UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA DE DESARROLLO, APLICADA A UN PROTOTIPO DE MÁQUINA FRESADORA CNC

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELECTRICISTA

SEBASTIÁN ALEJANDRO VALERIO GUERRERO

PROFESOR GUÍA:
MAURICIO BAHAMONDE BARROS

MIEMBROS DE LA COMISIÓN:
NESTOR BECERRA YOMA
HELMUTH THIEMER WILCKENS

SANTIAGO DE CHILE
SEPTIEMBRE 2007

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO
DE INGENIERO CIVIL ELECTRICISTA
POR: SEBASTIÁN VALERIO GUERRERO
FECHA: 12 DE SEPTIEMBRE DE 2007
PROF. GUÍA: SR. MAURICIO BAHAMONDE B.

"DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA DE DESARROLLO, APLICADA A UN PROTOTIPO DE MÁQUINA FRESADORA CNC"

El objetivo del presente Trabajo de Título es el de diseñar e implementar una Plataforma de Desarrollo orientada al área de control automático. Esta plataforma será aplicada a un prototipo de una máquina fresadora CNC, diseñada y construida para el propósito de este trabajo, cumpliendo el rol de controlador de la misma. La plataforma en sí podrá ser utilizada en el diseño y construcción de prototipos y como base para futuras aplicaciones específicas. Esto gracias a las distintas funciones con que cuenta y la documentación entregada en este trabajo.

La etapa de diseño está compuesta por una investigación sobre la tecnología a utilizar y los requerimientos que deberá cumplir. Este proceso entrega como resultado las especificaciones eléctricas y mecánicas de la plataforma y la matriz de la placa de circuito impreso para la construcción de la misma. En esta etapa se detallan los criterios y consideraciones realizadas de cada parte de la plataforma, de tal manera de dar cumplimiento a cada objetivo trazado.

En la etapa de implementación de la plataforma, la fase de construcción y armado de la placa de circuito impreso fue realizada por una empresa externa especializada en este tipo de trabajos.

Para comprobar el funcionamiento de la plataforma se construyó un prototipo de fresadora CNC, el cual forma parte del entorno de prueba creado especialmente para el presente trabajo. Se realizaron diferentes experimentos para poder observar el comportamiento de la plataforma como controlador. De esta manera, se obtuvieron resultados reales de la operación de la plataforma dentro de una aplicación y con ello se validó el diseño realizado.

Como conclusión al trabajo desarrollado se obtuvo la plataforma de desarrollo diseñada e implementada de acuerdo a los requerimientos planteados, los cuales fueron cuidadosamente verificados utilizando el entorno de prueba construido para tal propósito. De esta manera, se cuenta con una plataforma de desarrollo diseñada y construida en Chile, completamente funcional y lista para ser utilizada, siendo de gran ayuda en el diseño y construcción de aplicaciones en el área de: control automático; electrónica de potencia; electrónica de consumo; y, queda abierta a futuros usos.

Con mucho amor para ti.....

Agradecimientos

Todo este trabajo está dedicado a esa gran persona que me ha ayudado y apoyado más que nadie en todo este tiempo, además de entregarme todo su amor. Muchas gracias por haber estado ahí todo el tiempo conmigo, haberme aguantado todo este tiempo (prometo que lo retribuiré) cuando las cosas no salían como esperaba y celebrar cada paso que me acercaba al final. De verdad gracias Karen por todo tu amor, ya que fue fundamental para mí. Obviamente, espero que lo nuestro perdure en el tiempo y podamos cumplir todo lo que alguna vez hemos soñado juntos.

Agradezco también a mi familia, a mis padres y hermanos por el constante apoyo y la preocupación durante todos estos años. A mis futuros suegros y cuñada (espero que así sea), por hacerme sentir como uno más de su familia.

También agradezco a mi profesor guía Mauricio Bahamonde por los consejos y la ayuda brindada en el transcurso de esta memoria y por darme la oportunidad de reencantarme con el mundo de la electrónica en sus talleres de diseño. Se sufre bastante, pero el resultado vale la pena. Que mejor ejemplo que el trabajo de esta memoria.

Y por último, agradecer a compañeros y amigos de todos estos años de paso por la universidad donde guardo un buen recuerdo de cada uno de ellos. En especial muchas gracias a mis amigos Rayo, Ian, Jugo, Marcela, Jaimico, Eduardo, Claudio, Messen y muchos otros más.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1
Introducción

La investigación y el desarrollo de tecnologías es un tema que cada vez tiene mayor importancia en nuestro país. Precisamente esta es la principal motivación del presente trabajo de título. Esta misma motivación se incrementa por la participación del autor en el diseño e implementación de proyectos anteriores y a su visión de no ser tan sólo consumidores de tecnología, sino desarrolladores de la misma. Además, han sido de gran inspiración las memorias de Claudio Alarcón [1] y Jorge Ramírez [2] de la Universidad de Chile.

Una Plataforma de Desarrollo es básicamente un entorno que provee las herramientas necesarias en el diseño de nuevas aplicaciones. Gracias a que cuenta con múltiples funciones es bastante útil en el uso de diversos proyectos. Generalmente, es utilizada en el diseño y construcción de prototipos.

El propósito de este trabajo es investigar y entregar todos los aspectos técnicos de diseño e implementación, con el fin de ser una guía en el desarrollo de aplicaciones de naturaleza similar al de la plataforma de este trabajo. El resultado tangible de esta memoria, que es la construcción de la plataforma, permite contar con una gran herramienta a futuro en el área de la electrónica y con un gran potencial para fines educacionales. Un claro ejemplo de esto son los Talleres de Diseño de Sistemas Digitales, cursos dictados por el Departamento de Ingeniería Eléctrica de la Universidad de Chile, los cuales están a cargo de Mauricio Bahamonde (profesor guía de este trabajo).

Por otro lado, debido a que la plataforma es diseñada y construida en nuestro país, la reparación o mantención de la misma es perfectamente realizable en Chile, importando sólo los componentes necesarios para esta tarea. Cuando este problema ocurre con equipos importados, generalmente estos deben ser reparados en su país de origen.

Finalmente, el área de aplicación de este trabajo es el control de máquinas CNC, donde la sigla CNC significa *Computer Numerical Control* o Control Numérico Computacional. En pocas palabras, esta tecnología permite controlar una máquina específica de manera computacional, indicándole lo que debe hacer.

Para este trabajo se diseñará y construirá un prototipo de fresadora dado que es la que presenta ventajas en cuanto a su construcción y la posibilidad de crear diversas figuras. Esto en comparación con un torno mecánico el cual puede realizar sólo figuras con simetría cilíndrica.

1.1. Objetivos

1.1.1. Objetivo General

Diseñar e implementar una Plataforma de Desarrollo orientada al área de control de maquinaria. En este caso, a máquinas de Control Numérico Computacional (CNC).

1.1.2. Objetivos Específicos

Los objetivos específicos que se pretenden lograr mediante la realización de la presente memoria son:

- Diseñar e implementar los distintos bloques funcionales de la plataforma de desarrollo.

- Diseñar la plataforma de tal manera que pueda ser fácilmente utilizable en aplicaciones futuras.

- Construir un prototipo de máquina fresadora CNC, ajustada a los requerimientos con que se desee trabajar en la memoria. Se debe dejar en claro que la construcción de la máquina, es decir, la parte mecánica; es un objetivo secundario del trabajo de título.

- Diseñar e implementar un sistema de sensores que puedan asegurar el correcto funcionamiento y operación de la máquina.

- Desarrollar un software específico para la comunicación entre la información fuente y el prototipo de máquina fresadora CNC.

1.2. Estructura de la memoria

El presente documento consta de cinco capítulos, los cuales se comentan brevemente a continuación.

En el Capítulo 1 se entrega una introducción al tema, con el fin de situar al lector en el área donde se enmarca el trabajo. Asimismo, se explicitan tanto el objetivo general como los objetivos específicos del presente trabajo.

En el Capítulo 2 se entrega la información recopilada y que es necesaria en el desarrollo del trabajo. Para el lector este capítulo es fundamental para entender el resto de la memoria, si es que no se tiene un amplio conocimiento en la materia.

El Capítulo 3 trata sobre el diseño e implementación del proyecto realizado para este trabajo con un mayor énfasis en la Plataforma de Desarrollo. Este capítulo sirve de base para desarrollos futuros ya que presenta los aspectos técnicos más importantes en el diseño de un dispositivo de este tipo, desde los requerimientos hasta la implementación. Lo mismo ocurre para el prototipo de máquina fresadora y todo lo que esto implica, desde la mecánica hasta el programa que controla la máquina.

En el Capítulo 4 se realizan pruebas técnicas de manera de verificar el correcto funcionamiento de cada una de las partes de este proyecto. Gracias a esta verificación es posible

validar el diseño realizado en el Capítulo 3. Además, se entregan los resultados obtenidos con el fin de transparentar el desarrollo de las pruebas realizadas.

Finalmente, en el Capítulo 5 se concluye el trabajo realizado en base a los objetivos propuestos. Se explicitan los trabajos futuros en los cuales la plataforma de desarrollo representa un papel importante.

Capítulo 2
Revisión Bibliográfica

2.1. DSP

El título anterior se ha dejado como sigla debido a que existen dos definiciones para ésta: como sustantivo *Digital Signal Processor* (Procesador Digital de Señal); y, como verbo *Digital Signal Processing* (Procesamiento Digital de Señal) [3].

Un Procesador Digital de Señal es un computador rápido dentro de un chip, que ha sido optimizado para la detección, procesamiento y generación de señales del mundo real (como la voz, video, música, etc.). Las funciones descritas anteriormente son realizadas en tiempo real.

El Procesamiento Digital de Señal se refiere a la manipulación de señales que han sido generadas en el mundo real, donde estas señales son representadas como dígitos (números). Es decir, estas señales se digitalizan para su posterior procesamiento.

2.1.1. Características

La expresión matemática en la Ecuación 2.1 describe el proceso de filtrado digital:

Ecuación 2.1

$$y(x) = \sum_{k=0}^{K} b(k)x(i-k) - \sum_{m=1}^{M} a(m)y(i-m)$$

Sin profundizar mucho en la matemática de esta relación, se puede ver que este proceso requiere una cierta cantidad de operaciones: sumas; restas; y, multiplicaciones. Los valores $b(k)$ y $a(m)$ son dos tablas de coeficientes del filtro guardados en memoria. Estos coeficientes serán multiplicados por un vector de muestras de entradas $x(i-k)$ y salidas $y(i-m)$ anteriores, respectivamente. Cada vez que una nueva muestra ingresa al sistema esta toma el primer lugar en la tabla de muestras y la más antigua es eliminada, esto debido a que se trabaja con tablas de largo fijo. Lo mismo ocurre con la tabla de salidas. Este proceso indica que un dispositivo DSP debe ser diseñado para acceder y administrar eficientemente sus áreas de memoria y, simultáneamente, manejar el flujo de datos desde y hacia los puertos de datos. La Figura 2.1 ilustra este proceso.

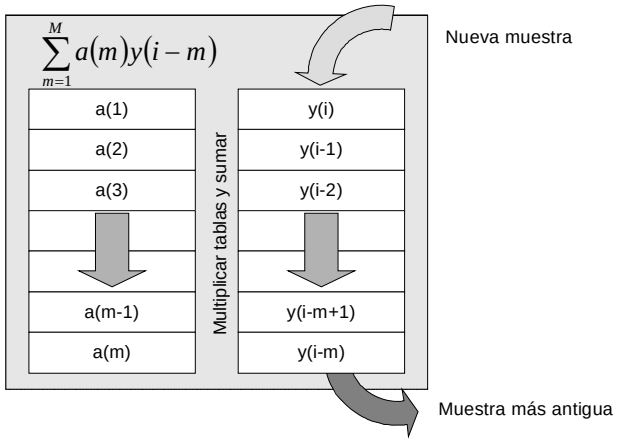


Figura 2.1 Coeficientes del Filtro y Muestras.

Otro ejemplo de algoritmos DSP es la Transformada Rápida de Fourier o FFT[1]. La FFT usualmente es descrita por el número de muestras usadas en su cálculo, donde el número de muestras siempre es una potencia de 2 (por ejemplo: 128; 256; o, 512 puntos FFT). Se puede

[1] *Fast Fourier Transform.*

estimar la cantidad de multiplicaciones necesarias para cada punto de salida del resultado de la FFT según la Ecuación 2.2.

Ecuación 2.2

$$\text{Número de multiplicaciones} = \frac{N}{2}\log_2 N$$

Por ejemplo, si se tienen 128 muestras (N=128), el procesador DSP deberá realizar 448 multiplicaciones para obtener tan sólo un punto de la FFT.

Estos ejemplos se mencionan con el propósito de resaltar la naturaleza de los algoritmos DSP. En un típico sistema que usa un dispositivo DSP, las muestras arriban en un período regular de tiempo desde la entrada al DSP. En cada llegada de muestras se debe realizar todo el cálculo necesario que requiera cada algoritmo y el resultado debe ser presentado en la salida del DSP, todo esto antes de que llegue la próxima muestra de datos. Esta característica es esencial para cualquier dispositivo que requiera implementar exitosamente algoritmos DSP en tiempo real.

Los algoritmos básicos de procesamiento de señales mencionados serán usados sobre muestras de datos, calculando una suma (acumulación) de una serie de productos. Un DSP debe ser capaz de acumular series de resultados de multiplicaciones, conocida como *Multiplity Accumulate* (MAC) o Multiplicación Acumulada.

Cabe destacar que el DSP está provisto de hardware dedicado para realizar las operaciones de multiplicación y MAC. El resultado obtenido de una multiplicación entre dos operandos usualmente requiere un ciclo de reloj del procesador DSP. En comparación con un típico microprocesador, este requiere de aproximadamente 80 ciclos de reloj para realizar una multiplicación de 16 bits[2].

En cuanto a la administración de la memoria, el DSP debe ser capaz de acceder a las áreas de memoria de la manera más eficiente posible de modo que el procesador no pierda tiempo

[2] *Binary Digit* o Dígito Binario.

esperando que los datos sean cargados[3]. Para el caso de la operación MAC, la cual se ejecuta reiteradamente por pequeños períodos de tiempo de manera muy rápida, es importante que una eficiente operación MAC no se "congele" mientras se espera que los datos sean cargados de la memoria. Los dispositivos DSP proveen una cierta cantidad de mecanismos para cargar los datos y determinar su dirección dentro de la memoria.

Otra característica importante en el diseño de muchos DSPs es la estructura o arquitectura de la misma memoria. Muchos microprocesadores son diseñados en base a la arquitectura Von Neumann, donde las instrucciones y los datos comparten el mismo espacio de memoria y son accesados por los mismos buses de dirección y de datos.

La mayoría de los DSPs usan la arquitectura Harvard donde las instrucciones y los datos son guardados en sectores separados de memoria con accesos independientes. Esto es gracias a que las áreas de programa (instrucciones) y de datos poseen su propio bus de direcciones y de datos. Esta arquitectura habilita al DSP a acceder a instrucciones y datos simultáneamente, entregando una gran ventaja en cuanto a rapidez sobre un microprocesador convencional [4]. Además, se tiene la libertad de que el ancho (tamaño en bits) de los buses sea diferente, lo que permite que el ancho de los datos sea distinto que el de las instrucciones.

2.1.2. Interfaces

En la parte anterior se habló sobre la operación del DSP en cuanto a la parte central del mismo. A este se le denomina *Core* o núcleo. Las interfaces o periféricos del DSP permiten la comunicación entre lo que está fuera del DSP con el núcleo. La Figura 2.2 un dispositivo hipotético con algunas de las funciones o módulos típicos de un DSP.

[3] Ciclo *Fetch* o Captura de una Instrucción.

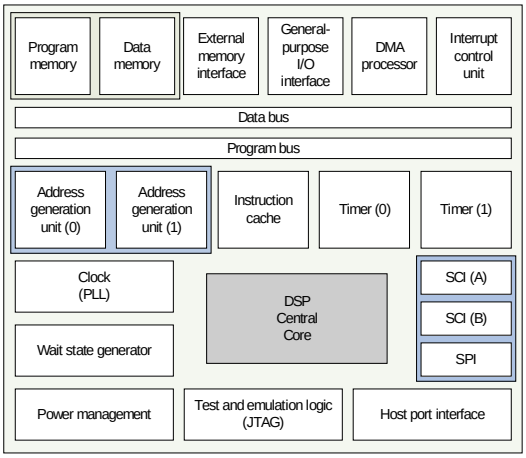


Figura 2.2 Dispositivo DSP hipotético.

Como se puede ver en la Figura 2.2 el núcleo se encuentra en el centro del dispositivo, rodeado por los componentes que están fuera de este pero que pertenecen al DSP. La mayoría de estos componentes o módulos están capacitados para ejecutarse en paralelo con el núcleo, tomando un pequeño tiempo del ciclo de instrucción del núcleo DSP. Un ejemplo de esto es un *on-chip timer* (temporizador interno), el cual puede ser inicializado para interrumpir al DSP cada un cierto tiempo predefinido.

Los componentes existentes dentro de un DSP son muchos y varían de acuerdo a la marca, familia o propósito de cada DSP.

Memoria

La mayoría de los dispositivos DSP están provistos con una cantidad limitada de *On-chip Read-Only Memory* o memoria sólo lectura interna (ROM) y *Random Access Memory* o memoria de acceso aleatorio (RAM). Además, existen varios tipos de memorias RAM como: D*ual-Access RAM* o RAM de doble acceso; y, *Single-Access RAM* o RAM de acceso simple.

La *On-chip ROM* es parte del espacio de memoria del programa y, en algunos casos, forma parte del espacio de memoria de los datos. En dispositivos con una pequeña cantidad de ROM,

esta contiene un *boot loader*[4] el cual sirve para cargar de una ROM externa, por ejemplo, durante la secuencia de inicio. Además, el *boot loader* permite que el código de la aplicación o programa pueda ser cargado desde una interfaz serial o a través del uso de JTAG[5].

La *On-chip Dual-Access RAM* (DARAM) recibe este nombre ya que cada bloque de esta memoria puede ser accedida dos veces por cada ciclo de máquina, es decir, puede leer y escribir en un bloque de DARAM en el mismo ciclo.

En la *On-chip Single-Access RAM* (SARAM) cada bloque es accesible una vez solamente por cada ciclo de máquina, siendo de escritura o de lectura. En ambos casos, las memorias siempre son mapeadas y su principal función es la de almacenar valores de datos.

Interfaz de E/S de Propósito General

Se conoce como *General Purpose Input/Output* (GPIO) o Entrada/Salida de Propósito General. Esta interfaz contiene puertos de señales digitales de entrada y/o salida. Generalmente se trata de un puerto paralelo el cual se puede configurar como entrada o salida. Cuando es configurado como salida, se puede escribir en un determinado registro para que controle la salida de cada pin del puerto. Cuando es configurado como entrada, se puede detectar el estado del pin de entrada leyendo el estado de un registro interno asociado al puerto.

Puertos Seriales

Un puerto serie es una interfaz entre distintos dispositivos (microcontroladores, computadores, periféricos, etc.) donde la información es enviada bit a bit, pudiendo enviar un sólo un bit a la vez. Dentro de un dispositivo DSP los puertos seriales más comunes son: *Serial Communication Interface (SCI)* o Interfaz de Comunicación Serial y *Serial Peripheral Interface (SPI)* o Interfaz Serial de Periféricos.

---

[4] Cargador.

[5] *Join Test Action Group*.

El módulo SCI utiliza una comunicación asíncrona. Debido a esto sólo necesita dos líneas de comunicación: la línea de transmisión (TX) y de recepción (RX). Generalmente, este módulo se utiliza para comunicar el DSP con un computador personal (PC).

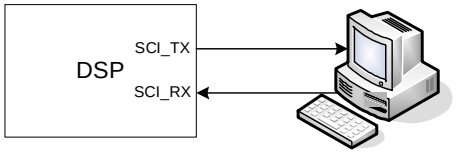


Figura 2.3 Comunicación mediante SCI.

El módulo SPI es un estándar de comunicaciones usado principalmente para la transferencia de información entre dispositivos electrónicos. Se caracteriza por utilizar una comunicación síncrona en modo Maestro-Esclavo *(Master-Slave)*, donde sólo un dispositivo Maestro puede controlar varios dispositivos Esclavos. Esto es gracias a que este estándar es de topología Bus. Incluye: una línea de reloj *(Clock SCLK)*; una línea de datos de entrada *(Master Input - Slave Output MISO)*; una línea de datos de salida *(Master Output - Slave Input MOSI)*; y, un pin de selección *(Chip Select CS)* que conecta y desconecta la operación del dispositivo con el que se desea comunicar. Se deberán utilizar tantos pines CS como dispositivos esclavos se tengan.

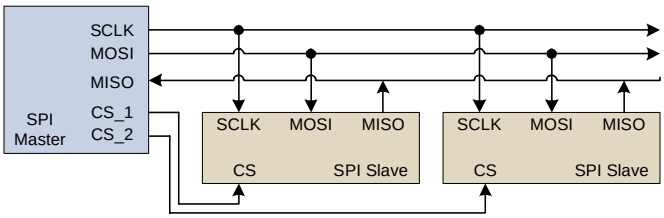


Figura 2.4 Comunicación mediante SPI.

2.2. Control Numérico Computacional

2.2.1. Introducción

El Control Numérico Computacional o *Computer Numerical Control* (CNC) es el proceso de fabricación de piezas mecanizadas. Esto es realizado y supervisado mediante un controlador

computarizado. Este controlador usa motores para mover cada eje de la herramienta de la máquina y actualmente regula la dirección, velocidad y longitud para el movimiento realizado por cada motor [5][6].

Un programa es cargado, por un operador, dentro del computador o controlador de la máquina CNC para ser ejecutado. El programa consiste de datos de puntos numéricos en conjunto con comandos de control y funciones específicas para el controlador.

Control Numérico o *Numerical Control* (NC) es el término original de esta tecnología y continúa utilizándose indistintamente con el término CNC. La tecnología NC ha sido uno de los desarrollos más grandes en la industria de la manufactura en los últimos 50 años.

2.2.2. La evolución de la tecnología NC

Los principios de la fabricación NC se encuentran en la Revolución Industrial. En los inicios se usó para automatizar la producción en correas transportadoras, poleas y levas. Pero la mayor parte del proceso todavía se hacía de manera manual. Los maquinistas de esos días producían piezas de mayor calidad pero no a grandes volúmenes. No fue hasta la Segunda Guerra Mundial que la industria se dio cuenta de que se necesitaban ambos requerimientos al mismo tiempo, es decir, calidad y cantidad.

En 1952, la primera máquina de 3 ejes numéricamente controlada fue creada. El controlador de esta máquina fue el computador Whirlwind desarrollado por el MIT[6]. En 1954, el control numérico fue anunciado al público. En las primeras generaciones máquinas NC requerían cintas de papel perforadas con el código a ejecutar. Avanzadas investigaciones y desarrollos más complejos trajeron nuevas generaciones de máquinas NC. La consecuente introducción del Control Numérico Computarizado, donde un computador es usado para controlar la máquina, eliminó la dependencia de las abultadas y frágiles cintas de papel.

[6] *Massachusetts Institute of Technology.*

El control numérico directo y el distribuido (DNC)[7] son términos que describen la comunicación hacia la máquina desde un computador remoto. En el NC directo, bloques de instrucciones del programa son enviados a la máquina tan rápido como ella los pueda ejecutar. En el NC distribuido el programa completo o múltiples programas son comunicados a la máquina CNC o varias máquinas CNC, usualmente vía una comunicación serial RS232[8]. Este proceso fue posible incrementando la capacidad de memoria de los controladores CNC.

2.2.3. Aplicaciones NC

Desde los días de la primera fresadora NC, las aplicaciones de esta tecnología han variado bastante. Algunos ejemplos de éstas se describen a continuación.

- Máquinas Fresadoras. Las máquinas CNC Fresadoras usan un cortador rotatorio para el movimiento de corte y un movimiento lineal para la alimentación. El material es empujado en el cortador, o el cortador es empujado al material, en caminos rectos o curvos tridimensionales, para producir los elementos deseados de una pieza. La pieza terminada es creada mediante la remoción de todo el material innecesario desde la pieza de trabajo.

- Tornos. Los Tornos CNC rotan la pieza de trabajo en contra de un único punto de una herramienta para producir movimiento de corte. La herramienta se alimenta a lo largo o en la pieza de trabajo para producir el movimiento de alimentación.

- Centros de Mecanizado. Los centros de mecanizado son máquinas CNC más sofisticadas que frecuentemente combinan las tecnologías de fresado y torneado.

- Máquinas EDM. Una Máquina de Descarga Eléctrica o *Electrical Discharge Machine* (EDM) usa chispas eléctricas para hacer una cavidad en una pieza de metal.

---

[7] *Distributed Numerical Control.*

[8] *Recommended Standard 232:* Estándar de Comunicación Serial.

2.2.4. Motores en Máquinas CNC

Dentro de las aplicaciones NC y en la robótica en general, es bastante común encontrar motores del tipo Paso a Paso (PaP) o *Stepper* como se les conoce comúnmente. Estos motores son ideales en la construcción de mecanismos donde se necesiten movimientos de precisión. La característica principal de estos motores es el hecho de poder moverlos un paso a la vez por cada pulso que se le aplique. Este paso puede variar desde 90° hasta pequeños movimientos de tan solo 1.8° [7].

Existen dos tipos básicos de motores Stepper: de imán permanente; y, de reluctancia variable. En el presente trabajo sólo se analizará el primer caso. Además, los motores Stepper de imán permanente se dividen en dos tipos: Unipolar y Bipolar [8].

Un motor bipolar consiste en un imán permanente rotatorio rodeado por polos en el estator que contienen bobinas. Estas bobinas son energizadas en una cierta secuencia para permitir el movimiento del rotor.

Una manera de realizar esto es energizar las bobinas en la secuencia AB/CD/BA/DC (BA significa que se alimenta la bobina AB pero en el sentido opuesto). Esta secuencia es conocida como *one phase on* o una fase encendida en paso completo. Solo una fase es energizada a la vez. Otra forma es energizar ambas fases. Se le conoce como *two phase on* y es el modo normal de operación de un motor bipolar, ya que entrega el torque más alto.

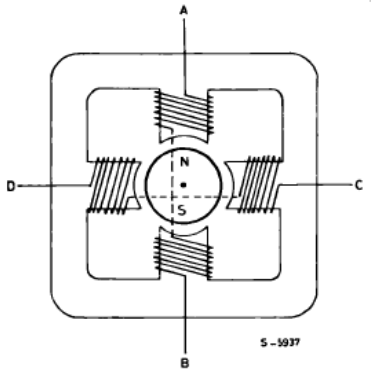


Figura 2.5 Motor Stepper Bipolar.

El motor mostrado en la Figura 2.5 puede realizar un avance de 90° por paso. Motores reales poseen múltiples polos para reducir el ángulo de paso a unos pocos grados.

El motor unipolar es idéntico a la máquina bipolar descrita anteriormente exceptuando las bobinas bifilares que se utilizan para invertir el flujo del estator. Este tipo de motores son más fáciles de controlar ya que las bobinas no se deben energizar en un sentido y luego en otro, como en el caso del motor bipolar.

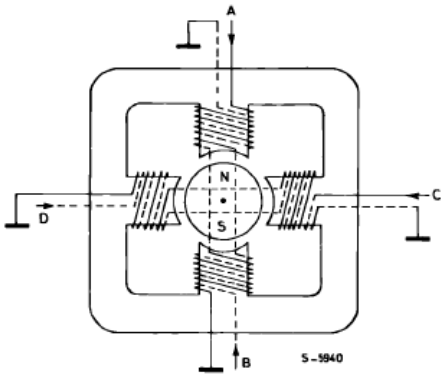


Figura 2.6 Motor Stepper Unipolar.

2.3. DXF

El Formato de Dibujo de Intercambio o *Drawing Interchange Format* o *Drawing Exchange Format* (DXF) [9] es un formato del archivo de datos CAD[9], desarrollado por Autodesk como una solución para permitir interoperabilidad de los datos entre AutoCAD y otros programas. DXF fue introducido originalmente en diciembre de 1982 como parte de AutoCAD 1.0, y pensado para proporcionar una representación exacta de los datos en el formato nativo de AutoCAD DWG (*drawing*[10]).

[9] *Computer Aided Design*.

[10] Dibujo.

2.3.1. Estructura del Archivo DXF

Las versiones ASCII de DXF pueden ser leídas con un editor de texto normal. La estructura está compuesta por secciones. La organización básica de un archivo DXF es:

- HEADER. Sección que contiene la información general acerca del dibujo.
- CLASSES. Mantiene la información de clases definidas por la aplicación que son referenciadas en otras secciones.
- TABLES. Esta sección contiene definiciones de distintos parámetros.
- BLOCKS. En esta sección se definen Bloques dentro del dibujo que agrupan a ciertas entidades.
- ENTITIES. Muestra las entidades en el dibujo, incluyendo cualquier referencia a un Bloque.
- OBJECTS. Contiene datos que se aplican a objetos no gráficos.
- THUMBNAILIMAGE. Contiene la imagen preliminar del archivo DXF.
- END OF FILE. Fin del archivo.

2.3.2. Funciones de Dibujo

El estándar DXF contiene una amplia gama de funciones o entidades, de las cuales se mencionan algunas formas básicas.

LINE

Dibuja una línea recta entre dos puntos extremos. Entrega el punto de inicio y el punto final.

CIRCLE

Dibuja un círculo. Entrega el punto central y el radio.

