F ü g g e l é k

R o b o t

A r o b o t i k a h á r o m a l a p t ö r v é n y e :

(I s a a c A s i m o v : É n é s a r o b o t . 2 0 5 8)

1. *A robotnak nem szabad kárt okoznia emberi lényben vagy tétlenül tűrnie, hogy emberi lény bármilyen kárt szenvedjen.*
2. *A robot engedelmeskedni tartozik az emberi lények utasításainak, kivéve, ha ezek az utasítások az első törvény előírásaiba ütköznének.*
3. *A robot tartozik saját védelméről gondoskodni, amennyiben ez nem ütközik az első és a második törvény előírásaiba.*

I . B e v e z e t ő a r o b o t i k á b a

1 . A r o b o t o k f e l é p í t é s e é s a l k a l m a z á s a

1.1 Történeti háttér

Politika I. című művében Arisztotelész már megfogalmazta koncepcióját olyan eszközökről, amelyek képesek automatikusan, a saját erejükből tevékenykedni. Az ókor első működő automatáját valószínűleg az ókori Alexandriában építette Archimédész kortársa, Kteszibiosz, az időszámításunk előtti III. században, igaz, puszta szórakoztatás céljából. Az i.sz. I. században a szintén alexandriai Héró ötletes szerkezeteiről részletes és pontos leírások maradtak fenn. Többek között árusító automatákat, valamint egy automata színházat tervezett és épített meg - ez utóbbi, a *Dionűszosz apoteózisa,* egy önjáró miniatűr templom tetején játszódott.

Alexandria arab bevétele (641) után is tovább élt az automaták építésének hagyománya, bár főleg vallásos célokat használt. A középkor automata-tervezői érdeklődésüket és valószínűleg technikai ismereteiket is az arab kultúrkörből merítették. Az előrelépést főleg az acélrugó feltalálása volt a XV. században, amely az automaták működtetésére nagy sikerrel volt alkalmazható az ókorban használt kősúlyok és a (hevítéssel előállított) nagynyomású levegő helyett.

Ezeket vallásos tárgyú jelenetekhez, vagy a nézőközönség megbabonázásának céljaira használták fel. Ilyen automatáknak (és a későbbi bonyolult automata planetáriumoknak) nagy szerepük volt abban, hogy a mechanisztikus világkép oly nagy teret nyert a XVII. századra (Descartes). A XVIII. században az egyre bonyolultabb és tökéletesebb automaták konstruktőrei között **Jacques Vaucanson**t és **Pierre Jacquet-Droz**t érdemes megemlíteni: ez utóbbi nevéhez fűződik a legfontosabb életfunkciókat (beleértve az anyagcserét!) imitálni képes mechanikus kacsa (1738) és a ma is működőképes három remekmű: az író, a rajzoló és az orgonista (1775).

Csak a XX. század második felében válnak az automaták és robotok *kulturális* objektumokból egyre inkább *technológiai* objektumokká. 1928-ban a londoni gépipari kiállítást egy "mesterséges ember" nyitja meg rövid beszéddel és udvarias gesztusokkal.

Az első mai szemmel is iparinak nevezhető robot az amerikai Unimation cég "PUMÁ"-ja volt a 60-as évek közepén. 1987-ben Nyugatnémet országban több, mint száz cég 250 különböző robotot kínált. A nagyvilágban 1990.-ben már kb. 200 000 robot működött.

*Robotiká*ról , mint a robotok kifejlesztésének és felhasználásának tudományáról a 70-es évek végétől lehet beszélni, korábban ez a terület a matematikához (kibernetika), a számítástudományhoz (programozás) és az automatizálás technológiájához (szabályozástechnológia) volt kapcsolható. Jelenleg a robotika önálló - bár erősen interdiszciplináris jellegű tudományterület.

A robotok az irodalomban is kivívták népszerűségüket: elég, ha Offenbach "Hoffmann meséi" *Olympi*ájára, H.G. Wells *Világok háborújá*ra, vagy Čapek *R..U.R* (Rossum univerzális robotjai) c. regényére gondolunk, amely utóbbiból maga a *robot* szó is származik (*robota* = nehéz munka, robot).A *Csillagok háborúja* c. filmben szereplő R2-D2 és C3-PO a fantasztikus regények gólemjeinek, androidjainak, kiborgjainak legújabb változatai.

1.2 Ipari robotok

Mobil robotok egyelőre nem játszanak alapvető szerepet átfogó ipari alkalmazások terén. Jelenleg meglepő tulajdonságokkal rendelkező, leginkább játékra, házibuli-meglepetésként alkalmazható laboratóriumi modellek vannak - a japánok ebben úttörők - (fonó, kétlábon járó, orgonán játszó robotok, stb).

A mai **ipari robot** (IR) helybe rögzített konstrukció. "Az ipari robotok egyetemesen felhasználható, többtengelyű, mozgó automaták, melyeknek mozgása a mozgásfázisok sorozata és útvonalai /szögei tekintetében szabadon, azaz mechanikus beavatkozás nélkül programozhatók, szükség esetén szenzorral szabályozhatók. Felszerelhetők fogókkal, szerszámokkal és egyéb eszközökkel és képesek manipulációs és termelő feladatok elvégzésére." Sokféle, robothoz hasonló eszközt *nem* tekintünk robotnak: például egyes **adagoló eszközöket**, (pl. hengerművek), melyek funkcióját csak a *hardware* változtatásával állíthatjuk át, **manipulátorokat,** (pl. radioaktív izotópok), távirányítható **telefaktorokat** (holdkomp), **CNC** (**C**omputed **N**umerical **C**ontrol), **DNC** (**D**irect **N**umerical **C**ontrol) elven működő eszközöket.

Az ipari robotok két fő csoportba sorolhatók: **"standing"** (alaphoz rögzített) és **"portal"** (három további független tengely körül forgatható keretbe, "portál"-ba szerelt, ezáltal lényegesen megnövelt munkaterületű) robotok.

1.3. Alapvető fogalmak és definíciók

1.3.1 Robotfizika

Kinematika: helyzetek, irányok, sebességek és gyorsulások vizsgálata. A pillanatnyi kinematikai paraméterek pontos és hatékony beállítása a robotika egyik legfontosabb feladata.
Statika: emelők, hajlító és forgatónyomatékok, szilárdság, torziós jelenségek. Dinamika: Erők (Coriolis-, súrlódási, stb.) és hatásaik (tehetetlenség, csavarodás, rezgések).
Szenzortechnológia: Erők és nyomatékok érzékelése (kontakt szenzorok); vizuális, mágneses, kapacitív, induktív és lézerszenzorok (nem-kontakt szenzorok). Kísérleti stádiumában gyors fejlődés tapasztalható; ipari alkalmazásai egyelőre csak a lézeres és vizuális szenzoroknak vannak. A modern szenzorok feladata a vezérlő számítógép "modell-világának" folyamatos adatellátása, ezáltal a robot és környezete közötti kapcsolat lehetővé tétele. (Ld. még "Model World".)

