TAMPEREEN AMMATTIKORKEAKOULU
Kone- ja tuotantotekniikka
Modernit tuotantojärjestelmät

Tutkintotyö

Joni Nieminen 25.4.2005

PARAMETRIOHJELMOINTI FANUC 21i-MB –OHJAUKSESSA

Työn ohjaaja Ins. Tomi-Pekka Nieminen (AMK)
Valvojana Lehtori Markku Nieminen, Tampereen ammattiopisto
Tampere 2005

TAMPEREEN AMMATTIKORKEAKOULU
Kone- ja tuotantotekniikka
Modernit tuotantojärjestelmät

Nieminen, Joni Parametriohjelmointi FANUC 21i-MB –ohjauksessa

Tutkintotyö 49 sivua + 15 liitesivua
Työn ohjaaja Ins. Tomi-Pekka Nieminen (AMK)
Työn valvojana Lehtori Markku Nieminen
Toukokuu 2005
Hakusanat fanuc, ohjelmointi, parametri, makro

TIIVISTELMÄ

Opinnäytetyössä perehdytään Fanuc 21i-MB –ohjauksen parametriohjelmointiin käymällä läpi ohjelmointiin kuuluvat parametrit ja niiden käyttö esimerkkien avulla.

Tarkoituksena oli luoda parametriohjelmoinnin opetukseen soveltuva materiaali, jossa asia opetetaan esimerkkien avulla ja annetaan mahdollisuus soveltaa opinnäytetyössä käsiteltyjä esimerkkejä omien ohjelmien kirjoittamisessa.

Työn tuloksena syntyi joukko parametriohjelmia, joiden toimivuus on testattu NC-työstökoneella. Ohjelmat antavat perusteet ja mallin parametriohjelmien itsenäiselle valmistukselle. Materiaali sopii ammattikorkeakoulun, ammattioppilaitoksen sekä teollisuuden NC-ohjelmoijien ja koneistajien koulutukseen.

TAMPERE POLYTECHNIC
Mechanical and Production Engineering
Modern Production Systems
Nieminen, Joni Custom Macro Programming in Fanuc 21i-MB CNC Control
Engineering Thesis 49 pages, 15 appendices
Thesis Supervisor Tomi-Pekka Nieminen (Eng)
Supervisor Markku Nieminen
May 2005
Keywords fanuc, programming, parameter, custom macro

ABSTRACT

In this Thesis we concentrate on custom macro programming with Fanuc 21i-MB CNC control by going trough the basic parameters involved in macro programming with example programs.

The main purpose was to create teaching material that goes trough basics custom macro programming with examples and give means to create your own custom macros.

As a result we had a group of custom macros which have been tested on CNC-machines. Macros give the basics and example for independent creation of own custom macros. This thesis can be used as a teaching material for example in vocational schools, polytechnics and in industry to train CNC programmers and machine operators.

SISÄLLYSLUETTELO

1 JOHDANTO

Tämä opinnäytetyö käsittelee parametri- eli muuttuja- tai makro-ohjelmointia Fanuc 21i-MB -ohjauksessa. Työ on tarkoitettu parametriohjelmoinnin perusteiden opetteluun malliohjelmien avulla. Työssä ei käsitellä Fanuc-koodikielen perusohjelmointia, vaan se on tarkoitettu henkilöille, jotka osaavat entuudestaan Fanuc-koodikielen tai vastaavan perusteet ja kykenevät itsenäisesti kirjoittamaan ohjelmia sen avulla. Parametriohjelmointi vaatii myös hyvää matemaattista hahmotuskykyä ja perusymmärrystä matematiikan laskusäännöistä. Myös ne on hyvä kerrata ennen tämän opinnäytetyön lukemista.

Opinnäytetyö pyrkii antamaan tietoa siitä, kuinka ohjelmia voidaan lyhentää tekemällä parametriohjelmoinnilla makroja eli työkiertoja. Työssä käsitellään aluksi parametreihin liittyvä teoria, jonka jälkeen käydään neljän esimerkin avulla läpi parametriohjelmoinnin perusrakenne ja toimintatapa. Näitä esimerkkejä soveltaen lukija kykenee laatimaan omiin tarpeisiinsa soveltuvia ohjelmia.

2 PARAMETRIOHJELMOINTI

Parametriohjelmoinnilla tarkoitetaan NC-ohjelmointia, jossa tietyt arvot on korvattu parametreilla eli muuttujilla, jotka voivat eri tilanteissa saada eri arvoja. Sitä kutsutaan myös muuttujaohjelmoinniksi.

Parametriohjelmia käytetään pääsääntöisesti ohjelmoimaan usein toistuvia muotoja ja työvaiheita. Tällaisia ovat muun muassa poraukset, poterot ja reikäpiirit.

Parametriohjelma on käytöltään hyvin samantyyppinen aliohjelman kanssa. Aliohjelma sisältää kiinteän ohjelman, joka on aina samanmuotoinen ja suorittaa samat liikkeet. Parametriohjelman suorittama muoto ja liikkeet taas määräytyvät ennalta määriteltyjen muuttujien perusteella ja voivat olla joka kerta erilaiset. Parametrialiohjelma ja normaali aliohjelma erotetaankin jo nimessä. Parametrialiohjelmaa kutsutaan makroksi.

3 MUUTTUJAT /1/

Normaalissa ohjelmoinnissa liikekäskyt ohjelmoidaan antamalla G-koodille ja liikematkalle lukuarvot, esimerkiksi G0 X100. Parametriohjelmassa voidaan lukuarvot tarvittaessa korvata muuttujalla. Taulukossa 1 on esimerkki muuttujan käytöstä ohjelmassa.

TAULUKKO 1. Parametrien käyttö ohjelmassa

```
#1=100;
G0 X#1;
```

3.1 Muuttujien määrittely

Fanuc-ohjelmointikielessä muuttujat merkitään #-merkillä, jonka perään tulee käytettävän muuttujan numero, esimerkiksi #1. Muuttujan numero voidaan määrittää myös matemaattisella lausekkeella, jolloin se tulee kirjoittaa hakasulkeiden sisään, esimerkiksi #[#1+#2-3].

Valtaosa NC-koneen sisäisistä toiminnoista toteutetaan erilaisten parametrien avulla. Onkin siis tärkeää tietää, mitä parametreja voidaan käyttää.

TAULUKKO 2. Parametrien jaottelu /1, s. 285/

Parametrin numero	Muuttujan tyyppi	Toiminto
#0	Aina tyhjä	Tämän muuttujan arvo on aina tyhjä eikä sille voi määrittää muuta arvoa.
#1 – #33	Paikalliset muuttujat	Paikallisia muuttujia voi käyttää makrojen sisällä vain tiedon tallentamiseen. Virran katkaisun yhteydessä näissä muuttujissa olevat tiedot häviävät. Makron lähtöarvot tallennetaan näihin muuttujiin.
#100 – #149 (#199) #500 – #531 (#999)	Yleismuuttujat	Yleismuuttujia voidaan käyttää eri makrojen välillä. Virran katkaisun yhteydessä muuttujat 100–149 tyhjenevät, mutta muuttujat 500–531 säilyttävät arvonsa. Koneeseen on mahdollisuus saada optiona muuttujat 150–199 ja 532–999.
#1000 –	Järjestelmä-muuttujat	Järjestelmämuuttujia käytetään erilaisten järjestelmätietojen lukemiseen ja määrittämiseen, esimerkiksi terän paikka- ja kompensointitiedot.

Parametrit jaotellaan neljään eri ryhmään taulukon 2 mukaan.

Muuttujien arvoksi voidaan määrittää arvo 0 (nolla) tai mikä tahansa luku seuraavissa rajoissa

$-10^{47} – -10^{-29}$

$10^{-29} – 10^{47}$

Mikäli muuttujan arvo ei sovi näihin arvoihin, ilmestyy koneen ohjaukseen hälytysilmoitus asiasta.

Vaikka koneen normaalissa ohjelmoinnissa lukuarvojen perään täytyisikin ohjelmoida piste, voidaan se muuttujan arvon määrittelyssä jättää pois. Esimerkiksi ohjelmoitaessa #1=123 muuttujan #1 arvoksi tallentuu 123.000.

3.2 Muuttujien käyttö

Parametriohjelmoinnissa korvataan koodin lukuarvo tarvittaessa muuttujalla, johon on tallennettu haluttu arvo (taulukko 1). Muuttuja voidaan kirjoittaa myös matemaattisella lausekkeella, jolloin se kirjoitetaan hakasulkeisiin. Muuttujaan tallennettu arvo pyöristyy automaattisesti käyttökohteen mukaiseen tarkoitukseen.

Esimerkki: Kone paikoittaa terän 1/1000 mm tarkkuudella. Parametriin #1 on tallennettu arvo 12.3456. Tällöin komento G0 X#1 toteuttaa komennon G0 X12.346.

Mikäli muuttujassa on eri etumerkillä varustettu arvo, sen voi muuttaa laittamalla muuttujan eteen miinusmerkin (-), esimerkiksi G0 X-#1. NC-ohjelmassa arvot nolla ja tyhjä ovat eri asia. Tämä pätee myös parametriohjelmoinnissa. Jos komennossa kutsutaan muuttujaa, jonka arvo on tyhjä, jättää kone sen huomioimatta. Taulukossa 3 nähdään esimerkki tyhjän muuttujan käytöstä.

TAULUKKO 3. Tyhjän muuttujan käyttäminen

#1=0; #2=<tyhjä> (eli sama kuin #0) G0 X#1 Y#2; ⇔ G0 X0.;

Muuttuja #0 on aina tyhjä. Sitä ei voi muuttaa, mutta sen voi lukea. Muuttujia voi käyttää minkä tahansa koodin kanssa lukuun ottamatta seuraavia tapauksia:

- Ohjelmanumero O#1;
- Vaihenumero /#2 G0 X100.;
- Lausenumero N#3 Y200.;

3.3 Järjestelmämuuttujat

Järjestelmämuuttujat ovat parametriohjelmoijan ainoa tietolähde koneen tilasta. Toimiva parametriohjelma ei ole riippuvainen pääohjelmassa ohjelmoiduista koodeista, mutta sen täytyy palauttaa koneelle sama tila, kuin missä se oli ennen makrokutsua. Ohjelma kohtaisesti käytössä olevat koodit voivat olla erilaisia, joten ainoa keino selvittää ne on tarkistaa käytössä olevat koodit järjestelmä parametreista. Tarkemmat tiedot parametrien selvityksistä ovat liiteessä 1.

4 OHJELMANAJON HALLINTA /1/

Tietyissä tilanteissa ohjelman toteutus yhtenä kokonaisuutena on ohjelman toiminnan kannalta tärkeää. Hyvä esimerkki tällaisesta tilanteesta on kierteitystyökierto. Siinä on tärkeää, ettei käyttäjä pysty vaikuttamaan kierron aikana syöttöön tai ajamaan kiertoa lauseittain läpi. Parametri #3003 ohjaa lauseittain ajoa ja M-koodien toteuttamista taulukon 4 mukaan

TAULUKKO 4. Parametrin #3003 arvojen vaikutus ohjelmaan

#3003 arvo	Lauseittainajo	M-koodien tarkkailu
0 (Oletus)	Käytössä	Odotetaan toteutumista
1	Ei käytössä	Odotetaan toteutumista
2	Käytössä	Ei odoteta toteutumista
3	Ei käytössä	Ei odoteta toteutumista

#3003 arvo koneen käynnistymisen jälkeen on 0, eli yksittäislauseajo on mahdollinen ja M-koodien toteutuminen tarkistetaan ennen seuraavaa lausetta (esimerkiksi M3, kara on käynnistynyt).

Parametri #3004 puolestaan ohjaa ohjelman pysäytysnapin, syötön säätökytkimen ja tarkan pysähtymisen toteuttamista taulukon 5 mukaan.

