

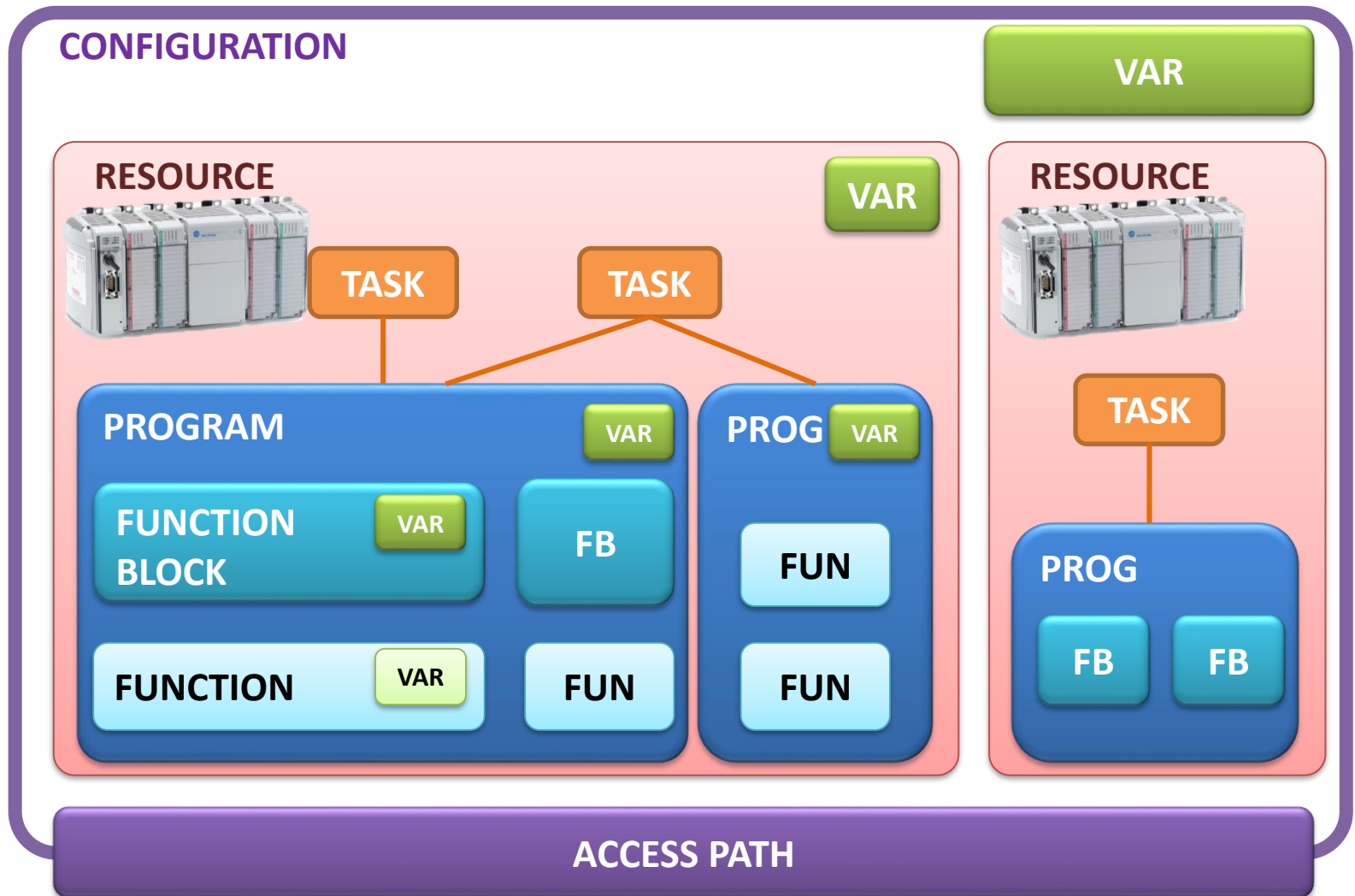
Az IEC 61131-3 szabvány programozási nyelvei

Utasításlista és Funkcióblokk-diagram

Programozható irányítóberendezések
és szenzorrendszerek

KOVÁCS Gábor
gkovacs@iit.bme.hu

Áttekintés



Programszervezési egységek

POU típus és név

Deklarációs rész:

- Interfész változók
- Lokális változók
- Globális változók

POU törzs: programkód

- Ladder Diagram (LD)
- Instruction List (IL)
- Function Block Diagram (FBD)
- Structured Text (ST)
- Sequential Function Chart (SFC)

PROGRAM prog_name

PROGRAM ConveyorControl

FUNCTION_BLOCK fb_name

FUNCTION_BLOCK Pusher

FUNCTION fun_name : DataType

FUNCTION IsReady : BOOL

Utasításlista (Instruction list, IL)

- Szöveges programozási nyelv
- Alacsony szintű, gépi kódhoz közeli
 - Alapszintű műveletek
 - A programvezérlési lehetőségek korlátozottak
 - A program futása teljes egészében kézben tartható
- Bármely más IEC 61131-3 kompatibilis nyelven írt program leírható IL-lel



A teljes szöveges kód végrehajtódik ciklusonként!

Utasítások felépítése

Operátor**Módosító** **Operandus**

LDN **%I0.1**

S **MyBool**

NOT

- **Operátor (operator)**
 - Az operátor mnemonikja
- **Módosító (modifier)**
 - **N**: operandus negálása
 - **C**: feltételes végrehajtás
 - **(**: egymásba ágyazás
 - A módosítók használata az operátortól függ
- **Operandus (operand)**
 - Egy operandus vagy egy sem
 - Literális vagy változó (bemenet, kimenet, memória, közvetlen)
 - Változó adattípusú

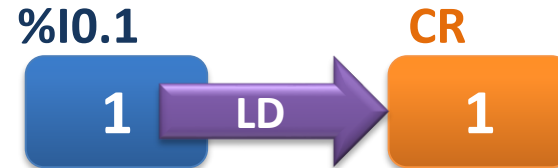
Az akkumulátor

CR – Current Result

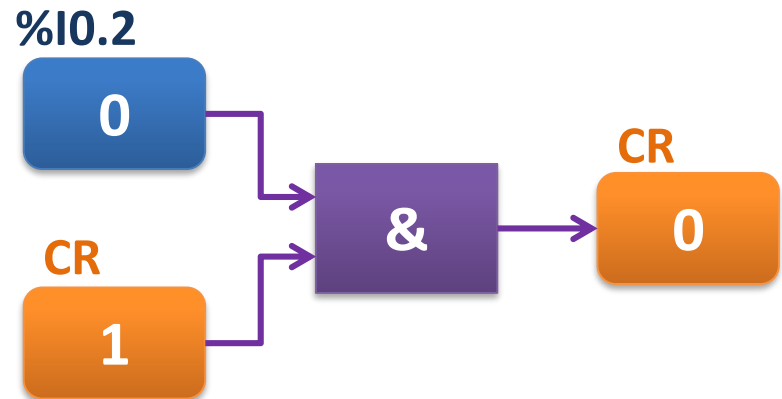
- Általános regiszter
- A műveletek első operandusa mindig az akkumulátor (kivétel: érték betöltése)
- A művelet eredménye az akkumulátorba kerül
- Az akkumulátor adattípusa a művelettől és az operandusok adattípusától függ

