

# **ISaGRAF**

**3.4-es verzió**

**KEZELÉSI ÚTMUTATÓ**

**ICS Triplex ISaGRAF Inc.**

Az ebben a dokumentációban szereplő információk előzetes bejelentés nélkül változhatnak, és azok nem jelentenek semmiféle kötelezettségvállalást a ICS Triplex ISaGRAF Inc. részéről. A szoftver, amely az ebben a dokumentumban ismertetett bármilyen adatbázisokban szereplő információkat tartalmaz, egy licenz-szerződés vagy titkossági szerződés értelmében kerül átadásra, és kizárólag azon szerződés kikötéseinek betartásával használható illetve másolható. A szoftvernek a licenz- illetve titoktartási szerződésben kifejezett módon megengedett módon történő másolásától eltérő módon történő másolása törvényellenes. A ICS Triplex ISaGRAF Inc. írásos engedélye nélkül tilos ennek a kézikönyvnek bármely részét, bármely célból, bármilyen formában, illetve bármilyen, akár elektronikus, akár mechanikus módon, sokszorosítani, ideértve a fénymásolást és az adatrögzítést is.

© 2003 ICS Triplex ISaGRAF Inc. Minden jog fenntartva.  
Nyomtatta a ICS Triplex ISaGRAF Inc., Kanadában

Az ISaGRAF a ICS Triplex ISaGRAF Inc. bejegyzett védjegye.  
Az MS-DOS a Microsoft Corporation bejegyzett védjegye.  
A Windows a Microsoft Corporation bejegyzett védjegye.  
A Windows NT a Microsoft Corporation bejegyzett védjegye.  
Az OS-9 és az ULTRA-C a Microsoft Corporation bejegyzett védjegyei.  
A VxWorks és Tornado a Wind River Systems, Inc. bejegyzett védjegyei.

Minden egyéb márka- illetve terméknév azok megfelelő tulajdonosainak a védjegyei vagy bejegyzett védjegyei.

# Tartalomjegyzék

<b>A. KEZELÉSI ÚTMUTATÓ</b>	<b>11</b>
<b>A.1 Kezdeti tudnivalók</b>	<b>12</b>
A.1.1 Az ISaGRAF telepítése	12
A.1.2 On-line információk használata	15
A.1.3 Egy minta alkalmazás	15
<b>A.2 Projektek kezelése</b>	<b>20</b>
A.2.1 Projektek létrehozása és projektekkel történő munkavégzés	20
A.2.2 Több projektcsoporttal való munkavégzés	22
A.2.3 Beállítási lehetőségek	23
A.2.4 Eszközök	23
<b>A.3 Programok kezelése</b>	<b>24</b>
A.3.1 Egy projekt komponensei	24
A.3.2 Programokkal való munkavégzés	26
A.3.3 A kódgeneráló eszközök futtatása	29
A.3.4 Egyéb ISaGRAF eszközök	30
A.3.5 Parancsok hozzáadása az Eszközök menühöz	31
A.3.6 Az alkalmazás szimulációja és hibakeresése	31
<b>A.4 Az SFC szerkesztő használata</b>	<b>34</b>
A.4.1 A SFC nyelv fő témái	34
A.4.2 Az SFC diagram bevitele	36
A.4.3 Egy meglevő SFC diagrammal történő munkavégzés	38
A.4.4 A 2. szintű programozás bevitele	39
A.4.5 Az SFC gyűjtemény használata	43
<b>A.5 A Blokkdiagram szerkesztő használata</b>	<b>45</b>
A.5.1 Az FC nyelv alapjai	45
A.5.2 Egy Blokkdiagram bevitele	46
A.5.3 Egy meglevő diagrammal történő munkavégzés	49
A.5.4 2. szintű programok bevitele	50

A.5.5	2. szint programozása Gyors LD-vel	51
A.5.6	Megjelenítési lehetőségek	52
<b>A.6</b>	<b>A Gyors LD szerkesztő használata</b>	<b>53</b>
A.6.1	Az LD nyelv alapjai	53
A.6.2	Egy LD diagram bevitele	55
A.6.3	Egy meglevő diagrammal történő munkavégzés	58
A.6.4	Megjelenítési lehetőségek	59
<b>A.7</b>	<b>Az FBD/LD szerkesztő használata</b>	<b>61</b>
A.7.1	Az FBD/LD nyelvek alapjai	61
A.7.2	Egy FBD diagram bevitele	63
A.7.3	Egy meglevő diagrammal történő munkavégzés	65
A.7.4	Megjelenítési lehetőségek	67
A.7.5	Stílusok és a módosítások nyomon követése	67
<b>A.8</b>	<b>A szövegszerkesztő használata</b>	<b>69</b>
A.8.1	Szerkesztő parancsok	69
A.8.2	Beállítási lehetőségek	70
<b>A.9</b>	<b>További programszerkesztőkkel kapcsolatos tudnivalók</b>	<b>71</b>
A.9.1	Egyéb ISaGRAF eszközök meghívása	71
A.9.2	A program paraméterei	71
A.9.3	A „Fájl” menü egyéb parancsai	72
A.9.4	A program-napló frissítése	73
A.9.5	Egy változó kiválasztása a szótárból	74
A.9.6	Az output ablak	75
<b>A.10</b>	<b>A szótárszerkesztő használata</b>	<b>76</b>
A.10.1	A fő szótáráblak	78
A.10.2	Változók kezelése	78
A.10.3	Tárgyak leírása	80
A.10.4	Gyors deklarálás	82
A.10.5	Modbus SCADA címező hozzárendelési lista	83
A.10.6	Információcsere más alkalmazásokkal	83
<b>A.11</b>	<b>Az I/O csatlakozásszerkesztő használata</b>	<b>88</b>
A.11.1	I/O kártyák definiálása	89
A.11.2	A kártyák paramétereinek beállítása	90
A.11.3	I/O csatornák csatlakoztatása	90

A.11.4	Közvetlenül képviselt változók	91
A.11.5	Számozás	91
A.11.6	Egyedi védelmek beállítása	92
<b>A.12</b>	<b>Konverziós táblázatok létrehozása</b>	<b>94</b>
A.12.1	Fő parancsok	94
A.12.2	Egy táblázat pontjainak beville	95
A.12.3	Szabályok és korlátozások	95
<b>A.13</b>	<b>A kódgenerátor használata</b>	<b>97</b>
A.13.1	Fő parancsok	97
A.13.2	Fordítási beállítások	98
A.13.3	C forráskód készítése	100
A.13.4	Információk megtekintése	101
A.13.5	Erőforrások definiálása	101
<b>A.14</b>	<b>Kereszthivatkozások</b>	<b>107</b>
<b>A.15</b>	<b>A grafikus hibakereső használata</b>	<b>109</b>
A.15.1	A hibakereső ablak	109
A.15.2	Az alkalmazás vezérlése	110
A.15.3	Beállítási lehetőségek	112
A.15.4	„Írás” parancsok	112
A.15.5	Online módosítás	114
A.15.6	DDE cserék	117
<b>A.16</b>	<b>Változók kémlelési listája</b>	<b>119</b>
<b>A.17</b>	<b>ST és IL programok hibakeresése</b>	<b>121</b>
<b>A.18</b>	<b>Hibakeresés a Fényszóróval</b>	<b>122</b>
A.18.1	A grafikus elrendezés felépítése	122
A.18.2	A lista elrendezés	124
A.18.3	A tétel stílusának definiálása	125
A.18.4	A „Fájl” menü parancsai	126
A.18.5	Megjegyzés ISaGRAF V3.2 felhasználók számára	126
<b>A.19</b>	<b>Alkalmazások feltöltése</b>	<b>127</b>
A.19.1	Egy projekt feltöltése:	127
A.19.2	Kommunikációs beállítások	127

A.19.3	A projekt előkészítése feltöltéshez	128
A.19.4	Hogyan van a tömörített forrás a célon tárolva	128
A.19.5	Memóriaigények a célon	129
A.19.6	A feltöltött projekttel kapcsolatos tudnivalók	129
A.19.7	Illeszthetőségi kérdések	129
<b>A.20</b>	<b>A Diagnózis eszköz használata</b>	<b>130</b>
<b>A.21</b>	<b>Az ISaGRAF szimulátor használata</b>	<b>131</b>
A.21.1	Kapcsolatok a hibakeresővel	131
A.21.2	I/O szimuláció	131
A.21.3	Könyvtár komponensek	132
A.21.4	Beállítási lehetőségek	132
A.21.5	Input állapotok kimentése és visszatöltése	133
A.21.6	A ciklus profilozó	133
A.21.7	Szimuláció szkriptek	134
<b>A.22</b>	<b>A Könyvtárrendező használata</b>	<b>142</b>
A.22.1	Könyvtárelemek kezelése	142
A.22.2	I/O konfiguráció	144
A.22.3	Komplex I/O berendezés	145
A.22.4	I/O kártya	146
A.22.5	Az IEC nyelveken írt funkciók és blokkok	147
A.22.6	„C” Funkciók és funkcióblokkok	149
A.22.7	Konverziós funkciók	150
<b>A.23</b>	<b>Az Archiváló segédprogram használata</b>	<b>151</b>
A.23.1	Az archívumrendező előhívása	151
A.23.2	Beállítási lehetőségek	152
A.23.3	Biztonsági másolat és visszatöltés	152
A.23.4	Archívumfájlok	153
<b>A.24</b>	<b>Egy teljes dokumentum kinyomtatása</b>	<b>154</b>
A.24.1	A tartalomjegyzék testre szabása	154
A.24.2	Beállítási lehetőségek	155
<b>A.25</b>	<b>Jelszóvédelem</b>	<b>158</b>
<b>A.26</b>	<b>Haladó programozási eljárások</b>	<b>161</b>
A.26.1	Az ISaGRAF eszközökkel kapcsolatos további tudnivalók	161

A.26.2	Zárt bemenetek/kimenetek (I/O-k) és virtuális I/O-k	161
A.26.3	PC-PLC kapcsolat érvényesítése	164
A.26.4	ISaGRAF fájlkönyvtárak	164
A.26.5	Alkalmazás szimbólumok	166
A.26.6	Az ISaGRAF „NAGY” (WDL) workbench korlátai	170

## **B. NYELVI HIVATKOZÁS 173**

### **B.1 Projekt architektúra 174**

B.1.1	Programok	174
B.1.2	Ciklikus és szekvenciális műveletek	174
B.1.3	Gyermek SFC és FC programok	175
B.1.4	Funkciók és alprogramok	175
B.1.5	Funkcióblokkok	177
B.1.6	Leíró nyelv	178
B.1.7	Végrehajtási szabályok	178

### **B.2 Közös tárgyak 180**

B.2.1	Alapvető típusok	180
B.2.2	Állandó kifejezések	180
B.2.3	Változók	182
B.2.4	Megjegyzések	186
B.2.5	Definiált szavak	186

### **B.3 Az SFC nyelv 188**

B.3.1	Az SFC diagram fő formátuma	188
B.3.2	Alapvető SFC komponensek	188
B.3.3	Divergenciák és konvergenciák	190
B.3.4	Makró lépések	192
B.3.5	A lépéseken belüli tevékenységek	193
B.3.6	Átmenetekhez csatolt feltételek	199
B.3.7	SFC dinamikus szabályok	200
B.3.8	SFC program hierarchia	201

### **B.4 Blokkdiagram nyelv 203**

B.4.1.	FC komponensek	203
B.4.2.	FC komplex struktúrák	206
B.4.3.	FC dinamikus viselkedés	207
B.4.4.	FC ellenőrzés	207

<b>B.5.</b>	<b>Az FBD nyelv</b>	<b>208</b>
B.5.1.	Az FBD diagram fő formátuma	208
B.5.2.	RETURN utasítás	209
B.5.3.	Ugrások és címkék	209
B.5.4.	Logikai tagadás	210
B.5.5.	Funkciók vagy funkcióblokkok meghívása az FBD-ből	211
<b>B.6.</b>	<b>Az LD nyelv</b>	<b>212</b>
B.6.1.	Áramsínek és összekötő vonalak	212
B.6.2.	Többszörös kapcsolat	213
B.6.3.	Alapvető LD kontaktok és tekercsek	214
B.6.4.	RETURN utasítás	220
B.6.5.	Ugrások és címkék	220
B.6.6.	Blokkok az LD-ben	221
<b>B.7.</b>	<b>ST nyelv</b>	<b>223</b>
B.7.1.	ST fő szintaxis	223
B.7.2.	Kifejezések és zárójelek	223
B.7.3.	Funkciók vagy funkcióblokkok meghívása	224
B.7.4.	ST specifikus logikai operátorok	225
B.7.5.	Alapvető ST utasítások	227
B.7.6.	ST bővítések	233
<b>B.8.</b>	<b>Az IL nyelv</b>	<b>238</b>
B.8.1.	IL fő szintaxis	238
B.8.2.	IL operátorok	239
<b>B.9.</b>	<b>Standard operátorok, funkcióblokkok, és funkciók</b>	<b>246</b>
B.9.1.	Standard operátorok	246
B.9.2.	Standard funkcióblokkok	267
B.9.3.	Standard funkciók	284
<b>C.</b>	<b>CÉL KEZELÉSI ÚTMUTATÓ</b>	<b>327</b>
<b>C.1.</b>	<b>Bevezetés</b>	<b>328</b>
<b>C.2.</b>	<b>Telepítés</b>	<b>329</b>

<b>C.3.</b>	<b>Az ISaGRAF DOS céllal történő munkavégzés elkezdése</b>	<b>330</b>
C.3.1.	Az ISaGRAF futtatása: ISA.EXE	330
C.3.2.	Különleges jellemzők:	331
<b>C.4.</b>	<b>Az ISaGRAF OS9 céllal történő munkavégzés elkezdése</b>	<b>335</b>
C.4.1.	Az ISaGRAF egyszeres feladat futtatása: isa	335
C.4.2.	Az ISaGRAF multitaszkok futtatása: isaker, isatst, isanet	336
C.4.3.	Különleges jellemzők:	340
<b>C.5.</b>	<b>Az ISaGRAF VxWorks céllal történő munkavégzés elkezdése</b>	<b>345</b>
C.5.1.	A rendszer forráskezelő: isassr.o	345
C.5.2.	Az isa.o, isakerse.o és isakeret.o közös jellemzői	345
C.5.3.	Az ISaGRAF egyszeres feladat futtatása: isa.o	346
C.5.4.	Az ISaGRAF multitaszkok futtatása: isakerse.o és isakeret.o	348
C.5.5.	Különleges jellemzők:	352
<b>C.6.</b>	<b>Az ISaGRAF NT céllal történő munkavégzés elkezdése</b>	<b>357</b>
C.6.1.	Az ISaGRAF futtatása	357
C.6.2.	A beállítási lehetőségekről szóló általános tudnivalók	357
C.6.3.	Különleges jellemzők:	363
C.6.4.	Felhasználói interfész	367
<b>C.7.</b>	<b>„C” programozás</b>	<b>373</b>
C.7.1.	Áttekintés	373
C.7.2.	„C” konverziós funkciók	374
C.7.3.	„C” funkciók	379
C.7.4.	„C” FUNKCIÓBLOKKOK	387
C.7.5.	Programfordítási és kapcsolási módszerek	402
<b>C.8.</b>	<b>Modbus kapcsolat</b>	<b>409</b>
C.8.1.	MODBUS hálózat és protokoll	409
C.8.2.	ISaGRAF kivitelezés	410
<b>C.9.</b>	<b>Áramkimaradás kezelése</b>	<b>416</b>
C.9.1.	Alapismeretek	416
C.9.2.	Alkalmazás-változók biztonsági másolata	417
C.9.3.	Program állapot biztonsági másolat:	420
<b>C.10.</b>	<b>Függelék: Hibalista és leírás</b>	<b>422</b>

<b>D</b>	<b>SZÓSZEDET</b>	<b>433</b>
<b>E</b>	<b>ÁLTALÁNOS TÁRGYMUTATÓ</b>	<b>441</b>

## **A. Kezelési útmutató**

## A.1 Kezdeti tudnivalók

Ez a fejezet az ISaGRAF workbench telepítésével foglalkozik. Ez egy ISaGRAF alkalmazás rövid példáját is tartalmazza, amely röviden vázolja a felhasználó számára annak fő jellemzőit, és lehetővé teszi az ISaGRAF azonnali használatát.

### A.1.1 Az ISaGRAF telepítése

Ez a fejezet az ISaGRAF Workbench telepítését, valamint a számítógép-alkalmazás fejlesztéshez történő beállítását tartalmazza.

#### **Hardver- és szoftverigények**

Az ISaGRAF Workbench bármilyen, a Windows 3.1-es verziójának minimális igényeit kielégítő számítógépre telepíthető. Az alkalmazásfejlesztéshez azonban javasoljuk a következő hardvert:

- Egy 80486-os, vagy attól magasabb változatú mikroprocesszort alkalmazó személyi számítógép  
(Javasoltan Pentium processzor)
- 8 megabyte konvencionális és kiterjesztett memória  
(16 megabyte javasolt)
- Egy 89 mm-es (3,5 hüvelykes) (1.44 megabyte-os) lemezmeghajtó
- Egy merevlemez legalább 20 megabyte szabad területtel
- Egy VGA vagy SVGA grafikus adapter, valamint megfelelő monitor
- Egy egér (a grafikus fejlesztőeszközökhöz szükséges)
- Egy párhuzamos LPT1 port (a hardveres védelemhez szükséges)

Az ISaGRAF workbench telepítése előtt a következő szoftvernek már jelen kell lennie a rendszeren:

- 386-os kiemelt üzemmódban futó Windows 3. 1-es verzió
- Windows 95
- Windows NT 3. 51-es vagy 4. 00-ás verzió



#### **A telepítőprogram használata**

Az ISaGRAF workbench telepítése az ISaGRAF INSTALL (TELEPÍTÉS) nevű telepítőprogramjával történik. Ez a program az ISaGRAF szoftvert az ISaGRAF CD-ROM-ról vagy a floppy lemezekről a felhasználó merevlemezére másolja. Az INSTALL emellett a Programrendező ablakhoz hozzáad egy „ISaGRAF” nevű csoportot is, és a telepített **EXE** alkönyvtárban létrehoz egy „ISA.ini” elnevezésű inicializáló fájlt.

Az INSTALL egy Windows program, amelyet a Windows Programrendezőjéből, vagy a Windows 95 Start menüjéről kell futtatni. Az ISaGRAF telepítéséhez a következő lépéseket kell végrehajtani:

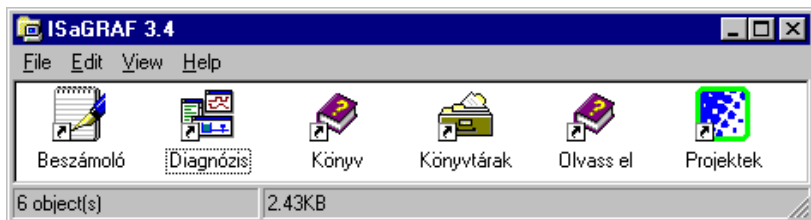
- Helyezze be az ISaGRAF CD-ROM-ot vagy az 1. floppy lemezt a megfelelő meghajtóba.
- A Programrendezőből illetve a Start menüből futtassa a CD-ROM főkönyvtárában levő „SETUP.EXE” fájlt, illetve floppy lemezek esetén az „A:\INSTALL.EXE” fájlt.
- A telepítés elvégzéséhez kövesse az on-line utasításokat. Javasoljuk, hogy az ISaGRAF Workbench-et egy új alkönyvtárba telepítse annak érdekében, hogy elkerülje a fájlok összekeverését a korábbi ISaGRAF verziókból származó fájlokkal.

Az INSTALL megkérdezi, hogy szükségesek-e a következő komponensek:

- ISaGRAF végrehajtandó programok
- On-line információs- és súgó fájlok
- ISaGRAF standard könyvtárak
- ISaGRAF minta alkalmazások

Az ISaGRAF első alkalommal történő telepítésekor határozottan javasoljuk valamennyi komponenst telepítését. A későbbiekben azonban az ISaGRAF Workbench újratelepítésével lehetséges a további komponensek utólagos hozzáadása is.

Az ISaGRAF főkönyvtár alapértelmezett neve „ISAWIN”. Ez lehetővé teszi az „ISaGRAF for Windows” ugyanarra a lemezre történő telepítését, mint amelyre az ISaGRAF for DOS valamelyik verziója már telepítve van. Az ISaGRAF lemezstruktúrával kapcsolatos további információk a „Haladó módszerek” című fejezet „ISaGRAF alkönyvtárak” című részében olvashatók. Miután valamennyi ISaGRAF fájlt át lett másolva, a következő csoport lesz a Programrendező Ablakhoz adva:



Az alábbiakban a fő ISaGRAF ikonok vannak felsorolva:

**Projekt:** .....Projekt kezelés  
**Könyvtárak:** .....Könyvtárak kezelése  
**Könyv:** .....Az ISaGRAF-al kapcsolatos on-line információk  
**Diagnózis:** .....A végfelhasználó számára szolgáló diagnosztikai eszköz  
**Olvass el:** .....Az ISaGRAF új verziójával kapcsolatos tudnivalók  
**Jelentés:** .....Standard hibabejelentő nyomtatvány

Amennyiben bármilyen problémát tapasztal, használja a standard hibabejelentő nyomtatványt. Nyissa meg azt, töltsse ki a kért tételeket, és a Fájl/Kimentés más néven menüparancs használatával mentse ki azt egy adott fájlnevével. Ezután faxon vagy emailben küldje el ezt a fájlt a ICS Triplex ISaGRAFhoz.

## ⇒ **Rendszerfájlok frissítése**

A telepítés befejeztekor, a számítógép újraindítását megelőzően, módosítani kell a CONFIG.SYS fájlt. Az ISaGRAF könyvtár elérési útját nem szükséges beírni a PATH változóba.

Az ISaGRAF nem használ semmilyen MS-DOS környezet változót. A CONFIG.SYS fájlba azonban be lehet írni a következő utasításokat:

```
files=20  
buffers=20
```

Az ISaGRAF Workbench egy soros portot használ az ISaGRAF cél PLC-vel történő kommunikációjához. Az ISaGRAF számára az alapértelmezett soros port a COM1. Amennyiben az egér szintén soros portot használ, úgy az egér számára a COM2-t kell kijelölni annak érdekében, hogy az alapértelmezett COM1 specifikáció érvényes maradjon bármely új ISaGRAF alkalmazás számára.

A CONFIG.SYS fájl módosítása után újra kell indítani a számítógépet ahhoz, hogy a változások érvénybe lépjenek.

## ⇒ **Windows NT felhasználók számára szóló fontos tudnivaló:**

Ha a Workbench-et Windows NT 3.51 vagy 4.00 alatt használják, akkor a következő sorokat be kell írni az \ISAWIN\EXE alkönyvtárban levő ISA.ini fájl [WS001] bekezdésébe

```
[WS001]  
NT=1  
Isa=C:\ISAWIN  
IsaExe=C:\ISAWIN\EXE  
IsaApl=C:\ISAWIN\APL1  
IsaTmp=C:\ISAWIN\TMP
```

Ez feltétlenül szükséges az RS kommunikációhoz.

## ⇒ **A hardveres védelmi kulcs**

Az ISaGRAF szoftvert egy hardveres védelmi kulcs védi az illegális másolatok készítése ellen. Az ISaGRAF workbench legtöbb funkciója azonban akkor is rendelkezésre áll, ha a kulcs nincs behelyezve. A hardveres védelmi kulcs az ISaGRAF Workbench opcióját is meghatározza, valamint behatárolja a fejlesztett alkalmazások maximális méretét. Ha a kulcs nincs behelyezve, illetve nincs megfelelő módon csatlakoztatva, az ISaGRAF Workbench bizonyos funkciói nem működnek. Ez NORMÁLIS viselkedésnek számít. Annak biztosítása érdekében, hogy a kulcs megfelelő módon csatlakoztatva legyen, bármelyik ISaGRAF ablakból válassza ki a „**Névjegy...**” lehetőséget a „**Súgó**” menüről. Ekkor megjelenik az ISaGRAF workbench rendelkezésre álló opciója.

A kulcs a számítógép bármelyik párhuzamos portjához csatlakoztatható. Ha a gép egynél több párhuzamos porttal rendelkezik, akkor javasoljuk, hogy a nyomtatót és a kulcsot különböző portokhoz csatlakoztassa. Bizonyos PC/nyomtató konfigurációnál előfordulhat, hogy a rendszer nem ismeri fel a kulcsot, ha az egy „off-line” állapotú nyomtatóhoz van csatlakoztatva. Ilyen esetben válassza le a nyomtatót, vagy indítsa el azt „on-line” állapotban, és indítsa újra az ISaGRAF workbench-et.

Megjegyzendő, hogy az **ISaGRAF-32** Workbench-hez nincs szükség kulcsra.



**Windows NT felhasználók számára szóló fontos tudnivaló:**

Windows NT rendszereken a Sentinel/Rainbow™ meghajtónak telepítve kell lennie ahhoz, hogy a védelmi kulcs látható legyen. Ehhez egy külön lemez van mellékelve.

### A.1.2 On-line információk használata

Az ISaGRAF workbench-el a következő témakörökhöz on-line információ van telepítve:

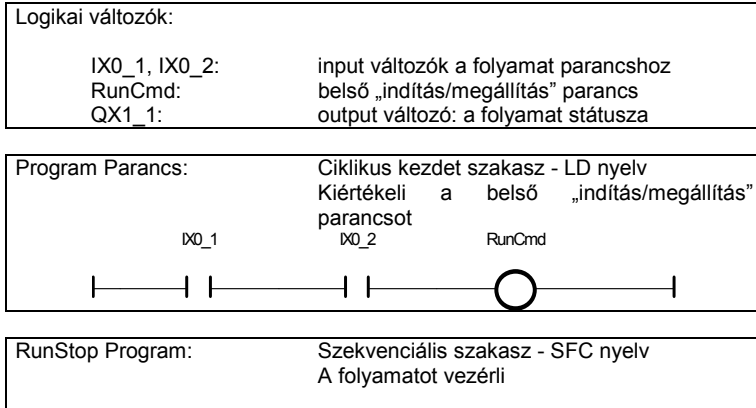
- ISaGRAF nyelvi hivatkozások
- Teljes Kezelési útmutató (bármelyik ISaGRAF eszközhöz)
- Műszaki megjegyzés a könyvtárakban levő elemekhez

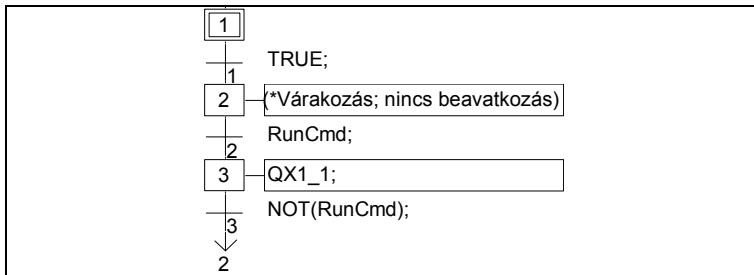
Bármelyik ISaGRAF ablakból válassza ki a „**Súgó**” menü választási lehetőségeit az online információ megjelenítéséhez.

### A.1.3 Egy minta alkalmazás

Ez a fejezet lépésről-lépésre elmagyarázza valamennyi alpműveletet, amelyek egy rövid, de komplett többnyelvű alkalmazás elkészítéséhez, megtervezéséhez, generálásához, és teszteléséhez szükségesek.

Az alábbiakban ennek az alkalmazásnak a teljes specifikációja látható, amely keverve tartalmaz mind LD, mind SFC ábrázolásokat:





## Kezdet **Az ISaGRAF workbench futtatása**

Az ISaGRAF workbench futtatásához futtassa a Windows Start menüjéről az „ISaGRAF” csoportban levő „Projektek” parancsot.



### **A projekt létrehozása**

A projektet (melynek neve „RunStop”) a „Fájl” menü „Új” parancsával, vagy az Új gombbal lehet létrehozni. A nyitott párbeszédablakban:

Írja be a projekt nevét: **„RunStop”**

Válassza ki az I/O konfigurációt: **„Sim\_Boo”**

Nyomja meg az **„OK”** gombot.

Ezzel létre lett hozva a projekt.



### **A projekt megnyitása**

A projekt programjainak definiálása az ISaGRAF programkezelő ablakának megnyitásával történik. Használja a projektkezelő ablak „Megnyitás” parancsát, vagy kattintson az egérrel duplán a projekt nevére, illetve használja a Szerkesztés gombot.



### **A programok létrehozása**

Ekkor megnyílik az üres programrendező ablak (nincsenek definiálva programok). Az első program a „Fájl” menü „Új” parancsával, vagy az „Új” gomb használatával hozható létre. A nyitott párbeszédablakban:

Írja be a program nevét: **„Parancs”**.

Válassza ki a **„Gyors LD”** nyelvet.

Válassza ki a **„Ciklus kezdete”** szakaszt.

Nyomja meg az **„OK”** gombot a program létrehozásához.

A második programhoz ugyanezt a műveletet kell megismételni:

Használja a „Fájl” menü „Új” parancsát, vagy az „Új” nyomógombot. A nyitott párbeszédablakban:

Írja be a program nevét: **„RunStop”**

Válassza ki az **„SFC”** nyelvet.

Válassza ki a **„Szekvenciális”** szakaszt.

Nyomja meg az **„OK”** gombot a program létrehozásához.

Ezzel most a programok létre lettek hozva. Azok megjelennek a programkezelő ablakban.



## A változók deklarálása

A programok bevétele előtt deklarálni kell a programozásban használandó belső változót. Ez a „Fájl” menü „Szótár” parancsával, vagy a Szótár gombbal történik. A projekt létrehozásakor az I/O változók automatikusan deklarálásra kerülnek.



Ekkor megnyílik a szótár ablak. A „Fájl” menü „Egyéb” almenüjében levő „Globális változók” választási lehetőségével, majd a „Logikaiak” parancssal válassza ki a „Globális” logikai szótárat. A Globális tárgyak és Logikai gombok ugyanilyen módon alkalmazhatók erre a célra.



A „Szerkesztés” menü „Új” parancsa új logikai változók létrehozására szolgál. A Tárgyak beillesztése gomb ugyancsak használható. A megnyitott párbeszédablakban írja be a belső változó leírását:

név:

**RunCmd**

megjegyzés:

**Futás/Megállás parancs: belső**

attribútum:

Válassza ki a „**Belső**” attribútumot

Nyomja meg a „**Tárolás**” gombot: a változó létrehozódik.

Nyomja meg a „**Mégsem**” gombot a párbeszédablakból történő

kilépéshez.

Végül lépjen ki a szótárszerkesztőből, és mentse ki a bevitt módosításokat: „Fájl” menü – „**Kilépés**” parancs. A módosítások kimentéséhez kattintson az „**IGEN**” gombra.



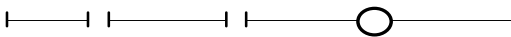
## A Gyors LD program szerkesztése

A „Parancs” LD program szerkesztésének elkezdéséhez kattintson duplán annak nevére a Programkezelő ablakban, vagy használja a Szerkesztés gombot.



Ekkor kinyílik az ISaGRAF Gyors LD Szerkesztő ablak. A munkaterület megnöveléséhez méretezze át az ablakot, hogy a teljes képernyőterületet használni tudja.

**F2 F3** Nyomja meg az F2 és F3 gombot:  
(“ ”)



Társítson változókat az LD szimbólumokhoz: a kurzor mozgatásához használja a billentyűzet nyíl gombjait. Vigye a kurzort az egyes szimbólumokra, és nyomja meg az Enter billentyűt. Ekkor megnyílik a változó kiválasztó párbeszédablak.

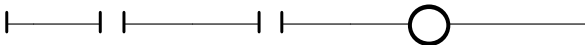
Az első kontakthoz írja be a következőt a változó kiválasztó ablakba: IX0\_1, majd nyomja meg az Enter billentyűt.

A második kontakthoz írja be a következőt a változó kiválasztó ablakba: IX0\_2, majd nyomja meg az Enter billentyűt.

A tekercshez írja be a következőt a változó kiválasztó ablakba: RunCmd, majd nyomja meg az Enter billentyűt.

Ezzel kész a program. Ennek eredménye a következő:

IX0\_1 IX0\_2 RunCmd



Lépjen ki a szerkesztőből, és mentse ki a bevitt módosításokat: „Fájl” menü – „**Kilépés**” parancs. A módosítások kimentéséhez kattintson az „**IGEN**” gombra.



## Az SFC program szerkesztése

A „RunStop” SFC program szerkesztésének elkezdéséhez kattintson duplán annak nevére a Programkezelő ablakban, vagy használja a Szerkesztés gombot.



Ekkor kinyílik az SFC Szerkesztő ablak. A munkaterület megnöveléséhez méretezze át az ablakot, hogy a teljes képernyőterületet használni tudja:



A kezdeti lépés már létezik, és az ki van választva. A kezdeti lépés utáni üres cella kiválasztásához nyomja meg a billentyűzeten levő „Lefelé” nyíl billentyűt (0,1)

**F4 F3**

Nyomja meg az F4, majd az F3 billentyűt egy lépés és egy átmenet beillesztéséhez.

**F4 F3**

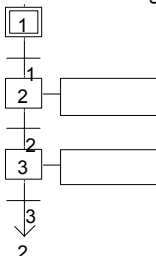
Nyomja meg az F4, majd az F3 billentyűt egy további lépés és átmenet beillesztéséhez.

**F5**

Nyomja meg az F5 billentyűt egy lépésre történő ugrás beillesztéséhez, és válassza ki a GS2-t az ugrás célpontjaként.



Ezzel kész a diagram. A cellák méretének megnöveléséhez illetve a 2. szintű utasítások megjelenítéséhez szükséges hely teremtetéséhez nyomja meg az eszközsoron levő „Zoom” gombot. Íme a diagram:



A „2”-es átmenet programozásának beviteléhez a billentyűzet nyíl billentyűivel válassza ki azt, és nyomja meg az „Enter” billentyűt. Ekkor kinyílik a 2. szintű programozási ablak. Írja be a 2. szintű programozást a 2. átmenethez:

**RunCmd;**

**^TAB**

Nyomja meg a „Control + Tab” billentyűket, hogy a fókuszt visszavigye az SFC diagramra, vigye a kiválasztást a 3. lépésre, és nyomja meg az „Enter” billentyűt 2. szintű szövegének szerkesztéséhez:

**QX1\_1;**

Majd végezze el ugyanezt a 3-as átmenet szövegének beviteléhez:

**Not (RunCmd);**

**^F4**

Nyomja meg a „Control + F4” billentyűket a 2. szintű ablak bezárásához. Ezzel kész az SFC program. Lépjen ki a szerkesztőből a „Fájl” menü „Kilépés” parancsával, és az „IGEN”-re kattintva mentse ki a módosításokat.



## Az alkalmazás kód felépítése

Az alkalmazás kód felépítéséhez használja a „Készítés” menü „Alkalmazás készítése” parancsát a Programkezelő ablakból, vagy az Eszközsoron levő gombot.

Amikor kész a kódgenerálás, megjelenik egy párbeszédablak, amely megkérdezi, hogy a kódgenerálásból **most** kilép, vagy **folytatja** vele a munkavégzést: Nyomja meg a „**Kilépés**” gombot.



### **Szimuláció**

Az ISaGRAF kernel szimulátor futtatásához használja a „**Hibakeresés**” menü „**Szimuláció**” parancsát a Programkezelő ablakból, vagy az Eszközsoron levő gombot.

A Szimulátor ablak megjelenésekor az alkalmazás tesztelhető. Ebben a példában mind az 1, mind a 2 inputokat (zöld gombok) meg kell nyomni ahhoz, hogy a folyamat fusson (output piros LED lámpák).

A szimulációból való kilépéshez zárja be a **Hibakereső** ablakot: „**Fájl**” menü – „**Kilépés**” parancs.

## A.2 Projektek kezelése

Az ISaGRAF projektkezelő eszköz futtatásához az egérrel kattintson duplán az ISaGRAF csoportban levő „Projektek” ikonra. Ekkor kinyílik a „Projektkezelés” ablak. Egy projekt egy cél PLC-n futtatott PLC huroknak felel meg. A felső ablak a létező projektek listáját tartalmazza. A kiválasztott projekt szöveges leírója az alsó ablakban látható.



### **Ablakok átméretezése**

A megfelelő ablakok átméretezéséhez egyszerűen kattintson a projektlista és a leíró között levő elválasztóra (kettéválasztó). A leíró ablak nem rejthető el teljesen. Ez mindig tartalmaz legalább egy sornyi szöveget.



### **Elválasztók beillesztése**

Bármelyik projekt neve elé beilleszthető egy elválasztó vonal. Ez lehetővé teszi a lista elrendezésében ugyanahhoz az alkalmazáshoz csatolt projektek csoportosítását. A kiválasztott projekt elé a „**Szerkesztés / Elválasztó átváltása**” paranccsal illeszthető be illetve törölhető egy elválasztó.



### **Projektek áthelyezése a listában**

Egy projektnek a listán belüli áthelyezéséhez először azt ki kell választani (kiemelni). Ezután kattintson a nevére, és vonszolja a listán levő új helyre. A projekt vonszolása közben a bal margón egy kis nyíl jelzi, hogy hová lesz helyezve. A kiválasztott projekt sorról-sorra történő áthelyezéséhez a „**Szerkesztés**” menü „**Áthelyezés**” parancsát is lehet használni. Megjegyzendő, hogy ha a kiválasztott projekt elé egy elválasztó lett helyezve, akkor a projekttel együtt az is áthelyezésre kerül.

### A.2.1 Projektek létrehozása és projektekkel történő munkavégzés

A projektrendező menü parancsai új projektek létrehozására, azok szerkesztésére, valamint a meglévő projektek kezelésére szolgálnak.



#### **Egy új projekt létrehozása**

Egy új projekt létrehozásához először be kell írni annak nevét. Ekkor egy üres projekt lesz létrehozva, benne levő tárgyak nélkül. Az újonnan létrehozott projekthez egy I/O konfigurációt lehet csatolni. Ezt az I/O konfigurációt a könyvtárban definiálni kell. Ha egy konfiguráció lett kiválasztva, akkor az ISaGRAF automatikusan beállítja az I/O csatlakozást és deklarálja a megfelelő I/O változókat az új projekt szótárban. Egy projekt létrehozásánál vagy átnevezésénél a következő elnevezési szabályokat kell betartani:

- a név nem haladhatja meg a **8** karaktert
- az első karakternek **betűnek** kell lennie
- a többi karakterek lehetnek **betűk**, **számjegyek**, vagy aláhúzás karakter
- a projekt neve nem érzékeny a kisbetű/nagybetű beállításra

Amikor létre lett hozva egy projekt, a „**Szerkesztés / Megjegyzés szöveg beállítása**” paranccsal a projekt nevével együtt a listában megjelenő szöveget lehet beírni.



### **A projektleíró szerkesztése**

A „**Projekt / Projektleíró**” parancs a projekt szöveges leírójának szerkesztésére szolgál. Ez a dokumentum teljes mértékben azonosítja a projektet a projektlistában levő többi projekt közül. A projektleíró a projekt élettartama során tett megjegyzések rögzítésére is szolgálhat.



### **A projekt szerkesztése**

A „**Fájl / Megnyitás**” parancs megnyitja a Programkezelő ablakot a kiválasztott projekthez. Ebből az ablakból a projekt teljes tartalma (programok, alkalmazás paraméterek, stb.) kezelhető. A projekt szerkesztése a projekt nevére történő dupla kattintással is elérhető.



### **A módosítások előzményei**

Az ISaGRAF rendszer egy projekt komponensére vonatkozó valamennyi módosítást egy előzményfájlban tárol. Az előzményekben minden módosítást egy cím, dátum, és időpont azonosít. Az előzményfájl az utolsó **500** módosítást tartalmazza. Minden projekthez egy előzményfájl tartozik. A projekt módosításainak előzményei a projekt programjaihoz csatolt „napló” fájlokat egészítik ki. A felhasználó a „**Projekt / Előzmények**” parancs segítségével a projekt módosítások előzményeinek megtekintését, vagy nyomtatását végezheti el. A felhasználó a fő listáról egy vagy több tételt választhat ki, és a következő gombokat nyomhatja meg:

OK .....bezárja ezt az ablakot

Nyomtatás .....a lista tartalmát a nyomtatóra küldi

Súgó .....megjeleníti ennek a párbeszédablaknak a súgóját

[törlés] Kiválasztott eltávolítja (törl) a listáról kiválasztott sorokat.

[törlés] Minden ....a teljes listát törli

Keresés .....a listában megkeres egy mintát

A „**Keresés**” gomb fölött levő párbeszédablak egy keresési minta beírására szolgál. Ez a funkció nem érzékeny a kisbetű/nagybetű beállításra. Amikor a keresés elérkezik a lista aljára, akkor továbbfolytatódik a lista tetejétől, egészen a kiindulási helyig.



### **Egy teljes dokumentum kinyomtatása**

A felhasználó a „**Projekt / Nyomtatás**” parancs segítségével a kiválasztott projekthez egy komplett dokumentumot állíthat össze és nyomtathat ki. Ez a dokumentum a kiválasztott projekt bármely komponensét (program, változó, paraméterek, stb.) csoportosíthatja. Egy specifikus (nem komplett) dokumentum felépítéséhez a felhasználónak csak annak tartalomjegyzékét kell definiálnia.



### **Jelszóvédelem**

A felhasználó a „**Projekt / Jelszó beállítása**” parancs segítségével a kiválasztott projekt eszközeihez és adataihoz jelszóvédelmet definiálhat. A jelszó szintekkel és adatvédelemmel kapcsolatos további információk ennek a kézikönyvnek a végén, a „**Jelszavas védelem**” című fejezetben találhatók. A jelszavak csak a kiválasztott

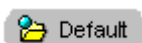
projektekre vonatkoznak. Ezek nincsenek hatással az ISaGRAF könyvtárakra és más projektekre.

## A.2.2 Több projektcsoporthal való munkavégzés

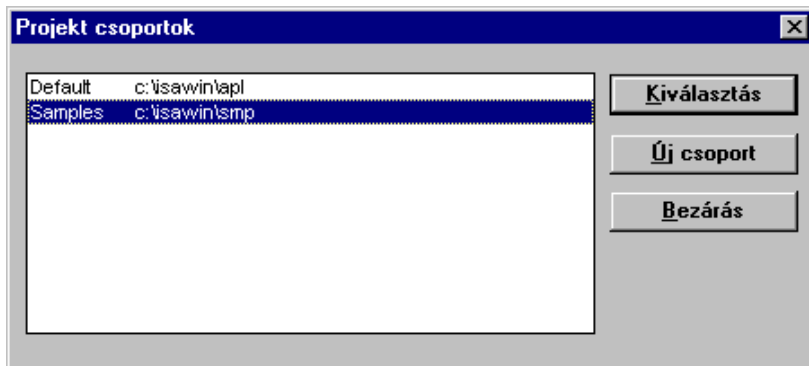
Egy ISaGRAF projekt a lemezen egy alkönyvtárnak felel meg, amelyben a projekt összes fájlja van tárolva. Egy „Projektcsoport” az ugyanabban a főkönyvtárban csoportosított projekt alkönyvtárak listáját jelenti. Egy projektcsoporthoz egy név azonosít. Az ISaGRAF alapértelmezésként két projektcsoporthoz hoz létre:

„**Alapértelmezett**” az „\ISAWIN\APL”-ban: ez az Ön munkaterülete  
 „**Minták**” .....az „\ISAWIN\SMP”-ben: az ISaGRAF workbench-el szállított  
 minta alkalmazások

A pillanatnyilag kiválasztott projektcsoport neve az eszközsorban látható, a projektcsoport kiválasztására szolgáló gomb mellett:



Egy meglevő csoport kiválasztásához vagy egy új létrehozásához a **„Fájl / Projektcsoport kiválasztása”** is futtatható. Ekkor kinyílik a következő párbeszédablak:



Válasszon ki egy csoportot a listából, és annak aktiválására nyomja meg a **„Kiválasztás”** gombot a projektkezelő listában. A kiválasztás a nevére való dupla kattintással is elvégezhető. Egy új csoport létrehozására az **„Új csoport”** parancsot kell használni. Ez a parancs vagy egy meglevő alkönyvtárhoz jelöl ki egy csoport nevet, vagy egy új csoportot hoz létre egy új alkönyvtárral.

**Megjegyzés:** Csoportokat nem lehet kiválasztani vagy létrehozni, ha más ISaGRAF ablakok (programkezelő, szerkesztők, stb.) nyitva vannak.

### A.2.3 Beállítási lehetőségek

A **„Beállítások”** menü parancsai az eszközsor megjelenítésére vagy elrejtésére, valamint a szöveg karakterfontjának kiválasztására, illetve a Projektkezelő „automatikus bezárás” módjának beállítására szolgálnak. A kiválasztott karakterfont a projektleíró megjelenítésére használatos, és ezt használja az összes ISaGRAF szövegszerkesztő is.

Ha a **„Projektkezelő nyitva tartása”** választás ki van kapcsolva, akkor a Projektkezelő ablak egy projekt bevitелеkor automatikusan bezáródik.

### A.2.4 Eszközök

Az **„Eszközök”** menü parancsai más ISaGRAF alkalmazások futtatására használhatók: Az **„Eszközök / Projektek archiválása”** parancs az ISaGRAF archívumkezelőt futtatja projektek kimentésére, illetve visszatöltésére. Az **„Eszközök / Közös adatok archiválása”** parancs az összes projekt által használt fájlok (mint pl. közös definiált szavak) kimentésére vagy visszatöltésére szolgál.

Az **„Eszközök / Könyvtárak”** parancs az ISaGRAF könyvtárrendezőt futtatja egy külön ablakban.

Az **„Eszközök / IL program importálása”** egy szövegfájlban egyetlen IL programként leírt projekt importálására szolgál, a PLC Fájl megnyitás csereformátumának megfelelően.

## A.3 Programok kezelése

A programkezelő ablak az alkalmazás programjait mutatja (amelyeket moduloknak vagy programozási egységeknek is neveznek), és menüiben csoportosítja a rendelkezésre álló parancsokat, a projekt architektúrájának létrehozására, valamint a szerkesztők, a programfordító, és a hibakereső futtatására. Ez az ablak egy alkalmazás fejlesztése közben a workbench kernel. A Programkezelő ablak a Projektkezelő ablak „Megnyitás” parancsának futtatásakor kinyílik.

### A.3.1 Egy projekt komponensei

Egy projekt alkotóelemeit **programoknak** nevezik. A program egy olyan logikai egység, amely a vezérlés végrehajtásának egy részét írja le. A globális változók (pl. I/O változók) az alkalmazás bármelyik programjában használhatók. A helyi változókat csak egy program használhatja. A programok egy **faszerkezetű hierarchiában** vannak felsorolva, amelyek különböző **logikai szekciókra** oszlanak. Az ablak a programokat és a közöttük levő kapcsolatokat mutatja. A „**Felső szintű**” programok a hierarchikus faszerkezet bal oldalán jelennek meg.

#### ⇒ **Felső szintű programok**

A felső szintű programok a hierarchikus faszerkezet bal oldalán jelennek meg. Az első három szekció felső szintű programjai mindig aktívak, és azok a következők sorrendben vannak végrehajtva a futtatási ciklus (leolvasás) során:

- (Inputok olvasása)
- A **KEZDÉS** szekció felső szintű programjainak végrehajtása
- A **SZEKVENCIÁLIS** szekció felső szintű programjainak végrehajtása
- A **BEFEJEZÉS** szekció felső szintű programjainak végrehajtása
- (Outputok frissítése)

A „**Kezdés**” vagy „**Befejezés**” szekciók programjai ciklikus műveleteket írnak le. Ezek az időtől függetlenek. A „**Szekvenciális**” szekció programjai szekvenciális műveleteket írnak le, ahol az Időváltozó kimondottan az alapvető műveleteket különbözteti meg. A „**Kezdés**” szekció fő programjai az egyes futtatási időciklusok kezdetén kerülnek szisztematikusan végrehajtásra. A „**Befejezés**” szekció főprogramjai az egyes futtatási időciklusok végén kerülnek szisztematikusan végrehajtásra. A „**Szekvenciális**” szekció fő programjai az **SFC** vagy **FC** szabályok alapján kerülnek végrehajtásra, és azoknak az **SFC** vagy **FC** nyelvben írottaknak kell lenniük. A ciklikus szekciók programjai nem lehetnek leírva az **SFC** vagy **FC** nyelvben. Bármelyik szekció bármely programjának lehet egy vagy több **alprogramja**.

#### ⇒ **Funkciók és funkcióblokkok**

A „**Funkciók**” szekció programjait a projektben bármelyik szekció bármelyik programja meghívhatja. Egy funkció egy algoritmus, amely több input értékből egy output értéket dolgoz fel. Egy funkció algoritmus csak illékony közbenső változókkal működik, amelyek az egyik meghívástól a másikig kitörölődnek. Ebből az következik,

hogy egy funkció soha nem hívhat meg egy funkcióblokkot. A „**Funkciók**” szekció programjai nem írhatók le **SFC** vagy **FC** nyelvben.

A funkcióktól eltérően a „**Funkcióblokkok**” egy input értékekre ható algoritmust rejtett statikus adatokkal társítanak, amelyeket a rendszer a funkcióblokk minden egyes felhasználásakor lemásol (instancionál). A „**Funkcióblokkok**” szekció programjait a projektben levő bármely szekció bármely programja meghívhatja. Ezek nem programozhatók **SFC** vagy **FC** nyelvben.

## Alprogramok

Az alprogramok egyetlen (SFC, FC, vagy egyéb) apaprogramhoz rendelt funkciók. Egy alprogramot csak szülőprogramja hajthat végre (hívhat meg). Mindegyik szekció mindegyik programjának lehet egy vagy több alprogramja. Egy alprogram leírásához az **SFC** vagy **FC** kivételével bármilyen nyelv felhasználható.

## Gyermek SFC és FC programok

A **gyermek SFC program** egy olyan párhuzamos program, amelyet szülőprogramja indíthat el, illetve szüntethet meg. Mind a szülőprogramot, mind a gyermekprogramot **SFC** nyelvvel kell leírni.

Amikor egy szülőprogram elindít el egy gyermek **SFC** programot, akkor az egy **SFC** vezérjelet helyez a gyermekprogram minden egyes kezdeti lépésbe. Amikor egy szülőprogram megszüntet egy gyermek **SFC** programot, akkor kitörli a gyermek lépéseiben létező összes vezérjelet.





A szekvenciális szekció bármelyik **FC** programja vezérelhet más **FC** alprogramokat. Egy **FC** apaprogram egy **FC** alprogram végrehajtása közben blokkolódik (várakozik). Egy apa **FC** programban és annak egyik **FC** alprogramjában nem lehet egyidejűleg műveleteket végezni.



## A programok és alprogramok közötti kapcsolatok:

Az alprogramokat és gyermek programokat a hierarchikus faszerkezetben egy vonal köti össze szülő programjukkal. Egy **SFC** program és egy **SFC** gyermek program közötti kapcsolat nyíllal van lezárva. Megjegyzendő, hogy az ilyen kapcsolat **párhuzamos** műveleteket jelképez.

## Programozási nyelvek

Minden programot **egyetlen** nyelv ír le. Ez a nyelv, amely a program létrehozásakor van kiválasztva, később nem változtatható meg. Az **FBD** diagramok azonban részben tartalmazhatnak **LD**-t, az **LD** diagramok pedig tartalmazhatnak funkcióblokk meghívásokat. A rendelkezésre álló grafikus nyelvek: az **SFC** (Szekvenciális funkciódiagram), az **FC** (Blokkdiagram), az **FBD** (Funkcionális blokkdiagram), és az **LD** (Létradiagram). A rendelkezésre álló literális nyelvek: az **ST** (Strukturált szöveg), és az **IL** (Utasításlista). Az **SFC** és **FC** nyelvek a szekvenciális szekció fő- és gyermek programjaihoz vannak fenntartva. Mindegyik program nyelve egy ikonnal van jelezve a program neve mellett a Programkezelő ablakban. Az alábbiakban az egyes nyelvek ikonjai láthatók:

	SFC.....Szekvenciális funkciódiagram
	FC .....Blokkdiagram
	FBD.....Funkcionális blokkdiagram
	LD .....Létradiagram (a Gyors LD szerkesztővel beírva)

	ST .....	Strukturált szöveg
	IL .....	Utasításlista

### A.3.2 Programokkal való munkavégzés

A „**Fájl**” menü a programok létrehozására, frissítésére, vagy módosítására szolgáló összes parancsot csoportosítja. Ez az alkalmazás programok tartalmának beadására szolgáló megfelelő szerkesztők indítását is végzi.



#### **Egy új program létrehozása**

A „**Fájl**” menü „**Új**” funkciója felső szintű, gyermek-, vagy apa programok létrehozását teszi lehetővé az egyes program szekciókban. A beírandó első információ az új program neve, az elnevezési szabályoknak megfelelően:

- egy név maximális hossza **8** karakter
- az első karakternek **betűnek** kell lennie
- a többi karakternek **betűnek**, **számjegynek**, vagy aláhúzás („\_”) karakternek kell lennie
- egy program elnevezése nem érzékeny a kisbetű-nagybetű beállításra

Ezután az új program leírására kiválasztott szerkesztő nyelvet kell kiválasztani:

SFC .....	Szekvenciális funkciódiagram
FC .....	Blokkdiagram
FBD .....	Funkcionális blokkdiagram (részben tartalmazhat LD-t)
LD .....	Létradiagram, a Gyors LD szerkesztővel beírva
ST .....	Strukturált szöveg
IL .....	Utasításlista

Végül a program végrehajtásához egy végrehajtási stílust kell kiválasztani:

Kezdés .....	a „Kezdés” szekció felső szintje
Szekvenciális .....	a „Szekvenciális” szekció felső szintje
Befejezés .....	a „Befejezés” szekció felső szintje
Funkció .....	a „Funkciók” szekcióban
Funkcióblokk .....	a „Funkcióblokkok” szekcióban
... Gyermek .....	egy létező program SFC vagy FC gyermeke

Az első öt lehetőség valamelyikének kiválasztásával a program egy **Kezdés**, **Befejezés**, **Szekvenciális**, **Funkciók**, vagy **Funkcióblokkok** szekció felső szintjére van téve. Ez utóbbi kiválasztása azt jelzi, hogy az új program egy **SFC** gyermek program, vagy egy **FC** alprogram, vagy egy alprogram. Ne feledje, hogy egy felső szintű szekvenciális programot az **SFC** vagy **FC** nyelvben kell leírni, és hogy az **SFC** és **FC** nyelvek nem használhatók ciklikus programokhoz és azok alprogramjaihoz.



#### **Megjegyzések beírása az egyes programokhoz**

Az ISaGRAF lehetővé teszi a projekt minden programjához egy leíró szöveg csatolását. Ez a megjegyzés szöveg a program neve mellett kisebb karakterfonttal írva jelenik meg. A kiválasztott programhoz csatolt megjegyzés beírásához vagy

annak megváltoztatásához a „**Fájl / Program megjegyzés szöveg**” parancsot kell használni.



### **Egy program tartalmának szerkesztése**

Ez a parancs egy program tartalmának a módosítását teszi lehetővé. A program bevitelére használt szerkesztő az ahhoz a programhoz kiválasztott nyelvtől függ. A program szerkesztése egyedi ablakokban történik, így párhuzamos ablakokkal lehetőség van egyszerre egynél több program szerkesztésére is. Az **ENTER** gomb megnyomása lehetővé teszi a kiemelt program szerkesztését. A felhasználó a programot a nevére való dupla rákattintással is szerkesztheti.



### **A „napló” fájl szerkesztése**

Minden programhoz csatolva van egy **napló fájl**. Ez egy szövegfájl, amely a program élettartama során a benne elvégzett módosításokról szóló feljegyzéseket tartalmazza. A naplófájl bármikor szerkeszthető, szabadon módosítható, vagy kinyomtatható. A program forráskódjának módosítására használt szerkesztő elhagyásakor automatikusan megnyílik egy ablak, a naplólistához szolgáló megjegyzések beírásához. Ezek a megjegyzések a pontos dátum és időpont feltüntetésével kerülnek a naplófájlba beillesztésre.



### **A változósztár**

A felhasználó a „**Fájl / Szótár**” parancs segítségével a szótárszerkesztőt futtathatja, amelyben a projekthez deklarált változók találhatók. A változók lehetnek globálisak (amelyeket a projektben levő valamennyi program ismer), vagy lehetnek a kiválasztott program számára lokálisak. A szótárszerkesztő **definiált szavak** deklarálására is használható, amelyek egy program kódjában egy név vagy egy kifejezés lecserélésére használt szemantikus álnévek.



### **Egy funkció, alprogram, vagy funkcióblokk paramétere**

A „**Fájl / Paraméterek**” parancssal a felhasználó a kiválasztott alprogram, funkció, vagy funkcióblokk meghívási és visszatérési paramétereit definiálhatja. Ez a parancs hatástalan akkor, ha a Programkezelő ablakban a „**Kezdés**” vagy „**Befejezés**” szekció fő programja, vagy egy SFC program van kiválasztva.

Az alprogramoknak, funkcióknak, vagy funkcióblokkoknak maximum **32** paramétere (input vagy output) lehet. Egy funkciónak vagy alprogramnak mindig egy (és csak egyetlen) visszatérési paramétere van, amelynek a neve a funkció nevével megegyező kell, hogy legyen annak érdekében, hogy eleget tegyen az ST nyelv írási konvencióinak.

Az ablak bal felső oldalán látható lista a paramétereket mutatja, az alábbi meghívási modell szerinti sorrendben: először a meghívó paraméterek, utoljára pedig a visszatérési paraméterek. Az ablak alsó része a listában pillanatnyilag kiválasztott paraméter részletes leírását mutatja. Egy paraméterhez bármelyik ISaGRAF adat típus használható. A visszatérési paramétereknek a meghívó paraméterek után kell elhelyezkedniük a listában. A paraméterek elnevezéseinek a következő szabályoknak kell eleget tenniük:

- a név hossza nem haladhatja meg a 16 karaktert
- az első karakternek betűnek kell lennie
- a többi karakternek betűnek, számjegynek, vagy aláhúzás karakternek kell lennie

- az elnevezés nem érzékeny a kisbetű/nagybetű beállításra

A „**Beillesztés**” parancs egy új paraméternek a kiválasztott paraméter elé történő beillesztésére szolgál. A „**Törlés**” parancs a kiválasztott paraméter törlésére szolgál. Az „**Elrendezés**” parancs automatikusan átrendezi (sorrendbe állítja) a paramétereket úgy, hogy a visszatérési paraméterek a lista végére kerülnek.

## **Egy program áthelyezése a hierarchikus faszerkezetben**

A „**Fájl**” menü „**Átnevezés/áthelyezés**” parancsa egy program nevének megváltoztatására, vagy annak a hierarchikus faszerkezetben egy más helyre történő áthelyezésére használatos. A meglévő program leíró nyelve azonban nem változtatható meg. Ennek a parancsnak a futtatásakor a program létrehozására használt ablak kinyílik, és az összes mező a kiválasztott program attribútumaival töltődik fel. Egy program neve módosítható, és egy másik szekciója illetve szülő programja a hierarchikus faszerkezetben történő áthelyezéshez kiválasztható.

A „**Fájl**” menü „**Programok átrendezése**” menüje az ugyanolyan szintű programok listája és az apa közötti kimondott sorrend megadására szolgál. Ha a kiválasztott program a felső szinten van, akkor a parancs a kiválasztott szekció felső szintű programjainak az átrendezését végzi el. Ha a kiválasztott program alsóbb szinten van, akkor a parancs csak azokat az SFC gyermek- és alprogramokat rendezi el, amelyeknek ugyanaz az apja, mint a kiválasztott programé. Amikor ki van nyitva a „**Programok elrendezése**” párbeszédablak, akkor ki kell választani az áthelyezni kívánt programot, és annak listában történő mozgatásához meg kell nyomni a megfelelő „**Fel**”, vagy „**Le**” gombot a listában.



## **Programok másolása**

Egy programról készített másolat elkészítéséhez a programlistáról válassza ki a forrásprogramot, és futtassa a „**Fájl / Másolás**” parancsot. Ennek a parancsnak a futtatásakor a program létrehozására használt ablak kinyílik, és az összes mező a kiválasztott program attribútumaival töltődik fel. Írja be a rendeltetési program nevét, valamint a hierarchikus faszerkezet szekcióiban betöltött helyét. Ha a rendeltetési program nem létezik, akkor az automatikusan létrehozásra kerül a megadott helyen. Ha a rendeltetési program már létezik, akkor az felülíródik. A programmal az összes lokális deklaráció és definiált szó átmásolásra kerül. A rendeltetési program leíró nyelvének meg kell egyeznie a forrásprogramhoz használt nyelvvel. Az „**OK**” gomb megnyomása végrehajtja a program másolását.

A „**Fájl**” menü „**Másolás másik projektbe**” parancsa a kiválasztott programot egy másik projektbe másolja, ugyanazzal a névvel. A kiválasztott program gyermek SFC programjai és alprogramjai azzal együtt kimásolhatók. A cél projektben a kiválasztott program és gyermekprogramja neveinek nem szabad léteznie. Ezzel a parancssal a programok nem írhatók felül. A programmal az összes lokális deklaráció és definiált szó átmásolásra kerül.



## **Programok törlése**

Egy program törléséhez a programlistáról válassza ki a programot, és futtassa a „**Fájl / Törlés**” parancsot. Egy gyermek- vagy alprogramokkal rendelkező program nem törölhető. Egy gyermek- vagy alprogramokkal rendelkező program törléséhez

először a gyermek- vagy alprogramot kell törölni. A programmal együtt az összes lokális deklaráció és definiált szó szintén törlésre kerül.



### **Funkció vagy funkcióblokk importálása könyvtárból**

Az „**Eszközök / Importálás könyvtárból**” parancs egy, a könyvtárban leírt, IEC nyelveken írt funkcióknak vagy funkcióblokkoknak a megnyitott projekt „**Funkciók**” vagy „**Funkcióblokkok**” szekciójába történő bemásolására szolgál. Az importált funkcióval a hozzá csatolt lokális változók és definiált szavak átmásolásra kerülnek. Amikor egy funkció sikeresen importálásra került a könyvtárból, akkor az a „**Fájl / Átnevezés/Áthelyezés**” parancs alkalmazásával a hierarchikus faszervezetben egy másik szekcióba illetve helyre áthelyezhető. A nevek ütközésének elkerülése érdekében az importált funkciót vagy funkciónevet a projekt területre történő importáláskor át kell nevezni. Egy funkció esetében ne feledje a visszatérési paramétert is átnevezni.



### **Funkció vagy funkcióblokk exportálása könyvtárba**

Az „**Eszközök / Exportálás könyvtárba**” parancs a (megnyitott projekt) „**Funkciók**” vagy „**Funkcióblokkok**” szekciójának a megfelelő könyvtárba küldésére szolgál. Az exportált funkcióval a hozzá csatolt lokális változók és definiált szavak átmásolásra kerülnek. Az exportált funkciót vagy funkcióblokkot az ISaGRAF Könyvtárrendezőből újra fel lehet fordítani (megerősíteni) annak biztosítása érdekében, hogy az egy könyvtár környezetben használható. A könyvtár funkciói és funkcióblokkjai nem használhatnak globális változókat.

## **A.3.3 A kódgeneráló eszközök futtatása**

A „**Létrehozás**” menü parancsai a kódgenerátor futtatására, és az alkalmazás kód létrehozásakor a választási lehetőségek, valamint további adatok bevitelére használhatók. Ezekkel az eszközökkel kapcsolatos további információk ennek a dokumentumnak a „**A kódgenerátor használata**” című fejezetében találhatók.



### **Az alkalmazás kód létrehozása**

A „**Létrehozás**” parancs elindítja a projekt kódjának generálását. Ennek a parancsnak a futtatása előtt megfelelő módon be kell állítani a célkód generálásának beállítási lehetőségeit. A célkód generálása előtt minden, még meg nem erősített programot ellenőrizni kell annak érdekében, hogy előkerüljenek az esetleges szintaxishibák. Az ISaGRAF egy növekményes fordítót tartalmaz, amely a már lefordított programokat nem fordítja újra.



### **A kiválasztott program megerősítése**

A felhasználó a „**Megerősítés**” parancssal a listában jelenleg kiválasztott program szintaxisának megerősítését végezheti el. Ha egy program hiba érzékelése nélkül megerősítést kapott, akkor a kódgenerálás közben nem lesz újra megerősítve mindaddig, amíg tartalma illetve a tőle függő definiált szavak vagy változók meg nem változnak.

 **Egy módosítás szimulációja**

Az „Érintés” parancs az összes program módosítását szimulálja, hogy a következő kódgeneráláskor azok mindegyike megerősítést kapjon.

**Alkalmazás futtatási beállítások**

Ez a parancs egy párbeszédablakot nyit ki, amelybe az alkalmazás végrehajtás fő futtatási paramétereit vannak beírva. Ezek közé tartozik a ciklusidőzítés programozás, a futtatási hibakezelés, az indítási üzemmód, és a megtartott változók hardver kivitelezése. Az ezzel a paranccsal kapcsolatos további magyarázatok ennek a dokumentumnak „A kódgenerátor használata” című fejezetében találhatók.

 **Fordítási beállítások**

Ez a parancs az ISaGRAF kódgenerátor által a cél kód felépítésére és optimalizálására használt lehetőségek beállítására szolgál. Az ezzel a paranccsal kapcsolatos további magyarázatok ennek a dokumentumnak „A kódgenerátor használata” című fejezetében találhatók.

 **Erőforrások definiálása**

Az „erőforrás” egy felhasználó által definiált adat (pl. egy fájl), amelyet a célkóddal egyesíteni kell, hogy azzal letölthető legyen. Az erőforrás definíciós fájl formátumával kapcsolatos további magyarázatok ennek a dokumentumnak „A kódgenerátor használata” című fejezetében találhatók.

**A.3.4 Egyéb ISaGRAF eszközök**

A „Projekt” menü a kiválasztott projekthez az ISaGRAF eszközöket futtató parancsokat csoportosítja. Ezekkel az eszközökkel kapcsolatban ennek a kézikönyvnek az azokra vonatkozó fejezeteiben található bővebb információ.

**I/O változók összekapcsolása**

Az „IO csatlakozás” parancs az ISaGRAF I/O változó-csatlakozás szerkesztőt futtatja. Ez az eszköz a projekt szótárban deklarált I/O változók és a megfelelő I/O hardver közötti kapcsolatot hozza létre.

**A kereszthivatkozás szerkesztő futtatása**

A felhasználó a „Kereszthivatkozások” parancs segítségével a projekt kereszthivatkozásainak kiszámítását, megtekintését, vagy nyomtatását végezheti el. A kereszthivatkozások az összes változó előfordulását mutatják meg a felhasználónak a programok forráskódjában, az egész projektre kiterjedően. Ez a funkció nagyon hasznos egy változóhoz vagy bármely globális erőforráshoz való hozzáférés érzékelésére, vagy egy globális változónak a forráskódban levő összes előfordulásának felsorolására.

 **A projektleíró bevitele**

A „Projektleíró” parancs a projekt szöveges leírójának szerkesztésére szolgál. Ez a dokumentum teljes mértékben azonosítja a projektet a projektlistán levő többi

projekt közül. A projektleíró a projekt élettartama során tett megjegyzések rögzítésére is szolgálhat. A projektleíró a Projektkezelő ablakban van megjelenítve.



### **Egy teljes dokumentum kinyomtatása**

A felhasználó a „Projekt dokumentum nyomtatása” parancs segítségével a kiválasztott projekthez egy komplett dokumentumot állíthat össze és nyomtathat ki. Ez a dokumentum a kiválasztott projekt bármely komponensét (program, változó, paraméterek, stb.) csoportosíthatja. Egy specifikus (nem komplett) dokumentum felépítéséhez a felhasználónak csak annak tartalomjegyzékét kell definiálnia.



### **A módosítások előzményei**

Ez a parancs egy párbeszédablakot nyit ki, amelyben a projekt módosításainak előzményei jelennek meg. Az ezzel a paranccsal kapcsolatos további magyarázatok ennek a dokumentumnak „Projektek kezelése” című fejezetében találhatók.

## **A.3.5 Parancsok hozzáadása az Eszközök menühöz**

Az ISaGRAF lehetőséget biztosít arra, hogy az „Eszközök” menübe további parancsokat illesszenek be. A beillesztendő új, felhasználó által definiált parancsok az „**ISAWIN\COMISA.MNU**” szövegfájlban vannak leírva. Maximum 10 parancsot lehet beilleszteni. A megjegyzések bármelyik sorra beilleszthetők, és azoknak „,” karakterrel kell kezdődniük. Mindegyik parancs két sornyi szövegben van leírva, a következő szintaxisnak megfelelően:

M=menu\_string  
C=command\_line

A menü karaktorsor az „Eszközök” menüben megjelenítendő szöveg. A parancs sor bármely MS-DOS- vagy Windows- végrehajtható, amely kiegészíthető argumentumokkal. A parancs sorban „%A” karaktereket lehet használni a nyitott projekt nevének a behelyettesítésére, illetve „%P” karaktereket a kiválasztott program nevének a behelyettesítésére. A következő példa a „Notepad”-ot futtatja a kiválasztott program szerkesztésére (ST és IL programokkal használandó):

M=Szerkesztés a Jegyzetkönyvvel  
C=Notepad.exe \isawin\apl\%A\%P.lsf

## **A.3.6 Az alkalmazás szimulációja és hibakeresése**

A „Hibakeresés” menü parancsa az ISaGRAF grafikus hibakereső futtatására szolgál, akár szimulációs, akár valódi csatlakoztatott módban.



### **Szimuláció**

A „Szimuláció” parancs a hibakeresőt szimulációs módban nyitja meg. Ebben a módban egy másik, szimulátornak nevezett ablak nyílik ki. Ez a parancs nagyon hasznos egy alkalmazás tesztelésére olyankor, amikor a célgép nem áll rendelkezésre. A szimulátor elindítása bezárja a Programkezelő ablakot. A Programkezelő ablak aztán a hibakeresés módban ismét kinyílik, miután mind a

hibakereső, mind a szimuláció ablakok kinyíltak. A szimulátor nem indítható el, ha a célkód nem lett generálva. A szimulátor nem indítható el, ha gyermek ablakok (szerkesztők, kódgenerálás, I/O csatlakozás, stb.) vannak nyitva. Ezek mindegyikét be kell zárni ennek a parancsnak a futtatása előtt. Ez a parancs az ISaGRAF szerkesztők menüiből is elérhető.



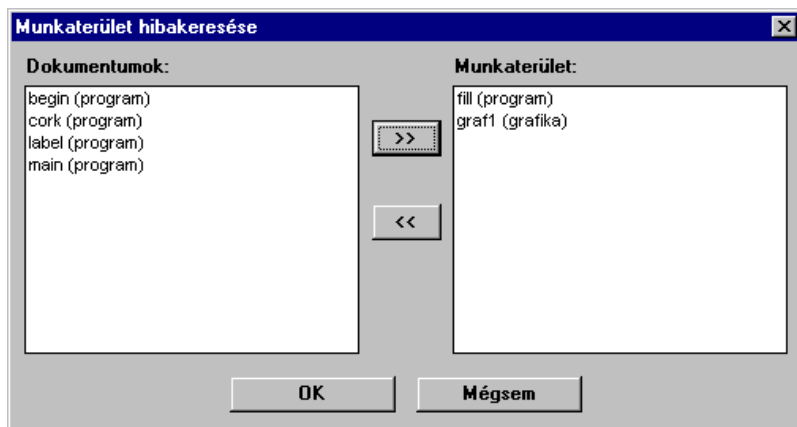
### **Tényleges hibakeresés**

A „**Hibakeresés**” parancs kinyitja a hibakereső fő ablakát, és bezárja a Programkezelő ablakot. A Programkezelő ablak aztán a hibakeresés módban ismét kinyílik, amint létrejött a hibakereső és a cél alkalmazás közötti kommunikáció. A szimulátor nem indítható el, ha a célkód nem lett generálva. A szimulátor nem indítható el, ha gyermek ablakok (szerkesztők, kódgenerálás, I/O csatlakozás, stb.) vannak nyitva. Ezek mindegyikét be kell zárni ennek a parancsnak a futtatása előtt. Ez a parancs az ISaGRAF szerkesztők menüiből is elérhető.



### **A hibakereső munkaterület előkészítése**

A „**Hibakereső / Munkaterület**” parancs lehetővé teszi a kiindulási munkaterülethez egy dokumentumlista definiálását. Ezek a dokumentumok lehetnek programok, Fényszóráó grafikák, illetve változólisták. A korábbi ISaGRAF verzióktól származó grafikák és idődiagram listák szintén a projekt dokumentumokkal vannak felsorolva. A kezdeti munkaterületen definiált dokumentumok automatikusan kinyílnak, amikor szimulációs Online figyelemmel kísérés van elindítva.



A párbeszédablak a bal oldalon a projekt meglévő dokumentumait, a jobb oldalon pedig a kiindulási munkaterülethez kiválasztott dokumentumokat mutatja. A dokumentumok egyik listáról a másikra történő áthelyezéséhez a „>>” és „<<” nyomógombokat kell használni. Mindegyik projekt saját dokumentum listával rendelkezik a kiindulási munkaterülethez.



## A kapcsolat beállítása

A „**Kapcsolat beállítása**” paranccsal a felhasználó a gazda PC-n levő hibakereső és a cél ISaGRAF rendszer közötti kommunikációhoz használt kapcsolat paramétereit definiálhatja.

A „**Szolgá szám**” a cél ISaGRAF rendszert vagy feladatot azonosítja. Ez **1** és **255** között lehet. A használt célrendszer kiszolgáló száma a cél beszállítójának kezelési útmutatójában található.

A „**Kommunikációs port**” az ISaGRAF workbench és a cél közötti hardver médiát jelenti. Ez lehet egy soros port neve, vagy „**Ethernet**”, rezervált TCP-IP kommunikáció, amely 1.1-es Verziójú „Winsock”-ot használ.

Az „**Idő túllépés**” a célrendszer számára a kommunikációs műveletekre adott, egy hibakereső kérdés vége és válaszána kezdete közötti időtartam. Ez az időtartam **ezredmásodpercekben** van kifejezve. A „Próbálkozások” mező a hibakereső által egy kommunikációs művelethez végrehajtott automatikus próbálkozások száma, mielőtt az kommunikációs hibát érzékelne.



## A soros kapcsolat beállítása

Amikor egy soros port (COM1...4) van kiválasztva, az egyéb soros kapcsolati paraméterek a „**Beállítás**” gombbal érhetők el.

Ezzel az adatátviteli sebesség, paritás, és formátum választható ki. Ha az „**Adatáramlás vezérléshez**” a „**hardver**” lett kiválasztva, akkor az ISaGRAF Workbench vezérli a CTS és DSR vonalakat a hardver kézfogás működéséhez az adatcserék során.



## Az Ethernet kapcsolat beállítása

Amikor kommunikációs portként „Ethernet” van kiválasztva, akkor a „**Beállítás**” gomb szolgál a TCP-IP kommunikációhoz az „Internet cím” és „Internet port” számának beírására.

Ezek a mezők a Kártyahely interfész által meghatározott standard formátumokat használják. A Workbench a TCP-IP kommunikációhoz az 1.1-es verziójú WINSOCK.DLL könyvtárat használja. Ennek a fájlnek a merevlemezen megfelelő módon telepítve jelen kell lennie. Amennyiben az ISaGRAF cél futtatásakor az külön nincs specifikálva, akkor az „**1100**” az alapértelmezett port szám.








## A.4 Az SFC szerkesztő használata

Az SFC nyelv egy szekvenciális folyamat műveleteinek a leírására szolgál. Ez egy folyamat különböző lépéseinek és az aktív lépések változtatását lehetővé tevő feltételeknek a jelzésére egyszerű grafikus jeleket alkalmaz. Az SFC program az ISaGRAF grafikus SFC szerkesztő segítségével vihető be. Az SFC az IEC 1131-3 szabvány magját képezi. A többi nyelvek általában a lépéseken belüli műveleteket és az átmenetek logikai feltételeit írják le. Az ISaGRAF grafikus SFC szerkesztő lehetővé teszi komplett SFC programok bevitelét. Ez egyesíti a grafikai és szövegszerkesztési képességeket, ezzel lehetővé téve mind az SFC diagram, mind a megfelelő műveletek és feltételek bevitelét.

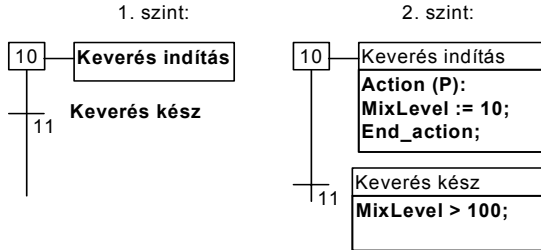
### A.4.1 A SFC nyelv fő témái

Az SFC nyelv szekvenciális folyamatok leírására szolgál. Ez a folyamat ciklust több, jól definiált, egymást követő **lépésre** (önálló szituációkra) osztja fel, amelyeket **átmenetek** választanak el egymástól. Az SFC nyelvvel kapcsolatos további részletes információk az ISaGRAF Nyelvi hivatkozás kézikönyvben találhatók.

Az SFC komponenseket **irányított vonalak** kötik egymáshoz. Egy vonal alapértelmezett iránya **felfelé**, vagy **lefelé** mutat. Az SFC diagram felépítéséhez használt alapvető grafikus komponensek a következők:

	.....Kezdeti lépés
	.....Lépés
	.....Átmenet
	.....Ugrás egy lépésre
	.....Makró lépés
	.....Makró kezdeti lépés
	.....Makró befejező lépés

Az SFC programozás általában két különböző szintre van szétválasztva: Az **1. szint** a grafikus diagramot, a lépések és átmenetek referenciaszámait, valamint a lépésekhez és átmenetekhez csatolt megjegyzéseket mutatja. A **2. szint** a lépéseken belüli műveletek, vagy az átmenetekhez csatolt feltételek **ST** vagy **IL** programozása. A műveletek vagy feltételek utalhatnak más nyelveken (**FBD**, **LD**, **ST** vagy **IL**) írt **alprogramokra**. Az alábbiakban 1. szintű és 2. szintű programozás példái láthatók:

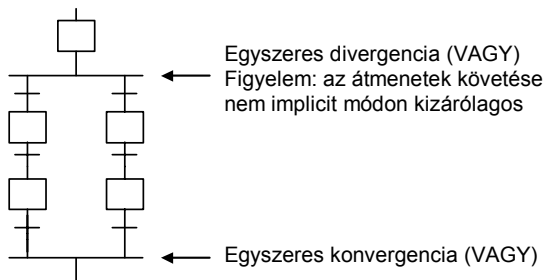


Egy lépés 2. szintű programozása egy szövegszerkesztőben van beírva. Ez tartalmazhat ST-ben vagy IL-ben programozott művelet blokkokat. Egy átmenet 2. szintű programozása vagy az IL illetve ST szöveges nyelvekben, vagy a Gyors LD szerkesztővel írható be.

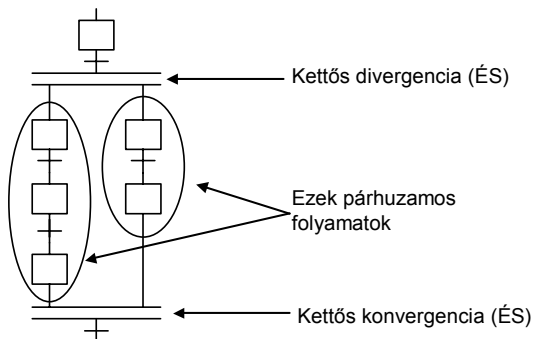


## Divergenciák és konvergenciák

A divergenciák és konvergenciák a lépések és átmenetek közötti **többszörös kapcsolatok** jelölésére szolgálnak. Az egyszeres divergenciák vagy konvergenciák különböző **inkluzív** lehetőségeket jeleznek a folyamat különböző részletei között.

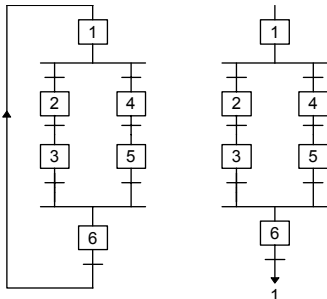


A kettős divergenciák **párhuzamos** folyamatokat jelölnek.



## Ugrás egy lépésre

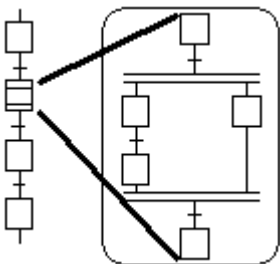
Az SFC szerkesztő csak **felfelé** vagy **lefelé** mutató irányokban engedi meg a kapcsolatok megrajzolását. Egy lépésre történő **ugrás** a diagram felső részéhez való kapcsolat jelölésére használható. A következő diagramok egyenértékűek:



Egy átmenetre történő ugrás tilos, és azt explicit módon egy kettős (ÉS) konvergenciaként kell jelezni.

## Makró lépések

Egy makró lépés a lépések és átmenetek **egyedülálló** csoportjának **egydi** jelölése. A makró lépés egy **kezdő lépéssel** kezdődik, és egy **befejező lépéssel** ér véget.



Egy makró lépés részletes jelölését ugyanabban az SFC programban kell leírni. A makró lépés szimbólum **referenciaszámának** meg kell egyeznie a makró kezdeti lépésének referenciaszámával. Egy makró lépés leírása tartalmazhat egy másik makró lépést.

### A.4.2 Az SFC diagram bevitele

Egy SFC diagram megrajzolásához egyszerűen csak a diagram jelentős komponenseit kell bevezetni. Az SFC szerkesztő minden, két elemet összekötő egyszerű vonalat automatikusan megrajzol. Egy SFC komponensek a diagramba helyezéséhez a kiválasztást a megfelelő helyre kell vinni, és a szerkesztő

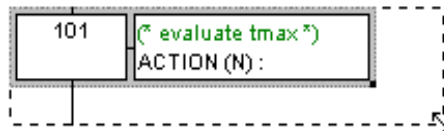
eszközsorából ki kell választani a komponens típusát. A szimbólum a pillanatnyi helyre beillesztődik. A következő billentyűzet gombokat szintén lehet alkalmazni:

F2:	.....Egy kezdeti lépés beillesztése
F3:	.....Egy egyszeres lépés beillesztése
F4:	.....Egy átmenet beillesztése
F5:	.....Egy lépésre való ugrás beillesztése
F6:  F7:	.....Egy VAGY divergencia illetve konvergencia beillesztése / Elágazások hozzáadása
*F6:  *F7:	.....Egy ÉS divergencia illetve konvergencia beillesztése / Elágazások hozzáadása
F8:	.....Egy makró lépés beillesztése
F9:  *F9:	.....Kezdeti vagy befejező lépés beillesztése egy makró lépés fő részéhez

(A „” szimbólum a SHIFT gomb használatát jelzi)

A **szerkesztő rács mátrix cellákat** mutat. A szerkesztő beállítási lehetőségei segítségével a felhasználó a diagram bevitele közben megjelenítheti, illetve elrejtetheti a szerkesztőrácsot. A szerkesztőrács az SFC diagram kezdeti elrendezéséhez vagy a diagram egy részének kiválasztásához nagyon hasznos. A rács megjelenítésére illetve elrejtésére a „**Beállítási lehetőségek / Elrendezés**” parancsot kell használni.

Az ISaGRAF SFC szerkesztő mindig kijelzi a pillanatnyi helyet a mátrixban. A fókuszolt cella szürke színnel van jelölve. Az annak jobb alsó sarkában levő kis négyzet a cellák szabad átméretezésére használható. Ilyen módon a cellák X/Y aránya is megváltoztatható.





## **Egy divergencia vagy konvergencia létrehozása**



A divergenciákat és konvergenciákat mindig **balról jobbra** kell megrajzolni. Egy divergencia vagy konvergencia megrajzolásához annak **bal oldali elágazásait** a diagram területre kell helyezni. A rajz típusa (egyszeres vagy kétszeres) ezek közül a gombok közül valamelyiknek az eszközsorról való kiválasztásával történik.

F6:  F7:	.....Egy VAGY divergencia illetve konvergencia beillesztése / Elágazások hozzáadása
*F6:  *F7:	.....Egy ÉS divergencia illetve konvergencia beillesztése / Elágazások hozzáadása

## Elágazások hozzáadása divergenciákhoz

Mindegyik **kiegészítő elágazás** indulási és **befejezési** pozíciója ezeknek az eszközösről levő gomboknak a használatával helyezhető a divergencia vagy konvergencia vonalra. Az új elágazások beillesztése előtt a divergencia vagy konvergencia bal sarkának jelen kell lennie. A jobb sarkok stílusa (egyszeres vagy kettős) megegyezik a fő bal sarokéval. Jobb sarkok nem helyezhetők el, ha a fő bal sarok nem lett hozzáadva.

  .....Egy VAGY divergencia illetve konvergencia beillesztése / Elágazások hozzáadása

  .....Egy ÉS divergencia illetve konvergencia beillesztése / Elágazások hozzáadása



### Egy makró lépés beillesztése

Ez a gomb egy makró lépésnek a fő diagramba történő beillesztésére szolgál. A makró lépés fő részének ugyanabban az SFC programban máshol bevittként meg kell lennie.



### Egy makró lépés fő része

A makró lépéseket ugyanabban az SFC programban kell leírni, mint a fő diagramot. Egy makró lépésnek egy **kezdeti lépéssel** kell kezdődnie, és egy **befejező lépéssel** kell befejeződnie. A makró telepítéseként leírt aldiagramnak **önálló**nak kell lennie. A makró kezdeti lépésnek ugyanazzal a **referenciával** kell rendelkeznie, mint a fő elágazás makró lépés szimbólumának.

## A.4.3 Egy meglevő SFC diagrammal történő munkavégzés

A diagramban egy négyzet alakú terület kiválasztására akár az egeret, akár a billentyűzet nyíl billentyűit lehet használni. Az egész kiválasztott terület szürke színnel van jelölve. A „**Szerkesztés**” menü parancsai a következőkre használhatók:



### Kivágás / másolás / törlés / beillesztés parancsok

A „**Szerkesztés**” menüben a következő parancsok állnak rendelkezésre, amikor a szerkesztő eszközsorában ki van választva a „**nyíl**” gomb:

Kivágás .....A kiválasztott négyzet áthelyezése a képernyőről az SFC vágólapra

Másolás .....A kiválasztott négyzet másolása a képernyőről az SFC vágólapra

Törlés .....A kiválasztott négyzet kiürítése (törlése)

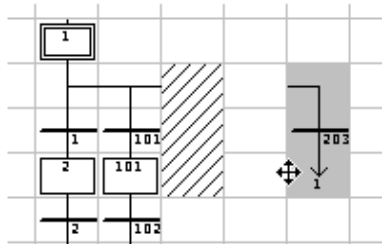
Beillesztés .....Az SFC vágólap tartalmának beillesztése a pillanatnyi helyre

A „**Szerkesztés / Beillesztés**” az SFC vágólap tartalmát a képernyőre másolja. A Másolás / Beillesztés parancsok mind az SFC diagramokkal, mind a 2. szintű lépés/átmenet programozással működik. Lehetséges egy programban levő diagram másolása, és egy másik SFC programba történő beillesztése is. Az elemek a pillanatnyilag kiválasztott pozíció elé vannak beillesztve.



## Elemek áthelyezése

Ha az SFC diagramban SFC elemek lettek kiválasztva, akkor azokat az egérrel vonszolva a diagram más helyére át lehet vinni. A kiválasztás vonszolása közben a kiválasztott elemek eredeti helye vonalkázással van bejelölve.



Az áthelyezett elemek új rendetési helyének üresnek kell lennie. SFC szimbólumok áthelyezése közben nem lehetséges a beillesztés.



## Lépések és átmenetek átszámozása

Minden lépést vagy átmenetet egy logikai szám azonosít be az SFC diagramban. A felhasználó a „**Szerkesztés / Átszámozás**” parancs segítségével sorrendben egymás után álló referenciaszámokat állíthat fel automatikusan a pillanatnyilag szerkesztett SFC program bármelyik lépéshez és átmenetéhez. Ha egy lépés száma meg lett változtatva, akkor valamennyi arra a lépésre történő ugrás automatikusan frissítésre kerül az új referenciaszámmal. (Ez a makró lépésekre és kezdeti lépésekre is érvényes)



## Közvetlen hozzáférés egy lépéshez vagy egy átmenethez

A felhasználó a „**Szerkesztés / Ugrás**” parancs segítségével egy meglevő lépéshez vagy átmenethez férhet hozzá. A görgetési pozíció automatikusan átállítódik úgy, hogy a lépés vagy átmenet láthatóvá váljon.



## Szövegek keresése és lecserélése

A „**Szerkesztés / Keresés és lecserélés**” parancs szöveg karaktersorok megkeresésére vagy lecserélésére használható a teljes programban (minden lépés és átmenet). A Keresés/Lecserélés párbeszédablak egy keresendő szöveg bevitelére és a 2. szintű programozás azon részének a megnyitására szolgál, ahol a szöveg található.

### A.4.4 A 2. szintű programozás bevitel

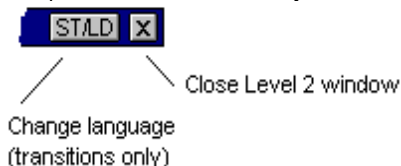
A 2. szintű szöveg beviteléhez a felhasználónak duplán rá kell kattintania a lépés vagy átmenet szimbólumára. A 2. szintű programozás az SFC ablak jobb oldalán jelenik meg. Az SFC diagram és a 2. szintű programozás közötti elválasztó vonal szabadon elmozgatható.

Egyidejűleg egy vagy kettő 2. szintű terület nyitható ki. A következő parancsok a billentyűzetten, az egérrel, vagy a „Szerkesztés” menün állnak rendelkezésre:

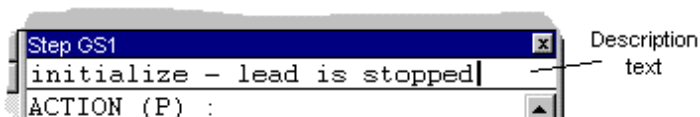
Billentyűzet	Egér	„Szerkesztés” menü	
Megnyitás a legutóbbi alapértelmezett ablakban:	Enter	Dupla kattintás	2. szint szerkesztése
Megnyitás külön ablakban:	Ctrl+Enter	Ctrl + Dupla kattintás	2. szint szerkesztése külön ablakban

Amikor két 2. szintű ablak látható, a közöttük levő elválasztó szabadon mozgatható. A 2. szintű címsortól jobbra levő gomb egy 2. szintű ablak bezárására szolgál.

A 2. szintű programozás alapértelmezett nyelve az **ST** (Strukturált Szöveg). Az átmenetekhez a 2. számú programozás a **Gyors LD** szerkesztővel is bevitethető. Az aktív nyelv átváltásához a 2. szintű címsorban levő „**ST/LD**” gombot kell használni. Ez a parancs csak akkor érvényes, ha a 2. szintű programozási ablak üres.



A 2. szintű ablak felső részén egy egyetlen sort tartalmazó szerkesztőablak jelenik meg. Ez egy rövid leíró szöveg beírására szolgál. Ez a szöveg IEC megjegyzésként fog megjelenni az SFC szimbólumok rajzában. Ez nagyon hasznos, és más parancsok, pl. az „Ugrás” is használják, valamint az SFC kinyomtatásokban is használatos az SFC lépések és átmenetek dokumentálására.



A „**Beállítások / Frissítés**” parancs bármikor használható, amikor a fő SFC diagram módosított 2. szintű programokkal való frissítésére 2. szintű ablakok vannak nyitva.



### **Egy változó név beillesztése**

Szöveges nyelvben történő programozáskor egy, a projekt szótárban deklarált változó kiválasztásához és annak a beszúrási jel pillanatnyi helyére történő beillesztéséhez nyomja meg ezt a gombot. Gyors LD-ben történő programozáskor ez a gomb a kiválasztott kontakthoz vagy blokk I/O paraméterhez csatlakoztatandó változó kiválasztására szolgál.



### **Egy Impulzus műveletblokk beillesztése a lépésbe**

Egy lépés 2. szintjének programozásakor ennek a gombnak a megnyomása egy Impulzus műveletblokk mintasablonjának beillesztésére szolgál a beszúrási jel pillanatnyi helyénél. Az alábbiakban egy Impulzus műveletblokk formátuma látható:

```

Action (P) :
    ST utasítás;
    ...
End_Action;

```

Az Impulzus műveletek olyan utasítások, amelyek csak egyszer kerülnek végrehajtásra, amikor a lépés aktívá válik. Az SFC programozással kapcsolatos további részletes információk az ISaGRAF nyelvi hivatkozásában találhatók.



### **Egy Nem tárolt műveletblokk beillesztése a lépésbe**

Egy lépés 2. szintjének programozásakor ennek a gombnak a megnyomása egy Nem tárolt műveletblokk mintasablonjának beillesztésére szolgál a beszúrási jel pillanatnyi helyénél. Az alábbiakban egy Nem tárolt műveletblokk formátuma látható:

```

Action (N) :
    ST utasítás;
    ...
End_Action;

```

A Nem tárolt műveletek olyan utasítások, amelyek minden PLC cikluson végrehajtásra kerülnek, amikor a lépés aktív. Az SFC programozással kapcsolatos további részletes információk az ISaGRAF nyelvi hivatkozásában találhatók.



### **Új P0 és P1 művelet minősítők**

Az ISaGRAF támogatja az új **P0** és **P1** művelet minősítőket. Egy lépés 2. szintjének programozásakor ezeknek a gomboknak a megnyomása egy P0 vagy P1 művelet blokk mintasablonjának beillesztésére szolgál a beszúrási jel pillanatnyi helyénél. Az alábbiakban az ilyen blokkok formátuma látható:

<pre> Action (P0) :     ST utasítás;     ... End_Action; </pre>	<pre> Action (P1) :     ST utasítás;     ... End_Action; </pre>
---	---

A P1 műveletek olyan utasítások, amelyek csak egyszer kerülnek végrehajtásra, amikor a lépés aktívvá válik (ugyanúgy, mint az Impulzus). A P0 műveletek olyan utasítások, amelyek csak egyszer kerülnek végrehajtásra, amikor a lépés inaktívvá válik. Az SFC programozással kapcsolatos további részletes információk az ISaGRAF nyelvi hivatkozásában találhatók.

## Logikai műveletek

Egyéb szöveg-szemantikák is rendelkezésre állnak, amelyek egy logikai változóra közvetlenül hatnak, a lépés műveletnek megfelelően. Az ilyen műveletek a **lépés aktivitási jelnek** egy belső vagy output logikai változóra történő csatolásából állnak. Az alapvető logikai műveletek szintaxisa a következő:

<boolean_variable> (N) ;	a lépés művelet jelet kijelöli a változóhoz
<boolean_variable> ;	ugyanaz a hatása (az N attribútum nem kötelező)
/ <boolean_variable> ;	a lépés művelet jel tagadását jelöli ki a változóhoz

Egyéb lehetőségek is rendelkezésre állnak egy logikai változó beállításához vagy visszaállításához, amikor a lépés aktívvá válik. A beállítási és visszaállítási logikai műveletek szintaxisa a következő:

<boolean_variable> (S) ;	A változót IGAZ-ra állítja be, amikor a lépés aktivitás jele IGAZ-zá válik
<boolean_variable> (R) ;	A változót HAMIS-ra állítja vissza, amikor a lépés aktivitás jele IGAZ-zá válik

## SFC műveletek

Egy gyermek SFC program vezérléséhez egyéb szöveg szemantikák is rendelkezésre állnak. Az SFC művelet egy SFC gyermek szekvencia, amely a lépés aktivitási jelnek megfelelően van elindítva vagy leállítva. Egy SFC műveletnek lehet **N** (Nem tárolt), **S** (Beállított), vagy **R** (Visszaállított) minősítője. Az alapvető SFC műveletek szintaxisa a következő:

<child_program> (N);	elindítja a gyermek szekvenciát, amikor a lépés aktívvá válik, és leállítja a gyermek szekvenciát, amikor a lépés inaktívvá válik
<child_program>;	ugyanaz a hatása, mint az előző szintaxisnak (az N attribútum nem kötelező)
<child_program> (S);	elindítja a gyermek szekvenciát, amikor a lépés aktívvá válik – semmit nem tesz, amikor a lépés inaktívvá válik
<child_program> (R);	leállítja a gyermek szekvenciát, amikor a lépés aktívvá válik – semmit nem tesz, amikor a lépés inaktívvá válik

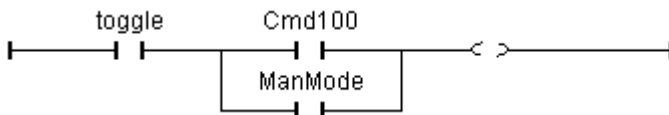
A műveletként specifikált SFC szekvenciának a jelenleg szerkesztett program létező **gyermek SFC programjának** kell lennie, amely az ISaGRAF programrendezővel lett létrehozva.

## ST-ben írt átmenetek

Egy átmenet 2. szintje egy logikai kifejezés. Ennek ST nyelven történő programozásához egyszerűen be kell írni a logikai feltételt, az ST szintaxisnak megfelelően. A kifejezés végére tetszés szerint beírható egy pontosvessző is.

## Gyors Létrában írt átmenetek

A Gyors LD szerkesztő egy átmenet 2. szintű feltételének programozására áll rendelkezésre. Ebben az esetben a diagram mindössze egyetlen létrafokból áll, egyetlen tekerccsel, amely az átmenetet képviseli. Az átmenet neve nincs megismételve a tekercs szimbólummal. Az alábbiakban egy Gyors LD-ben programozott átmenet feltétel példája látható:



A Gyors LD-ben történő programozáskor a kiválasztásnak a programozási logikai rácsban történő elmozdítására a billentyűzet nyíl gombjait, majd a szimbólumok beillesztéséhez a következő billentyűparancsokat kell használni:

- F2: .....egy kontakt beillesztése a kiválasztott szimbólum után / a létrafok elkezdése  
 F3: .....egy kontakt beillesztése a kiválasztott szimbólum elé  
 F4: .....egy kontakt beillesztése a kiválasztott szimbólummal párhuzamosan  
 F6: .....egy blokk beillesztése a kiválasztott szimbólum után  
 F7: .....egy blokk beillesztése a kiválasztott szimbólum elé  
 F8: .....egy blokk beillesztése a kiválasztott szimbólummal párhuzamosan

A funkcióbillentyűk megnyomása helyett a 2. szintű ablak alján levő funkcióbillentyű sorra is rá lehet kattintani.

Ha a kiválasztás egy kontakton vagy egy blokk I/O paraméteren van, akkor egy változó kiválasztásához illetve egy konstans érték beírásához a RETURN gombot kell megnyomni. Ha a kiválasztás egy funkcióblokkon van, akkor a funkcióblokk típusának kiválasztásához a RETURN gombot kell megnyomni. Ugyanez a hatás érhető el akkor is, ha egy szimbólumra duplán rákattint.

Ha egy kontakt van kiválasztva, akkor a kontakt típusának megváltoztatásához (direkt, tagadott, vagy impulzus detektálásos) a SZÓKÖZ billentyűt kell megnyomni. A Gyors LD lehetőségeivel kapcsolatos további információk ennek a dokumentumnak „A Gyors LD szerkesztő használata” című fejezetében találhatók.

### A.4.5 Az SFC gyűjtemény használata

Az ISaGRAF SFC szerkesztő egy SFC gyűjteményt kezel: ez bármely SFC diagramba beilleszthető SFC szerkezetek gyűjteménye. Az SFC gyűjtemény elemeibe tetszés szerint be lehet építeni a lépések és átmenetek 2. szintű programozását. Az „**Eszközök**” menü következő parancsait kell használni:

**Másolás az SFC gyűjteménybe** .....a kiválasztott elemeket az SFC  
.....gyűjteménybe másolja

**Beillesztés az SFC gyűjteményből**.....a pillanatnyi helyre egy elemet illeszt be az  
.....SFC gyűjteményből

Az SFC gyűjteménybe történő másoláskor (azaz egy új SFC gyűjtemény elem létrehozásakor) tetszés szerint kérhető a kiválasztott SFC szimbólumok 2. szintű programozásának beépítése.

## A.5 A Blokkdiagram szerkesztő használata

Az ISaGRAF grafikus Blokkdiagram szerkesztő lehetővé teszi komplett FC (Blokkdiagram) programok bevitelét ST, IL vagy Gyors LD nyelven programozott műveletekkel és tesztekkel (döntésekkel). A Blokkdiagram egy döntési diagram, ami szekvenciális műveletek leírására is használható, mivel lehetővé tesz bizonyos haladó jellemzőket, mint pl. a nem blokkolt visszafelé ugrásokat.

### A.5.1 Az FC nyelv alapjai

A Blokkdiagram (FC) egy szekvenciális műveletek leírására használt grafikus nyelv. Egy blokkdiagram Műveletekből és Tesztekből áll. A műveletek és tesztek között irányított kapcsolatok vannak, amelyek az adatáramlást képviselik. Az alábbiakban a Blokkdiagram nyelv grafikus komponensei vannak felsorolva:



**Az FC diagram kezdete:** Egy Blokkdiagram program elején a „kezdet” szimbólumnak kell állnia. Ez egyedi, és nem hagyható el. Ez a diagram aktiválásakor annak kezdeti állapotát jelzi.



**Az FC diagram vége:** Egy Blokkdiagram program végén egy „befejezés” szimbólumnak kell állnia. Ez egyedi, és nem hagyható el. Lehetséges, hogy a „Befejezés” szimbólumhoz nincs csatlakozás rajzolva (állandóan hurkolódó diagram), de a „Befejezés” szimbólumot akkor is meg kell rajzolni a diagram alá. Ez a diagram végrehajtásának befejezésekor annak végső állapotát jelzi.



**FC áramlás kapcsolatok:** Az áramlás kapcsolat egy vonal, amely a diagram két pontja közötti áramlást képvisel. A kapcsolat mindig egy nyíllal van lezárva. Két kapcsolat nem indulhat ugyanabból a forrás csatlakozási pontból.



**FC műveletek:** A művelet szimbólum végrehajtandó műveleteket jelez. Egy művelet egy számmal és egy névvel van azonosítva. Ugyanannak a diagramnak két különböző tárgya nem rendelkezhet ugyanazzal a névvel vagy logikai számmal. Egy művelet programozási nyelve ST, LD, vagy IL lehet. Egy művelet mindig kapcsolatokkal csatlakozik, amelyek egyike hozzá érkezik, a másik pedig tőle indul.



**FC tesztek:** Egy teszt egy logikai feltételt jelent. A teszt egy számmal és egy névvel van azonosítva. A csatolt ST, LD vagy IL kifejezés kiértékelésétől függően az áramlás vagy az „IGEN”, vagy a „NEM” útvonal felé irányítódik. ST szövegben programozva, a kifejezést tetszés szerint egy pontosvessző is követheti. LD-ben programozva, az egyedi tekerics képviseli a feltétel értékét.



**FC alprogram:** A rendszer lehetővé teszi az FC programok hierarchikus szerkezetének leírását. Az FC programok egy fastruktúrájú hierarchiába vannak szervezve. Minden FC program meghívhat más FC programokat. Az ilyen programot az azt hívó FC program gyermek programjának nevezik. Az FC alprogramokat meghívó FC programokat apa programnak nevezik. Az FC

programok egy fő fastruktúrájú hierarchiában vannak összekapcsolva egymással, „apa–gyermek” kapcsolatot használva: A blokkdiagramban levő alprogram szimbólum egy blokkdiagram alprogramhoz történő meghívást jelképez. A hívó FC program végrehajtása az alprogram végrehajtásának befejezéséig felfüggesztődik.



**FC I/O specifikus művelet:** Az I/O specifikus művelet szimbólum végrehajtandó műveleteket jelez. Akárcsak egyéb műveletek, úgy az I/O specifikus művelet is egy számmal és egy névvel van azonosítva. A hagyományos műveletekhez és az I/O specifikus műveletekhez ugyanaz a szemantika érvényes. Az I/O specifikus műveletek célja csupán a diagram könnyebb olvashatóságának elősegítése, és a fókusz áthelyezése a diagram elmozdíthatatlan részeire. Az I/O specifikus műveletek használata egy tetszés szerint választható lehetőség. Az I/O specifikus blokkok pontosan ugyanúgy viselkednek, mint a standard műveletek.



**FC csatlakozók:** A Csatlakozók a diagram két pontja közötti kapcsolat jelzésére szolgálnak, annak megrajzolása nélkül. A csatlakozót egy kör jelöli, amely az áramlás forrásával van összekötve. A csatlakozó rajzát a megfelelő oldalon (az adatáramlás irányától függően) a célpont azonosítása (általában a cél szimbólum neve) teszi teljessé. Egy csatlakozó mindig egy definiált Blokkdiagram szimbólumot céloz meg. A rendeltetési hely szimbólum annak logikai számával van azonosítva.



**FC megjegyzések:** A megjegyzés blokk olyan szöveget tartalmaz, amelynek a diagram szemantikájához nincs értelme. Ez a blokkdiagram dokumentum ablak szabad helyére bárhová beilleszthető, és célja a program dokumentálása.

## A.5.2 Egy Blokkdiagram bevitele

Egy diagram beviteléhez a grafikai területre elemeket (műveleteket, döntési teszteket, csatlakozókat, stb.) kell elhelyezni, és áramlási kapcsolatokat kell köztük rajzolni.



### Tárgyak beillesztése

Egy tárgynak a diagramba történő beillesztéséhez válassza ki az eszközsoron a megfelelő gombot, és kattintson a grafikai területen arra a helyre, ahová be kívánja azt illeszteni. Az elemet vagy egy üres területre lehet helyezni, vagy egy áramlatba lehet beilleszteni, annak kapcsolatára kattintva. Egy kapcsolat beillesztése csak felülről lefelé haladó függőleges kapcsolatokhoz van megengedve. A következő alapvető elemeket lehet beilleszteni:



.....ST-ben, IL-ben vagy Gyors LD-ben programozott művelet



.....IO specifikus művelet (egy bizonyos nem mozdítható műveletet emel ki)



.....ST-ben, IL-ben vagy Gyors LD-ben programozott teszt (döntés)



.....csatlakozó



.....hívás egy FC alprogramhoz



.....megjegyzés (leíró szöveg)

Az ISaGRAF Blokkdiagram szerkesztő egy hagyományos blokkdiagram struktúra listát is ajánl. Az ilyen struktúrák csak egy meglevő áramlási kapcsolatba illeszthetők be. Egy üres területre nem lehet őket beilleszteni:



.....Ha / Akkor / Különben (bináris választás)



.....Ismétlés ....-ig (egy feltételre vár)



.....Miközben (hurokba lép miközben egy feltétel igaz)



## Tárgyak kiválasztása

A legtöbb szerkesztési parancshoz grafikus tárgyak kiválasztása szükséges. Az ISaGRAF FC grafikus szerkesztő lehetővé teszi a diagram területén levő egy vagy több tárgy kiválasztását. A tárgyak kiválasztásához a szerkesztő eszközsorán ki kell jelölni a „**kiválasztás**” lehetőséget (egy nyilat tartalmazó gomb). Egyetlen tárgy kiválasztásához a felhasználónak csupán rá kell kattintania annak szimbólumára.

Tárgyak listájának a kiválasztásához az egeret a diagramba kell vonszolni, és ki kell jelölni vele egy négyzet alakú területet. A kiválasztási négyzetben levő összes grafikus tárgy „**kiválasztott**” jelölést kap.

Egy kiválasztott tárgy sötétkék színnel van megrajzolva, és annak grafikus szimbólumát kis fekete négyzetek veszik körül. Lehetséges egy tárgy többszörös kiválasztáshoz való hozzáadása illetve eltávolítása is, a **Shift** vagy **Ctrl** billentyűk lenyomása mellett rákattintva azok szimbólumára.

Egy új kiválasztás végrehajtásakor az összes előzőleg kiválasztott tárgy kiválasztott állapota eltávolítódik. A jelenlegi kiválasztás megszüntetéséhez egyszerűen kattintson az egérrel egy, a kiválasztott tárgyakat befoglaló négyzeten kívüli üres területre.

Az egyszeres kiválasztásnál a billentyűzet nyíl gombjait is lehet használni a diagramban az egyik tárgyról a másikra való átlépéshez. Áramlási kapcsolatok szintén kiválaszthatók.



## Megjegyzések beillesztése

Megjegyzések a diagram üres területére bárhová beilleszthetők. A megjegyzések nincsenek hatással a program végrehajtására. Ezek a diagram könnyebb olvashatóságát segítik elő. Egy megjegyzés blokk beillesztéséhez válassza ki a megfelelő gombot az eszközsorán, és az egérrel kattintson oda a diagramban, ahová a megjegyzést kell beilleszteni. Egy megjegyzés szövegének beviteléhez / megváltoztatásához kattintson duplán a megjegyzésre. Egy megjegyzés blokk szövegének beírásakor nem kell alkalmazni semmiféle speciális kezdeti vagy befejező karaktereket, mint pl. a "(\*" és a "\*)". A megjegyzés blokk méretének megváltoztatásához azt ki kell választani, és a keret sarkát a megfelelő méretűre kell vonszolni az egérrel.



## Áramlási kapcsolatok rajzolása

A meglevő elemek közötti áramlási kapcsolat megrajolásához ezt a gombot kell kiválasztani az eszközsorán. A kapcsolatot mindig az áramlás irányában kell megrajzolni. Először válassza ki egy FC elem nem csatlakoztatott output pontját, és

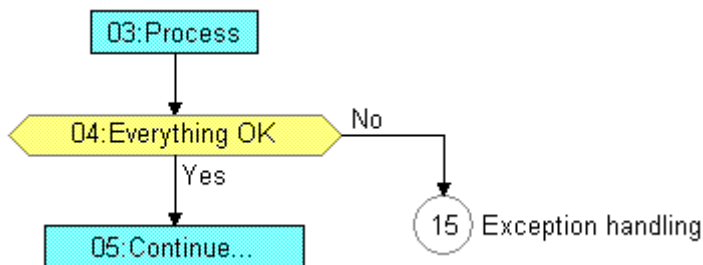
az egeret vonszolja a rendeltetési pontra a kapcsolat beillesztéséhez. A rendeltetési pont lehet vagy egy nem csatlakoztatott FC elem teteje (input pont), vagy bármely hely egy meglevő kapcsolaton. A kapcsolatok közötti konvergencia pontok a Blokkdiagramban kis szürke körökkel vannak jelölve. A konvergencia pontok a diagram átrendezéséhez szintén kiválaszthatók és áthelyezhetők.



### Csatlakozók használata

Az ISaGRAF Blokkdiagram szerkesztő lehetővé teszi a grafikus csatlakozók használatát egy látható áramlási kapcsolat helyett. A csatlakozók nagyon hasznosak a rendkívül hosszú kapcsolatok elkerüléséhez és a diagram olvashatóságának javításához. A csatlakozó nem használható egy másik FC programmal történő kapcsolat létrehozására.

A csatlakozó ugyanúgy van a diagramba helyezve, mint más FC tárgyak. Ezt egy kör jelképezi, amely a benne levő számmal utal a célzott elemre (az áramlási kapcsolat rendeltetési helye). A cél elem rövid leíró szövege a csatlakozó köre mellett van megjelenítve.



### Tárgyak áthelyezése

A diagramban levő tárgyak áthelyezéséhez először ki kell választani azokat, majd az egérrel a diagram megfelelő helyére kell vonszolni őket. Akár egyetlen elem, akár többszörös kiválasztás áthelyezhető. Az elemek áthelyezésükkor nem fedhetik át egymást. Az elemek áthelyezése nem használható azoknak egy meglevő kapcsolathoz történő csatlakoztatására.

Egy elem (művelet, teszt, stb.) áthelyezésekor az ISaGRAF Blokkdiagram szerkesztő automatikusan szintén áthelyez minden, az elem alá helyezett és ahhoz csatlakozó tárgyat. Ez a jellemző többszörös kiválasztás esetén nem működik.



### Tárgyak átméretezése

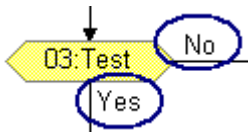
Egy áramlás bármelyik grafikai eleme (a „Kezdet” és „Befejezés” szimbólumok és csatlakozók kivételével) szabadon átméretezhető. Egy elem átméretezéséhez először azt ki kell választani. Ezután a köré rajzolt kis négyzeteket az egérrel vonszolva a méret megváltoztatható.

Amikor egy elem egy áramlási kapcsolathoz van csatlakoztatva, annak vízszintes átméretezése a bal és jobb oldali határvonalakra egyaránt hat, így az elem átméretezésekor továbbra is megfelelően középpontban marad a kapcsolaton.



### Egy teszt outputjainak felcserélése

Egy teszt (döntés) IGEN / NEM outputjainak helyei felcserélhetők. Ennek elvégzéséhez egyszerűen kattintson duplán a teszt szimbóluma mellett levő „Igen” vagy „Nem” jelre.



### A.5.3 Egy meglevő diagrammal történő munkavégzés

A „**Szerkesztés**” menü parancsai egy meglevő diagram megváltoztatására vagy befejezésére szolgálnak. Ezeknek a parancsoknak a legtöbbje a diagramban pillanatnyilag kiválasztott elemekre hat.



#### Egy diagram korrigálása

A DEL gombbal a kiválasztott elemeket lehet eltávolítani. A kiválasztott elemekkel együtt a függőben levő kapcsolatok is törölődnek. A DEL paranccsal törölt elemek a „**Szerkesztés / Visszavonás**” paranccsal állíthatók vissza. A DEL parancs a diagramban kiválasztott elemcsoportra is alkalmazható. A „**Szerkesztés**” menü „**Kivágás**”, „**Másolás**”, „**Beillesztés**” parancsai a kiválasztott elemek áthelyezésére vagy másolására szolgálnak.



#### Keresés és lecserélés

A „**Szerkesztés / Keresés és lecserélés**” parancsok szöveg karaktorsorok megkeresésére vagy lecserélésére használhatók a teljes programban (minden ST-ben, IL-ben, vagy Gyors LD-ben programozott művelet és teszt). A Keresés /Lecserélés párbeszédablak egy keresendő szöveg bevitelére és annak a programozási résznek a közvetlen megnyitására szolgál, ahol a szöveg található.



#### Egy elem közvetlen elérése

A felhasználó a „**Szerkesztés / Ugrás**” parancs segítségével egy, a diagramban létező grafikus elemhez juthat el. A görgetési pozíció automatikusan átállítódik úgy, hogy az elem láthatóvá váljon. Az elért elem kiválasztódik.



#### Elemek átszámozása

A „**Szerkesztés / Átszámozás**” parancs a Blokkdiagram elemeinek átszámozására szolgál. A diagramba helyezett FC elemek mindegyikét egy egyedi hivatkozási szám azonosítja. A hivatkozási számokat a szerkesztő adja meg, valahányszor új elemek vannak beillesztve. Az „**Átszámozás**” segítségével az elemek számozásai a diagramban betöltött helyüknek megfelelően átállíthatók. A számok felülről lefelé és balról jobbra haladva növekednek.

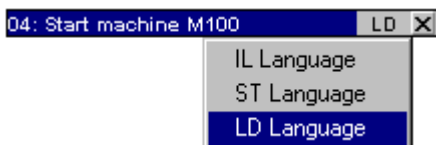
## A.5.4 2. szintű programok bevitele

A 2. szintű program beviteléhez a felhasználónak duplán rá kell kattintania a művelet vagy teszt szimbólumára. A 2. szintű programozás az FC ablak jobb oldalán jelenik meg. Az FC diagram és a 2. szintű programozás közötti elválasztó vonal szabadon elmozgatható. Egyidejűleg egy vagy kettő 2. szintű terület nyitható ki. A következő parancsok a billentyűzetten, az egérrel, vagy a „Szerkesztés” menün állnak rendelkezésre:

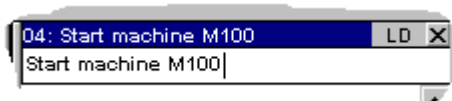
	<i>Billentyűzet</i>	<i>Egér</i>	<i>„Szerkesztés” menü</i>
Megnyitás a legutóbbi alapértelmezett ablakban:	Enter	Dupla kattintás	2. szint szerkesztése
Megnyitás külön ablakban:	Ctrl+Enter	Ctrl + Dupla kattint.	2. szint szerkesztése külön ablakban

Amikor két 2. szintű ablak látható, a közöttük levő elválasztó szabadon mozgatható. Egy 2. szintű ablak bezárására a 2. szintű címsortól jobbra levő gomb szolgál.

A 2. szintű programozás alapértelmezett nyelve az **ST** (Strukturált Szöveg). A programozási nyelv lehet **IL** vagy **Gyors LD** is. A kiválasztott nyelv neve a 2. szint címsorában, egy kis négyzetben látható. Az aktív nyelv megváltoztatásához a „Beállítások / 2. szintű nyelv beállítása” parancsot kell futtatni a menükről, vagy a négyzetbe kell kattintani. Ez a parancs csak akkor érvényes, ha a 2. szintű programozási ablak üres.



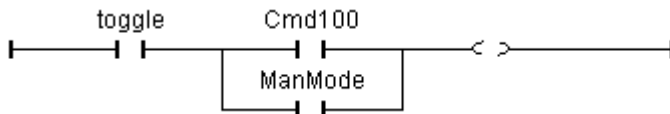
A 2. szintű ablak felső részén egy egyetlen sort tartalmazó szerkesztőablak jelenik meg. Ez egy rövid leíró szöveg beírására szolgál. Ez a szöveg IEC megjegyzésként fog megjelenni az FC szimbólumok rajzában. Ez nagyon hasznos, és más parancsok, pl. az „Ugrás” is használják, valamint az FC kinyomtatásokban is használatos az FC lépések és átmenetek dokumentálására.



A „Beállítások / Frissítés” parancs bármikor használható a fő FC diagram frissítésére 2. szintű programokkal, amikor a 2. szintű ablakok nyitva vannak.

### A.5.5 2. szint programozása Gyors LD-vel

A Gyors LD szerkesztő 2. szintű programozáshoz áll rendelkezésre. Egy döntési teszt esetében az LD diagram mindössze egyetlen létrafokból áll, egyetlen tekercssel, amely a döntést képviseli. A teszt neve nincs megismételve a tekercs szimbólummal. Az alábbiakban egy Gyors LD-ben programozott teszt példája látható:



A Gyors LD-ben történő programozáskor a kiválasztásnak a programozási logikai rácsban történő elmozdítására a billentyűzet nyíl gombjait, majd a szimbólumok beillesztéséhez a következő billentyűparancsokat kell használni:

- F2: .....egy kontakt beillesztése a kiválasztott szimbólum után / a létrafok elkezdése
- F3: .....egy kontakt beillesztése a kiválasztott szimbólum elé
- F4: .....egy kontakt beillesztése a kiválasztott szimbólummal párhuzamosan
- F5: .....egy tekercs beillesztése a kiválasztottal párhuzamosan (nem teszthez)
- F6: .....egy blokk beillesztése a kiválasztott szimbólum után
- F7: .....egy blokk beillesztése a kiválasztott szimbólum elé
- F8: .....egy blokk beillesztése a kiválasztott szimbólummal párhuzamosan
- F9: .....egy ugrás szimbólum beillesztése a kiválasztott tekercssel párhuzamosan (nem tesztekhez)

Egy ugrás egy létrafok nevéhez vezet. Egy létrafok nevének beírása az ENTER megnyomásával történhet olyankor, amikor a kiválasztás a létrafok tetején van. Az ISaGRAF szerkesztő memóriában tartja a már bevitt létrafok címkeket, függetlenül attól, hogy az létrafok névként vagy ugrási művelethez lett-e specifikálva. Az „Ugrás/Címke” párbeszédablak lehetővé teszi akár egy új címke bevitelét, akár egy már meglevő címke kiválasztását. Ha egy új nevet ír be, akkor az automatikusan a listához lesz adva. A kiválasztott név listáról történő eltávolítására az „Eltávolítás” parancs szolgál. Ez a diagramban kiválasztott létrafokon levő címkét nem távolítja el. Ahhoz egyszerűen meg kell nyomni az **OK** gombot, amikor a szerkesztő ablak üres.

A funkciógombok helyett az LD eszközsoron levő gombok is megnyomhatók.

Ha a kiválasztás egy kontakton vagy egy blokk I/O paraméteren van, akkor egy változó kiválasztásához illetve egy konstans érték beírásához az ENTER gombot kell megnyomni. Ha a kiválasztás egy funkcióblokkon van, akkor a funkcióblokk típusának kiválasztásához az ENTER gombot kell megnyomni. Ugyanez érhető el akkor is, ha egy szimbólumra duplán rákattint.

Ha egy kontakt van kiválasztva, akkor a kontakt vagy tekercs típusának megváltoztatásához (direkt, tagadott) a Control + SZÖKÖZ billentyűt kell megnyomni. A Gyors LD lehetőségeivel kapcsolatos további információk ennek a dokumentumnak „A Gyors LD szerkesztő használata” című fejezetében találhatók.

### A.5.6 Megjelenítési lehetőségek

A **„Beállítások / Elrendezés”** parancs egy párbeszédablakot nyit ki, amelyben a szerkesztő munkaterülettel és a diagramban rajzával kapcsolatos összes paraméter illetve beállítási lehetőség van csoportosítva. A „Munkaterület” csoport ablakban levő jelölőnégyzet a szerkesztő eszközsorainak és állapotsorának megjelenítésére vagy elrejtésére szolgál. A „Dokumentum” csoport ablakban levő beállítási lehetőségek a szerkesztőrács pontjainak megjelenítését illetve elrejtését, valamint a diagram fekete-fehérben vagy színesben történő ábrázolását teszik lehetővé.



Az eszközsor „Zoom” gombjával a pillanatnyi zoom arány változtatható meg. Ez a parancs egy művelethez vagy teszthez csatolt Gyors LD programmal történő munkavégzéskor is rendelkezésre áll.



Az eszközsoron levő „Rács” gomb a szerkesztőrács pontjainak megjelenítésére illetve elrejtésére szolgál. Ez a parancs egy művelethez vagy teszthez csatolt Gyors LD programmal történő munkavégzéskor is rendelkezésre áll.

A **„Beállítások / Fontok”** paranccsal az összes ISaGRAF grafikus dokumentumban használandó karakterfont neve választható ki. Egy ST vagy IL blokkból történő meghíváskor meghatározható a font mérete. A font egy grafikus nézethez (FC vagy Gyors LD) történő kiválasztásakor a font stílusának és méretének nincs jelentősége, ezért azokat nem kell megadni. Az ISaGRAF grafikus szerkesztők mindig a pillanatnyi zoom aránynak megfelelően számítják ki a font méretet.

## A.6 A Gyors LD szerkesztő használata

Az LD nyelv logikai kifejezések grafikus jelölését teszi lehetővé. A logikai ÉS, VAGY, NEM operátorokat a diagram topológiája jelzi explicit módon. A logikai input változók grafikus kontaktokhoz csatlakoznak. A logikai output változók grafikus tekercsekhez csatlakoznak. Az ISaGRAF Gyors LD szerkesztő egyszerű LD diagram beírást tesz lehetővé a billentyűzet vagy az egér használatával. Az elemeket a Gyors LD szerkesztő automatikusan egymáshoz kapcsolja és elrendezi a létrafokokon. A felhasználónak nem kell csatlakozásokat rajzolnia. A Gyors LD szerkesztő amellett a létrafokokat úgy rendezi el a diagramban, hogy a diagram által elfoglalt hely mindig optimális legyen.

### A.6.1 Az LD nyelv alapjai





Egy LD program **létrafokok** sorozatával van kifejezve, amelyeken kontaktok és tekercsek vannak elrendezve. Az alábbiakban egy LD diagram alapvető komponensei vannak felsorolva:

#### **A létrafok teteje (bal oldali áramsín)**

Minden létrafok egy bal oldali áramsínnel kezdődik, amely a kiindulási „IGAZ” állapotot képviseli. Az ISaGRAF Gyors LD szerkesztő automatikusan létrehozza a bal oldali áramsín, amikor a felhasználó elhelyezi a létrafok első kontaktját. Minden létrafoknak lehet egy logikai neve, amely az ugrás utasításokhoz címkeként használható.





#### **Kontaktok**

Egy kontakt a logikai adatáramlást módosítja, egy logikai változó állapotának megfelelően. A változó neve a kontakt szimbólum tetején van feltüntetve. Az ISaGRAF Gyors LD szerkesztő a következő kontakt típusokat támogatja:


-  .....közvetlen kontakt
-  .....tagadásos kontakt
-  .....kontakt pozitív (felfutó) éldetektálással
-  .....kontakt negatív (lefutó) éldetektálással

#### **Tekercsek**

A tekercs egy műveletet jelképez. A létrafok állapota (a tekercs bal oldalán levő kapcsolat állapota) egy logikai változó kényszerítésére használatos. A változó neve a tekercs szimbólum tetején van feltüntetve. Az ISaGRAF Gyors LD szerkesztő a következő tekercs típusokat támogatja:

-  .....közvetlen tekercs
-  .....tagadásos tekercs
-  .....„beállítás” művelet tekercs
-  .....„visszaállítás” művelet tekercs

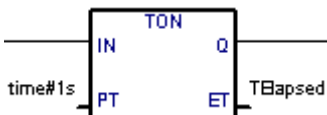
 .....tekercs pozitív (felfutó) éldetektálással

 .....tekercs negatív (lefutó) éldetektálással



## Funkcióblokkok

Egy LD diagramban levő blokk képviselhet egy funkciót, funkcióblokkot, alprogramot, vagy operátort. Annak első input és output paraméterei mindig a létrafokhoz vannak csatlakoztatva. Az egyéb input és output paraméterek a blokk keretén kívülre vannak írva szövegesen.



