Opinnäytetyö (AMK)

Kone- ja tuotantotekniikka

Koneautomaatio

2016

Lasse Tuominen

MAKRO-OHJELMOINTI TYÖSTÖKONEISSA

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Kone- ja tuotantotekniikka

11.12.2016 | 33

Lasse Tuominen

MAKRO-OHJELMOINTI TYÖSTÖKONEISSA

Opinnäytetyö käsittelee makro-ohjelmointia ja sen hyödyllisyyttä työstökoneissa. Työssä on tarkasteltu makro-ohjelmointiin liittyviä toimintoja ja muuttujia sekä ohjelmien erilaisia käyttökohteita. Lisäksi on esitetty ymmärrystä helpottavia esimerkkejä makro-ohjelmien toiminnasta.

Työn tavoitteena oli luoda materiaali, jota voidaan käyttää makro-ohjelmoinnin itseopiskelussa sekä oppimateriaalina ammatti- ja ammattikorkeakouluissa. Työ soveltuu oppimateriaaliksi ISO-koodi-ohjelmoinnin perusteet hallitseville. Kokeneemmat ohjelmoijat on huomioitu esittämällä uusia ideoita makrojen käyttösovelluksiin perinteisistä ohjelmointikäytännöistä poiketen.

Työn ohessa valmistetut makro-ohjelmat ovat sellaisenaan käyttökelpoisia monissa konepajoissa ja kouluissa. Osa ohjelmista sisältyy esimerkkeihin ja osa on työn liitteenä. Näiden makro-ohjelmien käyttöön ottamisella on helppo päästä alkuun makro-ohjelmoinnissa ja lisätä ohjelmoinnin tehokkuutta.

ASIASANAT:

Makro-ohjelmointi, ISO-koodi ohjelmointi, muuttuja ja NC-ohjelmointi

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Mechanical and Production Engineering

11.12.2016 | 33

Author Lasse Tuominen

MACRO PROGRAMING IN MACHINE TOOLS

The focus of this thesis was on macro programming and programming's utility in machine tools. The programming features and variables were also studied. Macro programming is easier to understand through examples and therefore example programs were included in the thesis. Moreover, different applications of macro programming were studied during the project.

The main purpose was to create self-study material of macro programming that can be used in vocational schools and in universities of applied sciences. In the thesis it is supposed that the user of the machinery has basic skills of ISO-code programming. More experienced users have been taken into consideration by including new ideas for the macro applications.

Macro programs which are presented as examples are applicable in machine tools of schools and engineering factories. Included macro programs are enabling an easy start of programming and improving the efficiency in programming practices. All of the presented example programs were tested to confirm proper functionality.

KEYWORDS:

Macro programing, custom macro, NC-programing, machine tools program.

SISÄLTÖ

LIITTEET

KUVAT

TAULUKOT

KÄYTETYT LYHENTEET JA SANASTO

ISO-koodi	=Yleinen ohjelmointikieli
NC	=Numeerisesti ohjattu
PMC	=Ohjelmoitava koneen ohjain
CAM	=Tietokoneavusteinen valmistus

1 JOHDANTO

Opinnäytetyö käsittelee makro-ohjelmointia työstökoneissa. Työssä keskitytään makrojen toimintaan ja niiden käyttöön työstöohjelmia tehtäessä. Myös koneen toimintoihin liittyvien makrojen toimintaa ja sovelluskohteita käsitellään. Erilaisten esimerkkien avulla on kuvattu makrojen toimintaa. Esimerkkien tarkoituksena on herättää uusia ideoita ja tapoja makro-ohjelmoinnin hyödyntämiseen.

Työ on suunnattu ISO-koodi-ohjelmoinnin perusteet hallitseville henkilöille, jotka haluavat kehittää osaamistaan. Parhaiten työ palvelee kokeneempaa ohjelmoijaa, joka tekee päivittäin ohjelmia, mutta myös kunnossapidossa työskentelevä henkilö voi hyödyntää työtä omassa työssään. Työ soveltuu käytettäväksi ammatti- ja ammattikorkeakoulujen opintoihin.

Työn avulla on mahdollista opetella itsenäisesti makro-ohjelmointia ja saada uusia ideoita makrojen käyttöön. Makro-ohjelmointia onkin lähestytty työssä useista suunnista, niin työstöratojen luonnin kuin koneen toimintojen osalta. Työn keskeisimpänä tavoitteena onkin, että sitä käytettäessä ja sovellettaessa teollisuudessa voidaan parantaa tuottavuutta ja laatua, mikä on erittäin tärkeä asia kilpailukyvyn kannalta.

2 ISO-KOODI OHJELMOINTI

Perinteisesti työstökoneita ohjataan ISO-koodi-ohjelmoinnin avulla. Puhekielessä puhutaan usein G-koodiohjelmoinnista. Tätä tapaa kutsutaan NC-ohjelmoinniksi, joka tarkoittaa numeerista ohjelmointia. ISO-koodi-ohjelmoinnin lisäksi on olemassa muitakin NC-ohjelmointikieliä, kuten esimerkiksi Mazatrol, Heidenhain ja Fagor. Ohjelmoinnissa käytettävät koodit ovat suurelta osin vakiintuneet eri ohjauksien välillä. Ohjelmat koostuvat erilaisista koodeista ja niille annetuista numeerisista arvoista. ISO-koodia käyttävien ohjausten ohjelmointikieli on pääosin samanlaista ohjauksen merkistä riippumatta.

Ohjelma koostuu lauseista ja lauseet sanoista. Lause päättyy aina puolipisteeseen ja mahtuu yleensä yhdelle riville. Yksittäinen koodi tai koordinaatti vastaa yhtä sanaa lauseessa. Esimerkiksi lause G1 X10; koostuu kahdesta sanasta, joista muodostuu yksi lause. Lause voi muodostua yhdestäkin sanasta.

G-koodeja on useita erilaisia moniin eri käyttötarkoituksiin. Koodit on listattu koneen manuaaleihin ja tietyille numeroille on omat toimintonsa. G-koodeilla kerrotaan ohjaukselle, miten konetta halutaan liikuttaa, ja minkä nollapisteen mukaan. Myös erilaiset työkierrot kutsutaan niiden avulla.

M-koodeilla kytketään päälle tai pois päältä erilaisia koneen toimintoja. Niistä useimmat ovat standardisoituja. Työkalunvaihdot ja muut vastaavat koneen toistuvat toiminnot kutsutaan M-koodien avulla.

Taulukko 1. Esimerkki ISO-koodi ohjelmasta.

O1000	
G90 G80 G49 G40;	Otetaan käyttöön absoluuttinen ohjelmointi. Poistetaan säde- ja pituuskompensoinnit, sekä työkiertotiedot.
T10;	Esivalitaan työkalu 10
M6;	Vaihdetaan työkalu
G54;	Otetaan käyttöön nollapiste
S1000 M3;	Käynnistetään kara nopeudella 1000kier./min
G0 X0 Y0 M8;	Paikoitetaan pikaliikkeellä sijaintiin X0 Y0.

	Käynnistetään leikkuuneste pumppu.
G43 H10 Z0;	Otetaan käyttöön työkalun pituuskompensointi arvo kohdasta 10, paikoitetaan Z0
G1 F200 X500;	Ajetaan syöttöliikkeellä X-akselia 500mm nopeudella 200mm/min
G0 Z100 M9;	Paikoitetaan Z100 pikaliikkeellä ja pysäytetään leikkuuneste pumppu.
M5;	Pysäytetään kara.
M30;	Lopetetaan ohjelma.

3 MAKRO-OHJELMOINTI

Normaalissa NC-ohjelmassa arvot annetaan aina numeroin, jolloin jokainen arvo joudutaan antamaan erikseen. Tämän takia ohjelmista tulee pitkiä ja ohjelmoinnista hidasta. Makro-ohjelmoinnissa arvot voidaan korvata muuttujilla. Muuttujien arvoja voidaan laskea erilaisten matemaattisten operaatioiden avulla.