Capítulo 3

Diseño e Implementación

3.1. Plataforma de Desarrollo

Uno de los objetivos más importantes de tener una plataforma de desarrollo es el contar con el hardware necesario para poder realizar distintas aplicaciones o prototipos de proyectos. Es decir, se busca que la plataforma sea "flexible". No está de más decir que estas aplicaciones deben ser de una naturaleza similar, esto debido a que el hardware a desarrollar probablemente pertenezca a una cierta familia. Por ejemplo, existen plataformas de desarrollo para audio, video, control, etc. En este caso, la plataforma está orientada al área de control.

Otro punto importante es que la plataforma de desarrollo sea "compatible". Esto se refiere, a la capacidad de interconectarse y comunicarse con otras tecnologías estándares y acordes a los tiempos actuales. Un ejemplo de esto es la comunicación mediante un puerto USB[11], vital en estos días ya que la mayoría de los computadores personales móviles o *laptops* ya no incluyen la clásica interfaz serial RS232. Esta interfaz todavía se sigue utilizando mucho en la comunicación y programación de microcontroladores y DSPs. Pero, por parte del autor, se piensa que ya es hora de saltar a las tecnologías actuales.

[11] Universal Serial Bus.

Por otro lado, también es de gran interés que el resultado tangible de este trabajo (placa de desarrollo) sea de un nivel profesional en cuanto a su diseño y construcción. Para ello se propone el uso de componentes de tecnología de montaje superficial o *Surface Mount Technology (SMT)*, con el fin de reducir: las dimensiones; el peso; y, las interferencias electromagnéticas; entre otras características de la placa.

Una plataforma destinada al área de control debe poseer o contar con ciertas características implementadas tanto en el hardware como en el software. La Figura 3.1 muestra a nivel de bloques, tales características.

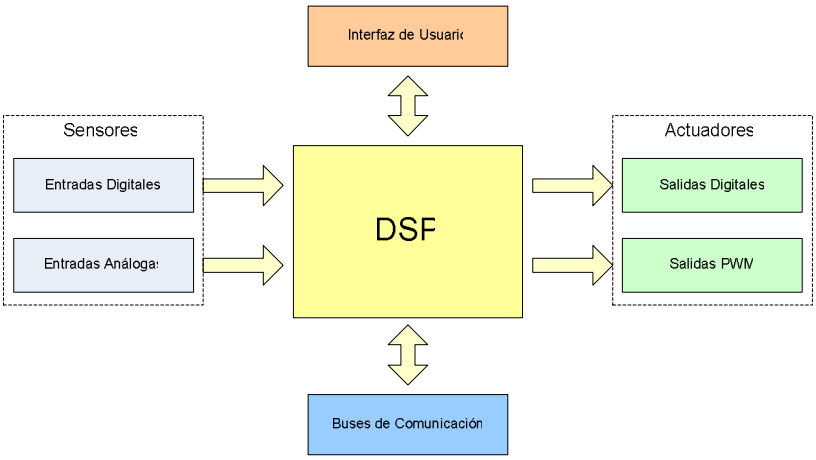


Figura 3.1 Estructura de una Plataforma de Desarrollo

Existen diversos componentes o bloques funcionales que están integrados dentro de un dispositivo DSP. Entre los más comunes se encuentran el bloque: Conversor Análogo a Digital; Moduladores por Ancho de Pulso; Puertos de Entrada/Salida de Propósito General; y, Puertos de Comunicación.

Otros componentes que no se encuentran dentro del DSP de todas maneras deben estar incluidos en la plataforma de desarrollo. Estos son: Alimentación (POWER); Reloj del Sistema (CLOCK); y, Comunicación al PC.

3.1.1. Requerimientos

Dependiendo de la función de cada uno de los bloques funcionales es que la plataforma podrá ser más o menos flexible para la implementación de distintas aplicaciones futuras a desarrollar. Entre los requerimientos a nivel de bloques es necesario que el dispositivo DSP que se utilizará contenga como mínimo:

- Conversor Análogo a Digital (ADC)
- Canales de Modulación por Ancho de Pulso (PWM)
- Puertos de Entrada/Salida de Propósito General (GPIO)
- Puerto de Comunicación Síncrono (SPI)
- Puerto de Comunicación Asíncrono (SCI)
- Temporizadores Internos (TIMER)
- Conexión JTAG

Estos requerimientos son básicamente a nivel de hardware. Por otro lado, existen otros requerimientos que tienen que ver con el software que se ejecutará dentro del DSP. Para una cierta aplicación basta con que el código programado dentro del DSP funcione y se ejecute correctamente. Pero en otras aplicaciones que requieren una mejor administración de los recursos del propio DSP mientras un código está siendo ejecutado. Es en estos casos donde la utilización de un Sistema Operativo Embebido es de gran importancia. Por esto, una característica más que el DSP deberá cumplir es el hecho de poder aceptar un sistema de este tipo.

En relación a la plataforma, esta debe cumplir con unos pocos requerimientos que no son propios del DSP y que permiten que esta sea lo más completa posible.

- Alimentación de la Plataforma
- Reloj del Sistema
- Comunicación al PC vía USB
- Comunicación Serial RS232

Considerando los requerimientos antes planteados la plataforma, desde el punto de vista funcional, debiera ser de la forma como se ve en la Figura 3.2.

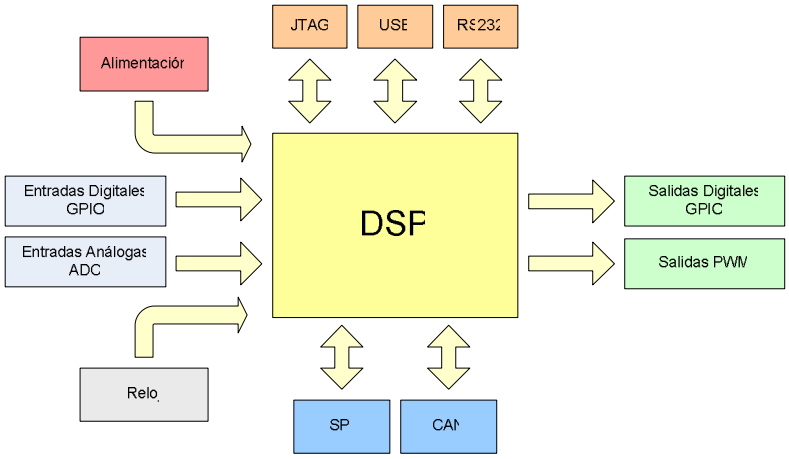


Figura 3.2 Módulos Plataforma de Desarrollo

3.1.2. Diseño

Uno de los principales requerimientos de la plataforma es que ésta sea muy flexible, es decir, que sea aplicable a una gran cantidad de proyectos dentro del área del control de dispositivos electromecánicos. La plataforma será más o menos flexible dependiendo del DSP que se utilizará.

DSP

Dentro de la amplia gama de dispositivos DSP existentes en el mercado, se eligió el modelo TMS320F2810 [10][11] de la familia C2000 de *Texas Instruments*. Esta elección se realizó en base a los siguientes argumentos:

- Dispositivo diseñado para Propósito General
- Capacidad de montar un Sistema Operativo (RTOS)
- Bajo Costo
- Uso previo de dispositivos DSP de la marca (Familia C2000)

Las características técnicas del DSP son bastantes, de las cuales se resaltan las principales o las que son de mayor importancia para este diseño en particular (cada una de estas características están explicadas en el Capítulo 2 de este trabajo):

- Procesador de 32 bits de Alto Rendimiento (TMS320C28x)
  - 16x16 bits y 32x32 bits Operaciones MAC.
  - Arquitectura de Bus Harvard.
- Memoria incorporada (On-chip)
  - Flash: 128 kBytes.
  - RAM: 36 kBytes SARAM.
- Periféricos de Control de Motores
  - Dos Administradores de Eventos.
  - Compatibles con DSP serie 240xA.
- Periféricos de Comunicación Serial
  - Dos interfaces de Comunicación Serial (SCI), Estándar UART.
  - Interfaz de Bus Serial (SPI).
  - Interfaz de Bus de Red de Area (eCAN).
- Conversor Análogo a Digital incorporado (ADC)
  - 16 Canales de 12 bits cada uno.
- Puertos de Propósito General de Entrada/Salida (GPIO)
  - Hasta 56 pines.
- Herramientas de Desarrollo incluidas
  - JTAG.
  - DSP/BIOS RTOS.

Alimentación

Para comenzar con el diseño de la placa lo primero es la elección del DSP. Luego de esto es necesario integrar un bloque de alimentación que satisfaga o cumpla con las especificaciones del DSP y otros circuitos integrados que se encuentran en la misma placa. Según las especificaciones eléctricas del DSP, este se energiza completamente a través de distintos pines de alimentación los cuales se detallan en la Tabla 3.1 (Sección 2.4 [10]).

Tabla 3.1 Pines de Alimentación TMS320F2810

Pin	Descripción
V_{DDA1}, V_{DDA2}	Alimentación Análoga ADC
V_{DD1}	Alimentación Digital ADC
V_{DDAIO}	Alimentación Análoga I/O[12]
V_{DD}	Alimentación Digital Core[13]
V_{DDIO}	Alimentación Digital I/O
V_{DD3VFL}	Alimentación Flash Core[14]
V_{SS}	Tierra de Alimentación

La Tabla 3.2 describe las condiciones de operación recomendadas por el fabricante. Hay que mencionar que para el voltaje de *Core* o Núcleo el DSP puede operar a 1.8 [V] con un reloj del sistema máximo (máximo recomendado) de 135 MHz; o a 1.9 [V] con un máximo de 150 MHz (Sección 6.2 [10]).

Tabla 3.2 Condiciones Recomendadas de Operación TMS320F2810

Pin		Mínimo	Nominal	Máximo	Unidad
V_{DDA1} , V_{DDA2} , V_{DDIO} , V_{DD3VFL}		3.14	3.3	3.47	V
V_{DD} , V_{DD1}	1.8 V (135MHz)	1.71	1.8	1.89	V
	1.9 V (150MHz)	1.81	1.9	2	
V_{SS}			0		V

Tomando en cuenta las especificaciones de voltaje, es necesario utilizar una fuente de alimentación dual, es decir, que entregue tanto 1.8 como 3.3 [V] para el DSP. En consecuencia, para evitar agregar fuentes de alimentación de distintos voltajes, cualquier circuito integrado que se necesite deberá alimentarse con 1.8 o 3.3 [V].

---

[12] *I/O: Input/Output Port* o Puerto de Entrada/Salida.

[13] *Digital Core:* Núcleo de la CPU.

[14] *Flash Core:* Núcleo de la Memoria Flash interna.

Otra especificación que se entrega, y que la alimentación de la plataforma debe suplir, es la corriente que el DSP consume. Debido a que esta es una plataforma para varios diseños es necesario situarse en el caso más extremo para poder estimar la corriente total máxima que el DSP podría consumir. Se determina el modo "Operacional" como:

- Todos los relojes de los periféricos están activados.
- Todos los pines PWM están funcionando a 100 kHz.
- Se envían datos continuamente por los puertos SCI (A y B) y CAN.

En la Tabla 3.3 se encuentra la corriente consumida en distintas condiciones. Estas pruebas fueron realizadas con un reloj de sistema de 150 MHz (Sección 6.4 [10]).

Tabla 3.3 Corriente Consumida TMS320F2810

Modo	I_{DD}		I_{DDIO}		I_{DD3VFL}		I_{DDA}	
	Típico	Máximo	Típico	Máximo	Típico	Máximo	Típico	Máximo
Operacional	195 mA	230 mA	15 mA	30 mA	40 mA	45 mA	40 mA	50 mA
Idle[15]	125 mA	150 mA	5 mA	10 mA	2 µA	4 µA	1 µA	20 µA
Standby[16]	5 mA	10 mA	5 µA	20 µA	2 µA	4 µA	1 µA	20 µA

Observando la Tabla 3.3 se tiene que el consumo máximo será de 230 [mA] para 1.8 o 1.9 [V] y de 125 [mA] para 3.3 [V], siendo 195 y 95 [mA] los valores típicos de corriente consumida para 1.8 o 1.9 [V] y 3.3 [V] respectivamente.

Tomando en cuenta las especificaciones descritas se requiere de un Regulador de Voltaje Dual. Nuevamente, de la amplia gama de marcas y modelos existentes se prefiere utilizar, y esto será para la mayoría de los circuitos integrados a usar, un componente de la misma marca que el DSP *Texas Instruments*. El regulador escogido es un TPS70151 *Dual-Output Low-Dropout Voltage*

[15] Modo Ocioso: Memoria Flash apagada, relojes de periféricos encendidos.

[16] Modo en Reposo: Memoria Flash apagada, relojes de periféricos apagados.

Regulator o un Regulador de Voltaje de Baja Caída (Entrada-Salida) con Salida Doble. La Tabla 3.4 resume sus principales características eléctricas obtenidas de su hoja de datos [12].

Tabla 3.4 Especificaciones Eléctricas TPS70151

Parámetro	Descripción	Mínimo	Nominal	Máximo	Unidad
V_{IN}	Voltaje de Entrada	2.7		6	V
V_{OUT1}	Voltaje de Salida Regulador 1	3.234	3.3	3.366	V
V_{OUT2}	Voltaje de Salida Regulador 2	1.764	1.8	1.836	V
I_{OUT1}	Corriente de Salida Regulador 1	0		500	mA
I_{OUT2}	Corriente de Salida Regulador 2	0		250	mA
t_{RESET}	Duración del Pulso RESET	80	120	160	ms

De la Tabla 3.2, Tabla 3.3 y Tabla 3.4 se corrobora que el circuito integrado TPS70151 cumple con las especificaciones eléctricas necesarias para el correcto funcionamiento del DSP, tanto de rangos de voltajes como los valores máximos de corriente consumida. Además, cabe destacar que el regulador de 3.3 [V], dado que puede entregar hasta 500 [mA], queda con un margen de 300 [mA] para alimentar otros componentes de la plataforma y externos a esta, mediante un conector en la placa (por ejemplo: alimentación de una placa anexa de bajo consumo).

Otro punto importante es la condición temporal de alimentación del DSP. Este requiere una secuencia específica de los voltajes +1.8 y +3.3 [V] y de la señal RESET. Para esto existen dos opciones de encendido.

Opción 1: Un circuito externo de secuencia de alimentación habilita V_{DDIO} y luego V_{DD1} (+1.8 o +1.9 [V]). Después se activa la rampa de +1.8 [V] y luego la rampa de +3.3 [V] para la memoria *Flash* (V_{DD3VFL}) y el bloque ADC (V_{DDA1}/V_{DDA2}/$AV_{DDREFBG}$).

Opción 2: Se habilitan todo los pines de alimentación de +3.3 (V_{DDIO}, V_{DD3VFL}, V_{DDA1}/ V_{DDA2}/V_{DDAIO}/ $AV_{DDREFBG}$) y luego se activa la rampa de +1.8 [V]. Esta rampa no debe superar los +0.3 [V] antes que V_{DDIO} llegue a +2.5 [V].

Se utiliza la segunda opción para este diseño ya que es más simple de implementar, además de ser el recomendado por el fabricante. La Figura 3.3 muestra la secuencia de encendido requerida por el DSP. De la figura se observan los tiempos críticos de la secuencia, lo cuales son: retardo máximo entre rampas de 10 [ms]; y, duración mínima del pulso RESET (XRS) de 1 [ms].

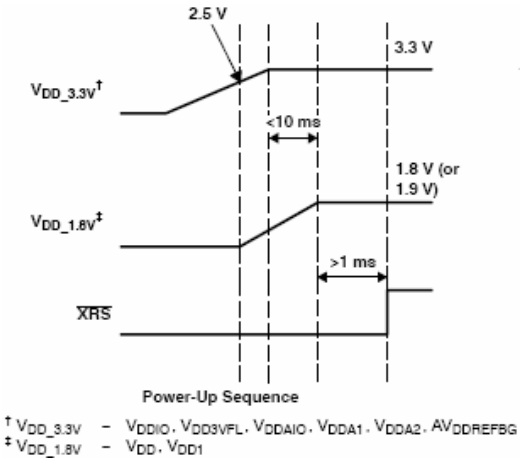


Figura 3.3 Secuencia de Encendido DSP

A pesar de que el integrado TPS70151 acepta hasta 6 [V], para proyectos futuros en que se tengan niveles de alimentación disponibles distintos a los recomendados (por ejemplo: una batería de 12 [V]) se decide agregar el circuito integrado LM7805 [13] el cual corresponde a un Regulador de Voltaje Lineal de Salida Fija que entrega una tensión continua nominal de 5 [V] y puede inyectar hasta 1 [A]. Este integrado acepta una tensión de entrada de 7 a 35 [V]. Debido a que en la salida existen 5 [V] fijos para alimentar el integrado TPS70151, la diferencia o caída de voltaje en la entrada quedará en el LM7805 el cual disipará aquella fracción de potencia no utilizada por la plataforma. Por este motivo no es recomendable alimentar la plataforma con más de 12 [V] para evitar excesos de temperatura en el regulador lineal LM7805. Como medida de precaución se le agregará un disipador de calor de aluminio al regulador de voltaje LM7805.

El disipador de calor a utilizar tiene una resistencia termal ($R_{\theta HS}$) de 24 [°C/W] y disipa una potencia de hasta 2.5 [W], el cual se puede ver en la Figura 3.4.

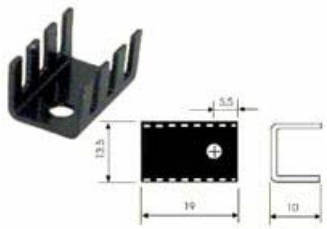


Figura 3.4 Disipador de Calor para LM7805

El regulador LM7805 posee una resistencia térmica de la juntura con el disipador ($R_{\theta JC}$) de 5 [°C/W] y de la juntura con el aire ($R_{\theta JA}$) de 65 [°C/W]. Además, este soporta una temperatura máxima (T_{max}) de 125 [°C]. Este valor es una restricción que se debe cumplir en la operación de la plataforma, donde para un voltaje de alimentación en particular se tiene la corriente máxima a consumir por el LM7805. Dado todo esto la Ecuación 3.1 muestra la relación entre voltaje y corriente de alimentación con los cuales se puede operar. Se supone para este caso una temperatura ambiente (T_{amb}) de 25 [°C].

Ecuación 3.1

$$T_{\max} = T_{amb} + \left(R_{\theta JC} + R_{\theta HS}\right) \times \left(V \cdot I\right)$$

La Ecuación 3.2 muestra la relación en el caso en que la potencia del LM7805 se disipe directamente en el aire, es decir, sin el uso de un disipador de calor.

Ecuación 3.2

$$T_{\max} = T_{amb} + R_{\theta JA} \times \left(V \cdot I\right)$$

Entonces, mediante las ecuaciones mostradas se calcula la curva de la corriente máxima a consumir en función de la tensión de alimentación del LM7805 considerando el disipador de calor de aluminio a utilizar. La Figura 3.5 muestra esta curva donde se compara la corriente máxima para el diseño sin disipador (color verde) y con el disipador de calor (color azúl).

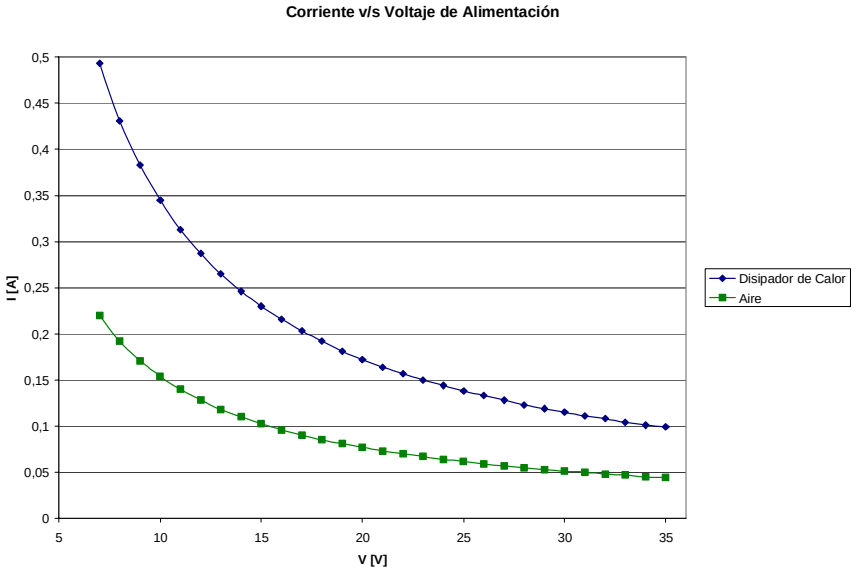


Figura 3.5 Curva Corriente Máxima en función del Voltaje de Alimentación

Por último, es importante utilizar condensadores de desacoplamiento cercanos a los pines de alimentación del componente DSP. Esto es debido principalmente a la conmutación a alta velocidad que ocurre dentro del DSP y que produce pequeñas oscilaciones en el voltaje de salida de una fuente regulada de tensión. Tales oscilaciones son interpretadas como ruido de alta frecuencia y podrían ocasionar inestabilidades en el funcionamiento del DSP. Para corregir este problema es que se utilizan tales condensadores. Para simplificar el cálculo de cada condensador se utiliza una Regla General Básica de Desacoplamiento [14], la cual propone los valores y cantidades típicas de los condensadores a utilizar. Además, existen algunas consideraciones para esta regla como el hecho de optimizar la capacitancia equivalente utilizando la mayor cantidad de condensadores como las dimensiones físicas del DSP lo permitan. La Figura 3.6 muestra la distribución propuesta de esta regla, desde una vista superior del DSP.

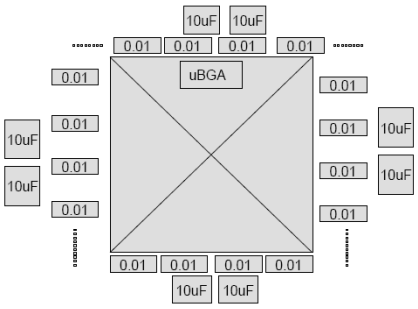


Figura 3.6 Regla General para Desacoplamiento

Reloj

Debido a la simplicidad y al bajo costo en que se incurre, como base del reloj oscilador del sistema se utilizará un cristal oscilador resonante de 25 MHz modelo CS2025 [15] de la marca Citizen America. Con esto y dado que el DSP puede multiplicar (Sección 3.8, 3.9 y 6.14 [10]), usando el bloque PLL[17], este valor hasta 5 veces, el reloj máximo del sistema a usar será de 125 MHz.

Comunicación

La comunicación de la plataforma está orientada a la conectividad con un computador personal mediante el módulo SCI (Sección 4.6 [10]) [16] del DSP. Este periférico emplea una comunicación asíncrona con un *baud rate* o tasa de transmisión configurable. Gracias a que el DSP posee dos módulos SCI (A y B) independientes es posible tener una comunicación serial con más de un *host*[18].

Típicamente en este tipo de controladores o microcontroladores se ha utilizado la comunicación serial RS232 para comunicar, y muchas veces programar, este tipo de elementos. De hecho, se puede comprobar de manera empírica que ante la necesidad de conectar un microcontrolador con un computador personal, por ejemplo, es prácticamente inmediato pensar en el clásico serial RS232. Pero debido a que las tecnologías en equipos computacionales avanzan bastante rápido, se puede ver que los equipos portátiles o *laptops* ya no poseen esta famosa interfaz ni mucho menos su conector, ya que la versatilidad y masificación del estándar USB ha reemplazado completamente a su antecesor.

Por esto mismo, es que esta plataforma de desarrollo debe contar con una interfaz USB y así dar un paso hacia adelante a las nuevas tecnologías. Además, para no dejar de lado la interfaz RS232, se tendrá incorporado el hardware necesario para tal función. Como se mencionó anteriormente el DSP cuenta con dos puertos seriales. En un puerto se tendrá comunicación USB

---

[17] *Phase Locked Loop:* Oscilador Enclavado por Fase.

[18] Equipo anfitrión.

y en el otro RS232 o viceversa, ya que se tendrá la opción de configurar en la misma placa el conexionado de los pines de transmisión y recepción de cada puerto SCI.

Para la comunicación vía USB se utilizará el circuito integrado CP2101 de la marca *Silicon Laboratories*, que es un *Single-Chip USB to UART Bridge* o un Puente USB a UART en un solo circuito [17]. Este circuito integrado tiene entre sus principales características la incorporación de un Regulador de Voltaje, un Oscilador y una Memoria EEPROM lo que lo hace completamente funcional añadiendo solo un par de componentes externos, ya que no necesita de fuentes de alimentación externas o un cristal oscilador resonante. Posee una tasa de transmisión que va desde los 300 a los 921.6 kbps[19]. La Figura 3.7 muestra el diagrama en bloques de la estructura interna de este circuito integrado.

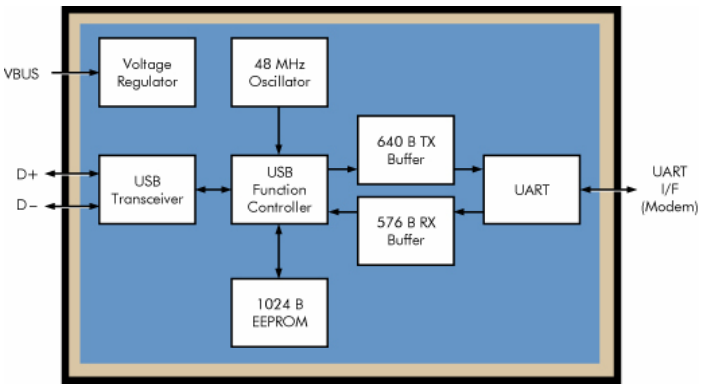


Figura 3.7 Diagrama en Bloques CP2101

Este circuito integrado se alimenta con 3.3 [V] por lo que es compatible con el bloque de alimentación de la plataforma. Además, tiene la opción de autoalimentarse mediante la conexión USB con el computador entregando hasta 500 [mA] a 3.3 [V] mediante un regulador de voltaje interno, esta opción no se ha utilizado ya que la plataforma posee alimentación propia. En otros diseños sería bastante práctico utilizar este modo ya que hace innecesario el uso de fuentes externas. En la Figura 3.8, Figura 3.9 y Figura 3.10 se muestran las distintas configuraciones que este integrado permite y que vale la pena destacar para su uso en futuras aplicaciones y que tomen como base o referencia el presente trabajo.

[19] Kilo Bits Por Segundos.

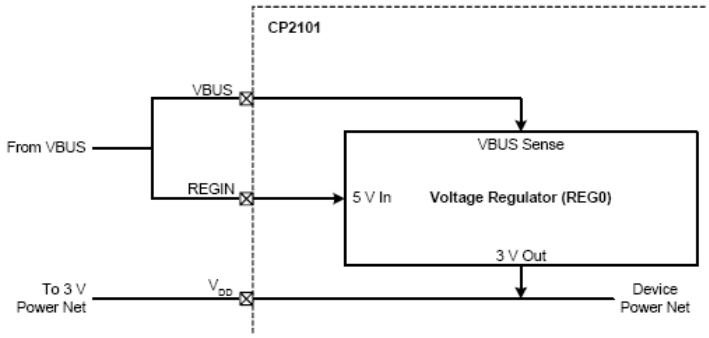


Figura 3.8 Alimentación por Bus USB

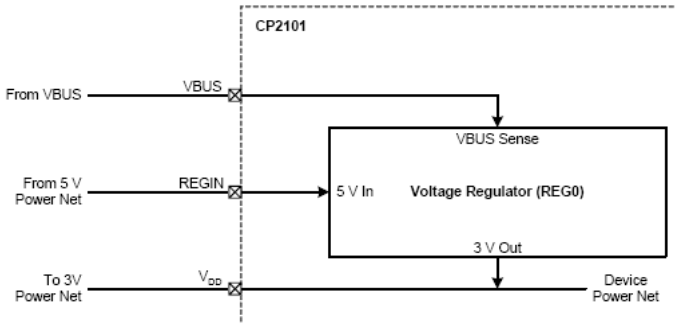


Figura 3.9 Alimentación Externa

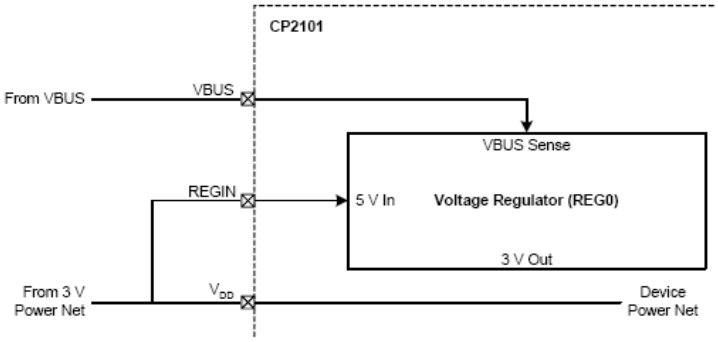


Figura 3.10 Alimentación Externa, Regulador interno no utilizado

En la parte del computador este dispositivo crea un *Virtual COM Port (VCP)* [18] o Puerto de Comunicación Virtual del tipo COM por lo que actúa de manera transparente entre la comunicación serial entre el computador y el DSP.

En cuanto a la comunicación vía estándar RS232 se utilizará el circuito integrado MAX3221 de Texas Instruments que es un *Single-Channel RS-232 Line Driver/Receiver* más conocido como un *transceiver*[20] serial RS232 de un solo canal [19].

Este circuito integrado se puede conectar a dispositivos con tensión de líneas de transmisión de 3 a 5.5 [V]. Hay que recordar que este voltaje es de 3.3 [V] para el DSP. Posee una tasa de transmisión máxima de 250 kbps bastante menor que la de su equivalente USB, otra razón más para preferir la comunicación USB sobre la serial RS232.