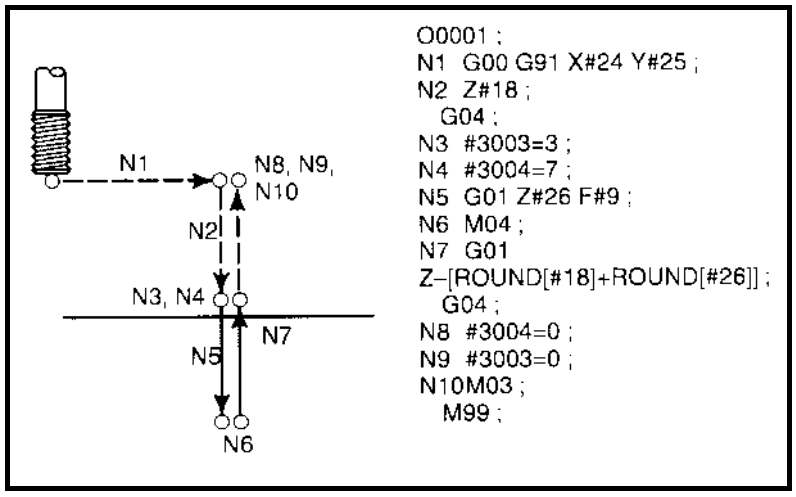
TAULUKKO 5. Parametrin #3004 arvojen vaikutus ohjelmaan

#3003 arvo	Ohjelman pysäytys	Syöttökytkin	Tarkka pysäytys
0 (Oletus)	Toteutetaan	Noudatetaan	Toteutetaan
1	Ei toteuteta	Noudatetaan	Toteutetaan
2	Toteutetaan	Ei noudateta	Toteutetaan
3	Ei toteuteta	Ei noudateta	Toteutetaan
4	Toteutetaan	Noudatetaan	Ei toteuteta
5	Ei toteuteta	Noudatetaan	Ei toteuteta
6	Toteutetaan	Ei noudateta	Ei toteuteta
7	Ei toteuteta	Ei noudateta	Ei toteuteta

Kun ”ohjelman pysäytys” vaihtoehto on ’Ei toteuteta’:

1. Kun *Ohjelman pysäytys* -nappia pidetään pohjassa kone pysähtyy yksittäislauseajon pysähdystilaan, mikäli yksittäislauseajoa ei ole kytketty pois parametrilla #3003.
2. Jos *Ohjelman pysäytys* -nappia painetaan ja sitten vapautetaan, syttyy napin merkkivalo, mutta kone ei pysähdy ennen seuraava lausetta, jossa parametrin #3004 arvo sallii ohjelman pysäyttämisen.

Kuvassa 1 nähdään tyypillinen esimerkki parametrin #3003 käytöstä.



KUVA 1. Parametrien #3003 ja #3004 käyttö ohjelmassa

5 OPERAATTORIT /1/

Muuttujia käsitellään NC-ohjelmassa eri tavoin. Muuttujan arvoa voidaan muuttaa matemaattisten ja loogisten operaattoreiden avulla ja sen suhdetta toisiin muuttujiin tai lukuihin voidaan verrata vertailuoperaattoreiden avulla. Operaattoreiden käyttö on oleellinen osa toimivan parametriohjelman kirjoittamista.

5.1 Matemaattiset ja loogiset operaattorit

Muuttujia voidaan käsitellä erilaisilla matemaattisilla funktioilla. Nämä on listattu taulukossa 6. Taulukossa merkintä #i voi olla joko muuttuja tai lauseke, ja merkinnät #j ja #k voivat olla joko muuttujia tai lukuarvoja. Matemaattiset ja loogiset operaattorit on esitelty taulukossa 6.

TAULUKKO 6. Matemaattiset ja loogiset operaattorit

Toiminto	Merkintä	Huomiot
Määritys	#i=#j;	
Yhteenlasku	#i=#j+#k;	
Vähennyslasku	#i=#j-#k;	
Tulo	#i=#j*#k;	
Osamäärä	#i=#j/#k;	
Sini	#i=SIN[#j];	Kulmat ovat asteina.
Arkussini	#i=ASIN[#j];	Eli 90°30’ on 90.5°.
Kosini	#i=COS[#j];	
Arkuskosini	#i=ACOS[#j];	
Tangentti	#i=TAN[#j];	
Arkustangentti	#i=ATAN[#j]/[#k];	
Neliöjuuri	#i=SQRT[#j];	Pyöritykset tapahtuvat kokonaislukuihin.
Itseisarvo	#i=ABS[#j];	#i=ROUND[1.6789]=2
Pyöristys	#i=ROUND[#j];	#i=FIX[1.6789]=1
Pyöristä alas	#i=FIX[#j];	#i=FUP[1.6789]=2
Pyöristä ylös	#i=FUP[#j];	ROUND toiminto NC-koodissa:
Luonnollinen logaritmi	#i=LN[#j];	X[ROUND[1.6789]=X1.679
10-kantainen logaritmi	#i=EXP[#j];	
OR	#i=#jOR#k;	Loogiset operaattorit verrataan binäärimuotoisina bitti bitiltä.
XOR	#i=#jXOR#k;	
AND	#i=#jAND#k;	

ROUND-funktio pyöristää luvut tasalukuihin, paitsi jos sitä käytetään NC-osoitteen (esimerkiksi koordinaatin) ilmaisemiseen, jolloin se pyöristyy osoitteen tarkkuuteen. Esimerkiksi #1=ROUND[1.2345] antaa arvon #1=1.235.

Määritettäessä jotakin yllä olevista operaattoreista NC-ohjelmassa voidaan kirjoittaa koko operaattori tai vain kaksi ensimmäistä kirjainta, esimerkiksi ROUND tai RO. Kuten matematiikassa yleensäkin, voidaan laskutoimitusten järjestys määrittää sulkeiden avulla.
HUOMIO! Käytä [hakasulkeita], kaarisulkeilla ilmoitetaan (ohjelmahuomautukset).

5.2 Vertailuoperaattorit

Yksi parametriohjelmoinnin oleellisimmista vaiheista on muuttujien vertaaminen ja toimintojen teko vertaustuloksen mukaan. Tähän tarvitsemme vertailuoperaattoreita, jotka ovat taulukossa 7.

TAULUKKO 7. Vertailuoperaattorit

Operaattori	Matemaattinen	Merkitys
EQ	$=$	Yhtä suuri kuin
NE	$\neq$	Eri suuri kuin
GT	$>$	Suurempi kuin
GE	$\geq$	Suurempi tai yhtä suuri kuin
LT	$<$	Pienempi kuin
LE	$\leq$	Pienempi tai yhtä suuri kuin

Operaattorit on ohjelmassa kirjoitettava hakasulkeisiin, esimerkiksi [#1 EQ #2].

6 MUUTTUJIEN VIRHEET /1/

Lukuarvojen puute on ehkä suurin parametriohjelmoinnin vaikeus. Ohjelmoitsija ei kirjoita ohjelmaan lukuarvoa, johon koneen tulisi mennä, vaan hänen tulee kyetä päättelemään, minkä arvon kukin muuttuja saa ohjelman eri kohdissa. Tämä voi olla oletettua vaikeampaa, sillä NC-ohjauksen tapa suorittaa laskut ja toimenpiteet aiheuttaa arvoihin erilaisia virheitä.

6.1 Binäärilaskenta

Koska ohjaus suorittaa kaikki laskutoimitukset binäärilukuina, siitä muodostuu ohjelmoijalle ongelma. Binäärilukuina laskettaessa tulee laskutoimituksissa pieni virhe. Esimerkiksi #1 ollessa 0.002 ei #2=#1*1000 anna arvoksi 2 vaan 1.99999997. Tämä saattaa aiheuttaa virheellisiä tilanteita esimerkiksi vertailuoperaattoreita käytettäessä.

Oletetaan, että lauseessa IF [#1 EQ #2] sekä #1 että #2 on mainitun tyyppinen virhe. Tällöin lauseen vastaus ei välttämättä olekaan oletetun kaltainen. Tällaisissa tilanteissa onkin parempi tutkia kahden muuttujan erotuksen suuruutta.
IF [ABS[#1 - #2] LT 0.001]
Mikäli kahden muuttujan välinen erotus on pienempi kuin sallittu virhe (tässä tilanteessa 0.001), voidaan olettaa, että #1 ja #2 ovat yhtä suuria.

6.2 Tallennustarkkuus

Toinen virhemahdollisuus on tallennustarkkuudessa. Muuttujien arvot tallennetaan kahdeksan luvun tarkkuudella, jolloin yrittäessäsi antaa seuraavat määritykset
#1=9876543210123.456
#2=9876543277777.777
tallentuu muuttujien arvoiksi
#1=9876543200000.000
#2=9876543300000.000
#2 ja #1 erotushan on todellisuudessa 67654.321 mutta laskettaessa #3=#2-#1 saadaan #3=100000.000 .

6.3 Käyttötarkkuus

Kolmas kohta, jossa muuttujien arvot voivat olla virheelliset, on NC-osoitteet. Ohjaus pyöristää muuttujien arvot automaattisesti osoitteen käyttämään tarkkuuteen. Esimerkiksi #1 ollessa 12.3456 toteutuu X#1 muodossa X12.346. Tämä voi aiheuttaa ongelmia esimerkiksi seuraavanlaisissa tilanteissa:

```
#1=1.2345;
#2=2.3456;
G0 G90 X0;  lähtötilanteessa X on nollassa
G0 G91 X-#1;          terä liikkuu 1.235 mm, eli X-1.235:en
G1 X-#2 F200;         terä liikkuu 2.346 mm, eli X-3,581:en
G0 X[#1+#2];          koska 1.2345+2.3456=3.5801 on liikematka
                      3.580, mikä ei palauta terää X0:aan
```

Tämä virhe johtuu siitä, että laskutoimitus tehtiin ennen pyöristystä. Virhe voidaan välttää suorittamalla pyöristys ennen yhteenlaskua. G0 X[ROUND[#1]+ROUND[#2]] suorittaa viimeisen lauseen halutulla tavalla.

7 SILMUKAT JA HYPPYKOMENNOT /1/

Jotta parametriohjelma saataisiin toistamaan haluttua väliä tai hyppäämään tiettyyn ohjelman kohtaan, käytetään erilaisia silmukka- tai hyppykomentoja. Käytettävissä olevat silmukka- ja hyppykomennot on lueteltu taulukossa 8.

TAULUKKO 8. Silmukka- ja hyppykomennot

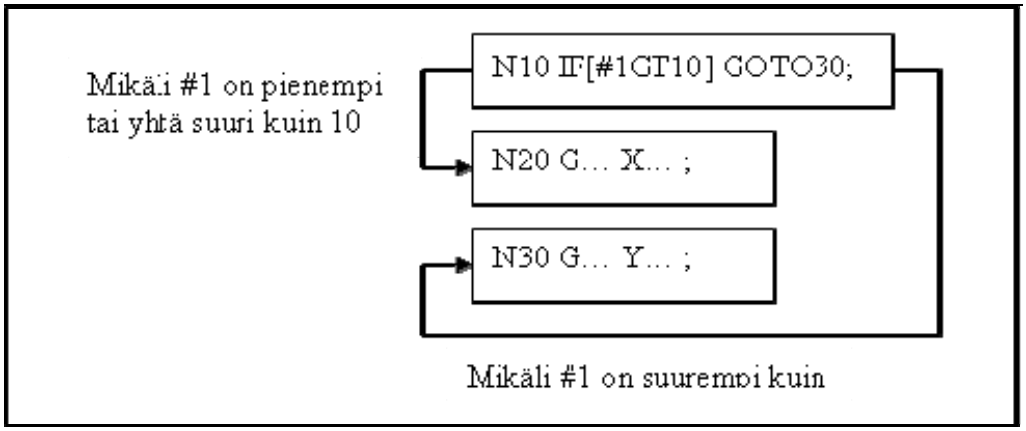
Komento	Toiminto
GOTO	Ehdoton hyppy
IF	Ehdollinen hyppy
WHILE – END	Ehdollinen silmukka

7.1 GOTO-komento

GOTO-käsky pakottaa ohjelman suorituksen hyppäämään määrättyyn lauseeseen, joka kirjoitetaan komennon perään.
Esimerkiksi GOTO100 tai GOTO#10.

