Universidad Autónoma de Querétaro
Facultad de Ingeniería
Doctorado de Ingeniería

SISTEMA DE CONTROL NUMÉRICO COMPUTARIZADO DE ARQUITECTURA ABIERTA, MODULAR Y PORTÁTIL

TESIS

Que como parte de los requisitos para obtener el grado de
Doctor en Ingeniería

Presenta:

M. en C. Roberto Augusto Gómez Loenzo

Dirigido por:

Dr. Gilberto Herrera Ruiz

SINODALES

Dr. Gilberto Herrera Ruiz Presidente	Firma
Dr. Pedro Daniel Alaniz Lumbreras Secretario	Firma
Dr. René de Jesús Romero Troncoso Vocal	Firma
Dra. Rebeca del Rocío Peniche Vera Suplente	Firma
Dr. Ramón Gerardo Guevara González Suplente	Firma

Dr. Gilberto Herrera Ruiz
Director de la Facultad

Dr. Luis Gerardo Hernández Sandoval
Director de Investigación y Postgrado

Centro Universitario
Querétaro, Qro.
Junio de 2009
México

RESUMEN

Desde la década de 1990 se han propuesto diversas arquitecturas abiertas para sistemas de Control Numérico por Computadora (CNC) con el propósito de estandarizar la forma en que estos sistemas son fabricados y ensamblados y, así, poder facilitar la colaboración entre distintos fabricantes de equipo, lo cual permitiría la entrada al mercado de nuevos competidores y la correspondiente reducción de precios para el usuario final. Sin embargo, estos estándares propuestos no han recibido gran aceptación por parte de los fabricantes de equipo de CNC debido a que su adopción implicaría un cambio drástico en su modelo de negocios, tanto desde el punto de vista de su posición dominante en el mercado como desde la manera en que ellos mismos diseñan, prueban y fabrican sus productos.

En consecuencia, la aceptación de estos estándares depende no solo de la disposición de los principales fabricantes a cambiar de modelo de negocios sino a cambiar su base tecnológica, es decir, el producto en sí que están comercializando. Por esta razón el presente trabajo propone no estandarizar la base tecnológica de los fabricantes de equipo de CNC sino estandarizar su proceso de desarrollo y permitir que la estandarización del producto final resultante ocurra *de facto*, es decir, en la práctica de forma paulatina.

La propuesta del presente trabajo es desarrollar un sistema de CNC para centros de maquinado con base en un proceso estandarizado de desarrollo de software. El enfoque propuesto se basa en definir de manera precisa los requerimientos del sistema de CNC por medio de casos de uso y con base en los mismos dar inicio al proceso de diseño definiendo colaboraciones que realicen a los casos de uso por medio de diagramas de secuencia. Los diferentes componentes empleados en los diagramas de secuencia servirán como la base para la modularización y posterior diseño e implementación del sistema de control. Se presenta el caso de estudio para un centro de maquinado. Si un fabricante ya cuente con una base tecnológica existente para el desarrollo del sistema, bastará con adaptar la misma al proceso de desarrollo propuesto para la creación de nuevos productos.

(**Palabras clave:** Control Numérico por Computadora (CNC), arquitectura abierta, modularización, proceso de desarrollo de software.)

SUMMARY

Since the 1990s various open architectures for Computer Numerical Control (CNC) systems have been proposed in order to standardize the way these systems are manufactured and assembled and, thus, facilitating collaboration between different equipment manufacturers, which would allow the entry of new providers into the market with a corresponding reduction in prices for the end user. However, these proposed standards have not received wide acceptance by manufacturers of CNC equipment, because its adoption would mean a drastic change in their business model, both in terms of their dominant position in the market and in the way they design, test and manufacture their products.

Consequently, the acceptance of these standards depends not only on the willingness of the major manufacturers to change their business model but to change their technology base, i.e., the product itself being marketed. For this reason this paper proposes not to standardize the technology base of the manufacturers of CNC equipment, but to standardize their development process and enable the standardization of the resulting final product to happen *de facto*, i.e., gradually in practice.

The proposal of this work is to develop a system for CNC machining centers using a standardized software development process. The proposed approach is based on defining the precise requirements of the CNC system through use cases, and based on them to initiate the design process to define collaborations realizing the use cases by sequence diagrams. The various components used in the sequence diagrams serve as the basis for the modularization and subsequent design and implementation of the control system. We present a case study for a machining center. If a manufacturer already has an existing technology base to develop the system, adapting it to the proposed development process for creating new products will suffice.

(**Key words:** Computer Numerical Control (CNC), open architecture, modularization, software development process.)

A Dios, a mi familia (en especial a mi hermana Claudia, q.e.p.d.), a mis amigos (en especial a Carlos Coria Silva, q.e.p.d.), a mi país
—por hacer todo esto posible.
Gracias por todo.

AGRADECIMIENTOS

Este trabajo de investigación no habría sido posible sin la ayuda del personal de la Facultad de Ingeniería de la Universidad Autónoma de Querétaro. Deseo agradecer al personal de mantenimiento que tanto nos ha apoyado en el laboratorio, en particular a Victor, y al personal administrativo que siempre nos hace la vida más fácil, en particular a la Sra. Gloria y a Felipe (y que muchas veces hacen cosas extraordinarias sin que se los reconozcan y solo se acuerdan de ellos cuando las cosas no son perfectas). Quiero también agradecer a los estudiantes que hay ayudado durante las investigaciones tal como Rafael Porrás que programó el analizador léxico para código G siguiendo el diseño de máquina de estado que le proporcioné.

También quiero agradecer a mis asesores por todo el apoyo que me han dado a pesar de no estar siempre de acuerdo con mis decisiones. Gracias Dr. Gilberto por siempre tenerme paciencia a pesar de mis soluciones a veces ‘demasiado matemáticas’ para un área de ingeniería. Estos agradecimientos no serían lo mismo sin un agradecimiento especial para la Dra. Peniche que tuvo la paciencia de escuchar horas de explicaciones técnicas sobre software y centros de maquinado y no perder la paciencia en el proceso y por recordarme siempre que hay un plan a seguir aún cuando las investigaciones no siempre parecen dar los resultados que uno espera.

Sin mis amigos tampoco habría sido posible hacer este desarrollo; no los menciono porque son muchos, pero saben que siempre me acuerdo de ustedes y les agradezco sus buenos deseos.

Gracias a mi familia por apoyarme durante mis estudios, en especial a mi mamá y a mi hermana (q.e.p.d.).

Gracias a mi país y a Dios.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Parte I

Análisis

I. INTRODUCCIÓN

I.1. Metalmecánica en la economía mexicana

La economía mexicana se encuentra actualmente inmersa en un proceso de globalización que le exige competir con economías desarrolladas y otras economías emergentes. Mientras que las economías desarrolladas han comenzado a migrar su fuerza laboral hacia el sector de servicios, otras economías emergentes, tales como la China, ofrecen una competencia directa en el sector de manufactura. Los bajos precios de los productos chinos y de otros países emergentes en el mercado internacional obligan a las empresas mexicanas que deseen exportar sus productos a ofrecer, un menor precio o una mayor calidad. En virtud de que la mano de obra china y de otras economías emergentes es de menor costo, se vuelve necesario para las industrias mexicanas ofrecer una mayor calidad en el mercado internacional, lo cual implica que se debe cumplir con los estándares de calidad internacionales. Para este fin es necesario introducir tecnologías que mejoren la calidad de los productos mexicanos sin elevar drásticamente su costo, así como ofrecer productos de alto valor agregado. Como buena parte del costo de la manufactura de los productos en la economía mexicana es derivado del costo de la mano de obra y de que los estándares de calidad internacionales requieren de una alta precisión durante el proceso de manufactura, es menester introducir métodos de manufactura que reduzcan la necesidad de intervención humana, es decir, se requiere automatizar en la medida de lo posible los procesos de manufactura. Las máquinas herramienta son un aspecto muy importante en el proceso de automatización industrial ya que permiten eliminar tanto el costo como la imprecisión asociados a la fuerza laboral humana.

A partir de la entrada en vigor del Tratado de Libre Comercio para América del Norte (TLCAN) y de otros tratados comerciales firmados en años recientes, se ha incrementado grandemente la demanda de productos del área metalmecánica. Por esta razón, la industria metalmecánica ha incrementado su relevancia para la economía mexicana en años recientes. Tan sólo durante el periodo 2002–2004 el sector mexicano de manufactura correspondió al 19 % del total del Producto Interno Bruto (PIB), al mismo tiempo que la participación de la industria metalmecánica en el PIB generado por la manufactura fue del 29 %. El subsector de manufactura con el crecimiento más substancial, en el mismo periodo, fue la industria metalmecánica con un incremento estimado del 15 %. El submercado más importante para las ventas de máquinas herramienta y equipo para trabajo en metal son los fabricantes de aparatos electrodomésticos y fabricantes de autopartes.

Según la Asociación Mexicana de Distribuidores de Maquinaria (AMDM), las perspectivas de demanda para 2005 son alentadoras, en virtud de la cantidad de proyectos de gran envergadura, principalmente en el centro y norte de México. Además, el Consejo Nacional de la Industria Maquiladora de Exportación (CNIME) espera una mayor demanda de máquinas herramienta en virtud de que se eliminarán algunas restricciones para la entrada de las

mismas al mercado doméstico.

Todo esto sugiere una demanda que se encuentra actualmente limitada únicamente por la falta de recursos financieros. La falta de capital y los créditos costosos (cuando los hay) para la adquisición de equipo nuevo ha forzado a muchos fabricantes mexicanos a buscar alternativas como la actualización de su equipo. En muchos casos una buena alternativa ha sido comprar equipo usado, darle mantenimiento, y reacondicionarlo, o bien, comprar maquinaria ya reacondicionada de los Estados Unidos. Según la AMDM, un 40 % del equipo adquirido por fabricantes mexicanos es reacondicionado. Esto ha sido posible principalmente debido a la disponibilidad de partes de repuesto y a la tecnología que facilita las conversiones. La importación de partes, ya sean mecánicas, eléctricas o electrónicas, debido al TLCAN, se han vuelto mucho más sencillas y rápidas.

Buena parte del costo del reacondicionamiento de maquinaria consiste en la adaptación de un controlador numérico por computadora (CNC) al equipo, así que el desarrollo de tecnología nacional a menor costo de lo que se ofrecían el mercado permitiría ofrecer una ventaja competitiva a los fabricantes nacionales y, en particular, a los de la región. Un CNC basado en PC ofrece una excelente razón costo/beneficio, así que el desarrollo de ésta tecnología a partir de una arquitectura abierta, modular y portátil ofrece grandes beneficios económicos potenciales para la industria mexicana.

I.2. Evolución de las máquinas herramienta

A lo largo de su historia el hombre siempre a diseñado artefactos que le faciliten las tareas cotidianas; desde la invención de la rueda, la humanidad se ha esforzado por hacer cada día más fácil su trabajo y en gran medida se le debe a las nuevas necesidades que surgen continuamente. En el pasado la manufactura de partes metálicas era hecha a mano; la persona encargada de hacerlo utilizaba una herramienta para desbastar, una para troquelar, otra para perforar, etc., y casi cada tarea a realizar requería de una herramienta especial para hacerlo. Independientemente del proceso de fabricación utilizado o las herramientas empleadas, el hombre necesitaba realizar todo el trabajo físico; afortunadamente, con el avance de la tecnología el trabajo se fue realizando paulatinamente sin tanta intervención de la mano del hombre.

Una *máquina herramienta* (del inglés, *machine tool*[1]) es un dispositivo mecánico, utilizado generalmente para producir componentes metálicos de maquinaria mediante la eliminación selectiva de metal. Se considera que las primeras máquinas herramienta aparecieron cuando se eliminó la participación directa del hombre sobre la manufactura de piezas. Por ejemplo, el primer torno, el cual es un caso particular de máquina herramienta, fue inventado por Jacques de Vaucanson en 1751, quien implementó ajustar el instrumento de corte en una cabeza ajustable mecánicamente, con lo cual el operario no realizaba el trabajo físico de la manufactura. Sin embargo no fue hasta el año de 1800 que Henry Maudslay inventó el primer torno metalúrgico práctico, el cual era simplemente una herramienta que sujetaba al material deseado con una abrazadera y rotaba, devastando el material hasta conseguir la forma deseada. Una de las primeras fresadoras, que es otra clase de máquina herramienta, fue inventada en 1818 por Eli Whitney, la cual consistía de una mesa, donde se ponía el material

[1]Tal vez una traducción más adecuada podría ser *herramientas [de/para] máquina*, en virtud de que en este caso la palabra *machine* es un adjetivo de la palabra *tool*. Desafortunadamente, el término ‘máquina herramienta’ se ha estandarizado, y se asume incorrectamente que en este caso ‘herramienta’ es un adjetivo sin inflexiones.

que fuera a ser trabajado, y un husillo, al cual se le fijaba una herramienta de corte; el proceso de manufactura se realizaba moviendo la mesa alrededor y debajo de la herramienta de corte tanto como se necesitara para darle la forma final al material.

Con el tiempo se crearon mejores máquinas herramienta perfeccionando principalmente los aspectos de precisión e independencia del hombre. Los movimientos que se hacían en las primeras máquinas herramienta estaban controlados por un complejo sistema de engranes, palancas, manubrios y manivelas, que se giraban dependiendo de los diales graduados en los manubrios y se movía la herramienta de corte una cantidad apropiada. Cada parte que se hacia con estas máquinas requirió al operador repetir los movimientos en la misma secuencia y con las mismas dimensiones, lo que significaba un trabajo muy agotador y monótono; además la cantidad de piezas iguales era limitado y se desperdiciaba una gran cantidad de material.

I.3. Evolución del control numérico

Afortunadamente, desde la invención de esta clase de maquinaria, han habido avances continuos en su tecnología debido a las nuevas necesidades que surgen cada día en la producción de partes mecánicas. Entre las principales características que se han mejorado están: productividad, calidad, rapidez, precisión en cortes y flexibilidad. Estas necesidades en la industria, y el gran avance que ha habido en el área de microelectrónica y computación, han contribuido a que día a día las máquinas herramienta dependan menos de la mano del hombre a tal grado de que la máquina haga casi todo el trabajo. Este proceso evolutivo dio origen al control numérico (CN).

En efecto, el crecimiento industrial junto con los beneficios que el desarrollo de las computadoras electrónicas aportó en la precisión de los cálculos impulsaron el desarrollo a una nueva era de máquinas herramienta que empleaba esta tecnología en la exactitud de los cortes hechos por la herramienta. La idea del CN nació hacia finales de los 40's y principios de los 50's cuando John T. Parsons, de la Corporación Parsons, en 1947 diseñaba plantillas para las hélices de helicópteros. Parsons junto con Frank Stulen tenían 17 puntos en cada lado de una plantilla, los cuales eran unidos con una curva francesa y después estas plantillas se trasladaban a la sección de corte. En ese entonces la tolerancia de error que se tenia era de ± 0.007 pulgadas, sin embargo, Parsons quería construir con mayor precisión las helices y le propuso a Stulen extrapolar los 17 puntos a 200, lo que daría una separación menor de 0.0015 pulgadas. Stulen atendió de inmediato el pedido de Parsons e hizo un programa en tarjetas perforadas para hacer los cálculos de los datos necesarios. En poco tiempo la corporación Parsons estaba haciendo plantillas más rápido y exactas como nunca antes se había hecho.

Cuando, en 1948, John T. Parsons recibió unos dibujos de la Base Wright-Patterson de la Fuerza Área, que eran los planes para hacer una nueva ala de avión, se llegó a la conclusión de que el ala estaba muy pesada para el vuelo por lo cual debía de ser modificada y se proponía un nuevo diseño que no podía ser realizado por ninguna máquina de la época: se tenía que desbastar la superficie del ala un espesor determinado aproximadamente 800 veces y era imposible concebir que un hombre, trabajando 8 horas al día, no se equivocara. En ese momento Parsons pensó que lo que estaban haciendo con las helices en dos ejes se podía aplicar a este problema adelgazando el ala en tres dimensiones y empezó a trabajar con la idea de ajustar los datos de una curva al movimiento de una maquina de tres ejes. La Corporación

Figura 1.1: John T. Parsons, precursor del CNC.

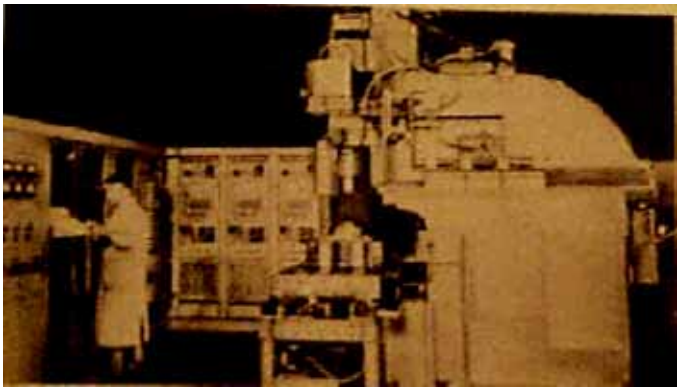


Figura 1.2: Fresadora del MIT.

Parsons dio a conocer esta idea y para el 3 de Diciembre del mismo año once expertos de la Fuerza Aérea fueron a la Corporación Parsons para conocer más a fondo el proyecto, y en junio de 1949 se concedió el contrato a John T. Parsons, quien contrató a tres empresas para realizar una máquina capaz de hacer lo propuesto: la Corporación Snyder, que se encargaría de construir la máquina herramienta, IBM, quien equiparía a la máquina herramienta con un lector de cintas magnéticas o tarjetas y el MIT (Massachusetts Institute of Technology) para instalar los servomecanismos a la máquina. Finalmente la Fuerza Aérea contrató y compro 120 equipos que repartió a través de las industria aeronáutica para probar y empezar a usar el proceso. En 1951 el seguimiento del proyecto fue asumido por el MIT, quienes habían estado experimentando con varios tipos de procesos de control y tenían experiencia con otros proyectos de la Fuerza Aérea que databan de la Segunda Guerra Mundial y vieron esto como una oportunidad de ampliar su propia investigación en mecanismos de control. En 1952 el MIT muestra la primera máquina de CN: una gran fresadora que fabricaba piezas exitosamente con movimientos de herramientas de corte simultáneos en los tres ejes (figura 1.2).

Se considera de control numérico a todo dispositivo capaz de dirigir el posicionamiento de un órgano mecánico móvil mediante órdenes elaboradas de forma totalmente automática a partir de informaciones numéricas en tiempo real. Se le llama de control numérico por que las ordenes que se le dan a la máquina herramienta son mediante instrucciones escritas en forma de números que se almacenan en registros especiales de una computadora.

Tras pasar por numerosas modificaciones y grandes avances la máquina de CN llegó a la industria. El proceso de producción empleando una máquina herramienta comenzaba en una oficina donde ingenieros programaban un dispositivo de datos, generalmente una cinta perforada, con la información necesaria para la mecanización de una pieza en forma de códigos. Cuando la rutina era demasiado compleja se utilizaba una computadora, como auxiliar, para las operaciones de cálculo y almacenamiento de órdenes. Después la cinta era colocada en un lector de cintas que estaba en la máquina herramienta y ésta realizaba los movimientos necesarios para la manufactura. Para obtener la precisión deseada y la forma de la pieza a trabajar la máquina era ajustada en otro departamento especial antes de colocarle el dispositivo de datos.

Conforme la ciencia de la computación avanzó y los desarrollos tecnológicos en el área de microelectrónica crecieron, el uso de la computadora en las máquinas de control numérico fue cada vez mayor; esto trajo un nuevo concepto de las máquinas con las que se hacia la manufactura de piezas llamadas máquinas de control numérico computarizado (CNC) donde todo el proceso de producción estaba *en una sola máquina.*

Un CNC es un sistema que permite controlar el movimiento de las máquinas herramienta y sus partes mediante el uso de una computadora que utiliza códigos numéricos. A diferencia del proceso de producción en las máquinas de CN, aquí es posible trabajar directamente sobre la máquina, es decir el programa de control se puede desarrollar en la misma computadora que es parte de la máquina herramienta mediante el uso de un teclado incorporado a la máquina. Existen muchas máquinas herramienta que usan CNC, tales como tornos, fresadoras, torno-fresadoras, soldadoras, estampadoras, troqueladoras, entre otros; sin embargo las que se tratan en este documento son el torno y la fresadora.

I.4. Investigación previa

El CHROM-1 fue el primer desarrollo de un control numérico por computadora mexicano (Herrera Ruiz y Molina, 1989). Incorporaba diversas características comunes presentes en un sistema de control numérico moderno, tales como interpolación lineal en 3 ejes, interpolación circular en los planos XY, YZ y XZ, la capacidad de interpretar el lenguaje descrito en el estándar (alemán) DIN-66025 y diversos ciclos enlatados. Su interfaz gráfica con el usuario utilizaba un editor de texto especializado con una interfaz de usuario basada en caracteres.

Posteriormente el Dr. Herrera comenzó a trabajar en la Universidad Autónoma de Queretaro y asesoró en el desarrollo de los proyectos CHROMA-1 y CHROMA-2, que eran una continuación del proyecto CHROM-1, en los cuales el Dr. Pedro Daniel Alaniz Lumbreras realizó mejoras paulatinas a diversos subsistemas del mismo. Durante el desarrollo del CHROMA-1 se extendieron las capacidades del sistema para operar con motores de corriente directa con retroalimentación y se ampliaron las capacidades del intérprete para operar un intercambiador automático de herramienta.

Se realizaron asimismo en Agosto de 2000 investigaciones para implementar un sistema de detección de fallas con la tesis de maestría “Detección del estado de ruptura de las herramientas de corte en procesos de fresado utilizando procesamiento señal” de Martín Alfredo Cornejo Aguilar que colaboraría con el sistema CHROMA-1 y posteriores. El el trabajo mencionado se implementó exitosamente un sistema monolítico que permitía controlar una fresadora al mismo tiempo que, con ayuda de una red neuronal, detectaba posibles rupturas

y astillamientos de la herramienta.

Al mismo tiempo que se desarrollaba el proyecto CHROMA-1 en el Laboratorio de Mecatrónica, se implementaba en el Laboratorio de Biotrónica un Sistema de Control Climático Inteligente (SCCI) para Invernaderos, bajo la iniciativa del M. en C. Juan José García Escalante y el Dr. Rodrigo Castañeda Miranda. En su primera versión de Enero de 2000 dicho sistema permitía administrar, por medio de riego programado, hasta 32 riegos por dos secciones. En Julio de ese mismo año el sistema fue extendido para operar el sistema de calefacción permitiendo 32 programaciones para el mismo. Un año después, en Julio de 2001 se integró el control de ventanas, permitiendo 8 programaciones para las mismas. A finales de este año se iniciaron las primeras pruebas de la primera interfaz gráfica de usuario experimental para sistemas TUNA, empleándose para la misma 256 colores y una resolución de 800×600 pixeles, la cual fue integrada en la versión del SCCI de Enero de 2002. El sistema gráfico continuó siendo mejorado para las versiones TUNA SCCI 5.0 de Enero de 2003 y TUNA SCCI 5.2 de Diciembre de 2003. Estas últimas versiones del sistema gráfico se caracterizaron por tener un grado de independencia relativo del resto del código de las aplicaciones mencionadas. Dicho sistema gráfico ofrecía, en consecuencia, la posibilidad de ser portado a otros sistemas TUNA.

Fue en Enero de 2004 que el Dr. Pedro Daniel Alaniz Lumbreras realizó los estudios y la planeación necesaria para proveer al sistema CHROMA-2 de una interfaz gráfica de usuario parecida a la que ya formaba parte del sistema TUNA SCCI 5.2. La interfaz gráfica de usuario fue integrada en el sistema CHROMA-2 siguiendo un enfoque parecido al que fue empleado para implementarla en el sistema TUNA SCCI 5.2, con ligeras modificaciones apropiadas para sistemas de control numérico por computadora (CNC) para máquinas herramienta. Dicha implementación, sin embargo, integra de manera no modular la interfaz gráfica de usuario con el resto de la aplicación.

En Abril de 2004 José Agustín Bravo Curiel sustentó la tesis de maestría “Desarrollo de un control modular para maquinaria aplicada una máquina de inyección de plástico” donde se implementó un controlador del tipo PLC para administrar las funciones de la inyectora de plástico junto con una interfaz gráfica de usuario; ambos módulos se comunicaban por medio de un conjunto bien definido de funciones y compartían el tiempo del microprocesador (la interfaz gráfica de usuario gozaba de la menor prioridad).

Asimismo, el Dr. Herrera dirigió la tesis de maestría “Desarrollo de la Ingeniería de Software para la implementación de un CNC” desarrollada por Aurora Femat Díaz en Noviembre de 2004 (Femat Díaz, 2004); en este trabajo se desarrolló un editor especializado junto con un compilador, pero el sistema resultante estaba diseñado específicamente para la plataforma Windows, lo cual dificultaba su implementación para otros sistemas operativos. Además, su diseño no era modular, lo cual dificultaba aún más su integración dentro de otros sistemas de control.

Este conjunto de innovadores desarrollos parciales, hasta cierto grado incompatibles entre sí, motivaron al desarrollo de un sistema modular de mayor alcance que sentaron las bases para:

1. continuar investigaciones en cada uno de estos campos de manera compatible entre sí,
2. evitar la duplicación de esfuerzos (al evitar trabajos repetidos en cada una de estas áreas) y

3. permitir la implementación de sistemas funcionales tanto experimentales como comerciales a partir de la misma base de código al elegir los módulos que sean pertinentes para cada proyecto (reduciendo los tiempos de desarrollo).

I.5. Hipótesis y objetivos

La hipótesis principal del presente trabajo es que:

> Es posible desarrollar un sistema de arquitectura abierta, modular y portátil para control numérico por computadora que reduzca los costos de investigación al mismo tiempo que reduce los tiempos de implementación para sistemas comerciales derivados de la misma.

Para demostrar esta hipótesis el presente trabajo tiene los siguientes objetivos:

- proponer una arquitectura abierta modular para sistemas de CNC,
- implementar en lenguaje ISO C++ (para portabilidad entre diferentes plataformas) un prototipo funcional para fresadora[2] que a su vez sea
- un modelo de referencia a seguir (en algún sistema operativo de amplia aplicación).

Para ofrecer un punto de partida apropiado para investigaciones posteriores la arquitectura deberá estar bien documentada, motivo por el cual se empleará el lenguaje de diseño gráfico de software UML (lenguaje de modelado unificado, por sus siglas en inglés) ya que ofrece un conjunto de diagramas estándar de amplia aceptación. Asimismo, para la implementación del sistema y sus módulos correspondientes se decidió usar tecnología de orientación objetos por las ventajas que ofrece al facilitar la modularización de los sistemas por medio de componentes de software con responsabilidades bien definidas e interacciones bien establecidas (por medio de la agregación, la composición, la herencia, etc.).

Una ventaja importante de la tecnología orientada a objetos es que nos permite emplear la misma base de código para probar módulos experimentales y módulos estables, en virtud de que ambos tipos de módulos pueden emplear la misma interfaz y echar mano del polimorfismo para ser intercambiables; de esta manera es posible ofrecer módulos funcionales una vez que la investigación ha entregado un producto estable.

Otra ventaja de establecer interfaces bien definidas es que es posible separar los detalles de implementación particulares para algún sistema operativo de los algoritmos de control y administración de la máquina herramienta, lo cual amplía grandemente la portabilidad del código del sistema de una plataforma a otra, en virtud de que basta con reimplementar los componentes del sistema que dependan fuertemente del sistema operativo (tales como la interfaz gráfica de usuario) sin necesidad de volver a trabajar en los algoritmos propios del sistema de control por computadora.

[2] Se elige un prototipo de fresadora y no de torno ya que si es posible mostrar un prototipo de 3 ejes, se asume que será posible desarrollar también uno de 2 ejes.

II. ESTADO DEL ARTE

II.1. Arquitectura de las máquinas herramienta

Hay muchos tipos diferentes de máquinas herramienta en la industria. Los dos tipos principales son los centros de maquinado (fresadoras) y los tornos.

II.1.1 Centro de maquinado

En la categoría de centros de maquinado se puede incluir desde simples fresadoras manuales hasta centros de maquinado de 5 grados de libertad. Los hay de diferentes tamaños, características, propósitos especiales, etc., pero todos tienen algo en común: sus principales ejes son el X y el Y (razón por la cual a veces se les llama máquinas XY). En esta misma categoría están algunas máquinas herramienta que no son fresadoras, principalmente ruteadores y perfiladores.

El estudio de las fresadoras es suficientemente representativo de la mayoría de los problemas que se presentan en los centros de maquinado, razón por la cual en el presente trabajo nos enfocaremos a la automatización de una fresadora controlada numéricamente por computadora. Para los efectos de esta tesis, definiremos a una fresadora de la siguiente manera:

> Una fresadora es una máquina capaz de realizar un movimiento de corte simultáneo en dos o más ejes al mismo tiempo y que usa una fresa como herramienta principal de corte (Smid, 2003).

Las máquinas fresadoras de CNC tienen una herramienta de corte rotatoria, sostenida por una abrazadera de metal llamada portaherramientas (mandril, broquero, etc.) que a su vez va sujeta a un eje giratorio conocido como husillo, y una mesa donde se coloca el

Figura 2.1: Ejemplo de una fresadora.

Figura 2.2: Ejemplo de un torno.

Figura 2.3: Panel de control de una máquina herramienta.

material. La alimentación de materia prima al cortador se hace ya sea empujando el cortador al material o viceversa, en trayectorias curvas o rectas tridimensionales y la pieza final es creada mediante la remoción de todo el material innecesario de la pieza de trabajo.

II.1.2 Torno

Un torno es una máquina herramienta que realiza movimientos de corte rotando la pieza a maquinar. Desde el punto de vista de la programación no hay mucha diferencia entre los diferentes tipos de tornos. En los tornos de CNC se rota la pieza en contra de un punto fijo de una herramienta para producir el corte en el material y la alimentación de material se hace de manera lineal contra la pieza de corte.

II.2. Interfaz hombre-máquina

Las máquinas herramienta de CNC emplean dos teclados especiales, a saber:

Panel de control Para introducir instrucciones alfanuméricas, tales como programas o instrucciones manuales directas (IMDs). Como se aprecia en la figura 2.3, un panel de control es parecido al teclado de una computadora personal.

Panel de operación Para operar la máquina en modo manual o para controlar la ejecución de un programa. Como se puede observar en la figura 2.4, un panel de operación tiene

Figura 2.4: Panel de operación de una máquina herramienta.

una interfaz más orientada al operador de la máquina que al programador de la misma al ofrecer maneras de operar los ejes de la máquina y de ajustar las velocidades y otras propiedades del movimiento de la máquina, ya sea bajo el control de un humano (modo manual) o de la computadora del sistema de control numérico por computadora (modo automático).

Entre las funciones más comunes de un panel de operación se encuentran un paro de emergencia general, un botón para iniciar la ejecución de un programa y otro para pausarla, así como teclas etiquetadas con el nombre de un eje seguido de una dirección (e.g., X+ o X-) destinadas a dar instrucciones relativas a los ejes durante la operación manual. Asimismo, es común encontrar selectores de velocidades y un generador de pulsos.

II.3. Funciones que realiza un sistema de CNC

En ambos tipos de máquina herramienta de CNC, fresadora y torno, la elaboración de piezas se hace mediante la eliminación selectiva de material, ya sea rotando la herramienta de corte o la pieza, respectivamente. La remoción del material hecha por la máquina debe de ser determinada por el usuario; para ello se deben de especificar características de control especiales como son la velocidad de rotación del husillo y la velocidad de avance de la herramienta de corte.

La *velocidad del husillo* es lo que determina qué tan rápido debe girar la herramienta de corte, en la fresadora, o la pieza, en el torno, para hacer una desbastación de material adecuada; esto depende del tipo de material con el que se esté trabajando y la herramienta de corte. Existen otros parámetros relacionados con la velocidad del husillo tales como el sentido de rotación y las unidades de medida. Estos parámetros son controlados por otras funciones conocidas como funciones misceláneas que se mencionan más adelante.

La *velocidad de avance* de la herramienta de corte indica qué tan rápido es cortado el material, es decir, con qué velocidad avanza la herramienta de corte por la superficie a trabajar, en el caso de la fresadora, o la pieza, en el caso del torno. La velocidad de corte puede ser medida de dos maneras: unidad de medida por minuto o por revoluciones en donde las unidades de medida pueden ser en milímetros o pulgadas dependiendo del sistema métrico que se utilice.

La velocidad de rotación del husillo y la velocidad de avance de la herramienta de

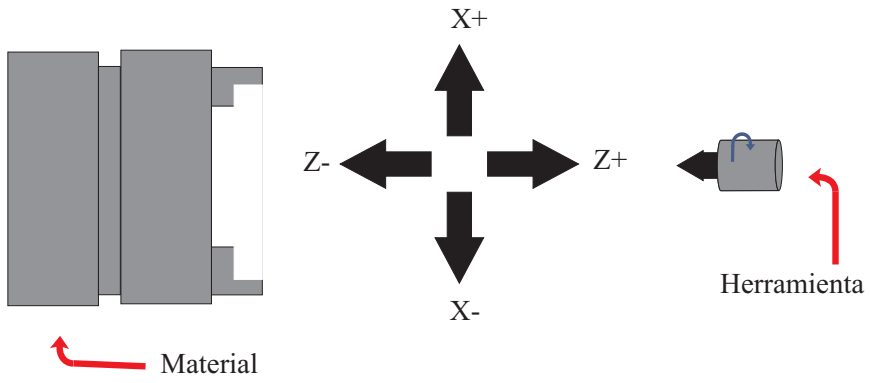


Figura 2.5: Ejes típicos de un torno.

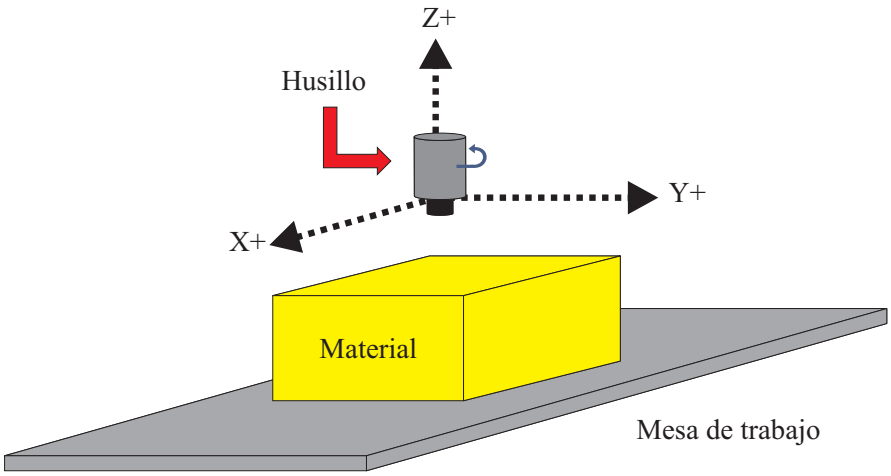


Figura 2.6: Ejes típicos de una fresadora.

corte son parámetros que nos permiten cortar la superficie del material a trabajar para obtener la pieza deseada, es decir, estas velocidades determinan cómo ha de moverse la herramienta. Para describir instrucciones de maquinado completas falta saber cuál es la *trayectoria* que ha de seguir la herramienta; por eso otro factor importante en las máquinas herramienta es la existencia de un sistema coordenado. Los sistemas coordenados que se usan en las maquinas herramienta son el sistema coordenado rectangular y el sistema coordenado polar. Cuando definimos un sistema coordenado usamos ejes de referencia intersectados en un origen y tomamos este punto de intersección como el punto de referencia o punto cero; sin embargo existen otros ejes adicionales que se usan para otras tareas especificas, como rotaciones, donde se definen otros puntos de referencia. La manera en se toman los ejes en las máquinas herramienta depende de la constitución (o arquitectura) cada una de las máquinas de CNC que se trabaje. Las máquinas herramienta usualmente tienen más de un eje coordenado de referencia. En el caso de la fresadora se tienen tres ejes, dos ejes horizontales, llamados eje X y eje Y, y un eje vertical, llamado eje Z y cuando hay un cuarto eje, este generalmente es de rotación. En el torno el número de ejes se reduce en uno, pues hay un eje que siempre esta fijo. Las figuras 2.6 y 2.5 muestran los ejes que generalmente se encuentran en este tipo de máquinas herramienta.

Para este trabajo se consideran una fresadora vertical de tres ejes (el eje X es paralelo a la parte más larga de la mesa, el eje Y a la parte más angosta y el eje Z corresponde al desplazamiento vertical del husillo) y un torno horizontal de dos ejes; sin embargo, los resultados pueden ser extendidos a otras máquinas herramienta del mismo tipo sin dificultad.

II.4. Modos de operación de un sistema de CNC

A grandes razgos, podemos decir que un sistema de CNC tiene dos modos de operación, a saber, el modo de operación manual y el modo de operación automático. El *modo de operación manual* se emplea para preparar una máquina herramienta antes de su operación automática y para maquinar piezas de forma manual. El *modo de operación automático* se emplea cuando se tiene un programa cargado en la memoria de la unidad de control del sistema de CNC y se desea ejecutarlo para producir alguna pieza.

El "modo de operación manual" en realidad está formado por los siguientes modos manuales de operación:

Inicio Sirve para ayudar a los ejes de la máquina a localizar su correspondiente punto de referencia (también conocido como cero), establecido usualmente con ayuda de uno o más interruptores de límite.

Rápido Se emplea para desplazar los ejes de la máquina a una velocidad de desplazamiento rápido (es decir, la velocidad empleada cuando se desea desplazar la herramienta sin realizar ningún corte). Esta velocidad de desplazamiento rápido usualmente se establece como una configuración del controlador.

Impuso También desplaza los ejes de la máquina pero a una velocidad acotada por la velocidad de impulso (en milímetros por minuto) establecida en el panel de operación.

Manivela Una manivela, en el caso de un sistema de CNC, es un generador de pulsos que permite seleccionar un eje a la vez junto con un multiplicador $\times 1$, $\times 10$ ó $\times 100$ del mínimo desplazamiento posible por parte del eje dado.

IMD Es posible ejecutar instrucciones que permitan cambiar el estado del controlador numérico para modificar parámetros tales como el sistema métrico a utilizar, o para realizar un cambio de herramienta por medio de instrucciones parecidas a las que se emplean para indicar conjuntos de pasos a realizar por la máquina en modo automático; este tipo de instrucciones se conocen como Instrucciones Manuales Directas.

Asimismo, el "modo de operación automático" en realidad requiere de los siguientes modos automáticos de operación:

Memoria Cuando el ejecutivo de programas está activo, lee un archivo de código G y va ejecutando las instrucciones contenidas en el mismo.

Editor Permite la modificación y creación de programas en código G para su posterior ejecución.

Administrador de archivos Usualmente un sistema de CNC puede trabajar con más de un programa, así que usualmente es necesario elegir un *programa activo* (el programa a editar o ejecutar) entre una colección de programas disponibles, así como copiar, borrar, transferir y realizar otras actividades de administración de archivos.

II.5. Modos manuales de operación

Los modos manuales de operación, salvo por el modo inicio, no son mutuamente excluyentes, así que no es inusual emplear más de uno de ellos al mismo tiempo, y es común que compartan la misma interfaz gráfica de usuario.

II.5.1 Modo inicio

Al encender la computadora encargada de las tareas de control de un sistema de CNC usualmente los registros que contienen la posición en que se encuentra cada uno de los ejes se inicializan a cero, a pesar de que nada garantiza que ésta será efectivamente la posición en que se encontrarán los ejes de la máquina herramienta. Por esta razón, para garantizar la operación correcta de la máquina, es necesario encontrar la verdadera posición de inicio de la máquina antes de intentar ejecutar cualquier clase de instrucción que dependa del posicionamiento correcto de los ejes de la máquina. El modo inicio fue creado con este propósito en mente, para permitir al usuario localizar la verdadera posición de inicio del sistema; usualmente el modo inicio es el modo en el que se encuentra el sistema de control al iniciar. La localización de la posición de inicio se realiza en la dirección indicada en el panel de operación por medio de la tecla de dirección de eje correspondiente (e.g., X+ inicia la búsqueda de la posición de inicio desplazando el eje X en la dirección positiva).

II.5.2 Modo rápido

A veces es necesario realizar un desplazamiento rápido de los ejes de la máquina para efectos de maquinado manual tales como el desplazamiento de la herramienta de corte para trabajar en otra parte de la pieza o para hacer un cambio herramienta. La velocidad de desplazamiento rápido es un valor configurado en la memoria del controlador, cuyo valor se puede variar ligeramente con opciones de porcentaje en el panel de operación (usualmente 25 %, 50 % y 100 %). El movimiento se realiza en la dirección elegida durante el tiempo que esté presionada la tecla de dirección de eje correspondiente (e.g., mientras se presione X+ el movimiento será en la dirección positiva del eje X).

II.5.3 Modo impulso

El modo impulso se emplea para desplazar los ejes de la máquina con un control de velocidad preciso, siendo éstas velocidades usualmente bajas, con el propósito de desbastar material durante la operación manual de la máquina. La velocidad de impulso se establece directamente por medio del panel de operación (aunque sería posible tener un enfoque parecido al de la velocidad de desplazamiento rápido almacenando un valor de referencia en la memoria del controlador y empleando una selección de porcentaje en el panel de operación). El movimiento se realiza en la dirección elegida durante el tiempo que esté presionada la tecla de dirección de eje correspondiente (e.g., mientras se presione X+ el movimiento será en la dirección positiva del eje X).

II.5.4 Modo manivela

Cuando resulta más relevante tener un control preciso de la posición de la herramienta durante un maquinado manual que tener un control preciso de la velocidad se emplea el modo manivela. El control de la posición se logra por medio de un generador de pulsos, donde cada pulso indica un incremento o decremento de la oposición del husillo.

II.5.5 Modo Introducción Manual de Datos (IMD)

El modo Introducción Manual de Datos (IMD) se emplea para introducir órdenes de configuración y de desplazamientos elaborados en los que se requiera controlar tanto la velocidad como el posicionamiento de manera precisa.

II.6. Modos automáticos de operación

La ejecución de un programa para un sistema de CNC (usualmente llamado programa de parte por contener las instrucciones de la parte a fabricar) se realiza en modo memoria, mientras que los modos editor y administrador de archivos en realidad sólo se usan para administrar los programas dentro del sistema.

II.6.1 Modo memoria

Este modo se emplea para ejecutar programas de control numérico por computadora que ya han sido creados previamente por medio del (modo) editor. Estos programas (véase la sección 2.7) establecen las velocidades, posiciones, trayectorias y demás factores que se emplean para describir un conjunto de actividades para maquinar una parte sin la intervención directa del ser humano. Durante la ejecución de estos programas es posible modificar la velocidad de avance y otras opciones del mismo programa por medio del panel de operación.

II.6.2 Modo editor

Se emplea para editar y crear nuevos programas de parte que habrán de ser ejecutados de manera automática en modo memoria. Los editores integrados dentro de un sistema de CNC usualmente ofrecen características especiales, desde la simple numeración automática de cada línea (bloque) hasta la simulación del programa de parte.

II.6.3 Modo administrador de archivos

En el modo de administración de archivos es posible copiar programas desde un dispositivo externo, borrar programas para liberar memoria, copiar un programa existente en un nuevo archivo para crear (después de las modificaciones pertinentes) un programa parecido al original, etc., así como establecer un enlace remoto para Control Numérico Directo (CND); el CND consiste en leer las instrucciones de un programa desde un dispositivo externo por medio de un puerto de comunicaciones.

II.7. Estructura de los programas de partes

Un programa CNC es un conjunto de instrucciones secuenciales relacionadas al maquinado de una parte o pieza. Estas instrucciones están en un formato que el CNC puede aceptar, interpretar y procesar; además, las instrucciones deben ajustarse a las especificaciones de la máquina herramienta que se esté utilizando (por ejemplo, es imposible especificar un desplazamiento mayor al que la máquina en particular puede realizar físicamente hablando).

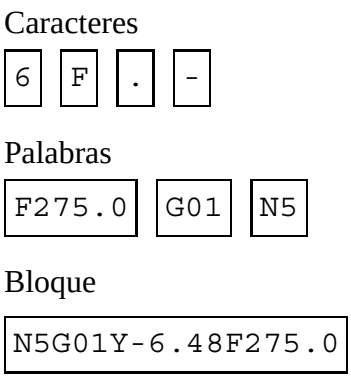


Figura 2.7: Ejemplos del formato de direccionamiento por palabra.

La estructura de un programa de parte (o programa simplemente) varía ligeramente dependiendo de la marca y modelo del controlador (e inclusive puede variar según la máquina a la cual está controlando), pero la idea general siempre es la misma. Un programa de parte usualmente empieza con un número de programa o identificación similar seguida por el bloque de instrucciones en orden lógico. El programa termina con un *código de transmisión* o un símbolo de terminación de programa tal como el signo de porcentaje (‘ %’). Alguna documentación interna o mensajes dirigidos al programador pueden estar entre los bloques dentro del programa en forma de comentarios.

Aunque a través de los años han surgido varios formatos y con el tiempo han ido cambiando, la misma idea se mantiene. A lo largo de la historia del control numérico han existido tres formatos de programación que fueron significativos en su tiempo, los cuales son listados a continuación según su orden de aparición:

1. Formato secuencial con tabuladores.

2. Formato fijo.

3. Formato de direcciones de palabra.

En la actualidad solo los controladores más viejos usan el formato secuencial o el fijo, pues éstos desaparecieron en los años setentas y fueron reemplazados por el más conveniente formato de dirección de palabra.

El formato de direcciones de palabra está basado en la combinación de una letra (la dirección) y un número entero o de punto flotante, como se muestra en la figura 2.7. El lenguaje de programación basado en este formato de direcciones de palabra se denomina código G de manera informal (véase la sección 3.1 y el capítulo 10). El número que sigue a la dirección puede llevar un signo positivo o negativo (o ninguno, en cuyo caso se asume positivo), según los valores que puedan ser almacenados en la dirección particular. Típicamente, la palabra direccionada se refiere a un registro especifico de la memoria del controlador, razón por la cual se dice que cada letra es una dirección (en memoria del controlador). Algunas palabras típicas son:

```
G01 M30 D25 X5.75 N105 H01 Y0 S2500
 Z-5.14 F12.0 T0505 T05 M01 B180.0
```

La dirección (letra) en el bloque define el significado de la palabra. Los espacios dentro de una palabra no son permitidos, sin embargo, se permiten entre palabras, i.e., antes de la letra; por ejemplo, `N123 G01` es válido pero `N1 23 G01` no. El dato indica la asignación numérica a la palabra y los valores válidos varían dependiendo de la dirección de la palabra; por ejemplo, la dirección de un número de línea N no puede estar seguido por un número con punto flotante como 2.4 pues ello carece de sentido.

Cualquier palabra es una serie de caracteres que definen una simple instrucción a la unidad de control y a la máquina. Los ejemplos anteriores de palabras típicas tiene el siguiente significado en un programa en código G:

`G01` Comando preparatorio (tipo de movimiento).

`M30` Función miscelánea (fin de programa).

`D25` Selección del número de compensación.

`X5.75` Coordenada en el eje X (valor positivo).

`N105` Número de bloque.

`H01` Número de compensación de la longitud de la herramienta.

`Y0` Coordenada en el eje Y (valor cero).

`S2500` Función de la velocidad del husillo.

`Z-5.14` Coordenada en el eje Z (valor negativo).

`F12.0` Función de velocidad de avance.

`T0505` Selección de la herramienta (para tornos).

`T05` Selección de la herramienta (para fresadoras).

`/M01` Función miscelánea con símbolo de salto de bloque.

Las palabras individuales son agrupadas en un conjunto llamado *bloque* para formar una secuencia de código de programa; las palabras que conforman un bloque no se procesan en el orden en que aparecen en el bloque sino siguiendo una lógica apropiada para su ejecución en la máquina, ejecutando algunas palabras simultáneamente en algunos casos y otras en un orden que depende de las palabras presentes en el bloque. Por ejemplo, se puede formar un bloque con las palabras correspondientes a un número de secuencia N, un comando preparatorio G, una función miscelánea M, una compensación D o H, una coordenada X, Y o Z, la función de velocidad F, una función del husillo S, una función de la herramienta T, etc. Otro ejemplo es el siguiente bloque que muestra un movimiento rápido de herramienta a la posición absoluta `X13.0 Y4.6` con el refrigerante encendido:

```
N25 G90 G00 X13.0 Y4.6 M08
```

donde

N25 es el número de bloque o secuencia,

G90 indica modo absoluto,

G00 indica modo de movimiento rápido,

X13.0 Y4.6 especifica las coordenadas de la locación y

M08 activa la función de refrigerante.

