HADMÉRNÖK

V. Évfolyam 2. szám - 2010. június

Deák Ferenc
deak@nct.hu

LÉTRADIAGRAM FORDÍTÓK ELMÉLETE PLC VEZÉRLÉSEK SZÁMÁRA III.

Absztrakt

A létradiagram egyszerű, programozási képzettséggel nem rendelkező szakemberek számára is érthető nyelv, ennek köszönhető, hogy az elmúlt 20-25 évben az egyik legnépszerűbb ipari programozási formává vált. Bár létradiagramos PLC (Programmable Logic Controller) rengeteg létezik a piacon, az ezeket feldolgozó fordítóprogramok irodalma meglehetősen szegényes. Az alábbi cikk az NCT új vezérléscsaládja számára kidolgozott PLC fordító elméleti hátterét mutatja be. Ez az algoritmus szakít az irodalom által ismertetett megközelítéssel, és gráfelméleti oldalról közelíti meg a kérdést, ami egy jól áttekinthető, szemléletes többlépcsős megoldást eredményez.

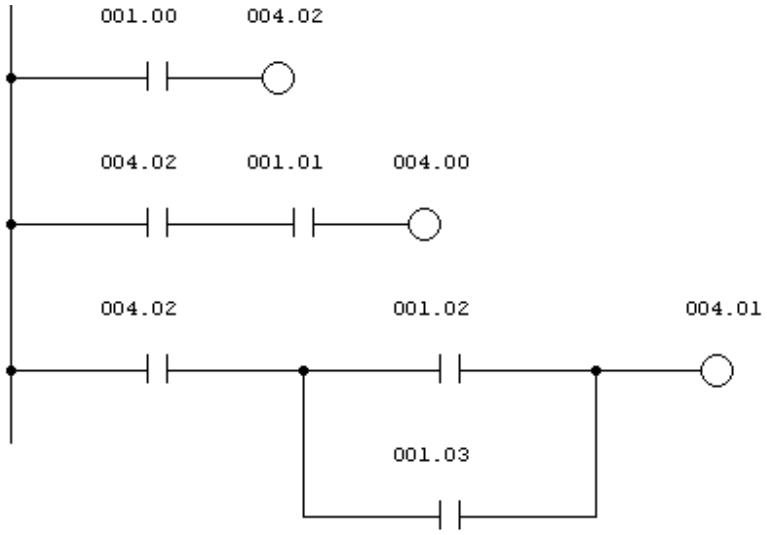
Relay ladder logic has become one of the most popular discrete control programming systems in the last 20-25 years. Programmable Logic Controllers (PLCs) usually can be programmed by wiring up relay contacts and coils on screen. This virtual circuit is transformed into a list of instructions in sequence. In this paper, the translation theory of relay ladder logic for new generation NCT controllers is examined. In contrast to solutions accessible in the literature this algorithm is multiphase, expressive and based on graph theory.

Kulcsszavak: *PLC, Létradiagramm, fordító ~ PLC, Programmable logic controllers, relay ladder logic, compiler*

BEVEZETÉS

Az előző két részben a fordítási algoritmust ismertetettük. Most két olyan esetről lesz szó, amivel az általános algoritmus megértését nem kívántuk elbonyolítani. Különösen fontos a számlálók esete, mivel a kereskedelemben kapható PLC-kkel ellentétben, ez az algoritmus lehetővé teszi, hogy számláló ne csak a létrafok jobboldalán állhasson, hanem bárhol, és kimenetét feltételként szabadon fel lehessen használni.