Usein makro-ohjelmia käytetään pääohjelmassa aliohjelmina. Tietynlaisille muodoille, työvaiheille tai paikoituksille on mahdollista tehdä omat makroaliohjelmat, joissa muuttujien avulla annetaan arvot ja mitat. Esimerkiksi ympyrätaskunjyrsinnälle voidaan tehdä oma makroaliohjelma. Parhaimmillaan pääohjelma koostuu pelkistä makroaliohjelmien kutsuista, joissa määritellään vain arvot ja mitat muuttujille. Tällöin ohjelmista tulee hyvin lyhyitä ja ohjelmointi on nopeaa.

Nykyään käsiohjelmoinnista on siirrytty paljon CAM-ohjelmointiin. CAM-ohjelmoinnissa ohjelma tehdään tietokoneohjelmalla mahdollisesti jo valmiina olevan 3D-mallin pohjalta. Tämä on hyvä ja tehokas tapa tehdä ohjelmia, mutta makro-ohjelmoinnillakin on omat hyvät ja tehokkaat puolensa. Kun työstettävän kappaleen muodot ovat kohtuullisen yksinkertaisia ja koneessa on hyvät makrot, voi makro-ohjelmointi olla nopeampaa kuin CAM-ohjelmointi. Makrojen käyttöä voi puoltaa myös koneen rajallinen muistikapasiteetti, sillä CAM-ohjelma tekee ohjelmista yleensä hyvin pitkiä, jolloin ohjelma kuormittaa koneen muistia. Lisäksi CAM-ohjelman korkea hinta teettää usein ongelmia. Makro-ohjelmointia joudutaan joka tapauksessa käyttämään koneen omiin toimintoihin liittyviin ohjelmiin, sillä ne käyttävät muuttujia. Työkalunvaihto-, paletinvaihto- ja mittausohjelmien teko olisi mahdotonta ilman makro-ohjelmoinnin mahdollistamia muuttujia. Esimerkiksi mittausohjelmassa koneen pysähdyttyä mitta-anturin kosketukseen, on akselin sijainti luettava sen muuttujan arvosta.

4 MUUTTUJAT

Työstökoneen ohjelmoinnissa lukuarvot annetaan usein numeroina. Makro-ohjelmoinnissa on mahdollista korvata koodien numerot ja koordinaattiakselien arvot muuttujilla. Esimerkiksi paikoitettaessa konetta haluttuun paikkaan käskyllä G1 F500 X0 Y0 voidaan makro-ohjelmassa korvata kaikki numeroarvot muuttujilla G1 F#7 X#10 Y#12. Tällöin ohjelma käyttää lukuarvoina kyseisten muuttujien arvoja. Esimerkiksi muuttujien arvojen ollessa #7=200, #10=50 ja #12=100 vastaa ohjelma samaa kuin G1 F200 X50 Y100.

4.1 Muuttujien jaottelu

Muuttujat on jaoteltu ryhmiin niiden käyttötarkoituksen mukaan. Jaottelu eroaa hieman ohjauksien valmistajien kesken.

Muuttujanumeron tyyppi pitää aina tarkistaa koneenohjauksen käsikirjasta, sillä samanlaisissakin ohjauksissa saattaa olla eroja. Tähän vaikuttaa moni asia, kuten koneen lisävarusteet ja käyttötarkoitus. Yleismuuttujia #100-#199 ja #500-#999 ei aina ole kaikissa peruskoneissa käytössä, toisin kuin monitoimisissa koneissa. Kaikki yleismuuttujat on myös mahdollista saada käyttöön lisäoptiona. Järjestelmämuuttujien käyttö on aina konekohtaista, vaikka yleisesti tietyille toiminnoille onkin vakioitunut omat muuttujansa. (Fanuc 2003, 13-293.)

4.2 Tyhjät muuttujat

Tyhjällä muuttujalla ei ole arvoa, jolloin koneen ohjaus jättää sen huomioimatta. Muuttujan arvo ei ole tyhjä, vaikka sen arvo olisi nolla. Tämä on syytä huomioida ohjelmaa laadittaessa. Tarkastellaan esimerkkiä, jossa kone on edellisellä ohjelmarivillä paikoitettu asemaan X100 Y200 ja seuraavalla ohjelmarivillä on X#0 Y#0. Tällöin ohjauksen luettua rivin, akselien asemassa ei tapahdu mitään muutosta, sillä ohjaus jättää huomioimatta molemmat käskyt, koska ne ovat tyhjiä. Kirjoitettaessa seuraavalle riville X0 Y0, kone paikoittaa akselit pisteeseen X0 Y0. (Fanuc 2003, 298.)

Tyhjän muuttujan arvoa ei voi muuttaa, mutta sen voi lukea. Jos jonkin muuttujan arvon halutaan olevan tyhjä, se voidaan asettaa tyhjäksi lukemalla tyhjä muuttuja. Esimerkiksi luettaessa #5=#0 saa muuttuja #5 arvon tyhjä. Tyhjää muuttujaa onkin kätevä käyttää, jos halutaan, että ohjelmassa jätetään huomioimatta jokin osoite ja sen arvo.

4.3 Paikalliset muuttujat ja muuttujien argumentit

Paikallisia muuttujia käytetään makro-ohjelmien lähtöarvojen tallentamiseen. Tämän tekee helpoksi se, että kutakin paikallista muuttujaa varten on ennalta määrätty tietty kirjain eli argumentti. Tällöin pääohjelman makrokutsuun tarvitsee kirjoittaa vain argumentti ja sen arvo. Paikallisiin muuttujiin voidaan tallentaa tietoja makro-ohjelman sisällä ja niitä voidaan käyttää laskutoimituksissa samanlaisesti kuin muitakin muuttujia. Kun koneesta katkaistaan virrat, tiedot paikallismuuttujista häviävät. Virtakatkon jälkeen paikallismuuttujien alkutila on aina tyhjä. Tämä on hyvä ottaa huomioon ohjelmaa tehtäessä. Esimerkiksi mittauksesta saatuja tietoja ei kannata tallentaa paikallisiin muuttujiin, jos niitä halutaan käyttää myöhemmin. (Mazak 2012, 13-123.)

Taulukko 2. Paikallismuuttujat ja argumentit (Fanuc 2002, 317.).

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	#10	S	#19	Z	#26

4.4 Yleismuuttujat

Yleismuuttujille ei ole mitään tiettyä käyttötarkoitusta. Yleismuuttujat eroavat toisistaan tiedon tallentumisen suhteen virtakatkoksissa. Muuttujat #100-#199 tyhjenevät virtojen katkaisun yhteydessä, kun taas muuttujat #500-#999 eivät tyhjene. Niitä voidaan käyttää vapaasti erilaisissa sovelluksissa. Usein niitä käytetään apuna laskutoimituksissa,

esimerkiksi tiedon tallentamiseen. Makro-ohjelmassa olevien laskurien arvot on yleensä kätevä tallentaa yleismuuttujiin, jolloin voidaan vaikuttaa laskurin tilaan virtakatkon jälkeen. (Mazak 2012, 13-123.)

4.5 Järjestelmämuuttujat

Järjestelmämuuttujat antavat hyvin monenlaisia tietoja muun muassa koneen tilasta, akselien sijainnista, kompensointiarvoista ja nollapistetiedoista. Tiettyjen muuttujien arvot vaikuttavat koneen toimintaan ja käyttäytymiseen.

Järjestelmämuuttujat ovat aina konekohtaisia ja yhteydessä koneen toimintoihin. Niitä käytettäessä onkin varmistettava, että käytetään oikeaa muuttujaa. Väärää muuttujaa käytettäessä saatetaan sekoittaa koneen ohjaus, mikä pahimmassa tapauksessa saattaa aiheuttaa jopa mekaanisia vaurioita.

Työkalun mittatiedot ovat luettavissa ja asetettavissa järjestelmämuuttujista. Esimerkiksi Fanuc 0i ohjauksessa työkalujen pituusmitat on tallennettu muuttujiin #11001 - #11400. Järjestelmämuuttujista on luettavissa ja asetettavissa myös kaikki muut työkalun mittatietoja koskevat arvot, kuten säde ja kompensointiarvot.