Az akkumulátor

LD %I0.1



AND %I0.2



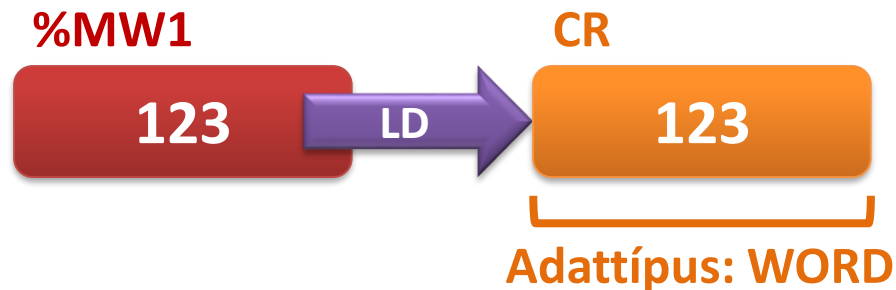
ST %Q0.1



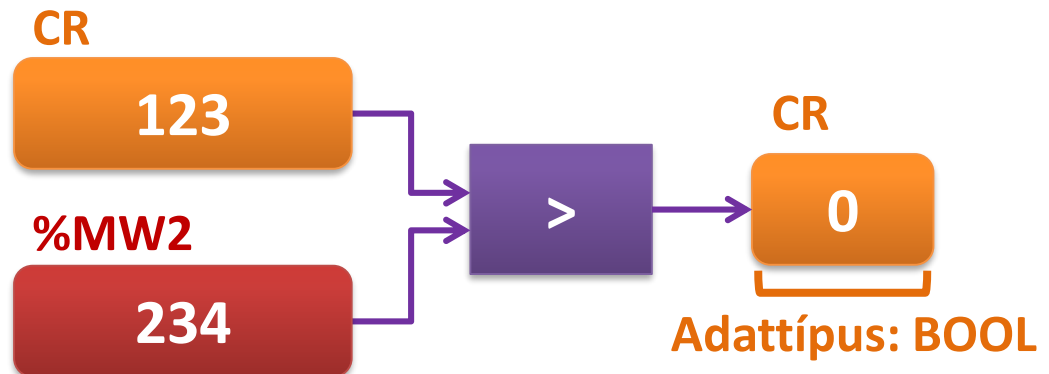
Az akkumulátor

Az akkumulátor adattípusa a **művelettől** és az **operandustól** függően változik





LD %MW1



GT %MW2



Bitműveletek

Utasítás	Leírás	Létra-szimbólum	Példa
LD	Bit betöltése az akkumulátorba		LD %I0.1
LDN	Bit negálása és betöltése az akkumulátorba		LDN %M11
ST	Akkumulátor-érték tárolása egy biten		ST %M21
STN	Akkumulátor-érték negáltjának tárolása egy biten		STN %Q0.2

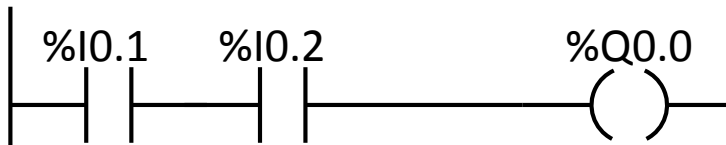
Retentív hozzárendelések

Utasítás	Leírás	Létra-szimbólum	Példa
S	Bit 1-be állítása ha az akkumulátor-érték 1	$\text{—}(\text{S})\text{—}$	S %Q0.1
R	Bit 0-ba állítása ha az akkumulátor-érték 1	$\text{—}(\text{R})\text{—}$	R %M12

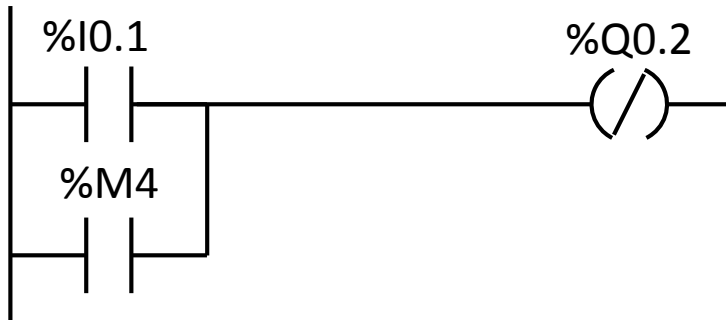
Bit-logikai műveletek

Utasítás	Leírás	Példa
AND / ANDN	AND / NAND művelet az akkumulátor-értéken és az operanduson, az eredmény az akkumulátorba kerül	AND %I0.1
OR / ORN	OR / NOR művelet az akkumulátor-értéken és az operanduson, az eredmény az akkumulátorba kerül	ORN %M11
XOR / XORN	XOR / XORN (ekvivalencia) művelet az akkumulátor-értéken és az operanduson, az eredmény az akkumulátorba kerül	XOR %M21
NOT	Az akkumulátor-érték negálása	NOT

Bit-logikai műveletek



```
LD  %I0.1  
AND %I0.2  
ST  %Q0.0
```



```
LD  %I0.1  
OR  %M4  
STN %Q0.2
```

Aritmetikai műveletek

Utasítás	Leírás	Példa
ADD	Operandus hozzáadása az akkumulátor-értékhez	ADD %MW2
SUB	Operandus kivonása az akkumulátor-értékből	SUB %MW11
MUL	Akkumulátor-érték szorzása	MUL 3
DIV	Akkumulátor-érték elosztása az operandussal	DIV 2
MOD	Maradékképzés (modulo) az akkumulátor-értéken az operandussal	MOD 7

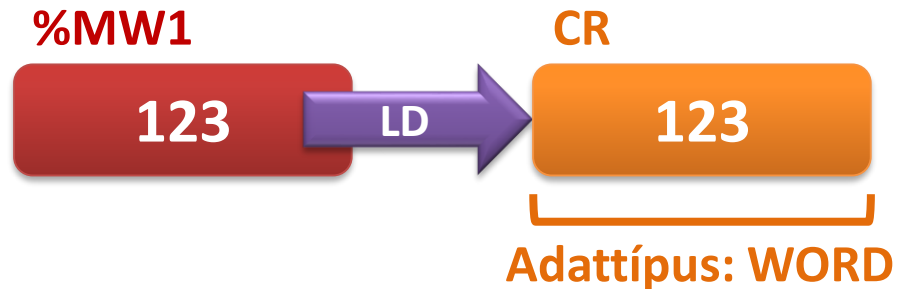
Összehasonlító operátorok

- Az akkumulátor és az operandus értékét hasonlítják össze: CR ?? OP
- Az eredmény egy logikai érték, ami az akkumulátorba kerül