## A létrafok vége (jobb oldali áramsín)

A létrafok egy jobb oldali áramsínnel fejeződik be. A Gyors LD szerkesztő használatával a jobb oldali áramsín automatikusan beillesztődik, amikor a felhasználó elhelyez egy tekercsset.



## Ugrás szimbólum

Egy ugrás szimbólum mindig egy létrafok címkére, azaz egy, ugyanabban a diagramban máshol definiált létrafok névre hivatkozik. Ez a létrafok végén van elhelyezve. Ha a létrafok állapota IGAZ, akkor a diagram végrehajtása közvetlenül erre a cél létrafokra ugrik. Megjegyzendő, hogy a visszafelé történő ugrások veszélyesek, mert bizonyos esetekben ezek a PLC hurok blokkjához vezethetnek.



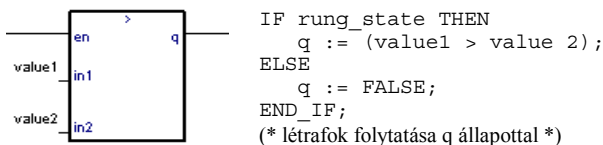
## Visszatérés szimbólum

A visszatérés szimbólum a létrafok végén van elhelyezve. Ez azt jelzi, hogy a program végrehajtásának meg kell állnia, ha a létrafok állapota IGAZ.



## Az „EN” input

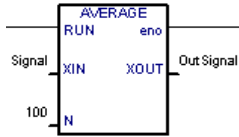
Egyes operátorokon, funkciókon, vagy funkcióblokkokon az első inputnak nincs logikai adat-típusa. Mivel az első inputot mindig a létrafokhoz kell csatlakoztatni, ezért az első pozícióba egy „EN” nevű másik input kerül automatikusan beillesztésre. A blokk csak akkor kerül végrehajtásra, ha az EN input is IGAZ. Az alábbiakban egy összehasonlítás operátor, valamint annak ST-ben kifejezett kódja látható:



## Az „ENO” output

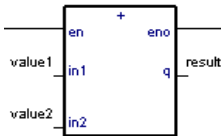
Egyes operátorokon, funkciókon, vagy funkcióblokkokon az első outputnak nincs logikai adat-típusa. Mivel az első outputot mindig a létrafokhoz kell csatlakoztatni,

ezért az első pozícióba egy „**ENO**” nevű másik output kerül automatikusan beillesztésre. Az **ENO** output mindig ugyanazt az állapotot veszi fel, mint a blokk első inputja. Az alábbi példában egy ÁTLAG funkcióblokk, valamint annak ST-ben kifejezett kódja látható:



```
AVERAGE(rung_state, Signal, 100);
OutSignal := AVERAGE.XOUT;
eno := rung_state;
(* létrafok folytatása eno állapottal *)
```

Bizonyos esetekben mind az **EN**, mind az **ENO** szükséges. Az alábbi példában egy aritmetikai operátor, valamint annak ST-ben kifejezett kódja látható:



```
IF rung_state THEN
    result := (value1 + value2);
END_IF;
eno := rung_state;
(* létrafok folytatása eno állapottal *)
```

## HHH A Gyors LD szerkesztő korlátai

Az ISaGRAF Gyors LD szerkesztő editor nem engedi meg egy létrafok folytatását (további kontaktok vagy tekercsek beillesztését) egy tekercstől jobbra. Ha ugyanazon a létrafokon több outputot kell létrehozni, akkor a megfelelő tekercseket párhuzamosan kell megrajzolni.

### A.6.2 Egy LD diagram bevitele

A Gyors LD szerkesztő valamennyi szerkesztő parancsa elérhető vagy a billentyűzettel, vagy az egérrel.



#### A szerkesztő rács

Az LD diagram bevitele egy logikai mátrixban történik. A mátrix minden cellája maximum egy LD szimbólumot tartalmazhat. A pillanatnyi kiválasztás elmozdításához használja a billentyűzet nyíl gombjait, vagy kattintson egy cellára. A kiválasztott cella negatív árnyalattal van jelölve. Bizonyos kivágási/másolási/beillesztési műveletekhez egyszerre több cella is kiválasztható. Ennek elvégzéséhez egyszerűen vonszolja az egér kurzorát a diagramban a megfelelő helyre. A billentyűzettel ehhez a nyíl gombokat és a SHIFT gombot együttesen kell használni.

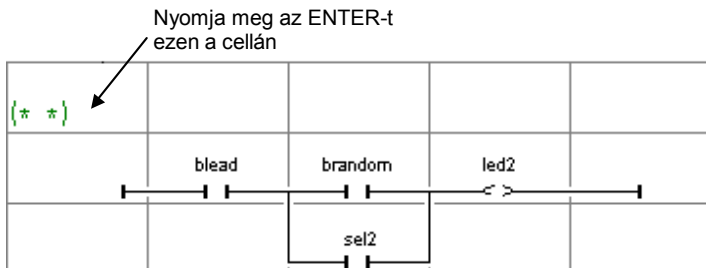


#### Egy új létrafok elkezdése

Egy új létrafoknak a diagramhoz adásához vigye a kiválasztást a legutolsó létező létrafok után levő helyre, és illesszen be egy kontaktot (nyomja meg az F2 gombot, vagy az LD eszközsor megfelelő gombját). Ekkor egy új létrafok lesz létrehozva, egy kontakttal és egy tekercssel.

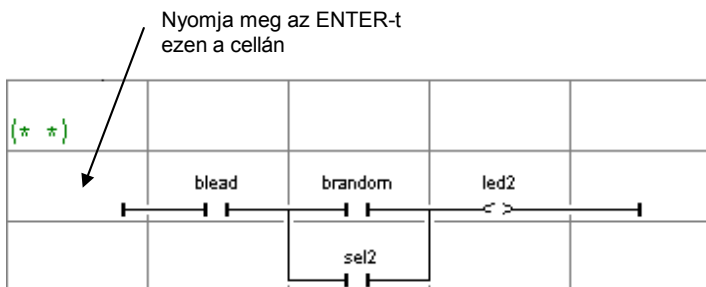
## A létrafok megjegyzés bevitele

Minden létrafok maximum két sornyi szöveggel dokumentálható. Egy létrafok megjegyzés szöveg beírásához vigye a kiválasztást a létrafokra, és nyomja meg az ENTER gombot, vagy kattintson duplán erre a cellára az egérrel:



## A létrafok címke bevitele

Minden létrafok ellátható egy azonosító névvel. Ezt a nevet ugrási műveletekhez cél-címkeként lehet használni. Egy létrafok címkéjének beírásához vagy annak módosításához vigye a kiválasztást a létrafokra, és nyomja meg az ENTER gombot, vagy kattintson duplán erre a cellára az egérrel:



Az ISaGRAF Gyors LD szerkesztő memóriában tartja a már bevitt létrafok címkéket, függetlenül attól, hogy az létrafok névként vagy ugrási művelethez lett-e specifikálva. Az „Ugrás/Címke” párbeszédablak lehetővé teszi akár egy új címke bevitelét, akár egy már meglévő címke kiválasztását.

Ha egy új nevet ír be, akkor az automatikusan a listához lesz adva. A kiválasztott név listáról történő eltávolítására az „**Eltávolítás**” parancs szolgál. Ez a diagramban kiválasztott létrafokon levő címkét nem távolítja el. Ahhoz egyszerűen meg kell nyomni az **OK** gombot, amikor a szerkesztő ablak üres.

## Szimbólumok beillesztése egy létrafokra

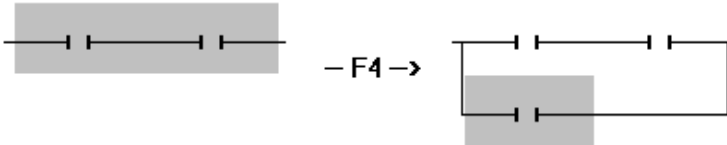
A szimbólumok (kontaktok, tekercsek, blokkok, stb.) meglévő létrafokra történő beillesztése mindig a pillanatnyi kiválasztásnak megfelelően történik. A létrafokon ki kell választani egy érvényes cella helyet, és a beillesztéshez a következő funkcióbillentyűk valamelyikét meg kell nyomni:

- F2 .....egy kontakt a kiválasztott szimbólum elé (bal oldalra)  
 F3 .....egy kontakt a kiválasztott szimbólum után (jobb oldalra)  
 F4 .....egy kontakt a kiválasztott szimbólummal párhuzamosan  
 F6 .....egy blokk a kiválasztott szimbólum elé (bal oldalra)  
 F7 .....egy blokk a kiválasztott szimbólum után (jobb oldalra)  
 F8 .....egy blokk a kiválasztott szimbólummal párhuzamosan

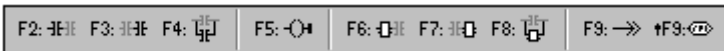
A következő parancsok érvényesek olyankor, amikor a kiválasztás a létrafok outputján (tekercs) van:

- F5 .....egy tekercs beillesztése a kiválasztottal párhuzamosan  
 F9 .....egy „Ugrás” szimbólum beillesztése a kiválasztottal párhuzamosan  
 Shift + F9 .....egy „Visszatérés” szimbólum beillesztése a kiválasztottal párhuzamosan

A párhuzamos beillesztéskor (F4/F8), ha egy létrafok több kontaktja van együtt kiválasztva, a szimbólum a kiválasztott elemek csoportjával párhuzamosan lesz beillesztve. Az alábbiakban egy példa látható:



A szimbólumok diagramba történő beillesztéséhez a „Beillesztés” menü parancsait is lehet használni. Az egérrel az LD eszközsoron rá lehet kattintani a beilleszteni kívánt szimbólumra:



## Szimbólumok bevitele

Egy változó szimbólumnak egy kontakttal vagy tekerccsel történő társításához válassza ki azt, és nyomja meg az ENTER gombot. Az egérrel kattintson duplán a kontaktra vagy tekercsre. Ekkor megjelenik egy változó kiválasztó ablak. Ennek az ablaknak a használatával kapcsolatos további információk a „További tudnivalók a programszerkesztőkről” című fejezetben találhatók. Egy funkciónak, funkcióbloknak, vagy operátornak egy blokkhoz történő társításához nyomja meg az ENTER gombot, amikor a kiválasztás annak a négyzetén belül van. Egy változó szimbólumnak egy input vagy output blokk paraméterhez történő társításához a kiválasztásnak a megfelelő helyen kell lennie, a blokk négyzetén **kívül**.

A szövegek bevételéhez általában párbeszédablakok szolgálnak, ide értve a változó illetve blokk-kiválasztó listákat is. Ha a „Beállítások” menüben ki lett választva a „Manuális billentyűzet beírás” mód, akkor a változó szimbólumok és blokknevek beírása közvetlenül egy szövegszerkesztő ablakba történik. Írja be az új szöveget, és annak megerősítéséhez nyomja meg a „Bevitel” gombot, vagy nyomja meg a „Mégsem” gombot módosításból való kilépéshez és a szövegszerkesztő ablak

bezárásához. A „manuális billentyűzet beírás” módhoz használt szövegszerkesztő ablakot az egérrel nem lehet bezárni.



### **A kontaktok és tekercsek típusának módosítása**

A kiválasztott kontakt vagy tekercs típusát a „**Szerkesztés / Tekercs/kontakt típus megváltoztatása**” paranccsal lehet megváltoztatni. Egy kontakt lehet közvetlen, tagadásos, illetve pozitív vagy negatív éldetektálásos. Egy tekercs lehet közvetlen, tagadásos, beállító vagy visszaállító, illetve pozitív vagy negatív éldetektálásos. A SZÓKÖZ billentyű lenyomásának ugyanez a hatása.



### **Egy létrafok beillesztése egy diagramba**

A „**Szerkesztés / Létrafok beillesztése**” parancs egy új létrafokot illeszt be a diagramba a kiválasztott létrafok elé. A létrafok egy kontakttal és egy tekercssel indul.

## **A.6.3 Egy meglevő diagrammal történő munkavégzés**

A „**Szerkesztés**” menü parancsai egy meglevő diagram megváltoztatására vagy befejezésére szolgálnak. Ezeknek a parancsoknak a legtöbbje a diagramban pillanatnyilag kiválasztott elemekre hat.



### **Egy diagram korrigálása**

A DEL gombbal a kiválasztott elemeket lehet eltávolítani. Egy tekercset, valamint ugrás illetve visszatérés szimbólumot nem lehet eltávolítani, ha az egy létrafok egyetlen outputja. A DEL paranccsal törölt elemek a „**Szerkesztés / Visszavonás**” paranccsal állíthatók vissza. A DEL parancs a diagramban kiválasztott elemcsoportra is alkalmazható. Ha a kiválasztás a létrafok megjegyzés szövegén van, akkor a DEL paranccsal az visszaállítható. Ha a kiválasztás a létrafok tetején van, akkor a DEL paranccsal az egész létrafok eltávolítható.



### **Szimbólumok másolása**

A „**Szerkesztés**” menü „**Kivágás**”, „**Másolás**”, „**Beillesztés**” parancsai a kiválasztott elemek áthelyezésére vagy másolására szolgálnak. Ezek a parancsok nem hatnak a létrafok megjegyzésekre. A „**Szerkesztés / Speciális beillesztés**” parancs lehetővé teszi az elemek beillesztését a következők szerint:

- a kiválasztott elem elé (bal oldalra)
- a kiválasztott elem után (jobb oldalra)
- a kiválasztott elemmel párhuzamosan



### **Egész létrafokok kezelése**

Ha a kiválasztás a létrafok fejsorán (a bal oldali áramsínen) van, akkor valamennyi szerkesztési parancs (törlés, másolás, kivágás) az egész létrafokra fog hatni. Ez leegyszerűsíti a diagramban a létrafokok átrendezését azzal, hogy csak az első oszlopban kell a kiválasztást áthelyezni. Lehetséges a kiválasztás függőleges kiterjesztése is, hogy az több létrafok fejrészét is tartalmazzon. Ebben az esetben a szerkesztési parancsokat létrafokok egész listájára lehet alkalmazni.

## ☐ **Keresés és lecserélés**

A **„Szerkesztés / Keresés”** és **„Szerkesztés / Lecserélés”** menü parancsok a diagramban levő szövegek megkeresésére és lecserélésére szolgálnak. Csak teljes neveket lehet megkeresni. A keresés a kontaktokon, tekercseken, blokk neveken, blokk paramétereken, és futási címkéken kerül végrehajtásra. Ez nem használható egy létrafok megjegyzésében levő karaktorsor megkeresésére. A Lecserélés parancs nem használható egy blokk típusának a megváltoztatására. A keresés a pillanatnyi kiválasztás pozíciójától indulva történhet felfelé és lefelé. Ez a diagram határának elérésekor „hurokba lép”. A változónevek gyors kereséséhez a következő billentyűzet parancsok is rendelkezésre állnak:

**ALT+F2** megkeresi következő, a pillanatnyilag kiválasztott elem változónevével megegyező nevű elemet. Ez a lehetőség funkcióblokkokra és létrafokokra szintén alkalmazható.

**ALT+F5** megkeresi következő, a pillanatnyilag kiválasztott elem változónevével megegyező nevű tekercset. Ez a lehetőség elsősorban hibakeresési módban használatos annak a létrafoknak a gyors megkeresésére, amely egy gyanús változót kényszerít ki.

### **A.6.4 Megjelenítési lehetőségek**

A **„Beállítások”** menü parancsai a képernyőn levő LD diagram rajzának testre szabására, valamint bizonyos típusú információk megjelenítésére vagy elrejtésére szolgálnak.

## ☐ **Létrafok megjegyzések**

A létrafok megjegyzéseknek az egész diagramban történő megjelenítésére illetve elrejtésére a **„Beállítások / Létrafok megjegyzések”** parancsot kell használni. Egy nagyméretű diagram tömörített formájú megtekintéséhez a létrafok megjegyzések elrejtése lehet szükséges, mivel a szerkesztő mátrixban minden megjegyzés egy sort foglal el. Ez a beállítás nem befolyásolja a meglévő létrafok megjegyzések tartalmát, és bármikor visszakapcsolható.

## ☐ **Nevek és álnévek**

Minden kontakthoz, tekercshez, vagy blokk I/O paraméterhez rendelt változót saját szimbolikus neve azonosít. Az ISaGRAF Gyors LD szerkesztő minden változóhoz bevezeti az **„álnév”** fogalmát is. A változó álneve a változó megjegyzésének szövege, az első „:” karakter előtt megcsonkítva. Ennek maximális hossza 16 karakter. Az alábbiakban példák láthatók:

<i>változó megjegyzés:</i>	<i>álnév:</i>
rövid szöveg	rövid szöveg
hosszú szöveg, elválasztó nélkül	hosszú szöveg, n-el
rövid szöveg: hosszú leírás	rövid szöveg

Az álnévek nincsenek hatással az LD diagram végrehajtására, és azokat szintaktikai szempontból megjegyzéseknek kell tekinteni. A változó álneve automatikusan kivonódik a változó megjegyzésből, amikor a név a változó listából van kiválasztva. Ez manuálisan nem változtatható meg. A változó azonosítás

megjelenítési módjának kiválasztásához a **„Beállítások / Kontaktok és tekercsek”** parancsokat kell használni. A következő módok állnak rendelkezésre:

- csak a változóneveknek a megjelenítése
- csak a változó álneveknek a megjelenítése
- mind a nevek, mind az álnevek megjelenítése

A Gyors LD szerkesztő a változó álnevek szótárban történő megváltoztatásakor nem frissíti automatikusan az LD dokumentumokat. A szerkesztett diagramban az összes álnév frissítéséhez a **„Beállítások / Kontaktok és tekercsek / Álnevek frissítése”** parancsot kell használni. A **„Beállítások / Kontaktok és tekercsek”** menüben a **„Megnyitáskor mindig frissítés”** lehetőség is beállítható, amelynek eredményeként az ISaGRAF egy Gyors LD program megnyitásakor minden alkalommal automatikusan frissíti az összes használt álnévet. Figyelmeztetés: Ennek a lehetőségnek a beállítása jelentős mértékben megnövelheti egy program megnyitásához szükséges időt.

## **Rajzolási beállítások**

A **„Beállítások / Elrendezés”** parancs egy párbeszédablakot nyit ki, amelyben a szerkesztő munkaterülettel és a grafikus LD diagram rajzával kapcsolatos összes paraméter illetve beállítási lehetőség van csoportosítva.

A „Munkaterület” csoport ablakban levő jelölőnégyzet a szerkesztő eszközsorának, állapot sorának, és LD eszközsorának a megjelenítésére vagy elrejtésére szolgál. A „Dokumentum” csoport ablakban levő beállítási lehetőségek a szerkesztőrács pontjainak megjelenítését illetve elrejtését, valamint a rajzban a színek használatát teszik lehetővé.



A „Zoom” csoport beállítási lehetőségei egy fő zoom arány kiválasztását teszik lehetővé. Az alapértelmezett zoom arányok közötti átváltása a szerkesztő eszközsorban levő „zoom” gombbal is lehetséges.



A szerkesztőrácsban levő cellák X/Y méretaránya ugyancsak tesztre szabható. Ez az utóbbi beállítás az alapértelmezett cellaszélesség csökkentésére használható olyankor, amikor főleg rövid változóneveket használnak. Az X/Y méretarány változtatása az Elrendezés párbeszédablakba való belépés nélkül a szerkesztő eszközsorban levő „szélesség” gombbal is lehetséges.

A **„Beállítások / Fontok”** paranccsal az összes ISaGRAF grafikus dokumentumban használandó karakterfont neve választható ki. A font kiválasztásakor a font stílusának és méretének nincs jelentősége, ezért azokat nem kell specifikálni. Az ISaGRAF grafikus szerkesztő mindig a kiválasztott zoom aránynak megfelelően számítja ki a font méretet.

## A.7 Az FBD/LD szerkesztő használata

Az ISaGRAF grafikus FBD/LD szerkesztő lehetővé teszi komplett FBD programok bevitelét, amelyek részben LD-t is tartalmazhatnak. Ez egyesíti a grafikus- és szövegszerkesztési lehetőségeket, így diagramok és a megfelelő inputok valamint outputok egyaránt bevitelűk. Mivel ez a szerkesztő főként az FBD nyelvre irányul, ezért a tisztán LD diagramokat az ISaGRAF Gyors LQ szerkesztővel kell bevinni.

### A.7.1 Az FBD/LD nyelvek alapjai

Az **FBD** nyelv számos különböző típusú egyenletet képvisel grafikus formában. Az **operátorokat** négyszögletes funkciókeretek képviselik. A funkció inputok a keret bal oldalához csatlakoznak. A funkció outputok a jobb oldalhoz csatlakoznak. A diagram inputok és outputok (**változók**) **logikai kapcsolatokkal** csatlakoznak a funkciókeretekhez. Egy funkciókeret outputja csatlakoztatható egy másik keret inputjához.

Az **LD** nyelv logikai kifejezések grafikus megjelenítését teszi lehetővé. A logikai **ÉS**, **VAGY**, **NEM** operátorokat a diagram topológiája képviseli explicit módon. A logikai input változók grafikus **kontaktokhoz** csatlakoznak. A logikai output változók grafikus **tekercsekhez** csatlakoznak. A kontaktokat és tekercseket **vízszintes vonalak** kötik össze egymással, illetve a bal- és jobb oldali áramsínekkel. Minden vonalszegmens logikai **IGAZ** vagy **HAMIS** állapotban van. Valamennyi közvetlenül egymáshoz kapcsolódó szegmens logikai állapota megegyező. Valamennyi, a bal oldali **függőleges áramsínhez** csatlakoztatott vízszintes vonal **IGAZ** állapotú.

Az LD és FBD diagramok mindig balról jobbra, és felülről lefelé haladva vannak értelmezve. Az LD és FBD nyelvekkel kapcsolatos további részletes információk az ISaGRAF Nyelvi hivatkozás kézikönyvben található. A következőkben az LD és FBD nyelvek FBD/LD szerkesztő által támogatott alapvető grafikus komponensei láthatók:



#### **Bal oldali áramsín**

A létrafokokat balról egy **bal oldali áramsínhez** kell csatlakoztatni, amely a kiindulási „IGAZ” állapotot képviseli. Az ISaGRAF FBD szerkesztő lehetővé teszi bármilyen logikai szimbólumnak a bal oldali áramsínhez történő csatlakoztatását is.



#### **Jobb oldali áramsín**

A tekercsek a jobb oldalon egy **jobb oldali áramsínhez** csatlakoztathatók. Ez az ISaGRAF FBD/LD szerkesztő használata közben egy választható lehetőség. Ha egy tekercs nincs csatlakoztatva a jobb oldalon, akkor az egy jobb oldali áramsínt tartalmaz a saját rajzán belül.



#### **LD függőleges „VAGY” csatlakozás**

Az LD függőleges csatlakozás mind a bal, ind a jobb oldalon elfogad több csatlakozást is. Minden jobb oldalon levő csatlakozás a bal oldalon levő csatlakozások VAGY kombinációjával egyenértékű.



## Kontaktok

Egy kontakt a logikai adatáramlást módosítja, egy logikai változó állapotának megfelelően. A változó neve a kontakt szimbólum tetején van feltüntetve. Az ISaGRAF FBD/LD szerkesztő a következő kontakt típusokat támogatja:

- .....közvetlen kontakt
- .....tagadásos kontakt
- .....kontakt pozitív (felfutó) éldetektálással
- .....kontakt negatív (lefutó) éldetektálással



## Tekercsek

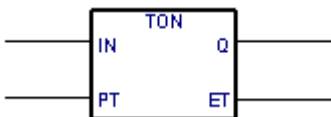
A tekercs egy műveletet jelképez. Ezt bal oldalon egy logikai szimbólumhoz (pl. egy kontakthoz) kell csatlakoztatni. A változó neve a tekercs szimbólum tetején van feltüntetve. Az ISaGRAF FBD/LD szerkesztő a következő tekercs típusokat támogatja:

- .....közvetlen tekercs
- .....tagadásos tekercs
- .....„beállítás” művelet tekercs
- .....„visszaállítás” művelet tekercs



## Funkcióblokkok

Az FBD diagramban egy blokk képviselhet egy funkciót, funkcióblokkot, alprogramot, vagy operátort. Az inputoknak és outputoknak változókhöz, kontaktokhoz illetve tekercsekhez, vagy más blokk inputokhoz illetve outputokhoz kell csatlakozniuk. A formális paraméternevek a blokk keretén belül vannak feltüntetve.



## Címkék

A címkék a diagramban bárhová elhelyezhetők. A címkék ugrási utasítások rendeltetési helyeiként használatosak, a diagramon belüli végrehajtási sorrend megváltoztatására. A címkék nincsenek más elemekhez csatlakoztatva. A diagram olvashatóságának biztosítása érdekében ajánlatos a címkéket a diagram bal oldalán elhelyezni.



## Ugrások

Egy ugrás szimbólum mindig egy, a diagramban máshol elhelyezett címkére utal. Ennek bal oldali csatlakozását egy logikai ponthoz kell kapcsolni. Ha a bal oldali csatlakozás IGAZ, akkor a diagram végrehajtása közvetlenül erre a cél címkére ugrik. Megjegyzendő, hogy a visszafelé történő ugrások veszélyesek, mert bizonyos esetekben ezek a PLC hurok blokkjához vezethetnek.



### **Visszatérés szimbólum**

A visszatérés szimbólum egy logikai ponthoz kapcsolódik. Ez azt jelzi, hogy a program végrehajtásának meg kell állnia, ha a létrafok állapota IGAZ.



### **Változók**

A diagramban a változók kis négyzetekben vannak jelölve, amelyek balról vagy jobbról a diagram más elemeihez csatlakoznak.



### **Csatlakozási kapcsolatok**

A csatlakozási kapcsolatok a diagramban levő elemek közé rajzolva találhatók. A kapcsolatokat mindig egy output ponttól egy input ponthoz kell rajzolni (az adatáramlás irányában).



### **Csatlakozási kapcsolatok logikai tagadással**

Egyes logikai kapcsolatokat jobb oldali szélükön egy kis körrel jelölnek. Ez a kapcsolat által szállított információ logikai tagadását jelképezi.



### **A felhasználó által definiált sarkok**

A kapcsolatokon lehetséges a felhasználó által definiált pontok definiálása. Ezek segítségével a felhasználó manuálisan vezérelheti egy kapcsolat útkiválasztását. Ha nincs elhelyezve sarok, akkor az ISaGRAF FBD/LD szerkesztő egy alapértelmezett útkiválasztási algoritmust alkalmaz.

## **A.7.2 Egy FBD diagram beville**

Egy diagram beville során a grafikai területre elemeket (blokkokat, változókat, kontaktokat, tekercseket, stb.) kell elhelyezni, és kapcsolatokat kell közéjük rajzolni.



### **Tárgyak beillesztése**

Egy tárgynak a diagramba történő beillesztéséhez válassza ki az eszközsoron a megfelelő gombot, és kattintson a grafikai területen arra a helyre, ahová be kívánja azt illeszteni.



### **Tárgyak kiválasztása**

A legtöbb szerkesztési parancshoz grafikus tárgyak kiválasztása szükséges. Az ISaGRAF LD/FBD grafikus szerkesztő lehetővé teszi a diagramban levő egy vagy több tárgy kiválasztását. A tárgyak kiválasztásához a szerkesztő eszközsorán ki kell jelölni a „**kiválasztás**” lehetőséget (egy nyilat tartalmazó gomb). Egyetlen tárgy kiválasztásához a felhasználónak csupán rá kell kattintania annak szimbólumára. Több tárgy kiválasztásához az egeret a diagramba kell vonszolni, és ki kell jelölni vele egy négyzet alakú területet. A kiválasztási négyzetben levő összes grafikus tárgy „**kiválasztott**” jelölést kap. Egy kiválasztott tárgyat a grafikus szimbóluma köré rajzolt kis fekete négyzetek jelölnek. Egy új kiválasztás végrehajtásakor az összes előzőleg kiválasztott tárgy kiválasztott állapota megszűnik. A jelenlegi kiválasztás megszüntetéséhez egyszerűen kattintson az egérrel egy, a kiválasztott tárgyakat befoglaló négyzeten kívüli üres területre.



## Megjegyzések beillesztése

A diagramban bárhová elhelyezhetők megjegyzések. A megjegyzések nincsenek hatással a program végrehajtására. Ezek a diagram könnyebb olvashatóságát segítik elő. Egy megjegyzés blokk beillesztéséhez válassza ki annak gombját az eszközsoron, és az egér vonszolásával válassza ki azt a területet, ahová a megjegyzést kell rajzolni. Ezután írja be a megjegyzés szövegét. Egy megjegyzés blokk szövegének beírásakor nem kell alkalmazni semmiféle speciális kezdeti vagy befejező karaktereket, mint pl. a „(“ és a „)”. A megjegyzés blokk méretének megváltoztatásához azt ki kell választani, és a keret sarkát a megfelelő méretűre kell vonszolni az egérrel.



## Tárgyak áthelyezése

A diagramban levő tárgyak áthelyezéséhez először ki kell választani azokat, majd az egérrel a kiválasztott területet a diagram megfelelő helyére kell vonszolni. Csatlakoztatott tárgyak áthelyezéséhez egyszerűen a diagramban levő grafikus szimbólumokat kell áthelyezni. Az ISaGRAF LD/FBD szerkesztő automatikusan átrajzolja az áthelyezett tárgyak közötti csatlakozási vonalakat, azok új helyének megfelelően.



## Kapcsolatok rajzolása

A meglevő elemek csatlakozási pontjai közötti kapcsolat megrajolásához ezeknek a gomboknak az egyikét kell kiválasztani az eszközsoron. Ha a felhasználó egy csatlakozási ponttól a diagramon levő üres helyhez rajzol kapcsolatot, akkor az automatikusan lezáródik egy, a felhasználó által definiált sarokkal azért, hogy tovább folytathassa egy másik szegmens rajzolását.



## A kapcsolati rajz megváltoztatása

Az „Eszközkök / Vonal áthelyezése” parancs egy, a diagramban kiválasztott kapcsolat automatikus útkiválasztásának megváltoztatására szolgál. Ez a parancs hatástalan, ha a kapcsolat egy felhasználó által definiált sarokhoz csatlakozik. Ha egy vonal három szegmensből lett megrajzolva, akkor ez a parancs a második szegmens helyzetét változtatja meg. Az alábbiakban példák láthatók:



## Egy kapcsolat típusának módosítása

A kapcsolat típusa (logikai tagadással vagy a nélkül) egyszerűen megváltoztatható úgy, hogy duplán rákattint a jobb szélére.



## LD létrafokok rajzolása

Egy új LD létrafok megrajolásánál először illessze be a bal oldali áramsínt. Ezután helyezzen el egy tekercset: az automatikusan az áramsínhez lesz kapcsolva. A további kontaktok és függőleges VAGY csatlakozások új csatlakozási kapcsolat megrajolása nélkül, közvetlenül beilleszthetők a létrafok vonalába.

Amikor egy új LD kontaktot vagy tekercset a szerkesztő terület üres helyére illesztenek be, az új vízszintes létrafok vonal automatikusan megrajzolásra kerül az újonnan beillesztett elemtől a jobb és bal oldalon már meglévő áramsínhez. Ez a vonal nem lesz automatikusan megrajzolva, ha az új kontaktot vagy tekercset nem az áramsínek közé helyezik. Ezután az újonnan beillesztett kontakt vagy tekercs szabadon áthelyezhető a megrajzolt létrafokon. A szerkesztő által egy LD kontakt vagy tekercs szimbólum beillesztése során létrehozott vízszintes vonalakat ki lehet választani és törölni. Egy meglévő létrafok vízszintes vonalára egy új LD kontakt vagy tekercs szimbólumot lehet beilleszteni. A szerkesztő automatikusan kivágja a létrafokot és azt az újonnan beillesztett kontakt vagy tekercs bal és jobb oldali csatlakozási pontjaihoz csatlakoztatja.



### **Többszörös kapcsolatok**

Bármelyik **output** pont jobb oldalán létrehozható többszörös csatlakozás. Ez azt jelenti, hogy az információ a diagramban levő több más ponthoz **közvetítve** lesz. Ugyanaz az állapot átvitelre kerül mindegyik jobb oldali szélhez. Egy output csatlakozási pont jobb oldalára korlátlan számú vonal rajzolható. Két csatlakozási vonal jobb oldali szélé nem csatlakoztatható ugyanarra az **input** pontra, a következő LD szimbólumok kivételével:



.....jobb oldali áramsín



.....többszörös csatlakozás a bal (VAGY) operátorhoz

Ezeknek a szimbólumoknak korlátlan számú inputjuk lehet.

## **A.7.3 Egy meglévő diagrammal történő munkavégzés**

A „**Szerkesztés**” menü parancsai egy meglévő diagram megváltoztatására vagy befejezésére szolgálnak. Ezeknek a parancsoknak a legtöbbje a diagramban pillanatnyilag kiválasztott elemekre hat.



### **Egy diagram korrigálása**

A DEL gombbal a kiválasztott elemeket lehet eltávolítani. A kiválasztott elemekkel együtt a függőben levő kapcsolatok is törölődnek. A DEL paranccsal törölt elemek a „**Szerkesztés / Visszavonás**” paranccsal állíthatók vissza. A DEL parancs a diagramban kiválasztott elemcsoportra is alkalmazható. A „**Szerkesztés**” menü „**Kivágás**”, „**Másolás**”, „**Beillesztés**” parancsai a kiválasztott elemek áthelyezésére vagy másolására szolgálnak.



### **Keresés és lecserélés**

A „**Szerkesztés / Keresés**” és „**Szerkesztés / Lecserélés**” menü parancsok a diagramban levő szövegek megkeresésére és lecserélésére szolgálnak. Csak teljes neveket lehet megkeresni. A keresés a kontaktokon, tekercseken, blokk neveken, változókon, és címkéken kerül végrehajtásra. Ez nem használható egy megjegyzés szövegében levő karaktersor megkeresésére. A Lecserélés parancs nem használható egy blokk nevének a lecserélésére. A keresés a pillanatnyi kiválasztás pozíciójától indulva történhet felfelé és lefelé. Ez a diagram határának elérésekor „hurokba lép”.



## A végrehajtási sorrend megjelenítése

Amikor az FBD diagram hátrafelé irányuló hurkokat tartalmaz, a végrehajtás sorrendje nem tudja követni az egyszerű balról-jobbra / felülről-lefelé történő módszert. A zavarok elkerülése érdekében az „**Eszközök / Végrehajtási sorrend megjelenítése**” paranccsal, vagy a **Control + F1** gombok megnyomásával megjeleníthető a programfordításkor alkalmazandó végrehajtási sorrend. A műveletekhez vezető szimbólumok (tekercek, beállított változók és funkcióblokkok) mellett 1-től N-ig számozott jelzőcímkék láthatók.



## Szimbólumok és szövegek bevétele

A megfelelő szimbólum vagy szöveg beviteléhez kattintson duplán az egérrel a kívánt szimbólumra. Ez változókra, kontaktokra illetve tekercsekre, valamint megjegyzés szövegekre és címkekre egyaránt érvényes. Egy kontakt vagy tekercs esetében azok típusának (közvetlen, tagadásos, stb.) megváltoztatása is lehetséges.

A szövegek beviteléhez általában párbeszédablakok szolgálnak, ide értve a változó illetve blokk-kiválasztó listákat is. Ha a „**Beállítások**” menüben ki lett választva a „**Manuális billentyűzet beírás**” mód, akkor a változó szimbólumok és blokknevek beírása közvetlenül egy szövegszerkesztő ablakba történik. Írja be az új szöveget, és annak megerősítéséhez nyomja meg a „**Bevitel**” gombot, vagy nyomja meg a „**Mégsem**” gombot a módosításból való kilépéshez és a szövegszerkesztő ablak bezárásához. A „manuális billentyűzet beírás” módhoz használt szövegszerkesztő ablakot az egérrel nem lehet bezárni.

Ha a „**Beállítások**” menüben az „**Automatikus beírás**” mód lett kiválasztva, akkor a változó szimbólumot közvetlenül minden új kontakt vagy tekercs beillesztése után be kell írni. A szimbólumot egy változó vagy címke beillesztését követően mindig azonnal be kell írni.



## Funkcióblokk típus kiválasztása

Egy blokk típusának megváltoztatása az egérrel a blokkra történő dupla rákattintással történik. A blokk típusa a rendelkezésre álló operátorok, funkciók, és funkcióblokkok listájából választható ki. Ez a parancs lehetővé teszi az input pontok számának megváltoztatását is egy kommutatív operátor esetében. (pl. ÉS, VAGY, ÖSSZEAD, SZOROZ, stb.)



## Szabad hely létrehozása

Az FBD rajzolási területen az egér jobb oldali gombjának megnyomására megjelenik egy előbukkanó menü. Ez a következő parancsokat tartalmazza, amelyek segítségével a kurzor helyénél szabad terület illeszthető be, illetve távolítható el:

**Sorok beillesztése:** Ez a parancs vízszintes szabad területet illeszt be, amely a rácsköznek megfelelő 4 sorból áll, és a felbukkanó menü kurzorhelyétől indul.

**Sorok törlése:**....Ez a parancs használaton kívüli vízszintes szabad területet (sorokat) távolít el, amely a felbukkanó menü kurzorhelyétől indul. Ez a parancs nem használható FBD elemek eltávolítására.

Amikor nyitva van a felbukkanó menü, az FBD rajzban levő szürke vonal mutatja, hogy az üres terület hol lesz beillesztve, illetve honnan lesz eltávolítva.

#### A.7.4 Megjelenítési lehetőségek

A „**Beállítások**” menü parancsai a képernyőn levő FBD diagram rajzának testre szabására szolgálnak.



##### Az elrendezés testre szabása

A „**Beállítások / Elrendezés**” parancs egy párbeszédablakot nyit ki, amelyben a szerkesztő munkaterülettel és a grafikus diagramban levő rajzzal kapcsolatos összes paraméter illetve beállítási lehetőség van csoportosítva. A „Munkaterület” csoport ablakban levő jelölőnégyzet a szerkesztő eszközsorainak és állapotsorának megjelenítésére vagy elrejtésére szolgál. A „Dokumentum” csoport beállítási lehetőségei a szerkesztőrács pontjainak megjelenítését illetve elrejtését teszik lehetővé.



A „Zoom” csoport beállítási lehetőségei egy fő zoom arány kiválasztását teszik lehetővé. Az alapértelmezett zoom arányok közötti átváltás a szerkesztő eszközsorban levő „zoom” gombbal is lehetséges.

A „**Beállítások / Fontok**” parancssal az összes ISaGRAF grafikus dokumentumban használandó karakterfont neve választható ki. A font kiválasztásakor a font stílusának és méretének nincs jelentősége, ezért azokat nem kell specifikálni. Az ISaGRAF grafikus szerkesztő mindig a kiválasztott zoom aránynak megfelelően számítja ki a font méretet.

#### A.7.5 Stílusok és a módosítások nyomon követése

Az ISaGRAF LD/FBD szerkesztő lehetővé teszi egy LD/FBD diagram bármely komponenséhez egy grafikus stílus kijelölését. A stílus elsősorban speciális diagram színezést jelent. Az ISaGRAF emellett azonban a verziók kezelése céljából a diagramok módosításainak nyomon követéséhez is stílusokat alkalmaz.

Megjegyzendő, hogy szimuláció vagy Online hibakeresés közben a stílusok nem láthatók, mivel ezekben a módokban a színek (piros és kék) a kémlelt változók IGAZ / HAMIS állapotának kiemelésére szolgálnak.



##### **Előre meghatározott stílusok**

A következő stílusok előre meg lettek határozva:

**Normál** ..... Alapértelmezett rajz (fekete). A módosítások nyomon követéséhez a „normál” stílus azt jelzi, hogy az ezzel a stílussal rendelkező elemek az eredeti diagram részét képezik. A „Normál” stílusú elemek általában a végrehajtás közben kerülnek leolvasásra.

- Módosított** .....A „módosított” jelölésű elemek rózsaszínűek. A módosítások nyomon követéséhez a „módosított” stílus azoknak az elemeknek a kiemelésére szolgál, amelyek a diagram eredeti kiadása után lettek hozzáadva illetve megváltoztatva. A „Módosított” stílusú elemek általában a végrehajtás közben kerülnek leolvasásra.
- Törölt** .....A „törölt” jelölésű elemek szürke színűek, szaggatott vonalakkal. Az ilyen elemek a diagram végrehajtása közben figyelmen kívül vannak hagyva. Ez a stílus az eredeti kiadás után eltávolított elemek nyomon követésére szolgál olyankor, amikor verziók kezelése szükséges.
- Testre szabott**....Az ISaGRAF LD/FBD szerkesztő az előre meghatározott stílusok mellett lehetővé teszi a diagram valamelyik részéhez bármely szín alkalmazását. Az ilyen elemek „Testre szabott” stílusúaknak számítanak. A „Testre szabott” stílus használata nincs hatással a diagram végrehajtására a futtatás során.

Egy stílusnak a kiválasztott elemekre történő alkalmazására a „Szerkesztés” menüben levő „Stílus” almenü szolgál.

## **Módosítások nyomon követése**

A stílusok használata és a „Törölt” stílus rendelkezésre állása lehetővé teszi egy meglevő diagramban az automatikus módosítás nyomon követést. A módosítások nyomon követésének engedélyezésére illetve letiltására a „**Szerkesztés / Stílus**” menü „**Módosítások bejelölése**” parancs szolgál.

Amikor a „Módosítások bejelölése” lehetőség be van állítva, valamennyi, a diagramhoz adott illetve megváltoztatott elem automatikusan a „Módosított” stílusra lesz beállítva. Amikor egy elem a „Törlés” vagy „Kivágás” parancsral van törölve, az nem lesz vizuálisan eltüntetve a diagramból, hanem egyszerűen „Törölt” stílusra lesz beállítva. Ez lehetővé teszi a felhasználó számára, hogy automatikusan nyomon kövessen minden, a diagramba bevitt módosítást.

A „Törölt” stílusra beállított elemeknek az LD/FBD diagramból történő tényleges eltávolításához a „**Szerkesztés/Stílus/Az összes törölt tétel eltávolítása**” parancsot kell használni. Ezt a parancsot a pillanatnyi kiválasztás nem befolyásolja, és az mindig az egész diagramra vonatkozik.

Egy „Törölt” stílusra beállított elem „visszaállításához” válassza ki a kívánt elemet, és állítsa be hozzá a „**Normál**”, „**Módosított**”, vagy valamelyik „**Testre szabott**” stílust. Az ilyen művelet érvénytelen csatlakozásokhoz (ugyanahhoz az input ponthoz egynél több kapcsolat van csatlakoztatva) vezethet, ami a következő program megerősítés során lesz érzékelve.

## A.8 A szövegszerkesztő használata

Ez a fejezet csak az ISaGRAF szövegszerkesztő jellemzőit és parancsait ismerteti, elsősorban az ST és IL programok forráskódjának beírásához.

### A.8.1 Szerkesztő parancsok

Az „**Eszközök**” menü parancsai a szerkesztett szövegen történő munkavégzésre használhatók: Ezeknek a parancsoknak a legtöbbje a diagramban pillanatnyilag kiválasztott karakterekre hat, vagy a beszúrási jel pillanatnyi helyén hajt végre egy műveletet.



#### **Kivágás és beillesztés**

A DEL gombbal a kiválasztott szöveget lehet eltávolítani. A DEL paranccsal törölt elemek a „**Szerkesztés / Visszavonás**” paranccsal állíthatók vissza. A „**Szerkesztés**” menü „**Kivágás**”, „**Másolás**”, „**Beillesztés**” parancsai a kiválasztott szöveg áthelyezésére vagy másolására, illetve más alkalmazás által a Vágólapra másolt szövegrészek beillesztésére szolgálnak.



#### **Keresés és lecserélés**

A „**Szerkesztés / Keresés**” és „**Szerkesztés / Lecserélés**” menü parancsok a programban levő szövegek megkeresésére és lecserélésére szolgálnak. Bármilyen karaktersort meg lehet keresni. A keresés végrehajtható felfelé vagy visszafelé, a beszúró jel pillanatnyi helyéről indulva. Ez nem lép „hurokba” a program határának elérésekor.



#### **Ugrás sorra**

A felhasználó a „**Szerkesztés / Ugrás sorra**” parancs segítségével a beszúró jelet egy adott számú sorra viheti. Ez nagyon hasznos lehet egy, az ISaGRAF programfordító által egy ST vagy IL programban észlelt, és számmal hivatkozott hibát tartalmazó sorhoz történő hozzáférésre.



#### **Szimbólum beillesztése a szótárból**

A „**Szerkesztés / Változó beillesztése**” parancs a beszúró jel helyénél egy, a projekt szótárában deklarált változó vagy tárgy szimbólumának beillesztésére szolgál. A szimbólum az ennek a dokumentumnak a „Programszerkesztőkről szóló további tudnivalók” című fejezetében ismertetett közös változóválasztó ablakból választható ki.



#### **Fájl beillesztése**

A „**Szerkesztés / Fájl beillesztése**” parancs egy fájl teljes tartalmát illeszti be a beszúró jel pillanatnyi helyére. Megjegyzendő, hogy csak tisztán ASCII szövegfájlok kezelhetők ezzel a paranccsal.

## A.8.2 Beállítási lehetőségek

A „**Beállítások**” menü parancsai a szerkesztő eszközsorainak megjelenítésére vagy elrejtésére, és a karakterfont kiválasztására szolgálnak. A kiválasztott karakterfont lesz alkalmazva az ISaGRAF Workbench-ben történő összes szövegszerkesztéshez.

A „**Beállítások / Kulcsszavak megjelenítése**” parancs egy ST/IL program forráskódjának bevitelénél egy eszköztáblak megjelenítésére vagy elrejtésére szolgál, amely az ST vagy IL nyelv leggyakoribb kulcsszavait csoportosítja. A megfelelő kulcsszavának vagy operátorának a beszűrő jel pillanatnyi helyénél történő beillesztéséhez kattintson az eszközsorban levő gombra.

## A.9 További programszerkesztőkkel kapcsolatos tudnivalók

Ez a fejezet az összes ISaGRAF programszerkesztőre tekintve közös szerkesztési lehetőségekről nyújt hasznos információkat. Ez főleg a többi ISaGRAF eszközzel való kapcsolódásra, valamint az általános ISaGRAF párbeszédablakokra vonatkozik.

### A.9.1 Egyéb ISaGRAF eszközök meghívása



#### **A program megerősítése (fordítás)**

A „**Fájl / Megerősítés**” parancs az ISaGRAF kódgenerátort futtatja a pillanatnyilag szerkesztés alatt álló program programozási szintaxisának megerősítésére. Az SFC nyelv esetében mind az 1. szint, mind a 2. szint ellenőrzésre kerül. A szintaxis megerősítés befejezése után a programon történő munka továbbfolytatásához be kell zárni a kódgenerátor ablakot. Ha az alkalmazásban csak egy program van (a szerkesztés alatt álló program), akkor az alkalmazás kódja generálásra kerül amennyiben nem volt érzékelhető szintaxis hiba. A „**Beállítások / Programfordítási beállítások**” parancs a programfordítási és optimalizálási paraméterek beállítására szolgál. A programfordítással és kódgenerálással kapcsolatos további információk ennek a dokumentumnak „A kódgenerátor használata” című fejezetében találhatók.



#### **Az alkalmazás szimulációja vagy hibakeresése**

A „**Fájl / Szimuláció**” és „**Fájl / Hibakeresés**” parancsok az ISaGRAF grafikus hibakeresőt futtatják vagy szimulációs, vagy valós csatlakoztatott módban, és hibakeresési módban újra megnyitják a szerkesztett SFC programot. A hibakeresési módban a programba nem vihetők be módosítások.



#### **A változószótár szerkesztése**

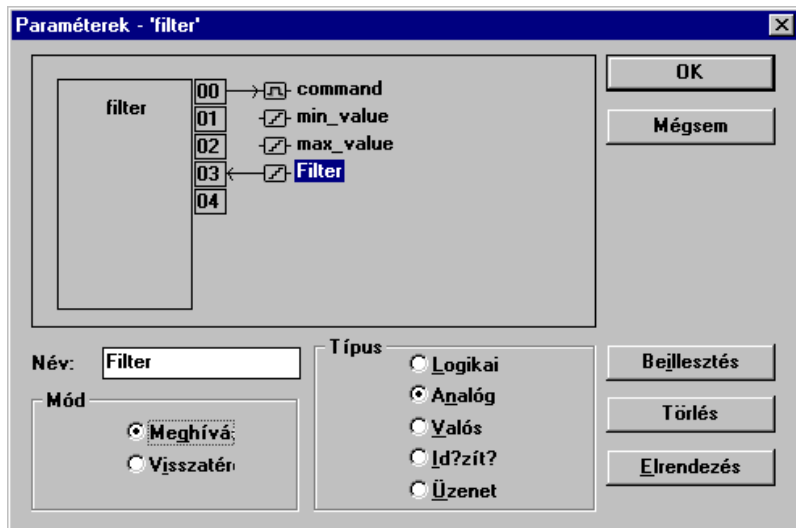
A „**Fájl / Szótár**” parancs a pillanatnyi alkalmazás és pillanatnyi program változó szótárának a szerkesztésére szolgál. Ez a felhasználó által definiált szavak szerkesztéséhez szükséges belépési pontokat is tartalmazza. A **lokális** deklarációk vagy definiált szavak a pillanatnyilag szerkesztés alatt álló programra vonatkoznak.

### A.9.2 A program paraméterei

Amikor a szerkesztett program egy funkció, funkcióblokk, vagy egy alprogram, akkor annak meghívási és visszatérési paramétereinek definiálására a „**Fájl / Paraméterek**” parancs szolgál. Ez a parancs hatástalan akkor, ha a szerkesztett program egy SFC vagy felső szintű program a **Kezdet** vagy **Befejezés** szekcióból.

Az alprogramoknak, funkcióknak, vagy funkcióblokkoknak maximum **32** paramétere (input vagy output) lehet. Egy funkciónak vagy alprogramnak mindig egy (és csak egyetlen) visszatérési paramétere van, amelynek a neve a funkció nevével megegyező kell, hogy legyen annak érdekében, hogy eleget tegyen az ST nyelv

írási konvencióinak. A következő párbeszédablak az alprogram paramétereinek a leírására szolgál:



Az ablak bal felső oldalán látható lista a paramétereket mutatja, az alábbi meghívási modell szerinti sorrendben: először a meghívó paraméterek, utoljára pedig a visszatérési paraméterek. Az ablak alsó része a listában pillanatnyilag kiválasztott paraméter részletes leírását mutatja. Egy paraméterhez bármelyik ISaGRAF adat típus használható. A visszatérési paramétereknek a meghívó paraméterek után kell elhelyezkedniük a listában. A paraméterek elnevezéseinek a következő szabályoknak kell eleget tenniük:

- a név hossza nem haladhatja meg a 16 karaktert
- az első karakternek betűnek kell lennie
- a többi karakternek betűnek, számjegynek, vagy aláhúzás karakternek kell lennie
- az elnevezés nem érzékeny a kisbetű/nagybetű beállításra

A „**Beillesztés**” parancs egy új paraméternek a kiválasztott paraméter elé történő beillesztésére szolgál. A „**Törlés**” parancs a kiválasztott paraméter törlésére szolgál. Az „**Elrendezés**” parancs automatikusan átrendezi (sorrendbe állítja) a paramétereket úgy, hogy a visszatérési paraméterek a lista végére kerülnek.

### A.9.3 A „Fájl” menü egyéb parancsai

Valamennyi programszerkesztő „Fájl” menüjében a következő parancsok állnak rendelkezésre:



#### **Másik program megnyitása**

A felhasználó a „Fájl / **Megnyitás**” parancs segítségével bezárhatja a pillanatnyilag szerkesztés alatt álló programot, és elkezdheti a pillanatnyi projekt egy ugyanolyan

nyelvű másik programjának a szerkesztését. Ez a funkció nem használható egy másik nyelven írt program szerkesztésére. Az újonnan kiválasztott program felváltja a szerkesztő ablakban a pillanatnyilag szerkesztett programot.



### A program nyomtatása

A **„Fájl / Nyomtatás”** parancs a szerkesztett programot a nyomtatóra küldi. Ez a parancs automatikusan futtatja az ISaGRAF dokumentum generátort, a szerkesztett program és a csatolt helyi változók kinyomtatására.

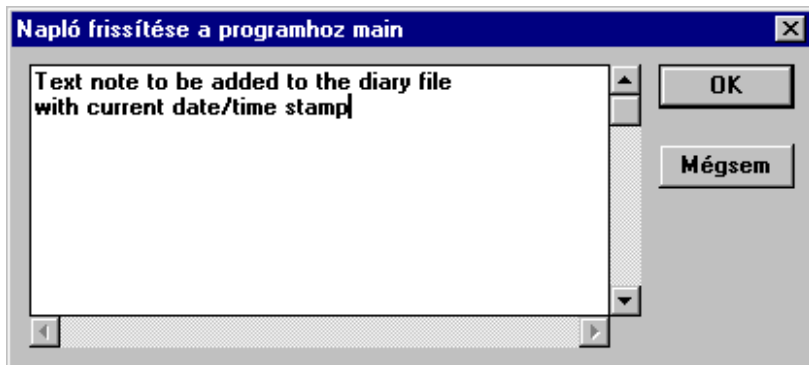
Bizonyos grafikus programoknál (SFC, FBD és Gyors LD) a **„Szerkesztés / Rajz másolása”** parancsot is lehet használni a diagram rajzának metafile formátumban a vágólapra történő másolásához, ahonnan az más alkalmazásokba, pl. szövegszerkesztőkbe beilleszthető. Az SFC programoknál a kimásolt metafile-ban csak 1. szintű információk (diagram, számozás, és 1. szintű megjegyzések) jelennek meg.

#### A.9.4 A program-napló frissítése

A szerkesztett programhoz csatolt naplófájl a **„Fájl / Napló”** parancssal írható be manuálisan. A naplófájl a program minden fordításakor automatikusan frissítődik szintaxis ellenőrző output üzenetekkel. A programfordítási outputok a programfordítás dátumával és időpontjával vannak ellátva.



Ha a programszerkesztők **„Beállítások”** menüjéből a **„Napló frissítés”** mód ki lett választva, akkor a program lemezre mentésekor minden alkalommal a következő párbeszédablak jelenik meg.

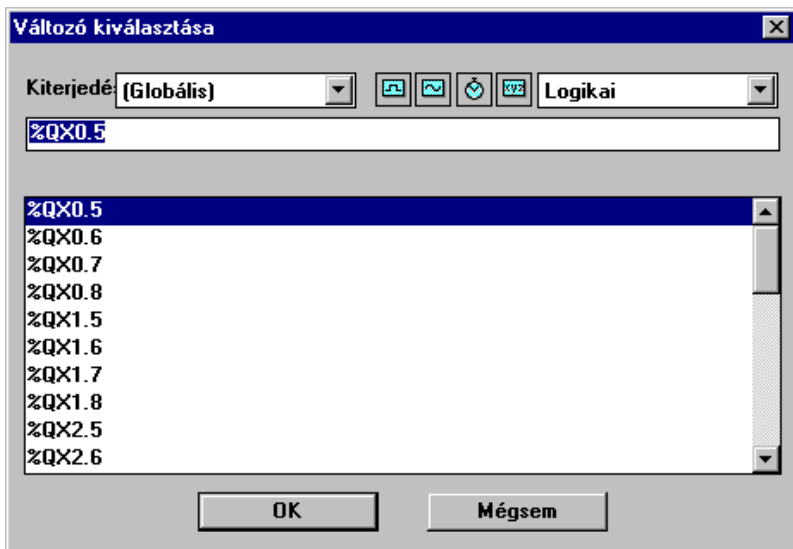


Az OK gomb megnyomására aztán a naplófájl végére elmentésre kerül a beírt szöveges megjegyzés a pillanatnyi dátum / időpont mutatóval ellátva. Ez a teljes programok karbantartásához nagyon hasznos, mivel hasznos információkat nyújt a program élettartamáról.

## A.9.5 Egy változó kiválasztása a szótárból



Egy szöveges program (ST vagy IL) szerkesztésekor a „**Szerkesztés / Változó beillesztése**” lehetővé teszi egy kiválasztott deklarált változó név beillesztését a beszűrő jel pillanatnyi helyére. LD vagy FBD programok szerkesztésénél a kontaktok, tekercsek, blokk I/O paraméterek, vagy FBD változó ablakok leírásához változó kiválasztása szükséges. Egy deklarált változó kiválasztásához mindkét esetben a következő párbeszédablak nyílik ki:



A globális és lokális változók közötti választás a „Kiterjedés” kiválasztó ablakkal történik. A jobb oldalon levő kiválasztó ablak az adattípus kiválasztását teszi lehetővé. A típus kiválasztási ablak mellett levő kis ikonokat a legutóbbi adattípusok kiválasztására lehet használni parancsikonokként:



.....Logikai



.....Integer / Valós



.....Időzítő



.....Üzenet

Egy változó kiválasztásához kattintson annak nevére a listában. Ekkor annak neve és a hozzá tartozó megjegyzés a lista elején jelenik meg. Az „OK” gomb megnyomása a kiválasztást megerősíti. Lehetőség van egy változó nevének a szerkesztés vezérlőben történő közvetlen beírására is, a lista használata nélkül.

## A.9.6 Az output ablak

Valamennyi nyelv-szerkesztő Eszközök menüjében a következő parancsok állnak rendelkezésre: Ezek információ megjelenítésére szolgálnak kisbetűs szöveg formájában a szerkesztőablak alján, és a programban való böngészésre használhatók:

- |  |  |
|--|--|
| <b>„Programfordító output megjelenítése”</b> | a szerkesztett program legutóbbi programfordításának hibaüzeneteit jeleníti meg az output ablakban.  |
| <b>„Keresés ...-ben”</b>                     | egy szöveg előfordulásait megkeresi a teljes szerkesztett programban, és azokat az output ablakban felsorolja. Az SFC és FC nyelveknél ez a parancs valamennyi 2. szintű programban keres. |
| <b>„Output ablak elrejtése”</b>              | bezárja az output lista ablakot  |

Ha az output ablakban hibaüzenetek vagy előfordulások vannak megjelenítve, akkor egy sorra duplán rákattintva a megfelelő helyen közvetlenül beállítódik a látható kiválasztás. Az SFC és FC nyelveknél ez a parancs megnyitja a megfelelő 2. szintű programozási ablakot.

## A.10 A szótárszerkesztő használata

Az ISaGRAF szótár az alkalmazás belső változóinak, I/O változóinak, funkcióblokk előfordulásainak, és „definiált szavainak” a deklarálására szolgáló szerkesztő eszköz. A szótár az alkalmazás deklarált változóit és funkcióblokk előfordulásait, valamint a konstans karaktersorokként definiált szavakat csoportosítja.

A változókat, funkcióblokkokat és definiált szavakat a forráskódban történő felhasználás előtt a szótárban deklarálni kell. A változók és definiált szavak az alábbi automatizálási nyelvek bármelyikével használhatók: SFC, FBD, LD, ST és IL. Az FBD nyelvben használt funkcióblokkokat nem kell deklarálni, mert az ISaGRAF FBD és Gyors LD szerkesztők automatikusan deklarálják a használt blokkok előfordulásait.



### Változók

A változók **tartományaik** és **típusaik** szerint vannak osztályozva. Ugyanazon az inputrácson csak ugyanolyan típusú és tartományú változók vihetők be. A változók alapvető tartományai a következők:



**GLOBÁLIS** ...a pillanatnyi projekt bármelyik programja által használhatók



**LOKÁLIS** .....csak egy program által használhatók

A változók alapvető típusai a következők:



**LOGIKAI** .....igaz/hamis bináris értékek



**ANALÓG** .....valós vagy integer értékek



**IDŐZÍTŐ** .....időértékek



**ÜZENET** .....karaktersorok

Egy változót névvel, megjegyzéssel, attribútummal, hálózati címmel, és egyéb specifikus mezőkkel lehet azonosítani. Az alapvető változó attribútumok a következők:

**BELSŐ** .....memória változó

**INPUT** .....egy inputeszközhöz kapcsolódó változó

**OUTPUT** .....egy outputeszközhöz kapcsolódó változó

**KONSTANS** .....csak olvasható belső változó (kezdeti értékkel)

Megjegyzés: Az **időzítők** mindig **belső** változók. Az **input** és **output** változók mindig **GLOBÁLIS** tartománnyal rendelkeznek.



### Definiált szavak

A definiált szó egy olyan álnevet jelent, amelyet bármelyik nyelvben egy szöveges karaktersor kicserélésére lehet használni. A kicserélt szöveg lehet egy változónév, egy állandó kifejezés, vagy egy összetett kifejezés. A definiált szavak tartományaik szerint vannak osztályozva. Ugyanazon az inputrácson csak ugyanolyan típusú és tartományú definiált szavak vihetők be. Az alapvető tartományok az alábbiak:



**KÖZÖS** .....bármelyik projekt bármelyik programja által használhatók



**GLOBÁLIS** ...a pillanatnyi projekt bármelyik programja által használhatók



**LOKÁLIS** .....csak egy program által használhatók

Egy definiált szót névvel, egy jól definiált ST szöveg előfordulás blokkal, és egy szabad megjegyzéssel lehet azonosítani.



### Funkcióblokk előfordulások

Az ST és IL nyelvekben használt funkcióblokkok előfordulásait a szótárban deklarálni kell. Mivel egy funkcióbloknak belső „rejtett” adatai vannak, ezért egy funkcióblokk minden példányát azonosítani kell. A következő példa a könyvtárban definiált „R\_TRIG” (felfutó él detektálás) funkcióblokkot mutatja, amely különböző változók éldetektálására szolgál. A blokk minden példányát egy egyedi névvel kell azonosítani. A blokk típusának elnevezése és paramétereinek definiálása a könyvtárrendező használatával történik:

**Blokk neve:** R\_TRIG  
**Paraméterek:** Input=CLK  
Output=Q

Az előfordulások elnevezése a szótárszerkesztővel történik:

**Előfordulás neve:** TRIG\_B1                      **Blokk neve:** R\_TRIG  
**Előfordulás neve:** TRIG\_B2                      **Blokk neve:** R\_TRIG

A deklarált előfordulások ST programokban használhatók:

```
TRIG_B1 (b1);
edge_b1 := TRIG_B1.Q;  (* b1 változó éldetektálás *)
TRIG_B2 (b2);
edge_b2 := TRIG_B2.Q;  (* b2 változó éldetektálás *)
```

A deklarált funkcióblokk előfordulások lehetnek **GLOBÁLISAK** (amelyeket a projektben levő valamennyi program ismer), vagy **LOKÁLISAK** (a projektben levő egyetlen program számára). Az FBD vagy LD nyelvben használt funkcióblokkokat nem kell deklarálni, mert az ISaGRAF FBD szerkesztő automatikusan deklarálja a használt blokkok előfordulásait.



(\* A funkcióblokkok neve mindig megegyezik a könyvtárban definiált blokk nevével. Az ISaGRAF FBD és Gyors LD szerkesztők automatikusan egy előfordulást deklarálnak valahányszor egy blokk beillesztésre kerül a diagramban \*)

Az FBD és Gyors LD szerkesztők által automatikusan deklarált funkcióblokk előfordulások a szerkesztett programhoz mindig **LOKÁLISAK**.



## Hálózati címek

A hálózati címek **opcionálisak**. Egy nem zéró hálózati címmel rendelkező változót egy külső rendszer (pl. egy folyamatmegjelenítő rendszer) a futtatás során **kémleni** tud. Általánosabban kifejezve: a hálózati cím egy azonosító mechanizmust tesz lehetővé minden olyan futtatási kommunikációs rendszer számára, amely nem tud szimbolikus neveket kezelni. Egy változóhoz hálózati cím a változó teljes leírása során, a változó létrehozásakor illetve módosításakor írható be.

### A.10.1 A fő szótárablak

A szótárszerkesztő ablak azonos típusú és tartományú változók listáját mutatja. A szerkesztett változók típusa és tartománya mindig a címsorban látható.



A szerkesztőablak csak a változó leírásának fő mezőit mutatja: név, attribútum és hálózati cím, valamint szöveges megjegyzés. A kiválasztott változó teljes leírása mindig az állapotsorban látható. A szerkesztendő változó tartomány kiválasztására a következő gombokat kell használni az eszközsorról:



**KÖZÖS** ..... bármelyik projekt bármelyik programja által használhatók



**GLOBÁLIS** ..... a pillanatnyi projekt bármelyik programja által használhatók



**LOKÁLIS** ..... csak egy program által használhatók

A szerkesztendő tárgy típusának kiválasztására a címsorral megjelenített „Tab” parancsot kell használni:

Logikaiak	Integerek/Válósók	Id?zítők	Üzenetek	FB előfordulások	Definiált szavak
Név	Attrib.	Cím	Megjegyzés		



Egy változó előtag nevének megkereséséhez az eszközsortól balra levő szövegbeviteli ablakot kell használni. Ebben az esetben a keresés az egész listára vonatkozóan, annak elejétől kezdve kerül végrehajtásra, a pillanatnyi kiválasztásnak megfelelően. A változó neveken és megjegyzésekben történő szöveges karakteresor keresésre, valamint a kiválasztás erre a változóra történő átváltására a „**Szerkesztés / Keresés**” parancs is használható. A keresés **soha sem érzékeny a kisbetű/nagybetű beállításra**.

### A.10.2 Változók kezelése

A „**Fájlok**” menü rendelkezésre álló parancsai a változók, funkcióblokk előfordulások, vagy definiált szavak teljes kiválasztott osztályára vonatkozóan érvényesek. A szerkesztendő tárgyak típusának és tartományának kiválasztására az „**Egyéb**” parancsot kell használni:



## Változók nyomtatása

A jelenleg szerkesztés alatt álló változók vagy definiált szavak listájának egy standard Windows™ nyomtatón történő kinyomtatására a „**Fájlok / Nyomtatás**” parancsot kell használni. A nyomtatás az ISaGRAF dokumentum generátor használatával történik. A kinyomtatott példány tartalmazza minden, a jelenleg szerkesztés alatt álló típusnak megfelelő változó vagy definiált szó teljes leírását.



## Új változók létrehozása

A felhasználó a „**Szerkesztés / Új**” parancs segítségével hozhat létre új változókat, funkcióblokk előfordulásokat, vagy definiált szavakat a kiválasztott tartományhoz és típushoz. Az új változók közvetlenül az elé a változó elé lesznek beillesztve, amelyre pillanatnyilag a kiválasztósor mutat. Ennek a parancsnak a futtatásakor megnyílik egy adatbeviteli ablak, a változó leírásának beírásához. Ha a leírás bevitele megtörtént, akkor a „**Tárolás**” gomb megnyomására az a listára kerül. A beviteli ablak automatikusan kinyílik, hogy a felhasználó ugyanazzal a „**Szerkesztés**” parancssal más változókat is be tudjon írni. A párbeszédablak „**Mégsem**” gombjának megnyomása félbeszakítja a változó létrehozásának folyamatát.



## Létező változók módosítása

A felhasználó a „**Szerkesztés**” menü „**Szerkesztés**” parancsa segítségével módosíthatja a kiválasztósor által pillanatnyilag jelzett változó leírását. Ennek a parancsnak a futtatásakor megnyílik egy adatbeviteli ablak, a változó leírásának módosításához. Ha a leírás bevitele megtörtént, akkor a „**Tárolás**” gomb megnyomása elfogadtatja a módosítást. A felhasználó a „**Következő**” és „**Előző**” gombok megnyomásával a módosító parancsot a szomszédos változókra is kiterjesztheti. A „**Mégsem**” gomb megnyomása bármilyen módosítás tárolása nélkül be zárja a párbeszédablakot.



## Kivágás és beillesztés

Az ISaGRAF szótárszerkesztő eszköz lehetővé teszi **több sor kiválasztását**. A jelenleg szerkesztés alatt álló változólistán történő munkavégzéshez számos parancs áll rendelkezésre. Az alábbiakban a „**Szerkesztés**” menü rendelkezésre álló parancsai vannak felsorolva:

**MÁSOLÁS** .....A kiválasztott változócsoporthoz a szótár vágólapjára másolja  
**KIVÁGÁS** .....A kiválasztott változócsoporthoz a szótár vágólapjára másolja, és egyúttal azt eltávolítja a szerkesztett listából  
**TÖRLÉS** .....A kiválasztott változócsoporthoz eltávolítja a szerkesztett listából  
**BEILLESZTÉS** ...A szótár vágólapjának tartalmát a kiválasztott változó elé illeszti be

A Másolás/Kivágás/Beillesztés funkciók különböző változólisták között is lehet alkalmazni. Ezek azonban különböző típusú tárgyak listái között nem használhatók.



## Változók osztályozása

Az „**Eszközök / Osztályozás**” parancs a jelenleg szerkesztés alatt álló lista változóit vagy definiált szavait osztályozza. A besorolás sorrendjét a változók attribútumai határozzák meg.

- először a belső változók
- Aztán az input változók
- végül az output változók

Az ugyanolyan attribútummal rendelkező változók alfabetikus sorrend szerint kerülnek osztályozásra. A definiált szavak mindig alfabetikus sorrend szerint vannak besorolva.



### Hálózati címek beállítása

A hálózati címek **opcionálisak**. Egy nem zéró hálózati címmel rendelkező változót egy külső rendszer (pl. egy folyamatmegjelenítő rendszer) a futtatás során **kémlelni** tud. Egy változóhoz hálózati cím a változó teljes leírása során, a változó létrehozásakor illetve módosításakor írható be. A felhasználó az „**Eszközök / Címek átszámozása**” parancs segítségével állíthat fel hálózati címeket egy egész változócsoporthoz számára. Ez a parancs futtatásakor a listán pillanatnyilag kiválasztott változócsoporthoz hat. Egy hexadecimális **alapcím** (a csoport első változójához rendelt cím) bevitelével a változócsoporthoz **egymás után következő címeket** fog kapni a hálózati címek beállításakor. Nulla alapcím bevitel az összes kiválasztott változó hálózati címét nullára állítja vissza.



### Logikai „igaz/hamis” karaktersorok importálása

Definiált szavak szerkesztése során a felhasználó az „**Eszközök / Igaz/Hamis definíciók importálása**” parancs segítségével az IGAZ és HAMIS állapotok képviseléséhez a logikai változókhoz csatolt karaktersorokat automatikusan nyelvi kulcsszavakként definiálhatja. Az ilyen karaktersorok általában a formattálás hibakereséséhez vannak definiálva. Amennyiben ezeket programokban kívánják használni, úgy definiált szavakként kell őket kinevezni. Ez a parancs logikai igaz/hamis karaktersorokat keres azokban a deklarációkban, amelyeknek tartománya megegyezik a definiált szavak szerkesztéséhez pillanatnyilag kiválasztott tartománnyal.

## A.10.3 Tárgyak leírása

Minden változóhoz, funkcióblokk előforduláshoz, vagy definiált szóhoz egy teljes leírást kell beírni. A leírás mezők az egyes tárgytípusokhoz különbözőek. Az alábbi mezők minden változótípushoz közzösek:

- Név** ..... A változó neve: az első karakternek betűnek kell lennie, a többi karakternek betűnek, számjegynek, vagy aláhúzás („\_”) karakternek kell lennie.
- Hálózati cím** ..... Hexadecimális hálózati cím (opcionális). Ha ez a mező nem nulla, akkor a változót a futtatásakor külső rendszerek kémlelhetik.
- Megjegyzés** ..... A változó leírására vonatkozó szabad megjegyzés.
- Megtartás** ..... Ez a beállítási lehetőség azt jelzi, hogy a változót biztonsági másolatként ki kell menteni.



Az alábbiakban egy **logikai** változó egyéb leíró mezői láthatók:

- Attribútum** ..... Egy belső, konstans, input vagy output változót specifikál.

„Hamis” **karaktorsor** ..... A hibakeresés idején hamis értékként alkalmazott karaktorsor  
 „Igaz” **karaktorsor** ..... A hibakeresés idején igaz értékként alkalmazott karaktorsor  
**Kezdetként igazra állítva** ..... Ha ez a beállítás ki van jelölve, akkor a kiindulási érték IGAZ, máskülönben a kiindulási érték HAMIS.



Az alábbiakban egy **integer vagy valós** változó egyéb leíró mezői láthatók:

**Attribútum** ..... Egy belső, konstans, input vagy output változót specifikál.  
**Formátum** ..... Egy integer vagy valós (lebegő) változót specifikál. A hibakeresés közben használt megjelenítési forma kiválasztható.  
**Egység karaktorsor** ..... A hibakeresés idején a fizikai egység azonosítására alkalmazott karaktorsor.  
**Konverzió** ..... A változóhoz csatolt konverziós táblázat vagy konverziós funkció neve (csak input vagy output változókhoz)  
**Kezdeti érték** ..... A változó kezdeti értéke (ugyanolyan formátumúnak kell lennie, mint a változó). Ha nincs specifikálva, akkor a kezdeti érték 0.



Az alábbiakban egy **időzítő** változó egyéb leíró mezői láthatók:

**Attribútum** ..... Egy belső vagy konstans változót specifikál.  
**Kezdeti érték** ..... A változó kezdeti értéke (időérték). Ha nincs specifikálva, akkor a kezdeti érték idő#0s.



Az alábbiakban egy **üzenet** változó egyéb leíró mezői láthatók:

**Attribútum** ..... Egy belső, konstans, input vagy output változót specifikál.  
**Maximális hossz** ..... Az üzenetben tárolható maximális karakterszámot határozza meg.  
**Kezdeti érték** ..... A változó kezdeti értéke (hossza nem haladhatja meg az üzenet kapacitását). Ha nincs specifikálva, akkor a kiindulási érték az üres karaktorsor.



Az alábbiakban egy **definiált szó** változó egyéb leíró mezői láthatók:

**Név** ..... Az ST forrásfájlokban használt név: az első karakternek betűnek kell lennie, a többi karakternek betűnek, számjegynek, vagy aláhúzás („\_”) karakternek kell lennie.  
**Definiálás** ..... Az ST szintaktikának megfelelő karaktorsor, amely a fordítás közben a definiált szót felváltja. Példa: Név = PI - Ekvivalencia = 3.14159  
**Megjegyzés** ..... A definiált előfordulásra vonatkozó szabad megjegyzés.



Az alábbiakban egy **funkcióblokk előfordulás** leíró mezői láthatók:

- Név** ..... Az ST forrásfájlokban használt előfordulás neve: az első karakternek betűnek kell lennie, a többi karakternek betűnek, számjegynek, vagy aláhúzás („\_”) karakternek kell lennie.
- Típus** ..... A könyvtárban levő megfelelő funkcióblokk neve.
- Megjegyzés** ..... A funkcióblokk előforduláshoz szolgáló szabad megjegyzés.

## A.10.4 Gyors deklaráció

Az **„Eszközök / Gyors deklaráció”** parancs lehetővé teszi több változónak egy időben történő deklarációját. A gyors deklarációval létrehozott változók elnevezése egy számozási konvenció alkalmazásával történik. Ehhez a következőket kell definiálni:

- az első és utolsó változó indexét (számát),
  - a változó szimbólumokban levő számok után hozzáadandó szöveget
  - a változó szimbólumokban levő szám kifejezésére használt számjegyek számát.
- Ezen felül specifikálni lehet a létrehozott változók (belső, input vagy output, stb.) alap attribútumait, valamint a változó típusától függően bizonyos jellemzőket („Megtartás” attribútum, integer vagy valós formátum, üzenet karaktorsor maximális hossza).

A változó száma elé mindig egy definiált szöveget kell beírni, mivel egy változó szimbólum nem kezdődhet számjeggyel. Ha a „számjegyek száma” „Auto”-ra van állítva, az ISaGRAF a változó számát a minimálisan szükséges számjegyek számára formátálja. Ha a számjegyek száma specifikálva van, az ISaGRAF kezdeti „0” karakterek hozzáadásával minden számot a specifikált hosszúságra formátál. A változó-számokhoz történő rögzített számú számjegyek beállítása nagyon hasznos lehet a helytelen szótárrendezés elkerülésére. Az alábbiakban néhány példa látható:

Példa: Ez a gyors deklarációs beállítás:

a következő három változót fogja létrehozni:

**Var9xx    Var10xx    Var11xx**

Példa: Ez a gyors deklarációs beállítás:

<b>Számozás:</b>		
Tól:	<input type="text" value="1"/>	Hoz:
	<input type="text" value="100"/>	
Számjegye	<input type="text" value="3"/>	
<b>Szimbólum:</b>		
Név:	<input type="text" value="MyVar"/>	## <input type="text"/>

100, **MyVar001**-től **MyVar100**-ig terjedő változót fog létrehozni

#### A.10.5 Modbus SCADA címező hozzárendelési lista

Az ISaGRAF „hálózati címek” gyakran használatosak az ISaGRAF rendszer és egy Modbus kommunikáción alapuló SCADA közötti kapcsolat létrehozására. Ebben az esetben a SCADA egy Modbus gazda, az ISaGRAF cél pedig egy Modbus kiszolgálóként szerepel. A hálózati címek virtuális Modbus hozzárendelési lista létrehozására használatosak minden olyan ISaGRAF változó számára, amelyeket a SCADA-tól kell vezérelni. Az **„Eszközök / Modbus SCADA címező hozzárendelési lista”** egy hatékony eszköz egy Modbus virtuális hozzárendelési listának az alkalmazás változóival történő gyors létrehozásához.

A hozzárendelés listázó eszközök két listát mutatnak. A felső a Modbus hozzárendelési lista egy szegmense (4096 hely), amely a hozzárendelt (hálózati címmel rendelkező) változókat mutatja. Az alsó lista hozzárendelés nélküli (definált hálózati cím nélküli) változókat mutat. Egy változó hozzárendeléséhez a „0” cím nem használható.

Egy változónak az egyik listából a másikba történő átviteléhez, és ezzel egy lista felépítéséhez a **„Szerkesztés”** menü **„Hozzárendelés”** illetve **„Eltávolítás”** parancsait kell használni. Ugyanezek a műveletek az egyik listában egy változó szimbólumára való dupla kattintással is elvégezhetők, ami az adott változót a másik listára küldi. A lista másik szegmensének megtekintéséhez bármikor használni lehet a **„Szegmens”** kinyíló listát.

A címek akár decimális, akár hexadecimális módon történő megjelenítéséhez a **„Beállítások”** menü parancsai használhatók bármikor.

A **„Szerkesztés / Keresés”** parancs egy deklarált változó megkeresésére szolgál, függetlenül attól, hogy az már hozzárendelt-e vagy sem.

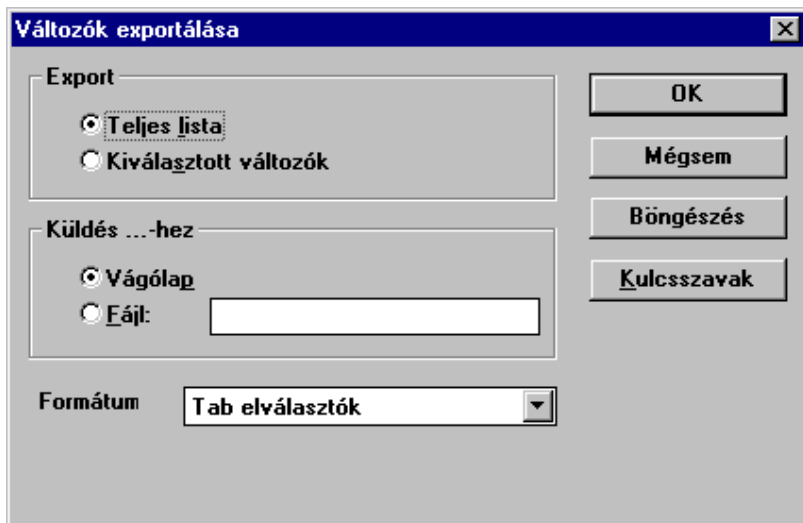
#### A.10.6 Információcsere más alkalmazásokkal

Az ISaGRAF szótárszerkesztő eszköz import/export funkciókkal rendelkezik a más alkalmazásokkal, pl. szövegszerkesztőkkel, táblázatkezelőkkel, adatbázis-rendezőkkel, stb. történő információcseréhez. Ezek a parancsok a **„Szerkesztés”** menüben vannak csoportosítva. A **„Szöveg exportálása”** parancs a szerkesztett tárgyak készletét leíró mezőkről egy tiszta ASCII szöveges leírást állít össze, és ezt a szöveget vagy a Windows vágólapra, vagy egy fájlban tárolja. Az ilyen információkat általában más alkalmazások használják fel. A **„Szöveg importálása”** parancs változó deklarálás leíró mezőket importál tisztán ASCII szöveges

formátumban, amely vagy a Windows vágólapon, vagy egy fájlban van tárolva, és a jelenleg szerkesztés alatt álló listát importált mezőkkel frissíti. Az ilyen információkat általában más alkalmazások állítják elő.

## Adatok exportálása

A következő párbeszédablak a „Szöveg exportálása” parancs futtatásakor jelenik meg. A felhasználó ennek segítségével vezérelheti az exportálási folyamatot.



A „**Teljes lista**” kiválasztása azt jelzi, hogy a teljes szerkesztett listát exportálni kell. Ebben az esetben a pillanatnyi kiválasztás figyelmen kívül lesz hagyva. A „**Kiválasztott változók**” lehetőség kiválasztása azt jelzi, hogy csak a kiemelt változók lesznek exportálva.

Ha a „**Vágólap**” beállítás ki van választva, akkor az exportált információ tiszta ASCII szöveg formájában a Windows vágólapján kerül tárolásra. A szöveg azután rendelkezésre áll más alkalmazások „Beillesztés” parancsaihoz. Ha a „**Fájl**” beállítás ki van választva, akkor az exportált információ egy ASCII fájlban kerül tárolásra. Ennek a fájlnak a teljes elérési útvonalát be kell írni. A „**Böngészés**” parancs segítségével megkereshető egy létező elérési útvonal.

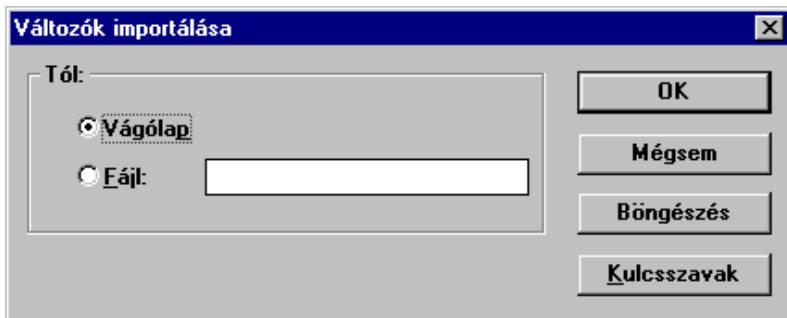
Ezután a felhasználónak ki kell választania az exportált szöveg formátumát. A rendelkezésre álló formátumok a későbbi fejezetekben kerülnek ismertetésre. Az „**OK**” gomb megnyomása végrehajtja az exportálási funkciót. A „**Mégsem**” gomb megnyomása bezárja a párbeszédablakot, és kilép az exportálási parancsból.

Az exportált szövegben a kiválasztott tárgyak valamennyi mezője el van tárolva, a standard deklarációs sorrendnek megfelelően. Az exportált szöveg első sora a mezők neveit tartalmazza. Minden tárgyat egy sornyi szöveg ír le. A „sor vége” elválasztó a standard „**od-0a**” MS-DOS szekvencia. Az első exportált sorban a mezők azonosítására szolgáló nevek a „**Kulcsszó**” gomb megnyomásával változtathatók meg. Ez a parancs későbbi fejezetekben lesz ismertetve.



## Adatok importálása

A következő párbeszédablak a „**Szöveg importálása**” parancs futtatásakor jelenik meg. A felhasználó ennek segítségével vezérelheti az importálási folyamatot.



Ha a „**Vágólap**” beállítás ki van választva, akkor az importált információ tiszta ASCII szöveg formájában a Windows vágólapjáról kerül betöltésre. Ha a „**Fájl**” beállítás ki van választva, akkor az importált szöveg egy ASCII fájlban kerül beolvasásra. Ennek a fájlnak a teljes elérési útvonalát be kell írni. A „**Böngészés**” parancs segítségével megkereshető egy létező elérési útvonal.

Az importálás funkció automatikusan felismeri az importált szövegben használt formátumot (elválasztókat). A rendelkezésre álló formátumok a későbbi fejezetekben kerülnek ismertetésre. Az „**OK**” gomb megnyomása végrehajtja az importálási funkciót. A „**Mégsem**” gomb megnyomása bezárja a párbeszédablakot, és kilép az importálási parancsból. Az első importált sorban a mezők azonosítására szolgáló nevek a „**Kulcsszó**” gomb megnyomásával változtathatók meg. Ez a parancs későbbi fejezetekben lesz ismertetve.

Az exportált szöveg első sorának a mezők neveit kell tartalmaznia, a következő sorokban alkalmazott sorrendnek megfelelően. Minden tárgyat egy sornyi szövegnek kell leírnia. A „sor vége” elválasztó a standard „**Od-Oa**” MS-DOS szekvencia. A mezők bármilyen sorrendben lehetnek. Ha vannak hiányzó mezők, akkor azok az importált tárgy leírásban automatikusan alapértelmezett értékekkel lesznek kitöltve. Ha egy importált tárgy már létezik a szerkesztett listában, akkor a felhasználónak meg kell erősítenie, hogy azt át akarja írni. Ekkor a tárgy leírása az importált mezőkkel frissítődik. Ha vannak hiányzó mezők, akkor azok nem kerülnek frissítésre az importált tárgy leírásban.



## Rendelkezésre álló szövegformátumok

Az alábbiakban az exportálási parancshoz rendelkezésre álló formátumok vannak felsorolva. Az importálási parancs automatikusan felismeri ezeket a formátumokat.

- tab elválasztók

**Leírás:** A mezők tab karakterekkel vannak elválasztva egymástól.

<b>Példa:</b>	Név szint	Attribútum belső	Megjegyzés belső számított vízszint
---------------	--------------	---------------------	--

alm1      output      fő alarm output

- vessző elválasztók

**Leírás:** A mezők vesszőkkel vannak elválasztva egymástól.

**Példa:** Név,Attribútum,Megjegyzés  
szint,belső,belső számított vízszint  
alm1,output,fő alarm output

- pontosvessző elválasztók

**Leírás:** A mezők pontosvesszőkkel vannak elválasztva egymástól.

**Példa:** Név;Attribútum;Megjegyzés  
szint;belső;belső számított vízszint  
alm1;output;fő alarm output

- vesszők és idézőjelek

**Leírás:** A mezők vesszőkkel vannak elválasztva egymástól.  
Minden mező idézőjelek közé van írva.

**Példa:** „Név” „Attribútum” „Megjegyzés”  
„szint” „belső” „belső számított vízszint”  
„alm1” „output” „fő alarm output”

## Kulcsszavak

Az első importált vagy exportált sorban a mezők azonosítására szolgáló nevek a „**Kulcsszó**” gomb megnyomásával változtathatók meg. Ez a parancs a következő párbeszédablakot nyitja meg:



Az ablak a tárgymezők listáját valamint a hozzájuk tartozó kulcsszavakat mutatja. Egy kulcsszó módosításához a felhasználónak ki kell választania egy mezőt a listából, és meg kell nyomnia a „**Módosítás**” gombot. A „**Kiindulási**” gomb megnyomása visszaállítja az eredeti kulcsszó-listát. A kulcsszavak elnevezéseinek a következő szabályoknak kell eleget tenniük:

- a név nem haladhatja meg a **16** karaktert

- az első karakternek **betűnek** kell lennie
- a többi karakter lehet **betű**, **számjegy**, vagy aláhúzás („\_”) karakter
- ugyanaz a név nem használható különböző kulcsszavakhoz

Az alábbiakban az ISaGRAF-ban található standard kulcsszavak láthatók:

Tárgy neve .....	<b>Név</b>
Szöveges megjegyzés .....	<b>Megjegyzés</b>
Hálózati cím .....	<b>Cím</b>
Attribútumok (belső, input, output) .....	<b>Attribútum</b>
Logikai „Hamis” karaktorsor .....	<b>Hamis</b>
Logikai „Igaz” karaktorsor .....	<b>Igaz</b>
Analóg formátum (valós vagy integer) .....	<b>Formátum</b>
Analóg egység karaktorsor .....	<b>Egység</b>
Analóg konverzió név .....	<b>Konverzió</b>
Üzenet maximális hossza .....	<b>MaxHossz</b>
Funkcióblokk könyvtár típus .....	<b>Könyvtár</b>
Definiált szó előfordulás .....	<b>Ekvivalencia</b>
Belső attribútum .....	<b>Belső</b>
Input attribútum .....	<b>Input</b>
Output attribútum .....	<b>Output</b>
Konstans attribútum .....	<b>Konstans</b>
Valós analóg formátum .....	<b>Valós</b>
Integer analóg formátum .....	<b>Integer</b>

## A.11 Az I/O csatlakozásszerkesztő használata








Az I/O csatlakozási művelet célja egy logikai kapcsolat létrehozása az alkalmazás I/O változói és a célgépen létező kártyák fizikai csatornái között. Ennek a kapcsolatnak a létrehozásához a felhasználónak azonosítania és kell, és be kell állítania a célgépen levő összes kártyát, és a megfelelő I/O csatornákra I/O változókat kell helyeznie.

A bal oldali lista a célgép **kártyahelyekkel** rendelkező keretét mutatja. Egy kártyahely vagy szabad, vagy azt egy I/O kártya illetve egy komplex berendezés használja. Minden kártyahelyet egy **sorszám** azonosít be. A keret maximum **255** kártyát tartalmazhat. A jobb oldalon levő lista a kártya paramétereit és a kiválasztott kártyához csatlakoztatott változókat mutatja. Egy kártya maximum **128** I/O csatornával rendelkezhet. Az egyedi I/O kártyák összes száma (beleértve az egyedi berendezéseket és a komplex berendezések kártyáit is) nem haladhatja meg a **255**-öt.






### Ikonok




Az előlapon látható ikonok a kártya csatornáikhoz csatlakoztatható változók típusát és attribútumait mutatják. Az ISaGRAF rendszer nem engedi meg különböző típusú változók csatlakoztatását ugyanahhoz a kártyához. Az alkalmazott ikonok jelentései a következők:

-  .....logikai típus
-  .....integer/valós típus (mindkét típusú változó csatlakoztatható)
-  .....üzenet típus
-  .....inputok – nincs csatlakoztatva csatorna
-  .....outputok – nincs csatlakoztatva csatorna
-  .....inputok – legalább egy csatorna van csatlakoztatva
-  .....outputok – legalább egy csatorna van csatlakoztatva

Az alábbiakban a kártyahelybe illesztett I/O eszköz típusát mutató ikonok vannak felsorolva:

-  .....komplex I/O berendezés
-  .....valódi I/O kártya
-  .....virtuális I/O kártya

Az alábbiakban egy paraméter vagy egy csatorna megrajzolására használt ikonok vannak felsorolva:

-  .....kártya paraméter
-  .....szabad csatorna
-  .....csatlakoztatott csatorna



## A kártyák áthelyezése a listában

A kiválasztott I/O kártyának a fő listában eggyel feljebb vagy lejjebb történő áthelyezéséhez ezeket az eszközsor gombokat, vagy a „**Szerkesztés / Kártya áthelyezése lefelé/felfelé**” menüparancsokat kell használni. A „**Szerkesztés / Kártyahely beszúrása**” parancs egy üres kártyahelyet illeszt be a kiválasztott pozícióhoz.

## A.11.1 I/O kártyák definiálása

A „**Szerkesztés**” menü a kiválasztott kártya definiálásához (paramétereinek beállításához), valamint az I/O változóknak csatornáikhoz történő csatlakoztatásához tartalmaz alapvető parancsokat.



### I/O kártya típusának kiválasztása

Az I/O változóknak egy kártyához történő csatlakoztatása előtt be kell írni a kártya azonosítását. Az ISaGRAF workbenchen rendelkezésre áll egy előre definiált kártyákból álló könyvtár. Ezt a könyvtárat valamelyik I/O eszköz beszállítója állíthatta össze. A kártya azonosításának a beállítására a „**Szerkesztés / Kártya/Berendezés beállítás**” parancs szolgál. Ez a parancs vagy egyetlen kártya, vagy komplex I/O berendezések ISaGRAF könyvtárból történő kiválasztására használható. A megfelelő kártya illetve berendezés beállítás egy kártyahelyre való kettős kattintással is elvégezhető.

Egy kártya valamennyi csatornája ugyanolyan típusú (logikai, integer/valós, vagy üzenet) és irányú (input vagy output). Az I/O csatlakozás során a valós és integer változók nincsenek megkülönböztetve. Egy komplex I/O berendezés egy különböző típusú vagy irányú csatornákkal rendelkező I/O eszközt jelent. Egy komplex I/O berendezést egyedi I/O kártyák listája képviseli. Ez a keretlistában csak egyetlen kártyahelyet foglal le.



### Egy kártya eltávolítása

A „**Szerkesztés / Kártyahely kiürítése**” parancs a jelenleg kiválasztott kártya vagy I/O berendezés eltávolítására szolgál. Ha a megfelelő csatornákhöz már vannak csatlakoztatva változók, akkor azok a kártyahely kiürítésekor automatikusan leválasztódnak.



### Valódi kártyák és virtuális kártyák

A „**Szerkesztés / Valódi/Virtuális Kártya**” parancs beállítja a jelenleg kiválasztott kártya vagy I/O berendezés érvényességét. Egy kártya érvényességének jelzésére a keretlistában a következő ikonok jelennek meg:



.....valódi I/O kártya



.....virtuális I/O kártya

**Valódi módban** az I/O változók közvetlenül kapcsolódnak a megfelelő I/O eszközökhöz. Az alkalmazás programban az input vagy output műveletek közvetlenül a megfelelő tényleges helyszíni I/O eszközök input vagy output állapotához kötődnek. **Virtuális módban** az I/O változók kimondottan belső változókként kerülnek feldolgozásra. Ezeket a hibakereső képes olvasni vagy

frissíteni, így a felhasználó szimulálni tudja az I/O feldolgozást, de valódi csatlakozás nincs létrehozva.



### Műszaki megjegyzések

Az „**Eszközök / Műszaki megjegyzés**” parancs megjeleníti a kiválasztott kártya vagy komplex berendezés online használati utasítását. A kártya műszaki megjegyzését az I/O kártya hardverbeszállítója készítette el. Ez az I/O kártya kezelésével kapcsolatos információkat tartalmaz. Ugyancsak tartalmazza paramétereinek a jelentését is.



### Csatlakoztatott változók eltávolítása

Az „**Eszközök / Kártyacsatornák felszabadítása**” parancs leválasztja a kiválasztott kártyán már csatlakoztatott valamennyi I/O változót.



### Megjegyzések definiálása szabad csatornához

Az „**Eszközök / Kártyacsatornák felszabadítása**” parancs leválasztja a kiválasztott kártyán már csatlakoztatott valamennyi I/O változót.

## A.11.2 A kártyák paramétereinek beállítása

Egy kártyaparaméter értékének beállításához a felhasználónak duplán rá kell kattintania a paraméter jobb oldali listában levő nevére. A másik lehetőség annak kiválasztása (kivilágítása), és a „**Szerkesztés**” menüről a „**Csatorna/paraméter beállítása**” parancs kiadása. A paraméterek a lista elején vannak felsorolva. Ezeket a listában a következő ikon képviseli:



.....kártya paraméter

A paraméter jelentését és beviteli formáját a megfelelő I/O kártya illetve berendezés beszállítója tervezte meg. A kártyaparaméterekkel kapcsolatos további információkért alkalmazza az „**Eszközök / Műszaki megjegyzés**” parancsot, vagy olvassa el a hardver kezelési utasítását.

## A.11.3 I/O csatornák csatlakoztatása

Egy csatorna csatlakozásának beállításához a felhasználónak duplán rá kell kattintania annak helyére a jobb oldali listában. A másik lehetőség annak kiválasztása (kivilágítása), és a „**Szerkesztés**” menüről a „**Csatorna/paraméter beállítása**” parancs kiadása. A csatornákat a következő ikonok képviselik a listában:



.....szabad csatorna



.....csatlakoztatott csatorna

A lista mindazokat a változókat tartalmazza, amelyek megfelelnek a kiválasztott kártyatípusnak és iránynak. Itt csak a még nem csatlakoztatott változók vannak felsorolva. A „**Csatlakoztatás**” gomb a kiválasztott csatornához csatlakoztatja a listából kijelölt változót. A „**Felszabadítás**” gomb eltávolítja (leválasztja) a változót a

kiválasztott csatornától. A „**Következő**” és az „**Előző**” gombok a kártya másik csatornájának a kiválasztására szolgálnak. A kiválasztott csatorna helye mindig meg van jelenítve a párbeszédablak fejlécében.

#### A.11.4 Közvetlenül képviselt változók

A szabad csatornák azok, amelyek nincsenek egy deklarált I/O változóhoz kapcsolva. Az ISaGRAF lehetővé teszi a **közvetlenül képviselt változók** használatát a programok forrásaiban, egy szabad csatorna képviselésére. Egy közvetlen képviselésű változó azonosítója mindig a „%” karakterrel kezdődik.

Az alábbiakban egy egyedi kártya egyik csatornájának közvetlen képviselői változójának elnevezési konvenciói szerepelnek. Az „**s**” a kártya kártyahelyének száma. A „**c**” a csatorna száma.

%IXs.c .....egy logikai input kártya szabad csatornája  
 %IDs.c .....egy integer input kártya szabad csatornája  
 %ISs.c .....egy üzenet input kártya szabad csatornája  
 %QXs.c .....egy logikai output kártya szabad csatornája  
 %QDs.c .....egy integer output kártya szabad csatornája  
 %QSs.c .....egy üzenet output kártya szabad csatornája

Az alábbiakban egy komplex berendezés egyik csatornája közvetlen képviselői változójának elnevezési konvenciói szerepelnek. Az „**s**” a berendezés kártyahelyének száma. A „**b**” a komplex berendezésen belüli egyedi kártya indexe. A „**c**” a csatorna száma.

%IXs.b.c .....egy logikai input kártya szabad csatornája  
 %IDs.b.c .....egy integer input kártya szabad csatornája  
 %ISs.b.c .....egy üzenet input kártya szabad csatornája  
 %QXs.b.c .....egy logikai output kártya szabad csatornája  
 %QDs.b.c .....egy integer output kártya szabad csatornája  
 %QSs.b.c .....egy üzenet output kártya szabad csatornája

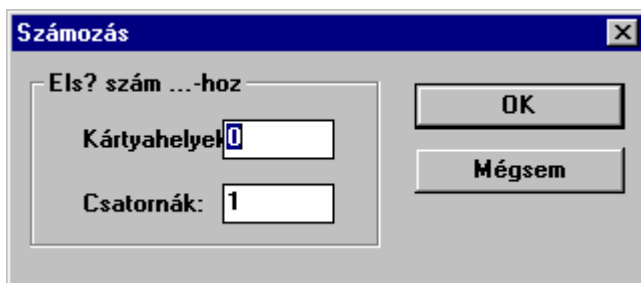
Az alábbiakban példák láthatók:

%QX1.6      Az 1. számú kártya 6. csatornája (logikai output)  
 %ID2.1.7    A 2. számú berendezésben levő 1. számú kártya 7. csatornája  
 (integer      input)

Egy közvetlen képviselői változónak nem lehet „**valós**” adattípusa.

#### A.11.5 Számozás

A számozási konvenciók beállítására a „**Beállítási lehetőségek / Számozás**” parancsot kell használni. A következő párbeszédablakban az első kártyahelyhez használt számot és az egyes kártyák első csatornájához használt számot lehet meghatározni:



Kiindulási értéként a kártyahelyek számozása a „0”, a csatornák számozása pedig az „1” indexnél kezdődik.

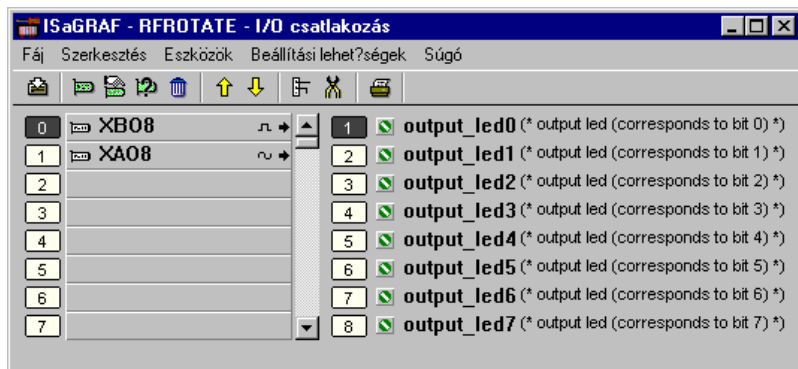
**Figyelmeztetés:** A számozási konvenciók megváltoztatásánál körültekintően kell eljárni, mivel az hatással van a közvetlenül képviselt változókhoz használt szimbólumokra, és fordítási hibákhoz vezethet, ha a meglevő programokban közvetlenül képviselt I/O változók vannak használva.

#### A.11.6 Egyedi védelmek beállítása

Az ISaGRAF workbench egy hierarchikus jelszókra épülő teljes adatvédelmi rendszert biztosít. Az I/O csatlakozás globálisan egy jelszóval levédhető. Ezen felül, az ISaGRAF lehetővé teszi egyedi védelem beállítását bármely I/O csatornához. Ez az alábbiakat feltételezi:

- a jelszók már meg lettek határozva a jelszó-definiáló rendszerben (a Projektrendező ablak „Projekt / Jelszó beállítása” parancsának használatával), így az egyedi védelemhez védelmi szintek állnak rendelkezésre.
- az egyedi védelemhez magasabb prioritású védelmi szinteket használnak, mint a globális I/O védelem.

Ha egy I/O csatorna egyedi védelemmel rendelkezik, akkor annak neve mellett egy kis ikon jelenik meg az I/O csatlakozási ablakban.



A kiválasztott csatorna egyedi védelmének beállításához vagy eltávolításához a **„Szerkesztés”** menü **„Védelem beállítása”** illetve **„Védelem eltávolítása”** parancsait kell használni. Mindkét parancs egy érvényes jelszó beírását kéri ahhoz, hogy a csatornához egy védelmi szintet lehessen csatolni. Ezután pedig valahányszor meg akar változtatni egy egyedi védelemmel rendelkező csatorna csatlakozását, be kell írnia egy megfelelő prioritású szintnek megfelelő jelszót.

Figyelmeztetés: Ha egy csatorna egy szinttel le van védve és a megfelelő jelszót eltávolítják a védelmi rendszerből, valamint amennyiben nincs definiálva magasabb szintű jelszó, egy új, megfelelő szintű jelszó definiálása nélkül a csatornát többé nem lehet megváltoztatni.

## A.12 Konverziós táblázatok létrehozása

Az ISaGRAF workbench lehetővé teszi a felhasználó számára konverziós táblázatok létrehozását. Egy konverziós táblázat egy analóg konverzió definiálásához használt pontok készletét jelenti. Egy konverziós táblázat egy analóg input vagy output változóhoz csatolható. Egy táblázat arányos összefüggést hoz létre elektromos értékek (olvasó vagy input érzékelő, vagy az output eszközhöz küldött értékek) és fizikai értékek (alkalmazás programozásban használt értékek) között.

A konverziós táblázatok szerkesztése az ISaGRAF szótáráblakban levő **„Eszközők / Konverzió”** parancs párbeszédablakával történik

Egy definiált konverziós táblázat a kiválasztott projekt bármelyik input vagy output analóg változó értékének a szűrésére használható. Egy konverziós táblázatnak egy változóhoz való csatlakoztatása az ISaGRAF szótár parancsaival, a változó deklaráló szerkesztővel történik. Ezután ki kell választani egy input vagy output analóg változót, és szerkeszteni kell annak paramétereit. Egy változó nem csatlakoztatható egy még nem definiált konverziós táblázathoz.

### A.12.1 Fő parancsok

A **„Konverziós táblázatok”** párbeszédablak a definiált konverziós táblázatok listáját mutatja, és egy létező táblázat szerkesztésére (pontjainak definiálásához), egy új táblázat létrehozására, valamint egy táblázat átnevezésére vagy törlésére szolgáló fő parancsokhoz szolgáló nyomógombokat tartalmaz. A **„Konverziós táblázatok”** párbeszédablakból történő kilépéshez nyomja meg az OK gombot, és mentse ki lemezre a konverziós táblázatokat.



#### **Egy új táblázat létrehozása**

A felhasználó az **„Új”** parancs segítségével hozhat létre egy új konverziós táblázatot. Minden projekthez maximum **127** konverziós táblázat hozható létre. Az alkalmazás végrehajtandó kódjába csak a használt (azaz analóg változókhoz csatolt) táblázatok kerülnek beillesztésre. A táblázatot a következő szabályoknak eleget téve kell elnevezni:

- a név nem haladhatja meg a **16** karaktert
- az első karakternek **betűnek** kell lennie
- a többi karakter lehet **betű**, **számjegy**, vagy aláhúzás („\_”) karakter
- a táblázat-név nem érzékeny a kisbetű/nagybetű beállításra



#### **Egy táblázat tartalmának módosítása**

A listáról kiválasztott táblázat pontjainak bevitelére a **„Szerkesztés”** parancs szolgál. Ez elvégezhető a táblázat nevére történő dupla kattintással is. Egy új konverziós táblázat létrehozásakor automatikusan előhívódik a **„Szerkesztés”** parancs. Minden táblázathoz legalább két pontot be kell vinni.

### A.12.2 Egy táblázat pontjainak bevitele

A felhasználó a „**Szerkesztés**” párbeszédablak segítségével definiálhatja egy konverziós táblázat pontjait. A párbeszédablak a bal oldalon a már definiált pontokat mutatja. A jobb alsó ablak a definiált táblázatot egy grafikus görbeként ábrázolja. A pontok bevitele az ablak parancsaival történik. A felhasználónak a pontok definiálásakor be kell tartania az ennek a fejezetnek a végén ismertetett szabályokat. A bal oldali ablak mindig a jelenleg szerkesztés alatt álló táblázat létező pontjait mutatja. A bal oldali oszlop a pontok elektromos (külső) értékét mutatja. A jobb oldali oszlop a fizikai (belső) értékeket mutatja. A felhasználónak a listáról ki kell választania a pontot ahhoz hogy annak értékét módosítsa, illetve hogy azt törölje. A lista utolsó választási lehetősége („... ..”) egy új pont definiálására szolgál. A jobb alsó ablak a jelenleg szerkesztés alatt álló táblázatot egy grafikus görbeként ábrázolja. Mivel ez a görbét részarányosan mutatja, ezért a tengelyek illetve koordináták nincsenek megjelenítve. Ezzel az ábrázolási formával gyorsan ellenőrizhető, hogy a görbe megfelelően definiálva van-e.

#### ⇒ Egy új pont definiálása

Egy új pont definiálásakor a lista utolsó választási lehetőségét („... ..”) kell kiválasztani. Ez egyébként egy új konverziós táblázat definiálásának kezdetekor a kiindulási mód. A felhasználónak be kell írnia minden pont elektromos (külső) és fizikai (belső) értékeit. Az értékek egyszerű pontosságú lebegőpontos számokként kerülnek tárolásra. Ne feledje, hogy egy görbe definiálásához legalább **két pontot** be kell vinni. Miután mindkét érték be lett írva, a „**Tárolás**” gomb megnyomása a pontot a táblázathoz adja. Mindegyik konverziós táblázathoz maximum **32** pont definiálható.

#### ⇒ Egy pont módosítása

Egy létező pont értékeinek módosításához azt először ki kell választani a listáról. Ezt követően bevethető a pont új elektromos (külső) és fizikai (belső) értékei. Az értékek egyszerű pontosságú lebegőpontos számokként kerülnek tárolásra. Miután mindkét érték be lett írva, a „**Tárolás**” gomb megnyomása a pontot frissíti a táblázatban.

#### ⇒ Egy pont törlése

Egy létező pont törlése annak a listáról való kiválasztásával, majd a „**Törlés**” gomb megnyomásával történik. Ne feledje, hogy egy görbe definiálásához legalább **két pontot** be kell vinni.

### A.12.3 Szabályok és korlátozások

Egy konverziós táblázat definiálásánál az alábbi szabályokat kell betartani. Az táblázatot mind input, mind output analóg változók konvertálására alkalmazható:

- Két pont nem definiálható ugyanazzal az elektromos értékkel
- A görbének folyamatosan növekednie vagy csökkennie kell
- Két pont nem definiálható ugyanazzal a fizikai értékkel

Egy projekt konverziós táblázatainak definiálásakor a következő korlátozások léteznek:

- Egy projektben legfeljebb **127** konverziós táblázat definiálható
- Egy konverziós táblázathoz legfeljebb **32** pont definiálható.

## A.13 A kódgenerátor használata

A kódgenerátor ablak a többi ISaGRAF Workbench ablak „Megerősítés” és „Létrehozás” parancsaira automatikusan kinyílik. A kódgenerátor ablak nem záródik be automatikusan amikor a kért kódgenerálási művelet befejeződött, hogy a felhasználó az ablak menüjéből továbbra is hozzáférhessen az összes kódgenerálási parancshoz és beállítási lehetőséghez.

### A.13.1 Fő parancsok

A „**Fájlok**” menü a program szintaxis ellenőrzéshez és kódgeneráláshoz szükséges parancsokat tartalmazza.

#### **Forráskód létrehozása**

A „**Létrehozás**” parancs a projekt egész kódját létrehozza. Ez a parancs ellenőrzi a deklarációk és programok szintaxisát, mielőtt bármilyen generálásra kerülne. A kódgenerálás közben érzékelve van minden hiba, ami az egyszeres programfordítás során nem volt detektálható. Ez vonatkozik a konverziós táblázatokra, I/O változó csatlakozásokra, és a könyvtárakkal való kapcsolatokra. A kódgenerálás leállítja egy program fordítását, ha hibák lettek érzékelve. Ezt a programot a kódgenerálás folytatása előtt ki kell javítani. A már ellenőrzött és legutóbbi „**Megerősítés**” műveleteik óta nem módosított (érezelt hibák nélküli) programok nem lesznek újrafordítva. A változó deklarálás megerősítés valamint az alkalmazás koherencia ellenőrzése mindig fel van dolgozva. A program ellenőrzése közben a „**Létrehozás**” művelet az ESCAPE billentyű lenyomásával szakítható félbe.

Megjegyzés: Ha egy program helyi változójának deklarációja módosítva lett, akkor az a program meg van erősítve. Ha egy globális változó lett módosítva, akkor az összes program meg van erősítve.

#### **Program szintaxis ellenőrzés**

A felhasználó a „**Program megerősítése**” parancs segítségével egyetlen program megerősítését végezheti el. A kiválasztott program fordításra kerül, még akkor is, ha a legutóbbi megerősítése óta nem is lett módosítva. A felhasználó a „**Szótár megerősítése**” parancs segítségével a projekt összes változó deklarálásának megerősítését végezheti el.

Az „**Összes program megerősítése**” a projekt összes programjának szintaxisát ellenőrzi, még akkor is, ha azok közül némelyike nem lett módosítva. Ez a parancs nem áll le, ha a programban hiba lett érzékelve. Ez a projekt programjaiban megmaradt összes hiba teljes felsorolására használható. Ez a parancs az **ESCAPE** billentyű megnyomásával szakítható félbe.

#### **Egy módosítás szimulációja**

Az „**Érintés**” parancs a projekt összes programjának módosítását szimulálja, hogy a következő „**Létrehozás**” művelet során azok mindegyike meg legyen erősítve. A „**Megnyitás**” parancs az utoljára megerősített program megnyitására szolgál. Ez a

parancs egy program közvetlen eléréséhez nagyon hasznos olyankor, amikor hibák lettek érzékelve.

### A.13.2 Fordítási beállítások

A „**Fordítási beállítások**” parancs az ISaGRAF kódgenerátor által a cél kód felépítésére és optimalizálására használt fő paraméterek beállítására szolgál. Ennek a parancsnak a célja a generálandó kód típusának kiválasztása a megfelelő ISaGRAF céloknak megfelelően, és az optimalizáló paraméterek beállítása a várt fordítási időnek és az alkalmazás futtatási igényeinek megfelelően.

A „**Feltöltés**” gomb egy második párbeszédablakot nyit meg újabb beállítási lehetőségekkel, amelyek lehetővé teszik a tömörített forráskód beépítését a letöltött kódba, a „Feltöltés” lehetőség bekapcsolása érdekében. További magyarázatok a „Feltöltés” dokumentációban találhatók.



#### Célok kiválasztása

A felső lista az előállítható rendelkezésre álló célkódok listáját tartalmazza. A „>>” jel a kiválasztott cél(oka)t jelzi. Az ISaGRAF Kódgenerátor maximum **3** különböző kódot tud előállítani ugyanabban a fordítási műveletben. A „**Kiválasztás**” és „**Kiválasztás vissza**” parancsok a szükséges célkódok listájának beállítására szolgálnak, a cél hardvernek megfelelően. Az alábbiakban a standard ISaGRAF célok vannak felsorolva:

- SZIMULÁCIÓ:** .....Ez a kód az ISaGRAF Szimulátorhoz szolgál a Workbench-en. A szimulátor nem futtatható, ha ez a cél nincs kiválasztva az alkalmazás kód előállítására.
- ISA86M:** .....Ez egy TIC kód (Célfüggetlen kód; „Target Independent Code”), amely az Intel alapú processzorokon telepített ISaGRAF kernelekhez tartozik. A processzor típusa csak a generált kódban levő bájt sorrend szempontjából lényeges.
- ISA68M:** .....Ez egy TIC kód (Célfüggetlen kód), amely a Motorola alapú processzorokon telepített ISaGRAF kernelekhez tartozik. A processzor típusa csak a generált kódban levő bájt sorrend szempontjából lényeges.
- SCC:** .....Ennek a célnak a kiválasztására az ISaGRAF programfordító strukturált „C” nyelvű forráskódot állít elő és kapcsol az ISaGRAF cél kernel könyvtárakhoz, egy beépített végrehajtható kód előállítására.
- CC86M:** .....Ennek a célnak a kiválasztására az ISaGRAF programfordító nem strukturált „C” nyelvű forráskódot állít elő és kapcsol az ISaGRAF cél kernel könyvtárakhoz, egy beépített végrehajtható kód előállítására. Ez a kiválasztás a V3.23-as verzió előtti ISaGRAF verziók illeszthetősége érdekében van biztosítva, amelyekben a strukturált „C” kódgenerálás és integrálás még nem volt támogatva.

Az Ön PLC-jére telepített ISaGRAF cél kernel típusa a hardver kézikönyvből tudható meg. Az ISaGRAF Workbench későbbi kiadásai támogathatnak majd egyéb cél típusokat (gépi kód, C forráskód, stb.) is.



## SFC feldolgozás

Az ISaGRAF SFC gép használatának bekapcsolásához jelölje be a **„Beépített SFC gép használata”** lehetőséget. **Ezt a módot kell előnyben létesíteni, mivel ez jobb futtatási teljesítményekhez vezet.** Bizonyos ISaGRAF cél kivitelezéseken, vagy még gyakrabban, ISaGRAF kód utófeldolgozáson alapuló testre szabott célokon azonban a cél gép hiányozhat. Ebben az esetben szükség lehet ennek a beállításnak az eltávolítására, és az ISaGRAF programfordítóra hagyni az SFC diagramok alsó szintű utasításokra történő fordítását. Ennek a beállításnak a használatával kapcsolatos bővebb információk a hardver dokumentációban találhatóak.



## Optimalizáló beállítások

Az alábbiakban az ISaGRAF Kódgenerátor által a célkód optimalizálására használt, a **„Programfordító beállítások”** párbeszédablakban kiválasztható paraméterek láthatók. Az **„Alapértelmezett”** gomb az összes beállítási lehetőség eltávolítására szolgál, a fordítási idő csökkentése érdekében.



Amikor be van állítva a **„Két optimalizálási lefutás”**, akkor az ISaGRAF Kód optimalizáló kétszer van lefuttatva. A második lefutás során elvégzett optimalizálások általában kevésbé jelentősek, mint az első lefutásban elvégzettek.



Amikor a **„Konstans kifejezések kiértékelése”** beállítás ki van választva, a programfordító kiértékeli a konstans kifejezéseket. A **„2 + 3”** numerikus kifejezést például **„5”** váltja fel a célkódban. Amikor ez a beállítás nincs kiválasztva, a konstans kifejezések a futtatáskor vannak kiszámítva.



Amikor a **„Használaton kívüli címkék elrejtése”** beállítás ki van választva, az Optimalizáló a használaton kívüli ugrások és címkék elrejtésével egyszerűsíti a rendszer ugrásait és címkéit.



Amikor a **„Változó másolás optimalizálása”** lehetőség ki van választva, az ideiglenes változók használata (amelyek közbeni eredmények tárolására vannak használva) optimalizálva van. Ez a beállítás általában a **„Kifejezések optimalizálása”** beállítással együtt van használva. Amikor ez a beállítás ki van választva, az Optimalizáló újra felhasználja a programban egynél többször használt kifejezések és alkifejezések eredményeit.



Amikor be van állítva a **„Használaton kívüli kód elrejtése”** lehetőség, akkor az Optimalizáló elrejt a jelentéktelen kódot. Ha pl. a következő utasítások vannak programozva: **„var := 1; var := X;”,** akkor a megfelelő generált kód mindössze a következő: **„var := X;”.**



Amikor be van állítva az **„Aritmetikai művelet optimalizálása”** lehetőség, akkor az Optimalizáló speciális operandusoknak megfelelően egyszerűsíti az aritmetikai műveleteket. Pl. az **„A + 0”** kifejezést például **„A”** váltja fel. Amikor be van állítva a **„Logikai műveletek optimalizálása”** lehetőség, akkor az Optimalizáló speciális

operandusoknak megfelelően egyszerűsíti a logikai műveleteket. Pl. az „**A & A**” kifejezést például „**A**” váltja fel.



Amikor be van állítva a „**Bináris döntési diagramok felépítése**” beállítás, akkor az Optimalizáló a logikai egyenleteket (**ÉS**, **VAGY**, **XVAGY** valamint **NEM** operátorokat keverve), egy csökkentett feltételes ugrási operátor-listával váltja fel. A fordítás csak akkor kerül operációra, ha az ugrási szekvencia várható végrehajtási ideje kevesebb, mint ami az eredeti kifejezéshez van elvárva.

A következő táblázat az egyes paramétereknek megfelelő várt optimalizálási és szükséges fordítási időket foglalja össze:

	nyereség (teljesítmények) .....	fordítási idő
2 lefutás futtatása	<b>xxxx</b> .....	(*)
Konstans kifejezés optimalizálás	<b>xxxxxxxx</b> .....	<b>xxxx</b>
Használton kívüli címkék rejtése	<b>xxxx</b> .....	<b>xxxxxxxx</b>
Változó másolás optimalizálása	<b>xxxx</b> .....	<b>xxxxxxxx</b>
Kifejezések optimalizálása	<b>xxxx</b> .....	<b>xxxxxxxx</b>
Használton kívüli kód rejtése	<b>xxxx</b> .....	<b>xxxxxxxx</b>
Aritmetikai művelet optimalizálás	<b>xxxxxxxx</b> .....	<b>xxxx</b>
Logikai műveletek optimalizálása	<b>xxxxxxxx</b> .....	<b>xxxx</b>
Bináris döntés-diagramok építése	<b>xxxxxxxxxxxx</b> .....	<b>xxxxxxxxxxxx</b>

(\*) a fordítási idő szintén meg van szorozva 2-vel.

### A.13.3 C forráskód készítése

Az ISaGRAF workbench lehetővé teszi forráskód „C” nyelvben történő elkészítését. Ebben az esetben az alkalmazás teljes tartalma, beleértve az SFC diagram leírást, adatbázis definíciót, és kódszekvenciákat is, „C” forrás formátumban van generálva. Két lehetőség van, amely két generált kód stílusnak feleltethető meg:

**CC86M** .....(C forráskód - V3.04) nem strukturált "C" forráskódot állít elő. Ezt a stílust akkor kell kiválasztani, ha a cél szoftver a 3.23-as előtti kiadású ISaGRAF kiadáson alapul.

**SCC** .....(strukturált C forráskód) egy strukturált "C" forráskódot állít elő. Ezt a stílust akkor kell előnyben részesíteni, ha a cél szoftver a 3.23-as, vagy azt követő kiadású ISaGRAF kiadáson alapul.

A projekt alkönyvtárban a következő két fájl kerül létrehozásra:

**APPLI.C**.....az alkalmazás közös forráskódja

**APPLI.H**.....közös „C” nyelvű definíciók

A strukturált „C” forráskód generálás esetében az alkalmazás mindegyik programjához egy „**C**” forrásfájl és egy „**H**” definíció fájl van létrehozva a közös „**APPLI.C**” és „**APPLI.H**” fájlokra felül. Ezeket a fájlokat a végső végrehajtható kód előállítás érdekében le kell fordítani és az ISaGRAF célkönyvtárakhoz kell őket kapcsolni. Az ajánlott kivitelezési eljárásokkal kapcsolatban az „ISaGRAF I/O fejlesztői eszközkészlet Kezelési útmutatója” nyújt tájékoztatást.

Megjegyzés: Az ISaGRAF alkalmazás „C” fordítása után egyes hibakeresési lehetőségek, pl. az alkalmazás letöltés, online módosítás, és töréspontok nem állnak többé rendelkezés.

### A.13.4 Információk megtekintése

A **„Szerkesztés”** menü a kódgenerálás vagy szintaxisellenőrző műveletek során felépített különböző szövegfájlok kódgenerátor ablakban történő megtekintéséhez szükséges parancsokat tartalmazza. A kódgenerátor ablak egy szöveges terület, amely a kódgenerálási vagy szintaxisellenőrzési műveletek közbeni üzeneteket tartalmazza. Minden információ el van tárolva a lemezen, hogy azokat a **„Szerkesztés”** menü parancsaival meg lehessen vizsgálni.

#### ☐ **Szerkesztő parancsok**

A **„Képernyő törlése”** parancs az ablak szövegterületének törlésére szolgál. Az ablak minden kódgenerálási vagy szintaxisellenőrzési művelet előtt automatikusan törlődik. A **„Másolás”** parancs a megjelenített szövegnek a Windows Vágólapjára történő másolására szolgál, hogy azt más alkalmazásokkal, pl. az ISaGRAF szövegszerkesztőkkel is használni lehessen.

#### ☐ **Programfordító output üzenetek megtekintése**

A **„Végrehajtási üzenetek”** parancs a legutóbbi **„Létrehozás”** vagy **„Megerősítés”** művelet során megjelenített összes üzenetet kijelzi az ablak szöveges területén. Ez az összes hibaüzenetre is vonatkozik.

A **„Szerkesztés”** menü egyéb választási lehetőségeivel a felhasználó figyelemmel kísérheti a szintaxis megerősítés és kódgenerálás közben létrehozott kiegészítő szövegfájlokat. Ezek a fájlok általában nincsenek felhasználva egy közös ISaGRAF projekthez.

### A.13.5 Erőforrások definiálása

A felhasználó a **„Beállítások”** menü **„Erőforrások”** parancsának segítségével erőforrásokat definiálhat. Az erőforrás bármely, a felhasználó által definiált, bármilyen formájú (fájl, értékek listája) adat (hálózati konfiguráció, hardver beállítás, stb.), amely a generált kóddal egyesíthető, hogy azzal együtt a cél PLC-be le legyen töltve. Az ilyen adatokat az ISaGRAF kernel közvetlenül nem működötteti, és azok általában a cél PLC-re telepített más szoftverhez vannak rendelve. A rendelkezésre álló erőforrásokkal kapcsolatos további információk a hardver kézikönyvben találhatók.

#### ☐ **Az erőforrás definíciós fájl**

Az erőforrások az ISaGRAF project többi fájljaival együtt tárolt **„Erőforrás definíciós fájlban”** vannak definiálva. Ez egy tisztán ASCII szövegfájl, amelyet az ISaGRAF Erőforrás fordító dolgoz fel. Ez a fordító az alkalmazás kód felépítésekor automatikusan futtatódik. Ez a fejezet ennek a fájlnak a szintaxisát ismerteti. Az erőforrás definíciós fájl az ST nyelv lexikai szabályait használja. A szövegbe bárhová be lehet illeszteni „(”-al kezdődő és „)”-el végződő megjegyzéseket. A

karaktorsorokat egyszeres aposztrófok választják el egymástól. A numerikus értékek beírásánál alkalmazott lexika formátumok magyarázata ennek a kézikönyvnek a második felében található.

## **Nyelvi hivatkozás**

Az alábbiakban egy erőforrás definíciós fájlban használt kulcsszavak és utasítások listája látható.

