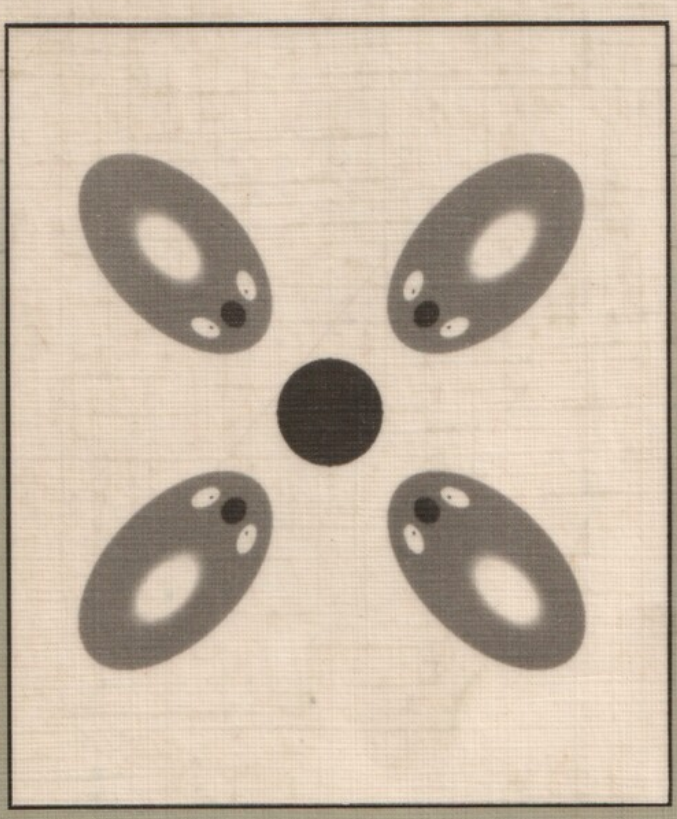
JARMO ALANDER - KARI NIEMI

ROBOTTIEN OHJAUS JA OHJELMOINTI



497

OTAKUSTANTAMO

DK 681.532.8 : 621.865.8
681.3.06 : 579.68

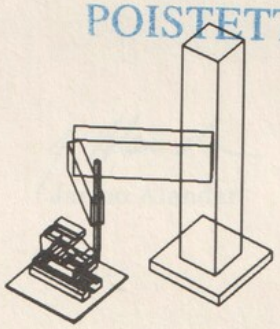
JARMO ALANDER - KARI NIEMI

ROBOTTIEN OHJAUS JA OHJELMOINTI

497

TKK KIRJASTO

POISTETTU



206384

TEKNILLISEN KORKEAKOULUN KIRJASTO
TEKNISKA HÖGSKOLANS BIBLIOTEK

OTAKUSTANTAMO

Tämän kirjan tekstin ja kuvien jäljentäminen ilman kustantajan ja tekijän lupaa painamalla, monistamalla, äänittämällä, valokuvaamalla tai muulla tavoin kielletään 8. heinäkuuta 1961 annetun tekijänoikeuslain nojalla.

ISBN 951—672—028—5

© Tekijät ja Otakustantamo 1987

Painopaikka:
Vaasa Oy
Vaasa 1988

ESIPUHE

Käsillä oleva kirja on syntynyt tekijöiden toimiessa robotiikan tutkijoina Teknillisen korkeakoulun Tietojenkäsittelyopin laboratoriossa. Tutkimusaiheina on ollut erityisesti tietokonemallien hyödyntäminen robottien ohjauksessa ja ohjelmoinnissa. Tutkimukseen on liittynyt teollisuuden kanssa tehty kokoonpanotehtävien rationalisointi- ja automatisointihanke. Kirja heijastelee monilta osiltaan tekijöiden kokemuksia ja ajatuksia tästä työtehtävästä ja suoritetuista kokeiluista. Erityisesti kokoonpanosovellutusten käsittely onkin laajempaa kuin mihin näiden sovellutusten nykyinen merkitys antaisi aihetta. Toisaalta kokoonpano sisältää monia vaikeita ja mielenkiintoisia ohjausongelmia. Lisäksi kokoonpano on erittäin tärkeä tuotantoprosessi, jonka merkitys robotiikassa kasvaa nopeasti tulevina vuosina.

Tekijät ovat toimittaneet kirjan painokuntoon LaTeX-nimisellä ohjelmistolla [Lam86][1]. Lopuksi haluamme vielä kittää kaikkia niitä henkilöitä ja organisaatioita, jotka ovat olleen apuna kirjaa tehtäessä.

Espoossa 16. 11. 1987 Jarmo Alander Kari Niemi

[1] Painovirheistä yms voi siten syyttää suoraan kirjoittajia.

Sisällys

1 JOHDANTO 7

1.1 Ihminen ja kone 8
1.2 Määritelmiä 9
1.3 Manipulaattori 10
1.4 Tarttujat 12
1.5 Anturit ja aistimet 13
1.6 Ohjausjärjestelmä 14
1.7 Ympäryslaitteet 15
1.8 Robottien käyttöalueet 16

2 OHJELMOINTI 19

2.1 Ohjelmointimenetelmät 19
2.1.1 Ohjelman suunnittelu 22
2.2 Robottien ohjelmointikielet 24
2.2.1 Yhteys yleisiin kieliin 24
2.2.2 Robottikielien ryhmittely 25
2.2.3 Robottikielien yleisiä ominaisuuksia 28
2.2.4 Robottikielien heikkouksia 29
2.3 Ohjelmointiesimerkki 32

3 MALLIT 35

3.1 Tuotannon mallittaminen 36
3.2 Tuotemalli 37
3.3 Hierarkiat 38
3.4 Ongelmanratkaisu 39
3.4.1 Tuotemalli sääntöinä 39
3.4.2 Sääntöjen etuja 40
3.5 Tehtaan ohjaus 41
3.5.1 Hierarkinen järjestelmä ja optimointi 41
3.6 Esimerkkejä 43
3.6.1 Valumallin työstö 43
3.6.2 Muuraus 45
3.6.3 Keskusten kokoonpano 48

4 ROBOTTIKIELIEN KUVAUS 53
4.1 AL 55
4.2 AML 58
4.2.1 AML/Entry 60
4.2.2 AML/2 63
4.3 ANORAD 64
4.4 ARL 64
4.5 ARLA 65
4.6 AUTOPASS 69
4.7 BAPS 71
4.8 EMILY 73
4.9 FUNKY 73
4.10 HELP 74
4.11 IRDATA 74
4.12 IRL 75
4.13 JARS 75
4.14 JOB 76
4.15 LAMA-S 76
4.16 LM 77
4.17 LPR 80
4.18 MAL 81
4.19 MAPLE 81
4.20 MCL 82
4.21 MHI 84
4.22 MINI 85
4.23 ML 86
4.24 PAL 87
4.25 PASRO 88
4.26 PLAW 89
4.27 Prolog 92
4.28 RAIL 93
4.29 RAPT 95
4.30 ROBEX 96
4.30.1 ROBEX-M 96
4.31 ROL 99
4.32 RPL 99
4.33 RSS 100
4.34 RVL/A 105
4.35 SERF 106
4.36 SIGLA 106
4.37 SRL 106
4.38 STROL/STROLIC 107
4.39 TEACH 107
4.40 TL-10 108
4.41 T3 110
4.42 VAL 111

4.42.1 VAL-II 112
4.43 WAVE 114
4.44 Yamaha 116

5 KIRJALLISUUTTA **121**

6 SANASTOA ja LYHENTEITÄ **125**

7 HAKEMISTO **127**

JOHDANTO

Useat kansalliset ja kansainväliset robottijärjestöt esittivät 80-luvun alussa optimistisia arvioita robottikannan kehityksestä. Vaikka näiden ennusteiden toteutuminen kaikilta osiltaan näyttääkin epätodennäköiseltä on robottien merkitys teollisuudessa selvästi kasvanut ja tulee varmasti edelleen nopeasti kasvamaan. *robottien merkitys*

Eräs kehitystä pahimmin jarruttava tekijä on ohjausjärjestelmien kehittymättömyys. Nykyiselle tilanteelle on tyypillistä, että robotin ohjausyksikkö hoitaa koko solun ohjauksen. Esimerkiksi kokoonpanosolussa on robotin usein huolehdittava myös kuljetinjärjestelmän, työkalujen ja komponenttien syötön ohjauksesta, jotka luonteeltaan poikkeavat monessa suhteessa pelkän robotin ohjauksesta, jota varten ohjaus on alunperin ja pääasiassa suunniteltu. Toisaalta tarvittavia oheislaitteita ei ole tavallisesti suunniteltu robottikäyttöön, mikä puolestaan hankaloittaa ohjausjärjestelmää. *ohjausjärjestelmä*

Ohjauksen onnistunut toteutus edellyttää robotin ohjausyksiköltä ominaisuuksia, joita niillä ei vielä useinkaan ole. Eräänä tärkeänä piirteenä mainittakoon kyky rinnakkaiseen prosessointiin. Tällä hetkellä näyttää kuitenkin siltä, että tulevaisuudessa robotin ohjausyksikkö hoitaa vain robotin ohjaamista. Solujen ohjaus siirtyy erillisille mikrotietokoneille, jotka toteuttavat solujen ohjausta ylemmältä ohjaustasolta tulevien tietojen perusteella. Tämä kehitys on sopusoinnussa tietotekniikassa nykyisin vallitsevan hajautuskehityksen kanssa. Pyritäänhän vaativat tietojenkäsittelytehtävät kapasiteettiongelmien välttämiseksi jakamaan useammaksi helpommin hallittavaksi osatehtäväksi, jotka edelleen sijoitetaan rinnakkain toimiville yksiköille. *hajautus*

Robottien ohjauksessa ohjelmointimenetelmillä ja -kielillä on huomattava merkitys. Nykyisin käytössä olevat robottien ohjelmointimenetelmät vaihtelevat yksinkertaisesta, käsin tapahtuvasta opetuksesta monipuoliseen tietokoneavusteiseen suunnitteluun ja ehkä tekoälyyn perustuvaan ohjelmointiin. Viimeksimainittu tapa on kuitenkin vielä harvinainen. Lähinnä tekstuaalisessa ohjelmoinnissa käytettävien robottikielien määrä on suuri. Lähes jokaisella robottitoimittajalla on oma kielensä, josta on lisäksi olemassa versioita mm. robotin tyypin ja tehtävän mukaan. *ohjelmointi*

Tässä kirjassa tarkastellaan robottien nykyisiä ohjaus- ja ohjelmointimenetelmiä sekä robottikieliä. Samalla arvioidaan todennäköisiä kehitysnäkymiä. Esitys ei korvaa robottikielikohtaisia manuaaleja, joita tarvitaan, jos halutaan laatia omia ohjelmia.

1.1 Ihminen ja kone

kuvitelmia

Erityisen mielenkiintoisia robotit ovat siksi, että ne niin paljon muistuttavat ihmistä. Näin on ainakin tieteiskuvitelmissa. Todellisuudessa robottia rasittavat kuitenkin lähestulkoon kaikki ne puutteet mitkä muitakin koneita. Murrell [Mur82] on vertaillut ihmiselle ja koneelle sopivia tehtäviä. Seuraavat tehtävät sopivat hänen mukaansa yleensä paremmin ihmiselle kuin koneelle:

- minimaalisten ärsykkeiden havaitseminen. Erityisesti ihmisen kuulo ja näkö ovat äärimmäisen kehittyneitä.
- asioiden laaja-alainen käsittely (ns. terve järki)
- improvisointia ja joustavuutta vaativat tilanteet. Koneet ovat ennalta ohjelmoituja.
- toiminnasta toiseen siirtyminen (useimmille koneille mahdotonta)
- pitkäaikaista muistia vaativat tehtävät
- avaruuden, syvyyden ja kokonaisuuden havaitsemista vaativat tehtävät
- yksityiskohtien arviointia vaativat tehtävät
- arviointia ja johtopäätöksiä vaativat tehtävät
- käsitejärjestelmästä toiseen siirtyminen
- induktiivista ajattelua vaativat tehtävät
- monimutkaisten arvioiden ja päätösten teko
- sisäisen tasapainon säilyttämistä vaativat tehtävät

Sen sijaan seuraavat tehtävät ovat hänen mukaansa yleensä koneelle soveliaampia:

- laskenta
- erottelu
- yhteensovittaminen
- nopeat reaktiot

- suuren voiman käyttö tarkasti ja varovasti
- tarkka toisto
- lyhytaikainen muisti
- deduktiivinen ajattelu (vrt. tekoäly ja säännöt)
- monimutkaiset samanaikaiset toiminnot. Ihminen pystyy keskittymään kerrallaan vain pariin tehtävään[1].
- yksinkertaiset kyllä/ei-päätökset

robotti vs. kone

Edellä esitetyt seikat on syytä pitää mielessä tutkittaessa robotisoinnin, tai yleensä automatisoinnin, mahdollisuuksia ja on ehkä hyvä huomata se itsestään selvä seikka, että *robotilla on yleensä helpompi korvata kone kuin ihminen.* On luultavaa, että osa tavanomaisista koneista tulee vähitellen korvautumaan joustavammilla konstruktioilla, joissa robotilla ja sitä ohjaavalla tietokoneella tulee olemaan keskeinen rooli. Mekaniikkaa tuskin kannattaa suunnitella aina uudelleen varsinkaan pieniä konesarjoja valmistettaessa. Kiinteään automaatioon nähden pienempi kapasiteetti tietysti omalta osaltaan hillitsee tämän kaltaista kehitystä. Suurempi este lienee kuitenkin se, että robottitekniikan sovellutusmahdollisuuksia ei pystytä havaitsemaan.

1.2 Määritelmiä

> Robotti — Ulkonaisesti ihmisen näköinen, useimmiten hyödyllistä työtä suorittava malli tai automaatti – V. Pekelis, [Pek83].

Robot Institute of America on määritellyt teollisuusrobotin seuraavasti: Robotti on uudelleen ohjelmoitavissa oleva monipuolinen[2] mekaaninen laite, joka on suunniteltu ohjelmoitavilla liikkeillä liikuttamaan kappaleita, osia, työkaluja tai erikoislaitteita.

Suomen Robotiikkayhdistys ry on tehnyt määritelmään seuraavat rajoitukset:

- robotin vapausasteiden tulee olla servo-ohjattuja
- vapausasteita on vähintään kolme ja lisäliike
- kaikki scara[3]-tyypin robotit luetaan teollisuusroboteiksi

[1] usein ei yhteenkään
[2] erilaisia tehtäviä
[3] scara = selective compliance assembly robot arm

Robotin määritelmä vaihtelee eri maissa huomattavastikin. Esimerkiksi japanilainen robotti-käsite on edellistä selvästi vapaamielisempi lukien teollisuusroboteiksi myös kahden vapausasteen ohjelmoitavat siirtimet. Erot määritelmissä keskittyvät ohjelmoitavuuden helppouteen ja liikkeiden monipuolisuuteen.

Robottien käyttö teollisuudessa on yleistynyt nopeasti viime vuosina, ja kasvun odotetaan edelleen jatkuvan. Eräiden ennusteiden mukaan robottien määrä kymmenkertaistuu kymmenessä vuodessa. Suomessa robottikannan kasvu on 30-50% vuosittain ja on ennustettu, että vuonna 1990 Suomessa on noin 1500 robottia. Maailman robotisoitunein maa on Ruotsi, jossa vuonna 1982 oli 43 robottia 10000 työntekijää kohti. Suomessa vastaava suhdeluku oli alle 1.

Robottien käyttö hyvinkin erilaisissa tehtävissä on aiheuttanut erilaisten robottityyppien kehittymisen. Suurimmat, järeät robotit voivat kantaa yli tuhannen kilon kuormia. Näitä käytetään teräsvalimoissa ja muussa raskaassa teollisuudessa.

Maalausrobotit on tarkoitettu kappaleiden yhtenäiseen maalaukseen ja lakkaukseen. Ne ovat nopeita liikkeissään, mutta eivät erityisen tarkkoja. Piirtämistä tai muuta kuviointia niillä ei yleensä pysty tekemään ilman apuvälineitä.

Prosessirobotissa on monipuolinen ohjausjärjestelmä ja antureita, joiden avulla robotista voidaan tehdä itseohjautuva. Tyypillinen prosessirobotin tehtävä on automaattisen työstökoneen palvelu eli työstettävien aihioiden ja valmiiden kappaleiden siirto koneeseen ja siltä pois.

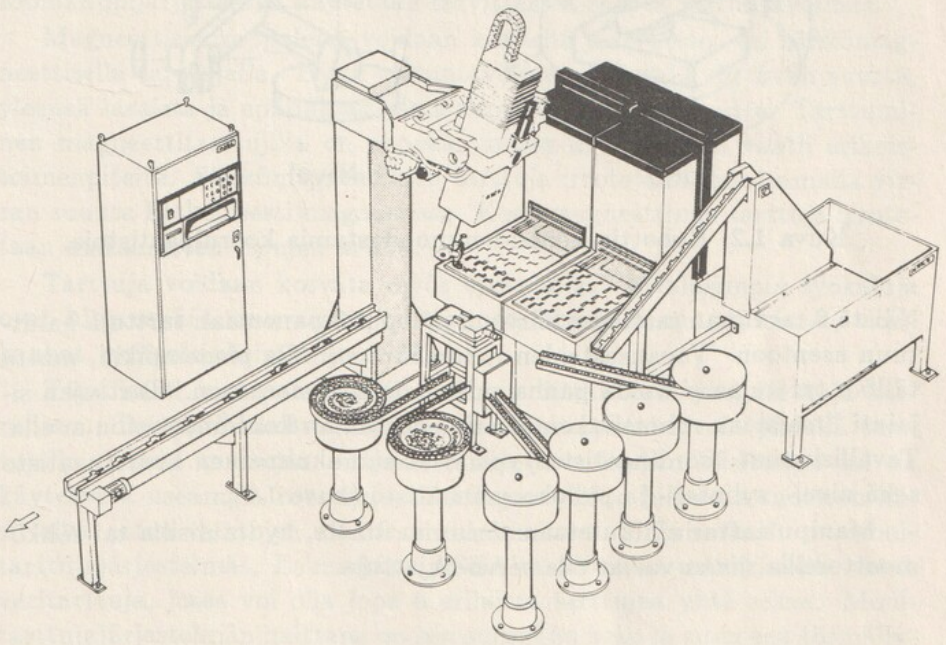
Kokoonpanorobotit ovat nopeita ja niissä on myös kehittynyt ohjausjärjestelmä. Muita hyvän kokoonpanorobotin ominaisuuksia ovat: lyhyet kappaleajat (2,5 - 10 s/kpl), pieni työkappaleeen paino, suuri tarkkuus (±0,05 mm), edullinen hinta/suorituskykysuhde ja yksinkertainen käyttö. Portaali- ja lineaariliikkeiset robotit täyttävät suuren osan näistä vaatimuksista. Kuitenkin scara-tyyppisellä kokoonpanorobotilla saavutetaan erityisen hyvä peitto edellä esitetyille vaatimuksille.

Hyvin yksinkertaisissa työtehtävissä voidaan käyttää *siirrintä*. Robotti voidaan koota myös moduleista. Nämä yleensä paineilmaohjatut modulaariset robotit ovat nopeita ja halpoja, mutta eivät kovinkaan joustavia, jos tehtävät vaihtuvat. Myös opetukseen on kehitetty erityisiä opetusrobotteja.

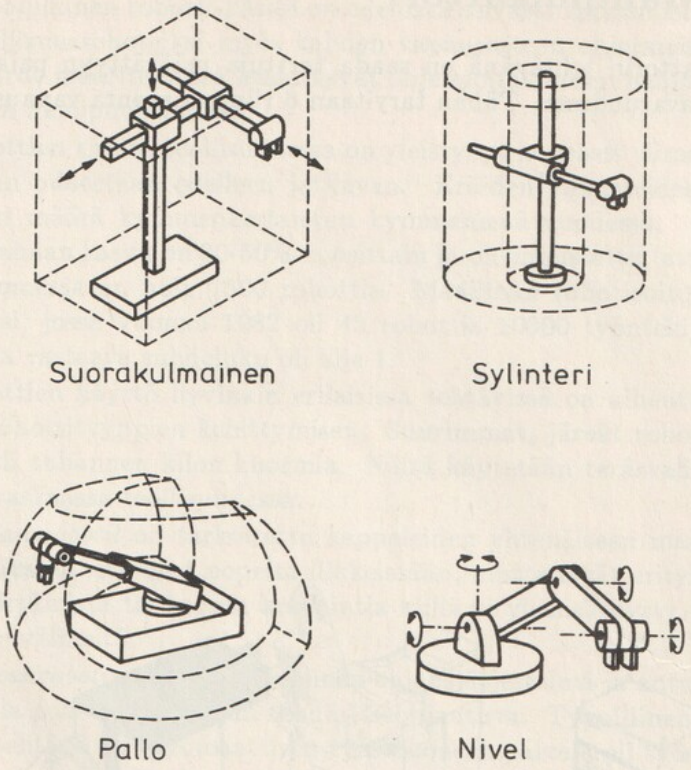
Tyypistä riippumatta robotissa voidaan erottaa seuraavat osat: manipulaattori, tarttuja, anturit ja aistimet sekä ohjausjärjestelmä. Lisäksi järjestelmään kuuluu ympäryslaitteita, kuten kuljettimia ja syöttölaitteita (kuva 1.1).

1.3 Manipulaattori

Manipulaattorin tehtävänä on saada tarttuja määrättyyn paikkaan ja asentoon avaruudessa. Tähän tarvitaan 6 riippumatonta vapausastetta.



Kuva 1.1: Esimerkki robottijärjestelmästä (ASEA).



Kuva 1.2: Robottirakenteiden muodostamia koordinaatistoja.

Näistä 3 tarvitaan tarttujan siirtoon ja loput 3 panemaan tarttuja haluttuun asentoon. Vapausasteiden lukumäärä voi olla pienempikin, mutta tällöin tarttujaa ei voida panna mielivaltaiseen asentoon. Tarttujan sijainti ilmaistaan robottirakenteen muodostaman koordinaatiston avulla. Tavallisimmat koordinaatistotyypit ovat suorakulmainen koordinaatisto sekä nivel-, sylinteri- ja pallokoordinaatisto (kuva 1.2).

Manipulaattoria liikutetaan pneumaattisilla, hydraulisilla tai sähkömoottoreilla, jotka voivat olla servo-ohjattuja.

1.4 Tarttujat

toimitapa

rakenne

mekaaniset

Roboteissa käytettävät tarttujat voidaan ryhmitellä useiden ominaisuuksien perusteella. Jako voidaan tehdä tarttujan toimitapaan perustuen kaksiasentoisiin, jatkuvatoimisiin, pneumaattisiin, hydraulisiin tai sähköisiin tarttujiin. Jako voidaan tehdä myös tarttujan rakenteen mukaan mekaanisiin, alipaine- ja magneettitarttujiin. Lisäksi käytössä on erikoisratkaisuja, kuten pisto-, adhessiivinen ja puhallustarttuja.

Suurimman ryhmän muodostavat mekaaniset tarttujat. Niissä tartunta voi olla jäykkä, joustava tai itsekeskittävä. Yleisin on keskittävä

tai kappaletta puristava jäykkä tartunta. Niveliä tarttujassa on tavallisesti kaksi. Lisäämällä niveliä saadaan aikaan esimerkiksi tartuntapintojen yhdensuuntaisliike. Tarttujien sormet voidaan päällystää tartuntaa helpottavalla materiaalilla kuten muovilla tai kumilla. Sormet voidaan myös muotoilla työkappaleen mukaan.

pneumaattinen

Tavallinen pneumaattisesti toteutettu joustava tarttuja on toiminnaltaan yksinkertainen, käyttövarma ja halpa. Yksinkertaisimmillaan joustava tarttuja on toisesta päästä umpinainen kumiholkki, joka joko paisutetaan työkappaleessa olevassa reiässä tai kutistetaan esimerkiksi akselinpään ympärille. Monimutkaisempi rakenne on kumisormitarttujilla, joilla voidaan tarttua muodoltaan hyvinkin vaihteleviin kappaleisiin.

Itsekeskittävässä tarttujassa on tunto- tai joustoelin, joka keskittää työkappaleen haluttuun paikkaan. Tällaista tarttujaa voidaan käyttää työssä, jossa robotin kohdistustarkkuus ei täysin riitä, esimerkiksi ruuvien asettamiseen ruuvinreikään.

imukuppi

Alipainetarttuja on tyypillisesti kumista, muovista tms. materiaalista valmistettu imukuppi, jolla tartutaan käsiteltävään kappaleeseen. Monikuppijärjestelmiä käytetään tarvittaessa suurta tartuntavoimaa.

magneetti

Magneettisia kappaleita voidaan käsitellä joko kesto- tai sähkömagneettisella tarttujalla. Hyvä tartuntavoima edellyttää riittävän suurta, yleensä tasaista ja epäpuhtauksista vapaata tartunta-aluetta. Tarttuminen magneettitarttujilla on nopeaa, mutta irrottaminen vaatii erikoistoimenpiteitä. Sähkömagneettinen tarttuja irrotetaan muuttamalla virran suunta hetkellisesti magneetissa. Kestomagneettinen tarttuja irrotetaan mekaanisten vipujen avulla.

työkalut

Tarttuja voidaan korvata myös työkalulla. Tavallisimpia työkaluja ovat hitsauspäät, ruuvaimet, niittaimet, kuumentimet, jyrsimet, valukauhat, polttoleikkaimet, jne.

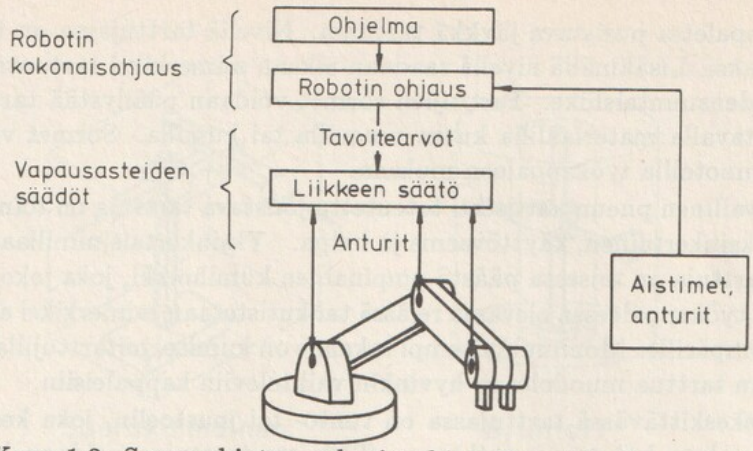
revolveri

Tavallisesti manipulaattoriin voidaan liittää vain yksi tarttuja kerrallaan. Tuotteen käsittelyssä selvitään kuitenkin harvoin yhdellä tarttujalla, jolloin tarttujaa on joko vaihdettava kesken käsittelyn tai on käytettävä useampia robotteja. Tarttujan vaihto hidastaa tuotteen käsittelyä, joten vaihtojen määrä on minimoitava. Eräs ratkaisu on monitarttujajärjestelmät. Esimerkiksi ASEAlta on tullut markkinoille revolveritarttuja, jossa voi olla jopa 6 erilaista tarttujaa yhtä aikaa. Monitarttujajärjestelmän haittana on sen suurehko koko ja siten sen törmäilyalttius.

1.5 Anturit ja aistimet

analoginen/ numeerinen

Kehittyneisiin robottijärjestelmiin kuuluu runsaasti erilaisia antureita ja aistimia. Robotin paikkaa, nopeutta tai kiihtyvyyttä mittaavan anturin arvoa voidaan käyttää sellaisenaan robotin toiminnan ohjaamiseen. Tavallisesti anturit jaetaan analogisiin ja numeerisiin antureihin. Jako voidaan tehdä myös mittaustavan tai sen perusteella, voidaanko mittauk-



Kuva 1.3: Servo-ohjatun robotin ohjausjärjestelmän periaate.

sesta suoraan lukea absoluuttiarvo vai muutos edelliseen mittausarvoon nähden.

aistin

Aistin eroaa anturista siinä, että sen antaman informaation käyttö vaatii yleensä tiedon edelleen prosessointia. Aistimia käytetään yleensä robotin ympäristön havainnointiin. Mitattavia suureita voivat olla esimerkiksi kohteen etäisyys, kosketus, kosketuksen paikka, voimat, vääntömomentit, kohteen luonteenomaiset piirteet tunnistusta varten ja kohteen muoto. Tällä hetkellä erityisen mielenkiinnon kohteena on tietokonenäön käyttö.

1.6 Ohjausjärjestelmä

tehtävät

Ohjausjärjestelmän tehtäviin kuuluu manipulaattorin liikkeiden ohjaus, tilannemäärityksen antaminen käyttäjälle, uusien määräysten vastaanotto käyttäjältä sekä ympäryslaitteiden kanssa kommunikointi. Ohjausjärjestelmän kannalta robotit voidaan jakaa kolmeen ryhmään: aidosti pisteohjatut, aidosti rataohjatut sekä yhdistetyt rata- ja pisteohjatut robotit.

ohjaustapa

Aidosti pisteohjatussa robotissa ohjausjärjestelmä on tavallisesti ohjelmoitava logiikka tai vastaava logiikkaohjaus. Servo-ohjausta käytetään aidosti rataohjatuissa ja yhdistetyissä rata- ja pisteohjatuissa roboteissa. Edellisessä kokonaisohjaus on melko yksinkertainen, mutta jälkimmäiseltä vaaditaan runsaasti tiedonkäsittely- ja laskentakapasiteettia. Servo-ohjauksella ohjausjärjestelmä on yleensä useamman tason hierarkinen järjestelmä, jossa ylemmällä, ohjelmatasolla ohjataan kokonaisliikettä ja alemmalla, säätötasolla ohjataan kunkin vapausasteen liikettä. Kuvassa 1.3 on kaksitasoisen servo-ohjauksen periaatteellinen malli.

johdattaminen

Yksinkertaisin robotin ohjelmointitapa on johdattaminen, jossa käyttäjä vie käsin tarttujaa haluttua rataa pitkin, joka sitten toistuu ohjelmaa ajettaessa. Robottia voidaan ohjelmoida myös kannettavalla ohjauslaitteella, jonka painonäppäimien tai ohjaussauvan avulla manipulaattori

siirretään toivottuun paikkaan. Käyttäjä antaa käskyt talletettavista sijainneista. Lisäksi hän voi ohjelmoida ratatyypin ja ratanopeuden.

Robottien joustavuuden ja soveltuvuuden parantaminen edellyttää kuitenkin tekstuaalista ohjelmointia päätteen ja ohjelmointikielen avulla. Tällöin ohjelmointi voidaan tehdä erillään robotista, joka voi samanaikaisesti toteuttaa aiemmin ohjelmoitua tehtävää. Tällaisessa ohjelmoinnissa on ohjelma testattava erityisen hyvin ennen käyttöönottoa esimerkiksi simuloimalla robottia ja sen työskentely-ympäristöä tietokoneella, jolloin ainakin osa virheistä paljastuu vahinkoa aiheuttamatta.

ohjelmointi

simulointi

Käytännössä ohjelmointi on usein yhdistelmä eri menetelmistä. Esimerkiksi robotti ohjelmoidaan tekstuaalisesti päätteellä ja ohjelman peruspisteet opetetaan käsiohjauksella. Robottien ohjelmointimenetelmiä ja -kieliä tarkastellaan lähemmin seuraavissa luvuissa.

hybridi menetelmä

1.7 Ympäryslaitteet

Robotti on laite, joka edellyttää joskus paljonkin ympäryslaitteita. Vaikka se usein onkin luonteeltaan muita laitteita palveleva, voidaan tällaista järjestelmää pitää ohjausteknisesti robottikeskeisenä. Tavanomaisia ympäryslaitteita ovat kuljettimet, syöttimet, välivarastot ja työstökoneet. Ympäryslaitteita ovat myös erilaiset anturit ja aistimet, jotka välittävät tietoa robotille ympäryslaitteiden tilasta.

Kuljettimella tuodaan työkappaleet robotin ulottuville. Kuljetin siihen rakennetun syöttimen kanssa huolehtii siitä, että kulloinkin määrättyyn paikkaan tulee kappale robotin poimittavaksi. Erilaisten aistimien avulla voidaan tinkiä syötön tarkkuusvaatimuksista.

kuljetin

Pakko-ohjatuissa kuljettimissa kappaleiden keskinäinen etäisyys on kiinteä, jolloin robotin ja kuljettimen tahdistaminen on helppoa. Vapaasti ohjatuissa kuljettimissa kappaleiden keskinäinen asema vaihtelee. Aistimien avulla voidaan tahdistaa robotin toiminta tuleviin kappaleisiin. Yksittäisohjatuissa kuljettimissa kappaleet viedään yksitellen prosessivaiheesta toiseen.

Syöttimien tehtävänä on asettaa työkappale tiettyyn paikkaan ja asentoon, jolloin robotti voi tarttua siihen oikealla tavalla. Tavallisimpia syöttimiä ovat makasinointilaitteet, keskipakoissyöttimet, erilaiset pyörivät siipipyörä-, mela- ja levysyöttimet, magneettisyöttimet, tärysyöttimet, jne. Myös ns. indeksipöytää voidaan pitää syöttimenä. Siinä työkappale viedään tarkoin määrättyyn asemaan robotin suhteen samalla kun vastakkaiseen kiinnityspisteeseen kiinnitetään uusi työkappale.

syötin

Välivarasto toimii tuotannon tasaajana eri laitteiden välillä. Taloudellisista syistä välivarastojen kokoa olisi pyrittävä minimoimaan pitäen kuitenkin samalla koneiden joutoaikaa mahdollisimman pienenä.

välivarasto

Toiminnallisesti robotit palvelevat usein erilaisia työstökoneita, jotka voidaan ryhmitellä kahdella tavalla. Tiukassa linjasovelluksessa tuote etenee tahdistetusti kaikkien työvaiheiden kautta tietyssä järjestyksessä.

työstökone

Linjassa robotteja on tavallisesti useita. Tuotantosolussa työstökoneet ovat ryhmittyneet robotin ympärille, ja robotin tehtävät ovat tavallisesti monipuolisempia kuin linjasovelluksessa. Koneiden määrän kasvaessa siirrytään joko linja- tai monisolujärjestelyyn.

kokoonpano

Kokoonpanossa robottia ympäröivät pääasiassa erilaiset komponenttien syöttimet, työkalut sekä valmiin osan poistavat kuljettimet. Erityisesti kokoonpanossa robotin joustavuudella on huomattava merkitys. Joustavuuden lähtökohta on järjestelmän ja komponenttien muunneltavuus niin, että kokoonpanon kohteena olevan sarjan suuruutta voidaan vaihtaa mielivaltaisesti pienimmän sarjakoon ollessa yksi. Lisäksi käsityövaiheiden ja automaatioasemien lukumäärän muuttamisen on oltava helposti toteutettavissa. Tällaisessa järjestelmässä on mahdollista käsitellä useita kokoonpanoyksiköitä samalla laitteistolla. Tyypillisesti joustava kokoonpanojärjestelmä muodostuu vapaatahtisista käsityöpisteistä ja automaattisista koneista.

1.8 Robottien käyttöalueet

tuotanto-määrä

Robottien soveltamisen taloudellisuutta voidaan tarkastella kuvan 1.4 esittämällä tavalla. Tästä havaitaan, että suurissa tuotantomäärissä erityisautomaatit tulevat kannattavammiksi. Vastaavasti pienillä tuotantomäärillä käsityö on taloudellisempaa. Työvoiman hinnan kohotessa ja robottien halvetessa tulee robottien käyttöalue laajentumaan erityisesti pienempien sarjakokojen suuntaan.

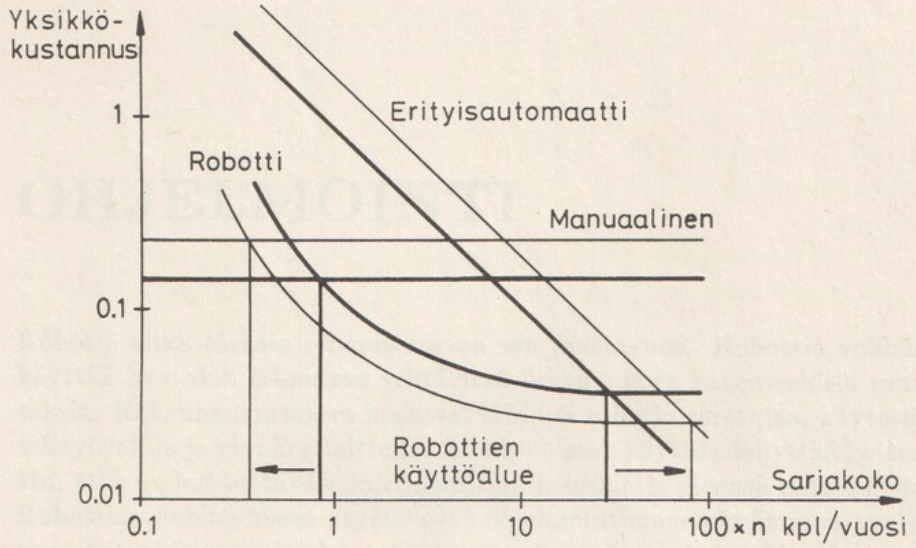
tasainen laatu

Robotisoinnilla saavutettava tuotannon kasvu aiheutuu suurimmaksi osaksi robotin kyvystä työskennellä "vuorotta". Varsinaista työn suoritusta robotilla voidaan harvoin nopeuttaa paljon. Tuotannon kasvun lisäksi robotilla valmistettujen tuotteiden laatu pysyy tasaisena, mitä ihmistyöllä on yleensä vaikea saavuttaa.

Tällä hetkellä robottien pääasialliset käyttöalueet ovat hitsaus, maalaus ja lakkaus, esineiden siirtely, lasinleikkuu, hionta, hiekkapuhallus, valaminen ja kokoonpano. Taulukossa 1 esitetään EC:ssä tehty ennuste robottien työtehtävien muuttumisesta vuoteen 1990 mennessä. Tästä havaitaan perinteisten sovellutusalueiden osuuden vähenevän ja erityisesti kokoonpanon kasvattavan osuuttaan.

tehtävä	1981	1990
Pistehitsaus	40%	3%
Koneiden panostus	29%	32%
Kokoonpano	6%	35%
Kaarihitsaus	6%	16%
Pintakäsittely	11%	4%
Muut	11%	17%

Taulukko 1. Robottien tehtävien muuttuminen [Paa85].



Kuva 1.4: Robottien edullisin käyttöalue automaatiossa erilaisilla tuotantomäärillä nyt ja tulevaisuudessa (nuolet).

Suomen Robotiikkayhdistyksen mukaan Suomessa oli vuoden 1986 lopulla 335 robottia, joista suurinta osaa käytettiin hitsauksessa [Teo87]. Taulukkoon 2 on koottu eri sovellutusalueiden prosentuaalinen osuus.

tehtävä	*osuus*
Hitsaus	46%
Kappaleen käsittely	25%
Pinnoittaminen	9%
Hionta	7%
Kokoonpano	4%
Tutkimus, koulutus ja kehitystyö	9%

Taulukko 2. Robottisovellutukset Suomessa 1986 [Teo87].