Conexiones

Los puntos antes descritos corresponden a bloques y conexiones externas necesarias entre la plataforma y el DSP principalmente. Pero existen módulos que son heredados directamente del DSP. Por esto mismo sólo es necesario dejar un conector para su conexionado posterior con otros dispositivos o placas.

- Conversor ADC
  - Conector 2x8 pines para 8 canales ADC.
- Puertos PWM/GPIO
  - 4 Conectores de 4 pines con 3 puertos PWM/GPIO cada uno.
- Puertos GPIO
  - GPIO1 Conector 2x5 pines 7 puertos GPIO.
  - GPIO2 Conector 4 pines 4 puertos GPIO.
- JTAG
  - Conector 2x7 pines. Conector estándar de 14 pines JTAG.
- Módulo de Comunicación SPI
  - Conector 5 pines.
- Módulo de Comunicación CAN
  - Mediante Conector GPIO 2.

[20] Transreceptor.

Diseño Final de Placa

El resultado de la etapa de diseño consiste en el *layout* o disposición de los diferentes componentes de la plataforma y su conexionado. La Figura 3.11 y Figura 3.12 muestran ambas caras del layout de la plataforma con el diseño final de la placa, diseño con el cual se realizará la fase de construcción y ensamble de la plataforma.

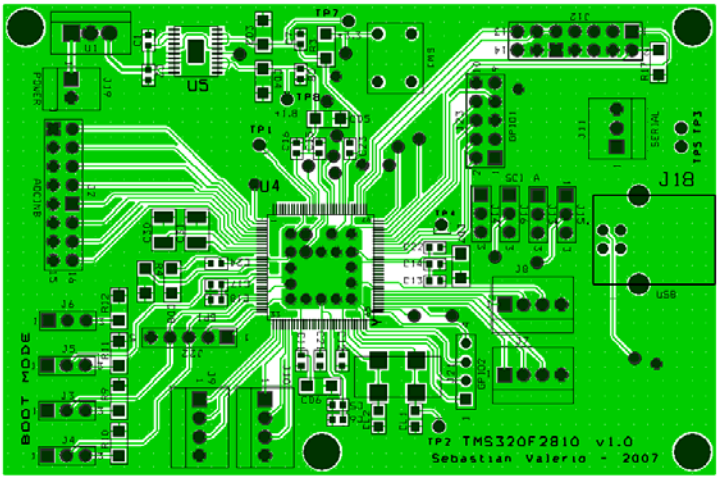


Figura 3.11 Cara Superior Plataforma

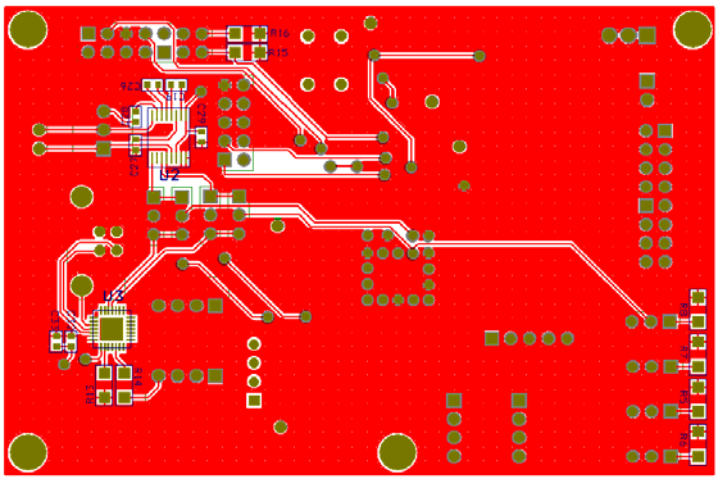


Figura 3.12 Cara Inferior Plataforma

La Tabla 3.5 muestra en detalle la descripción de cada conector y sus pines. Esto para facilitar la conexión de la plataforma y su correcta operación. Además, permite conocer los pines con que se puede conectar la plataforma a otras placas, por ejemplo, mediante el estándar serial SPI. Para mayor información se recomienda ver los esquemáticos en Anexo A.

Tabla 3.5 Detalle de Componentes y Conectores Plataforma de Desarrollo

Referencia	Pin	Descripción
U1		LM7805
U2		MAX3221
U3		CP2101
U4		TMS320F2810
U5		TPS70151
J2	Conector ADCINB	
	2 - 16	DSP_ADCINB[0-7]
	1 - 15	GND
J3 - J6		Modos de Booteo
J7	Conector PWM 1	
	1	DSP_GPIOA0 / DSP_PWM1
	2	DSP_GPIOA1 / DSP_PWM2
	3	DSP_GPIOA2 / DSP_PWM3
	4	GND
J8	Conector PWM 2	
	1	DSP_GPIOA3 / DSP_PWM4
	2	DSP_GPIOA4 / DSP_PWM5
	3	DSP_GPIOA5 / DSP_PWM6
	4	GND
J9	Conector PWM 3	
	1	DSP_GPIOB0 / DSP_PWM7
	2	DSP_GPIOB1 / DSP_PWM8
	3	DSP_GPIOB2 / DSP_PWM9
	4	GND
J10	Conector PWM 4	
	1	DSP_GPIOB3 / DSP_PWM10
	2	DSP_GPIOB4 / DSP_PWM11
	3	DSP_GPIOB5 / DSP_PWM12
	4	GND
J11	Conector RS232	

	1	SERIAL TX
	2	SERIAL RX
	3	GND
J12		Conector JTAG de 14 pines
J13 - J14		Configuración SCI A
J15 - J16		Configuración SCI B
J18		Conector USB Tipo B
J19	Conector de Alimentación	
	1	$+V_{CC}$
	2	GND
J23	Conector GPIO 1	
	1	DSP_GPIOE0
	2	DSP_GPIOA11
	3	DSP_GPIOE1
	4	DSP_GPIOA12
	6	DSP_GPIOA13
	7	DSP_GPIOA15
	8	DSP_GPIOA14
	9	+3.3 [V]
	5, 10	GND
J21	Conector GPIO 2	
	1	DSP_GPIOD5
	2	DSP_GPIOD6
	3	DSP_GPIOF6 / DSP_CANTX
	4	DSP_GPIOF7 / DSP_CANRX
J22	Conector SPI	
	1	DSP_SPISOMIA
	2	DSP_SPISIMOA
	3	DSP_SPISTEA
	4	DSP_SPICLKA
	5	GND

Especificaciones

Algunas especificaciones, que son el resumen de la etapa de diseño de la plataforma, se mencionan a continuación.

- Alimentación de Entrada
  - Voltaje: 7 a 35 [V].
  - Corriente Máxima: 750 [mA].
  - Corriente Máxima Asegurada: 500 [mA] (con 7 [V] en alimentación).
- Reloj del Sistema
  - Cristal Oscilador: 25 MHz.
  - Máximo con Multiplicador: 125 MHz.
- Comunicación USB
  - USB 2.0
  - Tasa de Transmisión Máxima 1 Mbps[21].
- Comunicación Serial RS232
  - Tasa de Transmisión Máxima 250 kbps.
- 12 puertos PWM
  - Voltaje: 3.3 [V].
- Hasta 23 puertos GPIO
  - Voltaje: 3.3 [V].
  - Frecuencia Máxima de Conmutación: 20 MHz.
- Conversor ADC de 8 Canales.
  - Voltaje de Entrada: 0-3 [V].
  - Reloj ADC: 1 kHz – 25 MHz.
  - Frecuencia Máxima de Muestreo: 12.5 MSPS[22].
  - Resolución: 12 bits.
- Buses de Comunicación
  - SPI

[21] Mega Bit Por Segundo.

[22] Mega Samples Per Second: Un Millón de Muestras Por Segundo.

- CAN

- Interfaz JTAG
  - Conector Estándar JTAG de 14 pines.
- RTOS
  - Habilitado para DSP/BIOS.

3.1.3. Implementación

Una vez que se tiene el *layout* se pasa a la fase de implementación o construcción de la plataforma. Esta fase consiste en la creación de la Placa de Circuito Impreso, más conocida como PCB[23], la adquisición de los componentes y finalmente el montaje de estos en la placa.

En la Figura 3.13 se puede ver la Plataforma de Desarrollo real la cual se ha denominado "TMS320F2810 v1.0" por el modelo de DSP utilizado y a la primera versión implementada.

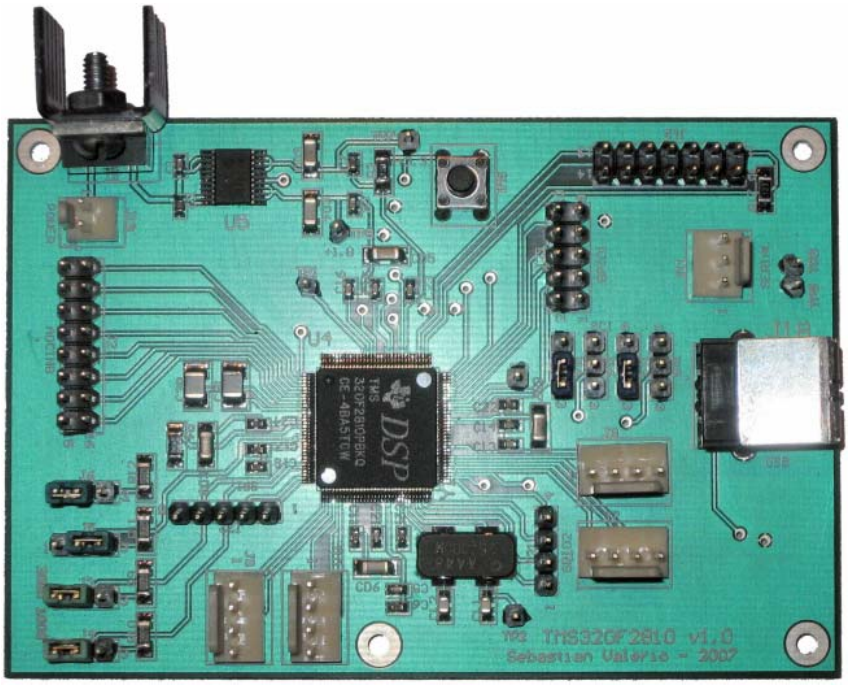


Figura 3.13 Plataforma de Desarrollo TMS320F2810 v1.0

[23] *Printed Ciruit Board.*

3.2. Prototipo Fresadora CNC

El propósito principal de este trabajo es el diseño y la implementación de la Plataforma de Desarrollo. Como se ha especificado anteriormente está orientada, en un principio, al control de dispositivos electromecánicos, para ser aún más precisos, a dispositivos o máquinas CNC. Para respaldar el diseño realizado se deben realizar numerosas pruebas, las que son fundamentalmente del tipo eléctricas.

Desde el punto de vista del autor de este trabajo no basta con tener una Plataforma de Desarrollo completamente especificada. Un aspecto muy importante y que muchas veces se deja de lado es el funcionamiento real de algún diseño en particular. En otras palabras, tener la posibilidad de obtener valores de variables o mediciones reales más allá de una simulación.

En vista de lo anterior es necesario contar con un banco de pruebas o *test bench*, el cual (para este trabajo en particular) consistirá en componentes de:

- Mecánica
  - Prototipo Fresadora.
- Hardware
  - Motores Eléctricos.
  - Electrónica de Control para Motores.
  - Sensores.

No está demás aclarar que este entorno de pruebas no está disponible por lo que es necesario implementarlo o construirlo, por lo que al igual que la plataforma debe pasar primero por una fase de diseño. Por otro lado, además de servir para pruebas y verificación del funcionamiento de la Plataforma, este prototipo podrá utilizarse de manera normal. En otras palabras, al finalizar este trabajo se contará con una "Máquina Fresadora CNC controlada por una Plataforma de Desarrollo en base a un DSP" totalmente funcional.

3.2.1. Requerimientos

Debido a que la construcción de este prototipo de Fresadora CNC no es el objetivo principal de este trabajo y considerando que, valga la redundancia, se trata de un Prototipo es que se optará por un diseño simple pero completamente funcional y acorde al propósito de este Trabajo de Título.

Mecánica

La máquina fresadora deberá contener a lo menos:

- 3 ejes independientes
    - Ejes X, Y y Z en coordenadas cartesianas.
- Sistema de Guía
    - Independiente para cada eje.
- Sistema de Transmisión
    - Independiente para cada eje.

Hardware

El movimiento de cada eje estará a cargo de un sistema de transmisión alimentado por un motor eléctrico. Para controlar cada motor se debe contar con la electrónica de control para el tipo de motor a utilizar. El requerimiento para esta etapa es que la entrada de voltaje de ésta acepte el nivel de 3.3 [V] como un valor "1 lógico" para evitar el uso de optoacopladores[24] entre esta etapa y la plataforma de desarrollo. Además, se debe contar con un sensor de borde con el fin de detectar el origen del sistema de ejes mecánico del prototipo. En resumen:

- Motor Eléctrico
- Electrónica de Control

[24] Aislador eléctrico óptico.

- Niveles Lógicos de entrada compatible con 3.3 [V].

- Sensores
  - Detector de Origen.

3.2.2. Diseño

Como se mencionó se busca de un diseño lo más simple posible pero que este sea completamente funcional en su aplicación. Dentro de las dimensiones físicas del prototipo de máquina se pretende que no sea superior a un metro lineal, tanto en el largo como en el ancho de su base.

Por otro lado, y como se ha mencionado anteriormente, el objetivo de construir este prototipo es complementario a la finalidad de este trabajo. Por ello, se propone la idea de reciclar materiales y componentes mecánicos ya existentes. Un buen ejemplo de esto son las impresoras de matriz de punto de carro ancho, ya que contienen: un sistema de transmisión; guías, y motores eléctricos; entre otros componentes. Además, tanto el sistema de guías como el de transmisión tienen un recorrido o largo suficiente para el propósito del prototipo.

Entonces, dado que se contará con los componentes básicos o esenciales para la construcción del prototipo es que en esta etapa se realizará el diseño de la estructura del mismo y de la electrónica necesaria para manejar el movimiento de cada motor eléctrico.

Los componentes reciclados que se obtuvieron son:

- Sistema de Transmisión
  - Correa dentada.
  - Poleas para correa dentada (Eje motor y retorno).
- Sistema de Guía
  - Barra Cilíndrica de Acero Inoxidable.
- Motores Eléctricos
  - Motor Stepper Unipolar.

Mecánica

El sistema de transmisión para los ejes X e Y es un conjunto entre una correa dentada y dos poleas (una en cada extremo de la correa). Con esto se logra convertir el movimiento rotacional del motor Stepper, acoplado a una de las poleas, en un movimiento lineal. En la Figura 3.14 se observan estos componentes.

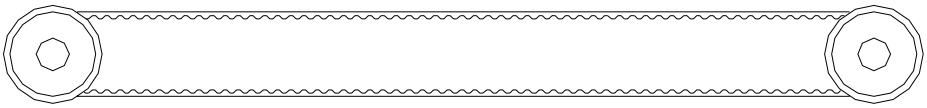


Figura 3.14 Correa Dentada con Poleas

Para el eje Z se utilizará un tornillo sinfín con una tuerca para producir el movimiento lineal a través del motor Stepper acoplado directamente en un extremo del tornillo. Como la tuerca se mantiene fija en su eje de rotación (no gira con el tornillo), esta se desliza a lo largo del tornillo sinfín. Se ha escogido este sistema ya que permite que la tuerca no se mueva (producto de la fuerza de gravedad) cuando el tornillo no está girando, lo que si podría suceder con las correas dentadas a menos que el motor estuviera frenado (mayor consumo de corriente). Además, este sistema presenta ventajas comparativas en cuanto a: precisión; torque; y, potencia transmitida. La Figura 3.15 muestra este sistema.

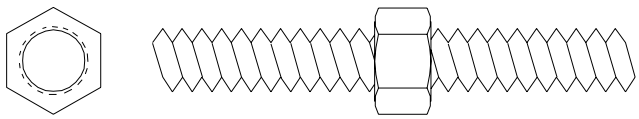


Figura 3.15 Tornillo sinfín con Tuerca

El sistema de guías para los ejes X e Y está conformado por un eje o barra de acero inoxidable. Este eje tiene 480 [mm] de largo y 12 [mm] de diámetro lo que permite un recorrido adecuado para el diseño del prototipo. Para realizar el movimiento se ha diseñado un deslizador el cual recorrerá el eje de acero. Este deslizador será de Technyl, el cual es suficientemente rígido y su roce con la barra de acero es adecuadamente bajo. La Figura 3.16 muestra las caras de este deslizador donde se pueden ver las perforaciones para la barra de acero y para los tornillos de fijación a la base correspondiente. La Figura 3.17 muestra el montaje de este sistema.

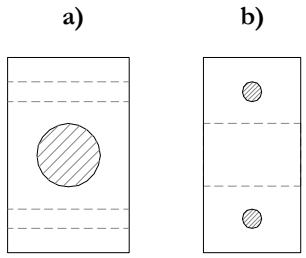


Figura 3.16 Deslizador. a) Cara lateral. b) Cara frontal.

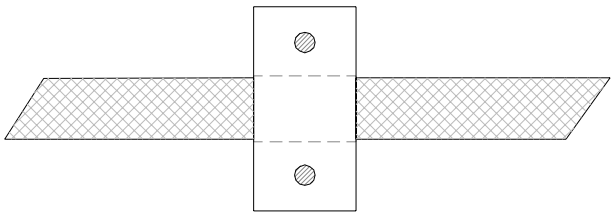


Figura 3.17 Sistema de Guiado Lineal

Para el eje Z se utiliza el mismo sistema pero el eje es una barra de acero inoxidable de 6,5 [mm] de diámetro.

Entonces, se tienen definidos los sistemas de guía y de transmisión de cada eje. El cabezal del prototipo de máquina es el lugar que contiene la herramienta que realizará el fresado. Este cabezal está directamente acoplado en el eje Z, es decir, el cabezal se mueve directamente sobre este eje. A su vez el conjunto mecánico del eje Z está acoplado directamente al carro que se mueve a lo largo del eje X. Finalmente, sobre el eje Y se mueve todo el eje X (junto al eje Z). De esta manera se puede realizar el movimiento a cualquier punto dentro del área mecánica del prototipo de máquina fresadora.

Hardware

El motor Stepper a utilizar es el modelo PJJQ114ZA-K de la marca *Kyushu Matsushita Electric*. Las características de este motor obtenidas desde su placa se pueden ver en la Tabla 3.6.

Tabla 3.6 Características Motor Stepper

Tipo	Unipolar
Voltaje Nominal	3.1 [V]
Impedancia Enrollado	3.4 [Ω]
Avance	1.8 °/paso

La Figura 3.18 muestra el motor Stepper a utilizar. Además, se puede observar la placa con las características principales de este motor.

Figura 3.18 Motor Stepper PJJQ114ZA-K

A pesar de que el motor Stepper es del tipo Unipolar, este puede ser utilizado en modo Bipolar realizando las conexiones que se indican en la Figura 3.19. Con esto se obtiene el doble de torque y la corriente consumida se reduce a la mitad con respecto al modo Unipolar.

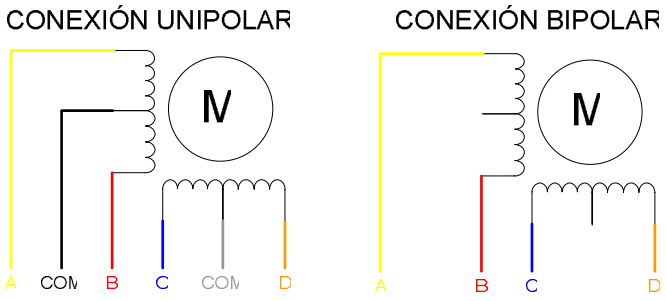


Figura 3.19 Conexión en Modo Unipolar y Bipolar del Motor Stepper

El control de cada motor Stepper se realizará en base a dos circuitos integrados: el L297 de *STMicroelectronics* que es un *Stepper Motor Controller* o Controlador de Motor Paso a Paso [20]; y, el L298 del mismo fabricante que corresponde a un *Dual Full-Bridge Driver* o Controlador de Puente Completo Doble [21].

El L297 es el encargado de recibir ciertas señales de control y con esto generar la secuencia de encendido de cada bobina del motor. Luego, esta secuencia es enviada al L298 el cual activa cada una de sus salidas (conectadas al motor) dependiendo de la secuencia enviada. Es decir, el circuito integrado L297 corresponde a la etapa de control mientras que el L298 corresponde a la etapa de potencia del controlador de motor Stepper.

Dentro de las señales de control que el L297 recibe, se utilizarán:

- *Clock o Reloj*
    - Reloj de Paso. Esta señal corresponde a la frecuencia con que el controlador realizará la secuencia, es decir, a la frecuencia de giro del motor.
- *Clockwise/Counterclockwise Direction (CW/CCW) o Sentido Horario/Antihorario*
    - Dirección en el sentido o en contra de las agujas del reloj. Sentido de giro del motor.
- *Enable o Habilita*
    - Habilitador del sistema de control. Permite activar o desactivar el movimiento del motor.

Estas señales son de tipo binaria, es decir, corresponden a un 0 o 1 lógico. En la Tabla 3.7 se observan las características eléctricas del circuito integrado L297.

Tabla 3.7 Especificaciones Eléctricas L297

Parámetro	Descripción	Mínimo	Típico	Máximo	Unidad
V_S	Voltaje de Alimentación	4.75		7	V
I_S	Corriente de Entrada		50	80	mA

V_{IL}	Voltaje de Señal de Entrada (Nivel BAJO)		0.6	V
V_{IH}	Voltaje de Señal de Entrada (Nivel ALTO)	2	V_S	V
V_{EL}	Voltaje de Señal de Habilitación (Nivel BAJO)		1.3	V
V_{EH}	Voltaje de Señal de Habilitación (Nivel ALTO)	2	V_S	V

La plataforma DSP entrega un 1 lógico con un valor de 3.3 [V]. De lo que se puede ver en la Tabla 3.7 el valor menor con que se detecta o acepta como un 1 lógico es de 2 [V] lo que es bastante inferior a los 3.3 [V] entregados por la plataforma. Se tiene un margen de 1.3 [V] de seguridad para la activación de estas señales de control. Como el L297 corresponde a lógica TTL[25] no es necesaria realizar ninguna conexión extra entre el DSP y el controlador L297 [22].

El L297 tiene cuatro salidas lógicas (A, B, C y D) que corresponden a la secuencia con la cual se alimentarán las bobinas del motor Stepper. Esta alimentación, como se mencionó anteriormente, la realiza el circuito integrado L298 la cual recibe esta secuencia mediante los pines A, B, C y D además de otros dos de habilitación (INH1 e INH2) para cada puente del L298. El L298 posee cuatro pines de salida los que se conectan a cada conector de las dos bobinas a utilizar del motor Stepper. La Figura 3.20 muestra el esquema de la conexión del controlador de motor Stepper.

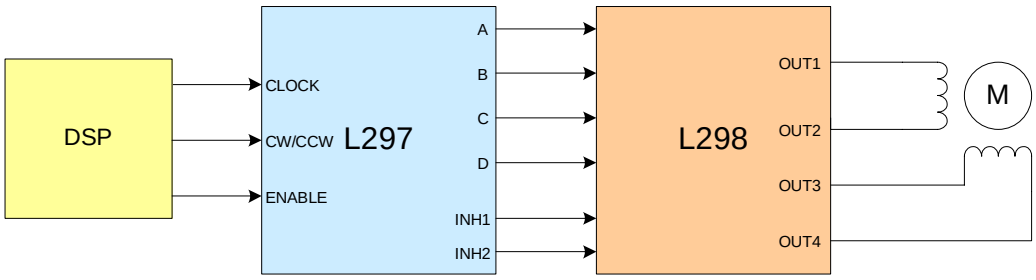


Figura 3.20 Esquema Controlador Motor Stepper

[25] *Transistor-Transistor Logic* o Lógica Transistor a Transistor

En la Tabla 3.8 se observan las características eléctricas del integrado L298. Con estos datos, más los del circuito integrado L297 se tendrán los parámetros necesarios para definir los voltajes de alimentación y la corriente necesaria para la placa controladora del motor Stepper.

Tabla 3.8 Especificaciones Eléctricas L298

Parámetro	Descripción	Mínimo	Típico	Máximo	Unidad
V_S	Voltaje de Alimentación Potencia	V_{IH}+2.5		46	V
V_{SS}	Voltaje de Alimentación Lógica	4.5	5	7	V
I_S	Corriente de Entrada Potencia	4		70	mA
I_{SS}	Corriente de Entrada Lógica	3		36	mA
V_{IL}	Voltaje de Señal de Entrada (Nivel BAJO)	-0.3		1.5	V
V_{IH}	Voltaje de Señal de Entrada (Nivel ALTO)	2.3		V_{SS}	V
I_O	Pico de Corriente de Salida (por Canal)			3	A

La alimentación de la placa controladora se realizará mediante una fuente externa de voltaje, la cual entrega 5 [V] y 12 [V] de corriente continua. Los 5 [V] se utilizarán para energizar el L297 y el circuito de lógica del L298 (pin V_{SS}). El voltaje de alimentación de potencia del L298 es el mismo con el cual se alimenta cada bobina del motor Stepper. Dado que el circuito L298 necesita alimentarse con un voltaje mínimo de V_{SS}+2.5 [V] y siendo V_{SS} de 5 [V], se necesitarán como mínimo 7.5 [V]. Entonces, el voltaje de potencia será de 12 [V] ya que es el valor disponible más cercano al requerido.

Con todas las especificaciones ya mencionadas se obtiene la Placa Controladora de Motor Stepper. La Figura 3.21 muestra el *layout* de la placa resultante.

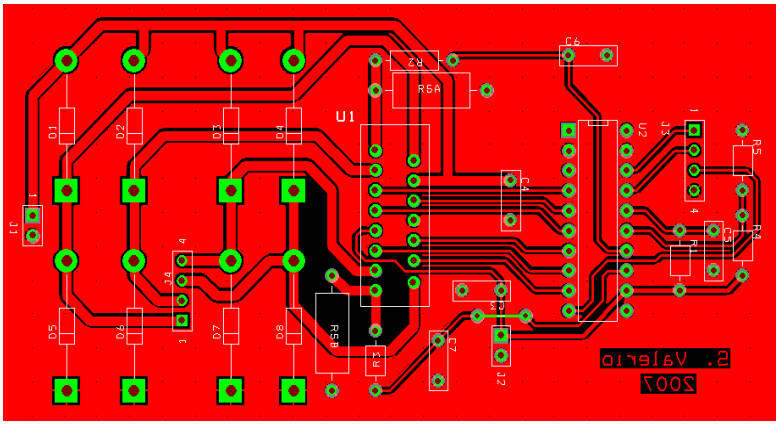


Figura 3.21 Layout Placa Controladora Motor Stepper

Como se puede ver en la esta placa tiene solamente una cara debido a los componentes que se utilizaron en su diseño. La nomenclatura utilizada en la placa se ve en la Tabla 3.9.

Tabla 3.9 Placa Controladora Motor Stepper

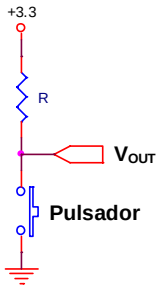
Referencia	Pin	Descripción
U1	Circuito Integrado L298	
U2	Circuito Integrado L297	
J1	Conector Alimentación 1	
	1	Alimentación V_S (L298)
	2	GND
J2	Conector Alimentación 2	
	1	Alimentación V_S (L297) y V_{SS} (L298)
	2	GND
J3	Conector Plataforma DSP (Control)	
	1	CLOCK
	2	CW/CCW
	3	ENABLE
	4	GND
J4	Conector Motor Stepper	
	1	A
	2	B

	3	C
	4	D

Por último, es necesario contar con Detectores del Origen del sistema de ejes mecánicos. Es decir, se requieren detectores de bordes con el propósito de que los motores puedan moverse hasta la posición inicial u origen del prototipo.

El diseño más simple, y el que se va a utilizar, es en base a pulsadores ubicados en el borde deseado de cada eje. Cuando la base de cada eje active uno de estos pulsadores, estos enviarán una señal hacia la plataforma DSP. Esta señal es del tipo binaria nuevamente ya que se necesita saber si es que la base está o no en el origen.

Del conector GPIO1 de la plataforma de desarrollo DSP se dejaron disponibles dos pines, uno conectado a 3.3 [V] y el otro conectado a la Tierra (GND) de la misma plataforma. Con esto, la placa de sensores o detectores puede alimentarse desde la misma plataforma y, consecuente a ello, se tendrá el voltaje requerido por una entrada digital del dispositivo DSP. La Figura 3.22 muestra el circuito a utilizar para este propósito.



Pulsador	V_{OUT}	Señal Digital
0	3.3 [V]	1
1	0 [V]	0

Figura 3.22 Esquemático Circuito Pulsador

Este circuito simple es replicado y se obtiene una placa capaz de conectar cinco pulsadores. La resistencia a utilizar será de un valor de 10 [kΩ] por lo que el consumo de corriente cuando el pulsador esté activado será de tan solo 0.33 [mA]. Finalmente se tiene la Placa de Sensores, la cual se puede observar en la Figura 3.23.

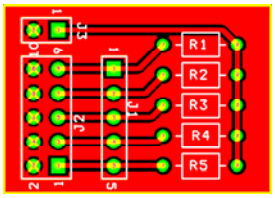


Figura 3.23 Layout Placa Sensores

Además, se especifica los conectores disponibles para esta placa los cuales se encuentran en la Tabla 3.10.

