UNIVERSIDAD PANAMERICANA

ESCUELA DE INGENIERIA

CON ESTUDIOS INCORPORADOS A LA U.N.A.M.

GENERANDO PROGRAMACION AVANZADA CNC EN TU PC

(MACRO PROGRAMACION)

TESIS

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA

(AREA ELECTROMECANICA)

P R E S E N T A :

GUILLERMO ROSADO BOSQUE GOMEZ

ASESOR: DR. STANISLAW RACZYNSKI GAWIN

MEXICO, D. F. 1999

A mis maestros:

Tomas Morán

Gustavo Alarcon

Contenido

Prólogo

El conocimiento no puede ocultarse

Hace algún tiempo tal vez alguna persona , o yo mismo no habría imaginado que intentaría escribir un texto donde expusiera temas avanzados de programación en Control Numérico Computarizado (CNC). El curso que tomé de CNC como parte de mi formación profesional, no me interesó en demasía, ni me entusiasmó tanto como otras materias.

Egresando de la carrera, tuve la oportunidad de entrar a trabajar en una empresa cuyo giro era precisamente la venta y reparación de máquinas con esa tecnología. Al principio estaba preocupado, mis conocimientos de programación no eran suficientes. Sin embargo, para mi grata sorpresa, me dí cuenta que todas las máquinas de la empresa se manejaban en torno a un control conversacional sumamente amigable que pronto aprendí. Un control conversacional toma su nombre del hecho que interactúa con el usuario en base a imágenes y formas, no con código, por lo que se facilita enormemente su programación.

De esa época recuerdo dos hechos en forma particular que fueron las razones fundamentales para que me adentrara en el estudio del CNC y este texto pudiera salir a la luz:

En una exposición de maquinaria en la cual participé, los encargados del taller de maquinado de una empresa dedicada a fabricar maquinaria pesada para la industria de la construcción se acercaron a platicar conmigo. Tenían un problema de maquinado y pidieron mi opinión al respecto. En cierta pieza debían perforar una serie de barrenos entre dos placas paralelas soldadas entre sí. Cada placa tenía diferente espesor, con una distancia de separación entre ellas de aproximadamente 1". Las placas estaban soportadas por una pieza bastante pesada que hacía que la mejor opción para el maquinado fuera la de perforar el barreno desde la placa superior; acabando de perforar la primera placa, avanzar con movimiento rápido hasta la superficie de la segunda ; programar otra vez avance y terminar de barrenar la segunda placa, para posteriormente, retirar la herramienta con movimiento rápido y porseguir hasta finalizar todos los barrenos. Así estaban ellos mecanizando la pieza, sin embargo, todas sus máquinas tenían control conversacional y estaban programando dicha operación en EIA –ISO *. Querían saber si de algún modo podían usar algo semejante al control conversacional , algo que sólo les permitiera rellenar las variables adecuadas y que ese "algo" generara código de maquinado, pero no solamente código para las medidas de esas placas en particular; sino que también cubriera un gran rango de piezas posibles.

Ni ellos ni yo lo sabíamos, pero lo que ellos querían era un macroprograma.

---

* EIA – ISO : Denominación del conjunto de códigos estándar utilizados para programación de maquinaria CNC por la Asociación Internacional de Fabricantes de Máquinas Herramientas

No pude ayudarles por falta de mayor conocimiento; busqué una excusa y me desembaracé del problema. Hasta la fecha recuerdo el incidente, sobre todo porque ahora que conozco un poco más , sé que la solución era sencillísima, por no decir risible. El hecho de no tener el conocimiento suficiente, en el instante correcto, es el primer motivo de este libro.

El segundo hecho se da en un encuentro que tuve en un taller donde instalé una máquina nueva que implicaba impartir un curso sobre su operación. Esa máquina tenía control EIA –ISO, no conversacional, control semejante al resto de las máquinas que operaban ahí. Al platicar con el dueño del taller sobre la temática del curso que siempre se da cuando se instala una máquina, me indicó que lo único que podía enseñarles a su gente era a programar con macro, pues dominaban todo lo demás. Comencé a buscar libros sobre el tema en la biblioteca de manuales que estaban a nuestra disposición en el trabajo. Más me di cuenta de que la información era escasa y no fácil de dominar. Continué buscando información en otros lugares con resultados parecidos. Con el material que pude reunir me propuse aprender a programar con macro.

El aprendizaje no fue sencillo. En parte porque no tenía la suficiente información ni personas relativamente cercanas para disipar las dudas que surgían conforme avanzaba en el estudio. Muchas veces desee tener un libro técnico de macro programación que abordara situaciones reales, con temas que pudiera tomar directamente del papel y aplicar a la máquina, que me orientara sobre cómo maquinar, cómo desarrollar mis macros de manera óptima , que me brindara ejemplos, etc... El poder tener disponible un texto que hable exclusivamente de macro programación es el segundo motivo de este libro.

El propósito de este texto es ser un instrumento de apoyo para personas relacionadas con el campo del CNC que requieran entender los fundamentos de la programación avanzada vía macro, plantear y desarrollar sus macros y en general elevar su nivel de programación.

No es un texto básico. Requiere de ciertos antecedentes de conocimientos para que puede ser plenamente comprendido y aprovechado. Difícilmente se asimilará si no hay experiencia previa de programación en CNC , bases de programación en P.C y sólidos conocimientos matemáticos. Es por ello que considero está dirigido a estudiantes de ingeniería que están recibiendo un curso avanzado de manufactura, profesores de la materia , programadores de CNC que deseen aprender mejores técnicas y técnicos especializados en máquinas herramientas que busquen elevar su nivel .

Este primer volumen está enfocado totalmente a programación avanzada para centro de maquinado, el segundo habla exclusivamente de torno. Esta división me pareció adecuada para facilitar la exposición de los temas, además aquellas personas que estén interesadas en uno u otro tipo de maquinaria podrán seleccionar el texto más adecuado a sus intereses.

Esperando que los conceptos manejados aquí te sean de utilidad :

Guillermo Bosque

CAPITULO 1

Conceptos de macroprogramación.

¿ Que es macro programación ?

La macroprogramación consiste básicamente en definir matemáticamente por medio de variables adecuadas la trayectoria de una herramienta a través de una forma geométrica, no importando lo compleja que ésta pueda llegar a ser - de hecho, prácticamente cualquier forma geométrica puede ser modelada matemáticamente - posteriormente asignar valores a dichas variables y generar código EIA- ISO.

Puede decirse que la macro programación, es el secreto mejor guardado referente a la programación CNC. Secreto, en el sentido de que a pese a ser una herramienta poderosa , es poco explotada.
De las personas relacionadas con el control numérico con quienes he tenido contacto (programadores, operadores, personal de mantenimiento, instructores, etc..), puedo contar a aquéllas que realmente tienen un concepto claro de lo que implica programar usando macros, esto es, conocedoras del potencial de las aplicaciones que pueden generarse vía macro programación, desarrollarlas en macro programas y utilizarlas en forma eficaz. La mayoría desconoce de hecho que exista esta herramienta.

Las razones de esto son varias. En primer lugar creo que la macro programación no es una herramienta cuyo conocimiento esté suficientemente difundido . No es fácil encontrar en nuestro medio información sobre el tema debido a que la intención original del fabricante no era desarrollar un sistema de programación avanzada, sino más bien contar solamente con un medio de comunicación directo con el control de la máquina que permitiera instalar y manejar dispositivos y accesorios como probetas de medición, alimentadores de barra, carrusel de herramientas , brazos robot, etc..
Otra razón importante para su poco uso, es que el programar con macro no es fácil ; se requieren conocimientos básicos de programación en P.C, bases matemáticas muy sólidas y cierta experiencia de maquinado; factores no siempre fáciles de conjuntar entre personas cercanas a la máquina CNC.

Puede mencionarse además , que generalmente es una opción no incluida en el controlador de la máquina, por lo que debe pagarse una cantidad extra para adquirirla . Y por lo regular no adquirimos algo a menos que sepamos claramente los beneficios que puede reportar . Obviamente, si desconocemos la capacidad de la herramienta, desconocemos sus beneficios, es más difícil que se adquiera y la tengamos disponible en nuestro control.

La mayor parte de las veces la opción de programación en macro está oculta en tu control y sólo mediante la introducción de un código especial, - conocido por el fabricante – puedes tener acceso a ella. Otras veces la opción está ahí como estándar , pero es posible que nunca te hayas dado cuenta de ello.

Una forma de saber si uno tiene disponible la opción de macro en su control es revisar el manual de programación de la máquina; otra es intuitivamente. Por ejemplo, si la máquina posee dispositivos especiales tales como una probeta de medición, o un alimentador de barra, con seguridad la opción viene incluida, pues es necesaria para controlar dichos dispositivos ; sin embargo, la mejor manera es preguntar directamente al distribuidor o fabricante de nuestro control. Pero recuerda al preguntar, que la programación con macro recibe muchos nombres, según el tipo de controlador de la máquina. Algunos la llaman macro versión A, otros macro versión B, custom macro, programación paramétrica, etc..

A lo largo de este texto la llamaré con el nombre que creo representa a cabalidad el concepto que maneja macro programación.

Este capítulo tratará de brindar un panorama general del qué y del cómo, para poder comprender mejor los alcances de esta herramienta. Más la razón de escribir este texto, no es explicar a profundidad la macroprogramación directamente en tu control CNC . Muchas veces a pesar de que puedas tener el conocimiento de la técnica de programación o llegar a adquirirla, si tu control no está equipado con esta opción, no será posible aplicarla. Lo que buscamos es que el concepto de macro quede plenamente entendido tanto en su idea, como en la forma de generarla, para que pueda ser programada y utilizada en una herramienta estándar como lo es una P.C, a partir de ésta, generar código EIA - ISO y posteriormente trasladarlo a tu controlador.

Aplicaciones de la macro programación.

Las aplicaciones de una herramienta están en razón proporcional a su potencial y la macro programación no es la excepción, sus aplicaciones son prácticamente infinitas, tantas como grande sea la imaginación. Imagina el poder generar código de maquinado para familias de partes con un solo programa, fresar cajeras, hacer ranuras, diseñar ciclos de taladrado, de careado, de ranurado , maquinar levas, realizar ciclos de mandrinado , desarrollar código para

maquinado de formas complejas, incluso poder manejar dispositivos de tu máquina. Estas son solamente algunas de las aplicaciones que puedes generar con esta herramienta.

Cabe hacer un pequeño paréntesis para responder a una pregunta, que creo ya te habrá surgido ¿ No es mejor usar un CAD - CAM para generar código de maquinado ? , la respuesta es ambivalente: si en tu taller se están maquinando formas altamente complejas como moldes de botellas, troqueles de superficie irregular o cualquier clase de forma geométrica complicada en 3D, la respuesta es obvia , naturalmente que sí , el CAD -CAM, es una herramienta indispensable en este caso. Por el contrario, si haces maquinados repetitivos en un centro de maquinado o posees tornos de dos ejes, es conveniente sumergirte en lo que es macro programación . Recuerda que un CAD -CAM no es una opción integrada en tu control y que muchas veces no es accesible ni redituable adquirirlo .En cambio, desarrollar una macro sólo te costará unas hojas de papel, medio lápiz y unas horas de trabajo.

Nuestro principal interés es procurar desarrollar tu habilidad para diseñar macros que generen patrones estándar de mecanizado, que se adapten a cualquier tipo de control con técnicas avanzadas de programación CNC que permitan un maquinado más eficiente y seguro.

Usando macro programación en tu control CNC.

De manera general, podemos decir que la macro programación puede dividirse en dos categorías básicas: la relativa a cómo se programa, y la relativa al propio controlador. La primera de ellas es prácticamente común y es consistente de fabricante a fabricante; por lo tanto, lo que se hable aquí de la primera categoría casi puedes asegurar se aplica a tu tipo de control (siempre y cuando tengas liberada la opción de macro programación). Las opciones relativas al controlador varían de acuerdo al tipo de control usado en tu máquina ; hablaremos de ellas brevemente ya que por su variedad no podemos abarcar todas.

Modo de programación.

Para llamar modo macro usamos el comando G65 (en EIA –ISO), al cual sigue el número de macro programa que se va a utilizar y finalmente una serie de variables representadas por letras, las cuales representan las variables que se van a ingresar a la macro y sobre las que se ha programado.

Una típica llamada a macro puede ser como sigue:

G65 P1000 A 0.25 B 8. 0 F 0.006 Z 1.0

Más adelante, si tú ya has programado en cualquier lenguaje de computadora, notarás lo fácil que es seguir la lógica de una macro; de hecho la programación estándar de una macro en tu

controlador recuerda mucho al lenguaje C o BASIC, pero con ciertas variaciones relativas a la lógica del controlador como veremos.

Técnicas con variables.

Las variables son lugares que pueden almacenar valores factibles de cambiar a lo largo del programa. Piensa en variables como en los *offsets* de las herramientas. Podemos declarar un *offset* en nuestro control y dependiendo de la declaración de código que hagamos (G43, G41 o G42), la máquina interpretará dicho *offset* como compensación de longitud o compensación de radio izquierda o derecha. El control no sabe cómo se aplicará el *offset* almacenado hasta que corramos el programa y lo asocie con el código correcto.

La analogía entre *offsets* y variables resulta completa. Los *offsets*, como las variables, son lugares donde se almacena un valor, valor que no se conoce hasta que se referencia de algún modo durante la ejecución del programa .El uso de variables es más flexible que el de los *offsets*. Un *offset* implica un tipo de compensación para la herramienta, mientras que una variable puede representar casi todo.

Podemos distinguir 4 tipos de variables en un controlador CNC:

Variables que se usan en el llamado a macro.

Como vimos al ejecutar un llamado a macro, éste viene acompañado de letras a las cuales siguen valores numéricos seguidos de un punto decimal. Estas letras representan las variables que pasan a la macro números que indican como ésta debe comportarse. Por ejemplo en el siguiente código que llama a macro

G65 P1000 A 0.25 B 8. 0 F 0.006 Z 1.0

si la letra Z referenciada en la llamada a macro anterior representa para nosotros la altura inicial a la cual queremos que llegue la pieza antes de comenzar el maquinado, le estamos informando a la macro que debe llegar a una altura de 1.0" ; si por el contrario le indicamos Z 1.25 , la altura inicial será de 1.25".

Las letras que se usan para pasar valores a la macro las llamaremos variables de dirección. No todas las letras del abecedario pueden usarse para pasar variables, porque pueden generar conflicto entre la interpretación de las ordenes y el código EIA – ISO normal; estas letras son: N, O, G, P y L.

La lista de letras que es permitida como variables de dirección es:

A, B , C , D , E , F , H , I , J , K , M , Q , R , S , T , U , V , W , X , Y , y Z .

El programador debe tener una clara idea de todas las variables que necesita para desarrollar su macro, las que representan el trazo matemático de la herramienta, además de las variables que afectan a la geometría de la pieza y aquéllas que afectan las condiciones del mecanizado; para ello siempre es conveniente dibujar la geometría a mecanizar así como el recorrido generado por la herramienta en el mecanizado, (al cual llamaremos de ahora en adelante, "*tool path*"), representando en un dibujo, dichas variables.

Variables Locales .

Una vez que pasamos a la macro los argumentos en forma de letras, éstas no pueden referenciarse dentro de la macro con las mismas letras, debido a que el control puede confundirse con las usadas para programar diferentes comandos (X,Y, Z para posicionar los ejes, F para el avance , H para compensación de altura, etc..).

Por esta razón las letras que representan variables se representan dentro de la macro como variables de tipo local; dichas variables varían en un rango de #1 a # 26 y se relacionan como se muestra en la siguiente tabla:

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

Aunque es fácil de ver que estas variables de tipo local no siguen un orden lógico, esta representación es la que deben seguir dentro de un macro programa.

Consideremos por ejemplo, la llamada a macro dada anteriormente:

G65 P1000 A 0.25 B 8.0 F 0.006 Z 1.0

Dentro del programa O 1000 (mi macro programa), no puedo usar las letras A, B, F o Z para representar valores, debo buscar en la tabla dada anteriormente sus equivalentes, así , la letra A estará representada en la macro por # 1, B por # 2, F por # 9 y Z por # 26. Nota que para referirnos a las variables del control, tengo que indicarlas por medio del signo # , más el número que las referencia .

Aunque tal vez esto no tenga mucho sentido todavía, cuando veamos un ejemplo, las cosas empezarán a verse con mayor claridad.

Cabe hacer mención aquí que las variables de tipo local son variables de tipo temporal, es decir, el control solamente las recuerda en el instante en que el macro programa es ejecutado; al finalizar éste (con el código M99) , todas las variables de tipo local quedan olvidadas ; esto es, quedan vacantes y sin valor.

Variables comunes.

Las variables comunes sirven a 2 propósitos básicos: uno es proporcionar un lugar para alojar los resultados de las operaciones matemáticas inherentes al macro programa; las variables representadas de este modo, pueden utilizarse para referenciar algo dentro de la macro o pueden usarse para representar prácticamente cualquier cosa. Estas variables, también nos sirven para referenciar un valor asignado a una variable local, así como el valor de cualquier comando que *exista* en nuestro control. Por ejemplo si nosotros asignamos a la variable común # 120 el valor de 0.25 y si el control lee la siguiente instrucción:

G00 X # 120

Es lo mismo que el control leyera :

G00 X 0.25

El número de variables comunes del que podemos echar mano al programar varía de control a control, generalmente son 50, del rango # 100 a # 149. Es posible aumentar este rango , mediante un pago adicional al fabricante.

Las variables de tipo común son un poco más permanentes que las variables locales. En la mayoría de los controles el valor asignado a ellas es recordado hasta que la máquina es apagada; posteriormente vuelven a quedar vacantes, es decir, sin valor.

Variables comunes de tipo permanente:

Hay un cierto tipo de variable común que el control recuerda aún después de haber apagado la máquina; a excepción de esto, este tipo de variable es prácticamente igual a las variables nombradas anteriormente.

Existen en un control normal al menos 10 variables de tipo permanente cuyo rango varía del # 500 al #509.

Este tipo de variables se usan generalmente , cuando por ejemplo, queremos usarlas para llevar un registro de la vida útil de nuestra herramienta, alojando en esta variable la cantidad de tiempo utilizado por la misma y poder permitir que el control tome la decisión de ejecutar un comando automático de cambio de herramienta.

Variables del sistema.

Las variables del sistema nos permiten acceder a muchas funciones del control de CNC, pudiéndolas incluir en nuestro macro programa. Entre éstas podemos citar: *offsets* de herramientas, coordenadas de posición actual , modo del control (incremental o absoluto , avance por revolución o por minuto, modo en pulgadas o en mm, etc..).

El rango reservado en el controlador para estas variables, generalmente va del # 1000 al # 7000, aunque como ya se dijo, estos números pueden variar, dependiendo del fabricante.

Operaciones aritméticas.

Una de las características que hacen más poderosa la programación con macros, es la posibilidad de usar funciones matemáticas y trigonométricas . Para guardar los resultados de estas operaciones, por lo general usamos las variables de tipo común (variables en el rango #100 a #149). La mayoría de las operaciones matemáticas, te serán familiares: la igualdad es representada por el signo de igualdad (=) , la adición por el signo más (+), la sustracción por el signo menos (-) , la multiplicación por un asterisco (*) y la división por una barra diagonal (/).
A continuación damos una lista de ejemplos de operaciones comunes:

#101 = 3 (A la variable 101 se le asigna el valor de 3)
#101 = 3 +1 (A la variable 101 se le asigna el valor de 4)
#101 = 3 - 1 (A la variable 101 se le asigna el valor de 2)
#101 = 3 *3 (A la variable 101 se le asigna el valor de 9)
#101 = 3 / 2 (A la variable 101 se le asigna el valor de 1.5)

Es posible combinar operaciones, pero como en cualquier lenguaje de programación, hay un orden de precedencia que debemos respetar ; la multiplicación posee el valor más alto, después viene la división, la adición y al final la sustracción. Por ejemplo, los dos comandos siguientes asignan un valor diferente a la variable #105.

#105 = 3 +5 *2 (#105 es igual a 13)
#105= (3 + 5) *2 (#105 es igual a 16)

Con el ejemplo anterior te darás cuenta que la multiplicación tiene una precedencia mayor que la suma, pero si indicamos con paréntesis la precedencia de la adición, el controlador realiza esta primero y posteriormente la multiplicación.
Al agrupar nuestras operaciones por medio de paréntesis forzamos el orden de las operaciones, así, siempre que tengamos duda de cómo el control interpreta una operación aritmética, podemos usar paréntesis para asegurarnos de que ésta sea en la forma deseada.

Es posible el referenciar a una variable otra variable previamente calculada ; así podemos hacer lo siguiente:
#105 = 3

106 = 4

#108 = #105 + # 106

En este caso a la variable # 108, le estamos asignando la suma del contenido de las variables #105 y #106, lo cual sería lo mismo que:

#108 =3 + 4

También es válido:

#105 = 1

#105 =#105 + 1

En este ejemplo, a la variable # 105 le asignamos el valor de 1; posteriormente este valor lo incrementamos en uno y tendrá el valor de 2 y así sucesivamente. Esta técnica muestra cómo una variable puede usarse como contador.

Aplicando aritmética avanzada.

Si sólo dispusiéramos de las cuatro operaciones básicas para macro programar; resultaría muy difícil pensar en realizar cosas complicadas ; afortunadamente el controlador nos proporciona operaciones matemáticas más avanzadas que se traducen en una programación más poderosa.

He aquí una lista de las funciones matemáticas avanzadas más comunes que puedes encontrar en tu controlador :

Seno:

#101 = SIN [30] - se le asigna a la variable # 101 el valor del seno del 30°

Coseno:

#101 = COS [30] - se le asigna a la variable # 101 el valor del coseno de 30°

Tangente:

#101 = TAN [30] - se le asigna a la variable # 101 el valor de la tangente de 30°

Arco Tangente:

#101 = ATAN [1] - se le asigna a la variable # 101 el valor del ángulo cuya tangente es 1 .

(Algunos controladores poseen 2 funciones arcotangentes, la diferencia entre ellas es únicamente el rango de ángulo que acepta).

Raíz Cuadrada:

#101 = SQRT[100] - se le asigna a la variable # 101 el valor de la raíz cuadrada de 100

Valor absoluto:

#101 = ABS [-3] - se le asigna a la variable # 101 el valor del valor absoluto de -3

Redondeo

#101 = ROUND[8.9] - se le asigna a la variable # 101 el valor del número entero más cercano

Redondeo bajo:

#101 = FIX [3.6] - se le asigna a la variable # 101 el valor del número entero más bajo en el intervalo del punto decimal, en este caso 3 .

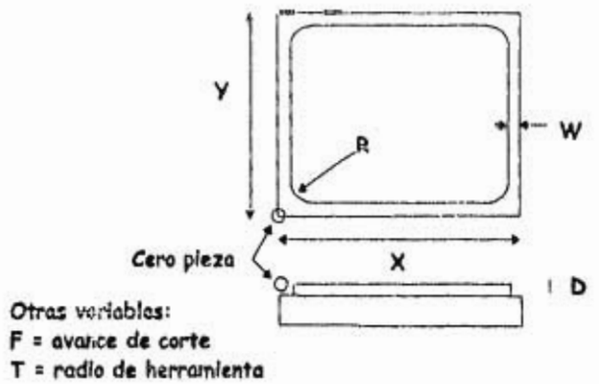
Redondeo arriba

#101 = FUP[3.6] - se le asigna a la variable # 101 el valor del entero más alto en el intervalo del punto decimal, en este caso 4.

Dependiendo de la versión del controlador, encontrarás operaciones matemáticas más complejas, incluyendo operaciones lógicas , por lo que será recomendable que hojees tu manual. Recuerda que puedes asociar condiciones lógicas mediante el uso de paréntesis y usar los mismos para forzar el orden de una operación..

Mostremos un ejemplo de macro programación.

Hasta ahora hemos visto las herramientas que tenemos a nuestra disposición para macro programar , sin embargo es tiempo ya de mostrar un ejemplo, que aunque sencillo, utilice los conceptos explicados anteriormente.



La figura anterior representa una familia de partes elaboradas en el taller, la cual debe maquinarse en su contorno exterior .

El maquinado de esta pieza puede variar, si por ejemplo, el material de trabajo es más corto o más largo, si el radio del contorno varía, así como la profundidad ; además , no todas las piezas son del mismo material. Por lo tanto te darás cuenta que en esta familia de partes, lo relacionado con su maquinado es variable.

En este caso, tú como programador, podrías ahorrarte bastante trabajo si tuvieras un solo programa, en el cual ingresaras las variables adecuadas necesarias para generar el código de maquinado en forma automática. Mejor aún , si decides por alguna razón durante el maquinado, por ejemplo, aumentar o disminuir el número de cortes radiales , no tendrás que volver a reescribir código, simplemente cambias la variable adecuada en ese programa y listo. Ese único programa es una macro.

Antes de desarrollar nuestra macro, es conveniente dibujar e identificar en nuestra pieza, todas aquellas variables que indiquen claramente al control , como ésta debe ser maquinada.

Al largo de la pieza le asignaremos la letra Y , al ancho la letra X; W será la profundidad radial de corte; R el radio de las esquinas; D la profundidad de corte en Z; F el avance y T el radio del cortador.

El asignado de los nombres de variables que elegimos es arbitrario, en el momento de desarrollar tus macro programas podras elegir cualquier otra letra para representarlas.

Te darás cuenta de que hemos decidido colocar el cero pieza en la esquina inferior izquierda de nuestro material; esto deberá indicársele claramente al operador cuando éste referencie la pieza de trabajo sobre la mesa de la máquina. Cualquier macro deberá escribirse para que sin importar dónde decida el operador tomar el cero pieza, ésta tenga la capacidad de generar el código de maquinado correcto. De manera general, nuestro código deberá ser siempre lo suficientemente inteligente para prever cualquier acción que se le pueda ocurrir al operador y aun así , funcionar. Esto nos lleva a la regla de oro de la macro programación:

Nuestras macros deberán ser lo suficientemente flexibles para poder generar siempre un código eficiente y seguro de mecanizado

Por razón de ser un primer ejemplo y porque lo importante es que comprendas la forma en que se genera el código de mecanizado, hemos decidido simplificarlo, estableciendo un punto constante de referencia en la pieza - nuestro cero de trabajo - .En posteriores ejemplos añadiremos código que permitirá, entre otras cosas, la libre determinación del usuario del punto de referencia en la pieza de trabajo.

El siguiente programa contiene el llamado a macro. Observa que el formato de llamada es muy similar a un código M98 (llamada a subprograma), con excepción de que estamos pasando números a ese subprograma. Al llamar a la macro, ésta tomará el control del mecanizado para posteriormente retornarlo al programa que hace la llamada; en este caso el programa principal. Es posible que una macro invoque a otra dentro de sí misma; este procedimiento se conoce como anidación o "nesting" . Dependiendo de la capacidad del control será el número máximo de anidación que podemos invocar; aunque por lo regular es siete.

Programa principal:

O1000 (Número de programa)
G54 S420 M03 (Asignación de cero pieza, husillo a 420 RPM, sentido horario)
G00 X -.6 Y -.6 (Movimiento rápido a posición de claro radial para comenzar maquinado)
G43 H01 Z1 (Instalando compensación de la herramienta, llegar a Zinicial igual a 1)
G00 Z0.1 (Moverse a posición de claro)
G65 P500 X 5 Y 4 D 0.25 W 0.25 R 0.25 F 5 T 0.5 (Llamada a macro, asignación de variables de dirección)
G91 G28 Z0 (retorno a punto de referencia en Z)
G28 X 0 Y 0 (retorno a punto de referencia XY)
M30 (Fin de programa)

Con el programa anterior , aunque no lo creas, se realiza el maquinado completo de nuestra pieza. La línea que realiza todo el trabajo es la que hace el llamado a macro (G65). En esta línea hemos indicado los argumentos necesarios para que el programa genere el código adecuado a las dimensiones de la pieza de trabajo. Antes de revisar la macro debes tener presente que las variables de dirección no pueden ser programadas en nuestra macro como letras, deben ser convertidas a su variable local equivalente, para ello nos auxiliaremos de la siguiente tabla relacionando la variable de dirección con su variable local correspondiente:

X # 24	Y#25	D#7	W#23	R#18	F#9	T#20

MacroPrograma:

O500 (Numero de macro programa)
#101 = #20 + 0.1 (Aseguramos que la herramienta este en posición de claro en X)
#102 = #24 - #23 - #18 (Establece la coordenada X en el principio del radio derecho)
#103 = #23 + #18 (Establece la coordenada X al final del radio izquierdo)
#104 = #23 - #20 (Establece la posición X y Y en la esquina inferior)

#105 = #23 + #18 (Establece la coordenada Y en la parte superior de los radios inferiores)
#106 = #25- #23 -#18 (Establece la cordenada Y en la parte inferior de los radios superiores)
#107 = #24 - #23 + #20 (Establece la coordenada X final del radio derecho inferior)
#108 = #20 + #18 (Establece el arco a maquinar)
#109 = #25 - #23 + #20 (Establece la coordenada Y superior)
G00 X-#101 Y-#104 (Nos movemos a la posición inicial)
G01 Z -#7 F#9 (Bajamos a posición de corte, con avance)
G01 X #102 (Nos movemos en X hasta el inicio del primer arco)
G03 X #107 Y #105 R #108 (Posición final de maquinado del arco inferior derecho)
G01 Y #106 (Nos movemos al principio del arco superior derecho)
G03 X#102 Y #109 R#108 (Posición final de maquinado del arco superior derecho)
G01 X#103 (Nos movemos en X al inicio del arco superior izquierdo)
G03 X#104 Y #106 R#108 (Posición final de maquinado del arco superior izquierdo)
G01 Y #105 (Nos aproximamos en Y al inicio del arco inferior izquierdo)
G03 X#103 Y#104 R#108 (Posición final de maquinado del arco inferior izquierdo)
G01 Z 0.1 (Con avance nos movemos al plano R)
G00 Z 1 (Movimiento rápido al plano inicial)
M99 (Fin de macro programa)

Observa que no hemos dispuesto números de secuencia en el macro programa (números N); esto es porque los utilizaremos en una opción mas avanzada que discutiremos más adelante.
Es conveniente que revises cuidadosamente el ejemplo anterior realizando todos los cálculos matemáticos indicados en la macro. Siguiéndola paso a paso la comprenderás a profundidad. Es importante que entiendas totalmente el ejemplo antes de que proseguir tu estudio.
Observa que todos los cálculos se realizan al principio y se almacenan en variables comunes. Esto se hace generalmente para que el control gane tiempo al realizar las operaciones matemáticas, ya que dependiendo de la capacidad del control y de la complejidad de las operaciones matemáticas, si programamos los cálculos en puntos intermedios de nuestro programa, es posible que el control genere pausas indeseadas durante el maquinado mientras se realizan las operaciones, dando como consecuencia imperfecciones en éste.

Usando números de secuencia .

Los números de secuencia (números N) son usados en la macro programación , para indicar puntos de referencia de pasos programados, a partir de los cuales si se cumple determinada condición, el programa brinca a ejecutarlos.

Aunque siempre nuestros números de secuencia deben llevar un orden ascendente, no será necesario especificarlos en todas las instrucciones del programa, sino solamente en aquéllas que lo requieran.

En conjunto con los números de secuencia usamos otros comandos en macroprogramación los cuales veremos a continuación:

Salto Incondicional GOTO :

En numerosas ocasiones querrás repetir un comando o brincar una secuencia de órdenes programadas; en este caso se usa el comando GOTO.

```
GOTO 100
N67 G00 X 2
     G01 X 4.5 Y2.3 F3
    G00 Z 1
N100
     .....
```

Del ejemplo anterior, el comando GOTO indica que se debe brincar a la línea 100 y proseguir desde ahí la ejecución del programa.

Te darás cuenta que el comando GOTO por sí solo no es de mucha utilidad. Para que sea una herramienta más eficiente se programa por lo general acompañado de un condicional lógico (IF).

Usar el IF en nuestras macros, nos permite que ellas tomen decisiones en base a dos y solamente a dos condiciones posibles. Si la condición es verdadera el programara brincará a la secuencia establecida en el IF, caso contrario seguirá el flujo normal de programación.

Existen 6 comparadores lógicos que nos permiten establecer nuestra condición lógica. Cada uno de ellos es una abreviación en dos palabras de su significado en inglés (es más fácil memorizarlos si recuerdas su significado en este idioma). Los comparadores lógicos son:

EQ Igual a (Equal To)
NE No igual a (Not equal to)
LT Menor que (Less than)
LE Menor o igual que (Less or equal to)
GT Mayor que (Greater than)
GE Mayor o igual que (Greater or equal to)

Existen aplicaciones prácticamente ilimitadas en lo referente a cómo podemos usar IF en conjunción con nuestros operadores lógicos; a continuación veremos algunas.

Usando IF para pasar Banderas.

Una bandera es una variable que asume 2 valores (cierto o falso) y en base al valor que tenga la bandera la macro ejecutará una instrucción especifica. Por ejemplo, digamos que queremos maquinar una pieza en dos materiales diferentes; uno de ellos requiere que usemos soluble y el otro no. Usaremos la variable S (variable local # 19) en nuestro llamado a macro , asignándole el valor 1 para indicar a nuestro programa que se prenda el soluble y el valor de 0 para apagarlo. Las siguientes instrucciones nos ayudarán a lograr nuestro propósito:

```
IF [ # 19 EQ 0 ] GOTO 7
M08
GOTO 8
N7  M09
N8.....
```

La primera línea establece si la variable #19 es igual a 0 (soluble apagado); si es así ,el programa brinca a la línea 7, donde se encuentra el comando de soluble apagado (M09); si no es igual a 0, la condición es falsa, por lo que se ejecuta la siguiente línea que prende el soluble (M08) , posteriormente usamos un GOTO para brincar el comando M09 y proseguir la ejecución de nuestro programa

Otro ejemplo de utilización de banderas es cuando diseñamos la macro para que la herramienta pueda maquinar en dirección horario o antihorario, cuando queremos programar cambio de giro de husillo, etc...

Usando IF para presetear variables.

Conforme avances y domines las técnicas para macroprogramar, te darás cuenta que en ciertos momentos es ventajoso que la macro suponga ciertos datos, aún si estos no son ingresados a ella.

Siempre que una variable de dirección no es especificada en nuestro llamada a macro, la variable local queda indefinida o vacante. Es importante no confundir el concepto vacante o vacío con una variable a la cual se le asigna un cero, porque cero es un valor.

Generalmente, la representación de vacío en cualquier control es # 0 . Para probar si una variable ha sido incluida o no en nuestro llamado a macro, la comparamos con la variable de vacancia , si el resultado es falso la variable ha sido especificada. En caso contrario, dicha variable asumirá el valor supuesto por nuestra macro; en este caso se dice que la variable ha sido preseteada .

Hablemos de un ejemplo para aclarar esto . Se tiene una cierta familia de partes en nuestro taller, y has escrito una macro para maquinarla.

U (variable # 21) representa en tu macro, una cierta profundidad de maquinado, la cual generalmente es de 0.750" . Deseas incluir esta profundidad como un valor establecido en tu macro, es decir, como una variable preseteada, para evitar que el operador ingrese este valor de manera continua y pueda cometer un error, sin embargo, quieres mantener esta profundidad como variable en caso de que se necesite cambiarla.

A continuación escribiremos el código que te permitirá lograr tus propósitos:

```
IF [ #21 NE #0 ] GOTO 5
# 21 = 0.750
N5
```

En este caso si la variable # 21 ha sido incluida en nuestro llamado a macro, tiene un valor ingresado por el operador. por lo que el IF resulta verdadero y brincamos a la línea 5. Caso contrario nuestra variable está vacante, se ejecuta la línea posterior al IF y le asignamos el valor por defecto de 0.750".

Probando datos incorrectos ingresados por el usuario.

Una macro bien escrita debe ser capaz de checar la entrada del usuario y advertirle si ha cometido una equivocación en el ingreso de los datos. Para tal efecto usamos nuestros operadores condicionales para comparar entre la entrada ingresada y lo que nosotros predecimos como correcto. Si el resultado es aceptable, proseguimos la ejecución normal del programa, caso contrario, podemos enviar un mensaje avisando del error. Por ejemplo, si incluimos en nuestra macro una variable T (variable local #20) que represente el diámetro de la herramienta, podemos checar si nuestro valor ingresado es mayor que cero.

```
.....
IF [ #20 GT 0 ] GOTO 3
M00 ( Valor incorrecto de entrada para herramienta )
N3
.....
```

En este caso, si la variable #20 es mayor que cero brincamos a la línea 3 y proseguimos la ejecución normal de nuestro programa, caso contrario, programamos M00 (paro de máquina) y mandamos un mensaje a la pantalla indicando el error al operador.

Muchas veces podremos predecir en qué casos la entrada será incorrecta y generará un error. En el ejemplo anterior, sabemos clararemente que el diámetro de la herramienta no puede ser igual o menor que cero, digamos que este error es obvio, sin embargo habrá errores que necesitemos deducir a partir de ecuaciones más complicadas , incluso habrá situaciones de error que únicamente detectaremos hasta que ocurran.

Algunos controladores nos permiten forzar a la máquina a un estado de alarma desde nuestra macro. En estos controladores se asigna una variable de sistema, (generalmente # 3000), para incluir un comando que genere la alarma.
Veamos cómo trabaja esto utilizando nuestro ejemplo anterior .

.....

IF [#20 GT 0] GOTO 3

#3000 = 80 (Valor incorrecto de entrada para herramienta)

N3

....

En este caso si la condición lógica es falsa , el control lee la variable #3000, y la máquina se coloca en estado de alarma. Nota además que tenemos la oportunidad de colocar en la alarma un número acompañada de un mensaje que se desplegará en la pantalla de nuestro control. Esto es particularmente útil en macro programas largos, puesto que el operador puede identificar la causa del error fácilmente.
Aunque una alarma se comporta de manera relativamente similar con respecto al comando M00, su uso siempre será más ventajoso, ya que para para porseguir el programa el operador debe corregir o resetear la alarma; en cambio con el comando M00, el operador puede continuar ejecutando el programa simplemente oprimiendo botón de inicio de ciclo.

Generando ciclos de repetición.

Uno de las grandes ventajas de programar con macro es que nos permite utilizar ciclos de repetición. Aunque su sintaxis difiere entre fabricante y fabricante, la idea es prácticamente la misma, permitimos repetir una serie de movimientos hasta que se cumpla la condición establecida de antemano por nosotros.
Tratemos de explicar esto utilizando nuestro nuestro primer ejemplo de macro programación. Quizás tengamos un material que no nos permita un corte único de 0.250" de profundidad en Z con la herramienta que tenemos disponible, tendremos entonces que programar en escalón el corte de 0.250", por ejemplo, con profundidad de 0.050" en 0.050" . Una solución podría ser escribir nuestra línea de comando de macro tantas veces como sea la profundidad de corte axial, hasta completar el maquinado .

Programa principal:

O1000 G54 S420 M03 G00 X -.6 Y -.6

G43 H01 Z1

G00 Z0.1

G65 P500 X 5 Y 4 D 0.05 W 0.25 R 0.25 F 5 T 0.5

```
G65  P500 X 5 Y 4 D 0.01  W 0.25 R 0.25 F 5 T 0.5
G65  P500 X 5 Y 4 D 0.015 W 0.25 R 0.25 F 5 T 0.5
G65  P500 X 5 Y 4 D 0.020 W 0.25 R 0.25 F 5 T 0.5
G65  P500 X 5 Y 4 D 0.250 W 0.25 R 0.25 F 5 T 0.5
G91 G28 Z0
G28 X 0 Y 0
M30
```

El programa anterior nos muestra esta idea. Te darás cuenta de que en cada llamada a macro hemos incrementado la variable local D (que representa nuestra profundidad) en la profundidad de corte deseada. Esto no es lo óptimo, puesto que solamente funciona para cortes incrementados en 0.050" ; sin embargo demuestra perfectamente bien la mecánica de un ciclo; en este caso incrementamos la profundidad de corte en Z, hasta la profundidad deseada realizando en cada incremento el mecanizado en el plano XY.

Entendiendo cómo se comporta un ciclo, modificaremos nuestra macro de forma que acepte incrementos de corte en Z; para esto necesitaremos una nueva variable en nuestra línea de comando a macro, la cual será H (variable local #11), además de algunas de las funciones matemáticas anteriormente descritas.

Por razones de mayor claridad, en el siguiente programa explicamos entre paréntesis únicamente los cambios que hemos realizado en éste.

```
O500 ( Numero de macro programa )
#101 =  #20 +0.1
#102 = #24 - #23 - #18
#103 = #23 +  #18
#104 = #23 - #20
#105 = #23 + #18
#106 = #25- #23 -#18
#107 = #24 -  #23 + #20
#108 = #20 + #18
#109 = #25 -  #23 + #20
#110 = FIX[ #7 / #11 ]  ( Redondeamos hacia abajo número de cortes en dirección Z )
#111 = # 11  * ABS[ #110 –[ #7 / #11]] ( Calculamos el residuo en Z )
IF [ #110 LE 0 ] GOTO 15  ( Probamos el número de pasadas )
#113 =#11 ( Nuestra variable que incrementará el corte en Z )
GOTO 16
```

```
N15 #113 = #112 ( asignando residuo en Z a la profundidad de corte )
N16 G00 X-#101 Y-#104
N20 IF [ #113 GT # 7 ] GOTO 100 ( Condición de fin de ciclo )
G00  Z 0.1
G01 Z -#113 F#9 ( Bajamos a posición de corte calculada )
G01 X #102
G03 X #107 Y #105 R #108
G01 Y #106
G03 X#102 Y #109 R#108
G01 X#103
G03 X#104 Y #106 R#108
G01 Y #105
G03 X#103 Y#104 R#108
G01 Z 0.1
G00 Z 1
#111 =#111-1 ( Decrementamos contador )
IF [ #111 LE 0 ] GOTO 25 ( Probamos condición de contador )
#113 = #113+#11 ( Incrementamos profundidad de corte en Z)
GOTO 16 ( Loop )
N25 IF [ #112 GT 0 ]  GOTO 30 ( Probamos condición de residuo en Z )
GOTO N100  ( Si falla condición acaba programa )
N30 #113 = #113+#112 ( A la profundidad de corte agregamos residuo )
GOTO 16
N100 M99
```

Esta versión mejorada de nuestro macro programa no solamente acepta incrementos de corte en Z, sino que es también capaz de considerar un residuo en Z, como resultado de dividir la profundidad de corte total en Z, entre la profundidad de corte incremental.

Veamos cómo funciona esta macro paso a paso:
Supongamos que nuestra profundidad de corte total en Z (variable local # 7) es de 0.250" y el operador programa cortes incrementales de 0.160" (que especificamos en la variable local #11), obviamente , sólo podremos realizar un corte de 0.160" más otro de 0.090" que nos completen las 0.250". Para obtener el número de veces que baja la herramienta en Z, usamos la función FIX , la cual redondea hacia abajo el resultado de dividir 0.250 entre 0.160, almacenándolo en la variable común #110, cuyo valor en el caso del ejemplo es 1.

La variable #111 tiene como función almacenar el residuo en Z , que en este caso son precisamente las 0.090".

A continuación, por medio de un IF, nuestra macro pregunta si la variable #110 es cero. Esta condición es necesaria para checar si el operador ha ingresado un valor de profundidad de corte axial mayor que la profundidad de corte total en Z , de ser así, el programa brinca a la línea 15, en donde almacenamos en nuestra variable #113 el residuo en Z, que en este ceso tomará el valor de la profundidad de corte total . Si nuestra variable #110 no es cero , el IF es falso, como ocurre para los valores dados en el ejemplo, por lo que almacenamos en la variable #113 el valor de #11 prosiguiendo de ahí a la línea 16 . En la línea 16 posicionamos la herramienta para inicio de corte, la indicamos porque es un punto que nos sirve de referencia para el maquinado, tantas veces como se requiera repetir el ciclo.

En la siguiente línea (N20) tenemos un condicional que finaliza el programa cuando #113 es mayor que # 7. Para nuestro ejemplo #113 vale 0.160" que es menor a 0.250" por lo que la condición es falsa y proseguimos la ejecución de la macro.

A continuación bajamos a la profundidad de corte especificada por #113, ejecutando posteriormente las instrucciones que realizan el perfilado de nuestra pieza. Dete cuenta de que hemos incluido en dichas instrucciones la salida y llegada en Z de la herramienta a un punto seguro (0.100" y 1"), hacemos esto para que en caso de que sea necesario repetir el ciclo , la herramienta esté en un punto libre de interferencia para dirigirse al punto de inicio de corte .

Decrementamos a la variable #111 en 1, (ahora #111 vale 0), preguntando a continuación por medio de un IF si el contador es menor o igual que cero. Si nuestra condición es verdadera , como es este caso, brincamos a la línea 25, en caso contrario, incrementamos a la variable #113 en #11 y regresamos a la línea 16.

En nuestra línea 25, usamos otro IF para preguntar si nuestro residuo es mayor que cero, si es así brincamos a la línea 30, en caso contrario,el resultado de la división de la profundidad de corte total (variable #7) entre la incremental (variable #11) ha sido exacta , por lo que terminamos nuestra macro en la siguiente línea.

En nuestra línea 30 incrementamos a la variable #113 el residuo en Z, (#113 vale ahora 0.250"), brincando de nuevo a la línea 16. Colocamos nuestra herramienta nuevamente en posición de

inicio de corte en el plano XY. Entramos a ciclo, ya que #113 sigue siendo menor que #7 (0.250" no es más grande que 0.250") . Bajamos a profundidad de corte nuestras 0.250" y realizamos de nuevo el maquinado de la pieza.

Decrementamos #111 en 1, ahora esta variable es negativa . De ahí la razón de usar como comparador lógico en el siguiente condicional menor o igual. Dicho condicional es verdadero por lo que el programa brinca a la línea 25. Incrementados #113 en el residuo brincamos a la línea 16; la herramienta se vuelve a colocar en posición de inicio de corte, pero el IF de ciclo resulta verdadero ya que #113 que vale ahora 0.340" y es mayor que 0.250" , por lo que ahí termina nuestro macro programa.

Este ejemplo nos ha servido para darnos cuenta de la aplicación de la mayoría de los puntos que hemos platicado. Aunque también ha servido para ver cómo aumenta el grado de complejidad de nuestra macro, cuando programamos teniendo en mente las diversas situaciones que pueden presentarse a la hora de que el operador ingrese datos a la máquina.

Aplicaciones relativas al control (variables del sistema).

Hemos mencionado ya anteriormente, que la macroprogramación no fue diseñada para ser usada como herramienta de programación , sino para proveer una interfase entre el fabricante y el control de la máquina. Es por ello que podemos accesar a través de ella a numerosas herramientas relativas todas al controlador. Pero tanto el modo como el grado de acceso a dichas herramientas varían de fabricante a fabricante. Esto hace que sea imposible el describirlas completamente , razón por la cual hablaremos sólo de algunas de ellas. Para una información más precisa necesitas recurrir al manual del control de tu máquina.

Acceso a los *offsets* de herramientas.

La mayoría de las veces podremos accesar a los valores de los *offsets* de las herramientas dentro de nuestra macro. De hecho podremos cambiar el valor de los *offsets* dentro de la misma. Para hacerlo, necesitas indagar en el manual de tu control en particular, el rango de las variables de sistema destinadas para almacenar *offsets* . Digamos que un control en particular utiliza la serie #2000 para almacenar los valores de los *offsets*.

El comando siguiente lee el valor del *offset* registrado en la variable #2001 y lo almacena en la variable #101

```
#101 = #2001
```

Una aplicación de esta técnica ocurre cuando diseñamos una macro de modo que use coordenadas de centro de herramienta en lugar de utilizar comandos de compensación por radio, evitándonos los problemas inherentes al uso de estos comandos. – de hecho éste es mi método favorito de programar - . Sin embargo debemos incluir en nuestra macro el radio o diámetro de la herramienta elegida por el operador. Una opción puede ser incluir en nuertro llamado a macro, una variable de dirección destinada a almacenar el radio de la herramienta. Pero en programas especialmente largos, encontrarás que podemos quedar escasos de variables. La opción más lógica es leer directamente de las variables de *offset* registradas en la máquina, el correspondiente a la herramienta que estamos utilizando, asignándolo posteriormentre a una variable común tal como está mostrado líneas arriba. En este caso el operador debera declarar los valores de *offset* de las herramientas usadas en la macro, en el mismo orden establecido en nuestro programa.