SPECIÁLIS ESETEK: IDEIGLENES RELÉK KIBONTÁSA



1.ábra. PLC program, melyben ideiglenes relé alkalmazása javasolt

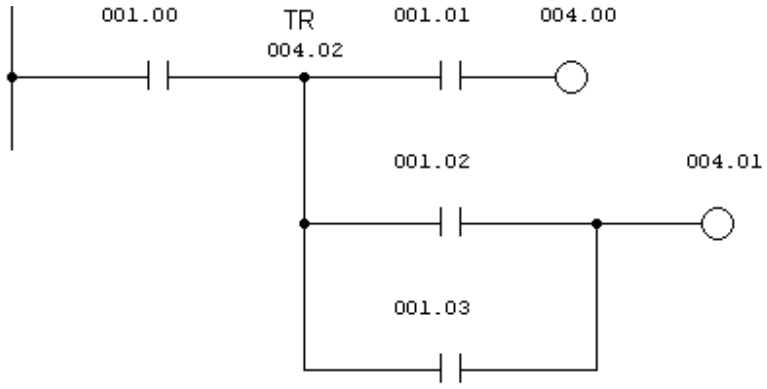
Ha az érintkezőket a diagramban vezetékdarabok felhasználásával sorosan illetve párhuzamosan kötjük, akkor köztük ÉS illetve vagy kapcsolatokat hozhatunk létre, így összetett logikai kifejezéseket valósíthatunk meg.

Ha valamilyen logikai kifejezés eredményét többször fel szeretnénk használni a PLC programban, akkor két dolgot tehetünk:

a) A logikai kifejezést annyi példányban megismételjük, ahányszor szükségünk van rá. Ez két problémát vet fel. Egyrészt, ha módosítani, akarjuk, akkor az összes előfordulását módosítani kell, aminek elmulasztása hibákhoz vezethet, másrészt a kifejezés értékét többször ki kell számítanunk. Másrészt a kifejezést minden előfordulásakor újra meg újra ki kell értékelnie a PLC-nek, ami nem gazdaságos
b) Létrehozunk egy ideiglenes relét. A diagramban (1.ábra) a többször felhasznált kifejezést a 001.00 érintkező szimbolizálja. (Az 1.ábra egy meglehetősen egyszerű esetet mutat be,

az ideiglenes relé alkalmazása természetesen csak akkor éri meg, ha a 001.00 érintkező helyén valamilyen bonyolultabb kifejezés áll.) Az ideiglenes relé a 004.02-es relé, amelyet igazi kimenetként nem használunk fel, egyetlen célja, hogy további kifejezésekben szerepeljen.

Az fenti ábrán (1.ábra) szereplő kapcsolást egyszerűbben is rajzolhatjuk, ezt mutatja a 2. ábra, ahol az ideiglenes relét egy vezetékdarabban rejtjük el, és TR jellel jelöljük.

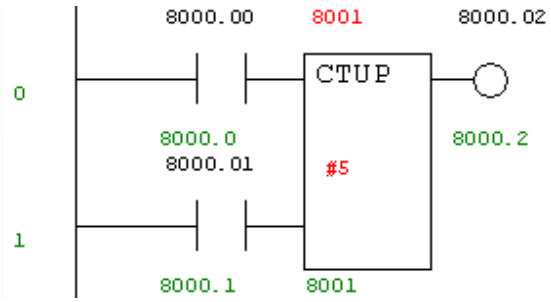


2. ábra. Ideiglenes relé alkalmazása

Az ideiglenes relé alkalmazása azonban a fordító algoritmus módosítását igényli. Az új algoritmus a következő:

1) Létrafokokra bontás.
2) Gráf reprezentációra alakítás
3) Ideiglenes relék kifejtése. Ennek során a 2. ábrának megfelelő gráfot átalakítja a PLC fordító az 1. ábrának megfelelő gráffá. (Erre a pontra egy példa a „Speciális esetek: számlálók" fejezetben található).
4) Mivel az előző lépésben egy létrafokból több keletkezik, ezért a létrafokokra bontást meg kell ismételni. A létrafokokra bontás megegyezik az első lépésben alkalmazott módszerrel, mely egy korábbi fejezetben került ismertetésre, azzal a különbséggel, hogy itt a létrafokokra bontás már a gráf reprezentáción történik.
5) A fordítás többi lépése megegyezik a korábban leírtakkal.

SPECIÁLIS ESETEK: SZÁMLÁLÓK



3. ábra. Számláló a létradiagramban

Az cikkben bemutatott számlálónak baloldalon egy léptető, és egy reset bemenete van. Kimenete a számláló típusától és állapotától függően aktív, vagy nem aktív. A kimenet opcionális, amennyiben nincs kimenete, akkor ezt a fordító pótolja egy a PLC program írója számára láthatatlan „nyelő”- vel.