El controlador procesará cualquier bloque como una unidad completa, *nunca parcialmente*. La mayoría de los controladores permiten un orden aleatorio de la palabra en el bloque con tal de que se especifique primero el número de bloque N.

En años recientes se ha hecho la propuesta de desplazar el código G por un tipo de programación de más alto nivel conocido como STEP-NC pero, en virtud de que dichos desarrollos todavía son experimentales, en el presente trabajo orientado a desarrollar un sistema que reproduzca el estado del arte *actual* no considera incluir dicho desarrollo.

III. CASOS DE USO

III.1. Operación típica de un sistema de CNC

Aunque han habido esfuerzos por estandarizar tanto la arquitectura como la operación de los sistemas de CNC y existen diversos estándares tanto para la arquitectura como para la operación de los mismos, desafortunadamente, los estándares *de facto* difieren de los estándares oficiales. Tal es el caso, por ejemplo, del estándar de programación conocido como código G, el cual se define en el estándar RS-274D de la Electronic Industries Alliance (EIA) de Estados Unidos y en el estándar DIN 66025 europeo (alemán), además del estándar internacional ISO 6983-1:1982; a pesar de estos esfuerzos estandarización, cada fabricante introduce características particulares en su sistema de control que se apartan de cualquiera de estos estándares. Por esta razón, muchos fabricantes de sistemas de CNC han hecho el intento por estandarizarse con los sistemas GE Fanuc, aunque la misma evolución propia de los sistemas GE Fanuc ha dificultado las cosas para otros fabricantes. Es por esta razón que un análisis de requerimientos de un sistema de CNC es en buena parte, en virtud de la falta de un estándar *de facto*, un *diseño de requerimientos* hasta cierto grado (en cualquier caso hay que apegarse al estándar *de facto* lo más posible).

El objetivo del presente capítulo es *organizar* la información presentada en el capítulo 2 (y agregar algo más de detalle a la misma) en una forma que represente de manera eficiente la forma en que un sistema de CNC es usualmente empleado, pero sin incluir particularidades específicas de alguna tecnología en especial. En ese sentido, este capítulo estudia el *qué* se desea hacer con un sistema de CNC (en algunos casos también será menester hacer referencia al *para qué* se desea hacerlo) sin preocuparse (mucho) por el *cómo* se hace.

III.2. Metodología casos de uso

La metodología casos de uso, o simplemente los *casos de uso*, fue una propuesta de Ivar Jacobson como una forma de recabar de manera sistemática toda la información obtenida durante el análisis previo al diseño de un sistema. Este análisis está orientado a identificar las necesidades del usuario; aunque pueda percibirse como algo evidente, la identificación de necesidades del usuario se basa en responder dos preguntas clave:

1. ¿Qué quiere hacer el usuario?

2. ¿Quién es el usuario?

El *qué* se modela como un caso de uso, y en *quién* como un actor.

En diagrama típico de un caso de uso (véase la figura 3.3) representa a los actores por medio de monitos de palitos y a los casos de uso por medio de óvalos que contienen un pequeño texto que describe la acción que se desea realizar. Las figuras, sin embargo, en cualquier caso son bastante fáciles de entender aún sin una explicación detallada. Se dice que un

Figura 3.1: El caso de uso A incluye al caso de uso B.

Figura 3.2: El caso de uso B extiende al caso de uso A.

caso de uso A *incluye* a otro caso de uso B si el conjunto de actividades de A *siempre* incluye a las de B (figura 3.1). Por otra parte se dice que un caso de uso B *extiende* a otro caso de uso A si el conjunto de actividades de A *podría* incluir a las de B bajo ciertas circunstancias (figura 3.2).

III.3. Caso de uso general del sistema de CNC

A partir de la descripción general de los sistemas CNC dada en el capítulo 2, podemos establecer el caso de uso general del sistema de CNC como se ilustra en la figura 3.3.

Figura 3.3: Caso de uso general para un sistema de CNC.

Descripción Pasos necesarios para maquinar una pieza con un sistema de CNC.

Actores Computadora remota, programador de CNC y operador de CNC, los cuales son casos particulares de un usuario de CNC.

Suposiciones Los usuarios de CNC han recibido el entrenamiento apropiado para usar el sistema de CNC.

Pasos Un programador necesita obtener y estudiar los dibujos de un diseño para poder discutir los métodos maquinado más apropiados con un operador de la máquina herramienta (quien conoce las particularidades de la misma); esta discusión que ocurre entre el programador y el operador de la máquina tiene como objetivo recabar la información necesaria para permitir al programador escribir un programa de parte. Entre los temas a tratar se encuentran:

- La manera más apropiada de sujetar la pieza a la máquina herramienta, en especial en el caso de las fresadoras que requieren, en buena cantidad de casos, sujeción especial.
- La elección de las herramientas de corte más apropiadas.
- Las velocidades para el husillo y para los ejes.

A continuación se espera que el programador y el operador de la máquina herramienta cooperen para hacer ejecuciones de prueba del programa recién creado, verificando efectivamente que la máquina herramienta sea apropiada para el trabajo y el sistema de control de la misma sea apropiada para el código G recién desarrollado (como ya fue mencionado en la sección 3.1, no existe realmente una buena compatibilidad entre distintos sistemas basados en código G); una vez superada esta etapa de prueba, el operador de la máquina podrá repetir el ciclo de maquinado sin intervención del programador.

En algunos casos no se requiere de la elaboración de un programa de parte antes del maquinado de una parte si el trabajo de la parte es suficientemente sencillo y el tamaño del lote es significativamente pequeña. En estos casos el operador puede proveer a la máquina herramienta de instrucciones paso a paso para realizar el trabajo requerido y se dice que se opera la máquina herramienta *manualmente*.

Los cuatro casos de uso contenidos dentro del sistema en el diagrama de la figura 3.3 dependen, al menos en su mayor parte, del sistema de control de la máquina herramienta.

No funcional Debe haber una buena comunicación entre el programador y el operador, y en muchos casos ambas actividades recaerán en un mismo usuario. El proceso de maquinado debe ser realizado de la manera más precisa posible. Se deben abaratar costos sin sacrificar la calidad de la parte.

III.4. Casos de uso para configurar la máquina herramienta

Caso de uso 1 Preparación de la máquina herramienta (figura 3.4).

Descripción La máquina herramienta es preparada para procesar correctamente programas ejecutados en forma automática o para procesar instrucciones manuales.

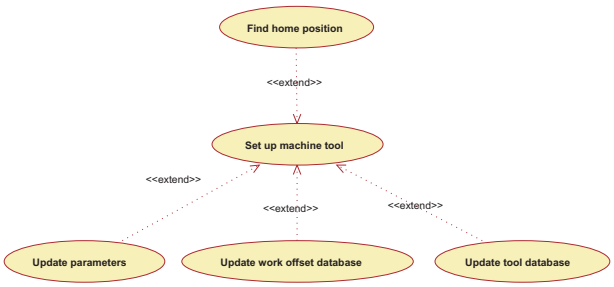


Figura 3.4: Casos de uso para la configuración de la máquina herramienta.

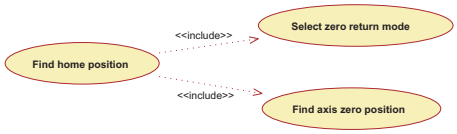


Figura 3.5: Casos de uso para retorno a cero.

Actores Operador de CNC.

Suposiciones Ninguna.

Pasos Primero se localiza el cero de la máquina y después se actualizan las bases de datos.

No funcional Siempre se debe localizar el cero de la máquina antes de ejecutar cualquier programa y, de preferencia, también antes de usar la máquina en modo manual.

III.4.1 Localización del cero de la máquina

Caso de uso 1.1 Retorno a cero (figura 3.5).

Descripción Este es el modo de operación en el que inicia el sistema. El usuario va eligiendo un eje a la vez y el controlador de movimiento va localizando la posición de referencia o inicio del mismo.

Actores Operador de CNC.

Suposiciones El sistema de control acaba de iniciar, o se desea localizar nuevamente el cero de la máquina.

Pasos Se elige el modo de regreso a cero y luego se va eligiendo un eje a la vez hasta que se ha encontrado el cero en cada uno de los ejes.

No funcional La edición de desplazamientos de trabajo a partir de la posición actual de la herramienta debe ser deshabilitada mientras no se haya encontrado en cero de la máquina.

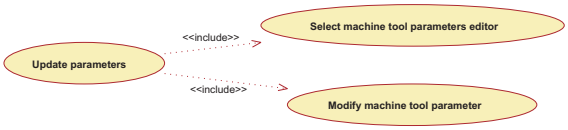


Figura 3.6: Casos de uso para actualizar parámetros de la máquina herramienta.

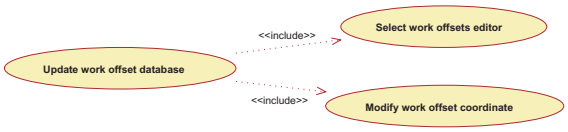


Figura 3.7: Casos de uso para actualizar base de datos de desplazamientos de trabajo.

III.4.2 Actualización de parámetros de la máquina

Caso de uso 1.2 Actualizar parámetros de la máquina (figura 3.6).

Descripción Además de las configuraciones usuales, los sistemas de control numérico deben configurar parámetros relativos a opciones por omisión (como el tipo de movimiento por omisión—usualmente G01—, por ejemplo) o la configuración del controlador de movimiento (tal como el valor de las constantes de un PID).

Actores Operador de CNC.

Suposiciones Se tiene la autorización apropiada para modificar los parámetros de la máquina; estas configuraciones podrían estar protegidas por contraseña.

Pasos Se elige la opción de modificar los parámetros de la máquina y se editan los campos que se desee modificar. Se puede guardar la configuración o descartar los cambios al abandonar esta opción.

No funcional Toda modificación debe ser documentada.

III.4.3 Actualización de bases de datos

Caso de uso 1.3 Actualizar base de datos de desplazamientos de trabajo (figura 3.7).

Descripción Un desplazamiento de trabajo es lo que usualmente se conoce como cero de parte, es decir, el punto de referencia empleado para un programa de parte. Puede haber un número dado de desplazamientos de trabajo almacenados en la base de datos del controlador numérico.

Actores Operador de CNC.

Suposiciones Se ha localizado previamente el cero de la máquina si se desea copiar el valor correspondiente de la posición actual de la máquina.

Pasos Después de elegir la base de datos de desplazamientos de trabajo, una coordenada de un desplazamiento de trabajo se puede editar de dos maneras distintas, a saber, editando directamente su valor o copiando el valor correspondiente de la posición actual

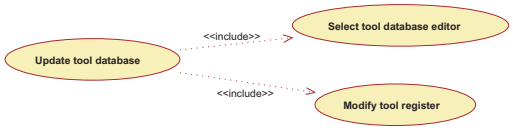


Figura 3.8: Casos de uso para actualizar base de datos de propiedades de herramientas.

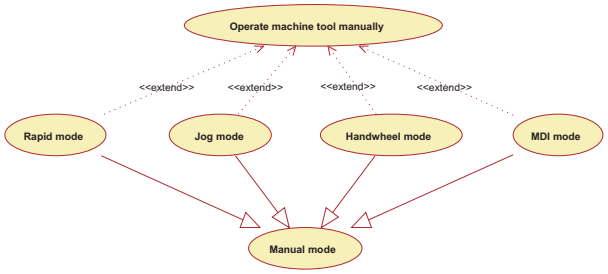


Figura 3.9: Casos de uso para la operación manual de la máquina herramienta.

de la máquina con respecto al cero de la máquina. Se pueden guardar los cambios o descartarse al abandonar el editor de la base de datos.

No funcional Se debe verificar todo desplazamiento de trabajo empleado en un programa de parte antes de ejecutarlo.

Caso de uso 1.4 Actualizar base de datos de herramientas (figura 3.8).

Descripción Las herramientas tienen un diámetro, una longitud y una forma particular que debe ser conocida de antemano para poder compensar propiamente estas medidas durante la ejecución de un programa de parte.

Actores Operador de CNC.

Suposiciones Las medidas introducidas en la base de datos son correctas.

Pasos Se elige la base de datos de herramientas, se selecciona un registro y se edita su contenido. Se pueden guardar los cambios o descartarse al abandonar el editor de la base de datos.

No funcional La información de la base de datos de herramientas siempre debe coincidir con las herramientas instaladas en la máquina, y también se debe verificar que las herramientas de la base de datos de herramientas sean apropiadas para la ejecución del programa activo.

III.5. Casos de uso para operación manual de la máquina herramienta

Caso de uso 2 Operación manual de la máquina herramienta (figura 3.9).

Descripción La operación manual del sistema involucra controlar los diversos dispositivos de la máquina herramienta por medio de instrucciones que el operador introduce en el sistema de control y que deben ser acatadas de manera inmediata, sin un medio de almacenamiento de las mismas de por medio.

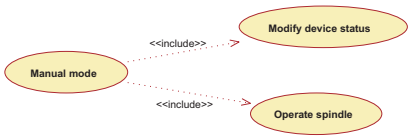


Figura 3.10: Casos de uso para la operación manual común.

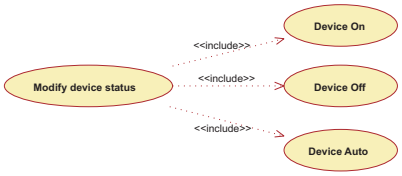


Figura 3.11: Casos de uso para modificar estado de dispositivo.

Actores Operador de CNC.

Suposiciones Ninguna.

Pasos Se selecciona un modo manual de operación y se introducen instrucciones directamente a través de la interfaz de usuario de la máquina herramienta.

No funcional La operación manual de la máquina se emplea en actividades simples que no requieren previamente la creación de un programa de parte.

III.5.1 Operaciones manuales comunes

Caso de uso 2.1 Modo de operación manual común (figura 3.10).

Descripción Los modos de operación manual usuales comparten mucha de su funcionalidad básica pues todos permiten operar el husillo y los dispositivos de la máquina al mismo tiempo que los ejes.

Actores Operador de CNC.

Suposiciones Se ha elegido alguno de los modos de operación manual de la máquina.

Pasos Se elige un modo manual de la máquina y el usuario opera el husillo (iniciar giro en dirección normal u opuesta, detener giro, incrementar o decrementar velocidad del husillo) o alguno de los dispositivos de la máquina (encender o apagar dispositivo, o dejarlo en modo automático) tales como el refrigerante o una banda transportadora de viruta.

No funcional Se debe tener cuidado al operar la máquina en modo manual pues una acción equivocada puede dañar la herramienta, a la pieza de trabajo, a la misma máquina o lesionar al usuario. Se deben tener en cuenta todas las medidas de seguridad mientras se opera la máquina.

Caso de uso 2.1.1 Modificar estado de dispositivo (figura 3.11).

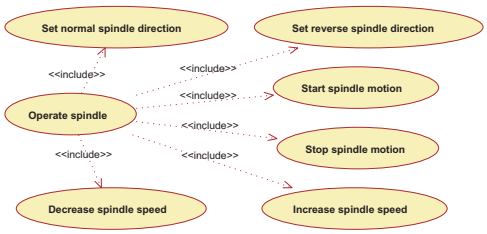


Figura 3.12: Casos de uso para operar husillo.

Descripción La máquina-herramienta puede contar con dispositivos tan diversos tales como luces para iluminar el área de trabajo, refrigerante para enfriar la herramienta durante el proceso de maquinado y una banda transportadora de viruta.

Actores Operador de CNC.

Suposiciones Ninguna.

Pasos El dispositivo que se desee se puede encender o apagar. Si el dispositivo lo soporta, se puede dejar el dispositivo en modo automático para que sea controlado por instrucciones de un programa o instrucciones introducidas en modo IMD.

No funcional Se deben tener en cuenta todas las medidas de seguridad mientras se opera la máquina.

Caso de uso 2.1.2 Operar husillo (figura 3.12).

Descripción El husillo se puede emplear durante la operación manual cuando se desea maquinar piezas.

Actores Operador de CNC.

Suposiciones Ninguna.

Pasos Durante la operación manual el husillo se puede emplear para maquinar piezas. Se puede establecer la dirección de giro (normal o en reversa), se puede iniciar el giro o detenerse y se puede incrementar o decrementar la velocidad del husillo.

No funcional Se deben tener en cuenta todas las medidas de seguridad mientras se opera la máquina.

III.5.2 Modo rápido

Caso de uso 2.2 Modo de operación manual rápido (figura 3.13).

Descripción Este modo de operación manual se emplea para posicionar la herramienta rápidamente de una ubicación a otra por medio de un impulso de velocidad de avance rápido.

Actores Operador de CNC.

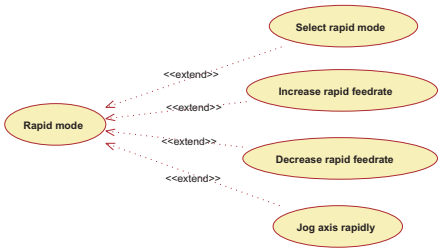


Figura 3.13: Casos de uso para el modo de operación manual rápido.

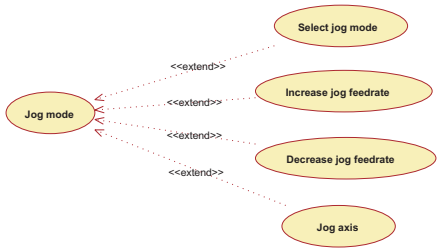


Figura 3.14: Casos de uso para el modo de operación manual impulso.

Suposiciones La herramienta no está maquinando ninguna pieza mientras se desplazan los ejes en modo rápido.

Pasos Se elige un eje y una dirección a la vez y la máquina desplaza el eje en la dirección indicada por medio de un impulso de velocidad de avance rápido. Por impulso se entiende que el movimiento dura mientras se mantenga la elección (usualmente por medio de un botón u otro dispositivo de retorno automático).

No funcional La velocidad de avance rápido debe ser lo más elevada posible dentro de los límites de seguridad para permitir al operador de CNC mover la herramienta en el espacio de trabajo en poco tiempo, así que usualmente será una velocidad de avance que no es apropiada para maquinado.

III.5.3 Modo impulso

Caso de uso 2.3 Modo de operación manual impulso (figura 3.14).

Descripción Este modo de operación manual se emplea para avanzar la herramienta de corte a una velociada de impulso apropiada para el maquinado de una pieza.

Actores Operador de CNC.

Suposiciones La herramienta estará cortando el material de la pieza la mayor parte del tiempo en que se use este modo manual de operación.

Pasos Se elige un eje y una dirección a la vez y la máquina desplaza el eje en la dirección indicada por medio de un impulso de baja velocidad que se puede regular. Por impulso se entiende que el movimiento dura mientras se mantenga la elección (usualmente por medio de un botón u otro dispositivo de retorno automático).

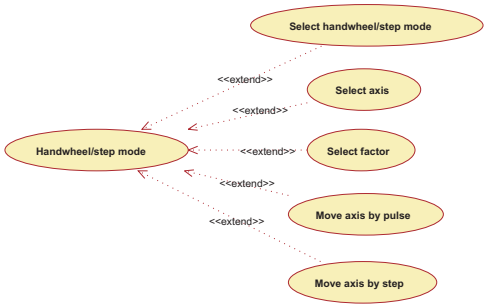


Figura 3.15: Casos de uso para el modo de operación manual manivela.

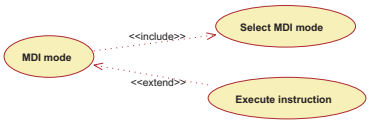


Figura 3.16: Casos de uso para el modo de operación manual introducción manual de datos (IMD).

No funcional La velocidad de impulso de maquinado debe ser baja para evitar daños a la herramienta, pero lo suficientemente elevada para evitar un decenso en la productividad.

III.5.4 Modo manivela

Caso de uso 2.4 Modo de operación manual manivela/a pasos (figura 3.15).

Descripción Esta forma de operación se emplea para tener un control preciso de la herramienta en forma manual.

Actores Operador de CNC.

Suposiciones Se desea maquinar una pieza de manera manual o se desea ubicar el cero de la pieza.

Pasos Se elige un eje a mover junto con un factor de incremento (usualmente se ofrecen como opciones $\times 1$, $\times 10$ o $\times 100$). A continuación se emplea un generador de pulsos para indicar un incremento de posición (ya sea en dirección positiva o negativa) preciso. Otra opción es generar un único incremento de las dimensiones indicadas por el multiplicador por medio de un dispositivo de retorno automático (usualmente un boton etiquedado con el eje que se desea mover, en cuyo caso se puede mover cualquier eje en cualquier momento).

No funcional Los incrementos deben ser pequeños dependiendo del tipo de material y de herramienta para evitar daños a la herramienta.

III.5.5 Modo introducción manual de datos (IMD)

Caso de uso 2.5 Modo de operación manual IMD (figura 3.16).

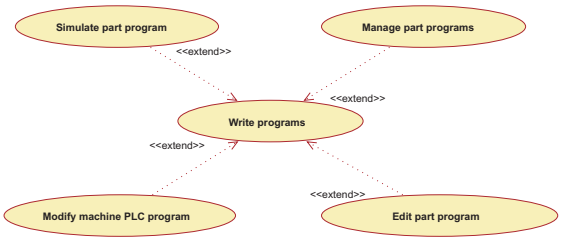


Figura 3.17: Casos de uso para la escritura de programas en código G.

Descripción Este modo de operación es empleado cuando se quiere fabricar una pieza sin crear un programa primero pero se necesita de alta precisión para trayectorias que no son simples como una una línea recta paralela a un eje.

Actores Operador de CNC.

Suposiciones La velocidad de maquinado es baja para evitar daños a la herramienta.

Pasos Se introduce una instrucción en algún lenguaje de programación comprendido por la máquina y se ejecutan las instrucciones inmediatamente.

No funcional Se debe tener entrenamiento apropiado para usar la máquina de esta manera.

III.6. Casos de uso para escritura de programas

Caso de uso 3 Escritura de programas.

Descripción El usuario de CNC administra los programas existentes, crea nuevos programas, edita programas existentes, simula programas o modifica el programa PLC interno.

Actores Programador de CNC y operador de CNC.

Suposiciones Ninguna.

Pasos El usuario puede crear nuevos programas de parte o modificar los ya existentes por medio de un editor de texto integrado en el controlador. Durante la edición de un programa es posible simularlo. También es posible copiar, borrar y transferir programas por medio del administrador de archivos. Existe la posibilidad de modificar el programa del PLC, aunque esto debe ser realizado únicamente por personal calificado para el mantenimiento de la máquina.

No funcional El operador de la máquina debe abstenerse de modificar los programas de la misma en la medida de lo posible, a menos que tenga entrenamiento apropiado.

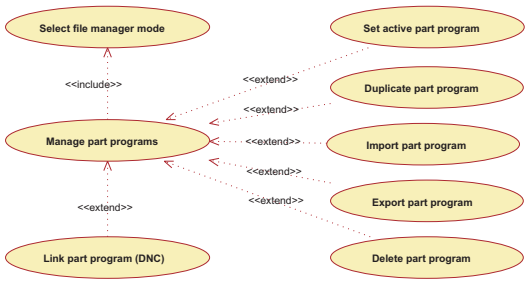


Figura 3.18: Casos de uso para la administración de programas de parte.

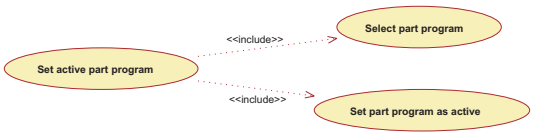


Figura 3.19: Casos de uso para establecer programa activo.

III.6.1 Administración de programas de parte

Caso de uso 3.1 Administración de programas de parte (figura 3.18).

Descripción El programador de la máquina establecerá cuál es el programa de parte activo (el que se ejecutará en modo automático o se cargará por omisión en el editor), podrá copiar o borrar programas existentes, así como transferir (ya sea importar o exportar) programas de parte. Asimismo, es posible establecer un enlace para control numérico directo para la lectura de un programa de parte almacenado en una máquina remota.

Actores Programador de CNC.

Suposiciones El usuario de CNC está entrenado y autorizado para modificar el contenido de la memoria de la máquina herramienta.

Pasos Si el usuario desea crear un programa nuevo, lo puede hacer de dos formas distintas, ya sea desde el editor con la opción correspondiente o desde el administrador de archivos. El editor debe ofrecer las funcionalidades estándar de cualquier editor de texto, y además ofrecerá la opción de simular el programa se esté editando. Desde el administrador de archivos es posible elegir el programa de parte activo, copiar un programa de parte, importar o exportar un programa de parte, borrar un programa de parte o establecer un vínculo para control numérico directo (es decir, establecer el programa de parte activo como un programa leído de manera remota).

No funcional Los programas de parte están escritos en código G o en algún otro formato estándar como STEP-NC.

Caso de uso 3.1.1 Establecer programa de parte activo (figura 3.19).

Descripción Se elige un programa de parte para que sea ejecutado de manera automática o editado por omisión.

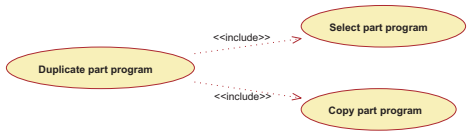


Figura 3.20: Casos de uso para duplicar programa de parte.

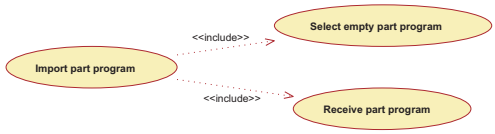


Figura 3.21: Casos de uso para importar programa.

Actores Programador de CNC y operador de CNC.

Suposiciones Al menos hay un programa almacenado en memoria.

Pasos El usuario selecciona un archivo y el sistema ofrece la opción, entre otras, de establecer el programa seleccionado como programa activo, a lo que el sistema responde estableciendo el nombre/número del programa a ejecutar cuando se seleccione el modo automático.

No funcional Un programa de parte contiene la información necesaria para realizar el máximo número de operaciones de maquinado posibles para una pieza dada.

Caso de uso 3.1.2 Duplicar programa de parte (figura 3.20).

Descripción Se crea una copia de un programa existente con la intención de modificar dicha copia posteriormente.

Actores Programador de CNC.

Suposiciones Al menos hay un programa almacenado en memoria.

Pasos Es otra de las opciones que se ofrece al usuario al seleccionarse un archivo en el administrador de archivos; el usuario indica el nombre/número de programa que tendrá la copia y el sistema copia el archivo y se actualiza la base de datos de programas.

No funcional Toda creación de un programa nuevo debe ser documentada.

Caso de uso 3.1.3 Importar programa de parte (figura 3.21).

Descripción Importar programas de parte generados en otro sistema o previamente respaldados.

Actores Programador de CNC.

Suposiciones El sistema externo espera la aceptación de recepción por parte del sistema de control.

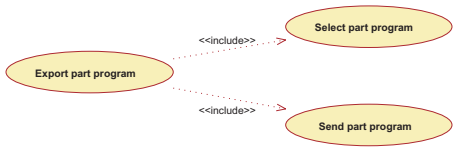


Figura 3.22: Casos de uso para exportar programa.

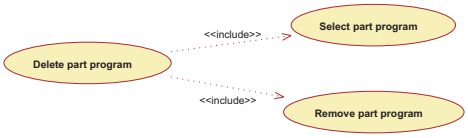


Figura 3.23: Casos de uso para borrar programa de parte.

Pasos Se selecciona desde el administrador de archivos la opción de importar un programa de parte, se indica un nombre o número de programa y se indica al sistema que comience con la recepción del archivo.

No funcional Se debe evitar el ingreso de programas de parte a la computadora sin previa autorización.

Caso de uso 3.1.4 Exportar programa de parte (figura 3.22).

Descripción Exportar programas para respaldarlos o usarlos en otra máquina herramienta.

Actores Programador de CNC.

Suposiciones Al menos hay un programa almacenado en memoria.

Pasos Se selecciona la opción de exportar, se elige el archivo a exportar y se solicita al administrador de archivos que inicie la transferencia hacia el exterior.

No funcional Se debe evitar el acceso no autorizado a los programas de parte para evitar espionaje industrial.

Caso de uso 3.1.5 Borrar un programa de parte (figura 3.23).

Descripción Eliminar un programa de parte que ya no sea necesario.

Actores Programador de CNC.

Suposiciones Existe al menos un programa de parte en la memoria del controlador.

Pasos Se selecciona el programa a eliminar y se elige la opción eliminar.

No funcional Se debe evitar el borrado de programas de parte a la computadora sin previa autorización.

Caso de uso 3.1.6 Establecer enlace para control numérico directo (figura 3.24).

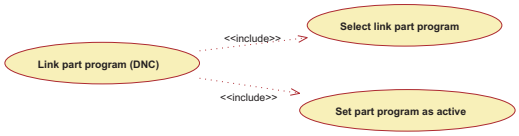


Figura 3.24: Casos de uso para exportar programa.

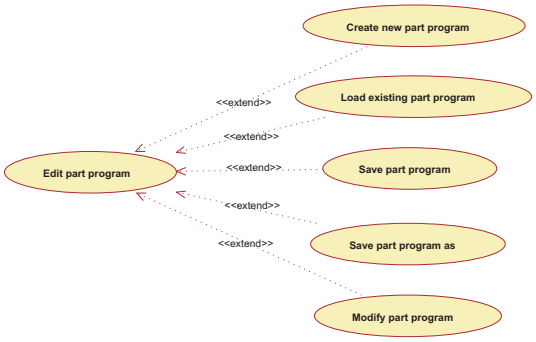


Figura 3.25: Casos de uso para la edición de programas de parte.

Descripción Enlazar el sistema de control con algún dispositivo que provea un flujo de caracteres con las instrucciones de un programa (esto usualmente se conoce como control numérico directo, o DNC por sus siglas en inglés).

Actores Programador de CNC.

Suposiciones El sistema externo espera la aceptación de recepción por parte del sistema de control.

Pasos Se elige la opción DNC y se establece la fuente de datos; el ejecutivo de programas de parte solicitará los datos desde la fuente especificada en vez de desde un programa almacenado localmente.

No funcional DNC debe emplearse solo para programas que no caben en la memoria principal de la computadora.

III.6.2 Edición de programas

Caso de uso 3.2 Edición de programas de parte (figura 3.25).

Descripción El usuario puede crear nuevos programas, abrir programas existentes para modificarlos, guardar programas que haya modificado con el mismo nombre o con otro.

Actores Programador de CNC.

Suposiciones El programador conoce el lenguaje en el que está hecho el programa de parte.

Pasos Para crear un nuevo programa se elige un número o un nombre para el mismo y se agrega una descripción (opcional), ante lo cual se crea una nueva entrada en la base de datos de programas. Una vez creado un programa y abierto en el editor se puede

guardar con el nombre o número original o con alguno distinto, o también es posible cargar un programa ya existente. Ya sea un programa nuevo o un programa cargado desde la base de datos, el editor usado para editar el programa es un editor estándar. No se puede abandonar el editor sin guardar o descartar los cambios primero.

No funcional Es recomendable que la interfaz para editar un programa emplee un teclado lo más parecido al de una computadora personal.

Caso de uso 3.2.1 Crear nuevo programa de parte. (figura 3.25).

Descripción El sistema permite crear y editar un nuevo programa de parte.

Actores Programador de CNC.

Suposiciones Existe suficiente espacio en memoria para crear un nuevo programa.

Pasos Un nuevo programa se puede crear desde el editor eligiendo la opción nuevo programa. Si el programa que se está editando actualmente no ha sido guardado desde la última modificación, se le da la opción al usuario de guardar los cambios. El nuevo programa está vacío y no tiene asignado ni nombre ni número.

No funcional Cada uno de los programas creados debe ser documentado.

Caso de uso 3.2.2 Cargar programa de parte (figura 3.25).

Descripción Se elige un programa almacenado en la memoria del sistema para abrirlo en el editor y poder modificarlo.

Actores Programador de CNC.

Suposiciones Existe al menos un programa en la memoria del sistema.

Pasos En el editor se elige la opción de cargar en memoria (o abrir) un programa. Si el programa que se está editando actualmente no ha sido guardado desde la última modificación, se le da la opción al usuario de guardar los cambios. El programa una vez abierto queda listo para ser modificado.

No funcional Es recomendable hacer modificaciones extensas en un programa en un sistema externo al controlador.

Caso de uso 3.2.3 Guardar programa de parte (figura 3.25).

Descripción Almacenar en memoria permanente el contenido de un programa modificado en el editor.

Actores Programador de CNC.

Suposiciones Existe suficiente memoria en el sistema para almacenar la nueva versión del programa.

Pasos Se elige la opción de guardar el contenido del editor en disco y el sistema guarda los cambios.

No funcional Es recomendable hacer modificaciones extensas en un programa en un sistema externo al controlador.

Caso de uso 3.2.4 Guardar programa de parte con un nuevo nombre o número (figura 3.25).

Descripción Almacenar en memoria permanente el contenido de un programa modificado en el editor con un nuevo nombre o número.

Actores Programador de CNC.

Suposiciones Existe suficiente memoria en el sistema para almacenar la nueva versión del programa.

Pasos Se elige la opción de guardar el contenido del editor con otro nombre o número y el sistema guarda los cambios en el archivo indicado.

No funcional Es recomendable hacer modificaciones extensas en un programa en un sistema externo al controlador.

Caso de uso 3.2.5 Modificar programa de parte (figura 3.25).

Descripción Se agregan o borran instrucciones o se modifican las existentes.

Actores Programador de CNC.

Suposiciones El programador tiene conocimientos sobre la forma de modificar un programa de parte.

Pasos El programador puede agregar un carácter imprimible, borrar un carácter, introducir una nueva línea, desplazarse por el editor, sobreescribir o suprimir caracteres, etc., de la misma forma en que se hace en un editor de textos estándar.

No funcional Es recomendable que la interfaz para editar un programa emplee un teclado lo más parecido al de una computadora personal.

III.6.3 Simulación de programas

Caso de uso 3.3 Simular programa de parte (figura 3.26).

Descripción Durante la escritura de un programa es común verificar si cumple con su propósito por medio de simulación.

Actores Programador de CNC.

Suposiciones El simulador lee el código desde el editor para evitar la necesidad de guardar un programa antes de simularlo y para poder simular modificaciones de programas que se puedan descartar posteriormente.

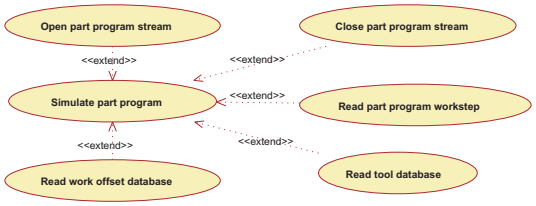


Figura 3.26: Casos de uso para simular un programa de parte.

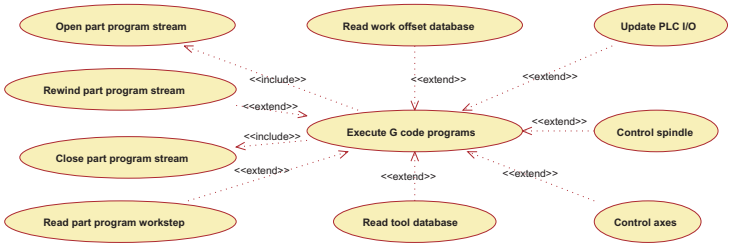


Figura 3.27: Casos de uso para ejecutar un programa de parte.

Pasos El usuario elige la opción de simular programa desde el editor y pasa a la pantalla de simulación hasta que indica lo contrario. La simulación se puede interrumpir en cualquier momento.

No funcional Todo programa se debe simular después de cada modificación.

III.6.4 Modificación del programa del PLC

Caso de uso 3.4 Modificar programa del PLC (figura 3.17).

Descripción En algunos casos, ya sea por la adición o remoción de equipo especial, es necesario modificar el programa del PLC de la máquina herramienta.

Actores Programador de CNC.

Suposiciones El programador de CNC está capacitado para editar el programa del PLC y ha documentado la modificación.

Pasos El programador de CNC elige la opción de modificar el programa del PLC, modifica el programa y puede elegir la opción de guardar los cambios o de revertirlos para regresar a la última versión guardada del programa del PLC.

No funcional El programador de CNC debe documentar cualquier modificación hecha al programa del PLC de la máquina herramienta y explicar cualquier cambio en los procedimientos usuales de maquinado al operador de CNC.

III.7. Casos de uso para ejecución de programas

Caso de uso 4 Ejecución de programas (figura 3.27).

Descripción Ejecución automática de programas.

Actores Operador de CNC.

Suposiciones El programa a ejecutar (activo) ha sido verificado, es decir, está libre de errores.

Pasos Se elige el modo de operación automático y se verifica que no existan condiciones que eviten el inicio de la operación automática. Se presiona el botón de inicio de ciclo en el panel de operación o su equivalente y se inicia el maquinado. Diferentes opciones de ejecución aplican, tales como las opciones de omitir bloque en caso del código G, o la ejecución en seco en cualquier lenguaje de programación usado.

No funcional Se deben cuidar todas las medidas de seguridad durante la operación de la máquina.

Parte II

Diseño

IV. COLABORACIONES

IV.1. Diseño por medio de colaboraciones

Una etapa importante en el diseño de cualquier programa de cómputo es pasar de una vista del sistema del tipo caja negra, como se conceptualiza durante la recopilación de los casos de uso, a una vista del sistema 'de caja gris' en la que el desarrollador deja a un lado la visión monolítica del sistema y la reemplaza por una visión basada en las actividades que realizan diversas 'entidades' dentro del mismo sistema. En otras palabras, este es el primer paso del diseño en el que el desarrollador "echa a volar su imaginación" y comienza a conceptualizar los papeles que juegan (o interpretan) diversas entidades, hasta cierto grado borrosas (es decir, no muy bien definidas), dentro del sistema con el propósito de *llevar a cabo* o *realizar* las actividades descritas en cada caso de uso. Así pues, *una* colaboración es simplemente la forma en que un diseñador imagina la forma en que una colección de entidades podrían implementar la funcionalidad requerida para realizar *el* caso de uso de interés; puede haber más de una colaboración asociada con un caso de uso en virtud de que un caso de uso puede ser una situación compleja que involucre varias acciones.

La siguiente lista enumera los casos de uso del sistema y muestra las colaboraciones asociadas por medio de viñetas:

1. Configurar máquina herramienta.

   1.1 Retorno a cero.

   - Seleccionar modo retorno a cero.
   - Encontrar posición cero para un eje.

   1.2 Actualizar parámetros de la máquina herramienta.

   - Seleccionar parámetros de la máquina herramienta.
   - Modificar parámetro de la máquina herramienta.

   1.3 Actualizar base de datos de desplazamientos de trabajo.

   - Seleccionar base de datos de desplazamientos de trabajo.
   - Modificar registro de coordenada de desplazamiento de trabajo.

   1.4 Actualizar base de datos de herramientas.

   - Seleccionar base de datos de herramientas.
   - Modificar registro de propiedad de herramienta.

2. Operar manualmente.

   2.1 Modo manual (común).

2.1.1 Modificar estado de dispositivo (refrigerante, banda transportadora, etc.)

- Encender dispositivo.
- Apagar dispositivo.
- Dejar dispositivo en modo automático.

2.1.2 Operar husillo.

- Establecer giro en dirección normal.
- Establecer giro en reversa.
- Iniciar giro.
- Detener giro.
- Incrementar velocidad del husillo.
- Decrementar velocidad del husillo.

2.2 Modo rápido.

- Seleccionar modo rápido.
- Incrementar velocidad de avance rápido.
- Decrementar velocidad de avance rápido.
- Impulsar eje rápidamente.

2.3 Modo impulso.

- Seleccionar modo impulso.
- Incrementar velocidad de impulso.
- Decrementar velocidad de impulso.
- Impulsar eje.

2.4 Modo manivela/a pasos.

- Seleccionar modo manivela/a pasos.
- Seleccionar eje.
- Seleccionar factor (incremento por pulso/paso).
- Mover eje por pulso.
- Mover eje un paso.

2.5 Modo IMD.

- Seleccionar modo IMD.
- Ejecutar instrucción.

3. Escribir programas.

3.1 Administrar programas.

- Seleccionar modo administrador de archivos.

3.1.1 Establecer programa de parte activo.

- Elegir programa de parte de la lista de programas.
- Establecer programa de parte como activo.

3.1.2 Duplicar programa de parte.
- Elegir programa de parte de la lista de programas.
- Copiar programa de parte seleccionado.

3.1.3 Importar programa de parte.
- Seleccionar hueco en lista de programas de parte.
- Importar programa de parte.

3.1.4 Exportar programa de parte.
- Elegir programa de parte de la lista de programas.
- Exportar programa de parte.

3.1.5 Borrar programa de parte.
- Elegir programa de parte de la lista de programas.
- Borrar programa de parte.

3.1.6 Establecer enlace para control numérico directo.
- Elegir “control numérico directo” de la lista de programas.
- Establecer programa de parte como activo.

3.2 Editar programas.
- Seleccionar modo editor.

3.2.1 Crear nuevo programa de parte.
- Crear nuevo programa de parte.

3.2.2 Cargar programa de parte existente.
- Cargar programa de parte existente.

3.2.3 Guardar cambios a programa de parte.
- Guardar cambios a programa de parte.

3.2.4 Guardar programa de parte con un nuevo identificador.
- Guardar programa de parte con un nuevo identificador.

3.2.5 Modificar programa de parte.
- Modificar programa de parte.

3.3 Simular programa.
- Abrir flujo de programa de parte.
- Cerrar flujo de programa de parte.
- Leer paso de trabajo de programa de parte.
- Leer base de datos de desplazamientos de trabajo.
- Leer base de datos de herramientas.

3.4 Modificar programa del PLC.
- Modificar programa del PLC.

4. Ejecutar programa de parte.
- Seleccionar modo memoria.

- Abrir flujo de programa de parte (iniciar ciclo).
- Rebobinar flujo de programa de parte.
- Cerrar flujo de programa de parte (abandonar modo memoria).
- Leer paso de trabajo de programa de parte.
- Leer base de datos de desplazamientos de trabajo.
- Leer base de datos de herramientas.
- Controlar husillo.
- Controlar ejes.
- Actualizar entradas y salidas del PLC.

Un tipo de diagrama de gran utilidad para definir una *colaboración* que *realiza* las actividades descritas en un caso de uso es el diagrama de secuencia. Un diagrama de secuencia representa a una entidad por medio de un rectángulo con el nombre de la entidad debajo del cual se traza una línea vertical (usualmente punteada) que representa la *línea de vida* de la misma. Existen dos tipos de diagramas de secuencia:

Diagramas de secuencia de interpretación Representan papeles (o roles) representados (o jugados) por una entidad dada.

Diagramas de secuencia (a secas) Representan objetos de software pertenecientes a alguna clase.

Emplearemos diagramas de secuencia de interpretación para desglosar la funcionalidad necesaria para realizar los casos de uso del capítulo 3 (y que corresponden a las viñetas de la lista anterior).

IV.2. Colaboraciones para configurar la máquina herramienta

IV.2.1 Retorno a cero

Colaboración 1 Seleccionar modo retorno a cero.

Descripción Se selecciona el modo de retorno a cero cuando se desea encontrar la posición de inicio de la máquina (se recomienda después de un paro de emergencia) y cuando se acaba de iniciar el sistema de control numérico.

Diagrama Véase la figura 4.1.

Pasos El usuario de CNC indica, a través de un teclado o algún otro mecanismo de comunicación (podría ser comunicación en red), que se desea elegir el modo retorno a cero. El núcleo del CNC recibe la solicitud del administrador de comunicaciones e indica al ejecutivo que requiere habilitar el modo retorno a cero a lo que se responde con el identificador del modo seleccionado. Si el modo seleccionado en el manejador de operación manual es efectivamente el modo retorno a cero entonces se selecciona la pantalla de configuraciones con el despliegue correspondiente al modo retorno a cero.

Colaboración 2 Encontrar la posición cero para un eje.

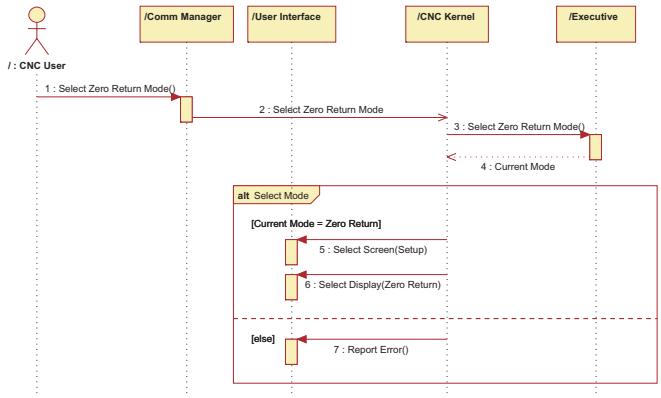


Figura 4.1: Diagrama de secuencia para seleccionar modo retorno a cero.

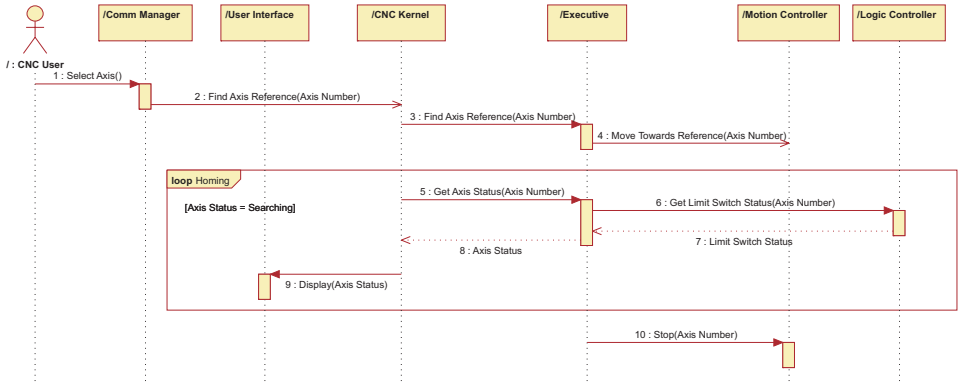


Figura 4.2: Diagrama de secuencia para encontrar la posición cero para un eje.

Descripción En el modo retorno a cero se elige un eje a la vez para localizar su posición de inicio (cero máquina).

Diagrama Véase la figura 4.2.

Pasos El usuario de CNC indica, a través de un teclado o algún otro mecanismo de comunicación (podría ser comunicación en red), que se desea encontrar la posición cero para algún eje. El núcleo del CNC recibe la solicitud del administrador de comunicaciones e indica al ejecutivo que requiere encontrar la posición cero para algún eje a lo que se responde solicitando controlador de movimiento que inicie el movimiento de búsqueda. Se entra entonces en un ciclo en el cual se actualiza el estado del eje en la interfaz gráfica hasta que se encontraba la oposición cero del eje.

IV.2.2 Actualizar parámetros de la máquina herramienta

Colaboración 3 Seleccionar parámetros de la máquina herramienta.

Descripción Para editar un parámetro de la máquina herramienta primero es necesario seleccionar la pantalla correspondiente en la interfaz gráfica de usuario.

Diagrama Véase la figura 4.3.

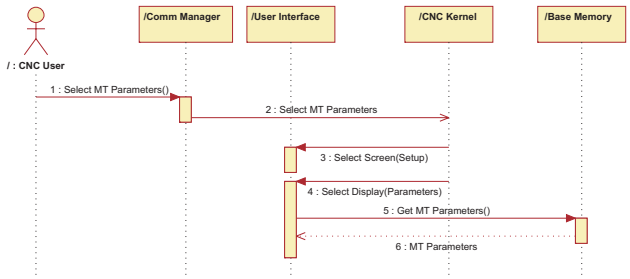


Figura 4.3: Diagrama de secuencia para seleccionar parámetros de la máquina herramienta (en la interfaz gráfica de usuario).

