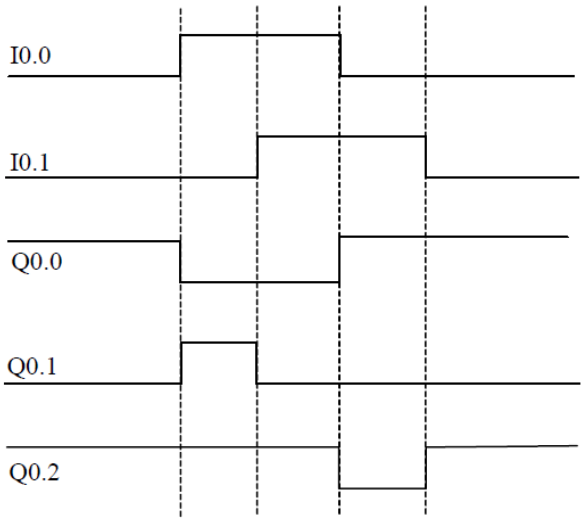
LD	IL
	LD %I0.0 ST %Q0.0
	LD %I0.0 AND %I0.1 ST %Q0.1
	LD %I0.0 OR %I0.1 ST %Q0.2

Az előző példában bemutatott három programrész működésének szemléltetése:



LD	IL
	LDN %I0.0 ST %Q0.0
	LD %I0.0 ANDN %I0.1 ST %Q0.1
	LD %I0.0 ORN %I0.1 ST %Q0.2

Az előző példában bemutatott három programrész működésének szemléltetése:



6.2.2. Azonnali kontaktusok (digitális bemenetek)

	Név	Használat	Csoport	
LD	Alapesetben nyitott	*bit* I		☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
	Alapesetben zárt	*bit* /I		
IL	LDI	LDI bit	C	
	LDNI	LDNI bit		
	ANDI	ANDI bit	P	
	ORI	ORI bit		
	ANDNI	ANDNI bit		
	ORNI	ORNI bit		

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
bit	Bemenet	BOOL	I (CPU)

Az azonnali bemeneti kontaktus, a program végrehajtása során a fizikai bemenet értékét a végrehajtás pillanatában megkapja, bemeneti memória területet állapotától függetlenül. Csak a központi egység digitális bemeneteire használható, állapotára a bemeneti szűrők beállításai hatástalanok. Gyors jelek fogadására használható, mivel a bemenet állapota nem a ciklus kezdetekor kerül beolvasásra, hanem a végrehajtás pillanatában.

- **LD**

  Amikor a hozzárendelt bit egyenlő 1-el, az alaphelyzetben nyitott kapcsoló bezár (on) és a tőle jobbra levő elem aktiválódik.

  Amikor a bit egyenlő 0-val, az alaphelyzetben zárt kapcsoló zárva marad (on) és a tőle jobbra levő elem aktiválódik.

- **IL**

Az alap helyzetben nyitott kapcsoló megjelenhet a programban mint LDI, ANDI vagy ORI utasítás.

Az LDI utasítás betölti a bitet és a CR-t egyenlővé teszi vele.

Az ANDI utasítás ÉS kapcsolatba hozza a bitet és a CR-t, az eredmény CR-ben képződik

Az ORI utasítás, VAGY kapcsolatba hozza a bitet és a CR-t, az eredmény CR-ben képződik

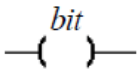
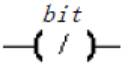
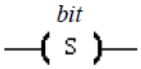
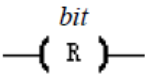
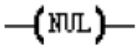
Az alap helyzetben zárt kapcsoló megjelenhet a programban mint LDNI, ANDNI vagy ORNI utasítás.

Az LDNI utasítás betölti a bit értékének negáltját és a CR-t egyenlővé teszi vele.

Az ANDNI utasítás NEM-ÉS kapcsolatba hozza a bitet és a CR-t, az eredmény CR-ben képződik

Az ORNI utasítás, NEM-VAGY kapcsolatba hozza a bitet és a CR-t, az eredmény CR-ben képződik

6.2.3. Digitális kimenetek

	Név	Használat	Csoport	
LD	Kimenet			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	Negált kimenet			
	Set Kimenet			
	Reset kimenet			
	Üres kimenet			
IL	ST	ST *bit*	U	
	STN	STN *bit*		
	R	R *bit*		
	S	S *bit*		

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
bit	Kimenet	BOOL	Q, V, M, SM, L

- **LD**

  A **kimeneti bit** a tőle balra levő programsor állapotát elmenti a kimeneti memóriába

  A **negált kimeneti bit** a tőle balra levő programsor állapotának inverzét elmenti a kimeneti memóriába.

  Ha a programsor értéke 1, akkor a kimeneti memória értéke is 1 lesz **set kimenet** használata esetén. Ha a programsor értéke 0, akkor a kimenet változatlan marad, vagyis ha előtte be volt kapcsolva akkor állapotát megőrzi.

  Ha a programsor értéke 1, akkor a **reset kimenet** a hivatkozott bit állapotát törli, ha a programsor értéke 0, akkor a kimenet értéke változatlan marad.
  Az **üres kimenet** a programrész végének a jelzésére szolgál, funkciója nincsen.

- **IL**

  A digitális kimenet megjelenhet a programban, mint ST, STN, R vagy S utasítás.

  Az **ST** utasítás a CR állapotát eltárolja a kimeneti memóriába.

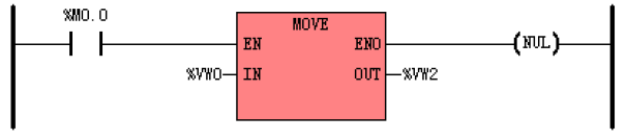
  Az **STN** utasítás a CR állapotának negáltját eltárolja a kimeneti memóriába.

  Az **R** utasítás funkciója: kimeneti memória értéke is 0 lesz, ha CR értéke 1, ellenkező esetben a kimeneti memória értéke nem változik.

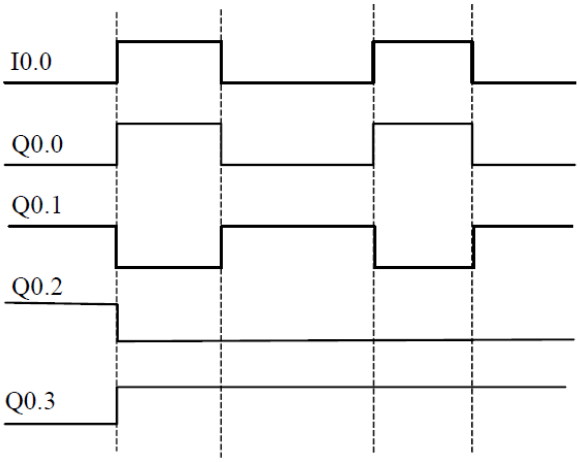
  Az **S** utasítás funkciója: kimeneti memória értéke is 1 lesz, ha CR értéke 1, ellenkező esetben a kimeneti memória értéke nem változik.

  ST, STN, R és S utasítások nem befolyásolják CR állapotát.

Alkalmazási példa

LD	IL
	LD %I0.0 ST %Q0.0 STN %Q0.1 R %Q0.2 S %Q0.3
	LD %M0.0 MOVE %VW0, %VW2

Az előző példában bemutatott programrészek működésének szemléltetése:
A mintaprogram lefutása előtt Q0.1 és Q0.2 magas szinten volt.



6.2.4. Azonnali digitális kimenetek

	Név	Használat	Csoport	
LD	Azonnali kimenet	*bit* —(I)—		☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
	Azonnali Set kimenet	*bit* —(SI)—		
	Azonnali Reset kimenet	*bit* —(RI)—		
IL	STI	STI *bit*	U	
	RI	RI *bit*		
	SI	SI *bit*		

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
bit	Kimenet	BOOL	Q (CPU)

Azonnali digitális kimenetek csak a központi egység kimeneti pontjain használhatók.

- **LD**

  Azonnali digitális kimenet állapota a végrehajtásakor megjelenik a fizikai kimeneten és a kapcsolódó kimeneti memória címen is.

  Azonnali set kimenet esetén, ha a programsor értéke 1, akkor a végrehajtást követően a fizikai kimenet és a kimeneti memória értéke is 1 lesz. Ha a programsor értéke 0, akkor a kimenet változatlan marad, vagyis ha előtte be volt kapcsolva akkor állapotát megőrzi.

  Azonnali reset kimenet esetén, ha a programsor értéke 1, akkor a végrehajtást követően a fizikai kimenet és a kimeneti memória értéke 0 lesz. Ha a programsor értéke 0, akkor a kimenet változatlan marad.

- **IL**

  Az azonnali digitális kimenet megjelenhet a programban, mint STI, RI vagy SI utasítás.

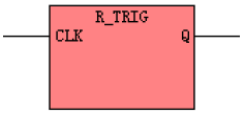
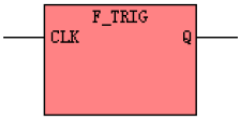
  Az **STI** utasítás azonnal eltárolja CR állapotát a kimeneti memóriába. Az állapot azonnal megjelenik a fizikai kimeneten is.

  Az **RI** utasítás funkciója: kimeneti memória és a fizikai kimenet értéke is 0 lesz, ha CR értéke 1, ellenkező esetben az értékek nem változnak.

  Az **SI** utasítás funkciója: kimeneti memória és a fizikai kimenet értéke is 1 lesz, ha CR értéke 1, ellenkező esetben az értékek nem változnak.

Az STI, RI és SI utasítások nem befolyásolják CR állapotát.

6.2.5. Felfutó és lefutó él figyelés

	Név	Használat	Csoport	
LD	Felfutó él detektálás			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	Lefutó él detektálás			
IL	R_TRIG	R_TRIG	P	
	F_TRIG	F_TRIG		

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
CLK (LD)	Bemenet	BOOL	Program összeköttetés
Q (LD)	Kimenet	BOOL	Program összeköttetés

- **LD**

  Az R_TRIG utasítás a CLK bemenetén érzékeli a felfutóél váltást. Vagyis ha a CLK bemeneten 0 → 1 váltás történik, Q kimenet egy ciklusidőre bekapcsol, majd azt követően kikapcsol.

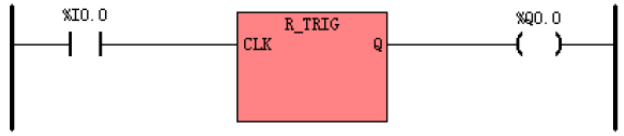
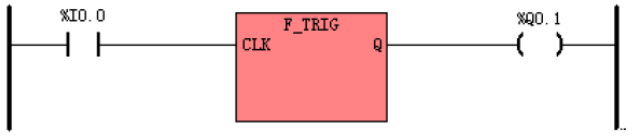
  Az F_TRIG utasítás a CLK bemenetén érzékeli a lefutóél váltást. Vagyis ha a CLK bemeneten 1 → 0 váltás történik, Q kimenet egy ciklusidőre bekapcsol, majd azt követően kikapcsol.

- **IL**

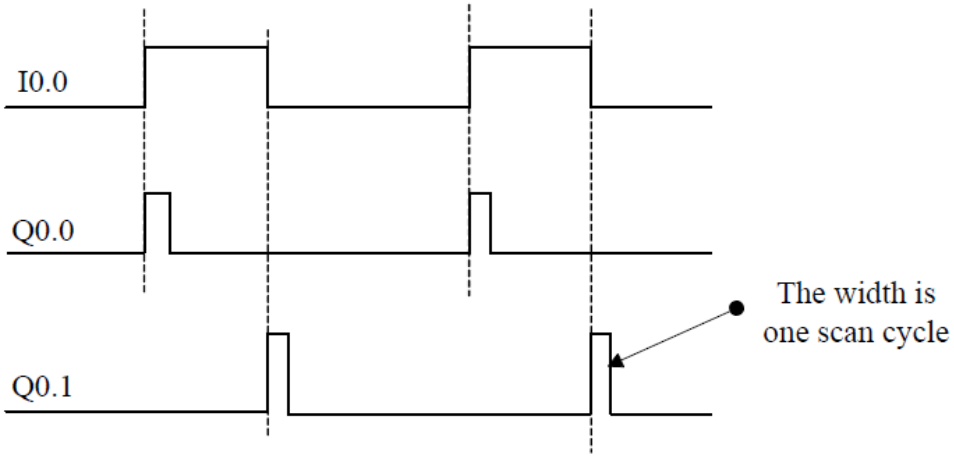
  Az R_TRIG utasítás a CLK bemenetén érzékeli a felfutóél váltást. Vagyis ha a CLK bemeneten 0 → 1 váltás történik, Q kimenet egy ciklusidőre bekapcsol, majd azt követően kikapcsol.

  Az F_TRIG utasítás a CLK bemenetén érzékeli a lefutóél váltást. Vagyis ha a CLK bemeneten 1 → 0 váltás történik, Q kimenet egy ciklusidőre bekapcsol, majd azt követően kikapcsol.

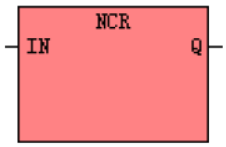
➢ Példa

LD	IL
	LD %I0.0 R_TRIG ST %Q0.0
	LD %I0.0 F_TRIG ST %Q0.1

Az előző példában bemutatott programrészek működésének szemléltetése:

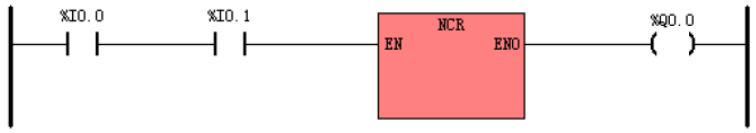


6.2.6. NCR (negálás)

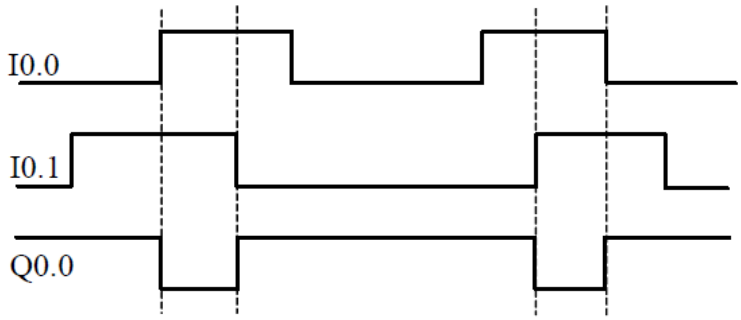
	Név	Használat	Csoport	
LD	NCR			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	NCR	NCR	P	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
CLK	Bemenet	BOOL	Program összeköttetés
Q	Kimenet	BOOL	Program összeköttetés

- **LD, IL**
  Az NCR utasítás az összekötés állapotát ellentettjére változtatja, ha 1 volt 0 lesz, ha 0 volt 1 lesz. Az első lefutó élig tartja a változtatott állapotot.

LD	IL
	LD %I0.0 AND %I0.1 NCR ST %Q0.0

Az előző példában bemutatott programrész működésének szemléltetése:



6.2.6. Kétállapotú elemek

A bistabil elemek az IEC61131-3 szabványban definiált funkció blokkok, két típus érhető el, a Set Dominant Bistable (SR), ahol a SET bemenet élvez elsőbbséget, és a Reset Dominant Bistable (RS), ahol pedig a RESET bemenetnek van elsőbbsége.

6.2.7.1. SR Set domináns bistabil utasítás

	Név	Használat	Csoport	
LD	SR			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	SR	LD S1 SR SRx, R	P	

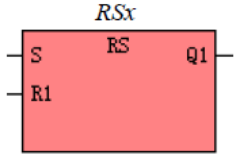
Paraméter	Bemenet/Kimenet	Adat típus	Használható memória terület
SRx	-	SR	SR
S1	Bemenet	BOOL	Program összeköttetés
R	Bemenet	BOOL	I, Q, V, M, SM, L, T, C, RS, SR
Q1	Kimenet	BOOL	Program összeköttetés

Az SR egy bistabil elem ahol a set bemenet dominál. Ha a set és reset bemenet egyidejűleg 1, akkor a Q1 kimenet és az Srx értéke is 1 lesz.

SR utasítás igazságtáblája:

S1	R	Q1, SRx
0	0	Előző érték
0	1	0
1	0	1
1	1	1

6.2.7.2. RS Reset domináns bistabil utasítás

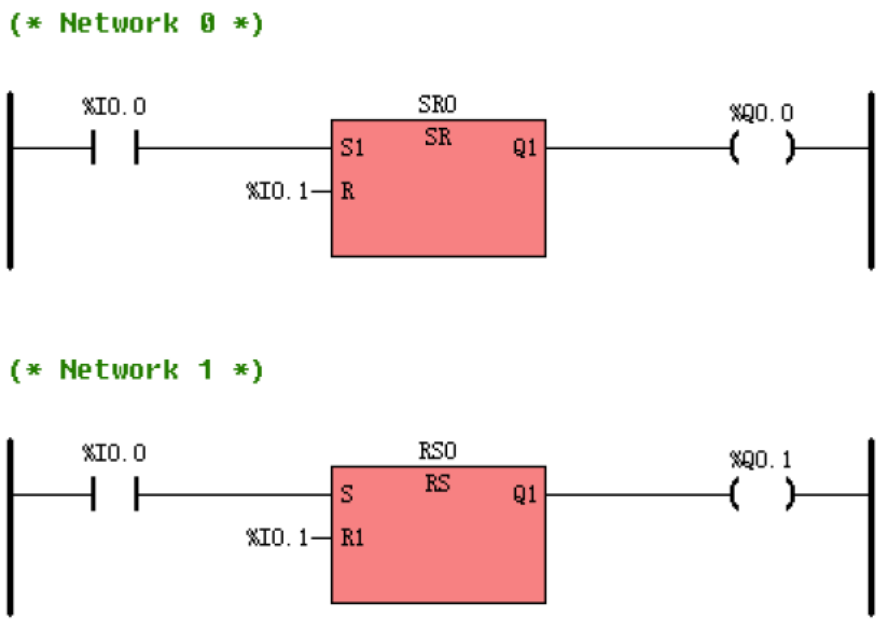
	Név	Használat	Csoport	
LD	RS			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	RS	LD *S* *RS RSx, R1*	P	

Paraméter	Bemenet/Kimenet	Adat típus	Használható memória terület
RSx	-	RS	RS
S	Bemenet	BOOL	Program összeköttetés
R1	Bemenet	BOOL	I, Q, V, M, SM, L, T, C, RS, SR
Q1	Kimenet	BOOL	Program összeköttetés

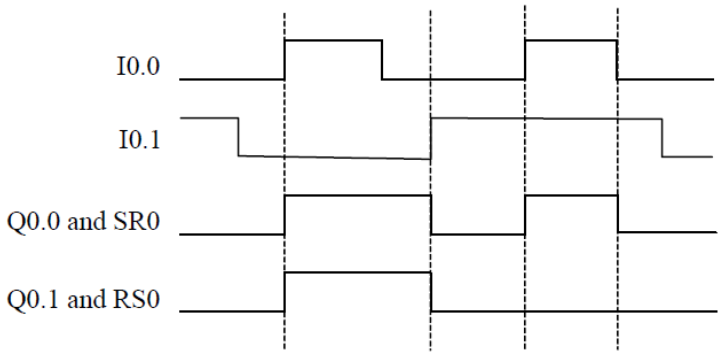
Az RS egy bistabil elem ahol a Reset bemenet dominál. Ha a set és reset bemenet egyidejűleg 1, akkor a Q1 kimenet és az Srx értéke is 0 lesz.

RS utasítás igazság táblája:

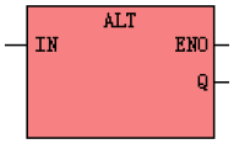
R1	S	Q1, SRx
0	0	Előző érték
0	1	1
1	0	0
1	1	0

LD	IL
	(* Network 0 *) LD %I0.0 SR SR0, %I0.1 ST %Q0.0 (* Network 1 *) LD %I0.0 RS RS0, %I0.1 ST %Q0.1

A programrész működésének szemléltetése:



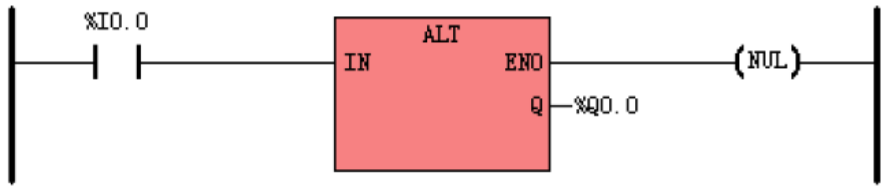
6.2.8. Kimenet állapot váltás (ALT)

	Név	Használat	Csoport	
LD	ALT			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	ALT	ALT Q	U	

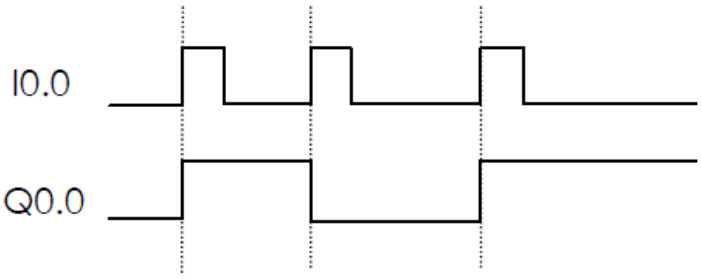
Paraméter	Bemenet/Kimenet	Adat típus	Használható memória terület
IN (LD)	Bemenet	BOOL	Program összeköttetés
Q	Kimenet	BOOL	Q, V, M, SM, L

- **LD**
  Az ALT utasítás az IN bemenetre érkező jel felfutó él hatására a Q kimenet állapotát megváltoztatja és meg is tartja azt az állapotot tehát, ha 0 volt akkor 1 lesz, ha 1 volt akkor 0 lesz.

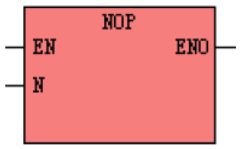
- **IL**
  Az ALT utasítás az IN bemenetre érkező jel felfutó él hatására a Q kimenet állapotát megváltoztatja, ha 0 volt akkor 1 lesz és meg is tartja azt az állapotot tehát, ha 1 volt akkor 0 lesz.

LD	IL
	LD %I0.0 ALT %Q0.0

A programrész működésének szemléltetése



6.2.9. NOP No Operation

	Név	Használat	Csoport	
LD	NOP			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	NOP	NOP *N*	U	

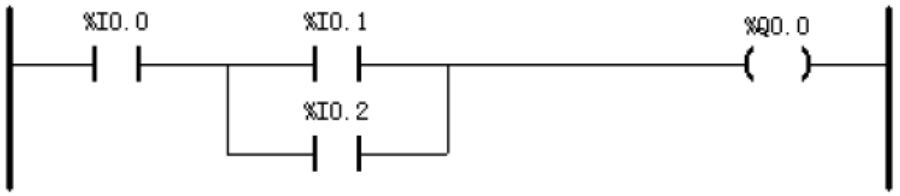
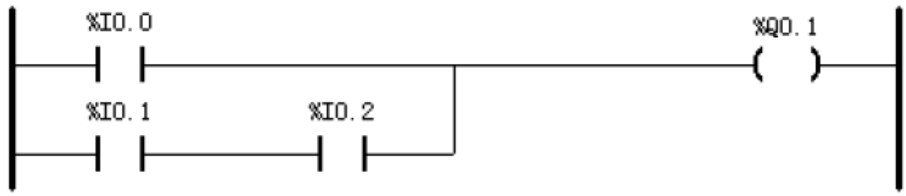
Paraméter	Bemenet/Kimenet	Adat típus	Használható memória terület
N	Bemenet	INT	Konstans (pozitív)

A NOP utasítás nem végez műveletet, így nincs hatással a program végrehajtására. A program a következő utasításával folytatódik.
A NOP utasítást általában késleltetésre használják a programban, az N operandus pozitív egész konstans.

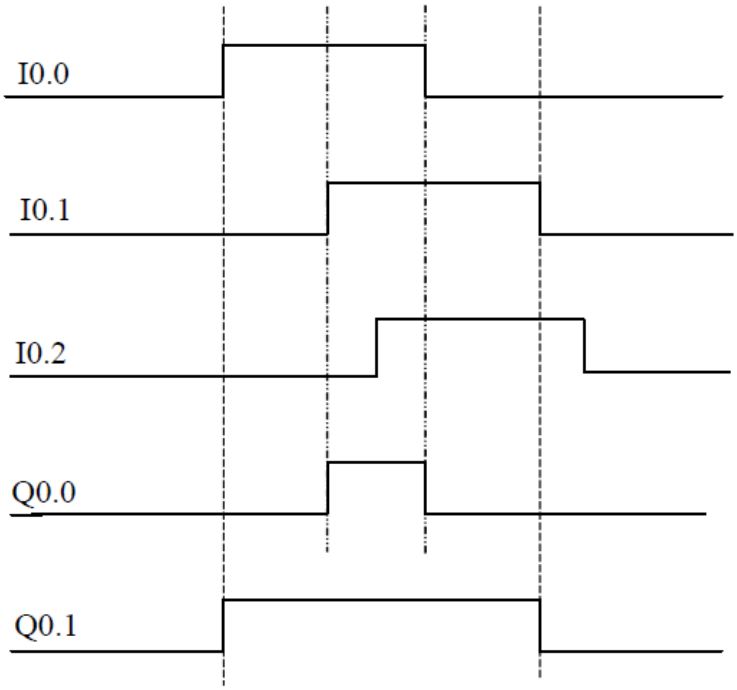
6.2.10. Zárójeles műveletek

	Név	Használat	Csoport	
IL	AND(	AND(	U	☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	OR(	OR(		
	)	)	P	

A zárójeles műveletek csak IL programozási nyelvben érhetők el, mint egyszerű utasítások.

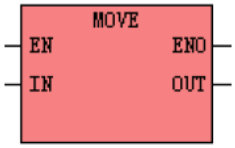
LD	IL
	LD %I0.0 AND(LD %I0.1 OR %I0.2) ST %Q0.0
	LD %I0.0 OR(LD %I0.1 AND %I0.2) ST %Q0.1

A fenti műveletet a program a következőképpen hajtja végre: első lépésben CR értékét ideiglenesen eltárolja, a zárójelben levő műveletet végrehajtja, majd az eredményt és / vagy kapcsolatba hozza az eltárolt CR értékével. A művelet végrehajtását követően az eredmény a CR-be kerül eltárolásra.



6.3. Adatmozgató utasítások

6.3.1. MOVE

	Név	Használat	Csoport	
LD	MOVE			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	MOVE	MOVE *IN, OUT*	U	

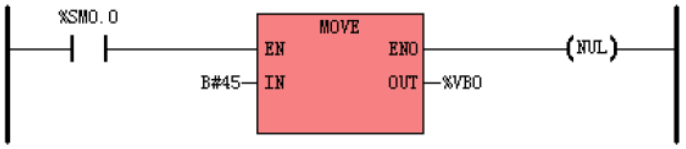
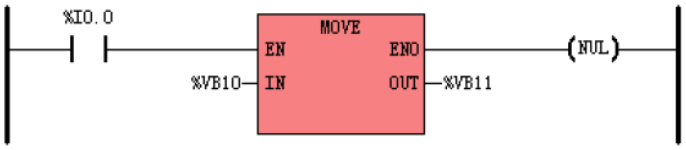
Paraméter	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	BYTE, WORD, DWORD, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, konstans, mutató
OUT	Kimenet	BYTE, WORD, DWORD, INT, DINT, REAL	Q, M, V, L, SM, AQ, mutató

A MOVE utasítás az IN bemenetén megadott értéket átmásolja OUT kimenet által megadott címre. Az IN-nek és OUT-nak azonos típusúnak kell lennie.

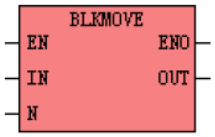
- **LD**
  Ha EN=1, az utasítás végrehajtódik.

- **IL**
  Ha a CR=1, az utasítás végrehajtódik, és nem lesz hatással CR-re.

A következő példákban szereplő %SM0.0 egy speciális regiszter, melynek az értéke mindig 1.

LD		Mivel %SM0.0 értéke mindig 1, ezért a MOVE utasítást mindig végrehajtja a program, vagyis B#45 érték a %VB0 címre menti.
		Ha %I0.0=1, akkor a %VB10 értéke a %VB11-be kerül kimentésre. Ha %I0.0=0, MOVE utasítás nem kerül végrehajtásra.
IL	LD %SM0.0 (* CR értékének 1-be állítása %SM0.0-val *) MOVE B#45, %VB0 (* B#45 érték mentése %VB0 regiszterbe *)	
	LD %I0.0 (* CR=1, ha %I0.0 is 1 *) MOVE %VB10, %VB11 (* Ha CR = 1, akkor % VB10 értéke %VB11 változóba kerül *) (*Ha CR=0, akkor az utasítás nem kerül végrehajtásra, %VB11 értéke változatlan marad*)	

6.3.2. BLKMOVE (Block move)

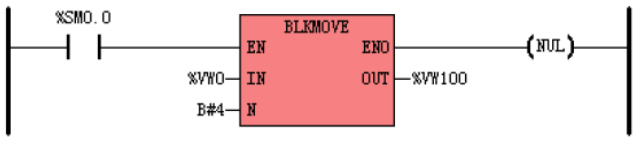
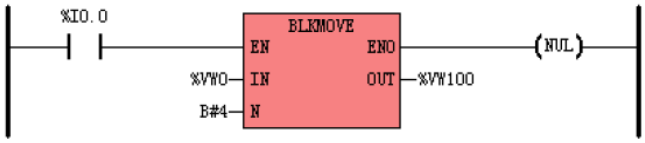
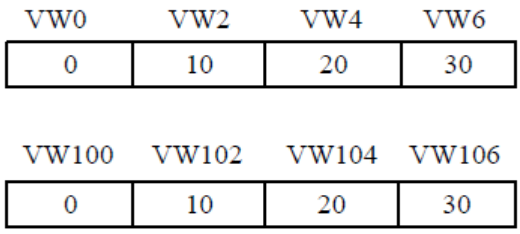
	Név	Használat	Csoport	
LD	BLKMOVE			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	BLKMOVE	BLKMOVE *IN, OUT, N*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	BYTE, WORD, DWORD, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, konstans, mutató
N	Bemenet	BYTE	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE, WORD, DWORD, INT, DINT, REAL	Q, M, V, L, SM, AQ

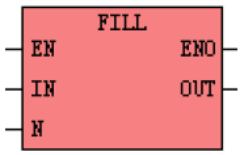
Az IN és OUT azonos típusú kell, hogy legyen.

A BLNKMOVE utasítás áttesz N darabszámú változót az IN címről az OUT címmel kezdődő memória területre.

- **LD**
  Ha EN=1, az utasítás végrehajtódik.
- **IL**
  Ha CR=1, az utasítás végrehajtódik, és nem lesz hatással CR-re.

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>Mivel %SM0.0 értéke mindig 1, ezért VW0 - VW6 közötti adatok %VW100 - %VW106 címre kerülnek.</td></tr>
<tr><td> 
</td><td>Ha %I0.0=1, akkor VW0 - VW6 közötti adatok %VW100 - %VW106 címre kerülnek. Ellenkező esetben az utasítás nem kerül végrehajtásra.</td></tr>
<tr><td>IL</td><td colspan="2">LD %SM0.0 (* CR értékének 1-be állítása %SM0.0-val *)
BLKMOVE %VW0, %VW100, B#4 (* VW0 - VW6 közötti adatok %VW100 - %VW106 címre kerülnek *)</td></tr>
<tr><td></td><td colspan="2">LD %I0.0 (* CR=1, ha %I0.0 is 1 *)
BLKMOVE %VW0, %VW100, B#4 (* Ha CR=0, utasítás nem kerül végrehajtásra*)
(*Ha CR=1, akkor VW0-VW6 közötti adatok %VW100-%VW106 címre kerülnek *)</td></tr>
<tr><td>Eredmény</td><td colspan="2"> 
</td></tr>
</table>

6.3.3. Memória terület feltöltése (FILL)

<table>
<tr><td rowspan="2">LD</td><td>Név</td><td>Használat</td><td>Csoport</td><td rowspan="3">☑ CPU304
☑ CPU304EX
☑ CPU306
☑ CPU306EX
☑ CPU308</td></tr>
<tr><td>FILL</td><td> 
</td><td></td></tr>
<tr><td>IL</td><td>FILL</td><td>FILL IN, OUT, N</td><td>U</td></tr>
</table>

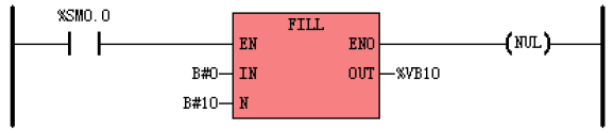
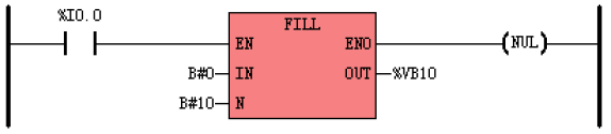
Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	BYTE	konstans
N	Bemenet	BYTE	konstans
OUT	Kimenet	BYTE	M, V, L

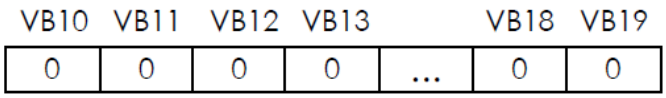
A FILL utasítás N számú változót egyenlővé tesz az IN bemeneten megadott konstanssal, az OUT

címtől kezdődően.

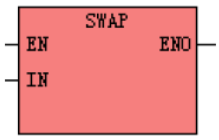
- **LD**
  Ha EN=1, az utasítás végrehajtódik

- **IL**
  HA CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>Mivel %SM0.0 értéke mindig 1, ezért minden ciklusban %VB10-től 10 változó (%VB10 ~ %VB19) B#0 értéket vesz fel.</td></tr>
<tr><td> 
</td><td>Ha %I0.0 értéke 1, akkor %VB10-től 10 változó (%VB10 ~ %VB19) B#0 értéket vesz fel.
Ellenkező esetben az utasítás nem kerül végrehajtásra.</td></tr>
<tr><td>IL</td><td colspan="2">LD %SM0.0 (* CR értékének 1-be állítása %SM0.0-val *)
FILL B#0, %VB10, B#10 (* %VB10-től 10 változó (%VB10 ~ %VB19) B#0 értéket vesz fel. *)

LD %I0.0 (* CR=1, ha %I0.0 is 1 *)
FILL B#0, %VB10, B#10 (* Ha CR=0, utasítás nem kerül végrehajtásra*)
(*Ha %I0.0 értéke 1, akkor %VB10-től 10 változó (%VB10 ~ %VB19) B#0 értéket vesz fel. *)</td></tr>
<tr><td>Eredmény</td><td colspan="2"> 
</td></tr>
</table>

6.3.4. Csere (Swap)

<table>
<tr><td rowspan="2">LD</td><td>Név</td><td>Használat</td><td>Csoport</td><td rowspan="2">☐ CPU304
☐ CPU304EX
☐ CPU306
☑ CPU306EX
☑ CPU308</td></tr>
<tr><td>SWAP</td><td> 
</td><td></td></tr>
<tr><td>IL</td><td>SWAP</td><td>SWAP IN</td><td>U</td></tr>
</table>

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet/Kimenet	WORD, DWORD	Q, M, V, L, SM

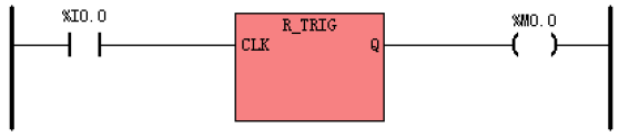
A SWAP utasítás felcseréli a legnagyobb helyi értékű byte-ot a legkisebb helyi értékű byte-tal egy szavas (IN) változón belül, vagy felcseréli a legnagyobb helyiértékű szót a legkisebb helyiértékű szóval egy dupla szavas (IN) változón belül.

- **LD**
  Ha EN=1, az utasítás végrehajtódik

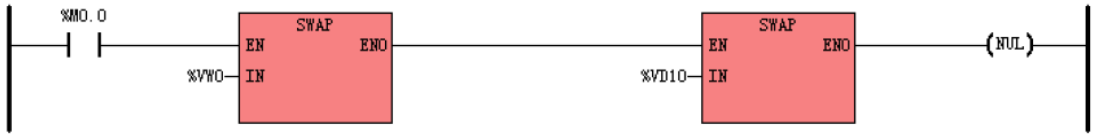
- **IL**
  HA CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

LD

(* Network 0*)
(* Az %I0.0 bemenet felfutó élére a program felcseréli a legnagyobb helyiértékű bájtot a legkisebb helyiértékű bájttal %VW0 változón belül és a %VD10 duplaszó méretű változó legnagyobb helyiértékű szavát cseréli fel a legkisebb helyiértékűvel.*)



(* Network 1*)

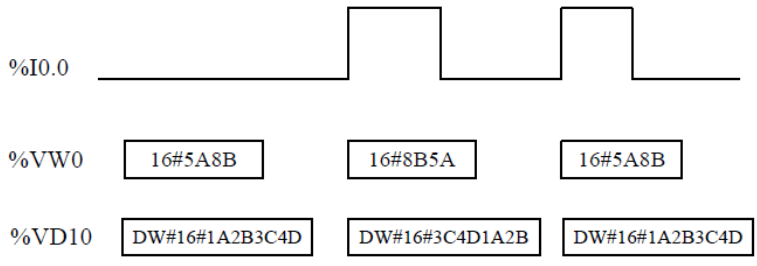


IL

```
(* Network 0 *)
LD %I0.0
R_TRIG (* %I0.0 felfutó él detektálás *)
SWAP %VW0 (*  felcseréli a legnagyobb helyiértékű byte-ot a legkisebb helyiértékű byte-tal %VW0 változón belül *)
SWAP %VD10 (* a %VD10 duplaszó méretű változó legnagyobb helyiértékű szavát cseréli fel a legkisebb helyiértékűvel.*)
```

Eredmény

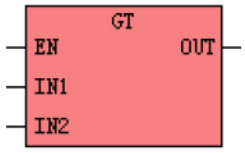
A %VW0 kezdeti értéke W#16#5A8B,
A %VD10 kezdeti értéke pedig DW#16#1A2B3C4D.



6.4. Összehasonlító utasítások

Minden összehasonlító utasításnál a Byte-os összehasonlítások előjel nélküliek. INT, DINT és REAL összehasonlítások pedig előjelesek.

6.4.1. Nagyobb mint (GT)

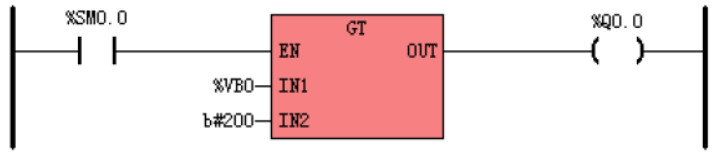
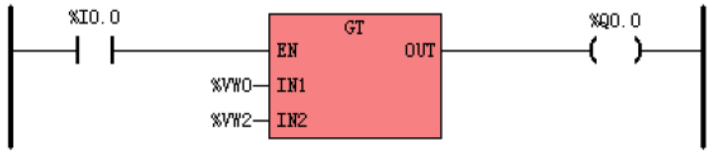
	Név	Használat	Csoport	
LD	GT			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	GT	GT *IN1, IN2*	P	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN1	Bemenet	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, konstans, mutató
IN2	Bemenet	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, konstans, mutató
OUT (LD)	Kimenet	BOOL	Program összeköttetés

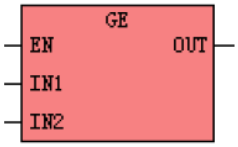
IN1 és IN2 adattípusának azonosnak kell lennie.

- **LD**
  Ha EN=1, és IN1 nagyobb, mint IN2, akkor a kimenet bekapcsol
  Ha EN=0, az utasítás nem kerül végrehajtásra, kimenet kikapcsol.

- **IL**
  Ha CR=1, az utasítás végrehajtásra kerül, és ha IN1 nagyobb, mint IN2 akkor a CR-be 1 kerül.
  Ha CR=0, az utasítás nem hajtódik végre és a CR-be 0 kerül.

LD		Mivel %SM0.0 értéke mindig 1, így a GT utasítás mindig végrehajtódik. Ha a %VB0 értéke nagyobb, mint B#200, Q0.0 bekapcsol, ellenkező esetben pedig kikapcsol.
		Ha %I0.0 = 1, akkor GT utasítás végrehajtódik. Ha a %VW0 értéke nagyobb, mint %VW2 értéke, Q0.0 bekapcsol, ellenkező esetben pedig kikapcsol.
IL	LD %SM0.0 (* CR értékének 1-be állítása %SM0.0-val *) GT %VB0, B#200 (* Ha VB0 nagyobb, mint B#200, akkor CR értéke 1 lesz, ellenkező esetben 0*) ST %Q0.0 (* Q0.0 felveszi CR értékét *)	
	LD %I0.0 (* CR=1, ha %I0.0 is 1 *) GT %VW0, %VW2 (* Ha VW0 nagyobb, mint VW2, akkor CR értéke 1 lesz, ellenkező esetben 0*) (* Ha CR=0, utasítás nem kerül végrehajtásra*) ST %Q0.0 (* Q0.0 felveszi CR értékét *)	

6.4.2. Nagyobb vagy egyenlő (GE)

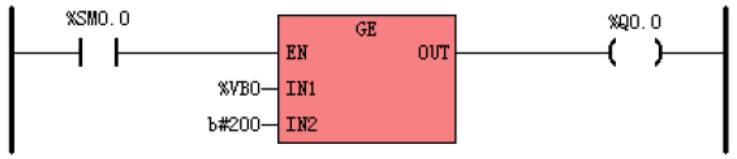
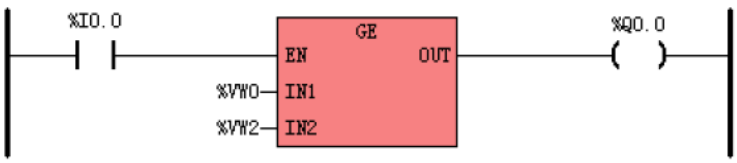
	Név	Használat	Csoport	
LD	GE			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	GE	GE *IN1, IN2*	P	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN1	Bemenet	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, konstans, mutató
IN2	Bemenet	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, konstans, mutató
OUT(LD)	Kimenet	BOOL	Program összeköttetés

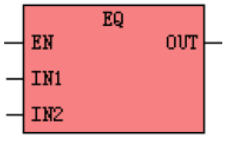
IN1 és IN2 adattípusának meg kell egyeznie.

- **LD**
  Ha EN=1, és IN1 nagyobb vagy egyenlő, mint IN2, a kimenet bekapcsol, ellenkező esetben pedig kikapcsol.
  Ha EN=0, az utasítás nem hajtódik végre és az OUT kimenet 0 lesz.

- **IL**
  Ha CR=1, és IN1 nagyobb vagy egyenlő, mint IN2 akkor CR-be 1 kerül, ellenkező esetben pedig CR értéke 0 lesz.
  Ha CR=0, az utasítás nem hajtódik végre és az CR-be 0 kerül.

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>Mivel %SM0.0 értéke mindig 1, így a GE utasítás mindig végrehajtódik. Ha a %VB0 értéke nagyobb, vagy egyenlő, mint B#200, Q0.0 bekapcsol, ellenkező esetben pedig kikapcsol.</td></tr>
<tr><td> 
</td><td>Ha %I0.0 = 1, akkor GE utasítás végrehajtódik. Ha a %VW0 értéke nagyobb, vagy egyenlő, mint %VW2 értéke, Q0.0 bekapcsol, ellenkező esetben pedig kikapcsol.</td></tr>
<tr><td>IL</td><td colspan="2">LD %SM0.0 (* CR értékének 1-be állítása %SM0.0-val *)
GE %VB0, B#200 (* Ha VB0 nagyobb vagy egyenlő, mint B#200, CR 1 értéket vesz fel, ellenkező esetben CR 0 lesz*)
ST %Q0.0 (* Q0.0 felveszi CR értékét *)</td></tr>
<tr><td></td><td colspan="2">LD %I0.0 (* CR=1, ha %I0.0 is 1 *)
GE %VW0, %VW2 (* Ha VW0 nagyobb, mint VW2, akkor CR értéke 1 lesz, ellenkező esetben 0*)
(* Ha CR=0, utasítás nem kerül végrehajtásra*)
ST %Q0.0 (* Q0.0 felveszi CR értékét *)</td></tr>
</table>

6.4.3. Egyenlő (EQ)

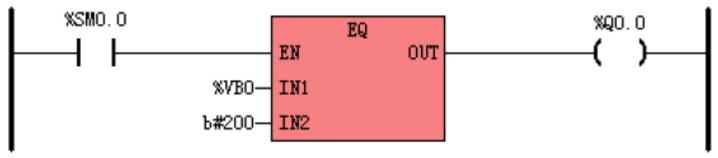
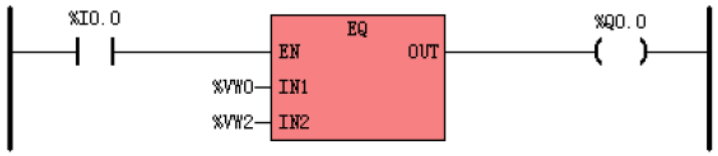
<table>
<tr><th></th><th>Név</th><th>Használat</th><th>Csoport</th><th rowspan="3">☑ CPU304
☑ CPU304EX
☑ CPU306
☑ CPU306EX
☑ CPU308</th></tr>
<tr><td>LD</td><td>EQ</td><td> 
</td><td></td></tr>
<tr><td>IL</td><td>EQ</td><td>EQ IN1, IN2</td><td>P</td></tr>
</table>

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN1	Bemenet	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, konstans, mutató
IN2	Bemenet	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, konstans, mutató
OUT(LD)	Kimenet	BOOL	Program összeköttetés

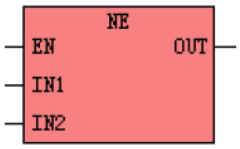
IN1 és IN2 adattípusának meg kell egyeznie.

- **LD**
  Ha EN=1, és IN1 egyenlő IN2-vel, a kimenet bekapcsol, ellenkező esetben pedig kikapcsol.
  Ha EN=0, az utasítás nem hajtódik végre és az OUT kimenet 0 lesz.

- **IL**
  HA CR=1, és IN1 egyenlő IN2-vel akkor CR-be 1 kerül, ellenkező esetben pedig CR értéke 0 lesz.
  Ha CR=0, az utasítás nem hajtódik végre és az CR-be 0 kerül.