OHJELMOINTI

Robotin ehkä tärkein ominaisuus on sen joustavuus. Robottia voidaan käyttää hyvinkin erilaisissa tehtävissä ilman suuria rakenteellisia muutoksia. Rakennemuutokset koskevat lähinnä robotin tarttujaa, käytettäviä työkaluja ja ympäryslaitteita. Monipuolinen käyttö edellyttää kuitenkin, että uuden tehtävän ohjelmointi on helppoa ja yleensä mahdollista. Robottien kehittymisen myötä ovat ohjelmointimenetelmät monipuolistuneet ja ohjelmointikielet parantuneet. Samalla on kuitenkin yhteensopimattomien kielten lukumäärä kasvanut ja standardoinnin tarve tullut ilmeiseksi.

joustavuus

Robottien ohjelmointimenetelmät voidaan jakaa kahteen ryhmään: suoraan (engl. on-line) ja erillään (engl. off-line) ohjelmointiin. Suorassa menetelmässä robotti on kiinteästi kytkettynä ohjelmointijärjestelmään, joten se on irrotettava tuotannosta ohjelmoinnin ajaksi. Tehtävät ohjelmoidaan tavallisesti ohjausyksikköön sijoitetulla näppäimistöllä tai käsiohjaimella. Erillinen ohjelmointi tapahtuu erillään robotista, yleensä ohjelmointijärjestelmässä käyttäen jotakin kehittynyttä ohjelmointikieltä. Joskus käytetään myös ohjelmointirobottia.

on-line
off-line

Robottien ohjelmointikielien kehitys on nopeaa. Olemassa olevista kielistä ilmestyy uusia, entistä kehittyneempiä versioita lähes vuosittain. Lisäksi useat tutkimuslaitokset julkistavat jatkuvasti uusia ohjelmointijärjestelmiä ja -kieliä. Suurin osa näistä on kuitenkin tarkoitettu vain tutkimuskäyttöön.

lukuisia kieliä

2.1 Ohjelmointimenetelmät

Robottien ohjelmoinnissa käytetään monia menetelmiä. Erityisesti vanhemmissa roboteissa ohjelmointimenetelmä riippui täysin robotin tyypistä ja käyttötarkoituksesta. Nykyiset menetelmät voidaan jakaa neljään päätyyppiin seuraavasti:

menetelmät

- mekaanis-rakenteellinen menetelmä,
- johdattamismenetelmä,

- opetusmenetelmä ja
- tekstuaalinen ohjelmointi.

mekaaniset rajoittimet

Mekaanis-rakenteellinen menetelmä on yksinkertaisin, tuskin ohjelmoinniksi kutsuttava. Robotti ohjelmoidaan määräämällä vapausasteiden toimintajärjestys ja siirtämällä käsin liikkeiden mekaanisia rajoittimia. Menetelmää käytetään yksinkertaisten, lähinnä pneumaattisten siirtimien ohjelmoinnissa. Vapausasteita tämän tyyppisissä laitteissa on tavallisesti yhdestä kolmeen, joten tiukasti tulkiten ne eivät ole robotteja ollenkaan.

johdattaminen

Johdattamismenetelmä on varsinaisista ohjelmointimenetelmistä vanhin ja laajimmalle levinnyt. Ohjelmointi tapahtuu kuljettamalla robottia käsin haluttujen pisteiden kautta. Tarvittavissa pisteissä talletetaan nivelien asennot muistiin. Lisäksi talletetaan tarttujan sulkemis- ja avaamiskomennot sekä työkalujen toimintaan liittyvät komennot. Tuloksena saatava ohjelma on siten sarja nivelkoordinaatteja ja tarttujan sekä työkalujen ohjaussignaaleja. Ohjelmaa toteutettaessa robotti liikkuu muistiin talletettujen pisteiden kautta talletusjärjestyksessä toimittaen kussakin pisteessä annetut tehtävät.

liikerata

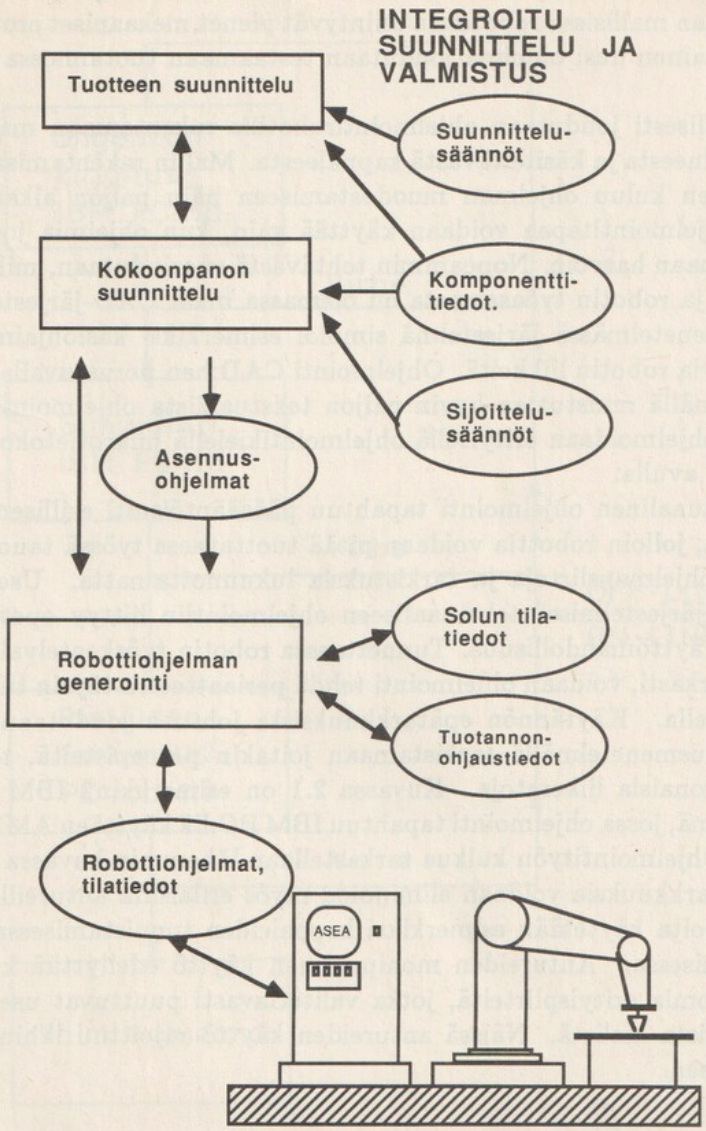
Erillisten pisteiden sijasta on eräissä roboteissa mahdollista tallettaa jatkuva liikerata. Tämä on edellistä huomattavasti tarkempi menetelmä ja sitä käytetään esimerkiksi maalausroboteissa, joissa liikeradan muodolla, toistotarkkuudella ja -nopeudella on ratkaiseva merkitys. Johdattamismenetelmä on erittäin helppo käyttää ja toteuttaa. Koska toteutuksessa ei tarvita monipuolista ja kallista tietokonetta, oli se aikaisemmin hyvin suosittu teollisuudessa. Menetelmää käytetään edelleenkin maalaus- ja hitsausroboteissa sekä yksinkertaisissa materiaalinkäsittelyroboteissa, mutta opetusmenetelmä ja tekstuaalinen ohjelmointi ovat syrjäyttämässä sen useimmissa sovellutuksissa.

rajoitukset

Johdattamismenetelmällä on eräitä merkittäviä rajoituksia. Tärkein on toistorakenteen puuttuminen. Esimerkiksi ohjelmoitaessa robottia noutamaan esineitä paletilta, on jokainen nouto ohjelmoitava erikseen, mikä on hyvin työlästä. Lisäksi ohjelmointivirheen korjaaminen on vaikeaa. Usein joudutaan ohjelmoimaan koko tehtävä uudelleen. Toinen menetelmän käyttöä ankarasti rajoittava tekijä on se, että antureita ei voida käyttää.

opetusmenetelmä

Johdattamismenetelmän haittoja on pyritty poistamaan opetusmenetelmällä ohjelmoitavissa roboteissa. Ohjelmointi tapahtuu erillisellä ohjelmointilaitteella, käsiohjaimella. Halutut pisteet etsitään liikuttamalla robottia käsiohjaimessa olevilla näppäimillä tai ohjaussauvalla. Opetusmenetelmässä käytettävät ohjelmointikielet ovat jo pidemmälle kehittyneitä mahdollistaen monimutkaisempien liikeratojen ohjelmoinnin sekä mm. antureiden käytön. Haittana on suora ohjelmointi, eli ohjelmoinnissa joudutaan käyttämään robottia apuna, jolloin se on irrotettava tuottavasta työstä. Tämä voidaan periaatteessa välttää erillisellä ohjelmointirobotilla, jonka avulla laaditut ohjelmat voidaan nopeasti siir-



Kuva 2.1: Esimerkki robotin ohjelmointijärjestelmästä (Oy Sähkövaruste Ab).

tää tuotantoon. Ohjelmointirobotin käyttöä haittaa ympäristössä ja jopa saman mallisissa roboteissa esiintyvät pienet mekaaniset erot, joiden takia jokainen uusi ohjelma joudutaan testaamaan tuotannossa olevalla robotilla.

malli

Tavallisesti joudutaan ohjelmointirobotille rakentamaan malli työskentelyalueesta ja käsiteltävästä kappaleesta. Mallin rakentamisaika mukaanlukien kuluu ohjelman muodostamiseen näin paljon aikaa. Tällaista ohjelmointitapaa voidaan käyttää vain, kun ohjelmia joudutaan muuttamaan harvoin. Nopeammin tehtävästä suoriudutaan, mikäli kappaleesta ja robotin työasemasta on olemassa malli CAD-järjestelmässä. Tässä menetelmässä järjestelmä simuloi esimerkiksi käsiohjaimella toteutettavia robotin liikkeitä. Ohjelmointi CAD:hen perustuvalla opetusmenetelmällä muistuttaa hyvin paljon tekstuaalista ohjelmointia, jossa robotti ohjelmoidaan erityisellä ohjelmointikielellä mikrotietokoneen tai päätteen avulla.

CAD

tekstuaalinen ohjelmointi

Tekstuaalinen ohjelmointi tapahtuu pääsääntöisesti erillisenä ohjelmointina, jolloin robottia voidaan pitää tuottavassa työssä tauotta vain lyhyitä ohjelmansiirtoja ja tarkistuksia lukuunottamatta. Useissa ohjelmointijärjestelmissä tekstuaaliseen ohjelmointiin liittyy opetusmenetelmän käyttömahdollisuus. Tunnettaessa robotin työskentelyalueen rakenne tarkasti, voidaan ohjelmointi tehdä periaatteessa täysin tuotannon ulkopuolella. Käytännön epätarkkuuksista johtuen joudutaan kuitenkin opetusmenetelmällä tarkistamaan joitakin peruspisteitä, toisinaan jopa kokonaisia liikeratoja. Kuvassa 2.1 on esimerkkinä IBM robottijärjestelmä, jossa ohjelmointi tapahtuu IBM PC:llä käyttäen AML/Entry-kieltä. Ohjelmointityön kulkua tarkastellaan lähemmin kuvassa 2.2 .

anturit

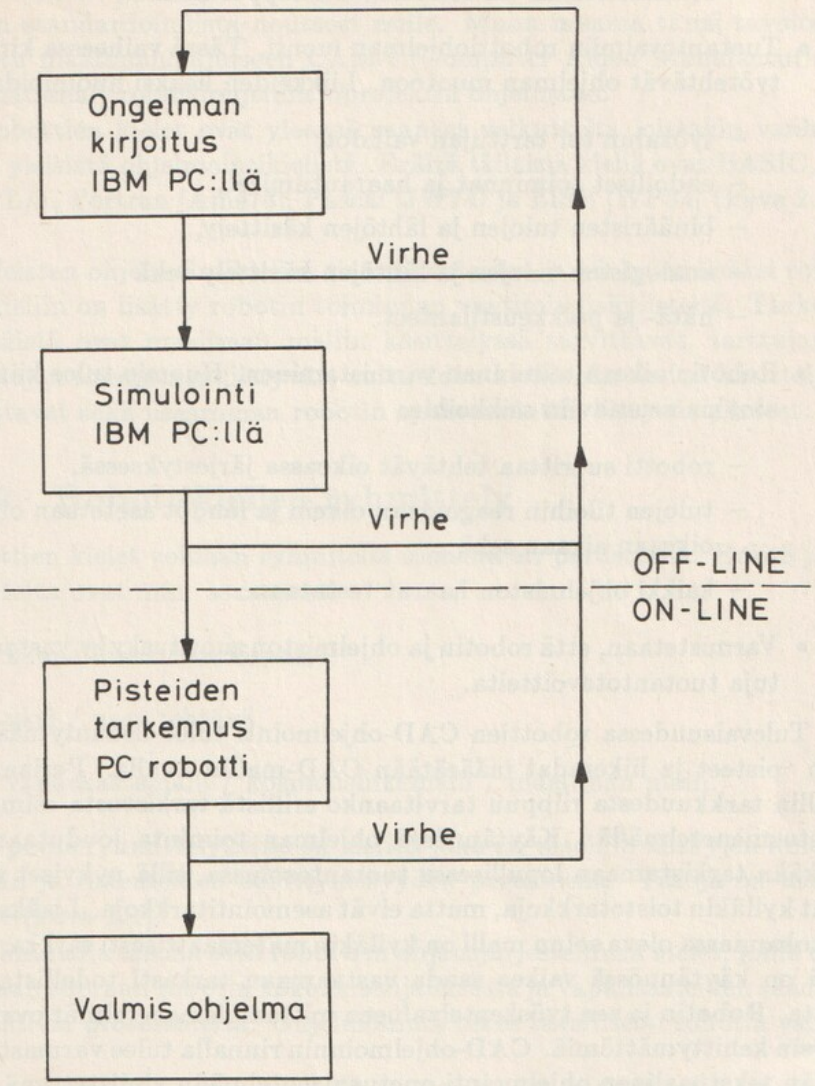
Epätarkkuuksia voidaan eliminoida myös erilaisilla antureilla ja aistimilla, joita käytetään esimerkiksi kappaleiden tunnistamisessa ja paikallistamisessa. Antureiden monipuolinen käyttö edellyttää kuitenkin kieleltä omia erityispiirteitä, jotka valitettavasti puuttuvat useimmista kaupallisista kielistä. Näissä antureiden käyttö rajoittuu lähinnä tilan testaukseen.

2.1.1 Ohjelman suunnittelu

Robotin toiminnan yksityiskohtainen suunnittelu ja ohjelmointi aloitetaan, kun robotin tehtävä ja työskentely-ympäristö on alustavasti päätetty. Ohjelmointi voidaan jakaa seuraaviin vaiheisiin [Kan84].

- Työtehtävien ja liikeratojen muodostus siten, että
  - kaikki tarvittavat pisteet saavutetaan vaadituissa orientaatioissa,
  - yksittäiset liikeradat eivät sisällä konfiguraation muutoksia,
  - liikkeiden välipisteet (esim. suoraviivainen liike) eivät sisällä saavuttamattomia pisteitä[1] ja

[1] robotin työskentelyalue on usein varsin monimutkainen ja ei-konveksi



Kuva 2.2: Tyypillinen työnkulku erillään ohjelmoinnissa.

  - robotti liikkuu vapaalla liikealueella eikä törmää esimerkiksi oheislaitteisiin tai käsiteltäviin kappaleisiin.

- Tuotantovalmiin robottiohjelman luonti. Tässä vaiheessa kirjataan työtehtävät ohjelman muotoon. Liikkeiden lisäksi huomioidaan
  - työkalun tai tarttujan vaihdot,
  - ehdolliset toiminnat ja haarautumiset,
  - binääristen tulojen ja lähtöjen käsittely,
  - analogisten tulojen ja lähtöjen käsittely sekä
  - hätä- ja poikkeustilanteet.

- Robotin oikean toiminnan varmistaminen. Huomio tulee kiinnittää ainakin seuraaviin seikkoihin:
  - robotti suorittaa tehtävät oikeassa järjestyksessä,
  - tulojen tiloihin reagoidaan oikein ja lähdöt asetetaan oikein ja oikeaan aikaan sekä
  - kaikki ohjelmiston haarat testataan.

- Varmistetaan, että robotin ja ohjelmiston suorituskyky vastaa asetettuja tuotantotavoitteita.

CAD-malli

Tulevaisuudessa robottien CAD-ohjelmointi tulee lisääntymään, jolloin pisteet ja liikeradat määrätään CAD-mallin avulla. Periaatteessa mallin tarkkuudesta riippuu tarvitaanko erillistä tarkistusta esimerkiksi opetusmenetelmällä. Käytännössä ohjelman toiminta joudutaan vielä pitkään tarkistamaan lopullisessa tuotantosolussa, sillä nykyiset robotit ovat kylläkin toistotarkkoja, mutta eivät asemointitarkkoja. Lisäksi CAD-tietokannassa oleva solun malli on kylläkin matemaattisesti tarkka, mutta sitä on käytännössä vaikea saada vastaamaan tarkasti todellista tilannetta. Robotin ja sen työskentelyalueen mittaamismenetelmät ovat myös täysin kehittymättömiä. CAD-ohjelmoinnin rinnalla tulee varmasti säilymään tekstuaalinen ohjelmointi opetusmenetelmään yhdistettynä.

2.2 Robottien ohjelmointikielet

Tässä kappaleessa on tarkoitus tutustua robottien ohjelmointikielien yleisimpiin piirteisiin ja yhteyksiin muihin, lähinnä yleisiin ja NC-ohjelmointikieliin.

2.2.1 Yhteys yleisiin kieliin

Robottijärjestelmien kehittyessä ovat myös niiden ohjelmointimenetelmät ja -kielet kehittyneet melkoisesti. Yksinkertaisista NC-ohjelmointia muistuttavista kielistä on siirrytty monipuolisiin strukturoituihinkieliin. Robottikielien kehittely jatkuu edelleen, mutta jo nyt niiden lukumäärä

alkaa olla liian suuri. Käytännössä lähes jokainen suurempi robottien valmistaja on pyrkinyt omaan kieleen. Viime aikoina ovat vaatimukset kielien standardoinnista nousseet esille. Muun muassa tämä tavoite on kirjattu maailmanlaajuiseen CAM-I (Computer Aided Manufacturing - International) robottiohjelmistoprojektin ohjelmaan.

yleiskielet

Robottien kielet ovat yleensä saaneet vaikutteita joistakin vanhemmista yleisistä ohjelmointikielistä. Eräitä tällaisia kieliä ovat BASIC, Algol, PL/1, Fortran [Ame78], Pascal [JW74] ja LISP [WP84] (kuva 2.3).

lisäpiirteet

Yleisten ohjelmointikielien tietorakenteiden ja käskyjen lisäksi robottien kieliin on lisätty robotin toiminnan vaatimia lisäpiirteitä. Tärkeimpiä näistä ovat maailman mallin käsittelyssä tarvittavat, tarttujan ja työkalujen toimintaan liittyvät, antureiden keräämän tiedon käsittelyssä tarvittavat sekä useamman robotin synkronointiin liittyvät piirteet.

2.2.2 Robottikielien ryhmittely

Robottien kielet voidaan ryhmitellä monella eri perusteella. Eräitä jakoperusteita ovat mm. seuraavat:

- kaupallinen / tutkimus,
- yksi / usea tehtävä,
- vapausastemalli / kokonaisliikemalli / maailman malli.

Ehkä paras ryhmittelykeino on kielien jako eri tasoihin käskyjen ilmaisuvoiman ja rakenteiden kehittyneisyyden perusteella. Tasoja on tavallisesti 5 (kuva 2.4).

vapausasteet

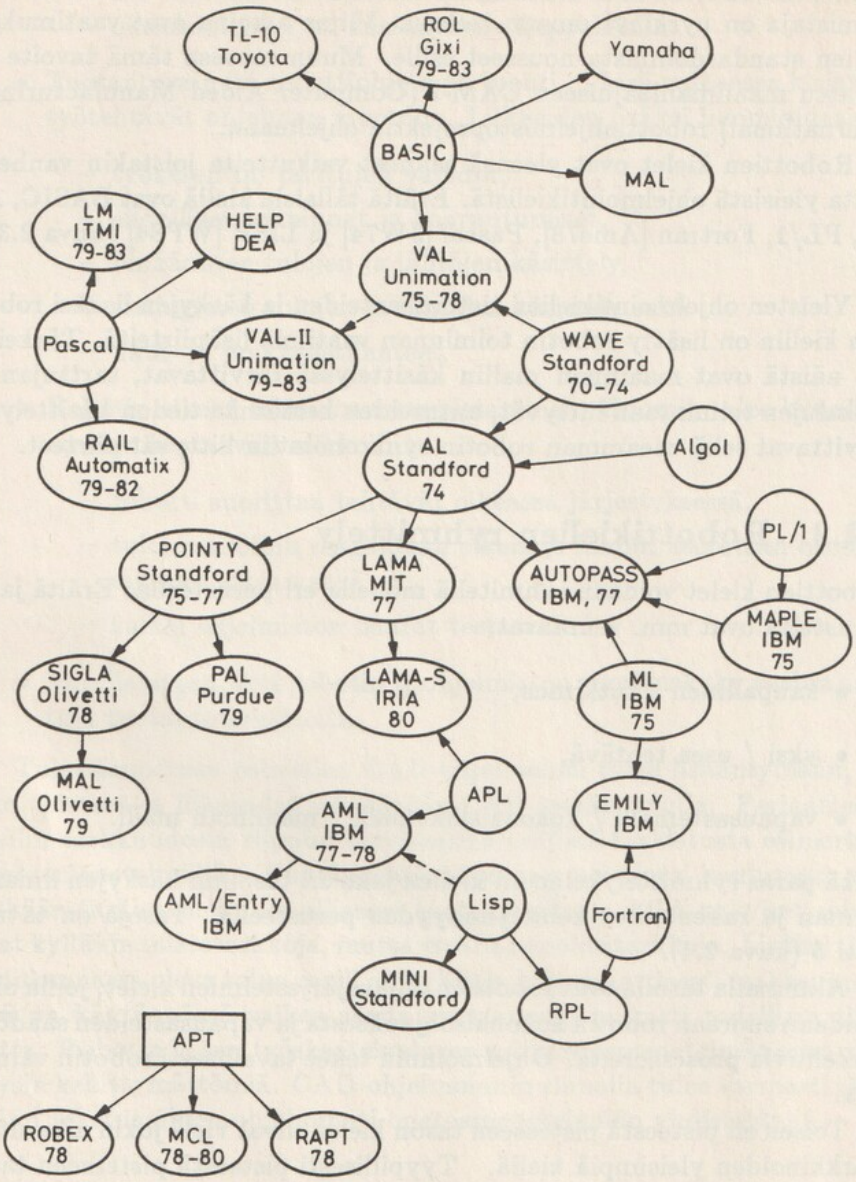
Alimmalla tasolla ovat robottien ohjausjärjestelmien kielet, joilla ohjelmoidaan suoraan robotin kokonaisohjauksesta ja vapausasteiden säädöstä huolehtivia prosessoreita. Ohjelmoinnin tekee tavallisesti robotin valmistaja.

pisteestä pisteeseen

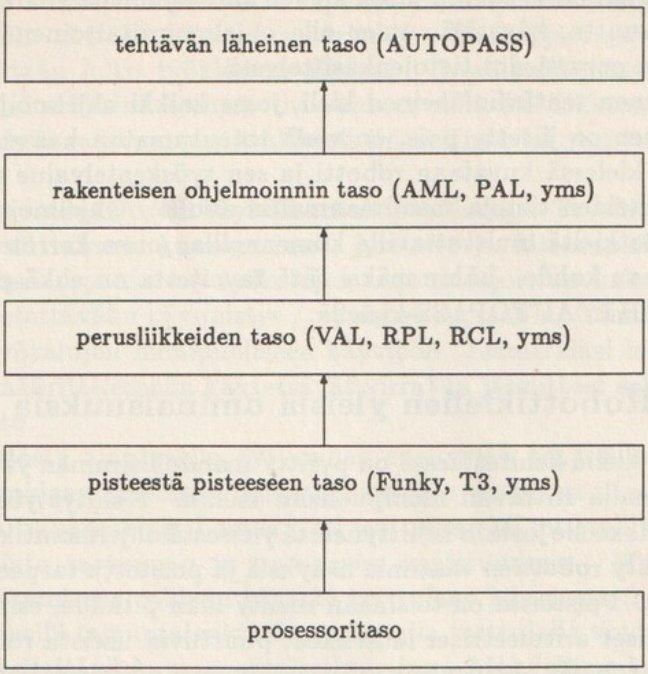
Toisen eli pisteestä pisteeseen tason kielet olivat vielä jokin aika sitten markkinoiden yleisimpiä kieliä. Tyypillisesti pisteestä pisteeseen tason kielen käskyt liittyvät enemmän robotin liikkeisiin kuin suoritettavaan tehtävään. Kielissä on mahdollisesti alkeelliset aliohjelma- ja hyppymahdollisuudet sekä yksinkertaiset ympäristöliitännät. Tason 2 kielillä ohjelmointi on usein yhdistelmä tekstuaalista ohjelmointia ja opetusmenetelmän käyttöä. Ohjelmointi on lähes yksinomaan suoraa ohjelmointia. Esimerkkinä tason 2 kielistä ovat T3 ja FUNKY.

perusliikkeet

Tällä hetkellä markkinoilla olevat robottien kielet ovat pääasiassa tason 3 eli perusliikkeiden tason kieliä. Myös nämä toimivat robottien liikkeiden tasolla, mutta komennot ovat nyt selväkielisempiä. Tason 3 kielissä on paremmat aliohjelmarakenteet, myös hyppyjä ja silmukoita voidaan käyttää. Ympäristöliitännät ovat paremmat mahdollistaen mm. näköjärjestelmän käytön. Tämän tason kielistä mainittakoon RCL, PAL,



Kuva 2.3: Eräitä robottien kieliä ja niiden yhteydet yleisiin ohjelmointikieliin. Erillisenä ryhmänä ovat NC-koneiden ohjelmoinnissa käytettävästä APT-kielestä kehitetyt kielet.



Kuva 2.4: Ohjelmointikielien jako viiteen tasoon.

Sigla, Anorad, Emily ja VAL. Näistä viimeksi mainittu on levinnyt laajalle PUMA-robottien kielenä.

Selvästi edellistä tasoa kehittyneempiä ovat rakenteisen tason kielet. *rakenteet*
Nämä sisältävät monia kontrollirakenteita, mahdollistavat koordinaatistomuunnosten ja suhteellisten koordinaatistojen käytön. Antureiden yhteydessä käytettävät käskyt ovat yleensä samanlaisia kuin tason 3 kielissä, mutta erilaisten antureiden ja aistimien käyttömahdollisuudet vaihtelevat melkoisesti. Esimerkiksi PAL-kielessä ei ole antureiden käyttömahdollisuutta lainkaan, AL- ja HELP-kielissä voidaan käyttää kosketusantureita tarttujissa jne. Useissa tämän tason kielissä on mahdollista usealla robotilla suoritettava rinnakkainen prosessointi. Tämä tehdään tavallisesti ohjelmien keskinäisellä tahdistuksella. Kielien liikekomennot ilmaisevat yleisimmin robotin tarttujan liikkeitä. Edellä esitettyjen lisäksi rakenteisen tason kieliä ovat muun muassa AML, MCL ja MAPLE.

Huolimatta lukuisista etuisuuksista alemman tason kieliin verrattuna *ongelmia*
on rakenteisen tason kielillä vielä eräitä heikkouksia. Esimerkiksi robottien törmäysten esto rinnakkaisessa prosessoinnissa on vielä ratkaisematta. Eräissä tapauksissa koordinaatistomuunnosten yksinkertaistaminen parantaisi kielien käyttöä. Lisäksi olisi tutkittava, onko tämäntasoisiin kieliin mahdollista liittää robotin päättelykykyä lisääviä piirteitä.

Yleisesti tarkasteltuna ohjelmointi rakenteisen tason kielillä lisää robottien joustavuutta, erityisesti ohjelmien rakenteellisuus helpottaa ohjelmoijan työtä. Ohjelmointi tapahtuu yleensä erillään ohjelmointina. Oh-

jelmoinnin kannalta tämän tason kielien haittapuolena voidaan pitää niiden vaativuutta, käyttäjän tulee olla ohjelmointitaitoinen ja hänen on tunnettava perustiedot tietojenkäsittelystä.

tulevaisuus

Todellinen tehtävänläheinen kieli, jossa kaikki ohjelmoijan kannalta epäoleellinen on jätetty pois, on vielä toteutumaton haave. Tehtävänläheisessä kielessä kuvataan robotti ja sen työskentelyalue sekä käsiteltävien kohteiden tietoja maailmanmallin avulla. Ohjelmointi tapahtuu luonnollista kieltä muistuttavilla komennoilla, joissa kerrotaan toiminta ja toiminnan kohde. Lähimmäksi tätä tavoitetta on ehkä päässyt IBM kehittämällään AUTOPASS-kielellä.

2.2.3 Robottikielien yleisiä ominaisuuksia

Robottien kieliä kehitettäessä on pyritty mahdollisimman yksinkertaisiin mutta samalla riittävän monipuolisiin kieliin. Kehitystyössä on usein lähdetty liikkeelle jostain kehittyneestä yleisestä ohjelmointikielestä. Kieleen on tehty robottien vaatimia lisäyksiä ja poistettu tarpeettomia ominaisuuksia. Poistoissa on toisinaan menty liian pitkälle, esimerkiksi tuiki tarpeelliset aritmeettiset lausekkeet puuttuvat useista robottikielistä. Myöskään robotin ohjausyksikön ja järjestelmään liitetyn tietokoneen väliseen kommunikointiin ei ole yleensä kiinnitetty tarpeeksi huomiota.

tietotyypit

Luonnollisesti robottien kielien ominaisuudet vaihtelevat melkoisesti. Usein on mahdollista käyttää vain tietyntyyppistä tietoa, tavallisesti joko kokonaislukuja tai reaalilukuja. Parhaimmissa kielissä tietotyyppejä on kuitenkin useita. Esimerkiksi AL-kielessä voidaan käyttää reaalilukuja, vektoreita, kiertokulmia annettujen vektoreiden suhteen, eri kohteisiin kiinnitettyjä koordinaatistoja, koordinaatistojen välisiä suhteita, rinnakkaisen toiminnan tahdistukseen tarvittavia tietoja sekä taulukkoja. Myös koordinaatistojen käyttömahdollisuudet vaihtelevat. Yksinkertaisimmissa kielissä voidaan käyttää vain robottiin kiinnitettyä, sen rakenteesta riippuvaa koordinaatistoa. Tavallisesti voidaan määritellä muitakin koordinaatistoja. Esimerkiksi komponenttikohtaisilla koordinaatistoilla on mahdollista ohjelmoida vaikkapa kokoonpanotehtävä.

aritmetiikka

Aritmeettisten ja loogisten lausekkeiden puuttuminen hankaloittaa eräiden kaupallisten robottikielien käyttöä. Erityisesti erilläänohjelmoinnissa lausekkeiden käyttö on kuitenkin käytännössä lähes välttämätöntä. Tästä syystä on hyvin todennäköistä, että puute tullaan lähiaikoina poistamaan useimmista kielistä .

kontrolli-rakenteet

Luonnollisesti myös kontrollirakenteet vaihtelevat. Ohjelmointia helpottavia kontrollirakenteita ovat tavallisesti `IF ... THEN ... ELSE ... -`, `WHILE ... DO ... -`, `GOTO-`, `CALL-` ja `FOR ... TO ... DO-`, rakenteet. Myös erilaiset ehdolliset ja ehdottomat haarautumiset ovat mahdollisia. Näistä esimerkkinä mainittakoon AML/Entryn [IBM84a] `BRANCH(osoite)-` ja `TESTC(nimi,arvo,osoite)` -lauseet.

liikkeet

Robotin liikkeiden hallitsemisessa tarvittavat piirteet ovat oleellinen osa kieltä. Liikkeet voidaan toteuttaa pisteohjatusti, jolloin robotti kul-

kee lähtö- ja kohdepisteen välillä vapaasti rakenteilleen ominaisella tavalla. Tällöin riittää pisteiden määrittely. Rataohjattujen robottien reitti määrätään koko työkierron aikana. Tavallisesti liikekomentojen yhteydessä tai erillisinä komentoina voidaan määrätä liikkeen nopeus ja tarkkuus. Myös ratatyyppi, tavallisesti lineaarinen tai ympyräliike, voidaan antaa.

työkalut

Robotilla voidaan usein käyttää erilaisia työkaluja. Tavallisimpia ovat ruuvaimet, niittaimet, kuumentimet, jyrsimet ja hitsauspäät. Yksinkertaisimmissa tapauksissa työkalujen ohjaus rajoittuu kaksitasoisesta portista lähetettävään käynnistys / pysäytys signaaliin. Tämä ei kuitenkaan riitä työkalujen monipuoliseen käyttöön. Esimerkiksi hitsauksessa joudutaan määrittelemään käytettävät virrat ja jännitteet sekä hitsauslangan syöttö.

anturit

Ympäristöstä ajantasalla pysyminen edellyttää antureiden käyttöä. Antureilla voidaan tarkkailla kuljettimien ja syöttimien tilaa sekä seurata robotin työskentelyä. Esimerkiksi tarttujaan kiinnitetyillä antureilla voidaan todeta tartunnan ja irrotuksen onnistuminen. Myös kohteiden paikallistamiseen voidaan käyttää tarttujaan kiinnitettyä optista anturia. Erityisellä tartuntaleveyttä mittaavalla tarttujalla voidaan tarkistaa onko käsiteltävä komponentti oikea, jne. Mahdollisuuksia on monia. Oman erityisen ryhmän muodostavat näköaistin käyttöön soveltuvat ohjelmointikielet (RPL, MCL, RSS jne). Tyypillisesti kielessä on kohteen määrittelyyn, tunnistamiseen, paikallistamiseen ja seurantaan liittyviä ominaisuuksia.

liitännät

Kaksitasoiset anturit voidaan yhdistää robotin DI-portteihin. Tällöin antureiden tuottamaa tietoa tarkastellaan ulkoisten liityntöjen käsittelyyn tarkoitetuilla välineillä. Tavallisesti testataan tai muutetaan määrätyn portin tilaa sekä ryhdytään tarvittaviin toimenpiteisiin. Kaksitasoisten porttien lisäksi eräissä roboteissa on analogisia portteja.

2.2.4 Robottikielien heikkouksia

kaupalliset kielet

Vuonna 1983 oli markkinoilla seuraavat robottien ohjelmointikielet: AL, AML, HELP, MCL, RAIL ja VAL. Tämän jälkeen on markkinoille tullut mm. ASEAn ARLA-kieli sekä joukko japanilaisia kieliä. Jälkimmäisistä mainittakoon Hitachin ARL, STROL, Yamaha ja Toyotan TL-10.

Kielien tarkempi esittely on kohdassa "Robottikielien kuvaus". Seuraavassa tarkastellaan hieman kaupallisissa kielissä olevia puutteita. Käsittely perustuu pääasiassa artikkeliin [Sor83]. [2]

kommunikointi

Tuotannon joustavuus edellyttää hyvää ohjausjärjestelmää, mikä puolestaan edellyttää toimivaa kommunikointia laitteiden välillä. Vielä nyt useimmista roboteista puuttuu täysin toimiva kommunikointijärjestelmä. Kommunikointi rajoittuu pääasiassa ohjelmien lataamiseen ja DI/DO-porttien käyttöön. Kehitystä parempaan on havaittavissa, mutta edistyminen on kuitenkin toistaiseksi ollut hidasta. Tarvittavia ominaisuuk-

[2] Tarkastelujärjestys ei kuvaa kyseisen ominaisuuden tärkeyttä.

sia ovat muun muassa robotin kyky vastaanottaa tarvittavia tietoja CAD-järjestelmältä ohjelman suorituksen aikana[3] sekä kyky tiedottaa ulospäin solun tapahtumista.

AML/Entry

Tällä hetkellä robottien kommunikointikyky on lasten kengissä. Esimerkkinä mainittakoon AML/Entry [IBM84b], jonka versio 4.1 on kommunikointikyvyltään eräs kehittyneimmistä kielistä. Siinä on kaksi komentoa, joiden avulla voidaan välittää tietoa[4] PC:ltä (GETC) tai PC:lle (PUTC). Komentoja käytettäessä on siirrettävät luvut talletettava etukäteen kommunikoinnista huolehtivaan ohjelmaan. Ohjelma valvoo robotin työskentelyä ja lähettää luvun kerrallaan robotille tämän niitä pyytäessä. Robotilta saatavia lukuja voidaan käyttää esimerkiksi tilastoinnissa ja toiminnan ohjauksessa. Komentojen käyttö edellyttää tavallisesti vain ohjelmoinnissa käytettävän PC:n jatkuvaa kytkentää robotin ohjausyksikköön.

Robottien kommunikointi solun muiden työkalujen kanssa on myös puutteellista. Esimerkiksi oheislaite käynnistetään robotin ohjelmassa tällä hetkellä kahdessa vaiheessa:

1. lähetä pulssi, joka käynnistää laitteen,
2. tarkista, että laite käy.

STROKE

Tavallisesti tähän tarvitaan vähintään kaksi ellei useampiakin komentoja. Erilaisten koneiden käyttö on robotille kuitenkin tavallista, joten olisi edullista pystyä hoitamaan esimerkiksi koneen käynnistys yhdellä komennolla HELP-kielen STROKE-lauseen tapaan:

```
STROKE(feeder,sensor,kicker,tries).
```

Lauseella lähetetään pulssi linjalle **feeder**, jonka jälkeen tarkistetaan linjalta **sensor** onko kyseinen komponentti paikoillaan syöttölaitteessa. Mikäli komponenttia ei ole, lähetetään pulssi linjalle **kicker**. Tämän vaikutuksesta pyrkii laitteisto irrottamaan kiinni jääneen komponentin. Yritysten lukumäärän ilmoittaa parametri **tries**.

MAP

Parannusta robotin, ympäryslaitteiden ja ohjausjärjestelmän väliseen kommunikointiin on kuitenkin luvassa. Kehitteillä on uusi paikallisverkko säännöstö MAP (Manufacturing Automation Protocol), jolla voidaan liittää yhteen erityyppisiä tuotantolaitteita samaan tietokoneverkkoon. Tämä edellyttää tosin joitakin muutoksia robottien kielissä.

[3] esimerkiksi työskentelypisteitä
[4] kokonais- ja reaalilukuja

kuljetin

Myös kuljettimet ovat hankalia robotteja ohjelmoitaessa. Tällä hetkellä kyetään seuraamaan vain suoraviivaista, robotin koordinaatistossa x- tai y-koordinaatin suuntaista kuljetinta. Kaarevien ratojen seuraaminen edellyttäisi robottien kieliltä nykyistä parempia laskennallisia ominaisuuksia. Jo aritmeettisten lausekkeiden käyttömahdollisuus parantaisi tilannetta usein huomattavasti.

paletti

Palettien käyttö komponenttien syötössä oli eräs tärkeimmistä syistä robottikielien synnylle. On nimittäin erittäin pitkästyttävää opettaa käsin esimerkiksi kaikki 10×10 paletin sata pistettä. Mm. AML/Entry-kielessä on monta palettien käsittelyyn tarkoitettua komentoa. Esimerkiksi paletin määrittelyssä riittää 3 nurkkapistettä sekä komponenttien lukumäärä yhteensä että lukumäärä rivillä. Komponenttien indeksointi paletilla voisi tosin olla helpompaa. Nykyiset robottien ohjelmointikielet ymmärtävät vain 2-D paletteja.

robotti-riippumattomuus

Tarkasteltavilla kielillä laaditut ohjelmat ovat sidottuja tiettyyn robottiin. Ohjelmillahan kuvataan robotin liikkeitä eikä varsinaista tehtävää. Tästä syystä kaikki muutokset työasemalla aiheuttavat muutoksia myös robotin ohjelmassa. Esimerkiksi jos kahden robotin työasemalta toinen roboteista vaurioituu, pysähtyy tavallisesti myös toisen robotin työskentely, vaikka se ulottuisikin käsiteltävään kappaleeseen. Tämä lienee korjattavissa ylemmän tason ohjauksella.

Myös tehtävien siirtäminen robotilta toiselle edellyttää, että siirrettävä tehtävä on etukäteen ohjelmoitu kyseiselle robotille. Ohjelmien robottisidonnaisuuden takia ei ohjelman siirto robotilta toiselle yleensä onnistu. Jopa samaa mallia olevissa roboteissa voi olla niin suuria eroja, että siirrettävissä ohjelmissa olevat pisteet on määritettävä uudelleen, vaikka työympäristöt muuten olisivatkin identtiset.

prioriteetti

Robotin tehtävien paneminen tärkeysjärjestykseen on myös vaikeata. Usein koneita palvelevalla robotilla olisi aikaa tehdä jotain muutakin, mutta näiden "toisarvoisten" tehtävien määrittely on nykyisillä ohjauskielillä hankalaa. Tilanteen parantamiseksi tarvitaan kunnolliset prioriteettimäärittelyt sekä keskeytysten käsittelyt. Rinnakkainen prosessointi on jossakin määrin mahdollista AL- ja VAL-II-kielissä, mutta täydellinen rinnakkainen prosessointi on vielä mahdotonta.

törmäykset

Oman hankalan ryhmänsä muodostavat robotin työskentelyalueella sijaitsevat esteet. Niiden esittäminen nykyisissä ohjauskielissä on mahdotonta. Mikäli esteet voitaisiin esittää joidenkin yksinkertaisten primitiivien avulla, olisi esteettömien liikeratojen määrääminen suhteellisen helppoa.

tasot

Nykyaikaisen robotin ohjausjärjestelmän sisältämistä alajärjestelmistä mainittakoon jäsentäjä, tulkki, liikeradan muodostaja, liikkeen säätö sekä servojärjestelmä. Liikkeen säätö asettaa työkalun liikeradan tietojen perusteella nivelien liikkeet. Varsinainen liike toteutetaan servojärjestelmällä. Edellä kuvatut toiminnot on tavallisesti hajautettu usealle prosessorille, joiden ohjelmointi tapahtuu konekielillä tai vastaavan tasoisilla kielillä. Näin saavutetaan riittävä nopeus, mutta samalla mene-

tetään siirrettävyys. Tulevaisuudessa olisikin kehitettävä korkeatasoinen kieli, jolla voitaisiin ohjelmoida kaikkia robottijärjestelmän tasoja.

laajennukset

Haluttaessa laajentaa nykyisiä robottijärjestelmiä uusilla prosessoreilla, joudutaan tekemään paljon työtä. Mikään ohjauskieli ei sovi sellaisenaan tämäntyyppisiin laajennuksiin. Uusien prosessoreiden liittäminen järjestelmään edellyttäisi ohjauskieleltä muun muassa kykyä kuvata mahdolliset prosessorit ja liitännät. Ohjauskielellä pitäisi myös pystyä arvioimaan eri prosessoreilta annetun ohjelman suorittamiseen kuluvaa aikaa sekä kyetä hajauttamaan toimintoja käytettävissä oleville prosessoreille. Lisäksi prosessoreiden käytön optimointi tulisi olla mahdollista. Myös tällaisessa toiminnassa esiintyvien virheiden korjaus edellyttää uusia piirteitä ohjauskieliltä.

standardointi

Eräs suurimmista haitoista nykyisissä ohjauskielissä on niiden robottiriippuvuus. Tämä estää muun muassa erilaisten robottien rinnakkaisen työskentelyn. Riippuvuus vaatii myös robotin ostajaa pitäytymään aiemmin hankituissa robottimalleissa, vaikka muualta löytyisikin tehtävään paremmin sopiva robotti. Ohjauskielien standardointi onkin yksi tärkeimmistä ongelmista. Oleellinen osa tulevaisuuden robottiriippumattomassa ohjauskielessä on robottien kuvaus. Kuvauksessa tulee voida vastata muun muassa seuraaviin kysymyksiin: nivelien lukumäärä, tyyppi, ulottuvuudet, todelliset ja nimelliset parametrit sekä käsivarren muoto.