1.3.2 Mechanika

A klasszikus ipari robot **teste** a környezethez rögzített tengely (test-tengely) körül szabadon elforgatható - általában 360^{o}-nál kisebb szöggel. A robottesthez a **felsőkar** a **vállízület**en, az **alsókar** a felsőkarhoz a **könyökízület**en, a **kéz** az alsókarhoz a **csuklóízület**en keresztül kapcsolódik. A kézhez különböző **effektorok** (fogó, hegesztő- vagy szórópisztoly, stb.) rögzíthető. A robottestet, a felső- és alsókart külön motorok mozgatják; a kéz, ezáltal az effektor térbeli **orientációját** pedig 3 motorral lehet beállítani. A robotnak tehát legalább 6 tengelye (ízülete), s mindegyik tengelyhez külön motorja van. A konstrukcióra jellemző határokon belül mindegyik tengely a többihez **függetlenül** mozgatható. Bár a tengelyek általában forgatást tesznek lehetővé, egyes esetekben beszélhetünk **transzlációs** tengelyekről is (Scara-elven működő **hajlókarú robotok**, mozgatható robotkeretek esetén).

A kéz és ezáltal az effektor helyzetét az ún.**TCP**-vel (tool center point) jellemezzük; ez a pont az effektor típusától függ: hegesztőpisztoly esetében a TCP az elektróda-csúcsok közötti felezőpont, egy fúró esetében a fúró hegye, stb. A TCP-t a kéz síkjának középpontjához viszonyítjuk.

A **munkaterület**, **WS** (working space) azon pontok összessége a térben, amelyet a TCP elérhet. A WS-t a konstrukció geometriája határozza meg, határoló felületei általában síkok, henger-, kúp- és gömbfelületek.
Ha $P \ni$WS a TCP számára elérhető pont, akkor annak ismételt megközelítése esetén a TCP a P körüli *variációs gömb* belsejébe esik. Az **ismétlési pontosság** a legnagyobb variációs gömb sugara, ha P bejárja WS-t. Az ismétlési pontosság a robot alkatrészeinek precizitásától és merevségétől függ.
Ha $Q \ni$WS tetszőlegesen kijelölt pont (függetlenül attól, hogy a TCP elérheti), akkor az ismételt megközelítések esetén a TCP és Q között adódó legnagyobb eltérést **pozícionálási pontosság**nak nevezzük. A pozícionálási pontosság egyrészt az ismétlési pontosságtól, másrészt a pozícióméró rendszer felbontásától függ.

1.4. A robotok felhasználása

Az USA-ban először 1965-ben, Európában a hetvenes évek elején jelentek meg az ipari robotok, elsősorban az autóiparban. Elterjedésükre jellemző példa, hogy míg 1980-ban egy BMW-14 előállítására 192, 1985-ben már csak 89 órára volt szükség.

80-as évek végén Németországban körülbelül 14.000 ipari robot működött szerelési (ponthegesztés, vonalhegesztés, összeszerelés, zománcozás, polírozás, stb.) és termelési (szerszámgépgyártás, hengerelés, kovácsolás, fröccsöntés, stb.) folyamatokban, viszonylag kis százalékban (2,8 %) kutatási és oktatási célokra is.

Terjedésük elsősorban azokban az országokban gyors, ahol a bérek magasak (Németország, Svédország). Itt több-száz ezer munkahely szűnt-szűnik meg a robotok alkalmazása miatt.

A belátható jövőben elsősorban

- a mezőgazdaságban (birkanyírás, fejés),
- a mély tengerekben (pl. a transzatlanti kábel karbantartása, elsüllyedt kincsek keresése, stb.),
- az űrkutatásban (űrállomások összeszerelése),
-a gépipari tervezésben,
- a hadiiparban (automata tankok és fegyverrendszerek, majd intelligens harci robotok) is várható az elterjedésük.

A távolabbi jövőben a determinisztikus alapon működő robotokat felválthatják a feladat-orientált, szenzorvezérlésű, automatikus adaptálódásra képes robotok.

- A különböző **CAP** (**C**omputer **A**ided **P**lanning), **CAD** (**C**omputer **A**ided Design), **CAM** (**C**omputer **A**ided **M**anufacturing) és **CIM** (**C**omputer **I**ntegrated **M**anufacturing) folyamatok teljesen integrálhatóak lehetnének segítségükkel.

-A szenzorok tökéletesítése által az emberi érzékelés utánzása, sőt, némely területen meghaladása is elképzelhető.

-Integrált szakértői rendszerek kifejlesztése (a robotok független, racionális és célszerű döntések meghozatalára képesek).

A további fejlődés már a fantasztikus regényekben megjósolt lehetőségek megvalósítása lenne: intelligens, független mozgásra és cselekvésre képes robotrendszerek, emberhez hasonló (antropomorf) androidok.

2. Vezérlési módok

A legalapvetőbb feladatok egyike a TCP mozgatása egy adott indulási helyről az adott végpozícióba. Különbséget teszünk egyszerű **mozgásvezérlés** és **összetett szabályozófunkciók** között. Az első vezérlési mód mindazon funkciókat foglalja magában, amikor a pozíció-mérőrendszer adatait visszacsatolva

szabályozzuk a motorokat. Az összetett szabályozófunkciók magasabbrendűek az előzőnél és általában indítják azokat. Mindezeket egy példával világítjuk meg:

A mozgás szabályozását a ROB3 belsejében elhelyezett elektronika végzi. Ez az elektronikus szabályozó rendszer a parancsokat ASCII szekvenciák (American Standard Code for Information Interchange) formájában kapja a vezérlő számítógéptől az RS232 *interface*-en keresztül, és elvégzi a kiindulási beállítást. Ezzel egyidőben folyamatosan összehasonlítja a pillanatnyi pozíciót a beállított pozícióval **minden egyes** motornál. Ha ez a két változó megegyezik, a mozgás leáll. Zavarok esetén rezgések keletkezhetnek (*hardware* hiba). Minden magasabb szabályozó funkció a ROB3 vezérlő számítógépében helyezkedik el.

Az ipari robotok motorjának **vezérlő egysége** a következő részekből áll:

- egy rendkívül precíz szögmérő-kódolóval megvalósított pozíció-mérőrendszer. Ez lehet, pl. egy üvegkorong, amelynek peremén elhelyezett rovátkák (> 2000) forgás közben impulzust keltenek egy foto-elektromos kapun. Így a forgássebességtől függően maximum 800kHz-es helyzetjelző jelek keletkeznek a vezérlő egység számára. Az érvényes beállított helyzettel (tehát nem a véghelyzettel) való összehasonlítás egy ún. **követő hibajelet** generál. A követő hibajelből (a **drift kompenzáció** hatásának figyelembe vételével) képződik a sebesség-beállító jel a motorvezérlés részére. Ez utóbbi biztosítja, hogy a TCP ne végezzen a beállított helyzet körül rezgéseket. Ez az elem azért szükséges, mert az ipari robotok motorjai, egyenáramú motorok, vagy újabban háromfázisú motorok. A probléma nem létezik léptető motorok esetében (itt egyéb nehézségek vannak).