También es posible lo siguiente:
#2001 = #101

En este caso estamos modificando con la variable #101 el valor del *offset* registrado en #2001. Aunque esta aplicación no tiene tantos usos en lo que se refiere a programación, podemos utilizarla, por ejemplo, cuando estamos usando una probeta de medición o palpador, dispositivo que utilizamos para sensar electrónicamente las dimensiones de una pieza y requerimos una tolerancia de al menos diezmilésimas .
Digamos que hemos rectificado un barreno y comandamos la probeta a sensar el diámetro mecanizado del agujero. Una vez que hemos registrado éste en una variable dentro de nuestra macro, lo comparamos con el diámetro original programado. La diferencia de diámetros dividida entre dos, será el monto que tendremos que aumentar a nuestro *offset* de radio de herramienta.
Nota:
En este caso y en general, siempre que usemos un ciclo de repetición en combinación con un palpador necesitamos programar un condicional con un valor término que evite que caigamos en un *loop* sin final, puesto que por más que queramos, nunca podremos acercarnos exactamente a la medida programada debido al error de punto flotante inherente a toda operación matemática realizada por procesador.

Generando alarmas en el control.
Ya habíamos hablado un poco de las alarmas. Sin embargo su importancia al escribir macro programas es tal que merece ocuparnos un poco más de ellas.
Una macro bien escrita debe ser capaz de verificar todas las variables ingresadas por el operador. De este modo aseguraremos que desarrolle el código esperado por nosotros.

Retomemos el llamado a macro que hemos escrito para usarlo como ejemplo de lo anterior:

G65 P500 X 5 Y 4 D 0.250 W 0.25 R 0.25 F 5 T 0.5 H 0.50

Las variables de dirección usadas en nuestra llamada a macro junto con su correspondiente variable local, se enlistan a continuación:

X # 24	Y#25	D#7	W#23	R#18	F#9	T#20	H#11

Ahora incluiremos las siguientes instrucciones al principio de nuestra macro:

```
O500 ( Numero de macro programa )
IF [ #24 NE #0 ] GOTO 5
#3000 = 1001 ( Falta ingresar coordenada X )
N5 IF [ #25 NE #0 ] GOTO 6
#3000 = 1002 ( Falta ingresar coordenada Y )
N6 IF [ #7 NE #0 ] GOTO 7
#3000 = 1003 ( Falta ingresar profundidad de corte total Z )
N7 IF [ #23 NE #0 ] GOTO 8
#3000 = 1004 ( Falta ingresar profundidad de corte radial W )
N8 IF [ #9 NE #0 ] GOTO 9
#3000 = 1005 ( Falta ingresar avance F )
N9 IF [ #20 NE #0 ] GOTO 10
#3000 = 1006 ( Falta ingresar radio de herramienta )
N10 IF [ #18 NE #0 ] GOTO 11
#18 =0
N11 IF [#11 NE #0 ] GOTO 12
#11=#7
N12
.....
```

Nota como la macro genera una alarma y el motivo de la falla cuando compara las variables ingresadas con la variable de vacancia (#0). Además ,observa cómo preseteamos las variables que no son importantes para el desarrollo de nuestra macro. Si el operador no incluye el radio , simplemente se maquinará un cuadrado con esquinas y si no introduce la profundidad de corte

simplemente se maquinará un cuadrado con esquinas y si no introduce la profundidad de corte incremental en Z, preseteamos esta variable de modo que sea igual a la profundidad de corte en Z total.

Si posees un controlador más antiguo con poca capacidad de memoria o has diseñado una macro con mucho más variables, quizás pienses que la técnica anterior no es la más adecuada a tus propósitos , por ello emplearemos otro método más corto y sencillo para filtrar las entradas a nuestra macro.

```
O500 ( Numero de macro programa )
IF [ #24 NE #0 ] GOTO 98
IF [ #25 NE #0 ] GOTO 98
IF [ #7 NE #0 ] GOTO 98
IF [ #23 NE #0 ] GOTO 98
IF [ #9 NE #0 ] GOTO 98
IF [ #20 NE #0 ] GOTO 98
IF [ #18 NE #0 ] GOTO 11
#18 =0
N11 IF [#11 NE #0 ] GOTO 12
#11=#7
N12 ..... ( Se ejecuta programa )
GOTO 99
N98 #3000 = 1001 ( Falta ingresar variable en macro 500 )
N99 M99
```

Como te darás cuenta , todas las variables de importancia para el maquinado comparten una alarma en común la cual se desplegará en pantalla si el operador olvida ingresar alguna de ellas, siendo entonces tarea relativamente sencilla para él, rellenar correctamente los datos.

Acceso al Panel de Control.

Ciertos controladores permiten el controlar diversos comandos del panel de control de la máquina desde nuestro macro programa. Dependiendo de la complejidad del control se puede acceder a comandos sencillos como el inicio de ciclo o el avance por bloque, hasta el bloqueo total de máquina o el paro de emergencia.

Esta opción puede aplicarse por ejemplo, cuando diseñamos un ciclo de machueleado. Es crítico para la operación el bloquear el botón de paro de avance (feed hold), así como el de desfase de avance (feed override). Un ciclo diseñado por el fabricante normalmente toma control de estos

comandos desconectándolos temporalmente, misma acción que debemos tener en cuenta en nuestra macro de machueleado de forma que no afecte el maquinado.
El método para controlar estas funciones varía de control en control; para obtener información al respecto deberás checar tu manual.

Acceso a Modo de Operación.

En la mayoría de los controladores que permiten la programación por macro es posible checar dentro del macro programa el estado actual de modo de operación del control. Por estado actual queremos dar a entender el modo en que el controlador interpretará los datos ingresados. Esto es, si estamos en modo incremental (G91) el controlador se comportará diferente que si estamos en modo absoluto (G90) .

Esta opción es útil cuando programamos un modo de operación distinto en nuestro macro programa al que esta prefijado en el control . Por ejemplo, digamos que queremos programar una macro usando mm y queremos regresar al modo original de operación prefijado en el control de nuestro control (ya sea este mm o pulgadas) . Almacenamos entonces el modo original de operación en una variable de tipo común , instalamos comando de mm (G21) y ejecutamos la macro, la cual realizará los movimientos programados en el sistema de unidades seleccionado, terminando el programa regresaremos al modo original de operación.
Como nota explicativa de este ejemplo, aclararemos que la macro deberá leer y almacenar también en variables comunes los datos que sea necesario convertir al sistema de unidades que se quiera cambiar, tal como los *offsets* de las herramientas.

Creando Códigos G, M y T .

Algunos tipos de controladores permiten crear códigos G, M y T propios. De hecho, todos los fabricantes diseñan sus ciclos enlatados utilizando macro programación. También algunos códigos especiales necesarios para controlar diversos dispositivos o que realizan ciertos movimientos especiales de mecanizado, son diseñados a partir de macro programas. Para nosotros solamente representan un código G más.
Para saber si puedes acceder a esta opción y como utilizarla , debes indagar en tu propio manual de programación.

Verificando Macro Programas.

El verificar macro programas es más difícil que el verificar un programa CNC estándar. Esto se debe a que tenemos que predecir el comportamiento de la macro bajo cualquier tipo de entrada ingresada por el operador. Un macro programa se diseña para operar en conjunto con una máquina de CNC , lo que nos obliga a ser extremadamente cautelosos en su prueba.

El método de prueba que yo sugiero para verificar una macro es el siguiente:

- Escribe tu macro en papel y revisa su lógica, ingresa valores en ella checando todas las operaciones matemáticas. Anota en orden las coordenadas que vas obteniendo.
- Dibuja las coordenadas obtenidas , verificándolas contra tu pieza de trabajo comprueba que la herramienta se encuentre donde debe estar .
- Si en tu macro existen ciclos de repetición, comprueba que terminan de forma satisfactoria, esto es, sin caer en un *loop* sin final, sin terminar con un valor de más o con un valor de menos.
- Ingresa valores extremos a aquéllos que normalmente ingresarías, tratando de detectar los errores que puedan ocurrir, por ejemplo una división entre cero.
- Estando seguro de que tu lógica es correcta, ingresa la macro en tu control .Córrela bloqueando la máquina para verificar errores de sintaxis.
- Coloca los *switchs* de rápido y avance en mínimo. Activa el comando de *Single Block* , desbloquea la máquina y corre el macro programa observando el movimiento en todos los ejes, sin pieza de trabajo.
- Estando seguro ya de los movimientos de tu macro y de que tu lógica ha sido correcta , maquina la primera pieza con precaución .

CAPITULO 2

Técnicas Avanzadas.

Herramientas necesarias para empezar a macro programar.

El capítulo anterior sirvió para introducir el concepto de macro programación, así como conocer diversas técnicas para programar utilizando esta herramienta.

Todo lo explicado te puede servir si quieres programar una macro directamente en tu control ; pero muchas veces no tendrás ocasión, sea porque tu máquina no viene equipada con esta opción o porque simplemente difícilmente puedes desperdiciar tiempo de maquinado mientras dominas la técnica.

Imagínate el poder desarrollar una librería de macros en tu P.C , creando patrones estándar de maquinado que te pueden servir para ahorrar tiempo en la creación de un nuevo programa. Quizás necesites, por ejemplo, un ciclo de taladrado en picoteo, el hecho de que ya lo tengas programado en tu computadora , que solo rellenes variables y la máquina te genere el código haciendo el trabajo por tí es realmente atractivo y lo que es mejor, al poco tiempo de generar macros en tu P.C, te darás cuenta que posees más recursos de memoria, mayor rapidez de cálculo y sobre todo mayor flexibilidad que en lo general te ofrezca cualquier control.

Para desarrollar esto, necesitarás conocer bien por lo menos un lenguaje de programación. Todos los ejemplos de este texto están escritos en lenguaje PASCAL , sin embargo , el lenguaje que tú domines es suficiente para empezar a generar macro programación.

Necesitarás también sólidas bases de matemáticas; no necesitas ser un Claudio Pita [*], pero te darás cuenta que entre mejor las domines, podrás desarrollar aplicaciones más complejas. Si posees solamente nociones básicas de geometría analítica y trigonometría, sólo generaras macros sencillas. Para desarrollar macros complejas , por ejemplo el maquinado de un escalón con figura arbitraria o generación de código para maquinado de superficie en 3D, necesitas entender algo de métodos numéricos.

También necesitas tener experiencia previa de mecanizado, sea en tomo o en fresa; por lo menos en mi experiencia propia, sé lo difícil que puede ser llegar a dominar algo sin haberlo palpado antes físicamente. Elementos de mecanizado como ranura, careado, cortador recto, te deben ser ya familiares.

---

[*] Claudio Pita (alias El Grandioso) : Matemático, escritor y filósofo. Profesor de Matemáticas Avanzadas en la Universidad Panamericana.

Necesitarás una buena dosis de imaginación, creatividad y paciencia. El programar macro no es fácil ; te pondré un ejemplo: es como cuando te enfrentar a armar un rompecabezas gigante, cuando está armado parece que fue tarea fácil, pero solo tú que enfrentaste el problema, sabes su grado de dificultad .

Es lo mismo al programar una macro, en general lo que vale de ahí no es el código de programación, sino la solución matemática que encontraste para generar el *tool path* de mecanizado , lo que la hace realmente valiosa.

Por lo tanto, siempre que programemos macro procuraremos generar un *tool path* inteligente. Para ello, a la par de estudiar algunas técnicas avanzadas de programación , las cuales veremos en este capítulo, revisaremos fundamentos del corte de materiales que nos servirán para deducir y programar en forma óptima el mecanizado.

Determinando G02 Versus G03.

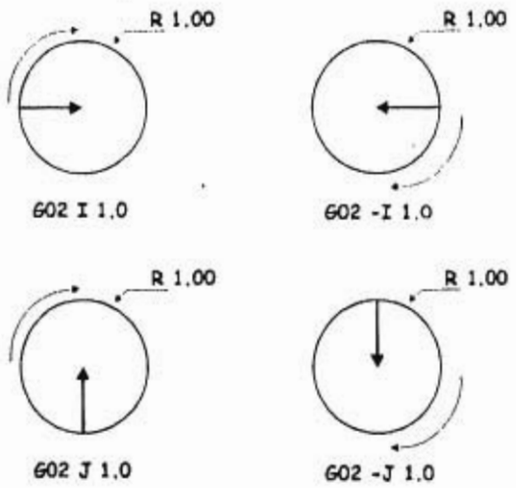
Con seguridad te preguntarás por qué en un capítulo donde se habla de técnicas avanzadas se introduce algo tan elemental como pueden ser los comandos radiales G02 y G03 . Esto se debe a que nuestros macro programas serán diseñados utilizando básicamente los comandos G01, G02 y G03 y te sorprenderá saber las veces que la mala interpretación de los dos últimos comandos en particular, causa errores en nuestra programación. Así que será mejor el repasarlos un poco. Además, quizás descubras que estos comandos pueden tener usos especiales o avanzados, que puedan servirte en el desarrollo de macro programas que requieran por ejemplo, la selección del plano del mecanizado.

Los comandos de movimiento circular con sentido horario G02, y sentido anti horario G03 interpolan una trayectoria circular en nuestro mecanizado. Al usar estos comandos se incluyen por lo regular las coordenadas que indican el final del movimiento, aparte de una referencia que permite al control conocer el centro del arco que está trazando la herramienta.

En la mayoría de los controles nuevos se utiliza para indicar esta referencia, la letra "R", con la cual se indica al control el radio del arco que queremos generar . Partiendo de esto, el control interpola leyendo el comando anterior y posterior y genera un trazo de herramienta correcto.

En controles más antiguos se utilizan vectores directores, (I, J , K , para eje X, Y y Z respectivamente) que indican a partir del punto de inicio de nuestro comando de arco, la distancia al centro del arco que queremos generar. Siempre será más fácil programar con el primer método, indicando únicamente el radio de maquinado, pero esto no es lo óptimo. Si nuestra idea es generar macros que se acoplen a cualquier tipo de control será necesario pues utilizar I, J y K . Por eso todas las macros en este texto, que usen comandos circulares, serán escritas usando vectores directores en lugar de comando por Radio.

Existe otra razón que nos hace ser congruentes al usar vectores directores. Cuando queremos que la herramienta maquine un círculo completo, es posible omitir al escribir el comando las coordenadas finales del arco (puesto que vienen siendo las mismas con las que iniciamos) . Sin embargo, si en lugar de indicar al control por medio de un vector director , el centro del arco a maquinar y utilizamos la letra R para indicar el radio, la herramienta no se moverá y el control simplemente ignorará este comando saltándose al próximo.



Las figuras anteriores muestran cómo se programa un círculo completo utilizando solamente su vector director.

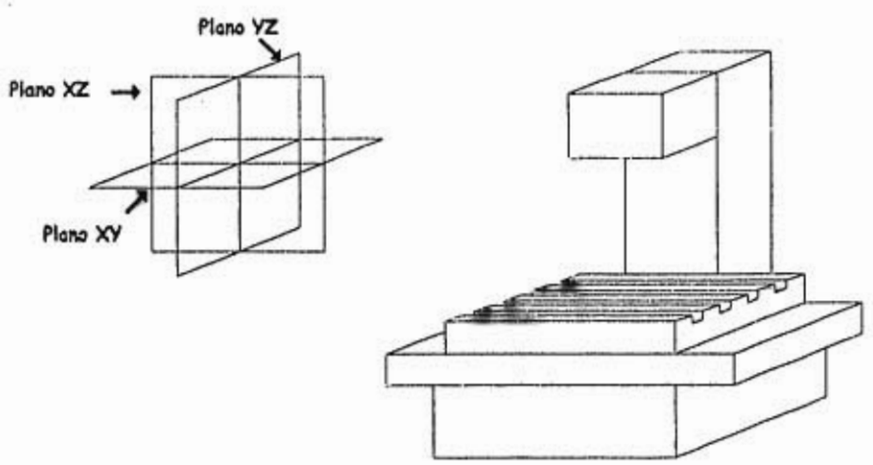
Hay un punto fino que también debemos considerar cuando usamos comandos circulares y es el decidir el sentido del movimiento de ta herramienta cuando se cambia el plano de maquinado. La regla de oro para determinar dirección de G02 o G03 en cualquier control es la siguiente:

El movimiento en sentido horario y anti horario (G02 / G03) debe ser evaluado al mirar el movimiento de la herramienta desde el extremo positivo del eje perpendicular donde no ocurre el movimiento.

En un centro de maquinado con tres ejes, si estás maquinando en el plano XY (como normalmente ocurre), determinar el uso de G02 o G03, es tan fácil como el observar la pieza desde arriba. Así, si la orientación del plano de la pieza encaja con la forma en que ésta se ha

preparado para el maquinado, para programar G02 o G03 simplemente tendrás que ver el plano de mecanizado.

Pero si no estás maquinando en el plano XY, quizás evaluar la elección de G02 y G03 ya no sea tan fácil. Por ejemplo, si el movimiento circular tiene lugar en el plano XZ, deberás evaluar G02 o G03 desde el lado positivo del eje Y. Para un centro de maquinado vertical, esto significa que deberás evaluar el movimiento de la herramienta desde el lado donde se encuentra la columna, esto es , desde la parte posterior de la máquina. Por el contrario si decidimos maquinar en el plano YZ, el movimiento de la herramienta deberá ser juzgado desde el lado positivo del eje X, esto es, del lado derecho de la máquina.



Relación entre los tres planos de un centro de maquinado vertical.

También es importante considerar la selección del plano de trabajo, cuando comandamos movimientos circulares. Debemos indicar al control el plano correcto para el maquinado por medio de un código G; dicho comando será modal . En la mayoría de los controladores, para especificar el plano XY se usa el comando G17, dicho comando por lo general se inicializa al prender la máquina , es decir , será el modo de trabajo estándar de la maquina , por esta razón si generalmente trabajas en el plano XY (como normalmente será) no habrá necesidad de especificar el comando G17 en tus programas.

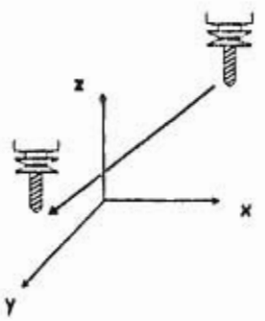
Si hacos movimientos circulares, entre otras cosas en el plano XZ o YZ, deberás incluir el comando de selección de plano adecuado antes del movimiento circular. Para EIA -- ISO, G18, selecciona el plano XZ y G19, selecciona YZ.

Usando Z inicial y Plano R.

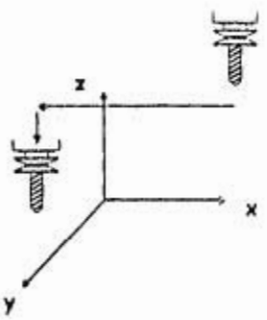
Esta técnica es de uso exclusivo para centro de maquinado; es una herramienta muy sencilla pero extremadamente útil.

Al aproximarnos a maquinar la pieza de trabajo podemos aproximarnos de 2 maneras distintas para comenzar el maquinado. Una de ellas es moviendo los 3 ejes de manera simultánea hasta llegar a la pieza de trabajo ; la otra, es posicionar primero en XY y posteriormente bajar en Z.

La primera técnica es más ventajosa, pues avanzamos una menor distancia para posicionarnos y por consiguiente, reducimos el tiempo ciclo; la segunda tiene a su favor, el ser más segura ya que la herramienta es desplazado mediante dos movimientos subsecuentes (como puede observarse en la figura) y asegura que la herramienta libra cualquier obstáculo en su camino.



Posicionamiento rápido en 3 ejes.



Posicionamiento rápido en el plano XY
Bajada en Z.

Siempre será más conveniente usar la primera técnica, ya que si disponemos de un equipo CNC , debemos procurar aprovechar todas las ventajas que nos reporta , como es la velocidad de posicionamiento en rápido .Pero ante todo, debemos procurar programar *siempre* con precaución. Para aprovechar ambas ventajas lo que se hace es definir un plano de referencia en Z arriba del cero pieza en el eje Z (nuestra Z inicial) , con la distancia que juzguemos conveniente para que la herramienta libre los obstáculos que pueda encontrar en su camino hacia la pieza de trabajo, de tal suerte que la herramienta se desplace transversalmente primero en X, Y y Z , llegue al plano inicial referido, y baje con desplazamiento únicamente en Z.

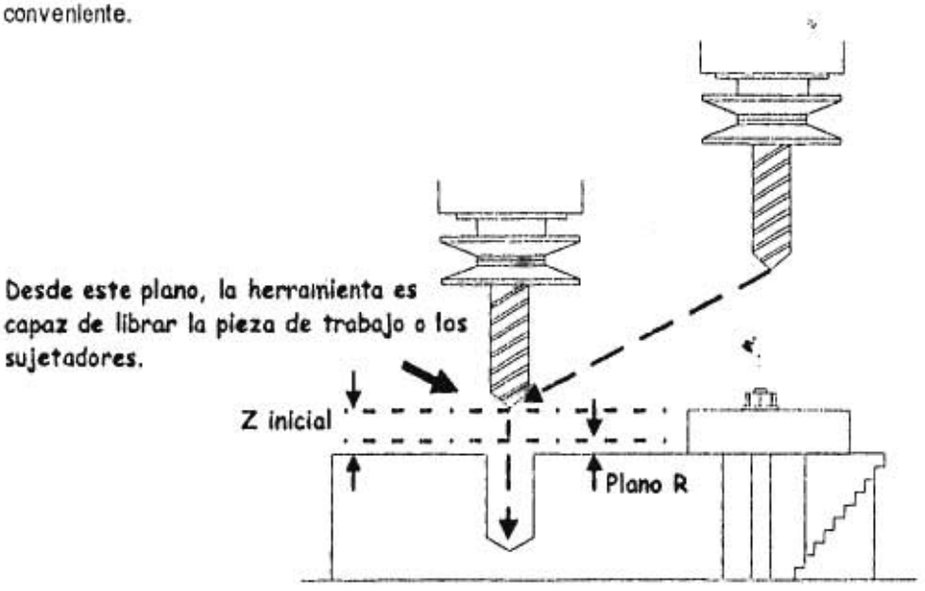
El siguiente código muestra el uso de esta técnica.

```
G00 X xinicial Y yinicial G43 Z 1.0 H01
 G00 Z 0.1
```

G01 Z -0.25 F 3.5

La primera linea del programa desplaza con movimiento rápido a nuestra herramienta simultáneamente en los tres ejes, hasta llegar a una distancia de 1 inch. - nuestro plano de referencia - por encima de nuestra pieza en Z, instalando a la vez compensación de altura de la herramienta. Posteriormente date cuenta que la herramienta avanza con movimiento rápido en Z hasta un claro de 0.100 inch. sobre la pieza, para después bajar a profundidad de corte en este eje. Este espacio se conoce como plano R y es un plano que define el cambio de movimiento rápido (G00) a avance con corte (G01) un espacio de claro antes de comenzar el maquinado, permitiéndonos ganar tiempo ciclo adicional.

Es importante remarcar que tanto la elección del valor de esta Z inicial , así como la distancia del plano R es totalmente a juicio del programador y puede tomar cualquier valor que él juzgue conveniente.



Es una buena idea programar en nuestras macros, tanto nuestra Z inicial como nuestro plano R, como variables dentro de las mismas , de modo que podamos cambiar fácilmente el valor más conveniente para nuestro mecanizado.

Técnicas de Formateo .

Todo programa CNC debe contener líneas que permitan su correcta ejecución , código que haga que dicho programa funcione correctamente y con seguridad. Este código se inserta al inicio y al

final del programa CNC que realmente desarrolla el maquinado. El código preparatorio del programa se llama formato.
Este código generalmente consiste en comandos que nos aseguran que la máquina se encuentra en posición correcta al inicio del maquinado, que esté en el modo de operación adecuado para trabajar (modo absoluto o incremental , pulgadas o mm. , etc..) , que las condiciones de corte sean las correctas, etc..

Siempre será importante el formatear tus programas. La mejor razón para ello, es permitir que tu código pueda ser llamado por bloques independientes . Muchas veces el operador necesita remaquinar alguna parte de la pieza y requiere solamente llamar a una sola herramienta de las que integran el programa. Por ejemplo, digamos que has diseñado un programa para maquinar un barreno con una cierta tolerancia y piensas utilizar una broca de centros, broca normal y una rima para el rectificado. Al correr el programa descubres que la broca no ha maquinado a la profundidad suficiente y en ese caso querrás remaquinar solamente con la broca y posteriormente finalizar con la rima, maquinar desde el principio utilizando la broca de centros será solamente una pérdida de tiempo. Si se trata solamente de un barreno no tiene mayor importancia , pero si es una cantidad considerable de barrenos nos hace pensar la necesidad de tener una manera de optimizar nuestro programa .

Para lograrlo, debemos tener en mente diseñar cada programa de modo que cada herramienta sea independiente del resto; esto tiene más sentido si piensas en cada herramienta que realice una operación determinada , como un mini programa autosuficiente , capaz de activar las funciones requeridas para su correcto desempeño por sí mismo.

Hablemos de un ejemplo más específico que pueda convencer por qué usar la técnica de formateo. Digamos que estás programando en un centro de maquinado con 2 herramientas que maquinan en secuencia ; por comodidad llamémosles herramienta 1 y 2 ; ambas con 400 RPM. Conforme vas programando, en el código que llama a la herramienta 2 decides ya no especificar las RPM a 400 , ya que este comando es modal. Todo está perfecto cuando corres el programa en forma secuencial, la herramienta 2 sigue tras la herramienta 1 con las RPM requeridas; pero imagínate que el operador corre todo el programa antes de descubrir que la segunda herramienta a hecho algo mal y decide correr el programa a partir de la segunda herramienta . ¿ Apostarías que la segunda herramienta rota a 400 RPM ?, mejor no, pues perderías. Lo mejor que podría pasamos en este caso sería que el control mandara una señal de alarma antes de posicionarse e iniciar el corte, lo peor podría ser un daño severo a la máquina, no digamos a la herramienta.

Las cuatro maneras de formatear

Dependiendo del estilo y de la antigüedad de tu máquina CNC , habrá 2 ó 4 formas básicas de formateo.

Para aquellas máquinas que no posean un cambio automático de herramienta , como algunos centros de maquinado o máquinas EDM , añadiremos código de formato solamente al inicio y al final de nuestro programa.

Con las máquinas que poseen dispositivo automático de cambio de herramienta o aquéllas que poseen torreta de cambio automática (tales como tornos , prensas, etc..), se debe formatear de acuerdo a lo siguiente:

1. Formateo de inicio de programa.
2. Formateo de inicio de herramienta.
3. Formateo de fin de herramienta.
4. Formateo de fin de programa.

Es conveniente aclarar que el método de formateo que te presentamos es solamente una guía para el desarrollo de *tu* método y que en función de la antigüedad de tu control, serán aplicables algunos de los comandos expuestos a continuación. Desarrollamos este formato para que observes la estructura general del mismo y puedas desarrollar una secuencia aplicable a tus requerimientos . Si la sigues, con toda seguridad no cometerás errores básicos en tu programa.

Ahora presentaremos diferentes formas de formatear un programa según la máquina que estés usando; procuraremos casi siempre en los ejemplos de programas EIA - ISO , aclarar entre paréntesis el uso de comandos y su función , para mayor facilidad de comprensión. En un programa EIA - ISO normal no es necesario el uso de paréntesis.

Formateo para centro de maquinado vertical

Formato de inicio de programa:

O0001 (Número de programa)
G91 G28 Z0 (En modo incremental, confirmar que el eje Z este en posición de referencia
- siempre será conveniente el mandar primero a posición de referencia el eje Z por seguridad -)
G28 X 0 Y 0 (Confirmar que el eje X y Y estén en posición de referencia)
G90 G54 G80 G49 G40 G20 G94 (Instalar modo absoluto , definir localización máquina del cero pieza , desactivar - si es que están activados - modo de ciclo repetitivo, cancelar compensación de altura de herramienta , compensación de radio; confirmar programación en pulgadas y avance en unidades sobre minuto)

Formato de inicio de herramienta:

T01 M06 (Realizar cambio de herramienta .En caso de que la máquina tenga ya la herramienta adecuada en el husillo el control ignora el comando de cambio de herramienta)
G00 X xinicial Y yinicial G43 H01 Z 1.0 S300 M03 M08 (Movimiento rápido a inicio de maquinado en X , Y y Z inicial, instalar compensación de altura para la herramienta y prender husillo con RPM especificadas en sentido horario, si es necesario, prender soluble)
G00 Z 0.1 (Mover herramienta a plano R)
Inicia secuencia de maquinado....

Formato de fin de herramienta :

Termina secuencia de maquinado...
G00 Z 0.1 (Mover herramienta a plano R o a Z inicial)
G91 G28 Z 0 G49 M09 M19 (En modo incremental, mandar husillo a posición de referencia en Z - en el retorno cancelar compensación de altura - .Si se instaló algún comando de tipo modal, cancelarlo, apagar soluble , orientar husillo para cambio de herramienta)
T02 (Rotar la siguiente herramienta en el magazine, lista para cambio automático)
M01 (Comando de paro opcional)

Formato de fin de programa:

G91 G28 Z 0 M19 G49 (Mandar a posición de referencia en Z , orientar herramienta en el husillo, cancelar compensación de longitud)
T01 M06 (Instalar la primera herramienta en el husillo)
G28 X 0 Y 0 (Mandar a posición de referencia el husillo en el eje X y Y)
M30 (Fin de programa)

Formateo para centro de maquinado horizontal.

La principal diferencia entre el formato de código para un centro de maquinado vertical y otro horizontal, radica en el sitio donde tiene lugar el cambio de herramienta.Si para un centro de maquinado vertical el cambio tiene lugar en la posición de referencia en Z, no ocurre lo mismo para un centro horizontal, donde por lo general el cambio de herramienta ocurre en la posición de referencia YZ .

Además existe otro punto que debemos considerar al programar nuestro formato para centro de maquinado horizontal. La mayoría viene provisto con un dispositivo mecánico que permite rotar la pieza de trabajo; este dispositivo se conoce como indexador .

Por seguridad **siempre** debemos observar dos reglas para programar rotación del indexador ; no seguirlas podría implicar un daño severo a la máquina.

1° Verificar al inicio y fin de programa que el indexador se encuentre en su posición de referencia.

2° Siempre que se programe cambio de herramienta, antes de mover el husillo se debe mandar a punto de referencia el indexador.

Formato de inicio de Programa:

O0001 (Número de programa)

G91 G28 Z0 (En modo incremental, confirmar que el eje Z esté en posición de referencia)

G28 X 0 Y0 (Ejes XY en posición de referencia)

G90 G54 G80 G49 G40 G20 G94 (Instalar modo absoluto , definir localización máquina del cero pieza , desactivar - si es que están activados - modo de ciclo repetitivo, cancelar compensación de altura de herramienta , compensación de radio; confirmar programación en pulgadas, avance en unidades sobre minuto)

Formato de inicio de herramienta:

T01 M06 (Realizar cambio de herramienta . En caso de que la máquina tenga ya la herramienta adecuada en el husillo el control ignora el comando de cambio de herramienta)

G00 X xinicial Y yinicial G43 H01 Z 1.0 S300 M03 M08 (Movimiento rápido a inicio de maquinado en X , Y y Z inicial, instalar compensación de altura para la herramienta y prender husillo con RPM especificadas en sentido horario, si es necesario, prender soluble)

G00 Z 0.1 (Mover herramienta a plano R)

Inicia secuencia de maquinado....

Formato de fin de herramienta :

Termina secuencia de maquinado...

G00 Z 0.1 (Mover herramienta a plano R o a Z inicial si el maquinado es vertical)

G91 G28 Y0 Z 0 G49 M09 M19 (En modo incremental, mandar husillo a posición de referencia en Z - en el retorno cancelar compensación de altura - .Si se instaló algún comando de tipo modal cancelarlo, apagar soluble , orientar husillo para cambio de herramienta)

T02 (Rotar la siguiente herramienta en el magazine, lista para cambio automático)
M01 (Comando de paro opcional)

Formato de fin de programa:

G91 G28 G49 Y0 Z 0 M19 (Mandar a posición de referencia en YZ , orientar herramienta en el husillo, cancelar compensación de longitud)
T01 M06 (Instalar la primera herramienta en el husillo)
G28 X 0 (Mandar a posición de referencia el husillo en el eje X)
M30 (Fin de programa)

Formato para torno CNC.

Antes de explicar las técnicas de formateo para torno, debemos aclarar que existen diferencias más marcadas para los tornos en el código G de un fabricante a otro, que para los centros de maquinado. Aunque este problema afecta a lo que es la técnica de formateo, solamente lo hace en dos áreas que nos conciernen. Una tiene que ver con el uso de comandos para asignar el cero y la otra en cómo se asigna el modo incremental o absoluto.

Comandos usados para asignar el cero pieza.

El comando más usado para asignar el cero pieza es G50 ; este comando sirve también para indicar a la máquina las máximas RPM y será también el que nosotros utilicemos en nuestro formato. Pero quizás tu torno acepte G92 en lugar de G50 ; esto es debido principalmente a que muchos fabricantes norteamericanos de maquinaria CNC, tienden a usar el mismo código tanto en sus centros de maquinado como en sus tornos; no hay problema, tanto el G50, como el G92 funcionan de la misma manera y tienen el mismo propósito.

Hablemos un poco más del comando G50, ya dijimos que sirve también para limitar las RPM del torno. ¿ Cómo funciona esto ? , al indicar al control por medio de este comando, el cero pieza , en la misma línea de programa indicamos con una S las máximas RPM que puede utilizar el control en el maquinado. Este comando es útil sobre todo cuando se maquina una pieza cuya geometria o peso no permite una adecuada sujeción con las mordazas del *chuck* . Lo que se hace es montar la pieza , alinearla y posteriormente hacerla rotar incrementando las RPM en el torno. Cuando veamos que la pieza comienza a vibrar, observamos las máximas RPM en nuestra pantalla y reducimos dicho valor en un 10 % por seguridad ; dicho valor serán nuestras máximas RPM permitidas.

Este comando también se relaciona con algo llamado CSS (constant surface speed) . Al programar una velocidad de corte en torno en operaciones donde varia el diámetro de la pieza conforme la herramienta se desplaza a través de ella - como un careado, un cilindrado, etc.. - , el control compensa esta variación en el radio, aumentando o disminuyendo las RPM , con objeto de mantener una velocidad de corte uniforme, y por tanto , un buen terminado en la pieza.
Si hemos indicado un límite a las RPM con G50 , el control ajustará la velocidad de corte programada , de modo que sea compatible con las RPM máximas permitidas.

El modo CSS está comandado por G96 , dicho código es modal en cualquier torno . Cuando la herramienta no se desplaza a través del contorno de la pieza , sino que maquina por el centro de la misma, por ejemplo un barrenado, instalamos el modo G97; la diferencia entre estos dos códigos es que en el comando G96 la velocidad de corte se da en unidades sobre minuto y en G97 es en RPM.

Es importante recordar esto, sobre todo en los formatos de inicio y fin de herramienta. Siempre que requieras instalar un comando G97 , por ejemplo, para barrenar por el centro deberás incluir como parte de tu formato de fin de herramienta un comando G96.

Comandos para asignar modo incremental y absoluto en un torno.
La otra diferencia en el formateo está relacionada en cómo se asignan los comandos para indicar modo incremental o absoluto. Otra vez , el formateo que presentaremos aquí es el modo más usado por los controles manufacturados en E.U . Con este método no se requiere un código G para asignar modo incremental o absoluto. El control reconoce por medio de letras, si el movimiento va a ser en modo absoluto o incremental. Si usamos X o Z , el movimiento es en modo absoluto. Si usamos U o W el movimiento es incremental.
Hacemos notar que U especifica modo incremental en el eje X y W en el eje Z y que en algunos controles es posible combinar modo incremental y absoluto .
También es importante que recuerdes , que por lo general , tanto U como X representan medidas en relación al diámetro .
El otro método usado por algunos controles, mantiene consistencia con los centros de maquinado. Un comando G90 nos indicará modo absoluto y a su vez G91 nos indicará modo incremental .

Formato de inicio de programa:

O0001 (Número de programa)
G28 U 0 W 0 (Comando que asegura que la máquina se encuentre en su punto de referencia)

G50 X Referencia Z Referencia S 3000 (Asignar cero pieza , limitar rotación del husillo a 3000 RPM)

Formato de inicio de herramienta:

G00 T0101 M41 (Indexar primera herramienta , instalar *offset* de herramienta , seleccionar rango de husillo)
G96 S350 M03 (Instalar modo de CSS, comandar velocidad de husillo en unidades sobre minuto, sentido CW)
G00 X xinicial Z zinicial M08 (mover herramienta a punto próximo antes de iniciar maquinado, prender soluble en el trayecto)
G00 .... (Inicia secuencia de maquinado)

Formato de fin de herramienta:

G01 ... (Termina secuencia de maquinado, herramienta en punto libre)
G00 X referencia Z referencia M09 T0100 (Herramienta viaja a posición de indexado, se apaga el soluble, en el trayecto cancelamos *offset*)
M01 (Parada opcional)

Formato de fin de programa:

G00 X referencia Z referencia T0200 (Regreso a posición de referencia , se cancela *offset* de la última herramienta)
M30 (Fin de programa)

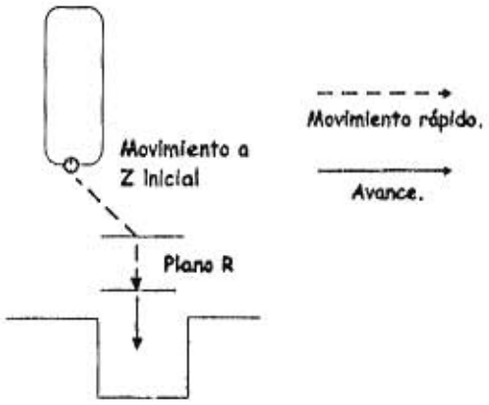
Conclusiones acerca del formateo:
A través de la explicación anterior, espero te habrás dado cuenta de varias cosas importantes:
La lógica que sigue un código de formato, la importancia de un código de formato y las ventajas que reporta a nuestro estilo de programación.
Esta técnica será de uso imprescindible tanto en tus programas CNC como en las macros que desarrolles. Más nosotros raramente incluiremos en las macros de este texto código de formateo, puesto que es más importante que comprendas la mecánica de el código que genera el maquinado para cada una de ellas; pudiendo confundirte el código que introduzcamos para formateo.

Uso de parámetros dentro de tu macro.

¿ Qué es un parámetro ? . Los parámetros son valores prefijados en el control que pueden ser modificados para controlar el desempeño del control ya sea en el maquinado, en las condiciones de corte , en el mecanismo mismo de la máquina, etc..

Los parámetros sirven como una interfase directa para modificar características específicas del control . Un control moderno posee numerosos parámetros. Entre más sofisticado sea el control o la máquina, ofrece mayor número de ajustes que pueden realizarse al mismo, así es común encontrar parámetros de maquinado; de máquina o mecanismo, de husillo, de servo, etc..

Los parámetros más interesantes para nosotros se refieren a los de mecanizado. Básicamente estos parámetros fueron creados para hacer más flexible la ruta de la herramienta y el maquinado en general en controladores de tipo conversacional. En estos controladores, el usuario programa mediante imágenes y formas en lugar de ingresar código EIA –ISO. Los parámetros de mecanizado funcionan para poder personalizar estos controles conversacionales . Por ejemplo, hemos hablado ya de un plano R ; este plano define la mínima distancia en la cual se da un cambio de movimiento rápido a movimiento con avance en Z como se observa en la siguiente figura.



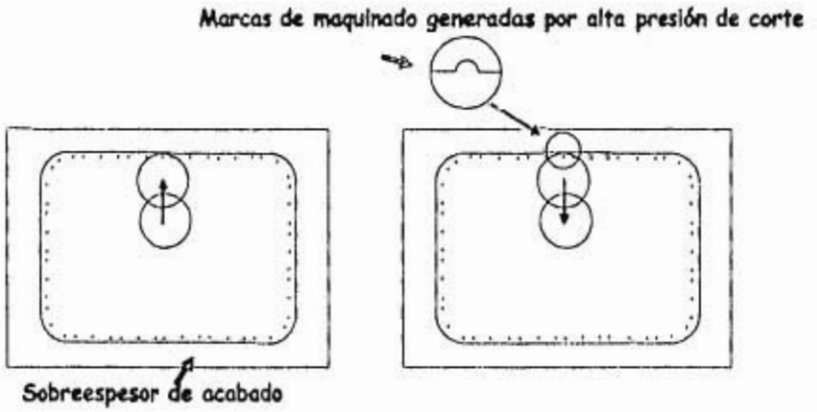
El plano R se conoce como parámetro de claro axial y le es asignado por el fabricante por lo general un valor de 0.100". Supongamos que esta distancia me parece excesiva para una cierta pieza que necesito maquinar .Entonces cambio el valor que considero correcto para este parámetro en la pantalla de parámetros de mi control así siempre que el control conversacional utilice este plano R usará el nuevo valor programado.

Nosotros retomaremos esta idea en los macro programas que diseñemos . Al definir nuestro patrón de mecanizado procuraremos definir ciertas condiciones en él que permitan ser modificadas más tarde por el operador , de tal manera que el código que generemos presente alta flexibilidad y pueda "absorber" todas las situaciones que se presenten. Además, siempre será mejor desarrollar un solo código para una sola aplicación, que varios para cubrir variaciones del mismo. En todos los ejemplos procuraremos advertir que puede ser tomado como parámetro, esto es con la finalidad, de que posteriormente sea fácil identificar en tus macros tus propios parámetros o crearlos de acuerdo a tu conveniencia o necesidad.

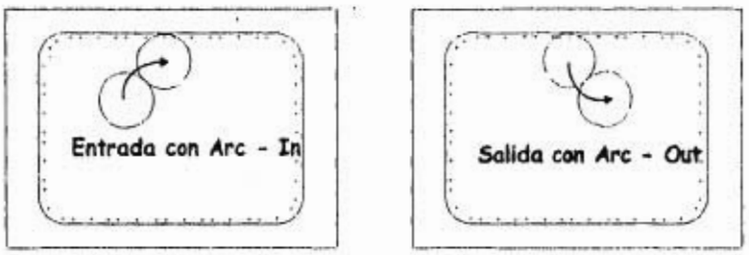
Usando Arc - In y Arc Out:

En los centros de maquinado es común programar movimientos circulares con cortadores rectos generalmente usados para generar un contorno en la pieza o fresar un agujero. La técnica que puedes aprender aquí, evitará que al maquinar la pieza se dejen marcas del mecanizado de la herramienta sobre ella, marcas que probablemente hagan que nuestra pieza quede fuera de especificación.Estas marcas son consecuencia de la presión que ejerce la pieza sobre la herramienta, tendiéndola a alejar de la ruta de maquinado. Este efecto es conocido como "tool pressure " .

La siguiente figura muestra lo que generalmente pasa cuando la trayectoria de aproximación del cortador es generada de manera perpendicular a la superficie a maquinar. La herramienta contacta de manera violenta con la superficie que va a ser mecanizada, alcanzando casi instantáneamente toda la profundidad de corte . Debido a esta razón la herramienta debe soportar una gran presión al empezar el maquinado. Entre más profundo sea nuestro desbaste, más violento será el contacto.



Siempre que maquines formas interiores o cuando te aproximas directamente a maquinar una superficie exterior de un material con baja maquinabilidad o cuya viruta no se desprenda fácilmente encontrarás que el maquinado tiene resultados óptimos si la herramienta se acerca a la superficie formando un arco tangente a ella y sale de ella formando también un arco tangente. Esto se llama ARC - IN y ARC - OUT. La siguiente figura muestra cómo la herramienta se aproxima gradualmente a la superficie mediante la técnica descrita de Arc - In y Arc – Out, aunque la curvatura de entrada de la herramienta es exagerada, sirve para mostrar la clase de movimiento que se pretende con esta técnica. Observa cómo la herramienta entra y sale suavemente de la superficie de maquinado.

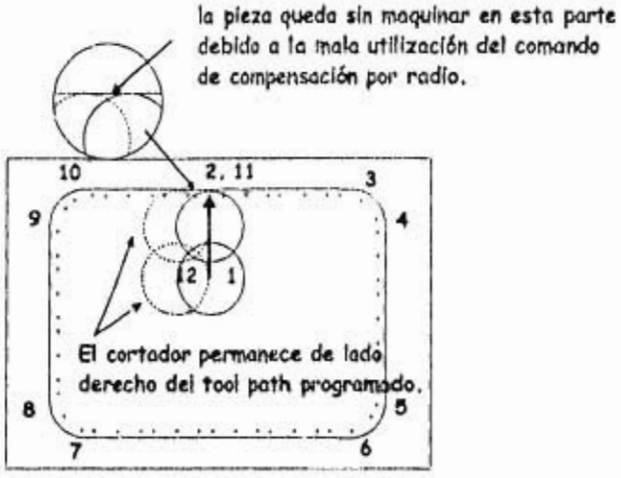


Esta técnica tiene aplicación solamente cuando la herramienta aproxima directamente a la superficie que va a ser maquinada , ya sea para desbaste o solamente para acabado. Aunque esta condición existe cuando maquinamos un contorno exterior, generalmente se aplica al maquinar en el interior.

Habrá ocasiones en que no será posible aplicar esta técnica o no sea práctico hacerlo, sea porque la relación entre el diámetro del radio de premaquinado y nuestro cortador no nos permita programar un arco de entrada, porque no interesa el acabado del interior, por las características del material o porque el barreno lleve otro proceso posterior. Sin embargo, en todos los ejemplos de este texto, procuramos programar usando esta técnica.

Los movimientos de aproximación de Arc - in y Arc - Out son necesarios por dos razones. La primera es fácil de entender y visualizar. *Siempre que sea posible* es conveniente el dispersar la presión de la herramienta producida durante el maquinado ; esto permite a la herramienta el alcanzar lentamente la presión máxima conforme es alcanzada la profundidad total de corte, permitiéndonos que las áreas de aproximación y retracción sean maquinadas de manera suave y aceptable.

La segunda razón para usar las técnicas de Arc - In y Arc - Out es más difícil de visualizar . De hecho se aplica solamente si se está usando compensación de radio de la herramienta. Si has usado ya esta técnica, sabes que hay numerosas reglas que la gobiernan y que no deben ser rotas. Sabrás que una vez que instalamos compensación de radio, el cortador se mantendrá en el lado derecho o izquierdo de los movimientos rectos o circulares que se programen a continuación, dependiendo si el movimiento comandado es G41 o G42 . En algunos controles, esto puede presentar problemas al generar los movimientos de aproximación y retracción, en especial , si no has comprendido cómo funciona la compensación de radio específicamente para tu control.



Por ejemplo, la figura anterior nos muestra los movimientos de un programa que usa compensación por radio; en este caso programamos compensación en el punto 2 (con G42) .

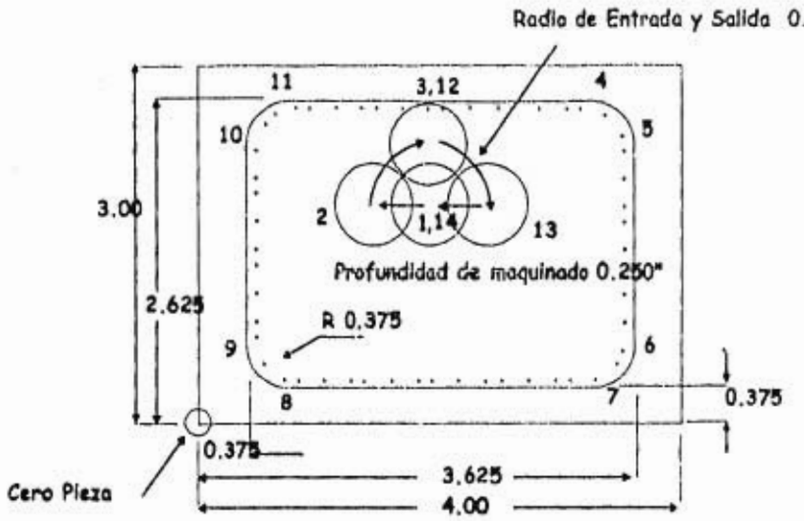
A partir de este punto el control mantendrá al cortador en el lado derecho de todos los movimientos lineales o circulares que a continuación se programen . Al final de estos movimientos, cuando el cortador llega al punto 11 marcado en el dibujo, el controlador mira hacia adelante para ver el siguiente movimiento , en este caso es al punto 12 , (aún bajo la influencia de la compensación por radio) . Como el controlador sigue manteniendo la herramienta en el lado derecho del movimiento programado, (en este caso una línea), habrá una pequeña porción de la superficie que quedará sin maquinar.