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>Mivel %SM0.0 értéke mindig 1, így a EQ utasítás mindig végrehajtódik. Ha VB0 értéke B#200, akkor a kimenet bekapcsol, ellenkező esetben pedig kikapcsol.</td></tr>
<tr><td> 
</td><td>Ha %I0.0 = 1, akkor EQ utasítás végrehajtódik. Ha VW0 egyenlő VW2-vel, akkor kimenet bekapcsol, ellenkező esetben pedig kikapcsol.</td></tr>
<tr><td>IL</td><td colspan="2">LD %SM0.0 (* CR értékének 1-be állítása %SM0.0-val *)
EQ %VB0, B#200 (* Ha VB0 egyenlő B#200-al, akkor CR értéke 1 lesz, ellenkező esetben 0*)
ST %Q0.0 (* Q0.0 felveszi CR értékét *)</td></tr>
<tr><td></td><td colspan="2">LD %I0.0 (* CR=1, ha %I0.0 is 1 *)
EQ %VW0, %VW2 (* Ha VW0 egyenlő VW2-vel, akkor CR értéke 1 lesz, ellenkező esetben 0*)
(* Ha CR=0, utasítás nem kerül végrehajtásra*)
ST %Q0.0 (* Q0.0 felveszi CR értékét *)</td></tr>
</table>

6.4.4. Nem egyenlő (NE)

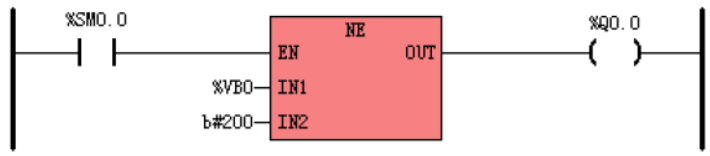
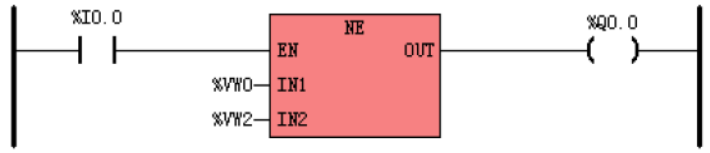
	Név	Használat	Csoport	
LD	NE			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	NE	NE *IN1, IN2*	P	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN1	Bemenet	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, konstans, mutató
IN2	Bemenet	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, konstans, mutató
OUT(LD)	Kimenet	BOOL	Program összeköttetés

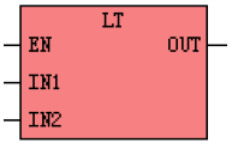
IN1 és IN2 adattípusának meg kell egyeznie.

- **LD**
  Ha EN=1, és IN1 nem egyenlő IN2-vel, a kimenet bekapcsol, ellenkező esetben pedig kikapcsol.
  Ha EN=0, az utasítás nem hajtódik végre és az OUT kimenet 0 lesz.

- **IL**
  HA CR=1, és IN1 nem egyenlő IN2-vel akkor CR-be 1 kerül, ellenkező esetben pedig CR értéke 0 lesz.
  Ha CR=0, az utasítás nem hajtódik végre és az CR-be 0 kerül.

LD		Mivel %SM0.0 értéke mindig 1, így a NE utasítás mindig végrehajtódik. Ha VB0 értéke nem egyenlő B#200-al, akkor a kimenet bekapcsol, ellenkező esetben pedig kikapcsol.
		Ha %I0.0 = 1, akkor NE utasítás végrehajtódik. Ha VW0 nem egyenlő VW2-vel, akkor kimenet bekapcsol, ellenkező esetben pedig kikapcsol.
IL	LD %SM0.0 (* CR értékének 1-be állítása %SM0.0-val *) NE %VB0, B#200 (* Ha VB0 értéke nem egyenlő B#200-al, akkor a CR 1-lesz, ellenkező esetben pedig CR 0-lesz *) ST %Q0.0 (* Q0.0 felveszi CR értékét *)	
	LD %I0.0 (* CR=1, ha %I0.0 is 1 *) NE %VW0, %VW2 (* Ha VW0 nem egyenlő VW2-vel, akkor CR 1 lesz, ellenkező esetben pedig CR 0 lesz. *) (* Ha CR=0, utasítás nem kerül végrehajtásra*) ST %Q0.0 (* Q0.0 felveszi CR értékét *)	

6.4.5. Kisebb mint (LT)

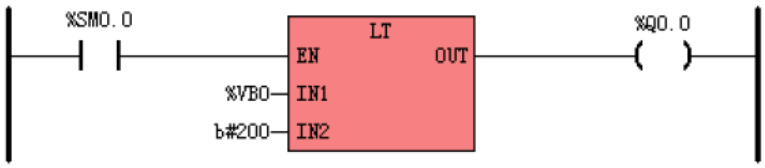
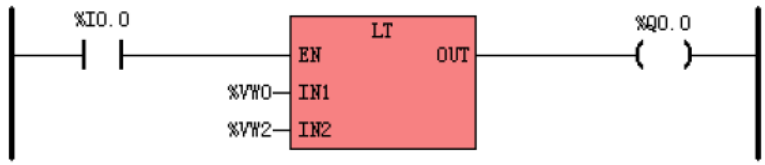
	Név	Használat	Csoport	
LD	LT			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	LT	LT *IN, IN2*	P	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN2	Bemenet	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, konstans, mutató
IN2	Bemenet	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, konstans, mutató
OUT(LD)	Kimenet	BOOL	Program összeköttetés

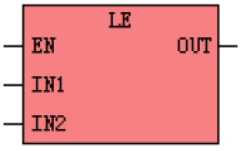
IN1 és IN2 adattípusának meg kell egyeznie.

- **LD**
  Ha EN=1, és IN1 kisebb mint IN2, a kimenet bekapcsol, ellenkező esetben pedig kikapcsol.
  Ha EN=0, az utasítás nem hajtódik végre és az OUT kimenet 0 lesz.

- **IL**
  HA CR=1, és IN1 kisebb mint IN2, akkor CR-be 1 kerül, ellenkező esetben pedig CR értéke 0 lesz.
  Ha CR=0, az utasítás nem hajtódik végre és a CR 0 állapotban marad.

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>Mivel %SM0.0 értéke mindig 1, így a LT utasítás mindig végrehajtódik. Ha VB0 értéke kisebb mint B#200, akkor a kimenet bekapcsol, ellenkező esetben pedig kikapcsol.</td></tr>
<tr><td> 
</td><td>Ha %I0.0 = 1, akkor LT utasítás végrehajtódik. Ha VW0 kisebb mint VW2, akkor kimenet bekapcsol, ellenkező esetben pedig kikapcsol.</td></tr>
<tr><td>IL</td><td colspan="2">LD %SM0.0 (* CR értékének 1-be állítása %SM0.0-val *)
LT %VB0, B#200 (* Ha VB0 értéke kisebb mint B#200, akkor a CR 1-lesz, ellenkező esetben pedig CR 0-lesz *)
ST %Q0.0 (* Q0.0 felveszi CR értékét *)</td></tr>
<tr><td>IL</td><td colspan="2">LD %I0.0 (* CR=1, ha %I0.0 is 1 *)
LT %VW0, %VW2 (* Ha VW0 kisebb mint VW2, akkor CR 1 lesz, ellenkező esetben pedig CR 0 lesz. *)
(* Ha CR=0, utasítás nem kerül végrehajtásra*)
ST %Q0.0 (* Q0.0 felveszi CR értékét *)</td></tr>
</table>

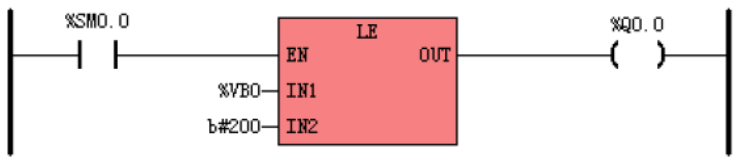
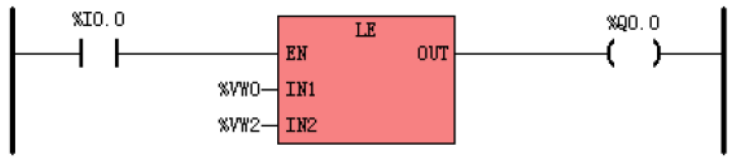
6.4.6.Kisebb, vagy egyenlő (LE)

<table>
<tr><td rowspan="2">LD</td><td>Név</td><td>Használat</td><td>Csoport</td><td rowspan="3">☑ CPU304
☑ CPU304EX
☑ CPU306
☑ CPU306EX
☑ CPU308</td></tr>
<tr><td>LE</td><td> 
</td><td></td></tr>
<tr><td>IL</td><td>LE</td><td>LE IN1, IN2</td><td>P</td></tr>
</table>

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN1	Bemenet	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, konstans, mutató
IN2	Bemenet	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, konstans, mutató
OUT(LD)	Kimenet	BOOL	Program összeköttetés

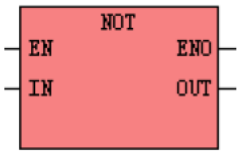
- **LD**
  Ha EN=1, és IN1 kisebb, vagy egyenlő mint IN2, a kimenet bekapcsol, ellenkező esetben pedig kikapcsol.
  Ha EN=0, az utasítás nem hajtódik végre és az OUT kimenet 0 marad.

- **IL**
  HA CR=1, és IN1 kisebb, vagy egyenlő mint IN2, akkor CR-be 1 kerül, ellenkező esetben pedig CR értéke 0 lesz.
  Ha CR=0, az utasítás nem hajtódik végre és a CR 0 állapotban marad.

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>Mivel %SM0.0 értéke mindig 1, így a LE utasítás mindig végrehajtódik. Ha VB0 értéke kisebb, vagy egyenlő mint B#200, akkor a kimenet bekapcsol, ellenkező esetben pedig kikapcsol.</td></tr>
<tr><td> 
</td><td>Ha %I0.0 = 1, akkor LE utasítás végrehajtódik. Ha VW0 kisebb, vagy egyenlő mint VW2, akkor kimenet bekapcsol, ellenkező esetben pedig kikapcsol.</td></tr>
<tr><td rowspan="2">IL</td><td colspan="2">LD %SM0.0 (* CR értékének 1-be állítása %SM0.0-val *)
LE %VB0, B#200 (* Ha VB0 értéke kisebb, vagy egyenlő mint B#200, akkor a CR 1-lesz, ellenkező esetben pedig CR 0-lesz *)
ST %Q0.0 (* Q0.0 felveszi CR értékét *)</td></tr>
<tr><td colspan="2">LD %I0.0 (* CR=1, ha %I0.0 is 1 *)
LE %VW0, %VW2 (* Ha VW0 kisebb, vagy egyenlő mint VW2, akkor CR 1 lesz, ellenkező esetben pedig CR 0 lesz. *)
(* Ha CR=0, utasítás nem kerül végrehajtásra*)
ST %Q0.0 (* Q0.0 felveszi CR értékét *)</td></tr>
</table>

6.5. Logikai műveletek

6.5.1. Negálás (NOT)

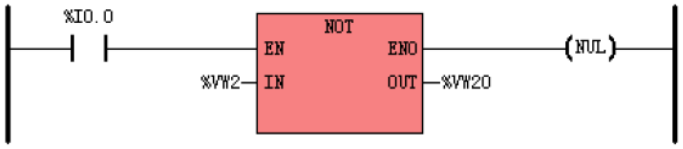
	Név	Használat	Csoport	
LD	NOT			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	NOT	NOT *OUT*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE, WORD, DWORD	Program összeköttetés

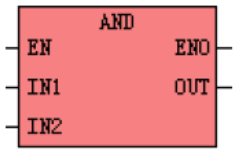
IN és OUT adattípusának meg kell egyeznie.

- **LD**
  Ha EN=1, az OUT kimeneten az IN bemenet ellentettje jelenik meg. Ha IN=1, akkor OUT 0, ellenkező esetben pedig OUT 1 lesz.
  Ha EN=0, az utasítás nem hajtódik végre.

- **IL**
  HA CR=1, az OUT kimeneten az IN bemenet ellentettje jelenik meg. Ha IN=1, akkor OUT 0, ellenkező esetben pedig OUT 1 lesz. A művelet a CR-re nincs hatással.
  Ha CR=0, az utasítás nem hajtódik.

LD		Ha I0.0=0 NOT utasítás nem kerül végrehajtásra. Ha I0.0=1, a NOT utasítás a VW2 minden bitjét ellenkezőjére változtatja, és az eredmény a VW20-ba kerül.
IL	LD %I0.0 (* CR=1, ha %I0.0 is 1 *) NOT %VW20 (* Ha CR=1, az utasítás VW2 minden bitjét ellenkezőjére változtatja, és az eredmény a VW20-ba kerül. *) (* Ha CR=0 az utasítás nem kerül végrehajtásra *)	
Eredmény	A fenti LD program futásának eredménye, ha VW2 = W#16#5555 akkor VW20 = W#16#AAAA értéket vesz fel.	

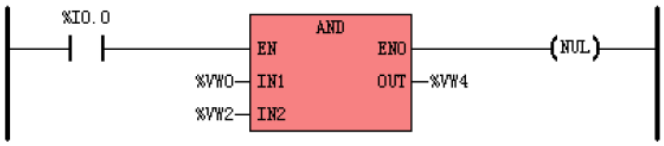
6.5.2. Logikai ÉS (AND)

	Név	Használat	Csoport	
LD	AND			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	AND	AND *IN, OUT*	U	

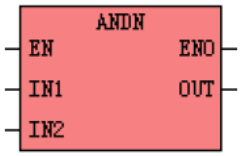
Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN1	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
IN2	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE, WORD, DWORD	Q, M, V, L, SM

- **LD**
  IN1,IN2 és OUT adattípusának meg kell egyeznie.
  Ha EN=1, az utasítás "ÉS" kapcsolatba hozza az IN1 és IN2 bemenetet, majd az eredményt az OUT-ra teszi.
  Ha EN=0, az utasítás nem hajtódik végre.

- **IL**
  IN és OUT adattípusának meg kell egyeznie.
  HA CR=1, az utasítás "ÉS" kapcsolatba hozza az IN és OUT bemenetet, majd az eredményt az OUT-ra teszi. A művelet a CR értékére nincs hatással.
  Ha CR=0, az utasítás nem hajtódik végre.

LD		Ha I0.0=0 AND utasítás nem kerül végrehajtásra. Ha I0.0=1, a AND művelet ÉS kapcsolatban hozza VW0 és VW2 bitjeit, és az eredmény a VW4-be kerül.
IL	LD %I0.0 (* CR=1, ha %I0.0 is 1 *) AND %VW0, %VW2 (* Ha CR=1, az utasítás ÉS kapcsolatban hozza VW0 és VW2 bitjeit, és az eredmény a VW4-be kerül. *) (* Ha CR=0 az utasítás nem kerül végrehajtásra *)	
Eredmény	A fenti LD program futásának eredménye, ha VW0= W#16#129B és VW2 = W#16#960F, akkor VW4 = W#16#120B értéket vesz fel.	

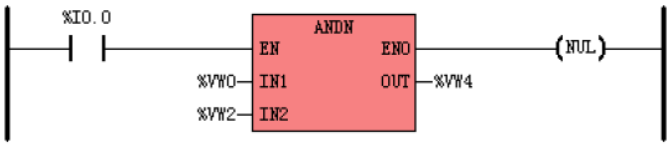
6.5.3. Logikai negált ÉS (ANDN)

	Név	Használat	Csoport	
LD	ANDN			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	ANDN	ANDN *IN, OUT*	U	

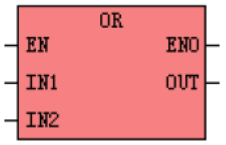
Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN1	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
IN2	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE, WORD, DWORD	Q, M, V, L, SM

- **LD**
  IN1,IN2 és OUT adattípusának meg kell egyeznie.
  Ha EN=1, az utasítás bitenkénti "ÉS" kapcsolatba hozza az IN1 és IN2 bemenetet majd invertálja a biteket és az OUT-ra teszi az eredményt.
  Ha EN=0, az utasítás nem hajtódik végre.

- **IL**
  IN és OUT adattípusának meg kell egyeznie.
  HA CR=1, az utasítás "ÉS" kapcsolatba hozza az aktuális IN és OUT bitjeit, majd invertálja a biteket és az OUT-ra teszi. Nincs hatással a CR-re.
  Ha CR=0, az utasítás nem hajtódik végre.

LD		Ha I0.0 = 0, ANDN utasítás nem hajtódik végre. Ha I0.0 = 1, ANDN utasítás végrehajtódik, bitenkénti ÉS kapcsolatba hozza VW0 és VW2-t, majd az eredmény bitjeit invertálja, a végeredmény a VW4-re kerül.
IL	LD %I0.0 (* CR=1, ha %I0.0 is 1 *) ANDN %VW0, %VW2 (* Ha CR=1, akkor az utasítás ÉS kapcsolatba hozza *) (*VW0 és VW2-t, majd az eredményt bitenként invertálja. *) (*A végeredmény VW2-be kerül*) (* Ha CR=0, az utasítás nem kerül végrehajtásra *)	
Eredmény	A fenti LD program futásának eredménye, ha VW0= W#16#129B és VW2 = W#16#960F, akkor VW4 = W#16#EDF4 értéket vesz fel.	

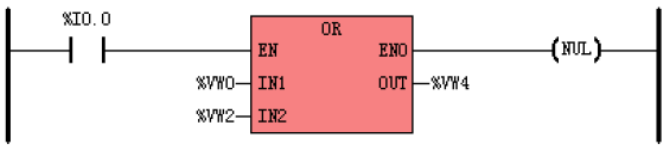
6.5.4. Logikai VAGY (OR)

	Név	Használat	Csoport	
LD	OR			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	OR	OR *IN, OUT*	U	

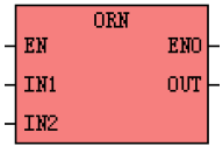
Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN1	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
IN2	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE, WORD, DWORD	Q, M, V, L, SM

- **LD**
  IN1,IN2 és OUT adattípusának meg kell egyeznie.
  Ha EN=1, az utasítás "VAGY" kapcsolatba hozza az IN1 és IN2 bitjeit, majd az eredmény az OUT-ra kerül.
  Ha EN=0, az utasítás nem hajtódik végre.

- **IL**
  IN és OUT adattípusának meg kell egyeznie.
  HA CR=1, az utasítás "VAGY" kapcsolatba hozza IN1 és OUT bitjeit, majd a biteket az OUT-ra teszi. Nincs hatással a CR-re.
  Ha CR=0, az utasítás nem hajtódik.

LD		Ha I0.0=0, OR utasítás nem hajtódik végre. Ha I0.0=1, akkor az utasítás VAGY kapcsolatba hozza VW0 és VW2 bitjeit, az eredmény a VW4-re kerül.
IL	LD %I0.0 (* CR=1, ha %I0.0 is 1 *) OR %VW0, %VW2 (* Ha CR=1, akkor az utasítás VAGY kapcsolatba hozza VW0*) (* és VW2 bitjeit, majd az eredmény a VW2-re kerül*) (* Ha CR=0, az utasítás nem kerül végrehajtásra *)	
Eredmény	A fenti LD program futásának eredménye, ha VW0= W#16#5555 és VW2 = W#16#AAAA, akkor VW4 = W#16#FFFF értéket vesz fel.	

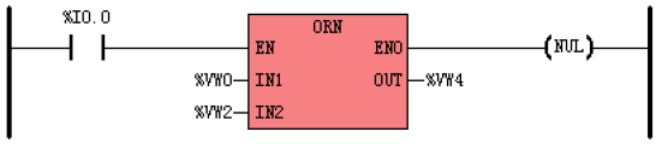
6.5.5. Logikai negált VAGY (ORN)

	Név	Használat	Csoport	
LD	ORN			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	ORN	ORN *IN, OUT*	U	

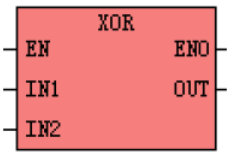
Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN1	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
IN2	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE, WORD, DWORD	Q, M, V, L, SM

- **LD**
  IN1,IN2 és OUT adattípusának meg kell egyeznie.
  Ha EN=1, az utasítás "VAGY" kapcsolatba hozza az IN1 és IN2 majd invertálja a biteket, az eredmény az OUT-ra kerül.
  Ha EN=0, az utasítás nem hajtódik végre.

- **IL**
  IN és OUT adattípusának meg kell egyeznie.
  HA CR=1, az utasítás "VAGY" kapcsolatba hozza IN és OUT bitjeit, majd invertálja a biteket, az eredmény az OUT-ra kerül. Nincs hatással a CR-re.
  Ha CR=0, az utasítás nem hajtódik.

LD		Ha I0.0=0, ORN utasítás nem hajtódik végre. Ha I0.0=1, ORN utasítás végrehajtódik, bitenkénti VAGY kapcsolatba hozza VW0 és VW2-t, majd az eredmény bitjeit invertálja, a végeredmény a VW4-re kerül.
IL	LD %I0.0 (* CR=1, ha %I0.0 is 1 *) ORN %VW0, %VW2 (* Ha CR=1, akkor az utasítás VAGY kapcsolatba hozza *) (*VW0 és VW2-t, majd az eredményt bitenként invertálja. *) (*A végeredmény VW2-be kerül*) (* Ha CR=0, az utasítás nem kerül végrehajtásra *)	
Eredmény	A fenti LD program futásának eredménye, ha VW0= W#16#129B és VW2 = W#16#960F , akkor VW4 = W#16#6960 értéket vesz fel.	

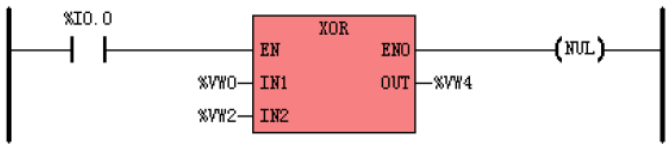
6.5.6. Kizáró vagy (XOR)

	Név	Használat	Csoport	
LD	XOR			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	XOR	XOR *IN, OUT*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN1	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
IN2	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE, WORD, DWORD	Q, M, V, L, SM

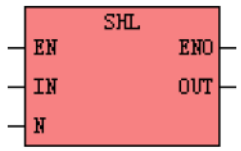
- **LD**
  IN1,IN2 és OUT adattípusának meg kell egyeznie.
  Ha EN=1, az utasítás "KIZÁRÓ-VAGY" kapcsolatba hozza az IN1 és IN2 bitjeit, az eredmény az OUT-ra kerül.
  Ha EN=0, az utasítás nem hajtódik végre.

- **IL**
  IN és OUT adattípusának meg kell egyeznie.
  HA CR=1, az utasítás "KIZÁRÓ-VAGY" kapcsolatba hozza IN és OUT bitjeit, az eredmény az OUT-ra kerül. Nincs hatással a CR-re.
  Ha CR=0, az utasítás nem hajtódik.

LD		Ha I0.0=0, XOR utasítás nem hajtódik végre. Ha I0.0=1, akkor az utasítás kizáró-vagy kapcsolatba hozza VW0 és VW2 bitjeit, az eredmény a VW4-re kerül.
IL	LD %I0.0 (* CR=1, ha %I0.0 is 1 *) XOR %VW0, %VW2 (* Ha CR=1, akkor az utasítás KIZÁRÓ-VAGY kapcsolatba *) (*VW0 és VW2-t, az eredmény VW2-be kerül*) (* Ha CR=0, az utasítás nem kerül végrehajtásra *)	
Eredmény	A fenti LD program futásának eredménye, ha VW0= W#16#9514 és VW2 = W#16#B9A1 , akkor VW4 = W#16#2CB5 értéket vesz fel.	

6.6 Léptetés / Forgatás utasítások

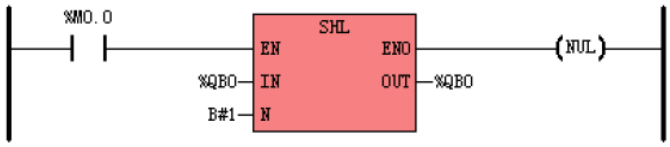
6.6.1. Léptetés balra (SHL)

	Név	Használat	Csoport	
LD	SHL			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	SHL	SHL *OUT, N*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
N	Bemenet	BYTE	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE, WORD, DWORD	Q, M, V, L, SM

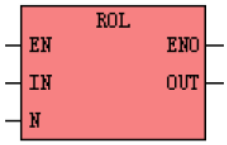
- **LD**
  IN és OUT adattípusának meg kell egyeznie.
  Ha EN=1, az utasítás balra eltolja az IN változót N bittel, a kilépő bitek helyére 0 kerül. Az eredmény OUT-ra kerül.
  Ha EN=0, az utasítás nem hajtódik végre.

- **IL**
  IN és OUT adattípusának meg kell egyeznie.
  Ha CR=1, az utasítás balra eltolja az OUT értékét N bittel ,a kilépő bitek helyére 0 kerül. Az eredmény OUT-ra kerül. A művelet nincs hatással CR-re.
  Ha CR=0, az utasítás nem hajtódik végre.

LD		Ha M0.0=0, SHL utasítás nem hajtódik végre. Ha I0.0=1, akkor az utasítás eggyel balra lépteti QB0 bitjeit, az eredmény a QB0-ra kerül.
IL	LD %M0.0 (* CR=1, ha %M0.0 is 1 *) SHL %QB0, B#1 (* Ha CR=1, az utasítás QB0 bitjeit balra lépteti eggyel, és az*) (*eredmény QB0-ba kerül *) (* Ha CR=0, az utasítás nem kerül végrehajtásra *)	
Eredmény	QB0 B#2#10000001	

	1. végrehajtás	2. végrehajtás	3. végrehajtás	4. végrehajtás
QB0	B#2#00000010	B#2#00000100	B#2#00001000	B#2#00010000

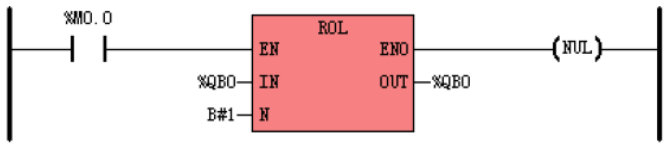
6.6.2. Forgatás balra (ROL)

	Név	Használat	Csoport	
LD	ROL			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	ROL	ROL *OUT, N*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
N	Bemenet	BYTE	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE, WORD, DWORD	Q, M, V, L, SM

- **LD**
  IN és OUT adattípusának meg kell egyeznie.
  Ha EN=1, az utasítás balra forgatja (MSB-től LSB-ig) IN bitjeit N-szer, az eredmény az OUT-ba kerül.
  Ha EN=0, az utasítás nem hajtódik végre.

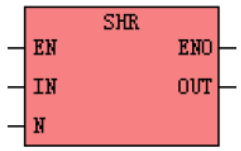
- **IL**
  IN és OUT adattípusának meg kell egyeznie.
  Ha CR=1, az utasítás balra forgatja (MSB-től LSB-ig) OUT bitjeit N-szer, az eredmény az OUT-ba kerül. A művelet nincs hatással CR értékére.
  Ha CR=0, az utasítás nem hajtódik végre.

LD		Ha M0.0=0, ROL utasítás nem hajtódik végre. Ha I0.0=1, akkor az utasítás eggyel balra forgatja QB0 bitjeit, az eredmény a QB0-ra kerül.
IL	LD %M0.0 (* CR=1, ha %M0.0 is 1 *) ROL %QB0, B#1 (* Ha CR=1, az utasítás QB0 bitjeit balra forgatja eggyel, és az*) (*eredmény QB0-ba kerül *) (* Ha CR=0, az utasítás nem kerül végrehajtásra *)	

Eredmény

QB0	B#2#10100001			
	1. végrehajtás	2. végrehajtás	3. végrehajtás	4. végrehajtás
QB0	B#2#01000011	B#2#10000110	B#2#00001101	B#2#00011010

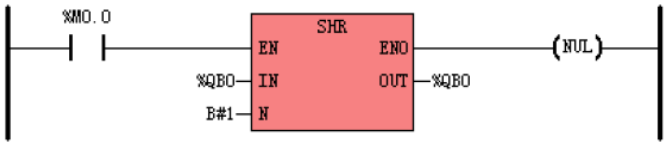
6.6.3. Léptetés jobbra (SHR)

	Név	Használat	Csoport	
LD	SHR			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	SHR	SHR *OUT, N*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
N	Bemenet	BYTE	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE, WORD, DWORD	Q, M, V, L, SM

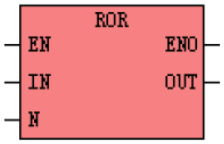
- **LD**
  IN és OUT adattípusának meg kell egyeznie.
  Ha EN=1, az utasítás jobbra eltolja az IN változót N bittel, a kilépő bitek helyére 0 kerül. Az eredmény OUT-ra kerül.
  Ha EN=0, az utasítás nem hajtódik végre.

- **IL**
  IN és OUT adattípusának meg kell egyeznie.
  Ha CR=1, az utasítás jobbra eltolja az OUT értékét N bittel ,a kilépő bitek helyére 0 kerül. Az eredmény OUT-ra kerül. A művelet nincs hatással CR-re.
  Ha CR=0, az utasítás nem hajtódik végre.

LD		Ha M0.0=0, SHR utasítás nem hajtódik végre. Ha I0.0=1, akkor az utasítás eggyel jobbra lépteti QB0 bitjeit, az eredmény a QB0-ra kerül.
IL	LD %M0.0 (* CR=1, ha %M0.0 is 1 *) SHL %QB0, B#1 (* Ha CR=1, az utasítás QB0 bitjeit jobbra lépteti eggyel, és az*) (*eredmény QB0-ba kerül *) (* Ha CR=0, az utasítás nem kerül végrehajtásra *)	
Eredmény	QB0 B#2#10000001	

	1. végrehajtás	2. végrehajtás	3. végrehajtás	4. végrehajtás
QB0	B#2#01000000	B#2#00100000	B#2#00010000	B#2#00001000

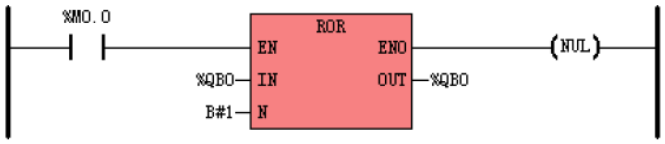
6.6.4. Forgatás jobbra (ROR)

	Név	Használat	Csoport	
LD	ROR			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	ROR	ROR *OUT, N*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	BYTE, WORD, DWORD	I, Q, M, V, L, SM, konstans
N	Bemenet	BYTE	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE, WORD, DWORD	Q, M, V, L, SM

- **LD**
  IN és OUT adattípusának meg kell egyeznie.
  Ha EN=1, az utasítás jobbra forgatja (LSB-től MSB-ig) IN bitjeit N-szer, az eredmény az OUT-ba kerül.
  Ha EN=0, az utasítás nem hajtódik végre.

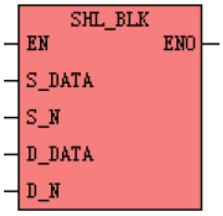
- **IL**
  IN és OUT adattípusának meg kell egyeznie.
  Ha CR=1, az utasítás jobbra forgatja (LSB-től MSB-ig) OUT bitjeit N-szer, az eredmény az OUT-ba kerül. A művelet nincs hatással CR értékére.
  Ha CR=0, az utasítás nem hajtódik végre.

LD		Ha M0.0=0, ROL utasítás nem hajtódik végre. Ha I0.0=1, akkor az utasítás eggyel jobbra forgatja QB0 bitjeit, az eredmény a QB0-ra kerül.
IL	LD %M0.0 (* CR=1, ha %M0.0 is 1 *) ROL %QB0, B#1 (* Ha CR=1, az utasítás QB0 bitjeit jobbra forgatja eggyel, és az*) (*eredmény QB0-ba kerül *) (* Ha CR=0, az utasítás nem kerül végrehajtásra *)	

Eredmény

QB0	B#2#10100001			
	1. végrehajtás	2. végrehajtás	3. végrehajtás	4. végrehajtás
QB0	B#2#11010000	B#2#01101000	B#2#00110100	B#2#00011010

6.6.5. SHL_BLK (bitlánc eltolása balra)

	Név	Használat	Csoport	
LD	SHL_BLK			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	SHL_BLK	SHL_BLK *S_DATA, S_N, D_DATA, D_N*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
S_DATA	Bemenet	BOOL	I, Q, M, V, L
S_N	Bemenet	INT	I, Q, M, V, L, SM, T, C, AI, AQ, konstans, mutató
D_DATA	Kimenet/Bemenet	BOOL	Q, M, V, L
D_N	Kimenet	INT	Q, M, V, L, SM, T, C, AI, AQ, konstans, mutató

Az utasítás a D_DATA címen kezdődő értéket, mely mérete D_N bit méretű, balra lépteti S_N bittel. A D_DATA cím legkisebb helyiértékű bitjétől S_N méretű bit kerül az S_DATA címről.

- **LD**

  Ha EN=1, az utasítás végrehajtódik.

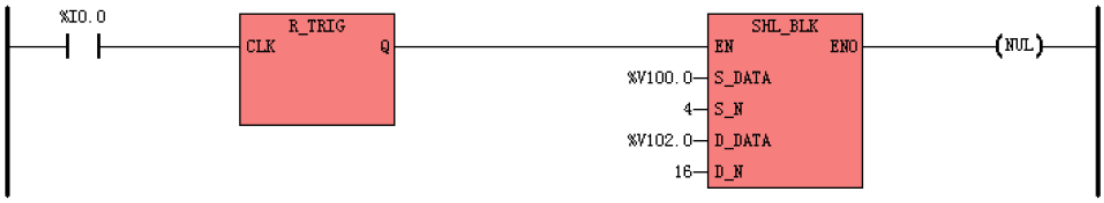
- **IL**

  Ha CR=1, az utasítás végrehajtódik, művelet nincs hatással CR értékére.

LD

(* Network 0 *)
(* Kezdeti értékek felvétele*)

(* Network 1 *)
(*léptetés I0.0 felfutó élére*)

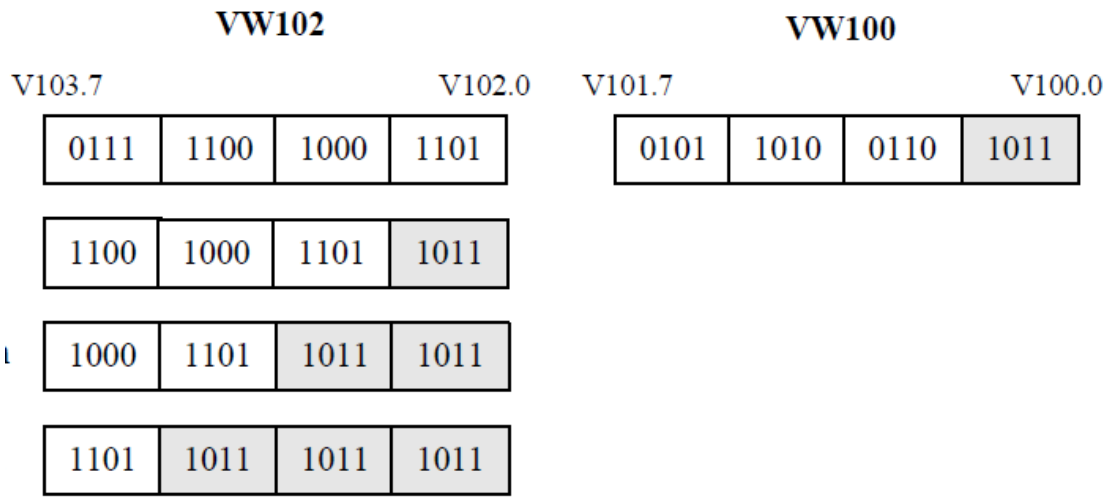


IL

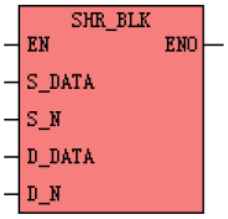
```
(* Network 0 *)
(*Kezdeti értékek felvétele*)
LD %SM0.1
MOVE 16#5A6B, %VW100
MOVE 16#7C8D, %VW102
(* Network 1 *)
(*léptetés I0.0 felfutó élére*)
LD %I0.0
R_TRIG
SHL_BLK %V100.0, 4, %V102.0, 16
```

Eredmény

A fenti példaprogram eredménye, minden egyes sor egy újabb végrehajtást jelent.



6.6.6. SHR_BLK (bitlánc eltolása jobbra)

	Név	Használat	Csoport	
LD	SHR_BLK			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	SHR_BLK	SHR_BLK *S_DATA, S_N, D_DATA, D_N*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
S_DATA	Bemenet	BOOL	I, Q, M, V, L
S_N	Bemenet	INT	I, Q, M, V, L, SM, T, C, AI, AQ, konstans, mutató
D_DATA	Kimenet/Bemenet	BOOL	Q, M, V, L
D_N	Kimenet	INT	Q, M, V, L, SM, T, C, AI, AQ, konstans, mutató

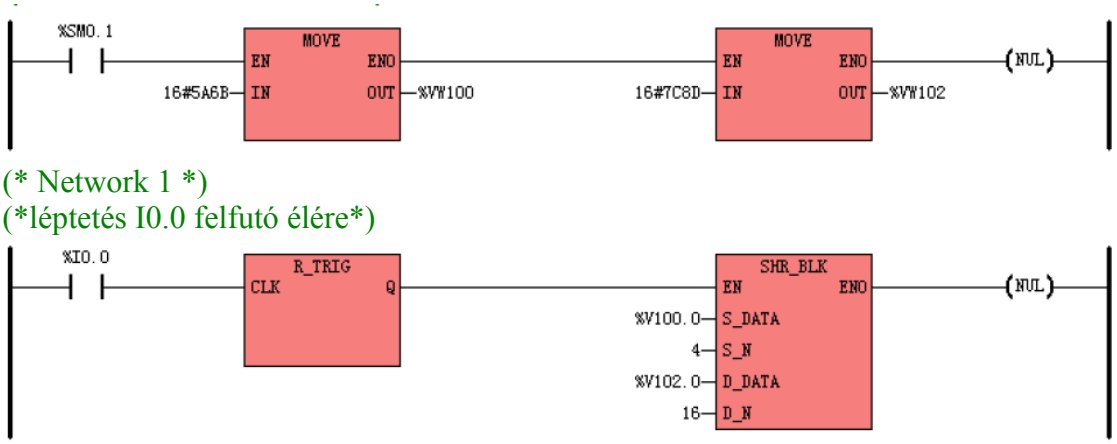
Az utasítás a D_DATA címen kezdődő értéket, mely mérete D_N bit méretű, jobbra lépteti S_N bittel. A D_DATA cím legkisebb helyiértékű bitjétől S_N méretű bit kerül az S_DATA címről.

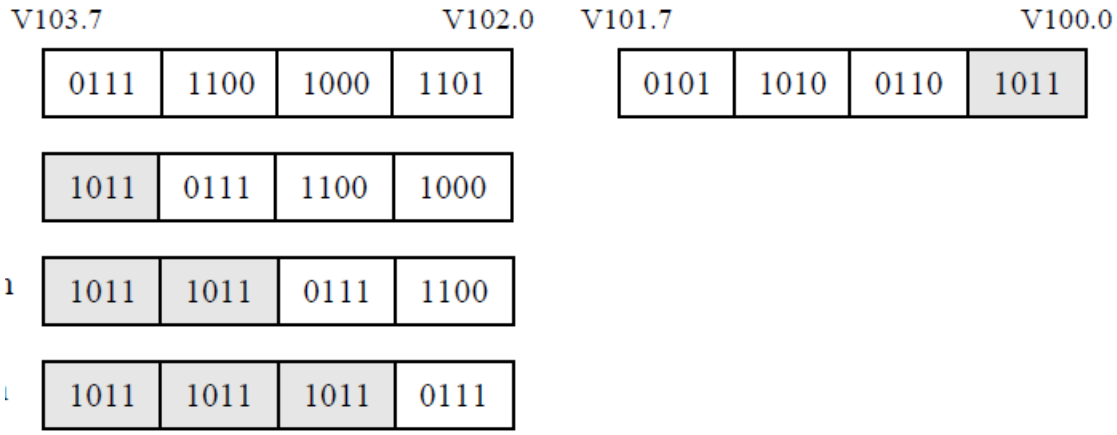
- **LD**

  Ha EN=1, az utasítás végrehajtódik.

- **IL**

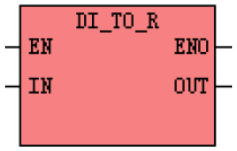
  Ha CR=1, az utasítás végrehajtódik, művelet nincs hatással CR értékére.

<table>
<tr><td>LD</td><td>(* Network 0 *)
(* Kezdeti értékek felvétele*)
 

(* Network 1 *)
(*léptetés I0.0 felfutó élére*)
</td></tr>
<tr><td>IL</td><td>(* Network 0 *)
(* Kezdeti értékek felvétele*)
LD %SM0.1
MOVE 16#5A6B, %VW100
MOVE 16#7C8D, %VW102
(* Network 1 *)
(*léptetés I0.0 felfutó élére*)
LD %I0.0
R_TRIG
SHR_BLK %V100.0, 4, %V102.0, 16</td></tr>
<tr><td>Eredmény</td><td>A fenti példaprogram eredménye, minden egyes sor egy újabb végrehajtást jelent.
 
</td></tr>
</table>

6.7 Konvertáló utasítások

6.7.1. DI_TO_R (DINT to REAL)

	Név	Használat	Csoport	
LD	DI_TO_R			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	DI_TO_R	DI_TO_R *IN, OUT*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	DINT	I, Q, M, V, L, SM, HC, konstans
OUT	Kimenet	REAL	V, L

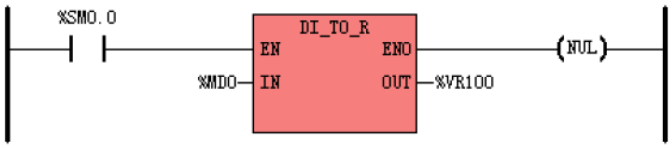
Ez az utasítás átkonvertálja a DINT típusú értéket (IN), REAL típussá, az eredmény az OUT-ra kerül.

- LD

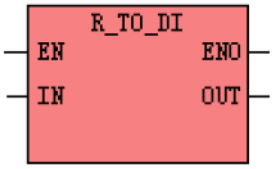
  Ha EN=1, az utasítás végrehajtásra kerül

- IL

  Ha CR=1, az utasítás végrehajtásra kerül, művelet nincs hatással CR értékére.

LD		Mivel SM0.0 értéke mindig 1, ezért DI_TO_R utasítás mindig végrehajtódik. Az MD0 címen található értéket REAL típussá alakítja és az eredmény VR100 címre kerül.
IL	LD %SM0.0 (* Mivel SM0.0 mindig 1, CR értéke is 1 lesz*) DI_TO_R %MD0, %VR100 (* Átkonvertálja az MD0 címen található számot REAL típussá, és az eredmény VR100 címre kerül. *)	
Eredmény	A fenti mintaprogram futásának eredménye: MD0 VR100 DI#123 123.0 DI#-9876 -9876.0	

6.7.2. R_TO_DI (REAL to DINT)

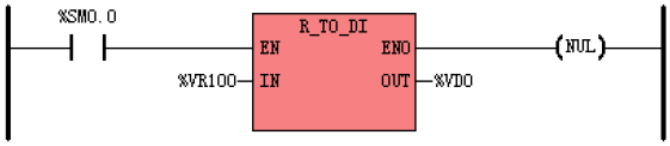
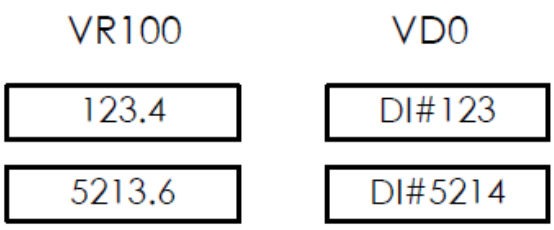
	Név	Használat	Csoport	
LD	R_TO_DI			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	R_TO_DI	R_TO_DI *IN, OUT*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	REAL	V, L, konstans
OUT	Kimenet	DINT	M, V, L, SM

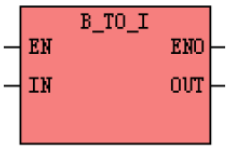
Ez az utasítás átkonvertálja a REAL típusú értéket (IN), DINT típussá, az eredmény az OUT-ra kerül. A konvertálás során a törtrészeket a program levágja.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, művelet nincs hatással CR értékére.

LD		Mivel SM0.0 értéke mindig 1, ezért R_TO_DI utasítás mindig végrehajtódik. A VR100 címen található REAL értéket DINT típussá alakítja és az eredmény VD0 címre kerül.
IL	LD %SM0.0 (* Mivel SM0.0 mindig 1, CR értéke is 1 lesz*) R_TO_DI %VR100, %VD0 (* A VR100 címen található REAL értéket DINT típussá alakítja és az eredmény VD0 címre kerül. *)	
Eredmény		

6.7.3. B_TO_I (BYTE to INT)

	Név	Használat	Csoport	
LD	B_TO_I			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	B_TO_I	B_TO_I *IN, OUT*	U	

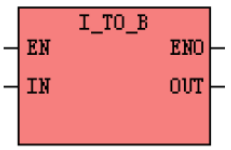
Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	BYTE	I, Q, M, V, L, SM, konstans
OUT	Kimenet	INT	Q, M, V, L, SM, AQ

Ez az utasítás átkonvertálja a BYTE típusú értéket (IN), INT típussá, az eredmény az OUT-ra kerül.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, művelet nincs hatással CR értékére.

6.7.4. I_TO_B (INT to BYTE)

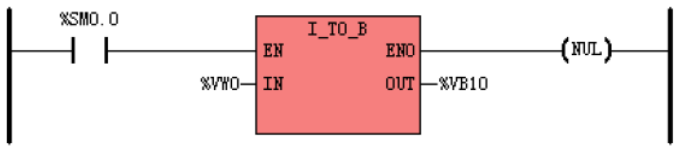
	Név	Használat	Csoport	
LD	I_TO_B			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	I_TO_B	I_TO_B *IN, OUT*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	INT	I, Q, M, V, L, SM, AI, AQ, T, C, konstans
OUT	Kimenet	BYTE	Q, M, V, L, SM

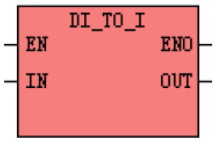
Ez az utasítás átkonvertálja az INT típusú értéket (IN), BYTE típussá, az eredmény az OUT-ra kerül.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, művelet nincs hatással CR értékére.

LD		Mivel SM0.0 értéke mindig 1, ezért I_TO_B utasítás mindig végrehajtódik. A VW0 címen található INT értéket BYTE típussá alakítja és az eredmény VB10 címre kerül.
IL	LD %SM0.0 I_TO_B %VW0, %VB10	
Eredmény	VW0 / VB10 24 / B#24 255 / B#255 I#16#FFFD / B#16#FD	

6.7.5 DI_TO_I (DINT to INT)

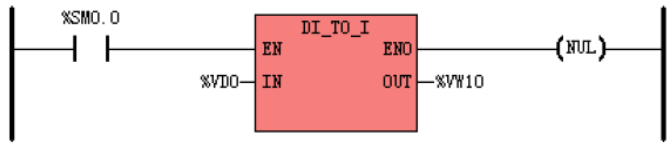
	Név	Használat	Csoport	
LD	DI_TO_I			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	DI_TO_I	DI_TO_I *IN, OUT*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	DINT	I, Q, M, V, L, SM, HC konstans
OUT	Kimenet	INT	Q, M, V, L, SM, AQ

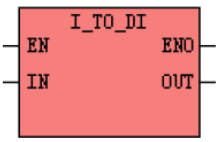
Ez az utasítás átkonvertálja az DINT típusú értéket (IN), INT típussá, az eredmény az OUT-ra kerül.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, művelet nincs hatással CR értékére.