A számlálók belső működése a fordítási algoritmus szempontjából érdektelen, viszont abban lényegesen eltérnek a többi létradiagram-elemtől, hogy a több helyen is szerepelnek a mátrixban (mivel több lábuk van a baloldalon). Ezt úgy lehet elérni, hogy a mátrixban nem maguk a létradiagram elemek szerepelnek, hanem csak referenciák melyek a létradiagram elemre mutatnak. Ezt a megoldást mutatja be a 1. és 2. táblázat.

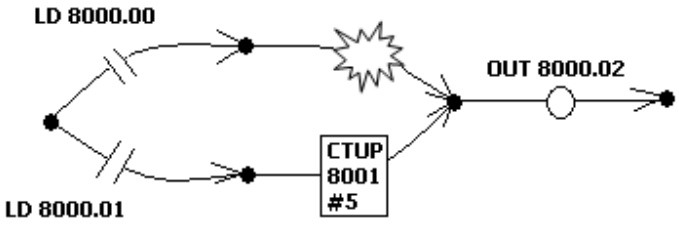
Azonosító	**Típus**	**Cím**	**Index**
1	Számláló	8001	#5
2	Érintkező	8000.00	-
3	Érintkező	8000.01	-
4	T-elágazás	-	-
5	T-elágazás	-	-
6	Tekercs	8000.02	-

1. táblázat. Létradiagram elemek felsorolása

4	2	1	6
5	3	1	-

2. táblázat. Létradiagram elemek a mátrixban

Az ebből fordított gráf alakot mutatja a 4. ábra.



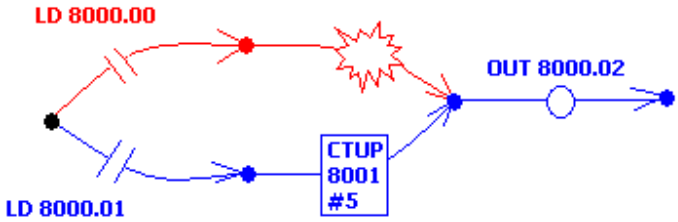
4. ábra. Számláló gráf alakja

A számláló első előfordulását egy speciális – itt „CSILLAG”-gal jelölt – élnek fordítjuk, a második előfordulás maga a számláló. A „CSILLAG”-nál a felső, a második előfordulásnál az alsó lábat vesszük figyelembe a baloldalon. A jobboldali láb mindkét esetben ugyanarra az OUT élhez csatlakozik.

A „CSILLAG”-ot az OUT-hoz és az ideiglenes relékhez hasonlóan ún. jobboldali elemnek tekintjük. A „CSILLAG”-ra és az OUT-ra alkalmazzuk a „Speciális esetek: ideiglenes relék kibontása” fejezetben leírt algoritmus 3. pontját a következőképpen:

1. Kezdetben minden elem színtelen.
2. Előbb a „CSILLAG”, majd az OUT mint jobboldali elemből kiindulva színezünk.
3. Választunk egy eddig még nem használt színt, és nyilakkal ellentétes irányban haladva beszínezzük a még nem színes éleket.

A lépés után a gráfot a 5. ábra mutatja.



5. ábra. A gráf a másodlagos létrafokokra bontás után

A különböző színű másodlagos létrafokokat egymástól függetlenül alakítjuk utasításlistává: a 1. mnemonic kód az első, az 2. mnemonic-kód a második létrefokot mutatja.

Egy összefésülési lépésben a CTUP –AND LD-ot CTUP-ra cseréljük, így az AND LD eltűnik. A két létrafokból keletkezett utasításlistát egymás után fűzzük, a „CSILLAG” parancsokat egyszerűen töröljük (3. mnemonic kód).

```
LD 8000.00
„CSILLAG”
```

Mnemonic kód 1: első létrafok

```
LD 8000.01
CTUP
8001
#5
AND LD
OUT 8000.02
```