Koneen hallintaan liittyvien ohjauspaneelin kytkinten toimintaan voidaan vaikuttaa järjestelmämuuttujien avulla. Muuttujien #3003 ja #3004 arvoilla voidaan vaikuttaa yksittäislauseajoon ja syöttönopeuteen vaikuttavien kytkinten toimintaan. Erityisesti kierteitystyökierrossa nämä toiminnot pitää saada kytkettyä pois päältä, ettei syöttöä pääse muuttamaan työstön aikana. Toimintatavan valinta tapahtuu antamalla muuttujille tietty arvo, joka vastaa tiettyä toimintatapaa. Toimintoja vastaavat arvot on taulukoitu ohjausten ohjekirjoissa.

5 OPERAATTORIT

Operaattorien avulla muuttujien arvoille voidaan tehdä laskutoimituksia. Muuttujan arvo voidaan määritellä tietyn suuruiseksi. Vertailuoperaattorien avulla voidaan vertailla muuttujien arvoja.

5.1 Matemaattiset operaattorit ja funktiot

Matemaattiset operaattorit on kuvattu seuraavassa taulukossa. Operaattorilla annetaan aina arvo muuttujalle, taulukon esimerkeissä #a:lle. Akselin paikoitusarvo voidaan poikkeuksena antaa operaation tuloksena. Esimerkiksi X [=#a*5]. Operaattorin arvoina voi olla lukuarvoja, muuttujia tai lausekkeita. Taulukon esimerkeissä ne on merkitty #b ja #c. Työstökoneissa lauseen pituus on usein rajallinen. Tämä tulee huomioida, jos matemaattisissa operaatioissa arvoja korvataan lausekkeilla. Tällöin operaatiosta tulee helposti niin pitkä, ettei sitä pystytä kirjoittamaan koneen ohjaukseen. Tämä ongelma poistuu, kun lausekkeiden arvot lasketaan ennen operaatiota ja tallennetaan vapaisiin muuttujiin.

Ohjelmoinnissa täytyy käyttää sulkeina hakasulkeita, koska ohjaus huomioi tavallisissa sulkeissa olevat merkit ainoastaan kommentteina. Matemaattisissa lausekkeissa hakasulkeet toimivat normaaleiden laskusääntöjen mukaan.

Taulukko 3. Matemaattiset operaattorit (Fanuc 2003,309.).

Toiminto	Esimerkki
Määritys	#a=#b
Yhteenlasku	#a=#b+#c
Vähennyslasku	#a=#b-#c
Tulo	#a=#b*#c
Osamäärä	#a=#b/#c
Sini	#a=SIN[#b]
Arkussini	#a=ASIN[#b]
Kosini	#a=COS[#b]

Arkuskosini	#a=ACOS[#b]
Tangentti	#a=TAN[#b]
Arkustangentti	#a=ATAN[#b]/[#c]
Neliöjuuri	#a=SQRT[#j]
Itseisarvo	#a=ABS[#b]
Pyöristys	#a=ROUND[#b]
Pyöristys alas	#a=FIX[#b]
Pyöristys ylös	#a=FUP[#b]
Luonnollinen logaritmi	#a=LN[#b]
10-kantainen logaritmi	#a=EXP[#b]

5.2 Loogiset operaattorit

Loogisia operaattoreita tarvitaan harvoin makro-ohjelmia tehdessä, usein niitä ei tarvita lainkaan työstämiseen käytettävien ohjelmien kirjoittamisessa. Loogisten operaattorien käyttö rajoittuukin usein koneen toimintoihin liittyviin makroihin. Seuraavasta taulukosta selviää näiden operaattorien toiminta.

Taulukko 4. Loogiset operaattorit (Fanuc 2003, 309.).

Toiminto	Esimerkki
OR	#a=#bOR#c
XOR	#a=#bXOR#c
AND	#a=#bAND#c

Loogiset operaatiot toimivat samaan tapaan kuin vastaavat digitaaliset piirit. Muuttujia käsitellään siis bitti bitiltä yleisesti tunnetun totuustaulukon mukaan. Seuraavassa taulukossa selviää OR-piirin toiminta.

Taulukko 5. OR-piirin toiminta.

A	B	C
0	0	0
1	0	1
1	1	0
1	1	1

Lisäksi voidaan tehdä BCD- ja BIN-koodimuunnoksia seuraavassa taulukossa esitellyillä operaatioilla. Muunnoksia käytetään lähinnä signaalimuunnoksiin PMC:sä . (Fanuc 2003, 309.)

Taulukko 6. BCD- ja BIN-koodimuunnokset (Fanuc 2003, 309.).

Toiminto	Esimerkki
Muunnos BCD-koodista BIN-koodiin	#a=BIN[#c]
Muunnos BIN-koodista BCD-koodiin	#a=BCD[#c]

5.3 Vertailuoperaattorit

Vertailuoperaattorien avulla voidaan ohjelmassa tehdä erilaisia päätelmiä vertaamalla muuttujan arvoa toiseen muuttujaan tai lukuarvoon. Vertailuoperaattori kirjoitetaan aina hakasulkeisiin. Esimerkiksi: [#a GT #b] tai [#a GT 15]. Ensimmäisessä esimerkissä verrataan muuttujan a arvoa muuttujan b arvoon ja toisessa esimerkissä verrataan muuttujan a arvoa lukuun 15. Vertailuoperaattoreita käytetään aina jonkin päätöksen teon yhteydessä, kuten silmukka- ja hyppykäskyjen yhteydessä.

Taulukko 7. Vertailuoperaattorit (Fanuc 2003, 316.).

Operaattori	Matemaattinen merkki	Vertaus
EQ	=	Yhtä suuri kuin
NE	≠	Eri suuri kuin
GT	>	Suurempi kuin
GE	≥	Suurempi tai yhtä suuri kuin
LT	<	Pienempi kuin
LE	≤	Pienempi tai yhtä suuri kuin

Seuraavasta taulukosta selviää vertailuoperaattorien mahdollisuudet, taulukossa on esitetty kaikki tunnetut matemaattiset vertaukset. Yhtä suuri kuin -vertausta EQ ei tule sekoittaa matemaattiseen operaattoriin, jolla määritetään muuttuja luvun tai toisen muuttujan kanssa yhtä suureksi. (Mazak 2012, 13-155.)