LD		Mivel SM0.0 értéke mindig 1, ezért DI_TO_I utasítás mindig végrehajtódik. A VD0 címen található DI értéket I típussá alakítja és az eredmény VW10 címre kerül.
IL	LD %SM0.0 DI_TO_I %VD0, %VW10	
Eredmény	VD0 / VW10 DI#12345 / 12345 DI#-234 / -234 DI#16#7A8B9C1D / I#16#9C1D	

6.7.6. I_TO_DI (INT to DINT)

	Név	Használat	Csoport	
LD	I_TO_DI			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	I_TO_DI	I_TO_DI *IN, OUT*	U	

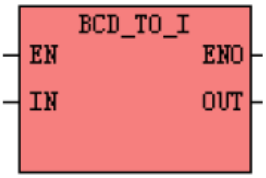
Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	INT	I, Q, M, V, L, SM, AI, AQ, T, C, konstans
OUT	Kimenet	DINT	Q, M, V, L, SM

Ez az utasítás átkonvertálja az INT típusú értéket (IN), DINT típussá, az eredmény az OUT-ra kerül.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, művelet nincs hatással CR értékére.

6.7.7. BCD_TO_I (BCD to INT)

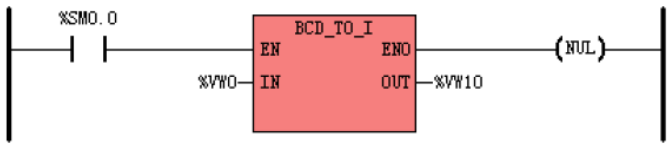
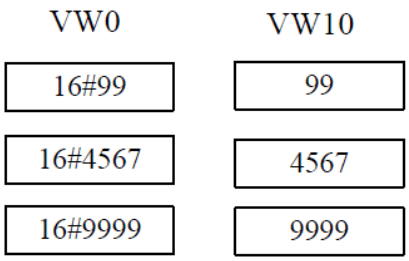
	Név	Használat	Csoport	
LD	BCD_TO_I			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	BCD_TO_I	BCD_TO_I *IN, OUT*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	WORD	I, Q, M, V, L, SM, konstans
OUT	Kimenet	INT	Q, M, V, L, SM, AQ

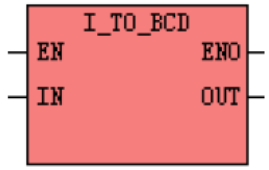
Ez az utasítás átkonvertálja az IN bemenetre érkező BCD kódot, INT típussá, az eredmény az OUT-ra kerül.
Megj.: A program 8421 BCD kódot támogatja. Az IN érvényességi tartománya 0 --- 9999 BCD.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, művelet nincs hatással CR értékére.

LD		Mivel SM0.0 értéke mindig 1, ezért BCD_TO_I utasítás mindig végrehajtódik. A VW0 címen található BCD kódot I típussá alakítja és az eredmény VW10 címre kerül.
IL	LD %SM0.0 BCD_TO_I %VW0, %VW10	
Eredmény		

6.7.8. I_TO_BCD (INT to BCD)

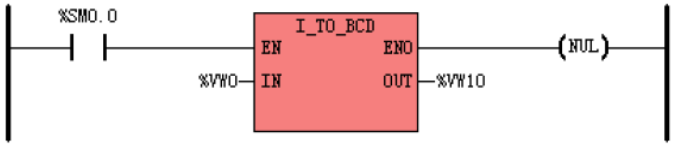
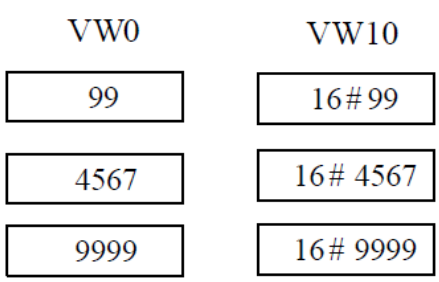
	Név	Használat	Csoport	
LD	I_TO_BCD			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	I_TO_BCD	I_TO_BCD *IN, OUT*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	INT	I, Q, M, V, L, SM, AI, AQ, T, C, konstans
OUT	Kimenet	WORD	Q, M, V, L, SM

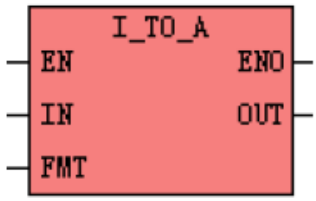
Ez az utasítás átkonvertálja az IN bemenetre érkező INT számot, BCD számmá, az eredmény az OUT-ra kerül.
Megj.: A program 8421 BCD kódot támogatja. Az IN érvényességi tartománya 0 --- 9999 BCD.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, művelet nincs hatással CR értékére.

LD		Mivel SM0.0 értéke mindig 1, ezért I_TO_BCD utasítás mindig végrehajtódik. A VW0 címen található INT értéket BCD típussá alakítja és az eredmény VW10 címre kerül.
IL	LD %SM0.0 I_TO_ BCD %VW0, %VW10	
Eredmény		

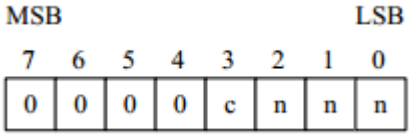
6.7.9. I_TO_A (INT to ASCII)

	Név	Használat	Csoport	
LD	I_TO_A			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	I_TO_A	I_TO_A *IN, OUT, FMT*	U	

Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
IN	Bemenet	INT	I, Q, M, V, L, SM, AI, AQ, T, C, konstans
FMT	Bemenet	BYTE	I, Q, M, V, L, SM
OUT	Kimenet	BYTE	Q, M, V, L, SM

Ez az utasítás átkonvertálja az INT típusú egész számot, ASCII kóddá, a formátumot az FMT bemenet határozza meg, és az eredményt a kimeneti pufferbe helyezi az OUT címtől kezdődően. Ha a konvertálás eredménye negatív akkor azt egy mínusz jel (-) fogja jelezni. A pufferbe az értékek jobbra vannak csoportosítva és a szabad byte-okat szóközzel tölti ki (ASCII kód: 32).

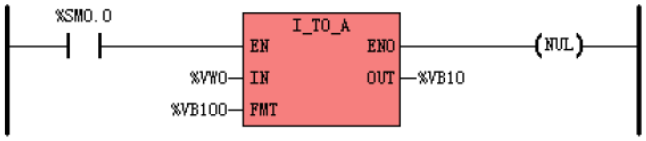
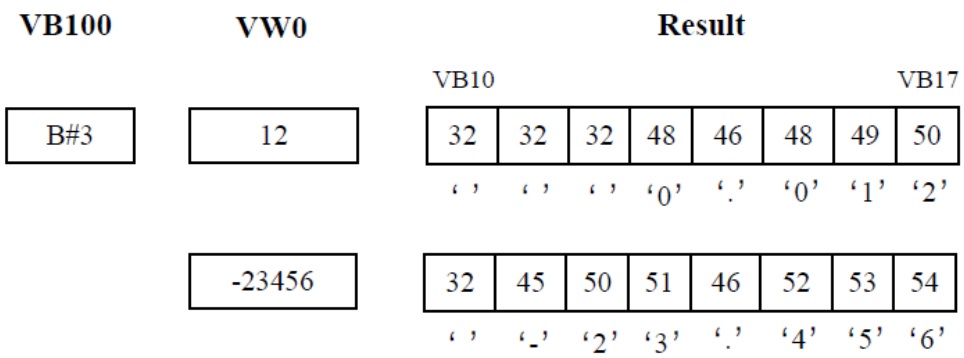
Az FMT bemenet meghatározza az ASCII kód formátumát.



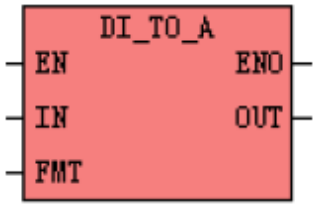
1) nnn → A tizedes jegy utáni számok, értéke 0 - 5 között lehet, 0 esetén nincs tizedes jegy.
2) c → Ez jelzi, hogy mi legyen a tizedes elválasztó jel, 0 esetén pont az elválasztó (ASCII 46), egy esetén pedig vessző (ASCII 44).
3) A felső négy bitnek nullának kell lennie.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, művelet nincs hatással CR értékére

LD		Mivel SM0.0 értéke mindig 1, ezért I_TO_A utasítás mindig végrehajtódik. A VW0 címen található INT értéket ASCII szöveggé alakítja, az eredmény a VB10 címtől kezdődően kerül a memóriába.
IL	LD %SM0.0 I_TO_A %VW0, %VB10, %VB100	
Eredmény		

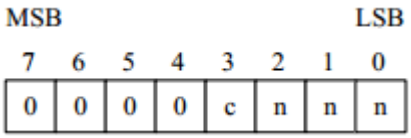
6.7.10. DI_TO_A (DINT to ASCII)

	Név	Használat	Csoport	
LD	DI_TO_A			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	DI_TO_A	DI_TO_A *IN, OUT, FMT*	U	

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	DINT	I, Q, M, V, L, SM, HC, konstans
FMT	Bemenet	BYTE	I, Q, M, V, L, SM
OUT	Kimenet	BYTE	Q, M, V, L, SM

Ez az utasítás átkonvertálja a DINT értéket, ASCII kóddá, és az eredményt a kimeneti pufferbe helyezi az OUT címtől kezdődően. Ha a konvertálás eredménye negatív akkor azt egy mínusz jel (-) fogja jelezni. A pufferbe az értékek jobbra vannak csoportosítva és a szabad byte-okat szóközzel tölti ki (ASCII 32).

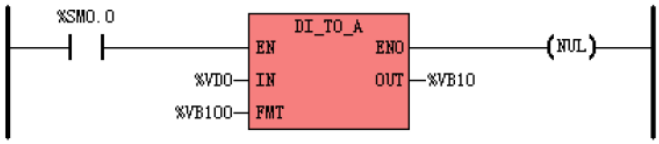
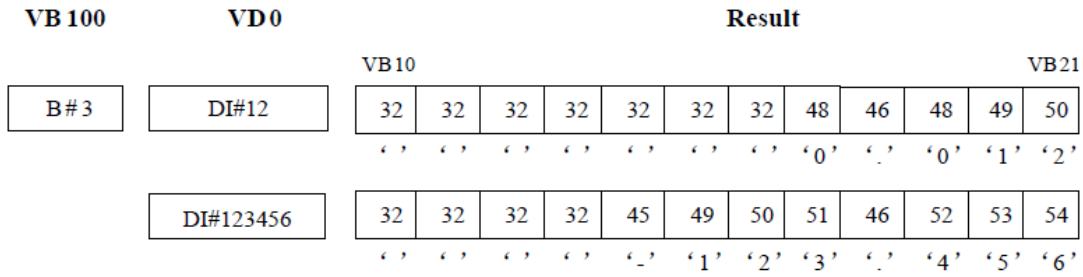
Az FMT bemenet meghatározza az ASCII kód formátumát.



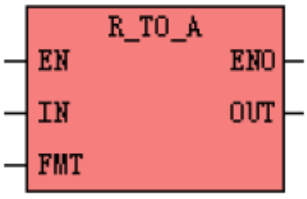
1.) nnn → A tizedes jegy utáni számok, értéke 0 - 5 között lehet, 0 esetén nincs tizedes jegy.
2.) c → Ez jelzi, hogy mi legyen a tizedes elválasztó jel, 0 esetén pont az elválasztó (ASCII 46), egy esetén pedig vessző (ASCII 44).
3.) A felső négy bitnek nullának kell lennie.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, művelet nincs hatással CR értékére

LD		Mivel SM0.0 értéke mindig 1, ezért DI_TO_A utasítás mindig végrehajtódik. A VD0 címen található INT értéket ASCII szöveggé alakítja, az eredmény a VB10 címtől kezdődően kerül a memóriába.
IL	LD %SM0.0 DI_TO_A %VD0, %VB10, %VB100	
Eredmény		

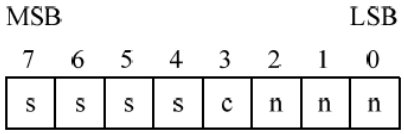
6.7.11. R_TO_A (REAL to ASCII)

	Név	Használat	Csoport	
LD	R_TO_A			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	R_TO_A	R_TO_A *IN, OUT, FMT*	U	

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	REAL	V, L, konstans
FMT	Bemenet	BYTE	I, Q, M, V, L, SM
OUT	Kimenet	BYTE	Q, M, V, L, SM

Ez az utasítás átkonvertálja az REAL értéket, ASCII kóddá, és az eredményt a kimeneti pufferbe helyezi az OUT címtől kezdődően. Ha a konvertálás eredménye negatív, akkor azt egy mínusz jel (-) fogja jelezni. Ha az IN bemenet decimális része nagyobb, az *nnn* az FMT-ben (mely a tizedesjegy utáni számok számát adja meg) akkor az értéket a program konvertálás előtt kerekíti. A pufferbe az értékek jobbra vannak csoportosítva és a szabad byte-okat szóközzel tölti ki (ASCII 32).

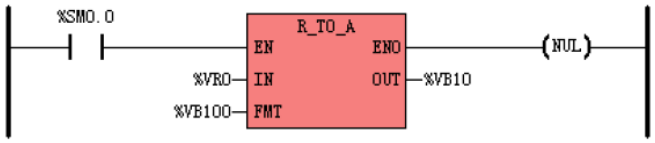
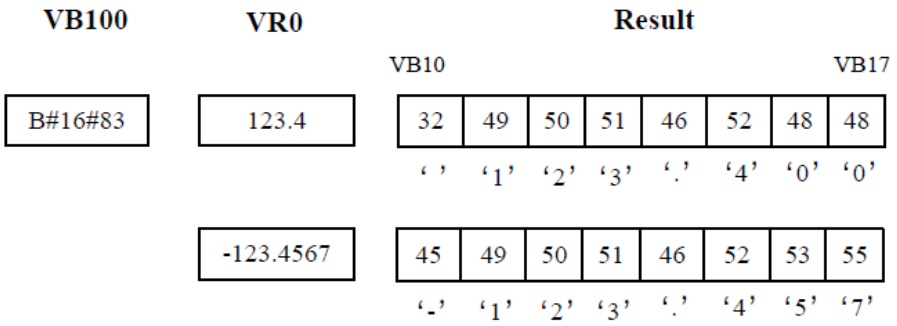
Az FMT határozza meg az ASCII kód formátumát, valamint a kimeneti puffer méretét.



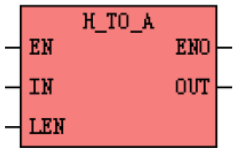
1.) nnn → A tizedes jegy utáni számok, értéke 0 - 5 között lehet, 0 esetén nincs tizedes jegy.
2.) c → Ez jelzi, hogy mi legyen a tizedes elválasztó jel, 0 esetén pont az elválasztó (ASCII 46), egy esetén pedig vessző (ASCII 44).
3.) ssss → A kimeneti puffer méretét adja meg, értéke 3 – 15 között lehet, és nagyobbnak kell lennie mint *nnn* -nek.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül

- IL
  Ha CR=1, az utasítást végrehajtja, és nincs hatással a CR-re.

LD		Mivel SM0.0 értéke mindig 1, ezért R_TO_A utasítás mindig végrehajtódik. A VR0 címen található REAL értéket ASCII szöveggé alakítja, az eredmény a VB10 címtől kezdődően kerül a memóriába.
IL	LD %SM0.0 R_TO_A %VR0, %VB10, %VB100	
Eredmény		

6.7.12. H_TO_A (Hexadecimal to ASCII)

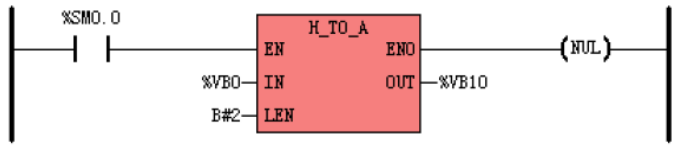
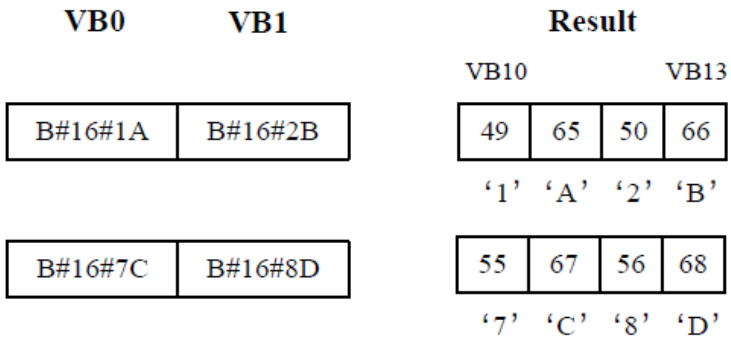
	Név	Használat	Csoport	
LD	H_TO_A			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	H_TO_A	R_TO_A *IN, OUT, LEN*	U	

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	BYTE	I, Q, M, V, L, konstans
FMT	Bemenet	BYTE	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE	Q, M, V, L, SM

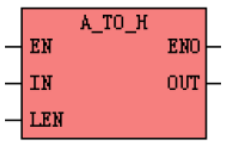
Ez az utasítás LEN számú hexadecimális számjegyet, az IN címtől kezdődően, átkonvertálja ASCII kóddá, az eredményt a kimeneti pufferbe menti az OUT címtől kezdődően.
Megjegyzés: Minden 4 bináris számjegy alkot 1 hexadecimális számjegyet, így ha minden bemeneti byte két hexadecimális számjegyet tartalmaz, akkor a kimeneti puffer mérete LEN*2 byte lesz.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül

- IL
  Ha CR=1, az utasítást végrehajtja, és nincs hatással a CR-re.

LD		Mivel SM0.0 értéke mindig 1, ezért H_TO_A utasítás mindig végrehajtódik. A VB0 címen található hexadecimális számot ASCII szöveggé alakítja, az eredmény a VB10 címtől kezdődően kerül a memóriába.
IL	LD %SM0.0 H_TO_A %VB0, %VB10, B#2	
Eredmény		

6.7.13. A_TO_H (ASCII to Hexadecimal)

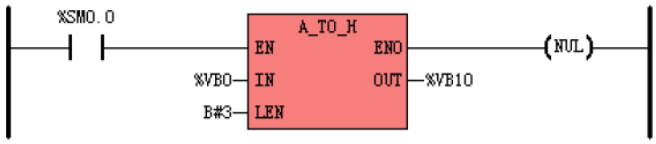
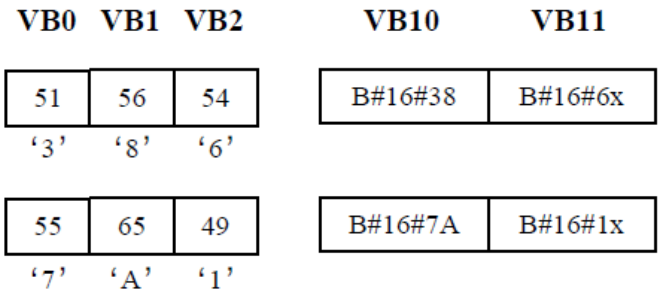
	Név	Használat	Csoport	
LD	A_TO_H			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	A_TO_H	A_TO _H *IN, OUT, LEN*	U	

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	BYTE	I, Q, M, V, L, SM
LEN	Bemenet	BYTE	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE	Q, M, V, L, SM

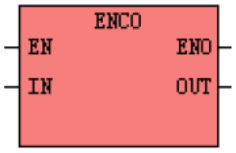
Ez az utasítás LEN számú ASCII karaktert, az IN címtől kezdődően, átkonvertálja hexadecimális számmá, az eredményt a kimeneti pufferbe helyezi az OUT címtől kezdődően.
Megjegyzés: Minden 4 bináris számjegy alkot 1 hexadecimális számjegyet, tehát, minden bájt a bemeneten, ami ASCII karakterből áll 4 bináris számjegyet foglal a memóriából a kimeneti pufferben.
Az ASCII bemeneti tartomány: B#16#30 és B#16#39 között (karakterek 0 → 9 ig), és B#16#41 és B#16#46 között (karakterek A → F ig).

- LD
  Ha EN=1, az utasítás végrehajtásra kerül

- IL
  Ha CR=1, az utasítást végrehajtja, és nincs hatással a CR-re.

<table>
<tr><td>LD</td><td> 
</td><td>Mivel SM0.0 értéke mindig 1, ezért A_TO_H utasítás mindig végrehajtódik. A VB0 címen található, 3 byte méretű ASCII szöveget hexadecimális számmá alakítja, az eredmény a VB10 címtől kezdődően kerül a memóriába.</td></tr>
<tr><td>IL</td><td colspan="2">LD %SM0.0
A_TO_H %VB0, %VB10, B#3</td></tr>
<tr><td>Eredmény</td><td colspan="2"> 
</td></tr>
</table>

6.7.14. ENCO (Kódoló)

	Név	Használat	Csoport	
LD	ENCO			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	ENCO	ENCO *IN, OUT*	U	

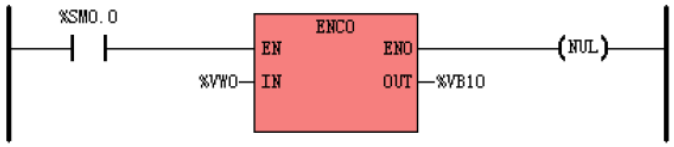
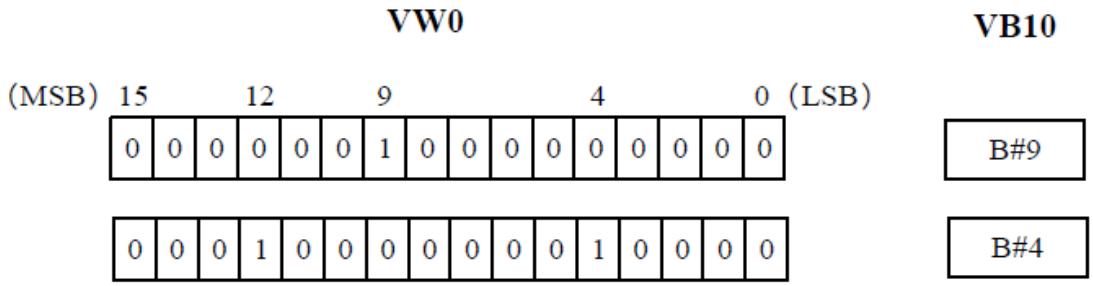
Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	WORD	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE	Q, M, V, L, SM

Az utasítás megvizsgálja a bemeneti WORD értékét, a legkisebb helyi értékű bittől kezdődően, és a kimeneten megjelenik, hogy a bemeneti szó hányadik bitje tartalmaz először 1-et.

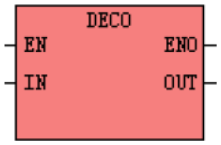
Megjegyzés: Ha IN=0 az eredmény értelmezhetetlen.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül

- IL
  Ha CR=1, az utasítást végrehajtja, és nincs hatással a CR-re.

LD		Mivel SM0.0 értéke mindig 1, ezért ENCO utasítás mindig végrehajtódik. A VB10 kimenetre kerül, hogy a bemenet VW0 hányadik helyiértékén van először 1.
IL	LD %SM0.0 ENCO %VW0, %VB10	
Eredmény		

6.7.15. DECO (Dekódoló)

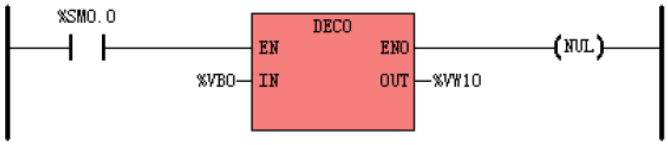
	Név	Használat	Csoport	
LD	DECO			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	DECO	DECO *IN, OUT*	U	

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	BYTE	I, Q, M, V, L, SM, konstans
OUT	Kimenet	WORD	Q, M, V, L, SM

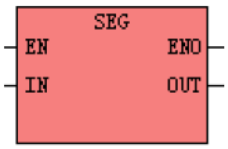
Az utasítás a kimeneti WORD, IN bemeneten megadott bitjét 1-be állítja. A kimenet összes többi bitje törlődik.

- LD
  Ha EN=1, az utasítás végrehajtódik

- IL
  Ha CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

LD		Mivel SM0.0 értéke mindig 1, ezért DECO utasítás mindig végrehajtódik. A VW10 kimeneti memória, VB bemeneten megadott bitjét a program 1-be állítja.
IL	LD %SM0.0 DECO %VB0, %VW10	
Eredmény	**VB0** **VW10** (MSB) 15 9 4 0 (LSB) B#9 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 B#16#D4 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0	

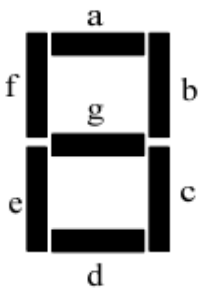
6.7.16. SEG (7-szegmenses kijelző)

	Név	Használat	Csoport	
LD	SEG			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	SEG	SEG *IN, OUT*	U	

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	BYTE	I, Q, M, V, L, SM, konstans
OUT	Kimenet	BYTE	Q, M, V, L, SM

Az utasítás hétszegmenses kijelző vezérlésére alkalmas bit mintát állít elő, a bemeneti változó legkisebb 4 bitje alapján.

IN (LSD)	Display	*OUT* (- g f e d c b a)
0	0	0 0 1 1 1 1 1 1
1	1	0 0 0 0 0 1 1 0
2	2	0 1 0 1 1 0 1 1
3	3	0 1 0 0 1 1 1 1
4	4	0 1 1 0 0 1 1 0
5	5	0 1 1 0 1 1 0 1
6	6	0 1 1 1 1 1 0 1
7	7	0 0 0 0 0 1 1 1

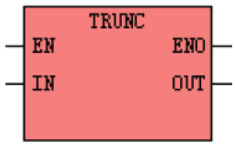


IN (LSD)	Display	*OUT* (- g f e d c b a)
8	8	0 1 1 1 1 1 1 1
9	9	0 1 1 0 0 1 1 1
A	A	0 1 1 1 0 1 1 1
B	B	0 1 1 1 1 1 0 0
C	C	0 0 1 1 1 0 0 1
D	D	0 1 0 1 1 1 1 0
E	E	0 1 1 1 1 0 0 1
F	F	0 1 1 1 0 0 0 1

- LD
  Ha EN=1, az utasítás végrehajtódik

- IL
  Ha CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

6.7.17. TRUNC (csonkítás)

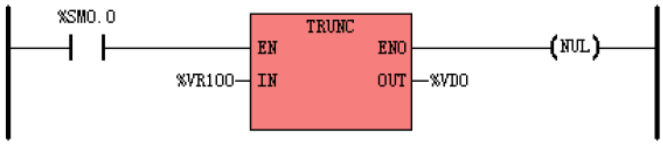
	Név	Használat	Csoport	
LD	TRUNC			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	TRUNC	TRUNC *IN, OUT*	U	

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	REAL	V, L, konstans
OUT	Kimenet	BYTE	M, V, L, SM

Ez az utasítás az IN bemeneten REAL értéket konvertálja DINT értékké, az eredmény az OUT kimenetre kerül, az IN tizedes részét pedig levágja.

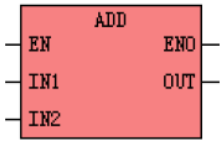
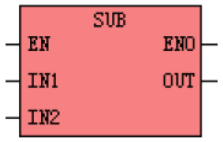
- LD
  Ha EN=1, az utasítás végrehajtódik

- IL
  Ha CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

LD		Mivel SM0.0 értéke mindig 1, ezért a TRUNC utasítás mindig végrehajtódik. A VR100 értéken található REAL értéket DINT számmá alakítja. A tizedes részeket a program levágja.
IL	LD %SM0.0 TRUNC %VR100, %VD0	
Eredmény	VR100 / VD0 123.4 / DI#123 5213.6 / DI#5213	

6.8. Számokkal végezhető műveletek

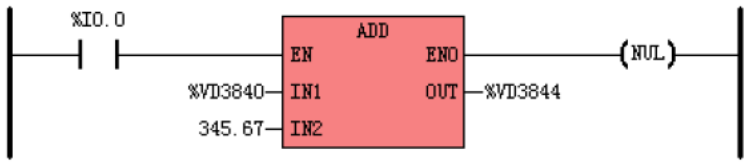
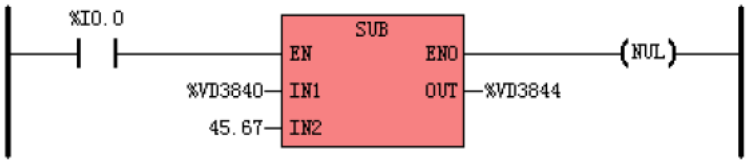
6.8.1. ADD és SUB (összeadás, kivonás)

	Név	Használat	Csoport	
LD	ADD			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	SUB			
IL	ADD	ADD *IN, OUT*	U	
	SUB	SUB *IN, OUT*		

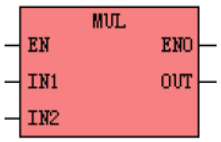
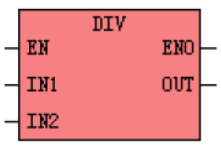
Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN1	Bemenet	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, konstans
IN2	Bemenet	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, konstans
OUT	Kimenet	INT, DINT, REAL	Q, AQ, M, V, L, SM

- **LD**
  IN1, IN2, és OUT adattípusának meg kell egyeznie.
  Ha EN=1, az ADD utasítás a következő műveletet hajtja végre: OUT=IN1+IN2, a SUB utasítás pedig: OUT=IN1-IN2.

- **IL**
  IN1 és OUT adattípusának meg kell egyeznie.
  Ha CR=1, az ADD utasítás a következő műveletet hajtja végre: OUT=OUT+IN1, a SUB utasítás pedig: OUT=OUT-IN1.

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>Ha I0.0=0, ADD utasítás nem kerül végrehajtásra.
Amennyiben 1, akkor VD3840 számhoz hozzáad 345,67-et, és az eredmény a VD3844-re kerül.</td></tr>
<tr><td> 
</td><td>Ha I0.0=0, SUB utasítás nem kerül végrehajtásra.
Amennyiben 1, akkor VD3840 számból kivon 45,67-et, és az eredmény a VD3844-re kerül.</td></tr>
<tr><td rowspan="2">IL</td><td>LD %I0.0
ADD 345.67, %VD3840</td><td>(* CR létrehozása I0.0 bemenettel *)
(* Ha CR=1: VD3840 = VD3840 +245.67 *)
(* Ha CR=0: utasítás nem kerül végrehajtásra*)</td></tr>
<tr><td>LD %I0.0
SUB 45.67, %VD3840</td><td>(* CR létrehozása I0.0 bemenettel *)
(* Ha CR=1: VD3840 = VD3840 - 45.67 *)
(* Ha CR=0: utasítás nem kerül végrehajtásra*)</td></tr>
</table>

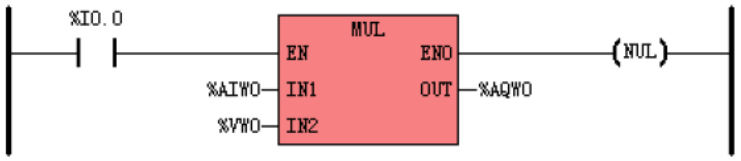
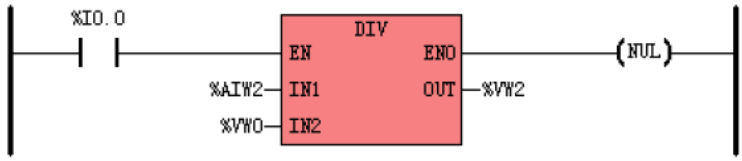
6.8.2. MUL és DIV (szorzás, osztás)

<table>
<tr><td rowspan="4">LD</td><th>Név</th><th>Használat</th><th>Csoport</th><td rowspan="4">☑ CPU304
☑ CPU304EX
☑ CPU306
☑ CPU306EX
☑ CPU308</td></tr>
<tr><td>MUL</td><td> 
</td><td></td></tr>
<tr><td>DIV</td><td> 
</td><td></td></tr>
<tr><td rowspan="2">IL</td><td>MUL</td><td>MUL IN, OUT</td><td rowspan="2">U</td></tr>
<tr><td>DIV</td><td>DIV IN, OUT</td></tr>
</table>

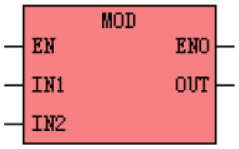
Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN1	Bemenet	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, konstans
IN2	Bemenet	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, konstans
OUT	Kimenet	INT, DINT, REAL	Q, AQ, M, V, L, SM

- LD
  IN1, IN2, és OUT adattípusának meg kell egyeznie.
  Ha EN=1, a MUL utasítás a következő műveletet hajtja végre: OUT=IN1*IN2, a DIV utasítás pedig: OUT=IN1/IN2.

- IL
  IN1 és OUT adattípusának meg kell egyeznie.
  Ha CR=1, a MUL utasítás a következő műveletet hajtja végre: OUT=OUT*IN1, a DIV utasítás pedig: OUT=OUT/IN1.
  A MUL és a DIV utasítások nincsenek hatással CR-re.

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>Ha I0.0=0, MUL utasítás nem kerül végrehajtásra.
Amennyiben 1, akkor AIW0 értékét megszorozza VW0-val, és az eredmény AQW0-ra kerül.</td></tr>
<tr><td> 
</td><td>Ha I0.0=0, DIV utasítás nem kerül végrehajtásra.
Amennyiben 1, akkor AIW2 értékét elosztja VW0-val, és az eredmény VW2-re kerül.</td></tr>
<tr><td rowspan="2">IL</td><td colspan="2">LD %I0.0 (* CR létrehozása I0.0 bemenettel *)
MUL %AIW0, %VW0 (* Ha CR=1: VW0 = VW0 × AIW0 *)
(* Ha CR=0: utasítás nem kerül végrehajtásra*)</td></tr>
<tr><td colspan="2">LD %I0.0 (* CR létrehozása I0.0 bemenettel *)
DIV %AIW2, %VW0 (* Ha CR=1: VW0 = VW0 ÷ AIW2 *)
(* Ha CR=0: utasítás nem kerül végrehajtásra*)</td></tr>
</table>

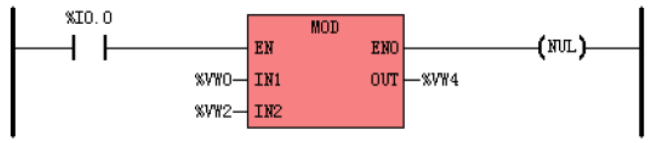
6.8.3. MOD (maradék képzés)

	Név	Használat	Csoport	
LD	MOD			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	MOD	MOD *IN, OUT*	U	

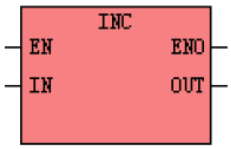
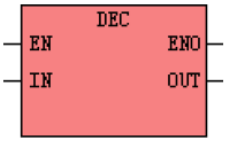
Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN1	Bemenet	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, konstans
IN2	Bemenet	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, konstans
OUT	Kimenet	INT, DINT, REAL	Q, AQ, M, V, L, SM

- LD
  IN1, IN2, és OUT adattípusának meg kell egyeznie.
  Ha EN=1, az utasítás elosztja IN1-et IN2 vel, és a maradék az OUT-ra kerül.

- IL
  IN1 és OUT adattípusának meg kell egyeznie.
  Ha CR=1, az utasítás elosztja az OUT-ot az IN1-el és maradék az OUT kimenetre kerül.
  A MUL és a DIV utasítások nincsenek hatással CR-re.

LD		Ha I0.0=0, MOD utasítás nem kerül végrehajtásra. Amennyiben 1, akkor VW0 értékét elosztja VW2-vel, a VW4 kimenetre az osztás maradéka kerül.
IL	LD %I0.0 (* CR létrehozása I0.0 bemenettel *) MOD %VW0, %VW4 (* Ha CR=1: VW4/VW0, maradéka VW4-re kerül*) (* Ha CR=0: utasítás nem kerül végrehajtásra*)	
Eredmény	A fenti LD minta futásának az eredménye: Address VW0 VW2 Value 8 3 Address VW4 Value 2	

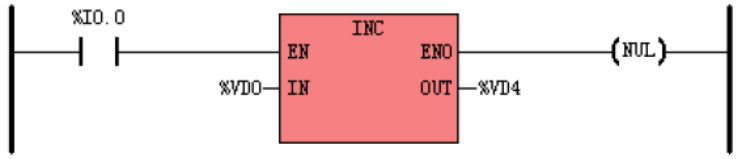
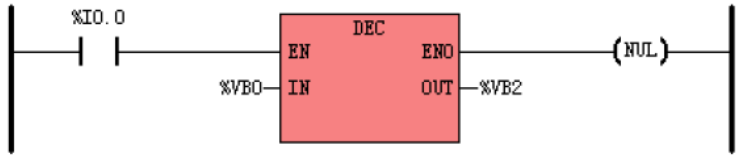
6.8.4. INC és DEC (növelés, csökkentés)

<table>
<tr><th></th><th>Név</th><th>Használat</th><th>Csoport</th><td rowspan="4">☑ CPU304
☑ CPU304EX
☑ CPU306
☑ CPU306EX
☑ CPU308</td></tr>
<tr><td rowspan="2">LD</td><td>INC</td><td> 
</td><td></td></tr>
<tr><td>DEC</td><td> 
</td><td></td></tr>
<tr><td rowspan="2">IL</td><td>INC</td><td>INC OUT</td><td rowspan="2">U</td></tr>
<tr><td>DEC</td><td>DEC OUT</td></tr>
</table>

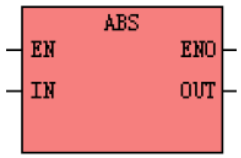
Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	BYTE, INT, DINT,	I, Q, AI, AQ, M, V, L, SM, T, C, HC, konstans
OUT	Kimenet	BYTE, INT, DINT,	Q, AQ, M, V, L, SM

- **LD**
  IN és OUT adattípusának meg kell egyeznie.
  Ha EN=1, az INC utasítás a következő műveletet hajt végre: OUT=IN+1, a DEC utasítás pedig: OUT=IN-1.

- **IL**
  Ha CR=1, az INC utasítás a következő műveletet hajt végre: OUT=OUT+1, a DEC utasítás pedig: OUT=OUT-1. Nincsenek hatással CR-re.

LD		Ha I0.0=0, INC utasítás nem kerül végrehajtásra. Amennyiben 1, akkor VD4 = VD0+1
		Ha I0.0=0, DEC utasítás nem kerül végrehajtásra. Amennyiben 1, akkor VB2 = VB0-1
IL	LD %I0.0 INC %VD4	(* CR létrehozása I0.0 bemenettel *) (* Ha CR=1: VD4 =VD4 + 1 *) (* Ha CR=0: utasítás nem kerül végrehajtásra*)
	LD %I0.0 DEC %VB2	(* CR létrehozása I0.0 bemenettel *) (* Ha CR=1: VB2 = VB2 - 1 *) (* Ha CR=0: utasítás nem kerül végrehajtásra*)

6.8.5. ABS (Abszolút érték)

	Név	Használat	Csoport	
LD	ABS			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	ABS	ABS *IN, OUT*	U	

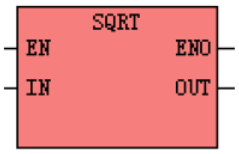
Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, konstans
OUT	Kimenet	INT, DINT, REAL	Q, AQ, M, V, L, SM, mutató

Az utasítás az IN bemenet abszolút értékét kiszámolja, az eredmény az OUT-ra kerül.
IN és OUT adattípusának meg kell egyeznie.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül.

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, és nincs hatással CR-re.

6.8.6. SQRT (négyzetgyök vonás)

	Név	Használat	Csoport	
LD	SQRT			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	SQRT	SQRT *IN, OUT*	U	

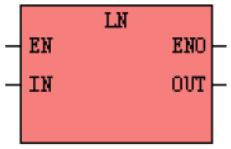
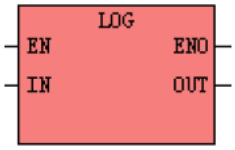
Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	REAL	V, L, konstans, mutató
OUT	Kimenet	REAL	V, L, mutató

Az utasítás az IN bemenetből négyzetgyökét kiszámolja és az eredmény az OUT-ra kerül.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül.

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, és nincs hatással CR-re.

6.8.7. LN, LOG (természetes és tízes alapú logaritmus)

	Név	Használat	Csoport	
LD	LN			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
	LOG			
IL	LN	LN *IN, OUT*	U	
	LOG	LOG *IN, OUT*		

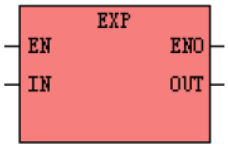
Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	REAL	V, L, konstans, mutató
OUT	Kimenet	REAL	V, L, mutató

Az LN utasítás az IN bemenet természetes logaritmusát számolja ki, az eredmény az OUT-ra kerül, a következő képlat alapján: $OUT=\log_e(IN)$.
A LOG utasítás kiszámolja tízes alapú logaritmusát az IN bemenetnek, az eredmény az OUT-ra kerül, a következő képlat alapján: $OUT=\log_{10}(IN)$.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül.

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, és nincs hatással CR-re.

6.8.8. EXP

	Név	Használat	Csoport	
LD	EXP			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	EXP	EXP *IN, OUT*	U	

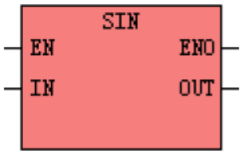
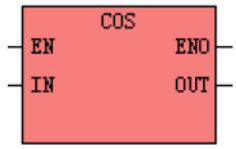
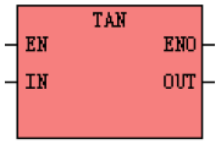
Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	REAL	V, L, konstans, mutató
OUT	Kimenet	REAL	V, L, mutató

Ez az utasítás kiszámolja az exponenciális hatványát az IN bemenetnek, az eredmény az OUT kimenetre kerül, a következő képlet alapján: $OUT=e^{IN}$.

- LD
  Ha EN=1, az utasítás végrehajtásra kerül.

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, és nincs hatással CR-re.

6.8.9. SIN, COS, TAN

	Név	Használat	Csoport	
LD	SIN			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
	COS			
	TAN			
IL	SIN	SIN *IN, OUT*	U	
	COS	COS *IN, OUT*		
	TAN	TAN *IN, OUT*		

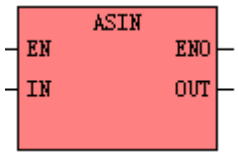
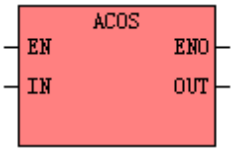
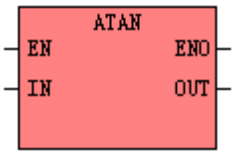
Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	REAL	V, L, konstans, mutató
OUT	Kimenet	REAL	V, L, mutató

A SIN utasítás kiszámolja az IN bemeneti érték szinuszát, az eredmény az OUT kimenetre kerül a következő képlet szerint: OUT=SIN(IN).
A COS utasítás kiszámolja az IN bemeneti érték koszinuszát, az eredmény az OUT kimenetre kerül a következő képlet szerint: OUT=COS(IN).
A TAN utasítás kiszámolja az IN bemeneti érték tangensét, az eredmény az OUT kimenetre kerül a következő képlet szerint: OUT=TAN(IN).

- LD
  Ha EN=1, az utasítás végrehajtásra kerül.

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, és nincs hatással CR-re.

6.8.10. ASIN (arc-sin), ACOS(arc-cos), ATAN(arc-tan)

	Név	Használat	Csoport	
LD	ASIN			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
	ACOS			
	ATAN			
IL	ASIN	ASIN *IN, OUT*	U	
	ACOS	ACOS *IN, OUT*		
	ATAN	ATAN *IN, OUT*		

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	REAL	V, L, konstans, mutató
OUT	Kimenet	REAL	V, L, mutató

A ASIN utasítás kiszámolja az IN bemeneti érték arc-szinuszát, az eredmény az OUT kimenetre kerül a következő képlet szerint: OUT=ARCSIN(IN).
A ACOS utasítás kiszámolja az IN bemeneti érték arc-koszinuszát, az eredmény az OUT kimenetre kerül a következő képlet szerint: OUT=ARCCOS(IN).
A ATAN utasítás kiszámolja az IN bemeneti érték arc-tangensét, az eredmény az OUT kimenetre kerül a következő képlet szerint: OUT=ARCTAN(IN).

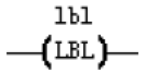
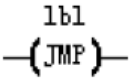
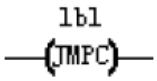
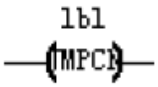
- LD
  Ha EN=1, az utasítás végrehajtásra kerül.

- IL
  Ha CR=1, az utasítás végrehajtásra kerül, és nincs hatással CR-re.

6.9. Programvezérlő utasítások

Az IL programnyelv esetén, az ugrás (jump) és a visszatérés (return) utasítás nincs hatással CR-re, tehát CR értéke változatlan marad, erre ügyelni kell a programozás során.

6.9.1. LBL és JMP utasítások

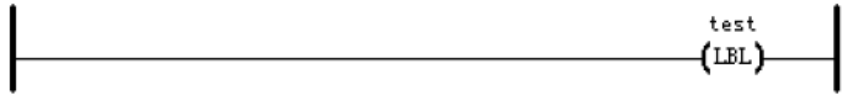
	Név	Használat	Csoport	
LD	LBL			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	JMP			
	JMPC			
	JMPCN			
IL	LBL	*lbl:*	U	
	JMP	JMP *lbl*		
	JMPC	JMPC *lbl*		
	JMPCN	JMPCN *lbl*		

Operandus	Leírás
lbl	Érvényes azonosító

- **LD**
  A LBL utasítás címke definiálására használatható, mellyel az ugrás utasítás célja határozható meg. A címke azonosítójának újradefiniálása tilos. Ez az utasítás feltétel nélkül végrehajtódik, tehát nincs szükség semmilyen feltételre a bal oldalán. A KincoBuilder program az LBL sorában levő összes utasítást figyelmen kívül hagyja.
  A **JMP** utasítás feltétel nélküli ugrást hajt végre a programban az lbl címkével megjelölt programrészhez.
  A **JMPC** utasítás feltételes ugrást hajt végre a programban az lbl címkével megjelölt programrészhez, ha a tőle balra lévő logikai értékek igaz állapotba kerülnek.
  A **JMPCN** utasítás feltételes ugrást hajt végre a programban az lbl címkével megjelölt programrészhez, ha a tőle balra lévő logikai értékek hamis állapotba kerülnek.

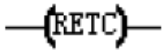
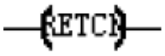
  Az ugrás utasítás és a hozzá tartozó cél címkéje ugyanazon POU-n belül kell lennie.

- **IL**
  A címke definíciójának új sorban kell lennie, a címke azonosítójának újradefiniálása tilos.
  A JMP utasítás feltétel nélküli ugrást hajt végre a programban az lbl címkével megjelölt networkbe.
  A **JMP** utasítás feltétel nélküli ugrást hajt végre a programban az lbl címkével megjelölt programrészhez.
  A **JMPC** utasítás feltételes ugrást hajt végre a programban az lbl címkével megjelölt programrészhez, ha CR=1
  A **JMPCN** utasítás feltételes ugrást hajt végre a programban az lbl címkével megjelölt programrészhez, ha CR=0
  Az ugrás utasítás és a hozzá tartozó cél címkéje ugyanazon POU-n belül kell lennie.

