Mach3 Brain Editor

Alapok példákkal

Bevezető

A Mach3 rendelkezik egy kiegészítő funkcióval, amely lehetővé teszi bemenetek és kimenetek PLC szerű kezelését. Ez a Brain Editor, ami a 2007-es kiadása óta az 1-es verziószámot viseli. Ebből sejthető, hogy a fejlesztésével, hibajavításokkal azóta nem foglakoztak.

A használata mégis sok esetben hasznos és esetenként elkerülhetetlen. Az általam ismert hibákat többször is megemlítem és egy összefoglalóba is beteszem.

Nagyon fontos, hogy mielőtt valaki nekilát egy Brain (jelentése: agy, agyvelő) szerkesztésének, legyen legalább alapszintű ismerete a PLC felhasználási területéről és programozása elvéről.

Ennek a tudásnak a megszerzéséhez javaslom áttanulmányozni

Bablena András: PLC programozás kezdőknek I.

című könyvecskéjét. A módszertant jól bemutatja, a legfőbb elvi hibák elkerüléséhez pár szabály ismertetésével nagy segítséget ad.

1.

A **Brain** egy egységét „**Lobe**”-nak nevezte el a fejlesztő. (A **lobe** talán legjobb magyar megfelelője a lebeny.)

Az egy állományban található „lebenyek” alkotják az „agyat”. Mivel nekem az „agy” kicsit sutának tűnik, maradok a Brain névnél.

Tehát egy Brain lebenyekből áll. Egy lebeny kezdődik a baloldalon elhelyezkedő bemenő jelekkel, adatokkal és záródik a jobb oldalon egy kimenő értékkel.

Fontos: egy lebeny több bemenő értékkel kezdődhet, de minden esetben csak egy lezáró kimeneti értéke lesz! (Nem is lehet másként szerkeszteni, így ez nem hordoz hibalehetőséget.)

A lebeny bemenete sokféle lehet.

Kétállapotú (0 - 1; hamis – igaz; false - true) jelek, belső változók, DRO-k értékei, MODBUS regiszterek tartalma. A kétállapotú jelek lehetnek fizikai elnevezett bemenetek és kimenetek, LED-ek, Mach belső állapotjelzői, MODBUS regiszterek bitjei.

Figyelem!

A belső változók (V0-V99) bitjei is megadhatóak bemenetként. De ezek használata a Mach3 teljes összeomlását okozza! Ha a Brain már azelőtt engedélyezve volt és futott, mielőtt egy váltózó bitjére utaló bemenetet bele szerkesztettünk, akkor emiatt a Mach3 már el sem fog indulni!

Megoldás: nevezzük át a Braint, indítsuk a Mach3-at, majd töröljük a hiba okát!

DRO-k, MODBUS regiszterek, belső változók bemenő (analóg) értékeit összehasonlításokban, képletekben használhatjuk fel leggyakrabban.

A lebenyek létrehozásakor figyeljünk, hogy minden bemeneti elem után legyen egy **No Operation** (későbbiekben **NOP**) vagy **Invert** elem! Ezek elhagyása esetén a szerkesztő nem minden esetben fog jelezni, de a használatuk kifizetődő. (Könnyebb szerkeszthetőség.)

Ha a feladat nem egyszerű, érdemes bevezetni jelzőket (külföldiül marker vagy merker). Erre a célra két lehetőség van. A felhasználói LED-ek és a felhasználói DRO-k.

Figyelem!

A Brain-ből használt LED-eknek van egy bosszantó tulajdonságuk. A beírt érték ellentétét tárolják. Ez valószínűleg a program hibája.

A DRO-k nem logikai érték tárolására lettek kitalálva, de használhatjuk arra is egy kis trükkel. A trükk az, hogy egy analóg értékből összehasonlítással logikai értéket állítunk elő.

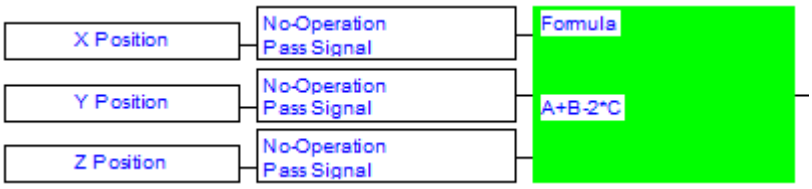
2.

A Brain Editor

Ikonok:

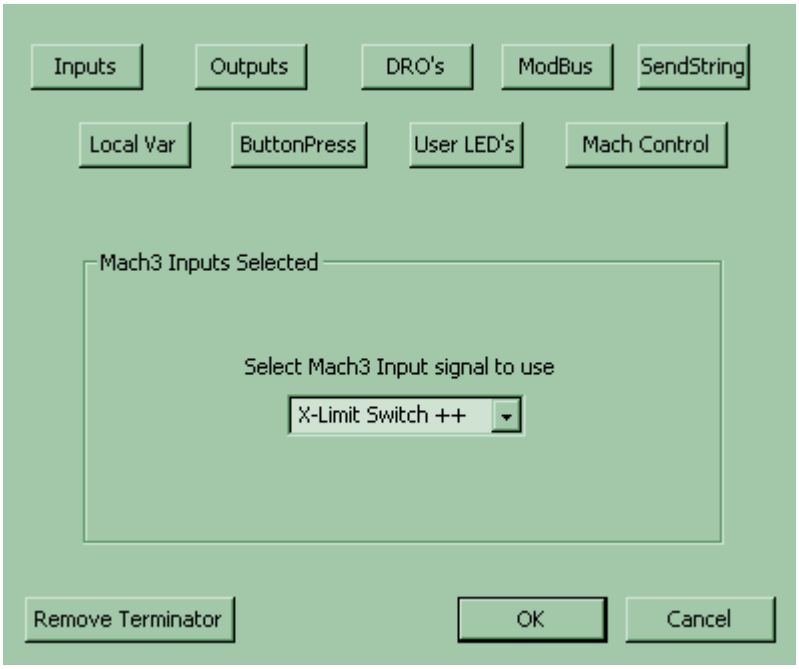
+ : ezzel az ikonnal adhatunk elemeket a lebenyekhez. Ha egyetlen előzőleg létrehozott elem sincs kijelölve (zöld színnel kiemelt téglalap), akkor egy új bemenetet ad a Brain-hez.