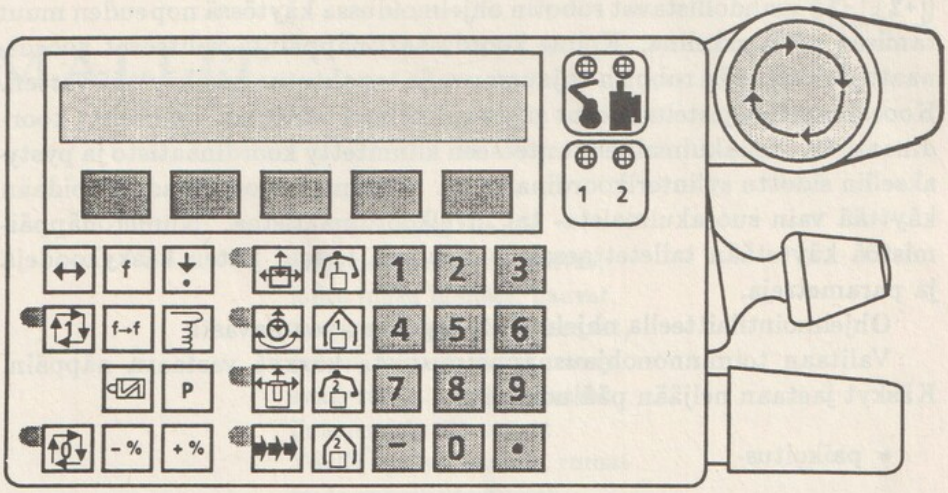
monimutkaisuus

Kuten edellä esitetystä selviää, on nykyisin käytössä olevissa jopa kehittyneissäkin robottikielissä paljon puutteita. Uusia, yhä parempia kieliä kehitettässä on kuitenkin pidettävä mielessä, että kielen monipuolistuessa se samalla monimutkaistuu. Robottien ohjelmointi tapahtuu useimmiten tehdasolosuhteissa ohjelmoijan ohjelmointitaidon ollessa tavallisesti hyvin vähäinen. Tämä edellyttää luonnollista kieltä muistuttavaa, tehtävänläheistä ja yksinkertaista kieltä sekä hyvää ohjelmointiympäristöä.

2.3 Ohjelmointiesimerkki

Tarkastellaan ohjelmointia ASEAn ohjelmointilaitteella, joka on esitetty kuvassa 2.5. Ohjelmointilaitteessa on ohjaimet robotin käsikäyttöä, ohjelman käskyjen sisäännäppäilyä ja tarttujien käyttöä varten.

Ohjausosassa olevalla sauvalla liikutetaan robottia käsiohjauksella mihin tahansa työalueen pisteeseen. Liikuttelun aikana tulee sauvan kanssa yhteenkytketty turvalevy painaa alas. Vipukytkimillä voidaan valita neljä erilaista ohjausvaihtoehtoa. Vasen vipukytkin määrittää, ohjataanko robottia vai robottijärjestelmään kuuluvia ulkoisia akseleita. Oikealla vipukytkimellä määrätään, suoritetaanko liikkeet valitun koordinaattijärjestelmän vai paikallaan pysyvän työkalupisteen (lyh. TKP) mukaan. Ohjausosaan kuuluu lisäksi `hätäpysäytys`-näppäin.



Kuva 2.5: ASEAn ohjelmointilaite.

Ohjelmointilaitteen näppäimet voidaan jakaa neljään ryhmään:

- monitoimi-,
- toiminnonohjaus-,
- numero- ja
- ohjausnäppäimet.

Monitoiminäppäimiä käytetään valittaessa erilaisia toimintoja, joilla annetaan sisään ohjelman käskytyyppejä ja tietoja, editoidaan ohjelmia ja käytetään robottia käsiohjauksella. Kunkin näppäimen toiminto ilmenee näppäimen yläpuolella olevan kaksirivisen näytön alimmalta riviltä.

Toiminnonohjausnäppäimillä puolestaan ohjataan monitoiminäppäinten käyttöä. Toiminnonohjausnäppäimet ovat `automatiikka`, `käsky`, `paikka`, `korjaus` ja `käsin`. `Automatiikka`-näppäimellä voidaan suorittaa seuraavat toiminnot: automaattinen ohjelmanajo, tulojen simulointi ja aikaisemmin ohjelmoitujen pisteiden siirto. `Käsky`-näppäin mahdollistaa minkä tahansa ohjelmakäskyn, paitsi paikoituskäskyn antamisen. Paikoituskäskyjen ja niiden argumenttien antaminen tapahtuu `paikka`-näppäimen ohjaamana. `Korjaus`-näppäimellä asetetaan robotti editointitilaan, jolloin voidaan korjata ohjelmassa ilmenneitä virheitä. `Käsin`-

näppäimellä toteutetaan erinäisiä käsinohjaus- ja tiedonsiirtotoimintoja, kuten ohjelmien lukua, tallennusta jne. Näiden lisäksi on erityisesti hitsaussovelluksia varten olemassa [toiminta]-näppäin, jolla toteutetaan kaarihitsausrobottien ohjelmoinnissa tarvittavia käskyjä. Myös näköjärjestelmän ohjelmointiin on olemassa erityinen [havainto]-näppäin.

Muita ohjausnäppäimiä ovat [seis], [korjaa nopeus], [koordinaatti], [vaihda], [inkrementti] ja [tartuntaelin]. Nopeuden korjaus-näppäimet ([+%], [-%]) mahdollistavat robotin ohjelmoidussa käytössä nopeuden muuttamisen ±5% askelina. Kolme koordinaattinäppäintä valitsevat koordinaattijärjestelmän robotin ohjaussauvalla tapahtuvaa käsikäyttöä varten. Koordinaattijärjestelmät ovat suorakulmainen jalustaan kiinnitetty koordinaatisto, suorakulmainen ranteeseen kiinnitetty koordinaatisto ja pystyakseliin sidottu sylinterikoordinaatisto. Ohjelmaa suoritettaessa voidaan käyttää vain suorakulmaista- tai nivelkoordinaatistoa. Numeronäppäimistöä käytetään talletettaessa numeerista tietoa, kuten käskykoodeja ja parametreja.

Ohjelmointilaitteella ohjelmointi tapahtuu seuraavasti:

Valitaan toiminnonohjausnäppäimistöstä käskyä vastaava näppäin. Käskyt jaetaan neljään pääluokkaan:

- paikoitus-
- ohjaus-
- kaarihitsaus- ja
- havaintojärjestelmäkäskyt.

Tämän jälkeen selataan monitoiminäppäimistöä kunnes etsitty käsky löytyy. Käskyyn lisätään argumentteja edellisen monitoiminäppäimien avulla kunnes käskyrivi on valmis.

MALLIT

... kuvastaen itseään
järki hourii tieteet, kaavat,
joilta muka homeet, naavat,
kyvyt mielteet, sielunhaavat,
suorat, käyrät, syvät, aavat,
syntyvän ja kuolleen laavat
ilmenemisluvan saavat.
Täällä syntyy kauniit, rumat
taide-epämuodostumat —
Mutta nytpä alkaa näytös!
—
Lauri Viita, Betonimylläri,
Kootut runot, 1977

Sekä tieteellistekninen että arkipäivän ajattelu ja päätöksenteko perustuvat malleihin. Monet arkipäivan tavat ja tottumukset perustuvat jonkinlaisiin malleihin siitä, mitä normaalisti eri tilanteissa on odotettavissa. Fysiikka ja matematiikka ovat paljolti kaavojen, matemaattisten mallien, käsittelyä. Matemaattisten mallien käsittely tietokoneella on usein melko helppoa ja saatavat tulokset yleensä hyvin tarkkoja. CAD perustuu kolmiulotteisen geometrian mallittamiseen ja käsittelyyn.

tavat

matematiikka

Tietokoneen avulla malleja voidaan käsitellä automaattisesti, jolloin voidaan yhdistää tietokoneen nopeus ja matemaattisten mallien tarkkuus. Tietokoneen yhteydessä matemaattisen mallin käsite on varsin laaja sisältäen monipuolisen joukon erilaisia tietorakenteita. Ongelmana tietokoneen suorittamassa "päättelyssä" on lähinnä se, että tietokoneohjelma keskittyy kerrallaan yhteen mitättömän pieneen[1] asiaan[2]. Ihminen puolestaan käsittelee asioita osittain rinnakkaisesti ja hyvin laaja-alaisesti mutta hitaasti.

päättely

Esimerkkinä yksinkertaisesta matemaattisesta mallista tarkastellaan

[1] muutaman bitin mittaiseen
[2] Tämä riippuu tietysti paljolti tarkastelutasosta.

robotin käsivarren tapaisten mekanismien mallittamista. Mekanismeissa, kuten roboteissa, tarvittavia koordinaatistojen muunnoksia (translaatio ja rotaatio-operaatiot) on kätevintä kuvata 4×4 matriisien avulla. Matriisin muoto on:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & d_1 \\ a_{21} & a_{22} & a_{23} & d_2 \\ a_{31} & a_{32} & a_{33} & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

missä alkiot a_{ij} $(i,j = 1,2,3)$ kuvaavat orientaatiota ja kolmen alkion vektori $(d_1 \;\; d_2 \;\; d_3)$ translaatiota. Menetelmän merkinnällinen kätevyys käy selvästi ilmi tarkasteltaessa ketjua (esim. robotin käsivarsi), jossa kunkin osan i muunnos on kuvattu edellisen kaltaisella matriisilla M_i. Ketjun pään asema voidaan esittää tulona

$$M_1 M_2 \cdots M_i.$$

Robotin ohjaaminen muunnosmatriisien M_i avulla on siis laskennallisesti suoraviivaista. Yleensä ongelma on kuitenkin käänteinen: tunnetaan kärkipisteen paikka ja orientaatio, jolloin matriisit M_i on ratkaistava. Käänteinen ongelma ei ole niinkään helppo ja se joudutaan ratkaisemaan aina mekanismikohtaisesti[3]. Tämä on kuitenkin yleensä robotin valmistajan eikä sovellutusohjelmoijan ongelma.

3.1 Tuotannon mallittaminen

ohjaus

Tarkastellaan seuraavaksi mallien käyttöä tuotantolaitoksessa ja erityisesti robottien ohjauksessa. Ohjausinformaation määrä, laatu ja käyttäjät vaihtelevat huomattavasti tehtaan eri ohjaustasoilla ja osissa. Ylimmillä tasoilla päätökset tehdään paljon pidemmällä aikaperspektiivillä kuin alemmilla tasoilla. Vastaavasti ylemmillä tasoilla rajoitutaan vain oleellisimpien tietojen ja tapahtumien käsittelyyn, kun alimmilla tasoilla on oltava jatkuvasti perillä myös tarkemmista yksityiskohdista.

yleinen

Hyvin erilaisista ja erisuuntaisista vaatimuksista johtuen koko tehdasta ohjaavan järjestelmäkokonaisuuden on oltava mahdollisimman yleinen ja joustava. Ohjausjärjestelmän rakenteessa ei saa liiaksi rajoittaa osasysteemien ohjauksen suunnittelua ja päinvastoin. Ohjauksen tulee olla mahdollisimman itsenäinen ja sen liittäminen muuhun järjestelmään mahdollisimman selkeä.

Tietotekniikan osalta merkittävät ilmiöt ovat piiritekniikan suoma suuri laskentakapasiteetti, hajautetut järjestelmät, asiantuntijajärjestelmät sekä ohjelmoinnin huono tuottavuus. Näistä lähtökohdista voidaan päätellä ne suuntaviivat, joita on ainakin noudatettava, jotta saavutettaisiin onnistunut ja järkevä kokonaistulos:

[3] Tutkimusta varten on kehitetty myös yleisiä mekanismien hallintaohjelmia [Ket86]

- pyritään ratkaisemaan oleelliset ongelmat mahdollisimman yksinkertaisilla ja tehokkailla menetelmillä,
- käytetään mahdollisimman paljon valmista ohjelmistoa,
- oleellinen tehokkuuslisä saavutetaan hajautuksella,
- yksinkertainen ongelmanratkonta automaattisin menetelmin[4] ja
- pyritään välttämään aikaavievää ja virhealtista optimointia mikäli suinkin mahdollista.

Analogia esimerkiksi tuotesuunnitteluun on ilmeinen:

- mahdollisimman vähän komponentteja[5],
- tuotteen tai komponentin tulee täyttää oleelliset toiminnalliset vaatimukset,
- käytetään mahdollisimman paljon standardiosia ja
- suoritetaan rutiinilaskenta tietokoneella.

Edellä esitetyt seikat saattavat tuntua itsestään selviltä, mutta valitettavasti aina jokin tai jotkut niistä pyrkivät suunnitteluprosessin aikana unohtumaan ikävin seurauksin.

3.2 Tuotemalli

tietokanta

Keskeinen yhteinen käsite tuotteen ja tuotannon suunnittelussa sekä esimerkiksi kokoonpanon ohjauksessa on tuotemalli. Tuotemallilla tarkoitetaan yksinkertaisesti tuotteen suunnittelun, valmistuksen ja kokoonpanon kannalta tarpeellisten tietojen joukkoa. Tuotemalli on siis eräänlainen tietokanta, joka sisältää tuotteesta riippuen mitä erilaisinta tietoa tuotteesta ja sen valmistusprosessista. Tämä vaatii sinällään vielä paljon kehittämistä suunnitteluvälineiltä. Esimerkiksi CAD:lta saadaan nykyisin lähinnä vain tuotteen ja tuotantosolun geometriaan liittyviä tietoja, eli CAD:llä voidaan suunnitella:

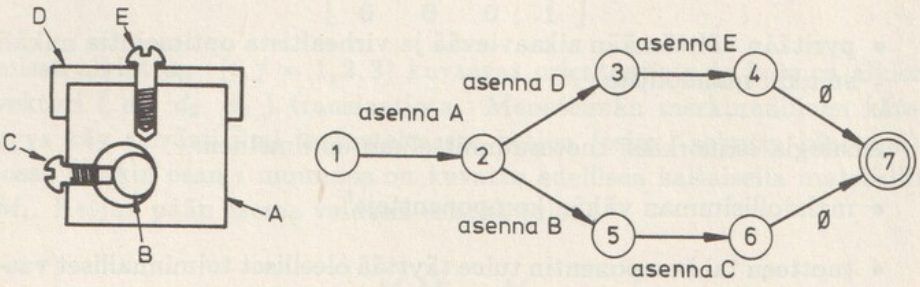
- tuotteen geometria,
- tarvittavien työkalujen yms. geometria ja
- kokoonpanopiirrokset yms.

simulointi

Tämän lisäksi monilla CAD-järjestelmillä on nykyisin mahdollista osittain suunnitella ja simuloida NC-koneiden, robotisoitujen kokoon-

[4] esim. Prolog [Bra86]

[5] Parempi karsia komponentteja kuin optimoida turhia.



Kuva 3.1: Yksinkertainen tuote ja sen kokoonpanoa kuvaava tietorakenne [AN86].

panosolujen tai yleisemmmin erilaisten mekanismien toimintaa. Tämä on kuitenkin valmistuksen kannalta yleensä riittämätöntä. Esimerkiksi kokoonpanon kannalta tuotemallin tulisi sisältää riittävät tiedot komponenteista, kokoonpanojärjestyksestä ja geometriasta sekä tuotteeseen liittyvää yksityiskohtaista tietoa kokoonpanoliikkeistä, -työkaluista yms. Tavoitteena on kuitenkin, että tuotemalli olisi mahdollisimman riippumaton varsinaisen tuotantojärjestelmän yksityiskohdista kuten robotti- ja työstökonekohtaisista ohjaustiedoista, jotka puolestaan sisältyvät tuotantosoluja kuvaaviin malleihin tai tietokantoihin.

3.3 Hierarkiat

Tyypillisesti niin tuotteet kuin tuotantolaitoksetkin ovat rakenteeltaan hierarkisia. Tuotemalli sisältää esimerkiksi alikokoonpanoja tai usean työstövaiheen sarjoja. Alimmalla tasolla ovat yksittäisen työstö- tai kokoonpanovaiheen tapahtumat [Ran85a].

Kuvassa 3.1 on esimerkki yksinkertaisesta kokoonpanosta ja sitä vastaavasta kokoonpanon tuotemallista. Tuote koostuu rungosta A, tapista B, laipasta D ja kahdesta kiinnitysruuvista (C ja E). Tuotemalli sisältää 7 solmua (1: aloitus, 7: lopetus) ja 7 tilasiirtymää (asennetaan runko työskentelyalueelle, asennetaan tappi reikään, asennetaan laippa ja kiinnitetään ruuvit C ja E) sekä kaksi välitöntä siirtymistä lopputilaan.

Malli kiinnittää osittain työvaiheiden suoritusjärjestyksen. Esimerkkitapauksessa tulee C suorittaa ennen D:tä, mutta esimerkiksi E:n ja B:n

järjestys voidaan valita vapaasti kokoonpanosolun tilan mukaan[6]. Vapaus työvaiheiden järjestyksessä antaa solulle joustavuutta myös virhetilanteiden hallintaan; ensimmäinen virhe[7] ei välttämättä johda koko prosessin keskeyttämiseen, vaan työtä jatketaan niiltä osin kuin on mahdollista.

3.4 Ongelmanratkaisu

suunnittelu

Tuotemalli syntyy tuotteen suunnitteluprosessin tuloksena, joko täysin automaattisesti yksinkertaisissa tapauksissa tai vaativammissa tapauksissa suunnittelijoiden ja tietokoneohjelmien vuorovaikutuksessa. Suunnittelutehtäviin löytyy harvoin selkeitä algoritmisia ratkaisuja. Tyypillisesti suunnitellaan pitkälti ns. sormituntumalla eli erilaisten heuristisen menetelmien ja (peukalo)sääntöjen avulla. Näiden käsittely on ainakin osittain mahdollista sääntöpohjaisten taitotuki- eli asiantuntijajärjestelmien ja yksinkertaisen automaattisen ongelmanratkaisun avulla. Laboratorioissa onkin kehitetty järjestelmiä, jotka pystyvät tuottamaan esimerkiksi NC-ohjelmat tuotteen geometriatiedoista joissakin erikoistapauksissa (reiät). Eräitä järjestelmiä on jo ehditty kokeilla myös tuotannossa.

Ilmeisesti lähivuosina asiantuntijajärjestelmiä tullaan käyttämään yksinkertaisten suunnittelutehtävien suorittamiseen joko täysin automaattisesti tai ne toimivat suunnittelijan apuna vapauttaen hänet yksitoikkoisesta, yksityiskohtaisesta, työläästä ja virhealttiista rutiinityöstä.

3.4.1 Tuotemalli sääntöinä

säännöt

Tuotemalli voidaan esittää, paitsi verkkona kuten edellä, myös asiantuntijajärjestelmälle paremmin sopivana sääntöjoukkona. Säännöt voidaan puolestaan esittää useimmista ohjelmointikielistä tutun `IF ... THEN ...` -rakenteen tapaan [Wat86]:

```
IF (ehto) THEN (toimenpiteet)
```

Kuvan 3.1 esimerkkituotteen kokoonpano voidaan esittää sääntöinä:

```
IF (aloitus) THEN (asenna A)
IF (asennettu A) THEN (asenna B)
IF (asennettu A) THEN (asenna D)
IF (asennettu B) THEN (asenna C)
IF (asennettu D) THEN (asenna E)
IF (asennettu C & asennettu E) THEN (valmis)
```

[6] mahdollisuus optimointiin

[7] esimerkiksi puuttuva komponentti

rinnakkaisuus

Sääntöpohjaisen järjestelmän periaate on se, että aina kun ehto on voimassa, vastaavat toimenpiteet voidaan suorittaa riippumatta esimerkiksi sääntöjen kirjoitusjärjestyksestä. Toiminta on sääntöjen osalta[8] rinnakkaista, toisin kuin klassisissa ohjelmointikielissä, joissa ehdot suoritetaan varsin tiukasti ennalta määrätyssä järjestyksessä. Tämä sopii hyvin ongelmiin, joissa tiedot käsittelyaikana eivät oleellisesti muutu. Tilanne on täysin päinvastainen esim. robotin ohjauksessa, jossa jatkuvasti pitää seurata rinnakkaisesti lukuisia antureita.

3.4.2 Sääntöjen etuja

poikkeustilanteet

Virhetilanteiden tai muiden tärkeiden tapahtumien jatkuva tarkkailu on yksinkertaista sääntöjen avulla. Lisäksi uusia sääntöjä esim. ennalta arvaamattomista virhetilanteista toipumiseksi voidaan helposti lisätä tarvittaessa kesken muun toiminnan[9]. Toinen etu on se, että sääntöpohjainen järjestelmä rohkaisee käyttämään runsaasti erilaisia ehtoja ja tarkistuksia esim. antureiden tiloista, eikä niinkään pelkästään "sokeasti määräilemään" tekemään sitä ja tätä kuten klassiset ohjelmointikielet.

Mitä alemmaksi ohjaushierarkiassa mennään, sitä lyhytjänteisemmäksi ja rinnakkaisemmaksi tietojenkäsittely muuttuu. Sääntöpohjaisessa järjestelmässä tällainen rinnakkaisuus on "sisäänrakennettuna".

Vakava ongelma robotteja käytettäessä on niiden törmääminen ohjelmavirheen tms. takia työskentelyalueella olevaan esteeseen. Jos robotin kapasiteetti ja nopeus ovat suuria, voi vahinko olla melkoinen sekä robotille että sen ympäristölle. On siis järkevää tarkistaa aina, että robotin käsivarren nopea liike suoritetaan vain, jos koura on ylhäällä[10]. Tämä onnistuu kahdella yksinkertaisella säännöllä:

```
IF (nopealiike & z=0) THEN (liiku)
IF (nopealiike & z!=0) THEN (varoita("nopealiike!?"),
                             hidasliike)
```

Samanlaisen turvan luominen algoritmisessa ohjauskielessä edellyttää vastaavien ehtojen liittämisen jokaiseen nopeaan siirtymiseen. Tämä on tietysti mahdollista ja helppoakin strukturoidussa ohjelmassa, mutta tilanne käy nopeasti hyvin hankalaksi, jos näiden ehtojen lisäksi halutaan ohjata toimintaa esimerkiksi erilaisilta antureilta saatujen tietojen perusteella. Ja aina on mahdollisuus, että jossakin erikoistapauksessa tarvittavat ehdot syystä tai toisesta unohtuvat, ovat väärässä paikassa tai ovat muuten virheellisiä ...

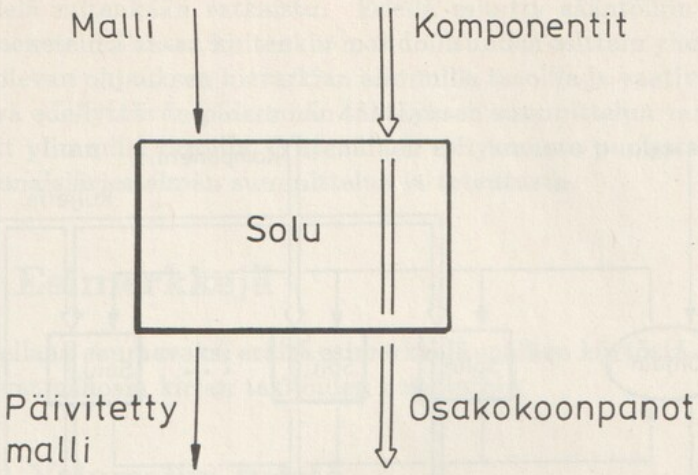
Toistettakoon vielä, että sääntöpohjaisen järjestelmän suurimpia etuja on se, että säännöt ovat varsin riippumattomia ja niitä on helppo lisäillä tarpeen vaatiessa. Näin kehitetään vähitellen järjestelmää yhä paremmaksi ja täydellisemmäksi. Mitä monimutkaisempi ohjausjärjestelmä on,

---

[8] näennäisesti

[9] tulkkaava järjestelmä

[10] so. vapaalla alueella; koordinaatti $z = 0$;tarkastellaan scara-tyyppistä robottia



Kuva 3.2: Tuotantosolun tieto- ja materiaalivirrat.

sitä edullisemmaksi tilanne muuttuu sääntöpohjaiselle järjestelmälle algoritmiseen verrattuna. Kun sääntöpohjainen ohjausjärjestelmä sisältää runsaasti ei-triviaalia tietämystä ja monimutkaisia päättelyketjuja, sitä voidaankin jo hyvällä syyllä kutsua asiantuntijaohjausjärjestelmäksi.

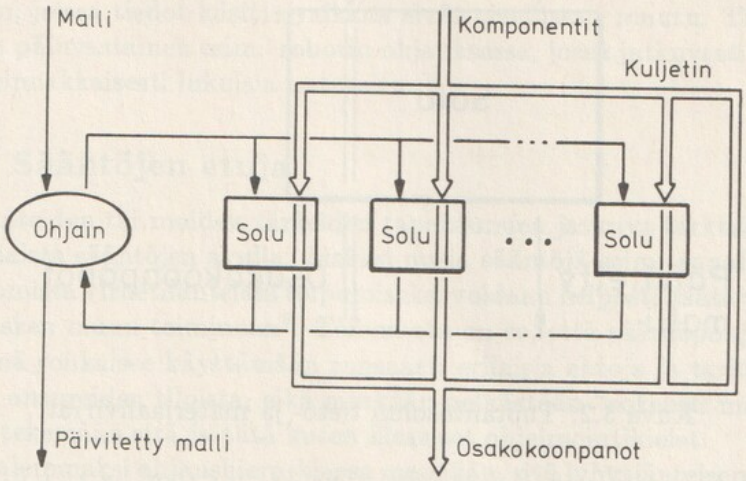
3.5 Tehtaan ohjaus

Tehtaan voidaan ohjauksen kannalta ajatella koostuvan kuvan 3.2 mukaisista soluista, jotka syötteinä saavat komponentteja ja tilauksia tuotemallien muodossa. Nämä solut voivat puolestaan koostua edelleen alisoluista jne. Kuvan 3.3 tapaan ne voivat muodostaa hierarkisia solukkoja.

3.5.1 Hierarkinen järjestelmä ja optimointi

Ohjauksen kannalta tyypillinen tehdas on hierarkinen järjestelmä, jossa ohjausviestejä kulkee pääasiassa ylhäältä alas (komennot) ja alhaalta ylös (kuittaukset ja virhediagnostiikka). Ohjauksen tehtävänä on tuottaa halutut palvelut käytettävissä olevilla resursseilla mahdollisimman tehokkaasti. Kyseessä on optimointitehtävä, jota ei yleensä voi ratkaista tarkasti. Hierarkinen rakenne on edullinen hajoitettaessa ohjaustehtävä mahdollisimman itsenäisiin osiin, jotka ovat kokonaisongelmaa helpommin ratkaistavissa. Albus et al. esittävät seuraavat seitsemän ohjaushierarkian tasoa [AMBF83]:

- tilausten käsittely, prosessin suunnittelu,
- tuotannon ajoitus, resurssien allokointi,



Kuva 3.3: Tuotantosolukko.

- erien muodostus,
- yksinkertaiset tehtävät,
- perusliikkeet,
- x, y, z liikeradat ja
- servojen ohjaus.

Nämä liittyvät luontevasti tehtaan rakennehierarkiaan:

- tehdas,
- verstas,
- solu,
- työasema,
- robotti sekä
- servot ja muut toimilaitteet

Jotta joustava ja ajantasalla oleva ohjaus olisi mahdollinen, tulee järjestelmän pystyä ylläpitämään mallia ympäristöstään saamiensa tietojen perusteella[11]. Malli saattaa sisältää huomattavasti enemmän tietoa

[11] anturit, aistimet, fysiikan lait, "tervejärki"

kuin päällisin puolin voisi ajatella, eikä ongelmaa ole yleisessä tapauksessa vielä mitenkään ratkaistu. Edellä esitetty sääntöihin perustuva ohjausmenetelmä antaa kuitenkin mahdollisuuden osittain yhdistää ajantasalla olevan ohjauksen hierarkian alimmilla tasoilla ja vaativaa loogista päättelyä edellyttävän pidemmän tähtäyksen suunnittelun vaatimat menetelmät ylimmillä tasoilla. Yhtenäinen esitysmuoto puolestaan helpottaa kokonaisjärjestelmän suunnittelua ja toteutusta.

3.6 Esimerkkejä

Tarkastellaan seuraavaksi eräitä esimerkkejä mallien käytöstä. Esimerkit perustuvat pääosin kirjan tekijöiden kokeiluihin.

3.6.1 Valumallin työstö

"styrox"-malli

Eräs tapa valmistaa valuja on käyttää polystyreenistä valmistettuja malleja [TH80] ja [Ran85b]. Polystyreenimallien eduista mainittakoon työstön helppous ja siisteys, mallien keveys ja mahdollisuus hyvin monimutkaisiin muotoihin[12]. Lisäksi polystyreenimallien käyttö sopii hyvin piensarjatuotantoon, jossa CAD:n ja valmistuksen tietotekninen liittäminen toisiinsa on aikaa myöten lähes välttämätöntä.

geometria

simulaattori

Valettavan kappaleen mallittaminen suoritettiin AGX-ohjelmistolla [Tak84] tai [Tak87]. Käytännöllisistä syistä[13] suunniteltu kappale on kahden profiilin leikkaus. Kappale suunniteltiin ensin profiileina, joista muodostettiin automaattisesti varsinaisen valukappaleen malli (kuva 3.4). CAD:n geometriatietojen avulla konstruoitiin edelleen robotin tarvitsema työstöohjelma koordinaattitietoineen.

Haluttaessa voidaan robottiohjelman toimintaa seurata yksinkertaisen simulaattorin avulla. Näin ainakin periaatteessa osa muuten vasta valmistuksessa ilmenevistä lähinnä geometrisista ongelmista voidaan havaita ja korjata.

NC-kone vs. robotti

Tavanomaisiin työstökoneisiinkin on saatavana polystyreenin työstöön sopivia työkaluja. Robottiin verrattuna NC-työstökone on kuitenkin kömpelö ja kallis. Valumallien työstössä riittää hyvin vaatimattomankin robotin tarkkuus. Robotilla on vapausasteitakin yleensä enemmän kuin NC-koneella. Työstön lisäksi robotti voi suorittaa myös muita tarvittavia toimenpiteitä kuten työstetyn kappaleen vaihdon, varsinkin jos robotin työkalut ovat automaattisesti vaihdettavissa.

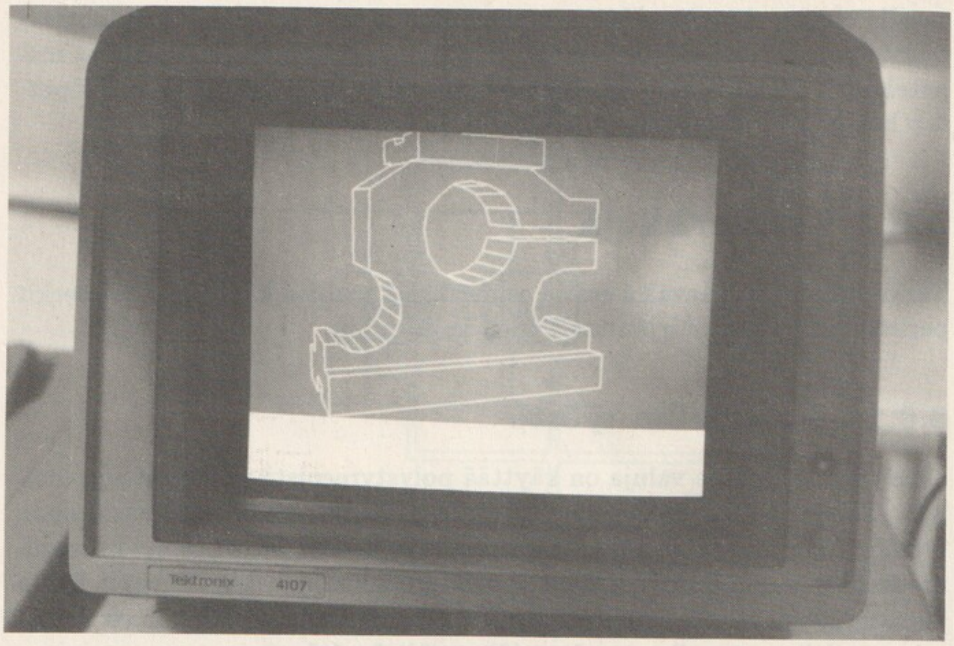
robotti

Käytettävissämme oli IBM 7547 scara-tyyppinen neljän vapausasteen[14] robotti, joka on tarkoitettu lähinnä kokoonpanotehtäviin. Tämän robotin etuina ovat sen suuri tarkkuus (0.05mm) ja kapasiteetti (20kg), joten sillä onnistuu suhteellisen raskaankin työkalun tarkka liikuttelu. Huonoina

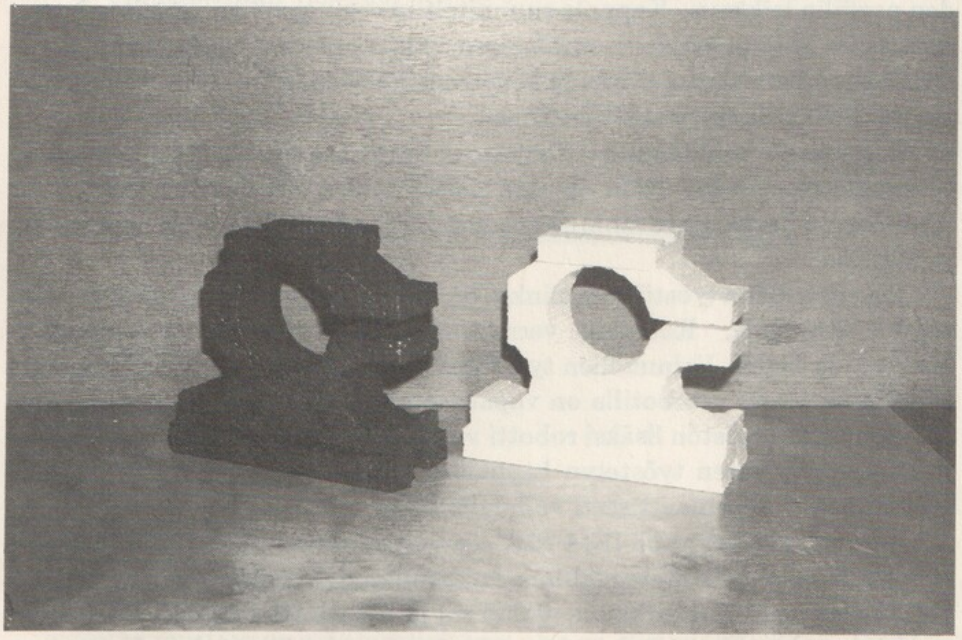
[12] ominaisuuksia, jotka varsinkin kirjoittajien laboratorioympäristössä ovat eduksi

[13] käytetyssä robotissa oli vain 4 vapausastetta.

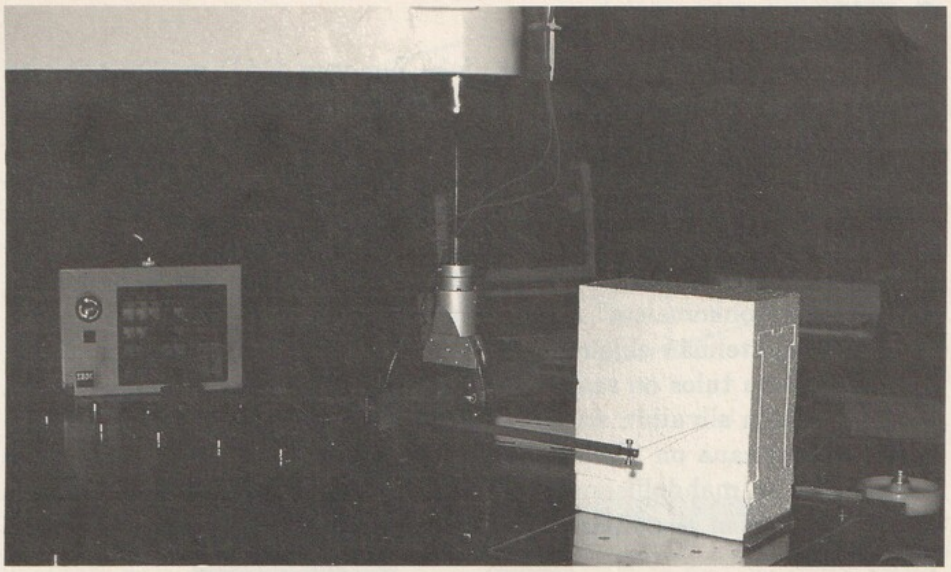
[14] x, y, z ja roll



Kuva 3.4: CAD-malli.



Kuva 3.5: Malli ja valettu kappale.



Kuva 3.6: Työstävä robotti.

puolina tämän tehtävän yhteydessä mainittakoon nopeuden säädön kömpelyys ja scara-robotille tyypillinen vapausasteiden niukkuus. Polystyreenin työstö vastuslangalla onnistui kuitenkin melko hyvin ohjaamalla työkalua pisteestä pisteeseen ja pitämällä kussakin pisteessä pieni tauko (0.1s), jotta langan lämpötila pysyi riittävänä[15].

Kappaleen valu suoritettiin valuraudasta TKK:n Valimotekniikan laboratoriossa (kuva 3.5). Valettu kappale painoi noin 8 kg.

Kokeessa robottia käytettiin vain polystyreenimallin valmistamiseen. Periaatteessa robottia voitaisiin käyttää myös esim. valettujen kappaleiden puhdistukseen ja valupurseiden yms. poistamiseen. Myös näissä tehtävissä voitaisiin robotin ohjelmat tuottaa automaattisesti CAD:n mallitiedoista. Ongelmana olisivat lähinnä anturit, työkalut ja -menetelmät.

3.6.2 Muuraus

Rakentamisen automatisointi, elementtitekniikkaa lukuunottamatta, on ollut varsin vähäistä. Rakentaminen sisältää kuitenkin joitakin työvaiheita, jotka voisivat olla suhteellisen helposti automatisoitavissa CAD:n ja robotiikan avulla. Tällaisia raskaita ja/tai muuten epämiellyttäviä työvaiheita ovat mm. maalaus ja muuraus. Tarkastellaan lähemmin muuraukseen ja sen CAD-liittymään liittyviä periaatteellisia ohjaus- ja ohjelmointikysymyksiä.

[15] Virtalähteenä käytimme yksinkertaista akkulaturia. Tuotantokäytössä kannattaa käyttää rataohjattua robottia ja lisäksi langan lämpötila kannattaa vakioida takaisinkytkennän avulla.

Ongelma

Tarkoitus on selvittää algoritmia, joka annetun CAD-mallin pohjalta tuottaa riittävät tiedot ko. mallin esittämän konstruktion rakentamiseksi annetun tiilivalikoiman puitteissa. TKK:n Tietojenkäsittelyopin laboratoriossa on laadittu BIN-niminen mallittamisohjelmisto [TKH*84], joka soveltuu erityisen hyvin suorakulmaisista osista (lohkoista) koostuvien esineiden mallittamiseen. Normaalisti nämä BIN -mallit ovat approksimaatioita, mutta esim. juuri muurattujen rakennusten osalta ne kuvaavat kohteen "lohkomaista" rakennetta hyvin.

