

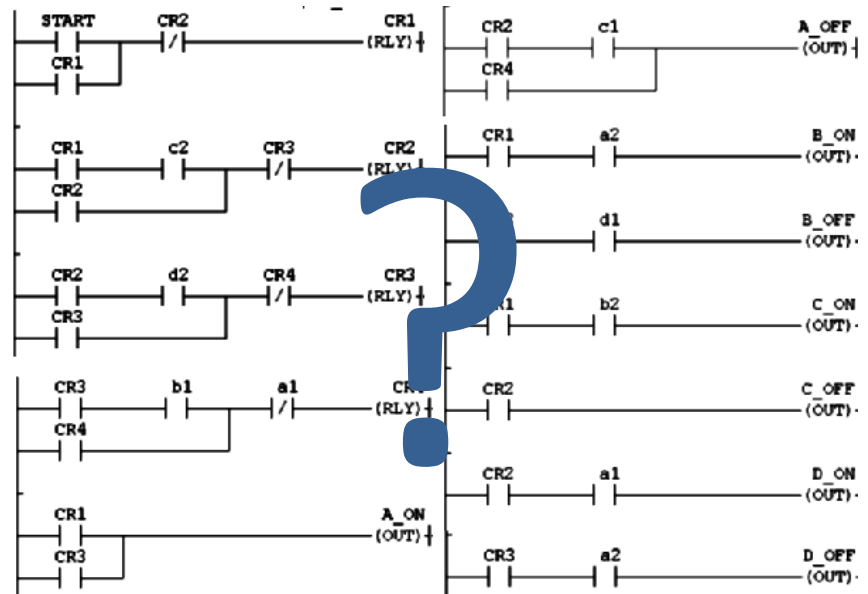
Az IEC 61131-3 szabvány szoftvermodellje

Programozható irányítóberendezések
és szenzorrendszerek

KOVÁCS Gábor
gkovacs@iit.bme.hu

A SMORES-elv

- A jó program
 - Scalable
 - MODular
 - Reusable
 - Extensible
 - Simple



Az IEC 61131 szabvány kialakítása

- A gyártók egymástól függetlenül fejlesztenek
- Kialakulnak jól használható, általánosan elterjedő elvek és módszerek
- Ezek kezelése azonban nem egységes
- 1979: egy egységes szabvány kialakításának kezdete
- 1985: az IEC 61131 szabvány első változata

Az IEC 61131 szabvány

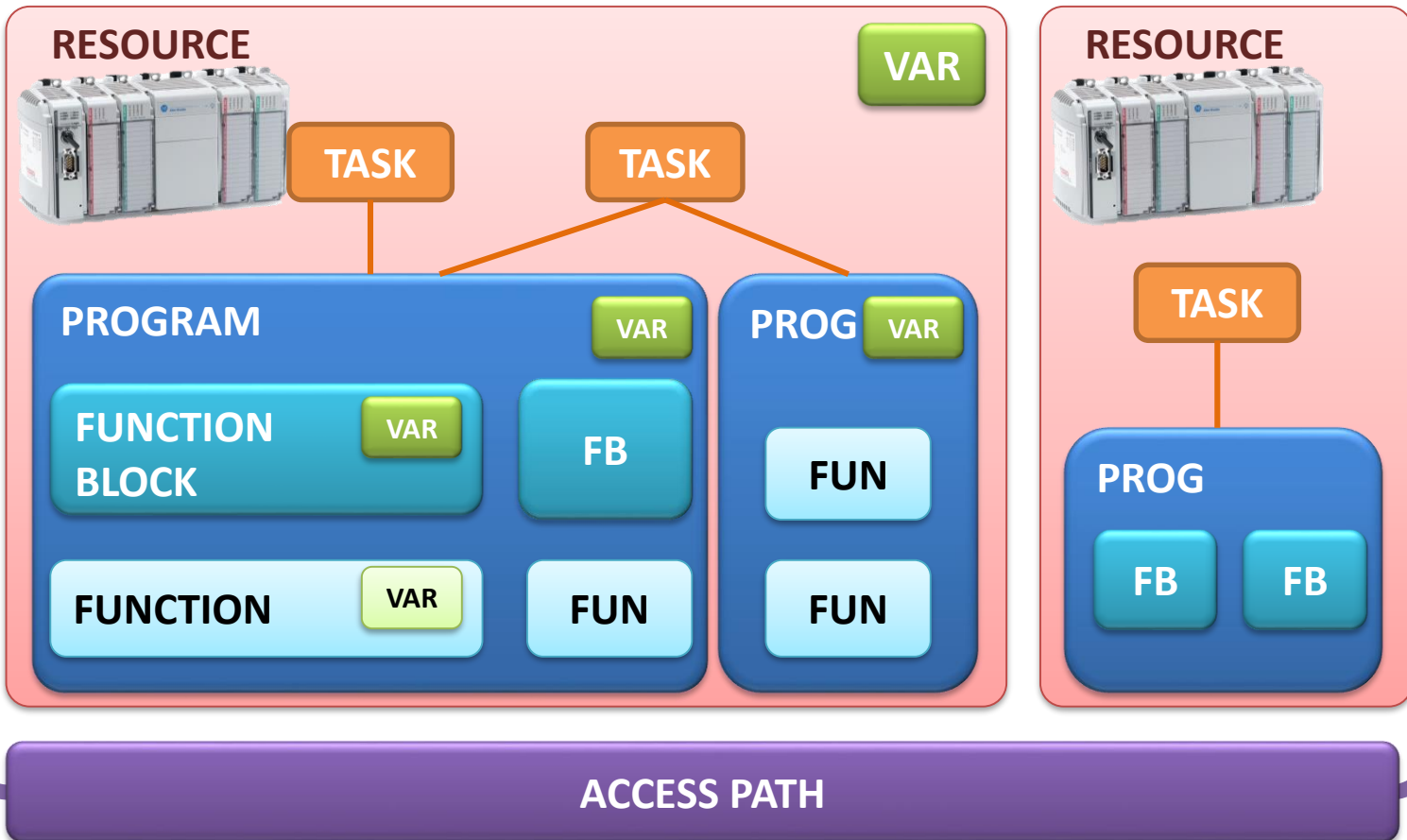
- IEC 61131 – Programozható irányítóberendezések (*Programmable controllers*)
- A szabvány részei
 1. Általános információk (ed. 2, 2003)
 2. Eszközökkel szemben támasztott követelmények (ed. 2, 2007)
 - 3. Programozási nyelvek (és szoftvermodell) (ed. 3, 2013)**
 4. Felhasználói útmutatások (ed. 2, 2004)
 5. Kommunikáció (2000)
 6. Funkcionális biztonság (2012)
 7. Fuzzy logikai irányítás (2000)
 8. Programozási nyelvek implementációs útmutatása (2003)
 9. Digitális szenzor-interfész (2013)

Az IEC 61131-3 szabvány alkalmazása

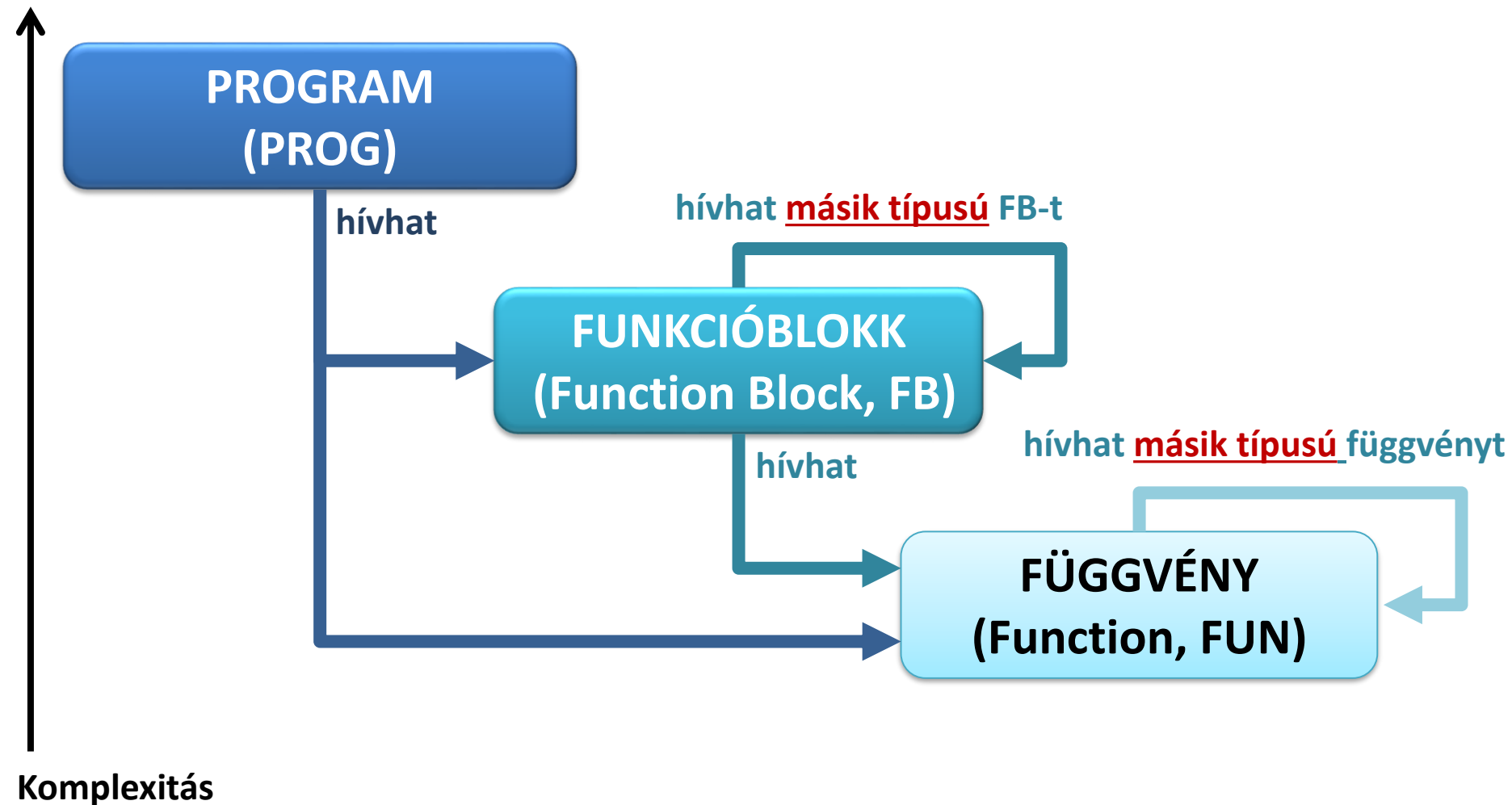
- A szabványt a nagyobb gyártók eleinte inkább csak ajánlásnak tekintették
- Az elterjedt komplex fejlesztői környezetek többé-kevésbé megfelelnek a szabványnak (*IEC 61131-3 compliant*)
- Az alapelvek teljesülnek, de
 - más megnevezések
 - néhány (esetenként fontos) részletkérdésben nem szabványosak

Áttekintés

CONFIGURATION



Programszervezési egységek (Program Organisation Unit – POU)



Függvény

- Cél: az egyszerű PLC-műveletek körének kiterjesztése
- Azonos paraméterekkel hívva mindig ugyanazt az eredményt szolgáltatja
- Nem emlékezik: az előző hívás eredményét elfelejti
- Hívhat egy másik függvényt
- Példa: $\sin(x)$

Funkcióblokk

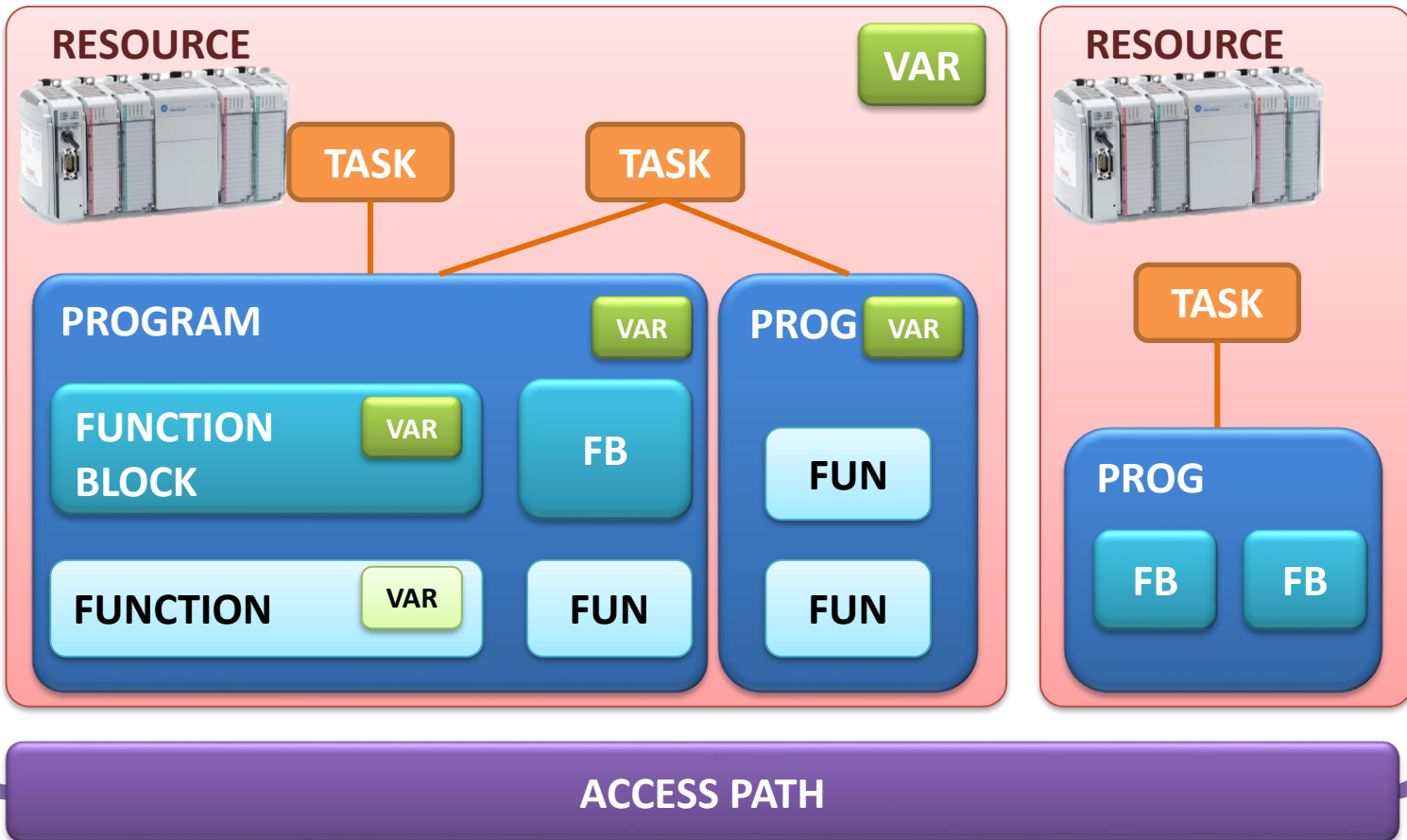
- Független, a feldolgozó algoritmussal egységbe foglalt adatstruktúra
- Leggyakrabban használt építőelem
 - Standard funkcióblokkok (pl. időzítők)
 - Felhasználói funkcióblokkok
- A funkcióblokk hívhat
 - Függvényeket
 - Más típusú funkcióblokkokat

Program

- „Főprogram”
- Felhasználhat fizikai objektumokat (ki- és bemenetek) is
- Magas szintű működést ír le a megfelelő függvények és funkcióblokkok hívásával

Áttekintés

CONFIGURATION



Taszkok és erőforrások

- Taszkok (*Task*)
 - A POUk végrehajtásáért felelnek
 - Egy taszk több programot vagy FB-t is végrehajthat
 - Egy program vagy FB több taszkhoz is rendelhető
- Erőforrások (*Resource*)
 - Taszkok fizikai erőforrásokhoz (CPU) rendelése
 - Változók fizikai objektumokhoz (kimenetek, bemenetek, fix memóriacím) való implementáció-specifikus hozzárendelése