Több operandusú elemek is léteznek. Ezek a bemenő elemek kijelölése után adandók az adott lebenyhez. Ilyen elemek például: logikai műveletek (AND, OR, EOR), képlet (Formula), összehasonlítások.



- : ez az ikon a kijelölt elem eltávolítására való. Lezárt lebeny utolsó elemét kijelölve a lezárást is eltávolítja. (A szerkesztő nem minden esetben engedi az elemek eltávolítást, ha sorrendben a törlendő alatt lezárt lebeny van. Sokszor ez bosszantó, mert a lejjebb szereplő lebenyt a hibátlansága ellenére vissza kell törölni, hogy a felette lévő szerkeszthető legyen.)

: zárja a lebenyt. Itt kell megadni, hogy a lebeny eredménye hova kerüljön. A lehetőségek azok, amelyeket a szerkesztő felajánl.

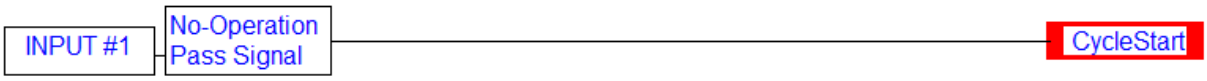


Első gyakorlat: A legegyszerűbb Brain

Készítsünk el az első Brain-ünket, majd teszteljük is! Legyen első feladatunk egy külső jel és a Start gomb egymáshoz rendelése.

A legegyszerűbb Brain csak egy lebenyt tartalmaz. A lebenyben egy bemenő jel van, ami módosítás nélkül megy a lezáró kimentre. Ez a példa pont ilyen.

Az **INPUT#1** után lévő **No Operation** (üres utasítás) szükséges, e nélkül a lebeny nem zárható!



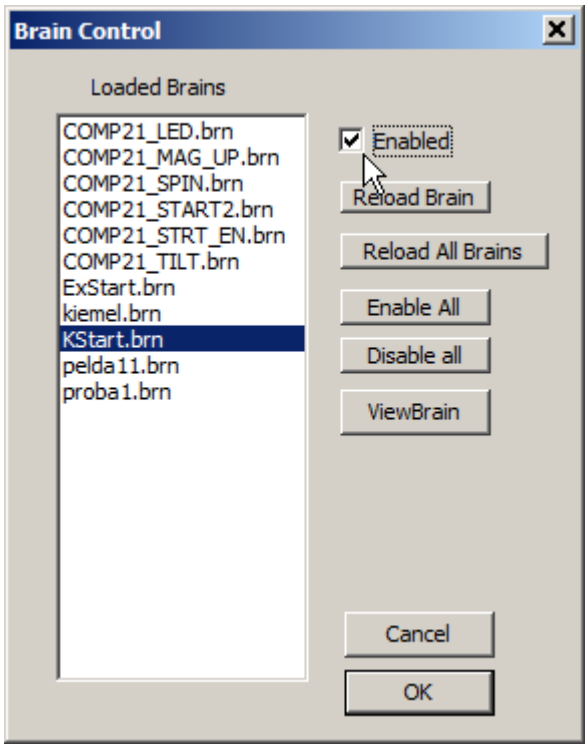
A Mach3 Operator menü Brain Editor… elemével indíthatjuk a szerkesztőt. Nevet nem fontos megadni.

- A + ikonnal (fent részletezve) adjuk hozzá a bemenő jelet, ami az Input-ok lenyíló listájából választott **INPUT#1** legyen!
- Kattintsunk a létrejött téglalap közepére, hogy az zöldre váltson! Ezzel kijelöljük azt az elemet, amelynél a lebenyt folytatni akarjuk.
- A + ikonnal adjunk hozzá egy üres utasítást (No Operation). A most hozzáadott üres utasítást jelöljük ki!
- Zárjuk a lebenyt a fejre állított zöld **T**-re hasonlító jellel. A lezárásnál a ButtonPress (Gomb megnyomása) alatt válasszuk a CycleStart-ot!

Ezzel a szerkesztés kész. Mentsük el a Braint ***KStart*** néven! A szerkesztőt bezárhatjuk.

Ahhoz, hogy egy Brain fusson, engedélyezni kell! Ezt az Operator menü Brain Control …

menüpont alatt tehetjük meg. A Mach3 indítása után létrehozott Brain nem jelenik meg a listában.



Fontos!

Minden esetben, ha Brain-t szerkesztettünk, a teszt előtt kattintsunk a Reload All Brains (Minden Brain újratöltése) gombra!

Automatikusan nem töltődik be az új verzió!

Jelöljük ki a ***KStart.brn***-t és pipáljuk ki az *Enabled*-et! Ezzel engedélyezzük a kijelölt Braint. Ha tiltani akarjuk, kattintással eltávolíthatjuk a pipát.

Ha nem vagyunk ott a CNC gépünknél, ahol fizikailag adottak a be- és kimenetek, működtető elemek, kapcsolók, akkor is tudunk Brain-t fejleszteni és tesztelni. Ugyanis a bemenetek emulálhatóak billentyűzetről.

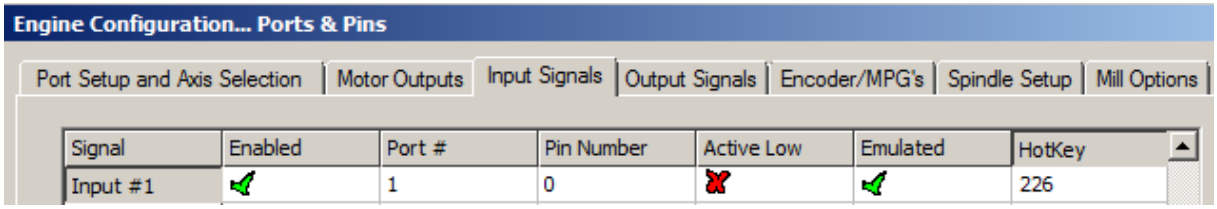
Fontos!