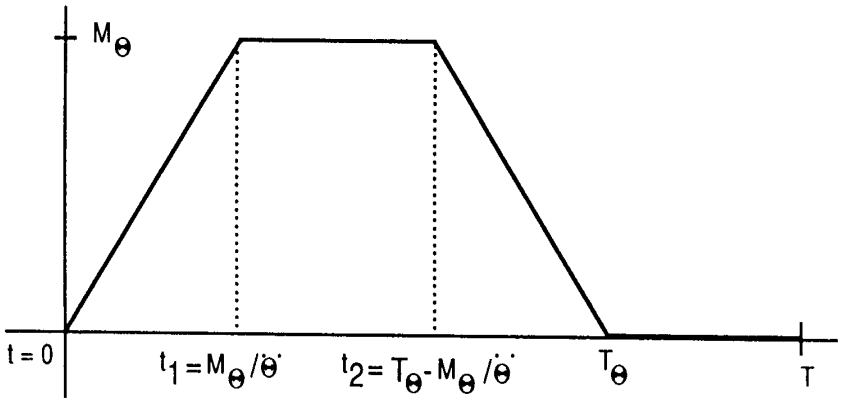
A mozgásvezérlésnek általában két alapvető módját különböztetjük meg: a **PTP-vezérlés**t (**P**oint-to-point) és az **útvonalvezérlés**t Az előbbi esetén csak a mozgás kiinduló- és végpontja alapvető; a **TCP** által bejárt útvonal másodrendű. Az utóbbi esetén viszont elsőrendű, hogy a TCP, amennyire csak lehet, az előre meghatározott útvonalon mozogjon.

Fontos tudnunk, hogy a következő két vezérlési mód leírása csak kivonatos, de érdeklődők a laborban angol nyelven megtalálhatják a részletesebb irodalmat is.

2.1 PTP vezérlés

A PTP vezérlés esetén a mozgás kiinduló- (1) ill. véghelyzetét (2) a robottengelyek körüli forgásszögek értékeinek megadásával jellemezzük. A pillanatnyi helyzetben minden egyes tengelyre nézve kiszámoljuk a megfelelő szögkülönbségeket. Tiszta PTP vezérlés esetén az egyes motorok egymástól függetlenül, a lehető leggyorsabban végrehajtják a hozzájuk tartozó tengely körüli forgatást. Minthogy a szögkülönbségek a mozgás folyamán változnak, a mozgást kiváltó motorok különböző időpontokban működnek. Ennek következtében a mozgás szaggatottá válhat.

Nézzük meg a C.1.-es ábrát:

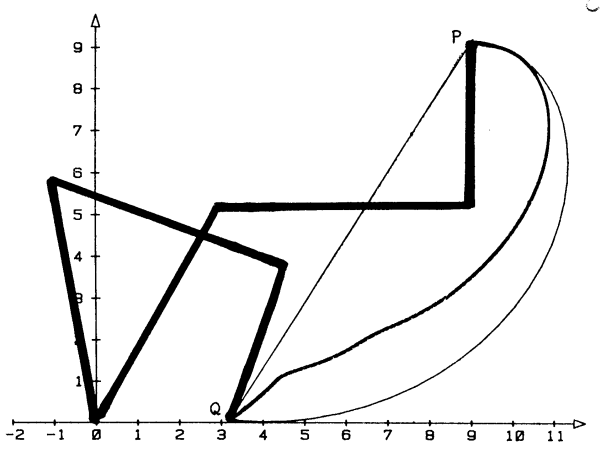


C.1. ábra.

Ha a mozgás részleteit vizsgáljuk, azt látjuk, hogy a forgatás állandó Θ szöggyorsulással indul, ezután a max. szögsebességet elérve állandó szögsebességgel halad egy ideig, majd állandó szöggyorsulással fékez. A teljes ΔΘ szögelforduláshoz szükséges T_θ időt az elemi kinematikából ismert módon, integrálással számíthatjuk ki (M_θ a szögsebesség):

$$T_\Theta = \Delta\Theta / M_\Theta + M_\Theta / \ddot{\Theta} T_{\dot{e}}$$

A teljes PTP elmozdulás ideje az egyes tengelyek körüli elfordulási idők közül a legnagyobb. Az eredmény azért fontos számunkra, mert főleg nehéz terhek mozgatása során a PTP mozgás szaggatottsága a robot ízületeinek és áttételeinek gyors kopásához vezethet. Ipari robotok esetében ezt célszerű elkerülni, az ún. **szinkron PTP-vezérlés**re való áttéréssel. A mi esetünkben ennek a neve **tengely-interpoláció**. A mozgatásban résztvevő összes tengely azonos időpontban kezdi és végzi mozgását Ennek megvalósításához a gyorsabb mozgásra képes tengelyeknek a leglassabbhoz, kell igazodniuk. Az egyszerűség kedvéért a gyorsulásokat nem változtatjuk, csak a maximális szögsebességek csökkennek le a megfelelő értékre.
Az C.2. ábrán láthatjuk a tiszta és szinkron PTP útvonalak közötti különbséget: a felső *P* pont, függőlegesen felfelé állított fogóval, a kiinduló helyzet. A *Q* végpontba való mozgás a felsőkar - alsókar síkban jön létre az A1 test-tengely részvétele nélkül.



C.2.ábra.

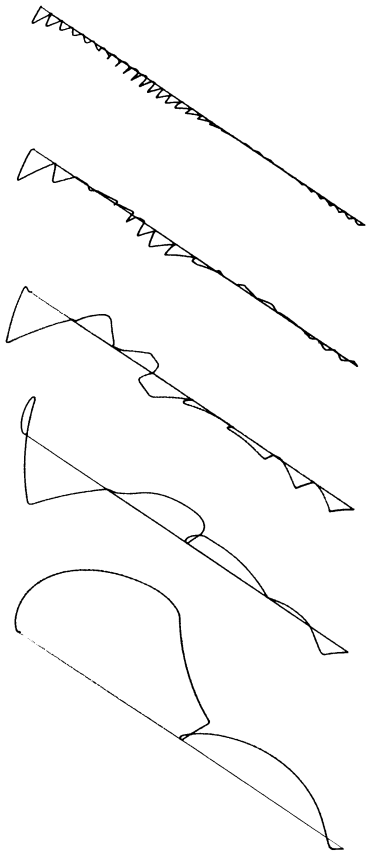
Látható, hogy a szinkron PTP- mozgás egyenletes ívű, bár lényegesen eltér a *PQ* egyenes szakasztól. Ilyen eltérések váratlan mellékhatásokkal járhatnak: pl. akadályba ütközés, ami nem fordulhat elő tiszta PTP mozgás esetén.