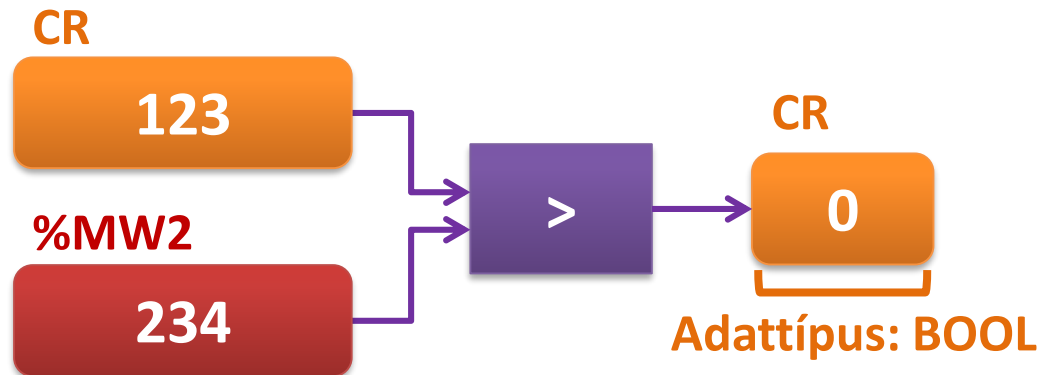
Mnemonic	Eredmény TRUE, ha
EQ	Az operandus és az akkumulátor-érték egyenlő (=)
NE	Az operandus és az akkumulátor-érték nem egyezik meg (<>)
GT	Az akkumulátor-érték nagyobb mint az operandus
GE	Az akkumulátor-érték nagyobb vagy egyenlő mint az operandus
LT	Az akkumulátor-érték kisebb mint az operandus
LE	Az akkumulátor-érték kisebb vagy egyenlő mint az operandus

Összehasonlítás - Példa

LD %MW1

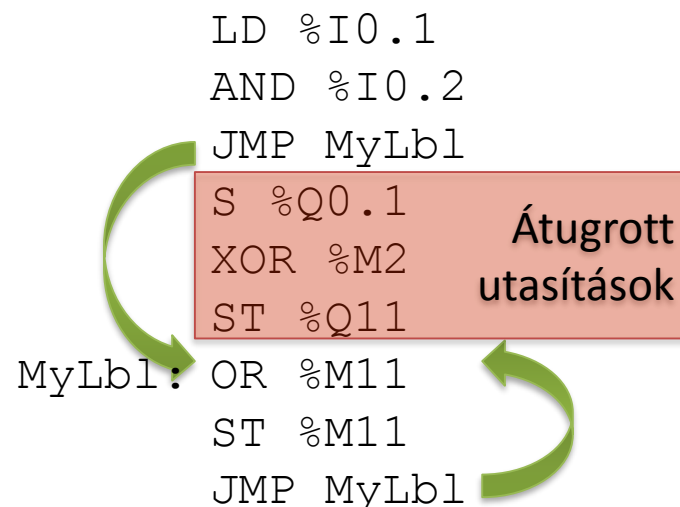


GT %MW2



Ugró és hívó utasítások

- Minden utasítás (sor) címkézhető
- A program tetszőleges címkeire ugorhat
 - Előre ugrás
 - Hátra ugrás – veszélyes!
- Az ugrás során az akkumulátor-érték nem változik



Utasítás	Leírás
JMP	Címkére ugrás
CAL	FB-példány hívása
RET	Visszatérés a hívó POU-ba

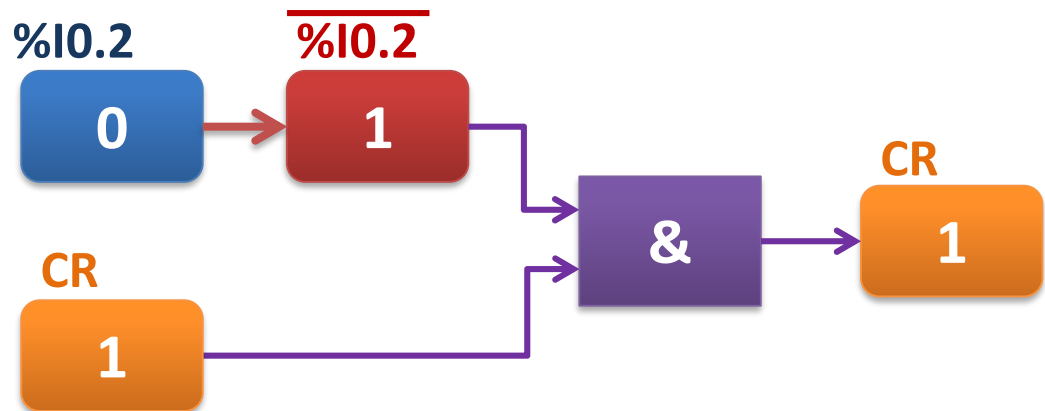
Módosítók

- Egyes utasítások működése módosító-postfixekkel befolyásolható
 - Feltételes végrehajtás – C
 - Operandus negálása – N
 - Egymásba ágyazás – (
- A módosítók kombinálhatók

Operandus negálása

- Az operandust a művelet elvégzése előtt negáljuk
- Használható a következő utasításokkal
 - Betöltés és tárolás: LDN, STN
 - Bit-logikai műveletek: ANDN, ORN, XORN

```
LD 1  
ANDN %I0.2
```



Feltételes végrehajtás

- Az utasítás csak akkor hajtódik végre, ha az akkumulátor értéke TRUE
- Feltételes ugrások, FB-hívások és visszatérés
- A következő utasításokkal használható:
 - JMP: JMPC, JMPCN
 - CAL: CALC, CALCN
 - RET: RETC, RETCN

Feltételes végrehajtás - példa

- Elvárt működés:

IF (%I0.0 AND NOT %I0.1)

THEN %MW1:=%MW1+1

ELSE %MW1:=%MW2;

LD %I0.0

ANDN %I0.1

JMPC L_THEN

LD %MW2

ST %MW1

JMP L_END

(* vagy RET *)

L_THEN:

LD %MW1

ADD 1

ST %MW1

L_END:

(* további műveletek *)

Feltételes végrehajtás - példa

- Elvárt működés:

IF (%I0.0 AND NOT %I0.1)