Az emulálás csak akkor működik, ha a Mach3 konfigurációja olyan hardverelemeket tartalmaz, amelyek léteznek! Tehát alapesetben akkor is fel kell telepíteni a LPT port meghajtó programját, ha nem akarunk az adott géppel valódi CNC-t vezérelni! Ha külső mozgásvezérlőnk van, akkor azt kell telepíteni és csatlakoztatni is a teszt idejére! (UC100, UC300 például.)

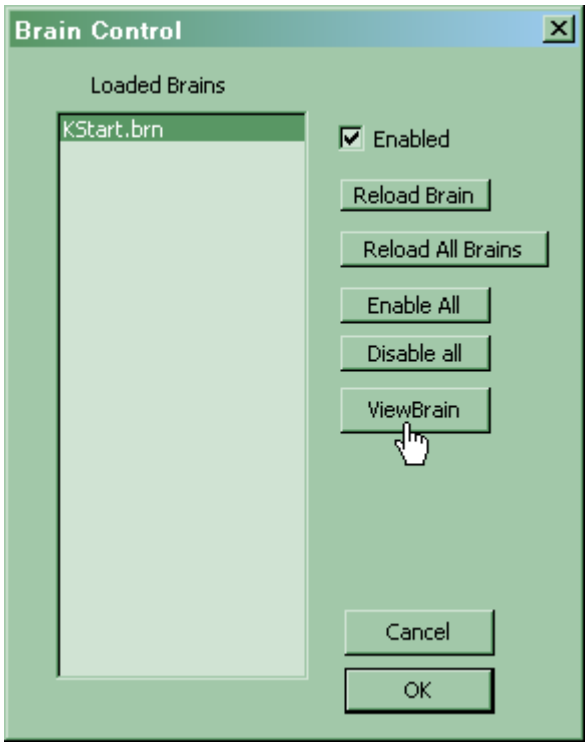
Az emulálás feltétele, hogy az adott bemenet engedélyezzük, adjunk meg egy létező Port számot! A Pin-t 0 (nulla) értékre érdemes beállítani, így egy valódi bemeneti érték nem zavarhat bele a dologba. Meg kell adni még egy billentyűt vagy kombinációt, amivel a jelet aktiváljuk! Itt javasolnám a magyar billentyűzet ékezetes gombjait. A Mach3 fejlesztői sok billentyűzet-kombinációt már előre meghatároztak, de ezekkel nem „számoltak”.

(Csak most, csak egyszer: a Mach3 kezelésének, konfigurálásának ismerete részemről minden „*brénbuherátor”* felé feltételezett. Semmi olyat nem magyarázok, ami ebbe a körbe tartozik.)

Állítsuk be az **INPUT#1**-et emulálásra! A billentyű legyen a kis hosszú **í**!



- A Brain Control ablakában válasszuk ki azt a Brain, amit szeretnénk a tesztelés alatt figyelni! (KStart.brn)
- Kattintsunk a ViewBrain gombra, majd nyomjunk OK-t!



A képernyő felbontásától függően átméretezhetjük a Mach3 képernyőjét, hogy a Brain-t jól láthassuk a teszt alatt.

Ahhoz, hogy az **INPUT#1** jelet a hozzárendelt billentyűvel emulálni tudjuk, a Mach3 ablakának kell az aktívnak lenni!

Ha a hozzárendelt billentyűt megnyomjuk, akkor az **INPUT#1** és a többi elem is zöldre vált. A zöld szín a magas logikai szintet jelöli. (1 vagy igaz (true) értéknek is mondhatjuk.) A kék az alacsony szintet. (0 vagy hamis (false).)

Töltsünk be egy programot most a Mach3-ba és úgy teszteljük, hogy valóban jól működik e a Start gomb megnyomása a Brain segítségével!

<u>Második gyakorlat:</u> Öntartó kör és bontó ág

Gyakori feladat egy öntartó kör létrehozása. A következő példa ezt mutatja be. Itt két új elem jelenik meg: összehasonlítás és a logikai VAGY. Most markernak DRO-t használunk.

A DRO-k nem logikai érték tárolására lettek kitalálva, de használhatjuk arra is egy kis trükkel. A trükk az, hogy egy analóg értékből összehasonlítással állítunk elő logikai értéket.

A felhasználó részére szabadon hagyott DRO-k 1000-2254-ig terjednek.

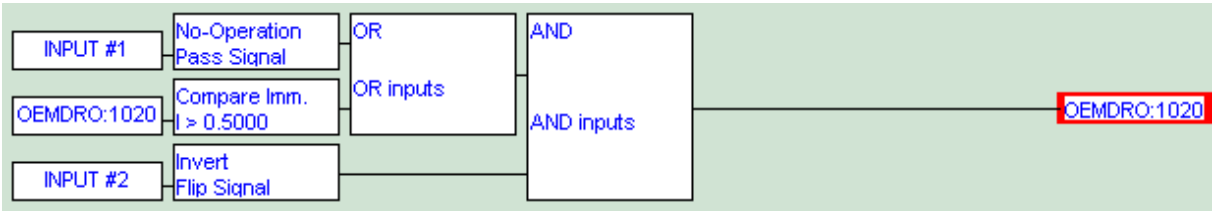
A működését teszteljük a már beállított **INPUT#1** bemenet emulálásával!

Figyelem!

A gyakorlás során csak az éppen tesztelt Brain-t engedélyezzük!

Mint látjuk, ha a lebeny egyszer magas szintű bemenetet kapott, akkor a kimenet már úgy is marad. Nincs kialakítva az úgy nevezett „bontó ág”. Pótoljuk a következő módon!

- Nyissuk meg az előbbi Brain-t!
- Távolítsuk el a lezárást! (Jelöljük ki az OR elemet és kattintsunk a zöld fordított T-re! A felugró ablakban nyomjuk meg a Remove Terminator gombot!)
- A + ikonnal adjunk hozzá a lebenyhez egy új bemenetet. Legyen ez az INPUT#2!
- Invertáljuk az INPUT#2 jelét!
- Jelöljük ki az utolsó elemeket és adjunk hozzájuk egy ÉS kapcsolatot!
- Zárjuk a lebenyt OEMDRO1020 kimenettel!