LD	IL
(* Network 0: *) 	(* NETWORK 0 *) test**:** ...
(* Network 4: *) 	(* NETWORK 4 *) LD %I0.0 JMPC test

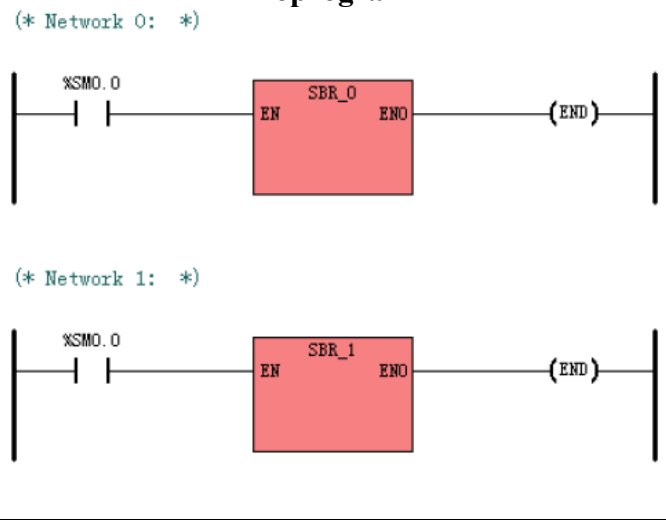
6.9.2. Visszatérés (return) utasítás

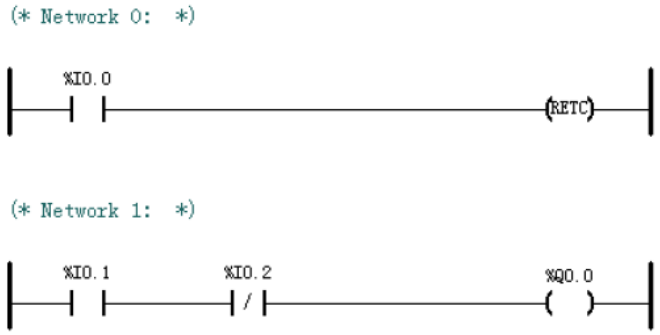
Megjegyzés: A visszatérés csak szubrutinban vagy megszakítás rutinban használható.

	Név	Használat	Csoport	
LD	RETC			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	RETCN			
IL	RETC	RETC	U	
	RETCN	RETCN		

- **LD**
  A **RETC** utasítás kilépésre használható szubrutinból vagy megszakításból, program futtatása visszatér a főprogramba, ha bal oldali feltétel igazzá válik.
  A **RETCN** utasítás kilépésre használható szubrutinból vagy megszakításból, program futtatása visszatér a főprogramba, ha bal oldali feltétel hamissá válik

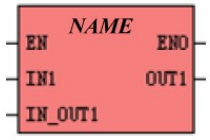
- **IL**
  A **RETC** utasítás kilépésre használható szubrutinból vagy megszakításból, program futtatása visszatér a főprogramba, ha CR=1
  A **RETCN** utasítás kilépésre használható szubrutinból vagy megszakításból, program futtatása visszatér a főprogramba, ha CR=0

<table>
<tr>
<td>LD</td>
<td>
Főprogram


SRB_0 szubrutin


</td>
<td>Mivel az SM0.0 értéke mindig 1, ezért az SRB_0 szubrutin mindig meghívásra kerül. Ha I0.0 nincs bekapcsolva, SRB_0 szubrutin Network 1 része is végrehajtásra kerül. Ha I0.0 be van kapcsolva, akkor visszatér a program futása a főprogramba.</td>
</tr>
<tr>
<td>IL</td>
<td colspan="2">
Main Program:
LD %SM0.0 (* CR létrehozása SM0.0-val *)
CAL SBR_0 (* SBR_0 szubrutin hívása *)
CAL SBR_1 (* SBR_1 szubrutin hívása*)
SBR_0:
LD %I0.0 (* CR létrehozása I0.0-val *)
RETC (* Ha CR=1, SBR_0 futtatása megszakad, és visszatér a főprogramba *)
LD %I0.1 (* Ha RETC nem kerül végrehajtásra, a következő programrész fut le*)
ANDN %I0.2
ST %Q0.0
</td>
</tr>
</table>

6.9.3. CAL (Szubrutin hívása)

	Név	Használat	Csoport	
LD	CAL			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	CAL	CAL *NÉV, paraméter 1, paraméter 2, ...*	U	

Ezt az utasítás egy NAME elnevezésű szubrutin meghívására és végrehajtására használjuk. A meghívott szubrutint előzőleg létre kell hozni.
A CAL utasítást használhatjuk paraméterekkel vagy anélkül. Ha paraméterekkel használjuk, az aktuális paraméterek adat és a változó típusainak meg kell egyezni a változó táblázatban definiált paraméterekkel.

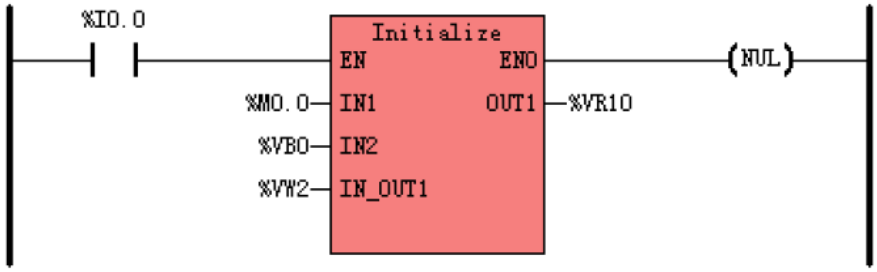
- **LD**
  A létrehozott szubrutinok megtalálhatók a bal oldalt látható kezelő elem *Instructions* fülét kiválasztva az *SBR* bejegyzés alatt.

- **IL**
  HA CR=1, a vezérlő szubrutint meghívja és végrehajtja.
  A CAL utasítás nincs hatással a CR-re, de CR értéke a szubrutin futása során megváltozhat.

LD

Főprogram

```
(* Network 0 *)
(* call the subroutine 'Initialize' *)
```



Az *Initialize* szubrutin változó táblája

Address	Symbol	Var Type	Data Type	Comment
%L0.0	IN1	VAR_INPUT	BOOL	
%LB16	IN2	VAR_INPUT	BYTE	
%LW22	IN_OUT1	VAR_IN_OUT	INT	
%LD18	OUT1	VAR_OUTPUT	REAL	

IL

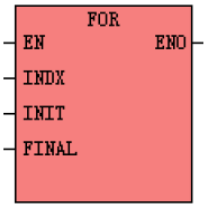
Főprogram

```
(* Network 0 *)
(*'Initialize' szubrutin hívása, paraméterekkel*)
LD %I0.0
CAL Initialize, %M0.0, %VB0, %VW2, %VR10
```

Az *Initialize* szubrutin változó táblája

	Address	Symbol	Var Type	Data Type	Comment
	%LD.0	IN1	VAR_INPUT	BOOL	
	%LB16	IN2	VAR_INPUT	BYTE	
	%LW22	IN_OUT1	VAR_IN_OUT	INT	
▶	%LD18	OUT1	VAR_OUTPUT	REAL	

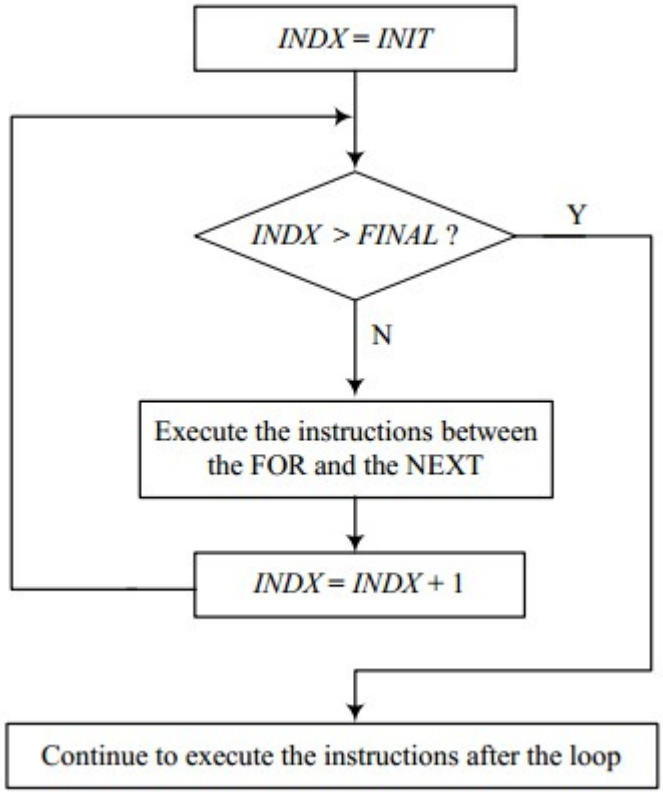
6.9.4. FOR/NEXT (ciklus szervezés)

	Név	Használat	Csoport	
LD	FOR			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
	NEXT			
IL	FOR	FOR *INDX, INIT, FINAL*	U	
	NEXT	NEXT		

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
INDX	Bemenet	INT	M, V, L, SM
INIT	Bemenet	INT	M, V, L, SM, T, C, konstans
FINAL	Kimenet	INT	M, V, L, SM, T, C, konstans

A FOR/NEXT utasításokkal ciklus szervezhető, ahol a ciklusban levő utasítások *n*-szer ismétlődnek. Meg kell adni a ciklusváltozót (INDX), kezdőértékét (INIT), és a végértékét (FINAL). A NEXT utasítás jelzi a ciklus végét, a FOR és NEXT közötti utasítások kerülnek megadott szám szorzatával történő végrehajtásra. Minden egyes FOR utasításhoz kell, hogy tartozzon NEXT utasítás, csak párban használhatók.
Ha egy FOR/NEXT ciklus tartalmazhat másik FOR/NEXT ciklust, maximálisan nyolc utasítás mélységig.

A folyamat végrehajtását a következő ábrán látjuk:



FOR/NEXT utasítás használatakor a következőkre kell ügyelni:

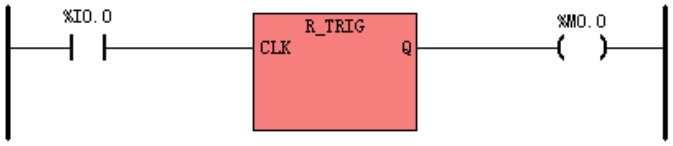
- A FOR utasítás csak a második lehet egy network-ben
- A NEXT utasítás csak külön network-ben használható
- Ha a ciklusban megváltozik a végérték változó értéke, akkor megváltozik a kilépési feltétel is.
- Ha a ciklus futási ideje hosszabb, mint a CPU watchdog-ba beállított idő, akkor a központi egység hibával újra indul.

- **LD**
  Ha EN=1, az utasítás végrehajtásra kerül.

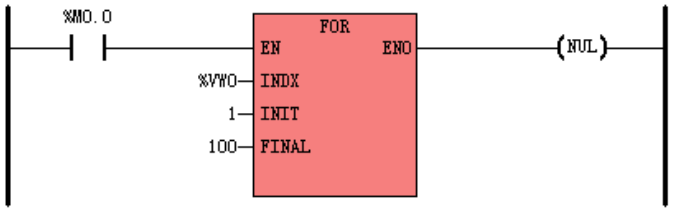
- **IL**
  Ha CR=1, az utasítás végrehajtásra kerül, és nincs hatással CR-re.

LD

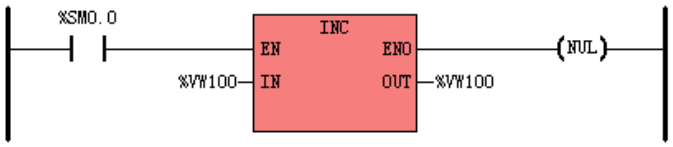
(* Network 0 *)
(* I0.0 felfutó élének hatására a ciklus 100-szor kerül végrehajtásra *)



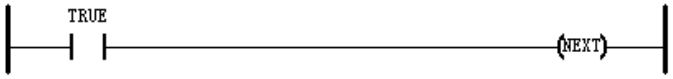
(* Network 1 *)



(* Network 2 *)



(* Network 3 *)



IL

```
(* Network 0 *)
(* I0.0 felfutó élének hatására a ciklus 100-szor kerül végrehajtásra *)
LD %I0.0
R_TRIG
ST %M0.0
(* Network 1 *)
LD %M0.0
FOR %VW0, 1, 100
(* Network 2 *)
LD %SM0.0
INC %VW100
(* Network 3 *)
LD TRUE
NEXT
```

6.9.5. END (Ciklus befejezése)

	Név	Használat	Csoport	
LD	END			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	END	END	U	

Az utasítás a főprogramban használható, az aktuális ciklus befejezésére.
Az utolsó utasítás utána a Kinco Builder program automatikusan meghívja az END utasítást.

- **LD**
  Ha a horizontális összekötő állapota 1 lesz, az utasítás végrehajtásra kerül.

- **IL**
  Ha CR=1, az utasítás végrehajtásra kerül. Nincs hatással a CR-re.

6.9.6. STOP (CPU megállítása)

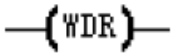
	Név	Használat	Csoport	
LD	STOP			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	STOP	STOP	U	

Ez az utasítás befejezi a program futtatását és a CPU-t RUN üzemmódból STOP üzemmódba kapcsolja.

- **LD**
  Ha a horizontális összekötő állapota 1 lesz, az utasítás végrehajtásra kerül.

- **IL**
  Ha CR=1, az utasítás végrehajtásra kerül. Nincs hatással a CR-re.

6.9.7. WDR (Watchdog reset)

	Név	Használat	Csoport	
LD	WDR			☑ CPU304 ☑ CPU304EX ☑ CPU306
IL	WDR	WDR	U	☑ CPU306EX ☑ CPU308

Ez az utasítás újraindítja a CPU watchdog időzítőjét.
A WDR utasítás használatával növelhetjük a PLC ciklus idejét, elkerülve, hogy a watchdog hibát jelezzen, tehát a program hosszabb időt kap a végrehajtásra. Az utasítás használatakor körültekintően kell eljárni, ugyanis a következő funkciók csak a ciklusidő lefutása után érhetőek el:

- CPU öndiagnosztika
- Bemenetek olvasása (mintavételezés a fizikai bemeneti csatornákról, és ezen értékek kimentése a bemeneti memória területre)
- Kommunikáció
- Kimenetek kiírása (kimeneti memórián tárolt értékek kiírása a fizikai kimeneti csatornákra)
- Időzítés 10ms és100ms időzítők futtatása

- **LD**
  Ha a horizontális összekötő állapota 1 lesz, az utasítás végrehajtásra kerül.

- **IL**
  Ha CR=1, az utasítás végrehajtásra kerül. Nincs hatással a CR-re.

6.10. Megszakítások

A megszakítások használatának célja, hogy növeljük a KINCO-K3 PLC reagálásának hatékonyságát előre definiált belső vagy külső eseményekre.
A KINCO-K3 tíz megszakítást képes kezelni, amelyek egyedi számmal vannak jelölve.
Ha engedélyezni szeretnénk egy megszakítást, az ATCH utasítást kell használnunk.
A DTCH utasítással megszüntethető a kapcsolat a megszakítási szám és a megszakítási rutin között, melyet egy megadott esemény bekövetkezésekor kívánunk futtatni.

6.10.1 Megszakítások kezelése

A megszakítás rutin egyszer fut le, ha a hozzárendelt feltételek teljesülnek. A megszakítási rutin legutolsó utasítása végrehajtását követően a program végrehajtása visszatér a főprogramba. A RETC vagy a RETCN utasításokkal is kiléphetünk a megszakítási rutinból.
A megszakítások használatának célja, hogy a KINCO-K3 PLC gyorsan reagáljon a különleges eseményekre, így javasolt a lehető legrövidebb, optimalizált megszakítási rutinok használata.

6.10.2. Megszakítás prioritási sor

A különböző események különböző prioritási szinttel rendelkeznek. A megszakítási eseményeket a vezérlő sorba állítja prioritási szint és idő szerint; azonos prioritással rendelkező megszakításokat a következő elv szerint kezeli: "Elsőnek érkezik elsőnek kerül kiszolgálásra", a magasabb prioritású csoportok előnyt élveznek. Ha csak egy megszakítás rutin érkezik akkor az azonnal végrehajtásra kerül. Ha egy megszakítás rutint futása elkezdődik, másik megszakítás nem szakíthatja meg. Ha egy megszakítás rutin közbe egy másik megszakítás vagy megszakítások érkeznek, azokat a vezérlő sorba állítja prioritás és idő szerint.

6.10.3. KINCO-K3 által támogatott megszakítás események típusai

A KINCO-K3 következő megszakítás események típusait támogatja:

- Kommunikációs port megszakítása
  Ez a típusú megszakítás a legnagyobb prioritású.
  Szabad protokoll kommunikációs módba használható.
  Az adás és vétel utasításokkal kezelhető a teljes kommunikáció folyamata.

- I/O megszakítások
  Ezeknek a megszakításnak közepes szintű prioritása van.
  Ez a megszakítás tartalmaz fel/lefutó él megszakítást, HSC (gyors számláló) megszakításokat és PTO (gyors kimenet) megszakításokat.
  A fel/lefutó él megszakításokat csak a CPU első négy DI csatornáján (%I0.0~%I0.3) használhatjuk. Ezen bemenetek állapotváltozásának figyelésére alkalmazhatjuk, ha a figyelt bemenetek a jelváltozás gyorsabb, mint a CPU ciklusideje.
  A PTO megszakítás azonnal bekövetkezik, amikor a kimenetre a megadott számú impulzus teljesül. Egy jellemző alkalmazási terület a léptető motorok vezérlése.

- Időzítő megszakítások
  Ezek a legalacsonyabb prioritású megszakítások.
  Ez a megszakítás tartalmaz időzített megszakításokat és időzítő T2 és T3 megszakításokat.
  Az időzített megszakítás periodikusan következik be (ms), és ezek periodikus feladatokat lát el.
  Az időzítő megszakítás azon nyomban bekövetkezik, ahogy a T2 vagy T3 eléri az előre beállított értékét.

6.10.4. Megszakítás eseménylista

Esemény szám	Leírás	Típus	Prioritás
32	PORT 1: XMT kész	Kommunikációs port megszakításai	legmagasabb
31	PORT 1: RCV kész		
30	PORT 0: XMT kész		
29	PORT 0: RCV kész		
28	PTO 0 kész	I/O megszakítások	
27	PTO 1 kész		
26	I0.0 felfutó él		
25	I0.0 lefutó él		
24	I0.1 felfutó él		
23	I0.1 lefutó él		
22	I0.2 felfutó él		
21	I0.2 lefutó él		
20	I0.3 felfutó él		
19	I0.3 lefutó él		
18	HSC0 CV=PV		
17	HSCO irány megváltozott		
16	HSCO belső reset		
15	HSC1 CV=PV		
14	HSC1 irány megváltozott		
13	HSC1 belső reset		
12	HSC2 CV=PV		
11	HSC2 irány megváltozott		
10	HSC2 belső reset		
9	HSC3 CV=PV		
8	HSC4 CV=PV		
7	HSC4 irány megváltozott		
6	HSC4 belső reset		
5	HSC5 CV=PV		
4	Időzített megszakítás 1, periódus SMW24-be adható meg, egység: ms 1 ~ 65535	Időzítő megszakítások	
3	Időzített megszakítás 2, periódus SMW22-be adható meg, egység: ms 1 ~ 65535		
2	Időzítő T3 ET=PT		
1	Időzítő T2 ET=PT		legalacsonyabb

6.10.5. ENI (megszakítás engedélyezés), DISI (megszakítás tiltás)

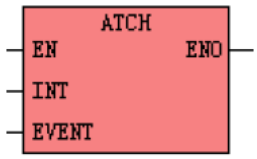
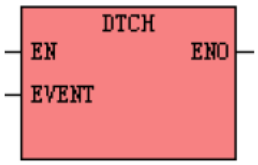
	Név	Használat	Csoport	
LD	ENI			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	DISI			
IL	ENI	ENI	U	
	DISI	DISI		

Az ENI utasítás globálisan engedélyez minden ATCH utasítással engedélyezett megszakítást.
A DISI utasítás globálisan letiltja az összes megszakítást.
Amikor a CPU-t RUN módba kapcsol, a megszakítások alapértelmezetten engedélyezve lesznek.

- **LD**
  Ha a horizontális összekötő állapota 1 lesz, az utasítás végrehajtásra kerül.

- **IL**
  Ha CR=1, az utasítás végrehajtásra kerül. Nincs hatással a CR-re.

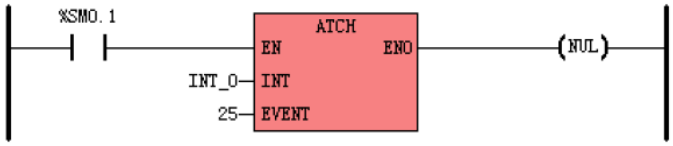
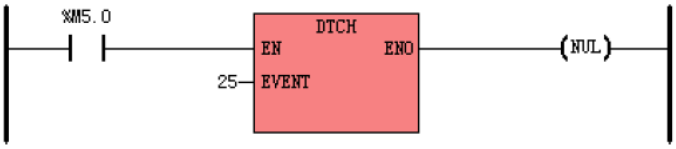
6.10.6. ATCH és DTCH utasítások (csatol / lecsatol)

	Név	**Használat**	**Csoport**	
LD	ATCH			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	DTCH			
IL	ATCH	ATCH *INT, EVENT*	U	
	DTCH	DTCH *EVENT*		

Operandus	**Bemenet/Kimenet**	**Adat típus**	**Leírás**
INT	Bemenet		Egy létező megszakítás neve
EVENT	Bemenet	INT	Konstans, megszakítás esemény száma

- **LD**
  Ha EN=1 az ATCH utasítás összekapcsol egy megszakítási eseményt egy megszakítás rutinnal, és engedélyezi a megszakítást. Összekapcsolhatunk több eseményt egy megszakítás rutinhoz, de egy eseményt csak egy megszakítás rutinhoz kapcsolhatunk.
  Ha EN=1 a DTCH utasítás szétválasztja az összekötött megszakítás eseményt és a megszakítás rutint, és a megszakítási eseményt letiltja.

- **IL**
  Ha CR=1 az ATCH utasítás összekapcsol egy megszakítás eseményt egy megszakítás rutinnal és engedélyezi a megszakítás eseményt. Ez az utasítás nincs hatással CR-re.
  Ha CR=1 a DTCH utasítás szétválasztja az összekötött megszakítási eseményt és a megszakítás rutint, és a megszakítási eseményt letiltja. Ez az utasítás nincs hatással CR-re.

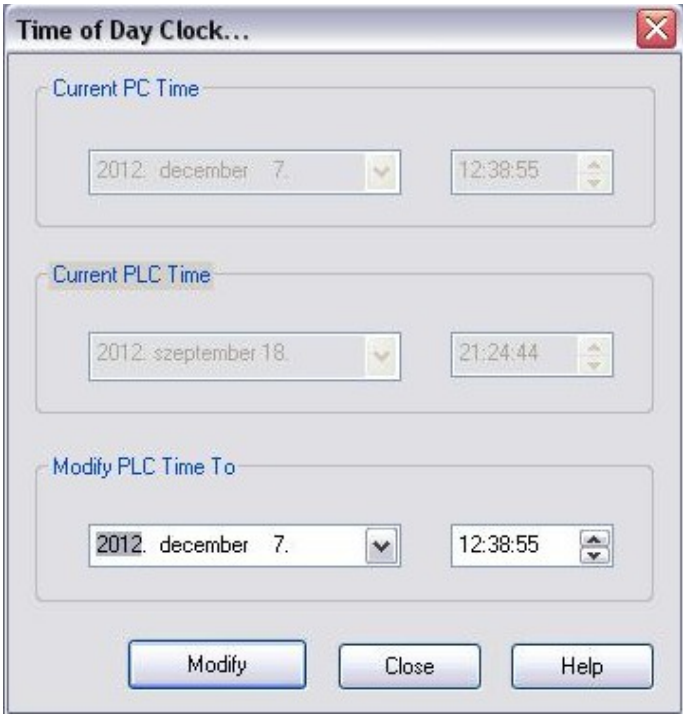
LD	(* NETWORK 0 *) (*Az első ciklusban, 25-ös megszakítást összerendeli INT_0 megszakítási rutinnal*)  (* NETWORK 1 *) (*Ha M5.0=1, 25-ös megszakítást letiltja a PLC *) 
IL	(* NETWORK 0 *) LD %SM0.1 ATCH INT_0, 25 (*Az első ciklusban, 25-ös megszakítást összerendeli INT_0 megszakítási rutinnal*) LD %M5.0 (* CR létrehozása M5.0 -val *) DTCH 25 (*Ha CR=1, 25-ös megszakítást letiltja *)

6.11. RTC - Valós idejű óra

A valós idejű órát (RTC) a CPU tartalmazza, mely rendelkezik naptár funkcióval is. A valós idejű óra/naptár BCD formátumot használ, automatikusan beállítja a szökőévet. A valós idejű óra tartalmát egy szuper kapacitás védi, mely normál hőmérsékleten csak 72 órán át képes az adatok védelmére, amennyiben a PLC nincs tápfeszültség alatt.

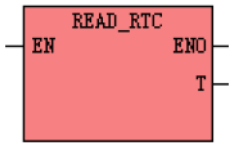
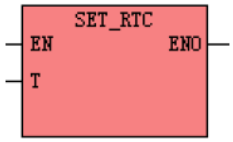
6.11.1. RTC beállítása

A pontos időt és dátumot javasolt beállítani használat előtt.
Használjuk a [PLC] → [Time of Day Clock...] menüt, a "Time of Day Clock..." párbeszédablak megnyitásához, és az RTC online beállításához, mint ahogy a következő ábrán látjuk:



- Current PC Time: A számítógép aktuális idejét mutatja
- Current PLC Time: A PLC aktuális idejét mutatja
- Modify PLC Time To: Itt adható meg a PLC kívánt ideje.
- Modify: Kattintsunk erre a gombra és a CPU modulba kerül a beírt idő és dátum, innentől a PLC a megadott dátumot, és időt használja.

6.11.2. READ_RTC and SET_RTC

	Név	Használat	Csoport	
LD	READ_RTC			☐ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	SET_RTC			
IL	READ_RTC	READ_RTC *T*	U	
	SET_RTC	SET_RTC *T*		

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
T	Bemenet (SET_RTC)	BYTE	V
	Kimenet READ_RTC		

A READ_RTC utasítással kiolvashatjuk az aktuális dátumot és időt a PLC-ből és egy 8 byte-os T kezdőcímű idő pufferbe írja.
A SET_RTC utasítással beírhatjuk a dátumot és időt egy 8 byte-os, T kezdőcímű idő pufferbe.

A tárolási dátum/idő formátum a következő táblázatban látható:
Megj.: Minden érték BCD kódolású

V bájt	Jelentés	Megjegyzés
T	Hét	Tartomány: 1~7, jelentése: 1=Hétfő, 7=Vasárnap
T+1	Másodperc	Tartomány: 0~59
T+2	Perc	Tartomány:0~59
T+3	Óra	Tartomány:0~23
T+4	Nap	Tartomány:1~31
T+5	Hónap	Tartomány:1~12
T+6	Év	Tartomány:0~99
T+7	Század	Értéke 20, nem változik

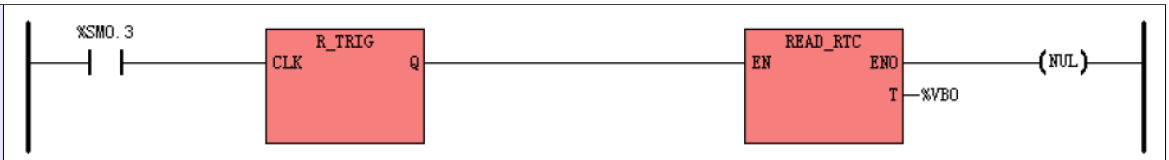
Megjegyzés:

A központi egység nem ellenőrzi a megadott dátum és idő helyességét (pl. február 30), így a megadásakor körültekintően kell eljárni.

- **LD**
  Ha a horizontális összekötő állapota 1 lesz, az utasítás végrehajtásra kerül.

- **IL**
  Ha CR=1, az utasítás végrehajtásra kerül. Nincs hatással a CR-re.

LD	(* NETWORK 0 *) (*Valós idejű óra olvasása másodpercenként*)



(* NETWORK 1 *)

(*Q0.0 kimenet bekapcsol 9:00 – 18:00 között*)

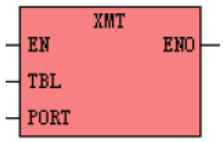
IL

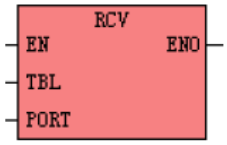
```
(* NETWORK 0 *)
(*Valós idejű óra olvasása másodpercenként*)
LD %SM0.3
R_TRIG
READ_RTC %VB0
(* NETWORK 1 *)
(*Q0.0 kimenet bekapcsol 9:00 – 18:00 között*)
LD %SM0.0
GE %VB3, B#16#9
LT %VB3, B#16#18
ST %Q0.0
```

6.12. Kommunikációs utasítások

Ezek az utasítások szabad-protokoll kommunikációra használhatók. A szabad-protokoll kommunikációs mód teljes hozzáférést biztosít a CPU kommunikációs portjaihoz. Ismert kommunikációjú eszközök illesztésére használható, az ASCII és bináris protokollok megvalósítására is használható. A központi egység, típustól függően, egy vagy két soros porttal rendelkezik, melyek alapértelmezetten Modbus RTU Slave-ként vannak paraméterezve.
A kommunikációs paraméterek megadhatók, úgy mint átviteli sebesség, paritás, adatbitek és stopbitek száma, a PLC Hardware ablakában.

6.12.1. XMT és RCV

	Név	Használat	Csoport	
LD	XMT			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308

	RCV			
IL	XMT	XMT *TBL, PORT*	U	
	RCV	RCV *TBL, PORT*		

Operandus	**Bemenet/Kimenet**	**Adat típus**	**Memória terület**
TBL	Bemenet	BYTE	I, Q, M, V, L, SM
PORT	Bemenet	INT	Konstans (0 vagy 1)

Az **XMT** utasítás a pufferben található adatokat kiküldi egy megadott soros porton szabad protokoll kommunikációs módban. Az adatpuffer a TBL címen kezdődik, az első byte megadja a küldendő byte-ok számát, majd ezt követik az adatok. Ha SM87.1 = 1, amikor a központi egység elküldte az utolsó karaktert a pufferből, akkor létrejön XMT-complete megszakítási esemény. (megszakítási esemény szám 30 PORT 0 esetén és 32 PORT 1 esetében) Ha a küldendő byte-ok száma 0, akkor az XMT utasítás nem kerül végrehajtásra, és a megszakítási esemény sem jön létre.

Az **RCV** utasítás egy megadott soros portról fogad adatokat szabad protokoll kommunikációs módban, a fogadott adatok egy pufferbe kerülnek. Az adatpuffer a TBL címtől kezdődik, az első byte a fogadott byte-ok számát tartalmazza, majd az adatok következnek. Az RCV utasítás kezdő és végfeltételeit meg kell határozni. Ha SM87.1=1 amikor a központi egység befejezi az adatok fogadását (akár hibásan is), akkor létrejön az RCV-complete megszakítási esemény. (megszakítási esemény szám 29 PORT 0 esetén és 31 PORT 1 esetében)

LD nyelvnél az EN bemenettel engedélyezhető XMT és RCV utasítások végrehajtása.
IL nyelvnél az CR-rel engedélyezhető XMT és RCV utasítások végrehajtása. Az eredmény nincs hatással a CR-re.

Státusz és vezérlő regiszterek szabad kommunikációs módban

Az XMT és RCV utasítások írhatnak vagy olvashatnak bizonyos regisztereket az SM memória területen.

1. **SMB86 --- Fogadás státusz regiszter**

Bit (olvasható)		**Státusz**	**Leírás**
PORT 0	**PORT 1**		
SM86.0		1	Paritás hibát észlelt, de a fogadás nem szakad meg
SM86.1		1	Fogadás befejeződött, mert a fogadás elérte a maximum karaktert. (SMB94)
SM86.2		1	Fogadás befejeződött, időtúllépés miatt (SMW92)
SM86.3		1	Fogadás befejeződött, mert a rendszer túllépte az időkorlátot.
SM86.4		-	fenntartva
SM86.5		1	Fogadás befejeződött, mert felhasználó által definiált End karaktert fogadott. (SMB89)
SM86.6		1	Fogadás befejeződött paraméter hiba vagy hiányzó Start vagy End feltétel miatt.
SM86.7		1	Fogadás befejeződött mert a felhasználó megszakította a parancsot (SM87.7)

2. **SMB87 --- Fogadás vezérlő regiszter**

Bit		Státusz	Leírás
PORT 0	**PORT 1**		
SM87.0		-	Fenntartva
SM87.1		0	Nem engedélyezett a XMT és RCV megszakítások
		1	Engedélyezi a XMT és RCV megszakításait
SM87.2		0	SMW92-t figyelmen kívül hagyja.
		1	Időtúllépés engedélyezése, ha SMW92 regiszterben megadott idő letelik fogadás közben
SM87.3		-	Fenntartva
SM87.4		0	SMW90-t figyelmen kívül hagyja.
		1	Átvált a valós fogadásra ha az SMW90-ben lévő időintervallumot túllépi.
SM87.5		0	SMB89-t figyelmen kívül hagyja.
		1	SMB89-ben megadott END karakter engedélyezése
SM87.6		0	SMB88-at figyelmen kívül hagyja.
		1	SMB88-ben megadott START karakter engedélyezése
SM87.7		0	RCV funkció tiltása, legerősebb prioritású
		1	RCV funkció engedélyezve

3. **Egyéb vezérlő regiszterek**

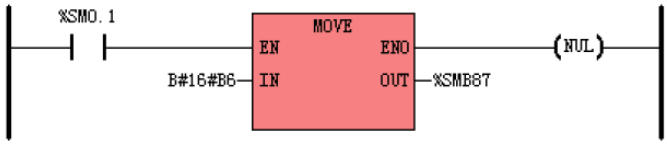
PORT 0	PORT 1	Leírás
SMB88		Felhasználó által definiált START karakter tárolása. Az RCV utasítás végrehajtását követően tényleges adat fogadás csak akkor történik meg, ha az adatok START karakterrel kezdődnek. A fogadott adatok között az első adatként a START karakter is eltárolásra kerül. SM87.6-ot 1-be kell állítani, hogy SMB88 elérhető legyen.
SMB89		Felhasználó által definiált END karakter tárolása. Amennyiben a központi egység END karaktert fogad, az adatok fogadását megszakítja. SM87.5-öt 1-be kell állítani, hogy SMB89 elérhető legyen.
SMW90		A felhasználó által definiált fogadási készenléti időt tárolja. (1~60,000ms). Az RCV utasítás végrehajtását követően, a készenléti idő letelte után a központi egység adat fogadási módba kapcsol, függetlenül, hogy fogadott-e START karaktert, vagy sem. SM87.4-et 1-be kell állítani, hogy SMW90 elérhető legyen.
SMW92		A felhasználó által definiált időtúllépési időt tárolja (1~60,000ms). Az RCV utasítás futása közben, ha a megadott ideig nem érkezik adat, a központi egység az adatok fogadását megszakítja. SM87.2-et 1-be kell állítani, hogy SMW92 elérhető legyen.
SMW94		A maximum fogadható karakterek számát tárolja (1~255). A központi egység az adatok fogadását megszakítja, ha a megfelelő számú adat beérkezett, függetlenül az egyéb feltételektől.

Szabad protokoll kommunikációs módban, az alapértelmezett időtúllépés értéke 90 másodperc. Az időtúllépés a következőképpen működik: az RCV utasítás végrehajtását követően, ha a megadott ideig nem érkezik adat, a CPU az adatok fogadását megszakítja.

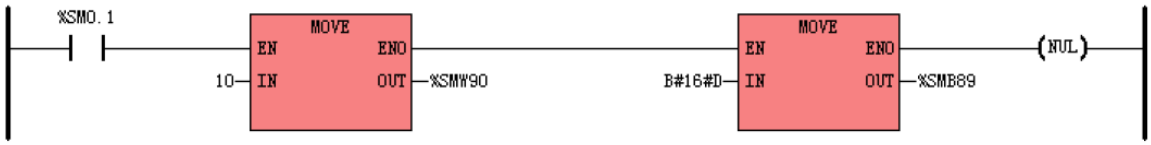
A következő példa a szabad protokoll kommunikációt mutatja be. A példában a CPU karaktereket fogad, és a RETURN karakter az END karakter. Ha az adatok fogadása hiba nélkül befejeződik, , a fogadott adatokat a központi egység visszaküldi, majd vár az új adatokra. Ha az adatok fogadása közben valamilyen hiba lép fel, (kommunikációs hiba, időtúllépés, stb.) akkor a fogadott adatokat a központi egység eldobja, majd vár az új adatokra.

LD

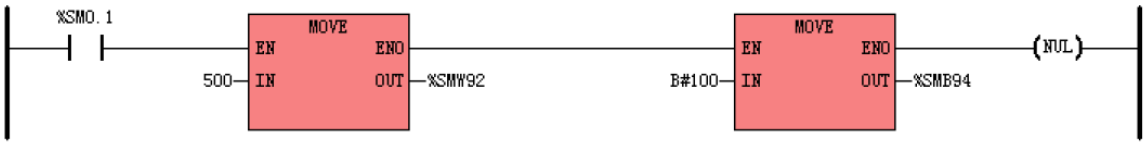
(* NETWORK 0 *)
(*Szabad protokoll kommunikáció inicializálása, START és END feltételek engedélyezése*)



(* NETWORK 1 *)
(*Készenléti idő beállítása 10ms-ra, az END feltétel pedig RETURN vagyis ASCII 13-as karakter*)

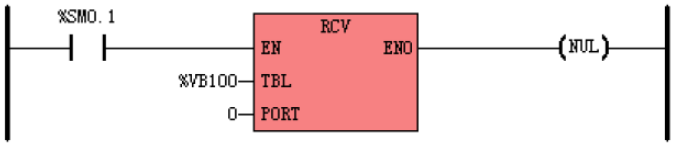


(* NETWORK 2 *)
(*Időtúllépés beállítása 500ms-ra, maximálisan fogadható karakterek száma pedig 100*)



(* NETWORK 3 *)
(*RCV-complete megszakítás hozzárendelése EndReceive szubrutinhoz, az XMT-complete megszakítás hozzárendelése EndSend szubrutinhoz*)

(* NETWORK 4 *)
(*Adatfogadás indítása egyszer, a PLC indulásakor*)

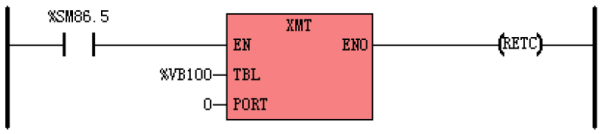


LD

EndReceive (INT00): Az RCV-complete megszakítás rutin

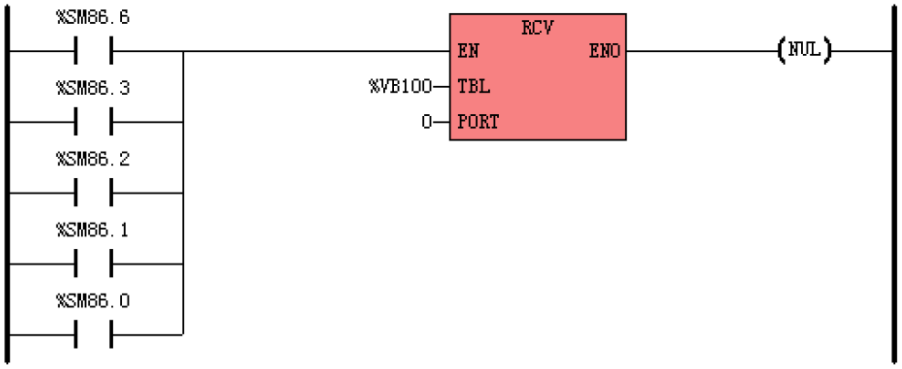
(* NETWORK 0 *)

(*Ha az END karakter beérkezett, a fogadott adatokat visszaküldi a PLC*)



(* NETWORK 1 *)

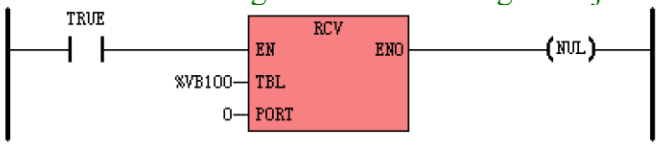
(*Ha az adatok fogadása sikertelen volt, fogadás újraindítása*)



EndSend (INT01): Az XMT-complete megszakítás rutin

(* NETWORK 0 *)

(*Adatok hibátlan fogadása után adat fogadás újraindítása*)



IL

```
(* NETWORK 0 *)
(*Szabad protokoll kommunikáció inicializálása, START és END feltételek
engedélyezése*)

LD %SM0.1
MOVE B#16#B6, %SMB87

(* NETWORK 1 *)
(*Készenléti idő beállítása 10ms-ra, az END feltétel pedig RETURN vagyis ASCII 13-as
karakter*)

LD %SM0.1
MOVE 10, %SMW90
MOVE B#16#D, %SMB89

(* NETWORK 2 *)
(*Időtúllépés beállítása 500ms-ra, maximálisan fogadható karakterek száma pedig 100*)

LD %SM0.1
MOVE 500, %SMW92
MOVE B#100, %SMB94

(* NETWORK 3 *)
(*RCV-complete megszakítás hozzárendelése EndReceive szubrutinhoz, az XMT-
complete megszakítás hozzárendelése EndSend szubrutinhoz*)

LD %SM0.1
ATCH EndReceive, 29
ATCH EndSend, 30

(* NETWORK 4 *)
(*Adatfogadás indítása egyszer, a PLC indulásakor*)

LD %SM0.1
RCV %VB100, 0
```

IL	**EndReceive (INT00): Az RCV-complete megszakítás rutin** (* NETWORK 0 *) (*Ha az END karakter beérkezett, a fogadott adatokat visszaküldi a PLC*) LD %SM86.5 XMT %VB100, 0 RETC (* NETWORK 1 *) (*Ha az adatok fogadása sikertelen volt, fogadás újraindítása*) LD %SM86.6 OR %SM86.3 OR %SM86.2 OR %SM86.1 OR %SM86.0 RCV %VB100, 0 **EndSend (INT01): Az XMT-complete megszakítás rutin** (* NETWORK 0 *) (*Adatok hibátlan fogadása után adat fogadás újraindítása*) LD TRUE RCV %VB100, 0

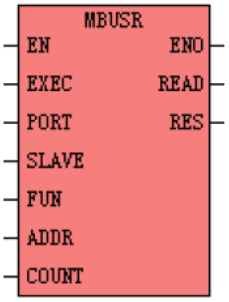
6.12.2. Modbus RTU MASTER utasítások

A Modbus RTU protokollt széleskörűen alkalmazzák az iparban. A KINCO K3 sorozat bizonyos központi egységei alkalmasak, hogy Modbus RTU MASTER-ként működjenek.
Megjegyzés: Modbus RTU MASTER mód csak a PORT1 porton lehetséges.

Modbus RTU MASTER konfigurációhoz a következő lépéseket kell elvégezni

1. Konfiguráljuk a Port 1 kommunikációs paramétereit a PLC Hardware ablakában, jelöljük ki a Modbus MASTER módot.
2. Hívjuk meg a MBUSR vagy MBUSW utasítást a programban.

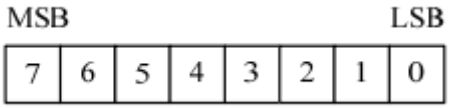
6.12.2.1. MBUSR (Modbus RTU Master Read)

	Név	Használat	Csoport	
LD	MBUSR			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	MBUSR	MBUSR *EXEC, PORT, SLAVE, FUN, ADDR,COUNT, READ, RES*	U	

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
EXEC	Bemenet	BOOL	I, Q, V, M, L, SM, RS, SR
PORT	Bemenet	INT	Konstans (1)
SLAVE	Bemenet	BYTE	I, Q, M, V, L, SM, Konstans
FUN	Bemenet	INT	Konstans (MODBUS funkció kód)
ADDR	Bemenet	INT	I, Q, M, V, L, SM, AI, AQ, Konstans
COUNT	Bemenet	INT	I, Q, M, V, L, SM, AI, AQ, Konstans
READ	Kimenet	BOOL, WORD, INT	Q, M, V, L, SM, AQ
RES	Kimenet	BYTE	Q, M, V, L, SM

Az utasítás segítségével adatokat olvashatunk MODBUS SLAVE eszközökből. A használható funkció kódok: 1 (DO állapot olvasása), 2 (DI állapot olvasása), 3 (AO olvasása) és 4 (AI olvasása). A PORT paraméterrel megadható a használni kívánt port száma, a SLAVE paraméterrel pedig a slave eszköz címe adható meg, melynek értéke 1~31 lehet. A FUN paraméterrel adható meg a küldeni kívánt funkció kód, az ADDR paraméterrel pedig a Modbus regiszter cím definiálható. A COUNT pedig az olvasandó regiszterek számát határozza meg.
Az EXEC bemenet felfutó élére indul a kommunikáció. Ha az MBUSR utasítás végrehajtódik, az EXEC bemenet felfutó élére egy kérést küld ki. A kérés elküldését követően vár a slave eszköz válaszára, azt megkapva ellenőrzi a CRC kódot, valamint azt, hogy a válasz megfelelő-e. A helyes adatok a kimeneti pufferbe kerülnek a READ címtől kezdődően, a nem megfelelő adatok pedig eldobásra kerülnek.
A READ paraméterrel megadható az adatok fogadására használható puffer kezdőcíme. A paramétere adattípusának meg kell egyeznie a funkció kód adattípusával. Ha a funkciókód 1 vagy 2, akkor READ-nek BOOL típusúnak kell lennie, amennyiben a funkciókód 3 vagy 4, akkor a típusa INT vagy WORD lehet.
A RES a kommunikáció állapotát, valamint az esetleges hibakódokat tartalmazza, értéke nem módosítható, csak olvasható.

A RES állapot kimenet felépítése a következő:

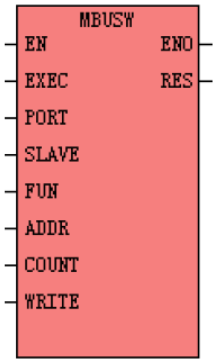


Bit7--- mutatja, hogy a kommunikáció befejezett e vagy sem: 0=nem befejezett, 1=befejezett.
Bit6--- fenntartva
Bit5--- érvénytelen SLAVE paraméter
Bit4--- érvénytelen COUNT paraméter
Bit3--- érvénytelen ADDR paraméter
Bit2--- a megadott port foglalt
Bit1--- időtúllépés
Bit0--- a fogadott üzenet hibás, CRC hiba, rendszerhiba...stb.