### **ULONGDATA**

**Jelentése:** Egy erőforrást specifikál, amely integer értékek listája. Az értékek a célkódban tömörítetlen 32 bites integereként vannak tárolva. Az értékek az erőforrás definíciós fájlban meghatározott sorrendben vannak tárolva. Az értékeket vesszővel kell elválasztani egymástól. Az erőforrás nevének hossza nem haladhatja meg a 15 karaktert

**Szintaxis:** **ULONGDATA** '<resource\_name>'  
**BEGIN**  
...target\_selection...  
...list of values...  
**END**

**Példa:** ULongData 'MYDATA'  
Begin  
...  
0, -1, 100\_000, (\* decimális \*)  
16#A0B1, 2#1011\_0101 (\* hexadecimális, bináris \*)  
End

### **VARLIST**

**Jelentése:** Egy erőforrást specifikál, amely változó címek listája. A változókat az erőforrás definíciós fájlban levő nevük azonosítja. A változó címek a célkódban előjel nélküli 16 bites integereként vannak tárolva. A címek az erőforrás definíciós fájlban meghatározott sorrendben vannak tárolva. A címeket vesszővel kell elválasztani egymástól. Az erőforrás nevének hossza nem haladhatja meg a 15 karaktert

**Szintaxis:** **VARLIST** '<resource\_name>'  
**BEGIN**  
...target\_selection...  
... list of variable names...  
**END**

**Példa:** VarList 'LIST'  
Begin  
...  
Var100, MyParameter, Command, Alarm  
End

### **BINARYFILE**

**Jelentése:** Egy bináris fájl erőforrást specifikál. A forrás adatok egy MS-DOS fájlban vannak tárolva. A cél erőforrás definíciót egy cél elérési útvonal név egészíti ki. Az ISaGRAF Erőforrás fordító a sor vége karaktereket nem konvertálja. Az erőforrás nevének hossza nem haladhatja meg a **15** karaktert

**Szintaxis:** **BINARYFILE** '<resource\_name>'  
**BEGIN**  
 ...target selection...  
 FROM '<source\_pathname>'  
 TO '<destination\_pathname>'  
**END**

**Példa:** BinaryFile 'MYFILE'  
 Begin  
 ...  
 From 'c:\user\config.bin'  
 To '/dd/user/appl/config.dat'  
 End

## TEXTFILE

**Jelentése:** Egy szövegfájl erőforrást specifikál. A forrás adatok egy ASCII fájlban vannak tárolva. A cél erőforrás definíciót egy cél elérési útvonal név egészíti ki. Az ISaGRAF Erőforrás fordító a sor vége karaktereket a cél gazdarendszer konvencióinak megfelelően konvertálja. Az erőforrás nevének hossza nem haladhatja meg a **15** karaktert

**Szintaxis:** **TEXTFILE** '<resource\_name>'  
**BEGIN**  
 ...target selection...  
 FROM '<source\_pathname>'  
 TO '<destination\_pathname>'  
**END**

**Példa:** TextFile 'MYFILE'  
 Begin  
 ...  
 From 'c:\user\config.bin'  
 To '/dd/user/appl/config.dat'  
 End

## TARGET

**Jelentése:** Egy célkód nevét specifikálja, amelynek tartalmaznia kell az erőforrást. A kezelt célokkal kapcsolatos bővebb információk az előző fejezetben (programfordító beállítások) találhatóak. A „**Cél**” utasítás ugyanabban az erőforrás blokkban egynél többször is előfordulhat, több cél kiválasztásához. Ez az utasítás nem használható, ha a „**BármilyenCél** („AnyTarget”)” utasítás specifikálva van.

**Szintaxis:** **TARGET** '<target\_name>'

```
Példa:  BinaryFile 'MYFILE'
        Begin
          Target 'ISA86M'
          Target 'ISA86M'
          ...
        End
```

### ANYTARGET

*Jelentése:* Azt specifikálja, hogy az erőforrást a Kódgenerátorral épített összes célkóddal egyesíteni kell. Az ISaGRAF Kódgenerátor több különböző célkódot tud előállítani egy „**Létrehozás**” parancson belül. Ez az utasítás nem használható, ha egy vagy több „**Cél**” utasítás specifikálva van.

*Szintaxis:* **ANYTARGET**

```
Példa:  ULongData 'MYDATA'
        Begin
          AnyTarget
          ...
        End
```

### FROM

*Jelentése:* Egy **BinárisFájl** vagy **SzövegFájl** erőforrás elérési útvonalának nevét specifikálja (azon a PC-n, ahol az ISaGRAF Workbench van telepítve). Az útvonalnév komponenseinek elválasztására szolgáló karaktereknek (meghajtó, alkönyvtár, előtag, utótag) az MS-DOS rendszer konvencióinak megfelelőnek kell lenniük.

*Szintaxis:* **FROM** '<target pathname>'

```
Példa:  BinaryFile 'MYFILE'
        Begin
          ...
          From 'c:\user\config.dat'
          To '/dd/user/appl/config.dat'
        End
```

### TO

*Jelentése:* Egy **BinárisFájl** vagy **SzövegFájl** erőforrás rendeltetési útvonalának nevét specifikálja (a célrendszeren). Az útvonalnév komponenseinek elválasztására szolgáló karaktereknek (meghajtó, alkönyvtár, előtag, utótag) a cél gazdarendszer konvencióinak megfelelőnek kell lenniük.

*Szintaxis:* **TO** '<target pathname>'

```
Példa:  TextFile 'MYFILE'
        Begin
          ...
```

```

      From 'c:\user\config.dat'
      To '/dd/user/appl/config.dat'
End

```

## **Példa**

Az alábbiakban egy erőforrás definíciós fájl teljes példája szerepel:

```

(* erőforrás definíciós fájl *)

ULongData 'DATA1'          (* értékek listája *)
Begin
  Target 'ISA86M'           (* csak ehhez a célhoz *)
    1, 0, 16#1A2B3C4D, +1, -1 (* numerikus értékek *)
End

VarList 'VLIST1'           (* változók listája *)
Begin
  Target 'ISA86M'           (* csak ehhez a célhoz *)
    Valvel, StateX, Command, Alrml (* változónevek *)
End

BinaryFile 'FILE1'         (* bináris fájl erőforrás *)
Begin
  AnyTarget                 (* az összes célhoz utalva *)
    From 'c:\user\updatef.bin' (* forrásfájl a PC-n *)
    To 'updatef.cfg'          (* cél fájl a PC-n *)
End

TextFile 'FILE2'           (* szövegfájl erőforrás *)
Begin
  Target 'ISA86M'
    From 'c:\nw\nwbd.txt'     (* forrásfájl a PC-n *)
    To '/nw/dat/nwbd'        (* cél fájl a PC-n *)
End

```

## **Erőforrás fordítás**

Ha az erőforrás definíciós fájlban erőforrások lettek bevíve, akkor az ISaGRAF kódgenerálás végén megjelenik egy párbeszédablak. Az erőforrás fordító futatásához nyomja meg a „**Fordítás elkezdése**” gombot. Az output üzenetek és hibák a fő vezérlőben lesznek megjelenítve. Az erőforrás fordítás elkerüléséhez nyomja meg a „**Kilépés**” gombot. Ebben az esetben nem lesznek erőforrások adva az ISaGRAF kódhoz.

## **Megvalósítás**

Az ISaGRAF nem korlátozza az erőforrások számát, valamint az adatsorok és fájlok méretét. Az erőforrások a generált kód végén, egy erőforrás jegyzékkel vannak eltárolva. Az alábbiakban az erőforrás jegyzék formátuma látható (C elnevezéseket használva):

```

RESOURCE:
{
  long nbres;          /* definiált erőforrások száma */
  {

```

```

char name[16];           /* erőforrás neve */
long type;               /* erőforrás adattípus */
long size;               /* adatblokk pontos mérete */
uint32 data;
uint32 path_offset;      /* egy karaktersorra mutat */
} /* nb of records */

```

Az alábbiakban a „típus” mező lehetséges értékei láthatók:

- 1 = bináris fájl
- 2 = szövegfájl
- 3 = ulong adatok (ebben az esetben nincs használva a path\_offset mező)
- 4 = változólista (ebben az esetben nincs használva a path\_offset mező)

A szövegfájloknál a sor vége karaktereket az erőforrás fordító a célrendszer konvencióinak megfelelően fordítja le. Minden mutató 32 bites eltolással rendelkezik a megfelelő struktúra címétől. Minden erőforrás név és útvonalnév NULLA lezárású karaktersor. Az útvonal nevek és adatok az erőforrás jegyzékét követik.

## A.14 Kereszthivatkozások

Az ISaGRAF workbench egy kereszthivatkozás szerkesztőt tartalmaz, amely a felhasználó számára a projekt programjaiban a deklarált változók teljes áttekintését, valamint alkalmazásaik helyét nyújtja. A kereszthivatkozás célja a projektben deklarált összes változó felsorolása, és az egyes programok forrásánál a forráskód azon részeinek a megkeresése, ahol azok a változók használva vannak. A kereszthivatkozások nagyon hasznosak egy változó életciklusának globális megtekintéséhez. Ezek segítenek a mellékhatások megkeresésében, és lecsökkentik a projekt karbantartás közbeni megértéséhez szükséges időt. A kereszthivatkozások egy projekt teljes szótárának globális megtekintésére is használhatók, így a nem használt változók könnyen megtalálhatók és a projekt összetettsége felmérhető.

A bal oldali lista a projekt deklarált tárgyait (programok, változók, és definiált szavak), valamint a projektben hivatkozott könyvtárelemeket (funkciók és funkcióblokkok) mutatja. A jobb oldali lista az első listában pillanatnyilag kiválasztott tárgy előfordulásait mutatja a programban.

Egy előfordulás leírása tartalmazza a program nevét, az FC vagy SFC lépések, átmenetek, vagy tesztek számát, plusz szöveges nyelvknél a sorok számát, LD vagy FBD diagramoknál pedig a koordinátákat. Gyors LD diagramokhoz a leírást a létrafok száma egészíti ki. Ha a változó outputként (vagy tegercsként) van használva, akkor a létrafok számát egy csillag („\*”) karakter követi.

A **„Beállítások”** menü **„Használatlan változók megjelenítése”** beállítás kiválasztásával a fő listában olyan változók is megjelennek, amelyek nincsenek használva az alkalmazás programjaiban.

### **Tárgy típusának kiválasztása**

Mivel egy projekt óriási számú deklarált tárgyat csoportosíthat, ezért a szerkesztő eszközsoron levő kombinált ablak azoknak a tárgyaknak a kiválasztására szolgál, amelyeket meg kell jeleníteni az ablakban. Ez lehetővé teszi a felhasználó számára a kiválasztott információkhoz való hozzáférést.

A kereszthivatkozások minden újraszámításakor a kiválasztás visszaállítódik **„Valamennyi tárgy”**-ra, a teljes lista megjelenítése érdekében.

### **Kereszthivatkozások újraszámítása**

A **„Fájl / Újraszámítás”** parancs bármikor használható a kereszthivatkozások frissítésére a más ISaGRAF szerkesztő ablakokban bevitt módosításoknak megfelelően.

### **Kereszthivatkozások exportálása**

Az **„Eszközök / Export”** parancs a kereszthivatkozások teljes listájának egy ASCII szövegfájlba történő kiírására szolgál. Ez a fájl ezután más alkalmazásokkal, pl. a Windows NotePad-dal vagy szövegszerkesztővel kinyitható.



### **Szótárhibák**

A **„Szerkesztés / Szótárhibák”** parancs egy párbeszédablakban megjeleníti a projekt szótár betöltésekor érzékelt hibák listáját.



### **Statisztikai adatok**

Az „**Eszközök / Statisztikai adatok**” parancs egy párbeszédablakban megjeleníti a projektben deklarált tárgyak és változók számát, a változó típusok és attribútumok szerint. Ennek a parancsnak egyik alkalmazása a projektben deklarált I/O változók számának megismerését szolgálja annak érdekében, hogy az biztosan lefordítható legyen olyan esetben, amikor az ISaGRAF Workbench korlátozott verziója van használva.



### **Keresés a tárgy listában**

A felhasználó a „**Szerkesztés / Keresés**” parancs segítségével egy tárgyat közvetlenül kiválaszthat a szerkesztő listából. A keresett tárgy nem található, ha az nincs ténylegesen felsorolva (egy kiválasztott megjelenítés használatakor). Egy tárgy keresése előtt ajánlatos az eszközsoron az „**Összes**” választást aktivizálni.



### **Program megnyitása**

A jobb oldali lista a kiválasztott tárgy előfordulásait mutatja a nyitott projekt forrásfájljaiban és az I/O csatlakozásban. A felhasználó a „**Szerkesztés / Program megnyitása**” parancs segítségével egy programot közvetlenül megnyithat ott, ahol a tárgy megjelenik. Lehetséges az egérrel egy előfordulásra történő dupla kattintás is (az előfordulás listában) a megfelelő program megnyitásához.

## A.15 A grafikus hibakereső használata

Az ISaGRAF egy teljes grafikus és szimbolikus hibakeresőt tartalmaz. A hibakeresőt a programrendező ablak „Hibakeresés” parancsa indítja el, a cél PLC-be letöltött alkalmazás irányításához. Ebben a módban a hibakereső egy hardverkapcsolaton keresztül kommunikál a célrendszerrel. A programrendező ablak „Szimuláció” parancsa egyidejűleg futtatja a hibakeresőt és egy komplett célszimulátort. Ez lehetővé teszi a felhasználó számára, hogy alkalmazását letesztelje olyankor is, amikor a cél I/O rendszere még nincs befejezve. A hibakereső ablak tartalmazza az egész alkalmazás vezérléséhez szükséges parancsokat.

Ha a cél PLC-ben levő alkalmazás ugyanaz, mint a workbenchen levő alkalmazás, akkor a hibakereső elindulásakor hibakeresési módban automatikusan kinyitja a **programrendező ablakot**. Ennek az ablaknak a parancsai felhasználhatók más ISaGRAF ablakok (grafikus és szövegszerkesztők, szótár, változólisták, I/O csatlakozás, stb.) kinyitására. Minden ablak, amely egy adott hibakeresési munkafázis során lett kinyitva, **„hibakeresési módban”** működik, ami azt jelenti, hogy a szerkesztési parancs le van tiltva. A megjelenített programkomponensek (lépések, átmenetek, változók, stb.) pillanatnyi futtatási státuszukkal vagy értékükkel jelennek meg. Egy tárgyra duplán rákattintva megváltozik annak státusza, illetve értéke a cél alkalmazásban.

A hibakeresőt **szimulációs módban** futtatva az ISaGRAF célrendszerrel történő kommunikáció megáll. A hibakereső csak a szimulátor ablakkal kommunikál. Mivel a célrendszer ebben a módban nem létezik, ezért a hibakereső menü a **„letöltés”**, **„megállítás”** illetve **„aktiválás”** parancs le van tiltva.

### A.15.1 A hibakereső ablak

A hibakereső ablak csak a teljes alkalmazás státuszáról tartalmaz információkat. Ez a többi ISaGRAF ablakhoz kapcsolva egy teljes interaktív hibakereső rendszert alkot. Az érzékelt futtatási hibák a hibakereső ablak alsó részén jelennek meg. A **„Beállítások”** menü parancsai a hibalista eltüntetésére, megjelenítésére, vagy kiürítésére szolgálnak.

A vezérlőpanel (a hibakereső menü alatti terület) a cél alkalmazás státuszát, valamint a végrehajtási ciklusidőzítéssel kapcsolatos információkat mutatja. A lehetséges cél státuszok listája a következő:

**Naplózás:** ..... A hibakereső kommunikációt hoz létre a célrendszerrel.

**Leválasztva:** ..... A hibakereső nem tud kommunikálni a célrendszerrel. Ellenőrizze, hogy a csatlakozókábel és a kommunikációs paraméterek megfelelőek legyenek.

**Nincs alkalmazás:** ..... A csatlakozás rendben van, de jelenleg nem létezik ISaGRAF alkalmazás a célrendszerben. Töltsön le egy alkalmazást.

**Az alkalmazás aktív:** ..... A csatlakozás rendben van, és a célrendszerben van egy aktív alkalmazás. A hibakereső most létrehozza a kommunikációt ezzel az alkalmazással, amennyiben az megegyezik a Workbenchen levő alkalmazással.

**FUTÁS:**..... A cél alkalmazás „Valós idejű” módban van.  
**MEGÁLLÍTÁS:**..... A cél alkalmazás „Ciklustól-ciklusig” módban van.  
**Megszakítási pont:**..... A cél alkalmazás „Ciklustól-ciklusig” módban van, mert egy megszakítási ponttal találkozott.  
**Végzetes hiba:**..... A cél alkalmazás sikertelen lett, mert egy súlyos hiba következett be.

Az alábbiakban a futtatási ciklusidőzítéssel kapcsolatos információk láthatók:

**Engedélyezett:**..... programozott időzítés.  
**Pillanatnyi:**..... az utolsó komplett végrehajtási ciklus pontos időzítése.  
**Maximális:**..... az alkalmazás elindítása óta érzékelt maximális időzítés.  
**Túlsordulás:**..... a megengedett időzítéstől magasabb időzítésű detektált végrehajtási ciklusok száma.

Minden időérték ezredmásodpercben van megadva. A hibakereső szimulációs módban történő használata közben az időértékek nem láthatók.

## A.15.2 Az alkalmazás vezérlése

A „**Fájl**” és „**Vezérlés**” menük a jelenleg szerkesztés alatt álló ISaGRAF alkalmazásnak az ISaGRAF célrendszerre történő telepítéséhez és vezérléséhez szükséges parancsokat tartalmazzák.

Megjegyzés: Ezek közül a parancsok közül némelyik nem áll rendelkezésre szimuláció közben, mert a szimulátor által feldolgozás alatt álló alkalmazást az ISaGRAF Workbench automatikusan telepíti.



### **A cél alkalmazás megállítása**

A „**Fájl / Alkalmazás megállítása**” parancs megállítja az ISaGRAF célrendszerben jelenleg aktív alkalmazás végrehajtását.



### **A cél alkalmazás aktiválása**

A „**Fájl / Alkalmazás elindítása**” parancs futtatja a célrendszerben jelenleg aktív alkalmazást. Egy alkalmazás letöltését követően automatikusan elindul, így az „**Elindítás**” parancsot nem kell használni. Az „**Elindítás**” parancsot általában a „**Leállítás**” parancs után használják.

Megjegyzés: Egy új alkalmazás letöltése csak akkor lehetséges, ha előbb a cél alkalmazást megállítják (inaktívválik).



### **Az alkalmazás letöltése**

A „**Fájl / Letöltés**” parancs az alkalmazás kódjának a célrendszerbe történő letöltésére szolgál. Válassza ki a letöltendő kód típusát a célrendszer processzorának és az alkalmazás beállításainak megfelelően.



### **A verziószám megjelenítése**

A „**Fájl / Verziószám megjelenítése**” parancs a Workbench és a cél alkalmazások teljes azonosításának megjelenítésére szolgál. A Workbench alkalmazás az, amely jelenleg meg van nyitva az ISaGRAF Workbenchben. A cél alkalmazás az, amely

jelenleg végrehajtás alatt van az ISaGRAF PLC-ben. A következő adatok kerülnek megjelenítésre:

**VERZIÓ:** ..... Ez az alkalmazás kódjának verziószáma. Ezt a számot a kódgenerátor számította ki.

**DÁTUM:** ..... Ez a kód felépítésének dátumát és időpontját mutatja.

**CRC:** ..... Ez egy ellenőrző összeg, amely a szimbólumtábla tartalmával lett kiszámítva. Ezt a számot a kódgenerátor számította ki. Ez az érték a változószótár tartalmától függ.

Megjegyzés: A „**Verziószám megjelenítése**” parancs szimuláció közben is rendelkezésre áll. Valódi hibakeresési módban ez a parancs nem használható, ha a cél PLC nincs csatlakoztatva.



### **Online módosítás**

A „**Fájl / Alkalmazás frissítése**” parancs lehetővé teszi a felhasználó számára a futó cél alkalmazás „online módosítását”. Ez a parancs ennek a fejezetnek a későbbi részeiben részletesen ismertetve lesz. A hibakereső szimulációs módban történő használata közben ez a parancs nem áll rendelkezésre.



### **Valós idejű mód**

A „**Vezérlés / Valós idejű**” parancs nem áll rendelkezésre olyankor, amikor az alkalmazás aktív. Ez a cél alkalmazást normál „valós idejű” módba váltja: Normál mód: a végrehajtási ciklusokat a programozott ciklusidőzítés indítja el.



### **Ciklustól-ciklusig mód**

A „**Vezérlés / Ciklustól-ciklusig**” parancs nem áll rendelkezésre olyankor, amikor nincs aktív alkalmazás. Ez a cél alkalmazást normál „valós idejű” módba váltja: Ebben a módban a ciklusok egyesével, a felhasználó által a hibakereső menüről beadott „**Egy ciklus végrehajtása**” parancsoknak megfelelően kerülnek végrehajtásra.



### **Egy ciklus végrehajtása**

Amikor a célciklustól- ciklusig módban van, a „**Vezérlés / Egy ciklus végrehajtása**” parancs egy ciklus végrehajtását futtatja.



### **A ciklus időzítése**

A „**Vezérlés / Ciklusidőzítés megváltoztatása**” parancs segítségével a felhasználó módosíthatja a programozott ciklusidőzítést. Ennek az időnek a neve „**Engedélyezett**” a hibakereső vezérlősor ablakban. A ciklusidőzítés módosítása előtt be kell állítani a „**Ciklustól-ciklusig**” módot. A ciklusidőzítést egész számként, ezredmásodpercekben kell beírni.



### **Az összes megszakítási pont eltávolítása**

A „**Vezérlés / Az összes megszakítási pont eltávolítása**” parancs eltávolítja az összes jelenleg telepített (akár már bekövetkezett, akár még aktív) megszakítási pontokat az egész alkalmazásból. A meglévő megszakítási pontok nem lesznek automatikusan eltávolítva a hibakereső ablak bezárásakor.

## **I/O változók kireteszelése**

A „**Vezérlés / Összes IO változók kireteszelése**” parancs felbontja az alkalmazásban jelenleg reteszelt állapotban levő összes I/O változó reteszelését. Amikor egy I/O változó reteszelve van, a megfelelő I/O eszköznél nem lesz végrehajtva input vagy output státuszváltozás. Az alkalmazás vagy a hibakereső továbbra is írhat az I/O-hoz csatolt változókat. A jelenleg reteszelt I/O változók nem lesznek automatikusan kireteszelve a hibakereső ablak bezárásakor.

### **A.15.3 Beállítási lehetőségek**

A „**Beállítási lehetőségek**” menü a hibakereső ablakban megjelenő információk vezérlését teszi lehetővé.

## **A kommunikációs paraméterek**

A kommunikáció időzítési paramétereket a hibakereső aktív állapotában lehet állítani. Itt csak a kommunikációs **időtúllépéseket** lehet beállítani. A többi kommunikációs paramétert (átviteli sebesség, paritás, stb.) a programrendező ablak „**Hibakeresés**” menüjéről kell beállítani.

A „**Kommunikációs időtúllépés**” az az idő, amelyen belül a célrendszernek meg kell kezdenie a válaszadást egy workbench kérésre. A „**Ciklikus frissítési időtartam**” az az időtartam, amelyre a hibakeresőnek az „**olvasás**” parancs elküldéséhez van szüksége a megnyitott ablakokban levő adatok frissítésére.

Minden időérték egész számként, **ezredmásodpercben** van megjelenítve és beírva. A kommunikáció időzítési paramétereket a hibakereső szimulációs módban történő használata során nem lehet állítani.

## **Megjelenítési lehetőségek**

A „**Ciklusidőzítés megjelenítése**” beállítás segítségével a felhasználó megjelenítheti, illetve eltüntetheti a **ciklusidőzítés** értékek megjelenítését a hibakereső vezérlősorában. Ennek a lehetőségnek a beállításakor minden ciklusidőzítési komponens (engedélyezett, pillanatnyi, maximális, túlsordulások) megjelenik, és frissítésre kerül. Ennek a lehetőségnek a letiltása csökkenti a hibakereső kommunikációs terhelését.

Amikor a „**Hibák megjelenítése**” lehetőség be van állítva, az érzékelt futtatási hibák a hibakereső ablak alsó részén jelennek meg. Ha ez a lehetőség le van tiltva, a hibalista bezáródik. Ennek a lehetőségnek az eltávolítása csökkenti a hibakereső megjelenítési és kommunikációs terhelését. A „**Beállítások / Hibák törlése**” parancs kitörli a hibakeresőben jelenleg megjelenített futtatási hibák listáját.

A „**Beállítások / Ablak minimalizálása**” parancs a hibakereső ablak méretét lecsökkenti úgy, hogy az egy mindig felül levő panelként jelenik meg, amely csak az alkalmazás státuszát és a leggyakrabban használt parancsok grafikus gombjait tartalmazza.

### **A.15.4 „Írás” parancsok**

Az ISaGRAF szimbolikus hibakereső számos parancsot kínál az alkalmazás komponensek **értékeinek** vagy **státuszának** a megváltoztatásához. A változtatandó komponens kiválasztása egy szerkesztőablakban annak nevére

illetve rajzára való **kettős kattintással** történik, amikor a hibakereső ablak nyitva van.



## Változók

Egy változó státuszát a nevére történő kettős kattintással lehet megváltoztatni a következő ablakok valamelyikében:

- Szótár
- Változók vagy idődiagramok listái
- LD vagy FBD Programok
- I/O csatlakozás

A hibakereső párbeszédablak a következő parancsokat kínálja:

- A változók átirása egy új értékre
- A változó **Reteszélése** (csak I/O változóknál)
- A változó **Kireteszelése** (csak reteszelt I/O változóknál)
- Egy időzítés változó **Elindítása** vagy **Leállítása** (állítsa be az automatikus frissítési módot)

A logikai **IGAZ** és **HAMIS** értékeket képviselő szimbolikus értékek a szótárban az adott logikai változóhoz definiált karaktorsorok. Egy „**Írás**” parancshoz meghatározott analóg értéket egész vagy reális szám formájában kell megadni, a változó szótárban levő definíciójának megfelelően. Az „**Írás**” parancsnál egy üzenethez meghatározott karaktorsor nem lehet hosszabb, mint az adott változóhoz a szótárban kapcsolt üzenet-kapacitás.



## SFC tárgyak

Egy **SFC program** vezérlési működésének az alkalmazás hibakeresése közben történő megfigyeléséhez a Programrendező ablakban levő „**Fájl**” menü parancsai szolgálnak. Az SFC programot a programok listájából kell kiválasztani. A következő parancsok állnak rendelkezésre:

**SFC program indítása:** . Engedélyezi a kiválasztott programot azzal, hogy minden kezdeti lépésébe elhelyez egy SFC vezérlőjelet.

**SFC program megszüntetése:** Megszünteti a kiválasztott programot, összes létező vezérlőjelének eltávolításával.

**SFC program befagyasztása:** Felfüggeszti a kiválasztott program végrehajtását.

**SFC program újraindítása:** .... Újraindít egy felfüggesztett (befagyasztott) programot.

Gyermekprogramoknál ezek a parancsok a „**GSTART**”, „**GKILL**”, „**GFREEZE**” és „**GRST**” funkcióknak felelnek meg a programozási nyelvben.

Az alkalmazás hibakeresése során egy vezérlési művelet egy **SFC lépésben** úgy tehető láthatóvá, ha annak grafikus képére duplán rákattint az SFC szerkesztőablakban. A hibakereső párbeszédablakban a következő parancsok állnak rendelkezésre:

- Egy megszakítási pont telepítése a lépés **aktiválására**
- Egy megszakítási pont telepítése a lépés **deaktiválására**
- A lépéshez adott megszakítási pont **Törlése**

**Megjegyzés:** Megszakítási pontok aktiválása és deaktiválása nem adható ugyanahhoz a lépéshez.

Az alkalmazás hibakeresése során egy vezérlési művelet egy **SFC átmenetben** úgy tehető láthatóvá, ha annak grafikus képére duplán rákattint az SFC szerkesztőablakban. A hibakereső párbeszédablakban a következő parancsok vannak felkínálva:

- Egy **megszakítási pont** telepítése az átmenet törlésén
- Az átmenethez adott megszakítási pont **Törlése**
- Az összes átmenet (vezérlőjelek áthelyezése vagy hozzáadása) manuális **törlése**

**Feltételes törlés:** az átmenetet követő lépéseken egy vezérlőjel van létrehozva. A megelőző lépésekben létező vezérlőjelek eltávolításra kerülnek. **Feltétel nélküli törlés:** az átmenetet követő lépéseken egy vezérlőjel van létrehozva. A megelőző lépésekben létező vezérlőjelek nem kerülnek eltávolításra.

### A.15.5 Online módosítás

Az „Online módosítás” lehetőség segítségével a felhasználó a folyamat futása közben módosíthatja az alkalmazást. Erre néha vegyi folyamatoknál lehet szükség, ahol bármiféle félbeszakítás veszélyeztetheti a termelést, illetve a biztonságot. Ezt a funkciót **nagyon körültekintően** kell használni. Előfordulhat. Hogy az ISaGRAF esetleg nem detektálja az összes lehetséges konfliktust, amit ezeknek az online változásoknak az eredményeképpen a felhasználó által generált változások idézhetnek elő.

#### **Kódszekvenciák**

Mivel az ISaGRAF számos lehetőséget kínál a változók, programok, vagy I/O kártyák hibakeresőből történő eléréséhez, ezért az itt ismertetett „Online módosítás” funkció csak a kódszekvenciák módosítására vonatkozik. Egy kódszekvencia az egy sorban végrehajtott ST, IL, LD vagy FBD utasítások teljes készletét jelenti. Egy „ciklus kezdete” vagy „ciklus vége” programban a kódszekvencia a programba írt utasítások teljes listáját jelenti. Egy SFC programban a kódszekvencia egy lépésnek vagy átmenetnek a 2-es szintű programozása. Az „Online módosítás” egy vagy több kódszekvenciának a PLC végrehajtási ciklus megszakítása nélkül történő cseréjéből áll. Mivel az SFC vezérlőjelek vezérlése kritikus fontosságú, **ezért egy SFC struktúra nem módosítható egy lépés, átmenet, vagy egy SFC program hozzáadásához, átszámozásához, illetve eltávolításához.**

#### **Változók**

Mivel a változó adatbázis az alkalmazás kimondottan kritikus fontosságú részét képezi, ezért ahhoz más folyamatok bármikor hozzáférhetnek (egyidejűleg több feladatot futtató PLC-n). A változók értékei a hibakeresőből is módosíthatók. Ennélfogva az ISaGRAF **nem teszi lehetővé a felhasználó számára egy változó hozzáadását, átnevezését, vagy eltávolítását online módon.** Ettől függetlenül azonban módosítható az, hogy egy változó milyen módon van felhasználva az alkalmazásban. Ugyancsak lehetséges a „nem használt” belső vagy I/O változók

lefoglalása az alkalmazás első verziójában azért, hogy a későbbi módosítások hasznosítani tudják azokat.

Az ISaGRAF cél adatbázisban különböző stílusú változók léteznek. A korlátozások mindegyikükre vonatkozik:

#### *- Deklarált változók*

Ezek azok, amelyek az ISaGRAF szótárban vannak deklarálva. Ezek nem változtathatók és nem nevezhetők át online változtatáshoz. Ajánlatos bizonyos mennyiségű többlet változó deklarálása és inicializálása az alkalmazásban még akkor is, ha azokat az adott időpontban nem használják. Ezek a többlet változók lehetővé teszik a későbbi módosítások elvégzését anélkül, hogy meg kellene változtatni az alkalmazás adatellenőrző összegét.

#### *- Funkcióblokkok előfordulásai*

Minden „C”-ben vagy IEC-ben írt funkcióblokk az ISaGRAF cél valós idejű adatbázisában tárolt adatoknak felel meg. Funkcióblokk előfordulások hozzáadásakor vagy eltávolításakor többé nem lehetséges az Online változtatás. Ezért jobb ST-ben dolgozni a szótárban deklarált FB előfordulásokkal, mint blokkokat hozzáadni (amelyek új automatikusan deklarált előfordulásoknak fognak megfelelni) Gyors LD vagy FBD diagramokban. Az ISaGRAF könyvtárban rendelkezésre álló funkcióblokkok definíciójának bármiféle változtatása ugyancsak lehetetlenné teszi az OnLine változtatást.

#### *- Lépések*

Minden SFC lépés egy adatnak felel meg, amelyben az SFC lépés dinamikus attribútumai (aktivitás ideje és jelzője) vannak tárolva. Az SFC lépések hozzáadása vagy eltávolítása megváltoztatja az alkalmazás adatbázisát, és tilossá válik az OnLine változtatás.

#### *- A fordítóprogramok által allokalált rejtett változók*

Az ISaGRAF Fordítóprogram a komplex kifejezések megoldásához „rejtett” átmeneti változókat generál. Bizonyos esetekben egy kifejezés megváltoztatása egy eltérő láthatatlan átmeneti változókészlethez vezethet, és ez lehetetlenné teheti az OnLine változtatást. Ennek a helyzetnek az elkerülése érdekében a következő sorokat lehet beírni az ISA.INI fájlba annak érdekében, hogy minden programhoz egy minimális számú átmeneti változó allokalása ki legyen kényszerítve még akkor is, ha azok nincsenek felhasználva az alkalmazás első verziójának fordításakor. Az itt megadott értékek csak példaként szerepelnek:

```
[DEBUG]
MNTVboo=8      ; logikai értékekhez
MNTVana=4      ; integer és reális értékekhez
MNTVtmr=4      ; időzítés értékekhez
MNTVmsg=2      ; üzenetekhez
```

Amikor egy ilyen beállítás lett az ISA.INI-be írva, a fordítóprogram egy figyelmeztető üzenetet közöl, ha az alkalmazás egy újabb fordítása az allokalált átmeneti változók számánál magasabb számhoz vezet.

## Inputok és outputok

Mivel az ISaGRAF I/O rendszer nagyon nyitott, ezért jobb, ha a szükséges módosításokat a megfelelő hardver adott jellemzőinek felhasználásával az OEM (Eredeti készülék gyártó) vezeti be. Az ISaGRAF rendszer **nem teszi lehetővé, hogy a felhasználó online módon egy I/O változót adjon hozzá, csatlakoztasson vagy eltávolítson, illetve egy I/O kártya leírását módosítsa.** Olyan műveletek, mint pl. a kártyaparaméterek módosítása illetve I/O csatornák reteszelése a standard OEM jellemzők valamint a „**MŰKÖDTETÉS**” funkció használatával állnak rendelkezésre.

## Futtatási műveletek

Egy futó alkalmazás módosítása a következő műveletekből áll:

- az alkalmazás forráskódjának módosítása a workbenchen
- az új alkalmazás kód generálása
- az új alkalmazás kód letöltése a „**Frissítés**” paranccsal a „**Letöltés**” parancs helyett
- a régi alkalmazásból az újra történő átváltás a PLC végrehajtási ciklusok között a „**Frissítés végrehajtása**” paranccsal.

Ez az eljárás garantálja, hogy a cél PLC mindig egy teljes és megbízható futó alkalmazással rendelkezzen, és lehetővé teszi a felhasználó számára a minta műveletek időzítésének nagyon biztonságos és hatékony módon történő vezérlését. Ez amellet lehetővé teszi a felhasználó számára, hogy a projektet a lehető leggyakrabban lehessen módosítani. Magától a folyamatól függetlenül az „Online módosítás” alapvetően ugyanaz, mint a normál „**Megállítás**”, „**Elindítás**” és „**Letöltés**” parancsok együttes. A különbségek csupán abból állnak, hogy nem vész el változó állapot, és az átváltási idő nagyon rövid (általában 1 agy 2 ciklusidőnyi). Az átváltás közben nincs módosítva változó, és **valamennyi belső, input, vagy output változó megtartja ugyanazt az** alkalmazás módosítása előtti és az utáni **értéket.** Az átváltás közben nem történik beavatkozás végrehajtás, és az **SFC vezérlőjelek nem kerülnek áthelyezésre.**

## Memória igények

Az „Online módosítás” képesség támogatása érdekében a cél PLC-nek szabad memóriakapacitással kell rendelkeznie ahhoz, hogy lehetővé tegye az alkalmazás kód módosított változatának eltárolását. Az átváltási művelet közben az alkalmazás kód mindkét változatának a PLC memóriájában kell lennie.

## Korlátozások

Mint az már fentebb említve lett, csak a kódszekvenciák módosítása van megengedve. A változó definíciója, az alkalmazás paraméterei, és az I/O csatlakozások nem módosíthatók. Az alkalmazás módosított változatának letöltésekor az ISaGRAF összehasonlítja a módosított és a futó alkalmazást annak érdekében, hogy bármilyen veszélyes változást érzékelhessen. Ha az átváltás veszélyesnek illetve lehetetlennek tűnik, akkor egy letöltési hiba generálódik. Az ISaGRAF által végrehajtott egyik biztonsági intézkedés a szimbólumtáblázat ellenőrző összegének összevetése azért, hogy bármiféle változónak, programnak, vagy SFC elem nevének a változása érzékelhető legyen. Ha az átváltás lejátszódásakor egy lépés aktív, akkor annak nem tárolt (N) műveletei elvesznek. Az új lépés aktiválási műveletei nem kerülnek végrehajtásra. A lépés

deaktiválásakor végrehajtott műveletek átvivődnek az új alkalmazás kódba. Ha egy átmenet érvényes az átváltás végrehajtásakor, akkor annak fogadtatási egyenlete frissítődik. Az új letöltött alkalmazás kódjáról nem készül biztonsági másolat a PLC-n. A biztonsági másolat az a verzió lesz, amely korábban a standard letöltési parancsokkal lett letöltve.



## Műveletek

Egy futó alkalmazás kódjának frissítéséhez a következő műveleteket kell elvégezni:

- Mielőtt egy futó alkalmazáson bármiféle változtatást végezne, ajánlatos a pillanatnyi projektről egy eltérő névvel ellátott másolatot készíteni. A módosításokat el lehet végezni a másolatokon.
- Egy program szerkesztése előtt a felhasználónak ellenőriznie kell, hogy a szerkesztő eszközök „**Napló frissítése**” lehetősége be van-e állítva, mert ez megkönnyíti a későbbi program-karbantartást.
- Ha egy vagy több szekvencia lett módosítva (az SFC struktúrák és a program-hierarchia módosítása nélkül), akkor az új alkalmazás kódját a workbenchen generálni kell a letöltés előtt.
- A hibakeresőt a régi projektből használva, a felhasználónak csatlakoztatnia kell a cél PLC-t és végre kell hajtania minden olyan műveletet, amely az alkalmazás frissítését gyorsabbá vagy biztonságosabbá teszi.
- A hibakeresőt az új projektből használva, a felhasználónak csatlakoztatnia kell a cél PLC-t. Ha megváltozott az alkalmazás neve, akkor a cél adatbázis hozzáférhetetlenné válik. A felhasználónak a „**Fájl / Frissítés**” parancsot kell lefuttatnia.
- A módosított alkalmazás letöltése a „**Frissítés később**” lehetőség kiválasztásával történik. Ez az adatátvitel közben némileg lelassíthatja a PLC-t.
- Amikor a letöltés befejeződött, a felhasználó futtathatja a „**Fájl / Frissítés végrehajtása**” parancsot ahhoz, hogy az átváltás a legmegfelelőbb pillanatban történhessen meg. Az átváltás időtartama 1 vagy 2 ciklusidőtartam lesz.
- Ha az átváltás helyesen lett végrehajtva, akkor a módosított futó alkalmazás programjai jelenítődnek meg. Ha nem, akkor a meglévő futó alkalmazás változatlan marad.

### A.15.6 DDE cserék

Az ISaGRAF hibakereső tartalmaz egy DDE (Dinamikus adatcsere; Dynamic Data Exchange) szerveret. Az ISaGRAF hibakereső és más alkalmazások közé egy tanácsadó hurok illeszthető be annak érdekében, hogy a nem ISaGRAF alkalmazásokban levő változók pillanatnyi értéke dinamikusan meg legyen jelenítve. Az ISaGRAF hibakereső DDE szervere csak „tanácsadás” és „taszítás” műveleteket támogat. „Kérés” műveleteket csak azokhoz a változókhoz lehet használni, amelyek egy tanácsadási hurokban már ki lettek kémlelve. Más DDE szolgáltatások (pl. „végrehajtás”) nem állnak rendelkezésre. Amikor egy változón létre lett hozva egy tanácsadó hurok, akkor annak a változónak az értéke a kliens alkalmazásban minden változásakor frissítésre kerül. Bármely típusú változó kémlelhető. A dinamikus kapcsolat azonosítása a következő nevekből áll:

Szolgáltatás név: „ISaGRAF”

Téma név: .....Az ISaGRAF projekt neve

Tétel név: .....A változó neve

Ha a változó egy programot tekintve helyi jellegű, akkor annak neve után zárójelben apaprogramjának nevének kell szerepelnie, a következő szintaktika szerint:

**változó\_neve(program\_neve)**

Az ISaGRAF hibakereső DDE szervere a hibakereső által éppen kikémlelés alatt álló ISaGRAF alkalmazáshoz van rendelve. Az ISaGRAF szerver maximum **256** változót kémlelhet. A DDE szerver az ISaGRAF hibakereső csatlakoztatott vagy szimulációs módban történő futásakor egyaránt használható. A frissítési időtartam a hibakereső és az ISaGRAF célrendszer vagy szimulátor közötti kommunikációjához létrehozott időtartam.

## A.16 Változók kémlelési listája

A felhasználó a Hibakereső ablak „**Kémlelés**” menüjének „**Kémlelési listák**” parancsa segítségével változók nem folyatóságos listáját készítheti el, amelyek pillanatnyi értékükkel frissítődnek. A listák az alkalmazás hibakeresése során kerülnek létrehozásra. A listák a lemezen elmenthetők, és más hibakeresési eljárások során ismét megnyithatók. Egy lista maximum **32** változót tartalmazhat. Egy listán belül különböző típusú változók keverhetők. Egy listába beilleszthetők helyi és globális változók. Egy változólista egy adott projekthez tartozik. A változólisták nagyon hasznosak egy alkalmazás funkcionális teszteléséhez. Ezek lehetővé teszik a felhasználó számára a változások figyelemmel kísérését egy vezérelt folyamat kis részén belül, az alkalmazás programokban levő megfelelő forráskódtól függetlenül. A változólisták ST és IL szöveges programok hibakereséséhez is hasznosak. A felhasználó egy programban használt változók készletét egyszerűen egy listába csoportosíthatja a programozott utasítások végrehajtásának kézbe tartása vagy figyelemmel kísérése érdekében.

Az ISaGRAF a listában levő mindegyik változónak megjeleníti a nevét, pillanatnyi értékét, és megjegyzés szövegét. Az oszlopok a lista címsorában levő elválasztó vonalak egérrel történő vonzolásával átméretezhetők.



### **Listák kimentése a merevlemezre**

A „**Fájl**” menü parancsai a változólisták létrehozására, megnyitására, valamint kimentésére szolgálnak. Az ISaGRAF nem korlátozza egy projekt listáinak számát. A lemezre kimentendő változólisták elnevezésénél a következő szabályokat kell betartani:

- a név nem haladhatja meg a **8** karaktert
- az első karakternek **betűnek** kell lennie
- a többi karakter lehet **betű**, **sorszámjegy**, vagy aláhúzás karakter
- a listák elnevezése nem érzékeny a kisbetű/nagybetű beállításra.

A listaszerkesztő ugyanabban az ablakban egyszerre egynél több változólistát nem tud megjeleníteni. A listaszerkesztő azonban egyszerre több példányban is futtatható annak érdekében, hogy egyszerre több lista kémlélhető legyen.



### **Változók beillesztése a listába**

A „**Szerkesztés / Beillesztés**” parancs egy újabb változót illeszt be a listába. A változó neve a projekt szótárban definiált tárgy listában van kiválasztva. Ilyen módon a felhasználónak nem kell manuálisan beírnia az azonosítót. A változó a listában pillanatnyilag kiválasztott változó elé lesz beillesztve. A lista maximum **32** változót tartalmazhat. Ugyanaz a változó csak egyszer jelenhet meg a táblázatban.



### **A kiválasztott változó megváltoztatása**

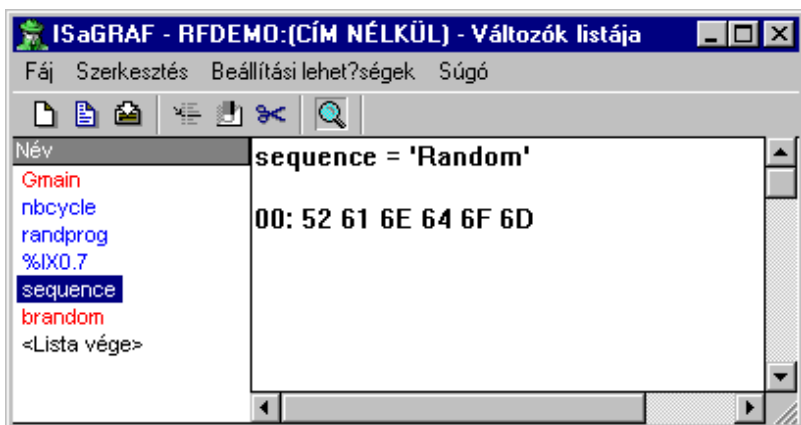
A „**Szerkesztés / Módosítás**” parancs a kiválasztott változót egy másik változóra cseréli le. A listáról a kiválasztott változó eltávolítására a „**Kivágás**” parancs is használható.



## Tárkiírás megjelenítés

Bármikor át lehet váltani a listázásos és folyamatos tárkiírási megjelenítési mód között. A megjelenítési mód átváltásához nyomja meg az eszközsoron levő „zoom” gombot, vagy használja a **„Beállítások / Tárkiírás”** parancsot.

A „Tárkiírás” módban csak egy változó érték van megjelenítve. Annak értéke az ablak felső részében számszerű/szimbolikus formában jelenik meg, és bináris „tárkiírás” formátumban is kijelzésre kerül. Ez a mód lehetővé teszi, hogy a változó érték minden bájtnak hexadecimális értéke kémlelhető legyen.



A „Tárkiírási” megjelenítés a nem nyomtatandó karaktereket tartalmazó üzenet karaktorsorok kémleléséhez és megértéséhez nagyon hasznos.

## A.17 ST és IL programok hibakeresése

ST és IL programok szimulációja vagy online hibakeresése közben a program szövegébe nem lehet módosítást bevinni.

### IL

IL programoknál az utasítások lista nézetben vannak formattálva. Egy utasításban használt változó pillanatnyi értéke ugyanazon a soron van megjelenítve. Egy utasításra duplán rákattintva megváltoztatható a megfelelő változó értéke.

### ST

ST programoknál a szerkesztőablakba egy Kémlelési lista ablak van beépítve. A nézeteket a közöttük levő elválasztó vonal egérrel történő vonszolásával lehet átméretezni.

Az ISaGRAF a listában levő mindegyik változónak megjeleníti a nevét, pillanatnyi értékét, és megjegyzés szövegét. Az oszlopok a lista címsorában levő elválasztó vonalak egérrel történő vonszolásával átméretezhetők.



### **Listák kimentése a merevlemezre**

A **„Fájl / Lista kimentése”** parancs a lemezre kimenti a változók listáját, a szerkesztett program nevével megegyező név alatt. Ez a lista automatikusan újra betöltődik, valahányszor az ST vagy IL program hibakereső módban meg van nyitva. Ez a lista a hibakereső ablak **„Kémlelés / Kémlelési lista”** parancs által futtatott Kémlelési lista eszközzel ugyancsak szabadon megnyitható és módosítható.



### **Változók beillesztése a listába**

A **„Szerkesztés / Változó beillesztése”** parancs egy újabb változót illeszt be a listába. A változó neve a projekt szótárban definiált tárgy listában van kiválasztva. Ilyen módon a felhasználónak nem kell manuálisan beírnia az azonosítót. A változó a listában pillanatnyilag kiválasztott változó elé lesz beillesztve. A lista maximum **32** változót tartalmazhat. Ugyanaz a változó csak egyszer jelenhet meg a táblázatban.



Amikor az ST szövegben ki van emelve egy változó neve, akkor az eszközsorról ennek a gombnak a megnyomásával, vagy a **„Szerkesztés / Kémlelés kiválasztás”** futtatásával a változó közvetlenül a beépített kémlelési listára küldhető.



### **A kiválasztott változó megváltoztatása**

A **„Szerkesztés / Változó módosítása”** parancs a kiválasztott változót egy másik változóra cseréli le. A listáról a kiválasztott változó eltávolítására a **„Változó kivágása”** parancs is használható.

## A.18 Hibakeresés a Fényszóróval

Az ISaGRAF Fényszóró eszköz lehetővé teszi a felhasználó számára figyelési listák definiálását, amelyek vagy grafikus ábrák, vagy egy lista formájában jeleníthetők meg a hibakeresés során. A grafikus tételeknek az ISaGRAF projekt változóihoz kell kapcsolódniuk. A grafikus ábra „online” van definiálva és animációval ellátva.

Egy változó értékének a kényszerítéséhez kattintson duplán a megfelelő tételre a grafikus vagy lista elrendezésből, vagy nyomja meg az ENTER gombot, ha a tétel ki van választva.

A dokumentum a „**Fájl / Reteszelés**” parancs segítségével reteszeltető is (ami bármiféle módosítást megakadályoz). Ha egy dokumentum reteszelve van, attól még lehet kényszeríteni a változókat, a szimbólumukra való dupla rákattintással.

### A.18.1 A grafikus elrendezés felépítése

Egy diagram háttérképekből (bitmapok vagy metafile-ok), és egy grafikus tételkészletből áll, amely a hibakeresés közben animációval lesz ellátva. A diagram beviteléhez a következő műveleteket kell elvégezni: Háttérképek beillesztése, grafikus tételek beillesztése, tárgyak kapcsolása a projekt változóihoz



#### **Háttérképek**

A háttérképek „bitmap” (.BMP) vagy „metafile” (.WMF) fájlok. A grafikus elrendezésben levő képek száma nincs korlátozva. A képek a grafikus elrendezésben áthelyezhetők vagy átméretezhetők. Ezek a lista elrendezésben nem jelennek meg. A képek felépítése más eszközökkel történik. A Fényszóró nem tartalmaz festő eszközt. A „**Beállítások / Háttérszín**” parancs a grafikus elrendezésben egy üres helyet kitöltő összefüggő szín kiválasztására szolgál.

Megjegyzés: A bitmapok sok memóriát fogyasztanak. Ajánlatos a kép megfelelő méretezése, és a bitmap négyzeten belül a kihasználatlan terület minimálisra csökkentése.

123

#### **Egyszeres szöveg megjelenítés**

Az „egyszeres szöveg” egy négyzetbe írt szöveget jelent. A megjelenített szöveg a csatolt változó értéke. Így egy ilyen tétel üzenet karaktersor változóhoz kapcsolható. A szöveg megjelenítésre szolgáló négyzet lehet átlátszó, vagy azt ki lehet tölteni egy színnel. A szöveg megjelenítésére használt karakter font a tétel átméretezésekor úgy állítódik be, hogy alkalmazkodjon a négyzet magasságához.

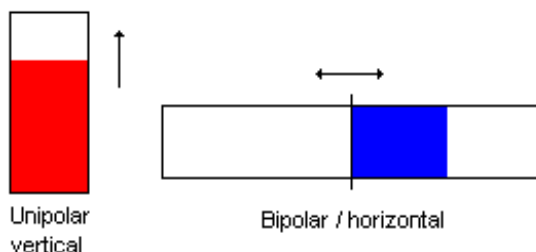


#### **Egypólusú és kétpólusú oszlopdiagramok**

Az oszlopdiagram egy színezett résszel ellátott téglalap, amely a csatolt változó számszerű értékét mutatja. A téglalap többi része választhatóan szintén feltölthető egy színnel. Az oszlopdiagram lehet függőleges vagy vízszintes.

Az egypólusú oszlopdiagramok bármelyik irányba nőhetnek: felfelé, lefelé, balra, vagy jobbra.

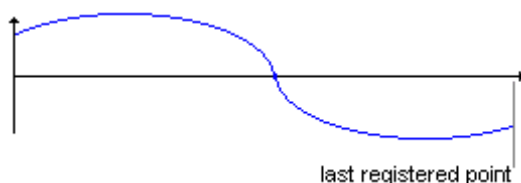
A kétpólusú oszlopdiagramok vagy pozitív vagy negatív irányba nőhetnek, a csatolt változó értékének megfelelően. Egy kétpólusú oszlopdiagram esetében a megengedett maximális érték mind a negatív, mind a pozitív skálánál megegyezik.



## Görbék

Egy dokumentumba beilleszthető egy görbe. A görbe a csatolt változó előzményeit mutatja. Bár ez nem egy pontos mérési eszköz, azonban hasznos hibakeresési információkat nyújt a különböző változók közötti összhangról.

Egy görbe egy változó utolsó 200 értékét tárolja. A görbe tételnek a grafikus elrendezésben történő átméretezésekor a minták száma nem változik.



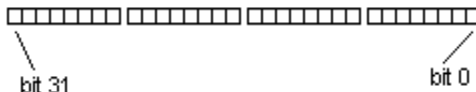
## Logikai ikonok

A „logikai ikon” tétel egy bináris állapot megjelenítésére használatos. Egy ikon (.ICO) fájl a HAMIS vagy 0 értékhez van definiálva. Egy másik ikon az összes többi nem zéró értékhez van definiálva. Mivel a Fényszóró nem tartalmaz ikonszerkesztőt, ezért az ikonfájlokat valamilyen másik eszközzel kell elkészíteni.



## Bit mezők

Egy „bit mező” tétel a grafikus panelben egy integer érték 32 bitjét mutatja. A legkevésbé szignifikáns bit mindig a jobb oldalon szerepel. Nem ajánlatos a bit mezőt más adattípusokhoz, pl. valós analóg értékekhez használni, mivel a megjelenített információ félreértésekhez vezethet.



### **Tételek kiválasztása, áthelyezése, vagy átméretezése**

A legtöbb szerkesztési parancshoz szükség van a grafikus tárgyak kiválasztására. A Fényszóró lehetővé teszi a diagramban levő egy vagy több tárgy kiválasztását. A tárgyak kiválasztásához a szerkesztő eszközsorán ki kell jelölni a „**kiválasztás**” lehetőséget (egy nyilat tartalmazó gomb). Egyetlen tárgy kiválasztásához a felhasználónak rá kell kattintania annak szimbólumára. Tárgyak listájának a kiválasztásához az egeret a diagramba kell vonszolni, és ki kell jelölni vele egy négyzet alakú területet. A kiválasztási négyzetben levő összes tárgy „**kiválasztott**” jelölést kap. Egy kiválasztott tárgyat a grafikus szimbóluma köré rajzolt kis fekete négyzetek jelölnék.

Egy új kiválasztás végrehajtásakor az összes előzőleg kiválasztott tárgy kiválasztott állapota megszűnik. A jelenlegi kiválasztás(ok) eltávolításához egyszerűen kattintson az egérrel egy, a kiválasztott tárgyakat befoglaló négyzeten kívüli üres területre.

A tárgyak áthelyezéséhez először ki kell választani azokat. Ezután az egeret a kiválasztott tétel keretére kell helyezni, és azt más helyre kell vonszolni.

Egy tárgy átméretezéséhez először azt ki kell választani. Ezután az egér kurzort a kiválasztott tétel keretén látható kis négyzetek egyikére kell helyezni, és azt a tárgy átméretezéséhez egy másik helyre kell vonszolni. A képek szintén átméretezhetőek. Ebben az esetben a megfelelő bitmap vagy metafile az új specifikált tétel-négyyszög kitöltéséhez a megfelelő irányba kinyújtódik.



### **Tételek csoportosítása / csoportok felbontása**

A tételek egybe csoportosíthatók, hogy egyetlen tételként lehessen őket kezelni. Egy csoport elkészítéséhez a grafikus elrendezésben válassza ki a tételeket, és futtassa a „**Szerkesztés / Csoportosítás**” parancsot. A „**Szerkesztés / Szétválasztás**” parancs a kiválasztott csoport tételeinek egyéni tételekké történő visszaállítására szolgál.

Egy csoport tartalmazhat egy képet. Egy csoport egy másik csoportot is tartalmazhat.

Amikor a tételek csoportosítva vannak, akkor stílusuk többé nem módosítható. A csoport tételei továbbra is megjelennek, de azok nem használhatók (dupla kattintással) a csatolt változók értékének módosítására.

Egy csoport a lista elrendezésben mindössze egyetlen sorként jelenik meg.

## **A.18.2 A lista elrendezés**



Ennek a gombnak a megnyomásával bármikor át lehet váltani a grafikus és a listázásos elrendezés között. Ezen kívül a „**Beállítások / Lista Grafikus elrendezés**” parancs is használható erre a célra.

A lista elrendezésben a tételek egy hagyományos lista ablakban jelennek meg. Minden tétel magassága rajzolási stílusának megfelelően van kiszámítva. A lista elrendezésben a képek (bitmapok és metafile) nem láthatók. A lista elrendezésben lehetséges a kiválasztás, amelyet a tétel stílusának beállítására, vagy egy változó

értékének a megváltoztatására kell használni. Ebben a módban többszörös kiválasztás illetve az azt használó parancsok nem állnak rendelkezésre.



A listában levő tételek sorrendje a „**Szerkesztés / Áthelyezés a listában**” parancssal változtatható meg. A listából először ki kell választani az áthelyezni kívánt tételt.

### A.18.3 A tétel stílusának definiálása

Egy létező tétel grafikus stílusa és beállításai a grafikus területen annak szimbólumára duplán rákattintva, vagy pedig, ha a tétel a grafikus vagy lista elrendezésben ki van választva, akkor a „**Szerkesztés / Stílus beállítása**” parancs futtatásával módosítható. A „Stílus” párbeszédablak akkor is kinyílik, amikor a dokumentumhoz egy új tétel van hozzáadva. Ez a következő, a felhasználó által kiválasztandó információkat csoportosítja:

#### **Grafikus stílus és beállítások:**

Egy tétel megjelenítési stílusa (egyszeres szöveg, oszlopdiagram, görbe, stb.) dinamikusan változtatható. Amikor előtér- és háttérszínek vannak használva, akkor azok a megfelelő ablakok használatával tesztre szabhatóak. Amikor a stílus „logikai ikon”, akkor specifikálni kell a megfelelő .ICO fájlok elérési útvonalát. A lemezen létező ikonfájlok böngészéséhez a vezérlők mellett levő „...” gombokat kell használni.

#### **Skála:**

Ez az oszlopdiagramokban és görbékben megjeleníthető maximális érték. Kétpólusú oszlopdiagramoknál és görbékénél mind a negatív, mind a pozitív tengelyhez ugyanaz az abszolút érték van használva.

#### **Változó név**

Ha a „**Név**” mező az aktív mező, akkor a szerkesztő vezérlő mellett levő „...” gomb megnyomásával a felhasználó megkeresheti a projekt szótárban már deklarált változók neveit.

#### **Felirat:**

A grafikus elrendezésben a grafikus tétellel együtt egy feliratot lehet megjeleníteni. A felirat szövegének helye (felül, alul, balra, vagy jobbra) tesztre szabható. A felirat a változó nevének és értékének bármilyen kombinációja lehet, szöveggént formázva. A felirat tesztre szabása nincs hatással a lista elrendezésre.

#### **Parancs változó:**

Ha be van állítva a „Parancs változó” lehetőség, akkor a felhasználó a hibakeresés közben a tétel grafikus szimbólumára történő dupla rákattintással módosíthatja a kapcsolt változó értékét.

## A.18.4 A „Fájl” menü parancsai

A „Fájl” menü azokat a parancsokat tartalmazza, amelyek segítségével a felhasználó a teljes dokumentumot kezelheti.



A „Fájl” menü „Új” parancsa egy új dokumentum szerkesztését kezdi el. Az ISaGRAF nem korlátozza egy projekthez definiált dokumentumok számát. Az új diagram szerkesztése előtt bezáródik az előzőleg nyitva levő diagram. A Fényszóró nem használható több diagram egyszerre történő szerkesztéséhez. Azonban lehetőség van több Fényszóró ablak egyidejűleg történő megnyitására úgy, hogy mindegyik egy másik dokumentum szerkesztését végzi.



A felhasználó a „Fájl / Megnyitás” parancs segítségével bezárhatja a pillanatnyilag szerkesztés alatt álló dokumentumot, és elkezdheti a pillanatnyi projekt egy másik dokumentumának a szerkesztését. Az újonnan kiválasztott dokumentum felváltja a szerkesztő ablakban a pillanatnyilag szerkesztett dokumentumot. Az új dokumentum kiválasztásakor a „Törlés” gomb egy meglévő fájl törlésére használható, a projekt alkönyvtárának „kitakarításához”. Egy diagramban hivatkozott ikon- és bitmap fájlok nem lesznek törölve a diagram törlésekor.



A „Fájl” menü „Kimentés” parancsa lemezre menti a pillanatnyilag szerkesztett dokumentumot. Ha az egy új, cím nélküli dokumentum, akkor a felhasználónak a kimentés előtt azt el kell neveznie. Egy dokumentumot a következő szabályoknak eleget téve kell elnevezni:

- A név hossza nem haladhatja meg a 8 karaktert
- Az első karakternek **betűnek** kell lennie
- A többi karakternek **betűnek**, **számjegynek**, vagy **aláhúzás** karakternek kell lennie
- Az elnevezés nem érzékeny a kisbetű/nagybetű beállításra

A „Fájl” menü „Kimentés más néven” parancsa lehetővé teszi a felhasználó számára, hogy a pillanatnyilag szerkesztett dokumentumot más névvel mentse ki,

## A.18.5 Megjegyzés ISaGRAF V3.2 felhasználók számára

A Fényszóró a V3.0-as vagy V3.2-es verziójú ISaGRAF eszközeivel felépített grafikáit és idődiagram listáit tudja olvasni. Az ilyen fájlok a „**Megnyitás**” párbeszédablakban jelennek meg, származásuk leírásával együtt. A fájlok a Fényszóróval olvashatók és szabadon módosíthatók.

Egy ISaGRAF V3.2 grafika megnyitásakor a dokumentum automatikusan „Reteszelt” megjelölést kap. Kapcsolja ki a „Fájl” menü „Reteszelés” lehetőségét, ha a grafikát meg akarja változtatni.

Ha egy ISaGRAF 3.2 grafika vagy idődiagram lista meg van nyitva, akkor a Fényszóró mindig felajánlja annak natív Fényszóró formátumban történő kimentését. Egy ilyen dokumentum bezárásakor rendszerint megjelenik a „Kimentés más néven” párbeszédablak.

## A.19 Alkalmazások feltöltése

Az ISaGRAF támogatja a célban tárolt alkalmazás feltöltését. A feltöltési eljárás a céllal kommunikál a beépített tömörített forráskód (EZS) betöltésére, majd a betöltött projektnek a workbench környezetben történő visszatöltésére.

A csatlakoztatott célrendszeren futó projekt akkor tölthető fel, ha a cél verziója V3.22, vagy attól későbbi, és ha a tömörített forráskód be lett építve az alkalmazással. A forráskód feltöltéshez történő beépítése egy beállítási lehetőségként választható ki.

### A.19.1 Egy projekt feltöltése:

A „**Feltöltés**” párbeszédablak az ISaGRAF Projektrendező „**Fájlok**” parancsával futtatható. A feltöltés nem egy, a Workbench-en meglevő projektre vonatkozik. A projektrendező listában pillanatnyilag kiválasztott projekt nem áll kapcsolatban a feltöltési mechanizmussal. A célon futó alkalmazás feltöltéséhez a következőket kell tenni:

- 1- ellenőrizze, hogy a cél megfelelően csatlakoztatva van-e
- 2- állítsa be a kommunikációs paramétereket a csatlakozási kapcsolatnak megfelelően
- 3- nyomja meg a „Futás” gombot

A beépített tömörített forrás (EZS) feltöltése és kicsomagolása néhány másodpercet vehet igénybe. A párbeszédablakban megjelenő üzenetek közlik, amikor a feltöltés befejeződött, illetve ha esetleges hiba következett be.

Az ISaGRAF projekt létrehozásakor alkalmazott név a kommunikáció során a célban olvasott név. Ha ez a név már használatban van egy, a workbench-ben létező projekthez, akkor egy üzenet jelenik meg vagy annak felülírására, vagy egy használaton kívüli név kiválasztására. A feltöltés befejezésekor a betöltött projektek projektként történő regisztrálása nem akadályozható meg. A feltöltött projekt ezután készen áll, és megnyitható.

### Lehetséges hibák

Egy projekt feltöltésekor a következő hibák következhetnek be. A hibáról a „Feltöltés” párbeszédablak ad tájékoztatást.

- a céllal nem hozható létre kommunikáció
- a csatlakoztatott cél egy 3.22 előtti verziójú ISaGRAF rendszer
- a célban nem fut alkalmazás
- a célban nem fut alkalmazás

### A.19.2 Kommunikációs beállítások

A „**Beállítás**” gomb megnyomásával a felhasználó az ISaGRAF workbench és a cél ISaGRAF rendszer közötti feltöltési kommunikációhoz használt kapcsolat paramétereit definiálhatja. Ügyelni kell arra, hogy a feltöltés futtatása előtt a konfigurált paraméterek megfeleljenek a csatlakoztatott célnak.

### A.19.3 A projekt előkészítése feltöltéshez

Ha a későbbiekben lehetővé akarja tenni a feltöltést, akkor az ISaGRAF Kódgenerátort értesíteni kell arról, hogy az alkalmazás kóddal be kell építeni a tömörített forráskódot. Ehhez nyomja meg a „**Programfordító beállítások**” párbeszédablak „**Feltöltés**” gombját. Egy másik párbeszédablak választási lehetőségként lehetővé teszi a tömörített forráskód beépítését. Ebben az esetben csak a minimálisan szükséges forrásfájlok lesznek beépítve. Ahhoz, hogy a választható fájlok is be legyenek építve, másik kijelölő négyzeteket kell használni.

**Fontos megjegyzés:** A könyvtárak nem kerülnek letöltésre a beépített forráskóddal. Ez tartalmazza a funkciókat és funkcióblokkokat, valamint I/O kártyákat és berendezéseket is.



#### **Választható fájlok**

A minimálisan szükséges forráskódon kívül a következő fájlok szintén beépíthetők. Ezek választható lehetőségek, mivel ezek kiválasztása a célon többlet memóriaigényt okoz.

*Projekt leíró:* Ha nincs beépítve, akkor a projekt leíró a feltöltés után csak a feltöltés dátumát fogja mutatni.

*Jelszóvédelem:* A feltöltés funkció nincs jelszóvédelemmel ellátva. Ha a feltöltött projektet le akarja védeni, akkor be kell építenie annak jelszóvédelmi rendszerét a forráskóddal.

*Megjegyzések a nem csatlakoztatott I/O csatornákhöz:* Az ISaGRAF lehetőséget nyújt a nem csatlakoztatott I/O csatornákhöz leíró szöveg beírására. Ezt a lehetőséget ne válassza ki olyankor, amikor csak csatlakoztatott I/O-kkal dolgozik.

*A módosítások előzményei:* Ez a projekt globális módosítási előzményeit jelenti.

*Napló fájlok:* Az egyes programok napló fájljai a felhasználó által írt megjegyzéseket, valamint a programfordító programra utaló output üzeneteinek előzményeit tartalmazza. A napló fájlok beépítése sok memóriát foglalhat le a célban.

*Változók és idődiagramok listái:* Ezek a hibakeresés közben létrehozott fájlok, amelyek változónév listákat tartalmaznak listázáshoz vagy idődiagram figyelemmel kíséréséhez.

*Grafikák, ikonok, és bitmapok:* Ide tartoznak az ISaGRAF grafikái, valamint az összes csatolt ikon és bitmap fájl, amennyiben azok a projekt alkönyvtárában helyezkednek el. Figyelmeztetés: A napló fájlok beépítése sok memóriát foglalhat le a célban.

### A.19.4 Hogyan van a tömörített forrás a célon tárolva

A beépített tömörített forrás (EZS) a generált kódban erőforrásokkal van tárolva. A generált erőforrás neve „EZS”. Ha a forráskód beépítés ki lett választva, akkor ezt a

nevet nem lehet másik erőforráshoz választani. A forráskód beépítése nem jelent semmiféle korlátozást az erőforrás definiálására nézve. A felhasználó által írt erőforrás definiáló fájlra nincs hatással a forrás beépítés.

Az erőforrásokkal kapcsolatos további részletek és információk az ISaGRAF dokumentáció Kódgenerátorral kapcsolatos részében található.

### A.19.5 Memóriaigények a célon

A beépített tömörített forrás (EZS) kód a célban az alkalmazás kóddal tárolt többlet memóriát igényel. Általános becslés szerint a minimum EZS (a forrás beépítéshez nincs további lehetőség beállítva) mérete a végrehajtható kód méretének másfélszerese legyen. Ez azt jelenti, hogy az EZS beépítése a letöltött kód méretét 2,5-szeresre fogja növelni.

Bizonyos, szegmentált memória alapú célrendszerekre speciális korlátozások vonatkozhatnak. Mivel az EZS erőforrásként kerül tárolásra a generált kódban, ezért azokat ugyanabban az adatszégmensben kell tárolni, mint az alkalmazás kódot.

### A.19.6 A feltöltött projekttel kapcsolatos tudnivalók

A feltöltött projekt az újrafordításhoz szükséges összes fájlt és adatot tartalmazza. Ez a korábbi programfordítás során kiválasztott beállításoktól függően segédfájlokat, pl. projekt leíró és projekt napló fájlokat is tartalmazhat.

A projekt hibakeresése vagy figyelése előtt azt le kell fordítani (el kell készíteni).

Figyelmeztetés: Mivel az ISaGRAF a programfordítási dátum jellemzőt használja az alkalmazások összehasonlításához, ezért a hibakereső megnyitásakor az az üzenet fog megjelenni, hogy a workbench- és cél alkalmazások eltérő verzióköddel rendelkeznek.

Fontos megjegyzés: A könyvtárak nem kerülnek letöltésre a beépített forráskóddal. A feltöltött alkalmazás újrafordítása előtt ügyelnie kell arra, hogy a megfelelő könyvtár funkciók és funkcióblokkok telepítve legyenek az Ön ISaGRAF workbench-ével.

### A.19.7 Illeszthetőségi kérdések

A feltöltést az ISaGRAF cél és workbench 3.22-es vagy attól későbbi verziója támogatja. A feltöltés támogatására a kommunikációs protokoll bővítve lett.

A 3.03-tól 3.21-ig terjedő verziójú ISaGRAF rendszereken alapuló célokban nincs bekorlátozva a tömörített forráskód (EZS) beépítése. A beépített információ azonban ebben az esetben nem tölthető fel, mivel az ilyen cél nem támogatja a szükséges kommunikációs szolgáltatásokat.

## A.20 A Diagnózis eszköz használata

A „**Diagnózis Eszköz**” az ISaGRAF hibakereső eszköz része. Ez lehetővé teszi a felhasználó számára, hogy a folyamat vizsgálata és vezérlése céljából egy előre meghatározott változókészlettel dolgozzon. Az ISaGRAF hibakereső eszköz egy nagyteljesítményű eszköz, amely magas szintű funkciókat tartalmaz. A Diagnózis eszköz egy biztonságos lehetőség a cél alkalmazás végső futási műveletekhez vagy karbantartási célokhoz történő vezérléséhez. Az ISaGRAF Diagnózis eszköz közvetlenül a Programrendezőben levő ISaGRAF csoportból van futtatva, a következő ikonnal:



A létező projektek listája egy párbeszédablakban jelenik meg. Ez lehetővé teszi a korlátozott ISaGRAF hibakereső futtatását egy létező, már letöltött ISaGRAF alkalmazáson. Az „**OK**” gomb megnyomása elindítja a korlátozott hibakeresőt a kiválasztott projekten. Nyomja meg a „**Mégsem**” gombot a párbeszédablakból történő kilépéshez. A „**Beállítás**” parancs az ISaGRAF Workbench és a cél PLC közötti kommunikációs kapcsolat beállítására szolgál. Az ezzel a paranccsal kapcsolatos további magyarázatok ennek a dokumentumnak „**Programok kezelése**” című fejezetében találhatók.

**Megjegyzés:** Az ISaGRAF Diagnózis Eszköz (korlátozott hibakereső) nem használható a cél PLC-ben futó alkalmazás letöltésére, megállítására, vagy frissítésére. Nem hajtható végre semmilyen művelet, ha a Diagnózis eszköz ablakban kiválasztott projekt nem ugyanaz, mint ami a PLC-n telepítve és futtatva van.

A korlátozott ISaGRAF futtatásakor, ha az megfelelően csatlakoztatva van a cél alkalmazáshoz, a következő parancsok állnak rendelkezésre:

- Változólisták kémlélése
- Grafikus dokumentumok kémlélése a Fényszórával

## A.21 Az ISaGRAF szimulátor használata




Az ISaGRAF Kernel szimulátor a hibakeresővel indul el, amikor a Programkezelés ablak „Hibakeresés” menüjének „Szimuláció” parancsa futtatva van. A kernel szimulátor egy komplett ISaGRAF célrendszer, amely támogatja az ISaGRAF valamennyi standard jellemzőjét és a ICS Triplex ISaGRAF által szállított standard könyvtár valamennyi „C” funkcióját valamint funkcióblokkját. Az I/O kártyák egy ablakban grafikusan vannak szimulálva. Bármilyen típusú I/O kártya szimulálható. Az I/O csatlakozás során „Virtuális kártyákként” definiált kártyák ugyancsak a szimulációs ablakban jelennek meg.




### A.21.1 Kapcsolatok a hibakeresővel

A kernel szimulátor teljes kommunikációt támogat az ISaGRAF hibakeresővel, így a szimuláció közben bármelyik hibakeresési lehetőség használható. A kernel szimulátor mindig a pillanatnyi ISaGRAF alkalmazáson működik. Szimuláció közben az „Indítás”, „Megállítás”, „Letöltés”, vagy „Feltöltés” hibakereső parancsok nem állnak rendelkezésre. A szimulátor nem használható, ha a cél kód felépítése előtt a programfordító beállításokban nem lett kiválasztva a „SZIMULÁCIÓ” cél. A szimulátor ablak bezárásakor a hibakereső ablaka (és a hibakeresés során megnyitott valamennyi ISaGRAF ablak) szintén bezáródik.

### A.21.2 I/O szimuláció

Az I/O kártyák a szimulátor ablakban jelennek meg, nevük és kártyahely számuk szerint. Minden standard ISaGRAF I/O típus (logikai, analóg, vagy üzenet) kezelve van. Az input kártyák csatornáit speciális gombokkal és mezőkkel vannak megjelenítve. Az output kártyák csatornáit grafikus állapotjelző lámpákkal és adatmezőkkel vannak megjelenítve.

-  **Logikai inputok:** Egy logikai inputot egy négyzet alakú zöld gomb jelképez. A csatorna száma az I/O gombbal együtt van feltüntetve. Az input érték a gomb megnyomásakor IGAZ. A gombra való rákattintás megváltoztatja a megfelelő I/O értéket. Az egér jobb oldali gombja az input beállítására szolgál csak olyankor, amikor a gomb meg van nyomva.
-  **Logikai outputok:** A logikai outputot egy kis kör képviseli. A csatorna száma az I/O-val együtt van feltüntetve. Az output érték a grafikus szimbólum kiemelésekor IGAZ.
-  **Analóg inputok:** Egy analóg input csatorna egy egyszerű számmező, ahol a megfelelő input értékét lehet bevinni. A keretre kattintva megjelenik a beszűrő jel. Ekkor beírható a csatorna új értéke. A beírás után nincs szükség az **ENTER** gomb megnyomására. Az analóg inputok akár decimális, akár hexadecimális formában beírhatók. A pillanatnyi érték a fel/le gombokkal növelhető vagy csökkenthető.

-  Analóg outputok: Egy analóg output csatorna egy numerikus output mező. Az output érték akár decimális, akár hexadecimális számként megjeleníthető. A felhasználó nem végezhet műveletet egy output csatornán.
-  Üzenet inputok: Egy üzenet input csatorna egy egyszerű szöveg mező, ahol a megfelelő input értékét lehet bevinni. A keretre kattintva megjelenik egy hiányjel. Ekkor beírható a csatorna új értéke. A beírás után nincs szükség az **ENTER** gomb megnyomására.
-  Üzenet outputok: Egy analóg output csatorna egy numerikus output mező. A felhasználó nem végezhet műveletet egy output csatornán.

### A.21.3 Könyvtár komponensek

Az ISaGRAF szimulátor teljes mértékben támogatja a ICS Triplex ISaGRAF által szállított standard konverziókat, funkciókat, és funkcióblokkokat. Az alábbiakban a támogatott tárgyak listája látható:

#### **Konverziós funkciók:**

bcd, scale

#### **Funkciók:**

abs, acos, ArCreate, ArRead, ArWrite, ascii, asin, atan, char, cos, delete, expt, find, insert, left, limit, log, max, mid, min, mlen, mod, mux4, mux8, odd, rand, replace, right, rol, ror, sel, shl, shr, sin, sqrt, tan, trunc

#### **Funkcióblokkok:**

average, blink, cmp, ctd, ctu, ctud, derivate, f\_trig, hyster, integral, lim\_alm, r\_trig, rs, sema, sr, stackint, tof, ton, tp

A felhasználó által definiált konverziók, „C” funkciók és funkcióblokkok általában nincsenek integrálva az ISaGRAF Szimulátorral. Az ilyen tárgyak általában a célrendszer szoftver és hardver erőforrásainak a használatára vannak tervezve. Az ilyen erőforrások általában nem állnak rendelkezésre a Windows rendszeren. Az ISaGRAF Szimulátor a következő standard viselkedést nyújtja a felhasználó által definiált konverziókhoz, funkciókhoz, vagy funkcióblokkokhoz:

- Amikor a szimulátor egy új konverziót dolgoz fel, akkor az egy „nulla” konverzióval cserélődik le. Ez azt jelenti, hogy az analóg változók fizikai értéke mindig megegyezik az elektromos értékkel (ahogy az a Szimulátor panelen be van adva illetve meg van jelenítve).
- Amikor a szimulátor egy új „C” funkciót vagy funkcióblokkot futtat, az semmilyen műveletet nem dolgoz fel. Az eredmény érték nem lesz beállítva.

### A.21.4 Beállítási lehetőségek

A felhasználó a „**Beállítások**” menü parancsai segítségével vezérelheti az I/O-k megjelenítését a szimulátor panelen. A felhasználó a hibakeresés során bármikor beállíthatja vagy eltávolíthatja ezeket a lehetőségeket.

- ☐ Amikor be van állítva a „**Színes megjelenítés**”, akkor az I/O csatornák színes bitmapokként vannak megjelenítve. Ha egyes LCD képernyőkön nem lehet a színeket megkülönböztetni egymástól, akkor a felhasználónak el kell távolítania ezt a beállítást, hogy tisztán fekete-fehér input és output grafikák legyenek megjelenítve az I/O csatornákhöz.
- ☐ Amikor be van állítva a „**Változó nevek**” lehetőség, akkor minden I/O csatorna mellett megjelenik egy felirat a csatlakoztatott I/O változó nevével. Ennek a beállításnak az eltávolításával a felhasználó csökkentheti a szimulátor panel méretét.
- ☐ Amikor a „**Hexadecimális értékek**” lehetőség be van állítva, akkor minden input vagy output analóg csatorna hexadecimális formátumban van beírva vagy megjelenítve.
- ☐ Amikor a „**Mindig felül**” lehetőség be van állítva, akkor a szimulátor ablak mindig látható lesz, még akkor is, ha az input egy másik ablakra összpontosul.

### A.21.5 Input állapotok kimentése és visszatöltése

Az ISaGRAF szimulátor használatával az input csatornák manuális műveleteken vannak átkényszerítve, a szimuláció panel kapcsoló gombjaira és szerkesztő vezérléseire hatva. Az „**Eszközők**” menü következő parancsainak használatával az összes input csatorna állapota bármikor kimenthető illetve visszatölthető:

#### Input séma betöltése

Az input csatornák értékeit egy, az „Input séma kimentése” parancssal a lemezen létrehozott fájlban tárolt értékekre állítja be.

#### Input séma kimentése

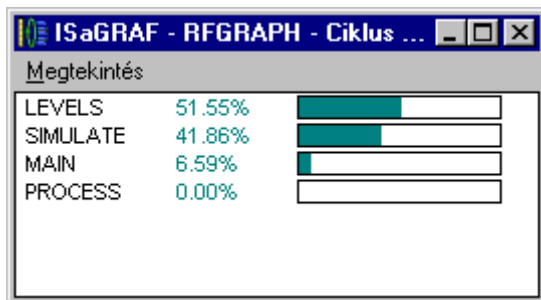
Az input csatornákat egy fájlba menti ki, hogy azok az „Input séma betöltése” parancssal később visszatölthetők legyenek. A fájl a projekt alkönyvtárában van eltárolva, így azt az ISaGRAF archiváló segédprogram a többi projekt fájllal együtt elmenti.

Megjegyzés: A lemezre csak névvel ellátott input csatornák (amelyekhez változó van csatlakoztatva) kerülnek kimentésre.

### A.21.6 A ciklus profilozó

Az ISaGRAF Ciklus profilozó egy hatékony diagnosztikai eszköz, amely azt mutatja, hogy a ciklusidő hogyan oszlik meg egy alkalmazás különböző programjai, funkciói, és funkcióblokkjai között. Ez az eszköz nagyon hasznos az alkalmazás működésének gyors diagnosztizálásához, és a programozó számára megmutatja a kód optimalizálásra szoruló részeit.

A Ciklus profilozó az ISaGRAF Szimulátor ablak menüinek „**Eszközők / Ciklus profilozó**” parancsával futtatható. Ez mindegyik programhoz, funkcióhoz, vagy funkcióblokkhoz megjeleníti a ciklusidő annak végrehajtására fordított százalékát:



Amikor a „**Nézet / Átlag**” lehetőség be van állítva, akkor a megjelenített információ az alkalmazás elindulása óta, vagy a „**Nézet / Visszaállítás**” parancs legutóbbi futtatása óta számított százalékos átlag lesz.

Ha a „**Nézet / Átlag**” lehetőség nincs beállítva, akkor a megjelenített információk az utolsó ciklus végrehajtása óta érvényes mértékeket mutatja. Ez a lehetőség akkor is használható, amikor az alkalmazás „**Ciklustól ciklusig**” módban van, az alkalmazás tartalmától függő mérési készletek eléréséhez.

A „**Nézet / Másolás**” parancs a program neveknek és százalékoknak a Windows Vágólapjára történő másolására szolgál, ASCII formátumban. Ezután az adatok szöveges dokumentumokba vagy általános táblázatkezelőkbe beilleszthető.

### Fontos megjegyzések

Ezek nem pontos mértékek. A százalékos számítás a TIC utasítás számolásán alapul, figyelembe véve a különböző utasítás végrehajtási időket. A számítások nem tartalmazzák a „C” funkciókban és funkcióblokkokban eltöltött időt.

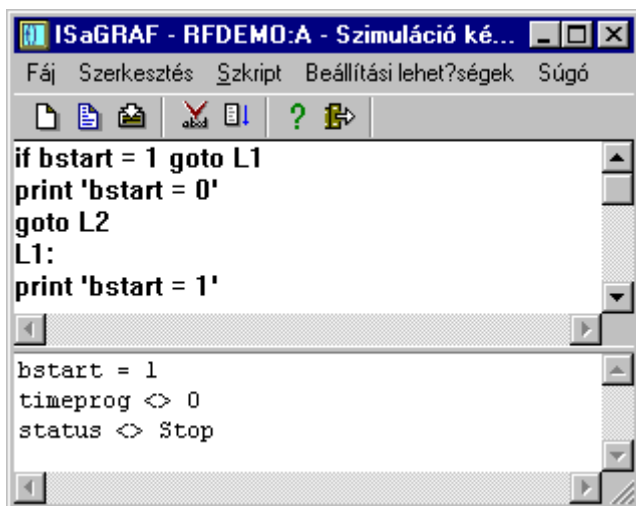
A funkciókhoz vagy funkcióblokkokhoz feltüntetett idő az alkalmazás programjaitól ugyanabban a ciklusban érkező „meghívási idők” összege.

Az idő számítás a TIC kódon alapul, és nem nyújt megbízható információt, ha a tényleges alkalmazás kód „C” nyelvben van generálva és „C” programfordítóval lett felépítve.

## A.21.7 Szimuláció szkriptek

Az ISaGRAF szimulátor tartalmaz egy eszközt szimuláció szkriptek építésére és futtatására. A szkript bármelyik egyszerű ST-hez hasonló szöveges nyelven leírható, és tesztek automatizálására használatos az ISaGRAF szimulátorral.

A szimulációs szkript szerkesztő a Szimulátor ablak „**Eszközök / Szimuláció szkriptek**” parancsával futtatható. Az alábbiakban a szkript szerkesztő kerete látható:



A felső ablak a szövegszerkesztő, ahol a szkript utasításai vannak beírva. Ezt ugyanúgy kell használni, mint más ISaGRAF szövegszerkesztőket, és olyan kiemelt lehetőségeket biztosít, mint pl. egy változó szimbólum egérrel történő kiválasztása. A „Beállítások” menü parancsaival beállíthatók a tab szélességek illetve a karakter font kiválasztása.

Az alsó ablak a szkript futtatásakor kapott üzenet outputokat mutatja. Az ablakok közötti elválasztó vonal szabadon vonszolható az ablakok átméretezéséhez. Az output ablak a szkript szerkesztése közben elrejtethető, de a szkript futtatásakor minden alkalommal automatikusan megjelenik.

## Szkriptek szerkesztése

A szkript fájlok kezelésére a „Fájl” menü parancsai szolgálnak:

**Új**.....egy új, cím nélküli szkriptet hoz létre

**Megnyitás** .....fájlról betölt egy meglevő szkriptet

**Kimentés**.....a szkript szövegét és az output ablak tartalmát kimenti a projekt alkönyvtárba

**Kimentés más néven** a szkriptet más néven menti ki

Az ISaGRAF projekt alkönyvtárban minden szkripthez két fájl van létrehozva:

<szkriptnév>.SCC ..... a szkript szövege (utasítások)

<szkriptnév>.SCO ..... az output ablak tartalma

Ahol <szkriptnév> a szkript neve. Mindkét fájl standard szöveg fájl, amelyeket bármilyen szövegszerkesztővel meg lehet nyitni.



Egy szkript szerkesztésekor a „**Szerkesztés / Szimbólum beillesztése**” parancs lehetővé teszi egy deklarált változó kiválasztását és beillesztését a hiányjel helyére.



## Szkriptek futtatása

A szkripteket futtatás előtt ellenőrizni és fordítani kell. Ha szükséges, akkor a szintaxis ellenőrzése automatikusan megtörténik a „Futás” parancs végrehajtásakor. A „**Szkript**” menüről a következő parancsokat kell használni:



**Ellenőrzés** .....ellenőrzi a szintaxist és lefordítja a szkriptet

**Szkript futtatása** .....elkezdí a pillanatnyilag szerkesztett szkript végrehajtását

Egy új, cím nélküli szkript esetében azt ellenőrzése előtt ki kell menteni (és nevet kell neki adni). Egy elnevezett szkript esetében az automatikusan kimentődik a szintaxis ellenőrzés előtt.

A szkript futása közben annak tartalmát nem lehet megváltoztatni. A szkript végének elérését egy üzenet jelzi. Egy futó szkript a „**Szkript**” menü következő parancsával szintén félbeszakítható:



**Szkript félbeszakítása** .....félbeszakítja a futó szkriptet

A szkript végrehajtása a ciklusok között történik. A ciklusba programozott végtelen hurok esetében az ISaGRAF szimulátor biztosítja, hogy ez a hurok mindig meg legyen szakítva azért, hogy az ISaGRAF ciklusok mégis végre legyenek hajtva és az egyéb ISaGRAF alkalmazások ne legyenek blokkolva. Az ISaGRAF szkript fordító dönt a szkript végrehajtásának megszakításáról, ha ugyanabban a ciklusban ugyanaz a „címké” egynél többször előfordul. A szkript végrehajtást normál módon a „Ciklus” vagy „Várakozás” utasítások is megszakíthatják.



## Szkript leíró nyelv

A szkript leíró nyelv egy nagyon egyszerű szöveges nyelv, amely hasonló az ST-hez, de minden utasítás külön sorban van beírva, és azt nem kell pontosvesszővel lezárni. A rendelkezésre álló utasítások megismeréséhez és egy kulcsszónak a hiányjel helyére történő beillesztéséhez az eszközsor következő gombját kell használni:



utasítás beillesztése (kulcsszó és súgó megjegyzésekként)

Különböző utasítás típusok léteznek: Az első egy változó kijelölése (kényszerítése):  
:= .....kijelölés

Más utasítások lehetővé teszik üzenetek megjelenítését az output ablakban:

**Nyomatás** " .....egy szöveges karaktersort vagy egy változó értéket jelenít meg

**Idő nyomtatás** .....a pillanatnyi idő jellemzőt adja ki

Egyéb utasítások a szkript utasítások és az ISaGRAF ciklus szinkronizálására szolgálnak:

**Ciklus** .....az ISaGRAF szimulátorral végrehajtat egy ciklust

**Várakozás** .....várakozik egy meghatározott ideig

Egyéb utasítások a szkript áramlásának vezérlésére szolgálnak:

**Címkék** .....a szkriptben bárhová elhelyezhetők

**Ugrás** .....feltétel nélküli ugrás egy címkére

**Ha, akkor ugrás ...-ra** feltétel nélküli ugrás egy címkére

**End** .....befejezi a szkriptet

A szkript nyelv nem veszi figyelembe a kisbetű-nagybetű beállításokat. Bármelyik szövegsor végére beilleszthetők megjegyzések. A megjegyzések írhatók az ST szabályainak megfelelően („\*” és „”) karakterek között), vagy azokat egy „” karakter előzheti meg.

## ":=" Kijelölés

**Jelentése:** Egy ISaGRAF változó értékét kényszeríti. Ez lehet egy belső változó, egy input csatorna, vagy egy output csatorna.

**Szintaxis:**

```
<varname> := <constant_expression>
<varname> = <constant_expression>
```

**Argumentumok:** a <varname> egy deklarált alkalmazási változó érvényes szimbóluma, vagy egy közvetlen képviselő I/O változó, amely „%” írási konvenciót alkalmaz.

a <constant\_expression> egy érvényes konstans kifejezés, amely megfelel a specifikált változó típusának. Logikaiaknál a „HAMIS” és „IGAZ” helyett „0” és „1” is használható. Időzítőknél a „T#” vagy „TIME#” előtag elhagyható.

**Megjegyzések:** Egy szkripttel kényszerített input változót nem kell reteszelni. A megfelelő input csatorna rajza felújításra kerül, amikor egy szkript kényszerít egy input változót.

**Figyelmeztetés:** ne kényszerítsen egy konverzióhoz csatolt input vagy output analóg változót, mert a szkript végrehajtása nem támogatja a konverziós funkciókat vagy táblázatokat.

**Példa:**

```
MyBooVar := 1 (* ugyanaz, mint IGAZ *)
MyIntVar := 1234
MyRealVar := 1.2345
MyMsgVar := 'Hello'
MyTmrVar := t#12s
```

## Nyomatás

**Jelentése:** Egy karaktersort vagy egy változó értékét ír az output ablakba. A szöveg egy új sorként kerül megjelenítésre az output ablakban már megjelenített szöveg végén.

**Szintaxis:**

```
Print '<text>'
Print <varname>
```

**Argumentumok:** a <text> bármilyen szöveg, egyszeres idézőjelek közé téve

a *<varname>* egy deklarált alkalmazási változó érvényes szimbóluma, vagy egy közvetlen képviselő I/O változó, amely „%” írási konvenciót alkalmaz.

Megjegyzések: A változó értékek megjelenítése mindig az IEC konvencióknak megfelelően van formattálva.

Példa:       Print 'Hello'  
              Print MyBooVar

Output:       Hello  
              MyBoovar = IGAZ

## IdőNyomtatás

*Jelentése:* A pillanatnyi időjelzőt megjeleníti az output ablakban. A szöveg egy új sorként kerül megjelenítésre az output ablakban már megjelenített szöveg végén.

*Szintaxis:*   **PrintTime**

Megjegyzések: Az időjelző formátuma a Windows rendszer pillanatnyi beállításainak felel meg.

Példa:       Print 'A pillanatnyi idő:'  
              PrintTime

Output:       A pillanatnyi idő:  
              15:45:22

## Ciklus

*Jelentése:* Felfüggeszti a szkript végrehajtását a következő ISaGRAF ciklus elvégzéséig.

*Szintaxis:*   **Cycle**

Megjegyzések: A szkript utasítások az ISaGRAF ciklus elején kerülnek végrehajtásra. Ha a szimulátor „Ciklustól ciklusig” módban van, akkor a „ciklus” utasítást azonnal egy ciklus követi. A szkript következő utasításai lesznek végrehajtva a hibakereső következő „Egy ciklus végrehajtása” parancsára.

Példa:       (\* az ISaGRAF program A-t B-be másolja \*)  
              A := 0  
              Cycle  
              Print B  
              A := 1

```
Print B (* nincs végrehajtva ciklus / B nincs 1-re
állítva *)
Cycle
Print B
```

Output: B = 0  
B = 0  
B = 1

## Várakozás

**Jelentése:** Felfüggeszti a szkript végrehajtását, amíg egy késleltetés le nem telik.

**Szintaxis:** `Wait <delay>`

Argumentumok: a `<delay>` az időállandó kifejezésekre vonatkozó konvenciók szerint kifejezett késleltetés. A „T#” vagy „TIME#” előtag elhagyható. A késleltetés értékének 10 ezredmásodperc és 1 óra között kell lennie.

Megjegyzések: A „Várakozás” utasítás nem pontos, mivel a gazda Windows rendszertől függ. A késleltetést emellett plusz vagy mínusz egy ISaGRAF ciklus pontosságúnak kell tekinteni. Amikor a „Várakozás” utasítás sorra kerül, az ISaGRAF ciklusok a késleltetés leteltéig és a szkript végrehajtásának folytatásáig kerülnek végrehajtásra.

Példa: `PrintTime`  
`Wait 2s`  
`PrintTime`

Output: 15:45:27  
15:45:29

## Címkék

**Jelentése:** Címkék a szkriptben bárhová elhelyezhetők. Ezek „Ugrás” utasítások rendeltetési helyeként használatosak, és lehetővé teszik az áramlás vezérlést a szkript utasításokhoz.

**Szintaxis:** `<labelname>:`

Argumentumok: a `<labelname>` egy egyedi név, az ISaGRAF változó elnevezési konvencióknak megfelelően: 16 karakterre korlátozódik, egy betűvel kezdődik, amit betűk, számjegyek, vagy aláhúzás karakterek követnek. A definiált címkét egy „:” karakternek kell követnie.

**Megjegyzések:** Nem szabad utasítást elhelyezni arra a sorra, ahol egy címke van definiálva. A címke neve nem lehet ugyanaz, mint egy deklarált ISaGRAF változó szimbólum.

**Példa:**

```
(* egy végtelen hurokkal rendelkező szkript példája *)
loop:
PrintTime
Wait 1s
Goto loop
```

## Ugrás

**Jelentése:** Feltétel nélküli ugrás egy címkére

**Szintaxis:** `Goto <labelname>`

**Argumentumok:** a `<labelname>` a szkriptben definiált címke neve.

**Megjegyzések:** Meg vannak engedve visszafelé történő ugrások is. Egy végtelen hurok esetében a szkript végrehajtása automatikusan meg van szakítva mindegyik hurkon azért, hogy az ISaGRAF ciklusok végrehajtása meg legyen őrizve.

**Példa:**

```
Print 'Before Jump'
Goto MyLabel
Print 'Within Jump' (*soha nincs elvégezve *)
MyLabel:
Print 'After Jump'
```

**Output:**

```
Before Jump
After Jump
```

## Ha Ugrás

**Jelentése:** Feltétel nélküli ugrás egy címkére. A feltétel vagy egy összehasonlítás két ISaGRAF változó között, vagy egy összehasonlítás egy változó és egy konstans kifejezés között.

**Szintaxis:**

```
If <var1> test <var2> Goto <labelname>
If <var1> test <constant_expr> Goto <labelname>
```

A rendelkezésre álló összehasonlítási **tesztek**:

```
=      igaz, ha mindkét tagnak ugyanaz az értéke
<>    igaz, ha a tagoknak különböző értékeik vannak
<      igaz, ha az első tag kisebb mint a második tag
<=    igaz, ha az első tag kisebb vagy egyenlő mint a
      második tag
```

> igaz, ha az első tag kisebb mint a második tag  
 >= igaz, ha az első tag kisebb mint a második tag

Argumentumok: a <var1> <var2> a deklarált alkalmazás változók érvényes szimbólumai, vagy közvetlen képviselő I/O változók, amelyek „%” írási konvenciót alkalmaznak.

a <constant\_expr> egy érvényes konstans kifejezés, amely megfelel a specifikált változó típusának. Logikaiaknál a „HAMIS” és „IGAZ” helyett „0” és „1” is használható. Időzítőknél a „T#” vagy „TIME#” előtag elhagyható.

a <labelname> a szkriptben definiált címke neve.

Megjegyzések: Meg vannak engedve visszafelé történő ugrások is. Egy végtelen hurok esetében a szkript végrehajtása automatikusan meg van szakítva mindegyik hurkon azért, hogy az ISaGRAF ciklusok végrehajtása meg legyen őrizve.

Példa: (\* Ez a szkript hurokba lép mindaddig, amíg a MyVar IGaz nem lesz \*)  
 Loop:  
 If MyVar = TRUE Goto TheEnd  
 Print MyVar  
 Goto Loop  
 TheEnd:

## Vége

**Jelentése:** Befejezi a szkriptet

**Szintaxis:** **End**

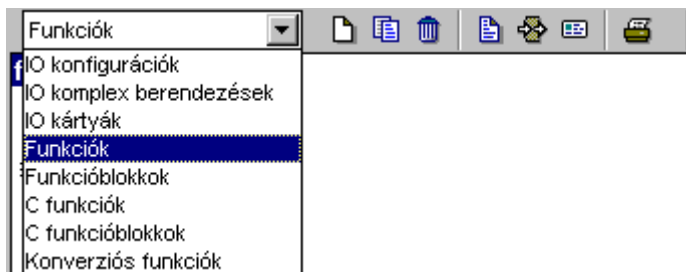
Megjegyzések: A szkript utolsó sorára nem kötelező „Befejezés” utasítást tenni.

Példa: (\* Ez a szkript hurokba lép mindaddig, amíg a MyVar IGaz nem lesz \*)  
 Loop:  
 If MyVar = FALSE Goto Continue  
 End  
 Continue:  
 Print MyVar  
 Goto Loop

## A.22 A Könyvtárrendező használata

Az ISaGRAF könyvtárak egy standard interfészt biztosítanak az automatizálás fejlesztés és az ISaGRAF célrendszer szoftver és hardver lehetőségei között. Minden interfész típushoz egy könyvtár létezik. Az ISaGRAF Workbench Könyvtárrendező a hardver beszállítójához vagy a szoftvermérnökhöz van rendelve. Ez a személy a könyvtárrendezőt az által létrehozott tárgyak ISaGRAF programozási interfészének leírására használja.

Az ISaGRAF Workbench Könyvtárrendező az ISaGRAF könyvtárak egyikének elemeit mutatja. Az ablak bal oldali területén a kiválasztott könyvtár **elemeinek listája** látható. A jobb oldalon az elemek listájából jelenleg kiválasztott elem **műszaki megjegyzése** (kezelési útmutató) látható. A könyvtárrendező menüi az aktív könyvtár elemeinek létrehozására, definiálására, vagy módosítására szolgáló parancsokat tartalmazzák. A „**Fájl / Másik könyvtár**” parancs az ISaGRAF könyvtárak egyikének kiválasztását teszi lehetővé. Az eszközsortól balra levő összetett ablak ugyancsak használható egy könyvtár kiválasztására:



### A.22.1 Könyvtárelemek kezelése

Az elemek létrehozására, valamint a nyitott könyvtárban levő elemeken való munkavégzésre a „**Fájl**” menü parancsai használhatók.



#### **Egy új elem létrehozása**

A „**Fájl**” menü „**Új**” parancsa egy új elemet hoz létre a kiválasztott könyvtárban. Az új elem az alábbi szabályok szerint kerül beírásra:

- egy név maximális hossza **8** karakter
- az első karakternek **betűnek** kell lennie
- a többi karakternek **betűnek, számjegynek**, vagy („\_”) karakternek kell lennie
- egy könyvtárelem elnevezése nem érzékeny a kisbetű-nagybetű beállításra.

Minden könyvtárelemhez egy szöveges megjegyzés kapcsolódik. Ez a megjegyzés az elem létrehozásakor van beírva. Egy új elem létrehozásakor a következőket kell beírni:

- annak definíciója egy I/O konfigurációkhoz,
- annak paraméterei egy I/O kártyához,

- annak felhasználói interfésze egy funkcióhoz vagy funkcióblokkhoz.

Egy „C” konverzió, „C” funkció, vagy „C” funkcióblokk létrehozásakor annak forráskódjának teljes kerete automatikusan létrehozódik.



### **Meglevő elemekkel történő munkavégzés**

A felhasználó a „**Fájl / Átnevezés**” paranccsal az elemek listájáról kiválasztott elem nevét vagy megjegyzését változtathatja meg. A felhasználó a „**Fájl / Másolás**” paranccsal az aktív könyvtárban kiemelt elemet ugyanabban a könyvtárban egy másik elembe másolhatja. Ha a rendeltetési elem már létezik, akkor annak teljes tartalma felülíródik. Ha a rendeltetési elem nem létezik, akkor az automatikusan létrehozásra kerül. A „**Fájl / Törlés**” parancs az aktív könyvtárból eltávolítja a pillanatnyilag kiválasztott elemet. Az „**Átnevezés**”, „**Másolás**”, vagy „**Törlés**” parancsok a következő elem komponensekre hatnak:

- műszaki megjegyzés
- egy I/O konfiguráció komplett definíciója
- paraméterek egy I/O kártyához, vagy komplex berendezéshez
- interfész definíció egy funkcióhoz vagy funkcióblokkhoz
- forráskód IEC nyelven írt funkcióhoz vagy funkcióblokkhoz
- forráskód egy C konverzióhoz, funkcióhoz, vagy funkcióblokkhoz



ha az elem egy „C” konverzió, „C” funkció, vagy „C” funkcióblokk, akkor annak neve egy „**Átnevezés**” vagy „**Másolás**” parancsra nem lesz automatikusan frissítve a csatolt forráskódban.



ha az elem egy IEC nyelven írt funkció, akkor az „**Átnevezés**” vagy „**Másolás**” parancs nem változtatja meg a visszatérési paraméter nevét.



### **Jelszóvédelem beállítása**

A felhasználó a „**Fájl / Jelszó beállítása**” parancs segítségével a nyitott könyvtárban levő kiválasztott elemhez egy jelszóvédelmet definiálhat. A jelszó szintekkel és adatvédelemmel kapcsolatos további információk ennek a kézikönyvnek a végén, a „**Jelszóvédelem**” című fejezetben található. A jelszavak csak a kiválasztott elemre vonatkoznak. Ezek nincsenek hatással az ISaGRAF könyvtárak más elemeire.



### **Funkciók és funkcióblokkok fordítása**

Amikor IEC nyelvben írt funkciók vagy funkcióblokkok könyvtára van kiválasztva, akkor a „**Fájl**” menü „**Megerősítés (fordítás)**” parancsa használható a kiválasztott elem szintaxisának ellenőrzésére és tárgykódjának létrehozására. Az IEC nyelveken írt funkciókat és funkcióblokkokat hibák nélkül le kell fordítani, mielőtt az ISaGRAF projektekben használni lehetne őket. Ez a parancs nincs hatással, ha másik könyvtár van kiválasztva.



### **Műszaki megjegyzések**

A felhasználó a „**Fájl / Műszaki megjegyzés**” parancs segítségével az aktív könyvtárban kiválasztott elemhez műszaki megjegyzést írhat be. A műszaki megjegyzés bevitele az ISaGRAF szövegszerkesztővel történik. Egy elem műszaki megjegyzése annak **kezelési útmutatója**. Az elem felhasználója ezt egy ISaGRAF

projektbe történő integrálás során olvassa el. Az elem használatáról szóló műszaki megjegyzésnek tartalmaznia kell annak fő funkcióját, programozási interfészének és paramétereinek részletes magyarázatát, valamint környezetét és korlátait.

A felhasználó az **„Eszközök / Standard megjegyzés formátum”** parancs segítségével egy standard szöveg formátumot definiálhat a pillanatnyilag kiválasztott könyvtárban levő valamennyi elemhez. Egy új elem műszaki megjegyzésének szerkesztésekor ez a formátum lesz elsődlegesként használva. Ennek segítségével a felhasználó optimalizálhatja a műszaki megjegyzés szerkesztését.



## Paraméterek

Egy elem paraméterei az elem által nyújtott számítógép műveletek és az elemnek egy ISaGRAF alkalmazásban történő felhasználása közötti **interfészt** határozza meg. A paraméterek jelentése eltér az egyes típusú könyvtár elemekhez.

Egy I/O konfiguráció paraméterei a konfiguráció komplett I/O kártya készletét, valamint az I/O csatornákhöz használt alapértelmezett változó neveket definiálják. Egy I/O kártya vagy egy komplex berendezés paraméterei a kártya fizikai és logikai konfigurációját definiálják. Egy funkció vagy funkcióblokk paraméterei az elem interfészét definiálják, az ST nyelv funkció hívási konvencióinak megfelelően. Egy konverziós funkciónak nincsenek paraméterei, mivel az egy standard, előre definiált interfészt használ.



## Forráskód

Az ISaGRAF Workbench lehetővé teszi a programozó számára egy könyvtár konverzió, funkció, vagy funkcióblokk forráskódjának kezelését. Egy IEC nyelvben írt funkció vagy blokk forráskódja egy, a funkcióhoz csatolt nyelvvel leírt szöveg illetve diagram. A „C” komponensek („C” funkciók, „C” funkcióblokkok, és konverziós funkciók) forráskódja két külön fájlra osztozik: egy **forrás fejléc**, amely az **interfész** pontos definícióját tartalmazza az elem paraméter definíciójának megfelelően, és egy **forráskód** fájl, amely az elem műveletének használatát tartalmazza.

Az ISaGRAF workbench a forráskód fájlt egy új könyvtár elem létrehozásakor generálja. Ugyanakkor létrehozza és frissíti a forrás fejléct is, a paraméterek definícióinak megfelelően. A programozó az ISaGRAF szövegszerkesztő használatával egészítheti ki a forráskód fájlt.



## Könyvtárelemek archiválása

Az **„Eszközök / Archiválás”** menü parancs az ISaGRAF archívumkezelőt futtatja könyvtárelemek kimentésére, illetve visszatöltésére. Az **„Archiválási parancs”** kiadása előtt először ki kell választania egy könyvtárt. Az archívumkezelő egyszerre csak egy könyvtárhoz mutatja az elemek listáját.

### A.22.2 I/O konfiguráció

Az ISaGRAF I/O konfigurációs könyvtár egyszerű módszert biztosít az új ISaGRAF projektek inicializálására, előre definiált I/O konfigurációval. Egy I/O konfiguráció a következőket definiálja:

- I/O kártyák készlete
- I/O kártyák paramétereinek alapértelmezett értékei

- I/O csatornák alapértelmezett nevei

Amikor egy új ISaGRAF projekt van létrehozva egy könyvtár I/O konfigurációval, akkor a megfelelő I/O csatlakozás automatikusan beállítódik, és a csatorna neveknek megfelelő I/O változók automatikusan deklarálásra kerülnek a projekt szótárban.



Egy I/O konfiguráció definíciója az ISaGRAF I/O Csatlakozási eszközzel történik (ugyanaz az eszköz van használva egy projekten belül). Ennek az eszköznek a használatával kapcsolatos további információk ennek a kézikönyvnek az „I/O Csatlakozás” című fejezetében található. Egy új I/O kártyának a konfigurációba illesztésekor az új kártya valamennyi csatornája standard alapértelmezett nevekkal kerül deklarálásra. Egy I/O csatorna standard alapértelmezett nevének formátuma a következő:

**<irány><típus><kártyahely\_száma>\_<csatorna\_száma>**

Az első karakter az I/O csatorna irányát mutatja:

"I" .....input csatorna  
 "Q" .....output csatorna

A második karakter az I/O csatorna típusát mutatja:

"X" .....logikai  
 "D" .....analóg  
 "M" .....üzenet

Az alábbiakban standard I/O csatorna neveinek példái láthatók:

**IX0\_7** .....logikai input - kártya #0 - csatorna #7  
**QD2\_4** .....integer output - kártya #2 - csatorna #4

Egy I/O csatornához csatolt alapértelmezett név módosítására az I/O Csatlakozás szerkesztő „I/O Csatornák csatlakoztatása” parancsa szolgál.

### A.22.3 Komplex I/O berendezés

Egy kártya valamennyi csatornája ugyanolyan típusú (logikai, integer/valós, vagy üzenet) és irányú (input vagy output). Egy komplex I/O berendezés egy különböző típusú vagy irányú csatornákkal rendelkező I/O eszközt jelent. Egy komplex I/O berendezést egyedi I/O kártyák listája képviseli. Ez a keretlistában csak egyetlen kártyahelyet használ.



Egy komplex I/O berendezés definiálásához definiálni kell az I/O berendezést definiáló egyedi kártyák listáját. Emellett be kell írni mindegyik kártya részletes paramétereit is. Az egyedi I/O kártyák listájának beville egy párbeszédablakban történik.

A felhasználó a „**Hozzáadás**” gomb megnyomásával adhat hozzá egy kártyát a pillanatnyi lista végéhez. A „**Beillesztés**” parancs egy új kártyának a listáról pillanatnyilag kiválasztott kártya elé történő beillesztésére szolgál. A listáról kiválasztott kártya eltávolítására a „**Törlés**” parancs szolgál. Az „**Átnevezés**” és

„**Paraméterek**” gomb a kiválasztott kártya nevének és paramétereinek a megváltoztatására szolgál. Egy kártya paramétereinek részletes magyarázata a következő fejezetben található. Egy komplex I/O berendezés maximum **16** egyedi I/O kártyát csoportosíthat. Egy kártya neve (egy I/O berendezésen belül) nem haladhatja meg a **8** karaktert.

#### A.22.4 I/O kártya

Az ISaGRAF I/O kártya könyvtár egy standard interfészt definiál az alkalmazás változó és a cél hardver között. Az alkalmazás leírása során valamennyi I/O változó a cél I/O kártyák csatornához van csatlakoztatva. Egy ISaGRAF I/O kártyát egy **név** és egy, a **beszállítóját** azonosító „**OEM kulcs kód**” definiál. Az egyéb I/O kártya paraméterek az I/O kártya topológiáját (csatornák száma, csatorna irány és típus), valamint annak hardver vagy szoftver konfigurációját írják le.



##### **I/O kártya paraméterek**

Egy I/O kártyához két típusú paraméter létezik: közös paraméterek, amelyek bármilyen ISaGRAF könyvtár kártyához definiálva vannak, és OEM paraméterek, amelyek a kártya telepítésére nézve specifikusak, és amelyeket a hardver beszállítója biztosít. A közös paraméterek az I/O kártya paraméter definiáló ablakának felső részében vannak beírva. Ezek a paraméterek (plusz az I/O kártya neve) azonosítják az ISaGRAF standard I/O kártya interfészt.

Az „**OEM kulcs kód**” egy egyszerű szám, amely a **hardver beszállítóját** azonosítja. Ugyanazon beszállító által definiált valamennyi kártyának azonos OEM kulcs kóddal kell rendelkeznie. Az OEM kulcs kód egy **16 bites előjel nélküli szó**, amely **hexadecimális** formában van beírva. A ICS Triplex ISaGRAF számára fenntartott OEM kód „1”.

A fő paraméterek az I/O kártya topológiáját definiálják. A **csatornák száma** a kártyán rendelkezésre álló csatornák számát határozza meg. A kártya **típusa** a kártya csatornához csatlakoztatható változók típusát jelenti. Az **irány** azt határozza meg, hogy a kártyához csatlakoztatott változók **input** vagy **output** változók-e.

Megjegyzés: Különböző típusú vagy irányú I/O változók nem csoportosíthatók ugyanazon az ISaGRAF I/O kártyán. Ez a jellemző egy komplex berendezést igényel.



##### **Az OEM paraméterek**

Az OEM paraméterek az I/O kártya paraméter definiáló ablakának alsó részében vannak beírva. Ezeket a paramétereket az I/O kártya hardver beszállítója határozta meg, és azok a kártyára specifikusak. Egy kártyához legfeljebb **16** OEM paraméter létezik. Egy kártya lehet OEM paraméter nélkül is. Az ISaGRAF könyvtárrendező lehetővé teszi a hardver beszállító számára mindegyik paraméter azonosításának és formátumának a definiálását, valamint annak módját, hogy az automatizálási programozó azokat hogyan írja be.

A bal oldalon levő ablak az OEM paraméterek listáját tartalmazza. Minden paramétert egy **név** és egy **0** és **15** közötti logikai **szám** azonosít. A jobb oldali terület a listában kiválasztott paraméter részletes leírását tartalmazza. A paraméter teljes leírásának eléréséhez azt ki kell választani a listából. A „**Kiürítés**” gomb

megnyomása visszaállítja a paraméter leírását, és eltávolítja azt a paraméterek listájából.

Figyelmeztetés: Ez a parancs nem „vonható vissza”.

Egy paraméter neve a megfelelő input mező azonosítására szolgál az I/O kártya csatlakoztatása során, ha a mezőt az automatizálási kezelőnek definiálnia kell. Egy paraméter nevének a következő szabályoknak kell eleget tennie:

- egy név maximális hossza **16** karakter
- az első karakternek **betűnek** kell lennie
- a többi karakternek **betűnek, számjegynek**, vagy („\_”) karakternek kell lennie

A paraméter típusa a paraméter **belső formátumát**, és az alkalmazás I/O csatlakoztatása közben **input formátumát** definiálja. Az alábbiakban a rendelkezésre álló belső formátumok listája látható:

**szó** .....előjel nélküli 16 bites szó  
**hosszú** .....előjel nélküli 32 bites szó  
**szó hexa** .....előjel nélküli 16 bites szó  
**hosszú hexa** .....előjel nélküli 32 bites szó  
**logikai** .....előjel nélküli 16 bites szó (csak a legalacsonyabb bit van használva)  
**karakter**.....előjel nélküli 16 bites szó (csak a legalacsonyabb bit van használva)  
**karaktorsor** .....16 bites tömb, amely egy nulla-lezárásos karaktersort tartalmaz  
**lebegő** .....egyszeres pontosságú 32 bites lebegő érték

A rendelkezésre álló input formátumok az alábbiakban vannak felsorolva:

**szó** .....előjel nélküli decimális szó  
**hosszú** .....decimális hosszú szó  
**szó hexa** .....előjel nélküli hexadecimális szó  
**hosszú hexa** .....előjel nélküli hexadecimális hosszú szó  
**logikai** .....„igaz” vagy „hamis”  
**karakter**.....egyetlen karakter  
**karaktorsor** .....ascii karaktorsor (max. 15 karakter)  
**lebegő** .....egyszeres pontosságú lebegő érték

A „**hozzáférés**” ablak annak definiálására szolgál, hogy a végfelhasználó hogyan érheti el a paramétert. Ha a „**Felhasználó által definiálva**” lehetőség be van állítva, akkor a paraméter az I/O kártya csatlakoztatása közben input mezőként látható. A paraméter szerkesztéshez alapértelmezett értéként az OEM paraméter alapértelmezett értéke van használva. Ha a „**Rejtett**” lehetőség be van állítva, akkor a paraméter egy konstans, és nem jelenik meg az I/O kártya csatlakozási ablakban. Az OEM paraméter alapértelmezett értéke a konstans paraméter értékét definiálja. A „**Csak olvasás**” választási lehetőség azt jelzi, hogy a paraméter a felhasználó számára látható, de azt nem lehet módosítani. Annak alapértelmezett értéke konstans értéként van használva.

## A.22.5 Az IEC nyelveken írt funkciók és blokkok

Az ISaGRAF IEC nyelveken írt funkciók és funkcióblokkok könyvtárát kezeli. Az ilyen funkciók vagy funkciók leírására rendelkezésre álló nyelvek az **FBD**

(Funkcióblok diagram), az **LD** (Létradiagram), az **ST** (Strukturált szöveg), vagy az **IL** (Utasítási lista). Megjegyzendő, hogy az LD és FBD nyelvek ugyanabban a diagramban keverhetők egymással. Az **SFC** nyelv (Szekvenciális funkcióábra) nem használható a könyvtárban egy funkció vagy egy blokk leírására. Egy könyvtárelemhez csatolt nyelv a funkció létrehozásakor kerül kiválasztásra, és az később nem változtatható meg.



## **Fordítás**

A könyvtárban definiált funkciókat és funkcióblokkokat le kell fordítani (megerősíteni), mielőtt egy ISaGRAF projektben használni lehetne őket. A funkciókkal és blokkokkal kapcsolatban a könyvtár részéről semmi más nem kell megváltoztatni. Az LD/FBD grafikus szerkesztő egy projekten belül történő alkalmazásakor a könyvtár elemei automatikusan megjelennek a kiválasztó menü ablakában.



Egy, a könyvtárban definiált funkció meghívhatja a könyvtár más funkcióit. Az ISaGRAF rendszer azonban **nem támogatja a rekurzív funkcióhívást**. Egy IEC nyelven írt funkcióblok nem hívhat meg más funkcióblokkot (sem IEC, sem „C” nyelvben).



## **Forráskód bevitele**

Egy könyvtár funkció vagy funkcióblok forráskódja a standard ISaGRAF eszközökkel vihető be: grafikus szerkesztő az LD vagy FBD programokhoz, szövegszerkesztő az ST vagy IL programokhoz. Ezekkel az eszközökkel kapcsolatban ennek a kézikönyvnek az azokra vonatkozó fejezeteiben található bővebb információ. Az ISaGRAF Kódgenerátor a grafikus vagy szövegszerkesztő ablakból közvetlenül meghívható, egy könyvtár funkció vagy blokk forráskódjának fordítására.



## **Helyi változók szótára**

Egy könyvtár funkciónak vagy funkcióbloknak lehetnek helyi változói és helyi definiált szavai. A funkció forráskódjának szerkesztése során a változó deklaráció eléréséhez a felhasználónak a „Fájl” menü „Szótár” parancsának parancsait kell futtatnia a szerkesztő ablakban.



Egy könyvtár funkció vagy funkcióblok nem tud elérni egy globális változót vagy funkcióblok előfordulást. Egy funkció helyi változóit a funkció fő részében kell inicializálni.

Egy IEC nyelven írt funkcióblok helyi változói a blokk projektben való használatakor minden alkalommal másolásra (instanciálásra) kerülnek. Egy előfordulás helyi változói az egyik meghívástól a másikig megtartják értéküket.



## **Az interfész definiálása**

A funkcióknak, vagy funkcióblokkoknak maximum **32** paramétere (input vagy output) lehet. Egy funkciónak mindig egy (és csak egyetlen) visszatérési paramétere van, amelynek a neve a funkció nevével megegyező kell, hogy legyen annak érdekében, hogy eleget tegyen az ST nyelv írási konvencióinak.

Az ablak bal felső oldalán látható lista a paramétereket mutatja, az alábbi meghívási modell szerinti sorrendben: először a meghívó paraméterek, utoljára pedig a visszatérési paraméterek. Az ablak alsó része a listában pillanatnyilag kiválasztott paraméter részletes leírását mutatja. Egy paraméterhez bármelyik ISaGRAF adat típus használható. A visszatérési paramétereknek a meghívó paraméterek után kell elhelyezkedniük a listában. A paraméterek elnevezéseinek a következő szabályoknak kell eleget tenniük:

- a név hossza nem haladhatja meg a 16 karaktert
- az első karakternek betűnek kell lennie
- a többi karakternek betűnek, számjegynek, vagy aláhúzás karakternek kell lennie
- Az elnevezés nem érzékeny a kisbetű/nagybetű beállításra

A „**Beillesztés**” parancs egy új paraméternek a kiválasztott paraméter elé történő beillesztésére szolgál. A „**Törlés**” parancs a kiválasztott paraméter törlésére szolgál. Az „**Elrendezés**” parancs automatikusan átrendezi (sorrendbe állítja) a paramétereket úgy, hogy a visszatérési paraméterek a lista végére kerülnek.

## A.22.6 „C” Funkciók és funkcióblokkok

A „C” funkciók és funkcióblokkok **computer funkciók**, amelyek az automatizálási alkalmazástól vannak meghívva, az ST nyelvű funkció meghívó interfésznek megfelelően.

A funkciók **szinkron** folyamatok. Az ISaGRAF cél alkalmazás a funkció végrehajtása közben felfüggesztődik. A funkcióblokkok műveleteket és statikus rejtett adatokat társítanak. Pl. egy „számláló” funkcióblokk a számlálási műveletet és a számlálás eredményét egyaránt képviseli. A funkciók és funkcióblokkok használhatók a standard automatizálási nyelvi lehetőségek befejezésére, vagy rendszer erőforrások elérésére.



A paraméter definíciós ablak a funkció vagy funkcióblokk meghívó paraméterei nevének és típusának a definiálására szolgál. A „**Szerkesztés**” menü parancsai a kiválasztott funkció vagy funkcióblokk paramétereinek definiálására használatosak. Egy funkciónak maximum **31** meghívó paramétere, és mindig csak **egyetlen** visszatérési paramétere lehet. Egy funkcióblokknak maximum **32** paramétere lehet, meghívási és visszatérési paraméterek tetszés szerinti keverékével. Az alábbiakban az ISaGRAF típusok és „C” típusok közötti összefüggés látható:

<b>LOGIKAI</b>	előjel nélküli hosszú	előjel nélküli 32 bites szó: 1=igaz / 0=hamis
<b>ANALÓG</b>	hosszú	előjel nélküli integer 32 bites szó
<b>VALÓS</b>	lebegő	egyszeres pontosságú lebegő érték
<b>IDŐZÍTŐ</b>	előjel nélküli hosszú	előjel nélküli integer 32 bites szó
(az egység		1 ms)
<b>ÜZENET</b>	kar. *	karaktorsor

amikor egy „C” funkcióhoz vagy funkcióblokkhoz egy üzenet érték van küldve, az nem tartalmazhat nulla karaktereket. A „C” kódhoz küldött karaktorsor nulla lezárású.

Egy funkció vagy funkcióblokk „C” forráskódjának kezelésével, valamint az ISaGRAF célrendszerbe való új elem integrálásával kapcsolatos további információk az ISaGRAF Cél kezelési útmutatójában találhatók.

## A.22.7 Konverziós funkciók

Egy konverziós funkció egy, az ISaGRAF I/O kezelő által mindannyiszor meghívott „C” funkció ahányszor ezt a konverziót használó analóg változók vannak a projektbe beadva, vagy a projektből kiadva.

A funkció kapcsolatot hoz létre a változó **elektromos értéke** (az input érzékelőn olvasva, vagy az output eszközhöz küldve) és annak **fizikai értéke** (az alkalmazás programozásban használva) között. Ennélfogva tehát a funkció két részre oszlik: input konverzió és output konverzió. Az ISaGRAF könyvtárrendező lehetővé teszi a felhasználó számára egy konverziós funkció „C” forráskódjának vezérlését.

Egy konverzió használható egy **integer** vagy **valós** analóg változóhoz. Ebből az következik, hogy a konverziós funkció interfészét mindig lebegő értékek határozzák meg. Az interfész mindegyik konverziós funkcióhoz ugyanaz. Ennek az interfésznek a „C” definíciója a „**TACNODEF.H**” definíciós fájlban van elvégezve.

Egy konverziós funkció „C” forráskódjának kezelésével, valamint egy új elemnek az ISaGRAF célrendszerbe való integrálásával kapcsolatos további információk az ISaGRAF Cél kezelési útmutatójában találhatók.

## A.23 Az Archiváló segédprogram használata

Az ISaGRAF archiváló segédprogramja lehetővé teszi az ISaGRAF projektek és könyvtárak floppyra illetve egy biztonsági másolatokat tartalmazó fájlkönyvtárba történő kimentését. Az ISaGRAF archivumrendező egy párbeszédablak, amelyet az ISaGRAF Projektrendező vagy Könyvtárrendező ablakaiból lehet előhívni.



A megbízható archivumok létrehozásához valamint fenntartásához javasoljuk az alábbi vezérelvek betartását:

- A kimentett tárgy nevét és leírását írja rá a lemez címkéjére
- Ne mentsen ki projekteket és könyvtárakat ugyanarra a lemezre
- Ne mentsen ki különböző projekteket ugyanarra a lemezre

### A.23.1 Az archivumrendező előhívása

Egy projekt, vagy közös adatok kimentéséhez illetve visszatöltéséhez az „Archívum” párbeszédablak a Projektrendező ablak **„Eszközök / Archivum”** menüjéből hívható elő:

Az „Archívum” párbeszédablak a ISaGRAF Könyvtárrendező **„Eszközök / Archivum”** parancsával is futtatható a Könyvtárrendezőben pillanatnyilag kiválasztott könyvtár elemeinek kimentéséhez illetve visszatöltéséhez:



#### **Projektek**

Egy projekt mindig teljes egészében kerül kimentésre. A projekt valamennyi komponense (program forrásfájlok, tárgykód és végrehajtható alkalmazás kód) együtt, ugyanabban az archivumfájlban van elmentve. A **„Tömörítés”** beállítás kiválasztásával csökkenthető a projekt archivum mérete.



#### **Könyvtárelemek**

Az ISaGRAF könyvtárak elemeit egyedileg ki lehet menteni. Egy könyvtárelem valamennyi komponense (műszaki megjegyzés, definíció, interfész, forráskód) együtt, ugyanabban az archivumfájlban van elmentve.