Konfiguráció és Access path

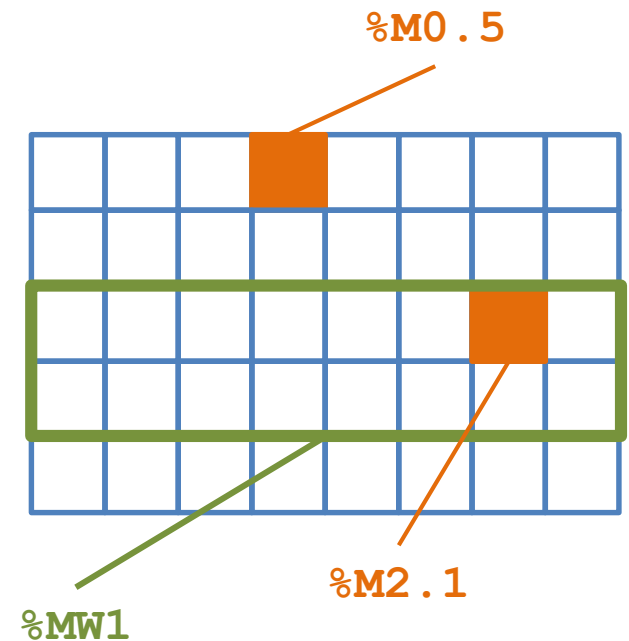
- Konfiguráció (*Configuration*)
 - Erőforrások egy csoportját fogja össze
 - Több erőforráson érvényes globális változók használata
- Access path
 - Kommunikációs csatorna különböző konfigurációk között
 - Kényelmes és egyszerű változócsere biztosítása

Azonosítók

- Az elemekre azonosítóikkal hivatkozhatunk
- Betűk, számok, alulvonás (pl. `LEVEL_SWITCH_1`)
- Az azonosítók betűvel kezdődnek és nem végződhetnek `__`-re (két alulvonás)
- Az azonosítók nem case sensitive-ek

Memóriamodell

- Memóriaobjektumok egységes elérése
- Nincs dedikált bit-, szó-, duplaszó-memória
- Változódeklarálás a magas szintű nyelvekben megszokott módon
- A közvetlen értékek (fizikai objektumok, pl. bemenetek) használata korlátozott



Programszervezési egységek (*Program Organization Unit, POU*)



A POU-k szerkezete

POU típus és azonosító

Deklarációs rész

- Interfész-változók
- Helyi változók
- Globális változók

POU törzs: utasítások

- 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

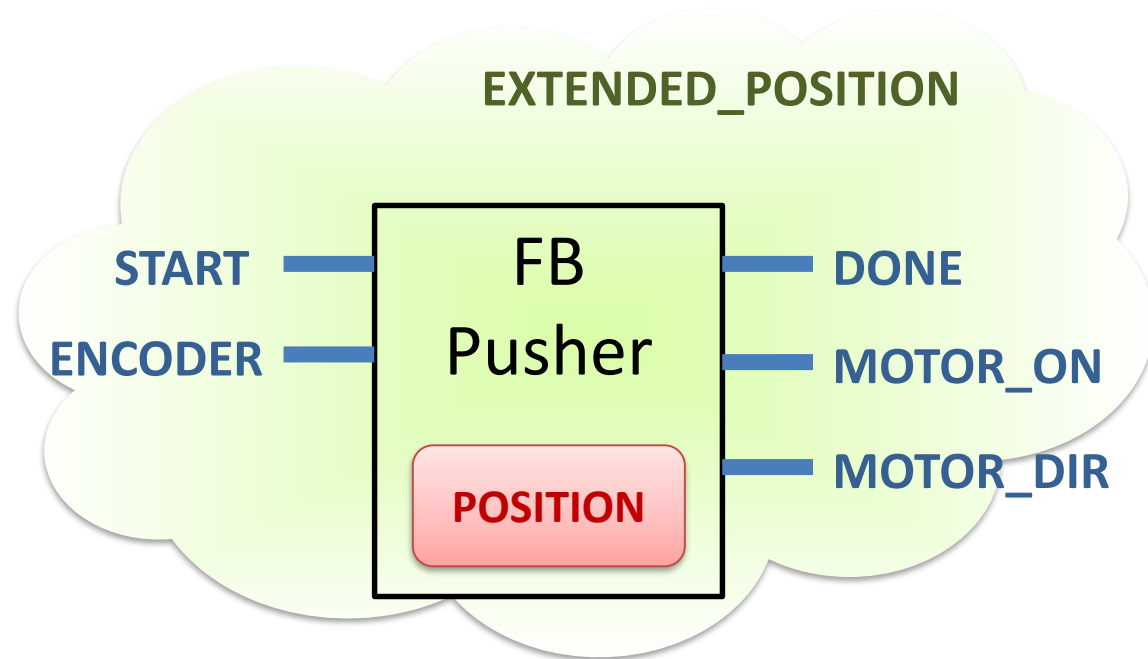
FUNCTION IsReady : BOOL

Deklarációs rész

- A POU-ban használt változókat explicite deklarálni kell
 - Adattípus (ld. később)
 - Attribútumok (ld. később)
- A változók automatikusan kerülnek hozzárendelésre a memóriacímekhez (kivéve ha explicite deklaráljuk, mely fizikai objektumhoz tartoznak)

Változótípusok

- A változók deklarálása típus szerinti csoportokban
 - VAR ... END_VAR direktívák
- Változótípusok
 - Lokális
 - Interfész
 - Globális



Változótípusok – Lokális változók

- VAR – lokális változó
 - Hozzáférés: csak POU-n belül
 - Megőrzi az értékét a hívások között (kivétel: függvény)
- VAR_TEMP – Ideiglenes változó
 - Hozzáférés: csak POU-n belül
 - A memórfoglalás dinamikusan történik a POU hívásakor, az értéke nem őrződik meg
 - Függvényekben nem megengedett

Változótípusok – Interfész változók

- VAR_INPUT
 - Hívó POU: Írás / Olvasás
 - Hívott POU: Olvasás
- VAR_OUTPUT
 - Hívó POU: Olvasás
 - Hívott POU: Írás / Olvasás
- VAR_IN_OUT
 - Hívó POU: Írás / Olvasás
 - Hívott POU: Írás / Olvasás

Interfész változók átadása

- VAR_IN és VAR_OUT: érték szerint
 - Az érték kerül átadásra (pl. 4, „text”)
 - A hívott POU nem tudja módosítani a bemeneti paraméter értékét
 - A hívó POU nem tudja módosítani a kimeneti paraméter értékét
- VAR_IN_OUT: referencia szerint
 - A paraméter memóriacíme kerül átadásra
 - A hívott és a hívó POU is módosíthatja a paraméter értékét

Paraméterátadás

Hívó POU

```
PROGRAM P1
VAR
    MyInt1 : INT;
    MyInt2 : INT;
    MyInt3 : INT;
    MyFB    : FBType;
END_VAR

MyInt1:=4;
MyInt3:=2;
MyFB (  InVar:=MyInt1,
        IOVar:=MyInt3,
        OutVar=>MyInt2 )
```

...

Hívás

Hívott POU

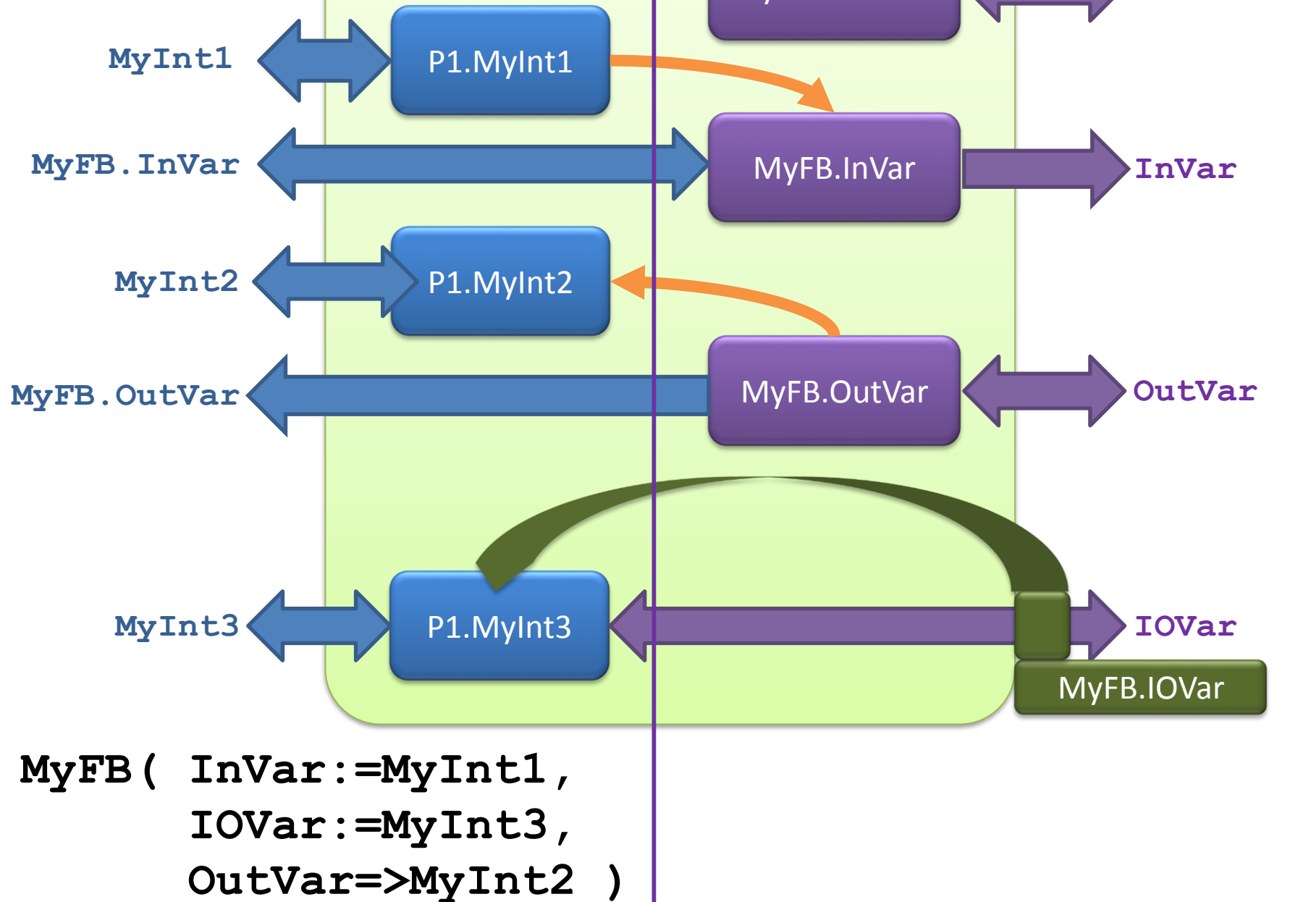
```
FUNCTION_BLOCK FBType
VAR_INPUT
    InVar : INT;
END_VAR
VAR_OUTPUT
    OutVar : INT;
END_VAR
VAR_IN_OUT
    IOVar : INT;
END_VAR
VAR
    LocalVar : INT;
END_VAR
```

...

Hívó POU (P1)

Memória

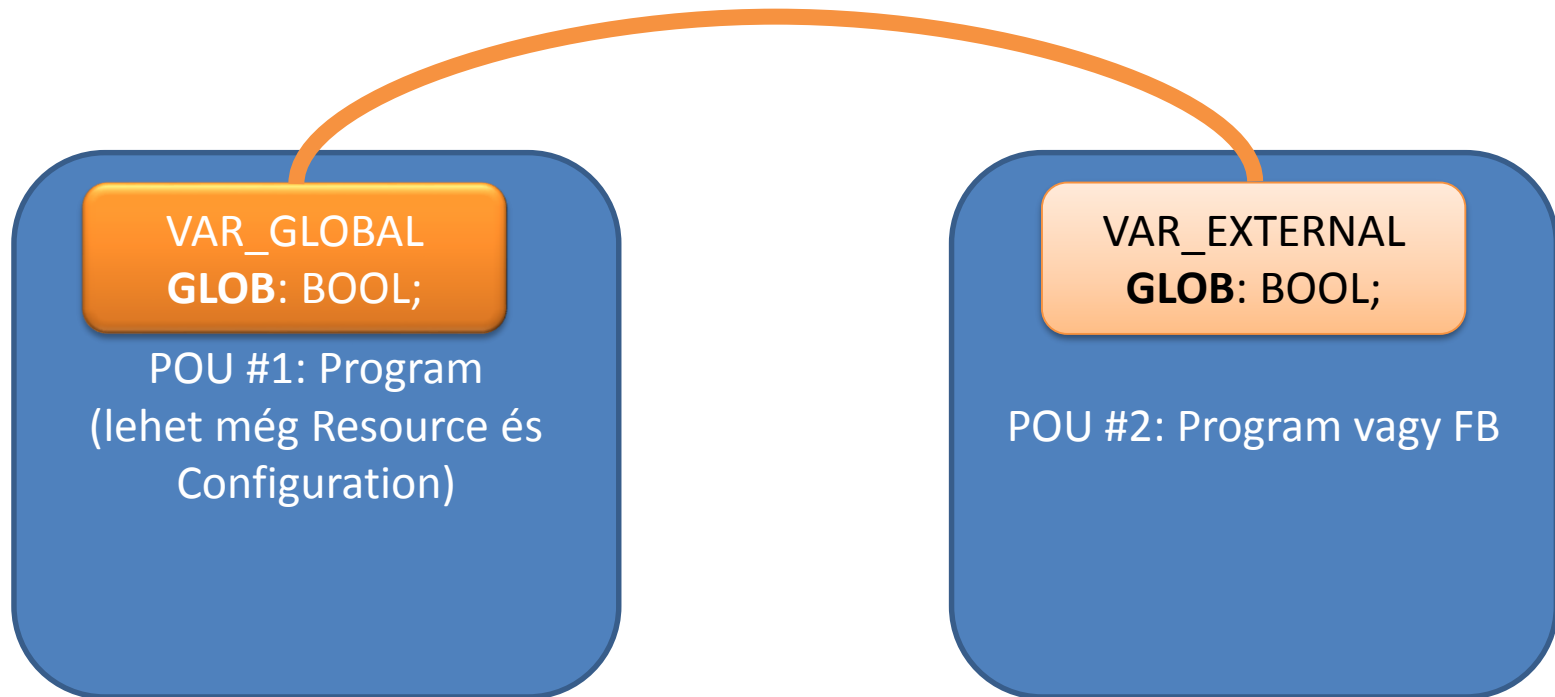
Hívott POU (MyFB)



Globális változók

- A globális változók több POU-ból is elérhetők
- Azonosítás változó-azonosítójuk (nevük) alapján
- Csak program típusú POU-ban hozhatók létre, de FB-ben is használhatók
- Azok a POU-k használhatják, melyeket a változót létrehozó POU hív
- Szerepeltetés a deklarációs részben
 - VAR_GLOBAL: abban a POU-ban, ahol deklaráljuk
 - VAR_EXTERNAL: a többi POU-ban, ahol használjuk

Globális változók



Access path változók

- Adatcsere a konfigurációk között
- Csak Program típusú POU-ban használható
- External változóként viselkedik

POU típusonként megengedett változótípusok

Változótípus		Program	Funkcióblokk	Függvény
Lokális változók	VAR	+	+	+
	VAR_TEMP	+	+	-
Interfész változók	VAR_INPUT	+	+	+
	VAR_OUTPUT	+	+	+
	VAR_IN_OUT	+	+	+
Globális változók	VAR_EXTERNAL	+	+	-
	VAR_GLOBAL	+	-	-
	VAR_ACCESS	+	-	-

Hozzáférés a változókhoz

Változó típus		Belső hozzáférés	Külső hozzáférés
Lokális változók	VAR	RW	—
	VAR_TEMP	RW	—
Interfész változók	VAR_INPUT	R	RW
	VAR_OUTPUT	RW	R
	VAR_IN_OUT	RW	RW
Globális változók	VAR_EXTERNAL	RW	RW
	VAR_GLOBAL	RW	RW
	VAR_ACCESS	RW	RW