Esto implica que el área de retracción de la herramienta, estará incorrectamente maquinada; habrá una porción sin remover. Este problema surge debido a que el controlador sigue manteniendo la herramienta del lado derecho del contorno programado. Aunque en la mayoría de los controladores hay maneras de solucionar este problema - basadas en cómo se cancela el comando de compensación de radio - , es más común que al encontrarse con este problema, el

operador remaquine y muchas veces agrave el problema en lugar de solucionarlo; ya que al repasar con la herramienta esta superficie, está bajo la influencia de poca o ninguna presión , pudiendo perder dimensión la pieza y como consecuencia quedar inservible.

Si se usan las técnicas de entrada y salida por Arc - In y Arc - Out, no hay maquinado incorrecto, puesto que, en el momento de aproximación o retracción de la herramienta, el comando de compensación será instalado o cancelado mientras el cortador no está en contacto con la superficie de maquinado.

El siguiente ejemplo que sigue nos muestra - para propósitos de aclaración solamente - como usar Arc - In y Arc - Out , al usar compensación por radio en nuestro control.



(Programa de ejemplo en EIA - ISO para demostrar el uso de Arc - In y Arc - Out, con comando de compensación de radio)
O0010 (Número de programa)
G28 G91 Z0 (En incremental, mover husillo a punto de referencia en z)
T01 M06 (Cambio de herramienta No. 1)
G54 G90 S350 M03 (Modo absoluto, cero p'eza, 350 RPM, en sentido horario)
G00 X2. Y 1.875 (Rápido al punto 1)
G43 H01 Z0.1 (Rápido a plano R, instalando compensación de altura para la herramienta 1).
G01 Z -0.25 F 3.5 (Bajando en Z a profundidad de corte)

G42 D 01 X 1.25 (Instalando compensación derecha de radio; movimiento al punto 2)
G02 X2. Y 2.625 R 0.75 (Arc - In al punto 3)
G01 X3.25 (Movimiento recto al punto 4)
G02 X3.625 Y2.25 R 0.375 (Movimiento circular al punto 5)
G01 Y 0.75 (Movimiento recto al punto 6)
G02 X 3.25 Y 0.375 R 0.375 (Movimiento circular al punto 7)
G01 X 0.75 (Movimiento recto al punto 8)
G02 X 0.375 Y 0.75 R 0.375 (Movimiento circular al punto 9)
G01 Y 2.25 (Movimiento recto al punto 10)
G02 X 0.75 Y 2.625 R 0.375 (Movimiento circular al punto 11)
G01 X2.0 (Movimiento recto al punto 12)
G02 X 2.75 Y 1.875 R 0.75 (Arc - Out al punto 13)
G40 G01 X2. (Cancelando compensación, movimiento recto al punto 14)
G00 Z 1.0 (Rápido arriba de la superficie de trabajo)
G91 G28 Z0 (Retorno a punto de referencia Z)
G28 X 0 Y 0 (Retorno a punto de referencia X y Y)
M30 (Fin de programa)

En los siguientes capítulos para nuestros ejemplos de macro programación, procuraremos utilizar lo menos posible los comandos de compensación por radio (G41y G42), debido a nuestra intención aclarada previamente de desarrollar macro programas que puedan adaptarse a cualquier control - la compensación por radio no sigue las mismas reglas de aplicación, sobre todo en controles antiguos en donde hay que programar vectores directores que sirven para orientar a la herramienta en su camino - . Sin embargo siempre que se requiera aplicaremos la técnica de Arc - In y Arc - Out , programando en torno al centro de la herramienta.

CAPITULO 3

Diseñando macroprogramas con mecanizado relativo al centro.

Revisadas en el capítulo anterior algunas de las técnicas que usaremos para desarrollar nuestros macro programas, es hora de ponerlas en práctica.

Este capítulo está dedicado por completo al diseño de macroprogramación para mecanizado con relación a un centro. Esto es, las coordenadas de nuestros macro programas se referirán exclusivamente al centro geométrico de la figura a maquinar.

Desarrollaremos varios ejemplos de rectificado de un barreno utilizando cortador recto. Iremos desde la técnica de rectificado más sencilla hasta una más avanzada para realizar este tipo de maquinado.

Asimismo se diseñará un ciclo de taladrado. Conjuntaremos las macros de fresado para generar una aplicación más poderosa y desarrollaremos nuestros propios patrones de maquinado.

Es necesario decir como procederemos para lograrlo.

Para cada macro programa diseñado, daremos una explicación general del patrón del mecanizado de esa geometría en particular y cuando se considere, información técnica al respecto. Dibujaremos en nuestra geometría dicho patrón de mecanizado para mostrar la secuencia propuesta a seguir (sin ecuaciones) . Posteriormente deduciremos matemáticamente esa trayectoria auxiliándonos básicamente con gráficas, en las cuales se traza de manera independiente cada secuencia del maquinado. Indicaremos los parámetros de maquinado, necesarios, para finalmente resumir nuestro algoritmo en ecuaciones y después escribirlr el código en PASCAL. Al final de cada macro viene una secuencia de maquinado a manera de ejemplo, desarrollada en este lenguaje, utilizando el algoritmo explicado.

Este método , largo si se quiere, me ha demostrado ser eficaz para desarrollar macro programación . Es esencial primero entender cómo se comporta la herramienta al realizar el mecanizado, las diversas situaciones que pueden surgir en el curso del mismo y la mecánica de corte adecuada , antes de pensar siquiera en empezar la programación .En pocas palabras primero debemos pensar y después programar.

Sería ideal que intentaras replantear cada una de las macros ofrecidas en este texto y llegaras a lograr mejores soluciones a las que yo obtuve . Disfruta y recuerda, la práctica hace al maestro.

Macroprogramación de un fresado circular (Circular Mill Sencillo):

Una situación común que nos encontramos al maquinar, es cuando tenemos que rectificar un barreno o maquinario a un cierto diámetro no estándar. Una manera rápida de lograrlo, será maquinar y dar el diámetro especificado por medio de un cortador recto. Esta técnica se conoce comúnmente como Circular Mill. Consiste básicamente en aproximar la herramienta al centro del agujero, bajar en Z hasta la profundidad de corte axial , avanzar a la profundidad de corte radial y comandar un movimiento circular que a la vez desbasta y rectifica. Posteriormente retirar la herramienta al centro del barreno y si es necesario, repetir esta mecánica hasta maquinar por completo el agujero, como puede observarse en la figura 1.

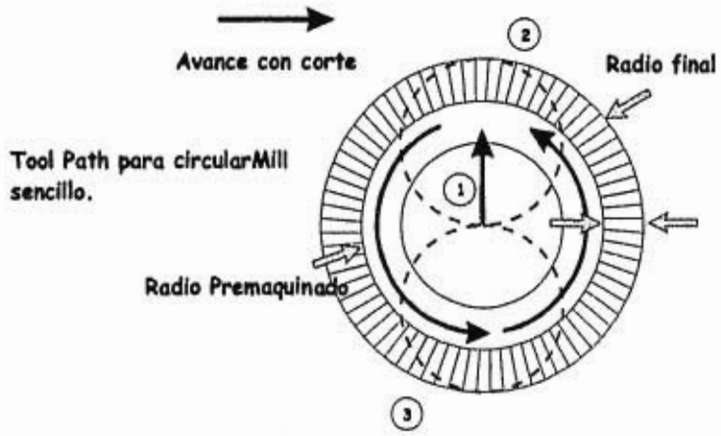


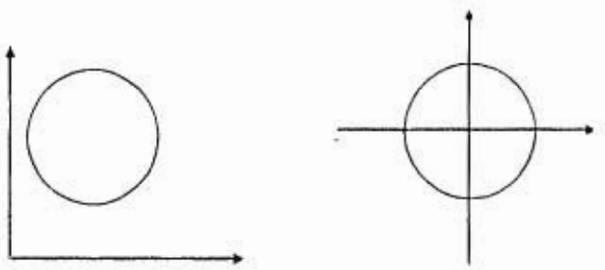
FIGURA 1. SECUENCIA DE MECANIZADO PARA CIRCULAR MILL

Esta sencilla técnica, no es la óptima; debido a que si el material es duro, tiende a generar una altísima presión en el corte radial, por las razones explicadas en el capítulo anterior. Sin embargo, muchas veces será necesario utilizarla, sea porque carecemos de variedad en los diámetros de los cortadores rectos a nuestra disposición; porque el diámetro final a maquinar con respecto al diámetro de premaquinado es solamente un poco mayor, e incluso, porque el acabado del barreno no es crítico. A lo largo de este texto mostraremos técnicas mejoradas de Circular Mill.

Nuestra primera macro a desarrollar será pues el Circular Mill sencillo. Con la figura 1, te podrás dar cuenta, que hay tres coordenadas que definen el patrón de mecanizado. Hemos elegido de manera arbitraria, desarrollar código para rectificar en sentido antihorario únicamente. Una macro completa de Circullar Mill, deberá ser capaz de desarrollar maquinado tanto en sentido horario como antihorario. Se deja al lector la tarea de completar nuestra macro.

La figura 1 también nos muestra las variables implícitas para resolver la macro.
Resumamos pues nuestra mecánica de mecanizado, (que más adelante traduciremos a ecuaciones) :

1- Primero aproximamos la herramienta en el plano XY al centro del agujero a mecanizar con movimiento rápido (punto1). Bajamos la herramienta en Z hasta Z inicial con movimiento rápido programando compensación de altura.
2- Bajamos en Z hasta al plano R da igual modo, con movimiento rápido.
3- Con avance hasta la profundidad de corte axial.
4- Igualmente con avance nos desplazamos hasta la posición 2 en movimiento recto
5- En sentido antihorario desplazamos al cortador hasta la posición 3.
6- En sentido antihorario nos desplazamos al punto 2
7- Regresamos al punto 1, si es necesario bajamos otra vez en corte en Z, y reiniciamos el maquinado o proseguimos maquinando en dirección radial hasta obtener el radio final requerido.



Barreno cuyo diámetro pertenece al mismo cuadrante

Barreno cuyo diámetro no pertenece al mismo cuadrante.

FIGURA 2

Planteada nuestra ruta de mecanizado, posiblemente te surja una inquietud al revisarlo: el por qué dividir el movimiento circular del cortador y establecer un punto intermedio (3) y no comenzar el movimiento en el punto 2 y terminarlo en este punto directamente. Esto se debe al hecho de que algunos controladores, por lo general muy antiguos, no permiten programar movimiento circular completo cuando existe cambio de signo con respecto al punto de inicio y punto final de dos coordenadas (X o Y) a lo largo de un cuadrante. Es decir , si nuestro controlador pertenece a esta categoría y elegimos colocar nuestro barreno en una posición cualquiera de modo tal que su diámetro pertenezca todo al mismo cuadrante, no tendremos problema alguno (figura 2 lado izquierdo) , pero si se encuentra ubicado de modo tal que lo atraviese una línea de cuadrante

(figura 2 lado derecho), el controlador marcará error. Este es un punto que debe tenerse en cuenta a la hora de programar macros que supuestamente funcionen para cualquier tipo de controlador.

Deducción matemática del *tool path.*

Procedamos a definir las variables necesarias para poder plantear nuestras ecuaciones:

Sea Xini la coordenada X del centro del círculo. Yini la coordenada Y del centro del círculo. Radprem el radio de premaquinado incial. Radpremfin el radio de premaquinado final . Dtx la profundidad de corte radial. Rtool el radio de la herramienta, J el vector director del comando circular usado.

Nuestra coordenada inicial será pues Xini,Yini.

Para deducir la coordenada 2 observemos la figura 3:

C será la distancia desde el extremo del radio de la herramienta en la posición 1, hasta el centro de la herramienta en la posición 2 .

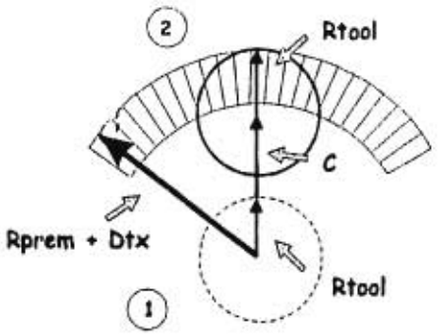


FIGURA 3

Es evidente que podemos plantear las siguientes ecuaciones:

Rprem +Dtx = 2Rtool + C

De donde: Rprem – 2Rtool +Dtx = C

Lo que avanza la herramienta en Y de la posición 1 a la posición 2 será el radio de la herramienta mas C, por lo tanto, ya tenemos las coordenadas de la posición 2.

Y desplazada = Yini +Rtool +C = Yini +Rtool +Rprem –2Rtool +Dtx = Yini +Rprem +Dtx -Rtool

Coordenada 2 G01 a Xini, Yini+ Rprem +Dtx –Rtool

Planteamos nuestra coordenada número 3:

Nuestra geometría es simétrica (estamos maquinando un círculo) por lo que la coordenada número 3 será la misma que la coordenada 2 , pero con su coordenada en Y negativa (ver figura

4) . Como estamos describiendo un movimiento radial (sentido antihorario) será necesario escribir los vectores directores I , J. En este caso unicamente existe vector J, el cual como sabemos establece la distancia desde el inicio del arco que estamos describiendo al centro de éste; esta distancia ya la hemos calculado , es al radio de la herramienta más C.

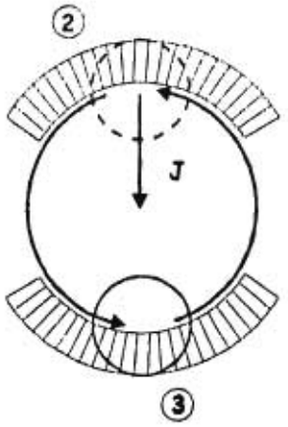


FIGURA 4

Ahora podemos escribir la coordenada 3.

Coordenada 3: G03 a Xini , Yini - (Radprem +Dtx-Rtool)

J = - (Radprem + Dtx – Rtool)

La siguiente coordenada (4) , establece el movimiento del cortador al punto de inicio 2 . Las ecuaciones correspondientes a este punto son:

Coordenada 4 : G03 a

Xini, Yini +(Radprem + Dtx – Rtool)

J = Radprem + Dtx – Rtool (el vector J es positivo porque su dirección es hacia el eje Y +)

Finalmente regresamos el cortador al punto inicial (el centro del barreno), por lo que las coordenadas finales de la macro serán:

Coordenada 5: G01 a Xini, Yini

Hemos definido ya el trazo matemático de la herramienta para empezar a programar nuestra macro, únicamente falta por definir un parámetro que utilizaremos para ubicar nuestro plano R ; al cual llamaremos D41 y le asignaremos un valor de 0.100".

Antes de proseguir es importante que notes cómo se buscó el relacionar las variables importantes del maquinado con las ecuaciones que planteamos en la macro. El radio de premaquinado, la profundidad de corte y el radio de la herramienta, son las variables que realmente importan para definir la geometría del mecanizado y en base a ellos se desarrolla todo el planteamiento matemático.

El algoritmo para Circular Mill se puede resumir como sigue:

1. Con movimiento rápido, movemos la herramienta en XY al centro de nuestro barreno (Xini, Yini), avanzando en Z hasta Z inicial, instalando compensación de altura .
2. Movemos la herramienta en Z hasta el plano R (Z = 0.100").
3. Programando avance, movemos el cortador en Z hasta profundidad de corte axial.
4. Desplazamos el cortador con G01 a Xini, Yini+ Rprem +Dtx –Rtool.
5. Interpolación: G03 a Xini , Yini - (Radprem +Dtx-Rtool) J = - (Radprem + Dtx – Rtool).
6. Interpolación (retorno a punto de corte): G03 a Xini, Yini +(Radprem + Dtx – Rtool) J = Radprem + Dtx – Rtool .
7. Retorno con G01 a Xini, Yini .De ser necesario bajamos otra vez en corte en Z y reiniciamos el maquinado o proseguimos maquinando en dirección radial hasta obtener el diámetro final requerido.

Programa en PASCAL que genera el código de maquinado:

Nota:

Por lo general los programas que escribamos en PASCAL, no solamente generaran código para el algoritmo deducido. Serán capaces de considerar también un residuo tanto en la profundidad de corte radial como en el axial , así como permitir ingresar (cuando sea pertinente para la aplicación) espesor de acabado tanto en dirección radial como en dirección de corte en Z.

```
Const
D4l=0.1; { Parámetro que define el plano R inicial }
S4=1E-5; { Parámetro que permite corregir punto flotante  }
Zinicial=1.0;

procedure circumillA(xini,yini,F,Ff,Pcr,Radprem,Radpremfi,Rtool,Zllegada,
Pcztotal,Pcz,Finz,FinR:real;retorno:boolean;var arctrampa:text);
var
Dtx,Radpreml,residuor,residuoz,Dtz,Dtzl:real;
Banderanegra,Banderapirata:boolean;
nr,nz:shortint;
begin
Banderanegra:=true;
```

```
assign(arctrampa,'KREATOR.fil');
rewrite(arctrampa);
writeln(arctrampa,'(CIRCULARMILL)');
writeln(arctrampa,'G00 X',xini:6:4,'Y',yini:6:4,'Z',(D41+Zllegada):6:4);
close(arctrampa);
nz:=trunc((Pcztotal-Finz)/pcz);
residuoz:=puntoflotante(Pcz*(frac((Pcztotal-Finz)/Pcz)));
if nz<=0 then
 Dtzl:=residuoz
else
 Dtzl:=Pcz;
Dtz:=Dtzl;
 while (Dtz<=(Pcztotal-Finz)) and BanderaNegra do
  begin
   append(arctrampa);
   writeln(arctrampa,'G01 Z',(Zllegada-Dtz):6:4,'f',F:4:4);
   nz:=nz-1;
   if (nz=0) and (residuoz<>0) then
     Dtz:=Dtz+residuoz
   else if (nz=0) and (residuoz=0) then
    begin
     if finz=0 then
      BanderaNegra:=false
     else
      Dtz:=Dtz+finz;
    end
  else if (nz<0) and (residuoz<>0) then
   begin
    if (finz=0) then
     BanderaNegra:=false
   else
     Dtz:=Dtz+finz
   end
  else if nz>0 then
   Dtz:=Dtz+Dtzl;
 radpreml:=radprem;
 nr:=trunc((RadPremfi-Finr)-radprem)/Pcr);
 residuor:=puntoflotante((frac((RadPremfi-Finr)-radprem)/pcr))*Pcr);
 if nr<=0 then
  Dtx:=residuor
 else
 Dtx:=pcr;
Banderapirata:=true;
 While (Radpreml<=(Radpremfi-Finr)) and Banderapirata do
  begin
   append(arctrampa);
   writeln(arctrampa,'G01 X',xini:6:4,'Y',(yini-(Radprem+Dtx-Rtool)):6:4,'F',F:6:4);
```

```
   writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini+Radprem+Dtx-Rtool):6:4,'J',(Radprem+Dtx-
Rtool):6:4);
   writeln(arctrampa,'G03 X',Xini:6:4,'Y',(yini-(Radprem+Dtx-Rtool)):6:4,'J',-(Radprem+Dtx-
Rtool):6:4);
   writeln(arctrampa,'G01 X',Xini:6:4,'Y',(yini-(Radprem+Dtx-Rtool)+Pcr+D43):6:4);
   close(arctrampa);
   nr:=nr-1;
   if (nr=0) and (residuor <> 0) then
    Dtx:=Dtx+residuor
   else if (nr=0) and (residuor=0) then
     begin
    if finr=0 then
     Banderapirata:=false
    else
     Dtx:=Dtx+finr
     end
   else if (nr<0) and (residuor<>0) then
     begin
      if finr=0 then
       Banderapirata:=false
     else
       Dtx:=Dtx+finr
    end
 else if nr>0 then
    Dtx:=Dtx+Pcr;
   Radprem1:=Radprem1+Dtx;
  end; { final del ciclo while para fresar}
      append(arctrampa);
      writeln(arctrampa,'G00 X', xini:6:4,'Y',Yini:6:4);
      close(arctrampa);
 end; { final del ciclo while para bajar en z}
append(arctrampa);
 if retorno then
  writeln(arctrampa,'G00 Z',Zinicial:6:4)
else
 writeln(arctrampa,'G00 Z',D41:6:4);
close(arctrampa);
end
```

Ejemplo de maquinado utilizando la macro de Circular Mill.

Se pide rectificar el barreno indicado en la figura 5.

Como se nos está pidiendo 2 cortes radiales de 0.125" , pensamos que un solo corte de 0.250" en Z es demasiado para la herramienta. Programaremos entonces dos cortes en escalón también de 0.125" en Z.

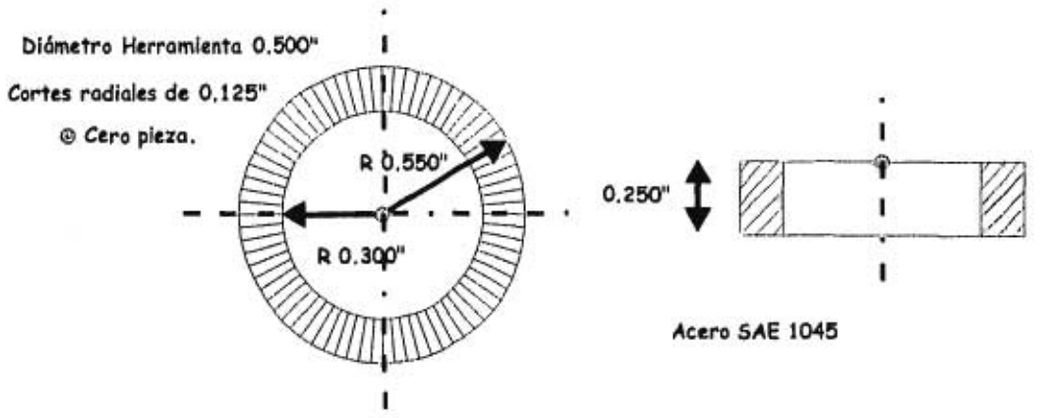


FIGURA 5

Al ingresar estos datos en nuestra macro se genera el siguiente código que maquina la pieza:

(CIRCULARMILL)

```
G00 X 0 Y 0          Z0.10000
G01                  Z-0.1250 F 0.0150
G01 X 0 Y-0.1750              F 0.0150
G03 X 0 Y 0.1750              J 0.1750
G03 X 0 Y-0.1750              J-0.1750
G01 X 0 Y-0.0490
G01 X 0 Y-0.3                 F 0.0150
G03 X 0 Y 0.3                 J 0.3
G03 X 0 Y-0.3                 J-0.3
G01 X 0 Y-0.1740
G00 X 0 Y 0
G01                  Z-0.250  F 0.0150
G01 X 0 Y-0.1750              F 0.0150
G03 X 0 Y 0.1750              J 0.1750
G03 X 0 Y-0.1750              J-0.1750
G01 X 0 Y-0.0490
G01 X 0 Y-0.3                 F 0.0150
G03 X 0 Y 0.3                 J 0.3
G03 X 0 Y-0.3                 J-0.3
G01 X 0 Y-0.1740
G00 X 0 Y 0
G00 Z 1.0
```

Generando la Macro de *Short Milling*:

Una mejor alternativa para el fresado circular la da esta técnica. Con ayuda de la figura 6 puedes seguir los pasos del maquinado. Nos aproximamos en rápido de forma diagonal una cierta distancia con el cortador, tanto como sea posible acercarse a la pared del barreno , posteriormente usamos la técnica de Arc –In para avanzar hasta la profundidad de corte radial especificada, maquinamos con Interpolación , salimos del corte con Arc – Out y posteriormente otra vez en rápido regresamos al centro del barrreno.

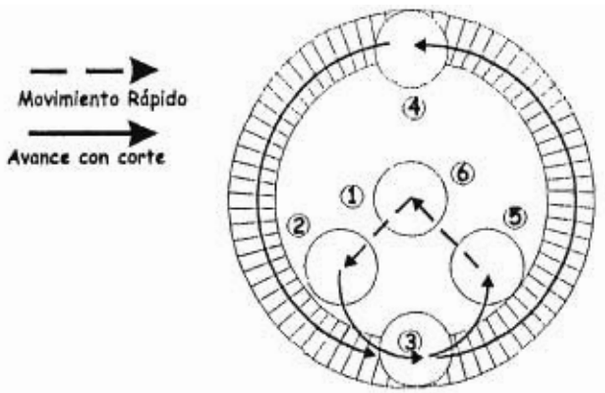
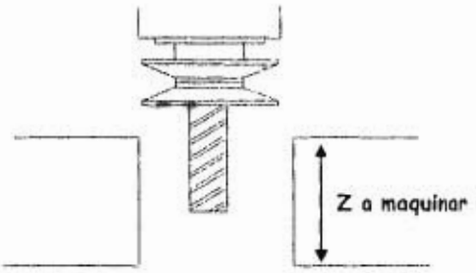


FIGURA 6

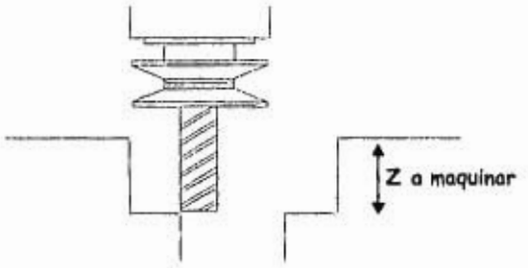
Como puedes observar en la figura , requiere una mejor relación de diámetro de premaquinado con el diámetro de cortador para que pueda ser programada .
Con ayuda de la misma figura podemos ir deduciendo la secuencia de maquinado.

1. Movemos la herramienta al centro del barreno en el plano XY; simultáneamente nos desplazamos en Z hasta Zinicial instalando compensación de altura.
2. Movemos la herramienta en Z hasta llegar al plano R.
3. Bajamos hasta profundidad de corte en Z.
4. En rápido desplazamos la herramienta hasta la posición 2.
5. Ingresamos con avance y con Arc –In a profundidad de corte radial en la posición 3.
6. Interpolamos moviendo el cortador hasta la posición 4.
7. Regresamos interpolando también, a la posición 3.
8. Salimos con Arc – Out a la posición 5.
9. Regresamos en rápido al centro del barreno , si es necesario bajamos otra vez en corte en Z y reiniciamos el maquinado o porseguimos maquinando en dirección radial hasta obtener el radio final deseado.

Esta técnica es ideal cuando se necesite maquinar una gran cantidad de barrenos. Pero debemos tener en cuenta que para aplicarla no deberá existir interferencia en Z por lo menos en la profundidad de corte axial, cuando la herramienta se coloque en rápido en la posición 2, como podemos ver en la figura 7, caso contrario la herramienta corre el riesgo de tallarse y romperse.



Agujero sin interferencia en Z para maquinar con " Short Milling "



Agujero con interferencia en Z para aplicar " Short Milling "

FIGURA 7

Deducción matemática del *tool path*:

Xini y Yini serán las coordenadas del centro del barreno. Radprem el radio de premaquinado inicial. Pcr nuestra profundidad de corte radial.

Obviamente la primera coordenada será el centro de nuestro barreno Xini, Yini.

Ahora, ¿ cúal sera nuestra coordenada en la posición 2 ? . De manera arbitraria elegimos colocar la herramienta con movimiento rápido, en :

$$x = Xini - \left(\frac{RadPrem + Pcr}{2}\right) \qquad y = Yini - \left(\frac{Radprem + Pcr}{2}\right) + Rtool$$

Decimos de manera arbitraria, puesto que no hay una deducción matemática seria del por qué se eligieron estas ecuaciones para calcular esta primera coordenada. Al estar deduciendo la posición inicial de la herramienta para esta macro, encontré que esta combinación produce buenos resultados para el posicionamiento inicial , colocando la herramienta lo más cerca posible de la

pared del barreno (observa la figura 8) . Tal vez te parezca esto un poco fuera de lugar, ya que esta primera coordenada establece la posición del c ortador con movimiento rápido dentro de nuestro barreno, siendo precisamente la posición más crítica ; pero más adelante deduciremos una ecuación que nos asegure que funcionará correctamente y sin peligro alguno.

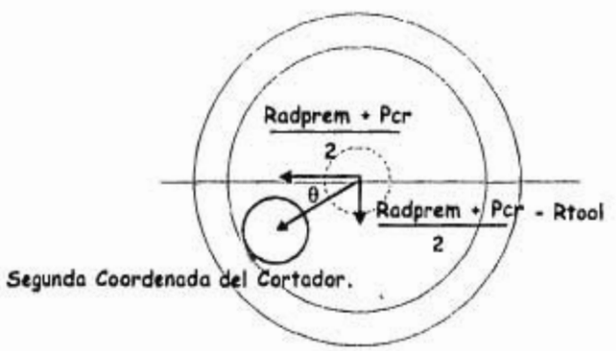
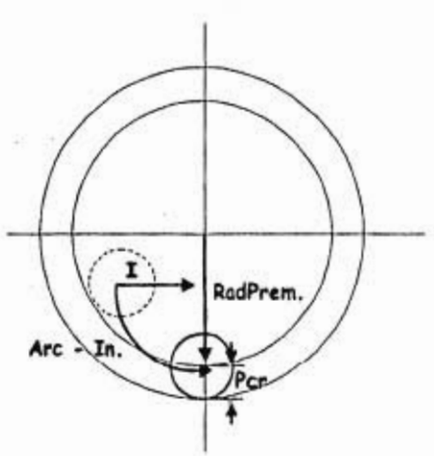


FIGURA 8

Posteriormente nos movemos con Arc -In hasta profundidad de corte radial . Como se ilustra a la izquierda de la figura 9 .

G03 a: x = Xini y = Yini – (Radprem + Pcr – Rtool)



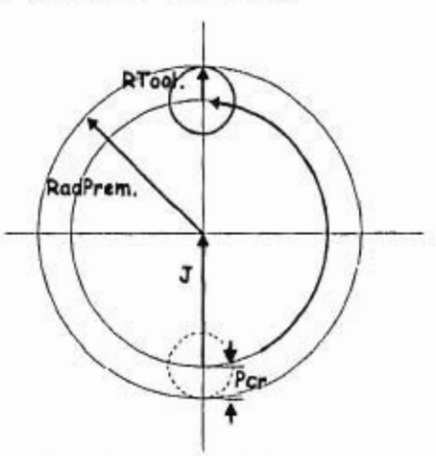


FIGURA 9

Como estamos maquinando en torno al centro de nuestro barreno nuestro vector director I estará dado por:

$$I = \frac{\text{Radprem} + \text{Pcr}}{2}$$

La cuarta coordenada será la simétrica a la anterior, pero ahora tendremos un vector J como podemos ver en la parte derecha de la figura 9 .

Dada la simetría del *tool path* , nuestro vector director en J sera igual al incremento de la coordenada Y.

Por lo que nuestra coordenada será:

G03 a: x = Xini

y = Yini + Radprem +Pcr –Rtool

J = Radprem +Pcr -Rtool

Completamos nuestro arco regresando a la posición dada en la segunda coordenada, nuestro vector J será ahora negativo .

La quinta coordenada será:

G03 a : x = Xini

y = Yini – (Radprem + Pcr – Rtool)

J = - (Radprem + Pcr – Rtool)

Saldremos del maquinado utilizando Arc – Out ; las coordenadas serán las simétricas de cuando entramos con Arc –In , como podemos ver en la figura 10:

Arc – Out G03 a :

$$x = Xini + \frac{Radprem + Pcr}{2}$$

$$y = Yini - \left[\left(\frac{Radprem + Pcr}{2}\right) - Rtool\right]$$

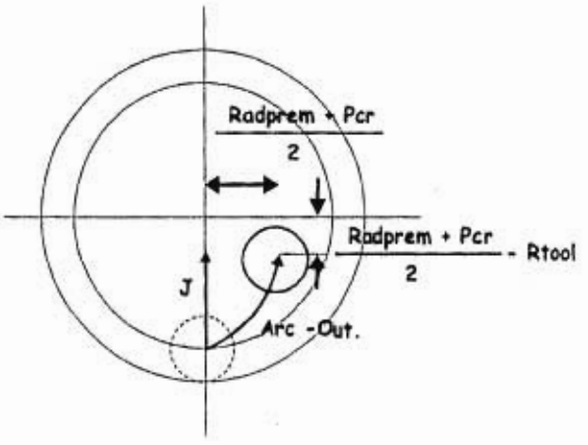


FIGURA 10

Más ahora, tenemos un vector director en J, el cual apunta al centro del círculo de radio (Radprem + Pcr) / 2 , siendo éste el valor precisamente de dicho vector J.

$$J = \frac{Radprem + Pcr}{2}$$

Nuestra última coordenada sera el retorno con movimiento rápido de la herramienta al centro del barreno (Xini, Yini).
G00 a : Xini, Yini.

Condición necesaria para inicio de ciclo:

Habrá ciertas macros para las cuales necesitemos que se cumplan ciertos requisitos que nos garantizen que funcionarán correctamente y que serán seguras. Esta macro es una de ellas. De la secuencia de maquinado te podrás dar cuenta que tenemos una coordenada crítica cuando la herramienta avanza en rápido para colocarse en la posición 2 . Necesitamos deducir una ecuación que nos asegure se cumplan, ciertas condiciones en nuestra geometría , para permitir que esta macro sea ejecutada o no.

Retomemos la primera figura de la secuencia de mecanizado. Observa que hemos indicado en ella un ángulo θ formado por las componentes de nuestra segunda coordenada.

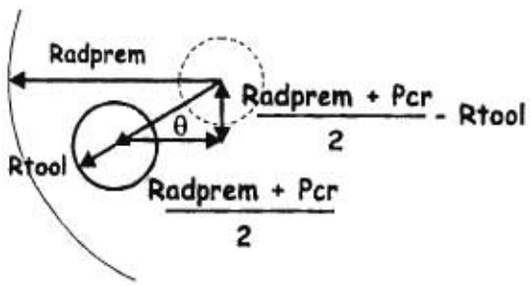


FIGURA 11

Calculemos ese ángulo θ:

$$\theta = \arctan\left[\frac{\text{Radprem} + \text{Pcr} - 2\text{Rtool}}{\text{Radprem} + \text{Pcr}}\right]$$

Ahora la condición para que podamos maquinar utilizando esta técnica de manera segura será que el radio de premaquinado inicial en el barreno sea lo suficientemente grande para permitir a la herramienta desplazarse en rápido y acomodarse en la posición 2 sin interferencia con la pared del barreno. De otro modo, la distancia definida por la hipotenusa de las dos coordenadas iniciales más el radio de la herramienta , más un cierto parámetro que llamaremos D44 sea menor o igual que el radio de premaquinado del barreno.

Traduciendo esto a ecuación:

$$\text{Radprem} \geq \frac{\text{Radprem} + \text{Pcr}}{2}\cos\theta + \text{Rtool} + \text{D44}$$

Simplificando obtenemos:

$$\text{Radprem} \geq \frac{\text{Pcr}\cos\theta + \phi\text{Tool} + \text{D44}}{(2 - \cos\theta)}$$

Esta será la condición necesaria para poder iniciar el ciclo de maquinado.
Debemos hacer notar dos cosas: La primera es que el numerador para la ecuación que calcula el ángulo θ no debe ser negativo, así esta ecuación nos restringe la posición en que se acomoda el cortador. La segunda ecuación restringe al cortador para posicionarse en rápido en la posición 2. Observa además que el parámetro D44 establece la distancia de claro radial entre la posición 2 y la pared del barreno.
Si D44 es cero, el extremo de la herramienta estará en contacto con la pared del barreno.

Resumamos pues nuestro algoritmo para *Short Milling*:

1. Probamos condición de inicio de ciclo de cumplirse, procedemos con secuencia de maquinado caso contrario termina nuestro programa.
2. En rápido , movemos la herramienta al centro del barreno en el plano XY (Xini, Yini) simultáneamente nos desplazamos en Z hasta Zinicial instalando compensación de altura.
3. Movemos la herramienta en Z hasta llegar al plano R. (Definido por parámetro)
4. Bajamos hasta profundidad de corte en Z.
5. En rápido desplazamos la herramienta hasta la coordenada definida por:

$$x = \text{Xini} - \left(\frac{\text{RadPrem} + \text{Pcr}}{2}\right) \qquad y = \text{Yini} - \left(\frac{\text{Radprem} + \text{Pcr}}{2}\right) + \text{Rtool}$$

6. Ingresamos con avance utilizando Arc –In a profundidad de corte radial en la posición definida por:

   G03 a: x = Xini

   $y = \text{Yini} - (\text{Radprem} + \text{Pcr} - \text{Rtool})$ $\qquad I = \frac{\text{Radprem} + \text{Pcr}}{2}$

7. Interpolando movemos el cortador hasta:

   G03 a: x = Xini

y = Yini + Radprem +Pcr –Rtool

J = Radprem +Pcr -Rtool

8. Completamos nuestro arco regresando a la posición 3.

G03 a : x = Xini

y = Yini – (Radprem + Pcr – Rtool)

J = - (Radprem + Pcr – Rtool)

9. Salimos con Arc – Out a la siguiente coordenada:

G03 a : $x = Xini + \frac{Radprem + Pcr}{2}$ $y = Yini - \left[\left(\frac{Radprem + Pcr}{2}\right) - Rtool\right]$

$J = \frac{Radprem + Pcr}{2}$

10. Regresamos en rápido al centro del barreno , si es necesario bajamos otra vez en corte en Z y reiniciamos el maquinado o porseguimos maquinando en dirección radial hasta obtener el radio final deseado.

Programa en Pascal para la macro de *Short Milling*.

```
const
D41= 0.1;{valor para plano R en z}
S4= 1E-5;
D44=0.1;{valor de clearancia para iniciar shortmill}
Zinicial=1.0;

procedureshortmill(Xini,Yini,Radprem,PCR,F,Facabado,PcDt,Pcztotal,Finz,
Pcz,Rtool,Zllegada:real;retorno:shortint; var Arctrampa:text);
var
Dtz,Dtx,residuox,residuoz,Radpremt,FS:real;
nz,nr:integer;
Banderapirata,BanderAcabadoz,BanderaNegra:boolean;
begin
assign(arctrampa,'Short.Fil');
{append(arctrampa);}
rewrite(arctrampa);
writeln(arctrampa,'G01 Z',(D41+Zllegada):6:4,'F',F:6:4);
close(arctrampa);
nz:=trunc((Pcztotal-Finz)/Pcz);
residuoz:=puntoflotante(pcz*frac((pcztotal-Finz)/Pcz));
Banderapirata:=true;
Banderacabadoz:=false;
If (nz<=0) then
Dtz:=residuoz
else
Dtz:=Pcz;
While (Dtz<=Pcztotal) and BanderaPirata do
 begin
 append(arctrampa);
 if not(Banderacabadoz) then
 writeln(arctrampa,'G01 Z',(Zllegada-Dtz):6:4,'F',F:6:4)
 else
 writeln(arctrampa,'G01 Z',(Zllegada-Dtz):6:4,'F',Facabado:6:4);
 close(arctrampa);
```

```
nr:=trunc((PcDt/2)-radprem)/Pcr);
residuox:=puntoflotante(Pcr*frac(((PcDt/2)-radprem)/pcr)));
if (nr<=0) then
Dtx:=residuox
else
Dtx:=pcr;
Radpremt:=Radprem;
BanderaNegra:=true;
while (Radpremt<=(PcDt)/2) and BanderaNegra do
 begin
  If BanderAcabadoz then
  Fs:=Facabado
  else
  Fs:=F;
  append(arctrampa);
  writeln(arctrampa,'G00 X',(Xini-(Radpremt+Dtx)/2):6:4,'Y',Yini-(((Radpremt+Pcr)/2)-
Rtool):6:4);
  {Inicia ARC-IN}
  writeln(arctrampa,'G03 X',Xini:6:4,'Y',(Yini-(Radpremt+Dtx-
Rtool)):6:4,'I',(Radpremt+Dtx)/2:6:4,'F',Fs:6:4);
  writeln(arctrampa,'G03 X',Xini:6:4,'Y',(Yini+(Radpremt+Dtx-Rtool)):6:4,'J',(Radpremt+Dtx-
Rtool):6:4);
  writeln(arctrampa,'G03 X',Xini:6:4,'Y',(Yini-(Radpremt+Dtx-Rtool)):6:4,'J',-
(Radpremt+Dtx-rtool):6:4);
  writeln(arctrampa,'G03 X',(Xini+(Radpremt+Dtx)/2):6:4,'Y',(Yini-((Radpremt+Dtx)/2-
rtool)):6:4,'J',(Radpremt+Dtx)/2:6:4);
  writeln(arctrampa,'G00 X',Xini:6:4,'Y',Yini:6:4);
  close(arctrampa);
 nr:=(nr-1);
 if (nr<=0) then
 begin
  if residuox=0 then
   BanderaNegra:=false
  else
   Radpremt:=Radpremt+residuox
 end
 else
  Radpremt:=Radpremt+Dtx
 end; {while}
nz:=nz-1;
if (nz=0) and (residuoz<>0) then
 Dtz:=Dtz+residuoz
else if (nz=0) and (residuoz=0) then
 begin
  if finz=0 then
   Banderapirata:=false
  else
    begin
   BanderAcabadoz:=true;
   Dtz:= Dtz+Finz;
    end
 end
else if (nz<0) and (residuoz<>0) then
begin
 if finz=0 then
  Banderapirata:=false
 else
   begin
    BanderAcabadoz:=true;
    Dtz:= Dtz+Finz;
   end
end
else if nz>0 then
 Dtz:=Dtz+Pcz;
end;
end;
```

Ejemplo de maquinado utilizando la macro de ***Short Mill.***

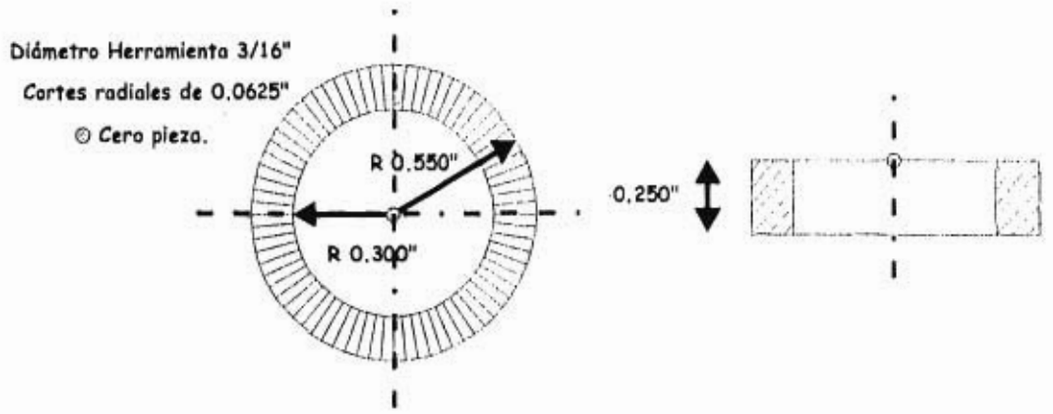


FIGURA 12

Se pide maquinar la siguiente pieza: Es la misma que la del ejemplo de Circular Mill , salvo que hemos cambiado el diámetro del cortador y la profundidad de corte radial para satisfacer nuestra condición de inicio de ciclo.

Comprobemos que esto es cierto.

$$\theta = \text{arctang}\left[\frac{0.3 + 0.0625 - 0.1875}{0.3 + 0.0625}\right] = 25.77^{\circ}$$

$$\text{Radprem} \geq \frac{0.0625\cos(25.77) + 0.1875 + 0.075}{2 - \cos(25.77)} = 0.2899$$

Se cumple nuestra condición de inicio de ciclo.

Como podrás observar, le hemos asignado un valor de 0.075" al parámetro D44.

Ingresando los datos anteriores a nuestra macro , considerando ademas realizar 2 cortes de 0.125" en Z, obtenemos el siguiente código.

```
G01 X 0 Y 0            Z 0.1      F0.2000
G01                    Z-0.1250   F0.2000
G00 X-0.1813  Y-0.0875
G03 X 0       Y-0.2688   I 0.1813 F0.2000
G03 X 0       Y 0.2688   J 0.2688
G03 X 0       Y-0.2688   J-0.2688
G03 X 0.1813  Y-0.0875   J 0.1813
G00 X 0       Y 0
G00 X-0.2125  Y-0.1188
G03 X 0       Y-0.3313   I 0.2125 F0.2000
G03 X 0       Y 0.3313   J 0.3313
G03 X 0       Y-0.3313   J-0.3313
```

```
G03 X0.2125   Y-0.1188   J 0.2125
G00 X 0       Y 0
G00 X-0.2438  Y-0.1500
G03 X 0       Y-0.3938   I 0.2438 F0.2000
G03 X 0       Y 0.3938   J 0.3938
G03 X 0       Y-0.3938   J-0.3938
G03 X0.2438   Y-0.1500   J 0.2438
G00 X 0       Y 0
G00 X-0.2750  Y-0.1813
G03 X 0       Y-0.4563   I 0.2750 F0.2000
G03 X 0       Y 0.4563   J 0.4563
G03 X 0       Y-0.4563   J-0.4563
G03 X 0.2750  Y-0.1813   J 0.2750
G00 X 0       Y 0
G01                    Z-0.2500    F0.2000
G00 X-0.1813  Y-0.0875
G03 X 0       Y-0.2688   I 0.1813 F0.2000
G03 X 0       Y 0.2688   J 0.2688
G03 X 0       Y-0.2688   J-0.2688
G03 X 0.1813  Y-0.0875   J 0.1813
G00 X 0       Y 0
G00 X-0.2125  Y-0.1188
G03 X 0       Y-0.3313   I 0.2125 F0.2000
G03 X 0       Y 0.3313   J 0.3313
G03 X 0       Y-0.3313   J-0.3313
G03 X 0.2125  Y-0.1188   J 0.2125
G00 X 0       Y 0
G00 X-0.2438  Y-0.1500
G03 X 0       Y-0.3938   I 0.2438 F0.2000
G03 X 0       Y 0.3938   J 0.3938
G03 X 0       Y-0.3938   J-0.3938
G03 X0.2438   Y-0.1500   J 0.2438
G00 X 0       Y 0
G00 X-0.2750  Y-0.1813
G03 X 0       Y-0.4563   I 0.2750  F0.2000
G03 X 0       Y 0.4563   J 0.4563
G03 X 0       Y-0.4563   J-0.4563
G03 X 0.2750  Y-0.1813   J 0.2750
G00 X 0       Y 0
```

El cual realiza el maquinado de la pieza.

Conjuntando condiciones para inicio de fresado:

Hemos diseñado ya dos macro programas para fresado , cada una con sus desventajas y ventajas propias. La técnica de Circular Mill nos permite maquinar un barreno con una relación prediámetro de maquinado diámetro del cortador pequeña . El *Short Milling* requiere que se cumpla una ecuación de restricción para que pueda ser utilizada con seguridad. Con la primer técnica podemos dejar marcas de mecanizado en la pieza, en la segunda utilizamos las técnicas de entrada por arco para evitarlas. La primera es ideal para un desbaste, la segunda parece perfecta para un acabado.

Todas estas consideraciones nos sugieren el conjuntar estas dos macros para obtener una que reúna las ventajas de ambas .

Necesitamos entonces pensar que situaciones podamos encontrar , para deducir las ecuaciones que permitan decidir cuándo aplicar una u otra.

Digamos que queremos utilizar el Circular Mill sencillo únicamente para desbaste , ¿ cómo podremos modelar esta condición ?. Se desbasta cuando nuestro espesor de sobrematerial es considerable, con respecto a la diferencia del radio final del barreno con respecto al radio de premaquinado. De otro modo:

Pcr > Radio Final – Radio premaquinado.

Ecuación que nos indicará la condición de desbaste en nuestro algoritmo.

Sigamos modelando mentalmentre nuestra macro, ahora queremos que en cuanto acabemos de desbastar o en cuanto la profundidad de corte radial sea lo bastante pequeña aplicar la técnica de *Short Milling*.

Retomando la ecuación anterior y cambiando el sentido de la desigualdad de la ecuación anterior tenemos:

Pcr < Radio Final – Radio premaquinado.

Esta ecuación nos está indicando la condición para proceder a *Short Milling*.