#### **Közös adatok**

A Projektrendező ablak **„Eszköz / Archivum / Közös adatok”** parancsa lehetővé teszi a felhasználó számára az ISaGRAF Workbenchben létező „közös tartományú” adatok biztonsági kimásolását illetve visszatöltését. Ez a parancs nincs hatással az ISaGRAF könyvtárakra. Az alábbiakban az ezzel a paranccsal másolható fájlok listája szerepel:

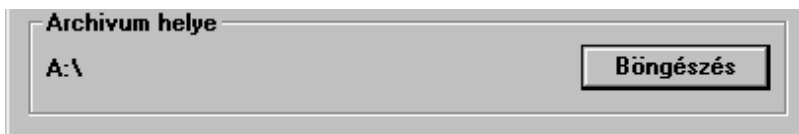
**common.eqv**.....közös definiált szavak

**oem.bat**.....a felhasználó által definiált MS-DOS parancsfájl

Ezek a fájlok egyenként, eredeti formájukban kerülnek kimentésre az archivumlemezre. A megfelelő archivumfájlok soha nincsenek tömörítve.

### A.23.2 Beállítási lehetőségek

Az ISaGRAF archívumokhoz használt elérési útvonal a párbeszédablak alján látható. A lemezen való böngészéshez illetve egy másik archívumlemez és alkönyvtár kiválasztásához nyomja meg a „Böngészés” gombot.



Ha a „Tömörítés” beállítási lehetőség ki lett választva, akkor a „Biztonsági másolat” művelet során készített valamennyi archívumfájl tömörítve lesz. Ez a beállítási lehetőség nagyon hasznos egy nagyméretű projekt archívumfájl méretének csökkentésére azért, hogy az egyetlen lemezre kimenthető legyen. Könyvtárkomponensekhez általában nincs szükség az archívum tömörítésére. Az ISaGRAF Archívumrendező egy archívumfájl visszatöltésekor automatikusan felismeri az archívumfájl státuszát (tömörített vagy tömörítés nélküli). Ebből következően tehát a „Tömörítés” beállítási lehetőség nem befolyásolja a „Visszatöltés” műveletet.



### A.23.3 Biztonsági másolat és visszatöltés

A „Workbench” lista (a bal oldalon) a merevlemezre telepített ISaGRAF Workbenchben létező tárgyakat sorolja fel. Az „Archívum” lista (a jobb oldalon) a meghatározott archívumlemezen és alkönyvtárban kimentett tárgyakat mutatja.

#### ☰ Biztonsági másolat

Egy tárgy archívumba történő kimentéséhez válassza ki a tárgyat a bal oldali listából (az ISaGRAF workbenchben levő tárgyak listája), és nyomja meg a „Biztonsági másolat” gombot. A listából egynél több tárgy is kiválasztható. A „Biztonsági másolat” gomb letiltódik, ha a **jobb** oldali listából választanak ki egy elemet (visszatöltési mód).

#### ☰ Visszatöltés

Egy tárgynak az archívumból az ISaGRAF Workbenchbe történő visszamásolásához válassza ki a tárgyat a **jobb** oldali listából (archivált tárgyak), és nyomja meg a „Visszatöltés” gombot. A listából egynél több tárgy is kiválasztható. A „Visszatöltés” gomb letiltódik, ha a **bal** oldali listából választanak ki egy elemet (biztonsági másolat mód).

#### A.23.4 Archívumfájlok

Az ISaGRAF archívumrendező minden kimentett tárgyhoz egy egyedi archívumfájlt hoz létre. Az archívumfájl neve megegyezik a tárgy nevével. A fájl típusát a fájlkiterjesztés jelzi. A használt fájlkiterjesztések az alábbiakban vannak felsorolva:

.pia .....projekt  
.bia .....I/O kártya  
.iia .....funkció, IEC nyelven  
.aia .....funkcióblokk, IEC nyelven  
.uia .....C funkció  
.fia .....C funkcióblokk  
.cia .....C konverziós funkció  
.ria .....I/O konfiguráció  
.xia .....I/O berendezés

## A.24 Egy teljes dokumentum kinyomtatása

A felhasználó az ISaGRAF Dokumentumkészítő segítségével a kiválasztott projekthez egy komplett dokumentumot állíthat össze és nyomtathat ki. Ez a Projektrendező vagy a Program ablakok „**Projekt / Nyomtatás**” parancsaival hívható meg, egy komplett dokumentum kinyomtatására. A Dokumentumkészítő egyetlen ISaGRAF dokumentum tartalmának kinyomtatására az összes többi ISaGRAF szerkesztő „**Nyomtatás**” parancsával is futtatható. A Dokumentumkészítő azonban mindkét esetben ugyanazokat a lehetőségeket biztosítja.

A „**Szerkesztés**” menü parancsai a projekt dokumentumba illesztendő elemeinek a definiálására szolgálnak. Ennek elvégzésével a felhasználó a kívánt dokumentum „tartalomjegyzékét” állítja össze. A projekt dokumentumba a projekttel kapcsolatos bármiféle információ (programok, változók, beállítások, I/O csatlakozás stb.) beilleszthető. Ebben a dokumentumban nem jelenhet meg más projektből vagy az ISaGRAF könyvtárakból származó információ.



A „**Fájl / Nyomtatás**” parancs elkészíti a dokumentumot, és azt a nyomtatóra küldi, a specifikált tartalomjegyzéknek megfelelően. A „Nyomtatás” feladat több percet vehet igénybe, a dokumentum elkészítése és formattálása miatt. Az ISaGRAF Workbench bármilyen más parancsának a futtatása előtt kimondottan ajánlatos megvárni, amíg a „Nyomtatási feladat” befejeződik az ISaGRAF Dokumentumkészítő ablakban. Az egész dokumentum elkészítése nagy helyet igényelhet a merevlemezén. A lemez megtelését egy hibaüzenet jelzi. Ilyen esetben a felhasználónak vagy helyet kell felszabadítani a lemezen fájlok eltávolításával, vagy csökkentenie kell a nyomtatási feladat méretét. A „**Nyomtatás**” parancs futásakor megjelenik egy párbeszédablak. A felhasználó ennek segítségével beírhat egy megjegyzést, amely az tényleges nyomtatási parancs leírását tartalmazza. Ezek a megjegyzések egy előzményfájlban vannak eltárolva, és minden későbbi dokumentum (beleértve a jelenlegit is) első oldalán kinyomtatásra kerülnek.

### A.24.1 A tartalomjegyzék testre szabása

A „**Szerkesztés**” menü a dokumentum „Tartalomjegyzékének” definiálásához szükséges parancsokat tartalmazza. A felhasználó a különböző parancsok segítségével egy alapértelmezett táblázatot használhat (a projekt valamennyi komponensével), egy specifikus táblázatot hozhat létre (csak bizonyos komponensekkel), vagy a táblázatban levő tételeket áthelyezheti illetve módosíthatja.



#### **Az alapértelmezett lista**

A „**Szerkesztés**” menü „**Alapértelmezett lista**” parancsa egy standard tartalomjegyzéket definiál a dokumentumhoz, amely a projekt valamennyi komponensét tartalmazza. A standard táblázat a következőkből áll:

- Projekt leíró
- Hierarchikus faszerkezet (a programok közötti kapcsolatok)
- Forráskód bármelyik programhoz
- Naplófájl bármelyik programhoz

- Közös definíciók
- Globális definíciók
- Lokális definíció bármelyik programhoz
- Globális változók
- Lokális változók bármelyik programhoz
- Alkalmazás beállítási lehetőségek
- I/O Csatlakozás
- Változók listája
- Konverziós táblázatok
- Sűrített kereszthivatkozások
- Részletezett kereszthivatkozások
- Deklarálás összefoglalás
- Hálózati címek hozzárendelése
- Módosítások előzményei

A tartalomjegyzék a „**Fájl / Kimentés**” parancs segítségével lemezre menthető. Ez a parancs nem áll rendelkezésre, amikor a dokumentumkészítő egy ISaGRAF szerkesztőből egyetlen dokumentum kinyomtatására van futtatva.



### **Kivágás és beillesztés**

A táblázat sorrendjének testre szabásához a listában levő tételek a „**Szerkesztés / Kivágás**” és „**Szerkesztés / Beillesztés**” parancsok segítségével vihetők át más helyre. A Dokumentumkészítő lehetővé teszi több tétel egyszerre történő kiválasztását, s így egy több tételből álló csoport is kivágható és áthelyezhető.



### **A táblázat törlése**

A „**Szerkesztés / Visszaállítás**” parancs alaphelyzetbe állítja vissza a tartalomjegyzéket, hogy az egyszeres tételek beillesztésével teljesen újjáépíthető legyen.



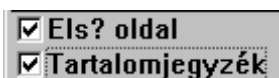
### **Tételek beillesztése a táblázatba**

A „**Szerkesztés / Beillesztés**” parancs futásakor megjelenik a „**Tétel hozzáadása**” párbeszédablak. A felhasználó ennek segítségével tételeket (a projekt komponenseit) illeszthet be a tartalomjegyzékbe.

Egy programra vonatkozó tételhez a „**Program**” kombinált ablakot kell használni a program nevének kiválasztásához. A kiválasztott tételnek a tartalomjegyzékbe történő hozzáadásához meg kell nyomni a „**Hozzáadás**” gombot. Ugyanaz a tétel csak egyszer jelenhet meg a táblázatban.

## **A.24.2 Beállítási lehetőségek**

A „**Beállítások**” menü parancsai a készített dokumentum formátumának definiálására és testre szabására szolgálnak. A további beállítási lehetőségek a Dokumentumkészítő ablak gombjairól érhetők el közvetlenül:



Ha a „**Font oldál**” lehetőség be lett állítva, akkor a dokumentum elején egy fejléc oldal nyomtatódik ki, amely a projekt címét és a nyomtatási előzményeket tartalmazza. Ha ez a lehetőség nem lett beállítva, akkor az első oldal az első kinyomtatandó tétellel kezdődik.

Ha a „**Tartalomjegyzék**” lehetőség be lett állítva, akkor a készített dokumentum végén egy tartalomjegyzék nyomtatódik ki.

Kezdetben egyik beállítási lehetőség sincs beállítva, ha a Dokumentumkészítő egy ISaGRAF szerkesztő (program, szótár, stb.) „**Nyomtatás**” parancsával van futtatva.



## SFC diagramok

A „**Külön SFC szintek**” beállítási lehetőség arra utasítja a rendszert, hogy minden SFC programnál először az SFC 1. szintjét (diagramok és megjegyzések), majd a 2. szintű programozást nyomtassa ki. Ha ez a lehetőség nincs beállítva, az 1. és 2. szintek együtt jelennek meg ugyanabban a nyomtatásban.



## Oldal formátum


A „**Beállítások**” menü „**Oldal formátum**” parancsa a Dokumentumkészítő által egy oldal formattálásakor kezelt fő paraméterek definiálására használatos. A következő paraméterek specifikálhatók:

- Bal margó: (1 vagy 2 centiméter, vagy nincs margó)
- Oldal keret: Ha ez a lehetőség be van állítva, akkor a nyomtatott oldalakat egy keret veszi körbe.



## Oldal címsablon

A „**Beállítások**” menü „**Oldal cím**,” parancsa az oldalak aljára nyomtatott címléc tartalmának definiálására használatos. Ennek a keretnek a standard elrendezése a következő:

	Szöveg 1	ISaGRAF – „PrName” Projekt	dátum
	Szöveg 2		
	Szöveg 3	<b>Felhasználó definiált cím</b>	oldal

A főcím első sorát (az ISaGRAF projekt nevével), a pillanatnyi dátumot, és az oldalszámot a Dokumentumkészítő generálja automatikusan, és azokat nem lehet változtatni.

A keret bal oldalán levő három szövegsort (szöveg1, szöveg2, szöveg3), valamint a főcím második sorát a felhasználó definiálja. A felhasználó a bal oldali keretbe nyomtatott jelképet is megváltoztathatja. A felhasználónak egy másik jelkép használatához meg kell adnia egy bittérkép formátumú képfájl (.BMP) elérési útvonalát. A kép bármilyen méretű lehet. Az a kinyomtatott oldal pontos méretének megfelelően ki lesz nyújtva, illetve össze lesz zsugorítva. A párbeszédablakban a jelkép területre való kattintásra megjelenik az új specifikált kép. A képfájlnak a „**Nyomtatás**” parancs futtatásakor jelen kell lennie a lemezen (a megadott alkönyvtárban, a megadott fájlneven).



## **Karakterfontok kiválasztása**

A „Beállítások” menü „Szöveg font” és „Cím font” parancsai a szövegek és a dokumentum valamely tétalcímének nyomtatáskor használt fontok definiálására használatosak. A szövegekhez és címekhez a karakterek mérete és stílusa ugyancsak kiválasztható. Egy font kiválasztása a Windows által definiált standard párbeszédablakkal történik. Egy adott szöveg (literális programok, diagramokon belüli nevek) a kiválasztott méretű, stílusú, és fontú karakterekkel kerül kinyomtatásra. A címekhez kiválasztott fonttal csak a címek lesznek kinyomtatva.

ha a karakterek fontjai nincsenek definiálva, akkor a printer standard fontja lesz használva bármely szöveghez, a következő stílusokkal:

- „Normál” stílus szövegekhez és diagramokon belüli nevekhez
- „Kövér” stílus címekhez

## A.25 Jelszóvédelem

Az ISaGRAF Workbench egy teljes adatvédelmi rendszert tartalmaz, amely lehetővé teszi a felhasználó számára, hogy a projekteket és könyvtárelemeket jelszóvédelemmel lássa el. Egy könyvtárelem lehet egy I/O konfiguráció, I/O kártya vagy komplex berendezés, IEC nyelven megírt funkció vagy funkcióblokk, „C” funkció, funkcióblokk, vagy konverziós funkció. Egy adott szótárvédelmi adatbázis egyetlen projekthez van rendelve, és azt több projekt nem alkalmazhatja közösen.

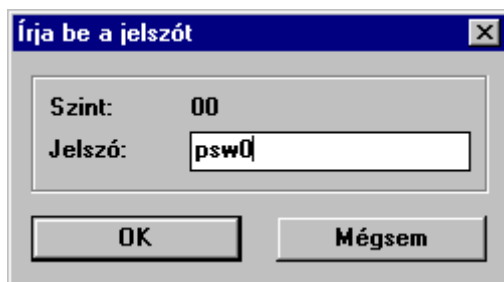
### Védelmi szintek

A felhasználó egy projekten vagy könyvtáron belül maximum **16** hozzáférési szintet definiálhat, amelyek mindegyikéhez külön jelszó tartozik. A hozzáférési szintek egy hierarchikus rendszerben vannak besorolva. Ezeket **0** és **15** közötti számok jelzik. A legmagasabb hozzáférési szint **0**-val van jelölve. Ha egy felhasználó ismeri a jelszót, akkor a megfelelő hozzáférési szinttel védett összes tételhez, valamint az összes alacsonyabb szinttel védett tételhez hozzá tud férni. Egy projekt valamennyi elemi parancsa vagy könyvtáreleme külön-külön levédhető egy-egy hozzáférési szinttel. Pl. az ISaGRAF menüből az „Alkalmazási kód készítése” parancs külön levédhető. Az elemi adat lehet egy program, egy beállítási lehetőségeket tartalmazó lista, egy könyvtárelem műszaki megjegyzése, stb.

### Jelszóvédelem definiálása

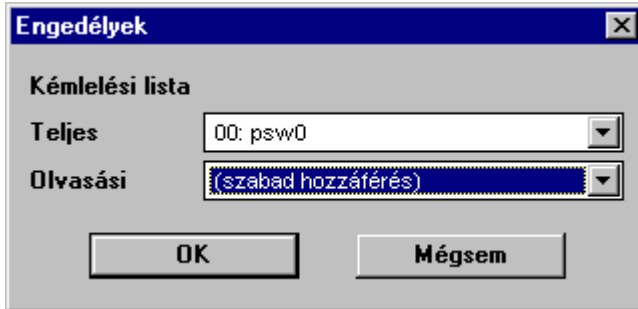
Egy projekt vagy egy könyvtárelem jelszoinak és hozzáférési szintjeinek a meghatározására az ISaGRAF menük „**Jelszó beállítása**” parancsa szolgál. Ez parancs az ISaGRAF Projektrendező (egy projekthez), vagy az ISaGRAF Könyvtárrendező (egy könyvtárelemhez) menüből hívható meg. Ennek a parancsnak az első alkalommal történő futtatásához nincs szükség jelszóra. Amennyiben már definiálva lettek jelszók, a felhasználó az általa ismert legmagasabb szintű jelszó beírásával férhet hozzá ehhez a parancshoz. A magasabb szinten levő jelszókat és védett tételeket tehát nem lehet módosítani. A „**Jelszó beállítás**” parancs lehetővé teszi a felhasználó számára, hogy meghatározza a különböző hozzáférési szinteknek megfelelő jelszókat, és a definiált szintekkel levédje az elemi parancsokat vagy adatokat.

A jelszókat (a védelmi szinteknek megfelelően) a felső listában egy sorra történő dupla kattintással lehet beírni. A jelszó bevitelére a következő párbeszédablak szolgál.



Az alsó területen látható lista azokat a különböző tételeket (adatokat vagy funkciókat) tartalmazza amelyek levédhetők, valamint feltünteti az „olvasási hozzáférés” illetve „teljes hozzáférés” engedélyekhez rendelt pillanatnyi védelmi szinteket. Egy védelmi szint „olvasási” joggal történő ellátásával megakadályozható, hogy a megfelelő jogosultsággal nem rendelkező felhasználók akár csak kinyithassanak vagy kinyomtathassanak egy dokumentumot.

A kiválasztott tétel illetve adat jogosultságának beállításához kattintson duplán egy sorra az alsó listában. Ekkor kinyílik a következő párbeszédablak:



Mindkét jogosultság beállítható vagy „szabad hozzáférésre”, vagy egy jelszóval definiált védelmi szintre. Teljes hozzáférési jogosultság nem csatolható egy olyan szinthez, amelynek alacsonyabb prioritása van, mint az olvasási jogosultságra kiválasztott szintnek.

Megjegyzendő, hogy bizonyos, az ISaGRAF Workbench használata során természetesen látható dokumentumoknál (mint amilyen pl. egy projekt leíró), az olvasási hozzáférés nem védhető le jelszóval.



### **Hozzáférés a védett adatokhoz**

A Workbench elindításakor nem kell megadni jelszót illetve a felhasználó nevét. Valahányszor egy felhasználó egy védett adathoz vagy funkcióhoz akar hozzáférni, ehhez először egy párbeszédablakban be kell írnia a jelszót.

Amennyiben a felhasználó a megfelelő jelszót (vagy egy magasabb szinthez csatolt jelszót) írta be, úgy továbbléphet. A felhasználó által beírt minden jelszó elmentésre kerül a memóriában, így a felhasználónak azt később nem kell ismét beírnia. A tárolt jelszók minden olyan alkalommal megőrzésre kerülnek, amikor egy ISaGRAF eszköz egy másik ISaGRAF eszközből van futtatva (pl. a Projektrendező futtatja a Programrendezőt). Az utolsó ISaGRAF ablak bezárásakor a tárolt jelszók elvesznek. A projekt-szerkesztés során beírt jelszókat, illetve a Könyvtárrendező vagy az Archivumrendező használatát nem lehet több felhasználó között megosztani. Ha a felhasználó rossz jelszót ír be, akkor nem tudja futtatni a kiválasztott funkciót.



### **Az archivumrendezőhöz történő kapcsolatok**

Egy tárgy (projekt vagy könyvtárelem) archivumlemezre történő kimentésekor az „Archívumba készített biztonsági másolat” adatvédelmi tétel kerül előhívásra. Ez a Workbenchben (merevlemezen) a tárgyhöz csatolt adatvédelmi rendszernek felel

meg. Amennyiben az archívumlemezen már létezik a szóban forgó tárgy, annak adatvédelmi rendszere nem lesz tesztelve. Az ISaGRAF Archívumrendező „**Biztonsági másolat**” parancsa az adatvédelmi információt a tárggyal együtt kimentí az archívumlemezre.

Egy olyan tárgy visszatöltésekor amely már létezik a Workbenchben (merevlemezen), a „**Felülírás az archívumban szereplővel**” című adatvédelmi tétel kerül előhívásra. Ez a Workbenchben (merevlemezen) a tárggyhoz csatolt adatvédelmi rendszernek felel meg. Az archívumlemezen levő adatvédelmi rendszere nem lesz tesztelve. Ha ez a parancs érvényesítésre kerül, akkor a visszatöltött adatvédelmi információ felülírja a merevlemezen levőt.

## **Változók és I/O csatornák egyedi védelmének beállítása**

Az ISaGRAF workbench egy hierarchikus jelszókra épülő teljes adatvédelmi rendszert biztosít. A változó-deklaráció és I/O csatlakozás globálisan egy jelszóval levédhető. Ezen felül, az ISaGRAF lehetővé teszi egyedi védelem beállítását bármely változóhoz vagy I/O csatornához. Ez az alábbiakat feltételezi:

- a jelszók már meg lettek határozva a jelszó-definiáló rendszerben (a Projektrendező ablak „**Projekt / Jelszó beállítása**” parancsának használatával), így az egyedi védelemhez védelmi szintek állnak rendelkezésre.
- az egyedi védelemhez magasabb prioritású védelmet használnak, mint a globális változó- vagy I/O védelem.

Ha egy változó vagy egy I/O csatorna egyedi védelemmel rendelkezik, akkor annak neve mellett egy kis ikon jelenik meg a szótár- illetve I/O csatlakozási ablakban.

A kiválasztott változó vagy csatorna egyedi védelmének beállításához vagy eltávolításához a szótár- vagy az I/O csatlakozási ablakban levő „**Szerkesztés**” menü „**Védelem beállítása**” illetve „**Védelem eltávolítása**” parancsokat kell használni. Mindkét parancs egy érvényes jelszó beírását kéri ahhoz, hogy a változóhoz vagy csatornához egy védelmi szintet lehessen rendelni. Ezután pedig valahányszor meg akar változtatni egy egyedi védelemmel rendelkező változót vagy egy csatorna csatlakozását, be kell írnia egy megfelelő prioritású szintnek megfelelő jelszót.

**Figyelmeztetés:** Ha egy változó vagy egy csatorna egy szinttel le van védve és a megfelelő jelszót eltávolítják a védelmi rendszerből, valamint amennyiben nincs definiálva magasabb szintű jelszó, egy új, megfelelő szintű jelszó definiálása nélkül a változót illetve csatornát többé nem lehet megváltoztatni.

## A.26 Haladó programozási eljárások

Ez a fejezet az ISaGRAF Workbench-el és a célrendszerrel kapcsolatos bővebb információkat tartalmaz. Javasoljuk, hogy ennek a fejezetnek az elolvasása előtt a felhasználó alaposan ismerje meg az ISaGRAF eszközeit és módszereit.

### A.26.1 Az ISaGRAF eszközökkel kapcsolatos további tudnivalók

Az ISaGRAF szerkesztő eszközök használata során a felhasználó a **jobb oldali egér gomb** megnyomásával egy előugró menüt hívhat elő, amely a fő szerkesztőparancsokat tartalmazza. A menü a kurzor pillanatnyi helyénél nyílik ki. Ez azért nagyon hasznos, mert így a kivágási és beillesztési parancsok során csökkenthető az egérrel elvégzendő műveletek száma.

Az ISaGRAF eszközök támogatják a **többszörös végrehajtást**. Bár ugyanaz az eszköz nem nyitható ki kétszer ugyanannak a dokumentumnak a szerkesztéséhez, ugyanazzal az eszközzel azonban ki lehet nyitni különböző ablakokat különböző objektumok párhuzamos műveletekkel történő szerkesztéséhez.

Az eszköztárakon levő grafikus gombokról szóló információkat egyéb parancsok segítségével lehet beszerezni. Egy eszköztár tartalmának előugró menün történő megjelenítéséhez kattintson duplán egy üres helyre az eszköztáron. Az egér kurzorát egy grafikus nyomógomb fölé tartva megjelenik a nyomógombnak megfelelő parancs szövege.

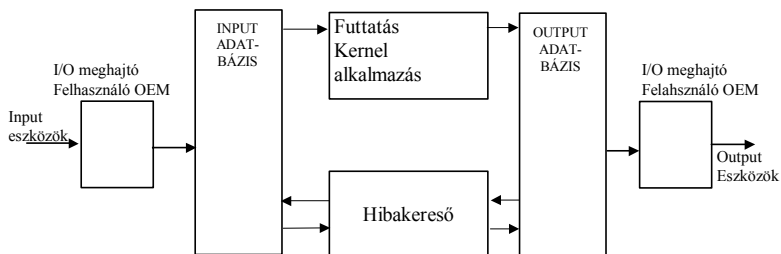
### A.26.2 Zárt bemenetek/kimenetek (I/O-k) és virtuális I/O-k

Egy I/O kártya **virtuálisként** történő definiálása leválasztja a fizikai I/O csatornák feldolgozását. Ha egy kártya virtuálisként van definiálva, az ISaGRAF kernelműveletek nem változnak. Az egyetlen különbség az, hogy az input-érzékelők nincsenek leolvasva, és az output eszközök nincsenek frissítve. Ebben a módban az ISaGRAF hibakereső használatával módosíthatók az input értékek. A **Virtuális** attribútum egy egész kártyára vonatkozik. Ez az I/O kártya definiálása során, az alkalmazás kódgenerálása **előtt** van beprogramozva. A **virtuális** attribútum egy **statikus** jellemző, amely az alkalmazás leállításakor és újraindításakor kerül eltávolításra.

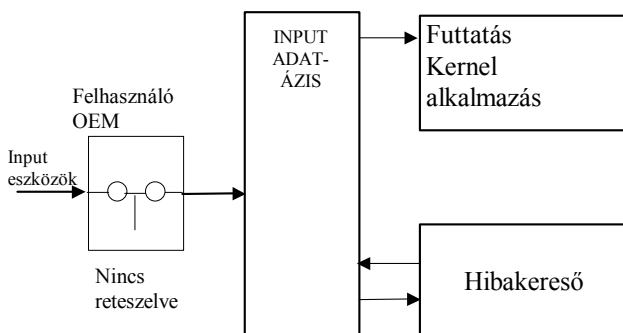
Egy másik lehetőség az I/O változó **zárolása**. Ez egy fizikai eszköz és a megfelelő ISaGRAF I/O változó leválasztásából áll. A változózárolás valamint a zárolás felbontása a hibakeresőn keresztül kerül végrehajtásra. A változózárolás **dinamikus** művelet, és nincs memorizálva az alkalmazás újraindulásakor. A **reteszelés** művelet egyszerre csak **egyetlen** változóra (egy I/O csatornára) vonatkozik. Az alábbi ábra a fő I/O vezérlési jellemzőket mutatja:

	<i>Virtuális attribútum</i>	<i>Reteszelés parancs</i>
kiválasztó eszköz	I/O kártya csatlakozás	hibakereső
definíció	statikus	dinamikus
kiválasztási mód	kártya	változó
alkalmazás	érvényesítés és tesztek	karbantartás

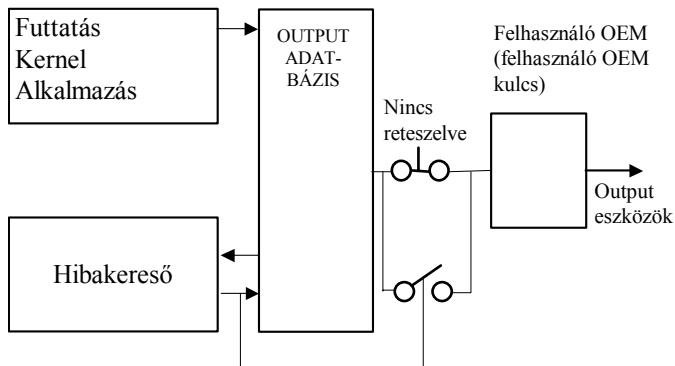
A következő ábra az ISaGRAF feladatok közötti I/O adatáramlást ismerteti:



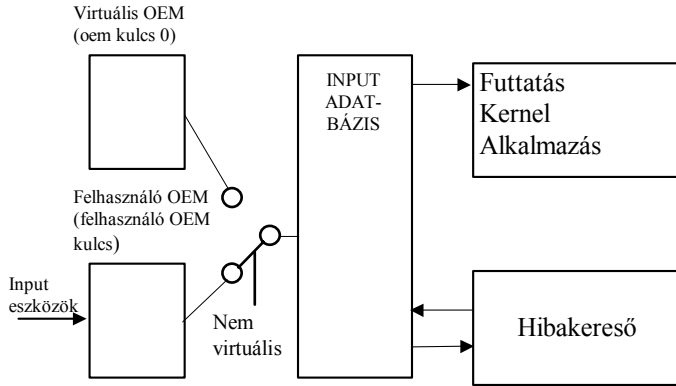
Ha egy inputváltozó reteszelve van, akkor az adatbázishoz való különböző hozzáférések nem változnak, de az inputeszköz leválasztódik. Az inputértékek a hibakeresővel állíthatók be, és azokat az ISaGRAF kernel dolgozza fel:



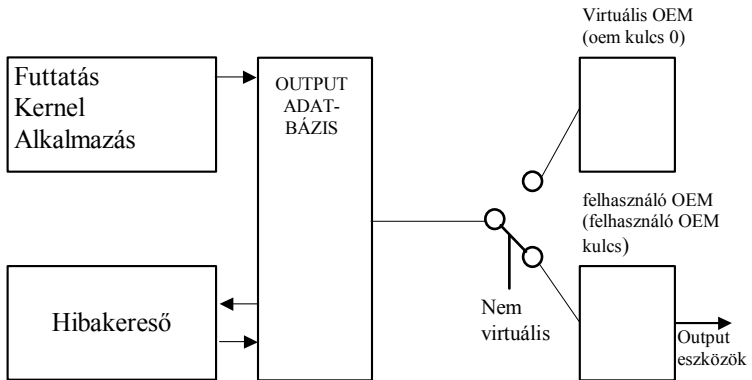
Ha egy outputváltozó van reteszelve, akkor a futtatási kernel és az output meghajtó leválasztódik. Ebben az esetben továbbra is lehetséges az outputeszközhöz való hozzáférés, az outputmeghajtón keresztül, az ISaGRAF hibakeresővel:



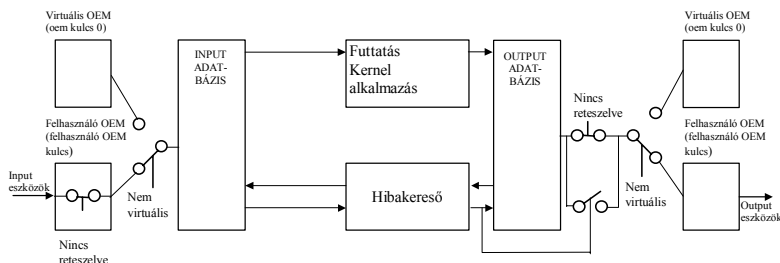
Egy input virtuális attribútumának a beállításakor az input adatbázis és a hozzá tartozó inputeszközök leválasztódnak. Egy virtuális I/O meghajtó felváltja a valódit.



A virtuális attribútum beállítása inputkártyákra és outputkártyákra vonatkozóan azonos szabályok szerint történik. Outputkártyáknál az ISaGRAF kernel frissíti az output adatbázist. Ez az adatbázis és a hozzá tartozó outputeszközök azonban le vannak választva. Egy virtuális I/O meghajtó felváltja a valódit.



Az összes lehetőség összefoglalása:



## A.26.3 PC-PLC kapcsolat érvényesítése

Az ISaGRAF workbench és a cél PLC közötti rossz kommunikációval kapcsolatos legtöbb problémát egy „leválasztva” üzenet jelzi a hibakereső ablakban. Bármiféle diagnosztikai vizsgálat elvégzése előtt érvényesíteni kell a kommunikációt amikor **nincs aktív alkalmazás** a cél PLC-ben. Ilyen módon a soros kommunikációs kapcsolat önmagában, a végrehajtással kapcsolatos behatásoktól elkülönítve érvényesíthető.

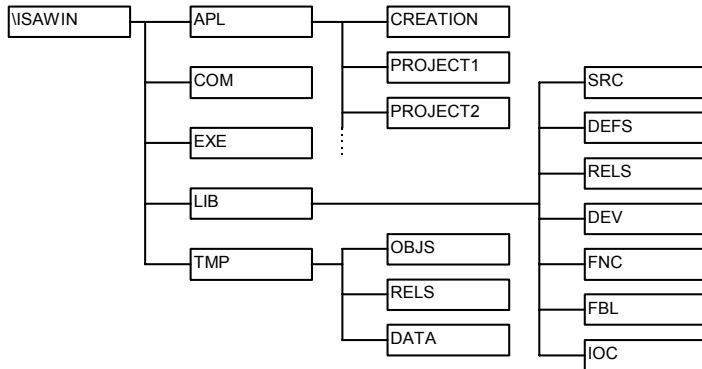
A konverziós funkciók és C funkciók leírásához használt „C” nyelv lehetővé teszi a célrendszerhez történő közvetlen hozzáférést. Egy ilyen szoftverkomponensben levő programozási hiba rendszerhibákat vagy az ISaGRAF rendszer rendellenes viselkedését okozhatja. Ilyen problémák előfordulhatnak olyankor, amikor az I/O meghajtók az ISaGRAF I/O eszközkészlettel lettek kifejlesztve. Pl. Egy I/O kártya érvénytelen buszcímhez történő csatlakoztatása rendszerhibákat okozhat. A következő táblázat a hibadiagnosztika rövid összefoglalását tartalmazza:

státusz	kontextus	Diagnosztika
„leválasztva” (letöltés előtt)		<ul style="list-style-type: none"> <li>- a cél nem fut</li> <li>- hiányzó kábel / nem megfelelő kábel</li> <li>- érvénytelen kapcsolat paraméterek</li> <li>- az ISaGRAF cél rosszul lett telepítve</li> </ul>
„leválasztva” (letöltés után)	ciklustól-ciklusig kiindulási mód	<ul style="list-style-type: none"> <li>- érvénytelen I/O konfiguráció</li> <li>- rendszer összeomlás</li> </ul>
	valós idő kiindulási mód	<ul style="list-style-type: none"> <li>- érvénytelen I/O konfiguráció</li> <li>- rendszer összeomlás („C” programozás következtében)</li> </ul>
„nincs alkalmazás”		<ul style="list-style-type: none"> <li>- nincs letöltve az alkalmazás</li> <li>- az alkalmazás nem indult el („C” programozás következtében)</li> <li>- Intel/Motorola összeférhetetlenség</li> <li>- Érvénytelen verziójú cél</li> </ul>

## A.26.4 ISaGRAF fájlkönyvtárak

Az ISaGRAF Workbench egy feladatorientált fájlkönyvtár-szerkezettel működik. Ennek az architektúrának a főkönyvtárát az ISaGRAF telepítése során a

felhasználó határozza meg. Az ISaGRAF főkönyvtár alapértelmezett neve **ISAWIN**. A telepítő program által létrehozott standard lemez-architektúra a következő:



Az alábbiakban a standard ISaGRAF alkönyvtárak vannak felsorolva

## KÖNYVTÁR TARTALOM

<b>APL</b>	az ISaGRAF projektek főkönyvtára minden projekt egy-egy alkönyvtárnak felel meg, amely a projekt összes adatát tartalmazza Más projekt-csoportokhoz egyéb könyvtárak is létezhetnek. Az ISaGRAF telepítőprogram létrehoz egy „ <b>SMP</b> ” alkönyvtárat, amely minta alkalmazásokat tartalmaz.
<b>COM</b>	„közös” tartományú adatok Az adatokat bármelyik projekt használhatja
<b>EXE</b>	ISaGRAF programok és súgófájlok
<b>LIB</b>	ISaGRAF könyvtárak: - elemek listái - az egyes elemek paraméterei vagy interfésze - technikai megjegyzések
<b>LIB\IOC</b>	forráskód I/O konfigurációkhoz
<b>LIB\FNC</b>	IEC nyelveken írt funkciók forráskódja
<b>LIB\FBL</b>	IEC nyelveken írt funkcióblokkok forráskódja
<b>LIB\SRC</b>	konverziók és C funkciók forráskódja
<b>LIB\DEFS</b>	konverziók és C funkciók forrás-fejszora
<b>LIB\RELS</b>	Konverziók és C funkciók tárgykódja
<b>LIB\DEV</b>	parancsfájlok „C” könyvtárak kidolgozásához fájlkészítők, kapcsolatlisták, stb.
<b>TMP</b>	Ideiglenes fájlok: a TMP alkönyvtárai az ISaGRAF Kódgenerátor számára vannak fenntartva, és azokat nem lehet törölni.

Az alkönyvtárak a lemezen más helyre átvihetők. Ha a felhasználó egy standardtól eltérő architektúrával rendelkezik, akkor az alkönyvtárak elérési útvonálát az ISaGRAF **EXE** alkönyvtárában levő **ISA.ini** inicializáló fájl **WS001** szekciójában kell deklarálni. Az alábbiakban a **WS001** szekció elemei vannak felsorolva:

<b>Isa</b>	az ISaGRAF architektúra főkönyvtára
<b>IsaExe</b>	ISaGRAF programok és súgófájlok főkönyvtára
<b>IsaApl</b>	az ISaGRAF projektek főkönyvtára
<b>IsaTmp</b>	ideiglenes fájlok könyvtára
<b>IsaSrc</b>	könyvtár forráskód könyvtára
<b>IsaDefs</b>	könyvtár forrás fejsorok könyvtára

Megjegyzendő, hogy amennyiben az IsaTmp sort egy másik könyvtárhoz rendeli, úgy az új könyvtárban létre kell hoznia az OBJS, RELS és DATA alkönyvtárakat. A következő példa a **WS001** szekcióban szereplő sorokat alkalmazza az standard ISaGRAF lemez-architektúra átdefinálására:

```
;file c:\ISAWIN\EXE\ISA.ini
```

```
[WS001]
Isa=c:\isawin
IsaExe=c:\isawin\exe
IsaApl=c:\isawin\apl
IsaTmp=c:\isawin\tmp
IsaSrc=c:\isawin\lib\src
IsaDefs=c:\isawin\lib\defs
```

Ha az ISaGRAF célhoz „C” funkciókat vagy funkcióblokkokat akar adni, akkor az **ISAWIN\LIB\DEV** könyvtár szolgál az alábbi fejlesztőfájlok tárolására: parancsfájlok, fájlkészítők, térképek, stb. Az **ISAWIN\LIB\RELS** alkönyvtár a „C” fordítás során generált tárgyfájlok és a LINK („KAPCSOLAT”) műveletekhez szükséges ISaGRAF „C” könyvtárak tárolására szolgál.

## A.26.5 Alkalmazás szimbólumok

Egy ISaGRAF alkalmazás minden egyes tárgyára annak nevével (amely a változó deklarálásakor lett beírva), és egy, a kódgenerátor által kiszámított belső **virtuális címmel** történik hivatkozás. Egy változó virtuális címe nem a változó deklarálásakor beírt **hálózati cím**. A virtuális címek kommunikációs tevékenységekhez, valamint az OEM beállítást alkalmazó speciális „C” alkalmazásokhoz használatosak. Amikor az ISaGRAF kódgenerátor fut, az a projekt összes tárgyának (változó, program, lépés,...) nevei és virtuális címei közötti logikai kapcsolatokat tartalmazó ASCII fájlt készít. Ez a fájl bármelyik felhasználói alkalmazásból egyszerűen lekérdezhető az ISaGRAF statikus adatbázisával kapcsolatos információkért. A fájl neve **„APPLI.TST”**, és az ISaGRAF projekt fájlkönyvtárában található: **„ISAWIN\APL\proname”** (ahol „proname” a projekt neve). Ez a fejezet az **„APPLI.TST”** fájl részletes formátumát ismerteti. A leírásban az alábbi fő elnevezéseket alkalmaztuk:

<b>VA</b>	virtuális cím
<b>ATTR</b>	egy változó attribútuma
<b>USP</b>	„C” funkció

Az alábbiakban egy változó attribútumainak lehetséges értékei láthatók. Ezek az értékek az **„attribútumok”** mezőkben találhatóak:

+X	belső változó
+C	csak olvasható belső változó
+I	input változó
+O	output változó

A virtuális címek kivételével valamennyi szám decimális egészként van feltüntetve. A virtuális címek (**VA**) hexadecimális 4 számjegyű számként vannak kifejezve, és a „!” karakter előzi meg őket. Például:

123            ez egy decimális szám  
!A003        ez egy hexadecimális virtuális cím

Az „**APPLI.TST**” fájl fő szerkezete az alábbi. A fájl **blokkok** listájaként van felépítve. Egy-egy blokk **rekordok** listája. Minden rekordot egy sornyi szöveg ír le. Minden blokk egy fejléccel kezdődik, amely egy sornyi szöveggént szerepel.

Kezdő blokk  
leíró blokkok  
befejező blokk

Egy blokk általános szintaktikája a következő:

```
@ <block_name> <arguments>
#record...
#record...
...
```

Az alkalmazásról szóló fő információt tartalmazó első blokk szerkezete az alábbi:

```
@ISA_SYMBOLS,<appli_crc>
#NAME,<appli_name>,<version>
#DATE,<creation_date>
#SIZE,G=<nbprg>,S=<nbstep>,T=<nbtra>,L=0,P=<nbpro>,V=<nbvar>
#COMMENT,ics triplex isagraf
```

**appli\_crc**.....alkalmazás szimbólumok ellenőrző összege  
**appli\_name**.....az alkalmazás neve  
**version**.....az ISaGRAF workbench verziószáma  
**creation\_date**.....az alkalmazás generálásának dátuma  
**nbprg** .....programok száma  
**nbstep**.....SFC lépések száma  
**nbtra**.....SFC átmenetek száma  
**nbpro** .....használt „C” funkciók száma  
**nbvar**.....változók összes száma

A fájl végét jelző utolsó blokk szerkezete az alábbi:

```
@END_SYMBOLS
```

Az alkalmazás programjait leíró blokk szerkezete az alábbi:

```
@PROGRAMS, <nbprg>
#<va>, <name>
#...
```

**nbprg** .....az ebben a blokkban definiált programok száma

**va** .....a program virtuális címe

**name** .....a program neve

Az alkalmazás SFC lépéseinek leírására használt blokk szerkezete az alábbi:  
Megjegyzendő, hogy minden nem SFC programhoz egy virtuális lépés van definiálva:

```
@STEPS, <nbsteps>
#<va>, <name>, <father>
#...
```

**nbsteps** .....az ebben a blokkban definiált lépések száma

**va** .....a lépés virtuális címe

**name** .....a lépés neve

**father** .....az apa virtuális címe

Az alkalmazás SFC átmeneteinek leírására használt blokk szerkezete az alábbi:

```
@TRANSITIONS, <nbtrans>
#<va>, <name>, <father>
#...
```

**nbtrans** .....az ebben a blokkban definiált átmenetek száma

**va** .....az átmenet virtuális címe

**name** .....az átmenet neve

**father** .....az apa virtuális címe

Az alkalmazás logikai változóit leíró blokk szerkezete az alábbi:

```
@BOOLEANS, <nb_boo>
#<va>, <name>, <attr>, <program>, <eq_false>, <eq_true>
#...
```

és, amennyiben a változók száma meghaladja a 4095-öt:

```
X#(1.<varno>), <name>, <attr>, <program>, <eq_false>, <eq_true>
```

**nb\_boo** .....az ebben a blokkban levő változók száma

**va** .....a változó virtuális címe

**varno** .....a cím tartománya (logikai adattípuson belül)

**name** .....a változó neve

**attr** .....a változó attribútuma

**program** .....a szülőprogram virtuális címe

.....vagy egy globális változónál: „I0000”

**eq\_false** .....hamis értéknél használt karaktersor

**eq\_true** .....igaz értéknél használt karaktersor

Az alkalmazás analóg változóit leíró blokk szerkezete az alábbi:

```
@ANALOGS, <nb_ana>
#<va>, <name>, <attr>, <program>, <format>, <unit>
#...
```

és, amennyiben a változók száma meghaladja a 4095-öt:

```
X# (2.<varno>) , <name>, <attr>, <program>, <format>, <unit>
```

**nb\_ana** .....az ebben a blokkban levő változók száma  
**va** .....a változó virtuális címe  
**varno** .....a cím tartománya (analóg adattípuson belül)  
**name** .....a változó neve  
**attr** .....a változó attribútuma  
**program** .....a szülőprogram virtuális címe  
.....vagy egy globális változónál: „!0000”  
**format** .....= „I”, egy integer változónál  
.....= „F”, valós változónál  
**unit** .....egység karaktersor

Az alkalmazás időzítő változóit leíró blokk szerkezete az alábbi:

```
@TIMERS, <nb_tmr>
#<va>, <name>, <attr>, <program>
#...
```

és, amennyiben a változók száma meghaladja a 4095-öt:

```
X# (3.<varno>) , <name>, <attr>, <program>
```

**nb\_tmr** .....az ebben a blokkban levő változók száma  
**va** .....a változó virtuális címe  
**varno** .....a cím tartománya (időzítés adattípuson belül)  
**name** .....a változó neve  
**attr** .....a változó attribútuma (mindig +X belső):  
**program** .....a szülőprogram virtuális címe  
.....vagy egy globális változónál: „!0000”

Az alkalmazás üzenet-változóit leíró blokk szerkezete az alábbi:

```
@MESSAGES, <nb_msg>
#<va>, <name>, <attr>, <program>, <max_len>
#...
```

és, amennyiben a változók száma meghaladja a 4095-öt:

```
X# (4.<varno>) , <name>, <attr>, <program>, <max_len>
```

**nb\_msg** .....az ebben a blokkban levő változók száma  
**va** .....a változó virtuális címe  
**varno** .....a cím tartománya (üzenet adattípuson belül)

**name** .....a változó neve  
**attr** .....a változó attribútuma  
**program** .....a szülőprogram virtuális címe  
.....vagy egy globális változónál: „I0000”  
**max\_len** .....maximális hossz (deklarált kapacitás)

Az alkalmazás „C” funkcióit leíró blokk szerkezete az alábbi:

```
@USP, <nb_usp>
#<va>, <name>
#...
```

**nb\_usp** .....az ebben a blokkban definiált C funkciók száma  
**va** .....a C funkció virtuális címe  
**name** .....a C funkció neve

Az alkalmazásban használt „C” funkcióblokk előfordulásokat leíró blokk szerkezete az alábbi:

```
@FBINSTANCES, <nb_fb>
#<va>, <inst_name>, <fb_name>
#...
```

**nb\_fb** .....az ebben a blokkban levő C funkcióblokk előfordulások száma  
**va** .....a C funkcióblokk előfordulás virtuális címe  
**inst\_name** .....a C funkcióblokk előfordulás neve  
**fb\_name** .....a referencia C funkcióblokk neve

## A.26.6 Az ISaGRAF „NAGY” (WDL) workbench korlátai

Az ISaGRAF Workbenchben használt tárgyakra vonatkozóan bizonyos korlátozások léteznek. Természetesen a használt számítógép konfigurációjától (rendelkezésre álló memória és lemezkapacitás), valamint az ISaGRAF célrendszer lehetőségeitől (rendelkezésre álló memória, rendelkezésre álló hardver- és szoftverforrások, stb.) függően számos egyéb gyakorlati korlátozás is létezik. A következő számok azokat az abszolút határokat mutatják, amelyeket nem lehet túllépni.

### ☐ **Egy projektnél:**

Tárgy	Maximum	Megjegyzések
Programok	255	fő, al-, és gyermek programok csoportosítása

A hierarchiában levő szintek 20

A Workbenchre telepített projektek számát csak a merevlemezen rendelkezésre álló szabad hely nagysága korlátozza.

### ☐ **Neveknél:**

Név:	Maximum	Megjegyzések
------	---------	--------------

Projekt	8 karakter	
Program	8 karakter	
Változó	16 karakter	+ 60 karakter a megjegyzéshez
Definiált szócímke	16 karakter	
Definiált ekvivalencia	255 karakter	+ 60 karakter a megjegyzéshez
Konverziós táblázat	16 karakter	
Változók listája	16 karakter	
funkció / f.blokk (lib)	8 karakter	ez C funkciókra, C funkcióblokkokra, vagy az IEC nyelveken írt funkciókra vonatkozik
funkció paraméter (lib)	16 karakter	ez C funkciókra, C funkcióblokkokra, vagy az IEC nyelveken írt funkciókra vonatkozik
IO kártya	8 karakter	
IO konfiguráció	8 karakter	
Kártya oem paraméter	16 karakter	
Konverziós funkció	8 karakter	

### ☐ **Szerkesztés (egy programnál):**

<i>Tárgy</i>	<i>Maximum</i>	<i>Megjegyzések</i>
SFC sorok	600	
SFC oszlopok	20	
SFC lépések	4095	az egész projektre, lépések, kezdeti lépések, kezdő és befejező lépések csoportosítása az egész alkalmazásra
SFC átmenetek	4095	
LD/FBD szerkesztés	200 oszlop 2000 sor	ez a szerkesztő terület mérete cella egységekben.
Gyors LD szerkesztés	nincs korlátozás	a korlátozást csak a PC kapacitása szabja meg
IL címkék	251	ugyanabban az IL programban
Szövegszerkesztés	40 KByte	vagy kevesebb, a rendszer konfigurációjától függően

### ☐ **A szótárnál (egy projekthez):**

<i>Tárgy</i>	<i>Maximum</i>	<i>Megjegyzések</i>
Logikai változók	65535	
Analog változók	65535	integer és valós változókat csoportosítva
Időzítők	65535	
Üzenet változók	65535	
Definiált szavak	4095	ugyanabban a listában (megegyező tartomány)
Definiált szavak	255	ugyanabban a programban használva
Konverziós táblázatok	127	az alkalmazásban használva
Pontok egy táblázatban	32	ugyanabban a konverziós táblázatban definiálva

A logikai, analóg, illetve üzenet-változókhoz megadott határértékek a belső, input, és output értékeket összevonva értendők. Ez tartalmazza a rejtett ideiglenes, illetve a fordítók által allokalult változókat is. A közösen szerkesztett változók száma (megegyező típus, megegyező kör) a szótárszerkesztőben nem haladhatja meg a 16000-et. A PC konfigurációjától függően ez a határérték 16000-től kevesebb is lehet. Az alkalmazás nem futtatható V3.21 vagy annál korábbi verziójú ISaGRAF célon, ha az egy típusú változók összes száma meghaladja a 4095-öt. A hálózati címeteket használó standard „Modbus” kapcsolat nem alkalmazható, ha az egy típusú változók száma meghaladja a 4095-öt.

## **IO kapcsolatok:**

<i>Tárgy</i>	<i>Maximum</i>	<i>Megjegyzések</i>
IO Kártyák	256	ugyanahhoz az alkalmazáshoz definiálva (kártyák, vagy összetett berendezések)

Az I/O kártyák száma, beleértve az egyedi kártyákat és összetett berendezéseket is, nem haladhatja meg a 256-ot.

IO csatornák	128	ugyanazon a kártyán
--------------	-----	---------------------

## **Könyvtárakhoz:**

<i>Tárgy</i>	<i>Maximum</i>	<i>Megjegyzések</i>
Funkciók (IEC nyelv)	255	együtt telepítve a könyvtárban
Funkcióblokkok (IEC nyelv)	255	együtt telepítve a könyvtárban
C funkciók	255	együtt telepítve a könyvtárban
C funkcióblokkok	255	együtt telepítve a könyvtárban
funkcióblokkok előfordulások	4095	ugyanolyan típusú funkcióblokkhoz ugyanabban az alkalmazásban
Funkció input paraméterek	31	ez C funkciókra és az IEC nyelveken írt funkciókra vonatkozik
Funkcióblokk paraméterek	32	szabadon elosztva az input- és output paraméterek között. Legalább 1 outputparaméter szükséges.
Konverziós funkció	128	együtt telepítve a könyvtárban
IO konfigurációk	255	együtt telepítve a könyvtárban
IO kártyák	255	együtt telepítve a könyvtárban
Komplex IO berend.	255	együtt telepítve a könyvtárban
Kártya OEM paraméterek	16	

## **B. Nyelvi hivatkozás**

## B.1 Projekt architektúra

Egy ISaGRAF projekt több programozási egységre, ún. **programokra** van felosztva. A projekt programjai egy fastruktúrán belül kapcsolódnak egymáshoz. A programok az **SFC**, **FC (Blokkdiagram)**, **FBD**, **LD**, **ST** vagy **IL** grafikus illetve szöveges nyelvek bármelyikével leírhatók.

### B.1.1 Programok

A **program** egy olyan logikai programozási egység, amely a folyamat **változói** közötti műveleteket írja le. A programok vagy **szekvenciális**, vagy **ciklikus** műveleteket írnak le. A ciklikus programok az egyes célrendszer ciklusok során kerülnek végrehajtásra. A szekvenciális programok végrehajtása az **SFC** vagy az **FC** nyelv dinamikus szabályait követi.

A programok egy fastruktúrájú hierarchiában kapcsolódnak egymáshoz. A hierarchia tetején elhelyezkedő programokat a rendszer aktiválja. Az alprogramokat (a hierarchia alsóbb szintjén) azok apaprogramjaik aktiválják. Egy program az alábbi grafikus vagy szöveges nyelvek bármelyikével leírható:

**Szekvenciális funkciódiagram** (SFC), magas szintű programozáshoz

**Blokkdiagram** (FC), magas szintű programozáshoz

**Funkcióblokk diagram** (FBD), ciklikus komplex műveletekhez

**Létradiagram** (LD), csak logikai műveletekhez

**Strukturált szöveg** (ST), bármilyen ciklikus műveletekhez

**Utasításlista** (IL), alacsony szintű műveletekhez

Egy adott program nem keverhet több nyelvet, kivéve az LD és FBD nyelveket, amelyeket egy diagramon belül lehet kombinálni.

### B.1.2 Ciklikus és szekvenciális műveletek

A programok hierarchiája négy fő **szekción**a, illetve csoportra osztható:

<b>Kezdés</b>	az egyes célciklusok kezdetén végrehajtott programok
<b>Szekvenciális</b>	az SFC vagy FC dinamikus szabályait követő programok
<b>Befejezés</b>	az egyes célciklusok végén végrehajtott programok
<b>Funkciók</b>	nem célrendelt alprogramokból álló készlet

A „**Kezdés**” vagy „**Befejezés**” szekciókban levő programok ciklikus műveleteket írnak le, és nem időfüggők. A „**Szekvenciális**” szekcióban levő programok szekvenciális műveleteket írnak le, ahol az időváltozó konkrétan szinkronizálja az alpműveleteket. A „**Kezdeti**” szekció főprogramjai az egyes futtatási időciklusok kezdetén kerülnek rendszeres végrehajtásra. A „**Befejező**” szekció főprogramjai az egyes futtatási időciklusok végén kerülnek rendszeres végrehajtásra. A „**Szekvenciális**” szekció fő programjai az SFC illetve FC dinamikus szabályoknak megfelelően kerülnek végrehajtásra.

A „**Funkciók**” szekció programjai alprogramok, amelyeket a projektben levő bármely más program meghívhat. A „**Funkció**” szekció egyik programja ennek a szekciónak egy másik programját hívhatja meg.

A szekvenciális szekció fő- és gyermekprogramjait az **SFC** vagy **FC** nyelvvel kell leírni. A ciklikus szekciók (**kezdeti** és **befejező**) programjai nem írhatók le az **SFC** vagy **FC** nyelvvel. Bármelyik szekció bármely programjának lehet egy vagy több alprogramja. A szekvenciális szekció bármelyik programjának lehet egy vagy több **SFC** illetve **FC** gyermekprogramja (saját programozási nyelvének megfelelően). Az alprogramok nem írhatók le az **SFC** vagy **FC** nyelvvel.

A **Kezdeti** szekció programjait általában magas szintű szűrt változók kiépítéséhez az inputeszközökön végrehajtott előzetes műveletek leírására használják. Az ilyen változókat gyakran a **Szekvenciális** szekció programjai használják. A **Befejező** szekció programjait általában az **Szekvenciális** szekció által működtetett változókon az értékek outputeszközökre küldését megelőzően végrehajtott biztonsági műveletek leírására használják.

### B.1.3 Gyermek SFC és FC programok

A szekvenciális szekció bármelyik **SFC** programja vezérelhet más **SFC** programokat. Az ilyen alacsonyabb szintű programokat **gyermek SFC programoknak** nevezik. Egy **gyermek SFC program** egy olyan párhuzamos program, amelyet szülőprogramja elindíthat, megszüntethet, befagyaszthat, vagy újraindíthat. Minden gyermekprogram szülőprogramját az **SFC** nyelvvel kell leírni. Egy gyermekprogramnak lehetnek helyi változói és definiált szavai.

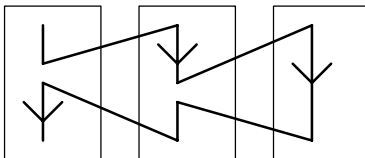
Amikor egy szülőprogram indít el egy gyermek **SFC** programot, akkor egy **SFC vezérjelet** helyez (aktivál) a gyermekprogram minden egyes kezdeti lépésbe. Ezt a parancsot a **GSTART** utasítás írja le. Amikor egy szülőprogram megszüntet egy gyermek **SFC** programot, akkor kitörli a gyermek **lépéseiben** létező összes vezérjelet. Ezt a parancsot a **GKILL** utasítás írja le.

Amikor egy szülőprogram befagyaszt egy gyermek **SFC** programot, akkor felfüggeszti annak végrehajtását. A felfüggesztett gyermekprogram ezt követően a **GRST** utasítás használatával indítható el újra.

A szekvenciális szekció bármelyik **FC** programja vezérelhet más **FC** alprogramokat. Egy **FC** apaprogram egy **FC** alprogram végrehajtása közben blokkolódik (várakozik). Egy apa **FC** programban és annak egyik **FC** alprogramjában nem lehet egyidejűleg műveleteket végezni.

### B.1.4 Funkciók és alprogramok

Egy alprogram vagy egy funkció végrehajtását szülőprogramja irányítja. A szülőprogram végrehajtása az alprogram illetve a funkció befejezéséig felfüggesztődik:



## fő alprogramok

Bármelyik bekezdés bármely programjának lehet egy vagy több alprogramja. Egy alprogramnak csak egyetlen apaprogram a tulajdonosa. Egy gyermekprogramnak lehetnek helyi változói és definiáltjai. Egy alprogram leírásához az **SFC** vagy **FC** kivételével bármilyen nyelv felhasználható. A „**Funkciók**” szekció programjai alprogramok, amelyeket a projektben levő bármely más program meghívhat. A többi alprogramtól eltérően ezek nincsenek egyetlen apaprogramhoz rendelve. A „**Funkció**” szekció egyik programja ennek a bekezdésnek egy másik programját hívhatja meg. Egy funkció helye lehet a Könyvtár.

Figyelmeztetés: Az ISaGRAF rendszer azonban nem támogatja a **rekurzív funkcióhívást**. A „**Funkciók**” bekezdés valamelyik programjának saját maga vagy egyik meghívott alprogramja által történő meghívása futtatási hibát okoz.

Figyelmeztetés: Egy funkció vagy alprogram nem „tárolja” helyi változóinak értékét. Egy funkció vagy alprogram nincs instancionálva, s így nem tud funkcióblokkokat meghívni.

Egy alprogram interfészét valamennyi meghívó illetve visszatérő paraméteréhez egy-egy **típus** és **egyedi név** megadásával egyértelműen definiálni kell. Az **ST** nyelv konvenciójának támogatása érdekében a visszatérő paraméter nevének meg kell egyeznie az alprogram nevével.

A következő táblázat azt mutatja, hogy egy alprogram fő részében hogyan kell a különböző nyelveken beállítani a visszatérő paraméter értékét:

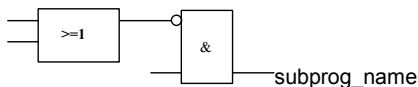
**ST:** jelölje ki a visszatérő paramétert nevének felhasználásával (ugyanaz a név, mint az alprogram neve):

```
subprog_name := <expression>;
```

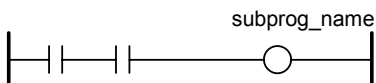
**IL:** a pillanatnyi eredmény értéke (IL regiszter)  
a szekvencia végén a visszatérő paraméterben van eltárolva:

```
LD 10
ADD 20 (* visszatérő paraméter értéke = 30 *)
```

**FBD:** jelölje ki a visszatérő paramétert nevének felhasználásával

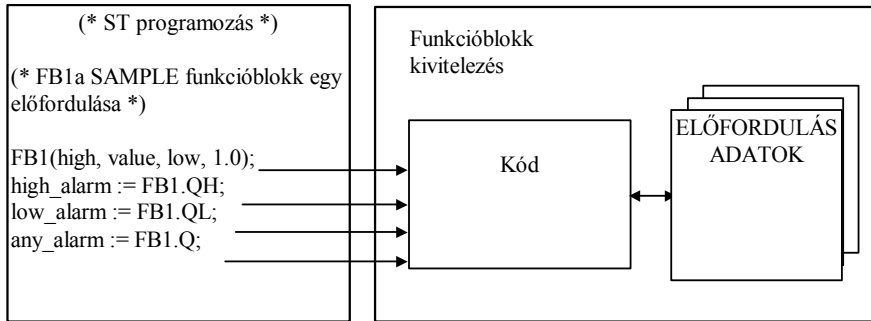


**LD:** használjon egy tekercs szimbólumot a visszatérő paraméter nevével:



## B.1.5 Funkcióblokkok

A funkcióblokkok a következő nyelveket használhatják: LD, FBD, ST vagy IL. A funkcióblokkok instancionálva vannak. Ez azt jelenti, hogy egy funkcióblokk helyi változói minden előforduláshoz be vannak másolva. Egy programban levő blokk meghívásakor valójában a blokk előfordulását hívjuk meg: ugyanaz a kód kerül meghívásra, de a használt adatok az adott előforduláshoz kijelölt adatok. Az előfordulás változóinak értékei az egyik ciklustól a másikig eltárolódnak.



### Figyelmeztetések:

- Az IEC nyelvek egyikével írt funkcióblokk nem tud más funkcióblokkokat meghívni: az instancionáló mechanizmus csak magának a blokknak a helyi változóit kezeli. Az alábbiakban azoknak a standard funkcióblokkoknak a listája látható, amelyeket nem lehet egy IEC funkcióblokkon belül használni:

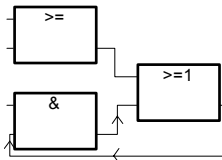
SR, RS, R\_Trig, F\_Trig, SEMA, CTU, CTD, CTUD, TON, TOF, TP, CMP, StackInt, AVERAGE, HYSTER, LIM\_ALARM, INTEGRAL, DERIVATE, BLINK, SIG\_GEN

- Ugyanezért nem használhatók Pozitív vagy Negatív kontaktok illetve tekercsek, vagy Beállító és Visszaállító tekercsek.

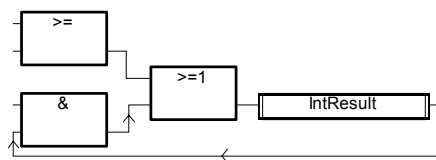
- Az időzítők elindítását és leállítását végző TSTART és TSTOP funkciók nem használhatók egy funkcióblokkban 3.0x célokhoz. ez a 3.20-as és azt követő változatú céloknál működik.

- Ha a funkcióblokkban hurok van szükség, a hurok elvégzése előtt helyi változót kell használni. Lásd az alábbi példát:

Ez nem fog működni:



Ez rendben van:



### B.1.6 Leíró nyelv

Egy program az alábbi grafikus vagy szöveges nyelvek bármelyikével leírható:

**Szekvenciális funkciódiagram** (SFC), magas szintű műveletekhez

**Blokkdiagram** (FC), magas szintű műveletekhez

**Funkcióblokk diagram** (FBD), ciklikus komplex műveletekhez

**Létradiagram** (LD), csak logikai műveletekhez

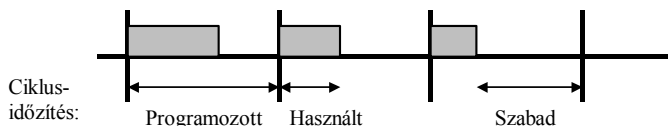
**Strukturált szöveg** (ST), bármilyen ciklikus műveletekhez

**Utasításlista** (IL), alacsony szintű műveletekhez

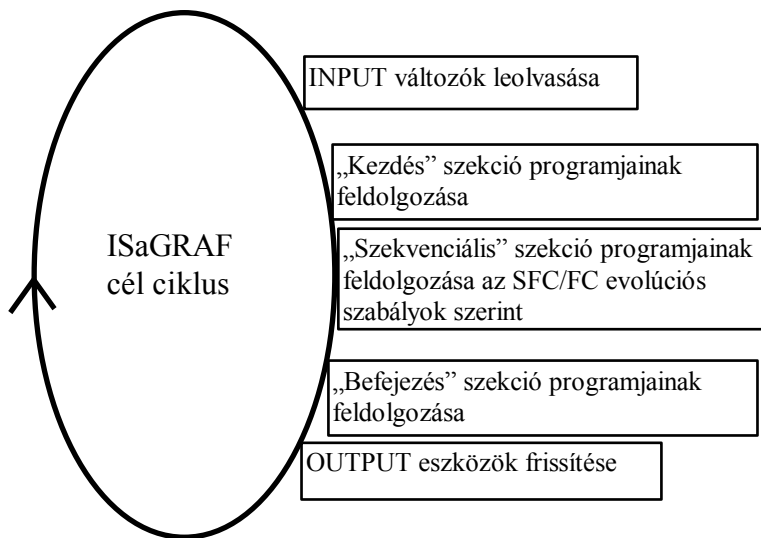
Ugyanaz a program nem keverhet több nyelvet. Egy program leírásához használt nyelv a program létrehozásakor kerül kiválasztásra, és az később nem változtatható meg. Az egyetlen kivétel az, hogy FBD és LD kombinálható egy programban.

### B.1.7 Végrehajtási szabályok

Az ISaGRAF egy **szinkronizált** rendszer. Minden műveletet egy órajel indít el. Az órajel alapidőtartamát ciklusidőzítésnek hívják:



Egy célciklus során feldolgozott alapvető műveletek a következők:



Ez a rendszer a következőket teszi lehetővé:

- annak garantálása, hogy egy inputváltozó egy cikluson belül megtartja ugyanazt az értéket,
- annak garantálása, hogy egy outputeszköz egynél többször nem kerül frissítésre egy cikluson belül,
- különböző programokból történő biztonságos munkavégzés ugyanazon a globális változón,
- a teljes alkalmazás válaszidejének felbecslése és vezérlése.

## B.2 Közös tárgyak

Ezek az ISaGRAF programozási adatbázis fő jellemzői és közös **tárgyai**. Az ilyen tárgyak bármilyen, az **SFC**, **FC**, **FBD**, **LD**, **ST** vagy **IL** nyelvek valamelyikén írt programban használhatók.

### B.2.1 Alapvető típusok

Minden, egy (bármely nyelven írt) programban használt állandót, kifejezést, vagy változót egy típussal kell jellemezni. A grafikus műveleteknél és literális utasításoknál típus koherenciát kell követni. A rendelkezésre álló alapvető programozási tárgyak a következők:

LOGIKAI:	logikai (igaz vagy hamis) érték
ANALÓG:	integer vagy valós (lebegő) folyamatos érték
IDŐZÍTŐ:	időérték
ÜZENET:	karakterstring

Megjegyzés: Az időzítők egy napnál kisebb értékeket tartalmaznak, és nem lehet őket dátumok tárolására használni.

### B.2.2 Állandó kifejezések

Az állandó kifejezések egy típusra vonatkoznak. Ugyanaz az elnevezés nem használható különböző típusú állandó kifejezések képviselésére.

#### B.2.2.1 Logikai állandó kifejezések

Összesen két logikai állandó kifejezés létezik:

<b>IGAZ</b>	1-ás integer értéknek felel meg
<b>HAMIS</b>	0-ás integer értéknek felel meg

Az „Igaz” és „Hamis” kulcsszavak nem érzékenyek a kisbetű/nagybetű beállításra.

#### B.2.2.2 Integer analóg állandó kifejezések

Az integer állandó kifejezések előjeles hosszú integer (32 bit) értékeket jelentenek: - **2147483647**-től **+2147483647**-ig. Az integer analóg állandókat a következő bázisok egyikével lehet kifejezni. Az integer állandóknak egy **előtaggal** kell kezdődniük, amely a használt alapokat azonosítja:

Alap	Előtag	Példa
<b>DECIMÁLIS</b>	(nincs)	<b>-908</b>
<b>HEXADECIMÁLIS</b>	<b>"16#"</b>	<b>16#1A2B3C4D</b>
<b>OKTÁLIS</b>	<b>"8#"</b>	<b>8#1756402</b>
<b>BINÁRIS</b>	<b>"2#"</b>	<b>2#1101_0001_0101_1101</b>

A számjegyek különválasztására az aláhúzás karaktert („\_”) lehet használni. Ennek nincs külön jelentősége, csak az állandó kifejezés olvashatóságának megkönnyítésére szolgál.

### B.2.2.3 Valós analóg állandó kifejezések

A valós analóg konstans kifejezések vagy **decimális**, vagy **tudományos** formában írhatók. A **tizedespont** („.”) az integer és decimális részeket választja el egymástól. Egy valós állandó kifejezést a tizedespont használatával kell megkülönböztetni egy integer állandó kifejezéstől. A tudományos forma az „E” vagy „F” betűt használja a **mantissza** és a **hatványkitevő** különválasztására. Egy valós tudományos kifejezés kitevő részének egy **-37** és **+37** közötti előjeles integer értéknek kell lennie.

<b>3.14159</b>	<b>-1.0E+12</b>
<b>+1.0</b>	<b>1.0F-15</b>
<b>-789.56</b>	<b>+1.0E-37</b>

Az „**123**” kifejezés nem jelent valós állandó kifejezést. Ennek helyes valós formája „**123.0**”.

### B.2.2.4 Időzítő állandó kifejezések

Az időzítő állandó kifejezések **0 másodperc** és **23h59m59s999ms** közötti időértékeket képviselnek. A megengedett legalacsonyabb egység az ezredmásodperc. Az állandó kifejezésekben használt standard időegységek a következők:

<b>Óra</b>	Az órák száma után a „h” betűnek kell állnia
<b>Perc</b>	A percek száma után az „m” betűnek kell állnia
<b>Másodperc</b>	A másodpercek száma után a „s” betűnek kell állnia
<b>Ezredmásodperc</b>	Az ezredmásodpercek száma után az „ms” betűnek kell állnia

Az időállandó kifejezésnek a „**T#**” vagy „**TIME#**” előtaggal kell kezdődnie. Az előtagok és egységjelző betűk nem érzékenyek a kisbetű/nagybetű beállításra. Előfordulhat, hogy egyes mértékegységek nem jelennek meg. Az alábbiakban az időzítés állandó kifejezések példái láthatók:

<b>T#1H450MS</b>	1 óra, 450 ezredmásodperc
<b>time#1H3M</b>	1 óra, 3 perc

A „0” kifejezés nem egy időértéket jelent, hanem egy analóg állandót.

### B.2.2.5 Üzenet karaktorsor állandó kifejezések

A karaktorsor vagy üzenet állandó kifejezések karaktorsorokat képviselnek. A karakterek előtt és után egy-egy idézőjelnek kell állni. Például:

**'EZ EGY ÜZENET'**

**Figyelmeztetés:** A „” aposztróf karakter nem használható egy karaktorsor állandó kifejezésen belül. Egy karaktorsor állandó kifejezést a program forráskód egy sorában kell kifejezni. Hossza nem haladhatja meg a 255 karaktert, a szóközőket is beleértve.

Az üres karaktorsor állandó kifejezést két hiányjel képviseli, amelyek között nincs szóköz vagy tabulátor karakter:

**” (\* ez egy üres karaktorsor \*)**

A dollárjel („\$”) egy speciális karakter, amelyet más speciális karakterek követnek, egy karaktorsor állandó kifejezésben egy nem nyomtatandó karakter jelölésére használható:

Szekvencia	Jelentése	ASCII (hexa)	Example
\$\$	'\$' karakter	16#24	'I paid \$\$5 for this'
\$'	hiányjel	16#27	'Enter '\$Y\$' for YES'
\$L	soremelés	16#0a	'next \$L line'
\$R	kocsi vissza	16#0d	' Ilo \$R He'
\$N	új sor	16#0d0a	'This is a line\$N'
\$P	új oldal	16#0c	'lastline \$P first line'
\$T	tabulálás	16#09	'name\$Tsize\$Tdate'
\$hh (*)	bármilyen karakter	16#hh	'ABCD = \$41\$42\$43\$44'

(\*) A "hh" a kifejezett karakter ASCII kódjának hexadecimális értéke.

## B.2.3 Változók

A változók lehetnek **LOKÁLISAK** egy programra nézve, vagy lehetnek **GLOBALISAK**. A lokális változókat csak egy program használhatja. A globális változók a projekt bármelyik programjában használhatók. A változók neveinek a következő szabályoknak kell eleget tenniük:

- a név nem haladhatja meg a **16** karaktert
- az első karakternek **betűnek** kell lennie
- a többi karakter lehet **betű**, **számjegy**, vagy aláhúzás karakter

### B.2.3.1 Fenntartott kulcsszavak

Az alábbiakban a fenntartott kulcsszavak listája látható. Ilyen azonosítókat nem lehet egy program, egy változó, vagy egy „C” funkció illetve funkcióblokk elnevezésére használni:

A	ANA, ABS, ACOS, ADD, ANA, AND, AND_MASK, ANDN, ARRAY, ASIN, AT, ATAN,
B	BCD_TO_BOOL, BCD_TO_INT, BCD_TO_REAL, BCD_TO_STRING, BCD_TO_TIME, BOO, BOOL, BOOL_TO_BCD, BOOL_TO_INT, BOOL_TO_REAL, BOOL_TO_STRING, BOOL_TO_TIME, BY, BYTE,
C	CAL, CALC, CALC_N, CALN, CALNC, CASE, CONCAT, CONSTANT, COS,
D	DATE, DATE_AND_TIME, DELETE, DINT, DIV, DO, DT, DWORD,

E	ELSE, ELSIF, EN, END_CASE, END_FOR, END_FUNCTION, END_IF, END_PROGRAM, END_REPEAT, END_RESSOURCE, END_STRUCT, END_TYPE, END_VAR, END_WHILE, ENO, EQ, EXIT, EXP, EXPT,
F	FALSE, FEDGE, FIND, FOR, FUNCTION,
G	GE, GFREEZE, GKILL, GRST, GSTART, GSTATUS, GT,
I	IF, INSERT, INT, INT_TO_BCD, INT_TO_BOOL, INT_TO_REAL, INT_TO_STRING, INT_TO_TIME,
J	JMP, JMPC, JMPCN, JMPN, JMPNC,
L	LD, LDN, LE, LEFT, LEN, LIMIT, LINT, LN, LOG, LREAL, LT, LWORD,
M	MAX, MID, MIN, MOD, MOVE, MSG, MUL, MUX,
N	NE, NOT,
O	OF, ON, OPERATE, OR, OR_MASK, ORN,
P	PROGRAM
R	R, REDGE, READ_ONLY, READ_WRITE, REAL, REAL_TO_BCD, REAL_TO_BOOL, REAL_TO_INT, REAL_TO_STRING, REAL_TO_TIME, REDGE, REPEAT, REPLACE, RESSOURCE, RET, RETAIN, RETC, RETCN, RETN, RETNC, RETURN, RIGHT, ROL, ROR,
S	S, SEL, SHL, SHR, SIN, SINT, SQRT, ST, STN, STRING, STRING_TO_BCD, STRING_TO_BOOL, STRING_TO_INT, STRING_TO_REAL, STRING_TO_TIME, STRUCT, SUB, SYS_ERR_READ, SYS_ERR_TEST, SYS_INITALL, SYS_INITANA, SYS_INITBOO, SYS_INITTMR, SYS_RESTALL, SYS_RESTANA, SYS_RESTBOO, SYS_RESTTMR, SYS_SAVALL, SYS_SAVANA, SYS_SAVBOO, SYS_SAVTMR, SYS_TALLOWED, SYS_TCURRENT, SYS_TMAXIMUM, SYS_TOVERFLOW, SYS_TRESET, SYS_TWRITE, SYSTEM,
T	TAN, TASK, THEN, TIME, TIME_OF_DAY, TIME_TO_BCD, TIME_TO_BOOL, TIME_TO_INT, TIME_TO_REAL, TIME_TO_STRING, TMR, TO, TOD, TRUE, TSTART, TSTOP, TYPE,
U	UDINT, UINT, ULINT, UNTIL, USINT,
V	VAR, VAR_ACCESS, VAR_EXTERNAL, VAR_GLOBAL, VAR_IN_OUT, VAR_INPUT, VAR_OUTPUT,
W	WHILE, WITH, WORD,
X	XOR, XOR_MASK, XORN

Minden, aláhúzás karakterrel ('\_') kezdődő kulcsszó belső kulcsszó, és nem használható szöveges utasításokban.

### B.2.3.2 Közvetlenül képviselt változók

Az ISaGRAF lehetővé teszi a **közvetlenül képviselt változók** használatát a programok forrásaiban, egy szabad csatorna képviselésére. A szabad csatornák azok, amelyek nincsenek egy deklarált I/O változóhoz kapcsolva. Egy közvetlen képviselésű változó azonosítója mindig a „%” karakterrel kezdődik.

Az alábbiakban egy egyedi kártya egyik csatornájának közvetlen képviselőtű változójának elnevezési konvenciói szerepelnek. Az „s” a kártya kártyahelyének száma. A „c” a csatorna száma.

%IXs.c egy logikai input kártya szabad csatornája

%IDS.c egy integer input kártya szabad csatornája

%ISs.c egy üzenet input kártya szabad csatornája

%QXs.c egy logikai output kártya szabad csatornája  
 %QDs.c egy integer output kártya szabad csatornája  
 %Qss.c egy üzenet output kártya szabad csatornája

Az alábbiakban egy komplex berendezés egyik csatornája közvetlen képviselőtű változójának elnevezési konvenciói szerepelnek. Az „s” a berendezés kártyahelyének száma. A „b” a komplex berendezésen belüli egyedi kártya indexe. A „c” a csatorna száma.

%IXs.b.c egy logikai input kártya szabad csatornája  
 %IDs.b.c egy integer input kártya szabad csatornája  
 %ISs.b.c egy üzenet input kártya szabad csatornája  
 %QXs.b.c egy logikai output kártya szabad csatornája  
 %QDs.b.c egy integer output kártya szabad csatornája  
 %Qss.b.c egy üzenet output kártya szabad csatornája

Az alábbiakban példák láthatók:

%QX1.6 Az 1. számú kártya 6. csatornája (logikai output)  
 %ID2.1.7 A 2. számú berendezésben levő 1. számú kártya 7. csatornája (integer input)

Egy közvetlen képviselőtű változónak nem lehet „valós” adattípusa.

### B.2.3.3 Logikai változók

A „Booleusi típusú” **logikait** jelent. Az ilyen változók a következő logikai értékek egyikét vehetik fel: **IGAZ** vagy **HAMIS**. A logikai változókat általában logikai kifejezésekben használják. A logikai változók a következő **attribútumok** egyikével rendelkezhetnek:

**Belső:** a program által frissített memóriaváltozó  
**Konstans:** csak olvasható belső változó (kezdeti értékkel)  
**Input:** egy inputeszközhöz csatlakoztatott változó (a rendszer frissíti)  
**Output:** egy outputeszközhöz csatlakoztatott változó

**Figyelmeztetés:** Egy logikai változó deklarálásánál a hibakeresés közben az „igaz” és „hamis” értékek felváltására karaktorsorokat lehet definiálni. Ezek a karaktorsorok nem használhatók a programban, csak akkor, ha azokat az adott nyelvhez „**definiált szavakként**” vitték be.

### B.2.3.4 Analóg változók

Az analóg **folyamatosat** jelent. Az ilyen változók előjellel ellátott integer vagy valós (lebegő) értékekkel rendelkeznek. Egy analóg változóhoz a következő formátumok állnak rendelkezésre:

**Integer** 32 bites előjellel ellátott integer: **-2147483647**-től **+2147483647**-ig  
**Valós** standard IEEE 32 bites lebegő érték (egyszeres pontosság)  
 1 előjel bit + 23 mantissza bit + 8 kitevő bit

A VALÓS analóg kitevő értéke nem lehet kevesebb mint **-37**, illetve nem lehet nagyobb mint **+37**. Az analóg változók a következő **attribútumok** egyikével rendelkezhetnek:

<b>Belső</b>	a program által frissített memóriaváltozó
<b>Konstans:</b>	csak olvasható belső változó (kezdeti értékkel)
<b>Input</b>	egy inputeszközhöz csatlakoztatott változó (a rendszer frissíti)
<b>Output</b>	egy outputeszközhöz csatlakoztatott változó

Megjegyzés: Amikor egy I/O eszközhez egy valós változó van csatlakoztatva, a megfelelő I/O meghajtó az ekvivalens integer értéket működteti.

Figyelmeztetés: Az integer és valós analóg változók vagy állandó kifejezések nem keverhetők ugyanabban az analóg kifejezésben.

### B.2.3.5 Időzítő változók

Az időzítő **órát** vagy **számlálót** jelent. Az ilyen változóknak időértékeik vannak, és általában időkifejezésekben használják őket. Egy időzítés értéke nem haladhatja meg a **23h59m59s999ms**-t, és nem lehet negatív érték. Az időzítés változók 32 bites szavakban vannak tárolva. A belső képviselő pozitív számú ezredmásodpercekben történik. A logikai változók a következő **attribútumok** egyikével rendelkezhetnek:

<b>Belső</b>	a program által kezelt memória változó, amelyet az ISaGRAF rendszer frissít
<b>Konstans:</b>	csak olvasható belső változó (kezdeti értékkel)

Figyelmeztetés: Az időzítés változóknak nem lehet INPUT vagy OUTPUT attribútuma.

Az időzítés változókat az ISaGRAF rendszer automatikusan frissíteni tudja. Amikor egy időzítés **aktív**, akkor annak értéke automatikusan növekszik a célrendszer valós idejű órájának megfelelően. Egy időzítés vezérléséhez a következő **ST** nyelvű utasítások használhatók:

<b>TSTART</b>	elindítja az időzítés automatikus frissítését
<b>TSTOP</b>	megállítja az időzítés automatikus frissítését

### B.2.3.6 Üzenet karaktorsor változók

Az üzenet vagy karaktorsor változók karaktorsorokat tartalmaznak. A karaktorsor hossza a folyamat műveletei közben változhat. Egy üzenet változó hossza nem haladhatja meg a változó deklarálásakor meghatározott kapacitást (maximális hossz). Az üzenet kapacitás maximális értéke 255 karakter. Az üzenet változók a következő **attribútumok** egyikével rendelkezhetnek:

<b>Belső</b>	a program által frissített memóriaváltozó
<b>Konstans:</b>	csak olvasható belső változó (kezdeti értékkel)
<b>Input</b>	egy inputeszközhöz csatlakoztatott változó (a rendszer frissíti)
<b>Output</b>	egy outputeszközhöz csatlakoztatott változó

A karaktorsor változók a standard ASCII táblázat (**0** és **255** közötti ASCII kód) bármelyik karakterét tartalmazhatják. Egy karaktorsorban a nulla karakter is létezhet. A standard

ISaGRAF könyvtár bizonyos „C” funkciói nem kezelik helyesen azokat az üzeneteket, amelyek nulla (0) karaktereket tartalmaznak.

## B.2.4 Megjegyzések

Megjegyzéseket szabadon lehet beilleszteni a literális nyelvekbe, mint amilyen az **ST** és **IL** nyelv. Egy megjegyzésnek a „(” speciális karakterekkel kell kezdődnie, és a „)” karakterekkel kell befejeződnie. Megjegyzések akárhová beilleszthetők egy **ST** programban, és azok egynél több sorba is írhatók.

Az alábbiakban a megjegyzések példái láthatók:

```
counter := ival; (* kijelöli a fő számlálót *)
(* ez egy megjegyzés, amely
két sorban van kifejezve *)
c := counter (* a megjegyzéseket bárhová el lehet helyezni *) + base_value + 1;
```

Összefonódott megjegyzések nem használhatók. Ez azt jelenti, hogy a „(” karakterek nem használhatók egy megjegyzésen belül.

Figyelmeztetés: Az **IL** nyelv csak egy utasítássor utolsó komponenseként fogadja el a megjegyzést.

## B.2.5 Definiált szavak

Az ISaGRAF rendszer lehetővé teszi az állandó kifejezések, igaz és hamis kifejezések, kulcsszavak vagy összetett **ST** kifejezések átdefiníálását. Ennek eléréséhez a megfelelő kifejezésnek egy **azonosító** nevet kell adni. Például:

<b>YES</b>	egyenlő	<b>TRUE</b>
<b>PI</b>	egyenlő	<b>3.14159</b>
<b>OK</b>	egyenlő	<b>(auto_mode AND NOT (alarm))</b>

Amikor egy ilyen ekvivalencia van definiálva, akkor annak **azonosítója** egy **ST** programban bárhol használható a csatolt kifejezés lecserélésére. Az alábbiakban a definiálásokat használó **ST** programozás példája látható:

```
If OK Then
    angle := PI / 2.0;
    isdone := YES;
End_if;
```

A definiált szavak egy programra nézve lehetnek **LOKÁLISAK**, **GLOBALISAK**, vagy **KÖZÖSEK**.

A lokális definiált szavakat csak egy program használhatja.

A globális definiált szavak a projekt bármelyik programjában használhatók.

A közös definiált szavak bármelyik projekt bármelyik programjában használhatók.

Megjegyzendő, hogy a közös definiált szavak az Archívumrendezőtől külön tárolhatók.

**Figyelmeztetés:** Amikor ugyanaz az azonosító kétszer van definiálva eltérő **ST** ekvivalenciákkal, az utolsó definiált kifejezés lesz használva. Például:

Definiálni:	<b>OPEN</b>	egyenlő	<b>FALSE</b>
	<b>OPEN</b>	egyenlő	<b>TRUE</b>
jelentése:	<b>OPEN</b>	egyenlő	<b>TRUE</b>

A definiált szavak elnevezéseinek a következő szabályoknak kell eleget tenniük:

- a név nem haladhatja meg a **16** karaktert
- az első karakternek **betűnek** kell lennie
- a többi karakter lehet **betű**, **számjegy**, vagy aláhúzás („\_”) karakter

**Figyelmeztetés:** Egy definiált szó nem használhat egy definiált szót saját definíciójában, így például az alábbi nem engedhető meg:

<b>PI</b>	egyenlő	<b>3.14159</b>
<del><b>PI2</b></del>	<del>egyenlő</del>	<del><b>PI*2</b></del>

a teljes ekvivalenciát állandók vagy változók és műveletek használatával kell megírni:

<b>PI2</b>	egyenlő	<b>6.28318</b>
------------	---------	----------------

## B.3 Az SFC nyelv

A Szekvenciális funkciódiagram („Sequential Function Chart”; SFC) egy **grafikus** nyelv, amely **szekvenciális** műveletek leírására használatos. A folyamatot jól definiált **lépések** készlete képviseli, amelyek **átmenetekkel** kapcsolódnak egymáshoz. Minden átmenethez egy **logikai feltétel** van csatolva. A lépéseken belüli **műveletek** egyéb nyelvek (**ST**, **IL**, **LD** és **FDB**) alkalmazásával vannak részletezve.

### B.3.1 Az SFC diagram fő formátuma

Egy SFC program **lépések** és **átmenetek** grafikus együttese, amelyeket **irányított kapcsolatok** kötnek össze. A divergenciákat és konvergenciákat többszörös csatlakozási kapcsolatok képviselik. A komplett program bizonyos részei a diagramban külön lehetnek választva, és ezeket egyetlen szimbólum képviselheti, amelyet **makró lépésnek** neveznek. Az SFC alapvető **grafikus szabályai** a következők:

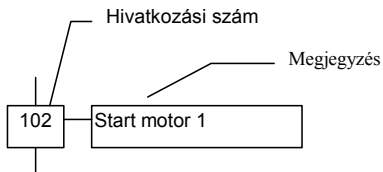
- Egy lépést nem követhet egy másik lépés
- Egy átmenetet nem követhet egy másik átmenet

### B.3.2 Alapvető SFC komponensek

Az SFC nyelv alapvető komponensei (grafikus szimbólumok) a következők: lépések és kezdeti lépések, átmenetek, irányított kapcsolatok, és egy lépésre történő ugrások.

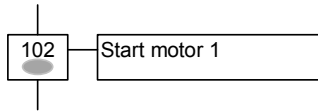
#### B.3.2.1 Lépések és kezdeti lépések

Egy lépést egyetlen **négyzet** képvisel. Minden lépésre **hivatkozás** történik egy számmal, amely a lépés négyzet alakú szimbólumába van írva. A lépés fő leírása egy téglalapba van írva, amely a lépés szimbólumához van kapcsolva. Ez a leírás egy **szabad megjegyzés** (nem a programozási nyelv része). A fenti információt a lépés **1. szintjének** nevezik:

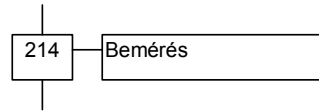


A futtatáskor egy **vezérlőjel** jelzi, hogy a lépés **aktív**:

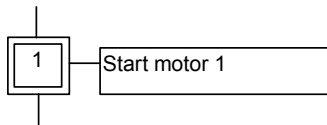
Aktív lépés:



Inaktív lépés:



Egy SFC program **kezdeti helyzete kezdeti lépésekkel** van kifejezve. Egy kezdeti lépésnek **dupla keretes** grafikus szimbóluma van. A program elindításakor mindegyik kezdeti lépésre automatikusan egy vezérlőjel kerül elhelyezésre.

**Kezdeti lépés:**

Egy SFC programnak **legalább egy** kezdeti lépést kell tartalmaznia.

Ezek egy lépés attribútumai. Ilyen mezők bármelyik másik nyelvben is használatosak lehetnek.

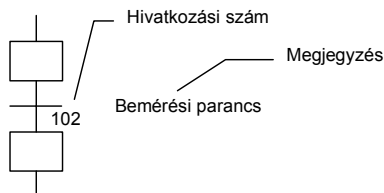
**GSnnn.x** ..... a lépés tevékenysége (logikai érték)

**GSnnn.t** ..... a lépés tevékenység időtartama (időérték)

(ahol **nnn** a lépés hivatkozási száma)

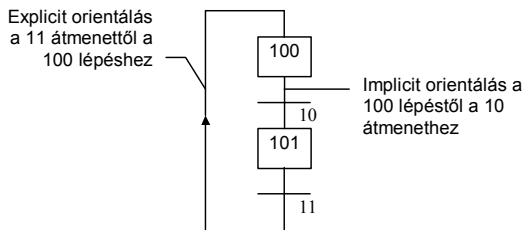
### B.3.2.2 Átmenetek

Az átmeneteket egy, a csatlakozási kapcsolatot keresztező kis vízszintes vonal jelzi. Minden átmenetre egy számmal történik **hivatkozás**, az átmenet szimbóluma mellett van feltüntetve. Az átmenet fő leírása az átmenet szimbólumának jobb oldalára van írva. Ez a leírás egy **szabad megjegyzés** (nem a programozási nyelv része). A fenti információt az átmenet **1. szintjének** nevezik:



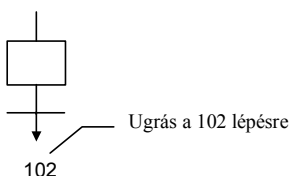
### B.3.2.3 Irányított kapcsolatok

A lépések és átmenetek összekapcsolására egyszeres vonalak szolgálnak. Ezek irányított (orientált) kapcsolatok. Amikor az irányítás nincs kifejezetten megadva, akkor a kapcsolat felülről lefelé orientálódik.

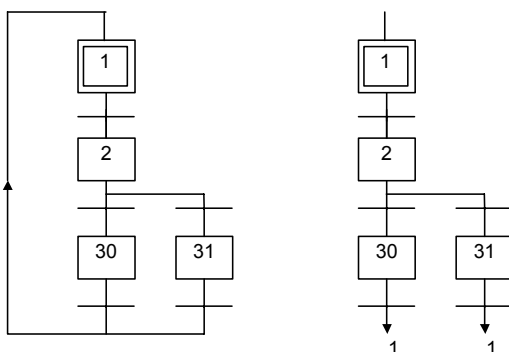


### B.3.2.4 Ugrás egy lépésre

Az Ugrás szimbólumok egy átmenettől egy lépéshez való csatlakozási kapcsolat jelzésére használhatók, anélkül, hogy meg kellene rajzolni egy vonalat. Az ugrás szimbólumot a rendeltetési lépés számával kell hivatkoztatni:



Az ugrás szimbólumot nem szabad egy lépéstől egy átmenetbe való kapcsolat jelképezésére használni. Példák az ugrásokra a következő diagramok egyenértékűek:

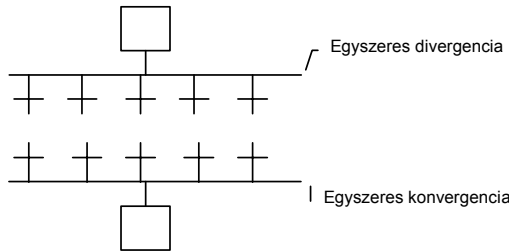


### B.3.3 Divergenciák és konvergenciák

A divergenciák **többszörös csatlakozási kapcsolatok** az egyik SFC szimbólumtól (lépéstől vagy átmenettől) számos egyéb SFC szimbólumhoz. A Konvergenciák többszörös csatlakozási kapcsolatok egynél több SFC szimbólumtól egy másik szimbólumhoz. A divergenciák és konvergenciák lehetnek egyszeresek vagy kétszeresek.

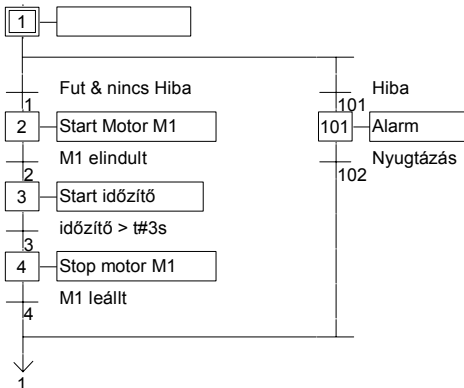
### B.3.3.1 Egyszeres divergenciák

Egy egyszeres divergencia egy többszörös kapcsolat egy lépéstől számos átmenethez. Ez lehetővé teszi az aktív vezérlőjel átadását számos elágazás egyikéhez. Az egyszeres divergencia egy többszörös kapcsolat egy lépéstől számos átmenethez. Az egyszeres konvergenciát általában egy egyszeres divergencián indított SFC elágazások csoportosítására használják. Az egyszeres divergenciákat és konvergenciákat egyszeres vízszintes vonalak jelképezik.



**Figyelmeztetés:** Egy egyszeres divergencia kezdetén a különböző átmenetekhez csatolt feltételek **nem implicit módon kizárólagosak**. A kizárólagosságot explicit módon részletezni kell az átmenetek feltételeiben annak biztosítására, hogy a futtatáskor egy elágazásba csak egyetlen vezérlőjel jusson el. Az alábbiakban az egyszeres divergencia és konvergencia példája látható:

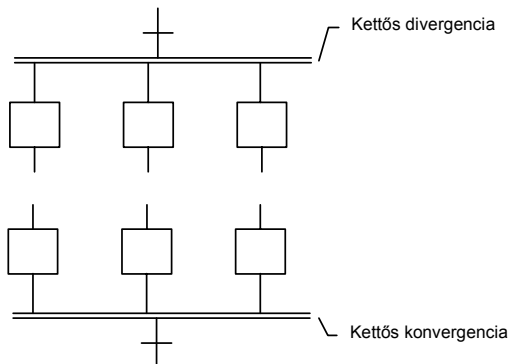
#### (\* SFC program egyszeres divergenciával és konvergenciával \*)



### B.3.3.2 Kettős divergenciák

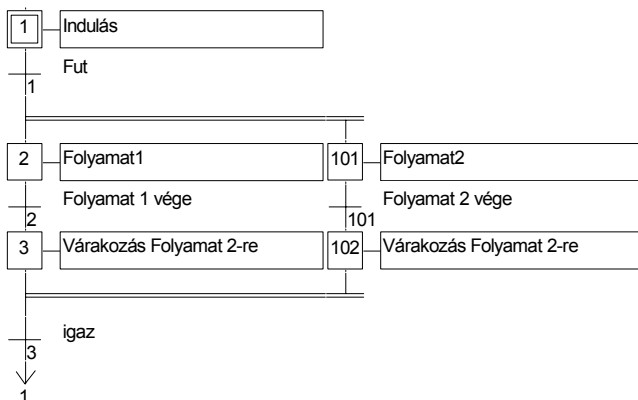
Egy kétszeres divergencia egy többszörös kapcsolat egy átmenettől számos lépéshez. Ez a folyamat párhuzamos műveleteinek felel meg. A kétszeres divergencia egy többszörös kapcsolat számos lépéstől ugyanahhoz az átmenethez. A kétszeres konvergenciát általában

egy kétszeres divergencián indított SFC elágazások csoportosítására használják. A kétszeres divergenciákat és konvergenciákat kettős vízszintes vonalak jelképezik.



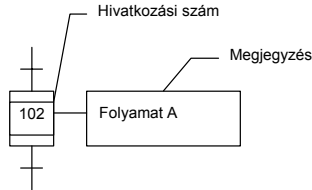
Kettős divergencia és konvergencia példája:

(\* SFC program kétszeres divergenciával és konvergenciával \*)



### B.3.4 Makró lépések

Egy makró lépés lépések és átmenetek egyedi csoportjának egyedi képviselője. A makró lépés fő részének ugyanabban az SFC programban máshol van beadva. Ez a fő SFC diagramban egyetlen szimbólumként jelenik meg. Ez a makró lépéshez használt szimbólum:



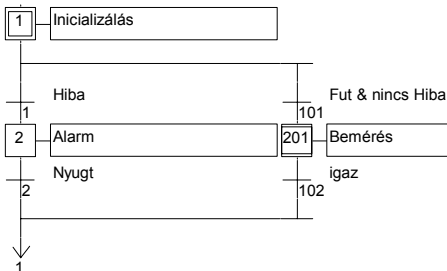
A makró lépés szimbólumába írt hivatkozási szám a makró lépés fő részében levő első lépés hivatkozási száma. A makró lépés fő részének egy **kezdeti lépéssel** kell kezdődnie, és egy befejező lépéssel kell befejeződnie. A diagramnak önállóan kell lennie. Egy kezdeti lépésnek nincs felső kapcsolata (nincs visszafelé történő átmenete). Egy befejező lépésnek nincs alsó kapcsolata (nincs előrefelé történő átmenete). Egy makró lépés szimbóluma elhelyezhető egy másik makró lépés fő részébe.

**Figyelmeztetés:** Mivel a makró lépés lépések és átmenetek **egyedi** készlete, ezért ugyanaz a makró lépés nem használható egynél többször egy SFC programban.

Egy makró lépés példája

(\* SFC program makró lépéssel \*)

(\* Fő diagram: \*)



(\* A makró lépés fő része \*)



### B.3.5 A lépéseken belüli tevékenységek

Egy SFC lépés **2. szintje** a lépés tevékenység során elvégzett **műveletek** részletes leírása. Ez a leírás az **SFC literális jellemzők** és más nyelvek, pl. Strukturált szöveg (ST) használatával történik. A műveletek alapvető típusai a következők:

- Logikai műveletek
- ST-ben programozott impulzus műveletek
- ST-ben programozott nem tárolt műveletek
- SFC műveletek

Ugyanabban a lépésben több művelet (ugyanolyan, vagy eltérő típusú) írható le. A valamelyik más nyelv használatát lehetővé tevő speciális tulajdonságok a következők:

- Alprogramok hívása
- Utasításlista (IL) nyelvi konvenció

### B.3.5.1 Logikai műveletek

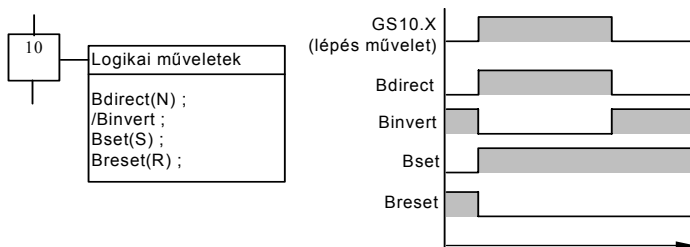
A logikai műveletek egy logikai változót a lépés műveletével jelölnek ki. A logikai változó lehet output vagy belső. Ez egy lépés művelet leindulásakor vagy megállásakor mindannyiszor kijelölésre kerül. Az alapvető logikai műveletek szintaxisai a következők:

<b>&lt;boolean_variable&gt; (N) ;</b>	a lépés művelet jelet kijelöli a változóhoz
<b>&lt;boolean_variable&gt; ;</b>	ugyanaz a hatása (az N attribútum nem kötelező)
<b>/ &lt;boolean_variable&gt; ;</b>	a lépés művelet jel tagadását jelöli ki a változóhoz

Egyéb lehetőségek is rendelkezésre állnak egy logikai változó beállításához vagy visszaállításához, amikor a lépés aktívvá válik. A beállítási és visszaállítási logikai műveletek szintaxisai a következők:

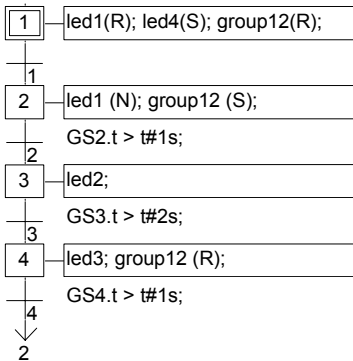
<b>&lt;boolean_variable&gt; (S) ;</b>	A változót IGAZ-ra állítja be, amikor a lépés aktivitás jele IGAZ-zá válik
<b>&lt;boolean_variable&gt; (R) ;</b>	A változót HAMIS-ra állítja vissza, amikor a lépés aktivitás jele IGAZ-zá válik

A logikai változónak OUTPUT-nak vagy BELSŐ-nek kell lennie. A következő SFC programozás a következő viselkedést idézi elő:



Logikai műveletek példája:

## (\* LOGIKAI műveleteket használó SFC program \*)

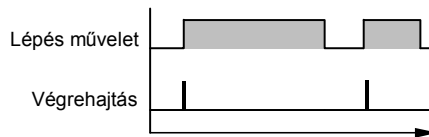


## B.3.5.2 Impulzus műveletek

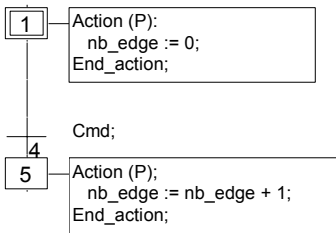
Egy impulzus művelet ST vagy IL utasítások listája, amelyek csak **egyszer** vannak végrehajtva, egy lépés **aktiválásakor**. Az utasítások a következő SFC szintaxis szerint vannak megírva:

**ACTION (P) :**  
(\* ST utasítások \*)  
**END\_ACTION;**

A következőkben egy impulzus művelet eredményei láthatók:



Impulzus művelet példája:

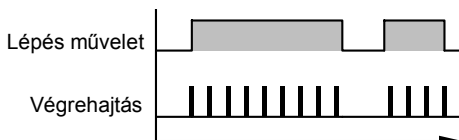


### B.3.5.3 Nem tárolt műveletek

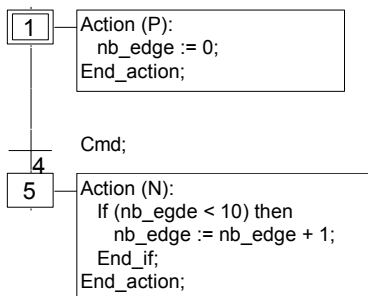
Egy nem tárolt (normál) művelet ST vagy IL utasítások listája, amelyek a lépés egész **aktív** periódusa alatt **minden ciklusnál** végre vannak hajtva. Az utasítások a következő SFC szintaxis szerint vannak megírva:

```
ACTION (N) :
    (* ST utasítások *)
END_ACTION;
```

A következőkben egy nem tárolt művelet eredményei láthatók:



Nem tárolt művelet példája:



### B.3.5.4 SFC műveletek

Egy SFC művelet egy gyermek SFC szekvencia, amely a lépés műveleti jelnek megfelelően van elindítva vagy megállítva. Egy SFC műveletnek lehet **N** (Nem tárolt), **S** (Beállított), vagy **R** (Visszaállított) minősítője. Az alapvető SFC műveletek szintaxisai a következők:

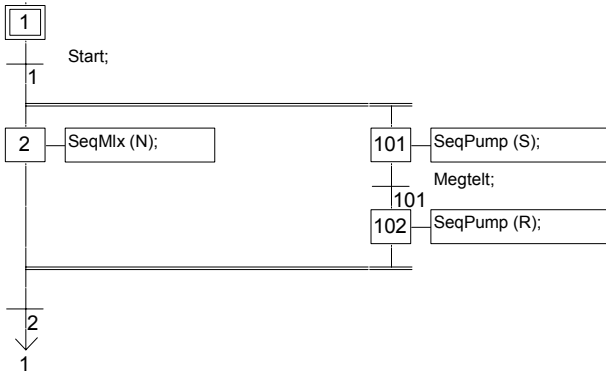
<b>&lt;child_prog&gt; (N);</b>	elindítja a gyermek szekvenciát amikor a lépés aktívva válik, és leállítja a gyermek szekvenciát amikor a lépés inaktívva válik
<b>&lt;child_prog&gt; ;</b>	ugyanaz a hatása (az N attribútum nem kötelező)
<b>&lt;child_prog&gt; (S);</b>	elindítja a gyermek szekvenciát, amikor a lépés aktívva válik. Semmi sem történik akkor, amikor a lépés inaktívva válik
<b>&lt;child_prog&gt; (R);</b>	leállítja a gyermek szekvenciát, amikor a lépés aktívva válik. Semmi sem történik akkor, amikor a lépés inaktívva válik

A műveletként specifikált SFC szekvenciának a jelenleg szerkesztett program **gyermek SFC programjának** kell lennie. Megjegyzendő, hogy az **S** (Beállítás) vagy **R** (Visszaállítás)

minősítők használata egy SFC művelénél ugyanazzal a hatással jár, mint a **GSTART** és **GKILL** utasítások egy **ST** impulzus műveletbe programozva.

Az alábbiakban egy SFC művelet példája látható: A fő SFC program neve **Father** (Apa). Ennek két SFC gyermeke van, amelyek neve **SeqMlx** és **SeqPump**. Az apa SFC program SFC programozása a következő:

(\* SFC műveleteket használó SFC program \*)



### B.3.5.5 Funkciók és funkcióblokkok meghívása egy műveletből

Alprogramok, funkciók, vagy funkcióblokkok (ST, IL, LD vagy FBD nyelven írva), illetve „C” funkciók és „C” funkcióblokkok közvetlenül meghívhatók egy SFC művelet ablakból, a következő szintaxis alapján:

Alprogramoknál, funkcióknál, és „C” funkcióknál:

```

ACTION (P) :
    result := sub_program ( ) ;
END_ACTION;
  
```

vagy

```

ACTION (N) :
    result := sub_program ( ) ;
END_ACTION;
  
```

Funkcióblokkoknál „C”-ben, vagy ST-ben, IL-ben, LD-ben, FBD-ben:

```

ACTION (P) :
    Fbinst(in1, in2);
    result1 := Fbinst.out1;
    result2 := Fbinst.out2;
END_ACTION;
  
```

vagy

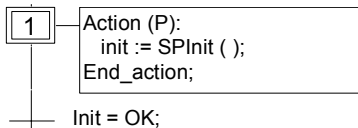
```

ACTION (N) :
    Fbinst(in1, in2);
    result1 := Fbinst.out1;
    result2 := Fbinst.out2;
END_ACTION;

```

A részletes szintaxis az ST nyelvi fejezetben található.  
 Egy alprogram hívás példája műveleti blokkokban:

**(\* SFC program alprogram meghívással egy műveleti blokkban \*)**



#### B.3.5.6 IL konvenció

Az Utasításlista (IL) programozás közvetlenül bevihető egy SFC művelet blokkban, a következő szintaxis alapján:

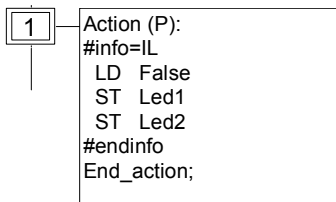
```

ACTION (P) :          (* vagy N *)
#info=IL
    <utasítás>
    <utasítás>
    ....
#endinfo
END_ACTION;

```

A speciális „#info=IL” és „#endinfo” kulcsszavakat pontosan így kell beírni, és azok **érzékenyek a kisbetű-nagybetű beállításra**. A kulcsszavakba, valamint azok elé és után nem illeszthetők be szóköz és tab karakterek. Az alábbiakban egy IL program példája látható egy művelet blokkban:

**(\* SFC program IL szekvenciával egy műveleti blokkban \*)**



### B.3.6 Átmenetekhez csatolt feltételek

Minden átmenethez egy **logikai kifejezés** van csatolva, amely az átmenet tisztázásának feltétele. A feltétel általában ST nyelvvvel, vagy az LD nyelv használatával, illetve a Gyors LD szerkesztővel van kifejezve. Ez az átmenet **2. szintje**. Azonban más struktúrák is használhatók:

- ST nyelvi konvenció
- LD nyelvi konvenció
- IL nyelvi konvenció
- Funkciók meghívása egy átmenetből

Figyelmeztetés: Ha az átmenethez nincs kifejezés csatolva, akkor az alapértelmezett feltétel **IGAZ**.

#### B.3.6.1 ST konvenció

A **Strukturált szöveg** (ST) nyelv használható egy átmenethez csatolt **feltétel** leírására. A teljes kifejezésnek **logikai** típusúnak kell lennie, és egy **pontosvesszővel** kell befejeződnie, a következő szintaxisnak megfelelően:

**< boolean\_expression > ;**

A kifejezés lehet egy IGAZ vagy HAMIS konstans kifejezés, egyetlen input, vagy egy belső logikai változó, illetve változók kombinációja, amely egy logikai értékhez vezet. Az alábbiakban ST programozás példája látható átmenetekhez:

(\* SFC program, ST programozással átmenetekhez \*)



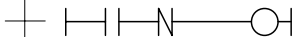
— Fut & Nincs hiba;

#### B.3.6.2 LD konvenció

A **Létradiagram** (LD) nyelv használható egy átmenethez csatolt **feltétel** leírására. A diagram egyetlen tekercset tartalmazó létrafokból áll. A tekercs értéke képviseli az átmenet értékét. Az alábbiakban LD programozás példája látható átmenetekhez:



Fut Hiba



#### B.3.6.3 IL konvenció

Az Utasításlista (IL) programozás közvetlenül bevihető egy SFC átmenet leírására, a következő szintaxis alapján:

```
#info=IL
    <instruction>
    <instruction>
    ....
```

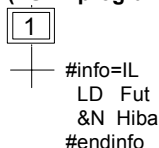
**#endinfo**

A **pillanatnyi eredmény** által tartalmazott érték (IL regiszter) az IL szekvencia végén az eredményül kapott feltétel átmenethez csatolását okozza:

<b>current result = 0</b>	→	a feltétel <b>HAMIS</b>
<b>current result &lt;&gt; 0</b>	→	a feltétel <b>IGAZ</b>

A speciális „#info=IL” és „#endinfo” kulcsszavakat pontosan így kell beírni, és azok **érzékenyek a kisbetű-nagybetű beállításra**. A kulcsszavakba, valamint azok elé és után nem illeszthetők be szóköz és tab karakterek. Az alábbiakban IL programozás példája látható átmenetekhez:

(\* SFC program, IL programozással átmenetekhez \*)



### B.3.6.4 Funkciók meghívása egy átmenetből

Bármely alprogram, vagy egy funkció (FBD, LD, ST vagy IL nyelven írva), illetve egy „C” funkció meghívható egy átmenethez csatolt feltétel kiértékelésére, a következő szintaxis szerint:

**< sub\_program > ( ) ;**

Az alprogram vagy a funkció által visszaadott értéknek logikainak kell lennie, és az a következő feltételt eredményezi:

<b>return value = FALSE</b>	→	a feltétel <b>HAMIS</b>
<b>return value = TRUE</b>	→	a feltétel <b>IGAZ</b>

Egy átmenetben hívott alprogram példája:

(\* SFC program, alprogram meghívással, átmenetekhez \*)



### B.3.7 SFC dinamikus szabályok

Az SFC nyelv **öt** dinamikus szabálya a következő:



### Kezdeti helyzet

A kezdeti helyzetet a **kezdeti lépések** jellemzik, amelyek definíciójuk szerint a művelet kezdetén aktív állapotban vannak. **Legalább egy** kezdeti lépésnek jelen kell lennie minden SFC programban.



### Egy átmenet tisztázása

Egy átmenet vagy **be van kapcsolva**, vagy **ki van kapcsolva**. Akkor számít bekapcsolt állapotúnak, amikor a megfelelő átmeneti szimbólumához kapcsolt összes megelőző lépés **aktív**, egyébként ez ki van kapcsolva. Egy átmenet csak akkor **tisztázható**, ha:

- be van kapcsolva, és
- a hozzá tartozó átmeneti feltétel igaz.



### Aktív lépések állapotának változása

Egy átmenet tisztázása egyidejűleg a közvetlenül utána levő lépések aktív állapotához, és a közvetlenül előtte álló lépések inaktív állapotához vezet.



### Átmenetek egyidejű tisztázása

Az egyidejűleg tisztázott átmenetek jelzésére kettős vonalak használhatók. Ha az ilyen átmeneteket külön tüntetik fel, akkor a megelőző lépések műveleti állapota (GSnnn.x) használható feltételeik kifejezésére.



### Egy lépés egyidejű aktiválása és deaktiválása

Ha a működés során egy lépés egyidejűleg aktiválva és deaktiválva van, akkor az aktiválás kap prioritást.

## B.3.8 SFC program hierarchia

Az ISaGRAF rendszer lehetővé teszi az SFC programok függőleges szerkezetének leírását. Az SFC programok egy **fastruktúrájú hierarchiába** vannak szervezve. Mindegyik SFC program vezérelhet (elindíthat, leállíthat, stb.) más SFC programokat. Az ilyen programokat az őket vezérlő SFC program **gyermekének** nevezik. Az SFC programok egy fő **fastruktúrájú hierarchiában** vannak összekapcsolva egymással, „**apa gyermek**” kapcsolatokkal:



A hierarchikus struktúrából adódó alapszabályok a következők:

- Azoknak az SFC programoknak, amelyeknek nincs apa programjuk, „**fő**” SFC program a neve.
- A fő SFC programokat a rendszer aktiválja az alkalmazás elindulásakor.
- Egy programnak több gyermek programja lehet
- Egy program gyermek programjának nem lehet egynél több apa programja
- Egy gyermek programot csak apa programja vezérelheti
- Egy program nem vezérelheti egyik gyermek programjának a gyermek programját

Egy apa SFC program által gyermek programja vezérléséhez megtehető alapvető műveletek a következők:

Kezdet	<b>(GSTART)</b> elindítja a gyermek programot: annak mindegyik kezdeti lépését aktiválja. A gyermek program gyermekei nem indulnak el automatikusan.
Leállítás	<b>(GKILL)</b> Leállítja a gyermek programot, annak valamennyi aktív lépésének deaktiválásával. A gyermek program valamennyi gyermeke szintén leállítódik.
Befagyasztás	<b>(GFREEZE)</b> Felfüggeszti a program végrehajtását (deaktiválja az összes aktív lépés műveleteit, és felfüggeszti az átmenet számítást), valamint memorizálja a program lépéseinek állapotát, hogy a program újraindítható legyen. A gyermek program valamennyi gyermeke szintén befagyasztódik.
Újraindítás	<b>(GRST)</b> Újraindít egy befagyasztott SFC programot, az összes felfüggesztet lépés újra aktiválásával. A program gyermekei nem indulnak el automatikusan.
Státusz bekérése	<b>(GSTATUS)</b> Bekéri egy gyermek program pillanatnyi státuszát (aktív, inaktív, vagy befagyasztott).

## B.4. Blokkdiagram nyelv

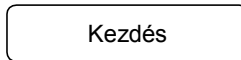
A **Blokkdiagram** („Flow Chart; **FC**”) egy **szekvenciális műveletek** leírására használt grafikus nyelv. Egy blokkdiagram **Műveletekből** és **Tesztekből** áll. A műveletek és tesztek között **irányított (orientált) kapcsolatok** vannak, amelyek az adatáramlást képviselik. A divergenciákat és konvergenciákat többszörös csatlakozási kapcsolatok képviselik. A műveletek és tesztek az ST, LD, vagy IL nyelvekkel írhatók le. Bármilyen nyelvű funkciók és funkcióblokkok (az SFC kivételével) meghívhatók a műveletekből és tesztekből. Egy blokkdiagram program meghívhat egy másik blokkdiagram programot. A meghívott FC program a hívó FC program **alprogramja**.

### B.4.1. FC komponensek

Az alábbiakban a Blokkdiagram nyelv grafikus komponensei vannak felsorolva:

#### **Az FC diagram kezdete**

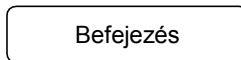
Egy Blokkdiagram program elején a „**kezdés**” szimbólumnak kell állnia. Ez egyedi, és nem hagyható el. Ez a diagram aktiválásakor annak kezdeti állapotát jelzi. Az alábbiakban a „kezdés” szimbólum rajza látható:



A „Kezdés” szimbólum mindig rendelkezik egy kapcsolattal (alul) a diagram többi tárgyához. A blokkdiagram érvénytelen, ha a „Kezdés”-től nincs csatlakozás rajzolva egy másik tárgyhöz.

#### **Az FC diagram vége**

Egy Blokkdiagram program végén egy „**befejezés**” szimbólumnak kell állnia. Ez egyedi, és nem hagyható el. Lehetséges, hogy a „Befejezés” szimbólumhoz nincs csatlakozás rajzolva (állandóan hurkolódó diagram), de a „Befejezés” szimbólumot akkor is meg kell rajzolni a diagram aljára. Ez a diagram végrehajtásának befejezésekor annak végső állapotát jelzi. Az alábbiakban a „befejezés” szimbólum rajza látható:



A „Befejezés” szimbólum általában rendelkezik egy kapcsolattal (felül) a diagram többi tárgyához. Lehetséges, hogy egy blokkdiagramnak nincs csatlakozása a „Befejezés” tárgyhöz (állandóan hurkolódó diagram). A „Befejezés” tárgy ebben az esetben is látható a diagram alján.

#### **FC áramlás kapcsolatok**

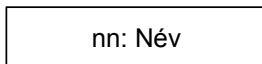
Az áramlás **kapcsolat** egy vonal, amely a diagram két pontja közötti áramlást képvisel. A kapcsolat mindig egy nyíllal van lezárva. Az alábbiakban egy kapcsolat rajza látható:



Két kapcsolat nem indulhat ugyanabból a forrás csatlakozási pontból.

### FC műveletek

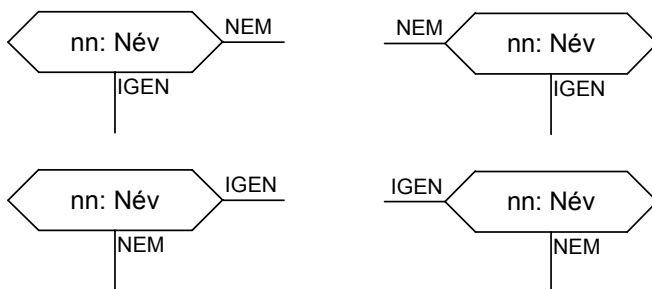
A végrehajtandó műveleteket egy **művelet** szimbólum jelzi. Egy művelet egy számmal és egy névvel van azonosítva. Az alábbiakban egy „művelet” szimbólum rajza látható:



Ugyanannak a diagramnak két különböző tárgya nem rendelkezhet ugyanazzal a névvel vagy logikai számmal. Egy művelet programozási nyelve ST, LD, vagy IL lehet. Egy művelet mindig kapcsolatokkal csatlakozik, amelyek egyike hozzá érkezik, a másik pedig tőle indul.

### FC feltételek

Egy **feltétel** egy logikai **tesztet** jelent. A feltétel egy számmal és egy névvel van azonosítva. A csatolt ST, LD vagy IL kifejezés kiértékelésétől függően az áramlás vagy az „IGEN”, vagy a „NEM” útvonal felé irányítódik. Az alábbiakban egy feltétel szimbólum lehetséges rajzai láthatók:



Ugyanannak a diagramnak két különböző tárgya nem rendelkezhet ugyanazzal a névvel vagy logikai számmal. Egy próba programozása vagy

- egy kifejezés ST-ben, vagy
- egyetlen létrafok LD-ben, úgy hogy az egyedi tekercshez nem csatlakozik szimbólum, vagy
- több utasítás IL-ben. Az IL regiszter (vagy pillanatnyi eredmény) a feltétel kiértékelésére szolgál.

ST szövegben programozva a kifejezést tetszés szerint egy pontosvessző is követheti. LD-ben programozva, az egyedi tekercs képviseli a feltétel értékét. Az alábbi feltétel, amely:

- 0 vagy HAMIS, az áramlást NEM felé irányítja
- 1 vagy IGAZ, az áramlást IGEN felé irányítja

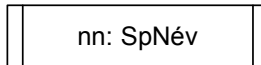
Egy próba mindig egy érkező kapcsolathoz csatlakozik, és mindkét előre mutató csatlakozást definiálni kell.

## FC alprogram

A rendszer lehetővé teszi az FC programok függőleges szerkezetének leírását. Az FC programok egy **fastruktúrájú hierarchiába** vannak szervezve. Minden FC program meghívhat más FC programokat. Az ilyen program neve a hívó FC program **gyermek programja**. Azokat az FC programokat a neve, amelyek FC alprogramokat hívnak meg, **apa program** a neve. Az FC programok egy fő fastruktúrájú hierarchiában vannak összekapcsolva egymással, „apa gyermek” kapcsolatokkal:



A blokkdiagramban levő **alprogram** szimbólum egy blokkdiagram alprogramhoz történő hívást jelképez. A hívó FC program végrehajtása az alprogram végrehajtásának befejezéséig felfüggesztődik. Egy blokkdiagram alprogramot, akár csak más programokat, funkciókat, vagy funkcióblokkokat, egy szám és egy név azonosít. Az alábbiakban egy „alprogram meghívás” szimbólum rajza látható:



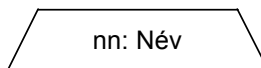
Ugyanannak a diagramnak két különböző tárgya nem rendelkezhet ugyanazzal a logikai számmal. Az FC hierarchikus struktúrából adódó alapszabályok a következők:

- Azoknak az FC programoknak, amelyeknek nincs apa programjuk, fő FC program a neve.
- A fő FC programokat a rendszer aktiválja az alkalmazás elindulásakor.
- Egy programnak több gyermek programja lehet
- Egy program gyermek programjának nem lehet egynél több apa programja
- Egy gyermek programot csak apa programja hívhat meg
- Egy program nem hívhatja meg egyik gyermek programjának a gyermek programját

Ugyanaz az alprogram többször is megjelenhet az apa diagramban. Egy blokkdiagram alprogram meghívás az aldiagram teljes végrehajtását jelenti. Az apa diagram végrehajtása a gyermek diagram végrehajtása közben felfüggesztődik. Az alprogram meghívó blokkoknak ugyanazokat a csatlakozási szabályokat kell betartaniuk, mint a művelethez definiáltaknak.

## FC I/O specifikus művelet

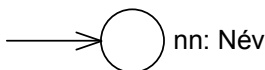
Az I/O **specifikus művelet** szimbólum végrehajtandó műveleteket jelez. Akárcsak egyéb műveletek, úgy az I/O specifikus művelet is egy számmal és egy névvel van azonosítva. A hagyományos műveletekhez és az I/O specifikus műveletekhez ugyanaz a szemantika érvényes. Az I/O specifikus műveletek célja csupán a diagram könnyebb olvashatóságának elősegítése, és a diagram át nem vihető részeire való összpontosítás. Az I/O specifikus műveletek használata egy tetszés szerint választható lehetőség. Az alábbiakban egy „I/O specifikus művelet” szimbólum rajza látható:



Az I/O specifikus blokkok ugyanúgy viselkednek, mint a hagyományos műveletek. Ez jelenti tulajdonságaikat, ST, LD vagy IL programozásukat, és csatlakozási szabályait.

## FC csatlakozók

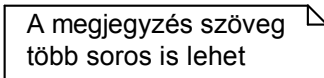
A **Csatlakozók** egy kapcsolatot képviselnek a diagram két pontja között, annak megrajzolása nélkül. A csatlakozót egy kör jelöli, amely az áramlás forrásával van összekötve. A csatlakozó rajzát a megfelelő oldalon (az adatáramlás irányától függően) a célpont azonosítója (általában a cél szimbólum neve) teszi teljessé. Az alábbiakban egy csatlakozó standard rajza látható:



Egy csatlakozó mindig egy definiált blokkdiagram szimbólumot céloz meg. A rendeltetési hely szimbólum annak logikai számával van azonosítva.