BIN -järjestelmän ohjelmilla voidaan mallia muokata, eli suunnitella, kunnes haluttu tulos on saavutettu. Tämän jälkeen malli voidaan jakaa horisontaalisiin siivuihin siten, että yksi siivu vastaa aina yhtä tiilikerrosta. Ongelmana on tämän jälkeen "täyttää" siivut tiilillä siten, että saatu tulos on mahdollisimman tukeva. Toisin sanoen päällekkäin osuvien tiilien pitäisi olla siten lomittain, että vertikaaliset raot eivät mene päällekkäin.

Ratkaisu perustuu rekursiiviseen proseduuriin `lay`, joka kokeilee kaikkia tiilivaihtoehtoja kussakin nurkkapisteessä. Tiilistä valitaan ensimmäinen, joka ei asetu pystyraon reunalle. Jos tällaista ei ole, valitaan se joka jättää pienimmän pystyraon[16]. Kun tiilet on alunperin pantu valikoimaan pinta-alan mukaan pienenevään suuruusjärjestykseen, ohjelma samalla pyrkii myös käyttämään mahdollisimman suuria tiiliä. Muuttamalla valintakriteerejä, voidaan tarvittaessa tuottaa erilaisia ladontoja.

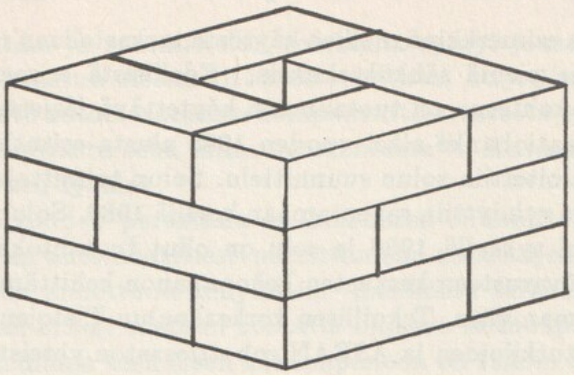
Tiilet ladotaan yksi kerrallaan ja suoritettava optimointi on luonteeltaan lokaalia eli tiilikohtaista. Mikäli ladonnassa joudutaan umpikujaan, peruutetaan, ja yritetään uudelleen seuraavaksi edullisimman tiilen avulla[17]. Ohjelman laatijan suureksi hämmästykseksi tätä peruutusmekanismia ei käytännössä yleensä tarvita, joten algoritmi on varsin tehokas (lineaarinen), vaikka pahimmassa tapauksessa voitaisiin aina joutua purkamaan lähes valmis ladonta, kunnes kaikki mahdollisuudet on käyty läpi (kvadraattinen).

Muurausjärjestelmä

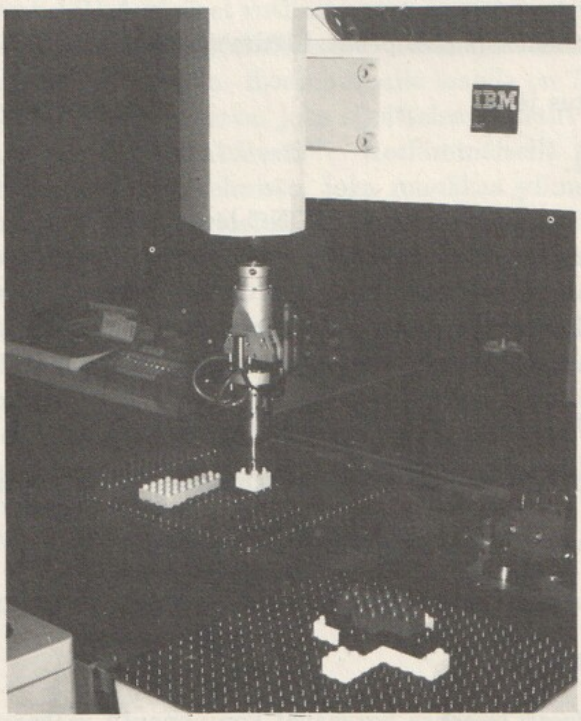
Talonrakennus voidaan muurauksen osalta periaatteessa automatisoida suunnittelemalla rakennus BIN-järjestelmän tapaisella ohjelmalla ja ratkaisemalla tiilien koordinaatit edellä esitetyn kaltaisella ohjelmalla. Loput järjestelmän ongelmat liittyvät mm:

- anturointiin (mm. paikan määritys)
- tiilien käsittelyyn ja kuljetukseen sekä
- laastin automaattisen syöttöön,

[16] Raon pituudesta vähennetään ensin sidottujen ts. tiilen alle jäävien rakojen pituus.
[17] ns. backtracking- eli peruutusmenetelmä



Kuva 3.7: Tiilien muuraus.



Kuva 3.8: Robotti rakentamassa.

jotka kaikki voidaan periaatteessa ratkaista. Menetelmää on kokeiltu IBM scara-robotilla käyttämällä tiilinä Duplo[18] -rakennuspalikoita (kuva 3.8).

[18]Duplo on LEGO:n tavaramerkki

3.6.3 Keskusten kokoonpano

Kolmantena esimerkkinä mallien käytöstä tarkastellaan robottisolua, joka kokoonpanee pieniä sähkökeskuksia. Edellisistä esimerkeistä poiketen tarkastelun kohteena on tuotannossa käytettävä järjestelmä[19]: Kokoonpanoautomaatiohanke alkoi vuoden 1985 alusta esitutkimuksella. Syksyllä 1985 aloitettiin solun suunnittelu. Solun toimittaja selvisi keväällä 1986 ja solua ryhdyttiin rakentamaan kesällä 1986. Solun asennus Mikkeliin tapahtui syksyllä 1986 ja solu on ollut tuotantokäytössä keväästä 1987. Sähkövarusteen keskusten kokoonpanon kehittäminen suoritettiin yrityksen oman väen, Teknillisen korkeakoulun Tietojenkäsittelyopin laboratorion tutkijoiden ja ASEAN robottiosaston yhteistyönä.

Tuotettavat keskukset koostuvat neljästä pääosasta:

- kehys eli runko, johon komponentit ja kannet asennetaan,
- komponentit (sulakepesät, liittimet, releet yms.),
- johdotus ja
- kannet.

Kannet ja kehykset valmistetaan CNC-levytyöstökoneella Mikkelissä. Kehys hitsataan levyosista robotilla. Osa komponenteista valmistetaan Porvoon komponenttitehtaalla loput ostetaan yrityksen ulkopuolelta. Johdot ostetaan keloina ja katkaistaan määrämittaan automaattikoneella. Automaation kannalta tuotteessa on mm. seuraavat positiiviset piirteet:

- osien tuotanto osittain automatisoitu[20],
- komponenttien kiinnitys yksinkertainen ns. sneppimekanismi ja
- asennussuunta sama kaikilla komponenteilla.

Toisaalta automatisoinnin kannalta hankalia asioita ovat mm.:

- johdotus,
- sähköisesti toisensa korvaavat komponentit mekaanisesti erilaisia,
- kansien kiinnitys ruuveilla ja
- komponenttien läpiviennit.

Keskusten kokoonpano on ollut toistaiseksi käsityötä. Ainoita apuvälineitä ovat olleet ruuvin väännin ja erilaiset kohdistusta helpottavat sabluunat. Tämä oli lähtökohta, kun kokoonpanon automaatiota alettiin tutkia. Tutkijoille ja suunnittelijoille annettiin vapaat kädet ideoida tuotannon kehittämistä tarvittaessa hyvinkin radikaalisti.

[19] Oy Sähkövaruste Ab, Mikkeli

[20] pellin käsittely, kehysten hitsaus, joidenkin komponenttien valmistus

Kokoonpanosolu

Ideoiden karsinnan ja suunnittelun tuloksena syntyi joustava kokoonpanosolu, joka koostuu Asean IRb1000-robotista, kuljettimesta, paternoster-tyyppisestä automaattisesta komponenttivarastosta ja tarvittavasta ohjauselektroniikasta sekä mikrotietokoneesta[21]. Kuvassa 3.9 on esitetty robotti ja sen työalue.

Solun joustavuuden perustana on keskuksen suunnitteluohjelmisto, joka pystyy täysin automaattisesti suorittamaan annettujen komponenttien sijoittamisen annettuun kehykseen. Saaduista tarkoista mitoitustiedoista luodaan sitten edelleen robottia ohjaava kokoonpano-ohjelma. Muutoksien tekeminen keskuksen kokoonpanoon on tällöin erittäin helppoa. Periaatteessa kaikki tuotettavat keskukset voivat olla erilaisia.

Robotin ohjelmointi on jaettu kolmeen tasoon (kuva 3.10). Alimmalla tasolla ovat ARLA-kieliset rutiinit, jotka huolehtivat komponenttikohtaisista kokoonpanon yksityiskohdista[22]. Alimman tason rutiinit ohjelmoidaan käsin robotin avulla. Korkeimmalla tasolla on Turbo Prolog-kielinen [TUR86] ohjelma `sijoita`, joka sijoittelee annetut komponentit haluttuun kehykseen automaattisesti[23]. Keskimmäisellä tasolla on C-kielellä [KR78] toteutettu ohjelmisto, joka muokkaa ylimmältä tasolta saamansa sijaintitiedot ja muodostaa niistä alimman tason komponenttikohtaisten aliohjelmien kanssa ARLA-kielisen ohjelman, joka kokoaa annetun keskuksen.

Ohjelmiston jaolla on pyritty muun muassa hyödyntämään erilaisten ohjelmointikielien piirteitä ja ratkaisemaan kullakin kielellä, sille parhaiten sopiva tehtävä. Ylimmän tason tehtävä vaatii kehittyneitä tietorakenteita ja tehokasta ongelman ratkaisua. Keskimmäisellä tasolla vaaditaan sujuvaa tiedostojen ja tekstin käsittelyä. Alimmalla tasolla ohjataan suoraan itse robottia.

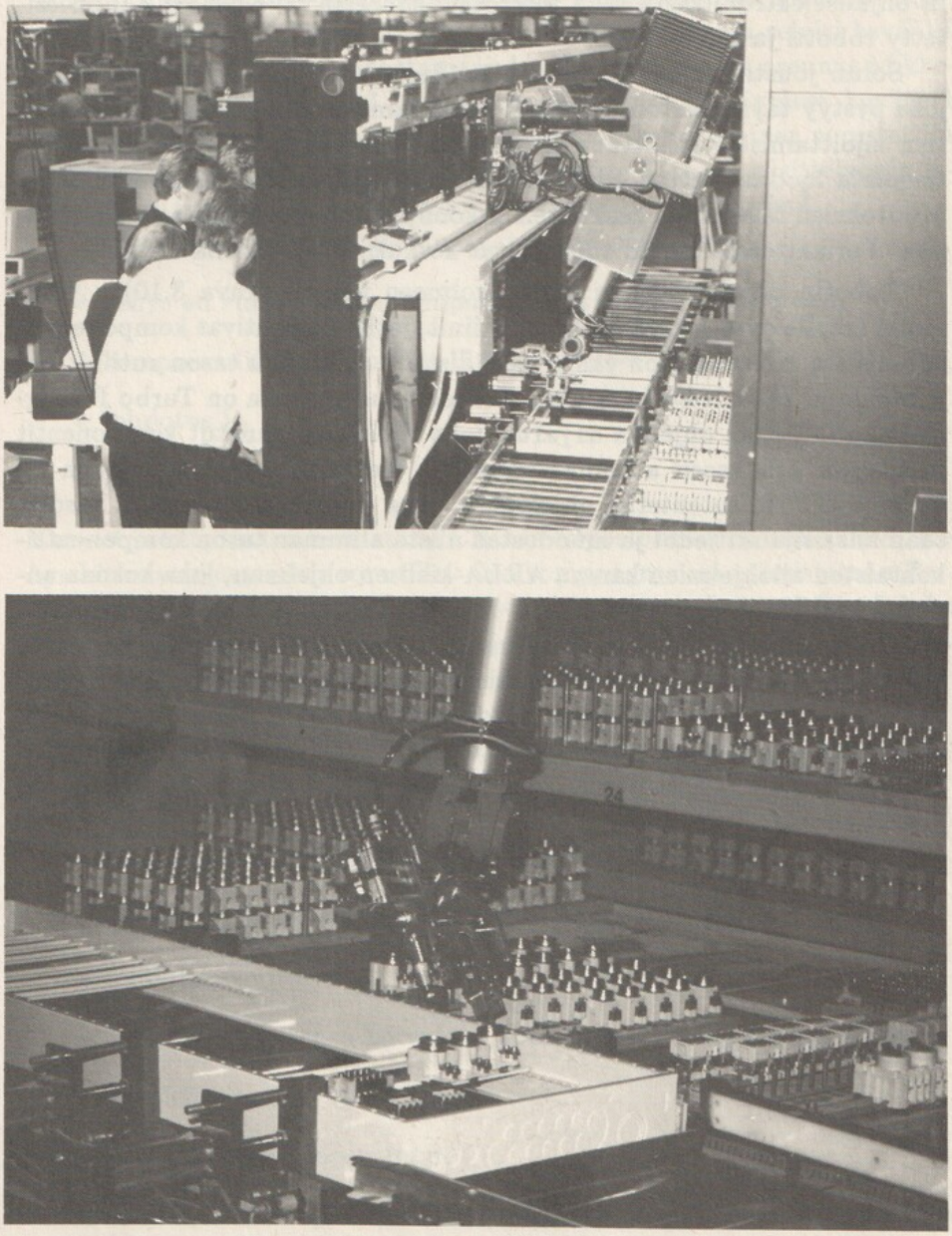
Lähteet: [Ala87a], [Ala87b], [AR88] ja [TKA88].

---

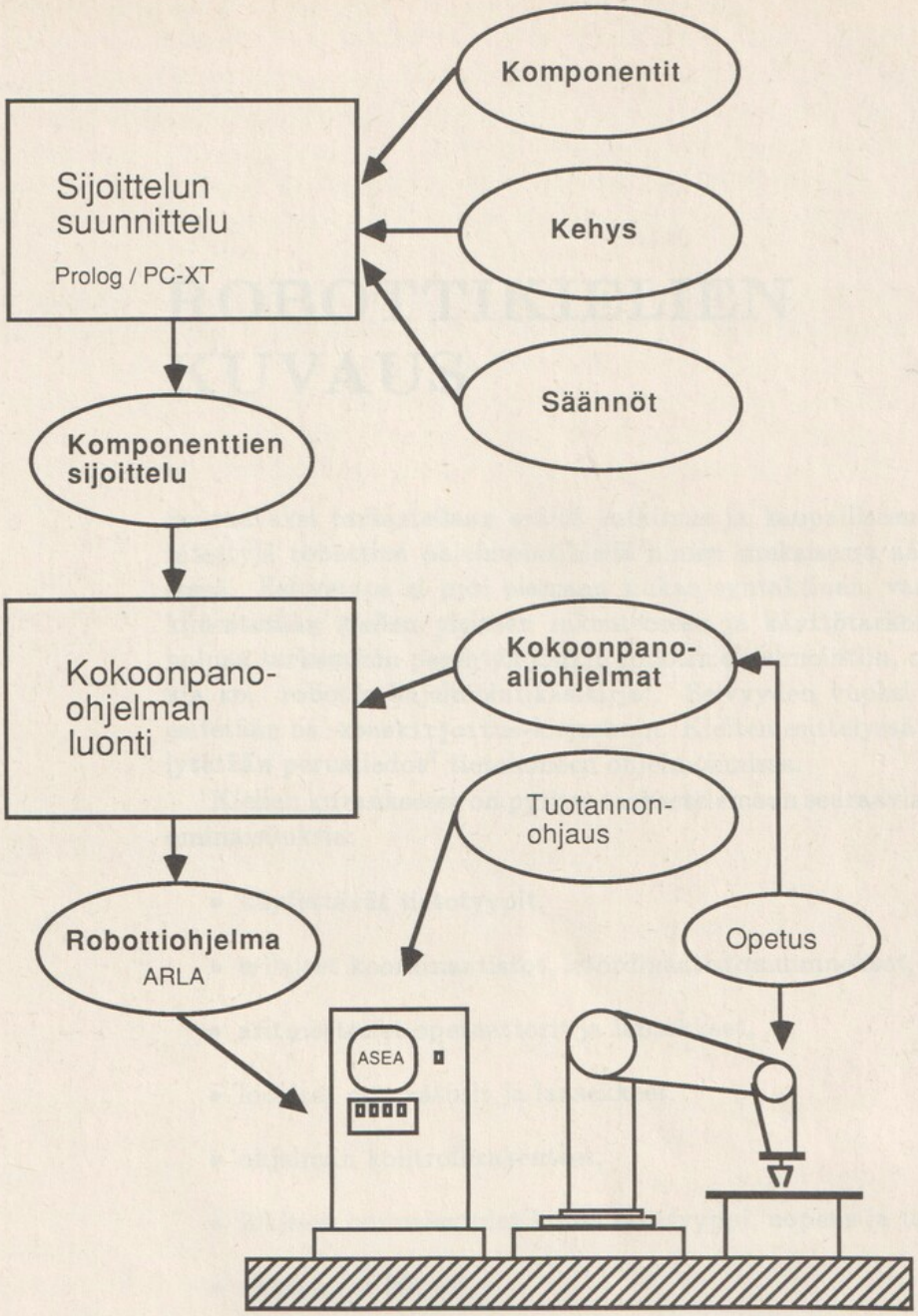
[21] IBM PC/XT

[22] sneppimekanismi on eri komponenteilla hieman erilainen

[23] käyttäjä voi myös editoida tulosta yksinkertaisen graafisen editorin avulla



Kuva 3.9: Robotti kokoaa keskusta.



Kuva 3.10: Ohjelmiston rakenne.

ROBOTTIKIELIEN KUVAUS

Seuraavaksi tarkastellaan eräitä tutkimus ja kaupalliseen käyttöön kehitettyjä robottien ohjelmointikieliä nimen mukaisessa aakkosjärjestyksessä. Esitystapa ei pyri olemaan tiukan syntaktinen, vaan päähuomio kiinnitetään kielien yleiseen rakenteeseen ja käyttötarkoitukseen. Jos haluaa tarkemmin perehtyä jonkin robotin ohjelmointiin, on syytä hankkia ko. robotin ohjelmointikäsikirja[1]. Selvyyden vuoksi ohjelmateksti esitetään ns. `konekirjoitus`-kirjasimin. Kielten esittelyssä lukijalta edellytetään perustiedot[2] tietokoneen ohjelmoinnissa.

Kielien kuvauksessa on pyritty tarkastelemaan seuraavia robottikielien ominaisuuksia:

- käytettävät tietotyypit,
- erilaiset koordinaatistot, koordinaatistomuunnokset,
- aritmeettiset operaattorit ja lausekkeet,
- loogiset operaatorit ja lausekkeet,
- ohjelman kontrollirakenteet,
- liikkeen ominaisuudet kuten ratatyyppi, nopeus ja tarkkuus,
- työkalujen käyttö,
- anturit ja aistimet (erityisesti näkö),
- ulkoiset liitynnät (DI/DO[3]-portit) ja
- rinnakkainen prosessointi.

---

[1] Useat robottien maahantuojat tarjoavat jopa suomenkielisiä käsikirjoja.

[2] esim. Pascal-kielen tuntemus

[3] DI = Digital Input ja DO = Digital Output

Tiedot perustuvat lähinnä seuraaviin robottikielivertailututkimuksiin [BS82], [PL83], [GG85], [Loz83] ja [GSCT84] sekä joidenkin kielien kohdalla käsikirjoihin. Lähteiden epätasaisuuden takia ei kaikkia kieliä ole voitu tarkastella yhtä yksityiskohtaisesti. Lisäksi kaikista kielistä ei ollut mahdollista saada viimeisintä tietoa.

Kaikkia esitettyjä kieliä ei enää käytetä. Käytöstä poistuneita kieliä on otettu mukaan lähinnä siksi, että niistä voi seurata kielien kehitystä.

4.1 AL

Wave-kielestä saatujen kokemusten perusteella ryhdyttiin Stanfordin yliopistossa vuonna 1974 kehittämään AL-ohjelmointikieltä. Tämä oli kunnianhimoinen yritys kehittää korkeantasoinen kieli, johon sisältyisi kaikki robotin ohjaamisessa tarvittavat piirteet sekä silloiset yleisten kielien[4] parhaat puolet. Ensimmäiset AL-kieliset (Arm Language) ohjelmat laadittiin vuoden 1975 loppupuolella ja ensimmäinen versio kielestä valmistui lopullisesti seuraavana vuonna. AL-kieli on luokiteltavissa rakenteisen tason kieliin mutta se sisältää myös piirteitä tehtävänläheisistä kielistä.

yleinen

Tämän hetkistä AL-kielen tutkimusversiota käytettäessä tarvitaan ohjelmoinnissa ja ohjelman kääntämisessä PDP-10-tietokonetta, robotin ohjaus tapahtuu PDP-11/45:llä. Kehitteillä on suppeampi kaupallinen, lähinnä teollisuuden käyttöön tuleva versio, jonka käytössä riittää PDP-11/45.

PDP-11/45

AL-kieli muistuttaa syntaksiltaan Algolia. Siihen on lisätty robotin ja ympäryslaitteiden ohjauksessa tarvittavia välineitä. AL-kieltä voidaan käyttää esimerkiksi useamman robotin samanaikaisen työskentelyn ohjaukseen. Seuraavassa esitetään joitakin AL-kielen ominaisuuksia.

Algol

AL-kielessä on mm. seuraavat tietotyypit:

tyypit

SCALAR	reaaliluku,
VECTOR	kolmen komponentin muodostama vektori,
ROT	kiertokulma tietyn vektorin suhteen,
FRAME	määrittää kohteeseen liittyvän koordinaatiston sijainnin,
TRANS	kuvaa eri koordinaatistojen välisen suhteen ja
EVENT	tapahtuma; usean robotin toiminnan synkronointi[5].

Näitä perustietotyyppejä voidaan käyttää myös taulukkoina (ARRAY).

Ohjausrakenteet ovat tavanomaiset BEGIN ... END -lohkot, IF ... THEN ... ELSE ... , WHILE ... DO ... , DO <lause> UNTIL<ehto>, FOR-silmukat ja CASE-lauseet. Lauseet voidaan suorittaa rinnakkaisesti lohkojen COBEGIN ... COEND avulla. Parametroituja aliohjelmia ja funktioita voidaan myös käyttää.

rakenteet

AL-kielen paikalliset koordinaatistot voidaan liittää toisiinsa AFFIX-käskyllä. Koordinaatistojen suhde voi olla vakio, kuten kiinteästi toisiinsa liitetyillä kappaleilla. Suhde voi olla myös joustava. Muutettaessa toisen kappaleen paikkaa päivittyy myös tähän liitetyn kappaleen paikka automaattisesti. Päivitys voi aiheuttaa toisen kappaleen paikan muutoksen (RIGIDLY) tai paikka pysyy muuttumattomana (NONRIGIDLY). Koordinaatistojen liittäminen AFFIX-käskyllä voidaan kumota käskyllä UNFIX.

koordinaatistot

[4] lähinnä Algol ja Pascal

[5] SIGNAL<tapahtuma> ja WAIT<tapahtuma> -rakenteiden avulla.

liikkeet

AL-kielessä voidaan liike tai paikan muutos toteuttaa usealla eri tavalla. Kappaleen tai tarttujan liike tiettyyn pisteeseen karteesisessa koordinaatistossa esitetään komennolla `MOVE <mika> TO <mihin>`, johon voidaan liittää myös useita liikkeeseen vaikuttavia ja liikkeen seurantaan liittyviä termejä. `DURATION`-termi käyttää liikkeen suorittamiseen tavallista pidemmän ajan. `DEPROACH`-termi määrittelee lähestymis- ja poistumispisteet kohteesta. `SPEED_FACTOR`-termillä muutetaan liikkeen nopeutta. Liikerata voidaan pakottaa `VIA`-termillä kulkemaan tietyn pisteen kautta. Mikäli liikkeen loppupisteen tarkkuudella ei ole väliä, voidaan liikettä nopeuttaa `NO_NULLING`-termillä. Toisaalta hyvin tarkasti (ja hitaasti) päätepisteeseen saavutaan `NULLING`-termillä. Robotin ulointa vartta voidaan vaaputtaa käyttämällä `WOBBLE`-termiä. Liikkeen laajuus voidaan myös määrätä. Liikkeen aikana voidaan tarkkailla robottiin vaikuttavia vääntöjä ja voimia. Komennoilla `FORCE` ja `TORQUE` voidaan pysäyttää robotin liike määrätyn suuruisen voiman tai väännön vaikuttaessa siihen.

anturit

Robotin jokaisessa tarttujassa on mahdollista käyttää kosketusanturia. Näiden sekä `CENTER`-komennon avulla on mahdollista toteuttaa itseohjautuva kappaleeseen tartunta seuraavasti: Tarttuja sulkeutuu kunnes toinen antureista koskettaa kappaletta. Tämän jälkeen robotin varsi ja tarttuja liikkuvat kunnes molemmat anturit koskettavat kappaletta ja tarttuja sulkeutuu lopullisesti. Näin hieman epämääräisessä paikassa olevaan esineeseen voidaan tarttua liikuttelematta sitä.

tarttuja

Tarttujan avaaminen ja sulkeminen tapahtuu komennoilla `OPEN` ja `CLOSE`. Myös näiden komentojen yhteydessä voidaan käyttää `NULLING`- ja `NO_NULLING`-termejä. Työkaluna käytettävän ruuvaimen kierrosnopeutta ja -suuntaa voidaan säädellä komennolla `OPERATE`.

Kaiken kaikkiaan AL-kieli on yksi kehittyneimmistä robottien kielistä, jolla on ollut huomattava vaikutus useisiin myöhempiin kieliin. Edellä esitettiin vain joitakin AL-kielen tärkeimpiä piirteitä. Lisäpiirteitä selviää seuraavaksi esitettävässä esimerkissä, jossa robotti poimii kappaleen ja asettaa sen aukkoon [Loz83]:

Lähteet: [Blu85], [BS82], [PL83], [GSCT84], [Loz83] ja [Muj80].

```
BEGIN "insert peg into hole"
  FRAME peg_bottom,peg_grasp,hole_bottom,hole_top;
  (The coordinates frames represent actual positions of object features,
  not hand positions)
  peg_bottom <- FRAME(nilrot,VECTOR(20,30,0)*inches);
  hole_bottom <- FRAME(nilrot,VECTOR(25,35,0)*inches);
  (Grasping positions relative to peg_bottom)
  peg_grasp <- FRAME(ROT(xhat,180*degrees),3*zhat*inches);
  tries <- 2;
  grasped <- FALSE;
  (The top of the hole is defined to have a fixed relation to the bottom)
  AFFIX hole_top to hole_bottom RIGIDLY AT TRANS(nilrot,3*zhat*inches);
  OPEN bhand TO beg_diameter+1*inches;
  (Initiate the motion to the peg, note the destination frame)
  MOVE barm TO beg_bottom*peg_grasp;
  WHILE NOT grasped AND i<tries DO
    BEGIN "attempt grasp"
    CLOSE bhand TO 0*inches;
    IF bhand < peg-diameter/2
      THEN BEGIN "no object in grasp"
        OPEN bhand TO peg_diameter+1*inches;
        MOVE barm TO 0-1*inches; (0 indicates current location)
        END
      ELSE grasped <- TRUE;
    i <- i+1;
    END
  IF NOT grasped THEN ABORT ("Failed to grasp the peg");
  (Establish a fixed relation between arm and peg)
  AFFIX peg_bottom TO barm RIGIDLY;
  (Note that we move the peg_bottom, not barm)
  MOVE peg_bottom TO hole_top;
  (Test if a hole is below us)
  MOVE barm TO 0-1*inches;
    ON FORCE(zhat)>10*ounces DO ABORT ("No Hole");
    (Exert downward force, while complying to side forces)
  MOVE peg_bottom to hole_bottom DIRECTLY
    WITH FORCE_FRAME=station IN WORLD
    WITH FORCE(zhat)=-10*ounces
    WITH FORCE(xhat)=0*ounces
    WITH FORCE(yhat)=0*ounces
    SLOWLY;
END "insert peg in hole"
```

4.2 AML

yleinen

IBM:n kehittämän AML-kielen (**A** **M**anufacturing **L**anguage) ensimmäinen versio julkistettiin vuonna 1977. Kielen kehittelyssä tavoitteena oli, kuten AL-kielenkin kohdalla aivan uuden ohjelmointikielen luominen lisäämällä yleisten ohjelmointikielten hyviksi todettuihin ominaisuuksiin robotin ja ympäryslaitteiden ohjaamisessa tarvittavia piirteitä. Kun AL-kielessä painopiste on kieleen rakennetuissa robotin ohjauksesta monipuolisesti hoitavissa komennoissa, sisältää AML enemmän koko järjestelmän hallintaan liittyviä komentoja. Näistä mainittakoon AML-kieleen sisältyvät opetusmenetelmän ja näköjärjestelmien käyttömahdollisuudet.

IBM Series/1

Vuosien kuluessa on AML:stä kehitetty useita kaupallisia ja puhtaasti tutkimuskäyttöön tarkoitettuja versioita. Viimeksi mainitut yleensä monipuolisempina ja laajempina vaativat toimiakseen IBM Series/1 tai vastaavan minitietokoneen. Laaja versio AML-kielestä on käytössä myös IBM:n myymässä 7565-portaalirobotissa. Viime vuosina on IBM:ltä tullut markkinoille joukko scara-tyyppisiä robotteja, joiden ohjaamisessa käytetään suppeampaa, AML/Entry, versiota. Tämä versio sopii IBM PC/XT mikrotietokoneisiin. Scara-robottien ohjaamisessa tällä hetkellä käytettävä versio[6] on vielä keskeneräinen. Odotettavissa on monia uusia versioita. Seuraavaksi tarkastellaan tärkeimpiä AML-kielen piirteitä. Aluksi käsitellään laajempaa, lähinnä 7565-robottien versiota ja lopuksi joitakin scara-robotteissa käytettävien versioiden ominaisuuksia.

LISP & APL

AML:ssa on havaittavissa eräitä yhteisiä piirteitä LISP- ja APL-kielien kanssa. Sekä AML että LISP eivät tee eroa järjestelmän ja käyttäjän ohjelmien välille. AML:n aggregaatit muistuttavat LISP:n listoja, lisäksi aggregaattien ja listojen käsittely näissä kielissä on lähes samanlaista. AML-kielen tietotyypit voivat olla skalaareita tai aggregaatteja[7]. Skalaariset tietotyypit voivat olla edelleen kokonaislukuja, reaalilukuja, merkkijonoja, viitteitä, tunnuksia ja osoitteita. Aggregaatti puolestaan on joukko[8] skalaareita tai aggregaatteja. Aggregaatti voi sisältää useita eri tietotyyppejä (epähomogeeninen aggregaatti).

lausekkeet

AML-kieltä voidaan pitää lausekeorientoituneena kielenä, sillä jokaista oikein muodostettua lauseketta voidaan edelleen käyttää osana toista lauseketta. Aritmeettiset operaatiot ovat tavanomaiset: `+`, `-`, `*` ja `/`sekä `IDIV` (kokonaisjako). Vertailuoperaatiot ovat `EQ`, `NE`, `LT`, `LE`, `GE`, `GT` ja `VS`, missä viimeksi mainittu operaatio antaa tulokseksi -1, 0 tai 1 riippuen siitä, onko ensimmäinen operandi pienempi, yhtäsuuri tai suurempi kuin toinen. AML-kielen loogiset operaatiot ovat `AND`, `OR`, `NOT` ja `XOR`. Epätavallista moniin muihin robottien ohjelmointikieliin verrattuna on mahdollisuus käyttää viitelausekkeita. Esimerkiksi `&A` viittaa muuttujaan `A`. Viittaus kumotaan `!`:llä. AML-kielessä on myös hyvät mahdollisuudet käsitellä aggregaatteja ja operoida niillä. Aggregaatteja voi-

[6] Versio 4.0

[7] eli ryhmittymiä

[8] lista

daan esimerkiksi yhdistellä ja niihin voidaan soveltaa erilaisia sijoitusoperaatioita. Aggregaatista voidaan valita haluttu alkio, joka voidaan helposti merkitä, jne.

aliohjelmat

Ohjauslauseet ovat tyypilliset IF ... THEN ... ELSE ... , WHILE ... DO ... , REPEAT ... UNTIL ... -lauseet sekä BEGIN ... END -lohko. AML-kielinen ohjelma perustuu aliohjelmien käyttöön. Kielessä onkin monipuoliset aliohjelmien käsittelymahdollisuudet. Aliohjelmakutsuun voidaan liittää muodollisia parametreja, aliohjelmaa voidaan kutsua rekursiivisesti. Erityisiä aliohjelmakomentoja ovat mm.:

RETURN	palauttaa aliohjelmasta arvon kutsukohtaan,
BRANCH	siirrytään virhetilanteessa tiettyyn osoitteeseen,
QUIT	siirrytään halutulle tulkin komentotasolle ja
APPLY	aliohjelmakutsu annetulla parametrijoukolla.

laskenta

Erityisesti tutkimuskäyttöön tarkoitetussa versiossa on runsaasti valmiita laskennallisia aliohjelmia. Tavanomaisten trigonometristen funktioiden lisäksi on aliohjelmat muun muassa merkkijonojen ja aggregaattien pituuden laskentaan sekä matriisien ja vektoreiden käsittelyyn soveltuvia aliohjelmia. Viimeksimainituista useimmat on lainattu APL-kielestä.

liikkeet

Myös robotin ohjaamisessa ja ympäristön hallinnassa tarvittavia komentoja on runsaasti. Robotin liikkeitä ohjataan esimerkiksi MOVE, AMOVE, DMOVE ja GUIDE-komennoilla, jotka ovat nivelkohtaisia. Esimerkiksi komento MOVE(1,10.2); siirtää nivelen 1 pisteeseen 10.2. AMOVE(1,10.2); aiheuttaa saman siirron mutta siirron aikana voidaan ohjelmassa suorittaa rinnakkaisia toimia, kuten laskentaa. Parametreina voi olla useita niveliä ja vastaavasti pisteitä esimerkiksi aggregaattina. Liikkeen nopeuteen voidaan vaikuttaa komennoilla SPEED, ACCEL ja DECEL. AML-kielen tutkimusversiossa on erityinen komento GUIDE, jolla robotti voidaan asettaa käsiohjaukselle. Komennolla WAITMOVE odotetaan liikkeen loppuunsuorittamista ja komennolla STOPMOVE pysäytetään suoritettava liike. Samalla palautuu parametrien aggregaatti, jota voidaan käyttää myöhemmin soveltamalla APPLY-aliohjelmaa komentoon AMOVE. Robotin liikkeen kohde ja asento saadaan tulostettua komennoilla QGOAL ja QPOSITION.

IO

AML-kielessä on myös erityiset komennot kommunikointiin I/O -porttien välityksellä. Esimerkiksi komennolla SENSIO voidaan haluttu portti aktivoida tai halutusta portista voidaan lukea, lähinnä antureista saatavaa tietoa. Erityisen käyttökelpoinen on komento MONITOR, jonka avulla on mahdollista seurata tietyin aikavälein useita antureita samanaikaisesti.

Edellä on esitetty joukko AML:n tärkeimpiä ominaisuuksia. Tarvittaessa tarkempaa selvitystä kannattaa tutustua viitteeseen [TSM82]. Seuraavassa esimerkkinä jo AL-kielen yhteydessä esitetty kappaleen siirtoohjelma [Loz83]:

```
PICKUP: SUBR(PART_DATA, TRIES);
  MOVE(GRIPPER,DIAMETER(PART_DATA)+0.2);
  (<1,2,3>,XYZ_POSITION(PART_DATA)+<0,0,1>);
  TRY_PICKUP(PART_DATA, TRIES);
  END;
TRY_PICKUP: SUBR(PART_DATA, TRIES);
  IF TRIES LT 1 THEN RETURN ('NO PART');
  DMOVE(3, -1.0);
  IF GRASP(DIAMETER(PART_DATA))='NO PART'
    THEN TRY_PICKUP(PART_DATA, TRIES-1);
  END;
GRASP: SUBR(DIAMETER, F);
  FMONS: NEW APPLY($MONITOR; PINCH_FORCE(F));
  MOVE(GRIPPER, 0, FMONS);
  RETURN (IF QPOSITION(GRIPPER) LE DIAMETER/2
            THEN 'NO PART'
            ELSE 'PART');
  END;
INSERT: SUBR(PART_DATA, HOLE);
  FMONS: NEW APPLY($MONITOR, TIP_FORCE(LANDING_FORCE));
  MOVE(<1,2,3>, HOLE+<0,0,.25>);
  DMOVE(3,-1.0, FMONS);
  IF QMONITOR(FMONS)=1
    THEN RETURN('NO HOLE');
  MOVE(3, HOLE(3)+PART_LENGTH(PART_DATA));
  END;
PART_IN_HOLE: SUBR(PART_DATA, HOLE);
  PICKUP (PART_DATA, 2.);
  INSERT (PART_DATA, HOLE);
  END;
```

4.2.1 AML/Entry

scara

IBM:n markkinoimien scara-tyyppisten kokoonpanorobottien ohjelmoinnissa käytetään AML:n supistettua versiota AML/Entry:ä, josta on poistettu scara-roboteissa käyttökelvottomien piirteiden lisäksi useita ohjelmoinnin kannalta välttämättöminä pidettäviä ominaisuuksia, kuten ohjausrakenteet, lausekkeet jne. Tosin uusimmissa AML/Entryn versioissa on eräitä ominaisuuksia palautettu. Esimerkiksi viimeisimpään versioon 4.1 on lisätty mm. lausekkeiden käyttömahdollisuus. Myös kommunikointi robotin ja ohjelmoinnissa käytettävän IBM PC:n välillä voisi olla helpompaa. Tällä hetkellä ohjausohjelmat laaditaan IBM PC:llä, josta ne käännöksen jälkeen siirretään ohjausyksikköön. Suorituksen aikana on ohjelman kulkuun PC:n välityksellä vaikea puuttua. Edistystä tässä suhteessä ovat versiossa 4.0 olevat komennot PUT ja GET, joiden avulla voidaan välittää tietoa sekä robotille että robotilta.

Nykyisin käytössä olevien AML/Entry versioiden toiminnot voidaan jakaa seuraavasti:

- robotin varren ja tarttujan liikkeet,
- työskentelypisteiden määritys johdattamalla,
- I/O-ohjaus,
- ohjelman ohjaus,
- palettien käsittely,
- lineaarinen liike,
- liikkeen nopeus ja tarkkuus sekä
- kommunikointi ohjaustietokoneen kanssa.

AML/Entryssä voidaan käyttää myös aggregaatteja. Versiossa 4.0 on lisäksi mahdollista yhdistää pisteitä tai muita alkioita ryhmiksi GROUP määrittelyllä. Tarkasteltavaan alkioon viitataan ryhmän nimen ja alkiota vastaavan indeksin avulla.

Ohjelmointi AML/Entry:llä perustuu aliohjelmien käyttöön. Toistorakenteet, kuten `IF ... THEN ... ELSE ...` ja `WHILE ... DO ...` puuttuvat, mutta toisto voidaan toteuttaa `ITERATE`-komennolla. *ITERATE*

Robotin varren liikettä ohjataan komennoilla `PMOVE` ja `DPMOVE`. `PMOVE`-komennossa ilmoitetaan liikkeen päätepiste koordinaatteina tai aikaisemmin määritellyn pisteen nimi parametrina. `DPMOVE`-komennolla siirretään vartta ilmoitettuun suuntaan. Z-akselin liike määrätään `ZMOVE`-komennolla ilmoittamalla liikkeen päätepisteen z-koordinaatti. Liikkeiden nopeutta voidaan säätää `PAYLOAD`-komennolla. Lineaarinen liike toteutetaan `LINEAR`-komennolla. Liikkeen tarkkuutta voidaan muuttaa komennolla `ZONE`. Viivettä liikkeisiin saadaan komennolla `DELAY`. Liikkeen ohjauksessa voidaan käyttää myös `GUARDI` komentoa, joka tarkkailee asetetun DI-portin tilaa ja lopettaa liikkeen portin tietyssä tilassa. Liikkeen pysähtyessä robotin asento voidaan `WHERE`-komennolla tallettaa ja siirtää `PUT`-komennolla isäntätietokoneelle. Samalla ilmoitetaan esimerkiksi virhetilanteesta.