Más tenemos una restricción para inicio de ciclo deducida ya anteriormente, esta condición era:

$$\text{Radprem} \geq \frac{\text{Pcr}\cos\theta + \phi\,\text{Tool} + \text{D44}}{(2 - \cos\theta)} \qquad \theta = \arctan\left[\frac{\text{Radprem} + \text{Pcr} - 2\text{Rtool}}{\text{Radprem} + \text{Pcr}}\right]$$

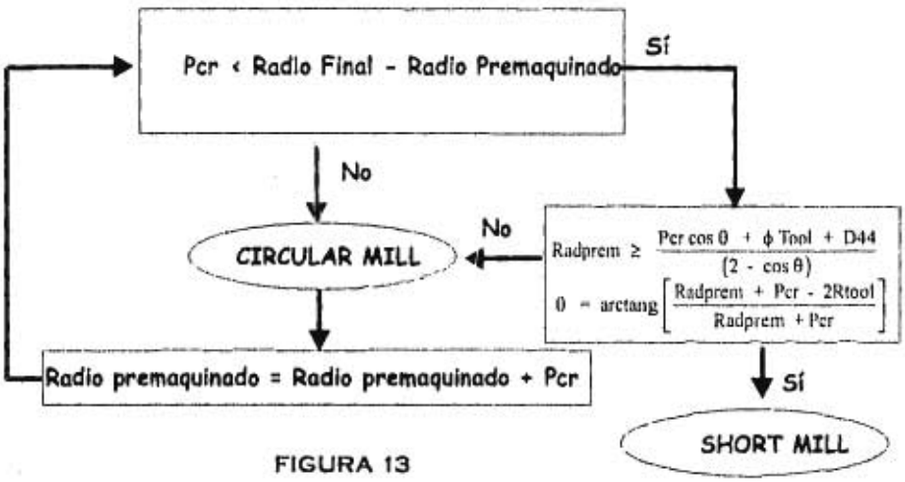


FIGURA 13

Con estas condiciones podemos plantear el algoritmo indicado en la figura 13.

En base a este algoritmo desarrollamos el siguiente código en PASCAL que combina las dos técnicas de fresado:

```
const
D41=0.1;{Parámetro de plano R}
S4=1E-5; {parámetro de punto flotante}
Zinicial=1.0;{Nuestra Z inicial}
D44=0.1;{Parámetro para inicio de Short Mill}

procedure combi(xini,yini,F,Ff,Pcr,Radprem,Radfi,Rtool,Zllegada,
Pcztotal,Pcz,Finz,FinR:real;retorno:shortint;var arctrampa:text);
var
Dtx,radvar,residuoz,Dtz,Dtzl,teta,residuor:real;
BanderaBlanca,BanderaNegra:boolean;
I,nradial,naxial:shortint;
begin
assign(arctrampa,'prue.fil');
rewrite(arctrampa);
writeln(arctrampa,'COMBINACION');
writeln(arctrampa,'G00 X',xini:6:4,'Y',yini:6:4,'G43 H01 Z',(Zllegada+Zinicial):6:4);
writeln(arctrampa,'G00 X',xini:6:4,'Y',yini:6:4,'Z',(Zllegada+D41):6:4);
close(arctrampa);
naxial:=trunc((Pcztotal-finz)/Pcz);
residuoz:=puntoflotante(Pcz*(frac((Pcztotal-finz)/Pcz)));
nradial:=trunc(((Radfi-finR)-Radprem)/Pcr);
residuor:=puntoflotante(Pcr*(frac(((Radfi-finR)-Radprem)/Pcr)));
radvar:=radprem;
BanderaBlanca:=true;
BanderaNegra:=true;
 if nradial<=0 then
  Dtx:=residuor
 else
  Dtx:=pcr;
 if naxial<=0 then
  Dtz:=residuoz
 else
  Dtz:=Pcz;
  Dtzl:=Dtz;
 While ( Dtz<=Pcztotal) do
  begin
  append(arctrampa);
  writeln(arctrampa,'G01 Z',(Zllegada-Dtz):6:4);
  For I:=1 to nradial do
   begin
    if radvar+Dtx-2*Rtool< 0 then
     BanderaBlanca:=false
    else
     begin
      teta:=arctan((radvar+Dtx-2*Rtool)/(radvar+Dtx));
      BanderaBlanca:=true;
     end;
     if BanderaBlanca then
      begin
      if (Radvar>=(Dtx*cos(teta)*2*Rtool+D44)/(2-cos(teta))) and (Dtx<(Radfi-Radvar)) then
       begin
        append(arctrampa);
        writeln(arctrampa,'G00 X',(xini-(Radvar+Dtx)*0.5):6:4,'Y',(yini-((Radvar+Dtx)/2-
Rtool)):6:4);
        writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini-(Radvar+Dtx-
Rtool)):6:4,'I',((Radvar+Dtx)/2):6:4,'F',F:6:4);
        writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini+Radvar+Dtx-Rtool):6:4,'J',(Radvar+Dtx-
Rtool):6:4);
        writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini-(Radvar+Dtx-Rtool)):6:4,'J',-
(Radvar+Dtx-Rtool):6:4);
        write(arctrampa,'G03 X',(xini+(Radvar+Dtx)*0.5):6:4);
        writeln(arctrampa,'Y',(yini-((Radvar+Dtx)*0.5-Rtool)):6:4,'J',(Radvar+Dtx)*0.5:6:4);
```

```
        writeln(arctrampa,'G00 X',xini:6:4,'Y',yini:6:4);
        close(arctrampa);
       end
      else
       begin
        append(arctrampa);
        writeln(arctrampa,'G01 X',xini:6:4,'Y',(yini-(Radvar+Dtx-Rtool)):6:4,'F',F:6:4);
        writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini+Radvar+Dtx-Rtool):6:4,'J',(Radvar+Dtx-
Rtool):6:4);
        writeln(arctrampa,'G03 X',Xini:6:4,'Y',(yini-(Radvar+Dtx-Rtool)):6:4,'J',-
(Radvar+Dtx-Rtool):6:4);
        writeln(arctrampa,'G01 X',Xini:6:4,'Y',yini:6:4);
        close(arctrampa);
       end
     end
    else
     begin
      append(arctrampa);
      writeln(arctrampa,'G01 X',xini:6:4,'Y',(yini-(Radvar+Dtx-Rtool)):6:4,'F',F:6:4);
      writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini+Radvar+Dtx-Rtool):6:4,'J',(Radvar+Dtx-
Rtool):6:4);
      writeln(arctrampa,'G03 X',Xini:6:4,'Y',(yini-(Radvar+Dtx-Rtool)):6:4,'J',-(Radvar+Dtx-
Rtool):6:4);
      writeln(arctrampa,'G01 X',Xini:6:4,'Y',yini:6:4);
      close(arctrampa);
     end;
    radvar:=radvar+Dtx;
    end;{ del for de n radial }
  if residuor <> 0 then
  begin
   if radvar+residuor-2*Rtool< 0 then
     BanderaBlanca:=false
   else
    begin
     teta:=arctan((radvar+residuor-2*Rtool)/(radvar+residuor));
     BanderaBlanca:=true;
    end;
    if BanderaBlanca then
      begin
       if (Radvar>=(residuor*cos(teta)+2*Rtool+D44)/(2-cos(teta))) and (residuor<(Radfi-
Radvar)) then
        begin
         append(arctrampa);
         writeln(arctrampa,'G00 X',(xini-(Radvar+residuor)*0.5):6:4,'Y',(yini-
((Radvar+residuor)/2-Rtool)):6:4);
         writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini-(Radvar+residuor-
Rtool)):6:4,'I',((Radvar+residuor)/2):6:6,'F',f:6:4);
         writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini+Radvar+residuor-
Rtool):6:4,'J',(Radvar+residuor-Rtool):6:4);
         writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini-(Radvar+residuor-Rtool)):6:4,'J',-
(Radvar+residuor-Rtool):6:4);
         write(arctrampa,'G03 X',(xini+(Radvar+residuor)*0.5):6:4);
         writeln(arctrampa,'Y',(yini-((Radvar+residuor)*0.5-
Rtool)):6:4,'J',(Radvar+residuor)*0.5:6:4);
         writeln(arctrampa,'G00 X',xini:6:4,'Y',yini:6:4);
         close(arctrampa);
        end
       else
        begin
         append(arctrampa);
         writeln(arctrampa,'G01 X',xini:6:4,'Y',(yini-(Radvar+residuor-
Rtool)):6:4,'F',F:6:4);
         writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini+Radvar+residuor-
Rtool):6:4,'J',(Radvar+residuor-Rtool):6:4);
         writeln(arctrampa,'G03 X',Xini:6:4,'Y',(yini-(Radvar+residuor-Rtool)):6:4,'J',-
(Radvar+residuor-Rtool):6:4);
         writeln(arctrampa,'G01 X',Xini:6:4,'Y',yini:6:4);
         close(arctrampa);
        end
      end
     else
      begin
```

```
      append(arctrampa);
      writeln(arctrampa,'G01 X',xini:6:4,'Y',(yini-(Radvar+residuor-Rtool)):6:4,'F',F:6:4);
      writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini+Radvar+residuor-
Rtool):6:4,'J',(Radvar+residuor-Rtool):6:4);
      writeln(arctrampa,'G03 X',Xini:6:4,'Y',(yini-(Radvar+residuor-Rtool)):6:4,'J',-
(Radvar+residuor-Rtool):6:4);
      writeln(arctrampa,'G01 X',Xini:6:4,'Y',yini:6:4);
      close(arctrampa);
      end;
      radvar:=radvar+residuor;
    end;
   if finr <>0 then
   begin
    if radvar+finr-2*Rtool< 0 then
      BanderaBlanca:=false
    else
     begin
      teta:=arctan((radvar+finr-2*Rtool)/(radvar+finr));
      BanderaBlanca:=true;
     end;
     if BanderaBlanca then
       begin
       if (Radvar>=(finr*cos(teta)+2*Rtool+D44)/(2-cos(teta))) then
        begin
         append(arctrampa);
         writeln(arctrampa,'G00 X',(xini-(Radvar+finr)*0.5):6:4,'Y',(yini-((Radvar+finr)/2-
Rtool)):6:4);
         writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini-(Radvar+finr-
Rtool)):6:4,'I',((Radvar+finr)/2):6:4,'F',F:6:4);
         writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini+Radvar+finr-
Rtool):6:4,'J',(Radvar+finr-Rtool):6:4);
         writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini-(Radvar+finr-Rtool)):6:4,'J',-
(Radvar+finr-Rtool):6:4);
         write(arctrampa,'G03 X',(xini+(Radvar+finr)*0.5):6:4);
         writeln(arctrampa,'Y',(yini-((Radvar+finr)*0.5-
Rtool)):6:4,'J',(Radvar+finr)*0.5:6:4);
         writeln(arctrampa,'G00 X',Xini:6:4,'Y',yini:6:4);
         close(arctrampa);
        end
        else
        begin
         append(arctrampa);
         writeln(arctrampa,'G01 X',xini:6:4,'Y',(yini-(Radvar+Dtx-Rtool)):6:4,'F',F:6:4);
         writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini+Radvar+Dtx-Rtool):6:4,'J',(Radvar+Dtx-
Rtool):6:4);
         writeln(arctrampa,'G03 X',Xini:6:4,'Y',(yini-(Radvar+Dtx-Rtool)):6:4,'J',-
(Radvar+Dtx-Rtool):6:4);
         writeln(arctrampa,'G01 X',Xini:6:4,'Y',yini:6:4);
         close(arctrampa);
        end
       end
     else
      begin
      append(arctrampa);
      writeln(arctrampa,'G01 X',xini:6:4,'Y',(yini-(Radvar+finr-Rtool)):6:4,'F',F:6:4);
      writeln(arctrampa,'G03 X',xini:6:4,'Y',(yini+Radvar+finr-Rtool):6:4,'J',(Radvar+finr-
Rtool):6:4);
      writeln(arctrampa,'G03 X',Xini:6:4,'Y',(yini-(Radvar+finr-Rtool)):6:4,'J',-
(Radvar+finr-Rtool):6:4);
      writeln(arctrampa,'G01 X',Xini:6:4,'Y',yini:6:4);
      close(arctrampa);
      end;
     radvar:=radprem;
    end; {maquinado radial}
   naxial:=naxial-1;
   If naxial > 0 then
    Dtz:=Dtz+Dtz1;
   If (naxial <= 0) and (residuoz<>0) and (finz<>0) then
    begin
     if BanderaNegra then
     begin
      Dtz:= Dtz+residuoz;
```

```
      BanderaNegra:=false;
     end
    else
     Dtz:=Dtz+finz
  end
 else if (naxial<=0) and (residuoz = 0) and (finz<>0) then
    Dtz:=Dtz+finr
 else if (naxial <=0) and (residuoz<>0) and (finr=0) then
   Dtz:=Dtz+residuoz
end;{while}
end; {fin del procedimiento}
```

Veamos el código que genera este programa a través de un ejemplo:

Ejemplo para macro programa de fresado combinado:

Se pide maquinar la siguiente pieza:

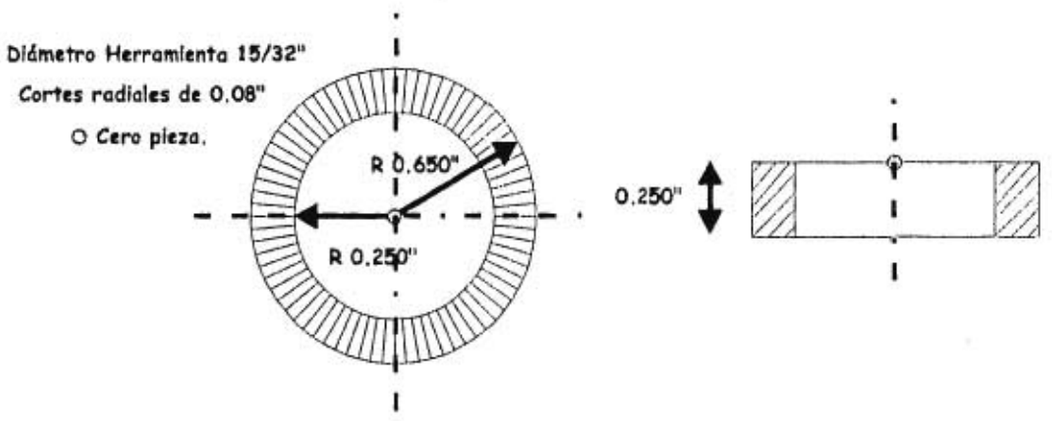


FIGURA 14

Nuestra profundidad de corte en Z será de 0.125" .

Dejaremos para acabado radial un espesor de 0.002" y para acabado axial uno de 0.005".

Ingresando estos datos en la macro, obtenemos el siguiente código:

```
G00 X 0      Y 0      G43 H01 Z 1.0000
G00 X 0      Y 0              Z 0.1000
G01                           Z-0.1250  F10.0000
G01 X 0      Y-0.0956                   F10.0000
G03 X 0      Y 0.0956         J 0.0956
G03 X 0      Y-0.0956         J-0.0956
G01 X 0      Y 0
G01 X 0      Y-0.1756                   F10.0000
G03 X 0      Y 0.1756         J 0.1756
G03 X 0      Y-0.1756         J-0.1756
G01 X 0      Y 0
G01 X 0      Y-0.2556                   F10.0000
G03 X 0      Y 0.2556         J 0.2556
G03 X 0      Y-0.2556         J-0.2556
G01 X 0      Y 0
```

```
G01 X 0      Y-0.3356                    F10.0000
G03 X 0      Y 0.3356          J 0.3356
G03 X 0      Y-0.3356          J-0.3356
G01 X 0      Y 0
G01 X 0      Y-0.4136                    F10.0000
G03 X 0      Y 0.4136          J 0.4136
G03 X 0      Y-0.4136          J-0.4136
G01 X 0      Y 0
G00 X-0.325 Y-0.0906                 (Inicia ciclo de Short Mill)
G03 X 0      Y-0.4156          I 0.3250 F8.0000
G03 X 0      Y 0.4156          J 0.4156
G03 X 0      Y-0.4156          J-0.4156
G03 X 0.325 Y-0.0906           J 0.3250
G00 X 0      Y 0
G01                            Z-0.2450 F10.0000
G01 X 0      Y-0.0956                    F10.0000
G03 X 0      Y 0.0956          J 0.0956
G03 X 0      Y-0.0956          J-0.0956
G01 X 0      Y 0
G01 X 0      Y-0.1756                    F10.0000
G03 X 0      Y 0.1756          J 0.1756
G03 X 0      Y-0.1756          J-0.1756
G01 X 0      Y 0
G01 X 0      Y-0.2556                    F10.0000
G03 X 0      Y 0.2556          J 0.2556
G03 X 0      Y-0.2556          J-0.2556
G01 X 0      Y 0
G01 X 0      Y-0.3356                    F10.0000
G03 X 0      Y 0.3356          J 0.3356
G03 X 0      Y-0.3356          J-0.3356
G01 X 0      Y 0
G01 X 0      Y-0.4136                    F10.0000
G03 X 0      Y 0.4136          J 0.4136
G03 X 0      Y-0.4136          J-0.4136
G01 X 0      Y 0
G00 X-0.325 Y-0.0906              (Inicia ciclo de acabado con Short Mill)
G03 X 0      Y-0.4156       I 0.3250  F8.0000
G03 X 0      Y 0.4156       J 0.4156
G03 X 0      Y-0.4156       J-0.4156
G03 X 0.325 Y-0.0906        J 0.3250
G00 X 0      Y 0
G01                         Z-0.2500 F10.0000
G01 X 0      Y-0.0956                 F10.0000
G03 X 0      Y 0.0956       J 0.0956
G03 X 0      Y-0.0956       J-0.0956
G01 X 0      Y 0
G01 X 0      Y-0.1756                 F10.0000
G03 X 0      Y 0.1756       J 0.1756
G03 X 0      Y-0.1756       J-0.1756
G01 X 0      Y 0
G01 X 0      Y-0.2556                 F10.0000
G03 X 0      Y 0.2556       J 0.2556
G03 X 0      Y-0.2556       J-0.2556
G01 X 0      Y 0
G01 X 0      Y-0.3356                 F10.0000
G03 X 0      Y 0.3356       J 0.3356
G03 X 0      Y-0.3356       J-0.3356
```

```
G01 X 0      Y 0
G01 X 0      Y-0.4136                F10.0000
G03 X 0      Y 0.4136       J 0.4136
G03 X 0      Y-0.4136       J-0.4136
G01 X 0      Y 0
G00 X-0.3250Y-0.0906                     (Ultimo ciclo de Short Milling )
G03 X 0      Y-0.4156       I 0.3250 F8.0000
G03 X 0      Y 0.4156       J 0.4156
G03 X 0      Y-0.4156       J-0.4156
G03 X 0.325 Y-0.0906        J 0.3250
G00 X 0      Y 0
```

Una aplicación avanzada: *Spiral Milling.*

Como el título lo indica, esta es una aplicación avanzada. Avanzada en el concepto de la forma de mecanizado que vamos a diseñar. Esta macro sirve para fresar un barreno en centro de maquinado. Pero a diferencia de las técnicas vistas anteriormente para tal efecto, esta presenta tres grandes ventajas . La primera es que podemos aplicarla a partir de una relación pequeña entre el prediámetro del barreno y el diámetro de la herramienta , esto es , permite maquinar dispersando la presión de la herramienta sin necesidad de tener un claro suficiente para utilizar la técnica de Arc- In. La segunda es que podemos utilizarla para remover de manera continua un gran volumen de sobrematerial. Y la última es que es fácilmente programable.

La figura 15 nos muestra en qué consiste este mecanizado e ilustra 7 coordenadas con el fin de facilitar la deducción de la macro, situándonos en puntos concretos .Más las ecuaciones que deduciremos servirán para n cortes. Observa que el cortador parece que describe un espiral , (de donde toma su nombre la técnica) , aunque en realidad es una serie de arcos, cada uno de mayor diámetro que el anterior. Cada vez que el cortador alcanza el lado izquierdo del arco, continúa su recorrido por otro de mayor diámetro.

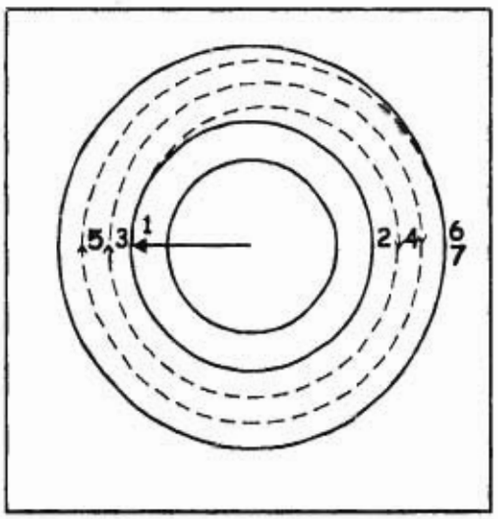


FIGURA 15

Asimismo se puede observar cómo se va incrementando suavemente la profundidad de corte, lo que nos ayuda a dispersar la presión del maquinado.

Cuando alcanzamos la profundidad de corte radial total, (en este caso el punto 6), la herramienta describe un movimiento de arco completo, acabando de maquinar todo el barreno. Posteriormente retraeremos la herramienta utilizando movimiento de Arc – Out (no indicado en la figura).

Deducción matemática del *tool path*

Elegimos arbitrariaremente aproximamos a mecanizar la pieza por la izquierda del cortador en la dirección X, como lo indica la figura 16. Definimos a Xini, Yini como las coordenadas del centro del barreno. En la figura es fácil ver que la primera coordenada será:

x1 = Xini – Radprem –Rtool

y1= Yini , donde Radprem será igual al prediámetro de nuestro barreno y Rtool al radio de nuestro cortador.

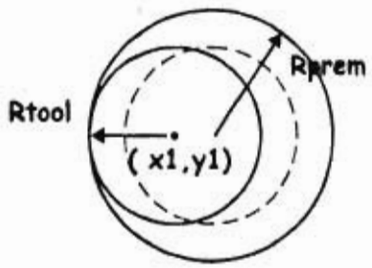


FIGURA 16

En sentido horario desplazamos el cortador de modo tal que creemos un arco , cuya distancia entre el extremo derecho del radio de premaquinado y su punto final sea igual a la profundidad de corte. Ver figura 17 izquierda. Nuestra coordenada en X estará dada por el Radprem + Pcr – Rtool, más la posición inicial de X .Y será igual a Yini.

FIGURA 17

Por lo tanto:

x2 = Xini +Radprem +Pcr –Rtool ; y2 = Yini

Veamos ahora cómo podemos calcular nuestro vector director :

El arco que se forma al desplazar la herramienta posee un diámetro igual al de premaquinado más la profundidad de corte radial. Por lo que el centro del arco estará desplazado a la derecha de nuestro radio de premaquinado en Pcr /2 . Ahora en la figura 17 derecha observa que únicamente tendremos vector director en I.

De la definición de vector director , tenemos que I será la distancia incremental desde el punto de inicio de corte, el centro de nuestro cortador en la posición x1,y1 o Xini – (Radprem – Rtool), Yini hasta el centro de nuestro arco (Xini +Pcr /2, Yini). Por lo tanto I = Pcr /2 + (Radprem – Rtool) o de otro modo:

$$I = \frac{\phi\ Prem + Pcr - \phi\ Tool}{2} \qquad (1)$$

Para indicar cómo obtener la tercera coordenada recurrimos a ejemplificar con la siguiente figura.

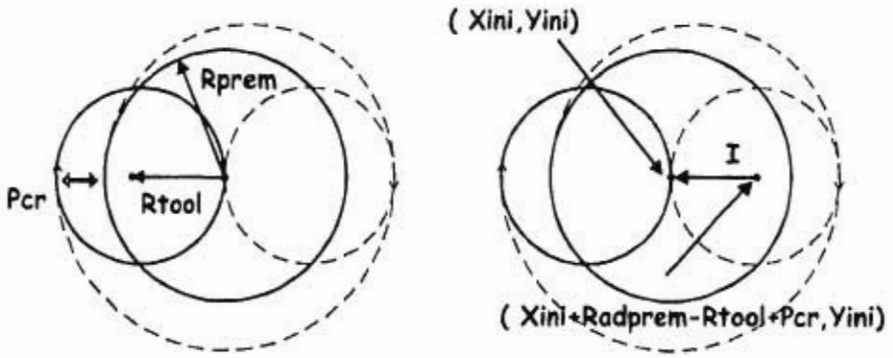


FIGURA 18

Desplazamos la herramienta creando otra vez un arco cuya distancia entre su punto final y el extremo izquierdo de nuestro radio de premaquinado sea igual a la profundidad de corte radial. Nota que ahora las coordenadas de su centro son las mismas que las del radio de premaquinado.

De la figura izquierda es fácil deducir las coordenadas del centro de nuestra herramienta :

x3 = Xini – (Radprem –Rtool +Pcr); y3 = yini

Nuestra coordenada en x3, es la misma que en x2 , pero con signo negativo.

Ahora se deduce I utilizando otra vez el concepto de distancia. Veamos la figura 18 (parte derecha), nuestro punto inicial será el radio de la herramienta en la posición 2. Nuestro punto final será el centro del radio de premaquinado o sea nuestro punto inicial, por lo que:

$$I = -(Radprem + Pcr\ Rtool) \qquad (2)$$

Para facilitar la programación de nuestro algoritmo, veamos si hay manera de relacionar la ecuación 2 de nuestro vector director con la que hemos deducido anteriormente :
Multiplicando y dividiendo por dos nuestra ecuación número 2 tenemos:

$$I = -\frac{[RadPrem + Pcr - Rtool]}{2} x2 = -\frac{\phi\ Prem + 2Pcr - \phi\ Tool}{2}$$

Que es nuestra misma ecuación 1 , con la salvedad de que se ha incrementado nuestra variable de corte radial, como lógicamente era de esperar que ocurriera.
Calculemos nuestra cuarta coordenada:

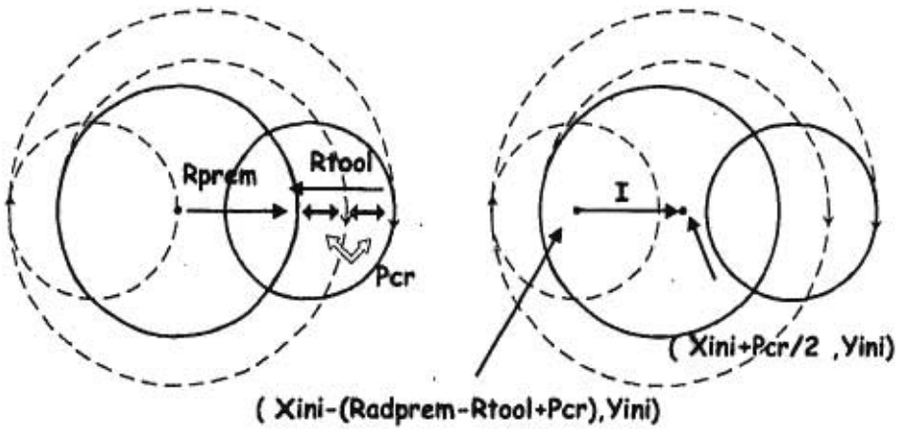


FIGURA 19

De manera similar, de nuestra figura 19 parte izquierda, la posición del centro del cortador estará dada por:
x4 = Rprem +2Pcr –Rtool , y4 = yini .
Nota como ya se incrementó esta ecuación en la profundidad de corte radial.
Nuevamente deducimos I usando la figura 19 parte derecha y la fórmula de distancia entre dos puntos:

$$I = \frac{Pcr}{2} + Radprem + Pcr - Rtool = \frac{\phi Prem + 3Pcr - \phi Tool}{2}$$

Nos damos cuenta de que ya se empieza a repetir cierto patrón, mismo que podemos modelar matemáticamente. Por abreviar ya no deduciremos la siguiente coordenada, sin embargo la escribiremos para observar el patrón que siguen nuestras ecuaciones y puedas seguir nuestro razonamiento:

x5 = Xini- (Radprem –Rtool + 2Pcr); y5 = Yini

Con nuestro vector director igual a:

$$I = \frac{\phi\ Prem + 4Pcr - \phi Tool}{2}$$

Podemos decir ya que las coordenadas para el mecanizado circular pueden generalizarse a partir de las siguientes ecuaciones:

Para n = 2, 4, 6, 8, 10 ... G02 a

$$X_n = Xini + Radprem + \frac{n}{2} \times Pcr - Rtool$$

$$Y_n = Yini \qquad (3)$$

$$I = \frac{\phi\ Prem - \phi\ Tool + (n-1)Pcr}{2}$$

Para n = 3,5,7,9, 11... G02 a

$$X_n = Xini - (Radprem + \frac{(n-1)}{2} \times Pcr - Rtool)$$

$$Y_n = Yini \qquad (4)$$

$$I = -\frac{(\phi\ Prem - \phi\ Tool + (n-1)Pcr)}{2}$$

N se calculará tomando la parte entera del resultado de dividir la diferencia del radio final y el radio de premaquinado entre la profundidad de corte radial.

Así tenemos: $n\ entera = Int\ \frac{Radiofinal - Radioprem.}{Pcr}$

Donde Int indica que se tomará la parte entera de la operación.

Consideraremos también un residuo de n previendo el caso de que la operación anterior tenga parte fraccionaria.

$$n\ residuo = Fracc\ [\ \frac{Radiofinal - Radioprem.}{Pcr}\ ] \times Pcr$$

Fracc indica el valor fraccionario de la operación.

Ahora deduciremos las n veces que tendremos que realizar nuestro maquinado. Observa las dos figuras siguientes: En la izquierda n ha salido exacta , en este caso su valor ha sido 3,

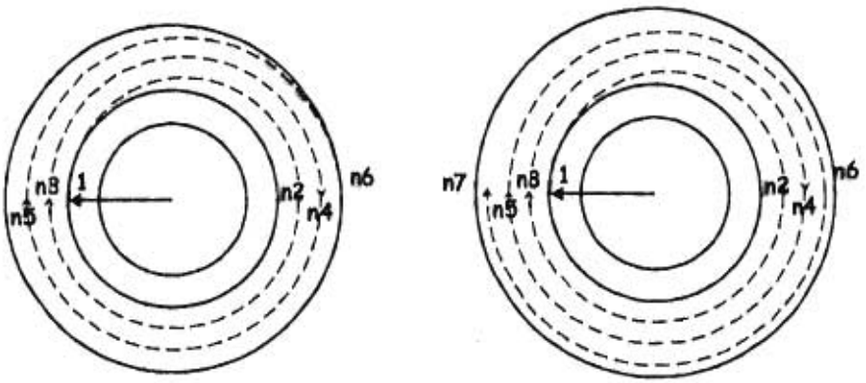


FIGURA 20

necesitaremos pues emplear 5 veces las ecuaciones deducidas anteriormente para interpolación circular . Empezaremos desde n = 2 hasta n= 6, (nuestra coordenada 1 fue deducida para aproximación lineal de la herramienta al punto de corte). De manera general puede decirse, que si n entera no tiene residuo , el número de veces que utilizaremos las ecuaciones de interpolación será de 2 veces n entera.

En la figura de la derecha, n residuo es diferente de cero, esto quiere decir que la distancia entre el punto final de nuestro arco descrito por n6 y el extremo derecho de nuestro radio final es precisamente ese residuo. Utilizaremos en este caso, una ecuación de interpolación adicional , n = 7 , con objeto de crear un arco que posicione a la herramienta en el extremo izquierdo y a partir de ahí, generar otro arco (cuyo diámetro estará incrementado en n residuo). También es posible afirmar de manera general, que para n residuo diferente de cero, el número de ecuaciones de interpolación necesarias será de 2 n entera +1.

Ahora si n residuo ha salido diferente de cero, ¿ cúales serán las ecuaciones que definan las cooordenadas del arco que describe el cortador ? .La siguiente figura (21) nos ayudará a deducirlo.

De la figura de la derecha , se puede deducir que las coordenadas de fin de arco serán:

Xfin = Xini +Rfinal –Rtool

Yfin = Yini

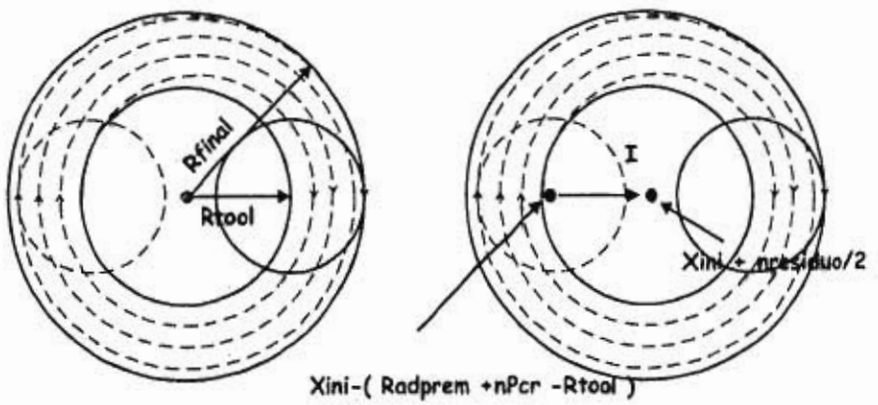


FIGURA 21

Veamos ahora la figura de la izquierda para deducir I. La coordenada en X del último arco en que hemos maquinado con Pcr , será impar, por lo que usaremos :

$$X_n = \text{Xini} - (\text{Radprem} + \frac{(n-1)}{2} \times \text{Pcr} - \text{Rtool})$$

Para dicha coordenada n será igual a 2 n entera +1 sustituyendo tenemos:

$$X_n = \text{Xini} - (\text{Radprem} + (\text{n entera}) \text{ Pcr} - \text{Rtool})$$

I será la distancia incremental entre esta coordenada y el centro del arco final que describe la herramienta.

Este arco tendrá un diamétro igual al diámetro de premaquinado aumentado n entera veces la profundidad de corte radial, más n residuo. Su centro estará en Xini + nresiduo / 2.

Por lo tanto:

$$I = \frac{\text{nresiduo} + \phi \text{Prem} - \phi \text{Tool} + 2 \text{ n entera Pcr}}{2}$$

Una vez definidas las ecuaciones para la coordenada donde nuestra herramienta alcanza la profundidad de corte total procedemos a definir las que realizan el maquinado de la última espiral. Dividimos el movimiento de interpolación en dos, por las razones expuestas cuando desarrollamos la macro de Circular Mill.. Nuestro primer movimiento será al extremo izquierdo del radio. Observamos en la figura 22 que las coordenadas están dadas por:

Para el medio arco del lado izquierdo:

G02 a Xini – (Radfinal –Rtool)
Yini
I = - (Rfinal – Rtool)

Para completar nuestro medio arco:

G02 a Xini + (Radfinal –Rtool)
Yini
I = Rfinal +Rtool

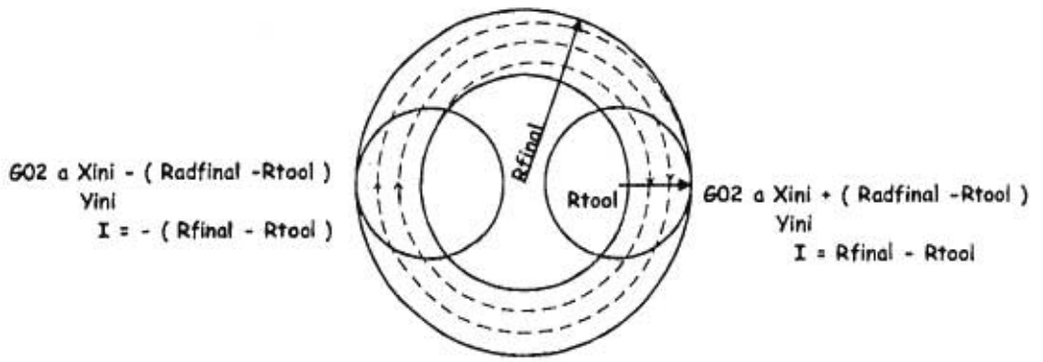


FIGURA 22

Ahora solo falta retraer la herramienta, usamos para ello la técnica de Arc –Out.

Daremos la oportunidad al programador de escojer la distancia de retracción que él considere conveniente utilizando para ello un parámetro al cual llamaremos E25.

Obtendremos una distancia adecuada de retracción si retiramos a X y Y en el radio final multiplicado precisamente por E25. (E25 será dado en porcentaje)

De otro modo:

Nuestra ecuación final en X está dada por: Xini + (Radfinal –Rtool)

Restándole la distancia de retracción tenemos:

$$Xini + Radfinal - Rtool - Radfinal \times E25$$

La cual simplificamos como:

$$Xini + Radfinal(1 - E25) - Rtool$$

Que es nuestra ecuación de retracción en X.

La ecuación de retracción en Y será: Yini – Radfinal x E25

Con I = - E25 x Radfinal

Terminando la deducción matemática, procedamos a resumir nuestro algoritmo.

1. Se realiza el cálculo previo de n entera así como de n residuo.
2. En rápido ubicamos al cortador en el centro del barreno en el plano XY, moviendo simultáneamente la herramienta a Z inicial instalando compensación de longitud.
3. Desplazamos la herramienta en Z hasta inicio del plano R (definido por parámetro).
4. Con avance, el cortador baja a la profundidad de corte en Z.
5. Movemos la herramienta a punto de inicio de corte:
   G01 a X = Xini – (Rprem –Rtool) Y = Yini
6. En base a n entera , utilizaremos las ecuaciones 3 y 4 para deducir las coordenadas de nuestro cortador desde n = 2 hasta 2 n entera. Si nresiduo ha sido cero, pasamos directamente al inciso 8; caso contrario ejecutamos el siguiente (7).

7 . Si n residuo es diferente de cero usamos la ecuación 4 con n = 2n entera + 1. Con esta ecuación se programa un arco que coloca el cortador del lado izquierdo de nuestro barreno. A partir de ahí utilizamos las siguientes ecuaciones, que llevan a nuestro cortador otra vez al extremo derecho del barreno, maquinando el espesor que quedó como residuo.

G02 a Xfin = Xini +Rfinal –Rtool

Yfin = Yini

$$I = \frac{\text{nresiduo} + \phi\,\text{Prem} - \phi\,\text{Tool} + 2\,\text{n entera} \times \text{Pcr}}{2}$$

11. Programamos un arco completo , dividiéndolo en dos segmentos.

Primer Segmento: G02 a Xini – (Radfinal –Rtool)

Yini

I = - (Rfinal –Rtool)

Segundo segmento: G02 a Xini + (Radfinal –Rtool)

Yini

I = + (Rfinal – Rtool)

12. Retraemos la herramienta con Arc – Out . Las ecuaciones de retracción son:

G02 a : Xini + Radfinal(1 - E25) - Rtool

Yini –Radfinalx E25

I = - E25 x Radfinal

13. En rápido, desplazamos la herramienta al centro del barreno. Si es necesario cortar más en Z, nos movemos al punto 4 y ejecuamos el algoritmo a partir de ese punto. Si ya hemos alcanzado la profundidad de corte axial total, procedemos a sacar la herramienta del barreno en Z , moviéndonos primero al plano R y posteriormente a Z inicial; y con esto terminamos el maquinado.

Programa en PASCAL para la macro de *Spiral Milling*.

```
const
D41=0.1; {parámetro de plano R}
S4=1E-5; {punto flotante}
Zinicial=1.0; {nuestra Z inicial}
E25=0.25;{parámetro de retracción}
var
xini,yini,F,Ff,Pcr,Radprem,Radfi,tooldia,Zllegada,Pcztotal,Pcz,
Finz,FinR:real;
retorno:boolean;
archivo:text;

procedure SpiralMill(xini,yini,F,Pcr,Radprem,Radfi,Rtool,Zllegada,
Pcztotal,Pcz,Finz:real;retorno:boolean;var arctrampa:text);
var
Dtx,residuor,residuoz,Dtz,Dtzl:real;
Banderanegra:boolean;
nr,nz,i:shortint;
begin
 Banderanegra:=true;
 assign(arctrampa,'Spiral.fil');
 rewrite(arctrampa);
 writeln(arctrampa,'(SPIRAL MILL)');
 writeln(arctrampa,'G00 X',xini:6:4,'Y',yini:6:4,'Z',(D41+Zllegada):6:4);
 close(arctrampa);
 nz:=trunc((Pcztotal-Finz)/pcz);
 residuoz:=puntoflotante(Pcz*(frac((Pcztotal-Finz)/Pcz)));
 if nz<=0 then
  Dtzl:=residuoz
 else
 Dtzl:=Pcz;
  Dtz:=Dtzl;
 while (Dtz<=(Pcztotal-Finz)) and BanderaNegra do
  begin
   append(arctrampa);
   writeln(arctrampa,'G01 Z',(Zllegada-Dtz):6:4,'F',F:4:4);
   nz:=nz-1;
   if (nz=0) and (residuoz<>0) then
     Dtz:=Dtz+residuoz
   else if (nz=0) and (residuoz=0) then
    begin
     if finz=0 then
      BanderaNegra:=false
     else
      Dtz:=Dtz+finz;
    end
  else if (nz<0) and (residuoz<>0) then
   begin
    if (finz=0) then
     BanderaNegra:=false
   else
     Dtz:=Dtz+finz
   end
  else if nz>0 then
   Dtz:=Dtz+Dtzl;
 nr:=trunc((Radfi-radprem)/Pcr);
 residuor:=puntoflotante((frac((Radfi-radprem)/pcr))*Pcr);
 if nr<=0 then
  Dtx:=residuor
 else
 Dtx:=pcr;
append(arctrampa);
writeln(arctrampa,'G01 X',( xini-(radprem-Rtool) ):6:4,'Y',yini:6:4,'F',F:6:4);
close(arctrampa);
  For I:=2 to (2*nr) do
  begin
   append(arctrampa);
```

Programa en PASCAL para la macro de *Spiral Milling*.

```
const
D41=0.1; {parámetro de plano R}
S4=1E-5; {punto flotante}
Zinicial=1.0; {nuestra Z inicial}
E25=0.25;{parámetro de retracción}
var
xini,yini,F,Ff,Pcr,Radprem,Radfi,tooldia,Zllegada,Pcztotal,Pcz,
finz,FinR:real;
retorno:boolean;
archivo:text;

procedure SpiralMill(xini,yini,F,Pcr,Radprem,Radfi,Rtool,Zllegada,
Pcztotal,Pcz,Finz:real;retorno:boolean;var arctrampa:text);
var
Dtx,residuor,residuoz,Dtz,Dtzl:real;
Banderanegra:boolean;
nr,nz,I:shortint;
begin
 Banderanegra:=true;
 assign(arctrampa,'Spiral.til');
 rewrite(arctrampa);
 writeln(arctrampa,'(SPIRAL MILL)');
 writeln(arctrampa,'G00 X',xini:6:4,'Y',yini:6:4,'Z',(D41+Zllegada):6:4);
 close(arctrampa);
 nz:=trunc((Pcztotal-Finz)/pcz);
 residuoz:=puntoflotante(Pcz*(frac((Pcztotal-Finz)/Pcz)));
 if nz<=0 then
  Dtzl:=residuoz
 else
 Dtzl:=Pcz;
  Dtz:=Dtzl;
 while (Dtz<=(Pcztotal-Finz)) and BanderaNegra do
  begin
   append(arctrampa);
   writeln(arctrampa,'G01 Z',(Zllegada-Dtz):6:4,'F',F:4:4);
   nz:=nz-1;
   if (nz=0) and (residuoz<>0) then
     Dtz:=Dtz+residuoz
   else if (nz=0) and (residuoz=0) then
    begin
     if finz=0 then
      BanderaNegra:=false
     else
      Dtz:=Dtz+finz;
    end
  else if (nz<0) and (residuoz<>0) then
   begin
    if (finz=0) then
     BanderaNegra:=false
   else
     Dtz:=Dtz+finz
   end
  else if nz>0 then
   Dtz:=Dtz+Dtzl;
 nr:=trunc((Radfi-radprem)/Pcr);
 residuor:=puntoflotante((frac((Radfi-radprem)/pcr))*Pcr);
 if nr<=0 then
  Dtx:=residuor
 else
 Dtx:=pcr;
append(arctrampa);
writeln(arctrampa,'G01 X',( xini-(radprem-Rtool) ):6:4,'Y',yini:6:4,'F',F:6:4);
close(arctrampa);
  For I:=2 to (2*nr) do
  begin
   append(arctrampa);
```

```
    If odd(I) then
     writeln(arctrampa,'G02 X',(xini-( Radprem +(0.5*Dtx*(I-1))-
rtool)):6:4,'Y',yini:6:4,'I',-(Radprem-Rtool+(I-1)*Dtx*0.5):6:4)
     else
    writeln(arctrampa,'G02 X',(Xini+Radprem+(I*0.5*Dtx)-
rtool):6:4,'Y',(yini):6:4,'I',((2*Radprem+(I-1)*Dtx-2*Rtool)*0.5):6:4);
   close(arctrampa);
  end;
 if residuor <>0 then
  begin
   append(arctrampa);
   writeln(arctrampa,'G02 X',(xini-(Radprem+nr*Dtx-rtool)):6:4,'Y',yini:6:4,'I',-(Radprem-
rtool+nr*Dtx):6:4);
   writeln(arctrampa,'G02 X',(xini+Radfi-rtool):6:4,'Y',yini:6:4,'I',-(residuor*0.5+Radprem-
rtool+nr*Dtx):6:4);
  close(arctrampa);
  end;
append(arctrampa);
writeln(arctrampa,'G02 X',(xini-(Radfi-Rtool)):6:4,'Y',yini:6:4,'I',-(Radfi-Rtool):6:4);
writeln(arctrampa,'G02 X',(xini+(Radfi-Rtool)):6:4,'Y',yini:6:4,'I',(Radfi-Rtool):6:4);
writeln(arctrampa,'G02 X',(xini+Radfi*(1-E25)-Rtool):6:4,'Y',(yini-Radfi*E25):6:4,'I',(-
E25*Radfi):6:4);
writeln(arctrampa,'G00 X', xini:6:4,'Y',Yini:6:4);
close(arctrampa);
end; {bajada en z}
append(arctrampa);
 if retorno then
  writeln(arctrampa,'G00 Z',Zinicial:6:4)
else
 writeln(arctrampa,'G00 Z',D41:6:4);
close(arctrampa);
end;
```

Ejemplo de maquinado utilizando *Spiral Milling*.

En este ejemplo desarrollaremos dos códigos en EIA – ISO. El primero será con una profundidad de corte radial de 0.125", de tal manera que n entera salga exacta. Para nuestro segundo código Pcr será de 0.120" de modo que obtengamos un residuo. Esto es con el fin de que rastrees el algoritmo paso a paso y tengas oportunidad de comprobar las ecuaciones que hemos deducido. Para ambos ejemplos realizaremos dos cortes en Z de 0.125" cada uno.