Hosszú útvonalak esetén mind a tiszta, mind a szinkron PTP-mozgatásnál célszerű a kezdő- és a végpont közé **köztes pontokat** beiktatni. Ezt az eljárást **interpolációnak** nevezzük (lásd az említett angol nyelvű irodalmat).

A szinkron PTP-vezérlés a ROB3 esetén nem lehetséges; itt az lenne szükséges, hogy a gyorsulások minden egyes tengely esetén külön szabályozhatók legyenek, ami megdrágítaná a modellt.

Lehetséges azonban - köztes pontok beiktatásával - a ROB3 működtetése interpolációs üzemmódban. Ha a tervezett útvonalat a lehető legtöbb köztes pont beiktatásával, kis szakaszokból álló tiszta PTP-mozgások sorozatával közelítjük, **sok-pontos vezérlés**ről (multi-point control) beszélünk. Valójában ez nem jelenti a vezérlésnek egy új típusát. Lineáris és cirkuláris interpolációt az angol nyelvű irodalomban találhatjuk meg.

Az C. 3.ábrán tiszta PTP útvonalak sorozatát láthatjuk, melyek alulról fölfelé növekvő finomsággal (1,3,7,15 és 31 köztes pont beiktatásával) közelítenek egy egyenes szakaszt.



C.3.ábra.

2.2 Útvonalvezérlés

Sokszor szükség van egy szerszám, vagy akár egy munkadarab a lehető legnagyobb pontossággal - egy adott térbeli görbe mentén - előre megadott **útvonal-sebesség**gel való mozgatására.(Ilyen feladat lehet a vonalhegesztés, ragasztás, egy fúró adott helyre való merőleges beállítása, stb.). Ezekben az esetekben nem fogadhatók el a PTP-mozgatásra jellemző rángások és éles irányváltoztatások. Ekkor célszerű az útvonalvezérlés vagy CP - vezérlés (Continiuous Path control) alkalmazása. A PTP-mozgatással ellentétben a CP-mozgatás alkalmával az útvonal pillanatnyi kinematikai jellemzői - helyzet, vonalmenti sebesség és gyorsulás - határozzák meg minden egyes motor működését. A funkcionális függés transzformációs egyenletek útján adható meg.

Ebben az üzemmódban a vezérlés kemény követelményeket támaszt a mozgató- és vezérlőegységekkel szemben. Különösen nagy feladat hárul a vezérlő processzorra, hogy az a számítási műveleteket elfogadható időn belül végezze el.

A ROB3 nem képes ebben az üzemmódban működni. Ehelyett, a meghatározott útvonalon való tartást a lehető legtöbb köztes pont beiktatásával, sok-pontos vezérlés útján, a ROB2R pozícióméró rendszere maximális felbontásának kihasználásával végezzük.

3. Programozás

Programozás alatt a továbbiakban nem az egyszerű mozgás-vezérlés gépi rutinjait, hanem az egyes modulok, eljárások és függvények alkalmazását fogjuk érteni.

3.1. Program-eljárások

Elektromechanikus programozás:

Az előre huzalozott programot csak *hardware úton* lehet változtatni. Egyszerű feladatokra használható robotok, úgymint adagolóeszközök és hengerművek esetén használatos; ez mára már jelentőségét vesztette.

Modern ipari robotok esetében jóval inkább a **tanulás előjátszás alapján** módszer kerül előtérbe. Ennek több, a felhasználói területtől függő változata létezik.

Mozgás programozása visszajátszással (Play-back):

A "tanító" végigjátssza a TCP mozgatását, amelyet egy adott időfelbontásban rögzítünk. A program futása során a mozgás minden fázisa PTP-vezérlés útján ismétlődik. A play-back módszer még mindig használatos, pl. polírozó-robotok programozásához.

Master-slave programozás:

A programozó a robotot egy áttétel segítségével (pl. egy kis modell-robot megfelelően arányos vezérlése útján) mozgatja. A "master" minden tevékenységét (nemcsak a mozdulatokat!) tárolja, és azonos időfelbontásban, azonnal le is játssza a "slave" számára.

Mivel ez a módszer viszonylag bonyolult, elsősorban ott használják, ahol az előző, betanító módszer nem lehetséges (pl. atomerőművek aktív zónája). Abban az esetben, ha a tárolási folyamat kimarad, **teleoperáció**ról, vagy **telemanipuláció**ról beszélünk.

Betanító (Teach-in) programozás:

Ezt a módszert jelenleg is használják majdnem minden ipari robotnál. A programozó a robotot egy mozgatón keresztül vezérli - vagy a programozási helyek durva beállításával vagy a tanítás-vezérlő programozó eszköz (teach control device) manuális használatával. Evvel a nagyobb zsebszámológéphez hasonlatos eszközzel a robot, biztonsági okokból, csak erősen csökkentett sebességgel működtethető. Nem csak a helyzetek, de az interpolációk, gyorsulások, sebességek, késleltetési idők és szenzor-utasítások is a programba kerülnek.

Megjegyzendő, hogy a modern rendszerekben általában csak a program-keret tanítása történik evvel a módszerrel. A mozgás részletei **on-line** módon kerülnek a programba.
A ROB3 TCP mozgásának paramétereit teach-in és **off-line** módon is programozhatjuk.

Off-line programozás:

A programozás szöveges formában írt program segítségével történik. Az on-line programozással ellentétben, amely rendkívül mozdulat-orientált, az off-line programozás inkább **feladat-orientált**. Ahelyett, hogy a robot minden mozdulatát külön beprogramoznánk (mint pl. a teach-in módszer esetén), a feladat absztrakt megfogalmazására helyezzük a hangsúlyt. A programozás hatékonyságát egyes rendszerek felhasználóbarát szolgáltatásokkal segítik (pl. az Apple **Mackintosh** a grafikus tervezési elemekkel). A módszernek, sok előnye mellett (a programozás már a tervezési fázisban lehetséges, stb.), alapvető hibája többek között, hogy a programozás hibái sokszor csak az első próbafutás alatt derülnek ki. A hibakeresés és -javítás csak ismételt futások során végezhető, ami növeli a robotok improduktív idejét.

A másik hátrány a programozás absztraktságának következménye, nevezetesen az, hogy a robot egyes funkciókat ellátó részeit és azok kapcsolatát ideálisnak tételezik fel, így nem tud előre számolni a gyakorlatban előforduló pontatlanságokkal és egyéb zavaró körülmények (hőmérséklet, páratartalom, rezgések, stb.) hatásával.