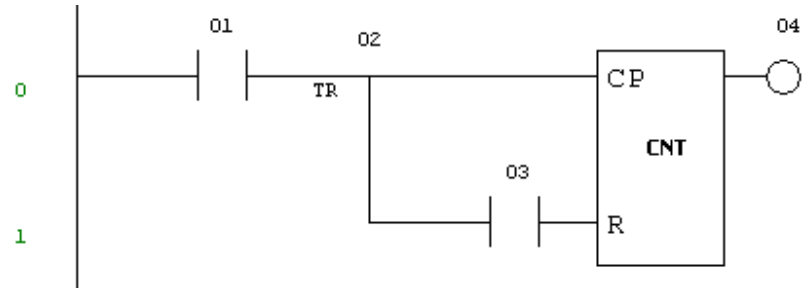
2. Mnemonic kód: második létrafok

```
LD 8000.00
LD 8000.01
CTUP
8001
#5
OUT 8000.02
```

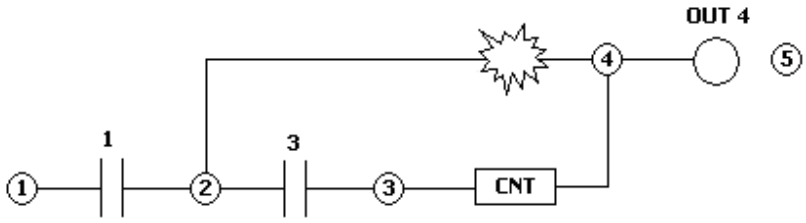
3. mnemonic kód: „CSILLAG”-ok törlése után

A CTUP utasítás a végrehajtás során két adatot vesz ki a veremből (léptetőjel és reset), és egyet tesz bele (az OUT számára).

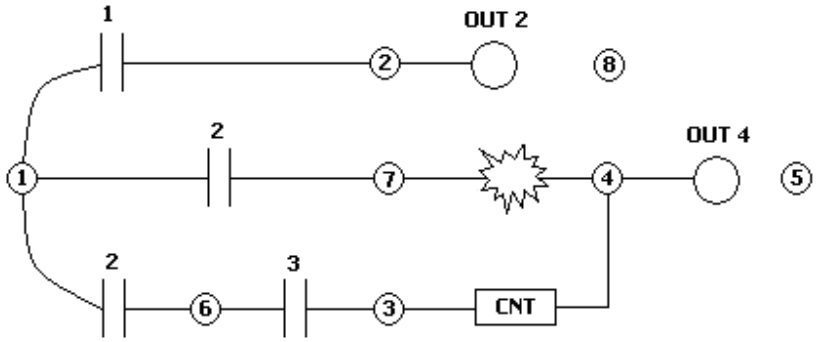
A fordító helyes működéséhez azonban módosítanunk kell a soros összevonás szabályait. A 6. ábrán látható létra feldolgozásának lépéseit a 7-9. ábra mutatja.



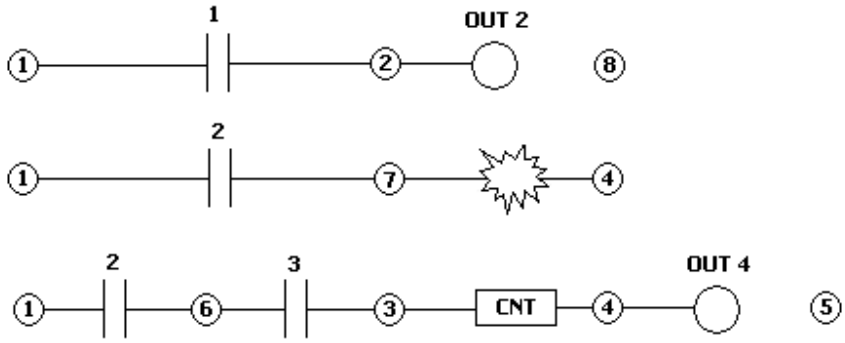
6. ábra. Példa a soros összevonás módosítására



7. ábra. Gráf alak



8. ábra. A TR-ek kibontása után



9. ábra. Másodlagos létrafokok

A harmadik létrafok esetében nem mindegy, hogy a soros összevonásokat milyen sorrendben hajtjuk végre. A 7.és 8. mnenomickód mutatja, hogy mi keletkezi, ha a 6-3 pontok közötti éllel kezdjük az átalakítást. A másik létrafokkal együtt a kész programot mutatja a 9. mnemonic-kód, ami hibás, mivel többet teszünk a verembe, mint amennyit kiveszünk. (Veremtartalom változásának összege nem zérus).

```
LD 2
LD 3
CNT
ANDLD
ANDLD
OUT 4
```