7.2 IF-komento

IF-käsky on ehdollinen versio GOTO-käskystä. Lauseessa määritellään jokin ehto, jonka toteutuessa samassa lauseessa oleva GOTO- tai THEN-käsky toteutetaan. Muussa tapauksessa ohjelman suoritus jatkuu normaalisti seuraavaan lauseeseen. IF-komennon toimintaa on kuvattu kuvassa 2.

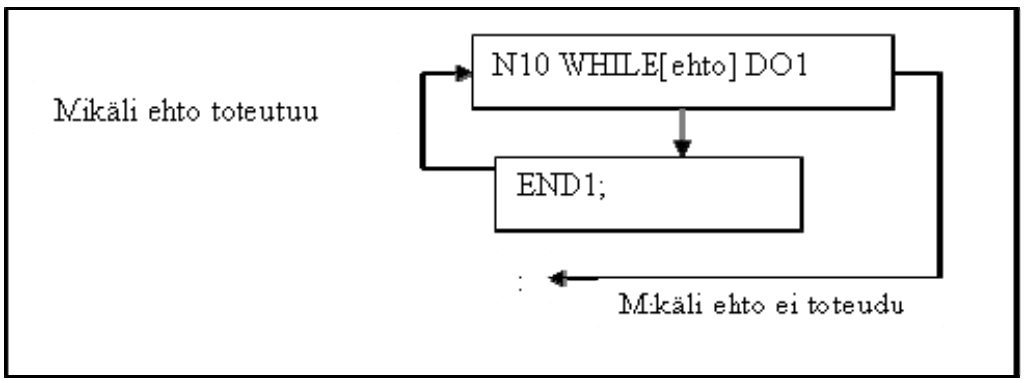


KUVA 2. IF-käskyn toimintaperiaate GOTO-komennon kanssa

Vastaavasti voidaan ehdon täyttyessä suorittaa jokin muuttujakomento. Esimerkiksi IF[#1EQ10] THEN #3=0.
Tällöin #1 ollessa 10 saa #3 arvon 0.

7.3 WHILE–END-komento

WHILE-komento on silmukkakomento jonka avulla voidaan määrittää tietyn ohjelman pätkän suorittamisehto. Mikäli WHILE-lauseen ehto täyttyy, suoritetaan sen ja END-lauseen välissä oleva ohjelman pätkä. Mikäli ehto ei täyty, ohjelman suoritus jatkuu END-lauseen jälkeisestä lauseesta. WHILE-komennon toimintaa on kuvattu kuvassa 3.



KUVA 3. WHILE -komennon toimintaperiaate

Sisäkkäisiä WHILE-silmukoita voi olla kolme kappaletta, jolloin eri silmukoiden alku- ja loppukohdat erotellaan numeroilla 1, 2 ja 3. Numero kirjoitetaan sekä WHILE-lauseen DO-komennon perään että silmukan lopettavan END-komennon perään.

8 OHJELMOINTIRAJOITUKSIA /1/

Hyppy-, silmukka- ja vertailukomentojen käytöllä on muutamia rajoituksia, jotka tulee muistaa niitä käytettäessä.

8.1 IF-komento

Ehdollisen hypyn avulla voidaan hypätä eri kohtiin ohjelmassa. Sillä ei kuitenkaan voi hypätä WHILE-silmukan ulkopuolelta lauseeseen, joka sijaitsee silmukan sisällä, mutta sillä voidaan hypätä ulos silmukan sisältä ja näin katkaista se.

8.2 WHILE-komento

Vaikka ohjelma sallii kolmen sisäisen silmukan ohjelmoinnin, on siinä kuitenkin yksi rajoitus: Mikäli silmukka sisältää toisen silmukan, on sisemmän silmukan loppu määriteltävä ennen ulomman silmukan loppua. Suljettujen silmukoiden numerot voidaan käyttää uudestaan, mikäli edellinen samaa numeroa käyttänyt silmukka on jo päättynyt.

8.3 DO-komento

DO-käsky on aina muistettava sitoa WHILE-komentoon. Mikäli se määritetään ilman WHILE-ehtoa, syntyy DO- ja END-komentojen välille loputon silmukka.

8.4 Vertailuoperaattorit

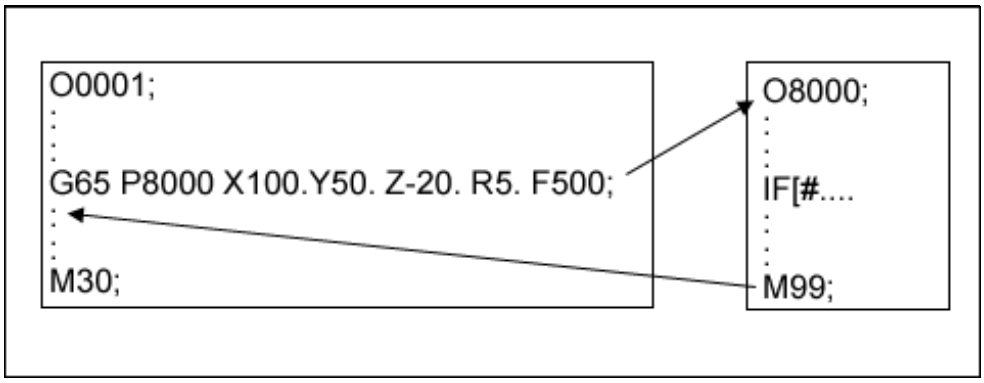
Mikäli ehtolauseessa käytetään operaattoreita EQ tai NE, antavat muuttujien arvot tyhjä ja nolla eri tulokset. Muilla vertailuoperaattoreilla molempia arvoja kohdellaan kuten arvoa nolla.

9 PARAMETRIOHJELMAN KUTSUMINEN /1/

Parametriohjelma voidaan kutsua käyttöön eri tavoilla. Seuraavaksi käsitellään eri tapoja kutsua makro.

9.1 Yksinkertainen kutsu (G65)

Makro voidaan kutsua käyttäen komentoa G65, jolloin makron ohjelmanumero määritellään P-kirjaimen avulla. Esimerkiksi G65 P9010 L2. Kutsun L-kirjan ilmaisee toistojen lukumäärän eli kuinka monta kertaa makro kutsutaan peräjälkeen. Toistojen lukumäärä voidaan jättää pois kutsusta, jolloin makro toteutetaan vain kerran. Makrokutsun toimintaperiaate on kuvattu kuvassa 4.



KUVA 4. G65-kutsun käyttö

Kuten aiemmin mainittiin, täytyy makron lähtöarvot määrittää ennen sen kutsumista. Ne voidaankin määrittää kahdella eri tavalla, joista toinen käsitellään tässä. Toinen tapa taas liittyy 3D-pistepilven tyyppiseen määritykseen.

Yleisin tapa on kirjoittaa lähtöarvot samaan lauseeseen kutsun kanssa, jolloin tulee käyttää parametrien kirjaintunnuksia, jotka ovat taulukossa 9.

TAULUKKO 9. Parametrien kirjaintunnukset

Tunnus	Muuttuja	Tunnus	Muuttuja	Tunnus	Muuttuja
A	#1	I	#4	T	#20
B	#2	J	#5	U	#21
C	#3	K	#6	V	#22
D	#7	M	#13	W	#23
E	#8	Q	#17	X	#24
F	#9	R	#18	Y	#25
H	#11	S	#19	Z	#26

Kirjoittamalla esimerkiksi kuvan 4 mukainen lause
G65 P8000 X100. Y50. Z-20. R5. F500 ;
määritetään muuttujat seuraavasti

X100. ⇨ #24=100
Y50. ⇨ #25=50
Z-20. ⇨ #26=20
R5. ⇨ #18=5
F500. ⇨ #9=500

jonka jälkeen kutsutaan ohjelma numero O8000. Kirjaimia G, L, N, O ja P ei voida käyttää muuttujien määrityksiin. Mikäli jollekin muuttujalle ei tarvitse määrittää arvoa, voidaan se jättää pois muuttujamäärityksestä.

Makron sisällä voi kutsua toisen makron. Näin voidaan kutsua peräti neljä allekkaista makroa. Jokaiseen makroon voidaan määrittää muuttujat #1-#33, jotka määritetään ennen seuraavan makron kutsua ja palautetaan ennalleen kutsutun makron päättyessä.

9.2 Modaalinen kutsu (G66)

Modaalinen kutsu toimii samoin kuin G65-käsky – sillä lisäyksellä, että ohjelmoitaessa G66 makro toteutetaan myös jokaisen kutsua seuraavan paikoituksen jälkeen. Kutsu toimii siis samoin kuin esimerkiksi poraustyökierto G81. G66 kumotaan komennolla G67. Myös G66-kutsulla voidaan kutsua allekkaisia makroja.

9.3 G-makrokutsu

Makro voidaan kutsua myös G-koodin avulla. Tällöin ohjelmanumeron tulee olla välillä O9010–O9019. Tämän tavan perusesimerkki on Fanucin omat työkierrot, kuten G81 ja G83. G81-koodi ei siis tosiasiassa ole itsessään työkierto, vaan koodi, jolla kutsutaan jokin tietty ohjelmanumero, jossa kyseinen työkierto on. Kunkin ohjelman kutsuun käytettävä G-koodi määritetään vastaavaan parametriin taulukon 10 mukaisesti.

TAULUKKO 10. G-koodikutsun parametrinumerot

Ohjelmanumero	Parametrinumero
O9010	6050
O9011	6051
O9012	6052
O9013	6053
O9014	6054
O9015	6055
O9016	6056
O9017	6057
O9018	6058
O9019	6059

Esimerkiksi kutsuttaessa ohjelma O9010 koodilla G81 on parametrinumeron #6050 arvoksi määritettävä 81. Muuttujien lähtöarvot määritellään kuten G65- ja G66-kutsuilla. Tällä kutsutavalla yhdistetään siis normaalin G65-kutsun G- ja P-koodi yhdeksi G-koodiksi.
Esimerkiksi määrittämällä parametrin #6010 arvoksi 81, voidaan lause
G65 P9010 X100. Y50. Z-20. R5. F500 ;
kirjoittaa muotoon
G81 X100. Y50. Z-20. R5. F500 ;
Tietyt G-koodit , kuten esimerkiksi G1,on varattu ohjauksen omaan käyttöön, joten niitä ei voi käyttää makrokutsussa.

Mikäli makro on kutsuttu G-, M- tai T-koodilla, et voi kutsua toista makroa käyttäen mitään näistä koodeista, vaan niissä tulee käyttää esimerkiksi G65-komentoa. Kutsutuissa makroissa tällaisia G-, M- ja T-koodeja kohdellaan kuten vastaavia normaaleja koodeja.

9.4 M-makrokutsu

Makron voi kutsua myös M-koodin avulla. Kuten G-koodillakin kutsuttaessa on käytettävä ohjelmanumero määrätty. Tässä tilanteessa numeron tulee olla välillä O9020–O9029. Vastaavat parametrit on esitetty taulukossa 11.

TAULUKKO 11. M-koodikutsun parametrinumerot

Ohjelmanumero	Parametrinumero
O9020	6080
O9021	6081
O9022	6082
O9023	6083
O9024	6084
O9025	6085
O9026	6086
O9027	6087
O9028	6088
O9029	6089

Muuttujien lähtöarvot määritellään kuten G65- ja G66-kutsuilla. Tietyt M-koodit on varattu ohjauksen omaan käyttöön, kuten esimerkiksi M30, joten niitä ei voi käyttää makrokutsussa. Tällä kutsutavalla yhdistetään siis normaalin G65-kutsun G- ja P-koodi yhdeksi M-koodiksi.