- **LD**
  Ha EN=1, az utasítás végrehajtódik

- **IL**
  Ha CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

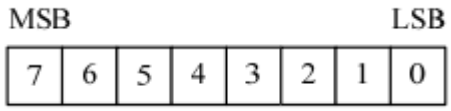
6.12.2.2. MBUSW (Modbus RTU Master Write)

	Név	Használat	Csoport	
LD	MBUSW			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	MBUSW	MBUSW *EXEC, PORT, SLAVE, FUN, ADDR, COUNT, READ, RES*	U	

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
EXEC	Bemenet	BOOL	I, Q, V, M, L, SM, RS, SR
PORT	Bemenet	INT	Konstans (1)
SLAVE	Bemenet	BYTE	I, Q, M, V, L, SM, Konstans
FUN	Bemenet	INT	Konstans (MODBUS funkció kód)
ADDR	Bemenet	INT	I, Q, M, V, L, SM, AI, AQ, Konstans
COUNT	Bemenet	INT	I, Q, M, V, L, SM, AI, AQ, Konstans
READ	Kimenet	BOOL, WORD, INT	Q, M, V, L, SM, AQ
RES	Kimenet	BYTE	Q, M, V, L, SM

Az utasítással adatokat írhatunk Modbus Slave eszközbe. Az alkalmazható funkció kódok 5 (DO írása), 6 (AO írása), 15 (több Do írása) és 16 (több Ao írása).
A PORT paraméter határozza meg a használt kommunikációs portot. A SLAVE határozza meg a slave címet, mely 1-31 között lehet. A FUN határozza meg a funkció kódot. Az ADDR határozza meg a Modbus regiszter címet. A COUNT a regiszterek számát adja meg, mely maximálisan 32 lehet. A WRITE paraméter a küldőpuffer kezdőcímét adja meg, mely a küldendő adatot vagy adatokat tartalmazza. A WRITE adattípusának meg kell egyeznie a funkció kód típusával, ha a funkció kód 5 vagy 15, akkor a WRITE típusa BOOL lehet, ha 6 vagy 16 esetén a WRITE típus INT vagy WORD lehet.
Az EXEC bemenet felfutó élére indul a kommunikáció. Ha az MBUSW utasítás végrehajtódik, az EXEC bemenet felfutó élére elküldi a kívánt adatokat. A kérés elküldését követően vár a slave eszköz válaszára, azt megkapva ellenőrzi a CRC kódot, valamint azt, hogy a válasz megfelelő-e.
A RES a kommunikáció állapotát, valamint az esetleges hibakódokat tartalmazza, értéke nem módosítható, csak olvasható.

A RES állapot kimenet felépítése a közvetkező:



Bit7--- mutatja, hogy a kommunikáció befejezett e vagy sem: 0=nem befejezett, 1=befejezett.
Bit6--- fenntartva
Bit5--- érvénytelen SLAVE paraméter
Bit4--- érvénytelen COUNT paraméter
Bit3--- érvénytelen ADDR paraméter
Bit2--- a megadott port foglalt
Bit1--- időtúllépés
Bit0--- a fogadott üzenet hibás, CRC hiba, rendszerhiba...stb.

- **LD**
  Ha EN=1, az utasítás végrehajtódik

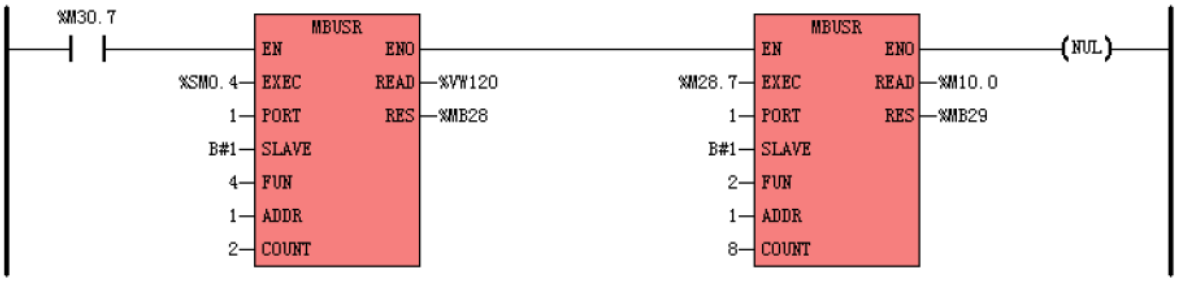
- **IL**
  Ha CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

6.12.2.3. MBUSR és MBUSW példa

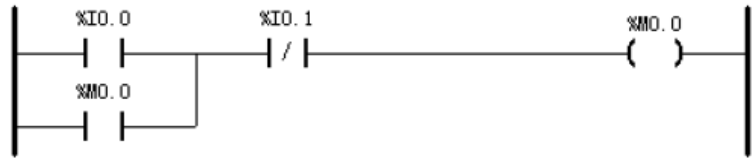
LD

(* NETWORK 0 *)
(*M30.7 jelzi, hogy MBUSW utasítás végrehajtódott-e vagy sem*)

(* NETWORK 1 *)
(*Ha port1 szabad, akkor MBUSR utasítás végrehajtódik, minden 2 másodpercben adatokat olvas 1-es slave eszköztől. Először 1., majd 2. AI regisztert olvassa, majd DI regisztereket olvassa 1-től 8-ig.*)

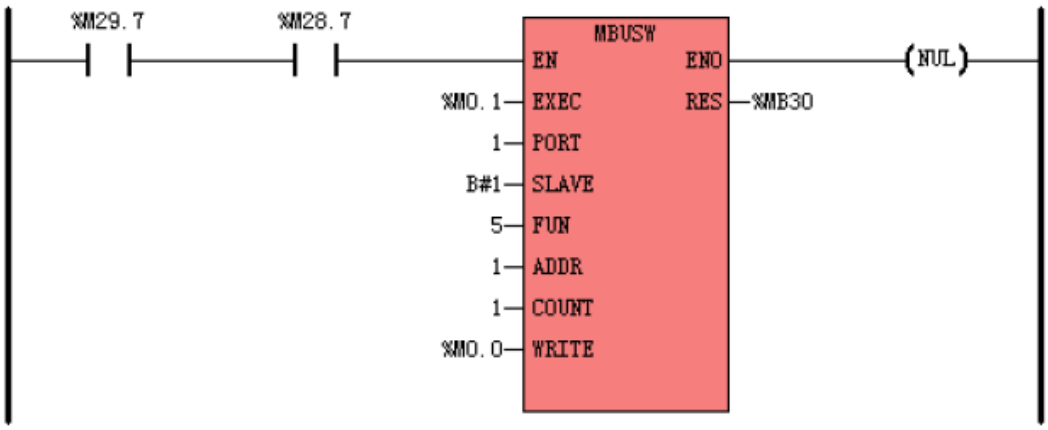


(* NETWORK 2 *)



(* NETWORK 3 *)

(* NETWORK 4 *)
(* Ha port1 szabad, akkor MBUSW utasítás végrehajtódik, ha I0.0 vagy I0.1 bekapcsolt, akkor M0.0 értéke bekerül a slave 1 első DO regiszterébe. *)



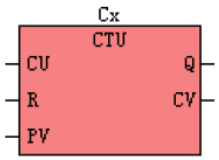
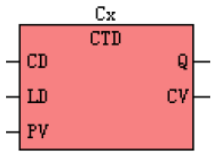
IL

```
(* NETWORK 0 *)
(*M30.7 jelzi, hogy MBUSW utasítás végrehajtódott-e vagy sem*)
LD %SM0.1
S %M30.7

(* NETWORK 1 *)
(*Ha port1 szabad, akkor MBUSR utasítás végrehajtódik, minden 2 másodpercben adatokat olvas 1-es slave eszköztől. Először 1., majd 2. AI regisztert olvassa, majd DI regisztereket olvassa 1-től 8-ig.*)
LD %M30.7
MBUSR %SM0.4, 1, B#1, 4, 1, 2, %VW120, %MB28
MBUSR %M28.7, 1, B#1, 2, 1, 8, %M10.0, %MB29

(* NETWORK 2 *)
LD %I0.0
OR %M0.0
ANDN %I0.1
ST %M0.0

(* NETWORK 3 *)
LD %I0.0
OR %I0.1
ST %M0.1

(* NETWORK 4 *)
(* Ha port1 szabad, akkor MBUSW utasítás végrehajtódik, ha I0.0 vagy I0.1 bekapcsolt, akkor M0.0 értéke bekerül a slave 1 első DO regiszterébe. *)
LD %M29.7
AND %M28.7
MBUSW %M0.1, 1, B#1, 5, 1, 1, %M0.0, %MB30
```

6.13. Számlálók

6.13.1. CTU (felfelé számláló) és CTD (lefelé számláló)

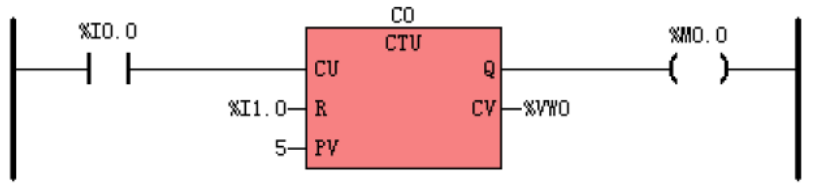
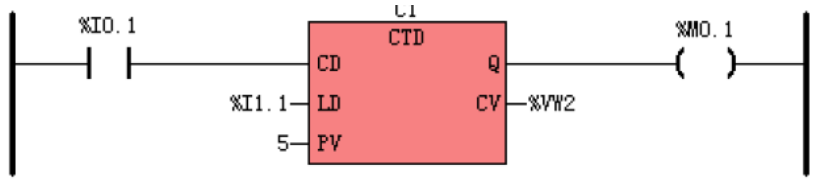
A számláló az egyik IEC61131-3 szabványban definiált funkció blokk, három típusa van: CTU, CTD és CTUD.

	Név	Használat	Csoport	
LD	CTU			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	CTD			
IL	CTU	CTU *Cx, R, PV*	P	
	CTD	CTD *Cx, LD, PV*		

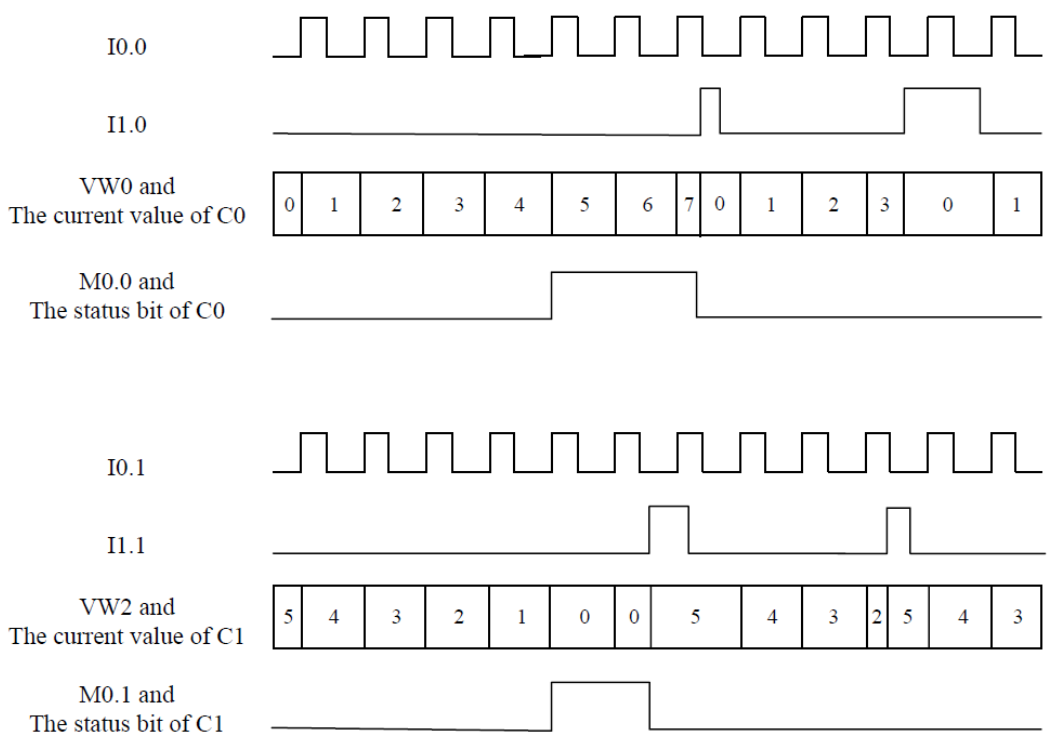
Operandus	Bemenet/Kimenet	Adat típus	Memória terület
Cx	-	számláló példányosítás	C
CU	Bemenet	BOOL	program
R	Bemenet	BOOL	I, Q, M, V, L, SM, T, C, RS, SR
CD	Bemenet	BOOL	program
LD	Bemenet	BOOL	I, Q, M, V, L, SM, T, C, RS, SR
PV	Bemenet	INT	I, Q, M, V, L, SM, AI, AQ, Konstans
Q	Kimenet	BOOL	Program
CV	Kimenet	INT	Q, M, V, L, SM, AQ

- **LD**
  A CTU számláló a CU bemenetre érkező jelek felfutó éleit számolja. Ha a CV aktuális értéke egyenlő vagy nagyobb mint a PV beállított érték, a Q kimenet és a Cx státuszbit átvált 1-re. Cx törlődik ha az R engedélyezve van. Ha a számláló túllépi PV értékét tovább számol addig amíg el nem éri az INT maximum értéket (32767).
  A CTD számláló a CD bemenetre érkező jel felfutó élének hatására lefele számol. Ha a CV aktuális értéke egyenlő vagy kisebb mint a PV beállított érték, a Q kimenet és a Cx státuszbit átvált 1-re. Cx törlődik ha az R engedélyezve van. Ha a számláló túllépi PV értékét tovább számol addig amíg el nem éri 0-t.

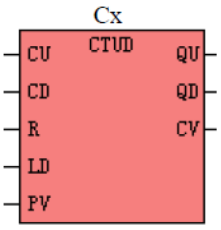
- **IL**
  A CTU számláló CR felfutó élét számolja. Ha a CV aktuális értéke egyenlő vagy nagyobb mint a PV beállított érték, a Q kimenet és a Cx státuszbit átvált 1-re. Cx törlődik ha az R engedélyezve van. Ha a számláló túllépi PV értékét tovább számol addig amíg el nem éri az INT maximum értéket (32767).
  A CTD számláló a CR felfutó élének hatására lefele számol. Ha a CV aktuális értéke egyenlő vagy kisebb mint a PV beállított érték, a Q kimenet és a Cx státuszbit átvált 1-re. Cx törlődik ha az R engedélyezve van. Ha a számláló túllépi PV értékét tovább számol addig amíg el nem éri 0-t.
  A programrész lefutása után CR felveszi Cx státuszbit értékét.

LD	IL
(* NETWORK 0 *)  (* NETWORK 1 *) 	(* NETWORK 0 *) LD %I0.0 CTU C0, %I1.0, 5 ST %M0.0 (* NETWORK 1 *) LD %I0.1 CTD C1, %I1.1, 5 ST %M0.1

Időfüggvény



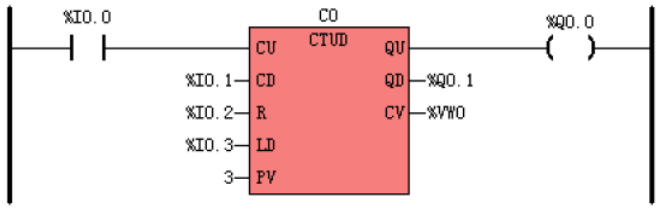
6.13.2. CTUD (fel-le számláló)

	Név	Használat	Csoport	
LD	CTUD			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	CTUD	CTUD *Cx, CD, R, LD, PV, QD*	P	

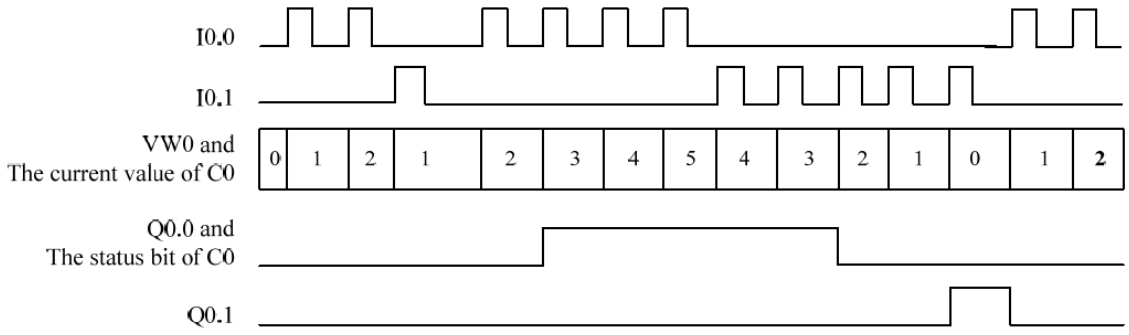
Operandus	Bemenet/Kimenet	Adat típus	Memória terület
Cx	-	számláló példányosítás	C
CU	Bemenet	BOOL	program
R	Bemenet	BOOL	I, Q, M, V, L, SM, T, C, RS, SR
CD	Bemenet	BOOL	program
LD	Bemenet	BOOL	I, Q, M, V, L, SM, T, C, RS, SR
PV	Bemenet	INT	I, Q, M, V, L, SM, AI, AQ, Konstans
Q	Kimenet	BOOL	Program
CV	Kimenet	INT	Q, M, V, L, SM, AQ

- **LD**
  A **CTUD** számláló a CU bemenet felfutó élére eggyel megnöveli az értékét, CD bemenet felfutó élének hatására pedig csökkenti azt, az aktuális érték CV-be kerül. Ha CV értéke nagyobb vagy egyenlő, mint a beállított érték (PV), QU kimenet bekapcsol, és Cx státuszbit átvált 1-re. Ha CV értéke 0, akkor QD kimenet bekapcsol, ellenkező esetben pedig kikapcsolt állapotban marad. Ha az R bemenet aktív, Cx és CV értéke törlődik. Ha az LD bemenet engedélyezett, PV értéke beíródik Cx és Cv-be. Amennyiben R és LD bemenet egyszerre aktív, R kap nagyobb prioritást.

- **IL**
  A **CTUD** számláló a CR felfutó élére eggyel megnöveli az értékét, CD bemenet felfutó élének hatására pedig csökkenti azt, az aktuális érték CV-be kerül. Ha CV értéke nagyobb vagy egyenlő, mint a beállított érték (PV), QU kimenet bekapcsol, és Cx státuszbit átvált 1-re. Ha CV értéke 0, akkor QD kimenet bekapcsol, ellenkező esetben pedig kikapcsolt állapotban marad. Ha az R bemenet aktív, Cx és CV értéke törlődik. Ha az LD bemenet engedélyezett, PV értéke beíródik Cx és Cv-be. Amennyiben R és LD bemenet egyszerre aktív, R kap nagyobb prioritást.
  A programrész lefutása után CR felveszi Cx státuszbit értékét.

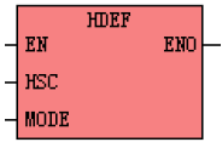
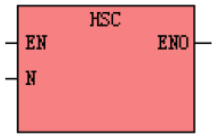
LD	IL
	LD %I0.0 CTUD C1, %I0.1, %I0.2, %I0.3, 4 ,%Q0.1 ST %Q0.0

Időfüggvény



6.13.3. Nagy sebességű számláló utasítások

A nagy sebességű számlálók nagy sebességű impulzusok számlálására használhatók, melyek frekvenciája gyorsabb, mint a központi egység végrehajtási ideje.

	Név	Használat	Csoport	
LD	HDEF			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	HSC			
IL	HDEF	HDEF *HSC, MODE*	U	
	HSC	HSC *N*		

Operandus	Bemenet/Kimenet	Adat típus	Leírás
HSC	Bemenet	INT konstans (0~5)	HSC szám
MODE	Bemenet	INT konstans (0~11)	Műveleti mód
N	Bemenet	INT konstans (0~5)	HSC szám

A **HDEF** utasítás definiálja a nagy sebességű számláló műveleti módját (MODE). Ez az utasítás minden nagy sebességű számlálóhoz alkalmas, 11 különböző műveleti mód adható meg. A mód határozza meg a nagy sebességű számlálóhoz tartozó bementi órajelet, számlálás irányát valamint a start és reset beállításokat.
A **HSC** (nagy sebességű számláló) utasítás konfigurálja és működteti a nagy sebességű számlálót, aminek a számát az N határozza meg.
Az IL nyelvben CR dönti el, hogy az utasítások végrehajtódjanak-e vagy sem. Az utasítás nincs hatással a CR-re.

6.13.3.1. Nagy sebességű számlálók a KINCO-K3 PLC esetén

Tulajdonság	CPU304	CPU306
Nagy sebességű számláló Egyfázisú Kétfázisú	2 számláló (HSC0 és HSC1) 2, 20KHz 2, 10KHz	6 számláló (HSC0 és HSC5) 6, 30KHz 4, 20KHz

A HSC3-nak és HSC5-nek egy működési módja van; HSC0-nak és HSC4-nek 1, valamint 11 működési módja van HSC1-nek és HSC2-nek.
Minden nagy sebességű számlálónak ugyanazok a funkciói vannak ugyanabban a műveleti módban.

A gyors számlálók bemenetei a következőképpen használhatók:

- Ha a reset bemenet igaz, törli az aktuális értéket, mindaddig, amíg az érték igaz.
- Ha a start bemenet igaz, a számláló számol. Amikor hamis, a számláló értéke változatlan marad, és a bejövő impulzusokat figyelmen kívül hagyja.
- Ha a reset bemenet igaz és a start bemenet hamis, akkor a reset bemenet nem kerül végrehajtásra, a számláló megőrzi az értékét. Ha mindkét bemenet igaz, akkor a számláló értéke törlődik.
- Egyfázisú számlálónál, külső számlálási irány vezérlés esetén, ha az irány bemenet aktív, akkor a számláló felfelé számol, ellenkező esetben pedig lefelé.

HSC 0				
Mód	**Leírás**	**I0.1**	**I0.0**	**I0.5**
0	Egyfázisú fel/le számláló belső irányvezérlővel	Órajel		
1			Reset	
2			Reset	Start
3	Egyfázisú fel/le számláló külső irányvezérlővel	Órajel		Irány bemenet
4			Reset	Irány bemenet
6	Kétfázisú számláló fel/le órajel bemenettel	Fel órajel	Le órajel	
9	A/B kétfázisú négyszögjel számláló	„B” órajel	„A” órajel	

HSC 1					
Mód	**Leírás**	**I0.3**	**I0.7**	**I1.2**	**I1.3**
0	Egyfázisú fel/le számláló belső irányvezérlővel			Órajel	
1		Reset			
2		Reset	Start		
3	Egyfázisú fel/le számláló külső irányvezérlővel			Órajel	Irány bemenet
4		Reset			Irány bemenet
5		Reset	Start		Irány bemenet
6	Kétfázisú számláló fel/le órajel bemenettel			Le órajel	Fel órajel
7		Reset			
8		Reset	Start		
9	A/B kétfázisú négyszögjel számláló			„B” órajel	„A” órajel
10		Reset			
11		Reset	Start		

HSC 2					
Mód	**Leírás**	**I0.6**	**I1.1**	**I1.4**	**I1.5**
0	Egyfázisú fel/le számláló belső irányvezérlővel			Órajel	
1		Reset			
2		Reset	Start		
3	Egyfázisú fel/le számláló külső irányvezérlővel			Órajel	Irány bemenet
4		Reset			Irány bemenet
5		Reset	Start		Irány bemenet
6	Kétfázisú számláló fel/le órajel bemenettel			Le órajel	Fel órajel
7		Reset			
8		Reset	Start		
9	A/B kétfázisú négyszögjel számláló			„B” órajel	„A” órajel

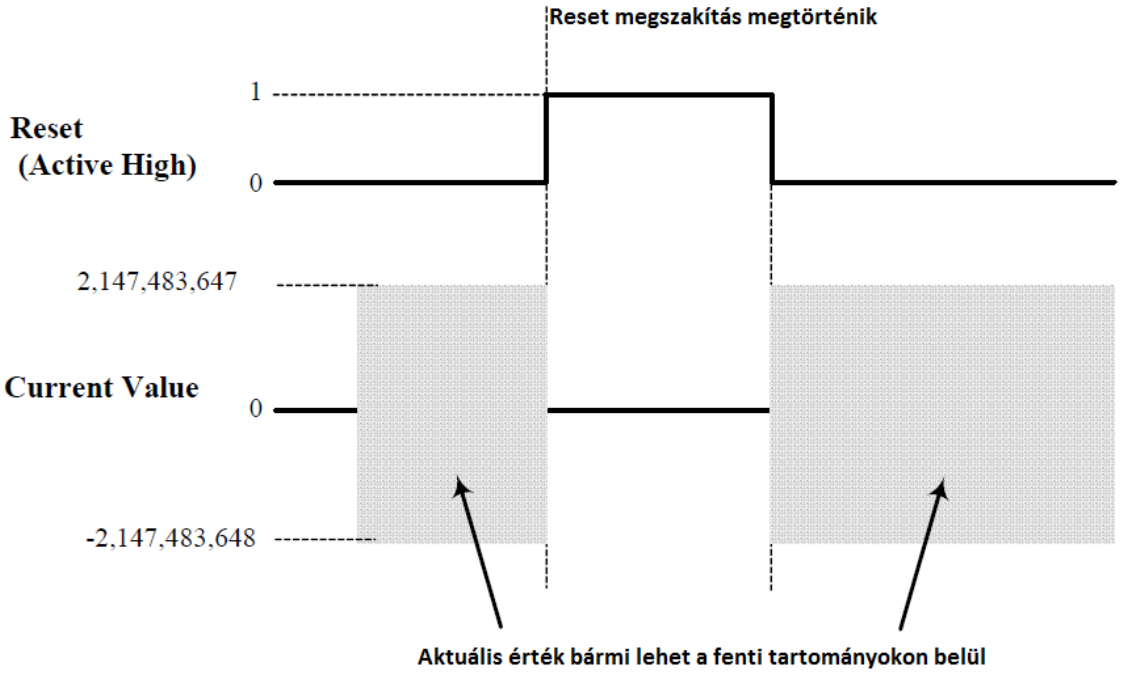
HSC 3		
Mód	**Leírás**	**I0.0**
0	Egyfázisú fel/le számláló belső irányvezérlővel	Órajel

HSC 4				
Mód	**Leírás**	**I0.2**	**I1.0**	**I1.1**
0	Egyfázisú fel/le számláló belső irányvezérlővel	Órajel		
1			Reset	
2			Reset	Start
3	Egyfázisú fel/le számláló külső irányvezérlővel	Órajel		Irány bemenet
4			Reset	Irány bemenet
6	Kétfázisú számláló fel/le órajel bemenettel	Fel órajel	Le órajel	
9	A/B kétfázisú négyszögjel számláló	„B” órajel	„A” órajel	

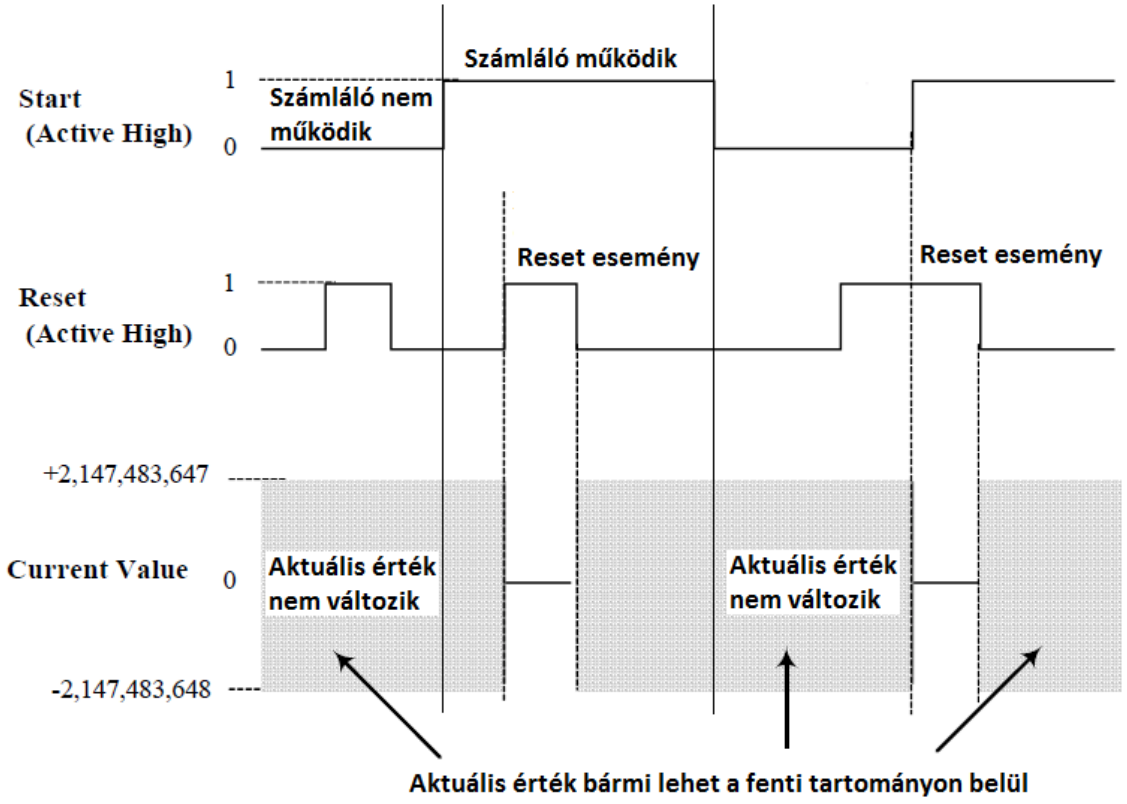
HSC 5		
Mód	**Leírás**	**I0.3**
0	Egyfázisú fel/le számláló belső irányvezérlővel	Óra

6.13.3.3. Gyors számlálók működése

- Reset és Start bemenetek működése



A fenti ábra szemlélteti, hogyan működik a RESET bemenet, amennyiben a START bemenet inaktív.



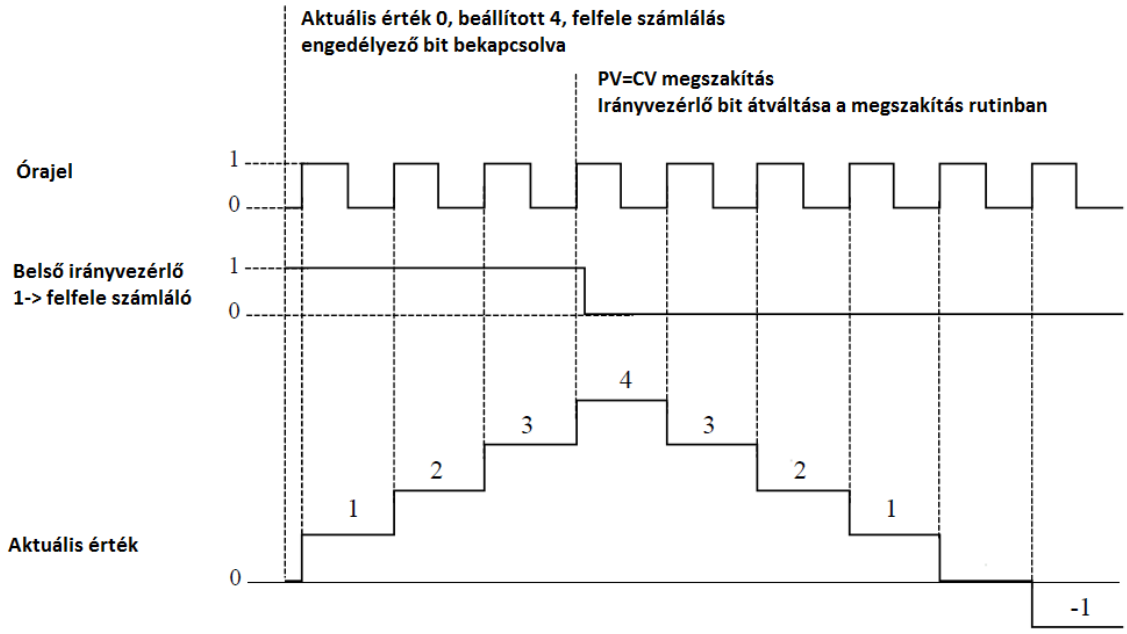
A fenti ábra szemlélteti, hogyan működik a RESET és a START bemenet gyors számláló esetén

HSC0-nak, HSC1-nek, HSC2-nek és HSC4-nek van 3 vezérlő bitje, melyeket a RESET és START aktív szintjeinek kiválasztására használhatunk és, kiválaszthatjuk, hogy 1x vagy 4x számlálási sebességet szeretnénk. A bitek a számláló vezérlő byte-jaiban találhatók, és a HSC utasítás meghívása után érvényesek.

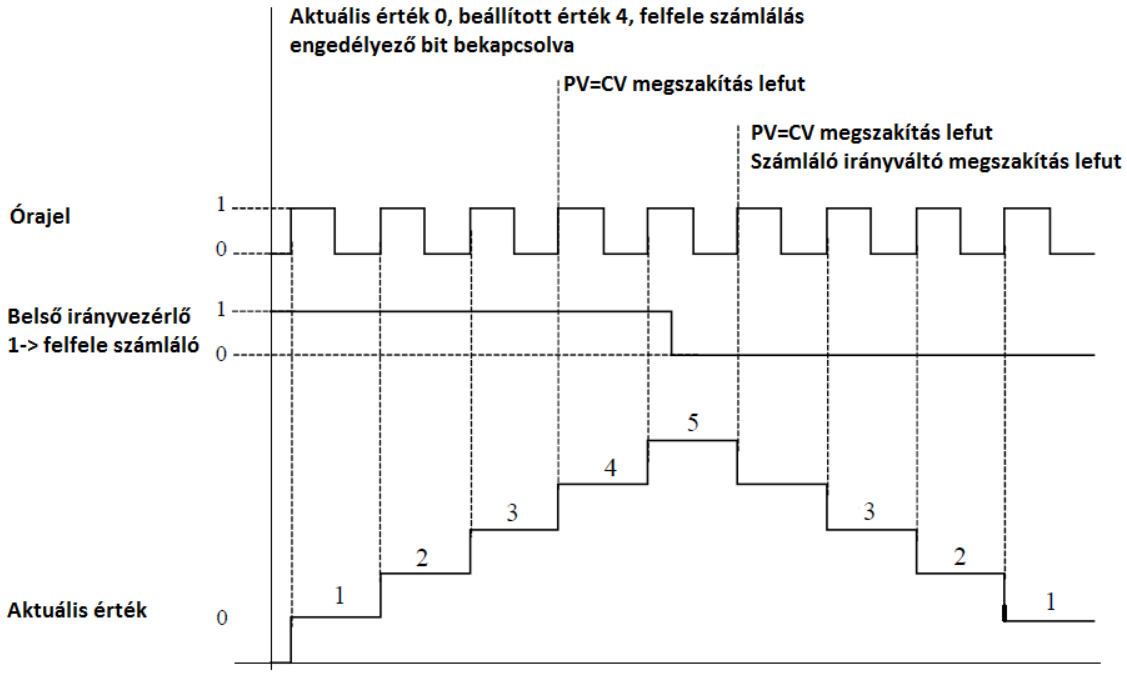
HSC0	HSC1	HSC2	HSC4	Leírás
SM37.0	SM47.0	SM57.0	SM147.0	Vezérlő bit a Reset aktív szintjéhez: 0=Aktív magas, 1=Aktív alacsony
SM37.1	SM47.1	SM57.1	SM147.1	Vezérlő bit a Start aktív szintjéhez: 0=Aktív magas, 1=Aktív alacsony
SM37.2	SM47.2	SM57.2	SM147.2	Vezérlő bit a számlálási sebességnek megadásához (kétfázisú négyszögjel esetén) 0= 4x számlálási seb., 1=1x számlálási seb.

Mielőtt végrehajtaná a HSC utasítást, ezeket e vezérlő biteket a kívánt állapotba kell állítani. Máskülönben a számláló az alapértelmezett módot fogja választani, és az alapértelmezett beállítások: reset és start bemenetek aktív magas szintűek, a számlálási sebesség egyszeres.
A HSC utasítás meghívása után a számláló beállítása már nem változtatható.

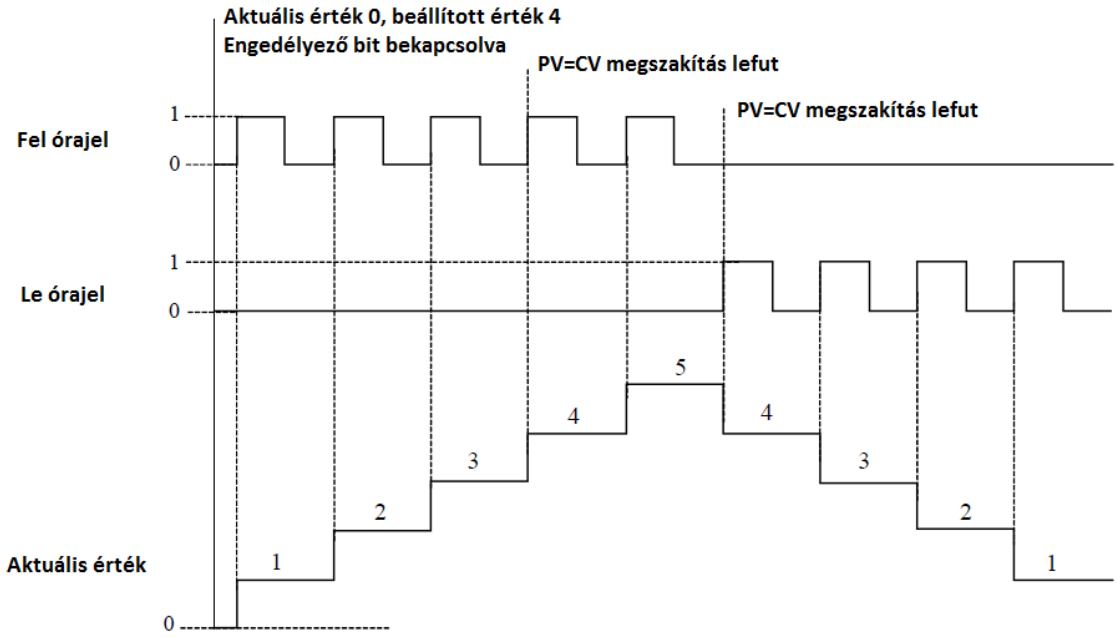
Az alábbi ábrák bemutatják a gyors számlálók különböző működési módjait.



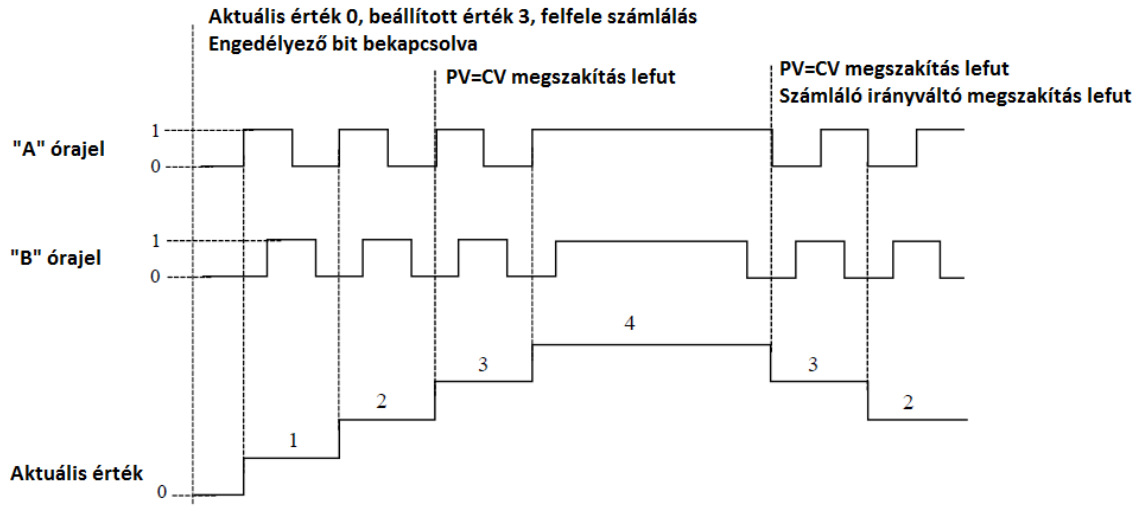
Mode 0, Mode 1 és Mode 2 mód működése



Mode 3, Mode 4 és Mode 5 működési mód



Mode 6, Mode 7 és Mode 8 mód működése



Mode 9, Mode 10 és Mode 11 mód működése
kétfázisú négyszögjel, egyszeres számlálással

6.13.3.4. Vezérlő bájt konfigurálása

A gyors számláló és a működési módjának beállítása után végezhető el a dinamikus paraméterek beállítása, melyek funkciója az alábbi táblázatban látható:

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Leírás
SM37.3	SM47.3	SM57.3	SM137.3	SM147.3	SM157.3	Számlálási irány: 0=fel, 1=le
SM37.4	SM47.4	SM57.4	SM137.4	SM147.4	SM157.4	Számlálási irány írása a HSC-be: 0=nem, 1=igen
SM37.5	SM47.5	SM57.5	SM137.5	SM147.5	SM157.5	Új beállított érték beírása a HSC-be: 0=nem, 1=igen
SM37.6	SM47.6	SM57.6	SM137.6	SM147.6	SM157.6	Új aktuális érték beírása a HSC-be: 0=nem, 1=igen
SM37.7	SM47.7	SM57.7	SM137.7	SM147.7	SM157.7	HSC engedélyezés: 0=elérhető, 1=nem elérhető

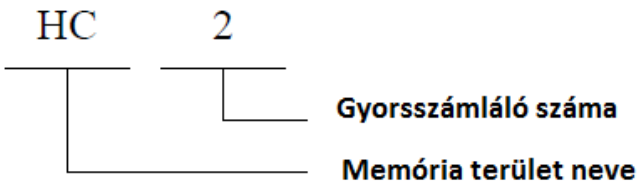
- **Aktuális érték és beállított érték konfigurálása**

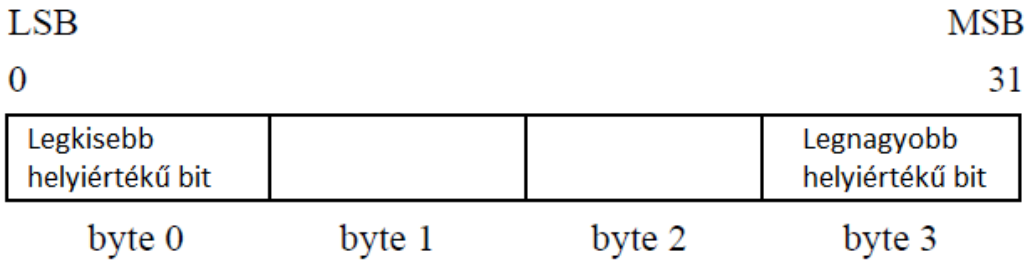
Minden nagy sebességű számláló rendelkezik 32 bites aktuális értékkel (használható mint kezdőérték) és 32 bites beállított értékkel. Az aktuális érték és a beállított érték is előjeles, dupla integer típusú. Abban az esetben ha új aktuális értéket vagy új beállított értéket akarunk a nagy sebességű számlálóba írni, a vezérlő byte-ot és az SM byte-okat (amik az aktuális értéket és/vagy a beállított értéket tárolják) kell először konfigurálni, és ezután kell a HSC utasítást végrehajtani. A következő táblázat mutatja, hogyan tárolják az SM byte-ok az aktuális és a beállított értékeket.

	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5
Új aktuális érték	SMD38	SMD48	SMD58	SMD138	SMD148	SMD158
Új beállított érték	SMD42	SMD52	SMD62	SMD142	SMD152	SMD162

Hozzáférés a nagy sebességű számláló aktuális értékéhez

A nagy sebességű számláló aktuális számlálási értéke csak olvasható és dupla integer típusban jeleníthető meg (32 bites). A nagy sebességű számláló aktuális számlálási értéke a hozzáférhető (HC) memória területet és a számláló számot használja; például, HC0 memória cím a HSC0 aktuális értékét mutatja, mint ahogy ez a következő ábrán is látható.





- Megszakítás engedélyezés

Mindegyik számláló mód támogatja a PV=CV megszakítást. Melyben megvalósítható a belső reset bemenet vagy a belső reset megszakítás, valamint a belső számlálási irányváltó bemenet vagy a belső irányváltó megszakítás. Minden fenti funkció külön-külön engedélyezhető vagy tiltható.

6.13.3.5. A státusz bájt

Az SM területen, minden nagy sebességű számlálónak külön státusz byte-ja érhető el. A státusz byte-ok különböző bitjeinek a jelentése a következő:

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Leírás
SM36.0	SM46.0	SM56.0	SM136.0	SM146.0	SM156.0	Fenntartva
SM36.1	SM46.1	SM56.1	SM136.1	SM146.1	SM156.1	Fenntartva
SM36.2	SM46.2	SM56.2	SM136.2	SM146.2	SM156.2	Fenntartva
SM36.3	SM46.3	SM56.3	SM136.3	SM146.3	SM156.3	Fenntartva
SM36.4	SM46.4	SM56.4	SM136.4	SM146.4	SM156.4	Fenntartva
SM36.5	SM46.5	SM56.5	SM136.5	SM146.5	SM156.5	Aktuális számlálási irány: 0=le, 1=fel
SM36.6	SM46.6	SM56.6	SM136.6	SM146.6	SM156.6	Aktuális érték egyenlő a beállított értékkel: 0=nem egyenlő, 1=egyenlő
SM36.7	SM46.7	SM56.7	SM136.7	SM146.7	SM156.7	Aktuális érték nagyobb a beállított értéknél: 0=nem nagyobb, 1=nagyobb

6.13.3.6. Nagy sebességű számláló programozása

A következő lépéseket kell végrehajtani gyors számláló programozásakor:

- Vezérlő bájt meghatározása.
- Aktuális (ebben az esetben kezdeti érték) és beállított érték meghatározása.
- (Opcionális) Megszakítás rutin engedélyezése az ATCH utasításban.
- Számláló és a működési mód meghatározása HDEF utasítással.
  Megjegyzés.: A HDEF utasítást csak egyszer tudja végrehajtani a nagy sebességű számláló miután a CPU RUN üzemmódba került.
- HSC utasítással indíthatjuk a nagy sebességű számlálót.

A következőkben egy példán keresztül látható a HSC0 beállítása, lépésről lépésre. Ajánlott egy külön szubrutint készíteni, mely tartalmazza a HDEF utasítást és egyéb inicializáláshoz szükséges utasításokat. Legcélszerűbb a szubrutint a SM0.1 bittel meghívni a főprogramból, ugyanis ez a bit csak a CPU indulásakor, egy ciklusig igaz.

- **HSC0 használata, beállítása**

  A példában a gyors számláló Mode 9-es módba lesz beállítva.

  1. Az inicializáló szubrutinba állítsuk be az SMB37 vezérlő byte-ot, mely a HSC0 számlálóhoz tartozik.
     Például (1x számlálási sebesség), SMB37=b#16#FC :
     - HSC0 engedélyezése
     - Aktuális érték beírása HSC0-ba
     - Beállított érték beírása HSC0-ba
     - Számlálás iránya: felfele számláló
     - Start bemenetet és a reset bemenetet aktív magas szintre állítása
  2. SMD38-ba töltjük a kívánt aktuális értéket (32-bit). Ha 0-t töltünk be SMD38 törlődik.
  3. SMD42-ba töltjük a kívánt beállított értéket (32-bit).
  4. (Opcionális) Kapcsoljuk a CV=PV eseményt (18-as esemény) egy megszakítás rutinhoz, hogy valós időben reagáljon az aktuális érték egyenlő beállított érték eseményre.
  5. (Opcionális) Kapcsoljuk a számlálási irányváltó eseményt (17-es esemény) egy megszakítás rutinhoz, hogy valós időben reagáljon a központi egység.
  6. (Opcionális) Kapcsoljuk a külső reset eseményt (16-os esemény) egy megszakítás rutinhoz, hogy a program valós időben reagáljon a külső reset eseményre.
  7. Hajtsuk végre a HDEF utasítást, HSC bementet állítsuk 0-ba, a MODE-ot pedig 9-re.
  8. Hajtsuk végre a HSC utasítást, így véglegesíthető a HSC0 beállítása, és a központi egység el is indítja a számlálót.