Tarttujan avaamisesta ja sulkemisesta huolehtivat vastaavasti `RELEASE` ja `GRASP`-komennot. Kokoonpanotehtävissä joudutaan usein poimimaan esineitä erilaisilta paleteilta, makasiineilta, jne. Tätä varten on useissa kielissä valmiit komennot, joiden avulla palettien käsittely on helppoa. AML/Entry:ssä voidaan määritellä paletti `STATIC PALLET`-määrittelyllä. Palettien käsittelyssä käytettäviä komentoja ovat: *paletti*

`NEXTPART` lisää paletin kappalelaskuria yhdellä,

`SETPART` annetaan lähtöarvo kyseiselle laskurille,

`GETPART` vie tarttujan seuraavaksi noudettavan kappaleen kohdalle,

PREVPART vähentää paletin kappalelaskurin arvoa yhdellä, eli laskuri siirtyy osoittamaan edellistä kappaletta paletilla ja

TESTP tarkistaa osoittaako paletin kappalelaskuri haluttuun kappaleeseen. Jos osoittaa, siirrytään annettuun osoitteeseen, muuten jatketaan ohjelman suoritusta.

koordinaatisto

Version 4.0 komennolla REGION, on mahdollista määritellä robotin työskentelyalueelle uusia koordinaatistoja. Tätä ominaisuutta voidaan käyttää hyödyksi esimerkiksi CAD:ltä saatavien tietojen siirrossa ohjausohjelmaan ja työskentelypisteiden kohdistuksessa.

Erilaisia DI/DO-porttien käyttöön liittyviä komentoja ovat WRITEO, TESTI ja WAITI. Laskurit esitellään COUNTER-määrittelyllä. Laskureihin liittyviä komentoja ovat CSTATUS, DECR, INCR, SETC, TESTC ja COMPC, joista viimeksi mainitulla verrataan kahta laskuria vertailuoperaatioilla <, <=, >, >=, = ja <>.

Robotin ohjauksessa tarvittavien komentojen lisäksi AML/Entry tarjoaa joukon ohjelmointivaiheessa tärkeiksi osoittautuvia editointikomentoja, joita tässä ei kuitenkaan tarkastella lähemmin. Lopuksi esimerkkinä lyhyt komentojen WHERE ja GUARDI käyttöä selventävä ohjelma [IBM84b]:

```
MAIN:  SUBR
  CARD_POINT: NEW PT(0,450,0,0);
  CARD_POINT2: NEW PT(0,350,0,0);
  STOP_POINT: NEW 5;              -- the DI point to guard
  C: STATIC COUNTER;
  NEW_PLACE: NEW PT(0,0,0,0);

  PMOVE(CARD_POINT);              -- move to the start point
  GUARDI(STOP_POINT);             -- guard for the stop point
  PAYLOAD(11);
  LINEAR(1);
  PMOVE(CARD_POINT2);             -- move to the new point
  MSTATUS(C);                     -- query the motion complete code
  COMPC(C<>0,HERE);               -- if stopped by guard
                                  -- send point to host
  NOGUARD;                        -- disable motion guard
  LINEAR(5);
  PAYLOAD(5);
  BRANCH(EN);
HERE: WHERE(NEW_PLACE);           -- read the stop location
  PUT(NEW_PLACE);                 -- send location to host
EN:
END;
```

Lähteet: [GSCT84], [Loz83], [PL83], [TSM82], [IBM84a] ja [IBM84b].

4.2.2 AML/2

AML/2 on uudempi AML-kielen pelkistetty versio kuin AML/Entry. Pelkistystä ei ole enää viety yhtä pitkälle, joten kieli on huomattavasti käyttökelpoisempi kuin AML/Entry. Kieltä käytetään IBM:n symmetristen scara-robottien ohjauksessa. Kieli sisältää normaalit aritmeettiset operaattorit +, -, * ja /sekä fumktiot `abs`, `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `atan2`, `sqrt`, `mod`, `min` ja `max`. Käyttäjä voi myös määritellä omia fuktioita. Ohjauslauseet ovat tyypilliset `IF ... THEN ... ELSE ...` , `WHILE ... DO ...` , `REPEAT ... UNTIL ...` -lauseet sekä `BEGIN ... END` -lohko. Tietotyypit ovat `INT`, `LONG_INT`, `REAL`, `LONG_REAL`, `BOOLEAN` ja `STRING`, joka voi olla joko tavanomainen merkeistä (`CHAR`) koostuva tai bittijono. Bittijonojen käsittelyä varten on loogiset operaattorit `AND`, `OR`, `XOR`, `NOT`, `ROT`, `SHIFT` ja `ASHIFT`. Merkkijonojen käsittelyä varten on omat rutiininsa.

scara

aliohjelmat

Nopeuden säätöä varten on komento `ACCEL`, jolla voidaan valita liikkeen nopeus portaattomasti 0 – 100% maksiminopeudesta. Liikkuminen voi tapahtua lineaarisesti tai ympyrärataa pitkin.

Lähde: [AML86].

4.3 ANORAD

NC

ANORAD on Anorad Corporationin kehittämä laajennettu NC-kieli. Ohjelmoitavien matemaattisten lausekkeiden, muuttujien, hyppyjen ja aliohjelmien ansiosta kieli on luettavissa perusliikkeiden tason kieliin. Kaupallista versiota kielestä käytetään Anomatic roboteissa.

NC-kielille ominaisesti ANORAD-kielen komennot ovat lyhyitä ja epähavainnollisia. Esimerkkiksi suoraviivaisen liikkeen komento on `G1`. Kontrollirakenteiden vähäisyys heikentää myös ANORAD-kieltä.

Kieli tarjoaa suppeahkot liitäntämahdollisuudet ympäryslaitteisiin. Antureista vain kosketusantureiden käyttö on mahdollista. Samoin mahdollisten työkalujen määrä on pieni[9]. Rinnakkaista prosessointia ei ole.

Lähde: [BS82].

4.4 ARL

Hitachi

Viime vuosina on markkinoille ilmestynyt joukko japanilaisia robottien ohjelmointikieliä. Näistä Hitachin roboteissa käytettävä ARL on tyypillinen perusliikkeiden tasoon lukeutuva kieli. Rakenteeltaan Pascal-kieltä muistuttavalla ARL-kielellä voidaan määritellä useita eri tasoisia robotin ohjausmenetelmiä. Robottia voidaan ohjata tahdistetusti oheislaitteiden kanssa. Myös antureiden käyttö on mahdollista. ARL-kielessä voidaan käsitellä kaksitasoista (`on/off`) ja 8 bitin tarkkuudella analogista tietoa. Pascal-kielen valinta ARL:n perustaksi johtui sen suosiosta ja suhteellisen helposta kieliopista. Lisäksi kielen lohkorakenne sopii hyvin robottikieleen.

Pascal

Liikekomentona on tyypillinen `MOVE`-komento, johon voidaan liittää tarkennuksena `S` ilmaisemaan nopeutta sekä `H` ilmaisemaan käsiteltävää tarttujaa. Liike usean, käsiohjaimella määritetyn, pisteen kautta toteutetaan komennolla `PERFORM`.

Ehdollisia rakenteita ovat muun muassa `IF ... THEN ... ELSE ...` ja `WAIT ... ON`-lauseet. Komentoja `ENABLE` ja `DISABLE` käytetään tarkkailtaessa tiettyjen ehtojen toteutumista robotin liikkeiden aikana.

ARL-kieli on hyvin joustava ja siihen on helppo tehdä lisäyksiä Pascal-kielen kaltaisilla proseduureilla.

Lähde: [Jap84].

[9] lähinnä ruuvain

4.5 ARLA

ARLA (ASEA Robot Language) kehitettiin alunperin ohjelmointilaitteella ohjelmoitavaksi kieleksi. Kehityksen myötä joutui myös ASEA tuomaan markkinoille oman erillään ohjelmointijärjestelmänsä. Kieli pysyi kuitenkin lähes muuttumattomana, jolloin aiemmin ohjelmointilaitteella ohjelmoimaan oppinut käyttäjä voi helposti siirtyä järjestelmän käyttäjäksi. *ohjelmointilaite*

Ohjelma sisältää pääohjelman sekä joukon aliohjelmia [10]. Aliohjelmat numeroidaan ja kutsutaan numeron perusteella. Aliohjelma voi myös kutsua toista aliohjelmaa. Korkeintaan kolmea aliohjelmaa voidaan kutsua sisäkkäin ja kutsutusta aliohjelmasta on palattava sitä kutsuneeseen aliohjelmaan. *aliohjelmat*

Valmis ja testattu ohjelma tallennetaan tavallisesti levykkeelle, josta se tarvittaessa ladataan ohjausyksikön muistiin. Ohjelma voidaan ladata levykkeeltä käyttämällä AUTOGET-komentoa. Paloittelemalla ohjelma sopiviin osiin sekä lataamalla ja ajamalla osat halutussa järjestyksessä, voidaan käyttää robotin muistia huomattavasti suurempia ohjelmia. *muisti*

ASEAn robottien liikkeet pisteestä pisteeseen ohjelmoidaan käyttämällä opetusmenetelmää. Paikoituskäskyä ohjelmoitaessa valitaan ensiksi koordinaattijärjestelmä ja liikutetaan robotti haluttuun paikkaan ohjaussauvalla. Paikka-näppäintä painamalla talletetaan karkea paikka. Tämän jälkeen voidaan valita paikoitusnopeus (V%), joka ilmaistaan %:na perusnopeudesta. Paikoitustarkkuus, FINE, argumenttia käytetään ohjelmoitaessa robotin paikkoja tarkasti. Valittavana on kaksi tarkkuutta: joko suuri tai pieni nollavyöhyke. *opetus*

Vaaputusliike voidaan ohjelmoida WEAVE-komennolla. Vaaputuksen kuvio määritetään erityisessä aliohjelmassa, jonka numero sisältyy WEAVE-komennon argumenttiin. Myös ympyräliike voidaan ohjelmoida antamalla paikkakomennolle CIRCLE-argumentti. Haluttaessa suorittaa liike samanlaisena toistuvasti suhteessa tiettyyn pisteeseen, käytetään viitepisteargumenttia REFP.

Robotin ohjelmoitu paikka voidaan tallettaa myös rekisteriin, jonka numero sisältyy käskyn STO LOC argumentteihin. Siirtyminen talletettuun paikkaan tapahtuu käskyllä POS LOC. Edellisen paikoituskäskyn argumentit voidaan sijoittaa käsiteltävälle paikoituskäskylle SAME-käskyllä.

Esimerkkinä paikoituskäskyistä esitetään käskyrivit, joilla robotti siirretään paikkaan A suuren nollavyöhykkeen tarkkuudella sekä rekisteriin 5 talletettuun paikkaan lisättynä x-akselin suuntaisella, 5 mm siirroksella.

```
40  POS V% = 60% FINE L
70  POS V% = 60% LOCATION 5 OFFSET X = 5
```

Ohjauskäskyt voidaan jakaa kolmeen ryhmään: liikkeenhallintakäskyt, ympäryslaitteiden ohjauskäskyt ja ohjelmanhallintakäskyt. Liikkeenhallintakäskyjä ovat perus- ja maksiminopeuksien määrityksessä käytettävä

[10] enintään 999 tai 9999 järjestelmästä riippuen

VELOCITY-, työkalupisteen kutsumisessa käytettävä TCP-, koordinaattijärjestelmän valinnassa käytettävä COORD sekä viitekehyskäsky FRAME.

TCP

Järjestelmässä voidaan määritellä käsikäyttöisesti 9 eri työkalupisteen koordinaatit. Oletusarvoisesti (TCP = 0) työkalupiste on tartuntaelimen asennuslaipan keskellä, sen etupinnalla. COORD-käskyllä voidaan ohjelman suorituksen aikana vaihtaa käytettävä koordinaattijärjestelmä. FRAME-käskyllä voidaan ohjelmassa siirtää suorakulmainen jalustaan sidottu koordinaatisto minne tahansa työalueella.

Ympäryslaitteiden ohjauksessa käytetään käskyjä GRIPPER ja OUTPUT. GRIPPER-käskyä käytetään tarttujan ja hitsauspihtien hallinnassa. Samalla käskyllä huolehditaan tartuntaelimen sulkemisesta ja avaamisesta, tulkinta riippuu edeltävästä tilasta. OUTPUT-käskyä käytetään robottijärjestelmän digitaalisiin lähtöihin yhdistettyjen ympäryslaitteiden ohjauksessa. Käskyn yhteydessä määritellään ohjaussignaali, joita on neljä eri vaihtoehtoa: SET, RESET, INVERT (vaihda) ja PULSE.

Ohjelmanhallintakäskyillä ohjataan käskyjen, aliohjelmien ja ohjelmien suoritusjärjestystä. Käskyjen suoritusjärjestystä ohjataan käskyillä WAIT, JUMP, REG, INTERRUPT ja RETURN. WAIT-käsky viivyttää seuraavan käskyn suorittamista. JUMP on hyppykäsky. Hyppy voi olla ehdoton tai ehdollinen. Ehdottomassa hypyssä argumenttina annetaan seuraavaksi suoritettavan käskyn numero. Ehdollisessa hypyssä testataan joko annetun tulon tila tai rekisterin arvo. Rekisterin arvoa muutetaan REG-käskyllä, joko korvaamalla rekisterissä oleva arvo uudella tai kasvattamalla tai pienentämällä olemassa olevaa arvoa. INTERRUPT-käskyllä voidaan keskeyttää suoritettava käsky, suoritettava ohjelma tai kutsua aliohjelmia 1 - 5 (vain näitä). Varsinainen aliohjelmien kutsukäsky on kuitenkin CALL, jonka argumenttina on aliohjelman numero. Pääohjelman ja jokaisen aliohjelman päättymisen merkkinä käytetään käskyä RETURN. Ohjelmien suoritusjärjestystä voidaan ohjata käskyllä GET B. Käskyllä ladataan kalvolevyltä robotin muistiin valitut ohjelmat ajoa varten.

Seuraavassa esimerkissä robotti poraa reikiä työkappaleeseen. Uuden työkappaleen odotusaikana robotti siirtyy odotusasemaan.

```
------------
Paaohjelma
------------
10   V = 500 MM/S MAX = 500 MM/S   Perus ja maksiminopeus
20   TCP 7                          TKP poran karjessa
30   ROBOT COORD                   Nivelkoordinatisto
40   POS V = 100%                   Karkeapaikoitus porausp.
50   CALL PROG 11 R11               Kuvioaliohjelman kutsu,
                                          sisaltaa porauspaikat
60   RECT COORD                     Suorakulm. koordinaatisto
70   CALL PROG 12                   Porausaliohjelman kutsu
80   ROBOT COORD                    Siirtyminen nivelkoord.
90   JUMP TO 10 IF R11 0            Onko kaikki reiat porattu?
100  POS V = 100% FINE              Siirtyminen odotusasemaan
110  WAIT UNTIL INP 1 = 1           Uusi tyokappale syotetaan,
120  RETURN                         tulo asetetaan manuaalisesti.

------------
Kuvioaliohjelma PROG 11
------------
10   POS V = 100% FINE              Tarkka paikoitus reikaan 1
20   RETURN                         1. kuviomoduli
30   POS V = 100% FINE              Tarkka paikoitus reikaan 2
40   RETURN                         2. kuviomoduli
50   POS V = 100% FINE              Tarkka paikoitus reikaan 3
60   SET R11 0                      Kuviomodulilaskuri asetetaan
                                          alkuperaiseen arvoonsa
70   RETURN                         3. kuviomoduli

------------
Porausaliohjelma PROG 12
------------

5    POS V = 25% REFP ON             Kyseinen robotin paikoitus
                                          tulee viitepisteeksi
                                          porausta suotitettaessa
10   SET OUTPT 1                     Aloita poraus
20   POS V = 25% FINE                Tarkka paikoitus loppupist.
30   POS V = 100%                    Ylos porauksen jalkeen
                                          viitepistetta varten
40   RESET OUTPT 1                   Poraus lopetetaan
45   POS V = 100% REFP OFF           Viitepisteen vaikutus lakkaa
50   RETURN                          Paluu paaohjelmaan.
------------
```

Edellä on käsitelty ohjelmointia ohjelmointilaitteella. ASEAn erillään ohjelmoinnissa käytettävä kieli eroaa vain vähän edellä esitetystä. Luonnollisesti uusien robottityyppien (heiluri- ja scara-robotit) mukana tulee uusia piirteitä kieleen. Tärkein ero on tietysti paikoituskäskyissä. Ohjelmointilaitteella ohjelmoitaessa jokainen piste opetetaan robotille. Tämä ei luonnollisesti onnistu erillään ohjelmoinnissa, jossa pisteet annetaankin jalustaan kiinnitetyssä suorakulmaisessa koordinaatistossa.

Lähteet: [BR85] ja [Toi84].

4.6 AUTOPASS

AUTOPASS (**Auto**mated **P**arts **A**ssembly **S**ystem) -kieli kehitettiin IBM Thomas J. Watson tutkimuskeskuksessa. Tutkimuskäyttöön tarkoitettu, lähinnä tehtävänläheisiin kieliin lukeutuva kieli julkistettiin vuonna 1977. AUTOPASS-kielen perustana on PL/1-kieli ja se tarjoaakin useita PL/1:n ohjauksellisia ja tiedonkäsittelyllisiä piirteitä. AUTOPASS:n englanninkielisiä lauseita muistuttavat lauseet voidaan jakaa kahteen luokkaan: kokoonpanoon liittyvät ja muut lauseet. Kokoonpanoon liittyvät lauseet voidaan edelleen jakaa kolmeen ryhmään: aseman muutokseen, työkalujen hallintaan ja komponenttien kiinnityksestä huolehtivien osien käsittelyyn liittyvät lauseet. Muita lauseita ovat esimerkiksi kontrollilauseet, maailman mallin luomisessa tarvittavien geometristen muuttujien määrittelyt ja lukuarvojen sijoittaminen niihin sekä järjestelmän toiminnan tarkistuksiin liittyvät lauseet. *PL/1*

Aseman muutoksista huolehtivilla lauseilla voidaan kuvata erilaisia kokoonpanotoimenpiteitä, kuten komponenttien noutoa ja järjestelyä sekä manipulaattorin liikkeitä. Yksi monipuolisimmista tähän ryhmään kuuluvista lauseista on PLACE-lause, jonka syntaksi on:

```
PLACE <objekti1> <toimenpiteen laatu> <objekti2> <tartuntatilanne>
    <lopputilanne> <rajoitukset> <then-hold>.
```

Toimenpiteen laatua kuvataan IN- ja ON-komennoilla, jotka ilmaisevat esimerkiksi komponentin siirron luonnetta. Komento ON kertoo kohteen olevan avoin. Kappale siirretään avoimelle pöydälle. Vastaavasti IN kertoo kappaleen asetettavan esimerkiksi toisen kappaleen sisään. Muita aseman muutoksissa käytettäviä lauseita ovat INSERT, LIFT, PUSH, MOVE, GRASP, RELEASE, EXTRACT, LOWER, SLIDE, ORIENT ja TURN.

Työkalujen käsittelyssä käytetään lauseita: OPERATE, CLAMP, UNCLAMP, LOAD, UNLOAD, FETCH, REPLACE, SWITCH, LOCK ja UNLOCK. Esimerkkinä jälleen ryhmän monipuolisimman lauseen syntaksi:

```
OPERATE <tyokalu> <lisalaitteet> <tyoskentelypiste>
          <kiinnitys> <lopputila>
          <tyokalun toimintaparametrit> <then-hold>.
```

<Then-hold>, joka esiintyi myös PLACE-lauseessa ilmaisee manipulaattorin asentoa toimenpiteen jälkeen.

Komponenttien kiinnittämisessä käytettäviä lauseita ovat: ATTACH, DRIVE IN, RIVET, FASTEN ja UNFASTEN. Yleisen kiinnityslauseen, ATTACH-lauseen, syntaksi on seuraavanlainen:

```
<kiinnitin> <kaksiosainen kiinnitys>
          TO <kiinnityskohde> <kiinnityspuoli>
          <kiinnitys> <lopputila> <kaytettava tyokalu>.
```

Kaksiosaisella kiinnityksellä tarkoitetaan kiinnityslaitetta, jossa on kaksi erillistä komponentteja kiinnitettäessä yhdistettävää osaa. Esimerkiksi ruuvi ja mutteri. Jokaisen edellä esitetyn ryhmän lauseen loppuun voidaan liittää käytettävän manipulaattorin määrittelevä komento WITH <hand-name>.

Automaattisen kokoonpanon onnistuminen edellyttää jatkuvaa operaatioiden seurantaa. Useissa lauseissa on jo valmiina tarkistusoperaatio, mutta tämä on useimmiten riittämätön. AUTOPASS -kielen lauseella VERIFY voidaan suorittaa virheen testausta ja vaadittavia toipumisrutiineja. VERIFY-lauseen syntaksi on:

```
VERIFY <tarkistuslista> <toimintolista>.
```

Muita tähän luokkaan kuuluvia lauseita ovat muun muassa OPEN STATE OF, CLOSED STATE OF ja NAME.

AUTOPASS-kieli tarjoaa mahdollisuudet myös usean robotin samanaikaiselle työskentelylle lauseella IN PARALLEL DO. Ohjelmoijan on kuitenkin huolehdittava törmäyksenestosta.

Esimerkkinä ohjelmoinnista AUTOPASS-kielellä on seuraavassa lyhyt ruuvinkiinnitysohjelma. Ohjelman lauseet muistuttavat englanninkielisiä lauseita, joten englanninkielentaitoiselle sen tulkitseminen on helppoa, kunhan selvittää nimien merkityksen [Loz83]:

```
1. OPERATE nutfeeder WITH car-ret-tab-nut AT fixture.nest
2. PLACE bracket.hole IN fixture SUCH THAT bracket.bottom
       CONTACTS car-ret-tab-nut.top
       AND bracket.hole IS ALIGNED WITH fixture.nest
3. PLACE interlock ON bracket SUCH THAT
       interlock.hole IS ALIGNED WITH bracket.hole
       AND interlock.base CONTACTS bracket.top
4. DRIVE IN car-ret-intlk-stud INTO car-ret-tab-nut
       AT interlock.hole
       SUCH THAT TORQUE IS EQ 12.0 IN-LBS USING air-driver
       ATTACHING bracket AND interlock
5. NAME bracket interlock car-ret-intlk-stud car-ret-tab-nut
       ASSEMBLY support-bracket
```

Lähteet: [BS82], [Loz83] ja [LW77].

4.7 BAPS

Bosch
IRDATA

BAPS-kieltä käytetään Boschin scara-roboteissa. Kieli on rakenteinen ja se käännetään edelleen IRDATA-koodiksi [Ger85]. Kielen kehittely jatkuu eikä kaikkia manuaalissa [Bos86] mainuttuja piirteitä ole vielä toteutettu[11]. Kieli sisältää tietotyypit piste[12] `POINT`, desimaaliluku `DEC`, kokonaisluku `INTEGER`, teksti `TEXT`, merkki `CHARA` ja looginen[13] `BINARY`, joita kaikkia voidaan käyttää myös taulukkoina `ARRAY`.

Ohjausrakenteet ovat tyypilliset `IF ... THEN ... ELSE ... , JUMP osoite, REPEAT n TIMES ... REPEAT_END` sekä aliohjelma `SUBROUTINE nimi ... RETURN-` — ja pääohjelma `PROGRAM nimi ... END` -rakenteet. Pääohjelmat voivat kutsua toisiaan ja aliohjelmia. Tiedot pääohjelmista toisiin välitetään `EXTERNAL`- määrittelyjen avulla, koska parametreja ei ole. Muita ohjauskomentoja ovat `WAIT`, `WAIT UNTIL`, `PAUSE` ja `HALT`.

Lausekkeet voivat sisältää tavalliset aritmeettiset operaattorit +, -, * ja / sekä funktiot `SIN`, `COS`, `ATAN`, `SQRT` ja `MOD`. Loogisia muuttujia varten ovat operaattorit `AND`, `OR` ja `NOT`. Lisäksi kieli sisältää tavanomaiset vertailuoperaattorit.

Liikekomentoja on kaksi, absoluuttinen `MOVE` ja suhteellinen `MOVE_REL`, joihin molempiin voidaan liittää useita tarkenteita. Käytettävissä on vapaa, suoraviivainen ja ympyräliike sekä synkronoitu[14] liike. Yhdellä liikekäskyllä voidaan käydä läpi useita pisteitä. Liikkeen nopeus `V` ja kiihtyvyys `A` ovat asetettavissa samoin aika `T`.

Esimerkkinä kielestä esitetään manuaalin [Bos86] `DEMO`-ohjelma, jolla esine viedään aloituspisteestä `POINT_A` lopetuspisteeseen `POINT_B`. Aloitus- ja lopetuspisteet annetaan opettamalla.

Lähde: [Bos86].

[11] esimerkiksi ympyräliike
[12] x,y,z ja kulma
[13] totuusarvot 0 ja 1
[14] esimerkiksi kuljettimeen

```
PROGRAM DEMO

      OUTPUT: 1 = VALVE

      UP = (0, 0, 100, 0)
      DOWN = (0, 0, -100, 0)

      OPEN
      MOVE TO POINT_A
      MOVE_REL DOWN
      CLOSE
      MOVE_REL UP

      MOVE TO POINT_B
      MOVE_REL DOWN
      OPEN
      MOVE_REL UP
      HALT

      SUBROUTINE OPEN
            VALVE = 1
            WAIT 1
      RETURN

      SUBROUTINE CLOSE
            VALVE = 0
            WAIT 2
      RETURN

END
```

4.8 EMILY

EMILY on IBM kehittämä off-line ohjelmointikieli. Lähinnä perusliikkeiden tai rakenteellisten kielien tasoon kuuluva EMILY-kieltä voidaan pitää ML-kielen parannettuna versiona. Tutkimuskäytössä EMILY-kielellä ohjattiin IBM-robotteja, ohjelmointi tapahtui IBM 370/145 ja System 7 tietokoneilla. *ML*

Syntaksiltaan Fortran-kieltä muistuttavan EMILY-kielen tietotyypit ovat kokonaisluku ja -taulukko. Kontrollirakenteita ovat `BRANCH`, `LOOPFOR ... ENDFOR` ja `BRCOND`. Robottien liikkeitä voitiin ohjata myös nivelkohtaisesti. Liikekomennoista mainittakoon `MOVE` ja `JOINT MOTION`-komennot. *Fortran*

Antureiden käytössä tarvittavia komentoja on paljon. Myös kahden robotin rinnakkainen prosessointi on mahdollista. Merkittävänä puutteena voidaan toisaalta pitää oheislaitteiden ja työkalujen käsittelyssä tarvittavien komentojen puuttumista. *anturit*

Lähde: [BS82].

4.9 FUNKY

Myös FUNKY on IBM:n kehittämä robottien ohjauskieli. Pisteestä pisteeseen tason kieliin kuuluvaa FUNKY-kieltä käytettiin IBM-robottien ohjauksessa. Ohjelmointi suoritettiin opetusmenetelmällä kuljettamalla robottia tarvittavien pisteiden kautta ja tallettamalla halutut toiminnot muistiin funktionäppäimillä. Järjestelmän tietokoneena oli IBM System 7.

Kielessä käytettäviä tietotyyppejä olivat kokonaisluvut, vektorit ja skalaarit. Kontrollirakenteita ei FUNKY:ssa, kuten ei muissakaan tämän tasoisissa kielissä ole. Myös antureiden ja ympäryslaitteiden käyttömahdollisuudet ovat olemattomat. Tavallisesti vain tarttujassa käytettiin tartunnan keskittämisessä tarvittavia antureita. Työkaluna voitiin käyttää ruuvainta.

Lähde: [BS82].

4.10 HELP

Rakenteellisten kielien ryhmään kuuluvaa HELP-kieltä markkinoidaan General Electric Company'n valmistamien A12 Allegro Robot System -robottien ohjauskielenä. Perustuen lähinnä Pascal-ja Fortran-kieliin sisältää HELP näille tyypillisen proseduurien käyttöön nojautuvan rakenteen.

Käytettävät tiedot voivat olla tyypiltään reaalilukuja, taulukoita tai ASCII-merkkijonoja. Geometrisia tietotyyppejä tai rakenteita ei HELP sisällä, vaikka robotin liikkeen ohjaus tapahtuu xyz-koordinaatistossa käyttämällä nivelkohtaisia komentoja. HELP tarjoaa erinomaiset mahdollisuudet useamman robotin samanaikaiselle, synkronoidulle työskentelylle. Jopa 12 vapausasteen rinnakkainen prosessointi on mahdollista. Kieli sisältää tavanomaiset ohjausrakenteet kuten `WHILE ... DO ...`, `IF ... THEN ... ELSE ...` ja `FOR ... TO ... DO ...`-lause.

Ympäryslaitteiden ja antureiden kanssa kommunikointiin on joukko tämän tason ohjauskielille tyypillisiä komentoja. Lähinnä kosketusantureihin liittyviä komentoja ovat `IF TESTP`, `TEST` ja `HIGH`. Ympäryslaitteita varten on muun muassa komennot `SET`, `RESET`, `PULSE` ja `STROKE`. Näköjärjestelmän käyttömahdollisuutta ei ole, eikä myöskään työkalujen käsittelyyn tarkoitettuja komentoja.

Lähteet: [BS82], [GSCT84] ja [Loz83].

4.11 IRDATA

IRDATA (**I**ndustry **R**obot **Data**) -välikieli on ehdotus kansainväliseksi standardiksi[15], jolla pyritään helpottamaan ohjelmien tuottamista ja ajamista erilaisilla roboteilla. Kieltä on kehitetty erityisesti Saksan liittotasavallassa ja sitä käytetään ainakin Bosch-robottien välikielenä [Bos86].

Bosch

Lähteet: [WN84] ja [Ger85].

[15]ISO/TC 194 N 37 (1985)

4.12 IRL

IRL (Intuitive Robot Language) on Sveitsiläisen Microbo yhtiön kehittämä robottien ohjauskieli. Yhtiö on erikoistunut erittäin tarkkoihin, alunperin kellojen valmistuksessa käytettäviin robotteihin. Viime vuosina yhtiö on pyrkinyt ilmeisistä syistä laajentamaan huipputarkkojen robottien sovellutusaluetta.

Lähinnä pisteestä pisteeseen tai perusliikkeiden tason kieliin lukeutuva IRL perustuu numeerisen ohjauksen (NC) kieliin. Näin ollen se omaa melko vähäiset ohjelmointiominaisuudet. Tietotyyppejä on tavallista vähemmän. Käytettävissä on kokonaislukutyyppiset muuttujat ja lukujen vertailuoperaatiot. Taulukoita ja reaalilukuja ei ole.

Aritmeettiset operaatiot ja vertailut kuuluvat IRL-kieleen. Mutta matemaattisia funktioita ei ole. Myös kontrollirakenteiden määrä on suppea, haarautuminen annettuun osoitteeseen on kuitenkin mahdollista. Rinnakkaista prosessointia ei IRL-kielessä ole.

Lähde: [PL83].

4.13 JARS

JARS kehitettiin tutkimustarkoituksia varten NASA:n laboratoriossa[16] vuonna 1979. Tavoitteena oli kehittää korkeatasoinen ohjelmointiympäristö, jota voitaisiin käyttää robotin manipuloinnin, ohjauksen ja ohjelmointijärjestelmien tutkimuksessa. Kieltä käytetään pääasiassa JPL:ssa. *JPL*

JARS perustuu Pascal-kieleen, johon on lisätty monia robotin ohjauksen vaatimia ominaisuuksia. Se sisältää kaikki Pascal-kielen tietotyypit, muun muassa tavallista monipuolisemmat geometriset tietotyypit. Kielessä on mahdollista määritellä kohteeseen liittyvän koordinaatiston paikka ja asento. *Pascal*

Robotin liike ja asento on mahdollista esittää nivelkulmien, vektorien, tarttujan siirtymän ja kierron avulla. Liike voidaan suorittaa pisteohjatusti, jolloin liike päätepisteiden välillä tapahtuu vapaasti robotin rakenteille ominaisella tavalla. Myös suoraviivainen liike päätepisteiden välillä on mahdollista. Liike voidaan myös paloitella, jolloin tarttuja kulkee usean ennakolta määrätyn pisteen kautta.

Näköjärjestelmän ja lähestymisantureiden (rajakytkimien) käytö on mahdollista.

Puutteena voidaan pitää JARS-kielen kykenemättömyyttä rinnakkaiseen prosessointiin. Lisäksi kielessä on mahdollista käytää vain kahta binääristä ulostuloa.

Lähde: [GSCT84].

[16] Jet Propulsion Laboratory (JPL)

4.14 JOB

Hitachi BASIC

JOB-kieli on Hitachin roboteissa käytettävä BASIC-pohjainen ohjelmointikieli. Kieli sisältää kaikkiaan 28 käskyä tai niihin rinnastettavaa kielen elementtiä. Aritmetiikka tapahtuu pinon avulla joidenkin taskulaskinten tai assemblerien tapaan. Tehtävät voidaan rakentaa ohjelmista (`PROGRAM`) ja töistä (`JOB`), joita molempia voi olla korkeintaan sata kappaletta[17]. Työt voivat koostu ohjelmista ja töistä. Sen sijaan ohjelmat eivät voi sisältää töitä. Ohjelma askelia voi olla nykyisessä versiossa korkeintaan 2000, joista yhdessä työssä korkeintaan 1000.

Ohjausrakenteita ovat muun muassa `IF ... THEN ... ELSE ...` , `DO n, DO WHILE, BEGIN ... END` ja työn kutsu `CALL n` sekä ohjelman kutsu `PERFORM n`. Kieli sisältää tavallisimmat aritmeettiset ja loogiset operaatiot.

Lähteet: [Hita] ja [Hitb].

4.15 LAMA-S

APL

LAMA-S on ranskalaisen IRIA-tutkimuskeskuksen kehittämä, VAL-kieltä muistuttava robottikieli. Alunperin kieli kehitettiin erityisesti vammaisten apuna toimivien robottien ohjaukseen. LAMA-S-kielen toteutuksessa käytettiin APL-kieltä.

rinnakkaisuus

Eri koordinaatistoissa toteutettavien liikekomentojen lisäksi LAMA-S sisältää mm. mahdollisuuden rinnakkaiseen prosessointiin. Suoritusjärjestyksestä huolehditaan erityisellä lohkorakenteella siten, että esimerkiksi rinnakkaislohkossa olevat käskyt suoritetaan samanaikaisesti. Myös tavanomaiset ohjausrakenteet sisältyvät LAMA-S kieleen.

LAMA-S kielinen ohjelma käännetään suoritusta varten PRIMA-kieliseksi ohjelmaksi.

Lähteet: [Loz83] ja [GG85].

[17] n = 0-99

4.16 LM

Hyvin paljon AL-kieltä muistuttava LM (Langage de Manipulation) ohjauskieli kehitettiin IMAG:n robotiikan laboratoriossa Grenoblessa vuonna 1979, missä sitä käytettiin ROBITRONin ja BARRAS-Provencen valmistamien robottien ohjauksessa. Ohjelmoinnissa ja ohjelmien ajossa käytettiin DEC LSI 11/2-tietokonetta. Teollisessa sovellutuksessa LM-kieltä käytetään ranskalaisen SCEMI yhtiön robottien ohjauksessa.

Rakenteellisten kielien tasoon lukeutuvalla LM-kielellä kokoonpano kuvataan sekä robotin tarttumaelimen toimintojen että antureista saatavan tiedon avulla.

LM-kielellä on muun muassa seuraavia ominaisuuksia:

- usean robotin samanaikainen ohjaus,
- robotin ympäristön kuvaus karteesisilla koordinaatistoilla,
- robottien liikkeiden määritys liikuteltavaan kappaleeseen kiinnitetyn koordinaatiston perusteella,
- useita liikeradan toteutusperiaatteita,
- toiminnan keskeytys ennalta määrätyissä pisteissä,
- robotin liikkeiden ja tarttujan toiminnan ohjaus antureista saatavan tiedon avulla,
- hyvinkin erilaisten antureiden käyttömahdollisuus ja
- robotin toiminnan koordinointi ympäryslaitteiden kanssa.

Kuten LM-kielen kehittelyssä perustana käytetty Pascal-kieli muodostuu LM-kielinen ohjelma pääohjelmasta ja proseduureista. LM-kielessä on myös monia Pascal-kielestä kopioituja piirteitä, kuten symboliset muuttujat, taulukot, tietotyypit (kokonaisluvut, reaaliluvut ja totuusarvot), kontrollikäskyt (`IF`, `WHILE`, `GOTO`), proseduurit, ja I/O-käskyt. Lisäksi LM sisältää useita erityisesti robottien ohjauksessa tarvittavia ominaisuuksia, joita tarkastellaan seuraavassa hieman lähemmin.

Robotin ympäristöä kuvataan karteesisilla koordinaatistoilla. Käytettävät koordinaatistot määritellään järjestelmän kattavalla referenssikoordinaatistolla `STATION`, joka kalibroidaan tarvittaessa. Koordinaatistojen määrittely tapahtuu siirtämällä ja kiertämällä jo määriteltyjä koordinaatistoja. Tätä varten on LM-kielessä käytettävissä geometriset tietotyypit `vector`, `transform` ja `frame`, jotka muistuttavat AL-kielessä määriteltyjä vastaavia tietotyyppejä. Lisäksi LM-kielessä on määritelty yksikkövektorit `VX`, `VY` ja `VZ`.

LM-kielen monipuolisista toiminnoista mainittakoon vektorin määrittely `VECTOR (x,y,z)`, vektorien yhteenlasku, vektorin pituuden määritys `LENGTH(v)`, kierto x radiaania vektorin v määrittämän akselin ympäri

ROTATION(v,x), kahden muunnoksen yhdistäminen (t1*t2), muunnoksen soveltaminen koordinaatistoon (f*t) sekä kahden koordinaatiston välisen muunnoksen ratkaisu TRANSF(f1,f2).

Tavanomaisten Pascal-kielestä tuttujen symbolisten muuttujien lisäksi LM-kielisissä ohjelmissa käytetään "tilamuuttujia"[18], joiden sisällöstä huolehtii tulkki. Ohjelmoija ei tilamuuttujien arvoon voi vaikuttaa ja kukin tilamuuttuja on määriteltävä toteutuskohtaisesti. Seuraavassa esitetään esimerkkinä joitakin IMAG:n robotiikan laboratoriossa käytettyjä tilamuuttujia.

- ROBOT(i) /i=1,2,.../: i:nteen käsivarteen kiinnitetyn koordinaatiston sisältävä muuttuja.
- MOVE.TIME(i): i.nnen robotin viimeisimmän liikesarjan päättymisestä kulunut aika.
- MOVING(i): Looginen muuttuja, jonka arvo on TRUE i:nnen robotin liikkuessa ja muulloin FALSE.
- FX(i), FY(i) ja FZ(i): reaaliarvoisia muuttujia, jotka mittaavat voiman komponentteja ROBOT(i)-koordinaatistossa.

Kahden koordinaatiston f1 ja f2 liittäminen voidaan toteuttaa ATTACH-lauseella seuraavasti: ATTACH f1 WITH f2. Tämä on toiminnaltaa samanlainen kuin AL-kielen AFFIX-lause. Muutokset toisen koordinaatiston asennossa päivittävät automaattisesti toisen koordinaatiston asennon siten, että suhde koordinaatistojen välillä säilyy muuttumattomana. Kahta tilamuuttujaan liittyvää koordinaatistoa ei voi liittää yhteen. Liitos purkautuu lauseella: DETACH f1 AND f2.

Robotin liikekomento LM-kielessä on syntaksiltaan seuraavanlainen: MOVE f <rata> I UNTIL b I I / I. Komennolla siirretään koordinaatistoa f <rata>:ssa kuvattua rataa pitkin kunnes ehdollinen lause UNTIL b on tosi.

Mikäli liikekomento päättyy symboliin "/", odottaa tulkki liikkeen päättymistä ennen seuraavan lauseen suorittamista. Muussa tapauksessa aloitetaan seuraavan lauseen suoritus heti liikkeen alkaessa. Mikäli myös seuraava lause sisältää liikekomennon, odottaa tulkki edellisen liikkeen päättymistä.

Termille <rata> on olemassa esimerkiksi seuraavanlaisia esitystapoja:

- TO dest ilmaisee siirtymän määränpään, väliasemat määräytyvät interpoloimalla,
- TO dest VIA int1, int2, ..., intp rata kulkee siten, että tarkasteltava kordinaatisto siirtyy lähes int1, int2, jne. asemien kautta lopulliseen asentoon dest,

[18] engl. state variables

- BY t koordinaatisto siirtyy pitkin suoraviivaista rataa kiertyen samalla tasaisesti t:n ilmaisemalla tavalla ja
- AS rec liike tapahtuu etukäteen määrättyä rataa pitkin, radan pisteet sijaitsevat tiedostossa rec.