4. mnemonic-kód: AND LD kifésülése előtt

```
LD 2
LD 3
CNT
OUT 4
```

5. mnemonic kód: AND LD kifésülése után

A másik két létrafokkal együtt (a zárójel a veremtartalom változását jelzi):

```
LD 1        (+1)
OUT 2       (-1)
LD 2        (+1)
LD 2        (+1)
LD 3        (+1)
CNT         (-1)
OUT 4       (-1)
```

6. mnemonic kód: Az összes létrafok együtt

A 10-12. mnemonic kód mutatja a helyes megoldást: 1-6 éllel kell kezdenünk. A 1-3. pszeudó kód, és a 10-11. ábra mutatja a helyes algoritmust.

```
LD 2
LD 3
AND LD
CNT
ANDLD
OUT 4
```

7. Mnemonic kód: Eredmény, ha az 1-6 éllel kezdünk

```
LD 2
AND 3
CNT
OUT 4
```

8. Mnemonic kód: az összevonások után

```
LD 1        (+1)
OUT 2       (-1)
LD 2        (+1)
LD 2        (+1)
AND 3       (0)
CNT         (-1)
OUT 4       (-1)
```

9. Mnemonic kód: A többi létrafokkal együtt

```
Amíg van változás
{
        Felesleges „CSILLAG”-ok törlése
        Soros-párhuzamos összevonások
        „CSILLAG” és számláló összevonások
}
```

1. pszeudó kód: Összevonások algoritmusa

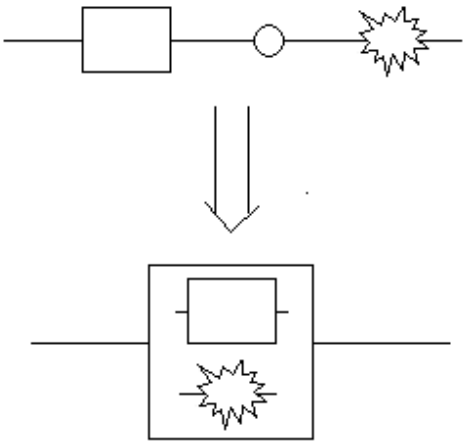
```
Amíg van változás
{
        Minden élre
        {
                Megvizsgáljuk, hogy össze lehet-e vonni másik éllel.

                Ha igen
                        Akkor soros-párhuzamos összevonást végzünk
        }
}
```

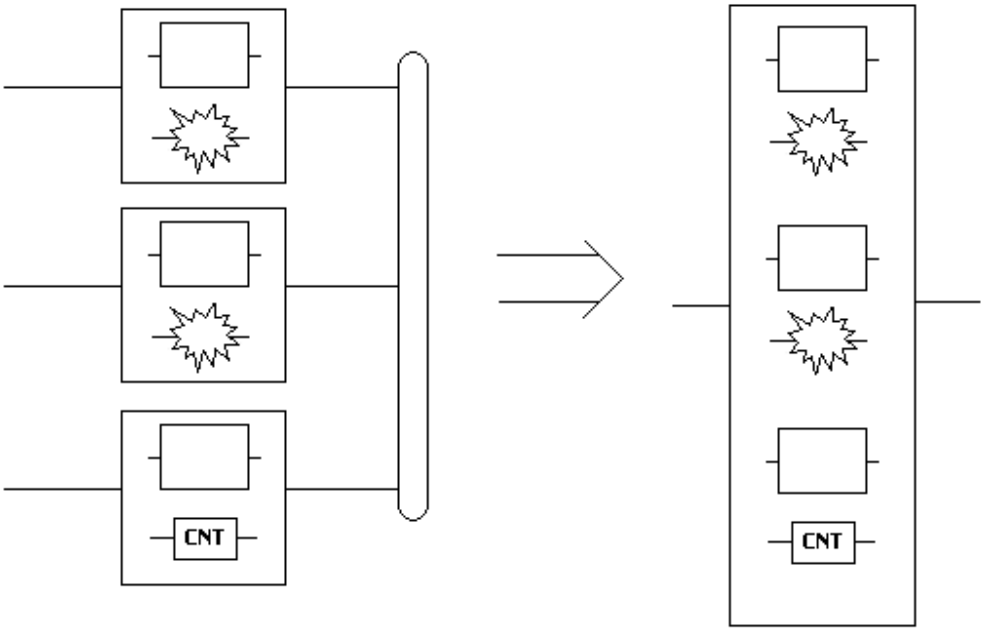
2. pszeudó kód: Soros-párhuzamos összevonás algoritmusa

```
Minden élre
{
        Számláló vagy „CSILLAG” összevonást végzünk
}
```