Esimerkiksi määrittämällä parametrin #6080 arvoksi 81, voidaan lause
G65 P9020 X100. Y50. Z-20. R5. F500 ;
kirjoittaa muotoon
M81 X100. Y50. Z-20. R5. F500 ;

Mikäli makro on kutsuttu G-, M- tai T-koodilla, et voi kutsua toista makroa käyttäen mitään näistä koodeista, vaan niissä tulee käyttää esimerkiksi G65-komentoa. Kutsutuissa makroissa tällaisia G-, M- ja T-koodeja kohdellaan kuten vastaavia normaaleja koodeja.

9.5 M-aliohjelmakutsu

M-koodilla voidaan kutsua myös aliohjelma vastaavalla tavalla kuin M98-komennolla mutta ilman ohjelmanumeron määritystä. Ero yllä mainittuun makrokutsuun on siinä, että aliohjelmakutsussa ei voida määrittää muuttujien lähtöarvoja kutsun yhteydessä. Tällä tavalla voidaan siis kutsua esimerkiksi aliohjelmia, jotka toistuvat usein ja aina samankokoisena. Vastaavat parametrit on esitetty taulukossa 12.

TAULUKKO 12. M-aliohjelmaikutsun parametrinumerot

Ohjelmanumero	Parametrinumero
O9001	6071
O9002	6072
O9003	6073
O9004	6074
O9005	6075
O9006	6076
O9007	6077
O9008	6078
O9009	6079

Tietyt M-koodit on varattu ohjauksen omaan käyttöön, kuten esimerkiksi M30, joten niitä ei voi käyttää makrokutsussa. Tällä kutsutavalla yhdistetään siis normaalin M98-kutsun M- ja P-koodi yhdeksi M-koodiksi.

Esimerkiksi määrittämällä parametrin #6071 arvoksi 81, voidaan lause
M98 P9001 ;
kirjoittaa muotoon
M81 ;

Mikäli makro on kutsuttu G-, M- tai T-koodilla, et voi kutsua toista makroa käyttäen mitään näistä koodeista, vaan niissä tulee käyttää esimerkiksi G65-komentoa. Kutsutuissa makroissa tällaisia G-, M- ja T-koodeja kohdellaan kuten vastaavia normaaleja koodeja.

9.6 T-aliohjelmakutsu

Viimeisin tapa kutsua makro tai aliohjelma on käyttää T-koodia. Asettamalla parametrin 6001 viidennen bitin arvoksi 1, voidaan T-koodin avulla kutsua ohjelma O9000. Ohjelmassa oleva T-koodi kutsuu aina saman ohjelman ja koodin arvo tallennetaan yleismuuttujaan #149. Tällöin ohjelmassa oleva lause
T16 ;
määrittää muuttujan #149 arvoksi 16 ja kutsuu ohjelman O9000. Tämä ei varsinaisesti sovellu koneistusohjelmien parametriohjelmointiin, vaan sen avulla voidaan luoda esimerkiksi koneelle halutun tyyppinen työkalunvaihtosekvenssi.

Mikäli makro on kutsuttu G-, M- tai T-koodilla, et voi kutsua toista makroa käyttäen mitään näistä koodeista, vaan niissä tulee käyttää esimerkiksi G65-komentoa. Kutsutuissa makroissa tällaisia G-, M- ja T-koodeja kohdellaan kuten vastaavia normaaleja koodeja.

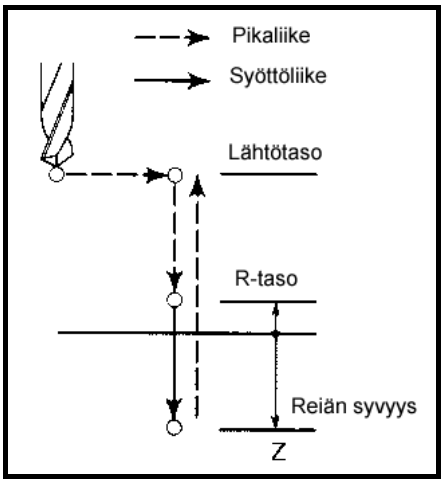
10 OHJELMAN KIRJOITTAMINEN

Makron kirjoittamisessa tulee muistaa, että kaikki makron sisällä tehtävät koneen tilamuutokset tulee myös palauttaa lähtötilaansa. Koska jokaisen koneen ohjaus on aina hieman erilainen ja ohjaukseen ostetut lisäoptiot vaihtelevat, voi ohjauksessa olla useita eri käyttötiloja, joita ei toisessa koneessa välttämättä ole, esimerkiksi napakoordinaatisto. Ei siis kannata tehdä makroa, joka toimisi aina samalla tavalla koneen lähtötilasta riippumatta. Konekohtaisia makroja suunniteltaessa ja parametriohjelmointitaitojen kasvaessa tämän tyyppinen räätälöinti on toki mahdollista.

Toinen muistettava asia on makron laatiminen niin, että se toimii kaikissa ohjausversioissa. Esimerkiksi automaattinen kulmanpyöristys on maksullinen lisäoptio, joten ei voida olettaa, että se olisi kaikissa ohjauksissa. Tämän takia makroissa käytetäänkin G2- ja G3-koodia kaikissa kaariliikkeissä.

10.1 Makroesimerkki

Jotta voisimme käsitellä makron kirjoittamista on hyvä ensin katsoa, minkälainen yleisesti käytetty porausmakro – G81 – on. Esimerkissä porausmakro kutsutaan modaalisella makrokutsulla. Makron muuttujat on esitelty kuvassa 5.



KUVA 5. Porausmakron muuttujat

Pääohjelma

O0001;

T1 M6;

G54 G90;	G90 tallentaa arvon 90 parametriin #4003
G0 X100. Y0. S1500 F500 M3;	G0 tallentaa arvon 0 parametriin #4001
G43 Z100. H1 M8;	F500 tallentaa arvon 500 parametriin #4109
G66 P9010 Z-20. R5. F200;	Makrokutsu kutsuu ohjelman O9010 ja määrittää parametreille #26 (Z), #18 (R) ja #9 (F) arvot -20, 5 ja 200.

X0. Y100.;

X-100. Y0.;

G67;

M30;

Makro

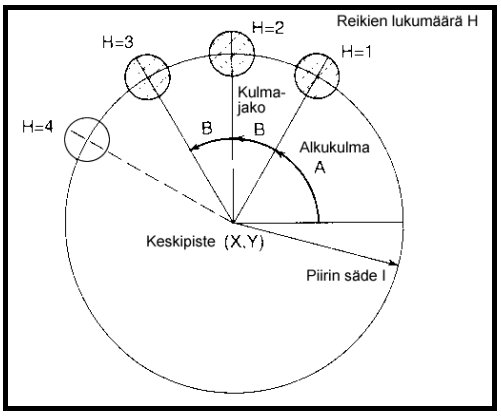
```
O9010;
#1=#4001;                    Haetaan käytössä oleva
                             G0/1-koodi
#3=#4003;                    Haetaan käytössä oleva
                             G90/91-koodi
#4=#4109;                    Haetaan käytössä oleva
                             syöttöarvo
#5=#5003;                    Talletetaan terän kärjen
                             Z-koordinaatti
G0 G90 Z#18;                 Siirretään terä R-tasolle
G1 Z#26 F#9;                 Poraus syöttöliikkeellä
                             Z-arvoon syötöllä F
IF[#4010 EQ 98] GOTO 1;      Tarkistetaan oliko paluu
                             R-tasolle vai lähtötasolle
G0 Z#8;                      Paluu R-tasolle
GOTO 2;
N1 G0 Z#5;                   Paluu lähtötasolle
N2 G#1 G#3 F#4;              Palautetaan makrokutsua
                             edeltäneet koodit
M99;
```

Yllä olevassa esimerkissä on hyvin yleinen esimerkki makrosta ja sen perusrakenteesta. Kuten pääohjelman huomautuksista nähdään, tallentuvat tietyt koodit järjestelmäparametreihin. Näistä voidaan hakea koneen tilatiedot makron alussa ja näin pystytään myös palauttamaan sama tila makron lopussa. Yllä olevassa pääohjelmassa kutsutaan yleisesti käytettyä G81-poraustyökiertoa, joten jos parametrit on määritetty luvun 9.3 mukaisesti, voidaan G66 P9010 korvata koodilla G81.

Yllä olevan esimerkin pääohjelman makrokutsussa on tyypillinen esimerkki sen rakenteesta. Kutsukoodina käytettiin tässä tilanteessa G66-koodia, jolloin kutsuttavan ohjelman numero annettiin P-koodin avulla. Näiden perään kirjoitettiin makron lähtötiedot taulukon 9 mukaisilla tunnuksilla.

10.2 Sisäkkäiset makrot

Kuten on todettu, allekkaisia makroja voi kutsua neljä kappaletta. Tyypillisesti käytössä on kuitenkin vain kaksi makroa, kuten esimerkiksi reikäpiirimakrossa.



KUVA 6. Reikäpiirin muuttujat

Pääohjelma

```
O0001;
T1 M6;
G54 G90;
G0 X100. Y0. S1500 F500 M3;
G43 Z100. H1 M8;
G65 P9100 X100. Y50. R30. Z-50. F500 I100.
A0. B45. H5;
X0. Y100.;
X-100. Y0.;
G67;
M30;
```

G65-muuttujien selitykset:
X= piirin keskipiste (X)
Y= piirin keskipiste (Y)
R= R-taso
F= Poraussyöttö
I= piirin säde
A= 1. reiän kulma
B= jakokulma
H= reikien lukumäärä

Makro

O9100;	
#3=#4003;	Haetaan käytössä oleva G90/91-koodi
G81 Z#26 R#18 F#9 K0;	Määritellään poraustyökierto (HUOM. K0)
IF[#3 EQ 90] GOTO 1;	Tarkistetaan on käytössä G90 ohjelmointi
#24=#5001+#24;	Jos käytössä oli G91, niin lasketaan piirin
#25=#5002+#25;	keskipisteen absoluuttiset koordinaatit
N1 WHILE[#11 GT 0] DO 1;	Silmukka toteutetaan, niin kauan kuin porattavia reikiä on jäljellä.
#5=#24+#4*COS[#1];	Lasketaan porattavan reiän keskipisteen
#6=#25+#4*SIN[#1];	X- ja Y-koordinaatit
G90 X#5 Y#6;	Paikoitusreiälle ja poraus
#1=#1+#2;	Seuraavan reiän kulma
#11=#11-1;	Jäljellä olevien reikien lukumäärän päivitys
END 1;	Silmukan loppu.
G#3 G80;	G90/91-tilan palautus ja porauskierron kumoaminen.
M99;	

Sisäkkäisiä makroja kutsuttaessa on kaikkien makrojen lähtötiedot annettava pääohjelmassa, jotta niiden avulla voidaan kutsua alemmat makrot. Sisäkkäisten makrojen lähtöarvot voivat toki riippua edeltävän makron sisällä suoritetuista toimista, jolloin makro itse määrittää seuraavan makron lähtötietoja ja tällöin ei näitä tietoja tarvitse antaa etukäteen.

Seuraavassa esimerkissä pääohjelma kutsuu reikäpiirimakron, joka puolestaan kutsuu G81-poraustyökiertoa. Sekä reikäpiirin että poraustyökierron tarvitsemat lähtötiedot annetaan pääohjelmassa reikäpiirimakron kutsun yhteydessä.