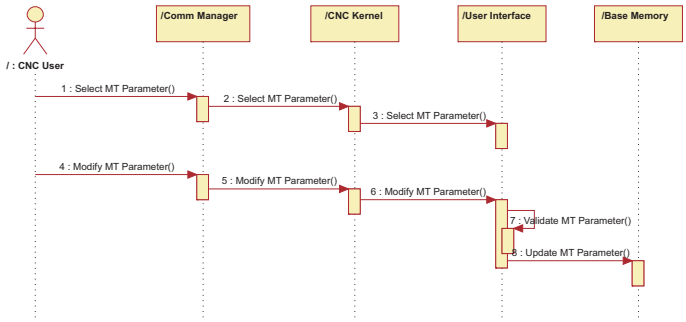


Figura 4.4: Diagrama de secuencia para modificar un parámetro de la máquina herramienta (en la interfaz gráfica de usuario).

Pasos El usuario de CNC indica al Núcleo del CNC, a través de un teclado o algún otro mecanismo de comunicación (podría ser comunicación en red), que se desea seleccionar la pantalla de configuración en la modalidad de parámetros de la máquina herramienta. El núcleo solicita a la interfaz gráfica y seleccione la pantalla y modalidad deseadas.

Colaboración 4 Modificar parámetro de la máquina herramienta.

Descripción Los parámetros de las máquinas herramienta almacenados en la memoria base se pueden modificar con ayuda de la interfaz gráfica del usuario.

Diagrama Véase la figura 4.4.

Pasos El usuario de CNC indica al Núcleo del CNC, a través de un teclado o algún otro mecanismo de comunicación (podría ser comunicación en red), que se desea seleccionar un parámetro de la máquina herramienta en la pantalla y modalidad apropiadas en la interfaz gráfica el usuario con el propósito de modificar su valor. Una vez hecha la modificación de interfaz gráfica avisa de la misma a la memoria base.

IV.2.3 Actualizar base de datos de desplazamientos de trabajo

Colaboración 5 Seleccionar base de datos de desplazamientos de trabajo. Esta colaboración es casi idéntica a la colaboración 3, pero los desplazamientos de trabajo se almacenan en la base de datos, no en la memoria base.

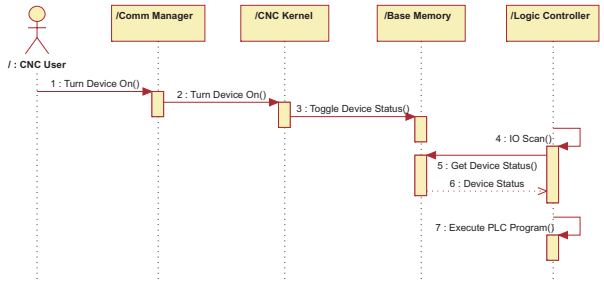


Figura 4.5: Diagrama de secuencia para encender un dispositivo.

Colaboración 6 Modificar registro de coordenada de desplazamiento de trabajo. Esta colaboración es casi idéntica a la colaboración 4, pero los desplazamientos de trabajo se almacenan en la base de datos, no en la memoria base.

IV.2.4 Actualizar base de datos de herramientas

Colaboración 7 Seleccionar base de datos de herramientas. Esta colaboración es casi idéntica a la colaboración 3, pero las propiedades de las herramientas se almacenan en la base de datos, no en la memoria base.

Colaboración 8 Modificar registro de propiedad de herramienta. Esta colaboración es casi idéntica a la colaboración 4, pero las propiedades de las herramientas se almacenan en la base de datos, no en la memoria base.

IV.3. Colaboraciones para operar manualmente la máquina herramienta

IV.3.1 Modo manual común

Modificar estado de dispositivo

Colaboración 9 Encender dispositivo.

Descripción Durante la operación en cualquiera de los modos manuales, además de dar órdenes a los ejes y al husillo de la máquina, también es deseable encender dispositivos tales como refrigerante o la luz de iluminación.

Diagrama Véase la figura 4.5.

Pasos El usuario de CNC indica al Núcleo del CNC, a través de un teclado o algún otro mecanismo de comunicación (podría ser comunicación en red), que se desea encender un dispositivo, así que se modifica el estado del dispositivo en la Memoria base. Cuando el controlador lógico de la localidad de memoria el programa del mismo hace que el controlador lógico active el dispositivo.

Colaboración 10 Apagar dispositivo. Esta colaboración es casi idéntica a la colaboración 9, salvo que el valor escrito en Memoria base indica al Controlador lógico que el dispositivo debe ser deshabilitado.

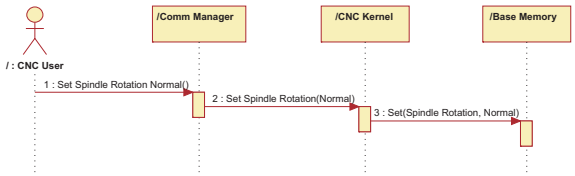


Figura 4.6: Diagrama de secuencia para establecer la dirección normal del husillo.

Colaboración 11 Dejar dispositivo en modo automático. Esta colaboración es casi idéntica a la colaboración 9, salvo que el valor escrito en Memoria base indica al Controlador lógico que el dispositivo debe permanecer en modo automático.

Operar husillo

Colaboración 12 Establecer giro en dirección normal.

Descripción El husillo puede girar en dirección normal por el reversa. Es importante indicar al sistema la dirección de giro del mismo antes de iniciar su rotación.

Diagrama Véase la figura 4.6.

Pasos El usuario de CNC indica al Núcleo del CNC, a través de un teclado o algún otro mecanismo de comunicación (podría ser comunicación en red), que se desea establecer la dirección de giro del husillo en dirección normal, así que se modifica la dirección de giro del husillo en la Memoria base.

Colaboración 13 Establecer giro en reversa. Esta colaboración es idéntica a la colaboración 12 salvo por el sentido de giro.

Colaboración 14 Iniciar giro. Esta colaboración es idéntica colaboración 9, salvo que el valor escrito en Memoria base indica al Controlador lógico que el husillo debe ser habilitado.

Colaboración 15 Detener giro. Esta colaboración es idéntica colaboración 9, salvo que el valor escrito en Memoria base indica al Controlador lógico que el husillo debe ser deshabilitado.

Colaboración 16 Incrementar velocidad del husillo. Esta colaboración es idéntica a la colaboración 12 salvo que el valor escrito incrementa la velocidad del husillo y no su sentido de giro.

Colaboración 17 Decrementar velocidad del husillo. Esta colaboración es idéntica a la colaboración 12 salvo que el valor escrito decrementa la velocidad del husillo y no su sentido de giro.

IV.3.2 Modo rápido

Colaboración 18 Seleccionar modo rápido. Esta colaboración es casi idéntica a la colaboración 1, salvo por el modo seleccionado en el ejecutivo y la la pantalla elegida en la interfaz gráfica.

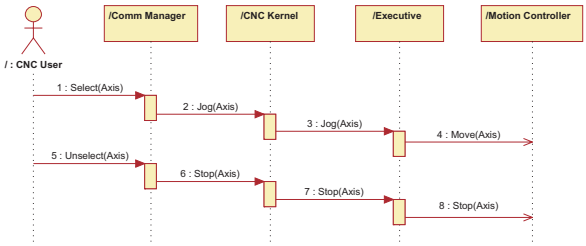


Figura 4.7: Diagrama de secuencia para impulsar un eje.

Colaboración 19 Incrementar velocidad de avance rápido. Esta colaboración es idéntica a la colaboración 12 salvo que el valor escrito incrementa la velocidad de avance rápido.

Colaboración 20 Decrementar velocidad de avance rápido. Esta colaboración es idéntica a la colaboración 12 salvo que el valor escrito decrementa la velocidad de avance rápido.

Colaboración 21 Impulsar eje rápidamente. Esta colaboración es idéntica a la colaboración 25; lo único que cambia ese modo de operación en el que se encuentra el sistema de control en un momento dado.

IV.3.3 Modo impulso

Colaboración 22 Seleccionar modo impulso. Esta colaboración es casi idéntica a la colaboración 1, salvo por el modo seleccionado en el ejecutivo y la la pantalla elegida en la interfaz gráfica.

Colaboración 23 Incrementar velocidad de impulso. Esta colaboración es idéntica a la colaboración 12 salvo que el valor escrito incrementa la velocidad de impulso.

Colaboración 24 Decrementar velocidad de impulso. Esta colaboración es idéntica a la colaboración 12 salvo que el valor escrito decrementa la velocidad de impulso.

Colaboración 25 Impulsar eje.

Descripción Cuando se desea mover un eje de manera continua (sin preocuparse por pasos discretos) se puede impulsar un eje durante el tiempo en que permanezca presionado el botón correspondiente.

Diagrama Véase la figura 4.7.

Pasos El usuario de CNC indica, a través de un teclado o algún otro mecanismo de comunicación (podría ser comunicación en red), que se desea iniciaron el movimiento de un eje de manera continua, es decir, impulsar el eje. Cuando ya no se desea mover el eje, de igual manera, el usuario de CNC indica a través del mismo medio de comunicación que se desea detener el movimiento; usualmente el usuario indica su intención de mover y de detener el eje por medio de un botón de retorno automático cuando el usuario es humano.

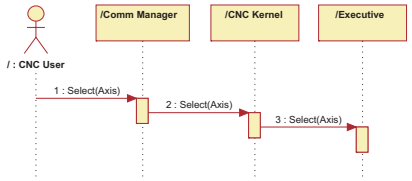


Figura 4.8: Diagrama de secuencia para seleccionar un eje en el generador de pulsos.

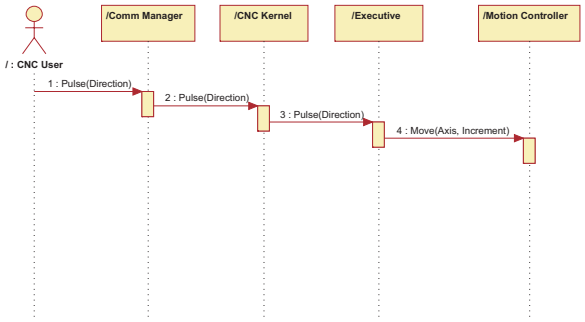


Figura 4.9: Diagrama de secuencia para mover un eje con el generador de pulsos.

IV.3.4 Modo manivela/a pasos

Colaboración 26 Seleccionar modo manivela/a pasos Esta colaboración es casi idéntica a la colaboración 1, salvo por el modo seleccionado en el ejecutivo y la la pantalla elegida en la interfaz gráfica.

Colaboración 27 Seleccionar eje.

Descripción El modo manivela/a pasos emplea un generador de pulsos para un eje que debe ser previamente seleccionado.

Diagrama Véase la figura 4.8.

Pasos El usuario de CNC indica, a través de un teclado o algún otro mecanismo de comunicación (podría ser comunicación en red), que se desea elegir el eje en cuestión, ante lo cual el Núcleo del CNC le hace saber de la elección al Ejecutivo.

Colaboración 28 Seleccionar factor (incremento por pulso/paso). Esta colaboración es idéntica a la colaboración 27, salvo que es un factor de incremento lo que se establece.

Colaboración 29 Mover eje por pulso.

Descripción En el modo manivela es posible mover un eje por medio de un pulso producido por un generador de pulsos.

Diagrama Véase la figura 4.9.

Pasos El usuario de CNC indica, a través de un teclado o algún otro mecanismo de comunicación (podría ser comunicación en red), que se desea mover un eje por medio de un

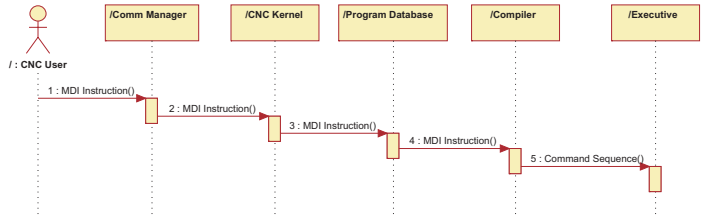


Figura 4.10: Diagrama de secuencia para ejecutar instrucciones en modo IMD.

pulso al Núcleo del CNC, quien hace saber de este hecho al Ejecutivo quien se encarga de solicitar el movimiento en la dirección y magnitud apropiadas al Controlador de movimiento.

Colaboración 30 Mover eje un paso. Esta colaboración es idéntica a la colaboración 29, sólo que en este caso la acción se indica por medio de un botón de selección de un eje.

IV.3.5 Modo IMD

Colaboración 31 Seleccionar modo IMD. Esta colaboración es casi idéntica a la colaboración 1, salvo por el modo seleccionado en el ejecutivo y la la pantalla elegida en la interfaz gráfica.

Colaboración 32 Ejecutar instrucción.

Descripción En modo IMD es posible ejecutar instrucciones que aparecerían como parte de un programa de parte pero de forma interactiva.

Diagrama Véase la figura 4.10.

Pasos El usuario de CNC indica, a través de un teclado o algún otro mecanismo de comunicación (podría ser comunicación en red), que se desea ejecutar una instrucción de maquinado de manera interactiva, así que el Núcleo del CNC envía los comandos al flujo de datos para IMD de la Base de datos de programas, quien las envía a su vez al Compilador y al Ejecutivo para su procesamiento por parte del Control lógico y el Control de movimiento.

IV.4. Colaboraciones para escribir programas

IV.4.1 Administrar programas

Colaboración 33 Seleccionar modo administrador de archivos. Esta colaboración es casi idéntica a la colaboración 1, salvo por el modo seleccionado en el ejecutivo y la la pantalla elegida en la interfaz gráfica.

Establecer programa de parte activo

Colaboración 34 Elegir programa de parte de la lista de programas.

Descripción Antes de establecer un programa de parte como el programa activo hay que seleccionarlo de la lista de los programas disponibles.

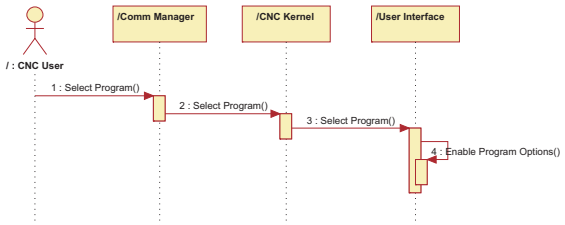


Figura 4.11: Diagrama de secuencia para seleccionar un programa de la lista de programas disponibles.

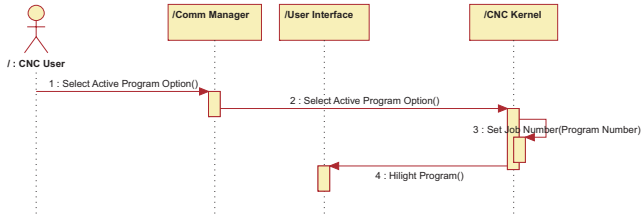


Figura 4.12: Diagrama de secuencia para establecer como activo el programa seleccionado en la lista de programas disponibles.

Diagrama Véase la figura 4.11.

Pasos El usuario de CNC indica, a través de un teclado o algún otro mecanismo de comunicación (podría ser comunicación en red), que se desea seleccionar un programa de la lista, así que el Núcleo del CNC lo hace saber a la Interfaz gráfica.

Colaboración 35 Establecer programa de parte como activo.

Descripción Es posible establecer un programa de parte seleccionado como activo eligiendo la opción correspondiente.

Diagrama Véase la figura 4.12.

Pasos El usuario de CNC indica, a través de un teclado o algún otro mecanismo de comunicación (podría ser comunicación en red), que se desea establecer como activo un programa seleccionado en la lista, así que el Núcleo del CNC lo establece como activo y lo hace saber a la Interfaz gráfica.

Duplicar programa de parte

Colaboración 36 Copiar programa de parte seleccionado.

Descripción El programa de parte actualmente seleccionado puede ser duplicado.

Diagrama Véase la figura 4.13.

Pasos El usuario de CNC indica, a través de un teclado o algún otro mecanismo de comunicación (podría ser comunicación en red), que se desea duplicar un programa seleccionado en la lista, así que el Núcleo del CNC lo duplica en la Base de datos de programas.

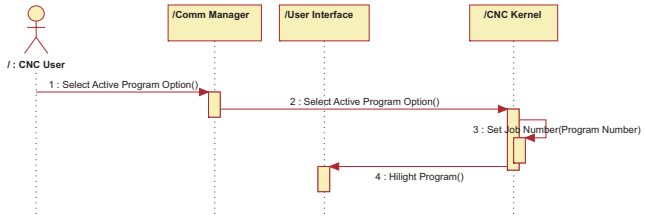


Figura 4.13: Diagrama de secuencia para duplicar el programa seleccionado en la lista de programas disponibles.

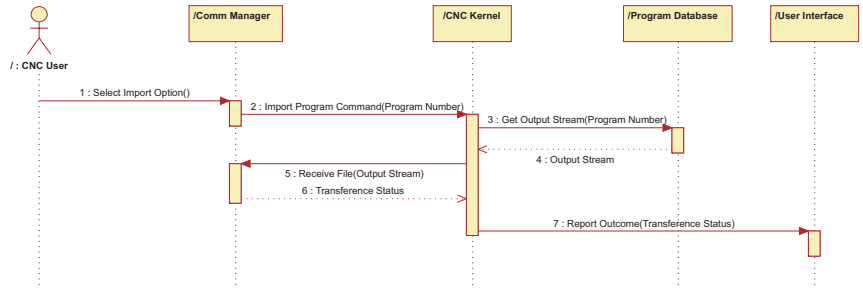


Figura 4.14: Diagrama de secuencia para importar un programa de parte.

Importar programa de parte

Colaboración 37 Seleccionar hueco en lista de programas de parte. Esta colaboración es casi idéntica a la colaboración 34, salvo que lo que se selecciona es un espacio vacío en vez de uno con un nombre de programa.

Colaboración 38 Importar programa de parte.

Descripción Cuando se desea recibir un programa producido en un sistema externo se emplea la opción importar programa de parte.

Diagrama Véase la figura 4.14.

Pasos El usuario de CNC indica, a través de un teclado o algún otro mecanismo de comunicación (podría ser comunicación en red), que se desea importar un programa y almacenarlo en el hueco de la lista seleccionado, ante lo cual el Núcleo del CNC solicita un flujo para escritura a la Base de datos de programas, el cual es usado para almacenar el programa recibido por medio del Administrador de comunicaciones.

Exportar programa de parte

Colaboración 39 Exportar programa de parte. Esta colaboración es casi idéntica a la colaboración 38, salvo por la dirección de transferencia del programa.

Borrar programa de parte

Colaboración 40 Borrar programa de parte. Esta colaboración es casi idéntica a la colaboración 36, salvo que en vez de duplicar el programa, el mismo es borrado.

Establecer enlace para control numérico directo

Colaboración 41 Elegir “control numérico directo” de la lista de programas. Esta colaboración es casi idéntica a la colaboración 34, salvo que lo que se selecciona es el espacio de la lista que corresponde al programa para “control numérico directo”.

IV.4.2 Editar programas

Colaboración 42 Seleccionar modo editor. Esta colaboración es casi idéntica a la colaboración 1, salvo por el modo seleccionado en el ejecutivo y la la pantalla elegida en la interfaz gráfica.

Crear nuevo programa de parte

Colaboración 43 Crear nuevo programa de parte.

Descripción El editor del sistema de CNC permite crear un nuevo programa de parte eligiendo la opción correspondiente.

Diagrama Véase la figura 4.15.

Pasos Los pasos a seguir son los mismos que en cualquier editor estándar.

Cargar programa de parte existente

Colaboración 44 Cargar programa de parte existente.

Descripción El editor del sistema de CNC permite abrir un programa de parte para editarlo eligiendo la opción correspondiente.

Diagrama Véase la figura 4.16.

Pasos Los pasos a seguir son los mismos que en cualquier editor estándar.

Guardar cambios a programa de parte

Colaboración 45 Guardar cambios a programa de parte.

Descripción El editor del sistema de CNC permite guardar un programa de parte eligiendo la opción correspondiente.

Diagrama Véase la figura 4.17.

Pasos Los pasos a seguir son los mismos que en cualquier editor estándar.

Guardar programa de parte con un nuevo identificador

Colaboración 46 Guardar programa de parte con un nuevo identificador.

Descripción El editor del sistema de CNC permite guardar un programa de parte con un nuevo identificador eligiendo la opción correspondiente.

Diagrama Véase la figura 4.18.

Pasos Los pasos a seguir son los mismos que en cualquier editor estándar.

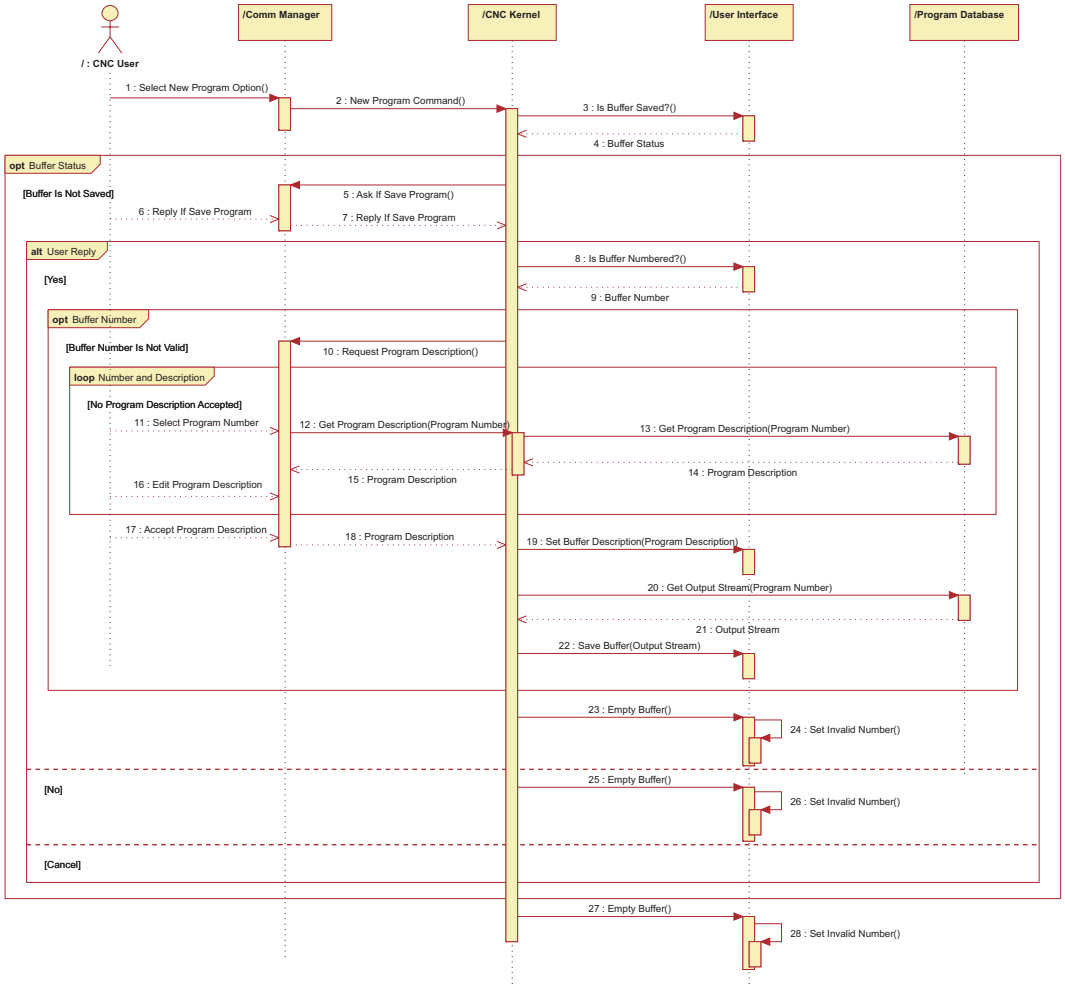


Figura 4.15: Diagrama de secuencia para crear un nuevo programa de parte.

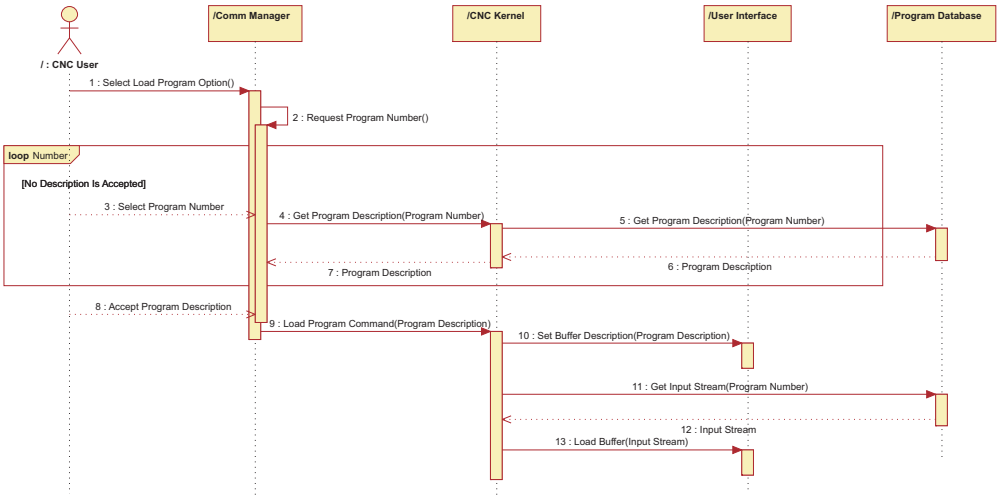


Figura 4.16: Diagrama de secuencia para abrir un programa de parte.

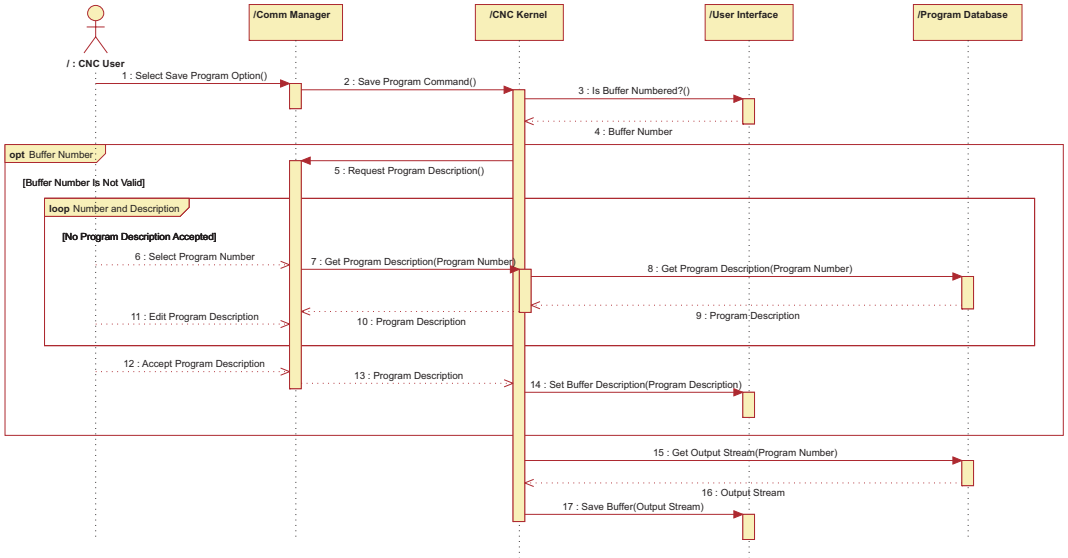


Figura 4.17: Diagrama de secuencia para guardar un programa de parte.

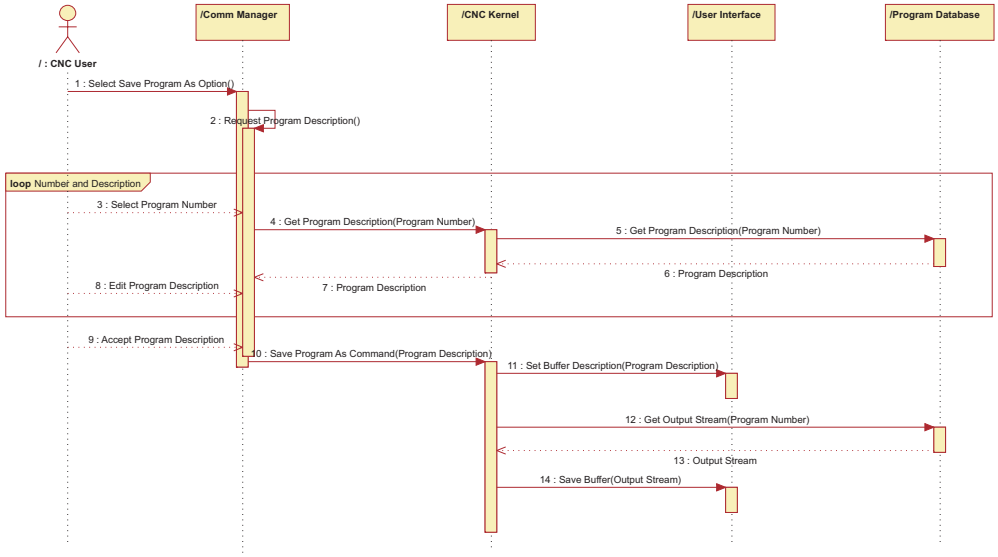


Figura 4.18: Diagrama de secuencia para guardar un programa de parte con un nuevo identificador.

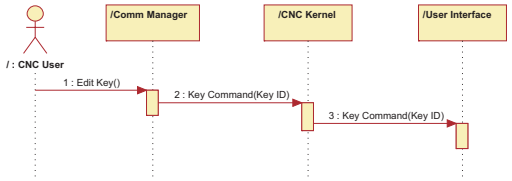


Figura 4.19: Diagrama de secuencia para modificar un programa de parte.

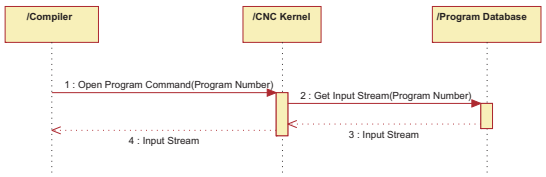


Figura 4.20: Diagrama de secuencia para abrir el flujo de un programa de parte.

Modificar programa de parte

Colaboración 47 Modificar programa de parte.

Descripción Es posible modificar un programa de parte empleando el panel de control o su equivalente.

Diagrama Véase la figura 4.19.

Pasos Los pasos a seguir son los mismos que en cualquier editor estándar.

IV.4.3 Simular programa

Colaboración 48 Abrir flujo de programa de parte.

Descripción Cuando se desea ejecutar un programa es necesario proporcionar un flujo de datos al Compilador.

Diagrama Véase la figura 4.20.

Pasos El Compilador solicita un flujo de datos de entrada al Núcleo del CNC quien a su vez lo obtiene de la Base de datos de programas.

Colaboración 49 Rebobinar flujo de programa de parte.

Descripción Cuando se desea ejecutar nuevamente el programa activo es necesario rebobinarlo primero.

Pasos El compilador simplemente rebobina el flujo de programa del cual está leyendo.

Colaboración 50 Cerrar flujo de programa de parte.

Descripción Cuando el sistema emplea un modo de operación distinto de la operación automática, debe cerrarse el programa activo.

Pasos El compilador simplemente cierra el flujo de programa del cual está leyendo.

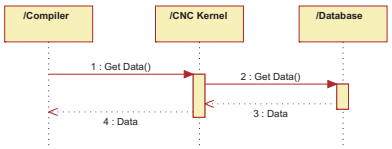


Figura 4.21: Diagrama de secuencia para leer la base de datos.

Colaboración 51 Leer paso de trabajo de programa de parte.

Descripción Durante la ejecución del programa activo es necesario leer un paso de trabajo para ejecutarlo.

Diagrama

Pasos El compilador lee un paso de trabajo del programa de parte.

Colaboración 52 Leer base de datos de desplazamientos de trabajo.

Descripción Los desplazamientos de trabajo se emplean durante la ejecución de las instrucciones de un programa para ubicar la posición de una parte a maquinar.

Diagrama Véase la figura 4.21.

Pasos El Compilador solicita un desplazamiento de trabajo al Núcleo del CNC quien lo obtiene de la base de datos.

Colaboración 53 Leer base de datos de herramientas.

Descripción Las características geométricas de las herramientas se emplean durante la ejecución de las instrucciones de un programa para compensar radios y otras longitudes de las mismas.

Diagrama Véase la figura 4.21.

Pasos El Compilador solicita alguna propiedad de alguna herramienta al Núcleo del CNC quien lo obtiene de la base de datos.

IV.4.4 Modificar programa del PLC

Colaboración 54 Modificar programa del PLC. Esta colaboración es casi idéntica a la colaboración 47, salvo por que el programa modificado es la lista de instrucciones del PLC.

IV.5. Colaboraciones para ejecutar programas

Para ejecutar programas también se requiere de

- Abrir flujo de programa de parte (al iniciar ciclo).
- Rebobinar flujo de programa de parte.
- Cerrar flujo de programa de parte (al abandonar modo memoria).

- Leer paso de trabajo de programa de parte.
- Leer base de datos de desplazamientos de trabajo.
- Leer base de datos de herramientas.

Colaboración 55 Seleccionar modo memoria. Esta colaboración es casi idéntica a la colaboración 1, salvo por el modo seleccionado en el ejecutivo y la la pantalla elegida en la interfaz gráfica.

Colaboración 56 Controlar husillo.

Descripción Cuando se quiere efectuar alguna acción relativa al husillo que ha sido solicitada por el usuario directamente o almacenada en un programa, es necesario solicitar la acción al Controlador de movimiento.

Pasos El ejecutivo solicita la acción al Controlador de movimiento.

Colaboración 57 Controlar ejes.

Descripción Cuando se quiere efectuar alguna acción relativa a los ejes que ha sido solicitada por el usuario directamente o almacenada en un programa, es necesario solicitar la acción al Controlador de movimiento.

Pasos El ejecutivo solicita la acción al Controlador de movimiento.

Colaboración 58 Actualizar entradas y salidas del PLC.

Descripción La actualización de las entradas y salidas del PLC es una actividad periódica (de segundo plano) realizada por el mismo.

Pasos El PLC lee los módulos de entrada, escribe sus valores en la Memoria base, realiza los cálculos necesarios y escribe los valores calculados en su módulo de salida.

V. MÓDULOS

V.1. Modularización

V.1.1 Módulos: paquetes y componentes

En general, al desarrollar productos integrados por varios componentes, se considera que una característica básica de 'buen' diseño es particionar a los sistemas en componentes de menor tamaño. La principal razón para particionar es la de reducir el problema general a uno más pequeño de proporciones menores que se pueda resolver más fácilmente (dentro de lo humanamente posible). En términos de software este proceso se llama 'modularización', y los elementos que resultan de este proceso se llaman 'módulos'. Un requerimiento básico es que los módulos sean suficientemente simples, lo cual nos permitirá:

- Comprender más fácilmente su propósito y estructura.
- Verificar que el resultado final corresponde al diseño original y validar que el diseño realmente resuelva el problema.
- Apreciar su interacción con otros módulos.
- Valorar su efecto sobre la estructura general del software y su operación.

Pero, ¿qué es exactamente un módulo? Hay muchas definiciones posibles; en nuestro caso se considera una "unidad de software que tiene funciones bien definidas e interfaces bien definidas para otros elementos del programa". Claramente, un módulo puede ser definido empleando una gran variedad de técnicas, desde un diseño procedural, pasando por uno orientado a objetos, hasta uno orientado unicamente a describir el procesamiento de los datos. Pero antes de hacer el diseño de cualquiera de los módulos, el diseñador debe responder a una pregunta importante: ¿de qué manera debe ser modularizado el sistema de manera correcta?

No hay una solución única a esta pregunta, pues en gran parte depende de la aplicación en particular, y aún hablando de una misma aplicación se pueden emplear distintos enfoques. Un enfoque muy común es modularizar una arquitectura en dos etapas:

1. Por paquetes, es decir, segmentando la aplicación en módulos de gran tamaño cada uno de los cuales se identifica con conjunto de actividades importantes que debe realizar la aplicación. Los paquetes propuestos para un sistema de control para fresadora se muestran en la figura 5.1.
2. Por componentes, contenidos dentro de los paquetes anteriores, representan una entidad funcional dentro de la aplicación.

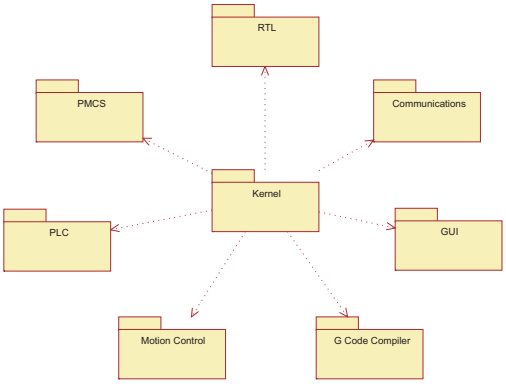


Figura 5.1: Paquetes propuestos para modularizar la arquitectura del sistema de control para una fresadora de CNC.

En la práctica los paquetes se pueden hacer corresponder con diferentes construcciones dentro de lenguaje de programación, tales como los paquetes de Java o los espacios de nombres de C++.

Podemos definir, burdamente, un componente como una entidad de software que provee servicios relacionados entre sí por medio de interfaces bien definidas. Usualmente los servicios son ofrecidos por un componente empleando una misma tecnología, algoritmo, o técnica. Un componente usualmente se implementa como una clase especial a partir de la cual se puede tener un único objeto y que hace las funciones de fachada, es decir coordinador de otras clases, con el propósito de ofrecer la funcionalidad establecida en las interfaces.

V.1.2 Interfaces: Una medida de independencia de los componentes

Los módulos del sistema no pueden existir en aislamiento. Cada uno está diseñado para realizar una parte del total de las funciones del sistema; por consiguiente deben comunicarse unos con otros. La experiencia muestra que la cantidad de interacción entre los módulos tiene un efecto significativo sobre la calidad del software. En los casos en los que los módulos están bien definidos, realizan funciones claras y tienen *interfaces* simples los niveles de interacción son bajos. En otras palabras, los módulos son relativamente independientes. Dichos diseños son relativamente fáciles de entender, además de ser confiables y de fácil mantenimiento cuando se ponen en servicio. Además el software muestra un alto grado de confiabilidad en virtud de que los cambios dentro de un módulo dado no se 'esparcen' a lo largo del programa.

Existen factores que influyen sobre la calidad de las interfaces entre los componentes:

Complejidad Cuando nuestro objetivo es un bajo nivel de interacción necesitamos distinguir entre la cantidad de la información y la complejidad de la información. Transferir una gran cantidad de datos entre componentes puede ser una tarea relativamente sencilla—si es que estamos lidiando con tipos de datos simples—. Por ejemplo, copiar un arreglo de 500 elementos es al mismo tiempo directo y fácil de comprender. Sin embargo, un componente que transfiera a través de su interfaz cinco variables con diferentes tipos de datos puede ser mucho más difícil de comprender. Por consiguiente, reducir

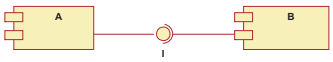


Figura 5.2: El componente A provee de información al componente B por medio de la interfaz I.

la complejidad de la información que debe ser transferida entre módulos es una meta importante del diseño.

Tipos de datos Dependiendo del tamaño del tipo de datos se invierte cierta cantidad de tiempo transfiriendo información de un módulo a otro. Efectivamente, el tipo de datos empleado para transferir información puede influir en la cantidad de trabajo que el módulo receptor invierte para acceder a la misma. Si el tipo de datos es una referencia a la información en el mismo espacio de direcciones entonces es posible acceder únicamente a aquella información que se desea, pero es necesario desreferenciar la información primero, i.e., sumar el desplazamiento adecuado a la dirección base antes de poder leer la información. Por otra parte si el tipo de datos es un valor copiado de otra ubicación de memoria en otro módulo, entonces la información está directamente disponible pero es necesario invertir un tiempo directamente proporcional al tamaño de los datos para la copia de los mismos.

Métodos de comunicación En el escenario más simple, el intercambio de información, se cuenta con dos componentes, digamos, A y B. El componente A puede proveer información al módulo B, usualmente a través de una función que pertenece la interfaz del componente A y cuya llamada es realizada en el componente B (figura 5.2). En un escenario análogo es posible que ambos componentes intercambien información por medio de llamadas mutuas, es decir, que al mismo tiempo que el componente A provea de información al B, el B provea de información al A. Es posible que el acoplamiento entre dos módulos sea más fuerte cuando, además de intercambiar información, uno de los módulos realice acciones de control en el otro módulo por medio de señales de control, tales como banderas o llamadas inversas (*callbacks* en inglés); igual que con el intercambio básico de información, el acoplamiento de control puede ser un proceso en ambos sentidos. El caso más general involucra tanto intercambio de información como acomplamiento de control en ambos sentidos.

V.2. Arquitecturas abiertas

Se dice que una arquitectura es abierta si está modularizada y todas sus interfaces están documentadas. En consecuencia, los componentes de una arquitectura abierta son intercambiables si ofrecen las mismas interfaces (y por lo tanto la misma funcionalidad) y puede haber más de un vendedor para cada uno de los componentes (figura 5.3). El hecho de que una arquitectura sea abierta no significa que sea la única arquitectura posible para el sistema implementado; simplemente significa que *alguien* ha hecho pública la forma en que los componentes de un sistema colaboran entre sí.

No se debe confundir el concepto de arquitectura abierta con el concepto de fuente abierta. Se dice que un programa de cómputo es de fuente abierta si la implementación de cada uno de sus componentes es pública, es decir, si el código fuente de los mismos está

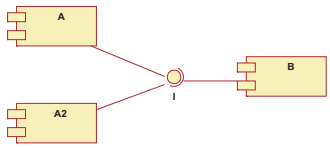


Figura 5.3: El componente A y el componente A_2 son intercambiables; cualquiera de los dos puede proveer de información al componente B por medio de la interfaz I.

disponible a cualquiera interesado en usarlo. De manera parecida a lo que ocurre con las arquitecturas abiertas, un programa de fuente abierta es solamente una de muchas opciones posibles para la implementación de un sistema.

De hecho, una arquitectura abierta puede permitir la coexistencia 'pacífica' entre componentes de fuente abierta y componentes de fuente cerrada si tanto unos como otros se apegan a las interfaces documentadas. De hecho, esa es la forma en que operan algunos componentes del sistema operativo Linux ofrecidos por fabricantes de componentes de fuente cerrada como Nvidia.

Así pues, es factible construir un modelo de negocios que incluya tanto a proveedores de fuente abierta como de fuente cerrada bajo el mismo concepto de arquitectura abierta. Puede existir, de hecho, un arquitectura de referencia de fuente abierta sobre la cual se puede verificar el correcto funcionamiento de cualquier componente de fuente cerrada.

V.3. Componentes para un sistema de CNC

A partir de los diagramas de secuencia del capítulo anterior es posible identificar diversas entidades que interactúan entre sí para desarrollar el trabajo del control numérico por computadora para máquinas herramienta. Algunos de estas entidades son objetos físicos, mientras que algunos otros son software encargado de la lógica de control. Los componentes se pueden identificar a partir de los diagramas de secuencia de manera explícita, tales como el controlador de movimiento.

En otros casos los papeles de los diagramas de secuencia no son 'interpretados' por un componente del software sino por solo una parte del componente; por ejemplo, el teclado si bien es un componente físico, también es un servicio proveído por el sistema que traduce las señales eléctricas del teclado físico en algo estándar que puede leer el programa de control sin necesidad de comprender de manera precisa cómo opera el dispositivo de hardware. De hecho, cualquier otro medio para generar las mismas señales podría ser igual de válido tal como un teclado físico verdadero para efectos prácticos. Así pues, un teclado puede ser visto de tres maneras distintas:

Dispositivo físico Como un dispositivo físico formado por botones que generan señales eléctricas,

Componente como una entidad de software que traduce señales de hardware a información que comprende el resto del programa de control,

Interfaz o como una entidad de software que envía señales (no necesariamente de hardware) o permite obtenerlas al resto del programa, es decir, sin preocuparse del verdadero origen de las señales del teclado (lo que permitiría la operación remota del equipo).

En realidad, determinar qué es un componente y qué es simplemente un servicio proveído como una interfaz es principalmente una cuestión de conveniencia en el diseño —no obstante, un diseño conveniente resulta ser algo bastante deseable. Para explicar mejor este punto podemos establecer una analogía con la organización de un gobierno de un país donde una agencia gubernamental puede ser la encargada de realizar ciertas funciones públicas y en cualquier momento una de dichas agencias puede separarse en otras agencias para que cada una de las nuevas agencias siga proveyendo los servicios de la agencia original (y otros servicios más u alguna ventaja de algún tipo). Por ejemplo, al momento de escribir estas líneas, el servicio de defensa del espacio aéreo y terrestre de México es proveído por la Secretaría de la Defensa Nacional (SEDENA), es decir, la SEDENA se encarga de administrar el ejército y la fuerza aérea, al mismo tiempo que la Secretaría de Marina (SEMAR) provee el servicio de defensa marítima; desde un punto de vista gráfico se puede ilustrar esta situación por medio de dos componentes, SEDENA y SEMAR, y tres interfaces o servicios, defensa aérea, defensa terrestre y defensa marítima; podemos representar esta relación entre componentes (secretarías) y e interfaces (servicios) como se muestra en la figura 5.4.

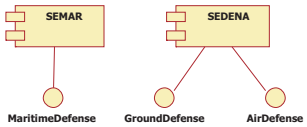


Figura 5.4: Componentes del ejército mexicano.

Después de realizar una inspección detallada (en realidad más detallada que la expuesta aquí) de los casos de uso y sus diagramas de secuencia asociados, podemos ver la necesidad de incluir en el sistema CNC los siguientes componentes, agrupados por paquetes, que se muestran en la figura 5.5 (las interfaces serán tratadas de manera individual en los capítulos siguientes):

Paquete Bibliotecas de tiempo de ejecución Está formado por los componentes que deben diseñarse e implementarse específicamente para una plataforma dada.

Componente Administrador de la aplicación Este componente se encarga de cargar en memoria al resto de los componentes que son esenciales para poder iniciar la ejecución del programa de control.

Componente Administrador de puertos Controla todos los dispositivos de comunicación, tales como teclados, interfaces de red, puertos de comunicación serial, etc., con el propósito de administrar todo el flujo de datos tanto entrante como saliente y ofrecer una interfaz independiente del dispositivo concreto que se esté empleando.

Componente Administrador de la interfaz de usuario Despliega para el usuario de CNC cualquier información que sea hecha pública por el sistema y facilita la interacción con el usuario al permitir también agrupar sucesiones de teclas y caracteres (provenientes del Componente Administrador de comunicaciones) para introducir bloques de información significativa tales como instrucciones de un programa.

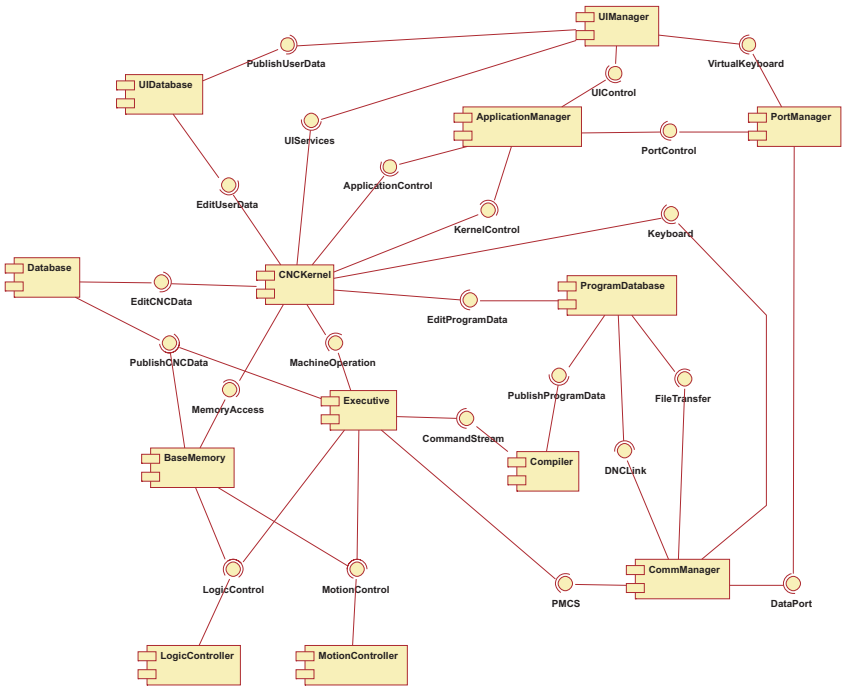


Figura 5.5: Componentes de la arquitectura abierta propuesta.

Paquete Núcleo del CNC Contiene componentes que únicamente tienen sentido en el contexto de un CNC, que son de suma importancia para el sistema y cuyo buen desempeño es crítico para una ejecución eficiente del controlador numérico.