Tabla 3.10 Placa de Sensores

Referencia	Pin	Descripción
J1	Conector Salidas Digitales (DSP)	
	1-5	Salida Digital 1-5
J2	Conector Pulsadores	
	2-10	Señal Pulsador 1-5
	1-9	GND
J3	Conector Alimentación	
	1	+3.3[V]
	2	GND

3.2.3. Implementación

La construcción del prototipo se realizó en base al diseño propuesto en 3.2.2 con los componentes reciclados disponibles. El resultado al que se llegó fue satisfactorio para su estructura mecánica y electromecánica (motores).

Mecánica

En las Figura 3.24 y Figura 3.25 se observa con claridad el resultado de la fase de construcción del prototipo de máquina fresadora CNC. Se utiliza una herramienta multipropósito de la marca Dremel como elemento de fresado, la cual se puede ver en las figuras referenciadas.

El material utilizado en la construcción del prototipo es, en su gran mayoría, madera MDF[26] de 12 y 15 [mm] de espesor. Además, se utilizó planchas de aluminio de 2 [mm] de espesor. Dadas estas características, se puede decir que la estructura del prototipo es bastante rígida.

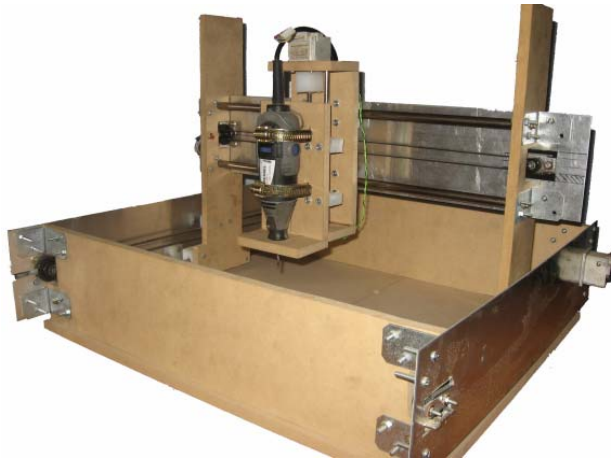


Figura 3.24 Prototipo Máquina Fresadora CNC (Vista Aérea)

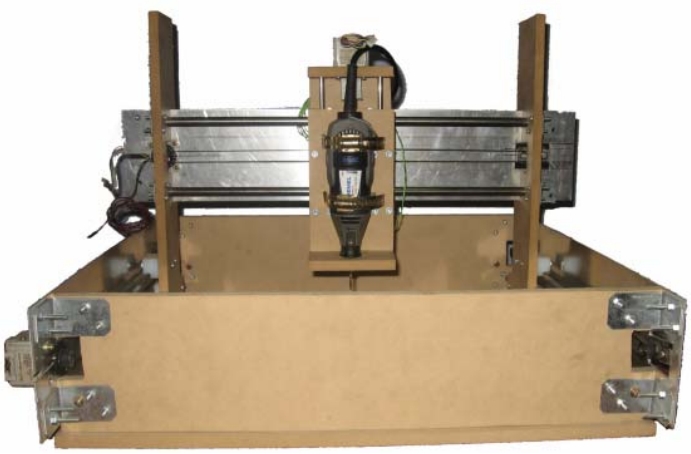


Figura 3.25 Prototipo Máquina Fresadora CNC (Vista Frontal)

La Figura 3.26 muestra una vista superior de la máquina donde se puede ver el sistema de ejes en coordenadas cartesianas utilizado. Desde esta vista el Eje X corresponde al eje horizontal y el Eje Y al vertical, mientras que el Eje Z es encuentra saliendo desde la base de la máquina. Además, se encuentra detallada en la figura la posición de cada uno de los motores Stepper utilizados.

[26] *Medium Density Fiberboard* o Tablero de Fibra de Densidad Media.

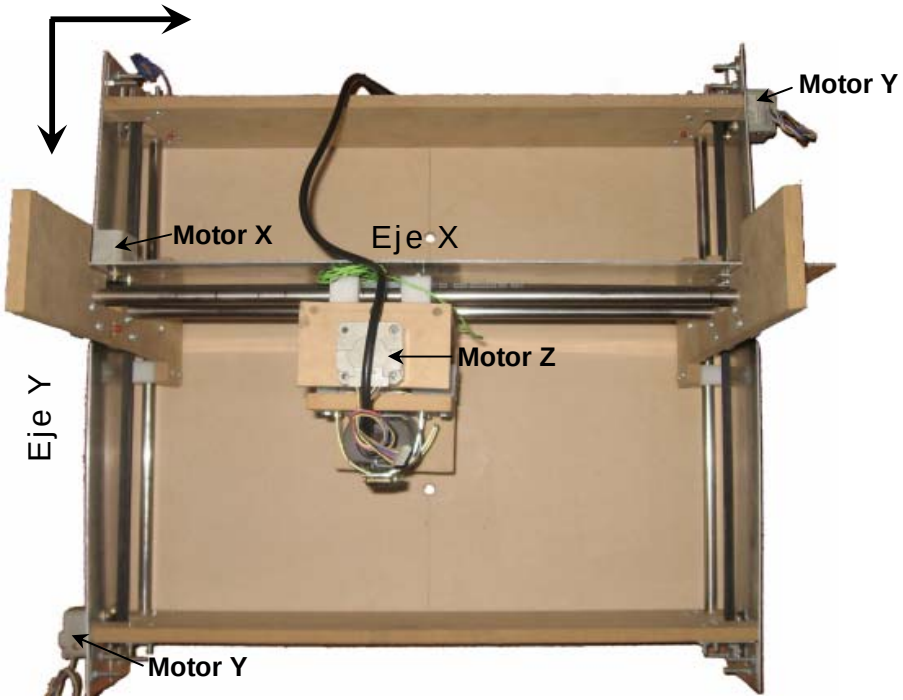


Figura 3.26 Detalle Prototipo Máquina Fresadora CNC (Vista Superior)

Cada uno de los ejes cuenta con un sistema independiente de guía y otro de transmisión. Para los ejes X e Y se utiliza el mismo tipo de guía y de transmisión, los cuales están detallados en 3.2.2. La Figura 3.27 muestra una vista más detallada del eje X, mientras que la Figura 3.28 hace un acercamiento al montaje de la polea acoplada al motor con la correa dentada de transmisión. En la Figura 3.29 se observa el mismo detalle para el Eje Y.

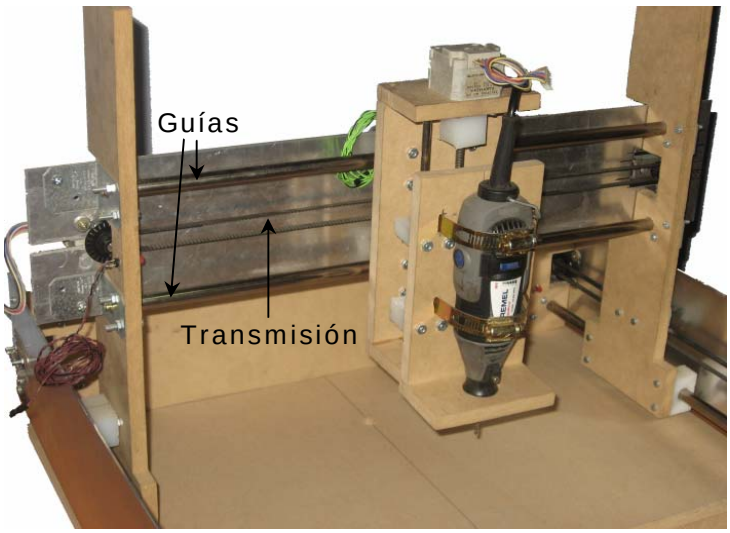


Figura 3.27 Eje X Prototipo

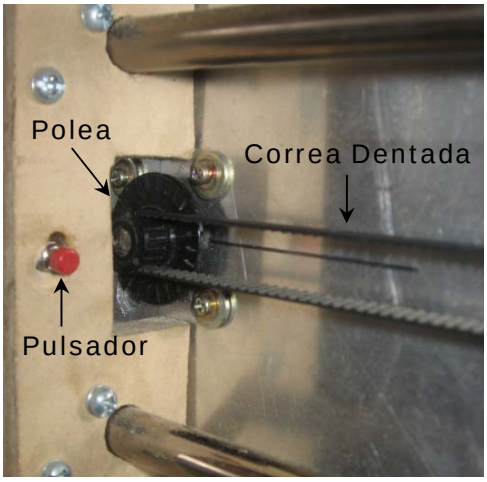


Figura 3.28 Montaje Polea y Correa Dentada Eje X

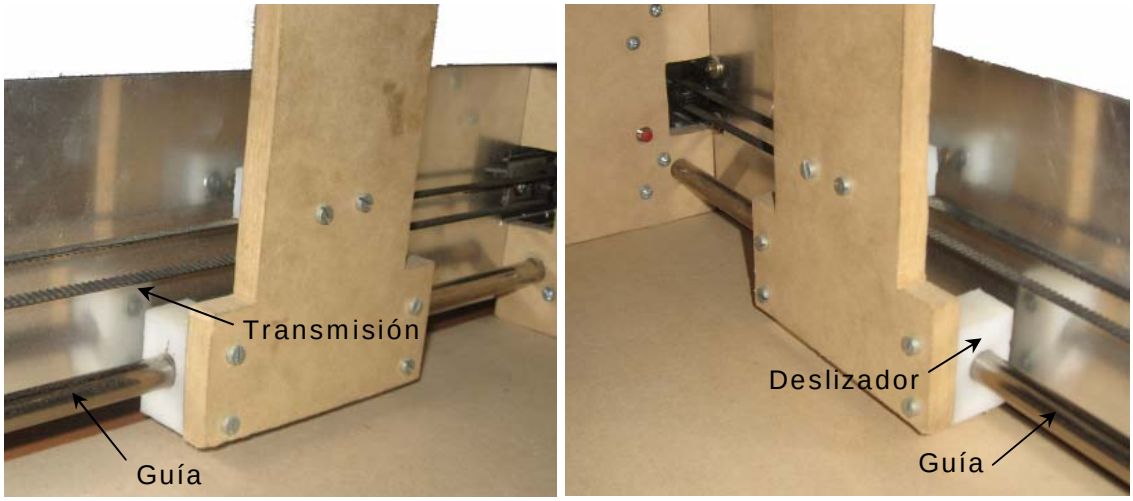


Figura 3.29 Eje Y Prototipo

El sistema de guía y de transmisión del Eje Z es distinto al de los mostrados anteriormente. La Figura 3.30 muestra el detalle de este eje donde se puede ver que la mecánica utilizada es más complicada que las anteriores, esto debido al uso del tornillo sinfín como sistema de transmisión mecánica. En la Figura 3.31 se puede observar en detalle el montaje del tornillo con la tuerca unida a un componente de fijación para realizar el movimiento lineal.

Figura 3.30 Eje Z Prototipo

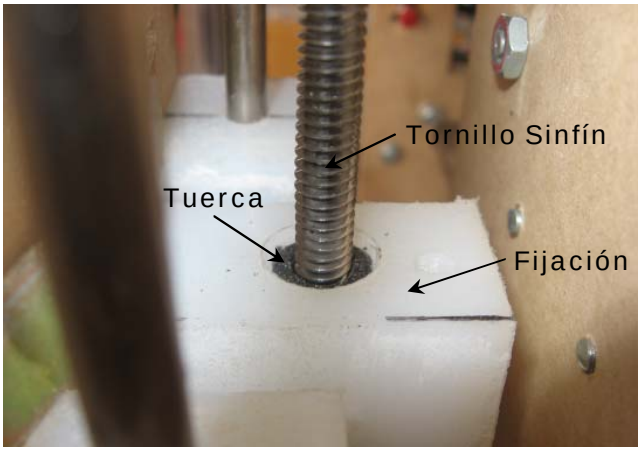


Figura 3.31 Sistema de Transmisión Eje Z

Finalmente, la Figura 3.32 muestra el Cabezal de Fresado el cual, mediante la mecánica implementada, puede moverse en las tres direcciones del sistema de ejes en coordenadas cartesianas de manera independiente. La posición del cabezal define cada punto (X, Y, Z) en el área utilizable de la máquina.

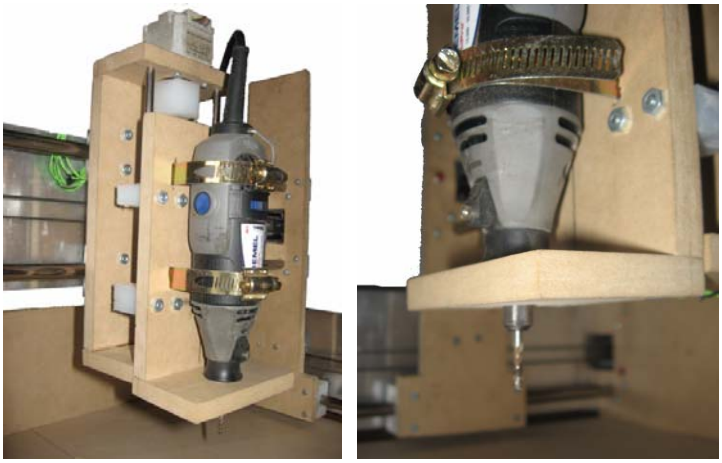


Figura 3.32 Cabezal de Fresado

Hardware

En las Figura 3.33 y Figura 3.34 se observan las placas de circuito impreso construidas de la Controladora de Motor Stepper y Sensores respectivamente. En ambos casos se observa la vista superior de las placas (Capa de Componentes).

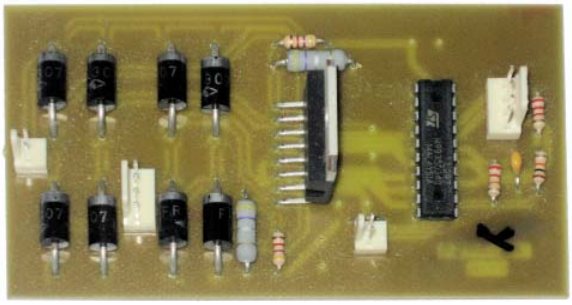


Figura 3.33 Placa Controladora Motor Stepper

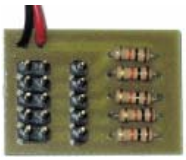


Figura 3.34 Placa Sensores

3.3. Software

Todo el hardware diseñado anteriormente no servirá de mucho si es que no tiene algún medio que pueda controlarlo. Este medio, en este caso, se refiere al dispositivo DSP el cual necesita de un programa específico que maneje los distintos periféricos de este. El programa será el encargado de enviar las señales de control necesarias para el movimiento de cada motor Stepper que se desee realizar.

3.4. Código DSP

Se puede decir que el programa es lo que define, en gran parte, una aplicación. Esto tiene mayor importancia para una Plataforma de Desarrollo, debido a que el hardware disponible en la plataforma es el mismo para distintas aplicaciones y lo único diferente será su código. Por ejemplo, con esta misma plataforma se podrán enviar las señales eléctricas de control para un Inversor Trifásico y realizar un procesamiento sobre una muestra de audio digitalizada, sólo cargando un código distinto para cada propósito.

3.4.1. Requerimientos

Para esta etapa es necesario que el código a implementar realice las siguientes tareas dentro del dispositivo DSP:

- Comunicación Serial Asíncrona
  - Envío y recepción de información hacia el computador *(host)*.
- Decodificación de Instrucciones
  - Instrucciones enviadas por el *host*.
  - Definir acciones en base a las Instrucciones.
- Generar señales de control para Motor Stepper
  - Señal oscilante de reloj (CLOCK).
  - Señal binaria de dirección o sentido de giro (CW/CCW).

- Señal binaria de habilitación (ENABLE).

- Lectura de Señales de Sensores
  - Habilitación de Puertos de Entrada Digitales.
  - Definir acciones de control en base a sensores.

3.4.2. Diseño

Para poder realizar todo el código y poder compilarlo se utilizó el software de *Texas Instruments "Code Composer Studio* Versión 3.1.0" [23]. Además, es necesario contar con los encabezados o *header* de los periféricos para el DSP en específico. Estos encabezados sirven para inicializar y utilizar los distintos periféricos disponibles en el DSP, tales como el módulo de comunicación SCI o los puertos GPIO. Estos encabezados están disponibles y no son producto de este trabajo [24].

Configuración

El primer paso, en cuanto al código, es definir el multiplicador con el cual el módulo PLL aumentará la frecuencia de entrada proveniente del cristal resonante de 25 MHz. En este caso se utiliza un multiplicador por 4 lo que lleva la frecuencia del sistema (SYSCLK) a 100 MHz, configurando el registro PLLCR [10][25].

El segundo paso para es definir e inicializar la comunicación con el equipo *host* que proveerá la información necesaria para la plataforma. Como se dijo anteriormente la comunicación se realizará mediante el módulo SCI a través de la interfaz USB. Los parámetros que se deben definir para el SCI son: cantidad de bits a transmitir; modo de transmisión; y, tasa de transmisión.

La tasa de transmisión se calcula de acuerdo a la Ecuación 3.3. Donde *Baud Rate* es la tasa de transmisión, LSPCLK es el *Low Speed Peripheral Clock* o Reloj de Periféricos de Baja Velocidad y BRR es el *Baud Rate Register* o Registro de la Tasa de Transmisión que es la concatenación de dos registros (SCIHBAUD y SCILBAUD) [16].

Ecuación 3.3

$$\text{Baud Rate} = \frac{\text{LSPCLK}}{(\text{BRR}+1)\times 8}$$

La Tabla 3.11 muestra los principales registros que se deben configurar del módulo de comunicación SCI.

Tabla 3.11 Principales Registros Módulo SCI

Bits	Registro	Descripción
0-2	SCICCR	Selecciona el largo de bits por carácter (de uno a ocho bits).
5	SCICCR	Habilita (1) o deshabilita (0) la paridad.
6	SCICCR	Par (1) o Impar (0).
7	SCICCR	Bit de parada. Un bit (0) o dos bits de parada (1).
0-15	SCILBAUD	Bits menos significativos de la tasa de transmisión.
0-15	SCIHBAUD	Bits más significativos de la tasa de transmisión.

En este caso, la tasa de transmisión será de 57600 bps y se tiene un LSPCLK de 25 MHz (esto dada la configuración del reloj del sistema). Entonces, de la Ecuación 3.3:

$$\text{BRR} = \frac{\text{LSPCLK}}{8\times(\text{Baud Rate})} - 1$$

$$\text{BRR} = \frac{25\times 10^6}{8\times 57600} - 1$$

$$\text{BRR} = 53$$

Se necesita habilitar doce puertos GPIO como salidas, estos serán las señales de control para la placa controladora del motor Stepper. Como se requiere controlar cuatro motores: uno para el eje X; dos para el eje Y; y, uno para el eje Z; cada uno se controla mediante tres señales (CLOCK; CW/CCW; y, ENABLE). Por esto es que se utilizan 12 salidas GPIO. Los registros que se deben configurar para que estos pines operen como puertos GPIO son, en este caso: GPAMUX; GPADIR; GPBMUX; y, GPBDIR. Físicamente, estos puertos corresponden a los conectores PWM/GPIO de la plataforma de desarrollo. Además, se requiere habilitar cuatro entradas GPIO para las señales de la placa de sensores (pulsadores), las cuales se encuentran

cableadas al conector GPIO2 de la plataforma. En este caso se deben configurar los registros: GPDMUX; GPDDIR; GPFMUX; y, GPFDIR. El detalle de estos registros de encuentra en la Tabla 3.12 [25].

Tabla 3.12 Principales Registros Módulo GPIO

Bits	Registro	Descripción
0-5	GPAMUX	Multiplexador puerto A. PWM (1) o GPIO (0).
0-5	GPADIR	Dirección puerto A. Lectura (1) o Escritura (0).
0-5	GPBMUX	Multiplexador puerto B. PWM (1) o GPIO (0).
0-5	GPBDIR	Dirección puerto B. Lectura (1) o Escritura (0).
5-6	GPDMUX	Multiplexador puerto D. Función Principal (1) o GPIO (0).
5-6	GPDDIR	Dirección puerto D. Lectura (1) o Escritura (0).
6-7	GPFMUX	Multiplexador puerto F. CAN (1) o GPIO (0).
6-7	GPFDIR	Dirección puerto F. Lectura (1) o Escritura (0).

Por último, se debe configurar un *Timer* o Temporizador interno el cual servirá como base de tiempo para la señal de reloj de los motores Stepper. Es decir, será el equivalente a la frecuencia máxima a la cual los motores Stepper podrán girar. Hay que recordar que la plataforma puede trabajar con un reloj del sistema de hasta 125 MHz, un valor demasiado grande para las velocidades con que el motor puede girar. La función de este temporizador es generar cada cierto tiempo una interrupción en el DSP, con lo cual se podrán generar las señales de control y realizar tareas anexas tales como la lectura de sensores.

Para ejemplificar el uso de este recurso se tendrá el siguiente caso. Se necesita enviar un tren de pulsos de la señal CLOCK cada 4 [ms]. Para esto el temporizador deberá interrumpir al DSP cada 2 [ms] e invertir el valor lógico de la señal CLOCK en cada interrupción, es decir, si hay un 1 lo cambiará por un 0 y viceversa. Este ejemplo se puede ver en la Figura 3.35.

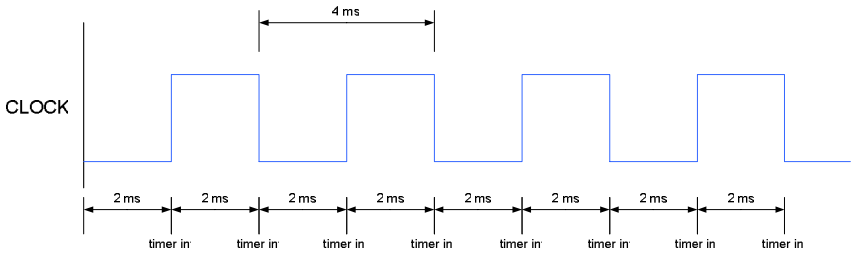


Figura 3.35 Ejemplo Recurso *Timer*

Los registros necesarios para configurar el *timer* son: T1PR; y, T1CON. El primero es el registro del período del *timer* mientras que el segundo es el registro de control del mismo [26][27].

Instrucciones

Se define un grupo de instrucciones con lo cual el computador *host* y la plataforma DSP podrán entenderse. Las instrucciones son elementales para el funcionamiento del prototipo de fresadora siendo muy simples en su contenido. Estas serán enviadas por el computador y ejecutadas completamente por el dispositivo DSP. A continuación se muestran las instrucciones a utilizar y una breve descripción de su funcionamiento:

- CERO: establece un cero (origen del sistema de ejes de coordenadas) relativo en la actual posición del cabezal del prototipo.
- MOVER: mueve el cabezal a la posición deseada.
- LINEA: dibuja una línea recta entre dos puntos (X, Y, Z) entregados.
- CIRCULO: dibuja un circulo con centro en un punto en particular (X, Y, Z) y con un radio R.
- ACK: respuesta del DSP hacia el computador indicando la ejecución correcta de la instrucción al terminar.

Debido a que la transmisión serial permite el envío de datos en forma binaria a 8 bits, se ha establecido una codificación para el grupo de instrucciones. La Tabla 3.13 muestra esta codificación a 8 bits y los parámetros que cada instrucción necesita.

Tabla 3.13 Set de Instrucciones

Instrucción	Codificación (Hexadecimal)	Parámetros
CERO	0x10	Sin parámetros.
MOVER	0x11	Punto Final: X2, Y2, Z2.
LINEA	0x12	Punto Inicial: X1, Y1, Z1. Punto Final: X2, Y2, Z2.
CIRCULO	0x13	Punto Central: X1, Y1, Z1. Profundidad: Z2 Radio: R
ACK	0x01	Sin parámetros.

Formato

Los parámetros de cada instrucción son coordenadas o distancias absolutas en el caso del radio del círculo. Estos, por definición, tienen unidades de centímetros. Se ha escogido el uso de dos bytes en secuencia para transmitir cada parámetro hacia el DSP. El primer byte (8 bits) se utiliza para enviar la parte entera mientras que en el segundo se transmite la parte decimal. En este caso, dado que se utilizan 8 bits para cada parte, se podrá enviar un parámetro con valor máximo de 255.255 [cm] (donde el punto separa la parte entera de la decimal). En la Figura 3.36 se observa con mayor claridad este procedimiento.

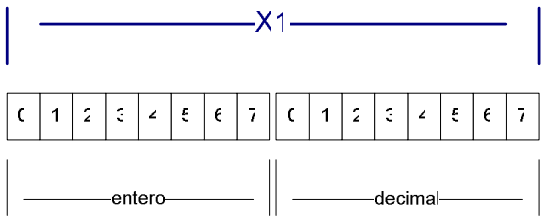


Figura 3.36 Formato de Parámetros

Movimientos

El resultado visible de los movimientos de cada motor Stepper es un conjunto de procesos los cuales son desarrollados por el DSP y el controlador Stepper diseñado en 3.2.2 (Hardware). El DSP genera: el reloj de paso; el sentido de giro; y, la habilitación del controlador. A partir de esto el controlador ejecuta una secuencia de pasos. Avanzará cierta cantidad de pasos del motor Stepper de acuerdo a la cantidad de pulsos recibidos en el reloj y el motor se moverá a la frecuencia de este mismo reloj. Es decir, si se envían 100 pulsos a una frecuencia de 1 Hz, el motor girará en 100 pasos a una frecuencia de un paso por segundo. En base a esta información se define la forma en la cual se generarán las señales de control para el correcto funcionamiento de cada instrucción.

Mover

Este es el caso más simple puesto que dada la posición actual en que se encuentra el cabezal se deberá mover una cierta distancia en la dirección correcta. Por ejemplo, si el cabezal se encuentra en el punto (3, 4, 3)[27] y se desea moverlo a la posición (7, 2, 3), el cabezal se deberá mover 4 espacios hacia la derecha en el eje X y 2 espacios hacia arriba en el eje Y, esto desde la vista superior de la base del prototipo. Para esto se enviarán las siguientes señales de control:

- Motor Eje X
  - CLOCK: 4 pulsos.
  - CW/CCW: 0.
  - ENABLE: 1.
- Motor Eje Y
  - CLOCK: 2 pulsos.
  - CW/CCW: 1.
  - ENABLE: 1.

[27] (X, Y, Z) en coordenadas cartesianas.

La Figura 3.37 muestra la vista superior del ejemplo mencionado anteriormente, mientras que en la Figura 3.38 se observan las señales de control generadas.

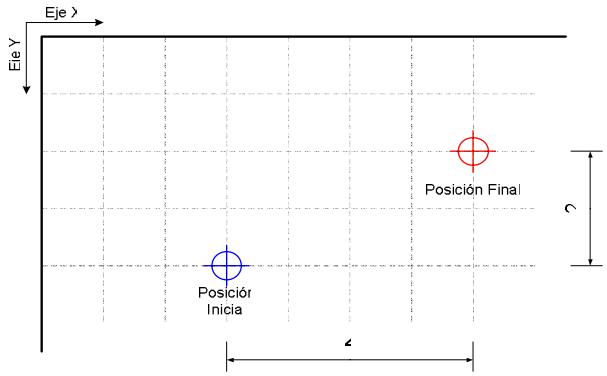


Figura 3.37 Ejemplo Instrucción MOVER (Vista Superior)

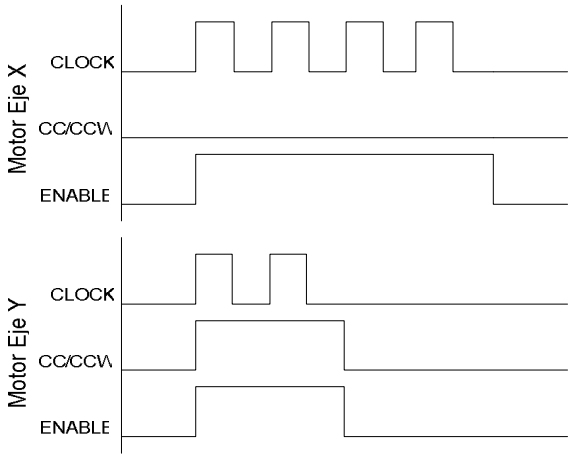


Figura 3.38 Ejemplo Instrucción MOVER (Señales de Control)

Línea

Esta instrucción es bastante parecida a la MOVER ya que se trata de un movimiento lineal desde un punto a otro. La diferencia radica en que los motores se deben mover simultáneamente para dibujar una línea recta, es decir, deben encenderse y apagarse en los mismos tiempos. Esto para el caso de una diagonal, cuando se trata de una línea en un solo eje no existe problema alguno.

Para realizar correctamente el movimiento de los motores tal que resulte una diagonal, un motor deberá andar más rápido que el otro o viceversa. En el ejemplo mostrado en la función

MOVER el motor del Eje Y debería andar a la mitad de la frecuencia que el motor del Eje X, con esto el punto temporal de inicio y fin coinciden perfectamente.

Existe una relación entre la cantidad de pasos y la frecuencia de cada reloj con el que cada motor se moverá. Esta relación resulta del análisis del tiempo que se demora en moverse la cantidad de pasos. Si "Pasos X" es la cantidad de pasos a mover y "Clock X" la frecuencia del reloj de paso del motor del Eje X, se tiene la Ecuación 3.4.

Ecuación 3.4

$$\frac{\text{Pasos X}}{\text{Clock X}} = \text{Tiempo Ejecutado}$$

Entonces, como los motores de los Ejes X e Y cumplen la y como el tiempo de ejecución debe ser el mismo para ambos motores (condición para la dibujar una diagonal), las frecuencias de los relojes de éstos deberán cumplir la Ecuación 3.5.