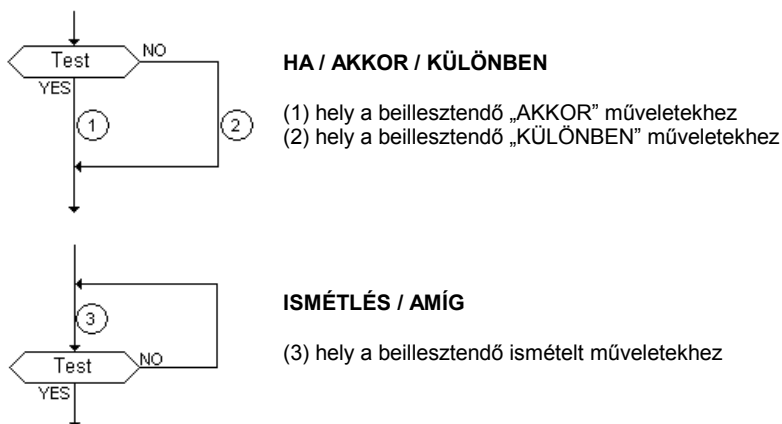
## FC megjegyzések FC110

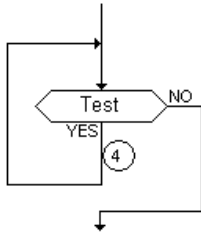
A **megjegyzés** blokk olyan szöveget tartalmaz, amelynek a diagram szemantikájában nincs értelme. Ez a blokkdiagram dokumentum ablak szabad helyére bárhová beilleszthető, és célja a program dokumentálása. Az alábbiakban egy „megjegyzés” szimbólum rajza látható:



## B.4.2. FC komplex struktúrák

Ez a fejezet a blokkdiagramban definiálható **komplex struktúra** példákat mutat. Az ilyen struktúrák összekapcsolt alaptárgyak kombinációi.





### MIKÖZBEN / ELVÉGZÉS

(3) hely a beillesztendő ismételt műveletekhez

#### B.4.3. FC dinamikus viselkedés

Egy blokkdiagram **végrehajtása** a következőképpen magyarázható el:

- A Kezdés szimbólum elvégez egy célciklust
- A Befejezés szimbólum elvégez egy célciklust és befejezi a diagram végrehajtását. Ennek a szimbólumnak az elérése után a diagram egyetlen művelete sem kerül többé végrehajtásra.
- Az áramlás mindannyiszor megszakad, amikor egy olyan tétel (művelet, döntés) kerül sorra, ami ugyanabban a célciklusban egyszer már el lett érve. Ilyen esetben az áramlás a következő ciklusban folytatódik.

**Megjegyzés:** Az SFC ellenére egy művelet nem egy stabil állapot. Miközben a művelet szimbóluma ki van világosítva, nincsenek ismétlési utasítások.

#### B.4.4. FC ellenőrzés

A csatolt ST, LD vagy IL programozáson kívül magára a blokkdiagramra bizonyos egyéb **szintaktikai szabályok** is vonatkoznak. Az alábbiakban a fő szabályok listája látható:

- Minden szimbólum valamennyi „csatlakozási” pontját be kell huzalozni. (a „Befejezés” szimbólumhoz való csatlakozás elhagyható)
- Minden szimbólumot egymáshoz kell kapcsolni (nem szabad elkülönült résznek lenni)
- Minden csatlakozónak érvényes rendeltetési hellyel kell rendelkeznie

Egyéb kisebb jelentőségű szintaxishibákról üzenet készülhet:

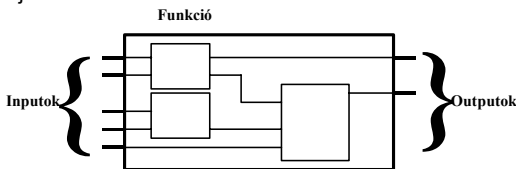
- Az üres műveletek (programozás nélkül) a futtatás időbeosztásánál lépésként vannak figyelembe véve
- Az üres próbák (programozás nélkül) „mindig igaz”-ként vannak figyelembe véve

## B.5. Az FBD nyelv

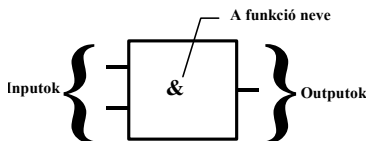
A **Funkcionális blokkdiagram** (FBD) (Functional Block Diagram) egy grafikus nyelv. Ez lehetővé teszi a programozó számára komplex eljárások felépítését az ISaGRAF könyvtárban létező **funkcióknak** a grafikus diagram területen történő **bekötésével**.

### B.5.1. Az FBD diagram fő formátuma

Az FBD diagram egy **input változók** és **output változók** közötti funkciót ír le. Egy funkció úgy írható le, mint **elemi funkcióblokkok** készlete. Az input és output változók **összekötő vonalakkal** vannak a blokkokhoz csatlakoztatva. Egy funkcióblokk outputja csatlakoztatható egy másik blokk inputjához is.



Egy FBD programmal működtetett teljes funkció felépítése standard **elemi** funkcióblokkok segítségével történik az ISaGRAF könyvtárból. Minden funkcióblokk meghatározott számú **input csatlakozási ponttal** és meghatározott számú **output csatlakozási ponttal** rendelkezik. Egy funkcióblokkot egy **négyzet** jelképez. Az inputok annak **bal** oldali széléhez vannak csatlakoztatva. Az outputok annak **jobb** oldali széléhez vannak csatlakoztatva. Egy elemi funkcióblokk egyetlen **funkciót** hajt végre annak inputjai és outputjai között. A blokk által elvégzendő funkció neve annak négyzet alakú jelébe van írva. Egy blokk minden inputjának vagy outputjának jól meghatározott **típusa** van.



Egy FBD program input változóinak funkcióblokkok input csatlakozási pontjaihoz kell csatlakozniuk. Minden változó típusának ugyanolyannak kell lennie, mint amilyen a kapcsolódó inputhoz van elvárva. Egy FBD diagram inputja lehet egy **konstans** kifejezés, bármilyen **belső** vagy **input** változó, illetve **output** változó.

Egy FBD program output változóinak funkcióblokkok output csatlakozási pontjaihoz kell csatlakozniuk. Minden változó típusának ugyanolyannak kell lennie, mint amilyen a kapcsolódó blokk outputhoz van elvárva. Egy FBD diagram outputja lehet bármilyen **belső** vagy **output** változó, illetve a program neve (csak **alprogramokhoz**). Amikor egy output a pillanatnyilag szerkesztett alprogram neve, akkor az az alprogram visszatérési értékének (amelyet a hívó program ad vissza) kijelölését jelenti.

Az input és output változókat, a funkcióblokkok inputjait és outputjait **csatlakozó vonalak** kötik össze. A diagram alábbiakban felsorolt két logikai pontjának **összekötésére** egyszeres vonalak használhatók:

- Egy input változó és egy funkcióblokk inputja
- Egy funkcióblokk outputja és egy másik blokk inputja
- Egy funkcióblokk outputja és egy output változó

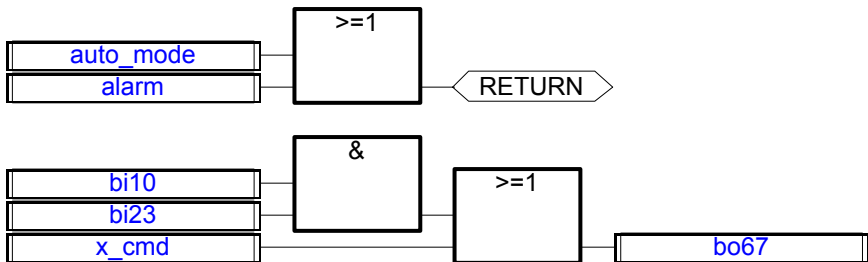
A csatlakozás **irányított**, ami azt jelenti, hogy a vonal bal oldali szélétől jobb oldali széléhez szállítja a kapcsolódó adatokat. A csatlakozó vonal bal és jobb oldali szélének **ugyanolyan típusúnak** kell lennie.

Többszörös jobb oldali csatlakozás használható egy információ közlésére a bal oldali szélétől a jobb oldali szélék mindegyikéhez. A csatlakozás valamennyi szélének ugyanolyan típusúnak kell lennie.

### B.5.2. RETURN utasítás

A „**RETURN**” kulcsszó diagram outputként jelentkezhet. Ennek egy funkcióblokk logikai output pontjához kell csatlakoznia. A RETURN utasítás a program **feltételes végét** jelenti: ha az utasításhoz csatlakozó keret outputja logikai **IGAZ** értékű, akkor a diagram vége (hátralevő része) nem kerül végrehajtásra.

(\* Egy RETURN utasítást alkalmazó FBD program példája \*)



(\*ST ekvivalencia: \*)

```
If auto_mode OR alarm Then
    Return;
```

```
End_if;
```

```
bo67 := (bi10 AND bi23) OR x_cmd;
```

### B.5.3. Ugrások és címkék

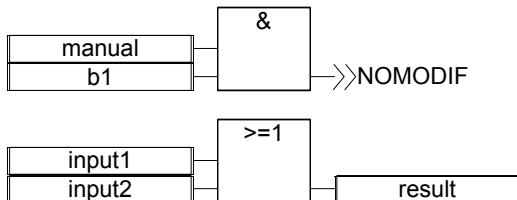
A címkék és ugrások a diagram végrehajtására szolgálnak. Egy ugrás vagy címke szimbólum jobb oldalára nem csatlakoztatható más tárgy. A következő elnevezések használatosak:

**>>LAB**..... ugrás egy címkére (a címke neve „LAB”)

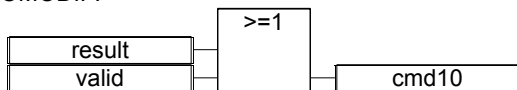
**LAB:**..... egy címke definíciója (a címke neve „LAB”)

Ha az ugrás szimbólum **bal** oldalán levő csatlakozási vonal logikai **IGAZ** állapotú, akkor a program végrehajtása közvetlenül a megfelelő címke szimbólum utánra ugrik.

(\* Egy címkéket és ugrásokat alkalmazó FBD program példája \*)



NOMODIF:



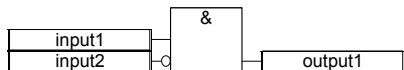
(\* IL Equivalencia: \*)

	ld	manual
	and	b1
	jmpc	NOMODIF
	ld	input1
	or	input2
	st	result
NOMODIF:	ld	result
	or	valid
	st	cmd10

### B.5.4. Logikai tagadás

Egy, a jobb oldali szélén egy funkcióblokk inputjához csatlakozó egyszeres csatlakozó vonal lezárására egy **logikai tagadás** alkalmazható. A tagadást egy kis kör képviseli. Egy logikai tagadás használatakor a csatlakozási vonal bal és jobb oldali széleinek **LOGIKAI** típusúnak kell lennie.

(\* Egy logikai tagadást alkalmazó FBD program példája \*)



(\*ST ekvivalencia: \*)

output1 := input1 AND NOT (input2);

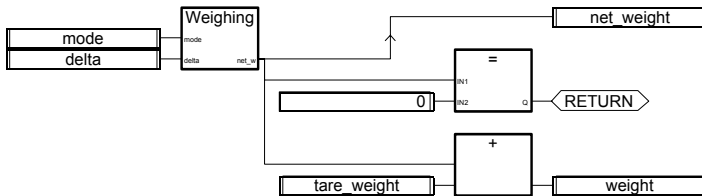
### B.5.5. Funkciók vagy funkcióblokkok meghívása az FBD-ből

Az FBD nyelv lehetővé teszi alprogramok, funkciók, vagy funkcióblokkok meghívását. Egy alprogramot, vagy funkciót illetve funkcióblokkot egy funkció keret képvisel. A keretbe írt név az alprogram vagy funkció illetve funkcióblokkok neve.

Egy alprogram vagy funkció esetén a visszatérési érték a funkciókeret egyetlen outputja.

Egy funkcióblokknak egynél több outputja is lehet.

(\* Egy ALPROGRAM blokkot alkalmazó FBD program példája \*)



(\* ST Equivalencia \*)

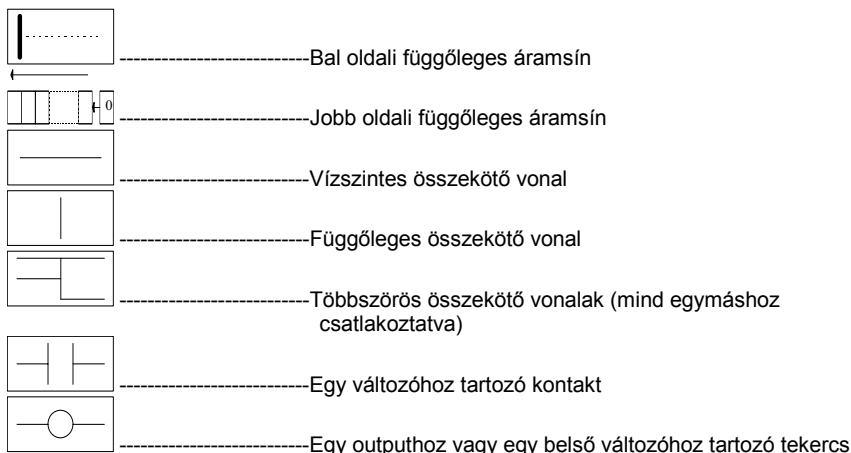
net\_weight := weighing (mode, delta); (\* alprogram meghívása \*)

Ha (net\_weight = 0) Then Return; End\_if;

weight := net\_weight + tare\_weight;

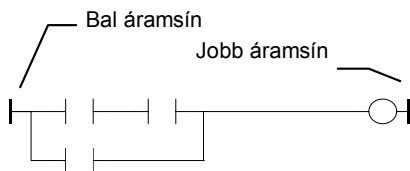
## B.6. Az LD nyelv

A Létradiagram (LD) logikai egyenletek grafikus megjelenítése, amely **kontaktokat** (input argumentumokat) és **tekercseket** (output eredményeket) egyesít. Az LD nyelv lehetővé teszi **logikai** adatok tesztjeinek és módosításainak a leírását, **grafikus szimbólumoknak** a program diagramjába történő elhelyezésével. Az LD grafikus szimbólumok a diagramban pontosan úgy vannak elrendezve, mint egy elektromos kontakt diagramban. Az LD diagramokat a bal és jobb oldalon függőleges **áramsínek** kötik össze. Az alábbiakban egy LD diagram alapvető komponensei vannak felsorolva:

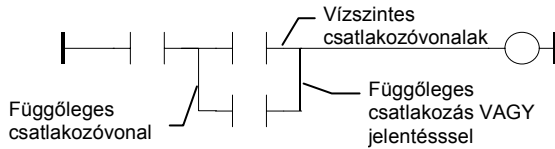


### B.6.1. Áramsínek és összekötő vonalak

Egy LD diagramot a bal és jobb oldalon függőleges vonalak határolnak be, amelyeknek neve **bal áramsín** illetve **jobb áramsín**.



Az LD diagram grafikus szimbólumai **összekötő vonalakkal** csatlakoznak az áramsínekhez vagy más szimbólumokhoz. Az összekötő vonalak függőlegesek vagy vízszintesek lehetnek.



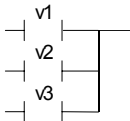
Minden vonalszegmens logikai **IGAZ** vagy **HAMIS** állapotban van. Valamennyi, közvetlenül egymáshoz kapcsolódó szegmens logikai állapota ugyanaz. Valamennyi, a bal oldali **függőleges áramsínhez** csatlakoztatott vízszintes vonal **IGAZ** állapotú.

### B.6.2. Többszörös kapcsolat

Egy vízszintes összekötő vonal által adott logikai állapot a vonal bal és jobb oldali szélein megegyezik. A vízszintes és függőleges összekötő vonalak kombinálásával lehetséges **többszörös csatlakozások** kialakítása. Egy többszörös csatlakozás széleinek logikai állapota a logikai szabályokat követi.

Egy **bal oldalon levő többszörös csatlakozás** egy függőleges vonal bal oldalához kapcsolódó **egynél több** vízszintes vonalat egyesít a **jobb** oldalához csatlakoztatott **egyetlen** vonallal. A jobb oldali szél logikai állapota mindegyik bal szél között **LOGIKAI VAGY**.

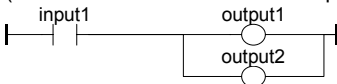
(\* Többszörös BAL csatlakozás példája \*)



(\* a jobb oldali szél állapota (v1 VAGY v2 VAGY v3) \*)

Egy **jobb oldalon levő többszörös csatlakozás** egy függőleges vonal jobb oldalához kapcsolódó **egynél több vonalat** egyesít a **bal** oldalához csatlakoztatott **egyetlen** vízszintes vonallal. A bal oldali szél logikai állapota mindegyik jobb szélre átadódik.

(\* Többszörös JOBB csatlakozás példája \*)



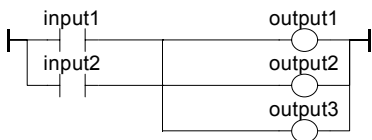
(\*ST ekvivalencia: \*)

output1 := input1;

output2 := input1;

Egy **bal és jobb oldalon levő többszörös csatlakozás** egy függőleges vonal **bal** oldalához kapcsolódó **egynél több** vízszintes vonalat egyesít a **jobb** oldalhoz csatlakoztatott **egynél több** vonallal. Mindegyik jobb oldali szél logikai állapota mindegyik bal szél között **LOGIKAI VAGY**.

(\* Többszörös BAL és JOBB csatlakozás példája \*)



(\* ST Equivalencia \*)

output1 := input1 OR input2;

output2 := input1 OR input2;

output3 := input1 OR input2;

### B.6.3. Alapvető LD kontaktok és tekercsek

Az input kontaktokhoz több szimbólum áll rendelkezésre:

- Közvetlen kontakt
- Invertált kontakt
- Kontaktok éldetektálással

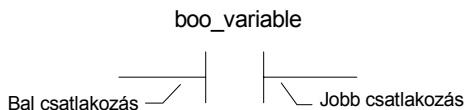
Az output tekercsekhez több szimbólum áll rendelkezésre:

- Közvetlen tekercs
- Invertált tekercs
- BEÁLLÍTÁS tekercs
- VISSZAÁLLÍTÁS tekercs
- Tekercsek éldetektálással

A változó neve a tekercs ezeknek a szimbólumnak a tetején van feltüntetve.

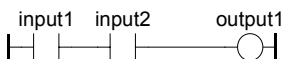
#### ☐ **Közvetlen kontakt**

A közvetlen kontakt egy **logikai műveletet** tesz lehetővé egy **összekötő vonal** állapot és egy logikai **változó** között.



A kontakttól jobbra levő összekötő vonal állapota a bal összekötő vonal állapota és a kontaktra vonatkozó változó értéke között **LOGIKAI ÉS**.

(\* KÖZVETLEN kontaktok alkalmazási példája \*)

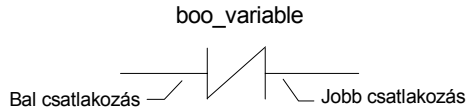


(\* ST Equivalencia \*)

output1 := input1 AND input2;

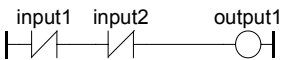
### ☐ **Invertált kontakt**

Az invertált kontakt egy **logikai műveletet** tesz lehetővé egy **összekötő vonal** állapota és egy logikai **változó** logikai tagadása között.



A kontakttól jobbra levő összekötő vonal állapota a bal összekötő vonal állapota és a kontaktra vonatkozó változó értékének **logikai tagadása** között **LOGIKAI ÉS**.

(\* INVERTÁLT kontaktok alkalmazási példája \*)

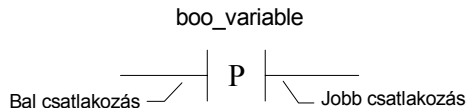


(\* ST Equivalencia \*)

output1 := NOT (input1) AND NOT (input2);

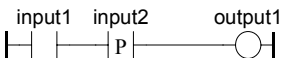
### ☐ **Kontaktok felfutó él detektálással**

Ez a kontakt (pozitív) egy **logikai műveletet** tesz lehetővé egy **összekötő vonal** állapota és egy logikai **változó** felfutó éle között.



A kontakttól jobbra levő összekötő vonal állapota **IGAZ**-ra van állítva, amikor a bal összekötő vonal állapota **IGAZ**, és a vonatkozó változó állapota HAMIS-ról IGAZ-ra **fut fel**. Minden más esetben HAMIS-ra van visszaállítva.

(\* FELFUTÓ ÉLŰ kontaktok alkalmazási példája \*)



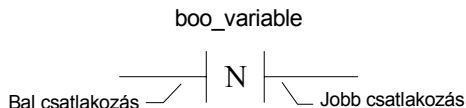
(\* ST Equivalencia \*)

output1 := input1 AND (input2 AND NOT (input2prev));

(\* input2prev az input2 értéke az előző ciklusnál \*)

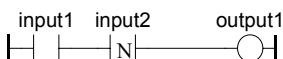
### ☐ **Kontaktok lefutó él detektálással**

Ez a kontakt (negatív) egy **logikai műveletet** tesz lehetővé egy **összekötő vonal** állapota és egy logikai **változó** lefutó éle között.



A kontakttól jobbra levő összekötő vonal állapota **IGAZ**-ra van állítva, amikor a bal összekötő vonal állapota **IGAZ**, és a vonatkozó változó állapota IGAZ-ról HAMIS-ra **fut le**. Minden más esetben HAMIS-ra van visszaállítva.

(\* LEFUTÓ ÉLŰ kontaktok alkalmazási példája \*)



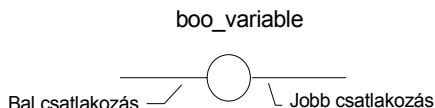
(\* ST Equivalencia \*)

output1 := input1 AND (NOT (input2) AND input2prev);

(\* input2prev az input2 értéke az előző ciklusnál \*)

### ⇒ Közvetlen tekercs

A közvetlen tekercsek egy **összekötő vonal** logikai állapotának **logikai outputját** teszik lehetővé.

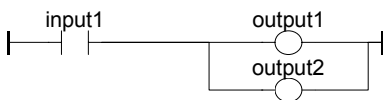


A vonatkozó változó a **bal oldali csatlakozás** logikai **állapotával** van kijelölve. A bal oldali szél logikai állapota átadódik a jobb szélre. A jobb csatlakozás a jobb oldali függőleges áramsinhez csatlakoztatható.

A vonatkozó logikai változónak **OUTPUTNAK** vagy **BELSŐNEK** kell lennie.

A vonatkozó név lehet a program neve (csak **alprogramoknál**). Ez megfelel az alprogram visszatérési érték kijelölésének.

(\* KÖZVETLEN tekercsek alkalmazási példája \*)



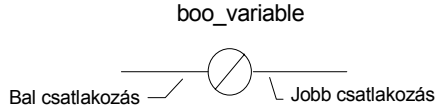
(\* ST Equivalencia \*)

output1 := input1;

output2 := input1;

### ⇒ Invertált tekercs

Az invertált tekercsek egy **logikai outputot** tesznek lehetővé egy **összekötő vonal** logikai **tagadásának** megfelelően.

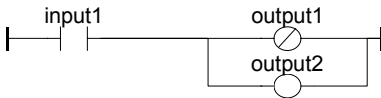


A vonatkozó változó a **bal oldali csatlakozás** állapotának logikai **tagadásával** van kijelölve. A bal oldali szél logikai állapota átadódik a jobb szélre. A jobb csatlakozás a jobb oldali függőleges áramsínhez csatlakoztatható.

A vonatkozó logikai változónak **OUTPUTNAK** vagy **BELSŐNEK** kell lennie.

A vonatkozó név lehet a program neve (csak **alprogramoknál**). Ez megfelel az alprogram visszatérési érték kijelölésének.

(\* INVERTÁLT tekercsek alkalmazási példája \*)



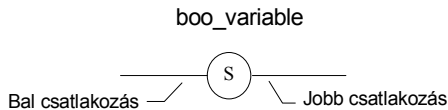
(\* ST Equivalencia \*)

output1 := NOT (input1);

output2 := input1;

## BEÁLLÍTÁS tekercs

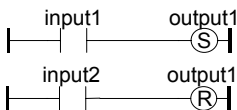
A „Beállítás” tekercsek lehetővé teszik egy **összekötő vonal** logikai állapotának **logikai outputját**.



A vonatkozó változó **IGAZRA VAN ÁLLÍTVA**, amikor a **bal oldali csatlakozás** logikai állapota **IGAZZÁ** válik. Az output változó ezt az értéket mindaddig megtartja, amíg egy „VISSZAÁLLÍTÁS” tekercs ellenkezőképpen nem rendelkezik. A bal oldali szél logikai állapota átadódik a jobb szélre. A jobb csatlakozás a jobb oldali függőleges áramsínhez csatlakoztatható.

A vonatkozó logikai változónak **OUTPUTNAK** vagy **BELSŐNEK** kell lennie.

(\* BEÁLLÍTÁS és VISSZAÁLLÍTÁS tekercsek alkalmazási példája \*)



(\* ST Equivalencia \*)

IF input1 THEN

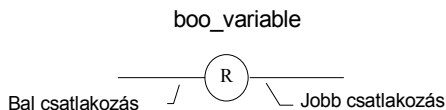
```

output1 := TRUE;
END_IF;
IF input2 THEN
  output1 := FALSE;
END_IF;

```

### ➤ **VISSZAÁLLÍTÁS tekercs**

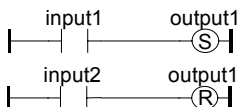
A „Visszaállítás” tekercsek lehetővé teszik egy **összekötő vonal** logikai állapotának **logikai outputját**.



A vonatkozó változó **HAMISRA VAN VISSZAÁLLÍTVA**, amikor a **bal oldali csatlakozás** logikai **állapota IGAZZÁ** válik. Az output változó mindaddig megtartja ezt az állapotot, amíg egy „BEÁLLÍTÁS” tekercs egy fordított rendelkezést nem hoz. A bal oldali szél logikai állapota átadódik a jobb szélre. A jobb csatlakozás a jobb oldali függőleges áramsinhez csatlakoztatható.

A vonatkozó logikai változónak **OUTPUTNAK** vagy **BELSŐNEK** kell lennie.

(\* BEÁLLÍTÁS és VISSZAÁLLÍTÁS tekercsek alkalmazási példája \*)



(\* ST Equivalencia \*)

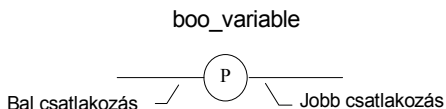
```

IF input1 THEN
  output1 := TRUE;
END_IF;
IF input2 THEN
  output1 := FALSE;
END_IF;

```

### ➤ **Tekercsek felfutó él detektálással**

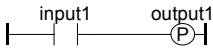
A „Pozitív” tekercsek lehetővé teszik egy **összekötő vonal** logikai állapotának **logikai outputját**. Az ilyen típusú tekercsek csak a Gyors LD szerkesztő használatával állnak rendelkezésre.



A vonatkozó változó **IGAZRA** van állítva, amikor a **bal oldali csatlakozás** logikai **állapota** HAMISRÓL IGAZRA fut fel. Az output változó minden más esetben HAMISRA van visszaállítva. A bal oldali szél logikai állapota átadódik a jobb szélre. A jobb csatlakozás a jobb oldali függőleges áramsínhez csatlakoztatható.

A vonatkozó logikai változónak **OUTPUTNAK** vagy **BELSŐNEK** kell lennie.

(\* „Pozitív” tekercs alkalmazási példája \*)



(\* ST Equivalencia \*)

IF (input1 and NOT(input1prev)) THEN

output1 := TRUE;

ELSE

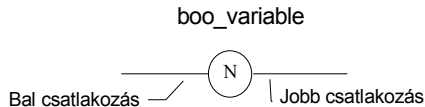
output1 := FALSE;

END\_IF;

(\* input1prev az input1 értéke az előző ciklusban \*)

### == Tekercsek lefutó él detektálással

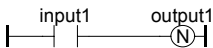
A „Negatív” tekercsek lehetővé teszik egy **összekötő vonal** logikai állapotának **logikai outputját**. Az ilyen típusú tekercsek csak a Gyors LD szerkesztő használatával állnak rendelkezésre.



A vonatkozó változó **IGAZRA** van állítva, amikor a **bal oldali csatlakozás** logikai **állapota** IGAZRÓL HAMISRA esik. Az output változó minden más esetben HAMISRA van visszaállítva. A bal oldali szél logikai állapota átadódik a jobb szélre. A jobb csatlakozás a jobb oldali függőleges áramsínhez csatlakoztatható.

A vonatkozó logikai változónak **OUTPUTNAK** vagy **BELSŐNEK** kell lennie.

(\* „Pozitív” tekercs alkalmazási példája \*)



(\* ST Equivalencia \*)

IF (NOT(input1) and input1prev) THEN

output1 := TRUE;

ELSE

output1 := FALSE;

END\_IF;

(\* input1prev az input1 értéke az előző ciklusban \*)

### B.6.4. RETURN utasítás

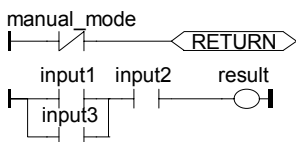
A **RETURN** címke outputként használható, a program feltételes végének jelzésére. A RETURN szimbólum jobb oldalára nem helyezhető el csatlakozás.



Ha a **bal csatlakozó** vonal **IGAZ** logikai állapotú, akkor a program a diagram további soraiba beírt egyenletek elvégzése nélkül befejeződik.

**Megjegyzés:** Amikor az LD program egy alprogram, akkor annak nevének a (meghívó program által visszaadott) visszatérési érték beállításához egy output tekercshez kell vonatkoznia.

(\* A RETURN szimbólum alkalmazási példája \*)



(\* ST Equivalencia \*)

```
If Not (manual_mode) Then RETURN; End_if;
result := (input1 OR input3) AND input2;
```

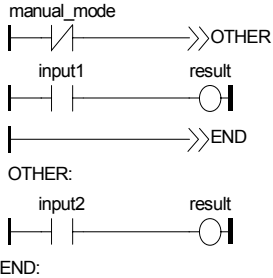
### B.6.5. Ugrások és címkék

A címkék, valamint a feltételes és feltétel nélküli UGRÁS szimbólumok a diagram végrehajtásának vezérlésére használhatók. A címke és az ugrás szimbólum jobb oldalára nem helyezhető el csatlakozás. A következő elnevezések használatosak:

**>>LAB**..... ugrás egy „LAB” nevű címkeére  
**LAB:**..... a „LAB” nevű címke definíciója

Ha az ugrás szimbólum **bal oldalán levő csatlakozás** logikai **IGAZ** állapotú, akkor a program végrehajtása a megfelelő címke szimbólum utánra kerül.

(\* Az UGRÁS és CÍMKE szimbólumok alkalmazási példája \*)



(\* IL Equivalencia: \*)

```

ldn      manual_mode
jmpc
ld       input1
st       result
jmp      END
  
```

OTHER:

```

ld       input2
st       result
  
```

END:

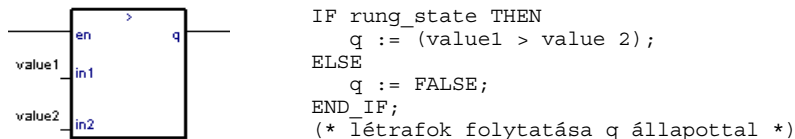
(\* a program vége \*)

### B.6.6. Blokkok az LD-ben

A Gyors LD szerkesztő használatával funkcióblokkok logikai vonalakhoz csatlakoztathatók. Egy funkció valójában lehet egy operátor, egy funkcióblokk, vagy egy funkció. Mivel nem minden blokknak van mindig egy logikai inputja és/vagy logikai outputja, ezért a blokkok elhelyezése az LD diagramban az EN, és ENO új paramétereknek a blokk interfészhez történő hozzáadásához vezet. Az EN, és ENO paraméterek nem lesznek hozzáadva, ha az FBD/LD szerkesztőt használja, mert a változó a szükséges típussal csatlakoztatható.

#### Az „EN” input

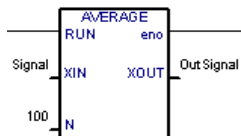
Egyes operátorokon, funkciókon, vagy funkcióblokkokon az első inputnak nincs logikai adat-típusa. Mivel az első inputot mindig a létrafokhoz kell csatlakoztatni, ezért az első pozícióba egy „EN” nevű másik input kerül automatikusan beillesztésre. A blokk csak akkor kerül végrehajtásra, ha az EN input is IGAZ. Az alábbiakban egy összehasonlítás operátor, valamint annak ST-ben kifejezett kódja látható:



#### Az „ENO” output

Egyes operátorokon, funkciókon, vagy funkcióblokkokon az első outputnak nincs logikai adat-típusa. Mivel az első outputot mindig a létrafokhoz kell csatlakoztatni, ezért az első pozícióba egy „ENO” nevű másik output kerül automatikusan beillesztésre. Az ENO output mindig

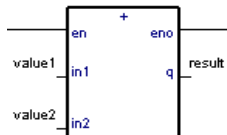
ugyanazt az állapotot veszi fel, mint a blokk első inputja. Az alábbi példában egy ÁTLAG funkcióblokk, valamint annak ST-ben kifejezett kódja látható:



```
AVERAGE(rung_state, Signal, 100);
OutSignal := AVERAGE.XOUT;
eno := rung_state;
(*létrafok folytatása eno állapottal *)
```

### ⇒ Az „EN” és „ENO” paraméterek

Bizonyos esetekben mind az **EN**, mind az **ENO** szükséges. Az alábbi példában egy aritmetikai operátor, valamint annak ST-ben kifejezett kódja látható:



```
IF rung_state THEN
    result := (value1 + value2);
END_IF;
eno := rung_state;
(* létrafok folytatása eno állapottal *)
```

## B.7. ST nyelv

Az ST (**Strukturált szöveg**) egy magas szintű strukturált nyelv, folyamatok automatizálásához. Ezt a nyelvet elsősorban olyan összetett folyamatok kivitelezéséhez használják, amelyek grafikus nyelvekkel nem fejezhetők ki könnyen. Az ST az alapértelmezett nyelv az **SFC** nyelv átmeneteihez csatolt lépéseken és feltételeken belüli műveletek leírására.

### B.7.1. ST fő szintaxis

Egy ST program ST **utasítások** listájából áll. Mindegyik utasítás egy pontosvessző („;”) elválasztóval fejeződik be. A forráskódban használt nevek (változó azonosítók, konstansok, nyelv kulcsszavak, stb.) **inaktív elválasztókkal** (szóköz karakter, sor vége, vagy tab stop), vagy **aktív elválasztókkal** vannak elválasztva, amelyek jól körülhatárolt jelentőséggel bírnak (pl. a „>” elválasztó egy „nagyobb mint” összehasonlítást jelez. A szövegbe szabadon beilleszthetők megjegyzések. A megjegyzések mindig „(\*”-el kezdődnek és „\*)”-el végződnek. Mindegyik utasítás egy pontosvessző („;”) elválasztóval fejeződik be. Az ST utasítások alapvető típusai a következők:

- **kijelölés** utasítás (változó := kifejezés;)
- **alprogram** vagy **funkció** meghívás
- **funkcióblokk** meghívás
- **kiválasztás** utasítások (IF, THEN, ELSE, CASE...)
- **iteráció** utasítások (FOR, WHILE, REPEAT...)
- **vezérlés** utasítások (RETURN, EXIT...)
- speciális utasítások más nyelvekkel, pl. **SFC**-vel való kapcsolatokhoz

Az aktív elválasztók, konstans kifejezések, és azonosítók közé szabadon beilleszthetők inaktív elválasztók. Az ST inaktív elválasztók a következők: **Szóköz** (üres) karakter, **Tab** és **Sor vége** karakter. A sorformázásos nyelvektől, pl. IL-től eltérően a sor vége karakterek a programban bárhol elhelyezhetők. Az ST program olvashatóságának javítását szolgáló inaktív elválasztók használatakor az alábbi szabályokat kell betartani:

- Egy sorra nem szabad egynél több utasítást írni
- A komplex utasítások eltolására tabulátorokat kell használni
- A sorok vagy bekezdések olvashatóságának javítására megjegyzéseket kell beilleszteni

### B.7.2. Kifejezések és zárójelek

Az ST kifejezések ST **operátorokat** és változó vagy konstans **operandusokat** kombinálnak. Minden egyes kifejezéshez (operandusokat egy ST operátorral kombinálva), az operandusok **típusának** ugyanannak kell lennie. Ennek az egyszeres kifejezésnek ugyanaz a típusa, mint az operandusé, és az használható összetettebb kifejezésben. Például:

```
(boo_var1 AND boo_var2) BOO típusa van
not (boo_var1)           BOO típusa van
(sin (3.14) + 0.72)      VALÓS ANALÓG típusa van
```

$(t\#1s23 + 1.78)$ 

egy érvénytelen kifejezés

A **Zárójelek** a kifejezés részleteinek elkülönítésére, valamint a műveletek explicit sorrendbe állítására szolgálnak. Ha egy komplex kifejezésnek nincsenek zárójelei, akkor a műveleti sorrendet értelemszerűen az ST operátorok közötti alapértelmezett **prioritás** adja meg. Például:

 $2 + 3 * 6$ 

egyenlő  $2+18=20$  -al

mivel a szorzás operátornak nagyobb prioritása van

 $(2+3) * 6$ 

egyenlő  $5*6=30$  -al

a prioritást a zárójelek határozzák meg

Figyelmeztetés: Egy kifejezésen belül maximum **8** zárójel-szint építhető egymásba.

### B.7.3. Funkciók vagy funkcióblokkok meghívása

A következő tárgyak mindegyikéhez használhatók standard ST funkció meghívások:

- Alprogramok
- IEC nyelveken írt könyvtár funkciók és blokkok
- „C” Funkciók és funkcióblokkok
- Típus konverzió funkciók

#### ≡ **Alprogramok vagy funkciók meghívása**

<b>Név:</b>	a meghívott alprogram neve vagy könyvtár funkció IEC nyelven vagy „C”-ben írva
<b>Jelentése:</b>	egy ST, IL, LD vagy FBD alprogramot, illetve funkciót vagy „C” funkciót hív meg és megszerzi annak visszatérési értékét
<b>Szintaxis:</b>	<b>&lt;variable&gt; := &lt;subprog&gt; (&lt;par1&gt;, ... &lt;parN&gt; );</b>
<b>Operandusok:</b>	A visszatérési értéknek és meghívási paraméternek követnie kell az alprogram által definiált interfészt:
<b>Visszatérési érték:</b>	az alprogram által visszaküldött érték

Az alprogram hívások bármilyen kifejezésben használhatók. Ezek bármilyen SFC átmenetben is alkalmazhatók.

1. példa: Alprogram meghívás

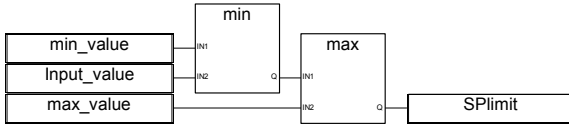
(\* Fő ST program: \*)

(\* egy analóg értéket szerez be, és azt egy korlátozott időértékre konvertál \*)

ana\_timeprog := SPLimit ( tprog\_cmd );

appl\_timer := tmr (ana\_timeprog \* 100);

(\* 'SPLimit' nevű FBD programot hívott \*)



## 2. példa: Funkció hívás

(\* komplex kifejezésekben használt funkciók:: min, max, right, mlen és left, standard „C” funkciók \*)

limited\_value := min (16, max (0, input\_value) );

rol\_msg := right (message, mlen (message) - 1) + left (message, 1);

### **Funkcióblokkok meghívása:**

**Név:** A funkcióblokk előfordulás neve

**Jelentése:** egy funkcióblokkot hív meg az ISaGRAF könyvtárból, vagy a felhasználói könyvtárból, és eléri annak visszatérési paramétereit

**Szintaxis:** (\* a funkcióblokk meghívása \*)

<blockname> ( <p1>, <p2> ... );

(megszerzi annak visszatérési paramétereit \*)

<result> := <blockname>. <ret\_param1>;

...

<result> := <blockname>. <ret\_paramN>;

**Operandusok:** a paraméterek olyan kifejezések, amelyek megfelelnek az ahhoz a funkcióblokkhoz specifikált paramétereknek

**Visszatérési érték:** Lásd a Visszatérési paraméterek beszerzésének szintaxisát.

Az egyes funkcióblokk paraméterek jelentéseit és típusát az ISaGRAF könyvtárban lehet megtalálni. A funkcióblokk előfordulását (a másolat nevét) a szótárban kell deklarálni

Példa:

(\* Egy funkcióblokkot meghívó ST program \*)

(\* a blokk előfordulását a szótárban deklarálni kell: \*)

(\* trigb1 : block R\_TRIG –felfutó él detektálás \*)

(\* funkcióblokk aktiválás ST nyelvből \*)

trigb1 (b1);

(\* hozzáférés visszatérési paraméterhez \*)

If (trigb1.Q) Then nb\_edge := nb\_edge + 1; End\_if;

## **B.7.4. ST specifikus logikai operátorok**

A következő logikai kifejezések az St nyelvre specifikusak:

- REDGE felfutó él detektálás

- FEDGE lefutó él detektálás

Egyéb standard logikai operátorok, pl.:

- NOT logikai tagadás

- AND (&)                      logikai ÉS  
 - OR                            logikai VAGY  
 - XOR                          logikai kizárólagos VAGY  
 használhatók. Ezek leírása a „Standard operátorok, funkcióblokkok és funkciók” című fejezetben található.

### ≡ **"REDGE" operátor**

**Név:** REDGE  
**Jelentése:** egy komplett logikai kifejezés felfutó élet értékeli ki  
**Szintaxis:** `<edge> := REDGE ( <boo_expression>, <memo_variable> );`  
**Operandusok:** az első operandus bármilyen logikai változó vagy komplex kifejezés  
 a második operandus egy belső logikai változó, amely a kifejezés  
 utolsó állapotának tárolására szolgál  
**Visszatérési érték:** IGAZ, amikor a kifejezés HAMISról IGAZra vált  
 Minden más esetben HAMIS

Egy kifejezés felfutó éle egynél többször nem detektálható ugyanabban a végrehajtási ciklusban a REDGE operátor használatával. Ez az operátor használható egy SFC átmenethez csatolt feltétel leírására.

**Figyelmeztetés:** A kifejezés utolsó állapotának tárolására használt „memória” logikai változó nem használható különböző kifejezések éleinek elindítására.

Amikor a kifejezés egy „xxx” nevű kifejezés, akkor ehhez a változóhoz a REDGE kifejezésekben egy „EDGE\_xxx” nevű egyedi belső változót kell deklarálni és használni. Ez a módszer biztosítja, hogy a memóriaváltozó nem kerül felülírásra az egyéb REDGE kiértékelések során.

Példa:

(\* REDGE operátort használó ST program \*)

(\* ez a program egy logikai input felfutó éleit számolja \*)

(\* Bi120 egy input logikai változó \*)

(\* Edge\_Bi120 a Bi120 változó állapot memóriája \*)

If REDGE (Bi120, Edge\_Bi120) Then

Counter := Counter + 1;

End\_if;

Megjegyzés : ez az operátor nem az IEC1131-3 normában van. Esetleg inkább az R\_TRIG standard blokk használható. Ez illeszthetőségi okokból lett meg hagyva.

### ≡ **"FEDGE" operátor**

**Név:** FEDGE  
**Jelentése:** egy logikai kifejezés felfutó élet értékeli ki  
**Szintaxis:** `<edge> := FEDGE ( <boo_expression>, <memo_variable> );`  
**Operandusok:** az első operandus bármilyen logikai változó vagy komplex kifejezés  
 a második operandus egy belső logikai változó, amely  
 a kifejezés utolsó állapotának tárolására szolgál

**Visszatérési érték:** IGAZ, amikor a kifejezés IGAZ-ról HAMIS-ra vált  
Minden más esetben HAMIS

Egy kifejezés lefutó éle egynél többször nem detektálható ugyanabban a végrehajtási ciklusban a REDGE operátor használatával. Ez az operátor használható egy SFC átmenethez csatolt feltétel leírására.

**Figyelmeztetés:** A kifejezés utolsó állapotának tárolására használt „memória” logikai változó nem használható különböző kifejezések éleinek elindítására.

Amikor a kifejezés egy „xxx” nevű kifejezés, akkor ehhez a változóhoz a FEDGE kifejezésekben egy „EDGE\_xxx” nevű egyedi belső változót kell deklarálni és használni. Ez a módszer biztosítja, hogy a memóriaváltozó nem kerül felülírásra az egyéb FEDGE kiértékelések során.

Példa:

(\* FEDGE operátort használó ST program \*)

(\* ez a program egy logikai input lefutó éleit számolja \*)

(\* Bi120 egy input logikai változó \*)

(\* Edge\_Bi120 a Bi120 változó állapot memóriája \*)

```
If FEDGE (Bi120, Edge_Bi120) Then
    Counter := Counter + 1;
End_if;
```

Megjegyzés : ez az operátor nem az IEC1131-3 normában van. Esetleg inkább az F\_TRIG standard blokk használható. Ez illeszthetőségi okokból lett meghagyva.

## B.7.5. Alapvető ST utasítások

Az ST nyelv alapvető utasításai a következők:

- Kijelölés
- RETURN utasítás
- IF-THEN-ELSIF-ELSE struktúra
- CASE utasítás
- WHILE iterációs utasítás
- REPEAT iterációs utasítás
- FOR iterációs utasítás
- EXIT utasítás

### **Kijelölés**

<b>Név:</b>	<b>:=</b>
<b>Jelentése:</b>	egy kifejezéshez kijelöl egy változót
<b>Szintaxis:</b>	<b>&lt;variable&gt; := &lt;any_expression&gt; ;</b>
<b>Operandusok:</b>	a változónak belső vagy output változónak kell lennie, és a kifejezésnek ugyanolyan típusnak kell lennie

A kifejezés lehet egy hívás egy alprogramra vagy funkcióra az ISaGRAF könyvtárból

Példa:

```
(* ST program, kijelölésekkel *)
```

```
(* változó <=< változó *)
```

```
bo23 := bo10;
```

```
(* változó <=< kifejezés *)
```

```
bo56 := bx34 OR alrm100 & (level >= over_value);
```

```
result := (100 * input_value) / scale;
```

```
(* kijelölés alprogram visszatérési értékkel *)
```

```
rc := PSelect ( );
```

```
(* kijelölés funkcióblokk meghívással *)
```

```
limited_value := min (16, max (0, input_value) );
```

## **RETURN utasítás**

**Név:** RETURN

**Jelentése:** Felfüggeszti a pillanatnyi program végrehajtását.

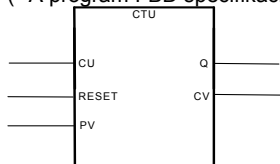
**Szintaxis:** RETURN ;

**Operandusok:** (nincs)

Egy SFC műveleti blokkban a RETURN utasítás csak annak a blokknak a végrehajtás befejezését jelzi.

Példa:

```
(* A program FBD specifikációja *) programozható számláló
```



```
(* A program ST kivitelezése, RETURN utasítást alkalmazva *)
```

```
If not (CU) then
```

```
    Q := false;
```

```
    CV := 0;
```

```
    RETURN; (* befejezi a programot *)
```

```
end_if;
```

```
if R then
```

```
    CV := 0;
```

```
else
```

```
    if (CV < PV) then
```

```
        CV := CV + 1;
```

```

    end_if;
end_if;
Q := (CV >= PV);

```

### IF-THEN-ELSIF-ELSE utasítás

**Név:** IF ... THEN ... ELSIF ... THEN ... ELSE ... END\_IF

**Jelentése:** két ST utasítás lista egyikét hajtja végre a kiválasztás egy logikai kifejezés értéke alapján történik

**Szintaxis:**

```

IF <boolean_expression> THEN
    <utasítás>;
    <utasítás>;
    ...
ELSIF <boolean_expression> THEN
    <utasítás>;
    <utasítás>;
    ...
ELSE
    <utasítás>;
    <utasítás>;
    ...
END_IF;

```

Az ELSE és ELSIF utasítások opcionálisak. Ha az ELSE utasítás nincs beírva, akkor nem lesz végrehajtv utasítás, ha a feltétel HAMIS.

Példa:

(\* IF utasítást használó ST program \*)

```

IF manual AND not (alarm) THEN
    level := manual_level;
    bx126 := bi12 OR bi45;
ELSIF over_mode THEN
    level := max_level;
ELSE
    level := (lv16 * 100) / scale;
END_IF;

```

(\* IF struktúra, ELSE nélkül \*)

```

If overflow THEN
    alarm_level := true;
END_IF;

```

### CASE utasítás

**Név:** CASE ... OF ... ELSE ... END\_CASE

**Jelentése:** több ST utasítás lista egyikét hajtja végre a kiválasztás egy integer kifejezés alapján történik

**Szintaxis:**

```

CASE <integer_expression> OF
    <érték> : <utasítások>;

```

```
<érték> , <érték> : <utasítások> ;  
...  
ELSE  
  <utasítások> ;  
END_CASE;
```

A Case értékeknek integer konstans kifejezéseknek kell lenniük. Több, vesszővel elválasztott érték vezethet ugyanahhoz az utasításlistához. Az ELSE utasítás opcionális.

Példa:

(\* CASE utasítást használó ST program \*)

```
CASE error_code OF  
  255:   err_msg := 'Division by zero';  
        fatal_error := TRUE;  
  1:     err_msg := 'Overflow';  
  2, 3:  err_msg := 'Bad sign';  
ELSE  
  err_msg := 'Unknown error';  
END_CASE;
```

## **WHILE utasítás**

**Név:** **WHILE ... DO ... END\_WHILE**  
**Jelentése:** iteráció struktúra egy ST utasítás csoporthoz  
A „folytatás” feltétel minden iteráció ELŐTT van kiértékelve  
**Szintaxis:** **WHILE <boolean\_expression> DO**  
 <utasítás> ;  
 <utasítás> ;  
 ...  
**END\_WHILE ;**

Figyelmeztetés: Mivel az ISaGRAF egy **szinkronos** rendszer, ezért az input változók nincsenek frissítve a WHILE iterációk közben. Egy input változó állapotának változása nem használható egy WHILE utasítás feltételének a leírására.

Példa:

(\* WHILE utasítást használó ST program \*)

(\* ez a program specifikus „C” funkciókat használ a karakterek olvasására \*)  
(\* egy soros porton \*)

```
string := ""; (* üres karaktersor *)  
nbchar := 0;  
  
WHILE ((nbchar < 16) & ComIsReady ( )) DO  
  string := string + ComGetChar ( );  
  nbchar := nbchar + 1;  
END_WHILE ;
```



## REPEAT utasítás

**Név:** REPEAT ... UNTIL ... END\_REPEAT  
**Jelentése:** iteráció struktúra egy ST utasítás csoporthoz  
 A „folytatás” feltétel minden iteráció UTÁN van kiértékelve  
**Szintaxis:** REPEAT  
                   <utasítás> ;  
                   <utasítás> ;  
                   ...  
                   UNTIL <boolean\_condition>  
                   END\_REPEAT ;

Figyelmeztetés: Mivel az ISaGRAF egy **szinkronizált** rendszer, ezért az input változók nincsenek frissítve a REPEAT iterációk közben. Egy input változó állapotának változása nem használható egy REPEAT utasítás befejezési feltételének a leírására.

Példa:

(\* REPEAT utasítást használó ST program \*)

(\* Ez a program specifikus „C” funkciókat használ a karakterek olvasására \*)  
 (\* egy soros porton \*)

```
string := ""; (* üres karaktorsor *)
nbchar := 0;
IF ComIsReady ( ) THEN
  REPEAT
    string := string + ComGetChar ( );
    nbchar := nbchar + 1;
  UNTIL ( (nbchar >= 16) OR NOT (ComIsReady ( )) )
  END_REPEAT ;
END_IF;
```



## FOR utasítás

**Név:** FOR ... TO ... BY ... DO ... END\_FOR  
**Jelentése:** korlátozott számú iterációt hajt végre,  
 egy integer analóg indexváltozót használva  
**Szintaxis:** FOR <index> := <mini> TO <maxi> BY <step> DO  
                   <utasítás> ;  
                   <utasítás> ;  
                   END\_FOR;  
**Operandusok:** **index:** belső analóg változó, bármely hurokban növelve  
**mini:** az index kiindulási értéke (az első hurok előtt)  
**maxi:** az indexhez megengedett maximális érték  
**step:** az index növekedése az egyes hurokokban

A [ BY step ] utasítás opcionális. Ha nincs specifikálva, akkor a növekedési lépés 1.

Figyelmeztetés: Mivel az ISaGRAF egy szinkronizált rendszer, ezért az input változók nincsenek frissítve a FOR iterációk közben.

Egy FOR utasítás „while” megfelelője a következő:

```

index := mini;
while (index <= maxi) do
    <utasítás> ;
    <utasítás> ;
    index := index + step;
end_while;

```

Példa:

(\* FOR utasítást használó ST program \*)  
 (\* ez a program egy karaktersor számjegyekaraktereit vonja ki \*)

```

length := mlen (message);
target := ""; (* üres karaktersor *)
FOR index := 1 TO length BY 1 DO
    code := ascii (message, index);
    IF (code >= 48) & (code <= 57) THEN
        target := target + char (code);
    END_IF;
END_FOR;

```

## ≡ **EXIT utasítás**

**Név:** **EXIT**  
**Jelentése:** kilép egy FOR, WHILE vagy REPEAT iteráció utasításból  
**Szintaxis:** **EXIT**;  
**Operandusok:** (nincs)

Az EXIT általában egy IF utasításon belül használatos, egy FOR, WHILE vagy REPEAT blokkban.

Példa:

(\* EXIT utasítást használó ST program \*)  
 (\* Ez a program egy karaktert keres egy karaktersorban \*)

```

length := mlen (message);
found := NO;
FOR index := 1 TO length BY 1 DO
    code := ascii (message, index);
    IF (code = searched_char) THEN
        found := YES;
        EXIT;
    END_IF;
END_FOR;

```

## B.7.6. ST bővítések

A következő funkciók az ST nyelv bővítései:

- TSTART - TSTOP: időzítő vezérlés

Egy gyermek SFC program vezérléséhez a következő utasítások és funkciók állnak rendelkezésre. Ezek az ACTION()-on belül alkalmazhatók: ... END\_ACTION; blokkok SFC lépésekben.

- GSTART	egy SFC programot indít
- GKILL	leállít egy SFC programot
- GFREEZE	befagyaszt egy SFC programot
- GRST	egy befagyasztott SFC programot újra indít
- GSTATUS	beszerzi egy SFC program pillanatnyi státuszát

**Figyelmeztetés:** Ezek a funkciók nem az IEC1131-3 normában vannak.

Egyszerű ekvivalens található a GSTART-hoz és a GKILL-hez, az SFC lépésben a következő szintaxist használva:

```
child_name(S); (* ekvivalens a GSTART(child_name)-el; *)
child_name(R); (* equivalentens a GKILL(child_name)-el; *)
```

Egy SFC lépés státuszának eléréséhez a következő mezők használhatók:

**GSnnn.x** logikai érték, amely a lépés tevékenységét jelenti  
**GSnnn.t** a lépés legutóbbi aktiválása óta eltelt idő  
 ("nnn" az SFC lépés referencia száma)

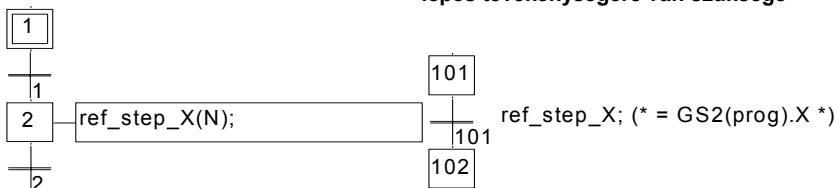
Lehetséges egy másik SFC programban deklarált lépés tevékenységének a tesztelése is, a következő szintaxis használatával:

**GSnnn(progname).x**

**Figyelmeztetés:** egy másik program lépésének ezzel a szintaxissal történő hivatkozása nem az IEC 1131-3 normában van. Annak az IEC szabályok betartásával történő elvégzésének egyszerű módja egy globális logikai változó deklarálásával történik abban a szótárban, amely a tesztelt lépés tevékenységét fogja képviselni (pl. ref\_step\_X). Ezután a lépésbe be kell illeszteni a változót az N minősítővel (ref\_step\_X(N);). Aztán a változó használható a lépés tevékenységét tesztelni akaró programban.

**Prog program**

**a másik program, amelynek a Prog program lépés tevékenységére van szüksége**

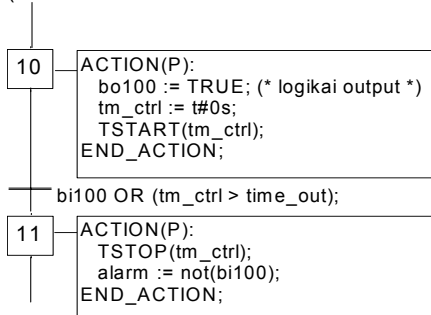


## **TSTART utasítás**

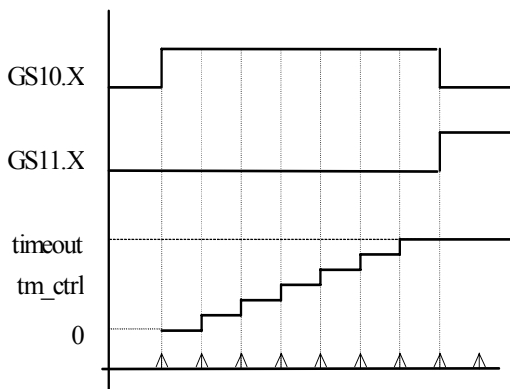
<b>Név:</b>	<b>TSTART</b>
<b>Jelentése:</b>	elindítja az időzítés változó számlálását az időzítés értéket a TSTART parancs módosítja, azaz a számlálás az időzítő pillanatnyi értékétől indul.
<b>Szintaxis:</b>	<b>TSTART ( &lt;timer_variable&gt; );</b>
<b>Operandusok:</b>	bármely inaktív időzítő változó
<b>Visszatérési érték:</b>	(nincs)

Példa:

(\* TSTART és TSTOP utasításokat használó SFC program \*)



Az idődiagram if bi100 mindig HAMIS:



Az időzítő egy ciklus közben ugyanazt az értéket tartja meg.

## **TSTOP utasítás**

<b>Név:</b>	<b>TSTOP</b>
<b>Jelentése:</b>	megállítja egy időzítő változó frissítését az időzítő értékét a TSTOP parancs nem módosítja
<b>Szintaxis:</b>	<b>TSTOP ( &lt;timer_variable&gt; );</b>

**Operandusok:** bármely aktív időzítő változó  
**Visszatérési érték:** (nincs)

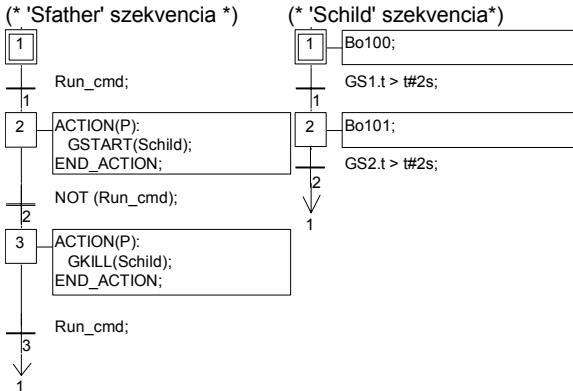
Példa: Lásd a TSTART-nál (a fentebb leírt funkcionál)

## **GSTART utasítás**

**Név:** **GSTART**  
**Jelentése:** elindít egy gyermek SFC programot, egy vezérlőjel elhelyezésével annak mindegyik kezdeti lépésébe  
**Szintaxis:** **GSTART ( <child\_program> );**  
**Operandusok:** A specifikált SFC programnak annak a programnak a gyermekének kell lenni, amelyben az utasítás van írva  
**Visszatérési érték:** (nincs)

A gyermek program gyermekeit a GSTART utasítás nem indítja el automatikusan.  
 Megjegyzés : Mivel a GSTART nem az IEC 1131-3 normában van, ezért egy gyermek SFC elindításához inkább az S minősítőt kell használni, az alábbi szintaxissal:  
 Child\_name(S);

Példa: A GSTART és GKILL használata



## **GKILL utasítás**

**Név:** **GKILL**  
**Jelentése:** leállít egy gyermek SFC programot, a vezérlőjelek eltávolításával, amelyek pillanatnyilag annak lépéseiben léteznek  
**Szintaxis:** **GKILL ( <child\_program> );**  
**Operandusok:** A specifikált SFC programnak annak a programnak a gyermekének kell lenni, amelyben az utasítás van írva  
**Visszatérési érték:** (nincs)

A gyermek program gyermekeit a specifikált programmal automatikusan leállítódnak.  
 Megjegyzés : Mivel a GKILL nem az IEC 1131-3 normában van, ezért egy gyermek SFC leállításához inkább az R minősítőt kell használni, az alábbi szintaxissal:

Child\_name(R);

Példa: Lásd a GSTART-nál (a fentebb leírt funkciónál)

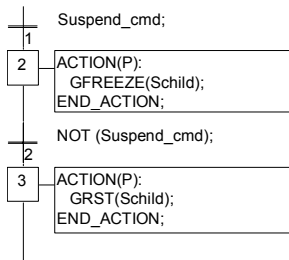
### **GFREEZE utasítás**

**Név:** GFREEZE  
**Jelentése:** Felfüggeszti egy gyermek SFC program végrehajtását. A befagyasztott program a GRST utasítással indítható újra.  
**Szintaxis:** GFREEZE ( <child\_program> );  
**Operandusok:** A specifikált SFC programnak annak a programnak a gyermekének kell lennie, amelyben az utasítás van írva  
**Visszatérési érték:** (nincs)

A gyermek program gyermekeit a specifikált programmal automatikusan befagyasztódnak.

Megjegyzés: A GFREEZE nem az IEC 1131-3 normában van.

Példa:



### **GRST utasítás**

**Név:** GRST  
**Jelentése:** újraindít egy gyermek SFC programot, amely a GFREEZE utasítással lett befagyasztva.  
**Szintaxis:** GRST ( <child\_program> );  
**Operandusok:** A specifikált SFC programnak annak a programnak a gyermekének kell lennie, amelyben az utasítás van írva  
**Visszatérési érték:** (nincs)

A gyermek program gyermekeit a GRST utasítás automatikusan újraindítja.

Megjegyzés: A GRST nem az IEC 1131-3 normában van.

Példa: Lásd a GFREEZE-nél (a fentebb leírt funkciónál)

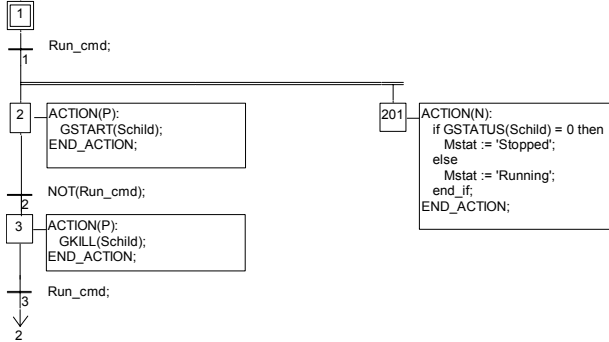
### **GSTATUS utasítás**

**Név:** GSTATUS  
**Jelentése:** egy SFC program pillanatnyi státuszát adja vissza  
**Szintaxis:** <ana\_var> := GSTATUS ( <child\_program> );  
**Operandusok:** A specifikált SFC programnak annak a programnak a gyermekének kell lennie, amelyben az utasítás van írva

**Visszatérési érték:**    0 = a program inaktív (le lett állítva)  
                               1 = a program aktív (el lett indítva)  
                               2 = a program be van fagyaszthatva

Megjegyzés: A GSTATUS nem az IEC 1131-3 normában van.

**Példa:**



## B.8. Az IL nyelv

Az **Utasításlista**, vagy IL egy alacsony szintű nyelv. Az utasítások mindig a **pillanatnyi eredményre** (vagy IL regiszterre) vonatkoznak. Az operátor a pillanatnyi érték és az operandus között végrehajtandó műveletet jelzi. A művelet eredménye ismét a pillanatnyi eredményben tárolódik el.

### B.8.1. IL fő szintaxis

Egy IL program **utasítások** listája. Mindegyik utasításnak egy új soron kell kezdődnie, és egy **operátort** kell tartalmaznia, amelyet tetszés szerint **módosítók**, valamint szükség esetén az adott művelethez vesszőkkel („,”) elválasztott **operandusok** egészítenek ki. Az utasítást egy **címke** előzheti meg, amelyet egy kettőspont („:”). követhet. Ha az utasításhoz egy **megjegyzés** van csatolva, akkor annak a sor utolsó komponensének kell lennie. A megjegyzések mindig „(”-el kezdődnek és „)”-el végződnek. Az utasítások közé üres sorok is bevihetők. Az üres sorokra megjegyzések helyezhetők el. Az alábbiakban az utasítás sorok példái láthatók:

<i>Címke</i>	<i>Operátor</i>	<i>Operandus:</i>	<i>Megjegyzések</i>
Kezdet	LD	IX1	(* nyomógomb *)
	ANDN	MX5	(* a parancs nincs tiltva *)
	ST	QX2	(* motor indítása*)

#### ☐ **Címkék**

Az utasítást egy **címke** előzheti meg, amelyet egy kettőspont („:”). követhet. A címke elhelyezhető egy üres sorra. A címkék bizonyos műveletek, pl. ugrások operandusaként használhatók. A címkék elnevezéseinek a következő szabályoknak kell eleget tenniük:

- a név nem haladhatja meg a **16** karaktert
- az első karakternek **betűnek** kell lennie
- a további karaktereknek **betűknek**, **számjegyeknek**, vagy aláhúzás („\_”) karaktereknek kell lenniük

Ugyanabban az IL programban ugyanaz a név nem használható egynél több címkéhez. Egy címke neve megegyezhet egy változó nevével.

#### ☐ **Operátor módosítók**

Az alábbiakban a rendelkezésre álló operátor módosítók vannak felsorolva. A módosító karakternek ki kell egészítenie az operátor nevét, és nem lehet közöttük üres hely:

- N** az operandus logikai tagadása
- (** késleltetett művelet
- +C** feltételes művelet

Az „**N**” módosító az operandus logikai tagadását jelzi. Az **ORN IX12** utasítás például a következőképpen értelmezendő: **eredmény := eredmény VAGY NEM (IX12)**.

A zárójel „(” módosító azt jelzi, hogy az utasítás kiértékelését a „)” záró zárójel operátor eléréséig késleltetni kell.

A „C” módosító azt jelzi, hogy a csatolt utasítást csak akkor kell végrehajtani, ha a pillanatnyi eredmény értéke logikai IGAZ (nem logikai értékek esetében 0-tól eltérő). A „C” módosító az „N” módosítóval kombinálható annak jelzésére, hogy az utasítást csak akkor kell végrehajtani, ha a pillanatnyi eredmény értéke logikai HAMIS (vagy nem logikai értékek esetében 0-tól eltérő).

### ≡ **Késleltetett műveletek**

Mivel csak egyetlen IL regiszter létezik (pillanatnyi eredmény), ezért bizonyos műveleteket esetleg késleltetni kell, hogy a végrehajtási sorrend vagy az utasítások megváltoztathatók legyenek. A késleltetett műveleteket zárójelek jelzik:

'(' egy módosító, a késleltetendő műveletet jelzi  
amely  
)' egy operátor, amely a késleltetett műveletet végrehajtja

A nyitó zárójel „(” módosító azt jelzi, hogy az utasítás kiértékelését a „)” záró zárójel operátor eléréséig késleltetni kell. Például a következő szekvencia:

```
AND(    IX12
OR      IX35
)
```

úgy értelmezendő, mint:

**eredmény := eredmény ÉS ( IX12 VAGY IX35 )**

### **B.8.2. IL operátorok**

A következő táblázat az IL nyelv standard operátorait foglalja össze:

<i>Operátor</i>	<i>Módosító k</i>	<i>Operandus:</i>	<i>Leírás</i>
LD	N	Változó, konstans	Betölti az operandust
ST	N	Változó	Tárolja a pillanatnyi eredményt
S		BOO változó	IGAZ-ra állít
R		BOO változó	Visszaállít HAMIS-ra
AND	N (	BOO	logikai ÉS
&	N (	BOO	logikai ÉS
OR	N (	BOO	logikai VAGY
XOR	N (	BOO	kizárólagos VAGY
ADD	(	változó, konstans	Összeadás
SUB	(	változó, konstans	Kivonás
MUL	(	változó, konstans	Szorzás
DIV	(	változó, konstans	Osztas

GT	(	változó, konstans	Teszt: >
GE	(	változó, konstans	Teszt: >=
EQ	(	változó, konstans	Teszt: =
LE	(	változó, konstans	Teszt <=
LT	(	változó, konstans	Teszt <
NE	(	változó, konstans	Teszt <>
CAL	C N	Funkcióblokk előfordulás neve	Meghív egy funkcióblokkot
JMP	C N	Címke	A címkére ugrik
RET	C N		Visszatér az alprogramtól
)			Végrehajtja a késleltetett műveletet

A következő fejezetben csak az IL nyelvre specifikus operátorok vannak leírva, a többi standard operátor a „Standard operátorok, funkcióblokkok és funkciók” című fejezetben található.

### **LD operátor**

**Művelet:** egy értéket tölt be a pillanatnyi eredménybe

**Engedélyezett módosítók:** N

**Operandus:** konstans kifejezés  
belső, input, vagy output változó

Példa:

```
(* LD MŰVELETEK PÉLDÁI *)
LDex:      LD      false      (* eredmény := HAMIS logikai konstans *)
           LD      true       (* eredmény := IGAZ logikai konstans *)
           LD      123        (* eredmény := integer konstans *)
           LD      123.1      (* eredmény := valós konstans *)
           LD      t#3ms      (* eredmény := idő konstans *)
           LD      boo_var1   (* eredmény := logikai változó *)
           LD      ana_var1   (* eredmény := analóg változó *)
           LD      tmr_var1   (* eredmény := időzítő változó *)
           LDN     boo_var1   (* eredmény := NEM (logikai változó) *)
```

### **ST operátor**

**Művelet:** a pillanatnyi eredményt egy változóban tárolja

ez a művelet nem módosítja a pillanatnyi eredményt

**Engedélyezett módosítók:** N

**Operandus:** belső, vagy output változó

Példa:

```
(* ST MŰVELETEK PÉLDÁI *)
STboo:     LD      false
           ST      boo_var1  (* boo_var1 := HAMIS *)
           STN     boo_var1  (* boo_var1 := IGAZ *)
STTana:     LD      123
           ST      ana_var1  (* ana_var1 := 123 *)
STtmr:     LD      t#12s
```

ST            tmr\_var1    (\* tmr\_var1 := t#12s \*)

### **S operátor**

**Művelet:** az IGAZ logikai értéket egy logikai változóban tárolja, ha a pillanatnyi eredmény logikai értéke IGAZ. Ha a pillanatnyi eredmény HAMIS, akkor nem lesz művelet feldolgozva. Ez a művelet nem módosítja a pillanatnyi eredményt

**Engedélyezett módosítók:** (nincs)

**Operandus:** output, vagy belső logikai változó

Példa:

```
(* S MŰVELETEK PÉLDÁI *)
SETex:      LD      true      (* pillanatnyi eredmény := IGAZ *)
             S       boo_var1 (* boo_var1 := TRUE *)
             (* pillanatnyi eredmény nincs módosítva *)
             LD      hamis     (* pillanatnyi eredmény := HAMIS *)
             S       boo_var1 (* semmi sem lett elvégezve - boo_var1 nem változott *)
*)
```

### **R operátor**

**Művelet:** az IGAZ logikai értéket egy logikai változóban tárolja, ha a pillanatnyi eredmény logikai értéke IGAZ. Ha a pillanatnyi eredmény HAMIS, akkor nem lesz művelet feldolgozva. Ez a művelet nem módosítja a pillanatnyi eredményt

**Engedélyezett módosítók:** (nincs)

**Operandus:** output, vagy belső logikai változó

Példa:

```
(* R MŰVELETEK PÉLDÁI *)
RESETex:    LD      true      (* pillanatnyi eredmény := IGAZ *)
             R       boo_var1 (* boo_var1 := HAMIS *)
             (* pillanatnyi eredmény nincs módosítva *)
             ST      boo_var1 (* boo_var1 := IGAZ *)
             LD      false     (* pillanatnyi eredmény := HAMIS *)
             R       boo_var1 (* semmi sem történt - boo_var1 nem változott *)
```

### **JMP operátor**

**Művelet:** a megadott címkére ugrik

**Engedélyezett módosítók:** C N

**Operandus:** ugyanabban az IL programban definiált címke

Példa:

(\* a következő példa egy analóg választó értékét teszteli (0 vagy 1 vagy 2)

(\* 3 output logikai egyikének beállításához. Az „egyenlő 01-val” teszt a következővel van elvégezve \*)

(\* a JMP operátor \*)

JMPex:	LD	selector	(* a kiválasztó 0 vagy 1 vagy 2 *)
	BOO		(* konverzió logikaiba *)
	JMPC	test1	(* ha kiválasztó = 0 akkor *)
	LD	true	
	ST	bo0	(* bo0 := igaz *)
	JMP	JMPend	(* a program vége *)
test1:	LD	selector	
	SUB	1	(* a kiválasztó csökkentése: most 0 vagy 1 *)
	BOO		(* konverzió logikaiba *)
	JMPC	test2	(* ha kiválasztó = 0 akkor *)
	LD	true	
	ST	bo1	(* bo1 := igaz *)
	JMP	JMPend	(* a program vége *)
test2:	LD	true	(* utolsó lehetőség *)
	ST	bo2	(* bo2 := igaz *)
JMPend:			(* az IL program vége *)

### RET operátor

**Művelet:** Befejezi a pillanatnyi utasításlistát. Ha az IL szekvencia egy alprogram, akkor a pillanatnyi eredmény visszatér a meghívó programhoz

**Engedélyezett módosítók:** C N

**Operandus:** (nincs)

Példa:

(\* a következő példa egy analóg választó értékét teszteli (0 vagy 1 vagy 2)

(\* 3 output logikai egyikének beállításához. Az „egyenlő 01-val” teszt a következővel van elvégezve \*)

(\* a JMPC operátor \*)

JMPex:	LD	selector	(* a kiválasztó 0 vagy 1 vagy 2 *)
	BOO		(* konverzió logikaiba *)
	JMPC	test1	(* ha kiválasztó = 0 akkor *)
	LD	true	
	ST	bo0	(* bo := igaz *)
	RET		(* vége 0 visszatérés *)
			(* kiválasztó csökkentése *)
test1:	LD	selector	
	SUB	1	(* a kiválasztó most 0 vagy 1 *)
	BOO		(* konverzió logikaiba *)
	JMPC	test2	(* ha kiválasztó = 0 akkor *) LD igaz
	ST	bo1	(* bo1 := igaz *)
	LD	1	(* valós kiválasztó érték betöltése *)
	RET		(* vége 1 visszatérés *)
			(* utolsó lehetőség *)
test2:	RETNC		(* visszatér, ha a kiválasztó *)
			(* érvénytelen értékű *)
	LD	true	
	ST	bo2	(* bo2 := igaz *)
	LD	2	(* valós kiválasztó érték betöltése *)
			(* vége 2 visszatérés *)

## „)” operátor

**Művelet:** Egy késleltetett műveletet hajt végre. A késleltetett műveletről a „(„ adott értesítést  
**Engedélyezett módosítók:** (nincs)

**Operandus:** (nincs)

Példa:

(\* A következő program késleltetett műveleteket illeszt: \*)

(\* res := a1 + (a2 \* (a3 - a4) \* a5) + a6; \*)

Késleltetett:	LD	a1	(* eredmény := a1; *)
	ADD(	a2	(* késleltetett ADD - eredmény := a2; *)
	MUL(	a3	(* késleltetett MUL - eredmény := a3; *)
	SUB	a4	(* eredmény := a3 - a4; *)
	)		(* késleltetett MUL végrehajtása - eredmény := a2 * (a3-a4); *)
	MUL	a5	(* eredmény := a2 * (a3 - a4) * a5; *)
	)		(* késleltetett ADD végrehajtása *)
			(* eredmény := a1 + (a2 * (a3 - a4) * a5); *)
	ADD	a6	(* eredmény := a1 + (a2 * (a3 - a4) * a5) + a6; *)
	ST	res	(* pillanatnyi eredményt a res változóban tárolja *)

## Alprogramok vagy funkciók meghívása

Egy alprogram vagy egy funkció (az IL, ST, LD, FBD vagy „C” nyelvek valamelyikében írva) az IL nyelvből van meghívva, nevét használva operátorként.

**Művelet:** egy alprogramot vagy funkciót hajt végre – az alprogram vagy funkció által visszatérített érték az IL pillanatnyi eredményében kerül eltárolásra

**Engedélyezett módosítók:** (nincs)

**Operandus:** Az első meghívó paraméternek a hívás előtt a pillanatnyi eredményben kell eltárolva lennie. A következők az operandus mezőben vannak kifejezve, vesszővel elválasztva.

Példa:

(\* Meghívó program : egy analóg értéket időértékre konvertál \*)

Fő:	LD	bi0	
	SUBPRO	bi1,bi2	(* alprogram meghívása, az analóg érték beszerzéséhez *)
érték *)	ST	result	(* eredmény := az alprogram által visszaadott
	GT	vmax	(* teszt érték túlszordulás *)
	RETC		(* visszatérés ha túlszordulás *)
	LD	result	
	MUL	1000	(* másodpercek átváltása ezredmásodpercekké *)
	TMR		(* átváltás időzítőre *)
	ST	tmval	(* a konvertált értéket egy időzítőben tárolja *)

(\* a meghívott alprogram neve 'SUBPRO' : kiértékeli az analóg értéket \*)

(\* három logikai input: in0, in1, in2 bináris értékeként megadott érték az alprogram három logikai input paramétere \*)

```

LD      in2
ANA
MUL      2      (* eredmény = ana (in2); *)
ST      temporary (* ideiglenes := eredmény *)
LD      in1
ANA
ADD      temporary (* eredmény := 2*ana (in2) + ana (in1); *)
MUL      2      (* eredmény := 4*ana (in2) + 2*ana (in1); *)
ST      temporary (* ideiglenes := eredmény *)
LD      in0
ANA
ADD      temporary (* eredmény := 4*ana (in2) + 2*ana (in1)+ana (in0); *)
ST      SUBPRO  (* a pillanatnyi eredményt visszaadja a hívó
                  programnak *)

```

### **Funkcióblokkok meghívása: CAL operátor**

**Művelet:** egy funkcióblokkot hív meg

**Engedélyezett módosítók:** C N

**Operandus:** A funkcióblokk előfordulása neve.  
A blokk input paramétereit a meghívás előtt LD/ST műveleti szekvencia használatával ki kell jelölni.  
Az output paraméterek ismertek, ha használva vannak.

1. példa:

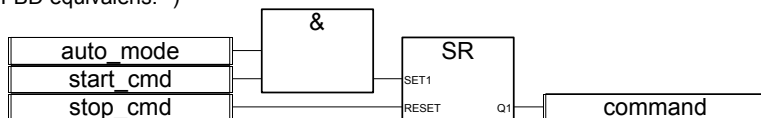
(\* SR funkcióblokk meghívása : az SR1 az SR egyik előfordulása \*)

```

LD      auto_mode
AND      start_cmd
ST      SR1.set1
LD      stop_cmd
ST      SR1.reset
CAL      SR1
LD      SR1.Q1
ST      command

```

(\* FBD equivalens: \*)



2. példa

(\*Feltételezzük, hogy R\_TRIG1 az R\_TRIG blokk egy előfordulása, és CTU1 a CTU blokk egy előfordulása\*)

```

LD      command
ST      R_TRIG1.clk
CAL      R_TRIG1

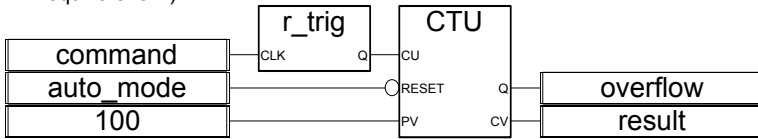
```

```

LD      R_TRIG1.Q
ST      CTU1.cu
LDN     auto_mode
ST      CTU1.reset
LD      100
ST      CTU1.pv
CAL     CTU1
LD      CTU1.Q
ST      overflow
LD      CTU1.cv
ST      result

```

(\* FBD equivalens: \*)



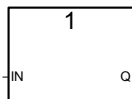
## B.9. Standard operátorok, funkcióblokkok, és funkciók

### B.9.1. Standard operátorok

Az alábbiakban az IEC nyelvek standard operátorai szerepelnek.

Adat manipuláció.....	Kijelölés, Analóg tagadás
Logikai műveletek.....	Logikai ÉS
	Logikai VAGY
	Logikai Kizárólagos VAGY
Aritmetikai műveletek .....	Összeadás
	Kivonás
	Szorzás
	Osztás
Logikai műveletek.....	Analóg bit-bit ÉS maszk
	Analóg bit-bit VAGY maszk
	Analóg bit-bit Kizárólagos VAGY maszk
	Bit-bit tagadás
Összehasonlítási tesztek: .....	Kisebb mint
	Kisebb mint, vagy egyenlő
	Nagyobb mint
	Nagyobb mint, vagy egyenlő
	Egyenlő
	Nem egyenlő
Adat konverzió.....	Konvertálás logikaira
	Konvertálás Integer analógra
	Konvertálás Valós analógra
	Konvertálás időzítőre
	Konvertálás üzenetre
Egyéb .....	Üzenet összefűzés
	Rendszer hozzáférés
	I/O csatoma működtetés

#### 1 nyereség



Argumentumok:

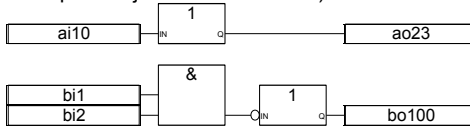
<b>IN</b>	bármilyen típus
<b>Q</b>	bármilyen típus

Leírás:

egy változó kijelölése egy másikba

Ez a blokk nagyon hasznos egy diagram inputjának és egy diagram outputjának a közvetlen összekapcsolására. Ez egy diagram outputhoz csatlakoztatott vonal állapotának a megfordítására is használható (egy logikai tagadás vonallal).

(\* FBD példa kijelölés Blokkokkal \*)



(\*ST ekvivalencia: \*)

ao23 := ai10;

bo100 := NOT (bi1 AND bi2);

(\* IL ekvivalencia \*)

LD ai10

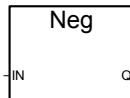
ST ao23

LD bi1

AND bi2

STN bo100

## NEG



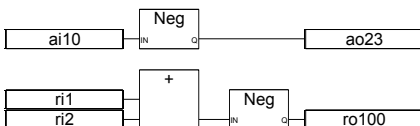
Argumentumok:

<b>IN</b>	INT-REAL	az inputnak és outputnak ugyanolyan formátumúnak kell lennie
<b>Q</b>	INT-REAL	

Leírás:

Egy változó tagadásának kijelölése.

(\* FBD példa, tagadás Blokkokkal \*)



(\*ST ekvivalencia: \*)

ao23 := - (ai10);

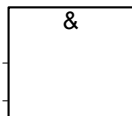
ro100 := - (ri1 + ri2);

(\* IL ekvivalencia \*)

LD ai10

MUL	-1
ST	ao23
LD	ri1
ADD	ri2
MUL	-1.0
ST	ro100

## & AND



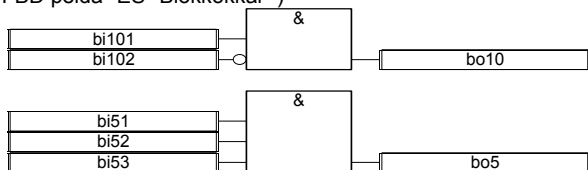
Megjegyzés : Ehhez az operátorhoz inputjainak a száma kettőnél többre kiterjeszthető.

Argumentumok: (Inputok)	LOGIKAI
output:	LOGIKAI az input kifejezések logikai ÉS-e

Leírás:

Logikai ÉS kettő vagy több kifejezés között.

(\* FBD példa "ÉS" Blokkokkal \*)



(\*ST ekvivalencia: \*)

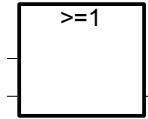
bo10 := bi101 AND NOT (bi102);

bo5 := (bi51 AND bi52) AND bi53;

(\* IL ekvivalencia \*)

LD	bi101	(* pillanatnyi eredmény := bi101 *)
ANDN	bi102	(* pillanatnyi eredmény := bi101 AND not(bi102) *)
ST	bo10	(* bo5 := pillanatnyi eredmény *)
LD	bi51	(* pillanatnyi eredmény := bi51; *)
&	bi52	(* pillanatnyi eredmény := bi51 AND bi52 *)
&	bi53	(* pillanatnyi eredmény := (bi51 AND bi52) AND bi53 *)
ST	bo5	(* bo5 := pillanatnyi eredmény *)

## >=1 OR



Megjegyzés : Ehhez az operátorhoz inputjainak a száma kettőnél többre kiterjeszthető.

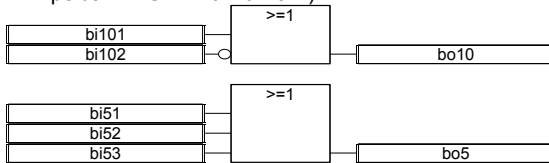
Argumentumok:

(Inputok) LOGIKAI  
output: LOGIKAI az input kifejezések logikai VAGY-ja

Leírás:

Kettő vagy több kifejezés logikai VAGY-ja.

(\* FBD példa "VAGY" Blokkokkal \*)



(\*ST ekvivalencia: \*)

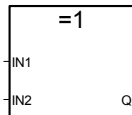
bo10 := bi101 OR NOT (bi102);

bo5 := (bi51 OR bi52) OR bi53;

(\* IL ekvivalencia \*)

LD bi101  
ORN bi102  
ST bo10  
LD bi51  
OR bi52  
OR bi53  
ST bo5

## =1 XOR



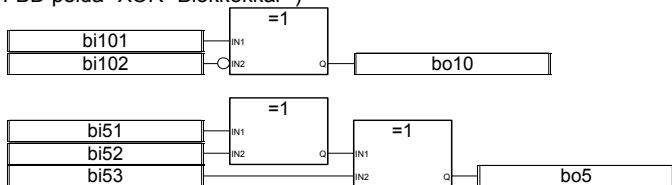
Argumentumok:

**IN1** LOGIKAI  
**IN2** LOGIKAI  
**Q** LOGIKAI a 2 input kifejezés logikai kizárólagos VAGY-ja

Leírás:

Két kifejezés közötti logikai kizárólagos VAGY.

(\* FBD példa "XOR" Blokkokkal \*)



(\*ST ekvivalencia: \*)

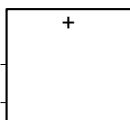
bo10 := bi101 XOR NOT (bi102);

bo5 := (bi51 XOR bi52) XOR bi53;

(\* IL ekvivalencia \*)

```
LD      bi101
XORN    bi102
ST      bo10
LD      bi51
XOR     bi52
XOR     bi53
ST      bo5
```

+



Megjegyzés : Ehhez az operátorhoz inputjainak a száma kettőnél többre kiterjeszthető.

Argumentumok:

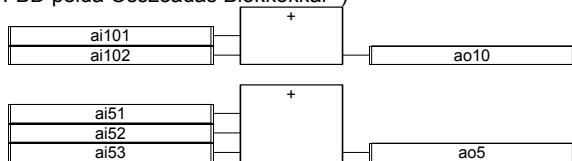
(inputok)      INT-REAL      lehet INTEGER vagy VALÓS (minden inputnak ugyanolyan formátumúnak kell lennie)

output:        INT-REAL      az input kifejezések előjeles összeadása

Leírás:

Kettő vagy több analóg változó összeadása.

(\* FBD példa Összeadás Blokkokkal \*)



```
(*ST ekvivalencia: *)
ao10 := ai101 + ai102;
ao5 := (ai51 + ai52) + ai53;
```

```
(* IL ekvivalencia *)
LD      ai101
ADD     ai102
ST      ao10
LD      ai51
ADD     ai52
ADD     ai53
ST      ao5
```

-



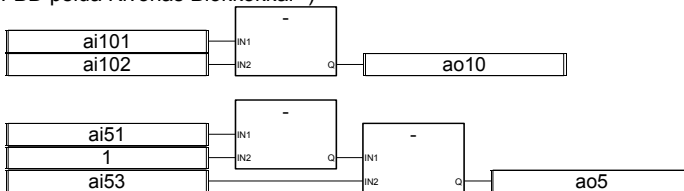
Argumentumok:

<b>IN1</b>	INT-REAL	lehet INTEGER vagy VALÓS
<b>IN2</b>	INT-REAL	(IN1 és IN2 ugyanolyan formátumúnak kell, hogy legyen)
<b>Q</b>	INT-REAL	kivonás (első - második)

Leírás:

Két analóg változó kivonása (első - második).

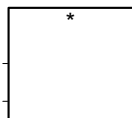
(\* FBD példa Kivonás Blokkokkal \*)



```
(*ST ekvivalencia: *)
ao10 := ai101 - ai102;
ao5 := (ai51 - 1) - ai53;
```

```
(* IL ekvivalencia *)
LD      ai101
SUB     ai102
ST      ao10
LD      ai51
SUB     1
SUB     ai53
ST      ao5
```

\*



Megjegyzés : Ehhez az operátorhoz inputjainak a száma kettőnél többre kiterjeszthető.

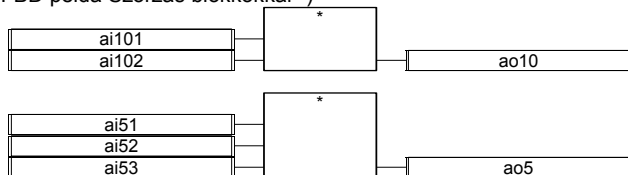
Argumentumok:

(inputok)	INT-REAL	lehet INTEGER vagy VALÓS (minden inputnak ugyanolyan formátumúnak kell lennie)
output:	INT-REAL	az input kifejezések előjeles szorzása

Leírás:

Két vagy több analóg változó szorzása.

(\* FBD példa Szorzás blokkokkal \*)



(\* ST equivalencia: \*)

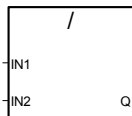
ao10 := ai101 \* ai102;

ao5 := (ai51 \* ai52) \* ai53;

(\* IL ekvivalencia \*)

```
LD      ai101
MUL     ai102
ST      ao10
LD      ai51
MUL     ai52
MUL     ai53
ST      ao5
```

/



Argumentumok:

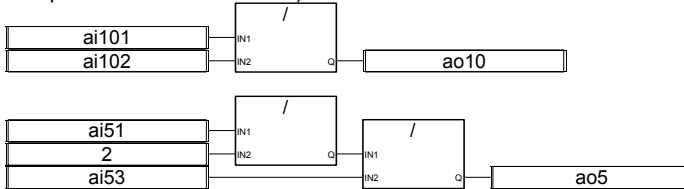
<b>IN1</b>	INT-REAL	lehet INTEGER vagy VALÓS (operandus)
<b>IN2</b>	INT-REAL	nem zéró analóg érték (osztó) (IN1 és IN2 ugyanolyan formátumúnak kell, hogy legyen)

**Q** INT-REAL előjeles integer vagy valós osztása IN2-vel

Leírás:

Két analóg változó osztása (az első osztva a másodikkal).

(\* FBD példa Osztás blokkokkal \*)



(\* ST Equivalencia \*)

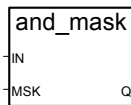
ao10 := ai101 / ai102;

ao5 := (ai51 / 2) / ai53;

(\* IL ekvivalencia \*)

```
LD      ai101
DIV     ai102
ST      ao10
LD      ai51
DIV     2
DIV     ai53
ST      ao5
```

## AND\_MASK



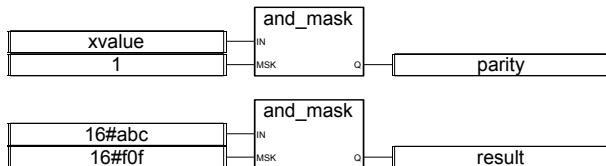
Argumentumok:

<b>IN</b>	INT	integer formátumúnak kell lennie
<b>MSK</b>	INT	integer formátumúnak kell lennie
<b>Q</b>	INT	bit-bit logikai ÉS az IN és az MSK között

Leírás:

Integer analóg ÉS bit-bit maszk.

(\* FBD példa Analóg AND\_MASK blokkokkal \*)



(\* ST Equivalencia \*)

paritás := AND\_MASK (xvalue, 1); (\* 1 ha xvalue páratlan \*)

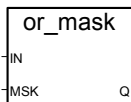
eredmény := AND\_MASK (16#abc, 16#0f); (\* egyenlő 16#a0c \*)

(\* IL ekvivalencia \*)

```

LD      xvalue
AND_MASK 1
ST      parity
LD      16#abc
AND_MASK 16#0f
ST      result
  
```

## OR\_MASK



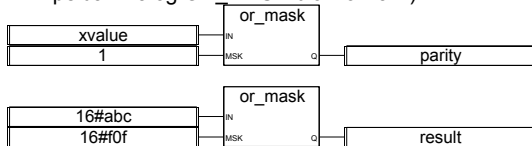
Argumentumok:

<b>IN</b>	INT	integer formátumúnak kell lennie
<b>MSK</b>	INT	integer formátumúnak kell lennie
<b>Q</b>	INT	bit-bit logikai VAGY az IN és az MSK között

Leírás:

Integer analóg VAGY bit-bit maszk.

(\* FBD példa Analóg OR\_MASK blokkokkal \*)



(\* ST Equivalencia \*)

is\_odd := OR\_MASK (xvalue, 1); (\* az értéket mindig páratlanná teszi \*)

eredmény := AND\_MASK (16#abc, 16#0f); (\* egyenlő 16#bf \*)

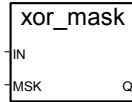
(\* IL ekvivalencia \*)

```

LD      xvalue
OR_MASK 1
ST      is_odd
LD      16#abc
  
```

OR\_MASK     16#f0f  
ST            result

## XOR\_MASK



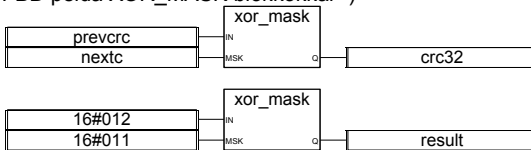
Argumentumok:

<b>IN</b>	INT	integer formátumúnak kell lennie
<b>MSK</b>	INT	integer formátumúnak kell lennie
<b>Q</b>	INT	bit-bit logikai Kizárólagos VAGY az IN és az MSK között

Leírás:

Integer analóg kizárólagos VAGY bit-bit maszk

(\* FBD példa XOR\_MASK blokkokkal \*)



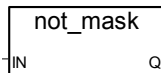
(\* ST Equivalencia \*)

```
crc32 := XOR_MASK (prevcrc, nextc);
result := XOR_MASK (16#012, 16#011); (* egyenlő 16#003 *)
```

(\* IL ekvivalencia \*)

```
LD      prevcrc
XOR_MASK nextc
ST      crc32
LD      16#012
XOR_MASK 16#011
ST      result
```

## NOT\_MASK



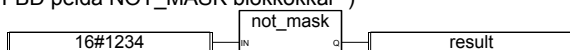
Argumentumok:

<b>IN</b>	INT	integer formátumúnak kell lennie
<b>Q</b>	INT	bit-bit tagadás az IN 32 bitjén

Leírás:

Integer analóg bit-bit tagadás maszk

(\* FBD példa NOT\_MASK blokkokkal \*)



(\*ST ekvivalencia: \*)

eredmény := NOT\_MASK (16#1234);

(\* az eredmény 16#FFFF\_EDCB \*)

(\* IL ekvivalencia \*)

LD 16#1234

NOT\_MASK

ST result

<



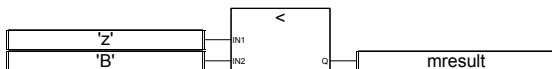
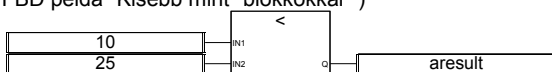
Argumentumok:

<b>IN1</b>	INT-REAL- TMR-MSG	
<b>IN2</b>	INT-REAL- TMR-MSG	(mindkét inputnak ugyanolyan típusúnak kell lennie)
<b>Q</b>	LOGIKAI	IGAZ ha IN1 < IN2

Leírás:

Ellenőrzi, hogy egy érték KISEBB MINT egy másik (analógon, időzítőn, vagy üzeneteken)

(\* FBD példa "Kisebb mint" blokkokkal \*)



(\* ST Ekvivalencia \*)

aresult := (10 < 25); (\* aresult IGAZ \*)

mresult := ('z' < 'B'); (\* mresult HAMIS \*)

(\* IL ekvivalencia \*)

LD 10

LT 25

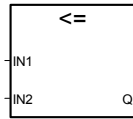
ST aresult

LD 'z'

LT 'B'

ST mresult

&lt;=



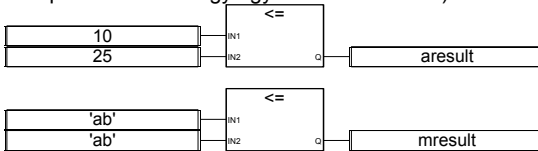
Argumentumok:

**IN1** INT-REAL-MSG  
**IN2** INT-REAL-MSG (mindkét inputnak ugyanolyan típusúnak kell lennie)  
**Q** LOGIKAI IGAZ, ha  $IN1 \leq IN2$

Leírás:

Ellenőrzi, hogy egy érték KISEBB MINT vagy EGYENLŐ egy másikkal (analógon, vagy üzeneteken)

(\* FBD példa "Kisebb vagy egyenlő" blokkokkal \*)



(\* ST Equivalencia \*)

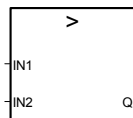
aresult := (10 &lt;= 25); (\* aresult IGAZ \*)

mresult := ('ab' &lt;= 'ab'); (\* mresult IGAZ \*)

(\* IL ekvivalencia \*)

LD 10  
 LE 25  
 ST aresult  
 LD 'ab'  
 LE 'ab'  
 ST mresult

&gt;



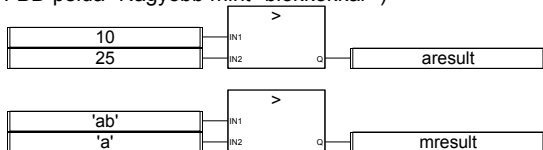
Argumentumok:

**IN1** INT-REAL-  
TMR-MSG  
**IN2** INT-REAL-  
TMR-MSG (mindkét inputnak ugyanolyan típusúnak kell lennie)  
**Q** LOGIKAI IGAZ ha  $IN1 > IN2$

Leírás:

Ellenőrzi, hogy egy érték NAGYOBB MINT egy másik (analogon, időzítőn, vagy üzeneteken)

(\* FBD példa "Nagyobb mint" blokkokkal \*)



(\* ST Equivalencia \*)

areult := (10 > 25); (\* areult HAMIS \*)

mresult := ('ab' > 'a'); (\* mresult IGAZ \*)

(\* IL ekvivalencia \*)

LD 10

GT 25

ST areult

LD 'ab'

GT 'a'

ST mresult

**>=**



Argumentumok:

**IN1** INT-REAL-MSG

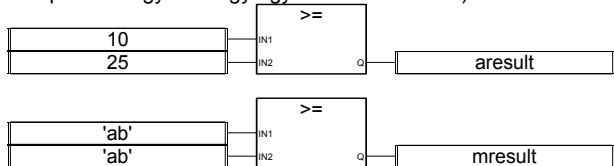
**IN2** INT-REAL-MSG (mindkét inputnak ugyanolyan típusúnak kell lennie)

**Q** LOGIKAI IGAZ ha IN1 >= IN2

Leírás:

Ellenőrzi, hogy egy érték NAGYOBB VAGY EGYENLŐ egy másikhoz képest (analogon, vagy üzeneteken)

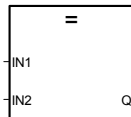
(\* FBD példa "Nagyobb vagy egyenlő" blokkokkal \*)



```
(* ST Equivalencia *)
aresult := (10 >= 25); (* aresult HAMIS *)
mresult := ('ab' >= 'ab'); (* mresult IGAZ *)
```

```
(* IL ekvivalencia *)
LD      10
GE      25
ST      aresult
LD      'ab'
GE      'ab'
ST      mresult
```

=



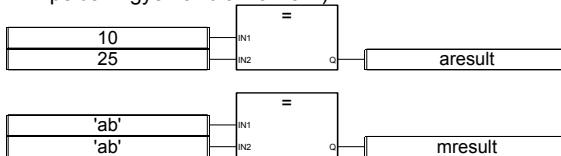
Argumentumok:

<b>IN1</b>	INT-REAL-MSG
<b>IN2</b>	INT-REAL-MSG (mindkét inputnak ugyanolyan típusúnak kell lennie)
<b>Q</b>	LOGIKAI IGAZ, ha IN1 = IN2

Leírás:

Ellenőrzi, hogy egy érték EGYENLŐ egy másikhoz képest (analogon, vagy üzeneteken)

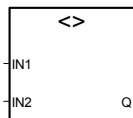
(\* FBD példa "Egyenlő" blokkokkal \*)



```
(* ST Equivalencia *)
aresult := (10 = 25); (* aresult HAMIS *)
mresult := ('ab' = 'ab'); (* mresult is IGAZ *)
```

```
(* IL ekvivalencia *)
LD      10
EQ      25
ST      aresult
LD      'ab'
EQ      'ab'
ST      mresult
```

&lt;&gt;



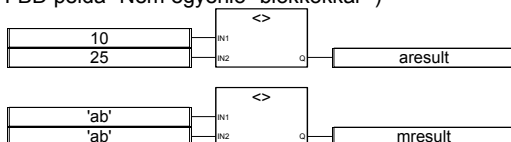
Argumentumok:

<b>IN1</b>	INT-REAL-MSG
<b>IN2</b>	INT-REAL-MSG (mindkét inputnak ugyanolyan típusúnak kell lennie)
<b>Q</b>	LOGIKAI IGAZ, ha első <> második

Leírás:

Ellenőrzi, hogy egy elem NEM EGYENLŐ egy másikhoz képest (analogon, vagy üzeneteken)

(\* FBD példa "Nem egyenlő" blokkokkal \*)



(\* ST Equivalencia \*)

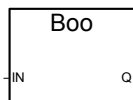
aresult := (10 &lt;&gt; 25); (\* aresult IGAZ \*)

mresult := ('ab' &lt;&gt; 'ab'); (\* mresult IGAZ \*)

(\* IL ekvivalencia \*)

LD	10
NE	25
ST	aresult
LD	'ab'
NE	'ab'
ST	mresult

## BOO



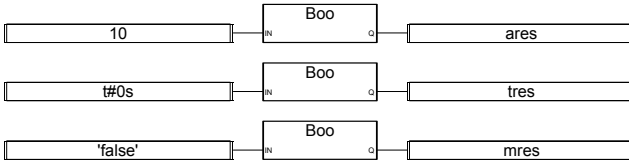
Argumentumok:

<b>IN</b>	ANY	bármely nem logikai érték
<b>Q</b>	BOO	IGAZ nem zero numerikus értékre FALSE zero numerikus értékre TRUE 'TRUE' üzenetre FALSE 'FALSE' üzenetre

Leírás:

Bármelyik változót logikaira konvertál

(\* FBD példa „Konvertálás logikai blokkokra” \*)



(\* ST Equivalencia \*)

ares := BOO (10);

tres := BOO (t#0s);

mres := BOO ('false');

(\* ares IGAZ \*)

(\* tres HAMIS \*)

(\* mres HAMIS \*)

(\* IL ekvivalencia \*)

LD 10

BOO

ST ares

LD t#0s

BOO

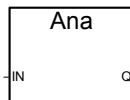
ST tres

LD 'false'

BOO

ST mres

## ANA



Argumentumok:

**IN** ANY

**Q** INT

bármely nem integer analóg érték

0 if IN is FALSE / 1 if IN is TRUE

ezredmásodpercek száma egy időzítőhöz

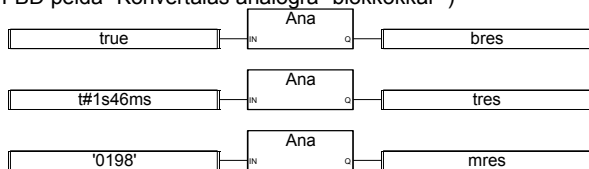
integer rész valós analóghoz

egy karaktersorral jelzett decimális szám

Leírás:

Bármely változót integerré alakít át

(\* FBD példa "Konvertálás analógra" blokkokkal \*)



(\* ST Equivalencia \*)

bres := ANA (true);

tres := ANA (t#1s46ms);

mres := ANA ('0198');

(\* bres 1 \*)

(\* tres 1046 \*)

(\* mres 198 \*)

(\* IL ekvivalencia \*)

LD true

ANA

ST bres

LD t#1s46ms

ANA

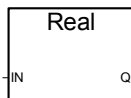
ST tres

LD '0198'

ANA

ST mres

## REAL



Argumentumok:

**IN**

BOO-INT-

TMR

**Q**

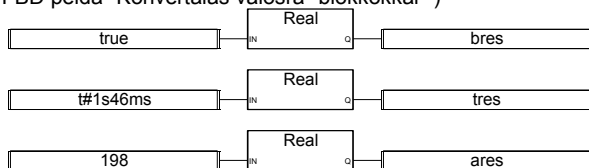
REAL

bármelyik nem valós analóg érték (nem üzenet)  
 0,0 ha IN HAMIS / 1,0 ha IN IGAZ  
 ezredmásodpercek száma egy időzítőhöz  
 azonos szám integer analóghoz

Leírás:

Bármely változót valósra konvertál

(\* FBD példa "Konvertálás valósra" blokkokkal \*)

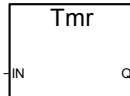


```
(* ST Equivalencia *)
bres := REAL (true);
tres := REAL (t#1s46ms);
ares := REAL (198);

(* bres 1.0 *)
(* tres 1046.0 *)
(* ares 198.0 *)
```

```
(* IL ekvivalencia *)
LD      true
REAL
ST      bres
LD      t#1s46ms
REAL
ST      tres
LD      198
REAL
ST      ares
```

## TMR



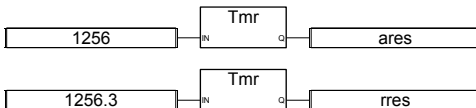
Argumentumok:

<b>IN</b>	INT-REAL	bármely nem időzítő érték IN (vagy IN integer része, ha az valós) a szám ezredmásodpercekben
<b>Q</b>	IDŐZÍTŐ	IN által képviselt időérték

Leírás:

Bármely analóg változót időzítőre konvertál

(\* FBD példa "Konvertálás időzítőre" blokkokkal \*)

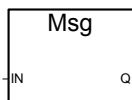


```
(* ST Equivalencia *)
ares := TMR (1256);
rres := TMR (1256.3);

(* ares := t#1s256ms *)
(* rres := t#1s256ms *)
```

```
(* IL Ekvivalencia *)
LD      1256
TMR
ST      ares
LD      1256.3
TMR
ST      rres
```

## MSG



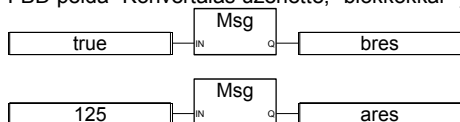
Argumentumok:

<b>IN</b>	BOO-INT-REA	bármely nem üzenet érték
<b>Q</b>	MSG	'false' vagy 'true' ha IN egy logikai decimális számbábrázolás, ha N analóg

Leírás:

Bármely változót egy üzenetté konvertál

(\* FBD példa "Konvertálás üzenetté;" blokkokkal \*)



(\* ST Equivalencia \*)

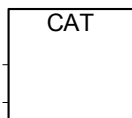
bres := MSG (true); (\* bres is 'IGAZ' \*)

ares := MSG (125); (\* ares '125' \*)

(\* IL ekvivalencia \*)

```
LD      true
MSG
ST      bres
LD      125
MSG
ST      ares
```

## CAT



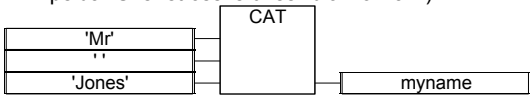
Megjegyzés : Ehhez az operátorhoz inputjainak a száma kettőnél többre kiterjeszthető.

Argumentumok:

(inputok)	MSG	(az összes üzenet hozzáadás hossza nem haladhatja meg az output üzenet kapacitást)
output:	MSG	az input üzenetek összefűzése

Leírás:  
Több üzenet egybefűzése

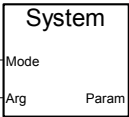
(\* FBD példa "Üzenet összefűzés" blokkokkal \*)



(\* ST Equivalencia \*) használja a + operátort \*)  
 myname := ('Mr' + ' ') + 'Jones';  
 (\* jelentése: myname := 'Mr Jones' \*)

(\* IL ekvivalencia \*)  
 LD               'Mr'  
 ADD             '  
 ADD             'Jones'  
 ST               myname

### SYSTEM



Argumentumok:

<b>Mode</b>	INT	a rendszer paramétert és az elérési módot azonosítja
<b>Arg</b>	INT-TMR	új érték egy "írás" hozzáféréshez
<b>Param</b>	INT	az elért paraméter értéke

Leírás:  
Hozzáférés a rendszer paramétereikhez

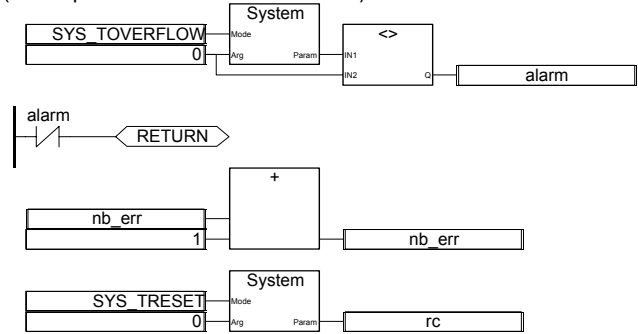
A következőkben a SYSTEM funkcióhoz rendelkezésre álló parancsok (előre definiált kulcsszavak) listája szerepel:

parancs	jelentése
SYS_TALLOWED	megengedett ciklusidőzítés olvasása
SYS_TCURRENT	pillanatnyi ciklusidőzítés olvasása
SYS_TMAXIMUM	maximális ciklusidőzítés olvasása
SYS_TOVERFLOW	ciklusidőzítés túlszordulások olvasása
SYS_TRESET	időzítő számlálók visszaállítása
SYS_TWRITE	ciklusidőzítés változtatása
SYS_ERR_TEST	futtatási hibák ellenőrzése
SYS_ERR_READ	legrégábbi futtatási hiba olvasása

Ezek a SYSTEM funkció előre definiált funkcióinak várt argumentumai:

parancs	argumentum	visszatérési érték
SYS_TALLOWED	0	megengedett ciklusidőzítés
SYS_TCURRENT	0	pillanatnyi ciklusidőzítés
SYS_TMAXIMUM	0	maximális érzékelt időzítés
SYS_TOVERFLOW	0	időzítés túlcsordulások száma
SYS_TRESET	0	0
SYS_TWRITE	új megengedett ciklusidőzítés	írt idő
SYS_ERR_TEST	0	0 ha nincs hiba érzékelve
SYS_ERR_READ	0	legrégebbi hibakód

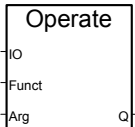
(\* FBD példa "Rendszer" blokkokkal \*)



(\* ST Equivalencia \*)

```
alarm := (SYSTEM (SYS_TOVERFLOW, 0) <> 0);
If (alarm) Then
    nb_err := nb_err + 1;
    rc := SYSTEM (SYS_TRESET, 0);
End_If;
```

**OPERATE**



Argumentumok:

<b>IO</b>	ANY	input vagy output változó
<b>Funct</b>	INT	kezelendő művelet
<b>Arg</b>	INT	argumentum egy I/O művelethez
<b>Q</b>	INT	visszatérés ellenőrzés

Leírás:

Hozzáférés egy I/O csatornához

Az OPERATE argumentumok jelentése az I/O interfész kivitelezésétől függ. Az OPERATE lehetőségekkel kapcsolatos bővebb információ a hardver kézikönyvben vagy a megfelelő I/O kártya műszaki megjegyzésében található.

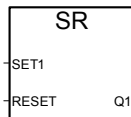
### B.9.2. Standard funkcióblokkok

Ezek az ISaGRAF rendszer által támogatott standard funkcióblokkok. Ezek a funkcióblokkok előre definiálva vannak, és azokat nem kell deklarálni a könyvtárban.

Logikaiak.....	SR	Beállítja a domináns bistabil
	RS	Domináns bistabil visszaállítása
	R_Trig	Felfutó él detektálás
	F_TRIG	Lefutó él detektálás
	SEMA	Szemafor
Számlálás.....	CTU	Fel számláló
	CTD	Le számláló
	CTUD	Fel-le számláló
Időzítők.....	TON	Késleltetés időzítés
	TOF	Késleltetésen kívüli időzítés
	TP	Impulzus időzítés
Integer analógok.....	CMP	Teljes összehasonlítás funkcióblokk
	StackInt	Integer analógok tömbje
Valós analógok:.....	AVERAGE	Futó átlag N mintára
	HYSTER	Logikai hiszterézis valósok különbségén
	LIM_ALRM	Felső/alsó határérték alarm hiszterézissel
	INTEGRAL	Integráció időre
	DERIVATE	Differenciálás időre
Jelgenerálás.....	BLINK	Villogó logikai jel
	SIG_GEN	Jelgenerátor

Megjegyzés: Amikor új "C" funkcióblokkok vannak létrehozva, azok az FBD nyelvből meghívhatóak.

#### SR



Argumentumok:

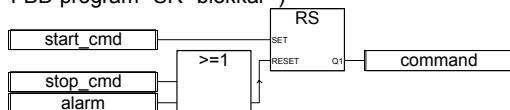
<b>SET1</b>	BOO	ha TRUE, Q1-et TRUE-ra állítja (domináns)
<b>RESET</b>	BOO	ha TRUE, Q1-et visszaállítja FALSE-ra
<b>Q1</b>	BOO	logikai memória állapot

Leírás:

Beállítja a domináns bistabilt: Lásd az alábbi Igaz táblázatot:

Set1	Reset	Q1	result Q1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(\* FBD program "SR" blokkal \*)



(\* ST Equivalencia \*) Feltételezzük, hogy az SR1 az SR blokk egy előfordulása \*)

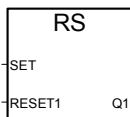
SR1((auto\_mode & start\_cmd), stop\_cmd);

command := SR1.Q1;

(\* IL Equivalencia: \*)

```
LD      auto_mode
AND     start_cmd
ST      SR1.set1
LD      stop_cmd
ST      SR1.reset
CAL     SR1
LD      SR1.Q1
ST      command
```

## RS



Argumentumok:

<b>SET</b>	BOO	ha TRUE, a Q1-t TRUE-ra állítja
<b>RESET1</b>	BOO	ha TRUE, visszaállítja a Q1-et FALSE-ra (domináns)
<b>Q1</b>	BOO	logikai memória állapot

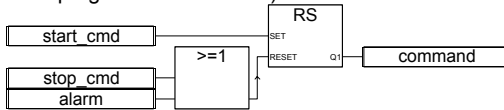
Leírás:

Domináns bistabil visszaállítása: Lásd az alábbi Igaz táblázatot:

Set	Reset1	Q1	result Q1
0	0	0	0
0	0	1	1
0	1	0	0

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

(\* FBD program "RS" blokkal \*)



(\* ST Equivalencia \*) Feltételezzük, hogy az RS1 az RS blokk egyik előfordulása \*)

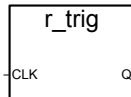
RS1(start\_cmd, (stop\_cmd OR alarm));

command := RS1.Q1;

(\* IL Equivalencia: \*)

```
LD      start_cmd
ST      RS1.set
LD      stop_cmd
OR      alarm
ST      RS1.reset1
CAL     RS1
LD      RS1.Q1
ST      parancs
```

## R\_TRIG



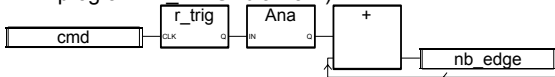
Argumentumok:

<b>CLK</b>	BOO	bármely logikai változó
<b>Q</b>	BOO	TRUE amikor a CLK FALSE-ről TRUE-ra emelkedik Minden más esetben FALSE

Leírás:

Egy logikai változó felfutó élét érzékeli

(\* FBD program "R\_TRIG" blokkal \*)



(\* ST Equivalencia \*) Feltételezzük, hogy az R\_TRIG1 az R\_TRIG blokk egyik előfordulása \*)

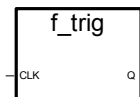
R\_TRIG1(cmd);

nb\_edge := ANA(R\_TRIG1.Q) + nb\_edge;

(\* IL Equivalencia: \*)

```
LD      cmd
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ANA
ADD     nb_edge
ST      nb_edge
```

## F\_TRIG



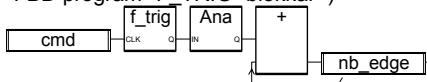
Argumentumok:

<b>CLK</b>	BOO	bármely logikai változó
<b>Q</b>	BOO	TRUE ha CLK TRUE-ból FALSE-ra változik Minden más esetben FALSE

Leírás:

Egy logikai változó lefutó élét érzékeli

(\* FBD program "F\_TRIG" blokkal \*)



(\* ST Equivalencia \*) Feltételezzük, hogy az F\_TRIG1 az F\_TRIG blokk egyik előfordulása \*)

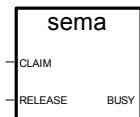
F\_TRIG1(cmd);

nb\_edge := ANA(F\_TRIG1.Q) + nb\_edge;

(\* IL Equivalencia: \*)

```
LD      cmd
ST      F_TRIG1.clk
CAL     F_TRIG1
LD      F_TRIG1.Q
ANA
ADD     nb_edge
ST      nb_edge
```

## SEMA



Argumentumok:

<b>CLAIM</b>	LOGIKAI	„teszt és beállítás” parancs
<b>RELEASE</b>	LOGIKAI	elengedi a szemafort
<b>BUSY</b>	LOGIKAI	a szemafor állapota

Leírás:

(\* "x" FALSE-ra inicializált logikai változó \*)

busy := x;

If claim Then

    x := True;

Else

    If release Then

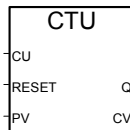
        busy := False;

        x := False;

    End\_if;

End\_if;

## CTU



Argumentumok:

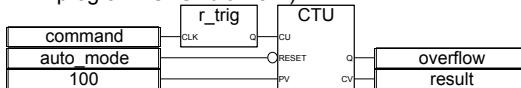
<b>CU</b>	BOO	számlálás input (akkor számol, amikor CU TRUE)
<b>RESET</b>	BOO	visszaállítás parancs (domináns)
<b>PV</b>	INT	programozott maximum érték
<b>Q</b>	BOO	túlsordulás TRUE ha CV = PV
<b>CV</b>	INT	counter result

**Figyelmeztetés:** A CTU blokk nem érzékeli a számlálás input (CU) felfutó vagy lefutó éleit. Azt egy "R\_TRIG" vagy "F\_TRIG" blokkal kell asszociálni, egy impulzusszámláló létrehozásához.

Leírás:

Számlál (integer) 0 -tól egy adott 1-1 értékig

(\* FBD program "CTU" blokkal \*)



(\* ST Equivalencia \*) Feltételezzük, hogy az R\_TRIG1 az R\_TRIG blokk egyik előfordulása, és a CTU1 a CTU blokk egyik előfordulása\*)

CTU1(R\_TRIG1(command),NOT(auto\_mode),100);

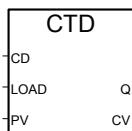
overflow := CTU1.Q;

result := CTU1.CV;

(\* IL Equivalencia: \*)

```
LD      command
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ST      CTU1.cu
LDN     auto_mode
ST      CTU1.reset
LD      100
ST      CTU1.pv
CAL     CTU1
LD      CTU1.Q
ST      overflow
LD      CTU1.cv
ST      result
```

## CTD



Argumentumok:

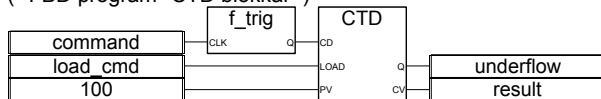
<b>CD</b>	BOO	számlálás input (lefelé számlálás ha CD TRUE)
<b>LOAD</b>	BOO	betöltés parancs (domináns) (CV = PV ha LOAD TRUE)
<b>PV</b>	INT	programozott kiindulási érték
<b>Q</b>	BOO	alulcsordulás: TRUE ha CV = 0
<b>CV</b>	INT	counter result

**Figyelmeztetés:** A CTD blokk nem érzékeli a számlálás input (CD) felfutó vagy lefutó élét. Azt egy "R\_TRIG" vagy "F\_TRIG" blokkal kell asszociálni, egy impulzusszámláló létrehozásához.

Leírás:

Számlálás (integer) egy adott értékről lefelé 0 1-1-re

(\* FBD program "CTD blokkal")



(\* ST Equivalencia \*) Feltételezzük, hogy az F\_TRIG1 az F\_TRIG blokk egyik előfordulása, a CTD1 pedig a CTD blokk egyik előfordulása\*)

CTD1(F\_TRIG1(command),load\_cmd,100);

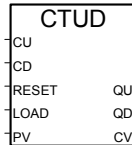
underflow := CTD1.Q;

result := CTD1.CV;

(\* IL Equivalencia: \*)

```
LD      command
ST      F_TRIG1.clk
CAL     F_TRIG1
LD      F_TRIG1.Q
ST      CTD1.cd
LD      load_cmd
ST      CTD1.load
LD      100
ST      CTD1.pv
CAL     CTD1
LD      CTD1.Q
ST      underflow
LD      CTD1.cv
ST      result
```

## CTUD



Argumentumok:

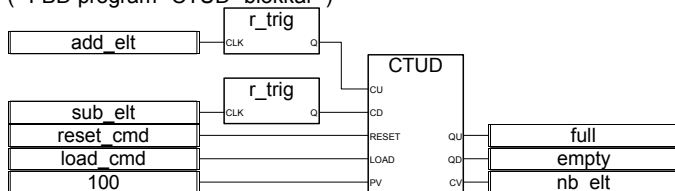
<b>CU</b>	BOO	felfelé számlálás (ha CU TRUE)
<b>CD</b>	BOO	lefelé számlálás (ha CD TRUE)
<b>RESET</b>	BOO	visszaállítás parancs (domináns) (CV = 0 ha RESET TRUE)
<b>LOAD</b>	BOO	töltés parancs (CV = PV ha LOAD TRUE)
<b>PV</b>	INT	programozott maximum érték
<b>QU</b>	BOO	túlcsordulás TRUE ha CV = PV
<b>QD</b>	BOO	alulcsordulás: TRUE ha CV = 0
<b>CV</b>	INT	counter result

**Figyelmeztetés:** A CTUD 1.blokk nem érzékeli a számlálás inputok (CU és CD) felfutó vagy lefutó éleit. Azt egy "R\_TRIG" vagy "F\_TRIG" blokkal kell asszociálni, egy impulzusszámláló létrehozásához.

Leírás:

Számlál (integert) 0 -tól egy adott 1-1 értékig  
vagy egy adott értéktől lefelé 0 1-1-ig

(\* FBD program "CTUD" blokkal \*)



(\* ST Equivalencia \*) Feltételezzük, hogy az R\_TRIG1 és R\_TRIG2 az R\_TRIG blokk két előfordulása, és a CTUD1 a CTUD blokk egy előfordulása\*)

CTUD1(R\_TRIG1(add\_elt), R\_TRIG2(sub\_elt), reset\_cmd, load\_cmd,100);

full := CTUD1.QU;

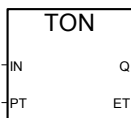
empty := CTUD1.QD;

nb\_elt := CTUD1.CV;

(\* IL Equivalencia: \*)

```
LD      add_elt
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ST      CTUD1.cu
LD      sub_elt
ST      R_TRIG2.clk
CAL     R_TRIG2
LD      R_TRIG2.Q
ST      CTUD1.cd
LD      reset_cmd
ST      CTUD1.reset
LD      load_cmd
ST      CTUD1.load
LD      100
ST      CTUD1.pv
CAL     CTUD1
LD      CTUD1.QU
ST      full
LD      CTUD1.QD
ST      empty
LD      CTUD1.CV
ST      nb_elt
```

## TON



Argumentumok:

IN

BOO

Ha felfutó élű, akkor növekvő belső időzítőt indít el

**PT**  
**Q**  
**ET**

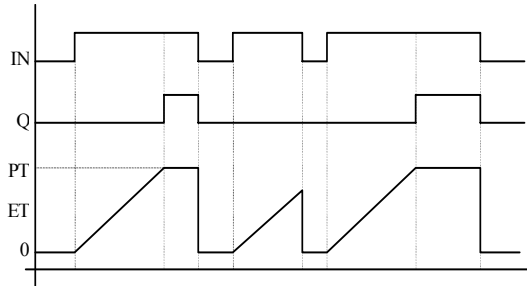
**TMR**  
**BOO**  
**TMR**

Ha lefutó élű, akkor megállítja és visszaállítja a belső időzítőt  
maximális programozott idő  
Ha TRUE, akkor a programozott idő letelt pillanatnyi eltelt idő

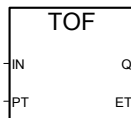
Leírás:

Egy belső időzítőt egy adott értékig növel.