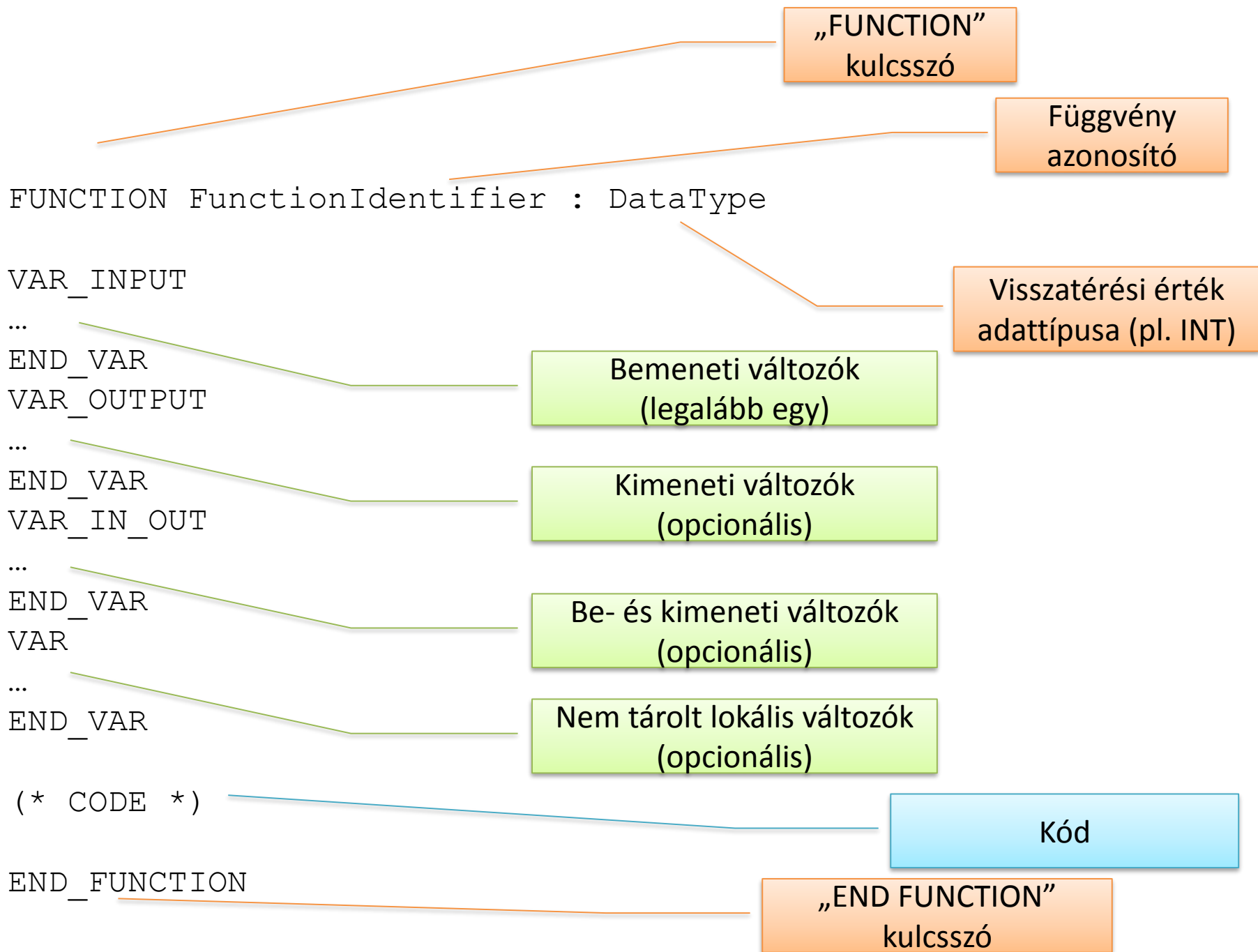
R: csak olvasható
RW: írható és olvasható

Programszervezési egységek típusai



Függvény

- Azonos bemenetek mellett mindig azonos eredménnyel tér vissza
- Nincs emlékezete: a változók értéke nem őrződik meg a hívások között
- Egy projekten belül látható és hívható minden POU számára
- Kötelezően egyetlen visszatérési érték, azonosítója a függvény azonosítójával egyezik meg
- Opcionálisan több kimeneti változó
- Függvények csak más függvényeket hívhatnak



Példa: dB

```
FUNCTION DB : REAL
```

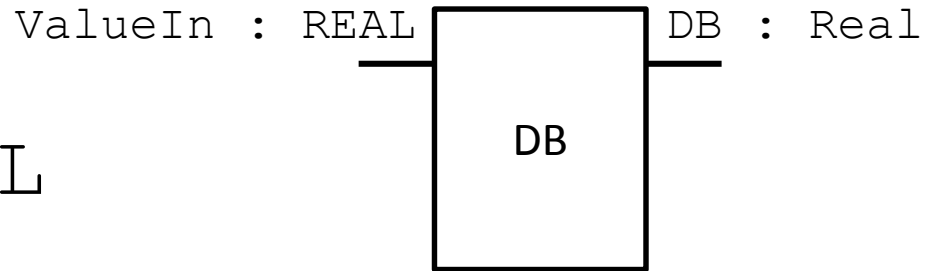
```
VAR_INPUT
```

```
    ValueIn: REAL;
```

```
END_VAR
```

```
DB:=20*LOG(ValueIn);
```

```
END_FUNCTION
```



Függvények tulajdonságai

- Overloaded függvények
 - Többféle adattípussal használhatók
 - Adattípus: ANY, ANY_INT stb.
- Kiterjeszthető (extensible) függvények
 - Tetszőleges számú bemenettel használhatók
- EN / ENO (létradiagram, FB diagram)
 - Enable In / Enable out
 - Nem logikai be- és kimenetű függvények beillesztése az áramútba

Standard függvények

- Minden IEC-61131 kompatibilis fejlesztői környezetben megtalálhatók
- Gyakran használt függvények a különféle műveletekre
- Csak standard függvények lehetnek overloaded és kiterjeszthető típusúak

Típuskonverziós függvények

- Típuskonverzió
 - *_TO_*, pl. REAL_TO_INT
- Csonkolás
 - TRUNC
- BCD-bináris konverzió
 - BCD_TO_*, pl. BCD_TO_WORD
 - *_TO_BCD, pl. WORD_TO_BCD

Numerikus és aritmetikai függvények

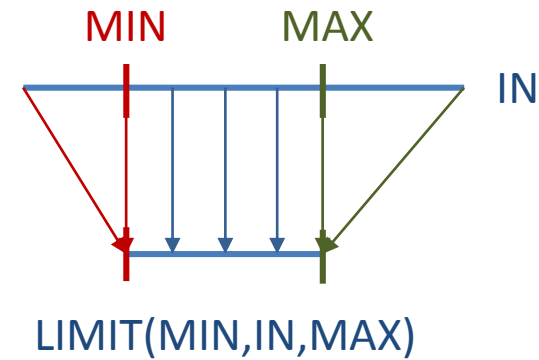
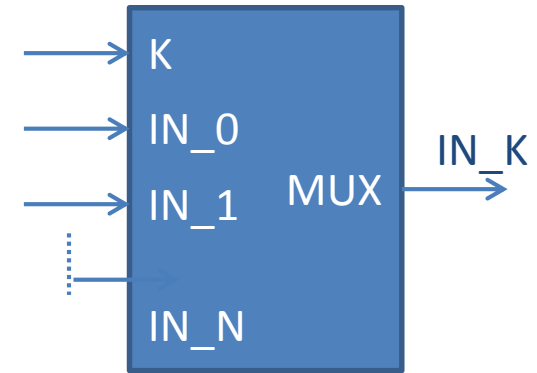
- Numerikus függvények:
 - ABS
 - SQRT
 - LN, LOG, EXP
 - SIN, COS, TAN, ASIN, ACOS, ATAN
- Aritmetikai függvények:
 - ADD (+), MUL (*): kiterjeszthető (extensible)
 - SUB (-), DIV (/), MOD
 - EXPT - $OUT := IN1^{IN2}$
 - MOVE - $OUT := IN$

Bit string függvények

- Bit shift függvények
 - SHL, SHR – eltolás N bittel, nulla betöltése
 - ROR, ROL – forgatás N bittel
- Bitenkénti logikai függvények
 - AND, OR, XOR – kiterjeszthetők
 - NOT

Kiválasztó függvények

- MUX
 - $A := \text{MUX}(1, B, C, D)$ eredménye: $A = C$
 - Kiterjeszthető, overloaded
 - SEL függvény: bináris kiválasztás két bemenet között
- MIN, MAX
 - Kiterjeszthető, overloaded függvény
- LIMIT
 - $\text{Limit}(\text{Min}, \text{In}, \text{Max})$
 - $\text{Limit} := \text{MIN}(\text{MAX}(\text{In}, \text{MinVal}), \text{MaxVal})$
 - Kiterjeszthető, overloaded



Összehasonlító függvények

- GT ($>$) , GE (\geq)
- EQ ($=$) , NE (\neq)
- LE (\leq) , LT ($<$)
- Kiterjeszthetők:
- $\text{GT}(\text{IN1}, \text{IN2}, \text{IN3}) =$
 $= (\text{IN1} > \text{IN2}) \& (\text{IN2} > \text{IN3})$

Karakterlánc-függvények

- LEN: hossz
- LEFT, RIGHT, MID: rész-sztring kiválasztása
- CONCAT: összefűzés (kiterjeszthető)
- INSERT, DELETE, REPLACE, FIND: rész-sztring beillesztése, törlése, cseréje, keresése

Felhasználói függvények

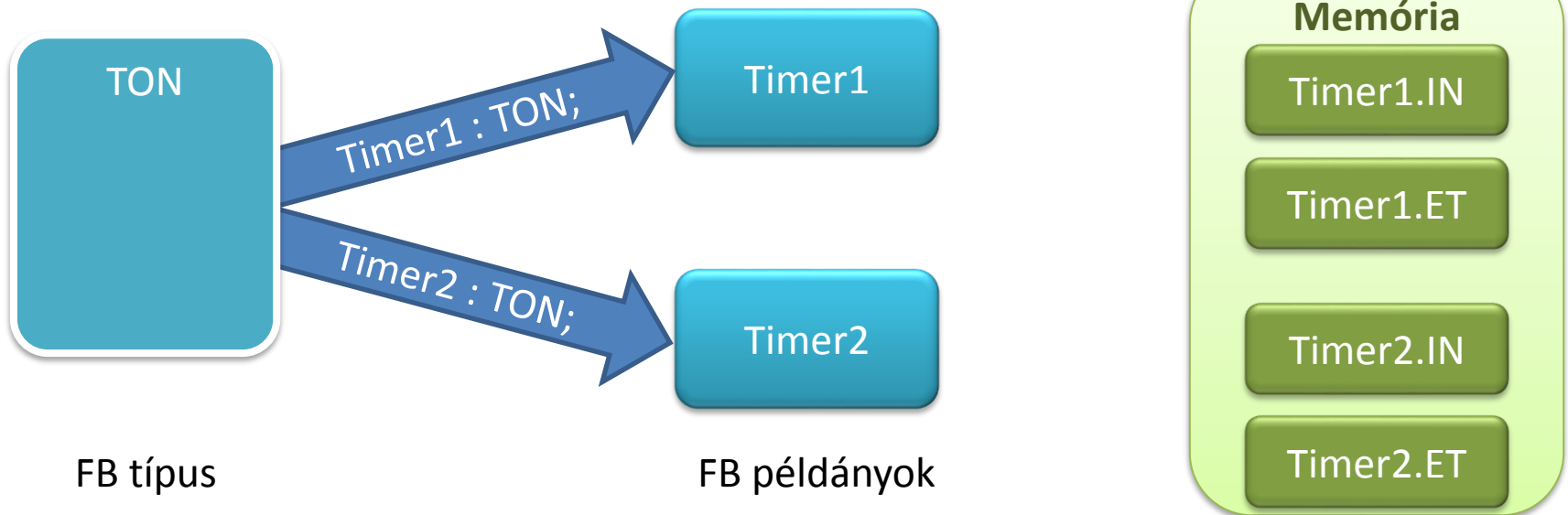
- A felhasználó tetszőleges függvényt definiálhat
- A felhasználói függvények az egész projektben elérhetők
- Hívhatnak standard és más felhasználói függvényeket is

Funkcióblokkok

„A funkcióblokk egy független, egy adatstruktúrát és az azon működő algoritmust egységbe foglaló entitás.”

Funkcióblokk példányosítása

- Funkcióblokk-típus (*FB type*) \approx Osztály (class)
- Funkcióblokk-példány (*FB instance*) \approx Objektum
 - A funkcióblokk-típust példányosítani kell, majd a példányt hívni
 - A változók az egyes példányokhoz rendelvek
 - A példányok közvetlenül nem befolyásolják egymás működését



Példány: egységbe foglalás

- Példányosítás: memória foglálás a POU-példány számára:
 - VAR_INPUT
 - VAR_OUTPUT
 - VAR
- Statikus memória szorosan a példányhoz csatolva: az értékek megőrződnek a hívások között
- VAR_TEMP: dinamikusan foglalt memória, nem őrzi meg az értékeket a hívások között
- Formailag egy változó deklarálásával egyezik meg:
`MyTimer: MyFBType;`

Példány: struktúra

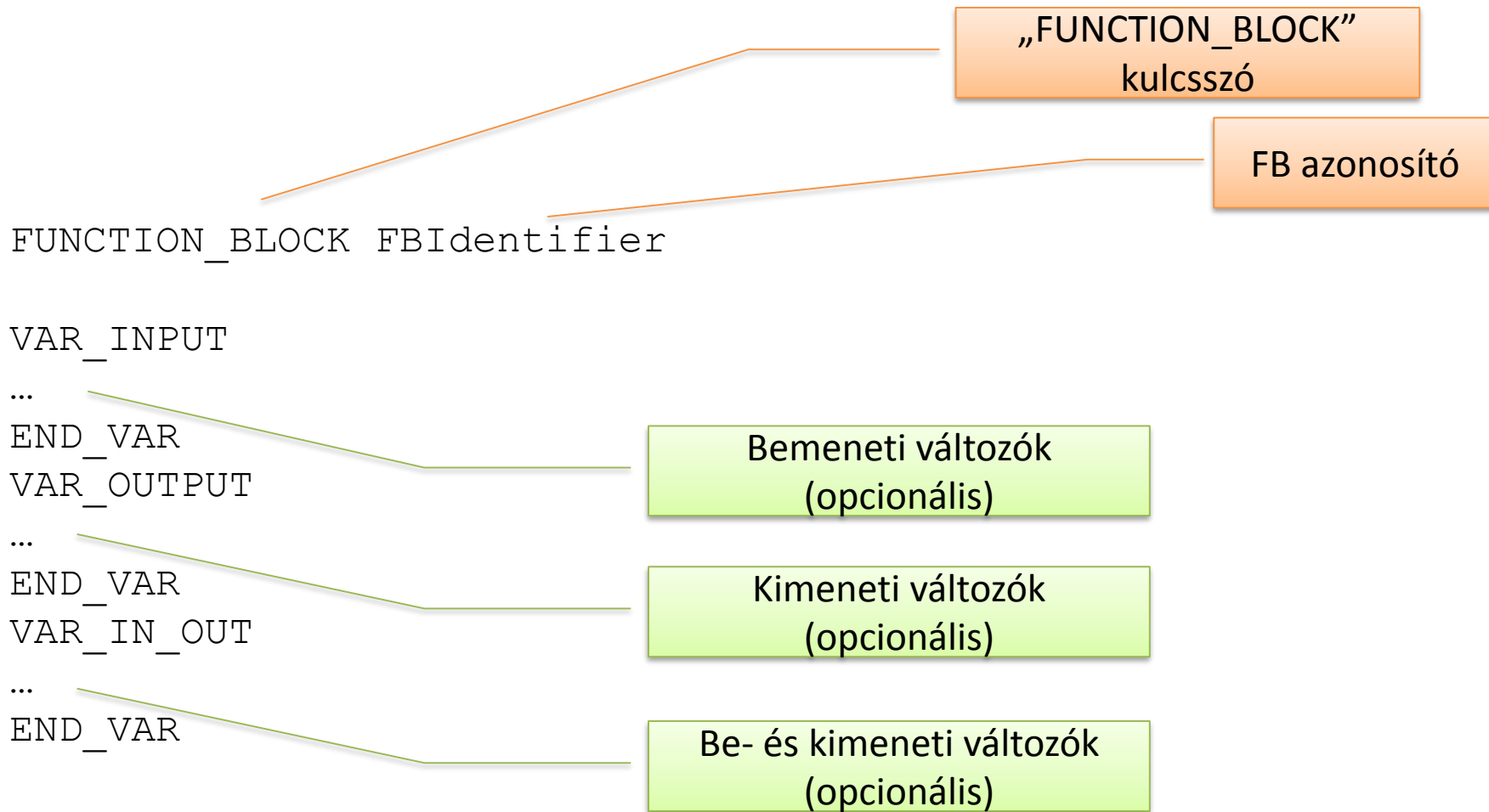
- Az FB-példány változói egy adatstruktúrát jelentenek
- Az FB-példány interfész változói a struktúra elemeiként elérhetők a hívó POU-ból
- Hivatkozás:

<FB_name>.<Var_name>

```
PROGRAM MyProg
VAR
    Timer1 : TON;
    MyBool : BOOL;
END_VAR

MyBool:=Timer1.Q;

END_PROGRAM
```



VAR

...