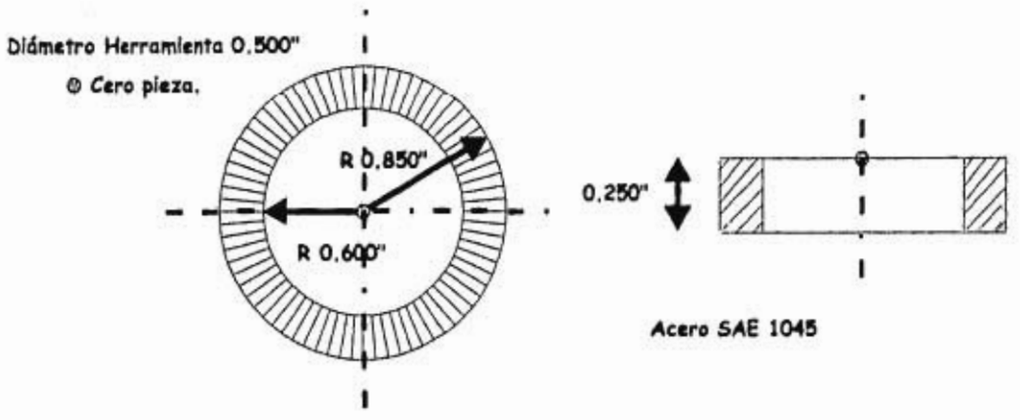


FIGURA 23

Para Pcr igual a 0.125" obtenemos el siguiente código:

(SPIRAL MILL)

```
G00 X 0       Y 0              Z 0.1000
G01                            Z-0.1250 F15.0000
G01 X-0.3500 Y 0                        F15.0000
G02 X 0.4750 Y 0                        I 0.4125
G02 X-0.4750 Y 0                        I-0.4750
G02 X 0.6000 Y 0                        I 0.5375
G02 X-0.6000 Y 0                        I-0.6000
G02 X 0.6000 Y 0                        I 0.6000
G02 X 0.3875 Y-0.2125                   I-0.2125
G00 X 0       Y 0
G01                            Z-0.2500 F15.0000
G01 X-0.3500 Y 0                        F15.0000
G02 X 0.4750 Y 0                        I 0.4125
G02 X-0.4750 Y 0                        I-0.4750
G02 X 0.6000 Y 0                        I 0.5375
G02 X-0.6000 Y 0                        I-0.6000
G02 X 0.6000 Y 0                        I 0.6000
G02 X 0.3875 Y-0.2125                   I-0.2125
G00 X 0       Y 0
G00                            Z1.0000
```

Para Pcr igual a 0.120" obtenemos el siguiente código:

```
(SPIRAL MILL)

G00 X 0       Y 0              Z 0.1
G01                            Z-0.1250  F15.0000
G01 X-0.3500 Y 0                        F15.0000
G02 X 0.4700 Y 0                        I 0.4100
G02 X-0.4700 Y 0                        I-0.4700
G02 X 0.5900 Y 0                        I 0.5300
G02 X-0.5900 Y 0                        I-0.5900
G02 X 0.6000 Y 0                        I-0.5950
G02 X-0.6000 Y 0                        I-0.6000
G02 X 0.6000 Y 0                        I 0.6000
G02 X 0.3875 Y-0.2125                   I-0.2125
G00 X 0       Y 0
G01                            Z-0.2500  F15.0000
G01 X-0.3500 Y 0                        F15.0000
G02 X 0.4700 Y 0                        I 0.4100
G02 X-0.4700 Y 0                        I-0.4700
G02 X 0.5900 Y 0                        I 0.5300
G02 X-0.5900 Y 0                        I-0.5900
G02 X 0.6000 Y 0                        I-0.5950
G02 X-0.6000 Y 0                        I-0.6000
G02 X 0.6000 Y 0                        I 0.6000
G02 X 0.3875 Y-0.2125                   I-0.2125
G00 X 0       Y0
G00                            Z1.0000
```

Diseño de un Ciclo de Taladrado.

Puede pensarse que el taladrado es una de las operaciones de maquinado más sencillas que puede haber. Porque simplemente hay que empujar una broca a través del material y así obtenemos nuestro barreno. En aparencia puede ser así , sin embargo ,dependiendo de la vida que se quiera dar a la herramienta, del material que se quiera maquinar, incluso del tipo de proceso , esto no puede ser lo óptimo. En realidad existe un sinnúmero de variaciones del ciclo de taladrado. Con seguridad tu control CNC viene equipado con varias de ellas, pero muchas veces tendremos que adecuar o diseñar por entero un ciclo en nuestro propio taller para maquinar cierta pieza de x material. De hecho lo complicado de un ciclo de taladrado no es el programarlo, sino diseñarlo para que opere de forma eficiente.

A continuación veremos tres ciclos típicos de taladrado, explicaremos su mecánica de operación, así como ciertos trucos que te ayudarán a diseñar uno propio.

Ciclo de Taladrado Normal:

La mecánica de este ciclo es muy simple y puede seguirse por la secuencia de números marcados en la figura 24. Quizás lo que se aparta un poco de la idea que teníamos de "empujar una broca" y merezca que los discutamos sean los puntos 4 y 5.

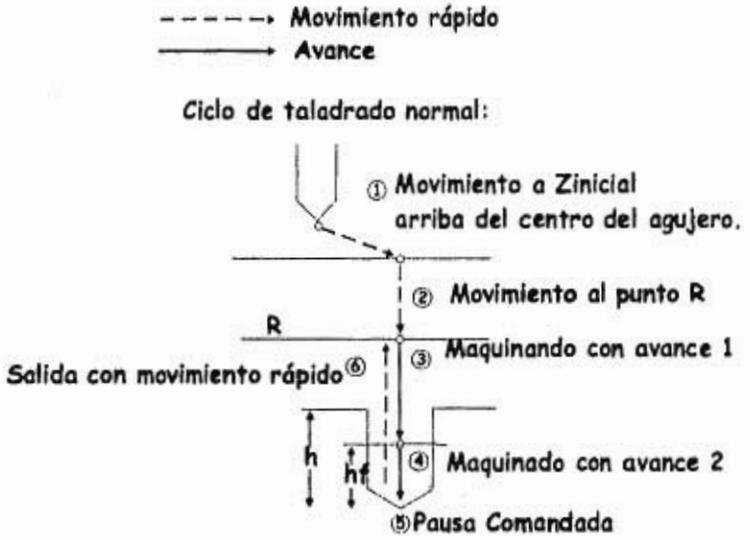


FIGURA 2.4

En el punto 4 avanzamos la herramienta con un avance inferior al programado a partir de una distancia llamada de desaceleración u "override " (hf) hasta completar el barrenado . Esto es a debido a la naturaleza misma del mecanizado por broca. Si el barreno es profundo la rebaba en el

fondo no puede salir fácilmente, ni hay refrigeración adecuada por soluble . Si proseguimos con el mismo avance, la herramienta corre el riesgo de sobrecalentarse, atascarse y por tanto romperse, por eso se maquina con una segunda velocidad de avance, la cual es generalmente un porcentaje de la primera.

Ahora ¿ cómo determinar hf ? . En realidad esto es una buena pregunta, como buena pregunta es cómo saber el porcentaje de disminución de avance que se debe programar. Es imposible generalizar, pues tanto hf como el porcentaje de reducción de avance están en función de la herramienta , del material y de las condiciones de corte. Una buena respuesta se puede obtener en los catálogos de los fabricantes de herramientas, donde puede conseguirse una aproximación de estos valores, en base al tipo de broca que piense utilizarse y el material que se va a maquinar. Sin embargo los valores correctos los obtendrás cuando físicamente maquines una pieza y los corrijas en base a los resultados del corte.
En mi experiencia personal he encontrado la siguiente relación que funciona relativamente bien.

Para:	h <= 1"	hf = 0
	1 < h <= 1.5"	hf = 0.2 x Diámetro barreno
	1.5"< h	hf = Diámetro barreno / 2

Donde h es la profundidad total del barreno.

Cuando programes una macro de ciclo de taladrado sencillo, es buena idea considerar declarar tanto hf como el porcentaje de reducción de avance como variables dentro de la misma, de modo que el operador pueda ingresar los valores que considere adecuados.
Ahora pasemos a hablar de la secuencia 5 . En ella estamos comandando una pausa en el maquinado por medio del comando G04 .
Siempre que requiramos una gran potencia de corte al terminar el maquinado, es conveniente programar una pausa para liberar la presión que genera el mecanizado. Esto es especialmente cierto para el taladrado. Conforme la broca se va introduciendo en el barreno, la presión generada tiende a empujar a la herramienta hacia afuera; si retraemos la broca tan pronto como acabamos de maquinar el agujero, éste no tendrá la profundidad programada.
Hablemos un poco más de este comando G04.
La función de este comando es retardar la ejecución de la siguiente línea , dependiendo del modo de avance en que se encuentre la máquina (G94 avance por minuto, G95 avance por revolución) esta interpretará la pausa ya sea en tiempo o en número de revoluciones respectivamente.
El comando G94 es seleccionado como modo inicial en la mayoría de los centros de maquinado, por lo que cuando usemos G04 especificaremos por lo regular una pausa en tiempo.

El formato de programación de G04 es como sigue:
G04 X_ o G04 P _ ; en el primer modo (X) el comando acepta punto decimal ; en el segundo (P) ignora el punto decimal aún si en la cantidad está especificado.
Hablemos ahora de la profundidad h en la que se programa el barrenado.

Fórmulas para programar profundidad de barrenado:

En la mayoría de los planos de taller, cuando se nos especifica una distancia de taladrado para un barreno no pasado, se nos pide la distancia que nos muestra la figura 25 a la izquierda, sin embargo siempre que usemos una broca que termine en punta debemos aumentar un poco más la profundidad especificada. Esto es por la misma geometría de la herramienta. Llamamos a esto compensación de taladrado.

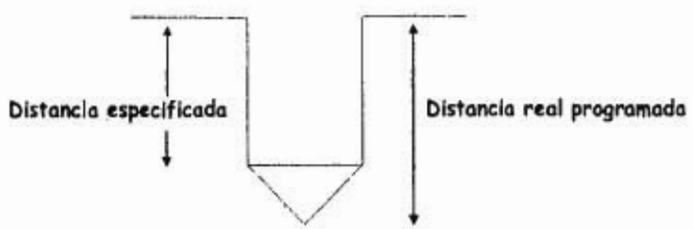


FIGURA 25

Hay una fórmula muy conocida para calcular ese pequeño monto que hay que agregar a nuestra profundidad de mecanizado, ésta es:
Distancia real = Distancia especificada + 0.3 x Diámetro Broca.
Esta fórmula es útil para brocas con ángulo a 118°, sin embargo falla cuando usamos una broca con otro ángulo.
La fórmula general que sirve para calcular compensación de taladrado con cualquier ángulo de herramienta y que usaremos en la programación de toda macro de taladrado es la siguiente:

Sea θ ángulo de nuestra broca; Para $0 < \theta < 180°$

$$\text{Compensación} = \frac{\text{Dia. Herramienta}}{2\tan[\theta/2]} + 0.317x\text{Dia. Herramienta}$$

Para $\theta = 180°$; Compensación = 0.317 x Dia.Herramienta

Para $\theta = 0$; Compensación = 0.

En el último caso con el ángulo igual a cero se contempla el uso de una broca de insertos .

Nota: Esta fórmula está diseñada para programarse usando unidades inglesas.Cuando se programe en milímetros únicamente habrá que sustituir 0.317 por 0.1.

¿Cúando utilizaremos ciclo normal? Cuando nuestra herramienta sea una broca de centros, en barrenos cuya profundidad no exceda 3 veces su diámetro, en materiales con viruta quebradiza y **siempre** que usemos broca de insertos. Las brocas de insertos son diseñadas para maquinar en una sola pasada. Si intentamos utilizar broca de insertos en combinación con los técnicas de picoteo (que veremos posteriormente), dará como resultado daño severo a los insertos y a nuestra herramienta.

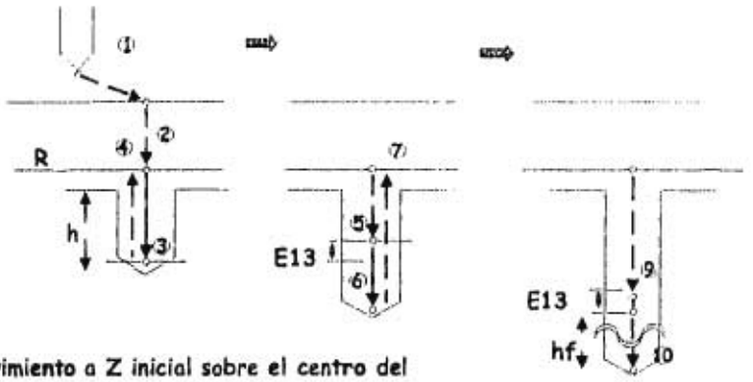
Ciclo de taladrado con picoteo y retorno a plano R:

En alguno materiales, el movimiento descrito en nuestro primer ciclo, (entrada con avance, salida en rápido) , causa que se generen tiras largas de rebaba. Estas tiras tienden a enredarse en la herramienta conforme penetra más y más profundo en el material. Provocando que la herramienta se atasque , pudiendo romperse ; además de que el acabado dejará mucho que desear. Para evitar esto, se usa lo que se denomina "picoteo". Consiste básicamente en retraer la herramienta despúes de haber penetrado una cierta distancia , penetrar otra vez y volver a retraer imitando se puede decir un "golpe de ariete" . El movimiento de penetrar y retraer , rompe las tiras de rebaba formadas por efecto del mecanizado, dando su nombre a esta técnica. Las técnicas de taladrado por picoteo se prestan increíblemente bien a ser desarolladas en un ciclo. Ya te imaginarás la longitud del programa si las programamos usando solamente G00 y G01 . Hay muchas variaciones de este ciclo de picoteo, las cuales estarán en función precisamente de la distancia que se retrae la herramienta. En el ciclo de la figura observamos un ciclo de picoteo con retorno al plano R. Básicamente consiste en taladrar una distancia especificada de picoteo h , (ya sea por el usuario o determinada en nuestra macro) regresando la herramienta con movimiento rápido hasta el plano R. Posteriormente avanzamos en rápido hasta una distancia de claro (especificada por parámetro el cual llamaremos E13), a partir de ésta se programa de nuevo avance y proseguiremos maquinando. Repetiremos esta mecánica hasta acabar de maquinar el barreno, tal como ilustra la secuencia de la figura 26.

Este ciclo está diseñado para utilizarse en materiales cuyo maquinado genere tiras largas de rebaba, que no se rompen fácilmente por el simple retroceso de la herramienta. Así como en agujeros cuya profundidad exceda al menos 3 veces el diametro del agujero a maquinarse. En ambos casos la rebaba tiende a atorarse en el fondo del barreno, siendo necesario "sacarla" , subiendo la herramienta hasta el plano R.

— — → Movimiento rápido
——→ Avance

Ciclo de taladrado profundo (picoteo 1):



① Movimiento a Z inicial sobre el centro del agujero a maquinar
② Movimiento al punto R
③ Maquinando con avance 1
④ Retorno al punto R
⑤ Movimiento a la posición determinada por E13
⑥ Maquinando con avance 1
⑦ Retorno al punto R
⑧ Movimiento a la posición determinada por E13
⑨ Repetición de los puntos 5 al 7 hasta distancia de cambio de avance
⑩ Maquinado con avance 2 Pausa Comandada

FIGURA 26

Hablemos un poco de la distancia h que penetra la herramienta antes de retraerse . Una broca estándar no esta diseñada por lo regular, para penetrar su profundidad máxima de corte en una pasada. Por regla general puede más o menos penetrar una distancia igual a tres veces su diámetro antes de que la rebaba se atasque en ella. Esta regla funciona bien en la práctica y a menos que se nos especifique directamente esta profundidad h , presetearemos dentro de nuestra macro a h igual a tres veces el diámetro de la herramienta seleccionada.

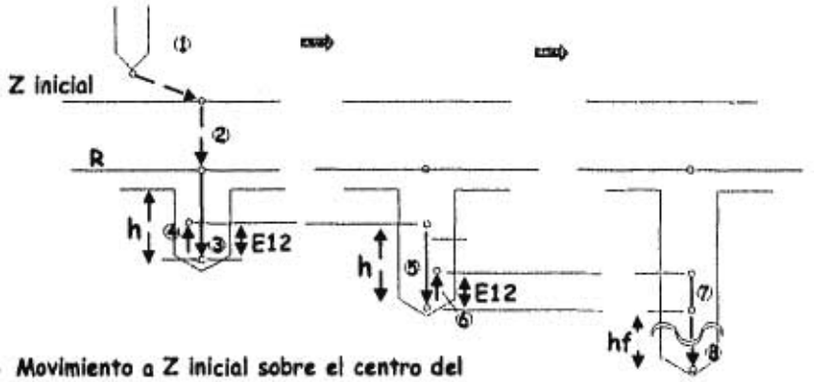
Hablemos ahora de la distancia de claro E13 , la herramienta avanzará en rápido desde el piano R hasta esta distancia , (la cual se suma positivamente a nuestra distancia h) , a partir de la cual programamos otra vez avance , para proseguir nuestro maquinado. Es claro que esta distancia debe ser especificada como parámetro , para que pueda ser ajustada libremente por el operador en nuestra macro.

Ciclo de taladrado rápido con picoteo.

Cuando surge la necesidad de usar picoteo, si maquinas materiales cuya rebaba tienda a romperse fácilmente mediante un pequeño retroceso de la herramienta o necesitas economizar tiempo,el ciclo anterior tal vez no sea el óptimo para ser utilizado en tu mecanizado.

— — → Movimiento rápido
——→ Avance

Ciclo de picoteo con retroceso rápido: FIGURA 27



① Movimiento a Z inicial sobre el centro del agujero a maquinar
② Movimiento al punto R
③ Maquinando con avance 1
④ Retorno al punto definido por E12
⑤ Maquinado a profundidad h
⑥ Retorno al punto definido por E12
⑦ Repetición de los puntos 5 y 6 hasta distancia de cambio de avance
⑧ Maquinado con avance 2 Pausa Comandada

En estos casos utilizamos un ciclo con picoteo rápido. Como puedes observar en la figura 27, retrocedemos con movimiento rápido únicamente la distancia de picoteo, para posteriormente proseguir nuestro máquinado.

¿Qué distancia es aconsejable retroceder para picoteo ? la respuesta es variable, podemos emplear valores desde 0.005" para materiales muy suaves hasta 0.100" para materiales menos dúctiles. Este rango de valores nos hace pensar que será buena idea el declarar esta distancia de retroceso como parámetro dentro de nuestra macro , en la figura 27 lo hemos identificado como E12.

Este será el ciclo que diseñaremos en un macroprograma. El desarrollo de los otros dos ciclos anteriores se deja como ejercicio al lector.

Diseño del ciclo de taladrado rápido con picoteo:

Primero establezcamos la distancia total que maquinará nuestra herramienta:

Distancia Total = PczT + Compensación.

Donde Pczt será la profundidad de corte programada por el usuario.

Para calcular la compensación de la broca usaremos la fórmula general de compensación dada anteriormente.

Esta distancia total podemos igualarla a la distancia h , que se maquinará con avance f1, más la distancia hf , que se maquinará con avance f2. La distancia h será maquinada en picoteo , la distancia hf será maquinada con avance recto.

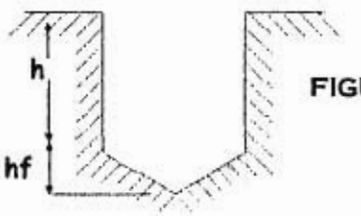


FIGURA 28

De donde:

PczT + Compensación = h +hf

Por lo que:

h = PczT +Compensación – hf

El operador ingresará en nuestra macro una profundidad de corte en picoteo , la cual llamaremos Pcz . Para determinar las n veces que nuestra herramienta " picoteará " en el barreno dividiremos la ecuación anterior entre Pcz.

$$nz = \mathrm{Int}\left[\frac{PczT + \text{Compensación} - hf}{Pcz}\right]$$

Int especifica la parte entera de la ecuación.

Tomamos la parte entera de la ecuación, por si el resultado de dividir nuestra ecuación entre la profundidad de penetración ingresada tiene un residuo.

Dicho residuo será calculado por la siguiente ecuación:

$$n\text{residuo } z = Pcz \times \mathrm{Fracc}\left[\frac{PczT + \text{Compensación} - hf}{Pcz}\right]$$

Donde Fracc implica la parte fraccionaria de la ecuación.

Ahora, será necesario pedir al operador una Z de llegada. Esta Z establece la coordenada inicial en este eje en la cual empezaremos a maquinar el barreno. Dependiendo del cero pieza asignado por el operador esta Z de llegada puede ser positiva, negativa o incluso cero como se muestra en la figura.

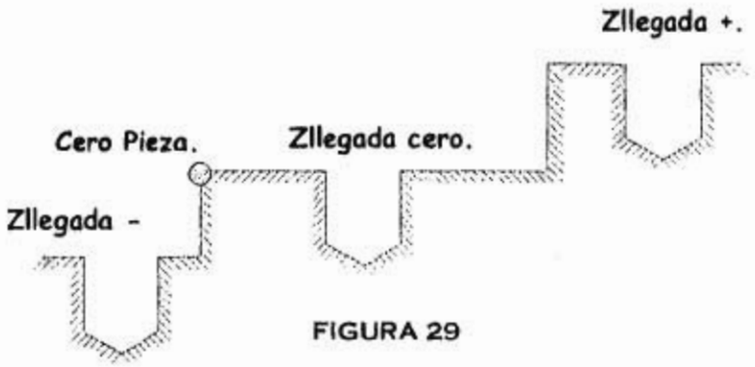


FIGURA 29

Nuestra broca se introduce n veces a partir de Z llegada, la ecuación que define este movimiento es:

G01 a : Z = (Zllegada – n x Pcz) F = f1

Indicamos en esta ecuación el avance (F) f1, puesto que estamos maquinando la distancia h de picoteo.

Para efectuar el picoteo, regresamos en rápido la distancia dada por el parámetro E12 por lo que:

G00 a: Z = Z llegada – n x Pcz + E12

Será la ecuación de retorno al punto definido por E12.

Hemos definido ya las ecuaciones para maquinar nuestra distancia h . Ahora debemos considerar las ecuaciones que maquinarán la distancia de residuo en Z , en caso de que dicho residuo exista.

Si existe residuo entonces:

G01 a : Z = (Z llegada – nz x Pcz – residuoz) F = f1

En esta ecuación n vale ya nz puesto que estamos seguros que hemos recorrido la distancia h entera.

Nuestro avance sigue siendo f1 , puesto que estamos maquinando el residuo de la distancia h.

El retorno del picoteo lo podemos calcular como:

G00 a: Z = Zllegada – nz x Pcz – residuoz

Ahora , definamos la ecuación que nos de el movimiento para hf, esta distancia deberá ser ingresada por el operador en nuestra macro.

Si hf es diferente de cero entonces:

G01 a : Z = (Z llegada – nz x Pcz – residuoz - hf) F = f2

Fíjate que para esta ecuación , no importa que el valor del residuo en z haya sido cero . También observa que ya incluimos un avance f2 .

Programaremos la pausa al final del maquinado con un parámetro al cual llamaremos E30 .

G04 X E30 .

Finalmente la herramienta saldrá en rápido del barreno, ya sea al plano R o a nuestra Z inicial.

Podemos resumir ya el algoritmo de este ciclo como sigue:

A partir del diámetro y del ángulo de la herramienta ingresada por el operador, calculamos la compensación de la broca con la fórmula dada anteriormente .

Calculamos las veces que nuestra herramienta penetrará al barreno por medio de la siguiente ecuación:

$$nz = \text{Int}\left[\frac{PczT + \text{Compensación} - hf}{Pcz}\right]$$

Si esta ecuación tuviera residuo, lo calcularemos por medio de la siguiente ecuación

$$\text{residuo } z = Pcz \times \text{Fracc}\left[\frac{PczT + \text{Compensación} - hf}{Pcz}\right]$$

1. La herramienta se mueve en rápido al centro de nuestro barreno en XY, simultáneamente movemos la herramienta a Z inicial, instalando compensación de altura.
2. Movemos la herramienta en Z al plano R.
3. Maquinamos la distancia de picoteo (h) desde n = 1 hasta nz , retrocediendo la distancia de picoteo E12, utilizando de manera conjunta las siguientes ecuaciones:
   G01 a : Z = (Zllegada – n x Pcz) F = f1 (Ecuación para maquinado).
   G00 a: Z = Z llegada – n x Pcz + E12 (Ecuación para retroceso).
   E12 será el parámetro que defina la distancia de picoteo.
4. Si existe residuo entonces emplearemos las siguientes ecuaciones para maquinar la distancia restante:
   G01 a : Z = (Z llegada – nz x Pcz – residuoz) F = f1
   G00 a: Z = Zllegada – nz x Pcz – residuoz

5. Para maquinar la distancia hf empleamos la siguiente ecuación:

   G01 a : Z = (Z llegada – nz x Pcz – residuoz - hf) F = f2

6. Comandamos pausa en el maquinado G04 X E30 (E30 será un parámetro para definir la pausa al final del maquinado).
7. Finalmente retiramos la herramienta del barreno, ya sea al plano r o a Z inicial.

Con esto hemos definido por completo nuestro algoritmo para ciclo de picoteo rápido.

El programa en PASCAL que continene nuestro algoritmo es el siguiente:

```
const
D41=0.1;        { parámetro para definir claro de llegada en z }
S4=5E-4;        { parámetro máquina para definir punto flotante }
E12=0.01;       { parámetro para definir retorno rápido de picoteo}
Zinicial=1.0; { definiendo llegada de herramienta }
E30 =0.5;       { parámetro de pausa de maquinado en el fondo}
var
archivo:text;

procedure pecking2(Xini,Yini,Diatool,Pcztotal,Pcz,ToolAngle,Zllegada,
F1,F2,hf:real;Xherramienta,retorno:shortint;var arctrampa:text);
var
residuoz,compensacion:real;
n,nz:integer;
begin
If ((0<(ToolAngle*pi/180)) and ((ToolAngle*pi/180)< pi) then
 compensacion:=Diatool/( (2*tan(ToolAngle*pi/360))+(0.3172*Diatool))
else if ((ToolAngle*pi/180)=pi)then
 compensacion:=0.3172*Diatool
 else
  compensacion:=0;
assign(arctrampa,'Taladrado.fil');
rewrite(arctrampa);
writeln(arctrampa,'(PECKING 2)');
writeln(arctrampa,'G00 X',xini:6:4,'Y',yini:6:4,'G43 H0',Xherramienta,'Z',Zinicial:6:4);
writeln(arctrampa,'G00 Z',(D41+Zllegada):6:4);
close(arctrampa);
nz:=trunc((Pcztotal+Compensacion - hf)/Pcz);
residuoz:=puntoflotante(Pcz*(frac((Pcztotal+Compensacion -hf)/Pcz)));
 For n:=1 to nz do
  begin
   append(arctrampa);
    writeln(arctrampa,'G01 Z',(Zllegada-(n *Pcz)):6:4,'F',f1:6:4 );
    writeln(arctrampa,'G00 Z',(Zllegada-(n *Pcz)+ E12):6:4);
    close(arctrampa);
   end;
If residuoz <> 0 then
 begin
  append(arctrampa);
  writeln(arctrampa,'G01 Z',(Zllegada-(n *Pcz)-residuoz):6:4,'F',f1:6:4 );
  writeln(arctrampa,'G00 Z',(Zllegada-(n *Pcz)-residuoz+ E12):6:4);
  close(arctrampa);
  end;
If hf <>0 then
 begin
  append(arctrampa);
  writeln(arctrampa,'G01 Z',(Zllegada-(n *Pcz)-residuoz -hf):6:4,'F',f2:6:4 );
  close(arctrampa);
 end;
If retorno=1 then
 begin
  append(arctrampa);
  writeln(arctrampa,'G04 X',E30:6:2);
  writeln(arctrampa,'G00 Z',Zinicial:6:4);
```

```
  close(arctrampa);
 end
else if retorno=0 then
 begin
  append(arctrampa);
  writeln(arctrampa,'G00 Z',(D41+Zllegada):0:4);
  close(arctrampa);
 end;
end;
```

Veámos como funciona nuestra macro con el siguiente ejemplo.

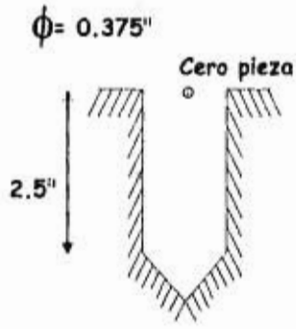


FIGURA 30

De la figura observamos la profundidad del barreno, el diámetro del mismo y la ubicación del cero pieza.

Utilizaremos broca de carburo con ángulo de 120°. La profundidad de corte antes de cada picoteo será 0.500". De la regla anterior que dimos para establecer la distancia de cambio de avance (hf) obtenemos un valor de 0.1875"

Los avances f1 y f2 serán de 20 y 12 inch. / min. respectivamente.

Ingresando estas variables a nuestra macro obtenemos el siguiente código de maquinado:

```
(PECKING 2)
G00 X0 Y0 G43 H01 Z1.
G00 Z0.1
G01 Z-0.5     F 20
G00 Z-0.49
G01 Z-1.      F 20
G00 Z-0.99
G01 Z-1.5     F 20
G00 Z-1.49
G01 Z-2.      F 20
G00 Z-1.99
G01 Z-2.4199 F 20
G00 Z-2.4099
G01 Z-2.6074 F 12
G04 X  0.50
G00 Z1.0000
```

El cual desarrolla el taladrado en picoteo rápido del barreno.

Diseño de tu propio ciclo enlatado:

Si has tenido experiencia manejando centro de maquinado te habrás topado con seguridad con ciclos enlatados o de repetición. Estos ciclos vienen dados por comandos que representan una

operación específica: taladrado, machueleado, mandrinado y en general , todas las operaciones de corte relativo a un centro.

Son particularmente útiles cuando maquinamos ciertos patrones repetitivos de maquinado, permitiendo reducir de manera considerable la longitud de nuestro programa.

En el comando de ciclo enlatado típico, especificamos los datos relativos al maquinado (tipo de ciclo, avance, profundidad, diámetro final, etc..) para nuestro primer agujero; posteriormente simplemente listamos las coordenadas siguientes de los agujeros que serán maquinados, el control desplazará entonces la herramienta a cada posición referida y realizará el maquinado exactamente igual como fue especificado para el primer barreno. Al finalizar, tenemos que cancelar el ciclo por medio del comando G80.

Hay dos reglas básicas que se aplican a todos los ciclos enlatados:

La primera es que todos son de carácter modal, esto es una vez que instalamos un ciclo enlatado en nuestro control, éste permanece en efecto hasta que es cancelado.

La segunda es referida a la mecánica del movimiento que sigue la herramienta:

- La herramienta se posiciona primero en el plano XY con movimiento rápido.
- Con movimiento rápido nos movemos en Z hasta nuestro plano R.
- El agujero es maquinado de acuerdo al ciclo seleccionado.
- La herramienta es retirada del barreno .

Si te das cuenta, hemos seguido este patrón de posicionamiento de herramienta, para todas las macros que hemos diseñado hasta ahora . Esto ha sido con la idea de poder diseñar nuestro propio ciclo de repetición aplicando estas mismas macros.

Un ciclo enlatado, en realidad usa solamente los comandos básicos de programación , representando en sí mismo una macro.Los ciclos que se encuentran en tu control, son diseñados por el fabricante del mismo. Recuerda que una de las aplicaciones fuertes de la macro programación es el permitir diseñar códigos G y M propios. Así el fabricante utilizando macro programación genera sus ciclos enlatados, mismos que te presenta despúes con nombres como G73, G84, G85, etc..

La única diferencia entre el código generado por el fabricante y el que hemos generado hasta ahora, es la primera regla básica mencionada anteriormente. El fabricante introduce un código especial que permite que su macro instrucción se convierta en modal. Esto es extremadamente útil, dado que son códigos que utilizamos por lo regular no en uno, sino en varios barrenos, ayudando cuando generamos un patrón básico de mecanizado.

Nuestras macros tienen la desventaja de que han sido calculadas en forma puntual, esto es, para un solo punto. Sería una tarea ardua, por ejemplo si tenemos un patrón de cincuenta barrenos dispuestos en círculo y debemos ingresar en nuestra macro de taladrado todas las coordenadas de los barrenos de dicho círculo.

Esto nos sugiere una idea y nos aproxima al concepto de comando modal usado en CNC. Buscaremos modelar patrones básicos de mecanizado, los cuales generen las coordenadas necesarias para el maquinado y posteriormente introduciremos tantas veces, como barrenos queramos mecanizar, dichas coordenadas en nuestras macros.

Existen un buen número de patrones básicos de mecanizado . A continuación mostramos sólo unos cuantos de ellos:

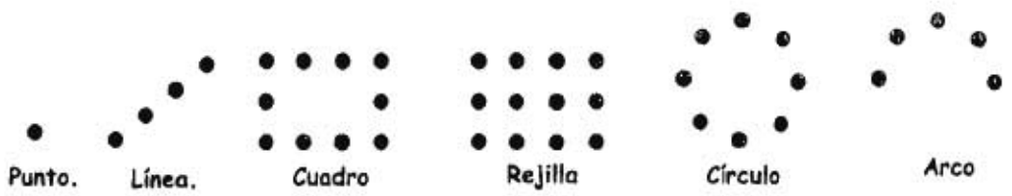


FIGURA 31

Por razones de espacio, modelaremos solamente el patrón de mecanizado en círculo y el de línea.

Patrón de mecanizado en Línea:

¿Cómo comenzamos a modelar un patrón de mecanizado ?. Primero debemos pensar en todas las variables que debemos incluir para que el operador pueda definir por completo la geometría deseada .

Sabemos que una línea está definida por dos coordenadas, pero también por una coordenada y un ángulo. Esta última forma es la que se acostumbra utilizar en los planos de mecanizado, por ser la más facil, y será la que usemos en nuestro patrón de mecanizado.

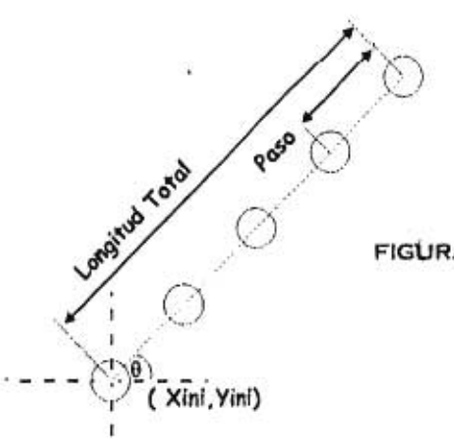


FIGURA 31

Definiremos entonces nuestra línea empezando por una coordenada , a partir de la cual se definirán el resto de las coordenadas que conforman dicha línea. En la figura 31 este primer punto está indicado como Xini, Yini.

Usualmente el ángulo que posee una línea esta acotado con respecto a una horizontal, por lo tanto, nuestro ángulo estará tomado a partir del eje X ; convención que seguiremos para cualquier ángulo que necesite ser introducido por el operador en las macros que diseñemos.

¿ Hasta donde se prolongará la línea ? , para contestar esta pregunta debemos preguntar al operador ya sea la longitud total o el paso entre barrenos. La fórmula que relaciona estos dos datos es la siguiente:

$$\text{Paso} = \frac{\text{Longitud Total}}{\text{No. agujeros} - 1}$$

Lo que nos hacer pensar que también necesitamos conocer el número de agujeros de la línea.

Habiendo definido ya todas las variables que necesitamos para modelar nuestra línea , es tiempo precisamente de comenzar a modelarla.

De la figura 31 observamos que se forma un triángulo rectángulo para cada barreno. Cada triángulo aumenta su hipotenusa en el paso con respecto al anterior. Procedamos a escribir unas cuantas coordenadas para ver esto con mayor claridad:

Para el primer barreno sus coordenadas serían:

Xini , Yini. (En este caso la hipotenusa vale cero).

Para el segundo barreno tenemos:

Xini + Paso x cos θ , Yini + Paso x sin θ

Para nuestro tercer barreno:

Xini + 2 Paso x cos θ , Yini + 2 Paso x sin θ

De manera general podemos escribir las siguientes ecuaciones:

$x = Xini + Paso \times (n - 1) \times \cos\theta$

$y = Yini + Paso \times (n - 1) \times \sin\theta$ Para n = 1 hasta n = No. agujeros.

Que son las ecuaciones que modelan las coordenadas en nuestro patrón de mecanizado de línea.

Patrón de mecanizado en Círculo:

Ahora modelaremos nuestro patrón de maquinado en círculo.Pensemos en las variables que tendremos que usar para definir nuestra geometría, el centro del círculo, el radio del mismo y el número de barrenos, parece ser todo lo que necesitamos.

Podemos también dar al operador la opción de escojer el ángulo del barreno inicial a maquinar; esto es, con la idea de hacer más flexible nuestro patrón de mecanizado.

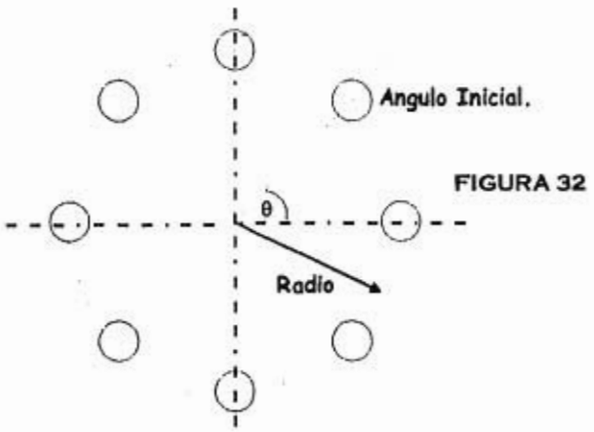


FIGURA 32

El ángulo entre los barrenos estará dado por la siguiente relación:

$$\theta = \frac{360^{\circ}}{\text{No. Agujeros}}$$

De la figura observamos que lo que aumenta entre barrenos es el ángulo θ con respecto al ángulo inicial.

Las ecuaciones que definirán el patrón de círculo empezando desde el primer barreno hasta el último serán entonces:

x = Xini + Radio cos (Anini)

y = Yini + Radio sen (Anini)

Anini = Anini + θ

Donde Anini será el ángulo inicial del primer barreno a maquinar. Cada vez que calculemos un nuevo barreno , incrementaremos en θ nuestras ecuaciones.

Los procedimientos en PASCAL que generan las coordenadas tanto para el patrón de línea como el patrón de círculo son los siguientes:

```
procedure macrolinea(xini,yini,tetal,Paso:real;Nho:integer;var arreglox,arreglo:arraypasal;
var
Dtx:real;
xca:integer;
begin
 Dtx:=0;
 For xca:=1 to Nho do
```

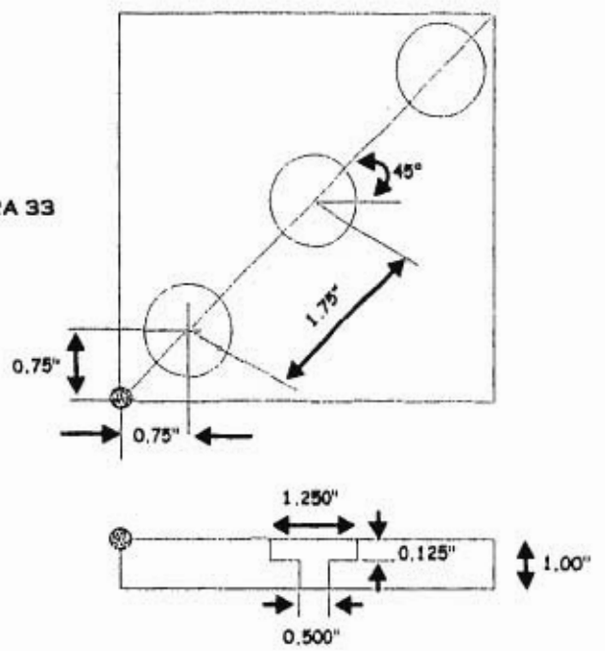
```
  begin
   arreglox[xca]:=Xini+Dtx*cos(tetal*pi/180);
   arregloy[xca]:=Yini+Dtx*sin(tetal*pi/180);
   Dtx:=Dtx+paso;
  end;
end;


procedure macrocirc(xini,yini,Aini,radio:real;Nho:integer;var arrehlox,arrehloy:arraypasa);
var
k:shortint;
teta,tetal:real;
begin
teta:=2*pi/Nho;
tetal:=Aini*pi/180;
For k:=1 to Nho do
 begin
  arrehlox[k]:=xini+radio*cos(tetal);
  arrehloy[k]:=yini+radio*sin(tetal);
  tetal:=tetal+teta
 end;
end;
```

Ambos procedimientos nos generan un arreglo de coordenadas, el cual pasamos a las macros de mecanizado, ya diseñadas por nosotros. Por cada coordenada ingresada a la macro de mecanizado ésta genera el código requerido hasta completar nuestro patrón de mecanizado.
Por razones de espacio, ejemplificaremos únicamente el patrón de mecanizado en línea.

Ejemplo para mecanizado en línea:

Utilizaremos la siguiente figura a manera de ejemplo.

FIGURA 33



Este procedimiento para patrón en línea será utilizando en conjunto la macro de taladrado por picoteo rápido con la de fresado por *Spiral Milling*.

El diámetro de la broca será de 0.500" . Este será el prediámetro para el maquinado con *Spiral Milling* el cual rectificará el barreno a 1.250" . para esta operación elegimos un cortador de 15/32.

Nota que el cero pieza se encuentra en la esquina inferior izquierda de nuestra figura.

Resumamos nuestras condiciones del mecanizado.

Para taladrado:

Broca de 0.500" con ángulo de 120°.

Para la caja:

Prediámetro de 0. 500". Diámetro final 1.250" . Profundidad de corte radial 0.125".

Con esto obtenemos el siguiente código: (Observa el código de formato que se ha insertado).

```
G91 G28                          Z 0
G28 X 0        Y 0
G90 G54 G49 G20 G94
T01 M06
(PECKING 2)
G00 X 0.7500  Y 0.7500 G43 H01 Z 1.0000  S450 M03 M08  ( Primer Barreno)
G00                            Z 0.1000
G01                            Z-0.5000  F10.0000
G00                            Z-0.4900
G01                            Z-1.0000  F10.0000
G00                            Z-0.9900
G01                            Z-1.1380  F10.000
G04 X  0.50
G00                            Z 0.1000
G00 X 1.9874  Y 1.9874                                ( Segundo Barreno )
G01                            Z-0.5000  F10.0000
G00                            Z-0.4900
G01                            Z-1.0000  F10.0000
G00                            Z-0.9900
G01                            Z-1.1380  F10.0000
G04 X  0.50
G00                            Z 0.1000
G00 X 3.2249  Y3.2249                                 ( Tercer Barreno )
G01                            Z-0.5000  F10.0000
G00                            Z-0.4900
G01                            Z-1.0000  F10.0000
G00                            Z-0.9900
G01                            Z-1.1380  F10.0000
G04 X  0.50
G00                            Z 0.1000
G00                            Z 1.00    M09
G91 G28                        Z 0       M19
T02 M06
(SPIRAL MILL)
G00 X 0.7500 Y 0.7500 G43 H2   Z 0.1000 S 900  M03 M08 ( Primer Fresado )
G01                            Z-0.2500 F15.0000
G01 X 0.4844 Y 0.7500                   F15.0000
```

```
G02 X 1.1406 Y 0.7500          I 0.3281
G02 X 0.3594 Y 0.7500          I-0.3906
G02 X 1.2656 Y 0.7500          I 0.4531
G02 X 0.2344 Y 0.7500          I-0.5156
G02 X 1.3906 Y 0.7500          I 0.5781
G02 X 0.1094 Y 0.7500          I-0.6406
G02 X 1.5156 Y 0.7500          I 0.7031
G02 X-0.0156 Y 0.7500          I-0.7656
G02 X 1.6406 Y 0.7500          I 0.8281
G02 X-0.1406 Y 0.7500          I-0.8906
G02 X 1.7656 Y 0.7500          I 0.9531
G02 X-0.2656 Y 0.7500          I-1.0156
G02 X 1.7656 Y 0.7500          I 1.0156
G02 X 1.4531 Y 0.4375          I-0.3125
G00 X 0.7500 Y 0.7500
G00                            Z 0.1000
G00 X1.9874Y1.9874                                  ( Segundo Fresado )
G01                            Z-0.250 F15.0000
G01 X 1.7218 Y 1.9874                  F15.0000
G02 X 2.3780 Y 1.9874          I 0.3281
G02 X 1.5968 Y 1.9874          I-0.3906
G02 X 2.5030 Y 1.9874          I 0.4531
G02 X 1.4718 Y 1.9874          I-0.5156
G02 X 2.6280 Y 1.9874          I 0.5781
G02 X 1.3468 Y 1.9874          I-0.6406
G02 X 2.7530 Y 1.9874          I 0.7031
G02 X 1.2218 Y 1.9874          I-0.7656
G02 X 2.8780 Y 1.9874          I 0.8281
G02 X 1.0968 Y 1.9874          I-0.8906
G02 X 3.0030 Y 1.9874          I 0.9531
G02 X 0.9718 Y 1.9874          I-1.0156
G02 X 3.0030 Y 1.9874          I 1.0156
G02 X 2.6905 Y 1.6749          I-0.3125
G00 X 1.9874 Y 1.9874
G00                            Z 0.100
G00 X 3.2249 Y 3.2249                               ( Tercer Fresado )
G01                            Z-0.2500 F15.0000
G01 X 2.9593 Y 3.2249                   F15.0000
G02 X 3.6155 Y 3.2249          I 0.3281
G02 X 2.8343 Y 3.2249          I-0.3906
G02 X 3.7405 Y 3.2249          I 0.4531
G02 X 2.7093 Y 3.2249          I-0.5156
G02 X 3.8655 Y 3.2249          I 0.5781
G02 X 2.5843 Y 3.2249          I-0.6406
G02 X 3.9905 Y 3.2249          I 0.7031
G02 X 2.4593 Y 3.2249          I-0.7656
G02 X 4.1155 Y 3.2249          I 0.8281
G02 X 2.3343 Y 3.2249          I-0.8906
G02 X 4.2405 Y 3.2249          I 0.9531
G02 X 2.2093 Y 3.2249          I-1.0156
G02 X 4.2405 Y 3.2249          I 1.0156
G02 X 3.9280 Y 2.9124          I-0.3125
G00 X 3.2249 Y 3.2249
G00                            Z 0.100
G00                            Z 1.0000 M09
G91 G28 Z 0  G49
G28 X 0 Y0
```

M30

Este ejemplo muestra completamente la poderosa herramienta que es la macro programación. Date cuenta la cantidad de código que se genera solamente al rellenar variables en nuestro macro programa, del trabajo que nos hemos ahorrado, además de las posibles equivocaciones en que hubieramos pudido incurrir si realizamos el trabajo a mano.

Con esto concluimos la deducción de los patrones de mecanizado de círculo y línea. Posiblemente estés un poco desilusionado, al ver lo sencillo que ha sido el modelarlos. Más , deberías estar sorprendido , ya que en muchos controles modernos que se programan en EIA – ISO, estos patrones de mecanizado se ofrecen como opción (con un costo extra claro está) para complementar los ciclos enlatados que ya vienen con la máquina. Admito que hemos modelado patrones muy simples , - el modelado de rejilla o de cuadro requiere un poco más de esfuerzo - , pero ha sido para que te des cuenta de lo fácil que es hacerlo y que este trabajo no refleja en absoluto el costo que normalmente deberías pagar por tener esta opción en tu control.

CAPITULO 4

Diseñando macroprogramas para superficies.

En este capítulo diseñaremos macroprogramas que generen código de maquinado para una superficie. Con superficie queremos abarcar todas aquellas geometrías cuyo código de maquinado no lo obtengamos refiriéndonos a su centro geométrico, como en el capítulo anterior. Necesitaremos identificar ahora para cada uno de los ejemplos , las coordenadas que nos ayuden a generar nuestro código.

Veremos cómo se diseña un ciclo de careado , tanto para una superficie rectangular como circular, generaremos código de maquinado para un *slot* recto, para un *slot* curvo y desarrollaremos varias soluciones para que las ecuaciones deducidas por nosotros generen código correcto, no importando donde el operador haya decidido tomar el cero pieza.

Notarás que aumenta el grado de complejidad de los ejemplos, tanto en la forma de definir el trazo de la herramienta, como en la matemática que generemos. Aumentando también el espacio de texto dedicado a la solución de cada macro. Es por ello que lamentablemente no podemos abarcar en este capítulo tantas cosas como quisierámos. Pero quizas podrás utilizar las ideas expuestas aquí para auxiliarte a resolver un problema concreto o plantear una macro de tu interés.

Diseño de un Ciclo de Careado:

Pocas operaciones son tan comunes en el taller como el careado o rectificado de una superficie. Prácticamente no hay pieza que no empiece su proceso de mecanizado mediante un rectificado, lo cual nos hace pensar que sería muy conveniente tener en nuestras macros, una macro que generara código para careado.

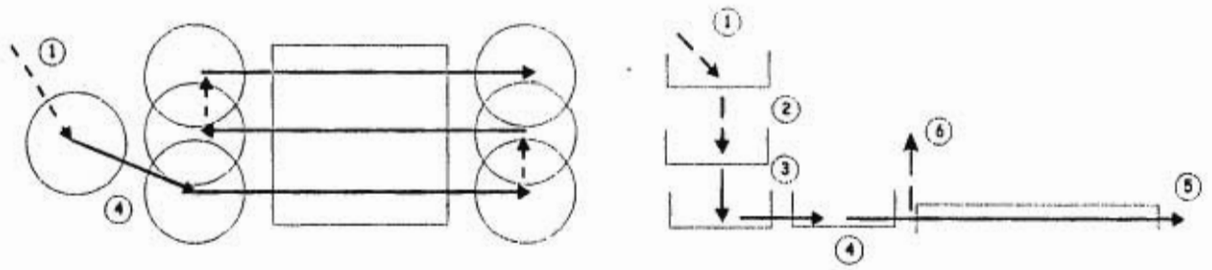
El rectificado consiste en asentar una superficie mediante una fresa planeadora o piña que corta el material con los insertos colocados en ella. Para rectificar superficies pequeñas o donde existe interferencia, será más conveniente emplear un cortador recto.