Időzítő diagram:



## TOF



Argumentumok:

**IN**

**BOO**

Ha lefutó élű, akkor növekvő belső időzítőt indít el  
Ha felfutó élű, akkor megállítja és visszaállítja a belső időzítőt

**PT**

**TMR**

maximális programozott idő

**Q**

**BOO**

Ha TRUE: összes idő nem telt le

**ET**

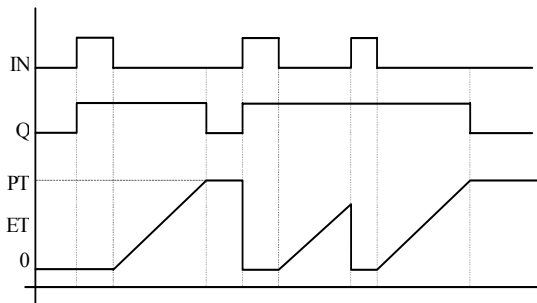
**TMR**

pillanatnyi eltelt idő

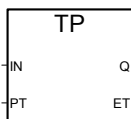
Leírás:

Egy belső időzítőt egy adott értékig növel.

Időzítő diagram:



**TP**



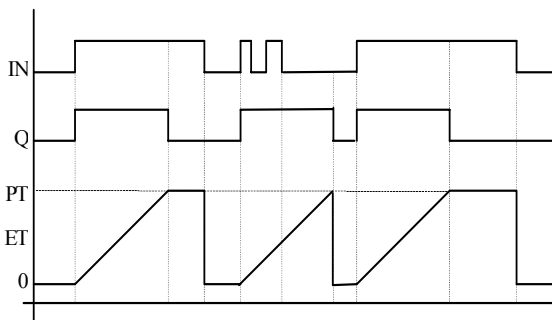
Argumentumok:

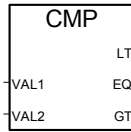
<b>IN</b>	BOO	Ha felfutó élű, akkor elindít egy növekvő belső időzítőt (ha az még nem növekszik) Ha FALSE, és csak amennyiben az időzítő letelt, akkor visszaállítja a belső időzítőt A számlálás közben az IN bármilyen változása hatástalan.
<b>PT</b>	TMR	maximális programozott idő
<b>Q</b>	BOO	Ha TRUE: az időzítő számlál
<b>ET</b>	TMR	pillanatnyi eltelt idő

Leírás:

Egy belső időzítőt egy adott értékig növel.

Időzítő diagram:



**CMP**

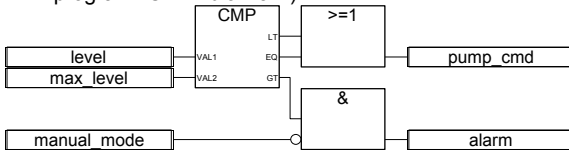
Argumentumok:

<b>VAL1</b>	INT	bármely előjeles integer analóg érték
<b>VAL2</b>	INT	bármely előjeles integer analóg érték
<b>LT</b>	BOO	TRUE ha val1 Less Than val2
<b>EQ</b>	BOO	TRUE ha val1 Equal to val2
<b>GT</b>	BOO	TRUE ha val1 Grater than val2

Leírás:

Összehasonlít két értéket: megmondja, ha azok egyenlőek, vagy ha az első kisebb vagy nagyobb, mint a második.

(\* FBD program "CMP" blokkal \*)



(\* ST Equivalencia \*) Feltételezzük, hogy az CMP1 a CMP blokk egy előfordulása \*)

CMP1(level, max\_level);

pump\_cmd := CMP1.LT OR CMP1.EQ;

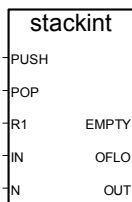
alarm := CMP1.GT AND NOT(manual\_mode);

(\* IL Equivalencia: \*)

```

LD      level
ST      CMP1.val1
LD      max_level
ST      CMP1.val2
CAL     CMP1
LD      CMP1.LT
OR      CMP1.EQ
ST      pump_cmd
LD      CMP1.GT
ANDN    manual_mode
ST      alarm
  
```

## STACKINT



Argumentumok:

<b>PUSH</b>	BOO	tolás parancs (csak felfutó élen) az IN értéket a halmaz tetejére adja
<b>POP</b>	BOO	felugrás parancs (csak felfutó élen) a halmazban törli az utolsó betöltött értéket (a halmaz tetejét)
<b>R1</b>	BOO	a halmazt visszaállítja üres állapotába
<b>IN</b>	INT	töltött érték
<b>N</b>	INT	alkalmazás által definiált halmaz mérete
<b>EMPTY</b>	BOO	TRUE ha a halmaz üres
<b>OFLO</b>	BOO	túlszámlálás TRUE ha a halmaz megtelt
<b>OUT</b>	INT	érték a halmaz tetején

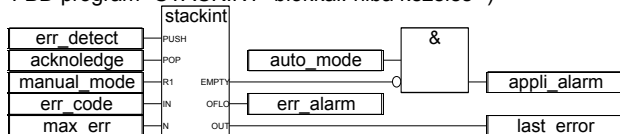
Leírás:

Integer értékek halmazát kezeli.

A "STACKINT" funkcióblokk egy felfutó él detektálást tartalmaz mind a PUSH, mind a POP parancsokhoz. A halmaz maximális mérete **128**. Az alkalmazás által definiált halmaz mérete **N** nem lehet kevesebb mint 1, vagy nagyobb mint 128.

Megjegyzendő, hogy az OFLO érték csak egy visszaállítás után érvényes (R1 legalább egyszer TRUE-ra és vissza FALSE-ra lett állítva).

(\* FBD program "STACKINT" blokkal: hiba kezelés \*)



(\* ST Equivalencia \*) Feltételezzük, hogy az STACKINT1 a STACKINT blokk egy előfordulása\*)

```
STACKINT1(err_detect, acknowledge, manual_mode, err_code, max_err);
appli_alarm := auto_mode AND NOT(STACKINT1.EMPTY);
err_alarm := STACKINT1.OFLO;
last_error := STACKINT1.OUT;
```

(\* IL Equivalencia: \*)

```
LD      err_detect
ST      STACKINT1.push
```

```
LD      acknowledge
ST      STACKINT1.pop
LD      manual_mode
ST      STACKINT1.r1
LD      err_code
ST      STACKINT1.IN
LD      max_err
ST      STACKINT1.N
CAL     STACKINT1
LD      auto_mode
ANDN    STACKINT1.empty
ST      appli_alarm
LD      STACKINT1.OFLO
ST      err_alarm
LD      STACKINT1.OUT
ST      last_error
```

**AVERAGE**



Argumentumok:

<b>RUN</b>	BOO	TRUE=run / FALSE=reset
<b>XIN</b>	REAL	bármely valós analóg érték
<b>N</b>	INT	alkalmazás által definiált minta szám
<b>XOUT</b>	REAL	érték futó átlaga

Leírás:

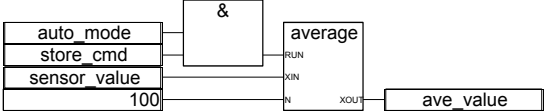
Egy értéket tárol mindegyik ciklusnál, és kiszámítja az összes, már tárolt érték átlagos értékét. Csak az utolsó N érték kerül eltárolásra.

A minták száma **N** nem haladhatja meg az **128** -at.

Ha a „**RUN**” parancs **FALSE** (visszaállítás mód), akkor az output érték megegyezik az input értékkel.

Amikor a tárolt értékek maximális N-je el lett érve, az utolsó tárolt érték kitörli az elsőt.

(\* FBD program "AVERAGE" blokkal: \*)



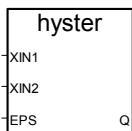
(\* ST Equivalencia \*) AVERAGE1 az AVERAGE blokk előfordulása \*)

```
AVERAGE1((auto_mode & store_cmd), sensor_value, 100);
ave_value := AVERAGE1.XOUT;
```

(\* IL Equivalencia: \*)

```
LD      auto_mode
AND     store_cmd
ST      AVERAGE1.run
LD      sensor_value
ST      AVERAGE1.xin
LD      100
ST      AVERAGE1.N
CAL     AVERAGE1
LD      AVERAGE1.XOUT
ST      ave_value
```

## HYSTER



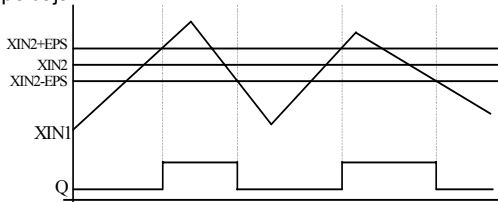
Argumentumok:

<b>XIN1</b>	REAL	bármely valós analóg érték
<b>XIN2</b>	REAL	tesztelni, ha XIN1 átlépte az XIN2+EPS-t
<b>EPS</b>	REAL	histerézis érték (zérótól nagyobbabnak kell lennie)
<b>Q</b>	BOO	TRUE ha XIN1 átlépte az XIN2+EPS-t, és még nincs az XIN2-EPS alatt

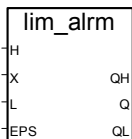
Leírás:

Histerézis egy felső határérték valós értékén.

Egy időzítő diagram példája:



## LIM\_ALARM



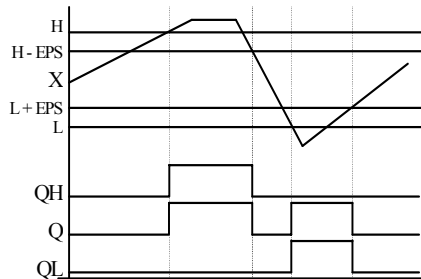
Argumentumok:

<b>H</b>	REAL	felső határérték
<b>X</b>	REAL	input: bármely valós analóg érték
<b>L</b>	REAL	alsó határérték
<b>EPS</b>	REAL	hiszterézis érték (zérótól nagyobbak kell lennie)
<b>QH</b>	BOO	„felső” alarm: TRUE ha X a H felső határérték felett van
<b>Q</b>	BOO	alarm output: TRUE ha X a határértékeken kívül van
<b>QL</b>	BOO	„alsó” alarm: TRUE ha X az L alsó határérték alatt van

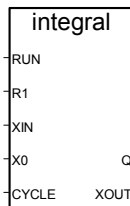
Leírás:

Hiszterézis egy valós értéken, felső és alsó határértékekhez.

Egy hiszterézis csak felső és alsó határértékekre van alkalmazva. A felső vagy alsó határértékekhez használt hiszterézis delta az EPS paraméter fele. Az alábbiakban egy időzítő diagram példája látható:



## INTEGRAL



Argumentumok:

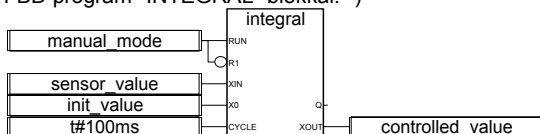
<b>RUN</b>	BOO	mód: TRUE=integrálás / FALSE=visszatartás
<b>R1</b>	BOO	visszaállítás felülírás
<b>XIN</b>	REAL	input: bármely valós analóg érték
<b>X0</b>	REAL	kezdeti érték
<b>CYCLE</b>	TMR	mintázási időtartam
<b>Q</b>	BOO	Not R1
<b>XOUT</b>	REAL	integrált output

Leírás:

Egy valós érték integrálása.

Ha a **"CYCLE"** paraméter értéke keisebb mint az ISaGRAF alkalmazás ciklusidőzítése, akkor a mintázási periódus az alkalmazás ciklusidőzítése.

(\* FBD program "INTEGRAL" blokkal: \*)



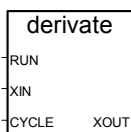
(\* ST Equivalencia \*) INTEGRAL1 az INTEGRAL blokk előfordulása \*)

INTEGRAL1(manual\_mode, NOT(manual\_mode), sensor\_value, init\_value, t#100ms);  
controlled\_value := INTEGRAL1.XOUT;

(\* IL Equivalencia: \*)

```
LD      manual_mode
ST      INTEGRAL1.run
STN     INTEGRAL1.R1
LD      sensor_value
ST      INTEGRAL1.XIN
LD      init_value
ST      INTEGRAL1.X0
LD      t#100ms
ST      INTEGRAL1.CYCLE
CAL     INTEGRAL1
LD      INTEGRAL1.XOUT
ST      controlled_value
```

## DERIVATE



Argumentumok:

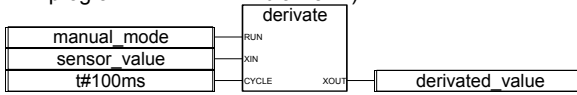
<b>RUN</b>	BOO	mód: TRUE=normál / FALSE=visszaállítás
<b>XIN</b>	REAL	input: bármely valós analóg érték
<b>CYCLE</b>	TMR	mintázási időtartam
<b>XOUT</b>	REAL	differenciált output

Leírás:

Egy valós érték differenciálása.

Ha a **"CYCLE"** paraméter értéke kisebb mint az ISaGRAF alkalmazás ciklusidőzítése, akkor a mintázási periódus az alkalmazás ciklusidőzítése.

(\* FBD program "DERIVATE" blokkal: \*)



(\* ST Equivalencia \*) A DERIVATE1 a DERIVATE blokk előfordulása \*)

DERIVATE1(manual\_mode, sensor\_value, t#100ms);

derivated\_value := DERIVATE1.XOUT;

(\* IL Equivalencia: \*)

```
LD      manual_mode
ST      DERIVATE1.run
LD      sensor_value
ST      DERIVATE1.XIN
LD      t#100ms
ST      DERIVATE1.CYCLE
CAL     DERIVATE1
LD      DERIVATE1.XOUT
ST      derived_value
```

## BLINK



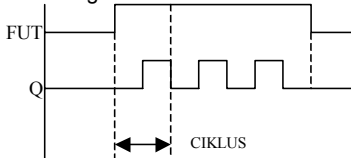
Argumentumok:

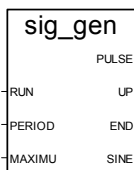
<b>RUN</b>	BOO	mód: TRUE=villogás / FALSE=az output hamisra történő visszaállítása
<b>CYCLE</b>	TMR	villogási időtartam
<b>Q</b>	BOO	villogó jel kiadása

Leírás:

Egy villogó jelet generál.

Időzítő diagram:



**SIG\_GEN**

Argumentumok:

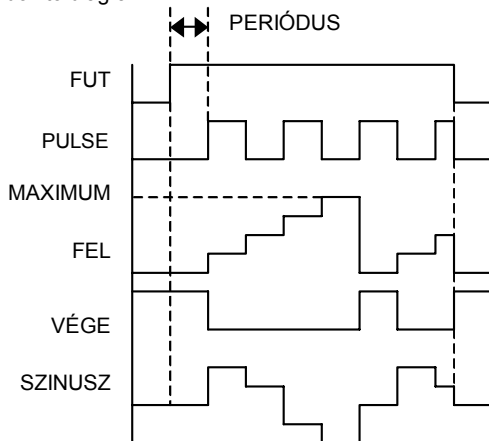
<b>RUN</b>	BOO	mód: TRUE=futás / FALSE=visszaállítás hamisra
<b>PERIOD</b>	TMR	mintázási időtartam
<b>MAXIMUM</b>	INT	maximális számlálási érték
<b>PULSE</b>	BOO	minden minta után invertálva
<b>UP</b>	INT	felfelé számláló, minden minta után növekszik
<b>END</b>	BOO	TRUE amikor a felfelé számlálás véget ér
<b>SINE</b>	REAL	szinuszjel (periódus = számlálás időtartam)

Leírás:

Különféle jeleket generál: egy logikain, integer felfelé számlálón, és valódi szinuszhullámon villog.

Amikor a számlálás eléri a maximális értéket, akkor újraindul 0-ról (zéróról). Tehát END csak az 1 PERIOD alatt tartja meg a TRUE értéket.

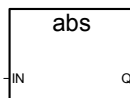
Időzítő diagram:



### B.9.3. Standard funkciók

Ezek az ISaGRAF rendszer által támogatott standard funkciók. Ezek a funkciók előre definiáltak, és azokat nem kell a könyvtárban deklarálni.

Math .....	ABS	Abszolút érték
	EXPT, POW	Kitevő, Hatványszámítás
	LOG	Logaritmus
	SQRT	Négyzetgyök
	TRUNC	A decimális rész levágása
Trigonometrikus.....	ACOS, ASIN,	Arkusz koszinusz, Arkusz szinusz,
	ATAN	Arkusz tangens
	COS, SIN,	Koszinusz, szinusz
	TAN	Tangens
Regiszter vezérlés .....	ROL, ROR	Forgatás balra, Forgatás jobbra
	SHL, SHR	Eltolás balra, Eltolás jobbra
Adat manipuláció.....	MIN, MAX,	Minimum, Maximum,
	LIMIT	Limit
	MOD	Modulo
	MUX4, MUX8	Multiplexer (4 vagy 8 tétellel),
	SEL	Bináris kiválasztó
	ODD	Páratlan paritás
	RAND	Véletlenszerű érték
Adat konverzió.....	ASCII	Karakter → ASCII kód
	CHAR	ASCII kód → Karakter
Karaktorsor kezelés.....	MLEN	Karaktorsor hosszának bekérése
	DELETE	AI-karaktorsor törlése
	INSERT	Karaktorsor beszúrása
	FIND,	AI-karaktorsor keresése
	REPLACE	AI-karaktorsor lecserélése
	LEFT, MID	Kivonás bal, közép
	RIGHT	vagy egy karaktersortól jobbra
	DAY_TIME	Időpont
Tömb manipuláció: .....	ARCREATE	Integer értékek tömbjének létrehozása
	ARREAD	Olvasás /
	ARWRITE	Tömb elem írása
Bináris fájl kezelés.....	F_OPEN	Egy bináris fájlt olvasási módban megnyit
	F_WOPEN	Egy bináris fájlt megnyitása írás módban
	F_CLOSE	Egy bináris fájl bezárása
	F_EOF	Egy bináris fájl végének tesztelése
	FA_READ	Egy analóg érték olvasása egy bináris fájlban
	FA_WRITE	Egy analóg érték írása egy bináris fájlba
	FM_READ	Egy üzenet karaktersor olvasása egy bináris fájlban
	FM_WRITE	Egy üzenet karaktersor írása egy bináris fájlba

**ABS**

Argumentumok:

**IN**

REAL

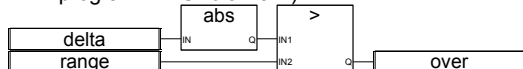
bármely előjeles valós analóg érték

**Q**                      **REAL**                      abszolút érték (mindig pozitív)

Leírás:

Egy valós érték abszolút (pozitív) értékét adja meg.

(\* FBD program "ABS" blokkal \*)



(\* ST Equivalencia \*)

over := (ABS (delta) > range);

(\* IL Equivalencia: \*)

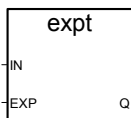
LD                      delta

ABS

GT                      interval

ST                      over

## EXPT



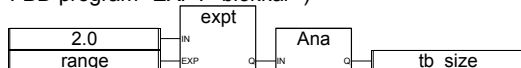
Argumentumok:

<b>IN</b>	<b>REAL</b>	bármely előjeles valós analóg érték
<b>EXP</b>	<b>INT</b>	integer analóg hatványkitevő
<b>Q</b>	<b>REAL</b>	(IN <sup>EXP</sup> )

Leírás:

A művelet valós eredményét adja: (alap<sup>hatványkitevő</sup>) „alap” az első argumentum, „hatványkitevő” pedig a második.

(\* FBD program "EXPT" blokkal \*)



(\* ST Equivalencia \*)

tb\_size := ANA (EXPT (2.0, range) );

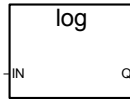
(\* IL Equivalencia: \*)

LD                      2.0

EXPT                      interval

ANA

ST                      tb\_size

**LOG**

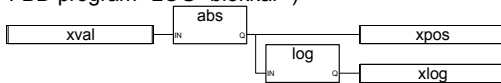
Argumentumok:

<b>IN</b>	REAL	zérótól nagyobbak kell lennie
<b>Q</b>	REAL	az input érték logaritmus (10-es alapú)

Leírás:

Kiszámítja egy valós érték (10-es alapú) logaritmusát.

(\* FBD program "LOG" blokkal \*)



(\* ST Equivalencia \*)

xpos := ABS (xval);

xlog := LOG (xpos);

(\* IL Equivalencia: \*)

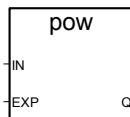
LD xval

ABS

ST xpos

LOG

ST xlog

**POW**

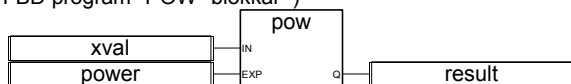
Argumentumok:

<b>IN</b>	REAL	hatványozandó valós analóg szám
<b>EXP</b>	REAL	hatvány (hatványkitevő)
<b>Q</b>	REAL	( $IN^{EXP}$ )
		1.0 ha IN nem 0.0 és EXP 0.0
		0.0 ha IN 0.0 és EXP negatív
		0.0 ha mind az IN, mind az EXP 0.0
		0.0 ha IN negatív, és Y nem egy integernek felel meg

Leírás:

A művelet valós eredményét adja: (alap<sup>hatványkitevő</sup>) „alap” az első argumentum, „hatványkitevő” pedig a második. A hatványkitevő egy valós érték.

(\* FBD program "POW" blokkal \*)



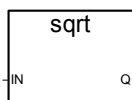
(\* ST Equivalencia \*)

result := POW (xval, power);

(\* IL Equivalencia: \*)

LD            xval  
 POW          power  
 ST            result

## SQRT



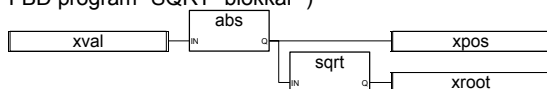
Argumentumok:

<b>IN</b>	REAL	nagyobbnak vagy egyenlőnek kell lennie nullától
<b>Q</b>	REAL	az input érték négyzetgyöke

Leírás:

Kiszámítja egy valós érték négyzetgyökét.

(\* FBD program "SQRT" blokkal \*)



(\* ST Equivalencia \*)

xpos := ABS (xval);  
 xroot := SQRT (xpos);

(\* IL Equivalencia: \*)

LD            xval  
 ABS  
 ST            xpos  
 SQRT  
 ST            xroot

**TRUNC**

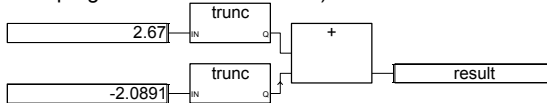
Argumentumok:

<b>IN</b>	REAL	bármely VALÓS analóg érték
<b>Q</b>	REAL	ha $IN > 0$ , akkor az inputtól kisebb vagy azzal egyenlő legmagasabb integer
		ha $IN < 0$ , akkor az inputtól nagyobb vagy azzal egyenlő legkisebb integer

Leírás:

Megcsonkít egy valós értéket úgy, hogy annak csak az integer része marad meg

(\* FBD program "TRUNC" blokkal \*)



(\* ST Equivalencia \*)

```
result := TRUNC (+2.67) + TRUNC (-2.0891);
```

```
(* jelentése: eredmény := 2.0 + (-2.0) := 0.0; *)
```

(\* IL Equivalencia: \*)

```
LD      2.67
```

```
TRUNC
```

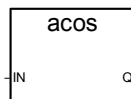
```
ST      temporary (* az első TRUNC ideiglenes eredménye*)
```

```
LD      -2.0891
```

```
TRUNC
```

```
ADD     temporary
```

```
ST      result
```

**ACOS**

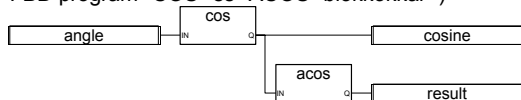
Argumentumok:

<b>IN</b>	REAL	a következő készletben kell lennie: [-1.0 .. +1.0]
<b>Q</b>	REAL	QREAL az input érték arkusz-koszinusza (a [0.0 ..PI]) = 0.0 érvénytelen inputhoz

Leírás:

Kiszámítja egy valós érték Arkusz koszinuszát.

(\* FBD program "COS" és "ACOS" blokkokkal \*)



(\* ST Equivalencia \*)

koszinusz := COS (szög);

eredmény := ACOS (cosine); (\* az eredmény megegyezik a szöggel\*)

(\* IL Equivalencia: \*)

LD angle

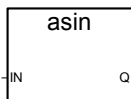
COS

ST cosine

ACOS

ST result

## ASIN



Argumentumok:

**IN**

REAL

**Q**

REAL

a következő készletben kell lennie: [-1.0 .. +1.0]

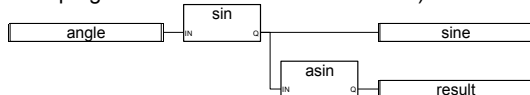
QREAL az input érték arkusz-szinusza (a [-PI/2 ..+PI/2] készletben van)

= 0.0 érvénytelen inputhoz

Leírás:

Kiszámítja egy valós érték Arkusz szinuszát.

(\* FBD program "SIN" és "ASIN" blokkokkal \*)



(\* ST Equivalencia \*)

szinusz := SIN (szög);

eredmény := ASIN (sine); (\* az eredmény megegyezik a szöggel\*)

(\* IL Equivalencia: \*)

LD angle

SIN

ST sine

ASIN

ST result

**ATAN**

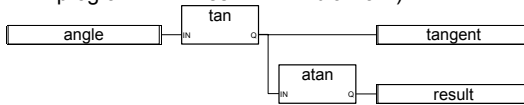
Argumentumok:

<b>IN</b>	REAL	bármely valós analóg érték
<b>Q</b>	REAL	az input érték arkusz-tangense (a $[-\pi/2 \dots +\pi/2]$ készletben van)
		= 0.0 érvénytelen inputhoz

Leírás:

Kiszámítja egy valós érték Arkusz tangensét.

(\* FBD program "TAN" és "ATAN" blokkal \*)



(\* ST Equivalencia \*)

tangens := TAN (szög);

eredmény := ATAN (tangent); (\* az eredmény megegyezik a szöggel\*)

(\* IL Equivalencia: \*)

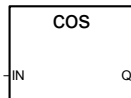
LD            angle

TAN

ST            tangent

ATAN

ST            result

**COS**

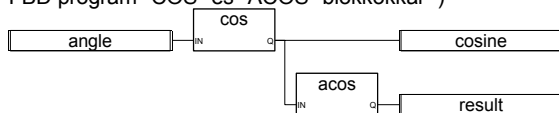
Argumentumok:

<b>IN</b>	REAL	bármely VALÓS analóg érték
<b>Q</b>	REAL	az input érték koszinusza (a $[-1.0 \dots +1.0]$ készletben van)

Leírás:

Kiszámítja egy valós érték Koszinuszát.

(\* FBD program "COS" és "ACOS" blokkokkal \*)



(\* ST Equivalencia \*)

koszinusz := COS (szög);

eredmény := ACOS (cosine); (\* az eredmény megegyezik a szöggel\*)

(\* IL Equivalencia: \*)

LD angle

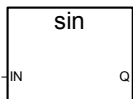
COS

ST cosine

ACOS

ST result

## SIN



Argumentumok:

**IN**

REAL

bármely VALÓS analóg érték

**Q**

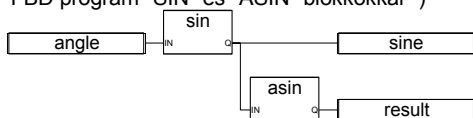
REAL

az input érték szinusa (a [-1.0 ..+1.0] készletben van)

Leírás:

Kiszámítja egy valós érték Szinusztát.

(\* FBD program "SIN" és "ASIN" blokkokkal \*)



(\* ST Equivalencia \*)

szinusz := SIN (szög);

eredmény := ASIN (sine); (\* az eredmény megegyezik a szöggel\*)

(\* IL Equivalencia: \*)

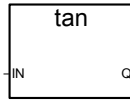
LD angle

SIN

ST sine

ASIN

ST result

**TAN**

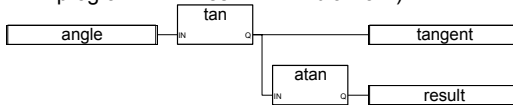
Argumentumok:

<b>IN</b>	REAL	nem lehet egyenlő $\pi/2$ modulo $\pi$ -vel
<b>Q</b>	REAL	az input érték tangense
		= $1E+38$ érvénytelen inputhoz

Leírás:

Kiszámítja egy valós érték tangensét.

(\* FBD program "TAN" és "ATAN" blokkal \*)



(\* ST Equivalencia \*)

tangens := TAN (szög);

eredmény := ATAN (tangent); (\* az eredmény megegyezik a szöggel\*)

(\* IL Equivalencia: \*)

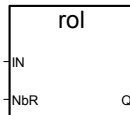
LD            angle

TAN

ST            tangent

ATAN

ST            result

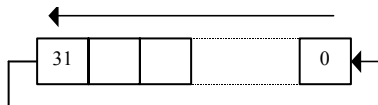
**ROL**

Argumentumok:

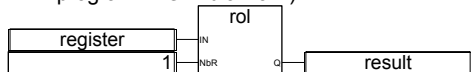
<b>IN</b>	INT	bármely integer analóg érték
<b>NbR</b>	INT	1 bit elforgatások száma (az [1..31] készletben)
<b>Q</b>	INT	balra forgatott érték
		nincs hatása, ha $NbR \leq 0$

Leírás:

Egy integer bitjeit balra forgatja. A forgatás 32 biten történik:



(\* FBD program "ROL" blokkal \*)



(\* ST Equivalencia \*)

eredmény := ROL (register, 1);

(\* regiszter = 2#0100\_1101\_0011\_0101\*)

(\* eredmény = 2#1001\_1010\_0110\_1010\*)

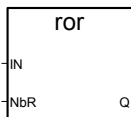
(\* IL Equivalencia: \*)

LD register

ROL 1

ST result

## ROR

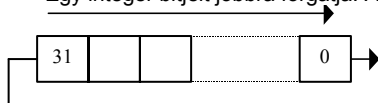


Argumentumok:

<b>IN</b>	INT	bármely integer analóg érték
<b>NbR</b>	INT	1 bit elforgatások száma (az [1..31] készletben)
<b>Q</b>	INT	jobbra forgatott érték
		nincs hatása, ha NbR <= 0

Leírás:

Egy integer bitjeit jobbra forgatja. A forgatás 32 biten történik:



(\* FBD program "ROR" blokkal \*)

-

(\* ST Equivalencia \*)

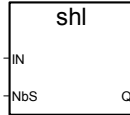
eredmény := ROR (register, 1);

(\* regiszter = 2#0100\_1101\_0011\_0101\*)

(\* eredmény = 2#1010\_0110\_1001\_1010 \*)

(\* IL Equivalencia: \*)

LD	register
ROR	1
ST	result

**SHL**

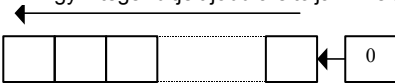
Argumentumok:

IN	INT	bármely integer analóg érték
NbS	INT	1 bit eltolások száma (az [1..31] készletben)
Q	INT	balra eltolt érték

nincs hatása, ha NbS ≤ 0  
0 a legalacsonyabb bit lecserélésére szolgál

Leírás:

Egy integer bitjeit jobbra eltolja. Az eltolás a 32 biten történik:



(\* FBD program "SHL" 1.blokkal \*)

-

(\* ST Equivalencia \*)

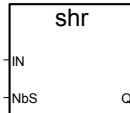
eredmény := SHL (register,1);

(\* regiszter = 2#0100\_1101\_0011\_0101\*)

(\* eredmény = 2#1001\_1010\_0110\_1010\*)

(\* IL Equivalencia: \*)

LD	register
SHL	1
ST	result

**SHR**

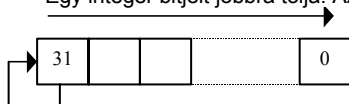
Argumentumok:

IN	INT	bármely integer analóg érték
NbS	INT	1 bit eltolások száma (az [1..31] készletben)
Q	INT	jobbra eltolt érték

nincs hatása, ha NbS <= 0  
a legmagasabb bit mindegyik eltolásnál másolásra kerül

Leírás:

Egy integer bitjeit jobbra tolja. Az eltolás a 32 biten történik:



(\* FBD program "SHR" blokkal \*)

(\* ST Equivalencia \*)

eredmény := SHR (register,1);

(\* regiszter = 2#1100\_1101\_0011\_0101 \*)

(\* eredmény = 2#1110\_0110\_1001\_1010 \*)

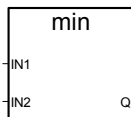
(\* IL Equivalencia: \*)

LD register

SHR 1

ST result

## MIN



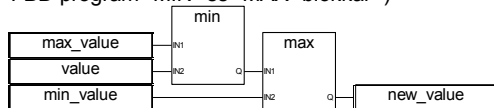
Argumentumok:

<b>IN1</b>	INT	bármely előjeles integer analóg érték
<b>IN2</b>	INT	(nem lehet VALÓS)
<b>Q</b>	INT	mindkét input érték minimuma

Leírás:

Két integer érték minimumát adja meg.

(\* FBD program "MIN" és "MAX" blokkal \*)



(\* ST Equivalencia \*)

new\_value := MAX (MIN (max\_value, value), min\_value);

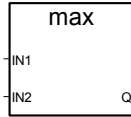
(\* az értéket a [min\_value..max\_value] készlethez köti \*)

(\* IL Equivalencia: \*)

LD max\_value

MIN value

MAX	min_value
ST	new_value

**MAX**

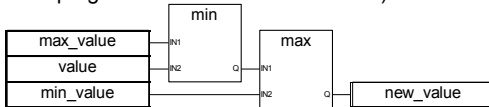
Argumentumok:

<b>IN1</b>	INT	bármely előjeles integer analóg érték
<b>IN2</b>	INT	(nem lehet VALÓS)
<b>Q</b>	INT	mindkét input érték maximuma

Leírás:

Két integer érték maximumát adja meg.

(\* FBD program "MIN" és "MAX" blokkal \*)



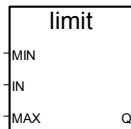
(\* ST Equivalencia: \*)

new\_value := MAX (MIN (max\_value, value), min\_value);

(\* az értéket a [min\_value..max\_value] készlethez köti \*)

(\* IL Equivalencia: \*)

LD	max_value
MIN	value
MAX	min_value
ST	new_value

**LIMIT**

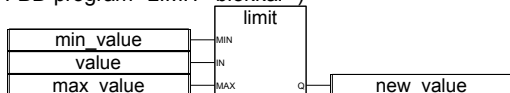
Argumentumok:

<b>MIN</b>	INT	minimum megengedett érték
<b>IN</b>	INT	bármely előjeles integer analóg érték
<b>MAX</b>	INT	maximum megengedett érték
<b>Q</b>	INT	input érték a megengedett tartományhoz kötve

# Leírás:

Egy integer értéket egy megadott intervallumban korlátoz. Vagy megtartja az értékét ha a minimum és maximum között van, vagy a maximumra változik ha fölötte van, vagy a minimumra változik, ha alatta van.

(\* FBD program "LIMIT" blokkal \*)



(\* ST Equivalencia \*)

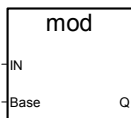
new\_value := LIMIT (min\_value, value, max\_value);

(\* az értéket a [min\_value..max\_value] készlethez köti \*)

(\* IL Equivalencia: \*)

```
LD      min_value
LIMIT   value, max_value
ST      new_value
```

## MOD



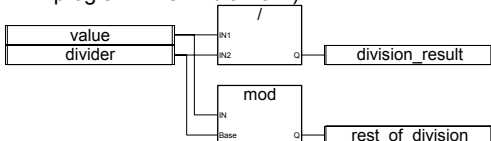
# Argumentumok:

<b>IN</b>	INT	bármely előjeles INTEGER analóg érték
<b>Alap</b>	INT	zérótól nagyobboknak kell lennie
<b>Q</b>	INT	modulo számítás (input MOD alap)
		-1-et ad vissza ha Base <= 0

# Leírás:

Kiszámítja egy integer érték maradékát.

(\* FBD program "MOD" blokkal \*)



(\* ST Equivalencia \*)

division\_result := (érték / osztó); (\* integer osztás \*)

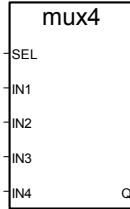
rest\_of\_division := MOD (érték, osztó); (\* az osztás többi része \*)

(\* IL Equivalencia: \*)

```
LD      value
DIV     divider
```

ST	division_result
LD	value
MOD	divider
ST	rest_of_division

## MUX4



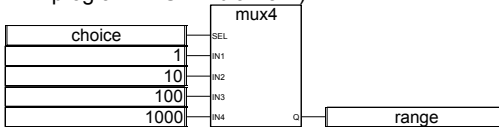
Argumentumok:

<b>SEL</b>	INT	kiválasztó integer érték (a [0..3] készletben kell lennie)
<b>IN1..IN4</b>	INT	bármely integer analóg érték
<b>Q</b>	INT	= value1 if SEL = 0 = value2 if SEL = 1 = value3 if SEL = 2 = value4 if SEL = 3 = 0 a kiválasztó minden más értékéhez

Leírás:

Multiplexer 4 tétellel: kiválaszt egy értéket 4 integer érték közül.

(\* FBD program "MUX4" blokkal \*)



(\* ST Equivalencia \*)

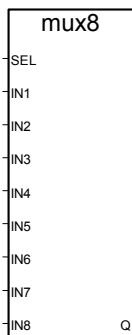
range := MUX4 (choice, 1, 10, 100, 1000);

(\* választ 4 előre definiált tartomány közül, ha pl. a választás 1, akkor a tartomány 10 lesz \*)

(\* IL Equivalencia: \*)

LD	választás
MUX4	1,10,100,1000
ST	intervallum

## MUX8



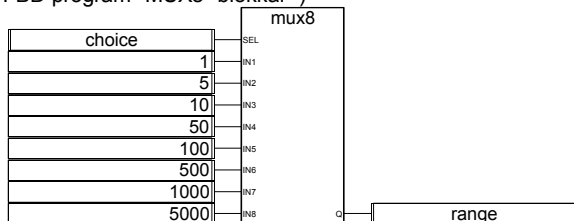
Argumentumok:

<b>SEL</b>	INT	kiválasztó integer érték (a [0.0,7] készletben kell lennie)
<b>IN1..IN8</b>	INT	bármely integer analóg érték
<b>Q</b>	INT	= value1 if selector = 0
		= value2 if selector = 1
		...
		= value8 if selector = 7
		= 0 a kiválasztó minden más értékéhez

Leírás:

Multiplexer 8 tétellel: kiválaszt egy értéket 8 integer érték közül.

(\* FBD program "MUX8" blokkal \*)



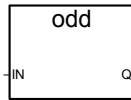
(\* ST Equivalencia \*)

range := MUX8 (choice, 1, 5, 10, 50, 100, 500, 1000, 5000);

(\* választ 8 előre definiált tartomány közül, ha pl. a választás 3, akkor a tartomány 50 lesz \*)

(\* IL Equivalencia: \*)

LD	választás
MUX8	1,5,10,50,100,500,1000,5000
ST	intervallum

**ODD**

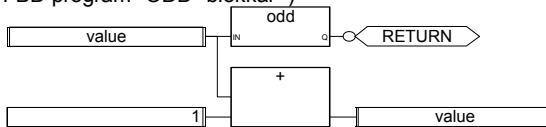
Argumentumok:

<b>IN</b>	INT	bármely előjeles integer analóg érték
<b>Q</b>	BOO	TRUE ha az input érték páratlan FALSE ha az input érték páros

Leírás:

Egy integer paritását vizsgálja: az eredmény páratlan vagy páros.

(\* FBD program "ODD" blokkal \*)



(\* ST Equivalencia \*)

If Not (ODD (value)) Then Return; End\_if;

value := value + 1;

(\* az értéket mindig párossá teszi \*)

(\* IL Equivalencia: \*)

LD value

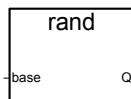
ODD

RETNC

LD value

ADD 1

ST value

**RAND**

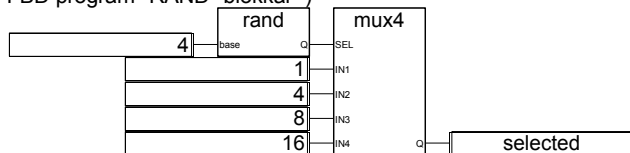
Argumentumok:

<b>base</b>	INT	meghatározza a megengedett számkészletet
<b>Q</b>	INT	véletlenszerű érték a [0..alap-1] készletben

Leírás:

Egy véletlenszerű integer értéket ad egy adott tartományban.

(\* FBD program "RAND" blokkal \*)



(\* ST Equivalencia \*)

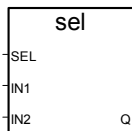
selected := MUX4 ( RAND (4), 1, 4, 8, 16 );

(\* 4 előre definiált érték egyikének véletlenszerű kiválasztása a RAND meghívás által kiadott érték a [0..3] készletben van, így a MUX4-tól 'kiválasztott' 'véletlenszerűen' az alábbi értéket fogja szerezni  
1 ha 0 van kiadva a RAND-tól,  
vagy 4 ha 1 van kiadva a RAND-tól,  
vagy 8 ha 2 van kiadva a RAND-tól,  
vagy 16 ha 3 van kiadva a RAND-tól,\*)

(\* IL Equivalencia: \*)

LD	4
RAND	
MUX4	1,4,8,16
ST	selected

## SEL



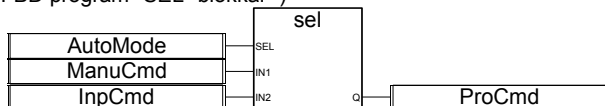
Argumentumok:

<b>SEL</b>	BOO	a kiválasztott értéket mutatja
<b>IN1, IN2</b>	INT	bármely integer analóg érték
<b>Q</b>	INT	= value1 if SEL is FALSE
		= value2 if SEL is TRUE

Leírás:

Bináris kiválasztó: kiválaszt egy értéket 2 integer érték közül.

(\* FBD program "SEL" blokkal \*)



(\* ST Equivalencia \*)

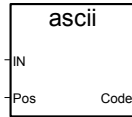
ProCmd := SEL (AutoMode, ManuCmd, InpCmd);

(\* Folyamat parancs kiválasztás \*)

(\* IL Equivalencia: \*)

LD	AutoMode
SEL	ManuCmd, InpCmd
ST	ProCmd

## ASCII



Argumentumok:

<b>IN</b>	MSG	bármilyen nem üres karaktorsor
<b>Pos</b>	INT	a kiválasztott karakter pozíciója
		a [1.. len] készletben (len az IN üzenet hossza)
<b>Code</b>	INT	a kiválasztott karakter kódja
		(a [0 .. 255])
		0-t ad vissza, ha Pos a karaktorsoron kívül van

Leírás:

Megadja egy üzenet karaktorsorban levő karakter ASCII kódját.

(\* FBD program "ASCII" blokkal \*)



(\* ST Equivalencia \*)

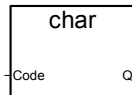
FirstChr := ASCII (üzenet, 1);

(\* FirstChr a karaktorsor első karakterének ASCII kódja \*)

(\* IL Equivalencia: \*)

LD	message
ASCII	1
ST	FirstChr

## CHAR



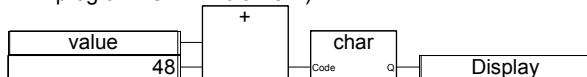
Argumentumok:

<b>Code</b>	INT	a [0..255] készletben levő kód
<b>Q</b>	MSG	egy karaktorsor
		a karakter ASCII kódja az input Kódban van megadva
		(ASCII kód modulo 256 használat)

Leírás:

Egy egy karakteres üzenet karaktersort ad egy adott ASCII kdból.

(\* FBD program "CHAR" blokkal \*)



(\* ST Equivalencia \*)

Display := CHAR ( value + 48 );

(\* az érték a [0..9] készletben van \*)

(\* 48 a '0' ASCII kódja \*)

(\* az eredmény egy '0' és '9' közötti egykarakteres karaktersor \*)

(\* IL Equivalencia: \*)

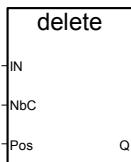
LD value

ADD 48

CHAR

ST Display

## DELETE



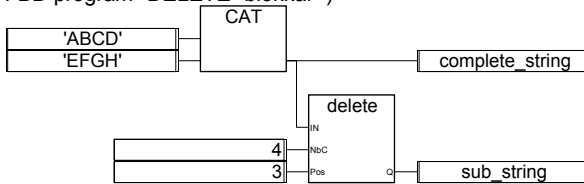
Argumentumok:

<b>IN</b>	MSG	bármilyen nem üres üzenet
<b>NbC</b>	INT	törendő karakterek száma
<b>Pos</b>	INT	az első törölt karakter helye (a karaktersor első karakterének a helye 1)
<b>Q</b>	MSG	módosított karaktersor üres karaktersor ha Pos < 1 kezdeti karaktersor ha Pos > IN karaktersor hossza kezdeti karaktersor ha NbC <= 0

Leírás:

Törli egy üzenet karaktersor egy részét.

(\* FBD program "DELETE" blokkal \*)



(\* ST Equivalencia \*)

complete\_string := 'ABCD' + 'EFGH'; (\* complete\_string is 'ABCDEFGH' \*)

sub\_string := DELETE (complete\_string, 4, 3); (\* sub\_string is 'ABGH' \*)

(\* IL Equivalencia: \*)

```
LD      'ABCD'
ADD     'EFGH'
ST      complete_string
DELETE  4,3
ST      sub_string
```

## FIND



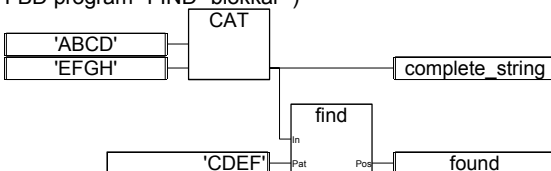
Argumentumok:

<b>In</b>	MSG	bármely üzenet karaktorsor
<b>Pat</b>	MSG	bármely nem üres karaktorsor (Minta)
<b>Pos</b>	INT	= 0 ha Pat al-karaktorsor nem található
		= a Pat al-karaktorsor első előfordulása első karakterének pozíciója
		(az első pozíció 1)
		ez a funkció <b>érzékeny a kisbetű-nagybetű beállításra</b>

Leírás:

Egy al-karaktorsort keres meg egy üzenet karaktersorban. Megadja az al-karaktorsor karaktersorának pozícióját.

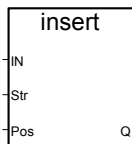
(\* FBD program "FIND" blokkal \*)



```
(* ST Equivalencia *)
complete_string := 'ABCD' + 'EFGH'; (* complete_string is 'ABCDEFGH' *)
found := FIND (complete_string, 'CDEF'); (* megtalálva 3 *)
```

```
(* IL Equivalencia: *)
LD      'ABCD'
ADD     'EFGH'
ST      complete_string
FIND    'CDEF'
ST      found
```

## INSERT



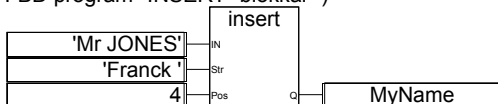
Argumentumok:

<b>IN</b>	MSG	kezdeti karaktorsor
<b>Str</b>	MSG	beillesztendő karaktorsor
<b>Pos</b>	INT	a beillesztés helye A beillesztés a beillesztési hely elé történik (az első érvényes hely 1)
<b>Q</b>	MSG	módosított karaktorsor üres karaktorsor ha Pos <= 0 mindkét karaktorsor összefűzése ha Pos nagyobb mint az IN karaktorsor hossza

Leírás:

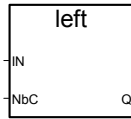
Egy al-karaktorsort illeszt be egy üzenet karaktersorba a megadott helyen.

(\* FBD program "INSERT" blokkal \*)



```
(* ST Equivalencia *)
MyName := INSERT ('Mr JONES', 'Frank ', 4);
(* MyName 'Mr Frank JONES' *)
```

```
(* IL Equivalencia: *)
LD      'Mr JONES'
INSERT  'Frank ',4
ST      MyName
```

**LEFT**

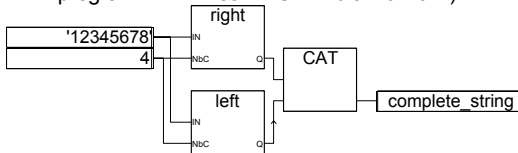
Argumentumok:

<b>IN</b>	MSG	bármilyen nem üres karaktersor
<b>NbC</b>	INT	kivonandó karakterek száma nem lehet nagyobb, mint az IN karaktersor hossza
<b>Q</b>	MSG	az IN karaktersor bal oldali fele (ennek hossza = NbC) üres karaktersor ha NbC <= 0 teljes IN karaktersor ha NbC >= IN karaktersor hossz

Leírás:

Kivonja egy üzenet karaktersor bal oldali felét. A kivonandó karakterek száma meg van adva.

(\* FBD program "LEFT" és "RIGHT" blokkokkal \*)



(\* ST Equivalencia \*)

```
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
```

(\* a complete\_string '56781234'

a RIGHT meghívásból kiadott érték '5678'

a LEFT hívásból kiadott érték '1234'

\*)

(\* IL Equivalencia: Először a LEFT-re való hívás történik meg \*)

LD '12345678'

LEFT 4

ST sub\_string (\* közbenső eredmény \*)

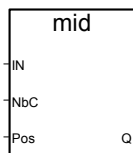
LD '12345678'

RIGHT 4

ADD sub\_string

ST complete\_string

## MID



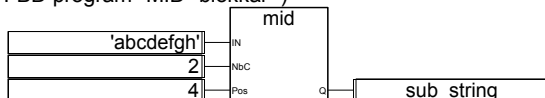
Argumentumok:

<b>IN</b>	MSG	bármilyen nem üres karaktorsor
<b>NbC</b>	INT	kivonandó karakterek száma nem lehet nagyobb, mint az IN karaktorsor hossza
<b>Pos</b>	INT	az al-karaktorsor pozíciója az al-karaktorsor első karaktere a Pos által mutatott lesz (az első érvényes hely 1)
<b>Q</b>	MSG	a karaktorsor középső része (annak hossza = NbC) üres karaktorsor ha a paraméterek nem érvényesek

Leírás:

Kivonja egy üzenet karaktorsor egy részét. A kivonandó karakterek száma és az első karakter pozíciója meg van adva.

(\* FBD program "MID" blokkal \*)



(\* ST Equivalencia \*)

sub\_string := MID ('abcdefgh', 2, 4);

(\* a sub\_string 'de' \*)

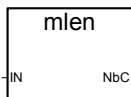
(\* IL Equivalencia: \*)

LD 'abcdefgh'

MID 2,4

ST sub\_string

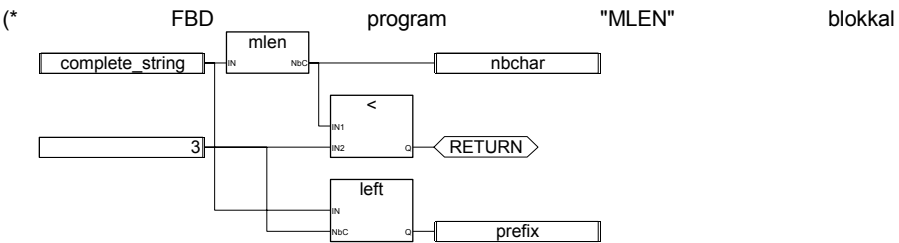
## MLEN



Argumentumok:

<b>IN</b>	MSG	bármely karaktorsor üzenet
<b>NbC</b>	INT	karakterek száma az IN karaktorsorban

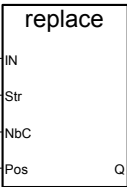
Leírás:  
Kiszámítja egy üzenet karaktersor hosszát.



(\* ST Equivalencia \*)  
nbchar := MLEN (complete\_string);  
If (nbchar < 3) Then Return; End\_if;  
prefix := LEFT (complete\_string, 3);  
(\* ez a program kiveszi a karaktersortól balra levő 3 karaktert, és az eredményt az előtag  
üzenet változóba teszi  
semmi sem történik, ha a karaktersor hossza 3 karaktertől kisebb \*)

(\* IL Equivalencia: \*)  
LD                      complete\_string  
MLEN  
ST                      nbchar  
LT                      3  
RETC  
LD                      complete\_string  
LEFT                    3  
ST                      prefix

REPLACE



Argumentumok:

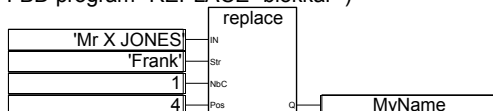
<b>IN</b>	MSG	bármely karaktersor
<b>Str</b>	MSG	beillesztendő karaktersor (az NbC karakterek lecserélésére)
<b>NbC</b>	INT	törölendő karakterek száma
<b>Pos</b>	INT	az első módosított karakter helye (az első érvényes hely 1)
<b>Q</b>	MSG	módosított karaktersor: - NbC karakterek a Pos pozíciónál vannak törölve

- aztán az Str al-karaktorsor beillesztésre kerül ebben a pozícióban  
 üres karaktorsort ad vissza ha  $Pos \leq 0$   
 karaktorsor összefűzést ad vissza (IN+Str) ha Pos nagyobb mint az IN karaktorsor hossza  
 az IN kiindulási karaktersort adja vissza ha  $NbC \leq 0$

Leírás:

Egy üzenet karaktersor egy részét egy új karaktersorral cseréli le.

(\* FBD program "REPLACE" blokkal \*)



(\* ST Equivalencia \*)

MyName := REPLACE ('Mr X JONES', 'Frank', 1, 4);

(\* MyName 'Mr Frank JONES' \*)

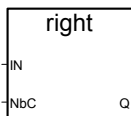
(\* IL Equivalencia: \*)

LD 'Mr X JONES'

REPLACE 'Frank',1,4

ST MyName

## RIGHT



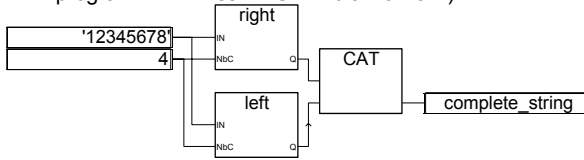
Argumentumok:

<b>IN</b>	MSG	bármilyen nem üres karaktersor
<b>NbC</b>	INT	nem lehet nagyobb, mint az IN karaktersor hossza
<b>Q</b>	MSG	a karaktersor jobb oldali része (hossza = NbC)
		üres karaktersor ha $NbC \leq 0$
		teljes karaktersor ha $NbC \geq$ karaktersor hossza

Leírás:

Kivonja egy üzenet karaktersor jobb oldali részét. A kivonandó karakterek száma meg van adva.

(\* FBD program "LEFT" és "RIGHT" blokkokkal \*)



(\* ST Equivalencia \*)

complete\_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);

(\* a complete\_string '56781234'

a RIGHT meghívásból kiadott érték '5678'

a LEFT hívásból kiadott érték '1234'

\*)

(\* IL Equivalencia: Először a LEFT-re való hívás történik meg \*)

LD '12345678'

LEFT 4

ST sub\_string (\* közbenső eredmény \*)

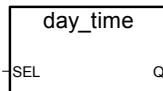
LD '12345678'

RIGHT 4

ADD sub\_string

ST complete\_string

## DAY\_TIME



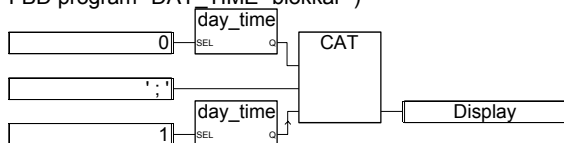
Argumentumok:

<b>SEL</b>	INT	output kiválasztás 0= pillanatnyi dátum beszerzése 1= pillanatnyi idő beszerzése 2= pillanatnyi nap beszerzése
<b>Q</b>	MSG	egy karaktersoron kifejezett idő/dátum 'YYYY/MM/DD' if SEL = 0 'HH:MM:SS' if SEL = 1 nap neve ha SEL = 2 (pl.: 'hétfő')

Leírás:

Egy üzenet karaktersorként megadja a dátumot, az időpontot, vagy a napot.

(\* FBD program "DAY\_TIME" blokkal \*)



(\* ST Equivalencia \*)

Display := Day\_Time (0) + ' ; ' + Day\_Time (1);

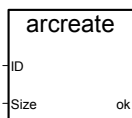
(\* A megjelenítési szöveg formátum: "YYYY/MM/DD ; HH:MM:SS" \*)

(\* IL Equivalencia: Először a day\_time(1) hívása történik \*)

```

LD      1
DAY_TIME
ST      hour_str    (* közbenső eredmény *)
LD      0
DAY_TIME
ADD     ' ; '
ADD     hour_str
ST      Megjelenítés
  
```

## ARCREATE



Argumentumok:

<b>ID</b>	INT
<b>Size</b>	INT
<b>ok</b>	INT

a tömb azonosítója (a [0..15] készletben kell lennie)

elemek száma a tömbben

végrehajtás státusza:

**1** = ha a tömb sikeresen létre lett hozva

**2** = érvénytelen tömb azonosító, vagy a tömb már létre lett hozva

**3** = érvénytelen méret

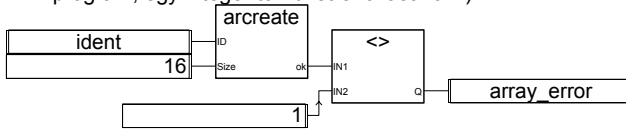
**4** = nincs elég memória

Leírás:

Integerek tömbjének a létrehozása.

**Figyelmeztetés:** Egy alkalmazásban legfeljebb **16** tömb van. A tömbök **integer analóg** értékeket tartalmaznak. A dinamikus memóriakiosztási műveletek végrehajtásakor ez a funkció rendszerhibát okozhat, ha a tömb mérete túl közel van a rendelkezésre álló memória méretéhez.

(\* FBD program, egy integer tömb létrehozásakor \*)



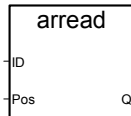
(\* ST Equivalencia \*)

array\_error := (ARCREATE (ident, 16) <> 1);

(\* IL Equivalencia: \*)

```
LD      ident
ARCREATE 16
NE      1
ST      array_error
```

## ARREAD



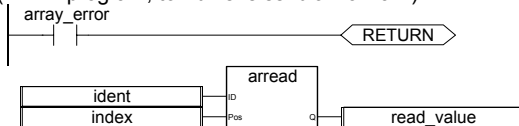
Argumentumok:

<b>ID</b>	INT	a tömb azonosítója (a [0..15] készletben kell lennie)
<b>Pos</b>	INT	az elem pozíciója a tömbben a [0 .. méret-1] készletben kell lennie
<b>value</b>	INT	az olvasott elem értéke 0 ha az argumentumok nem érvényesek

Leírás:

Kiolvás egy elemet egy integer tömbben.

(\* FBD program, tömb kezelési blokkokkal \*)



(\* ST Equivalencia \*)

```
If (array_error) Then Return; End_if;
read_value := ARREAD (ident, index);
(* a array_error az ARCREATE hívásból jön *)
```

(\* IL Equivalencia: \*)

```
LD      array_error
RETC
```

LD	ident
ARREAD	index
ST	read_value

## ARWRITE



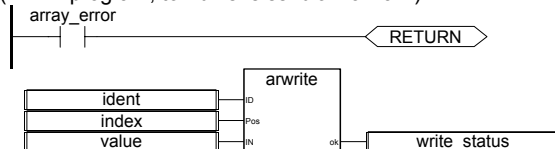
Argumentumok:

<b>ID</b>	INT	a tömb azonosítója (a [0..15] készletben kell lennie)
<b>Pos</b>	INT	az elem pozíciója a tömbben a [0 .. méret-1] készletben kell lennie
<b>IN</b>	INT	új érték az elemhez
<b>ok</b>	INT	végrehajtás státusza: <b>1</b> = az írás sikeres volt <b>2</b> = érvénytelen tömb azonosító <b>3</b> = érvénytelen index

Leírás:

Egy integer tömbben eltárol (beír) egy értéket.

(\* FBD program, tömb kezelési blokkokkal \*)



(\* ST Equivalencia \*)

```
If (array_error) Then Return; End_if;
write_status := ARWRITE (Ident, Index, value);
(* a array_error az ARCREATE hívásból jön *)
```

(\* IL Equivalencia: \*)

```
LD      array_error
RETCL
LD      ident
ARWRITE index,value
ST      write_status
```

**F\_ROPEN**

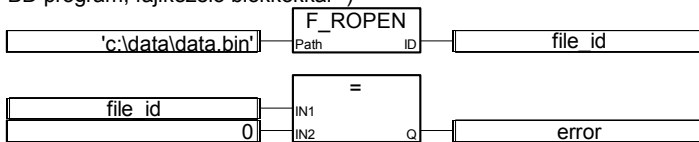
Argumentumok:

<b>Path</b>	MSG	fájl neve Ez tartalmazhatja a fájl elérési útját, a \ vagy / szimbólumot használva egy alkönyvtár meghatározásához. Az alkalmazás hordozhatóságának megkönnyítésére a / vagy \ egyenértékű.
<b>ID</b>	INT	fájl száma 0 ha egy hiba történik: a fájl nem létezik.

Leírás:

Egy bináris fájlt olvasási módban megnyit. Az FX\_READ-al és F\_CLOSE-al használandó. Ezt a funkciót az ISaGRAF szimulátor nem tartalmazza.

(\* FBD program, fájlkezelő blokkokkal \*)



(\* ST Equivalencia \*)

```

file_id := F_ROPEN('c:\data\data.bin');
error := (file_id=0);

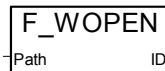
```

(\* IL Equivalencia: \*)

```

LD      'c:\data\data.bin'
F_ROPEN
ST      file_id
EQ      0
ST      hiba

```

**F\_WOPEN**

Argumentumok:

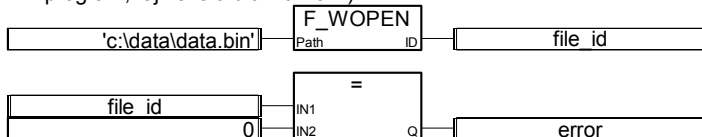
<b>Path</b>	MSG	fájl neve Ez tartalmazhatja a fájl elérési útját, a \ vagy / szimbólumot használva egy alkönyvtár meghatározásához. Az alkalmazás hordozhatóságának megkönnyítésére a / vagy \ egyenértékű.
<b>ID</b>	INT	fájl száma

0 ha egy hiba keletkezik. Ha a fájl már létezik, akkor felülíródik.

Leírás:

Egy bináris fájlt megnyit írás módban. Az FX\_WRITE-al és F\_CLOSE-al használandó. Ezt a funkciót az ISaGRAF szimulátor nem tartalmazza.

(\* FBD program, fájlkezelő blokkokkal \*)



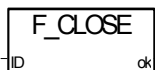
(\* ST Equivalencia \*)

```
file_id := F_WOPEN('c:\ data \ data.bin');
error := (file_id=0);
```

(\* IL Equivalencia: \*)

```
LD      'c:\data\data.bin'
F_WOPEN
ST      file_id
EQ      0
ST      hiba
```

## F\_CLOSE



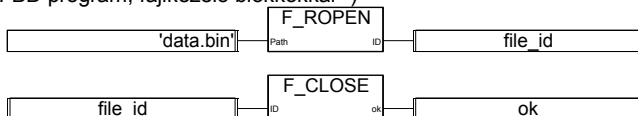
Argumentumok:

<b>ID</b>	INT	fájl száma: visszaadva az F_ROPEN vagy F_WOPEN által.
<b>ok</b>	BOO	return status TRUE ha a fájl bezárása OK FALSE ha egy hiba történt

Leírás:

Egy fájlt bezár az F\_ROPEN vagy F\_WOPEN funkciókkal. Ezt a funkciót az ISaGRAF szimulátor nem tartalmazza.

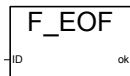
(\* FBD program, fájlkezelő blokkokkal \*)



```
(* ST Equivalencia *)
file_id := F_ROPEN('data.bin');
ok := F_CLOSE(file_id);
```

```
(* IL Equivalencia: *)
LD      'data.bin'
F_ROPEN
ST      file_id
F_CLOSE      (* file_id már a pillanatnyi IL eredményben van *)
ST      ok
```

## F\_EOF



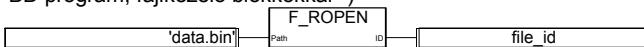
Argumentumok:

<b>ID</b>	INT	fájl száma: visszaadva az F_ROPEN vagy F_WOPEN által.
<b>ok</b>	BOO	fájl vége jelző TRUE ha a legutóbbi olvasási vagy írási procedúra meghíváskor el lett érve a fájl vége. Az FM_READ-dal egy fájlból kiolvasott legutolsó üzenet esetleg pontatlan lehet, ha a legutolsó karakter nem egy karaktorsor lezáró.

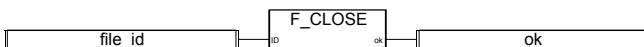
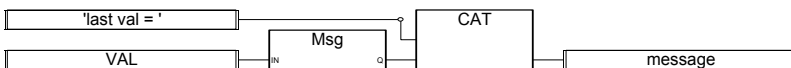
Leírás:

Ellenőrzi, hogy a fájl vége el lett-e érve.  
Ezt a funkciót az ISaGRAF szimulátor nem tartalmazza.

(\* FBD program, fájlkezelő blokkokkal \*)



not\_eof:



```
(* ST Equivalencia *)
file_id := F_ROPEN('data.bin');
```

```

WHILE not(F_EOF(file_id))
    VAL := FA_READ(file_id);
END_WHILE ;
MESSAGE := 'last val = ' + msg(VAL);
ok := F_CLOSE(file_id);

```

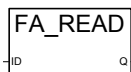
(\* IL Equivalencia: \*)

```

LD      'data.bin'
F_ROPEN
ST      file_id
LD      file_id
F_EOF
JMPNC   END_OF_FILE
NOT_EOF: LD      file_id
FA_READ
ST      VAL
LD      file_id
F_EOF
JMPNC   NOT_EOF (* ha nem eof, akkor tovább olvas *)
END_OF_FILE: LD      VAL
MSG
ST      val_msg (* VAL konverziója egy üzenetté *)
LD      'last val = '
ADD     val_msg
ST      MESSAGE
LD      file_id
F_CLOSE
ST      ok

```

## FA\_READ



Argumentumok:

<b>ID</b>	INT	fájl száma: F_ROPEN által visszaküldve.
<b>Q</b>	INT	integer analóg érték fájlból olvasva

Leírás:

ANALOG változókat olvas egy bináris fájlból. Az F\_ROPEN-nel és F\_CLOSE-al használandó.

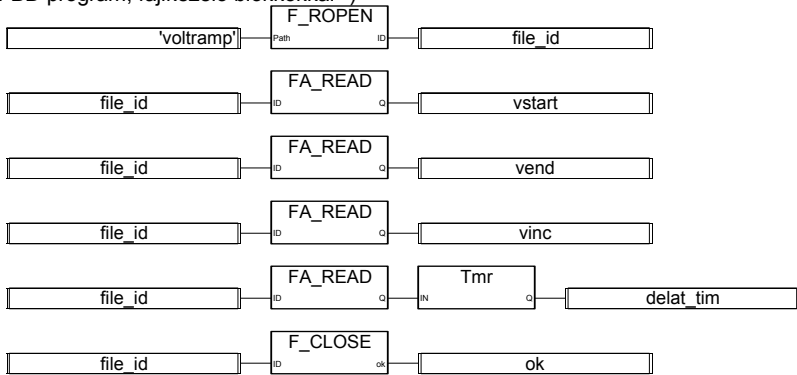
Ez az eljárás szekvenciális hozzáférést végez a fájlhoz, az előző pozícióból.

Az F\_ROPEN utáni első meghívás kiolvassa a fájl első 4 bájtyát, minden meghívás eltolja az olvasási mutatót.

Annak ellenőrzésére, hogy el lett-e érve a fájl vége, az F\_EOF-t kell használni.

Ezt a funkciót az ISaGRAF szimulátor nem tartalmazza.

(\* FBD program, fájlkezelő blokkokkal \*)



(\* ST Equivalencia \*)

```

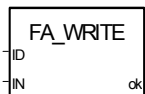
file_id := F_ROPEN('voltramp.bin');
vstart := FA_READ(file_id);
vend := FA_READ(file_id);
vinc := FA_READ(file_id);
delta_tim := tmr(FA_READ(file_id));
ok := F_CLOSE(file_id);
  
```

(\* IL Equivalencia: \*)

```

LD      'voltramp.bin'
F_ROPEN
ST      file_id
FA_READ      (* vstart olvasása *)
ST      vstart
LD      file_id
FA_READ      (* vend olvasása *)
ST      vend
LD      file_id
FA_READ      (* vinc olvasása *)
ST      vinc
LD      file_id
FA_READ      (* delta_tim olvasása *)
TMR      (* konverzió időzítéssé *)
ST      delta_tim
LD      file_id
F_CLOSE
ST      ok
  
```

## FA\_WRITE



Argumentumok:

<b>ID</b>	INT	fájl száma: a F_WOPEN által visszaadva.
<b>IN</b>	INT	a fájlba írandó integer analóg érték
<b>OK</b>	BOO	végrehajtás státusza: TRUE ha OK

Leírás:

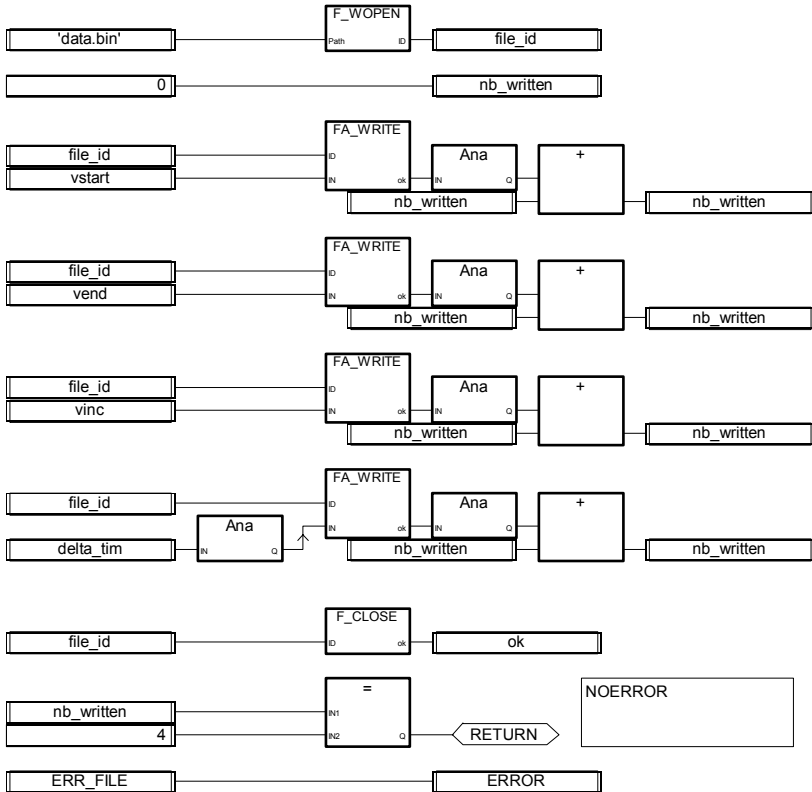
ANALOG változókat ír egy bináris fájlba.

Ez az eljárás szekvenciális hozzáférést végez a fájlhoz, az előző pozícióból.

Az F\_WOPEN utáni első meghívás beírja a fájl első 4 bájtyát, minden további meghívás eltolja az írási mutatót.

Ezt a funkciót az ISaGRAF szimulátor nem tartalmazza.

(\* FBD program \*)



(\* ST Equivalencia \*)

```

file_id := F_WOPEN('voltramp.bin');
nb_written := 0;
nb_written := nb_written + ana(FA_WRITE(file_id,vstart));
nb_written := nb_written + ana(FA_WRITE(file_id,vend));
nb_written := nb_written + ana(FA_WRITE(file_id,vinc));
nb_written := nb_written + ana(FA_WRITE(file_id,ana(delta_tim)));
ok := F_CLOSE(file_id);
IF ( nb_written <> 4) THEN
    ERROR := ERR_FILE;
END_IF;
  
```

(\* IL Equivalencia: \*)

```

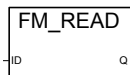
LD      'voltramp.bin'
F_WRITE
ST      file_id
  
```

```

LD      0
ST      nb_written
LD      file_id      (* vstart írása *)
FA_WRITE vstart
ANA
ADD      nb_written
ST      nb_written
LD      file_id      (* vend írása *)
FA_WRITE vend
ANA
ADD      nb_written
ST      nb_written
LD      file_id      (* vinc írása *)
FA_WRITE vinc
ANA
ADD      nb_written
LD      (* delta_tim írása *)
ANA      (* integerré konvertálás *)
ST      ana_delta_tim
LD      file_id
FA_WRITE ana_delta_tim
ANA
ADD      nb_written
ST      nb_written
F_CLOSE
ST      ok
LD      nb_written
EQ      4
RETC      (* visszatérés ha egyenlő 4-el *)
LD      ERR_FILE      (* különben hiba *)
ST      HIBA

```

## FM\_READ



Argumentumok:

<b>ID</b>	INT	fájl száma: F_OPEN által visszatadva.
<b>Q</b>	MSG	fájlból olvasott üzenet érték

Leírás:

Egy bináris fájlból MESSAGE változókat olvas.

Az F\_OPEN-nel és F\_CLOSE-al használandó.

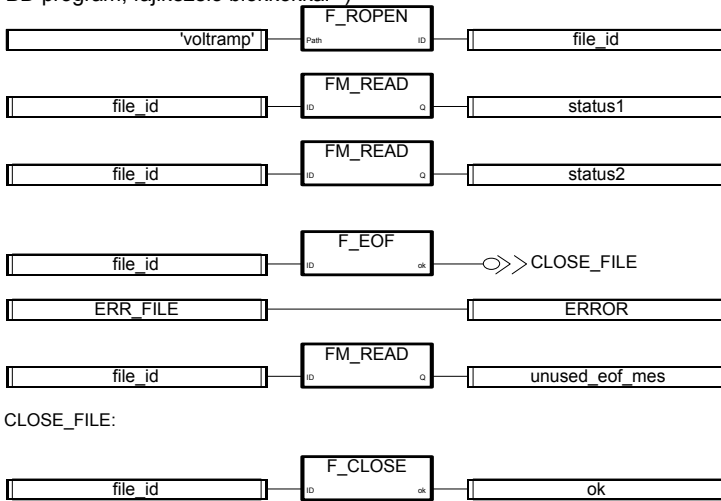
Ez az eljárás szekvenciális hozzáférést végez a fájlhoz, az előző pozícióból.

Az F\_OPEN utáni első meghívás kiolvassa a fájl első karaktersorát, minden meghívás eltolja az olvasási mutatót.

Egy karaktersorozat egy nullával (0), sor végével ('\ n'), vagy return-nal ('\ r') van lezárva; Annak ellenőrzésére, hogy el lett-e érve a fájl vége, az F\_EOF-t kell használni.

Ezt a funkciót az ISaGRAF szimulátor nem tartalmazza.

(\* FBD program, fájlkezelő blokkokkal \*)



(\* ST Equivalencia \*)

```

file_id := F_ROPEN('voltramp.bin');
status1 := FM_READ(file_id);
status2 := FM_READ(file_id);
IF (F_EOF(file_id)) THEN
    ERROR := ERR_FILE;
    unused_eof_mes := FM_READ(file_id);
END_IF;
ok := F_CLOSE(file_id);
  
```

(\* IL Equivalencia: \*)

```

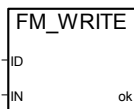
LD      'voltramp.bin'
F_ROPEN
ST      file_id
FM_READ      (* status1 olvasása *)
ST      status1
LD      file_id
FM_READ      (* status2 olvasása *)
ST      status2
LD      file_id
F_EOF
JMPNC    CLOSE_FILE (* ha nem lett elvégezve fájl végére ugrás *)
LD      ERR_FILE
ST      HIBA
LD      file_id
FM_READ      (* unused_eof_mes olvasása *)
ST      unused_eof_mes
  
```

```

CLOSE_FILE  LD      file_id
             F_CLOSE
             ST      ok

```

## FM\_WRITE



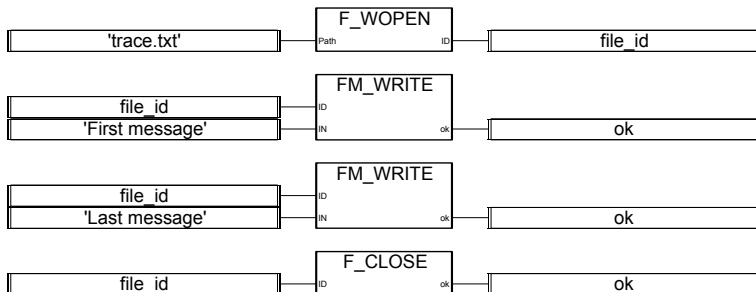
Argumentumok:

<b>ID</b>	INT	fájl száma: a F_WOPEN által visszaadva.
<b>IN</b>	MSG	a fájlba írandó üzenet érték
<b>ok</b>	BOO	végrehajtás státusza: TRUE ha sikerült

Leírás:

Egy bináris fájlba MESSAGE változókat ír.  
 Az F\_WOPEN-nel és F\_CLOSE-al használandó.  
 Egy üzenet nulla lezárásos karaktersorként van egy fájlba írva.  
 Ez az eljárás szekvenciális hozzáférést végez a fájlhoz, az előző pozícióból.  
 A F\_WOPEN utáni első meghívás a fájl első karaktersorát írja be,  
 minden további meghívás eltolja az írási mutatót.  
 Ezt a funkciót az ISaGRAF szimulátor nem tartalmazza.

(\* FBD program, fájlkezelő blokkokkal \*)



(\* ST Equivalencia \*)

```

file_id := F_WOPEN('trace.txt');
ok := FM_WRITE(file_id,'First message');
ok := FM_WRITE(file_id,'Last message');
ok := F_CLOSE(file_id);

```

(\* IL Equivalencia: \*)

```

LD      'trace.txt'
F_WOPEN

```

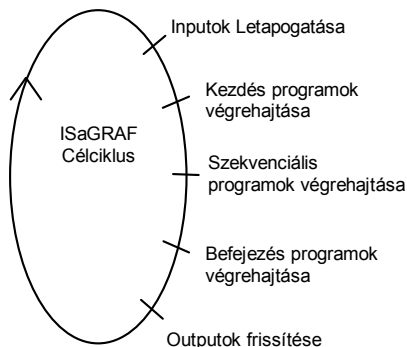
ST	file_id	
FM_WRITE	'First message'	(* első msg írása *)
ST	ok	
LD	file_id	
FM_WRITE	'Last message'	(* második msg írása *)
ST	ok	
LD	file_id	
F_CLOSE		
ST	ok	



## **C. Cél Kezelési útmutató**

## C.1. Bevezetés

Az ISaGRAF cél egy valós idejű szoftver, amely egy ISaGRAF alkalmazást futtat az Ön ipari számítógépén illetve kártyáján, a következő jól ismert séma szerint:



A célciklus a folyamat fizikai inputjainak a letapogatásából áll, az ISaGRAF workbench alkalmazási programjai szerinti alkalmazási adatok feldolgozására és vezérlésére,<sup>1</sup> majd a fizikai outputok frissítésére.

- Ennek a fejezetnek az első része azt ismerteti, hogy hogyan kell elkezdeni a munkavégzést egy adott rendszer céllal. Egymás utáni sorrendben: DOS, OS-9, VxWorks és NT cél. Először mindegyiknél azt találhatja, hogy hogyan kell az ISaGRAF célt futtatni. Ezután a következő specifikus lehetőségekről kap információkat: cél indítás a bekapcsoláskor, hibakezelés, általános viselkedés, stb.
- A második rész a felhasználói C funkciók, funkcióblokkok, és konverziós funkciók telepítési módszereivel foglalkozik, az ISaGRAF cél kiemelésére.
- A harmadik rész a Modbusról és az ISaGRAF megvalósításról nyújt tájékoztatást. Ez ismerteti a különböző funkciókódok keret formáit.
- A negyedik rész bizonyos eszközöket nyújt az áramkimaradáshoz és a cél újraindításához.

---

<sup>1</sup> Ez a kézikönyv feltételezi, hogy a felhasználó előtt ismert az ISaGRAF Workbench

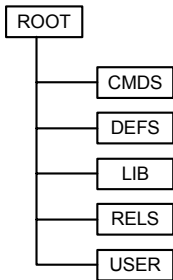
## C.2. Telepítés

A telepítés kb. 1 Mbyte szabad lemezterületet igényel. A lemezzel szállított install.bat fájl a PC-n egy meghatározott platformhoz szükséges összes fájlt telepíti.

Példa: a:\install a: c:\*path*

az a: lemezmeghajtóról a fájlokat a c:-re, a *path* alkönyvtárba telepíti.

A következő alkönyvtár architektúra van alkalmazva :



a GYÖKÉR alkönyvtár bizonyos eszközöket és olvass el fájlokat tartalmaz

a CMDS alkönyvtár végrehajtható fájlokat tartalmaz

a DEFS alkönyvtár fejsor definíciós fájlokat tartalmaz

a LIB alkönyvtár könyvtárakat tartalmaz

a RELS alkönyvtár áthelyezhető (tárgy) fájlokat tartalmaz

a USER (FELHASZNÁLÓ) alkönyvtár felhasználói „C” eljárásokat tartalmaz C funkciókhoz, funkcióblokkokhoz, és konverziós funkciókhoz (forrás és fejsor fájlok)

Ezután a telepített platformmal el lehet kezdeni e munkavégzést.

## C.3. Az ISaGRAF DOS céllal történő munkavégzés elkezdése

### C.3.1. Az ISaGRAF futtatása: ISA.EXE

Az MS-DOS telepítésben a cél egyetlen programot futtat: ISA.EXE. Az elkezdéshez egyszerűen futtassa a CMDS alkönyvtárból az isa -? Parancsot.

Egy ilyen rendszerben a műveletek kritikus fontosságúak lehetnek. A jó teljesítmény eléréséhez pl. ajánlatos nem túlterhelni a kommunikációs kapcsolatot. A cél program nem akadályozza meg a megszakítás által meghajtott rutinok futtatását.

#### **Kommunikációs kapcsolat és konfiguráció: -t Beállítási lehetőség**

Az ISaGRAF cél egy soros kapcsolatot alkalmaz a hibakereső kommunikációhoz. A port számát a -t beállítás határozza meg. Mivel a kommunikációs interfész úgy lett tervezve, hogy bármilyen géppel kompatibilis legyen, ezért a COM1, COM2 vagy COM3 portokat lehet használni, a BIOS verzióknak megfelelően.

**Nincs alapértelmezett érték:** Ha ez a beállítás nincs használva, akkor nem lehetséges a céllal való kommunikáció. Ilyen esetben a 7-es hibaszám jelenhet meg.

A DOS ISaGRAF céllal nem lehetséges az Ethernet kapcsolattal történő kommunikáció. A speciális kivitelezéssel kapcsolatban forduljon a beszállítóhoz.

A kommunikációs paramétereket az ISaGRAF elindítása előtt be kell állítani, hogy a felhasználó teljesen szabadon használhassa a szükséges paramétereket. A workbench hibakereső használatakor ellenőrizni kell, hogy a workbench kommunikációs paraméterei (lásd a kezelési útmutató Programok kezelése című fejezetét) megfelelőjenek a célnak.

#### Példa:

MODE COM1:9600,N,8,1

A következő értékekre állítja be a kommunikációs paramétereket:

- átviteli sebesség: 9600
- nincs paritásellenőrzés
- 8 bites adatok
- 1 stop bit

Megjegyzendő, hogy bizonyos BIOS verziókon a kiindulási 19200 baudos workbench beállítás nincs megengedve.

A CJ a workbench paraméterek beállítására az ISAMOD.EXE segédprogramot mellékelte:

ISAMOD COM1

Megfelel a következőnek: MODE COM1:19200,N,8,1

#### **Kiszolgáló szám: -s Beállítási lehetőség**

Ez a beállítás a cél kiszolgáló számát határozza meg. Ez 1 és 255 között lehet, kivéve a 13-as számot (\$0D). Ez a kiszolgáló szám van használva a kommunikációs kapcsolat

protokolljában. Ennek elsődleges célja az, hogy megkülönböztesse egymástól a kiszolgálókat, ha egynél több cél van összekapcsolva egymással. A workbench hibakereső használatakor ellenőrizni kell, hogy a workbench kiszolgáló paraméterei (lásd a kezelési útmutató Programok kezelése című fejezetét) megfeleljenek a célnak.

**Alapértelmezett érték:** Az alapértelmezett kiszolgáló szám 1 (ugyanaz, mint az egyes számú workbench)

### **Példák:**

**isamod COM1** A COM1-et 19200 baudra konfigurálja, paritás nélkül, 8 adat bittel, 1 stop bittel.

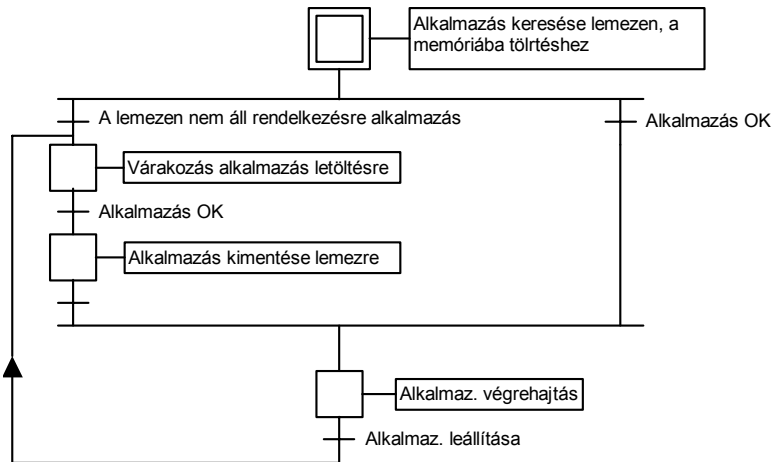
**isa -t=COM1** Az ISaGRAF célt az (1) kiindulási kiszolgáló számmal, és a COM1-el, mint kommunikációs porttal indítja el.

**isa -s=3 -t=COM1** Az ISaGRAF célt a 3 kiszolgáló számmal, és a COM1-el, mint kommunikációs porttal indítja el.

## **C.3.2. Különleges jellemzők:**

### **ISaGRAF elindítás**

A cél elindításakor a következő algoritmus kerül végrehajtásra.



#### **Definíciók**

Az alkalmazás kód a workbench által generált és letöltött, majd a célon végrehajtott bináris adatokat jelenti. Ezt a szimbólumtáblázat egészítheti ki.

Az alkalmazási szimbólumtáblázat egy ASCII adatbázis, amelyet a workbench generál és tölt le. Ez a táblázat kapcsolatot képez a szimbólum tárgyak és a belső céltárgyak között. Ez a célon nem szükséges, a felhasználó-specifikus szimbólumkezelés kivételével. A

szimbólumtáblázattal kapcsolatos bővebb információk a Haladó programozási módszerek c. Kezelési útmutatóban találhatók:

- **Alkalmazás biztonsági másolat :**

Amikor egy új alkalmazás a workbench hibakeresőről a célra le van töltve, az alkalmazás kód a cél pillanatnyi alkönyvtárába kimentésre kerül, a következő fájlnevvvel:

**ISAx1** ISaGRAF alkalmazás kód biztonsági másolat fájl (ahol x a kiszolgáló szám)

Emellett, ha előzőleg le lett töltve az alkalmazás szimbólumtáblázat, akkor az szintén kimentésre kerül a cél pillanatnyi alkönyvtárába, az alábbi fájlnevvvel:

**ISAx6** ISaGRAF alkalmazás szimbólum biztonsági másolat fájl (ahol x a kiszolgáló szám)

Az ISaGRAF cél elindításakor ezek az alkalmazás kód és alkalmazás szimbólum fájlok megkeresésre kerülnek, és betöltődnek a memóriába.

Ha nem áll rendelkezésre szimbólum fájl, akkor a cél elkezd futtatni az alkalmazás kódot, betöltött szimbólumok nélkül.

Ha nem áll rendelkezésre alkalmazás kód, akkor a cél vár egy alkalmazás letöltésére.

Annak érdekében, hogy a cél a bekapcsoláskor egy adott alkalmazással, a hibakereső kapcsolat használata nélkül induljon el, ezek a fájlok közvetlenül a cél pillanatnyi alkönyvtárába másolhatók ugyanarról a lemeztől, ha a workbench ugyanazon a PC-n van, vagy egy floppy lemezt használ. Ha a cél gépen nincs lemez, akkor lehet egy virtuális lemezt is használni.

Ha az ISaGRAF workbench a standard \ISAWIN alkönyvtárba van telepítve: akkor a MYPROJ projekt alkalmazás kód fájlja a következő:

\ISAWIN\APL\MYPROJ\appli.x8m

a MYPROJ projekt alkalmazás szimbólum fájlja a következő:

\ISAWIN\APL\MYPROJ\appli.tst

### Példa:

Ha az alábbi parancsot írja be abból az alkönyvtárból, ahol az isa.exe van telepítve:

copy \ISAWIN\APL\MYPROJ\appli.x8m isa11

akkor az isa.exe megkeresi és végrehajtja a „myproj” alkalmazást.

Mindezeket a parancsokat pl. egy csomagfájlba lehet csoportosítani, és azokat a workbench eszköz menüjéről el lehet indítani (lásd a programok kezelése c. Kezelési útmutatót).

## **Hibakezelés és output üzenetek**

Az ISaGRAF célszoftver magában foglalja a hibaérzékelés kezelését. A figyelmeztető hibaüzenetek listája és azok leírása a mellékletben található.

A hibaérzékelés feldolgozása az alábbiak szerint történik:

- Egy hiba egy hiba- és egy argumentum számból áll, amelyek az ISaGRAF hibarutinhoz vannak küldve
- Ha a workbench Létrehozás beállításainál be lett kapcsolva a hibaérzékelés jelző, akkor a hiba feldolgozásra kerül. Ha nem, akkor az információ elvész, és a hibakezelés befejeződik.

A feldolgozáskor:

- A hiba száma (decimális érték) és argumentuma (hexadecimális érték) megjelenítésre kerül az alapértelmezett stdout outputon
- A hibaszám és argumentum egy gyűrűs FIFO hibapufferbe tolódik, későbbi előhívás céljából. A hibapuffer mérete a workbench Létrehozás beállításoknál van beállítva. Amikor a puffer megtelik, akkor minden új bejövő hibánál a legrégebbi hiba elvész.
- A hibák vagy a hibakeresőből, vagy a futó alkalmazásból hozhatók elő, a SYSTEM hívással (lásd a Kezelési útmutatóban).

Amikor a hibakereső egy hibát érzékel, akkor a hiba ablakban megjelenik egy üzenet, amely leírja a hibát. Az alkalmazás kontextusától függően (fut, vagy nem), a hibakereső megjelenítheti vagy a tárgy nevét (változó, vagy program) ahonnan a hiba származik, vagy a négyzetes zárójelek közé zárt [x] argumentumot (decimális érték), amelynek minden hibánál más jelentése van.

Az alapértelmezett stdout outputon egy üdvözlő üzenet és a hibaértékek jelennek meg, amikor a cél elindul és egy hiba van érzékelve. Ha a megjelenítés nem kívánatos a standard output csatormán, akkor egy átirányító parancs használható, mint pl. a következő:

```
isa -t=COM1 -s=1 >NUL
```

## == Rendszer óra

Mivel az ISaGRAF cél úgy lett tervezve hogy bármilyen rendszeren fusson, ezért a ciklus szinkronizáláshoz illetve az időváltozó frissítéshez használt időhivatkozás a standard időérték, ami kb. 55 ezredmásodpercrek felel meg.

Ennélfogva tehát az időváltozókon nem lehet 55 ms-nál jobb pontosságot elérni. Ugyanezen ok miatt, egy 55 ms, vagy attól alacsonyabb, és zérótól eltérő megadott ciklusidőtartam egy ciklusidőtartam túlszordulási hibát fog generálni (62-es hiba), és nem indít el ciklusokat.

Annak előnye, ha nem módosítják a rendszer órajelet az, hogy a rezidens alkalmazásokat, illetve az alkalmazásba integrált C funkciókat és funkcióblokkokat soha nem fogja zavarni az ISaGRAF végrehajtása.

Amennyiben az Ön alkalmazása nagyobb pontosságot igényel, akkor kérjük, hogy egy speciális kivitelezés megoldása érdekében lépjen kapcsolatba a beszállítóval.

## == Kilépés billentyű

Egy alkalmazás nem ipari körülmények között, egy asztali PC-n történő tesztelése során előfordulhat, hogy a felhasználó esetleg meg akarja állítani az ISaGRAF-ot: ez a váratlan megállások elkerülése érdekében egy gombnyomás-kombinációval érhető el. Ez a gombnyomás-kombináció a következő:

**shift + ctrl + alt**

Természetesen, amennyiben el akarják kerülni hogy egy gomb lenyomására az ipari alkalmazás leálljon, akkor valamit tenni kell ezeknek a kombinációknak a letiltása érdekében.

Ezeknek a gyors kilépéseknek egy veszélyes mellékhatása az, hogy az IO kártya interfész nincs bezárva. Így tehát, az ISaGRAF cél leállításának tiszta megoldása a következő:

- állítsa le az alkalmazást a hibakeresőből (ez lezárja az IO kártyákat)
- állítsa le az ISaGRAF célt a billentyűzetről

### **Az alkalmazás mérete**

Minthogy az ISaGRAF MS-DOS cél Intel valós módhoz van tervezve, ezért az adatstruktúra maximális mérete 64K. A workbench által letöltött alkalmazás kódnak tehát nem szabad meghaladnia ezt a határértéket. Bizonyos nagyon ritka esetekben az ISaGRAF által allokalált belső struktúra szintén meghaladhatja ezt a határértéket, és a letöltés után az alkalmazás összeomlását okozhatja. Ezen felül, az összes rendelkezésre álló memória 640K konvencionális memóriára korlátozódik.

Amennyiben az Ön alkalmazása nagyobb memóriakapacitást igényel, akkor kérjük, hogy egy speciális kivitelezés megoldása érdekében lépjen kapcsolatba a beszállítóval.

## C.4. Az ISaGRAF OS9 céllal történő munkavégzés elkezdése

Legelőször is, valamilyen fájlátviteli eszközzel át kell vinnie a fájlokat (legalábbis a végrehajtandó fájlokat a CMDS alkönyvtárból) az OS-9 célra.

Az elkezdéshez egyszerűen futtassa az OS-9 rendszer CMDS alkönyvtárból az alábbi súgó (help) parancsokat:

```
isa -?
isaker -?
isatst -?
Isanet -?
```

### C.4.1. Az ISaGRAF egyszeres feladat futtatása: isa

Az ISaGRAF cél futtatható egyetlen feladatként. Egy ilyen konfigurációban azonban a műveletek kritikus fontosságúak lehetnek. A jó teljesítmény eléréséhez pl. ajánlatos nem túlterhelni a kommunikációs kapcsolatot. Az OS-9 multitaszking rendszeren ugyanazon a CPU-n futtathatók különböző egyfeladatos ISaGRAF célok, feltéve hogy kiszolgáló számuk és kommunikációs portjuk különböző.

Ez az egyfeladatos kivitelezés elsősorban gyenge hardverrel rendelkező platformokhoz lett tervezve, pl. olcsó kártyákkal illetve MS-DOS PC-kkel, illetve egy új platformon történő portoláskor egy első prototípus létrehozásánál. Éppen ezért a multitaszking ISaGRAF cél kivitelezést kell előnyben részesíteni.

Az ISaGRAF egyfeladatos cél nem akadályozza meg a háttérfolyamatok vagy a megszakítás által meghajtott rutinok futását.

#### ***⇒ Kommunikációs kapcsolat és konfiguráció: -t Beállítási lehetőség***

Az ISaGRAF egyfeladatos cél egy soros kapcsolatot alkalmaz a hibakereső kommunikációhoz. A leíró nevét a -t beállítás határozza meg.

**Nincs alapértelmezett érték:** Ha ez a beállítás nincs használva, akkor nem lehetséges a céllal való kommunikáció. Ilyen esetben a 7-es hibaszám jelenhet meg.

Az egyfeladatos kivitelezéssel nem áll rendelkezésre az Ethernet kapcsolatot alkalmazó kommunikáció.

A soros kapcsolatú eszköz bináris adatátviteli módban van megnyitva (nincs vezérlőkarakter, nincs XON/XOFF). A többi kommunikációs paramétereket az ISaGRAF elindítása előtt be kell állítani, hogy a felhasználó teljesen szabadon használhassa a szükséges paramétereket. A workbench hibakereső használatakor ellenőrizni kell, hogy a workbench kommunikációs paraméterei (lásd a kezelési útmutató Programok kezelése című fejezetét) megfeleljenek a célnak.

Példa:

```
xmode /t0 baud=19200
```

9200 baudos kommunikációs adatátviteli sebességet állít be a /t0 eszközön

### **⇒ Kiszolgáló szám: -s Beállítási lehetőség**

Ez a beállítás a cél kiszolgáló számát határozza meg. Ez 1 és 255 között lehet, kivéve a 13-as számot (\$0D). Ez a kiszolgáló szám van használva a kommunikációs kapcsolat protokolljában. Ennek célja az, hogy megkülönböztesse egymástól a kiszolgálókat, ha egynél több cél fut. A workbench hibakereső használatakor ellenőrizni kell, hogy a workbench kiszolgáló paraméterei (lásd a kezelési útmutató Programok kezelése című fejezetét) megfeleljenek egy létező célnak.

**Alapértelmezett érték:** Az alapértelmezett kiszolgáló szám 1 (ugyanaz, mint az egyes számú workbench)

### **⇒ Példák:**

**isa -t=/t0** Egy ISaGRAF célt az (1) kiindulási kiszolgáló számmal, és a /t0-al, mint kommunikációs porttal indítja el.

**isa -s=3 -t=/t1** Egy ISaGRAF egyfeladatos célt a (3) kiindulási kiszolgáló számmal, és a /t1-el, mint kommunikációs porttal indítja el.

**isa -t=/t0 &**

**isa -s=3 -t=/t1** Két ISaGRAF egyfeladatos célt indít el. Az egyiket az (1) kiindulási kiszolgáló számmal, és a /t0-al, mint kommunikációs porttal. A másikat a (3) kiindulási kiszolgáló számmal, és a /t1-el, mint kommunikációs porttal.

## **C.4.2. Az ISaGRAF multitaszkok futtatása: isaker, isatst, isanet**

Az ISaGRAF cél kernel valamint a kommunikációs kapcsolat válaszüzenetének javítása érdekében a cél két feladatra van osztva, ami különválasztja a kommunikációs feladatot (isatst vagy isanet kommunikációs feladat) az alkalmazás végrehajtásától (isaker kernel feladat).

Az ilyen architektúra rugalmasabb. Ez lehetővé teszi a felhasználó számára egynél több, ugyanahhoz a kernelhez kapcsolt kommunikációs feladat futtatását, vagy egy kommunikációs feladattal maximum 4 kernel futtatását. Ez bizonyos integrációkat, pl. ugyanazon az alkalmazáson egy folyamat vizualizálási kapcsolatot és a workbench hibakereső kapcsolatot, illetve ugyanazon a fizikai porton keresztül maximum 4 különböző alkalmazás egyszeres kapcsolatát teszi lehetővé.

A kernel és kommunikációs feladatok függetlenek, és külön-külön elágaztathatók. Az egyetlen kikötés az, hogy a kernel feladato(ka)t kell először elindítani, hogy az inicializálja saját rendszerkörnyezetét, és így a kommunikációs feladat(ok) kapcsolni tudja(ák) azt.

Az ISaGRAF multitaszk (többfeladatos) cél nem akadályozza meg a háttérprogramok vagy a megszakítás által meghajtott rutinok futását.

### **C.4.2.1. A kernel feladat futtatása: isaker**

### **⇒ Kiszolgáló szám: -s Beállítási lehetőség**

Ez a beállítás a cél kernel kiszolgáló számát határozza meg. Ez 1 és 255 között lehet, kivéve a 13-as számot (\$0D). Ez a kiszolgáló szám van használva a kommunikációs kapcsolat

protokolljában és a kernelhez kapcsolt kommunikációs feladat(ok) által. Ennek célja az, hogy megkülönböztesse egymástól a kiszolgálókat, ha egynél több cél fut.

**Alapértelmezett érték:** Az alapértelmezett kiszolgáló szám 1 (ugyanaz, mint az egyes számú workbench)

#### C.4.2.2. A soros kommunikációs feladat futtatása: isatst

##### ☐ **Kommunikációs kapcsolat és konfiguráció: -t Beállítási lehetőség**

Az isatst cél kommunikációs feladat egy soros kapcsolatot alkalmaz a hibakereső kommunikációhoz. A leíró nevét a -t beállítás határozza meg.

**Nincs alapértelmezett érték:** Ha ez a beállítás nincs használva, akkor nem lehetséges a céllal való kommunikáció. Ilyen esetben a 7-es hibaszám jelenhet meg.

Az isatst kivitelezéssel nem áll rendelkezésre egy Ethernet kapcsolatot alkalmazó kommunikáció.

A soros kapcsolatu eszköz bináris adatátviteli módban van megnyitva (nincs vezérlőkarakter, nincs XON/XOFF). A többi kommunikációs paramétereket az ISaGRAF elindítása előtt be kell állítani, hogy a felhasználó teljesen szabadon használhassa a szükséges paramétereket. A workbench hibakereső használatakor ellenőrizni kell, hogy a workbench kommunikációs paraméterei (lásd a kezelési útmutató Programok kezelése című fejezetét) megfelelőjenek a célnak.

Példa:

xmode /t0 baud=19200

9200 baudos kommunikációs adatátviteli sebességet állít be a /t0 eszközhöz

##### ☐ **Kiszolgáló szám: -s Beállítási lehetőség**

Ez a beállítás a cél kernel kiszolgáló számát (számait) határozza meg, amelyhez a kommunikációs feladat kapcsolva van. Ez 1 és 255 között lehet, kivéve a 13-as számot (\$0D). Ez a beállítási lehetőség maximum 4-szer megismételhető, 4 különböző kernel kiszolgáló összekapcsolásához. Ez a kiszolgáló szám van használva a kommunikációs kapcsolat protokolljában. Ennek célja az, hogy megkülönböztesse egymástól a kiszolgálókat, ha egynél több cél fut. A workbench hibakereső használatakor ellenőrizni kell, hogy a workbench kiszolgáló paraméterei (lásd a kezelési útmutató Programok kezelése című fejezetét) megfelelőjenek egy létező célnak (kernel és kommunikációs feladatok).

**Alapértelmezett érték:** Az alapértelmezett kiszolgáló szám 1 (ugyanaz, mint az egyes számú workbench)

##### ☐ **Kommunikációs feladat logikai száma: -c Beállítási lehetőség**

Ez a beállítás a kommunikációs feladat logikai számát határozza meg. Ez egynél több kommunikációs feladat egyszerre történő kezelésére szolgál. Ez 1 és 255 között lehet, és mindegyik kommunikációs feladathoz másnak kell lennie.

**Alapértelmezett érték:** Az utolsó -s specifikált beállítás van használva. Az alapértelmezett érték biztosítja a korábbi (3.0) ISaGRAF verziókkal való kompatibilitást.

#### **C.4.2.3. Az Ethernet kommunikációs feladat futtatása: isanet**

##### **⇒ Kommunikációs kapcsolat és konfiguráció: -t Beállítási lehetőség**

Az isanet cél kommunikációs feladat egy standard Ethernet kapcsolatot alkalmaz a hibakereső kommunikációhoz. A port számát a -t beállítás határozza meg.

**Nincs alapértelmezett érték:** Ha ez a beállítás nincs használva, akkor nem lehetséges a céllal való kommunikáció. Ilyen esetben a 7-es hibaszám jelenhet meg.

A workbench hibakereső használatakor ellenőrizni kell, hogy a workbench kommunikációs paraméterei (lásd a kezelési útmutató Programok kezelése című fejezetét) megfeleljenek a célnak.

Az ISaGRAF-nál az OS-9 cél a szerver, és a hibakereső a kliens, amely a megadott port számra kapcsolódik.

Az első hibakereső művelet Etherneten történő elkezdése előtt ellenőriznie kell, hogy az OS-9 Ethernet eszköz megfelelően konfigurálva van-e. Pl. egy ping küldhető az OS-9 rendszerre.

##### **⇒ Kiszolgáló szám: -s Beállítási lehetőség**

Ez a beállítás a cél kernel kiszolgáló számát (számait) határozza meg, amelyhez a kommunikációs feladat kapcsolva van. Ez 1 és 255 között lehet, kivéve a 13-as számot (\$0D). Ez a beállítási lehetőség maximum 4-szer megismételhető, 4 különböző kernel kiszolgáló összekapcsolásához. Ez a kiszolgáló szám van használva a kommunikációs kapcsolat protokolljában. Ennek célja az, hogy megkülönböztesse egymástól a kiszolgálókat, ha egynél több cél fut. A workbench hibakereső használatakor ellenőrizni kell, hogy a workbench kiszolgáló paraméterei (lásd a kezelési útmutató Programok kezelése című fejezetét) megfeleljenek egy létező célnak (kernel és kommunikációs feladatok).

**Alapértelmezett érték:** Az alapértelmezett kiszolgáló szám 1 (ugyanaz, mint az egyes számú workbench)

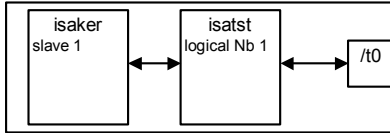
##### **⇒ Kommunikációs feladat logikai száma: -c Beállítási lehetőség**

Ez a beállítás a kommunikációs feladat logikai számát határozza meg. Ez egynél több kommunikációs feladat egyszerre történő kezelésére szolgál. Ez 1 és 255 között lehet, és mindegyik kommunikációs feladathoz másnak kell lennie.

**Alapértelmezett érték:** Az utolsó -s specifikált beállítás van használva. Az alapértelmezett érték biztosítja a korábbi (3.0) ISaGRAF verziókkal való kompatibilitást.

**C.4.2.4. Példák:**

**isaker &  
isatst -t=/t0**

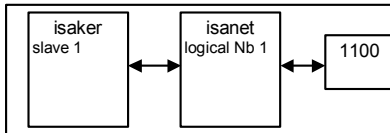


Elindít:

Egy ISaGRAF kernel feladatot az (1)-es alapértelmezett kiszolgáló számmal.

Egy ISaGRAF soros kommunikációs feladatot a /t0 com Porton, az (1)-es alapértelmezett kiszolgáló számhoz kapcsolva, és az alapértelmezett logikai számmal (utolsó megadott kiszolgáló szám = alapértelmezett = 1).

**isaker &  
isanet -t=1100**

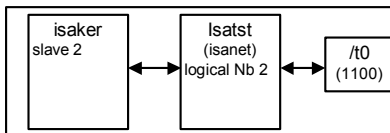


Elindít:

Egy ISaGRAF kernel feladatot az (1)-es alapértelmezett kiszolgáló számmal.

Egy ISaGRAF soros kommunikációs feladatot a 1100 számú Porton, az (1)-es alapértelmezett kiszolgáló számhoz kapcsolva, és az alapértelmezett logikai számmal (utolsó megadott kiszolgáló szám = alapértelmezett = 1).

**isaker -s=2 &  
isatst -t=/t0 -s=2 (illetve isanet -t=1100 -s=2)**

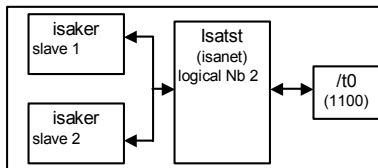


Elindít:

Egy ISaGRAF kernel feladatot az (2)-es alapértelmezett kiszolgáló számmal.

Egy ISaGRAF soros (Ethernet) kommunikációs feladatot a /t0 com Porton (Port szám: 1100), a (2)-es alapértelmezett kiszolgáló számhoz kapcsolva, és az alapértelmezett logikai számmal (utolsó megadott kiszolgáló szám = alapértelmezett = 2).

**isaker -s=1 &**

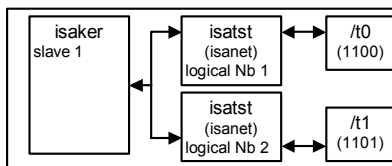
**isaker -s=2 &****isatst -t=/t0 -s=1 -s=2** (illetve isanet -t=1100 -s=1 -s=2)

Elindít:

Egy ISaGRAF kernel feladatot az (1)-es alapértelmezett kiszolgáló számmal.

Egy ISaGRAF kernel feladatot az (2)-es alapértelmezett kiszolgáló számmal.

Egy ISaGRAF soros (Ethernet) kommunikációs feladatot a /t0 com Porton (Port szám: 1100), az 1-es és 2-es alapértelmezett kiszolgáló számokhoz kapcsolva, és az alapértelmezett logikai számmal (utolsó megadott kiszolgáló szám = alapértelmezett = 2).

**isaker -s=1 &****isatst -t=/t0 -s=1 -c=1 &** (illetve isanet -t=1100 -s=1 -c=1 &)**isatst -t=/t1 -s=1 -c=2** (illetve isanet -t=1101 -s=1 -c=2)

Elindít:

Egy ISaGRAF kernel feladatot, 1-es kiszolgáló számmal.

Egy ISaGRAF soros (Ethernet) kommunikációs feladatot a /t0 com Porton (Port szám: 1100), a (1)-es alapértelmezett kiszolgáló számhoz kapcsolva, és 1-es logikai számmal.

Egy ISaGRAF soros (Ethernet) kommunikációs feladatot a /t1 com Porton (Port szám: 1101), a (1)-es alapértelmezett kiszolgáló számhoz kapcsolva, és 2-es logikai számmal.

Megjegyzés :

A soros és Ethernet kommunikációs feladatok keverhetők.

**C.4.3. Különleges jellemzők:****== Kommunikációs kapcsolat**

Mivel az OS-9 Soros Karakterkezelő nagyon flexibilis, ezért majdnem minden, a Microware által támogatott kétirányú fizikai eszköz használható:

Példa:

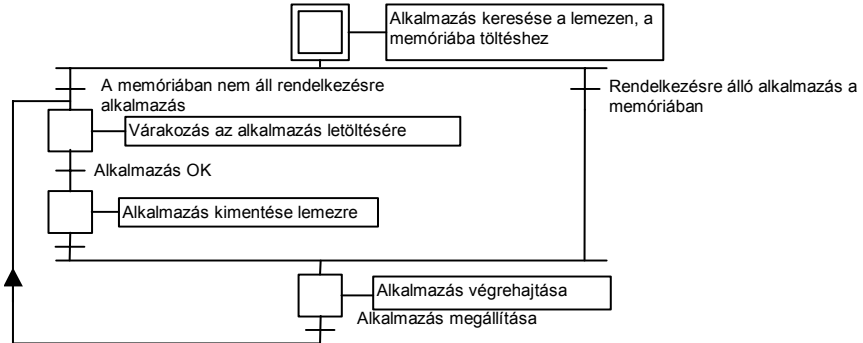
A soros kapcsolat lehet egy másik CPU-n elhelyezkedő fizikai porthoz vezető hálózati útvonal.

Ekkor a -t beállítás pl. a következőképpen lenne alkalmazva: -t=/nr/MASTER/t0

Ahol a kommunikációs kapcsolat egy ramnet hálózaton levő MASTER nevű CPU-n van deportolva. Az alkalmazott fizikai port /t0.

## ISaGRAF elindítás

A cél elindításakor a következő algoritmus kerül végrehajtásra.



### Definíciók

Az alkalmazás kód a workbench által generált és letöltött, majd a célon végrehajtott bináris adatokat jelenti. Ezt a szimbólumtáblázat egészítheti ki.

Az alkalmazási szimbólumtáblázat egy ASCII adatbázis, amelyet a workbench generál és tölt le. Ez a táblázat kapcsolatot képez a szimbólum tárgyak és a belső céltárgyak között. Ez a célon nem szükséges, a felhasználó-specifikus szimbólumkezelés kivételével. A szimbólumtáblázattal kapcsolatos bővebb információk a Kezelési útmutatóban találhatók: Haladó programozási eljárások.

### ISaGRAF OS-9 tárgyak és Multi-alkalmazás

Minden ISaGRAF publikus tárgy neve „ISAxn”-el kezdődik, ahol az **x** a kernel kiszolgáló száma, az **n** pedig egy adott jelentéssel rendelkező helyszám, kivéve az **ISAy3**-at, ahol az **y** a kommunikációs feladat logikai száma a multitaszk kivitelezésben.

Egy CPU-n egyidejűleg különböző alkalmazások(kernelek és kommunikációs feladatok) futhatnak, feltéve hogy azok mindegyikének eltérő kiszolgáló számaik és kommunikációs feladat logikai számaik vannak. Mindenesetre, különböző alkalmazások futtatásakor a felhasználónak ügyelnie kell bizonyos alkalmazás tárgyak, pl. I/O kártyák megosztott hozzáférésére. Pl. különböző alkalmazások (kernelek) különböző fizikai kártyákat használhatnak, hacsak valamilyen fajta I/O szerver vagy szemafor nincs kivitelezve az I/O meghajtó révén.

OS-9 tárgy nevek:

Lemez fájlok:

<b>ISAx1</b>	ISaGRAF alkalmazás kód biztonsági másolat fájl
<b>ISAx6</b>	ISaGRAF alkalmazás szimbólum biztonsági másolat fájl

Memória modulok:

<b>ISAx0</b>	ISaGRAF kernel rendszer adatok
--------------	--------------------------------

<b>ISAx1</b>	ISaGRAF alkalmazás kód
<b>ISAx2</b>	ISaGRAF kernel valós idejű adatbázis
<b>ISAy3</b>	ISaGRAF kommunikációs adatcsere puffer
<b>ISAx4</b>	ISaGRAF on line 1-es módosítás alkalmazás kód
<b>ISAx5</b>	ISaGRAF on line 2-es módosítás alkalmazás kód
<b>ISAx6</b>	ISaGRAF alkalmazás szimbólum

A felhasználónak ezért ügyelnie kell arra, hogy ne használja ugyanazokat a tárgy neveket.

#### • Alkalmazás biztonsági másolat

Amikor egy új alkalmazás a workbench hibakeresőről a célra le van töltve, az alkalmazás kód a cél pillanatnyi alkönyvtárába kimentésre kerül, a következő fájlnevvvel:

**ISAx1** ISaGRAF alkalmazás kód biztonsági másolat fájl (ahol x a kiszolgáló szám)

Emellett, ha előzőleg le lett töltve az alkalmazás szimbólumtáblázat, akkor az szintén kimentésre kerül a cél pillanatnyi alkönyvtárába, az alábbi fájlnevvvel:

**ISAx6** ISaGRAF alkalmazás szimbólum biztonsági másolat fájl (ahol x a kiszolgáló szám)

Az ISaGRAF cél elindításakor ezek az alkalmazás kód és alkalmazás szimbólum fájlok megkeresésre kerülnek, és betöltődnek a memóriába, mint ugyanolyan nevű adatmodulok.

Ezután, ha nem áll rendelkezésre szimbólum fájl, akkor a cél elkezd futtatni az alkalmazás kódot, betöltött szimbólumok nélkül.

Ha nem áll rendelkezésre alkalmazás kód a memóriában, akkor a cél vár egy alkalmazás letöltésére.

A cél bekapcsoláskor egy megadott alkalmazással, a hibakereső kapcsolat használata nélkül történő elindítása érdekében:

- Egy első lehetőség ezeknek a fájloknak a közvetlen bemásolása lehet a cél pillanatnyi alkönyvtár lemezére a PC gazdáról, ahol a workbench helyezkedik el, valamilyen fájlátviteli eszköz használatával. Ezeknek a műveleteknek a megkönnyítésére a workbench eszköz menü használható (lásd a Programok kezelése c. Kezelési útmutatót).
- Egy másik lehetőség az alkalmazás kód (és ha szükséges, akkor az alkalmazás szimbólumtáblázat) saját eszközökkel történő tárolása egy nem illékony memóriában (pl. PROM vagy EPROM), a PC gazdán levő fájlokból, ahol a workbench telepítve van.

Ezután a rendszer bekapcsolásakor, amennyiben szükséges (pl. a gyorsabb hozzáférés vagy töréspont kezelés miatt), az alkalmazás kódot (és ha szükséges, akkor az alkalmazás szimbólumtáblázat) saját eszközökkel a PROM-ból RAM-ba lehet tölteni **ISAx1** (és ha szükséges, akkor **ISAx6**) memória adatmodulként.

#### FIGYELEM!

Az ISaGRAF hibakereső töréspont kezelése nem futtatható megfelelően, ha az alkalmazás kód modul nem férhető hozzá íráshoz. Ez nem jelent problémát, mivel az alkalmazás normál esetben előzőleg teljesen le lett tesztelve.

Ha az ISaGRAF workbench a PC gazdán a standard \ISAWIN alkönyvtárba van telepítve: akkor a MYPROJ projekt alkalmazás kód fájlja a következő:

\ISAWIN\APL\MYPROJ\appli.x6m (megfelel isax1-nak a célon).

a MYPROJ projekt alkalmazás szimbólum fájlja a következő:  
 \\SAWINAPL\MYPROJ\appli.tst (megfelel isax6-nak a célon).

### ☐ **Hibakezelés és output üzenetek**

Az ISaGRAF célszoftver magában foglalja a hibaérzékelés kezelését. A figyelmeztető hibaüzenetek listája és azok leírása a mellékletben található.

A hibaérzékelés feldolgozása az alábbiak szerint történik:

- Egy hiba egy hiba- és egy argumentum számból áll, amelyek az ISaGRAF hibarutinhoz vannak küldve
- Ha a workbench Létrehozás beállításainál be lett kapcsolva a hibaérzékelés jelző, akkor a hiba feldolgozásra kerül. Ha nem, akkor az információ elvész, és a hibakezelés befejeződik.

A feldolgozáskor:

- A hiba száma (decimális érték) és argumentuma (hexadecimális érték) megjelenítésre kerül az alapértelmezett stdout outputon
- A hibaszám és argumentum egy gyűrűs FIFO hibapufferbe tolik, későbbi előhívás céljából. A hibapuffer mérete a workbench Létrehozás beállításoknál van beállítva. Amikor a puffer megtelik, akkor minden új bejövő hibánál a legrégebbi hiba elvész.
- A hibák vagy a hibakeresőből, vagy a futó alkalmazásból hozhatók elő, a SYSTEM hívással (lásd a Kezelési útmutatóban).

Amikor a hibakereső egy hibát érzékel, akkor a hiba ablakban megjelenik egy üzenet, amely leírja a hibát. Az alkalmazás kontextusától függően (fut, vagy nem), a hibakereső megjelenítheti vagy a tárgy nevét (változó, vagy program) ahonnan a hiba származik, vagy a négyzetes zárójelek közé zárt [x] argumentumot (decimális érték), amelynek minden hibánál más jelentése van.

Az alapértelmezett stdout outputon egy üdvözlő üzenet és a hibaértékek jelennek meg, amikor a cél elindul és egy hiba van érzékelve. Ha a megjelenítés nem kívánatos a standard output csatormán, akkor egy átirányító parancs használható, mint pl. a következő:

```
prog_name [options] >>>/nil
```

### ☐ **Ciklus időtartam, feladatok viselkedései, és feladatok prioritásai**

- Egy ISaGRAF ciklus végén, közvetlenül egy új elindulása előtt, a következő algoritmus van elvégezve:

Ha egy ciklusidőzítés van meghatározva (a workbench-ről: lásd a programok kezelése c. Kezelési útmutatót), akkor a CPU a hátralevő időtartamra (megadott ciklusidő – pillanatnyi alkalmazás) el lesz engedve. Ha ez a maradék időtartam negatív, akkor egy túlszordulás lesz generálva, és a CPU 1 ütemre el lesz engedve, az időzítés kikényszerítésére.

Ha nincs megadva ciklusidőzítés, vagy ha a maradék idő 1 ütem vagy attól kevesebb illetve zéró, akkor a CPU 1 ütemre eleresztődik az időzítés kikényszerítésére.

A cél időzítési pontossága az OS-9 rendszer ütemének felel meg.

Általában egy meghatározott ciklusidőzítés van használva a ciklusok elindítására vagy a CPU-nak az OS-9 rendszeren futó más feladatokhoz történő átadására.

- A kommunikációs feladat alvó állapotban van mindaddig, amíg nincsenek bejövő adatok a kommunikációs kapcsolaton keresztül. Ha szükséges, akkor ez a feladat a kernel feladattal lefolytatott kérdés/felelet protokollon keresztül szerez be információt a futó alkalmazásról. A kommunikációs feladat feltesz egy kérdést a kernel felé. A ciklus végén (a szinkron alkalmazás kép érdekében) a kernel válaszol a kommunikációs feladatnak.

Az ISaGRAF feladatok nem módosítják a nekik adott prioritásokat. A felhasználó ezeket a prioritásokat szabadon megváltoztathatja a fent ismertetett ISaGRAF feladatok viselkedéseinek és az átfogó alkalmazás igényeknek megfelelően.

Pl. annak biztosítására, hogy az ISaGRAF-ot nem veszi el egy alacsony prioritású feladat, az OS-9 feladatkezelő paraméterek, pl. a **MIN\_AGE** és **MAX\_AGE**, módosíthatók.

## ☐ **Terminál mód**

A cél soros kommunikációs protokoll felismer egy 3 kocsi vissza karakterből (\$0D) álló sorozatot, és ekkor a soros kapcsolat eszközön, amennyiben az rendelkezésre áll, elkezd egy OS-9 shell feladatot.

Ez lehetővé teszi, hogy bármelyik terminál egy OS-9 shell promptot kapjon az ISaGRAF cél soros kapcsolat használatával.

### Példa:

A gazda PC-től:

- Zárja be az ISaGRAF hibakeresőt.
- Indítson el egy Windows Terminál munkafázist (tartozékok csoport), a megfelelő kommunikációs paraméterekkel.
- Nyomjon meg 3 kocsi vissza-t

Ekkor be van jelentkezve egy OS-9 Shell-be.

- A terminál módból való kilépéshez írja be: **logout**.

### FIGYELEM!

A terminál módú munkafázisból mindig tiszta módon, csakis a logout használatával kell kilépni, mert máskülönben a workbench-el való következő kapcsolat sikertelen lesz.

## C.5.Az ISaGRAF VxWorks céllal történő munkavégzés elkezdése

Az ISaGRAF cél(ok) futtatásához egy pár parancsot kell végrehajtani a VxWorks rendszeren, a kommunikációs környezet beállítása, valamint az ISaGRAF cél(ok) elszaporítása érdekében. Ezeknek a parancsoknak mindegyike egy szkript fájlból indítható el. Ezek a következő fejezetekben lesznek ismertetve.

### C.5.1.A rendszer forráskezelő: isassr.o

Erre a modulra az ISaGRAF cél bármely konfigurációjában mindig szükség van, és ennek elsőnek kell lenni az ISaGRAF cél betöltött moduljai között. Ez lehetővé teszi többszörös célt futtató rendszer erőforrás kezelését.

### C.5.2.Az isa.o, isakerse.o és isakeret.o közös jellemzői

Az ISaGRAF futtatásához ezeknek a moduloknak az egyike tölthető be.

isa.o: lehetővé teszi ISaGRAF egyfeladatos célok elindítását (csak soros kommunikációs kapcsolat).

Isakerse.o: lehetővé teszi ISaGRAF többfeladatos célok elindítását (csak soros kommunikációs kapcsolat).

Isakeret.o: lehetővé teszi ISaGRAF többfeladatos célok elindítását (soros és/vagy Ethernet kommunikációs kapcsolat).

Ezek a modulok a következő fejezetekben lesznek ismertetve.

#### ☐ **Soros kommunikációs kapcsolat konfigurálása**

Az ISaGRAF cél alapvetően egy soros kapcsolatot alkalmaz a hibakereső kommunikációhoz. A kinyitáskor semmilyen konfiguráció nincs végrehajtva az ISaGRAF cél által meghatározott soros kapcsolati eszközön. Ezért a felhasználó teljesen szabadon használhatja a szükséges paramétereket. Ugyanakkor azonban bináris átviteli mód (RAW mód) szükséges. Ennek megfelelően az *ISAMOD ()* szubrutin áll rendelkezésre.

uchar ISAMOD

```
(
  char *desc,      /* Soros eszköz neve */
  uint32 baudrate /* Adatátviteli sebesség */
)
```

Leírás:

A megadott soros kapcsolati eszközt egy megadott adatátviteli sebességű bináris adatátvitelhez konfigurálja.

visszatérési érték:

0 ha sikeres, BAD\_RET ha hiba történik

A workbench hibakereső használatakor ellenőrizni kell, hogy a workbench kommunikációs paraméterei (lásd a kezelési útmutató Programok kezelése című fejezetét) megfeleljenek a célnak.

### ☐ **Rendszeróra sebesség**

A CLKRATE globális változót (uint32) nem kell inicializálni a VxWorks rendszeróra sebességhez. Ilyen módon használható:

CLKRATE = sysClkRateGet ()

A CLKRATE alapértelmezett értéke 60Hz.

### **C.5.3. Az ISaGRAF egyszeres feladat futtatása: isa.o**

Az ISaGRAF cél futtatható egyetlen feladatként. Egy ilyen konfigurációban azonban a műveletek kritikus fontosságúak lehetnek. A jó teljesítmény eléréséhez pl. ajánlatos nem túlterhelni a kommunikációs kapcsolatot. A VxWorks multitaszking rendszeren ugyanazon a CPU-n futtathatók különböző egyfeladatos ISaGRAF célok, feltéve hogy kiszolgáló számuk és kommunikációs portjuk különböző.