THEN %MW1:=%MW1+1

ELSE %MW1:=%MW2;

LD %I0.0

ANDN %I0.1

JMPCN L_ELSE

LD %MW1

ADD 1

ST %MW1

JMP L_END

L_ELSE:

LD %MW2

ST %MW1

L_END:

(* további műveletek *)

Művelet-precedencia

- $Y = A \& (B + C)$

```
LD A
AND B
OR C
ST Y
```

$Y = A \& B + C$

```
LD B
OR C
AND A
ST Y
```

Művelet-precedencia

- $Y = (A + B) \& (C + D)$

```
LD A
OR B
AND C
OR D
ST Y    Y = (A + B) & C + D
```

```
LD A
OR B
ST X
LD C
OR D
AND X
ST Y
```

```
LD A
OR B
AND (
    LD C
    OR D
)
ST Y
```

Egymásba ágyazás

$\%Q0.0 = \%I0.1 \& (\%I0.2 + (\%I0.3 \oplus \%M11))$

LD %I0.1

AND (

%I0.2

OR (

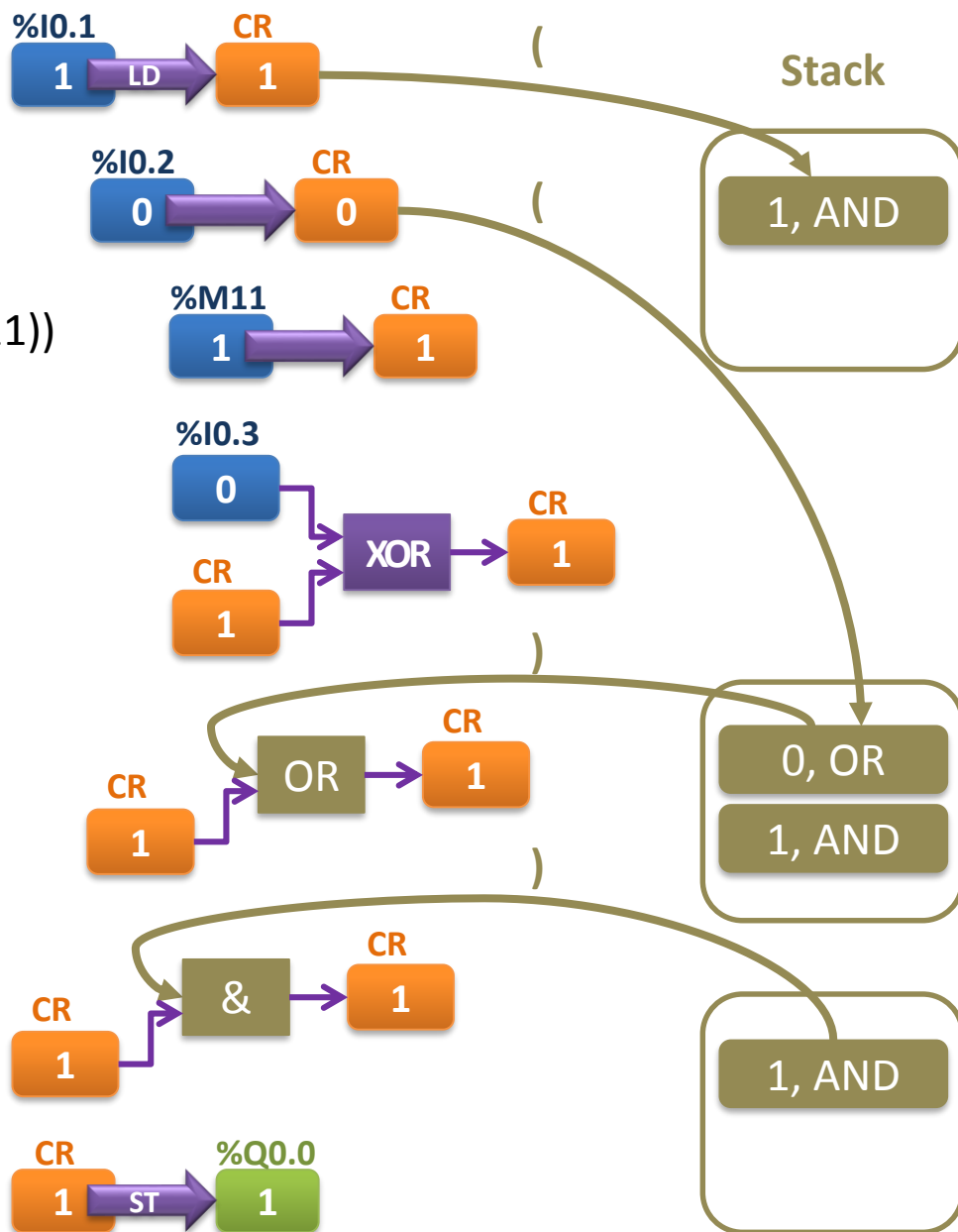
%M11

XOR %I0.3

)

)

ST %Q0.0



Függvényhívások

- IL-operátorként is használható függvények: hívás közvetlen értékekkel (pl. AND, ADD stb.)
- Más függvények (standard és felhasználói): formális paraméterek vagy közvetlen értékek
- A függvény visszatérési értéke az akkumulátorba kerül

Függvényhívás közvetlen értékekkel

- Az első paraméter mindig az akkumulátor
- A többi paraméter a függvényhívásban
- A visszatérési érték az akkumulátorba kerül

```
LD 12  
ADD 3
```

```
LD 0  
LIMIT 17, 10
```

Függvényhívás formális paraméterekkel

- Formális paraméterek megadása
 - Soronként, zárójelben
 - Bemenő paraméterek:
paraméter := érték
 - Kimenő paraméterek:
paraméter => cél-változó)
 - A paraméterek sorrendje tetszőleges,
el is hagyhatók
- A visszatérési érték az
akkumulátorba kerül

```
LIMIT (  
    EN:=TRUE ,  
    MN:=0 ,  
    IN:=MyInt ,  
    MX:=10 ,  
    ENO=>MyBool
```

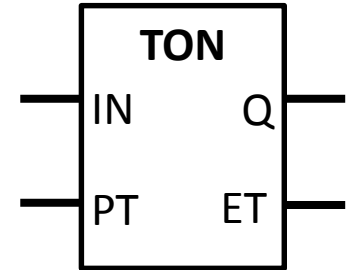
FB-hívások

- Funkcióblokk-példányok a CAL (CALC, CALCN) utasítással hívhatók
 - Hívás formális paraméterekkel
 - Hívás paraméter-hozzárendeléssel
 - Implicit hívás
- A CAL és RET utasítások az akkumulátor nem definiált értékre állítják
 - A hívott FB nem használhatja az akkumulátort érték betöltése előtt
 - A hívó POU nem használhatja az akkumulátort érték betöltése előtt