Robotin i nopeutta voidaan säätää SPEED I IMMEDIATE I i k-lauseella, missä k ilmaisee liikkeen nopeuden[19]. Lausetta IMMEDIATE käytettäessä nopeuden muutos tapahtuu välittömästi robotin ollessa liikkeellä. Robotin i liike voidaan pysäyttää välittömästi lauseella: STOP MOTION i.

Robotin ohjauskielessä tulee olla myös työkalujen käsittelyyn liittyviä komentoja. LM-kielessä on syntaksiltaan seuraavanlainen OPERATE-lause:

```
OPERATE tool i p1, p2, ...  pq I UNTIL b I I / I.
```

Lauseella käynnistetään robottiin i asennettu työkalu ja työskentelyä jatketaan kunnes ehdollinen lause on tosi. Työkalua käytetään parametrien p1, p2, jne. mukaisesti. Toteutettava toiminta pysäytetään lauseella: STOP OPERATION i. Esimerkkinä esitetään lause, joka sulkee kouraa kunnes leukojen puristusvoima on riittävän suuri:

```
OPERATE GRIPPER 1 0.0 UNTIL PRESSURE(1)>5.0.
```

Edellä on esitetty eräitä LM-kielen ominaisuuksia. Erityisen selvästi yhteys Pascal-kieleen näkyy seuraavassa esimerkissä, jossa robotti asentaa tappia reikään [Loz83]:

```
PROCEDURE INSERT (PIN FRAME, HOLE FRAME, FZMAX REAL, EPS REAL,
   NMAX INTEGER) BOOLEAN;
INTEGER N; VECTOR LATF; FRAME HOLE1;
BEGIN
    N := 0; HOLE1 := HOLE*TRANSLATION(-VZ,10);
    WHILE N < NMAX DO
    BEGIN
        MOVE PIN BY TRANSF (PIN, HOLE1) UNTIL FZ(1) > FZMAX /;
        IF DISTANCE(PIN, HOLE) < 1 THEN RETURN(TRUE)
        ELSE BEGIN
            LATF := VECTOR(FX(1), FY(1); 0);
            MOVE PIN BY TRANSLATION(LATF, EPS) /;
            EPS := EPS/2; N := N+1;
        END
    END
    WRITE "INSERTION FAILED"; RETURN(FALSE);
END
```

Lähteet: [Loz83], [PL83] ja [LM81].

[19] k = 0 , minimi ja k = 1 maksimi

4.17 LPR

LPR (**L**angage de **P**rogrammation de **R**obot) -kieli kehitettiin alunperin ohjaamaan Renault'n valmistamia ACMA-robotteja. Ensimmäinen versio kielestä valmistui vuonna 1978, mutta vaatimusten kasvaessa kehitettiin myöhemmin uusi, tehokkaampi versio. Kehitystyön perustana käytettiin Grafcet-kieltä. Kieli ei ole kaupallisesti saatavilla, sitä käytetään toistaiseksi vain Renault'n omien robottien ohjelmoinnissa. Ohjelmointi tapahtuu VAX-, Mitra- tai Philips-minitietokoneilla, joista valmis koodi ladataan robotin mikroprosessoreille.

LPR sisältää muun muassa tavanomaiset tietotyypit: kokonaisluvut, reaaliluvut, totuusarvot ja taulukot. Aritmeettiset ja loogiset operaatiot sekä matemaattiset vertailut ovat mahdollisia. Ohjelman kontrollointiin on käytettävissä osoitteet.

Usean robotin rinnakkainen prosessointi on mahdollista. Robottien liikerata voi olla koordinoitu tai lineaarinen. Radalle voidaan määrätä kauttakulkupisteitä ja liike voidaan keskeyttää tarvittaessa.

Lähde: [PL83].

4.18 MAL

Milanon teknillisessä korkeakoulussa kehitettyä MAL[20] -kieltä käytetään kaksoismanipulaattorilla varustetun Supersigma-robotin[21] ohjauksessa. BASIC-kieltä muistuttava MAL tarjoaa mahdollisuudet mm. tehtävien tahdistukselle ja samanaikaiselle suoritukselle sekä monipuoliset liikekomennot että aistimien liitännät. BASIC-kielestä poiketen aliohjelmat ovat parametroitavissa.

BASIC

MAL-ohjelman suoritus tapahtuu kahdessa vaiheessa. Ensimmäisessä vaiheessa käännetään MAL-kielinen ohjelma välikoodiksi, joka sitten seuraavassa vaiheessa tulkataan. Välikoodiohjelmalla ohjataan robotin hierarkista monimikroprosessorijärjestelmää, jossa kullakin nivelellä on oma mikroprosessori. Myös voima-antureita ohjataan erillisillä mikroprosessoreilla. Robottiin voidaan tietysti liittää myös kaksitasoisia antureita. Esimerkiksi Supersigman tarttujien sormiin on asennettu valodiodit, joilla voidaan seurata tarttumisen onnistumista.

Supersigman käsivarsien törmäilyä ei voida estää ohjelmallisesti vaan lähinnä mekaanisin keinoin.

Lähde: [GG85].

4.19 MAPLE

MAPLE on IBM:n kehittämä PL/1-kieleen perustuva robotin ohjauskieli. Rakenteellisen tason kieliin lukeutuva MAPLE muistuttaa paljon AL-kieltä. Ohjelmoinnissa käytettiin IBM 370/145 ja System 7 tietokoneita. Tutkimuskäyttöön tarkoitetun MAPLE-kielen käyttö jäi vähäiseksi kehittyneempien kielien[22] valmistuttua.

PL/1

MAPLE-kielessä on tavanomaisten tietotyyppien lisäksi mm. koordinaatistot, pisteet, viivat ja tasot. Ohjauslauseet ovat monipuoliset sisältäen esimerkiksi `IF ... THEN ... ELSE ...` , `DO ... END`, `GOTO`, `WHILE ... DO ...` ja `BEGIN ... END` -rakenteet. Kieli sisältää myös useita koordinaatistojen muunnoksissa käytettäviä komentoja, joista mainittakoon `TRANSLATED`, `ROTATED`, `FIXED` ja `RELATIVE`.

Robotin liikuttelussa käytettäviä komentoja ovat `MOVE TO ... BY` , `ROTATE TO ROTATE` ja `BY SWEEP TO ... BY` . Myös usean robotin rinnakkainen prosessointi on mahdollista. MAPLE sisältää myös erilaisten voima- ja kosketusantureiden käytössä tarvittavia komentoja. Näkö-järjestelmään tai työkalujen käyttöön liittyviä komentoja MAPLE:ssa ei ole.

Lähde: [BS82].

---

[20] Multipurpose Assembly Language

[21] modifioitu Olivetti SIGMA-robotti

[22] lähinnä AUTOPASS ja AML

4.20 MCL

APT

MCL (Manufacturing Control Language) kehitettiin USA:n Puolustusministeriön rahoittamassa ICAM-projektissa . Tavoitteena oli suunnitella teollisuuden käyttöön mahdollisimman yksinkertainen mutta monipuolinen robottijärjestelmien ohjauskieli. Kehitystyön perustana käytettiin NC-laitteiden ohjelmoinnissa suosittua APT-kieltä. Pyrkimyksenä oli kehittää kieli, jonka APT-kielen osaava konepajamies oppisi nopeasti.

PDP-11

ICAM:n kehittämää MCL-versiota käytetään suurehkossa IBM 370-tietokoneessa. Myöhemmin McAuton kehittämä suppeampi kaupallinen versio toimii PDP-11 tietokoneessa. Toistaiseksi MCL-kieltä käytetään vain Cincinnati T3 ja Westinghouse Allegro robottijärjestelmissä.

Erityisesti robotisoitujen solujen ohjaukseen soveltuva MCL-kieli sisältää mm. seuraavat tietotyypit: merkkijonot, reaaliluvut, taulukot, kooordinaatistot ja totuusarvot (`logic`). Robotin ohjauksessa oleellisia geometrisia tietotyyppejä ovat vektorit, muunnokset ja kiertymät. MCL sisältää seuraavat ohjausrakenteet: lauseiden nimet (osoitteet), aliohjelmat, `IF ... THEN ... ELSE ...` ja `WHILE ... DO ...` .

liikkeet

Robotin liikkeet voivat tapahtua suoraviivaisesti kahden annetun pisteen välillä, ympyräratoja tai geometrisesti määriteltyjä ratoja pitkin. Liikekomentoja ovat `FROM`, `GOTO`, `WKPNT`, `GOFWD`, `GOLFT` ja `GORGT`.

IO

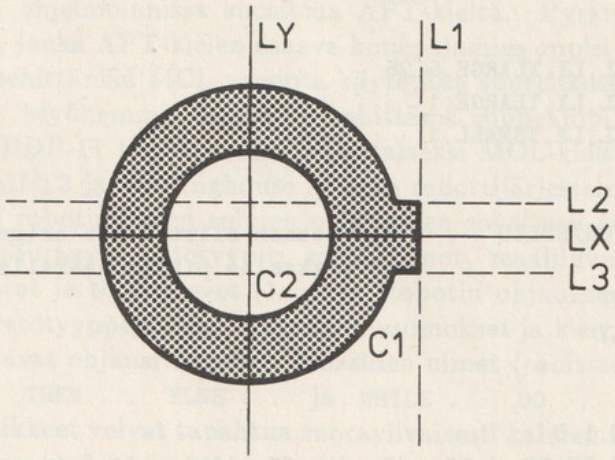
Usean robotin ja työkalun käyttö on solussa mahdollista, koska MCL-kieli sisältää runsaasti liitäntöjä. Digitaalisia ja analogisia portteja on kutakin yli 200. Myös rinnakkainen prosessointi on mahdollista käyttämällä `INPAR`-lausetta. Myös antureiden käyttöä varten on runsaasti komentoja, erityisesti huomiota on kiinnitetty näkökyvyn käyttöön. Esimerkiksi `REGION`-lauseella voidaan määritellä näkökentältä kappaleeseen kuuluvia alueita[23]. Määrittely muodostetaan liikekomennoilla sekä alueen laatua ilmaisevalla termillä. Viimeksi mainituista tavallisimpia ovat `SOLID`, `HOLE` ja `WINDOW`. `REGION`-lauseella muodostetut alueet yhdistetään `PROJEC`-lauseella ja muodostunut kappale paikannetaan komennolla `LOCATE` [WF83].

Esimerkkinä ohjelmoinnista MCL-kielellä esitetään osa näkökykyä käyttävästä paletointiohjelmasta. Kysymyksessä on näkökentässä olevan, kuvassa 4.1, esitettävän kappaleen määrittely [WF83].

Lähteet: [BS82], [GSCT84], [Loz83] ja [WF83].

[23] kuva on 2-dimensioinen

```
   .
   .
   .
PRTP=P/0,0,0              Maaritellaan kuvankasittelyssa
C1=C/0,0,4                tarvittavia vakioita (kts. seur. kuva).
C2=C/0,0,5
LX=L/XAXIS
LY=L/YAXIS
L1=L/PARALEL,LY,XLARGE,5.25
L2=L/PARALEL,LX,YLARGE,1
L3=L/PARALEL,LX,YSMALL,1
   .
   .
OUTLIN=REGION/SOLID       Aloitetaan siirtymalla origosta
   DNTCUT                    y-akselia pitkin kaarelle C2
   FROM/PRTP
   GO/ON, LY
   AUTOPS
   CUT
   INDIRV/0,1,0
   GOFWD/LY,ON,C2         Kaarta C2 pitkin siirrytaan
   GORGT/C2,ON,L2         kaaren ja suoran L2 leikkausp.
   GOLFT/L2,ON,L3         josta edelleen L2:n ja L3:n
   GORGT/L1,ON,L3         leikkauspisteeseen, jne.
   GORGT/L3,ON,C2
   GOLFT/C2,ON,LY         Tultaessa uudelleen kaaren C2 ja
   GOFWD/C2,ON,LY         suoran L2 leikkauspisteeseen on
ENDOF/REGION              alueen ulkoreuna valmis.
INNER=REGION/HOLE
   DNTCUT                 Maariteltaessa alueen sisareuna
   FROM/PRTP              lahdetaan origosta x-akselin suuntaan
   GO/ON,LX
   AUTOPS
   CUT
   INDIRV/1,0,0
   GOFWD/LX,ON,C1         Siirrytaan x-akselia pitkin
   GORGT/C1,ON,LX         kunnes saavutaan kaarelle C1
   GOFWD/C1,ON,LX         jossa kaannytaan oikealle ja
ENDOF/REGION              jatketaan kunnes ollaan jalleen
                          posit. x-akselilla.
PART1=PROJEC/OUTLIN;INNER       Maaritellyt alueet yhdistetaan
                                2-dim. malliksi PART1.
```



Kuva 4.1: Esimerkkiohjelmassa määriteltävä kappale.

4.21 MHI

MHI (**M**echanical **H**and **I**nterpreter) on yksi varhaisimmista robottien ohjaukseen tarkoitetuista kielistä. Se kehitettiin MIT[24]:ssa vuosina 1960 – 1961 alunperin MIT:n MH-1 robotin ohjaamiseen. MH-1 robottia ohjelmoitiin aikalaisistaan poiketen tavanomaisella tietokoneella. Lisäksi se sisälsi runsaasti kosketusantureita ja voima-antureita, jotka itse asiassa vaikuttivat merkittävästi kielen rakenteeseen.

Ohjelmointi MHI-kielellä perustui robotin liikkeiden valvontaa, esimerkiksi liikutetaan robotin kouraa kunnes tietty anturi ilmaisee, että päätepiste on saavutettu. MHI ei sisällä mitään aritmeettisia tai muita ohjausrakenteita. Tästä huolimatta se oli valmistuttuaan runsaasti aikaansa edellä.

MHI-kielen perusosat ovat:

`move`	liikkeen suunta ja nopeus
`until`	anturin testaus
`ifgoto`	ehdollinen hyppy ja
`ifcontinue`	ehdollinen alirutiinin kutsu.

Seuraavaksi esitetään esimerkki MHI-kielellä ohjelmoinnista [Loz83]:

[24]Massachusetts Institute of Technology

```
a, move x for 120          ;Move along x with speed 120
   until s1 10 rel lol     ;until sense organ 1 indicates a decrease of

                           ;10, relative to the value at start of this
                           ;step (condition 1)
   until s1 206 lol abs stp  ;or until sense organ 1 indicates 206
                           ; or less absolute, then stop (condition 2)
   ifgoto f1, b            ;if condition 1 alone is fulfilled go to
                           ;sequence b
   ifgoto t, f2            ;if at least condition 2 is fulfilled go to
                           ;sequence c
   ifcontinue t, a         ;in all other cases continue sequence a.
```

Lähde: [Loz83].

4.22 MINI

MIT:ssa vuosina 1972 - 1976 kehitetty ja käytetty MINI ohjauskieli on puhtaasti LISP-kielen laajennus. LISP-kieleen lisättiin vain eri laitteiden reaaliaikaiseen valvontaan liittyviä funktioita. Etuna tästä on se, että ohjelmoija voi halutessaan lisätä uusia ominaisuuksia MINI-kieleen ottaen luonnollisesti huomioon robottijärjestelmien asettamat rajoitukset. Kielestä puuttuvat mm. sisäänrakennetut geometriset ja ohjausrakenteet, mutta ne voidaan täysin korvata LISP-kielen funktioilla.

Eräänä rajoittavana tekijänä MINI-kielessä voidaan pitää robotin nivelkohtaista ohjausta, missä kutakin niveltä joudutaan ohjaamaan toisistaan riippumatta. Jokaiselle vapausasteelle asetetaan pääteasento tai rajoitetaan vaikuttavat voimat.

MINI:n perustoiminnot ovat:

- `GETM` luetaan paikka- ja voima-anturit sekä
- `WAIT` odotetaan annetun ehdon toteutumista.

Esimerkkinä MINI-kielestä esitetään keskeinen osa ohjelmasta, joka asettaa esimerkiksi pultin reikään [Loz83]:

```
(DEFUN MOVE-ABOVE (P OFFSET)
  ;set x, y, z, goals and wait till they are reached
  (X=(X-LOCATION P))
  (Y=(Y-LOCATION P))
  (Z=(PLUS(Z-LOCATION P) OFFSET))
  (WAIT'(AND(?X)(?Y)(?Z))))
(DEFUN INSERT (HOLE)
  (MOVE-ABOVE HOLE 0.25)
  ;define a target 1 inch below current position
  (SETQ ZTARGET (DIFFERENCE (GETM ZPOS) 1.0))
  ;move down until a contact force is met or until
  ;the position target is met
  (FZ=LANDING-FORCE)
  (WAIT'(OR(?FZ)(SEQ(GETM ZPOS)ZTARGET)))
  (COND ((SEQ (GETM ZPOS) ZTARGET)
       ;if the position goal was met i.e. no surfase encountered
       ;comply with lateral forces
       (FX=0) (FY=0)
       ;and push down until enough resistance is met.
       (FZ=INSERTION-FORCE)
       (WAIT'(FZ)))
       (T;if a surface was encountered
       (ERROR INSERT))))
```

Lähde: [Loz83].

4.23 ML

ML on IBM:n kehittämä, toiminnoiltaan lähinnä konekieltä muistuttava, robottikieli. Ominaisuuksista mainittakoon karteesisen robotin nivelkohtaiset liikekomennot, antureiden käyttö ja rinnakkainen prosessointi.

Robottien liikkeitä voidaan valvoa antureiden avulla SENSOR[25]-komennolla. ML-kielellä on mahdollista ohjata kahden robotin samanaikaista työskentelyä. Myös tehtävien yksinkertainen synkronointi on mahdollista.

Lähde: [Loz83].

[25]SENSOR -komennolla tarkkailtavan anturin aktivoituminen pysäyttää kaiken toiminnan, mikä on selvästi haitallista.

4.24 PAL

PAL-ohjauskieli poikkeaa selvästi muista aiemmin[26] kehitetyistä robottien ohjauskielistä. PAL-kieliset ohjelmat rakentuvat nimittäin kohteiden ja robotin tarttujan sijainnin ilmaisevista koordinaattiyhtälöistä. Eräät yhtälöiden välisistä muunnoksista, kuten esimerkiksi kappaleiden yksityiskohtien[27] sijainnit kappaleissa esitetään ohjelmassa eksplisiittisesti. Muut koordinaatistot määritellään implisiittisesti yhtälöillä.

koordinaatti-yhtälöt

PAL-kielinen ohjelma käsittelee robotin tärkeimpiä piirteitä määritteleviä peruskoordinaatistoja. Esimerkiksi voidaan määritellä, että Z edustaa robotin sijaintia maailman mallissa, T6 edustaa robotin kuudennen[28] varren asemaa Z:n suhteen ja E työkalun asemaa T6:n suhteen. Näin määriteltynä työkalun asema robotin sijainnin suhteen esitetään seuraavasti: Z + T6 + E, missä "+" tarkoittaa ko. koordinaatistojen välistä muunnosta (kuva 4.2).

Edelleen esimerkiksi yhtälö

```
Z + T6 + E = CAM + BKT + GRASP.
```

Asettaa tarttujan avatussa asennossa telineen yläpuolelle. Telineen sijainti on määrätty kameran suhteen näköjärjestelmällä.

PAL-kielen MOV <exp>-komennolla siirretään "yleinen" robotin työkalukoordinaatisto ARM + TOL paikkaan <exp>. Yksinkertaisissa liikkeissä työkalun sijainti ilmaistaan robotin paikan suhteen, jolloin ARM = Z + T6 ja TOL = E. ARM voidaan määritellä myös siten, että liike tapahtuu mielivaltaisen paikan suhteen. Yllä esitetty esimerkki voidaan kirjoittaa muotoon:

```
-BKT - CAM +Z +T6 + E = GRASP,
```

jolloin ARM on - BKT - CAM + Z + T6 ja tehtävä voidaan edelleen kirjoittaa lyhyesti: MOVE GRASP.

Toisena esimerkkinä tarkastellaan tapin asettamista telineen reikään. Tehtävä voidaan ratkaista määriteltyjen koordinaatistojen avulla seuraavasti:

```
- FIXTURE + Z + T6 + E - GRASP + HOLE = PIN.
```

Asettamalla ARM = - FIXTURE + Z + T6 + E - GRASP ja TOL = E - GRASP, on edellä esitetty lyhyesti: MOV PIN.

PAL-kielen kehittely on vielä kesken. Muun muassa antureiden käyttöä on tarkoitus laajentaa.

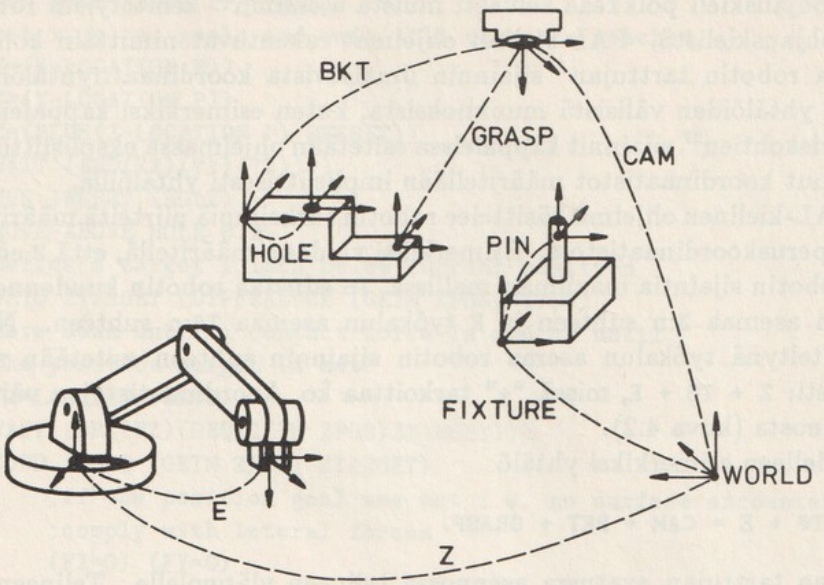
Lähde: [Loz83].

---

[26] ennen vuotta 1978

[27] esimerkiksi reiät ja ulkonemat

[28] viimeisen



Kuva 4.2: PAL-esimerkkien koordinaatistoja.

4.25 PASRO

AL & Pascal

PASRO (**Pas**cal for **Ro**bots) on saksalaisen Biomatic yhtiön tarjoama robottikieli, jota on kokeiltu mm. Microrobot-roboteissa. Kieli perustuu Pascal-kieleen, johon on lisätty robotin toimintojen vaatimia tietotyyppejä ja proseduureja. Nämä on talletettu erityiseen kirjastoon, josta mikä tahansa standardi Pascal-kääntäjä voi ottaa ne käyttöön. Kielen kehittely perustuu paljolti myös AL-kielestä saatuihin kokemuksiin.

Liikkeiden ohjaus tapahtuu joko pisteestä pisteeseen tai rataohjatusti.

Lähde: [GG85].

4.26 PLAW

PLAW (Programming Language for Arc Welding) -kieli on Komatsun kehittämä Komatsu-RW hitsausrobottien ohjauskieli. Kieli on hyvä esimerkki siitä, miten aikaisemmin opetusmenetelmällä ohjelmoidut työtehtävät tullaan antureiden käyttöönoton myötä ohjelmoimaan tekstuaalisesti. NC-kielen tyyppisen PLAW-kielen ohjelmoinnissa käytetään PDP-11 tietokonetta. *NC*

Kaarihitsauksen ohjelmointiin kehitetty PLAW sisältää kolme erilaista koordinaatistoa: robottiin kiinnitetty peruskoordinaatisto, hitsauskaareen kiinnitetty ja hitsauskaaren projektiokoordinaatisto (kuva 4.3).

PLAW-kielisissä ohjelmissa käsitellään hitsauskaaren asemaan ja kaaren kärjen paikkaa esittäviä paikkatietoja. Osa pisteistä määrätään käsin opettamalla ja osa antureiden avulla. Paikkaa ilmaisevia tietotyyppejä ovat: käsin opetetut, käsin opetetut ja antureilla tarkastettavat sekä antureilla määrättävät pisteet. Liikkekomentoja ovat muun muassa.

`MOVE Pn`	siirry pisteeseen `Pn`,
`SMOVE Pn1,Pn2`	lin. interpolaatio pisteiden `Pn1` ja `Pn2` välillä,
`CMOVE Pn1,Pn2,Pn3`	interpolaatio pitkin ympyränkaarta ja
`WMOVE (n1,n2)`	muuta hitsauskaaren asentoa.

Hitsaukseen liittyviä komentoja ovat esimerkiksi

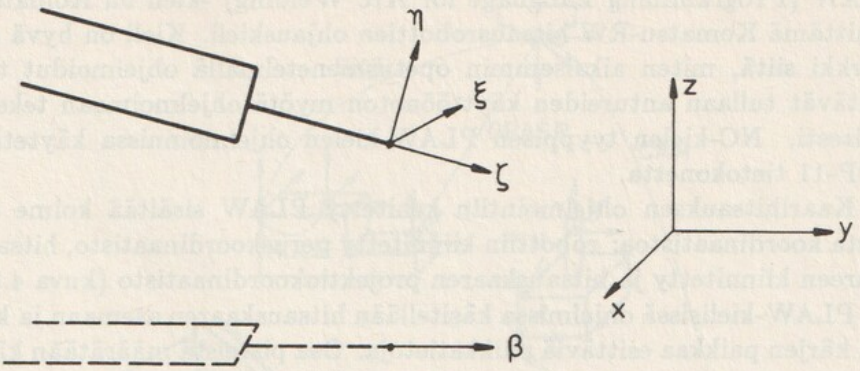
`SARC`	aloita hitsaus,
`EARC`	lopeta hitsaus,
`SPEED n`	hitsauskaaren kärjen nopeus,
`ADATA i,u`	hitsausvirta ja -jännite sekä
`WEAVE r,p`	kaaren edestakainen liike.

Toimintojen ohjauskomennot voidaan jakaa kahteen ryhmään: antureiden ohjauskomennot sekä adaptiiviseen toimintaan littyvät komennot. Ensiksi mainituista esimerkkinä komennot:

`SENSE`	anturin aktivointi ja
`GET En`	robotin aseman luku antureiden avulla.

Adaptiivisia ohjauskomentoja käytetään liikeratojen ja koordinaatistojen seurannassa. Näistä mainittakoon `CALL`-lauseessa käytettävät komennot:

`MEMO`	tallettaa radan muistiin,
`PLAY`	suorittaa muistiin talletetun radan,



Kuva 4.3: PLAW-kielessä käytettävät koordinaatistot.

`FPLAY` suorittaa (ja poistaa) muistiin talletetun radan,

`TRANS (n1,n2,n3)` siirtää koordinaatistoa suuntaan `(n1,n2,n3)`,

`TRANS Pn` siirtää sen hetkisen sijainnin pisteeseen `Pn` ja

`TCAN` lopettaa `TRANC`-komennon vaikutuksen.

Kommunikointi oheislaitteiden kanssa tapahtuu releiden välityksellä. Näihin liittyvät relekohtaiset `ON` ja `OFF` -komennot. Ohjauskäskyjä ovat muun muassa.

`GOTO` hyppykäsky,

`CALL n` aliohjelman `n` kutsu,

`RET` aliohjelmasta paluu,

`SFLAG n` aseta lippu,

`RFLAG n` nollaa lippu,

`IFLAG n,m` lipun testaus (lippu `n` = `m`),

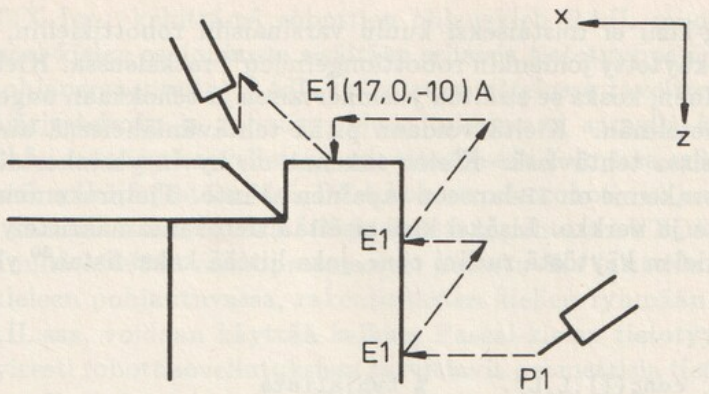
`ISGNL n,m` testataan kanava `n`, mikäli tosi siirrytään kohtaan `m`,

`WAIT n` odotetaan kunnes kanava `n` tosi sekä

`MOVE` -lauseen yhteydessä käytettävä

`ITOCH n,m` mikäli kanava `n` tosi, siirrytään lauseeseen `m`.

Esimerkkinä ohjelmoinnista PLAW-kielellä esitetään hitsauksen aloituskohdan etsiminen kosketusanturilla (kuva 4.4) [AUT81].



Kuva 4.4: PLAW-ohjelmointiesimerkin merkintöjä.

```
  MOVE P1                          ;P1 is taught previously
  SENSE 30,50                       ;Send parameter to the electrode
                                    ;touch sensor
  MOVE P1(30,0,0)A ITOCH 4,2       ;If #4 signal (the electrode touch signal)
                                    ;turns true while the robot is
                                    ;moving 30 mm from P1 toward X direction,
                                    ;jump to statement #2.
                                    ;If not, execute next statement.
  GOTO 5
2 GET E1                           ;Define E1 as the present position.
  MOVE E1(-10,0,-10)A
  MOVE E1(17,0,-10)A ITOCH 4,2
3 GET  E1
  MOVE E1(0,0,10)A ITOCH 4,4
  GOTO 5
4 GET E1
  MOVE E1(12,0,-12)A
  WMOVE (,45)                      ;Change the torch posture without
                                    ;moving the torch tip.
  MOVE E1(12,0,8)A
  SARC
    :
    :
    :
5 ON 8                            ;Error process routine
    :
    :
```

Lähteet: [PL83] ja [AUT81].

4.27 Prolog

Prolog-kieli ei toistaiseksi kuulu varsinaisiin robottikieliin, vaikka sitä onkin käytetty joidenkin robottiongelmien[29] ratkaisussa. Kieli on mielenkiintoinen, koska se sisältää yksinkertaisen ja tehokkaan ongelmanratkaisumenetelmän. Kieltä voidaan pitää tehtävänläheisenä ainakin yksinkertaisissa tehtävissä. Kielen rakenne on hyvin yksinkertainen. Ainoa ohjausrakenne on IF-lauseen tapainen sääntö. Tietorakenteista käytössä on lista ja verkko. Lisäksi kieli sisältää tietokantamäärittelyn. Esimerkkinä kielen käytöstä rutiini conc, joka liittää kaksi listaa[30] yhteen[31]:

```
conc([],L,L).      % tyhjalista

conc([X|L1], L2, [X|L3]) :- conc(L1,L2,L3).
```

Kun ensimmäinen lista on tyhjä tulos on yksinkertaisesti toisen parametrin ilmoittama lista (ensimmäinen sääntö), muuten tehtävä ratkaistaan käymällä läpi ensimmäistä listaa alkio kerrallaan[32], kunnes ensimmäinen lista loppuu (toinen sääntö). Lista kuvataan hakasulkujen avulla. [] on tyhjä lista. [X|Y] on lista, jonka ensimmäinen alkio on X ja loput listasta on (listamuuttuja) Y. Muuttujia merkitään isoin alkukirjaimin. Olkoonkin, että rutiini conc käyttää hyväksi kielen omaa listarakennetta, silti on vaikea kuvitella, miten tehtävän voisi enää yksinkertaisemmin kuvata. Esimerkiksi tavanomaisissa kielissä välttämätön indeksien ja simukoiden käyttö on Prolog-kielessä vältettävissä rekursion avulla. Kaikki muuttujat, tietokantaa lukuunottamatta, ovat paikallisia.

Prologin tärkein käyttökohde on tekoälysovellukset esimerkiksi robotiikassa. Myös tietokantatehtävät ovat usein Prolog-kielelle soveliaita. Erityistä huomiota kieli on saanut ns. viidennen sukupolven tietokoneen kehittelyn yhteydessä Japanissa. Tarkoitus on, että monimutkaiset tekoälyä vaativat tehtävät ohjelmoidaan juuri Prolog-kielellä. Ohjelmointikielten kehityksen kannalta Prolog on varmasti yhtä merkittävä kuin esimerkiksi Fortran, Algol, Lisp ja Simula [DMN68] ovat aikanaan olleet.

Ensimmäisiä Prolog-toteutuksia on vaivannut tulkattavalle kielelle tyypillinen hitaus. Sopivasti kieltä rajoittamalla[33] on kuitenkin päästy varsin tehokkaisiin käännettäviin toteutuksiin jopa mikrotietokoneissa [TUR86].

Lähteet: [Bra86] ja [TUR86].

---

[29] lähinnä geometriaa, liikkeitä ja tehtäviä koskevia

[30] 1. ja 2. parametri

[31] 3. parametri

[32] samalla siirretään ensimmäisen listan ensimmäinen alkio toiseen listaan

[33] lähinnä tyyppien ja määrittelyjen osalta

4.28 RAIL

AUTOMATIX Inc. kehittämä robottien ohjauskieli RAIL muodostuu laajasta Pascal-kielen osajoukosta sisältäen erilaisia tietotyyppejä ja korkeatasoisia ohjausrakenteita. RAIL-kielen suunnittelussa tavoitteena oli saada näköjärjestelmän ja robotin ohjaus toimimaan samalla kielellä. Kieltä on tähän mennessä sovellettu periaatteellisesti kolmessa erilaisessa järjestelmässä: näköjärjestelmässä, DEA kokoonpanorobotin ohjauksessa ja Hitachin kaarihitsausrobotissa. Ohjelmointi tapahtuu AUTOMATIX Inc. kehittämillä M68000 mikroprosessoriin perustuvilla laitteilla. *Pascal*

Pascal-kieleen pohjautuvassa, rakenteellisten kielien ryhmään lukeutuvassa RAIL:ssa, voidaan käyttää kaikkia Pascal-kielen tietotyyppejä. Lisäksi erityisesti robottisovellutuksissa tarvittavia geometrisia tietotyyppejä ovat koordinaatistot ja radat. Kieli sisältää tyypilliset Pascal-kielen ohjausrakenteet.

Robottien liikkeet voivat olla pisteohjattuja, jolloin robotin rakenne määrää radan pisteiden välillä. Myös suoraviivainen liike annettujen pisteiden välillä on mahdollista. Kieli sisältää myös hitsaussovelluksissa tarvittavia, hitsausjännitettä ja hitsauslangan syötöä ohjaavia komentoja. Näkökyvyn lisäksi vain rajakytkimiä voidaan käyttää toimintojen seurannassa. Usean robotin rinnakkaista prosessointia ei ole.

Esimerkkinä seuraavassa kaarihitsausohjelma, jonka ensimmäisessä osassa[34] hitsataan osat ja toisessa osassa suoritetaan puhdistus [FVdB82].

Lähteet: [PL83], [GSCT84], [Loz83] ja [FVdB82].

[34] funktiossa

```
FUNKTION WELD_PARTS
BEGIN
    FOR PART_NO = 1 TO 10 DO
    BEGIN
    ;
    ;move to HOME position, wait for next part, clamp it down,
    ;weld it, and unclamp
    ;
        MOVE SLEW HOME
        WAIT UNTIL PART_READY == ON
        CLAMP = ON
        APPROACH 2.0 FROM PART
        WELD PART WITH SPEEDSCHED(2), WELDSCHED(5)
        DEPART 2.0
        CLAMP = OFF
    END
    ;
    CLEAN_TORCH
END

FUNKTION CLEAN_TORCH
BEGIN
    ;
    ;Brush out the torch nozzle, then spray it.
    ;
    APPROACH 2.0 FROM CLEANER_BRUSH
    BRUSH = ON
    MOVE CLEANER_BRUSH
    DEPART 2.0
    BRUSH = OFF
    ;
    MOVE CLEANER_SPRAY
    SPRAY = ON
    WAIT 2 SEC
    SPRAY = OFF
    DEPART 2.0
END

INPUT PORT PART_READY 1
OUTPUT PORT CLAMP 1
OUTPUT PORT BRUSH 2
OUTPUT PORT SPRAY 3
```

4.29 RAPT

RAPT (**Robot APT**) on eräs APT-kielestä[35] kehitetty robottikieli. Alkuperäistä RAPT versiota käytettäessä laaditaan APT-tyyppisellä kielellä geometristen relaatioiden avulla kuvaus kokoonpanosta. Kuvaus muutetaan VAL-kieliseksi ohjelmaksi. RAPT ohjelmat sisältävät tarvittavien komponenttien, robotin ja työaseman kuvauksen sekä kokoonpanosuunnitelman. Suunnitelma muodostuu listasta geometrisia relaatioita, jotka ilmaisevat kokoonpanon tilan kunkin kokoonpanovaiheen jälkeen. Ohjelmat ovat siten käytettävästä robotista täysin riippumattomia.

APT
VAL

Haittana on pidettävä aistimien käyttön puuttumista. Aistimet tuleekin ottaa huomioon lopullisessa, VAL-kielisessä, ohjelmassa. Kokoonpanosuunnitelma ei sisällä myöskään minkäänlaista törmäystarkastelua.

Suunnitelmassa kappaleet kuvataan paikan ja joidenkin piirteiden avulla. Piirteitä ovat esimerkiksi taso-, sylinteri- ja pallopinnat. Relaatioista tärkeimmät ovat "vasten" ja "sovita" sekä erilaiset kierrot ja siirrot. RAPT-ohjelmalla muodostetaan näistä relaatioista verkko, jota pyritään minimoimaan sääntöjoukon avulla. Minimoidun verkon perusteella muodostetaan varsinainen VAL-kielinen kokoonpano-ohjelma robotille.

piirteet

Lähde: [GG85].

[35] APT on NC-koneiden ohjelmoinnissa yleisesti käytetty kieli.

4.30 ROBEX

APT

Saksan Liittotasavallassa, Aachenissa teollisuusrobottien ohjaukseen kehitetty ROBEX (**Robo**ter **EX**APT) -kieli muistuttaa myös rakenteeltaan ja syntaksiltaan APT-kieltä. ROBEX-kielen ensimmäisten versioiden ominaisuudet rajoittuivat pääasiassa työkappaleiden käsittelyssä tarvittaviin erikoispiirteisiin. Puutteita oli esimerkiksi työkalujen käsittelyssä. Rajoituksia oli myös antureiden käyttössä[36].

Kielen lauseet jaetaan viiteen ryhmään: geometriset, ohjaus-, liike-, tekniset ja ohjelmointilauseet. Geometrisia lauseita käytetään kuvattaessa työkappaleita ja työskentely-ympäristöä. Määrittelyistä mainittakoon kaksiulotteiset `POINT` (piste), `LINE` (suora) ja `CIRCLE` (ympyrän kaari) sekä kolmiulotteiset `CYLINDER` (sylinteri) ja `SPHERE` (pallo). Ohjauslauseilla tarkoitetaan tässä ohjausyksikön ja kääntäjän toiminnan ohjauksesta huolehtivia lauseita. Näiden avulla saadaan muun muassa yksityiskohtaiset virheilmoitukset.

Paikan ja orientaation muutokset työkalun tai tarttujan keskipisteen suhteen toteutetaan liikelauseilla. Liikkeet voidaan kuvata absoluutti- tai inkrementtiarvojen avulla. Edellisessä tapauksessa ilmoitetaan liikkeen päätepisteen koordinaatit, jälkimäisessä (pienen) siirtymän suuruus ja suunta. Myös ulkoisten laitteiden tahdistus ja robotin ohjaus opetusmenetelmällä on mahdollista.

Tekniset lauseet sisältävät muun muassa tarttujan toimintaa ohjaavia, liikkeiden nopeutta säätéleviä ja paletointiin liittyviä lauseita. Ohjelmointilauseet käsittävät esimerkiksi ehdolliset ja ehdottomat hyppylauseet sekä IO-tiedon käsittelylauseet.

4.30.1 ROBEX-M

ROBEX-kielen kehittyneemmässä versiossa, ROBEX-M-kielessä, tietotyypit ovat kokonaisluvut, reaaliluvut, matriisit, pisteet, kehykset `FRAME`, totuusarvot ja merkkijonot, joista voidaan muodostaa myös 1, 2 tai 3-ulotteisia taulukkoja.

ROBEX-M sisältää kaikki tavanomaiset aritmeettiset (`+`, `-`, `*` ja `/`sekä `**`) ja loogiset (`AND`, `OR`, `EXOR` ja `NOT`) sekä vertailuoperaatiot (`=`, `<>`, `<`, `>`, `>=`, `<=`).

Geometrisia määrittelyjä ovat `POINT`, `FRAME`, `MATRIX`, `TRASYS` sekä `ORIGIN`. `POINT`-määrittelyllä kuvataan tarvittavia pisteitä robotin karteesisessa koordinaatistossa. `FRAME` määrittelyllä voidaan siirtää robotin peruskoordinaatistoa haluttuun paikkaan ja asentoon. `MATRIX`-määrittelyllä puolestaan suoritetaan koordinaatistomuunnos. Uusi vertailukoordinaatisto voidaan määritellä `ORIGIN` käskyllä.