Ecuación 3.5

$$\frac{\text{Pasos X}}{\text{Clock X}} = \frac{\text{Pasos Y}}{\text{Clock Y}}$$

La Figura 3.39 muestra la vista superior del ejemplo de esta instrucción, mientras que en la Figura 3.40 se observan las señales de control generadas.

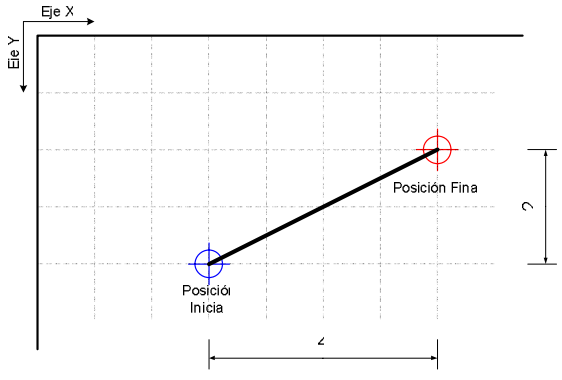


Figura 3.39 Ejemplo Instrucción LINEA (Vista Superior)

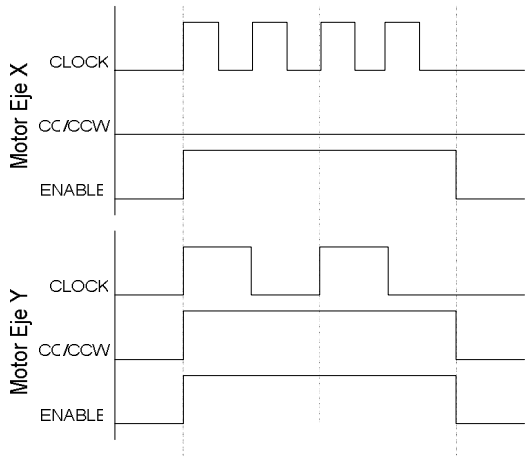


Figura 3.40 Ejemplo Instrucción LINEA (Señales de Control)

Círculo

Para la instrucción CIRCULO primero hay que moverse al punto central utilizando la instrucción MOVER. Luego, se puede aproximar un círculo mediante líneas rectas contiguas. Mientras mayor sea el número de rectas que dibujan el círculo mayor será la resolución de este. Las Figura 3.41 y Figura 3.42 muestran de manera gráfica la resolución de esta aproximación aumentando el número de rectas, donde N es el número de rectas utilizadas en la aproximación.

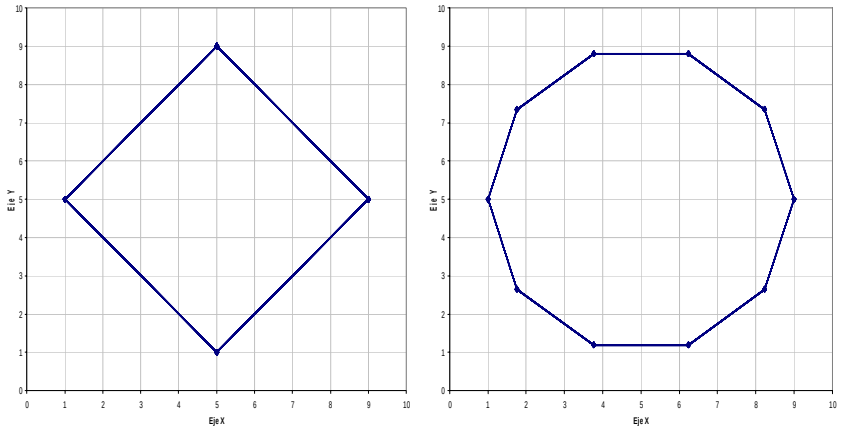


Figura 3.41 Aproximación de un Círculo mediante Rectas con N=4 y N=10

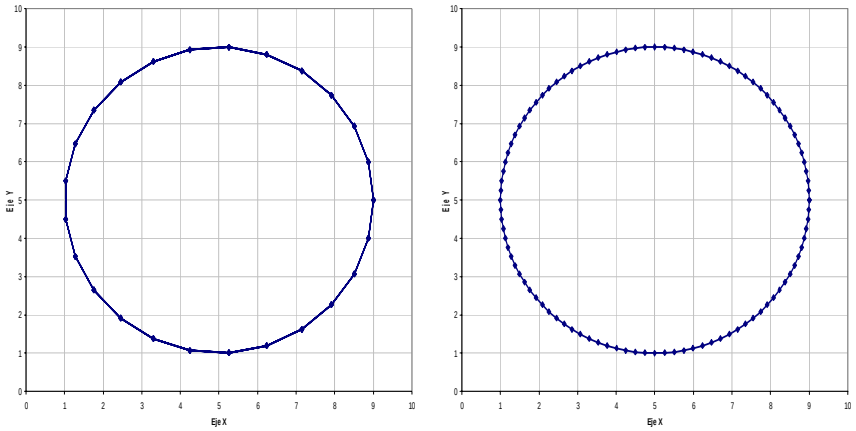


Figura 3.42 Aproximación de un Círculo mediante Rectas con N=25 y N=100

Dado que la instrucción LINEA se encuentra disponible, se utilizará esta misma para dibujar cada recta. De esta manera se pueden realizar funciones más complejas como la de dibujar una curva o un círculo. En este caso se usará una cantidad de 100 rectas para dibujar un círculo. Esto es en el caso de círculos de un mayor tamaño.

Diagrama de Flujo

El programa comienza moviendo el cabezal del prototipo de la fresadora hacia el cero absoluto de la máquina hasta que cada pulsador envíe la señal de que se encuentra en su origen. Luego de esto, el programa entra en un bucle preguntando si es que existe alguna instrucción recibida. Si es que la hay es necesario decodificarla y ejecutarla. Por último, una vez completada la instrucción se envía un ACK hacia el equipo *host* para confirmar la ejecución exitosa de la instrucción y vuelve al bucle de inicial para recibir otra instrucción.

La Figura 3.43 es el diagrama de flujo de la operación del programa a cargar en el dispositivo DSP. Existen algunas variables que son necesarias: variable de la posición actual de el cabezal (pos_x, pos_y, pos_z); variable de pasos a mover en cada motor (paso_x, paso_y, paso_z); variable de la división del reloj de paso (div_x, div_y, div_z) para realizar el comando LINEA; y, una variable que establece si la máquina fresadora está trabajando o no (ocupado).

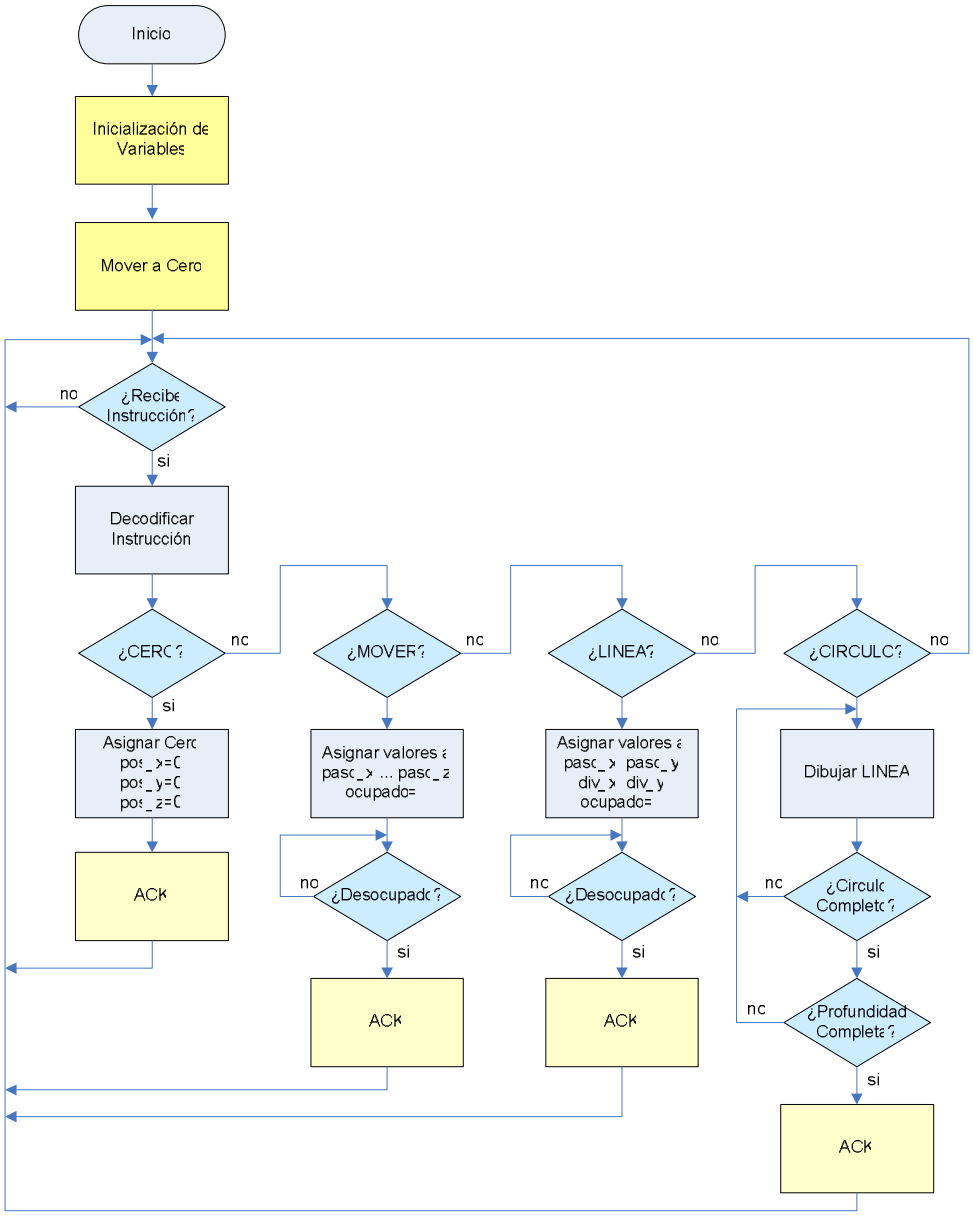


Figura 3.43 Diagrama de Flujo del Programa

3.4.3. Implementación

El código a desarrollar se compila mediante las herramientas *Code Composer Studio* mencionadas anteriormente. Con esto se genera un archivo en formato COFF[28] el cual es

[28] *Common Object File Format.*

cargado en la memoria *Flash* interna del DSP mediante la utilidad *SDFlash* [28] de *Spectrum Digital* vía comunicación serial [29][30][31].

En la Figura 3.44 se observa la aplicación *SDFlash* para cargar el código compilado en el DSP. Como se puede ver primero la memoria *Flash* es borrada para luego ser programada por la utilidad. Finalmente el código es verificado con el propósito de asegurar que el DSP ha sido correctamente programado.

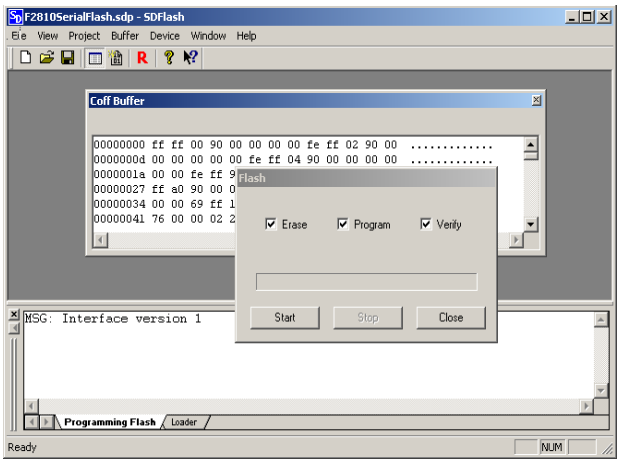


Figura 3.44 Utilidad SDFlash

3.5. Programa DXF

Este programa es la interfaz gráfica utilizada en el computador el cual provee la información obtenida desde un archivo DXF hacia la plataforma de desarrollo DSP.

3.5.1. Requerimientos

Debido a que este programa es un complemento del *test bench* de la plataforma, los requerimientos para este serán mínimos. Estos son:

- Leer y decodificar instrucciones y parámetros
  - Instrucciones de tipo LINE y CIRCLE.

- Almacenar los datos en el programa.

- Comunicación Serial con Plataforma.
  - Utilizar puerto COM con la plataforma (USB o RS232).
- Envío de Instrucciones
  - Instrucciones de tipo LINE y CIRCLE compatibles con el Código DSP.

3.5.2. Diseño

Para realizar el programa se utilizará la utilidad *Visual Basic 6.0* de *Microsoft* debido al conocimiento y uso previo de esta herramienta en proyectos anteriores. Además, esta utilidad permite realizar aplicaciones en ambiente *Windows* de manera simple.

Primero el programa debe abrir un archivo en formato DXF, lo que no presenta mayor problema puesto que este formato es texto plano (sin formato) y puede ser leído con cualquier editor de texto como "Block de Notas" de *Windows*. Luego, se busca alguna de las instrucciones a decodificar (LINE y CIRCLE). Entonces, cuando se ha detectado el encabezado de una figura se deben leer los parámetros de la instrucción. La Tabla 3.14 detalla el formato en que se encuentran estos datos dentro del archivo DXF [9].

Tabla 3.14 Instrucciones DXF

Instrucción	Parámetro	Descripción
LINE	10	Punto X inicial.
	20	Punto Y inicial.
	30	Punto Z inicial.
	11	Punto X final.
	21	Punto Y final.
	31	Punto Z final.
CIRCLE	10	Punto X del centro.
	20	Punto Y del centro.
	30	Punto Z del centro.
	40	Radio del círculo.

La Figura 3.45 muestra un ejemplo del formato de un archivo DXF. Como se ve este es leído con una herramienta tan básica como el "Block de Notas". En amarillo están resaltadas las líneas importantes del archivo, es decir, las que se deben extraer.

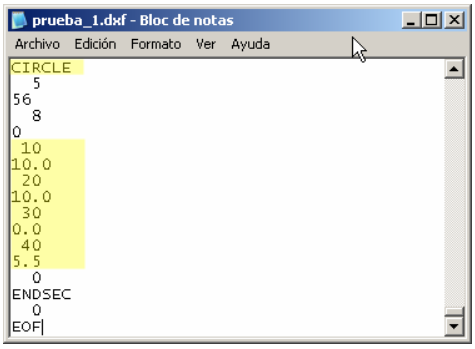


Figura 3.45 Formato Archivo DXF

Una vez que se ha leído el archivo completamente y se ha obtenido la información, esta es guardada en el programa. Luego, se puede enviar cada instrucción con sus parámetros a la plataforma DSP mediante el puerto serial.

3.5.3. Implementación

El programa es compilado usando la misma utilidad *Visual Basic* generando un archivo ejecutable. Esta aplicación recibe el nombre de "DXF a CNC" puesto que es precisamente la función que debe realizar. En la Figura 3.46 se puede observar el programa realizado.

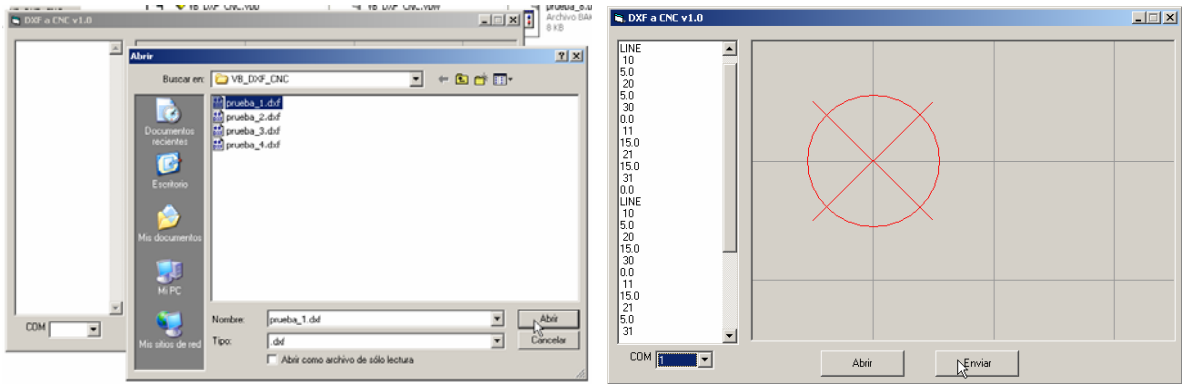


Figura 3.46 Aplicación DXF a CNC

Capítulo 4

Pruebas y Resultados

Para poder verificar el correcto funcionamiento de cada componente se realizan una serie de pruebas que varían de acuerdo a la naturaleza de la unidad bajo prueba. Mediante estas pruebas se valida el diseño realizado en conjunto con su implementación.

4.1. Plataforma de Desarrollo

Las pruebas realizadas a la plataforma de desarrollo son completamente de tipo eléctricas. De estas se destacan medidas de voltaje y corriente utilizadas por la plataforma en operación normal así como pruebas de la interfaz de comunicación de la misma. Estas pruebas fueron realizadas en el "Laboratorio de Electrónica" del "Edificio de Electrotecnologías" de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile, contando con equipos e instrumentación de alto nivel.

4.1.1. Alimentación

Para realizar mediciones de voltaje se utiliza un osciloscopio digital modelo 54622A de la marca *Agilent Technologies*. Para las medidas de corriente continua se usa un multímetro digital modelo DM-332 de la marca *EZ-Digital*. Como fuente de poder externa se utiliza el modelo GP-4303TP de la marca *EZ-Digital*.

Se utilizan los *testpoints* o puntos de prueba de la plataforma para realizar pruebas de voltaje de salida del regulador TPS70151. Debido a que las salidas de este regulador no se pueden intervenir para realizar mediciones de corriente, sólo se medirá la corriente total consumida por la plataforma. La Tabla 4.1 detalla los valores obtenidos para las pruebas de voltaje y corriente de la plataforma de desarrollo.

Tabla 4.1 Mediciones Plataforma de Desarrollo

Parámetro	Descripción	Valor	Unidad
V_{IN}	Voltaje Entrada	9	V
$V_{+3.3}$	Salida +3.3V	3.30 ± 0.125	V
$V_{+1.8}$	Salida +1.8V	1.82 ± 0.095	V
I_0	Corriente Consumida	172	mA
$t_{rise\ +3.3}$	Tiempo de Subida +3.3V	660	µs
$t_{rise\ +1.8}$	Tiempo de Subida +1.8V	273	µs
t_{RESET}	Duración del Pulso RESET	104	ms

En la Figura 4.1 se pueden observar los oscilogramas de las mediciones de las salidas de voltaje realizadas para régimen permanente, es decir, cuando los niveles de tensión se encuentran estables. Para ello se mide el voltaje promedio *(Avg)* y el voltaje pico a pico *(Pk-Pk)* de la señal continua. El voltaje pico a pico se mide con la finalidad de establecer el *ripple* o rizado de la señal.

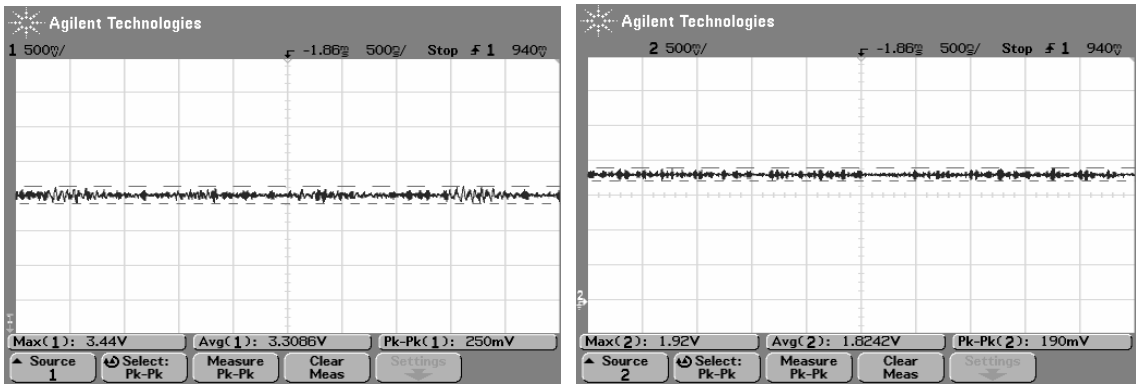


Figura 4.1 Salida de Voltaje de +3.3 y +1.8

En el transiente de estas señales se puede ver la secuencia de encendido de las mismas. En este caso la rampa de +3.3 está primero que la de +1.8 [V]. Esto se puede ver en la Figura 4.2. Además, en la Figura 4.3 se observan las mediciones para establecer el tiempo de subida para cada una de las rampas de voltaje.

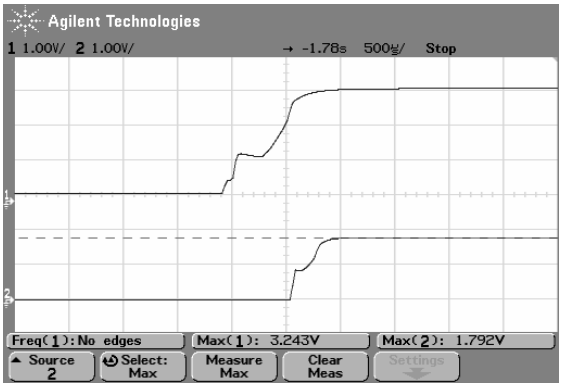


Figura 4.2 Transiente de +3.3 y +1.8 [V]

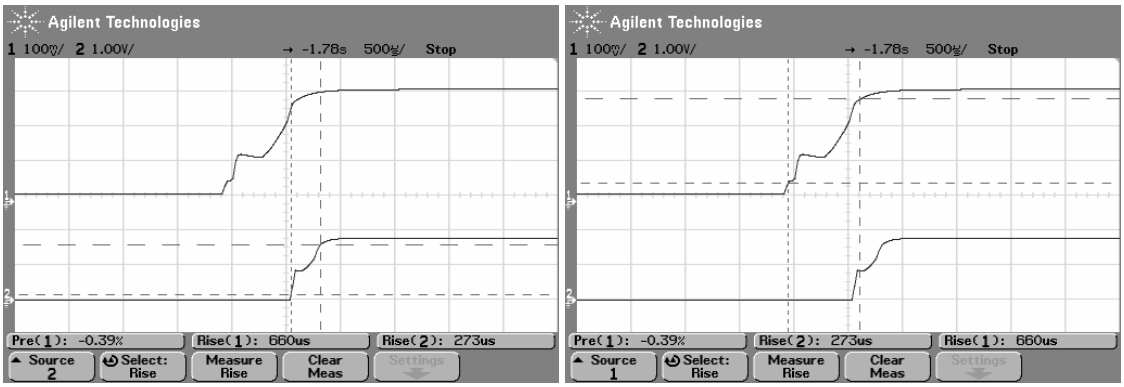


Figura 4.3 Tiempos de Subida para +3.3 y +1.8 [V]

Además de verificar la secuencia de encendido y los voltajes de alimentación del DSP es necesario comprobar el funcionamiento correcto del *Reset* del Administrador de Energía, ya que si este no opera de manera correcta el DSP podría funcionar de manera errática. La función *Reset* es fundamental debido a que le indica al DSP el momento preciso en que este debe iniciarse. Esto ocurre una vez que los voltajes de alimentación se encuentran estables. En la Figura 4.4 se observa la función *Reset* y se verifica que esta opera de manera correcta, ya que se tiene un lapso de tiempo de 104 [ms] desde que se estabiliza la salida de +3.3 [V] hasta que se activa el DSP mediante el *Reset*.

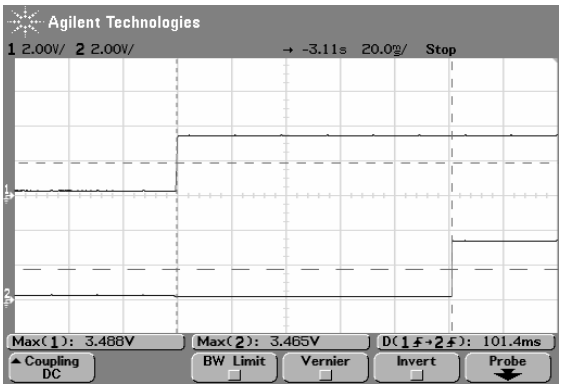


Figura 4.4 Función Reset

Por otro lado se realiza una lectura del cristal resonante el cual posee un valor nominal de 25 MHz. La Figura 4.5 muestra claramente el valor real de este, el cual es muy preciso. En base a esto se verifica el correcto funcionamiento de esta etapa.

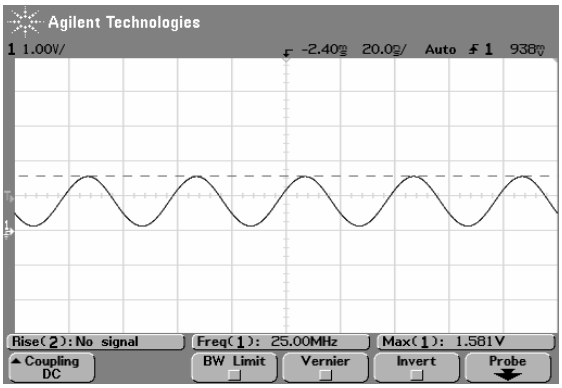


Figura 4.5 Frecuencia de Oscilación del Cristal Resonante

Finalmente, los resultados de las pruebas realizadas al bloque de alimentación de la plataforma verifican y validan el diseño propuesto. Al no presentar ni detectar error alguno en las principales variables eléctricas, se puede decir con seguridad que la plataforma se encuentra en condiciones de operar normalmente sin la necesidad de realizar alguna modificación.

4.1.2. Comunicación

Las pruebas de comunicación se realizan sobre las interfaces USB y RS232 de la plataforma de desarrollo. En estas pruebas se verifica la conectividad, es decir, el enlace creado

entre el computador y la plataforma. Además, se realiza una prueba de error en la comunicación para diferentes tasas de transmisión. Con esto se establece la máxima tasa de transmisión real en base las pruebas realizadas.

La verificación de la conectividad de la interfaz serial RS232 se puede realizar cuando existe comunicación entre el computador y el DSP. Para el caso de la interfaz USB la verificación se realiza utilizando la tecnología *Plug and Play*[29] del dispositivo USB soportada por *Windows XP*[30]. Al momento de conectar el cable USB entre la plataforma y el computador, este último reconoce el nuevo *hardware* agregado instalando los controladores necesarios junto al puerto de comunicación virtual. Esto se puede ver la Figura 4.6.

Figura 4.6 Instalación del Controlador USB

De esta manera se comprueba que existe conectividad entre la plataforma y el computador mediante la interfaz USB. El único problema aparente es que al instalar el puerto de comunicación virtual (VCP) este tiene una tasa de transmisión máxima de 128 kbps lo que eventualmente podría limitar una transferencia de datos a alta velocidad. Este problema es solucionable diseñando una aplicación que utilice el CP2101 directamente y no el VCP, con esto se puede lograr una transmisión de hasta 920 kbps aproximadamente. La idea del puerto virtual precisamente es evitar modificar aplicaciones ya existentes para una comunicación serial RS232 actuando de manera transparente ante esta. De este modo la aplicación es indiferente a la interfaz ya que la interpreta como un puerto COM más.

Como método para medir el error en la transmisión se utiliza un "autoeco" el cual realiza un envío de datos por el computador y el DSP responde con estos mismos valores. Este método se utiliza en cada prueba variando la tasa de transmisión. En esta ocasión, la cadena de caracteres

[29] Conectar y Usar.

[30] Sistema Operativo utilizado en la prueba.

enviados para realizar la prueba es: "abcde0123456789ABCDE". En esta cadena hay 20 caracteres y cada uno se transmite en su código ASCII[31] en modo binario con un ancho de 8 bits. La Tabla 4.2 muestra los resultados obtenidos mediante esta prueba.

Tabla 4.2 Prueba de Transmisión Serial

Tasa de Transmisión [bps]	**Datos Erróneos**		**Error %**	
	USB	RS232	USB	RS232
19200	0	0	0	0
38400	0	0	0	0
57600	0	0	0	0
115200	0	0	0	0
230400	0	-	0	-
460800	0	-	0	-
921600	20	-	100	-

Con estos datos se establece la tasa de transmisión máxima de la plataforma de desarrollo, la cual está acotada por las especificaciones del DSP. Por esto es que para una tasa mayor a 500 kbps se encuentran errores en la comunicación. Es más, a la tasa máxima permitida de la interfaz USB el porcentaje de error llega al 100% de lo cual se deduce que existe una conexión pero no una comunicación entre la plataforma y el computador.

Por el lado de la interfaz RS232 no se detecta ningún error puesto que la tasa máxima permitida en el lado del computador es de 128 kbps, bastante menor a la tasa de transmisión crítica del DSP. Es por esto que para valores mayores a 128 kbps la prueba no es realizable ya que el mismo computador no permite habilitar el puerto COM a esta velocidad.

[31] *American Standard Code for Information Interchange* o Código Estadounidense Estándar para el Intercambio de Información.

4.1.3. Programación del DSP

Además de medio de comunicación de la plataforma con un computador, la interfaz USB permite la programación del dispositivo DSP a través del modo de programación SCI implementado internamente en el DSP. Esta prueba es realizada con el propósito de verificar la programación del DSP a través de la utilidad *SDFlash* mencionada anteriormente. Esta misma aplicación sirve como herramienta de prueba ya que en el ciclo completo de carga del código existen procesos de: conexión con el DSP; borrado de la memoria *Flash*; programación del código; y, verificación del código cargado. La Figura 4.7 muestra la programación completa del DSP indicando que cada proceso ha concluido con éxito.