FB-hívás formális paraméterekkel

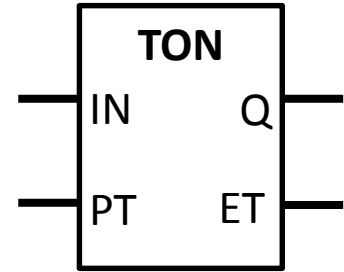
```
VAR
    TimerIn, TimerOut:    BOOL;
    TimePassed:           TIME;
    Timer1:               TON;
END_VAR

...
CAL Timer1 (
    IN:=TimerIn,
    PT:=T#5s,
    Q=>TimerOut,
    ET=>TimePassed
)
...
```



FB-hívás paraméter-hozzárendeléssel

```
VAR
    TimerIn, TimerOut:    BOOL;
    TimePassed:           TIME;
    Timer1:               TON;
END_VAR
```



```
...
LD  T#5s
ST  Timer1.PT
LD  TimerIn
ST  Timer1.IN
CAL Timer1
LD  Timer1.Q
ST  TimerOut
LD  Timer1.ET
ST  TimePassed
...
```

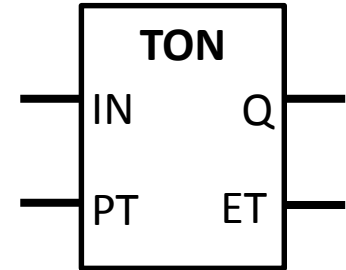
Implicit FB-hívás

```
VAR
    TimerIn, TimerOut:    BOOL;
    TimePassed:           TIME;
    Timer1:               TON;
END_VAR
```

```
...
LD T#5s
PT Timer1
LD TimerIn
IN Timer1
```

```
LD Timer1.Q
ST TimerOut
LD Timer1.ET
ST TimePassed
```

```
...
```



Példa: éldetektálás

- Az IEC61131-3 szabvány szerint az éldetektálás az **R_TRIG** and **F_TRIG** funkcióblokkokkal lehetséges

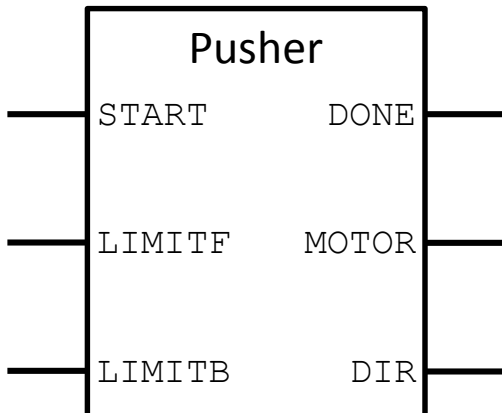
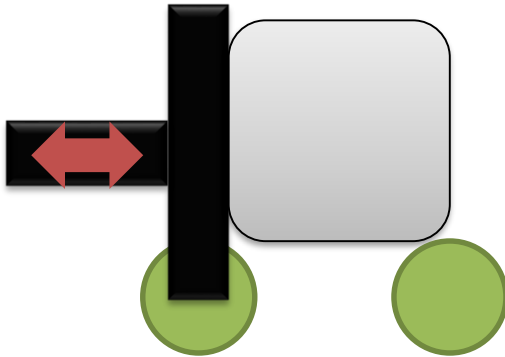


```
VAR
    RisingEdge:      R_TRIG;
    FallingEdge:     F_TRIG;
    AT %I0.0:        BOOL;
    AT %M0:          BOOL;
END_VAR

...
LD %I0.0
ST RisingEdge.CLK
CAL RisingEdge
LD RisingEdge.Q
S %M0

...
CAL FallingEdge(
    CLK:=%I0.0,
    Q=>%M0;
)
```


Példa: Tologató



- A tologató hátsó vég helyzetéből a START bemenet felfutó élére indul
- Addig mozog előre, amíg az első végálláskapcsoló nem jelez
- Ekkor irányt vált és a hátsó végálláskapcsoló jelzéséig mozog hátra
- Amikor visszatért a kiindulási helyzetbe, a DONE kimenetet egyetlen ciklus idejére igazra állítja

Tologató – Deklarációs rész

```
FUNCTION_BLOCK Pusher
VAR_INPUT
    Start:      BOOL;
    LimitF:     BOOL;
    LimitB:     BOOL;
END_VAR
VAR_OUTPUT RETAIN
    Motor:      BOOL:=FALSE;
    Dir:        BOOL:=FALSE;
    Done:       BOOL:=FALSE;
END_VAR
VAR
    R_Back:     R_TRIG;
    R_Start:    R_TRIG;
END_VAR
```

Tologató – Programkód

```
CAL R_Back(CLK:=LimitB,  
          Q=>Done)
```

```
LD Start
```

```
ST R_Start.CLK
```

```
CAL R_Start
```

```
LD R_Start.Q
```

```
AND LimitB
```

```
JMPC MoveFwd
```

```
LD LimitF
```

```
JMPC MoveBwd
```

```
LD R_Back.Q
```

```
R Motor
```

```
RET
```

```
MoveFwd: LD 1
```

```
S Motor
```

```
S Dir
```

```
RET
```

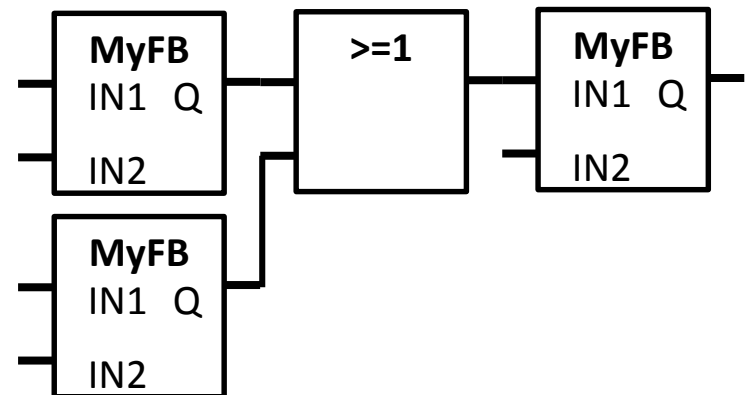
```
MoveBwd: LD 1
```

```
R Dir
```

```
RET
```

Funkcióblokk-diagram (Function Block Diagram, FBD)

- Magas szintű grafikus nyelv
- Nem logikai műveletek végzésére
- Függvények és FB-k közötti adatfolyam leírása
- Eredet: adatfolyam-diagram
 - Blokkok: függvények és FB-példányok
 - Összeköttetések: változók



FBD-hálózat (network)

- FBD-programkódok szervezőeleme
- A hálózatok felülről lefelé hajtódnak végre
- A hálózatokhoz címkét kapcsolhatunk
- A címkékre ugorhatunk
- A hálózatokat összekötőkkel (connector) kapcsolhatjuk össze