3. pszeudó kód: „CSILLAG” és számláló összevonások algoritmusa



10. ábra. A csillag összevonás



11. ábra. A számláló összevonás

IPARI ALKALMAZÁSOK

Az új PLC első alkalmazása rendhagyó módon egy kovácsoló gyártósor irányítása volt. A PLC önállóan – NC modul nélkül – üzemel, és egy hidraulikus prést, egy körasztalt, egy indukciós kemencét és önálló robotvezérlőkön keresztül három ipari robot munkáját irányítja. Azóta egy fogazógép, egy dörzshegesztőgép és egy karusszel eszterga felújítását is ezzel a vezérléssel oldottuk meg, és várhatóan 2010 tavaszától már a sériagyártás is megkezdődik.

TOVÁBBI FEJLESZTÉSEK

A szöveges PLC nyelvek vitathatatlan előnye, hogy a PLC programok szerkesztése a jól kidolgozott általános célú szerkesztőkkel megoldható. A szöveges kód – bár nehezebben érthető – de nagyon rugalmas. Ezt a létradiagram esetében felhasználóbarát szerkesztő kidolgozásával ellensúlyozható. A PLC szerkesztőnek együtt kell működnie a villamos tervezőrendszerekkel is. A hatékonyságjavítás fontos területe, az egyre összetettebb PLC utasítások beépítése, mellyel a PLC program tömörebbé, a PLC program megírása lényegesen gyorsabbá tehető.

Bár az első őspéldányok elkészítése óta az NCT létradiagram szerkesztője sokat fejlődött, mégis célszerű a szerkesztőt erőteljesen fejleszteni, mivel minden munkaóra, amit PLC programozók munkáját gyorsító megoldások kidolgozásával töltünk, többszörösen megtérül.

FELHASZNÁLT IRODALOM

[1] IEC 61131-3 Programmable controllers - Part 3: Programming languages International Standard, International Electrotechnical Commission, 2003

[2] John T. Welch: Translating unrestricted relay ladder logic into Boolean form. Computers in Industry, 20 (1992) 45-61

[3] NCT szerszámgép vezérlések PLC programozási leírása
http://www.nct.hu/pdf/NC_Documents/Magyar/Telepites/magplc.pdf

[4] H. A. Barker, J. Song, P. Townsend: A rule based procedure for generating programmable logic controller code from graphical input in the form of ladder diagrams, Eng. Appli. of AI, 1989, Vol. 2, December

[5] R. Wareham, Ladder diagram and sequential function chart languages in programmable controllers, Can. Mach. Metalwork., December 1988, pp. 25-26.

[6] R. Devanathan, Computer aided design of relay ladder diagram from functional specification" Int. Conf. on Industrial Electronics, Control and Instrumentation – IECON, 1990, pp. 527-531.

[7] D. Harel, Statecharts: a visual formalism for complex systems, Sci. Comput. Programming Vol. 8, 1987, pp. 231-274.

[8] M. Courvoisier, R. Valette, J. Bigou and P. Esteban, A programmable controller based on a high level specification tool, IECON 1983, pp. 174-179.

[9] T. Murata, N. Komoda, K. Matsumaoto and K. Haruna, A Petri-net based controller for flexible and maintainable sequence control and its applications in factory automation, IEEE Trans. Ind. Electron, Vol. IE-33, No. 1, February 1986, pp. 1-8.

[10] IEC PAS 62407 Real-time Ethernet control automation technology (ETHERCATTM), International Electrotechnical Commission, 2005