A teszteléshez az **INPUT#2** bemenet emulálását is állítsuk be! Legyen a hozzárendelt billentyű az é!

Ha mindent jól csináltunk, akkor a **INPUT#2** bontja, az **INPUT#1** zárja a kört.

A markerek céljára a DRO-knál sokkal kézenfekvőbb az egyébként is kétállapotú LED-ek használata. Egy kis nehézséget okoz és nagy odafigyelést kíván azonban a már említett fordított működés.

Figyelem!

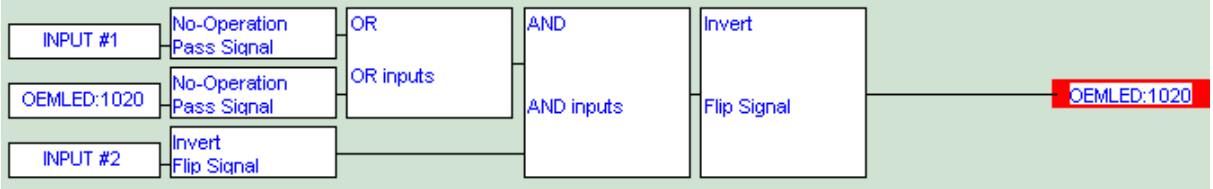
A Brain-ből használt LED-eknek van egy bosszantó tulajdonságuk. A beírt érték ellentétét tárolják. Ez valószínűleg a program hibája.

Ennek figyelembevételével az előbbi megoldást dolgozzuk át LED-re! Először azt kell eldöntenünk, hogy hol fogjuk a jelet invertálni? A LED jelének beállítása előtt, vagy a jel feldolgozása során?

Én az első módszert alkalmazom, így csak egyszer kell megoldani az invertálást. Ha ugyanazt a LED-et többször bemenetként használjuk, akkor minden esetben kellene erre figyelnünk. Másik érv a módszer mellett, hogy a Screenset-en esetleg jelenlévő LED így fog helyesen működni.

A felhasználó részére szabadon hagyott LED-ek 1000-2254-ig terjednek.

Átdolgozva a Brain:



Teszteljük!

Első ránézésre kicsit zavaros. Ugyanaz a jel a bemeneti oldalon a valós szintet, míg a kimeneti oldalon invertáltat mutat. Könnyen bele lehet gabalyodni…

Figyelem!

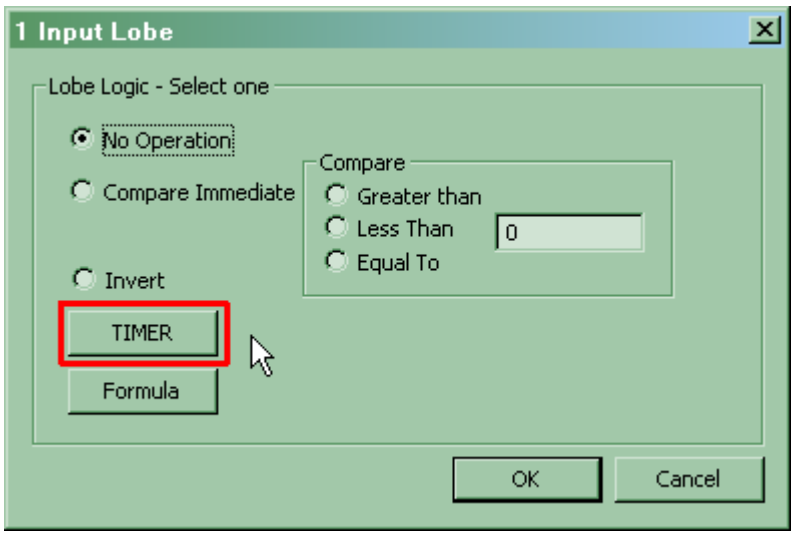
A csak marker célra használt LED vagy DRO nem kell, hogy létezzen a Screenset-ben.

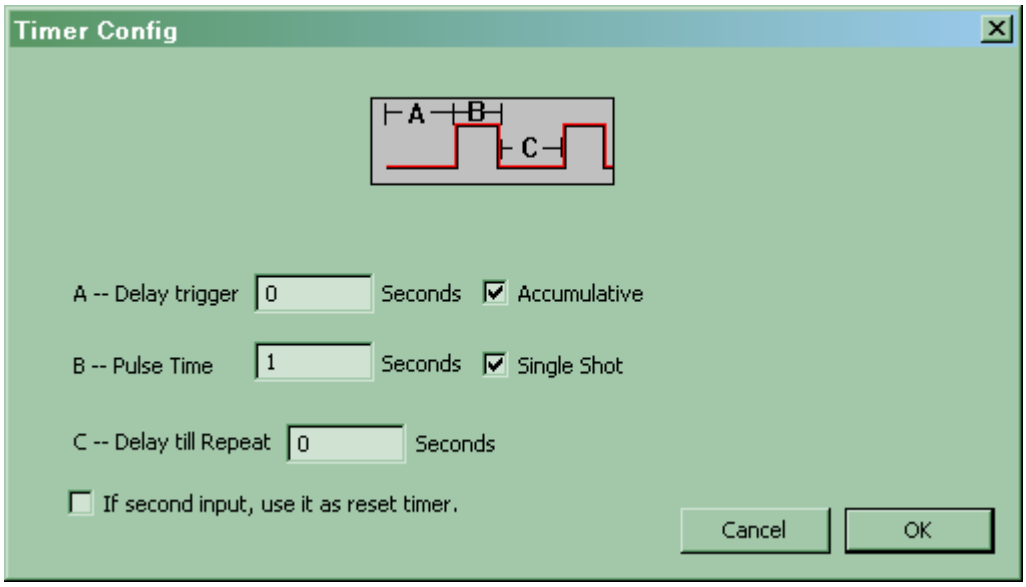
<u>Harmadik gyakorlat:</u> Időzítők

Lehetőségünk van a lebenyekben időzítőket is használni. Ezeket késletetésre, ütemezésre, vagy akár folyamatos négyszögjel kiadására is felhasználhatjuk.