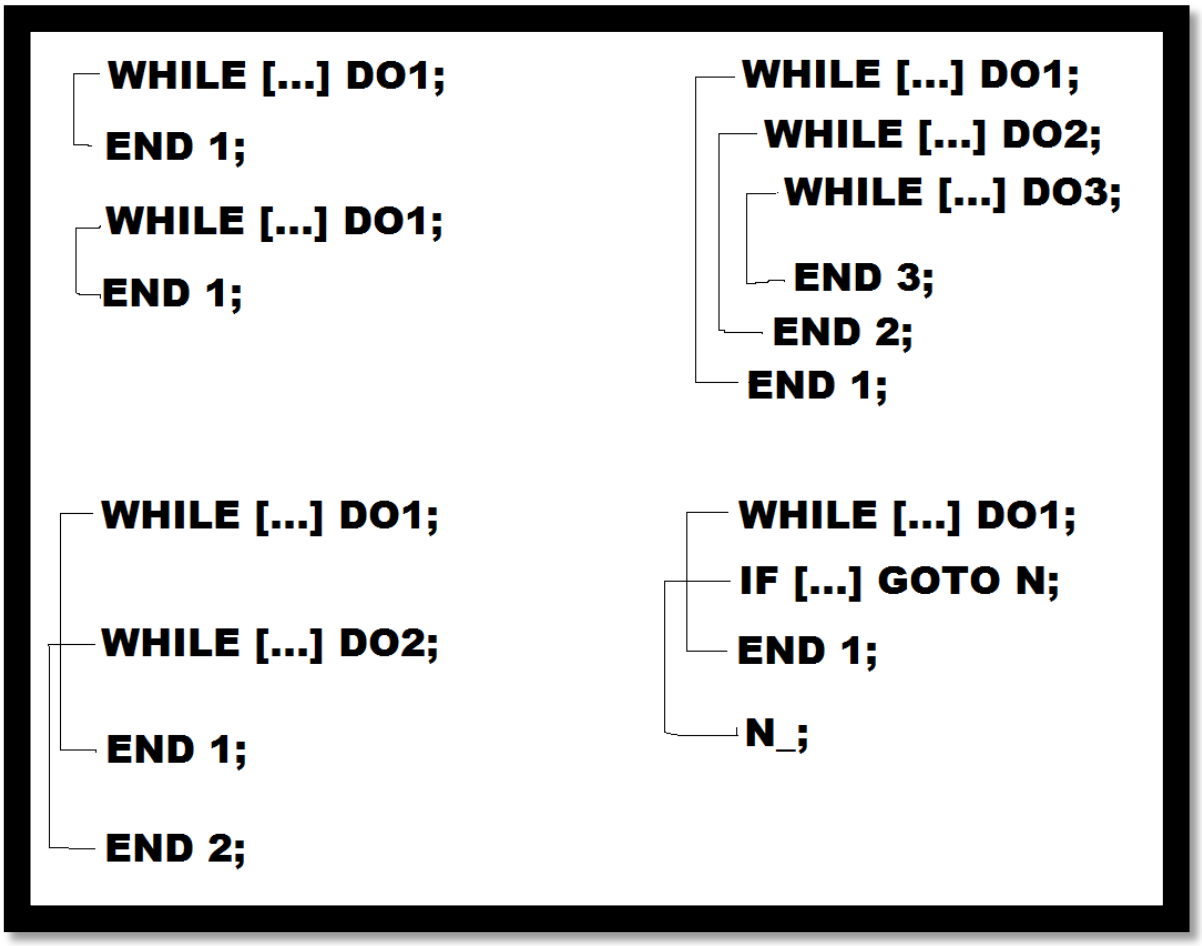
6 SILMUKAT JA HYPPYKÄSKYT

Silmukat ja hyppykäskyt ovat yksi erittäin tärkeä osa makro-ohjelmointia, koska näillä käskyillä ohjelma saadaan toistamaan tiettyjä kohtia ja hyppäämään määriteltyihin kohtiin joko ehdollisesti tai ehdotta.

6.1 Silmukat

Silmukat toteutetaan WHILE ja END -komennoilla. Silmukka alkaa aina WHILE -komennolla ja päättyy END -komentoon tai hyppykäskyyn, joita käsitellään seuraavassa kappaleessa. WHILE -komentoon sisältyy ehtolause, jossa yleensä verrataan jonkin muuttujan arvoa joko lukuun tai toiseen muuttujaan. Tällöin toinen ehtolauseessa oleva muuttuja toimii laskurina, joka laskee toistojen määrän. Laskuri on helppo toteuttaa. Otetaan käyttöön jokin vapaa yleismuuttuja, jonka arvo asetetaan nollaksi ennen silmukkaa ja silmukan lopussa muuttujan arvoon lisätään aina yksi. Ehtolauseen toteutuessa ohjelma hyppää END -komentoa seuraavalle riville.

Silmukoita voidaan ohjelmoida sisäkkäin maksimissaan kolme kappaletta. Ohjelmoitaessa silmukoita sisäkkäin pitää aloitus- ja lopetuskohdat numeroida. Numerointi tulee tehdä ainoastaan numeroilla 1, 2 ja 3. Silmukoista täytyy poistua järjestyksessä. Jos ollaan kolmannessa silmukassa, sieltä pitää poistua ennen kuin voidaan poistua ensimmäisestä tai toisesta silmukasta. Silmukan sisällä voidaan kuitenkin hypätä useita kertoja toiseen tai kolmanteen silmukkaan, jos silmukasta tullaan pois ennen uutta hyppyä. (Mazak 2012, 13-156.)



Kuva 1. Silmukoiden käyttö.

6.2 Hyppykäskyt

Hyppykäskyjä on kahdenlaisia, ehdottomia ja ehdollisia. Ehdoton hyppykäsky GOTO pakottaa ohjelman hyppäämään tietylle riville ohjelmassa. Käskyn perään tulee ohjelmoida rivinumero, johon halutaan hypätä. Käskyn perään voidaan ohjelmoida myös muuttuja, jolloin ohjelma hyppää muuttujan arvoa vastaavalle riville. Ohjelmassa pitää olla ohjelmoituna kyseinen rivinumero N-osoitteeseen. Ehdollinen hyppykäsky on muutoin samanlainen, mutta ennen GOTO -käskyä tulee ehtolause IF, jonka ehdon on toteuduttava ennen kuin hyppy toteutetaan.

IF-käskyn yhteydessä voidaan käyttää myös THEN -käskyä. Tällöin ehdon toteutuessa suoritetaan ainoastaan THEN -käskyn perässä oleva lauseke. (Mazak 2012, 13-155.)

7 OHJELMIEN KUTSUMINEN

7.1 Makro-ohjelman kutsuminen

Makro-ohjelma voidaan kutsua usealla eri tavalla. Tyypillisin makrokutsu on jokin koneen vakiotoiminto. Esimerkiksi työkalua vaihdettaessa kutsutaan työkalua T:llä ja sen numerolla. Työkalun vaihto tapahtuu koodilla M6, jolloin M6 kutsuu tietyn makro-ohjelman, jossa muuttujana on annettu työkalun numero. Tätä toimintoa kutsutaan M-makrokutsuksi. Koneessa olevat vakiotyökierrot kutsutaan G-makrokutsulla. Tällöin tietyllä G-koodilla kutsutaan ennalta määritettyä ohjelmanumeroa, johon makro on kirjoitettu.

Itse tehtyjä makro-ohjelmia voidaan kutsua yksinkertaisella makrokutsulla tai modaalisella makrokutsulla, jolloin makro toistetaan paikoituksen jälkeen. Nämä ovatkin yleisimmät tavat itse tehtyjä makroja kutsuttaessa. Koneissa on kuitenkin usein muutamia vapaita G- ja M-koodeja, joille on koodattu valmiiksi oma ohjelmanumero. Näille ohjelmanumeroille on mahdollista kirjoittaa omia makro-ohjelmia, joita voi kutsua M- tai G-koodilla.

7.2 Yksinkertainen makrokutsu

Kun makro-ohjelma tarvitsee kutsua vain kerran tiettyyn kohtaan, kannattaa käyttää yksinkertaista makrokutsua G65. Kutsu tapahtuu komennolla G65, jonka yhteydessä annetaan makro-ohjelman numero. Samaan lauseeseen tulee antaa kaikki tarvittavat muuttujien lähtötiedot. Esimerkiksi G65 P3000 F0.5 Z3 H10;.

7.3 Modaalinen makrokutsu

Tehtäessä useita toistoja samalla makrolla ja samoilla lähtöarvoilla eripaikoissa, on kannattavaa käyttää modaalista makrokutsua. Kutsuttaessa makro modaalisesti komennolla G66 toistetaan makro jokaisen kutsua seuraavan paikoituksen jälkeen. Makron käyttö voidaan lopettaa komennolla G67. Näin ollen modaalisen makrokutsun käyttö poikkeaa yksinkertaisesta makrokutsusta vain toistettavuuden osalta.

7.4 Tavallisen aliohjelman kutsuminen makro-ohjelmaan

On harvinaista, että makro-ohjelmaan kutsutaan tavallista aliohjelmaa, mutta tietyissä tilanteissa se on erittäin kätevää. Yksinkertaisille muodoille, kuten neliö- ja ympyrätaskuille, tarkoitetut rouhintamakrot voidaan tehdä matemaattisten funktioiden ja hyppykäskyjen avulla perinteistä makro ohjelmointia käyttäen. Usein tämä ei kuitenkaan riitä, kun kappaleet ovat epäsäännöllisen muotoisia ja muotoja on paljon.