- **Számlálási irány megváltoztatása (Mode 0, 1 és 2 esetében)**

  1. Töltsük a kívánt vezérlő byte-ot az SMB 37-be
     SMB37=b#16#90: Számláló engedélyezése, valamint lefele számlálás
  2. Hajtsuk végre a HSC utasítást, így véglegesíthető a HSC0 beállítása, és a központi egység el is indítja a számlálót.

- **Új aktuális érték betöltése a számlalóba (összes módban használható)**

1. Töltsük a kívánt vezérlő byte-ot az SMB 37-be
   SMB37=b#16#C0 Számláló engedélyezés, új aktuális érték beírása HSC0-ba

2. Töltsük a kívánt aktuális értéket az SMD38-ba, 0 esetén SMD38 törlődik.
3. Hajtsuk végre a HSC utasítást, így véglegesíthető a HSC0 beállítása, és a központi egység el is indítja a számlálót

- **Új beállított érték betöltése a számlalóba (összes módban használható)**

1. Töltsük a kívánt vezérlő byte-ot az SMB 37-be
   SMB37=b#16#A0 Számláló engedélyezése, új beállított érték beírása HSC0-ba

2. Töltsük a kívánt beállított értéket az SMD42-be.
3. Hajtsuk végre a HSC utasítást, így véglegesíthető a HSC0 beállítása, és a központi egység el is indítja a számlálót

- **Nagy sebességű számláló tiltása (összes módban használható)**

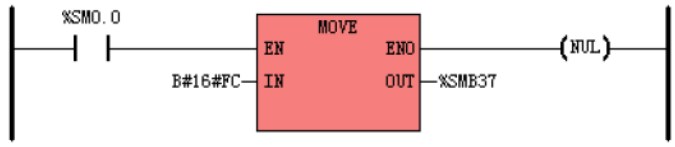
1. Töltsük a kívánt vezérlő byte-ot az SMB 37-be
   SMB37=b#16#00 Számláló tiltása.
2. Hajtsuk végre a HSC utasítást, így tiltható a HSC0 számláló.

A mintaprogram a HSC0 használatát mutatja be.
HSC0 használatához tartozó Inicializáló szubrutin.

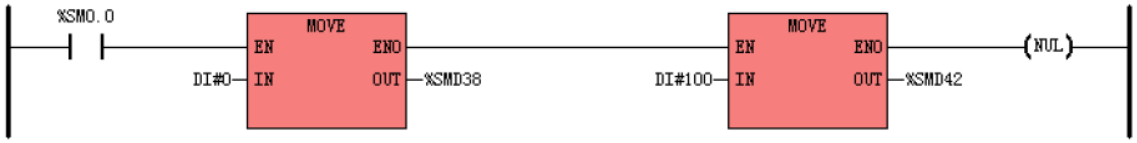
LD

Initialize szubrutin

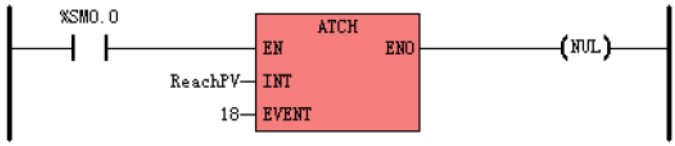
(* NETWORK 0 *)
(* 1x számlálás, HSC0 engedélyezése, aktuális érték és beállított érték változtatásának engedélyezése, felfele számláló, START és RESET bemenet aktív magas szinttel vezérlehető *)



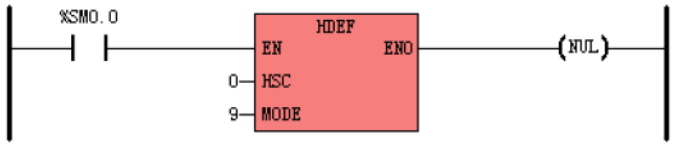
(* NETWORK 1 *)
(* Új aktuális és beállított értékek beállítása*)



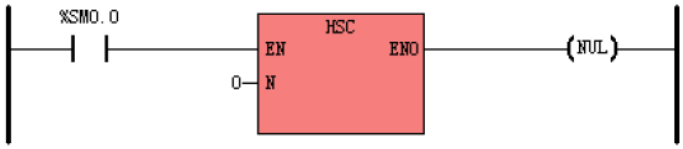
(* NETWORK 2 *)
(* CV=PV esemény (18-as) összekapcsolása ReachPV megszakítás rutinnal *)



(* NETWORK 3 *)
(* HSC0 számláló beállítása 9-es módba*)



(* NETWORK 4 *)
(* HSC0 elindítása, konfiguráció véglegesítése*)

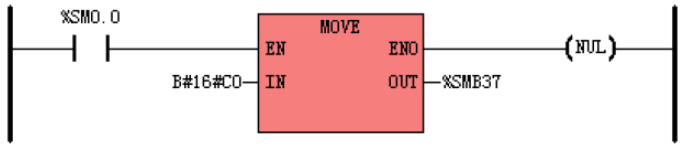


Az előző példához tartozó ReachPV megszakítás rutin és a főprogram

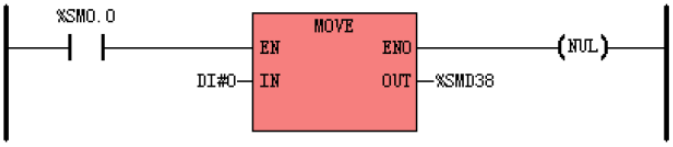
LD

ReachPV megszakítás rutin

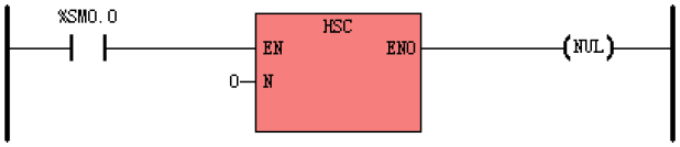
(* NETWORK 0 *)
(* Számláló aktuális érték módosításának engedélyezése*)



(* NETWORK 1 *)
(* Aktuális érték beállítása 0-ra, az újbóli számláláshoz*)

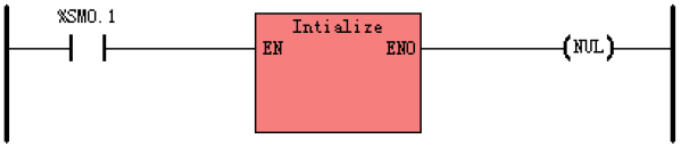


(* NETWORK 2 *)
(* HSC0 elindítása, konfiguráció véglegesítése *)



Főprogram

(* NETWORK 0 *)
(* Initialize szubrutin meghívása*)



IL

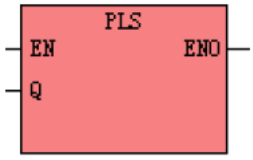
Initialize szubrutin

```
(* NETWORK 0 *)
(* 1x számlálás, HSC0 engedélyezése, aktuális érték és beállított érték változtatásának engedélyezése, felfele számláló, START és RESET bemenet aktív magas szinttel vezérlehető *)

LD %SM0.0
MOVE B#16#FC, %SMB37


(* NETWORK 1 *)
(* Új aktuális és beállított értékek beállítása*)

LD %SM0.0
MOVE DI#0, %SMD38
MOVE DI#100, %SMD42

(* NETWORK 2 *)
(* CV=PV esemény (18-as) összekapcsolása ReachPV megszakítás rutinnal *)

LD %SM0.0
ATCH ReachPV, 18

(* NETWORK 3 *)
(* HSC0 számláló beállítása 9-es módba*)

LD %SM0.0
HDEF 0, 9


(* NETWORK 4 *)
(* HSC0 elindítása, konfiguráció véglegesítése*)

LD %SM0.0
HSC 0
```

IL	**ReachPV** megszakítás rutin
	
	(* NETWORK 0 *)
	(* Számláló aktuális érték módosításának engedélyezése*)
	
	LD %SM0.0
	MOVE B#16#C0, %SMB37
	
	(* NETWORK 1 *)
	(* Aktuális érték beállítása 0-ra, az újbóli számláláshoz*)
	
	LD %SM0.0
	MOVE DI#0, %SMD38
	
	(* NETWORK 2 *)
	(* HSC0 elindítása, konfiguráció véglegesítése *)
	
	LD %SM0.0
	HSC 0
	
	Főprogram
	
	(* NETWORK 0 *)
	(* Initialize szubrutin meghívása*)
	
	LD %SM0.1
	CAL Initialize

6.13.4. Nagy sebességű kimenetek kezelése

A központi egységen található nagy sebességű kimenetek használhatók, mint impulzus kimenetek (PTO) vagy impulzus szélesség modulált (PWM) kimenetek.

	Név	Használat	Csoport	
LD	PLS			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	PLS	PLS *Q*	U	

Operandus	Bemenet/Kimenet	Adat típus	Leírás
Q	Bemenet	INT konstans (0 vagy 1)	Impulzus kimenet engedélyezés: Ha 0, akkor Q0.0 kimenet Ha 1, akkor Q0.1 kimenet

A PLS utasítással hozzárendelhető kívánt PTO/PWM konfiguráció a megadott Q kimenethez.
LD programozási nyelvben az EN bemenettel engedélyezhető a PLS végrehajtása.
Az IL programban a CR értéke határozza meg a PLS utasítás végrehajtását. Az utasítás futtatása nincs hatással CR értékére.

6.13.4.1. Nagy sebességű impulzus kimeneti módok

A KINCO-K3 két PTO/PWM generátort szolgáltat, melyek összerendelhetők fizikai kimenetekkel, impulzus kimeneti vagy impulzus szélesség modulációs módban. A kimeneti frekvencia elérheti a 20kHz értéket. Amennyiben a generátor Q0.0 kimenethez kapcsolódik, akkor lehet PWM0 vagy PTO0, Q0.1 kimenet esetén pedig PWM1 vagy PTO1. A PTO/PWM generátorok lefoglalják a hozzárendelt fizikai kimenetek memória címeit, vagyis amíg a gyors kimenet funkció aktív, addig a ciklikus program nem tudja az értéküket módosítani. Amennyiben a funkció nem aktív, a kimenetek hagyományos módon is elérhetővé vállnak.
Az SM memória területen található regiszterekkel végezhető el a beállításuk. Itt található a vezérlő bájt (8 bit), a ciklusidő és az impulzusszélesség értéke (16 bites unsigned integer), és az impulzus számláló (32 bites unsigned double integer). A kívánt értékek beállítását követően, a végrehajtás a PLS utasítással végezhető el. Az SM memória területen található kapcsolódó regiszterek alapértelmezett értéke 0.

Figyelem!
Nagy sebességű kimenet használata előtt győződjön meg róla, hogy az alkalmazott PLC tranzisztoros, és nem jelfogós kimenetű!

- **PWM (impulzusszélesség moduláció)**

A PWM üzemmód folyamatos impulzusszélesség modulált kimenetet szolgáltat, ahol megadható a ciklusidő és az impulzus szélessége, milliszekundumban vagy mikroszekundumban. A ciklusidő értéke lehet 50 ~ 65 535μs vagy 2 ~ 65 535ms.
Az impulzusszélesség értéktartománya pedig 0 ~ 65 535μs vagy 0 ~ 65 535ms közötti értéket vehet fel. Amennyiben a beállított impulzusszélesség nagyobb, mint a ciklusidő, akkor kitöltési tényező 100%-os értéket vesz fel, a hozzárendelt kimenet folyamatosan be lesz kapcsolva. Ha a megadott impulzusszélesség 0, akkor a kitöltési tényező 0% értéket vesz fel, a hozzárendelt kimenet mindig ki lesz kapcsolva.

A PWM kimenetek esetén a következő működési módok alkalmazhatók.

- **Szinkron működési mód**
  A szinkron működési mód akkor használható, ha az időalap (μs vagy ms) változtatására nincs szükség. Ez a mód a ciklusok között zökkenőmentes átmenetet tesz lehetővé. Tipikus alkalmazása, a kitöltési tényező változtatása, fix ciklusidő esetén, így nem szükséges az időalap változtatása.

- **Aszinkron működési mód**
  Akkor használható, ha szükséges az időalap változtatása. Használatakor előfordulhat, hogy a PWM jel folyamatossága megszakad, a kimeneti jelbe nem kívánt vibráció kerül. Időalap változtatása az SM67.4 vagy SM77.4 vezérlő bitekkel végezhető el.

- **PTO (Impulzus kimenet)**

A PTO üzemmód négyszögjelet állít elő (50%-os kitöltési tényezővel) a kimeneten, melyben állítható a ciklusidő (μs vagy ms időalappal) és a kimeneti impulzusok száma.
A ciklusidő 50~65535μs vagy 2~65535ms vagy 2~65535ms tartományban állítható. Amennyiben a ciklusidőnek megadott szám páratlan (pl. 35ms), a kimeneten jelentkezhet némi jeltorzulás. A kimeneti impulzusok száma 1 ~ 4 294 967 295 között lehet, ha a megadott szám 0, akkor a kimeneti impulzus száma 1 lesz.

- Single-Segment Pipelining
  Ez a mód akkor használható, ha szükséges a kapcsolódó SM regiszterek értékeit változtatni működés közben. PTO üzemmódban az SM regiszterek módosíthatók az elvárások szerint, majd újra kell futtatni a PLS utasítást. A megváltoztatott jelforma eltárolásra kerül, mindaddig amíg az előző be nem fejezte a lefutását. Az átmenti tárolóban csak egy jelforma tárolható, amint az eredeti működés befejeződött, az új jelforma kerül alkalmazásra, és az átmeneti tároló kiürül. A különböző jelformák között az átmenet zökkenőmentesen megy végbe, kivéve a következő eseteket: időalap megváltoztatása vagy az aktív impulzus befejezése után nem érkezik az új jelformára vonatkozó beállítás.

- Multi-Segment Pipelining
  Ebben a módban a központi egység automatikusan kiolvassa az impulzusra vonatkozó beállításokat a profil táblából, mely a V memória területen található. A módhoz kapcsolódó időalap az SMB67 (PTO0-hoz tartozó) vagy az SMB167 (PTO1-hez tartozó) regiszterben adható meg.
  A profiltábla kezdőcíme az SMW168 (PTO0-hoz tartozó) vagy SMW178 (PTO1-hez tartozó) regiszterben adható meg. Az időalap mikroszekundum vagy milliszekundum lehet, és a profiltáblában található összes értékre vonatkozik, és működés közben nem módosítható. A szegmensek hossza 8 bájt, mely tartalmazza a ciklusidőt (16 bit, WORD), a ciklusidő inkrementális érték (16 bit, INT) és az impulzus számlálót (32 bit, DWORD)

A profiltábla felépítését a következő táblázat mutatja:

Byte eltolás[1]	Hossz	Szegmens	Leírás
0	16-bit	-	Szegmensek száma (1~64)
1	16-bit	1	Kezdő ciklusidő (az időalap 2~65 535-szerese)
3	16-bit		Periódusidő növelés pulzusonként (-32 768~32 767-szer az időalap)
5	32-bit		Impulzus számláló (1~4,294,967,295)
9	16-bit	2	Periódus idő (az időalap 2~65535-szerese)
11	16-bit		Periódusidő növelés pulzusonként (-32768~32767-szer az időalap)
13	32-bit		Impulzus számláló (1~4,294,967,295)
…	…	…	…

1 → A byte eltolás oszlopban található értékek relatív értékek, melyek a profiltábla kezdőcímétől számolhatók.

Megjegyzés:
A profiltábla kezdőcíme páratlan számként adható meg a V memóriaterületen, mint például VB3001.

A ciklusidő automatikusan növelhető vagy csökkenthető impulzusonként, a megadott mértékben. A pozitív érték hatására a ciklusidő növekedni fog, negatív szám hatására csökkenni, nulla esetén pedig változatlan marad.

6.13.4.2. PTO/PWM kimenetek beállítása

A PTO/PWM beállítása az SM memóriaterületen található regiszterekkel végezhető el, illetve bizonyos státuszinformációk is elérhetők. A PTO/PWM jelek karakterisztikája megváltoztatható az SM regiszterek módosításával, majd a PLS utasítás futtatásával.

Q0.0	Q0.1	Vezérlő bitek
SM67.0	SM77.0	(PTO/PWM) Periódusidő frissítés: 0=nincs frissítés, 1= frissítés
SM67.1	SM77.1	(PWM) Impulzusszélesség idő frissítése: 0=nincs frissítés, 1= frissítés
SM67.2	SM77.2	(PTO) Impulzus számlálás frissítése: 0=nincs frissítés, 1= frissítés
SM67.3	SM77.3	(PTO/PWM) Időalap: 0=1µs, 1=1ms
SM67.4	SM77.4	(PWM) Működési mód 0=aszinkron frissítés, 1=szinkron frissítés
SM67.5	SM77.5	(PTO) Single-Segment / Multiple-Segment 0= single , 1= multiple
SM67.6	SM77.6	PTO vagy PWM mód: 0=PTO, 1=PWM
SM67.7	SM77.7	(PTO/PWM) Engedélyezés 0=nem engedélyezett, 1=engedélyezés
Q0.0	**Q0.1**	**Egyéb regiszterek**
SMW68	SMW78	(PTO/PWM) Periódus idő értéke, Tartománya: 2~65 535
SMW70	SMW80	(PWM) Impulzusszélesség értéke, Tartománya: 0~65 535
SMD72	SMD82	(PTO) Impulzus számlálás értéke, Tartománya: 1~4 294 967 259
SMB166	SMB176	A használatban lévő szegmensek száma Csak multi-szegmens PTO művelethez
SMW168	SMW178	A profiltábla kezdőcíme a V területen Csak multi-szegmens PTO művelethez

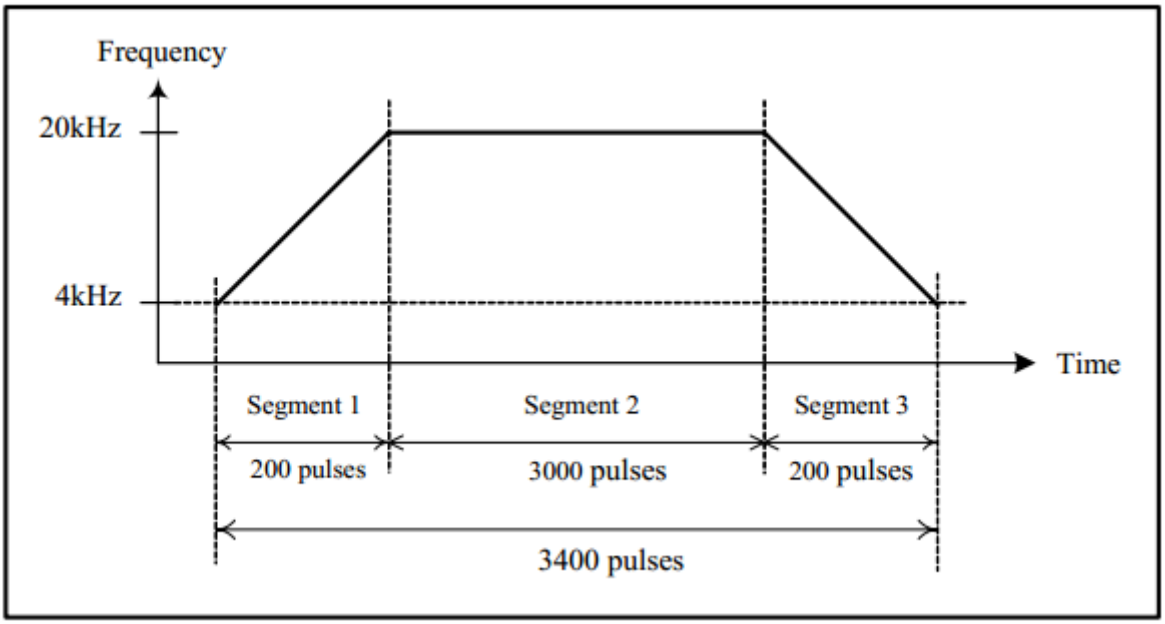
A következő táblázat a PTO/PWM státusz bitjeit mutatja be.

Q0.0	Q0.1	Státusz bit
SM66.4	SM76.4	PTO profil befejeződik, inkrementális számítási hiba miatt 0=nincs hiba; 1= működés megszakítva
SM66.5	SM76.5	PTO profil befejeződik felhasználói parancs miatt: 0=nincs megszakítva be; 1=működés megszakítva
SM66.6	SM76.6	PTO alulcsordulás / túlcsordulás 0=nincs; 1= túlcsordulás / alulcsordulás
SM66.7	SM76.7	PTO várakozik 0= folyamatban; 1= várakozás

A PTO üres-járás bit (SM66.7 vagy SM76.7) jelzi, hogy a impulzussorozat a kimeneten befejeződött, a kapcsolódó megszakítás rutin végrehajtódik. Multi szegmens működési mód esetében a megszakítás rutin akkor kerül végrehajtásra ha a profiltábla befejeződött.

6.13.4.3. Profiltáblázat számítása

Multi-Segment Pipelining funkció számos esetben alkalmazható, kiválóan használható léptetőmotor vezérléshez. Például léptetőmotorhoz megadható a gyorsítási sebesség, az állandó sebesség és a lassítási szakasz is. Összetettebb profiltáblázat is létrehozható, mely akár 64 szegmenst is tartalmazhat.
A következő ábra bemutatja egy léptetőmotor vezérlés esetén a profiltábla egy lehetséges felépítését . A profil 3 szegmensből áll, az első szegmens a gyorsítási szakasz, a második szakasz az állandó sebességű rész, az utolsó szakasz pedig a motor lassítása.



A fenti ábrán látható példában a gyorsítási szakaszban a kimeneti frekvencia 4kHz-ről 20 Khz-ig változik a sebesség, 200 impulzusig. A második szakasz, a kimeneti jel frekvenciája 20 Khz 3000 pulzusig, a lassítási szakasz pedig 200 impulzusig tart, és a kimeneti frekvencia 20kHz-ről 4kHz-re változik. Mivel a profiltáblázatban a frekvencia helyett ciklusidőt kell megadni, így a frekvencia értéket át kell számítani. A kezdeti és az utolsó ciklusidő értéke 250µs, a maximális frekvenciához kapcsolódó ciklusidő pedig 50µs.

A következőkben látható, hogyan kell számítani a ciklusonkénti növekvő értéket.

A szegmensben ciklusonként növekvő érték = (Etsegn – Itseg)/Qseg
Ahol: Etseg = A szegmens utolsó ciklusideje
Itseg = A szegmens kezdeti ciklusideje
Qseg = a szegmensben az impulzusok száma

A fenti példában a ciklusonkénti növekvő érték a következőképpen alakul:

Szegmens 1 (gyorsítás) = -1
Szegmens 2 (állandó sebesség) = 0
Szegmens 3 (lassítás) = -1

Feltételezzük, hogy a profil táblázat a V területen a VB701-en kezdődik.
A következő táblázatban a profiltáblázat értékek találhatóak.

Bájt eltolás	**Érték**	**Megjegyzés**	
VB701	3	Szegmensek száma	
VW702	250	Kezdeti periódusidő	Szegmens 1
VW704	-1	Periódusidő növekmény	
VD706	200	Impulzusok száma	
VW710	50	Kezdeti periódusidő	Szegmens 2
VW712	0	Periódusidő növekmény	
VD714	3000	Impulzusok száma	
VW718	50	Kezdeti periódusidő	Szegmens 3
VW720	1	Periódusidő növekmény	
VD722	200	Impulzusok száma	

A szegmensek közötti sima átváltás (kimeneti jel folytonossága) rendkívül fontos.

6.13.4.4. PTO műveletek

A következő példákban a PTO0 beállítása és használata látható.

- **PTO inicializálása (Single-Segment működés)**

A példában alkalmazzuk az SM0.1 bitet (csak a PLC indulásakor 1) szubrutin hívásához, mely a beállításhoz szükséges utasításokat tartalmazza. Mivel SM0.1 miatt a szubrutin csak egyszer fut le, így csökkenthető a program végrehajtási idő, valamint áttekinthetőbb programszerkezetet eredményez.

A következő lépésekben bemutatjuk a PTO0 konfigurálását inicializáló szubrutinban:

1. Töltsük be a kívánt vezérlő állapotot az SMB67-be:
   Például: SMB67=B#16#86
   - PTO/PWM funkció engedélyezése
   - PTO művelet kiválasztása
   - Időalap 1μs
   - Impulzus számlálás és periódus idő érték frissítés engedélyezése.
2. Betöltjük a periódus idő értékét az SMW68-ba
3. Betöltjük az impulzus számláló értékét az SMD72-be.
4. (Opcionális) Összekapcsolja PTO0-befejezése eseményt (esemény 28) egy megszakítás rutinnal, hogy a program valós időbe reagáljon a PTO0 befejezésére.
5. PLS utasítás végrehajtásakor a CPU beállítja PTO0-t, és elindítja.

- **PTO periódus idejének változtatása (Single-Segment működés)**
  Kövesse ezeket a lépéseket a PTO periódus idejének módosításához:

1. Töltsük be a kívánt vezérlő állapotot az SMB67-be:
   Például: SMB67=B#16#81:
   - PTO/PWM funkció engedélyezése
   - PTO művelet kiválasztása
   - Időalap 1μs
   - Impulzus számlálás és periódus idő érték frissítésének engedélyezése.

2. Betöltjük a periódus idő értékét az SMW68-ba.
3. Futtassa a PLS utasítást, hogy a CPU konfigurálja a PTO0-t és elindítsa azt. Ha volt aktív PTO folyamatban, akkor a befejezést követően a PTO hullámforma már az új periódusidővel kerül a kimenetre.

- **PTO impulzus számlálás változtatása (Single-Segment működés)**

  Kövesse ezeket a lépéseket a PTO impulzus számlálás módosításához:
1. Töltsük be a kívánt vezérlő állapotot az SMB67-be:
   Például: SMB67=B#16#84:
   - PTO/PWM funkció engedélyezése
   - PTO művelet engedélyezése
   - Időalap 1μs
   - Impulzus számlálás és periódus idő érték frissítés engedélyezése.

2. Betöltjük a periódus idő értékét az SMW72-be.
3. Futtassuk a PLS utasítást, hogy a CPU konfigurálja a PTO0-t és elindítsa. Ha volt aktív PTO folyamatban, akkor a befejezést követően a PTO hullámforma már az új periódusidővel kerül a kimenetre

- **PTO periódusidő és impulzus számlálás változtatása (Single-Segment működés)**

  Kövesse ezeket a lépéseket a PTO periódusidő és impulzus számlálás módosításához:
1. Töltsük be a kívánt vezérlő állapotot az SMB67-be:
   Például: SMB67=B#16#85:
   - PTO/PWM funkció engedélyezése
   - PTO művelet engedélyezése
   - Időalap 1μs
   - Impulzus számlálás és periódus idő érték frissítés engedélyezése.

2. Betöltjük a periódus idő értékét az SMW68-be.
3. Betöltjük az impulzus számláló értékét az SMD72-be.
4. Futtassuk a PLS utasítást, hogy a CPU konfigurálja a PTO0-t és elindítsa. Miután a folyamatban lévő aktív PTO befejeződik generálódik az új PTO hullámforma a frissített periódusidővel.

- **PTO inicializálása (Multi-szegmens művelet)**

A példában alkalmazzuk az SM0.1 bitet (csak a PLC indulásakor 1) szubrutin hívásához, mely a beállításhoz szükséges utasításokat tartalmazza. Mivel SM0.1 miatt a szubrutin csak egyszer fut le, így csökkenthető a program végrehajtási idő, valamint áttekinthetőbb program szerkezetet eredményez.

A következő lépések bemutatják, hogyan lehet a PTO0-t inicializálni a egy szubrutinba:

1. Töltsük be a kívánt vezérlő állapotot az SMB67-be:
   Például: SMB67=B#16#A0:
   - PTO/PWM funkció engedélyezése
   - PTO művelet kiválasztása
   - Multi-szegmens működés kiválasztása
   - Időalap 1μs
2. Betöltünk egy páratlan számot az SMW168-ba, mely a profiltábla kezdőcíme lesz.
3. V területet használjuk a profil táblázat megadására
4. (Opcionális) Összekapcsolhatjuk PTO0-befejezése eseményt (esemény 28) egy megszakítás rutinnal, hogy a program valós időben reagáljon PTO0 vége eseményre.
5. Futtassuk a PLS utasítást, hogy a CPU konfigurálja a PTO0-t és elindítsa. Miután a folyamatban lévő aktív PTO befejeződik generálódik az új PTO hullámforma a frissített periódusidővel.

6.13.4.5. PWM műveletek

A következőkben bemutatunk egy PWM0 példát, hogy hogyan konfiguráljuk és használjuk a PTO/PWM generátort a programban.

- **PWM kimenet inicializálása**

A példában alkalmazzuk az SM0.1 bitet (csak a PLC indulásakor 1) szubrutin hívásához, mely a beállításhoz szükséges utasításokat tartalmazza. Mivel SM0.1 miatt a szubrutin csak egyszer fut le, így csökkenthető a program végrehajtási idő, valamint áttekinthetőbb programszerkezetet eredményez.

A következő lépések bemutatják, hogyan lehet a PWM0-t inicializálni a egy szubrutinba:

1. Töltsük be a kívánt vezérlő állapotot az SMB67-be:
   Például: SMB67=B#16#D3:
   - PTO/PWM funkció engedélyezése
   - PWM művelet kiválasztása
   - Időalap 1μs
   - Impulzus szélesség és ciklusidő módosításának engedélyezése
2. Betöltünk az SMW68-ba a periódusidő értékét.
3. Betöltünk az SMW70-be az impulzusszélesség értékét.
4. Futtassuk a PLS utasítást, hogy a CPU konfigurálja a PTO0-t és elindítsa.

- **PWM kimenet impulzusszélesség változtatása**

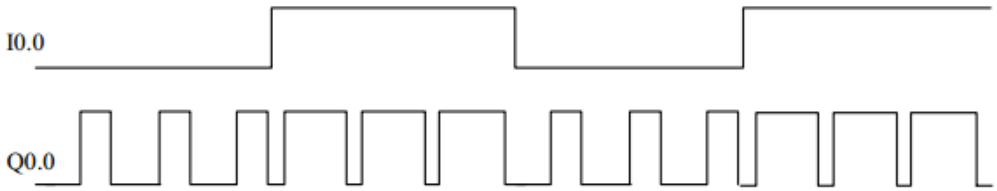
A következő lépések bemutatják, hogyan változtassuk a PWM kimeneten az impulzusszélességet (feltételezzük, hogy az SMB67-be újratöltjük B#16#D2-t vagy B#16#DA-t):

1. Betöltjük az SMW70-be az impulzusszélesség értékét (16-bit).
2. Futtassuk a PLS utasítást, hogy a CPU konfigurálja a PTO0-t és elindítsa.

Példa PWM használatára

- **PWM**

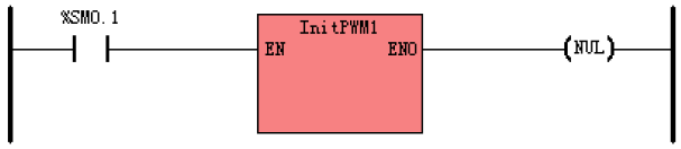
PWM1-t (Q0.1 kimenet) használunk a példába.
Ha az I0.0 értéke hamis, akkor a kitöltési tényező 40%, ha igaz akkor pedig 80%.
A példát az alábbi időidiagram szemlélteti.



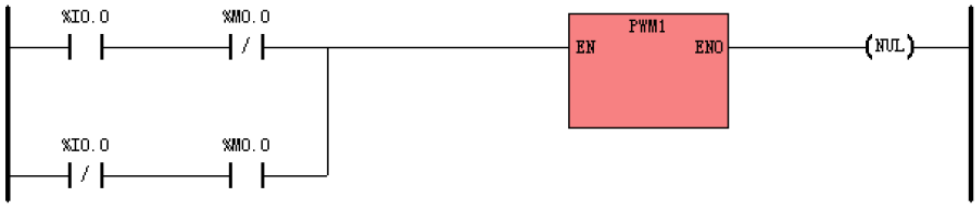
LD

Főprogram

(* Network 0 *)
(* SM0.1 bittel InitPWM1 szubrutin hívása, PWM1 inicializálásához*)



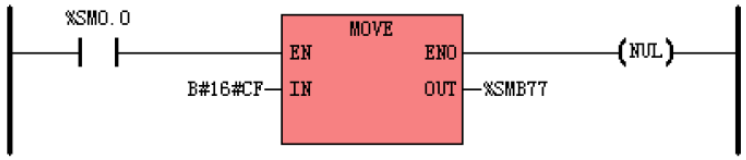
(* Network 1 *)
(* Ha I0.0 állapota megváltozik, PWM1 szubrutin meghívásával változtatható az impulzus szélessége *)



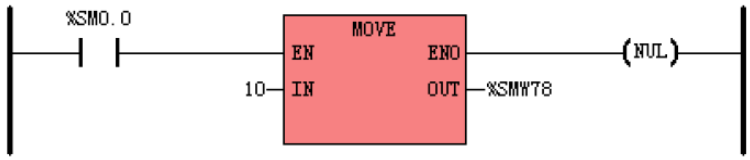
LD

InitPWM1 szubrutin

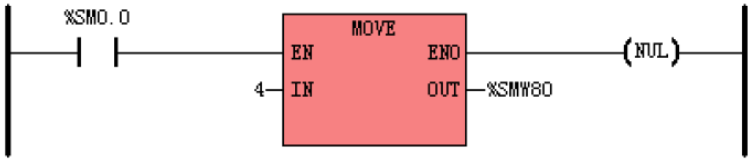
(* Network 0 *)
(* PWM1 kiválasztása, időalap 1ms, ciklusidő és impulzusszélesség módosításának engedélyezése*)



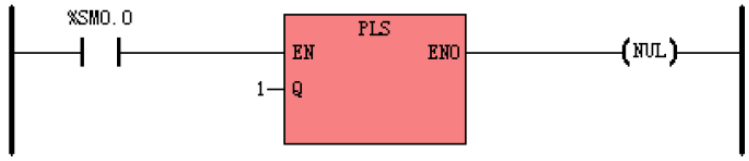
(* Network 1 *)
(* PWM1 ciklusidejének beállítása 10ms-ra*)



(* Network 2 *)
(* PWM1 impulzus szélességének beállítása 4ms-ra*)



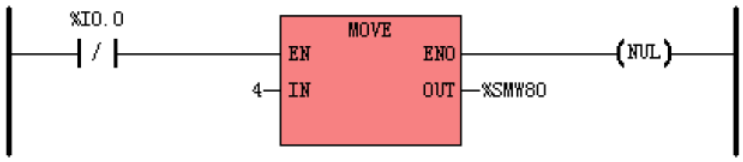
(* Network 3 *)
(* PWM1 végrehajtása*)



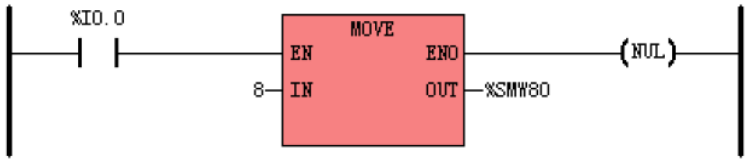
LD

PWM1 szubrutin

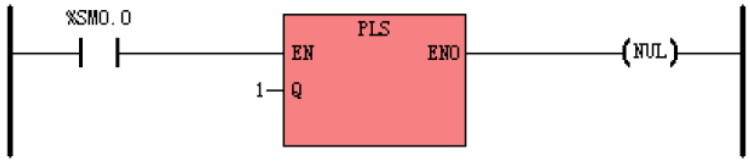
(* Network 0 *)
(* Ha I0.0 bemenet ki van kapcsolva, akkor PWM1 beállítása 4ms-ra*)



(* Network 1 *)
(* Ha I0.0 bemenet be van kapcsolva, akkor PWM1 beállítása 8ms-ra*)



(* Network 2 *)
(* PWM1 végrehajtása*)



IL

Főprogram

```
(* Network 0 *)
(* SM0.1 bittel InitPWM1 szubrutin hívása, PWM1 inicializálásához*)

LD %SM0.1
CAL InitPWM1

(* Network 1 *)
(* Ha I0.0 állapota megváltozik, PWM1 szubrutin meghívásával változtatható az impulzus szélessége *)

LD %I0.0
ANDN %M0.0
OR(
LDN %I0.0
AND %M0.0
)
CAL PWM1

(* Network 2 *)

LD %I0.0
ST %M0.0
```

IL

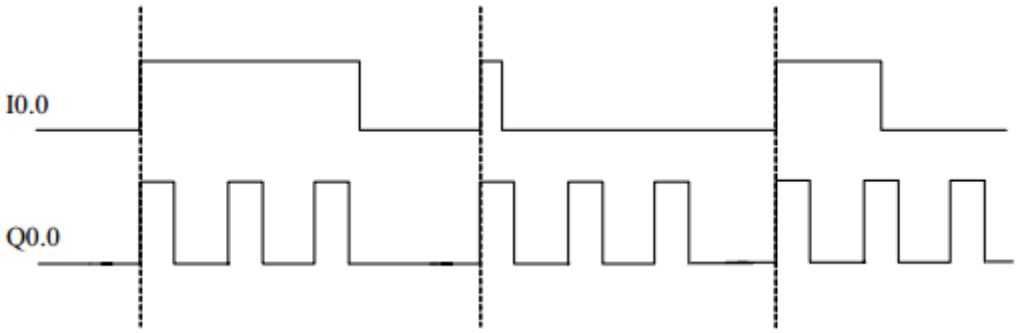
InitPWM1 szubrutin

```
(* Network 0 *)
(* PWM1 kiválasztása, időalap 1ms, ciklusidő és impulzusszélesség módosításának engedélyezése*)

LD %SM0.0
MOVE B#16#CF, %SMB77

(* Network 1 *)
(* PWM1 ciklusidejének beállítása 10ms-ra*)

LD %SM0.0
MOVE 10, %SMW78

(* Network 2 *)
(* PWM1 impulzusszélesség beállítása 4ms-ra*)

LD %SM0.0
MOVE 4, %SMW80

(* Network 3 *)
(* PWM1 végrehajtása*)

LD %SM0.0
PLS 1
```

IL	**PWM1 szubrutin**
	(* Network 0 *)
	(* Ha I0.0 bemenet ki van kapcsolva, akkor PWM1 beállítása 4ms-ra*)
	
	LDN %I0.0
	MOVE 4, %SMW80
	
	(* Network 1 *)
	(* Ha I0.0 bemenet be van kapcsolva, akkor PWM1 beállítása 8ms-ra*)
	
	LD %I0.0
	MOVE 8, %SMW80
	
	(* Network 2 *)
	(* PWM1 végrehajtása*)
	
	LD %SM0.0
	PLS 1

Példa PTO használatára (Single-Segment működés)

A PTO0 funkció használatát mutatjuk be a következő példában, mely a Q0.0 kimenethez kapcsolódik.
A példában a I0.0 bemenet felfutó élének hatására a kimeneten jelenjen meg 3 impulzus.

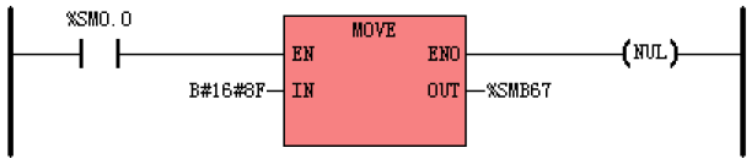


LD	**Főprogram** (* Network 0 *) (* PTO0 funkció elindítása I0.0 felfutó élének hatására*) 

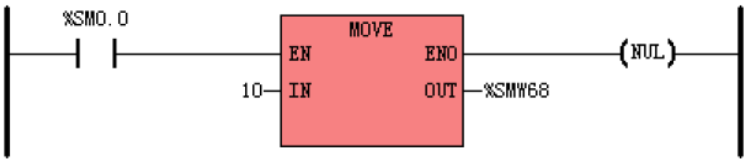
LD

PTO0 alprogram

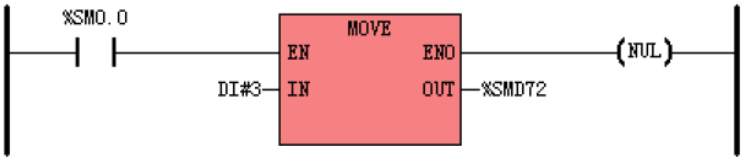
(* Network 0 *)
(* Single-Segment működés kiválasztása PTO0 esetén, 1ms időalap beállítása, ciklusidő és impulzusszám módosításának engedélyezése*)



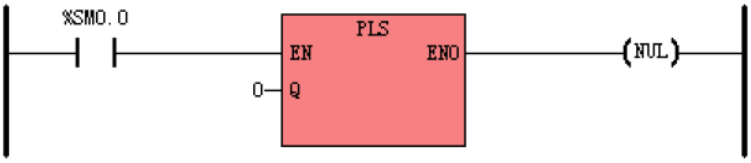
(* Network 1 *)
(*Ciklusidő beállítása 10ms-ra*)



(* Network 2 *)
(*Impulzusszám beállítása 3-ra*)



(* Network 3 *)
(*PTO0 végrehajtása*)



IL

Főprogram

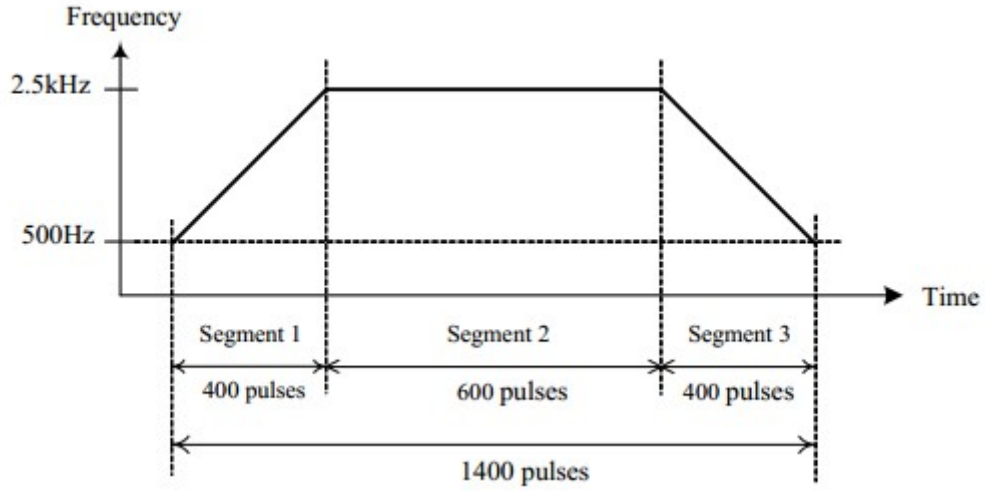
```
(* Network 0 *)
(* PTO0 funkció elindítása I0.0 felfutó élének hatására*)

LD %I0.0
R_TRIG
CAL PTO0
```

PTO0 alprogram

```
(* Network 0 *)
(* Single-Segment működés kiválasztása PTO0 esetén, 1ms időalap beállítása, ciklusidő és
impulzusszám módosításának engedélyezése*)

LD %SM0.0
MOVE B#16#8F, %SMB67

(* Network 1 *)
(*Ciklusidő beállítása 10ms-ra*)

LD %SM0.0
MOVE 10, %SMW68

(* Network 2 *)
(*Impulzusszám beállítása 3-ra*)

LD %SM0.0
MOVE DI#3, %SMD72

(* Network 3 *)
(*PTO0 végrehajtása*)

LD %SM0.0
PLS 0
```

Példa PTO használatára (Multi-Segment működés)

A PTO0 funkció használatát mutatjuk be a következő példában, a funkció az I0.0 bemenettel indítható.
A következő ábra szemlélteti a megvalósítani kívánt többszegmenses profilt.