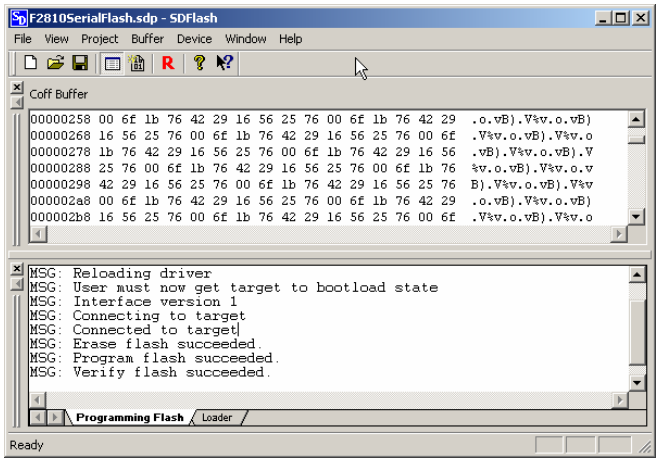


Figura 4.7 Programación DSP

4.1.4. Señales de Control

Esta prueba permite observar las señales de control generadas para el movimiento de un motor Stepper. La variable a verificar en este punto es el voltaje de salida del DSP, el cual debe ser compatible con la placa Controladora de Motor Stepper. En su fase de diseño se estipuló que estas señales son de tipo digital entre 0 y 3.3 [V].

Para esta prueba se realiza un programa de ejemplo el cual envía 5 pulsos en la señal CLOCK y un uno lógico en CW/CCW. Con estas señales el motor debe moverse 5 pasos en la dirección contraria a las agujas del reloj. La Figura 4.8 muestra el oscilograma de las señales

CLOCK y CW/CCW mientras que en la Figura 4.9 se observan las señales CLOCK y ENABLE. Mediante estos resultados se verifica tanto el funcionamiento del programa de ejemplo como los niveles de voltaje requeridos.

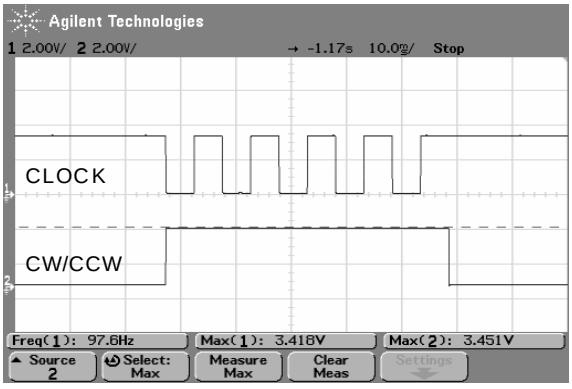


Figura 4.8 Señales de Control CLOCK y CW/CCW

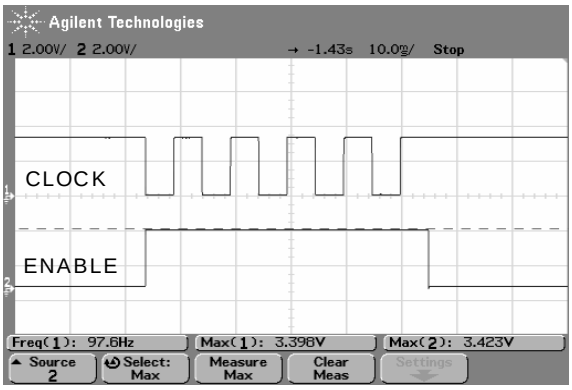


Figura 4.9 Señales de Control CLOCK y ENABLE

4.2. Prototipo Fresadora CNC

Las pruebas requeridas para verificar el funcionamiento del prototipo son de tipo mecánicas y eléctricas. Debido a la simplicidad del diseño obtenido para el prototipo de la máquina, las pruebas a realizar sobre esta son igualmente simples.

4.2.1. Mecánica

Las pruebas sobre esta parte de la máquina son principalmente mediciones de longitud. Los valores obtenidos sirven para calcular los parámetros necesarios para la configuración del conjunto entre la plataforma DSP y la máquina en sí. Los parámetros de cada eje a obtener son: recorrido; y, resolución. Cuando se habla de recorrido se refiere a la longitud máxima que el cabezal puede desplazarse en ese eje. La resolución es el mínimo avance que puede realizar el cabezal en el mismo eje.

Para obtener el recorrido se mide la longitud entre la cual el cabezal se pude mover. Esto es desde un extremo hasta el otro. En cambio, para obtener la resolución es necesario enviar una cierta cantidad de pulsos y medir la distancia recorrida por el cabezal. De esta manera la resolución se calcula en base a la Ecuación 4.1.

Ecuación 4.1

$$\text{resolución} = \frac{\text{distancia recorrida}}{\text{cantidad de pasos}} \left[\text{cm} \diagup \text{paso} \right]$$

La Tabla 4.3 muestra los parámetros obtenidos para cada eje. La resolución para los ejes X e Y es de igual valor puesto que se utiliza el mismo sistema de transmisión. A pesar de que el sistema de guía tiene la misma longitud para los ejes X e Y, este último tiene un menor recorrido ya que hay que tener en cuenta el tamaño del cabezal. Si se realiza el recorrido completo sobre este eje de un extremo a otro, el cabezal toparía antes con el borde de la máquina. Es por esto que se define un recorrido menor para el eje Y de manera de proteger al cabezal de fresado.

Tabla 4.3 Parámetros Prototipo Fresadora CNC

Parámetro	Eje			Unidad
	X	Y	Z	
Recorrido	35	25	10	cm
Resolución	0.02	0.02	0.0005	cm/paso

4.2.2. Hardware

Esta prueba se aplica sobre la placa controladora del motor Stepper con el fin de medir la corriente total consumida por el motor al variar el voltaje aplicado en las bobinas de este. Con estos valores se puede obtener una curva de la corriente total consumida en función del voltaje de polarización para este motor. Esta prueba es realizada con el motor en vacío, es decir, sin carga mecánica aplicada con el fin de evitar datos erróneos ante posibles variaciones de la carga. De esta manera este experimento es repetible y por ende su resultado de mayor confianza.

La Tabla 4.4 muestra los valores obtenidos de la corriente consumida por el motor. El voltaje de polarización para la electrónica lógica es de 5.07 [V] con una corriente consumida de 57.2 [mA].

Tabla 4.4 Corriente Total Consumida por el Motor Stepper

Voltaje [V]	Corriente [A]
5.08	0.57
7.03	0.94
9.05	1.31
12.05	1.93

4.3. Programa DXF

Esta prueba es bastante simple ya que la aplicación "DXF a CNC" carga un archivo en formato DXF y se comprueba que las líneas del archivo con la información de cada figura sean rescatadas. Se abre el archivo "prueba_1.dxf" creado en AutoCAD el cual contiene dos líneas en diagonal y un círculo con centro en la intersección de ambas líneas rectas. El resultado de esta prueba se observa en la Figura 4.10.

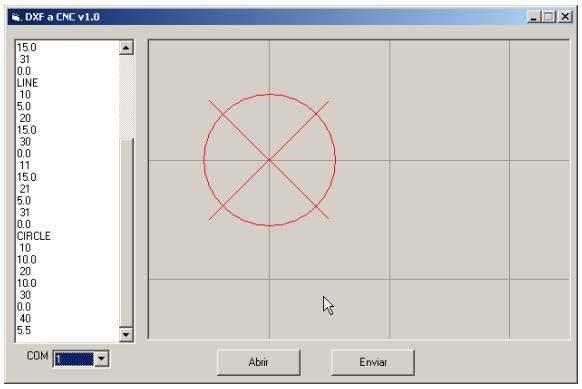


Figura 4.10 Prueba Programa DXF

4.4. Control Prototipo Fresadora CNC

Esta es la prueba de fuego del sistema completo ya que involucra a todos los componentes. Cada uno de estos ha sido sometido a pruebas independientes, se ha verificado su funcionamiento y por consiguiente se ha validado su diseño. Esta prueba es precisamente el *test bench* mencionado en 3.2, donde la electrónica y la mecánica son utilizadas en conjunto en pro del resultado final.

Para esta etapa se ha elaborado una interfaz gráfica de usuario (GUI[32]) en el computador con el propósito de comandar la fresadora a través de las instrucciones insertas en el control del DSP. Esta aplicación permite posicionar el cabezal en un punto deseado y dibujar las figuras disponibles, entre otras funciones. Todas estas instrucciones se encuentran en detalle en 3.4.2. La Figura 4.11 muestra la GUI elaborada para esta prueba la cual ha sido creada con el nombre de "Controlador CNC".

[32] *Graphical User Interface.*

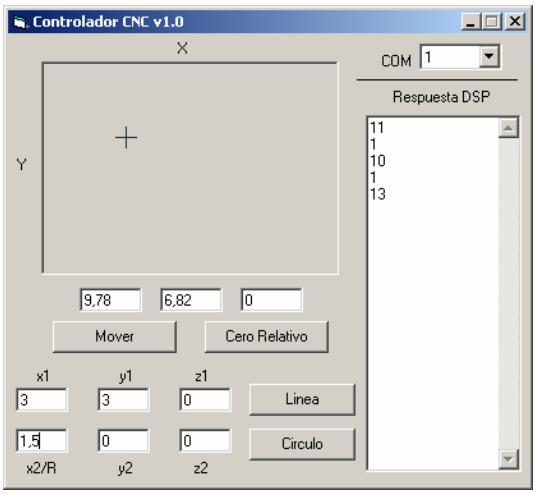


Figura 4.11 Controlador CNC GUI

Primero se realizan pruebas sobre el movimiento (instrucción MOVER) en cada eje de la máquina. Para esto se envía mediante la aplicación el punto hasta donde el cabezal debe desplazarse en un eje. Con esto se compara el valor ingresado con el real el cual se mide con un Pie de Metro, lo que asegura una precisión de una décima de milímetro. La Tabla 4.5 muestra los valores obtenidos y el error calculado para esta instrucción.

Tabla 4.5 Instrucción MOVER

Eje	Valor Ingresado [cm]	Valor Real [cm]	Error %
X	1	0.91	9
	3	2.92	2.7
	5	4.95	1
Y	1	0.95	5
	3	2.98	0.67
	5	5	0
Z	0.5	0.55	10
	1	1.1	10
	3	3.3	10

Se puede decir que los resultados obtenidos son satisfactorios considerando las limitaciones del prototipo, en cuanto a su mecánica. Además, los resultados obtenidos pueden ser

mejorados ajustando los parámetros en la conversión de unidades de longitud, en este caso centímetros, a pasos reales de los motores Stepper.

Se requiere realizar pruebas a la implementación de las funciones para dibujar figuras. En 3.4.2 se definieron las figuras LINEA y CIRCULO las cuales se someten a una prueba de precisión y aproximación del dibujo. Para esto se envía la instrucción para generar un círculo con radio igual a 2 [cm]. Luego se dibujan 4 líneas rectas que forman un cuadrado de 5 [cm] de longitud en cada lado. El cuadrado debe bordear al círculo dibujado previamente. Con esto se verifica que las coordenadas enviadas sean interpretadas y ejecutadas correctamente por el DSP. La Figura 4.12 muestra una secuencia de imágenes obtenidas de la ejecución de este ejemplo, el cual se denomina "Prueba 1".

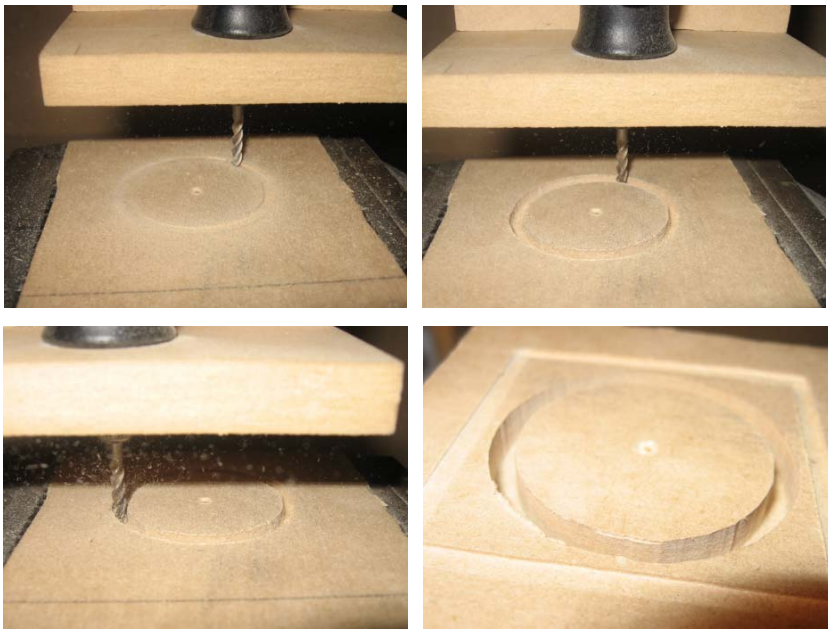


Figura 4.12 Secuencia Construcción Prueba 1

En la Figura 4.13 se observa el resultado obtenido del círculo el cual tiene un diámetro de 4 [cm] de longitud, es decir, un radio de 2 [cm]. La medición se realiza en los puntos centrales de la figura realizada debido a que el punto de referencia para el programa en el DSP es el centro del cabezal fresador. Se realiza la misma comprobación en el cuadrado dibujado la cual arroja un resultado similar al del círculo. Esto se puede ver en la Figura 4.14. Con esto se verifica la precisión de la ejecución de la figura a dibujar y se comprueba que la aproximación mediante

múltiples líneas rectas del círculo es adecuada. Como dato anexo, en esta prueba se aproximó el círculo mediante 100 líneas rectas.

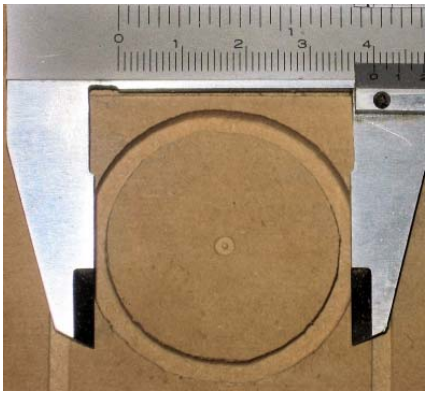


Figura 4.13 Medida Círculo

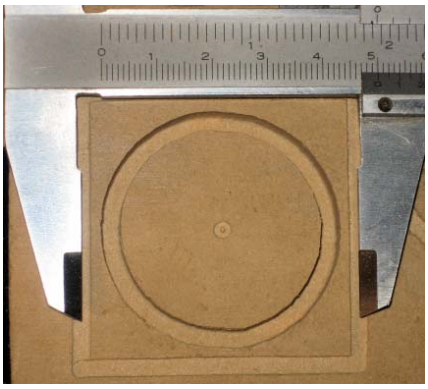


Figura 4.14 Medida Cuadrado

Una segunda prueba es realizada utilizando la aplicación de control y el programa "DXF a CNC". El primero se utiliza para posicionar el cabezal en el área donde se encuentra el material (madera MDF) y establecer un cero relativo. En este cero relativo se fija la profundidad con que se realizará el dibujo. Con esto, la aplicación "DXF a CNC" envía las instrucciones del dibujo cargado, el cual se puede observar en la Figura 4.15. Este ejemplo es denominado como "Prueba 2".

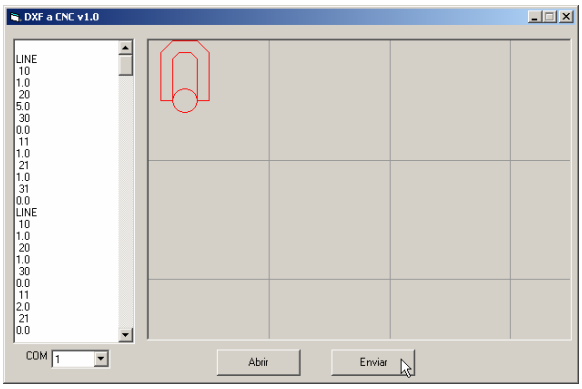


Figura 4.15 Aplicación DXF a CNC para Prueba 2

Al igual que en la Prueba 1, se puede observar la fase de construcción de este ejemplo en la Figura 4.16. El programa primero envía las instrucciones para dibujar líneas y luego los círculos.

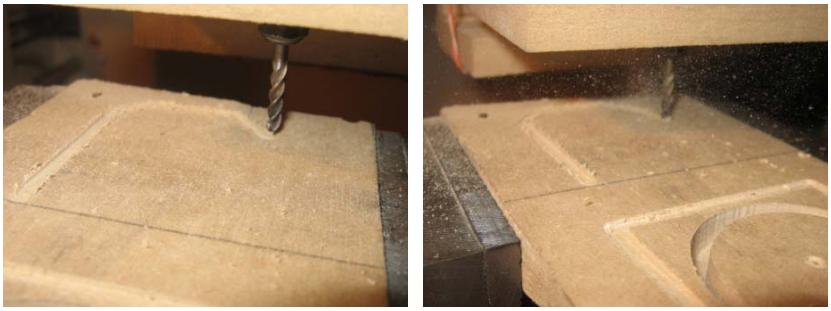


Figura 4.16 Secuencia Construcción Prueba 2

El resultado final de la construcción de este ejemplo puede verse en la Figura 4.17 donde claramente se observa la pérdida de precisión por parte de la máquina. La causa de este problema fue la profundidad con que se fijó el cero relativo ya que la herramienta de fresado se movió realizando el corte del material a esa profundidad. Al no contar con sensores del movimiento real de los motores pueden surgir problemas como este tipo. Estos sensores no fueron contemplados en el diseño del prototipo ya que, por parámetros constructivos, los motores Stepper poseen un alto grado de precisión en sus movimientos y se confió en esta característica.

Figura 4.17 Resultado Prueba 2

El problema de la precisión no es de gran importancia para el propósito con que fue diseñado el prototipo. Con lo sucedido en el ejemplo anterior se realiza una repetición de la prueba ("Prueba 3") pero esta vez se redujo la profundidad con que se fresa el material. La Figura 4.18 muestra el resultado obtenido, el cual es mucho mejor que el anterior.

Figura 4.18 Resultado Prueba 3

Entonces, para operaciones futuras, se debe tener en consideración las limitaciones mecánicas de la máquina para evitar tener errores como los detectados en estas pruebas. En especial con la profundidad con que se desee fresar el material. Para dibujos más profundos se recomienda dibujar el plano de la figura y repetir esto aumentando de a poco la profundidad del dibujo. De hecho, este método fue utilizado para dibujar el círculo en el "Prueba 1".

Capítulo 5
Conclusiones y Trabajos Futuros

5.1. Conclusiones

Como resultado final de este trabajo se obtiene una Plataforma de Desarrollo lista para ser utilizada en alguna aplicación en el área de control. En este documento se plasmó la demostración del funcionamiento de esta plataforma, actuando como controlador de un prototipo de máquina fresadora CNC.

Con respecto al objetivo general, el cual se refiere al diseño e implementación de la plataforma, se puede decir que se ha cumplido completamente. En una primera etapa, se explicitaron los principales requerimientos que debía cumplir la plataforma. Este paso es fundamental para el posterior diseño, el cual es realizado tomando en cuenta tales requerimientos. En esta fase de diseño se dimensionaron las capacidades con que debía contar la plataforma y se entregaron las características técnicas de los componentes utilizados. Finalmente, se obtiene como resultado el *layout* para la construcción de la placa de circuito impreso y una lista de especificaciones de la misma plataforma. Todo este proceso se encuentra detallado en el Capítulo 3.

En el mismo Capítulo 3 se detalla el diseño y la implementación del entorno de prueba de la plataforma. Para esto se construyó un prototipo mecánico de una máquina fresadora y se desarrolló todo el software necesario para estos fines. En este entorno de prueba o *test bench* la

plataforma de desarrollo juega el papel central, ya que es el encargado de realizar el control de la máquina. La idea de este *test bench* fue demostrar y comprobar el funcionamiento real de la plataforma. Gracias a esto se realizaron pruebas y mediciones que verificaron el funcionamiento de la misma.

El Capítulo 4 trató sobre las pruebas realizadas y los resultados obtenidos. Con estos resultados se validó completamente el diseño, ya que se comprobó el funcionamiento de las especificaciones planteadas al término de la etapa de diseño. La mayor parte de las pruebas efectuadas a la plataforma se realizaron con equipos e instrumentos de altísimo nivel, lo cual da más confianza a los resultados obtenidos.

Uno de los puntos al que se le dio mayor importancia en el diseño de la plataforma fue el hecho de utilizar una interfaz de comunicación USB en vez de la clásica RS232, la cual se podría decir que está en extinción en los actuales equipos computacionales personales. Con esto se da un gran paso ya que esta interfaz posee una mayor tasa de transmisión, lo que se traduce en la capacidad de enviar y recibir una mayor cantidad de datos en un menor tiempo.

De esta manera se pueden dar por cumplidos todos los objetivos específicos planteados, ya que se construyó el prototipo de fresadora junto al sistema de sensores para el funcionamiento de este. Asimismo, se desarrolló todo el software necesario para la decodificación y envío de la información de un archivo en formato DXF.

Como conclusión final de este trabajo se puede decir que se cuenta con un dispositivo controlador de altas prestaciones, diseñado y construido en Chile. Este mismo es capaz de ser utilizado en otras aplicaciones y, como su nombre lo dice, esta plataforma de desarrollo es una base para futuros proyectos en el área de la electrónica.

5.2. Trabajos Futuros

La plataforma de desarrollo construida permite ser utilizada en diversos proyectos en el área de control de máquinas u otros componentes eléctricos o electromecánicos. Un ejemplo claro de esto es en aplicaciones de inversores o cicloconvertidores de corriente alterna para el

control de velocidad de motores de inducción trifásicos. Esto último debido a que cuenta con salidas de 12 señales PWM y 8 entradas para el conversor de señales análogas a digital incorporado.

Por otro lado, es de gran interés comprobar el rendimiento de la plataforma, más específicamente del DSP utilizado, en áreas donde no fue diseñado. Un trabajo a futuro sería aprovechar la plataforma en aplicaciones de procesamiento de señales de audio para observar la respuesta del DSP. Además, debido a que el tratamiento de señales digitales de audio es más complejo, sería de interés integrar un Sistema Operativo Embebido (DSP/BIOS para DSPs *Texas Instruments*) para la administración de los recursos dentro del DSP. Este sistema operativo es fundamental cuando se trata del procesamiento de señales en tiempo real.

Finalmente, se puede utilizar la plataforma desarrollada para la construcción de una máquina CNC industrial. Obviamente el diseño de esta máquina debe ser apoyado por personas ligadas al área de la mecánica. Sería interesante llegar a un producto final de tipo industrial utilizando la tecnología desarrollada en este trabajo.

Referencias

[1] Alarcón, C., "Diseño e Implementación de Interfaz de Programación para Plataforma orientada al control de Vehículos Autónomos", Memoria para optar al Título de Ingeniero Civil Electricista, Universidad de Chile, 2006.

[2] Ramírez, J., "Diseño y Construcción de Plataformas Reprogramables de Comunicación de Sensores vía protocolo CAN, aplicado al control de Vehículos Autónomos", Memoria para optar al Título de Ingeniero Civil Electricista, Universidad de Chile, 2006.

[3] Bateman, A. y Paterson-Stephens, I., "The DSP Handbook", cap. 1-4, Prentice Hall, 2002.

[4] Horowitz, P. y Hill, W., "The Art of Electronics. 2nd Edition", cap. 10-11, Cambridge University Press, 1989.

[5] Nanfara, F., Uccello, T., y Murphy, D., "The CNC Workshop Version 2.0", cap. 1, Schroff Development Corp., 2002.

[6] Williams, G., "CNC Robotics: Build your own Workshop Bot", cap. 1-2, McGraw-Hill, 2003.

[7] ST Microelectronics, "AN470 Application Note: The L297 Stepper Motor Controller", Nov. 2003.

[8] TodoRobot.com, "Tutorial sobre Motores Paso a Paso", <http://www.todorobot.com.ar/informacion/tutorial%20stepper/stepper-tutorial.htm>.

[9] Autodesk, "AutoCAD 2002: DXF Reference Guide", 2001.

[10] Texas Instruments, "TMS320F2810, TMS320F2811, TMS320F2812, TMS320C2810, TMS320C2811, TMS320C2812 Digital Signal Processors Data Manual (Rev. N)" (SPRS174N) <http://www.ti.com/lit/gpn/tms320f2810>.

[11] Texas Instruments, "TMS320F2810, TMS320F2811, TMS320F2812, TMS320C2810, TMS320C2811, TMS320C2812 DSP Silicon Errata" (SPRZ193J) < http://www.ti.com/litv/pdf/sprz193j>.

[12] Texas Instruments, "TPS701XX: Dual-Output Low-Dropout Voltage Regulators with Power-Up Sequencing For Split-Voltge DSP Systems (Rev. D)" (SLVS222D) < http://www.ti.com/lit/gpn/tps70151>.

[13] Fairchild Semiconductor, "LM78XX/LM78XXA 3-Terminal 1A Positive Voltage Regulator" < http://www.fairchildsemi.com/ds/LM/LM7805.pdf>.

[14] Texas Instruments, "High-Speed DSP Systems Design Reference Guide" (SPRU889) Sección 4 <http://focus.ti.com/lit/ug/spru889/spru889.pdf>.

[15] Citizen America, "CS20: AT-CUT Crystal Unit Datasheet" <http://www.citizencrystal.com/images/pdf/ms-cs20.pdf>.

[16] Texas Instruments, "TMS320X28XX, 28XXX DSP Serial Communication Interface (SCI) Reference Guide (SPRU051B)" <http://www.ti.com/litv/pdf/spru051b>.

[17] Silicon Laboratories, "CP2101: Single-Chip USB to UART Bridge Datasheet" <https://www.mysilabs.com/public/documents/tpub_doc/dsheet/Microcontrollers/Interface/en/CP2101.pdf>.

[18] Silicon Laboratories, "CP210X USB to UART Bridge VCP Drivers" <https://www.mysilabs.com/tgwWebApp/public/web_content/products/Microcontrollers/USB/en/mcu_vcp.htm>.

[19] Texas Instruments, "MAX3221: 3-V to 5.5-V Single-Channel RS-232 Line Driver/Receiver with ±15-kV ESD Protection Datasheet (Rev. M)" (SLLS348M) <http://www.ti.com/lit/gpn/max3221>.

[20] STMicroelectronics, "L297: Stepper Motor Controller Datasheet" <http://www.st.com/stonline/products/literature/ds/1334/l297.pdf>

[21] STMicroelectronics, "L298: Dual Full-Bridge Driver Datasheet" <http://www.st.com/stonline/products/literature/ds/1773/l298.pdf>.

[22] Texas Instruments, "3.3 V DSP Digital Motor Control" (SPRA550) <http://www.ti.com/litv/pdf/spra550>.

[23] Texas Instruments, "Code Composer Studio IDE" <http://focus.ti.com/docs/toolsw/folders/print/ccstudio.html>.

[24] Texas Instruments, "C281X C/C++ Header Files and Peripheral Examples" (SPRC097) <http://www.ti.com/litv/zip/sprc097>.

[25] Texas Instruments, "TMS320X281X: DSP System Control and Interrupts Reference Guide" (SPRU078D) <http://www.ti.com/litv/pdf/spru078d>.

[26] Texas Instruments, "TMS320X28XX, 28XXX: DSP Peripheral Reference Guide (Rev. D)" (SPRU566D) <http://www.ti.com/litv/pdf/spru566d>

[27] Texas Instruments, "TMS320F281X: DSP Event Manager (EV) Reference Guide (Rev. E)" (SPRU065E) <http://www.ti.com/litv/pdf/spru065e>

[28] Spectrum Digital, "SDFlash Windows GUI Utility" <http://emulators.spectrumdigital.com/utilities/sdflash>.

[29] Spectrum Digital, "TMS320F28XX: SDFlash Serial RS232 Flash Programming Algos V3.1" <http://emulators.spectrumdigital.com/utilities/sdflash/c2000/sdf28xx_v3_1_serial.zip>.

[30] Spectrum Digital, "TMS320F28XX: SDFlash Serial RS232 Flash Programming Reference Guide" <http://emulators.spectrumdigital.com/utilities/sdflash/c2000/SDFlash_Serial_RefGuide_v3_1.pdf>.

[31] Texas Instruments, "TMS320F281X: Flash Programming API Revision Change" (SPRAAB6) <http://www.ti.com/litv/pdf/spraab6>

Anexo A

Esquemáticos

El resultado de la etapa de diseño de la Plataforma de Desarrollo es el circuito esquemático de cada parte de esta. Se detallan los circuitos integrados utilizados, componentes, conectores y conexiones realizadas.

Además, se entregan los esquemáticos de las placas para los motores Stepper y Sensores.

A.1 Plataforma de Desarrollo: Circuito Principal DSP CORE y conexiones I/O.

A.2 Plataforma de Desarrollo: Módulos Periféricos I/O.

A.3 Plataforma de Desarrollo: Conectores.