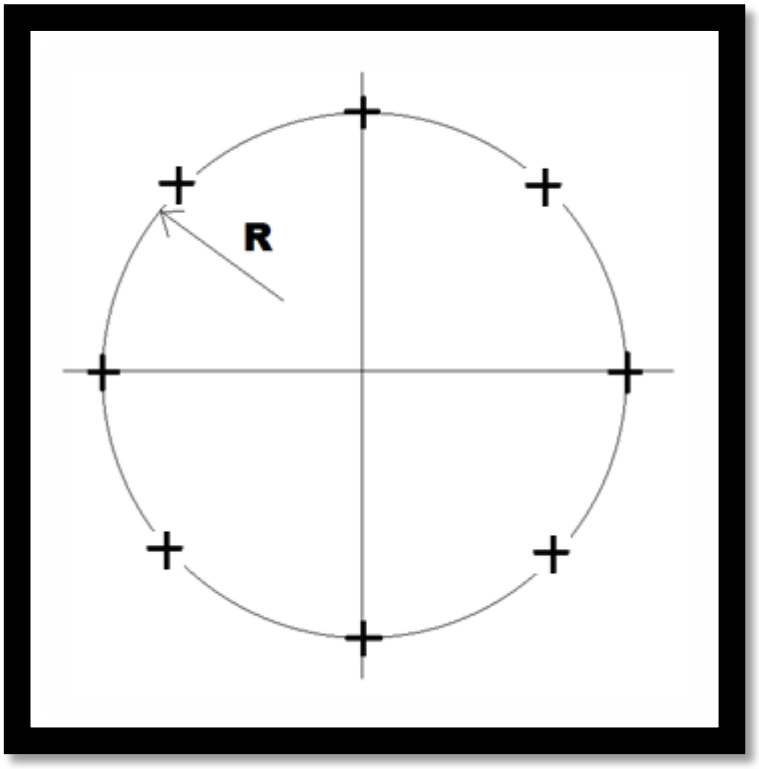
Makron sisältä on kätevää kutsua tavallista aliohjelmaa, johon muodon rata on kirjoitettu koordinaatein. Makro-ohjelmassa lasketaan terälle kompensointiarvoja annettujen parametrien mukaan. Parametreina voidaan käyttää lastun vahvuutta ja määrää sivu- ja syvyyssuunnassa. Tällöin saadaan aikaan muotoa toistava rouhintamakro, joka olisi lähes mahdoton toteuttaa perinteistä makro-ohjelmointia käyttäen.

Samaa menetelmää käyttämällä voidaan tehdä makroja, joilla saadaan aikaan pyöristyksiä muodon reunoihin. Lisäksi on mahdollista tehdä makro, joka tekee muodon kyljistä kaltevat. Tämä on kätevää erityisesti muotteja tehtäessä, kun työstetään muotin päästöllisiä pintoja. Muodon poikkileikkauksesta voidaan tehdä lähes millainen vain, kunhan muodon sivu- ja pystysuuntaiset koordinaatit ovat matemaattisesti laskettavissa.

Koordinaatit ovat määritettävissä esimerkiksi tilanteessa, jossa pallopäisellä jyrsintapilla koneistetaan pyöristystä monimuotoisen kappaleen reunoihin. Silloin on kätevää kirjoittaa kappaleen ulkomuotoa vastaava rata lähestymis- ja poistumisliikkeineen tavalliseen aliohjelmaan. Itse pyöristykseen liittyvät laskutoimitukset tehdään makro-ohjelmassa ja tarvittavat terän kompensointiarvot asetetaan voimaan, minkä jälkeen ajetaan aliohjelmassa olevaa rataa. Laskutoimitukset ja kompensoinnit tehdään aina ennen uutta kierrosta radalla. Toistokertoja tehdään kunnes muoto on valmis.

8 ESIMERKIT

Reikäpiirimakro on yksi yleisimpiä makroja ja se voidaan toteuttaa monella eri tavalla. Esimerkissä on kuvattu eräs tapa tehdä reikäpiirimakro, jossa on pelkkä paikoitus. Makro voidaan kutsua esimerkiksi poraus- tai kierteitystyökierron perään, jolloin paikoitukset tehdään makron mukaisesti. Usein kuitenkin reikäpiirimakroon sisällytetään jo jokin työkierto, kuten poraus. Työkierto on helppo tehdä paikoitusta edeltävälle riville ja lisätä tarvittavat muuttujat lähtötietoihin.



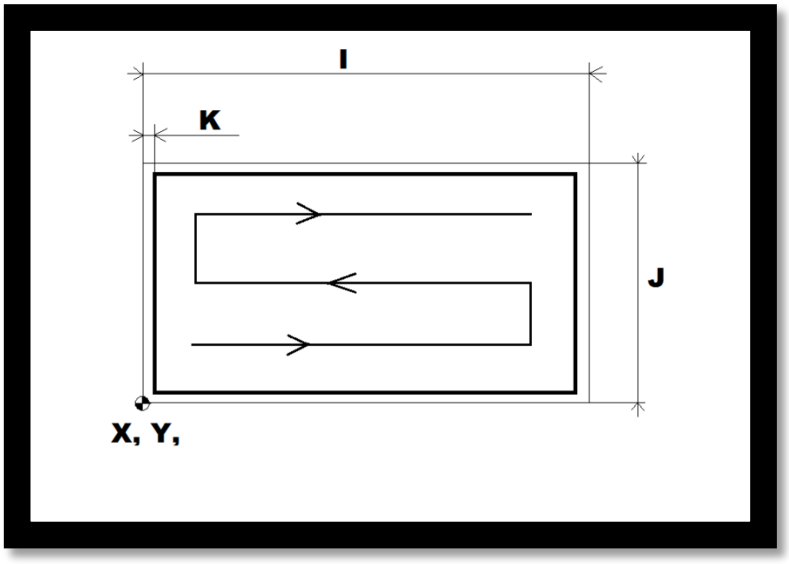
Kuva 2. Reikäpiirimakro, lähtöarvot.

Taulukko 8. Reikäpiirimakro.

	G65 P3000 R100 A0 K4	Pääohjelman makrokutsu A=aloituskulma K=reikien määrä
	O3000;(REIKÄPIIRI MAKRO)	Makro-ohjelman nro
	#32=1;	Asetetaan laskuri muuttujaan arvo 1

	WHILE [#32 LE ABS [#6]]D01;	Silmukka ehdolla
	#33=#1 + 360 * [#32 - 1] / #6;	Lasketaan pisteen kulma
	#101=#30 + #18 * COS[#33];	Lasketaan X-koordinaatti
	#102=#31 + #18 * SIN[#33];	Lasketaan Y-koordinaatti
	X #101 Y #102;	Paikoitetaan X ja Y
	#32=#32 + 1;	Lisätään laskuri muuttujaan 1
	END 1;	Silmukan loppu
	M99;	Poistutaan ohjelmasta

Poteron jyrsinnässä makrolla saadaan yleisimmin suurin ajansäästö perinteiseen tapaan verrattuna. Esimerkissä onkin kuvattu eräs tapa toteuttaa poteron jyrsintä makrolla. Lisäksi esimerkistä selviää sisäkkäisten silmukoiden käyttö. Myös muuttujan numeron laskenta on ratkaistu kätevästi tekemällä laskutoimitus hakasulkeisiin heti risuaitamerkin jälkeen kohdassa #27=#[2000+#7].



Kuva 3. Poteron jyrsintämakro, lähtöarvot.

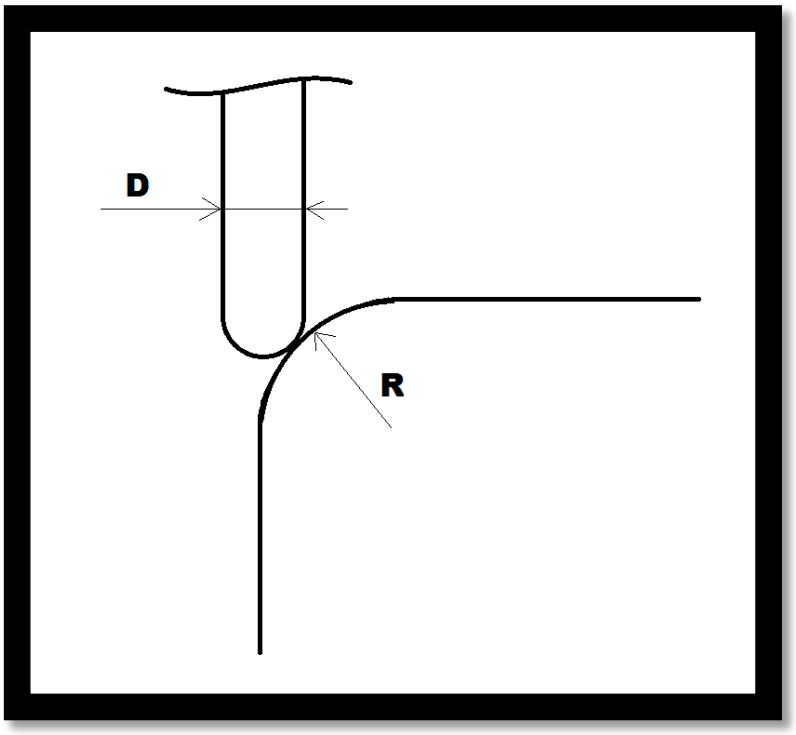
Taulukko 9. Poteron jyrsintämakro.