Este proceso, aparentemente sencillo, requiere un breve análisis para que podamos desarrollar un *tool path* inteligente.

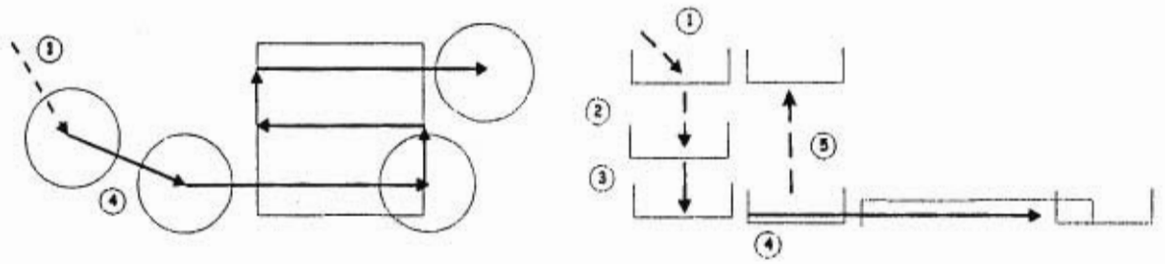
Hay numerosas formas en las cuales el cortador puede entrar a rectificar la pieza. Elegiremos una en base a varios factores: en base a la forma en que dispongamos la sujección de la pieza, sea para evitar interferencia con los *clamps* o sujetadores o con la misma pieza ; para dirigir las fuerzas de corte hacia los sujetadores; para lograr una dirección especificada de acabado; por el tipo de material que estamos trabajando o por el acabado pedido, etc..

En las siguiente figuras podrás observar varias maneras en que se puede entrar a una pieza para rectificarla: (Los números indican la secuencia de operación).

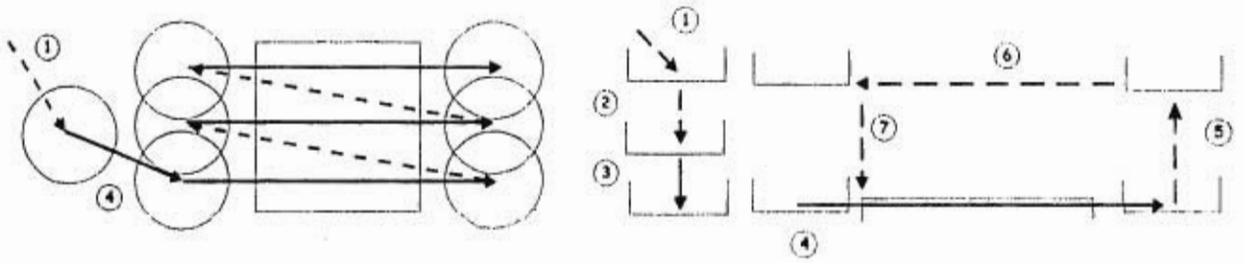
Movimiento rápido.
Avance con corte



Patrón de Mecanizado para Rectificado Bidireccional en X



Patrón de Mecanizado para Rectificado Bidireccional corto en X



Patrón de Mecanizado para Rectificado Unidireccional en X

FIGURA 1

Como te darás cuenta, estamos mostrando solamente patrones de mecanizado en dirección X; sin embargo , los ejemplos anteriormenle descritos también pueden mecanizarse en dirección Y, por

lo que tenemos Y Bidireccional, Y Unidireccional y Y Bidireccional corto, por mencionar solamente algunos.

Necesitamos ahora conocer un poco más de la mecánica de corte del rectificado para poder empezar a programar nuestra macro:

Relación entre el diámetro del cortador y su posicionamiento para corte:

El espesor de corte que va a remover la herramienta por inserto, se le llama capa no deformada de corte.

La posición y el tamaño del cortador determina el espesor de la capa no deformada de corte a la entrada del inserto. Si posicionamos nuestro cortador de modo que no sobresalga del ancho de corte, como se ve en la figura 2 , la capa de corte no deformada a la entrada de inicio del corte por el inserto es teóricamente cero. Recordando que el arranque por viruta es una deformación plástica , tendremos que el inserto maquinará conforme avanza el corte en una superficie más dura , dureza causada por la misma deformación del mecanizado, por lo que la vida de nuestro inserto será muy pobre. Más aún, si una rebaba se llegase a atorar en el inserto, será probable que se adhiera entre el inserto y la superficie maquinada, sin posibilidad para que el cortador pueda botarla, pudiendo fracturar la orilla de nuestro inserto.

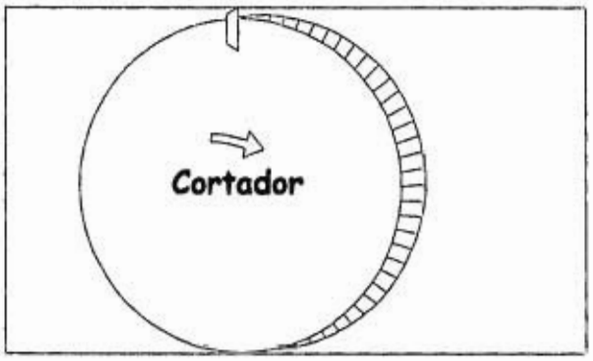


FIGURA 2

Para lograr la máxima vida de nuestra herramienta, nos debemos asegurar de que la capa no deformada de corte a la entrada de nuestro inserto, sea lo más grande posible, de modo que el inserto evite la superficie endurecida que resulta de la deformación del corte. Evitando también que las rebabas se atoren entre el inserto y la superficie mecanizada.

La posición del cortador afecta también el ángulo de entrada del inserto; este ángulo nos ayuda a determinar si el choque del cortador con la pieza al entrar al corte, es absorbido por una sección fuerte o débil del inserto, como podemos ver en las siguientes figuras:

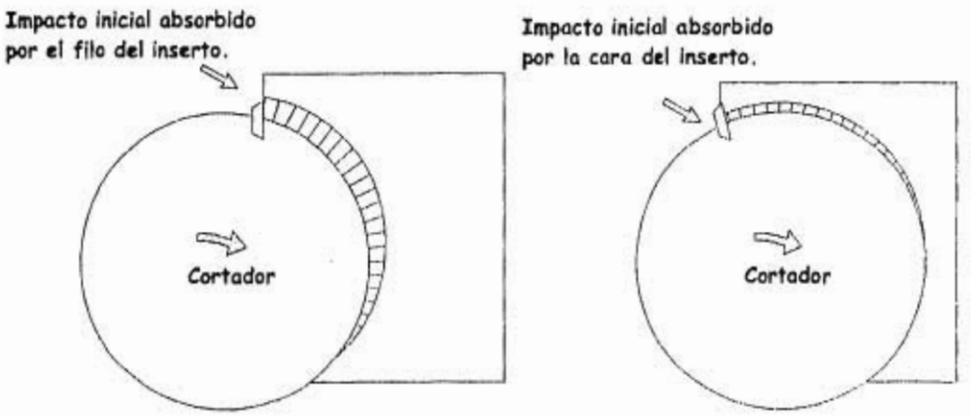


FIGURA 3

En la figura de la izquierda se nota que el impacto de corte inicial ocurre en la orilla cortante del inserto, su punto más débil, donde es más propenso a astillarse o a romperse. La figura de la derecha, muestra al inserto absorbiendo el impacto en la cara (una sección más fuerte). Viendo las dos figuras anteriores podemos deducir , en general, que *siempre que la línea de centros del cortador pase a través de la pieza de trabajo, el ángulo de entrada del inserto será favorable; además de que obtendremos la máxima capa de corte no deformada.Para lograr esto por lo general, deberemos posicionar el cortador de modo que su diámetro sobresalga entre 1/4 o 1/3 de nuestra pieza de trabajo.*

En conclusión a lo anterior, podemos pensar que un dato importante de entrada de nuestra macro será la profundidad de corte radial, ya sea dada en número o en porcentaje de corte, de modo tal que el usuario pueda especificar el porcentaje de corte deseado con el diámetro del cortador elegido.

Conociendo los puntos de aproximación:

Además de conocer cómo posicionar nuestro cortador a la entrada del mecanizado para obtener un ángulo favorable de corte, debemos colocarlo antes, en una posición segura de modo que el cortador baje en Z, se acomode en XY y vaya al punto de inicio de corte. Esto, porque el cortador (y en general cualquier herramienta de rectificado), están diseñados para soportar esfuerzo radial

no axial, es decir , no podemos bajar con nuestro cortador directamente en Z sobre nuestra pieza de trabajo y comenzar el maquinado , pues con seguridad la herramienta se rompería .
Debemos asegurarnos además, que el cortador libre cualquier obstáculo que encuentre en su camino a la pieza, sean sujetadores, dispositivos, etc..
Si observas, en la figura 1 que muestra diversos patrones de rectificado, el punto 3 representa nuestro punto de aproximación y el punto 4 nuestro punto de inicio de corte. Por lo general, siempre que programemos un cortador para maquinar una superficie seguiremos esta ruta de mecanizado.

Diseñando la macro de careado para superficie rectangular:

Al conocer ya un poco más de la mecánica de corte del fresado, podemos proceder a programar la macro del careado.
Por razones de espacio desarrollaremos únicamente la macro de rectificado bidireccional en X .
En la figura 4 que ilustra el mecanizado para este tipo de rectificado, los números servirán para el desarrollo de la secuencia de los pasos a programar:

1. Con movimiento rápido, desplazamos la herramienta por el plano XY hasta nuestro punto de aproximación .Bajamos rápidamente a Z inicial, instalando compensación de longitud de la herramienta.
2. Nos movemos , igualmente en rápido, a la altura de claro en Z o plano R .
3. Con avance nos movemos hasta la profundidad de corte.
4. Avanzamos en XY a nuestro posición de inicio de corte o punto de claro radial.
5. Recorremos la pieza con el cortador hasta librarla, salimos al punto de claro radial, avanzamos nuestro cortador en Y a profundidad de corte o a porcentaje de corte y repetimos la operación a la inversa hasta completar el maquinado.
6. Salimos con el cortador en Z al punto R. Si es necesario, repetimos el punto 1 para posicionarnos en XY y repetimos la mecánica de corte.

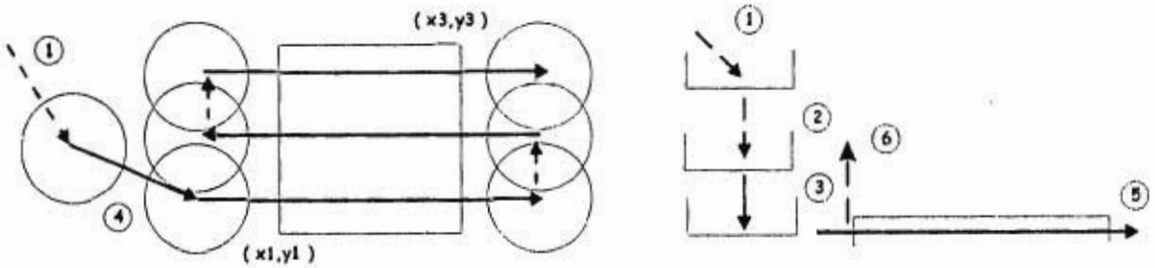


FIGURA 4

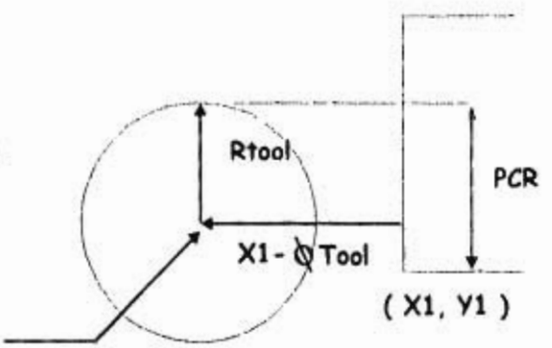
De la figura 4, te podrás dar cuenta que hemos marcado dos coordenadas, las únicas que necesitamos para definir nuestra geometría; dichas coordenadas deberán ser introducidas por el operador.

Sea, AproX y AproY nuestras coordenadas de aproximación en X y Y respectivamente .

Definamos de manera arbitraria AproX = X1 - ØTool. Con esto aseguramos que la herramienta posee el espacio necesario para bajar en Z con seguridad, pero a la vez lo bastante cerca de la pieza para no desperdiciar tiempo de maquinado.

Para deducir AproY , es necesario examinar la figura 5:

FIGURA 5



Coordenadas de centro de herramienta

AproY = Y1+ Pcr –Rtool Donde Pcr será igual a la profundidad de corte radial.

Como mencionamos anteriormente, además de permitirle al usuario el especificar directamente la profundidad de corte radial, debemos permitir que introduzca como opción en lugar de ésta, el porcentaje de corte que desee; por lo tanto, también debemos deducir la ecuación que nos de la coordenada Y de la posición del centro del cortador en función del porcentaje de corte y del diámetro de la herramienta.

Sea Porcentaje de Corte = Pcr / ØTool

Por lo que : (Porcentaje de Corte) x (ØTool) = Pcr

Sustituyendo en la ecuación que define AproY tenemos:

AproY = (Porcentaje de Corte) x (ØTool) – Rtool = 2 x (Porcentaje de Corte) x (RTool) – Rtool

Factorizando llegamos a:

AproY = Y1 + (ØTool) x (Porcentaje de Corte – ½) , que es la ecuación deseada.

Con estas ecuacciones hemos definido las coordenadas de aproximación a la pieza en el plano XY; deduciremos a continuación las coordenadas marcadas con el número 4 . El punto de inicio de corte.

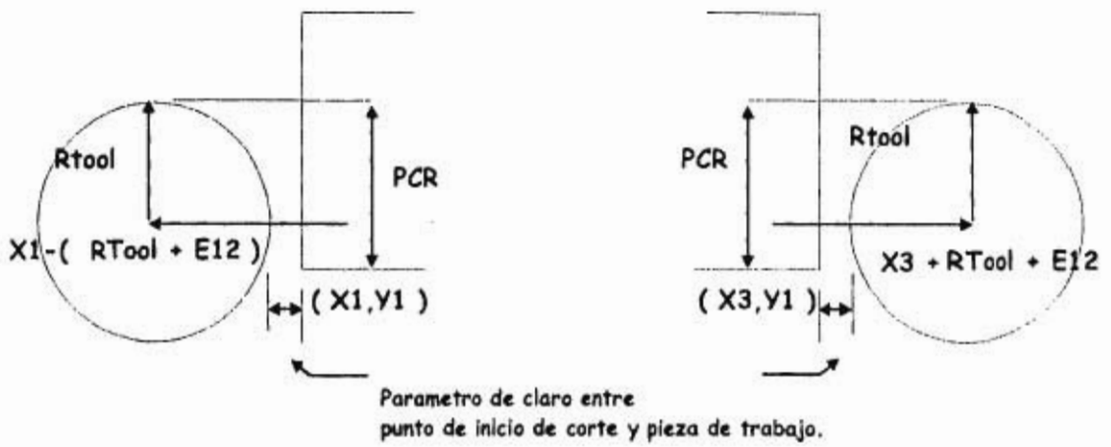


FIGURA 6

Nuestra coordenada en X será ahora la posición X1, menos el radio de la herramienta, más un parámetro de claro radial, al cual denominaremos E12, dicho parámetro sirve de cojín entre el cortador y la pieza de trabajo, antes de comenzar el corte.

X inicio corte = X1 – (Rtool + E12)

Yinicio corte = AproY = Y1 + Pcr – Rtool , o definiendolo en porcentaje de corte :

Yinicio corte = Y1 + (∅Tool) x (Porcentaje de Corte – ½)

Nuestra coordenada de salida del cortador será :

X salida = X3 +Rtool +E12

Ysalida = Y1 +Pcr –Rtool = Y1 + (∅Tool) x (Porcentaje de Corte – ½)

Solamente nos falta el plantear una última coordenada y ésta es determinar cuanto aumenta la posición Y, después de que el cortador termina su recorrido en X.

De la figura 7 podemos ver que lo que se repite es Pcr ; deducimos pues, que lo que nuestra coordenada aumenta en Y es n veces Pcr .

N será igual a la distancia entre la coordenada Y3 y Y1, dividido entre la profundidad de corte radial.

De otro modo:

$$n = \frac{ABS(Y3 - Y1)}{Pcr}$$

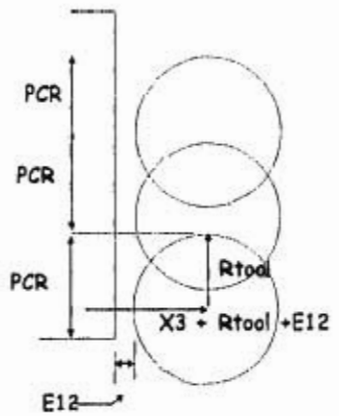


FIGURA 7

Por lo tanto podemos ya plantear nuestro algoritmo:

1. Movemos en rápido nuestra herramienta a AproX = X1 - ØTool. , AproY = Y1+ (n) x Pcr – Rtool , instalando compensación en Z inicial. (N inicialmente vale 1)
2. Bajamos rápidamente al plano R Al cual declararemos como parámetro dentro de nuestra macro .
3. Bajamos con avance a profundidad de corte en Z.
4. Nos movemos a punto de inicio de corte: X inicio corte = X1 – (Rtool + E12); Yinicio corte = Y1 + (n) x Pcr – Rtool.
5. Salimos a Xsalida = X3 + (Rtool +E12) , Ysalida = Y1 + (n) x Pcr - Rtool
6. Incrementamos Ysalida en 2 , 3...n y nos movemos al inciso 4 , porsiguiendo hasta terminar el maquinado.

Si te has dado cuenta, hemos deducido todas las coordenadas de nuestra macro a partir de la esquina inferior izquierda; es decir a partir del primer cuadrante. Más hemos mencionado ya de manera previa, que procuraríamos diseñar nuestras macros de forma que el usuario pudiera especificar el cero pieza en cualquier lugar que él quisiera y aún así generaríamos el código correcto de mecanizado. En nuestras macros anteriores no habíamos tenido problemas al respecto puesto que eran simétricas y su centro geométrico estaba bien definido, (el centro de nuestro barreno) , pero ahora nuestra disposición geométrica es diferente. Encararemos este problema usando algo que se llama Imagen de espejo:

Imagen de Espejo:

La función espejo, es una ayuda ofrecida generalmente por nuestro equipo CNC, la cual por lo regular es mal aplicada o entendida. Aunque en nuestro controlador, dicha función puede tener

limitaciones en lo referente a su uso, procuraremos explicarla, para que una vez comprendida, podamos programar nuestra propia función espejo.

Como el nombre nos lo indica, la función espejo es usada *para generar una serie de movimientos que representan el reflejo de nuestro tool path.* Lo que ocurre cuando usamos esta función, es que el control multiplica por menos el eje sobre el cual decidimos crear el espejo. Es decir, por ejemplo si programamos una posición de X 5.0 y aplicamos la función espejo, sobre el eje X, el control leerá X –5.0. Dependiendo de lo sofisticado del control podremos espejear en uno o dos ejes simultáneamente . Debemos agregar además que bajo la función espejo, G41 se convierte en G42, y G03 se vuelve interpolación horaria, así como G02 antihoraria.

Una vez aclarado el concepto de función espejo, veamos como nos puede ayudar a resolver nuestro problema:

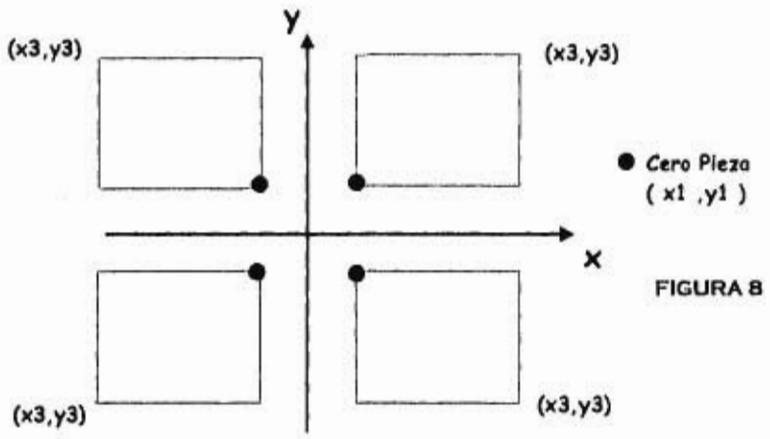


FIGURA 8

De la figura 8 te podrás dar cuenta varias formas en las cuales el operador puede acomodar el cero pieza. Las cuatro que presentamos son las principales. De hecho el operador podrá programar la pieza cruzando los ejes coordenados . Si te fijas la figura en el primer cuadrante, (sobre la que hemos deducido nuestras ecuaciones) se está reflejando en cada eje. Esto nos da una idea para resolver el problema:

Llamemos a dos variables mirrorX y mirrorY. Dichas variables tomaran los siguientes valores cuando se cumplan cualquiera de las siguientes condiciones:

Si X1 < X3 y Y1 < Y3 entonces mirrorX = 1 y mirror Y = 1

Si X1 > X3 y Y1 < Y3 entonces mirrorX = -1 y mirror Y = 1

Si X1 > X3 y Y1 > Y3 entonces mirrorX = -1 y mirror Y = - 1

Si X1 < X3 y Y1 > Y3 entonces mirrorX = 1 y mirror Y = -1

Nuestro algoritmo está ya completo y queda pues como sigue:

1. Movemos en rápido nuestra herramienta a AproX = X1 – [ØTool.] x [mirror X] , AproY = Y1 + [(n) x Pcr –Rtool]x [mirrorY], instalando compensación en Zinicial. (N toma el valor de1)
2. Bajamos rápidamente a posición Z de claro .
3. Bajamos con avance a profundidad de corte en Z.
4. Nos movemos a punto de inicio de corte: X inicio corte = X1 – [Rtool + E12]x [mirrorX]; Yinicio corte = Y1 + [(n) x Pcr – Rtool]x [mirrorY] .
5. Salimos a Xsalida = X3 + [Rtool +E12]x [mirrorX] , Ysalida = Y1 + [(n) x Pcr - Rtool]x [mirrorY]

Incrementamos Ysalida en 2 , 3...n y nos movemos al inciso 4 , porsiguiendo hasta terminar el maquinado.

Programa en PASCAL que genera el código de maquinado para Fresado :

```
const
E9=0.12;{parametro que define el plano R en Z para el corte}
E12=0.100;{parametro que protege al Facemill al acercarse al maquinado}
Zinicial=1; {nuestra Z inicial}
S4=1E-5;{Nuestro punto flotante}
var
archivo:text;

procedure FacemillXBidir(x1,y1,x3,y3,WR,Tooldia,Zllegada,Pcztotal,Pcz,Finz,
Fdesbaste,Facabado:real;SalidaCompleta:shortint;var arctrampa:text);
var
nradial,mirrorx,mirrory,I,K,nz:shortint;
residuoz,Dtz,Ztemp:real;
BanderaRadial:Boolean;
begin
 BanderaRadial:=false;
 nradial:=Trunc((Abs(y3-y1)/WR));
 If Frac((Abs(y3-y1)/WR)) <>0 then
 nradial:=nradial+1;
 if (x1<x3) and (y1<y3) then
  begin
   mirrorx:=1;
   mirrory:=1;
  end
 else if (x1>x3) and (y1<y3) then
  begin
   mirrorx:=-1;
   mirrory:=1;
  end
 else if (x1>x3) and (y1>y3) then
 begin
  mirrorx:=-1;
  mirrory:=-1;
  end
 else if (x1<x3) and (y1>y3) then
  begin
   mirrorx:=1;
   mirrory:=-1;
  end;
nz:=trunc((Pcztotal-Finz)/Pcz);
residuoz:=puntoflotante(pcz*frac((pcztotal-Finz)/Pcz));
```

```
If nz<=0 then
Dtz:=residuoz
else
Dtz:=Pcz;
assign(arctrampa,'Xbidir.Fl)');
rewrite(arctrampa);
writeln(arctrampa,'FMXBIDIR');
writeln(arctrampa,'G00 X',(X1-ToolDia*mirrorx):6:4,'Y',(y1+(WR-
(ToolDia/2))*mirrory):6:4,'Z',(Zinicial+Zllegada):6:4);
writeln(arctrampa,'G00 X',(X1-ToolDia*mirrorx):6:4,'Y',(y1+(WR-
(ToolDia/2))*mirrory):6:4,'Z',(E9+Zllegada):6:4);
writeln(arctrampa,'G00 X',(X1-ToolDia*mirrorx):6:4,'Y',(y1+(WR-
(ToolDia/2))*mirrory):6:4,'Z',(Zllegada-Dtz):6:4);
writeln(arctrampa,'G00 X',(X1-((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+(WR-
(ToolDia/2))*mirrory):6:4);
close(arctrampa);
For K:=1 to nz do
 begin
  append(arctrampa);
  ztemp:=Zllegada-(Dtz*nz);
   If ((nz <> 1) and (BanderaRadial)) then
    begin
write(arctrampa,'G01 X',(X1-((ToolDia/2)+E12)*mirrorx):6:4);
writeln(arctrampa,'Y',(y1+(WR-(ToolDia/2))*mirrory):6:4,'Z',ztemp:6:4,'F',fdesbaste:6:4);
     BanderaRadial:=False;
    end;
     For I:= 1 to nradial do
      begin
       if odd(I) then
        begin
writeln(arctrampa,'G01 X',(X3+((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+(I*WR-
(ToolDia/2))*mirrory):6:4,'F',fdesbaste:6:4);
If  (I<> nradial) then
writeln(arctrampa,'G01 X',(X3+((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+((I+1)*WR-
(ToolDia/2))*mirrory):6:4,'F',fdesbaste:6:4);
        end
        else
        begin
writeln(arctrampa,'G01 X',(x1-((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+(I*WR-
(ToolDia/2))*mirrory):6:4,'F',fdesbaste:6:4);
writeln(arctrampa,'G01 X',(x1-((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+((I+1)*WR-
(ToolDia/2))*mirrory):6:4,'F',fdesbaste:6:4);
        end;
     BanderaRadial:=True;
      end;
 If Salidacompleta= 1 then
    begin
     writeln(arctrampa,'G01 Z',(Zllegada+E9):6:4);
     writeln(arctrampa,'G00 Z',(Zllegada+Zinicial):6:4,'F',Fdesbaste:6:4);
     writeln(arctrampa,'G00 X',(X1-((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+(WR-
(ToolDia/2))*mirrory):6:4);
     if (k <>nz) then
     writeln(arctrampa,'G00 Z',(Zllegada+E9):6:4);
    end
 else
  begin
     writeln(arctrampa,'G01 Z',(Zllegada+E9):6:4);
     writeln(arctrampa,'G00 X',(X1-((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+(WR-
(ToolDia/2))*mirrory):6:4);

  end;
close (arctrampa);
end;
If (residuoz<>0) then
begin
 append(arctrampa);
If SalidaCompleta=1 then
 writeln(arctrampa,'G00 Z',(Zllegada+E9):6:4);
 write(arctrampa,'G01 X',(X1-((ToolDia/2)+E12)*mirrorx):6:4);
 writeln(arctrampa,'Y',(y1+(WR-(ToolDia/2))*mirrory):6:4,'Z',(Zllegada-
(Dtz*nz+residuoz)):6:4);
 For I:= 1 to nradial do
```

```
  begin
  if odd(I) then
    begin
writeln(arctrampa,'G01 X',(X3+((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+(I*WR-
(ToolDia/2))*mirrory):6:4,'F',fdesbaste:6:4);
writeln(arctrampa,'G01 X',(X3+((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+((I+1)*WR-
(ToolDia/2))*mirrory):6:4,'F',fdesbaste:6:4);
    end
    else
   begin
writeln(arctrampa,'G01 X',(x1-((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+(I*WR-
(ToolDia/2))*mirrory):6:4,'F',fdesbaste:6:4);
writeln(arctrampa,'G01 X',(x1-((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+((I+1)*WR-
(ToolDia/2))*mirrory):6:4,'F',fdesbaste:6:4);
   end;
  end;
If Salidacompleta= 1 then
    begin
     writeln(arctrampa,'G01 Z',(Zllegada+E9):6:4);
     writeln(arctrampa,'G00 Z',(Zllegada+Zinicial):6:4,'F',Fdesbaste:6:4);
     writeln(arctrampa,'G00 X',(X1-((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+(WR-
(ToolDia/2))*mirrory):6:4);
    end
 else
  begin
     writeln(arctrampa,'G01 Z',(Zllegada+E9):6:4);
     writeln(arctrampa,'G00 X',(X1-((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+(WR-
(ToolDia/2))*mirrory):6:4);
  end;
 close(arctrampa);
end; {residuoz}
If finz <>0 then
 begin
append(arctrampa);
write(arctrampa,'G01 X',(X1-((ToolDia/2)+E12)*mirrorx):6:4);
writeln(arctrampa,'Y',(y1+(WR-(ToolDia/2))*mirrory):6:4,'Z',(Zllegada-
(Dtz*nz+residuoz+Finz)):6:4);
 For I:= 1 to nradial do
  begin
  if odd(I) then
   begin
writeln(arctrampa,'G01 X',(X3+((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+(I*WR-
(ToolDia/2))*mirrory):6:4,'F',fdesbaste:6:4);
If  (I<> nradial) then
writeln(arctrampa,'G01 X',(X3+((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+((I+1)*WR-
(ToolDia/2))*mirrory):6:4,'F',fdesbaste:6:4);
   end
  else
  begin
writeln(arctrampa,'G01 X',(x1-((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+(I*WR-
(ToolDia/2))*mirrory):6:4,'F',fdesbaste:6:4);
writeln(arctrampa,'G01 X',(x1-((ToolDia/2)+E12)*mirrorx):6:4,'Y',(y1+(I*WR-
(ToolDia/2))*mirrory):6:4,'F',fdesbaste:6:4);
  end;
end;

     writeln(arctrampa,'G01 Z',(Zllegada+E9):6:4);
     writeln(arctrampa,'G01 Z',(Zllegada+Zinicial):6:4,'F',Fdesbaste:6:4);
close(arctrampa);
 end; {finz}
end;
```

Ejemplo para Careado:

Utilizaremos la siguiente figura para ejemplificar nuestra macro de careado. La utilizaremos también para desarrollar un ejemplo posterior de macro programación de *Slot* recto.

Usaremos una piña de 1" para el careado. Nuestra profundidad de corte radial serán 0.750".

De la figura 9 podemos observar que nuestro cero pieza ha sido colocado de modo que generaremos nuestras coordenadas en el segundo cuadrante.

Observa que x1, y1 será igual al origen y x3,y3 será igual a (-6, 4.25) .

Carearemos una profundidad de corte de 0.100" , nota además que nuestra Z de llegada será igual a las 0.100" de modo que estaremos maquinando por encima del cero pieza.

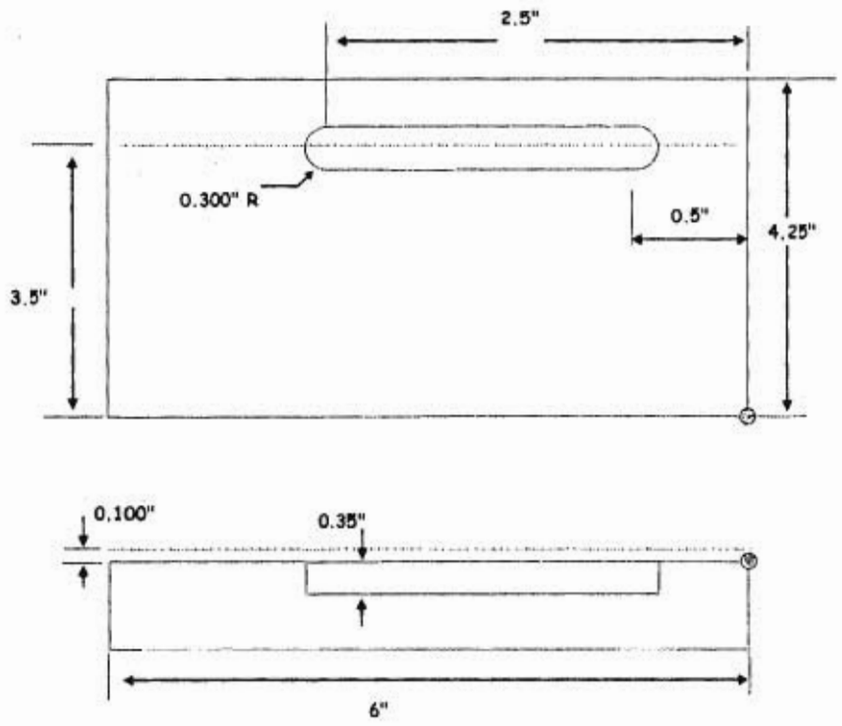


FIGURA 9

Ingresando estos datos en nuestra macro obtenemos el siguiente código EIA - ISO.

```
(FMXBIDIR)

G00 X  1.0000 Y 0.2500     Z 1.1000
G00 X  1.0000 Y 0.2500     Z 0.2200
G00 X  1.0000 Y 0.2500     Z 0.0000
G00 X  0.6000 Y 0.2500
G01 X -6.6000 Y 0.2500              F 10.0000
G01 X -6.6000 Y 1.0000              F 10.0000
G01 X  0.6000 Y 1.0000              F 10.0000
G01 X 0.6000  Y 1.7500              F 10.0000
G01 X -6.6000 Y 1.7500              F 10.0000
G01 X -6.6000 Y 2.5000              F 10.0000
G01 X 0.6000  Y 2.5000              F 10.0000
G01 X 0.6000  Y 3.2500              F 10.0000
```

```
G01 X-6.6000  Y 3.2500                F 10.0000
G01 X-6.6000  Y 4.0000                F 10.0000
G01 X 0.6000  Y 4.0000                F 10.0000
G01 X 0.6000 Y 4.7500                 F10.0000
G01                          Z 0.2200
G00                          Z 1.1000
G00 X 0.6000 Y 0.2500
```

Generando Careado para una superficie circular:

Para completar la exposición que hemos realizado, diseñaremos también un ciclo de careado para cuando rectifiquemos una superficie circular. Posiblemente pienses que podríamos aprovechar el hecho geométrico de que es posible incribir un círculo en un cuadrado y aprovechar algunas de las macros ya diseñadas para superficie rectangular. De hecho podríamos utilizar el ciclo de rectificado bidireccional corto, con más o menos buenos resultados , como podemos observar la figura 10.

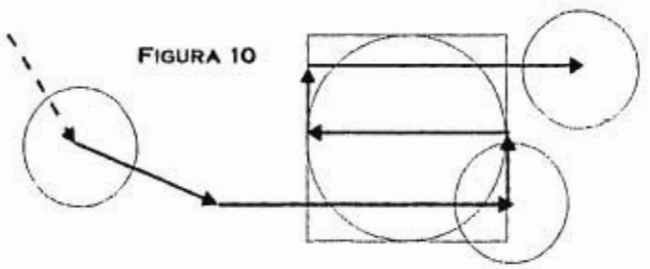


FIGURA 10

Esto no es lo óptimo, puesto que estamos adaptando el *tool path* concebido para una geometría a otra diferente pudiendo perder algo de tiempo al mecanizar. Compara con la figura siguiente (11) en la que mostramos el patrón que diseñaremos para rectificar superficie circular. Como ves este patrón de mecanizado aprovecha plenamente la geometría de la pieza que va a ser maquinada. puesto que generamos el maquinado en círculo precisamente

Deducción matemática del *tool path*:

Listemos primero los pasos del mecanizado, mismos que puedes observar en la figura.

1. Con movimiento rápido, desplazamos la herramienta por el plano XY hasta nuestro punto de aproximación .Bajamos rápidamente a Z inicial, instalando compensación de longitud de la herramienta.
2. Nos movemos , igualmente en rápido, a la altura de claro en Z o plano R .
3. Con avance movemos hasta la profundidad de corte en Z.

4. Avanzamos en XY a nuestro posición de inicio de corte o punto de claro radial.
5. Penetramos con el cortador hasta la profundidad de corte radial o el porcentaje de corte especificado
6. Interpolamos describiendo la mitad de un arco hasta las coordenadas simétricas opuestas .
7. Completamos el arco regresando a la posición del cortador en el punto 5.
8. Repetimos desde el paso 5 al 7 incrementando la profundidad de corte, hasta que el diámetro del círculo que reste de maquinar, sea menor al diámetro del cortador , llegando a este punto, recorremos en forma lineal con el cortador hasta la primera posición obtenida en 6 .
9. Salimos con el cortador en forma lineal a posición de claro radial.
10. La herramienta saie en Z para acabar la secuencia demaquinado.

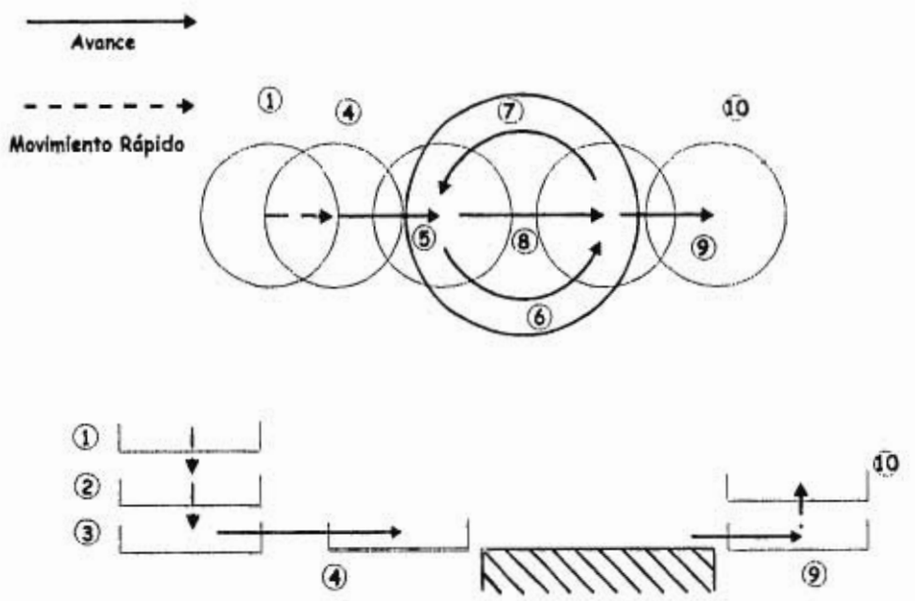


FIGURA 11

En la macro anterior de careado habíamos definido una variable n que nos servía para desplazar la herramienta sobre el eje Y. También definiremos a n como variable en esta macro, pero nos servirá en esta ocasión para deducir como se comportará nuestra macro ante diversas situaciones que pueden surgir para este mecanizado.

Definamos n = ϕ Círculo - ϕ Tool

Ahora de la siguiente figura veamos lo que nos implica la ecuación anterior:

n < 0 Implica que :

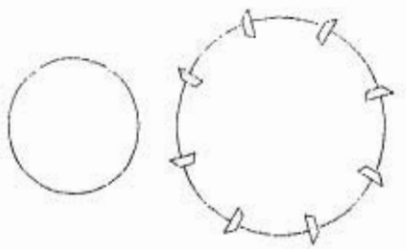


FIGURA 12

En este caso el diámetro del cortador es mayor que el diámetro a maquinar. Lo que significa que podemos rectificar nuestra superficie moviendo linealmente el cortador sobre ella .

n = 0 Implica que :

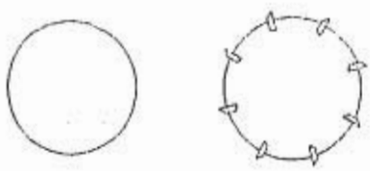


FIGURA 13

Ahora en la figura 13 , el diámetro del cortador es igual al de la superficie que maquinaremos. A primera vista la solución correcta parece ser, como en el caso anterior pasar linealmente al cortador sobre nuestra superficie. Esto por lo general , no dará buenos resultados ya que será difícil que el cortador limpie de forma total la superficie.

Plantearemos entonces el entrar a cortar con interpolación un porcentaje de corte calculado por nosotros, el cual estableceremos por medio de un parámetro al cual llamaremos E50. El operador podrá manipular este parámetro para regular la profundidad de corte del cortador. Una vez que el diámetro a maquinar sea menor que el diámetro del cortador procederemos a maquinar como en el caso anterior.

n > 0 Implica que :

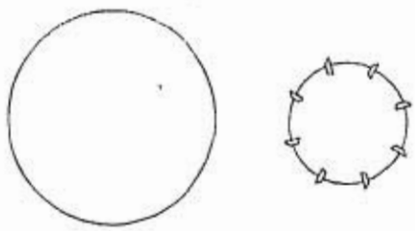


FIGURA 14

Nuestro último caso, será cuando el diámetro del cortador sea mayor que el diámetro a maquinar, seguirá la secuencia de maquinado mostrada en la figura 11 y será el primero que deduciremos:

Para n > 0:

Este es una geometria en el cual tenemos que calcular las coordenadas usando un ángulo; en la figura que muestra la secuencia de mecanizado nuestra herramienta inicia a 180° , pero para tener mayor flexibilidad debemos permitimos la elección libre de un ángulo de inicio. Posiblemente pienses que el introducir un ángulo en los datos requeridos por nuestra macro podría confundir al operador, podríamos entonces presetear dicho ángulo dentro de nuestra macro de modo que podamos mantenerlo como variable y modificarlo cuando creamos conveniente; además de que facilitará enormente el desarrollo de las ecuaciones.

Definamos nuestros puntos de aproximación como:

Aprox = Xini+(Rcírculo + ϕTool) cos θ

Aproy = Yini(Rcírculo + ϕTool) sin θ

Donde Rcírculo será igual al radio de la superficie a maquinar ; ϕTool el diámetro de la herramienta y θ el ángulo que se forma entre las coordenadas de aproximación de nuestra herramienta y el eje X.

Recordamos que la elección de dichos puntos de aproximación son arbitrarios y puedes modificarlos a tu conveniencia.

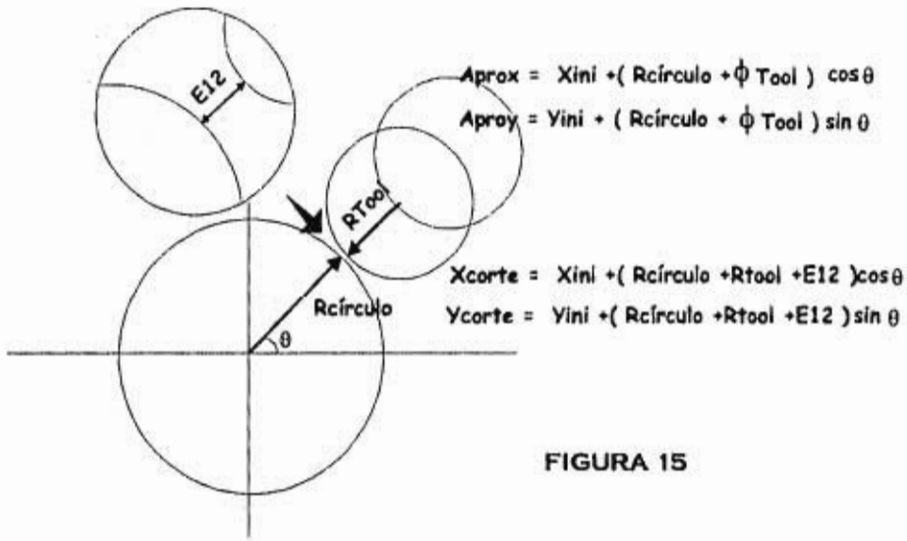


FIGURA 15

Nuestras cooordenadas de Inicio de corte serán definidas como:

X Inicio corte = xini+ (Rcírculo + Rtool + E12) cos θ

Y inicio corte = yini+ (Rcírculo + Rtool + E12) sin θ

Donde E12 será nuestro parámetro de claro radial ya definido anteriormente.

Entramos a profundidad de corte radial , con las siguientes ecuaciones:

G01 a: x1 = xini + (Rcírculo + Rtool - n Pcr) cos θ

y1 = yini + (Rcírculo + Rtool - n Pcr) sin θ

Desarrollamos la mitad de un arco y avanzamos 180° (π radianes) en sentido antihorario.

(Una macro completa de careado circular debe permitir al operador la opción de decidir sentido de corte ya sea CW o CCW). Nuestras siguientes ecuaciones serán:

G03 a : $x1_\pi$ = xini + (Rcírculo + Rtool - n Pcr) cos (θ + π) = - (Rcírculo +Rtool - nPcr) cos θ

$y1_\pi$ = yini + (Rcírculo + Rtool - n Pcr) sin (θ + π) = - (Rcírculo +Rtool –nPcr) sin θ

Usamos el subíndice π en estas ecuaciones para diferenciarlas de las anteriores . Ya que nuestra n sigue siendo la misma.

Para un ángulo cualquiera tenemos que usar dos vectores directores los cuales estarán dados como:

I_π = (Rcírculo + Rtool - n Pcr) cos (θ + π) = - (Rcírculo + Rtool - n Pcr) cos θ

J_π = (Rcírculo + Rtool - n Pcr) sin (θ + π)= - (Rcírculo + Rtool - n Pcr) sin θ

Completamos nuestro arco en sentido antihorario, regresando al punto de inicio del arco , por lo que nuestras ecuaciones quedan como:

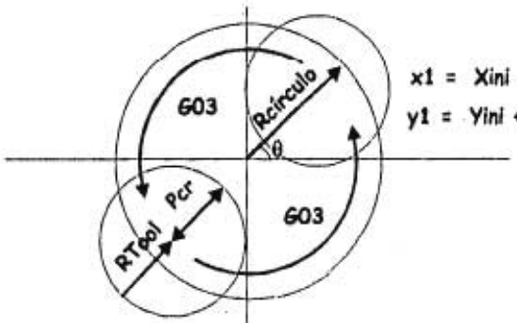
G03 a: x1 =xini +(Rcírculo + Rtool - n Pcr) cos θ

y1 = yini +(Rcírculo + Rtool - n Pcr) sin θ

Que son nuestras mismas ecuaciones de ingreso a corte radial, pero ahora tenemos que añadirles los vectores directores correspondientes que son:

I = (Rcírculo + Rtool - n Pcr) cos θ

J = (Rcírculo + Rtool - n Pcr) sin θ



x1 = Xini +(Rcírculo +Rtool -nPcr)cosθ

y1 = Yini +(Rcírculo +Rtool -nPcr) sinθ

FIGURA 16

x1 π = Xini -(Rcírculo +Rtool -nPcr)cosθ

y1 π = Yini -(Rcírculo +Rtool -nPcr) sin θ

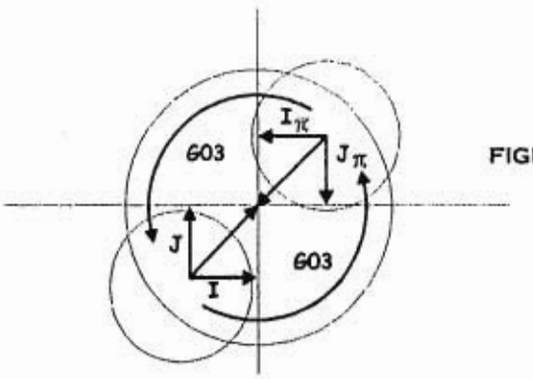


FIGURA 17

Todo este desarrollo podemos seguirlo a partir de las figuras 16 y 17.

La variable n será incrementada después de finalizar esta secuencia de maquinado

Ahora platiquemos cómo finalizaremos nuestro ciclo. Si te das cuenta en la ecuación de penetración radial que hemos planteado, conforme aumentamos n, esta ecuación tiende a hacerse más pequeña . Llegaremos al punto en que el diámetro del círculo que nos resta por maquinar sea menor o igual al diámetro del cortador.

Por lo que nuestro ciclo finalizará cuando :

$\phi Tool > Rcírculo + Rtool - nPcr$, de otro modo:

$Rtool > Rcírculo - nPcr$.