Ez az egyfeladatos kivitelezés elsősorban gyenge hardverrel rendelkező platformokhoz lett tervezve, pl. olcsó kártyákkal illetve MS-DOS PC-kkel, illetve egy új platformom történő portoláskor egy első prototípus létrehozásánál. Éppen ezért a multitaszking ISaGRAF cél kivitelezést kell előnyben részesíteni.

Az ISaGRAF egyfeladatos cél nem akadályozza meg a háttérfolyamatok vagy a megszakítás által meghajtott rutinok futását.

### ☐ **A kiszolgáló(k) regisztrálása**

Egy ISaGRAF célt kiszolgáló száma jellemez. Ez 1 és 255 között lehet, kivéve a 13-as számot (\$0D). Ez a kiszolgáló szám van használva a kommunikációs kapcsolat protokolljában. Ennek célja az, hogy megkülönböztesse egymástól a kiszolgálókat, ha egynél több cél fut. Ezért tehát az ISaGRAF cél(ok) feladatának elindítása előtt az(oka)t regisztrálni kell. Ennek megfelelően az *isa\_register\_slave()* szubrutin áll rendelkezésre.

```
uchar isa_register_slave
(
    uchar slave /* * kiszolgáló szám */
)
```

#### Leírás:

Egy új kiszolgáló regisztrációt ad a többcélos kezelő rendszerhez.

#### visszatérési érték:

0 ha sikeres, BAD\_RET ha hiba történik

### ☐ **Alkalmazás biztonsági másolat fájl tárolóegység**

A TSK\_FUNIT globális változó (char \*) egy, az alkalmazás biztonsági másolatának útvonalát tartalmazó karaktersorhoz inicializálható. Az ISaGRAF cél egyszerűen az fopen, fread, fwrite, fclose standard fájlkezelő rutinokat használja az alkalmazás biztonsági másolat fájlhoz.

Az alapértelmezett érték egy üres karaktersor (""), annak specifikálására, hogy nincs tárolóegység.

#### Példa:

TSK\_FUNIT = "gazda neve:/C:/ISaGRAF/target/apl/"

Az alkalmazás fájl biztonsági másolat alkönyvtárként a *master\_name* PC-n levő C: gyökerében levő ISaGRAF\target\apl\ határozandó meg. Ügyeljen arra, hogy ne felejtse el

beírni a záró \ jelet, mert máskülönben a biztonsági másolat az ISaGRAF\target\ alkönyvtárban lesz elkészítve, apl előtagú fájlnevekkel.

Ez a változó szükség esetén minden szaporítás előtt, mindegyik célhoz különböző útvonal egységekre állítható be.

Az alkalmazás biztonsági másolat fájlokkal kapcsolatos bővebb információ az Alkalmazás biztonsági másolatáról szóló fejezet részletes jellemzőkről szóló részében található.

### == **Ciklusvezérlés vége**

A TSK\_NBTCKSCHED (uint 32) változó egy, az ütem késleltetését meghatározó értékre állítható be, amelyet az ISaGRAF cél a ciklus végén használ.

Az alapértelmezett érték 0 (ugyanolyan prioritású feladat időzítés).

Ez a változó szükség esetén minden szaporítás előtt, mindegyik célhoz különböző értékekre állítható be.

A pontos jellemzőkről a Ciklus időtartamról, feladat viselkedésekről, és feladat prioritásokról szóló fejezetben találhatók részletesebb információk.

### == **ISaGRAF cél szaporítás**

Ha be lett állítva a konfigurációs környezet, az utolsó lépés az ISaGRAF cél(ok) szaporítása: isa\_main.

```
uchar isa_main
(
    uchar slave,      /* kiszolgáló szám */
    char *com         /* Soros eszköz neve */
)
```

#### Leírás:

Elindít egy ISaGRAF célt.

#### visszatérési érték:

egy zérótól eltérő értéket ad vissza, ha hibák történtek.

A kiszolgáló szám ugyanaz, mint ami a kiszolgáló regisztrálásról szóló fejezetben került megtárgyalásra.

Egynél több cél is elindítható, feltéve hogy azok kiszolgáló száma és kommunikációs portjuk eltér egymástól.

A workbench hibakereső használatakor ellenőrizni kell, hogy a workbench kiszolgáló paraméterei (lásd a kezelési útmutató Programok kezelése című fejezetét) megfeleljenek egy létező célnak.

### == **Példa**

Ez a példa azt mutatja, hogyan kell elindítani egy ISaGRAF egyfeladatos célt 1-es kiszolgáló számmal és /tyCo/1 soros kapcsolati eszközzel.

A pillanatnyi gazda alkönyvtár az, ahová a cél van telepítve.

```
load isassr.o module
```

```
ld < RELS/isassr.o
```

```
load isa.o module
```

```
ld < CMDS/isa.o
```

soros kommunikáció konfigurálása  
**ISAMOD ("tyCo/1", 19200)**

Rendszeróra sebesség  
**CLKRATE = sysClkRateGet ()**

kiszolgáló regisztrálás  
**isa\_register\_slave (1)**

Fájlátroló egység (kihagyható, az alapértelmezés beállítása miatt)  
**TSK\_FUNIT = ""**

Ciklus vége vezérlés (kihagyható, az alapértelmezés beállítása miatt)  
**TSK\_NBTCKSCHED = 0**

ISaGRAF cél szaporítás  
**sp (isa\_main, 1, "tyCo/1")**

#### **C.5.4. Az ISaGRAF multitaszkok futtatása: isakerse.o és isakeret.o**

Az ISaGRAF cél kernel valamint a kommunikációs kapcsolat válaszüzenetének javítása érdekében a cél két feladatra van osztva, ami különválasztja a kommunikációs feladatot (kommunikációs feladat) az alkalmazás végrehajtásától (kernel feladat).

Az ilyen architektúra rugalmasabb. Ez lehetővé teszi a felhasználó számára egynél több, ugyanahhoz a kernelhez kapcsolt kommunikációs feladat futtatását, vagy egy kommunikációs feladattal maximum 4 kernel futtatását. Ez bizonyos integrációkat, pl. ugyanazon az alkalmazáson egy folyamat vizualizálási kapcsolatot és a workbench hibakereső kapcsolatot, illetve ugyanazon a fizikai porton keresztül maximum 4 különböző alkalmazás egyszeres kapcsolatát teszi lehetővé.

A kernel és kommunikációs feladatok függetlenek, és külön-külön szaporíthatók. Az egyetlen kikötés az, hogy a kernel feladato(ka)t kell először elindítani, hogy az inicializálja saját rendszerkörnyezetét, és így a kommunikációs feladat(ok) kapcsolni tudja(ák) azt.

Az ISaGRAF multitaszk (többfeladatos) cél nem akadályozza meg a háttér folyamatok vagy a megszakítás által meghajtott rutinok futását.

Két modul van javasolva, a kommunikációs hardver képességeitől függően:

- Kernel és soros kapcsolat: isakerse.o

Ez a modul a kernel feladat(ok) és soros kommunikációs feladat(ok) elindítását teszi lehetővé.

- Kernel és soros vagy/és Ethernet kapcsolat: isakeret.o

Ez a modul a kernel feladat(ok) és soros vagy/és Ethernet kommunikációs feladat(ok) elindítását teszi lehetővé.

Az ISaGRAF elindításának módja ugyanaz az isakerse.o és az isakeret.o moduloknál, azzal a kivétellel, hogy az isakeret.o-nál az ISaGRAF kommunikációs feladat(ok) indításánál az Ethernet kapcsolathoz kommunikációs eszköz név paraméterként meghatározható vagy egy soros eszköz neve, vagy egy port szám: tst\_main\_ex (lásd lentebb).

Az ISaGRAF-nál a VxWorks cél a szerver, és a hibakereső a kliens, amely a megadott port számra kapcsolódik.

### ☐ **A kernel(ek) regisztrálása**

Egy ISaGRAF kernelt kiszolgáló száma jellemez. Ez 1 és 255 között lehet, kivéve a 13-as számot (\$0D). Ez a kiszolgáló szám van használva a kommunikációs kapcsolat protokolljában és a kernelhez kapcsolt kommunikációs feladat(ok) által. Ennek célja az, hogy megkülönböztesse egymástól a kiszolgálókat, ha egynél több cél fut. Ezért tehát az ISaGRAF kernel(ek) feladatának elindítása előtt az(oka)t regisztrálni kell. Ennek megfelelően az *isa\_register\_slave()* szubrutin áll rendelkezésre.

```
uchar isa_register_slave
(
    uchar slave /* kiszolgáló szám */
)
```

#### Leírás:

Egy új kernel kiszolgáló regisztrációt ad a többcélos kezelő rendszerhez.

#### visszatérési érték:

0 ha sikeres, BAD\_RET ha hiba történik

### ☐ **A kommunikációs feladat(ok) regisztrálása**

Egy ISaGRAF kommunikációs feladatot annak logikai száma jellemez. Ez egynél több kommunikációs feladat egyszerre történő kezelésére szolgál. Ez 1 és 255 között lehet, és mindegyik kommunikációs feladathoz másnak kell lennie. Ezért tehát az ISaGRAF kommunikációs feladat(ok) elindítása előtt az(oka)t regisztrálni kell. Ennek megfelelően az *isa\_register\_com()* szubrutin áll rendelkezésre.

```
uchar isa_register_com
(
    uchar com_id /* com. Feladat azonosító */
)
```

#### Leírás:

Egy új kommunikációs feladat regisztrációt ad a többcélos kezelő rendszerhez.

#### visszatérési érték:

0 ha sikeres, BAD\_RET ha hiba történik

### ☐ **Alkalmazás biztonsági másolat fájl tárolóegység**

A TSK\_FUNIT globális változó (char \*) egy, az alkalmazás biztonsági másolatának útvonalát tartalmazó karaktorsorhoz inicializálható. Az ISaGRAF cél egyszerűen az fopen, fread, fwrite, fclose standard fájlkezelő rutinokat használja az alkalmazás biztonsági másolat fájlhoz. Az alapértelmezett érték egy üres karaktorsor (""), annak specifikálására, hogy nincs tárolóegység.

#### Példa:

```
TSK_FUNIT = "gazda neve:/C:/ISaGRAF/target/apl/"
```

Az alkalmazás fájl biztonsági másolat alkönyvtárként a *gazda\_név* PC-n levő C: gyökerében levő ISaGRAF\target\apl\ határozandó meg. Ügyeljen arra, hogy ne felejtse el beírni a záró \ jelet, mert máskülönben a biztonsági másolat az ISaGRAF\target\ alkönyvtárban lesz elkészítve, apl előtagú fájlnevekkel.

Ez a változó szükség esetén minden kernel szaporítás előtt, mindegyik elindítandó célhoz különböző útvonal egységekre állítható be.

Az alkalmazás biztonsági másolat fájlokkal kapcsolatos bővebb információ az Alkalmazás biztonsági másolatról szóló fejezet részletes jellemzőkről szóló részében található.

### == **Ciklusvezérlés vége**

A TSK\_NBTCKSCHED (uint 32) változó egy, az ütem késleltetését meghatározó értékre állítható be, amelyet az ISaGRAF cél a ciklus végén használ.

Az alapértelmezett érték 0 (ugyanolyan prioritású feladat időzítés).

Ez a változó szükség esetén mindegyik kernelhez, minden kernel szaporítás előtt különböző értékre állítható be.

A pontos jellemzőkről a Ciklus időtartamról, feladat viselkedésekről, és feladat prioritásokról szóló fejezetben találhatók részletesebb információk.

### == **ISaGRAF kernel szaporítás**

Ha be lett állítva a konfigurációs környezet, az egyik utolsó lépés az ISaGRAF kernel(ek) terjesztése: isa\_main.

```
uchar isa_main
(
    uchar slave,      /* kiszolgáló szám */
    char *com         /* NINCS HASZNÁLVA Egy üres karaktorsor elfogadható */
)
```

#### Leírás:

Elindít egy ISaGRAF kernel feladatot.

#### visszatérési érték:

egy zérótól eltérő értéket ad vissza, ha hibák történtek.

A kiszolgáló szám ugyanaz, mint ami a kiszolgáló regisztrálásról szóló fejezetben került megtárgyalásra.

Egynél több kernel is elindítható, feltéve hogy azok kiszolgáló száma eltér egymástól.

### == **ISaGRAF kommunikációs feladat szaporítás**

Ha be lett állítva a konfigurációs környezet, az egyik utolsó lépés az ISaGRAF kommunikációs feladat(ok) terjesztése: tst\_main\_ex.

```
uchar tst_main_ex
(
    char *com,        /* Kommunikációs eszköz neve */
    uchar *slave,     /* Egy 4 Byte-os mező helye, amely a kernel kiszolgálót adja meg a kapcsolathoz */
    uchar com_id      /* kommunikációs feladat azonosító */
)
```

Leírás:

Elindít egy ISaGRAF kommunikációs feladatot.

viSSzatérési érték:

egy zérótól eltérő értéket ad vissza, ha hibák történtek.

Ez a 4 Byte-os mező a kernel kiszolgáló(k) számát határozza meg, amelyhez a kommunikációs feladat kapcsolva van. Ha 4-nél kevesebb kernel kiszolgálóra van szükség, akkor a mezőket zéróval kell kitölteni. Ha elindult a feladat, erre a fájlra többé nincs szükség. A kommunikációs eszköz neve a kommunikációs kapcsolathoz használandó soros eszköznek felel meg.

Egynél több cél is elindítható, feltéve hogy azok feladat azonosítói eltérnek egymástól.

A workbench hibakereső használatakor ellenőrizni kell, hogy a workbench kommunikációs kapcsolat paraméterei (lásd a kezelési útmutató Programok kezelése című fejezetét) megfeleljenek egy létező célnak (kernel és kommunikációs feladatok).

 **Példa:**

Ez a példa azt mutatja, hogyan kezdjünk:

Egy ISaGRAF kernel feladatot, 1-es kiszolgáló számmal.

Egy 1-es számmal azonosított ISaGRAF kommunikációs feladatot, amely az 1-es kernel kiszolgálóhoz és a /tyCo/1 soros kapcsolati eszközhöz van kapcsolva, soros kapcsolattal.

Egy 2-es számmal azonosított ISaGRAF kommunikációs feladatot, amely az 1-es kernel kiszolgálóhoz és az 1100 port számhoz van kapcsolva, Ethernet kommunikációs kapcsolathoz.

A pillanatnyi gazda alkönyvtár az, ahová a cél van telepítve.

load isassr.o module

**ld < RELS/isassr.o**

load isakeret.o module (Betölthető az isakerse.o is, ha nincs szükség Ethernet kommunikációs kapcsolatra)

**ld < CMDS/isakeret.o**

soros kommunikáció konfigurálása

**ISAMOD ("/tyCo/1", 19200)**

Rendszeróra sebesség

**CLKRATE = sysClkRateGet ()**

kiszolgáló regisztrálás

**isa\_register\_slave (1)**

kommunikáció regisztrálás

**isa\_register\_com (1)**

**isa\_register\_com (2)**

Fájlátoló egység (kihagyható, az alapértelmezés beállítása miatt)

**TSK\_FUNIT = ""**

Ciklus vége vezérlés (kihagyható, az alapértelmezés beállítása miatt)

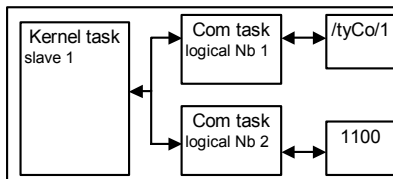
**TSK\_NBTCKSCHED = 0**

ISaGRAF kernel szaporítás  
**sp (isa\_main, 1, "")**

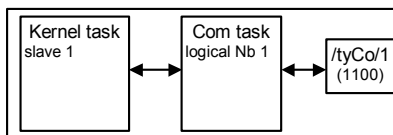
Kommunikációs feladat, kiszolgálók kapcsolat  
**SlavesLink = 0x01000000**

ISaGRAF kommunikációs feladat szaporítás  
**sp (tst\_main\_ex, "/tyCo/1", &SlavesLink, 1)**  
**sp (tst\_main\_ex, "1100", &SlavesLink, 2)**

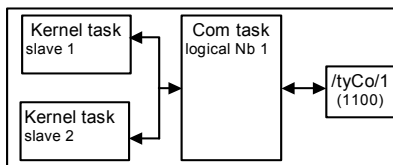
Ez a beindulás az alábbi ábrának felel meg



Az alábbi alapvető konfigurációk ugyancsak rendelkezésre állnak.



A legalapvetőbb konfiguráció egy soros (Ethernet) kapcsolaton egy kommunikációs feladattal asszociált kernel feladatból áll.

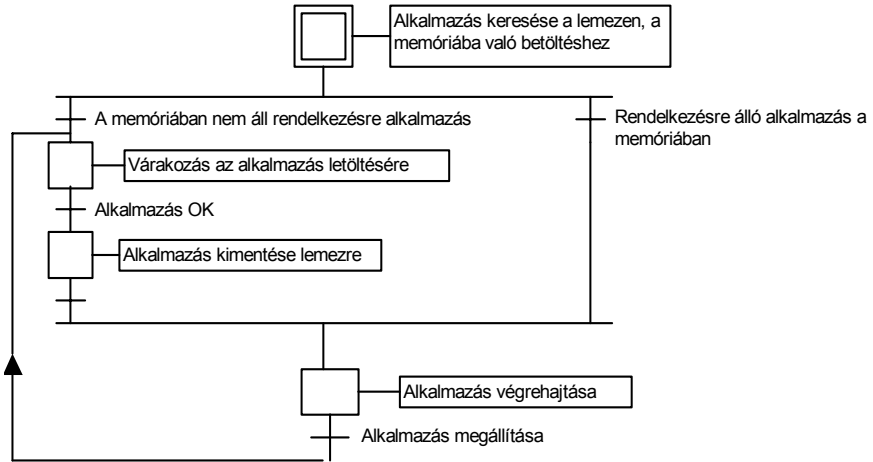


Egy másik konfiguráció egy soros (Ethernet) kapcsolaton egy kommunikációs feladattal asszociált 2 kernelből áll. Ebben az esetben a SlavesLink = 0x01020000.

### C.5.5. Különleges jellemzők:

#### ISaGRAF elindítás

A cél elindításakor a következő algoritmus kerül végrehajtásra.



### • Definíciók

Az alkalmazás kód a workbench által generált és letöltött, majd a célon végrehajtott bináris adatokat jelenti. Ezt a szimbólumtáblázatot egészítheti ki.

Az alkalmazási szimbólumtáblázatot egy ASCII adatbázis, amelyet a workbench generál és tölt le. Ez a táblázat kapcsolatot képez a szimbólum tárgyak és a belső céltárgyak között. Ez a célon nem szükséges, a felhasználó-specifikus szimbólumkezelés kivételével. A szimbólumtáblázattal kapcsolatos bővebb információk a Kezelési útmutatóban találhatók: Haladó programozási eljárások.

A lemez fájl egység útvonala az ISaGRAF cél induláskor van definiálva, a TSK\_FUNIT globális változó használatával (alapértelmezett érték = "", ami azt specifikálja, hogy nincs lemez fájl egység)

### • ISaGRAF multi-alkalmazások

Egy CPU-n egyidejűleg különböző alkalmazások(kernelek és kommunikációs feladatok) futhatnak, feltéve hogy azok mindegyikének eltérő kiszolgáló számaik és kommunikációs feladat logikai számaik vannak. Mindenesetre, különböző alkalmazások futtatásakor a felhasználónak ügyelnie kell bizonyos alkalmazás tárgyak, pl. I/O kártyák megosztott hozzáférésére. Pl. különböző alkalmazások (kernelek) adott fizikai kártyákat használhatnak, ha csak valamilyen fajta I/O szerver vagy szemafor nincs kivitelezve az I/O meghajtó révén.

### • Alkalmazás biztonsági másolat

Amikor egy új alkalmazás a workbench hibakeresőről a célra le van töltve, az alkalmazás kód a cél pillanatnyi alkönyvtárába kimentésre kerül (a cél az fopen, stb. standard fájlkezelő rutinokat használja), a következő fájl névvel:

**pathISAx1** ISaGRAF alkalmazás kód biztonsági másolat fájl (ahol x a kiszolgáló szám)

Emellett, ha előzőleg le lett töltve az alkalmazás szimbólumtáblázata, akkor az szintén kimentésre kerül a cél pillanatnyi alkönyvtárába, az alábbi fájl névvel:

**pathISAx6** ISaGRAF alkalmazás szimbólum biztonsági másolat fájl (ahol x a kiszolgáló szám)

Az *útvonal (path)* az ISaGRAF cél beindulásakor van megadva, a TSK\_FUNIT globális változó használatával. Egy üres karaktorsor ("" ) azt adja meg, hogy nincs lemez fájl egység (alapértelmezett érték).

Az ISaGRAF cél elindításakor ezek az alkalmazás kód és alkalmazás szimbólum fájlok megkeresésre kerülnek, és betöltődnek a memóriába.

Ezután, ha nem áll rendelkezésre szimbólum fájl, akkor a cél elkezd futtatni az alkalmazás kódot, betöltött szimbólumok nélkül.

Ha nem áll rendelkezésre alkalmazás kód a memóriában, akkor a cél vár egy alkalmazás letöltésére.

A cél bekapcsoláskor egy megadott alkalmazással, a hibakereső kapcsolat használata nélkül történő elindítása érdekében:

- Egy első lehetőség ezeknek a fájloknak a közvetlen bemásolása lehet az alkalmazás biztonsági másolat tároló egységére a PC gazdáról, ahol a workbench helyezkedik el, valamilyen fájlátviteli eszköz használatával. Ezeknek a műveleteknek a megkönnyítése a workbench „Eszközök” menüje használható (lásd a Programok kezelése c. Kezelési útmutatót).
- Egy másik lehetőség az alkalmazás kód (és ha szükséges, akkor az alkalmazás szimbólumtáblázat) saját eszközökkel történő tárolása egy nem illékony memóriában (pl. PROM vagy EPROM), a PC gazdán levő fájlokból, ahol a workbench telepítve van.

Ezután a rendszer bekapcsolásakor, amennyiben szükséges (pl. a gyorsabb hozzáférés vagy töréspont kezelés miatt), az alkalmazás kódot (és ha szükséges, akkor az alkalmazás szimbólumtáblázatot) saját eszközökkel a PROM-ból RAM-ba lehet tölteni.

Ezután, az ISaGRAF beindulásakor (közvetlenül a feladat szaporítás előtt) meg kell adnia az(oka)t a címe(ke)t, ahol az alkalmazás kód (valamint, ha szükséges, akkor az alkalmazás szimbólumtáblázat) a memóriában elhelyezkedik. Ilyen módon inicializálnia kell az SSR globális változót a következők szerint:

SSR[x][1].space = *alkalmazás kód cím helye*

És, ha szükséges:

SSR[x][6].space = *alkalmazás szimbólumtábla cím helye*

Ilyen módon leírható egy rövid procedura. Az SSR globális változó egy str\_ssr struktúra típusként van deklarálva, amely a tasy0ssr.h fájlban van definiálva.

#### **FIGYELEM!**

Az ISaGRAF hibakereső töréspont kezelése nem futtatható megfelelően, ha az alkalmazás kód modul nem férhető hozzá íráshoz. Ez nem jelent problémát, mivel az alkalmazás normál esetben előzőleg teljesen le lett tesztelve.

Ha az ISaGRAF workbench a PC gazdán a standard \ISAWIN alkönyvtárba van telepítve: akkor a MYPROJ projekt alkalmazás kód fájlja a következő:

\ISAWIN\APL\MYPROJ\appli.x6m (megfelel isax1-nak a célon).

a MYPROJ projekt alkalmazás szimbólum fájlja a következő:

\\SAWIN\APL\MYPROJ\appli.tst (megfelel isax6-nak a célon).

## ▬ Hibakezelés és output üzenetek

Az ISaGRAF célszoftver magában foglalja a hibaérzékelés kezelését. A figyelmeztető hibaüzenetek listája és azok leírása a mellékletben található.

A hibaérzékelés feldolgozása az alábbiak szerint történik:

- Egy hiba egy hiba- és egy argumentum számból áll, amelyek az ISaGRAF hibarutinhoz vannak küldve
- Ha a workbench Létrehozás beállításainál be lett kapcsolva a hibaérzékelés jelző, akkor a hiba feldolgozásra kerül. Ha nem, akkor az információ elvész, és a hibakezelés befejeződik.

A feldolgozáskor:

- A hiba száma (decimális érték) és argumentuma (hexadecimális érték) megjelenítésre kerül az alapértelmezett stdout outputon
- A hibaszám és argumentum egy gyűrűs FIFO hibapufferbe tolódik, későbbi előhívás céljából. A hibapuffer mérete a workbench Létrehozás beállításoknál van beállítva. Amikor a puffer megtelik, akkor minden új bejövő hibánál a legrégibbi hiba elvész.
- A hibák vagy a hibakeresőből, vagy a futó alkalmazásból hozhatók elő, a SYSTEM hívással (lásd a Kezelési útmutatóban).

Amikor a hibakereső egy hibát érzékel, akkor a hiba ablakban megjelenik egy üzenet, amely leírja a hibát. Az alkalmazás kontextusától függően (fut, vagy nem), a hibakereső megjelenítheti vagy a tárgy nevét (változó, vagy program) ahonnan a hiba származik, vagy a négyzetes zárójelek közé zárt [x] argumentumot (decimális érték), amelynek minden hibánál más jelentése van.

Egy hiba érzékelésekor a célon hibaértékek kerülnek megjelenítésre az alapértelmezett stdout outputon. Így a kijelzés VxWorks rutinok, pl. az alábbiak használatával irányítható

`ioGlobalStdSet()`

vagy `ioTaskStdSet()`

Az utóbbi esetben sem a kernel, sem a kommunikációs feladat nem tud hibákat generálni.

## ▬ Ciklusidőtartam, feladatok viselkedései, és feladatok prioritásai

- Egy ISaGRAF ciklus végén, közvetlenül egy új elindulása előtt, a következő algoritmus van elvégezve:

Ha egy ciklusidőzítés van meghatározva (a workbench-ről: lásd a programok kezelése c. Kezelési útmutatót), akkor a CPU a hátralevő időtartamra (megadott ciklusidő – pillanatnyi alkalmazás) el lesz engedve. Ha ez a maradék időtartam negatív, akkor egy túlszordulás lesz generálva, és a CPU el lesz engedve a TSK\_NBTCKSCHED-hez (az ISaGRAF beindulásakor beállított változó), az időzítés kikényszerítésére.

Ha nincs megadva ciklusidőzítés, vagy ha a maradék idő 1 ütem vagy attól kevesebb illetve zéró, akkor a CPU TSK\_NBTCKSCHED ütemre eleresztődik, az időzítés kikényszerítésére.

A cél időzírtési pontossága a VxWorks rendszer ütemének felel meg.

Általában egy meghatározott ciklusidőzítés van használva a ciklusok elindítására vagy a CPU-nak az VxWorks rendszeren futó más feladatokhoz történő átadására.

- A kommunikációs feladat alvó állapotban van mindaddig, amíg nincsenek bejövő adatok a kommunikációs kapcsolaton keresztül. Ha szükséges, akkor ez a feladat a kernel feladattal lefolytatott kérdés/felelet protokollon keresztül szerez be információt a futó alkalmazásról. A kommunikációs feladat feltesz egy kérdést a kernel felé. A ciklus végén (a szinkron alkalmazás kép érdekében) a kernel válaszol a kommunikációs feladatnak.

Az ISaGRAF feladatok nem módosítják a nekik adott prioritásokat. A felhasználó ezeket a prioritásokat szabadon megváltoztathatja a fent ismertetett ISaGRAF feladatok viselkedéseinek és az átfogó alkalmazás igényeknek megfelelően.

## C.6. Az ISaGRAF NT céllal történő munkavégzés elkezdése

### C.6.1. Az ISaGRAF futtatása

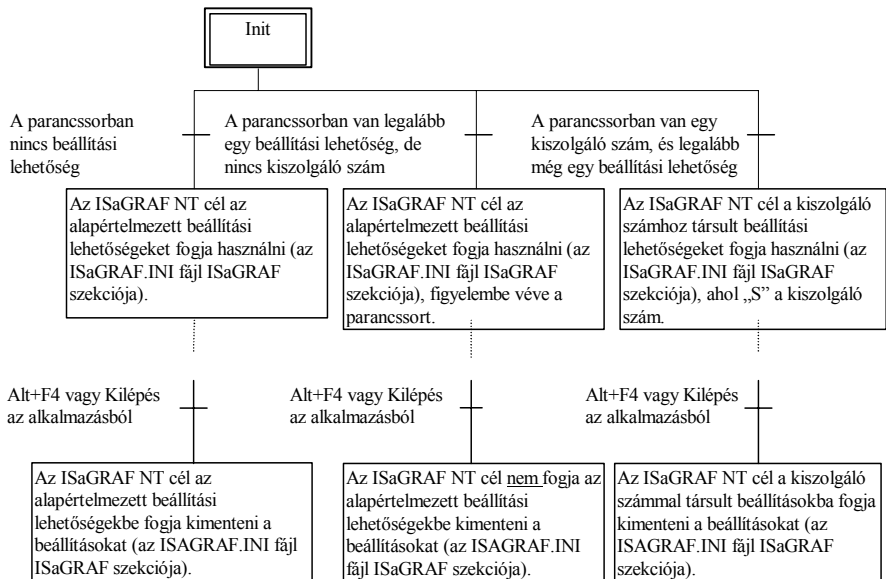
Az NT telepítésben a cél egyetlen programként fut: WISAKER.EXE, amely többször is elindítható. Ez lehetővé teszi tetszés szerinti számú ISaGRAF NT cél jelenlétét, mivel mindegyik előfordulásnak saját kiszolgáló száma van.

A cél program nem akadályozza meg a megszakítás által meghajtott rutinok futtatását.

A WISAKER szoftver a Windows NT 3.51-es, vagy azt követő verziói alatti futtatásra van tervezve.

### C.6.2. A beállítási lehetőségekről szóló általános tudnivalók

A beállítási lehetőségek az alábbi ábra szerint vannak kimentve és előhívva:



megjegyzendő, hogy az ISaGRAF.INI fájl a pillanatnyi munka-alkönyvtárban van kimentve.

### **Kiszolgáló szám: -s Beállítási lehetőség**

Ez a beállítás a cél kiszolgáló számát határozza meg. Ez 1 és 255 között lehet, kivéve a 13-as számot (\$0D). Ez a kiszolgáló szám van használva a kommunikációs kapcsolat protokolljában. Ennek elsődleges célja az, hogy megkülönböztesse egymástól a kiszolgálókat, ha egynél több cél van ugyanarra a gazda workbench-re kapcsolva, vagy ha egynél több cél fut ugyanazon a PC-n. A workbench hibakereső használatakor ellenőrizni kell, hogy a workbench kiszolgáló beállításai (lásd a kezelési útmutató Programok kezelése című fejezetét) megfeleljenek a cél beállításainak.

**Alapértelmezett érték:** Az alapértelmezett kiszolgáló szám 1, vagy az ISaGRAF.INI fájlban levő szám.

Példa:

WISAKER.EXE -s=2

Felhasználói interfész: Ez az ablak az ISaGRAF NT cél fő ablakából a Beállítások/Kiszolgáló” parancsra jelenik meg.



Ennek a beállításnak az értéke az egér vagy a nyíl billentyűk (Fel/Le) használatával módosítható. Ennek használatához az ISaGRAF NT célt újra kell indítani.

### **Kommunikációs kapcsolat és konfiguráció: -t Beállítási lehetőség**

Az ISaGRAF cél egy soros kapcsolatot vagy egy Ethernet kapcsolatot alkalmazhat a hibakereső kommunikációhoz.

A port számát a -t beállítás határozza meg. Mivel a kommunikációs interfész úgy lett tervezve, hogy bármilyen géppel kompatibilis legyen, ezért a COM1, COM2, COM3, vagy COM4 portokat lehet használni soros kommunikációra, és az 1100-tól kezdődő számú port számokat lehet használni Ethernet kommunikációra.

**Alapértelmezett érték:** Az alapértelmezett kommunikációs port 1100 az Ethernethez, illetve COM1 a soros kommunikációhoz, vagy az ISaGRAF.INI fájlban levő szám.

MEGJEGYZENDŐ: Az alapértelmezett kommunikációs kapcsolat az Ethernet.

Példák:

WISAKER -t=COM2

WISAKER -t=1101

**Soros konfiguráció:**

Bizonyos beállítási lehetőségek csak akkor használhatók, ha meg lett határozva a -t=COMx beállítás.

Az alábbiakban a soros kapcsolat konfigurációs beállítási lehetőségei szerepelnek:

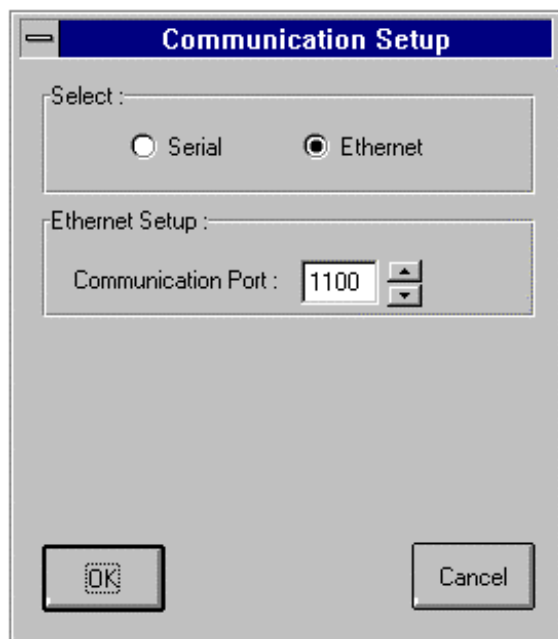
Beállítási lehetőség	Értékek	Jelentése
<b>baud</b>	600	Adatátviteli sebesség
	1200	
	2400	
	4800	
	9600	
	<b>19200</b>	
<b>paritás</b>	<b>n</b>	Nincs paritás
	<b>e</b>	Páros
	<b>o</b>	Páratlan
<b>adat</b>	7 vagy 8	Bitek száma
<b>stop</b>	1 vagy 2	A stop bit hossza
<b>áramlás</b>	<b>h</b>	Hardver vezérlés:
	<b>n</b>	Nincs vezérlés

Az alapértelmezett értékek: 19200, nincs paritás, 8 adat bit, 1 stop, nincs áramlás vezérlés.

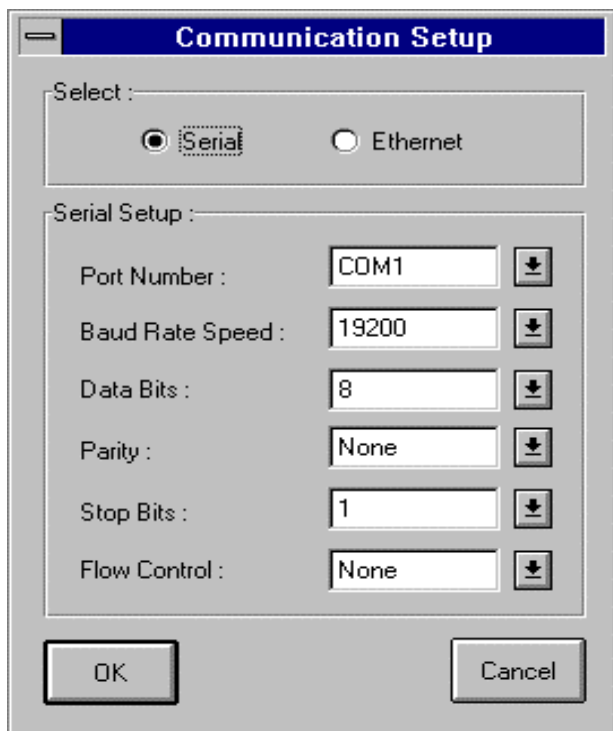
Példa:

WISAKER -t=COM1 baud=1200 data=8 parity=n stop=2

Felhasználói interfész: Ez az ablak az ISaGRAF NT cél fő ablakából a Beállítások/Kommunikáció” parancsra jelenik meg.



Vagy soros, vagy Ethernet kommunikáció választható ki. Az Ethernet kommunikáció lehetővé teszi a port számának megváltoztatását. Ennek a port számnak ugyanannak kell lennie, mint ami a workbench PC-PLC Kapcsolat specifikációjában van.



A soros kommunikáció kiválasztásakor megjelenik a konfiguráció. Ennek a konfigurációnak ugyanannak kell lennie, mint ami a workbench PC-PLC Kapcsolat specifikációjában van.

### ***Virtuális kártyák grafikus szimulációja: -x Beállítási lehetőség***

Ha ez a beállítás ki van választva, akkor az I/O csatlakozás szerkesztőben (lásd „A” rész) a virtuálisnak deklarált kártyák szimulálva lesznek.

A lehetséges értékek 0 vagy 1, ahol a 0 azt jelenti, hogy nincs szimuláció, az 1 pedig azt jelenti, hogy a szimuláció be van kapcsolva.

**Alapértelmezett érték:** Az alapértelmezett érték 0, vagy az ISaGRAF.INI fájlban levő szám.

#### Példa:

WISAKER -x=1 szimulálni fogja a virtuális kártyákat

Felhasználói interfész: A menüpont kijelölése illetve ki nem jelölése jelzi a beállítási lehetőség állapotát. A szimulált kártyák egy grafikus panelben jelennek meg.

### **Az ISaGRAF NT cél prioritása: -p Beállítási lehetőség**

Mivel a cél NT alatt fut, ezért nagyon hasznos egy prioritási szint meghatározása. Lehetséges pl., egy időkritikus ISaGRAF alkalmazás futtatása egy magasabb prioritású célon belül, miközben a háttérben egy vagy több cél fut alacsonyabb prioritásokkal.

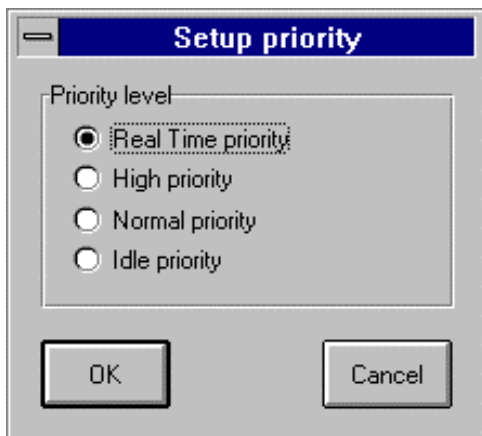
A lehetséges értékek: 0, 1, 2, vagy 3. 0 a legmagasabb prioritás, 3 pedig a legalacsonyabb prioritás.

#### Példák:

WISAKER -p=0

WISAKER -p=1

Felhasználói interfész: Ez az ablak az ISaGRAF NT cél fő ablakából a Beállítások/Prioritás” parancsra jelenik meg.



A legmagasabb prioritás a valós idő, a legalacsonyabb pedig az inaktív prioritás.

0: Valós idő prioritás

1: Magas prioritás

2: Normál prioritás

3: Inaktív prioritás

### **Példák:**

**wisaker -t=COM1**

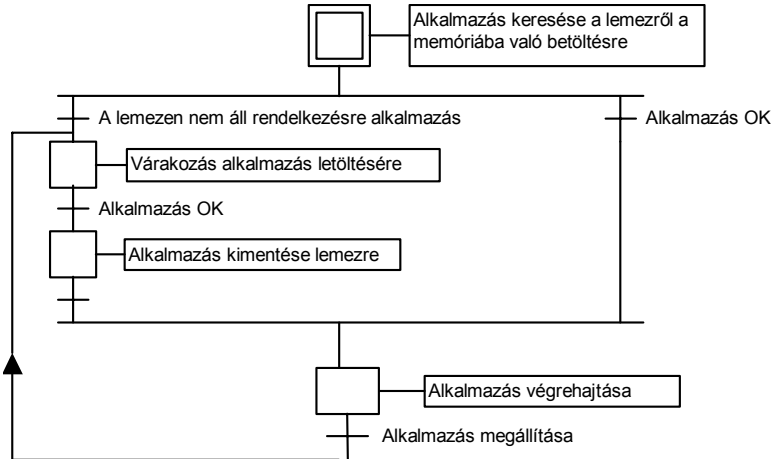
Az ISaGRAF célt az (1) kiindulási kiszolgáló számmal, és a COM1-el, mint kommunikációs porttal indítja el.

**wisaker -s=3 -t=COM1** Az ISaGRAF célt a 3 kiszolgáló számmal, és a COM1-erl, mint kommunikációs porttal indítja el.

### C.6.3. Különleges jellemzők:

#### ISaGRAF elindítás

A cél elindításakor a következő algoritmus kerül végrehajtásra.



#### • Definíciók

Az alkalmazás kód a workbench által generált és letöltött, majd a célon végrehajtott bináris adatokat jelenti. Ezt a szimbólumtáblázat egészítheti ki.

Az alkalmazási szimbólumtáblázat egy ASCII adatbázis, amelyet a workbench generál és tölt le. Ez a táblázat kapcsolatot képez a szimbólum tárgyak és a belső céltárgyak között. Ez nem szükséges a célon, kivéve felhasználó specifikus szimbólumok kezelését, mint pl. a DDE lehetőség, vagy I/O-k szimulációja szimbólumnév jellemzővel. A szimbólumtáblázattal kapcsolatos bővebb információk a Kezelési útmutatóban találhatók: Haladó programozási eljárások.

#### • ISaGRAF multi-alkalmazások

Egy CPU-n egyidejűleg különböző alkalmazások futhatnak, feltéve hogy azok mindegyikének eltérő kiszolgáló számaik és kommunikációs feladat logikai számaik vannak. Mindenesetre, különböző alkalmazások futtatásakor a felhasználónak ügyelnie kell bizonyos alkalmazás tárgyak, pl. I/O kártyák megosztott hozzáférésére. Pl. különböző alkalmazások adott fizikai kártyákat használhatnak, ha csak valamilyen fajta I/O szerver vagy semafor nincs kivitelezve az I/O meghajtó révén.

#### • Alkalmazás biztonsági másolat

Amikor egy új alkalmazás a workbench hibakeresőről a célra le van töltve, az alkalmazás kód a cél pillanatnyi alkönyvtárába kimentésre kerül, a következő fájlnevvvel:

**ISAx1** ISaGRAF alkalmazás kód biztonsági másolat fájl (ahol x a kiszolgáló szám)

Emellett, ha előzőleg le lett töltve az alkalmazás szimbólumtáblázat, akkor az szintén kimentésre kerül a cél pillanatnyi alkönyvtárába, az alábbi fájlnevvvel:

**ISAx6** ISaGRAF alkalmazás szimbólum biztonsági másolat fájl (ahol x a kiszolgáló szám)

Az ISaGRAF cél elindításakor ezek az alkalmazás kód és alkalmazás szimbólum fájlok megkeresésre kerülnek, és betöltődnek a memóriába.

Ha nem áll rendelkezésre szimbólum fájl, akkor a cél elkezd futtatni az alkalmazás kódot, betöltött szimbólumok nélkül.

Ha nem áll rendelkezésre alkalmazás kód, akkor a cél vár egy alkalmazás letöltésére.

Annak érdekében, hogy a cél a bekapcsoláskor egy adott alkalmazással, a hibakereső kapcsolat használata nélkül induljon el, ezek a fájlok közvetlenül a cél pillanatnyi alkönyvtárába másolhatók ugyanarról a lemeztől, ha a workbench ugyanazon a PC-n van, vagy egy floppy lemez használ.

Ha az ISaGRAF workbench a standard \ISAWIN alkönyvtárba van telepítve: akkor a MYPROJ projekt alkalmazás kód fájlja a következő:

\ISAWIN\APL\MYPROJ\appli.x8m

a MYPROJ projekt alkalmazás szimbólum fájlja a következő:

\ISAWIN\APL\MYPROJ\appli.tst

#### Példa:

Ha az alábbi parancsot írja be abból az alkönyvtárból, ahol a WISAKER.EXE van telepítve:

copy \ISAWIN\APL\MYPROJ\appli.x8m isa11

akkor a WISAKER.EXE megkeresi és végrehajtja a „myproj” alkalmazást.

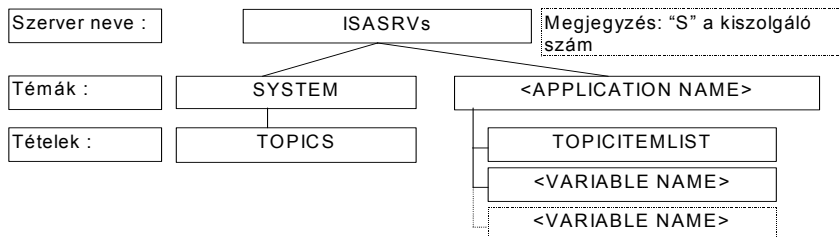
Mindezeket a parancsokat pl. egy csomagfájlba lehet csoportosítani, és azokat a workbench eszköz menüjéről el lehet indítani (lásd a programok kezelése c. Kezelési útmutatót).

### **DDE specifikáció**

Az ISaGRAF NT cél egy DDE (Dinamikus adatcsere; Dynamic Data Exchange) szerver. Bármely szoftver, amely kliens lehet, változók kicserélése céljából csatlakoztatható a célhoz. Az MSEXCEL pl. grafikákat animálhat, amelyeknek értékei az ISaGRAF célból a DDE-n keresztül érkeznek.

A DDE lehetőséghez a célon meg kell lennie az alkalmazás szimbólumtáblázatának.

A DDE tárgyak a következőképpen vannak definiálva:



« ISASRVs » a DDE szerver neve, 's' a kiszolgáló szám.

« SYSTEM » egy standard téma, amely hozzáférést biztosít a « TOPICS » tételhez,

« TOPICS » a pillanatnyilag definiált témák listáját adja meg: a rendszer és annak az alkalmazásnak a neve, amely jelenleg az ISaGRAF NT célban fut.

« APPLICATION NAME » az alkalmazás neve.

« TOPICITEMLIST » a pillanatnyi téma alatt rendelkezésre álló tételek listája, amely a DDE-n át elérhető változók listája.

« VARIABLE NAME » egy változó neve.

### DDE tájékoztató hurok sebesség az ISaGRAF NT célhoz: -d Beállítási lehetőség

A DDE kliens általában akkor kérdezi le a változókat, amikor csak szüksége van azokra. Ez sok változó esetén hosszú időt vehet igénybe. Létezik egy másik mód, amit tájékoztató módnak neveznek (tájékoztató hurok), amelyben a szerver maga csak módosított változókat küld. Így a kommunikáció minimálisra és így hatékonyra van csökkentve. Ebben a módban a szerver időnként megvizsgálja a tájékoztató változóknak nevezett változókat, hogy megtudja, melyiket kell elküldeni. Ezt az időtartamot DDE tájékoztató hurok sebességnek nevezik.

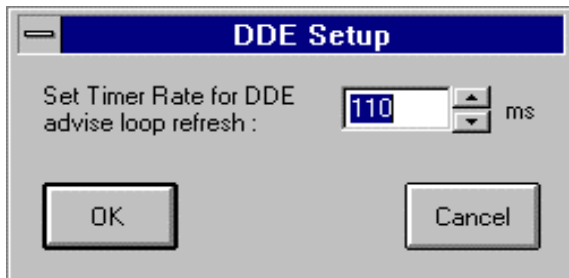
Ezzel a beállítással lehetőség van a DDE tájékoztató hurok sebességének beállítására (ms-ban).

**Alapértelmezett érték:** Az alapértelmezett érték 1000 ms, vagy az ISaGRAF.INI fájlban levő érték.

Példa:

WISAKER -d=100

Felhasználói interfész: Ez az ablak az ISaGRAF NT cél fő ablakából a Beállítások/DDE" parancsra jelenik meg.



### Hibakezelés és output üzenetek

Az ISaGRAF célszoftver magában foglalja a hibaérzékelés kezelését. A figyelmeztető hibaüzenetek listája és azok leírása a mellékletben található.

A hibaérzékelés feldolgozása az alábbiak szerint történik:

- Egy hiba egy hiba- és egy argumentum számból áll, amelyek az ISaGRAF hibarutinhoz vannak küldve
- Ha a workbench Létrehozás beállításainál be lett kapcsolva a hibaérzékelés jelző, akkor a hiba feldolgozásra kerül. Ha nem, akkor az információ elvész, és a hibakezelés befejeződik.

A feldolgozáskor:

- A hiba száma (decimális érték) és argumentuma (hexadecimális érték) megjelenítésre kerül az outputon (a WISAKER.EXE ablaka).
- A hibaszám és argumentum egy gyűrűs FIFO hibapufferbe tolódik, későbbi előhívás céljából. A hibapuffer mérete a workbench Létrehozás beállításoknál van beállítva. Amikor a puffer megtelik, akkor minden új bejövő hibánál a legrégebbi hiba elvész.
- A hibák vagy a hibakeresőből, vagy a futó alkalmazásból hozhatók elő, a SYSTEM hívással (lásd a Kezelési útmutatóban).

Amikor a hibakereső egy hibát érzékel, akkor a hiba ablakban megjelenik egy üzenet, amely leírja a hibát. Az alkalmazás kontextusától függően (fut, vagy nem), a hibakereső megjelenítheti vagy a tárgy nevét (változó, vagy program) ahonnan a hiba származik, vagy a négyzetes zárójelek közé zárt [x] argumentumot (decimális érték), amelynek minden hibánál más jelentése van.

A cél elindulásakor egy üdvözlő üzenet jelenik meg a kijelzőn. Ez egy kiszolgáló számból, a kommunikáció konfigurációból, és a DDE szerver nevéből áll.

## **Rendszer óra**

Mivel az ISaGRAF cél úgy lett tervezve hogy bármilyen rendszeren fusson, ezért a ciklus szinkronizáláshoz illetve az időváltozó frissítéshez használt időhivatkozás a standard időérték, ami 10 ezredmásodperc.

Ennélfogva tehát az időzítő változókon nem lehet 10 ms-nál jobb pontosságot elérni. Ugyanezen ok miatt, egy 10 ms, vagy attól alacsonyabb, és zérótól eltérő megadott ciklusidőtartam egy ciklusidőtartam túlszordulási hibát fog generálni (62-es hiba). További információkért lásd a következő fejezetet.

Amennyiben az Ön alkalmazása nagyobb pontosságot igényel, akkor kérjük, hogy egy speciális kivitelezés megoldása érdekében lépjen kapcsolatba a beszállítóval.

## **Ciklusidőtartam és cél viselkedés**

Egy ISaGRAF ciklus végén, közvetlenül egy új elindulása előtt, a következő algoritmus van elvégezve:

Ha egy ciklusidőzítés van meghatározva (a workbench-ről: lásd a programok kezelése c. Kezelési útmutatót), akkor a CPU a hátralevő időtartamra (megadott ciklusidő – pillanatnyi alkalmazás) el lesz engedve. Ha ez a maradék időtartam negatív, akkor egy túlszordulás lesz generálva, és a CPU 1 ütemre el lesz engedve, az időzítés kikényszerítésére.

Ha nincs megadva ciklusidőzítés, vagy ha a maradék idő 1 ütem vagy attól kevesebb illetve zéró, akkor a CPU 1 ütemre eleresztődik az időzítés kikényszerítésére.

A cél időzítési pontossága a Windows NT rendszer ütemének felel meg.

Általában egy meghatározott ciklusidőzítés van használva a ciklusok elindítására vagy a CPU-nak a Windows NT rendszeren futó más feladatokhoz történő átadására.

## Kilépés billentyű

Egy alkalmazás nem ipari körülmények között, egy asztali PC-n történő tesztelése során előfordulhat, hogy a felhasználó esetleg meg akarja állítani az ISaGRAF-ot: ez a váratlan megállások elkerülése érdekében egy gombnyomás-kombináció megnyomásával érhető el. Ez a gombnyomás-kombináció a következő:

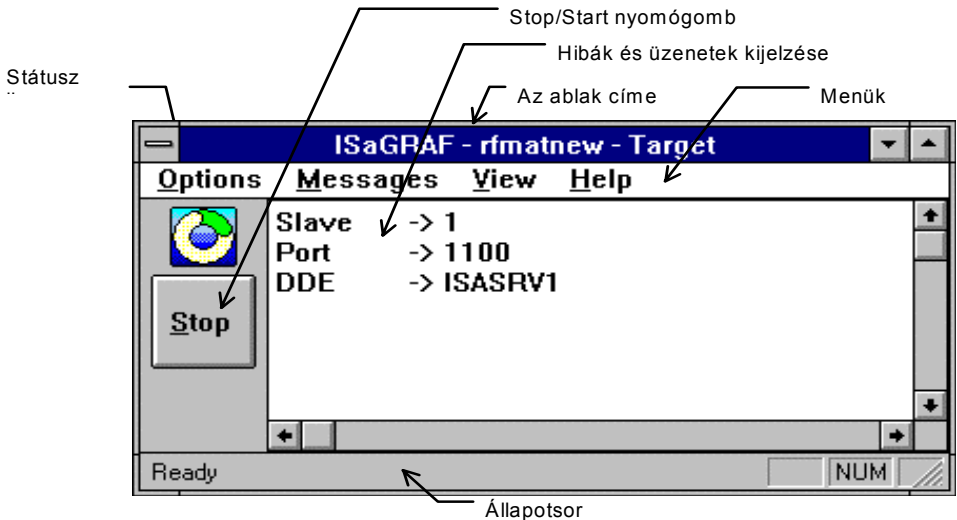
**alt + F4**

Ezeknek a gyors kilépéseknek egy veszélyes mellékhatása az, hogy az IO kártya interfész nincs bezárva. Így tehát, az ISaGRAF cél leállításának tiszta megoldása a következő:

- állítsa le az alkalmazást a hibakeresőből, vagy a Stop/Start nyomógommbal (ez lezárja az IO kártyákat)
- állítsa le az ISaGRAF célt a rendszer menüről

### C.6.4. Felhasználói interfész

Ez az ISaGRAF NT cél felhasználói interfésze:



Az alábbiakban a fő tételek vannak felsorolva  
 egy ablak cím  
 egy menüsor  
 egy futási státusz ikon  
 egy Start/Stop nyomógomb  
 egy hiba és üzenetek output  
 és egy állapotsor.

Az ablak címe a következőt tartalmazza: « ISaGRAF - name\_of\_appli - target », ahol a name\_of\_appli a futó alkalmazás neve. Ha nincs futó alkalmazás, akkor ez csak a következőt tartalmazza: « ISaGRAF - - Target ».

## **ISaGRAF NT target menüsor:**

A menüsor négy menüt tartalmaz:

Beállítási lehetőségek

Üzenetek

Nézet

Súgó

- **„Beállítások” menü**

(lásd az NT: Beállítási lehetőségekről szóló általános tudnivalók első fejezetét is)

A **„Beállítások”** menü a futási beállítások elérését teszi lehetővé. Ez a következő beállításokat ajánlja fel:

**Kiszolgáló** hozzáférést biztosít a kiszolgáló szám módosításához. A módosított beállítás csak a cél következő elindulásakor lép érvénybe. Ez a lehetőség nem áll rendelkezésre, ha a cél a parancssoron legalább egy beállítási lehetőséggel lett elindítva.

**Kommunikáció** hozzáférést biztosít a kommunikáció konfigurálásához. A módosított beállítás csak a cél következő elindulásakor lép érvénybe. Ez a lehetőség nem áll rendelkezésre, ha a cél a parancssoron legalább egy, az –s beállítástól eltérő beállítási lehetőséggel lett elindítva.

**DDE** hozzáférést biztosít a DDE tájékoztató hurok sebességének módosításához. A módosított beállítás csak a cél következő elindulásakor lép érvénybe. Ez a lehetőség nem áll rendelkezésre, ha a cél a parancssoron legalább egy, az –s beállítástól eltérő beállítási lehetőséggel lett elindítva.

**I/O szimulációja** kijelölése illetve ki nem jelölése jelzi a beállítási lehetőség állapotát. A módosított beállítás csak az alkalmazás következő Stop/Start váltásánál lép érvénybe.

**Prioritás** hozzáférést biztosít a prioritás módosításához. A módosított beállítás azonnal aktiválódik.

**Alapértelmezett beállítások** előhossa az alapértelmezett futási beállításokat, a következőkhöz:

- Kommunikáció

- DDE

- az ablak képernyő koordinátái

A módosított beállítás csak a cél következő elindulásakor lép érvénybe. Ez a lehetőség nem áll rendelkezésre, ha a cél a parancssoron legalább egy, az –s beállítástól eltérő beállítási lehetőséggel lett elindítva.

- **„Üzenetek” menü**

Az **„Üzenetek”** menü az output kezelésére szolgál. Ez a következő két parancsot tartalmazza:

**Nyugtázás** megszünteti a hibaüzenetknél a villogó piros jelzést.

**Törlés** teljesen kitörli a kijelzést.

**ISaGRAF NT cél ikon:**

Az ikon a cél állapotait mutatja:

- ha az alkalmazás fut, akkor az ikon forog
- ha nincs alkalmazás (vagy az alkalmazás leállt), akkor az ikon megáll
- hibák vagy üzenetek vannak a kijelző ablakban. Az ikon közepe pirosra változik. A villogás leállításához kiválasztható az « Üzenetek » menü « Nyugtázás » tétele, vagy ugyanannak a menünek a « Törlés » tétele (ügyelni kell arra, hogy ez a tétel teljesen kitörli a kijelző ablakot). A hibákkal kapcsolatos bővebb tudnivalók a Hibakezelés és az Output üzenetek c. Fejezetekben található.

A különböző állapotok az alábbi táblázatban vannak összefoglalva:

	nincs hiba	hibák, vagy üzenetek (a közepe piros)
futó alkalmazás		
nincs alkalmazás		

### ISaGRAF NT cél Start/Stop nyomógomb:

A Start/Stop nyomógomb teljesen azonos a hibakereső start/stop funkciójával. A nyomógombban levő szöveg az alkalmazás futási állapotát jelzi. Ha az alkalmazás fut, akkor a szöveg « Stop » lesz, ha az alkalmazás megállt (vagy ha nincs futó alkalmazás), a szöveg « Start » lesz (megjegyzendő, hogy ha nincs alkalmazás és a start művelet lett kérve, akkor a nyomógomb átvált a Stop módba, majd visszajön a Start módba).

### ISaGRAF NT target, általános információk:

A „Nézet / információ” parancsra a következő párbeszédablak általános tájékoztatást ad a cél konfigurációról és a futó alkalmazásról:

**ISaGRAF NT kernel : Global Information**

General Setup

Slave number:

Communication:

DDE Setup

Advise loop rate:  ms

Server name:

Topics (Items) names:

Application

Status:

Running mode:

Code size:  bytes

Data size:  bytes

Három tételek van felsorolva:

- a) Általános beállítások:
  - A kiszolgáló szám
  - a kommunikáció konfiguráció (Ha a kommunikációs kapcsolat Ethernet, akkor a port számon felül a pillanatnyi NT rendszeren rendelkezésre álló IP címek listája is megjelenik)
- b) DDE beállítás
  - a tájékoztató hurok sebessége
  - a DDE szerver neve
  - a DDE témák és tételek nevei. Ez általános információ, és nem valós értékeket jelez. valójában a < > jelek között levő értékeket a valós értékekkel kell behelyettesíteni.
- c) Alkalmazás
  - Az alkalmazás státusza, amely egy futó alkalmazás esetén annak neve, illetve „Nincs alkalmazás”, ha nincs futó alkalmazás.
  - Az alkalmazás futási módja, amely azt jelzi, hogy az alkalmazás a szoftver processzoron keresztül fut-e. Ez ebben az esetben a következő karaktersort tartalmazza: « Feldolgozott szoftver ». Vagy az alkalmazás egy C programfordítóval lett-e fordítva. Ez ebben az esetben a következő karaktersort tartalmazza: « C fordítású ». Ha nincs futó alkalmazás, akkor a következő karaktersort tartalmazza: « Nincs alkalmazás ».
  - A kód mérete byte-okban. Ha a futási mód « C fordításos », akkor ez a mező zéró.
  - Az adatok mérete byte-okban. Ez a futtatási belső adatok és a változó adatbázis összege.

### **Virtuális kártyák ISaGRAF NT cél szimulációja:**

Ha ki van választva az « I/O szimulációja » beállítás, akkor a következő alkalmazás elinduláskor az alábbi ablak fog megjelenni:

32bits Boards Simulator					
Options					
0:0	1:0	2:0	3:0	4:0	5:0
1 <input type="checkbox"/>	1 <input type="radio"/>	1 <input type="radio"/>	1 <input type="radio"/>	1 <input type="radio"/>	1 <input type="checkbox"/>
2 <input type="checkbox"/>	2 <input type="radio"/>	2 <input type="radio"/>	2 <input type="radio"/>	2 <input type="radio"/>	2 <input type="checkbox"/>
3 <input type="checkbox"/>	3 <input type="radio"/>	3 <input type="radio"/>	3 <input type="radio"/>	3 <input type="radio"/>	3 <input type="checkbox"/>
4 <input type="checkbox"/>	4 <input type="radio"/>	4 <input type="radio"/>	4 <input type="radio"/>	4 <input type="radio"/>	4 <input type="checkbox"/>
5 <input type="checkbox"/>	5 <input type="radio"/>	5 <input type="radio"/>	5 <input type="radio"/>	5 <input type="radio"/>	5 <input type="checkbox"/>
6 <input type="checkbox"/>	6 <input type="radio"/>	6 <input type="radio"/>	6 <input type="radio"/>	6 <input type="radio"/>	6 <input type="checkbox"/>
7 <input type="checkbox"/>	7 <input type="radio"/>	7 <input type="radio"/>	7 <input type="radio"/>	7 <input type="radio"/>	7 <input type="checkbox"/>
8 <input type="checkbox"/>	8 <input type="radio"/>	8 <input type="radio"/>	8 <input type="radio"/>	8 <input type="radio"/>	8 <input type="checkbox"/>

Az Ön I/O csatlakozás konfigurációjától függően több-kevesebb (és eltérő) kártya, illetve több-kevesebb (és eltérő) változó lesz feltüntetve. Az egyes kártyák tetején levő « s:b » számok a kártyahely azonosítót (s), illetve a kártya azonosítót (b) jelzik. A számlálás zérótól kezdődik, és az nem módosítható.

A „32bits Kártya szimulátor” ablak a Start/Stop alkalmazás állapottal működik. Tehát, ha van virtuális kártyával rendelkező (vagy szimulátorkártyákat alkalmazó) futó alkalmazás, és az « I/O szimuláció » jelző be van jelölve, akkor ez az ablak fog megjelenni. Ugyanakkor pedig a Stop nyomógomb megnyomására ez bezárul. Ez az ablak az I/O meghívásokkal együtt működik.

A „Beállítások” menü két tételt ajánl fel:

**Változó nevek** a változók neveit sorolja fel, de csak akkor, ha az ütem kód előtt le lett töltve a szimbólumtáblázat.

**Hexadecimális értékek** minden integert hexadecimális formában tüntet fel az alapértelmezett decimális forma helyett

A változónevek az alábbiak megfelelően néznek ki:

32bits Boards Simulator					
Options					
0:0	1:0	2:0	3:0	4:0	5:0
1 <input type="checkbox"/> ROW0	1 <input type="radio"/> LED00	1 <input type="radio"/> LED10	1 <input type="radio"/> LED20	1 <input type="radio"/> LED30	1 <input type="checkbox"/> COL0
2 <input type="checkbox"/> ROW1	2 <input type="radio"/> LED01	2 <input type="radio"/> LED11	2 <input type="radio"/> LED21	2 <input type="radio"/> LED31	2 <input type="checkbox"/> COL1
3 <input type="checkbox"/> ROW2	3 <input type="radio"/> LED02	3 <input type="radio"/> LED12	3 <input type="radio"/> LED22	3 <input type="radio"/> LED32	3 <input type="checkbox"/> COL2
4 <input type="checkbox"/> ROW3	4 <input type="radio"/> LED03	4 <input type="radio"/> LED13	4 <input type="radio"/> LED23	4 <input type="radio"/> LED33	4 <input type="checkbox"/> COL3
5 <input type="checkbox"/>	5 <input type="radio"/>	5 <input type="radio"/>	5 <input type="radio"/>	5 <input type="radio"/>	5 <input type="checkbox"/>
6 <input type="checkbox"/>	6 <input type="radio"/>	6 <input type="radio"/>	6 <input type="radio"/>	6 <input type="radio"/>	6 <input type="checkbox"/>
7 <input type="checkbox"/>	7 <input type="radio"/>	7 <input type="radio"/>	7 <input type="radio"/>	7 <input type="radio"/>	7 <input type="checkbox"/>
8 <input type="checkbox"/>	8 <input type="radio"/>	8 <input type="radio"/>	8 <input type="radio"/>	8 <input type="radio"/>	8 <input type="checkbox"/>

## C.7. „C” programozás

### C.7.1. Áttekintés

Ez a kézikönyv az ISaGRAF elveit valamint a Workbench eszközöket ismerő felhasználók számára készült. Tisztán automatizálási alkalmazásoknak a ICS Triplex ISaGRAF standard könyvtárakból **konverziós funkciók**, **„C” funkciók** és **funkcióblokkok** felhasználásával történő kifejlesztését követően lehetőség van „felhasználó definiálású” konverziós funkciók, „C” funkciók, és funkcióblokkok kidolgozására. Ez lehetővé teszi a felhasználó számára az ISaGRAF cél PLC kiemelését új könyvtárak létrehozásával, a munkaállomás flexibilitásának és hardver platformjának a maximális kiaknázása érdekében.

Egy „C” fejlesztő rendszerrel, valamint a „C” programozás bizonyos előzetes ismeretével, ez a kézikönyv lehetővé teszi a felhasználó számára ISaGRAF cél PLC-jének a lehető legjobb vezérléshez történő testre szabását. Az ilyen fejlesztések javítják a cél PLC teljesítményét, valamint az automatizálási programozó számára az ISaGRAF Workbench-el való fejlesztés kényelmét és minőségét.

Az ebben a dokumentumban szereplő információk nem csak egy adott célrendszerre vonatkoznak. Egyes jellemzők azonban (mint pl. a multitasking képességek) nem alkalmazhatók bizonyos egyfeladatos (monotasking) rendszereknél.

#### **Standard ISaGRAF workbench jellemzők**

Az ISaGRAF Workbench számos funkciót kínál a „C” komponens könyvtárak kezelésére az automatizálás fejlesztés területén. Az automatizálás programozásánál egy „C” konverzió, funkció, vagy funkcióblokk egy **„fekete doboz”**, amelyet teljes egészében interfésze definiál.

Az ISaGRAF Könyvtárrendező a komponenseknek a meglévő könyvtárakhoz adására, valamint a „C” kivitelezés interfészének definiálására, illetve ezeknek a komponenseknek az **ST/FBD** programozásban történő alkalmazására szolgál. Az ISaGRAF Könyvtárrendező emellett a „C” forráskód keretének automatikus generálását is biztosítja a konverziókhoz, funkciókhoz és funkcióblokkokhoz, és eszközöket tartalmaz az ilyen „C” forrásfájlok szerkesztésére. A Könyvtárrendező további tudnivalóival kapcsolatos további információk az **ISaGRAF kezelési útmutatóban** találhatók.

#### **„C” nyelv fejlesztés**

Az ISaGRAF Workbench nem tartalmaz semmiféle „C” programfordító vagy keresztfordító eszközt. A felhasználónak rendelkeznie kell egy, az ISaGRAF célrendszerhez utalt „C” programfordítóval ahhoz, hogy „C” komponenseit az ISaGRAF kernelbe integrálja.

Egy keresztfordító használata során az ISaGRAF Workbench felajánlja a felhasználói belépési pontokat egy felhasználó által definiált MS-DOS parancsfájl (.bat) DOS ablakban történő futtatására. A keresztfordítónak egy DOS emulációs ablakban kell futnia. Ha nem, akkor a fordítók és kapcsolatok tisztán MS-DOS kontextusban történő futtatása előtt a Windowst be kell zárni.

## **Műszaki megjegyzések**

Az ISaGRAF könyvtárrendező lehetővé teszi a felhasználó számára mindegyik könyvtár komponenshez egy szöveges leírás megírását. Ez a **műszaki megjegyzés** a felhasználó tájékoztatója a kifejlesztett „C” komponenshez, és az az automatizálás programozó előnyére szolgál, a megfelelő konverziók, funkciók, vagy funkcióblokkok ISaGRAF alkalmazásokban történő leírásához.

A konverziót, „C” funkciót vagy funkcióblokkot pontosan definiálni kell a műszaki megjegyzésben, hogy az automatizálás programozó ténylegesen úgy használhassa azt, mint egy csomagban levő ISaGRAF funkciót. Egy „C” funkciónál a műszaki megjegyzésnek a következőket kell leírnia:

- ☐ a funkció által feldolgozott részletes funkciót
- ☐ meghívó paramétereinek teljes leírását
- ☐ visszatérési értékek jelentését
- ☐ meghívó paramétereinek és visszatérési értékének részletes leírását
- ☐ az alkalmazás kontextusát

Egy „C” funkcióbloknál a műszaki megjegyzésnek a következőket kell leírnia:

- ☐ a blokkot aktiváló funkció által feldolgozott részletes funkciót
- ☐ meghívó paramétereinek teljes leírását
- ☐ visszatérési értékek jelentését
- ☐ meghívó és visszatérési paramétereinek a részletes leírását
- ☐ az alkalmazás kontextusát

Egy konverziós funkciónál a műszaki megjegyzésnek a következőket kell leírnia:

- ☐ a konverzió pontos jelentését, amikor az egy input változóval van használva
- ☐ a konverzió pontos jelentését, amikor az egy output változóval van használva
- ☐ azoknak az értékeknek a határait, amelyeket a konverzió fel tud dolgozni

A műszaki megjegyzések a következőkről is tartalmazhatnak információkat:

- ☐ a konverzió, funkció vagy funkcióblokk teljes beazonosítása
- ☐ karbantartásával és frissítéseivel kapcsolatos bármiféle információ
- ☐ a támogatott célrendszer
- ☐ a speciális multitaszkling jellemzők
- ☐ a szükséges rendszerszolgáltatások, memória, meghajtók, stb.

### **C.7.2. „C” konverziós funkciók**

Az ISaGRAF Workbench egy **lineáris konverziós** segédprogramot tartalmaz, egyszerű analóg I/O konverzióknak az ISaGRAF cél PLC-n futtatáskor történő elvégzésére. Ez a segédprogram nem igényel semmiféle „C” fejlesztést, mivel az szigorúan növekvő vagy csökkenő folyamatos funkciókra korlátozódik. Az ezekkel az eszközökkel kapcsolatos további információk az ISaGRAF kezelési útmutatóban találhatók.

A konverziós funkciók lehetővé teszik a felhasználó számára bármilyen komplex konverzió alkalmazását, a „C” nyelvben leírt specifikus műveletekkel. Egy konverziós funkció alapvetően

**mind az inputokhoz, mind az outputokhoz** definiálva van. Még ha az egyik irány nincs is használva, a konverzióknak az ISaGRAF kernelbe történő bevezetése előtt a kivitelezést és teszteket végre kell hajtani annak érdekében, hogy megakadályozható legyen egy rossz hívás miatt bekövetkező rendszerösszeomlás.

A konverziós funkciók a „C” nyelvben vannak megírva, majd azok le vannak fordítva, és integrálva vannak az ISaGRAF kernellel. Az új konverziós funkciók ISaGRAF projektokban történő alkalmazása előtt a megnövekedett kernelt telepíteni kell az ISaGRAF cél PLC-re. Az ISaGRAF Szimulátorban nem integrálhatók új konverziós funkciók. Az ISaGRAF alkalmazásokat a nem standard konverziós funkciók beillesztése **előtt** kell szimulálni.

A ICS Triplex ISaGRAF által írt standard konverziók „C” forráskódja az ISaGRAF Workbench-el együtt kerül telepítésre. ezek új funkciók létrehozásához példaként alkalmazhatók. Javasoljuk, hogy **ne módosítsák** a standard funkciókat, hogy azok bármelyik ISaGRAF alkalmazásban használhatók legyenek. Az ISaGRAF Workbench-el szállított standard konverziókat az ISaGRAF szimulátor támogatja.

**Figyelmeztetés:** A konverziós funkciók **szinkronizált** műveletek, amelyeket a futtatáskor az ISaGRAF I/O kezelő aktivál az alkalmazás ciklus input vagy output fázisai során. A konverziós funkció végrehajtására fordított idő az ISaGRAF alkalmazás **ciklusidőzítésébe** beleszámít. A felhasználónak ügyelnie kell arra, hogy egy konverziós funkcióba ne legyen beleprogramozva egy „várakozás művelet”, nehogy az ISaGRAF ciklus feldolgozása szükségtelenül meghosszabbodjon.

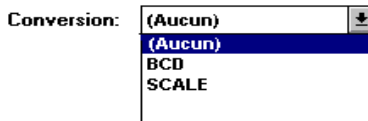
### ⇒ **Egy funkció hozzáadása az ISaGRAF könyvtárhoz**

Egy új konverziós funkciónak a Workbench oldalon az ISaGRAF könyvtárhoz adására az ISaGRAF Könyvtárrendezőt kell alkalmazni. A konverziós funkció könyvtár kiválasztásakor a „Fájlok” menü „Új” parancsát kell használni. A Workbench-en nem kell paramétert definiálni, mivel a konverziós funkciók egy standard, előre definiált interfészt használnak.

Amikor létre lett hozva egy új konverziós funkció, akkor meg kell írni annak **műszaki megjegyzését**. Az új konverziós funkció „C” forráskódjának keretét az ISaGRAF Könyvtárrendező automatikusan generálja.

### ⇒ **Egy konverzió alkalmazása egy ISaGRAF projektben**

Definiált konverziós funkciók a kiválasztott projekt bármelyik input vagy output analóg változó értékének a szűrésére használhatók. Egy konverziós funkciónak egy változóhoz csatolására a változó deklaráció szerkesztőt kell futtatni, ki kell választani egy input- vagy output változót, majd szerkeszteni kell annak paramétereit. Egy analóg I/O változóhoz csatolt konverziós funkció beállításához az analóg deklarációs ablak „**konverzió**” mezője szolgál:



A listában mind a konverziós funkciók, mind a táblázatok megjelennek. Ez azt jelenti, hogy ugyanaz a név nem használható egy funkcióhoz és egy táblázathoz. Egy változó nem

csatolható egy olyan konverziós funkcióhoz, amely még nincs definiálva illetve az ISaGRAF kernelbe integrálva.

### **Standard „C” interfész**

Egy konverziós funkció interfésze mindig ugyanolyan formátumú. A hívási és visszatérési paraméterek egy struktúrán mennek keresztül. Ezt a struktúrát a „**TACN0DEF.h**” fájl határozza meg:

```
/*
    Név: tacn0def.h
    Cél konverziók definíciós fájl
*/

#define DIR_INPUT 0          /* irány = input konverzió */
#define DIR_OUTPUT 1        /* irány = output konverzió */

typedef int32  T_ANA;        /* integer ANA típus */
typedef float  T_REAL;       /* valós ANA típus */

typedef struct {             /* konverziós struktúra */
    uint16 number;           /* konverziós szám (fenntartva) */
    uint16 direction;        /* konverziós irány */
    T_REAL *before;          /* konverzió előtti érték */
    T_REAL *after;           /* konverzió utáni érték */
} str_cnv;

#define ARG_BEFORE (*(arg->before))
#define ARG_AFTER  (*(arg->after))
#define DIRECTION (arg->direction)

/* eof */
```

A "**str\_cnv**" struktúra teljes egészében leírja az interfészt. Egy „C” konverziós funkció egyetlen paramétere egy ilyen struktúrára mutató index. A „**szám**” mező a konverziós funkció logikai száma (az ISaGRAF könyvtárban betöltött hely), és azt nem kell használni a programozásban.

Az „**irány**” **mező** azt jelzi, hogy a konverziót egy input változóra vagy egy output változóra kell-e alkalmazni. Ez a **DIR\_INPUT** értéket tartalmaz egy input konverzióhoz, illetve a **DIR\_OUTPUT** értéket egy output konverzióhoz.

Az „**előtte**” mező a konverzió előtti értékre mutat. Ennek a mezőnek egy input konverziónál más a jelentése, mint egy output konverziónál. Ez egy input konverzióhoz az elektromos értéket képviseli (az input eszközön olvasva), amikor az **irány** mező a **DIR\_INPUT** értéket

veszi fel. Ez egy output konverzióhoz a fizikai értéket képviseli (a programozott egyenletekben használva), amikor az **irány** mező a **DIR\_OUTPUT** értéket veszi fel.

Az „**utána**” mező a konverzió utáni értékre mutat. Ennek a mezőnek egy input konverziónál más a jelentése, mint egy output konverziónál. Ez egy input konverzióhoz a fizikai értéket képviseli (a programozott egyenletekben használva), amikor az **irány** mező a **DIR\_INPUT** értéket veszi fel. Ez egy output konverzióhoz az elektromos értéket képviseli (az input eszközön olvasva), amikor az **irány** mező a **DIR\_OUTPUT** értéket veszi fel.

A programozó a „C” konverziós funkcióhoz átadott struktúra **előtte** és **utána** mezőinek közvetlen eléréséhez az „**ARG\_BEFORE**” és „**ARG\_AFTER**” definíciókat használhatja. A feldolgozott értékek **egyszeres pontosságú lebegő értékek**. Az eredmény egy hosszú integerré van átalakítva, ha a konverzió egy integer analóg változón van elvégezve. Ez azt jelenti, hogy ugyanaz a konverzió használható mind valós, mind integer analóg I/O változókhoz.

## ▬ **Forráskód**

Mivel a konverziós funkció mind input, mind output analóg változókra alkalmazható, ezért a funkció „C” forráskódja két fő részre van osztva: az input konverzióra és az output konverzióra. Az interfész struktúra **irány** mezője szolgál az alkalmazandó konverzió kiválasztására. Az ISaGRAF Könyvtárrendező a konverziós funkció létrehozásakor automatikusan generálja a funkció komplett keretét. Ez tartalmazza a fő kiválasztó „**IF**” („**HA**”) struktúráját. Az alábbiakban egy konverziós funkció standard kerete látható:

```
/*
konverziós funkció
név: sample
*/

#include <tasy0def.h>
#include <tacn0def.h>

void CNV_sample (str_cnv *arg)
{
    if (DIRECTION == DIR_INPUT) { /*INPUT CONV*/

    }
    else { /*OUTPUT CONV*/

    }
}

/* A következő funkció az ISaGRAF I/O kezelővel levő kapcsolatot mutatja, a
konverzió nevének használatával. Ezt a funkciót teljes egészében az ISaGRAF
Könyvtárrendező generálta. */
```

```
UFP cnvdef_sample (char *name)
```

```
{  
    sys_strerror (name, "SAMPLE");    /* megadja a konverzió nevét */  
    return (CNV_sample);             /* a kivitelezési funkciót adja vissza */  
}
```

A funkció specifikus részének befejezésére a legjobb módszer két külön lokális funkció megírása az input konverzióhoz illetve az output konverzióhoz. Ezek a funkciók a fő algoritmus által lesznek meghívva, amint az az előző példában a fő **IF** struktúrában levő megjegyzésekben látható.

A „**TASY0DEF.H**” beletartozó fájl az ISaGRAF kernelből a rendszertől független definíciókhoz szükséges. Ez az **UFP** típus definícióját is tartalmazza, amely egy érvénytelenített funkcióra mutató indexet képvisel, és amely a deklarálási funkcióhoz használatos.

### **≡ A projektek és a „C” kivitelezés közötti kapcsolatok**

Egy konverziós funkció és a konverzióknak egy ISaGRAF projektben történő alkalmazása közötti logikai kapcsolat a konverzió nevével van létrehozva. A konverziós funkció „C” forráskódjához egy „deklarálás” funkció van adva. Ez a funkció csak egyszer van meghívva, amikor az alkalmazás elindul, és a kivitelezendő funkciónak megfelelő konverzió nevet jelzi az ISaGRAF I/O kezelője felé. Az alábbiakban egy ilyen deklarálás funkció standard formája látható:

```
UFP cnvdef_xxx (char *name)
```

```
{  
    strcpy (name, "XXX");             /* megadja a konverzió nevét */  
    return (CNV_xxx);                 /* a kivitelezési funkciót adja vissza */  
}  
/* xxx a konverzió neve */
```

A **strcpy** utasításhoz használt funkció nevét **nagybetűvel** kell írni. Ezt a konverzió kivitelezési funkció nevében és a deklarálás funkció nevében kisbetűvel kell írni.

A „**CNV\_**” és „**cnvdef\_**” előtagok használata a kivitelezési funkciónál és definíciós funkciónál lehetővé teszi a felhasználó számára egy konverzió elnevezését a „C” nyelv egyik fenntartott kulcsszavával, vagy a „C” ISaGRAF könyvtárak egyik létező funkciójának nevével.

A deklarálási funkcióhoz egyéb utasítások is adhatók az ezzel a konverzióval kapcsolatos bármiféle specifikus inicializálási művelet megvalósításához. Az ISaGRAF rendszer lehetővé teszi a felhasználó számára annak biztosítását, hogy ez a funkció **csak egyszer** legyen meghívva, az alkalmazás elindulásakor.

A deklarálási funkció bármilyen integrált konverziós funkcióhoz meg van hívva, még ha az nincs is az ISaGRAF alkalmazásban felhasználva. Az ISaGRAF kernel végzetes hibával megáll, ha egy, az alkalmazásban használt konverzió nincs integrálva a kernelbe.

Új funkcióknak a kernellel történő összekapcsolása előtt a felhasználónak egy másik, **„GRCNOLIB.C”** nevű „C” forrásfájlt kell írnia, és azt (a visszatartott konverziós funkciókkal együtt) a fájlok listájába kell illeszteni az összekapcsoló számára. A **„GRCNOLIB.C”** csak deklarációs funkciók tömbjét tartalmazza. Ez a tömb az alkalmazás inicializálások során van kiolvasva, egy dinamikus kapcsolat létrehozására a „C”-ben írt konverziós funkciókkal. Az alábbiakban egy ilyen fájl példája látható:

/\* "GRCN0LIB.c" fájl – Példa standard könyvtár konverziókkal \*/

```
#include <tasy0def.h> /* típus definícióhoz szükséges */
```

```
extern UFP cnvdef_scale (char *name);      /* dekl. funkció a SCALE konv.-hoz */
extern UFP cnvdef_bcd (char *name);       /* dekl. funkció a BCD konv.-hoz */
```

```
UFP_LIST CNVDEF[ ] = { /* deklaráció funkciók tömbje */
    /* integrált konverziós funkciókhoz */
    cnvdef_scale,
    cnvdef_bcd,
```

NULL };

(\* a fájl vége \*)

A **CNVDEF** tömböt egy NULL mutatóval kell lezárni. Ha ezek a feltételek nincsenek teljesítve, akkor bizonyos ütközések következhetnek be. Az új ISaGRAF kernel kapcsolása feloldatlan hivatkozásokat eredményez, ha a **CNVDEF** tömb nincs definiálva.

Ennek a fájlnak a megírásával egy új kernel építhető fel, amely az összes meglévő konverziót tartalmazza. Egy kernel egyetlen projekthez tesztre szabva is felépíthető, a **CNVDEF** tömbbe csak a projektben használt konverziókat beillesztve. Egy alkalmazás kódjának felépítésekor az ISaGRAF Kódgenerátor automatikusan generálja a **"GRCN0LIB.C"** fájlt. A fájl az ISaGRAF projekt alkönyvtárba van elhelyezve, és csak a projektben használt konverziókat csoportosítja.

## Korlátok

Az ISaGRAF könyvtár maximum **128** konverziós funkciót tartalmazhat. Egy konverziós funkcióban bármilyen típusú művelet feldolgozható. Megjegyzendő, hogy a funkciók az ISaGRAF ciklusban **szinkronizáltak** vannak meghívva, úgy hogy a funkció végrehajtása közvetlenül hat a ciklusidőzítésre.

### C.7.3. „C” funkciók

A „C” funkciók az **ST** és **FBD** nyelvek standard képességeinek fokozására használatosak. Ezek egy adott számításnak, rendszerhívásnak, kommunikációnak, vagy egy ISaGRAF alkalmazás és egyéb feladatok közötti párbeszédet szolgáló szolgáltatások készletének a

megvalósítására használatosak. A funkciók „C” nyelvben vannak megírva, majd azok le vannak fordítva, és integrálva vannak az ISaGRAF kernellel. Az új funkciók ISaGRAF projektekben történő alkalmazása előtt a megnövekedett kernelt telepíteni kell az ISaGRAF cél PLC-re.

Az ISaGRAF Szimulátorban nem integrálhatók új funkciók. Az ISaGRAF alkalmazásokat a nem standard funkciók beillesztése **előtt** kell szimulálni.

**Figyelmeztetés:** A funkciók **szinkronizált** műveletek, amelyeket a futtatáskor az ISaGRAF kernel aktivál az alkalmazás során. A funkció végrehajtására fordított idő az ISaGRAF alkalmazás **ciklusidőzítésébe** beleszámít. A felhasználónak ügyelnie kell arra, hogy egy funkcióba ne legyen beprogramozva egy „várakozás művelet”, nehogy az ISaGRAF ciklus feldolgozása szükségtelenül meghosszabbodjon.

### ⇒ **Egy funkció hozzáadása az ISaGRAF könyvtárhoz**

Egy új „C” funkciónak az ISaGRAF könyvtárhoz adására az ISaGRAF Könyvtárrendezőt kell alkalmazni a Workbench oldalon. A „C” funkció könyvtár kiválasztásakor a „**Fájlok**” menü „**Új**” parancsát kell használni. Amikor létre lett hozva egy új funkció, akkor meg kell írni annak **műszaki megjegyzését**. Az új funkció „C” forráskódjának keretét az ISaGRAF Könyvtárrendező automatikusan generálja.

A „**Szerkesztés**” menü „**Paraméterek**” parancsa az új funkció hívási és visszatérési paramétereinek a definiálására szolgál.

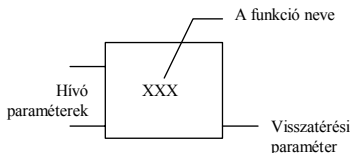
### ⇒ **Egy „C” funkció alkalmazása egy ISaGRAF projektben**

Egy ISaGRAF projekt programjaiban bármilyen integrált „C” funkció használható standard funkcióként. A „C” funkciók meghívhatók az **ST** és **FBD** nyelvekből, valamint az **SFC** nyelv speciális utasításaiából.

Egy „C” funkciónak az **ST** nyelvből történő meghívása a nyelv funkcióhívási konvencióit követi. A funkció meghívási paraméterei a funkció neve után vannak írva zárójelben, és azokat vessző választja el egymástól. A kifejezés a funkció által visszaadott értéket jelenti. Egy „C” funkció bármilyen kijelölési utasításba vagy komplex kifejezésbe beilleszthető. Az alábbiakban egy kijelölési utasításban levő „C” funkció meghívás példája látható:

**result := ProcName (par1, par2, ... parN);**

Egy **FBD** program bármilyen „C” funkciót meghívhat. Egy funkció standard funkciókeretként van használva. Annak paraméterei a keret bal oldalához csatlakoznak. A visszatérési paraméter a keret jobb oldalához csatlakoznak. Az alábbiakban egy ilyen funkciókeret standard formája látható:



Egy „C” funkció bármelyik **SFC** művelet keretből, vagy bármilyen, egy átmenethez csatolt logikai feltételben meghívható.

## ⇒ Egy „C” funkció interfészének definiálása

A **"Szerkesztés"** menü **"Paraméterek"** parancsa egy új funkció hívási és visszatérési paramétereinek a definiálására szolgál. Egy funkciónak maximum **31** meghívó paramétere, és mindig csak **egyetlen** visszatérési paramétere lehet.

Az ablak bal felső oldalán látható lista a „C” funkció paramétereit mutatja, az alábbi funkció meghívási prototípus szerinti sorrendben: először a meghívó paraméterek, utoljára pedig a visszatérési paraméter. Az ablak alsó része a listában pillanatnyilag kiválasztott paraméter részletes leírását mutatja:

- a paraméter neve
- a paraméter iránya (meghívás/visszatérés)
- a paraméter típusa

Egy paraméterhez bármelyik ISaGRAF adat típus használható. Logikai, Integer analóg, Valós analóg, Időzítő, vagy Üzenet. Az integer és valós analógokat meg kell különböztetni egymástól.

Az alábbiakban az ISaGRAF típusok és „C” típusok közötti összefüggés látható:

LOGIKAI	előjel nélküli hosszú	előjel nélküli 32 bites szó: 1=igaz / 0=hamis
ANALÓG	hosszú	előjel nélküli integer 32 bites szó
REAL	lebegő	egyszeres pontosságú lebegő érték
IDŐZÍTŐ	előjel nélküli hosszú	előjel nélküli integer 32 bites szó (az egység 1 ezredmásodperc)
ÜZENET	kar. *	karaktorsor

Amikor egy „C” funkcióhoz egy üzenet érték van küldve, az nem tartalmazhat nulla karaktereket. A „C” kódhoz küldött karaktorsor nulla lezárású. Ne feledje, hogy a visszatérés paraméternek kell az utolsónak lennie a listában. A paraméterek elnevezésénél az alábbi szabályokat kell betartani:

- a név hossza nem haladhatja meg a 16 karaktert
- az első karakternek betűnek kell lennie
- a többi karakternek betűnek, számjegynek, vagy aláhúzás karakternek kell lennie
- az elnevezés nem érzékeny a kisbetű/nagybetű beállításra

Ugyanaz a név nem használható a funkció egynél több paraméteréhez. Egy meghívási paraméternek nem lehet ugyanaz a neve, mint a visszatérési paraméternek. Ugyanaz a név **használható** különböző funkciók paramétereikhez. A visszatérési paraméter alapértelmezett neve „Q”. Ez a név szabadon módosítható. Egy paraméter neve a paraméter azonosítására szolgál a „C” forráskódban.

A **„Beillesztés”** parancs egy új paraméternek a kiválasztott paraméter elé történő beillesztésére szolgál. A **„Törlés”** parancs a kiválasztott paraméter törlésére szolgál. Az **„Elrendezés”** parancs automatikusan átrendezi (sorrendbe állítja) a paramétereket úgy, hogy a visszatérési paraméter a lista végére kerüljön. Az **„OK”** gomb megnyomása eltávolítja a funkció interfész definícióját, és bezárja a párbeszédablakot. A **„Mégsem”** gomb megnyomása a funkció interfész megváltoztatása nélkül bezárja a párbeszédablakot.

## ⇒ Funkció „C” interfész

Egy funkció interfésze paramétereinek definíciójától függ. A hívási és visszatérési paraméterek egy struktúrán mennek keresztül. Ez a struktúra a **"GRUS0nnn.H"** fájlban van definiálva, ahol az **"nnn"** a funkció logikai száma az ISaGRAF könyvtárban. Az alábbiakban a „C” interfész példája látható, a **„SIN”** funkcióhoz (szinusz számítás):

/\* Fájl: GRUS0255.h - "sample" funkció \*/

```
typedef long          T_BOO;
typedef long          T_ANA;
typedef float         T_REAL;
typedef long          T_TMR;
typedef char          *T_MSG;

typedef struct {
    /* MEGHÍVÁS */      T_REAL _param1;
    /* VISSZATÉRÉS */  T_REAL _param2;
} str_arg;

#define PARAM1        (arg->_param1)
#define PARAM2        (arg->_param2)
```

(\* a fájl vége \*)

Az alábbiakban az ISaGRAF típusok és a „C” típusok közötti összefüggés látható: Az ISaGRAF típusok a funkció definíciós fájljában „C” típusokként vannak definiálva.

logikai	T_BOO	hosszú (32 bit)
Integer analóg	T_ANA	hosszú
Valós analóg	T_REAL	lebegő (32 bit – egyszeres pontosság)
időzítő	T_TMR	hosszú
üzenet	T_MSG	char * (32 bit - kar. mutató)

A **"str\_arg"** struktúra minden mezője a funkció egy paraméterének felel meg. A visszatérési paraméter az utolsó a struktúrában. A meghívó paraméterek ugyanazzal a sorrenddel jelennek meg a struktúrában, mint ami a funkció definíciójához lett megállapítva. Egy nagybetűs azonosító van definiálva a struktúra egy paraméterhez való közvetlen hozzáférésre, amely a funkció „C” kivitelezéséhez van átadva. Az azonosítók nevei azok, amelyek a funkció definiálása során az ISaGRAF Könyvtárrendezővel lettek beadva.

A „C” definíciós fájl minden alkalommal frissítődik, amikor a funkció interfésze az ISaGRAF Könyvtárrendező használatával meg van változtatva. Ez teljes megfelelést biztosít a funkció kivitelezése és annak az ISaGRAF alkalmazások programjaiban történő használata között.

## ☐ **Forráskód**

Az alábbiakban egy „C” funkció standard kerete látható:

*/\* Felhasználói funkció példája - A szám "255" – A Név "SAMPLE" \*/*

```
#include "tasy0def.h"           /* ISaGRAF kernel közös definíciók */
#include "grus0255.h"           /* interfész definíció a 255-ös funkcióhoz */
```

```
void USP_sample (str_arg *arg)
{
    (* a funkció fő része *)
}
```

*/\* A következő funkció a funkció inicializálására és kivitelezésének deklarálására használatos. Ez az ISaGRAF I/O kernellel való kapcsolatot valósítja meg, a funkció nevének használatával. Ezt a funkciót teljes egészében az ISaGRAF Könyvtárrendező generálta. \*/*

```
UFP uspdef_sample (char *name)
{
    strcpy (name, "SAMPLE"); /* megadja a funkció nevét */
    return (USP_sample);     /* a kivitelezési funkciót adja vissza */
}
```

*(\* a fájl vége \*)*

A „TASY0DEF.H” beletartozó fájl az ISaGRAF kernelből a rendszertől független definíciókhoz szükséges. Ez az **UFP** típus definícióját is tartalmazza, amely egy érvénytelenített funkcióra mutató indexet képvisel, és amely a deklarálási funkcióhoz használatos.

## ☐ **A projektek és a „C” kivitelezés közötti kapcsolatok**

Egy „C” funkció kivitelezése és annak egy ISaGRAF projekt programjaiban történő alkalmazása közötti logikai kapcsolat a funkció nevével van létrehozva. A funkció „C” forráskódjához egy „deklarálás” funkció van adva. Ez a funkció csak egyszer van meghívva, amikor az alkalmazás elindul, és a kivitelezett funkciónak megfelelő „C” funkció nevet jelzi az ISaGRAF kernel felé. Az alábbiakban egy ilyen deklarálás funkció standard formája látható:

```
UFP uspdef_xxx (char *name)
{
    strcpy (name, "XXX"); /* megadja a funkció nevét */
    return (USP_xxx);     /* a kivitelezési funkciót adja vissza */
}
/* xxx a funkció neve */
```

A **strcpy** utasításhoz használt „C” funkció nevét **nagybetűvel** kell írni. Ezt a kivitelezési funkció nevében és a deklarálás funkció nevében kisbetűvel kell írni. Az „**USP\_**” és „**uspdef\_**” előtagok használata a kivitelezési funkciónál és definíciós funkciónál lehetővé teszi a felhasználó számára egy funkció elnevezését a „C” nyelv egyik fenntartott kulcsszavával, vagy a „C” ISaGRAF könyvtárak egyik létező funkciójának nevével.

A deklarálási funkcióhoz egyéb utasítások is adhatók, az ezzel a funkcióval kapcsolatos bármiféle specifikus inicializálási művelet létrehozásához. Az ISaGRAF rendszer lehetővé teszi a felhasználó számára annak biztosítását, hogy ez a funkció **csak egyszer** legyen meghívva, az alkalmazás elindulásakor. A deklarálási funkció bármilyen integrált „C” funkcióhoz meg van hívva, még ha az nincs is az ISaGRAF alkalmazás programjaiban felhasználva. Az ISaGRAF kernel végzetes hibával megáll, ha egy, az alkalmazásban használt „C” funkció nincs integrálva a kernelbe.

Új funkcióknak a kernellel történő összekapcsolása előtt a felhasználónak egy másik, „**GRUS0LIB.C**” nevű „C” forrásfájl kell írnia, és azt (a visszatartott funkciókkal együtt) a fájlok listájába kell illesztenie az összekapcsoló számára. A „**GRUS0LIB.C**” csak deklarációs funkciók tömbjét tartalmazza. Ez a tömb az alkalmazás inicializálások során van kiolvasva, egy dinamikus kapcsolat létrehozására a „C”-ben írt funkciókkal. Az alábbiakban egy ilyen fájl példája látható:

/\* "GRUS0LIB.c" fájl – Példa trigonometrikus funkciók használatára \*/

```
#include <tasy0def.h>                                /* típus definícióhoz szükséges */
```

```
extern UFP uspdef_fc1 (char *name);                  /* deklaráció funkciók */
extern UFP uspdef_fc2 (char *name);
extern UFP uspdef_fc3 (char *name);
extern UFP uspdef_fc4 (char *name);
```

```
UFP_LIST USPDEF[ ] = {                               /* deklaráció funkciók tömbje */
    /* integrált funkciókhoz */
    uspdef_fc1,
    uspdef_fc2,
    uspdef_fc3,
    uspdef_fc4,
```

```
NULL };
```

(\* a fájl vége \*)

Az **USPDEF** tömböt egy NULL mutatóval kell lezárni. Ha ezek a feltételek nincsenek teljesítve, akkor bizonyos ütközések következhetnek be. Az új ISaGRAF kernel kapcsolása feloldatlan hivatkozásokat eredményez, ha az **USPDEF** tömb nincs definiálva. Ennek a fájlnek a megírásával egy új kernel építhető fel, amely az összes meglévő funkciót tartalmazza. Egy

kernel egyetlen projekthez tesztre szabva is felépíthető, a **USPDEF** tömbbe csak a projektben használt funkciókat beillesztve. Egy alkalmazás kódjának felépítésekor az ISaGRAF Kódgenerátor automatikusan generálja a "**GRUS0LIB.C**" fájlt. A fájl az ISaGRAF projekt könyvtárba van elhelyezve, és csak a projektben használt funkciókat csoportosítja.

## ☐ **Korlátok**

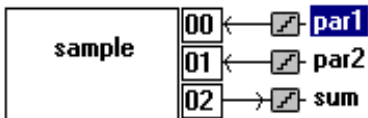
Az ISaGRAF könyvtár maximum **255** „C” funkciót tartalmazhat. Egy funkcióban bármilyen típusú művelet feldolgozható. Megjegyzendő, hogy a funkciók az ISaGRAF ciklusban **szinkronizáltan** vannak meghívva, úgy hogy a funkció végrehajtása közvetlenül hat a ciklusidőzítésre.

## ☐ **Teljes példa:**

Az alábbiakban egy „**minta**” funkció teljes programozása látható, amely csak egy összeadást hajt végre: Az alábbiakban a funkció műszaki megjegyzése látható:

név:	MINTA
leírás:	csak egy integer analóg összeadást hajt végre
létrehozás dátuma:	1992. július 1.-je
szerző:	ICS Triplex ISaGRAF
meghívás:	par1, par2: integer operandusok
visszatérés:	integer sum
prototípus:	sum := sample (par1, par2);

Az alábbiakban a funkció interfésze látható:



Az alábbiakban a funkció „C” forrás fejsora látható:

```
/* Fájl: GRUS0255.h - felhasználó C funkció definíciók - Név: minta */
```

```
/* standard ISaGRAF adattípusok definíciója */
```

```
typedef long T_BOO;
typedef long T_ANA;
typedef float T_REAL;
typedef long T_TMR;
typedef char *T_MSG;
```

/\* a meghívó és visszatérő paraméter struktúra definíciója \*/

```
typedef struct {  
    T_ANA _par1;          /* #1 meghívó paraméter */  
    T_ANA _par2;          /* #2 meghívó paraméter */  
    T_ANA _sum;           /* visszatérés paraméter */  
} str_arg;
```

/\* a meghívó és visszatérési paraméterek eléréséhez használt azonosítók \*/

```
#define PAR1              (arg->_par1)  
#define PAR2              (arg->_par2)  
#define SUM                (arg->_sum)
```

(\* a fájl vége \*)

Az alábbiakban a funkció „C” forráskódja látható: A C programozó csak a kiemelt karakterekkel nyomtatott sorokat írta be manuálisan.

/\* Fájl: GRUS0255.c - felhasználó C funkció - Név: MINTA \*/

```
#include "tasy0def.h"      /* típus definícióhoz szükséges */  
#include "grus0255.h"      /* C funkció forrás fejsor */
```

/\* C fő szolgáltatás: kiszámítja az összeadást \*/

```
void USP_sample (str_arg *arg)  
{  
    SUM = PAR1 + PAR2;  
}
```

/\* deklarációs szolgáltatás szükséges az ISaGRAF kernellel való dinamikus kapcsolathoz \*/

```
UFP uspdef_sample (char *name)  
{  
    strcpy (name, "SAMPLE");  
    return (USP_sample);  
}  
(* a fájl vége *)
```

### C.7.4. „C” FUNKCIÓBLOKKOK

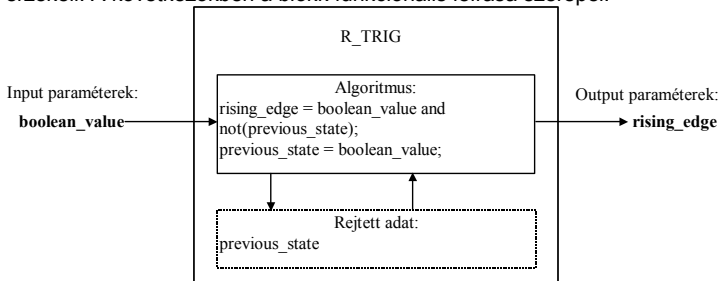
A „C” funkcióblokkok műveleteket és statikus adatokat társítanak. Ezek a „C” funkciók készletét egészítik ki, statikus tárgyak feldolgozását lehetővé téve. Ezek általában az **ST** és **FBD** nyelvek standard képességeinek fokozására használatosak. A funkcióktól eltérően, amelyek értékeket dolgoznak fel, a funkcióblokkok statikus adatokat tudnak feldolgozni. Ez azt jelenti, hogy egy funkcióblokk algoritmus kezelni tudja adatok idővel történő változásait.

A funkcióblokkok „C” nyelvben vannak írva, majd azok az ISaGRAF kernellel vannak lefordítva, és integrálva. Az új funkcióblokkok ISaGRAF projektekben történő alkalmazása előtt a megnövekedett kernelt telepíteni kell az ISaGRAF cél PLC-re. Az ISaGRAF Szimulátorban nem integrálhatók új funkcióblokkok. Az ISaGRAF alkalmazásokat a nem standard funkciók beillesztése **előtt** kell szimulálni.

**Figyelmeztetés:** A funkcióblokk meghívások **szinkronizált** műveletek, amelyeket a futtatáskor az ISaGRAF kernel aktivál az alkalmazás ciklus során. A funkcióblokk végrehajtására fordított idő az ISaGRAF alkalmazás **ciklusidőzítésébe** beleszámít. A felhasználónak ügyelnie kell arra, hogy egy funkcióblokkba ne legyen beprogramozva egy „várakozás művelet”, nehogy az ISaGRAF ciklus feldolgozása szükségtelenül meghosszabbodjon.

#### ≡ **Funkcióblokk előfordulások deklarálása**

Egy funkcióblokk egy olyan tárgy, amely műveleteket és sztatikus adatokat kombinál. Az alábbiakban az **„R\_TRIG”** funkcióblokk példája látható, amely egy logikai kifejezés felfutó élét érzékeli. A következőkben a blokk funkcionális leírása szerepel:



A **“previous\_state”** („előző állapot”) rejtett sztatikus változó az él kiszámításához szükséges. Ennek a változónak a **“TRIG”** funkcióblokk alkalmazásban történő mindegyik használatához másnak kell lennie. Az ST nyelvben használt funkcióblokkok előfordulásait a szótárban deklarálni kell. Mivel egy funkcióbloknak belső „rejtett” adatai vannak, ezért egy funkcióblokk minden példányát (előfordulását) egy egyedi névvel azonosítani kell. A blokk típusának elnevezése a könyvtárrendező használatával történik. Az előfordulások elnevezése a szótárszerkesztővel történik.

Az FBD nyelvben használt funkcióblokkokat nem kell deklarálni, mert az ISaGRAF FBD szerkesztő automatikusan deklarálja a használt blokkok előfordulásait. Az FBD szerkesztő által automatikusan deklarált funkcióblokk előfordulások mindig **LOKÁLISAK** a szerkesztett programhoz.

## ⇒ Egy funkcióblokk hozzáadása az ISaGRAF könyvtárhoz

Egy új „C” funkcióblokknak az ISaGRAF könyvtárhoz adására az ISaGRAF Könyvtárrendezőt kell alkalmazni a Workbench-ben. A „C” funkcióblokk könyvtár kiválasztásakor a „Fájlok” menü „Új” parancsát kell használni. Amikor egy új funkcióblokk lett létrehozva, akkor meg kell írni annak **műszaki megjegyzését**. Az új funkcióblokk „C” forráskódjának keretét az ISaGRAF Könyvtárrendező automatikusan generálja. A „Szerkesztés” menü **„Paraméterek”** parancsa az új funkcióblokk hívási és visszatérési paramétereinek a definiálására szolgál.

## ⇒ Egy „C” funkcióblokk alkalmazása egy ISaGRAF projektben

Egy ISaGRAF projekt programjaiban bármilyen integrált „C” funkcióblokk használható. A „C” funkcióblokkok az **ST** és **FBD** nyelvekből hívhatók meg.

Egy „C” funkcióblokknak az **ST** nyelvből történő meghívása a nyelv funkcióblokk hívási konvencióit követi. A funkcióblokk meghívási paramétere a funkció neve után vannak írva zárójelben, és azokat vessző választja el egymástól. A visszatérési paraméterek egyesével vannak elérve. Minden visszatérési paramétert egy név képvisel, amely a blokk előfordulás nevét és a paraméterek nevét egyesíti. A név komponensei ponttal vannak elválasztva egymástól. Például az alábbi név:

**FBINSTNAME.parname**

a **„FBINSTNAME”** nevű funkcióblokk előfordulás **„parname”** nevű visszatérési paraméterét képviseli.

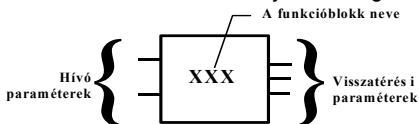
Az ST nyelvben használt funkcióblokkok előfordulásait a szótárban deklarálni kell. A blokk minden példányát (előfordulását) egy egyedi névvel kell azonosítani. Az alábbiakban az ISaGRAF szótárban levő előfordulás deklarálás példája látható:

előfordulás:	TRIG1	típus:	R_TRIG
	TRIG2		R_TRIG

Az alábbiakban ezeknek a deklarált előfordulásoknak egy ST programban való használatának példája látható:

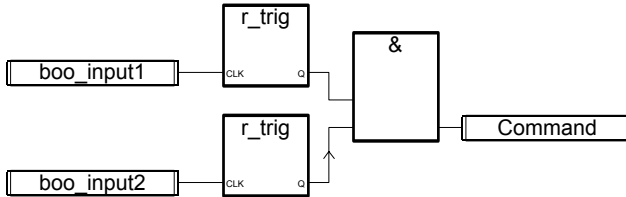
```
TRIG1 (boo_input1);
TRIG2 (boo_input2);
Command := (TRIG1.Q & TRIG2.Q);
```

Egy **FBD** program bármilyen „C” funkcióblokkot meghívhat. Egy funkcióblokk standard funkciókeretként van használva. Annak meghívó paramétere a funkció keret bal oldalához csatlakoznak. Annak visszatérési paramétere a keret jobb oldalához csatlakoznak. Egy funkció keret standard formátuma az alábbi módon jelenik meg:



Az FBD nyelvben használt funkcióblokkokat nem kell deklarálni, mert az ISaGRAF FBD szerkesztő automatikusan deklarálja a használt blokkok előfordulásait. Az FBD szerkesztő

által automatikusan deklarált funkcióblokk előfordulások mindig **LOKÁLISAK** a szerkesztett programhoz. Az alábbiakban az előző példa látható FBD nyelven programozva:



### **Egy „C” funkcióblokk interfészének definiálása**

A **"Szerkesztés"** menü **"Paraméterek"** parancsa egy új funkcióblokk hívási és visszatérési paramétereinek a definiálására szolgál. Egy funkcióbloknak maximum **32** paramétere lehet, meghívási és visszatérési paraméterek tetszés szerinti elrendezésével. Egy „C” funkciótól eltérően, egy funkcióbloknak több visszatérési paramétere lehet.

Az ablak bal felső oldalán látható lista a „C” funkcióblokk paramétereit mutatja, az alábbi funkció meghívási prototípus szerinti sorrendben: először a meghívó paraméterek, azután a visszatérési paraméterek. Az ablak alsó része a listában pillanatnyilag kiválasztott paraméter részletes leírását mutatja.

- a paraméter neve
- a paraméter iránya (meghívás/visszatérés)
- a paraméter típusa

Egy paraméterhez bármelyik ISaGRAF adat típus használható. Logikai, Integer analóg, Valós analóg, Időzítő, vagy Üzenet. Az integer és valós analógokat meg kell különböztetni egymástól. Az alábbiakban az ISaGRAF típusok és „C” típusok közötti összefüggés látható:

LOGIKAI	előjel nélküli hosszú	előjel nélküli 32 bites szó: 1=igaz / 0=hamis
ANALÓG	hosszú	előjel nélküli integer 32 bites szó
REAL	lebegő	egyszeres pontosságú lebegő érték
IDŐZÍTŐ	előjel nélküli hosszú	előjel nélküli integer 32 bites szó (az egység 1 ezredmásodperc)
ÜZENET	kar. *	karaktorsor

Amikor egy „C” funkcióhoz egy üzenet érték van küldve, az nem tartalmazhat nulla karaktereket. A „C” kódhoz küldött karaktorsor nulla lezárású. Ne feledje, hogy a visszatérés paraméternek kell az utolsónak lennie a listában. A paraméterek elnevezésénél az alábbi szabályokat kell betartani:

- a név hossza nem haladhatja meg a 16 karaktert
- az első karakternek betűnek kell lennie
- a többi karakternek betűnek, számjegynek, vagy („\_”) karakternek kell lennie
- az elnevezés nem érzékeny a kisbetű/nagybetű beállításra

Ugyanaz a név nem használható a funkcióblokk egynél több paraméteréhez. Egy meghívási paraméternek nem lehet ugyanaz a neve, mint egy visszatérési paraméteré. Ugyanaz a név

**használható** különböző funkcióblokkok paramétereire. Egy paraméter neve a paraméter azonosítására szolgál a „C” forráskódban.

A **„Beillesztés”** parancs egy új paraméternek a kiválasztott paraméter elé történő beillesztésére szolgál. A **„Törlés”** parancs a kiválasztott paraméter törlésére szolgál. Az **„Elrendezés”** parancs automatikusan átrendezi (sorrendbe állítja) a paramétereket úgy, hogy a visszatérési paraméterek a lista végére kerülnek. Az **„OK”** gomb megnyomása eltávolítja a funkcióblokk interfész definícióját, és bezárja a párbeszédablakot. A **„Mégsem”** gomb megnyomása a funkcióblokk interfész megváltoztatása nélkül bezárja a párbeszédablakot.

### **Funkcióblokk „C” interfész**

Egy funkcióblokk interfésze paramétereinek definíciójától függ. A hívási paraméterek egy struktúrán mennek keresztül. Ez a struktúra a **„GRFB0nnn.H”** fájlban van definiálva, ahol az **„nnn”** a funkcióblokk logikai száma az ISaGRAF könyvtárban. A visszatérési paramétereket logikai számok jelképezik, amelyek szintén a **„GRFB0nnn.h”** fájlban vannak definiálva. Az alábbiakban a „C” interfész egy példája látható, a **„LIM\_ALARM”** funkcióhoz (határértékek alarmja):

```
/* funkcióblokk interfész - név: minta */
```

```
/* standard ISaGRAF adattípusok */
```

```
typedef long T_BOO;
typedef long T_ANA;
typedef float T_REAL;
typedef long T_TMR;
typedef char *T_MSG;
```

```
/* meghívó paraméterek struktúrája */
```

```
typedef struct {
    /* MEGHÍVÁS */    T_BOO _par1;
    /* MEGHÍVÁS */    T_BOO _par2;
} str_arg;
```

```
/* hozzáférés a str_arg struktúra mezőéhez */
```

```
#define PAR1 (arg->_par1)
#define PAR2 (arg->_par2)
```

```
/* visszatérés paraméter logikai számok */
```

```
#define FBLPNO_Q1 0
#define FBLPNO_Q2 1
```

(\* a fájl vége \*)

Az alábbiakban az ISaGRAF típusok és a „C” típusok közötti összefüggés látható: Az ISaGRAF típusok a funkció definíciós fájljában „C” típusokként vannak definiálva.

logikai	T_BOO	hosszú (32 bit)
analóg	T_ANA	hosszú
valós	T_REAL	lebegő (32 bit – egyszeres pontosság)
időzítő	T_TMR	hosszú
üzenet	T_MSG	char * (32 bit - kar. mutató)

A „**str\_arg**” struktúra minden mezője a funkcióblokk egy paraméterének felel meg. A paraméterek ugyanazzal a sorrenddel jelennek meg a struktúrában, mint ami a funkcióblokk definíciójához lett megállapítva. Egy nagybetűs azonosító van definiálva a struktúra egy paraméterhez való közvetlen hozzáférésre, amely a funkcióblokk aktiválási szolgáltatás „C” kivitelezéséhez van átadva. Az azonosítók nevei azok, amelyek a funkcióblokk definíálása során az ISaGRAF Könyvtárrendezővel lettek beadva.

A visszatérési paraméterek számozásához alkalmazott sorrend ugyanaz, mint ami a funkcióblokk definíciójához lett megállapítva. Az első visszatérési paraméter logikai száma mindig **0**.

A „C” forrás programozásban a visszatérési paraméterek jelzésére számszerű értékek helyett definiált azonosítókat kell használni. Ez biztosítja, hogy a forrásfájl az interfész definíció módosítását követően könnyen átfordítható legyen.

A „C” definíciós fájl minden alkalommal frissítődik, amikor a funkcióblokk interfésze az ISaGRAF Könyvtárrendező használatával meg van változtatva. Ez teljes megfelelést biztosít a funkcióblokk kivitelezése és annak az ISaGRAF alkalmazások programjaiban történő használata között.

## ☐ **Forráskód**

Egy funkcióblokk „C” nyelvű kivitelezése három különböző belépési pontra van osztva:

- ☐ inicializálási szolgáltatás
- ☐ aktiválási szolgáltatás – a hívó paraméterek feldolgozása
- ☐ visszatérési paraméterek olvasási szolgáltatása

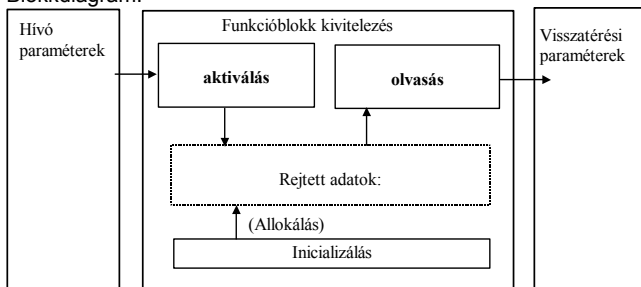
Ugyanannak a funkcióblokknak mindegyik előfordulásához ugyanaz a kód van használva, és az nincs duplikálva. Minden könyvtárelemhez egy sztatikus adatstruktúra tartozik. Az ilyen adatokhoz az ISaGRAF programozása nem tud közvetlenül hozzáférni, és azok a funkcióblokk előfordulás (rejtett változót) tartalmazzák.

Az „aktiválási szolgáltatás” mindegyik célciklusban mindegyik használt blokk mindegyik előfordulásához egyszer kerül meghívásra. Ez feldolgozza meghívó paramétereket, és frissíti a társult adatokat. Ez jelenti a funkcióblokk (fő algoritmusát).

Az „olvasási szolgáltatást” az ISaGRAF kernel hívja meg, egy előfordulás egy visszatérési paramétere pillanatnyi értékének a kiolvasására. Egy ilyen szolgáltatásban nem kell speciális

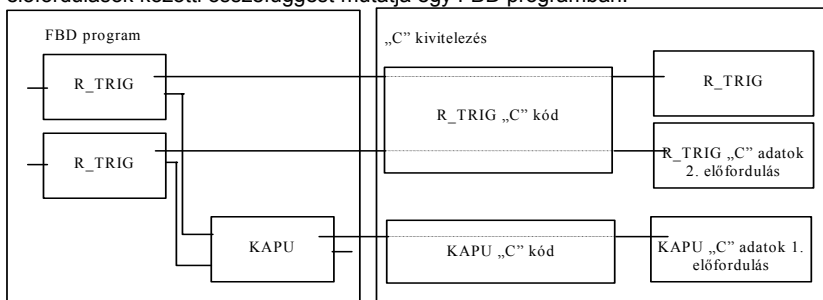
számítást elvégezni. Ez csupán a rejtett adatok és az ISaGRAF alkalmazás közötti adatátvitelt működteti.

Blokkdiagram:



#### • Funkcióblokk sztatikus adatok

Egy funkcióblokkok műveleteket és statikus adatokat társít. Ugyanannak a funkcióblokknak minden előfordulásához egy adatstruktúra van társítva. Valahányszor egy funkcióblokk használva van az ST vagy FBD programozásban, az egy előfordulásnak és egy adatstruktúrának felel meg. A következő példa a „C” adatstruktúrák és a funkcióblokk előfordulások közötti összefüggést mutatja egy FBD programban:



Az egyes előfordulások adatstruktúrájához szükséges memóriát az ISaGRAF rendszer allokalja, az alkalmazás indulásakor. Egy, a társult előfordulás adatstruktúrára mutató index átadásra kerül az „aktiválás” és „olvasás” szolgáltatásokhoz.

Az ISaGRAF Könyvtárrendező automatikusan generálja a „C” forráskód keretét az adatstruktúra típus definiálásához. Az adatstruktúra típusának neve mindig **str\_data**. A programozónak ezt a nevet nem szabad megváltoztatnia, hogy biztosítsa a szolgáltatás fejsorokkal való kompatibilitást. A rejtett adat általában belső változókat csoportosít a visszatérési paraméterek leképezésével. A funkcióblokk „olvasási” szolgáltatás csak a visszatérési paraméter elérésére van használva, és azt nem szabad más műveletek elvégzésére használni.

#### • Az inicializálási szolgáltatás

Egy funkcióblokk „inicializálás” szolgáltatását az ISaGRAF kernel hívja meg az alkalmazás elindulásakor. Ez a lehetővé teszi a „C” programozó számára, hogy a rendszertől memória

allokálását kérje egy előforduláshoz. Az alábbiakban az inicializálás szolgáltatás standard programozása látható:

```
uint16 FBINIT_xxx (uint16 hinstance)
/* "xxx" a f. blokk neve */
{
    return (sizeof (str_data));
}
```

A „**hinstance**” argumentum az előfordulás logikai száma. Ez ISaGRAF belső műveletekhez van fenntartva, és a szolgáltatás programozásához nem szabad használni. Az inicializálási szolgáltatás **egy** előfordulás adataihoz szükséges memória byte-ok számát adja vissza. A szükséges memória mennyisége (visszaadott érték) nem haladhatja meg a **64** Kbyte-ot. Ebben a szolgáltatásban nem szabad más műveletet elvégezni. Ennek a szolgáltatásnak a „C” forráskódját az ISaGRAF Könyvtárrendező automatikusan generálja a funkcióblokk létrehozásakor.

#### • Az aktiválási szolgáltatás

Az „aktiválási” szolgáltatás mindegyik célciklusban mindegyik használt blokk mindegyik előfordulásához egyszer kerül meghívásra. Ez a szolgáltatás a meghívó paramétereket dolgozza fel, és a fő funkcióblokk algoritmust futtatja a rejtett sztatikus adatok és a visszatérési paraméterek értékének a frissítése érdekében. Az alábbiakban az aktiválás szolgáltatás standard kerete látható:

```
void FBACT_xxx (
uint16 hinstance,                /* "xxx" a funkcióblokk neve */
                                /* az előfordulás logikai száma */
str_data *data,                 /* data:mutató az előfordulás adatstruktúrára*/
str_arg *arg                     /* mutató a meghívó paraméter
                                adatstruktúrájára */
)
{
}
```

A „**hinstance**” argumentum az előfordulás logikai száma. Ez ISaGRAF belső műveletekhez van fenntartva, és a szolgáltatás programozásához nem szabad használni. A „**data**” argumentum egy távoli mutató az előforduláshoz társult adatstruktúrára. A „**arg**” argumentum egy távoli mutató arra a struktúrára, amely a meghívó paraméterek értékét tartalmazza. A programozónak a funkcióblokk „C” fejsorában definiált azonosítókat kell használnia az „**arg**” struktúra mezőjéhez való hozzáféréshez.

Az „aktiválás” algoritmus feldolgozza meghívó paramétereket (amelyek az „**arg**” struktúrában vannak tárolva), és frissíti a „**data**” struktúra mezőit. Az alábbi

példa a **TRIG** (felfutó él detektálás) funkcióblokk „aktiválás” szolgáltatását mutatja:

```
/* a definíciók a funkcióblokk „C” fejsorában vannak tárolva */
```

```

typedef struct {
    T_BOO _clk;
} str_arg;

#define CLK    (arg->_clk)

/* funkcióblokk előfordulás adatstruktúra */

typedef struct {
    T_BOO prev_state;
    T_BOO edge_detect;
} str_data;

/* aktiválási szolgáltatás */

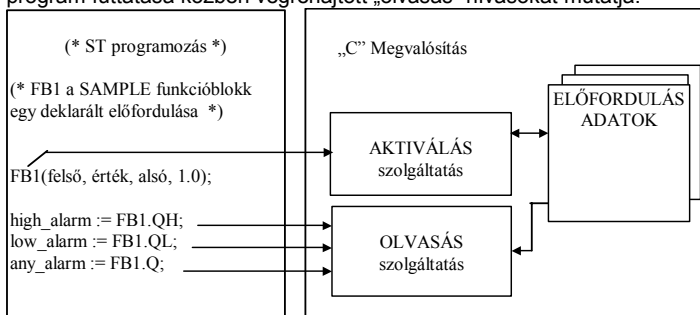
void FBACT_trig (uint16 hinstance, str_data *data, str_arg *arg)
{
    data->edge_detect = (T_BOO)(CLK && !data->prev_state);
    data->prev_state = CLK; /* meghívó paraméter */
}

```

Ennek a szolgáltatásnak a „C” forráskód keretét az ISaGRAF Könyvtárrendező automatikusan generálja a funkcióblokk létrehozásakor.

#### • A visszatérési paraméterek olvasása \*)

Az „olvasás” szolgáltatás mindannyiszor meg van hívva, valahányszor egy funkcióblokk előfordulás visszatérési paraméterére hivatkozás történik egy ST vagy FBD programban. Ez egy visszatérési paraméter értékének a megszerzésére szolgál. A következő példa egy ST program futtatása közben végrehajtott „olvasás” hívásokat mutatja:



Mivel az „olvasás” szolgáltatás ugyanabban a ciklusban egynél többször is meghívható ugyanahhoz a visszatérési paraméterhez vagy ugyanahhoz a funkcióblokk előforduláshoz,

ezért egy ilyen szolgáltatásban nem kell semmilyen speciális számítást elvégezni. Ez csupán a rejtett adatok és az ISaGRAF alkalmazás közötti adatátvitelt működteti. Az alábbiakban az olvasás szolgáltatás standard kerete látható:

*/\* egy visszatérési paraméter értékének másolására használt terjesztési művelet \*/*

```
#define BOO_VALUE      ((T_BOO *)value)
#define ANA_VALUE      ((T_ANA *)value)
#define REAL_VALUE     ((T_REAL *)value)
#define TMR_VALUE      ((T_TMR *)value)
#define MSG_VALUE      ((T_MSG *)value)
```

*/\* visszatérés paraméterek olvasási szolgáltatása: minden visszatérési paraméterhez meghívva \*/*

```
void FBREAD_xxx (                /* "xxx" a funkcióblokk neve */
uint16 hinstance,              /* az előfordulás logikai száma */
str_data *data,                /* mutató az előfordulás adatstruktúrára */
uint16 parno,                  /* az olvasás paraméter logikai száma */
void *value)                   /* puffer a paraméter értékének másolására */
{
    switch (parno) {
        case FBLPNO_XX: /* ... */ break;
        case FBLPNO_YY: /* ... */ break;
        /* .... */
    }
}
```

A „**hinstance**” argumentum az előfordulás logikai száma. Ez ISaGRAF belső műveletekhez van fenntartva, és a szolgáltatás programozásához nem szabad használni. A „**data**” argumentum egy távoli mutató az előforduláshoz társult adatstruktúrára.