Készítsünk egy Brain-t, amely az **INPUT#1** jel magas szintjére 1 másodpercre magasra állítja az **OUTPUT#1**-et!

A TIMER modult TIMER gomb megnyomásával tudjuk elérni.





A mezők működése, hatása:

A : a vezérlő jel legalább ilyen hosszan kell, hogy stabilan magas értéken legyen, hogy az időzítő elinduljon. Ha az Accumulative ki van pipálva, akkor a Brain a futása ideje alatt „öszegyűjti" a vezérlőjel magas szintjének idejeit. Így csak az első impulzus kiadás előtt vár A ideig, mert a tár nem törlődik automatikusan.

B : Az időzítő ilyen hosszú magas szintű jelet fog kiadni. Ha a Single Shot ki van pipálva, akkor csak egy impulzust ad, egyébként a beállításoknak megfelelő időzítéssel ismétli, amíg a vezérlőjel magas.

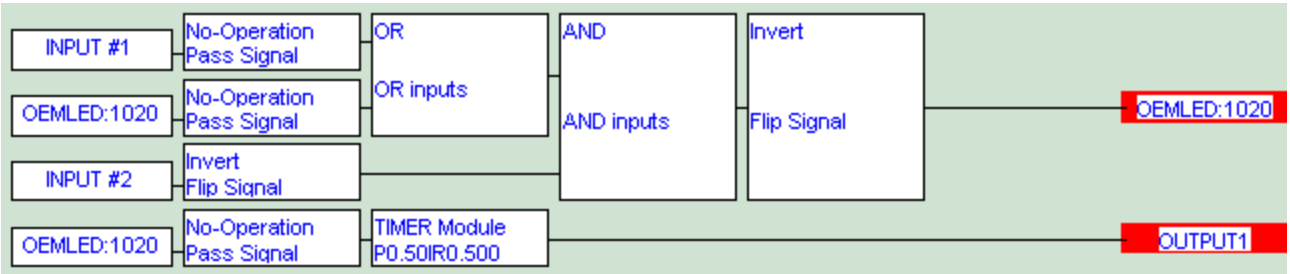
C : az ismétlések közötti idő. (Csak Single Shot kikapcsolása esetén értelmezett.)

Ha az időzítőnknek két logikai bemenete van, akkor az „alsó" **Reset**-ként működik. Megszakítja a pulzus kiadását és törli a késleltetés időzítőjét is.

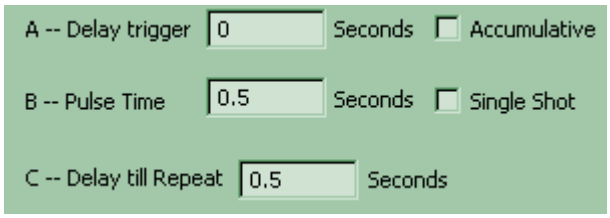
Kísérletezzünk különböző idők megadásával és különböző üzemmódokkal!

Végül készítsünk egy Braint, amely az **IINPUT#1** jelre elindít egy Timer-t, ami 1 másodperces ciklusidejű szimmetrikus négyszögjelet ad, amíg azt az **INPUT#2** meg nem szakítja!

Ez lesz az első olyan Brain a gyakorlatok során, amely több lebenyt tartalmaz. Elsőként készítünk egy öntartó kört bontó ággal. Ez beállít egy markert, amelyet a Timer vezérlőjeleként használunk fel.



Timer mező értékek:



Negyedik gyakorlat: Program Stop

A Mach3 programstop megoldása nem sikerült jóra.

A FeedHold nem állítja meg azonnal a futást. Bizonyos előre feldolgozott lépésszám után áll csak meg a léptetés.

A Stop gomb hatására azonnal abbamarad ugyan a Step jelek kiadása, de ez kiszámíthatatlan pozícióvesztést okozhat. Léptetőmotoros rendszernél szinte minden esetben, szervónál kisebb sebességnél, jó beállítások esetén nem biztos.

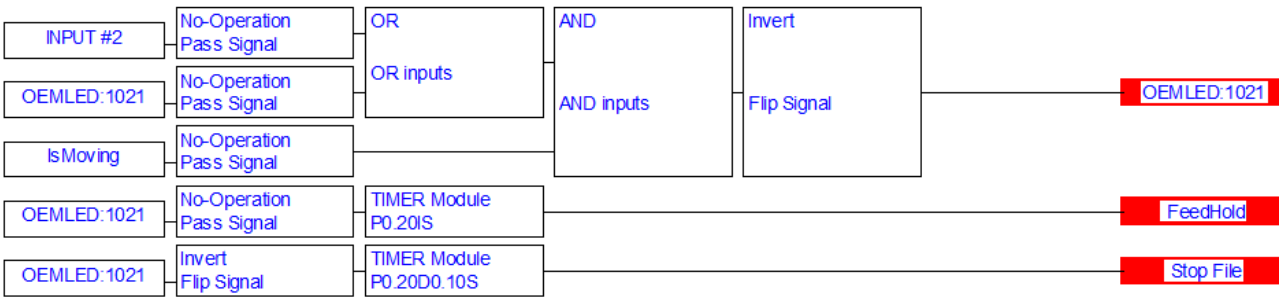
Az alap Screenset-en szerepel mindkét gomb (FeedHold és Stop). Ez komolyabb vezérlők esetén ismeretlen megoldás. Ha külső gombokkal akarjuk megvalósítani a főbb funkciók kezelését, akkor teljesen fölösleges a Start gomb mellé két Stop funkciót kezelő gombot tenni. Ha feltételezzük, hogy a Stop funkciót nem azonnali (vész) leállításra fogjuk használni, akkor készíthetünk egy Brain-t, amely úgy fogja kezelni a külső Stop gombot, hogy az ne okozzon pozícióvesztést.