11 ESIMERKKEJÄ

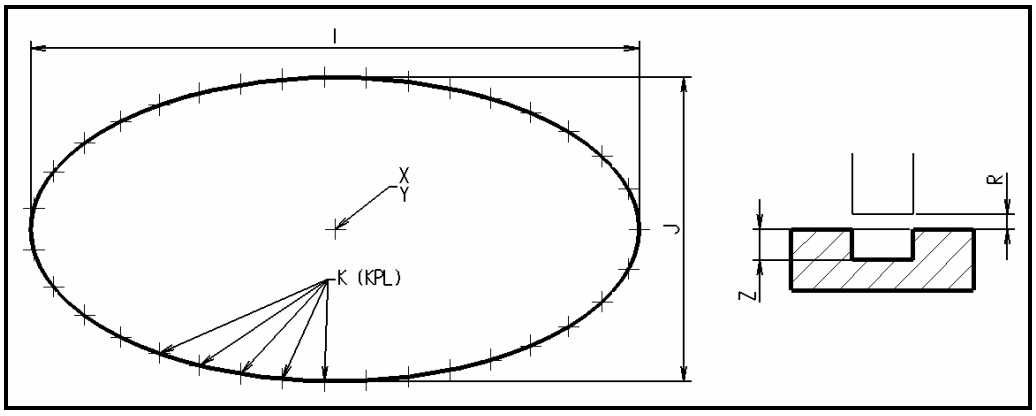
Seuraavaksi käydään läpi muutamia erilaisia parametriohjelmia ja niiden rakennetta. Näiden avulla saat käsityksen ohjelman perusrakenteesta ja hyvän yleiskatsauksen ohjelman kirjoittamiseen.

11.1 Ellipsi

Ellipsiohjelma on hyvin yksinkertainen parametriohjelma, jossa näkyy parametriohjelmille tyypillinen rakenne.
Ellipsimakro voidaan kutsua esimerkiksi G65-koodin avulla.

```
G65 P8003 X100 Y-100 Z-5 I80 J60 K36 R2 F200 (ELLIPSI)
```

G65-lauseessa on kutsuttavan ohjelman numero ja makron lähtötiedot.



KUVA 7. Ellipsin muuttujat

- P = MAKRON OHJELMANUMERO
- X = KESKIPISTE X (ABS) = #24
- Y = KESKIPISTE Y (ABS) = #25
- Z = MUODON SYVYYS Z (ABS) = #26
- I = ELLIPSIN LEVEYS = #4
- J = ELLIPSIN KORKEUS = #5
- K = LASKENTAPISTEIDEN MÄÄRÄ = #6
- R = R-TASO = #18
- F = JYRSINTÄSYÖTTÖ = #9

```
O8003(ELLIPSI)

(LAHTOTIETOJEN HAKU)
#1=#4001
#2=#4003
#3=#4019
#6=360/#6
#7=#5003
```

Lähtötiedoissa haetaan ennen kutsua voimassa oleva G0/1-koodi (#4001), G90/91-koodi (#4003), F-koodin arvo (#4019), sekä terän Z-koordinaatin arvo kutsuhetkellä. Lisäksi lasketaan laskentapisteiden välinen kulma (#6).

```
(ABS/INK TARKISTUS)
IF[#2EQ90]GOTO1
#24=#5001+#24
#25=#5002+#24
```

Tarkistetaan oliko ennen makrokutsua käytössä absoluuttinen vai inkrementaalinen ohjelmointi. Mikäli käytössä oli inkrementaalinen ohjelmointi, lasketaan annettuihin keskipisteen koordinaatteihin mukaan

terän tämänhetkiset koordinaattiarvot, jolloin saadaan keskipisteen sijainti absoluuttisessa koordinaatistossa.

```
(ALOITUSPARAMETRIT)
N1
#4=#4/2
#5=#5/2
#10=0(TYOSTOKULMA XY)
```

Aloitusparametreissa määritellään ensimmäisen lastun/paikoituksen laskenta-arvot. Yllä olevassa määritetään laskentasäteet puolittamalla ellipsin leveys ja korkeus sekä nollataan laskentakulman arvo.

```
(PAIKOITUS)
#11=#4*COS[#10]
#12=#3*SIN[#10]
G90G0X[#24+#11]Y[#25+#12]
Z#18
G1Z#26F[#9/2]
#10=#10+#6
```

Paikoituksessa tuodaan terä makrokierron alkupisteeseen. Ensin on laskettu ensimmäisen pisteen X-koordinaatti (#11) ja Y-koordinaatti (#12) suhteessa keskipisteeseen. Seuraavaksi on paikoitettu terä XY-tasossa muodon alkuun absoluuttisessa koordinaatistossa.
Sitten tuodaan terä alas R-tasolle ja poraudutaan jyrsintäsyvyyteen puolella jyrsintäsyötön arvosta ja lopuksi päivitetään laskentakulma seuraavaan arvoonsa.

```
(RATAKIERTO)
WHILE[#10LE360]DO1
G1F#9
#11=#4*COS[#10]
#12=#3*SIN[#10]
G90G1X[#24+#11]Y[#25+#12]
#10=#10+#6
END1
```

Ratakierto on ellipsimakron ainoa silmukka, joka on toteutettu WHILE-komennolla. Ehto silmukan toteutumiselle on, että niin kauan kuin laskentakulma on pienempi tai yhtä suuri kuin 360º, silmukka toteutetaan. Mikäli ehto ei toteudu, ohjelman suoritus jatkuu END1-rivin jälkeen.

Ensin silmukassa on määritelty liiketyyppi syöttöliikkeeksi. Koska NC-ohjaus lukee ohjelmaa muistiinsa, on vaarana muuttujalaskennan ennenaikainen suoritus. Tämä laskutoimituksen ennenaikainen suoritus voidaan estää kirjoittamalla esimerkiksi G1-lause ennen muuttujalaskentaa. Yleensä tätä vaaraa ei silmukoissa ole, koska ohjaus ei lue muistiinsa silmukan sisällä olevaa ohjelmaa ennen kuin suoritus pääsee samaan kohtaan.

Seuraavaksi lasketaan uudet X- ja Y-koordinaatit ja ajetaan terä niihin. Liike on suoraviivainen eli muoto on kulmikas. Muodon kulmikkuus vähenee laskentapisteiden määrän kasvaessa. Lopuksi päivitetään laskentakulma.

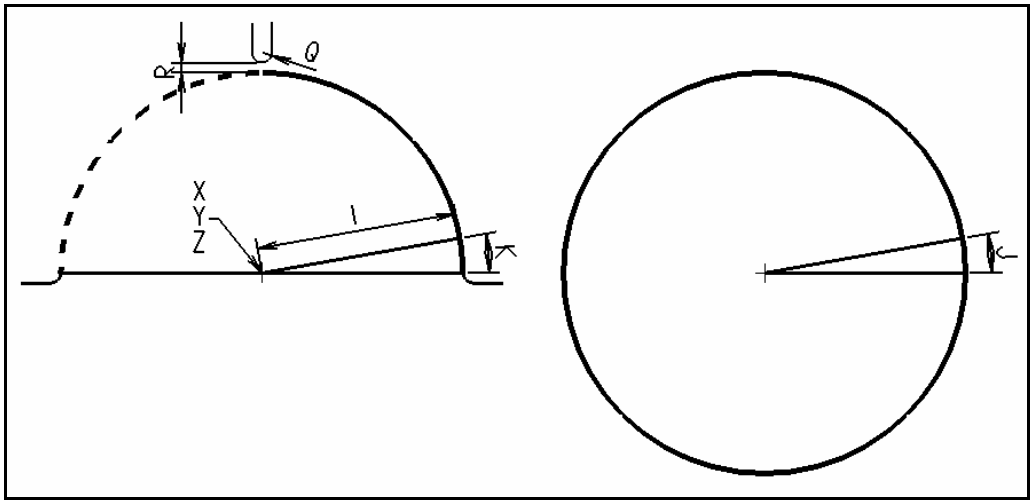
```
(MAKRON LOPETUS)
IF[#4010EQ98]GOTO2
G0Z#18
GOTO3
N2G0Z#7
N3G#1G#2F#3
M99
```

Makro lopetetaan tarkistamalla, onko voimassa G98 eli paluu makron jälkeen lähtötasolle. Mikäli G98 on voimassa, hypätään R-tasolle paluun yli ja nostetaan terä lähtötasolle. Jos se taas ei ole voimassa, nostetaan terä R-tasolle ja hypätäänkin lähtötasolle paluun yli. Lopuksi palautetaan ohjaukselle makrokutsua edeltänyt tila.

11.2 Pallopinnan viimeistely

Pallopinnan viimeistelymakro on kirjoitettu pallopäiselle terälle pinnan viimeistelyä varten.

```
G65P8002X-100Y-100Z0I50J10K10R2Q6F200(PALLOPINTA)
```



KUVA 8. Pallopinnan muuttujat

- P = MAKRON OHJELMANUMERO
- X = KESKIÖPISTE X (ABS) = #24
- Y = KESKIÖPISTE Y (ABS) = #25
- Z = KESKIÖPISTE Z (ABS) = #26
- I = PALLON SÄDE = #4
- J = KULMA-ASKEL (XY-SUUNTA) = #5
- K = KULMA-ASKEL (XZ-SUUNTA) = #6
- R = R-TASO (PALLON YLÄPUOLELLA, INK) = #18
- Q = PALLOPÄISEN TERÄN SÄDE = #17
- F = JYRSINTÄSYÖTTÖ = #9

```
O8002(PALLOPINTA)

(LAHTOTIETOJEN HAKU)
#1=#4001
#2=#4003
#3=#4019
#7=#5003

(PAIKOITUS)
G90G0X#24Y#25
Z[#26+#18+#17+#4]
```

Paikoitus pallopinnan yläpuolelle keskelle pintaa R-tason verran irti pinnasta.

```
(ALOITUSPARAMETRIT)
#10=0(TYOSTOKULMA XY)
#11=90(TYOSTOKULMA XZ)
#12=#4+#17(OHJELMOITAVAN PALLON SADE)
```

Työstörata lasketaan pallopäisen terän keskiöpisteelle.

```
(KEHAKIERTO)
WHILE[#10LT360]DO1
G90G0X#24Y#25
Z[#26+#17+#4]
```

Määritellään XY-suunnassa olevan lastun silmukka ja tuodaan terä kiinni kappaleeseen.

```
(PINTAKIERTO)
WHILE[#11GE0]DO2
G1F#9
#13=#12*SIN[#11](Z-KOORDINAATTI)
#14=#12*COS[#11](APULUKU)
#15=#14*SIN[#10](Y-KOORDINAATTI)
#16=#14*COS[#10](X-KOORDINAATTI)
G1X[#24+#16]Y[#25+#15]Z[#26+#13]
#11=#11-#6
END2
```

Lasketaan XZ-suuntaisen lastun liikeradan seuraava piste. Piste ja pallon keskiöhän ovat kuution vastakkaiset kulmat. Sen tähden voidaan koordinaatit määrittää trigonometrian avulla. Liikkeen jälkeen päivitetään XZ-suunnan laskentakulma. Jos laskentakulma on alle 0°, XZ-lastu on valmis ja silmukka päättyy.

```
G0Z[#26+#18+#17+#4]
#10=#10+#5
#11=90
END1
```

Terä nostetaan ylös ja päivitetään, seuraavan XY-suuntaisen lastun laskentakulma lasketaan ja palautetaan XZ-lastun alkuperäinen laskentakulma. Ulompi silmukka toteutuu, kunnes XY-suunan laskentakulma saavuttaa arvon 360º, jolloin pinta on valmis.