LD

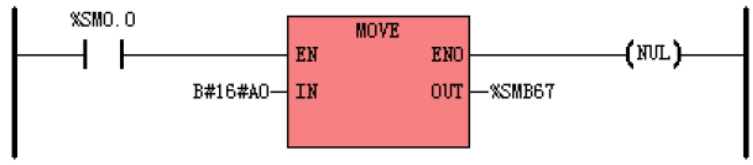
Főprogram

(* Network 0 *)
(* PTO0 funkció elindítása I0.0 felfutó élének hatására*)

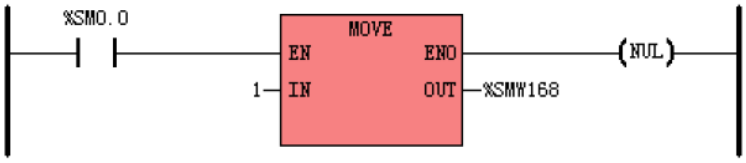
LD

PTO0 szubrutin

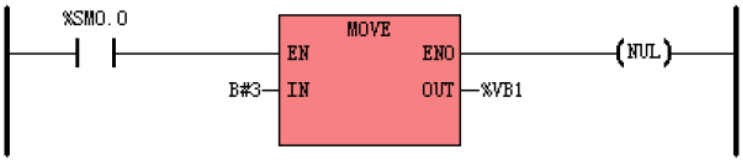
(* Network 0 *)
(* PTO0 funkció engedélyezése, többszegmenses működés, 1us időalappal*)



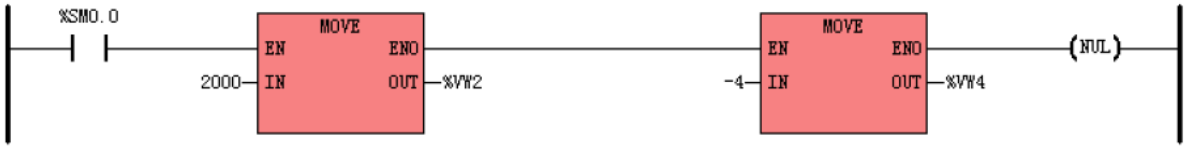
(* Network 1 *)
(* VB1 megadása, mint a profiltábla kezdőcíme*)



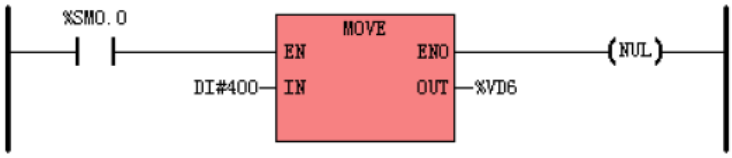
(* Network 2 *)
(* Szegmensek számának beállítása 3-ra*)



(* Network 3*)
(* 1-es szegmens: kezdő ciklusidő 2000us, ciklusidő növekmény értéke -4us*)



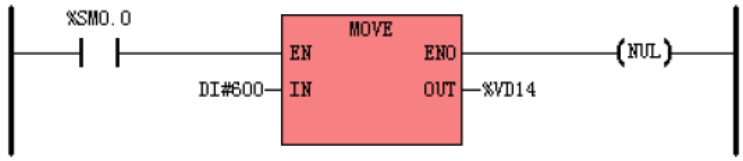
(* Network 4*)
(* 1-es szegmens: impulzusszám beállítása 400-ra*)



(* Network 5*)
(* 2-es szegmens: kezdő ciklusidő 400us, ciklusidő növekmény értéke -4us*)

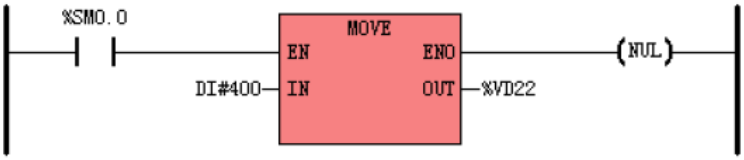
LD

(* Network 6*)
(* 2-es szegmens: impulzusszám beállítása 600-ra*)

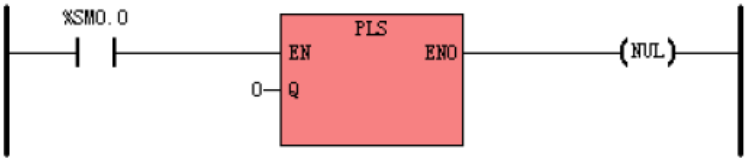


(* Network 7*)
(* 3-as szegmens: kezdő ciklusidő 400us, ciklusidő növekmény értéke 4us*)

(* Network 8*)
(* 3-as szegmens: i3mpulzusszám beállítása 400-ra*)



(* Network 9*)
(*PTO0 végrehajtása*)



IL

Főprogram

(* Network 0*)

```
LD %I0.0
R_TRIG
CAL PTO0 (* PTO0 indítása I0.0 felfutó élének hatására)
```

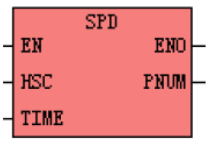
IL

PTO0 szubrutin

(* NETWORK 0 *)

```
LD %SM0.0
MOVE B#16#A0, %SMB67 (* PTO0 engedélyezése, többszegmenses működés, 1us
                        (*időalap*)
MOVE 1, %SMW168      (*VB1 kijelölése, mint a profiltábla kezdőcíme*)
MOVE B#16#03, %VB1   (*szegmensek számának beállítása 3-ra*)

(* 1. szegmens*)

MOVE 2000, %VW2      (* Kezdeti ciklusidő beállítása 2000us-ra*)
MOVE -4, %VW4        (* ciklusidő növekmény értéke -4us*)
MOVE DI#400, %VD6    (* impulzusok száma 400*)

(* 2. szegmens*)

MOVE 400, %VW10      (* Kezdeti ciklusidő beállítása 400us-ra*)
MOVE 0, %VW12        (* ciklusidő növekmény értéke 0us*)
MOVE DI#600, %VD14   (* impulzusok száma 600*)

(* 3. szegmens*)

MOVE 400, %VW18      (* Kezdeti ciklusidő beállítása 400us-ra*)
MOVE 4, %VW20        (* ciklusidő növekmény értéke 4us*)
MOVE DI#400, %VD22   (* impulzusok száma 400*)

PLS 0                (* PTO0 végrehajtása*)
```

6.13.5. SPD (Sebesség számítás)

	Név	Használat	Csoport	
LD	SPD			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	SPD	SPD *HSC, TIME, PNUM*	U	

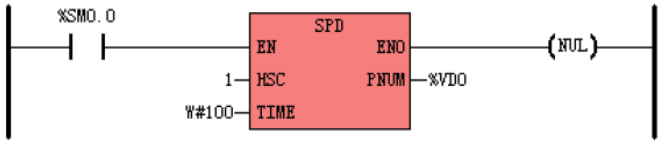
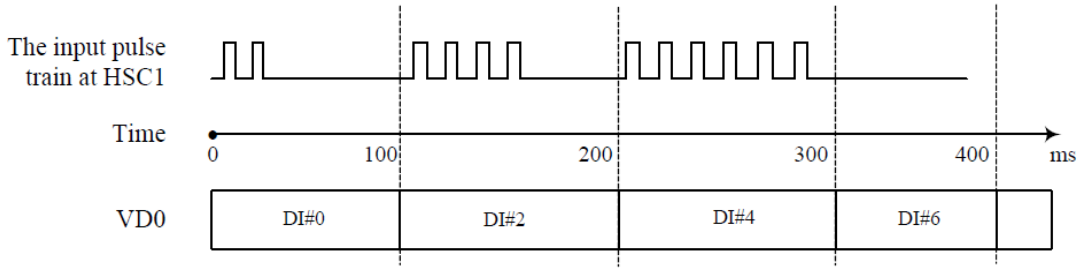
Operandus	Bemenet/Kimenet	Adat típus	Használható memória terület
HSC	Bemenet	INT	Konstans (0-5 HSC száma)
TIME	Bemenet	WORD	I, Q, M, V, L, AM, konstans
PNUM	Kimenet	DINT	Q, M, V, L, SM

Az utasítás nagy sebességű számláló bemenetekről érkező impulzusokat számolja, egy megadott időintervallumon belül (ms-ban), és az eredmény a PNUM kimenetre kerül.

- **LD**
  Ha EN=1, az utasítás végrehajtódik

- **IL**
  Ha CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

Példa:

<table>
<tr><td>LD</td><td> 
</td><td>Mivel SM0.0 mindig 1, ezért az SPD utasítás mindig végrehajtásra kerül. A HSC1 bemenetre érkező impulzusokat számolja 100ms időintervallumon belül. Az eredmény VD0-ra kerül.</td></tr>
<tr><td>IL</td><td colspan="2">LD %SM0.0
SPD 1, W#100, %VD0</td></tr>
<tr><td>Eredmény</td><td colspan="2">A programfutás eredménye a következő:
 
</td></tr>
</table>

6.14. Időzítők

Az időzítők a IEC61131-3 szabványnak megfelelő funkció blokkok, melynek három típusa van, TON (bekapcsolás késleltetés) , TOF (kikapcsolás késleltetés) és TP (impulzus adó) .

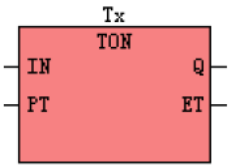
6.14.1. Időzítők felbontása

Három részre lehet felbontani az időzítőket, felbontásuk szerint.

	CPU304	**CPU306**
Felbontás	T0---T3: 1ms T4---T19: 10ms T20–T63: 100ms	T0---T3: 1ms T4---T19: 10ms T20–T127: 100ms
Max időzítés	32767*felbontás	32767*felbontás

Egy időzítő beállított és aktuális értéke a felbontás egész számú többszöröse, például ha 100 található egy 10ms-os időzítőben az 1000ms-nak felel meg.

6.14.2. TON (Bekapcsolás késleltetés)

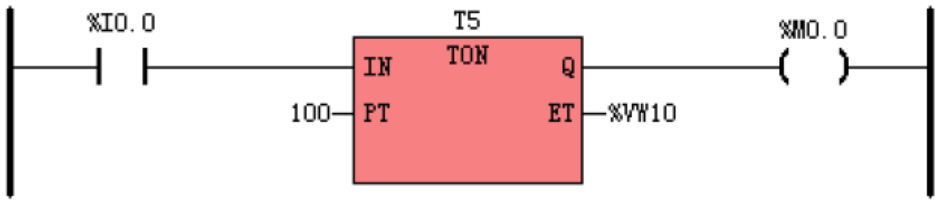
	Név	Használat	Csoport1	
LD	TON			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	TON	TON *Tx, PT*	P	

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
TX	-	Időzítő hivatkozás	T
IN	Bemenet	BOOL	Program összeköttetés
PT	Bemenet	INT	I, AI, AQ, M, V, L, SM, konstans
Q	Kimenet	BOOL	Program összeköttetés
ET	Kimenet	INT	Q, M, V, L, SM, AQ

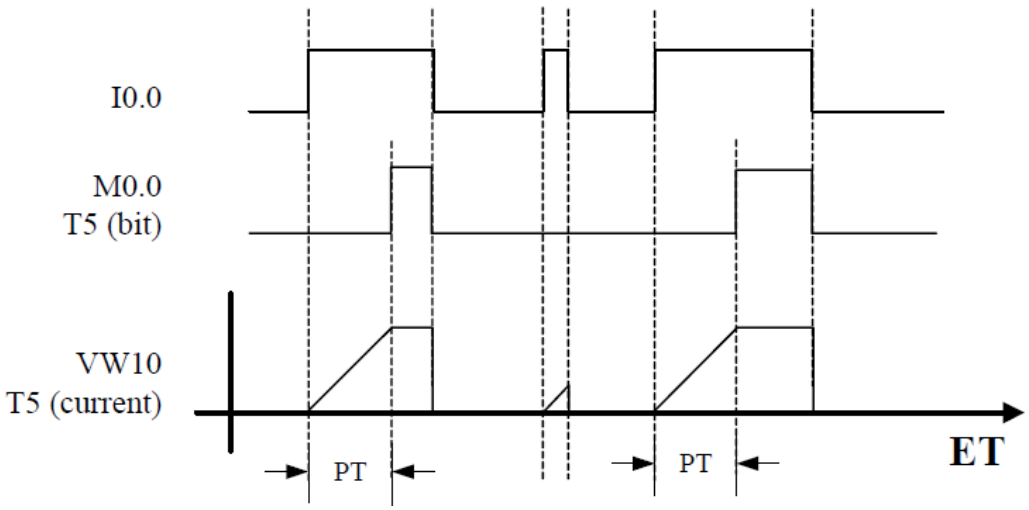
A Tx az időzítő funkcióblokk hivatkozása.

- **LD**
  A Tx időzítőt az IN bemenet felfutó élével indítható. Ha az eltelt idő nagyobb vagy egyenlő mint a beállított (PT), akkor a Q kimenet és a Tx státuszbit is logikai 1 értéket vesz fel. Ha az IN bemenet kikapcsol, Tx törlődik, és Q kimenet valamint Tx státuszbit is kikapcsol, valamint az aktuális érték 0 lesz.

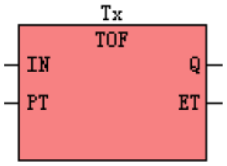
- **IL**
  Tx időzítő a CR felfutó élével indítható. Ha az aktuális érték nagyobb vagy egyenlő, mint a beállított, akkor a Tx státuszbitje bekapcsol. Ha CR 0 lesz, akkor Tx értéke törlődik, és a státuszbitje FALSE lesz. Minden ciklus végén CR értéke Tx státuszbit értékét veszi fel.

LD	IL
(* Network 0*) (* T5 időzítő TON típusú, és a beállított értéke 1000ms *) (* 100 a megadott értéke a T5 pedig 10 ms felbontású *)  	(* Network 0*) LD %I0.0 TON T5, 100 ST %M0.0

Időfüggvény



6.14.3. TOF (Kikapcsolás késleltetés)

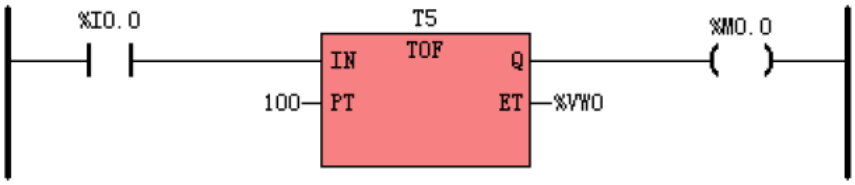
	Név	Használat	Csoport	
LD	TOF			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	TOF	TOF *Tx, PT*	P	

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
TX	-	Időzítő hivatkozás	T
IN	Bemenet	BOOL	Program
PT	Bemenet	INT	I, AI, AQ, M, V, L, SM, konstans
Q	Kimenet	BOOL	Program
ET	Kimenet	INT	Q, M, V, L, SM, AQ

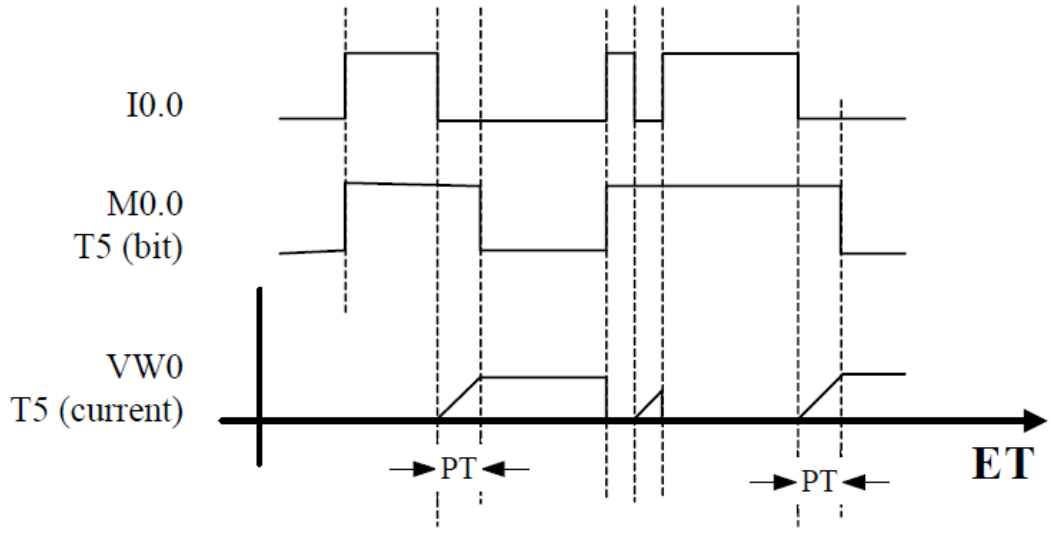
A Tx az időzítő funkcióblokk hivatkozása.

- **LD**
  Az időzítőt az IN bemenet lefutó éle indítja. Amikor a hátralevő idő nagyobb vagy egyenlő mint a beállított idő, akkor a Q kimenet és a Tx státuszbitje is nulla értéket vesz fel. Ha az IN bemenet bekapcsol, akkor Q kimenet és Tx státuszbit is bekapcsol.

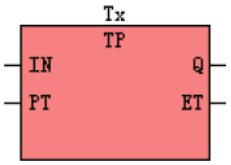
- **IL**
  Tx időzítését a CR lefutó éle indítja. Amikor a hátralevő idő nagyobb vagy egyenlő, mint a beállított akkor a Tx nulla értéket vesz fel. Amikor CR bekapcsol, akkor a státuszbit értéke 1 lesz. Minden ciklus végén CR értéke Tx státuszbit értékét veszi fel.

LD	IL
(* Network 0*) (* T5 időzítő TOF típusú, és a beállított értéke 1000ms *) (* 100 a megadott értéke a T5 pedig 10 ms felbontású *) 	(* Network 0*) LD %I0.0 TOF T5, 100 ST %M0.0

Időfüggvény



6.14.4. TP (Impulzus időzítő)

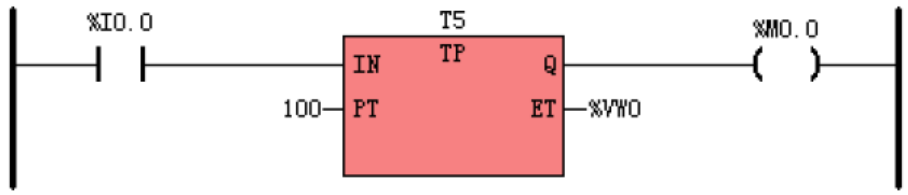
	Név	Használat	Csoport	
LD	TP			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	TP	TP *Tx, PT*	P	

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
TX	-	Időzítő hivatkozás	T
IN	Bemenet	BOOL	Program
PT	Bemenet	INT	I, AI, AQ, M, V, L, SM, konstans
Q	Kimenet	BOOL	Program
ET	Kimenet	INT	Q, M, V, L, SM, AQ

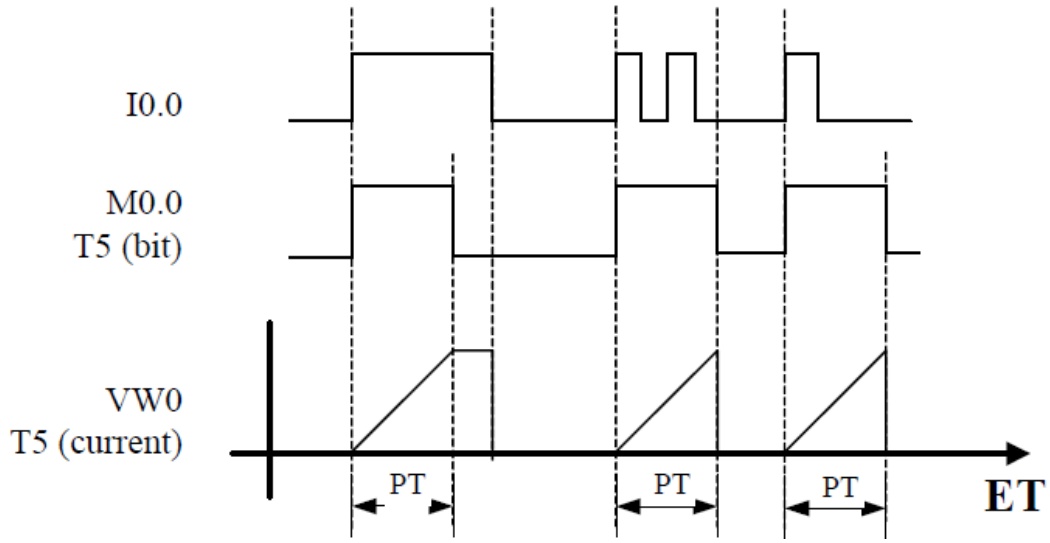
Tx a TP funkcióblokk hivatkozása. A TP utasítással megadott időtartamú impulzus hozható létre.

- **LD**
  Az IN bemenet felfutó élének hatására, Tx időzítő elindul, Q kimenet valamint Tx státuszbit bekapcsol, az időzítés ideje alatt (amíg eltelt idő kisebb, mint PT). Ha az aktuális érték eléri a beállított értéket Q kimenet és a státuszbit is nulla értéket vesz fel.

- **IL**
  CR felfutó élének hatására Tx időzítő elindul, Tx státuszbit értéke 1 lesz. Az érték 1 marad PT idő leteltéig. A megadott idő eltelte után státuszbit értéke 0 lesz. Minden ciklus végén CR értéke Tx státuszbit értékét veszi fel.

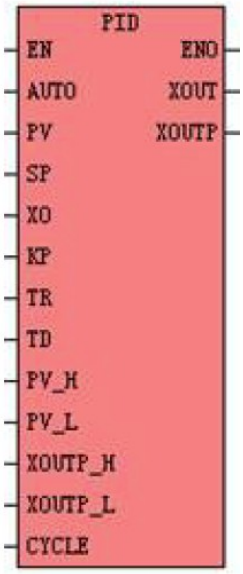
LD	IL
(* Network 0*) (* T5 időzítő TP típusú, és a beállított értéke 1000ms *) (* 100 a megadott értéke a T5 pedig 10 ms felbontású *) 	(* Network 0*) LD %I0.0 TP T5, 100 ST %M0.0

Időfüggvény



6.15. PID

A Kinco K3 sorozatú PLC rendelkezik PID utasítással, egy központi egységen belül maximálisan 8 PID blokk alkalmazható.

	Név	Használat	Csoport	
LD	PID			☐ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308 ☑ CPU406 ☑ CPU408
IL	PID	PID (AUTO, PV, SP, XO, KP, TR, TD, PV_H, PV_L, XOUTP_H, XOUTP_L, CYCLE, XOUT, XOUTP)	U	

Operandus	Be/Kimenet	Adat típus	Memória terület	Megjegyzés
AUTO	Bemenet	BOOL	I, Q, V, M, SM, L, T, C	Kézi/automata 0=Kézi,1=Automata üzemmód
PV	Bemenet	INT	AI, V, M, L	Aktuális érték
SP	Bemenet	INT	V, M, L	Beállított érték
XO	Bemenet	REAL	V, L	Manuális érték, tartomány [0.0,1.0]
KP	Bemenet	REAL	V, L	Arányossági tényező
TR	Bemenet	REAL	V, L	Integrálási idő (s).
TD	Bemenet	REAL	V, L	Deriválási idő (s).
PV_H	Bemenet	INT	V, L	Aktuális érték felső határa
PV_L	Bemenet	INT	V, L	Aktuális érték felső határa
XOUT_H	Bemenet	INT	V, L	XOUTP maximális értéke
XOUT_L	Bemenet	INT	V, L	XOUTP minimális értéke
CYCLE	Bemenet	DINT	V, M, L	Mintavételezési periódus (ms).
XOUT	Kimenet	REAL	V, L	Kimenet értéke [0.0,1.0]
XOUTP	Kimenet	INT	AQ, V, M, L	Átszámított kimeneti érték

- **LD**
  Ha EN=1, az utasítás végrehajtódik

- **IL**
  Ha CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

Kézi / automata üzemmód

- Ha az AUTO bemenet értéke 0, akkor a PID blokk kézi üzemmódban működik, ilyen esetben XO bemenet értéke jelenik meg az XOUT kimeneten.

- Ha az AUTO bemenet 1, akkor a PID blokk automatikus üzemmódban működik, vagyis a megadott paraméterek szerinti számításokat elvégzi, és az eredmény megjelenik az XOUT kimeneten.

PV és SP normalizálása

A PV és SP értéke egész számként adható meg a programban, de a PID algoritmus lebegőpontos számot igényel, ezért szükséges az értékek átszámítása. Az átszámítást PV, SP, PV_H és PV_L bemenetek szerint végzi a program.

PV normalizált értéke =k*PV+b
SP normalizált értéke = k*SP+b

Például, nyomás szabályzás esetén az elvárt nyomás 25MPa. A nyomás mérése nyomástávadót használunk, melynek mérési tartománya 0-40MPA, kimenete pedig 4-20mA. A jelátalakító a PLC analóg bemenetére csatlakozik, AIW0 címre, 4mA értéke 4000, 20mA pedig 20000.
A PID blokk bemeneteit az alábbi táblázat szerint kell megadni:

	Aktuális paraméter	**Megjegyzés**
PV	AIW0	Az aktuális érték az analóg bemenet értéke
SP	14000	14mA, mert az felel meg a 25MPa nyomásnak.
PV_L	4000	A távadó kimenetének minimális értéke
PV_H	20000	A távadó kimenetének maximális értéke

Kimeneti értékek

A PID funkcióblokk két kimenettel rendelkezik, XOUT és XOUTP.

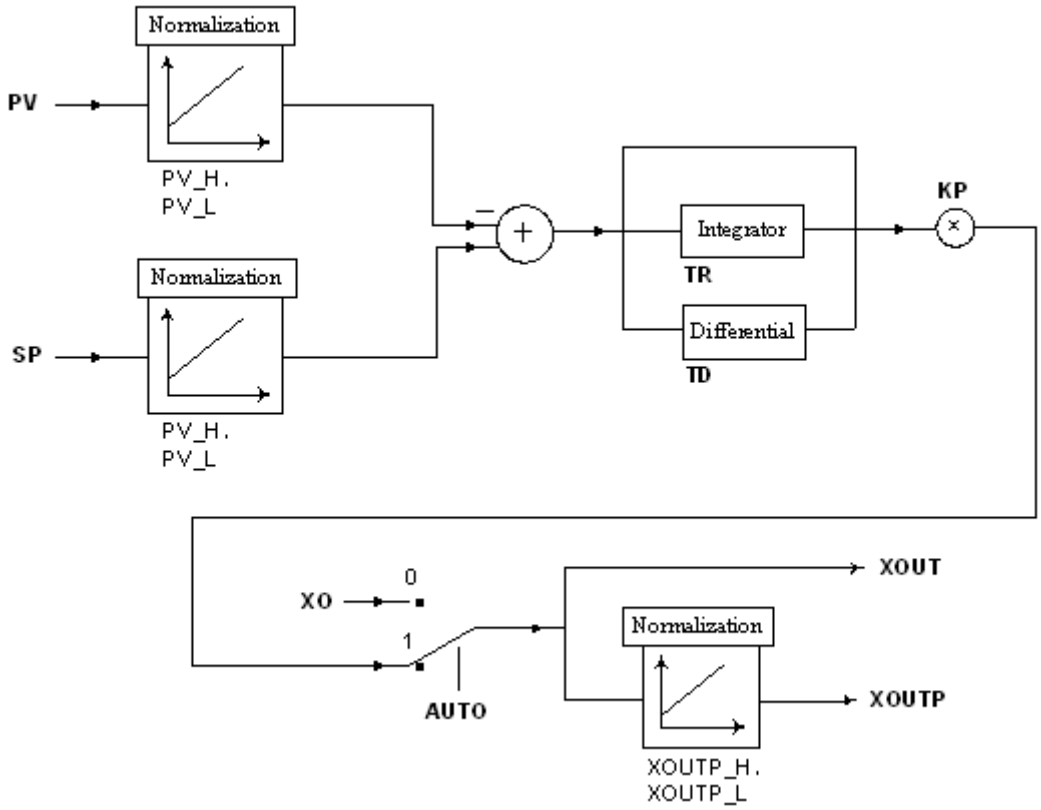
- XOUT 0.0 és 1.0 közötti értéket vehet fel, mely megfelel 0.0 és 100% közötti értékeknek.

- XOUTP kimenet értéke egész szám, egy számított érték, melyet a program az XOUT kimenetből és XOUTP_H és XOUTP_L számít, az alábbi képlet alapján.

  XOUTP=(XOUTP_H-XOUTP_L)*XOUT+XOUTP_L

Az XOUTP értéke már közvetlenül analóg kimenetre csatlakoztatható.

PID funkcióblokk működési diagram



IL	(* Network 0*) (* Aktuális paraméterek megadása*) LD %SM0.0 MOVE 7200, %VW0 (* SP *) MOVE 4000, %VW2 (* PV_L *) MOVE 20000, %VW4(* PV_H *) MOVE 4000, %VW6 (* XOUTP_L *) MOVE 20000, %VW8(* XOUTP_H *) (* Network 1 *) (* PID futtatása *) LD %SM0.0 PID %M0.0, %AIW0, %VW0, %VR100, %VR104, %VR108, %VR112, %VW2, %VW4, %VW6, %VW8, %VD10, %VR116, %AQW0

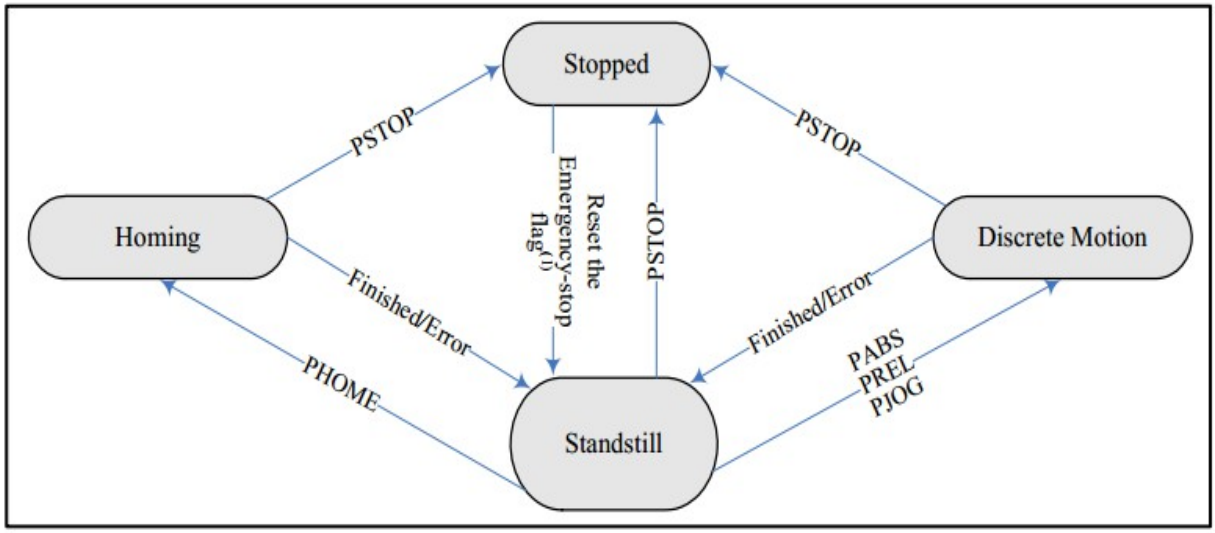
6.16. pozíció vezérlés

A Kinco-K3 PLC két gyors kimenetet (Q0.0 és Q0.1) biztosít pozicionálási feladatok elvégzésére, melyekkel két különálló tengely vezérelhető.

A pozíció vezérlés egy, a PTO/PWM módtól eltérő használati módja a gyors kimeneteknek. A pozíció vezérlés esetén a kimeneti frekvencia ugyanúgy elérheti a 20kHz-et, mint egyéb módok esetében.

6.16.1. Az alkalmazott modell

A következő ábrán egy egytengelyes vezérlés ábrája látható. A vezérelt tengely mindig az ábrán látható valamelyik állapot egyikét veszi fel. Az állapotok közötti váltás a nyilakon látható parancsokkal végezhető el.



(1) A Vészleállító visszajelző (Emergency-Stop flag) SM201.7/SM231.7 bitek, automatikusan 1-be kapcsolnak, amennyiben a PSTOP utasítás végrehajtásra kerül.

6.16.2. Viszonylagos változók

6.16.2.1. Iránybit kimenetek

Pozíció vezérlés esetén a két gyors kimenet mellé tartozik két iránybit kimenet is, melyekhez tartozó vezérlőbit az SM memóriaterületen található.

Nagy sebességű impulzus kimenet	**Q0.0**	**Q0.1**
Iránybit kimenet	Q0.2	Q0.3
Iránybit kimenet vezérlőbit	SM201.3	SM231.3

Az iránybit kimenet segítségével megadható a hajtott motor forgásiránya, 0 esetén a motor előre forog, 1 esetén pedig visszafele.
Az iránybit vezérlőbitek segítségével tiltható a forgásirány vezérlés működése. Ha a vezérlőbit értéke 0, akkor forgásirány vezérlés ki lesz kapcsolva. Ebben az esetben a Q0.2 és Q0.3 kimenetek hagyományos kimeneti pontként használhatók.

6.16.2.2. Státusz és vezérlő regiszterek

A Kinco K3 PLC minden gyors kimenetéhez tartozik egy vezérlő byte, mely a beállításokat tartalmazza. A státusz regiszterben található az aktuális érték is, mely növekszik ha a motor előre forog, csökken ha visszafele.
Megjegyzés: egy pozíció vezérlő utasítás befejezését követően az aktuális érték nem törlődik automatikusan, törlésről az alkalmazói programban kell gondoskodni.

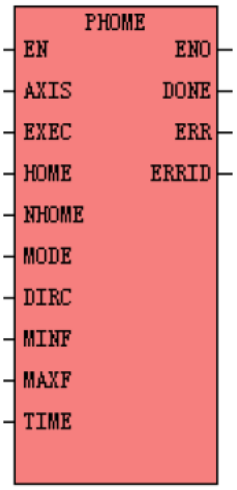
Q0.0	**Q0.1**	**Leírás**
SM201.7	SM231.7	Vész-Stop visszajelző Ha a bit értéke 1, pozicionáló utasítások nem kerülnek végrehajtásra. PSTOP utasítást követően a bit értéke 1 lesz, törléséről a programban kell gondoskodni.
SM201.0~SM201.2	SM201.0~SM201.2	Fenntartva
SM201.3	SM231.3	Iránybit kimenetek vezérlőbitje 1 → Irányváltó kimenet tiltása 0 → Irányváltó kimenet engedélyezése
SM201.0~SM201.2	SM201.0~SM201.2	Fenntartva
SMD212	SMD242	Az aktuális érték

6.16.2.3. Hiba azonosítók

Ha a pozíció vezérlés utasítások végrehajtása során hiba lép fel, a központi egység hibakódot hoz létre a hiba okának könnyebb azonosítására.

Hiba kód	**Leírás**
0	Nincs hiba
1	A tengely értéke nem 0 vagy 1.
2	A MINF értéke nagyobb mint a MAXF maximum értéke.
3	A MINF értéke kisebb mint a legalacsonyabb frekvencia (20Hz).
4	A TIME (gyorsulás/lassulás idő) értéke nem egyezik MINF és MAXF értékével.

6.16.3. PHOME (Kezdő pozíció felvétele)

	Név	Használat	Csoport	
LD	PHOME			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	PHOME	PHOME (*AXIS*, *EXEC*, *HOME*, *NHOME*, *MODE*, *DIRC*, *MINF*, *MAXF*, *TIME*, *DONE*, *ERR*, *ERRID)*	U	

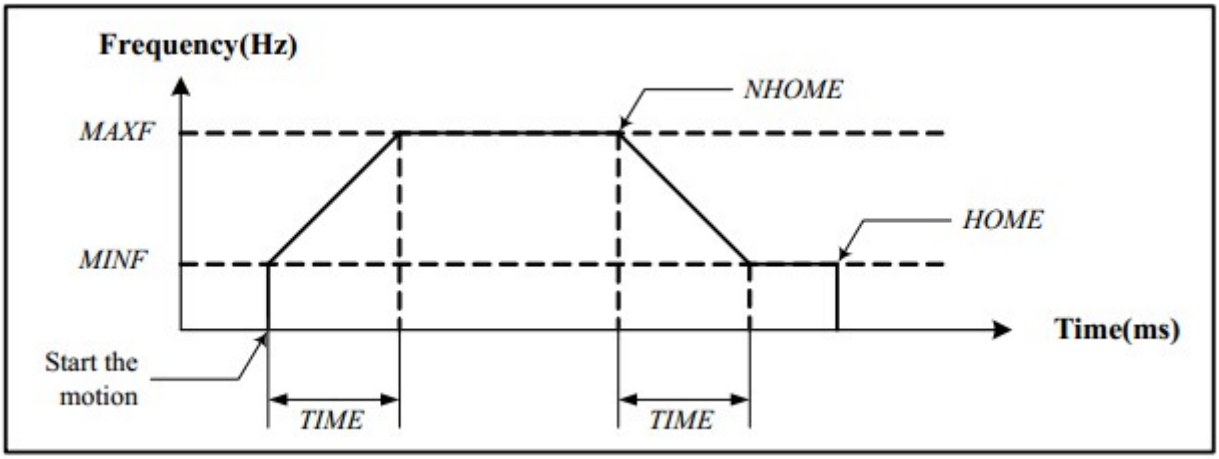
Operandus	Be/Kimenet	Adat típus	Memória terület
AXIS	Bemenet	INT	Konstans (vagy 1)
EXEC	Bemenet	BOOL	I, Q, V, M, L, SM, RS, SR
HOME	Bemenet	BOOL	I, Q, V, M, L, SM, RS, SR
NHOME	Bemenet	BOOL	I, Q, V, M, L, SM, RS, SR
MODE	Bemenet	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Konstans
DIRC	Bemenet	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Konstans
MINF	Bemenet	WORD	I, Q, M, V, L, SM, Konstans
MAXF	Bemenet	WORD	I, Q, M, V, L, SM, Konstans
TIME	Bemenet	WORD	I, Q, M, V, L, SM, Konstans
DONE	Kimenet	BOOL	Q, M, V, L, SM
ERR	Kimenet	BOOL	Q, M, V, L, SM
ERRID	Kimenet	BYTE	Q, M, V, L, SM

A következő táblázat a fenti bemenetek részletesebb leírását tartalmazza.

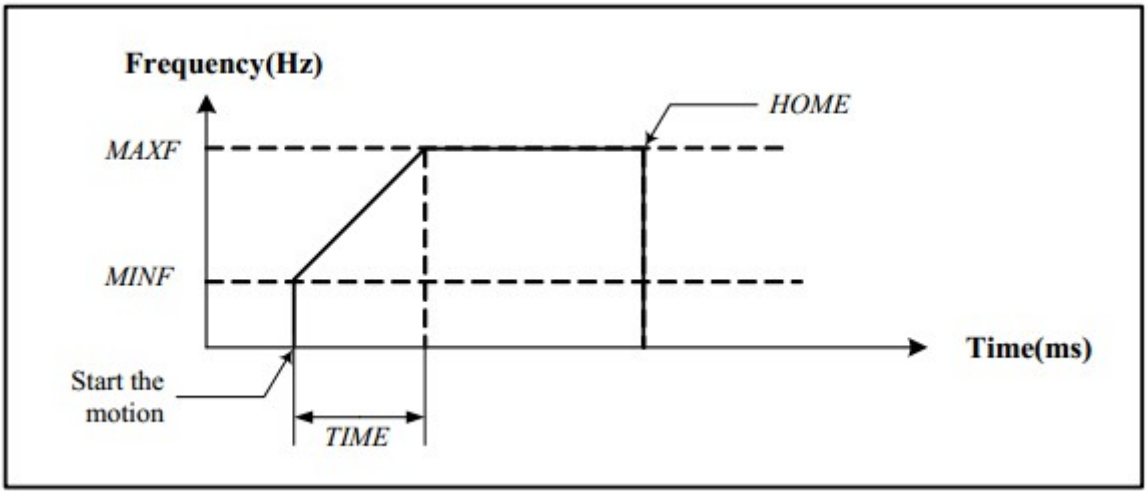
Operandus	Leírás
AXIS	A nagy sebességű kimenet kiválasztása, 0 → Q0.0 1 → Q0.1
EXEC	Ha EN bemenet 1, akkor az EXEC bemenet elindítja a kezdeti pozíció felvételét
HOME	Kezdeti pozíciót jelző érzékelő jele
NHOME	Kezdeti pozíció közelségét jelző érzékelő jele
MODE	Kezdeti mód megadása 0 → Kezdeti pozíció közeli jel, illetve kezdeti pozíciót jelző jelet is figyelembe veszi 1 → csak a kezdeti pozíciót jelző jel figyelése
DIRC	Motor forgásirányának megadása 0 → előre forgatás 1 → hátra forgatás
MINF	Kezdeti sebességhez tartozó frekvencia megadása, mértékegysége: Hz *Megjegyzés: MINF értéké kisebb vagy egyenlő lehet, mint 2KHz*
MAXF	Az impulzus kimenet legnagyobb sebességét adja meg. (Hz). MAXF tartománya: 20Hz~20KHz. MAXF-nek nagyobbnak vagy egyenlőnek kell lennie mint MINF.
TIME	Gyorsítás/lassítás idő megadása (ms). A pozíció vezérlési mód esetében a gyorsítási, lassítási idő megegyezik Gyorsítás idő az az idő, amíg felgyorsít MINF-ről MAXF-re. Lassítás idő az az idő, amíg lelassít MAXF-ről MINF-re.
DONE	Az utasítás befejeztét jelzi. 0=nincs befejezve, 1= befejezve.
ERR	Jelzi ha hiba lépett fel a végrehajtás közben. 0= nincs hiba, 1= hiba lépett fel.
ERRID	Hiba azonosítás. Ha ERR=1, ERRID leírja a hibainformációkat,

Az utasítással a tengely kezdeti pozícióba vezérelhető, felhasználva a kezdeti pozíció közelségét (NHOME) illetve a kezdeti pozíciót (HOME) jelző bemeneteket. A MODE bemenettel kiválasztható az alkalmazni kívánt mód. A kezdeti pozíció keresése közben, ha a DIRC bemenet értéke 0 (előre forgásirány), az aktuális értékek (SMD212/SMD242) növekednek, ha DIRC bemenet értéke 1 (vissza forgásirány) akkor az aktuális értékek csökkennek.

- Ha a MODE = 0 (HOME és NHOME jeleket is figyeli a blokk), a PHOME utasítás lelassítja a tengelyt, ha NHOME = 1, és megállítja, ha HOME = 1. A működést az alábbi időfüggvény szemlélteti.



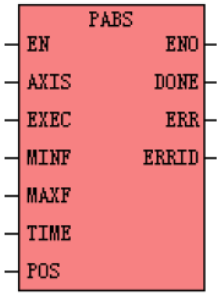
- Ha MODE=1 (csak a HOME jelet használja), PHOME utasítás megállítja a tengelyt, ha HOME = 1. A működést az alábbi időfüggvény szemlélteti.



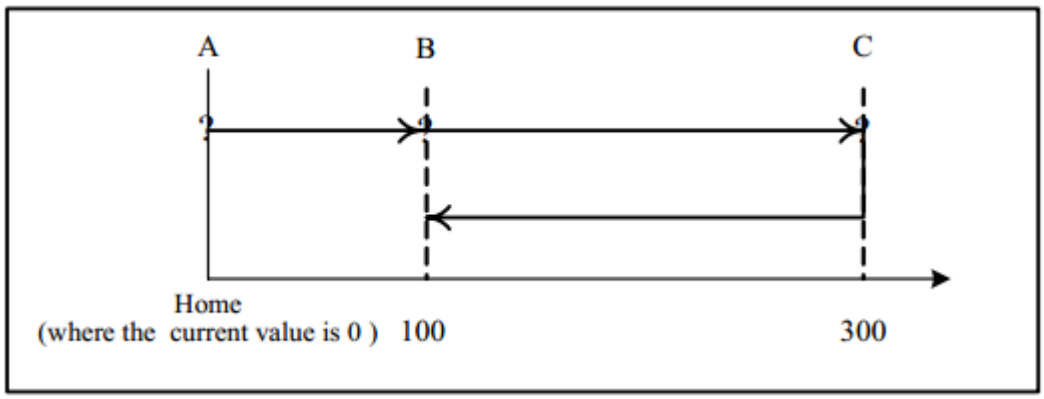
- **LD**
  Ha EN=1, az utasítás végrehajtódik

- **IL**
  Ha CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

6.16.4. PABS (abszolút pozicionálás)

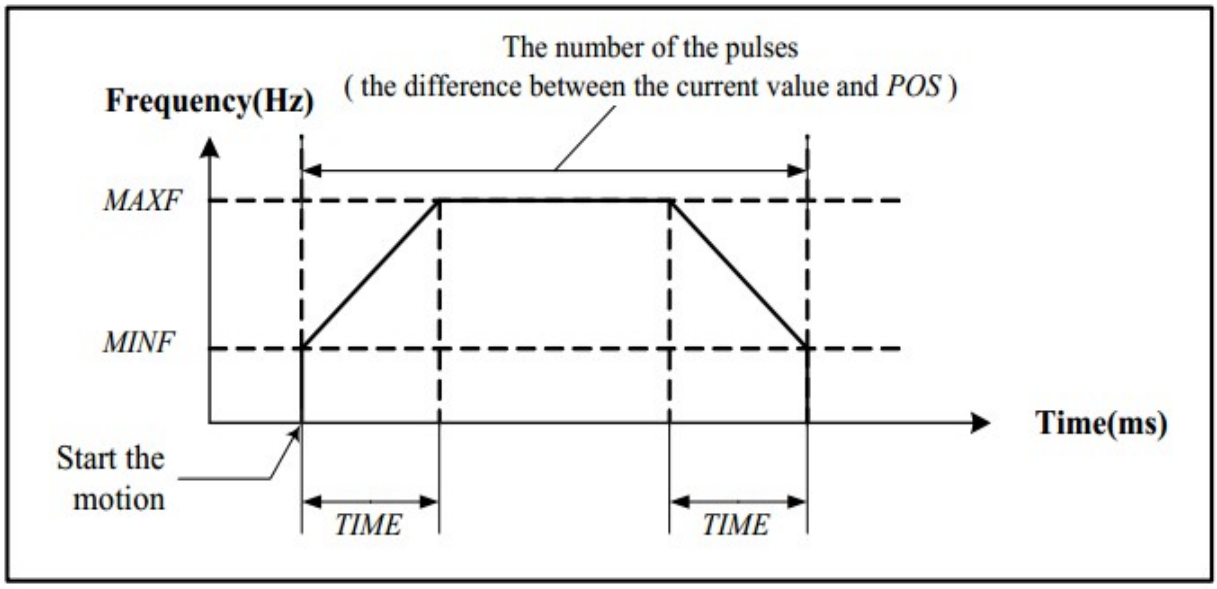
	Név	Használat	Csoport	
LD	PABS			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	PABS	PABS (*AXIS*, *EXEC*, *MINF*, *MAXF*, *TIME*, *POS*, *DONE*, *ERR*, *ERRID)*	U	

Operandus	Be/Kimenet	Adat típus	Memória terület
AXIS	Bemenet	INT	Konstans (vagy 1)
EXEC	Bemenet	BOOL	I, Q, V, M, L, SM, RS, SR
MINF	Bemenet	WORD	I, Q, M, V, L, SM, Konstans
MAXF	Bemenet	WORD	I, Q, M, V, L, SM, Konstans
TIME	Bemenet	WORD	I, Q, M, V, L, SM, Konstans
POS	Bemenet	DINT	I, Q, M, V, L, SM, HC, Konstans
DONE	Kimenet	BOOL	Q, M, V, L, SM
ERR	Kimenet	BOOL	Q, M, V, L, SM
ERRID	Kimenet	BYTE	Q, M, V, L, SM

Operandus	Leírás
AXIS	A nagy sebességű kimenet kiválasztása, 0 → Q0.0 1 → Q0.1
EXEC	Ha EN = 1, EXEC felfutó éle elindítja az abszolút pozicionálást
MINF	Kezdeti sebességhez tartozó frekvencia megadása, mértékegysége: Hz *Megjegyzés: MINF értéké kisebb vagy egyenlő lehet, mint 2KHz*
MAXF	Az impulzus kimenet legnagyobb sebességét adja meg. (Hz). MAXF tartománya: 20Hz~20KHz. MAXF-nek nagyobbnak vagy egyenlőnek kell lennie mint MINF.
TIME	Gyorsítás/lassítás idő megadása (ms). A pozíció vezérlési mód esetében a gyorsítási, lassítási idő megegyezik Gyorsítás idő az az idő, amíg felgyorsít MINF-ről MAXF-re. Lassítás idő az az idő, amíg lelassít MAXF-ről MINF-re.
POS	Cél pozíció megadása, mely az impulzusok számát jelenti a kezdeti pozíciótól, ahol az aktuális érték 0. A működés szemléltetése az alábbi ábrán látható. Ha a mozgatást A és B pont között szeretnénk elvégezni, akkor a pozíciót 100-ra kell beállítani. B és C közötti mozgatáshoz pedig 300-ra. A mozgatás C-től B-be pedig 100-ra kell állítani a cél pozíció értékét. 
DONE	Az utasítás befejeztét jelzi. 0=nincs befejezve, 1= befejezve.
ERR	Jelzi ha hiba lépett fel a végrehajtás közben. 0= nincs hiba, 1= hiba lépett fel.
ERRID	Hiba azonosítás. Ha ERR=1, ERRID leírja a hibainformációkat,

Az utasítással a tengely egy megadott abszolút pozícióba (POS bemenet) mozgatható, az impulzus kimenet mindaddig működik, amíg a cél pozíciót el nem érte a tengely. Ha az irányvezérlő bit (SM201.3 / SM231.3) értéke 0, PABS utasítás használja az irányvezérlő kimeneteket (Q0.2/Q0.3). Ha a cél érték nagyobb, mint az aktuális érték, akkor a motort előre forgatja, így az aktuális érték (SMD212/SMD242) folyamatosan növekszik. Ha a cél érték kisebb, mint az aktuális érték, akkor a motort visszafele forgatja, így az aktuális érték csökken.