Azt mondhatjuk, hogy a **tisztán** off-line módszer nem jó megoldás. A fenti hibák viszont elkerülhetők, ha a próbafutás előtt az új programot először a terminálon szimuláljuk. Ehhez természetesen szükség van egy teljes **geometriai** és **technológiai adatbank**ra, amely egyfelől az összes alkalmazni kívánt robot alkatrészeinek és perifériáinak (forgó asztalok, futószalagok, stb.) geometriai paramétereit, másfelől a munkafolyamatban szereplő egyéb adatokat (motorok és szenzorok karakterisztikái, teher-válaszfüggvények, szerszámok és munkadarabok fizikai jellemzői) tartalmazza. A szimuláció során a kimenet **real-time** módon 3-dimenziós grafika formájában képernyőre kerül, ahol diagnosztizálhatók és kijavíthatók a hibák - szinkronizációs problémák, ütközések, túlterhelés, stb.

A precíz TCP pozícionálási hibáit a **szenzortechnológia** alkalmazásával próbálják kiküszöbölni. Hagyományosan **tapintó** és **vizuális** elven működő szenzorokat használnak a robotok **adaptív vezérlése** céljából. Itt röviden olyan, **nem-determinisztikus** módszereket mutatunk be, ahol a TCP által bejárni tervezett útvonal a programozás alatt még nem minden részletében ismert. A program futása alatt a szenzorok tudósítják a központi műveleti egységet a robot belső "világát" érintő külső eseményekről, amelynek következtében a robot "modell-világában" megfelelő belső események keletkeznek. Ezáltal a szenzor által küldött információkra a robot adekvát módon képes reagálni. Ezen az elven működik például a **varrat-követő szenzor (seam trace sensor)**. A hegesztendő felületen a kiinduló hegesztési pontot teach-in módszerrel adjuk meg a robotnak. A varrat útvonala a munkadarabtól függ, a szenzor a hegesztés alatt folytonosan követi a hegesztő elektróda útját. A munkafolyamat egyes beprogramozandó lépéseinek száma és bonyolultsága lényegesen csökkenthető. Természetesen ennek a módszernek megvannak a korlátjai (alak-felismerési problémák, **reach-in-the-box** feladat). Ha egy dobozban, amelybe további feldolgozásra váró munkadarabok esnek, elég nagy a "rendetlenség", és a kiszemelt munkadarab kellően bonyolult alakú, akkor annak azonosítása, helyzetének meghatározása és a manipulátorral való megközelítése túlságosan is komplexnek bizonyulhat a robot alakfelismerő programja számára.

3.2 Programozási nyelvek

Elvben minden univerzális programnyelv alkalmas robotok programozására. A feladat speciális természetéből következően mégis célszerű volt olyan programnyelvek kifejlesztése, amelyeken a tipikus feladatelemek, mint koordináta-transzformációk, effektor- és szenzorvezérlés könnyen programozhatók. Ilyenek például az **SRL** (Structured Robot Language), a **PASRO** (PAScal for RObots), **ROBEX** (ROBot EXapt), **ARLA** (ASEA Robot Language), **BAPS** (Bosch Advanced Programming System), **SRCL** (Siemens Robot Control Language), **ROLF** (Robot Language Formula), **ROBOTstar**, **AL** (Assembly Language), **AML** (A Manufacturing Language), **HELP**, **VAL** (Variable Assembly Language), **SIGLA** (SIGma LAnguage).

3.3 Beállítás

3.3.1 Mechanikus beállítás

A TEST.RLN és a JUST.RLN programok futása után a felső- és alsókar közelítőleg vízszintes állásba került. A fogó felülete az alaplapra merőleges és szögben felfelé mutat. A programból kilépve kikapcsoljuk a tápfeszültséget. Kézzel beállítjuk az Y tengelyt a felsőkar-alsókar síkjával párhuzamosan

Vigyázzunk, hogy az egész robottestet forgassuk, és ne változzon a karok helyzete a testhez képest. Jegyezzük fel a fogó alsó sarkainak az alaplaphoz képesti helyzetét, és az alaplapot addig állítsuk a rögzítő csavarok állítása segítségével, amíg a két távolság meg nem egyezik. A test (Z) tengelyének az alaplapra vett vetületének egybe kell esnie az X tengellyel.

Forgassuk a testet, amíg az X tengely párhuzamos lesz az alsókar-felsőkar síkkal. Mérjük le újra a fogó és az alaplap távolságát. Állítsunk ismét a csavarokon, amíg a távolság meg nem egyezik az előbb beállított értékkel. Ezáltal a robottest merőlegesen áll az alaplaphoz képest.

Kapcsoljuk be ismét a tápfeszültséget és indítsuk el a GRIPPER.RLN programot. A robot a FIX 9 helyzetet veszi fel: a kar majdnem vízszintesen áll, és a fogó zárt helyzetben van. Ha a fogó nem teljesen zárt, le kell szerelni a felső kart, és a fogó kábelét a feszítő rugóhoz rögzítő karmokat laposfogóval addig hajlítsuk, amíg a fogó teljesen zárt helyzetbe kerül (kissé feszített rugó mellett).

3.7.2 Elektromos beállítás

Tegyük fel a mágneses központozó tűket a csukló- és a könyökízület meghosszabbításaként, majd állítsuk hosszukat úgy, hogy a hegyük az alaplapra merőleges és az alsókar-felsőkar síkjával párhuzamos síkba kerüljön. Ekkor válasszuk az **Adjustment**-et a menüben és végezzük el a beállítást az ott megadottak szerint.

A beállítás adatait tároljuk:

FIX 0: $\alpha_1 = 0^\circ$, $\alpha_2 = 0^\circ$, $\alpha_3 = 0^\circ$, $\alpha_4 = 0^\circ$, $\alpha_5 = 0^\circ$, a fogó helyzete 240.

Ez a pozíció lesz a további motortesztek után a programindítás kiindulási helyzete.

K í s é r l e t e z é s é s d e m o n s t r á c i ó a R O B 3 - a l

1. I n t e r p o l á c i ó s k í s é r l e t e k

1.1 Lineáris interpoláció

Anyagszükséglet: építőkocka, ping-pong labda és sín.

Feladat: A ping-pong labdát vezessük a sín vájatában az E pontig zárt fogóállásban. Aztán a fogó felemelésével engedjük a labdát visszagurulni a kiinduló helyzetbe. A feladat megismétlése céljából várakozzunk, amíg a labda megáll.

Megoldás: A zárt fogót emeljük kellő magasságba a kocka fölé, hogy a fogó hegyének Z koordinátája nagyobb legyen a ping-pong labda legnagyobb Z magasságánál. A fogó távolsága a labdától legyen 3-4 mm. Ebből a kiinduló helyzetből végezzünk lineáris interpolációt abba a véghelyzetbe, amelyben a fogó hegye kb. 5-6 mm-re van a sín fölött és az E ponttól kb. a labda sugarának megfelelő távolságra. A koordinátarendszer irányítása a kezdő és végállásban ugyanaz.