END_VAR

VAR_TEMP

...

END_VAR

VAR_EXTERNAL

...

END_VAR

(* CODE *)

END_FUNCTION_BLOCK

Retentív lokális
változók (opcionális)

Nem retentív lokális
változók (opcionális)

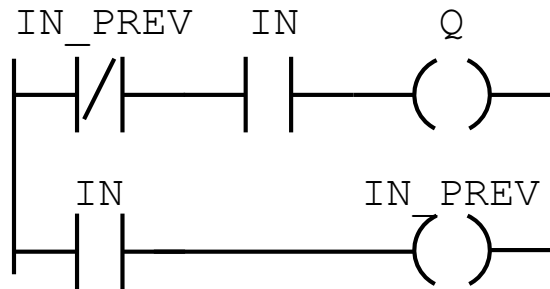
Kívül deklarált globális változók
(opcionális)

Kód

„END FUNCTION BLOCK”
kulcsszó

Példa: Éldetektálás

```
FUNCTION_BLOCK Rising
VAR_INPUT
    IN: BOOL;
END_VAR
VAR_OUTPUT
    Q: BOOL;
END_VAR
VAR
    IN_PREV: BOOL;
END_VAR
```



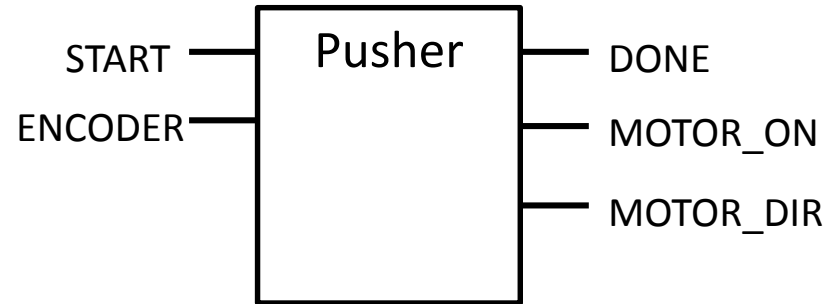
```
END_FUNCTION_BLOCK
```

Példa: Tologató

```
FUNCTION_BLOCK Pusher
VAR_INPUT
    ENCODER : BOOL;
    START : BOOL;
END_VAR
VAR_OUTPUT
    MOTOR_ON : BOOL;
    MOTOR_DIR : BOOL;
    READY : BOOL;
END_VAR
VAR
    POSITION : INT;
    Pos_Counter : CTUD;
END_VAR

(* CODE PART *)

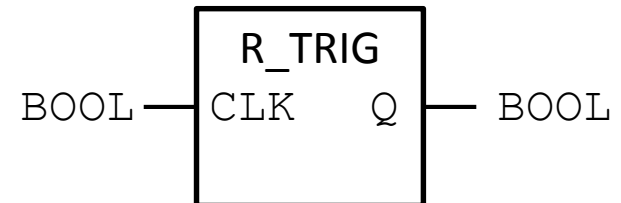
END_FUNCTION_BLOCK
```



Ha a START bemenet aktív, akkor indítsuk el a motort a pozitív irányba és számláljuk az enkóder impulzusait. Ha elértük a 100 értéket, akkor mozgassuk a tologatót negatív irányba a 0 érték eléréséig.

Standard funkcióblokkok

- Flip-flopok
 - SR – Set-domináns
 - RS – Reset-domináns
- Éldetektálás
 - R_TRIG: felfutó él (*Rising*)
 - F_TRIG: lefutó él (*Falling*)
- Számlálók
 - CTU, CTD, CTUD
- Időzítők
 - TON, TOF, TP

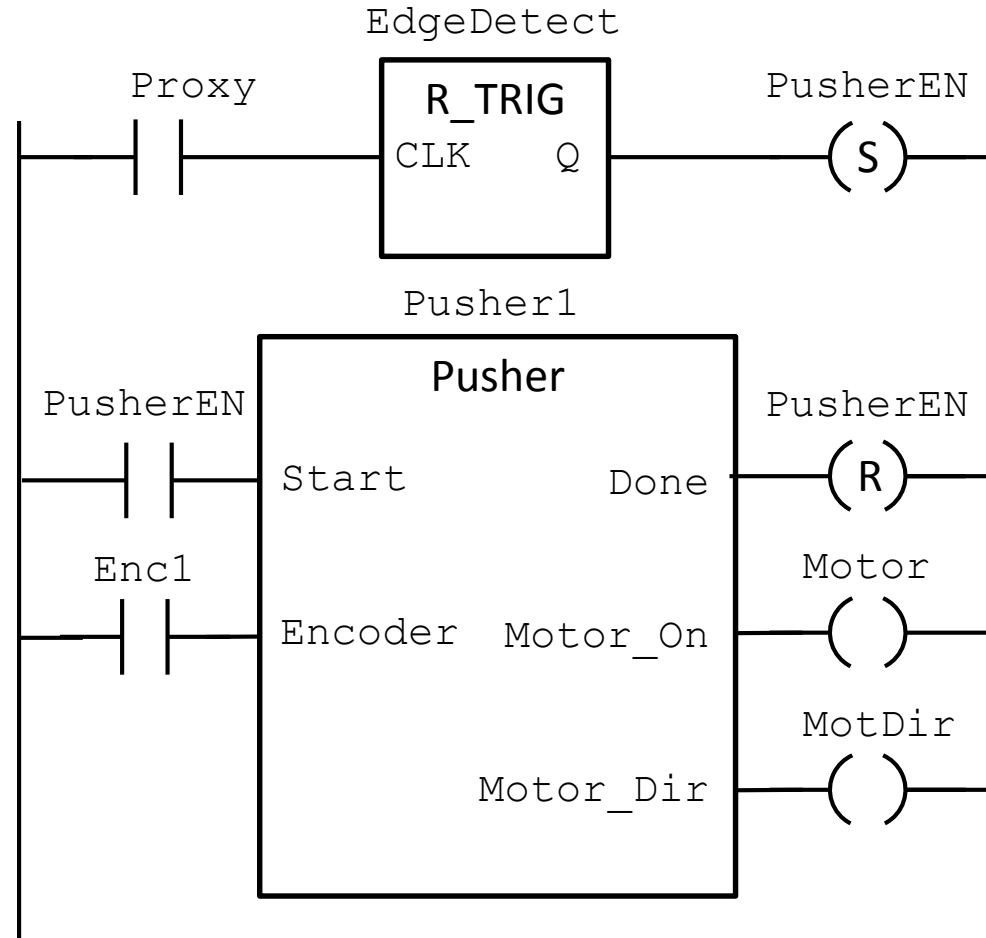


Programok

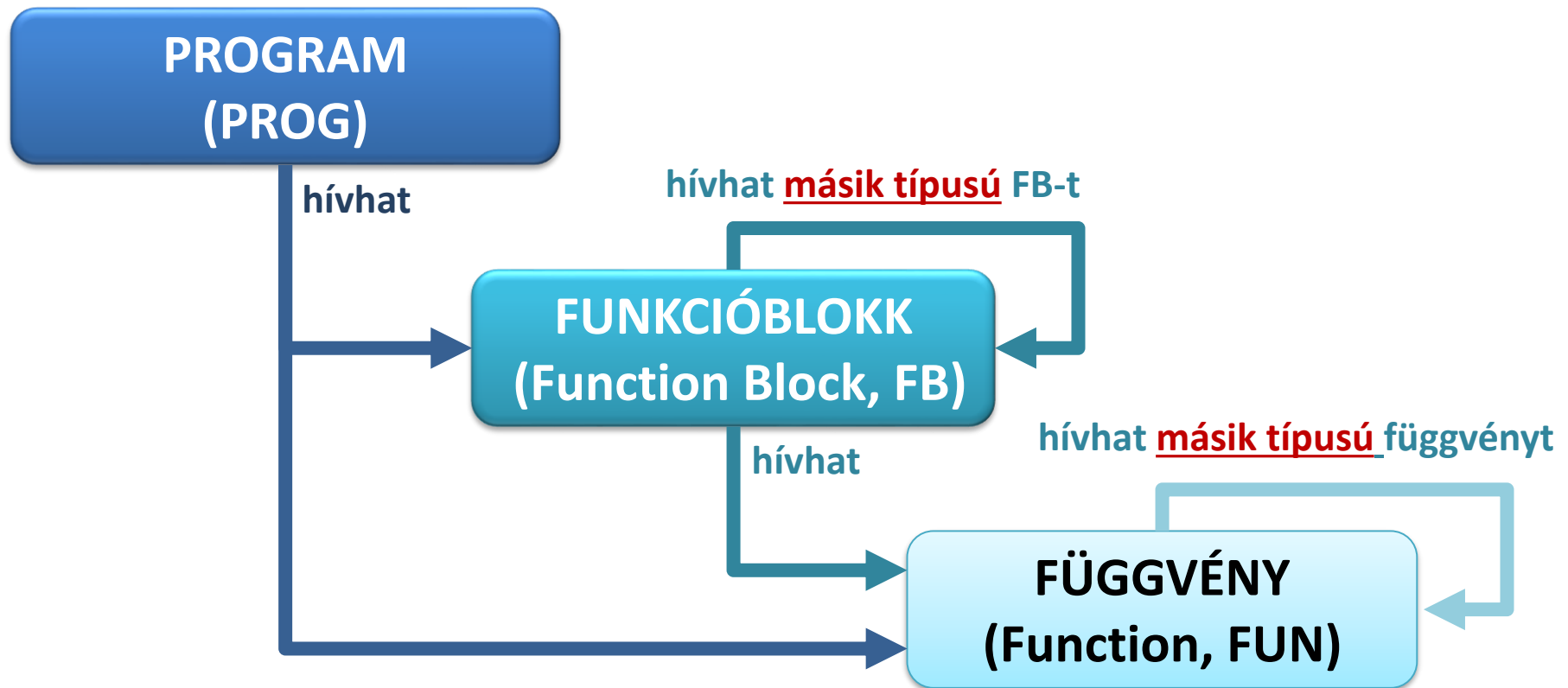
- „Főprogram”
- A funkcióblokkok képességein felül:
 - Közvetlen értékek (pl. bemeneti bitek) deklarálása
 - Globális változók deklarálása
 - Access path változók közvetlen használata
- Egy programot semmilyen más POU nem hívhat

Példa: tologató FB használata

```
PROGRAM PusherControl
VAR_INPUT
    Enc1      AT %I0.0: BOOL;
    Proxy     AT %I0.1: BOOL;
END_VAR
VAR_OUTPUT
    Motor     AT %Q0.0: BOOL;
    MotDir    AT %Q0.1: BOOL;
END_VAR
VAR
    EdgeDetect: R_TRiG;
    Pusher1:   Pusher;
    PusherEN:  BOOL;
END_VAR
```

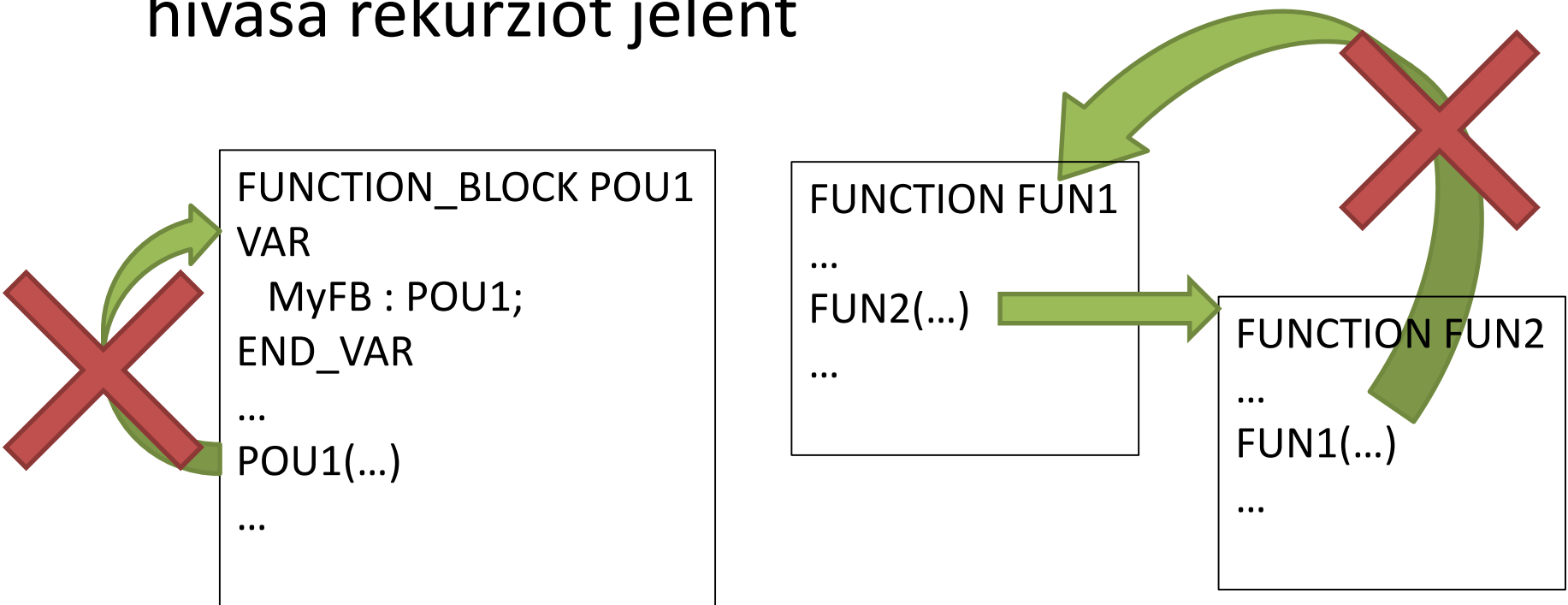


POU-k hívása



A rekurzió tiltott!