A módszer: A külső Stop hatására a Brain megnyomja a FeedHold gombot, majd vár, amíg a mozgások leállnak. Ezt követően kis késleltetéssel megnyomja a Stop gombot. Így megvalósul a valódi STOP, de nincs pozícióvesztés. Vészleállásra pedig ott a Vészstop gomb. (Illik ott lennie.)

Ilyen elven megvalósított Brain működik egy ipari koordináta-fúrógépen.

Teszteljük egy hosszú futásidejű programmal, hogy a többszöri Start-Stop is kipróbálható legyen!)

KStop.brn



INPUT#2 - külső Stop gomb jele

OEMLED1021 – Stop kérés markere

IsMoving – Mach3 belső állapotjelző. Magas, ha valamelyik tengely mozog.

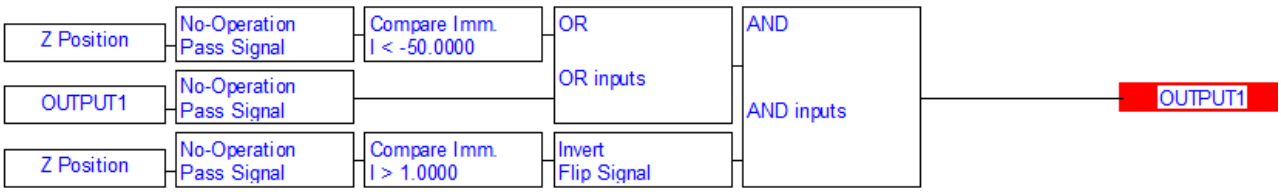
Működés:

- Első lebeny: ha külső STOP aktív **és** a szánok mozognak, akkor bekapcsolja a markert. (OEMLED1021). Az öntartás a marker és a Stop jel **logikai vagy** kapcsolatával van megoldva. A marker a bekapcsolását követőn mindaddig aktív marad, amíg a bontó ág jele magas. A bontó ágat a szánok mozgását jelző **IsMoving** adja.
- Második lebeny: a marker aktívvá válásakor azonnal megnyomódik a FeedHold gomb. A megnyomás 0,2 másodpercig tart.
- Harmadik lebeny: a marker törlése után 0,1 másodperccel a Stop gomb megnyomódik, szintén 0,2 másodpercre. A megnyomás pillanatában már a szánok állnak, pozícióvesztés emiatt nem lesz.

A tesztek szerin időzítések nélkül is működik a Brain. Komoly gyakorlati tapasztalatok híján azonban biztonságosabbnak gondolom az időzítős verziót.

Ötödik gyakorlat:

A következő Brain ***Antal Gábor*** problémafelvetésére készült, ami a következő: Eszterga Z koordináta értékéhez kötötten kell egy kimenetet működtetni (munkahenger vagy behúzó mágnes). Ez menetvágásnál a szerszám fogásból történő kiemelését szolgálja. (Nem volt vezérelt X tengely a gépen kialakítva.)



Teszteléséhez elegendő a Z tengely mozgatása.

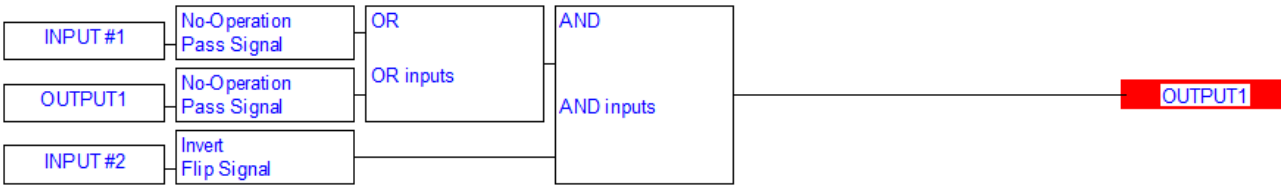
Működés:

- A mozgásokat a Brain alaphelyzetének beállításához a bontó ág összehasonlításában szereplő Z értéknél pozitívabbról kell indítani! (Itt Z+1-nél nagyobbról.)
- Ha a Z érték eléri vagy kisebb lesz, mint -50, akkor bekapcsolja az **OUTPUT#1** kimenetet, ami egyben az öntartást is megoldja.
- Amíg a Z értéke nem lesz nagyobb, mint a bontó ág összehasonlító értéke (+1), addig az **OUTPUT#1** bekapcsolt marad. Tehát ahhoz, hogy a ciklus újrainduljon, a Z mozgást úgy kell tervezni, hogy minden ciklus végén meghaladja a +1-et!

Bár a Mach3-at ritkán használjuk tartály feltöltésre, de ha belegondolunk, a fenti Brain két szintjelző használatával tökéletesen alkalmas lenne rá. Egyszerűsödne, mivel összehasonlításra nem lenne szükség, csak két bemenő jelre.

INPUT#1 – alsó szintjelző

INPUT#2 – felső szintjelző



Hatodik gyakorlat: Analóg adatok felhasználása.

Analóg bemeneti adatokkal végezhető műveletek:

- összehasonlítás (Compare)
- képletben számítások (Formula)
- átvezetés analóg kapcsolón (Analogue Switch)

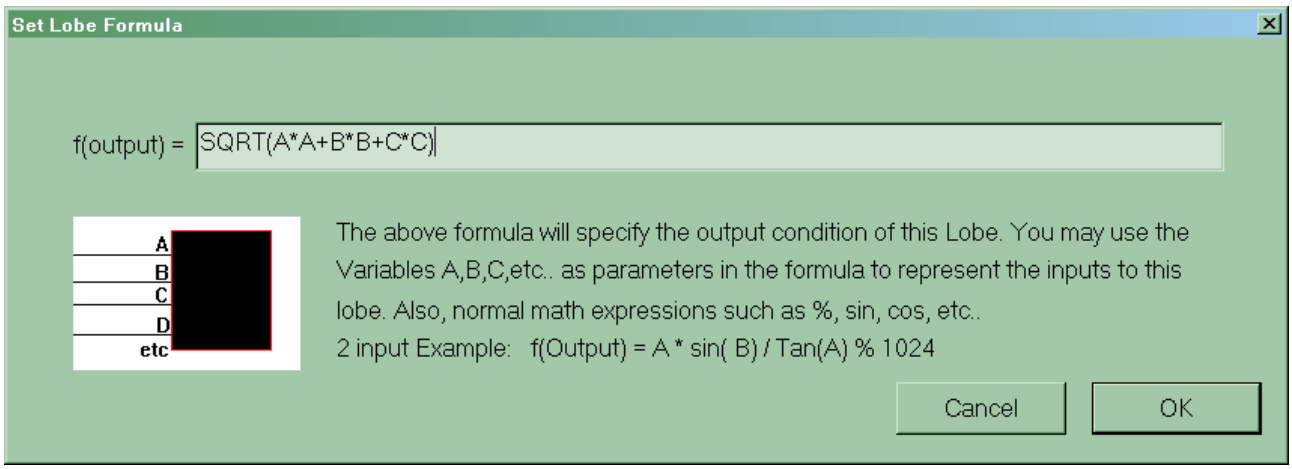
Összehasonlítás:

Egy analóg bemenet és egy állandó, vagy két analóg bemenet összehasonlítása. (Kisebb, nagyobb, egyenlő.) Az eredménye logikai érték, (1) igen vagy (0) nem.

Példa a fentebb említett Antal Gábor féle probléma megoldása.

Képlet:

A következő Brain kiszámolja és egy felhasználói DRO-ba írja a szerszám programozott pontja és a munkadarab nullpontja közötti távolságot. (Gyakorlati haszna nincs.)



A képletben használható matematikai funkciók, függvények nincsenek publikálva.

Ami általam ismert:

SQRT(x) – négyzetnyök-vonás (square root)

SIN(x) – szinusz : szöget radiánban kell megadni!

COS(x) – koszinusz : szöget radiánban kell megadni!

TAN(x) – tangens : szöget radiánban kell megadni!

ARCSIN(x) – arkusz szinusz: eredmény radiánban!

ARCCOS(x) – arkusz koszinusz: eredmény radiánban!

ARCTAN(x) – arkusz tangens: eredmény radiánban!

^ - hatványozás: 3^3=27

SIGN(x) – előjel függvény

LOG(x) – 10-es alapú logaritmus

EXP(x) – e hatványozása (LN ellentéte, e^x)

LN(x) – természetes (e) alapú logaritmus; e~=2,71828182845904523536028747135 27

ABS(x) – abszolút érték függvény

INT(x) – egész érték függvény (levágja a tizedes jegyeket)

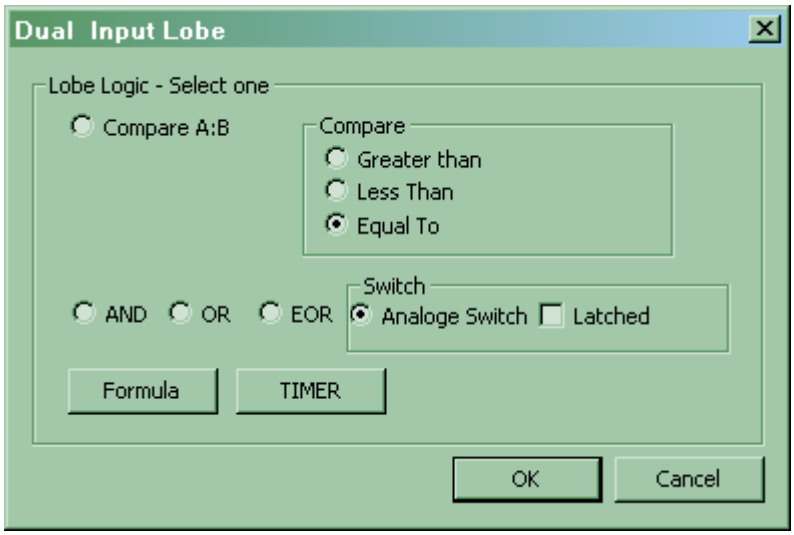
A matematika alapműveletek természetesen alkalmazhatóak. A feldolgozás sorrendje a megszokott.

Minden itt felsorolt függvény és funkció megtalálható és tesztelhető a ***funkciok.brn***-ben.

Ha valaki ismer ezen kívül még mást is, szívesen veszem, ha tájékoztat róla.

Analóg kapcsoló:

Egy analóg bemenettel rendelkező (A, tehát felső jel) és egy logikai értékkel (B) vezérelt elem. Ha a logikai érték magas (1), akkor átereszti az analóg értéket, ha alacsony (0), akkor nem. (Nulla analóg értéket enged tovább.)

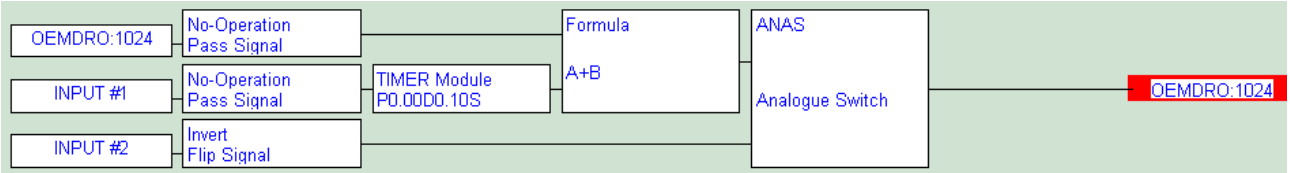


Az alábbi példa egy gyakori, a PLC-k többségében szereplő elem megvalósítását mutatja. Ez egy számláló, ami nem szerepel a Brain eszközei között.

INPUT#1 – a számláló léptető jele

INPUT#2 – a számláló Reset (törlés) jele

OEMDRO1024 – a számláló



A léptető jelre kis késleltetéssel indul egy nagyon rövid időzítésű impulzus, ami a számláló aktuális értékével együtt egy képletbe (Formula) jut, ahol a két érték összeadása történik. A logikai érték itt számként van feldolgozva (0 vagy 1). (A kis késleltetés egy fizikai érintkező pergésmentesítését szolgálja.) A képlet számítási eredménye egy analóg kapcsolón keresztül íródik a számlálóba. (A számlálót első használat előtt alaphelyzetbe kell állítani!)