Liikekomennoilla voidaan robottia ohjata pisteestä pisteeseen (`MOVE`). Siirtymä tiettyyn suuntaan toteutetaan komennolla `CPMOVE` sekä samalla

[36]Vain kaksitasoisia signaaleja voitiin käyttää.

tapahtuva työkalun orientaation muunnos komennolla COMOVE. Nivelkohtainen liike toteutetaan DRIVE-komennolla. Komennolla HOME robotti siirtyy vertailupisteeseen. *liikkeet*

ROBEX-M sisältää useita teknisiä komentoja. Näistä mainittakoon nopeuden asetus FEDRAT, siirtymäaika pisteiden välillä MOVTIM, kiihdytys ACCEL ja hidastus DECEL, robotin valinta MACHIN, työkalun valinta TOOLNO, tarttujan sulkeminen CLOSGR ja avaaminen OPENGR, ohjelman suorituksen keskeytys STOP sekä viive DELAY ja signaalin odotuskomento WAITSIGNAL.

Antureiden yhteydessä tarvittavia komentoja ovat

EVENT	kaksitasoinen sisääntulo,
SWITCH	kaksitasoinen lähtö,
RECEIV	ulkoisen tiedon luku sekä
SEND	tiedon lähetys.

Ohjausrakenteita ovat IF ... THEN ... ELSE ... -rakenteen lisäksi muun muassa:

DO ... DOEND	silmukka,
REPEAT ... UNTIL	toisto kunnes ehto toteutuu,
CASE	valinta,
SUBRTN ... RETURN	aliohjelma,
CYCLE	toistettavia komentoja,
JUMP	ehdoton hyppy,
PAREND	rinnakkainen prosessointi sekä
TASK ... TASKEND	tehtävän määrittely ja suorituksen päättyminen.

Esimerkkinä ROBEX-kielestä esitetään ohjelman osa, jolla toteutetaan tarttujan tai hitsauspään ohjaus [WF83].

Lähteet: [WN84] ja [WF83].

```
1    PARTNO / 'GRASP'
2
3    $$  Locations
4
5    Box = POINT / 130, 240, 1800
6
7    $$  Subroutine for grasping objects
8
9    SUBRTN / GRASP ( OBJ )
10       OPENGR
11       FEDRAT / PATH, 45
12       CPMOVE / OBJ
13       CLOSGR
15   RETURN
15
16   $$  Main program
17
18   GRASP ( BOX )
19
20   FINI
```

4.31 ROL

ROL[37] syntyi ranskalaisen GIXI yhtiön tutkimustyöstä, jonka tarkoituksena oli kehittää kaupallinen robottien ohjelmointijärjestelmä. BASIC-kieleen perustuvasta ROL:sta on muodostunut monipuolinen kieli, jota voidaan periaatteessa soveltaa useimpiin robotteihin. Vuonna 1983 julkistettua versiota voidaan modulaarisuutensa ansiosta käyttää niin yksinkertaisten kuin usean robotin muodostamien järjestelmien ohjaamisessa. Ohjelmointi tapahtuu IBM PC:llä.

Rakenteellisten kielien ryhmään kuuluvassa ROL:ssa on taulukkoja lukuunottamatta käytössä tavanomaiset robottien ohjauskielien tietotyypit sekä ohjausrakenteet. Myös aritmeettisia ja loogisia operaatioita sekä matemaattisia funktioita voidaan käyttää.

Pisteittäinen, suoraviivainen tai ympyrätaliike ovat mahdollisia, samoin usean robotin rinnakkainen ajo. Oheislaitteiden tahdistuksessa käytetään `signal-wait`-rakennetta.

Lähteet: [PL83].

4.32 RPL

Kehitettäessä[38] RPL-kieltä tavoitteena oli nykyaikaisten automaattisten tuotantojärjestelmien ohjaukseen mahdollisimman laajasti soveltuva ja helposti ohjelmoitava kieli, joka on tarkoitettu pääasiassa käyttöhenkilökunnalle, ei ammattimaisille ohjelmoijille.

RPL-kieli muistuttaa LISP-kieltä, johon on sovellettu FORTRAN-kielen syntaksia! Käytännössä ohjelmoija laatii aliohjelmakirjaston, josta sovellutuskohtaisesti kutsutaan tarvittavat aliohjelmat. Ohjelmoinnissa käytetään tavallisesti PDP-11/45- tai LSI 11- tietokonetta. Tähän mennessä RPL-kieltä on käytetty Unimation PUMA 550 robotissa sekä Machine Intelligence VS-100 näköjärjestelmässä.

LISP & Fortran

Lähinnä perusliikkeiden tason kieliin lukeutuva RPL sisältää kokonaisluvut, reaaliluvut ja taulukot. Geometrisia tietotyyppejä ei ole. Ohjausrakenteiden valikoima on tavallinen sisältäen osoitteet, `IF ... THEN ... ELSE ...` , `WHILE ... DO ...` , `DO ... UNTIL` , `CASE` , `FOR` ja `BEGIN ... END` .

Tavanomaisen pisteohjauksen lisäksi robotti voi liikkua suoraviivaisesti annettujen pisteiden välillä tai liikerata voidaan ohjelmoida kulkemaan usean välipisteen kautta. Liikekomento on `MOVETO`.

Järjestelmässä on runsaasti portteja sekä kommunikointi- ja anturikomentoja[39]. Työkalujen toimintaa ohjaavia komentoja ei ole, eikä myöskään usean robotin rinnakkaista ajoa.

Lähteet: [GSCT84] ja [BS82].

---

[37] Robot Language

[38] SRI International

[39] erityisesti näköjärjestelmän käyttöön liittyviä

4.33 RSS

RSS (Robot Servo System) robottien ohjelmointijärjestelmä ja -kieli kehitettiin Illinoisin yliopistossa 70-luvun lopulla. Kehitystyön perustana pidettiin anturiohjattujen robottien ohjelmointia. Järjestelmän ja kielen avulla ohjelmoija määrittelee suoraan erilaisia robotin toimintoja ohjaavia servoprosesseja. Prosessit voidaan jakaa paikkaa, asentoa, voimaa ja vääntöä sääteleviin prosesseihin. RSS-kieltä on käytetty muun muassa PDP-11/40 minitietokoneessa yhdessä näköjärjestelmästä huolehtivan PDP-10 tietokoneen kanssa.

servot

Tavanomainen robottien ohjelmointikieli sisältää lausekkeita ja komentoja, joilla määrätään esimerkiksi robotin liikeratoja ja liikkeiden nopeuksia. RSS-kielellä ei voida kuvata robotin toimintoja suoraan, vaan ohjelmoinnissa joudutaan kuvaamaan toimintoja toteuttavat servotoiminnot. Tästä syystä eräät toiminnot vaativat muita ohjelmointikieliä enemmän komentoja. Esimerkkinä mainittakoon tarttujan siirto annettuun pisteeseen. Esimerkiksi AL-kielessä tämä toteutetaan lauseella:

```
MOVE TO A.
```

Vastaava RSS-kielinen lause on:

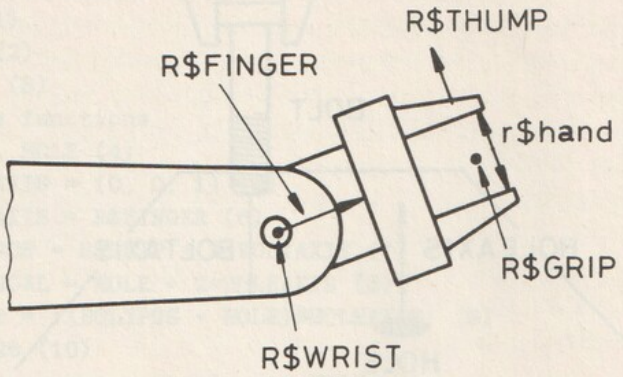
```
position point R$GRIP = A.
```

Tämä aloittaa servotoiminnon, jolla siirretään tarttujaa siten, että ero pisteen `A` ja tarttujan `R$GRIP` välillä pienenee. Toiminto jatkuu kunnes käyttäjä kumoaa sen tai toiminto, tarttujan liike, määritellään uudelleen. Näin ollen tarttujan siirtämisessä annettuun pisteeseen tarvitaan RSS-kielessä vähintään kaksi lausetta. Toisaalta useimmat säätöongelmat, esimerkiksi näköjärjestelmän yhteydessä, ovat tavallista helpompia toteuttaa.

RSS-kieltä on vaikea asettaa mihinkään kohdassa 2.2 esitettyyn luokkaan. Servotoimintojen ohjauksen perusteella kieli voitaisiin luokitella prosessoritason kieliin, mutta erityisesti antureiden monipuolinen käyttömahdollisuus korottaa tasoa. Rakenteinen ohjelmointi on kuitenkin mahdotonta.

Kaikki paikat RSS-kielisissä ohjelmissa esitetään vektoreina robotin karteesisessa koordinaatistossa. Manipulaattorin tilaa, ts. asemaa, kuvataan jatkuvasti päivittyvillä muuttujilla (kuva 4.5):

`R$GRIP`	robotin tartuntapiste,
`R$FINGER`	tarttujan sormien asento,
`R$THUMB`	suunta sormesta yksi sormeen kaksi,
`R$WRIST`	ranteen paikka sekä
`R$HAND`	tarttujan sormien välinen etäisyys.



Kuva 4.5: Tarttujan paikka- ja asentomuuttujat.

Tarttujan paikan määräämisessä voidaan käyttää piste, viiva ja tasomäärittelyä. Siirrettäessä tarttuja pisteeseen (-10, 40, 50) käytetään lausetta:

```
position point = (-10, 40, 50).
```

Lauseella:

```
position line = (-10, 40, 50);(0, 0, 1).
```

siirretään tarttuja kyseisen pisteen kautta kulkevalle, vektorin (0, 0, 1) suuntaiselle suoralle. Tasolle, joka kulkee kyseisen pisteen kautta ja jonka normaali on vektorin (0, 1, 0) suuntainen, siirretään tarttuja lauseella:

```
position plane = (-10, 40, 50);(0, 1, 0).
```

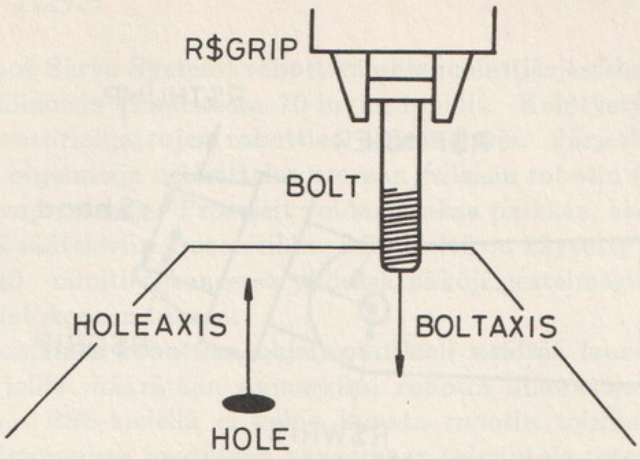
RSS-kielessä on käytettävissä normaalit aritmeettiset operaattorit. Lisäksi voidaan käyttää vektoreiden piste- ja ristituloa sekä määrätä vektoreiden pituudet ja suunta. Käyttäjä voi määritellä myös omia funktioita. Esimerkiksi

```
define DRILL = R\$GRIP + 10*R\$FINGER
```

määrittelee funktion `DRILL`, joka sisältää pisteen 10 cm:n etäisyydeltä tarttujasta sormien suuntaan. `define` määrittelyllä voidaan tallettaa muuttujan sen hetkinen sisältö toiseen muuttujaan käyttämällä aaltosulkuja:

```
define HOLE =  {DRILL} .
```

Antureista saatava tieto esitetään dynaamisesti vaihtuvilla funktioilla. Esimerkiksi ranteeseen vaikuttava voima ja vääntö on luettavissa vastaavista funktioista `R$FORCE` ja `R$TORQUE`. Ranteen voima ja vääntö tietyllä suuruudella ja suunnalla määritellään seuraavasti:



Kuva 4.6: Pultin asetus

```
force VECTOR
torque VECTOR.
```

Voiman ja väännön suunta ja suuruus ovat nyt VECTORin suunta ja suuruus. Näköjärjestelmän käytössä tarvitaan joitakin erikoisfunktioita. Esimerkkinä mainittakoon:

`vision BOLT`	kuvaa pultti (`BOLT`),
`locate BOLT`	paikanna pultti ja
`track BOLT`	seuraa (jatkuvasti) pulttia.

Liikkeiden nopeuksia säädellään määritellyllä aikafunktiolla. Esimerkiksi tarttujan poistuminen pisteestä `HOLE` y-akselin suuntaan nopeudella 2 cm/s toteutetaan lauseella:

```
position point R\$GRIP = HOLE - 2*time*(0, 1, 0).
```

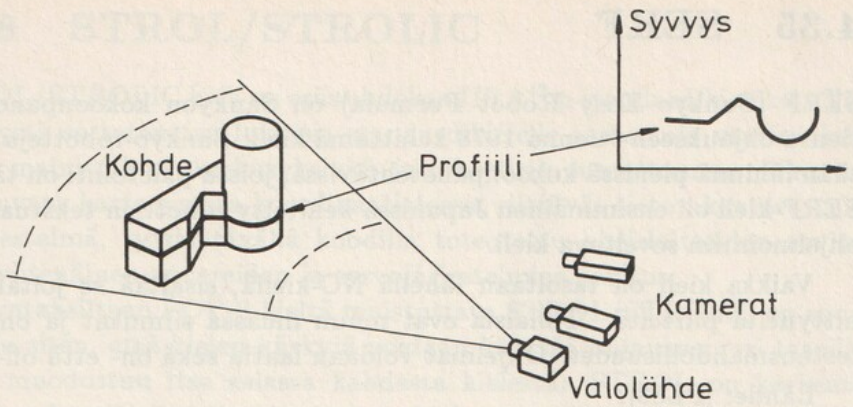
Ehdollisista rakenteista mainittakoon `wait until` -lause. Ohjelmien ohjauksessa puolestaan voidaan käyttää ehdollisia ja ehdottomia haarautumiskäskyjä.

Esimerkkinä ohjelmoinnista RSS-kielellä esitetään näköjärjestelmään perustuva pultin reikään asetusohjelma. Erinäiset muuttujat selvinnevät kuvasta 4.33 [Ges83].

```
;Bolt insertion routine
;Clifford C. Geschke 9-12-78
servo off (1)
force ZERO (2)
torque ZERO (3)
;Define some functions
vision BOLT, HOLE (4)
define HOLEAXIS = (0, 0, 1) (5)
define BOLTAXIS = R$FINGER (6)
define BOLTPOS = R$GRIP + 2*BOLTAXIS (7)
define BOLTGOAL = HOLE + 2*HOLEAXIS (8)
define error = !(BOLTPOS - HOLE)#HOLEAXIS! (9)
define f = 25 (10)
;Tell vision processor to locate HOLE
locate HOLE (11)
wait until flag found HOLE (12)
print HOLE found (13)
;Move to above estimated hole location
orient fixed BOLTAXIS; R$THUMB = -HOLEAXIS; (-1, 0; 0)
(14)
positin point BOLTPOS = BOLTGOAL (15)
servo on (16)
wait until !BOLTPOS - BOLTGOAL!; ls .5 (17)
;Locate and trac BOLT
trap 2 to lostit on flag lost BOL T (18)
locate BOLT                                    (19)
wait until flag found BOLT (20)
print BOLT found (21)
track BOLT (22)
define BOLTPOS = BOLT (23)
position line BOLTPOS = HOLE; HOLEAXIS (24)
;Insert BOLT into HOLE
fwait: force ZERO (25)
wait until error; ls .1; (26)
trap 1 to fwait on error; gtr .1; (27)
force f*BOLTAXIS (28)
wait until R$FORCE*BOLTAXIS; ls -.8*f; (29)
print All done! (30)
stop (31)
;Error routine if BOLT is lost
lostit: servo off (32)
forget BOLT (33)
print Lost BOLT (34)
stop (35)
```

Ohjelman riveillä 1 - 3 pysäytetään kaikki mahdollisesti toiminnassa olevat servoprosessit. Riveillä 4 - 10 määritellään erilaisia funktioita ja vakioita. Näköjärjestelmän avulla paikallistetaan reikä riveillä 11 - 13. Rivillä 14 asetetaan tarttuja sellaiseen asentoon, että pideltävän pultin ja reiän akselit ovat samansuuntaiset. Riveillä 15 - 17 liikutetaan tarttuja siten, että pultti tulee 2cm:n etäisyydelle reiästä. Riveillä 19 - 22 paikallistetaan pultti näköjärjestelmällä, pultin paikkaa päivitetään jatkuvasti. Jos pultti kadotetaan, pysäytetään kaikki servoprosessit ja annetaan virheilmoitus (rivit 18, 32 - 35). Pultti siirretään reiän yläpuolelle (23 - 24). Pultin ollessa 1mm:n etäisyydellä reiästä, työnnetään sitä annetulla voimalla reikään. Voiman ylittäessä tietyn rajan todetaan tehtävä suoritetuksi ja tarttuja irrotetaan pultista.

Lähde: [Ges83].



Kuva 4.7: Leikkausprofiilitiedon keruu RVL/A kielessä.

4.34 RVL/A

RVL/A on japanilainen[40] näköjärjestelmän käyttöön perustuva robotin ohjauskieli. Useista muista vastaavista poiketen tässä järjestelmässä tarkastellaan 3-D maailmaa. Kappaleita tarkastellaan leikkausprofiilien avulla (kuva 4.7). RVL/A-kielen kehittelyssä otettiin huomioon muun muassa seuraavat asiat: *3-D*

- kohteiden tunnistus ja mittaus kyettävä kuvaamaan suoraan,
- kohteiden mallitus ja kuvan ymmärtämistä helpottavien funktioiden laatiminen helppoa,
- 3-D tiedon käsittely mahdollista ja
- yksinkertainen ajantasainen näkötieto ja kohteiden mallitus.

RVL/A-kieli toteutettiin LISP-kielen kaltaisella PETL-kielellä.

Näköjärjestelmässä käsiteltävä tieto voidaan jakaa kolmelle tasolle: 1-ulotteinen tieto leikkausprofiilista, lista kohteen leikkausprofiileista sekä PETL-kielen attribuuttilistana esitettävä kohteen muotoa ja dimensioita kuvaava tieto. Profiilitiedon analysoinnin perusteella muodostetaan kohteesta malli määrittelyillä `APPEARANCE` ja `VIEW` (kuva 4.7). `APPEARANCE` kuvaa profiilin muotoa tietystä näköpisteestä tarkasteltuna.

Eräistä hyvistä ominaisuuksista huolimatta [41] pidetään RVL/A-kieltä vielä keskeneräisenä. Parantamisen varaa on muun muassa ohjelmoinnin helppoudessa sekä mallien määrittelyissä. Tulevaisuudessa RVL/A lienee yksi varteenotettavista aistimien käyttöön perustuvista ohjelmointikielistä.

Lähde: [Jap84]

[40] Electrotechnical Laboratory, Ibaraki

[41] esimerkiksi nopea mallitetun kohteen tunnistus

4.35 SERF

Sankyo

SERF (Sankyo Easy Robot Formula) on Sankyon kokoonpanorobottiensa ohjaukseen vuonna 1978 kehittämä kieli. Sankyo-robotteja käytetään lähinnä pienissä kokoonpanotehtävissä, joissa paletointi on tärkeää. SERF-kieli oli ensimmäinen Japanissa kehitetty robottien tekstuaaliseen ohjelmointiin soveltuva kieli.

NC

Vaikka kieli on tasoltaan lähellä NC-kieliä, sisältää se joitakin kehittyneitä piirteitä. Tällaisia ovat muun muassa silmukat ja ohjelman testausmahdollisuudet. Ohjelmat voidaan laatia sekä on- että off-line.

Lähde: [PL83].

4.36 SIGLA

Olivetti

Olivetin vuonna 1974 kehittämä SIGLA (Sigma Language) oli ensimmäinen kaupallinen teollisuusrobottien ohjauskieli. Sitä käytettiin pääasiassa Olivetin Cartesian Sigma robottien ohjauksessa. Sigma robotteja on kehitetty kokoonpanoon, pintakäsittelyyn ja kaarihitsaukseen.

Vaikka SIGLA on hyvin lähellä NC-kieliä, sisältää se useita kehittyneemmille kielille ominaisia piirteitä. Näistä mainittakoon usean robotin rinnakkainen prosessointi ja antureiden käyttö. Erityisesti tarttujaan sijoitetulla jännitysanturilla voidaan tarkkailla kokoonpanon onnistumista. Näistä huolimatta on kieli nykyisen mittapuun mukaan vanhentunut ja sen käyttö vähennee.

Lähteet: [PL83] ja [Loz83].

4.37 SRL

Pascal
AL

SRL (Structured Robot Language) on Karlsruhen yliopistossa kehitettävä robottikieli, joka sisältää piirteitä sekä AL-kielen siirrettävästä versiosta että Pascal-kielestä. Pascal-kielen tietotyyppien lisäksi SRL-kielessä on joitakin tekoälyyn liittyviä tietotyyppejä. SRL onkin eräs ensimmäisistä robottikielistä, joissa tekoälyn käyttö on otettu huomioon.

SRL-ohjelman määrittelyosa sisältää myös järjestelmän komponenttien kuvauksen, jolloin robottiohjelmaan on helppo liittää ympäristön hallintaan liittyviä toimenpiteitä. Ohjelman käskyt voidaan suorittaa peräkkäisesti, rinnakkaisesti, jaksottaisesti tai viivästetysti. SRL sisältää myös monipuoliset liikekomennot mahdollistaen suoraviivaisen ja ympyräliikkeen.

Ennen robotille siirtoa SRL-kielinen ohjelma käännetään koneriippumattomaksi IRDATA-välikoodiksi.

Lähteet: [GG85] ja [Blu85].

4.38 STROL/STROLIC

STROL/STROLIC kieli on eräs ehdokas JIRA[42]:n standardikieleksi. Tästä syystä se toteuttaa lukuisia standardikielelle asetettuja vaatimuksia, joista mainittakoon keskusyksikkö- ja robottiriippumattomuus, liikeratojen kuvaus karteesisessa koordinaatistossa, riittävät tietorakenteet, tulkkijärjestelmä, tyydyttävällä koodilla toteutettu oheislaitteiden sovitus sekä yhtenäinen antureiden ja servojärjestelmien sovitus. *JIRA*

Syntaksiltaan POP-2-kieltä muistuttava STROL/STROLIC on suunniteltu siten, että kielen käskyjä voidaan käyttää ohjauksen eri tasoilla. Kieli muodostuu itse asiassa kahdesta kielestä: STROL on korkeatasoinen robottikieli, jolla kirjoitetut ohjelmat muutetaan automaattisesti virtuaalisen tietokoneen (ohjausyksikön) konekieliseksi STROLIC-ohjelmaksi, jotka lähetetään edelleen tietokoneesta robotin ohjausyksikölle.

STROL-kielessä on muun muassa melko monipuoliset liikekomennot. Tavanomaisen `MOVE` -komennon lisäksi voidaan liike suorittaa lyhyin siirtymin `IMOVE` -komennolla sekä annetun vektorin osoittamaan pisteeseen `MOVEC` -komennolla. Suoraviivainen liike saadaan `MOVES` -komennolla.

Myös STROL/STROLIC-kieltä voidaan pitää keskeneräisenä. Erityisesti mikäli kieltä aiotaan esittää standardikieleksi, on siihen tehtävä lisäyksiä muun muassa tietotyyppeihin sekä parannettava laiteriippumattomuutta.

Lähde: [Jap84].

4.39 TEACH

TEACH-kieli kehitettiin vuosina 1975 - 1978 osana Bendix Corporationin PACS-ohjausjärjestelmää. PACS-järjestelmän erityisenä tavoitteena oli selvittää usean robotin ja oheislaitteen rinnakkaiseen toimintaan liittyviä ongelmia, joiden vähentämiseksi käytettiin muun muassa seuraavia ratkaisuja: *rinnakkaisuus PACS*

- Ohjelmat muodostuvat osittain järjestetyistä lauseista, jotka voidaan suorittaa jaksoittain tai rinnakkain.
- Joustava laitteiden määrittely.
- Kaikki liikkeet määritellään paikallisen koordinaatiston suhteen, jolloin liikkeiden määrittely uudelleen on helppoa.

Lähde: [Loz83].

[42] Japanese Industrial Robot Association

4.40 TL-10

BASIC

TL-10 (**T**oyota Robot **L**anguage - 10) on kuusiakselisten Toyota-robottien ohjelmointiin kehitetty, BASICin tapainen kieli. TL-10-ohjelmointiympäristö sisältää vähintään minitietokoneen, päätteen ja levykeaseman. Järjestelmään sisältyy usein käsiohjain, jolla ohjataan ohjelman suorituksen aikaisia toimintoja. Tässä tehtävässä voidaan käyttää myös näyttöpäätettä.

Sujuva erillään ohjelmointi edellyttää monipuoliset editointikomennot. Tavanomaisen editoinnin lisäksi on suora editointi mahdollista. Toisin sanoen ohjelmaa voidaan muokata siirtymättä varsinaiseen editointitilaan. Suoran editoinnin aikana voi robotti työskennellä normaalisti.

Tyypillinen TL-10-kielinen ohjelma muodostuu pääohjelmasta, joukosta aliohjelmia sekä taulukoista (kuva 4.8). Ero pääohjelman ja aliohjelman välillä on näennäinen, sillä jokainen aliohjelma voidaan ajaa itsenäisesti ilman varsinaista pääohjelmaa. Aliohjelma voi siten olla esimerkiksi jonkin toisen sovelluksen pääohjelmana. Aliohjelmalla voi olla parametrina kokonaisluku, reaaliluku, vektori, ratapiste tai ratapistetaulukko. Taulukoiden maksimikokoa ja lukumäärää ei ole määritelty. Myös taulukoiden käsittelykomentoja on TL-10-kielessä useita. Robotin ohjaus voidaan toteuttaa joko piste- tai rataohjatusti.

Ohjelmissa tarvittavat muuttujat kirjataan muuttujien tyyppiä vastaaviin, ohjelmista riippumattomiin muuttujatauluihin. Mahdolliset tyypit ovat kokonaisluku, reaaliluku ja vektorimuuttujat. Tarvittavat taulut linkitetään ohjelmaan editointivaiheessa, jolloin taulut liittyvät automaattisesti ohjelmaan sen käynnistyessä.

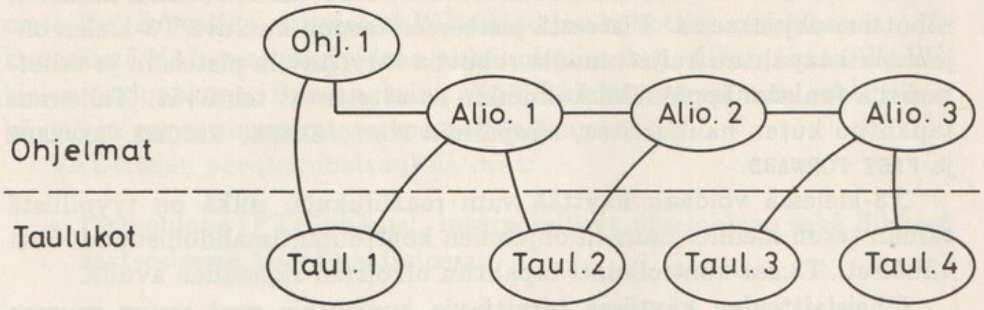
Esimerkkinä TL-10-kielestä esitetään aliohjelma, jossa robottia liikutetaan työkalukoordinaatistossa. Esimerkkiin sisältyy myös aliohjelman kutsu pääohjelmassa [NHK*83]:

Pääohjelman alussa annetaan ohjelman ja muuttujataulukon nimi. Aliohjelman `TOOL_MOVE` kutsu sisältää parametreina lähtöpisteen koordinaatit[43] sekä siirtymän `VEC1`[44]. Pisteen koordinaatit on talletettu taulukkoon `A`. Aliohjelmassa nimen ja muuttujataulun esittelyn jälkeen määritellään parametrit `POINT` ja `VECTOR` sekä asetetaan 4×4 yksikkömatriisi `P`. `0`:n paikkaosaan lisätään vektori `VECTOR`[45]. Tämän jälkeen matriisiin `P` lisätään robotin sijainti robotin karteesisessa koordinaatistossa. Lopuksi siirretään robotin tarttuja tai työkalu kyseiseen pisteeseen.

[43] tyyppi L = piste

[44] tyyppi V = vektori

[45] P vastaa homogeenista muunnosmatriisia



Kuva 4.8: TL-10-ohjelman rakenne.

```
     PROGRAM NAME --- MAIN_PRG
     TABLE NAME   --- MAIN_TAB
.
.
070     CALL TOOL_MOVE (L/A, V/VEC1)
.
.
--------------------------------

     PROGRAM NAME --- TOOL_MOVE
     TABLE NAME   --- TOOL_MOVE

010  ;   robot motion in tool coordinates
020  ;   POINT -- location in world coordinates
030  ;   VECTOR -- vector in tool coordinates
040  ;   UNIT_MATRIX is the 4 x 4 unit matrix
050  ;
060  DEC    (L/POINT, V/VECTOR)
070  SETL    P = UNIT_MATRIX
080  SHIFT   P, VECTOR      ;add VECTOR to the position vector in P
090  SETL    P = POINT + P
100  MOVES   P
110  RETURN
```

Lähde: [NHK*83].

4.41 T3

T3-kieltä käytetään Cincinnati Milacronin valmistamien saman nimisten robottien ohjauksessa. Pisteestä pisteeseen tasoon kuuluva T3-kielen ohjelmointi tapahtuu kuljettamalla robottia tarvittaviin pisteisiin ja talletttamalla funktionäppäimillä kulloinkin suoritettavat tehtävät. Tallennus tapahtuu kuten nauhureissa, näppäimin `PLAY`, `ERASE`, `RECORD`, `REVERSE` ja `FAST FORWARD`.

T3-kielessä voidaan käyttää vain reaalilukuja, mikä on tyypillistä tämän tason kielille. Samoin ohjelmien kontrollointimahdollisuudet ovat vähäiset. T3:ssa kontrollointi tapahtuu ulkoisten signaalien avulla.

Oheislaitteiden käytössä tarvittavia komentoja ovat muun muassa `WAIT`, `OUTPUT` ja `EXTERNAL`. Työkalujen ohjaus tapahtuu tarkkailemalla työkalun tilaa ja lähettämällä vaadittavia ohjaussignaaleja. Antureiden käyttömahdollisuudet ovat vähäiset. Mekaanisia kosketusantureita voidaan käyttää, muttei esimerkiksi näköjärjestelmää.

Lähde: [BS82].

4.42 VAL

Vuonna 1975 kehitettyä VAL (VicArm Language) -kieltä käytetään Unimate Inc. robottien, erityisesti PUMA-sarjan robottien ohjauksessa. Alkuperäistä VAL-versiota voidaan pitää minitietokoneelle sopivana WAVE-kielen osajoukkona. Parannuksina WAVE-kieleen verrattuna ovat tulkin käyttö ja parantuneet ratalaskentaominaisuudet. *WAVE*

VAL-kielen perusominaisuuksia ovat:

- Pisteohjaus (PTP-ohjaus), interpoloituva nivelohjaus sekä liikkeet karteesisessa koordinaatistossa.
- Koordinaatistojen määrittely ja käsittely sekä paikan määrittely mielivaltaisessa suorakulmaisessa koordinaatistossa.
- Kokonaisluvut, aritmeettiset operaatiot, ehdolliset haarautumiset sekä proseduurit.
- Kaksitasoisten linjojen käyttö antureiden yhteydessä ja oheislaitteiden tahdistuksessa.

Antureiden käyttö on rajoittunut kaksitasoisten antureiden, kuten rajakytkimien käyttöön. Tämä riittää kuitenkin liikkeiden valvonnassa. VAL perusversiota on sovellettu myös näköjärjestelmässä.

Kielenä VAL muistuttaa BASIC-kieltä. Mahdolliset tietotyypit ovat kokonaisluvut sekä koordinaatistot, nivelkulmat ja koordinaatistomuunnokset. Kielestä puuttuvat muun muassa merkkijonot ja aritmeettiset operaatiot eikä proseduureissa voi käyttää parametreja. *BASIC*

Kontrollirakenteet ovat `GOTO`, `IF ... THEN ...` ja `IFSIG ... THEN ....` Liikekomentoja ovat `GO`, `MOVE`, `MOVEI` ja `APPRO` . Komennoilla `MOVEIS` ja `APPROS` saadaan aikaan suoraviivainen liike. Voima- ja kosketusantureiden yhteydessä käytettäviä komentoja ovat esimerkiksi `CALIB`, `REACT`, `REACTI`, `SIGNAL` ja `WAIT`. VAL sisältää myös joitakin oheislaitteiden käsittelyssä tarvittavia komentoja kuten `IGNORE`, `IFSIG` ja `SIGNAL`.

Esimerkkinä jälleen tapin nouto ja asetus reikään. Voima-antureiden seurannassa käytetään erillisiä prosessoreita. Erityisesti linja 3 asettuu `high`-tilaan tarttujaan vaikuttavan voiman z-komponentin ylittäessä asetetun rajan [Loz83].

```
     SETI    TRIES=2
     REMARK  If the hand closes to less than 100 mm,
     REMARK  go to statement labelled 20
10   GRASP   100,20
     REMARK  Otherwise continue at statement 30
     GOTO    30
     REMARK  Open the fingers, displace down along world Z-axis
     REMARK  and try again
20   OPENI   500
     DRAW    0,0,-200
     SETI    TRIES=TRIES-1
     IF TRIES GE 0 THEN 10
     TYPE    NO PIN
     STOP
     REMARK  Move 300 mm above HOLE following a straight line
30   APPROS  HOLE,300
     REMARK  Monitor signal line 3 and call procedure ENDIT
     REMARK  to STOP the program
     REMARK  if the signal is activated during the next motion
     REACTI  3, ENDIT
     APPROS  HOLE,200
     REMARK  Did not feel force, so continue to HOLE
     MOVES   HOLE
```

4.42.1 VAL-II

Alunperin VAL suunniteltiin ohjaamaan liikkeitä ennalta määriteltyjen pisteiden kautta, jolloin tärkeät laskennalliset ominaisuudet, tietotyypit ja antureiden monipuoliset käyttömahdollisuudet puuttuivat. Näiden haittojen poistamiseksi kehitettiin vuonna 1982 VAL-II, joka tarjoaa kehittyneet kommunikointivälineet ja tukee monimutkaisten tehtävien ajantasaista ohjausta.

Pascal

Lähinnä Pascal-kieleen perustuva VAL-II kuuluu rakenteellisiin kieliin. VAL-kielen tietotyyppien lisäksi VAL-II sisältää myös reaaliluvut, loogiset muuttujat, merkkijonot ja taulukot. Lisäksi VAL-II:ssa voidaan käyttää monipuolisia koordinaatistojen muunnosmenetelmiä, joista mainittakoon yhdistetty muunnos. Tällä tarkoitetaan muunnosta, jossa toiseen koordinaatistoon (esim. paletti) määritelty piste kuvataan toisessa koordinaatistossa (esim. robotin tarttuja). Myös erilaisia lausekkeita ja aritmeettisia operaatioita on runsaasti. Näistä mainittakoon +, -, * ja /sekä MOD[46], looginen ja binäärinen tai/ja OR, BOR, AND ja BAND.

Laskennallisia tehtäviä varten on VAL-II:ssa tavanomaiset aritmeettiset ja trigonometriset funktiot. Lisäksi käytettävissä ovat paikan määrittelyssä ja paikkatietojen käytössä tarvittavia funktioita, joista esi-

[46]kokonaisjako modulo

merkkinä koordinaatistomäärittely FRAME, muunnoksen (siirtymän) X-komponentti DX ja kahden pisteen välinen etäisyys DISTANCE. Toiminnan tilaa kuvaavia funktioita ovat liikkeen nopeuden asetus SPEED ja suoritustilaa ilmaiseva STATE. Lisäksi VAL-II:ssa on eräitä laskennallisia ja laitteiston toimintaa ohjaavia funktioita. Viimeksi mainituista esimerkkinä ADC, jolla luetaan tieto AD-muuntimelta sekä käsiohjaimen näppämistön tilaa ohjaava PENDANT -funktio.

Myös kontrollirakenteita on lisätty. Uusia rakenteita ovat FOR ... END, DO ... UNTIL, WHILE ... DO ... END, CASE ... OF ... END, CALL ja RETURN-rakenteet. VAL-II:ssa on edelleen käskyt REACTE ja REACTI, joilla voidaan ulkoisten (esim. antureista tulevien) signaalien avulla ohjata työskentelyä. REACTI-käskyä sovellettaessa keskeytyy ohjelman suoritus välittömästi ja ohjelma siirtyy suorittamaan tiettyä aliohjelmaa. Virheen tapahtuessa ohjelman ajon aikana, voidaan REACTE-käskyllä siirtyä suorittamaan erityistä virheen korjaavaa aliohjelmaa.

VAL-II:n myötä myös robottien liikkeiden ohjaus monipuolistui. Liikkeet voidaan toteuttaa joko interpoloimalla nivelien parametreja siten, että jokaisen nivelen liike tulee suoritettua samassa ajassa tai interpoloimalla tarttujan rata karteesisessa koordinaatistossa. Lähestymis- ja etääntymistilanteissa voidaan käyttää myös rataohjausta, jolloin liikerata on määritelty jatkuvasti. Uusia liikekomentoja ovat muun muassa MOVET ja MOVEST, jotka toteuttavat samat liikkeet kuin komennot MOVE ja MOVES, mutta samalla voidaan liikekomentoon määritellä tarttujan toimintoja. Komentoihin voidaan edelleen liittää liikkeen nopeutta ja tarkkuutta määritteleviä parametreja sekä lisäyksiä rataohjaukseen. Sovelluksissa, joissa liikeradat sisältävät epätarkkuutta voidaan robottien liikkeitä ohjata myös antureiden avulla. VAL-II:ssa tämä toteutetaan käskyllä ALTER.

Yllä esitettyjen tapojen lisäksi voidaan VAL-II:ssa liikkeet toteuttaa matemaattisesti määriteltyjä ratoja pitkin. Toteutus tapahtuu liikekomentojen ja ratalaskennan samanaikaisena suorituksena. Esimerkkinä VAL-II-kielestä esitetään seuraavassa ohjelman osa, jossa robotin työkalua liikutetaan tasaisella nopeudella pitkin ympyrärataa. Rata määritellään keskipisteen ja säteen avulla. Reaalimuuttujat `start`, `last` ja `angle.step` kuvaavat kuljettavaa radan osaa. Reaalimuuttuja `radius` on määritelty radan säteeksi sekä `x.center` ja `y.center` vastaavasti keskipisteen koordinaateiksi. Ratatason määrää `cframe`.

```
FOR angle=start TO  last+start STEP angle.step
     x=radius*COS(angle)+x.center
     y=radius*SIN(angle)+y.center
     MOVE cframe: TRANS(x,y,0,90,-90,0)
END
```

Lähteet: [BS82], [GSCT84], [Loz83], [PL83] ja [SGI84].

4.43 WAVE

VAL

Stanfordissa vuosina 1970 - 1975 kehitetty WAVE-kieli on ensimmäinen yleiskäyttöinen robottien ohjauskieli. Varsinaisesti tutkimuskäyttöön tarkoitettua WAVE-kieltä voidaan pitää uutena kielenä, jonka syntaksi on modifioitu PDP-10 konekielestä. Liikeratojen erillään ohjelmointi tapahtui PDP-10:llä ja ohjelman suoritus eli robotin ohjaus PDP-6:lla. Myöhemmin WAVE-kielestä kehitettiin suppeampi kaupallinen kieli VAL.