- A POUk önmagukat sem közvetlenül sem közvetve nem hívhatják
- Egy azonos típusú POU másik példányának hívása rekurziót jelent



Formális és informális hívás

```
FUNCTION EXP: REAL
VAR_INPUT
    Base : REAL;
    Exponent : REAL;
END_VAR

(* EXP := BaseExponent *)

END_FUNCTION
```

- Formális hívás:

```
EXP ( Base:=2,
      Exponent:=RealVar) ;
```

- Informális hívás:

```
EXP (2, RealVar) ;
```

- Függvények formális és informális módon is hívhatók (kivétel: LD és FBD)
- Funkcióblokkok és programok csak formális paraméterekkel hívhatók

Paraméterek sorrendje

- Formális hívás
 - A paraméterek tetszőleges sorrendben szerepelhetnek
 - Egyes paraméterek elhagyhatók (ebben az esetben előző vagy kezdeti értékük szerepel)
- Informális hívás
 - A paraméterek sorrendje meg kell egyezzen a deklaráció során használttal
 - Bemeneti paraméterek nem hagyhatók el

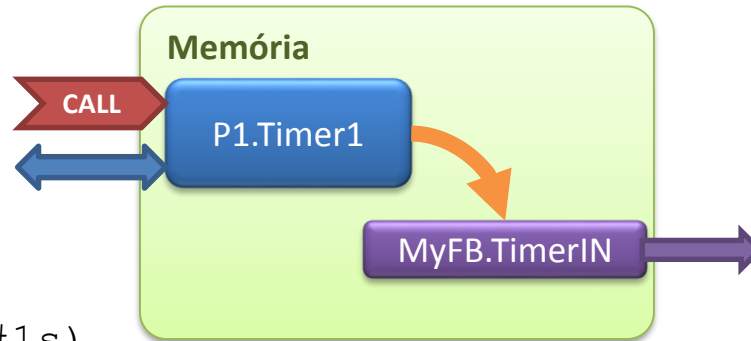
Funkcióblokk-példányok használata paraméterként

- Egy programnak vagy funkcióblokknak egy funkcióblokk-példányt is átadhatunk
 - Adatszerkezetként
 - Hívható „objektumként”
- Mikor hasznos?
 - Összetartozó adatok átadása
 - Közös használatú modul (pl. naplózás)

FB-példány átadása bemeneti paraméterként

```
PROGRAM MyProg  
VAR  
    Timer1: TON;  
    MyFB1: MyFB;  
END_VAR
```

```
Timer1.IN:=1;  
Timer1(IN:=1;PT:=T#1s)  
MyFB1(TimerIN:=Timer1);
```



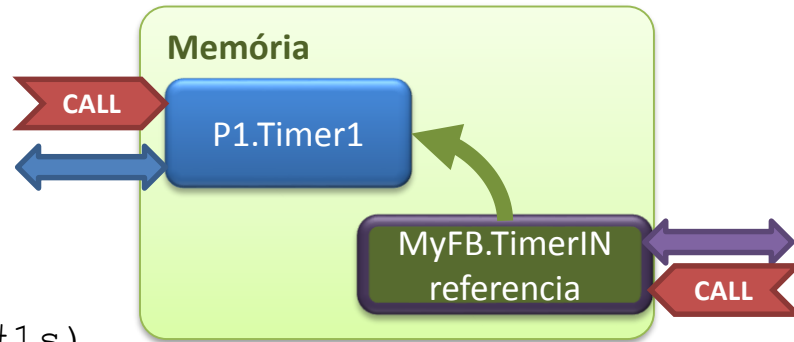
```
FUNCTION_BLOCK MyFB  
VAR_INPUT  
    TimerIN: TON;  
END_VAR  
VAR  
    Q: BOOL;  
    T: TIME;  
END_VAR  
  
A:=TimerIN.Q;  
T:=TimerIN.ET;
```

- Az FB-példány változóinak érték szerinti átadása
- A hívott POU csak olvashatja az átadott FB-példány be- és kimeneti változóit
- A hívott POU nem hívhatja az átadott FB-példányt

FB-példány átadása be- és kimeneti paraméterként

```
PROGRAM MyProg
VAR
    Timer1: TON;
    MyFB1: MyFB;
END_VAR
```

```
Timer1.IN:=1;
Timer1(IN:=1;PT:=T#1s)
MyFB1(TimerIN:=Timer1);
```



```
FUNCTION_BLOCK MyFB
VAR_IN_OUT
    TimerIN: TON;
END_VAR
VAR
    Q: BOOL;
    T: TIME;
END_VAR
```

```
TimerIN.PT:=T#9ms;
TimerIN(IN:=1);
```

- Az FB-példány referencia szerinti átadása – a referencia magára a teljes példányra vonatkozik!
- A hívott POU írhatja és olvashatja az átadott FB-példány be- és kimeneti változóit
- A hívott POU hívhatja az átadott FB-példányt

Programszervezési egységek (POU)

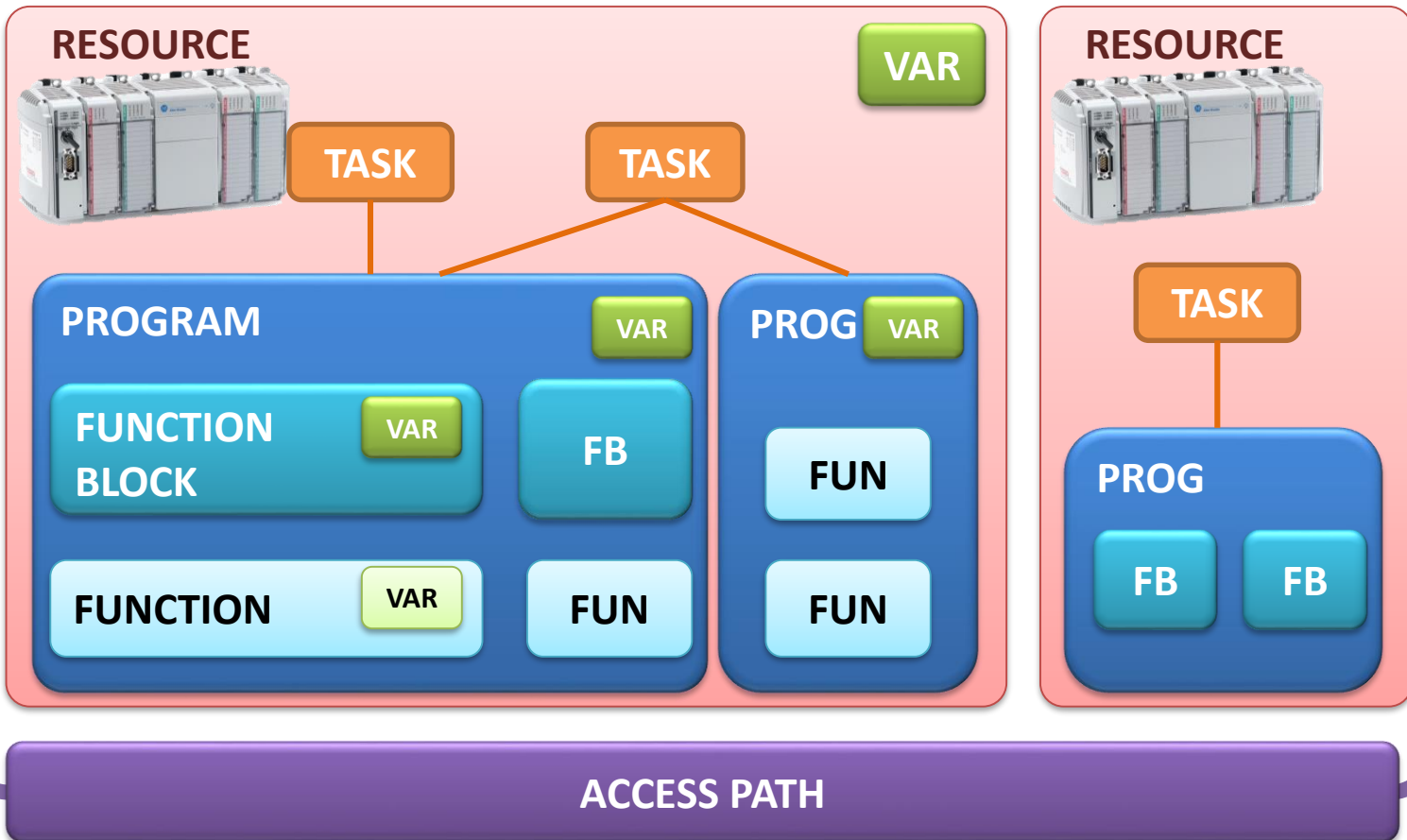
Jellemző	Függvény	Funkcióblokk	Program
FB példányok deklarációja	Nem	Igen	Igen
Függvényhívás	Igen	Igen	Igen
FB példány hívása	Nem	Igen	Igen
Program hívása	Nem	Nem	Nem
Overloading	Igen (standard fv)	Igen (standard FB)	Nem
Kiterjeszthetőség	Igen (standard fv)	Nem	Nem
Éldetektálás a bemeneten	Nem	Igen	Igen
EN/ENO (LD és FBD)	Igen	Igen	Nem
Rekurzív hívás	Nem	Nem	Nem

Programszervezési egységek (POU)

Jellemző	Függvény	Funkció-blokk	Program
Bemenő paraméterek (VAR_INPUT)	Igen	Igen	Igen
Kimenő paraméterek (VAR_OUTPUT)	Igen	Igen	Igen
Ki- és bemenő paraméterek (VAR_IN_OUT)	Igen	Igen	Igen
Visszatérési érték	Igen	Nem	Nem
Lokális változók (VAR)	Nem retentív	Retentív	Retentív
Ideiglenes lokális változók (VAR_TEMP)	Nem	Igen	Igen
Globális változók deklarálása (VAR_GLOB)	Nem	Nem	Igen
Globális változók használata (VAR_EXTERNAL)	Nem	Igen	Igen
Funkcióblokkok mint bemeneti paraméterek	Igen	Igen	Igen

Áttekintés

CONFIGURATION



Taszkok

- A programokat és FB-eket a taszkok rendelik az erőforrásokhoz (CPU-hoz)
- A programok és FB-példányok futását a taszkok vezérlik
 - Egy program vagy FB-példány több taszkhoz is tartozhat
 - Egy taszkhoz több program vagy FB-példány is tartozhat
- A taszkhoz rendeléssel a programból egy run-time objektum jön létre
- Ha a taszk ütemezésre kerül, egyszer lefuttatja a hozzá kapcsolt programokat és FB-eket

Taszk típusok

- Periodikus
 - Periodikusan fut (pl. 10 ms-onként)
- Eseményvezérelt
 - Egyszer fut le egy esemény hatására
 - Esemény: egy változó felfutó éle
- Program taszk-hozzárendelés nélkül
 - Ciklikus futás a legalacsonyabb prioritással
 - A többi taszk által fel nem használt CPU-időben fut

Taszkok prioritása

- Prioritási szintek (számuk implementációfüggő)
- 0: legmagasabb prioritás
- Több taszknak is lehet azonos prioritása

Taszk deklaráció

- TASK T (SINGLE, INTERVAL, PRIORITY)
- SINGLE (BOOL)
 - Logikai bemenet, felfutó éle jelenti az eseményt
 - Periodikus taszk esetén elhagyandó
- INTERVAL (TIME)
 - Periódusidő
 - Eseményvezérelt taszk esetén elhagyandó
- PRIORITY (INT)
 - Prioritás

Példa: Taszkok használata

```
TASK T_Periodic (INTERVAL:=t#10ms, PRIORITY:=10);
```

```
PROGRAM ConveyorControlInst WITH T_Periodic:  
    ConvControl (Start:=%I0.1, Fail=>GlobFail);
```

```
TASK T_Event (SINGLE:= AlarmIN, PRIORITY:= 0);
```

```
PROGRAM Emergency WITH T_Event :  
    Shutdown (TimeStamp=>LogEntry);
```

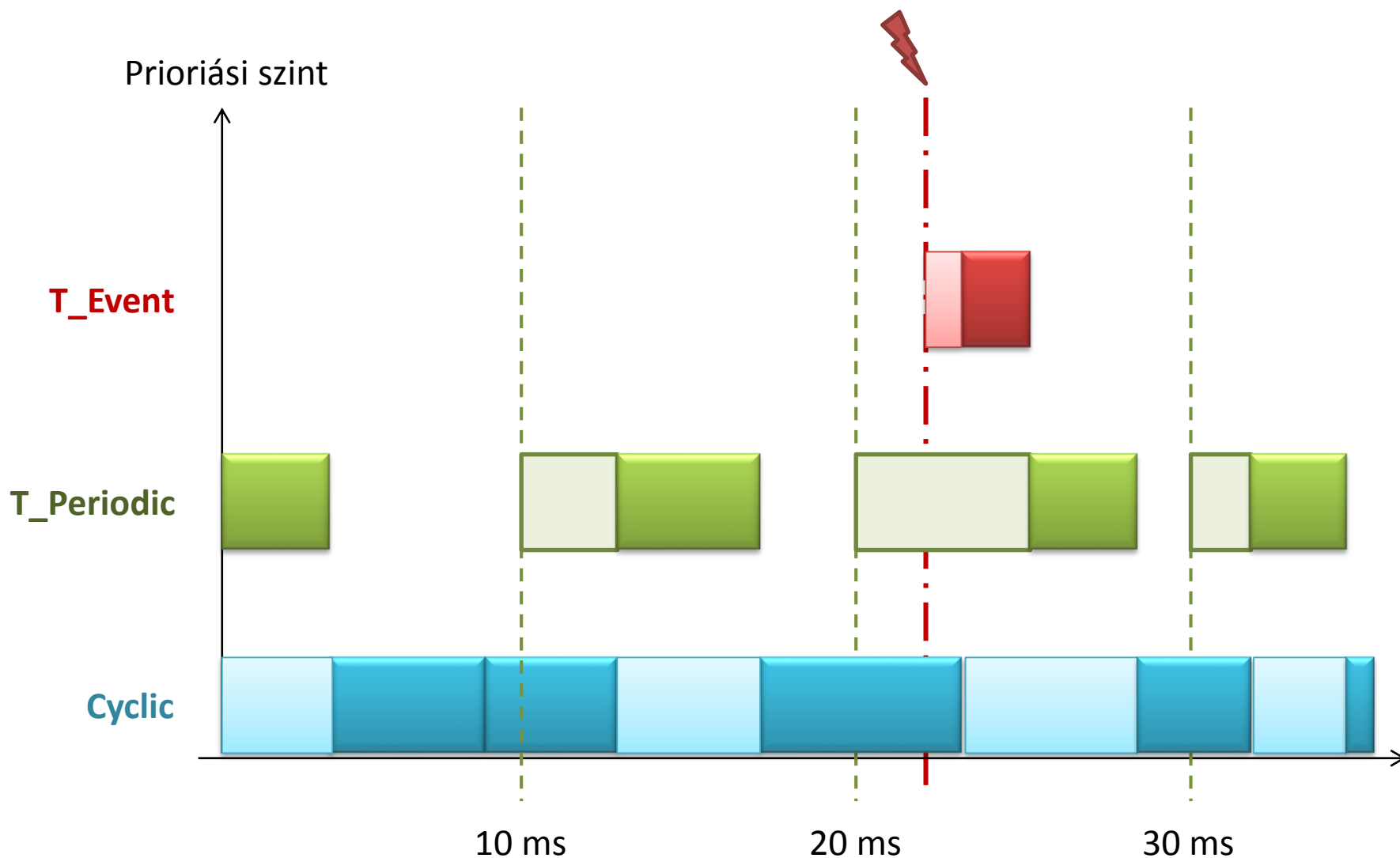
```
PROGRAM FailureLog WITH T_Event :  
    FailureLogger (In:=LogEntry);
```

```
PROGRAM Cyclic (In:=%I0.3);
```

Nem-preemptív ütemezés

- A taszkok nem szakíthatják meg egymás futását
- A futásra kész taszkokat sorba állítjuk
- Ha egy taszk befejezte a futást, a sorban álló legmagasabb prioritású taszk indul
- Azonos prioritású taszkokat a várakozás ideje szerint ütemezzük (kiékezés elkerülése)