Componente Núcleo del CNC Calendariza todas las actividades del sistema haciendo las veces de mediador.

Componente Base de datos Almacena toda la información relativa a los desplazamientos de trabajo y a las herramientas.

Componente Ejecutivo Se encarga de mantener toda la información sobre el estado del sistema durante la ejecución automática de los programas de parte almacenados por el Componente Base de datos de programas y cuyos comandos son suministrados por el Componente Compilador. También se emplea durante el modo de operación manual, principalmente como intermediario entre el Componente Núcleo del CNC y el Componente Controlador de movimiento. En ambos casos se encarga de la planeación de trayectorias necesaria para la compensación del diámetro de la herramienta, de la longitud de la herramienta y de la traducción de coordenadas de un sistema coordenado de la parte al sistema coordenado de la máquina herramienta.

Componente Memoria base Almacena toda la información necesaria para la operación del CNC, es decir, información tal como la velocidad del husillo, la velocidad de avance y el sistema coordenado actual.

Paquete Comunicaciones En este paquete yacen los componentes relacionados principalmente con protocolos de comunicación que permiten traducir entre un flujo de datos

(como los datos de un programa) y paquetes discretos (tales como mensajes de teclado o datagramas).

Componente Comunicaciones Este componente recibe o envía datos desde o hacia la Interfaz Puerto de datos (del Componente Administrador de puertos) para transferir archivos (véase los casos de uso 3.1.4 y 3.1.3), emplear teclados, realizar control numérico directo (véase el caso de uso 3.1.6) y emplear un sistema de monitorización y control del proceso.

También permite comunicarse con el sistema de monitorización y control de procesos que se esté empleando y enviar el estado del sistema de control (velocidad de avance, revoluciones por minuto del husillo, etc.) que son comparadas con las variables del proceso obtenidas por el sistema de monitorización y control de procesos para entonces recibir comandos de optimización o avisos de fallas.

Paquete Interfaz de usuario Contiene componentes relacionados con la publicación de datos para que el usuario pueda interactuar con el sistema.

Componente Interfaz de usuario Publica información interna del sistema por categorías, es decir, empleando 'diseño lógico' sin preocuparse en absoluto por la forma en que el usuario la verá (no se encarga de desplegar información, ese es trabajo del Componente Administrador de la interfaz de usuario en el Paquete Bibliotecas de tiempo de ejecución). Ofrece al sistema de control interfaces a través de las cuales se pueden actualizar los datos pertenecientes a cada una de las categorías que maneja el componente.

Paquete Compilador de programas de parte Contiene componentes capaces de almacenar la información relativa a los programas de parte y de traducirlos en un flujo de comandos para su ejecución automática.

Componente Base de datos de programas Almacena en memoria permanente la información correspondiente a todos los programas empleados dentro del sistema de CNC. Ofrece al Componente Compilador acceso a los programas por medio de flujos de datos a los mismos para lectura secuencial y diferencia un programa de otro por medio de un identificador único (que puede ser un número, una cadena de caracteres, o alguna otra entidad, según se decida en una implementación particular). Ni siquiera se asume que los programas se almacenen en archivos individuales o que sean archivos de texto.

Componente Compilador Accede a la base de datos de programas para interpretar la información contenida en los mismos y generar instrucciones para el Componente Ejecutivo con el fin de maquinar partes. Solo se asume que el compilador es capaz de comprender el formato en que están almacenados los programas, pero no se asume un lenguaje de programación en particular (ni siquiera se asume que el contenido de los programas sea un lenguaje de programación, podría ser una descripción geométrica simplemente).

Paquete Control de movimiento El control de movimiento involucra todas las partes del programa que tienen que ver con darle instrucciones precisas a los ejes de la máquina en lazo cerrado.

Componente Controlador de movimiento Se encarga de controlar al hardware de control de movimiento, de cualquier tipo que sea. Usualmente depende de bibliotecas ofrecídas por el fabricante (que tienden a emplear la misma interfaz sin importar cuál es el sistema operativo que se emplee).

Paquete Control lógico Contiene a los componentes que se encargan de administrar al hardware del controlador lógico durante la operación de la máquina.

Componente Controlador lógico Se encarga de controlar al hardware de control lógico, de cualquier tipo que sea.

Parte III

Implementación

VI. BIBLIOTECA DE TIEMPO DE EJECUCIÓN

Los componentes de la biblioteca de tiempo de ejecución (junto con los componentes encargados del control de movimiento y del control lógico) son componentes que deben ser reescritos todo el tiempo en buena parte con el propósito de poder 'portar' el sistema de control de una plataforma (combinación de sistema operativo y hardware) a otra. Dos plataformas distintas pueden, por ejemplo, ser el mismo hardware con distinto sistema operativo con modelos de programación realmente distintos entre sí. En este trabajo nos enfocamos en plataformas distintas que corren sobre el hardware de la computadora personal (PC, por sus siglas en inglés), es decir, plataformas que solo difieren en cuanto al sistema operativo. Esta elección por plataformas basadas en la PC no es accidental, sino que se debe al hecho de que la PC es un equipo de cómputo bastante bien estandarizado que permite la reparación de un equipo dañado en cuestión de horas gracias a su arquitectura abierta, lo cual lo hace ideal para un sistema de control de arquitectura abierta.

En el presente trabajo se presenta una versión de la biblioteca de tiempo de ejecución desarrollada en C++ (Deitel y Deitel, 2003) que es apropiada para sistemas operativos que implementan la API Win32 (Petzold y Yao, 1996). Una versión análoga para sistemas DOS o para Linux ofrece un grado de complejidad similar, así que será un buen ejemplo de los detalles técnicos a tratar durante la adaptación del sistema de control a una plataforma en particular.

VI.1. Componente Entorno de tiempo de ejecución

Por entorno de tiempo de ejecución se entiende aquella información necesaria para que una aplicación pueda ejecutarse dentro de un sistema operativo dado. Este componente no forma parte de la lista de componentes dada en el capítulo 5, pero fue incluido para facilitar la implementación de la biblioteca de tiempo de ejecución para Win32 ya que contiene información que es de utilidad para los demás módulos de este paquete.

Toda aplicación que emplea la interfaz gráfica estándar GDI (Graphic Device Interface) proveida por Win32 (GDI es una de varias bibliotecas gráficas disponibles para Win32, pero es la más común y la primera en existir) inicia su ejecución en la función `WinMain`, la cual cuenta con los argumentos

- `hInstance : HINSTANCE`,
- `hPrevInstance : HINSTANCE`,
- `lpszArgument : LPSTR` y
- `nCmdShow : int`;

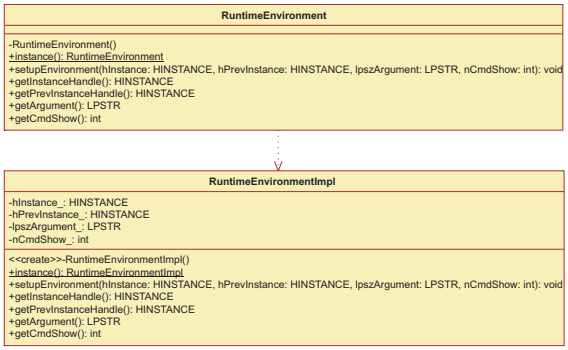


Figura 6.1: Componente Entorno de tiempo de ejecución por medio de una clase entenada.

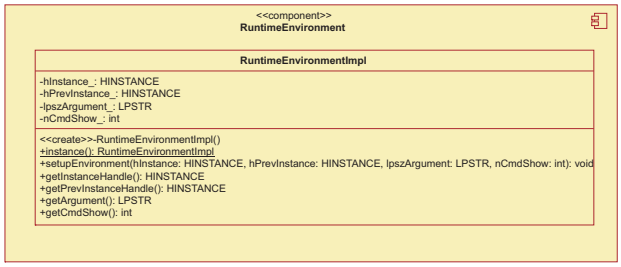


Figura 6.2: Componente Entorno de tiempo de ejecución conceptualmente.

estos argumentos se emplean en otras partes del sistema así que el componente Entorno de tiempo de ejecución (`RuntimeEnvironment`) los almacena para su uso posterior.

Este componente se implementa, al igual que los demás componentes, como un conjunto unitario (*singleton* en inglés), pero no publica ninguna interfaz estándar; véase Gamma et al. (1994) para una explicación detallada del patrón de diseño conjunto unitario. Asimismo, su implementación es por medio de una clase entenada o manejadora (también conocida como clase proxy en spanglish), un modismo (*idiom* en inglés) muy empleado en lenguaje C++ para separar una interfaz y su implementación; véase el primer capítulo de Box (1998) para una explicación detallada de las clases entenadas (se les llama *handler classes* en el libro). En la figura 6.1 se aprecia la implementación del componente Entorno de tiempo de ejecución por medio de una clase entenada siguendo la implementación de Scott Meyers (Alexandrescu, 2001). Sin embargo, si pensamos que la clase entenada representa al componente y que la clase de implementación hace las funciones de mediador (Gamma et al., 1994) entonces podemos conceptualizar al componente `RuntimeEnvironment` como se ilustra en la figura 6.2; esta es la forma en que presentaremos los componentes de ahora en adelante por simplicidad, salvo por el hecho de que además omitiremos los atributos para enfocarnos más en la estructura del software y menos en la implementación particular.

VI.2. Componente Administrador de la aplicación

Este componente se encarga de iniciar la ejecución del sistema de control numérico y de, en caso de ser necesario, terminar también su ejecución. Es el primer componente que se carga en memoria y también es el último en permanecer en ella durante la vida de la aplicación. Carga los demás componentes de la Biblioteca de tiempo de ejecución antes de

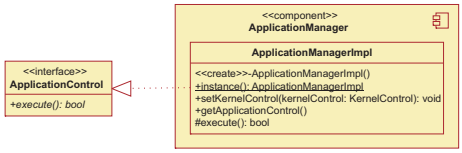


Figura 6.3: Componente Administrador de la aplicación.

empezar a ejecutar al Componente Núcleo del CNC; en realidad, aunque el Administrador de la aplicación entrega el control del sistema al Núcleo del CNC, el mismo componente Administrador de la aplicación es invocado desde el Núcleo de CNC 'intermitentemente' como un proceso de segundo plano. Este componente, sin embargo, permite que el Núcleo del CNC cargue en memoria a los demás componentes que no dependen de la plataforma particular.

Este enfoque difiere levemente del tomado en otras arquitecturas abiertas que responsabilizan de la carga en memoria de todos los componentes a una *única entidad*, al cual se le ha llegado a llamar *administrador de configuraciones* (OSACA, 1998) o inclusive *software núcleo* (Park et al., 2005). Una buena razón para no tomar este enfoque y responsabilizar de la carga de los módulos a dos diferentes 'configuradores' (i.e., Administrador de la aplicación y después Núcleo del CNC) es que precisamente al permitir al Núcleo del CNC cargar solo los módulos que no dependen de la plataforma específica, es más fácil portar el sistema de una plataforma a otra.

Como se menciona en la sección 6.1, las aplicaciones Win32 comienzan a ejecutarse en el punto de entrada `WinMain`; *conceptualmente* esa función forma parte del componente Administrador de la aplicación, quien carga al mismo componente Entorno de tiempo de ejecución, al componente Administrador de la interfaz de usuario de la sección 6.4 y al componente Administrador de puertos de la sección 6.3; puesto que estos componentes siempre están presentes sin importar el tipo de arquitectura de control numérico de la que se trate, técnicamente hablando, el componente Administrador de la aplicación no realiza ninguna tarea de configuración del sistema.

VI.2.1 Interfaz Control de la aplicación

La interfaz control de la aplicación está diseñada para que el Componente Núcleo del CNC pueda ejecutar las tareas de *todos los componentes* de este paquete simultáneamente. Los componentes de este paquete *podrían*[1] ya estar cargados en memoria antes de comenzar a ejecutarse, pero *podrían* estar hibernando en espera de que se les indique que comiencen a operar.

Esta interfaz tiene únicamente una función:

- `execute() : bool`.
  Cada que se llama esta función, se ejecutan las tareas necesarias para mantener funcionando a todos los componentes de la Biblioteca de tiempo de ejecución.

[1]No es obligatorio que lo estén.

VI.3. Componente Administrador de puertos

El Administrador de puertos se encarga, como su nombre lo sugiere, de administrar los puertos de datos (de comunicación) empleados en el sistema de control numérico. Hasta donde ha sido definido por los casos de uso del capítulo 3 (y por las correspondientes colaboraciones del capítulo 4) los puertos de comunicaciones de la computadora se emplean para las siguientes tareas principalmente:

1. Recibir información de los dispositivos de interfaz humana, es decir, de los teclados, principalmente del panel de operación cuya implementación no es estándar en una computadora personal (a diferencia del panel de control que puede ser implementado por medio de un teclado estándar de una computadora personal).

2. Supervisar el proceso de maquinado por medio de un Sistema de Monitorización y Control del Proceso (SMCP, o PMCS por sus siglas en inglés) de maquinado. Estos sistemas permiten detectar fallas de la herramienta empleando usualmente las corrientes de monitorización de los servoamplificadores de los ejes y del inversor del husillo.

3. Administrar archivos por medio de un programa que permita:

   - exportar archivos de programa del sistema de CNC a una computadora externa,
   - inversamente, introducir archivos de programa al sistema de CNC desde la computadora externa, y
   - realizar control numérico directo desde la computadora externa.

Ciertamente existen una gran cantidad de dispositivos físicos posibles para dar soporte a estas funciones. Con el propósito de mostrar la forma en que se puede ignorar la configuración física real de estos dispositivos por medio del Administrador de puertos, el componente de ejemplo hará las siguientes suposiciones:

1. Los dispositivos de interfaz humana serán un teclado estándar de computadora personal (panel de control) y un teclado de propósito especial *ad hoc* (panel de operación) con un protocolo de comunicación por medio de un puerto serial (RS-232) diseñado especialmente para el mismo.

2. El Sistema de Monitorización y Control del Proceso es un sistema de adquisición de datos sin sensores que se comunica con el sistema de CNC por medio del un puerto serial.

3. El programa que se comunica con el sistema de CNC para efectos de importar o exportar programas para su uso o respaldo, llamado de ahora en adelante *administrador externo de archivos*, realiza sus funciones por medio del protocolo TCP/IP.

4. El programa usado para control numérico directo también realiza sus funciones por medio del protocolo TCP/IP y, de hecho, por medio de la misma interfaz de red (e.g., la misma tarjeta de red Ethernet).

Así pues, se asume que el sistema cuenta con (entre otros) los siguientes dispositivos físicos:

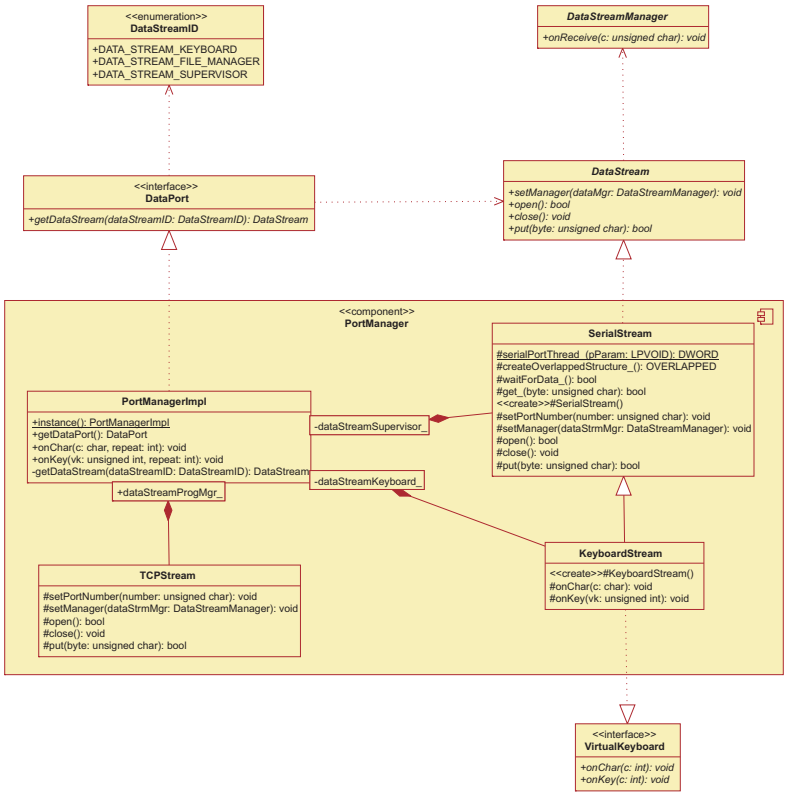


Figura 6.4: Componente Administrador de puertos.

1. Un teclado estándar de computadora personal.

2. Un teclado de propósito especial conectado a algún puerto serial (emplearemos COM1 en este caso particular).

3. Un Sistema de Monitorización y Control del Proceso sin sensores conectado a los servomotores del sistema de CNC y que se comunica por medio de algún puerto serial (emplearemos COM2 en este caso particular).

4. Un dispositivo de interfaz de red que permite utilizar el protocolo TCP/IP.

En la figura 6.4 se puede apreciar el diseño del componente Administrador de puertos, el cual ofrece tres flujos de datos, a saber:

`DATA_STREAM_KEYBOARD` para recibir los bytes correspondientes al flujo del teclado, con cada comando del teclado (véase el capítulo 8) encapsulado en un paquete SLIP; este componente convierte los paquetes SLIP en un flujo continuo de datos para su interpretación por parte del componente Administrador de comunicaciones.

`DATA_STREAM_FILE_MANAGER` para enviar y recibir los bytes correspondientes al Administrador externo de archivos. Los bytes se reciben y se envían como un flujo continuo de datos.

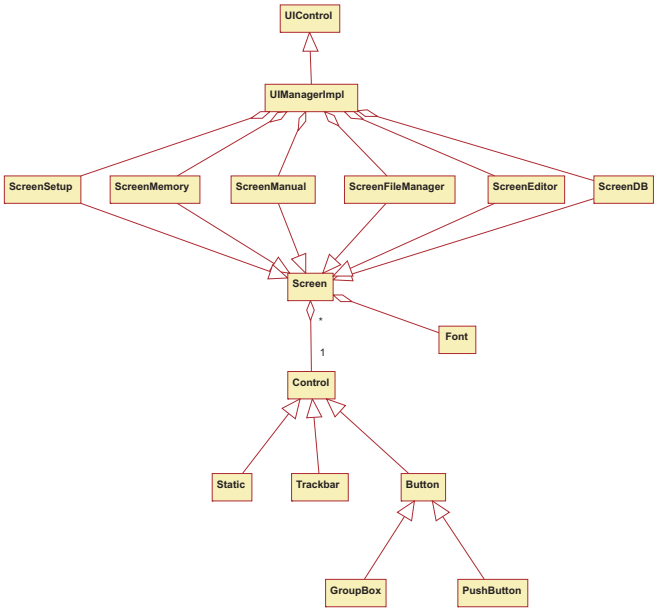


Figura 6.5: Organización del Componente Administrador de la interfaz de usuario.

`DATA_STREAM_SUPERVISOR` para enviar y recibir los bytes correspondientes al Sistema de Monitorización y Control del Proceso (SMCP) que se describe en el apéndice B. La comunicación también se realiza bidireccionalmente empleando el protocolo SLIP.

En todos los casos, estos flujos necesitan ser interpretados por parte del Administrador de comunicaciones ya que la información se recíbe y se envía de forma serializada.

Puesto que asumimos que el puerto serial COM1 será empleado para recibir las acciones solicitadas por medio del panel de operación, se implementó la clase SerialStream de la figura 6.4 para administrar un puerto de comunicaciones serial dado, y a partir de ella se obtiene la clase derivada `KeyboardStream` para integrar tanto la información que proviene del puerto serial COM1 como la que se introduce a través del teclado estándar de la computadora personal; la clase `KeyboardStream` implementa la interfaz `DataStream` que a su vez requiere de la interfaz `DataStreamManager`; esta última interfaz es realizada por cualquier clase que procese la información enviada por un flujo de datos (`DataStream`), así que los flujos de datos no necesitan un (gran) espacio de almacenamiento temporal por su propia cuenta. El puerto serial COM2 también es administrado por un objeto de la clase SerialStream.

VI.4. Componente Administrador de la interfaz de usuario

El componente Administrador de la interfaz de usuario tiene una interfaz bastante simple que permite seleccionar una pantalla en una modalidad dada. Cada pantalla contiene una lista de controles Win32 que permiten desplegar información al usuario, como se muestra en la figura 6.5. Cada uno de los controles está asociado a una entidad almacenada en el componente Interfaz de usuario (véase el capítulo 9).

VII. NÚCLEO DEL CNC

VII.1. Componente Núcleo

El núcleo del sistema (figura 7.1) está formado por un calendarizador Scheduler que almacena en una cola de prioridades CommandQueue a las diferentes solicitudes hechas al mismo en forma de comandos de CNC con formato CNCCommand. El calendarizador tiene una lista de modos de operación ModesList que cambian el comportamiento del mismo según el modo de operación elegido, es decir, se emplea el patrón de diseño estado (Gamma et al., 1994) para permitir la modificación del comportamiento del núcleo de forma eficiente y organizada.

VII.2. Componente Base de datos

Este componente cuenta con dos interfaces, a saber, una para editar su contenido, EditCNCData, y otra para accederlo, PublishCNCData. En esta Base de datos (véase la figura 7.2) se almacena la información de los desplazamientos de trabajo (WorkOffset) y de las herramientas (ToolData).

VII.3. Componente Ejecutivo

Su función principal es realizar la generación de trayectorias a partir de la sucesión de la sucesión de comandos proporcionada por el compilador (véase el capítulo 10). En virtud de que el Controlador de movimiento (capítulo 11) solo realiza trayectorias líneales y circulares básicas sin tomar en cuenta la compensación por diámetro o longitud de la herramienta y solo emplea el sistema coordenado de la máquina sin preocuparse por el sistema coordenado de la parte, el Ejecutivo debe acceder a la información almacenada en el componente Base de datos y calcular trayectorias que sean apropiadas para el componente Controlador de movimiento.

VII.4. Componente Memoria base

La Memoria base, como su nombre lo indica, funciona como área de memoria central para todas las operaciones del CNC, principalmente con las relacionadas con parámetros asociados al Controlador de movimiento y al Controlador lógico. Los datos contenidos en la Memoria base se acceden por medio de una interfaz simple que recibe un identificador de la zona de memoria (de manera parecida a como se emplean apuntadores para una memoria arbitraria) y regresa el valor almacenado en la misma; este mismo identificador se puede emplear para escribir en el registro de Memoria base.

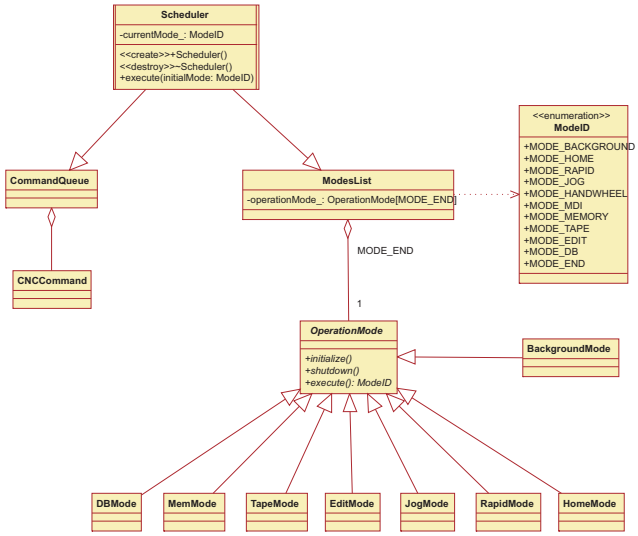


Figura 7.1: Diseño del componente Núcleo.

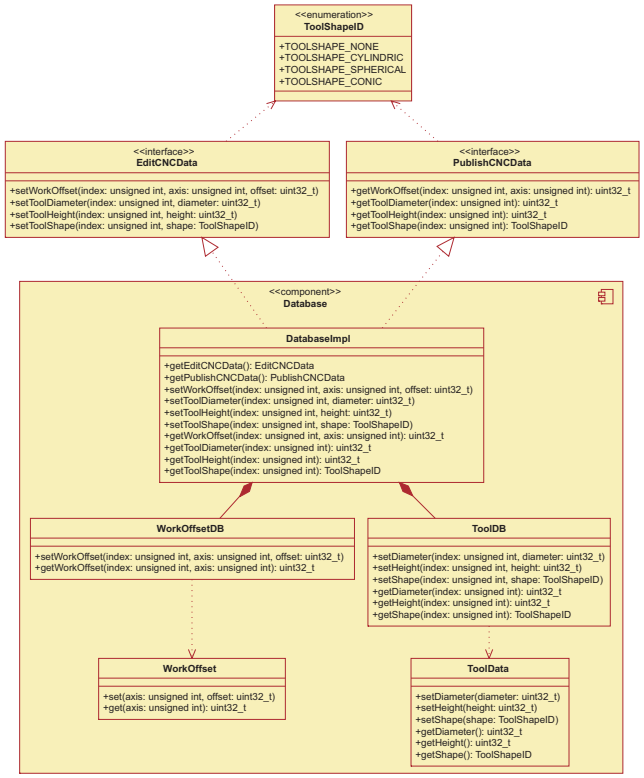


Figura 7.2: Diseño del componente Base de datos.

VIII. COMUNICACIONES

VIII.1. Protocolo de teclados

Como se menciona en la sección 2.2, hay dos teclados distintos con propósitos distintos para permitir la interacción entre el humano y la máquina. Sin embargo, desde un punto de vista de programación, lo más conveniente es integrar ambos teclados en un mismo flujo de datos; este es un buen ejemplo de un dispositivo lógico formado por más de un dispositivo físico.

VIII.1.1 Panel de operación

Después de realizar un estudio detallado de las distintas funciones que usualmente se ofrecen a través de un panel de operación, se definió el protocolo de comunicación del panel de operación hacia el sistema de CNC que se muestra en la tabla 8.1; en este protocolo se emplean dos o tres bytes, a saber, el primero para identificar la *categoría de dispositivo* con el que se realiza la acción y el (los) siguiente (s) para identificar la *acción deseada* que se está comunicando; cuando se envía más de un byte, el byte más significativo se envía primero. Para la transferencia de datos se emplea el protocolo SLIP para poder delimitar cada una de las instrucciones del teclado de manera precisa y que no queden intercaladas las intrucciones unas con otras aún si provienen de flujos distintos de datos; el protocolo SLIP es, sin embargo, manejado por el componente Administrador de puertos (véase la sección 6.3), de tal suerte que los datos son recibidos por el Administrador de comunicaciones como un flujo contínuo de datos.

Categoría de dispositivo	Identificador	Acción deseada	Valor
Carácter	00_{16}	Registrar carácter n	n (2 bytes)
Pulsación de tecla	01_{16}	Registrar carácter n	n
Paro de emergencia	02_{16}	Parar	00_{16}
		Continuar	01_{16}
Ciclo	03_{16}	Iniciar	00_{16}
		Retener	01_{16}
Modo	04_{16}	Elegir memoria	00_{16}
		Elegir IMD	01_{16}
		Elegir archivos	02_{16}
		Elegir editor	03_{16}
		Elegir manivela	04_{16}
		Elegir impulso	05_{16}
		Elegir rápido	06_{16}
		Elegir retorno a cero	07_{16}
Opción	05_{16}	Desactivar bloque a bloque	00_{16}

Categoría de dispositivo	Identificador	Acción deseada	Valor
		Activar bloque a bloque	01_{16}
		Desactivar paro opcional	02_{16}
		Activar paro opcional	03_{16}
		Desactivar omisión de bloque	04_{16}
		Activar omisión de bloque	05_{16}
		Desactivar ejecución en seco	06_{16}
		Activar ejecución en seco	07_{16}
		Desactivar candado MST	08_{16}
		Activar candado MST	09_{16}
		Desactivar candado máquina	$0A_{16}$
		Activar candado máquina	$0B_{16}$
Ajuste de avance	06_{16}	Establecer ajuste en n	n
Avance de impulso	07_{16}	Establecer avance en n	n (2 bytes)
Ajuste de avance rápido	08_{16}	Establecer ajuste en n	n
Eje	09_{16}	Ignorar eje i positivamente	$i0_{16}$
		Elegir eje i positivamente	$i1_{16}$
		Ignorar eje i negativamente	$i2_{16}$
		Elegir eje i negativamente	$i3_{16}$
Husillo	$0\mathrm{A}_{16}$	Rotación normal	00_{16}
		Rotación en reversa	01_{16}
		Iniciar giro	02_{16}
		Detener giro	03_{16}
		Incrementar velocidad	04_{16}
		Decrementar velocidad	05_{16}
Ajuste de husillo	$0\mathrm{B}_{16}$	Establecer ajuste en n	n
Manivela	$0\mathrm{C}_{16}$	Elegir eje X	00_{16}
		Elegir eje Y	01_{16}
		Elegir eje Z	02_{16}
		Elegir factor $\times 1$	03_{16}
		Elegir factor $\times 10$	04_{16}
		Elegir factor $\times 100$	05_{16}
		Incrementar	06_{16}
		Decrementar	07_{16}
Dispositivo i	$0\mathrm{D}_{16}$	Desactivar	$i0_{16}$
		Activar	$i1_{16}$
		Controlar automáticamente	$i2_{16}$

Tabla 8.1: Bytes del protocolo de comunicación del panel de operación.

Este protocolo describe las intenciones del usuario del sistema, no la forma en que solicita las acciones; así pues, se tienen categorías de dispositivos, no dispositivos o controles del panel de operación. Así, en vez de hablar de las acciones presionar y liberar el paro de

emergencia, se habla de las acciones parar y continuar, que es lo que el usuario desea que haga el sistema. Asimismo, se tiene la categoría de dispositivo ciclo (que en este caso no se refiere a un dispositivo, pero podemos ignorar la sutileza) con las acciones continuar e iniciar, cada una de las cuales se realiza usualmente por medio de dos botones distintos. Este protocolo, en consecuencia, permite al diseñador del panel de operación diseñarlo con la distribución que se desee.

VIII.1.2 Panel de control y la interfaz Teclado virtual

El panel de control se implementa por medio de un teclado de computadora personal estándar. En Win32 las aplicaciones reciben la información del teclado de la computadora personal por medio de una cola de mensajes que se envía directamente a la ventana con la que se trabaja (activa), así que es necesario enviar la información relativa a la actividad del usuario con el teclado hacia el administrador de puertos por medio de la interfaz Teclado virtual. Aún en plataformas en las cuales la información relativa al teclado estándar no es enviado a la interfaz gráfica de usuario sino que se puede acceder directamente a ella, la interfaz Teclado virtual puede ser de utilidad para enviar comandos no solo del panel de control sino también del panel de operación hacia el flujo de datos del teclado, de tal manera que, de ser necesario, sería posible emular el panel de operación completo a través de la interfaz gráfica de usuario.

VIII.2. Protocolo del Sistema de Monitorización y Control del Proceso

En el apéndice B de discute a detalle el diseño y la implementación de un sistema de monitorización y control del proceso de maquinado, el cual se comunica con el sistema de CNC por medio del Administrador de comunicaciones. Esta comunicación es llevada a cabo por medio de un enlace del tipo serial, proporcionado por el Administrador de puertos, empleando también el protocolo SLIP para distinguir un comando de otro.

Como se explica en el apéndice B, el Sistema de Monitorización y Control del Proceso (SMCP) reporta al CNC la condición de la herramienta y el ajuste de velocidad de avance recomendado, al mismo tiempo que el CNC debe reportar al SMCP la velocidad programada del husillo y la velocidad de avance programada. Así pues, en la tabla 8.2 se muestra el protocolo de comunicación entre el SMCP y el CNC, formado por un identificador de un byte y un valor de 2 bytes (se envía el byte más significativo primero).

Dato	Identificador	Dirección	Rango
Condición herram.	00_{16}	SMCP a CNC	De 0 a 100
Ajuste de vel. avance	01_{16}	SMCP a CNC	50 % a 150 %
Vel. del husillo	02_{16}	CNC a SMCP	0 a 65535 rpm
Vel. de avance	03_{16}	CNC a SMCP	0 a 65535 mm/min

Tabla 8.2: Bytes del protocolo de comunicación del SMCP.

VIII.3. Protocolo del Administrador externo de archivos

El administrador externo de archivos se comunica con el CNC por medio del protocolo TCP/IP a través del componente Administrador de puertos descrito en la sección 6.3. La interfaz que ofrece este Administrador de puertos es la de un flujo continuo de datos que se

adapta por medio del Administrador de comunicaciones para que se comporte como un flujo de datos estándar para un archivo común y corriente; de esta manera es posible importar, exportar y leer un archivo remoto como si fuera un archivo local.

IX. INTERFAZ DE USUARIO

IX.1. Componente Interfaz de usuario

El componente Interfaz de usuario funciona como una consulta de una base de datos al tomar información de la Memoria base y convertirla en cadenas de caracteres asociadas a un identificador en particular. Estas cadenas de caracteres tienen observadores asociados que pertenecen a los controles del Administrador de la interfaz de usuario (véase la sección 6.4), de tal suerte que cada vez que un valor de la Memoria base se actualiza entonces el Administrador de la interfaz de usuario es capaz de mostrar la modificación correspondiente.

Básicamente, el componente Interfaz de usuario es una colección de listas, llamadas categorías, que holgadamente corresponden a la información requerida en cada una de las pantallas y que están formadas por registros asociados como observadores a los valores contenidos en la Memoria base y que tienen:

1. Un identificador numérico,
2. un conjunto de instrucciones para calcular el valor deseado, y
3. un formato para fabricar una cadena de caracteres.

Esta información, junto con el identificador de la categoria permiten la manipulación sencilla de los registros del componente Interfaz de usuario.

X. LECTOR DE CÓDIGO G

X.1. Términos de programación básica

Como se mencionó en la sección 2.7, el formato de direcciones de palabra es el formato de programación más popular actualmente y se le conoce informalmente como código G; en la sección 3.1 se mencionan los distintos estándares que se han propuesto para el código G.

Hay cuatro términos básicos usados en la programación con código G y son la base para entender la terminología general del CNC:

- Carácter.
- Palabra.
- Bloque.
- Programa.

Cada uno de ellos es muy común e importante en la programación CNC y merece su propia explicación a detalle.

X.1.1 Carácter

Un carácter es la unidad más pequeña de un programa de CNC y puede caer en una de las tres categorías siguientes:

- Dígito
- Letra
- Símbolo

Los caracteres son combinados en palabras significativas. Esta combinación de dígitos, letras y símbolos es llamada *entrada alfa-numérica*.

Dígitos Hay diez dígitos validos para usar en un programa, que son los números del 0 al 9, y sirven para crear números enteros y de punto flotante en un rango predefinido con un determinado número de decimales, dependiendo del controlador particular.

Letras Se pueden usar las 26 letras del alfabeto inglés, al menos en teoría (pues depende del fabricante). La mayoría de los sistemas de control aceptan solo ciertas letras y rechazan otras. Algunos otros no distinguen entre mayúsculas y minúsculas, es decir, para ellos es la misma letra en mayúscula que en minúscula.

Símbolos Existen varios símbolos que se usan en adición a las letras y números. Los más comunes son los símbolos de punto decimal, menos, más, porcentaje, paréntesis y otros dependiendo del controlador particular; en la tabla 10.1 se muestra una lista completa de los símbolos usados en los controladores Fanuc; los corchetes se usan solo con características opcionales tales como las opciones de macros.

Símbolo	Descripción	Comentario
.	Punto decimal	Parte fraccional de un número
+	Signo de más	Valor positivo o signo de suma
-	Signo de menos	Valor negativo o signo de resta
*****	Signo de multiplicación	Signo de multiplicación
/	Signos de división	Salto de bloque o signo de división
( )	Paréntesis	Comentarios del programa o mensajes
%	Signo de porciento	Código de fin de programa
:	Dos puntos	Designación del número de programa
,	Coma	Solo se usa dentro de comentarios
[ ]	Corchetes	Argumentos en macros
;	Punto y coma	Fin de bloque (solo se despliega en la pantalla, no es programable
#	Signo de número	Definición de variable o llamada de macros
=	Signo de igual	Igualdad en macros

Tabla 10.1: Símbolos de los controladores Fanuc.

X.1.2 Palabra

Una palabra es una combinación de caracteres alfa-numéricos con un significado, creando una instrucción simple al sistema de control. Cada palabra empieza con una letra mayúscula (dirección, o registro) seguida de un número que representa un código del programa (como el tipo de movimiento, lineal o circular a realizar) o el valor real (como una coordenada o una velocidad). Las palabras típicas indican la posición de los ejes, velocidad de avance, velocidad de rotación, comandos preparatorios, funciones misceláneas y otras definiciones.

Cada palabra tiene una manera especifica de escribirse. El número de dígitos en la palabra depende de la letra y el número máximo de decimales varía dependiendo del fabricante del controlador, aunque usualmente es 8. Como ya se mencionó, no todas las letras pueden ser usadas, solo las que tienen un significado asignado están disponibles, excepto en los comentarios. Los símbolos pueden ser usados sólo en algunas palabras y su posición dentro de ellas está fija. Las limitaciones del controlador son muy importantes. Los símbolos

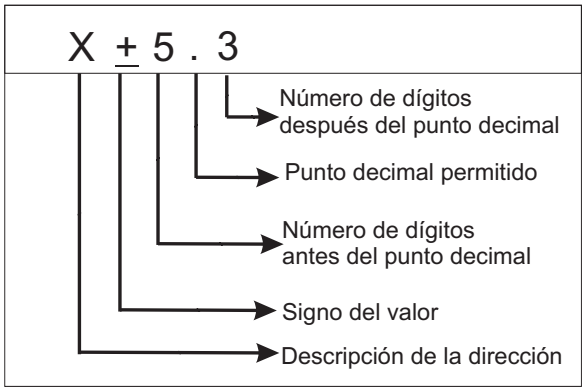


Figura 10.1: Forma abreviada de la notación.

complementan a los dígitos y letras y los proveen con un significado adicional. Los símbolos típicos de programación son el signo de menos, el de porciento y el punto decimal entre otros. Todos los símbolos son listados en una tabla más adelante.

Forma corta

Los fabricantes de controladores a menudo especifican el formato de entrada en forma abreviada como se muestra en la figura 10.1, donde se indica que X es una dirección que permite usar punto decimal, máximo cinco dígitos antes del punto decimal y máximo tres dígitos después de éste y acepta números positivos y negativos.

La ausencia de un punto decimal en la notación indica que el punto decimal no esta permitido (es decir, las notaciones 53 o 44 indican que se aceptan hasta 8 dígitos sin punto decimal), la ausencia de un signo positivo (+) significa que el valor de la dirección no puede ser negativo (es decir, que no se puede especificar un signo ni positivo ni negativo). Los siguientes ejemplos muestran como es la taquigrafía de la notación:

`G2` *Dos dígitos máximo, sin punto decimal ni signo*

`N5` *Cinco dígitos máximo, sin punto decimal ni signo*

`F5` *Cinco dígitos máximo, sin punto decimal ni signo*

`F3.2` *Cinco dígitos máximo tres de ellos antes del punto decimal y dos después, se permite el punto decimal pero no el signo*

Hay que ser cuidadoso cuando leamos un manual ya que las notaciones no son estándares: hay que revisar el manual de cada fabricante. La lista de las direcciones, el formato de su notación y descripción están en las siguientes tablas; están basadas en el típico sistema de control Fanuc.

Signos de más y menos

Algunos de los símbolos más comunes en la programación CNC son los símbolos algebraicos más (+) y menos (−). Cualquier dato en un comando de movimiento puede ser positivo o negativo. Por conveniencia todos los sistemas de control toman como positivo a un número que no tenga signo. Esta característica es llamada algunas veces como sesgo positivo

de los sistemas de control. Es decir, el sesgo positivo es un término que indica como valor positivo a un número que sea programado sin signo en una palabra. Por ejemplo, `X+125.0` es lo mismo que `X125.0`. El signo menos siempre debe ser programado: si el signo menos no está presente, el número se tomará como positivo.

`X-125.0` Valor negativo.

`X125.0` Valor positivo.

`X+125.0` Valor positivo(el signo es ignorado).

Los símbolos complementan a las letras y los dígitos y son una parte integral de la estructura del programa.

Formato del sistema de fresadora

La descripción de las direcciones varía para algunas de ellas dependiendo de si unidades de entrada son inglesas o internacionales.

La tabla 10.2 lista la descripción del formato inglés con el formato métrico entre paréntesis; la primera columna es la dirección, la segunda columna es el formato de la notación y la tercera columna es la descripción:

Dirección	Notación	Descripción
A	A+5.3	Rotando o posicionando ejes —unidades en grados— usado sobre el eje X.
B	B+5.3	Rotando o posicionando ejes —unidades en grados— usado sobre el eje Y.
D	D2	Compensación del radio de corte (algunas veces se utiliza la letra H).
F	F5.3	Función de velocidad de avance —puede variar.
G	G2	Comandos preparatorios.
H	H3	Compensación (posición de la herramienta y/o de la longitud de la herramienta).
I	I+4.4 (I+5.3)	Modificador del centro de arco para el eje X. Cantidad de cambio en ciclos fijos (X). Selección del vector de la esquina para el eje X (controladores viejos).
J	J+4.4 (J+5.3)	Modificador del centro de arco para el eje Y. Cantidad de cambio en ciclos fijos (Y). Selección del vector de la esquina para el eje Y (controladores viejos).
K	K+4.4 (K+5.3)	Modificador del centro de arco para el eje Z.
L	L4	Cuenta la repetición de ciclos fijos o de subprogramas.
M	M2	Funciones misceláneas.
N	N5	Número de bloque o secuencia.
O	O4	Número de programa (EIA) o (:4 para ISO).

Dirección	Notación	Descripción
P	P4	Llamada a subprograma o a macro.
	P3	Desplazamiento de trabajo.
	P8	Pausa en milisegundos.
	P5	Número de bloque cuando se usa M99.
Q	Q4.4 (Q5.3)	Longitud de perforación en ciclos fijos G73 y G83.
	Q+4.4 (Q+5.3)	Cantidad de cambio en ciclos fijos G76 y G87.
R	R+4.4 (R+5.3)	Punto de regreso en ciclos fijos. Designación del radio de arco.
S	S5	Velocidad del husillo en r/min.
T	T4	Función de herramienta.
X	X+4.4 (X+5.3)	Designación del valor del eje coordenado X.
	X5.3	Función de espera con G04.
Y	Y+4.4 (Y+5.3)	Designación del valor del eje coordenado Y.
Z	Z+4.4 (Z+5.3)	Designación del valor del eje coordenado Z.

Tabla 10.2: Notación para la fresadora.

X.1.3 Bloque

Así como la palabra es usada como una instrucción simple al sistema CNC, el bloque es usado como instrucciones multiples. Cada bloque es una colección de instrucciones simples (palabras) que se ejecutan juntas.

En el sistema de control cada bloque debe ser separado de los demás por un código especial llamado *End-Of-Block* (fin de bloque, en español) e indicado comunmente como EOB (por sus siglas en inglés) en el panel de control de la máquina herramienta. Cuando preparamos el programa en la computadora el fin de bloque viene dado por la tecla de *Intro*, y cuando lo hacemos en papel cada bloque debe ocupar una línea del papel. Cada bloque consiste de una serie de instrucciones que son ejecutadas juntas (aunque no necesariamente de manera simultánea, sino respetando un orden, como veremos posteriormente).

X.1.4 Programa

Un programa introducido en el sistema de control consiste de bloques individuales de instrucciones en un orden lógico.

Estructura típica

Comprender la estructura típica de un programa es absolutamente esencial, ya que ésta será usada para definir la sintaxis. El siguente es un programa de ejemplo que nos servirá como punto de inicio para mostrar la estructura típica de un programa.

Ejemplo 10.1 Cada bloque del programa está descrito con un comentario.

```
O0701 (ID MAX 15 CARATERES)
(JOSAFATH OTERO 07-ENE-01)

N1 G20
N2 G17 G40 G80 G49
N3 T01
N4 M06
N5 G90 G54 G00 X.. Y.. S.. M03 T02
N6 G43 Z2.0 H01 M08
(N7 G01 Z.. F..)
(---- MOVS. DE CORTE CON LA HERRAMIENTA T01 ----)
...
N33 G00 G80 Z2.0 M09
N34 G28 Z2.0 M05
N35 M01

N36 T02
N37 M06
N38 G90 G54 G00 X.. Y.. S.. M03 T03
N39 G43 Z2.0 H02 M08
(N40 G01 Z.. F..)
(---- MOVS. DE CORTE CON LA HERRAMIENTA T02 ----)
...
N62 G00 G80 Z2.0 M09
N63 G28 Z2.0 M05
N64  M01

N64 T03
N66 M06
N67 G90 G54 G00 X.. Y.. S.. M03 T01
N68 G43 Z2.0 H03 M08
(N69 G01 Z.. F..)
(---- MOVS. DE CORTE CON LA HERRAMIENTA T03 ----)
...
N86 G00 G80 Z2.0 M09
N87 G28 Z2.0 M05
N88 G28 X.. Y..
N89 M30
%
```

Como ya mencionamos, cada controlador tiene su propia sintaxis, pero podemos rastrear casi todas las variantes existentes como un derivado (o un descendiente) de la sintaxis que empleaban las primeras máquinas basadas en lectores de cinta perforada. En dicho formato de cinta perforada, que se muestra en la figura 10.2, se consideraba que el programa estaba formado por las siguientes secciones:

1. Inicio de la cinta.
2. Descripción.

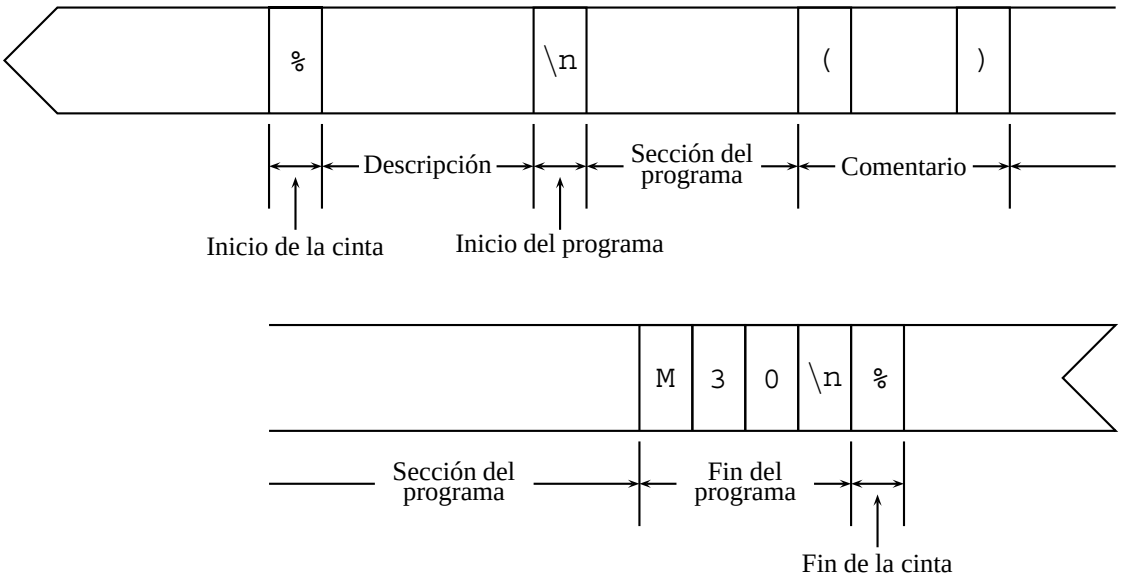


Figura 10.2: Un programa en una cinta perforada para el sistema Fanuc 6M.

3. Inicio del programa.
4. Sección del programa (con algunos comentarios).
5. Fin del programa.
6. Fin de la cinta.

Tanto el inicio de la cinta como el fin de la cinta (ambos `%`) eran símbolos necesarios ya que esa era la única administración de archivos disponible en el sistema. Ahora que la mayoría de los sistemas emplean sistemas de archivos subyacentes, la mayoría de los sistemas de control permiten omitir el inicio de la cinta, el fin de la cinta o ambos en virtud de que no son realmente necesarios ahora (aunque todavía se emplea en algunos casos en transmisión serial para, precisamente, detectar el fin de archivo ya que es común que estas transmisiones ocurran en texto plano). Asimismo, el inicio del programa es simplemente una nueva línea, así que muchos sistemas de control no se molestan en documentarlo; más aún, algunos sistemas de control ni siquiera incluyen a la descripción como parte del programa. La sección del programa incluye al número del programa que, también para algunos sistemas de control, es completamente opcional. Una consecuencia de esto es la aparición de muchos dialectos dentro de los sistemas de control.