	G65 P2500 X0 Y0 Z0 R5 Q10 I200 J50 K0,1 D8 F150 E 100;	Pääohjelman makrokutsu Q=Lastunsyvyys (Z) R=Lähtötaso F=XY-tason syöttö E=syöttö Z-

		suunnassa
	O2500;(POTERON JYRSINTÄ MAKRO)	Makro-ohjelman nro
	#27=#[2000+#7];	Asetetaan muuttujan arvot yhtä suuriksi, toisen muuttujan numero määritetään
	#28=#6+#27;	Asetetaan muuttujan arvo laskutoimituksen mukaan
	#29=#5-2*#28;	Asetetaan muuttujan arvo laskutoimituksen mukaan
	#30=2*#27*#23/100;	Asetetaan muuttujan arvo laskutoimituksen mukaan
	#31=FUP[#29/#30];	Asetetaan muuttujan arvo laskutoimituksen mukaan, tulos pyöristetään ylöspäin
	#32=29/#31;	Asetetaan muuttujan arvo laskutoimituksen mukaan
	#10=#24+#28;	Lasketaan X-koordinaatti
	#11=#25+#28;	Lasketaan Y-koordinaatti
	#12=#24+#4-#28;	Lasketaan X-koordinaatti
	#13=#26+#6;	Lasketaan apumuuttuja X+K
	G0 X#10 Y#11;	Paikoitus pikaliikkeellä lähtöpisteeseen
	Z#18;	Paikoitus pikaliikkeellä lähtöpisteeseen
	#14=#18;	#14:n asetetaan turvaetäisyys
	DO1;	
	#14=#14-#17;	#14:n huomioidaan lastun syvyys
	IF[#14 GE #13] GOTO1;	Verrataan #14 suuruutta maksimi syvyyteen, ja tehdään päätös, ollaanko jo pohjalla.
	#14=#13;	Asetetaan #14 maksimi jyrsintäsyvyyden arvo
N1	G1 Z#14 F#8;	Ajetaan pohjalle
	X#12 F#9;	jyrsitään poteroa auki toiseen laitaan

		X-suunnassa
	#15=1;	Asetetaan #15 arvo yhteen, (0001)
	WHILE[#15 LE #31]DO2;	Silmukka
	Y[#11+#15*#32];	Ajetaan Y:tä askel ylemmäs
	IF[#15 AND 1 EQ 0] GOTO2;	Jos #15(1) and 1 on yhtä suuri kuin 0 hypätään riville N2
	X#10;	Ajetaan X-akselia toiseen laitaan
	GOTO3;	Hyppy riville N3
N2	X#12;	Ajetaan X-akselia toiseen laitaan
N3	#15=#15 + 1;	Laskuri
	END2;	Toisen silmukan loppu
	G0 Z#18;	Pikaliikkeellä Z R-tasoon
	X#10 Y#11;	Ajetaan X ja Y takaisin lähtöpaikalle
	IF[#14 LE #13] GOTO4;	Jos #14 pienempi tai yhtä suuri kuin #13, hypätään riville N4
	G1 Z[#14+1] F[8*#8];	Ajetaan Z suurella syötöllä(8x) edellisen rouhinnan tasolle.
	END1;	Silmukan loppu
N4	M99;	Poistutaan ohjelmasta pääohjelmaan

Alla oleva esimerkki kuvaa makro-ohjelmaa, jossa pallopäisellä jyrsimellä tehdään pyöristys kulmaan X-Y -tasossa olevan vapaasti määriteltävän muodon mukaisesti. Tässä esimerkissä merkillepantavaa on se, kuinka tavallista aliohjelmaa voidaan käyttää makro-ohjelman sisällä. Muuttujina ovat ainoastaan arvot R ja D.



Kuva 4. Kulmanpyöristysmakro pallopääterällä, lähtöarvot.

Taulukko 10. Kulmanpyöristysmakro pallopääterällä.

	G65 P2000 D10 R5	Pääohjelman makrokutsu
	O2000;(KULMANPYÖRISTYS-MAKRO PALLOPÄÄTERÄLLÄ)	Makro-ohjelman nro.
	#19=0;	Asetetaan muuttujaan arvo 0
	G91 G0 Z-#18;	Ajetaan Z-akselia - #18 verran
N10	G91 G0 Z#26;	Ajetaan Z-akselia #26 verran
	G90;	Asetetaan absoluuttiohjelmointi käyttöön
	#19=#19+#26;	Lasketaan #19 uusi arvo
	#21=[#7/2+#18]	Lasketaan #21 arvo
	IF[#19GE#21]GOTO100;	Jos #19 suurempi tai yhtä suuri kuin #7/2+#18, hypätään riville 100
	#20=SQRT[#21*#21-[#19*#19]];	Lasketaan #20 uusi arvo

	#13400=#20-#18;	Asetetaan työkalun kompensointiarvoksi muuttujien #20 ja #18 erotus
	M98 P3000;	Hyppäys ohjelmaan 3000
	GOTO10;	Hyppäys riville 10
N100	M99;	Poistuminen takaisin pääohjelmaan
	O3000;(OHJELMA MUODOLLE)	Ohjelma, johon muoto kirjoitetaan
	G90 G80 G54 G40;	Otetaan käyttöön absoluuttinen ohjelmointi. Poistetaan säde- ja pituuskompensoinnit sekä työkiertotiedot.
	G0 X10 Y0;	Paikoitetaan pikaliikkeellä X10 Y0
	G41 G1 F600 X0 Y0;	Otetaan kompensointi käyttöön
	G1 X… Y…;	Muodon ensimmäinen piste
	…	Muodon muut pisteet
	…	
	…	
	G1 X0 Y0;	Muodon viimeinen piste
	G1 G40 X10 Y0;	Poistetaan kompensointi käytöstä ja paikoitetaan
	M99;	Poistutaan aliohjelmasta

9 TIEDONHAKU KONEKOHTAISISTA MANUAALEISTA

Koneen makro-ohjelmointiin liittyvien ohjeiden ja tietojen etsiminen koneen manuaaleista ei aina ole helppoa, sillä koneiden mukana tulee useita manuaaleja, jotka voivat olla useita satoja sivuja pitkiä. Manuaaleja on yleensä kahta eri sarjaa. Koneenvalmistaja on laatinut manuaalit koneen käytöstä, ohjelmoinnista, asennuksesta, huollosta, varaosista ja sähkökuvista. Lisäksi saattaa olla vielä lisävarustekohtaisia manuaaleja. Maahantuoja tekee usein tärkeimmistä manuaaleista vielä käännökset sopivalle kielelle. Koneen ohjauksen valmistajan manuaaleja on usein omansa niin parametreille, ohjelmoinnille, käyttäjälle, kunnossapidolle kuin lisävarusteillekin.

Makro-ohjelmoinnissa tarvittavia ohjeita ja tietoja kannattaa lähteä etsimään koneen ohjauksen valmistajan manuaaleista. Ohjelmointiin tai käyttöön liittyvistä manuaaleista löytyy usein makro-ohjelmoinnille oma kappale, josta löytyy yleensä kaikki tarvittava tieto. Kyseistä kappaletta ei kannata lähteä etsimään manuaaleista selaamalla, vaan katsomalla sisällysluettelosta. Kappale on usein nimellä Custom macro ja mittausmakroihin liittyvät asiat löytyvät usein Measure macro nimisestä kappaleessa tai samannimisestä manuaalista. Fanuc-ohjauksissa makro-ohjelmointiin liittyvä kappale Custom macro löytyy manuaalista nimellä Operator`s manual. Mazatrol-ohjauksissa asiaa käsittelevä kappale on usein manuaalissa EAI/ISO programing manual.