A program a következőképpen nézhet ki: (ld., pl. EXP4.RLM)

```
PTP   X 287.3    Y 3.0     Z -206.1   -39.3°    89.9°  -0.6°
PTP   X 288.0    Y 3.0     Z -243.2   -38.8°    90.2°  -0.6°
LIN   X 111.0    Y 225.1   Z -243.5   24.8°     90.1°  -63.7°
PTP   X 111.6    Y 226.1   Z -203.0   24.8°     89.8°  -63.7°
WAIT      Z 30
RPT
```

Az első PTP-utasítás végrehajtása után a fogó legyen csukva. Ha nem lenne, zárjuk be az utolsó paraméter (robot koordinátákban) 255-re való átállításával.

A LIN-mozgás folyamata alatt az aktuális koordinátarendszer irányítása az x tengelyben folyamatosan változik, olyan formán, hogy a fogó síkja állandóan a sínre merőlegesen áll. Az y tengelyben a fogó az alaplaphoz képest állandóan merőleges helyzetben marad. A visszagurulás ideje természetesen a sín meredekségétől függ, ezért a fenti programban a várakozási időt annak megfelelően állítsuk be.

1.2. Cirkuláris interpoláció

Anyagszükséglet: írótoll-tartó, írótoll, rajzpapír, rögzítő mágnesek.

Feladat: az ábra alapján rajzoljunk a robottal körívet.

Megoldás: A FIX 0 helyről indulunk, a fogóba beillesztjük az eszköztartót, a fogó pofáit széthúzva. Az írótollat egy gumigyűrűvel rögzítjük a tartóhoz. Azután helyezzük a fogót nagyjából függőlegesen a kiindulási pont (S) fölé, hogy a zárókupak alja kb. 5 mm magasan legyen. A fogót eresszük rá az alaplapra az S pontban, ameddig a zárókupak engedi. Amikor a zárókupak hozzáért az alaplaphoz, mozgassuk a fogót még lejjebb, egészen, amíg az írótoll a gumigyűrűt 4-5 mm-re megfeszíti. A finombeállítást legjobban Descartes-koordinátákban végezhetjük el, így van egy kis tartalékunk, hogy írás közben a Z irányú egyenetlenségek kiegyenlítődnek.

Most elvégezzük a cirkuláris interpolációt. Az ív már beprogramozott S kiindulási pontja után a Z közbenső pontot, majd az E végpontot programozzuk be. Ügyeljünk arra, hogy a fogó az alaplaphoz képest mindig függőlegesen (y tengely) legyen, és a fogó síkja nagyjából az ív érintő síkjában maradjon.

Példaprogram (a diszken EXP5.RLN):

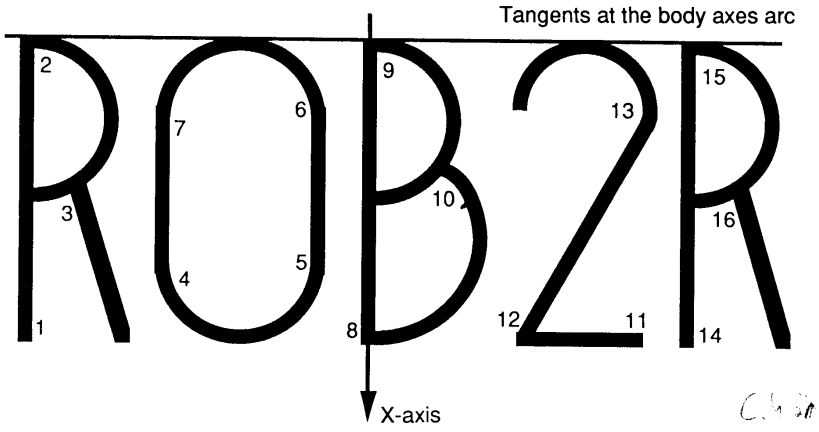
```
FIX 0
PTP   X 157.4    Y 127.1    Z -253.2    -13.4° 90.2°    -38,9°
PTP   X 157.7    Y 127.4    Z -257.6    -13.4° 90.0°    -38.9°
PTP   X 169.8    Y 225.4    Z -259.0    -59.7° 90.8°    -53.1°
CIR   X 77.8     Y 281.0    Z -259.3    -90.0° 90.2°    -74.9°
PTP   X 81.9     Y 302.6    Z -234.1    -90.0° 86.6°    -74.9°
WAIT D 1
RPT
```

A WAIT-utasítás az ESC megnyomása után az ívet még egyszer meghúzza. Ha az ESC után END következik, ez a program megszakítását eredményezi. Ebben az esetben a robot a FIX 0 pozícióban marad. A programot diszken való tárolás után először írótoll nélkül próbáljuk ki. A CIR-utasítás végrehajtása közben többször is kaphatunk *position is not in working space.* Ennek oka az lehet, hogy túl nagy ívet húzunk, és a 3-as tengely a megengedett szögtartomány határára kerül. Ilyen esetekben a hibaüzenetre az ENTER gombbal válaszoljunk, egészen addig, amíg a hibaüzenetek meg nem szűnnek. Ezután először a CIR és az azt megelőző PTP-utasítást töröljük (Delete), majd írjuk újra kisebb X és Y értékekkel.
A hibák kiküszöbölése után a programot a főmenüből RUN paranccsal indítsuk.

2. Írás

Anyagszükséglet: írótoll-tartó, írótoll, rajzpapír, rögzítő mágnesek.

Feladat: ROB2R írja le a saját nevét az ábrán látható módon és méretben.



C.4. ábra.

Megoldás: Az írás egyenes szakaszok és körívek rajzolásából áll. A 2.2-es kísérlethez hasonlóan a FIX 0 pozícióból indulunk beillesztett írószerkezettel, de feltett zárókupakkal.

Az ábrán a számok a LIN vagy CIR szakaszok elejére mutatnak. A vég- ill. középső pontokat programozzuk. Hogyha valamely LIN vagy CIR szakasz végpontja nem esik egybe a következő szakasz kiindulópontjával, akkor az írószerkezetet fel kell emelnünk a papírról vagy egy olyan LIN utasítással, amely csak a Z értéket változtatja, vagy pedig a felső kar mozgatásával PTP-utasítás segítségével. Miután az összes pontot beprogramoztuk, zárjuk a programot a következőkkel:

FIX 1 (így egybe láthatjuk az egész rajzot)
WAIT D 1 (ebben a helyzetben az írószerkezet könnyebben eltávolítható)
RPT (a program ismétlése a digitális bemenet "1" értéke vagy az ESC megnyomása által)

Mentsük ki a programot kilépve a szerkesztő üzemmódból, majd teszteljük "szárazon", azaz toll nélkül. Az első éles futás valószínűleg nem lesz teljesen kielégítő (vonalak átfedése, stb.). Futtassuk a programot lépésenként szerkesztő üzemmódban (ROB2R on-line). Előfordulhat, hogy nemcsak az egyes paraméterek módosítására, de további PTP és LIN utasítások beszúrására is szükség lehet. Gondos munkánk eredménye még mindig távol van a tökéletestől. Ennek okát korábban (lásd 2.2) már vázoltuk (a pozícióméró-rendszer alacsony felbontása, pontatlanság a hibajavításban, a pozícióméró-rendszer egyes potenciométereiben lévő egyenetlenségek, valamint a fogó viszonylag instabil pozícionálása az aktuális koordináta-rendszer y tengelye fölött.)