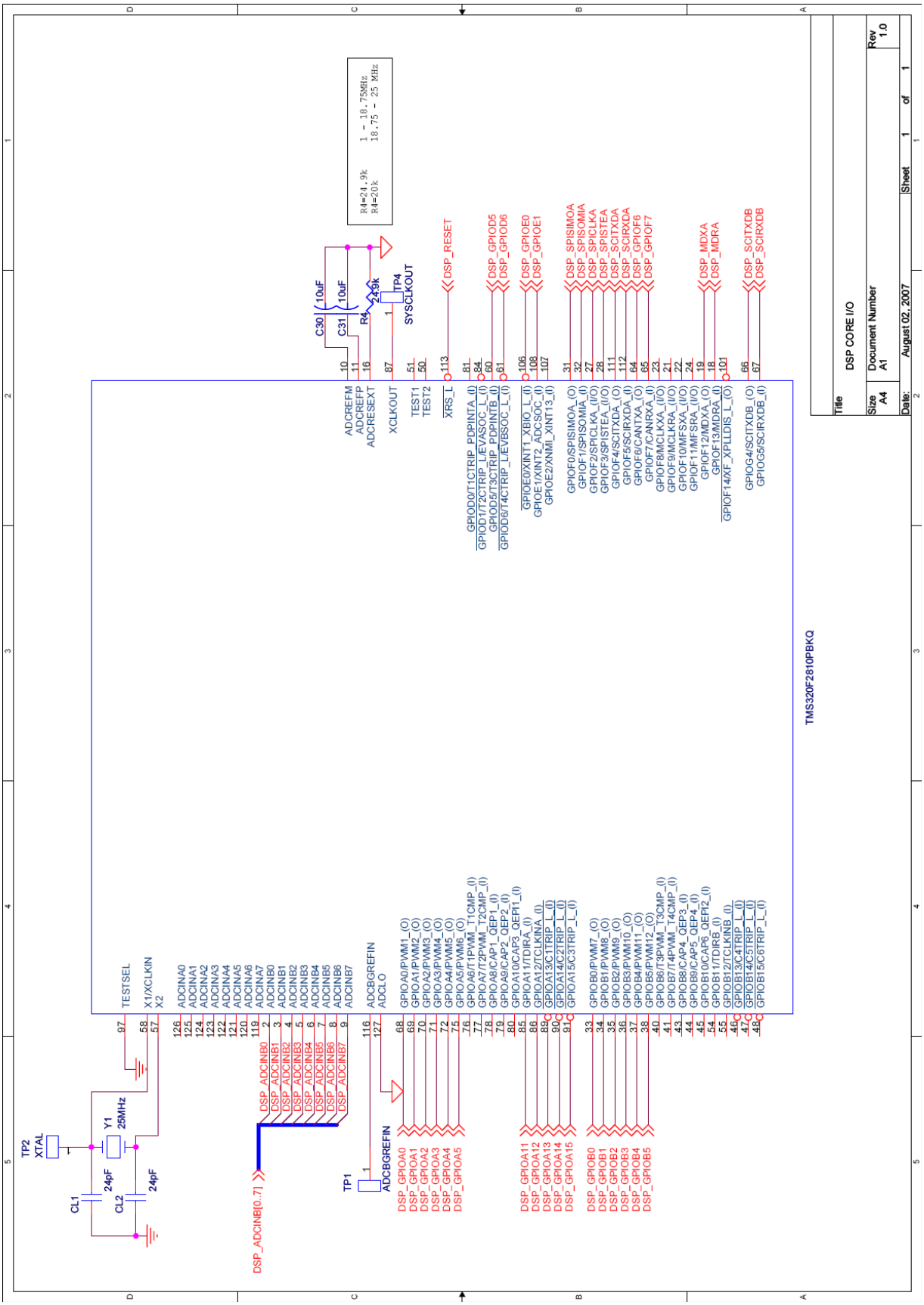
A.4 Plataforma de Desarrollo: Circuito de Alimentación.

A.5 Plataforma de Desarrollo: Módulo JTAG.

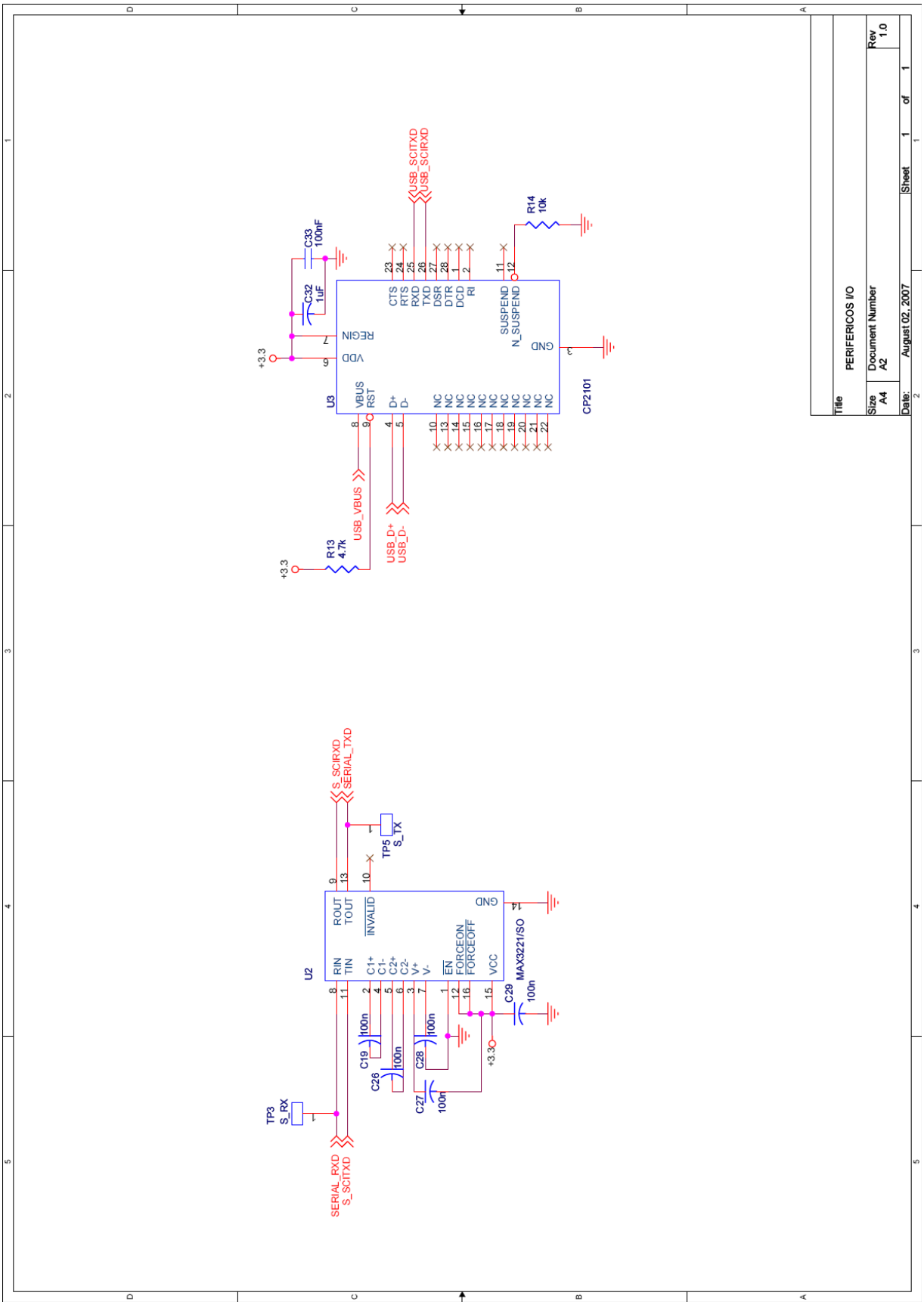
A.6 Controladora de Motor Stepper.

A.7 Sensores.

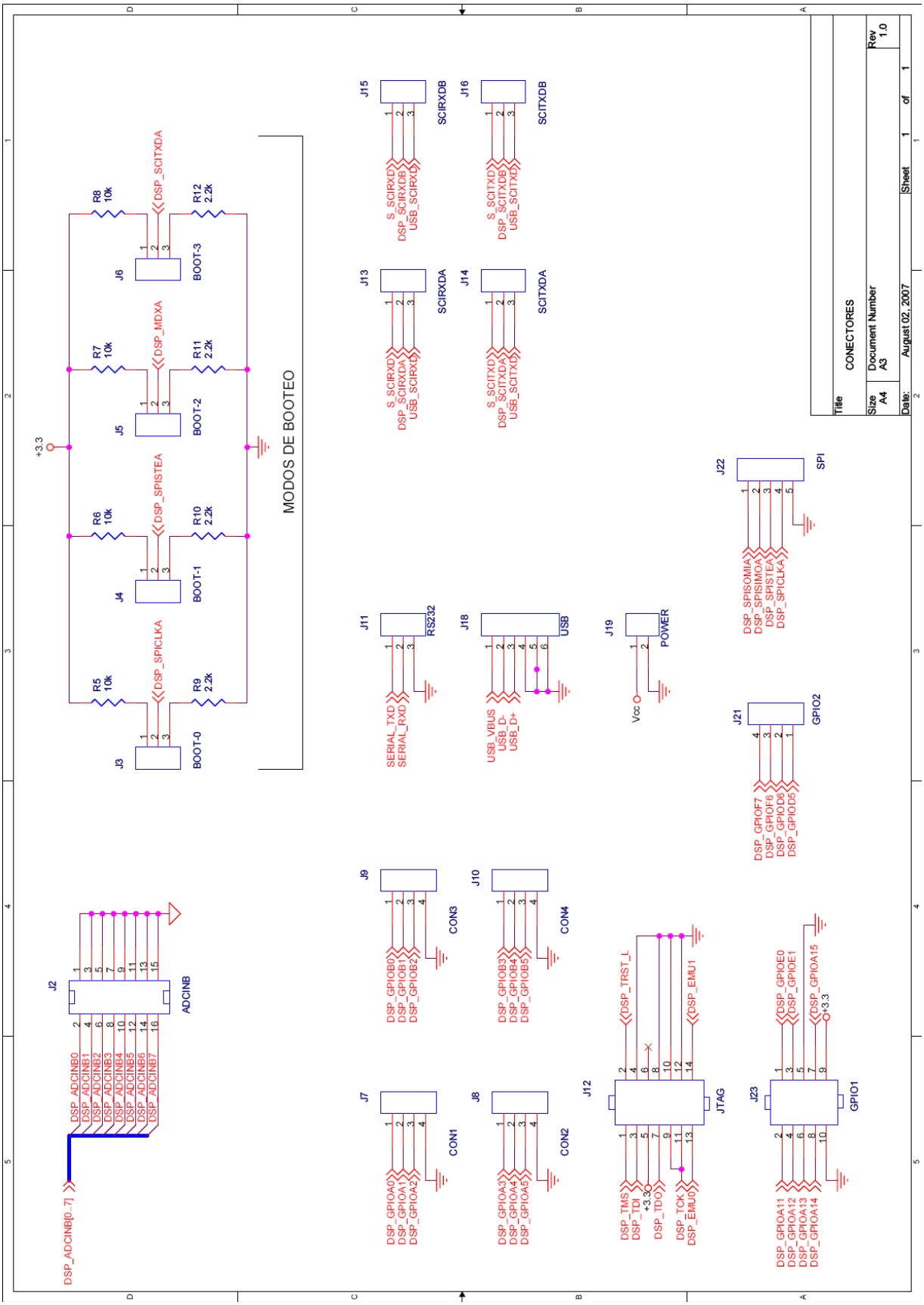
A.1. Plataforma de Desarrollo: Circuito Principal DSP CORE y conexiones I/O.



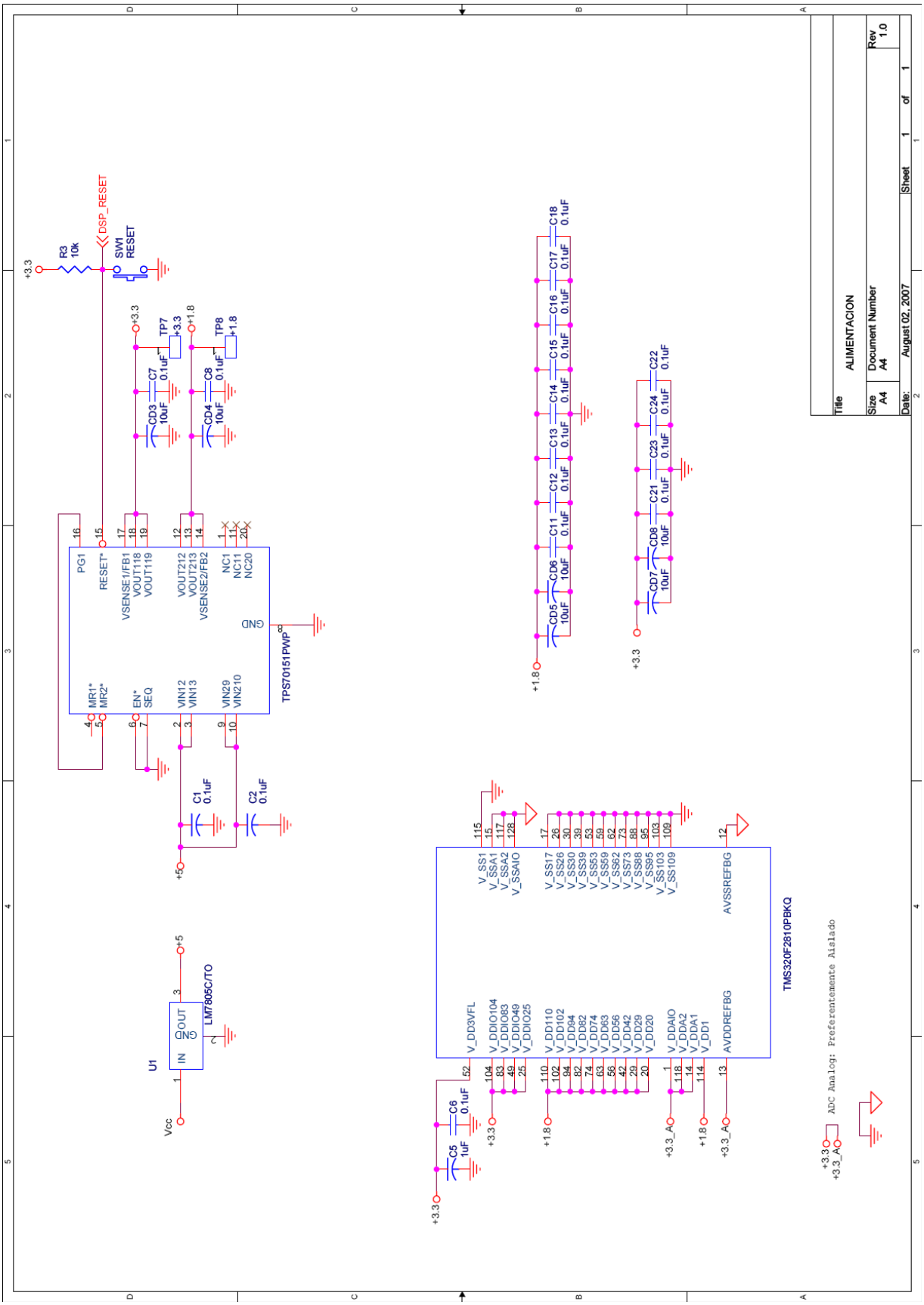
A.2. Plataforma de Desarrollo: Módulos Periféricos I/O.



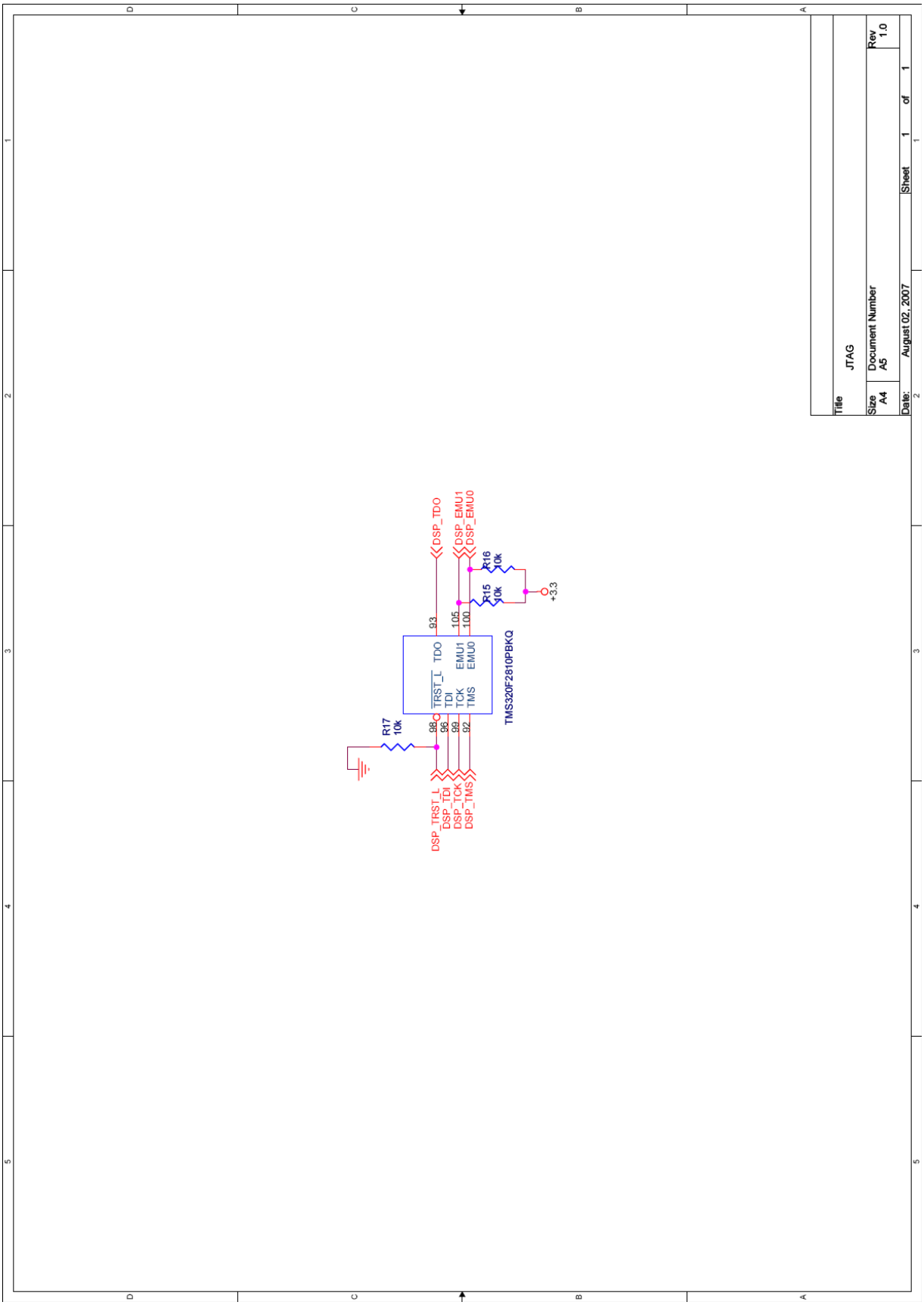
A.3. Plataforma de Desarrollo: Conectores.



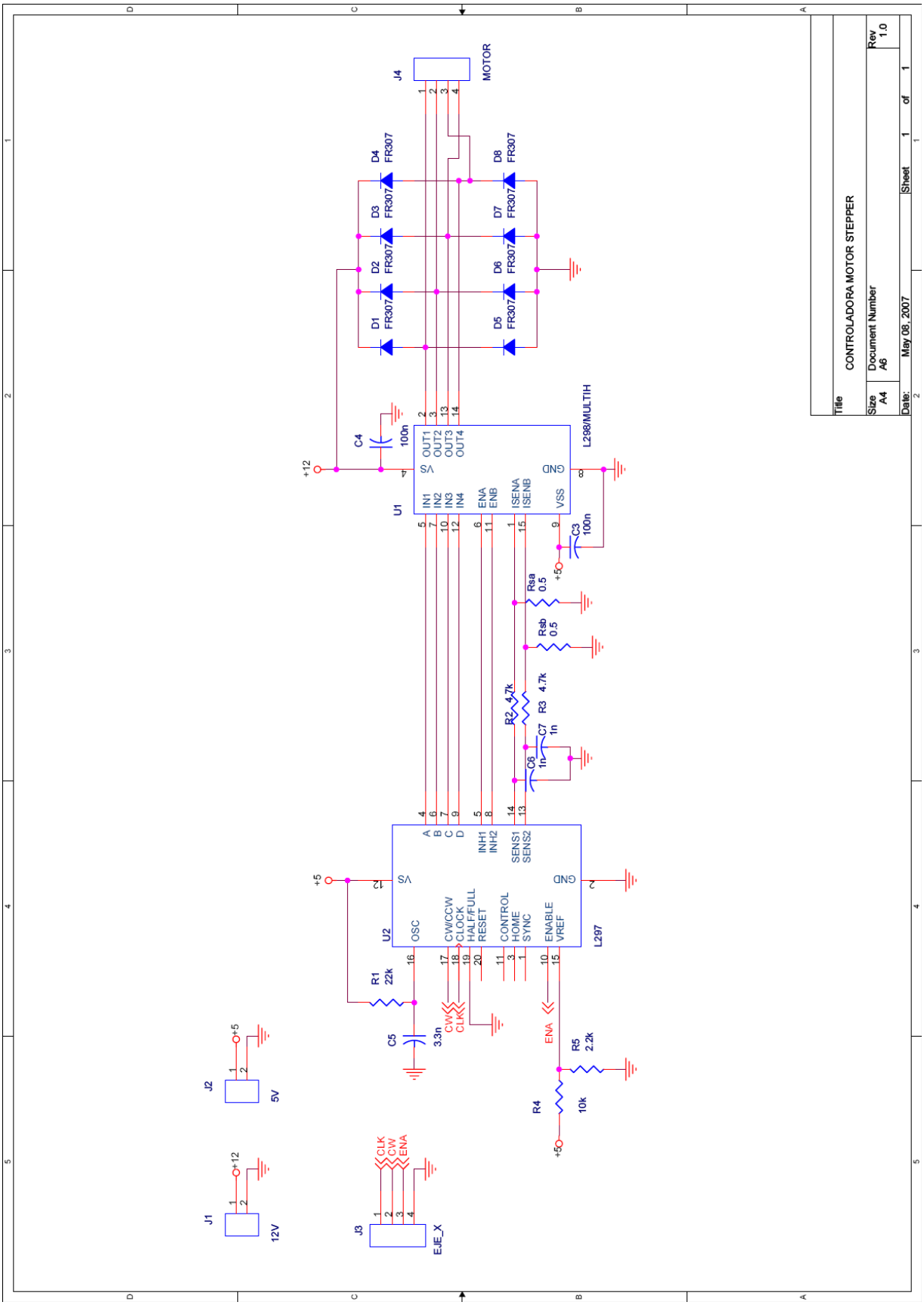
A.4. Plataforma de Desarrollo: Circuito de Alimentación.



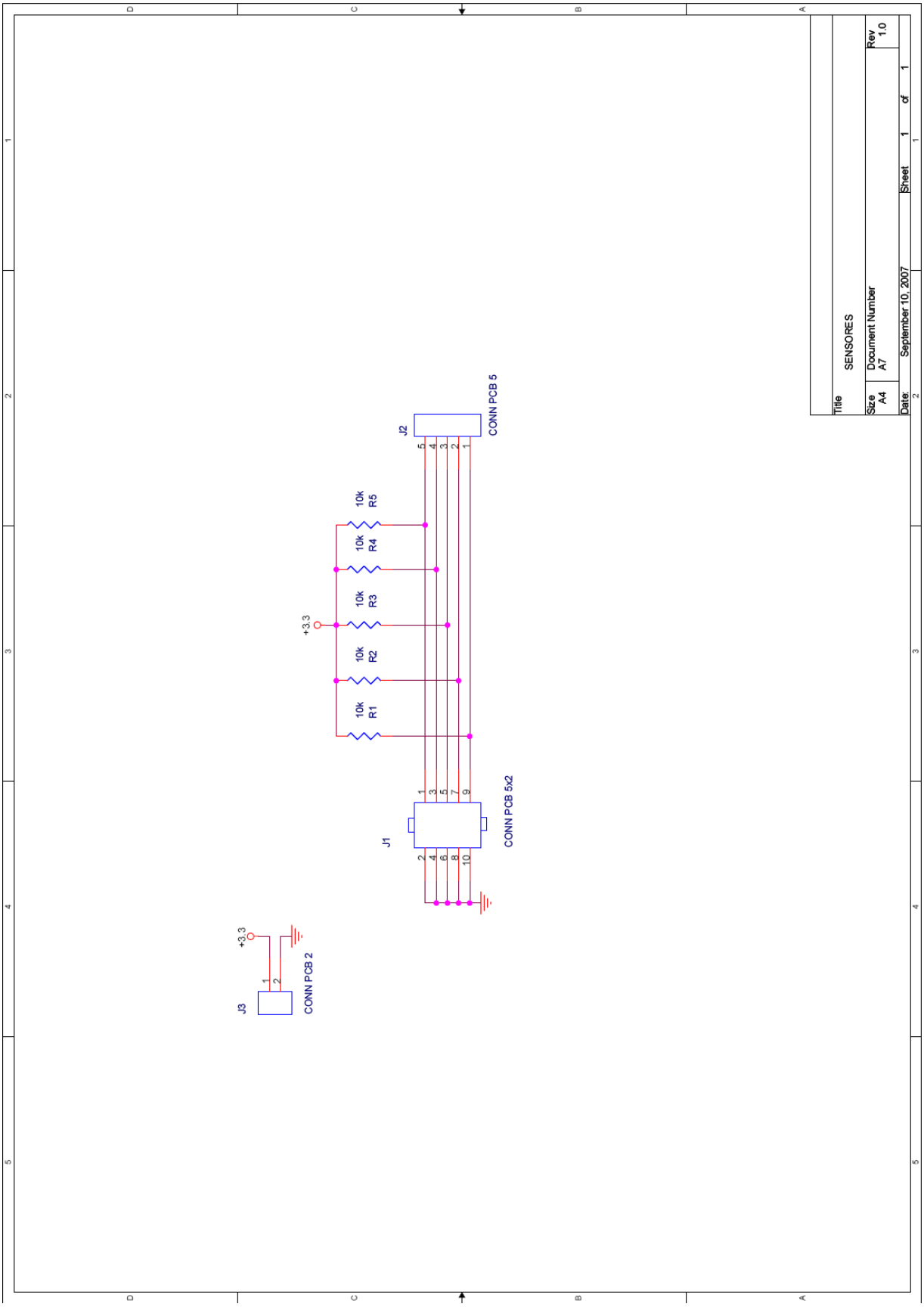
A.5. Plataforma de Desarrollo: Módulo JTAG.



A.6. Controladora de Motor Stepper.



A.7. Sensores.


SENSORES
Size A4
Document Number A7
Rev 1.0
Date: September 10, 2007
Sheet 1 of 1
J2
CONN PCB 5
10k R5
10k R4
10k R3
10k R2
10k R1
+3.3
J1
CONN PCB 5x2
J3
CONN PCB 2
+3.3

Anexo B

Planos Prototipo Máquina Fresadora

Se adjuntan planos realizados para la construcción de piezas mecánicas. Todas las medidas entregadas están en milímetros. Los planos no están a escala real. Las líneas en diagonal representan perforaciones en el material.

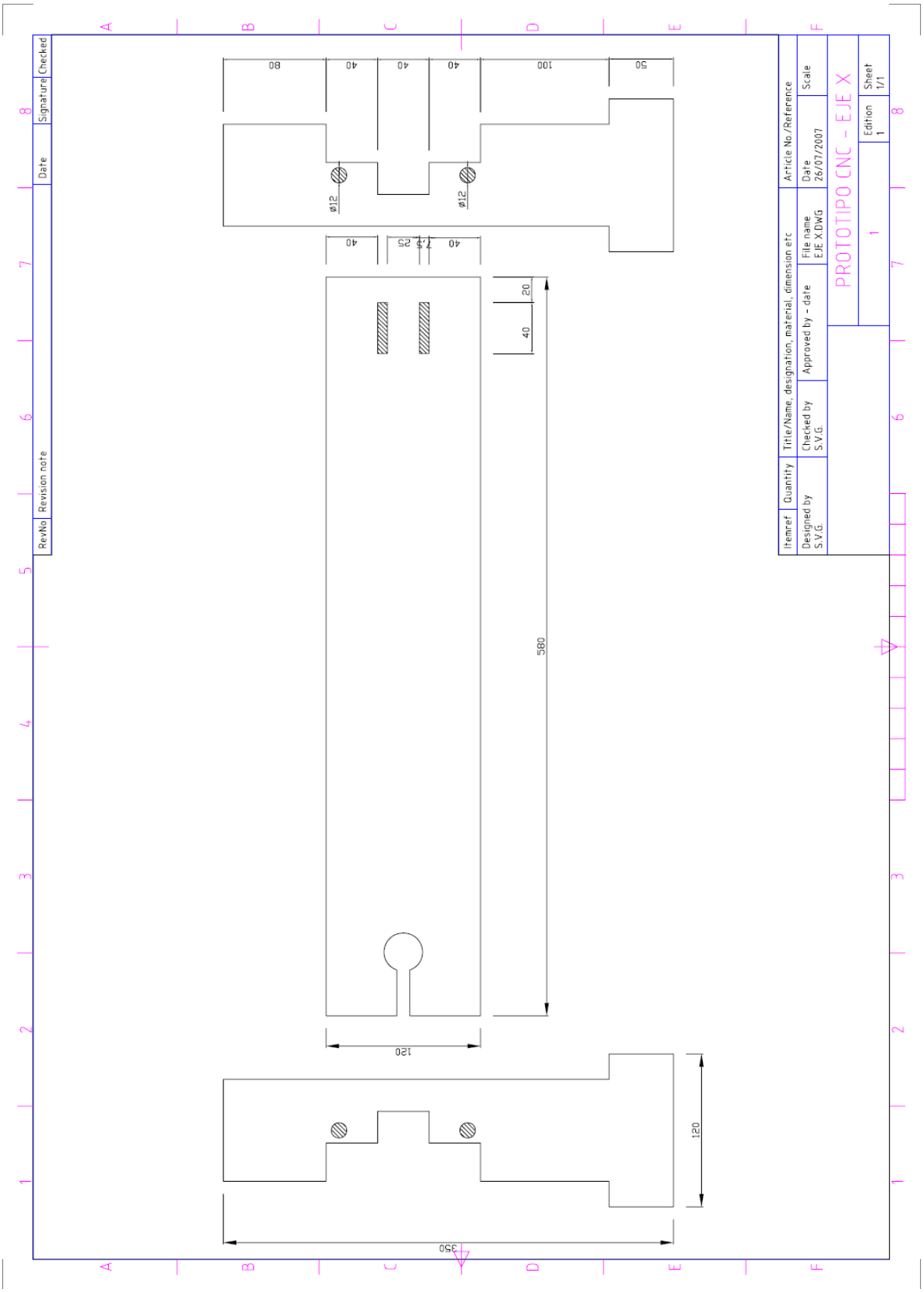
B.1 Piezas Eje X.