A „**parno**” argumentum annak a kívánt érték visszatérési paramétere előfordulásának a logikai száma. A visszatérési paraméterek azonosítására a funkcióblokk „C” fejsorában definiált azonosítókat kell használni. Az ilyen azonosítók az „**FBLPNO\_**” előtaggal kezdődnek. A „**value**” argumentum egy távoli mutató arra a pufferre, amelybe az elért visszatérési paraméter pillanatnyi értéke másolandó. Az ezzel az argumentummal mutatott adatok típusa a visszatérési paraméter ISaGRAF típusától függ. Az alábbiakban az ISaGRAF típusok és a puffer „C” típusok közötti összefüggés látható:

logikai	hosszú	előjel nélküli 32 bites szó (0=hamis, 1=igaz)
analóg	hosszú	32 bites előjellel ellátott szó
valós	float	32 bites egyszeres pontosságú lebegő érték
időzítő	hosszú	32 bites előjel nélküli szó (egysége: 1 ms)
üzenet	kar. *	karakterek tömbje

A következő makrók a másolási pufferhez való hozzáféréshez használatosak, az elért visszatérési paraméter típusának megfelelően:

#define	BOO_VALUE	((T_BOO *)value)
#define	ANA_VALUE	((T_ANA *)value)
#define	REAL_VALUE	((T_REAL *)value)
#define	TMR_VALUE	((T_TMR *)value)
#define	MSG_VALUE	((T_MSG *)value)

Ezek az általában használt programozott műveletek az érték vagy a paraméter ISaGRAF pufferbe másolására:

```
/* egy logikai paraméterhez: */
*BOO_VALUE = parameter_value;
/* egy integer analóg paraméterhez: */
*ANA_VALUE = parameter_value;
/* egy valós integer paraméterhez: */
*REAL_VALUE = parameter_value;
/* egy időzítő paraméterhez: */
*TMR_VALUE = parameter_value;
/* egy karaktersor paraméterhez: */
strcpy (*MSG_VALUE, parameter_value);
```

Ennek a szolgáltatásnak a „C” forráskód keretét az ISaGRAF Könyvtárrendező automatikusan generálja a funkcióblokk létrehozásakor.

- **A „C” forrásfájl példája**

Az alábbiakban egy „C” funkcióblokk kivitelezés standard kerete látható:

```
/* funkcióblokk (xxx a funkcióblokk neve) */

#include <tasy0def.h>
#include <grfb0nnn.h> /* nnn a könyvtárban levő funkcióblokk száma */

/* rejtett adatok struktúrája a blokkban levő minden előforduláshoz */
typedef struct {
    /* mezők definíciója */
} str_data;

/* inicializálási szolgáltatás: a szükséges rejtett adatok méretét adja vissza */
word FBINIT_xxx (uint16 hinstance)
{
    return (sizeof (str_data));
}
```

```

/* aktiválási szolgáltatás – a hívó paraméterek feldolgozása */
void FBACT_xxx (uint16 hinstance, str_data *data, str_arg *arg)
{
    /* ... */
}

/* egy visszatérési paraméter értékének másolására használt terjesztési művelet */
#define BOO_VALUE ((T_BOO *)value)
#define ANA_VALUE ((T_ANA *)value)
#define REAL_VALUE ((T_REAL *)value)
#define TMR_VALUE ((T_TMR *)value)
#define MSG_VALUE ((T_MSG *)value)

/* visszatérés paraméterek olvasási szolgáltatása: minden visszatérési paraméterhez
meghívva */
void FBREAD_xxx (uint16 hinstance, str_data *data, uint16 parno, void *value)
{
    switch(parno)
    {
        case FBLPNO_XX: *???_VALUE = ...; break;
        case FBLPNO_YY: *???_VALUE = ...; break;
        ....
    }
}

/* A következő funkció a funkcióblokk inicializálására és kivitelezésének
deklarálására használatos. Ez az ISaGRAF I/O kernellel való kapcsolatot valósítja
meg, a funkcióblokk nevének használatával. Ezt a szolgáltatást teljes egészében az
ISaGRAF Könyvtárrendező generálta. */

ABP fbldef_xxx (char *name, IBP *initproc, RBP *readproc)
{
    strcpy (name, "XXX");
    *initproc = (IBP)FBINIT_xxx;
    *readproc = (RBP)FBREAD_xxx;
    return ((ABP)FBACT_xxx);
}

(* a fájl vége *)

```

A „TASY0DEF.H” beletartozó fájl az ISaGRAF kernelből a rendszertől független definíciókhoz szükséges. Ez a kivitelezett szolgáltatásokhoz mutató távoli indexeket jelentő adattípusok definícióját is tartalmazza.

### ■ A projektek és a „C” kivitelezés közötti kapcsolatok

Egy „C” funkcióblokk kivitelezése és annak egy ISaGRAF projekt programjaiban történő alkalmazása közötti logikai kapcsolat a funkció nevének használatával van létrehozva. A funkcióblokk „C” forráskódjához egy „deklarálás” szolgáltatás van adva. Ez a szolgáltatás csak egyszer van meghívva, amikor az alkalmazás elindul, és a kivitelezett funkciónak megfelelő „C” funkcióblokk nevet jelzi az ISaGRAF kernel felé. Az alábbiakban egy ilyen deklarálási szolgáltatás standard formája látható:

```
ABP fbldef_xxx (char *name, IBP *initproc, RBP *readproc)
{
    strcpy (name, "XXX");           /* a funkcióblokk neve */
    *initproc = (IBP)FBINIT_xxx;    /* inicializálási szolgáltatás */
    *readproc = (RBP)FBREAD_xxx;    /* olvasási szolgáltatás */
    return ((ABP)FBACT_xxx);        /* aktiválási szolgáltatás */
}
/* .xxx a funkcióblokk neve */
```

A **strcpy** utasításhoz használt funkcióblokk nevét **nagybetűvel** kell írni. Ezt a kivitelezett szolgáltatások és a deklarálás szolgáltatás nevében kisbetűvel kell írni.

A „FBACT\_”, „FBINIT\_”, „FBREAD\_” és „fbldef\_” előtagok a kivitelezett szolgáltatásoknál és definíciós szolgáltatásnál lehetővé teszi a felhasználó számára egy funkcióblokk elnevezését a „C” nyelv egyik fenntartott kulcsszavával, vagy a „C” ISaGRAF könyvtárak egyik létező funkciójának nevével. A deklarálási szolgáltatáshoz nem szabad más utasítást adni.

A deklarálási funkció bármilyen integrált „C” funkcióblokkhoz meg van hívva, még ha az nincs is az ISaGRAF alkalmazás programjaiban felhasználva. Az ISaGRAF kernel végzetes hibával megáll, ha egy, az alkalmazásban használt „C” funkcióblokk nincs integrálva a kernelbe.

Új funkcióblokkoknak a kernellel történő összekapcsolása előtt a felhasználónak egy másik, „GRFB0LIB.C” nevű „C” forrásfájl kell írnia, és azt (a visszatartott funkcióblokkokkal együtt) a fájl listájába kell illesztenie az összekapcsoló számára. A „GRFB0LIB.C” csak deklarációs szolgáltatások tömbjét tartalmazza. Ez a tömb az alkalmazás inicializálások során van kiolvasva, egy dinamikus kapcsolat létrehozására a „C”-ben írt funkcióblokkokkal. Az alábbiakban egy ilyen fájl példája látható:

/\* Fájl: grfb0lib.c – kivitelezett funkcióblokkok \*/

```
#include <tasy0def.h>
```

```
extern ABP fbldef_fb1(char *name, IBP *init, RBP *read);
extern ABP fbldef_fb2(char *name, IBP *init, RBP *read);
```

```
FBL_LIST FBLDEF[ ] = {
    fbldef_fb1,
    fbldef_fb2,
```

```
NULL };
```

(\* a fájl vége \*)

A **FBLDEF** tömböt egy NULL mutatóval kell lezárni. Ha ezek a feltételek nincsenek teljesítve, akkor bizonyos ütközések következhetnek be. Az új ISaGRAF kernel kapcsolása feloldatlan hivatkozásokat eredményez, ha a **FBLDEF** tömb nincs definiálva.

Ennek a fájlnek a megírásával egy új kernel építhető fel, amely az összes meglevő funkcióblokkot tartalmazza. Egy kernel egyetlen projekthez tesztre szabva is felépíthető, a **FBLDEF** tömbbe csak a projektben használt funkcióblokkokat beillesztve. Egy alkalmazás kódjának felépítésekor az ISaGRAF Kódgenerátor automatikusan generálja a "**GRFB0LIB.C**" fájlt. A fájl az ISaGRAF projekt alkönyvtárba van elhelyezve, és csak a projektben használt funkcióblokkokat csoportosítja.

## ☐ **Korlátok**

Az ISaGRAF könyvtár maximum **255** „C” funkcióblokkot tartalmazhat. Egy funkcióban bármilyen típusú művelet feldolgozható. Minden típusú funkcióblokk ugyanabban a projektben maximum **255**-ször másolható (instancionálható).

Megjegyzendő, hogy a funkcióblokk szolgáltatások az ISaGRAF ciklusban **szinkronizáltan** vannak meghívva, úgy hogy a funkcióblokk végrehajtása közvetlenül hat a ciklusidőzítésre.

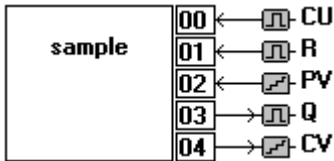
## ☐ **Teljes példa:**

Az alábbiakban egy „**mint**a” funkcióblokk teljes programozása látható, amely egy felfelé működő számláló:

Az alábbiakban a funkcióblokk műszaki megjegyzése látható:

név:	SAMPLE
leírás:	Fel számláló
létrehozás dátuma:	1994. február 1.-je
szerző:	ICS Triplex ISaGRAF
meghívás:	CU : számlálás input R : visszaállítás parancs PV : programozott maximum érték
visszatérés:	Q : max detektálás CV : számlálási eredmény
prototípus:	SAMPLE ( count, reset_command, maximum_value); max_detect := SAMPLE.Q; count_result := SAMPLE.CV;

Az alábbiakban a funkcióblokk interfésze látható:



Az alábbiakban a funkcióblokk „C” forrás fejsora látható:

```
/* funkcióblokk interfész - név: SAMPLE */

/* standard ISaGRAF adattípusok definíciója */

typedef long T_BOO;
typedef long T_ANA;
typedef float T_REAL;
typedef long T_TMR;
typedef char *T_MSG;

/* a meghívó paraméter struktúrájának definíciója */

typedef struct {
    T_BOO _cu;
    T_BOO _r;
    T_ANA _pv;
} str_arg;

/* a meghívó paraméterek eléréséhez használt azonosítók */

#define CU (arg->_cu)
#define R (arg->_r)
#define PV (arg->_pv)

/* visszatérés paraméterek logikai számozása */

#define FBLPNO_Q 0
#define FBLPNO_CV 1

(* a fájl vége *)
```

Az alábbiakban a funkcióblokk „C” forráskódja látható: A C programozó csak a kiemelt karakterekkel nyomtatott sorokat írta be manuálisan.

```

/* funkcióblokk - név: SAMPLE */

#include <tasy0def.h>          /* adattípus definícióhoz szükséges */
#include <grfb0255.h>         /* funkcióblokk C forrás fejsor */

/* az egy előfordulás adatai tartalmazó struktúra definíciója */

typedef struct {
    T_BOO overflow;        /* igaz: számlálási érték >= programozott érték */
    T_ANA value;          /* számlálás pillanatnyi érték */
} str_data;

/* inicializálási szolgáltatás: memóriát igényel az előfordulás adatokhoz */

word FBINIT_sample (uint16 hinstance)
{
    return (sizeof (str_data));
}

/* aktiválási szolgáltatás: felfelé számláló algoritmus */

void FBACT_sample (uint16 hinstance, str_data *data, str_arg *arg)
{
    if (R) data->value = 0;
    else if (CU && data->value < PV) (data->value)++;
    data->overflow = (data->value >= PV) ? (T_BOO)1 : (T_BOO)0;
}

/*paramétereknek az ISaGRAF pufferbe másolásához szükséges terjesztési művelet*/

#define BOO_VALUE ((T_BOO *)value)
#define ANA_VALUE ((T_ANA *)value)
#define REAL_VALUE ((T_REAL *)value)
#define TMR_VALUE ((T_TMR *)value)
#define MSG_VALUE ((T_MSG *)value)

/* olvasási szolgáltatás: egy visszatérési paraméter értékének megszerzése */

```

```
void FBREAD_sample (uint16 hinstance, str_data *data, uint16 parno, void
*value)
{
    switch (parno) {
        case FBLPNO_Q : *BOO_VALUE = data->overflow; break;
        case FBLPNO_CV : *ANA_VALUE = data->value; break;
    }
}
```

/\* Deklarációs szolgáltatás ISaGRAF kernellel való dinamikus kapcsolathoz \*/

```
ABP fbldef_sample (char *name, IBP *initproc, RBP *readproc)
{
    strcpy (name, "SAMPLE");
    *initproc = (IBP)FBINIT_sample;
    *readproc = (RBP)FBREAD_sample;
    return ((ABP)FBACT_sample);
}
```

(\* a fájl vége \*)

### C.7.5. Programfordítási és kapcsolási módszerek

Az ISaGRAF Workbench nem tartalmaz semmiféle „C” programfordítót vagy összekapcsolót. Ez a fejezet azonban ismerteti azokat a fő módszereket, amelyeknek az alkalmazásával az ISaGRAF Könyvtárrendező által létrehozott fájlok egyszerűen használhatók, és más eszközökre, pl. programfordítókra és összekapcsolókra átadhatók.

#### == C forrásfájlok

A konverziók, funkciók és funkcióblokkok „C” forrásfájlait az ISaGRAF Könyvtárrendező az **ISAWIN\LIB\DEFS** és **ISAWIN\LIB\SRC** alkönyvtárakba rakja. Egy forrásfájl neve a megfelelő konverzió, funkció vagy funkcióblokk számával van kialakítva az ISaGRAF könyvtárban. A használt fájlnevek az alábbiakban vannak felsorolva:

isawin\lib\defs\TACN0DEF.H	definíciós fájl konverziós funkciókhoz
isawin\lib\src\GRCN0nnn.H	egy konverziós funkció forrásfájlja
isawin\lib\defs\GRUS0nnn.H	egy funkció definíciós fájlja
isawin\lib\src\GRUS0nnn.C	egy funkció forrásfájlja
isawin\lib\defs\GRFB0nnn.H	egy funkcióblokk definíciós fájlja
isawin\lib\src\GRFB0nnn.C	egy funkcióblokk forrásfájlja

(az **nnn** a konverzió, funkció vagy funkcióblokk száma)

**Figyelmeztetés:** Könyvtárelemek átnevezésénél vagy másolásánál azok szövegét és programozási sorait az ISaGRAF Könyvtárrendező nem frissíti az új elem névnek és logikai számnak megfelelően. Ezeket manuálisan kell frissíteni a „C” forrásfájlban.

Az **ISAWIN\LIB\USPNUMS** fájl az ISaGRAF könyvtárban levő „C” funkciók nevei és logikai számai közötti összefüggést adja meg. Az alábbiakban egy ilyen fájl példája látható:

```
1      funct_A
10     funct_B
16     funct_C
```

Az **ISAWIN\LIB\FBLNUMS** fájl az ISaGRAF könyvtárban levő „C” funkcióblokkok nevei és logikai számai közötti összefüggést adja meg. Az alábbiakban egy ilyen fájl példája látható:

```
0      fbl_A
1      fbl_B
2      fbl_C
```

Az **ISAWIN\LIB\CNVNUMS** fájl az ISaGRAF könyvtárban levő konverziós funkciók nevei és logikai számai közötti összefüggést adja meg. Az alábbiakban ennek a fájlnak a példája látható a standard könyvtár konverzióihoz:

```
0      SCALE
1      BCD
```

Ezeket a fájlokat az ISaGRAF Könyvtárrendező automatikusan frissíti, valahányszor egy konverzió, funkció vagy funkcióblokk létrehozásra, átnevezésre, másolásra, vagy törlésre kerül. Az ISaGRAF Kódgenerátor egy alkalmazás felépítésekor automatikusan a következő fájlokat generálja:

<code>\isawin\apl\ppp\GRCN0LIB.C</code>	A projektben használt összes konverziós funkció tömbként történő deklarálása.
<code>\isawin\apl\ppp\GRUS0LIB.C</code>	A projektben használt összes funkció tömbként való deklarálása.
<code>\isawin\apl\ppp\GRFB0LIB.C</code>	A projektben használt összes funkcióblokk tömbként való deklarálása.

(a **ppp** az ISaGRAF projekt neve)

Ezek a fájlok kapcsolási műveletek közben használhatók egy új, a projekthez szánt ISaGRAF kernel felépítésére, amely csak a projektben használt konverziókat, funkciókat és funkcióblokkokat tartalmazza.

## ☐ **Forrásfájlok letöltése egy natív rendszerbe**

Az ISaGRAF Könyvtárrendező által létrehozott „C” forrás- és definíciós fájlok letölthetők a cél ISaGRAF rendszerre, ha az támogat egy natív programfordító eszközt. Ennek elvégzéséhez a Windows részét képező standard **TERMINAL** eszköz használható.

Amikor a célrendszeren forrásfájlok kezelése történik, akkor a definíciós fájlokat egy új letöltési művelettel frissíteni kell valahányszor az ISaGRAF Könyvtárrendező egy funkció interfészét módosítja.

A fájlok letöltésének parancssorait pl. egy csomagfájlba lehet csoportosítani, és azokat a workbench eszköz menüjéről el lehet indítani (lásd a Programok kezelése c. kezelési útmutatót).

### ☐ **Egy keresztfordító használata**

A forrásfájlok a PC-n közvetlenül is kezelhetők, ha rendelkezésre áll a cél PC vagy egy keresztfordító, amely a PC-n fut, a célrendszerhez generálva kódot.

Ebben az esetben a felhasználó az ISaGRAF Könyvtárrendezőt futtathatja a konverziók, funkciók vagy funkcióblokkok forrásainak módosításához.

A programfordító és összekapcsoló futtatásának parancssorait pl. egy csomagfájlba lehet csoportosítani, és azokat a workbench eszköz menüjéről el lehet indítani (lásd a Programok kezelése c. kezelési útmutatót).

Amikor a PC-n konverziók, funkciók és funkcióblokkok vannak lefordítva, a felhasználónak az alkalmazások futtatása előtt egyszerűen le kell töltenie az újonnan generált ISaGRAF kernelt (amely kapcsolva van az új komponensekkel) a célrendszerre. Ha a cél egy másik PC, akkor az új generált ISaGRAF kernel a célgépbe egy lemezzel, vagy egy hálózat alkalmazásával tölthető be.

### ☐ **Kapcsolás az ISaGRAF kernel könyvtárakhoz**

#### **Figyelmeztetés :**

A következőkben általános információk szerepelnek, amelyek nem biztos, hogy pontosan megfelelnek az Ön célrendszerének.

Mindenesetre a cél-lemezen található .TXT fájlok tájékoztatást nyújthatnak.

Az ISaGRAF céllemez számos segédprogram fájlt tartalmaz a konverziók, funkciók és funkcióblokkok fordításához és az ISaGRAF kernel könyvtárakhoz történő kapcsolásához.

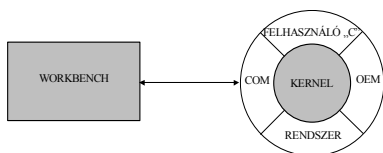
Két kivitelezés létezik:

- egyszeres feladatú ISaGRAF: minden funkció ugyanabban a programban van végrehajtva
- többfeladatos ISaGRAF: a kommunikációhoz egy külön feladat (vagy lánc) van szánva

A „C” komponensek mindkét esetben ugyanazokban a könyvtárakban vannak csoportosítva: a „C” programozó számára nincs különbség az egyszeres feladat vagy a multitaszk között. Egy egyfeladatos változatnál a felhasználói „C” könyvtárak az egyszeres feladathoz vannak kapcsolva (amelyeket általában **isa**-nak neveznek), míg a multitaszk változatnál a könyvtárak a kernel feladathoz vannak kapcsolva (melynek neve általában **isaker**).

**Fejlesztés  
rendszer**

**Cél  
rendszer**

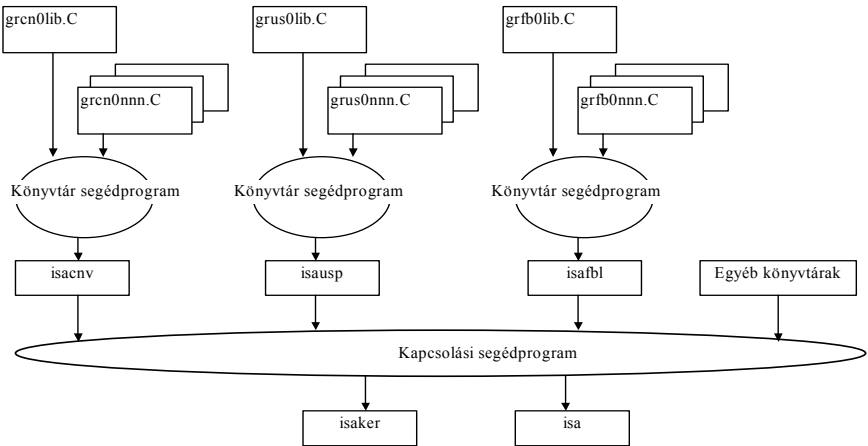


Az ISaGRAF szoftver belső része független a hardvertől. Ez az IEC nyelveket hajtja végre, és saját változó adatbázisa van.

A kernellel történő kapcsolat létrehozásánál az első lépés az adott projekthez szükséges összes konverzió, funkció és funkcióblokk könyvtárainak a felépítése:

Könyvtár	tartalom
ISAUSP	- GRUS0LIB tárgyfájl (deklarált funkciók tömbje) - az egyes integrált funkciók tárgyfájlja
ISAFBL	- GRFB0LIB tárgyfájl (deklarált funkcióblokkok tömbje) - az egyes integrált funkcióblokkok tárgyfájlja
ISACNV	- GRCN0LIB tárgyfájl (deklarált konverziók tömbje) - az egyes integrált konverziós funkciók tárgyfájlja

Ezután a programozónak ezeket az új könyvtárakat az ISaGRAF kernel többi tárgyfájljaival és könyvtáraival össze kell kapcsolnia. A felhasználó által elvégzett „C” fejlesztés integrálás különböző fázisai a következő ábrán vannak vázolva:



Az alábbiakban a kapcsolat során összekötendő tárgymodulok és könyvtárak pontos listája szerepel:

<u>Az isaker felépítéséhez:</u>		
Tárgymodul:	<b>tast0mai</b>	
Tárgymodul:	<b>tats0com</b>	
Kernel könyvtár:	<b>isaker</b>	
Kernel könyvtár:	<b>isaoem</b>	
Felhasználói könyvtár:	<b>isausp</b>	felhasználó által definiált funkciók felhasználó által definiált funkcióblokkok
Felhasználói könyvtár:	<b>isafbl</b>	

Felhasználói könyvtár:	<b>isacnv</b>	felhasználó által definiált konverziós funkciók
Kernel könyvtár:	<b>isasys</b>	
Rendszer könyvtárak:	(lásd a „C” programfordító kézikönyvét)	

Az isa felépítéséhez:

Tárgymodul:	<b>tast0mai</b>	
Tárgymodul:	<b>tast0com</b>	
Kernel könyvtár:	<b>isaker</b>	
Kernel könyvtár:	<b>isatst</b>	
Kernel könyvtár:	<b>isaoem</b>	
Felhasználói könyvtár:	<b>isausp</b>	felhasználó által definiált funkciók
Felhasználói könyvtár:	<b>isafbl</b>	felhasználó által definiált funkcióblokkok
Felhasználói könyvtár:	<b>isacnv</b>	felhasználó által definiált konverziós funkciók
Kernel könyvtár:	<b>isasys</b>	
Rendszer könyvtárak:	(lásd a „C” programfordító kézikönyvét)	

Lehetséges, hogy a fordítónak pontosan be kell tartania a tárgymoduloknak és könyvtáraknak az előző ábrán látható sorrendjét. A tárgymoduloknak és könyvtáraknak a célrendszernek megfelelő standard kiterjesztéseik vannak („lib”, „obj”, „l”, „r”...).

### Szükséges programfordítási és kapcsolási beállítások

A programfordítás és kapcsolat közben kényelmes beállítási lehetőségeket lehet kiválasztani. Ezek a konverziókban, funkciókban és funkcióblokkokban feldolgozott műveletek típusától függenek. Egyes műveletek a kapcsolat során más rendszerkönyvtárakat (matematika, grafika, stb.) igényelnek.

Az ISaGRAF Kernel valamennyi „C” forrásfájlla a **LARGE** memóriamodullal lett lefordítva. A programozónak ugyanazt a modellt kell használnia a konverziók, funkciók és funkcióblokkok fordításakor.

A „C” könyvtárkomponensek fordításához egy speciális konstanst kell definiálni. Ez jelzi a célrendszer típusát és processzorjait, hogy a konverziók, funkciók és funkcióblokkok forrása rendszerfüggetlen lehessen. Az alábbiakban ezeknek a konstans értékeknek a nevei láthatók:

**DOS** ..... DOS alapú rendszerekhez (INTEL processzor)  
**ISAWNT** ..... Windows-NT alapú rendszerekhez (INTEL processzor)  
**OS9** ..... OS9 rendszerhez (MOTOROLA processzor)  
**VxWorks** ..... VxWorks rendszerhez (MOTOROLA processzor)

Az ISaGRAF célszoftverrel szállított segéd parancsfájlok (a fordításhoz és kapcsoláshoz) azt mutatják meg, hogy hogyan kell egy kényelmes konstans értéket definiálni a fordító parancssorában.

### ☐ **Támogatott programfordító**

A konverziók, funkciók és funkcióblokkok, valamint az ISaGRAF Kernellel való kapcsolat fejlesztéséhez a következő programfordítók vannak támogatva:

Microsoft MSC 7.00 fordító	DOS alapú célokhoz
Microsoft MSVC 4.00 fordító	Windows-NT alapú célokhoz
Microware ULTRA-C fordító	OS-9 célokhoz
Tornado 1.0; GNU Toolkit 2.6	VxWorks célokhoz

Egyéb programfordítók használatához lépjen kapcsolatba a ICS Triplex ISaGRAF-lal.

### ☐ **Összefoglalás**

Az alábbiakban egy új konverzió, funkció vagy funkcióblokk kifejlesztéséhez elvégzendő műveletek összefoglalása szerepel.

- ⇒1. Az ISaGRAF Könyvtárrendezővel hozza létre az új elemet: nevezze el, és adjon hozzá megjegyzés szöveget. A „C” forrásfájl kerete automatikusan generálódik.
- ⇒2. Az ISaGRAF Könyvtárrendezővel írja le az interfészt (hívási és visszatérési paraméterek), ha az elem egy funkció vagy funkcióblokk. A „C” forrás fejsor fájl automatikusan generálódik.
- ⇒3. Az ISaGRAF Könyvtárrendezővel írja be az elem részletes műszaki megjegyzésének (kezelési útmutató) szövegét.
- ⇒4. Az ISaGRAF Könyvtárrendezővel a konverzió, funkció vagy funkcióblokk algoritmus „C” programozásának bevitelével tegye teljessé a „C” forrásfájl. Ezzel kész az elem forráskódja. Megjegyzendő, hogy más szerkesztő is használható.
- ⇒5. Válassza ki a Könyvtárrendező „**Logikai szám megjelenítése**” beállítását ahhoz, hogy megtudja, milyen logikai szám van az új elemhez csatolva. Ez a megfelelő „C” és „H” forrásfájlok útvonal neveiben van használva.
- ⇒6. Másolja / Töltse le a .C és .H fájlokat a célra (natív fordító esetén) vagy abba a megfelelő környezetbe (keresztfordítónál), ahová az ISaGRAF célkönyvtárak és feladatok vannak telepítve.
- ⇒7. Futtassa az új forrásfájl „C” fordítóját, és javítsa ki az esetleges szintaktikai hibákat.
- ⇒8. Illesse be az új elem deklarálási szolgáltatás nevét a „**GR??0LIB.C**” forrásfájlba, amely meghatározza a beillesztett elemek tömbjét.
- ⇒9. Futtassa a „C” fordítót a „**GR??0LIB.C**” fájl lefordításához.

- ⇒10. Illessze be a megfelelő könyvtár felépítéséhez használt tárgyfájlok listájában levő tárgymodul nevét.
- ⇒11. Futtassa a „C” könyvtárépítőt. Futtassa a „C” kapcsolót, egy új kernel felépítéséhez.
- ⇒12. Telepítse az újonnan létrehozott kernelt a célgépre.
- ⇒13. Írjon egy minta ISaGRAF alkalmazást, amely leteszteli az új elem aktiválását és interfészét.

## C.8. Modbus kapcsolat

Ha az alkalmazás teljesen ki lett fejlesztve és le lett tesztelve, azt egy folyamat vizualizáló rendszerhez lehet csatlakoztatni.

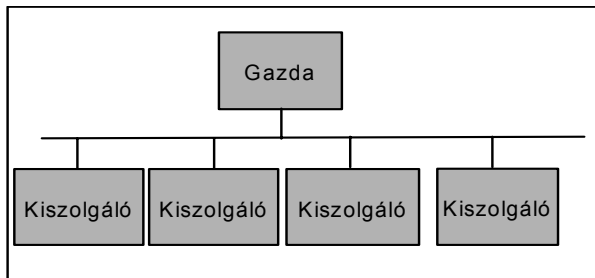
Az ISaGRAF egy nyitott rendszer, amely számos hálózati lehetőséget kínál.

A legegyszerűbb ipari hálózat a MODBUS/MODICON standard protokoll, amely majdnem minden folyamat vizualizáló rendszeren rendelkezésre áll, és amely csak egy soros kapcsolatot igényel (RS232, RS485, Current Loop).

Az ISaGRAF kommunikáció hibakereső protokoll MODBUS kompatibilis, hogy lehetővé tegye a változó olvasást/írást egy Modbus gazdáról.

### C.8.1. MODBUS hálózat és protokoll

Egy Modbus hálózat egyetlen gazdaállomásból (általában egy folyamat vizualizáló rendszer), és egy vagy több kiszolgáló állomásból (általában PLC-k) áll.



A gazda egyszerre egy igényt küld egy kiszolgálóhoz (annak kiszolgáló számát használva), és a következő kérdés elküldése előtt vár annak a kiszolgálónak a válaszára. A többi nem érintett kiszolgáló nem válaszol.

Minden keret egy kiszolgáló számot, egy igény számot és a megfelelő adatokat, valamint egy 16 bites ellenőrző összeget (CRC) tartalmaz.

Ha egy időtúllépési időtartamon belül nem érkezett válasz, akkor a kérdés bizonyos számú alkalommal megismételhető, mielőtt a gazda a kiszolgálót „leválasztott” állapotúvá nyilvánítja. Az időtúllépési értéket és a próbálkozások számát a gazdaállomáson kell beállítani úgy, hogy megfeleljen a kiszolgáló követelményeinek (az alkalmazástól, stb. függően).

Ha egy feldolgozási igény során hiba történik, akkor a kiszolgáló aárt válaszkereket küldése helyett egy hibaüzenetet adhat ki.

A Modbus egy Modicon protokoll, de nem egy nemzetközi standard, és számos különböző „Modbus” kompatibilis protokoll megvalósítás létezik, számos lehetséges eltérésekkel, mint pl. a következők:

- Megvalósított funkciókódok listája
- Cím hozzárendelés
- RTU (bináris kódok) vagy ASCII protokoll, stb.

### **C.8.2. ISaGRAF kivitelezés**

#### **Alkalmazás változó hozzáférés**

Az ISaGRAF kommunikációs kapcsolat öt Modbus funkciókódot ismer fel:

1	N bit olvasása
3	N szó olvasása
5	1 bit írása
6	1 szó írása
16	N szó írása

Az ISaGRAF alkalmazás változók „hálózati címükön” keresztül érhetők el, ha pl. azok definiálva lettek a workbench szótárban. Ezek a változók a következők lehetnek:

- Logikai vagy Analóg változók
- inputok, outputok, vagy belső változók
- lokális vagy globális változók.

Egy Logikai változó írásához az 5-ös, 6-os, vagy 16-os funkció használható. Az íráshoz egy IGAZ érték minden nem zéró érték.

Egy Logikai változó olvasásához az 1-es vagy 3-as funkció használható. Az 1-es funkcióval az értékek egy bit mezőben, a 3-as funkcióval pedig Byte-okban vannak előhozva (egy IGAZ érték a 0xFFFF-nak felel meg).

Egy analóg változó írásához a 6-os, vagy 16-os funkció használható. Az érték egy 16 bites integer, amely –32768-tól +32767-ig terjedhet (az ISaGRAF célváltozók 32 bitesek).

Egy analóg változó olvasásához a 3-as funkció használható. Az érték egy 16 bites integer, amely –32768-tól +32767-ig terjedhet. A cél oldalon az Analóg változók 32 bitesek, ezért a célon egy 16 bites tartományon túli érték (pozitív vagy negatív) a maximális 16 bites tartományú (pozitív vagy negatív) értékkel lesz olvasva.

A valós változók nem érhetők el egy Modbus igénnyel.

**Figyelmeztetés :**

Az ISaGRAF kivitelezés nem kezel olyan hibákat mint pl. az „ismeretlen modbus cím”.

**Jelölések:**

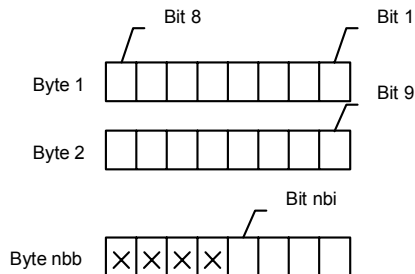
slv	kiszolgáló szám
nbw	szavak száma
nbb	byte-ok száma
nbi	bitek száma
addH	hálózati cím (Felső Byte)
addL	hálózati cím (Alsó Byte)
vH	érték (Felső Byte)
vL	érték (Alsó Byte)
V	Byte Érték
bfd	Bit mező (nbb Byte)
crcH	ellenőrző összeg (Felső Byte)
crcL	ellenőrző összeg (Alsó Byte)

**1. FUNKCIÓ: N bit olvasása**

nbi biteket olvas (Logikaiak), az addH/addL hálózati címtől kezdve

Kérdés	slv	01	addH	addL	00	nbi	crcH	crcL
Válasz	slv	01	nbb	bfd	...		crcH	crcL
			Byte 1		Byte nbb			

a bfd az nbb Byte-ok bit mezője, a következő formátummal:



A Bit 1 az addH/addL hálózati cínnél levő változó értékének felel meg.

A Bit nbi az addH/addL + nbi - 1 hálózati cínnél levő változó értékének felel meg.

Az X definiálatlan értéket jelent.

**3. FUNKCIÓ: N szó olvasása**

nbw szót olvas, az addH/addL hálózati címtől kezdve

Kérdés	slv	03	addH	addL	00	nbw	crcH	crcL
--------	-----	----	------	------	----	-----	------	------

Válasz	slv	03	nbb	vH	vL	...	crcH	crcL
--------	-----	----	-----	----	----	-----	------	------

az nbb a vH, vL byte-ok számának felel meg

## 5. FUNKCIÓ: 1 bit írása

Egy bitet (Logikai) ír az addH/addL hálózati címnél

Kérdés	slv	05	addH	addL	vH	00	crcH	crcL
--------	-----	----	------	------	----	----	------	------

Válasz	slv	05	addH	addL	vH	00	crcH	crcL
--------	-----	----	------	------	----	----	------	------

## 6. FUNKCIÓ: 1 szó írása

Egy szót ír az addH/addL hálózati címnél

Kérdés	slv	06	addH	addL	vH	vL	crcH	crcL
--------	-----	----	------	------	----	----	------	------

Válasz	slv	06	addH	addL	vH	vL	crcH	crcL
--------	-----	----	------	------	----	----	------	------

## 16. FUNKCIÓ: N szó írása

nbw szót ír, az addH/addL (nbb = 2nbw) hálózati címtől kezdve

Kérdés	slv	10	addH	addL	00	nbw	nbb	vH	vL	...	crcH	crcL
--------	-----	----	------	------	----	-----	-----	----	----	-----	------	------

Válasz	slv	10	addH	addL	00	nbw	crcH	crcL
--------	-----	----	------	------	----	-----	------	------

### Példák:

- 1. Funkció: 15 bitet olvas, a 0x1020 hálózati címtől kezdve, az 1-es kiszolgálón

Kérdés	01	01	10	20	00	0F	79	04
--------	----	----	----	----	----	----	----	----

Válasz	01	01	02	00	12	39	F1
--------	----	----	----	----	----	----	----

Az olvasott érték 0x0012, amely 00000000 00010010-t ad bit mezőként.  
Ebben a példában a 0x1029 és 0x102C változó IGAZ, az összes többi HAMIS.

- 16. Funkció: 2 szót ír a 0x2100 címnél az 1-es kiszolgálón, az írt értékek 0x1234 és 0x5678.

Kérdés	01	10	21	00	00	02	04	12	34	56	78	1C	CA
--------	----	----	----	----	----	----	----	----	----	----	----	----	----

Válasz	01	10	21	00	00	02	4B	F4
--------	----	----	----	----	----	----	----	----

## **Fájl átvitel**

A modern helyi buszokkal összehasonlítva a Modbus protokoll nagyon gyenge szolgáltatásokat nyújt, ha nincs kibővítvé specifikus gyártói funkciókódokkal.

A mi szituációnkban, az ISaGRAF egy erős és flexibilis számítógép alapon történő futtatásakor a Modbus protokollnak két megszorítása van:

- Csak ISaGRAF változókat lehet elérni
- Nehéz végrehajtani nagy mennyiségű adatok nagysebességű átvitelét

Ez az indoka annak, hogy az ISaGRAF egy sor „Modbushoz hasonló” fájl átviteli igényt, vagy egy „távolsi fájlkezelő” protokollt kínál. Ezek a jellemzők az alábbiak érdekében lettek kivitelezve:

- Bináris vagy ASCII fájl letöltés
- Bináris vagy ASCII fájl feltöltés
- Dinamikus adatcsere virtuális vagy fizikai osztott fájlon keresztül

Így tehát, az ISaGRAF kommunikációs kapcsolattal bármely, « az ISaGRAF-tól független » alkalmazás könnyen kommunikálhat egy távoli céllal.

A protokoll az alábbi koncepciókon alapul:

- Az ISaGRAF cél oldalon levő fájl neve **távolsi fájl**
- A gazda számítógépen levő fájl neve **helyi fájl**
- Egy fájlban mindegyik byte egy 32 bites **alapcímmel** plusz egy 16 bites **byte címmel** van elérve

Ígények léteznek a távolsi fájl kiválasztására, az alap cím kiválasztására, a távolsi fájlba történő olvasásra vagy írásra, a 16 bites byte cím használatával.

### 17. FUNKCIÓ: adat írása

az nbb a vH, vL byte-ok számának felel meg

Kérdés	slv	11	addH	addL	00	nbb	nbb	vH	vL	...	crcH	crcL
--------	-----	----	------	------	----	-----	-----	----	----	-----	------	------

Válasz	slv	11	addH	addL	00	nbb	crcH	crcL
--------	-----	----	------	------	----	-----	------	------

Ennek az igénynek a jelentése az addH/addL címtartománynak megfelelően különböző:

#### – **0xF000: Távoli fájlnev inicializálása**

az nbb a fájlnev karaktereinek számának felel meg, amely a vH vL mezőkben van meghatározva (ebben az esetben a Felsőnek és Alsónak nincs jelentősége) és a karaktorsor végéhez a **\0-t tartalmazza**.

ha a fájl nem létezik, akkor az létrehozásra kerül, írható + olvasható + végrehajtható attribútumokkal.

#### – **0xF002: Az alap cím megváltoztatása a megadott értékre**

az nbb-nek 4-el kell egyenlőnek lennie. Az első vH/vL byte a megadott érték Felső szavának felel meg. Bármilyen 32 bites érték elfogadható.

Minden későbbi olvasási vagy írási igény ezt az alap címet fogja használni. Ha ez az igény nincs használva, akkor az alapértelmezett alap cím értéke zéró.

– **0xF004: Fájltörölés**

az nbb-nek zéróval kell egyenlőnek lennie.

A fájl törölve lesz, ha létezik, és ha az lehetséges.

– **Nagyobb mint 0xF004: Fenntartva**

– **Kisebbs mint 0xF000: Byte-ok írása**

A megadott cím, ahová a byte-okat kell írni, az addH/addL-ban van megadva. Ennek F000-tól kisebbnek kell lennie. A megadott byte-ok (nbb byte az vH vL mezőkben meghatározva, ahol a Felső és Alsó többé nem bírhat jelentőséggel) a megadott sorrendben (balról jobbra) az előzőleg kiválasztott távoli fájlnevébe vannak beírva. Az írott kiindulási cím az előzőleg kiválasztott alap címhez adott meghatározott cím. ha az eredményül kapott címek meghaladják a jelenlegi fájl méretet, akkor a fájl kibővíthető. A fájl méretét nem lehet csökkenteni.

## 18. FUNKCIÓ: adat olvasása

Kérdés	slv	12	addH	addL	00	nbb	crcH	crcL
--------	-----	----	------	------	----	-----	------	------

Válasz	slv	12	nbb	V	V	...	crcH	crcL
--------	-----	----	-----	---	---	-----	------	------

A megadott cím, ahová a byte-okat kell olvasni, az addH/addL-ban van megadva. Ennek F000-tól kisebbnek kell lennie. Olvassa a meghatározott (nbb) számú byte-ot az előzőleg kiválasztott távoli fájlnevből, az előzőleg kiválasztott alap címhez adott megadott címtől kezdve (addH/addL bármilyen 16 bites értékkel).

Az értékek előhívása (V mezők balról jobbra) abban a sorrendben történik, ahogy azok a fájlban olvasva vannak.

### Példa:

Távoli fájlnevé kiválasztása: 'target.fil'.

Kérdés	01	11	F0	00	00	0B	0B	74	...	00	25	9F
--------	----	----	----	----	----	----	----	----	-----	----	----	----

Válasz	01	11	F0	00	00	0B	8F	0E
--------	----	----	----	----	----	----	----	----

Alap cím kiválasztása: 0x10000.

Kérdés	01	11	F0	02	00	04	04	00	01	00	00	76	11
--------	----	----	----	----	----	----	----	----	----	----	----	----	----

Válasz	01	11	F0	02	00	04	6E	CA
--------	----	----	----	----	----	----	----	----

4 byte írása: 0x107D0 abszolút cím, 01,02,03,04 értékek.

Kérdés	01	11	07	D0	00	04	04	01	02	03	04	28	6F
--------	----	----	----	----	----	----	----	----	----	----	----	----	----

Válasz	01	11	07	D0	00	04	FC	87
--------	----	----	----	----	----	----	----	----

4 byte Olvasása: 0x107D0 abszolút cím.

Kérdés	01	12	07	D0	00	04	B8	87
--------	----	----	----	----	----	----	----	----

Válasz	01	12	04	01	02	03	04	58	7D
--------	----	----	----	----	----	----	----	----	----

## C.9. Áramkimaradás kezelése

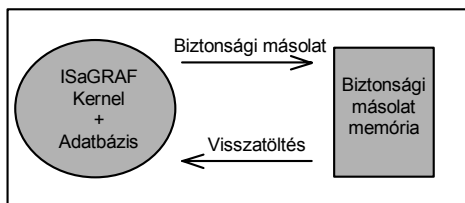
### C.9.1. Alapismeretek

Az áramkimaradás kezelése kritikus fontosságú egy alkalmazásnál, az alábbi három ok miatt:

- Ez a folyamat specifikációjától függ
- Ez a hardver képességeitől függ
- Ez a programozási módszerektől függ

Így tehát az ISaGRAF áramkimaradás kezelési megoldása nem egy teljes és abszolúte univerzális módszer, hanem alapelvek, módszerek, és eszközök együttese, amelyeket minden alkalmazáshoz, vagy legalábbis hardverhez meghatározott módon kell kombinálni.

Annak biztosításához, hogy egy folyamatvezérlő rendszer egy áramkimaradás után megfelelő módon újrainduljon, 3 problémát kell megoldani:



- Egy biztonsági adatmásolat elkészítése
- Az induláskor annak érzékelése, hogy áramkimaradás történt
- A biztonsági másolat adatainak visszatöltése

A második problémának nem lehet standard szoftveres megoldása, de a rendszer szállítója esetleg biztosíthatja az ISaGRAF alkalmazásból vagy egy C programból a hardver állapothoz való hozzáféréshez szükséges eszközöket.

Ezen felül lényeges annak eldöntése, hogy mely adatokat kell eltárolni és visszatölteni. Definíáljunk 2 fajta adatot:

- Alkalmazás változók:  
Mint folyamat változók, pl. a feldolgozott tételek száma, az áramkimaradás dátuma, az alkalmazás paraméterek értékei, stb.  
Mint program változók, pl. számlálók, időzítők, közbenső értékek, és jelzők.
- Program állapot:  
Mint aktív lépések hivatkozása, az egyes C programok státusza, stb.

A következő fejezetben ez a két eset lesz tanulmányozva annak megismertetésére, hogy az ISaGRAF milyen megoldásokat tud nyújtani.

## C.9.2. Alkalmazás-változók biztonsági másolata

### ☐ **Visszatartott változók**

A workbench változószerkesztő lehetővé teszi mindegyik belső változóhoz (nem IO változó) a „visszatartott” attribútum kiválasztását.

Mindegyik célciklus végén a visszatartott változók a meghatározott memóriahelyre vannak másolva. Ez a memóriahely általában egy akkumulátoros háttéráramforrással biztosított RAM.

Ha a beinduláskor legalább egy változó „visszatartott” attribútummal rendelkezik, akkor az ISaGRAF a visszatartott változókat keresni fogja:

- Ha ugyanaz az alkalmazás futott előzőleg, akkor az ISaGRAF felismeri a tárolt értékeket, és azokat minden „visszatartott” változóhoz kijelöli.
- Ha az előző alkalmazás eltérő volt, vagy ha előzőleg nem futott alkalmazás, akkor az ISaGRAF felismeri, hogy a „visszatartott” értékek nem érvényesek, és minden „visszatartott” változót nullára állít vissza.

A különböző típusú változók tárolására szolgáló memóriaterület meghatározása a workbench-ben, a **Létrehozás** menüben van specifikálva: **Alkalmazás futtatási beállítások ; visszatartott változók**.

A megadott karaktersornak a következő formátumúnak kell lennie:

**boo\_add , boo\_size , ana\_add , ana\_size , tmr\_add , tmr\_size , msg\_add , msg\_size**

a következőkkel:

**boo\_add:** Logikai változók tárolásához használt hexadecimális cím. Mindig zérótól eltérőnek kell lennie.

**boo\_size:** Az ezen a címen rendelkezésre álló hexadecimális méret, byte-okban. A tároláshoz logikai változónként egy byte szükséges.

**ana\_add:** Analóg változók tárolásához használt hexadecimális cím. Mindig zérótól eltérőnek kell lennie.

**ana\_size:** Az ezen a címen rendelkezésre álló hexadecimális méret, byte-okban. Minimum négy byte mindig szükséges, plusz tárolandó analóg változónként négy byte.

**tmr\_add:** Időzítő változók tárolásához használt hexadecimális cím. Mindig zérótól eltérőnek kell lennie.

**tmr\_size:** Az ezen a címen rendelkezésre álló hexadecimális méret, byte-okban. A tároláshoz időzítő változónként öt byte szükséges.

**msg\_add:** Üzenet változók tárolásához használt hexadecimális cím. Mindig zérótól eltérőnek kell lennie.

**msg\_size:** Az ezen a címen rendelkezésre álló hexadecimális méret, byte-okban. A tároláshoz üzenet változónként 256 byte szükséges.

#### **Szükségletek:**

- Minden típus minden mezőjét specifikálni kell még akkor is, ha esetleg nincs szükség arra, hogy minden típusú változóról biztonsági másolatot készítsen. Ilyen esetben a nem kívánt

típusú változó(k) a méretéhez zérót (kivéve az analógokat, amelyeknél négy byte-ot), és egy zérótól eltérő címet kell megadnia.

**Példa:**

Tegyük fel, hogy a következőkről biztonsági másolatot kell készíteni:

- 20 Logikai változó
- 0 Analóg változók
- 0 Időzítő változó
- 3 Üzenet változók

A biztonsági másolat memória a 0xA2F200 hexadecimális címen van.

Tegyük fel, hogy:

A logikaiak az 0xA2F200 címen lesznek tárolva, a pontosan szükséges 20 byte mérettel.

A minimum szükséges 4 byte méretű analógok a 0xA2F214 címen lesznek tárolva.

Az időzítők címe 0xA2F200 lesz, és zéró mérettel van specifikálva.

Az üzenetek a logikaiak a 0x A2F218 címen lesznek tárolva, a pontosan szükséges 256\*3 byte mérettel.

Ekkor a workbench-be bevitt karaktersornak a következőnek kell lennie:

A2F200,14,A2F214,4,A2F200,0,A2F218,300
--

## **RENDSZER funkció hívás**

Ha az alkalmazás változók legtöbbjét tárolni kell, akkor a RENDSZER funkció lehetőségeit kell használni egy teljes változókészlet kezelésére (a RENDSZER funkcióval kapcsolatban a kezelési útmutatóban található bővebb információ). Megjegyzendő, hogy itt a biztonsági másolatot és a visszatöltést a programozó az alkalmazás szintjén kezeli.

Mindenekelőtt a biztonsági másolat memória helyét kell definiálnia egy adott típusú változóhoz, vagy az összes változó típushoz:

**<new\_address> := SYSTEM(SYS\_INITxxx,<address>);**

ahol:

- <address> a biztonsági másolat memória címének helye (16# érték a hexadecimális formátumhoz). Ennek páros címnek kell lennie, mert máskülönben a művelet sikertelen lesz.
- a SYS\_INITxxx lehet:
  - \* SYS\_INITBOO az összes logikai változóhoz szóló biztonsági másolat memóriahelyének definiálásához.
  - \* SYS\_INITANA az összes analóg változóhoz szóló biztonsági másolat memóriahelyének definiálásához.
  - \* SYS\_INITTMR az összes időzítő változóhoz szóló biztonsági másolat memóriahelyének definiálásához.
  - \* SYS\_INITALL az összes logikai, analóg, és időzítő változóhoz szóló biztonsági másolat memóriahelyének definiálásához.
- <new\_address> a következő szabad címet hívja elő, ami más szóval az <address> + biztonsági másolatként másolt változók mérete (byte-okban), a SYS\_INITxxx szerint. Ez lehetővé teszi a szükséges biztonsági másolat memória méret ellenőrzését. Ha a művelet sikertelen, akkor a <new\_address> zérót kap.

Ekkor kérhet egy biztonsági másolatot. Ez a procedura az alkalmazásban bármikor hívható, és a biztonsági másolat a pillanatnyi ciklus végén, és csak egyszer lesz elvégezve. Ha a hardver egy logikai inputot vagy egy C funkciót ad át, hogy a felhasználót egy áramkimaradás bekövetkeztéről tájékoztassa, és lehetővé tesz legalább egy ISaGRAF ciklust az összeomlás előtt, akkor a biztonsági másolat esetleg csak az áramkimaradás érzékelésekor készülhet el:

```
<error> :=SYSTEM(SYS_SAVxxx,0);
```

ahol:

- a SYS\_SAVxxx lehet:
  - \* SYS\_SAVBOO az összes logikai változó biztonsági másolatának kérésére.
  - \* SYS\_SAVANA az összes analóg változó biztonsági másolatának kérésére.
  - \* SYS\_SAVTMR az összes időzítés változó biztonsági másolatának kérésére.
  - \* SYS\_SAVALL az összes időzítés változó biztonsági másolatának kérésére.
- <error> egy zérótól eltérő hibastátuszt szerez be, amikor a művelet sikertelen lett (a SYS\_INITxxx nem lett hívva).

Végül a változókat vissza kell tölteni. Ez a procedura az alkalmazásban bármikor hívható, és a visszatöltés a pillanatnyi ciklus végén, és csak egyszer lesz elvégezve. Annak biztosítására, hogy a biztonsági másolattal másolt adatok érvényesek, egy analóg változót egy konstans értékre kell állítani, azt aláírásként használni:

```
<error> := SYSTEM(SYS_RESTxxx,0);
```

ahol:

- a SYS\_RESTxxx lehet:
  - \* SYS\_RESTBOO az összes logikai változó visszatöltésére.
  - \* SYS\_RESTANA az összes analóg változó visszatöltésére.
  - \* SYS\_RESTTMR az összes időzítő változó visszatöltésére.
  - \* SYS\_RESTALL az összes logikai, analóg, és időzítés változó visszatöltésére.
- <error> egy zérótól eltérő hibastátuszt szerez be, amikor a művelet sikertelen lett (a SYS\_INITxxx nem lett hívva).

Az alábbiakban a RENDSZER funkció biztonsági másolat változók kezelésére szolgáló összes parancsa van felsorolva:

parancs		Jelentése
előre definiált kulcsszó	Érték	
SYS_INITBOO	16#20	logikai biztonsági másolat inic.
SYS_SAVBOO	16#21	logikaiak mentése
SYS_RESTBOO	16#22	logikaiak visszatöltése
SYS_INITANA	16#24	analóg biztonsági másolat inic.
SYS_SAVANA	16#25	analógok mentése
SYS_RESTANA	16#26	analógok visszatöltése
SYS_INITTMR	16#28	időzítő biztonsági másolat inic.
SYS_SAVTMR	16#29	időzítők mentése
SYS_RESTTMR	16#2A	időzítők visszatöltése
SYS_INITALL	16#2C	minden típus biztonsági másolat inic.
SYS_SAVALL	16#2D	az összes típus mentése
SYS_RESTALL	16#2E	az összes típus visszatöltése

parancs (előre definiált kulcsszó)	Argumentum	Visszatérési érték
SYS_INITxxx	memória cím	következő szabad cím
SYS_SAVxxx	0	zéró, ha OK
SYS_RESTxxx	0	zéró, ha OK

### ☐ **Testre szabott kivitelezés**

Végül, C funkciókat vagy funkcióblokkokat használva specifikus felhasználói procedúrákat lehet kifejleszteni, egy akkumulátoros háttéráramforrással rendelkező memóriához való hozzáféréshez, az alkalmazásban változók bármikor történő biztonsági másolatához illetve visszatöltéséhez.

#### **Példák:**

##### **1) Egy alkalmazáshoz szánt procedúra:**

a backup, restore\_temp, restore\_date, restore\_cpt C felhasználó procedúra lenne.

**backup**(temperature, date, cnt);                      3 kritikus fontosságú adat tárolása

hőmérséklet := **restore\_temp**();                      hőmérséklet visszatöltése

dátum := **restore\_date**();                              dátum visszatöltése

cnt := **restore\_cnt**();                                  számláló visszatöltése

##### **2) Általános célú procedúrák:**

a backup\_init, backup, backup\_link, restore C felhasználói procedúra lenne.

save\_id := **backup\_init**(address, size);    egy    biztonsági    másolat    memóriatömb  
allokálása.

**backup**(save\_id, cpt1, 3);                      a cpt1 kimentése 3. elemként.

rest\_id := **backup\_link**(address, size)    a biztonsági másolat memória kapcsolása.

cpt1 := **restore**(rest\_id, 3);                      a cpt1 biztonsági másolat értékének  
visszatöltése.

### **C.9.3. Program állapot biztonsági másolat:**

Lehetséges minden alkalmazás program minden állapotának tárolása, de veszélyesnek tűnik minden program visszatöltése abba az állapotba, ahol a legutolsó biztonsági másolatnál volt, legalább 3 ok miatt:

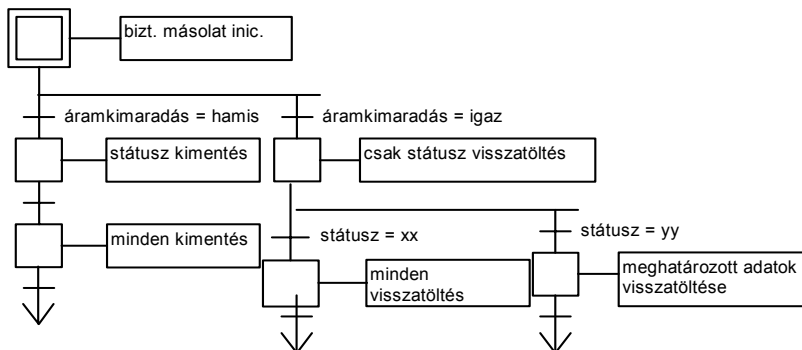
- Bizonyos folyamatok a visszaindítás előtt meghatározott műveletek elvégzését igénylik
- Egy komplett alkalmazás minden státuszával foglalkozni nagyon időigényes
- Bizonyos külső erőforrások, mint pl. C programok, perifériák, stb. nem indíthatók újra automatikusan

Legjobb megoldásnak az analóg vagy logikai változók biztonsági másolatának elkészítése látszik az eljárás státuszának leírására akkor, amikor a programozó úgy véli, hogy a visszaindítási szakaszok ezeket az információkat fel tudják használni.

Ekkor aztán lehetséges a folyamat befejezetlen, de intelligens „leképezéséből” SFC programok elindítása, megszüntetése, vagy befagyasztása, és változók inicializálása annak

érdekében, hogy az alkalmazás a megfelelő állapotba kerüljön. De az ISaGRAF nem tud automatikus beindulási procedúrát kínálni.

**Példa:**



## C.10. Függelék: Hibalista és leírás

### Hibalista:

Kód	Üzenet	Típus
1	nem tud memóriát allokálni a futtatási adatbázishoz	rendszer
2	inkorrekt alkalmazás adatbázis (Motorola/Intel)	alkalmazás
3	nem tud kommunikációs postaládát allokálni	rendszer
4	nem tudja a kernel adatbázist kapcsolni	rendszer
5	a kernelre küldött kérdés időtúllépése	rendszer
6	a kerneltől jövő válaszra várás időtúllépése	rendszer
7	nem tud kommunikációt kezdeményezni	rendszer
8	nem tud memóriát allokálni a visszatartott változókhoz	alkalmazás
9	az alkalmazás megállt	alkalmazás
10	túl sok egyidejű N vagy P művelet	alkalmazás
11	túl sok egyidejű beállítási művelet	alkalmazás
12	túl sok egyidejű visszaállítási művelet	alkalmazás
13	ismeretlen TIC utasítás	alkalmazás
16	nem tud válaszolni az adatolvasási igényre	rendszer
17	nem tud válaszolni az adatírási igényre	rendszer
18	nem tud válaszolni a hibakereső részfeladat igényre	rendszer
19	nem tud válaszolni a modbus igényre	rendszer
20	nem tud válaszolni a hibakereső alkalmazás igényre	rendszer
21	nem tud válaszolni a hibakeresőnek	rendszer
23	ismeretlen igénykód	rendszer
24	Ethernet kommunikációs hiba	rendszer
25	kommunikáció szinkronhiba	rendszer
28	nem tud memóriát allokálni az alkalmazáshoz	rendszer
29	nem tud memóriát allokálni az alkalmazás frissítéshez	rendszer
30	ismeretlen OEM kulcs kód	alkalmazás
31	nem tudja inicializálni a logikai input kártyát	alkalmazás
32	nem tudja inicializálni az analóg input kártyát	alkalmazás
33	nem tudja inicializálni az üzenet input kártyát	alkalmazás
34	nem tudja inicializálni a logikai output kártyát	alkalmazás
35	nem tudja inicializálni az analóg output kártyát	alkalmazás
36	nem tudja inicializálni az üzenet output kártyát	alkalmazás
37	nem tudja inputolni a logikai kártyát	alkalmazás
38	nem tudja inputolni az analóg kártyát	alkalmazás
39	nem tudja inputolni az üzenet kártyát	alkalmazás
40	nem tudja outputolni a logikai output változót	alkalmazás
41	nem tudja outputolni az analóg output változót	alkalmazás
42	nem tudja outputolni az üzenet output változót	alkalmazás
43	nem tudja működtetni a logikai változót	alkalmazás
44	nem tudja működtetni az analóg változót	alkalmazás
45	nem tudja működtetni az üzenet változót	alkalmazás
46	nem tudja megnyitni a kártyát	alkalmazás

47	nem tudja bezárni a kártyát	alkalmazás
50	nem tudja felülrni a logikai output változót	program
51	nem tudja felülrni az analóg output változót	program
52	nem tudja felülrni az üzenet output változót	program
61	ismeretlen rendszer igénykód	program
62	mintázási időtartam	program
63	a felhasználói funkció nincs kivitelezve	alkalmazás
64	integer osztása zéróval	program
65	a konverziós funkció nincs kivitelezve	alkalmazás
66	a funkcióblokk nincs kivitelezve	alkalmazás
67	standard funkció nincs kivitelezve	alkalmazás
68	valós osztása zéróval	program
69	érvénytelen művelet paraméterek	alkalmazás
72	az alkalmazás szimbólumok nem módosíthatók	alkalmazás
73	nem tud frissíteni: eltérő logikai változó készlet	alkalmazás
74	nem tud frissíteni: eltérő analóg változó készlet	alkalmazás
75	nem tud frissíteni: eltérő időzítő változó készlet	alkalmazás
76	nem tud frissíteni: eltérő üzenet változó készlet	alkalmazás
77	nem tud frissíteni: nem talál új alkalmazást	alkalmazás
> 100	specifikus OEM hibakód, a részletekért lépjen kapcsolatba a beszállítóval	

A 3 hibatípus a zavarok különböző forrásának felel meg:

– **Rendszerhibák:**

Az ilyen problémák oka valószínűleg a cél szoftver vagy hardver, és nem az alkalmazás beállítása vagy a programvégrehajtás.

Próbáljon meg a célon egy hard reset-et (az áram kikapcsolását) elvégezni, és próbáljon meg más alkalmazásokat futtatni.

Ezeket a hibákat be kell jelenteni az Önök ISaGRAF támogatási részlege felé.

– **Alkalmazás hibák:**

Az ilyen hibákat az alkalmazás paraméterek, méret, vagy tartalom okozza.

Ezeknek a hibáknak el kell tűnniük, ha egy ismert, és előzőleg érvényesített alkalmazást tölt be. Ha a probléma továbbra is fennáll, akkor az egy fentebb ismertetett rendszerhibává válik.

– **Programhibák:**

Az ilyen hibákat a program egy adott szekvenciája okozza.

Ezeknek a hibáknak el kell tűnniük, amikor az alkalmazást ciklusról-ciklusra módban indítják el, vagy amikor a kritikus program megállt.

**Hibák leírása:**

<b>1. nem tud memóriát allokálni a futtatási adatbázishoz</b>	<b>rendszer</b>
---	-----------------

Nincs rendelkezésre álló memória. Ellenőrizze a hardvert.

**2. inkorrekt alkalmazás adatbázis (Motorola/Intel)****alkalmazás**

A letöltött vagy biztonsági másolatként másolt alkalmazás fájl nem megfelelő. Ez a hiba akkor jelentkezik, ha az alkalmazás INTEL-hez lett generálva és MOTOROLA-hoz lett letöltve (vagy fordítva), illetve ha a fájl módosítva lett.

**3. nem tud kommunikációs postaládát allokálni****rendszer**

Ezt a hibát a kommunikációs feladat állítja elő, ha az nem tudja a 3-as helyet allokálni a feladatközi kommunikációhoz.

**4. nem tudja a kernel adatbázist kapcsolni****rendszer**

Ezt a hibát a kommunikációs feladat állítja elő, ha nem talál egy futó kernelt a parancssorában megadott kiszolgálószámmal.

**5. a kernelre küldött kérdés időtúllépése****rendszer**

A kommunikációs feladat nem tud igényt küldeni a kernel felé. A kernel valószínűleg nem fut, vagy foglalt.

**6. a kerneltől jövő válasza várás időtúllépése****rendszer**

A kommunikációs feladat nem tud választ fogadni a kerneltől. A kernel valószínűleg nem fut, vagy foglalt.

**7. nem tud kommunikációt kezdeményezni****rendszer**

Ez a figyelmeztetés akkor van előállítva, amikor a kommunikációs réteg nem tudja inicializálni a fizikai kapcsolatot. Ez a figyelmeztetés akkor is megjelenítésre kerül, amikor nincs meghatározva kommunikációs útvonal. Ez nem akadályozza meg a cél helyes futását, de az nem tud kommunikálni.

**8. nem tud memóriát allokálni a visszatartott változókhoz****alkalmazás**

Az ISaGRAF nem tudja kezelni a visszatartott változókat. Ennek a problémának két oka lehet:  
- a gazda célhoz paraméterként átadott karaktersor szintaktikailag nem helyes  
- az egyes blokkokhoz megadott memória mérete nem elegendő  
Ellenőriznie kell a „visszatérési paraméter” szintaxisát, és meg kell próbálnia kevesebb számú visszatartott változó alkalmazását.

**9. az alkalmazás megállt****alkalmazás**

Ez a figyelmeztetés mindannyiszor megjelenítésre kerül, ahányszor az alkalmazás a hibakeresőből lett leállítva.

**10. túl sok egyidejű N vagy P művelet****alkalmazás**

Ez a hiba akkor keletkezik, ha az egyik célciklusnak túl sok nem tárolt impulzus műveletet vagy ciklikus blokkot kell végrehajtania. A zavar helye CC módban megkereshető. Az egyidejű műveletek maximális száma SFC programonként 2 + 4.

**11. túl sok egyidejű beállítási művelet****alkalmazás**

Ez a hiba akkor keletkezik, ha az egyik célciklusnak túl sok beállító műveletet kell végrehajtania (amelyek egy lépés aktívvá válásakor vannak végrehajtva). A fent ismertetett módon kell eljárni.

**12. túl sok egyidejű visszaállítási művelet****alkalmazás**

Ez a hiba akkor keletkezik, ha az egyik célciklusnak túl sok visszaállító műveletet kell végrehajtania (amelyek egy lépés deaktiválásakor vannak végrehajtva). A fent ismertetett módon kell eljárni.

**13. ismeretlen TIC utasítás****alkalmazás**

A kernel egy programban valami rendelleneset észlelt az alkalmazás kódban (amelynek neve Cél Független Kód). Ennek két lehetséges magyarázata van:

- egy külső program valószínűleg írást végez az alkalmazás kódba. Próbálja megkeresni az összeomlás helyét CC módban, és ügyeljen rá, hogy egyetlen IO interfésznek se legyen helytelen paramétere.
- az Ön célja csökkentett utasításkészlettel rendelkezik, és az alkalmazás nem megengedett utasítást vagy változótípust használ.

**16. nem tud válaszolni az adatolvasási igényre****rendszer**

Egy kommunikációs hiba lett érzékelve egy specifikus 18-as funkciókódú (fájl olvasás) ISaGRAF Modbus igényre való válaszólkör. Ellenőrizze a csatlakozást és a rendszer konfigurációt mind a cél, mind a gazda oldalon.

**17. nem tud válaszolni az adatírási igényre****rendszer**

Egy kommunikációs hiba lett érzékelve egy specifikus 17-es funkciókódú (fájl írás) ISaGRAF Modbus igényre való válaszólkör. Ellenőrizze a csatlakozást és a rendszer konfigurációt mind a cél, mind a gazda oldalon.

**18. nem tud válaszolni a hibakereső részfeladat igényre****rendszer**

Egy kommunikációs hiba lett érzékelve egy hibakereső igényre való válaszólkör. Ellenőrizze a csatlakozást és a rendszer konfigurációt mind a cél, mind a gazda oldalon.

**19. nem tud válaszolni a modbus igényre****rendszer**

Egy kommunikációs hiba lett érzékelve egy Modbus igényre való válaszólkör. Ellenőrizze a csatlakozást és a rendszer konfigurációt mind a cél, mind a gazda oldalon.

**20. nem tud válaszolni a hibakereső alkalmazás igényre****rendszer**

Egy kommunikációs hiba lett érzékelve egy hibakereső igényre való válaszoláskor. Ellenőrizze a csatlakozást és a rendszer konfigurációt mind a cél, mind a gazda oldalon.

**21. nem tud válaszolni a hibakeresőnek****rendszer**

Egy kommunikációs hiba lett érzékelve egy hibakereső igényre való válaszoláskor. Ellenőrizze a csatlakozást és a rendszer konfigurációt mind a cél, mind a gazda oldalon.

**23. ismeretlen igénykód****rendszer**

Egy hibakereső igénynek nincs értelme.

**24. Ethernet kommunikációs hiba****rendszer**

Ez mindannyiszor bekövetkezik, valahányszor a csatlakozás a hibakereső bezárásakor bezáródik: a rendszer jól működik. Egyébként pedig azt jelenti, hogy egy Ethernet kommunikációs hiba lett érzékelve. Ellenőrizze a csatlakozást és a rendszer konfigurációt mind a cél, mind a gazda oldalon.

Egy második mező is meg van adva, és ez lehet:

- 1: küldés vagy fogadás közbeni hiba
- 2: a csatlakozó létrehozásakor bekövetkező hiba
- 3: a csatlakozó lekötésekor vagy annak figyelésekor bekövetkező hiba
- 4: egy új kliens elfogadásakor bekövetkező hiba

**25. kommunikáció szinkronhiba****rendszer**

A célon levő kommunikációs feladat és a gazda közötti rossz szinkronizálás. Ellenőrizze a csatlakozást és a rendszer konfigurációt (kommunikációs paraméterek) mind a cél, mind a gazda oldalon.

**28. nem tud memóriát allokálni az alkalmazáshoz****rendszer**

Nincs rendelkezésre álló memória. Ellenőrizze a hardvert az alkalmazás méretének megfelelően.

**29. nem tud memóriát allokálni az alkalmazás frissítéshez****rendszer**

Nincs rendelkezésre álló memória. Ellenőrizze a hardvert az alkalmazás méretének megfelelően.

**30. ismeretlen OEM kulcs kód****alkalmazás**

Az alkalmazás egy olyan kártyát használ, amelynek gyártási kódját a cél nem ismeri fel. Ellenőrizze az I/O csatlakozást a workbench-en, és használja a „VIRTUÁLIS” attribútumot a hibás kártya megkeresésére. Lehetséges, hogy az Ön workbench könyvtára nem felel meg a cél verziónak.

**31. nem tudja inicializálni a logikai input kártyát****alkalmazás**

Egy logikai input kártya meghibásodott. Ellenőrizze az I/O csatlakozást a workbench-en, és logikai kártyáinak paramétereit.

**32. nem tudja inicializálni az analóg input kártyát****alkalmazás**

Egy analóg input kártya meghibásodott. Ellenőrizze az I/O csatlakozást a workbench-en, és analóg kártyáinak paramétereit.

**33. nem tudja inicializálni az üzenet input kártyát****alkalmazás**

Egy üzenet input kártya meghibásodott. Ellenőrizze az I/O csatlakozást a workbench-en, és üzenet kártyáinak paramétereit.

**34. nem tudja inicializálni a logikai output kártyát****alkalmazás**

Egy logikai output kártya meghibásodott. Ellenőrizze az I/O csatlakozást a workbench-en, és logikai output kártyáinak paramétereit.

**35. nem tudja inicializálni az analóg output kártyát****alkalmazás**

Egy analóg output kártya meghibásodott. Ellenőrizze az I/O csatlakozást a workbench-en, és analóg output kártyáinak paramétereit.

**36. nem tudja inicializálni az üzenet output kártyát****alkalmazás**

Egy üzenet output kártya inic. meghibásodott. Ellenőrizze az I/O csatlakozást a workbench-en, és üzenet output kártyáinak paramétereit.

**37. nem tudja inputolni a logikai kártyát****alkalmazás**

Egy logikai inputkártya frissítése közben hiba lett érzékelve. Ellenőrizze az I/O csatlakozást a workbench-en, és kártyáinak a paramétereit.

**38. nem tudja inputolni az analóg kártyát****alkalmazás**

Egy analóg inputkártya frissítése közben hiba lett érzékelve. Ellenőrizze az I/O csatlakozást a workbench-en, valamint kártyáinak a paramétereit.

**39. nem tudja inputolni az üzenet kártyát****alkalmazás**

Egy üzenet inputkártya frissítése közben hiba lett érzékelve. Ellenőrizze az I/O csatlakozást a workbench-en, és kártyáinak a paramétereit.

**40. nem tudja outputolni a logikai output változót****alkalmazás**

Egy logikai outputkártya frissítése közben hiba lett érzékelve. Ellenőrizze az I/O csatlakozást a workbench-en, és kártyáinak a paramétereit.

**41. nem tudja outputolni az analóg output változót****alkalmazás**

Egy analóg outputkártya frissítése közben hiba lett érzékelve. Ellenőrizze az I/O csatlakozást a workbench-en, és kártyáinak a paramétereit.

**42. nem tudja outputolni az üzenet output változót****alkalmazás**

Egy üzenet outputkártya frissítése közben hiba lett érzékelve. Ellenőrizze az I/O csatlakozást a workbench-en, és kártyáinak a paramétereit.

**43. nem tudja működtetni a logikai output változót****alkalmazás**

Egy logikai változón egy MŰVELET hívás végrehajtása közben hiba lett érzékelve. Ellenőrizze a MŰVELET paramétereket és a kártya felhasználói megjegyzését.

**44. nem tudja működtetni az analóg változót****alkalmazás**

Egy analóg változón egy MŰVELET hívás végrehajtása közben hiba lett érzékelve. Ellenőrizze a MŰVELET paramétereket és a kártya felhasználói megjegyzését.

**45. nem tudja működtetni az üzenet változót****alkalmazás**

Egy üzenet változón egy MŰVELET hívás végrehajtása közben hiba lett érzékelve. Ellenőrizze a MŰVELET paramétereket és a kártya felhasználói megjegyzését.

**46. nem tudja megnyitni a kártyát****alkalmazás**

Az alkalmazás egy olyan kártyahivatkozást használ, amely ismeretlen a cél előtt. Ellenőrizze az I/O csatlakozást a workbench-en. Lehetséges, hogy az Ön workbench könyvtára nem felel meg a cél verzióknak.

**47. nem tudja bezárni a kártyát****alkalmazás**

Az alkalmazás egy olyan kártyahivatkozást használ, amely ismeretlen a cél előtt. Ellenőrizze az I/O csatlakozást a workbench-en.

**50. nem tudja felülírni a logikai output változót****program**

Két SFC szekvencia ugyanazt a logikai output változót írja ugyanabban a célciklusban. Ezt meg kell akadályozni, hogy elkerülhető legyen az I6O-k veszélyes viselkedése. Egy ilyen konfliktus esetén a hierarchiában legmagasabb programnak kell prioritást adni. Ha a két SFC program ugyanazon a szinten van, akkor annak kiszámíthatatlan az eredménye.

**51. nem tudja felülrni az analóg output változót****program**

Két SFC szekvencia ugyanazt az analóg output változót írja ugyanabban a célciklusban. Lásd a fenti megjegyzést.

**52. nem tudja felülrni az üzenet output változót****program**

Két SFC szekvencia ugyanazt az üzenet output változót írja ugyanabban a célciklusban. Lásd a fenti megjegyzést.

**61. ismeretlen rendszer igénykód****program**

Egy program érvénytelen kóddal használja a RENDSZER hívást.

**62. mintázási időtartam****program**

A célciklus periódusa hosszabb, mint ami a workbench-ben van specifikálva.

Egy multitaszking rendszeren ez azt jelenti, hogy nincs elegendő CPU idő egy ciklus végrehajtásához, még akkor is, ha a „pillanatnyi ciklus időtartam” kevesebb, mint a specifikált időtartam.

Egy egyfeladatos rendszeren ez mindig azt jelenti, hogy túl sok művelet van az egyik célciklusban.

Ennek a figyelmeztetésnek az eltávolítására számos lehetséges megoldás létezik:

- csökkentse a figyelmeztetés észlelésének pillanatában végrehajtott műveletek számát.
- csökkentse a vezérjelek és érvényes átmenetek számát, és optimalizálja a komplex feldolgozást, stb.
- csökkentse a más feladatok általi CPU megterhelést, hogy több időt adjon az ISaGRAF-nak.
- csökkentse a kommunikációs forgalmat, hogy több időt adjon az ISaGRAF-nak.
- használjon dinamikus ciklusidőtartam módosítást, a különböző folyamat stádiumokhoz a ciklusidőtartam adaptálásához.
- állítsa a ciklusidőtartamot zéróra, hogy az ISaGRAF kernel a lehető leggyorsabban fusson, bármilyen túlsordulás ellenőrzés nélkül.

**63. a felhasználói funkció nincs kivitelezve****alkalmazás**

Az alkalmazás egy olyan C funkciót használ, amely ismeretlen a célban. Lehetséges, hogy az Ön workbench könyvtára nem felel meg a cél verzióknak.

**64. integer osztása zéróval****program**

Egy program egy integert zéróval próbál osztani. Az alkalmazásnak az ilyes esetet meg kell akadályoznia, mert annak előre megjósolhatatlan hatásai lehetnek.

Amikor ez történik, az ISaGRAF a maximális analóg értéket teszi eredményként.

Ha az operandus negatív, akkor az eredmény invertálódik.

**65. a konverziós funkció nincs kivitelezve****alkalmazás**

Az alkalmazás egy olyan C konverziós funkciót használ, amely ismeretlen a célban. Lehetséges, hogy az Ön workbench könyvtára nem felel meg a cél verziónak. Amikor ez történik, az ISaGRAF nem konvertálja az értéket.

**66. a funkcióblokk nincs kivitelezve****alkalmazás**

Az alkalmazás egy olyan C funkcióblokkot használ, amely ismeretlen a célban. Lehetséges, hogy az Ön workbench könyvtára nem felel meg a cél verziónak.

**67. standard funkció nincs kivitelezve****alkalmazás**

Az alkalmazás egy olyan funkcióblokkot használ, amely ismeretlen a célban, bár annak a legtöbb célban rendelkezésre kell állnia. Kérdezze meg szállítóját.

**68. valós osztása zéróval****program**

Egy program egy valós analógot zéróval próbál osztani. Az alkalmazásnak az ilyes esetet meg kell akadályoznia, mert annak előre megjósolhatatlan hatásai lehetnek. Amikor ez történik, az ISaGRAF a maximális valós analóg értéket teszi eredményként. Ha az operandus negatív, akkor az eredmény invertálódik.

**69. érvénytelen művelet paraméterek****alkalmazás**

Az alkalmazás rossz paraméterekkel használ egy MŰVELET hívást. Ezt normál esetben a fordító kiszűri. Ez lehet egy időzítő paraméter, vagy egy változó, amely sem input, sem output.

**72. az alkalmazás szimbólumok nem módosíthatók****alkalmazás**

Egy alkalmazás frissítés megkísérlésekor a módosított alkalmazás nem indítható el, mert a szimbólumok különbözőek. Egy vagy több változó vagy funkcióblokk előfordulás lehetett a pillanatnyi alkalmazáshoz képest hozzáadva, eltávolítva, vagy módosítva.

**73. nem tud frissíteni: eltérő logikai változó készlet****alkalmazás**

A módosított alkalmazás nem indítható el, mert bizonyos logikai változó lett a pillanatnyi alkalmazáshoz képest hozzáadva, eltávolítva, vagy módosítva.

**74. nem tud frissíteni: eltérő analóg változó készlet****alkalmazás**

A módosított alkalmazás nem indítható el, mert bizonyos analóg változók lettek a pillanatnyi alkalmazáshoz képest hozzáadva, eltávolítva, vagy módosítva.

**75. nem tud frissíteni: eltérő időzítő változó készlet*****alkalmazás***

A módosított alkalmazás nem indítható el, mert bizonyos időzítő változók lettek a pillanatnyi alkalmazáshoz képest hozzáadva, eltávolítva, vagy módosítva.

**76. nem tud frissíteni: eltérő üzenet változó készlet*****alkalmazás***

A módosított alkalmazás nem indítható el, mert bizonyos üzenet változók lettek a pillanatnyi alkalmazáshoz képest hozzáadva, eltávolítva, vagy módosítva.

**77. nem tud frissíteni: nem talál új alkalmazást*****alkalmazás***

A módosított alkalmazás nem található a memóriában, valami hiba történhetett a letöltés közben.



## D Szószedet

<b>Alprogram</b>	Egy SFC-n kívüli bármilyen más nyelven írt program, amelyet egy másik, tulajdonos programjának nevezett program hív meg.
<b>Analóg</b>	Változók típusa. Ezek folyamatos integer, vagy valós változók.
<b>Apa SFC program</b>	Egy SFC program, amely más SFC programokat (amelyeket gyermekének neveznek) vezérel.
<b>Áramsín</b>	Egy létradiagram két szélén levő fő bal- és jobb oldali függőleges sín.
<b>Átmenet</b>	Alap SFC grafikus komponens. Egy átmenet különböző SFC lépések közötti állapotot képviseli. Egy átmenetre egy számmal történik hivatkozás. Minden átmenethez egy logikai feltétel van csatolva.
<b>Attribútum ...</b>	Változók osztálya. A rendelkezésre álló attribútumok: belső, input vagy output.
<b>Az SFC 1. szintje</b>	Egy SFC program fő leírása. Az 1. szint a diagramot (lépéseket és átmeneteket) valamint a csatolt megjegyzéseket csoportosítja.
<b>Az SFC 2. szintje</b>	Egy SFC program részletes leírása. Ez a lépéseken belüli műveletek és az átmenetekhez csatolt logikai állapotok leírása.
<b>Azonosító</b>	Egyedi szó, ami a programozásban egy változót vagy egy konstans kifejezést képvisel.
<b>Befejezés szekció</b>	Az egyes célciklusok végén végrehajtott ciklikus programok csoportja.
<b>Befejező lépés</b>	Egy SFC makró lépés fő részének utolsó lépése. Egy befejező lépés nincs kapcsolva semmilyen későbbi átmenethez sem.
<b>Belső</b>	Egy változó attribútuma, amely nincs egy input- vagy output eszközhöz kapcsolva.
<b>Blokkdiagram</b>	Egy áramlás tervezéséhez használt grafikus nyelv. A diagram az áramlásban a különböző útvonalak kiválasztását lehetővé tevő végrehajtandó műveletekből és döntésekből áll. A Blokkdiagram nyelv lehetővé teszi egymás utáni ciklusokban végrehajtandó hurkok megrajzolását.
<b>C forrás fejsor</b>	Szöveges fájl, amely egy C funkció vagy egy konverziós funkció programozásához szükséges „C” definíciókat és típusokat tartalmazza.
<b>C forráskód</b>	Szöveges fájl, amely egy funkció vagy egy konverziós funkció „C” forráskódját tartalmazza.
<b>C funkció</b>	A „C” nyelvben írt funkció, mely az ISaGRAF programokból (amelyek más nyelven vannak megírva) szinkronizált módon kerül meghívásra. A C

	funkciókat a ICS Triplex ISaGRAF szállítja, vagy azokat a felhasználó fejleszti ki.
<b>C nyelv</b>	Magas szintű literális nyelv, amely a számítógép műveletek, pl. C funkciók és konverziós funkciók leírására használatos.
<b>Cél</b>	Az ISaGRAF cél gép, amely az ISaGRAF kernel szoftvert támogatja.
<b>Célciklus</b>	Az ISaGRAF célrendszer aktiválásakor minden alkalommal végrehajtott műveletek készlete. A ciklusokat programozható ciklusidőzítés indítja el.
<b>Cella</b>	A grafikus mátrix elemi területe grafikus nyelvekhez, pl. SFC-hez, FBD-hez vagy LD-hez.
<b>Ciklikus</b>	Egy program attribútuma, amely mindig végre van hajtva.
<b>Ciklusidőzítés</b>	A cél végrehajtási ciklus időtartama.
<b>Ciklustól-ciklusig mód</b>	Végrehajtási mód: Ebben a módban a ciklusok egyesével, a hibakereső felhasználója által adott parancsoknak megfelelően kerülnek végrehajtásra.
<b>Címke (IL)</b>	Egy IL utasítássor elejére elhelyezett azonosító, amely az utasítást azonosítja, és a JMP műveletekhez operandusként használható.
<b>Csatlakozó (FC)</b>	FC grafikus komponens, amely egy, a blokkdiagram egyik pontja és egy FC művelet vagy teszt közötti kapcsolatot képvisel. Egy ugrás grafikus szimbóluma egy kis kör, amely a rendeltetési elem hivatkozási számával van ellátva.
<b>Definiált szó</b>	Egyedi azonosító, amely egy programban bármely kifejezés lecserélésére szolgál.
<b>Döntés (FC)</b>	(Tesztnek is nevezik). Egy logikai kifejezéshez csatolt blokkdiagram szimbólum. Az áramlás vagy IGEN, vagy NEM szimbólum outputhoz van irányítva, a kifejezés állapotától függően.
<b>Egy átmenet érvényesítése</b>	Egy változó attribútuma. Egy átmenet akkor van érvényesítve, amikor az összes megelőző lépés aktív.
<b>Egy átmenet tisztázása</b>	Futtatási művelet: valamennyi, a megelőző lépésekben létező vezérlőjelek eltávolításra kerülnek. Minden ezután lépésben egy vezérlőjel van létrehozva.
<b>Egy lépés aktivitása</b>	Egy lépés attribútuma, amelyet egy SFC vezérlőjel jelez. A lépéshez csatolt műveletek annak tevékenységének megfelelően van végrehajtva.
<b>Él</b>	Egy logikai változó változása. A felfutó él hamisról igazra történő változást jelent. A lefutó él igazról hamisra történő változást jelent.
<b>Elválasztó</b>	Egy literális nyelvben az azonosítók elválasztására használt speciális karakter (vagy karaktercsoport). Egy elválasztó képviselhet egy műveletet.

<b>Eszköztár</b>	Egy grafikus szerkesztőeszköz ablakának kis melléklapja, amely a grafikus komponensek kiválasztásának fő gombjait csoportosítja.
<b>FBD ...</b>	Funkcionális blokkdiagram nyelv.
<b>FC ...</b>	Jelentése: „Blokkdiagram”
<b>Felső szintű program</b>	A hierarchikus faszerkezet tetejére helyezett program. Egy felső szintű programot a rendszer aktivál.
<b>Feltétel (egy átmenethez)</b>	Egy SFC átmenethez csatolt logikai kifejezés. Az átmenet nem tisztázható, ha annak állapota hamis.
<b>Funkcióblokk</b>	Az FBD nyelv grafikus komponense, amely az ISaGRAF könyvtárakból egy standard elemi funkciót képvisel.
<b>Funkcionális blokkdiagram</b>	Grafikus nyelv: az egyenletek standard elemi blokkokból vannak felépítve az ISaGRAF könyvtárból, és azok a diagramban egymáshoz vannak kapcsolva.
<b>Futtatási hiba</b>	Az ISaGRAF célrendszer által a futtatáskor érzékelt alkalmazási hiba.
<b>Globális</b>	Változók vagy definiált szavak tartománya. Az ilyen tárgyak egy projekt bármelyik programjában használhatók.
<b>Gyermek SFC program</b>	Egy másik SFC program által (amelyet apának neveznek) vezérelt SFC program.
<b>Hálózati cím</b>	Opcionális hexadecimális cím, mindegyik változóhoz szabadon definiálva. Ezt a címet a Modbus protokoll használja, amikor a célrendszer egy külső rendszerhez van kötve.
<b>Hierarchia</b>	Egy projekt architektúrája, több programra osztva. A hierarchikus faszerkezet az apa programok és a gyermek programok közötti kapcsolatokat képviseli.
<b>Hivatkozási szám (SFC)</b>	Decimális szám (1-től 65535-ig), amely egy SFC lépést vagy átmenetet azonosít egy SFC programban.
<b>I/O csatlakozás</b>	Az alkalmazás változói és a célrendszeren létező kártyák csatornái közötti kapcsolatok leírása.
<b>I/O csatorna</b>	Egy I/O kártya egyszeres csatlakozási pontja. Egy I/O csatorna egy változót fogadhat.
<b>I/O kártya</b>	Hardver erőforrás. Egy I/O kártyát egy típus és egy irány (input vagy output) jellemez. Egy I/O kártya paraméterei az ISaGRAF könyvtárban vannak leírva.
<b>I/O változó</b>	Egy input- vagy outputeszközhöz csatlakoztatott változó. Egy I/O változónak

egy I/O kártya csatornájához kell csatlakoznia.

<b>Időzítő</b>	Változók típusa. Az ilyen változók időértékeket tartalmaznak, és a futtatáskor automatikusan frissíthetők az ISaGRAF rendszer által.
<b>IL ...</b>	Utasításlista nyelv.
<b>Impulzus művelet</b>	SFC művelet: ez utasítások listája, amelyek csak egyszer vannak végrehajtva, amikor a megfelelő lépés aktiválódik.
<b>Input ...</b>	Egy változó attribútuma. Az ilyen változók egy inputeszközhöz vannak kapcsolva.
<b>Integer</b>	Analog változók osztálya, amely egy előjeles integer 32 bites formátumban van tárolva.
<b>Karakter sor</b>	Egy üzenet változóban tárolt karakterek készlete.
<b>Kereszthivatkozások</b>	Az ISaGRAF workbench által a változók szótáráról, valamint azoknak a változóknak egy projektben történő használatáról számított információ.
<b>Késleltetett művelet (IL)</b>	Egy IL program művelete, amely a "(" utasításnak a programban később bekövetkező előfordulásakor van végrehajtva.
<b>Kezdeti helyzet</b>	Egy SFC program kezdeti lépéseinek készlete, amely a program kontextusát képviseli annak elindulásakor.
<b>Kezdeti lépés</b>	Egy makró lépés fő részének első lépése. Egy kezdeti lépés nincs kapcsolva semmilyen megelőző átmenethez sem.
<b>Kezdeti lépés</b>	Egy SFC program speciális lépése, amely a program elindulásakor aktiválódik.
<b>Kifejezés</b>	Operátorok és azonosítók együttese, amelyek egy érték kiértékelését képviselik.
<b>Konstans kifejezés</b>	Egy konstans érték leírására szolgáló literális kifejezés. Egy konstans kifejezés egy típushoz van szánva.
<b>Kontakt ...</b>	Egy LD program grafikus komponense. Ez egy input változó státuszát képviseli.
<b>Konverzió</b>	Egy input vagy output analóg változóhoz csatolt szűrő. A konverzió automatikusan alkalmazásra kerül valahányszor a változó input vagy output.
<b>Konverziós funkció</b>	„C”-ben írt funkció, amely egy konverziót ír le. Egy ilyen konverzióbármelyik analóg input vagy output analóg változóhoz csatlakozhat.
<b>Konverziós táblázat</b>	Pontok együttese, amelyek egy lineáris (szegmensenkénti) konverziót definiálnak. Ilyen konverzió bármelyik input vagy output analóg változóra alkalmazható.

<b>Könyvtár ...</b>	Hardver- vagy szoftver források együttese, amelyek közvetlenül beilleszthetők bármelyik alkalmazásba.
<b>Közös</b>	Definiált szavak tartománya Az ilyen tárgyak bármelyik projekt bármelyik programjában használhatók.
<b>Kulcsszó</b>	A nyelv fenntartott szava.
<b>LD ...</b>	Létradiagram nyelv.
<b>Lépés</b>	Az SFC nyelv alap grafikus komponense. Egy lépés a folyamat stabil szituációját képviseli, és az egy négyzetként van megrajzolva. Egy lépésre egy számmal történik hivatkozás. Egy lépés aktivitása a megfelelő műveletek végrehajtásának vezérlésére szolgál.
<b>Létradiagram</b>	Kontaktokat és tekerceket keverő grafikus nyelv logikai egyenletek tervezéséhez.
<b>Logikai ...</b>	Változók típusa. Az ilyen változók csak igaz vagy hamis értéket vehetnek fel.
<b>Logikai művelet</b>	SFC művelet: Egy logikai változó egy lépés aktivitási jelével van kijelölve.
<b>Lokális</b>	Változók vagy definiált szavak tartománya. Az ilyen tárgyak csak egy projekt egy programjában használhatók.
<b>Makró lépés</b>	SFC grafikus komponens. Egy makró lépés lépések és átmenetek egyedi csoportja, amelyeket egy egyedi szimbólum képvisel a fő diagramon, de amelyek külön-külön vannak leírva.
<b>Márix</b>	A szerkesztő terület logikai felosztása négyzetes cellákra egy grafikus nyelvű program szerkesztése során.
<b>Megjegyzés</b>	Egy programban levő szöveg, amelynek nincs hatása a program végrehajtására.
<b>Megjegyzés (SFC)</b>	Egy SFC lépéshez vagy átmenethez csatolt szöveg, amelynek nincs hatása a program végrehajtására.
<b>Megszakítási pont ...</b>	A felhasználó által a hibakeresés során egy SFC lépésre vagy átmenetre elhelyezett jel. A célrendszer megáll, amikor egy SFC vezérlőjel egy megszakítási pontra van áthelyezve.
<b>Modbus</b>	Gazda-kiszolgáló protokoll. Egy ISaGRAF célrendszer egy komplett architektúrában lehet egy Modbus kiszolgáló egy külső rendszerrel (pl. felügyelő rendszerekkel) való kapcsolathoz.
<b>Módosító (IL)</b>	Egy IL műveleti kulcsszó végére tett egyetlen karakter, amely módosítja a művelet jelentését.
<b>Műszaki</b>	Kezelési útmutató az ISaGRAF könyvtárak egy eleméhez (C funkció vagy funkcióblokk, konverziós funkció vagy I/O kártya). A műszaki megjegyzést az

<b>megjegyzés</b>	elem tervezője írja meg.
<b>Művelet</b>	Utasítások és kijelölések listája, amikor egy SFC program lépése aktív.
<b>Művelet (FC)</b>	Egy Blokkdiagram szimbóluma. Egy művelet utasítások listáját jelenti, amelyeket akkor kell végrehajtani, amikor a dinamikus áramlás egy művelet szimbólumhoz ér.
<b>Művelet (IL)</b>	Az IL nyelv alap utasítása. Egy művelet általában egy utasítás operandusával van társítva.
<b>Napló</b>	Szövegfájl, amely egy programban elvégzett változások összes megjegyzéseit tartalmazza. Mindegyik megjegyzést szerkesztésének dátuma egészíti ki.
<b>Nem tárolt művelet</b>	SFC művelet: ez utasítások listája, amelyek mindegyik célciklusban vannak végrehajtva, amikor a megfelelő lépés aktív.
<b>OEM kulcs kód (I/O kártya)</b>	Hexadecimális 16 bites kód, az ISaGRAF könyvtár minden I/O kártyájához definiálva. Az OEM kód a kártya beszállítóját azonosítja.
<b>OEM paraméter (I/O kártya)</b>	I/O kártya paraméter, amelyet a kártya tervezője definiál. Ez lehet egy konstans, vagy egy változó paraméter, amelyet a felhasználó az I/O csatlakozás során ad meg.
<b>Operandus (IL)</b>	Egy elemi IL utasítás által feldolgozott változó vagy konstans kifejezés.
<b>Output</b>	Egy változó attribútuma. Az ilyen változók a célgép egy output eszközéhez vannak kapcsolva.
<b>Paraméter (C funkció)</b>	Egy „C” funkciónak inputként adott érték. Egy paramétert egy típus jellemez.
<b>Paraméter (I/O kártya)</b>	Egy standard I/O kártya felhasználó által definiált, vagy standard paramétere. Egy felhasználó által definiált paramétert a felhasználó ír be az I/O csatlakozás során.
<b>Pillanatnyi eredmény (IL)</b>	Egy utasítás eredménye egy IL programban. A pillanatnyi eredmény egy utasítással módosítható, vagy egy változó beállítására használható.
<b>Program ...</b>	Egy projekt alap programozási egysége. Egy program egy nyelvel van leírva, és az a projekt hierarchikus architektúrájában van elhelyezve.
<b>Projekt</b>	Programozási terület, amely egy ISaGRAF alkalmazás összes információját (programok, változók, célkód, stb.) csoportosítja.
<b>Regiszter (IL)</b>	Egy IL szekvencia pillanatnyi eredménye.
<b>Retesztelt I/O</b>	Input vagy output változó, a felhasználó által a hibakeresőből küldött „Reteszelés” paranccsal logikailag leválasztva a megfelelő I/O eszköztől.

<b>SFC ...</b>	Szekvenciális funkciódiagram nyelv.
<b>ST</b>	Strukturált szöveg nyelv.
<b>Strukturált szöveg</b>	Magas szintű strukturált literális nyelv, amely kijelöléseket, magas szintű struktúrákat, (pl. Ha/Akkor/Máskülönben), és funkció meghívásokat kombinál.
<b>Szekció ...</b>	Ugyanazokkal a dinamikus szabályokkal végrehajtott programok csoportja.
<b>Szekció kezdete</b>	Az egyes célciklusok kezdetén végrehajtott ciklikus programok csoportja.
<b>Szekvenciális funkciódiagram</b>	Grafikus nyelv: A folyamatot lépések készleteként van definiálva, amelyeket átmenetek kapcsolnak egymáshoz. A lépésekhez műveletek vannak csatolva. Az átmenetek logikai állapotokként vannak részletezve.
<b>Szekvenciális szekció</b>	Egy projekt programjainak csoportja. Azoknak a programoknak a végrehajtása az SFC nyelv dinamikus szabályait követi.
<b>Szótár</b>	Belső, input, vagy output változók, illetve definiált szavak készlete, amely egy projekt programjaiban van használva.
<b>Szülő program</b>	Egy bármilyen nyelven írt program, amely más nem SFC programot (amelyet annak alprogramjának neveznek) vezérel (hív meg).
<b>Tartomány</b>	Programok készlete, amelyek egy tárgyat használhatnak. Az előre definiált ISaGRAF tartományok: közös, globális, és lokális.
<b>Tekercs ...</b>	Egy LD program grafikus komponense, amely egy output érték kijelölését képviseli.
<b>Teszt (FC)</b>	(Döntésnek is nevezik). Egy logikai kifejezéshez csatolt blokkdiagram szimbólum. Az áramlás vagy IGEN, vagy NEM szimbólum outputhoz van irányítva, a kifejezés állapotától függően.
<b>Típus</b>	Ugyanolyan formátumú változók osztálya. Az alaptípusok: logikai, analóg, időzítő, és üzenet.
<b>Ugrás egy lépésre</b>	SFC grafikus komponens, amely egy kapcsolatot képvisel egy átmenettől egy lépéshez. Egy ugrás grafikus szimbóluma egy nyíl, amely a rendeltetési lépés hivatkozási számával van ellátva.
<b>Utasítás</b>	Alap ST komplett művelet.
<b>Utasítás ...</b>	Egy IL program elemi művelete, egyetlen sornyi szöveggént beírva.
<b>Utasításlista</b>	Alacsony szintű literális nyelv, amely elemi műveletek szekvenciális listájaként van beírva.
<b>Üzenet</b>	Változó típus. Az ilyen változók változtatható hosszúságú karaktersorokból állnak.

<b>Valódi kártya</b>	A célgépen egy I/O eszközhöz fizikailag csatlakoztatott I/O kártya.
<b>Valós</b>	Analóg változók osztálya, amely egy lebegő IEEE egyszeres pontosságú 32 bites formátumban van tárolva.
<b>Valós idejű mód</b>	Futtatási normál végrehajtási mód: a végrehajtási ciklusokat a programozott ciklusidőzítés indítja el.
<b>Változó</b>	Elemi adatok egyedi képviselője, amelyek egy projekt programjaiban vannak feldolgozva.
<b>Vezérlőjel (SFC)</b>	Grafikus jelző egy SFC program aktív lépéseinek jelzésére.
<b>Virtuális kártya</b>	Egy I/O kártya, amely nincs fizikailag csatlakoztatva egy I/O eszközhöz a célgépen.
<b>Visszatérési érték (egy alprogramé)</b>	Egy alprogram által annak végrehajtása végén visszaadott érték. A visszatérési érték a tulajdonos program műveleteiben van felhasználva.

## E Általános tárgymutató

-, 251  
 \$ szekvencia, 182  
 %, 91, 183  
 &, 248  
 ) művelet (IL), 243  
 \*, 252  
 /, 252  
 :=, 137  
 := (ST kijelölés), 227  
 +, 250  
 <, 256  
 <=, 257  
 <>, 260  
 =, 259  
 =1, 249  
 >, 257  
 >=, 258  
 >=1, 249  
 1 nyereség, 246  
 2. szint, 50  
 2. szintű, 39

### A

A decimális rész levágása, 289  
 A program nyomtatása, 73  
 A projekt megnyitása, 21  
 A projektleíró szerkesztése, 21  
 ABS, 285  
 Abszolút érték, 285  
 ACOS, 289  
 Aktiválás, 110  
 Aktivitás időtartam, 233  
 Aktivitási időtartam, 189  
 Alkalmazás mérete, 371  
 Alkalmazás mérhető, 334  
 Al-karakter sor beszúrás, 306  
 Al-karakter sor keresés, 305  
 Al-karakter sor kivonás (balra), 307  
 Al-karakter sor kivonás (jobb), 310

Al-karakter sor kivonás (középső), 308  
 Al-karakter sor lecserélés, 309  
 Al-karakter sor törlés, 304  
 Alkönyvtár, 164  
 Álnév, 59  
 Alprogram, 25, 175, 197, **200**, 205, **211**, 433  
 Alprogram meghívás (ST), 224  
 Alprogram meghívás IL-ben, 243  
 ANA, 261  
 Analóg, 180, 181, 184, 374, 375, 433  
 AND, 248  
 AND\_MASK, 253  
 AnyTarget, 104  
 Apa SFC program, 201, 433  
 appli.tst, 332, 343, 355, 364  
 appli.x6m, 342, 354  
 appli.x8m, 332, 364  
 Áramlás, 47, 203, 206, 207  
 Áramsín, 53, 54, 61, 212, 433  
 Arc tangens, 291  
 Archiválás, 23  
 Archivum, 144, 151, 159  
 Archivum meghajtó, 152  
 Archivum fájl, 153  
 ARCREATE, 312  
 Argumentum, 148  
 Árkusz koszinusz, 289  
 Árkusz szinus, 290  
 ARREAD, 313  
 ARWRITE, 314  
 ASCII, 303  
 ASIN, 290  
 ATAN, 291  
 Átmenet, 34, 39, 113, 189, 433  
 Átszámolás, 39, 49  
**Attribútum, 433**  
 Átviteli sebesség, 33  
 AVERAGE, 279  
**Az SFC 1. szintje, 188, 189, 433**

**Az SFC 2. szintje, 193, 433**

**Azonosító, 433**

## B

Beállítás tekercs, 217

Beépített forráskód, 128

Befejezés, 24, 203

**Befejezés szekció, 433**

Befejező lépés, 38, 193, **433**

**Bekapcsolt átmenet, 201**

**Belső, 433**

Bináris kiválasztó, 302

BinaryFile, 102

Bit mező, 123

Bitmap, 122

Biztonsági másolat, 23, 144, 151, 152, 159

Biztonsági másolat fájl egység (VxWorks), 346, 349

BLINK, 283

Blokkdiagram, 45, 203, **433**

Blokkdiagram szerkesztő, 45

BOO, 260

**BY, 231**

## C

C forrás fejsor, 376, 382, 390, 402, **433**

C forráskód, 377, 383, 391, 402, **433**

C funkció, 149, 373, 379, **433**

C funkcióblokk, 149, 373

C kód, 100, 144

C nyelv, 373, 376, 377, 382, 390, 391, 402, **434**

C programfordító, 373, 402

**CAL operátor (IL), 244**

**CASE, 229**

Cat, 264

Cél, 98, **434**

Cél architektúra, 329

**Cél ciklus, 434**

**Cella, 434**

CHAR, 303

Ciklikus, 174, **434**

Ciklus, 138, 174, 178, 328

Ciklus profilozó, 133

Ciklus vége vezérlés (VxWorks), 347, 350

Ciklusidő, 333, 343, 355, 366

Ciklusidőzítés, 30, 111, 133, 265, 375, 380, 387, **434**

Ciklustól-ciklusig, 30, 111

**Ciklustól-ciklusig mód, 434**

Címke, 62, 139, **209**, 220

**Címke (IL), 238, 434**

CLKRATE, 346

CMP, 277

COS, 291

Csatlakozás, 63, 64, 65

Csatlakozó, 48, 206, **434**

Csatorna, 90, 91, 146, 160

Csatorna megjegyzés, 90

Csoport, 13, 22, 124

CTD, 272

CTU, 271

CTUD, 273

## D

DAY\_TIME, 311

DDE, 117

DDE (NT cél), 364, 368, 371

**Decimális, 181**

Definiált szó, 76, 80, 186, **434**

Deklarálás, 27, 76

DELETE, 304

DERIVATE, 282

Diagnózis, 130

Differenciálás, 282

Divergencia, 35, 37, 38, 190

Divergencia (FBD), 209

**DO, 230, 231**

Dokumentum, 21, 31, 154

Döntés, 45, 49, 50, 204, **434**

## E

**Egy átmenet érvényessége, 434**  
**Egy átmenet tisztázása, 201, 434**  
 Egy ciklus végrehajtása, 111  
**Egy lépés aktivítása, 188, 189, 201, 233, 434**  
 Egy makró lépés fő része, 193  
 Egyenlő, 259  
**Él, 434**  
 Él kontakt, 215  
 Előfordulás, 77, 80  
 Előzmények, 21, 31  
**ELSE, 229**  
**ELSIF, 229**  
 Eltolás balra, 295  
 Eltolás jobbra, 295  
**Elválasztó, 223, 434**  
 EN, 54  
**END\_CASE, 229**  
**END\_FOR, 231**  
**END\_IF, 229**  
**END\_REPEAT, 231**  
**END\_WHILE, 230**  
 ENO, 54  
 EPROM, 342, 354  
 EQ operátor (IL), 259  
 Érintés, 30, 97  
 Erőforrás, 30, 101  
 Erőforrás definíciós fájl, 101  
 Eszközök menü, 31  
**Eszköztár, 435**  
 Ethernet, 33  
**EXIT, 232**  
 Export, 83  
 EXPT, 286

## F

F\_CLOSE, 316  
 F\_EOF, 317  
 F\_OPEN, 315  
 F\_TRIG, 270

F\_WOPEN, 315  
 FA\_READ, 318  
 FA\_WRITE, 320  
 Fájl  
     fájl vége érzékelése, 317  
 Fájl beillesztése, 69  
 Fájl bezárása, 316  
 Fájl írása, 320, 324  
 Fájl megnyitás, 315  
 Fájl olvasás, 318, 322  
 FBD, 61, **208**, 380, 388, **435**  
 FBD áthelyezése, 64  
 FBD beillesztése, 65, 66  
 FBD elem beillesztése, 63  
 FBD elem kiválasztása, 63  
 FBD kivágása, 65  
 FBD másolása, 65  
 FBD megjegyzés, 64  
 FBD szerkesztő, 61, 71  
 FBD törlése, 65  
 FC, 45, 203, **435**  
 FC alprogram, 25, 205  
 FC áthelyezése, 48  
 FC átméretezése, 48  
 FC beillesztése, 49  
 FC csatlakozó, 48  
 FC elem beillesztése, 46  
 FC elem kiválasztása, 47  
 FC kapcsolat, 47  
 FC kivágása, 49  
 FC másolása, 49  
 FC megjegyzés, 47  
 FC szerkesztő, 45  
 FC törlése, 49  
**FEDGE, 226**  
 Fel számláló, 271  
 Felső szintű, 24  
**Felső szintű program, 435**  
 Feltétel, 204  
**Feltétel (egy átmenethez), 199, 435**  
 Feltöltés, 127  
 Feltöltés (beállítások), 128  
 Feltöltés (előkészítés), 128  
 Fényszóró, 122  
 Fényszóró áthelyezése, 124

Fényszóró átméretezése, 124  
 Fényszóró kiválasztása, 124  
 FIND, 305  
 FM\_READ, 322  
 FM\_WRITE, 324  
 Font, 157  
**FOR, 231**  
 Fordítás, 97, 143, 148  
 Fordítási beállítás, 30, 128  
 Fordítási beállítások, 98  
 Forgatás balra, 293  
 Forgatás jobbra, 294  
 Forráskód, 144  
 Frissítés, 111  
 From, 104  
 Function block, 143, 177  
 Funkció, 24, 27, 143, 147, 175  
 Funkció exportálás, 29  
 Funkció importálás, 29  
 Funkció meghívás IL-ben, 243  
 Funkcióblokk, 24, 27, 54, 62, 66, 77, 80,  
**208, 225, 387, 435**  
 Funkcióblokk előfordulás, 387  
 Funkcióblokk exportálás, 29  
 Funkcióblokk importálás, 29  
 Funkcióblokk meghívás IL/ben, 244  
 Funkcióhívás (ST), 224  
**Funkcionális blokkdiagram, 208, 435**  
 Futtatás, 30  
 Futtatási hiba, 30, 112, 265, **435**

## G

GE operátor (IL), 258  
 GFREEZE, 202, **236**  
 GKILL, 202, **235**  
**Globális, 182, 435**  
 Görbe, 122, 123, 126  
 Grafikus, 122, 126  
 GRST, 202, **236**  
 GSTART, 202, **235**  
 GSTATUS, 202, **236**  
 GT operátor (IL), 257  
 Gyermek, 25, 175  
**Gyermek SFC program, 201, 435**

Gyors LD, 43, 51, 53  
 Gyors LD szerkesztő, 71  
 Gyűjtemény, 43

## H

**Ha, 140**  
 HA, 206  
 Hálózati cím, 78, 80, 83, **435**  
 HAMIS, 80, 180  
 Háttérkép, 122  
 Hatványszámítás, 287  
 Hiba, 101  
 Hibakeresés, 32  
 Hibakereső, 109, 130  
 Hibakereső munkaterület, 32  
 Hibaüzenet, 75  
 Hierarchia, 24, 28, 174, **201, 435**  
 Hiszterézis, 280, 281  
**Hivatkozási szám, 188, 189, 190, 193,**  
**435**  
 Hozzáférfési jog, 158  
 HYSTER, 280

## I

I/O, 30, 88, 89, 90, 108, 112, 131, 144,  
 145, 146, 160, 161  
 I/O csatlakozás, 88, **435**  
**I/O csatorna, 435**  
 I/O csatorna MŰKÖDTETÉS, 266  
 I/O kártya, 146, **435**  
 I/O konfiguráció, 144  
 I/O összekapcsolás, 30  
 I/O specifikus, 204, 205  
 I/O változó, 374, 375, **435**  
 Időegység, 181  
 IdőNyomtatás, 138  
 Időtúllépés, 33  
 Időzítő, 80, 185, **436**  
**IF, 229**  
 IGAZ, 80, 180  
 Ikon, 123  
 Ikonok, 13  
 IL, 69, 121, 198, **199, 238, 436**

IL szerkesztő, 71  
 Import, 83  
 Impulzus, 41  
 Impulzus időzítés, 276  
 Impulzus művelet, 195, **436**  
 Input, 88, 108, 131, 133, 146, 178, **436**  
 INSERT, 306  
 Integer, 80, 180, **436**  
 Integer analógok tömbje, 278  
 INTEGRAL, 281  
 Interfész, 27, 148  
 Invertált kontakt, 215  
 Invertált tekercs, 216  
 ISA feladat (OS9), 335  
 ISA.EXE, 330  
 ISA.O (VxWorks), 345, 346  
 isa\_main, 347, 350  
 isa\_register\_slave, 346  
 ISAGRAF.INI (NT cél), 357  
 ISAKER feladat (OS9), 336  
 ISAKERET.O (VxWorks), 345, 348  
 ISAKERSE.O (VxWorks), 345, 348  
 ISAMOD (VxWorks), 345  
 ISAMOD.EXE, 330  
 ISANET feladat (OS9), 338  
 ISASSR.O (VxWorks), 345  
 ISATST feladat (OS9), 337  
 ISAx0, 341  
 ISAx1, 332, 341, 342  
 ISAx1 (NT cél), 363  
 ISAx1 (VxWorks), 353  
 ISAx2, 342  
 ISAx3, 342  
 ISAx4, 342  
 ISAx5, 342  
 ISAx6, 332, 341, 342  
 ISAx6 (NT cél), 364  
 ISAx6 (VxWorks), 354  
 ISMÉTLÉS, 206

## J

Jelgenerátor, 284  
 Jelszó, 21, 92, 143, 158

## JMP operátor (IL), 241

Jump, 209, 220

## K

Kapcsolat, 33, 63, 64, 65, 112, 127, 164, 203, 206, 207

### Kapcsolat (FBD), 209

### Kapcsolat (LD), 212

Kapcsolat (SFC), 189

Karaktersor, 181, **436**

Karaktersor hossza, 308

Kártya, 88, 89

Kártya áthelyezése, 89

Kártya paraméter, 90, 146

Kártya típus, 89

Kártya törlése, 89

Kártyahely, 89, 91

Kártyahely beillesztése, 89

Kémlelés, 119, 121, 122

Keresés, 39, 49, 59, 65, 69, 75

Kereszthivatkozás, 107

Kereszthivatkozások, 30, **436**

Késleltetés időzítés, 274

Késleltetésen kívüli időzítés, 275

Késleltetett művelet (IL), 239, 243, **436**

Kezds, 24, 110, 203

**Kezdeti helyzet, 189, 201, 436**

**Kezdeti lépés, 189, 193, 201, 436**

Kezdő lépés, 36, 38

Kezelőpanel, 109

**Kifejezés, 436**

Kijelölés, 246

**Kijelölés (ST-ben,**

**=), 227**

**Kikapcsolt átmenet, 201**

Kilépés billentyű (a célon), 333

Kilépés billentyű (NT cél), 367

Kireteszelés, 112

Kisebb mint, 256

Kisebb mint, vagy egyenlő, 257

Kiszolgáló Kapcsolat, 352

Kiszolgáló szám, 330, 336, 337, 338,

346, 349, 358, 368, 371

Kitevő, 286

Kivonás, 251  
 Kódgenerálás, 29, 97  
 Kommunikáció, 33, 112, 127, 164, 330, 335, 337, 338, 340, 345, 350, 358, 368, 371  
 Kommunikációs logikai szám, 337, 338, 349  
 Komplex I/O berendezés, 145  
 Konstans kifejezés, 180, **436**  
 Kontakt, 53, 62, **214, 436**  
 Kontakt beillesztése, 56  
 Kontakt invertált, 215  
 Kontakt közvetlen, 214  
 Kontakt negatív él, 215  
 Kontakt Pozitív él, 215  
 Kontakt típus, 58  
 Konvergencia, 35, 37, 38, 190  
 Konvertálás időzítőre, 263  
 Konvertálás integerre, 261  
 Konvertálás logikaira, 260  
 Konvertálás üzenetre, 264  
 Konvertálás valósra, 262  
 Konverzió, 94, **436**  
 Konverzió ASCII -> karakter, 303  
 Konverzió karakter -> ASCII, 303  
 Konverziós funkció, 150, 373, 374, **436**  
 Konverziós táblázat, 94, 95, **436**  
 Könyvtár, 23, 29, 89, 90, 107, 132, 142, 151, 373, **437**  
 Könyvtár átnevezése, 143  
 Könyvtár másolása, 143  
 Könyvtár törlése, 143  
 Könyvtárrendező, 142, 373, 375, 380, 388  
 Koszusz, 291  
 Közös, 151, **437**  
 Közvetlen kontakt, 214  
 Közvetlen tekercs, 216  
 Közvetlenül képviselt változó, 91, 183  
 Kulcsszó, 182, 239, **437**

## L

LD, 43, 51, 53, 61, 212, **437**  
 LD beillesztése, 58

LD kivágása, 58  
 LD másolása, 58  
**LD operátor (IL), 240**  
 LD szerkesztő, 53  
 LD törlése, 58  
 LE operátor (IL), 257  
 Le számláló, 272  
 Lecserélés, 39, 49, 59, 65, 69  
 LEFT, 307  
 Leíró, 20, 30  
 Lemez, 12  
 Lépés, 34, 39, 113, 188, **437**  
 Letöltés, 110  
 Létradiagram, 212, **437**  
 Létrafok, 53, 54, 58, 64  
 Létrafok beillesztése, 58  
 Létrafok címke, 56  
 Létrafok megjegyzés, 56, 59  
 Létrehozás, 29, 97  
 LIM\_ALRM, 281  
 LIMIT, 297  
 Lista kimentése, 119  
 LOG, 287  
 Logaritmus, 287  
 Logikai, 80, 184, **437**  
 Logikai művelet, 42, **194, 437**  
 Lokális, 148, 182, **437**  
 LT operátor (IL), 256

## M

Makró lépés, 36, 38, 192, **437**  
 Másik program, 73  
 Maszk az integer biteken (és), 253  
 Maszk az integer biteken (nem), 255  
 Maszk az integer biteken (vagy), 254  
 Maszk az integer biteken (xor), 255  
**Mátrix, 437**  
 MAX, 297  
 Maximum, 297  
 Megállítás, 110  
 Megerősítés, 29, 97, 148  
 Megjegyzés, 186, 206, **223, 238, 437**  
**Megjegyzés (SFC), 188, 189, 437**  
 Megszakítási pont, 111, 113, **437**

Memória, 12  
 Metafile, 122  
 MID, 308  
 MIKÖZBEN, 206  
 MIN, 296  
 Minimum, 296  
 MLEN, 308  
 MOD, 298  
**Modbus, 437**  
 MODBUS, 83, 409  
 Módosítások nyomon követése, 67  
 Módosító (IL), 238, 239, **437**  
 Módosított stílus, 67  
 Modulo, 298  
 MSG, 264  
 MŰKÖDTETÉS I/O csatorna, 266  
 Multi-alkalmazások, 341, 353, 363  
 Multiplexer 4 tétellel, 299  
 Multiplexer 8 tétellel, 300  
 Műszaki megjegyzés, 90, 143, 374, 375, 380, 388, **438**  
 Művelet, 45, 50, **193**, 198, 204, 205, **438**  
**Művelet (IL), 238, 239, 438**  
 MUX4, 299  
 MUX8, 300

## N

N minősítő, 41  
 Nagyobb mint, 257  
 Nagyobb mint, vagy egyenlő, 258  
 Napló, 27, **438**  
 NE operátor (IL), 260  
 NEG, 247  
 Negatív kontakt, 215  
 Negatív tekercs, 219  
 Négyzetgyök, 288  
 NEM, 63, 64  
 Nem egyenlő, 260  
 Nem tárolt, 41  
 Nem tárolt művelet, 196, **438**  
 Normál stílus, 67  
 NOT\_MASK, 255  
 NT (védelmi kulcs), 15  
 Nyelv, 25, 178

Nyereség 1, 246  
 Nyomtatás, 21, 31, 79, **137**, 154, 156

## O

ODD, 301  
 OEM kulcs kód, 146, **438**  
 OEM paraméter, 146  
**OEM paraméter (I/O kártya), 438**  
**OF, 229**  
 Oldal, 156  
 On Line, 32, 109  
 Online módosítás, 111, 114  
**Operandus (IL), 238, 239, 438**  
 Optimalizáló, 99  
 OR, 249  
 OR\_MASK, 254  
 OS-9 shell, 344  
 Összeadás, 250  
 Összehasonlítás, 277  
 Oszlopdiagram, 122  
 Osztályozás, 79  
 Osztás, 252  
 Output, 88, 108, 131, 146, 178, **438**  
 Output ablak, 75

## P

P minősítő, 41  
 P0 minősítő, 41  
 P1 minősítő, 41  
 Paraméter, 27, 148  
 Paraméter (C funkció, 381  
**Paraméter (C funkció), 438**  
 Paraméter (funkcióblokk), 389  
**Paraméter (I/O kártya), 438**  
 Paritásvizsgálat, 301  
 Paritás, 33  
**Pillanatnyi eredmény (IL), 238, 239, 438**  
 Pont, 94, 95  
 POW, 287  
 Pozitív kontakt, 215  
 Pozitív tekercs, 218  
 Prioritás, 368

Prioritási szint (NT cél), 362  
 Program, 24, 71, 133, 174, **438**  
 Program áthelyezése, 28  
 Program másolás, 28  
 Program megjegyzés, 26  
 Program megnyitása, 27, 108  
 Program szintaxis, 71  
 Program törlés, 28  
 Programfordítás, 29  
 Programfordító üzenet, 101  
 Programkezelő, 24  
 Projekt, 20, 151, **438**  
 Projekt áthelyezése, 20  
 Projekt dokumentum, 21, 31, 154  
 Projekt elválasztók, 20  
 Projekt lista, 20, 22  
 Projektcsoport, 22  
 Projektkezelő, 20  
 Projektleíró, 21, 30  
 PROM, 342, 354

## R

**R (visszaállítás) operátor (IL), 241**  
 R\_TRIG, 269  
 Rács, 55  
 RAND, 301  
 REAL, 262  
**REDGE, 226**  
**Regiszter (IL), 238, 438**  
 RENDSZER funkció, 418  
 Rendszeróra sebesség (VxWorks), 346  
**REPEAT, 231**  
 REPLACE, 309  
 RET operátor (IL), 242  
 Reteszelés, 112, 161  
**Reteszelt I/O, 438**  
**RETURN, 209, 220, 228**  
 RIGHT, 310  
 ROL, 293  
 ROR, 294  
 RS, 268

## S

S (készlet) operátor (IL), 241  
 Sarok, 63  
 SEL, 302  
 SEMA, 271  
 SFC, 34, 99, 113, 156, 188, 380, **439**  
 SFC áthelyezése, 39  
 SFC beillesztése, 38  
 SFC evolúciós szabályok, 200  
 SFC gyermek, 25, 42, 175  
 SFC gyűjtemény, 43  
 SFC kivágása, 38  
 SFC másolása, 38  
SFC szerkesztő, 34, 71  
 SFC törlése, 38  
 SHL, 295  
 SHR, 295  
 SIG\_GEN, 284  
 SIN, 292  
 Soros kapcsolat, 33  
 SQRT, 288  
 SR, 267  
 SSR[x][1].szóköz, 354  
 ST, 43, 69, 121, 223, 380, 388, **439**  
**ST operátor (IL), 240**  
 ST szerkesztő, 71  
 STACKINT, 278  
 Stílus, 67, 125  
**Strukturált szöveg, 223, 439**  
 SYSTEM, 265  
 Számláló felfelé/lefelé, 273  
 Szekció, 24, **439**  
**Szekció kezdete, 439**  
 Szekvenciális, 24, 174, 188  
 Szekvenciális funkciódiaagram, 188, **439**  
**Szekvenciális szekció, 439**  
 SZEMAFOR, 271  
 Szétválasztás, 124  
 Szimbólumok (alkalmazás  
     szimbólumok), 332  
 Szimbólumtáblázat, 166  
 Szimulátor, 31, 131, 133, 134, 375, 380,  
     387  
 Szinus, 292

Szkript, 134, 136  
 Szolga szám, 33  
 Szorzás, 252  
 Szótár, 27, 71, 76, 107, 148, 375, 387, **439**  
 Szöveg beillesztése, 69  
 Szöveg kivágása, 69  
 Szöveg másolása, 69  
 Szöveg törlése, 69  
 Szöveges megjelenítés, 122  
 Szövegszerkesztő, 69  
**Szülő program, 439**

## T

Tagadás, 247  
**Tagadás (FBD), 210**  
 Tagadásos kapcsolat, 63, 64  
 TAN, 293  
 Tangens, 293  
 Target, 103  
 Tárkiírás, 120  
 Tartalomjegyzék, 154  
 Tartomány, 76, 78, **439**  
 Tekercs, 53, 62, **214, 439**  
 Tekercs beállítás, 217  
 Tekercs beillesztése, 56  
 Tekercs invertált, 216  
 Tekercs közvetlen, 216  
 Tekercs negatív, 219  
 Tekercs pozitív, 218  
 Tekercs típus, 58  
 Tekercs visszaállítás, 218  
 Telepítés, 12  
 Terminál mód, 344  
 Teszt, 45, 49, 50, 109, 131, 204, **439**  
 TextFile, 103  
**THEN, 229**  
 Típus, 76, 78, 88, 107, 146, 180, **439**  
 TMR, 263  
 To, 104  
**TO, 231**  
 TOF, 275  
 Tömb írás, 314  
 Tömb létrehozás, 312

Tömb olvasás, 313  
 Tömörítés, 152  
 TON, 274  
 Törölt stílus, 67  
 TP, 276  
 TRUNC, 289  
 TSK\_FUNIT, 346, 349  
 TSK\_NBTCKSCHED, 347, 350, 355  
 tst\_main\_ex, 350  
**TSTART, 234**  
**TSTOP, 234**  
**Tudományos, 181**

## U

Ugrás, 39, 49, 54, 62, 69, 140  
 Ugrás egy lépésre, 36, 190, **439**  
 Új funkció, 26  
 Új funkcióblokk, 26  
 Új könyvtárelem, 142  
 Új létrafok, 55  
 Új program, 26  
 Új projekt, 20  
 Új változó, 79  
 ULongData, 102  
**UNTIL, 231**  
**Utasítás, 223, 238, 439**  
**Utasításlista, 238, 439**  
 Üzenet, 80, 120, 181, 185, **439**  
 Üzenet hossza, 308  
 Üzenet összekapcsolás, 264  
 Üzenetek hozzáadása, 264

## V

VAGY, 61  
 Valódi kártya, 89, **440**  
 Valós, 80, 181, **440**  
 Valós idejű, 30, 111  
**Valós idejű mód, 440**  
 Változó, 27, 41, 57, 63, 66, 69, 76, 107, 108, 113, 148, 160, 182, **208, 440**  
 Változó beillesztése, 41, 79  
 Változó kémlése, 119  
 Változó kivágása, 79

Változó másolása, 79  
Változó módosítása, 79  
Változó név, 182  
Változók listája, 119, 121, 124  
Várakozás, 139  
VarList, 102  
Védelem, 21, 92, 143, 158  
Védelmi kulcs, 14  
Védelmi szint, 158  
Vége, 141  
Végrehajtási sorrend, 66  
Véletlenszerű szám, 301  
Verziószám, 110  
**Vezérlőjel (SFC), 188, 440**  
Virtuális kártya, 89, 161, **440**  
Virtuális kártyák (szimuláció NT céllal),  
371  
Virtuális kártyák (szimuláció NT-vel),  
361, 368  
Visszatartott, 417

Visszatérés, 54, 63  
**Visszatérési érték, 440**  
Visszatöltés, 23, 144, 151, 152, 159

## W

**WHILE, 230**  
WISAKER.EXE (NT), 357

## X

XOR, 249  
XOR\_MASK, 255

## Z

**Zárójel, 224, 239**  
Zoom, 52, 59, 67