Para el sistema de control CONIN se decidió dividir un programa de código G es en los siguientes componentes:

1. Descripción e inicio del programa (nueva línea), es decir, se almacena la descripción en la primera línea del archivo aunque se edite de manera independiente.
2. Número del programa (parte de la sección del programa).
3. Bloques de instrucciones (parte de la sección del programa) y fin del programa.

4. Fin de la cinta (fin del archivo).

X.1.5 Comentarios y encabezado del programa

Los comentarios o mensajes pueden ser puestos en el programa poniéndolos entre paréntesis. Es muy común agregar comentarios al inicio de un programa para proveer de información útil tanto para el programador como para el operador. Esta serie de comentarios en la parte superior del programa es definida como el *encabezado del programa*, donde varias de las características del programa son definidas. El siguiente ejemplo es una muestra (algo exagerada) de los elementos que pueden ir dentro del encabezado del programa:

```
(-------------------------------------------------)
(NOMBRE DEL ARCHIVO.........................01234.NC)
(FECHA DE LA ÚLTIMA VERSIÓN.................07-ENE-01)
(HORA DE LA ÚLTIMA VERSIÓN.....................19:43)
(PROGRAMADOR............................JOSAFATH OTERO)
(MÁQUINA.....................................OKK - VMC)
(CONTROLADOR................................FANUC 15M)
(UNIDADES.....................................PULGADAS)
(NÚMERO DE TRABAJO.............................4321)
(OPERACIÓN.......................PERFORADO-TALADRADO)
(MATERIAL.....................................PLATA)
(MEDIDAS DE MATERIAL.......................8 x 6 x 2)
(CERO DEL PROGRAMA.............X0 - ESQUINA IZQUIERDA)
(                              Z0 - ESQUINA SUPERIOR )
(ESTADO................................NO VERIFICADO)
(-------------------------------------------------)
```

Dentro del programa cada herramienta puede ser definida:

```
(*** T03 - 1/4-20 LLAVE DE TAPÓN***)
```

Otros comentarios y mensajes pueden ser introducidos conforme sean necesarios.

X.2. Comandos preparatorios

La dirección G identifica un comando preparatorio, a menudo llamado el código G. Esta dirección tiene un solo objetivo, que es preseleccionar o preparar el controlador a una cierta condición, cierto modo o estado de operación. Por ejemplo la dirección G00 preselecciona el modo de movimiento rápido para la máquina herramienta, la dirección G81 preselecciona el ciclo de taladrado, etc. El término *comando preparatorio* se refiere a que el código G preparara el controlador para aceptar las instrucciones de programación de una manera específica.

X.2.1 Descripción y propósito

Un ejemplo ilustrará mejor el propósito de los comandos preparatorios; consideremos el siguiente bloque:

```
N7 X13.0 Y10.0
```

Una mirada a este bloque muestra que las coordenadas `X13.0Y10.0` están relacionadas a la posición final de la herramienta de corte cuando el bloque `N7` es ejecutado (es decir procesado por el compilador y luego ejecutado por el controlador de movimiento). El bloque no indica si las coordenadas están en modo absoluto o incremental o si las unidades están en el sistema inglés o el sistema métrico, tampoco se indica la manera en que se hace el movimiento a esta posición final. Si una mirada al bloque no puede establecer el significado del contenido del bloque, tampoco podrá el sistema de control debido a que la información proporcionada por el bloque es incompleta por lo que es inusable por si misma. Entonces se requieren instrucciones adicionales para el bloque. Por ejemplo, para hacer el bloque `N7` una herramienta de movimiento rápido usando dimensiones absolutas, todas estas instrucciones —comandos— deben ser especificados *antes o dentro* del bloque:

Ejemplo 10.2 Se puede configurar las dimensiones absolutas y el movimiento rápido dentro del bloque que contiene las coordenadas destino:

```
N7 G90 G00 X13.0 Y10.0
```

Ejemplo 10.3 O se puede configurar las dimensiones absolutas en un bloque y el movimiento rápido en otro bloque junto con las coordenadas destino:

```
N3 G90
N4 ...
N5 ...
N6 ...
N7 G00 X13.0 Y10.0
```

Ejemplo 10.4 O se pueden configurar las dimensines absolutas y el movimiento rápido en un mismo bloque y darse las coordenadas destino en otro bloque independiente:

```
N3 G90 G00
N4 ...
N5 ...
N6 ...
N7 X13.0 Y10.0
```

Ejemplo 10.5 O cada instrucción en su propio bloque:

```
N3 G90
N4 G00
N5 ...
N6 ...
N7 X13.0 Y10.0
```

Los cuatro ejemplos tendrán el mismo resultado de maquinado si no hay ningún cambio de códigos G en los bloques del `N4` al `N6` en los ejemplos 10.3, 10.4 y 10.5.

Describiremos brevemente los códigos G modales (los que permanecen activos a partir del punto del programa donde se activa) y no modales (los que permanecen activos

únicamente para el bloque del programa al que pertenecen). Cada sistema de control tiene su propia lista de códigos disponibles. Muchos códigos G son muy comunes y pueden ser encontrados en casi todos los sistemas de control mientras que otros son únicos a un sistema de control particular, sea cual sea la máquina herramienta. Debido a la naturaleza de las aplicaciones de maquinado, la lista de los típicos códigos G será diferente para los sistemas de fresado y los sistemas de torneado. Lo mismo aplica para otros tipos de máquinas.

La tabla 10.3 de código G es una lista detallada de los comandos preparatorios más comunes y usados para programar máquinas fresadoras CNC y centros de máquinado CNC. Los códigos G listados pueden no ser aplicables a una máquina o controlador en particular, por lo que se recomienda consultar el manual de la máquina o el controlador para asegurarse de cuáles comandos son aplicables a la máquina o controlador. También hay comandos que son opciones especiales de una máquina o controlador en especial y pueden ser aplicables a otras máquinas o controladores.

Código G	**Description**
G00	Posicionamiento rápido
G01	Interpolación lineal
G02	Interpolación circular en el sentido de las manecillas del reloj
G03	Interpolación circular en el sentido contrario de las manecillas del reloj
G04	Tiempo de espera (como un separador de bloque)
G09	Checar paro exacto - no modal
G10	Entrada de dato programable (ajuste de datos)
G11	Cancelar el modo de ajuste de datos
G15	Cancelar el comando de coordenadas polares
G16	Comando de coordenadas polares
G17	Designación del plano XY
G18	Designación del plano ZX
G19	Designación del plano YZ
G20	Unidades inglesas de entrada
G21	Unidades métricas de entrada
G22	Prohíbe la entrada de la herramienta a un área prohibida
G23	Permite la entrada de la herramienta a un área prohibida
G25	Detección de la fluctuación de la velocidad del husillo encendida
G26	Detección de la fluctuación de la velocidad del husillo apagada
G27	Checar la posición cero de la máquina
G28	Regresar a la posición cero de la máquina (punto de referencia 1)
G29	Regresar del punto cero de la máquina
G30	Regresar a la posición cero de la máquina (punto de referencia 2)
G31	Función de salto
G40	Cancelar la compensación del radio de corte
G41	Compensación del radio de corte —izquierda
G42	Compensación del radio de corte —derecha
G43	Compensación de la longitud de la herramienta —positivo
G44	Compensación de la longitud de la herramienta —negativo

Código G	Description
G45	Compensación de la posición—incremento simple
G46	Compensación de la posición—decremento simple
G47	Compensación de la posición—incremento doble
G48	Compensación de la posición—decremento doble
G49	Cancelar el ajuste de la longitud de la herramienta
G50	Cancelar la función de escala
G51	función de escala
G52	Ajustar el sistema de coordenadas locales
G53	Sistema de coordenadas de la máquina
G54	Ajustar las coordenadas de trabajo 1
G55	Ajustar las coordenadas de trabajo 2
G56	Ajustar las coordenadas de trabajo 3
G57	Ajustar las coordenadas de trabajo 4
G58	Ajustar las coordenadas de trabajo 5
G59	Ajustar las coordenadas de trabajo 6
G60	Posicionamiento simple de la dirección
G61	Modo de paro exacto—modal
G62	Sistema de control en el modo automático en las esquinas
G63	Modo de roscado
G64	Modo de corte
G65	Llamada de macro
G66	Llamada de macro—modal
G67	Cancelar llamada de macro modal
G68	Rotación del sistema de coordenadas
G69	Cancelar la rotación del sistema de coordenadas
G73	Ciclo de picotazo rápido (hoyo profundo)
G74	Ciclo de enroscado a la izquierda
G76	Ciclo de perforado fino
G80	Cancelar ciclo fijo
G81	Ciclo de taladrado
G82	Ciclo de taladrado espontaneo
G83	Ciclo de taladrado profundo
G84	Ciclo de enroscado a la derecha
G85	Ciclo de perforado
G86	Ciclo de perforado
G87	Ciclo de perforado al reves
G88	Ciclo de perforado
G89	Ciclo de perforado
G90	Modo de dimensionamiento absoluto
G91	Modo de dimensionamiento incremetal
G92	Registro de la posición de la herramienta
G98	Regreso al nivel inicial en un ciclo fijo

Código G	Description
G99	Regresar al nivel R en un ciclo fijo

Tabla 10.3: Códigos G más comunes para una fresadora

X.2.2 Códigos G en un bloque de programa

A diferencia de las funciones misceláneas, conocidas como las funciones M y descritas en la siguiente sección, varios comandos preparatorios pueden ser usados en un bloque simple sin estar en error lógico unos de otros:

```
N25 G90 G00 G54 X6.75 Y10.5
```

Este método de programación escrita es varios bloques más corta que la programación de un bloque simple:

```
N25 G90
N26 G00
N27 G54 X6.75 Y10.5
```

Ambos métodos parecerán idénticos durante un proceso continuo. Sin embargo, el segundo ejemplo, cuando sea ejecutado en el modo de ejecución *bloque por bloque*, cada bloque requerirá que se presione la tecla *Iniciar ciclo* una vez para ejecutarlo. El método más corto es más práctico, no solo por la longitud sino por la conexión lógica entre comandos individuales dentro del bloque.

Algunas reglas de aplicación y consideraciones generales se aplican a códigos G usados con otros datos en un bloque. La más importante de ellos es el asunto de la *modalidad*.

Modalidad de comandos G

Anteriormente usamos el ejemplo 10.4 para mostrar cómo se colocan los códigos G dentro de un bloque del programa.

Ejemplo 10.6 (Ejemplo 10.4 extendido) Si cambiamos la estructura ligeramente y llenamos los bloques que tienen puntos por datos reales, estos cinco bloques pueden resultar:

```
N3 G90 G00 X3.0 Y5.0
N4 X0
N5 Y20.0
N6 X15.0 Y22.0
N7 X13.0 Y10.0
```

Nótese que el comando de movimiento rápido `G00` solo aparece una vez en el programa en el bloque `N3`. De hecho el comando para el modo absoluto es `G90`. La razón de por que ni `G00` ni `G90` han sido repetidos es por que ambos comandos permanecen activos desde la primera vez que aparecen en el programa; el termino *modal* es usado para describir esta característica:

> Los *comandos modales* son aquellos comandos que mantienen su función sobre uno o varios bloques; desde el momento que son llamados hasta que son cancelados por una función (otro código G).

Esto significa que el sistema de control permanecerá en un cierto estado hasta que se le indique lo contrario. Como la mayoría de los códigos G son modales no hay necesidad de repetir un comando modal en cada bloque.

Ejemplo 10.7 (Ejemplo 10.6 procesado) Usando el ejemplo 10.6 una vez más, el controlador hará la siguiente interpretación durante la ejecución del programa:

```
N3 G90 G00 X3.0 Y5.0
N4 G90 G00 X0
N5 G90 G00 Y20.0
N6 G90 G00 X15.0 Y22.0
N7 G90 G00 X13.0 Y10.0
```

El programa no tiene ninguna aplicación practica moviéndose de una posición a otra a una velocidad rápida pero demuestra la modalidad del comando preparatorio. El propósito de valores modales es evitar duplicación de modos de programación. Los códigos G son usados tan a menudo que escribirlos en el programa puede ser tedioso. Afortunadamente la mayoría de los códigos G pueden ser aplicados solo una ves aprovechando que son comandos modales. En las especificaciones del sistema de control los comandos preparatorios son identificados como modales y no modales.

Conflicto de comandos en un bloque

El propósito de los comando preparatorios es seleccionar un modo de operación para la máquina. Si el comando `G00` de movimiento rápido es seleccionado, entones como es un comando especifico relacionado al movimiento de la herramienta y es imposible tener un movimiento rápido y un movimiento (de velocidad de) de corte al mismo tiempo, no podría aparecer en el mismo bloque el comando modal `G01` de movimiento lineal (de corte). Es decir, es imposible tener en el mismo bloque `G00` y `G01` activos simultáneamente. Tal combinación crea un conflicto de bloque. Si se programan códigos G conflictivos en un mismo bloque se utilizará el último código que fue programado y se ignorarán los demás. Consideremos el bloque

```
N74 G00 G01 X3.5 Y6.125 F20.0
```

En este ejemplo los dos comandos `G00` y `G01` están en conflicto uno con el otro. Como `G00` es el último de ellos en el bloque, será el que se ejecute. La velocidad de avance de la herramienta (programada por medio de la dirección `F`) será ignorada pues tiene que ver con el código `G01`.

Ahora bien, si intercambiamos el orden de los códigos G obtenemos

```
N74 G01 G00 X3.5 Y6.125 F20.0
```

Este ejemplo es opuesto al anterior. Aquí el código `G00` está al principio por lo que el comando `G01` será ejecutado y el movimiento se tomará como un movimiento de corte a la velocidad especifica de 20 pulgadas por minuto (en sistema inglés).

Orden de las palabras en un bloque

Generalmente los códigos G son programados al comienzo de un bloque después del número de bloque antes de otro dato significativo.

```
N40 G91 G01 Z-0.625 F8.5
```

Este es un orden tradicional basado en la idea de que si el propósito es el de preparar o preseleccionar el sistema de control a una cierta condición, los comandos preparatorios deberían ser puestos siempre en primer lugar. Debido a ésto, solo se permiten códigos que no producen conflictos unos con otros en el mismo bloque. Estrictamente hablando no tienen nada de malo rearreglar el bloque en el orden:

```
N40 G91 Z-0.625 F8.5 G01
```

Quizá inusual pero totalmente correcto. Hay situaciones, sin embargo, en las que uno debe tener cuidado con el orden de los comandos preparatorios de un bloque. Este es el caso con los códigos G del bloque

```
N40 Z-0.625 F8.5 G01 G91
```

El movimiento de corte `G01`, la velocidad de avance `F` y la profundidad `Z` serán ejecutadas usando el modo de dimensionamiento actualmente activo. Si el modo de dimensión actual es absoluto (`G90`), el movimiento en el eje Z será ejecutado como un valor absoluto, no como un valor incremental (`G91`). La razón para esta excepción es que Fanuc y muchos otros controladores permiten combinar los valores dimensionales en el mismo bloque. Esta puede ser una característica muy útil si se utiliza con cuidado. Una típica aplicación *correcta* se ilustra en el siguiente ejemplo:

```
(G20)
N45 G90 G00 G54 X1.0 Y1.0 S1500 M03 (G90)
N46 G43 Z0.1 H02
N47 G01 Z-0.25 F5.0
N48 X2.5 G91 Y1.5                    (G90 combinado con G91)
N49 ...
...
```

Los bloques del `N45` al `N47` están en modo absoluto. Antes de que el bloque `N48` sea ejecutado la posición absoluta de los ejes X y Y es $(1.0, 1.0)$. De este punto de inicio la locación objetivo es la posición absoluta de $X = 2.5$ combinado con el movimiento incremental de 1.5 pulgadas a través del eje Y. La posición absoluta resultante es $(X, Y) = (2.5, 2.5)$, haciendo un movimiento de 45°. En este caso, el comando `G91` permanecerá en efecto para todos los bloques subsecuentes hasta que `G90` sea programado. La mayoría probablemente escribiría el bloque `N48` en modo absoluto:

```
...
N48 X2.5 Y2.5
...
```

Normalmente o hay razón para cambiar entre los dos modos. Esto puede resultar una sorpresa muy desagradable para algunos. Habrá algunas ocasiones cuando esta técnica especial brinde buenos beneficios, por ejemplo, en subprogramas.

X.2.3 Grupos de comandos

El ejemplo de códigos G en conflicto en un bloque nos da una buena razón para agrupar los comandos preparatorios en colecciones que nos permitan identificar los posibles conflictos. Tiene sentido, por ejemplo, que los comandos de movimiento tales como `G00`, `G01`, `G02` y `G03` no puedan coexistir en el mismo bloque. Cada grupo, llamado el *grupo de código G*, tiene asignado un número de dos dígitos arbitrario. En un mismo bloque puede haber cualquier número de códigos G siempre y cuando no pertenezcan al mismo grupo, es decir, dos comandos preparatorios programados en un mismo bloque *están en conflicto* uno del otro si pertenecen al mismo grupo G.

Números de los grupos

Los números de los grupos son numerados desde el 00 hasta el 25. Este rango varia entre los diferentes modelos de controlador dependiendo de las características. Incluso pueden haber grupos para los controladores más nuevos donde se requieren más códigos G. El más extraño y quizá el más importante también es el *grupo 00*. Todos los comandos preparatorios en el grupo 00 son *no modales*. Están solo activos en el bloque donde son programados y si el comando ha de estar activo en más de un bloque, este debe ser programado en todos aquellos bloques donde deba de tener efecto; en la mayoría de los comandos no modales esta repetición no será usada muy a menudo.

Por ejemplo, un retardo (`G04`) es una pausa del programa medida en milisegundos y se necesita solo para la duración dentro del tiempo especifico y no más. No hay una necesidad lógica para programar retardo en dos o más bloques consecutivos. Después de todo, ¿cual es el beneficio de los siguientes tres bloques de instrucciones?

```
N56 G04 P2000
N57 G04 P3000
N58 G04 P1000
```

Los tres bloques contienen un retardo uno después del otro. El programa puede ser mucho más eficiente si simplemente metemos el valor total del retardo en un simple bloque:

```
N56 G04 P6000
```

Los siguientes grupos son típicos de los controladores Fanuc. La aplicabilidad de los códigos G para fresadora o torno son especificadas por las letras F o T, respectivamente, en la columna **Tipo** de la tabla:

Grupo	**Descripción**	**Códigos G**				**Tipo**
00	Códigos G no modales	G04	G09	G10		F/T
		G11	G27	G28	G29	F/T
		G30	G31	G37		F/T
		G45	G46	G47	G48	F/T
		G52	G53	G65		F/T
		G51	G60	G92		F/T
		G50				T
		G70	G71	G72	G73	T
		G74	G75	G76		T

Grupo	Descripción	Códigos G	Tipo
01	Comandos de movimiento y ciclos de corte	G00 G01 G02 G03 G32 G35 G36 G90 G92 G94	F/T T T
02	Selección del plano	G17 G18 G19	F
03	Modo de dimensionamiento	G90 G91 (U y W para tornos)	F
04	Store strokes	G22 G23	F/T
05	Velocidad de avance	G93 G94 G95	T
06	Unidades de entrada	G20 G21	F/T
07	Ajuste del radio de corte	G40 G41 G42	F/T
08	Ajuste de la longitud de la herramienta	G43 G44 G49	F
09	Ciclos	G73 G74 G76 G80 G81 G82 G83 G84 G85 G86 G87 G88 G89	F F F F
10	Modo de regreso	G98 G99	F
11	Reflejos y escalas	G50 G68 G69	F T
12	Sistemas de coordenadas	G54 G55 G56 G57 G58 G59	F/T F/T
13	Modos de corte	G61 G62 G64 G63	F/T F
14	Modo macro	G66 G67	F/T
16	Rotación de coordenadas	G68 G69	F
17	Velocidad superficial constante	G96 G97	T
18	Entrada polar	G15 G16	F
24	Fluctuación de la velocidad del husillo	G25 G26	F/T

Tabla 10.4: Grupos más comunes en controladores FANUC

La relación de grupo tiene sentido en todos los casos. Una posible excepción es el grupo de *comandos de movimiento*, grupo 01, y el grupo de *ciclos enlatados*, grupo 09. La relación entre estos dos grupos es que si un código G del grupo 01 está en un ciclo enlatado, el ciclo enlatado es cancelado, pero el inverso no es cierto, es decir, un comando de movimiento activo no es cancelado por un ciclo fijo: el grupo 01 no es afectado por comandos del grupo 09.

En resumen, cualquier código G de un grupo dado reemplaza automáticamente otro código G del mismo grupo.

X.2.4 Tipos de código G

El sistema de control Fanuc ofrece una selección flexible de comandos preparatorios. Este hecho distingue a Fanuc de otros controladores. Considerando el hecho de que los controladores Fanuc se usan alrededor del mundo, la flexibilidad de comandos tiene sentido para permitir la configuración estándar del controlador y seguir los estilos establecidos

de cada pais. Un ejemplo típico es la selección de unidades de dimensión, pues en muchos países el sistema estándar es el sistema métrico; sin embargo en el norte de América el sistema usado es el sistema inglés. Ambos mercados son sustanciales en el comercio mundial y un fabricante habilidoso de sistemas de control trata de abarcar a los dos. Casi todos los fabricantes de controladores ofrecen una selección de los sistemas de dimensión.

El método de los controladores Fanuc usa un simple método de ajuste de parámetros. Seleccionando el parámetro específico del sistema, uno, dos o tres colecciones de códigos G pueden ser seleccionados, algo que es típico para un usuario geográfico particular. Aunque la mayoría de los códigos G son los mismos para cada tipo, el ejemplo más típico es el uso de la selección de unidades del sistema inglés o métrico. Muchos controladores viejos estadounidenses usaban G70 para unidades inglesas y G71 para unidades métricas. Los controladores Fanuc han usado tradicionalmente los comandos G20 y G21 para la selección de unidades, ya sean inglesas o métricas respectivamente. Todos los códigos G en este documento usan el grupo del tipo A por omisión,pues es el grupo más común.

Códigos G y punto decimal

Muchos controladores Fanuc más recientes incluyen códigos G con punto decimal, por ejemplo, `G72.1` (copia de rotación) o `G72.2` (copia paralela). Varios comandos en este grupo son relacionados a una máquina herramienta en particular o no son lo suficientemente generales para ser tratados en este documento.

X.3. Funciones misceláneas

Las direcciones M en un programa CNC identifican a una *función miscelánea*, algunas veces llamadas *funciones de la máquina*. No todas la funciones misceláneas están relacionadas a la operación de una máquina CNC pues varias de ellas están relacionadas al procesamiento del programa mismo.

X.3.1 Descripción y propósito

Dentro de la estructura de un programa CNC, los programadores necesitan a menudo algunos medios de activar ciertos aspectos de la operación de la máquina o controlar el flujo del programa. Sin la disponibilidad de dichos recursos, el programa estaría incompleto e imposible de correr. Primero vamos a ver las funciones misceláneas relacionadas al funcionamiento de la máquina.

Funciones relacionadas a la máquina

Varias operaciones físicas de las máquinas CNC deben ser controladas por el programa para asegurar el maquinado totalmente automatizado. Estas funciones generalmente usan la dirección M e incluyen las siguientes operaciones:

- Rotación del husillo (dextrógiro o levógiro).
- Cambiar rango de engrane (lento, medio, rápido).
- Cambio automático de la herramienta (si es que hay intercambiador automático de herramientas).
- Cambio automático de la paleta (cuando hay intercambiador automático de paletas).

- Operación de refrigerante (encendido o apagado).
- Movimiento de la contrapunta (hacia adentro o hacia afuera).

Estas operaciones varían debido a los diferentes diseños de los fabricantes de máquinas. Un diseño de máquina, desde el punto de vista de la ingeniería, está basado en ciertas aplicaciones primarias de maquinado. Una máquina de fresado CNC requiere diferentes funciones relacionadas a la máquina que un centro de maquinado CNC o un torno CNC. Una máquina de corte de alambre EDM controlada numéricamente tendrá muchas funciones únicas, típicamente relacionadas a ese tipo de maquinado y no son encontradas en otra máquina.

Inclusive dos máquinas diseñadas para el mismo tipo de trabajo, por ejemplo, dos centros de maquinado verticales tendrán funciones diferentes entre sí cuando tengan diferentes sistemas CNC u opciones significativamente diferentes. Diferentes modelos de máquinas del mismo fabricante también tendrían ciertas funciones únicas incluso con el mismo sistema CNC.

Todas las máquinas diseñadas para remover metal tienen ciertas características y capacidades en común; por ejemplo, la rotación del husillo puede tener tres y solo tres selecciones de posición en un programa:

- Rotación normal del husillo.
- Rotación inversa del husillo.
- Paro del husillo.

Además de estas tres posibilidades, hay una función llamada la orientación del husillo que es también una función relacionada a la máquina. Otro ejemplo es el refrigerante que puede empezar encendido o apagado.

Estas operaciones son típicas para la mayoría de las máquinas CNC. todas son programadas con una función M seguida por no más de dos dígitos, aunque algunos modelos de controladores permiten funciones M con tres dígitos, el controlador Fanuc 16/18, por ejemplo.

Fanuc también usa funciones M con tres dígitos en varias aplicaciones especiales, por ejemplo, la sincronización de dos torretas independientes sobre un torno de cuatro ejes. Todas estas y algunas otras funciones M son relacionadas a la operación de la máquina pertenecen al grupo conocido como *funciones misceláneas* o simplemente *funciones M* o *códigos M*.

Funciones relacionadas al programa

Además de las funciones de máquina, algunas funciones M son usadas para controlar la ejecución de un programa CNC. Una interrupción del programa requiere una función M, por ejemplo, durante el cambio de equipo de trabajo como la inversión de una parte. Otro ejemplo es una situación donde un programa llama uno o más subprogramas. En tal caso cada programa debe de tener una llamada de función, el número de repeticiones, etc. Las funciones M manejan estos requisitos.

Basados en los ejemplos previos, el uso de las funciones misceláneas cae en dos principales grupos, basados en una aplicación en particular:

- Control de la funciones máquinas.
- Control de la ejecución del programa.

Este documento solo se tratan las funciones misceláneas usadas por la mayoría de los controladores. Desafortunadamente, hay varias funciones que varían entre las máquinas y los sistemas de control. Estas funciones son llamadas *funciones especificas de la máquina*. Por esta razón, siempre hay que consultar la documentación para el modelo de máquina en particular y su sistema de control.

X.3.2 Aplicaciones típicas

Antes de estudiar las funciones M vamos a ver que tipo de actividad hacen estas funciones, independientemente de si tales actividades están relacionadas al funcionamiento de la máquina o del programa. También nótese la abundancia de dos modos de interruptor de manera semejante a encendido y apagado, dentro y fuera, adelante y atrás, etc. Siempre se debe checar primero el manual de la máquina por razones de consistencia; todas las funciones M de este documento están basadas en la tabla 10.5.

Código M	**Descripción**
M00	Paro obligatorio del programa
M01	Paro opcional del programa
M02	Fin del programa (con reinicio, sin rebobinar)
M03	Rotación normal del husillo
M04	Rotación inversa del husillo
M05	Paro del husillo
M06	Cambio automático de la herramienta (cuando hay intercambiador automático)
M07	Refrigerante encendido (brisa)
M08	Refrigerante encendido (empapar herramienta)
M09	Refrigerante apagado
M19	Orientación del husillo
M30	Fin de programa (con reinicio y rebobinado)
M48	Desactivar desaceleración de la velocidad de avance
M49	Activar desaceleración de la velocidad de avance
M60	Cambio automático de la paleta (cuando hay intercambiador automático de paleta)
M78	Abrazadera del eje B (no estándar)
M79	Eje B sin abrazadera (no estándar)
M98	Llamada a subprograma
M99	Fin de subprograma

Tabla 10.5: Funciones M más comunes en fresadoras FANUC.

Funciones especiales de MDI

Varias funciones M no pueden ser usadas en el programa CNC en lo absoluto. Este grupo es usado en el modo exclusivo de *entrada manual de datos* (MDI por sus siglas en inglés). Un ejemplo de tales funciones es el cambio de herramienta paso a paso para centros de maquinado, usada solo para propósitos de servicio mas nunca en el programa. Estas funciones están fuera del alcance de este documento.

Grupos de aplicación

Las dos categorías más grandes mencionadas anteriormente, serán divididas en varios grupos, basándonos en aplicaciones especificas de las funciones misceláneas dentro de cada grupo. Una lista de la distribución típica esta contenida en la siguiente tabla:

Grupo	**Funciones M típicas**
Programa	M00 M01 M02 M30
Husillo	M03 M04 M05 M19
Cambio de herramienta	M06
Refrigerante	M07 M08 M09
Accesorio	M10 M11 M12 M13 M17 M18 M21 M22 M78 M79
Enroscado	M23 M24
Rango de engranes	M41 M42 M43 M44
Desaceleración de la velocidad de avance	M48 M49
Subprogramas	M98 M99
Paletas	M60

Tabla 10.6: Grupos más comunes de las funciones M

La tabla no cubre todas las funciones M, ni siquiera todos los posibles grupos. Tampoco distingue entre máquinas. Por otro lado la tabla nos indica el tipo de aplicaciones de las funciones misceláneas que son usadas cada día en la programación CNC.

Las funciones misceláneas mencionadas en esta sección son usadas a través del documento y algunas de ellas aparecen más que otras reflejando su uso general en la programación. Las funciones que no corresponden a un sistema de control de una máquina en particular no son usados o no se necesitan. Sin embargo, la forma en que se usan estas funciones M son siempre similares en la mayoría de los sistemas de control y máquinas CNC. En esta sección solo se cubren con detalle las funciones más generales.

X.3.3 Funciones M en un bloque

A diferencia de los comandos preparatorios, las funciones M pueden ser programadas en un bloque sin algún otro dato que las complemente y serán ejecutados sin problemas,

por ejemplo

```
N45 M01
```

es un paro opcional. Este bloque es correcto, es decir, la única entrada de un bloque puede ser una función M. Contrario a los códigos G, solo una función M puede ser programada en un bloque a menos que el sistema de control permita múltiples funciones M en el mismo bloque; de no ser así ocurrirá un error del programa.

Un método práctico de programar ciertas funciones M es en un bloque que contiene una *herramienta de movimiento*. Por ejemplo encender el refrigerante y al mismo tiempo mover la herramienta de corte a una cierta parte que pueda ser requerida. Como allí no hay conflicto entre las instrucciones el bloque puede ser como

```
N56 G00 X12.9854 Y9.474 M08
```

En este ejemplo, en el bloque `N56`, el tiempo preciso que la función `M08` se activara no es importante. Aunque en otros casos el tiempo sí será relevante. Algunas funciones M deben estar en efecto antes o después de tomar ciertas acciones; por ejemplo, si aplicamos un movimiento en el eje Z combinado con la función M00 de *paro de programa* en el mismo bloque:

```
N319 G01 Z-12.8456 F20.0 M00
```

Esta es una situación más seria y se necesitan dos respuestas. La primera es qué realmente esta pasando y la otra es cuándo está pasando exactamente, es decir, cuándo es que la función M00 es activada. Hay tres posibilidades y tres preguntas por hacer:

1. ¿El paro del programa tomará lugar inmediatamente, cuando el movimiento apenas comience activado, en el principio del bloque?

2. ¿El paro de programa tomará lugar mientras se esta realizando el movimiento?

3. ¿El programa de paro tomará lugar cuando el comando de movimiento halla sido completado, en el final del bloque?

Sabemos que una de estas tres cosas pasará, ¿pero cuál? Cada función M esta diseñada de manera lógica y diseñada para tener *sentido común*. Las funciones M se dividen en dos grupos según el momento en que se activan:

- Las funciones M que se activan al principio del bloque (de manera simultanea al movimiento de la herramienta de corte).

- Las funciones M que se activan al final del bloque (cuando el movimiento de la herramienta ha sido completado)

Las funciones M nunca serán activadas durante la ejecución de un bloque pues no hay lógica en esto. ¿Cuál es el principio lógico de la función `M08`, encendido de refrigerante, en el bloque `N56` del ejemplo anterior? La respuesta es que el refrigerante debe de ser activado al mismo tiempo que empieza el movimiento de la herramienta. La respuesta correcta para el ejemplo anterior del bloque `N319` es que el paro de programa, función `M00`, debe ser activada después de que el movimiento de la herramienta ha sido completado. Esto tiene sentido, sin embargo existen muchas otras funciones, y cada una tiene su propia lógica.

Activación de las Funciones M

Miremos la tabla 10.5 de las funciones M; a cada una de ellas añadámosle un movimiento de herramienta y tratemos de determinar la manera en que ellas se comportarían basados en los ejemplos anteriores. Después de un breve análisis, podremos llegar a la conclusión correcta de las funciones M se activarían según las tablas 10.7 y 10.8.

M03	Rotación normal del husillo
M04	Rotación inversa del husillo
M06	Cambio automático de herramienta (intercambiador automático de herramienta)
M07	Refrigerante encendido (brisa)
M08	Refrigerante encendida (flujo abundante)

Tabla 10.7: Funciones M activadas al principio de un bloque.

M00	Paro de programa obligatorio
M01	Paro de programa opcional
M02	Fin de programa (con reinicio, sin rebobinar)
M05	Paro del husillo
M09	Refrigerante apagado
M30	Fin de programa (con reinicio y rebobinado)
M60	Cambio automático de la paleta (intercambiador automático de paleta)

Tabla 10.8: Funciones M activadas al final de un bloque.

Si llega a haber alguna incertidumbre de cómo interactuarán las funciones con algún comando de movimiento, lo más seguro es programarlas en un bloque separado. De esta manera la función siempre será procesada antes o después del bloque relevante. En la mayoría de las aplicaciones ésto es una solución segura.

Duración de las funciones M

El conocimiento de *cuándo* entran efecto las funciones M es seguido lógicamente por la pregunta de durante *cuánto* tiempo se activara la función. Algunas funciones misceláneas están activas solo en el bloque en el que aparecen mientras otras continuarán activas hasta ser canceladas por otra función miscelánea. Esto es similar a lo que pasa con los comandos preparatorios G; sin embargo, no se usa la palabra modal con las funciones M. Como ejemplo de la duración de una función tomemos la función M00 o M01, las cuales están activas solo en el bloque que son programadas. La función de refrigerante encendido, M08, estará activa hasta que sea cancelada o alterada por otra función como M00, M01, M02, M09 o M30. Comparemos las dos tablas 10.9 y 10.10.

M00	Paro de programa obligatorio
M01	Paro de programa opcional

M02	Fin de programa (con reset, sin rebobinar)
M06	Cambio automático de herramienta (ATC)
M30	Fin de programa (con reset y rebobinado)
M60	Cambio automático de la paleta (APC)

Tabla 10.9: Funciones M completadas en un bloque.

M03	Rotación normal del husillo
M04	Rotación inversa del husillo
M05	Paro del husillo
M07	Refrigerante encendido (brisa)
M08	Refrigerante encendida (flujo abundante)
M09	Refrigerante apagado

Tabla 10.10: Funciones M activadas hasta ser canceladas.

La clasificación es lógica y muestra sentido común. No hay necesidad de recordar funciones individuales o la actividad exacta que realizan. El mejor lugar para averiguar sobre las funciones M es el manual que viene con la máquina CNC.

X.3.4 Funciones de programa

Las funciones que controlan el procesamiento del programa pueden ser usadas para interrumpir el proceso de manera temporal (en medio del programa) o permanente (al final del programa). Existen varias funciones que pueden hacer esto.

Paro de programa

La función `M00` es definida como un paro de programa *incondicional* u *obligatorio*. En cualquier momento que el sistema de control encuentre esta función durante el procesamiento del programa todas las operaciones automáticas de la máquina se detendrán:

- Movimiento de todos los ejes.
- Rotación del husillo.
- Función refrigerante.
- Ejecución de programas.

El controlador no será reiniciado cuando la función `M00` sea procesada. Todos los datos significantes del programa activos en ese momento se mantendrán, tales como velocidad de avance, velocidad del husillo, coordenadas, etc. El programa podrá ser reactivado solo si se vuelve a presionar la tecla de inicio de ciclo. La función `M00` cancela la rotación del husillo y la función del refrigerante, por lo que estas tendrán que ser reprogramadas en bloques siguientes.

La función `M00` puede ser programada en un bloque individual o en un bloque que contiene otros comandos, usualmente movimiento de ejes. Si la función M00 es programada

junto con un comando de movimiento, el movimiento será completado primero y después el paro del programa sera ejecutado:

Ejemplo 10.8 Un `M00` programado después de un comando de movimiento:

```
N38 G00 X13.5682
N39 M00
```

Ejemplo 10.9 Un `M00` programado con un comando de movimiento:

```
N38 G00 X13.5682 M00
```

En ambos casos el comando de movimiento será completado antes que el paro de programa sea ejecutado. La diferencia entre los ejemplos 10.8 y 10.9 es clara durante una ejecución bloque por bloque (por ejemplo, durante un maquinado de prueba). No hay una diferencia práctica con el modo de procesamiento automático.

La función de paro de programa usada en un programa hace el trabajo del operador de CNC mucho más fácil. Es muy útil para muchos trabajos. Un uso muy común es la inspección de la parte sobre la máquina, mientras la parte esta siendo procesada; durante el paro, la parte o las condiciones de la herramienta pueden ser checadas o se pueden remover la viruta del maquinado que queda en un hoyo taladrado antes de que otra operación de maquinado se lleve a cabo. La función de paro de programa también es necesaria para cambiar los ajustes actuales en medio del programa, por ejemplo, para invertir una parte o para cambiar automáticamente una herramienta. En resumen podemos decir que la función de paro de programa, `M00`, es para la intervención manual durante el proceso del programa.

El controlador también ofrece un paro de programa *opcional*, la función `M01`. La regla principal de usar `M00` es la necesidad de *intervención manual* para *cada* parte maquinada. El cambio manual de la herramienta califica para el uso de `M00` por que *cada* parte lo necesita. Sin embargo un chequeo de dimensiones puede no calificar ya que no es frecuente y `M01` será una mejor opción. Aunque la diferencia entre las dos es poca; se puede ver en el tiempo de maquinado de varias piezas. Supongamos que tenemos que maquinar varias partes, si nos detenemos a checar las dimensiones de cada parte (utilizando la función `M00`) cuando esta es finalizada tardaríamos mucho tiempo en el maquinado de muchas piezas. En cambio si solo nos detenemos solo cuando sea necesario (usando la función `M01`) esto nos ahorraría tiempo de trabajo.

Al usar la función `M00`, siempre se debe informa al operador por qué ha sido usada y su propósito. Se puede hacer de dos maneras:

- En la hoja de ajustes, hacer referencia la número de bloque que contiene a la función miscelánea M00 y describir la operación manual que ha sido realizada:

  BLOQUE N39: REMOVER VIRUTA.

- En el programa mismo, hacer un comentario con la información necesaria. El comentario debe estar encerrado entre paréntesis, ejemplos:

```
N39 M00 (REMOVER VIRUTA)

N39 X13.5682 M00 (REMOVER VIRUTA)

N39 X13.5682 M00
(REMOVER VIRUTA)
```

Paro de programa opcional

La función `M01` es un paro de programa opcional o condicional. Es similar a la función `M00` con una diferencia: al contrario de la función `M00`, cuando la función `M01` es encontrada en el programa, el proceso del programa no se parará a menos que el operador interfiera por medio del panel de control. El *paro de control*, ya sea palanca o botón en el panel de control, puede ser puesto en la posición de encendido o apagado. Cuando la función `M01` sea procesada, la posición del interruptor determinará si el programa se para o continua según:

Switch del paro opcional	**Resultados de M01**
Encendido	El proceso se detendrá
Apagado	El proceso no se detendrá

En caso de que la función `M01` no haya sido programada, el switch de paro opcional es irrelevante. Normalmente el switch debe estar en la posición de Apagado para la producción de piezas. Cuando el switch este activado, es decir en Encendido, la función `M01` se comportará como la función `M00`.

Una buena idea es programar la función `M01` al final de cada herramienta seguida de un espacio en blanco sin datos. Si el proceso del programa continua significa que el switch de paro opcional está deshabilitado, en Apagado, y no se pierde tiempo. Aunque si hay necesidad durante el proceso de maquinado solo hay que cambiar el interruptor a Encendido. Generalmente el tiempo invertido es justificado, por ejemplo, el cambio de la herramienta de corte, la inspección de dimensiones o checar la superficie de la parte trabajada.

Fin del programa

Cada programa debe incluir una función especial que defina el *fin* del programa actual. Por eso hay dos funciones M disponibles, la función `M02` y la función `M30`. Ambas son muy parecidas pero con propósitos distintos. La función `M02` termina el programa y no causará un regreso al principio del programa, es decir, no regresará al primer bloque programado. Y la función `M30` también terminará el programa pero está regresara al primer bloque programado. Es como la función de rebobinar en la grabadoras. Esto se quedo de los tiempos en los que las máquinas CN tenían lectoras de cintas y había que rebobinar las cintas que tenían el programa de maquinado de una parte. La función `M30` proporciona esta opción de “rebobinado del programa”.

Cuando el controlador lee la función de fin de programa `M02` o `M30`, se cancelan los movimientos de ejes, la rotación del husillo y usualmente se reinicia el sistema a las condiciones de iniciales. En algunos controladores el reinicio no es automático y el programador debe estar consiente de ello.

En la mayoría de los controladores modernos se cuenta con un parámetro de sistema que determina si `M02` y `M30` tienen el mismo significado o no. Este ajuste puede dar la capacidad de rebobinado, muy útil en situaciones donde los viejos programas deben ser usados en máquinas modernas y no hay que hacerles cambios.

En resumen, si el fin del programa es terminado por la función `M30`, se realizará el rebobinado y si se usa la función `M02` el rebobinado no será realizado.

Cuando escribamos un programa asegúrese que el último bloque tenga nada más que el `M30` como fin preferido (algunos sistemas de control más antiguos requieren que el `M30` vaya seguido del fin de archivo `%`):

```
N65 ...
N66 G91 G28 X0 Y0
N67 M30                 (FIN DEL PROGRAMA)
%
```

En algunos controladores, la función `M30` puede ser usada junto con los ejes de movimientos, aunque no es lo más recomendable:

```
N65 ...
N66 G91 G28 X0 Y0 M30  (FIN DEL PROGRAMA)
\%
```

El signo de porciento (%) después de `M30` es un *código de paro* especial. Este símbolo termina la carga de un programa de un dispositivo externo (como un dispositivo de cinta, o cuando el programa es transmitido por comunicación serial o de otro tipo). También es llamado *marcador de fin de archivo*.

Fin de subprograma

La ultima función para el fin de programa es M99. Es usada principalmente en los subprogramas. Típicamente la función M99 termina un subprograma y regresa al procesamiento del programa previo. Si M99 es usada en un programa estándar se crea un programa sin fin y tal situación es llamada ciclo infinito. M99 debe ser usado solo en subprogramas y no en programas estándares.

X.3.5 Funciones de máquina

Las funciones misceláneas relacionadas al funcionamiento de la máquina herramienta son parte de otro grupo. Esta sección describe a detalle las más importantes de estas funciones.

Funciones del refrigerante

La mayoría de las operaciones de remoción de metal requieren que la herramienta de corte esté empapada con un apropiado refrigerante. Para controlar el flujo del refrigerante en el programa, usualmente existen tres funciones misceláneas que proporcionan esta característica:

M07	Refrigerante encendido (brisa)
M08	Refrigerante encendido (empapar herramienta)
M09	Refrigerante apagado

Brisa es la combinación de una pequeña cantidad de aceite de corte combinado con aire comprimido. Esta característica depende del fabricante de la máquina herramienta si esta función es o no estándar para una máquina herramienta en particular. Algunos fabricantes reemplazan esta mezcla de aire y aceite por aire únicamente. En tales casos se cuenta con un equipo especial en la máquina. Si esta opción existe en la máquina la función miscelánea más común para activar el la brisa de aceite o aire es `M07`.

La función M08 es similar a M07. Ésta es por mucho la más común de las aplicaciones del refrigerante en la programación CNC. El refrigerante es usualmente una mezcla de aceite soluble en agua. Esta es mezcla se prepara previamente y se almacena en el tanque del refrigerante de la máquina herramienta. Empapar la orilla cortante de la herramienta es importante por tres razones:

- Disipar calor.
- Remover viruta.
- Lubricar.

La razón primaria para usar refrigerante es disipar el calor generado durante el corte apuntando a la orilla cortante. La segunda razón es remover las virutas del área de corte usando refrigerante a presión. Finalmente, el refrigerante también actúa como lubricante para facilitar la fricción entre la herramienta de corte y el material. El lubricante ayuda a extender al vida de la herramienta de corte y mejorar la superficie final.

Durante el acercamiento de la herramienta inicial hacia la parte o durante el regreso a la posición de cambio de herramienta el refrigerante no se requiere. Para apagar la función del refrigerante se usa la función `M09`. En realidad la función `M09` apaga el motor de la bomba del refrigerante.

Cada una de las tres funciones relacionadas al refrigerante puede ser programada en bloques separados o junto con comandos de movimiento. Hay una sutil pero importante diferencia entre en el orden y tiempo del procesado del programa, los siguientes ejemplos explican las diferencias:

Ejemplo 10.10 La brisa de aceite es prendida si es que esta disponible dicha función:

```
N110 M07
```

Ejemplo 10.11 El refrigerante es prendido:

```
N340 M08
```

Ejemplo 10.12 El refrigerante es apagado.

```
N500 M09
```

Ejemplo 10.13 Movimiento de ejes y refrigerante encendidos:

```
N230 G00 X11.5 Y10.0 M08
```

Ejemplo 10.14 El refrigerante es encendido:

```
N400 G00 Z 1.0 M09
```

Los ejemplos muestran las diferencias en el proceso de programación. Las reglas de programación del refrigerante son:

- Refrigerante encendido o apagado en un bloque separado llega a estar activo en el bloque en el cual es programado (ejemplos 10.10, 10.11 y 10.12).
- Refrigerante encendido, cuando es programado con los ejes de movimiento está activo simultáneamente con el movimiento de ejes.
- Refrigerante apagado, programado con ejes de movimiento estará activo solo cuando se complete el movimiento de los ejes.

El principal propósito de la función `M08` es encender el motor de la bomba. Esto no garantiza que la punta de corte reciba inmediatamente el refrigerante. En máquinas grandes con grandes pipas de refrigerante o máquinas con bombas de presurización de refrigerante lentas, se espera un retardo antes que el refrigerante llegue a su a la herramienta de corte.

El refrigerante debería de ser programado teniendo dos consideraciones en mente:

- No debe salpicar refrigerante fuera del área de trabajo (fuera de la máquina).
- Nunca deberá de haber una situación donde el refrigerante alcance una orilla caliente de la herramienta.

La primera condición es relativamente menor. Si la función de refrigerante es programada en el lugar incorrecto el resultado puede ser solo un inconveniente. El área húmeda alrededor del área de trabajo puede presentar condiciones de trabajo inseguras y deberían de ser corregidas rápidamente. Incluso situaciones más serias pueden ocurrir cuando de repente el refrigerante empieza a empapar una herramienta de corte que ya ha entrado al material. El cambio de temperatura en la orilla de corte puede causar que la herramienta de corte se rompa y dañe la parte. Las herramienta de carburo son más fáciles de ser afectadas por los cambios de temperatura que la herramientas de acero de alta velocidad. Una posibilidad de prevenir esto durante la programación es programar la función `M08` unos bloques antes del bloque del corte. Largas tuberías e insuficiente presión del refrigerante sobre la máquina puede demorar el inicio del flujo de refrigerante.

Funciones del husillo

Las funciones M disponibles para el husillo controlan su rotación y orientación, mientras que la dirección `S` controla su velocidad.

La mayoría de los husillos pueden rotar en ambas direcciones *en sentido de las manecillas del reloj* (dextrógiro)) y *en sentido contrario a las manecillas del reloj* (levógiro). La dirección de rotación siempre es relativa a la dirección hacia donde apunta el husillo. La rotación dextrógira programada como `M03` y la dirección levógira es programada como `M04`, asumiendo que el husillo puede rotar en ambas direcciones.