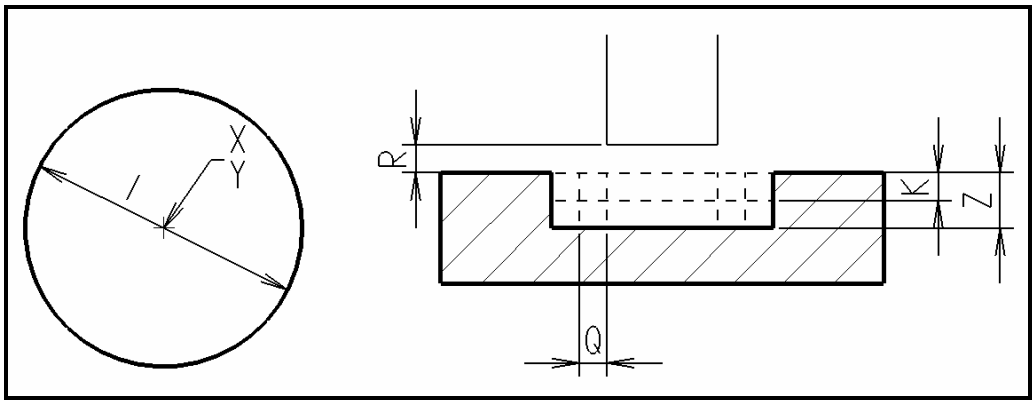
```
(MAKRON LOPETUS)
IF[#4010EQ98]GOTO1
G0Z[#26+#18+#17+#4]
GOTO2
N1G0Z#7
N2G#1G#2F#3
M99
```

Makro lopetetaan samoin kuin ellipsimakro.

11.3 Ympyrätasku

Ympyrätasku on hyvin yleisesti käytetty makro. Sen kirjoittamisessa on kuitenkin muutamia huomioitavia seikkoja.

```
G65P8000X-100Y100R2I25Z-20Q6K6F200D6(YMPYRATASKU)
```



KUVA 9. Ympyrätaskun muuttujat

- P = MAKRON OHJELMANUMERO
- X = KESKIPISTE X (ABS tai INK) = #24
- Y = KESKIPISTE Y (ABS tai INK) = #25
- R = R-TASO (ABS) = #18
- I = HALKAISIJA = #4
- Z = TASKUN SYVYYS = #26
- Q = LASTUN LEVEYS = #17
- K = LASTUN SYVYYS = #6
- F = JYRSINTÄSYÖTTÖ = #9

```
O8000(YMPYRATASKU)

(LAHTOTIETOJEN HAKU)
#1=#4001
#2=#4003
#3=#4019
IF[#7EQ#0]THEN#7=#4107
#8=#5003
```

Lähtötiedot ovat samat kuin aiemmissa makroissa, mutta lisäksi tarkistetaan, onko kompensointitieto D annettu. Mikäli sitä ei ole kutsussa määritelty, haetaan aikaisemmin määritelty arvo.

```
(MUUTTUJIEN TARKISTUS)
IF[ABS[#4/2]-ABS[#17]LT0]THEN#17=ABS[#4/2]
IF[ABS[#26]-ABS[#6]LT0]THEN#6=ABS[#26]
```

Muuttujien tarkistuksessa varmistetaan, ettei laskunpaksuus/-leveys ole suurempi kuin vastaava lopullinen arvo. Mikäli näin kuitenkin on, muutetaan lastunpaksuuden/-leveyden arvoksi vastaava loppuarvo.

```
(ENSIMMAISET LASTUT)
#13=ABS[#6]
#14=#17
#19=#14
```

Ensimmäisen lastun parametrit määritellään ja poistetaan lastun syvyyteen mahdollisesti annettu etumerkki sekä määritellään jyrsityn säteen seuranta-arvo (#14).

```
(ABS/INK TARKISTUS)
IF[#2EQ90]GOTO1
#24=#5001+#24
#25=#5002+#24

(PAIKOITUS)
N1G90G0X#24Y#25
Z#18

(TASKUN SYVYYSKIERTO)
WHILE[-#13GE#26]DO1
G90G1Z-#13F[#9/2]

(TASKUNJYRSINTAKIERTO)
WHILE[#14LE[#4/2]]DO2
G91G1G41X#17F#9D#7
G3X0Y0I-#14F#9
G1G40
IF[#14EQ[#4/2]]GOTO2
#14=#14+#17
IF[#14GT[#4/2]]THEN#17=#17-[#14-[#4/2]]
IF[#14GT[#4/2]]THEN#14=[#4/2]
END2
```

Terää siirretään lastun leveyden verran sivulle ja jyrsitään ympyrä. Tätä jatketaan kunnes seuraavaksi jyrsittävän taskun säde on lopullista sädettä suurempi tai yhtä suuri. Mikäli se on yhtä suuri, hypätään IF-lauseen avulla pois silmukasta, koska silloin tämä syvyys on valmis. Mikäli se on suurempi, muutetaan lastunleveysarvo vastaamaan jäljellä olevaa lastun leveyttä ja suoritetaan viimeinen lastu näillä arvoilla.

```
N2G90G1G40X#24Y#25
#14=#19
#17=#19
IF[-#13EQ#26]GOTO3
#13=#13+#6
IF[-#13LT#26]THEN#13=ABS[#26]
END1
```

Lastunleveyden ja jyrsityn säteen seuranta-arvot palautetaan. Tämän jälkeen tarkistetaan, onko jyrsitty syvyys sama kuin lopullinen syvyys. Jos on, hypätään IF-lauseen avulla ulos silmukasta. Sen jälkeen lasketaan uusi jyrsintäsyvyys ja tarkistetaan, ettei uusi syvyys ole suurempi kuin lopullinen syvyys. Jos uusi syvyys on suurempi, suoritetaan vastaava korjaus kuin säteen jyrsinnässä.

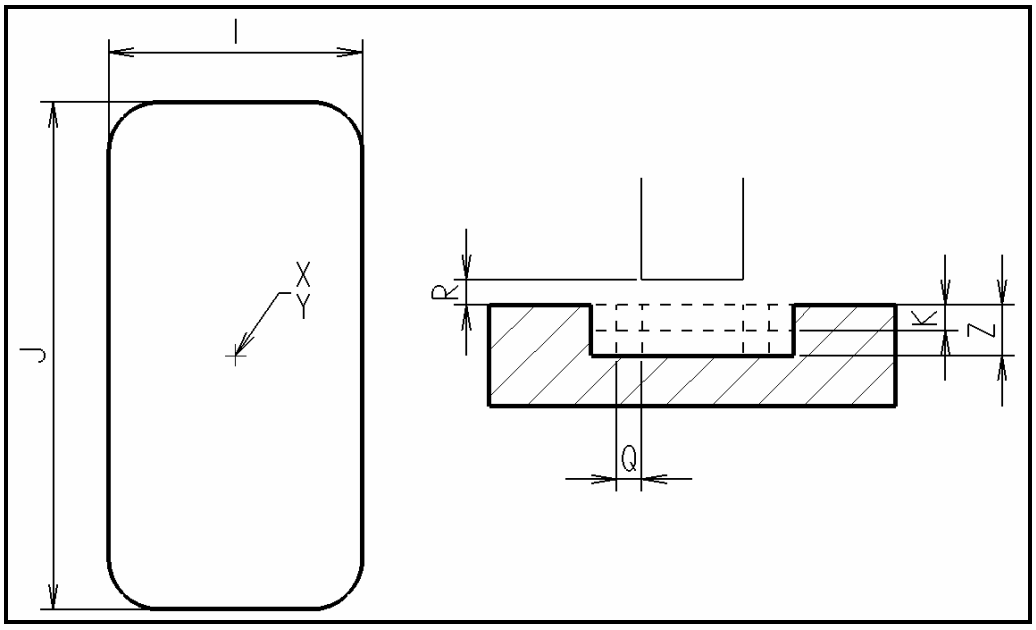
```
(MAKRON LOPETUS)
N3IF[#4010EQ98]GOTO4
G0Z#18
GOTO5
N4G0Z#8
N5G#1G#2F#3
M99
```

Makron lopetus noudattaa samaa kaavaa kuin aikaisemminkin.

11.4 Neliötasku

Toinen yleisesti käytetty makro on neliötasku. Kuten ympyrätaskussa, on tässäkin muutamia huomioitavia asioita.

```
G65P8001X100Y100R2I25J50Z-20Q6K8F200D6(NELIOTASKU)
```



KUVA 10. Neliötaskun muuttujat

- P = MAKRON OHJELMANUMERO
- X = KESKIPISTE X (ABS tai INK) = #24
- Y = KESKIPISTE Y (ABS tai INK) = #25
- R = R-TASO (ABS) = #18
- I = LEVEYS (X-SUUNTA) = #4
- J = LEVEYS (Y-SUUNTA) = #5
- Z = TASKUN SYVYYS = #26
- Q = LASTUN LEVEYS = #17
- K = LASTUN SYVYYS = #6
- F = JYRSINTÄSYÖTTÖ = #9

```
O8001(NELIOTASKU)

(LAHTOTIETOJEN HAKU)
#1=#4001
#2=#4003
#3=#4019
IF[#7EQ#0]THEN#7=#4107
#8=#5003

(MUUTTUJIEN TARKISTUS)
IF[ABS[#4/2]-ABS[#17]LT0]THEN#17=ABS[#4/2]
IF[ABS[#5/2]-ABS[#17]LT0]THEN#17=ABS[#5/2]
IF[ABS[#26]-ABS[#6]LT0]THEN#6=ABS[#26]
```

Tarkistetaan lastunleveydet/-syvyydet suhteessa taskun leveyksiin ja syvyyteen.

```
(TASKUN ASENNON MAARITYS)
#11=0
#12=0
IF[ABS[#4/2]-ABS[#5/2]GT0]THEN#11=ABS[ABS[#4/2]-ABS[#5/2]]
IF[ABS[#5/2]-ABS[#4/2]GT0]THEN#12=ABS[ABS[#5/2]-ABS[#4/2]]
```

Määritellään taskun muoto/asento eli ovatko sivun pituudet samat, ja jos eivät ole määritellään, kumpi on pitempi. Laskun perusteella muutetaan muodon jyrsinnän liikepituuksia (#11 = X-suunta ja #12 = Y-suunta) niin, että pitemmän sivun liike on vastaavasti pitempi. Näin leveys- ja korkeussuunnat valmistuvat yhtä aikaa.

```
(ENSIMMAISET LASTUT)
#13=ABS[#6]
#14=#11+#17
#15=#12+#17
#19=#14
#20=#15
#21=#17
```

Määritellään ensimmäisen lastun parametrit ja poistetaan lastun syvyyteen mahdollisesti annettu etumerkki sekä määritellään jyrsityn leveyden seuranta-arvo (#14).

```
(ABS/INK TARKISTUS)
IF[#2EQ90]GOTO1
#24=#5001+#24
#25=#5002+#24

(PAIKOITUS)
N1G90G0X#24Y#25
Z#18

(TASKUN SYVYYSKIERTO)
WHILE[-#13GE#26]DO1
G90G1Z-#13F[#9/2]
```

```
G91G41G1X#14F#9D#7
Y#15
X-[#14*2]
Y-[#15*2]
X[#14*2]
Y#15
G90G40
IF[#14EQ[#4/2]]GOTO2
#14=#14+#17
#15=#15+#17
IF[#14GT[#4/2]]THEN#17=#17-[#14-[#4/2]]
IF[#14GT[#4/2]]THEN#14=[#4/2]
IF[#15GT[#5/2]]THEN#15=[#5/2]
```

Ensimmäinen lastu jyrsitään ennen varsinaista taskunjyrsintäkiertoa, koska neliötaskussa ensimmäisen lastun liike voi olla suurempi kuin seuraavien, joten kompensointiliike on erilainen. Tämän takia ensimmäinen kierto on silmukan ulkopuolella.

```
(TASKUNJYRSINTAKIERTO)
WHILE[#14LE[#4/2]]DO2
G91G41G1X#17F#9D#7
Y#15
X-[#14*2]
Y-[#15*2]
X[#14*2]
Y#15
G90G40
IF[#14EQ[#4/2]]GOTO2
#14=#14+#17
#15=#15+#17
IF[#14GT[#4/2]]THEN#17=#17-[#14-[#4/2]]
IF[#14GT[#4/2]]THEN#14=[#4/2]
IF[#15GT[#5/2]]THEN#15=[#5/2]
END2
```

Jyrsitään muoto ja tarkistetaan sen koko. Sen perusteella tehdään lastun leveyksiin vastaavat muutokset,kuin ympyrätaskussa.

```
N2G90G1G40X#24Y#25
#14=#19
#15=#20
#17=#21
G1G40X#24Y#25
IF[-#13EQ#26]GOTO3
#13=#13+#6
IF[-#13LT#26]THEN#13=ABS[#26]
END1
```

Syvyys tarkistetaan kuten ympyrätaskussa.

```
(MAKRON LOPETUS)
N3IF[#4010EQ98]GOTO4
G0Z#18
GOTO5
N4G0Z#8
N5G#1G#2F#3
M99
```

Normaali lopetus.