10 TYÖSTÖARVOJEN ASETUS JA LASKENTA

ISO-koodi ohjelmoinnissa on totuttu antamaan työstöarvot aina erikseen työkalun ja työstökohdan mukaan valmiiksi laskettuina kierroksina ja pöytäsyöttöinä. Tämä aiheuttaa ylimääräistä työtä ja hidastaa ohjelmointia. Lisäksi työstöarvojen muuttaminen menetelmän vaihtuessa on työlästä. Keskustelevissa ohjauksissa työstöarvoja ei anneta erikseen joka kohtaan, vaan ohjelmalle annetaan tiedot halutusta leikkuunopeudesta ja hammaskohtaisesta syötöstä, joista ohjaus laskee todelliset kierrosnopeudet ja pöytäsyötöt.

Makro-ohjelmoinnin avulla olisi mahdollista helpottaa ISO-koodi-ohjelmoinnissa työstöarvojen määritystä. Mahdollisuuksia on paljon, vain mielikuvitus on rajana. Isoimmat hyödyt on saavutettavissa jyrsinnässä, koska muuttuvia tekijöitä on eniten.

Työstöarvojen laskenta voidaan sisällyttää toisen makron sisälle. Esimerkiksi tasopintojen plaanausta varten tehtyyn makroon voitaisiin sisällyttää tasojyrsinnässä käytettävien työstöarvojen laskenta. Annettavia lähtötietoja voivat olla leikkuunopeus, hammaskohtainen syöttö ja hampaiden määrä.

Jyrsimen pyörimisnopeuden laskemiseen tarvitaan tiedot halutusta leikkuunopeudesta ja jyrsimen halkaisijasta. Leikkuunopeus voidaan antaa muuttujan avulla makrokutsun yhteydessä. Halkaisijatieto saadaan lukemalla koneen offset taulukosta kyseisen työkalun D-arvo ja kertomalla se kahdella, koska arvo vastaa työkalun sädettä. Arvon muuttujanumero löytyy koneen manuaaleista. On muistettava, että nämä muuttujat ovat konekohtaisia ja ne on tarkistettava aina käytettäessä samaa ohjelmaa eri koneilla. Esimerkistä selviää, miten laskenta suoritetaan.

Pöytäsyöttö voidaan laskea kertomalla pyörimisnopeus hampaiden lukumäärällä ja hammaskohtaisella syötöllä. Lähtötiedot voidaan antaa makrokutsussa.

Taulukko 11. Esimerkki työstöarvojen laskennasta makrolla.

Pääohjelma	
…	
G65 P5000 V150 F0.1 E5;	Makrokutsu jossa V=Leikkuunopeus=#22 F=Hammaskohtainen syöttö=#9 E=Hampaiden määrä=#8

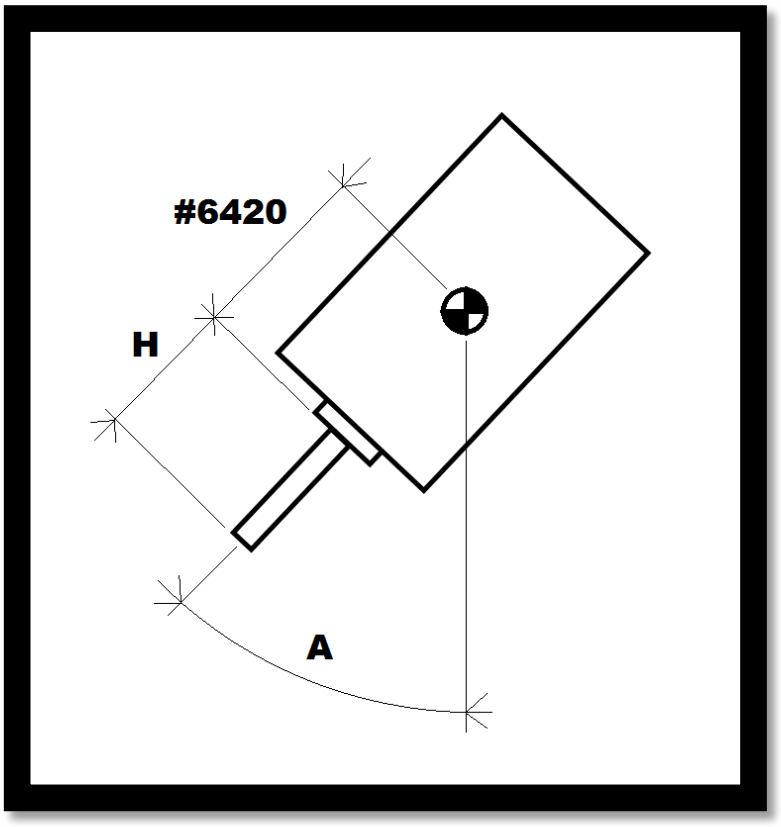
…	
Makro-ohjelma	
O5000;	
#19=#22/[#13001*3,14];	Lasketaan pyörimisnopeus muuttujaan #19. Terän halkaisija otetaan offset-taulukon muuttujasta.
#1=#19*#9*#8;	Lasketaan pöytäsyöttö muuttujaan #1
M3 S#19;	Asetetaan pyörimisnopeus muuttujalla #19
G94 F#1;	Asetetaan syöttö mm/min muuttujalla #1
M99;	Poistutaan ohjelmasta

11 TYÖKALUPISTEEN MUUTOSTEN LASKENTA

Nykyaikaisissa moniakselisissa työstökoneissa työkalupisteen muutosten laskenta on koneissa sisäänrakennettuna ja käyttäjän ei tarvitse miettiä muutoksia. Vanhemmissa tai alkeellisimmissa koneissa, joissa on manuaalisesti käännettävä kara, pöytä tai jokin muu lisälaite, ei yleensä ole valmista toimintoa työkalupisteen muutoksien laskennalle. Koordinaatiston siirrolle ja käännölle löytyy toiminnot lähes kaikista koneista. Ongelma vain on, paljonko koordinaatistoa pitää siirtää. Työstökeskuksissa, joissa karapäätä saadaan käännettyä manuaalisesti eri asentoihin, törmätään ongelmaan, jossa työkalupiste muuttuu. Tällöin alkuperäinen kappaleen nollapiste muuttuu. Samantapaiseen ongelmaan törmätään manuaalisesti kääntyvän pöydän kanssa tai NC-jakopäätä käytettäessä. Uusien työkalupisteiden laskenta käsin on usein työlästä ja aiheuttaa ylimääräistä työtä. Kaksiakselinen kääntö aiheuttaa laskentaa 3D-ympäristössä, jolloin pelkkä virheen mahdollisuuskin on jo niin suuri käsin laskettaessa, että erillinen laskentaohjelma tulee kysymykseen.

Makrosta saatavat arvot voidaan sijoittaa sift-taulukkoon. Paras ja turvallisin tapa on kuitenkin tehdä makrossa nollapistesiirto saatujen arvojen mukaan.

Seuraavassa esimerkissä on selvitetty uuden työkalupisteen laskentaa kääntyvä karaisessa koneessa. Laskentaa varten pitää tarkasti tietää mitta koneen karanpäästä kääntöakselin keskelle. Tämä voidaan mitata koneesta erikseen makroa tehtäessä. On tärkeää merkitä muistiin, missä kohtaa ohjelmaa mittaa käytetään. Jos mitta muuttuu vaikkapa ajettaessa koneella kolari, pitää muutos korjata ohjelmaan.



Kuva 5. Mitat makrokutsuun.

Taulukko 12. Esimerkki makro-ohjelma työkalupisteen muutosten laskentaan.

Pääohjelma	
…	
G65 P5100 A15 H150;	Makrokutsu A=kulma #1, Z-akseliin nähden (kääntö tapahtuu X-Z-tasossa) H=työkalun pituus #11 (voidaan myös lukea muuttujasta, kun se on tiedossa)
…	
Makro-ohjelma	
O5100;	
#24=[#11+#6420]*SIN[#1];	Lasketaan X-akselin siirtymä, #6420 on karan geometrinen mitta karanotsapinnasta kääntöakselin keskelle.
#26=[#11+#6420]*COS[#1];	Lasketaan Z-akselin siirtymä

G92 X#26 Z#24;	Nollapisteen siirto
M99;	Poistuminen ohjelmasta

12 LOPUKSI

Työssä käsiteltiin erilaiset muuttujat ja niiden käyttökohteet. Muuttujien käsittelyyn tarvittavat operaatiot ja ohjelmalliset toiminnot esiteltiin. Lisäksi kerrottiin makro-ohjelman ja aliohjelman kutsumistavat. Tiedonhakuun ja ongelmatilanteisiin annettiin neuvoja. Koneen toimintaan ja käyttöön liittyviä esimerkkejä ja sovelluskohteita käsiteltiin myös.