FBD-hálózatok elemei

- Függvény- és FB-hívások (blokkok)
- Vezetékek
- Futásvezérlési elemek
- Összekötők

Függvényhívás

```
FUNCTION MyFun : INT  
VAR_INPUT  
    In1 : BOOL;  
    In2 : BOOL;  
END_VAR  
VAR_OUTPUT  
    Out1 : BOOL;  
END_VAR  
VAR_IN_OUT  
    InOut : INT;  
END_VAR
```

Függvény neve

MyFun

Visszatérési érték
(informális)

IN1

IN2

InOut

OUT1

InOut

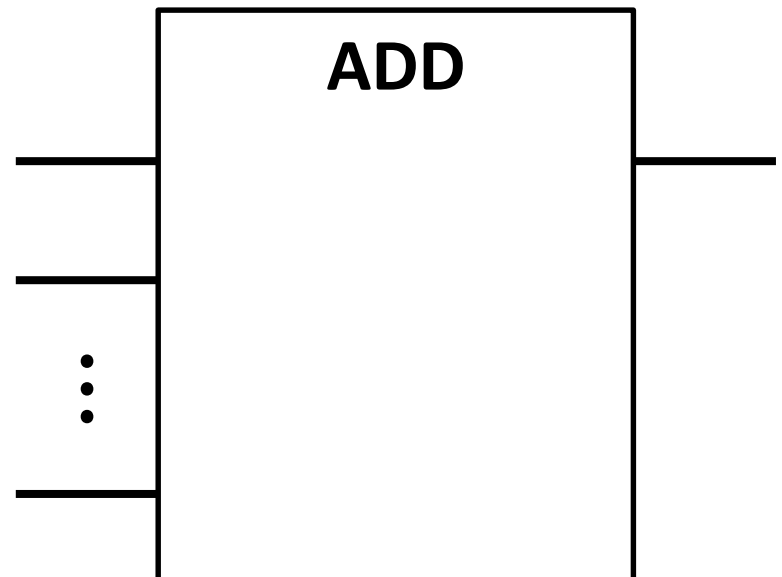
Bemeneti paraméter(ek)
(formális)

Be- és kimeneti paraméterek
(formális)

Kimeneti paraméter(ek)
(formális)

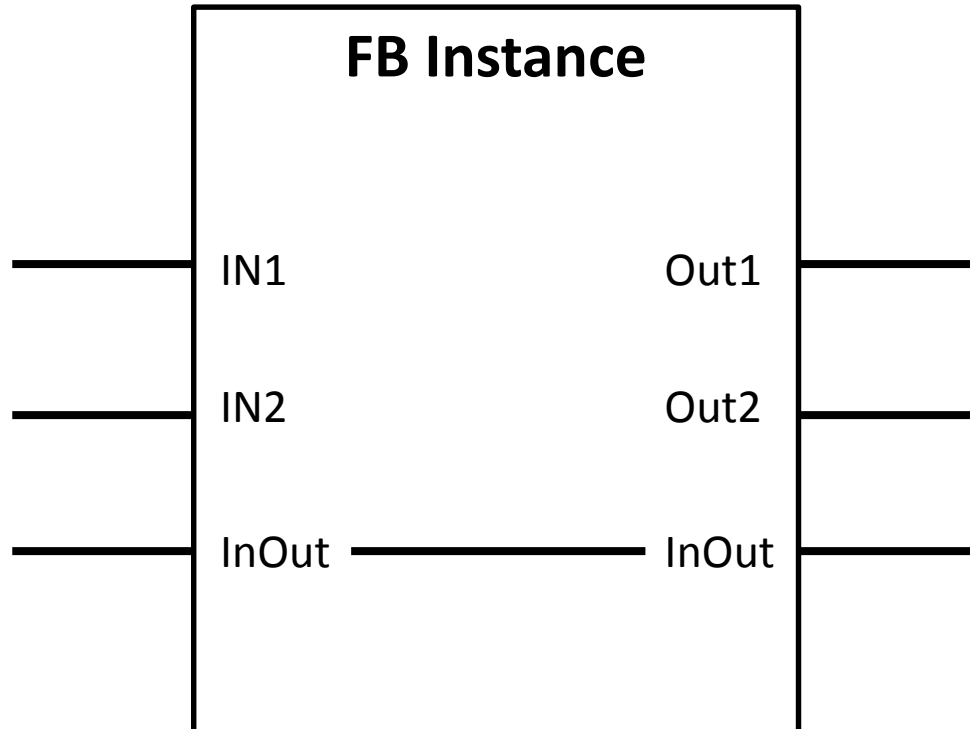
Függvényhívás

- Overloaded és kiterjeszthető függvények formális paraméterek nélkül hívhatók
 - ADD (+), MUL (*)
 - AND (&), OR (≥ 1), XOR ($= 2k+1$)



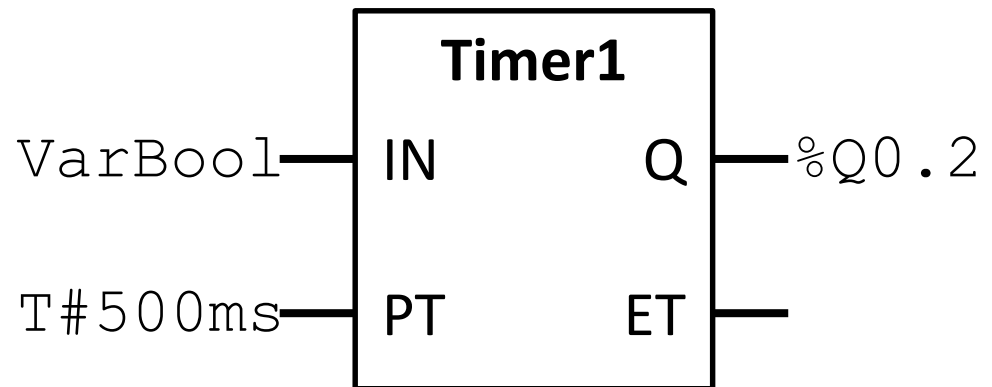
FB-hívás

Minden paraméter formális



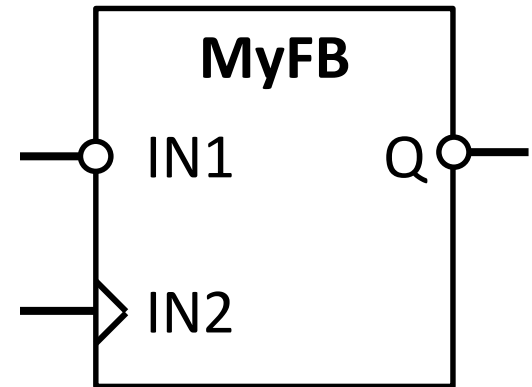
Blokkok be- és kimenetei

- Változók
- Literálisok
- Vezetékek
- Összekötők
- Nem bekötött
 - Bemenet: adattípus alapértelmezett értéke
 - Kimenet: nem történik hozzárendelés