Az ábrán látható írás 1 mm vonalvastagsággal és tízszeres ismétléssel készült.

A memóriában található EXP6.RLN példaprogram listája robotkoordinátákban is és Descartes-koordinátákban is meg van adva. Természetesen itt is szükség lehet tesztelésre és paraméterjavításra.

TEACHBOX

1. Bevezetés

A következő fejezetekben megismerjük, hogyan lehet számítógép nélkül a ROB3-at kézzel vezérelni, mozgatni.

A robot programozása a beépített vezérlés alapján a *Teachbox* tasztatúrája segítségével történik. Egyszerű utasításkészletet felhasználva egymás utáni pozíciók sorozatát állíthatjuk be, majd mozgásként tárolhatjuk.

Egy programon belül elágazási pontokat helyezhetünk el, ezáltal a különböző, eltárolt programok láncbafűzése lehetséges. Ezeket a programrészeket a 8 digitális bemenet lekérdezése függvényében szólíthatjuk meg - ilyen módon a robot külső eseményekre reagálhat. A hasonlóan beépített 8 digitális kimeneten keresztül a robot a programnak megfelelően a környezetét is vezérelheti. Lehetséges az időzítések beállítása, ciklusok programozása, stb.

Aki szívesen dolgozik számítógéppel, a ROB3-at bármely IBM-kompatibilis PC-vel vezérelheti az RS232 porton keresztül. A mellékelt software menü-vezérelt, a lehetséges utasítások listája folyamatosan

látható, kurzor segítségével a program listájához fűzhető. A rendszer ellenőrzi a szintaktika helyességét, a könnyen kezelhető *súgó* hatékony segítséget nyújt a programozáshoz.
BASIC, PASCAL és C nyelven írt rutinok állnak rendelkezésre *include-file*ok formájában, amelyek a soros portot kezelik.

2. Műszaki adatok

Ismétlési pontosság	: ±1 mm
Sebességi fokozatok száma	: 5
Megengedett teher	: 250 g
Legnagyobb sebesség	: 450 mm/s
Legnagyobb pályamenti sebesség	: 90 mm/s
Meghajtás:	egyenáramú szervomotor
Visszajelzés:	abszolút
soros port:	RS 232
párhuzamos port:	8 digitális be és kimenet
Tápfeszültség:	9V/3A DC

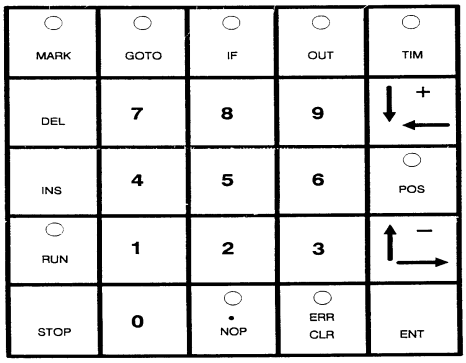
A programozás *Teachbox*, IBM-kompatibilis PC, XT, AT és TBPS-software, vagy saját software az RS232-portot kezelő rutinok segítségével.

3. Méretek és elrendezés

(Robot leírásban az 1.ábrán láthatóak)

4. A Teachbox

A ROB3 kiszolgálása számítógép nélkül is lehetséges a *Teachbox* segítségével. Ez egy 25 gombos tasztatúra, amely decimális számjegyek, parancsok és utasítások bevitelére alkalmas. A *Teachbox* a ROB3 I/O portjára csatlakozik, és a program indítása után kényelmesen le is vehető onnan. Rácsatlakozott *Teachbox* esetén 3 digitális kimenet (DA 1-3) és 5 digitális bemenet (DE 1-5) marad szabadon. Az utasításgombok mellett levő LEDek a helyes működést jelzik. Hibás utasítások vagy parancsok bevitele esetén az ERR lámpa gyullad ki, amely a CLR gombbal törölhető. Az utasításokat és parancsokat mindig az ENT gomb lenyomásával zárjuk. C.5. ábra:



*számjegy gombok: tengelyek és pozíciók beadása

*Parancsok (RUN, STOP, INS, DEL, POS, OUT): azonnal végrehajtásra kerülnek.

*Utasítások (POS, TIM, OUT, MARK, IF, GOTO, NOP): csak a futás (RUN vagy STEP) során kerülnek végrehajtásra. Az ENT gomb lenyomása csak a tárolást eredményezi.

5. Program-üzemmódok

- **Beviteli** üzemmód: A bekapcsolás után automatikusan ebben az üzemmódban vagyunk.
- Utasítások beadása.

- **Pozícionáló** üzemmód: A számjegy-gombok, ill. a gomb segítségével a tengelyek ill. pozícióik változtatása lehetséges. Direkt pozícionálás a POS (tengely - **a**). (pozíció - **n**) ENT sorrenddel történik. A **beviteli** üzemmódba a CLR gombbal térhetünk vissza.

- **Run** üzemmód:
  - A RUN. (marker - **m**) ENT gombok lenyomására a tárolt program végrehajtása történik
- az **m** markertől kezdődően.

- **Break**-üzemmód: **run** üzemmód esetén csak a STOP gomb aktív. A STOP, CLR gombok lenyomásával kerülhetünk **break** üzemmódba. A RUN, ENT gombokkal visszakerülünk a **run** üzemmódba; a STOP, ENT gombokkal pedig a többi üzemmód érhető el.

- **Step**-üzemmód: A RUN. (marker - m) ENT gombokkal a **step** üzemmódba kerülünk az **m** markernél. Innen a + gombbal lépésenként hajthatjuk végre a programot. A CLR gombbal kilépünk a **step** üzemmódból.

- **Teszt**-üzemmód: Ide a STOP. (marker - **m**) ENT gombok lenyomásával kerülhetünk. A + és a - gombokkal a tárolt programlépéseken mehetünk végig, anélkül, hogy azok végrehajtódnának.
- A lépéseket a LED-ek jelzik. A CLR gombbal lépünk ki.