Lähtökohtana WAVE-kielessä oli, että liikeradat voidaan periaatteessa suunnitella etukäteen riittävän tarkasti. Poikkeamat liikeradassa olisivat suorituksen aikana pieniä. Tästä syystä WAVE-kieleen sisällytettiin monipuoliset laskennalliset ominaisuudet, jotka puuttuvat vielä nykyäänkin useista kaupallisista kielistä. Käytetyt algoritmit olivat tosin kömpelöitä ja hitaita. Ohjelman suorituksen aikaisia, lähinnä ympäristön epätarkkuuksista johtuvia virheitä pyrittiin vähentämään valvomalla antureilla liikeratoja sekä käyttämällä mukautuvaa ohjausta. Viimeksimainittua ominaisuutta käytetään hyväksi nykyisissä scara-tyyppisissä roboteissa.

koordinaatistot

Heikkotasoisesta syntaksista huolimatta kieli sisältää paljon korkeatasoisia toimintoja. Sitä voidaankin pitää edelläkävijänä useissa robottien ohjelmointijärjestelmien ominaisuuksissa. Näistä mainittakoon erityisesti koordinaatistojen monipuolinen käyttö. WAVE-kielessä on mahdollista määritellä objektikohtaiset koordinaatistot, jotka edelleen voidaan esittää järjestelmän referenssikoordinatistoksi määritellyssä koordinaatistossa. Muita edistyksellisiä ominaisuuksia ovat tarttujan paikan kuvaus karteesisessa koordinaatistossa, nivelten liikkeiden koordinointi[47] sekä joustavat itseohjautuvat asennusliikkeet. Myös antureiden käytöllä on keskeinen sija. Käytettävät anturit ovat pääasiassa erilaisia kosketusantureita mutta valmiudet näköjärjestelmän käytölle ovat myös olemassa.

Heikkoutena voidaan pitää ohjausrakenteiden kömpelyyttä. Ehdottoman hyppykomennon `JUMP` lisäksi käytössä on virhetilan testauksessa käytettävät käskyt `SKIPE A` ja `SKIPN A`. Mikäli virhe numero `A` ilmenee, jätetään `SKIPE A`-käskyllä seuraava käsky suorittamatta. `SKIPN A`-käskyllä seuraavan rivin käsky suoritetaan, jolloin se tavallisesti sisältää jonkin virheenkorjausrutiinin. Ohjelman kulkua voidaan ohjata myös laskureiden avulla testaamalla laskurin sisältöä ja jatkamalla tai siirtymällä annettuun osoitteeseen laskurin sisällöstä riippuen. Tätä menetelmää käytetään melko yleisesti uudemmissakin kielissä.

Esimerkkinä WAVE-kielestä esitetään ohjelma, jossa paletilta noudettu kappale asetetaan reikään [Loz83].

Lähteet: [Loz83] ja [Pau77].

[47] jatkuva tasainen liike

```
  TRANS PIN...          ;Location of pin
  TRANS HOLE...         ;Location of hole

  ASSIGN TRIES 2        ;Number of pickup attempts
  MOVE PIN              ;Move to PIN. MOVE first moves in +Z,
                        ;then to a point above PIN, then -Z.

PICKUP:
  CLOSE 1               ;Pickup pin
  SKIPE 2               ;Skip next instruction if error 2 occurs
                        ;(Error 2: fingers closed beyond arg to CLOSE)
  JUMP OK               ;Error did not occur, goto OK
  OPEN 5                ;Error did occur, open the fingers
  CHANGE Z, -1, NIL, 0, 0       ;Move down one inch
  SOJG TRIES, PICKUP    ;Decrement TRIES, if not negative
                        ;jump to PICKUP
  WAIT NO PIN           ;Print "NO PIN" and wait for operator
  JUMP PICKUP           ;Try again when operator types PROCEED

OK:
  MOVE HOLE             ;Move above hole
  STOP FV, NIL          ;Stop on 50 oz.
  CHANGE, Z, -1, NIL, 0, 0    ;Try to go down one inch
  SKIPE 23              ;Error 23, failed to stop
  JUMP NOHOLE           ;Error did not occur (pin hit sutface)
  FREE 2, X, Y          ;Proceed with insertion by complying
                        ;with forces along x and y
  SPIN 2, X, Y          ;Also comply with torques about x and y
  STOP FV, NIL          ;Stop on 50 oz.
  CHANGE Z, -2, NIL, 0, 0     ;Make the insertion

NOHOLE:
  WAIT NO HOLE          ;Failed
```

4.44 Yamaha

Japanilainen Yamaha valmistaa ohjelmoitavien siirtimien lisäksi scara- ja portaalirobotteja. Eri robottityyppien ohjelmointikielet poikkeavat luonnollisesti hieman toisistaan. Seuraavassa tarkastellaan scara-tyyppisen

Yamaha KO1 Yamaha KO1-robotin ohjelmointijärjestelmää ja kieltä. Yamaha KO1-robottijärjestelmän pääkomponentit ovat robotti, ohjausyksikkö sekä tarvittaessa käsiohjain, kirjoitin ja magneettikuplamuisti. Ohjelmointi tapahtuu ohjausyksikössä olevalla näppäimistöllä. Ohjelmointiosaan kuuluu myös näyttöruutu. Ohjelmat voidaan haluttaessa listata ja tallettaa muistiin.

Ohjelmointijärjestelmän toiminnassa on erotettavissa viisi tilaa. Robotin akseleita voidaan liikutella käsin näppäimistöltä tai käsiohjaimella MANUAL-tilassa. SYSTEM-tilassa tarkistetaan ja asetetaan järjestelmän parametreja, talletetaan ohjelmia muistiin, listataan kirjoittimella jne. PROGRAM-tilassa kirjoitetaan robotin ohjelmat. Ohjelmien kirjoittaminen tapahtuu toimintonäppäimillä kuten ASEAn robottien ohjelmoinnissa. POINT DATA-tilassa määritetään ohjelmassa tarvittavat pisteet. Pisteet annetaan joko kirjoittamalla tai opettamalla liikuttaen robottia haluttuihin pisteisiin. OPERATION-tilassa suoritetaan ohjelmien ajo. Ajo voidaan puolestaan suorittaa askelittain tai automaattisesti.

Lausekkeet, jotka sisältävät vakioita, muuttujia ja operaattoreita, ovat rakenteeltaan samanlaiset kuin tavanomaisessa BASIC-kielessä, jota

BASIC kieli muutenkin paljon muistuttaa. Kielessä on tavalliset aritmeettiset ja vertailuoperaattorit. Myös ohjelman ohjausrakenteet ovat tavanomaiset

```
GOTO nn,
IF <lauseke> THEN nn,
FOR <muuttuja> = <lauseke1> TO <lauseke 2>
        STEP <lauseke 3> ...  NEXT <muuttuja>
```

sekä aliohjelman kutsu CALL nn. Edellä nn tarkoittaa osoitetta.

Liikekomennot ja pistemäärittelyt on Yamaha-kielessä toteutettu seuraavasti: Pistemäärittelyt voidaan tehdä joko antamalla numeerisesti ohjelmointilaitteelta pisteen koordinaatit tai kuljettamalla robotti haluttuun pisteeseen ja tallettamalla robotin asento muistiin. Järjestelmässä voidaan käyttää joko nivelkoordinaatistoa tai xy-koordinaatistoa, jonka origo sijaitsee oletusarvoisesti robotin jalustassa. xy-koordinaatistoa voidaan siirtää ja kiertää xy-tasossa. Näin voidaan määritellä enintään 5 erillistä koordinaatistoa.

Robotin liikkeet toteutetaan MOV tai CP-käskyllä. MOV-käskyllä liike tapahtuu pisteohjatusti, käskyn perään voidaan liittää nopeusmäärite VEL. CP-käskyllä liike on rataohjattua. Esimerkkinä rataohjattu liike pisteeseen P2 nopeudella, joka on 70% oletusarvonopeudesta:

```
CP P2 VEL 70
```

Antureiden sekä oheislaitteiden liitäntää varten on järjestelmässä joukko kaksitasoisia DI/DO-portteja. Portteja voidaan käyttää muuttujina

robotin ohjelmissa, jolloin voidaan esimerkiksi sijoittaa muuttujalle A DI-portin 4, 3 ja 1 bitin tila lauseella:

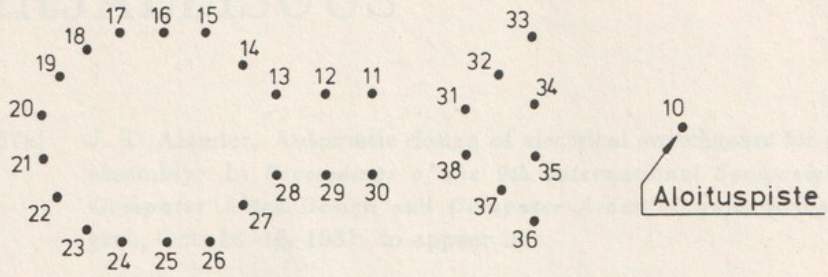
```
A = I bit 4, 3, 1
```

DI-portin tilaa voidaan WAIT-käskyllä käyttää ohjelman ohjaukseen. Esimerkiksi WAIT 15 käskyllä odotetaan kunnes DI-portin tila biteittäin on 00001111. Tämä voidaan toteuttaa myös käskyllä WAIT 00001111B. DO-portin tilaa muutetaan OUT -käskyllä. Esimerkiksi OUT 15 muuttaa DO-portin tilaksi 00001111.

Esimerkkinä Yamaha KO1-robotin ohjelmoinnista esitetään liiman levitysohjelma. Liimaa tiputetaan useaan pisteeseen (kuva 4.9) siten että pisteet 11 - 30 ovat alempana kuin pisteet 31 - 38. Robotti lähtee liikkeelle pisteestä 10 kun DI-portin 7. bitti on 1. DI-portin 2. bitin tila ilmaisee liimaventtiilin tilan, venttiili auki 2. bitti on 1. Liimanlevittimen liikkeistä kertoo 1. ja 0. bitti. DO-portin 7. bitti kertoo virhetilanteesta ja 2. bitillä avataan ja suljetaan liimanlevitin.

Lähde: [Yam84].

```
   "-----MAIN----
 1:WAIT BIT 7=0 "/OFF CHECK/
   WAIT BIT 7=1
   MOV P11 "/START/
   C=11
   CALL 50 "/DOWN, OPEN/       Tip is lowered, valve closes
   A=11 "/START #/             Specification of CP starting
   B=30 "/END  #/              point and point of completion.
   CALL 40                     CP movement subroutine
   C=30
   CALL 60 "/CLOSE, UP/        Valve closed, tip raised
   IF R<>0 THEN 9              Error check
   MOV P31
   C=31
   CALL 50 "/DOWN, OPEN/       Tip lowered, valve opened.
   A=31 "/START #/             Specification of CP starting
   B=38 "/END #/               point and point of completion.
   CALL 40                     CP movement subroutine
   C=38
   CALL 60 "/CLOSE, UP/        Valve closed, tip raised.
   IF R<>0 THEN 9              Error check
   MOV P10                     Movement to initial position
   GOTO 1
 9: "---EMERGENCY---           Error routine
   OUT BIT 7=1
   STOP
   "---SUBROUTINE---
40:"CP                         CP movement subroutine
   FOR E=A TO B
   CP PE
   NEXT E
   RETURN
   "-------
   PO=0 0 1 0 "/Z DOWN/        z -axis data for lowering tip
   P1=0 0 0 0 "/Z UP/          z-axis data for raising tip
50:" DOWN AND OPEN             Subroutine for lowring tip
   PD=PC+PO                    and opening valve
   MOV PD                      Tip descends
   WAIT BIT 0=1
   OUT BIT 2=1 "/OPEN VALVE/   Valve opens
   WAIT BIT 2=1
   RETURN
60:" CLOSE AND UP/             Subroutine for closing valve
   OUT BIT 2=0 "/CLOSE VALVE/  and raising tip.
   TIMER 10
   IF I BIT 2=1 THEN 69
   " OK
```



Kuva 4.9 : Esimerkkiohjelman pisteet.

```
   PD=PC+P1
   MOV PD "/UP/                Tip raises
   WAIT BIT 1=1
   R=O " =STATUS=
   RETURN
69:" NG
   R=1 "  =STATUS=
   RETURN
```

KIRJALLISUUS

[Ala87a] J. T. Alander. Automatic design of electrical switchgears for robot assembly. In *Proceedings of the 9th International Symposium on Computer Aided Design and Computer Aided Manufacturing*, Zagreb, Oct. 14.-15. 1987. to appear in.

[Ala87b] J. T. Alander. *Todella joustavan robotisoidun kokoonpanon uranuurtaja.* Prodeko, Espoo, 1987. (julkaisussa: Kehittyvä ptk-yritys).

[AMBF83] J. S. Albus, C. R. McLean, A. J. Barbera, and M. L. Fitzgerald. Hierarchical control for robots in an automated factory. In *Proceedings of the 13th ISIR/Robots 7*, Chicago, 1983.

[Ame78] *American national standard programming language FORTRAN.* New York, 1978. ANSI X3.9-1978.

[AML86] *AML/2 Manufacturing Control System, User's Quide.* International Business Machines Corporation, 1986.

[AN86] J. T. Alander and K. P. Niemi. Case: sähkökeskuksen kokoonpano robotilla. In *OtaDATA 86 Tietotekniikkapäivä*, 1986.

[AR88] J. T. Alander Automatic design and assembly of fuse boards. In *Proceedings of the 18th I.S.I.R. International Symposium on Industrial Robots*, International Federation of Robotics, Lausanne (Switzerland), 26. - 28. 4. 1988.

[AUT81] R. Abe, S. Ueno, and S. Tsujikado. The programming language for welding robot. In *The Proceedings of the 11th International Symposium on Industrial Robots*, pages 691–698, Tokio, 1981.

[Blu85] C. Blume. Implicit robot programming based on a high-level explicit system. In K. Rathmill, editor, *Robotic Assembly*, pages 231–242, IFS (Publications) Ltd., Bedford (U.K), 1985.

[Bos86] *Bosch SR 800, Robot Manual.* Robert Bosch GmbH, Erbach, 1986. IA-PM/UBN 086 (12/86).

[BR85] H. Brantmark and K.-G. Ramström. *ASEA Off-Line Programming System.* ASEA ROBOTICS, Sweden, 1985.

[Bra86] I. Bratko. *Prolog Programming for Artificial Intelligence.* Addison-Wesley, Wokingham, 1986.

[BS82] S. Bonner and K. G. Shin. A comparative study of robot languages. *IEEE Computer*, 15(12):82–96, 1982.

[DMN68] O. Dahl, B. Myhrhaug, and K. Nygaard. *Simula 67 Common Base Language*. Norsk Regnesentral, Oslo, 1968.

[FVdB82] J. W. Frankling and G. J. Van der Brug. Programming vision and robotics systems with RAIL. In *Proceedings of the Robots VI Conference*, Detroit, 1.-4. March 1982.

[Ger85] *German proposal industrial robot data - "IRDATA" interface*. Technical Report, DIN, 1985. ISO/TC 184/SC 2/GT 4N3.

[Ges83] C. C. Geschke. A system for programming and controlling sensor-based robot manipulators. *IEEE Transaction on pattern analysis and machine intelligence*, PAMI-5(1), January 1983.

[GG85] G. Gini and M. Gini. Robot languages in the eighties. In *Proceedings of the 1st Robotics Europe Conference*, Springer-Verlag, Brussel (1984), 1985.

[GSCT84] W. A. Gruver, B.I. Soroka, J. J. Craig, and T. L. Turner. Industrial robot languages: a comparative evaluation. *IEEE Transaction on Systems, Man and Cybernetics*, SMC-14(4):565–570, 1984.

[Hita] *Hitachi Process Robot JOB Function Manual*. Tokyo. (in English).

[Hitb] *Hitachi Robot JOB Function Manual, työtoiminnot, käyttöohje*. Espoo. (in Finnish).

[IBM84a] *IBM AML/Entry Version 3 User's Guide (8577125)*. International Business Machines Corporation, 1984.

[IBM84b] *IBM AML/Entry Version 4 User's Guide*. International Business Machines Corporation, 1984.

[Jap84] Japanese scene: japanese robot languages. *The Industrial Robot*, March 1984.

[JW74] K. Jensen and N. Wirth. *Pascal Users Manual*. Springer-Verlag, Berlin, 1974.

[Kan84] J. Kanerva. CAD-järjestelmän hyödyntäminen robottisovellutusten suunnittelussa. *Valokynä*, (3):15 – 19, 1984.

[Ket86] Timo Ketonen. *An implementation of a modeler for kinematics*. Raportti TKK-TKO-B61, Teknillinen korkeakoulu, Tietojenkäsittelyoppin laboratorio, Espoo, 1986.

[KR78] B. W. Kernighan and D. M. Ritchie. *The C Programming Language*. Prentice-Hall, Inc., Englewood Cliffs, 1978.

[Lam86] Leslie Lamport. LaTeX: *A Document Preparation System*. Addison-Wesley, Reading, 1986.

[LM81] J.-C. Latombe and E. Mazer. LM: a high-level programming language for controlling assembly robots. In *Proceedings of the 11th International Symposium on Industrial Robots*, pages 683–690, Tokio, 1981.

[Loz83] T. Lozano-Perez. Robot programming. *Proceedings of the IEEE*, 71(7):821–841, 1983.

[LW77] L. I. Lieberman and M. A. Wesley. AUTOPASS: an automatic programming system for computer controlled mechanical assembly. *IBM Journal of Research and Development*, 21:321–333, 1977.

[Muj80] M. S. Mujtaba. Current status of the AL manipulator programming system. In *Proceedings of the 10th International Symposium on Industrial Robots and the 5th International Conference on Industrial Robot Technology*, pages 119–127, Milano, 1980.

[Mur82] H. Murrell. *Ihminen ja kone*. Weilin+Göös, Espoo, 1982. (Republication of: H. Murrell, *Men and Machines*, Methuen & Co, London, 1976).

[NHK*83] M. Nakano, J. Hasegawa, H. Kuno, H. Moribe, T. Ikeda M. Koide, O. Shiroshita, and K. Sukigara. TL-10 a programming language for assembly robots. In *Proceedings of '83 International Conference on advanced Robotics*, Tokyo, 1983.

[Paa85] Pääkirjoitus. *Konepajamies*, (5), 1985. (viite EC:n ennusteeseen: Advanced Manufacturing Equipment in the Community).

[Pau77] R. Paul. Wave: a model-based language for manipulator control. *The Industrial Robot*, Mar. 1977.

[Pek83] V. Pekelis. *Aakkostietoa kybernetiikasta*. Kustannusliike 'Mir', Moskova, 1983.

[PL83] M. Parent and C. Laurgeau. Programming languages, robot technology. In *Logic and programming*, Kogan Page, London, 1983.

[Ran85a] P. G. Ranky. Design and simulation software for flexible assembly systems. In R. Lindholm, editor, *Proceedings of the 4th International Conference on Flexible Manufacturing Systems*, Stockholm, 15-17. October 1985.

[Ran85b] R. Rantala. Styrox-valumalli - nopea vaihtoehto. *Konepajamies*, (5), 1985.

[SGI84] B. E. Shimano, C. C. Geschke, and C. H. Spalding III. VAL-II: a robot programming language and control system. In M. Brady and R. Paul, editors, *Robotics Research*, pages 917–940, the MIT Press, Cambridge, Mass., 1984.

[Sor83] B. I. Soroka. What can't robot languages do? In *Proceedings of the 13th International Symposium on Industrial Robots and Robots 7*, pages 12/1 – 12/8, 1983.

[Tak84] T. Takala. Geometrisen mallittajan sisältävä käyttäjäliitäntä: hajautetun cad-järjestelmän runko. *Valokynä*, (3), 1984.

[Tak87] T. Takala. *Methodological and Structural Principles for Flexible Geometric Modelling in Computer Aided Design.* PhD thesis, 1987.

[Teo87] Teollisuusrobotit Suomessa 1986. 1987. Suomen robotiikkayhdistys ry, jäsentiedote 1/87.

[TH80] T. Toivio and E. Haasiosalo. Täysmuottikaavaus kaasuuntuvalla solupolystyreenillä. *Konepajamies*, (5), 1980.

[Tie85] Tietotekniikan liitto. *Atk-sanakirja, Finnish Dictionary of Information Processing.* Volume 93 of *Tietotekniikan liiton julkaisu*, Gummerus Oy, Jyväskylä, 1985.

[TKA88] J. Tuominen, and J. T. Alander. A model based automatic design and assembly of cable bundles. In *Proceedings of the 18th I.S.I.R. International Symposium on Industrial Robots*, International Federation of Robotics, Lausanne (Switzerland), 26. - 28. 4. 1988.

[TKH*84] M. Tamminen, P. Koistinen, J. Hämäläinen, O. Karonen, P. Korhonen, R. Raunio, and P. Rekola. *BIN: A dimension independent image processing system.* Report HTKK-TKO-C9, Helsinki University of Technology, Laboratory of Information Processing Science, 1984.

[Toi84] *Toisen sukupolven robotit IRB 6/2, IRB 60/2 ja IRB 90S/2.* ASEA, 1984.

[TSM82] R. H. Taylor, P. D. Summers, and J. M. Meyer. AML: a manufacturing language. *The International Journal of Robotics Research*, 1(3):19–41, 1982.

[TUR86] *TURBO PROLOG the natural language of artificial intelligence.* Borland International, Inc., Scotts Valley, 1986.

[Wat86] D. A. Waterman. *A Guide to Expert Systems.* Addison-Wesley, Reading, 1986.

[WF83] B. O. Wood and M. A. Fugelso. MCL, the manufacturing control language. In *The Proceedings of the 13th International Symposium on Industrial Robots and Robots 7*, pages 12/84 – 12/96, 1983.

[WN84] M. Weck and T. Niehaus. Off-line robot programming via standardised interfaces. *The Industrial Robot*, 177 – 179, September 1984.

[WP84] Patrick Henry Winston and Horn Berthold Klaus Paul. *LISP.* Addison-Wesley, Reading, 2 edition, 1984.

[Yam84] *Yamaha Assembly Robot KO1.* Yamaha Motor CO. Ltd, 1984.

SANASTOA ja LYHENTEITÄ

Sanasto on pääosin Atk-sanakirjan [Tie85] mukainen.

ajantasa(järjestelmä)	vastakohta eräjärjestelmä; engl. real time
akku	tietokoneen pää rekisteri; engl. accumulator
algoritmi	laskentasääntö eli -menetelmä
anturi	mittaelin
asiantuntijajärjestelmä	oik. taitotukijärjestelmä
assembly	konekieli (tietotekniikka)
assembly	kokoonpano (muu tekniikka)
binäärinen	kaksiarvoinen
bitti	binäärinumero
Boolean	totuusarvotyyppi (true/false tai on/off)
Boolen algebra	kaksi arvoisen logiikan algebra
CAD (Computer Aided Design)	tietokoneavusteinen suunnittelu
CADCAM	CAD:n ja CAM:n yhdiste
CAM (Computer Aided Manufacturing)	tietokoneavusteinen valmistus
CIM (Computer Integrated Manufacturing)	tietokoneen avulla integroitu valmistus
DI (Digital Input)	digitaalinen sisäänmenoportti
DO (Digital Output)	digitaalinen ulostuloportti
flag	lippu; totuusarvoinen (on/off) muuttuja
FMS (Flexible Manufacturing System)	joustava tuotantojärjestelmä
heksadesimaali(luku)	kuusitoistajärjestelmän luku
heuristinen	intuitioon ja kokeilun perustuva
indeksi	järjestysnumero (esim. taulukon alkiota osoittava)
IO (Input Output)	syöttö ja tulostus
juovakoodi	merkkien esitys juovina (engl. bar code)
kiikku	alkeismuistipiiri
kybernetiikka	ohjaustiede (koneet eliöt)
levyke	(taipuisa) magneettilevytaltio (engl. diskette)
lippu	(engl. flag) totuusarvoinen (on/off) muuttuja
liukuluku	eng. floating point number
off-line	erillinen (tietokoneesta tai robotista)

oktaali(luku)	kahdeksanjärjestelmän luku
on-line	kytketty (tietokoneeseen tai robottiin)
PC (Personal Computer)	henkilökohtainen (mikro)tietokone
proseduuri	aliohjelma, toimenpidesarja
RAM (Read And Write Memory)	lukipuolijohdemuisti
rekursiivinen	itseensä palautuva
robotti	ihmisen työtapaa jäljittelevä ohjelmoitava lait
ROM (Read Only Memory)	vain luettavissa oleva puolijohdemuisti
sana	tietokoneen luonteva muistiyksikkö (tavallisesti 8-64 bittiä)
semantiikka	merkitys (kielen)
servo	yleensä paikkatakaisinkytketty sähkömoottori
simulointi	jäljittely (esim. tietokoneella)
syntaksi	rakenne (kielen)
säätö	ohjaus palautteen avulla
tavu	tavallisesti yksi merkki eli kahdeksan bittiä
veräjä	logiikkapiiri

HAKEMISTO

A12 Allegro Robot System 74
Abe, R. 89
ACMA-robotti 80
aggregaatti (lista) 58
AGX-ohjelmisto 43
AL (Arm Language)-kieli 27, 28, 29, 31, 55, 81, 106
Albus, J.,S. 41
Algol-kieli 25
Allegro-robotti 82
AML (A Manufacturing Language) 27, 29, 58
AML/2 63
AML/Entry -kieli 29, 30, 31, 58, 60, 63
ANORAD-kieli 27, 64
anturi 13
APL-kieli 76
APT-kieli 26, 82, 95, 96
ARL-kieli 29, 64
ARLA (ASEA Robot Language) -kieli 29, 65
ASEA 11, 13, 29, 65
asiantuntijajärjestelmä 39
AUTOMATIX Inc. 93
AUTOPASS (Automated Parts Assembly System) -kieli 28, 69

BAPS-kieli 71
BARRAS-Provence 77
BASIC-kieli 25, 76, 81, 99, 108
Bendix Corporation 107
BIN-mallittaja 46
Biomatic 88
Blume, C. 55, 106
Bonner, S. 55, 64, 73, 74, 81, 82, 99, 110, 111
Bosch-robotti 71
Brantmark, H. 65
Bratko, Ivan 37, 92

CAD (Computer Aided Design) 22, 43
CAD-malli 24, 37, 44, 46
CAD-ohjelmointi 24
CAM-I (Computer Aided Manufacturing - International) 24
Cincinnati Milacron 110
Cincinnati T3 -robotti 82

DEA 93

Electrotechnical Laboratory 105
EMILY-kieli 27, 73

Fortran-kieli 25, 73, 99
Franklin, J., W. 93
FUNKY-kieli 25, 73

General Electric Company 74
Geschke, C., C. 100
Gini, G. 76, 88, 95, 106
GIXI 99
Grafcet-kieli 80
Gruver, W., A. 55, 74, 75, 82, 93, 99, 111

Haasiosalo, E. 43
hahmontunnistus 8
HELP-kieli 27, 29, 30, 74
hiekkapuhallus 16
hionta 16

Hitachi 29, 64, 76, 93
hitsaus 16, 29

I/O -portit 29
IBM 7547-robotti 43
IBM 21, 22, 28, 58, 63, 69, 73, 73, 81, 86
ICAM-projekti 82
ihminen 8
Illinoisin yliopisto 100
IMAG 77, 106
IRDATA-välikieli 74, 71
IRIA 76
IRL (Intuitive Robot Language) -kieli 75

JARS-kieli 75
Jet Propulsion Laboratory 75
JIRA (Japanese Industrial Robot Association) 107
JOB-kieli 76
johdattaminen 14, 20
järeä robotti 10

kaarihitsaus 89, 93
Karlsruhen yliopisto 106
Ketonen, Timo 36
kiinteä automaatio 9
kokoonpanorobotti 10
kokoonpano 16, 16, 69, 75, 95, 106
Komatsu-RW-robotti 89
Komatsu 89
koordinaatisto 12, 36
kuljetin 15, 31
käsiohjain 19

lakkaus 16
LAMA-S 76
Latombe, J.-C. 77
leikkuu 16
Lieberman, L., I. 69
liimaus 116
LISP-kieli 25, 85, 99, 105, 58
LM (Langage de Manipulation) -kieli 77
Lozano-Perez, T. 55, 74, 76, 77, 82, 84 85, 86, 87, 93, 106, 107, 111, 114
LPR (Langage de Programmation de Robot) 80

maailman malli 25
maalausrobotti 10, 20
maalaus 16
Machine Intelligence VS-100 99
magneettitarttuja 13
MAL (Multipurpose Assembly Language) -kieli 81
MAP (Manufacturing Automation Protocol) 30
MAPLE-kieli 27, 81
matriisi 36
MCL (Manufacturing Control Language) -kieli 82, 27, 29, 29
mekanismit 38
mekanismi 36
MH-1 -robotti 84
MHI (Mechanical Hand Interpreter) -kieli 84
Microbo 75
Microrobot-robotti 88
MINI -kieli 85, 85
MIT (Massachusetts Institute of Technology) 84
ML-kieli 73, 86
Mujtaba, M., S. 55
Murrel, H. 8
muunnos 36
muuraus 47

Nakano, M. 108
NASA 75
NC-ohjelmointi 24
näköjärjestelmä 29, 93, 99, 100, 105

ohjausjärjestelmä 14
ohjausyksikkö 7
ohjaus 7
ohjelmoinnin vaiheet 22

ohjelmointikielet 19
Algol 25
APL 76
BASIC 25, 76, 81, 99, 108
Fortran 25, 73, 99
LISP 25, 85, 99, 105, 58
Pascal 25, 75, 77, 88, 93, 106
PL/1 25, 81
Prolog 37, 92, 26
ohjelmointilaite (ASEA) 65
ohjelmoitava logiikka 14
Olivetti 81, 106
ongelmanratkaisu 39
optimointi 41, 46, 95

PACS -ohjausjärjestelmä 107
PAL-kieli 27, 87
paletointi 60, 106
paletti 31
Parent, M. 55, 75, 77, 80, 89, 93, 99, 106, 106, 111
Pascal-kieli 25, 75, 77, 88, 93, 106, 112
PASRO (Pascal for Robots) -kieli 88
Paul, R. 114
Pekelis, V. 9
perusliikkeiden tason kielet 25
peruutus (backtracking) 47
PETL-kieli 105
pisteestä pisteeseen tason kielet 25
pisteohjaus 14
PL/1-kieli 25, 81, 69
PLAW (Programming Language for Arc Welding) -kieli 89
pneumaattinen tarttuja 13
polystyreeni (Styrox) 43
POP-2 -kieli 107
PRIMA-kieli 76
prioriteetti 31
Prolog-kieli 37, 92
prosessirobotti 10
PUMA-robotti 27, 99, 111

RAIL-kieli 29, 93
rakentaminen 45
rakenteisen tason kielet 27
Ranky, P., G. 38
Rantala, R. 43
RAPT (Robot APT) -kieli 95
rataohjaus 14
RCL-kieli 27
Renault 80
revolveritarttuja 13
rinnakkaisuus 40
ROBEX-kieli 96
ROBEX-X -kieli 96
ROBITRON 77
Robot Institute of America 9
robottikielien tasot 27
robotti 8
ACMA- 80
Allegro- 82
Bosch- 71
Cincinnati T3 - 82
IBM 7547- 43
, järeä 10
kokoonpano- 10
Komatsu-RW- 89
maalaus- 10, 20
MH-1- 84
Microrobot- 88
prosessi- 10
PUMA- 27, 99, 111
Sigma- 106
Supersigma- 81
teollisuus- (määritelmä) 9
Toyota- 108
Yamaha KO1- 116
ROL (RObot Language) -kieli 99
rotaatio 36
RPL-kieli 29, 99
RSS (Robot Servo system) -kieli 100, 29
RVL/A -kieli 105

Sankyo 106
scara 60, 63
SCEMI 77
SERF (Sankyo Easy Robot Formula) -kieli 106
servo 14
Shimano, B., E. 111
SIGLA (SIGma LAnguage) -kieli 106, 27

Sigma-robotti 106
SIGNAL 55
simulaattori 43
simulointi 15, 38, 43
SRI International 99
SRL (Structured Robot Language) -kieli 106
Stanfordin yliopisto 114
STROL-kieli 107, 29
STROLIC-välikieli 107
Suomen Robotiikkayhdistys ry 9, 17
Supersigma-robotti 81
syötin 15

T3-kieli 110, 25
Takala, Tapio 43
tarttuja 12
tartuntaleveys 29
Taylor, R., H. 58
TEACH 107
tehtävänläheinen kieli 28
tekoäly 106, 39, 9
teollisuusrobotti (määritelmä) 9
TL-10 (Toyota Robot Language - 10) -kieli 108, 29
Toivio, T. 43
Toyota-robotti 108
Toyota 29
translaatio 36
tuotannon suunnittelu 37
tuotantosolu 41
tuotemalli 38, 39
tuotesuunnittelu 37
tuotteen suunnittelu 37
työkalujen käyttö 29
törmäykset 31
törmäystarkastelu 40

Unimate Inc. 111

vaaputus 65
VAL (VicArm Language) -kieli 111, 114, 27, 29, 76, 95
valaminen 16, 43
valumalli 43
VAL-II -kieli 111, 31, 112
vammaiset 76
vapausasteet 12
Viita, Lauri 35
välivarasto 15

Waterman, D., A. 39
WAVE-kieli 111, 114
Weck, M. 74, 96
Wood, B., O. 82

Yamaha KO1 -robotti 116
Yamaha 116, 29
ympäryslaitteet 15

TAVOITTEET KORKEALLA.

Tiedät mitä tahdot. Olet asettanut tavoitteesi. Tiedät myös, mitä tavoitteiden saavuttaminen edellyttää: kovaa työtä ja hyviä apuvälineitä.

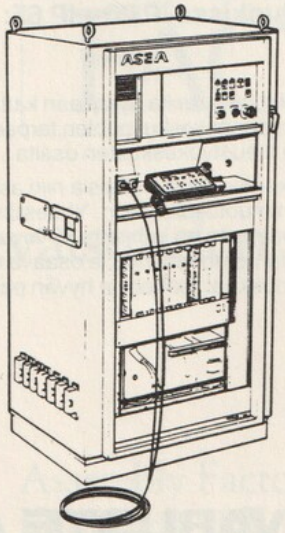
Tietotekniikka on yksi tämän päivän monipuolisimmista apuvälineistä – niin opiskelussa kuin työelämässäkin.

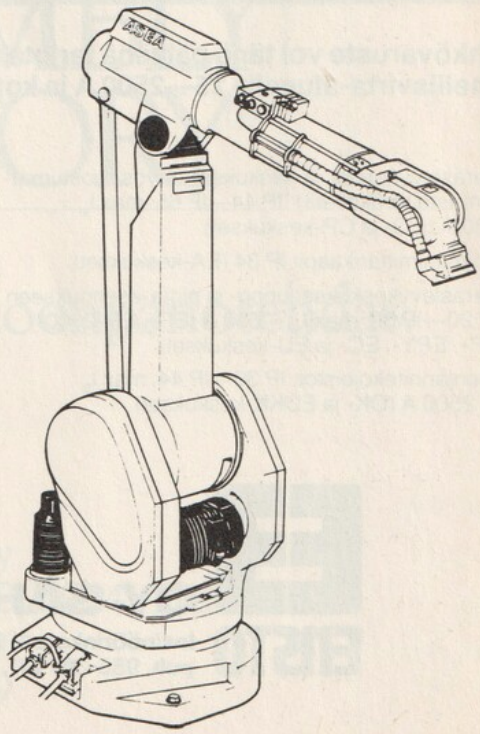
IBM osaa tietotekniikan mahdollisuudet. Tarjoamme ratkaisuja, jotka palvelevat sinua nyt ja tulevaisuudessa.

RATKAISU ON

IBM

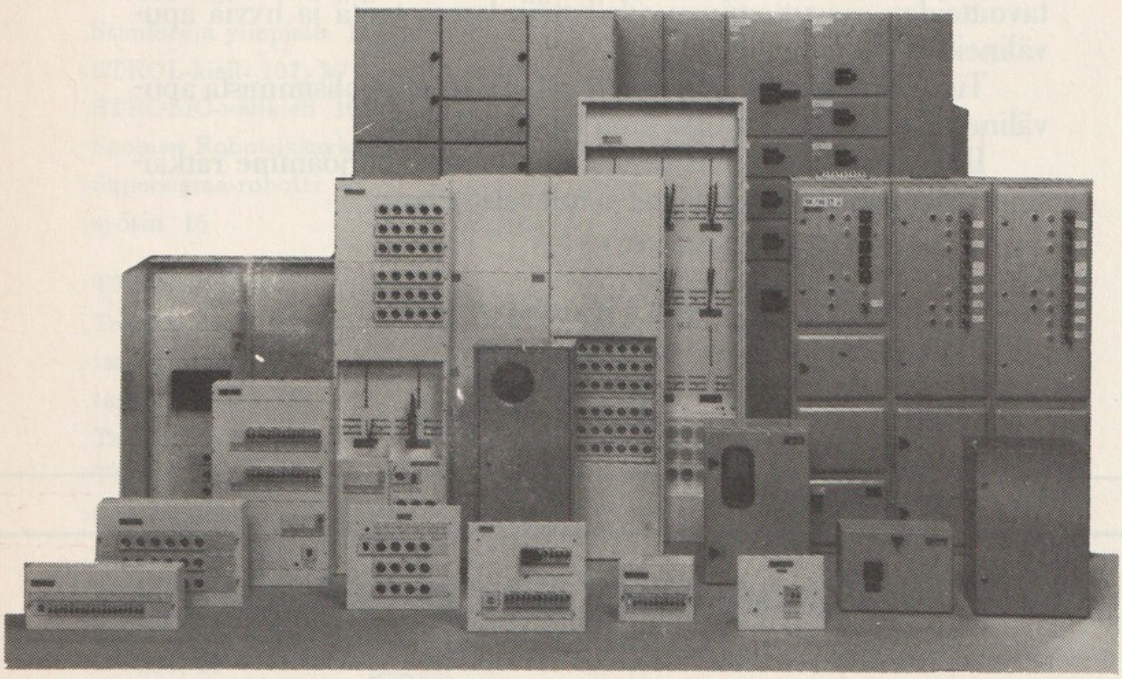
STRÖMBERG TEOLLISUUS





Strömberg Oy
PL 10
02611 ESPOO
Puh. (90) 59 111
Telex 124709
Telekopio (90) 596363

Oy Sähkövaruste Ab:n jakokeskusvalikoima



Sähkövaruste voi tänä päivänä tarjota 11 eri jakokeskusrakennetta nimellisvirta-alueella 25—2500 A ja kotelointiluokissa IP 20—IP 65:

- Teräslevykotelot ja keskukset (myös ruostumattomasta teräksestä): IP 44—IP 65: max I_e = 630 A (EO- ja CP-keskukset)
- Alumiinimittarikaapit IP 34 (EA-keskukset)
- Teräslevykeskukset uppo- ja pinta-asennukseen IP 20—IP 30: max I_e = 630 A (ER-. ER-2000-. EP-. EPY-. EC- ja EU-keskukset)
- Pienjännitekojeistot: IP 30—IP 44: max I_e = 2500 A (OK- ja EDKK-keskukset)

Näillä rakenteilla pystytään kattamaan lähes kaikki pienjännitekeskuspuolen tarpeet sekä vakio- että tilaustyökeskusten osalta.

Rakenteet ajanmukaisia niin asennettavuudeltaan kuin muotoiltaankin. Yhdestätoista rakenteesta kahdeksan on kehitetty 80-luvulla. Kehittyneet tuotantomenetelmät ja osaava henkilöstö takaavat laadukkaat tuotteet ja hyvän palvelun.

ENSTO

OY SÄHKÖVARUSTE AB

Insinöörinkatu 1, SF-50100 MIKKELI 10
puh. 955-367 211, telex 55159 svea sf

TEKNIIKAN TIETOKIRJAT OTAKUSTANTAMOSTA

OTAKUSTANTAMO

Oy Dipoli Ab, Jämeräntaival 3 A, 02150 Espoo, puh. 468 2178
Kirjakauppa: Otakaari 1 F, 02150 Espoo, puh. 468 2160

ASSEMBLY FACTORY®

Tulevaisuuden kokoonpanotehdas.

Assembly Factory Oy
Kaupintie 11 B, PL 30, 00441 Helsinki
Puhelin (90) 562 4566
Telefax (90) 562 4739

Art
of
CAD

all about
CAD
and CIM

call
(90)599299

Kirja on ensimmäinen robottien rakennetta, soveltamista, ohjausta, mallittamista, ohjelmointia ja ohjelmointikieliä käsittelevä suomenkielinen teos. Kirjassa esitetään yhteenveto noin 50 robottien ohjelmointikielestä. Esiteltäviin kieliin sisältyy sekä kaupallisia että tutkimusasteella olevia kieliä.
Kirja on tarkoitettu henkilöille, jotka ovat tai joutuvat tekemisiin robottijärjestelmien ja tuotantoautomaation kanssa esimerkiksi suunnittelu- ja rationalisointitehtävissä. Lukijoilta edellytetään ohjelmoinnin perusteiden hallintaa (esim. Pascal-kieli). Aiempi kokemus roboteista ei ole välttämätön.

OTAKUSTANTAMO
Oy Dipoli Ab, Jämerantaival 3 A, 02150 Espoo, puh. 468 2178
Kirjakauppa: Otakaari 1 F, 02150 Espoo, puh. 468 2160

TEKNILLISEN KORKEAKOULUN KIRJASTO OTA
120 00 42522

ISBN 951-672-028-5