12 YHTEENVETO

Tutkintotyössä käytiin aluksi läpi koneessa olevat erilaiset muuttujat ja niiden tarkoitus. Lisäksi sivuttiin myös lyhyesti mahdollisuutta vaikuttaa ohjelman ajon aikaiseen suoritustapaan.

Koska parametriohjelmoinnin perusteisiin kuuluu muuttujien käsittely ohjelman aikana, käytiin läpi erilaisia toimintoja, mitä niille voidaan tehdä. Niiden käsittelyssä syntyviä virheitä ja niiden vaikutusta ohjelmointiin esiteltiin myös. Seuraavaksi käsiteltiin silmukkakomennot sekä niiden käyttö ja rajoitukset. Viimeisenä ennen esimerkkiohjelmia esiteltiin myös erilaiset mahdollisuudet kutsua makro käyttöön.

Parametriohjelmoinnin perusrakenne ja tyypilliset piirteet esiteltiin neljän perusesimerkin avulla. Niissä ei käytetty kaikkia teoriaosuuden mahdollisuuksia toteuttaa ohjelma, vaan pyrittiin antamaan muutamia perusratkaisuja erilaisten ohjelmointitilanteiden ratkaisuun.

Tämän opinnäytetyön tarkoitus on ollut esitellä perusohjelmoinnin hallitsevalle käyttäjälle muuttujaohjelmoinnin tarjoamia mahdollisuuksia ja uskon, että hänellä on tämän työn avulla paremmat mahdollisuudet hyödyntää NC-ohjauksen tarjoamia ominaisuuksia.

LÄHTEET

GE Fanuc Automation Europe, Computer Numeric Controls Series 21i/210i-MB Operator’s Manual, s.284 – 342.

Heidenhain, TNC426 B / TNC 430 User Manual, s. 224 – 248

LIITTEET

LIITE 1 – Parametrilistaus /1, s.289 – 290/

LIITE 2 – Ellipsimakro

LIITE 3 – Pallopinnan viimeistelymakro

LIITE 4 – Ympyrätaskun jyrsintämakro

LIITE 5 – Neliötaskun jyrsintämakro

TAULUKKO 13. Työkalun kompensointitietoparametrit – Tyyppi A

Korjainnumero	Järjestelmäparametri
1	#10001 (#2001)
:	:
200	#10200 (#2200)
:	:
400	#10400

TAULUKKO 14. Työkalun kompensointitietoparametrit – Tyyppi B

Korjainnumero	Geometriakorjain	Kulumiskorjain
1	#11001 (#2001)	#10001 (#2001)
:	:	:
200	#11200 (#2200)	#10200 (#2200)
:	:	:
400	#11400	#10400

TAULUKKO 15. Työkalun kompensointitietoparametrit – Tyyppi C

Korjain-numero	Pituuskorjain		Sädekorjain	
	Geom.	Kulumis.	Geom.	Kulumis.
1	#11001 (#2001)	#10001 (#2001)	#13001	#12001
:	:	:	:	:
200	#11200 (#2200)	#10200 (#2200)	:	:
:	:	:	:	:
400	#11400	#10400	#13400	#12400

TAULUKKO 16. Modaalisten koodien parametrit

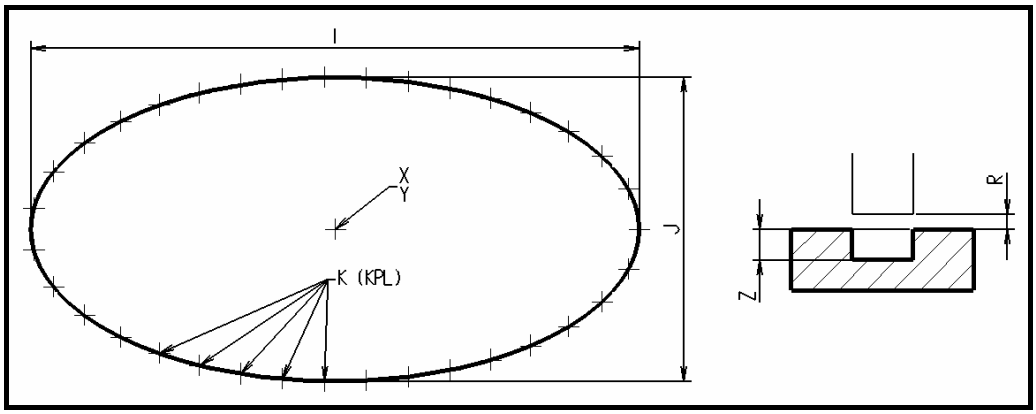
Parametri	Parametrin sisältämä koodinumero	
#4001	G00, G01, G02, G03, G33	(RYHMÄ 1)
#4002	G17, G18, G19	(RYHMÄ 2)
#4003	G90, G91	(RYHMÄ 3)
#4004		(RYHMÄ 4)
#4005	G94, G95	(RYHMÄ 5)
#4006	G20, G21	(RYHMÄ 6)
#4007	G40, G41, G42	(RYHMÄ 7)
#4008	G43, G44, G49	(RYHMÄ 8)
#4009	G73, G74, G76, G80-G89	(RYHMÄ 9)
#4010	G98, G99	(RYHMÄ 10)
#4011	G50, G51	(RYHMÄ 11)
#4012	G66, G67	(RYHMÄ 12)
#4013	G96, G97	(RYHMÄ 13)
#4014	G54-G59	(RYHMÄ 14)
#4015	G61-G6-4	(RYHMÄ 15)
#4016	G68, G69	(RYHMÄ 16)
:		
#4022		(RYHMÄ 22)
#4102	B-koodi	
#4107	D-koodi	
#4109	F-koodi	
#4111	H-koodi	
#4113	M-koodi	
#4114	Vaihenumero	
#4115	Ohjelmanumero	
#4119	S-koodi	
#4120	T-koodi	
#4130	P-koodi	
Esimerkiksi #4001 arvo voi olla 1, 2, 3 tai 33		

TAULUKKO 17. Sijaintitietojen parametrit

Parametrit	Sijaintitieto	Koordinaatti-järjestelmä	Työkalu-korjain	Luku mahdollista liikkeen aikana
#5001-#5004	Lauseen loppupiste	Kappale-koordinaatisto	Ei ole mukana	Kyllä
#5021-#5024	Hetkellinen sijainti	Kone-koordinaatisto	Sisältyy arvoihin	Ei
#5041-#5044	Hetkellinen sijainti	Kappale-koordinaatisto		
#5061-#5064	Ohituspisteen sijainti			Kyllä
#5081-#5084	Työkalun pituuskorjaimen arvo			Ei
#5101-#5104	Servon asema			

TAULUKKO 18. Nollapisteiden parametrit

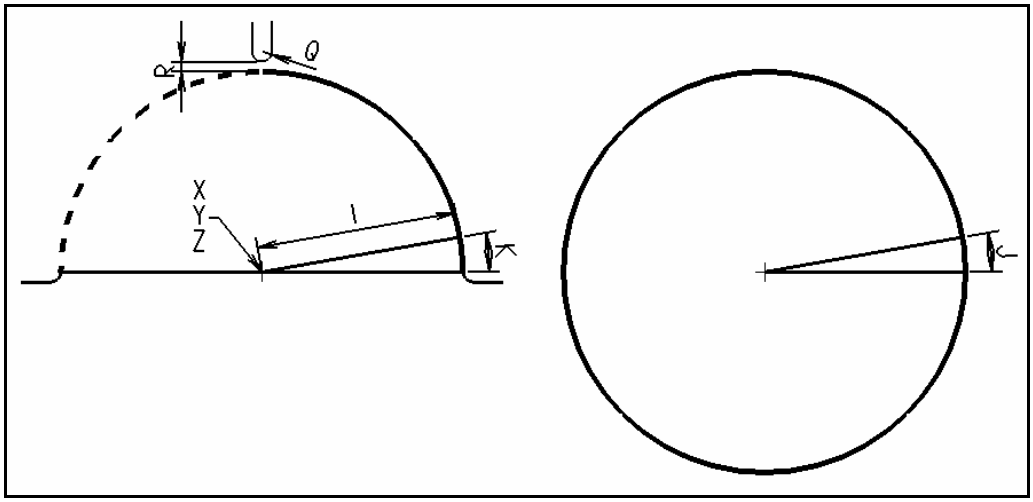
Akseli	Tarkoitus	Parametri
Ensimmäinen akseli X-akseli	Perussiirron arvo (EXT)	#5201
	G54 nollapisteen siirtoarvo	#5221
	G55 nollapisteen siirtoarvo	#5241
	G56 nollapisteen siirtoarvo	#5261
	G57 nollapisteen siirtoarvo	#5281
	G58 nollapisteen siirtoarvo	#5301
	G59 nollapisteen siirtoarvo	#5321
Toinen akseli Y-akseli	Perussiirron arvo (EXT)	#5202
	G54 nollapisteen siirtoarvo	#5222
	G55 nollapisteen siirtoarvo	#5242
	G56 nollapisteen siirtoarvo	#5262
	G57 nollapisteen siirtoarvo	#5282
	G58 nollapisteen siirtoarvo	#5302
	G59 nollapisteen siirtoarvo	#5322
Kolmas akseli Z-akseli	Perussiirron arvo (EXT)	#5203
	G54 nollapisteen siirtoarvo	#5223
	G55 nollapisteen siirtoarvo	#5243
	G56 nollapisteen siirtoarvo	#5263
	G57 nollapisteen siirtoarvo	#5283
	G58 nollapisteen siirtoarvo	#5303
	G59 nollapisteen siirtoarvo	#5323
Neljäs akseli A-akseli	Perussiirron arvo (EXT)	#5204
	G54 nollapisteen siirtoarvo	#5224
	G55 nollapisteen siirtoarvo	#5244
	G56 nollapisteen siirtoarvo	#5264
	G57 nollapisteen siirtoarvo	#5284
	G58 nollapisteen siirtoarvo	#5304
	G59 nollapisteen siirtoarvo	#5324



KUVA 11. Ellipsin muuttujat

- P = MAKRON OHJELMANUMERO
- X = KESKIPISTE X (ABS) = #24
- Y = KESKIPISTE Y (ABS) = #25
- Z = MUODON SYVYYS Z (ABS) = #26
- I = ELLIPSIN LEVEYS = #4
- J = ELLIPSIN KORKEUS = #5
- K = LASKENTAPISTEIDEN MÄÄRÄ = #6
- R = R-TASO = #18
- F = JYRSINTÄSYÖTTÖ = #9