B.2 Piezas Eje Y.

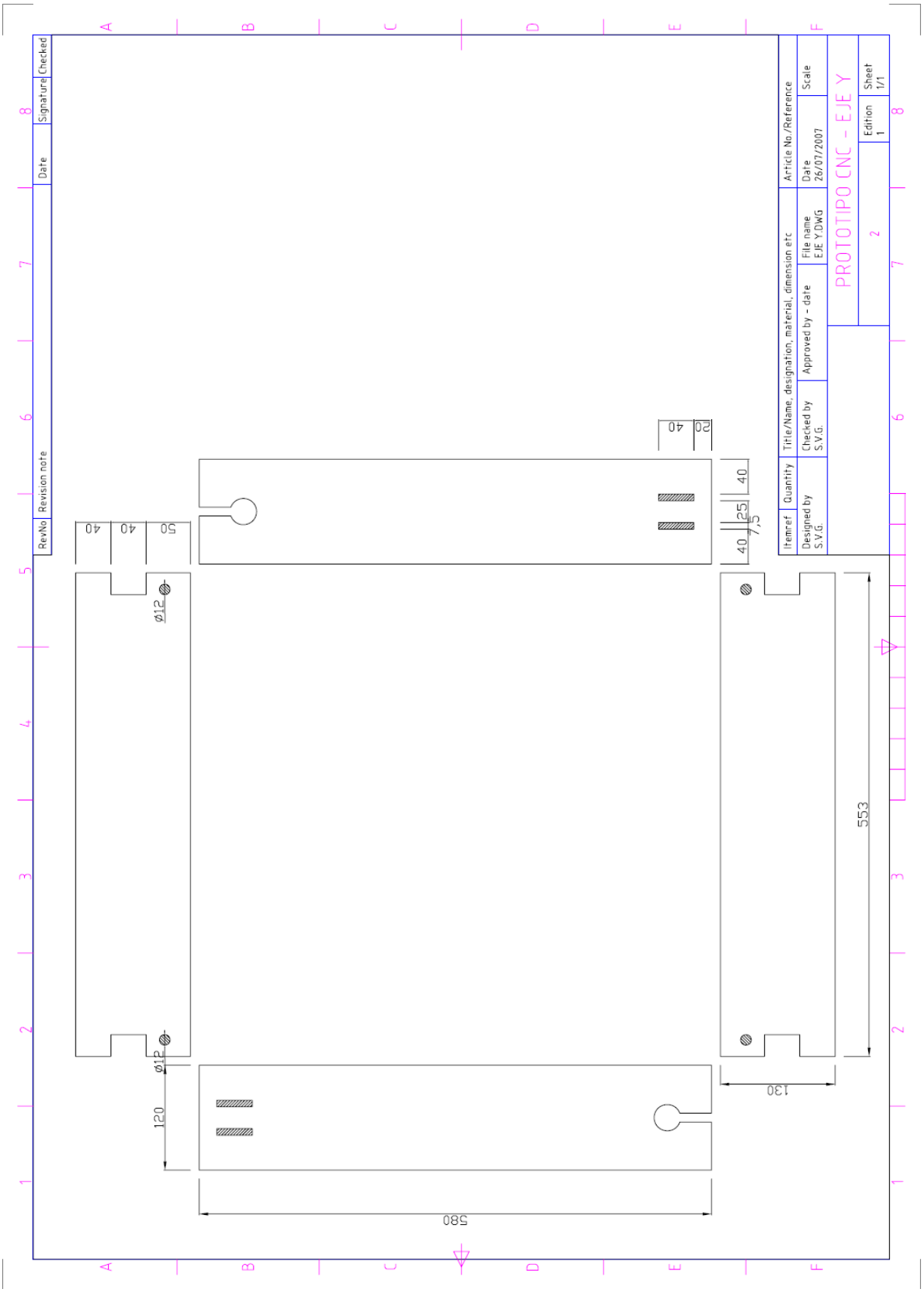
B.3 Piezas Eje Z.

B.4 Pieza Mecánica: Deslizador.

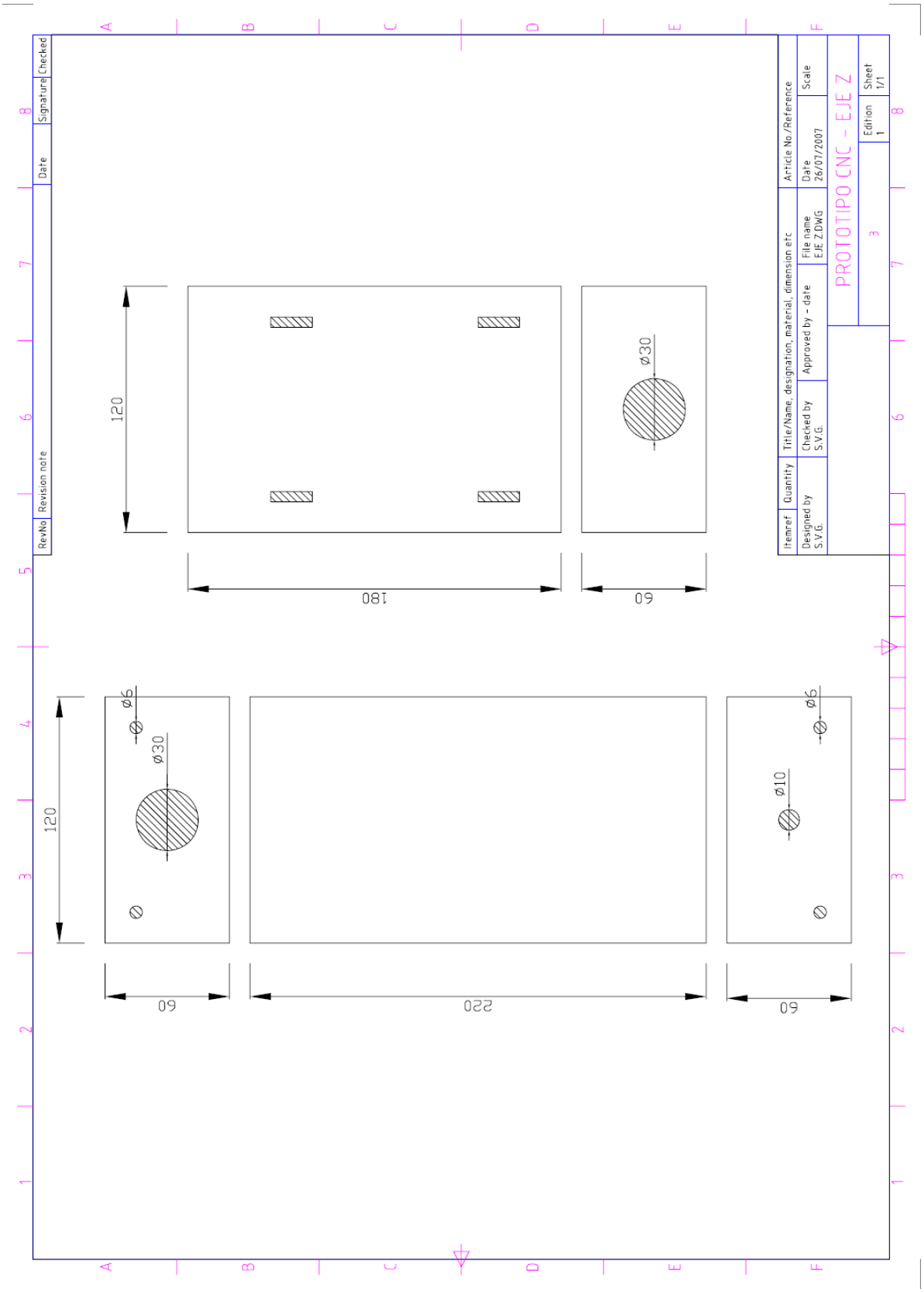
B.1. Piezas Eje X.



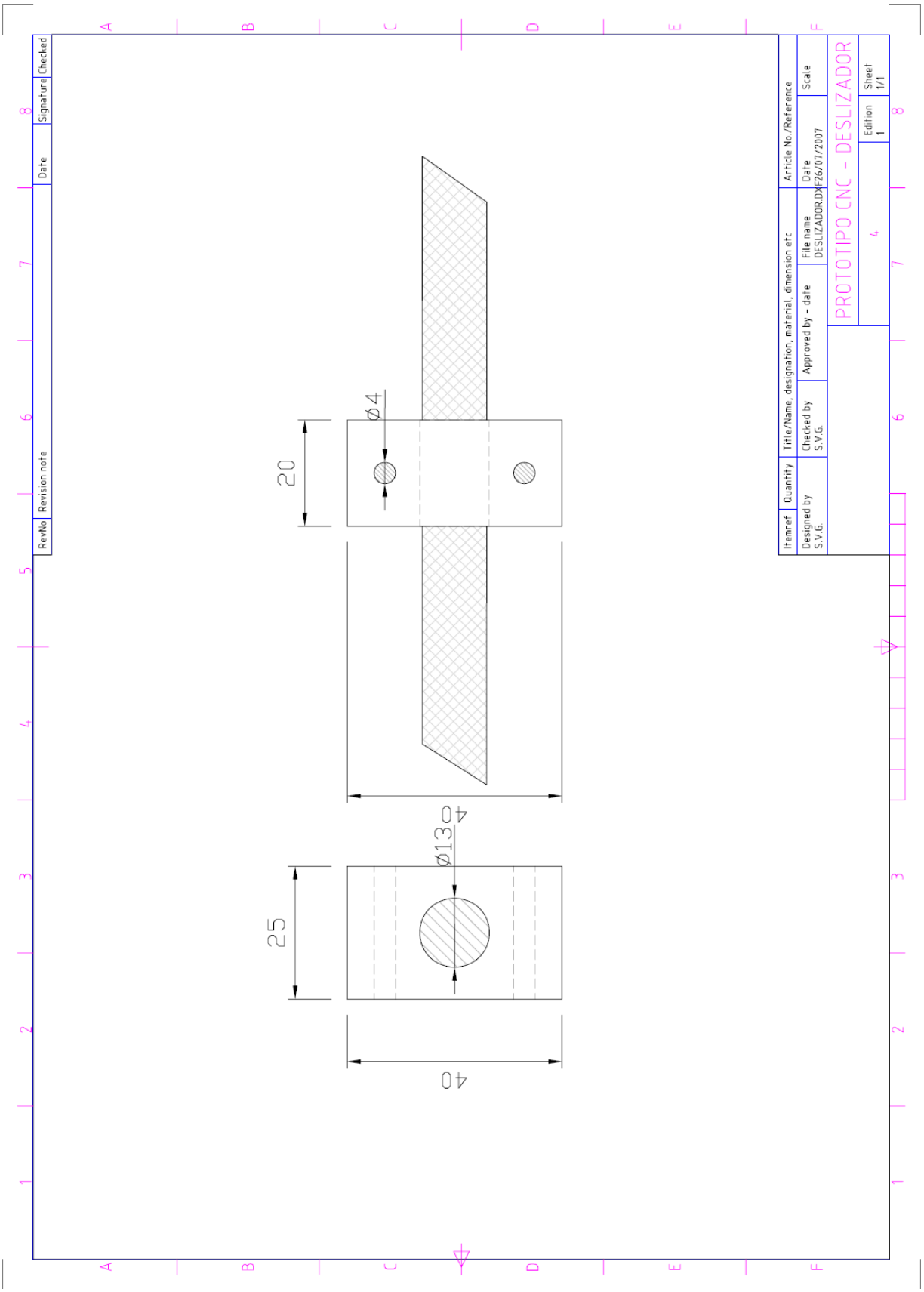
B.2. Piezas Eje Y.



B.3. Piezas Eje Z.



B.4. Pieza Mecánica: Deslizador.



Anexo C

Código Fuente DSP

Se entrega el código fuente para el DSP con el programa principal para la prueba del sistema completo, es decir, el "Controlador CNC" mediante la Plataforma de Desarrollo DSP TMS320F2810 v1.0.

C.1 Controlador CNC.

C.1. Controlador CNC.

```
//###########################################################################
//
// FILE:    main.c
// TITLE:   CONTROLADOR CNC
// DESCRIPTION:
//                    Codigo para controlar Fresadora CNC.
//             Funciones implementadas: CERO, MOVER, LINEA, CIRCULO.
//
//###########################################################################
//
// Original Author: Sebastián Valerio
//
//  Ver | dd mmm yyyy | Who  | Description of changes
// =====|=============|======|===============================================
// 1.00 |  7 Jun 2007 | S.V. | Primera version
//###########################################################################

#include "DSP281x_Device.h"                 // DSP281x Headerfile Include File
#include "DSP281x_Examples.h"               // DSP281x Examples Include File
#include "math.h"                           // Libreria con funciones matemáticas SIN() COS()

#define FACTOR_X      0.95                  // Factor de conversion de centimetros a pasos
#define FACTOR_Y      0.95                  // Factor de conversion de centimetros a pasos
#define FACTOR_Z      35                    // Factor de conversion de centimetros a pasos

Uint16 paso_x, paso_y, paso_z;              // cantidad de pasos
Uint16 pos_x, pos_y, pos_z;                 // posicion actual del cabezal
int dir_x, dir_y, dir_z;                    // direccion del motor
double div_x, div_y, div_z;                 // divisor de frecuencia
Uint16 x1, y1, z1, x2, y2, z2, r;           // parametros de instrucciones
Uint16 av_x, av_y, av_z;
int ocupado;                                // estado ocupado
int zero_x, zero_y, zero_z;

interrupt void eva_timer1_isr(void);


/********************** FUNCIONES VARIAS *********************/

void delay()
{
       int      i;
       for (i = 0; i < 10000; i++) {}
       }

// Mueve el cabezal a la posicion (dx2,dy2,dz2)

void mover(int dx2,int dy2,int dz2)
{
       int ax, ay, az;

       ocupado=1;

       if (pos_x==0) zero_x=0;
       if (pos_y==0) zero_y=0;

       ax=pos_x-dx2;
       ay=pos_y-dy2;
       az=pos_z-dz2;

       dir_x=(ax<0);
       dir_y=(ay<0);
       dir_z=(az<0);

       if(fmod(ax,2)>0) ax++;
       if(fmod(ay,2)>0) ay++;

       paso_x=abs(ax);
       paso_y=abs(ay);
       paso_z=abs(az);
```

```
	div_x=1;
	div_y=1;
	div_z=1;

	pos_x=pos_x-ax;
	pos_y=pos_y-ay;
	pos_z=pos_z-az;

	while(ocupado){
		delay();
		}

	zero_x=1;
	zero_y=1;
	}

// Dibuja una LINEA entre los puntos (dx1,dy1,dz1)- (dx2,dy2,dz2)

void dibLinea(int dx1,int dy1,int dz1,int dx2,int dy2,int dz2)
{
	int ax;
	int ay;
	int az;
	int i;

	mover(pos_x,pos_y,pos_z+1000);
	ocupado=1;

	while(ocupado){
		delay();
		}

	mover(dx1,dy1,pos_z);
	ocupado=1;

	while(ocupado){
		delay();
		}

	mover(pos_x,pos_y,dz1);
	ocupado=1;

	while(ocupado){
		delay();
		}

	ax=dx1-dx2;
	ay=dy1-dy2;
	az=dz1-dz2;

	dir_x=(ax<0);
	dir_y=(ay<0);
	dir_z=(az<0);

	paso_x=abs(ax);
	paso_y=abs(ay);
	paso_z=abs(az);

	if(paso_x>0 && paso_y>0)
	{
		div_x=paso_y;
		div_y=paso_x;

		// veo si son divisibles
		if(fmod(div_x,div_y)==0 || fmod(div_y,div_x)==0)
		{
			if(fmod(div_x,div_y)==0)
			{
				div_x=div_x/div_y;
				div_y=1;
				}
				else
				{
```

```
                        div_y=div_y/div_x;
                        div_x=1;
                        }
                }
                else
                {
                // busco la menor proporcion posible
                for(i=10;i>1;i--)
                {
                        while(fmod(div_x,i)==0 && fmod(div_y,i)==0)
                        {
                                div_x=div_x/i;
                                div_y=div_y/i;
                                }
                        }
                }
        }

        pos_x=pos_x-ax;
        pos_y=pos_y-ay;
        pos_z=pos_z-az;

        ocupado=1;

        while(ocupado){
                delay();
                }

        mover(pos_x,pos_y,pos_z+1000);
        ocupado=1;

        while(ocupado){
                delay();
                }

        scia_xmit(1); //envio ACK
        }

//funcion LINEA para CIRCULO

void dibLinea2(int dx1,int dy1,int dx2,int dy2)
{
        int ax, ay;
        int i;

        mover(dx1,dy1,pos_z);
        ocupado=1;

        while(ocupado){
                delay();
                }

        ax=dx1-dx2;
        ay=dy1-dy2;

        dir_x=(ax<0);
        dir_y=(ay<0);

        paso_x=abs(ax);
        paso_y=abs(ay);

        div_x=1;
        div_y=1;

        if(paso_x>0 && paso_y>0)
        {

                div_x=paso_y;
                div_y=paso_x;

                if(fmod(div_x,div_y)==0 || fmod(div_y,div_x)==0)
                {
                        if(fmod(div_x,div_y)==0)
```

```
                        {
                                div_x=div_x/div_y;
                                div_y=1;
                                }
                                else
                                {
                                div_y=div_y/div_x;
                                div_x=1;
                                }
                        }
                        else
                        {

                        for(i=10;i>1;i--)
                        {
                                while(fmod(div_x,i)==0 && fmod(div_y,i)==0)
                                {
                                        div_x=div_x/i;
                                        div_y=div_y/i;
                                        }
                                }
                        }
                }

        pos_x=pos_x-ax;
        pos_y=pos_y-ay;
        }


// Dibuja un CIRCULO con centro en (cx1,cy1,cz1) profundidad cz2 y radio r1

void dibCirculo(int cx1,int cy1,int cz1,int cz2,int r1)
{
        int ax1, ay1, ax2, ay2, az1;
        int i, pasos;
        double grados, radian;

        az1=cz1;
        radian=3.1415927;
        pasos=100;                      //Lineas para dibujar el circulo

        grados=2*radian/pasos;

        mover(pos_x,pos_y,cz1+1000);
        ocupado=1;

        while(ocupado){
                delay();
                }

        while(az1>=cz2){

                while(ocupado){
                        delay();
                        }

                mover(cx1+sin(0)*r1,cy1+cos(0)*r1,pos_z);
                ocupado=1;

                while(ocupado){
                        delay();
                        }

                mover(pos_x,pos_y,az1);
                ocupado=1;

                while(ocupado){
                        delay();
                        }

                for(i=0;i<pasos;i++)
                {
                        while(ocupado){
```

```
                        delay();
                        }

                    ax1=cx1+(int)(sin(grados*i)*r1);
                    ay1=cy1+(int)(cos(grados*i)*r1);

                    ax2=cx1+(int)(sin(grados*(i+1))*r1);
                    ay2=cy1+(int)(cos(grados*(i+1))*r1);

                    dibLinea2(ax1,ay1,ax2,ay2);
                    ocupado=1;

                    while(ocupado){
                        delay();
                        }
                    }
            az1=az1-25;
        }

        mover(pos_x,pos_y,cz1+1000);
        ocupado=1;

        while(ocupado){
            delay();
            }

        scia_xmit(1);  //envio ACK
        }

// Rutina principal

void main(void)
{
        unsigned int temp, data_cm, data_mm;

        // Inicializo variables
        temp=0;
        data_cm=0;
        data_mm=0;

        ocupado=0;

        pos_x=0;
        pos_y=0;
        pos_z=0;

        dir_x=0;
        dir_y=0;
        dir_z=0;

        div_x=1;
        div_y=1;
        div_z=1;

        av_x=1;
        av_y=1;
        av_z=1;

        zero_x=1;
        zero_y=1;
        zero_z=1;


        // Inicializo System Control: PLL, habilito Clock de Perifericos
        InitSysCtrl();

        EALLOW;

        GpioMuxRegs.GPAMUX.all=0x0000;                      // configurado como GPIO
        GpioMuxRegs.GPADIR.all=0xFFFF;                      // configurado como salida
        GpioMuxRegs.GPAQUAL.all=0x0000;

        GpioMuxRegs.GPBMUX.all=0x0000;
```

```
GpioMuxRegs.GPBDIR.all=0xFFFF;
GpioMuxRegs.GPBQUAL.all=0x0000;

GpioMuxRegs.GPDMUX.bit.T3CTRIP_PDPB_GPIOD5=0;       // configurado como GPIO
GpioMuxRegs.GPDMUX.bit.T4CTRIP_SOCB_GPIOD6=0;       // configurado como GPIO
GpioMuxRegs.GPDDIR.bit.GPIOD5=0;                    // configurado como entrada
GpioMuxRegs.GPDDIR.bit.GPIOD6=0;                    // configurado como entrada
GpioMuxRegs.GPDQUAL.all=0x0000;

GpioMuxRegs.GPFMUX.bit.CANTXA_GPIOF6=0;             // configurado como GPIO
GpioMuxRegs.GPFMUX.bit.CANRXA_GPIOF7=0;             // configurado como GPIO
GpioMuxRegs.GPFDIR.bit.GPIOF6=0;                    // configurado como entrada
GpioMuxRegs.GPFDIR.bit.GPIOF7=0;                    // configurado como entrada

GpioMuxRegs.GPFMUX.all=0x0030;              // configuro para trabajar como pin Sci A
                                            // Port F MUX - x000 0000 0011 0000
EDIS;

// Configuro Interrupciones
DINT;

InitPieCtrl();

IER = 0x0000;
IFR = 0x0000;

InitPieVectTable();

// Inicializo EV para usar con Timer
InitEv();

scia_fifo_init();            // Inicializo pila FIFO SCI
scia_init();                 // Inicializo SCI


// Inicio posicionamiento ie: busco el cero mecanico
paso_x=10000;
paso_y=10000;
paso_z=10000;

while(1)
{
if(scia_recibido())
       {
       temp=scia_rx();       //leo instruccion
       scia_xmit(temp);      //devuelvo instruccion (para retroalimentacion)

       if(temp==0x10)        // CERO
       {
              pos_x=0;
              pos_y=0;
              pos_z=0;
              }

       if(temp==0x11)        // MOVER
       {
              data_cm=scia_rx();
              data_mm=scia_rx();
              x1=(100*data_cm+data_mm)*FACTOR_X;

              data_cm=scia_rx();
              data_mm=scia_rx();
              y1=(100*data_cm+data_mm)*FACTOR_Y;

              data_cm=scia_rx();
              data_mm=scia_rx();
              z1=(100*data_cm+data_mm)*FACTOR_Z;

              mover(x1,y1,z1);
              }

       if(temp==0x12)        // LINEA
       {
```

```
                        data_cm=scia_rx();
                        data_mm=scia_rx();
                        x1=(100*data_cm+data_mm)*FACTOR_X;

                        data_cm=scia_rx();
                        data_mm=scia_rx();
                        y1=(100*data_cm+data_mm)*FACTOR_Y;

                        data_cm=scia_rx();
                        data_mm=scia_rx();
                        z1=(100*data_cm+data_mm)*FACTOR_Z;

                        data_cm=scia_rx();
                        data_mm=scia_rx();
                        x2=(100*data_cm+data_mm)*FACTOR_X;

                        data_cm=scia_rx();
                        data_mm=scia_rx();
                        y2=(100*data_cm+data_mm)*FACTOR_Y;

                        data_cm=scia_rx();
                        data_mm=scia_rx();
                        z2=(100*data_cm+data_mm)*FACTOR_Z;

                        dibLinea(x1,y1,z1,x2,y2,z2);
                        }

                if(temp==0x13)        // CIRCULO
                {
                        data_cm=scia_rx();
                        data_mm=scia_rx();
                        x1=(100*data_cm+data_mm)*FACTOR_X;

                        data_cm=scia_rx();
                        data_mm=scia_rx();
                        y1=(100*data_cm+data_mm)*FACTOR_Y;

                        data_cm=scia_rx();
                        data_mm=scia_rx();
                        z1=(100*data_cm+data_mm)*FACTOR_Z;

                        data_cm=scia_rx();
                        data_mm=scia_rx();
                        z2=(100*data_cm+data_mm)*FACTOR_Z;

                        data_cm=scia_rx();
                        data_mm=scia_rx();
                        r=(100*data_cm+data_mm)*FACTOR_X;

                        dibCirculo(x1,y1,z1,z2,r);
                        }

                temp=0;
                }
        }
}

interrupt void eva_timer1_isr(void)
{
   EvaTimer1InterruptCount++;
   // Enable more interrupts from this timer
   EvaRegs.EVAIMRA.bit.T1PINT = 1;
   // Limpio la interrupcion atendida
   EvaRegs.EVAIFRA.all = BIT7;
   PieCtrlRegs.PIEACK.all = PIEACK_GROUP2;


        // Inicio codigo

        // leo entradas (pulsador en cero)
        if(zero_x){
                if(!GpioDataRegs.GPDDAT.bit.GPIOD5){
                        zero_x=0;
```

```
                paso_x=0;
                pos_x=0;
                }
        else
                zero_x=1;
        }

if(zero_y){
        if(!GpioDataRegs.GPFDAT.bit.GPIOF6 && !GpioDataRegs.GPFDAT.bit.GPIOF7){
                zero_y=0;
                paso_y=0;
                pos_y=0;
                }
        else
                zero_y=1;
        }

if(!GpioDataRegs.GPDDAT.bit.GPIOD6){
        if(zero_z){
                zero_z=0;
                paso_z=0;
                pos_z=0;
                }
        }

// Movimiento Eje X
if(paso_x>0)
        {
        if(av_x>=div_x)
                {
                GpioDataRegs.GPADAT.bit.GPIOA2=1;           //enable Motor X
                GpioDataRegs.GPADAT.bit.GPIOA1=dir_x;       //direccion Motor X
                GpioDataRegs.GPADAT.bit.GPIOA0^=1;          //clock Motor X

                av_x=1;
            paso_x--;

                }
        else av_x++;
        }
else
{
        paso_x=0;
        av_x=1;
        GpioDataRegs.GPADAT.bit.GPIOA2=0;                   //disable Motor X
        }

// Movimiento Eje Y
if(paso_y>0)
{
        if(av_y>=div_y)
                {
                GpioDataRegs.GPBDAT.bit.GPIOB2=1;           //enable Motor Y 1
                GpioDataRegs.GPBDAT.bit.GPIOB5=1;           //enable Motor Y 2

                GpioDataRegs.GPBDAT.bit.GPIOB1=dir_y;       //direccion Motor Y 1
                GpioDataRegs.GPBDAT.bit.GPIOB4=dir_y;       //direccion Motor Y 2

                GpioDataRegs.GPBDAT.bit.GPIOB0^=1;          //clock Motor Y 1
                GpioDataRegs.GPBDAT.bit.GPIOB3^=1;          //clock Motor Y 2

                av_y=1;
                paso_y--;
                }
                else av_y++;
        }
else
        {
        paso_y=0;
        av_y=1;
        GpioDataRegs.GPBDAT.bit.GPIOB2=0;                   // disable Motor X 1
        GpioDataRegs.GPBDAT.bit.GPIOB5=0;                   // disable Motor X 2
        }
```

```
// Movimiento Eje Z
if(paso_z>0)
{
	if(av_z>=div_z)
		{
		GpioDataRegs.GPADAT.bit.GPIOA5=1;		//enable Motor Z
		GpioDataRegs.GPADAT.bit.GPIOA4=dir_z;		//direccion Motor Z
		GpioDataRegs.GPADAT.bit.GPIOA3^=1;		//clock Motor Z

		av_z=1;
		paso_z--;
		}
		else av_z++;

	}
else
	{
	paso_z=0;
	GpioDataRegs.GPADAT.bit.GPIOA5=0;			//disable Motor Z
	}

// Si hay motores activos
if(paso_x==0 && paso_y==0 && paso_z==0)
	ocupado=0;
	else
	ocupado=1;
}
```

Anexo D

Estructura CD de Archivos

En este anexo se detalla la estructura del CD adjunto en este trabajo. Este CD cuenta con documentos de la plataforma, utilidades y el código fuente del software desarrollado.

DXF/	
Docs/	Documentación sobre DXF
VB_Control_CNC/	Software desarrollado en Visual Basic
VB_DXF_CNC/	Software desarrollado en Visual Basic
Plataforma/	
Docs/	Documentación Plataforma de Desarrollo
Codigo/	Código fuente del programa cargado en DSP
PCB/	Esquemáticos y Layout de placas desarrolladas
Prototipo/	
Docs/	Documentación para Prototipo de Fresadora CNC
Planos/	Planos de piezas del prototipo en AutoCAD
Utilidades/	Programas y Controladores