Finalizando el ciclo, recorremos el cortador en forma lineal hasta el punto de inicio de corte simétrico al que iniciamos:

G01 a : $x = xini - (Rcírculo + Rtool + E12) \cos \theta$

$y = yini - (Rcírculo + Rtool + E12) \sin \theta$

Para salir posteriormente con movimiento rápido movemos el cortador hasta las coordenadas de aproximación opuestas 180° a aquéllas que iniciamos:

G00 a : $Xsalida = xini - (Rcírculo + \phi Tool) \cos \theta$

$Ysalida = yini - (Rcírculo + \phi Tool) \sin \theta$

En estas coordenadas el cortador ya puede escapar libremente en Z .

Para n = 0 :

Las ecuaciones de aproximación del cortador y de inicio de corte serán las mismas que para el

caso anterior .

Establezcamos ahora las ecuaciones de corte.

Habíamos definido ya el porcentaje de corte como:

Porcentaje de corte = Pcr / ϕ Tool

Donde el porcentaje de corte será igual al parámetro E50 por lo que:

E50 x ϕ Tool = Pcr

Las ecuaciones de profundidad de corte radial serán:

G01 a : x1 = xini + (Rcírculo + Rtool –E50x ϕ Tool) cos θ

y1 = yini + (Rcírculo + Rtool –E50x ϕ Tool) sin θ

Desplazamos el cortador 180° por medio de las siguientes ecuaciones:

X_n = xini - (Rcírculo + Rtool - E50x ϕ Tool) cos θ

Y_n = yini - (Rcírculo + Rtool - E50x ϕ Tool) sin θ

Con los siguientes vectores directores:

I_n = - (Rcírculo + Rtool - E50x ϕ Tool) cos θ

J_n = - (Rcírculo + Rtool - E50x ϕ Tool) sin θ

Regresamos a nuestro punto de inicio de corte:

G03 a: x1 = xini+(Rcírculo + Rtool - E50x ϕ Tool) cos θ

y1 = yini+(Rcírculo + Rtool - E50x ϕ Tool) sin θ

I = (Rcírculo + Rtool - E50x ϕ Tool) cos θ

J = (Rcírculo + Rtool - E50x ϕ Tool) sin θ

El cortador ya maquinó un porcentaje de la pieza , recorremos la distancia restante en forma lineal al punto simétrico que iniciamos, para posteriormente salir al punto de aproximación.

G01 a : x = xini- (Rcírculo +Rtool +E12) cos θ

y = yini- (Rcírculo +Rtool +E12) sin θ

G00 a : Xsalida = xini- (Rcírculo + ϕTool) cos θ

Ysalida = yini- (Rcírculo + ϕTool) sin θ

Que son las ecuaciones de escape del cortador ya deducidas anteriormente.

Para n < 0 :

Este caso presenta la deducción más fácil de todas.

Simplemente utilizaremos las ecuaciones de aproximación y salida del cortador para maquinar nuestra pieza.

G00 a : Aprox = xini+ (Rcírculo + ϕTool) cos θ

Aproy = yini+ (Rcírculo + ϕTool) sin θ

G00 a: X inicio corte = xini +(Rcírculo + Rtool + E12) cos θ

Y inicio corte = yini +(Rcírculo + Rtool + E12) sin θ

G01 a : x = xini- (Rcírculo +Rtool +E12) cos θ

y = yini- (Rcírculo +Rtool +E12) sin θ

G00 a : Xsalida = xini- (Rcírculo + ϕTool) cos θ

Ysalida = yini- (Rcírculo + ϕTool) sin θ

En realidad estas ecuaciones están definiendo un movimiento lineal del cortador.

Programa en PASCAL para Careado Circular:

```
const
E9=0.12;{parametro que define el plano R en Z para el corte}
E12=0.100;{parametro que protege al Facemill al acercarse al maquinado}
Zinicial=1; {nuestra Z inicial}
S4=1E-5;{Nuestro punto flotante}
E50=0.75;
var
archivo:text;

procedure carecir(xini,yini,WR,Rtool,Radcir,Zllegada,Pcztotal,Pcz,Finz,
Fdesbaste,Facabado:real;SalidaCompleta:shortint;var arctrampa:text);
const
teta=0;
var
n,nz:shortint;
residuoz,Dtz,Dtzl,Falterna,nradial:real;
BanderaRadial,BanderaAxial,BanderaNegra,BanderaAcabado,Retirada:Boolean;
begin
 BanderaAcabado:=false;
 BanderaAxial:=true;
 BanderaNegra:=true;
 Retirada:=false;
 nz:=trunc((Pcztotal-Finz)/Pcz);
 residuoz:=puntoflotante(pcz*frac((pcztotal-Finz)/Pcz));
  If nz<=0 then
   begin
    If residuoz<=0 then
     begin
      Dtz:=finz;
      BanderaAcabado:=true;
```

```
      BanderaAxial:=false;
     end
    else
     Dtz:=residuoz
   end
  else if nz>0 then
    Dtz:=Pcz;
Dtz1:=Dtz;
nradial:=(RadCir-Rtool);
assign(arctrampa,'Cirbidir.Fil');
rewrite(arctrampa);
writeln(arctrampa,'CareCir');
write(arctrampa,'G00
X',(xini+(RadCir+Rtool*2)*cos(teta)):6:4,'Y',(yini+(RadCir+Rtool*2)*sin(teta)):6:4);
writeln(arctrampa,'Z',(Zinicial+Zllegada):6:4);
write(arctrampa,'G00
X',(Xini+(RadCir+Rtool*2)*cos(teta)):6:4,'Y',(yini+(RadCir+Rtool*2)*sin(teta)):6:4);
writeln(arctrampa,'Z',(E9+Zllegada):6:4);
write(arctrampa,'G00
X',(xini+(RadCir+Rtool*2)*cos(teta)):6:4,'Y',(yini+(RadCir+Rtool*2)*sin(teta)):6:4);
writeln(arctrampa,'Z',(Zllegada-Dtz):6:4);
write(arctrampa,'G00
X',(RadCir+Rtool+E12)*cos(teta):6:4,'Y',(RadCir+Rtool+E12)*sin(teta):6:4);
writeln(arctrampa,'Z',(Zllegada-Dtz):6:4);
close(arctrampa);
 While (dtz<=Pztotal) and not(Retirada) do
  begin
   If BanderaAcabado then
    Falterna:=Facabado
   else
    Falterna:=Fdesbaste;
   if nradial=0 then
    begin
     append(arctrampa);
     write(arctrampa,'G01 X',(xini+(RadCir+Rtool*(1 -2*E50))*cos(teta)):6:4);
     writeln(arctrampa,'Y',(yini+(RadCir+Rtool*(1-2*E50))*sin(teta)):6:4,'F',Falterna:6:4);
     write(arctrampa,'G03 X',(xini-(RadCir+Rtool*(1-2*E50))*cos(teta)):6:4);
     write(arctrampa,'Y',(yini-(RadCir+Rtool*(1-2*E50))*sin(teta)):6:4);
     write(arctrampa,'I',-(RadCir+Rtool*(1-2*E50))*cos(teta):6:4);
     writeln(arctrampa,'J',-(RadCir+Rtool*(1-2*E50))*sin(teta):6:4,'F',Falterna:6:4);
     write(arctrampa,'G03 X',(xini+(RadCir+Rtool*(1-2*E50))*cos(teta)):6:4);
     write(arctrampa,'Y',(yini+(RadCir+Rtool*(1-2*E50))*sin(teta)):6:4);
     write(arctrampa,'I',(RadCir+Rtool*(1-2*E50))*cos(teta):6:4);
     writeln(arctrampa,'J',(RadCir+Rtool*(1-2*E50))*sin(teta):6:4,'F',Fdesbaste:6:4);
     write(arctrampa,'G01 X',(xini-(RadCir+Rtool+E12)*cos(teta)):6:4);
     writeln(arctrampa,'Y',(yini-(RadCir+Rtool+E12)*sin(teta)):6:4,'F',Falterna:6:4);
     close(arctrampa);
    end
   else if nradial< 0 then
    begin
     append(arctrampa);
     write(arctrampa,'G01 X',(Xini-(RadCir+Rtool+E12)*cos(teta)):6:4);
     writeln(arctrampa,'Y',(yini-(RadCir+Rtool+E12)*sin(teta)):6:4,'F',Falterna:6:4);
     close(arctrampa);
    end
   else
    begin
     n:=1;
     append(arctrampa);
      While (2*RadCir-n*WR >= 2*Rtool) do
       begin
        write(arctrampa,'G01 X',(xini+(RadCir+Rtool-n*WR)*cos(teta)):6:4);
        writeln(arctrampa,'Y',(yini+(RadCir+Rtool-n*Wr)*sin(teta)):6:4,'F',Falterna:6:4);
        write(arctrampa,'G03 X',(xini-(RadCir+Rtool-n*WR)*cos(teta)):6:4,'Y',(yini-
(RadCir+Rtool-n*Wr)*sin(teta)):6:4);
        writeln(arctrampa,'I',-(RadCir+Rtool-n*Wr)*cos(teta):6:4,'Y',-(RadCir+Rtool-
n*Wr)*sin(teta):6:4,'F',Falterna:6:4);
        write(arctrampa,'G03 X',(xini+(RadCir+Rtool-
n*WR)*cos(teta)):6:4,'Y',(yini+(RadCir+Rtool-n*Wr)*sin(teta)):6:4);
        writeln(arctrampa,'I',(RadCir+Rtool-n*Wr)*cos(teta):6:4,'Y',(RadCir+Rtool-
n*Wr)*sin(teta):6:4,'F',Falterna:6:4);
        n:=n+1;
```

```
        end; {while}
       close(arctrampa);
      end; {ultimo else}
     append(arctrampa);
     write(arctrampa,'G01 X',(xini-(Radcir+Rtool+E12)*cos(teta)):6:4);
     writeln(arctrampa,'Y',(yini-(Radcir+Rtool+E12)*sin(teta)):6:4,'F',Falterna:6:4);
     writeln(arctrampa,'G01 Z',(Zllegada+E9):6:4,'F',falterna:6:4);
     nz:=nz-1;
     if (nz<=0) and (residuoz<>0) and (finz<>0 ) then
      begin
       If BanderaNegra then
        begin
         Dtz:=Dtz+residuoz;
         BanderaNegra:=false;
        end
       else if not(BanderaNegra) then
        begin
         Dtz:=Dtz+finz;
         BanderaAcabado:=true;
         BanderaAxial:=false;
        end
      end
     else if (nz<=0) and (residuoz=0) and (finz<>0 ) then
      begin
       Dtz:=Dtz+finz;
       BanderaAcabado:=true;
       BanderaAxial:=false;
      end
     else if (nz<=0) and (residuoz<>0) and (finz=0) then
      begin
       Dtz:=Dtz+residuoz;
      end
     else if nz > 0 then
      Dtz:=Dtz+Dtzl
     else
      Retirada:=true;
    If salidacompleta=1 then
     writeln(arctrampa,'G00 Z',(Zllegada+ Zinicial):6:4);
    If BanderaAxial and not(Retirada) then
      begin
       write(arctrampa,'G00 X',(xini-(RadCir+Rtool+E12)*cos(teta)):6:4);
       writeln(arctrampa,'Y',(yini-(RadCir+Rtool+E12)*sin(teta)):6:4);
      If salidacompleta=1 then
       writeln(arctrampa,'G00 Z',(Zllegada+E9):6:4);
       writeln(arctrampa,'G01 Z',(Zllegada-Dtz):6:4,'F',Falterna:6:4);
      end;
     close(arctrampa);
   end;{while de z}
end;{procedimiento}
```

Ejemplo para Careado Circular:

Utilizaremos la figura 18 para el ejemplo de mecanizado. Esta misma pieza servirá para la macro de *Slot* Curvo que deduciremos posteriormente.

Observa que el cero pieza esta colocado en el centro del círculo a maquinar; además de manera semejante al ejercicio anterior está ubicado de modo tal que la superficie a rectificar está arriba del mismo.

Utilizaremos una herramienta de 2" para el careado.

La profundidad de corte axial será de 0.120" , para desbaste el cual maquinaremos con avance de 18 inch. / minuto , para acabado dejaremos un espesor de 0.005" maquinando con avance de 12 inch /min, el ángulo de entrada del cortador será cero.

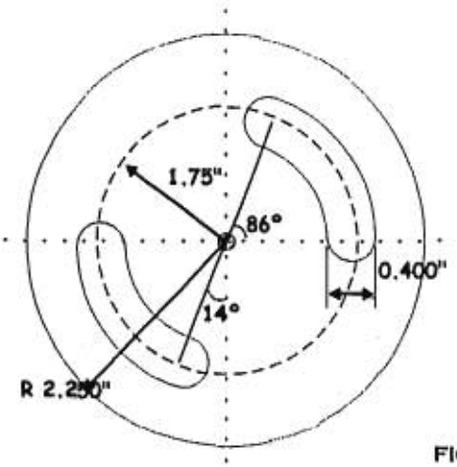
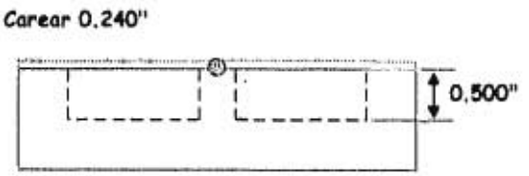


FIGURA 18



Ingresando estos datos a nuestra macro obtenemos el siguiente código:

```
  (CareCir)
G00 X 4.2500 Y 0     Z 1.2400
G00 X 4.2500 Y 0     Z 0.3600
G00 X 4.2500 Y 0     Z 0.1200
G00 X 3.3500 Y 0     Z 0.1200
G01 X 2.5000 Y 0                    F 18.0000
G03 X-2.5000 Y 0    I-2.5000 J 0    F 18.0000
G03 X 2.5000 Y 0    I 2.5000 J 0    F 18.0000
G01 X 1.7500 Y 0                    F 18.0000
G03 X-1.7500 Y 0    I-1.7500 J 0    F 18.0000
G03 X1.7500  Y 0    I 1.7500 J 0    F 18.0000
G01 X 1.0000 Y 0                    F 18.0000
G03 X-1.0000 Y 0    I-1.0000 J 0    F 18.0000
G03 X 1.0000 Y 0    I 1.0000 J 0    F 18.0000
G01 X-3.3500 Y 0                    F 18.0000
G01                 Z 0.3600        F 18.0000
G00                 Z 1.2400
G00 X-3.3500        Y 0
G00                 Z 0.3600
G01                 Z 0.0050        F 18.0000
G01 X 2.5000 Y 0                    F 18.0000
G03 X-2.5000 Y 0    I-2.5000 J 0    F 18.0000
G03 X 2.5000 Y 0    I 2.5000 J 0    F 18.0000
G01 X 1.7500 Y 0                    F 18.0000
```

```
G03 X-1.7500 Y 0    I-1.7500 J 0   F 18.0000
G03 X 1.7500 Y 0    I 1.7500 J 0   F 18.0000
G01 X 1.0000 Y 0                   F 18.0000
G03 X-1.0000 Y 0    I-1.0000 J 0   F 18.0000
G03 X 1.0000 Y 0    I 1.0000 J 0   F 18.0000
G01 X-3.3500 Y 0                   F 18.0000
G01                 Z 0.3600       F 18.0000
G00                 Z 1.2400
G00 X-3.3500 Y 0
G00                 Z 0.3600
G01                 Z 0            F 18.0000
G01 X 2.5000 Y 0                   F 12.0000 ( Inicia Acabado )
G03 X-2.5000 Y 0    I-2.5000 J 0   F 12.0000
G03 X 2.5000 Y 0    I 2.5000 J 0   F 12.0000
G01 X 1.7500 Y 0                   F 12.0000
G03 X-1.7500 Y 0    I-1.7500 J 0   F 12.0000
G03 X 1.7500 Y 0    I 1.7500 J 0   F 12.0000
G01 X 1.0000 Y 0                   F 12.0000
G03 X-1.0000 Y 0    I-1.0000 J 0   F 12.0000
G03 X 1.0000 Y 0    I 1.0000 J 0   F 12.0000
G01 X-3.3500 Y 0                   F 12.0000
G01                 Z 0.3600       F 12.0000
G00                 Z 1.2400
```

Macroprogramación de un *Slot* recto:

Un *slot* recto sirve para alojar generalmente una cuña , por lo que es un dispositivo que se maquina con regularidad en el taller. Procederemos pues a generar nuestra macro para *slot* recto. De la figura podemos observar la geometría de nuestra pieza y sacar a partir de ella, las

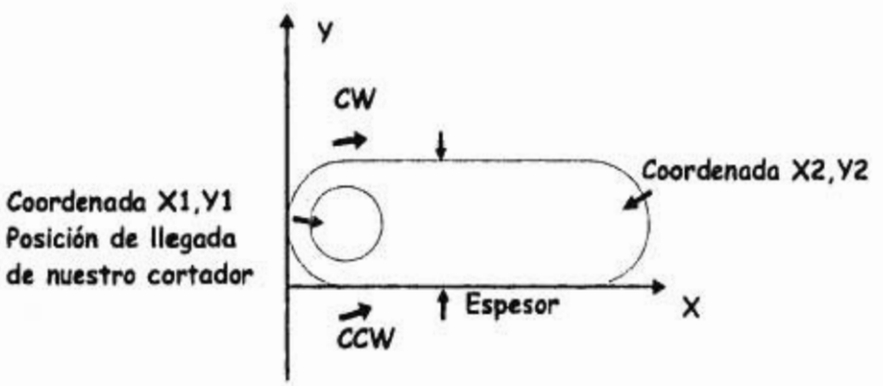


FIGURA 19

características geométricas que la definen completamente. Te darás cuenta que necesitamos las coordenadas que definen el centro del primer círculo (x1,y1), así como el espesor de nuestro *slot* y las coordenadas del segundo círculo (x2,y2) . Desarrollaremos nuestra macro en sentido de

corte horario; aunque en realidad y esto se deja como ejercicio al lector la macro deberá generar código para maquinar tanto en sentido CW como CCW .

Hemos dispuesto no sin cierta premeditación el colocar nuestra figura con el punto de inicio de corte (x1,y1) en el primer cuadrante de nuestros ejes coordenados, sin ángulo de inclinación alguno. Esto lo hacemos para facilitar la deducción del *tool path*. Ya que tomaremos como punto pivote la coordenada x1,y1. Sin embargo al final agregaremos las ecuaciones necesarias en nuestra macro para que podamos generar código de maquinado en cualquier posición que se encuentre el *slot*.

Deducción del código de maquinado:

Es obvio que nuestra primera coordenada será bajar en Z el cortador en la coordenada x1,y1, a la profundidad de corte axial, sin embargo debemos aclarar que para alargar la vida de nuestra herramienta (evitando incluso romperla), deberá maquinarse previamente cuando menos dos barrenos a la profundidad de corte, cuyo diámetro exceda o al menos sea igual al cortador que pensamos utilizar para maquinar nuestro *slot*, esto es como ya dijimos anteriormente, debido a que los cortadores se diseñan por lo general, para tener esfuerzo radial , no axial; es decir no pueden empujar hacia Z. Los barrenos sirven para que la herramienta descienda sin esfuerzo en Z, además de proporcionar un medio de desahogo para la rebaba. Esta tarea sin embargo es a elección del operador, por lo que no incluiremos el barrenado en nuestra macro.

Para nuestra segunda coordenada, la coordenada que inicia propiamente el corte, necesitamos entrar con Arc –In .Por razones de sencillez, estableceremos un arco de entrada a 90°, la herramienta se desplazara a la profundidad de corte radial (Pcr) tanto en X, como en Y, así tendremos únicamente vector director en X+, con magnitud igual a Pcr.

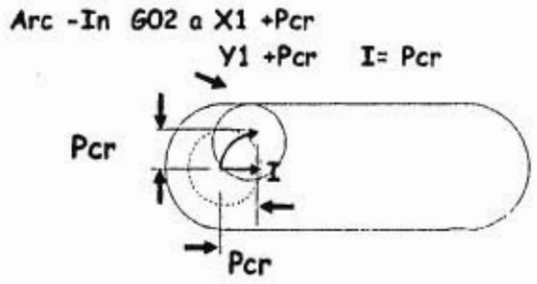


FIGURA 20

Nuestra tercera coordenada la obtendremos desplazando linealmente nuestro cortador, hasta el comienzo del segundo arco,como observamos en la figura 21:

G01 a X1 +distancia(X1,Y1,X2,Y2)
Y1 +Pcr

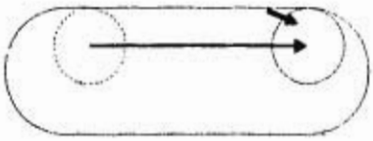


FIGURA 21

Con distancia (x1,y1,x2,y2) queremos decir ,la distancia efectiva desde el centro del primer círculo (x1,y1) al centro del segundo . De otro modo :

$$\text{distancia}\,(x1,y1,x2,y2) = \sqrt{\left[(X2 - X1)^2 + (Y2 - Y1)^2\right]}$$

Que es nuestra conocida fórmula de distancia entre dos puntos.

Si eres observador, te darás cuenta que para la posición específica de la figura en la que estamos deduciendo las ecuaciones, la resta de Y2 y Y1 vale cero, más debemos incluirlas en nuestra ecuación, para que ésta nos de los resultados correctos cuando el *slot* se encuente en una posición diferente a aquélla que estamos deduciendo.

Para nuestra cuarta coordenada, el cortador se desplaza hasta la mitad del segundo círculo, como podemos observar en la figura 22:

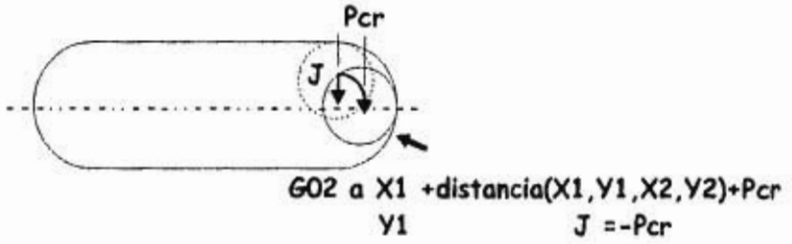


FIGURA 22

Nuestra coordenada en X se incrementa en la profundidad de corte radial, en Y regresamos a nuestro punto de partida, y nuestro vector director apunta en la dirección J-.

Es importante que observes cómo todas las coordenadas se van desarrollando a partir de nuestro punto pivote, (x1,y1). La elección de un punto pivote adecuado, nos asegura el desarrollar las ecuaciones de nuestra macro de la manera más óptima.

Nuestra quinta coordenada acabará de maquinar por completo el segundo círculo, nota que ahora decrementamos nuestra coordenada en Y.

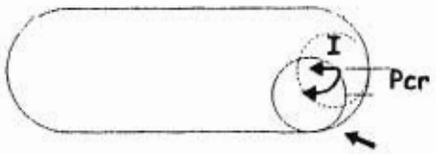


FIGURA 23

G02 a X1 +distancia(X1,Y1,X2,Y2)
Y1-Pcr I =-Pcr

Para la sexta coordenada, empleamos comando lineal G01, con el cual regresaremos a x1, manteniendo nuestra posición en Y.

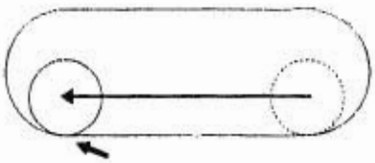


FIGURA 24

G01 a X1
Y1-Pcr

Maquinaremos la mitad de nuestro primer círculo en nuestra séptima coordenada:

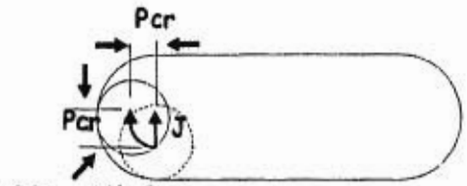


FIGURA 25

G02 a X1-Pcr
Y1 J=Pcr

De la figura 25, verás que avanzamos hasta el extremo derecho de nuestro primer círculo avanzando la profundidad de corte radial tanto en X, como en Y, nota también que el vector director es J+.

G02 a X1
Y1+Pcr I=Pcr

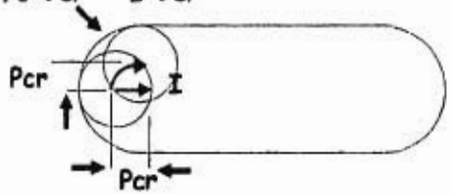


FIGURA 26

Nuestra octava coordenada (figura 26) nos servirá para finalizar de maquinar nuestro circulo , observa que hemos regresado a nuestro punto inicial en X, aunque nuestra distancia en Y, está aumentada por la profundidad de corte, el vector director apunta en la dirección I+.

La novena coordenada nos sirve para maquinar el claro que hemos dejado sin maquinar entre nuestra octava coordenada y la entrada del Arc –In , dicha distancia aparentemente será en X, la profundidad de corte, como podemos ver en la figura 27

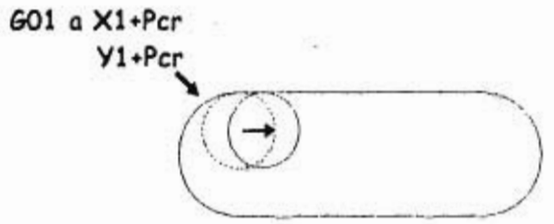


FIGURA 27

Sin embargo , cuando realizamos maquinado en el interior , existen algunos materiales en los cuales necesitamos remaquinar o repasar una pequeña distancia para asegurarnos de que el material quede bien maquinado. Este concepto se conoce como "overlap" . Y se ilustra en la siguiente figura. En ella entramos conArc –In en el punto 1 , pero en lugar de salir con Arc-Out en el mismo punto recorremos una pequeña distancia, repasando la pieza, finalizando en el punto 2.

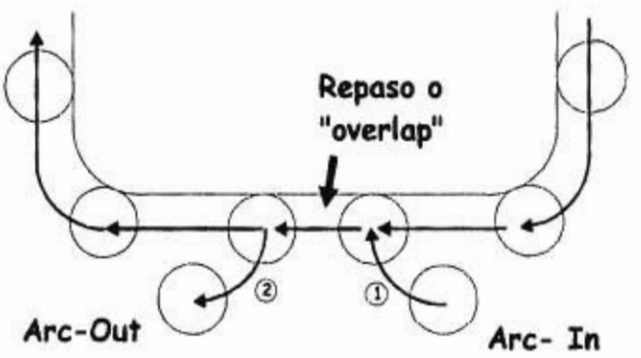


FIGURA 28

Generalmente esta distancia de remaquinado o repasado va de un rango de 0.010" a .030" . Nosotros definiremos dicha distancia de "overlap" en nuestra macro por medio de un parámetro al cual denominaremos E21.

Lo que en realidad estamos avanzando la herramienta en X, será la profundidad de corte más una distancia de "overlap" ; por lo que nuestras ecuaciones para la coordenada 9,quedan como sigue:

G01 A: x1 + Pcr + E21
y1 +Pcr

Es necesario ya el retirar la herramienta de la superficie de corte, utilizaremos Arc-Out para este efecto. Para nuestra coordenada 10, retraeremos la herramienta a la profundidad de corte radial en X y Y, nuevamente formando ún ángulo de 90º.Tal como observamos en la figura 29.

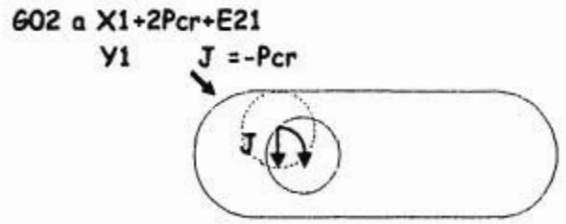


FIGURA 29

Para finalizar y dejar la herramienta lista para inicio de otro ciclo de corte - si éste fuera necesario – llevaremos la herramienta al punto de inicio con avance; por lo que nuestra coordenada número 11 sera:
G01 a x1 , y1.

Te darás cuenta lo fácil que ha sido determinar las ecuaciones para este ejemplo, a pesar de que al principio parecía un poco complicado. El hecho de elegir un punto pivote adecuado, la elección de nuestro Arc –In, y el deducir las ecuaciones a partir de una posición geométrica sencilla nos han echado la mano.

Necesitamos ahora el generalizar las ecuaciones que hemos deducido; esto es para generemos nuestro código de maquinado independientemente del punto donde sea programado el *slot* y del ángulo de inclinación que tenga éste .
La técnica de espejo no nos servirá en esta ocasión, necesitamos algo más eficaz, por ello debemos recurrir a ciertos útiles conceptos matemáticos .

Ecuaciones para transladar ejes:
Sean X y Y nuestros ejes de coordenadas originales, sean X' , Y' ejes paralelos a los anteriores . A nuestro origen inicial le llamaremos O y valdra (0, 0) , sea nuestro origen en X',Y' O', cuyas

coordenadas con respecto al punto O son h y k respectivamente. Sea un punto P ubicado en nuestro nuevo sistema de coordenadas, de la figura es claro que sus coordenadas con respecto a los nuevos ejes serán x',y', sin embargo con respecto a los ejes anteriores sus ecuaciones serán:

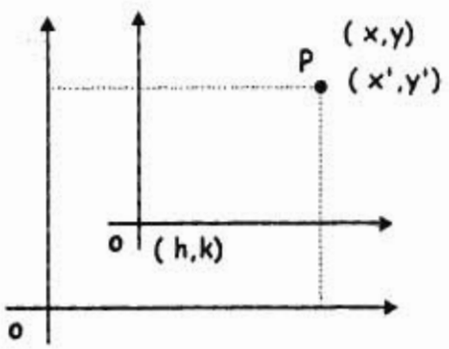


FIGURA 30

$x = h + x'$

$y = k + y'$

Que son nuestras ecuaciones para transladar ejes.

Ecuaciones para rotar ejes:

De manera similar: sean X, Y los ejes de nuestras coordenadas originales y sean X' , Y' nuestros nuevos ejes. O es nuestro origen común a ambos sistemas de ejes, y sea 0 el ángulo que están rotados los ejes X'Y'. Sean P (x,y) , coordenadas de un punto con respecto a nuestros ejes originales y (x',y') las coordenadas del mismo punto con respecto al nuevo sistema de ejes. Si queremos determinar x y en función de x',y' y teta llegamos a:

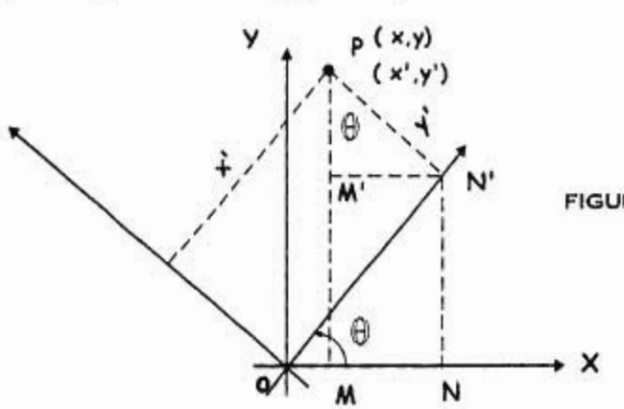


FIGURA 31

$x = OM = ON - MN = x'\cos\theta - y'\sin\theta$

$y = MP = MM' + M'P = NN' + M'P = x'\sin\theta + y'\cos\theta$

Por lo tanto las fórmulas de rotación θ de los ejes coordenadas son:

$x = x'\cos\theta - y'\sin\theta$

$y = x'\sin\theta + y'\cos\theta$.

Generalizando ecuaciones:

Ahora debemos saber cómo aplicar estas ecuaciones de transiación y rotación a nuestras ecuaciones que hemos deducido :

Nuestra primera coordenada como ya vimos , estaba definida como:

$X = x1 + Pcr$

$Y = y1 + Pcr$

$i = Pcr$

Las ecuaciones que definen nuestras coordenadas X y Y, se dividen en dos, una donde se encuentra definido nuestro punto pivote, (x1,y1) y otra que define el movimiento con respecto a éste. Te darás cuenta que siempre que rotemos nuestra figura, la rotamos con respecto a dicho punto pivote , es decir dicho punto no puede ser rotado. De otro modo, sin darnos cuenta hemos definido nuestras ecuaciones de tal forma que la primera componente de cada una de ellas está transladando los ejes coordenados, y la segunda parte es la que tendremos que multiplicar por el ángulo de inclinación que tenga nuestro *slot*, es decir multiplicaremos únicamente el segundo miembro de cada ecuación por las ecuaciones de rotación de ejes, y el resultado sumado al primer miembro nos dará la ecuación general que necesitamos.

Denominemos alfa al ángulo que existe entre nuestra coordenada (x1,y1) y nuestra coordenada (x2,y2).

$$\alpha = \arctan\left[(y2 - y1)/(x2 - x1)\right]$$

Nuestras ecuaciones de rotación de ejes son:

$x = x'\cos\alpha - y'\sin\alpha$

$y = x'\sin\alpha + y'\cos\alpha$

Para nuestras primeras ecuaciones $x' = Pcr$, $y' = Pcr$

Por lo que la componente x de rotación sera: $Pcr\cos\alpha - Pcr\sin\alpha = Pcr(\cos\alpha - \sin\alpha)$.

Y nuestra componente de rotación y es: $Pcr\sin\alpha + Pcr\cos\alpha = Pcr(\sin\alpha + \cos\alpha)$.

Nos falta también multiplicar nuestro vector director I por las ecuaciones de rotación, que reescribiremos de la siguiente manera:

$I = I_x\cos\alpha - J_y\sin\alpha$

$J = I_x \sin \alpha + J_y \cos \alpha$

En este caso I_x será igual a Pcr , e $J_y = 0$, de lo que nuestros vectores directores quedan como sigue:

$I = Pcr \cos \alpha$

$J = Pcr \sin \alpha$

Lo interesante es darnos cuenta de que a pesar de que deducimos nuestro vector director con componente únicamente en x, (I) , a la hora de generalizar nuestras ecuaciones , tenemos componente en X y Y. Esto nos muestra lo importante que resulta el simplificar el cálculo de nuestra macro – en este caso por medio de la posición geométrica - , para posteriormente utilizando las herramientas matemáticas adecuadas completarla.

Por lo que nuestras coordenadas para el Arc – In quedan como sigue:

$X_2 = x1 + Pcr (\cos \alpha - \sin \alpha)$.

$Y_2 = y1 + Pcr (\sin \alpha + \cos \alpha)$.

$I = Pcr \cos \alpha$

$J = Pcr \sin \alpha$

De manera similar, procederemos con el resto de las ecuaciones que hemos deducido:

Tercera coordenada.

$X_3 = x1 + [\text{distancia } (x1,y1,x2,y2)] \times (\cos \alpha) - (Pcr) \times (\sin \alpha)$

$Y_3 = y1 + [\text{distancia } (x1,y1,x2,y2)] \times (\sin \alpha) + (Pcr) \times (\cos \alpha)$

Cuarta coordenada.

$X_4 = x1 + [\text{distancia } (x1,y1,x2,y2) + Pcr] \times (\cos \alpha)$

$Y_4 = y1 + [\text{distancia } (x1,y1,x2,y2) + Pcr] \times (\sin \alpha)$

$I = Pcr \sin \alpha$

$J = -Pcr \cos \alpha$

Quinta coordenada.

$X_5 = x1 + [\text{distancia } (x1,y1,x2,y2)] \times (\cos \alpha) + (Pcr) \times (\sin \alpha)$

$Y_5 = y1 + [\text{distancia } (x1,y1,x2,y2)] \times (\sin \alpha) - (Pcr) \times (\cos \alpha)$

$I = -Pcr \cos \alpha$

$J = -Pcr \sin \alpha$

Sexta coordenada

$X_6 = x1 + Pcr \sin \alpha$

$Y_6 = y1 - Pcr \cos \alpha$

Séptima coordenada.

$X_7 = x1 + Pcr \cos \alpha$

$Y_7 = y1 + Pcr \sin \alpha$

$I = Pcr \sin \alpha$

$J = Pcr \cos \alpha$

Octava coordenada.

$X_8 = x1 - Pcr \sin \alpha$

$Y_8 = y1 + Pcr \cos \alpha$

$I = Pcr \cos \alpha$

$J = Pcr \sin \alpha$

Novena coordenada.

$X_9 = x1 + (Pcr) \times (\cos \alpha - \sin \alpha) + (E21) \times \cos \alpha$

$Y_9 = y1 + (Pcr) \times (\sin \alpha + \cos \alpha) + (E21) \times \sin \alpha$

Décima coordenada. (Arc – Out)

$X_{10} = x1 + (2Pcr + E21) \cos \alpha$

$Y_{10} = y1 + (2pcr + E21) \sin \alpha$

$I = Pcr \sin \alpha$

$J = -Pcr \cos \alpha$

Nuestra coordenada final es el retorno a x1,y1 con lo que completamos la deducción matemática del *tool path* para *slot* recto.

A continuación el código en PASCAL que genera el código de maquinado:

```
const
D41=0.1; {Valor de parametro para plano R}
Zinicial=1.0; { Nuestra Z inicial}
S4=1E-5; { Parámetro máquina para definir punto flotante}


procedure slotrecto(x1,y1,x2,y2,ancho,tooldia,Fzd,Fr,Ff,Pcr,Finr,Pcztotal,Pcz,
Zllegada,FinZ:real;subida,sentidocorte:shortint;var arctrampa:text);
var
alfa,Dtx,Dtz,residuor,residuoz,ancholr,Xtemp,Ytemp,Itemp,Jtemp:real;
nr,nz,signo:shortint;
Sentido:string[3];
Banderapirata,BanderaNegra:boolean;
```

```
begin
alfa:=Angulo(x1,y1,x2,y2);
if sentidocorte=1 then
 begin
  signo:=1;
  Sentido:='G02';
 end
else
begin
signo:=-1;
Sentido:='G03';
end;
assign(arctrampa,'SLOT.fil');
rewrite(arctrampa);
writeln(arctrampa,'G00 X',x1:6:4,'Y',y1:6:4,'Z',(D41+Zllegada):6:4);
close(arctrampa);
nz:=trunc((Peztotal-FinZ)/Pcz);
residuoz:=puntoflotante(Pcz*(frac((Peztotal-FinZ)/Pcz)));
Banderapirata:=true;
 if (nz<=0) then
  Dtz:=residuoz
 else
  Dtz:=Pcz;
While (Dtz<=Peztotal)and (BanderaPirata) do
   begin
    append(arctrampa);
    writeln(arctrampa,'G01 Z',(Zllegada-Dtz):6:4,'F',Fzd:6:4);
    close(archivo);
    nz:=(nz-1);
     if (nz=0) and (residuoz <>0) then
      Dtz:=Dtz+residuoz
     else if (nz=0) and (residuoz=0) then
       begin
        if (finz=0) then
         BanderaPirata:=false
       else
         Dtz:=Dtz+finz
       end
     else if (nz<0) and (residuoz<>0) then
       begin
        if (finz=0) then
         Banderapirata:=false
       else
         Dtz:=Dtz+Finz
        end
     else if (nz>0) then
      Dtz:=Dtz+Pcz;
   nr:=trunc(((Ancho-Tooldia)/2)-Finr)/Pcr);
   residuor:=puntoflotante((frac((((Ancho-Tooldia)/2)-Finr)/Pcr))*Pcr);
   BanderaNegra:=true;
    if (nr<=0) then
     Dtx:=residuor
    else
     Dtx:=pcr;
   While (Dtx+(Tooldia)/2<=ancho/2) and BanderaNegra do
    begin
     append(arctrampa);
     xtemp:=X1+Dtx*(cos(alfa)-sin(alfa));
     ytemp:=Y1+(Dtx*signo*(sin(alfa)+cos(alfa)));
     Itemp:=Dtx*cos(alfa);
     Jtemp:=Dtx*sin(alfa);

writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Fr:6:4
);
     xtemp:=X1+distancia(x1,y1,x2,y2)*cos(alfa)-Dtx*signo*sin(alfa);
     ytemp:=Y1+distancia(x1,y1,x2,y2)*sin(alfa)+Dtx*signo*cos(alfa);
     writeln(arctrampa,'G01 X',xtemp:6:4,'Y',ytemp:6:4,'F',Fr:6:4);
     xtemp:=X1+(distancia(x1,y1,x2,y2)+Dtx)*cos(alfa);
     ytemp:=Y1+(distancia(x1,y1,x2,y2)+Dtx)*sin(alfa);
     Itemp:=Dtx*Signo*sin(alfa);
```

```
     Jtemp:=-Dtx*Signo*cos(alfa);
writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Fr:6:1
);
     xtemp:=X1+distancia(x1,y1,x2,y2)*cos(alfa)+Dtx*signo*sin(alfa);
     ytemp:=Y1+distancia(x1,y1,x2,y2)*sin(alfa)-Dtx*signo*cos(alfa);
     Itemp:=-Dtx*cos(alfa);
     Jtemp:=-Dtx*sin(alfa);
writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Fr:6:4
);
     xtemp:=X1+Dtx*signo*sin(alfa);
     ytemp:=Y1-Dtx*signo*cos(alfa);
     writeln(arctrampa,'G01 X',xtemp:6:4,'Y',ytemp:6:4,'F',Fr:6:4);
     xtemp:=X1+Dtx*signo*cos(alfa);
     ytemp:=Y1+Dtx*signo*sin(alfa);
     Itemp:=-Dtx*signo*sin(alfa);
     Jtemp:=Dtx*signo*cos(alfa);
writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Fr:6:4
);
     xtemp:=X1-Dtx*signo*sin(alfa);
     ytemp:=Y1+Dtx*signo*cos(alfa);
     Itemp:=Dtx*cos(alfa);
     Jtemp:=Dtx*sin(alfa);
writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Fr:6:4
);
     xtemp:=X1+Dtx*(cos(alfa)-signo*sin(alfa));
     ytemp:=Y1+Dtx*(sin(alfa)+signo*cos(alfa));
     writeln(arctrampa,'G01 X',xtemp:6:4,'Y',ytemp:6:4,'F',Fr:6:4);
     xtemp:=X1+2*Dtx*cos(alfa);
     ytemp:=Y1+2*Dtx*sin(alfa);
     Itemp:=Dtx*sin(alfa);
     Jtemp:=-Dtx*signo*cos(alfa);
writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Fr:6:4
);
     xtemp:=X1;
     ytemp:=Y1;
     writeln(arctrampa,'G01 X',xtemp:6:4,'Y',ytemp:6:4,'F',Fr:6:4);
     close(arctrampa);
     nr:=(nr-1);
      if (nr=0) and (residuor<>0) then
        Dtx:=Dtx+residuor
      else if (nr=0) and (residuor=0) then
        begin
        if (finr=0) then
         BanderaNegra:=false
        else
         Dtx:=Dtx+Finr
        end
      else if (nr<0) and (residuor<>0) then
        begin
        if (finr=0) then
         BanderaNegra:=false
        else
        Dtx:=Dtx+Finr
       end
      else if (nr>0) then
      Dtx:=Dtx+Pcr;
    end; {fin del ciclo para fresar en desbaste}
    If subida=1 then
      begin
       append(arctrampa);
       writeln(arctrampa,'G01 Z',(D41+Zllegada):6:4,'F',Fzd:6:4);
      end;
  end; {fin del ciclo para bajar en z}
end;
```

Ejemplo para macro programa de *Slot* Recto.

Para completar la pieza que hemos careado previamente, tenemos que maquinar el *slot* que se

indica en la misma.

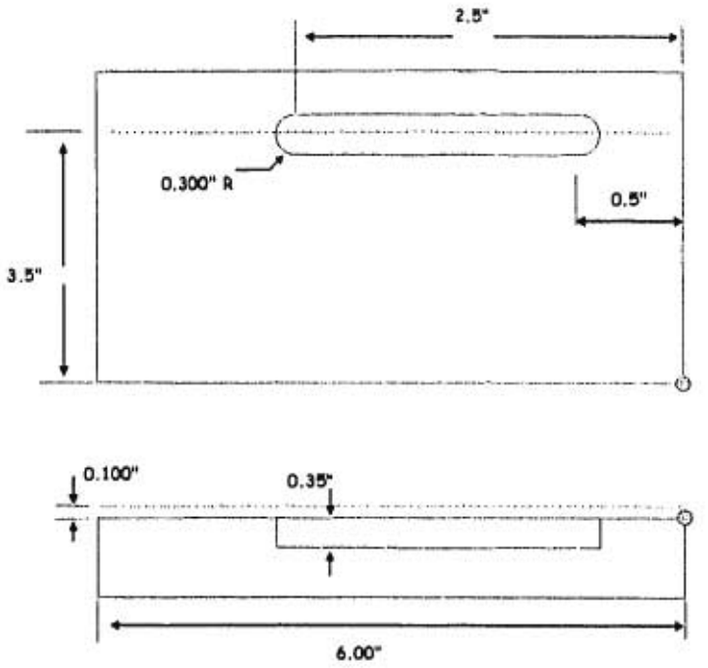


FIGURA 32

Usaremos un cortador de 0.250" .

Cortaremos en desbaste 0.025" en dirección radial ; en Z realizaremos 2 cortes de 0. 175" cada uno.

Nuestras coordenada serán x1 = - 0.5" , y1 = y2 = 3.5" x2 = -2.5" . El ancho del *slot* es 0.300".

Elegimos cortar en sentido horario.

Ingresando estos datos en nuestra macro obtenemos el siguiente código:

```
G00 X-0.5000     Y 3.5000          Z 0.1000
G01                                Z-0.1750     F 20.0000
G02 X-0.5250     Y 3.4750          I-0.0250     F 20.0000
G01 X-2.5000     Y 3.4750                       F 20.0000
G02 X-2.5250     Y 3.5000          J 0.0250     F 20.0000
G02 X-2.5000     Y 3.5250          I 0.0250     F 20.0000
G01 X-0.5000     Y 3.5250                       F 20.0000
G02 X-0.5250     Y 3.5000          J-0.0250     F 20.0000
G02 X-0.5000     Y 3.4750          I-0.0250     F 20.0000
G01 X-0.5250     Y 3.4750                       F 20.0000
G02 X-0.5500     Y 3.5000          J 0.0250     F 20.0000
G01 X-0.5000     Y 3.5000                       F 20.0000
G01                                Z 0.1000     F 20.0000
G01                                Z-0.3500     F 20.0000
G02 X-0.5250   Y 3.4750            I-0.0250     F 20.0000
```

```
G01 X-2.5000   Y 3.4750                          F 20.0000
G02 X-2.5250   Y 3.5000          J 0.0250        F 20.0000
G02 X-2.5000   Y 3.5250          I 0.0250        F 20.0000
G01 X-0.5000   Y 3.5250                          F 20.0000
G02 X-0.5250   Y 3.5000          J-0.0250        F 20.0000
G02 X-0.5000   Y 3.4750          I-0.0250        F 20.0000
G01 X-0.5250   Y 3.4750                          F 20.0000
G02 X-0.5500   Y 3.5000          J0.0250         F 20.0000
G01 X-0.5000   Y 3.5000                          F 20.0000
G01                              Z 0.1000        F 20.0000
```

Que como puedes comprobar realiza el maquinado de la pieza.

Macro programación de un *Slot* curvo:

Este es uno de nuestros ejemplos complejos , en lo referente a la deducción geométrica y matemática del *tool path* .

Un *slot* curvo tiene como finalidad generalmente, el alojar un dispositivo, que transmite un movimiento mecánico periódico; - por ejemplo una leva - hacia la pieza donde se encuentra maquinado el *slot*.

Aunque es una aplicación menos común de maquinado que el *slot* recto; no por ello deja de ser necesaria , además de que es un excelente ejemplo de macro programación .

Un *slot* curvo puede maquinarse en dos direcciones, sentido horario (CW) o anti horario (CCW) - véase figura - . Dependiendo del punto de inicio que elija el operador para colocar el cortador , ya sea inferior o superior será el sentido del maquinado.