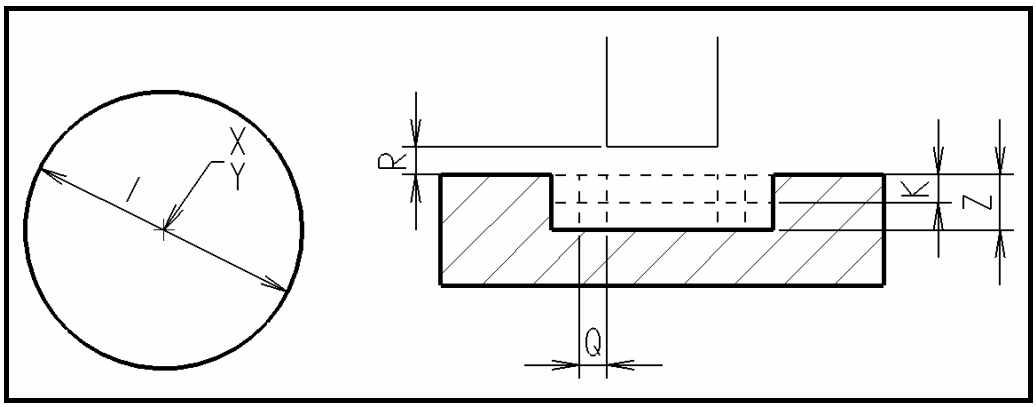
```
O8003(ELLIPSI)
(LAHTOTIETOJEN HAKU)
#1=#4001
#2=#4003
#3=#4019
#6=360/#6
#7=#5003
(ABS/INK TARKISTUS)
IF[#2EQ90]GOTO1
#24=#5001+#24
#25=#5002+#24
(ALOITUSPARAMETRIT)
N1
#4=#4/2
#5=#5/2
#10=0(TYOSTOKULMA XY)
(PAIKOITUS)
#11=#4*COS[#10]
#12=#3*SIN[#10]
G90G0X[#24+#11]Y[#25+#12]
Z#18
G1Z#26F[#9/2]
#10=#10+#6
(RATAKIERTO)
WHILE[#10LE360]DO1
G1F#9
#11=#4*COS[#10]
#12=#3*SIN[#10]
G90G1X[#24+#11]Y[#25+#12]
#10=#10+#6
END1
(MAKRON LOPETUS)
IF[#4010EQ98]GOTO2
G0Z#18
GOTO3
N2G0Z#7
N3G#1G#2F#3
M99
```



KUVA 12. Pallopinnan muuttujat

- P = MAKRON OHJELMANUMERO
- X = KESKIÖPISTE X (ABS) = #24
- Y = KESKIÖPISTE Y (ABS) = #25
- Z = KESKIÖPISTE Z (ABS) = #26
- I = PALLON SÄDE = #4
- J = KULMA-ASKEL (XY-SUUNTA) = #5
- K = KULMA-ASKEL (XZ-SUUNTA) = #6
- R = R-TASO (PALLON YLÄPUOLELLA, INK) = #18
- Q = PALLOPÄISEN TERÄN SÄDE = #17
- F = JYRSINTÄSYÖTTÖ = #9

```
O8002(PALLOPINTA)
(LAHTOTIETOJEN HAKU)
#1=#4001
#2=#4003
#3=#4019
#7=#5003
(PAIKOITUS)
G90G0X#24Y#25
Z[#26+#18+#17+#4]
(ALOITUSPARAMETRIT)
#10=0(TYOSTOKULMA XY)
#11=90(TYOSTOKULMA XZ)
#12=#4+#17(OHJELMOITAVAN PALLON SADE)
(KEHAKIERTO)
WHILE[#10LT360]DO1
G90G0X#24Y#25
Z[#26+#17+#4]
(PINTAKIERTO)
WHILE[#11GE0]DO2
G1F#9
#13=#12*SIN[#11](Z-KOORDINAATTI)
#14=#12*COS[#11](APULUKU)
#15=#14*SIN[#10](Y-KOORDINAATTI)
#16=#14*COS[#10](X-KOORDINAATTI)
G1X[#24+#16]Y[#25+#15]Z[#26+#13]
#11=#11-#6
END2
G0Z[#26+#18+#17+#4]
#10=#10+#5
#11=90
END1
(MAKRON LOPETUS)
IF[#4010EQ98]GOTO1
G0Z[#26+#18+#17+#4]
GOTO2
N1G0Z#7
N2G#1G#2F#3
M99
```



KUVA 13. Ympyrätaskun muuttujat

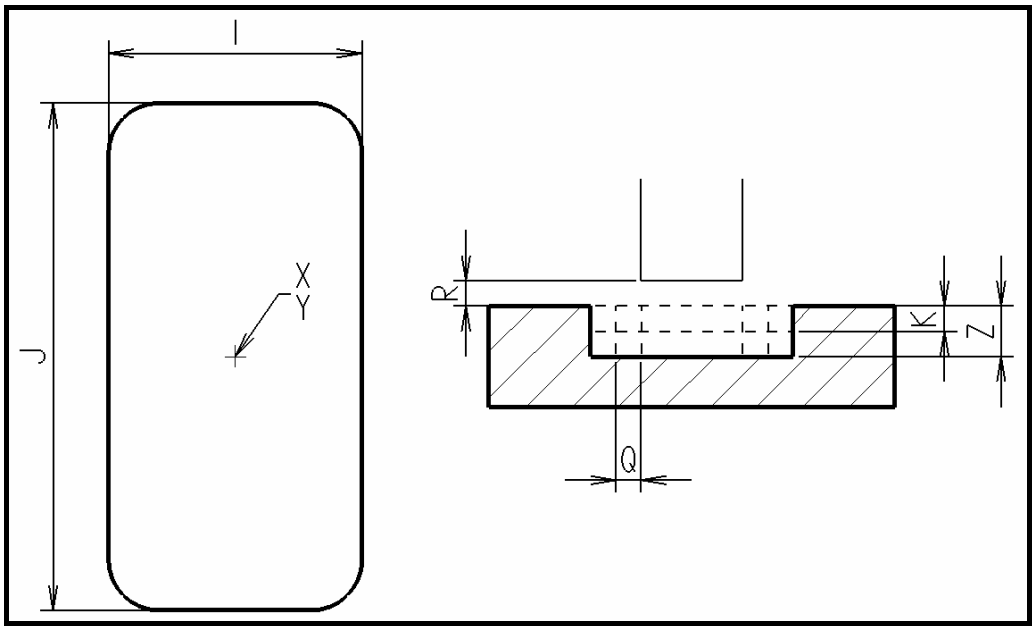
- P = MAKRON OHJELMANUMERO
- X = KESKIPISTE X (ABS tai INK) = #24
- Y = KESKIPISTE Y (ABS tai INK) = #25
- R = R-TASO (ABS) = #18
- I = HALKAISIJA = #4
- Z = TASKUN SYVYYS = #26
- Q = LASTUN LEVEYS = #17
- K = LASTUN SYVYYS = #6
- F = JYRSINTÄSYÖTTÖ = #9

```
O8000(YMPYRATASKU)
(LAHTOTIETOJEN HAKU)
#1=#4001
#2=#4003
#3=#4019
IF[#7EQ#0]THEN#7=#4107
#8=#5003
(MUUTTUJIEN TARKISTUS)
IF[ABS[#4/2]-ABS[#17]LT0]THEN#17=ABS[#4/2]
IF[ABS[#26]-ABS[#6]LT0]THEN#6=ABS[#26]
 (ENSIMMAISET LASTUT)
#13=ABS[#6]
#14=#17
#19=#14
(ABS/INK TARKISTUS)
IF[#2EQ90]GOTO1
#24=#5001+#24
#25=#5002+#24
 (PAIKOITUS)
N1G90G0X#24Y#25
Z#18
(TASKUN SYVYYSKIERTO)
WHILE[-#13GE#26]DO1
G90G1Z-#13F[#9/2]
(TASKUNJYRSINTAKIERTO)
WHILE[#14LE[#4/2]]DO2
G91G1G41X#17F#9D#7
G3X0Y0I-#14F#9
G1G40
IF[#14EQ[#4/2]]GOTO2
#14=#14+#17
IF[#14GT[#4/2]]THEN#17=#17-[#14-[#4/2]]
IF[#14GT[#4/2]]THEN#14=[#4/2]
END2
```

jatkuu …

… jatkuu

```
N2G90G1G40X#24Y#25
#14=#19
#17=#19
IF[-#13EQ#26]GOTO3
#13=#13+#6
IF[-#13LT#26]THEN#13=ABS[#26]
END1
(MAKRON LOPETUS)
N3IF[#4010EQ98]GOTO4
G0Z#18
GOTO5
N4G0Z#8
N5G#1G#2F#3
M99
```



KUVA 14. Neliötaskun muuttujat

- P = MAKRON OHJELMANUMERO
- X = KESKIPISTE X (ABS tai INK) = #24
- Y = KESKIPISTE Y (ABS tai INK) = #25
- R = R-TASO (ABS) = #18
- I = LEVEYS (X-SUUNTA) = #4
- J = LEVEYS (Y-SUUNTA) = #5
- Z = TASKUN SYVYYS = #26
- Q = LASTUN LEVEYS = #17
- K = LASTUN SYVYYS = #6
- F = JYRSINTÄSYÖTTÖ = #9

```
O8001(NELIOTASKU)
(LAHTOTIETOJEN HAKU)
#1=#4001
#2=#4003
#3=#4019
IF[#7EQ#0]THEN#7=#4107
#8=#5003
(MUUTTUJIEN TARKISTUS)
IF[ABS[#4/2]-ABS[#17]LT0]THEN#17=ABS[#4/2]
IF[ABS[#5/2]-ABS[#17]LT0]THEN#17=ABS[#5/2]
IF[ABS[#26]-ABS[#6]LT0]THEN#6=ABS[#26]
(TASKUN ASENNON MAARITYS)
#11=0
#12=0
IF[ABS[#4/2]-ABS[#5/2]GT0]THEN#11=ABS[ABS[#4/2]-ABS[#5/2]]
IF[ABS[#5/2]-ABS[#4/2]GT0]THEN#12=ABS[ABS[#5/2]-ABS[#4/2]]
(ENSIMMAISET LASTUT)
#13=ABS[#6]
#14=#11+#17
#15=#12+#17
#19=#14
#20=#15
#21=#17
(ABS/INK TARKISTUS)
IF[#2EQ90]GOTO1
#24=#5001+#24
#25=#5002+#24
(PAIKOITUS)
N1G90G0X#24Y#25
Z#18
(TASKUN SYVYYSKIERTO)
WHILE[-#13GE#26]DO1
G90G1Z-#13F[#9/2]
G91G41G1X#14F#9D#7
Y#15
X-[#14*2]
Y-[#15*2]
X[#14*2]
Y#15
```

jatkuu …

… jatkuu

```
G90G40
IF[#14EQ[#4/2]]GOTO2
#14=#14+#17
#15=#15+#17
IF[#14GT[#4/2]]THEN#17=#17-[#14-[#4/2]]
IF[#14GT[#4/2]]THEN#14=[#4/2]
IF[#15GT[#5/2]]THEN#15=[#5/2]
(TASKUNJYRSINTAKIERTO)
WHILE[#14LE[#4/2]]DO2
G91G41G1X#17F#9D#7
Y#15
X-[#14*2]
Y-[#15*2]
X[#14*2]
Y#15
G90G40
IF[#14EQ[#4/2]]GOTO2
#14=#14+#17
#15=#15+#17
IF[#14GT[#4/2]]THEN#17=#17-[#14-[#4/2]]
IF[#14GT[#4/2]]THEN#14=[#4/2]
IF[#15GT[#5/2]]THEN#15=[#5/2]
END2
N2G90G1G40X#24Y#25
#14=#19
#15=#20
#17=#21
G1G40X#24Y#25
IF[-#13EQ#26]GOTO3
#13=#13+#6
IF[-#13LT#26]THEN#13=ABS[#26]
END1
(MAKRON LOPETUS)
N3IF[#4010EQ98]GOTO4
G0Z#18
GOTO5
N4G0Z#8
N5G#1G#2F#3
M99
```