La función del husillo para programar un paro del husillo es `M05`; esta función esta función detendrá la rotación del husillo, sin importar la dirección de rotación que este tenga. Sobre muchas máquinas la función miscelánea debería de ser programada antes de revertir la rotación del husillo; por ejemplo:

```
M03
...          (HUSILLO GIRANDO HACIA LA DERECHA)
... (MAQUINANDO EN LA POSICIÓN ACTUAL) ...
...
M05                          (PARO DEL HUSILLO)
... (USUALMENTE UN CAMBIO DE HERRAMIENTA) ...
M04       (HUSILLO GIRANDO HACIA LA IZQUIERDA)
... (MAQUINANDO EN LA POSICIÓN ACTUAL) ...
...
```

La función `M05` también puede ser requerida cuando cambiamos engranes en tornos CNC. Si programamos un paro del husillo en un bloque donde exista movimiento de ejes, la función de paro se llevará a cabo después de que la acción de movimiento haya sido completada.

La última función de control del husillo es la función `M19`, llamada *la orientación del husillo*. La función `M19` provocará el paro del husillo en una posición orientada. Esta función es usada en principalmente durante los ajustes de la máquina y muy raramente en el programa. El husillo debe ser orientado en dos principales situaciones:

- Cambio automático de la herramienta.
- Cambio de herramienta durante una operación de taladrado.

Cuando usamos la función `M06` de cambio automático de herramienta en un programa no hay necesidad de programar la orientación del husillo en la mayoría de los centros de maquinado CNC. La orientación es parte del cambio de herramienta y se garantiza la posición correcta de todas las herramientas de corte.

La orientación del husillo es necesaria para ciertas operaciones de taladrado sobre los sistemas de fresado. Para salir de un taladrado se debe poner primero el husillo en la orientación correcta y después retirar la herramienta de corte.

En conclusión, la función `M19` es raramente programada, está disponible como ayuda en la programación para el operador de la máquina para los ajustes de trabajo, usando las operaciones de la entrada manual de datos.

X.4. Componente Compilador de código G

X.4.1 Analizador lexicográfico

El trabajo del analizador lexicográfico consiste en extraer los caracteres del flujo de entrada y agruparlos en símbolos para poder determinar posteriormente si la sintaxis del programa es correcta. Como vimos en la sección 10.1, además de letras y dígitos, solo se admiten los caracteres de la tabla 10.1, así que las letras de la A a la Z, junto con los dígitos del 0 al 9 y los caracteres de la tabla 10.1 formarán nuestro alfabeto de entrada Σ. Los símbolos léxicos necesarios para reconocer el código G, junto con su correspondiente expresión regular, se listan en la tabla 10.13, en la cual se emplean las notaciones $\texttt{letra} = (\texttt{A}|\cdots|\texttt{Z})$, $\texttt{dígito} = (\texttt{0}|\cdots|\texttt{9})$ y $\textvisiblespace$ para representar cualquier espacio en blanco, es decir, cualquier espacio o tabulador, `\n` para representar un salto de línea y `EOF` para representar el fin de archivo.

Símbolo léxico	Expresión regular	Nombre
Identificador de programa	:|O	TOKEN_PROGRAM_NUMBER
Dirección	letra	TOKEN_ADDRESS
Entero sin signo	$(+\|\varepsilon)$dígito*	TOKEN_UINT
Entero negativo	-dígito*	TOKEN_NINT
Flotante sin signo	$(+\|\varepsilon)$(dígito dígito*.dígito* |dígito*.dígito dígito*)	TOKEN_UFLOAT
Flotante negativo	$-$(dígito dígito*.dígito* |dígito*.dígito dígito*)	TOKEN_NFLOAT
Comentario	((letra|dígito)*)	TOKEN_COMMENT
Omitir bloque	/	TOKEN_BLOCK_DELETE
Espacio en blanco	␣	TOKEN_WHITE_SPACE
Código de transmisión	%	TOKEN_TX_CODE
Fin de bloque	;	TOKEN_EOB
Fin de línea		TOKEN_EOL
Fin de archivo	EOF	TOKEN_EOF

Tabla 10.13: Expresiones regulares de los símbolos léxicos para el código G.

Después de aplicar las técnicas de la sección A.2.3 a las expresiones regulares de la tabla 10.13 obtenemos el AFD del diagrama de estados UML de la figura 10.3. Para comprender a detalle el diagrama de ésta figura hace falta discutir la forma en que fue diseñado el analizador léxico y su implementación en C++.

La clase istream representa en C++ un flujo de entrada. Cualquier clase derivada, siempre y cuando provea la misma interfaz, puede ser considerada también una istream y, en consecuencia, puede ser empleada como fuente de datos para el analizador léxico. Si bien no es posible garantizar que la fuente de datos será un archivo abierto en un editor, cada línea del flujo se separa por medio del carácter de nueva línea `\n`, así que siempre es posible determinar la línea del flujo de entrada (ya sea un editor o un archivo) en que se encuentra el analizador léxico. Para aislar el código que se encarga de la administración del flujo de entrada se creó la clase StreamReader que se muestra en la figura 10.4 y se encarga de mantener un registro del carácter actualmente leido y del renglón y la columna en que se encuentra. Cuando se comienza a leer un archivo desde el principio, ya sea que se acabe de establecer el apuntador `inputStream_` por medio del método `setInputStream` (el constructor establece `inputStream_ = 0`) o se acabe de reiniciar el flujo de entrada con el método `resetInputStream`, se establecen los contadores para el renglón `row_` y para la columna `column_` a 1 ya que, por convención, se considera que el primer carácter de un archivo se ubica en la primera columna del primer renglón. Hay dos posibles operaciones básicas a realizar sobre un flujo de entrada, a saber,

- leer un carácter por medio del método protegido `getChar`, o bien
- "desleer" el último carácter leido con ayuda del método protegido `putBackChar`.

Además, es posible reportar el renglón y la columna actuales empleando los métodos `getRow` y `getColumn`, respectivamente. La operación de la clase es muy simple en general pues cada que se lee un carácter con `getChar` se incrementa la columna (`column_++`), a menos

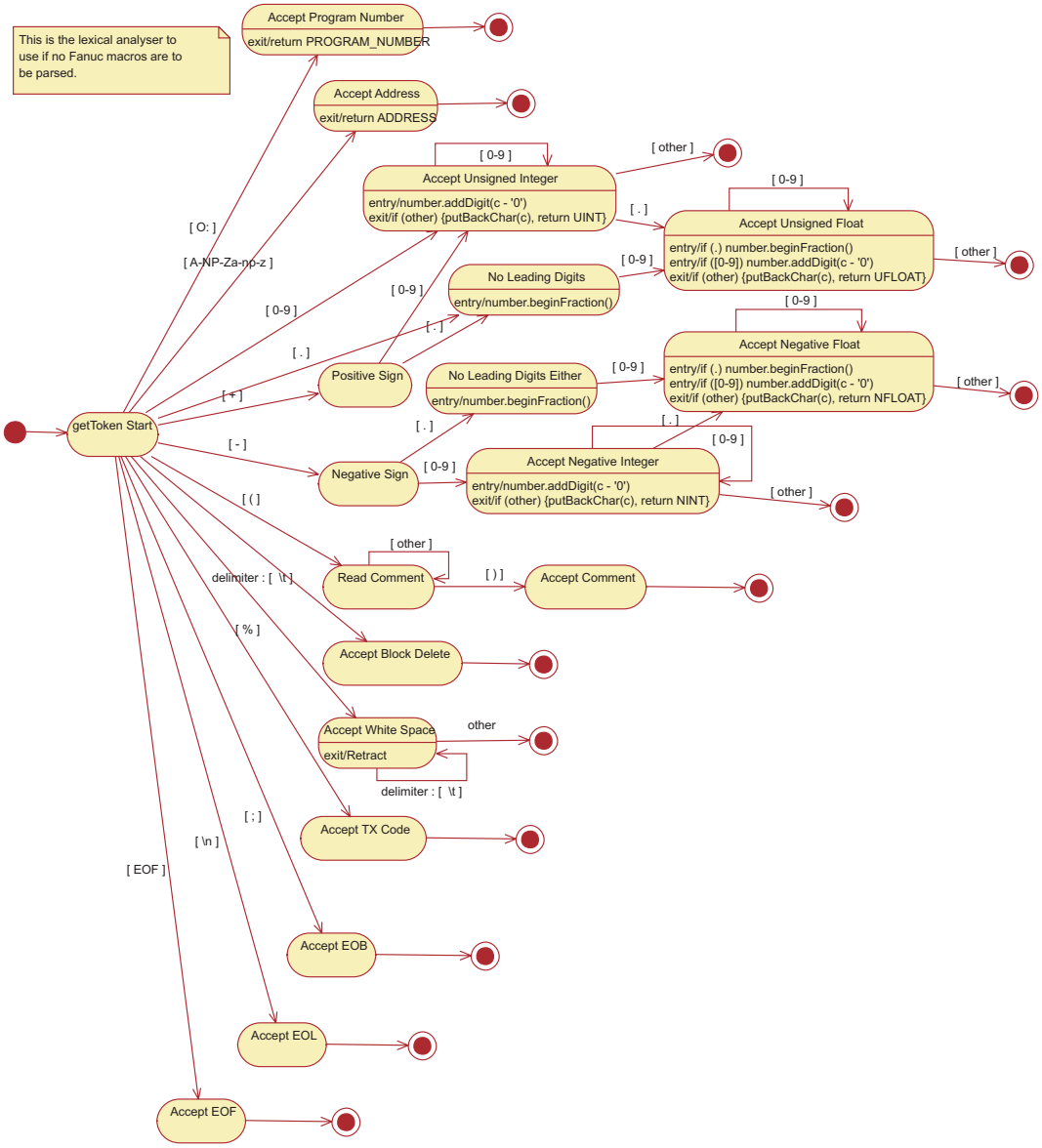


Figura 10.3: Diagrama de estados para el análisis léxico.

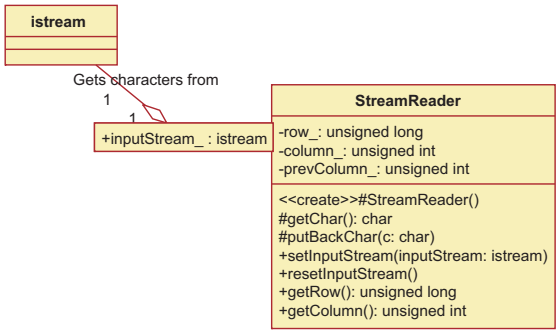


Figura 10.4: La clase StreamReader administra la lectura de un flujo de entrada istream.

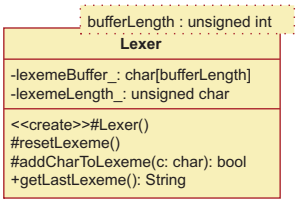


Figura 10.5: La clase Lexer simplifica el almacenamiento de lexemas.

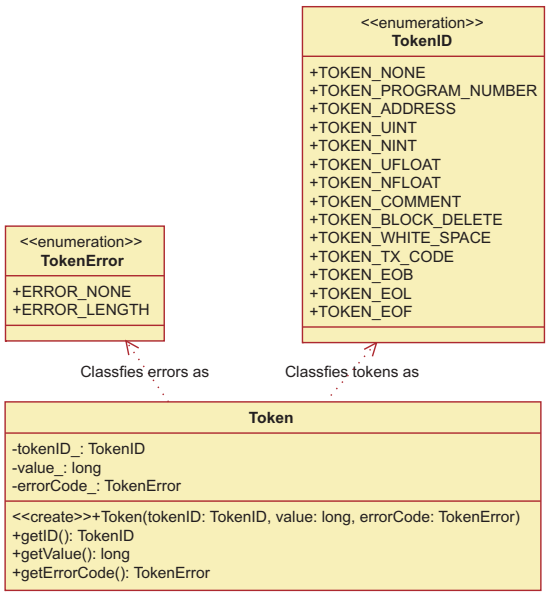


Figura 10.6: La clase Token representa a los símbolos léxicos.

que el carácter leido haya sido `\n` ya que en ese caso se establecerá `column_ = 1` y se incrementará `row_++`. Asimismo, al "desleer" un carácter con `putBackChar` simplemente se decrementa la columna (`column_++`), a menos que el caracter "desleido" sea `\n` ya que en ese caso se decrementará `row_-` y se establecerá `column_ = prevColumn_`, donde `prevColumn_` es un valor almacenado durante la ejecución de `getChar`. Los métodos `getChar` y `putBackChar` son métodos protegidos en linea para incrementar la eficiencia de cualquier clase derivada.

La clase Lexer de la figura 10.5 es una clase que simplifica el almacenamiento de lexemas al proveer el método `resetLexeme` para establecer el lexema a ε, el método `addCharToLexeme` para agregar un carácter al lexema y el método `getLastLexeme` para reportar el lexema almacenado.

Los símbolos léxicos se representan por medio de la clase Token de la figura 10.6 la cual almacena un identificador del tipo de enumeración TokenID (que corresponde al campo **Nombre** de la tabla 10.13) para discernir a cuál token nos referimos y un valor entero de 32 bits (long int) para reportar al analizador sintáctico de los valores numéricos de las intrucciones proporcionadas por el usuario; asimismo, se incluye un campo del tipo TokenError para reportar si el símbolo léxico ha excedido la máxima longitud permitida. En la tabla 10.14 se listan los símbolos léxicos del analizador léxico junto con el significado del valor entero y la condición de error. Cuando no hay error simplemente se regresa el identi-

ficador ERROR_NONE y cuando ocurre algún error se regresa ERROR_LENGTH; aunque no se descarta en un futuro agregar nuevos mensajes de error, por el momento todos los errores considerados tienen que ver con algún símbolo léxico que excede la longitud permitida ya que, como se aprecia en la tabla 10.2, las constantes numéricas de más de 8 dígitos no son permitidas.

Identificador	**Valor**	**Condición de error**
TOKEN_PROGRAM_NUMBER	Carácter leido (: u O)	No hay error.
TOKEN_ADDRESS	Carácter letra leido	No hay error.
TOKEN_UINT	Valor numérico	Se excede de 8 dígitos.
TOKEN_NINT	Valor numérico	Se excede de 8 dígitos.
TOKEN_UFLOAT	Valor numérico	Se excede de 8 dígitos.
TOKEN_NFLOAT	Valor numérico	Se excede de 8 dígitos.
TOKEN_COMMENT	0	No hay error.
TOKEN_BLOCK_DELETE	Carácter leido (/)	No hay error.
TOKEN_WHITE_SPACE	0	No hay error.
TOKEN_TX_CODE	Carácter leido (%)	No hay error.
TOKEN_EOB	Carácter leido (;)	No hay error.
TOKEN_EOL	Carácter leido (\n)	No hay error.
TOKEN_EOF	EOF	No hay error.

Tabla 10.14: Valores y condiciones de error de los símbolos léxicos.

Los valores numéricos de los símbolos léxicos que representan números, a saber, TOKEN_UINT, TOKEN_NINT, TOKEN_UFLOAT y TOKEN_NFLOAT se almacenan en un formato especial de 32 bits. Esto es posible debido a que los números empleados en programación de control numérico no requieren más de 8 dígitos:

- En sistema internacional el incremento mínimo es de 0.001mm y el desplazamiento máximo de 99999.999mm ≈ 100m.
- En sistema inglés el incremento mínimo es de 0.0001in $= 0.00254$mm y el desplazamiento máximo de 9999.9999in ≈ 254m.

Son muy pocas las aplicaciones que requieren de maquinaria más precisa o con dimensiones más grandes. Si bien no todos los números de un programa representan dimensiones, cualquier otro número tal como un índice de herramienta o un número de línea se puede representar con 8 dígitos.

El almacenamiento de los números en este formato de 32 bits es posible ya que, como

$$\lceil \log_2 99999999 \rceil = 27,$$

es posible almacenar un número entero de 8 dígitos junto con su signo empleando los 28 bits más significativos, a lo cual llamaremos el *significando*. Los 4 bits menos significativos se pueden emplear para almacenar un número con signo que represente un *exponente* de un factor 10 para desplazar el punto decimal de lugar. Así pues, si denotamos al significando

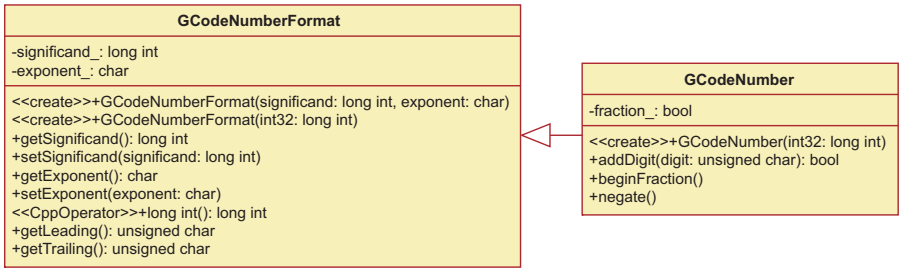


Figura 10.7: La clase GCodeNumberFormat almacena valores numéricos de los símbolos léxicos.

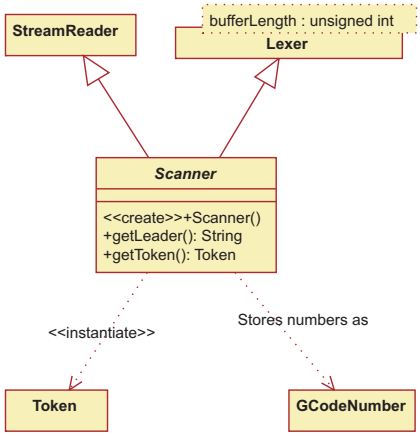


Figura 10.8: La clase Scanner implementa el analizador léxico por herencia múltiple y en colaboración con otras clases que representan a los símbolos léxicos y sus valores numéricos.

por s y al exponente por n entonces los 32 bits en su conjunto representarán al número

$$x = s \times 10^n.$$

El signo de x es el mismo que el de s y el punto decimal se recorre a la izquierda si $n < 0$; el punto decimal se puede recorrer hasta 8 posiciones a la izquierda (con $n = -8$) y hasta 7 posiciones a la derecha (con $n = 7$), aunque las posiciones hacia la derecha (es decir, $n > 0$) no serán utilizadas para representar número alguno dentro del sistema de control. Así, pues se tiene la clase GCodeNumberFormat de la figura 10.7 para almacenar números en este formato de 32 bits. Esta clase tiene un constructor a partir de un significando y un exponente y otro que nos permite crear el número a partir de un long int (con signo), así como métodos para leer y escribir el significando y el exponente, una conversión implícita a long int y los métodos `getLeading` y `getTrailing` para reportar el número de dígitos antes y después del punto decimal, respectivamente.

Para poder ensamblar fácilmente un número en formato GCodeNumberFormat dígito a dígito se creó la clase derivada GCodeNumber que además provee métodos para agregar un dígito a la vez (`addDigit`), indicar la posición del punto decimal (`beginFraction`) y para cambiar el signo del número (`negate`).

La clase Scanner de la figura 10.8 implementa al analizador léxico. Tiene dos modos de operación, dependiendo de la parte del programa que se esté reconociendo, como se

explica en más detalle en la sección que discute el diseño del analizador sintáctico. Cuando se emplea el método `getLeader` se extrae una descripción del programa por medio del AFD de la figura 10.3, mientras que si se emplea el método `getToken` se extrae un símbolo léxico a la vez de los que se encuentran en la tabla 10.13 por medio del AFD de la figura 10.11.

X.4.2 Analizador sintáctico

Como vimos en la figura 10.2, los principales componentes de un programa son:

Inicio Símbolo que indica que el lector de programas se encuentra al inicio de la cinta, es decir, al inicio del archivo.

Descripción Es una cadena de texto que describe el propósito del programa o cualquier otra información útil que permita distinguir al programa; no incluye a la nueva línea que lo da por terminado y la separa del resto del programa.

Número del programa Da inicio a la sección del programa. Es una palabra que identifica al programa por medio de un número entero sin signo.

Bloques de instrucciones El conjunto de bloques que conforma al conjunto de instrucciones contenidas en el programa. Dentro de los bloques puede haber comentarios. El fin del programa, usualmente `M30\n` debe incluirse en el último bloque del programa.

Fin Símbolo que indica que el lector de programas ha alcanzado el fin de la cinta, es decir, el fin del archivo.

Desde un punto gramatical, podemos establecer que

$$\begin{aligned}
\langle \text{programa} \rangle &\rightarrow \textbf{inicio}\ \langle \text{datos} \rangle \\
\langle \text{datos} \rangle &\rightarrow \textbf{descripción}\ \langle \text{instrucciones} \rangle \\
\langle \text{instrucciones} \rangle &\rightarrow \textbf{número}\ \langle \text{bloques} \rangle \\
\langle \text{bloques} \rangle &\rightarrow \textbf{bloque}\ \langle \text{bloques} \rangle \\
\langle \text{bloques} \rangle &\rightarrow \textbf{fin}
\end{aligned}$$

Esta gramática es regular y tiene asociado el AFD de la figura 10.9. En realidad los símbolos **descripción**, **número** y **bloque** no son terminales ya que a su vez están compuestos por entidades más básicas. No obstante, su utilidad para estructurar el análisis léxico es grande pues permiten dividir el análisis por secciones. La clase Parser mostrada en la figura 10.10 en la página siguiente provee el método `getProgramComponent` que regresa un identificador de la clase de enumeración ProgramComponentID para cada uno de los componentes de un programa. De esta manera es posible desglosar el análisis del código G en estados de la manera siguiente:

```
...
Scanner scanner;
Parser parser;
Block block;
```

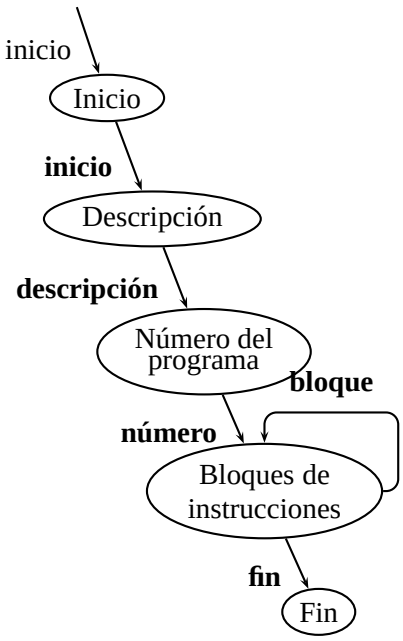


Figura 10.9: AFD para reconocer un programa de código G.

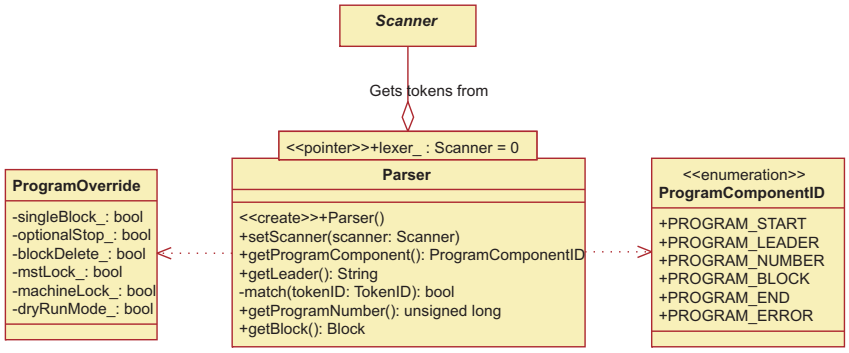


Figura 10.10: AFD para desglosar la descripción de un programa.

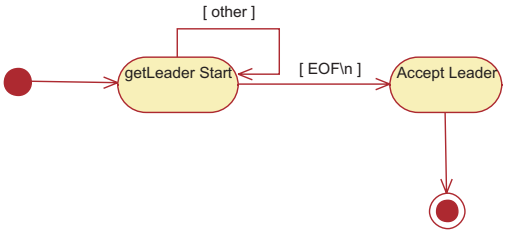


Figura 10.11: AFD para desglosar la descripción de un programa.

```
...
parser.setScanner(&scanner);
...
do {
        programComponent = parser.getProgramComponent();
        switch (programComponent) {
        case PROGRAM_START:
                ...
                break;
        case PROGRAM_LEADER:
                ...
                strcpy(leader, parser.getLeader());
                ...
                break;
        case PROGRAM_NUMBER:
                ...
                number = parser.getProgramNumber()
                ...
                break;
        case PROGRAM_BLOCK:
                ...
                block = parser.getBlock();
                ...
                break;
        case PROGRAM_END:
                ...
                break;
        case PROGRAM_ERROR:
                ...
                break;
        }
} while (programComponent != PROGRAM_END
        || programComponent != PROGRAM_ERROR);
```

Es posible dar una gramática que describa la forma en que cada uno de estos componentes es representado, pero resulta más directo presentar un diagrama de estados UML

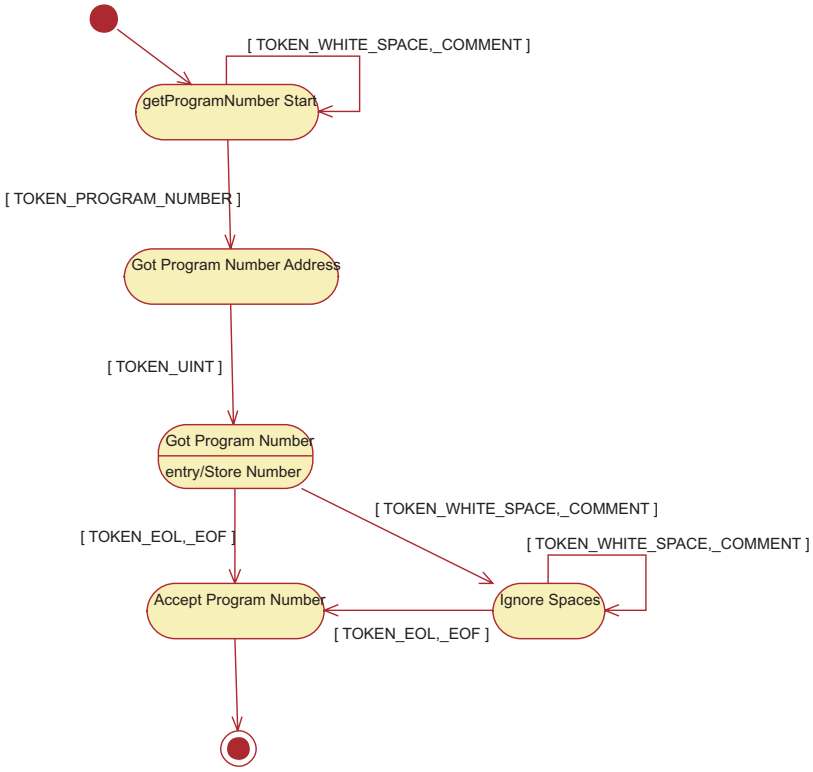


Figura 10.12: AFD para desglosar el número de un programa.

para cada uno de ellos. La figuras 10.11, 10.12 y 10.13 muestran los AFDs para desglosar la descripción, el número de programa y un bloque, respectivamente, que se implementan en los métodos `getLeader`, `getProgramNumber` y `getBlock`.

El diagrama de estados de la figura 10.11 tiene transiciones “disparadas” no por los símbolos léxicos de la tabla 10.13, sino por caracteres del flujo de entrada, razón por la cual su reconocimiento es realmente realizado en el método `getLeader` de la clase Scanner y dicho método es simplemente utilizado para implementar el método del mismo nombre de la clase Parser.

La clase Parser almacena la información de un bloque del archivo fuente por medio de la clase Block, que representa ésta información por medio de una lista de objetos de la clase Word, como se muestra en la figura 10.14. Cada uno de los bloques, como se aprecia en la figura 10.13, es clasificado en alguna de las categorías listadas por la clase de enumeración BlockID:

`BLOCK_NONE` El bloque no existe, es decir, ya se ha encontrado el fin de archivo.

`BLOCK_EMPTY` No se ha encontrado el fin de archivo, pero no hay una sola palabra en el bloque; el bloque muy bien podría en realidad ser una línea con un comentario.

`BLOCK_SKIPPED` El bloque comenzaba con el carácter `/` para omitir bloque fue ignorado debido a que la bandera `blockDelete_` de la clase ProgramOverride estaba habilitada; si la bandera está deshabilitada entonces simplemente se ignora este carácter.

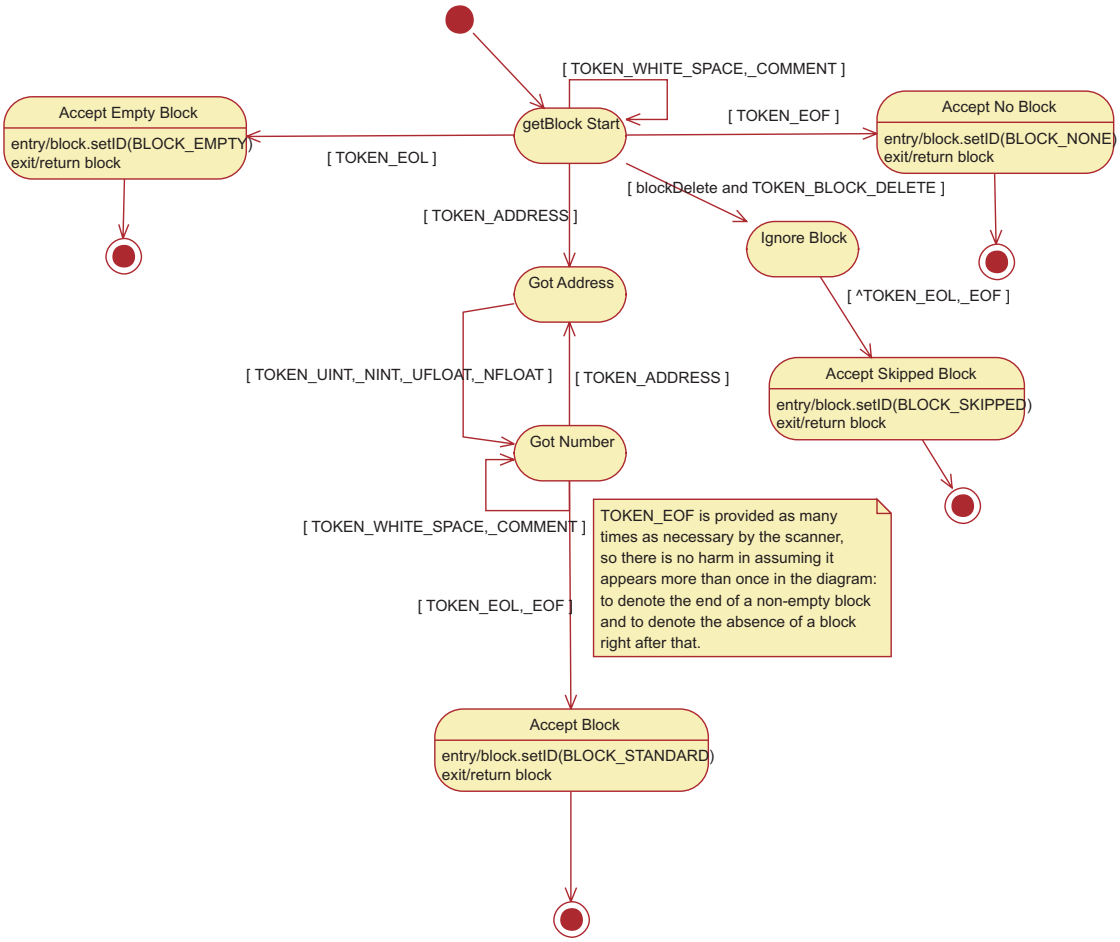


Figura 10.13: AFD para desglosar un bloque de un programa.

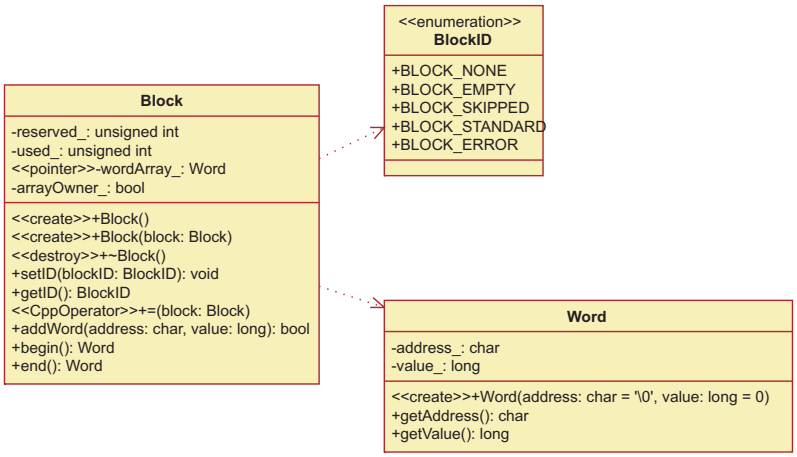


Figura 10.14: Estructura de datos para representar un bloque de un programa.

`BLOCK_STANDARD` El bloque ha sido leido exitosamente y contiene cada una de las palabras en el flujo de entrada.

`BLOCK_ERROR` Se ha encontrado un error en la lectura del bloque y la información podría estar incompleta.

Es notable que en el AFD de la figura 10.13 no considera al fin del programa `M30\n` como un bloque especial (véase la figura 10.2) debido a que los sistemas de control más recientes no requieren que sea la única palabra del último bloque, sino que puede aparecer junto con otras palabras en el último bloque. Verificar que el fin de programa `M30\n` realmente se encuentre presente y aparezca en el bloque correcto depende del significado que tenga cada palabra de cada bloque y, en ese sentido, corresponde más bien al analizador semántico garantizar que el programa se encuentre en buen estado.

Para evitar listas de palabras duplicadas, un bloque puede tener o no propiedad de las palabras contenidas en él; si tiene propiedad de las palabras que contiene, entonces su destructor eliminará la lista de palabras cuando ya no sea necesaria, o simplemente la ignorará si no es el propietario de la misma. De esta manera es posible evitar la duplicación de memoria cuando se emplea un constructor de copias o un bloque se asigna a otro. Aunque en esta versión del código la implementación fue "de bajo nivel", es posible implementar este comportamiento por medio de la plantilla de clases auto_ptr o algún otro apuntador inteligente.

La clase ProgramOverride contiene el estado de algunas banderas, usualmente controladas por interruptores en el panel de operación de la máquina-herramienta, que pueden modificar la forma en que el programa de control numérico es interpretado:

Single block Indica al sistema de control, cuando es verdadero, que se debe ejecutar el programa bloque por bloque, y el inicio de la ejecución de cada bloque se indicará por medio de la tecla Inicio de ciclo (la verde grande en el panel de operación).

Optional Stop Habilita (verdadero) o deshabilita (falso) el paro opcional (M01 de la tabla 10.5).

Block Delete Habilita (verdadero) o deshabilita (falso) la omisión de bloques precedidos por `/`.

MST Lock Deshabilita (verdadero) o habilita (falso) las acciones de las direcciones M, S y T.

Machine Lock Deshabilita (verdadero) o habilita (falso) todos los movimientos que involucran a los ejes de la máquina.

Dry Run Mode Habilita (verdadero) o deshabilita (falso) altas velocidades de desplazamiento para los ejes, permitiendo la ejecución del programa con todas las demás funciones en la manera usual. Se emplea para probar programas y se espera que no se use con una pieza a maquinar sujeta en la máquina.

De estas banderas solo la bandera `blockDelete_` es empleada por la clase Parser para decidir si se omiten los bloques precedidos por `/` o no.

X.5. Componente Base de datos de programas

La Base de datos de programa es un componente simple que ofrece flujos de lectura o de escritura de cada uno de los programas almacenados en la misma por medio de un identificador que consiste en una cadena de caracteres (que, según su implementación, podría ser el nombre del archivo que se desea emplear). Además, de ofrecer flujos de lectura o escritura, cada programa también tiene asociada una cadena descriptiva de hasta 256 caracteres.

XI. CONTROL DE MOVIMIENTO

XI.1. Políticas de control de movimiento

Dependiendo del equipo con el que se cuente para realizar control de movimiento, es posible implementar dos políticas de control de movimiento distintas:

1. control de movimiento punto a punto, o

2. control de movimiento continuo.

El control de movimiento punto a punto es necesario cuando el hardware de control de movimiento con el que se cuenta ofrece una funcionalidad muy básica que se limita al seguimiento de un punto de referencia y no ofrece la capacidad de generar trayectorias complejas interpoladas tales como un segmento rectilíneo o un segmento de arco; la generación de trayectorias típicas de la programación de sistemas de CNC se describe en la sección siguiente.

El control de movimiento continúo, por otra parte, se emplea cuando el hardware de control del movimiento si ofrece la capacidad de generar trayectorias por medio de comandos de alto nivel, tales como las tarjetas de control de movimiento comerciales (e.g. Galil o PMAC de DeltaTau).

XI.2. Generación de trayectorias interpoladas

XI.2.1 Segmentos lineales

Consideremos un segmento lineal entre los puntos (x_1, y_1) y (x_2, y_2), como se ilustra en la figura 11.1. Si definimos un parámetro t en el intervalo $[0, 1]$ entonces podemos calcular

$$\ell(t) = t(x_2, y_2) + (1 - t)(x_1, y_1).$$

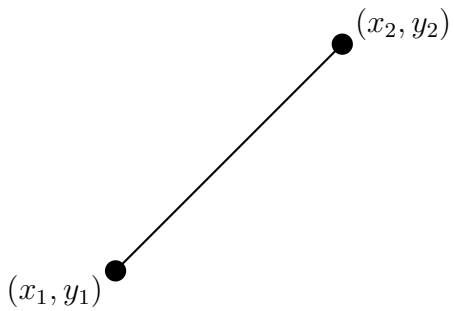


Figura 11.1: Segmento lineal entre los puntos (x_1, y_1) y (x_2, y_2).

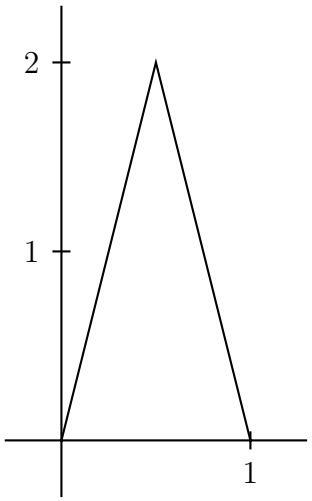


Figura 11.2: Gráfica de la función f.

La distancia entre los puntos (x_1, y_1) y (x_2, y_2) es

$$s = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

así que, si la máxima distancia que deseamos recorrer entre dos puntos de referencia es Δs entonces necesitaremos de al menos

$$n = \left\lceil \frac{s}{\Delta s} \right\rceil$$

puntos de referencia. Tomaremos entonces los puntos de referencia dados por

$$t = 0, \frac{1}{n}, \frac{2}{n}, \ldots, 1.$$

Sin embargo, si deseamos modificar la velocidad inicial (para vencer la fricción estática, por ejemplo), es recomendable reparametrizar la trayectoria. Para tal fin construiremos una función (monótona) creciente F tal que $F(0) = 0$ y $F(1) = 0$, lo cual es posible integrando la función

$$f(x) = \begin{cases} 4x, & 0 \leq x < \frac{1}{2}, \\ 4(1-x), & \frac{1}{2} \leq x \leq 1, \end{cases}$$

que describe un triángulo de área unitaria (figura 11.2). Al integrar obtenemos la función

$$F(x) = \begin{cases} 2x^2, & 0 \leq x < \frac{1}{2}, \\ -2x^2 + 4x - 1, & \frac{1}{2} \leq x \leq 1. \end{cases}$$

La máxima pendiente de esta función es $f(\frac{1}{2}) = 2$, así que para garantizar que la máxima distancia a recorrer entre dos puntos de referencia sea Δs tomaremos los $2n + 1$ valores

$$t = 0, \frac{1}{2n}, \frac{2}{2n}, \ldots, 1.$$

Entonces la parametrización será de la siguiente forma:

Algoritmo 11.1 Parametrización de un segmento rectilíneo que conecta los puntos (x_1, y_1) y (x_2, y_2):
establecer (x_1, y_1) y (x_2, y_2)
para $t = 0, \frac{1}{2n}, \frac{2}{2n}, \ldots, 1$ hacer:
    calcular $s = F(t)$
    calcular $(x, y) = \ell(s)$

XI.2.2 Segmentos circulares

Para una trayectoria circular que inicia en el punto (x_1, y_1), termina en el punto (x_2, y_2) y tiene radio r hay cuatro posibilidades (figura 11.3):

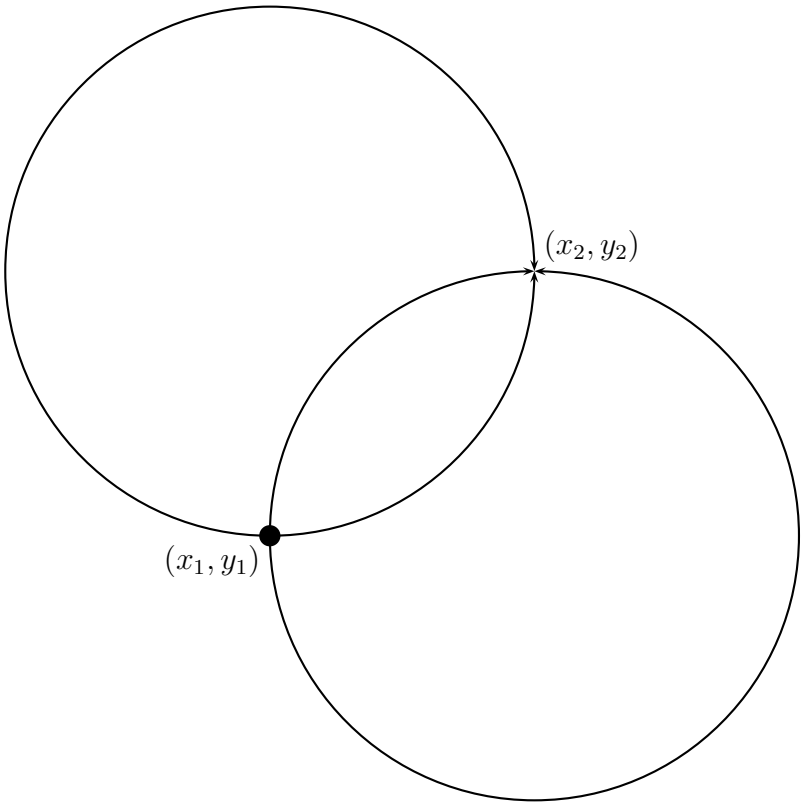


Figura 11.3: Hay cuatro posibles trayectorias circulares entre dos puntos.

1. Ir en el sentido de las manecillas del reloj, i.e., sentido horario, por un arco menor o igual a 180º.
2. Ir en el sentido de las manecillas del reloj, i.e., sentido horario, por un arco mayor a 180º.
3. Ir en contra del sentido de las manecillas del reloj, i.e., sentido antihorario, por un arco menor o igual a 180º.
4. Ir en contra del sentido de las manecillas del reloj, i.e., sentido antihorario, por un arco mayor a 180º.

En cualquiera de los cuatro casos lo que debemos primero determinar es el centro de la circunferencia a utilizar.

Dos posibilidades para el centro de la circunferencia

Si tanto (x_1, y_1) como (x_2, y_2) yacen sobre la misma circunferencia, entonces sus distancias al centro (x, y) de la circunferencia son iguales, i.e.,

$$\sqrt{(x-x_1)^2+(y-y_1)^2} = \sqrt{(x-x_2)^2+(y-y_2)^2},$$

o equivalentemente,

$$\begin{aligned}(x-x_1)^2+(y-y_1)^2 &= (x-x_2)^2+(y-y_2)^2,\\ -2x_1x+x_1^2-2y_1y+y_1^2 &= -2x_2x+x_2^2-2y_2y+y_2^2,\end{aligned}$$

de donde resulta la ecuación

$$2(x_2-x_1)x+2(y_2-y_1)y+x_1^2-x_2^2+y_1^2-y_2^2=0. \tag{11.1}$$

Además, como la distancia de (x_1, y_1) a (x, y) es r, también tenemos

$$(x-x_1)^2+(y-y_1)^2=r^2. \tag{11.2}$$

Al despejar y de (11.1) obtenemos

$$y=\frac{x_2^2-x_1^2+y_2^2-y_1^2}{2(y_2-y_1)}-\frac{x_2-x_1}{y_2-y_1}x, \quad y_2\neq y_1; \tag{11.3}$$

el caso $y_2 = y_1$ lo discutiremos al final. Al substituir (11.3) en (11.2) obtenemos

$$(x-x_1)^2+\left(\frac{x_2^2-x_1^2+(y_2-y_1)^2}{2(y_2-y_1)}-\frac{x_2-x_1}{y_2-y_1}x\right)^2=r^2.$$

Para resolver esta ecuación de manera más sencilla podemos definir

$$\begin{aligned}a&=\frac{x_2^2-x_1^2+(y_2-y_1)^2}{2(y_2-y_1)},\\ m&=-\frac{x_2-x_1}{y_2-y_1},\end{aligned}$$

para reescribirla como

$$(x-x_1)^2+(a+mx)^2=r^2,$$

o bien, en la forma usual para ecuaciones de segundo grado,

$$(m^2+1)x^2+2(am-x_1)x+a^2+x_1^2-r^2=0.$$

Por la fórmula general para resolver ecuaciones de segundo grado obtenemos

$$x = \frac{am - x_1 \pm \sqrt{(am - x_1)^2 - (m^2+1)(a^2 + x_1^2 - r^2)}}{m^2+1}.$$

Si definimos

$$\begin{aligned} b &= \frac{x_2^2 - x_1^2 + y_2^2 - y_1^2}{2(y_2 - y_1)}, \\ c &= am - x_1, \\ d &= (am - x_1)^2 - (m^2+1)(a^2 + x_1^2 - r^2) \end{aligned}$$

entonces podemos escribir las soluciones como

$$x = \frac{c \pm d}{m^2+1}$$

y calcular

$$y = mx + b.$$

El caso $y_2 = y_1$ en (11.1) describe una recta vertical (pues la mediatriz de un segmento horizontal es vertical), de modo que

$$x = \frac{x_2 + x_1}{2}. \tag{11.4}$$

Al substituir (11.4) en (11.2) obtenemos

$$\left(\frac{x_2 - x_1}{2}\right)^2 + (y - y_1)^2 = r^2,$$

de modo que

$$y = y_1 \pm \sqrt{r^2 - \left(\frac{x_2 - x_1}{2}\right)^2}.$$

Resumiendo, para encontrar los dos posibles centros (h_1, k_1) y (h_2, k_2) para la circunferencia se procede de la siguiente manera:

Algoritmo 11.2 Determinación de los dos posibles centros (h_1, k_1) y (h_2, k_2) para trazar un segmento de arco que una dos puntos (x_1, y_1) y (x_2, y_2):
establecer (x_1, y_1) y (x_2, y_2)
si $y_2 = y_1$ entonces:

$$\begin{aligned} h_1 &= h_2 = \frac{x_2 + x_1}{2} \\ k_1 &= y_1 + \sqrt{r^2 - \left(\frac{x_2 - x_1}{2}\right)^2} \\ k_2 &= y_1 - \sqrt{r^2 - \left(\frac{x_2 - x_1}{2}\right)^2} \end{aligned}$$

sino:

$$h_1 = \frac{c+d}{m^2+1}$$
$$h_2 = \frac{c-d}{m^2+1}$$
$$k_1 = mh_1 + b$$
$$k_2 = mh_2 + b$$

Selección del centro

En ambos casos (ya sea $y_2 = y_1$ o $y_2 \neq y_1$) obtenemos dos soluciones (h_1, k_1) y (h_2, k_2) distintas entre sí, cada una de las cuales corresponde a uno de los dos siguientes casos (mutuamente excluyentes):

1. el punto es centro de una rotación en sentido horario con arco menor o igual a 180º o centro de una rotación en sentido antihorario con arco mayor a 180º.

2. el punto es centro de una rotación en sentido antihorario con arco menor o igual a 180º o centro de una rotación en sentido horario con arco mayor a 180º.