A képlet kis módosításával a lefelé számlálást is meg tudjuk oldani. Próbáljuk ki!

A Mach a 3.43.0 verziójától rendelkezik egy új funkcióval, amely **..\Brains\AutoLoad** könyvtárban lévő összes Brain-t betölti és engedélyezi minden indításkor.

Végszó:

Ez a kis leírás a példáival és a közreadott ismeretekkel nem a teljességre törekvés jegyében született. Az elkészítése közben is több új dolgot tanultam és van még mit. ☺

Remélem, néhány olvasónak ösztönzést ad ez a kis szösszenet, hogy a misztikusnak gondolt korlátokat átlépve a gyakorlatban is kipróbálja és használja a Mach3 eme modulját. ☺

2014. április 1.

Béni

Két rövid bemutató a Brain szerkesztő használatáról:

http://youtu.be/832D4JNuK9Y/

http://youtu.be/5f6jlnjyRoY/

Hibák, figyelmeztetések:

Fontos: egy lebeny több bemenő értékkel kezdődhet, de minden esetben csak egy lezáró kimeneti értéke lesz! (Nem is lehet másként szerkeszteni, így ez nem hordoz hibalehetőséget.)

A belső változók (V0-V99) bitjei is megadhatóak bemenetként. De ezek használata a Mach3 teljes összeomlását okozza! Ha a Brain már előbb engedélyezve volt és futott, mielőtt egy váltózó bitjére utaló bemenetet beleszerkesztettünk, akkor emiatt a Mach3 már el sem fog indulni!

Megoldás: nevezzük át a Braint, indítsuk a Mach3-at, majd töröljük a hiba okát!

A Brain-ből használt LED-eknek van egy bosszantó tulajdonságuk. A beírt érték ellentétét tárolják. Ez valószínűleg a program hibája.

Minden esetben, ha Brain-t szerkesztettünk, a teszt előtt kattintsunk a Reload All Brains *(Minden Brain újratöltése) gombra!*

Automatikusan nem töltődik be az új verzió!

Az Inputok billentyűzetről emulálása csak akkor működik, ha a Mach3 konfigurációja olyan hardverelemeket tartalmaz, amelyek léteznek! Tehát alapesetben akkor is fel kell telepíteni a LPT port meghajtó szoftverét, ha nem akarunk az adott géppel valódi CNC-t vezérelni! Ha külső mozgásvezérlőnk van, akkor azt kell telepíteni és csatlakoztatni is a teszt idejére! (UC100, UC300 például.)

A gyakorlás során csak azt a Brain-t engedélyezzük, amelyiket éppen tesztelünk!

A csak marker célra használt LED vagy DRO nem kell, hogy létezzen a Screenset-ben.

A Mach3 indulásakor a DRO-k, LED-ek értéke bizonytalan. A megbízható működés megköveteli ezek alaphelyzetbe állítását. (Természetesen csak az általunk létrehozott és használt felhasználói elemekről van most szó.)

Egy Brain és egy makró között a kapcsolatot ezekkel az elemekkel oldjuk meg. Ilyenkor nem elég csak arra figyelnünk, hogy a működő Brain-ekben egy adott kimenet csak egyszer szerepeljen. Itt már a megvalósítandó feladat tervezésekor el kell döntenünk, hogy az adott kimenetet mi fogja beállítani. (Brain vagy makró.)

*Az **„egy kimenet csak egy helyen szerepeljen”** szabályt a makrókra is ki kell terjeszteni!*

Nagyon fontos!

Egy Brain nem tartalmazhat akármennyi elemet!

Nincs dokumentálva, hogy mennyi bemeneti elem esetén fog még biztosan működni. Van ahol a tapasztalatok alapján 50-et írnak, van ahol 40-et. Utóbbit javaslom tartani! Ha a feladataink megvalósításához ez nem elegendő, akkor azokat osszuk el több Brain-be. Jó módszer, ha ezt egy adott funkció vagy funkciócsoport szerint tesszük.

Egyidejűleg több Brain lehet aktív.

Ajánlott irodalom:

- ***Bablena András: PLC programozás kezdőknek I.***
- ***Mach3 Macro Programmers Reference Manual*** *(mellékelve)*

Hivatkozások:

UC100, UC300: Polgárdi Balázs által Mach3 alá fejlesztett USB-s mozgásvezérlő.

- - UC100: A Mach3-as program a CNC gépeket a printer porton keresztül képes vezérelni. A printer porton történő pontos időzített jelek előállítása az operációs rendszer részéről igen nagy feladat, és vannak korlátai is (nem ajánlott futtatni más programot a Mach3-on kívül, mert azok befolyásolhatják a kiküldött jelek “simaságát”). Az UC100-as ezt a problémát hidalja át. Csak csatlakoztatni kell a számítógép USB portjába és CNC motormeghajtó áramkörére szabványos printercsatlakozón keresztül. A Mach3 az UC100-as kimeneteit (12db) és bemeneteit (5db) ugyan úgy kezeli, mintha printer portot használnánk. A kimenő jeleket nagy pontossággal hardveresen állítja elő, és valamivel több, mint 1sec. memóriával puffereli. A hardveres kimenőjelek előállítása az operációs rendszer válláról sok feladatot levesz, így alkalmazhat kisebb processzorteljesítményű számítógépet is.
- UC300: Az UC100 nagyobb testvéreinek első tagja az UC300-5LPT modul. A modul úgy lett kialakítva, hogy 5db LPT portnyi ki és bemenetek vannak IDC-26 csatlakozókra kivezetve, melyek lábkiosztása kompatibilis a LPT porttal. Az 5LPT portból 2 normál és 3 bővített bemenetű. Összesen 36db 5V-os digitális kimenet és 49db 5V-os digitális bemenet található rajta, melyek szabadon konfigurálhatóak. A sok digitális ki és bemenet mellett 2 analóg kimenet és 2 analóg bemenetet is tartalmaz. Az analóg kimenetek közvetlen 0-10V-os jelet szolgáltatnak, így egyszerű illeszteni frekvenciaváltóhoz.

http://www.polgardidesign.hu/