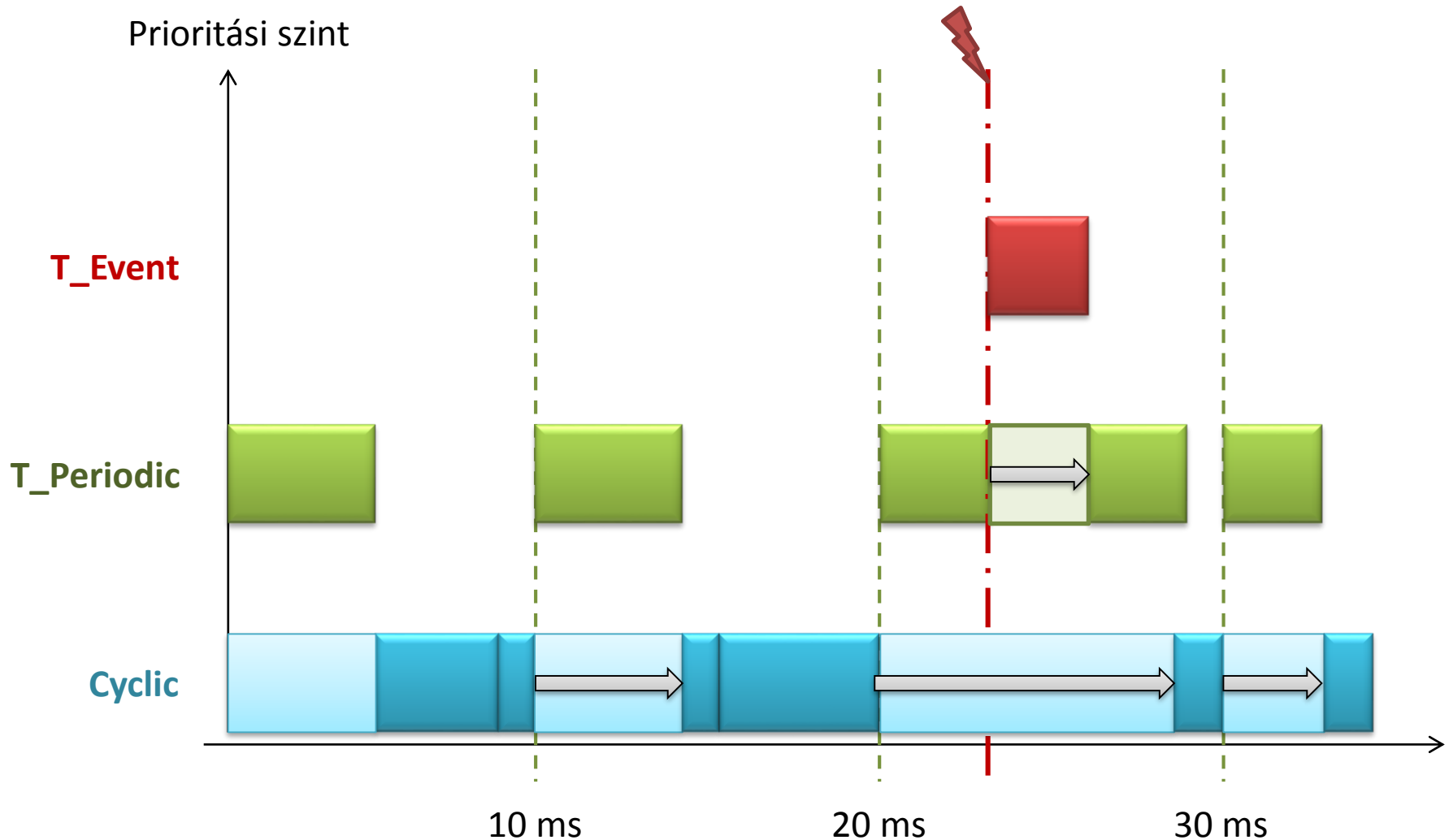
Nem-preemptív ütemezés



Preemptív ütemezés

- Egy magasabb prioritású taszk megszakíthatja egy alacsonyabb prioritású taszk futását
- A megszakított taszk futása felfüggesztődik, akkor indul újra, ha nincs nála magasabb prioritású taszk a sorban

Preemptív ütemezés



Ütemezési buktatók

- A magasabb prioritású taszkok megakadályozhatják az alacsonyabb prioritásúak futását
- A periodikus taszkok kihagyhatnak egy mintavételt
- Gondos tervezés szükséges

FB-példányok végrehajtása

- Az FB-példányokat programok deklarálják
- Egy FB-példány végrehajtását más taszkhoz is köthetjük, mint az őt deklaráló programét
- FB-példány végrehajtásának szabályai
 - Egy közvetlenül taszkhoz rendelt FB végrehajtása kizárólagosan a taszk feladata
 - Taszkhoz közvetlenül nem rendelt FB végrehajtása a szülő program végrehajtását követi

FB-példányok végrehajtása

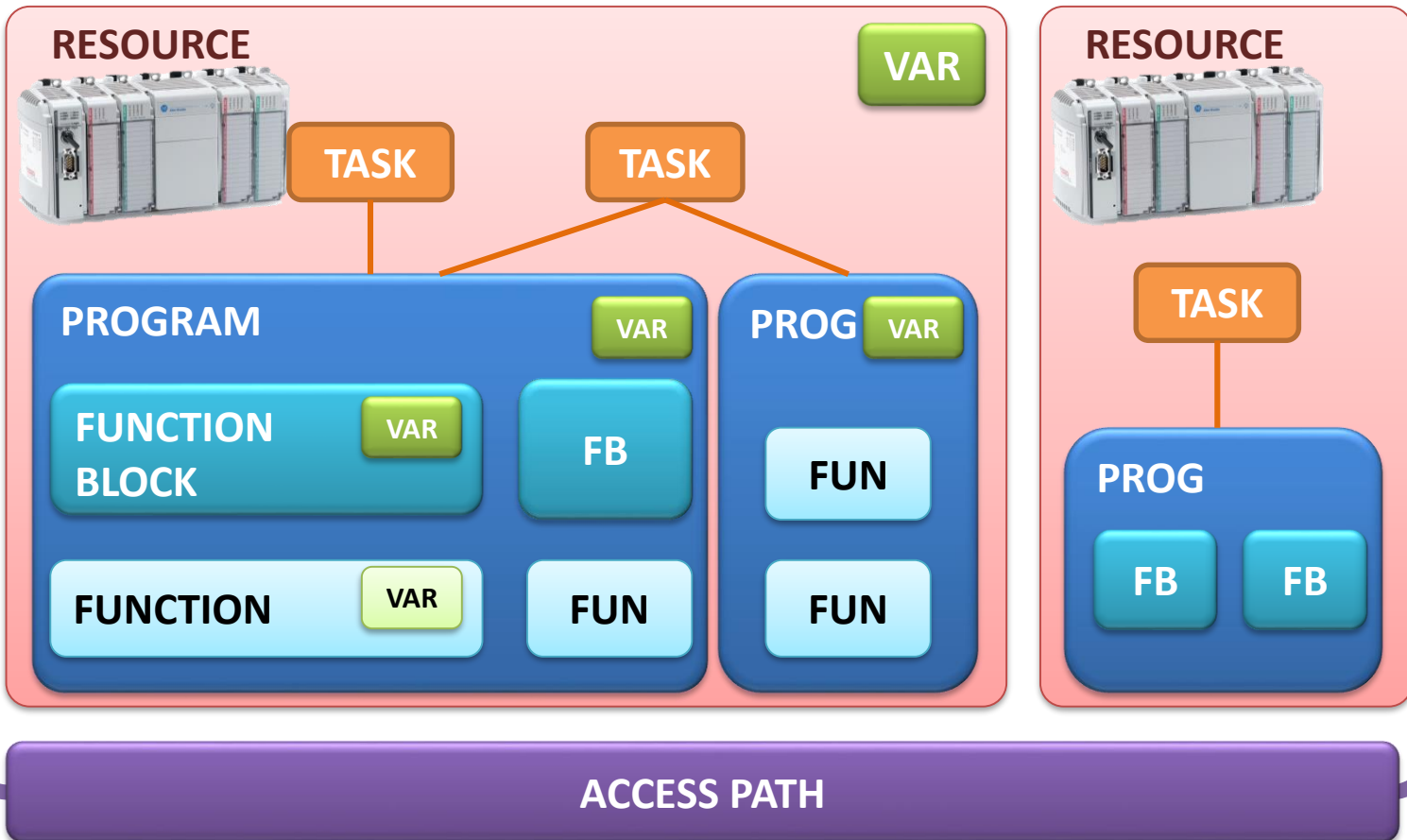
- A példányok csak akkor hajtódnak végre, ha
 - Taszkhoz kötötten ütemeződnek
 - A szülő program hívja őket



Ha deklarálunk egy funkcióblokkot, az nem fog automatikusan végrehajtódni minden ciklusban, csak akkor, ha a programban meghívjuk!

Áttekintés

CONFIGURATION



Erőforrások

- Az erőforrás (resource) egy CPU-t azonosít
- Erőforráshoz rendelhető objektumok:
 - Globális változók, amiket minden, az adott erőforráshoz rendelt POU elér
 - Taszkok
- Deklarálása nem kötelező az egyetlen erőforrást tartalmazó projekteken

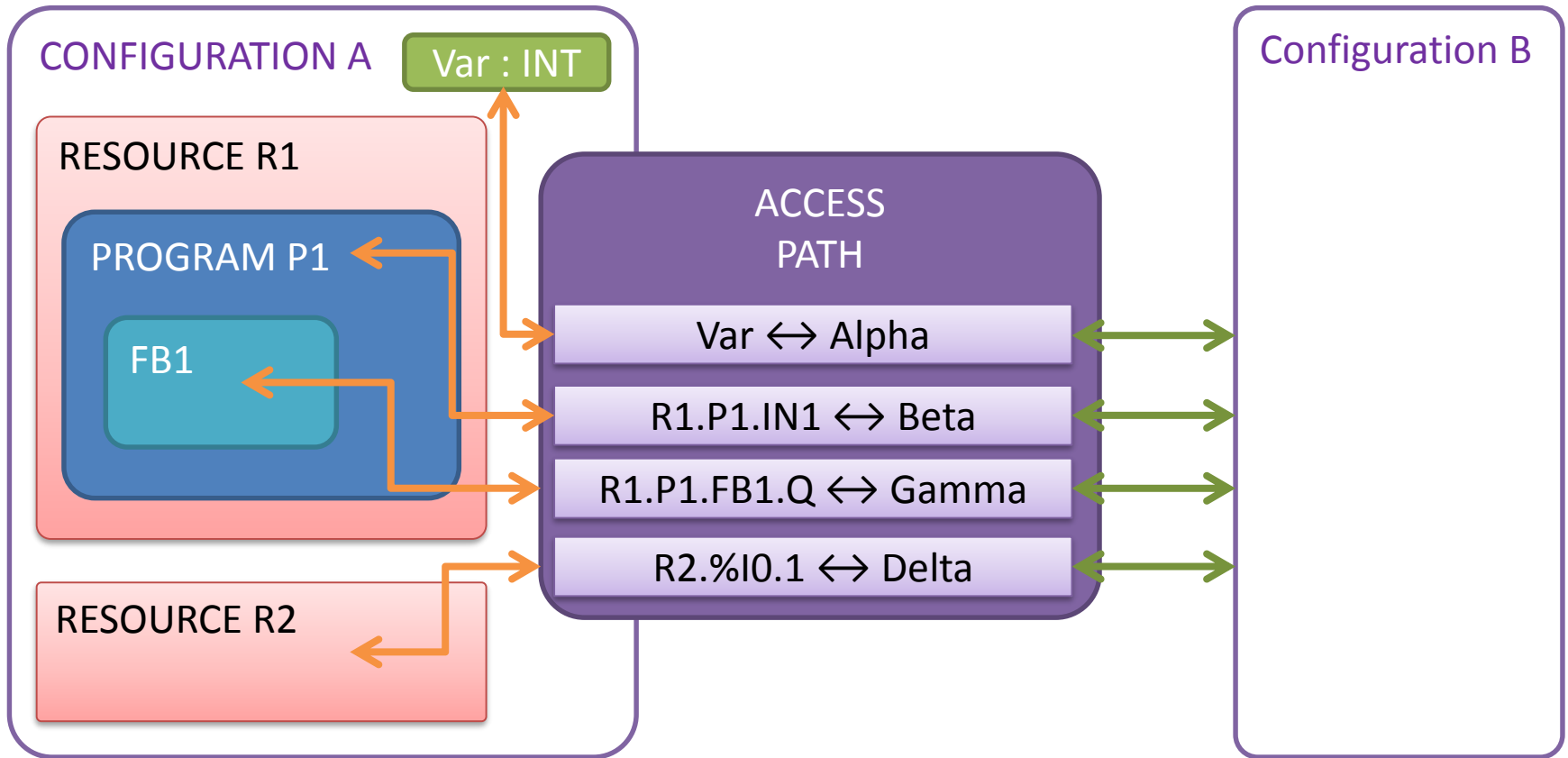
Példa: Erőforrás

```
RESOURCE ConveyorPLC ON CPU001  
VAR_GLOBAL  
    FailureLog : STRING;  
END_VAR  
TASK CyclicTask...  
TASK EventTask...  
END_RESOURCE
```

Konfiguráció

- Erőforrások összefogása
- Több erőforráson közösen használható globális változók deklarálása (VAR_GLOBAL)
- Példány-specifikus hozzárendelések és kezdeti értékek a változók számára (VAR_CONFIG)
- Access path definiálása (VAR_ACCESS)
 - Programok és FBk ki- és bemeneti valamint ki- és bemeneti változói
 - Globális változók
 - Közvetlen változók

Konfigurációk és access path változók



CONFIGURATION A

VAR_GLOBAL

Var : INT;

END_VAR

RESOURCE R1 ON CPU_Type1

PROGRAM P1 : ProgType(
FUNCTION_BLOCK FB1 : FBType1);

END_RESOURCE

RESOURCE R2 ON CPU_Type2

...

END_RESOURCE

VAR_ACCESS

Alpha : Var : INT READ_WRITE;

Beta : R1.P1.Out1 : BOOL READ_ONLY;

Gamma : R1.P1.FB1.In1 : BOOL;

Delta : R2.%I0.1 : BOOL READ_ONLY;

END_VAR

VAR_CONFIG

P1.Count : INT :=1;

P1.FB1.In2 AT R1.%I0.3 : BOOL;

END_VAR

END_CONFIGURATION

Adatok és változók

- A POU-k változótípusait már ismerjük
 - VAR_INPUT
 - VAR_OUTPUT
 - VAR_IN_OUT
 - VAR
 - VAR_TEMP
- Milyen adattípust vehetnek fel ezek?
- Pontosan hogyan deklaráljuk őket?
- Milyen tulajdonságaik lehetnek?

Adatok reprezentációja: literálisok

- Numerikus
 - Az _ (alulvonás) nem feldolgozott elválasztóként használható
 - Egész
 - Egész számok leírása
 - Bináris értékek leírása
 - Valós
- Karakterlánc
- Idő és dátum

Egész literálisok

- Decimális: előjellel vagy előjel nélkül
 - 123_321
 - -3
- Számrendszer megadása – csak előjel nélkül
 - 2#0000_1011
 - 8#13
 - 16#0B
- Bináris értékek
 - TRUE / FALSE
 - 1 / 0
- Bit string: decimális vagy számrendszerrel adott egész előjel nélkül

Valós literálisok

- Decimális formátum előjellel vagy előjel nélkül:
 - -12.0
 - 0.21
- Exponenciális formátum előjellel vagy előjel nélkül:
 - $1.34\text{E}-3 : 1.34 \cdot 10^{-3}$
 - $-1.0\text{E}3 : -1000$

Karakterláncok

- Egybájtos karakterláncok
 - `'Robin Hood'`
 - Különleges karakterek: `$` prefix (pl. `$'`, `$R`, `$$`)
 - Karakterkódok: `$<hex>`, ahol `<hex>` a karakter kétjegyű hexa-kódja
- Kétbájtos karakterláncok
 - `"Rózsa Sándor"`
 - Különleges karakterek: `$` prefix (pl. `$"`, `$R`, `$$`)
 - Karakterkódok: `$<hex>`, ahol `<hex>` a karakter négyjegyű hexa-kódja

Időtartam

- Formátum:
`<T | TIME>#<duration>`
- `duration`: értékek és egységek kombinációja csökkenő sorrendben
- Az időtartam-értékek egészek, kivéve az utolsó (valós)
- Az időtartam negatív is lehet
- A túlcsordulás megengedett, pl.
`t#25h = t#1d1h`

Szimbólum	Egység
d	Nap (day)
h	Óra (hour)
m	Perc (minute)
s	Másodperc (second)
ms	ms

`t#2d4h32m4s1.3ms`

`TIME#-92s`

Idő és dátum

- Date

<DATE | D>#<ÉÉÉÉ>-<HH>-<NN>

D#1986-02-11

- Time of day

<TIME_OF_DAY | TOD>#<óó>:<pp>:<mm.m>

TOD#19:07:21.6

- Date and Time

<DATE_AND_TIME | DT>#<ÉÉÉÉ>-<HH>-<NN>-
<óó>:<pp>:<mm.m>

DT#1986-02-11-19:07:21.6

Adattípusok

- Elemi adattípusok
 - Logikai
 - Numerikus
 - Idő, dátum és időtartam
 - Karakterlánc
- Származtatott adattípusok
 - Felsorolás
 - Tartomány
 - Tömb
 - Struktúra
- Általános adattípus (ANY)