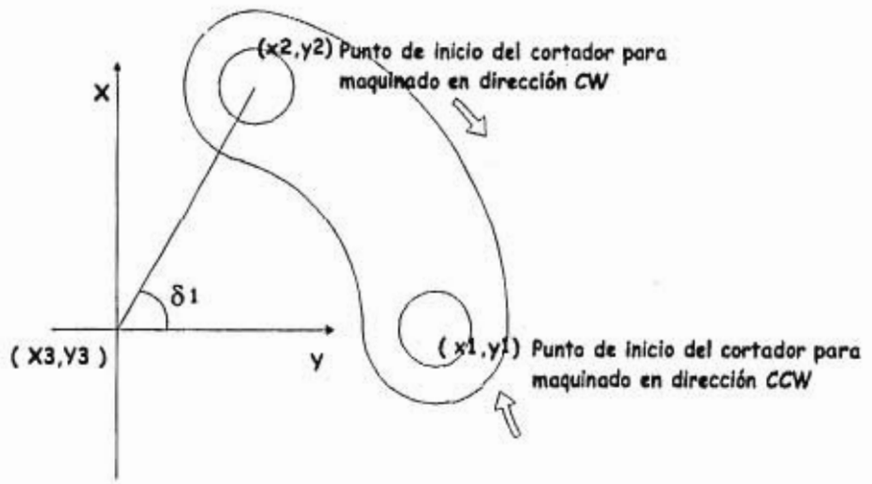


FIGURA 33

Será necesario entonces, generar una macro que tenga ecuaciones capaces de desarrollar el maquinado en una y otra dirección, dependiendo de la elección del usuario para la dirección del maquinado.

Deduciremos únicamente las coordenadas para maquinado en dirección CCW, más por ser un ejemplo complejo, dejaremos indicadas las coordenadas para el *tool path* en dirección CW, esto con el fin de que el lector pueda cotejar sus resultados.

Deducción matemática del *tool path* para *slot* curvo CCW:

De la figura 33 es posible observar, que un *slot* curvo es básicamente dos círculos cuyo diámetro define el ancho del *slot* . El punto de intersección de esos círculos define a su vez, el centro de un tercer círculo , que define el punto de conexión de las dos circunferencias y el punto pivote para la deducción de todas las coordenadas. Por tanto, observamos que hay 3 coordenadas básicas que debemos tomar en cuenta . La primera es el punto 1 con coordenadas x1,y1, que establece el centro del primer círculo y el punto de bajada del cortador para empezar el maquinado en dirección CCW ; el punto 2 con coordenada x2, y2, centro del segundo círculo y punto de bajada del cortador si el maquinado es en dirección CW y el punto pivote con coordenada x3,y3, que establece el centro del *slot*. Serán pues coordenadas que habrá que pedir al operador.

Podríamos pensar también en pedir el radio principal al usuario; pero dado la coordenada del centro de cualquiera de los 2 círculos y conociendo la del centro de nuestro círculo pivote; será fácil deducir este radio.

δ1 Será el ángulo que formen el vector que se forma de los puntos x1,y1 y x3,y3 con el vector que forman x2,y2 ,x3,y3.

Rp será la distancia del centro del círculo principal o pivote a la coordenada x1,y1 o a x2,y2, en otras palabras el radio de dicho círculo.

$$Rp = \sqrt{\left[(X3 - X1)^2 + (Y3 - Y1)^2\right]}$$

Dtx será la profundidad de corte radial en el maquinado.

Denominemos al primer vector, vector [13] , y al segundo vector [23].

Para calcular δ1utilizamos la fórmula de ángulo entre dos vectores:

$$\delta 1 = \cos^{-1}\left[\frac{(X3 - X1)x(X3 - X2) + (Y3 - Y1)x(Y3 - Y2)}{Rp^2}\right]$$

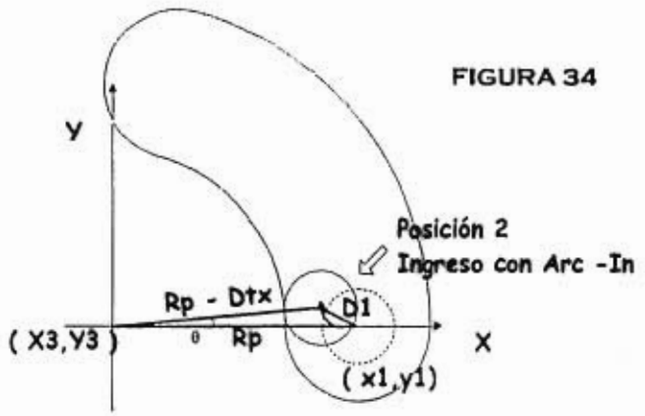
Nuestra primera coordenada, donde bajaremos el cortador en Z para iniciar el maquinado (dirección CCW) , como ya vimos anteriormente será x1, y1.

Antes de deducir la segunda coordenada, la cual realmente es la más difícil de establecer y en la cual reside realmente la dificultad de este ejemplo , es conveniente el asentar las reglas del juego, esto es, el modo en que vamos a encarar el problema geométrico. La geometría del problema nos sugiere el usar coordenadas polares para definir nuestro *tool path*, siendo efectivamente la manera óptima de hacerlo .

Date cuenta de que al calcular la entrada a corte para la segunda coordenada, debemos tener en mente usar la técnica de Arc - In comandando G02 para liberar en lo posible a la herramienta de la presión ejercida por el corte .

Para sacar la segunda coordenada del *tool path* necesitaremos analizar cuidadosamente la siguiente figura .

FIGURA 34



Entre nuestra primera coordenada y la segunda, observamos que se forma un triángulo ; del cual conocemos únicamente 2 de sus lados (Rp y Rp-Dtx). Para poder resolverlo, es necesario otro dato. Te darás cuenta de que no hay manera posible de obtener éste, así pues la única opción que tenemos es proponer ya sea un valor que complete alguna de las variables restantes que nos definen este triángulo o suponer la distancia que debe recorrer el cortador para así poder solucionar el triángulo.

La solución óptima consiste en asignar un valor al ángulo que forman los dos lados que conocemos ; - si te fijas dicho ángulo determinará que tan suave será la entrada del Arc - In hacia la superficie a maquinar - llamemos a dicho ángulo de entrada θ.

En mi experiencia de maquinado he encontrado que el ángulo ideal de entrada deberá tener un valor entre 3° y 10° . Para esta aplicación el valor elegido será de 5° ,el cual nos permite combinar un acceso suave y rápido a la superficie de corte.

Conociendo 0, será fácil establecer las coordenadas de nuestra segunda coordenada:

X_2 = X3 + (Rp - Dtx)x (cos 0).

Y_2 = Y3 +(Rp - Dtx) x (sen 0).

Falta conocer los vectores directores I y J que acompañen a nuestro comando radial G02. De nuestro triángulo anterior , podemos calcular el lado restante , al cual llamaremos D1, por ley de cosenos:

$$D1 = \sqrt{\left[Rp^2 + (Rp - Dtx)^2 - 2(Rp)(Rp - Dtx)\cos\theta\right]}$$

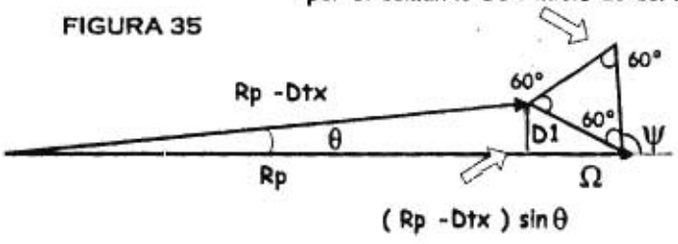
Simplificando llegamos a:

$$D1 = \sqrt{\left[2Rp(1-\cos\theta)x(Rp - Dtx) + Dtx^2\right]}$$

Ahora, ¿ cómo determinar el radio que debe describir el arco de nuestra herramienta?, si eliges hacer el radio igual a D1, al trazar una recta desde el inicio del radio al centro del mismo, notarás que su magnitud es igual a D1, (y por lo tanto igual a la magnitud de I y J) encontrarás lo mismo si trazas una recta desde el fin del arco a su centro, tal como lo muestra la figura 35:

Coordenada del centro del arco descrito por el comando G02 inicio de corte.

FIGURA 35



Hemos formado un triángulo equilátero , el cual puede ayudarnos a simplificar un poco las cosas, si recordamos que los ángulos internos de este triángulo forman 60° cada uno.

De el diagrama anterior, podemos observar como determinar los ángulos Ω y ψ que nos ayudarán a resolver el problema.

$$\Omega = \text{arcsen}\left[\frac{(Rp - Dtx)x\sin\theta}{\sqrt{2Rp(1-\cos\theta)(Rp - Dtx) + Dtx^2}}\right]$$

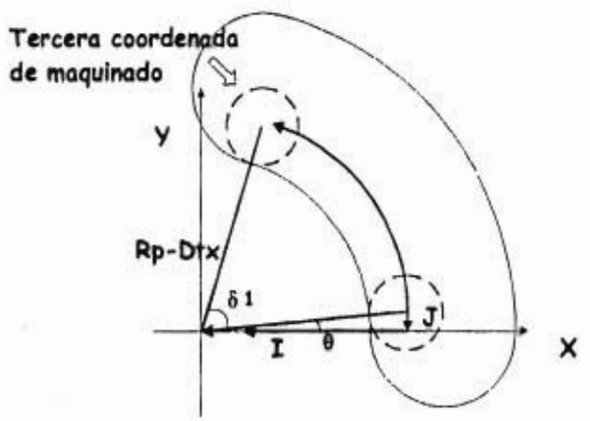
$\psi = 120° - \Omega$

Con esto, ya podremos definir I y J:

$$I = \sqrt{[2Rp(1-\cos\theta)x(Rp - Dtx) + Dtx^2]}\cos\psi$$

$$J = \sqrt{[2Rp(1-\cos\theta)x(Rp - Dtx) + Dtx^2]}\sin\psi$$

La tercera coordenada será dada con comando G03, como podemos observar en la figura 36. La misma figura, nos sirve para calcular las coordenadas y los vectores directores.



$X_3 = x3 + (Rp - Dtx)\ \cos\delta 1$

$Y_3 = y3 + (Rp - Dtx)\ \text{sen}\ \delta 1$

$I = -(Rp - Dtx)\ \cos\theta$

$J = -(Rp - Dtx)\ \text{sen}\ \theta$

FIGURA 36

Para el cálculo de nuestra cuarta coordenada, será necesario calcular otro ángulo , (observa la figura 37), al cual denominaremos θ1. Emplearemos Pitágoras para obtener el radio que se forma a partir del centro de nuestra herramienta con el punto pivote (x3 ,y3). El movimiento de nuestro cortador será con G02.

Nota que los vectores directores I, J representan en este caso, la profundidad radial que penetró el cortador .

De la figura tenemos:

$\theta 1 = \arctan (Dtx / Rp)$

$$X_4 = x3 + \sqrt{[Rp^2 + Dtx^2]}\cos(\theta 1 + \delta 1)$$

$$Y_4 = Y3 + \sqrt{[Rp^2 + Dtx^2]}\operatorname{sen}(\theta 1 + \delta 1)$$

$I = (Dtx) \cos (\delta 1)$

$J = (Dtx) \operatorname{sen} (\delta 1)$

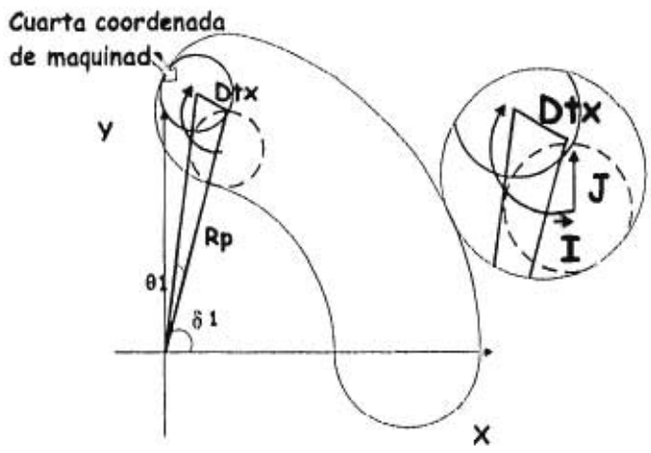


FIGURA 37

Calculemos la quinta coordenada, nuestra sentido seguira siendo G02.
De la figura es fácil observar que:

$X_5 = X3 + (Rp + Dtx) \cos (\delta 1)$

$Y_5 = Y3 + (Rp + Dtx) \operatorname{sen} (\delta 1)$

$I = (Dtx) \operatorname{sen} (\delta 1)$

$J = -(Dtx) \cos (\delta 1)$

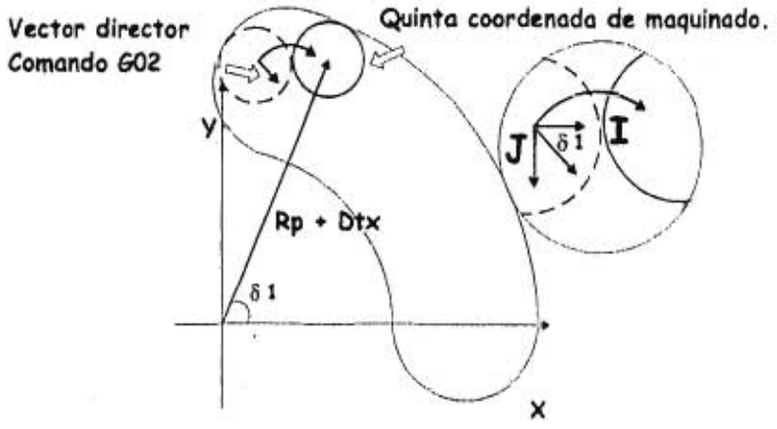


FIGURA 38.

La sexta coordenada será la más fácil de deducir; de la figura es claro que el movimiento del cortador sigue siendo G02:

$X_6 = X3 + (Rp + Dtx)$

$Y_6 = Y3$

$I = - (Rp + Dtx) \cos (\delta 1)$

$J = - (Rp + Dtx) \operatorname{sen} (\delta 1)$

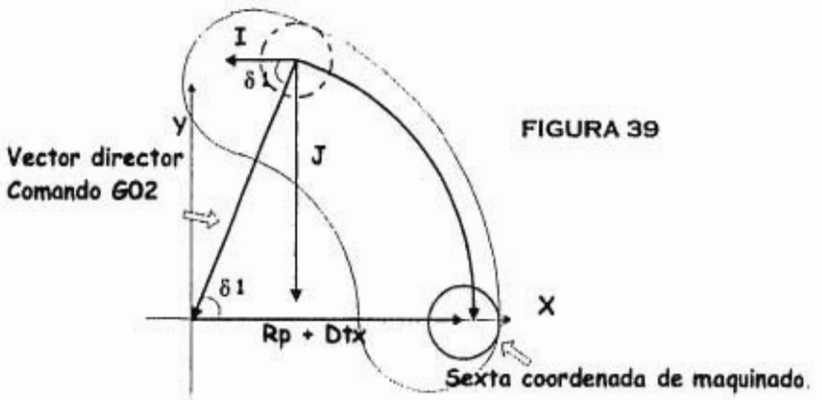


FIGURA 39

En la séptima coordenada, comenzamos a maquinar lo que es círculo inferior de nuestro *slot*, la dirección sigue siendo G02, deduciendo de nuevo las coordenadas tenemos:

$X_7 = X3 + \sqrt{(Rp^2 + Dtx^2)} \cos(\theta 1)$

$Y_7 = Y3 + \sqrt{(Rp^2 + Dtx^2)} \operatorname{sen}(\theta 1)$

$I = - Dtx$

$J = 0$

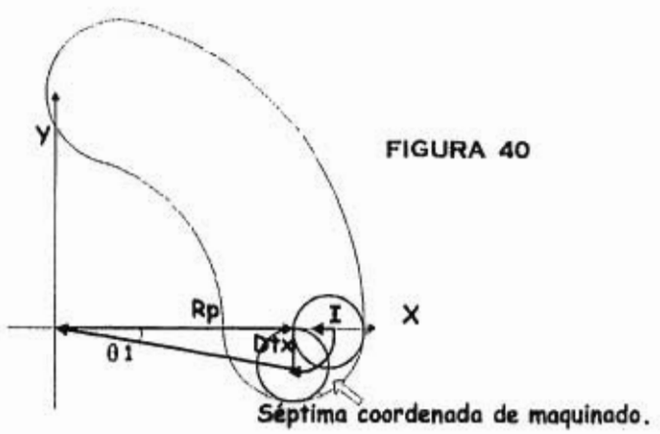


FIGURA 40

La octava coordenada no representa mayor problema (ver figura 41), con ella completamos el maquinado del círculo inferior de nuestro *slot* .La dirección de corte sigue siendo G02:

$X_8 = X3 + Rp - Dtx$ $\qquad I = 0$

$Y_8 = Y3$ $\qquad J = Dtx$

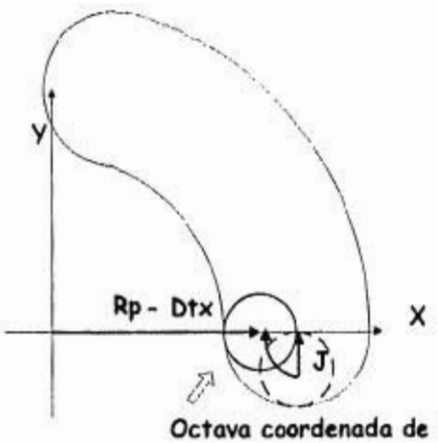


FIGURA 41

Octava coordenada de maquinado.

La novena coordenada, es un poco más difícil de visualizar, aquí tenemos que realizar un cambio en nuestro comando de corte, avanzando en dirección anti horario a partir del punto donde termina nuestro primer círculo maquinando hasta el punto donde contacta la herramienta en el Arc - in , para remover el material que quedó sin maquinar debido al ángulo de entrada θ de la herramienta. De la figura tenemos:

G03 a:

$X_9 = X3 + (Rp - Dtx)\cos \theta$

$Y_9 = Y3 + (Rp - Dtx)\text{sen } \theta$

$I = - (Rp - Dtx)$

$J = 0$

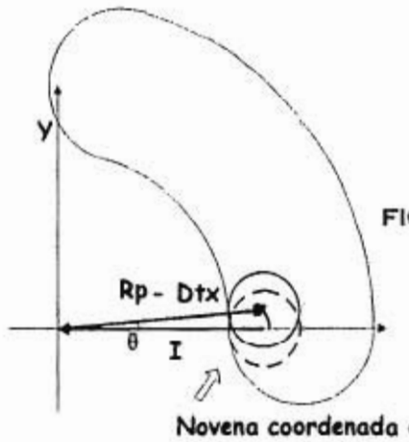


FIGURA 42

Novena coordenada de maquinado.

Para nuestra décima coordenada, usamos la técnica de Arc - Out , para salir del maquinado. La distancia de salida tanto en X , como en Y , será igual a Dtx.

Por lo que nuestras coordenadas serán:

G02 a :

$X_{10} = X3 + (Rp - Dtx)\cos 0 + Dtx$ $\quad I = Dtx$

$Y_{10} = Y3 + (Rp - Dtx)\sin 0 + Dtx$ $\quad J = 0$

Como podemos observar en la figura:

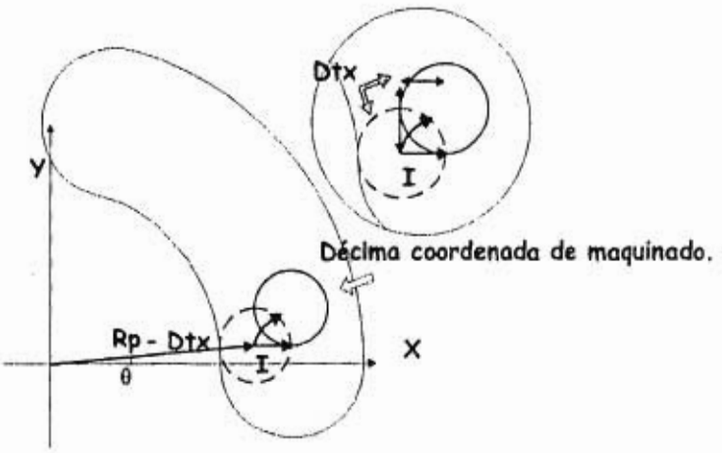


FIGURA 43

Finalmente regresamos el cortador a nuestro punto inicial de corte:

G01 a x1,y1.

Con esto completamos la deducción matemática del *Slot* Curvo, falta únicamente el generalizar nuestras ecuaciones por medio de la rotación de ejes.

Como mencionamos al inicio del planteamiento de esta macro, necesitamos también deducir las ecuaciones para el *tool path* en sentido CW . Por ello presentamos por medio de la siguiente tabla las ecuaciones tanto para sentido horario, como antihorario (ya generalizadas) . Dicha tabla servirá también para resumir nuestro algoritmo ya con ecuaciones.

- El operador elige el sentido de corte (CW o CCW). Ingresa las coordenadas x1,y1 y x2, y2 las cuales serán el punto de bajada del cortador en Z dependiendo del sentido de corte elegido. Ingresa también la coordenada x3,y3 que representa las coordenadas del centro del *slot*, el ancho de éste, el diámetro del cortador que piensa utilizar y la profundidad de corte radial (Dtx).
- Calculamos en base a los datos ingresados las siguientes ecuaciones que sirven para el desarrollo de la macro.

$$Rp = \sqrt{\left[(X3 - X1)^2 + (Y3 - Y1)^2\right]}$$

$$\delta 1 = \cos^{-1}\left[\frac{(X3 - X1)x(X3 - X2) + (Y3 - Y1)x(Y3 - Y2)}{Rp^2}\right]$$

$\psi = 120° - \Omega$

$$\Omega = \text{arcsen}\left[\frac{(Rp - Dtx)x\sin\theta}{\sqrt{2Rp(1 - \cos\theta)(Rp - Dtx) + Dtx^2}}\right]$$

$\theta 1 = \arctan(Dtx / Rp)$

Para CCW:

$$\alpha = \arctan\left(\frac{y3 - y1}{x3 - x1}\right)$$

PARA CW:

$$\alpha = \arctan\left(\frac{y3 - y2}{x3 - x2}\right)$$

Segunda coordenada:

G02 a:

$X_2 = X3 + (Rp - Dtx)\cos(\theta + \alpha)$

$Y_2 = Y3 + (Rp - Dtx)\ \text{sen}(\theta + \alpha)$

$$I = \sqrt{\left[2Rp(1 - \cos\theta)(Rp - Dtx) + Dtx^2\right]}\cos\psi$$

$$J = \sqrt{\left[2Rp(1 - \cos\theta)(Rp - Dtx) + Dtx^2\right]}\sin\psi$$

G03 a:

$X_2 = X3 + (Rp - Dtx)\cos(\delta 1 + \alpha - \theta)$

$Y_2 = Y3 + (Rp - Dtx)\ \text{sen}(\delta 1 + \alpha - \theta)$

$$I = \sqrt{\left[2Rp(1 - \cos\theta)(Rp - Dtx) + Dtx^2\right]}\cos(\psi - \alpha)$$

$$J = \sqrt{\left[2Rp(1 - \cos\theta)(Rp - Dtx) + Dtx^2\right]}\sin(\alpha - \psi)$$

Tercera coordenada:

G03 a:

$X_3 = x3 + (Rp - Dtx)\cos(\delta 1 + \alpha)$

G02 a:

$X_3 = x3 + (Rp - Dtx)\cos\alpha$

Para CCW:

$Y_3 = y3 + (Rp - Dtx) \operatorname{sen} (\delta 1 + \alpha)$

$I = - (Rp - Dtx) \cos (\theta + \alpha)$

$J = - (Rp - Dtx) \operatorname{sen} (\theta + \alpha)$

Para CW:

$Y_3 = y3 + (Rp - Dtx) \operatorname{sen} (\delta 1 + \alpha)$

$I = - (Rp - Dtx) \cos (\delta 1 + \alpha - \theta)$

$J = - (Rp - Dtx) \operatorname{sen} (\delta 1 + \alpha - \theta)$

Cuarta Coordenada:

G02 a:

$$X_4 = x3 + \sqrt{\left[Rp^2 + Dtx^2\right]}\cos(\theta 1 + \delta 1 + \alpha)$$

$$Y_4 = Y3 + \sqrt{\left[Rp^2 + Dtx^2\right]}\operatorname{sen}(\theta 1 + \delta 1 + \alpha)$$

$I = (Dtx) \cos (\delta 1 + \alpha)$

$J = (Dtx) \operatorname{sin} (\delta 1 + \alpha)$

G03 a:

$$X_4 = x3 + \sqrt{\left[Rp^2 + Dtx^2\right]}\cos(\theta 1 - \alpha)$$

$$Y_4 = Y3 + \sqrt{\left[Rp^2 + Dtx^2\right]}\operatorname{sen}(\alpha - \theta 1)$$

$I = (Dtx) \cos \alpha$

$J = (Dtx) \operatorname{sin} \alpha$

Quinta Coordenada:

G02 a:

$X_5 = X3 + (Rp + Dtx) \cos (\delta 1 + \alpha)$

$Y_5 = Y3 + (Rp + Dtx) \operatorname{sen} (\delta 1 + \alpha)$

$I = (Dtx) \operatorname{sen} (\delta 1 + \alpha)$

$J = -(Dtx) \cos (\delta 1 + \alpha)$

G03 a:

$X_5 = X3 + (Rp + Dtx) \cos \alpha$

$Y_5 = Y3 + (Rp + Dtx) \operatorname{sen} \alpha$

$I = - (Dtx) \operatorname{sen} \alpha$

$J = (Dtx) \cos \alpha$

Sexta Coordenada:

G02 a:

$X_6 = X3 + (Rp + Dtx) \cos \alpha$

$Y_6 = Y3 + (Rp + Dtx) \operatorname{sin} \alpha$

G03 a :

$X_6 = X3 + (Rp + Dtx) \cos (\delta 1 + \alpha)$

$Y_6 = Y3 + (Rp + Dtx) \operatorname{sin} (\delta 1 + \alpha)$

Para CCW :

$I = -(Rp + Dtx)\cos(\delta 1 + \alpha)$

$J = -(Rp + Dtx)\,\text{sen}(\delta 1 + \alpha)$

Séptima Coordenada:

G02 a :

$X_7 = X3 + \sqrt{(Rp^2 + Dtx^2)}\cos(\theta 1 + \alpha)$

$Y_7 = Y3 + \sqrt{(Rp^2 + Dtx^2)}\,\text{sen}(\theta 1 + \alpha)$

$I = -(Dtx)\cos\alpha$

$J = -(Dtx)\,\text{sin}\,\alpha$

Octava Coordenada:

G02 a :

$X_8 = X3 + (Rp - Dtx)\cos\alpha$

$Y_8 = Y3 + (Rp - Dtx)\,\text{sin}\,\alpha$

$I = -(Dtx)\,\text{sin}\,\alpha$

$J = (Dtx)\cos\alpha$

Novena Coordenada:

G03 a :

$X_9 = X3 + (Rp - Dtx)\cos(\theta + \alpha)$

$Y_9 = Y3 + (Rp - Dtx)\,\text{sen}(\theta + \alpha)$

Para CW:

$I = -(Rp + Dtx)\cos\alpha$

$J = -(Rp + Dtx)\,\text{sen}\,\alpha$

G03 a :

$X_7 = X3 + \sqrt{(Rp^2 + Dtx^2)}\cos(\delta 1 + \theta 1 + \alpha)$

$Y_7 = Y3 + \sqrt{(Rp^2 + Dtx^2)}\,\text{sen}(\delta 1 + \theta 1 + \alpha)$

$I = -(Dtx)\cos(\delta 1 + \alpha)$

$J = -(Dtx)\,\text{sin}(\delta 1 + \alpha)$

G03 a :

$X_8 = X3 + (Rp - Dtx)\cos(\delta 1 + \alpha)$

$Y_8 = Y3 + (Rp - Dtx)\,\text{sin}(\delta 1 + \alpha)$

$I = -(Dtx)\,\text{sin}(\delta 1 + \alpha)$

$J = (Dtx)\cos(\delta 1 + \alpha)$

G02 a:

$X_9 = X3 + (Rp - Dtx)\cos(\delta 1 + \alpha - \theta)$

$Y_9 = Y3 + (Rp - Dtx)\,\text{sen}(\delta 1 + \alpha - \theta)$

Para CCW:

$I = -(Rp - Dtx)\cos\alpha$

$J = -(Rp - Dtx)\cos\alpha$

Para CW:

$I = -(Rp - Dtx)\cos(\delta 1 + \alpha)$

$J = -(Rp - Dtx)\sin(\delta 1 + \alpha)$

Décima Coordenada:

G02 a :

$X_{10} = X3 + (Rp - Dtx)\cos(\theta + \alpha) + Dtx(\cos\alpha - \sin\alpha)$

$Y_{10} = Y3 + (Rp - Dtx)\sin(\theta + \alpha) + Dtx(\sin\alpha + \cos\alpha)$

$I = (Dtx)\cos\alpha$

$J = (Dtx)\sin\alpha$

G03 a:

$X_{10} = X3 + (Rp - Dtx)\cos(\delta 1 + \alpha - \theta) + Dtx(\cos\alpha - \sin\alpha)$

$Y_{10} = Y3 + (Rp - Dtx)\sin(\delta 1 + \alpha - \theta) + Dtx(\cos\alpha + \sin\alpha)$

$I = -(Dtx)\sin\alpha$

$J = (Dtx)\cos\alpha$

Finalmente regresamos el cortador con avance al punto inicial de partida. Si requerimos seguir cortando radialmente se aumenta la profundidad de corte y utilizamos de nuevo las ecuaciones anteriores.

A continuación el programa en PASCAL que genera el código de maquinado:

```
const
D41=0.1; {Valor de parametro para clearancia en z}
Zinicial=1.0;
S4=1E-5; { Parametro máquina para definir punto flotante}

procedure slotcurvo(x1,y1,x2,y2,ancho,tooldia,Fzd,Fd,Ff,Pcr,Finr,Pcztotal,Pcz,
Zllegada,FinZ,x3,y3,Rp:real;subida,sentidoslot:shortint;var arctrampa:text);
var
alfa,Dtx,Dtz,residuor,residuoz,anchoir,Xtemp,Ytemp,Itemp,Jtemp,teta,tetal,gama,
landa,fil,ft:real;
nr,nz,signo:shortint;
Sentido:string[3];
Banderapirata,BanderaNegra,CCW,BanderaAcabador,BanderaAcabador,Bandera:boolean;
begin
teta:=pi/36;
 If SentidoSlot = 0 then
  begin
  alfa:=angulo(x3,y3,x1,y1);
  CCW:=true;
```

```
  end
 else
  begin
   alfa:=angulo(x3,y3,x2,y2);
   CCW:=false;
  end;
assign(arctrampa,'SLOTCURV.fil');
rewrite(arctrampa);
writeln(arctrampa,'G00 X',x1:6:4,'Y',y1:6:4,'Z',(D41+Zllegada):6:4);
close(arctrampa);
nz:=trunc((Pcztotal-FinZ)/Pcz);
residuoz:=puntoflotante(Pcz*(frac((Pcztotal-FinZ)/Pcz)));
Banderapirata:=true;
 if (nz<=0) then
  Dtz:=residuoz
 else
  Dtz:=Pcz;
BanderaAcabadoz:=false;
Bandera:=false;
While (Dtz<=Pcztotal)and (BanderaPirata) do
   begin
    append(arctrampa);
    if not(BanderaAcabadoz) then
     writeln(arctrampa,'G01 Z',(Zllegada-Dtz):6:4,'F',Fzd:6:4)
    else
     writeln(arctrampa,'G01 Z',(Zllegada-Dtz):6:4,'F',Ff:6:4);
    close(archivo);
   nr:=trunc(((Ancho-Tooldia)/2)-Finr)/Pcr);
   residuor:=puntoflotante((frac(((Ancho-Tooldia)/2)-Finr)/Pcr))*Pcr);
   BanderaNegra:=true;
    if (nr<=0) then
     Dtx:=residuor
    else
     Dtx:=pcr;
   BanderaAcabador:=false;
   While (Dtx+(Tooldia)/2<=ancho/2) and (BanderaNegra) do
    begin
     if BanderaAcabador then
       Ft:=FF
     else
       ft:=Fr;
     tetal:=arctan(Dtx/Rp);
     landa:=acos(((x3-x1)*(x3-x2)+(y3-y1)*(y3-y2))/sqr(Rp));
     gama:=(2*pi/3)-asin((Rp-Dtx)*sin(teta))/(sqrt(2*Rp*(1-cos(teta))*(Rp-Dtx)+sqr(Dtx))));
     fil:=(2*pi/3)-(landa+asin((Rp-Dtx)*sin(teta)/sqrt(2*Rp*(1-cos(teta))*(Rp-
Dtx)+sqr(Dtx))));
     append(arctrampa);
     if CCW then
      begin
       xtemp:=x3+(Rp-DTX)*cos(teta+alfa);
       ytemp:=y3+(Rp-DTX)*sin(teta+alfa);
       Itemp:=sqrt(2*Rp*(1-cos(teta))*(Rp-Dtx)+sqr(Dtx))*cos(gama+alfa);
       Jtemp:=sqrt(2*Rp*(1-cos(teta))*(Rp-Dtx)+sqr(Dtx))*sin(gama+alfa);
       Sentido:='G02';
      end
     else
      begin
       xtemp:=x3+(Rp-DTX)*cos(landa+alfa-teta);
       ytemp:=y3+(Rp-DTX)*sin(landa+alfa-teta);
       Itemp:=sqrt(2*Rp*(1-cos(teta))*(Rp-Dtx)+sqr(Dtx))*cos(fil-alfa);
       Jtemp:=sqrt(2*Rp*(1-cos(teta))*(Rp-Dtx)+sqr(Dtx))*sin(alfa-fil);
       Sentido:='G03';
      end;
writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Ft:6:4
);
     if CCW then
      begin
       xtemp:=x3+(Rp-Dtx)*cos(landa+alfa);
       ytemp:=y3+(Rp-Dtx)*sin(landa+alfa);
       Itemp:=-(Rp-Dtx)*cos(teta+alfa);
       Jtemp:=-(Rp-Dtx)*sin(teta+alfa);
       Sentido:='G03';
```

```
      end
     else
      begin
       xtemp:=x3+(Rp-Dtx)*cos(alfa);
       ytemp:=y3+(Rp-Dtx)*sin(alfa);
       Itemp:=-(Rp-Dtx)*cos(landa+alfa-teta);
       Jtemp:=-(Rp-Dtx)*sin(landa+alfa-teta);
       Sentido:='G02';
      end;
writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Ft:6:4
);
      if CCW then
       begin
        xtemp:=x3+sqrt(sqr(Dtx)+sqr(Rp))*cos(teta1+landa+alfa);
        ytemp:=y3+sqrt(sqr(Dtx)+sqr(Rp))*sin(teta1+landa+alfa);
        Itemp:=Dtx*cos(landa+alfa);
        Jtemp:=Dtx*sin(landa+alfa);
        Sentido:='G02';
       end
      else
       begin
        xtemp:=x3+sqrt(sqr(Dtx)+sqr(Rp))*cos(teta1-alfa);
        ytemp:=y3+sqrt(sqr(Dtx)+sqr(Rp))*sin(alfa-teta1);
        Itemp:=Dtx*cos(alfa);
        Jtemp:=Dtx*sin(alfa);
        Sentido:='G03';
       end;
writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Ft:6:4
);
      if CCW then
       begin
        xtemp:=x3+(Rp+Dtx)*cos(landa+alfa);
        ytemp:=y3+(Rp+Dtx)*sin(landa+alfa);
        Itemp:=Dtx*sin(landa+alfa);
        Jtemp:=-Dtx*cos(landa+alfa);
        Sentido:='G02';
       end
      else
       begin
        xtemp:=x3+(Rp+Dtx)*cos(alfa);
        ytemp:=y3+(Rp+Dtx)*sin(alfa);
        Itemp:=-Dtx*sin(alfa);
        Jtemp:=Dtx*cos(alfa);
        Sentido:='G03';
       end;
writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Ft:6:4
);
      if CCW then
       begin
        xtemp:=X3+(Rp+Dtx)*cos(alfa);
        ytemp:=Y3+(Rp+Dtx)*sin(alfa);
        Itemp:=-(Rp+Dtx)*cos(landa+alfa);
        Jtemp:=-(Rp+Dtx)*sin(landa+alfa);
        Sentido:='G02';
       end
      else
       begin
        xtemp:=X3+(Rp+Dtx)*cos(landa+alfa);
        ytemp:=Y3+(Rp+Dtx)*sin(landa+alfa);
        Itemp:=-(Rp+Dtx)*cos(alfa);
        Jtemp:=-(Rp+Dtx)*sin(alfa);
        Sentido:='G03';
       end;
writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Ft:6:4
);
      if CCW then
       begin
        xtemp:=X3+sqrt(sqr(Rp)+sqr(Dtx))*cos(teta1+alfa);
        ytemp:=Y3+sqrt(sqr(Rp)+sqr(Dtx))*sin(teta1+alfa);
        Itemp:=-Dtx*cos(alfa);
        Jtemp:=-Dtx*sin(alfa);
        Sentido:='G02';
```

```
      end
      else
       begin
        xtemp:=X3+sqrt(sqr(Rp)+sqr(Dtx))*cos(landa+teta1+alfa);
        ytemp:=Y3+sqrt(sqr(Rp)+sqr(Dtx))*sin(landa+teta1+alfa);
        Itemp:=-Dtx*cos(landa+alfa);
        Jtemp:=-Dtx*sin(landa+alfa);
        Sentido:='G03';
       end;
writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Ft:6:4
);
      if CCW then
      begin
       xtemp:=X3+(Rp-Dtx)*cos(alfa);
       ytemp:=Y3+(Rp-Dtx)*sin(alfa);
       Itemp:=-Dtx*sin(alfa);
       Jtemp:=Dtx*cos(alfa);
       Sentido:='G02';
      end
      else
       begin
       xtemp:=X3+(Rp-Dtx)*cos(landa+alfa);
       ytemp:=Y3+(Rp-Dtx)*sin(landa+alfa);
       Itemp:=Dtx*sin(landa+alfa);
       Jtemp:=-Dtx*cos(landa+alfa);
       Sentido:='G03';
      end;
writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Ft:6:4
);
      if CCW then
       begin
        xtemp:=X3+(Rp-Dtx)*cos(teta+alfa);
        ytemp:=Y3+(Rp-Dtx)*sin(teta+alfa);
        Itemp:=-(Rp-Dtx)*cos(alfa);
        Jtemp:=-(Rp-Dtx)*sin(alfa);
        Sentido:='G03';
       end
       else
        begin
        xtemp:=X3+(Rp-Dtx)*cos(landa+alfa-teta);
        ytemp:=Y3+(Rp-Dtx)*sin(landa+alfa-teta);
        Itemp:=-(Rp-Dtx)*cos(landa+alfa);
        Jtemp:=-(Rp-Dtx)*sin(landa+alfa);
        Sentido:='G02';
       end;
writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Ft:6:4
);
      if CCW then
       begin
        xtemp:=X3+(Rp-Dtx)*cos(teta+alfa)+Dtx*(cos(alfa)-sin(alfa));
        ytemp:=Y3+(Rp-Dtx)*sin(teta+alfa)+Dtx*(sin(alfa)+cos(alfa));
        Itemp:=Dtx*cos(alfa);
        Jtemp:=Dtx*sin(alfa);
        Sentido:='G02';
       end
      else
        begin
        xtemp:=X3+(Rp-Dtx)*cos(landa+alfa-teta)+Dtx*(cos(alfa)-sin(alfa));
        ytemp:=Y3+(Rp-Dtx)*sin(landa+alfa-teta)+Dtx*(sin(alfa)+cos(alfa));
        Itemp:=-Dtx*sin(alfa);
        Jtemp:=Dtx*cos(alfa);
        Sentido:='G03';
       end;
writeln(arctrampa,Sentido,'X',xtemp:6:4,'Y',ytemp:6:4,'I',Itemp:6:4,'J',Jtemp:6:4,'F',Ft:6:4
);
      xtemp:=X1;
      ytemp:=Y1;
      writeln(arctrampa,'G01 X',xtemp:6:4,'Y',ytemp:6:4,'F',Fr:6:4);
      close(arctrampa);
      nr:=nr-1;
       if (nr=0) and (residuor<>0) then
         Dtx:=Dtx+residuor
```

```
      else if (nr=0) and (residuor=0) then
        begin
        if (finr=0) then
         BanderaNegra:=false
        else
         begin
          BanderaAcabador:=true;
          Dtx:=Dtx+Finr;
         end;
        end
      else if (nr<0) and (residuor<>0) then
        begin
        if (finr=0) then
         BanderaNegra:=false
        else
         begin
          BanderaAcabador:=true;
          Dtx:=Dtx+Finr;
         end;
       end
      else if (nr>0) then
      Dtx:=Dtx+Pcr;
    end; {fin del ciclo para fresar en desbaste}
   nz:=(nz-1);
    if (nz=0) and (residuoz <>0) then
      Dtz:=Dtz+residuoz
    else if (nz=0) and (residuoz=0) then
     begin
      if (finz=0) then
       BanderaPirata:=false
      else
       begin
        Dtz:=Dtz+finz;
        BanderaAcabadoz:=true;
        Bandera:=true;
        end
       end
     else if (nz<0) and (residuoz<>0) then
       begin
        if (finz=0) then
         Banderapirata:=false
       else
         begin
          Dtz:=Dtz+Finz;
          BanderaAcabadoz:=true;
          Bandera:=true;
         end
        end
     else if (nz>0) then
      Dtz:=Dtz+Pcz;
    If subida=1 then
      begin
       append(arctrampa);
       writeln(arctrampa,'G01 Z',(D41+Zllegada):6:4,'F',Fzd:6:4);
      end;
  end; {fin del ciclo para bajar en z}
end;
```

Ejemplo para la macro de *Slot* curvo:

Resta maquinar dos *slots* curvos para la pieza que hemos careado previamente en forma circular.

Dado que el ancho del *slot* es de 0.400" , emplearemos un cortador de 3/8 ". (0.375).

La profundidad de corte radial será de 0.0125" , la profundidad de corte en Z será de 0.250".

El cero pieza está ubicado en el centro de nuestra geometría; el *slot* ubicado en el primer cuadrante lo maquinaremos en sentido horario, el que está ubicado en el tercero lo haremos en sentido antihorario.

Las coordenadas del primer *slot* serán:

x1 = 1.75 x cos 86° = 0.1221 y1 = 1.75 x sin 86° = 1.7457

x2 = 1.75 y2 = 0

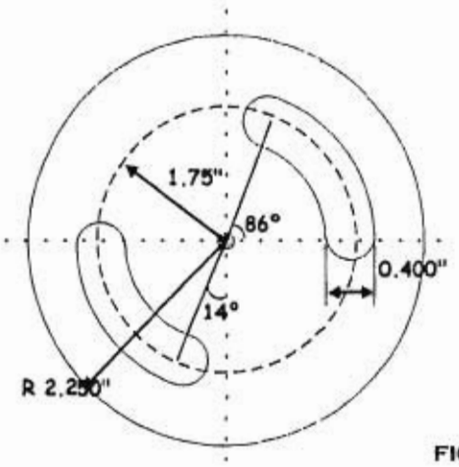


FIGURA 44

Carear 0.240"

Con estas coordenadas obtenemos el siguiente código de maquinado:

```
G00 X 0.1221 Y 1.7457          Z 0.1000
G01                            Z-0.2500        F 10.0000
G02 X-0.0303 Y 1.7372 I-0.0836 J 0.1277        F 20.0000
G03 X-1.7206 Y 0.2418 I 0.0303 J-1.7372        F 20.0000
G02 X-1.7347 Y 0.2312 I-0.0125 J 0             F 20.0000
G02 X-1.7453 Y 0.2453 I 0      J 0.0125        F 20.0000
G02 X 0.1229 Y 1.7582 I 1.7453 J-0.2453        F 20.0000
G02 X 0.1096 Y-1.7466 I 0      J-0.0125        F 20.0000
G02 X 0.1212 Y 1.7333 I-0.0125 J 0             F 20.0000
G03 X-0.0303 Y 1.7372 I-0.1212 J-1.7333        F 20.0000
G02 X-0.0419 Y 1.7506 I 0      J 0.0125        F 20.0000
G01 X 0.1221 Y 1.7457                          F 20.0000
G01                            Z-0.5000        F 10.0000
G02 X-0.0303 Y 1.7372 I-0.0836 J 0.1277        F 20.0000
```

```
G03 X-1.7206 Y 0.2418 I 0.0303 J-1.7372          F 20.0000
G02 X-1.7347 Y 0.2312 I-0.0125 J 0               F 20.0000
G02 X-1.7453 Y 0.2453 I 0      J 0.0125          F 20.0000
G02 X 0.1229 Y 1.7582 I 1.7453 J-0.2453          F 20.0000
G02 X 0.1096 Y-1.7466 I 0      J-0.0125          F 20.0000
G02 X 0.1212 Y 1.7333 I-0.0125 J 0               F 20.0000
G03 X-0.0303 Y 1.7372 I-0.1212 J-1.7333          F 20.0000
G02 X-0.0419 Y 1.7506 I 0      J 0.0125          F 20.0000
G01 X 0.1221 Y 1.7457                            F 20.0000
G01                            Z 0.100           F 10.0000
G00                            Z 1.0000
```

Para el segundo *slot* nuestras coordenadas son:

x1 = -1.75 y1 = 0

x2 = 1.75 x cos 256° = -0.4233 y2 = 1.75 x sin 256° = -1.698

Maquinando en CCW obtenemos el siguiente código:

```
G00 X-1.7500 Y 0                Z 0.1000
G01                             Z-0.2500          F 10.0000
G02 X-1.7309 Y-0.1514 I-0.1216 J-0.0923          F 20.0000
G03 X-0.4203 Y-1.6859 I 1.7309 J 0.1514          F 20.0000
G02 X-0.4112 Y-1.7011 I 0      J-0.0125          F 20.0000
G02 X-0.4263 Y-1.7102 I-0.0125 J 0               F 20.0000
G02 X-1.7625 Y 0      I 0.4263 J 1.7102          F 20.0000
G02 X-1.7500 Y 0.0125 I 0.0125 J 0               F 20.0000
G02 X-1.7375 Y 0      I 0      J-0.0125          F 20.0000
G03 X-1.7309 Y-0.1514 I 1.7375 J 0               F 20.0000
G02 X-1.7434 Y-0.1639 I-0.0125 J 0               F 20.0000
G01 X-1.7500 Y 0                                 F 20.0000
G01                             Z-0.5000          F 10.0000
G00 X-1.7500 Y 0                Z 0.1000
G01                             Z-0.2500          F 10.0000
G02 X-1.7309 Y-0.1514 I-0.1216 J-0.0923          F 20.0000
G03 X-0.4203 Y-1.6859 I 1.7309 J 0.1514          F 20.0000
G02 X-0.4112 Y-1.7011 I 0      J-0.0125          F 20.0000
G02 X-0.4263 Y-1.7102 I-0.0125 J 0               F 20.0000
G02 X-1.7625 Y 0      I 0.4263 J 1.7102          F 20.0000
G02 X-1.7500 Y 0.0125 I 0.0125 J 0               F 20.0000
G02 X-1.7375 Y 0      I 0      J-0.0125          F 20.0000
G03 X-1.7309 Y-0.1514 I 1.7375 J 0               F 20.0000
G02 X-1.7434 Y-0.1639 I-0.0125 J 0               F 20.0000
G01 X-1.7500 Y 0                                 F 20.0000
G01                             Z 0.1000          F 10.0000
G00                             Z 1.000
```

Ambos códigos completan el maquinado de los *slots* y el de nuestra pieza.

Conclusiones de Tesis:

Este texto ha pretendido exponer la macro programación como herramienta simplificadora del trabajo de programar una máquina CNC de la manera convencional.

Obviamente el lector, a través de la lectura del mismo, podrá obtener sus conclusiones acerca de las ventajas que representa el dominar esta herramienta; pero no está de más el exponer las mías para en conjunto redondearlas.

- La macro programación permite generar de manera rápida, eficiente y segura código de maquinado EIA – ISO para cualquier patrón de mecanizado que se pueda modelar matemáticamente.
- La macro programación permite automatizar operaciones a través de diseñar macros para maquinar familias de partes.
- Los macro programas permite que cualquier persona con pocos conocimientos en código EIA – ISO programe una máquina CNC en relativamente poco tiempo, por consiguiente reduce la necesidad de conseguir técnicos especializados en programación CNC para operar este tipo de maquinaria
- La macro programación en combinación con técnicas avanzadas de programación es una herramienta altamente flexible.
- Un macro programa permite modificar instantáneamente datos de corte, no importando la longitud del código de maquinado que se genere, cambiando únicamente las variables necesarias dentro del macroprograma.
- La macro programación es una herramienta muy poderosa.

Bibliografia:

Programming Manual For Mazatrol M Plus (EIA – ISO) For VTC –16 Machining Center.

A Mazak Corporation Publication

Catalog. Number H733PB0011E