Perinteisen ohjelmoinnin hallitsevalle henkilölle nämä tiedot antavatkin hyvän ja monipuolisen lähtökohdan omatoimiselle makro-ohjelmoinnille. Makro-ohjelmien luonnissa onkin usein rajana vain mielikuvitus. Työllä on ohjattu lukijaa käyttämään makroja myös muualla kuin työstöohjelmissa. CAM-ohjelmistojen yleistymisen myötä makrojen käyttö vähenee työstöohjelmia tehtäessä, tällöin pitäisikin keskittyä makrojen käytössä enemmän koneen toimintoihin.

Tämän opinnäytetyön tekeminen kehitti omaa osaamistani ohjelmoinnissa. Makrojen teosta tuli rutiininomaista, jolloin niitä on huomattavasti helpompaa ja sujuvampaa käyttää. Opinnäytetyötä tehdessä löysin koneiden ohjauksista useita uusia ominaisuuksia ja mahdollisuuksia toteuttaa ohjelmointia. Sain myös innostutettua muutaman Vexve Oy:n koneistajan makro-ohjelmoinnin pariin, kun testasin makroja Vexve Oy:n koneilla. Nämä kokemukset vahvistivat edelleen näkemykseni siitä, että makro-ohjelmointi on vielä nykypäivänäkin erittäin kilpailukykyinen tapa ohjelmoida työstökonetta.

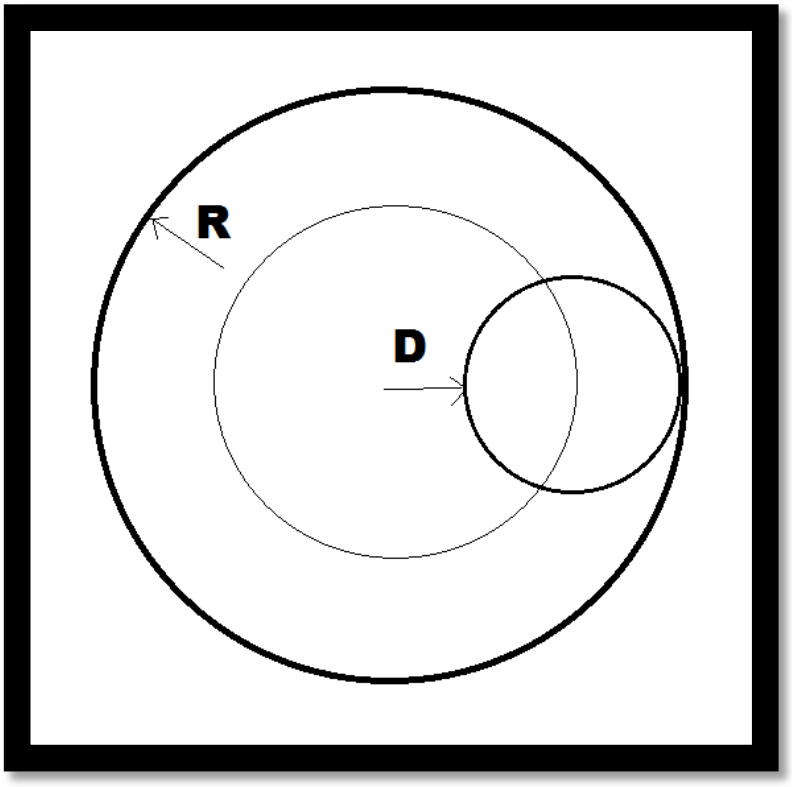
LÄHTEET

Fanuc. 2003. FANUC Series 0i-Mb Operator`s manual, 2. painos B-63844EN/02

Fanuc 2002. Fanuc Series 16i-TB Fanuc Series 18i-TB Fanuc Series 160i-TB Fanuc Series 180i-TB Operator`s manual, 1. painos B-63524EN/01

Mazak 2012. Programming manual Mazatrol Matrix 2 EIA/ISO Program

Reikäinterpolaatio-makro



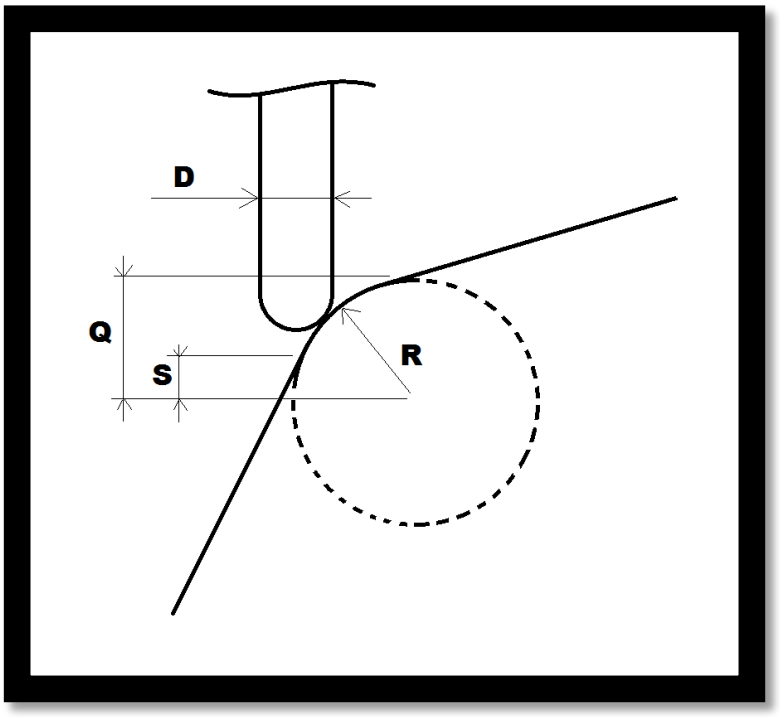
R=Reijän säde
D=Työkalun halkaisija
Z=Syöttö/per kierros
Q=Kierroksien määrä

Terä paikoitetaan ennen makrokutsua reiän keskelle.

	O1000;(REIKA INTERPOLAATIO MAKRO)
	#2=0;
	#1=[#18-[#7/2]];
	G91;
	G0 X#1;
N10	G90;
	G3 I[-1*#1] R#1 K#26;
	IF[#2GE#17]GOTO100;
	#2=#2+1
	GOTO10;
N100	G3 I[-1*#1] R#1;
	G91;
	G0 Z[#26*#17];

	G90;
	M99;

Kulmanpyöristys-makro vapaalle muodolle, aloitus- ja lopetuskorkeuden asetuksella, pallopääterällä



Q=#17=Lopetuskorkeus
S=#19=Aloituskorkeus
R=#18=Pyöristyksen säde
D=#7=Työkalun halkaisija
Z=#26=Lastun paksuus Z-suunnassa

Muuttujat #20 ja #21 ovat apumuuttujia. Muuttuja #13400 on työkalun D-arvo paikassa 400(muuttuja vaihtelee ohjauksittain).

Varsinainen muoto johon pyöristys halutaan, kirjoitetaan erilliseen aliohjelmaan.

	O2000(KULMANPYORISTYS MAKRO);
	G91 G0 Z[#19-#18];
N10	G91 G0 Z#26;
	G90;
	#19=#19+#26;
	IF[#19GE#7/2+#18-#17]GOTO100;
	#21=[#7/2+#18]
	#20=SQRT[#21*#21-[#19*#19]];
	#13400=#20-#18;

	M98 P3000;
	GOTO10;
N100	M99;

	O3000;(ALIOHJELMA MUODOLLE)
	G90 G80 G54 G40;
	G0 X(LÄHESTYMISPISTE) Y(LÄHESTYMISPISTE);
	G41 G1 F600 X(MUODON ENSIMMÄINENPISTE) Y(MUODON ENSIMMÄINENPISTE);
	G1 X… Y…;
	…
	…
	…
	G1 X(MUODON VIIMEINENPISTE) Y(MUODON VIIMEINENPISTE);
	G1 G40 X(LÄHESTYMISPISTE) Y(LÄHESTYMISPISTE);
	M99;