Para determinar cuál centro corresponde a cuál caso bastará con clasificar (h_1, k_1) (pues el otro centro (h_2, k_2) corresponderá al otro caso). Emplearemos la regla de la mano derecha aplicada al producto cruz de los vectores

$$\vec{\mathbf{u}} = \overrightarrow{(h_1, k_1)(x_1, y_1)} = \langle x_1 - h_1, y_1 - k_1 \rangle,$$
$$\vec{\mathbf{v}} = \overrightarrow{(h_1, k_1)(x_2, y_2)} = \langle x_2 - h_1, y_2 - k_2 \rangle$$

(en ese orden); el producto cruz de los vectores será un vector perpendicular al plano que, por la regla de la mano derecha, apuntará hacia abajo del plano xy (tendrá una componente en z negativa, hacia adentro del papel) en el caso 1 y apuntará hacia arriba del plano xy (tendrá una componente en z positiva, hacia afuera del papel) en el caso 2 (véase la figura 11.4). La componente en z de $\vec{\mathbf{u}} \times \vec{\mathbf{v}}$ es

$$z = (x_1 - h_1)(y_2 - k_1) - (y_1 - k_1)(x_2 - h_1).$$

Resumiendo, si $z = (x_1 - h_1)(y_2 - k_1) - (y_1 - k_1)(x_2 - h_1) < 0$ entonces estamos en el caso 1 y sino en el caso 2; desafortunadamente esta afirmación no permite separar las rotaciones horarias de las antihorarias, lo cual complica un poco la programación de sistemas de control de movimiento para máquinas herramienta. Afortunadamente, podemos simplificar aún más la clasificación si tomamos, por convención, que un radio r negativo denotará una rotación con arco mayor a 180º: nuestro nuevo criterio será el signo del producto rz. Podemos usar entonces el siguiente algoritmo para determinar el centro de una rotación en sentido horario:

Algoritmo 11.3 Determinar si el centro de una rotación en sentido horario es (h_1, k_1) o es (h_2, k_2):
calcular $z = (x_1 - h_1)(y_2 - k_1) - (y_1 - k_1)(x_2 - h_1)$
si $rz < 0$ entonces:

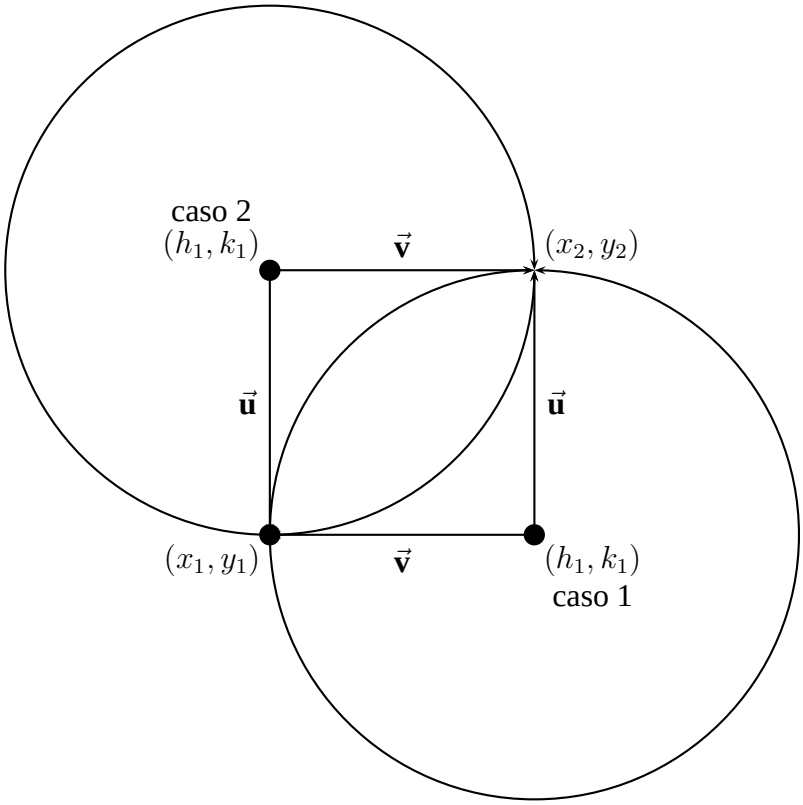


Figura 11.4: Hay dos casos para el centro del segmento de arco.

(h_1, k_1) es el punto buscado
sino:
(h_2, k_2) es el punto buscado

Un algoritmo muy parecido (pero con la condición $rz > 0$) nos permite determinar el centro de una rotación en sentido antihorario.

Longitud de arco

Podemos determinar fácilmente el ángulo inicial α y el ángulo final β del arco a trazar por medio de la función $\arctan_2$ disponible en la mayoría de los lenguajes de programación; no es una función trigonométrica estándar en virtud de que toma dos argumentos, a saber, las coordenadas de un punto (a, b) y nos regresa el ángulo que forma el segmento que va del origen al punto (a, b) con la horizontal en el intervalo $[pi, \pi]$. Nótese que, al tomar dos argumentos y no uno (como la función arco tangente clásica), esta función tiene información suficiente para determinar el cuadrante correcto en el que se encuentra el punto (a, b) y puede discernir del punto $(-a, -b)$, a diferencia de la función arco tangente tradicional para la cual

$$\arctan \frac{b}{a} = \arctan \frac{-b}{-a}.$$

Por esta razón a la función $\arctan_2$ a veces se le denomina *tangente inversa de cuatro cuadrantes*, para diferenciarla de la función $\arctan$ tradicional que nos regresa valores solo en el intervalo $[-\pi/2, \pi/2]$, i.e., en dos cuadrantes. Hay que tener cuidado, sin embargo, con el

orden de los argumentos de la función pues en algunos lenguajes de programación como el lenguaje C la tangente inversa de cuatro cuadrantes del punto (a, b), $\arctan_2(a, b)$, se denota por `atan2(b, a)`, es decir, con la ordenada primero y la abscisa después. En cualquier caso, ya sea que nuestra rotación sea en sentido horario o antihorario, si nuestro centro de rotación es (h, k), entonces los ángulos inicial α y final β se pueden determinar entonces como

$$\alpha = \arctan_2(h - x_1, k - y_1),$$
$$\beta = \arctan_2(h - x_2, k - y_2).$$

El ángulo θ subtendido por los puntos (x_1, y_1) y (x_2, y_2) se puede calcular entonces como

$$\theta = \begin{cases} \beta - \alpha, & \beta - \alpha \geq 0, \\ \beta - \alpha + 2\pi, & \beta - \alpha < 0. \end{cases}$$

Una vez conociendo θ, para hallar la longitud s del arco basta con hacer

$$s = r\theta.$$

Si, nuevamente, la máxima distancia que deseamos recorrer entre dos puntos de referencia es Δs y al mismo tiempo deseamos reparametrizar por medio de la función F, entonces calcularemos

$$n = \left\lceil \frac{s}{\Delta s} \right\rceil$$

y tomaremos los $2n + 1$ valores

$$t = 0, \frac{\theta}{2n}, \frac{2\theta}{2n}, \ldots, \theta,$$

y evaluaremos

$$c(t) = (h, k) + r(\cos[\alpha + t], \sin[\alpha + t]).$$

XI.3. Trayectorias con tarjeta de control de movimiento

Las tarjetas de control de movimiento comerciales usualmente cuentan con un lenguaje de programación especializado que permite utilizar a la misma como generador de trayectorias y, al mismo tiempo, como controlador de movimiento (para seguir los puntos interpolados). Como ejemplo, describiremos la funcionalidad básica de la tarjeta de control de movimiento DMC-18x2 de la empresa Galil Motion Control, Inc.

XI.3.1 Inicialización

Puesto que las tarjetas de control de movimiento también realizan, además de la generación, el seguimiento de la trayectoria, deben ser configuradas para reconocer la configuración de los dispositivos (servoamplificadores, interruptores del límite, etc.) con los que deberán trabajar. En la tabla 11.1 se se muestra un ejemplo (con una breve descripción de cada comando);

Comando Galil	Descripción
`CN1,-1`	Configurar la polaridad de los interruptores del límite y de los interruptores de inicio.
`VMXY`	Movimiento vectorial en el plano xy.
`LMXYZ`	Modo de interpolación lineal en los ejes x, y y z.
`OE1,1,1`	Desactivar motores si el error excede el límite especificado.
`ER8000,8000,8000`	Especificar el límite de error máximo permitído.
`KP257.54,320.63,71.50`	Establecer las constantes proporcionales del PID para los ejes x, y y z.
`KI93.35,85.71,33.88`	Establecer las constantes integrales del PID para los ejes x, y y z.
`KD1171.39,2509.63,289.25`	Establecer las constantes derivativas del PID para los ejes x, y y z.
`IT0.5,0.5,0.5`	Establecer constante de tiempo independiente para cada eje.
`VT0.5,0.5`	Establecer constante de tiempo vectorial para cada eje.

Tabla 11.1: Inicialización de una tarjeta Galil.

una descripción detallada del propósito de cada comando se puede encontrar en el manual de referencia de la tarjeta.

XI.3.2 Movimientos interpolados

Segmentos lineales

Para realizar un movimiento rápido de los ejes (G00) con la tarjeta Galil se puede emplear la sucesión de comandos `AC`, `DC`, `SP`, `PR` y `BG`.

Para realizar un movimiento lineal de los ejes (G01) con la tarjeta Galil se puede emplear la sucesión de comandos `LM`, `VS`, `VA`, `VD`, `LI`, `LE` y `BG`.

Arcos

Para realizar un movimiento circular (G02 y G03) con la tarjeta Galil se puede emplear la sucesión de comandos `CR`, `VS`, `VA`, `VD`, `VE` y `BG`.

XI.4. Componente Controlador de movimiento

Independientemente de la política de control de movimiento que se emplee, el componente Controlador de movimiento tiene como objetivo operar los ejes de la máquina herramienta para realizar trayectorias rectilíneas y circulares (segmentos de arco) en el plano coordenado y velocidad indicados. La compensación de la longitud de la herramienta o de su radio, así como la operación con respecto a sistemas coordenados distintos de la máquina es responsabilidad del Ejecutivo (véase la sección 7.3).

XII. CONTROL LÓGICO PROGRAMABLE

XII.1. Control lógico

Por *control lógico* se entiende el control de algún conjunto de dispositivos por medio de operaciones lógicas o, como también se les conoce, booleanas. Este tipo de control se originó en la industria automotriz donde reemplazó a grandes grupos de relevadores de control que se empleaban para la manufactura de automóviles. El control lógico es usualmente llevado a cabo por medio de dispositivos conocidos como *controladores lógicos programables* (PLCs, por sus siglas en inglés), los cuales usualmente son computadoras miniaturizadas con la capacidad de 'leer' entradas y 'escribir' salidas. Un PLC está formado por cuatro partes fundamentales:

Módulo de entrada Usualmente está conectado a interruptores límite (como los que se encuentran en las bancadas de una máquina herramienta), botones de retorno automático, interruptores y otros tipos de sensores. Muchos de estos dispositivos pueden emplear altos voltajes, así que el módulo de entrada a menudo está optoacoplado. Algunos PLCs pueden tener más de un módulo de entrada.

Módulo de salida Permite la generación de señales eléctricas de unos 110 Volts para controlar dispositivos de salida tales como solenoides, luces, contactores, motores y otros tipos de actuadores. La tensión de salida en realidad puede variar dependiendo del tipo de módulo, así que la tensión podría ser de solo 24 Volts, por ejemplo. Algunos PLCs pueden tener más de un módulo de salida, cada uno con una tensión de salida dada.

Módulo interno Contiene información de tipo lógico que se almacena en alguna memoria volátil y que se emplea para cálculos intermedios. Hay diferentes objetos que se encuentran 'almacenados' en el modulo interno, a saber, relevadores de control, temporizadores, contadores y secuenciadores.

Unidad de procesamiento central (CPU) Lee la información de los distintos módulos de entrada, los procesa con ayuda del módulo interno y publica los resultados por medio del módulo de salida. Emplea principalmente las operaciones lógicas de negación, conjunción y disyunción, pero también puede echar mano de comparaciones tales como 'mayor que' o 'menor o igual que'.

XII.1.1 Diagramas de terminales

Los dispositivos conectados a las terminales de un módulo de entrada o de salida deben estar siempre documentados para poder comprender el funcionamiento del programa que se ejecuta en un dispositivo lógico *programable* (PLC). Esto se debe a que, como se explica en la siguiente sección, el modelo de programación del PLC es meramente 'abstracto' y hay

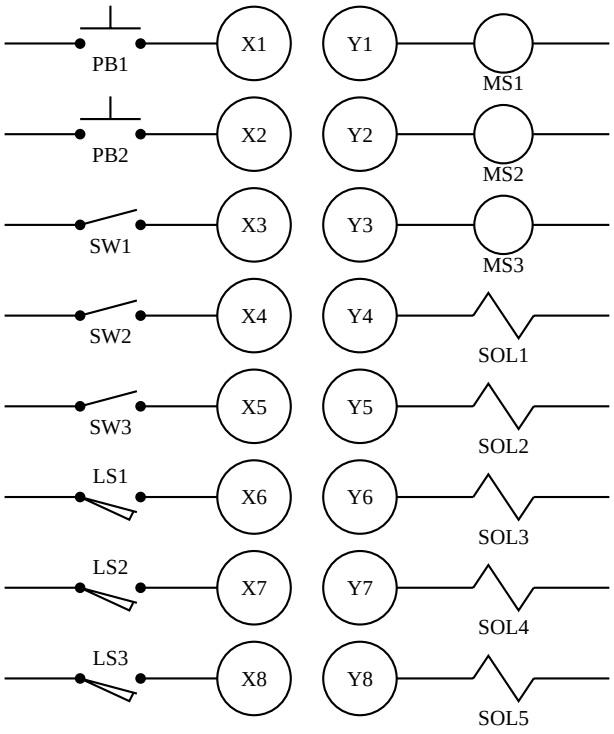


Figura 12.1: Diagrama de terminales para un PLC de 8 entradas y 8 salidas.

una gran diversidad de dispositivos que podrían estar conectados a la entrada de un PLC y seguir ofreciendo la misma entrada binaria: botones enclavados, botones de retorno automático, interruptores manuales, interruptores de límite y muchos otros más. Los diagramas de terminales, ya sean para el módulo de entrada o para el de salida simplemente muestran de manera explícita cuál dispositivo se encuentra conectado a cuál entrada o salida del PLC. En la figura 12.1 se muestra un diagrama de terminales para un PLC de 8 entradas y 8 salidas.

XII.1.2 Diagramas de escalera

Los *diagramas de escalera* son formas gráficas de representar la *lógica de escalera* que fue desarrollada para controlar dispositivos por medio de contactos (también llamados relevadores de control); este nombre se deriva del hecho que los diagramas empleados para describir la lógica de escalera efectivamente tienen la apariencia de una escalera, debido a que están formados por dos líneas verticales en los extremos izquierdo y derecho, que representan a las dos distintas polaridades de un circuito eléctrico, alimentación y tierra, respectivamente; estas líneas son unidas por “escalones” (líneas horizontales) que contienen una colección de contactos en serie o en paralelo seguidos de una bobina (de control), a veces llamada salida. Tradicionalmente estos diagramas se dibujaban en la pantalla de una computadora con código ASCII, y la notación que se empleaba entonces (véase la tabla 12.1 para apreciar la notación para contactos y bobinas) permanece hasta nuestros días. En un diagrama de escalera es posible representar las operaciones lógicas básicas por medio de combinaciones de contactos y bobinas. Además de las operaciones lógicas básicas, es posible representar contadores, temporizadores y secuenciadores, operaciones relacionales (tales como “mayor que” o “menor o igual que”) y operaciones aritméticas.

Notación	Descripción
`--] [--`	Contacto normalmente abierto
`--]/[--`	Contacto normalmente cerrado
`--( )--`	Bobina normalmente abierta
`--(/)--`	Bobina normalmente cerrada

Tabla 12.1: Notación ASCII para contactos y bobinas en diagramas de escalera.

A	B	AB
1	1	1
1	0	0
0	1	0
0	0	0

Tabla 12.2: Tabla de verdad de la conjunción.

Operaciones lógicas básicas

Conjunción La *conjunción* se define como una operación binaria y booleana, cuyo resultado es verdadero solamente en el caso en el cual sus dos 'entradas' (operandos) A y B son verdaderas y falso en cualquier otro caso. La conjunción se denota por yuxtaposición, es decir, la conjunción de A y B se denota por AB. Su tabla de verdad se muestra en la tabla 12.2. En un diagrama de escalera la conjunción se puede representar en un escalón con dos contactos (o combinaciones de contactos para expresiones más elaboradas) en serie como se ilustra en la figura 12.2.

Disyunción La *disyunción* se define como una operación binaria y booleana, cuyo resultado es verdadero en el caso de que alguna de sus dos 'entradas' (operandos) A y B sea verdadera y falso cuando ambas son falsas. La disyunción se denota por medio de la suma, es decir, la conjunción de A y B se denota por $A + B$. Su tabla de verdad se muestra en la tabla 12.3. En un diagrama de escalera la disyunción se puede representar en un escalón con dos contactos (o combinaciones de contactos para expresiones más elaboradas) en paralelo como se ilustra en la figura 12.3.

Negación La *negación* se define como una operación unaria y booleana, cuyos resultados opuestos al de su único operando. La negación se denota por medio de una barra trazada en la parte superior, es decir, la negación de A se denota por $\bar{A}$ (o bien, por $/A$ cuando no es posible emplear una barra en la parte superior, como cuando se representa en código ASCII) Su tabla de verdad se muestra en la tabla 12.4. En un diagrama de

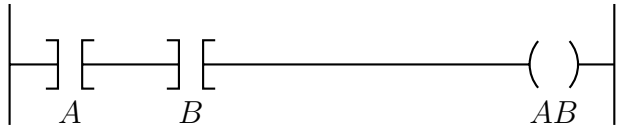


Figura 12.2: Representación de la conjunción AB de los contactos A y B en un diagrama de escalera.

A	B	$A+B$
1	1	1
1	0	1
0	1	1
0	0	0

Tabla 12.3: Tabla de verdad de la disyunción.

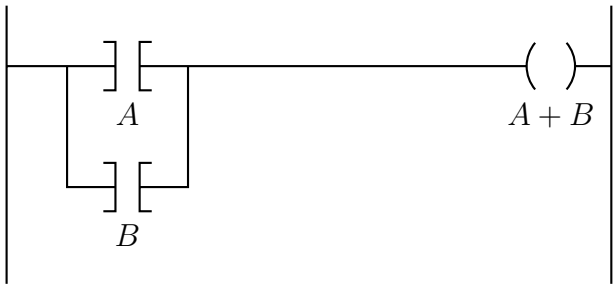


Figura 12.3: Representación de la disyunción $A + B$ de los contactos A y B en un diagrama de escalera.

escalera la negación se puede representar en un escalón con una bobina normalmente cerrada como se ilustra en la figura 12.4.

Registros, contadores, temporizadores y secuenciadores

No todos los escalones de un diagrama de escalera tienen como salida una bobina. En algunos casos el elemento a activar puede ser un objeto propio del PLC (que yace en la memoria del PLC) y que se emplea para instrucciones elaboradas de control, como los que se describen a continuación.

Registros Los registros son valores booleanos como analógicos almacenados en la memoria del PLC, los cuales pueden ser modificados a leer las entradas conectadas al PLC o escribir un valor temporal en la memoria del PLC de forma explícita al emplear el registro como una bobina (de control) en un escalón del diagrama.

Contadores Existen distintas variaciones de los contadores dependiendo del fabricante del PLC, tales como contadores ascendentes (los más comunes) y los descendentes. En cualquier caso, casi todos los contadores se comportan de manera parecida ya que todo contador tiene un valor de referencia (preset en inglés) establece hasta dónde se va a contar, un valor de inicialización (reset en inglés) que indica estén donde se va a contar y un valor acumulador (accumulator en inglés) que registra el número de objetos

A	$\bar{A}$
1	0
0	1

Tabla 12.4: Tabla de verdad de la negación.

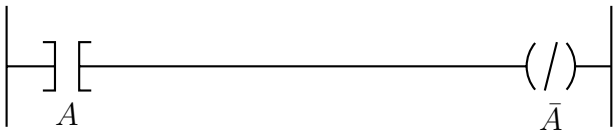


Figura 12.4: Representación de la negación $\bar{A}$ del contacto A en un diagrama de escalera.

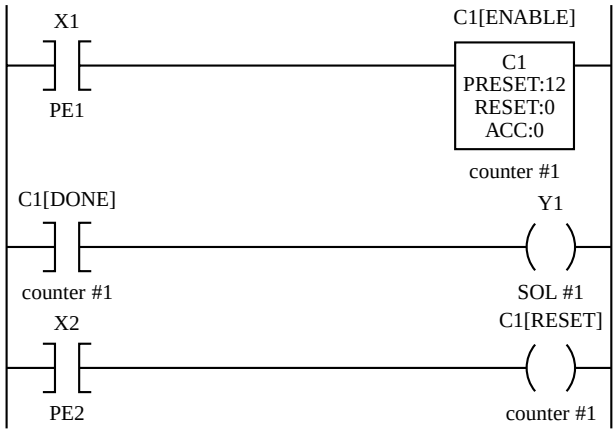


Figura 12.5: Un temporizador en un diagrama de escalera.

contados en un momento dado. Además de estos valores, se emplean tres bits con el objeto de enviar o recibir señales de el temporizador:

Reinicio Para reiniciar la cuenta del contador.

Habilitación Para incrementar la cuenta del contador.

Hecho Para indicar que el valor del acumulador ha alcanzado el valor de referencia.

La figura 12.5 muestra un diagrama escalera que ejemplifica el uso de un contador. En este ejemplo se tiene un contador C1 (contador 1) con un valor de referencia de 12 (debido a que se planea registrar 12 ítems con el mismo), un valor de inicialización de 0 y el acumulador con el valor 0 (lo que significa que el acumulador se inicializa a 0 aún sino se ha recibido la indicación de reinicializar su valor). Cada que el fotodiodo (PE1) conectado a la entrada X1 del PLC detecta un ítem se incrementa el contador por medio del bit de habilitación. Si en algún momento el acumulador del contador llega a ser igual al valor de referencia, entonces el bit de hecho pasar a estado alto y se activa el solenoide 1 conectado a la salida Y1 del PLC. Un segundo fotodiodo (PE2) conectado a la entrada X2 del PLC se encarga de habilitar el bit de reinicio del contador para reiniciar la cuenta.

Temporizadores Un temporizador es como un contador, pero en vez de contar ítems, cuenta unidades de tiempo base (que podrían ser décimas de segundo o milisegundos, según el PLC que se emplee). Los temporizadores, sin embargo, no requieren de un bit de reinicio ya que cuentan sólo cuando el bit de habilitación está en alto y se reinician automáticamente cuando está en bajo.

Secuenciadores Un secuenciador es simplemente una sucesión de valores que se asignan a una colección de salidas, y que se controla paso a paso. Los secuenciadores, sin embargo, carecen de utilidad en la programación de PLC para máquinas herramienta, así que no serán discutidos en este documento.

Operaciones relacionales

Las operaciones relacionales son operaciones binarias y booleanas (es decir, funciones proposicionales de dos argumentos) que regresan el valor verdadero cuando el par de operandos pertenece a la relación binaria, es decir, si R es una relación binaria, entonces $a\ R\ b$ es verdadero sí solo si $(a, b) \in R$. Las seis operaciones relacionales más comunes son:

Igual que Regresa verdadero cuando ambos operandos son iguales.

No igual que Regresa verdadero cuando ambos operandos son distintos.

Menor que Regresa verdadero cuando el primer operando es menor que el segundo.

Menor o igual que Regresa verdadero cuando el primer operando es menor o igual que el segundo.

Mayor que Regresa verdadero cuando el primer operando es mayor que el segundo.

Mayor o igual que Regresa verdadero cuando el primer operando es mayor o igual que el segundo.

Los argumentos estas operaciones relacionales se toman de la memoria del PLC o de alguna entrada analógica; para el caso de nuestro sistema de CNC para máquina herramienta los valores se pueden tomar del componente Memoria base (véase la sección 7.4), donde se almacena toda la información relativa a las variables involucradas en la operación del CNC y es posible almacenar constantes numéricas a través de un programa del PLC.

Operaciones aritméticas

También es posible emplear operaciones aritméticas en lugar de bobinas como salida de un escalón en un diagrama de escalera; las operaciones aritméticas tienen asociadas tres direcciones, a saber, las de los operandos (fuentes) y la del resultado (destino). Así pues, una operación aritmética se ejecutará siempre y cuando la condición requerida en su escalón del diagrama sea verdadera.

XII.2. Componente Controlador lógico

El componente Controlador lógico, cuyo diseño se muestra en la figura 12.6, emplea módulos de entrada, de salida e interno. Tanto el módulo de entrada como el de salida tienen una implementación que depende del equipo físico que se emplee. El módulo interno hace referencia a ubicaciones de memoria en la Memoria base (véase la sección 7.4).

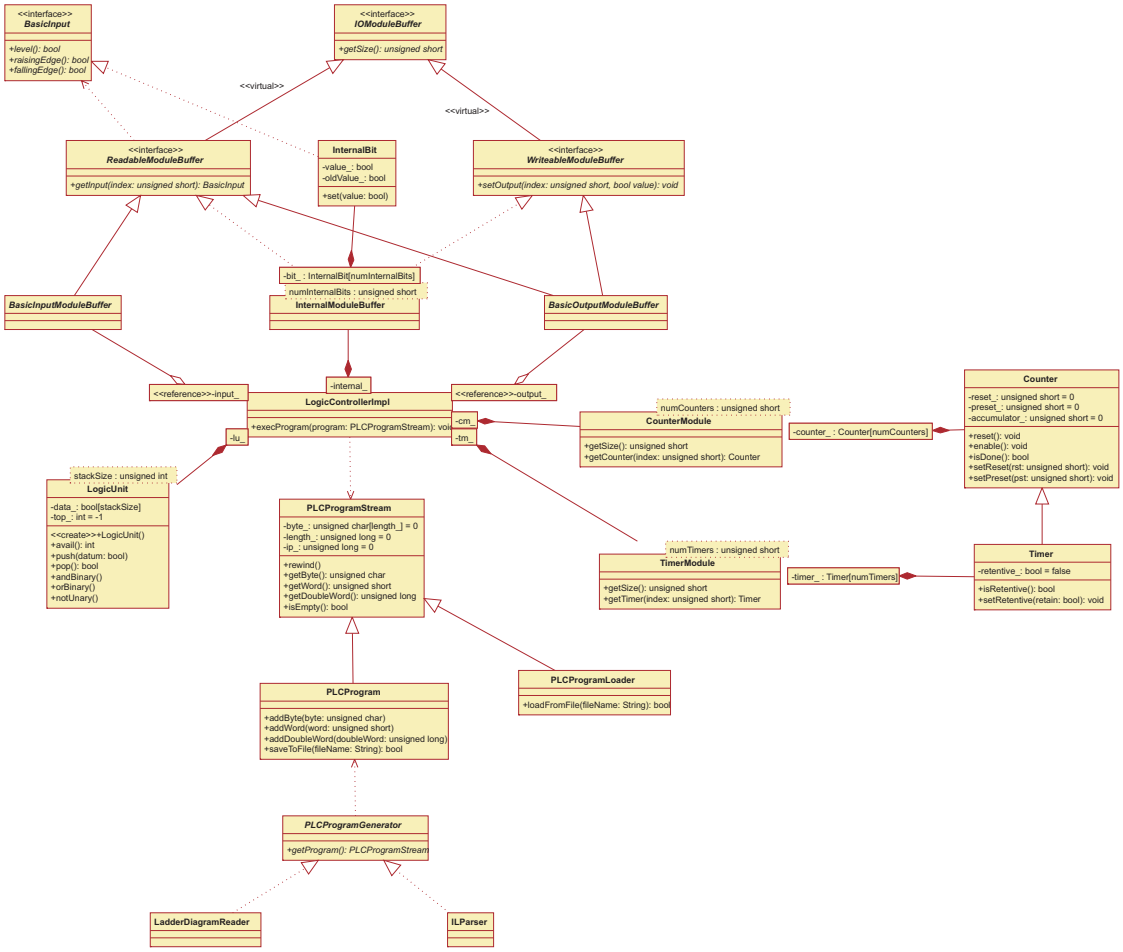


Figura 12.6: Diseño del componente Controlador lógico.

XIII. CONCLUSIONES

La metodología propuesta para desarrollar arquitecturas de control numérico se puede aplicar a partir de cero, pero en muchas situaciones, el resultado conjunto de los componentes puede ser muy similar a la de otras arquitecturas, por ejemplo, el conjunto de componentes puede ser casi idéntica para una fresadora que para un torno de control numérico; por lo tanto, este enfoque permite a los desarrolladores de adquirir experiencia mientras se implementan diversos sistemas de control numérico.

El centro de atención se desplaza de un conjunto estandarizado de componentes que ya han sido diseñados por un comité a la estandarización de un proceso de desarrollo que permite a los diseñadores y desarrolladores aprender sobre la marcha y mejorar continuamente sus técnicas.

Las técnicas de plantillas (o genéricos) permiten a los desarrolladores tener una base de código común para componentes parecidos utilizados en diferentes arquitecturas. Del mismo modo, las técnicas de plantillas también pueden permitir la creación de arquitecturas similares con ligeras variaciones, es decir, si una arquitectura simplemente implementa más casos de uso, entonces sólo hay la necesidad de añadir más módulos, y más interfaces o ampliar las interfaces, pero puede no requerir un rediseño mayor.

Del mismo modo, al ofrecer componentes diseñados especialmente para proporcionar portabilidad de una plataforma a otra, el procedimiento necesario para portar el sistema de control numérico a una arquitectura diferente es bastante concreto. Además, esta arquitectura basada en componentes se puede utilizar para desarrollar una arquitectura distribuida, si se emplea una tecnología de componentes tal como CORBA o DCOM.

A. TEORÍA DE COMPILADORES

A.1. Estructura de un compilador

A grandes rasgos un compilador es un programa que lee un conjunto de instrucciones, llamado *programa fuente*, y lo convierte a otro conjunto de instrucciones equivalente, llamado *programa objeto* (véase la figura A.1); dentro del proceso de conversión una de las

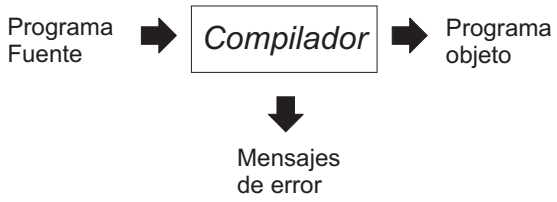


Figura A.1: Un compilador.

funciones principales a realizar por el compilador es informar al usuario la presencia de errores en el programa fuente.

Un compilador está compuesto de ocho módulos, seis de los cuales son fases que, desde el punto de vista conceptual, se agrupan en dos partes, el *análisis* y la *síntesis*, formadas por tres módulos cada una. Los dos módulos restantes interactuan con los demás durante todo el proceso de compilación, como se muestra en la figura A.2. El análisis es el encargado de descomponer el programa fuente en partes más simples y crea una representación intermedia del programa fuente. Durante el análisis se determinan las operaciones que indica el programa fuente y se registran en alguna estructura de datos, usualmente en un *árbol sintáctico*, donde cada nodo representa una operación y los hijos de un nodo son los argumentos de una operación; el código G es tan sencillo que no requiere de una estructura de datos tan compleja y basta con emplear una lista de palabras para representar cada bloque. El análisis consta de tres fases: el análisis lineal o léxico, el análisis jerárquico o sintáctico y el análisis semántico.

Análisis lineal o léxico En esta fase se lee el programa fuente de izquierda a derecha y se agrupa en componentes léxicos, llamados *símbolos*, los cuales son secuencias de caracteres con un significado colectivo. En esta fase normalmente se eliminan los espacios en blanco que hay en el programa fuente.

Análisis jerárquico o sintáctico Esta parte agrupa los símbolos del programa fuente en frases gramaticales que el compilador utilizará para sintetizar la salida. Generalmente estas frases gramaticales se representan mediante un árbol de análisis sintáctico, aunque en nuestro caso emplearemos simplemente una lista de palabras por cada bloque. La estructura jerárquica de un programa se expresa utilizando reglas recursivas llamadas gramáticas independiente del contexto.

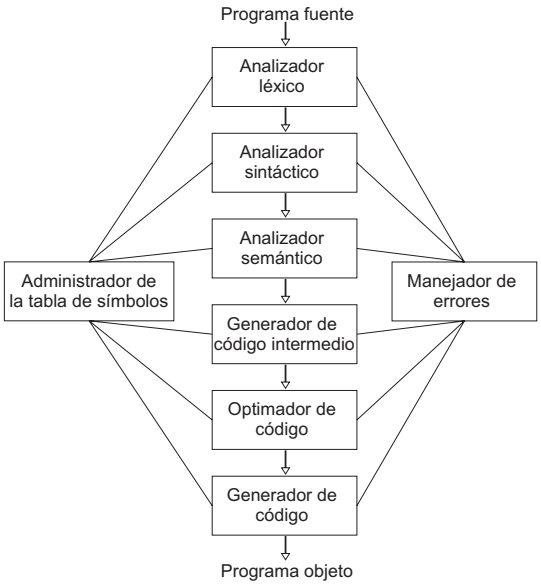


Figura A.2: Fases conceptuales de un compilador

Análisis semántico En esta fase se hace una revisión del programa fuente en busca de errores que pudieran existir en él; además de reunir información sobre los símbolos para fases posteriores.

La síntesis es la encargada de construir el programa objeto a partir de la representación intermedia generada durante el análisis y sus fases son: el generador de código intermedio, el optimizador de código y el generador de código.

Generador de código intermedio Se puede considerar este código intermedio como un programa para una máquina abstracta. Esta representación intermedia debe ser tal que sea fácil de producir y fácil de traducir al programa objeto.

Optimización de código El propósito de esta fase es generar un código intermedio más fácil y rápido de ejecutar.

Generación de código Esta fase es la última, es la que genera el programa objeto que por lo general es código ensamblador o código de máquina relocalizable.

Los dos módulos que interactúan con el análisis y síntesis son el administrador de la tabla de símbolos y el manejador de la tabla de errores. La principal función de estos dos módulos es recolectar información, ya sea de los errores generados a lo largo del proceso o de los atributos de las variables y funciones involucradas en cada una de las representaciones intermedias de las fases.

Manejador de errores Debido a que un compilador que se detiene en cada error que se encuentra no es tan útil, se emplea un manejador de errores. En esta fase se genera una tabla de errores de todas las fases del compilador y sirve para notificarle al usuario dónde está hay errores en el código fuente.

Símbolo	Lexemas de ejemplo	Patrón
entero sin signo	`110, 54`	cadena de dígitos
dirección	`G, M, N`	letra del alfabeto inglés
comentario	`(DIA. 1MM), (PROG. 2)`	texto entre paréntesis

Tabla A.1: Ejemplos de símbolos, los patrones con los que concuerdan sus lexemas.

Administrador de la tabla de símbolos Puesto que una de las funciones fundamentales del compilador es registrar los identificadores utilizados en el programa fuente y reunir información sobre los distintos atributos de cada identificador, se crea otra tabla que maneja toda esta información en el transcurso del proceso de compilación.

Sin embargo, existen otras maneras de agrupar estas fases. En la practica, una manera de hacerlo es agrupando las fases, o partes de fases, en dos etapas, a saber:

Etapa frontal Agrupa a las fases que tienen ver con el programa fuente y son independientes del programa objeto. Está formada principalmente por el análisis léxico y sintáctico, la creación de la tabla de símbolos, el análisis semántico y la generación de código intermedio. Aquí también se puede hacer cierta optimización de código y se manejan los errores correspondientes a cada una de las fases involucradas en esta etapa.

Etapa posterior Agrupa a las fases que tienen que ver con la generación el programa objeto a partir, no del lenguaje fuente sino, del lenguaje intermedio. Se encuentran involucradas las fases de optimización de código y la generación de código junto con le manejo de errores necesarios y las operaciones con la tabla de símbolos correspondientes.

A.2. Análisis léxico

El análisis léxico consiste en leer un flujo de caracteres de entrada y verificar si coinciden con algún *patrón* dado de una colección predeterminada; aquellas cadenas que coinciden con alguno de los patrones válidos forman un conjunto llamado *símbolo léxico* (o, simplemente, *símbolo*) y la cadena concreta se denomina *lexema*. En la tabla A.1 se muestran algunos ejemplos.

A.2.1 Lenguajes

Usualmente para la descripción de un patrón no se emplea una descripción verbal como en el ejemplo de la tabla A.1 ya que usualmente una descripción de este tipo no es suficientemente precisa. En su lugar se emplean las expresiones regulares que definiremos a continuación.

Cualquier conjunto finito de caracteres se denominará *alfabeto*. Una *cadena de caracteres* (o, simplemente, *cadena*) es una sucesión finita de símbolos tomados de algún alfabeto. Si s es una cadena, denotamos su longitud por $|s|$. Por ejemplo,

$$|\texttt{roberto}| = 7.$$

La cadena de longitud cero, que será denotada por ε, se llama *cadena vacía*. Se dice que un *lenguaje* es un conjunto de cadenas sobre cualquier alfabeto. Si x y y son dos cadenas, su concatenación xy es la cadena formada por los caracteres de x seguidos de los caracteres de y.

Por ejemplo, si $x = \texttt{agua}$ y $y = \texttt{fiestas}$ entonces $xy = \texttt{aguafiestas}$. La cadena ε es el elemento identidad para la concatenación, es decir, $s\varepsilon = \varepsilon s = s$ para cualquier cadena s.

Si L y M son lenguajes, entonces podemos su *concatenación* como

$$LM = \{st | s \in L, t \in M\}.$$

Podemos definir la *exponenciación* de un lenguaje como

$$L^n = \{s_1 s_2 \cdots s_n | s_1, s_2, \ldots, s_n \in L\}.$$

También de gran utilidad son la *cerradura de Kleene* de un lenguaje L,

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

y su *cerradura positiva*

$$L^+ = \bigcup_{i=1}^{\infty} L^i.$$

A.2.2 Expresiones regulares

A partir de la concatenación es posible definir las *expresiones regulares sobre un alfabeto* Σ de manera recursiva:

1. ε es una expresión regular que denota al conjunto $\{\varepsilon\}$, es decir, al conjunto que contiene a la cadena vacía.

2. Si $a \in \Sigma$, entonces a es una expresión regular que representa al conjunto $\{a\}$, es decir, al conjunto que contiene a la cadena a. Aunque usamos la misma notación para las tres, técnicamente, la expresión regular a es distinta de la cadena a y del carácter a.

3. Supongamos que r y s son expresiones regulares que denotan a los lenguajes $L(r)$ y $L(s)$, respectivamente. Entonces

   a) $(r)|(s)$ es una expresión regular que denota al lenguaje $L(r) \cup L(s)$.

   b) $(r)(s)$ es una expresión regular que denota al lenguaje $L(r)L(s)$.

   c) $(r)^*$ es una expresión regular que denota al lenguaje $(L(r))^*$.

   d) (r) es una expresión regular que denota al lenguaje $L(r)$, es decir, $(r) = r$.

Además será de utilidad introducir algunas notaciones: $(r)^n$ es una expresión regular que denota al lenguaje $(L(r))^n$ y $(r)^+$ es una expresión regular que denota al lenguaje $(L(r))^+$.

Ejemplo A.1 Sea $\Sigma = \{a, b\}$.

1. La expresión regular $a|b$ denota al conjunto $\{a, b\}$.

2. La expresión regular $(a|b)^2$ denota al conjunto $\{aa, ab, ba, bb\}$, el conjunto de las cadenas de longitud 2 formadas por a's y b's.

3. La expresión regular a^* denota al conjunto de cadenas de cero o más a's, es decir,

$$\{\varepsilon, a, aa, aaa, \ldots\}.$$

4. La expresión regular $(a|b)^*$ denota al conjunto de todas las cadenas que contienen cero o más veces las letras a y b, es decir, el conjunto de las cadenas formadas con las letras a y b. Otra expresión regular para este mismo conjunto es $(a^*b^*)^*$.

5. La expresión regular $a|a^*b$ denota al conjunto que contiene la cadena a y todas las cadenas formadas por cero o más a's seguidas de una b.

Se dice que dos expresiones regulares que representan al mismo lenguaje son *equivalentes*.

A.2.3 Autómatas finitos

Un *reconocedor* de un lenguaje es un programa que toma por entrada una cadena x y determina si una cadena pertenece a un lenguaje o no. Se puede crear un reconocedor por medio de un diagrama de transiciones llamado a autómata finito. Un autómata finito puede ser no determinístico o determinístico dependiendo si más de una transición puede salir de un estado al leer un mismo carácter de entrada. Ambos tipos de autómatas pueden emplearse para programar código par aun reconocedor, pero siempre hay un compromiso entre velocidad y tamaño del programa resultante: los autómatas determinísticos pueden ser más rápidos que los no determinísticos, pero generan programas más grandes.

Un *autómata finito no determinístico* (AFND) es un modelo matemático que consiste de

1. un conjunto S cuyos elementos se llaman *estados*,

2. un conjunto de caracteres de entrada Σ llamado *alfabeto de entrada*,

3. una *función de transición* que asigna pares formados por un estado y un símbolo de entrada (que puede ser un carácter del alfabeto o ε) a un conjunto de estados,

4. un estado s_0 considerado el *estado inicial* y

5. un conjunto de estados F considerados *estados finales*.

Un AFND se puede representar por medio de una gráfica dirigida etiquetada llamada *gráfica de transiciones* en la cual los nodos son los estados y las aristas etiquetadas con un símbolo de entrada representan a la función de transición. Un mismo carácter puede etiquetar a más de una arista que sale de un estado, y algunas aristas pueden estar etiquetadas con ε; cuando una arista esté etiquetada con ε se dirá que la transición correspondiente es una transición ε. Los estados finales se indican con nodos con doble círculo.

Ejemplo A.2 La gráfica de transiciones para un AFND que reconoce al lenguaje $(a|b)^*abb$ se muestra en la figura A.3. El conjunto de estados del AFND es $S = \{0, 1, 2, 3\}$ con $s_0 = 0$ y $F = \{3\}$. El alfabeto de entrada es $\Sigma = \{a, b\}$. La función de transición se describe en la tabla A.3.

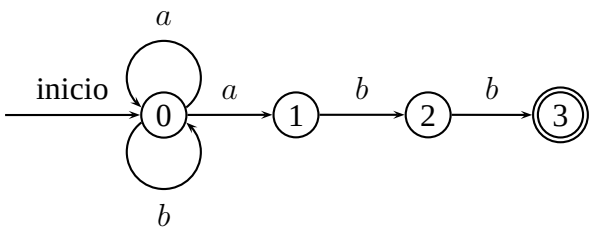


Figura A.3: Gráfica de transiciones para un AFND.

	Símbolo de entrada	
Estado	a	b
0	$\{0, 1\}$	$\{0\}$
1	–	$\{2\}$
2	–	$\{3\}$

Tabla A.3: Función de transición para el AFND de la figura A.3.

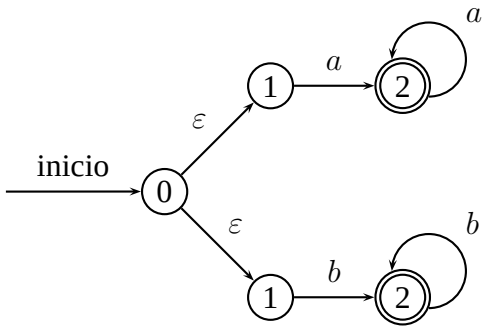


Figura A.4: AFND para la expresión regular $aa^*|bb^*$.

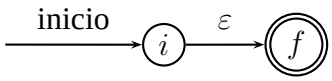
	Símbolo de entrada		
Estado	ε	a	b
0	$\{1, 3\}$	—	–
1	–	$\{2\}$	–
2	–	$\{2\}$	–
3	–	–	$\{4\}$
4	–	–	$\{4\}$

Tabla A.5: Función de transición para el AFND de la figura A.4.

Ejemplo A.3 El AFND de la figura A.4 reconoce al lenguaje dado por la expresión regular $aa^*|bb^*$. La función de transición se describe en la tabla A.5.

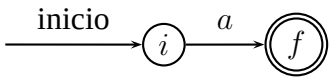
Existe un algoritmo recursivo, llamado la *construcción de Thompson*, que nos permite generar a partir de una expresión regular r un AFND que acepte al lenguaje $L(r)$ que corresponde a esa expresión regular. Consiste simplemente en aplicar las reglas 1 y 2 para cada uno de los símbolos básicos de la expresión regular y luego emplear la regla 3 recursivamente:

1. Para ε, se construye el AFND



   En este caso i es un nuevo estado inicial y f es un nuevo estado final. Claramente, este autómata finito reconoce al lenguaje $\{\varepsilon\}$.

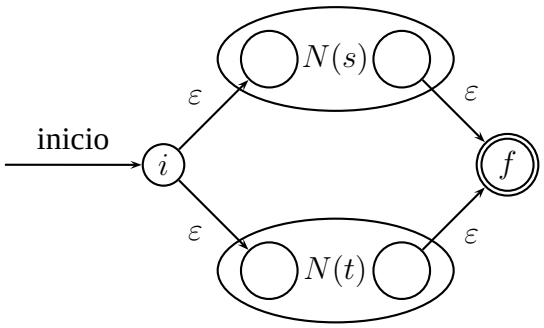
2. Para $a \in \Sigma$, se construye el AFND



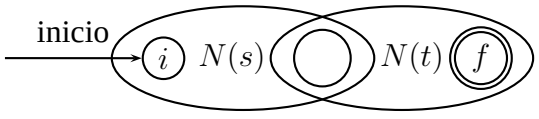
   Nuevamente, i es un nuevo estado inicial y f es un nuevo estado final. Este autómata finito reconoce al lenguaje $\{a\}$.

3. Supongamos que $N(s)$ y $N(t)$ son los AFNDs para las expresiones regulares s y t.

   a) Para la expresión regular $s|t$ se construye el siguente AFND $N(s|t)$:



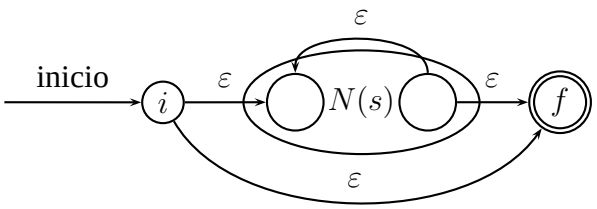
   b) Para la expresión regular st se construye el siguiente AFND $N(st)$:



   c) Para la expresión regular s^* se construye el siguiente AFND $N(s^*)$:

$\Delta \leftarrow \{\varphi_\varepsilon(s_0)\}$
mientras haya un estado no marcado $T \in \Delta$ **hacer lo siguiente**
marcar T
para cada símbolo de entrada a **hacer lo siguiente**
$U \leftarrow \varphi_\varepsilon(\tau(T, a))$
si $U \notin \Delta$ **entonces**
agregar U como un estado no marcado a Δ
$\delta(T, a) \leftarrow U$
fin para
fin mientras

Figura A.5: Creación de un AFD a partir de un AFND.



d) Para la expresión regular (s) entre paréntesis se emplea el mismo AFND $N(s)$ que se usa para la expresión sin paréntesis s.

Un *autómata finito determinístico* (AFD) es un caso especial de uno no determinístico en el cual

1. ningún estado tiene una transición ε, es decir, una transición ante la entrada ε, y

2. para cada estado s y carácter de entrada a, a lo más hay una arista etiquetada con a que sale de s.

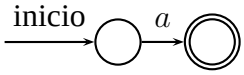
En otras palabras, un AFD a lo más tiene una transición a partir de cada estado para cada carácter de entrada.

Es más fácil programar un AFD que un AFND, así que resulta de gran interés poder convertir cualquier AFND en un AFD. La idea básica de esta conversión es que cada estado del AFD corresponde a un conjunto de estados del AFND, así que, en general, el AFD tendrá más estados de el AFND.

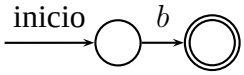
Debemos eliminar las aristas etiquetadas con ε (es decir, las transiciones ε) y evitar que haya más de una arista etiquetada con el mismo carácter. Para tal fin introducimos algunas definiciones. El conjunto $\varphi_\varepsilon(s)$ contiene a los estados a los que se puede llegar a partir del estado s empleando únicamente transiciones ε. Asimismo, el conjunto $\varphi_\varepsilon(T)$ contiene a los estados a los que se puede llegar a partir de un estado $s \in T$ del AFND empleando únicamente transiciones ε. El conjunto $\tau(T, a)$ contiene a los estados hacia los cuales hay una transición para el carácter de entrada a a partir de un estado $s \in T$ del AFND.

Podemos crear un AFD a partir del AFND para el mismo alfabeto de entrada Σ con un conjunto de estados D y una función de transición δ de la manera según el algoritmo de la figura A.5. Un estado del AFD será un estado final si contiene a alguno de los estados finales del AFND.