Bináris adattípusok

Adattípus	Leírás	Bit	Értékkészlet	Kezdeti érték
BOOL	Logikai	1	[0,1]	0
BYTE	Bit string	8	[0,...,16#FF]	0
WORD	Bit string	16	[0,...,16#FFFF]	0
DWORD	Bit string	32	[0,...,16#FFFF FFFF]	0
LWORD	Bit string	64	[0,...,16#FFFF FFFF FFFF FFFF]	0

Egy bit string adott bitjének indexelése: `BIT_STRING.BIT_NUMBER`

Numerikus adattípusok

Adattípus	Leírás	Bit	Értékkészlet	Kezd. érték
SINT	Short integer	8	$[-128, \dots, +127]$	0
INT	Integer	16	$[-32768, \dots, 32767]$	0
DINT	Double integer	32	$[-2^{31}, \dots, +2^{31} - 1]$	0
LINT	Long integer	64	$[-2^{63}, \dots, +2^{63} - 1]$	0
USINT	Unsigned short integer	8	$[0, \dots, 255]$	0
UINT	Unsigned integer	16	$[0, \dots, 65535]$	0
UDINT	Unsigned double integer	32	$[0, \dots, +2^{32} - 1]$	0
ULINT	Unsigned long integer	64	$[0, \dots, +2^{64} - 1]$	0
REAL	Single prec. floating point	32	IEEE 754 szabvány szerint	0.0
LREAL	Double prec. floating point	64		0.0

Dátum, idő és időtartam

Adattípus	Értelmezés	Kezdeti érték
DATE (D)	Dátum: ÉÉÉÉ-HH-NN	d#0001-01-01
TIME_OF_DAY (TOD)	Időpont: óó:pp:mm.m	tod#00:00:00
DATE_AND_TIME (DT)	Dátum és idő: ÉÉÉÉ-HH-NN- óó:pp:mm.m	dt#0001-01-01 00:00:00
TIME (T)	Időtartam: nap, óra, perc, másodperc, ms	t#0s

Karakterláncok

Adattípus	Leírás	Kezdeti érték
STRING	Változó hosszúságú karakterlánc 1 bájtos karakterekkel	" (üres)
WSTRING	Változó hosszúságú karakterlánc 2 bájtos karakterekkel	"" (üres)

Általános adattípusok

ANY

ANY_ELEMENTARY

ANY_DERIVED

ANY_MAGNITUDE

ANY_STRING

STRING,
WSTRING

ANY_DATE

DATE, DT, TOD

ANY_NUM

ANY_BIT

BOOL,
BYTE,
WORD,
DWORD,
LWORD

TIME

ANY_REAL

REAL, LREAL

ANY_INT

SINT,
INT,
DINT,
LINT,
USINT,
UINT,
UDINT,
ULINT

Típusos literálisok

- Ha nem explicite meghatározott, akkor a numerikus és szöveges literálisok típusa automatikusan kerül hozzárendelésre
- Típus megadása prefix-szel:
 - `INT#5`
 - `BOOL#0`
 - `WSTRING#'HELLO WORLD!'`

Származtatott adattípusok

- Típusdefiníció:
 - `TYPE...END_TYPE`
 - A típusdefiníciók az egész projekten belül érvényesek
- Származtatott adattípusok (explicit típusdefiníció nélkül is deklarálhatók)
 - Közvetlenül származtatott
 - Felsorolás
 - Tartomány
 - Tömb
 - Struktúra

Közvetlenül származtatott adattípusok

- Elemi adattípusok „újradefiniálása”
 - Más típusazonosító
 - Esetlegesen más kezdeti érték (csak az elemi típusnak megfelelő)

```
TYPE
```

```
    QINT          : LINT;
```

```
    TrueBool      : BOOL := TRUE;
```

```
END_TYPE
```

Felsorolás

- Egész adattípuson alapul
- Számértékek helyett szöveges konstansok használata
- 0-ról induló számozás

```
TYPE
```

```
    TRAFFIC_LIGHT: (RED, YELLOW, GREEN) ;
```

```
END_TYPE
```

Példa: felsorolás

```
TYPE
    TRAFFIC_LIGHT: (RED, YELLOW, GREEN);
END_TYPE
VAR
    Light_NW : TRAFFIC_LIGHT;
    Light_SE : (RED, YELLOW, GREEN);
END_VAR

...
Light_NW := RED;
Light_SE := GREEN;

...
```

Tartomány

- Bármilyen ANY_INT vagy abból származtatott típusra
- A megengedett értékkészlet korlátozott:
(Lower_Limit..Higher_Limit)
- A tartományon kívüli értékek esetén hibajelzés

```
TYPE
```

```
    Val_12_bit_ADC: (0..4095)
```

```
END_TYPE
```

```
VAR
```

```
    ADC_Val: Val_12_bit_ADC;
```

```
END_VAR
```

Tömb

- Bármilyen elemi vagy származtatott típusból (a szabvány 3. verziójától akár FB-példányok is)
- Többdimenziós tömbök támogatottak
- Az indexhatárok tetszőleges INT típusú adatok lehetnek (akár negatívak is!)

Példa: Tömbök használata

```
TYPE
```

```
    Matrix: ARRAY [1..16,1..16] OF REAL;
```

```
END_TYPE
```

```
VAR
```

```
    A      : Matrix;
```

```
    b      : ARRAY [-4..9] OF INT;
```

```
END_VAR
```

```
b[2] := 4;
```

```
M[1, 4] := 4.231;
```

Struktúra

- Hierarchikusan felépített adatszerkezet
- Elemei elemi vagy származtatott típusúak is lehetnek

```
TYPE
```

```
    Measurement:
```

```
    STRUCT
```

```
        TimeStamp: DATE_AND_TIME;
```

```
        Data: INT;
```

```
    END_STRUCT
```

```
    MeasurementLog:
```

```
    STRUCT
```

```
        InstrumentID: STRING;
```

```
        Log: ARRAY[1..100] OF Measurement;
```

```
    END_STRUCT
```

```
END_TYPE
```

Változók reprezentációja

- Szimbolikus reprezentáció
 - Szimbolikus névvel
 - Fizikai objektumhoz köthető az AT kulcsszóval – egyéb esetben automatikusan egy memóriaterülethez kötött
 - Bármilyen típusú POU-ban használható
- Közvetlen változók (directly represented variable)
 - Fizikai objektumokhoz közvetlenül kötött változó
 - Nem deklarálható függvényben és FB-ben

Közvetlen változók

- Formátum: %LocationPrefixSizePrefixNumber

Location		Size	
Prefix	Értelmezés	Prefix	Értelmezés
I	Bemenet	X vagy hiányzik	Bit
Q	Kimenet	B	Bájt méret (8 bit)
M	Memória	W	Szó méret (16 bit)
		D	Duplaszó méret (32 bit)
		L	Long word méret (64 bit)

Number: Előjel nélküli egész vagy hierarchikus location esetén . (pont) karakterrel elválasztott előjel nélküli egészek

Közvetlen változók - példák

Változó	Értelmezés	Alapértelmezett adattípus
%QX12 vagy %Q12	Egyetlen kimeneti modullal rendelkező kompakt PLC 12-es sorszámú kimenete	BOOL
%IX0.2 vagy %I0.2	A 0. modul 2. sorszámú bit-kimenete (geografikus cím)	BOOL
%IB2.0	A 2. sorszámú bemeneti modul első 8 bites csatornája	WORD
%MX14 vagy %M14	14. sorszámú memóriabit: rögzített memóriahely	BOOL
%MD23	23. sorszámú memóriaszó: rögzített memóriahely	DWORD

Közvetlen és szimbolikus változók deklarálása

VAR

AT %Q0.1 : BOOL;

AT %MW12 : SINT;

Switch1 AT %I0.2 : BOOL;

Motor3 AT %Q1.3 : BOOL;

END_VAR

...

%Q0.1 := TRUE;

Motor3 := 1;

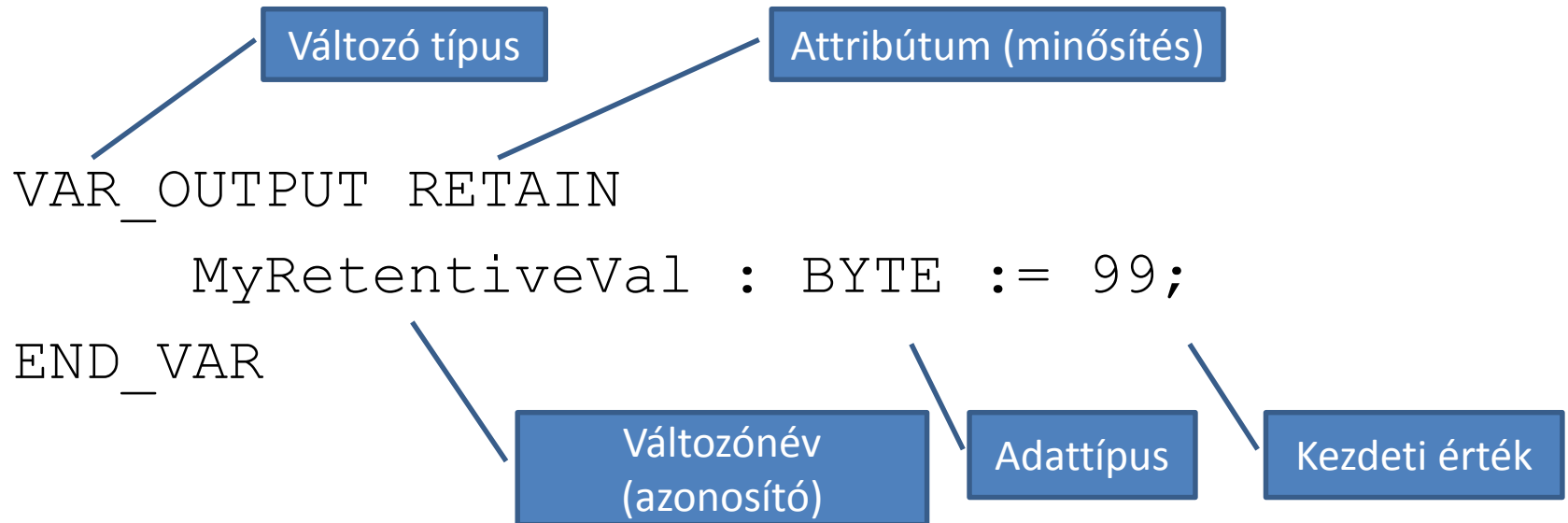
Bit kimenet közvetlen változója

Alapértelmezettől különböző típusú
közvetlen változó

Fizikai objektumhoz kötött szimbolikus
változó

A hozzárendelt érték az 1. kimeneti
modul 3. csatornáján jelenik meg

Változók deklarálása



Változótípus attribútumok és minősítések

- Az adott típusként (pl. VAR_INPUT) deklarált változók mindegyikére érvényesek
- RETAIN: retentív, értékét tápkimaradás esetén megőrzi (elemes táplálású RAM-ban tárolt)
- NON-RETAIN: nem retentív
- CONSTANT: konstans, nem módosítható

Változó attribútumok és minősítések

- Egyedileg adhatók meg a változókra
- Csak egy használható közülük
- Éldetektálás
 - R_EDGE
 - F_EDGE
 - Csak FBk és Programok VAR_INPUT típusú változóira
- Védelem VAR_ACCESS típusú változók számára
 - READ_ONLY
 - READ_WRITE

Attribútumok és minősítések

Változótípus	RETAIN / NON-RETAIN	CONSTANT	R_EDGE / F_EDGE	READ_ONLY / READ_WRITE
VAR	+	+	-	-
VAR_TEMP	-	+	-	-
VAR_INPUT	+	-	+	-
VAR_OUTPUT	+	-	-	-
VAR_IN_OUT	-	-	-	-
VAR_GLOBAL	+	+	-	-
VAR_EXTERNAL	-	+	-	-
VAR_ACCESS	-	-	-	+

Attribútumok és minősítések - példa

VAR_INPUT

Élérzékeny bemenet: akkor értékelődik ki 1-re, ha előző értéke 0, aktuális értéke pedig 1

RisingEdge : BOOL R_EDGE;

END_VAR

VAR_OUTPUT RETAIN

A kimeneti változók retentívek: értékük tápkimaradás esetén sem veszik el

IntOut : BOOL;

END_VAR

VAR CONSTANT

Konstans: a ConstNum változó 0xC4h értéke nem írható felül

ConstNum : INT := 16#C4;

END_VAR

Kezdeti érték

- Minden változónak van kezdeti értéke
- A kezdeti érték
 - Megadható deklarációról (egyedileg a változókra)
 - Örökölt az adattípusból
- A PLC indításakor a változók a következők szerint kapnak értéket (fentről lefelé haladva)

Feltétel	Melegindítás	Hidegindítás
Retentívként deklarált változó (RETAIN)	Telepes RAM-ban tárolt érték	
Nem retentív, de a deklarációban szerepel kezdeti érték	Deklarált kezdeti érték	Deklarált kezdeti érték
Egyik sem	Adattípusból származtatott kezdeti érték	Adattípusból származtatott kezdeti érték

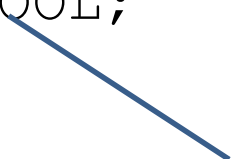
Kezdeti érték megadása deklarációban

- Kivételek
 - VAR_IN_OUT típusnak nincs kezdeti értéke
 - A globális változóknak ott adható kezdeti érték, ahol VAR_GLOBAL kulcsszóval deklaráljuk őket
- A kezdeti értéknek meg kell felelnie az adattípusnak
- Tömböknek és struktúráknak is adható kezdeti érték

VAR

```
MyIntNum : INT := 13;  
BePositive : BOOL := TRUE;  
BeNegative : BOOL;
```

END_VAR



A kezdeti érték az adattípus alapján öröklődik (FALSE)

FB-példány alapértelmezett paramétereinek megadása

- Az FB-példányok alapértelmezett paramétereit is megadhatók a deklarációkor
- Ezután hiányos paraméterekkel történő hívás esetén az FB az alapértelmezett paramétert kapja

VAR

Timer: TON (IN:=%IX0.1, PT:=T#2s)

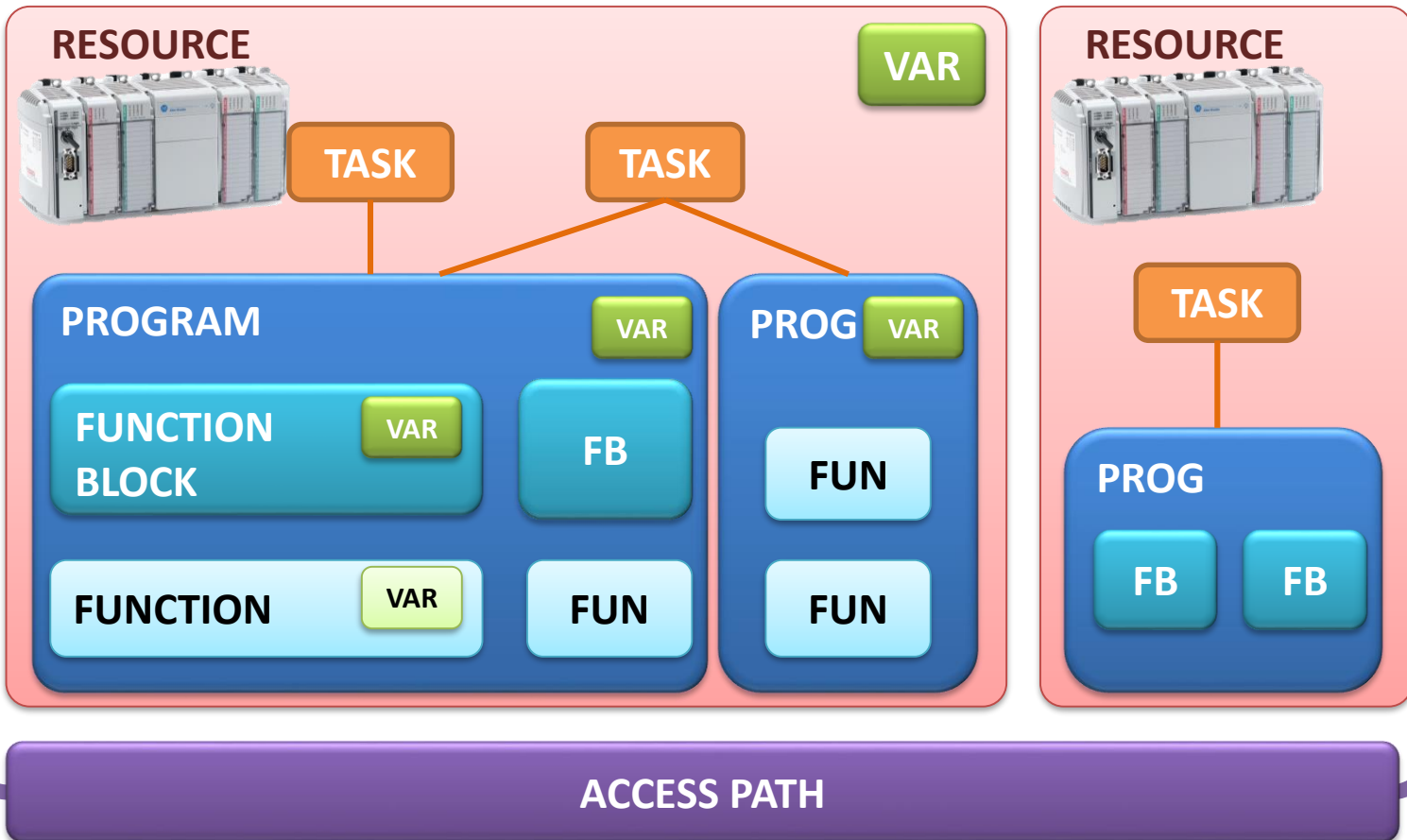
END_VAR

Timer();