Különleges be- és kimenetek

- Negált bemenet/kimenet : o
- Érzékeny bemenetek:
 - Felfutó él: >
 - Lefutó él: <

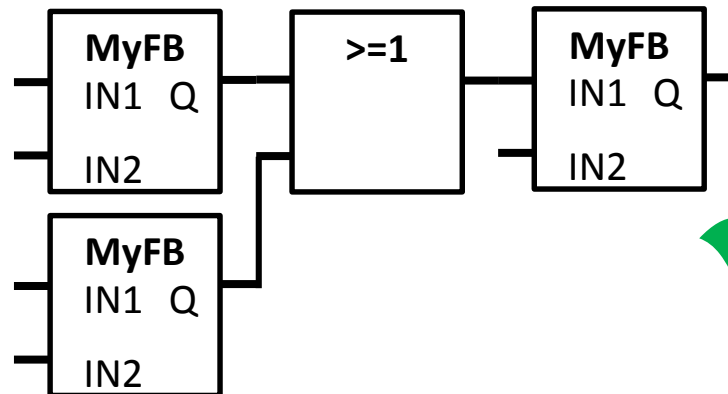
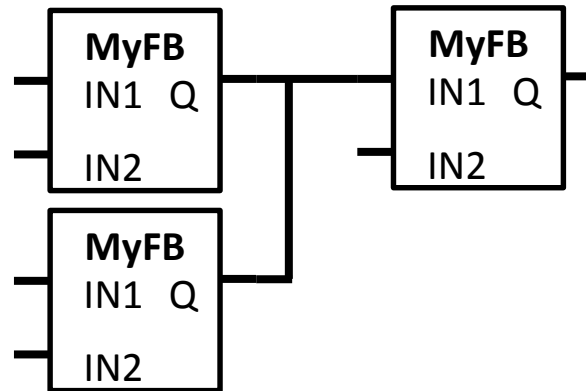
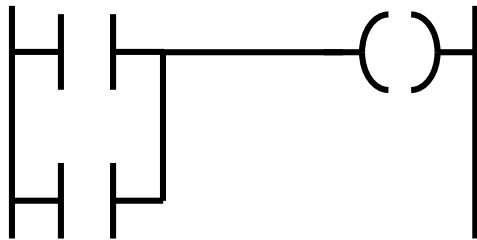


Vezetékek

- Blokkokat összekötő vízszintes vagy függőleges vezetékek
- Tetszőleges adattípusúak lehetnek
- Csak azonos adattípusú blokk be- és kimenetek köthetők össze
- Egy blokk-kimenet tetszőleges számú blokk-bemenethez köthető
- Egy blokk-bemenethez csak egyetlen jel köthető

Összekötések

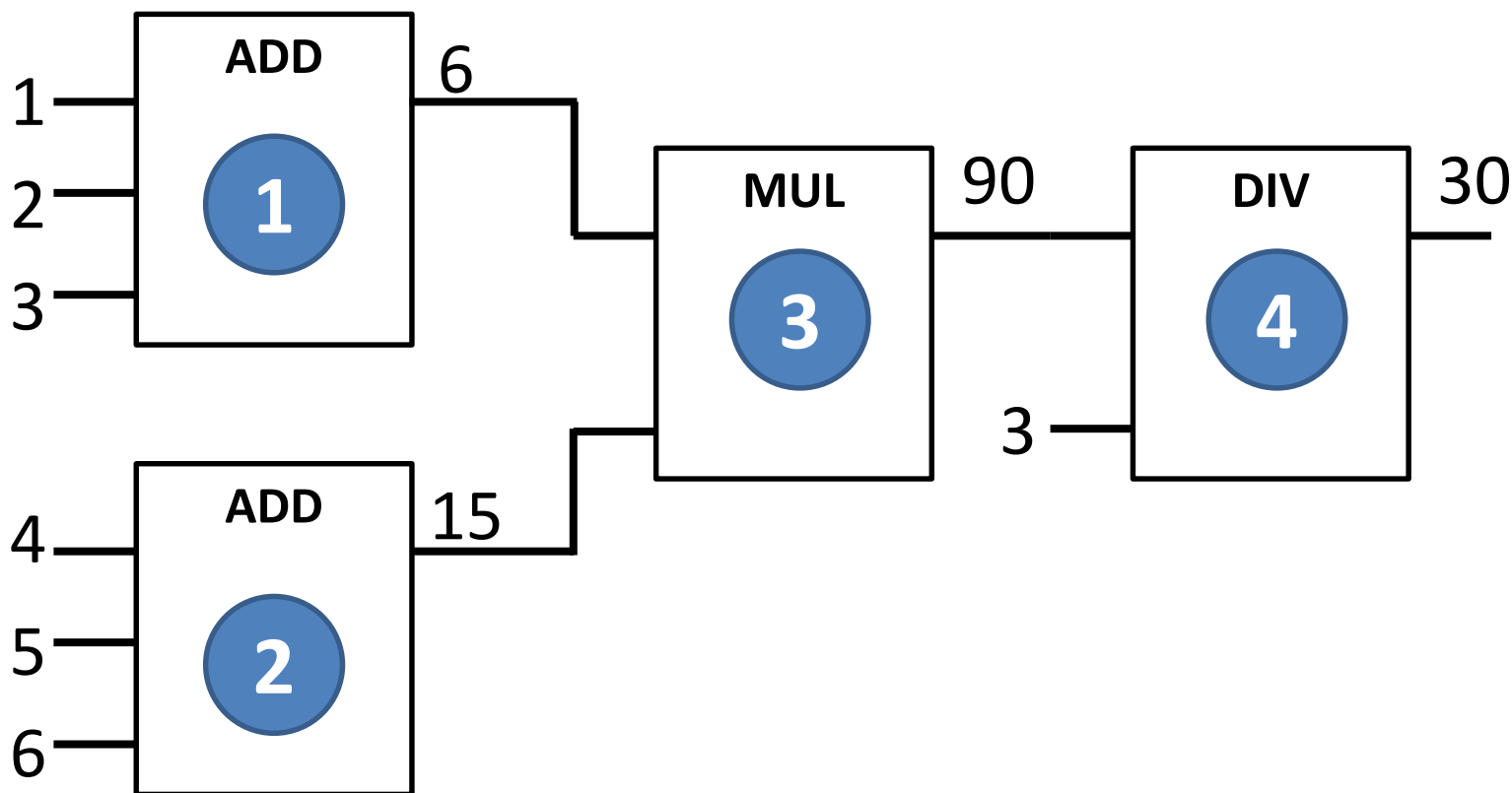
A huzalozott VAGY kapcsolat FBD-ben nem megengedett!