6. Az utasítások és parancsok áttekintése

Utasítások:

MARK **m** ENT	**m** markert helyez el.	m = 0 .. 118
POS ENT	pozíciót állít be	
TIM **t** ENT	várakozási időt állít 100 ms-nként	t = 1 .. 65535
GOTO **m** ENT	**m** markerre ugrik, végtelen ciklus	
GOTO **m . n** ENT	**m** markerre ugrik **n**-szer, n = 0 .. 255	
IF **i** ENT	várakozás i=low bemenetre	
IF **i . m** ENT	ha i=low, folytassa a programot **m** markertől, egyébként haladjon tovább	
OUT **k** +/- ENT	**k** kimenetet állítson/töröljön, k = 1 .. 8	
. ENT	NOP	
STOP 0 ENT	programstart	
INS **.** ENT	programvég	
DEL **.** ENT	fél-programvég; két program elválasztása	

Parancsok:

a +/- ENT	**a** tengelyt pozíció üzemmódba mozgatni
POS **a . n** ENT	**a** tengelyt **n** pozícióba mozgatnia = 1.. 6
OUT **. k +/-** ENT	**k** kimenetet közvetlenül állítson/töröljön
RUN **m** ENT	**m** markertől fut
RUN **. m** ENT	**m** markertől **step** üzemmódban fut
RUN ENT	(**break** üzemmódban) continue
RUN **.** ENT	(**step** üzemmódban) continue
STOP CLR	break, program leállás (tovább: RUN ENT)
STOP ENT	stop, program leállás; átmenet **bevitel** üzemmódba
STOP **.** ENT	átmenet **teszt** üzemmódba
STOP **. m** ENT	átmenet **teszt** üzemmódba, **m** markertől
INS ENT	utasítás beszúrása
DEL ENT	utasítás törlése
CLR	üzemmódváltás; hibatörlés

7. Az egyes utasítások magyarázata

- A programozás kezdete

STOP 0 ENT	programstart. Törli a tárat. Csak az első program kezdetén adható ki, különben az utasítást megelőző programrészeket is kitörli.
MARK 0 ENT	A 0 markert beteszi - ez a program kezdőcíme
CLR	Beviteli üzemmódba vált

Ezután következik maga a program (mozgások, ciklusok, be- és kiviteli utasítások).

- mozgások

a +/-	**a** tengely körüli jobbra-balra forgatás. Mivel a különböző tengelyek körüli forgatások egyidejűleg hajtódnak végre, egyszerre több tengely is pozícionálható ily módon.
CLR	Váltás **beviteli** üzemmódba
POS ENT	A végrehajtott mozgás tárolása
POS (**a** tengely) **.** (pozíció **n**) ENT	**a** tengely direkt forgatása az **n** pozícióba. Hasonlóan a relatív forgatáshoz, egyszerre több tengelyt is beállíthatunk a kívánt helyzetbe.
CLR	Váltás **beviteli** üzemmódba

- ciklusok

MARK **m** ENT **m** markert cikluskezdetre állít meg.

GOTO **m . n** ENT ugrás a cikluskezdetre. i=1 to **n**

- be- és kimeneti (I/O) utasítások

IF **i** ENT	várakozás, amíg az **i** bemenet LOW lesz
IF **i . m** ENT	ha i=low, a program az **m** markertől folytatódik, egyébként az utasításnak nincs hatása.
OUT **k** +/- ENT	**k** kimenetet állít/töröl. (Negatív TTL logika: "állít=set" esetén a kimenet LOW, "törlés = clear" esetén HIGH
OUT **. k** +/- CLR	Ez az utasítás az I/O portot teszteli a programírás vagy - megszakítás idején. Vigyázat, ha ENT-tel lezárjuk, akkor az ott lévő sort felülírja!!

- várakozás

TIM **t** ENT	Az utasítás hatására a program futása **t** x100 ms ($0 \le t \le 65535$) időre megszakad, majd automatikusan folytatódik.

- programmegszakítás

DEL **.** ENT Az utasítás hatására a program futása megszakad, és a RUN lámpa kigyullad. Az ENT gomb megnyomására a program újraindul. Az utasítás több egymást követő lineáris programot (ciklusmentes ?) is képes elválasztani. A CLR gomb a **bevitel** üzemmódba vált.

- program vége

INS **.** ENT Ez az utasítás kötelező minden program fizikai végén.

- teszt-üzemmód

STOP **. m** ENT /
STOP **.** ENTAz utasítás hatására a tárban levő utasításokat a megfelelő LED-ek mutatják (MARK,POS, TIM, OUT, IF, NOP) illetve a DEL utasításra a MARK és a NOP-LED, hiba esetén pedig az ERR-LED gyullad ki.
A CLR gombbal hagyjuk el az üzemmódot.

- beszúrás

INS ENT Az utasítással lehetőség van egy vagy több programsor beszúrására. Teszt-üzemmódban a kívánt sort szúrjuk be a további sorok ELÉ. Több sor esetén ismételten nyomjuk meg az INS ENT gombokat.

- törlés

DEL ENT A bevitel-utasításokhoz hasonlóan a kívánt sorra állunk. A mutatott sor az utasítás hatására törlődik, és a következő sorok eggyel előre lépnek. Ilyenformán az ismételt törlés is mindig a programvég felé történik.

- hozzáírás

Egy már lefutott program egyszerűen továbbírható olymódon, hogy a programvéget felülírjuk. A **teszt**-üzemmódban menjünk a program végére: az ERR-LED itt kigyullad. A
CLR gombbal váltsunk **bevitel-**üzemmódra, és adjuk be a további utasításokat, amelyek közül az utolsó természetesen az
INS **.** ENT kell legyen.

- felülírás

Korábban bevitt NOP utasítások, meglévő programrészek és
INS ENT révén beszúrt NOP utasítások egyszerűen felülírhatók.
A **teszt**-üzemmódban e célból menjünk felülírandó helyre, a
CLR gombbal váltsunk **bevitel-**üzemmódra, és adjuk be az új utasításokat.

- futás-leállás

RUN **m** ENT	Futás az **m** markertől kezdve
STOP CLR	A program leállítása tetszőleges helyen
RUN ENT	Folytatás a leállás helyétől. Egyes lépésenként a
RUN **. m** ENT	paranccsal, majd a + gombbal hajtható végre a program.

8. Üzembehelyezés és kikapcsolás-szállítás.

A *Teachbox* üzembehelyezéséhez szüksége van az alábbiakra:

- a ROB3,
- hálózat 2 piros/fekete összekötő kábellel,
- Teachbox,
- RS232-csatlakozó.

A 2 összekötő kábellel kössük be a hálózatba a ROB3-at. A Teachbox 25-pólusú csatlakozóját az I/O portba, a 9-pólusút pedig az RS232 portba dugjuk. A hálózati kapcsoló bekapcsolása után nyomjuk meg a RESET gombot a ROB3-on. A robot innen kezdve programozható vagy közvetlenül vezérelhető.
A programtár kikapcsolás után is megőrzi tartalmát egy Lítium-elem segítségével.

Figyelem:

A robotot még kikapcsolt állapotban sem szabad kézzel megmozdítani, mert a mechanikája károsodhat. Ha szállítható állapotba szeretnénk hozni, állítsuk ROB3-at a S1=128, S2=164, S3=140, S4=182, S5=14, S6=128 pozícióba!