= Timer(IN:=%IX0.1, PT:=T#2s);

Áttekintés

CONFIGURATION



Példa

Futószalag

- Motor KI/BE

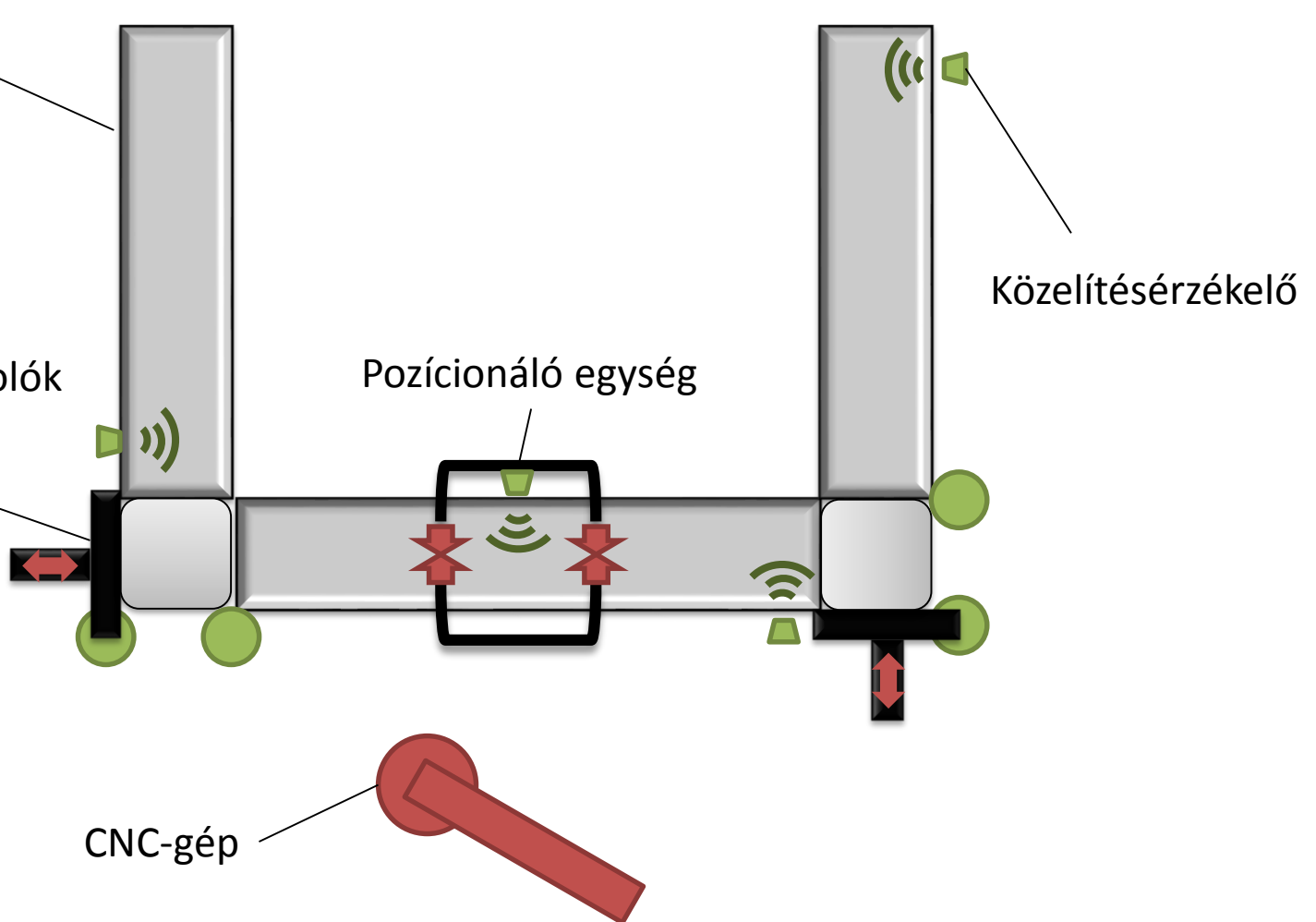
Tologató

- Motor KI/BE
- Motor irány
- Végálláskapcsolók
elől és hátul

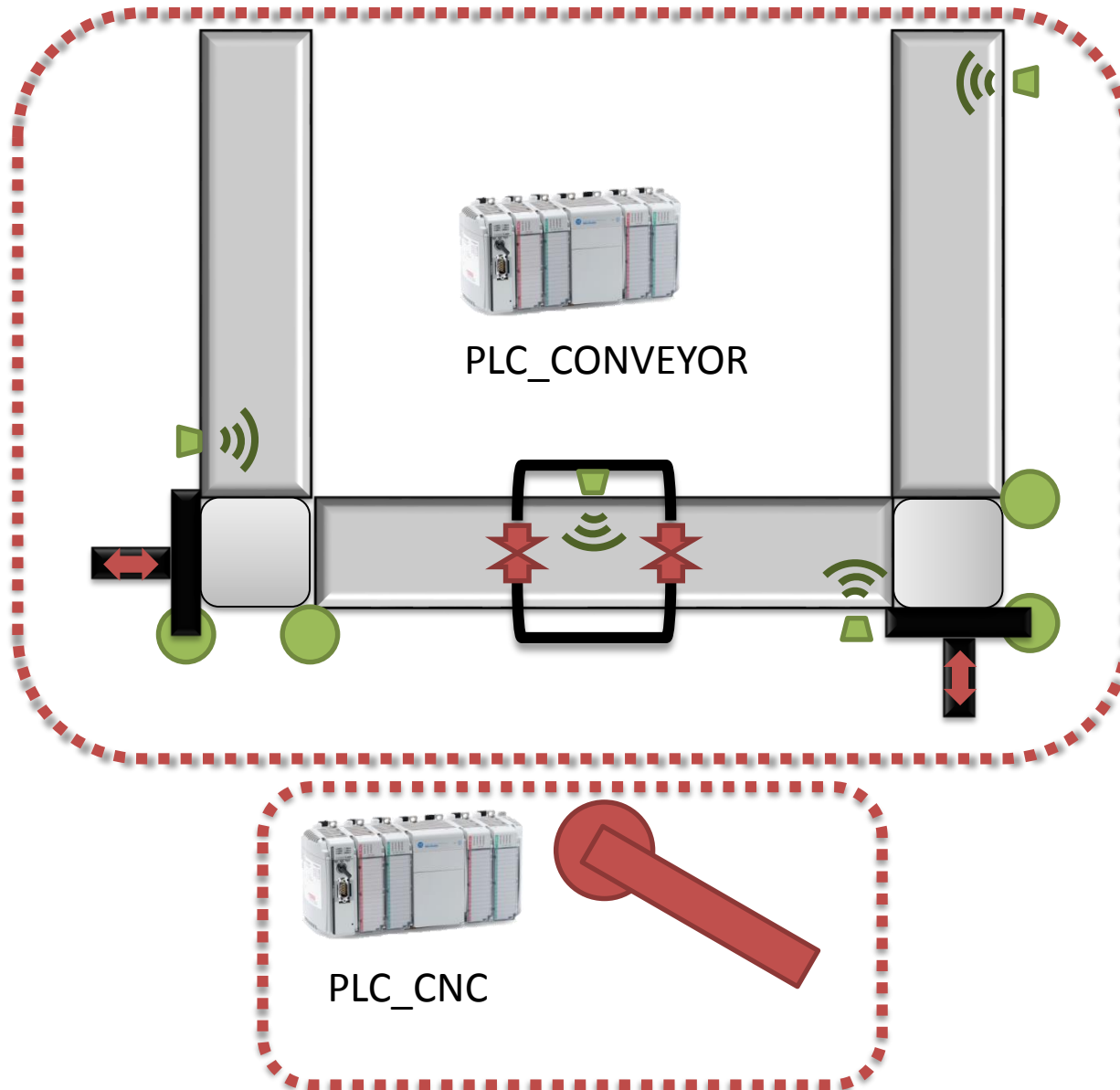
Pozícionáló egység

Közelítésérzékelő

CNC-gép



Példa



Példa – Erőforrások és taszkok

- Erőforrások (resource)
 - PLC_CONVEYOR
 - PLC_CNC
- Taszkok (task)
 - CNC gép irányítása: periodikus taszk a PLC_CNC-n
 - Futószalag-rendszer vezérlése: ciklikus taszk a PLC_CONVEYOR-on
 - Pozícionáló eszköz irányítása: eseményvezérelt taszk a PLC_CONVEYOR-on

Példa - POUk

- A CNC-gépet irányító program: ProgCNC
- Pozícionálást végző program: ProgPosControl
- A futószalag-rendszert vezérlő program:
ProgConv
 - FB a tologató irányítására: FBPusher
 - FB a futószalagok irányítására: FBConv

A tologató FB-ja

```
FUNCTION_BLOCK FBPusher  
VAR_INPUT
```

```
    Start:          BOOL    R_EDGE;
```

```
    LimitF:         BOOL;
```

```
    LimitB:         BOOL;
```

```
END_VAR
```

```
VAR_OUTPUT RETAIN
```

```
    Motor:          BOOL:=FALSE;
```

```
    Dir:            BOOL:=FALSE;
```

```
    Done:           BOOL:=FALSE;
```

```
END_VAR
```

```
(* Törzs: programkód *)
```

```
END_FUNCTION_BLOCK
```

A tologató a Start bemenet
felfutó élére indul

Végálláskapcsolók

Tápkimaradásból
visszatérve a tologató
mozog tovább

A hátsó véghelyzetbe
visszatérve a Done
kimenetet 1-be állítjuk

(*...*) : comment (ST nyelv)

A futószalag FB-ja

```
FUNCTION_BLOCK FBConv
```

```
VAR_INPUT
```

```
    Proxy          : BOOL;
```

```
END_VAR
```

```
VAR_OUTPUT
```

```
    ConvRun        : BOOL;
```

```
END_VAR
```

```
VAR RETAIN
```

```
    State          : (Idle,Running);
```

```
END_VAR
```

```
(* Törzs: programkód *)
```

```
END_FUNCTION_BLOCK
```

Közelítésérzékelő a szalag végénél

Szalag áll / mozog

Felsorolásként definiált állapotváltozó (értéke INT típusú)

A futószalag-rendszer programja

```
PROGRAM ProgConv
VAR_INPUT
    ProxySensor                : BOOL;
END_VAR
VAR_OUTPUT
    ConvMotor1                 : BOOL;
    AtPosition                  : BOOL := FALSE;
    PusherDone1                 : BOOL;
END_VAR
VAR_EXTERNAL
    PartCounter                 : UINT;
END_VAR
VAR
    ConvMotor2 AT %Q0.2         : BOOL;
    ConvMotor3 AT %Q0.3         : BOOL;
    ProxySensor2 AT %I0.2       : BOOL;
    ProxySensor3 AT %I0.3       : BOOL;
    Conv1                      : FBConv
        (Proxy:=ProxySensor1, ConvRun=>ConvMotor1);
    Conv2                      : FBConv
        (Proxy:=ProxySensor2, ConvRun=>ConvMotor2);
    Conv3                      : FBConv
        (Proxy:=ProxySensor3, ConvRun=>ConvMotor3);
    Pusher1                    : FBPusher;
    Pusher2                    : FBPusher;
END_VAR
```

Ezeket a változókat az erőforrásnál rendeljük fizikai objektumokhoz

Globális változó használata

Fizikai címek hozzárendelése a változókhoz

FB-példányok létrehozása az alrendszerekhez

A futószalag-rendszer programja

```
(* folytatás *)
```

```
(* Törzs: programkód *)
```



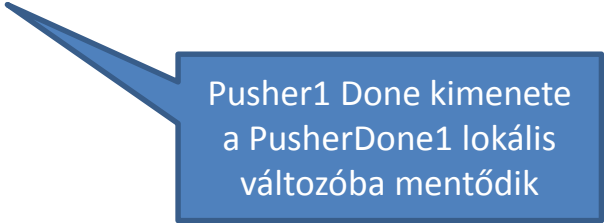
FB-példányok hívása

```
Conv1();
```

```
...
```

```
Pusher1(Start:=TRUE, Done=>PusherDone1);
```

```
(* Program Code *)
```



Pusher1 Done kimenete
a PusherDone1 lokális
változóba mentődik

```
END_PROGRAM
```

Konfiguráció

```
CONFIGURATION ManufCell
```

```
VAR_GLOBAL
```

```
    PartCount      : UINT:=0
```

```
END_VAR
```

```
RESOURCE PLC_CONV ON CLXL32E
```

```
    VAR
```

```
        ProxyPos AT %I3.1      : BOOL;
```

```
        AT %I0.0               : BOOL;
```

```
        AT %Q0.0               : BOOL;
```

```
    END_VAR
```

```
    TASK T_Pos (EVENT:ProxyPos, PRIORITY:=0);
```

```
    PROGRAM ProgInstancePosition: ProgPos WITH T_Pos;
```

```
    PROGRAM ProgConvControl: ProgConv
```

```
        (ProxySensor1:=%I0.0, ConvMotor1 => %Q0.0);
```

```
END_RESOURCE
```

Globális változó
létrehozása

Konfigurációban elérhető
változók

Eseményvezérelt taszk: a
pozícionáló egységet
vezérlő FB akkor fut, ha
érkezik munkadarab

Taszk-hozzárendelés nélkül
ProgConvControl ciklikusan
fut a legalacsonyabb
prioritással

Fizikai objektumok
hozzárendelése a program
bemeneti változóihoz

Konfiguráció

```
(* CONFIGURATION ManufCell continued *)
```

```
VAR_CONFIG
```

```
    PLC_CONV.ProgConvControl.Pusher1.LSFront AT %I3.0      : BOOL;  
    PLC_CONV.ProgConvControl.Pusher1.LSBack  AT %I3.1      : BOOL;  
    PLC_CONV.ProgConvControl.Pusher2.LSFront AT %I3.2      : BOOL;  
    PLC_CONV.ProgConvControl.Pusher2.LSBack  AT %I3.3      : BOOL;  
    PLC_CONV.ProgConvControl.Pusher1.VMotor  AT %QB4.0      : SINT;  
    PLC_CONV.ProgConvControl.Pusher1.LSBack  AT %QB4.1      : SINT;
```

```
END_VAR
```

```
VAR_ACCESS
```

```
    ManufacturedParts      : PartCount      : UINT READ_ONLY;
```

```
END_VAR
```

```
END_CONFIGURATION
```

Az egyes FB-k be- és kimeneteit konfiguráció-szinten fizikai objektumokhoz is rendelhetjük

A PartCount változót ManufacturedParts néven más konfigurációk is elérik, de csak olvashatják az értékét