Hálózat kiértékelése

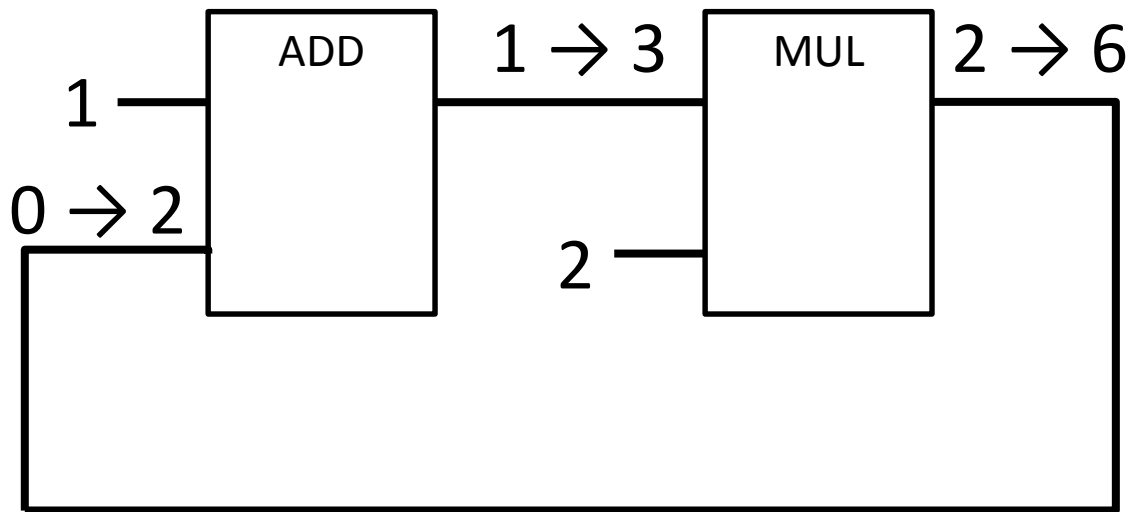
- Egy elem csak akkor értékelhető ki, ha minden bemenetén érvényes érték áll rendelkezésre
- Egy elem kiértékelése mindaddig nem fejeződhet be, amíg mindegyik kimenete nem kerül kiértékelésre
- Egy hálózat kiértékelése mindaddig nem fejeződhet be, amíg mindegyik eleme nem kerül kiértékelésre

Hálózat kiértékelése - példa



Visszacsatolás

- Visszacsatolási hurok különböző blokkok között
- Az utolsó ciklusban kapott kimeneti értéket csatolja vissza
- Első futáskor az adattípus kezdeti értékét kapja



Futásvezérlés

- Ugrás

- Csak feltételes

————>>> LABEL

- A megadott címkére ugrik, ha a bemenete igazra értékelődik ki

- Visszatérés

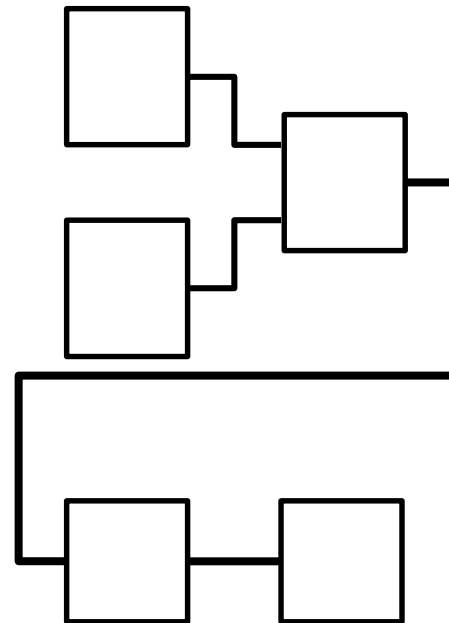
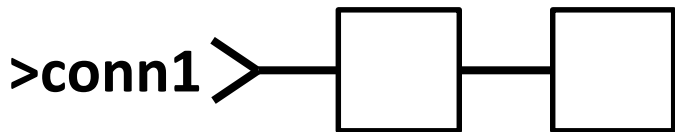
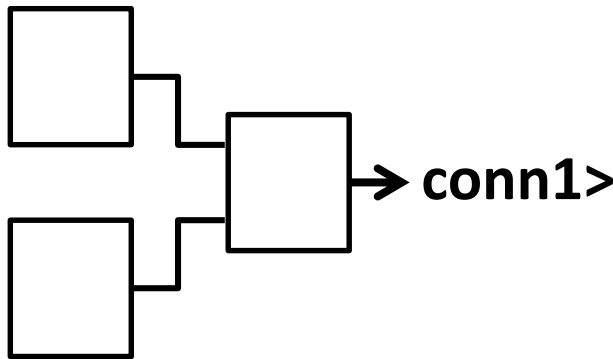
————<RETURN>

- Csak feltételes

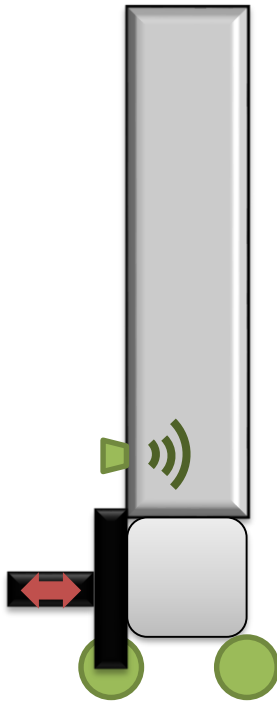
- Visszatér a hívó POU-ba, ha a bemenete igazra értékelődik ki

Összekötők

- Hálózaton belüli adattovábbításra szolgál
- „Új sor karakter” vagy hosszú nyíl
- Hasznos, ha a diagram szélessége korlátozott



Példa - futószalag



- Indításkor a futószalagnak működnie kell
- Ha a futószalag közelítésérzékelője jelez, akkor további 3 másodpercig járassuk a futószalagot, majd állítsuk le és indítsuk el a tologatót
- Ha a tologató végzett, indítsuk újra a futószalagot

Futószalag – Deklarációs rész

Program PusherConveyor

VAR

Proxy1:	BOOL AT %I0.0;
LF:	BOOL AT %I0.1;
LB:	BOOL AT %I0.2;
Conv1:	BOOL :=1 AT %Q0.0;
P1Mot:	BOOL AT %Q0.1;
P1Dir:	BOOL AT %Q0.2;
T1:	TP;
F_T:	F_TRIG;
Pusher1:	Pusher;
FF:	SR;

END_VAR

Futószalag - Programkód