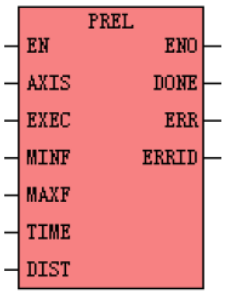
A működést az alábbi időfüggvény szemlélteti.



- **LD**
  Ha EN=1, az utasítás végrehajtódik

- **IL**
  Ha CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

6.16.5. PREL (relatív pozicionálás)

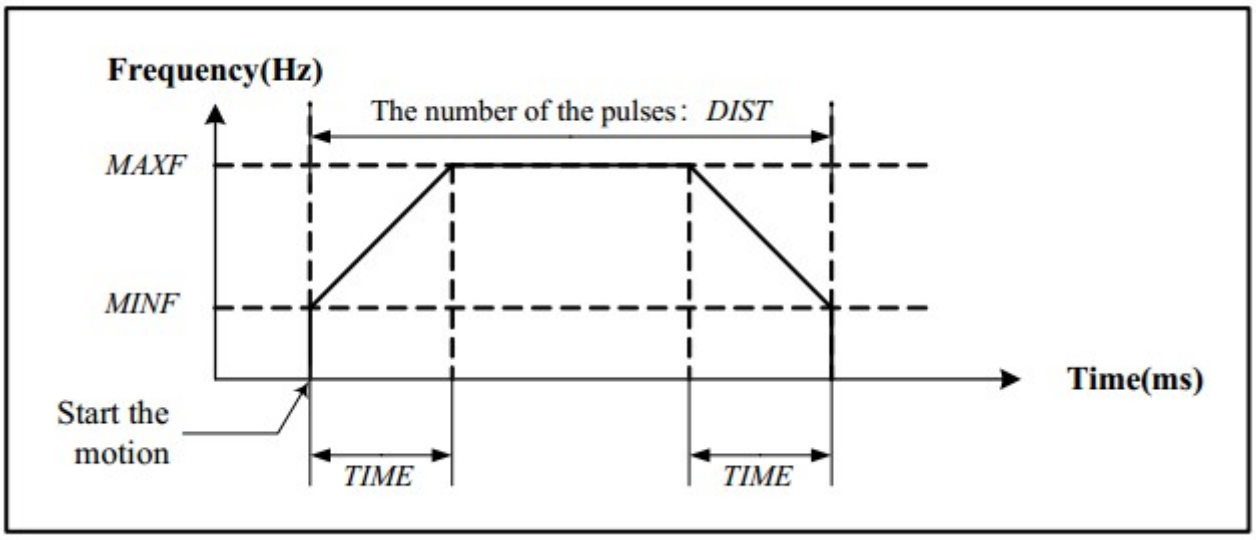
	Név	Használat	Csoport	
LD	PREL			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	PREL	PREL (AXIS, EXEC, MINF, MAXF, TIME, DIST, DONE, ERR, ERRID)	U	

Operandus	Be/Kimenet	Adat típus	Memória terület
AXIS	Bemenet	INT	Konstans (vagy 1)
EXEC	Bemenet	BOOL	I, Q, V, M, L, SM, RS, SR
MINF	Bemenet	WORD	I, Q, M, V, L, SM, Konstans
MAXF	Bemenet	WORD	I, Q, M, V, L, SM, Konstans
TIME	Bemenet	WORD	I, Q, M, V, L, SM, Konstans
POS	Bemenet	DINT	I, Q, M, V, L, SM, HC, Konstans
DONE	Kimenet	BOOL	Q, M, V, L, SM
ERR	Kimenet	BOOL	Q, M, V, L, SM
ERRID	Kimenet	BYTE	Q, M, V, L, SM

Operandus	Leírás
AXIS	A nagy sebességű kimenet kiválasztása, 0 → Q0.0 1 → Q0.1
EXEC	Ha EN = 1, EXEC felfutó éle elindítja a relatív pozicionálást
MINF	A sebességhez tartozó frekvencia megadása, mértékegysége: Hz
DIRC	Motor forgásirányának kiválasztása 0 → előre forgatás 1 → hátra forgatás
DONE	Az utasítás befejeztét jelzi. 0=nincs befejezve, 1= befejezve.
ERR	Jelzi ha hiba lépett fel a végrehajtás közben. 0= nincs hiba, 1= hiba lépett fel.
ERRID	Hiba azonosítás. Ha ERR=1, ERRID leírja a hibainformációkat,

Az utasítással a tengely egy megadott távolságra (DIST) mozgatható mozgatható a tengely. Ha az irányvezérlő bit (SM201.3/SM231.3) 0, akkor PREL utasítás használja az irányvezérlő kimeneteket (Q0.2/Q0.3). Ha a DIST értéke pozitív, akkor a motort előre forgatja, és az aktuális érték (SMD212/SMD242) növekszik. Ha a DIST negatív, a motor visszafele forog, az aktuális értét csökken.

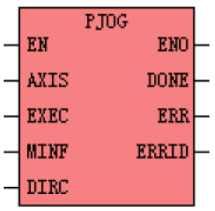
A működést az alábbi időfüggvény szemlélteti.



- **LD**
  Ha EN=1, az utasítás végrehajtódik

- **IL**
  Ha CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

6.16.6. PJOG (Léptetés)

	Név	Használat	Csoport	
LD	PJOG			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	PJOG	PJOG (*AXIS*, *EXEC*, *MINF*, *DIRC*, *DONE*, *ERR*, *ERRID)*	U	

Operandus	Be/Kimenet	Adat típus	Memória terület
AXIS	Bemenet	INT	Konstans (vagy 1)
EXEC	Bemenet	BOOL	I, Q, V, M, L, SM, RS, SR
MINF	Bemenet	WORD	I, Q, M, V, L, SM, Konstans
TIME	Bemenet	WORD	I, Q, M, V, L, SM, Konstans
DIRC	Bemenet	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Konstans
DONE	Kimenet	BOOL	Q, M, V, L, SM
ERR	Kimenet	BOOL	Q, M, V, L, SM
ERRID	Kimenet	BYTE	Q, M, V, L, SM

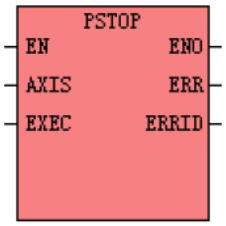
Operandus	Leírás
AXIS	A nagy sebességű kimenet kiválasztása, 0 → Q0.0 1 → Q0.1
EXEC	Ha EN = 1, EXEC felfutó éle elindítja a léptetést.
MINF	Kezdeti sebességhez tartozó frekvencia megadása, mértékegysége: Hz *Megjegyzés: MINF értéké kisebb vagy egyenlő lehet, mint 2KHz*
MAXF	Az impulzussorozat kimenet legnagyobb sebességet részletezi (Hz). MAXF tartománya: 20Hz~20KHz. MAXF-nek nagyobbnak vagy egyenlőnek kell lennie mint MINF.
TIME	Gyorsítás/lassítás idő részletezése (ms). Gyorsítás és lassítás ideje megegyezik a pozíció vezérlés utasításban. Gyorsítás idő az az idő amilyen gyorsan felgyorsít MINF-ről MAXF-re. Lassítás idő az az idő amilyen gyorsan lelassít MAXF-ről MINF-re.
DIRC	Elektromotor forgatás vezérlés részletezése: 0=előre forgatás; 1=hátra forgatás.
DONE	Az utasítás befejeztét jelzi. 0=nincs befejezve, 1= befejezve.
ERR	Jelzi ha hiba lépett fel a végrehajtás közben. 0= nincs hiba, 1= hiba lépett fel.
ERRID	Hiba azonosítás. Ha ERR=1, ERRID leírja a hibainformációkat,

Az utasítással a tengely léptetése valósítható meg, mely esetén a gyors kimenet impulzus kimenetként működik, melynek frekvenciáját a MINF paraméter adja meg. Ha az irányvezérlő bit (SM201.3/SM231.3) 0, akkor PJOG utasítás használja az irányvezérlő kimeneteket (Q0.2/Q0.3). Ha DIRC bemenet 0, akkor az aktuális érték (SMD212/SMD242) növekszik, ha pedig 1, akkor csökken.

- **LD**
  Ha EN=1, az utasítás végrehajtódik

- **IL**
  Ha CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

6.16.7. PSTOP (megállítás)

	Név	Használat	Csoport	
LD	PSTOP			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	PSTOP	PSTOP (AXIS, EXEC, ERR, ERRID)	U	

Operandus	Be/Kimenet	Adat típus	Memória terület
AXIS	Bemenet	INT	Konstans (vagy 1)
EXEC	Bemenet	BOOL	I, Q, V, M, L, SM, RS, SR
ERR	Kimenet	BOOL	Q, M, V, L, SM
ERRID	Kimenet	BYTE	Q, M, V, L, SM

Operandus	Leírás
AXIS	A nagy sebességű kimenet kiválasztása, 0 → Q0.0 1 → Q0.1
EXEC	Ha EN = 1, EXEC felfutó éle megállítja a mozgást
ERR	Jelzi ha hiba lépett fel a végrehajtás közben. 0= nincs hiba, 1= hiba lépett fel.
ERRID	Hiba azonosítás. Ha ERR=1, ERRID leírja a hibainformációkat,

Az utasítás a kiválasztott tengely mozgását megállítja, és a vészleállítást jelző bitet (SM201.7/SM231.7) beállítja 1-be. Mindaddig nem lehet a tengelyt mozgatni, amíg a vészleállítást jelző bitet nem törli az alkalmazó program.

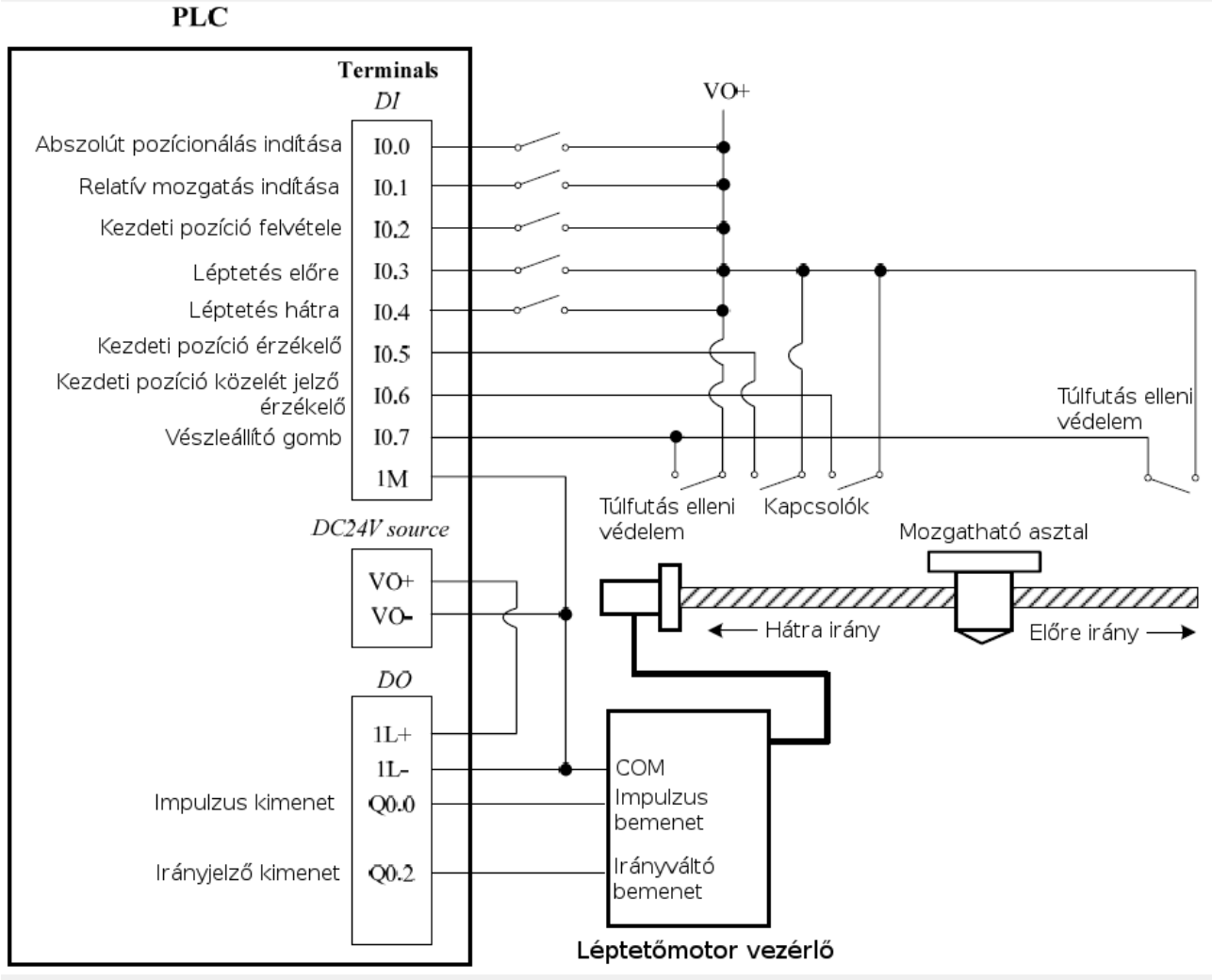
- **LD**
  Ha EN=1, az utasítás végrehajtódik

- **IL**
  HA CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

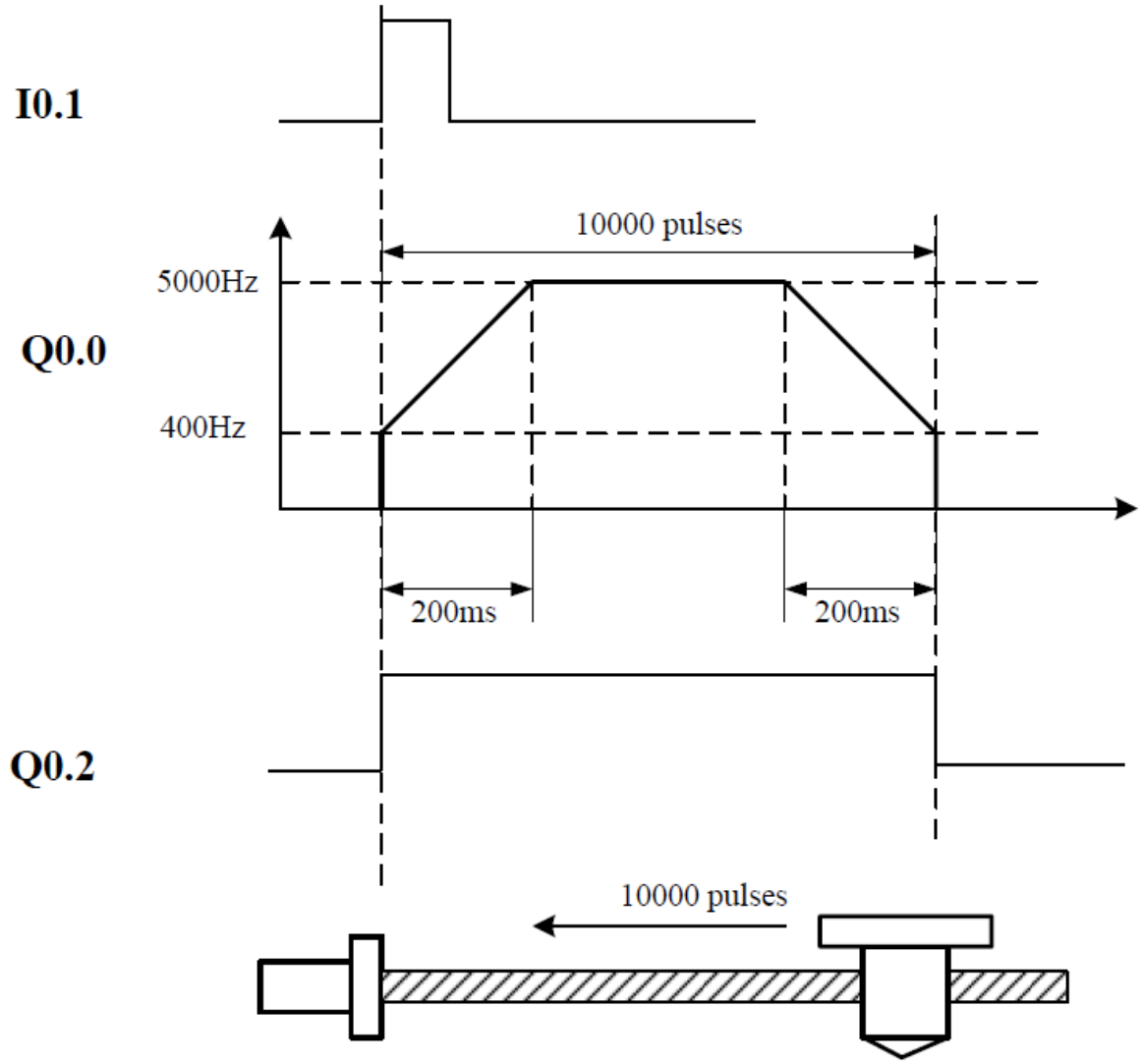
6.16.8. Példa

Egy lehetséges kialakítás pozíció vezérléshez

A következő ábrán bemutatott konfiguráción keresztül a következő funkcióblokkok kerülnek bemutatásra: PREL, PABS, PHOME, PJOG és PSTOP.

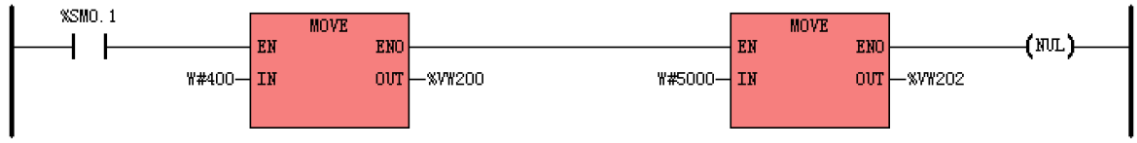


Példa relatív pozicionálásra

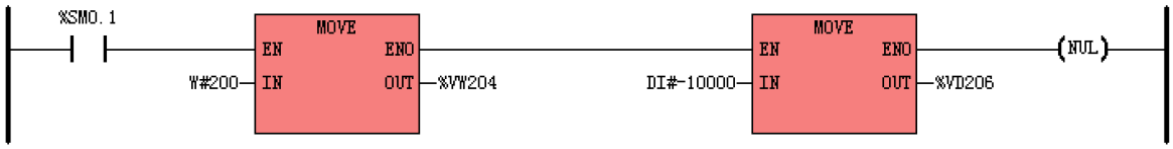


LD

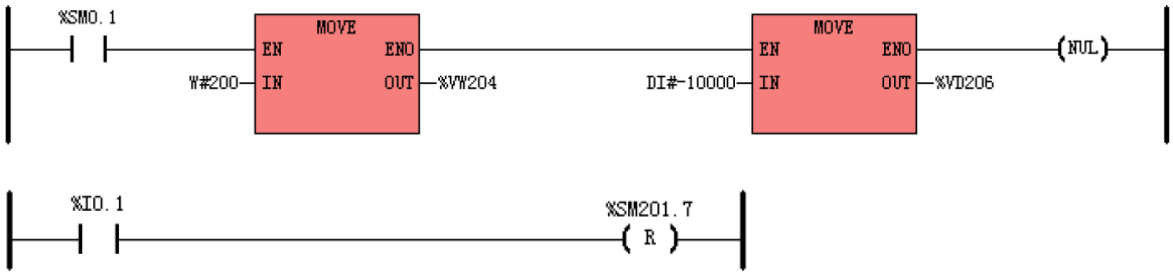
(* Network 0*)
(* Kezdeti frekvencia és a maximális frekvencia megadása *)



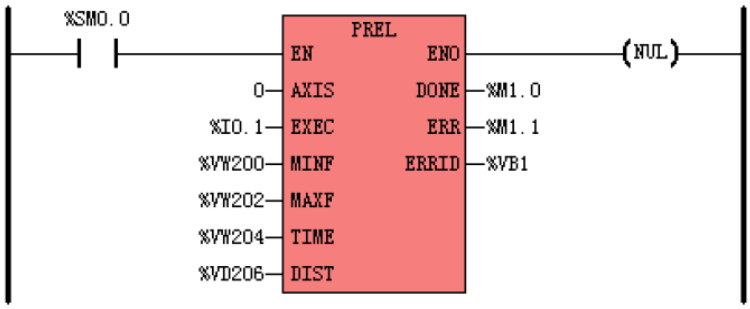
(* Network 1*)
(* Gyorsítási/lassítási idő és a távolság megadása *)



(* Network 2*)
(* Vészleállítást jelző bit törlése *)

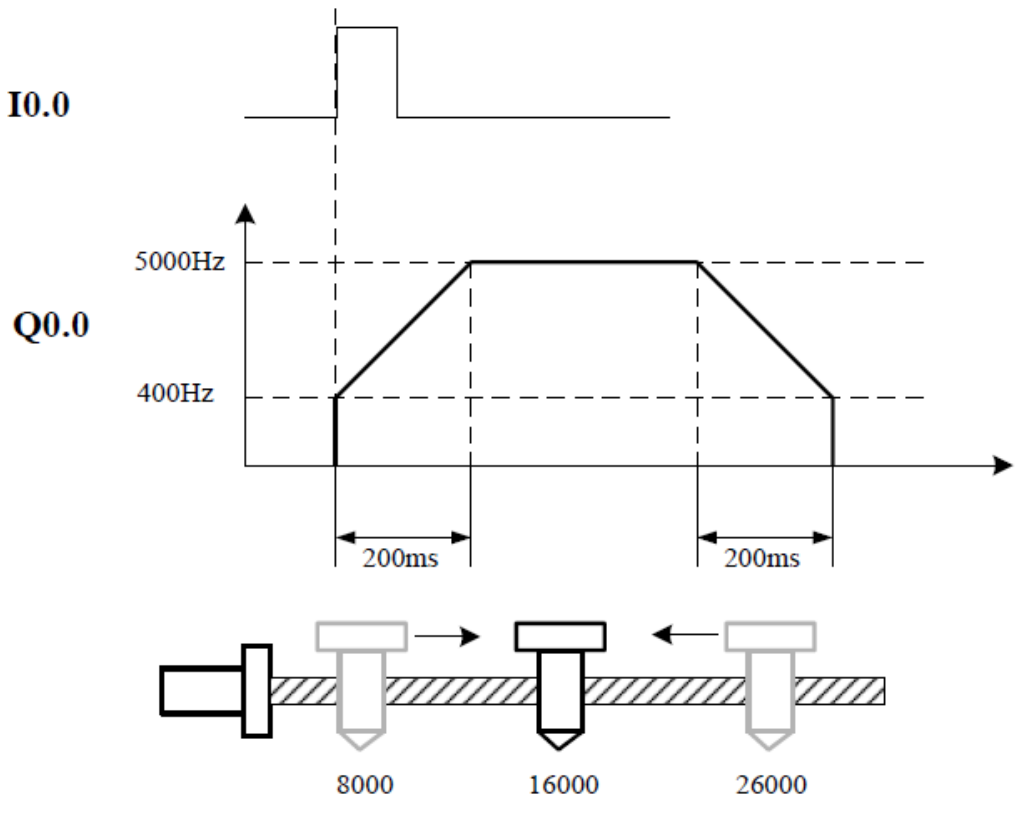


(* Network 3*)
(* PREL utasítás végrehajtása *)



IL	(* Network 0*)
	(* Kezdeti frekvencia és a maximális frekvencia megadása *)
	LD %SM0.1
	MOVE W#400, %VW200
	MOVE W#5000, %VW202
	
	(* Network 1*)
	(* Gyorsítási/lassítási idő és a távolság megadása *)
	LD %SM0.1
	MOVE W#200, %VW204
	MOVE DI#-10000, %VD206
	
	(* Network 2*)
	(* Vészleállítást jelző bit törlése *)
	LD %I0.1
	R %SM201.7
	
	(* Network 3*)
	(* PREL utasítás végrehajtása *)
	LD %SM0.0
	PREL 0, %I0.1, %VW200, %VW202, %VW204, %VD206, %M1.0, %M1.1, %VB1

Példa abszolút pozicionálásra

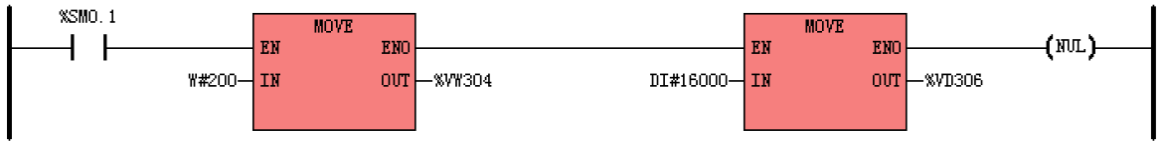


The table is required to move to the position ‘16000’.

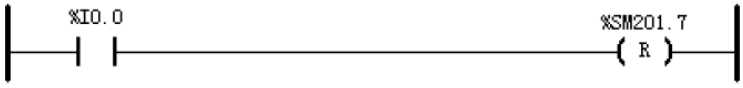
LD

(* Network 0*)
(* Kezdeti frekvencia és a maximális frekvencia megadása *)

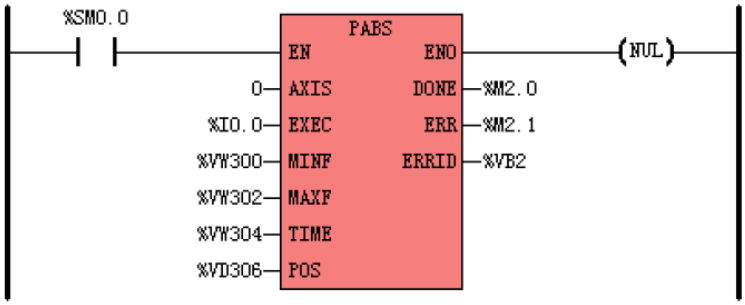
(* Network 1*)
(* Gyorsítási/lassítási idő és a cél pozíció megadása *)



(* Network 2*)
(* Vészleállítást jelző bit törlése *)

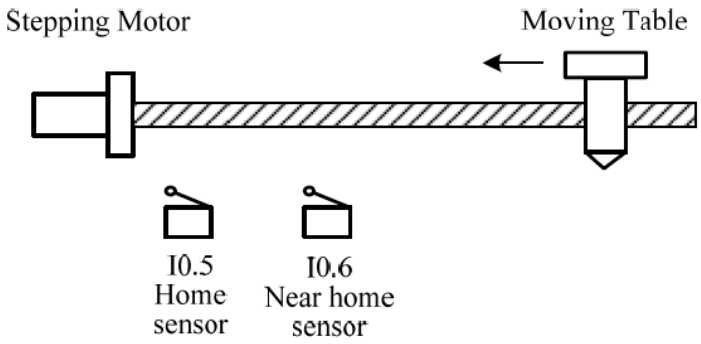


(* Network 3*)
(* PABS utasítás végrehajtása *)

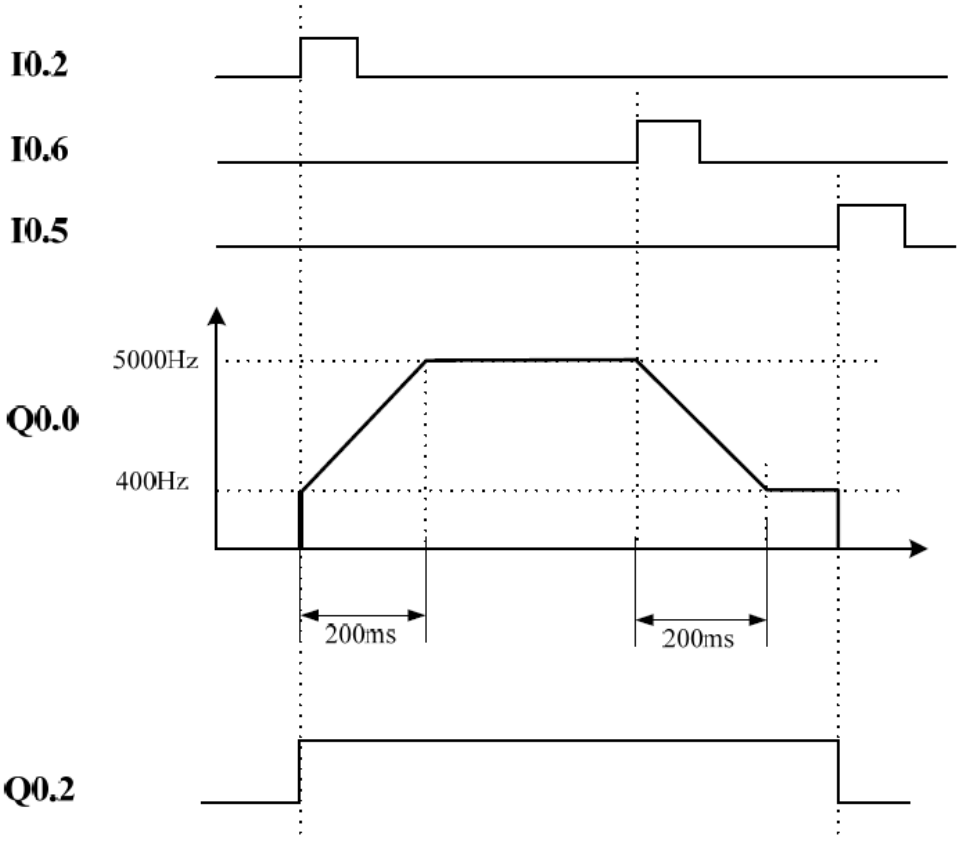


IL	(* Network 0*)
	(* Kezdeti frekvencia és a maximális frekvencia megadása *)
	LD %SM0.1
	MOVE W#400, %VW300
	MOVE W#5000, %VW302
	
	(* Network 1*)
	(* Gyorsítási/lassítási idő és a cél pozíció megadása *)
	LD %SM0.1
	MOVE W#200, %VW304
	MOVE DI#16000, %VD306
	
	(* Network 2*)
	(* Vészleállítást jelző bit törlése *)
	LD %I0.0
	R %SM201.7
	
	(* Network 3*)
	(* PABS utasítás végrehajtása *)
	LD %SM0.0
	PABS 0, %I0.0, %VW300, %VW302, %VW304, %VD306, %M2.0, %M2.1, %VB2

Példa kezdeti pozíció felvételére



I0.2 bemenettel indítható a kezdeti pozíció felvétele.
A mozgás közben Q0.2 irányváltó bemenet értéke egy, mert a kezdeti pozíció felvételéhez visszafele kell az asztalnak mozogni.



IL	(* Network 0*)
	(* Kezdeti pozíció és a kezdeti pozíció közelségét jelző érzékelők használata, hátrafelé*)
	(*forgásirány megadása*)
	LD %SM0.1
	MOVE 0, %VW396
	MOVE 1, %VW398
	
	(* Network 1*)
	(* Kezdeti és maximális frekvenciák, gyorsítási/lassítási idők megadása*)

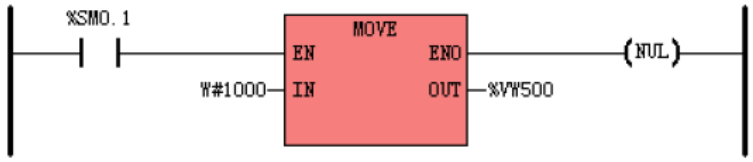
```
LD %SM0.1
MOVE W#400, %VW400
MOVE W#5000, %VW402
MOVE W#200, %VW404

(* Network 2*)
(* Vészleállítást jelző bit törlése *)
LD %I0.2
R %SM201.7

(* Network 3*)
LD %SM0.0
PHOME 0, %I0.2, %I0.5, %I0.6, %VW396, %VW398, %VW400, %VW402, %VW404,
%M3.0, %M3.1, %VB3
```

Példa léptetésre

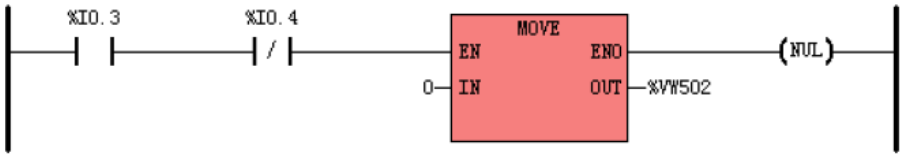
I0.3 bemenettel indítható az előre léptetés, I0.4 bemenettel pedig a hátraléptetés.
Ha I0.3 és I0.4 egy időben 1, akkor az előző irány kerül alkalmazásra.

LD

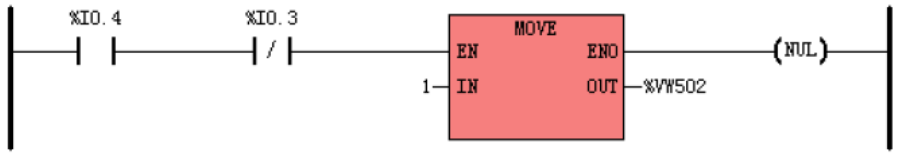
(* Network 0*)
(* PTO kimenet frekvenciájának megadása*)



(* Network 1*)
(* Irány megadása*)

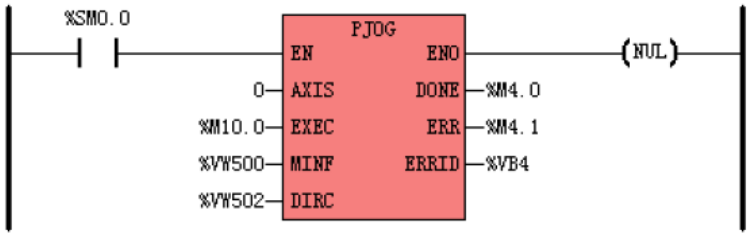


(* Network 2*)



(* Network 3*)

(* Network 4*)



IL

(* Network 0*)
(* PTO kimenet frekvenciájának megadása*)
LD %SM0.1

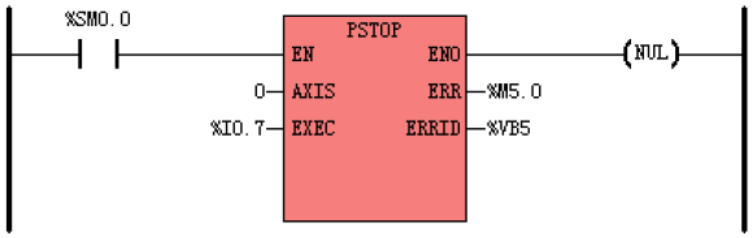
```
MOVE W#1000, %VW500

(* Network 1*)
(* Irány megadása*)
LD %I0.3
ANDN %I0.4
MOVE 0, %VW502

(* Network 2*)
LD %I0.4
ANDN %I0.3
MOVE 1, %VW502

(* Network 3*)
LD %I0.3
OR %I0.4
ST %M10.0
R %SM201.7

(* Network 4*)
LD %SM0.0
PJOG 0, %M10.0, %VW500, %VW502, %M4.0, %M4.1, %VB4
```

Mozgás megállítása

A fenti összeállításon látható két túlfutás elleni kapcsoló, melyek párhuzamosan csatlakoznak az I0.7 bemenetre, melyekkel üzemzavar esetén megállítható az éppen aktuális mozgás.

LD

(* Network 0*)

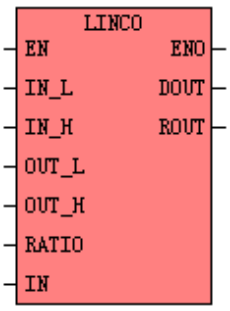


IL

```
(* Network 0*)

LD %SM0.0
PSTOP 0, %I0.7, %M5.0, %VB5
```

6.17. További utasítások

6.17.1. LINCO (lineáris átszámítás)

	Név	Használat	Csoport	
LD	LINCO			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	LINCO	LINCO (*IN_L, IN_H, OUT_L, OUT_H, RATIO, IN, DOUT, ROUT)*	U	

Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN_L	Bemenet	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Konstans
IN_H	Bemenet	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Konstans
OUT_L	Bemenet	REAL	V, L, Konstans
OUT_H	Bemenet	REAL	V, L, Konstans
RATIO	Bemenet	REAL	Konstans
IN	Bemenet	INT	I, Q, V, M, L, SM, T, C, AI, AQ
DOUT	Kimenet	DINT	Q, M, V, L, SM
ROUT	Bemenet	REAL	V, L

Megjegyzés: IN_L, IN_H, OUT_L és OUT_H mindegyike konstans vagy mindegyike változó kell, hogy legyen.
Az utasítás IN bemenet értékét felhasználva számítja az értéket, majd az eredmény megszorozza a RATIO bemenet értékével, és az eredmény ROUT-ra kerül. A DOUT kimeneten pedig az ROUT csonkolt (tizedes jegyek nélküli) értéke jelenik meg.

A LINCO utasítás funkcióját a következő egyenlet írja le:

$$ROUT = RATIO * (k*IN + b)$$

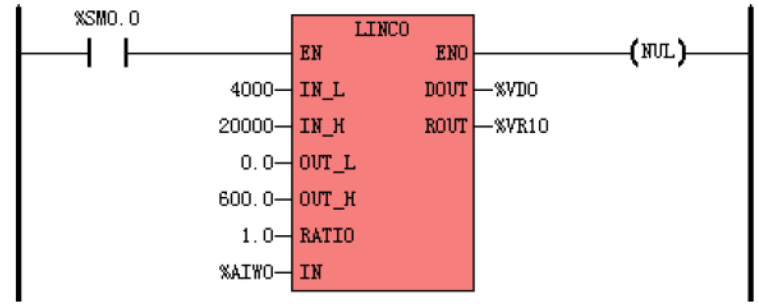
$$DOUT = \text{TRUNC}(ROUT)$$

$$k = \frac{OUT\_H - OUT\_L}{IN\_H - IN\_L}, \quad b = OUT\_L - k \times IN\_L.$$

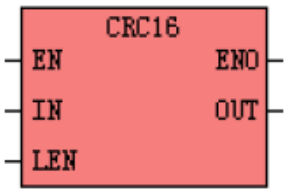
- **LD**
  Ha EN=1, az utasítás végrehajtódik

- **IL**
  Ha CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

Egy alkalmazási példa:
Tegyük fel, hogy a hőmérsékleti jelátalakító mérési tartománya 0~600°C, és a kimeneti jel pedig 4~20mA, mely az AIW0 regiszterhez kapcsolódik.

LD	
IL	LD %SM0.0 LINCO 4000, 20000, 0.0, 600.0, 1.0, %AIW0, %VD0, %VR10

6.17.2. CRC16

	Név	Használat	Csoport	
LD	CRC 16			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	CRC 16	CRC16 *IN*, *OUT, LEN*	U	

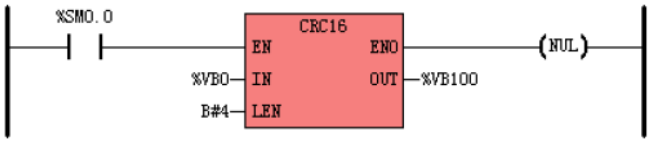
Operandus	Bemenet/Kimenet	Adat típus	Memória terület
IN	Bemenet	BYTE	I, Q, M, V, L, SM
LEN	Bemenet	BYTE	I, Q, M, V, L, SM, Konstans
OUT	Kimenet	BYTE	Q, M, V, L, SM

Az utasítás 16 bites CRC értéket számolja, a LEN változóban megadott változók felhasználásával, melyek az IN-től kezdődnek. Az eredmény 2 byte-ba kerül, melyek kezdetét az OUT adja meg.

- **LD**
  Ha EN=1, az utasítás végrehajtódik

- **IL**
  Ha CR=1, az utasítás végrehajtódik, és nincs hatással CR-re.

LD



Mivel SM0.0 értéke mindig 1, ezért CRC16 utasítás végrehajtásra kerül. VB0-tól kezdődően 4 byte felhasználásával számol, az eredmény magasabb helyiértékű byte-ja a VB100-ra kerül, az alacsonyabb pedig VB101-re.

IL

```
LD %SM0.0
CRC16 %VB0, %VB100, B#4
```

Eredmény

The data to be checked				16-bit CRC	
VB0	VB1	VB2	VB3	VB100	VB101
B#16#1A	B#16#2B	B#16#3C	B#16#4D	B#16#A6	B#16#1

MELLÉKLET „A”

Modbus RTU kommunikáció használata

A KINCO-K3 PLC-k minden esetben képesek Modbus RTU SLAVE-ként kommunikálni.

1. Használható funkciókódok

Típus	**Használható funkciókód**	**A PLC megfelelő memória területe**
DO (Digitális kimenet, 0XXXX)	01,05,15	Q, M
DI (Digitális bemenet, 1XXXX)	2	I, M
AO (Analóg kimenet, 4XXXX)	03,06,16	AQ, V
AI (Analóg bemenet, 3XXXX)	4	AI, V

2. Központi egységek esetén használható memória területek

CPU304

Terület	**Tartomány**	**Típus**	**Megfelelő Modbus regiszter**
I	I0.0---I0.7	DI	0---7
Q	Q0.0---Q0.5	DO	0---5
M	M0.0---M31.7	DI/DO	64---319
AI	---	AI	---
AQ	---	AO	---
V	VW0---VW2046	AI/AO	16---1039

CPU304EX és CPU306

Terület	**Tartomány**	**Típus**	**Megfelelő Modbus regiszter**
I	I0.0---I0.7	DI	0---63
Q	Q0.0---Q0.5	DO	0---63
M	M0.0---M31.7	DI/DO	64---319
AI	AIW0---AIW30	AI	0---15
AQ	AQW0---AQW30	AO	0---15
V	VW0---VW2046	AI/AO	16---2063

CPU306EX és CPU308

Terület	**Tartomány**	**Típus**	**Megfelelő Modbus regiszter**
I	I0.0---I31.7	DI	0---255
Q	Q0.0---Q31.7	DO	0---255
M	M0.0---M31.7	DI/DO	320---575
AI	AIW0---AQW62	AI	0---31
AQ	AQW0---AQW62	AO	0---31
V	VW0---VW4094	AI/AO	100---2147

MELLÉKLET „B”

SM regiszterek funkciói

Minden egyes programciklus végén, a PLC frissíti az SM (System Memory) memóriaterületet. Néhány SM regiszter csak olvasható, melyek információt adnak a rendszer állapotáról, néhány regiszter pedig írható, rendszerfunkciók módosításához.

1. **SMB0**

SMB0-t (SM0.0---SM0.7) a CPU minden ciklus végén frissíti. A bitek értéke csak olvasható.

SM bit	**Leírás**
SM0.0	Mindig bekapcsolva
SM0.1	Csak a PLC első ciklusakor 1, kezdeti értékek felvételére használható
SM0.2	Ha a RAM tartalma elveszett, akkor be van kapcsolva az első ciklus alatt, egyéb esetben pedig ki van kapcsolva.
SM0.3	Impulzus kimenet (50% kitöltéssel), 1s periódusidővel.
SM0.4	Impulzus kimenet (50% kitöltéssel), 2s periódusidővel.
SM0.5	Impulzus kimenet (50% kitöltéssel), 4s periódusidővel.
SM0.6	Impulzus kimenet (50% kitöltéssel), 60s periódusidővel.
SM0.7	Fenntartva

2. **SMW22 és SMW24**

SMW22 tárolja a 0 időzítő megszakítás (3-as esemény) ciklusidejét, értéke 1 ~ 65535 között lehet, mértékegysége ms, ha értéke 0 (alapértelmezett érték), akkor a megszakítás le van tiltva.

SMW24 tárolja az 1-es időzítő megszakítás (4-es esemény) ciklusidejét, értéke 1 ~ 65535 között lehet, mértékegysége ms, ha értéke 0 (alapértelmezett érték), akkor a megszakítás le van tiltva.

3. **SMW26 és SMW28**

Az SMW26 és SMW28 regiszterek tárolják a központi egységen található két potenciométer értékét. SMW26 az egyes számú potenciométer, az SM28 pedig a nullás potenciométer értékét tartalmazza.
A központi egység automatikusan frissíti értéküket, csak olvasható regiszterek.

4. **SMB31 és SMB32**

CPU304, CPU304EX és CPU 306 modellek esetén adatok permanens tárolásának vezérlésére használható változók. A permanens változók az értéküket megtartják, lekapcsolt központi egység esetén is.

MELLÉKLET „C”

Permanens adatmentés

A központi egységben található V memória terület egy megadott része használható adatok permanens tárolására. Ebben az esetben az adatok az FRAM területre kerülnek.

1. Permanens memóriaterület

A következő táblázat tartalmazza V memória megadott területét, melyről az adatok az FRAM területre menthetők.

	CPU304	**CPU304EX, CPU306, CPU306EX, CPU 308**
Mérete	128 bájt	255 bájt
Tartomány	VB1648~VB1775	VB3648~VB3902

2. Adatmentés módja

2.1. CPU306EX, CPU308 típusok esetében

A CPU306EX és CPU308 központi egységek esetében az adatok átmásolhatók a permanens memóriaterületről, ahonnan automatikusan átkerülnek az FRAM területre, nincs szükség vezérlőregiszterek használatára.

```
(*NETWORK 0*)
LD %SM0.0
MOVE %AIW0, %VW3648 (* AIW0 értékének permanens tárolása*)
SPD 1, W#1000, %VD4000  (* HSC1 impulzus frekvenciájának számítása *)
                        (* a frekvencia permanens tárolása *)
```

2.2. CPU304, CPU304EX, CPU306 típusok esetében

A fenti központi egységek esetében permanens adatmentéshez a következőképpen kell eljárni.

1. A tárolni kívánt adatokat mentsük a permanens memória területre
2. SMB31 és SMB32 regiszterek alkalmazásával kell az adatokat az FRAM területre áthelyezni

2.2.1. SM31.0, SM31.1 és SM31.7

SM31.1	SM31.0	Leírás
0	0	Egy byte (8 bit) értékének mentése
0	1	Egy byte (8 bit) értékének mentése
1	0	Egy Word (16 bit) értékének mentése
1	1	Egy DWord (32 bit) értékének mentése

SM31.7	Leírás
0	FRAM-ba mentés engedélyezése
1	FRAM-ba mentés tiltása

2.2.2. SMW32

A V terület címét tartalmazza, ahonnan az adat kimentésre kerül. Az érték a VB0-tól való eltolást jelenti.

2.2.3. Írás a FRAM-ba

A FRAM-ba írás parancsa: MOVE *offset*, %SMW32
Az ofszet egy INT érték, mely a VB0-tól való eltolást adja meg. Például, ha VB3600 értékét kívánjuk az FRAM területre menteni, akkor az ofszet értékét 3600-ra kell beállítani.
Megjegyzés: A programciklus végén történik a tényleges FRAM területre történő mentés.

```
(* NETWORK 0 *)
(* VB3649, VW3650, VD3652 értékének mentése FRAM-ba, M0.0 vezérlésével*)

LDN %M0.0           (* Ha M0.0 értéke 0 *)
MOVE B#0, SMB31 (* FRAM-ba írás tiltása *)

(* NETWORK 1 *)
LD %M0.0                        (* Ha M0.0 értéke 1 *)
MOVE B#2#10000001, SMB31        (* 1 byte mentése *)
MOVE 3649, %SMW32               (* VB3649 mentése FRAM-ba*)
MOVE B#2#10000010, SMB31        (* 1 word mentése (2 byte) *)
MOVE 3650, %SMW32               (* VW3650 mentése FRAM-ba *)
MOVE B#2#10000011, SMB31        (* 1 dword mentése (4 byte) *)
MOVE 3652, %SMW32               (* VD3652 mentése FRAM-ba *)
```