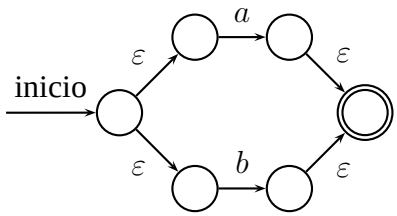
Ejemplo A.4 Podemos construir un AFND para reconocer al lenguaje definido por la expresión regular $(a|b)^*abb$. Para la expresión regular a construimos



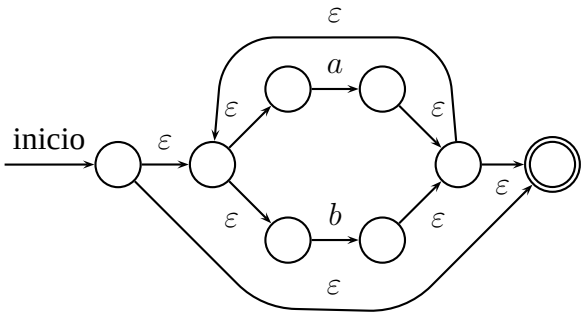
y para la expresión regular b construimos



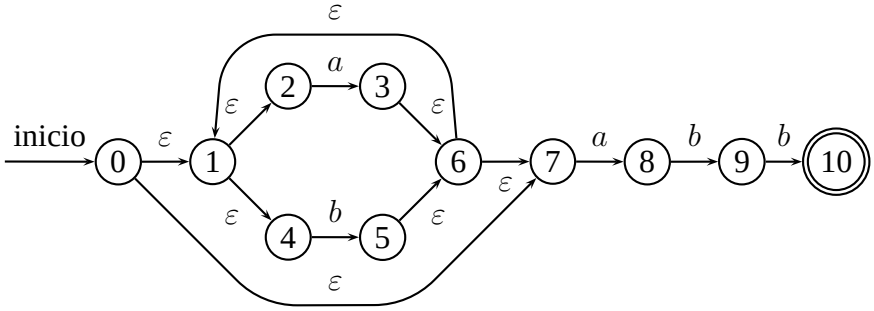
Entonces, para la expresión regular $a|b$ tenemos



de modo que la expresión regular para $(a|b)^*$ es



Por consiguiente, el AFND correspondiente a la expresión regular $(a|b)^*abb$ es



Es posible emplear el algoritmo de la figura A.5 para transformar este AFND en un AFD para el mismo alfabeto $\Sigma = \{a, b\}$. El estado inicial del AFD será

$$A = \varphi_\varepsilon(0) = \{0, 1, 2, 4, 7\}.$$

Se marca A y se calculan, para a y para b,

$$\varphi_\varepsilon(\tau(A, a)) = \varphi_\varepsilon(\tau(\{2, 7\}, a)) = \varphi_\varepsilon(\{3, 8\}) = \{1, 2, 3, 4, 6, 7, 8\},$$
$$\varphi_\varepsilon(\tau(A, b)) = \varphi_\varepsilon(\tau(\{4\}, b)) = \varphi_\varepsilon(\{5\}) = \{1, 2, 4, 5, 6, 7\},$$

respectivamente; así pues, se asignan los nombres

$$B = \{1, 2, 3, 4, 6, 7, 8\}, \quad C = \{1, 2, 4, 5, 6, 7\},$$

se agregan como estados no marcados a Δ y se establece

$$\delta(A, a) \leftarrow B, \quad \delta(A, b) \leftarrow C.$$

Se marca B y se calculan, para a y para b,

$$\varphi_\varepsilon(\tau(B, a)) = \varphi_\varepsilon(\tau(\{2, 7\}, a)) = \varphi_\varepsilon(\{3, 8\}) = \{1, 2, 3, 4, 6, 7, 8\} = B,$$
$$\varphi_\varepsilon(\tau(B, b)) = \varphi_\varepsilon(\tau(\{4, 8\}, b)) = \varphi_\varepsilon(\{5, 9\}) = \{1, 2, 4, 5, 6, 7, 9\},$$

respectivamente; en consecuencia, se asigna el nombre

$$D = \{1, 2, 4, 5, 6, 7, 9\},$$

se agrega como estado no marcado a Δ y se establece

$$\delta(B, a) \leftarrow B, \quad \delta(B, b) \leftarrow D.$$

Se marca C y se calculan, para a y para b,

$$\varphi_\varepsilon(\tau(C, a)) = \varphi_\varepsilon(\tau(\{2, 7\}, a)) = \varphi_\varepsilon(\{3, 8\}) = \{1, 2, 3, 4, 6, 7, 8\} = B,$$
$$\varphi_\varepsilon(\tau(C, b)) = \varphi_\varepsilon(\tau(\{4\}, b)) = \varphi_\varepsilon(\{5\}) = \{1, 2, 4, 5, 6, 7\} = C,$$

respectivamente; por lo tanto, se establece

$$\delta(C, a) \leftarrow B, \quad \delta(C, b) \leftarrow C.$$

Se marca D y se calculan, para a y para b,

$$\varphi_\varepsilon(\tau(D, a)) = \varphi_\varepsilon(\tau(\{2, 7\}, a)) = \varphi_\varepsilon(\{3, 8\}) = \{1, 2, 3, 4, 6, 7, 8\} = B,$$
$$\varphi_\varepsilon(\tau(D, b)) = \varphi_\varepsilon(\tau(\{4, 9\}, b)) = \varphi_\varepsilon(\{5, 10\}) = \{1, 2, 4, 5, 6, 7, 10\},$$

respectivamente; en consecuencia, se asigna el nombre

$$E = \{1, 2, 4, 5, 6, 7, 10\},$$

se agrega como estado *final* (contiene al estado 10) no marcado a Δ y se establece

$$\delta(D, a) \leftarrow B, \quad \delta(D, b) \leftarrow E.$$

Se marca E y se calculan, para a y para b,

$$\varphi_\varepsilon(\tau(E, a)) = \varphi_\varepsilon(\tau(\{2, 7\}, a)) = \varphi_\varepsilon(\{3, 8\}) = \{1, 2, 3, 4, 6, 7, 8\} = B,$$
$$\varphi_\varepsilon(\tau(E, b)) = \varphi_\varepsilon(\tau(\{4\}, b)) = \varphi_\varepsilon(\{5\}) = \{1, 2, 4, 5, 6, 7\} = C$$

y se establece

$$\delta(D, a) \leftarrow B, \quad \delta(D, b) \leftarrow E.$$

La función de transición δ se resume en la tabla A.7 y el AFD se representa en la figura A.6.

Estado	Símbolo de entrada	
	a	b
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

Tabla A.7: Función de transición para el AFND de la figura A.3.

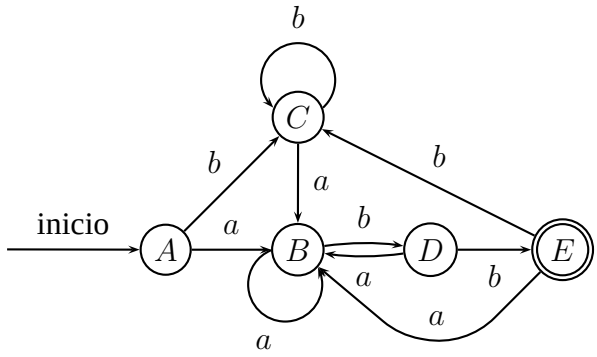


Figura A.6: AFD correspondiente a la función de transición de la tabla A.7.

.

A.3. Analizadores sintácticos

Una vez que es posible reconocer los diferentes símbolos léxicos de un lenguaje, el siguiente paso es verificar que dichos símbolos léxicos se encuentren en el orden correcto. Por ejemplo, en el código G se espera que después de cada dirección haya un número, ya sea entero o de punto flotante; asimismo, se espera que al inicio del programa se indique el número del programa. Para describir de manera precisa y recursiva la sintaxis de un programa se emplea una gramática. Afortunadamente, la gramática necesaria para describir un programa de código G es bastante simple y cae en la categoría llamada gramática independiente del contexto porque el significado de los símbolos léxicos no depende del contexto en que se encuentren, es decir, de otros símbolos léxicos que los rodeen.

Una *gramática independiente del contexto* es un modelo matemático formado por cuatro componentes:

1. Un conjunto T de símbolos léxicos, conocido como símbolos *terminales*.

2. Un conjunto N de *símbolos no terminales* tal que $N \cap T = \emptyset$.

3. Un subconjunto finito P de $N \times (N \cup T)^*$, cuyos elementos se denominan *producciones*.

4. Un *símbolo inicial* $\sigma \in N$.

Si G es la gramática (libre de contexto) formada por estos elementos entonces se escribe $G = (N, T, P, \sigma)$.

Las producciones de una gramática usualmente no se escriben como un par ordenado, como es lo estándar para un producto cartesiano, sino que en vez de usar la notación (A, B) se escribe

$$A \to B$$

para indicar que A *produce* B; también se dice que B *se deriva de* A. Además, si existen n producciones de la forma $A \to B_1, \ldots, A \to B_n$ entonces usualmente se escribe

$$A \to B_1 | \cdots | B_n.$$

Ejemplo A.5 Sea $G = (N, T, P, \sigma)$ una gramática con $N = \{\sigma, \alpha, \beta\}$, $T = \{a, b\}$ y P formado por

$$\begin{aligned} \sigma &\to a\beta | b\alpha, \\ \alpha &\to a | a\sigma | b\alpha\alpha, \\ \beta &\to b | b\sigma | a\beta\beta. \end{aligned}$$

Se puede demostrar que esta gramática describe al lenguaje formado por las cadenas que tienen igual número de los terminales a y b. Por ejemplo, la cadena $aabbab$ pertenece a este conjunto pues

$$\sigma \Rightarrow a\beta \Rightarrow aa\beta\beta \Rightarrow aab\beta \Rightarrow aabb\sigma \Rightarrow aabba\beta \Rightarrow aabbab,$$

donde '$\Rightarrow$' significa que hay una producción que permite hacer el reemplazo.

Una *gramática regular* es una gramática independiente del contexto cuyas producciones tienen alguna de las formas siguientes:

1. El lado derecho es la cadena vacía:

$$\alpha \to \varepsilon.$$

2. El lado derecho es un único símbolo terminal:

$$\alpha \to a.$$

3. El lado derecho es un único símbolo terminal seguido de un único símbolo no terminal:

$$\alpha \to a\beta.$$

Las gramáticas regulares tienen la propiedad de que se pueden expresar por medio de expresiones regulares. En efecto, existe una correspondencia biunívoca entre expresiones regulares y gramáticas regulares:

1. A una producción de la forma

$$\alpha \to \varepsilon$$

le corresponde la expresión regular ε.

2. A las producciones de la forma

$$\alpha \rightarrow a$$

se les asigna la expresión regular a.

3. Si hay producciones de la forma

$$\alpha \rightarrow \beta_1, \quad \ldots, \quad \alpha \rightarrow \beta_n,$$

se les asocia la expresión regular $\hat{\beta}_1|\cdots|\hat{\beta}_n$, donde $\hat{\beta}_i$ es la expresión regular que corresponde a β_i.

4. A una producción como

$$\alpha \rightarrow a\beta$$

se le asigna la expresión $a\hat{\beta}$, donde $\hat{\beta}$ es la expresión regular que corresponde a β.

5. Una producción de la forma

$$\alpha \rightarrow a\alpha$$

corresponde a la expresión regular a^*.

Las gramáticas regulares permiten simplificar en gran medida la creación de analizadores sintácticos debido a que es posible asociarles un AFD.

A.4. Implementación de máquinas de estado

Es fácil programar una AFD en lenguaje C/C++ empleando la sentencia para el flujo del control `switch`. Por ejemplo, para implementar un bloque de código que modele el AFD de la figura A.6 se puede declarar una enumeración

```
enum State {A, B, C, D, E};
```

y posteriormente declarar una variable

```
enum State state = A;
```

para controlar el ciclo

```
while (state != E) {
        switch (state) {
        case A:
                switch (getChar()) {
                case 'a':
                        state = B;
                        break;
                case 'b':
                        state = C;
                        break;
                }
                break;
```

```
        case B:
                switch (getChar()) {
                case 'a':
                        state = B;
                        break;
                case 'b':
                        state = D;
                        break;
                }
                break;
        case C:
                switch (getChar()) {
                case 'a':
                        state = B;
                        break;
                case 'b':
                        state = C;
                        break;
                }
                break;
        case D:
                switch (getChar()) {
                case 'a':
                        state = B;
                        break;
                case 'b':
                        state = E;
                        break;
                }
                break;
        }
}
```

donde la función `getChar` extrae caracteres de algún flujo de entrada.

A.5. Forma Backus-Naur

Una forma muy común de expresar gramáticas libres de contexto es usando la forma Backus-Naur creada por John Backus y Peter Naur. En esta representación los símbolos no terminales se representan entre paréntesis angulares y los símbolos terminales se representan por sí mismos, es decir, como ⟨no terminal⟩ y **terminal**. Es común separar el lado izquierdo de la producción del lado derecho por medio del símbolo ::= en vez de →, principalmente cuando el símbolo → no está disponible en el conjunto de caracteres.

B. MONITORIZACIÓN Y CONTROL DEL PROCESO DE FRESADO SIN SENSORES

B.1. Justificación

Un Sistema de Manufactura Inteligente (SMI) es un sistema cuyo objetivo principal es optimizar y automatizar uno o más pasos del proceso de manufactura por medio de un comportamiento inteligente—que se define como un proceso de toma de decisiones que imita al de un humano en bajo las mismas circunstancias—. Un componente autónomo debe ser tolerante a fallas con el fin de reducir al mínimo su interacción con otros componentes del sistema cuando ocurren fallas menores y debe optimizar el rendimiento del componente con el fin de aumentar la eficiencia global del sistema.

En el caso de procesos CNC hay diversos factores que son buenos candidatos para optimización, tales como tiempos de ciclo, la vida de la herramienta, tiempos de inactividad, etc. Si bien existen factores tales como tiempos de inactividad relacionados con la disponibilidad de materias primas, la demanda de productos y otras condiciones de gestión de la planta, hay una clase de problemas de optimización de procesos, tales como la duración y el tiempo de ciclo de vida útil herramienta, que dependen de relaciones sutiles entre las velocidades, la profundidad y las fuerzas de corte y de la disponibilidad de algoritmos adecuados para hacer frente a estas variables de proceso.

Hay varias opciones disponibles para obtener o calcular estas variables de proceso, dependiendo del factor a optimizar (Liang et al., 2004). Sin embargo, los métodos que requieren instrumentos especiales para medir las variables del proceso (por ejemplo, dinamómetros (Altintas, 1992)), aumentan los costos de producción debido a los tiempos de inactividad necesarios para la instalación y por el costo de la instrumentación del proceso en sí. Por esta razón, hay una serie de trabajos que proponen un enfoque sin sensores menos costoso; por ejemplo, esta técnica se ha aplicado con éxito para calcular la retroalimentación de velocidad para actuadores (Bharadwaja et al., 2004; Wang y Liu, 2006) y para detectar fallas mecánicas en los sistemas electromecánicos (Parlos et al., 2004). También se ha utilizado para detectar los daños en herramientas bajo diferentes condiciones (Romero Troncoso et al., 2003; Franco Gasca et al., 2006).

Sin embargo, a pesar que se trata de un área de investigación activa, no existe un modelo de referencia que sirva como base para probar diferentes algoritmos y estrategias, lo que lleva a la repetición de gran parte del trabajo.

Este documento propone un modelo de referencia de software que puede ser aplicado a diferentes configuraciones de hardware y permite al investigador probar diferentes algoritmos sin reimplementar la mayor parte del sistema. El marco de referencia para monitorización propuesto se basa en un enfoque sin sensores y una PC como sistema de adquisición y procesamiento de datos como se muestra en la figura B.1. El modelo de referencia

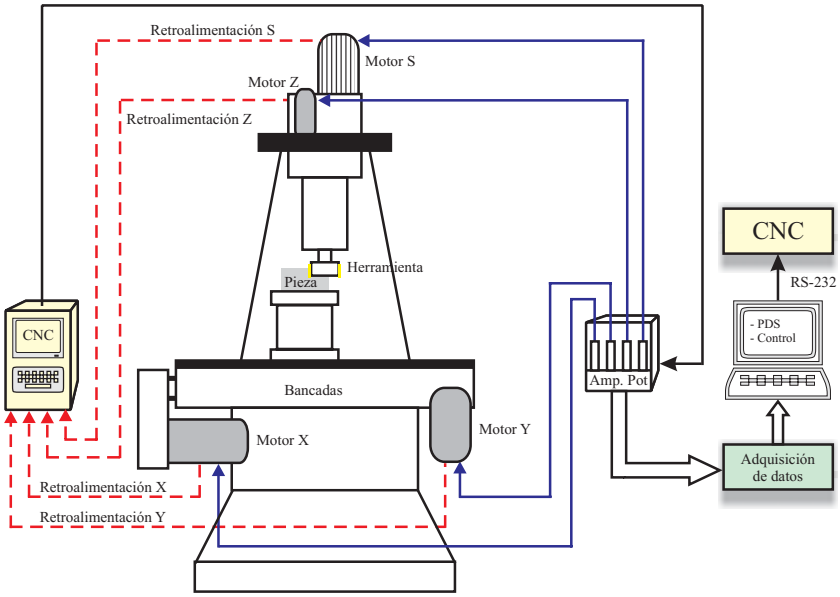


Figura B.1: Sistema de monitorización y control de procesos de fresado.

está bien descrito por medio de el Lenguaje Unificado de Modelado (UML) (Fowler, 2003) ya que su diseño orientado a objetos (OOD) se basa en patrones de diseño comunes (Gamma et al., 1994).

B.2. Control inteligente del proceso de fresado sin sensores

Uno de los principales objetivos de una unidad de control inteligente es hacer que el proceso de maquinado sea más eficiente mediante la optimización de las condiciones de corte para el sistema de CNC y la detección de fallas. Las fuerzas de corte son uno de los principales aspectos a considerar en los procesos de fresado ya que las mismas determinan los requerimientos de potencia de corte—las fuerzas de corte excesivas puede causar altas temperaturas por fricción o vibraciones inestables, lo que socava la vida de la herramienta, mientras que, por otra parte, las fuerzas de corte innecesariamente bajas aumentan el tiempo de ciclo y reducen la productividad global—. Los programadores de CNC pueden seleccionar diferentes fuerzas de corte configurando la velocidad de husillo, la velocidad y la profundidad de corte, pero esto requiere conocimientos de maquinado que toman tiempo en aprenderse (Smid, 2003); cuando se realizan procesos de maquinado la geometría de la herramienta y los materiales tienden a cambiar en cada caso, y el programador de CNC debe tener en cuenta estos factores, junto con las especificaciones especiales de la máquina de CNC. La mayoría de las veces el programador de CNC decide establecer valores conservadores de los parámetros del maquinado para hacer frente a las diferentes arquitecturas de máquina herramienta, dependiendo de que el operador de la máquina de CNC modifique estos parámetros de forma manual durante el proceso de maquinado para la reducción del tiempo de ciclo; sin embargo, éste no siempre es el caso.

Además, incluso cuando un programa de parte ya ha sido “afinado”, las diferencias en las propiedades de los materiales utilizados para producir diferentes lotes de partes son un factor a considerar, por lo que aunque hay métodos fuera de línea disponibles (Jerard et al.,

2000), se prefieren los métodos en línea (Haber et al., 2003; Lian et al., 2005). Así pues, los métodos en línea para el control de la fuerza de corte, con el fin de mantenerla dentro de los límites establecidos por la exigencias de la productividad y los factores que reducen la vida de la herramienta, son una gran mejora para cualquier proceso de maquinado. De hecho, los métodos en línea son también la mejor opción para detectar el desgaste y la rotura de la herramienta oportunamente, y esta detección puede llevarse a cabo al mismo tiempo que el proceso de fresado está siendo optimizado.

B.2.1 Medición sin sensores de la fuerza de corte

Durante el maquinado de una pieza las fuerzas de corte requeridas dependen del material empleado, la profundidad del corte d, la velocidad de avance F y la velocidad del husillo S. El material usado para maquinar la parte y la profundidad de corte quedan especificados por los requerimientos de la parte y en el plan de fabricación de la misma (ya sea que el conjunto de trayectorias a seguir sea generado por un programa de manufactura asistida por computadora o sea creado a mano por un programador experto), respectivamente, así que cualquier posibilidad de optimizar la fuerza de corte depende de la optimización de las velocidades de avance F y del husillo S.

Para aplicaciones de fresado, la velocidad del husillo S usualmente se programa en revoluciones por minuto (rev/min), como un número entero. Este parámetro debe ser calculado con base en la velocidad de corte tangencial V_{C} (m/min) recomendada para el material a maquinar y en el diámetro de la herramienta de corte D (Altintas, 2000); se puede demostrar que la fuerza de corte F_{C} es directamente proporcional a F/S.

Además, la corriente s proporcionada al servoamplificador o variador del husillo es directamente proporcional al torque del husillo, la cual a su vez es inversamente proporcional a S (Institute of Advanced Manufacturing Sciences, Inc., 1980) cuando la profundidad del corte d es constante y los ángulos de entrada y de salida del husillo subtienden un ángulo de $180°$. Similarmente, bajo estas mismas hipótesis las señales de corriente proporcionadas por los servoamplificadores de los ejes X, Y y Z, denotadas por x, y y z, respectivamente, son directamente proporcionales a las componentes X, Y y Z de la velocidad de avance F (Altintas, 1992), de modo que F es directamente proporcional a $\sqrt{x^2 + y^2 + z^2}$. Así pues,

$$F/S \propto s\sqrt{x^2 + y^2 + z^2}.$$

Ahora bien, puesto que la corriente s proporcionada por el variador del husillo es directamente proporcional a la corriente de monitorización s_{mon} del mismo, y las corrientes proporcionadas de los servoamplificadores de los ejes X, Y y Z también son directamente proporcionales a las respectivas corrientes x_{mon}, y_{mon} y z_{mon} de monitorización de los mismos, tenemos que

$$F_{\mathrm{C}} \propto F/S \propto s_{\mathrm{mon}} f_{\mathrm{mon}}, \tag{B.1}$$

donde

$$f_{\mathrm{mon}} = \sqrt{x_{\mathrm{mon}}^2 + y_{\mathrm{mon}}^2 + z_{\mathrm{mon}}^2}. \tag{B.2}$$

Hay tres escenarios posiblemente concurrementes cuando las suposiciones empleadas para deducir (B.1) no se cumplen:

1. La profundidad del corte d no es constante, es decir, se encuentra variando y su valor nominal puede ser mayor o menor de lo esperado, provocando que la tasa de remoción de material Q varíe en la misma proporción.

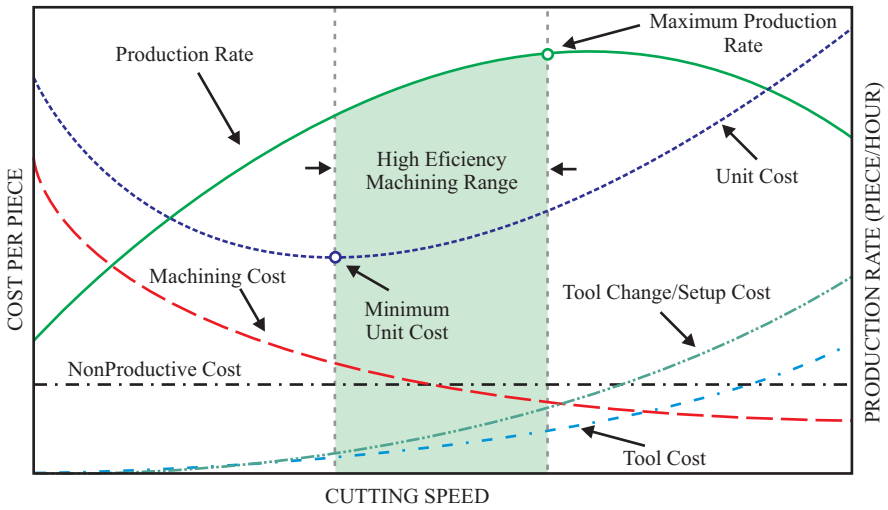


Figura B.2: Costo de maquinado y tasa de production contra la velocidad de corte.

2. Los ángulos de entrada y de salida del husillo subtienden un ángulo menor a $180°$, es decir, el ancho de corte w es menor de lo esperado, causando una reducción en Q.

3. La dureza del material difiere (localmente) de su valor nominal, causando que la potencia del husillo P_{S} se incremente o decremente.

En cualquier caso, P_{S} oscila y la fuerza de corte F_{C} también en consecuencia. Por lo tanto, mantener F_{C} constante es equivalente a mantener P_{S} constante, lo cual se puede lograr actualizando la velocidad de avance F.

B.2.2 Optimización del tiempo de ciclo

Nótese que, una vez que material para la parte a maquinar ha sido elegido, la velocidad del husillo S debe ser acotada superiormente según la información proporcionada por los manuales de maquinado: para preservar la vida de la herramienta, el incrementar la velocidad S del husillo más allá de los límites recomendados no es buena idea, y no ayuda en nada a reducir el tiempo de ciclo. Más aún, reducir la velocidad del husillo a tiempo de manera controlada (i.e., sin usar un paro de emergencia o algo parecido) puede ser infactible debido a la inercia del husillo. Por lo tanto, se recomienda que la velocidad S del husillo permanezca constante durante la totalidad del ciclo de maquinado. Por otra parte, incrementar la velocidad de avance F —dentro de los límites recomendados— sí reduce el tiempo de ciclo. Además, incrementar la velocidad de avance mientras se mantiene la velocidad del husillo constante tiene el mismo efecto que decrementar la velocidad del husillo y mantener la velocidad de avance constante; por lo tanto, en la práctica, no tiene caso reducir la velocidad del husillo.

Así pues, una vez que S ha alcanzado un valor constante durante el ciclo de maquinado, el avance de la bancada por diente F_{t} resulta ser proporcional a la velocidad de avance $F = F_{\mathrm{t}} n S$. Esto significa que si se ajusta la velocidad de avance entonces el avance de la bancada por diente F_{t} también se ajustará en la misma proporción junto con la fuerza de corte F_{C}. Como se ilustra en la figura B.2 de Stephenson y Agapiou (1997), la tasa de

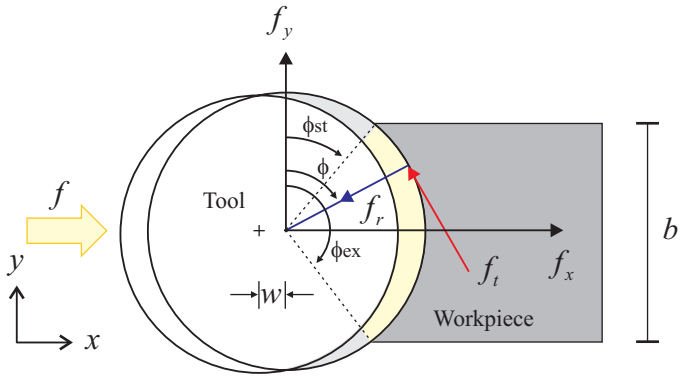


Figura B.3: Machining cost and production rate versus cutting speed.

producción depende tanto del tiempo de ciclo de maquinado como de los tiempos de inactividad: el tiempo de ciclo se reduce conforme el avance de la bancada por diente F_{t} aumenta, pero las velocidades de avance excesivas pueden provocar que las herramientas de corte se desgasten a una tasa acelerada, lo cual reduce a su vez la tasa de producción en virtud de los tiempos de inactividad involucrados en un cambio e instalación de una nueva herramienta; nótese que el costo unitario se reduce junto con el tiempo de ciclo, pero si las herramientas se desgastan a una tasa acelerada el costo unitario se incrementa debido a los costos de las herramientas y de su instalación. Por lo tanto, la velocidad de corte debe yacer en el rango entre el costo unitario mínimo y la tasa de producción máxima.

B.2.3 Detección de fractura de herramientas

La detección sin sensores de la fractura de herramientas usualmente se realiza por medio del procesamiento de la señal de monitorización de la fuerza de avance (B.2), la cual es directamente proporcional a la fuerza de corte f; esta señal contiene todos los datos relacionados a la fuerza ejercida por cada inserto sobre el material. Los insertos siguen una trayectoria trocoidal que genera variaciones periódicas en el grosor de la viruta—y por lo tanto en la fuerza de corte f—cada vez que el inserto remueve material, como se muestra en la figura B.3. El grosor de la viruta $0 \leq h(\phi) \leq w$ varía conforme el inserto entra y sale del material, y un modelo simplificado de esta variación está dado por $h(\phi) = w \sin \phi$ para $0 \leq \phi \leq \pi$. Los componentes cartesianos de la fuerza de corte f_i para el i-ésimo inserto se pueden calcular a partir de sus componentes radial f_{ri} y tangencial f_{ti} haciendo

$$\begin{aligned} f_{xi} &= -f_{ti} \cos \phi - f_{ri} \sin \phi, \\ f_{yi} &= f_{ti} \sin \phi - f_{ri} \cos \phi, \end{aligned} \tag{B.3}$$

para $\phi_{\mathrm{st}} \leq \phi \leq \phi_{\mathrm{ex}}$, donde ϕ_{st} es el ángulo de entrada y ϕ_{ex} es el ángulo de salida, y $f_{xi} = f_{yi} = 0$ cuando no se corta material. Por ende, los componentes cartesianos de la fuerza de corte f para una herramienta con N insertos están dadas por

$$f_x = \sum_{i=1}^{N} f_{xi}, \qquad f_y = \sum_{i=1}^{N} f_{yi}, \tag{B.4}$$

de donde

$$f = \sqrt{f_x^2 + f_y^2 + f_z^2}. \tag{B.5}$$

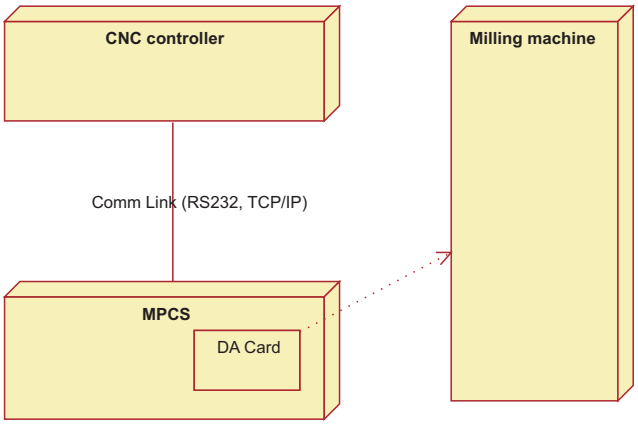


Figura B.4: MPCS deployment.

Por consiguiente, $f \propto f_{\text{mon}}$.

B.3. Arquitectura para control inteligente

A partir de la sección anterior podemos afirmar que los datos que un sistema de monitorización y control del proceso (SMCP) de fresado requiere están contenidos en las señales de monitorización de corriente x_{mon}, y_{mon}, z_{mon} y s_{mon}. El SMCP también depende de los valores F y S programados. Las señales de monitorización de corriente se muestrean directamente a partir del proceso de fresado mientras que los valores F y S deben ser enviados al SMCP cada vez que el controlador del centro de maquinado actualice estos valores o cuando sean solicitados (pueden ser solicitados cuando el SMCP empieza a operar).

B.3.1 Despliegue del sistema

Los parámetros F y S son enviados del CNC al SMCP por medio de un enlace de comunicación directo (en este caso, como se describe en la sección 6.2.1 en la página 3, la comunicación será llevada a cabo por el puerto COM2 del CNC), empleando el protocolo descrito en el capítulo 8; este mismo enlace de datos bidireccional es empleado para proveer de un ajuste de velocidad de avance al CNC con el fin de optimizar el tiempo de ciclo y para avisar de posibles condiciones de fractura de la herramienta cuando son detectadas. Las señales de monitorización de corriente x_{mon}, y_{mon}, z_{mon} y s_{mon} son muestreadas por el SMCP empleando una tarjeta de adquisición de datos (DA, por sus siglas en inglés) como se muestra en la figura B.4.

B.3.2 Arquitectura

Todo Sistema de Monitorización y Control del Proceso (SMCP) de fresado debe realizar tres tareas principales:

1. Muestrear los datos del proceso de fresado.

2. Filtrar los datos.

3. Procesar los datos, es decir, producir las señales de control.

Muestreo

El principal reto para cualquier SMCP que pretenda proveer de una interfaz estandarizada es proveer de una interfaz para recolectar las muestras que sea independiente del equipo físico particular que se esté empleando (hardware). Esto implica lidiar con dos cuestiones distintas:

1. Ofrecer un sistema de muestreo cuya interfaz sea independiente del equipo de adquisición de datos subyacente.

2. Implementar un format de representación de datos cuya interfaz permanezca igual sin importar cuál sea la implementación del sistema de muestreo, es decir, que dé soporte al polimorfismo.

Una interfaz uniforme para el sistema de muestreo, sin importar cuál sea el equipo de adquisición de datos subyacente, se puede implementar por medio una clase base abstracta (también conocida como interfaz) Sampler, a partir de la cual se puede derivar una clase concreta que se encargue de implementar la funcionalidad ofrecida a través de la interfaz al con base en el equipo de adquisición de datos concreto que se tenga; esto se puede hacer de dos maneras distintas, a saber, heredando de interfaz a partir de la clase Sampler o delegando la funcionalidad de la clase Sampler a una clase de implementación SamplerImpl. Esta segunda opción es más recomendable debido a que en la primera situación, cuando se hereda de la interfaz, el tamaño en memoria de la clase derivada que implementa la interfaz puede variar para cada equipo de adquisición de datos distinto, lo cual puede llegar a requerir una recompilación completa del software si es que se quiere cambiar de equipo de adquisición de datos. Así pues, al delegar la funcionalidad de la interfaz a una clase SamplerImpl y convertir a la clase Sampler en una clase manejadora (o entenada), es posible proveer diversas implementaciones del sistema de adquisición de datos aún después de que el SMCP ha sido instalado (e.g., desplegando cada clase de implementación en la forma de biblioteca de enlace dinámico)—la cantidad de memoria que emplea la clase manejadora no cambia nunca porque solo almacena una referencia de memoria a la clase de implementación—.

Ahora bien, puesto que cada tarjeta de adquisición de datos emplea su propio formato para representar al conjunto de datos que contiene las sucesiones de muestras de un conjunto dado de canales de muestreo—i.e., el almacenamiento de los datos es dependiente del dispositivo de hardware y de la forma en que el mismo ha sido configurado—, también emplearemos la técnica de delegar la funcionalidad de una interfaz para proveer de una interfaz DataSet común para cualquier conjunto de datos y al mismo tiempo mantener constantes los requerimientos de memoria de la clase manejadora; los ejemplares de la implementación DataSetImpl de la interfaz DataSet, sin embargo, no pueden ser creados por el código cliente porque ello requeriría conocimiento de la distribución en memoria de DataSetImpl, y esa información es precisamente lo que estamos tratando de ocultar del resto del sistema. Así pues, los objetos que representan a un conjunto de datos deben ser creados por una fábrica abstracta (Gamma et al., 1994), es decir, por una clase que provee una interfaz DataSetFactory para crear un DataSet (para cada eje del centro de maquinado) y su correspondiente delegado DataSetImpl. Si la interfaz *Sampler* hereda de (extiende a) la interfaz DataSetFactory, entonces la clase SamplerImpl puede crear ejemplares de la clase DataSet y de su correspondiente implementación DataSetImpl (véase la figura B.5).

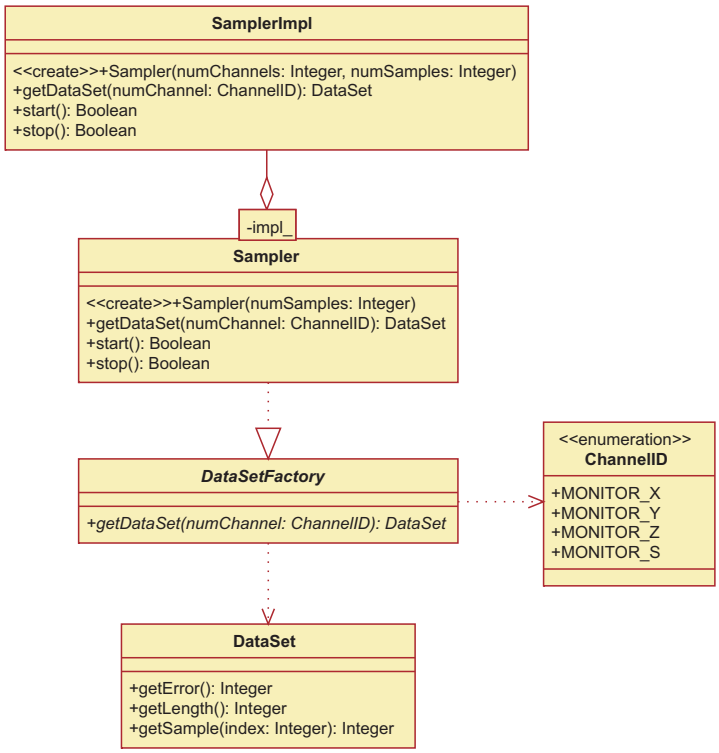


Figura B.5: Diseño del muestreador.

La implementación DataSetImpl del conjunto de datos puede implementar la interfaz *DataSet* en diferentes maneras, por ejemplo, como:

Vector Almacenando en un espacio de almacenamiento temporal los datos de cada canal de adquisición de datos (que en nuestro caso, corresponde a los datos de cada eje y del husillo por separado).

Iterador Por ejemplo, accediendo a espacio de almacenamiento temporal de datos compartido, donde los datos de cada canal están intercalados o distribuidos en diferentes segmentos del espacio de almacenamiento temporal, dependiendo del formato interno de la tarjeta de adquisición de datos. Esta técnica tiene la ventaja de que los datos no son duplicados.

Así pues, es posible implementar un catálogo de diferentes implementaciones del muestreador sin siquiera modificar la interfaz DataSet.

Filtrado

Para el esquema propuesto se obtienen cuatro conjuntos de datos distintos, a saber:

- Señal de monitorización de corriente s_{mon} del variador del husillo.
- Señal de monitorización de corriente x_{mon} del servoamplificador del eje X.
- Señal de monitorización de corriente y_{mon} del servoamplificador del eje Y.

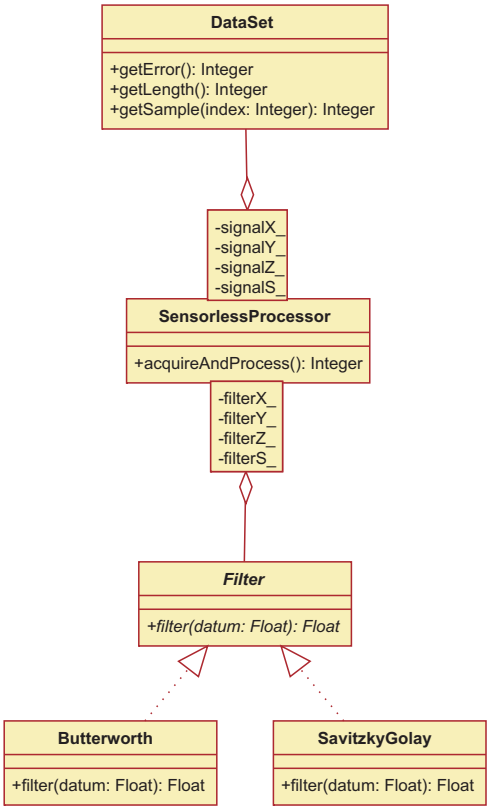


Figura B.6: Filtrado.

- Señal de monitorización de corriente z_{mon} del servoamplificador del eje Z.

Estas señales, sin embargo, contienen componentes no deseadas en la forma de ruido de alta frecuencia, conmutación de corriente del control y efectos de los tornillos de bolas, de modo que se requiere de un proceso de filtrado (Romero Troncoso et al., 2003). Por ejemplo, se puede emplear un filtro Butterworth pasa bajas para extraer las componentes principales de x_{mon}, y_{mon} y z_{mon} (Hamming, 1977) y un filtro suavizador de datos Savitzky-Golay para s_{mon} (Press et al., 2002). Puesto que hay muchas opciones para filtrar las señales, dependiendo de la situación que se presente, los diferentes algoritmos de filtrado se pueden encapsular en diferentes clases que implementan la misma interfaz Filter pero empleando una estrategia de filtrado distinta en el contexto de un SensorlessProcessor que recopila todas las muestras y filtra los datos; así pues, las distintas estrategias puede implementar filtros concretos tales como Butterworth o SavitzkyGolay. Tenemos entonces que la clase SensorlessProcessor actúa como mediador entre el sistema de adquisición de datos y el proceso de filtrado al desacoplar la adquisición de datos del filtraje particular que se desee emplear (figura B.6).

Procesamiento

El método SensorlessProcessor::acquireAndProcess llama al método process definido en la clase abstracta Processor, lo cual permite implementar diversas estrategias de procesamiento. Una clase Kernel concreta implementa la estrategia Processor y se comporta al mismo tiempo como un CuttingDataObserver, de tal suerte que cuenta con toda la información necesaria para realizar las tareas de monitorización y control del proceso. La

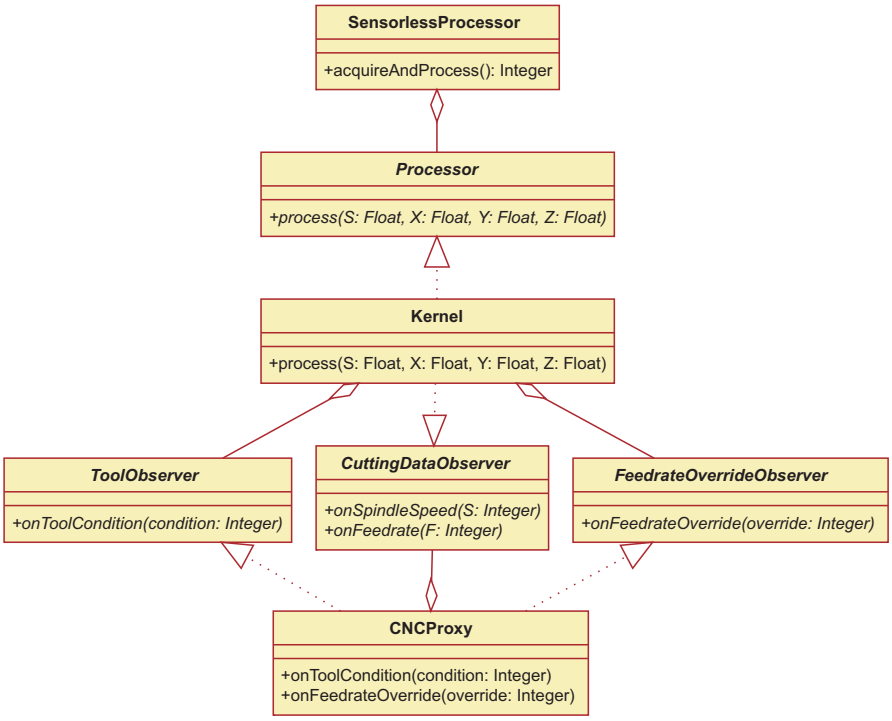


Figura B.7: Núcleo del SMCP.

interfaz ToolObserver se emplea para proveer datos sobre la condición de la herramienta a cualquier sistema que realice toma de decisiones con respecto a la condición de la herramienta. Asimismo, la clase FeedrateOverrideObserver es de utilidad para ofrecer información a cualquier sistema destinado a optimizar el tiempo de ciclo. En este caso, el observador concreto CNCProxy implementa tanto la interfaz ToolObserver como la interfaz FeedrateOverrideObserver con el fin de enviar información sobre la herramienta y sobre el ajuste de la velocidad de avance recomendado al sistema de CNC; al mismo tiempo, este componente provee al SMCP de la información requerida por la interfaz CuttingDataObserver, a saber, de la velocidad de avance F y de la velocidad del husillo S programadas. CNCProxy se comunica con el CNC por medio de una comunicación serial en el puerto COM2 del CNC.

BIBLIOGRAFÍA

Alexandrescu, A. *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-Wesley, 2001.

Altintas, Y. Prediction of cutting forces and tool breakage in milling from feedrate current monitoring. *ASME Transactions Journal of Engineering for Industry*, 114:386–392, 1992.

Altintas, Y. *Manufacturing automation: metal cutting mechanics, machine tool vibrations, and CNC design*. Cambridge University Press, Cambridge, UK, 2000. ISBN 0-521-65029-1.

Bharadwaja, R. M., Parlos, A. G., y Toliyat, H. A. Neural speed filtering for sensorless induction motor drives. *Control Engineering Practice*, 12:687–706, 2004.

Box, D. *Essential COM*. Addison Wesley Longman, 1998.

Deitel, H. M. y Deitel, P. J. *C++: cómo programar*. Pearson Educación con Prentice Hall, 2003.

Femat Díaz, A. Desarrollo de la ingeniería de software para la implementación de un CNC. Tesis de maestría, Universidad Autónoma de Querétaro, 2004.

Fowler, M. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. The Addison-Wesley Object Technology Series. Addison-Wesley Professional, 3 edición, 2003.

Franco Gasca, L. A., Herrera Ruiz, G., Peniche Vera, R., Romero Troncoso, R. d. J., y Leal Tafolla, W. Sensorless tool failure monitoring system for drilling machines. *International Journal of Machine Tools & Manufacture*, 46:381–386, 2006.

Gamma, E., Helm, R., Johnson, R., y Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

Haber, R., Alique, J., Alique, A., Hernández, J., y Uribe-Etxebarria, R. Embedded fuzzy-control system for machining processes: Results of a case study. *Computers in Industry*, 50:353–366, 2003.

Hamming, R. W. *Digital filters*. Dover Publications, Inc., New York, USA, 3 edición, 1977.

Herrera Ruiz, G. y Molina, A. CHROM-1, desarrollo de un control numérico mexicano. *Ciencia y desarrollo*, 15(85):95–100, 1989.

Institute of Advanced Manufacturing Sciences, Inc., editor. *Machining data book*, volumen 2. Braun-Brumfield, Inc., Ann Arbor, Michigan, USA, 1980.

Jerard, R. B., Fussell, B. K., Hemmett, J. G., y Ercan, M. T. Toolpath feedrate optimization: A case study. En *2000 NSF Design Manufacturing Research Conference*, Vancouver, British Columbia, Canada, jan 2000.

Lian, R.-J., Lin, B.-F., y Huang, J.-H. A grey prediction fuzzy controller for constant cutting force in turning. *Machine Tools and Manufacture*, 45:1047–1056, 2005.

Liang, S. Y., Hecker, R. L., y Landers, R. G. Machining process monitoring and control: The state-of-the-art. *Journal of Manufacturing Science and Engineering*, 126:297–310, 2004.

OSACA. *Specifications for the Industrial Implementation of OSACA*. Hümnos, 1998.

Park, S., Kim, S.-H., y Cho, H. Kernel software for efficiently building, re-configuring, and distributing an open CNC controller. *The International Journal of Advanced Manufacturing Technology*, 27(7-8):788–796, 2005. ISSN 0268-3768.

Parlos, A. G., Kim, K., y Bharadwaja, R. M. Sensorless detection of mechanical faults in electromechanical systems. *Mechatronics*, 14:357–380, 2004.

Petzold, C. y Yao, P. *Programming Windows 95*. Microsoft Press, 1996.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., y Flannery, B. P. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, 2 edición, 2002. ISBN 0-521-43108-5.

Romero Troncoso, R. d. J., Herrera Ruiz, G., Terol-Villalobos, I., y Jáuregui Correa, J. C. Driver current analysis for sensorless tool breakage monitoring of CNC milling machines. *International Journal of Machine Tools & Manufacture*, 43:1529–1534, 2003.

Smid, P. *CNC Programming Handbook: A Comprehensive Guide to Practical CNC Programming*. Industrial Press, Inc., New York, USA, 2 edición, 2003. ISBN 0-8311-3158-6.

Stephenson, D. A. y Agapiou, J. S. *Metal cutting theory and practice*. Marcel Dekker, Inc., New York, USA, 1997. ISBN 0-8247-9579-2.

Wang, H.-P. y Liu, Y.-T. Integrated design of speed-sensorless and adaptive speed controller for a brushless DC motor. *IEEE Transactions on Power Electronics*, 21:518–523, 2006.